

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

вычислительной техники

Кафедра

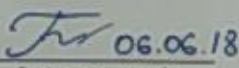
Вычислительная техника

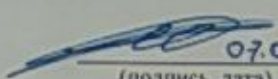
Направление подготовки 09.04.01 «Информатика и вычислительная техника»
Магистерская программа «Вычислительные машины, комплексы, комплексы,
системы и сети»

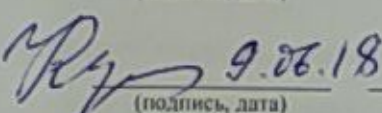
ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ
РАБОТА МАГИСТРА

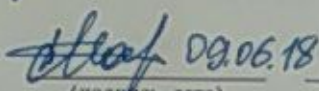
на тему

Информационная система предприятия

Студент  06.06.18 Дмитрий Сергеевич Гришин
(подпись, дата) (ФИО полностью)

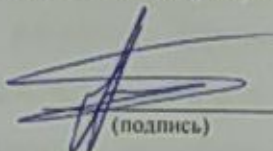
Научный руководитель  07.06.18 М.П. Синева
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр  9.06.18 А.В. Кучин
(подпись, дата) (фамилия, инициалы)

Рецензент к.т.н., доц. каф. ВИС  09.06.18 А.И. Мартынов
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 25.05.2018 г. № 38-09-12)

Заведующий кафедрой


(подпись)

Д.В. Пащенко

(фамилия, инициалы)

Работа защищена с отметкой лично (протокол заседания ГЭК от 14.06.18, № 10)

Секретарь ГЭК


(подпись)

Е.И. Гурин

(фамилия, инициалы)

Пенза, 2018

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

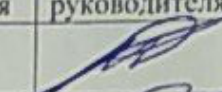
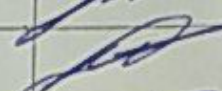
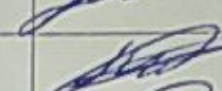
Кафедра

Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Д.В. Пашенко
« » 20__ г.

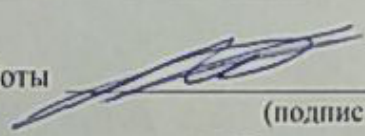
ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
МАГИСТРА

1. Студент гр. 16ВВм1 факультета ВТ Гришин Дмитрий Сергеевич.
2. Тема работы: Информационная система предприятия. Тема утверждена приказом ПГУ № 979 а/о от "3" октября 2016 г.
3. Руководитель работы: Синев М.П.
4. Задание на работу (назначение разработки, исходные данные и т.п.): Разработка системы оперативно контроля критических сетевых ресурсов. Исследование существующих систем сетевого мониторинга и обоснован выбор конкретных компонентов системы сетевого мониторинга.
5. Перечень подлежащих разработке вопросов: Аналитический обзор программных средств диагностики сетевых ресурсов. Проектирование системы сетевого мониторинга. Выбор инструментальных систем для разработки сетевого монитора. Программная реализация системы мониторинга. Отладка программного обеспечения.
6. Календарный график выполнения работы

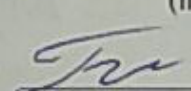
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Аналитический обзор программных средств диагностики сетевых ресурсов	25%	21.05.18 – 24.05.18	
2.	Проектирование системы сетевого мониторинга.	20%	25.05.18 – 27.05.18	
3.	Выбор инструментальных систем для разработки сетевого монитора.	20%	28.05.18 – 30.05.18	
4.	Программная реализация системы мониторинга.	15%	31.05.18 – 1.06.18	
5.	Отладка программного обеспечения.	20%	2.06.18 – 4.06.18	

Дата выдачи задания " ____ " _____ 20 ____

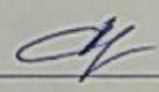
Руководитель магистерской работы


(подпись, дата)

Задание к исполнению принял студент


(подпись, дата)

Работу к защите допустить
Декан факультета

 А.Р. Рязанов
(подпись, дата)

13.06.2018

РЕФЕРАТ

СИСТЕМА КОНТРОЛЯ СЕТЕВЫХ РЕСУРСОВ, СИСТЕМА МОНИТОРИНГА СЕТИ, КОМПЬЮТЕРНЫЕ СЕТИ, АНАЛИЗАТОР СЕТЕВЫХ ПРОТОКОЛОВ

Магистерская диссертация содержит 84 листа, 10 рисунков, 1 приложение.

Объектом исследования являются компьютерные сети.

Предметом исследования является система контроля критических сетевых ресурсов.

Цель работы состояла в разработке системы оперативно контроля критических сетевых ресурсов. В работе проведено исследование существующих систем сетевого мониторинга и обоснован выбор конкретных компонентов системы сетевого мониторинга.

В результате работы над выпускной квалификационной работой магистра была создана функционирующая система сетевого мониторинга.

Было произведено тестирование системы мониторинга, в результате которого были получены корректные результаты работы программы.

СОДЕРЖАНИ

РЕФЕРАТ.....	3
СОДЕРЖАНИЕ.....	4
ВВЕДЕНИЕ.....	5
1 СИСТЕМЫ СЕТЕВОГО МОНИТОРИНГА.....	7
1.1 Программные средства диагностики.....	15
1.1.1 Анализаторы протоколов.....	16
1.1.2 Протоколы мониторинга.....	21
Протокол SNMP.....	21
1.2 Популярные системы управления сетями.....	31
1.3 Популярные системы мониторинга сетей.....	34
1.4 Выбор системы мониторинга.....	40
2 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ.....	43
2.1 Выбор инструментария для разработки системы.....	43
2.2 Установка необходимых компонентов.....	50
2.2.1 Установка драйвера QMYSQL.....	50
2.2.2 Установка библиотеки QXMPP.....	54
3 Разработка системы.....	55
2.3.1 Алгоритм работы системы.....	55
2.3.1.1 Алгоритм работы системы мониторинга.....	55
2.3.1.2 Алгоритм работы оповещения.....	56
2.3.2. Структура БД.....	58
2.4 Диаграмма классов.....	64
2.5. Отладка программного обеспечения.....	67
ЗАКЛЮЧЕНИЕ.....	73
Приложение А Листинги программы.....	74
ИСХОДНЫЙ КОД ПРОГРАММЫ.....	75

ВВЕДЕНИЕ

Несмотря на множество приемов и инструментов обнаружения и устранения неполадок в компьютерных сетях, «почва под ногами» сетевых администраторов все еще остается достаточно зыбкой. Компьютерные сети все чаще включают волоконно-оптические и беспроводные компоненты, наличие которых делает бессмысленным применение традиционных технологий и инструментов, предназначенных для обычных медных кабелей. Вдобавок к нему при скоростях свыше 100 Мбит/с традиционные подходы к диагностике зачастую перестают работать, даже если средой передачи является обычный медный кабель. Однако, возможно, наиболее серьезным изменением в компьютерных сетевых технологиях, с которым пришлось столкнуться администраторам, стал неизбежный переход от сетей Ethernet с разделяемой средой передачи к коммутируемым сетям, в которых в качестве коммутируемых сегментов часто выступают отдельные серверы или рабочие станции.

Правда, по мере осуществления технологических преобразований некоторые старые проблемы решились сами собой. Коаксиальный кабель, в котором выявить электротехнические неисправности всегда было труднее, чем в случае витой пары, становится редкостью в корпоративных средах. Сети Token Ring, главной проблемой которых была их несхожесть с Ethernet (а вовсе не слабость в техническом отношении), постепенно заменяются коммутируемыми сетями Ethernet. Порождающие многочисленные сообщения об ошибках протоколов сетевого уровня протоколы, такие, как SNA, DECnet и AppleTalk, замещаются протоколом IP. Сам же стек протоколов IP стал более стабильным и простым для поддержки, что

доказывают миллионы клиентов и миллиарды страниц Web в Internet. Даже закоренелым противникам Microsoft приходится признать, что подключение нового клиента Windows к Internet существенно проще и надежнее установки применявшихся ранее стеков TCP/IP сторонних поставщиков и отдельного программного обеспечения коммутируемого доступа.

Как бы многочисленные современные технологии ни затрудняли выявление неполадок и управление производительностью сетей, ситуация могла бы оказаться еще тяжелее, если бы технология ATM получила широкое распространение на уровне ПК. Свою положительную роль сыграло и то, что в конце 90-х, не успев получить признание, были отвергнуты и некоторые другие высокоскоростные технологии обмена данными, включая Token Ring с пропускной способностью 100 Мбит/с, 100VG-AnyLAN и усовершенствованные сети ARCnet. Наконец, в США был отклонен очень сложный стек протоколов OSI (который, правда, узаконен рядом правительств европейских стран).

Рассмотрим некоторые актуальные проблемы, возникающие у сетевых администраторов предприятий.

1 СИСТЕМЫ СЕТЕВОГО МОНИТОРИНГА

Иерархическая топология компьютерных сетей с магистральными каналами Gigabit Ethernet и выделенными портами коммутаторов на 10 или даже 100 Мбит/с для отдельных клиентских систем, позволила увеличить максимальную пропускную способность, потенциально доступную пользователям, как минимум в 10—20 раз. Конечно, в большинстве компьютерных сетей существуют узкие места на уровне серверов или маршрутизаторов доступа, поскольку приходящаяся на отдельного пользователя пропускная способность существенно меньше 10 Мбит/с. В связи с этим замена порта концентратора с пропускной способностью 10 Мбит/с на выделенный порт коммутатора на 100 Мбит/с для конечного узла отнюдь не всегда приводит к значительному увеличению скорости. Однако если учесть, что стоимость коммутаторов в последнее время снизилась, а на большинстве предприятий проложен кабель Категории 5,



Рисунок 1.1 - Традиционная локальная сеть с разделяемой средой передачи и анализатором протоколов

поддерживающий технологию Ethernet на 100 Мбит/с, и установлены сетевые карты, способные работать на скорости 100 Мбит/с сразу после перезагрузки системы, то становится ясно, почему так нелегко сопротивляться искушению модернизации. В традиционной локальной сети с разделяемой средой передачи анализатор протоколов или монитор может исследовать весь трафик данного сегмента сети.

Хотя преимущество коммутируемой сети в производительности иногда почти не заметно, распространение коммутируемых архитектур имело катастрофические последствия для традиционных средств диагностики. В сильно сегментированной сети анализаторы протоколов способны видеть только одноадресный трафик на отдельном порту коммутатора, в отличие от сети прежней топологии, где они могли тщательно исследовать любой пакет в домене коллизий. В таких условиях традиционные инструменты мониторинга не могут собрать статистику по всем «диалогам», потому что каждая «переговаривающаяся» пара оконечных точек пользуется, в сущности, своей собственной сетью.



Рисунок 1.2 – Коммутируемая сеть

В коммутируемой сети анализатор протоколов в одной точке может «видеть» только единственный сегмент, если коммутатор не способен зеркально отображать несколько портов одновременно.

Для сохранения контроля над сильно сегментированными сетями производители коммутаторов предлагают разнообразные средства для восстановления полной «видимости» сети, однако на этом пути остается немало трудностей. В поставляемых сейчас коммутаторах обычно поддерживается «зеркальное отображение» портов, когда трафик одного из них дублируется на ранее незадействованный порт, к которому подключается монитор или анализатор.

Однако «зеркальное отображение» обладает рядом недостатков. Во-первых, в каждый момент времени виден только один порт, поэтому выявить неполадки, затрагивающие сразу несколько портов, очень непросто. Во-вторых, зеркальное отражение может привести к снижению производительности коммутатора. В-третьих, на зеркальном порту обычно не воспроизводятся сбои физического уровня, а иногда даже теряются обозначения виртуальных локальных сетей. Наконец, во многих случаях не могут в полной мере зеркально отображаться полнодуплексные каналы Ethernet.

Частичным решением при анализе агрегированных параметров трафика является использование возможностей мониторинга агентов mini-RMON, тем более что они встроены в каждый порт большинства коммутаторов Ethernet. Хотя агенты mini-RMON не поддерживают группу объектов Capture из спецификации RMON II, обеспечивающих полнофункциональный анализ протоколов, они тем не менее позволяют оценить уровень использования ресурсов, количество ошибок и объем многоадресной рассылки.

Некоторые недостатки технологии зеркального отображения портов могут быть преодолены установкой «пассивных ответвителей», производимых, например, компанией Shomiti. Эти устройства представляют собой заранее устанавливаемые Y-коннекторы и позволяют отслеживать с помощью анализаторов протокола или другого устройства не регенерированный, а реальный сигнал.

Следующей актуальной проблемой, является проблема с особенностями оптики. Администраторы компьютерных сетей обычно используют специализированное оборудование диагностики оптических сетей только для решения проблем с оптическими кабелями. Обычное стандартное программное обеспечение управления устройствами на базе SNMP или интерфейса командной строки способно выявить проблемы на коммутаторах и маршрутизаторах с оптическими интерфейсами. И только немногие сетевые администраторы сталкиваются с необходимостью проводить диагностику устройств SONET.

Что касается волоконно-оптических кабелей, то причин для возникновения возможных неисправностей в них существенно меньше, чем в случае медного кабеля. Оптические сигналы не вызывают перекрестных помех, появляющихся от того, что сигнал одного проводника индуцирует сигнал на другом — этот фактор наиболее усложняет диагностическое оборудование для медного кабеля. Оптические кабели невосприимчивы к электромагнитным шумам и индуцированным сигналам, поэтому их не требуется располагать подальше от электромоторов лифтов и ламп дневного света, т. е. из сценария диагностики все эти переменные можно исключить.

Сила сигнала, или оптическая мощность, в данной точке на самом деле является единственной переменной, которую требуется измерить при поиске неисправностей в оптических сетях. Если же можно определить потери сигнала на всем протяжении оптического канала, то можно будет идентифицировать практически любую проблему. Недорогие дополнительные модули для тестеров медного кабеля позволяют проводить оптические измерения.

Предприятиям, развернувшим крупную оптическую инфраструктуру и самостоятельно ее обслуживающим, может понадобиться приобрести оптический временный рефлектометр (Optical Time Domain Reflectometer, OTDR), выполняющего те же функции для оптического волокна, что и рефлектометр для медного кабеля (Time Domain Reflectometer, TDR). Прибор действует подобно радару: он посылает импульсные сигналы по кабелю и анализирует их отражения, на основании которых он выявляет повреждения в проводнике или какую-либо другую аномалию, и затем сообщает эксперту, в каком месте кабеля следует искать источник проблемы.

Хотя различные поставщики кабельных соединителей и разъемов упростили процессы терминирования и разветвления оптического волокна, для этого по-прежнему требуется некоторый уровень специальных навыков, и при разумной политике предприятие с развитой оптической инфраструктурой вынуждено будет обучать своих сотрудников. Как бы хорошо ни была проложена кабельная сеть, всегда существует возможность физического повреждения кабеля в результате какого-либо неожиданного происшествия.

При диагностике беспроводных локальных сетей стандарта 802.11b также могут возникнуть проблемы. Сама по себе диагностика, столь же проста, как и в случае сетей Ethernet на базе концентраторов, так как беспроводная среда передачи информации разделяется между всеми обладателями клиентских радиоустройств. Компания Sniffer Technologies первой предложила решение для анализа протоколов таких сетей с пропускной способностью до 11 Мбит/с, и впоследствии большинство лидирующих поставщиков анализаторов представили аналогичные системы.

В отличие от концентратора Ethernet с проводными соединениями, качество беспроводных клиентских соединений далеко от стабильного. Микроволновые радиосигналы, используемые во всех вариантах локальной передачи, слабы и порой непредсказуемы. Даже небольшие изменения положения антенны могут серьезно сказаться на качестве соединений. Точки доступа беспроводной локальной сети снабжаются консолью управления устройствами, и это часто более действенный метод диагностики, чем посещение клиентов беспроводной сети и наблюдение за пропускной способностью и условиями возникновения ошибок с помощью портативного анализатора.

Хотя проблемы синхронизации данных и установки устройств, возникающие у пользователей персональных цифровых секретарей (PDA), более естественно соответствуют задачам группы технической поддержки, а не обязанностям сетевого администратора, нетрудно предвидеть, что в недалеком будущем многие такие устройства превратятся из отдельных вспомогательных средств, дополняющих ПК, в полноправных сетевых клиентов.

Как правило, операторы корпоративных беспроводных сетей будут (или должны) препятствовать развертыванию чрезмерно открытых систем, в которых любой пользователь, находящийся в зоне действия сети и обладающий совместимой интерфейсной картой, получает доступ к каждому информационному кадру системы. Протокол безопасности беспроводных сетей WEP (Wired Equivalent Privacy) обеспечивает аутентификацию пользователей, гарантию целостности и шифрование данных, однако, как это обычно случается, совершенная система безопасности осложняет анализ причин сетевых неполадок. В защищенных сетях с поддержкой WEP специалисты по диагностике должны знать ключи или пароли, защищающие информационные ресурсы и контролирующие доступ в систему. При доступе в режиме приема всех пакетов анализатор протоколов сможет видеть все заголовки кадров, но содержащаяся в них информация без наличия ключей будет бессмысленной.

При диагностировании туннелированных каналов, которые многие производители называют виртуальными частными сетями с удаленным доступом, возникающие проблемы аналогичны имеющим место при анализе беспроводных сетей с шифрованием. Если трафик не проходит через туннелированный канал, то причину неисправности определить нелегко. Это может быть ошибка аутентификации, поломка на одной из конечных точек или затор в общедоступной зоне Internet. Попытка использования анализатора протоколов для выявления высокоуровневых ошибок в туннелированном трафике будет пустой тратой сил, потому что содержание данных, а также заголовки прикладного, транспортного и сетевого уровней зашифрованы. Вообще, меры, принимаемые в целях повышения уровня безопасности корпоративных сетей, обычно затрудняют выявление неисправностей и проблем производительности. Межсетевые экраны, прокси-серверы и системы выявления вторжений могут дополнительно осложнить локализацию неполадок.

Таким образом, проблема диагностики компьютерных сетей является актуальной и в конечном счете, диагностирование неисправностей является задачей управления. Для большинства критически важных корпоративных систем, проведение продолжительных восстановительных работ не допустимо, поэтому единственным решением будет использование резервных устройств и процессов, способных взять на себя необходимые функции немедленно после возникновения сбоев. На некоторых предприятиях сети всегда имеют дополнительный резервный компонент на случай сбоя основного, т. е. $n + 2$ компонентов, где n — количество основных компонентов, необходимое для обеспечения приемлемой производительности. Если среднее время восстановления (Mean Time To Repair, MTTR) достаточно велико, то может понадобиться еще большая избыточность. Дело в том, что время устранения неисправности предсказать нелегко, а значительные затраты в течение непредсказуемого периода восстановления являются признаком плохого управления.

Для менее важных систем резервирование может оказаться экономически неоправданным, и в этом случае будет целесообразно вкладывать средства в наиболее эффективные инструменты (и в обучение персонала), чтобы максимально ускорить процесс диагностики и устранения неисправностей на предприятии. Кроме того, поддержку определенных систем можно доверить сторонним специалистам, либо привлекая их на предприятие по контракту, либо пользуясь возможностями внешних центров обработки данных, либо обращаясь к провайдерам услуг по сопровождению приложений (Application Service Providers, ASP) или провайдерам услуг управления. Помимо затрат наиболее значительным фактором, влияющим на решение об обращении к услугам сторонних организаций, можно считать уровень компетентности собственного персонала. Сетевые администраторы должны решить, не является ли некоторая конкретная функция настолько тесно связанной со специфическими задачами предприятия, что от стороннего специалиста нельзя будет ожидать более качественного выполнения работы, чем это будет сделано силами служащих компании.

Почти сразу после того, как были развернуты первые корпоративные сети, надежность которых оставляла желать лучшего, производители и разработчики выдвинули концепцию «самовосстанавливающихся сетей». Современные сети, безусловно, надежнее, чем они были в 90-х гг., но не потому, что неполадки стали самоустраниваться. Ликвидация сбоев программного обеспечения и аппаратных средств современных сетей все еще требуют вмешательства человека, и в ближайшей перспективе в таком положении дел не предвидится никаких принципиальных изменений. Методы и инструменты диагностики вполне соответствуют современной практике и технологиям, но они еще не достигли такого уровня, который позволил бы значительно сэкономить время сетевых администраторов в их борьбе с неполадками сетей и дефицитом производительности.

1.1 Программные средства диагностики

Среди программных средств диагностики компьютерных сетей, можно выделить специальные системы управления сетью (Network Management Systems) - централизованные программные системы, которые собирают данные о состоянии узлов и коммуникационных устройств сети, а также данные о трафике, циркулирующем в сети. Эти системы не только осуществляют мониторинг и анализ сети, но и выполняют в автоматическом или полуавтоматическом режиме действия по управлению сетью - включение и отключение портов устройств, изменение параметров мостов, адресных таблиц мостов, коммутаторов и маршрутизаторов и т.п. Примерами систем управления могут служить популярные системы HP OpenView, SunNetManager, IBMNetView.

Средства управления системой (System Management) выполняют функции, аналогичные функциям систем управления, но по отношению к коммуникационному оборудованию. Вместе с тем, некоторые функции этих двух видов систем управления могут дублироваться, например, средства управления системой могут выполнять простейший анализ сетевого трафика.

Экспертные системы. Этот вид систем аккумулирует человеческие знания о выявлении причин аномальной работы сетей и возможных способах приведения сети в работоспособное состояние. Экспертные системы часто реализуются в виде отдельных подсистем различных средств мониторинга и анализа сетей: систем управления сетями, анализаторов протоколов, сетевых анализаторов. Простейшим вариантом экспертной системы является контекстно-зависимая help-система. Более сложные экспертные системы представляют собой так называемые базы знаний, обладающие элементами искусственного интеллекта. Примером такой системы является экспертная система, встроенная в систему управления Spectrum компании Cabletron.

1.1.1 Анализаторы протоколов

В ходе проектирования новой или модернизации старой сети часто возникает необходимость в количественном измерении некоторых характеристик сети таких, например, как интенсивности потоков данных по сетевым линиям связи, задержки, возникающие на различных этапах обработки пакетов, времена реакции на запросы того или иного вида, частота возникновения определенных событий и других характеристик.

Для этих целей могут быть использованы разные средства и прежде всего - средства мониторинга в системах управления сетью, которые уже обсуждались ранее. Некоторые измерения на сети могут быть выполнены и встроенными в операционную систему программными измерителями, примером тому служит компонента ОС Windows Performance Monitor. Даже кабельные тестеры в их современном исполнении способны вести захват пакетов и анализ их содержимого.

Но наиболее совершенным средством исследования сети является анализатор протоколов. Процесс анализа протоколов включает захват циркулирующих в сети пакетов, реализующих тот или иной сетевой протокол, и изучение содержимого этих пакетов. Основываясь на результатах анализа, можно осуществлять обоснованное и взвешенное изменение каких-либо компонент сети, оптимизацию ее производительности, поиск и устранение неполадок. Очевидно, что для того, чтобы можно было сделать какие-либо выводы о влиянии некоторого изменения на сеть, необходимо выполнить анализ протоколов и до, и после внесения изменения.

Анализатор протоколов представляет собой либо самостоятельное специализированное устройство, либо персональный компьютер, обычно переносной, класса Notebook, оснащенный специальной сетевой картой и соответствующим программным обеспечением. Применяемые сетевая карта и программное обеспечение должны соответствовать топологии сети (кольцо, шина, звезда). Анализатор подключается к сети точно также, как и обычный узел. Отличие состоит в том, что анализатор может принимать все пакеты данных, передаваемые по сети, в то время как обычная станция - только адресованные ей. Программное обеспечение анализатора состоит из ядра, поддерживающего работу сетевого адаптера и декодирующего получаемые данные, и дополнительного программного кода, зависящего от типа топологии исследуемой сети. Кроме того, поставляется ряд процедур декодирования, ориентированных на определенный протокол, например, IPX. В состав некоторых анализаторов может входить также экспертная система, которая может выдавать пользователю рекомендации о том, какие эксперименты следует проводить в данной ситуации, что могут означать те или иные результаты измерений, как устранить некоторые виды неисправности сети.

Несмотря на относительное многообразие анализаторов протоколов, представленных на рынке, можно назвать некоторые черты, в той или иной мере присущие всем им:

Пользовательский интерфейс. Большинство анализаторов имеют развитый дружественный интерфейс, базирующийся, как правило, на Windows или Motif. Этот интерфейс позволяет пользователю: выводить результаты анализа интенсивности трафика; получать мгновенную и усредненную статистическую оценку производительности сети; задавать определенные события и критические ситуации для отслеживания их возникновения; производить декодирование протоколов разного уровня и представлять в понятной форме содержимое пакетов.

Буфер захвата. Буферы различных анализаторов отличаются по объему. Буфер может располагаться на устанавливаемой сетевой карте, либо для него может быть отведено место в оперативной памяти одного из компьютеров сети. Если буфер расположен на сетевой карте, то управление им осуществляется аппаратно, и за счет этого скорость ввода повышается. Однако это приводит к удорожанию анализатора. В случае недостаточной производительности процедуры захвата, часть информации будет теряться, и анализ будет невозможен. Размер буфера определяет возможности анализа по более или менее представительным выборкам захватываемых данных. Но каким бы большим ни был буфер захвата, рано или поздно он заполнится. В этом случае либо прекращается захват, либо заполнение начинается с начала буфера.

Фильтры. Фильтры позволяют управлять процессом захвата данных, и, тем самым, позволяют экономить пространство буфера. В зависимости от значения определенных полей пакета, заданных в виде условия фильтрации, пакет либо игнорируется, либо записывается в буфер захвата. Использование фильтров значительно ускоряет и упрощает анализ, так как исключает просмотр ненужных в данный момент пакетов.

Переключатели - это задаваемые оператором некоторые условия начала и прекращения процесса захвата данных из сети. Такими условиями могут быть выполнение ручных команд запуска и остановки процесса захвата, время суток, продолжительность процесса захвата, появление определенных значений в кадрах данных. Переключатели могут использоваться совместно с фильтрами, позволяя более детально и тонко проводить анализ, а также продуктивнее использовать ограниченный объем буфера захвата.

Поиск. Некоторые анализаторы протоколов позволяют автоматизировать просмотр информации, находящейся в буфере, и находить в ней данные по заданным критериям. В то время, как фильтры проверяют входной поток на предмет соответствия условиям фильтрации, функции поиска применяются к уже накопленным в буфере данным.

Методология проведения анализа может быть представлена в виде следующих шести этапов:

1. Захват данных.
2. Просмотр захваченных данных.
3. Анализ данных.
4. Поиск ошибок. (Большинство анализаторов облегчают эту работу, определяя типы ошибок и идентифицируя станцию, от которой пришел пакет с ошибкой.)
5. Исследование производительности. Рассчитывается коэффициент использования пропускной способности сети или среднее время реакции на запрос.
6. Подробное исследование отдельных участков сети. Содержание этого этапа конкретизируется по мере того, как проводится анализ.

Обычно процесс анализа протоколов занимает относительно немного времени - 1-2 рабочих дня.

Большинство современных анализаторов позволяют анализировать сразу несколько протоколов глобальных сетей, таких, как X.25, PPP, SLIP, SDLC/SNA, frame relay, SMDS, ISDN, протоколы мостов/маршрутизаторов (3Com, Cisco, Bay Networks и другие). Такие анализаторы позволяют измерять различные параметры протоколов, анализировать трафик в сети, преобразование между протоколами локальных и глобальных сетей, задержку на маршрутизаторах при этих преобразованиях и т. п. Более совершенные приборы предусматривают возможность моделирования и декодирования протоколов глобальных сетей, 'стрессового' тестирования, измерения максимальной пропускной способности, тестирования качества предоставляемых услуг. В целях универсальности почти все анализаторы протоколов глобальных сетей реализуют функции тестирования ЛВС и всех основных интерфейсов. Некоторые приборы способны осуществлять анализ протоколов телефонии. А самые современные модели могут декодировать и представлять в удобном варианте все семь уровней OSI. Появление ATM привело к тому, что производители стали снабжать свои анализаторы средствами тестирования этих сетей. Такие приборы могут проводить полное тестирование сетей ATM уровня E-1/E-3 с поддержкой мониторинга и моделирования. Очень важное значение имеет набор сервисных функций анализатора. Некоторые из них, например возможность удаленного управления прибором, просто незаменимы.

