ISBN 5-94052-066-0 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ. ПЕРЕСЛАВЛЬ-ЗАЛЕССКИЙ, 2004

УДК 681.324

Е. В. Ермилова, А. В. Карлаш, А. С. Нестеров, П. Г. Жбанов, Ю. В. Шевчук

Nadmin — система администрирования для региональных сетей

Аннотация. В течение предыдущих пяти лет администрирование городской сети города Переславля-Залесского и расчеты со всеми абонентами обеспечивались при помощи системы Nadmin. Система зарекомендовала себя надежной и удобной в эксплуатации. Однако за пять лет прогресс в программных и апаратных средствах достиг новых рубежей, а также было накоплено множество пожеланий к системе. Все это привело к необходимости разработки следующей версии системы. В данной работе описываются механизмы и методы реализации новой версии системы администрирования Nadmin.

Ключевые слова и фразы: Административная система, региональные сети, биллинговая система.

1. Введение

В системе телекоммуникаций (СТ) «Ботик» для управления системой и введения взаиморасчетов с абонентами используется административная система Nadmin. Система была реализована в 1997 году. За пять лет эксплуатации она показала себя с самой лучшей стороны, однако, был накоплен ряд пожеланий как со стороны абонентов, так и со стороны обслуживающего персонала, которые невозможно было удовлетворить внесением изменений в старую версию системы Nadmin. В настоящее время для устранения этих недостатков реализуется новая версия административной системы Nadmin.

Nadmin реализуется на языке программирования perl, так как он представляет собой мощный и удобный механизм для работы со многими программами и системами в ОС Linux (Debian), такими как:

Работа выполнялась при частичной поддержке суперкомпьютерной программы «СКИФ» Союзного государства и программы фундаментальных научных исследований ОИТВС РАН «Новые физические и структурные решения в инфотелекоммуникациях».

grep, tcpdump, rcs, chmod, ssh и т. д., а также для работы с сетевыми ресурсами. Новая версия административной системы реализуется, как для выполнения следующих функций:

(1) автоматизация деятельности администратора сети;
(2) учет трафика в реальном времени (real-time) каждого абонента;
(3) предоставление абонентам возможности управления собственными услугами;
(4) поддержка новых услуг, таких как: IP-телефония, веб-хостинг;
(5) возможность создания тарифных планов,

так и для удовлетворения следующих экономических потребностей:

(1) необходимость поддержания конкурентной способности СТ «Ботик» на рынке высоких технологий;
(2) повышение привлекательности СТ «Ботик» среди абонентов (этот аспект в бизнесе является одним из основных источников успеха и динамического развития);
(3) перенос рутинных операций с персонала СТ «Ботик» на абонентов, что позволит не увеличивать персонал СТ «Ботик», тем самым снизит эксплуатационные расходы;
(4) снижение расходов абонентов СТ «Ботик» (как за счет снижения эксплуатационных расходов, так и за счет предоставления возможностей управления своими расходами).

Сегодня очень простые запросы, которые абонент мог бы исполнить сам, выглядят как электронные письма с просьбой на адрес группы технической поддержки. Это отвлекает обслуживающий персонал от более важных дел и вынуждает его заниматься рутинными задачами, хотя с этой работой мог бы справиться сам абонент.

Тем самым становится актуальным вопрос о создании новой административной системы, которая обладала бы всей функциональностью для решения выше перечисленных задач. В новой версии административной системы Nadmin особое внимание было уделено дизайну и гибкости системы и возможности дальнейшего расширения без кардинальной перестройки всей системы.

Основная цель данной работы — создать новую административную систему Nadmin. Для этого предстоит решить следующие задачи:

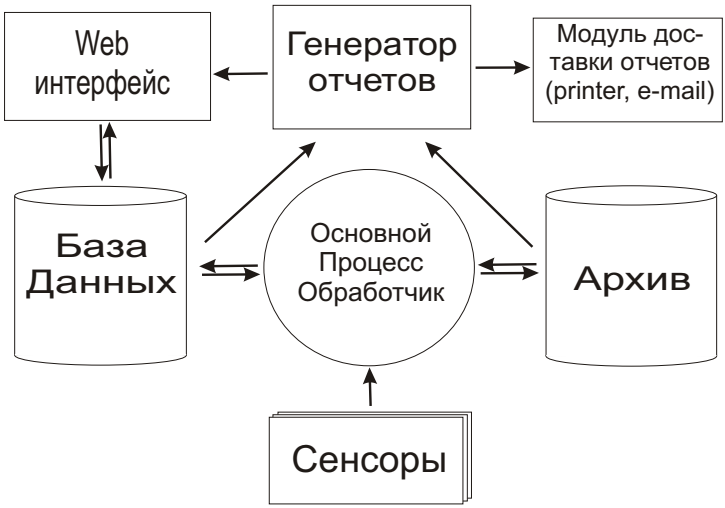

Рис. 1. Архитектура системы Nadmin

(1) исследовать и реализовать наиболее эффективную и удобную базу данных;
(2) исследовать и реализовать наиболее эффективные расчетную подсистему и административную подсистему;
(3) исследовать и реализовать веб-интерфейсы (пользовательский и администраторский).

2. Архитектура системы

Система Nadmin состоит из подсистем, которые перечисленны ниже.

База данных (БД). БД используется для хранения следующих объектов: абоненты, подключения и т. д. В системе Nadmin реализована собственная БД с использованием пакета RCS (подробнее см. раздел 4).

Архив. Архив исполузуется для долгосрочного хранения информации. В нем поддержана возможность сжатия данных, что позволяет хранить большие объёмы информации (подробнее см. раздел 8).

Сенсоры. Понятие сенсоров было введено для того, чтобы можно было легко добавлять в систему поддержку новых услуг. Сенсоры срабатывают от событий вида «оказана услуга *Service* некоторому потребителю с идентификатором *Id*» и формирует *порции статистики* для обработки *основным процессом-обработчиком* (подробнее см. раздел 3).

