P2P народная сеть Пандора

мир в твоих руках

Руководство пользователя и разработчика

1. Введение

Компьютерные сети прочно вошли в нашу жизнь. Мы узнаём новое, общаемся с близкими, заключаем сделки, участвуем в разработке проектов, управляем процессами.

В интернете много сервисов. Несмотря на кажущееся разнообразие, основные сервисы достаточно централизованы. Google имеет сервера в определенном месте, миллионы запросов проходят через этот центр и анализируются. Facebook, Vkontakte также имеют свои центры, миллионы сообщений людей проходят через центральный сервер социальной сети и отслеживаются. Skype на сегодня также имеет выделенные сервера, через которые проходит весь трафик. Сети электронной коммерции, разработки проектов, форумы — всё находится на чьих-либо центральных серверах. Вся информация проходит через единые центры и хранится там. Это общеизвестно. В чём проблема?

Проблема первая. Сегодня мы доверяем свою информацию обезличенным корпорациям. Остаётся искренне уповать на то, что во многих корпорациях работают порядочные люди, которые уважают наше доверие. *Но в корпорациях могут быть люди, которые распоряжаются вверенной им информацией преследуя свои умыслы*.

Проблема вторая. Сегодня во всех странах правительственные службы на законодательном уровне контролируют все информационные потоки. У каждого интернет-провайдера установлено оборудование спецслужб, которое полностью отслеживает весь трафик. Считается что правительства всегда поступают честно. *Но среди государственных служащих могут быть люди, которые преследуют интересы определенных групп*.

Проблема третья. Центральные точки в сети подвержены риску выйти из строя или быть уничтоженными во время внешнего вторжения. Случаи, когда сервера выходят из строя (или их выводят из строя), и системы перестают работать, случались неоднократно. Кроме того, некоторые владельцы просто останавливали свои сервера, иногда даже не уведомив нас. В этих случаях мы больше не можем пользоваться остановленными сервисами и зачастую теряем свои данные. *Централизованные сервисы физически уязвимы и подчинены воле определенных лиц*.

В трёх словах: **центры подвержены утечкам, цензуре и краху**.

Это три основные проблемы, но далеко не единственные. Скажите, на скольких сайтах вам приходилось регистрироваться и раз за разом вводить одну и ту же информацию? Сколько раз вам приходилось вникать в особенности каждого сервиса и перестраиваться под различные требования? Сколько программистов каждый день запускают очередной сайт, чтобы создать для нас новую головоломку?

Не слишком ли это, лишь для того чтобы:

1) узнавать новое;
2) общаться с близкими;
3) заключать сделки;
4) совместно разрабатывать проекты;
5) контролировать процессы.

Не слишком ли много порождено сущностей для наших простых задач?

Не слишком ли дорогую цену мы платим?

2. Что такое Пандора?

Pandora – это компьютерная программа, которая хранит ваши данные, данные ваших близких, друзей, партнёров и единомышленников. Пандора имеет встроенную криптографию, позволяет строить схемы доверия, в которых идёт защищенный обмен данными. Узлы Пандоры объединяются в подсети, а отдельные подсети связываются в глобальную мировую сеть.

Пандора предоставляет функционал социальной сети (наподобие facebook или vkontakte), средства голосового и видео общения (skype), публичной энциклопедии (wikipedia), деловой системы (1С), электронного магазина (ebay), платежной системы (paypal), реестра законов и стандартов (Консультант+), средства совместной работы над проектами (git), систему голосования и рейтингов (democratia2.ru). Таким образом, Пандора может использоваться для личного общения, ведения бизнеса, создания проектов и совместного управления обществом.