Таким образом, современные анализаторы протоколов WAN/LAN/DTM позволяют обнаружить ошибки в конфигурации маршрутизаторов и мостов; установить тип трафика, пересылаемого по глобальной сети; определить используемый диапазон скоростей, оптимизировать соотношение между пропускной способностью и количеством каналов; локализовать источник неправильного трафика; выполнить тестирование последовательных интерфейсов и полное тестирование ATM; осуществить полный мониторинг и декодирование основных протоколов по любому каналу; анализировать статистику в реальном времени, включая анализ трафика локальных сетей через глобальные сети.

1.1.2 Протоколы мониторинга

Протокол SNMP

SNMP (англ. Simple Network Management Protocol — простой протокол управления сетью) — это протокол управления сетями связи на основе архитектуры TCP/IP.

На основе концепции TMN в 1980—1990 гг. различными органами стандартизации был выработан ряд протоколов управления сетями передачи данных с различным спектром реализации функций TMN. К одному из типов таких протоколов управления относится SNMP. Протокол SNMP был разработан с целью проверки функционирования сетевых маршрутизаторов и мостов. Впоследствии сфера действия протокола охватила и другие сетевые устройства, такие как хабы, шлюзы, терминальные сервера, LAN Manager сервера, машины под управлением Windows NT и т.д. Кроме того, протокол допускает возможность внесения изменений в функционирование указанных устройств.

Эта технология, призвана обеспечить управление и контроль за устройствами и приложениями в сети связи путём обмена управляющей информацией между агентами, располагающимися на сетевых устройствах, и менеджерами, расположенными на станциях управления. SNMP определяет сеть как совокупность сетевых управляющих станций и элементов сети (главные машины, шлюзы и маршрутизаторы, терминальные серверы), которые совместно обеспечивают административные связи между сетевыми управляющими станциями и сетевыми агентами.

При использовании SNMP присутствуют управляемые и управляющие системы. В состав управляемой системы входит компонент, называемый агентом, который отправляет отчёты управляющей системе. По существу SNMP агенты передают управленческую информацию на управляющие системы как переменные (такие как «свободная память», «имя системы», «количество работающих процессов»).

Агент в протоколе SNMP - это обрабатывающий элемент, который обеспечивает менеджерам, размещенным на управляющих станциях сети, доступ к значениям переменных MIB, и тем самым дает им возможность реализовывать функции по управлению и наблюдению за устройством.

Программный агент - резидентная программа, выполняющая функции управления, а также собирающая статистику для передачу ее в информационную базу сетевого устройства.

Аппаратный агент - встроенная аппаратура (с процессором и памятью), в которой хранятся программные агенты.

Переменные, доступные через SNMP, организованы в иерархии. Эти иерархии и другие метаданные (такие, как тип и описание переменной) описываются Базами Управляющей Информации (Management Information Bases (MIBs)).

На сегодня существует несколько стандартов на базы данных управляющей информации. Основными являются стандарты MIB-I и MIB-II, а также версия базы данных для удаленного управления RMON MIB. Кроме этого, существуют стандарты для специальных MIB устройств конкретного типа (например, MIB для концентраторов или MIB для модемов), а также частные MIB конкретных фирм-производителей оборудования.

Первоначальная спецификация MIB-I определяла только операции чтения значений переменных. Операции изменения или установки значений объекта являются частью спецификаций MIB-II.

Версия MIB-I (RFC 1156) определяет до 114 объектов, которые подразделяются на 8 групп:

- System - общие данные об устройстве (например, идентификатор поставщика, время последней инициализации системы).
- Interfaces - описываются параметры сетевых интерфейсов устройства (например, их количество, типы, скорости обмена, максимальный размер пакета).
- AddressTranslationTable - описывается соответствие между сетевыми и физическими адресами (например, по протоколу ARP).
- InternetProtocol - данные, относящиеся к протоколу IP (адреса IP-шлюзов, хостов, статистика об IP-пакетах).

- ICMP - данные, относящиеся к протоколу обмена управляющими сообщениями ICMP.
- TCP - данные, относящиеся к протоколу TCP (например, о TCP-соединениях).
- UDP - данные, относящиеся к протоколу UDP (число переданных, принятых и ошибочных UDP-дейтаграмм).
- EGP - данные, относящиеся к протоколу обмена маршрутной информацией ExteriorGatewayProtocol, используемому в сети Internet (число принятых с ошибками и без ошибок сообщений).

Из этого перечня групп переменных видно, что стандарт MIB-I разрабатывался с жесткой ориентацией на управление маршрутизаторами, поддерживающими протоколы стека TCP/IP.

В версии MIB-II (RFC 1213), принятой в 1992 году, был существенно (до 185) расширен набор стандартных объектов, а число групп увеличилось до 10.

Агенты RMON

Новейшим добавлением к функциональным возможностям SNMP является спецификация RMON, которая обеспечивает удаленное взаимодействие с базой MIB.

Стандарт на RMON появился в ноябре 1991 года, когда Internet Engineering Task Force выпустил документ RFC 1271 под названием "Remote Network Monitoring Management Information Base" ("Информационная база дистанционного мониторинга сетей"). Данный документ содержал описание RMON для сетей Ethernet.

RMON — протокол мониторинга компьютерных сетей, расширение SNMP, в основе которого, как и в основе SNMP, лежит сбор и анализ информации о характере информации, передаваемой по сети. Как и в SNMP, сбор информации осуществляется аппаратно-программными агентами, данные от которых поступают на компьютер, где установлено приложение управления сетью. Отличие RMON от своего предшественника состоит, в первую очередь, в характере собираемой информации — если в SNMP эта информация характеризует только события, происходящие на том устройстве, где установлен агент, то RMON требует, чтобы получаемые данные характеризовали трафик между сетевыми устройствами.

До появления RMON протокол SNMP не мог использоваться удаленным образом, он допускал только локальное управление устройствами. База RMON MIB обладает улучшенным набором свойств для удаленного управления, так как содержит агрегированную информацию об устройстве, что не требует передачи по сети больших объемов информации. Объекты RMON MIB включают дополнительные счетчики ошибок в пакетах, более гибкие средства анализа графических трендов и статистики, более мощные средства фильтрации для захвата и анализа отдельных пакетов, а также более сложные условия установления сигналов предупреждения. Агенты RMON MIB более интеллектуальны по сравнению с агентами MIB-I или MIB-II и выполняют значительную часть работы по обработке информации об устройстве, которую раньше выполняли менеджеры. Эти агенты могут располагаться внутри различных коммуникационных устройств, а также быть выполнены в виде отдельных программных модулей, работающих на универсальных ПК и ноутбуках (примером может служить LANalyzerHvell).

Интеллект агентов RMON позволяет им выполнять простые действия по диагностике неисправностей и предупреждению о возможных отказах - например, в рамках технологии RMON можно собрать данные о нормальном функционировании сети (т. е. выполнить так называемый baselining), а потом выставлять предупреждающие сигналы, когда режим работы сети отклонится от baseline - это может свидетельствовать, в частности, о неполной исправности оборудования. Собрав воедино информацию, получаемую от агентов RMON, приложение управления может помочь администратору сети (находящемуся, например, за тысячи километров от анализируемого сегмента сети) локализовать неисправность и выработать оптимальный план действий для ее устранения.

Сбор информации RMON осуществляется аппаратно-программными зондами, подключаемыми непосредственно к сети. Чтобы выполнить задачу сбора и первичного анализа данных, зонд должен обладать достаточными вычислительными ресурсами и объемом оперативной памяти. В настоящее время на рынке имеются зонды трех типов: встроенные, зонды на базе компьютера, и автономные. Продукт считается поддерживающим RMON, если в нем реализована хотя бы одна группа RMON. Разумеется, чем больше групп данных RMON реализовано в данном продукте, тем он, с одной стороны, дороже, а с другой - тем более полную информацию о работе сети он предоставляет.

Встроенные зонды представляют собой модули расширения для сетевых устройств. Такие модули выпускаются многими производителями, в частности, такими крупными компаниями, как 3Com, Cabletron, Bay Networks и Cisco. (Кстати, 3Com и Bay Networks недавно приобрели компании Axon и ARMON, признанных лидеров в области разработки и производства средств управления RMON. Такой интерес к этой технологии со стороны крупнейших производителей сетевого оборудования лишний раз показывает, насколько нужным для пользователей является дистанционный мониторинг.) Наиболее естественным выглядит решение встраивать модули RMON в концентраторы, ведь именно из наблюдения за этими устройствами можно со-ставить себе представление о работе сегмента. Достоинство таких зондов очевидно: они позволяют получать информацию по всем основным группам данных RMON при относительно невысокой цене. Недостатком в первую очередь является не слишком высокая производительность, что проявляется, в частности, в том, что встроенные зонды часто поддерживают далеко не все группы данных RMON. Не так давно 3Com объявила о намерении выпустить поддерживающие RMON драйверы для сетевых адаптеров Etherlink III и Fast Ethernet. В результате окажется возможным собирать и анализировать данные RMON непосредственно на рабочих станциях в сети.

Зонды на базе компьютера - это просто подключенные к сети компьютеры с установленным на них программным агентом RMON. Такие зонды (к числу которых относится, например, продукт Cornerstone Agent 2.5 компании Network General) обладают более высокой производительностью, чем встроенные зонды, и поддерживают, как правило, все группы данных RMON. Они более дороги, чем встроенные зонды, но гораздо дешевле автономных зондов. Помимо этого, зонды на базе компьютера имеют довольно большой размер, что может иногда ограничивать возможности их применения.

Автономные зонды обладают наивысшей производительностью; как легко понять, это одновременно и наиболее дорогие продукты из всех описанных. Как правило, автономный зонд - это процессор (класса i486 или RISC-процессор), оснащенный достаточным объемом оперативной памяти и сетевым адаптером. Лидерами в этом секторе рынка являются компании Frontier и Hewlett-Packard. Зонды этого типа невелики по размеру и весьма мобильны - их очень легко подключать к сети и отключать от нее. При решении задачи управления сетью глобального масштаба это, конечно, не слишком важное свойство, однако если средства RMON применяются для анализа работы корпоративной сети средних размеров, то (учитывая высокую стоимость устройств) мобильность зондов может сыграть весьма положительную роль.

Объекту RMON присвоен номер 16 в наборе объектов MIB, а сам объект RMON объединяет в соответствии с документом RFC 1271, состоит из десяти групп данных.

- Statistics - текущие накопленные статистические данные о характеристиках пакетов, количестве коллизий и т.п.
- History - статистические данные, сохраненные через определенные промежутки времени для последующего анализа тенденций их изменений.

- Alarms - пороговые значения статистических показателей, при превышении которых агент RMON посылает сообщение менеджеру. Позволяет пользователю определить ряд пороговых уровней (эти пороги могут относиться к самым разным вещам - любому параметру из группы статистики, амплитуде или скорости его изменения и многому другому), по превышении которых генерируется аварийный сигнал. Пользователь может также определить, при каких условиях превышение порогового значения должно сопровождаться аварийным сигналом - это позволит избежать генерации сигнала "по пустякам", что плохо, во-первых, потому, что на постоянно горящую красную лампочку никто не обращает внимания, а во-вторых, потому, что передача ненужных аварийных сигналов по сети приводит к излишней загрузке линий связи. Аварийный сигнал, как правило, передается в группу событий, где и определяется, что с ним делать дальше.

- Host - данных о хостах сети, в том числе и об их MAC-адресах..
- HostTopN - таблица наиболее загруженных хостов сети. Таблица N главных хостов (HostTopN) содержит список N первых хостов, характеризующихся максимальным значением заданного статистического параметра для заданного интервала. Например, можно затребовать список 10 хостов, для которых наблюдалось максимальное количество ошибок в течение последних 24 часов. Список этот будет составлен самим агентом, а приложение управления получит только адреса этих хостов и значения соответствующих статистических параметров. Видно, до какой степени такой подход экономит сетевые ресурсы

- TrafficMatrix - статистика об интенсивности трафика между каждой парой хостов сети, упорядоченная в виде матрицы. Строки этой матрицы пронумерованы в соответствии с MAC-адресами станций - источников сообщений, а столбцы - в соответствии с адресами станций-получателей. Матричные элементы характеризуют интенсивность трафика между соответствующими станциями и количество ошибок. Проанализировав такую матрицу, пользователь легко может выяснить, какие пары станций генерируют наиболее интенсивный трафик. Эта матрица, опять-таки, формируется самим агентом, поэтому отпадает необходимость в передаче больших объемов данных на центральный компьютер, отвечающий за управление сетью.

- Filter - условия фильтрации пакетов. Признаки, по которым фильтруются пакеты, могут быть самыми разнообразными - например, можно потребовать отфильтровывать как ошибочные все пакеты, длина которых оказывается меньше некоторого заданного значения. Можно сказать, что установка фильтра соответствует как бы организации канала для передачи пакета. Куда ведет этот канал - определяет пользователь. Например, все ошибочные пакеты могут перехватываться и направляться в соответствующий буфер. Кроме того, появление пакета, соответствующего установленному фильтру, может рассматриваться как событие (event), на которое система должна реагировать заранее оговоренным образом.

- PacketCapture - условия захвата пакетов. В состав группы перехвата пакетов (packet capture) входят буфера для захвата, куда направляются пакеты, чьи признаки удовлетворяют условиям, сформулированным в группе фильтров. При этом захватываться может не пакет целиком, а, скажем, только первые несколько десятков байт пакета. Содержимое буферов перехвата можно впоследствии анализировать при помощи различных программных средств, выясняя целый ряд весьма полезных характеристик работы сети. Перестраивая фильтры на те или иные признаки, можно характеризовать разные параметры работы сети.

- Event - условия регистрации и генерации событий. В группе событий (events) определяется, когда следует отправлять аварийный сигнал приложению управления, когда - перехватывать пакеты, и вообще - как реагировать на те или иные события, происходящие в сети, например, на превышение заданных в группе alarms пороговых значений: следует ли ставить в известность приложение управления, или надо просто запротоколировать данное событие и продолжать работать. События могут и не быть связаны с передачей аварийных сигналов - например, направление пакета в буфер перехвата тоже представляет собой событие.

Данные группы пронумерованы в указанном порядке, поэтому, например, группа Hosts имеет числовое имя 1.3.6.1.2.1.16.4.

Десятую группу составляют специальные объекты протокола TokenRing.

Всего стандарт RMON MIB определяет около 200 объектов в 10 группах, зафиксированных в двух документах - RFC 1271 для сетей Ethernet и RFC 1513 для сетей TokenRing [28, с. 95].

Отличительной чертой стандарта RMON MIB является его независимость от протокола сетевого уровня (в отличие от стандартов MIB-I и MIB-II, ориентированных на протоколы TCP/IP). Поэтому, его удобно использовать в гетерогенных средах, использующих различные протоколы сетевого уровня.

1.2 Популярныe системы управления сетями

Система управления сетью (Network management system) - аппаратные и/или программные средства для мониторинга и управления узлами сети. Программное обеспечение системы управления сетью состоит из агентов, локализуемых на сетевых устройствах и передающих информацию сетевой управляющей платформе. Метод информационного обмена между управляющими приложениями и агентами на устройствах определяется протоколами.

Системы управления сетями должны обладать целым рядом качеств:

1. истинной распределенностью в соответствии с концепцией клиент-сервер,
2. масштабируемостью,
3. открытостью, позволяющей справиться с разнородным - от настольных компьютеров до мейнфреймов - оборудованием.

Первые два свойства тесно связаны. Хорошая масштабируемость достигается за счет распределенности системы управления. Распределенность означает, что система может включать несколько серверов и клиентов. Серверы (менеджерами) собирают данные о текущем состоянии сети от агентов (SNMP, CMIP или RMON), встроенных в оборудование сети, и накапливают их в своей базе данных. Клиенты представляют собой графические консоли, за которыми работают администраторы сети. Программное обеспечение клиента системы управления принимает запросы на выполнение каких-либо действий от администратора (например, построение подробной карты части сети) и обращается за необходимой информацией к серверу. Если сервер обладает нужной информацией, то он сразу же передает ее клиенту, если нет - то пытается собрать ее от агентов.

Ранние версии систем управления совмещали все функции в одном компьютере, за которым работал администратор. Для небольших сетей или сетей с небольшим количеством управляемого оборудования такая структура оказывается вполне удовлетворительной, но при большом количестве управляемого оборудования единственный компьютер, к которому стекается информация от всех устройств сети, становится узким местом. И сеть не справляется с большим потоком данных, и сам компьютер не успевает их обрабатывать. Кроме того, большой сетью управляет обычно не один администратор, поэтому, кроме нескольких серверов в большой сети должно быть несколько консолей, за которыми работают администраторы сети, причем на каждой консоли должна быть представлена специфическая информация, соответствующая текущим потребностям конкретного администратора.

Поддержка разнородного оборудования - скорее желаемое, чем реально существующее свойство современных систем управления. К числу наиболее популярных продуктов сетевого управления относятся четыре системы: Spectrum компании CabletronSystems, OpenView фирмы Hewlett-Packard, NetView корпорации IBM и Solstice производства SunSoft - подразделения Sun Microsystems. Три компании из четырех сами выпускают коммуникационное оборудование. Естественно, что система Spectrum лучше всего управляет оборудованием компании Cabletron, OpenView - оборудованием компании Hewlett-Packard, а NetView- оборудованием компании IBM.

При построении карты сети, которая состоит из оборудования других производителей, эти системы начинают ошибаться и принимать одни устройства за другие, а при управлении этими устройствами поддерживают только их основные функции, а многие полезные дополнительные функции, которые отличают данное устройство от остальных, система управления просто не понимает и, поэтому, не может ими воспользоваться.

Для исправления этого недостатка разработчики систем управления включают поддержку не только стандартных баз MIB I, MIB II и RMON MIB, но и многочисленных частных MIB фирм-производителей. Лидер в этой области - система Spectrum, поддерживающая около 1000 баз MIB различных производителей.

Другим способом более качественной поддержки конкретной аппаратуры является использование на основе какой-либо платформы управления приложения той фирмы, которая выпускает это оборудование. Ведущие компании - производители коммуникационного оборудования - разработали и поставляют весьма сложные и многофункциональные системы управления для своего оборудования. К наиболее известным системам этого класса относятся Optivity компании BayNetworks, CiscoWorks компании CiscoSystems, Transcend компании 3Com. Система Optivity, например, позволяет производить мониторинг и управлять сетями, состоящими из маршрутизаторов, коммутаторов и концентраторов компании BayNetwork, полностью используя все их возможности и свойства. Оборудование других производителей поддерживается на уровне базовых функций управления. Система Optivity работает на платформах OpenView компании Hewlett-Packard и SunNetManager (предшественник Solstice) компании SunSoft. Однако, работа на основе какой-либо платформы управления с несколькими системами, такими как Optivity, слишком сложна и требует, чтобы компьютеры, на которых все это будет работать, обладали очень мощными процессорами и большой оперативной памятью.

Тем не менее, если в сети преобладает оборудование от какого-либо одного производителя, то наличие приложений управления этого производителя для какой-либо популярной платформы управления позволяет администраторам сети успешно решать многие задачи. Поэтому разработчики платформ управления поставляют вместе с ними инструментальные средства, упрощающие разработку приложений, а наличие таких приложений и их количество считаются очень важным фактором при выборе платформы управления.

Открытость платформы управления зависит также от формы хранения собранных данных о состоянии сети. Большинство платформ-лидеров позволяют хранить данные в коммерческих базах данных, таких как Oracle, Ingres или Informix. Использование универсальных СУБД снижает скорость работы системы управления по сравнению с хранением данных в файлах операционной системы, но зато позволяет обрабатывать эти данные любыми приложениями, умеющими работать с этими СУБД.

1.3 Популярные системы мониторинга сетей

В Приложении А в виде таблицы показан обзор наиболее популярных систем сетевого мониторинга на сегодняшний день.

Системы мониторинга сети предоставляются как проприетарным программным обеспечением, так и программными продуктами с открытым исходным кодом.

Среди проприетарного программного обеспечения можно выделить такие системы, как:

1. 10-Страйк: Мониторинг Сети
2. Friendly Pinger

"10-Страйк: Мониторинг Сети "предназначена для мониторинга серверов и прочих сетевых устройств. Программа следит за работоспособностью сети и оповещает о неполадках. Позволяет узнать вовремя о произошедшем сбое (разрыв связи, завершение места на диске сервера, останов службы и т.п.) и устранить проблему с минимальными потерями времени.

Программа отслеживает состояние серверов, хостов, папок, файлов, баз данных, процессов и служб на компьютерах сети. Дополнительно администратор сети может использовать собственные скрипты в качестве проверок, и даже контролировать различные переменные в управляемых коммутаторах по протоколу SNMP.

Friendly Pinger - это мощное приложение для администрирования, мониторинга и инвентаризации компьютерной сети.

Возможности программы:

1. Визуализация компьютерной сети в красивой анимационной форме;
2. Отображение, какие компьютеры включены, а какие нет;
3. Пингование всех устройств за раз;
4. Оповещение в случае остановки/запуска серверов;
5. Инвентаризация программного и аппаратного обеспечения всех компьютеров в сети;
6. Слежение, кто "лазает" по Вашему компьютеру и какие файлы качает;
7. Назначение внешних команд (например, telnet, tracert, net.exe) устройствам;

8. Поиск HTTP, FTP, e-mail и других сетевых служб;
9. Отображение состояния сети на рабочем столе или Web странице;
10. Графический TraceRoute;
11. Открытие компьютеров в проводнике или Total Commander

Среди открытого программного обеспечения можно выделить следующие системы мониторинга:

1. Nagios
2. Zabbix

Nagios — программа мониторинга компьютерных систем и сетей с открытым кодом. Предназначена для наблюдения, контроля состояния вычислительных узлов и служб, оповещает администратора в том случае, если какие-то из служб прекращают (или возобновляют) свою работу.

Nagios первоначально была создана под именем Netsaint, разработана Этаном Галстадом (англ. Ethan Galstad). Он же поддерживает и развивает систему сегодня, совместно с командой разработчиков, которые занимаются как официальными, так и неофициальными плагинами.

Nagios предоставляет небывалый контроль за всеми компьютерами сети, позволяющий исправить ошибки до того, как они станут критическими, и быть уверенными в том, что вы знаете о своей сети все. Базовая структура Nagios очень проста. Один из компьютеров становится сервером Nagios, собирает информацию с указанных клиентских компьютеров и аккуратно отображает ее на web-странице. Для получения базовых сведений о клиентских компьютерах (пингуются ли они, и работают ли SSH и HTTP) на них не нужно устанавливать никакого ПО: всю работу сделает сервер. Для получения централизованных отчетов с дополнительной информацией (например, доступным дисковым пространством или использованием процессора) на клиентские компьютеры потребуется установить ПО. При возникновении аварийных ситуаций сервер может реагировать на них путем отправки электронного письма, вывода сообщения на экран или выполнения других действий, заданных в файле скрипта.

Программа чрезвычайно гибкая, настраиваемая и мощная. Обратной стороной тут неизбежно является сложность правильной настройки. На данном уроке мы рассмотрим конфигурацию центрального сервера и одного клиентского компьютера (все остальные настраиваются точно так же) и слегка коснемся других возможностей, которые можно исследовать после инсталляции. Это непросто, но стоит приложенных усилий, так как после установки Nagios поддержки почти не понадобится.

Первоначально Nagios была разработана для работы под Linux, но она также хорошо работает и под другими ОС, такими как Sun Solaris, FreeBSD, AIX и HP-UX.

Обзор возможностей:

1. Мониторинг сетевых служб (SMTP, POP3, HTTP, NNTP, ICMP, SNMP)
2. Мониторинг состояния хостов (загрузка процессора, использование диска, системные логи) в большинстве сетевых операционных систем
3. Поддержка удаленного мониторинга через зашифрованные туннели SSH или SSL
4. Простая архитектура модулей расширений (плагинов) позволяет, используя любой язык программирования по выбору (Shell, C++, Perl, Python, PHP, C# и другие), легко разрабатывать свои собственные способы проверки служб
5. Параллельная проверка служб
6. Возможность определять иерархии хостов сети с помощью «родительских» хостов, позволяет обнаруживать и различать хосты, которые вышли из строя, и те, которые недоступны

7. Отправка оповещений в случае возникновения проблем со службой или хостом (с помощью почты, пейджера, смс, или любым другим способом, определенным пользователем через модуль системы)
8. Возможность определять обработчики событий произошедших со службами или хостами для проактивного разрешения проблем
9. Автоматическая ротация лог-файлов
10. Возможность организации совместной работы нескольких систем мониторинга с целью повышения надёжности и создания распределенной системы мониторинга
11. Включает в себя утилиту nagiosstats, которая выводит общую сводку по всем хостам, по которым ведется мониторинг.

ZABBIX — [свободная](#) система мониторинга корпоративного класса и отслеживания статусов разнообразных сервисов [компьютерной сети](#), [серверов](#) и [сетевого оборудования](#), создан [Алексеем Владышевым](#) и в настоящее время активно разрабатывается и поддерживается Zabbix SIA.

Для хранения данных используется [MySQL](#), [PostgreSQL](#), [SQLite](#) или [Oracle](#). Веб-интерфейс написан на [PHP](#).

ZABBIX поддерживает несколько видов мониторинга:

1. Simple checks — может проверять доступность и реакцию стандартных сервисов, таких как SMTP или HTTP без установки какого-либо программного обеспечения на наблюдаемом хосте.
2. ZABBIX agent — может быть установлен на [UNIX-подобных](#) или [Windows](#) хостах для получения данных о нагрузке [процессора](#), использования сети, дисковом пространстве и т. д.

3. External check — выполнение внешних программ. ZABBIX также поддерживает мониторинг через [SNMP](#).

Zabbix предлагает:

1. автоматическое обнаружение серверов и других устройств в сети
2. распределенный мониторинг с централизованным администрированием через ВЕБ
3. поддержка обеих механизмов пуллеров и трапперов
4. серверное программное обеспечение для Linux, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, OS X
5. родные агенты с высокой производительностью (клиентское программное обеспечение для Linux, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, OS X, Tru64/OSF1, Windows NT4.0, Windows 2000, Windows 2003, Windows XP, Windows Vista)
6. мониторинг без агентов
7. безопасная аутентификация пользователей
8. гибкая система прав доступа пользователей
9. Web-интерфейс
10. гибкая система уведомлений по e-mail о предопределенных событиях
11. высокоуровневый (класса “Бизнес”) вид контроля ресурсов
12. журнал аудита

Почему организации отдают предпочтение Zabbix:

1. Открытое программное обеспечение
 2. агенты с высокой эффективностью для UNIX и WIN32 платформ
 3. легко изучаемый
 4. увеличивает рентабельность (простои очень дорого обходятся)
 5. низкая стоимость обслуживания
 6. очень простое конфигурирование
 7. централизованная система мониторинга. Вся информация (конфигурация и данные о производительности) хранятся в реляционной базе данных
 8. высокоуровневое дерево предоставляемых услуг
 9. очень простая установка
 10. поддержка SNMP (v1,v2,v3). Оба режима пуллера и траппера.
 11. возможность визуализации
- встроенный механизм очистки устаревших данных

1.4 Выбор системы мониторинга

В разрабатываемой системе нужно учесть несколько аспектов:

- Кроссплатформенность системы.
- Мобильность
- Простота конфигурации

Кроссплатформенное программное обеспечение — программное обеспечение, работающее более чем на одной аппаратной платформе и/или операционной системе. Является одним из главных аспектов, учитываемый при разработке данной системы мониторинга — возможность запуска на ОС семейства Linux, Windows, а так же мобильных платформах.

Мобильность — под этим термином подразумевается как возможность системных администраторов управлять системой в мобильном режиме(находясь не на рабочем месте), так и возможность запуска с мобильных устройств, таких как смартфоны, карманные компьютеры, планшетные компьютеры и т.д.

Системный администратор по специфике своей работы, не всегда может находится на своем рабочем месте и имеет место необходимость в своевременном уведомлении администратора о возникшей проблеме. Некоторые варианты такого уведомления могут реализованы по средством сотовых сетей и мобильных устройств связи: сообщения SMS, голосовые вызовы от систем мониторинга. Так же такая возможность может быть предоставлена с помощью систем мгновенного обмена сообщений — ICQ, XMPP(Jabber), Skype.

Запуск системы мониторинга с мобильных устройств обусловлена возможностью сбора кратковременных статистических данных о ресурсах сети, а так же для ее анализа и диагностики, либо невозможностью развернуть систему мониторинга стандартным способом.