Основной процесс-обработчик (ОПО). ОПО является центральным процессом системы, который выполняет сбор статистики от сенсоров, расчет стоимости оказанной услуги, модификацию лицевых счетов абонентов и архивацию полученной статистики в архив (подробнее см. раздел 6).

Генератор отчетов. Генератор отчетов по запросу, используя информацию из базы данных и из архива строит отчеты следующих видов:

- справка о состоянии лицевого счета абонента;
- перечень услуг, предоставляемых абоненту;
- технические сведения о той или иной услуге;
- общие сведения о состоянии системы;
- и др.

Сгенерированные отчеты бывают трёх видов: web-страница, печатная форма, отчет для отправки по e-mail (подробнее см. раздел 7).

Web-интерфейс. Web-интерфейс используется для настройки системы и получения части отчетов (подробнее см. раздел 7).

3. Сенсоры

3.1. Общие принципы реализации сенсоров. Для реализации статистической подсистемы в структуру системы Nadmin введено понятие сенсоров. Сенсор является обработчиком *сырой*[1] статистики.

[1]Сырая статистика появляется из различных источников из «внешнего» (по отношению к Nadmin) мира, в момент, когда абоненту оказывается та или иная услуга.

Сенсор берет сырую статистику от конкретного источника, обрабатывает ее и отдает ОПО. Обычно естественным источником статистической информации по использованию ресурсов пользователями через различные серверы системы являются лог-файлы этих серверов. Например, proxy-сервер Squid формирует свой стандартный лог-файл, обычно расположенный по адресу `/var/log/squid/access.log`, а соответствующий сенсор берет из него данные, которые являются для него сырой статистикой.

Таким образом, каждый сенсор умеет разбираться в конкретном формате сырой статистики, то есть он анализирует и сортирует поток сырой статистики из лог-файлов.

Обычно в ходе одного такта работы сенсора анализируется одна (очередная) строка из лог-файла. В ходе этого анализа сенсор определяет потребителя ресурса и объем потребленной информации. В конце сенсор формирует фрагмент детальной статистики для каждого потребителя (если было несколько потребителей ресурса), считает суммарный объем востребованного ресурса по каждому фрагменту детальной статистики и выносит эту информацию в заголовок фрагмента детальной статистики. Вместе заголовок и фрагмент детальной статистики образуют порцию статистики (см. рис. 2).

Далее порции статистики через поток (UNIX-pipe)[2] передаются для ОПО. За счет заголовка порции статистики отпадает необходимость в дальнейшем разборе детальной статистики, ОПО может относиться к детальной статистике как к данным, предназначенным только для показа пользователю.

Сенсор не обращается к базе данных Nadmin и к другим конфигурационным файлам системы, он сделан максимально автономным. Это сделано для того, чтобы UNIX-процесс, исполняющий код сенсора, требовал как можно меньше ресурсов (память, процессорное время). Кроме того, эта независимость от базы данных и других компонент системы позволяет сенсору работать на удаленном компьютере.

Сенсоры реализованы как обычные процедурные программы на языке `Perl`. Каждый сенсор запускается ОПО как программа-демон, постоянно находящаяся в рабочем состоянии. После получения очередной порции статистики из лог-файла и ее обработки сенсор «засыпает» на определенное время, по прошествии этого времени сенсор

[2] pipe — системный механизм, позволяющий устанавливать каналы связи между родственными процессами

«просыпается» и процесс повторяется. За счет постоянно открытого указателя (`handler`) на лог-файл сенсор всегда «помнит», до какого места в лог-файле он уже обработал статистику, таким образом, одна строка никогда не обрабатывается дважды.

Многие источники сырой статистики — серверы с лог-файлами — используют понятие ротации лог-файлов. Например, в конце каждых суток закрывается текущий лог-файл, переименовывается, после этого создается новый лог-файл, в который заносится информация, относящаяся к следующим суткам. Таким образом, сенсор должен уметь отслеживать ротацию лог-файлов: при смене лог-файлов открыть указатель на новый файл. Большинство сенсоров отслеживают данное событие как смену индексного описателя файла (`inode`). В случае изменения `inode` происходит закрытие старого лог-файла и открытие нового.

3.2. Класс `Mark.pm`. Класс предоставляет нужное всем сенсорам средство: *закладки* (marks). Класс включает в себя методы для работы с закладкой (создание, сохранение, чтение). Закладки позволяют сенсору, обрабатывающему лог-файл, возобновить работу после перезапуска в том месте, где он остановился. Содержимое закладки может быть разное для каждого конкретного сенсора. Как правило — это последняя обработанная строка из лог-файла. Для сенсора абонентской платы — это дата, за которую уже произведено начисление абонентской платы.

3.3. Формат порции статистики. Порция статистики, передаваемая сенсором для ОПО, имеет вид, специфичный для каждого конкретного сенсора, но общий формат включает в себя заголовок с результатами обработки детальной статистики, и далее, после пустой строки, саму детальную статистику. Заголовок содержит обязательные поля `Date1` и `Date2` (даты начала и конца периода, к которому относится статистика) и поле `Id`, по которому происходит идентификация потребителя и поиск услуги, к которой относится данная порция статистики. Пример порции статистики см. на рис. 2.

3.4. Особенности реализации сенсоров. Так как сенсоры могут обрабатывать сырую статистику от самых разных источников, имеющих разную природу, то каждый сенсор помимо общих черт, описанных выше, может иметь и свои особенности. Рассмотрим особенности реализации для некоторых сенсоров:

```
Date1: 20030606160105
Date2: 20030606160110
Id: 193.232.174.2
Count-in: 19356
Class-in: HIGH
Count-out: 1346
Class-out: NORMAL

144:120 81.117.239.98.3333 x.8081 tcp
144:120 81.117.239.98.3233 x.3128 tcp
144:120 81.117.239.98.3162 x.80 tcp
144:120 81.117.239.98.3268 x.8080 tcp
144:0 81.117.239.98.2982 x.1080 tcp
```

Данная порция статистики описывает следующее: компьютер с ip-адресом 193.232.174.2 за указанный период (2003/06/06 16:10:05–2003/06/06 16:10:10) принял 19356 байт с высоким приоритетом и отправил 1346 байт со средним приоритетом.