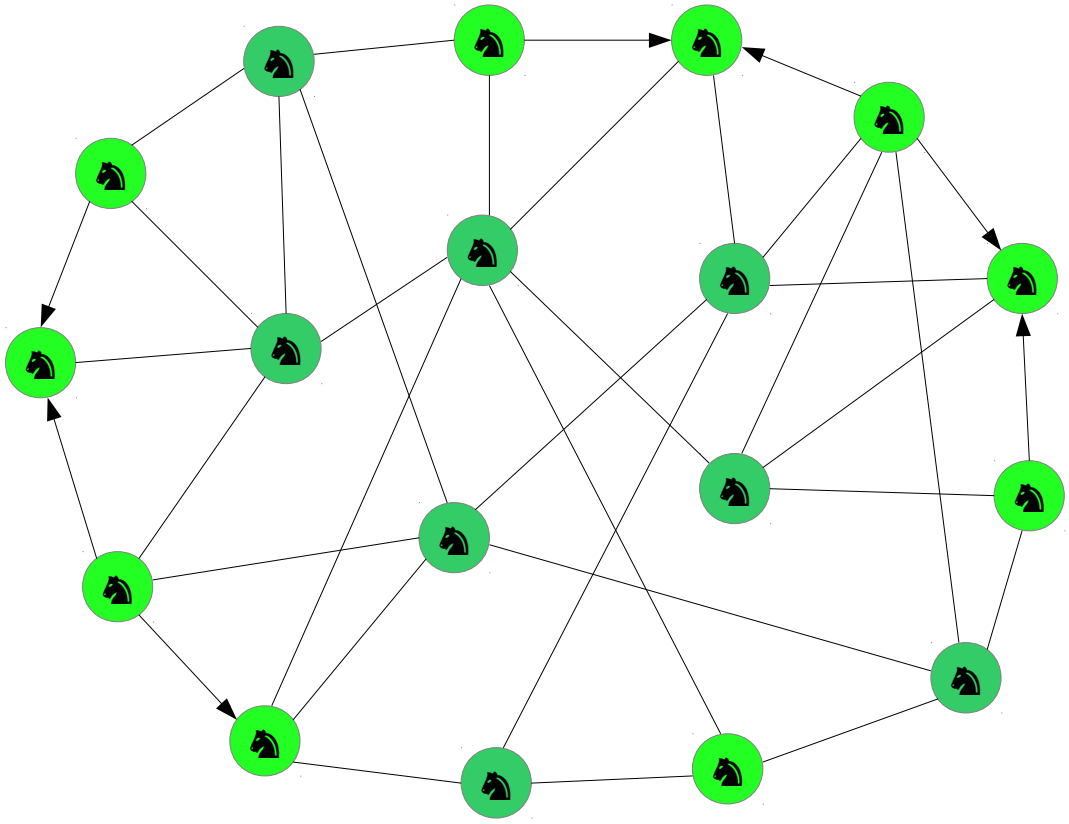


Рисунок 1: Строение p2p-сети Пандора

Сеть полностью децентрализованная. Невозможно вывести сеть из строя или взять её под контроль. Публичные данные (например, энциклопедические статьи) свободно курсируют между узлами, приватные данные распределяются по узлам, согласно схемам доверия.

Так как обмен данными происходит среди близких, друзей, коллег, партнеров по бизнесу, единомышленников, то вероятность утечек данных третьим лицам почти отсутствует. Фотографии ваших детей попадут только на компьютеры близких. Анекдоты вы будете травить только со своими друзьями. Ваш заказ увидит только продавец. Деловые бумаги разойдутся только партнерам. Над проектами работают только единомышленники.

Общение ответственное и конструктивное. Ответственное, потому что люди заботятся о своей репутации. Конструктивное, потому что никто не хочет поддерживать бесполезные данные, размещая их на своём компьютере.

На сегодня реализованы: ведение персональных анкет; обмен короткими сообщениями, видео и аудио звонки; движение товаров и услуг, электронная торговля; размещение резюме и поиск вакансий; публикация статей, фотографий и проектов; сбор подписей и голосование; управление схемами доверия между людьми.