Конфигурирование системы не должно занимать много времени и ресурсов системного администратора. Так же стоит заметить, что в организациях где существуют более 2х системных администраторов, может появится необходимость настройки системы мониторинга под каждого из системных администраторов, например, первый администратор следит за

состоянием серверов и оборудования в серверной А, а второй в серверной Б, и соответственно для второго администратора информационные сообщения для первого будут иметь «паразитный» характер.

Ни одна из вышеперечисленных систем мониторинга не удовлетворяет нашим требованиям, поэтому было решено разрабатывать собственную систему мониторинга.

Разрабатываемая система мониторинга будет обладать следующим функционалом:

1. Поддержка БД SQLite, MySQL. SQLite будет использоваться как в мобильной версии приложения, так и для PC платформы. MySQL будет использоваться только на PC платформе. В дальнейшем возможны реализация поддержки БД Oracle, PostgreSQL и MS SQL.
2. Простота конфигурирования. Основные настройки будут храниться в текстовом файле. Остальная конфигурация ложится на пользователя – добавление и удаления сетевых ресурсов для мониторинга.
3. Разделения на пользователей. Каждый пользователь системы будет иметь свой профиль со своими настройками для конфигурации объектов мониторинга.
4. Полное управление через XMPP протокол. Так же будет реализован web-интерфейс, который позволит просматривать статистику и конфигурировать объекты мониторинга.
5. Агенты для клиентских машин под управлением ОС Windows и Linux, для сбора информации и мониторинга состояния системы.

6. Приложение будет доступно в 2х редакциях – для мобильных платформ и для РС.

7. Будут поддерживаться следующие способы мониторинга:

7.1. Мониторинг доступности посредством протокола ICMP и SNMP;

7.2. Мониторинг доступности по протоколам http, smtp, telnet, ftp и прочих;

7.3. Агенты для сбора информации с систем.

На данный момент система находится в стадии разработки, и в данном дипломном проекте предоставлена с минимальным функционалом: реализована поддержка БД MySQL, управление через XMPP протокол, мониторинг посредством ICMP.

2 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Выбор инструментария для разработки системы.

В качестве среды разработки был выбран язык программирования C++ с кроссплатформенной библиотекой QT.

C++ — компилируемый статически типизированный язык программирования общего назначения.

Поддерживает разные парадигмы программирования, сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. Название «C++» происходит от названия языка C, в котором унарный оператор ++ обозначает инкремент переменной.

Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (например, видеоигры). Существует несколько реализаций языка C++ — как бесплатных, так и коммерческих. Наиболее популярны проект GNU, Microsoft, Intel и Embarcadero (Borland). C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

При создании C++ Бьёрн Страуструп стремился сохранить совместимость с языком C. Множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико — отчасти благодаря тому, что синтаксис C++ был основан на синтаксисе C.

Qt - кросс-платформенный инструментальный разработки ПО на языке программирования C++. Есть также «привязки» ко многим другим языкам программирования: Python - PyQt, PySide; Ruby — QtRuby ; Java — Qt Jambi; PHP PHP-Qt и другие.

Позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Существуют версии библиотеки для Microsoft Windows, систем класса UNIX с графической подсистемой X11, iOS, Android, Mac OS X, Microsoft Windows CE, QNX, встраиваемых Linux-систем и платформы S60. Также идёт портирование на Haiku.

Qt, в частности, используются в Autodesk Maya, Adobe Photoshop Elements, OPIE, Skype, VLC media player, VirtualBox, и Mathematica, и на European Space Agency, DreamWorks, Google, HP, KDE, Lucasfilm, Panasonic, Philips, Samsung, Siemens, Volvo и Walt Disney Animation Studios.

До недавнего времени библиотека Qt также распространялась ещё в одной версии: Qt/Embedded. Теперь эта платформа переименована в *Qtopia Core* и распространяется как отдельный продукт. *Qtopia Core* обеспечивает базовую функциональность для всей линейки платформ, предназначенных для разработки приложений для встраиваемых и мобильных устройств (КПК, смартфонов и т. п.).

Начиная с версии 4.5 Qt распространяется по 3 лицензиям:

1. Qt Commercial — для разработки ПО с собственной лицензией, допускающая модификацию самой Qt без раскрытия изменений.
2. GNU GPL — для разработки ПО с открытыми исходниками распространяемыми на условиях GNU GPL.
3. GNU LGPL — для разработки ПО с собственной лицензией, но без внесения изменений в Qt.

Примечание: Независимо от лицензии, исходный код Qt один и тот же.

До версии 4.0.0 под свободной лицензией распространялись лишь Qt/Mac, Qt/X11, Qt/Embedded, но, начиная с 4.0.0 (выпущенной в конце июня 2005), Qt Software «освободили» и Qt/Windows. Следует отметить, что существовали сторонние свободные версии Qt/Windows < 4.0.0, сделанные на основе Qt/X11.

Со времени своего появления в 1996 году библиотека Qt легла в основу тысяч успешных проектов во всём мире. Кроме того, Qt является фундаментом популярной рабочей среды KDE, входящей в состав многих дистрибутивов Linux.

Отличительная особенность Qt от других библиотек — использование *Meta Object Compiler (MOC)* — предварительной системы обработки исходного кода (в общем-то, Qt — это библиотека не для чистого C++, а для его особого наречия, с которого и «переводит» MOC для последующей компиляции любым стандартным C++ компилятором). MOC позволяет во много раз увеличить мощь библиотек, вводя такие понятия, как *слоты* и *сигналы*. Кроме того, это позволяет сделать код более лаконичным. Утилита

MOC ищет в заголовочных файлах на C++ описания классов, содержащие макрос `Q_OBJECT`, и создаёт дополнительный исходный файл на C++, содержащий *мета-объектный* код.

Qt позволяет создавать собственные плагины и размещать их непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Qt комплектуется визуальной средой разработки графического интерфейса «Qt Designer», позволяющей создавать диалоги и формы «мышью» (в режиме WYSIWYG). В поставке Qt есть «Qt Linguist» — графическая утилита, позволяющая упростить локализацию и перевод вашей программы на многие языки; и «Qt Assistant» — справочная система Qt, упрощающая работу с документацией по библиотеке, а также позволяющая

создавать кросс-платформенную справку для разрабатываемого на основе Qt ПО.

Начиная с версии 4.5.0 в комплект Qt включена среда разработки «Qt Creator» (ранее известная под кодовым названием Greenhouse) — кроссплатформенная свободная IDE для работы с фреймворком Qt, разработанная Trolltech (Nokia). Анонс проекта состоялся на Qt Developer Days в октябре 2008 года. Публичная бета-версия проекта была опубликована 30 октября 2008 года. Финальный релиз состоялся 3 марта 2009 года (вместе с выходом Qt 4.5), а исходный код доступен под лицензией LGPL.

Qt Creator обладает следующими особенностями:

1. Сделана специально для разработки на Qt.
2. Встроенный редактор форм (Qt Designer) и справочная система (Qt Assistant).
3. Контекстно-зависимая система помощи.
4. Расширяема плагинами.
5. Имеется графический фронтенд для GDB.
6. Поддержка отладки с помощью CDB.
7. Для создания проектов используется qmake (планируется поддержка Makefile и тестируется поддержка CMake).
8. Обобщённая подсветка синтаксиса, поддерживается большое количество языков программирования и разметки. Есть возможность создания своих стилей подсветки.
9. Возможность редактировать этапы сборки проекта.
10. Поддержка разработки на языках C/C++/QML.
11. QML-дизайнер.
12. Возможность разработки под Symbian и Maemo с отладкой в симуляторе или на устройстве.

Библиотека QT разделена на несколько модулей, для четвёртой версии библиотеки это:

1. QtCore — классы ядра библиотеки, используемые другими модулями;
2. QtGui — компоненты графического интерфейса;
3. QtNetwork — набор классов для сетевого программирования. Поддержка различных высокоуровневых протоколов может меняться от версии к версии. В версии 4.2.x присутствуют классы для работы с протоколами FTP и HTTP. Для работы с протоколами TCP/IP предназначены такие классы, как QTcpServer, QTcpSocket для TCP и QUdpSocket для UDP;
4. QtOpenGL — набор классов для работы с OpenGL;
5. QSql — набор классов для работы с базами данных с использованием языка структурированных запросов SQL. Основные классы данного модуля в версии 4.2.x: QSqlDatabase — класс для предоставления соединения с базой, для работы с какой-нибудь конкретной базой данных требует объект, унаследованный от класса QSqlDriver — абстрактного класса, который реализуется для конкретной базы данных и может требовать для компиляции SDK базы данных. Например, для сборки драйвера под базу данных FireBird/InterBase требует .h файлы и библиотеки статической линковки, входящие в комплект поставки данной БД;
6. QtScript — классы для работы с Qt Scripts;
7. QtSvg — классы для отображения и работы с данными Scalable Vector Graphics(SVG);
8. QtXml — модуль для работы с XML, поддерживается SAX и DOM модели работы;
9. QtDesigner — классы создания расширений QtDesigner'а для своих собственных виджетов;

10. QtUiTools — классы для обработки в приложении форм Qt Designer;
11. QtAssistant — справочная система;
12. Qt3Support — модуль с классами, необходимыми для совместимости с библиотекой Qt версии 3.x.x;
13. QTest — модуль для работы с UNIT тестами;
14. QtWebKit — модуль WebKit, интегрированный в Qt и доступный через её классы;
15. QtXmlPatterns — модуль для поддержки XQuery 1.0 и XPath 2.0;
16. Phonon — модуль для поддержки воспроизведения и записи видео и аудио, как локально, так и с устройств и по сети;
17. QtCLucene — модуль для поддержки полнотекстового поиска, применяется в новой версии Assistant в Qt 4.4;
18. ActiveQt — модуль для работы с ActiveX и COM технологиями для Qt-разработчиков под Windows.
19. QtDeclarative — модуль, предоставляющий декларативный фреймворк для создания динамических, настраиваемых пользовательских интерфейсов.

Также реализована технология WoC — widgets on canvas, с помощью которой реализована Plasma в KDE 4.1, Будет возможным использовать виджеты библиотеки Qt прямо в апплетах. Обеспечивает расположение виджетов на QGraphicsView с возможностью масштабирования и различных графических эффектов.

Библиотека использует собственный формат проекта, именуемый .pro файлом, в котором собрана информация о том, какие файлы будут скомпилированы, по каким путям искать заголовочные файлы и много другой информации. Впоследствии при помощи утилиты qmake из них получаются makefile для make-утилиты компилятора. Также есть возможность работы при помощи интеграторов с Microsoft Visual Studio

2003/2005/2008/2010. Совсем недавно стала доступна интеграция в Eclipse для версии библиотеки 4.x.x.

Одним из весомых преимуществ проекта Qt является наличие качественной документации, в отличие, например, от wxWidgets. Статьи документации снабжены большим количеством примеров. Исходный код самой библиотеки хорошо форматирован, подробно комментирован и легко читается, что также упрощает изучение Qt.

На Qt основана среда рабочего стола KDE, графический интерфейс мобильной ОС MeeGo и Qt Creator — среда разработки на Qt, написана с использованием этого фреймворка. Список наиболее популярных приложений использующих QT показан в приложении Б.

Исходя из вышеуказанных достоинств проекта QT, было решено разрабатывать систему мониторинга с помощью QT SDK version 1.1.3 с Qt libraries 4.7.4 for Windows. Скачивается данное программное обеспечение с официального сайта проекта qt.nokia.com и устанавливается согласно документации.

2.2 Установка необходимых компонентов

При разработке данного проекта нам потребуются библиотеки для работы с MySQL и протоколом XMPP.

Решено было использовать нативный для QT драйвер QMYSQL для работы с БД MySQL и библиотеку QXMPP для работы с протоколом XMPP соответственно.

2.2.1 Установка драйвера QMYSQL

Qt дает возможность создания платформо-независимых приложений для работы с базами данных, используя стандартные СУБД. Qt включает

«родные» драйвера для Oracle, Microsoft SQL Server, Sybase Adaptive Server, IBM DB2, PostgreSQL, MySQL и ODBC-совместимых баз данных. Qt включает специфичные для баз данных виджеты, а также поддерживает расширение для работы с базами данных любых встроенных или отдельно написанных виджетов.

Работа с базами данных в Qt происходит на различных уровнях:

1. Слой драйверов — Включает классы QSqlDriver, QSqlDriverCreator, QSqlDriverCreatorBase, QSqlDriverPlugin и QSqlResult. Этот слой предоставляет низкоуровневый мост между определенными базами данных и слоем SQL API.
2. Слой SQL API — Этот слой предоставляет доступ к базам данных. Соединения устанавливаются с помощью класса QSqlDatabase. Взаимодействие с базой данных осуществляется с помощью класса QSqlQuery. В дополнение к классам QSqlDatabase и QSqlQuery слой SQL API опирается на классы QSqlError, QSqlField, QSqlIndex и QSqlRecord.
3. Слой пользовательского интерфейса — Этот слой связывает данные из базы данных с дата-ориентированными виджетами. Сюда входят такие классы, как QSqlQueryModel, QSqlTableModel и QSqlRelationalTableModel.

Чтобы получить доступ к базе данных с помощью QSqlQuery и QSqlQueryModel, необходимо создать и открыть одно или более соединений с базой данных.

Qt может работать со следующими базами данных (из-за несовместимости с GPL лицензией, не все плагины поставляются с Qt Open Source Edition):

1. QDB2 — IBM DB2 (версия 7.1 и выше)

2. QIBASE — Borland InterBase
3. QMYSQL — MySQL
4. QOCI — Драйвер Oracle Call Interface
5. QODBC — Open Database Connectivity (ODBC) — Microsoft SQL Server и другие ODBC-совместимые базы данных
6. QPSQL — PostgreSQL (версия 7.3 и выше)
7. QSQLITE2 — SQLite версии 2
8. QSQLITE — SQLite версии 3
9. QTDS — Драйвер Sybase Adaptive Server

Для сборки драйвера MySQL нам понадобится установочный пакет MySQL. При разработке проекта использовался MySQL Server 5.5, данный программный продукт доступен для скачивания с официального ресурса проекта www.mysql.com.

При установке MySQL необходимо выбрать Custom Setup, где выбрать установку компонента Client C API Library. (рисунки 2.1) Каталог установки располагается по адресу `c:\Servers\MySQL_55`.

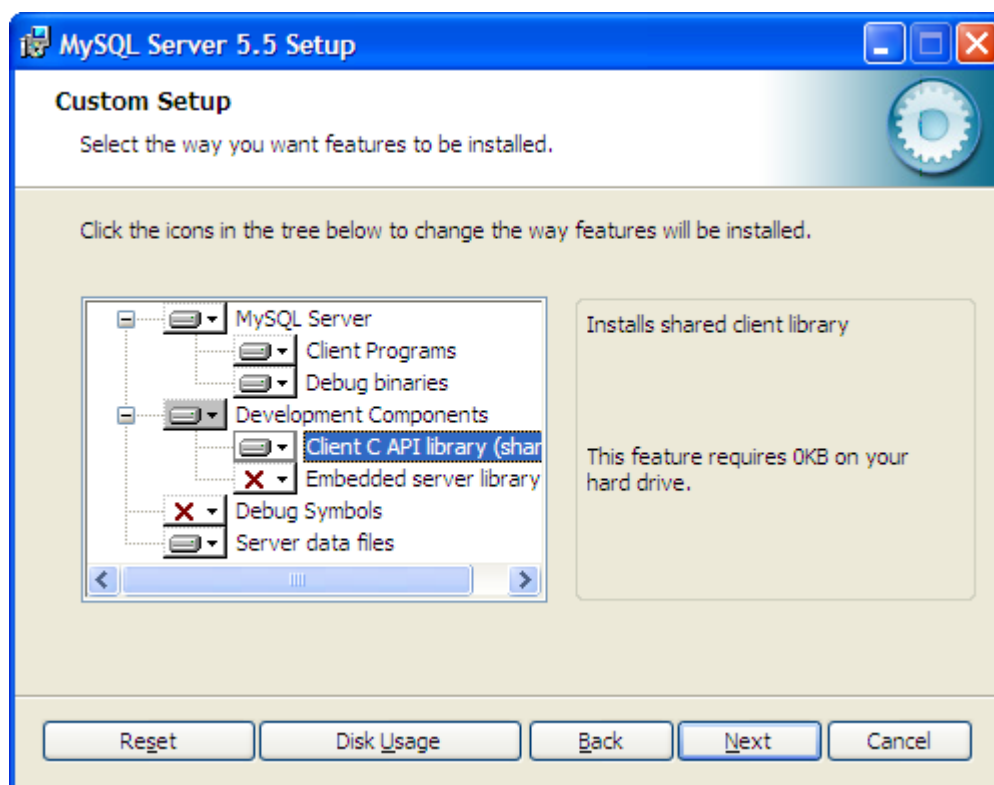


Рисунок 2.1. Установка MySQL Server 5.5

Далее необходимо войти в командную строку QT и выполнить:

```
# cd %QTDIR%\src\plugins\sqldrivers\mysql  
  
# qmake -o Makefile "INCLUDEPATH+=C:\Servers\MySQL_55\include"  
"LIBS+= C:\Servers\MySQL_55\lib\opt\libmysql.lib" mysql.pro  
  
# mingw32-make
```

После того, как сборка драйвера закончится, можно его будет использовать.

Для работы с БД в QT имеется модуль QtSQL, который помогает обеспечить однородную интеграцию БД в ваши Qt приложения и включает в себя:

Пространства имен:

QSql - Различные идентификаторы, используемые в разных библиотеках Qt SQL

Классы:

1. QSqlDatabase - Предоставляет соединение с базой данных
2. QSqlDriver - Абстрактный базовый класс для доступа к специфическим базам данных SQL
3. QSqlDriverCreator - Класс-шаблон, предоставляющий фабрику драйверов SQL для специфических типов драйверов
4. QSqlDriverCreatorBase - Базовый класс для фабрик драйверов SQL
5. QSqlDriverPlugin - Абстрактный базовый класс для пользовательских подключаемых модулей QSqlDriver
6. QSqlError - Информация об ошибке базы данных SQL

7. QSqlField - Управление полями в таблицах и представлениях базы данных SQL
8. QSqlIndex - Функции для управления индексами базы данных и их описания
9. QSqlQuery - Средства управления выражениями SQL и их выполнения
10. QSqlQueryModel - Модель данных только для чтения результирующей SQL-выборки
11. QSqlRecord - Заключает в себе запись базы данных
12. QSqlRelation - Содержит информацию о внешнем ключе SQL
13. QSqlRelationalDelegate - Делегат, используемый для отображения и редактирования данных из QSqlRelationalTableModel
14. QSqlRelationalTableModel - Модель данных для редактирования одной таблицы базы данных с поддержкой внешних ключей
15. QSqlResult - Абстрактный интерфейс для доступа к данным специфичной базы SQL
16. QSqlTableModel - Редактируемая модель данных для одной таблицы базы данных

2.2.2 Установка библиотеки QXMPP.

QXmpp – активно развивающийся проект, представляет кроссплатформенную C++ библиотекой для работы с протоколом XMPP, используя для этого TCP сокет и XMPP RFCs (RFC3920 и RFC3921), которые были воплощены в ее функциях и классах.

Для установки QXMPP нам необходимо взять с официального сайта code.google.com/p/qxmpp/ архив `qxmpp-0.3.0.tar.gz`, на момент написания данного проекта, библиотека имела стабильную версию 0.3.0. После распаковки архива, собрать библиотеку можно двумя способами:

1. Используя командную строку: зайти в папку, где располагается файл qxmpp.pro и выполнить программу mingw32-make для сборки библиотеки и mingw32-make install для установки.

```
# cd c:\QT\qxmpp-0.3.0\
```

```
# mingw32-make
```

```
# mingw32-make install
```

2. Используя QT Creator – достаточно открыть библиотеку, как проект и выполнить Build All и Install.

3 Разработка системы

2.3.1 Алгоритм работы системы

На данный момент система мониторинга имеет 2 модуля: модуль мониторинга объектов и модуль взаимодействия (оповещение о событиях, управление системой) с пользователями и администраторами через протокол XMPP.

2.3.1.1 Алгоритм работы системы мониторинга

Алгоритм работы системы мониторинга показан на рисунке 2.2. После запуска системы происходит чтения конфигурационного файла, откуда берутся данные об интервале опроса объектов, данные для подключения к БД. После этого создается таймер и система ожидает его срабатывания. Затем система получает данные из таблицы station и в соответствии с этими данными начинает опрос. Результаты опроса сравниваются с последним состоянием объекта и если оно изменилось (поле status в таблице station) то делается запись в таблицу events о событии, иначе система переходит в ожидание срабатывания таймера.

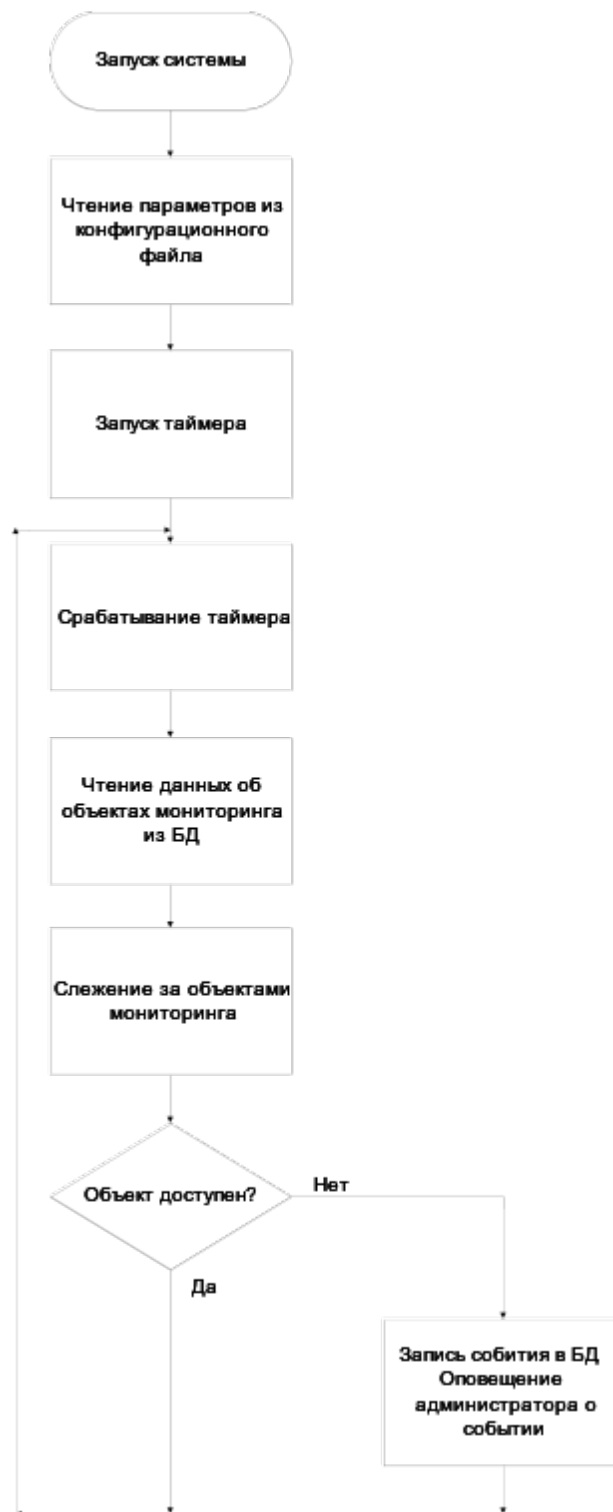


Рисунок 2.2. Алгоритм работы системы мониторинга

2.3.1.2 Алгоритм работы оповещения

Алгоритм системы оповещения показан на рисунке 2.3. После старта системы происходит чтения конфигурационного файла, откуда берется информация необходимая для подключения к БД и подключения к Jabber

серверу. Создается таймер, после срабатывания которого берутся данные из таблицы events и анализируются. Если данные изменились, то происходит уведомления пользователя о новых событиях.

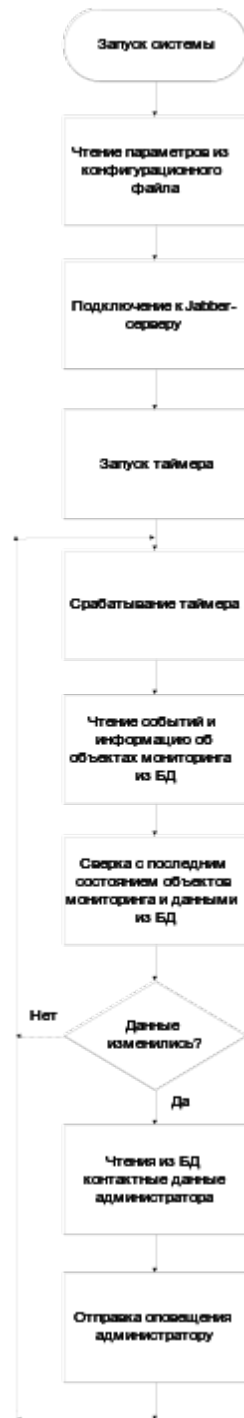


Рисунок 2.3. Алгоритм работы системы оповещения

2.3.2. Структура БД

При разработке данного проекта была создана БД pitoxm содержащая на данный момент 3 таблицы:

1. users – содержит информацию о пользователях системы(таблица 2.1).
2. station – содержит информацию об объекте мониторинга(таблица 2.2)
3. events – содержит события произошедшие с объектами мониторинга(таблица 2.3)

Таблица 2.1. Состав таблицы users

Имя поля	Тип	Описание
ID	int	Ключ. Уникальный идентификатор пользователя
UID	text	Имя учетной записи хтpp пользователя
Description	text	Дополнительное описание пользователя
Admin	bool	Флаг, показывает является ли пользователь администратором системы

Таблица 2.2. Состав таблицы station

Имя поля	Тип	Описание
ID	int	Ключ. Уникальный идентификатор объекта мониторинга
Server	text	Имя объекта мониторинга
IPadress	text	IP-адрес сервера
Status	bool	Содержит статус объекта мониторинга

Таблица 2.3. Состав таблицы events

Имя поля	Тип	Описание
ID_ev	int	Ключ. Уникальный идентификатор события
ID	int	Идентификатор объекта мониторинга
status	bool	Статус объекта мониторинга
date	date	Дата события
time	time	Время события

В настоящее время большинство проектов информационных систем (ИС) разрабатывается в соответствии с какой-либо методологией разработки ПО. Как следствие, разработчикам требуется инструмент для моделирования данных на этапах анализа и проектирования. Таким инструментом являются ER-модель (*entity-relationship model*).

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма сущность-связь (ER-диаграмма) (англ. *entity-relationship diagram*, ERD). Понятия ER-модель и ER-диаграмма часто ошибочно не различают, хотя для визуализации ER-моделей предложены и другие графические нотации:

1. Нотация Питера Чена. Множества сущностей изображаются в виде прямоугольников, множества отношений изображаются в виде ромбов. Если сущность участвует в отношении, они связаны линией. Если отношение не является обязательным, то линия пунктирная. Атрибуты изображаются в виде овалов и связываются линией с одним отношением или с одной сущностью. Пример нотации Питера Чена можно посмотреть на рисунке 2.4.

2. Crow's Foot. Пример отношения между сущностями согласно нотации Crow's Foot. Данная нотация была предложена Гордоном Эверестом (англ. Gordon Everest) под названием Inverted Arrow («перевёрнутая стрелка»), однако сейчас чаще называемая Crow's Foot («воронья лапка») или Fork («вилка»). Согласно данной нотации, сущность изображается в виде прямоугольника, содержащем её имя, выражаемое существительным. Имя сущности должно быть уникальным в рамках одной модели. При этом, имя сущности — это имя типа, а не конкретного экземпляра данного типа. Экземпляром сущности называется конкретный представитель данной сущности.