Рис. 2. Порция статистики

3.4.1. *Сенсор* ***squid***. Данный сенсор обрабатывает сырую статистику от squid-сервера. При обработке лог-файлов сенсор определяет, является ли очередной запрос внутренним (локальным, то есть запрашивается информация из домена `botik.ru`), внешним (вне домена `botik.ru`), или информация поступила пользователю из кеша сервера. Сенсор суммирует объем полученной информации для каждого типа запросов и отдает порцию статистки ОПО.

3.4.2. *Сенсор* ***socks***. Данный сенсор анализирует сырую статистику от socks-сервера. Сенсор суммирует входящий и исходящий трафик и в заголовке фрагмента передает эту информацию ОПО. Особенность сенсора: в лог-файлах socks-сервера при записи даты не указывается год. Сенсор проводит дополнительный анализ для определения полной даты.

3.4.3. *Сенсор* ***tproxy***. Данный сенсор берет сырую статистику от transproxy-сервера. Сенсор суммирует входящий и исходящий трафик и отдает эту информацию ОПО.

3.4.4. *Сенсор* ***exim***. Данный сенсор занимается анализом почтовой статистики. Особеность работы сенсора exim связана со следующим:

- информация о доставке письма в лог-файлах exim-сервера занимает не одну, а несколько строк;
- почтовый трафик часто является многоадресным.

В лог-файлах информация о письме соотносится как минимум с двумя строками. Первая (в порядке поступления) строка идентифицирует письмо как поступившее на учет. Данная строка позволяет определить отпавителя. Вторая сообщает о том, что письмо было доставлено по адресу, и по ней идентифицируется получатель.

Таким образом, в некий момент времени сенсор хранит информацию обо всех поступивших на учет, но еще не доставленных письмах. Эта информация должна сохраняться на случай сбоя работы сенсора, иначе при последующем запуске сенсора некоторые письма будут не учтены. Для этого после обработки очередной порции статистики сенсор сохраняет как дополнительную закладку (mark) информацию о письмах, уже поступивших на учет, но еще недоставленных получателю. При запуске сенсора эта информация извлекается и восстанавливается картина, которая была на момент сбоя работы сенсора.

Также почтовый трафик является многоадресным, то есть получателями почты могут быть компьютеры и списки рассылки (к которым можно отнести также перенаправления почты, организованные на сервере, в частности файлы `.forward`). Все это должно учитываться при анализе.

3.4.5. *Сенсор* ***`abplata`***. Этот сенсор в отличие от большинства сенсоров обращается к базе данных Nadmin. После анализа базы данных сенсор `abplata` выдает ОПО информацию для расчета абонентской платы. Второй особенностью сенсора является то, что порции статистики, передаваемые этим сенсором, не содержат детальной статистики.

Раз в сутки примерно в одно и тоже время сенсор анализирует объекты `Connect` и `Net` в базе данных (см. раздел 4). В ходе анализа определяется основной тип каждого подключения. Затем в соответствии с заранее заданным порядком типов (например, в системе Nadmin задается следующий порядок: `NET`, `DIALUP`, `LAN`, `MAIL`, `lan`) происходит сортировка подключений. Внутри типа сортировка подключений ведется в лексикографическом порядке. Также определяется, равно ли значение статуса подключения `off` (отключенное) или нет.

В отсортированном виде информация по подключениям передается ОПО. В зависимости от очередности поступления информации о

подключении ОПО определяет статус подключения (`pri` — первичное или `sec` — вторичное).

Также сенсор `abplata` занимается анализом информации по аренде FTP/WWW-пространства. Этот анализ выполняется на основании статистики по использованию дискового пространства и категории пользователей.

В случае, если сенсор должен выдать для ОПО информацию за предыдущие дни, то используются ретроспективные возможности (см. раздел 4) базы данных.

3.4.6. *Сенсор* ***ipphone***. Данный сенсор обращается к сайту фирмы, предоставляющей СТ «Ботик» услуги по ip-телефонии, и берет с него информацию о произведенных звонках. После подсчета суммарной стоимости звонков для каждого пользователя эта информация передается ОПО.

4. База данных

Основой системы является база данных (БД) по абонентам сети и предоставляемым им услугам.

4.1. Объекты базы данных в системе Nadmin.

Org: данные об абоненте сети (организации или физическом лице, с которым заключен договор на обслуживание);

Account: данные о лицевом счете абонента;

Connect: данные об одиночном подключении (одном компьютере) абонента, получающего одну или несколько из следующих услуг: `MAIL`, `DIALUP`, `LAN`;

Net: Данные о групповом подключении (подключения локальной сети) абонента;

Ftp: Данные о FTP-входе с правом записи, предоставленном абоненту для доступа к арендуемому им FTP/WWW-пространству;

IPphone: Данные об абоненте IP-телефонии;

WWWuserif: Данные о пользователе Web-интерфейса.

Subnet: Данные о подсетях (интервалах IP-адресов в СТ «Ботик»);

4.2. Реализация базы данных системы Nadmin. База данных реализована как дерево каталогов с текстовыми плоскими файлами под управлением системы RCS[3]. За счет использования RCS автоматически реализуются два важных оригинальных свойства базы данных:

- возможность получения не только текущего состояния объекта, но и его состояния на любой момент со времени его создания (ретроспективность базы данных);
- хранение информации о дате и авторе каждого изменения объектов в базе данных.