Разработка Пандоры ведётся программистами-добровольцами за счёт личных накоплений и пожертвований со стороны сообщества. Пандора является общественной собственностью и распространяется под свободной лицензией GNU GPLv2, также вы можете распространять код под лицензией GNU GPLv3 или позднее. Любой может безвозмездно устанавливать Пандору на свой компьютер, использовать в личных, коммерческих и общественных целях, изменять открытый программный код под свои нужды.

Пандора может работать на компьютере, ноутбуке, планшете под операционными системами GNU/Linux, Windows, Macosx и другими.

3. Кодекс пандорианца

Несколько тысяч лет назад человек потерял лицо и отошёл на второй план. На первое место вышли государства, правительства, министерства, корпорации, обезличенные деньги и другие абстрактные сущности.

Пандора вновь ставит ценность личности на первое место. Основная цель Пандоры – учёт интеллектуального и материального вклада каждого человека в развитие цивилизации; преобразование человека таким образом, чтобы он развивался и приносил максимальную пользу человечеству.

Создание идей и вещей, оказание услуг является положительным качеством, а разрушительное поведение является отрицательным качеством.

Созидание, синтез, конструктив, проектирование объявляются первичной целью человека. А разрушение, анализ, критика, деструкция объявляются подготовительными к созиданию, и порицаемыми в отрыве от него.

Тем не менее, помимо «светлого» (ответственного) общения, конструктивного, открытого, с ясными личностями, с авторизацией, в целях развлечения Пандора технически поддерживает «тёмное» (безответственное) общение, с виртуальными личностями в облаке добровольных анонимусов. Анонимное общение может быть частично ограничено или полностью заблокировано.

Для стабилизации работы народной сети Пандора разработан кодекс поведения, который регулирует отношения реальных людей, виртуалов и анонимусов:

1) реальный человек указывает свои настоящие имя, фамилию и дату рождения
виртуал указывает заведомо неиспользуемые имя и фамилию (без даты рождения)
анонимус указвает только имя, например Anonymous
2) если вам не нравятся ваши реальное имя и фамилия, то заведите новую анкету, но при этом сделайте связь «равно» на реальное имя, а также сделайте публичный комментарий к новому имени, поясняющий переименование. Дату и место рождения при этом оставьте настоящими
3) анонимусы и виртуалы не имеют права:

– выдавать себя за реальных людей
– вести деструктивную деятельность по отношению к реальным людям
– задавать в своём профиле дату рождения
4) если анонимус или виртуал нарушил правило, он по возможности вскрывается и публично наказывается, его ключ объявляется деструктивным
5) оценивать анкету анонимуса или виртуала бессмысленно, так как их может быть бесконечно много, вместо этого понижайте доверие их ключам
6) также нельзя оценивать реального человека, в существовании которого вы не уверены на 100%. в интернете кто-то может выдавать себя за другого и вредничать от его имени. поэтому когда понижаете карму реальному человеку, будьте уверены, что это именно тот человек. также старайтесь избегать эмоциональных оценок, поддавшись эмоциям в оценке человека, вы можете выставить себя неадекватным. лучше всего, когда вы оцениваете человека по его делам в реальной жизни, а не по тому, что он говорит в сети
7) анонимусы и виртуалы соприкасаясь с реальными людьми могут быть только позитивными и конструктивными, не пакостят, не тролят, не анализируют, не оскорбляют
8) люди не борятся с анонимусами и виртуалами пока те не лезут к ним
9) все пандорианцы (анонимы, виртуалы и реальные люди) защищают сеть Пандору от подавления и блокирования
10) каждый человек несёт личную ответственность за свои действия
11) человек объясняет свои действия своим мировоззрением и моральными установками
12) человек не может прятаться за государство, министерство или корпорацию
13) если совершаются деструктивные действия, необходимо выявлять всех лиц персонально
14) реальные люди ставят во главу угла созидание, конструктив и синтез
15) реальные люди избегают сосредоточения на критике, анализе и деструктиве
16) тем не менее, реальные люди оттачивают деструктивные способности и применяют их при вторжении деструкторов следуя принципу воспитательной контрдеструкции (ВКД)
17) смысл ВКД – подвергать встречному разрушению агрессора до тех пор, пока он не признает, что только в процессе созидания возникают полезные вещи или идеи. цель ВКД: переключение человека от критики и разрушения к проектированию и созиданию
18) ВКД – это деструктивная реакция на деструктора с целью его перевоспитания, а именно переключения его из режима разрушения в режим созидания. «воспитательная» означает, что вы должны давить на разрушителя настолько сильно и до тех пор, пока он не согласится изменить своё поведение, но при этом не уничтожить его (как морально, так и физически)
19) ВКД исходит из того, что бесперспективно строить с человеком, который настроен на разрушение. Бесперспективно вести конструктивный диалог с человеком, который настроен деструктивно. Это всё равно, что строить дом под бомбёжками и артобстрелом. Ломать легче, чем строить. Тот кто критикует, всегда будет иметь преимущество перед тем, кто пытается вести конструктивный диалог. Если вы хотите достроить дом, пора отложить мастерок, выключить бетономешалку, садиться за зенитное орудие, в танк и подавить агрессора. При этом нужно стараться не уничтожить его, а заставить признать, что первичная цель человека – созидание. Только после этого можно возвращаться к строительству. В этом и заключается суть ВКД
20) только созидание делает мартышку человеком (но не модная одежда или богатый папик)
21) пандорианцы защищают друг друга, особенно конструктивных членов
22) существует эволюция. существует революция
23) эволюция лучше революции. революция – крайняя мера
24) если есть выбор, лучше участвовать в эволюции, чем в революции
25) дураки и злыдни такие потому что в неведении. уничтожить легче, чем объяснить. но объяснить – правильнее. поэтому конструктивный диалог и ВКД – главные методы пандорианца