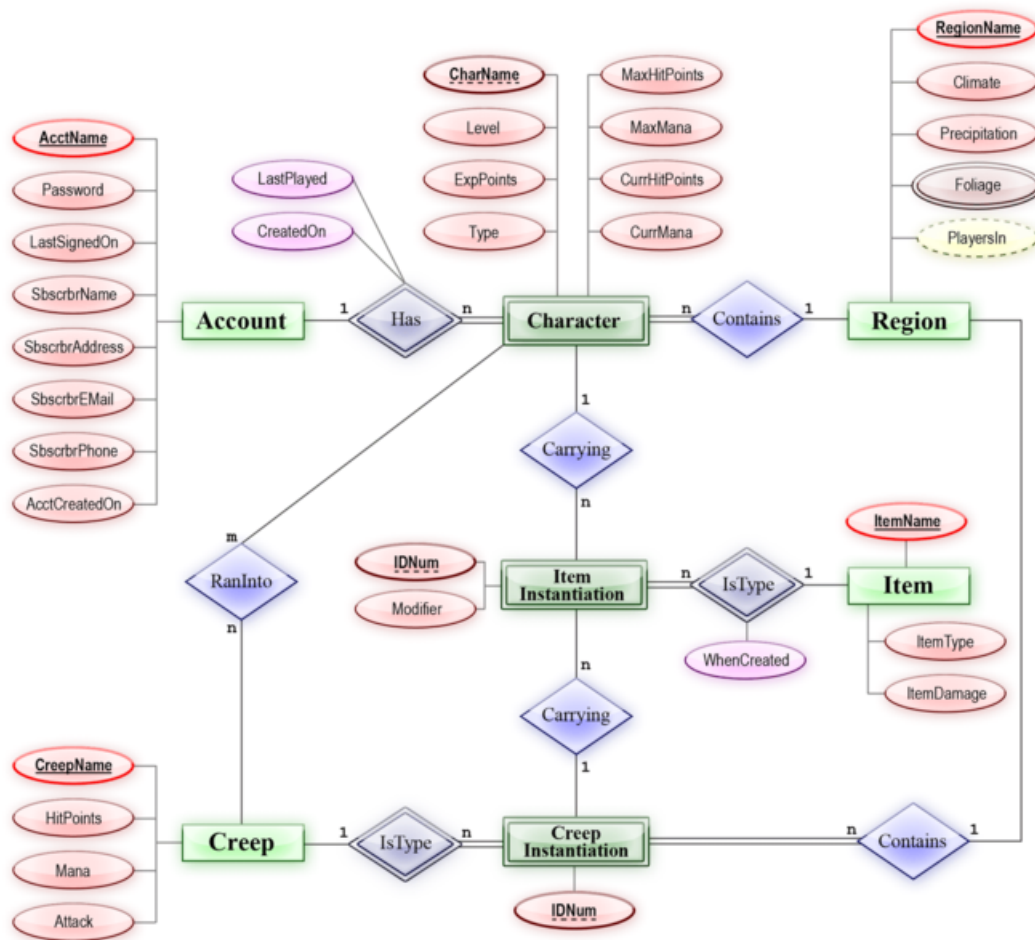


Рисунок 2.4. Простая ER-модель MMORPG использованием нотации Питера Чена

Связь изображается линией, которая связывает две сущности, участвующие в отношении(рисунок 2.5). Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Именование обычно выражается одним глаголом в изъявительном наклонении настоящего времени: «Имеет», «Принадлежит» и т. Д.; или глаголом с поясняющими словами: «Включает в себя», и т.п. Наименование может быть одно для всей связи или два для каждого из концов связи. Во втором случае, название левого конца связи указывается над линией связи, а

правого – под линией. Каждое из названий располагается рядом с сущностью, к которой оно относится.

Атрибуты сущности записываются внутри прямоугольника, изображающего сущность и выражаются существительным в единственном числе (возможно, с уточняющими словами). Среди атрибутов выделяется ключ сущности — избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности.

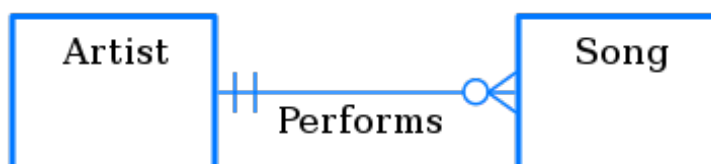


Рисунок 2.5. Изображение ER-модели Crow's Foot

Для нашей базы данных ER-диаграмма будет выглядеть следующим образом(рисунок 2.6).

Программа составлена и отлажена в бесплатной среде разработки VM Lab версии 3.14 (в окне 'About Visual Micro Lab' эта версия указана как 3.12).

Код программы написан на языке ассемблера, содержит более 1500 строк, занимает в памяти программ МК более 3 КБ. Задействованы все 32 регистра, 51 байт оперативной памяти (SRAM, ОЗУ), 45 байт энергонезависимой памяти EEPROM.

На первый взгляд, возможности МК ATmega16, имеющего 16 КБ программной памяти, 1 КБ SRAM, 512 байт EEPROM, и богатый набор периферийных устройств, кажутся избыточными. На самом деле, выбор МК "с запасом" сделан умышленно, чтобы иметь возможность

совершенствовать устройство и наращивать его функциональные возможности.

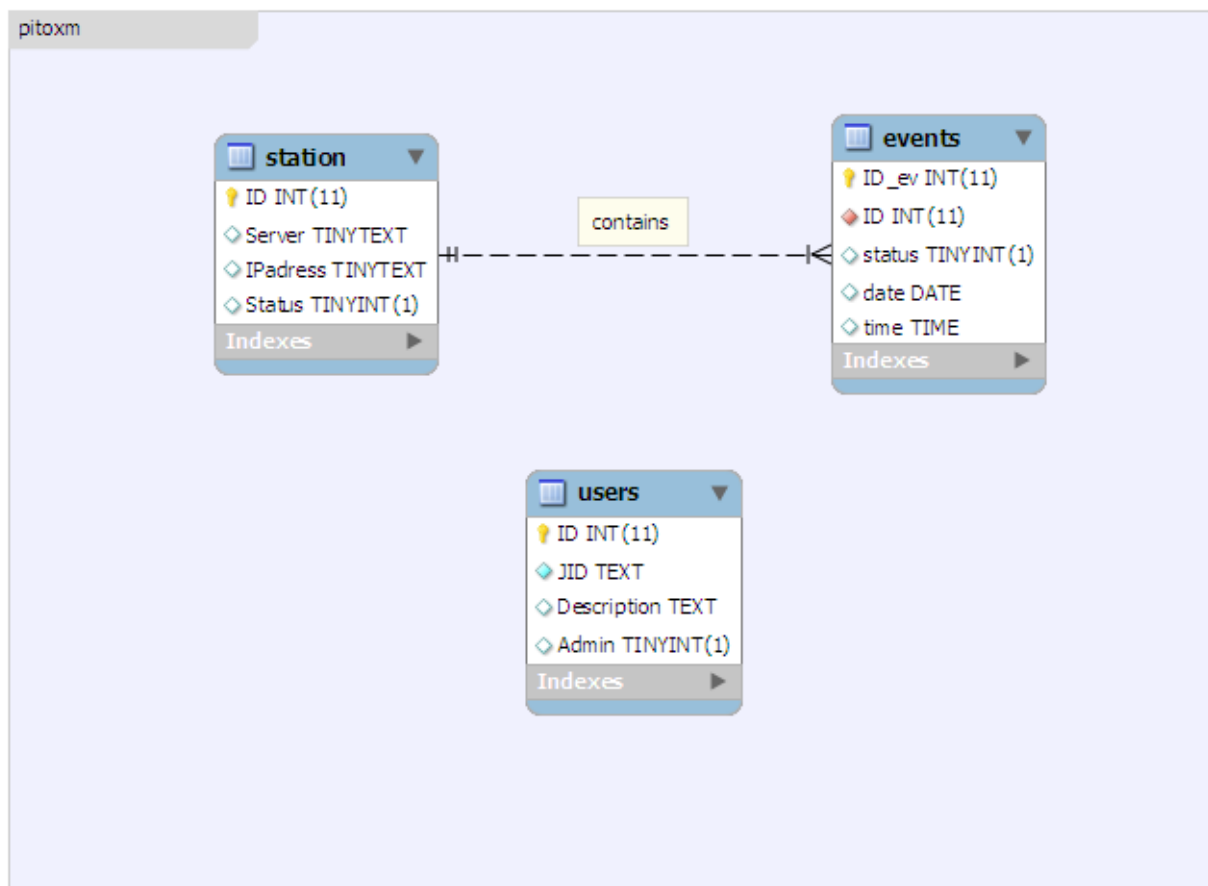


Рисунок 2.6. ER-диаграмма.

При необходимости код может быть перенесён на другие МК семейства ATmega.

Каждый блок начинается с названия файла, в котором размещён программный код данного блока. Подпрограммы обозначены блоком меньшего горизонтального размера. Среди них есть функции, т.е. подпрограммы, возвращающие значения. Для упрощения они тоже называются подпрограммами. Отличие подпрограмм, расположенных в файле ProceduresINT.asm, от подпрограмм в файле ProceduresEXT.asm

заключается в том, что первые предназначены только для данного проекта, а вторые являются универсальными и могут найти применение в других проектах. Блок Startup.asm тоже использует некоторые подпрограммы, но для упрощения восприятия рисунка эти связи на блок-схеме не показаны.

2.4 Диаграмма классов

Центральное место в объектно-ориентированном программировании занимает разработка логической модели системы в виде диаграммы классов. Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений.

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений. Диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи.

Диаграмма классов разрабатываемой системы на данный момент показан на рисунке 2.7.

1. в перечне предков QObject (или его потомок) должен быть объявлен первым
2. только один из предков может быть наследником QObject

Настоятельно рекомендуется использовать макрос Q_OBJECT при объявлении потомков QObject даже в том случае, если механизм сигналов и слотов использоваться не будет. Данный макрос используется метаобъектным компилятором для определения классов, которые нужно "дополнить" необходимым функционалом.

Объекты класса QObject (или его потомков) нельзя непосредственно копировать или передавать как значения так как это противоречит многим ключевых особенностям (иерархические структуры, привязка сигналов/слотов и т.д.). Конструктор копии и оператор присваивания QObject объявлены пустыми с помощью макроса Q_DISABLE_COPY(определен в файле qglobal.h) в секции private для предотвращения их использования или неявного создания компилятором. При создании собственных потомков QObject (или его наследников) следует также придерживаться этого правила.

NetMonitor – класс предоставляющий функции для опроса объектов и дальнейшего обрабатывания этих данных. Содержит в себе следующий функции:

Публичные:

```
NetMonitor();
```

```
int getstatus(QString host,int timereq);
```

```
~NetMonitor();
```

```
StationInfo *station;
```

```
void start();
```

```
bool LoadStationInfo();
```

```
void AddEventInBD(int ID,bool status);
```

Закрытые слоты:

```
void EndPingTimer();
```

Класс XmppBot обеспечивает функции для организации оповещений и управления системой. Содержит функции:

Публичные:

```
XmppBot();
```

```
~XmppBot();
```

```
void start();
```

Закрытые:

```
void messageProcessing(const QXmppMessage& message);
```

Публичные слоты:

```
void messageReceived(const QXmppMessage&);
```

```
void connectedtos();
```

```
void endUpdTimer();
```

2.5. Отладка программного обеспечения

Отладка системы мониторинга происходила с помощью IDE Qt Creator. Qt Creator не содержит отладчик. Он предоставляет подключаемый модуль отладчика, который действует как интерфейс между ядром Qt Creator и внешними отладчиками, такими как GNU Symbolic Debugger (gdb), Microsoft Console Debugger (CDB) и внутренний отладчик Java Script. Вы можете использовать эти отладчики для отладки языка C++.

Используйте режим Отладка в Qt Creator чтобы посмотреть состояние ваших проектов на Qt во время отладки(Рисунок 2.8).

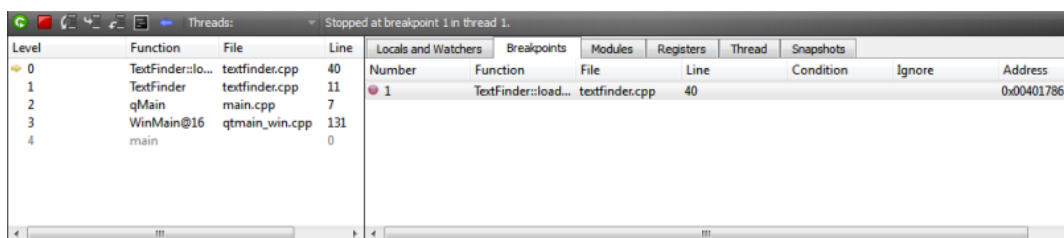


Рисунок 2.8. Окно отладчика в Qt Creator

В режиме Отладка вы можете взаимодействовать с отладчиком разными способами, включая следующие:

1. Пройти программу построчно или по каждой инструкции.
2. Прервать работающую программу.
3. Установить точки останова.
4. Просматривать содержимое стека вызовов.
5. Просматривать и изменять регистры и содержимое памяти отлаживаемого приложения.
6. Просматривать и изменять регистры и содержимое памяти локальных и глобальных переменных.
7. Просматривать список загруженных разделяемых библиотек.

Создать снимки текущего состояния отлаживаемого приложения и просмотреть их позже.

Qt Creator отображает сырую информацию, предоставляемую отладчиками, явным и лаконичным образом с целью упростить процесс отладки насколько возможно без ограничения возможностей отладчиков.

В дополнение к базовой функциональности IDE, предоставляемой просмотром стека, просмотром локальных и наблюдаемых переменных, регистров и так далее, Qt Creator имеет дополнительные особенности, делаая отладку основанных на Qt приложений проще. Подключаемый

модуль отладчика знает о внутреннем устройстве некоторых классов Qt, таких как QString, контейнеры Qt и, что особенно важно, QObject (и унаследованные от него классы), а также большинство контейнеров C++ Standard Library и некоторых расширений gcc и Symbian. Понимание их устройства используется для представления содержимого таких классов удобным способом.

Для просмотра примера того как отлаживать приложения в режиме Отладка смотрите Отладка примера приложения.

Для получения дополнительной информации о доступных в режиме Отладка функциях смотрите Взаимодействие с отладчиком.

Если вы установили Qt Creator в качестве части Qt SDK, GNU Symbolic Debugger был установлен автоматически и должен быть готов для начала отладки после создания нового проекта. Если вы хотите особую настройку, такую как использование инструментов отладки для Windows, смотрите Настройка отладчика.

Режимы действий

Подключаемый модуль отладчика работает в различных режимах в зависимости от того где и как процесс запущен и выполняется. Некоторые режимы доступны только для определённых операционных систем или платформ.

Вы можете запустить отладчик в следующих режимах:

1. Простой для отладки локально запущенных приложений, таких как GUI приложения на Qt.
2. Терминал для отладки локально запущенных процессов которым требуется консоль, обычно это приложения без GUI.

3. Подключённый для отладки локальных процессов запущенных вне Qt Creator.
4. Удалённый для отладки запущенных на другой машине процессов.
5. Ядро для отладки завершившихся аварийно процессов на Unix.
6. Post-mortem для отладки завершившихся аварийно процессов на Windows.
7. TRK для отладки процессов, запущенных на устройстве Symbian.

Когда вы нажмёте кнопку Начать отладку, отладчик запускается в подходящем режиме (простой, терминал или TRK), в зависимости от настроек сборки и запуска для активного проекта. Для запуска отладчика в других режимах воспользуйтесь пунктами меню Отладка.

Простой режим

Для запуска отладчика в простом режиме, нажмите кнопку Начать отладку для активного проекта или выберите Отладка > Начать отладку > Запустить внешнее приложение для отладки... и укажите исполняемый файл.

Режим терминала

Для запуска отладчика в режиме терминала, выберите Проекты > Настройки запуска и включите флажок Запускать в терминале. Затем нажмите кнопку Начать отладку для активного проекта.

Режим подключения

Для запуска приложения в режиме подключения, выберите Отладка > Начать отладку > Подключить к запущенному внешнему приложению..., и затем выберите процесс для подключения по имени или его ID.

Удалённый режим

Удалённый режим использует демона gdbserver, запущенного на удалённой машине.

Для запуска отладчика в удалённом режиме выберите Отладка > Начать отладку > Запустить и подключить к удалённому приложению....

Режим ядра

Режим ядра доступен только в Unix. Он отлаживает аварийно завершившиеся процессы с использованием файлов ядра (аварийные дампы), которые генерируются, если такая опция включена.

Для включения сохранения файлов ядра, введите следующую команду в командной строке в которой будет запущено приложение:

ulimit -c unlimited

Для запуска отладчика в режиме ядра выберите Отладка > Начать отладку > Подключить к дампу....

Режим Post-Mortem

Режим post-mortem доступен только на Windows где установлены инструменты отладки для Windows.

Программа установки Qt Creator спросит, хотите ли вы зарегистрировать Qt Creator как post-mortem отладчик. Для изменения этой настройки выберите Инструменты > Параметры... > Отладчик > Общее > Use Creator for post-mortem debugging.

Вы можете запустить отладчик в режиме post-mortem если приложение аварийно завершится в Windows. Нажмите в отображаемом операционной системой Windows сообщении об ошибке кнопку Debug in Qt Creator.

Режим TRK

Режим TRK это особый режим, доступный только для Symbian. Он отлаживает процессы запущенные на устройстве Symbian с использованием запущенного на устройстве приложения App TRK .

Для запуска отладчика в режиме TRK откройте проект, в качестве цели выберите Symbian и нажмите кнопку Начать отладку.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы была разработана система мониторинга, позволяющая проводить слежение как за коммутаторами, маршрутизаторами разных производителей, так и серверов различных платформ. В настоящее время система обладает следующим функционалом:

1. Мониторинг объектов посредством протокола ICMP(через утилиту ping)
2. Работа с СУБД MySQL
3. Полное управление через XMPP протокол, а так же получение уведомлений и статистики за последний день, неделю, месяц.
4. Минимальные требования к аппаратным ресурсам

В дальнейшем будут добавлены новые возможности для мониторинга и сбора информации об сетевых объектах такие как:

1. Мониторинг по службам http, smtp, ftp, telnet и прочие
2. Сбор информации по протоколу snmp
3. Агенты для операционных систем Window и Linux
4. Редакция системы для мобильных устройств

Приложение А Листинги программы

ИСХОДНЫЙ КОД ПРОГРАММЫ

Date : 12.05.2017

```
*****/
#include "90s2313.h"
#include "delay.h"
#define red PORTD.4
#define green PORTD.5
#define number_of_impuls 63
#define udervanie 1563
#define ofa_min 0x80
#define ofa_max 0xE8
#define kn_plus PIND.0;
#define kn_minus PIND.6;
unsigned char s_reg;
unsigned char ofa;
unsigned char mode;
bit infra, infra_old;
bit second_half_period;
bit ir_plus, ir_minus, ir_plus_old, ir_minus_old;
bit closed;
unsigned char num;
bit on_new, on_old, off_new, off_old;
bit enable_count;
bit error;
bit ir_p_otp, ir_m_otp;
unsigned int timer_ofa;
eeprom unsigned char pusto[1];
eeprom unsigned char plus[number_of_impuls];
eeprom unsigned char minus[number_of_impuls];
unsigned char serial[number_of_impuls];
flash unsigned char copyright[]="AMAR T0260";
unsigned int tic;
unsigned int clock;
bit second;
bit in_home;
#include "infrared.c"
void write(eeprom char *kuda)
{
    for(num=0;          num<number_of_impuls;          num++)
kuda[num]=serial[num];
    mode=7;
    num=0;
    serial[0]=0;
}
void verify(unsigned char temp)
{
    if((serial[num]<(temp-temp/4))
(serial[num]>(temp+temp/4))) error=1;
}
```

```

void infrared(void)
{
static unsigned int zader;
static unsigned char prop;
unsigned char temp;
if(zader)
{
    if(infra_old==1 && infra==1)
    {
        if(--zader)return;
        ir_minus=ir_plus=0;
        return;
    }
    zader=udervanie;
    return;
}
if(!prop)
{
    if(!infra) goto rc5_fr;
    return;
}
if(prop < 7)
{
    if(infra_old)
    {
        if(infra) goto rc5_count;
    }

else
    {
        if(!infra)
        {
rc5_count:            if(++serial[0] > 250) prop=0;
                        return;
        }

        if(serial[0] > 0x14) ++prop;
rc5_fr:    ++prop;
            serial[0]=0;
            return;
        }
    if(infra_old != infra)
    {
        ++num;
        serial[num]=0;
        return;
    }
    if(num<number_of_impuls && serial[num]<255) ++serial[num];
else
    {
        closed=1;
        prop=0;
        switch(mode)

```

```

    {
    case 5:
        write(plus);
        break;
    case 6:
        write(minus);
        break;
    default:
        error=0;
        for(num=0; num<number_of_impuls-1; num++)
            {
                temp=plus[num];
                if(temp==0xFF) break;
                verify(temp);
            }

if(!error)

        {
            ir_plus=1;
            ir_minus=0;
        }
        error=0;
        for(num=0; num<number_of_impuls-1; num++)
            {
                temp=minus[num];
                if(temp==0xFF) break;
                verify(temp);
            }
        if(!error)
            {
                ir_plus=0;
                ir_minus=1;
            }
        if(ir_plus || ir_minus) zader=udervanie;
        num=number_of_impuls;
        while(num) serial[--num]=0;
        break;
    }
}

#include "keyboard.c"
void keyboard(void)
{
#define press_time 10
static unsigned int counter;
static unsigned char press_plus;
static unsigned char press_minus;
static unsigned char press_key;
static unsigned int zader;
if(++counter<2000) return;
counter=0;
on_old=on_new;
off_old=off_new;

```

```

on_new=kn_plus;
off_new=kn_minus;
if(!on_old && !off_old && !on_new && !off_new)
{
    if(zader > press_time*3)
    {
        mode=0;
        in_home=1;
        clock=0;
    }
    else zader++;
    return;
}
if(on_old && !off_old && on_new && !off_new || (ir_minus
&& !in_home))
{
    if(zader>=press_time)
    {
        mode=1;
        in_home=0;
    }
    else zader++;
    return;
}
if(!on_old && off_old && !on_new && off_new || (ir_plus &&
!in_home))
{
    if(zader>=press_time)
    {
        mode=2;
        in_home=0;
    }
    else zader++;
    return;
}
if(on_old && !off_old && on_new && off_new && (mode!=5) &&
(mode!=6) || ir_m_otp)
{
    ir_m_otp=0;
    if(zader<press_time)
    {
        mode=3;
        in_home=0;
        ofa=ofa - (ofa - ofa_min)/4;
    }
    return;
}
if(!on_old && off_old && on_new && off_new && (mode!=5) &&
(mode!=6) || ir_p_otp)
{
    ir_p_otp=0;
    if(zader<press_time)
    {

```

```

        mode=4; //быстро включить
        in_home=0;
    }
    return;
}
//нажатие "+"
if(on_old && off_old && !on_new && off_new)
{
    enable_count=1;

    if(++press_plus>7)
    {
        mode=5;
        goto l1;
    }
    return;
}
if(on_old && off_old && on_new && !off_new)
{
    enable_count=1;
    if(++press_minus>7)
    {
        mode=6;
l1:
        clock=0;
    }
    return;
}
//
if(enable_count) ++press_key;
if(press_key>press_time*2)
{
    enable_count=0;
    press_key=0;
    press_plus=0;
    press_minus=0;
}
if(mode==1 || mode==2) mode=7;
zader=0;
}
#include "faza.c"
#pragma saveereg-
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    s_reg=SREG;
    #asm
        PUSH R30
    #endasm
    TCNT1H=0xFF;
    TCNT1L=0fa;
    TIMSK=0x82;
    #asm
        POP R30

```

```

#endasm
SREG=s_reg;
}
#pragma savereg+
#include "timer0.c"
#pragma savereg-
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
if(closed) #asm("RETI");
#asm
    PUSH R24
    PUSH R25
    PUSH R26
    PUSH R27
    PUSH R28
    PUSH R29
    PUSH R30
    PUSH R31
    IN    R30,SREG
    ORI   R30,0x80
    PUSH R30
#endasm
infra_old=infra;
infra=PINB.0;
if(ir_plus_old && !ir_plus && !in_home && (mode!=1) &&
(mode!=2)) ir_p_otp=1;
if(ir_minus_old && !ir_minus && !in_home && (mode!=1) &&
(mode!=2)) ir_m_otp=1;
ir_plus_old=ir_plus;
ir_minus_old=ir_minus;
#asm("SEI");
infrared();
keyboard();
if(++tic>=7813)
{
tic=0;
clock++; //0 to 4294967295
second=~second;
}
#asm
    POP  R30
    OUT  SREG,R30
    POP  R31
    POP  R30
    POP  R29
    POP  R28
    POP  R27
    POP  R26
    POP  R25
    POP  R24
#endasm
closed=0;
#asm

```

```

        RET
    #endasm
}
#pragma savereg+
#include "timer1.c"
#pragma savereg-
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
    //сохраняем регистры
    s_reg=SREG;
    #asm
        PUSH R30
    #endasm
    PORTB=0x00;
    #asm
        LDI R30,9
    __DELAY:
        DEC R30
        BRNE __DELAY
    #endasm
    PORTB=0b00011110;
    TCNT1H=0xFF;
    TCNT1L=0x64;
    if(second_half_period)
TIMSK=0x02;//запрещаем прерывания таймера1.
    second_half_period=1;
    #asm
        POP R30
    #endasm
    SREG=s_reg;
}
#pragma savereg+
void main(void)
{
    begin:
    #asm("cli");
    #include "init.c"
    // Input/Output Ports initialization
    // Port B initialization
    // Func0=In Func1=Out Func2=Out Func3=Out Func4=Out
Func5=In Func6=In Func7=In
    // State0=T State1=1 State2=1 State3=1 State4=1 State5=T
State6=T State7=T
    DDRB =0b00011110;
    PORTB=0b00011110;
    // Port D initialization
    // Func0=In Func1=In Func2=In Func3=In Func4=Out Func5=Out
Func6=In
    // State0=P State1=T State2=T State3=T State4=0 State5=1
State6=P
    DDRD =0b00110000;
    PORTD=0b01000001;
    // Timer/Counter 0 initialization

```

```

// Clock source: System Clock
TCCR0=0x01;
TCNT0=0x02;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x02;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 125,000 kHz
// Mode: Normal top=FFFFh
// OC1 output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1B=0x04;
TCCR1A=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1H=0x00;
OCR1L=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: Off
GIMSK=0x00;
MCUCR=0x03;
GIFR=0x40;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
//подготовкаданных
second_half_period=closed=0;
on_old=off_old=infra=infra_old=in_home=1;
mode=0;
ofa=ofa_min;
#asm("sei");
while (1)
{
    switch(mode)
    {
        case 0: #include "mode_0.c";
red=second;
green=second;
//4 часа
if(clock==28800) mode=4;//быстровключить
//еще полчаса
if(clock>32400)
{
    mode=3;//быстро погасить
    clock=0;
}
break;
        case 1: #include "mode_1.c";

```

```

if (GIMSK==0x00)
{
mode=7;
break;
}
if(++timer_ofa>7500)
{
timer_ofa=0;
if(ofa>ofa_min) ofa--;
else GIMSK=0x00;
}

break;
case 2: #include "mode_2.c";
if (GIMSK==0x00)
{
ofa=ofa_min;
GIMSK=0x40;
timer_ofa=0;
}
if(++timer_ofa>7500)
{
timer_ofa=0;
if(ofa<ofa_max) ofa++;
else mode=7;
}

break;
case 3: #include "mode_3.c";
if (GIMSK==0x00)
{
if(in_home) mode=0;
else mode=7;
break;
}
if(++timer_ofa>21000)
{
timer_ofa=0;
if(ofa>ofa_min) ofa--;
else GIMSK=0x00;
}

break;
case 4: #include "mode_4.c";
if (GIMSK==0x00)
{
ofa=ofa_min;
GIMSK=0x40;
timer_ofa=0;
}
if(++timer_ofa>330)
{
timer_ofa=0;
if(ofa<ofa_max) ofa++;
else
{

```

```

        if(in_home) mode=0;
        else mode=7;
    }
        break;
    case 5: #include "mode_5.c";
red=0;
green=second;
if(clock>20) mode=7;
        break;
    case 6: #include "mode_6.c";/
green=0;
red=second;
if(clock>20) mode=7;
        break;
    case 7: #include "mode_7.c";/
if(GIMSK==0x00) green=1;
        break;
    default:
        goto begin;
    }
#asm
    SBIS 0x16,0
    RJMP L1
    CBI 0x12,4
    RJMP L2
L1:
    SBI 0x12,4
L2:
#endasm
    }
}

```

РЕФЕРАТ

СИСТЕМА КОНТРОЛЯ СЕТЕВЫХ РЕСУРСОВ, СИСТЕМА МОНИТОРИНГА СЕТИ, КОМПЬЮТЕРНЫЕ СЕТИ, АНАЛИЗАТОР СЕТЕВЫХ ПРОТОКОЛОВ

Магистерская диссертация содержит 84 листа, 10 рисунков, 1 приложение.