Каждому корневому каталогу соответствует сущность базы данных (объекты `Org`, `Connect`, `Ftp`, `Net` и т.д.). Экземпляры сущностей представлены в виде файлов под управлением системы RCS, хранящиеся в соответствующих каталогах. Именем файла является индекс (номер экземпляра соответствующего объекта в БД) — `ID`. `ID` является сквозным для всех каталогов-сущностей, поэтому для сохранения значения последнего использованного значения индекса был заведён файл `last.id`.

Помимо каталогов с файлами база данных включает в себя так же набор индексных файлов. Индексные файлы содержат в себе основную информацию из базы данных на текущий момент времени, необходимую для быстрой работы системы.

Структура каталогов и файлов представлена на рис. 3:

4.3. Perl-интерфейс базы данных. Perl-интерфейс базы данных системы Nadmin представляет собой слой ПО, обеспечивающий доступ к базе данных для расчетной подсистемы и Web-интерфейсов администратора и пользователя. Интерфейс построен как иерархия Perl-классов (см. рис. 4). Использование объектно-ориентированного подхода позволяет сделать реализацию компактной и удобной.

Класс `Field`: представляет собой описатель одиночного поля базы данных. Из объектов класса `Field` строятся таблицы описателей полей объектов БД, которые используются для реализации массовых операций с полями.

Класс `RCSfile`: реализует набор базовых операций с файлами, находящимися под управлением системы контроля версий RCS.

[3] RCS — система управления версиями файлов.

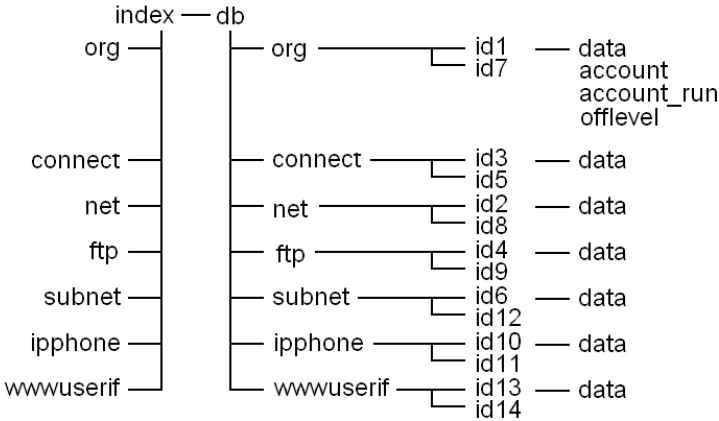

Рис. 3. База данных: структура каталогов и файлов

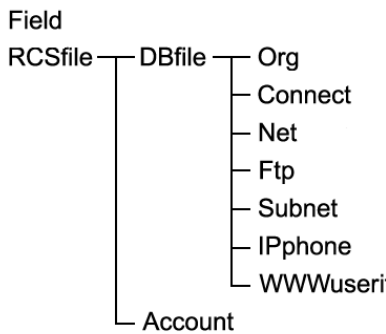

Рис. 4. Иерархия классов Perl-интерфейса базы данных

Класс `DBfile`**:** является производным от класса `RCSfile`. Самостоятельно не используется, только как базовый класс для конкретных объектов (`Org`, `Connect` и др.). Реализует функции, общие для этих объектов базы данных.

Классы `Org`, `Connect`, `Ftp`, `Net` и др.: являются производными классами от класса `DBfile`, представляющего объекты базы данных конкретных типов.

Класс `Account`: используется для доступа к лицевым счетам абонентов. Реализует функции просмотра, анализа, модификации лицевых счетов абонентов.

Класс `RXDfile`: реализует эффективный линейный поиск в мини-базе данных, представляющей собой файл со структурой: «регулярное выражение ... данные».

Класс `Mlists`: является производным от класса `RXDfile`. Используется для доступа к объекту `Mlists` в базе данных.

5. Лицевой счет и методы его обработки

Каждый абонент системы Nadmin имеет свой лицевой счет. На лицевой счет заносятся платежи абонента. С лицевого счета снимаются расходы абонента за различные услуги (подробнее см. раздел 6.1), а так же абонентская плата.

Лицевой счет является объектом Account базы данных (подробнее см. раздел 4) и представлен rcs-файлом ACCOUNT.

Иногда абонент может иметь несколько лицевых счетов. Такая ситуация решается вверением нумерации лицевых счетов:

- ACCOUNT;
- ACCOUNT2;
- ACCOUNT3 и т. д.

Объект Account описан в модуле Account.pm.

Все действия с лицевым счетом абонента делятся на несколько групп:

(1) занесение текущих расходов в run-файл лицевого счета (подробнее см. раздел 5.1);
(2) занесение на лицевой счет расходов за расчетный период и платежей;
(3) просмотр состояния лицевого счета;
(4) обработка изменений лицевого счета;
(5) обработка лицевых счетов во время перерасчета.

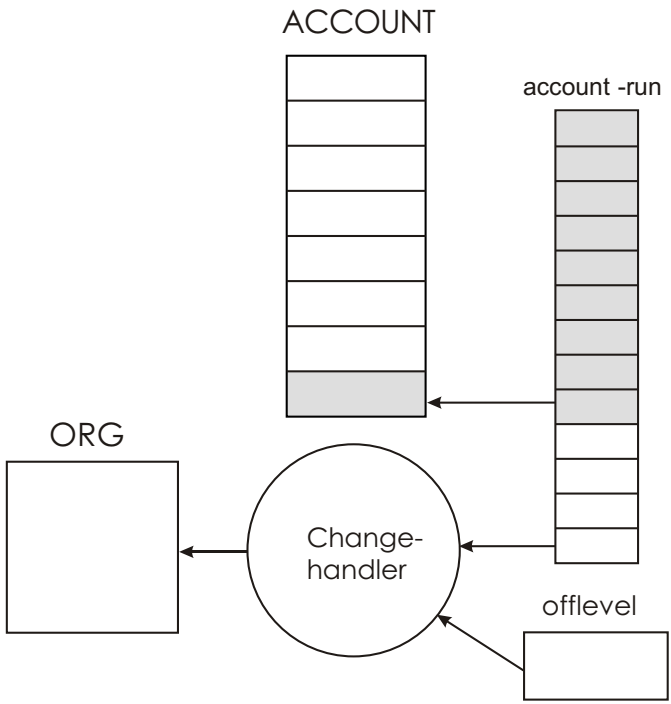

Рис. 5. Схема работы лицевого счета

5.1. Занесение текущих расходов в run-файл лицевого счета. Регистрация расходов абонента ведется в реальном времени.