26) агрессия – форма страха. нападает, значит боится. хочешь, чтобы не нападал – развей его страхи
27) реальный мир очень простой. или очень сложный. но мы этого никогда не узнаем, потому что строим в уме свой мир, о котором и судим
28) умозрительный мир человека может быть простым или сложным, это не имеет значения для общества, главное, чтобы мир человека вёл к созиданию
29) сильные вашего умозрительного мира могут оказаться не такими уж и сильными в реальном мире. и наоборот
30) но среди них есть опытные, и их опыт может пригодиться в созидании
31) лучший способ прекратить безобразие – перестать кормить безобразников
32) различай конструктивность, деструктивность и контрдеструктивность
33) подчиняй своё время Великому Балансу:
деструкция (критика, аналитика, провокации, троллинг, разрушение) - 30%
конструкция (конструирование, синтез, творчество, созидание, проектирование, строительство) - 60%
контрдеструкция (желательно, ВКД) - 10%
в зависимости от ситуации может быть другая пропорция, но всегда лучше, когда присутствует конструирование
34) группа (дом, улица, город, район, область, страна, планета) сама решает, что для неё легально, а что нет
35) легальность всегда относительна и всегда ограничена группой людей
36) легальность крупной группы определяется мелкими группами, из которых она состоит
37) не спрашивай, что группа может сделать для тебя, спроси, что ты можешь сделать для группы
38) если после долгих попыток группа не оценила твоего созидания, попробуй себя в другой деятельности. или в другой группе
39) не факт, что в другой группе ситуация изменится, возможно, ты нарушил Великий Баланс, возможно ты излишне деструктивен, это повод к перестроению своего поведения, будь конструктивнее
40) оголтелые критики с нарушенным балансом поведения должны подвергаться ВКД
41) любая агрессия со стороны анонимусов и виртуалов по отношению к реальным людям и их идеям должна восприниматься как угроза реальному миру
42) отсюда следует, что деструктивные действия анонимусов и виртуалов по отношению к реальным людям должны жёстко пресекаться.

4. Установка

Используйте соответствующий способ для вашей операционной системы.