Объектом исследования являются компьютерные сети.

Предметом исследования является система контроля критических сетевых ресурсов.

Цель работы состояла в разработке системы оперативно контроля критических сетевых ресурсов. В работе проведено исследование существующих систем сетевого мониторинга и обоснован выбор конкретных компонентов системы сетевого мониторинга.

В результате работы над выпускной квалификационной работой магистра была создана функционирующая система сетевого мониторинга.

Было произведено тестирование системы мониторинга, в результате которого были получены корректные результаты работы программы.

СОДЕРЖАНИЕ

РЕФЕРАТ	3
СОДЕРЖАНИЕ	4
ВВЕДЕНИЕ.....	5
1 СИСТЕМЫ СЕТЕВОГО МОНИТОРИНГА	7
1.1 Программные средства диагностики	15
1.1.1 Анализаторы протоколов	16
1.1.2 Протоколы мониторинга	21
Протокол SNMP	21
1.2 Популярные системы управления сетями	31
1.3 Популярные системы мониторинга сетей	34
1.4 Выбор системы мониторинга.....	40
2 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ	43
2.1 Выбор инструментария для разработки системы.	43
2.2 Установка необходимых компонентов.....	50
2.2.1 Установка драйвера QMYSQL	50
2.2.2 Установка библиотеки QXMPP.....	54
3 Разработка системы	55
2.3.1 Алгоритм работы системы.....	55
2.3.1.1 Алгоритм работы системы мониторинга	55
2.3.1.2 Алгоритм работы оповещения	56
2.3.2. Структура БД.....	58
2.4 Диаграмма классов	64
2.5. Отладка программного обеспечения	67
ЗАКЛЮЧЕНИЕ	73
Приложение А Листинги программы	74
ИСХОДНЫЙ КОД ПРОГРАММЫ.....	75

ВВЕДЕНИЕ

Несмотря на множество приемов и инструментов обнаружения и устранения неполадок в компьютерных сетях, «почва под ногами» сетевых администраторов все еще остается достаточно зыбкой. Компьютерные сети все чаще включают волоконно-оптические и беспроводные компоненты, наличие которых делает бессмысленным применение традиционных технологий и инструментов, предназначенных для обычных медных кабелей. Вдобавок к нему при скоростях свыше 100 Мбит/с традиционные подходы к диагностике зачастую перестают работать, даже если средой передачи является обычный медный кабель. Однако, возможно, наиболее серьезным изменением в компьютерных сетевых технологиях, с которым пришлось столкнуться администраторам, стал неизбежный переход от сетей Ethernet с разделяемой средой передачи к коммутируемым сетям, в которых в качестве коммутируемых сегментов часто выступают отдельные серверы или рабочие станции.

Правда, по мере осуществления технологических преобразований некоторые старые проблемы решились сами собой. Коаксиальный кабель, в котором выявить электротехнические неисправности всегда было труднее, чем в случае витой пары, становится редкостью в корпоративных средах. Сети Token Ring, главной проблемой которых была их несхожесть с Ethernet (а вовсе не слабость в техническом отношении), постепенно заменяются коммутируемыми сетями Ethernet. Порождающие многочисленные сообщения об ошибках протоколов сетевого уровня протоколы, такие, как SNA, DECnet и AppleTalk, замещаются протоколом IP. Сам же стек протоколов IP стал более стабильным и простым для поддержки, что доказывают миллионы клиентов и миллиарды страниц Web в Internet. Даже закоренелым противникам Microsoft приходится признать, что подключение нового клиента Windows к Internet существенно проще и

надежнее установки применявшихся ранее стеков TCP/IP сторонних поставщиков и отдельного программного обеспечения коммутируемого доступа.

Как бы многочисленные современные технологии ни затрудняли выявление неполадок и управление производительностью сетей, ситуация могла бы оказаться еще тяжелее, если бы технология ATM получила широкое распространение на уровне ПК. Свою положительную роль сыграло и то, что в конце 90-х, не успев получить признание, были отвергнуты и некоторые другие высокоскоростные технологии обмена данными, включая Token Ring с пропускной способностью 100 Мбит/с, 100VG-AnyLAN и усовершенствованные сети ARCnet. Наконец, в США был отклонен очень сложный стек протоколов OSI (который, правда, узаконен рядом правительств европейских стран).

Рассмотрим некоторые актуальные проблемы, возникающие у сетевых администраторов предприятий.

1 СИСТЕМЫ СЕТЕВОГО МОНИТОРИНГА

Иерархическая топология компьютерных сетей с магистральными каналами Gigabit Ethernet и выделенными портами коммутаторов на 10 или даже 100 Мбит/с для отдельных клиентских систем, позволила увеличить максимальную пропускную способность, потенциально доступную пользователям, как минимум в 10—20 раз. Конечно, в большинстве компьютерных сетей существуют узкие места на уровне серверов или маршрутизаторов доступа, поскольку приходящаяся на отдельного пользователя пропускная способность существенно меньше 10 Мбит/с. В связи с этим замена порта концентратора с пропускной способностью 10 Мбит/с на выделенный порт коммутатора на 100 Мбит/с для конечного узла отнюдь не всегда приводит к значительному увеличению скорости. Однако если учесть, что стоимость коммутаторов в последнее время снизилась, а на большинстве предприятий проложен кабель Категории 5,



Рисунок 1.1 - Традиционная локальная сеть с разделяемой средой передачи и анализатором протоколов

поддерживающий технологию Ethernet на 100 Мбит/с, и установлены сетевые карты, способные работать на скорости 100 Мбит/с сразу после

перезагрузки системы, то становится ясно, почему так нелегко сопротивляться искушению модернизации. В традиционной локальной сети с разделяемой средой передачи анализатор протоколов или монитор может исследовать весь трафик данного сегмента сети.

Хотя преимущество коммутируемой сети в производительности иногда почти не заметно, распространение коммутируемых архитектур имело катастрофические последствия для традиционных средств диагностики. В сильно сегментированной сети анализаторы протоколов способны видеть только одноадресный трафик на отдельном порту коммутатора, в отличие от сети прежней топологии, где они могли тщательно исследовать любой пакет в домене коллизий. В таких условиях традиционные инструменты мониторинга не могут собрать статистику по всем «диалогам», потому что каждая «переговаривающаяся» пара конечных точек пользуется, в сущности, своей собственной сетью.



Рисунок 1.2 – Коммутируемая сеть

В коммутируемой сети анализатор протоколов в одной точке может «видеть» только единственный сегмент, если коммутатор не способен зеркально отображать несколько портов одновременно.

Для сохранения контроля над сильно сегментированными сетями производители коммутаторов предлагают разнообразные средства для восстановления полной «видимости» сети, однако на этом пути остается немало трудностей. В поставляемых сейчас коммутаторах обычно поддерживается «зеркальное отображение» портов, когда трафик одного из них дублируется на ранее незадействованный порт, к которому подключается монитор или анализатор.

Однако «зеркальное отображение» обладает рядом недостатков. Во-первых, в каждый момент времени виден только один порт, поэтому выявить неполадки, затрагивающие сразу несколько портов, очень непросто. Во-вторых, зеркальное отражение может привести к снижению производительности коммутатора. В-третьих, на зеркальном порту обычно не воспроизводятся сбои физического уровня, а иногда даже теряются обозначения виртуальных локальных сетей. Наконец, во многих случаях не могут в полной мере зеркально отображаться полнодуплексные каналы Ethernet.

Частичным решением при анализе агрегированных параметров трафика является использование возможностей мониторинга агентов mini-RMON, тем более что они встроены в каждый порт большинства коммутаторов Ethernet. Хотя агенты mini-RMON не поддерживают группу объектов Capture из спецификации RMON II, обеспечивающих полнофункциональный анализ протоколов, они тем не менее позволяют оценить уровень использования ресурсов, количество ошибок и объем многоадресной рассылки.

Некоторые недостатки технологии зеркального отображения портов могут быть преодолены установкой «пассивных ответвителей», производимых, например, компанией Shomiti. Эти устройства представляют собой заранее устанавливаемые Y-коннекторы и позволяют отслеживать с помощью анализаторов протокола или другого устройства не регенерированный, а реальный сигнал.

Следующей актуальной проблемой, является проблема с особенностями оптики. Администраторы компьютерных сетей обычно используют специализированное оборудование диагностики оптических сетей только для решения проблем с оптическими кабелями. Обычное стандартное программное обеспечение управления устройствами на базе SNMP или интерфейса командной строки способно выявить проблемы на коммутаторах и маршрутизаторах с оптическими интерфейсами. И только немногие сетевые администраторы сталкиваются с необходимостью проводить диагностику устройств SONET.

Что касается волоконно-оптических кабелей, то причин для возникновения возможных неисправностей в них существенно меньше, чем в случае медного кабеля. Оптические сигналы не вызывают перекрестных помех, появляющихся от того, что сигнал одного проводника индуцирует сигнал на другом — этот фактор наиболее усложняет диагностическое оборудование для медного кабеля. Оптические кабели невосприимчивы к электромагнитным шумам и индуцированным сигналам, поэтому их не требуется располагать подальше от электромоторов лифтов и ламп дневного света, т. е. из сценария диагностики все эти переменные можно исключить.

Сила сигнала, или оптическая мощность, в данной точке на самом деле является единственной переменной, которую требуется измерить при

поиске неисправностей в оптических сетях. Если же можно определить потери сигнала на всем протяжении оптического канала, то можно будет идентифицировать практически любую проблему. Недорогие дополнительные модули для тестеров медного кабеля позволяют проводить оптические измерения.

Предприятиям, развернувшим крупную оптическую инфраструктуру и самостоятельно ее обслуживающим, может понадобиться приобрести оптический временный рефлектометр (Optical Time Domain Reflectometer, OTDR), выполняющего те же функции для оптического волокна, что и рефлектометр для медного кабеля (Time Domain Reflectometer, TDR). Прибор действует подобно радару: он посылает импульсные сигналы по кабелю и анализирует их отражения, на основании которых он выявляет повреждения в проводнике или какую-либо другую аномалию, и затем сообщает эксперту, в каком месте кабеля следует искать источник проблемы.

Хотя различные поставщики кабельных соединителей и разъемов упростили процессы терминирования и разветвления оптического волокна, для этого по-прежнему требуется некоторый уровень специальных навыков, и при разумной политике предприятие с развитой оптической инфраструктурой вынуждено будет обучать своих сотрудников. Как бы хорошо ни была проложена кабельная сеть, всегда существует возможность физического повреждения кабеля в результате какого-либо неожиданного происшествия.

При диагностике беспроводных локальных сетей стандарта 802.11b также могут возникнуть проблемы. Сама по себе диагностика, столь же проста, как и в случае сетей Ethernet на базе концентраторов, так как беспроводная среда передачи информации разделяется между всеми

обладателями клиентских радиоустройств. Компания Sniffer Technologies первой предложила решение для анализа протоколов таких сетей с пропускной способностью до 11 Мбит/с, и впоследствии большинство лидирующих поставщиков анализаторов представили аналогичные системы.

В отличие от концентратора Ethernet с проводными соединениями, качество беспроводных клиентских соединений далеко от стабильного. Микроволновые радиосигналы, используемые во всех вариантах локальной передачи, слабы и порой непредсказуемы. Даже небольшие изменения положения антенны могут серьезно сказаться на качестве соединений. Точки доступа беспроводной локальной сети снабжаются консолью управления устройствами, и это часто более действенный метод диагностики, чем посещение клиентов беспроводной сети и наблюдение за пропускной способностью и условиями возникновения ошибок с помощью портативного анализатора.

Хотя проблемы синхронизации данных и установки устройств, возникающие у пользователей персональных цифровых секретарей (PDA), более естественно соответствуют задачам группы технической поддержки, а не обязанностям сетевого администратора, нетрудно предвидеть, что в недалеком будущем многие такие устройства превратятся из отдельных вспомогательных средств, дополняющих ПК, в полноправных сетевых клиентов.

Как правило, операторы корпоративных беспроводных сетей будут (или должны) препятствовать развертыванию чрезмерно открытых систем, в которых любой пользователь, находящийся в зоне действия сети и обладающий совместимой интерфейсной картой, получает доступ к каждому информационному кадру системы. Протокол безопасности

беспроводных сетей WEP (Wired Equivalent Privacy) обеспечивает аутентификацию пользователей, гарантию целостности и шифрование данных, однако, как это обычно случается, совершенная система безопасности осложняет анализ причин сетевых неполадок. В защищенных сетях с поддержкой WEP специалисты по диагностике должны знать ключи или пароли, защищающие информационные ресурсы и контролирующие доступ в систему. При доступе в режиме приема всех пакетов анализатор протоколов сможет видеть все заголовки кадров, но содержащаяся в них информация без наличия ключей будет бессмысленной.

При диагностировании туннелированных каналов, которые многие производители называют виртуальными частными сетями с удаленным доступом, возникающие проблемы аналогичны имеющим место при анализе беспроводных сетей с шифрованием. Если трафик не проходит через туннелированный канал, то причину неисправности определить нелегко. Это может быть ошибка аутентификации, поломка на одной из конечных точек или затор в общедоступной зоне Internet. Попытка использования анализатора протоколов для выявления высокоуровневых ошибок в туннелированном трафике будет пустой тратой сил, потому что содержание данных, а также заголовки прикладного, транспортного и сетевого уровней зашифрованы. Вообще, меры, принимаемые в целях повышения уровня безопасности корпоративных сетей, обычно затрудняют выявление неисправностей и проблем производительности. Межсетевые экраны, прокси-серверы и системы выявления вторжений могут дополнительно осложнить локализацию неполадок.

Таким образом, проблема диагностики компьютерных сетей является актуальной и в конечном счете, диагностирование неисправностей является задачей управления. Для большинства

критически важных корпоративных систем, проведение продолжительных восстановительных работ не допустимо, поэтому единственным решением будет использование резервных устройств и процессов, способных взять на себя необходимые функции немедленно после возникновения сбоев. На некоторых предприятиях сети всегда имеют дополнительный резервный компонент на случай сбоя основного, т. е. $n + 2$ компонентов, где n — количество основных компонентов, необходимое для обеспечения приемлемой производительности. Если среднее время восстановления (Mean Time To Repair, MTTR) достаточно велико, то может понадобиться еще большая избыточность. Дело в том, что время устранения неисправности предсказать нелегко, а значительные затраты в течение непредсказуемого периода восстановления являются признаком плохого управления.

Для менее важных систем резервирование может оказаться экономически неоправданным, и в этом случае будет целесообразно вкладывать средства в наиболее эффективные инструменты (и в обучение персонала), чтобы максимально ускорить процесс диагностики и устранения неисправностей на предприятии. Кроме того, поддержку определенных систем можно доверить сторонним специалистам, либо привлекая их на предприятие по контракту, либо пользуясь возможностями внешних центров обработки данных, либо обращаясь к провайдерам услуг по сопровождению приложений (Application Service Providers, ASP) или провайдерам услуг управления. Помимо затрат наиболее значительным фактором, влияющим на решение об обращении к услугам сторонних организаций, можно считать уровень компетентности собственного персонала. Сетевые администраторы должны решить, не является ли некоторая конкретная функция настолько тесно связанной со специфическими задачами предприятия, что от стороннего специалиста

нельзя будет ожидать более качественного выполнения работы, чем это будет сделано силами служащих компании.

Почти сразу после того, как были развернуты первые корпоративные сети, надежность которых оставляла желать лучшего, производители и разработчики выдвинули концепцию «самовосстанавливающихся сетей». Современные сети, безусловно, надежнее, чем они были в 90-х гг., но не потому, что неполадки стали самоустраняться. Ликвидация сбоев программного обеспечения и аппаратных средств современных сетей все еще требуют вмешательства человека, и в ближайшей перспективе в таком положении дел не предвидится никаких принципиальных изменений. Методы и инструменты диагностики вполне соответствуют современной практике и технологиям, но они еще не достигли такого уровня, который позволил бы значительно сэкономить время сетевых администраторов в их борьбе с неполадками сетей и дефицитом производительности.

1.1 Программные средства диагностики

Среди программных средств диагностики компьютерных сетей, можно выделить специальные системы управления сетью (Network Management Systems) - централизованные программные системы, которые собирают данные о состоянии узлов и коммуникационных устройств сети, а также данные о трафике, циркулирующем в сети. Эти системы не только осуществляют мониторинг и анализ сети, но и выполняют в автоматическом или полуавтоматическом режиме действия по управлению сетью - включение и отключение портов устройств, изменение параметров мостов адресных таблиц мостов, коммутаторов и маршрутизаторов и т.п. Примерами систем управления могут служить популярные системы HP OpenView, SunNetManager, IBMNetView.

Средства управления системой (System Management) выполняют

функции, аналогичные функциям систем управления, но по отношению к коммуникационному оборудованию. Вместе с тем, некоторые функции этих двух видов систем управления могут дублироваться, например, средства управления системой могут выполнять простейший анализ сетевого трафика.

Экспертные системы. Этот вид систем аккумулирует человеческие знания о выявлении причин аномальной работы сетей и возможных способах приведения сети в работоспособное состояние. Экспертные системы часто реализуются в виде отдельных подсистем различных средств мониторинга и анализа сетей: систем управления сетями, анализаторов протоколов, сетевых анализаторов. Простейшим вариантом экспертной системы является контекстно-зависимая help-система. Более сложные экспертные системы представляют собой так называемые базы знаний, обладающие элементами искусственного интеллекта. Примером такой системы является экспертная система, встроенная в систему управления Spectrum компании Cabletron.

1.1.1 Анализаторы протоколов

В ходе проектирования новой или модернизации старой сети часто возникает необходимость в количественном измерении некоторых характеристик сети таких, например, как интенсивности потоков данных по сетевым линиям связи, задержки, возникающие на различных этапах обработки пакетов, времена реакции на запросы того или иного вида, частота возникновения определенных событий и других характеристик.

Для этих целей могут быть использованы разные средства и прежде всего - средства мониторинга в системах управления сетью, которые уже обсуждались ранее. Некоторые измерения на сети могут быть выполнены и встроенными в операционную систему программными измерителями,

примером тому служит компонента ОС Windows Performance Monitor. Даже кабельные тестеры в их современном исполнении способны вести захват пакетов и анализ их содержимого.

Но наиболее совершенным средством исследования сети является анализатор протоколов. Процесс анализа протоколов включает захват циркулирующих в сети пакетов, реализующих тот или иной сетевой протокол, и изучение содержимого этих пакетов. Основываясь на результатах анализа, можно осуществлять обоснованное и взвешенное изменение каких-либо компонент сети, оптимизацию ее производительности, поиск и устранение неполадок. Очевидно, что для того, чтобы можно было сделать какие-либо выводы о влиянии некоторого изменения на сеть, необходимо выполнить анализ протоколов и до, и после внесения изменения.

Анализатор протоколов представляет собой либо самостоятельное специализированное устройство, либо персональный компьютер, обычно переносной, класса Notebook, оснащенный специальной сетевой картой и соответствующим программным обеспечением. Применяемые сетевая карта и программное обеспечение должны соответствовать топологии сети (кольцо, шина, звезда). Анализатор подключается к сети точно также, как и обычный узел. Отличие состоит в том, что анализатор может принимать все пакеты данных, передаваемые по сети, в то время как обычная станция - только адресованные ей. Программное обеспечение анализатора состоит из ядра, поддерживающего работу сетевого адаптера и декодирующего получаемые данные, и дополнительного программного кода, зависящего от типа топологии исследуемой сети. Кроме того, поставляется ряд процедур декодирования, ориентированных на определенный протокол, например, IPX. В состав некоторых анализаторов может входить также экспертная система, которая может выдавать пользователю рекомендации о том, какие

эксперименты следует проводить в данной ситуации, что могут означать те или иные результаты измерений, как устранить некоторые виды неисправности сети.

Несмотря на относительное многообразие анализаторов протоколов, представленных на рынке, можно назвать некоторые черты, в той или иной мере присущие всем им:

Пользовательский интерфейс. Большинство анализаторов имеют развитый дружественный интерфейс, базирующийся, как правило, на Windows или Motif. Этот интерфейс позволяет пользователю: выводить результаты анализа интенсивности трафика; получать мгновенную и усредненную статистическую оценку производительности сети; задавать определенные события и критические ситуации для отслеживания их возникновения; производить декодирование протоколов разного уровня и представлять в понятной форме содержимое пакетов.

Буфер захвата. Буферы различных анализаторов отличаются по объему. Буфер может располагаться на устанавливаемой сетевой карте, либо для него может быть отведено место в оперативной памяти одного из компьютеров сети. Если буфер расположен на сетевой карте, то управление им осуществляется аппаратно, и за счет этого скорость ввода повышается. Однако это приводит к удорожанию анализатора. В случае недостаточной производительности процедуры захвата, часть информации будет теряться, и анализ будет невозможен. Размер буфера определяет возможности анализа по более или менее представительным выборкам захватываемых данных. Но каким бы большим ни был буфер захвата, рано или поздно он заполнится. В этом случае либо прекращается захват, либо заполнение начинается с начала буфера.

Фильтры. Фильтры позволяют управлять процессом захвата данных,

и, тем самым, позволяют экономить пространство буфера. В зависимости от значения определенных полей пакета, заданных в виде условия фильтрации, пакет либо игнорируется, либо записывается в буфер захвата. Использование фильтров значительно ускоряет и упрощает анализ, так как исключает просмотр ненужных в данный момент пакетов.

Переключатели - это задаваемые оператором некоторые условия начала и прекращения процесса захвата данных из сети. Такими условиями могут быть выполнение ручных команд запуска и остановки процесса захвата, время суток, продолжительность процесса захвата, появление определенных значений в кадрах данных. Переключатели могут использоваться совместно с фильтрами, позволяя более детально и тонко проводить анализ, а также продуктивнее использовать ограниченный объем буфера захвата.

Поиск. Некоторые анализаторы протоколов позволяют автоматизировать просмотр информации, находящейся в буфере, и находить в ней данные по заданным критериям. В то время, как фильтры проверяют входной поток на предмет соответствия условиям фильтрации, функции поиска применяются к уже накопленным в буфере данным.

Методология проведения анализа может быть представлена в виде следующих шести этапов:

1. Захват данных.
2. Просмотр захваченных данных.
3. Анализ данных.
4. Поиск ошибок. (Большинство анализаторов облегчают эту работу, определяя типы ошибок и идентифицируя станцию, от которой пришел пакет с ошибкой.)
5. Исследование производительности. Рассчитывается коэффициент

использования пропускной способности сети или среднее время реакции на запрос.

6. Подробное исследование отдельных участков сети. Содержание этого этапа конкретизируется по мере того, как проводится анализ.

Обычно процесс анализа протоколов занимает относительно немного времени - 1-2 рабочих дня.

Большинство современных анализаторов позволяют анализировать сразу несколько протоколов глобальных сетей, таких, как X.25, PPP, SLIP, SDLC/SNA, frame relay, SMDS, ISDN, протоколы мостов/маршрутизаторов (3Com, Cisco, Bay Networks и другие). Такие анализаторы позволяют измерять различные параметры протоколов, анализировать трафик в сети, преобразование между протоколами локальных и глобальных сетей, задержку на маршрутизаторах при этих преобразованиях и т. п. Более совершенные приборы предусматривают возможность моделирования и декодирования протоколов глобальных сетей, 'стрессового' тестирования, измерения максимальной пропускной способности, тестирования качества предоставляемых услуг. В целях универсальности почти все анализаторы протоколов глобальных сетей реализуют функции тестирования ЛВС и всех основных интерфейсов. Некоторые приборы способны осуществлять анализ протоколов телефонии. А самые современные модели могут декодировать и представлять в удобном варианте все семь уровней OSI. Появление АТМ привело к тому, что производители стали снабжать свои анализаторы средствами тестирования этих сетей. Такие приборы могут проводить полное тестирование сетей АТМ уровня Е-1/Е-3 с поддержкой мониторинга и моделирования. Очень важное значение имеет набор сервисных функций анализатора. Некоторые из них, например возможность удаленного управления прибором, просто незаменимы.

Таким образом, современные анализаторы протоколов WAN/LAN/DTM позволяют обнаружить ошибки в конфигурации маршрутизаторов и мостов; установить тип трафика, пересылаемого по глобальной сети; определить используемый диапазон скоростей, оптимизировать соотношение между пропускной способностью и количеством каналов; локализовать источник неправильного трафика; выполнить тестирование последовательных интерфейсов и полное тестирование ATM; осуществить полный мониторинг и декодирование основных протоколов по любому каналу; анализировать статистику в реальном времени, включая анализ трафика локальных сетей через глобальные сети.

1.1.2 Протоколы мониторинга

Протокол SNMP

SNMP (англ. Simple Network Management Protocol — простой протокол управления сетью) — это протокол управления сетями связи на основе архитектуры TCP/IP.

На основе концепции TMN в 1980—1990 гг. различными органами стандартизации был выработан ряд протоколов управления сетями передачи данных с различным спектром реализации функций TMN. К одному из типов таких протоколов управления относится SNMP. Протокол SNMP был разработан с целью проверки функционирования сетевых маршрутизаторов и мостов. Впоследствии сфера действия протокола охватила и другие сетевые устройства, такие как хабы, шлюзы, терминальные сервера, LAN Manager сервера, машины под управлением Windows NT и т.д. Кроме того, протокол допускает возможность внесения изменений в функционирование указанных устройств.

Эта технология, призвана обеспечить управление и контроль за устройствами и приложениями в сети связи путём обмена управляющей информацией между агентами, располагающимися на сетевых устройствах, и менеджерами, расположенными на станциях управления. SNMP определяет сеть как совокупность сетевых управляющих станций и элементов сети (главные машины, шлюзы и маршрутизаторы, терминальные серверы), которые совместно обеспечивают административные связи между сетевыми управляющими станциями и сетевыми агентами.

При использовании SNMP присутствуют управляемые и управляющие системы. В состав управляемой системы входит компонент, называемый агентом, который отправляет отчёты управляющей системе. По существу SNMP агенты передают управленческую информацию на управляющие системы как переменные (такие как «свободная память», «имя системы», «количество работающих процессов»).

Агент в протоколе SNMP - это обрабатывающий элемент, который обеспечивает менеджерам, размещенным на управляющих станциях сети, доступ к значениям переменных MIB, и тем самым дает им возможность реализовывать функции по управлению и наблюдению за устройством.

Программный агент - резидентная программа, выполняющая функции управления, а также собирающая статистику для передачу ее в информационную базу сетевого устройства.

Аппаратный агент - встроенная аппаратура (с процессором и памятью), в которой хранятся программные агенты.

Переменные, доступные через SNMP, организованы в иерархии. Эти иерархии и другие метаданные (такие, как тип и описание переменной) описываются Базами Управляющей Информации (Management Information

Bases (MIBs)).

На сегодня существует несколько стандартов на базы данных управляющей информации. Основными являются стандарты MIB-I и MIB-II, а также версия базы данных для удаленного управления RMON MIB. Кроме этого, существуют стандарты для специальных MIB устройств конкретного типа (например, MIB для концентраторов или MIB для модемов), а также частные MIB конкретных фирм-производителей оборудования.

Первоначальная спецификация MIB-I определяла только операции чтения значений переменных. Операции изменения или установки значений объекта являются частью спецификаций MIB-II.

Версия MIB-I (RFC 1156) определяет до 114 объектов, которые подразделяются на 8 групп:

- System - общие данные об устройстве (например, идентификатор поставщика, время последней инициализации системы).
- Interfaces - описываются параметры сетевых интерфейсов устройства (например, их количество, типы, скорости обмена, максимальный размер пакета).
- AddressTranslationTable - описывается соответствие между сетевыми и физическими адресами (например, по протоколу ARP).
- InternetProtocol - данные, относящиеся к протоколу IP (адреса IP-шлюзов, хостов, статистика об IP-пакетах).
- ICMP - данные, относящиеся к протоколу обмена управляющими сообщениями ICMP.
- TCP - данные, относящиеся к протоколу TCP (например, о

TCP-соединениях).

- UDP - данные, относящиеся к протоколу UDP (число переданных, принятых и ошибочных UDP-дейтаграмм).
- EGP - данные, относящиеся к протоколу обмена маршрутной информацией ExteriorGatewayProtocol, используемому в сети Internet (число принятых с ошибками и без ошибок сообщений).