Результат обсчета каждого слайса (подробнее см. раздел 6), поступившего от какого-либо сенсора (подробнее см. раздел 3), регистрируется в файле account_run (account_run2, account_run3 и т. д. в случае наличия нескольких лицевых счетов) для каждого абонента.

Каждая строка файла account_run содержит следующую информацию: начальная дата порции статистики, конечная дата порции статистики, накопительная стоимость траффика, класс услуги, название услуги, тип сенсора, объем входящего траффика, объем исходящего траффика, объем локального траффика.

Накопительная стоимость траффика получается суммированием стоимости траффика для слайса, представленного данной строкой, и стоимости траффика всех предыдущих слайсов, представленных в строках, расположенных выше данной строки.

Строки в account_run копятся в течение рассчетного периода.

Рассчетный период — отрезок времени, по истечени которого ситема Nadmin подводит итоги своей работы.

Длина рассчетного периода может быть единой для всех абонентов или может определяться тарифными планами абонентов.

В конце рассчетного периода просходит регистрация расходов на лицевом счету абонента, в файле ACCOUNT.

При этом строки файла account_run, пришедшие до начала периода ожидания слайсов, обрабатываются для формирования строки расхода в ACCOUNT, удаляются из файла account_run и архивируются (подробнее см. раздел 8).

Период ожидания слайсов — период фиксированной длины, в который возможно поступление запоздавшей по техническим причинам статистики.

5.2. Занесение на лицевой счет расходов за рассчетный период и платежей. Строки файла ACCOUNT бывают двух видов — строки расходов и строки платежей.

- *Строка расходов.* Строка расходов формируется из строк файла account_run (подробнее см. раздел 5.1). Она включает дату конца рассчетного периода, сумму расхода, остаток на счету, комментарий. Комментарий имеет вид: расход за Date1 — Date2, где Date1 — дата конца прошлого рассчетного периода (дата начала рассматриваемого рассчетного периода), Date2 — дата конца рассматриваемого рассчетного периода.
- *Строка платежа.* Платеж осуществляется с web-страницы администратора (подробнее см. раздел 7). Строка платежа включает дату регистрации платежа, сумму платежа, остаток на счету, комментарий. Комментарии, как правило, содержит информацию о том, кем и где был принят платеж и другую информацию.

5.3. Просмотр состояния лицевого счета. Для просмотра состояния лицевого счета используются методы Account.pm, позволяющие просмотреть заданное число последних записей и характеристики текущего состояния лицевого счета, которые перечислены ниже.

- *Остаток на лицевом счету на начало текущего рассчетного периода.* Он извлекается из файла ACCOUNT.
- *Сумма расходов за текущий рассчетный период.* Эта сумма извлекается из файла account_run.
- *Текущий остаток на лицевом счету.* Получается суммированием остатка лицевого счета на начало рассчетного периода и суммы расходов за текущий рассчетный период.
- *Текущий долг абонента.* Он равен нулю, если текущий остаток на лицевом счету положителен, иначе равен абсолютному значению текущего остатка на лицевом счету.

Просмотр состояния лицевых счетов абонентов пользователями системы Nadmin (как абонентами, так и администраторами) производится через web-интерфейс (подробнее см. раздел 7).

5.4. Обработка изменений лицевого счета. Есть три события, которые необходимо обрабатывать в системе Namdin:

(1) обнуление лицевого счета (либо пресечение лицевым счетом, так называемого, уровня отключения),
(2) пополнение лицевого счета,
(3) изменение уровня отключения.

Одной из особенностей новой версии рассчетно-статистической системы Nadmin является введение понятия уровня отключения.

Уровень отключения — это минимальное значение суммы на лицевом счету, при котором происходит отключение абонента. Уровень отключения формируется совместно системой и абонентом и может принимать разные значения.

- *Уровень отключения равен нулю.* Нуль является умолчательным значением уровня отключения.
- *Уровень отключения больше нуля.* Абонент всегда может сделать уровень своего отключения положительным (но не

больше суммы на его лицевом счету). Это необходимо, например тогда, когда абонент ставит перед собой цель тратить в какой-либо период фиксированную сумму со своего лицевого счета.

- *Уровень отключения меньше нуля.* Такой уровень отключения допустим не для всех абонентов. Если расходы абонента в течение нескольких рассчетных периодов не опускались до уровня отключения, то получается, что абонент кредитовал СТ «Ботик». Поэтому СТ «Ботик» может кредитовать данного абонента, разрешив ему делать свой уровень отключения меньше уровня по умолчанию. Минимальный уровень отключения для таких абонентов вычисляется как взвешенное среднее от сумм, на которые абонент кредитовал СТ «Ботик» в течение нескольких последних рассчетных периодов.

На каждое изменение состояния лицевого счета, а также уровня отключения, срабатывают обработчики лицевого счета. *Обработчик* — функция, описанная в модуле Change_handler.pm. Обработчик сравнивает остаток на лицевом счету с уровнем отключения и в зависимости от результатов сравнения отключает или подключает абонента.

События, которые подлежат обработке подробно рассмотрены в последующих разделах.

5.4.1. *Обнуление лицевого счета (пересечение уровня отключения.* Если на лицевом счету сумма достигает уровня отключения, абонент автоматически отключается от сети.

5.4.2. *Пополнение лицевого счета.* При внесении на лицевой счет абонента платежа происходит корректирование уровня отключения, а затем сравнение новой суммы на лицевом счету с полученным уровнем отключения. Если абонент был отключен и новая сумма на лицевом счету превышает уровень отключения, абонент автоматически подключается к сети.

5.4.3. *Изменение уровня отключения.* При попытке уменьшения уровня отключения, новое значение уровня отключения сравнивается с минимально допустимым уровнем отключения и с остатком на лицевом счету.

Если оно меньше минимального уровня, то происходит его замена на значение минимального уровня. Далее происходит его сравнение с суммой на лицевом счету.

Если значение уровня отключения больше суммы на лицевом счету, то происходит его замена на значение остатка на лицевом счету. Если абонент был подключен к сети, то при срабатывании обработчика изменений лицевого счета он от нее отключается.

Если сумма на лицевом счету больше уровня отключения, то при срабатывании обработчика изменений лицевого счета, абонент, если он был отключен, автоматически подключается к сети.

5.5. Обработка лицевых счетов во время перерасчета. Если обнаружены ошибки в обработке слайсов, работе справочника или функций биллинга за какой-либо период, то за этот период происходит перерасчет (подробнее см. раздел 6.5). На одной из стадий перерасчета происходит перерасчет (переписывание) файлов account_run и ACCOUNT. Для каждого абонента формируются новые строки файла account_run за период перерассчета.

Они подставляются в зависимости от начальной и конечной даты периода в текущий account_run и (или) в файлы account_run из архива.

Далее изменяется файл ACCOUNT: путем обработки строк измененных файлов account_run происходит изменение сумм расхода и остатка на лицевом счету за каждый рассчетный период, подверженный перерасчету.

6. Расчетная подсистема

В данном разделе рассматривается реализация расчетной части системы Nadmin. Она включает в себя:

(1) основной процесс-обработчик;
(2) механизм описания биллинговых функций;
(3) механизм расчета стоимости мегабайта;
(4) программу перерасчета статистики (в случае сбоя или ошибки).

6.1. Основной процесс-обработчик. Основной процесс-обработчик
(ОПО) является ядром расчетной системы. В его обязанности входит получение порций статистики от сенсоров и запуск соответствующей

расчетной (биллинговой) функции, которая производит расчет стоимости потребления некоторого ресурса[4]. Схема работы ОПО приведена на рисунке 7.

Процесс организован следующим образом: в самом начале работы ОПО происходит инициализация. Она заключается в построении справочника
(см. раздел 6.2), кэшировании необходимой для работы системы части базы данных, которая не попадает в справочник, и запуске сенсоров. При запуске сенсоров каждый из них обязательно сообщает ОПО свой тип. Это небходимо для того, чтобы ОПО с самого начала имел информацию, какой биллинговой функцией (см. раздел 6.3) должны обрабатываться порции статистики, приходящие от данного сенсора.

После завершения инициализации ОПО начинает принимать и обрабатывать порции статистики, приходящие от сенсоров.

Процеесс обработки порций статистики организован следующим образом: каждый сенсор связан с ОПО с помошью канала (pipe). Для каждого открытого канала работает функция select[5]. Обнаруженные на каком-либо канале данные складываются в буфер. Затем к содержимому буфера применяется парсер порций статистики.

Парсер просматривает буфер и пытается выбрать из него готовые порции статистики. Парсер забирает из буфера готовые порции статистики до тех пор, пока в буфере ничего не останется или же в буфере останется только одна незаконченная порция. Затем парсер разбивает каждую из готовых порций по полям и возвращает массив порций статистики, представленных с помощью хэша языка perl. Структура порции статистики приведена на рисунке 6.

Обработка каждой готовой порции статистики состоит из следующих шагов:

(1) *Определение услуги.* Услуга — объект базы данных, непосредственно связанный с абонентом, характеризующий род ресурса. Услуга небходима биллинговым функциям, так как в ней хранится информация, необходимая для вычисления стоимости потребленного ресурса. Услуга определяется по

[4]Ресурс — например, внешний траффик, ip-телефония, арендуемое ftp-пространство и т. д.

[5]Select — системная функция, позволяющая определять, на каких pipe‘ах есть данные. Она необходима для того, чтобы избежать засыпаний процесса при попытках обращения к pipe‘ам, где нет готовых данных.

```
Date1: <дата 1>
Date2: <дата 2>
Id: <идентификатор>
<название поля 1>: <значение поля 1>
<название поля 2>: <значение поля 2>
...
<название поля n>: <значение поля n>

<необработанная статистика>
<необработанная статистика>
<необработанная статистика>
...
```

Рис. 6. Структура порции статистики

идентификатору (Id) из порции статистики с помощью справочника (6.2).

(2) *Запуск биллинговой функции.* Каждый из объектов, относящихся к типу «Услуга», наследуется от базового класса с одноимённым названием. В этом классе описан метод billing, который запускается при обработке каждой порции статистики. Этому методу на вход подаётся порция статистики и информация об ее источнике (ресурс). В зависимости от источника порции статистики запускается та или иная биллинговая функция, которой передаётся информация, хранящаяся в заголовке порции статистики. Все биллинговые функции описываются в конфигурационном файле[6]. Биллинговая функция возвращает сумму (оценка стоимости потреблённого ресурса), которая заносится в виде расхода на счет организации, потребившей данный ресурс.

(3) *Архивация порции статистики.* После того, как подсчитана стоимость и проведены все бухгалтерские транзакции, порция статистики архивируется с помощью специального класса Archive.

6.2. Справочник. Справочник является вспомогательной частью для ОПО, хотя может использоваться самостоятельно. В его обязанности входит:

[6]Конфигурационный файл Nadmin — специальный файл, в котором описаны функции и переменные специфичные для сети, в которой эксплуатируется система Nadmin.