Из этого перечня групп переменных видно, что стандарт MIB-I разрабатывался с жесткой ориентацией на управление маршрутизаторами, поддерживающими протоколы стека TCP/IP.

В версии MIB-II (RFC 1213), принятой в 1992 году, был существенно (до 185) расширен набор стандартных объектов, а число групп увеличилось до 10.

Агенты RMON

Новейшим добавлением к функциональным возможностям SNMP является спецификация RMON, которая обеспечивает удаленное взаимодействие с базой MIB.

Стандарт на RMON появился в ноябре 1991 года, когда Internet Engineering Task Force выпустил документ RFC 1271 под названием "Remote Network Monitoring Management Information Base" ("Информационная база дистанционного мониторинга сетей"). Данный документ содержал описание RMON для сетей Ethernet.

RMON — протокол мониторинга компьютерных сетей, расширение SNMP, в основе которого, как и в основе SNMP, лежит сбор и анализ информации о характере информации, передаваемой по сети. Как и в SNMP, сбор информации осуществляется аппаратно-программными

агентами, данные от которых поступают на компьютер, где установлено приложение управления сетью. Отличие RMON от своего предшественника состоит, в первую очередь, в характере собираемой информации — если в SNMP эта информация характеризует только события, происходящие на том устройстве, где установлен агент, то RMON требует, чтобы получаемые данные характеризовали трафик между сетевыми устройствами.

До появления RMON протокол SNMP не мог использоваться удаленным образом, он допускал только локальное управление устройствами. База RMON MIB обладает улучшенным набором свойств для удаленного управления, так как содержит агрегированную информацию об устройстве, что не требует передачи по сети больших объемов информации. Объекты RMON MIB включают дополнительные счетчики ошибок в пакетах, более гибкие средства анализа графических трендов и статистики, более мощные средства фильтрации для захвата и анализа отдельных пакетов, а также более сложные условия установления сигналов предупреждения. Агенты RMON MIB более интеллектуальны по сравнению с агентами MIB-I или MIB-II и выполняют значительную часть работы по обработке информации об устройстве, которую раньше выполняли менеджеры. Эти агенты могут располагаться внутри различных коммуникационных устройств, а также быть выполнены в виде отдельных программных модулей, работающих на универсальных ПК и ноутбуках (примером может служить LANalyzerHvll).

Интеллект агентов RMON позволяет им выполнять простые действия по диагностике неисправностей и предупреждению о возможных отказах - например, в рамках технологии RMON можно собрать данные о нормальном функционировании сети (т. е. выполнить так называемый baselining), а потом выставлять предупреждающие сигналы, когда режим

работы сети отклонится от baseline - это может свидетельствовать, в частности, о неполной исправности оборудования. Собрав воедино информацию, получаемую от агентов RMON, приложение управления может помочь администратору сети (находящемуся, например, за тысячи километров от анализируемого сегмента сети) локализовать неисправность и выработать оптимальный план действий для ее устранения.

Сбор информации RMON осуществляется аппаратно-программными зондами, подключаемыми непосредственно к сети. Чтобы выполнить задачу сбора и первичного анализа данных, зонд должен обладать достаточными вычислительными ресурсами и объемом оперативной памяти. В настоящее время на рынке имеются зонды трех типов: встроенные, зонды на базе компьютера, и автономные. Продукт считается поддерживающим RMON, если в нем реализована хотя бы одна группа RMON. Разумеется, чем больше групп данных RMON реализовано в данном продукте, тем он, с одной стороны, дороже, а с другой - тем более полную информацию о работе сети он предоставляет.

Встроенные зонды представляют собой модули расширения для сетевых устройств. Такие модули выпускаются многими производителями, в частности, такими крупными компаниями, как 3Com, Cabletron, Bay Networks и Cisco. (Кстати, 3Com и Bay Networks недавно приобрели компании Axon и ARMON, признанных лидеров в области разработки и производства средств управления RMON. Такой интерес к этой технологии со стороны крупнейших производителей сетевого оборудования лишний раз показывает, насколько нужным для пользователей является дистанционный мониторинг.) Наиболее естественным выглядит решение встраивать модули RMON в концентраторы, ведь именно из наблюдения за этими устройствами можно составить себе представление о работе сегмента. Достоинство таких зондов очевидно: они позволяют получать

информацию по всем основным группам данных RMON при относительно невысокой цене. Недостатком в первую очередь является не слишком высокая производительность, что проявляется, в частности, в том, что встроенные зонды часто поддерживают далеко не все группы данных RMON. Не так давно 3Com объявила о намерении выпустить поддерживающие RMON драйверы для сетевых адаптеров Etherlink III и Fast Ethernet. В результате окажется возможным собирать и анализировать данные RMON непосредственно на рабочих станциях в сети.

Зонды на базе компьютера - это просто подключенные к сети компьютеры с установленным на них программным агентом RMON. Такие зонды (к числу которых относится, например, продукт Cornerstone Agent 2.5 компании Network General) обладают более высокой производительностью, чем встроенные зонды, и поддерживают, как правило, все группы данных RMON. Они более дороги, чем встроенные зонды, но гораздо дешевле автономных зондов. Помимо этого, зонды на базе компьютера имеют довольно большой размер, что может иногда ограничивать возможности их применения.

Автономные зонды обладают наивысшей производительностью; как легко понять, это одновременно и наиболее дорогие продукты из всех описанных. Как правило, автономный зонд - это процессор (класса i486 или RISC-процессор), оснащенный достаточным объемом оперативной памяти и сетевым адаптером. Лидерами в этом секторе рынка являются компании Frontier и Hewlett-Packard. Зонды этого типа невелики по размеру и весьма мобильны - их очень легко подключать к сети и отключать от нее. При решении задачи управления сетью глобального масштаба это, конечно, не слишком важное свойство, однако если средства RMON применяются для анализа работы корпоративной сети средних размеров, то (учитывая высокую стоимость устройств) мобильность

зондов может сыграть весьма положительную роль.

Объекту RMON присвоен номер 16 в наборе объектов MIB, а сам объект RMON объединяет в соответствии с документом RFC 1271, состоит из десяти групп данных.

- Statistics - текущие накопленные статистические данные о характеристиках пакетов, количестве коллизий и т.п.

- History - статистические данные, сохраненные через определенные промежутки времени для последующего анализа тенденций их изменений.

- Alarms - пороговые значения статистических показателей, при превышении которых агент RMON посылает сообщение менеджеру. Позволяет пользователю определить ряд пороговых уровней (эти пороги могут относиться к самым разным вещам - любому параметру из группы статистики, амплитуде или скорости его изменения и многому другому), по превышении которых генерируется аварийный сигнал. Пользователь может также определить, при каких условиях превышение порогового значения должно сопровождаться аварийным сигналом - это позволит избежать генерации сигнала "по пустякам", что плохо, во-первых, потому, что на постоянно горящую красную лампочку никто не обращает внимания, а во-вторых, потому, что передача ненужных аварийных сигналов по сети приводит к излишней загрузке линий связи. Аварийный сигнал, как правило, передается в группу событий, где и определяется, что с ним делать дальше.

- Host - данных о хостах сети, в том числе и об их MAC-адресах..

- HostTopN - таблица наиболее загруженных хостов сети. Таблица N главных хостов (HostTopN) содержит список N первых хостов,

характеризующихся максимальным значением заданного статистического параметра для заданного интервала. Например, можно затребовать список 10 хостов, для которых наблюдалось максимальное количество ошибок в течение последних 24 часов. Список этот будет составлен самим агентом, а приложение управления получит только адреса этих хостов и значения соответствующих статистических параметров. Видно, до какой степени такой подход экономит сетевые ресурсы

- TrafficMatrix - статистика об интенсивности трафика между каждой парой хостов сети, упорядоченная в виде матрицы. Строки этой матрицы пронумерованы в соответствии с MAC-адресами станций - источников сообщений, а столбцы - в соответствии с адресами станций-получателей. Матричные элементы характеризуют интенсивность трафика между соответствующими станциями и количество ошибок. Проанализировав такую матрицу, пользователь легко может выяснить, какие пары станций генерируют наиболее интенсивный трафик. Эта матрица, опять-таки, формируется самим агентом, поэтому отпадает необходимость в передаче больших объемов данных на центральный компьютер, отвечающий за управление сетью.

- Filter - условия фильтрации пакетов. Признаки, по которым фильтруются пакеты, могут быть самыми разнообразными - например, можно потребовать отфильтровывать как ошибочные все пакеты, длина которых оказывается меньше некоторого заданного значения. Можно сказать, что установка фильтра соответствует как бы организации канала для передачи пакета. Куда ведет этот канал - определяет пользователь. Например, все ошибочные пакеты могут перехватываться и направляться в соответствующий буфер. Кроме того, появление пакета, соответствующего установленному фильтру, может рассматриваться как событие (event), на которое система должна реагировать заранее оговоренным образом.

- PacketCapture - условия захвата пакетов. В состав группы перехвата пакетов (packet capture) входят буфера для захвата, куда направляются пакеты, чьи признаки удовлетворяют условиям, сформулированным в группе фильтров. При этом захватываться может не пакет целиком, а, скажем, только первые несколько десятков байт пакета. Содержимое буферов перехвата можно впоследствии анализировать при помощи различных программных средств, выясняя целый ряд весьма полезных характеристик работы сети. Перестраивая фильтры на те или иные признаки, можно характеризовать разные параметры работы сети.

- Event - условия регистрации и генерации событий. В группе событий (events) определяется, когда следует отправлять аварийный сигнал приложению управления, когда - перехватывать пакеты, и вообще - как реагировать на те или иные события, происходящие в сети, например, на превышение заданных в группе alarms пороговых значений: следует ли ставить в известность приложение управления, или надо просто запротоколировать данное событие и продолжать работать. События могут и не быть связаны с передачей аварийных сигналов - например, направление пакета в буфер перехвата тоже представляет собой событие.

Данные группы пронумерованы в указанном порядке, поэтому, например, группа Hosts имеет числовое имя 1.3.6.1.2.1.16.4.

Десятую группу составляют специальные объекты протокола TokenRing.

Всего стандарт RMON MIB определяет около 200 объектов в 10 группах, зафиксированных в двух документах - RFC 1271 для сетей Ethernet и RFC 1513 для сетей TokenRing [28, с. 95].

Отличительной чертой стандарта RMON MIB является его независимость от протокола сетевого уровня (в отличие от стандартов MIB-I и

MIB-II, ориентированных на протоколы TCP/IP). Поэтому, его удобно использовать в гетерогенных средах, использующих различные протоколы сетевого уровня.

1.2 Популярные системы управления сетями

Система управления сетью (Network management system) - аппаратные и/или программные средства для мониторинга и управления узлами сети. Программное обеспечение системы управления сетью состоит из агентов, локализующихся на сетевых устройствах и передающих информацию сетевой управляющей платформе. Метод информационного обмена между управляющими приложениями и агентами на устройствах определяется протоколами.

Системы управления сетями должны обладать целым рядом качеств:

1. истинной распределенностью в соответствии с концепцией клиент-сервер,
2. масштабируемостью,
3. открытостью, позволяющей справиться с разнородным - от настольных компьютеров до мейнфреймов - оборудованием.

Первые два свойства тесно связаны. Хорошая масштабируемость достигается за счет распределенности системы управления. Распределенность означает, что система может включать несколько серверов и клиентов. Серверы (менеджерами) собирают данные о текущем состоянии сети от агентов (SNMP, CMIP или RMON), встроенных в оборудование сети, и накапливают их в своей базе данных. Клиенты представляют собой графические консоли, за которыми работают администраторы сети. Программное обеспечение клиента системы управления принимает запросы на выполнение каких-либо действий от

администратора (например, построение подробной карты части сети) и обращается за необходимой информацией к серверу. Если сервер обладает нужной информацией, то он сразу же передает ее клиенту, если нет - то пытается собрать ее от агентов.

Ранние версии систем управления совмещали все функции в одном компьютере, за которым работал администратор. Для небольших сетей или сетей с небольшим количеством управляемого оборудования такая структура оказывается вполне удовлетворительной, но при большом количестве управляемого оборудования единственный компьютер, к которому стекается информация от всех устройств сети, становится узким местом. И сеть не справляется с большим потоком данных, и сам компьютер не успевает их обрабатывать. Кроме того, большой сетью управляет обычно не один администратор, поэтому, кроме нескольких серверов в большой сети должно быть несколько консолей, за которыми работают администраторы сети, причем на каждой консоли должна быть представлена специфическая информация, соответствующая текущим потребностям конкретного администратора.

Поддержка разнородного оборудования - скорее желаемое, чем реально существующее свойство сегодняшних систем управления. К числу наиболее популярных продуктов сетевого управления относятся четыре системы: Spectrum компании CabletronSystems, OpenView фирмы Hewlett-Packard, NetView корпорации IBM и Solstice производства SunSoft - подразделения SunMicrosystems. Три компании из четырех сами выпускают коммуникационное оборудование. Естественно, что система Spectrum лучше всего управляет оборудованием компании Cabletron, OpenView - оборудованием компании Hewlett-Packard, а NetView - оборудованием компании IBM.

При построении карты сети, которая состоит из оборудования других производителей, эти системы начинают ошибаться и принимать одни устройства за другие, а при управлении этими устройствами поддерживают только их основные функции, а многие полезные дополнительные функции, которые отличают данное устройство от остальных, система управления просто не понимает и, поэтому, не может ими воспользоваться.

Для исправления этого недостатка разработчики систем управления включают поддержку не только стандартных баз MIB I, MIB II и RMON MIB, но и многочисленных частных MIB фирм-производителей. Лидер в этой области - система Spectrum, поддерживающая около 1000 баз MIB различных производителей.

Другим способом более качественной поддержки конкретной аппаратуры является использование на основе какой-либо платформы управления приложения той фирмы, которая выпускает это оборудование. Ведущие компании - производители коммуникационного оборудования - разработали и поставляют весьма сложные и многофункциональные системы управления для своего оборудования. К наиболее известным системам этого класса относятся Optivity компании BayNetworks, CiscoWorks компании CiscoSystems, Transcend компании 3Com. Система Optivity, например, позволяет производить мониторинг и управлять сетями, состоящими из маршрутизаторов, коммутаторов и концентраторов компании BayNetwork, полностью используя все их возможности и свойства. Оборудование других производителей поддерживается на уровне базовых функций управления. Система Optivity работает на платформах OpenView компании Hewlett-Packard и SunNetManager (предшественник Solstice) компании SunSoft. Однако, работа на основе какой-либо платформы управления с несколькими системами, такими как

Optivity, слишком сложна и требует, чтобы компьютеры, на которых все это будет работать, обладали очень мощными процессорами и большой оперативной памятью.

Тем не менее, если в сети преобладает оборудование от какого-либо одного производителя, то наличие приложений управления этого производителя для какой-либо популярной платформы управления позволяет администраторам сети успешно решать многие задачи. Поэтому разработчики платформ управления поставляют вместе с ними инструментальные средства, упрощающие разработку приложений, а наличие таких приложений и их количество считаются очень важным фактором при выборе платформы управления.

Открытость платформы управления зависит также от формы хранения собранных данных о состоянии сети. Большинство платформ-лидеров позволяют хранить данные в коммерческих базах данных, таких как Oracle, Ingres или Informix. Использование универсальных СУБД снижает скорость работы системы управления по сравнению с хранением данных в файлах операционной системы, но зато позволяет обрабатывать эти данные любыми приложениями, умеющими работать с этими СУБД.

1.3 Популярные системы мониторинга сетей

В Приложении А в виде таблицы показан обзор наиболее популярных систем сетевого мониторинга на сегодняшний день.

Системы мониторинга сети предоставляются как проприетарным программным обеспечением, так и программными продуктами с открытым исходным кодом.

Среди проприетарного программного обеспечения можно выделить такие системы, как:

1. 10-Страйк: Мониторинг Сети
2. Friendly Pinger

"10-Страйк: Мониторинг Сети" предназначена для мониторинга серверов и прочих сетевых устройств. Программа следит за работоспособностью сети и оповещает о неполадках. Позволяет узнать вовремя о произошедшем сбое (разрыв связи, завершение места на диске сервера, останов службы и т.п.) и устранить проблему с минимальными потерями времени.

Программа отслеживает состояние серверов, хостов, папок, файлов, баз данных, процессов и служб на компьютерах сети. Дополнительно администратор сети может использовать собственные скрипты в качестве проверок, и даже контролировать различные переменные в управляемых коммутаторах по протоколу SNMP.

Friendly Pinger - это мощное приложение для администрирования, мониторинга и инвентаризации компьютерной сети.

Возможности программы:

1. Визуализация компьютерной сети в красивой анимационной форме;
2. Отображение, какие компьютеры включены, а какие нет;
3. Пингование всех устройств за раз;
4. Оповещение в случае остановки/запуска серверов;
5. Инвентаризация программного и аппаратного обеспечения всех компьютеров в сети;
6. Слежение, кто "лазает" по Вашему компьютеру и какие файлы качает;

7. Назначение внешних команд (например, telnet, tracer, net.exe) устройствам;
8. Поиск HTTP, FTP, e-mail и других сетевых служб;
9. Отображение состояния сети на рабочем столе или Web странице;
10. Графический TraceRoute;
11. Открытие компьютеров в проводнике или Total Commander

Среди открытого программного обеспечения можно выделить следующие системы мониторинга:

1. Nagios
2. Zabbix

Nagios — программа мониторинга компьютерных систем и сетей с открытым кодом. Предназначена для наблюдения, контроля состояния вычислительных узлов и служб, оповещает администратора в том случае, если какие-то из служб прекращают (или возобновляют) свою работу.

Nagios первоначально была создана под именем Netsaint, разработана Этаном Галстадом (англ. Ethan Galstad). Он же поддерживает и развивает систему сегодня, совместно с командой разработчиков, которые занимаются как официальными, так и неофициальными плагинами.

Nagios предоставляет небывалый контроль за всеми компьютерами сети, позволяющий исправить ошибки до того, как они станут критическими, и быть уверенными в том, что вы знаете о своей сети все. Базовая структура Nagios очень проста. Один из компьютеров становится сервером Nagios, собирает информацию с указанных клиентских компьютеров и аккуратно отображает ее на web-странице. Для получения базовых сведений о клиентских компьютерах (пингуются ли они, и

работают ли SSH и HTTP) на них не нужно устанавливать никакого ПО: всю работу сделает сервер. Для получения централизованных отчетов с дополнительной информацией (например, доступным дисковым пространством или использованием процессора) на клиентские компьютеры потребуется установить ПО. При возникновении аварийных ситуаций сервер может реагировать на них путем отправки электронного письма, вывода сообщения на экран или выполнения других действий, заданных в файле скрипта.

Программа чрезвычайно гибкая, настраиваемая и мощная. Обратной стороной тут неизбежно является сложность правильной настройки. На данном уроке мы рассмотрим конфигурацию центрального сервера и одного клиентского компьютера (все остальные настраиваются точно так же) и слегка коснемся других возможностей, которые можно исследовать после инсталляции. Это непросто, но стоит приложенных усилий, так как после установки Nagios поддержки почти не понадобится.

Первоначально Nagios была разработана для работы под Linux, но она также хорошо работает и под другими ОС, такими как Sun Solaris, FreeBSD, AIX и HP-UX.

Обзор возможностей:

1. Мониторинг сетевых служб (SMTP, POP3, HTTP, NNTP, ICMP, SNMP)
2. Мониторинг состояния хостов (загрузка процессора, использование диска, системные логи) в большинстве сетевых операционных систем
3. Поддержка удаленного мониторинга через зашифрованные туннели SSH или SSL
4. Простая архитектура модулей расширений (плагинов) позволяет, используя любой язык программирования по выбору (Shell, C++, Perl,

Python, PHP, C# и другие), легко разрабатывать свои собственные способы проверки служб

5. Параллельная проверка служб
6. Возможность определять иерархии хостов сети с помощью «родительских» хостов, позволяет обнаруживать и различать хосты, которые вышли из строя, и те, которые недоступны
7. Отправка оповещений в случае возникновения проблем со службой или хостом (с помощью почты, пейджера, смс, или любым другим способом, определенным пользователем через модуль системы)
8. Возможность определять обработчики событий произошедших со службами или хостами для проактивного разрешения проблем
9. Автоматическая ротация лог-файлов
10. Возможность организации совместной работы нескольких систем мониторинга с целью повышения надёжности и создания распределенной системы мониторинга
11. Включает в себя утилиту nagiosstats, которая выводит общую сводку по всем хостам, по которым ведется мониторинг.

ZABBIX — свободная система мониторинга корпоративного класса и отслеживания статусов разнообразных сервисов компьютерной сети, серверов и сетевого оборудования, создан Алексеем Владышевым и в настоящее время активно разрабатывается и поддерживается Zabbix SIA.

Для хранения данных используется MySQL, PostgreSQL, SQLite или Oracle. Веб-интерфейс написан на PHP.

ZABBIX поддерживает несколько видов мониторинга:

1. Simple checks — может проверять доступность и реакцию стандартных сервисов, таких как SMTP или HTTP без установки какого-либо программного обеспечения на наблюдаемом хосте.

2. ZABBIX agent — может быть установлен на UNIX-подобных или Windows хостах для получения данных о нагрузке процессора, использования сети, дисковом пространстве и т. д.
3. External check — выполнение внешних программ. ZABBIX также поддерживает мониторинг через SNMP.

Zabbix предлагает:

1. автоматическое обнаружение серверов и других устройств в сети
2. распределенный мониторинг с централизованным администрированием через ВЕБ
3. поддержка обеих механизмов пуллеров и трапперов
4. серверное программное обеспечение для Linux, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, OS X
5. родные агенты с высокой производительностью (клиентское программное обеспечение для Linux, Solaris, HP-UX, AIX, FreeBSD, OpenBSD, OS X, Tru64/OSF1, Windows NT4.0, Windows 2000, Windows 2003, Windows XP, Windows Vista)
6. мониторинг без агентов
7. безопасная аутентификация пользователей
8. гибкая система прав доступа пользователей
9. Web-интерфейс
10. гибкая система уведомлений по e-mail о предопределенных событиях
11. высокоуровневый (класса “Бизнес”) вид контроля ресурсов
12. журнал аудита

Почему организации отдают предпочтение Zabbix:

1. Открытое программное обеспечение
2. агенты с высокой эффективностью для UNIX и WIN32 платформ

3. легко изучаемый
4. увеличивает рентабельность (простои очень дорого обходятся)
5. низкая стоимость обслуживания
6. очень простое конфигурирование
7. централизованная система мониторинга. Вся информация (конфигурация и данные о производительности) хранятся в реляционной базе данных
8. высокоуровневое дерево предоставляемых услуг
9. очень простая установка
10. поддержка SNMP (v1,v2,v3). Оба режима пуллера и траппера.
11. возможность визуализации

встроенный механизм очистки устаревших данных

1.4 Выбор системы мониторинга

В разрабатываемой системе нужно учесть несколько аспектов:

- Кроссплатформенность системы.
- Мобильность
- Простота конфигурации

Кроссплатформенное программное обеспечение — программное обеспечение, работающее более чем на одной аппаратной платформе и/или операционной системе. Является одним из главных аспектов, учитываемый при разработке данной системы мониторинга – возможность запуска на ОС семейства Linux, Windows, а так же мобильных платформах.

Мобильность – под этим термином подразумевается как возможность системных администраторов управлять системой в мобильном режиме(находясь не на рабочем месте), так и возможность

запуска с мобильных устройств, таких как смартфоны, карманные компьютеры, планшетные компьютеры и т.д.

Системный администратор по специфике своей работы, не всегда может находиться на своем рабочем месте и имеет место необходимость в своевременном уведомлении администратора о возникшей проблеме. Некоторые варианты такого уведомления могут реализованы по средством сотовых сетей и мобильных устройств связи: сообщения SMS, голосовые вызовы от систем мониторинга. Так же такая возможность может быть предоставлена с помощью систем мгновенного обмена сообщений – ICQ, XMPP(Jabber), Skype.

Запуск системы мониторинга с мобильных устройств обусловлена возможностью сбора кратковременных статистических данных о ресурсах сети, а так же для ее анализа и диагностики, либо невозможностью развернуть систему мониторинга стандартным способом.

Конфигурирование системы не должно занимать много времени и ресурсов системного администратора. Так же стоит заметить, что в организациях где существуют более 2х системных администраторов, может появиться необходимость настройки системы мониторинга под каждого из системных администраторов, например, первый администратор следит за состоянием серверов и оборудования в серверной А, а второй в серверной Б, и соответственно для второго администратора информационные сообщения для первого будут иметь «паразитный» характер.

Ни одна из вышеперечисленных систем мониторинга не удовлетворяет нашим требованиям, поэтому было решено разрабатывать собственную систему мониторинга.

Разрабатываемая система мониторинга будет обладать следующим функционалом:

1. Поддержка БД SQLite, MySQL. SQLite будет использоваться как в мобильной версии приложения, так и для PC платформы. MySQL будет использоваться только на PC платформе. В дальнейшем возможны реализация поддержки БД Oracle, PostgreSQL и MS SQL.
2. Простота конфигурирования. Основные настройки будут храниться в текстовом файле. Остальная конфигурация ложится на пользователя – добавление и удаления сетевых ресурсов для мониторинга.
3. Разделения на пользователей. Каждый пользователь системы будет иметь свой профиль со своими настройками для конфигурации объектов мониторинга.
4. Полное управление через XMPP протокол. Так же будет реализован web-интерфейс, который позволит просматривать статистику и конфигурировать объекты мониторинга.
5. Агенты для клиентских машин под управлением ОС Windows и Linux, для сбора информации и мониторинга состояния системы.
6. Приложение будет доступно в 2х редакциях – для мобильных платформ и для PC.
7. Будут поддерживаться следующие способы мониторинга:
 - 7.1. Мониторинг доступности посредством протокола ICMP и SNMP;
 - 7.2. Мониторинг доступности по протоколам http, smtp, telnet, ftp и прочих;
 - 7.3. Агенты для сбора информации с систем.

На данный момент система находится в стадии разработки, и в данном дипломном проекте предоставлена с минимальным функционалом: реализована поддержка БД MySQL, управление через XMPP протокол, мониторинг посредством ICMP.

2 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

2.1 Выбор инструментария для разработки системы.

В качестве среды разработки был выбран язык программирования C++ с кроссплатформенной библиотекой QT.

C++ — компилируемый статически типизированный язык программирования общего назначения.

Поддерживает разные парадигмы программирования, сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. Название «C++» происходит от названия языка C, в котором унарный оператор ++ обозначает инкремент переменной.

Являясь одним из самых популярных языков программирования, C++ широко используется для разработки программного обеспечения. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (например, видеоигры). Существует несколько реализаций языка C++ — как бесплатных, так и коммерческих. Наиболее популярны проект GNU, Microsoft, Intel и Embarcadero (Borland). C++ оказал огромное влияние на другие языки программирования, в первую очередь на Java и C#.

При создании C++ Бьёрн Страуструп стремился сохранить совместимость с языком C. Множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и

компиляторами C++, довольно велико — отчасти благодаря тому, что синтаксис C++ был основан на синтаксисе C.

Qt - кросс-платформенный инструментальный разработки ПО на языке программирования C++. Есть также «привязки» ко многим другим языкам программирования: Python - PyQt, PySide; Ruby — QtRuby ; Java — Qt Jambi; PHP PHP-Qt и другие.

Позволяет запускать написанное с его помощью ПО в большинстве современных операционных систем путём простой компиляции программы для каждой ОС без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Qt является полностью объектно-ориентированным, легко расширяемым и поддерживающим технику компонентного программирования.

Существуют версии библиотеки для Microsoft Windows, систем класса UNIX с графической подсистемой X11, iOS, Android, Mac OS X, Microsoft Windows CE, QNX, встраиваемых Linux-систем и платформы S60. Также идёт портирование на Haiku.

Qt, в частности, используются в Autodesk Maya, Adobe Photoshop Elements, OPIE, Skype, VLC media player, VirtualBox, и Mathematica, и на European Space Agency, DreamWorks, Google, HP, KDE, Lucasfilm, Panasonic, Philips, Samsung, Siemens, Volvo и Walt Disney Animation Studios.

До недавнего времени библиотека Qt также распространялась ещё в одной версии: Qt/Embedded. Теперь эта платформа переименована в *Qtopia Core* и распространяется как отдельный продукт. *Qtopia Core* обеспечивает

базовую функциональность для всей линейки платформ, предназначенных для разработки приложений для встраиваемых и мобильных устройств (КПК, смартфонов и т. п.).