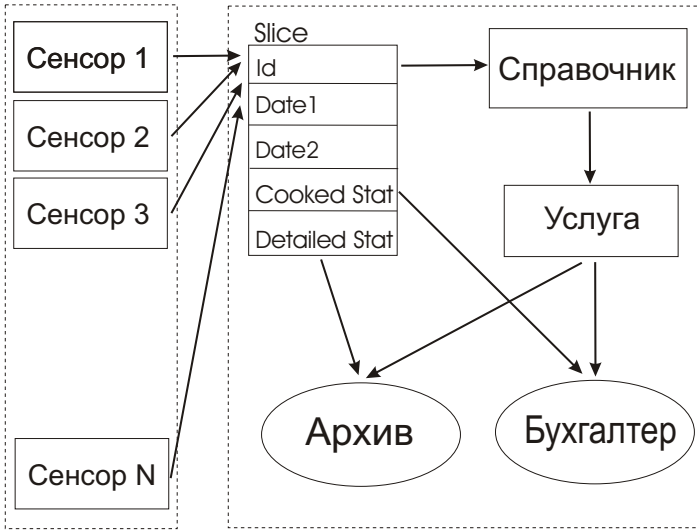

Рис. 7. Основной процесс-обработчик

(1) *Построение ретроспективного кэша.* Поскольку все услуги (в общем случае это объекты базы данных) хранятся вместе со всей историей их изменений (это достигнуто за счёт использования системы RCS), для нас важно включить историю изменений в кэш. При постороении кэша пользователем задаётся (или задаётся по умолчанию) глубина просмотра истории. Состояния кэшируемых объектов базы данных «откатываются» во времени на заданную глубину, и в кэш заносятся все состояния объекта до момента построения кэша. Ключом каждого состояния объекта становится дата и некоторый уникальный идентификатор, который строится из одного или нескольких полей объекта, это специфично для каждого типа объектов.

(2) *Определение услуги по идентификатору и дате.* После построения кэша к справочнику начинают поступать запросы на определение услуги по идентификатору и дате. Для этого просматривается кэш и выбирается объект с совпадающим с входным идентификатором и датой, ближайшей к входной.

(3) *Обновления кэша.* Во время работы справочника в базе данных могут происходить изменения (добавление, изменение или удаление данных/объектов). Для обновления кэша используется специальный монитор изменений базы даных, который при изменениях в базе данных информации, которые могут повлиять на содержимое справочника, посылает сигнал ОПО. ОПО, в свою очередь, приостанавливает работу и запускает механизм построения кэша, в результате чего строится новый справочник, содержащий замеченные сенсором изменения. После этого ОПО продолжает свою работу.

6.3. Биллинговые функции. Биллинговые функции являются специфичным механизмом определения стоимости оказанной услуги для каждого провайдера. Поэтому они вынесены в конфигурационный файл. С одной стороны, язык описания должен быть достаточно простым, с другой, автор биллинговых функций не должен быть ограничен специализированным метаязыком. Исходя из того, что авторами биллинговых функция чаще всего являются системные программисты, было решено использовать язык Perl для их написания.

Для удобства написания биллинговых функций разработано специальное пространство переменных (namespace) и поддерживается соглашение, что перед вызовом любой биллинговой финкции все эти переменные правильно проинициализированны. В это пространство имён входят все переменные, которые необходимы для определения стоимости оказанной услуги. Имена переменных совпадают с названиями полей из заголовка порции статистики, содержащие значения этих полей, ссылка на услугу, ссылка на объект содержащий информацию об абоненте. Такой подход значительно упрощает процесс написания биллинговых функций.

6.4. Расчет стоимости мегабайта. В СТ «Ботик», для расчетов с абонентами используется переменная стоимость цены одного мегабайта, которая вычисляется один раз в конце каждого расчетного периода. Поскольку все расчеты производятся в реальном времени, была использована идея предварительного и окончательного расчета.

В начале расчетного периода устанавливается предварительная цена мегабайта, и все расчеты ведутся с этой ценой. В конце расчетного периода со счетов абонентов снимается сумма, вычисленная

с предварительной ценой мегабайта. Затем производится вычисление реальной стоимости цены мегабайта. После этого вычисляется поправочный коэффициент — отношение реальной цены мегабайта к предварительной цене мегабайта.

После вычисления поправочного коэффициента у каждого абонента переменная часть расходов умножается на этот коэффициент. Если полученная сумма превышает ранее полученную сумму (с переменной ценой мегабайта), то вычисляется их разность, и со счетов абонентов снимается дополнительная сумма. Если же полученная сумма меньше ранее полученной суммы, то также вычисляется их разность и добавляется на счета абонентов. При равных результатах ничего не предпринимается.

6.5. Перерасчет. Во время расчетов могут происходить ошибки (в результате сбоя базы данных или еще по каким-то причинам). Для выявления и исправления таких ошибок используется перерасчет за заданный период времени.

Имеется две стадии перерасчета:

(1) перерасчет всей статистики за заданный период и формирование отчетов;
(2) подстановка перерасчитанной статистики вместо старой (не верной) статистики;

Первая стадия может исполняться в двух режимах:

(1) перерасчет всей статистики без пересоздания порций статистики;
(2) перерасчет всей статистики с пересозданием порций статистики;

В *первом режиме* порции статистики за заданный период целиком берутся из архива и целиком отдаются ОПО на пересчет. Во *втором режиме* порции статистики сначала извлекаются из архива и перед тем как пойти на пересчет ОПО, они отдаются функциям построения порций статистики в сенсоры, и только после этого они поступают в ОПО.

Таким образом, первый режим позволяет исправить ошибки, допущенные на уровне сенсоров, а второй режим исправляет ошибки, допущенные на уровне ОПО, например, в результате сбоя справочника.

7. Веб-интерфейс

Для реализации пользовательского интерфейса, при помощи которого абонент может управлять собственными услугами, необходимо было разработать инструментарий, позволяющий стоить налету веб-странички, наполненные различным содержанием, в зависимости от сведений из базы данных Nadmin о данном абоненте и о его подключениях. Такой инструментарий был разработан: для создания веб-страниц используются шаблоны веб-страниц. Шаблоном веб-страницы является обычный html-документ с особыми комментариями (псевдокоммертариями), которые заменяются на необходимые данные из системы Nadmin.