Начиная с версии 4.5 Qt распространяется по 3 лицензиям:

1. Qt Commercial — для разработки ПО с собственнической лицензией, допускающая модификацию самой Qt без раскрытия изменений.
2. GNU GPL — для разработки ПО с открытыми исходниками распространяемыми на условиях GNU GPL.
3. GNU LGPL — для разработки ПО с собственнической лицензией, но без внесения изменений в Qt.

Примечание: Независимо от лицензии, исходный код Qt один и тот же.

До версии 4.0.0 под свободной лицензией распространялись лишь Qt/Mac, Qt/X11, Qt/Embedded, но, начиная с 4.0.0 (выпущенной в конце июня 2005), Qt Software «освободили» и Qt/Windows. Следует отметить, что существовали сторонние свободные версии Qt/Windows < 4.0.0, сделанные на основе Qt/X11.

Со времени своего появления в 1996 году библиотека Qt легла в основу тысяч успешных проектов во всём мире. Кроме того, Qt является фундаментом популярной рабочей среды KDE, входящей в состав многих дистрибутивов Linux.

Отличительная особенность Qt от других библиотек — использование *Meta Object Compiler (MOC)* — предварительной системы обработки исходного кода (в общем-то, Qt — это библиотека не для чистого C++, а для его особого наречия, с которого и «переводит» MOC для последующей компиляции любым стандартным C++ компилятором). MOC позволяет во много раз увеличить мощь библиотек, вводя такие

понятия, как *слоты* и *сигналы*. Кроме того, это позволяет сделать код более лаконичным. Утилита

МОС ищет в заголовочных файлах на C++ описания классов, содержащие макрос `Q_OBJECT`, и создаёт дополнительный исходный файл на C++, содержащий *мета-объектный* код.

Qt позволяет создавать собственные плагины и размещать их непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Qt комплектуется визуальной средой разработки графического интерфейса «Qt Designer», позволяющей создавать диалоги и формы «мышью» (в режиме WYSIWYG). В поставке Qt есть «Qt Linguist» — графическая утилита, позволяющая упростить локализацию и перевод вашей программы на многие языки; и «Qt Assistant» — справочная система Qt, упрощающая работу с документацией по библиотеке, а также позволяющая создавать кросс-платформенную справку для разрабатываемого на основе Qt ПО.

Начиная с версии 4.5.0 в комплект Qt включена среда разработки «Qt Creator» (ранее известная под кодовым названием Greenhouse) — кроссплатформенная свободная IDE для работы с фреймворком Qt, разработанная Trolltech (Nokia). Анонс проекта состоялся на Qt Developer Days в октябре 2008 года. Публичная бета-версия проекта была опубликована 30 октября 2008 года. Финальный релиз состоялся 3 марта 2009 года (вместе с выходом Qt 4.5), а исходный код доступен под лицензией LGPL.

Qt Creator обладает следующими особенностями:

1. Сделана специально для разработки на Qt.
2. Встроенный редактор форм (Qt Designer) и справочная система (Qt Assistant).
3. Контекстно-зависимая система помощи.
4. Расширяема плагинами.
5. Имеется графический фронтенд для GDB.
6. Поддержка отладки с помощью CDB.
7. Для создания проектов используется qmake (планируется поддержка Makefile и тестируется поддержка CMake).
8. Обобщённая подсветка синтаксиса, поддерживается большое количество языков программирования и разметки. Есть возможность создания своих стилей подсветки.
9. Возможность редактировать этапы сборки проекта.
10. Поддержка разработки на языках C/C++/QML.
11. QML-дизайнер.
12. Возможность разработки под Symbian и Maemo с отладкой в симуляторе или на устройстве.

Библиотека QT разделена на несколько модулей, для четвёртой версии библиотеки это:

1. QtCore — классы ядра библиотеки, используемые другими модулями;
2. QtGui — компоненты графического интерфейса;
3. QtNetwork — набор классов для сетевого программирования.

Поддержка различных высокоуровневых протоколов может меняться от версии к версии. В версии 4.2.x присутствуют классы для работы с протоколами FTP и HTTP. Для работы с протоколами TCP/IP предназначены такие классы, как QTcpServer, QTcpSocket для TCP и QUdpSocket для UDP;

4. QtOpenGL — набор классов для работы с OpenGL;
5. QSql — набор классов для работы с базами данных с использованием языка структурированных запросов SQL. Основные классы данного модуля в версии 4.2.x: QSqlDatabase — класс для предоставления соединения с базой, для работы с какой-нибудь конкретной базой данных требует объект, унаследованный от класса QSqlDriver — абстрактного класса, который реализуется для конкретной базы данных и может требовать для компиляции SDK базы данных. Например, для сборки драйвера под базу данных FireBird/InterBase требует .h файлы и библиотеки статической линковки, входящие в комплект поставки данной БД;
6. QtScript — классы для работы с Qt Scripts;
7. QtSvg — классы для отображения и работы с данными Scalable Vector Graphics(SVG);
8. QtXml — модуль для работы с XML, поддерживается SAX и DOM модели работы;
9. QtDesigner — классы создания расширений QtDesigner'а для своих собственных виджетов;
10. QtUiTools — классы для обработки в приложении форм Qt Designer;
11. QtAssistant — справочная система;
12. Qt3Support — модуль с классами, необходимыми для совместимости с библиотекой Qt версии 3.x.x;
13. QtTest — модуль для работы с UNIT тестами;
14. QtWebKit — модуль WebKit, интегрированный в Qt и доступный через её классы;
15. QtXmlPatterns — модуль для поддержки XQuery 1.0 и XPath 2.0;
16. Phonon — модуль для поддержки воспроизведения и записи видео и аудио, как локально, так и с устройств и по сети;

- 17.QtCLucene — модуль для поддержки полнотекстового поиска, применяется в новой версии Assistant в Qt 4.4;
- 18.ActiveQt — модуль для работы с ActiveX и COM технологиями для Qt-разработчиков под Windows.
- 19.QtDeclarative — модуль, предоставляющий декларативный фреймворк для создания динамических, настраиваемых пользовательских интерфейсов.

Также реализована технология WoC — widgets on canvas, с помощью которой реализована Plasma в KDE 4.1, Будет возможным использовать виджеты библиотеки Qt прямо в апплетах. Обеспечивает расположение виджетов на QGraphicsView с возможностью масштабирования и различных графических эффектов.

Библиотека использует собственный формат проекта, именуемый .pro файлом, в котором собрана информация о том, какие файлы будут скомпилированы, по каким путям искать заголовочные файлы и много другой информации. Впоследствии при помощи утилиты qmake из них получаются makefile для make-утилиты компилятора. Также есть возможность работы при помощи интеграторов с Microsoft Visual Studio 2003/2005/2008/2010. Совсем недавно стала доступна интеграция в Eclipse для версии библиотеки 4.x.x.

Одним из весомых преимуществ проекта Qt является наличие качественной документации, в отличие, например, от wxWidgets. Статьи документации снабжены большим количеством примеров. Исходный код самой библиотеки хорошо форматирован, подробно комментирован и легко читается, что также упрощает изучение Qt.

На Qt основана среда рабочего стола KDE, графический интерфейс мобильной ОС MeeGo и Qt Creator — среда разработки на Qt, написана с

использованием этого фреймворка. Список наиболее популярных приложений использующих QT показан в приложении Б.

Исходя из вышеуказанных достоинств проекта QT, было решено разрабатывать систему мониторинга с помощью QT SDK version 1.1.3 с Qt libraries 4.7.4 for Windows. Скачивается данное программное обеспечение с официального сайта проекта qt.nokia.com и устанавливается согласно документации.

2.2 Установка необходимых компонентов

При разработке данного проекта нам потребуются библиотеки для работы с MySQL и протоколом XMPP.

Решено было использовать нативный для QT драйвер QMYSQL для работы с БД MySQL и библиотеку QXMPP для работы с протоколом XMPP соответственно.

2.2.1 Установка драйвера QMYSQL

Qt дает возможность создания платформо-независимых приложений для работы с базами данных, используя стандартные СУБД. Qt включает «родные» драйвера для Oracle, Microsoft SQL Server, Sybase Adaptive Server, IBM DB2, PostgreSQL, MySQL и ODBC-совместимых баз данных. Qt включает специфичные для баз данных виджеты, а также поддерживает расширение для работы с базами данных любых встроенных или отдельно написанных виджетов.

Работа с базами данных в Qt происходит на различных уровнях:

1. Слой драйверов — Включает классы QSqlDriver, QSqlDriverCreator, QSqlDriverCreatorBase, QSqlDriverPlugin и QSqlResult. Этот слой

предоставляет низкоуровневый мост между определенными базами данных и слоем SQL API.

2. Слой SQL API — Этот слой предоставляет доступ к базам данных. Соединения устанавливаются с помощью класса QSqlDatabase. Взаимодействие с базой данных осуществляется с помощью класса QSqlQuery. В дополнение к классам QSqlDatabase и QSqlQuery слой SQL API опирается на классы QSqlError, QSqlField, QSqlIndex и QSqlRecord.
3. Слой пользовательского интерфейса — Этот слой связывает данные из базы данных с дата-ориентированными виджетами. Сюда входят такие классы, как QSqlQueryModel, QSqlTableModel и QSqlRelationalTableModel.

Чтобы получить доступ к базе данных с помощью QSqlQuery и QSqlQueryModel, необходимо создать и открыть одно или более соединений с базой данных.

Qt может работать со следующими базами данных (из-за несовместимости с GPL лицензией, не все плагины поставляются с Qt Open Source Edition):

1. QDB2 — IBM DB2 (версия 7.1 и выше)
2. QIBASE — Borland InterBase
3. QMYSQL — MySQL
4. QOCI — Драйвер Oracle Call Interface
5. QODBC — Open Database Connectivity (ODBC) — Microsoft SQL Server и другие ODBC-совместимые базы данных
6. QPSQL — PostgreSQL (версия 7.3 и выше)
7. QSQLITE2 — SQLite версии 2
8. QSQLITE — SQLite версии 3
9. QTDS — Драйвер Sybase Adaptive Server

Для сборки драйвера MySQL нам понадобится установочный пакет MySQL. При разработке проекта использовался MySQL Server 5.5, данный программный продукт доступен для скачивания с официального ресурса проекта www.mysql.com.

При установке MySQL необходимо выбрать Custom Setup, где выбрать установку компонента Client C API Library. (рисунок 2.1) Каталог установки располагается по адресу `c:\Servers\MySQL_55`.

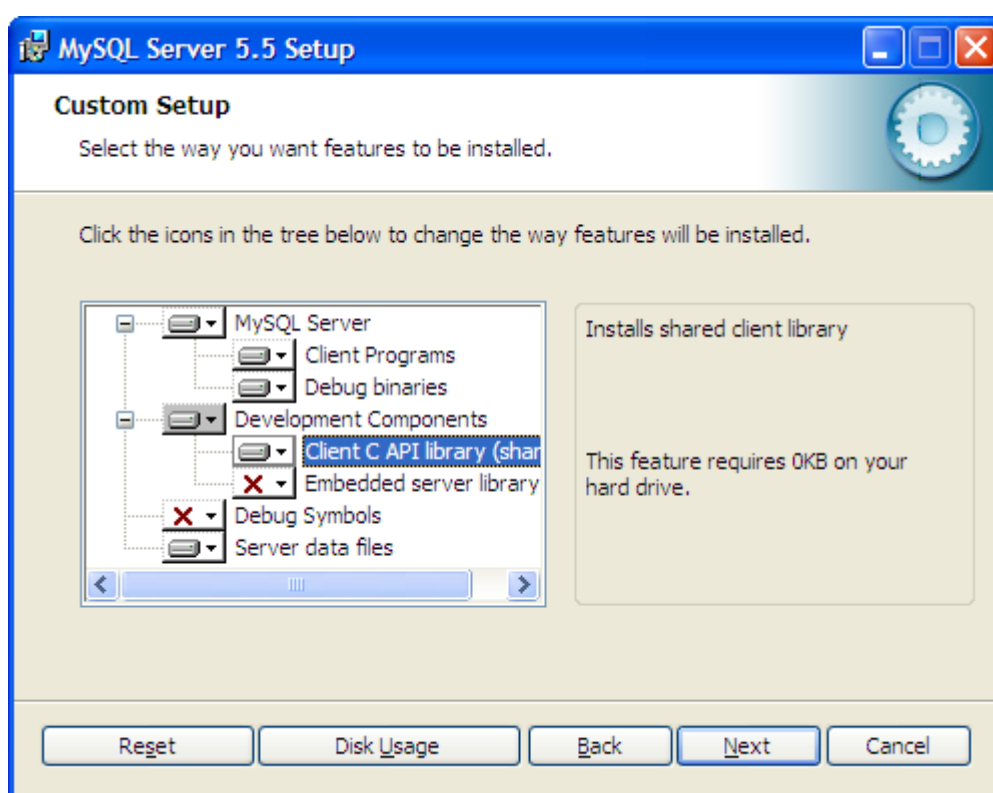


Рисунок 2.1. Установка MySQL Server 5.5

Далее необходимо войти в командную строку QT и выполнить:

```
# cd %QTDIR%\src\plugins\sqldrivers\mysql
```

```
# qmake -o Makefile "INCLUDEPATH+=C:\Servers\MySQL_55\include"  
"LIBS+= C:\Servers\MySQL_55\lib\opt\libmysql.lib" mysql.pro
```

```
# mingw32-make
```

После того, как сборка драйвера закончится, можно его будет использовать.

Для работы с БД в QT имеется модуль QtSQL, который помогает обеспечить однородную интеграцию БД в ваши Qt приложения и включает в себя:

Пространства имен:

QSql - Различные идентификаторы, используемые в разных библиотеках Qt SQL

Классы:

1. QSqlDatabase - Предоставляет соединение с базой данных
2. QSqlDriver - Абстрактный базовый класс для доступа к специфическим базам данных SQL
3. QSqlDriverCreator - Класс-шаблон, предоставляющий фабрику драйверов SQL для специфических типов драйверов
4. QSqlDriverCreatorBase - Базовый класс для фабрик драйверов SQL
5. QSqlDriverPlugin - Абстрактный базовый класс для пользовательских подключаемых модулей QSqlDriver
6. QSqlError - Информация об ошибке базы данных SQL
7. QSqlField - Управление полями в таблицах и представлениях базы данных SQL
8. QSqlIndex - Функции для управления индексами базы данных и их описания
9. QSqlQuery - Средства управления выражениями SQL и их выполнения
10. QSqlQueryModel - Модель данных только для чтения результирующей SQL-выборки
11. QSqlRecord - Заключает в себе запись базы данных

- 12. QSqlRelation - Содержит информацию о внешнем ключе SQL
- 13. QSqlRelationalDelegate - Делегат, используемый для отображения и редактирования данных из QSqlRelationalTableModel
- 14. QSqlRelationalTableModel - Модель данных для редактирования одной таблицы базы данных с поддержкой внешних ключей
- 15. QSqlResult - Абстрактный интерфейс для доступа к данным специфичной базы SQL
- 16. QSqlTableModel - Редактируемая модель данных для одной таблицы базы данных

2.2.2 Установка библиотеки QXMPP.

QXmpp – активно развивающийся проект, представляет кроссплатформенную C++ библиотеку для работы с протоколом XMPP, используя для этого TCP сокеты и XMPP RFCs (RFC3920 и RFC3921), которые были воплощены в ее функциях и классах.

Для установки QXMPP нам необходимо взять с официального сайта code.google.com/p/qxmpp/ архив `qxmpp-0.3.0.tar.gz`, на момент написания данного проекта, библиотека имела стабильную версию 0.3.0. После распаковки архива, собрать библиотеку можно двумя способами:

1. Используя командную строку: зайти в папку, где располагается файл `qxmpp.pro` и выполнить программу `mingw32-make` для сборки библиотеки и `mingw32-make install` для установки.

```
# cd c:\QT\qxmpp-0.3.0\
```

```
# mingw32-make
```

```
# mingw32-make install
```

2. Используя QT Creator – достаточно открыть библиотеку, как проект и выполнить Build All и Install.

3 Разработка системы

2.3.1 Алгоритм работы системы

На данный момент система мониторинга имеет 2 модуля: модуль мониторинга объектов и модуль взаимодействия (оповещение о событиях, управление системой) с пользователями и администраторами через протокол XMPP.

2.3.1.1 Алгоритм работы системы мониторинга

Алгоритм работы системы мониторинга показан на рисунке 2.2. После запуска системы происходит чтения конфигурационного файла, откуда берутся данные об интервале опроса объектов, данные для подключения к БД. После этого создается таймер и система ожидает его срабатывания. Затем система получает данные из таблицы station и в соответствии с этими данными начинает опрос. Результаты опроса сравниваются с последним состоянием объекта и если оно изменилось (поле status в таблице station) то делается запись в таблицу events о событии, иначе система переходит в ожидание срабатывания таймера.

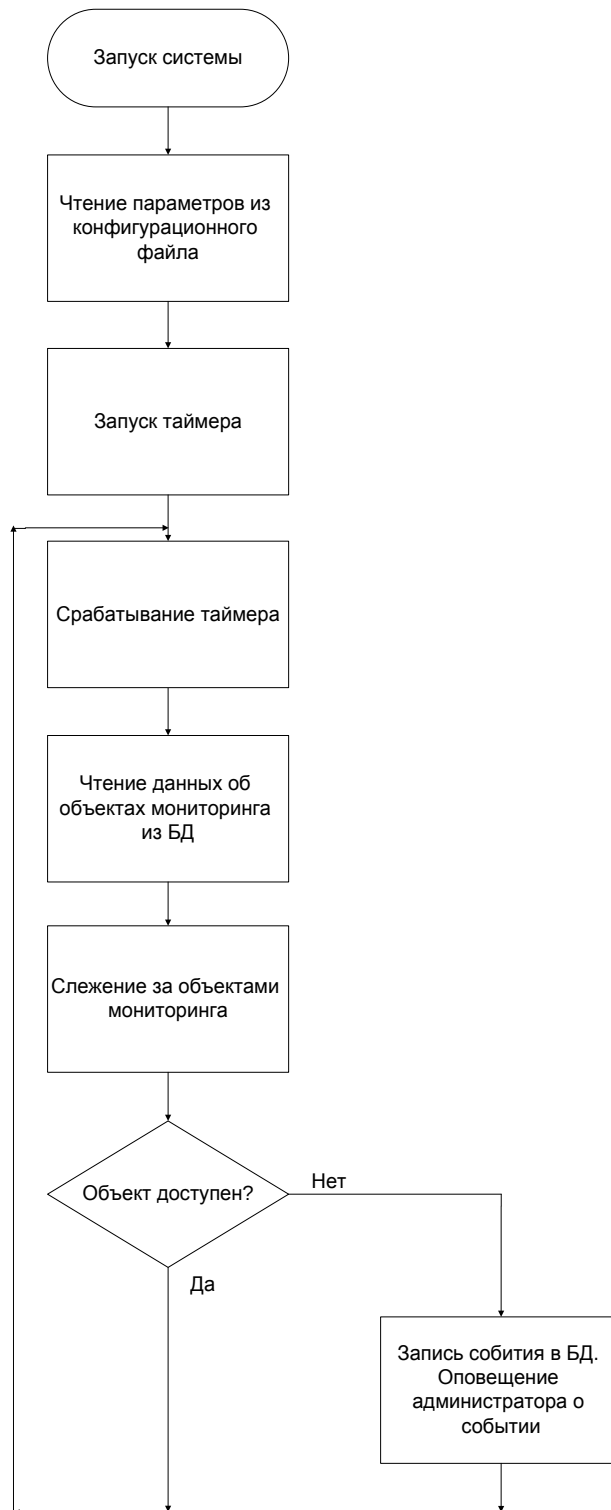


Рисунок 2.2. Алгоритм работы системы мониторинга

2.3.1.2 Алгоритм работы оповещения

Алгоритм системы оповещения показан на рисунке 2.3. После старта системы происходит чтения конфигурационного файла, откуда берется

информация необходимая для подключения к БД и подключения к Jabber серверу. Создается таймер, после срабатывания которого берутся данные из таблицы events и анализируются. Если данные изменились, то происходит уведомления пользователя о новых событиях.

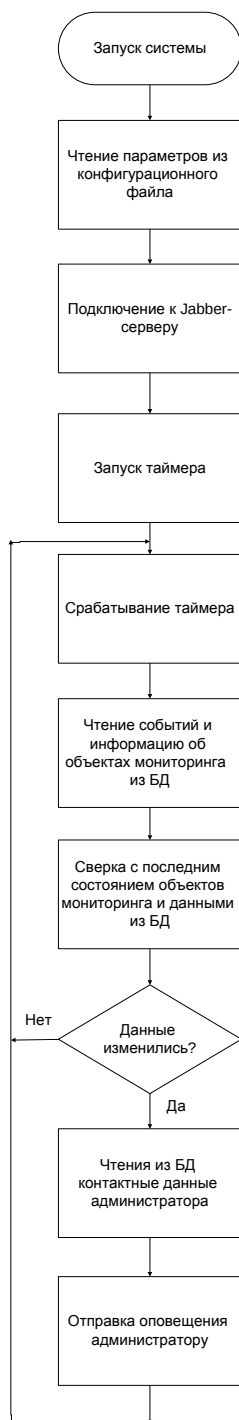


Рисунок 2.3. Алгоритм работы системы оповещения

2.3.2. Структура БД

При разработке данного проекта была создана БД pitoxm содержащая на данный момент 3 таблицы:

1. users – содержит информацию о пользователях системы(таблица 2.1).
2. station – содержит информацию об объекте мониторинга(таблица 2.2)
3. events – содержит события произошедшие с объектами мониторинга(таблица 2.3)

Таблица 2.1. Состав таблицы users

Имя поля	Тип	Описание
ID	int	Ключ. Уникальный идентификатор пользователя
JID	text	Имя учетной записи xmpp пользователя
Description	text	Дополнительное описание пользователя
Admin	bool	Флаг, показывает является ли пользователь администратором системы

Таблица 2.2. Состав таблицы station

Имя поля	Тип	Описание
ID	int	Ключ. Уникальный идентификатор объекта мониторинга

Server	text	Имя объекта мониторинга
IPadress	text	IP-адресс сервера
Status	bool	Содержит статус объекта мониторинга

Таблица 2.3. Состав таблицы events

Имя поля	Тип	Описание
ID_ev	int	Ключ. Уникальный идентификатор события
ID	int	Идентификатор объекта мониторинга
status	bool	Статус объекта мониторинга
date	date	Дата события
time	time	Время события

В настоящее время большинство проектов информационных систем (ИС) разрабатывается в соответствии с какой-либо методологией разработки ПО. Как следствие, разработчикам требуется инструмент для моделирования данных на этапах анализа и проектирования. Таким инструментом являются ER-модель (*entity-relationship model*).

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование ER-модели в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма сущность-связь (ER-диаграмма) (англ. entity-relationship diagram, ERD). Понятия ER-модель и ER-диаграмма часто ошибочно не различают, хотя для визуализации ER-моделей предложены и другие графические нотации:

1. Нотация Питера Чена. Множества сущностей изображаются в виде прямоугольников, множества отношений изображаются в виде ромбов. Если сущность участвует в отношении, они связаны линией. Если отношение не является обязательным, то линия пунктирная. Атрибуты изображаются в виде овалов и связываются линией с одним отношением или с одной сущностью. Пример нотации Питера Чена можно посмотреть на рисунке 2.4.

2. Crow's Foot. Пример отношения между сущностями согласно нотации Crow's Foot. Данная нотация была предложена Гордоном Эверестом (англ. Gordon Everest) под названием Inverted Arrow («перевернутая стрелка»), однако сейчас чаще называемая Crow's Foot («воронья лапка») или Fork («вилка»). Согласно данной нотации, сущность изображается в виде прямоугольника, содержащем её имя, выражаемое существительным. Имя сущности должно быть уникальным в рамках одной модели. При этом, имя сущности — это имя типа, а не конкретного экземпляра данного типа. Экземпляром сущности называется конкретный представитель данной сущности.

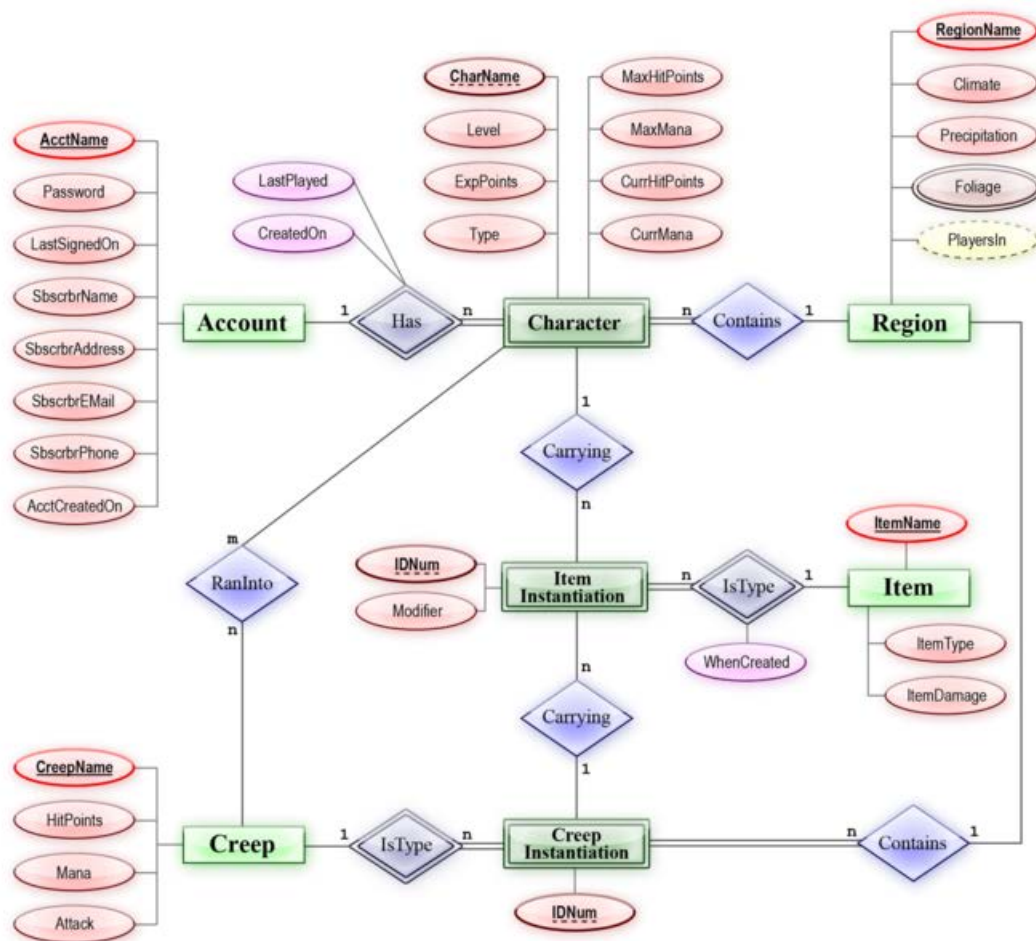


Рисунок 2.4. Простая ER-модель MMORPG использованием нотации Питера Чена

Связь изображается линией, которая связывает две сущности, участвующие в отношении(рисунок 2.5). Степень конца связи указывается графически, множественность связи изображается в виде «вилки» на конце связи. Модальность связи так же изображается графически — необязательность связи помечается кружком на конце связи. Именование обычно выражается одним глаголом в изъявительном наклонении настоящего времени: «Имеет», «Принадлежит» и т. Д.; или глаголом с поясняющими словами: «Включает в себя», и т.п. Наименование может быть одно для всей связи или два для каждого из концов связи. Во втором

случае, название левого конца связи указывается над линией связи, а правого – под линией. Каждое из названий располагается рядом с сущностью, к которой оно относится.

Атрибуты сущности записываются внутри прямоугольника, изображающего сущность и выражаются существительным в единственном числе (возможно, с уточняющими словами). Среди атрибутов выделяется ключ сущности — избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности.

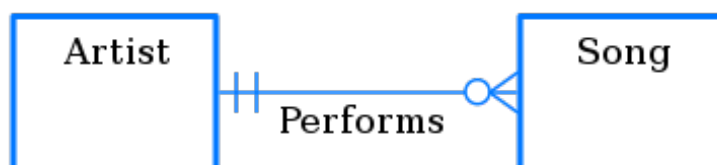


Рисунок 2.5. Изображение ER-модели Crow's Foot

Для нашей базы данных ER-диаграмма будет выглядеть следующим образом(рисунок 2.6).

Программа составлена и отлажена в бесплатной среде разработки VM Lab версии 3.14 (в окне 'About Visual Micro Lab' эта версия указана как 3.12).

Код программы написан на языке ассемблера, содержит более 1500 строк, занимает в памяти программ МК более 3 КБ. Задействованы все 32 регистра, 51 байт оперативной памяти (SRAM, ОЗУ), 45 байт энергонезависимой памяти EEPROM.

На первый взгляд, возможности МК ATmega16, имеющего 16 КБ программной памяти, 1 КБ SRAM, 512 байт EEPROM, и богатый набор периферийных устройств, кажутся избыточными. На самом деле, выбор

МК "с запасом" сделан умышленно, чтобы иметь возможность совершенствовать устройство и наращивать его функциональные возможности.

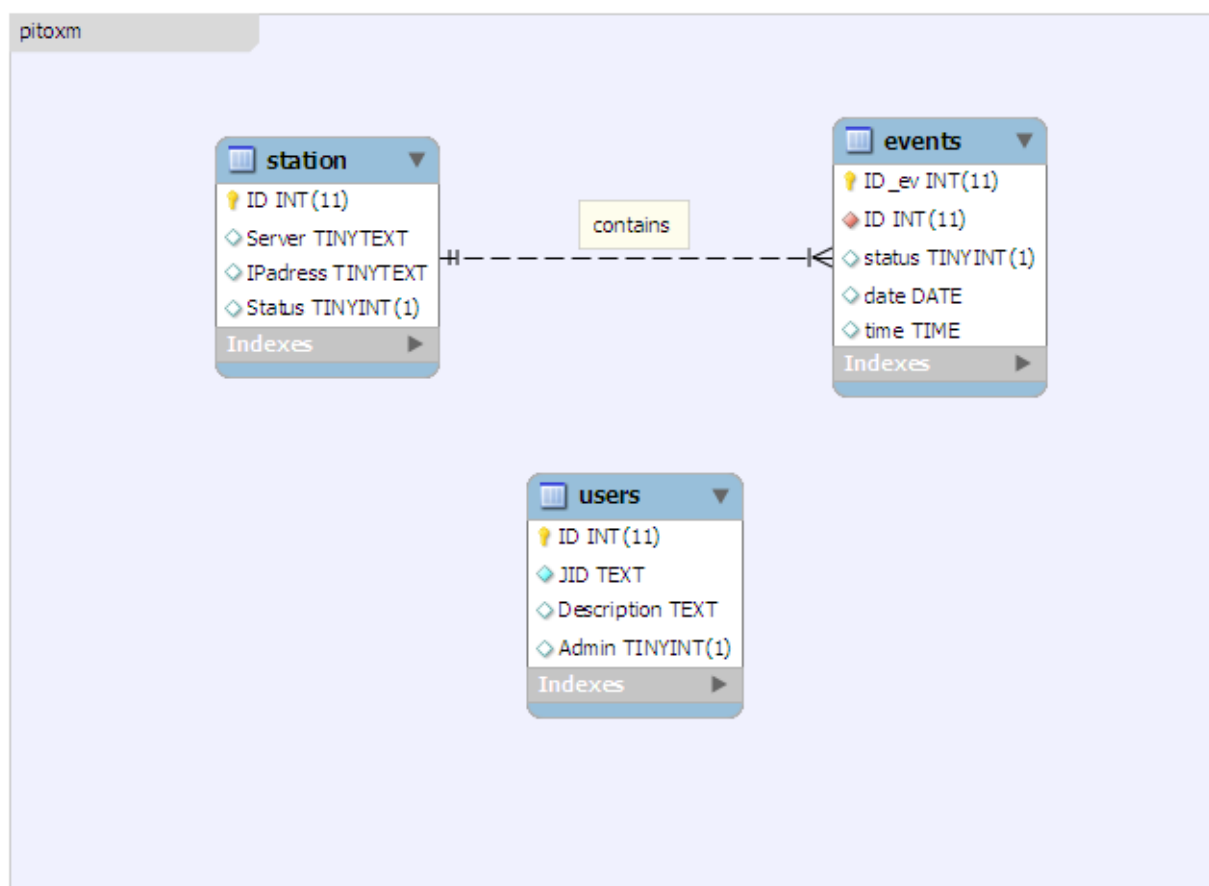


Рисунок 2.6. ER-диаграмма.

При необходимости код может быть перенесён на другие МК семейства ATmega.

Каждый блок начинается с названия файла, в котором размещён программный код данного блока. Подпрограммы обозначены блоком меньшего горизонтального размера. Среди них есть функции, т.е. подпрограммы, возвращающие значения. Для упрощения они тоже называются подпрограммами. Отличие подпрограмм, расположенных в

файле ProceduresINT.asm, от подпрограмм в файле ProceduresEXT.asm заключается в том, что первые предназначены только для данного проекта, а вторые являются универсальными и могут найти применение в других проектах. Блок Startup.asm тоже использует некоторые подпрограммы, но для упрощения восприятия рисунка эти связи на блок-схеме не показаны.

2.4 Диаграмма классов

Центральное место в объектно-ориентированном программировании занимает разработка логической модели системы в виде диаграммы классов. Диаграмма классов (class diagram) служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывать их внутреннюю структуру и типы отношений.

Диаграмма классов представляет собой граф, вершинами которого являются элементы типа «классификатор», связанные различными типами структурных отношений. Диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи.

Диаграмма классов разрабатываемой системы на данный момент показан на рисунке 2.7.

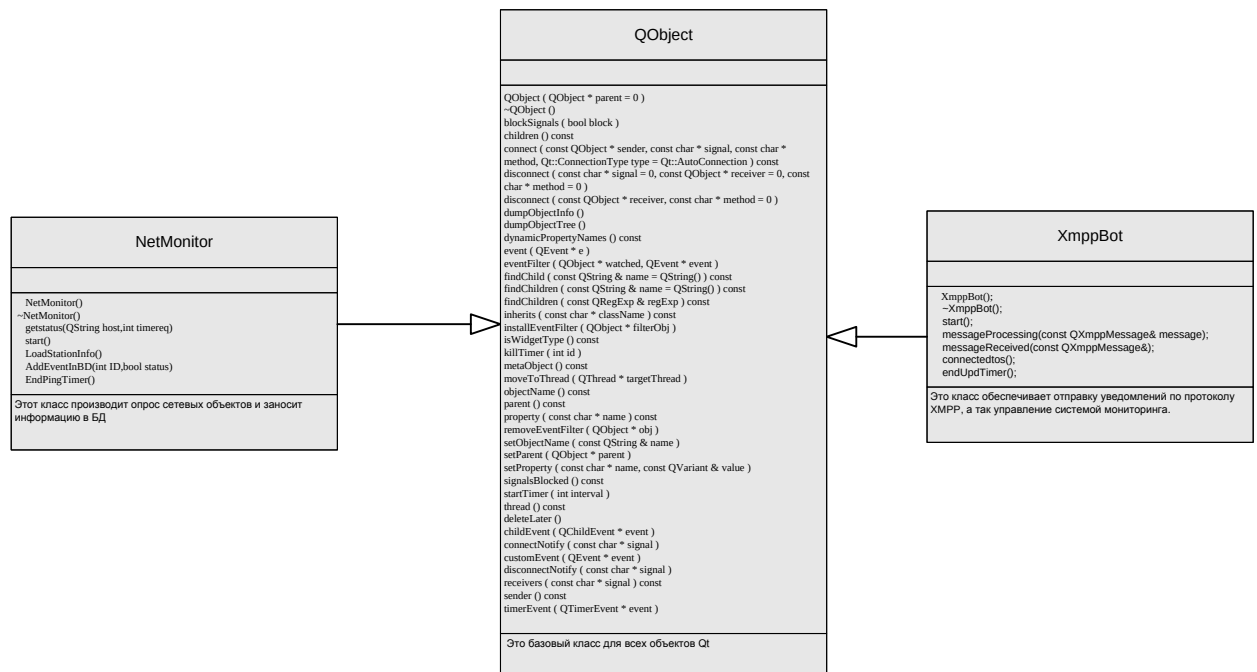


Рисунок 2.7. Диаграмма классов.

Из диаграммы видно, что в разрабатываемой системе пока имеются 2 класса: NetMonitor и XmppBot являются наследниками QObject, как и абсолютное большинство классов библиотеки QT.

Данный базовый класс содержит поддержку основополагающих механизмов, которые активно используются потомками:

1. сигналы и слоты
2. метаинформация
3. иерархии объектов
4. события, фильтры событий
5. свойства
6. приведение типов
7. таймеры

Использования QObject (или его потомка) в качестве предка при организации множественного наследования предполагает два ограничения:

1. в перечне предков QObject (или его потомок) должен быть объявлен первым
2. только один из предков может быть наследником QObject

Настоятельно рекомендуется использовать макрос Q_OBJECT при объявлении потомков QObject даже в том случае, если механизм сигналов и слотов использоваться не будет. Данный макрос используется метаобъектным компилятором для определения классов, которые нужно "дополнить" необходимым функционалом.

Объекты класса QObject (или его потомков) нельзя непосредственно копировать или передавать как значения так как это противоречит многим ключевых особенностям (иерархические структуры, привязка сигналов/слотов и т.д.). Конструктор копии и оператор присваивания QObject объявлены пустыми с помощью макроса Q_DISABLE_COPY(определен в файле qglobal.h) в секции private для предотвращения их использования или неявного создания компилятором. При создании собственных потомков QObject (или его наследников) следует также придерживаться этого правила.

NetMonitor – класс предоставляющий функции для опроса объектов и дальнейшего обрабатывания этих данных. Содержит в себе следующий функции:

Публичные:

```
NetMonitor();
```

```
int getstatus(QString host,int timereq);
```

```
~NetMonitor();
```

```
StationInfo *station;
```

```
void start();
```

```
bool LoadStationInfo();
```

```
void AddEventInBD(int ID,bool status);
```

Закрытые слоты:

```
void EndPingTimer();
```

Класс XmppBot обеспечивает функции для организации оповещений и управления системой. Содержит функции:

Публичные:

```
XmppBot();
```

```
~XmppBot();
```

```
void start();
```

Закрытые:

```
void messageProcessing(const QXmppMessage& message);
```

Публичные слоты:

```
void messageReceived(const QXmppMessage&);
```

```
void connectedtos();
```

```
void endUpdTimer();
```

2.5. Отладка программного обеспечения

Отладка системы мониторинга происходила с помощью IDE Qt Creator. Qt Creator не содержит отладчик. Он предоставляет подключаемый модуль отладчика, который действует как интерфейс между ядром Qt Creator и внешними отладчиками, такими как GNU Symbolic Debugger

(gdb), Microsoft Console Debugger (CDB) и внутренний отладчик Java Script. Вы можете использовать эти отладчики для отладки языка C++.

Используйте режим Отладка в Qt Creator чтобы посмотреть состояние ваших проектов на Qt во время отладки(Рисунок 2.8).

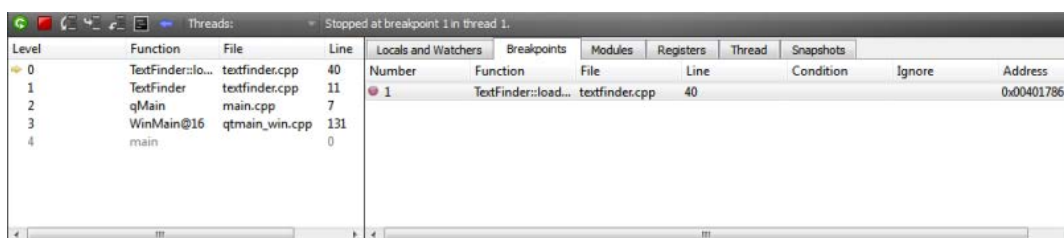


Рисунок 2.8. Окно отладчика в Qt Creator

В режиме Отладка вы можете взаимодействовать с отладчиком разными способами, включая следующие:

1. Пройти программу построчно или по каждой инструкции.
2. Прервать работающую программу.
3. Установить точки останова.
4. Просматривать содержимое стека вызовов.
5. Просматривать и изменять регистры и содержимое памяти отлаживаемого приложения.
6. Просматривать и изменять регистры и содержимое памяти локальных и глобальных переменных.
7. Просматривать список загруженных разделяемых библиотек.

Создать снимки текущего состояния отлаживаемого приложения и просмотреть их позже.

Qt Creator отображает сырую информацию, предоставляемую отладчиками, явным и лаконичным образом с целью упростить процесс отладки насколько возможно без ограничения возможностей отладчиков.

В дополнение к базовой функциональности IDE, предоставляемой просмотром стека, просмотром локальных и наблюдаемых переменных, регистров и так далее, Qt Creator имеет дополнительные особенности, делая отладку основанных на Qt приложений проще. Подключаемый модуль отладчика знает о внутреннем устройстве некоторых классов Qt, таких как QString, контейнеры Qt и, что особенно важно, QObject (и унаследованные от него классы), а также большинство контейнеров C++ Standard Library и некоторых расширений gcc и Symbian. Понимание их устройства используется для представления содержимого таких классов удобным способом.

Для просмотра примера того как отлаживать приложения в режиме Отладка смотрите Отладка примера приложения.

Для получения дополнительной информации о доступных в режиме Отладка функциях смотрите Взаимодействие с отладчиком.

Если вы установили Qt Creator в качестве части Qt SDK, GNU Symbolic Debugger был установлен автоматически и должен быть готов для начала отладки после создания нового проекта. Если вы хотите особую настройку, такую как использование инструментов отладки для Windows, смотрите Настройка отладчика.

Режимы действий

Подключаемый модуль отладчика работает в различных режимах в зависимости от того где и как процесс запущен и выполняется. Некоторые режимы доступны только для определённых операционных систем или платформ.

Вы можете запустить отладчик в следующих режимах:

1. Простой для отладки локально запущенных приложений, таких как GUI приложения на Qt.
2. Терминал для отладки локально запущенных процессов которым требуется консоль, обычно это приложения без GUI.
3. Подключённый для отладки локальных процессов запущенных вне Qt Creator.
4. Удалённый для отладки запущенных на другой машине процессов.
5. Ядро для отладки завершившихся аварийно процессов на Unix.
6. Post-mortem для отладки завершившихся аварийно процессов на Windows.
7. TRK для отладки процессов, запущенных на устройстве Symbian.

Когда вы нажмёте кнопку Начать отладку, отладчик запускается в подходящем режиме (простой, терминал или TRK), в зависимости от настроек сборки и запуска для активного проекта. Для запуска отладчика в других режимах воспользуйтесь пунктами меню Отладка.

Простой режим

Для запуска отладчика в простом режиме, нажмите кнопку Начать отладку для активного проекта или выберите Отладка > Начать отладку > Запустить внешнее приложение для отладки... и укажите исполняемый файл.

Режим терминала

Для запуска отладчика в режиме терминала, выберите Проекты > Настройки запуска и включите флажок Запускать в терминале. Затем нажмите кнопку Начать отладку для активного проекта.

Режим подключения

Для запуска приложения в режиме подключения, выберите Отладка > Начать отладку > Подключить к запущенному внешнему приложению..., и затем выберите процесс для подключения по имени или его ID.

Удалённый режим

Удалённый режим использует демона gdbserver, запущенного на удалённой машине.

Для запуска отладчика в удалённом режиме выберите Отладка > Начать отладку > Запустить и подключить к удалённому приложению....

Режим ядра

Режим ядра доступен только в Unix. Он отлаживает аварийно завершившиеся процессы с использованием файлов ядра (аварийные дампы), которые генерируются, если такая опция включена.

Для включения сохранения файлов ядра, введите следующую команду в командной строке в которой будет запущено приложение:

ulimit -c unlimited

Для запуска отладчика в режиме ядра выберите Отладка > Начать отладку > Подключить к дампу....

Режим Post-Mortem

Режим post-mortem доступен только на Windows где установлены инструменты отладки для Windows.

Программа установки Qt Creator спросит, хотите ли вы зарегистрировать Qt Creator как post-mortem отладчик. Для изменения этой настройки выберите Инструменты > Параметры... > Отладчик > Общее > Use Creator for post-mortem debugging.

Вы можете запустить отладчик в режиме post-mortem если приложение аварийно завершится в Windows. Нажмите в отображаемом операционной системой Windows сообщении об ошибке кнопку Debug in Qt Creator.

Режим TRK

Режим TRK это особый режим, доступный только для Symbian. Он отлаживает процессы запущенные на устройстве Symbian с использованием запущенного на устройстве приложения App TRK .

Для запуска отладчика в режиме TRK откройте проект, в качестве цели выберите Symbian и нажмите кнопку Начать отладку.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы была разработана система мониторинга, позволяющая проводить слежение как за коммутаторами, маршрутизаторами разных производителей, так и серверов различных платформ. В настоящее время система обладает следующим функционалом:

1. Мониторинг объектов посредством протокола ICMP(через утилиту ping)
2. Работа с СУБД MySQL
3. Полное управление через XMPP протокол, а так же получение уведомлений и статистики за последний день, неделю, месяц.
4. Минимальные требования к аппаратным ресурсам

В дальнейшем будут добавлены новые возможности для мониторинга и сбора информации об сетевых объектах такие как:

1. Мониторинг по службам http, smtp, ftp, telnet и прочие
2. Сбор информации по протоколу snmp
3. Агенты для операционных систем Window и Linux
4. Редакция системы для мобильных устройств

Приложение А Листинги программы

ИСХОДНЫЙ КОД ПРОГРАММЫ

Date : 12.05.2017

```
*****/
#include "90s2313.h"
#include "delay.h"
#define red PORTD.4
#define green PORTD.5
#define number_of_impuls 63
#define udervanie 1563
#define ofa_min 0x80
#define ofa_max 0xE8
#define kn_plus PIND.0;
#define kn_minus PIND.6;
unsigned char s_reg;
unsigned char ofa;
unsigned char mode;
bit infra, infra_old;
bit second_half_period;
bit ir_plus, ir_minus, ir_plus_old, ir_minus_old;
bit closed;
unsigned char num;
bit on_new, on_old, off_new, off_old;
bit enable_count;
bit error;
bit ir_p_otp, ir_m_otp;
unsigned int timer_ofa;
eeprom unsigned char pusto[1];
eeprom unsigned char plus[number_of_impuls];
eeprom unsigned char minus[number_of_impuls];
unsigned char serial[number_of_impuls];
flash unsigned char copyright[]="AMAR T0260";
unsigned int tic;
unsigned int clock;
bit second;
bit in_home;
#include "infrared.c"
void write(eeprom char *kuda)
{
    for(num=0; num<number_of_impuls; num++)
        kuda[num]=serial[num];
    mode=7;
    num=0;
    serial[0]=0;
}
void verify(unsigned char temp)
{
    if((serial[num]<(temp-temp/4))
(serial[num]>(temp+temp/4))) error=1;
}
```

```

void infrared(void)
{
static unsigned int zader;
static unsigned char prop;
unsigned char temp;
if(zader)
{
    if(infra_old==1 && infra==1)
    {
        if(--zader)return;
        ir_minus=ir_plus=0;
        return;
    }
    zader=udervanie;
    return;
}
if(!prop)
{
    if(!infra) goto rc5_fr;
    return;
}
if(prop < 7)
{
    if(infra_old)
    {
        if(infra) goto rc5_count;
    }

else
    {
        if(!infra)
        {
rc5_count:            if(++serial[0] > 250) prop=0;
                        return;
        }
        if(serial[0] > 0x14) ++prop;
rc5_fr:  ++prop;
        serial[0]=0;
        return;
    }
if(infra_old != infra)
{
    ++num;
    serial[num]=0;
    return;
}
if(num<number_of_impuls && serial[num]<255) ++serial[num];
else
{
    closed=1;
    prop=0;
    switch(mode)

```

```

    {
    case 5:
        write(plus);
        break;
    case 6:
        write(minus);
        break;
    default:
        error=0;
        for(num=0; num<number_of_impuls-1; num++)
        {
            temp=plus[num];
            if(temp==0xFF) break;
            verify(temp);
        }

if(!error)

        {
            ir_plus=1;
            ir_minus=0;
        }
        error=0;
        for(num=0; num<number_of_impuls-1; num++)
        {
            temp=minus[num];
            if(temp==0xFF) break;
            verify(temp);
        }
        if(!error)
        {
            ir_plus=0;
            ir_minus=1;
        }
        if(ir_plus || ir_minus) zader=udervanie;
        num=number_of_impuls;
        while(num) serial[--num]=0;
        break;
    }
}

#include "keyboard.c"
void keyboard(void)
{
#define press_time 10
static unsigned int counter;
static unsigned char press_plus;
static unsigned char press_minus;
static unsigned char press_key;
static unsigned int zader;
if(++counter<2000) return;
counter=0;
on_old=on_new;
off_old=off_new;

```

```

on_new=kn_plus;
off_new=kn_minus;
if(!on_old && !off_old && !on_new && !off_new)
{
    if(zader > press_time*3)
    {
        mode=0;
        in_home=1;
        clock=0;
    }
    else zader++;
    return;
}
if(on_old && !off_old && on_new && !off_new || (ir_minus
&& !in_home))
{
    if(zader>=press_time)
    {
        mode=1;
        in_home=0;
    }
    else zader++;
    return;
}
if(!on_old && off_old && !on_new && off_new || (ir_plus &&
!in_home))
{
    if(zader>=press_time)
    {
        mode=2;
        in_home=0;
    }
    else zader++;
    return;
}
if(on_old && !off_old && on_new && off_new && (mode!=5) &&
(mode!=6) || ir_m_otp)
{
    ir_m_otp=0;
    if(zader<press_time)
    {
        mode=3;
        in_home=0;
        ofa=ofa - (ofa - ofa_min)/4;
    }
    return;
}
if(!on_old && off_old && on_new && off_new && (mode!=5) &&
(mode!=6) || ir_p_otp)
{
    ir_p_otp=0;
    if(zader<press_time)
    {

```

```

        mode=4; //быстро включить
        in_home=0;
    }
    return;
}
//нажатие "+"
if(on_old && off_old && !on_new && off_new)
{
    enable_count=1;

if(++press_plus>7)
    {
        mode=5;
        goto l1;
    }
    return;
}
if(on_old && off_old && on_new && !off_new)
{
    enable_count=1;
    if(++press_minus>7)
    {
        mode=6;
l1:
        clock=0;
    }
    return;
}
//
if(enable_count) ++press_key;
if(press_key>press_time*2)
{
    enable_count=0;
    press_key=0;
    press_plus=0;
    press_minus=0;
}
if(mode==1 || mode==2) mode=7;
zader=0;
}
#include "faza.c"
#pragma save reg-
interrupt [EXT_INT0] void ext_int0_isr(void)
{
    s_reg=SREG;
    #asm
        PUSH R30
    #endasm
    TCNT1H=0xFF;
    TCNT1L=0fa;
    TIMSK=0x82;
    #asm
        POP R30

```

```

#endasm
SREG=s_reg;
}
#pragma savereg+
#include "timer0.c"
#pragma savereg-
interrupt [TIM0_OVF] void timer0_ovf_isr(void)
{
if(closed) #asm("RETI");
#asm
    PUSH R24
    PUSH R25
    PUSH R26
    PUSH R27
    PUSH R28
    PUSH R29
    PUSH R30
    PUSH R31
    IN    R30,SREG
    ORI   R30,0x80
    PUSH R30
#endasm
infra_old=infra;
infra=PINB.0;
if(ir_plus_old && !ir_plus && !in_home && (mode!=1) &&
(mode!=2)) ir_p_otp=1;
if(ir_minus_old && !ir_minus && !in_home && (mode!=1) &&
(mode!=2)) ir_m_otp=1;
ir_plus_old=ir_plus;
ir_minus_old=ir_minus;
#asm("SEI");
infrared();
keyboard();
if(++tic>=7813)
{
tic=0;
clock++;//0 to 4294967295
second=~second;
}
#asm
    POP  R30
    OUT  SREG,R30
    POP  R31
    POP  R30
    POP  R29
    POP  R28
    POP  R27
    POP  R26
    POP  R25
    POP  R24
#endasm
closed=0;
#asm

```

```

        RET
    #endasm
}
#pragma savereg+
#include "timer1.c"
#pragma savereg-
interrupt [TIM1_OVF] void timer1_ovf_isr(void)
{
    //сохраняем регистры
    s_reg=SREG;
    #asm
        PUSH R30
    #endasm
    PORTB=0x00;
    #asm
        LDI R30,9
    __DELAY:
        DEC R30
        BRNE __DELAY
    #endasm
    PORTB=0b00011110;
    TCNT1H=0xFF;
    TCNT1L=0x64;
    if(second_half_period)
TIMSK=0x02;//запрещаем прерывания таймера1.
    second_half_period=1;
    #asm
        POP R30
    #endasm
    SREG=s_reg;
}
#pragma savereg+
void main(void)
{
    begin:
    #asm("cli");
    #include "init.c"
    // Input/Output Ports initialization
    // Port B initialization
    // Func0=In Func1=Out Func2=Out Func3=Out Func4=Out
Func5=In Func6=In Func7=In
    // State0=T State1=1 State2=1 State3=1 State4=1 State5=T
State6=T State7=T
    DDRB =0b00011110;
    PORTB=0b00011110;
    // Port D initialization
    // Func0=In Func1=In Func2=In Func3=In Func4=Out Func5=Out
Func6=In
    // State0=P State1=T State2=T State3=T State4=0 State5=1
State6=P
    DDRD =0b00110000;
    PORTD=0b01000001;
    // Timer/Counter 0 initialization

```

```

// Clock source: System Clock
TCCR0=0x01;
TCNT0=0x02;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x02;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: 125,000 kHz
// Mode: Normal top=FFFFh
// OC1 output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
TCCR1B=0x04;
TCCR1A=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
OCR1H=0x00;
OCR1L=0x00;
// External Interrupt(s) initialization
// INT0: On
// INT0 Mode: Rising Edge
// INT1: Off
GIMSK=0x00;
MCUCR=0x03;
GIFR=0x40;
// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
// Analog Comparator Output: Off
ACSR=0x80;
//подготовкаданных
second_half_period=closed=0;
on_old=off_old=infra=infra_old=in_home=1;
mode=0;
ofa=ofa_min;
#asm("sei");
while (1)
{
    switch(mode)
    {
        case 0: #include "mode_0.c";
red=second;
green=second;
//4 часа
if(clock==28800) mode=4;//быстровключить
//еще полчаса
if(clock>32400)
{
    mode=3;//быстро погасить
    clock=0;
}
break;
        case 1: #include "mode_1.c";

```

```

if(GIMSK==0x00)
{
mode=7;
break;
}
if(++timer_ofa>7500)
{
timer_ofa=0;
if(ofa>ofa_min) ofa--;
else GIMSK=0x00;
}
break;
case 2: #include "mode_2.c";
if(GIMSK==0x00)
{
ofa=ofa_min;
GIMSK=0x40;
timer_ofa=0;
}
if(++timer_ofa>7500)
{
timer_ofa=0;
if(ofa<ofa_max) ofa++;
else mode=7;
}
break;
case 3: #include "mode_3.c";
if(GIMSK==0x00)
{
if(in_home) mode=0;
else mode=7;
break;
}
if(++timer_ofa>21000)
{
timer_ofa=0;
if(ofa>ofa_min) ofa--;
else GIMSK=0x00;
}
break;
case 4: #include "mode_4.c";
if(GIMSK==0x00)
{
ofa=ofa_min;
GIMSK=0x40;
timer_ofa=0;
}
if(++timer_ofa>330)
{
timer_ofa=0;
if(ofa<ofa_max) ofa++;
else
{

```

```

        if(in_home) mode=0;
        else mode=7;
    }
        break;
    case 5: #include "mode_5.c";
red=0;
green=second;
if(clock>20) mode=7;
        break;
    case 6: #include "mode_6.c";/
green=0;
red=second;
if(clock>20) mode=7;
        break;
    case 7: #include "mode_7.c";/
if(GIMSK==0x00) green=1;
        break;
    default:
        goto begin;
}
#asm
    SBIS 0x16,0
    RJMP L1
    CBI 0x12,4
    RJMP L2
L1:
    SBI 0x12,4
L2:
#endasm
}
}

```