7.1. Веб-интерфейс пользователя. Абоненту доступны следующие веб-страницы системы Nadmin:

(1) Страница «IP-телефония». Здесь абонент может:
   (a) создавать пользователей IP-телефонии для своей организации;
   (b) просматривать статистику о совершенных звонках.
(2) Страница «Лицевой счет абонента». Здесь абонент может:
   (a) просматривать состояние баланса организации;
   (b) просматривать статистику по внешнему траффику (ip, proxy, mail) подключений;
   (c) просматривать графики расходов организации и подключений;
   (d) просматривать детализацию статистики траффика по подключениям.
(3) Страница «Управление режимом отключения или подключения». Здесь абонент может:
   (a) устанавливать пароли для системы botikkey.
(4) Администрирование. Здесь абонент (администратор) может:
   (a) создавать пользователей для организации;
   (b) удалять пользователей у организации;
   (c) менять пароли пользователям у организации.

У организации может быть два вида пользователей веб-интерфейсом:

(1) обычный пользователь;
(2) администратор — только администратору позволенно пользоваться страницами администрирования.

7.2. Веб-интерфейс администратора сети. Администратор сети может пользоваться следующими страницами:

(1) страница «просмотр списка организаций с их описанием»;
(2) страница «просмотр списка организаций с датами подписания прейскурантов»;
(3) страница «просмотр списка организаций с данными о лицевом счете»;
(4) страница «просмотр списка всех подключений»;
(5) страница «просмотр списка модемных подключений»;
(6) страница «рассылка писем клиентам»;
(7) страница «расширенного поиска». Поиск может быть осуществлен в:
    (a) организациях;
    (b) подключениях;
    (c) FTP-входах;
    (d) подсетях.
(8) страница «печать счетов-фактур всех организаций за прошлый месяц»;
(9) страница «статистика для Госсвязьнадзор».

8. Архивация данных

Архивация данных производится при помощи модуля Archive. Этот класс обладает следующей функциональностью:

- сохранение данных в файл, название файла имеет вид: date1-date2, где указанные две даты означают, что в данном файле содержится информация за перод времени с date1 по date2;
- компактирование данных в tar.bz2-архив, где название файла имеет вид: date1-date2.tar.bz2;
- получение данных из архива производится по указанной дате или интервалу;
- удаление файлов из архива производится по указанной дате.

9. Состояние проекта и планы дальнейших работ

На сегодняшний день реализованы все основные компоненты системы. Система доведена до такого состояния, когда вся функциональность, которая обеспечивалась в предыдущей версии Nadmin,

поддержана. Кроме того реализованы многии запланированные расширения. Реализовано одно из основных свойств новой систеиы — обсчет статистики в реальном времени.

Система запущена в экспериментальную эксплуатацию, то есть система получает статистику из реальных источников и работает под реальной нагрузкой. По имеющимся результатам экспериментальной эксплуатации уже можно сделать вывод, что использованная архитектура полностью оправдывает себя.

Ближайшие планы — в процессе эксперементальной эксплуатации довести систему до такого состояния, когда ее можно будет ввести в практическую эксплуатацию. Для этого используется параллельная работа старой и новой версии системы Nadmin и сравниваются результаты их работы, что позволяет выявлять ошибки, допущенные при реализации новой версии системы.

Благодарности. Авторы благодарны сотрудникам лаборатории телекоммуникаций «Ботик», чьи пожелания и требования к функциональности новой версии системы Nadmin повлияли на принятие технических решений, в частности Бурчу Сергею Владимировичу за помощь в отладке сенсоров и основного процесса-обработчика.

Особая благодарность Абрамову Сергею Михайловичу за помощь в подготовке статьи, а также за плодотворные дискуссии и многочисленные обсуждения, во время которых были сформулированы идеи, лежащие в основе многих разделов данной статьи.

Список литературы

[1] Т. Кристиансен, Н. Торкингтон PERL: библиотека программиста: Издание официальное. — СПб: Питер, 2001. ↑

[2] Шевчук Ю. В.. 1999. *Методы построения экономически эффективных региональных компьютерных сетей*, Диссертация на соискание степени кандидата технических наук, Институт программных систем РАН, Переславль-Залесский. ↑

[3] Абрамов С. М., Пономарев А. Ю., Шевчук Ю. В.. 1999. *Широко доступный Интернет как путь в открытое общество. Опыт Переславля-Залесского*, Труды конференции «Интернет. Общество. Личность», СПб. ↑

[4] Абрамов С. М., Пономарев А. Ю., Шевчук Ю. В.. 1999. *Технология построения недорогих, но качественных гражданских сетей. Опыт Переславля и его перенос в другие регионы*, Тезисы докладов Всероссийской научной конференции «Научный сервис в сети Интернет», МГУ. ↑

[5] Абрамов С. М., Позлевич Р. В., Пономарев А. Ю., Шевчук Ю. В.. 1997. *Экономически эффективные технологии построения региональных сетей для науки и высшей школы*, Труды конференции Телематика'97, СПб. ↑

[6] *Сеть для всех и по разумным ценам* // Компьютерра, № 34, с. 28–30. (russian) ↑

Исследовательский центр мультипроцессорных систем ИПС РАН

E. V. Ermilova, A. V. Karlash, A. S. Nesterov, P. G. Zhbanov, Yu. V. Shevchuk. *PSI RAS Nadmin billing subsystem.* (in russian.)

Abstract. During the last 5 years administration of Pereslavl-Zalessky civil network was performed with the administrative system Nadmin. But during the 5 years passed a great progress in sofware and hardware, that lead to the need of development of new version of Nadmin. In this work properties, architectute and programming principles are being described.