Ubuntu

Выполните три команды в терминале:

```
sudo apt-add-repository -y ppa:pandora-net/ppa
sudo apt-get update
sudo apt-get install -y pandora-net
```

Windows

Скачайте инсталлятор и запустите под правами Администратора:

https://github.com/Novator/Pandora/releases/download/0.6-alpha/pandora_setup.exe

Debian

Скачайте пакет и установите под root'ом:

https://github.com/Novator/Pandora/releases/download/0.6-alpha/pandora-net_0.6-1ubuntu_all.deb

```
dpkg -i pandora-net_0.1-1ubuntu_all.deb
```

Unix/Bsd/Macosx

1) скачайте архив https://github.com/Novator/Pandora/archive/master.zip, распакуете в папку **/opt/Pandora/**

2) установите пакеты **ruby, ruby-sqlite3, ruby-gtk2, openssl, unzip, ruby-ncurses, screen, ruby-gstreamer, gstreamer0.10-ffmpeg, gstreamer0.10-x**

3) скопируйте ярлык **/opt/pandora/view/pandora.desktop** на Рабочий стол, исправьте пути внутри ярлыка при необходимости.

5. Ключ

Криптография (шифрование и электронная подпись) лежат в основе стабильной и безопасной работы Пандоры. Для работы криптографии нужен именной электронный ключ. Точнее говоря, пара ключей: открытый и закрытый.

При первом запуске, Пандора предложит сгенерировать новый ключ. Вместо создания нового вы можете загрузить старые ключи.

Ещё не реализовано:

Откройте список **Пандора-Ключи**, создайте новый ключ командой «**Создать**» [**Insert**]. В списке должны как минимум стоять галочки «**RSA**», «**Blowfish**», затем нажмите кнопку «**Сгенерировать**». Выберите свою анкету, к которой необходимо привязать данные ключи.

Использовать ранее созданный ключ можно командой «**Загрузить**».

Секретные ключи хранятся в зашифрованном виде. Активировать существующий ключ необходимо командой **Пандора-Авторизация**, при запросе ввести пароль. По умолчанию, Пандора активирует ключ при запуске. Здесь же можно изменить пароль или сгенерировать новый комплект ключей.

6. Человек

Для просмотра анкет откройте список **Мир-Люди**.

Если вы *впервые* используете Пандору, добавьте свою анкету. Рекомендуется как минимум ввести **имя**, **фамилию** и **дату рождения**. Постарайтесь сразу правильно указать свои данные, так как по ним формируется *уникальный идентификатор – панхэш*, а анкета человека и панхэш используются при генерации ключа. Также ваша анкета хранится вместе с ключами и будет загружена автоматически при повторной загрузке ключей с флешки.

7. Панхэш

Каждая запись в Пандоре имеет уникальный идентификатор, именуемый «**панхэш**». Панхэш состоит из усеченных хэшей полей. Текстовые поля хэшируются sha1, даты кодируются в 3 байта в днях от 01.01.1900, географическая координата кодируется в «катушечную» 4-байтовую координату (спираль от Северного полюса до Южного полюса), при ссылке на другие записи берется их панхэш.

Формула панхэша для каждого типа записи уникальна и задаётся в файле:

/opt/pandora/model/01-base.xml

атрибутами в виде hash="1:hash(3)", где:

1) первый «hash» – признак того, что это поле станет частью панхэша,

2) цифра «1» до двоеточия – позиция в панхэше,

3) второй «hash» – обработка поля. Доступны функции: sha1, date, raw, hash [равен sha1], coord, crc16, crc32, phash, pbirth, md5, sha256, sha224, sha384, sha512.

4) цифра «(3)» в скобках – длина компоненты панхэша, кооторую даёт это поле.

Например, панхэш **человека** (Person) состоит из следующих полей:

[type/lang:FirstName/LastName/BirthDate/BirthCity/FatherFN/MotherFN]

Наглядно панхэш может быть записан «сырыми» данными через слэш:

[person/ru: Линус/Торвальдс/28.12.1969/Хельсинки/Нильс/Анна]

Для человека (тип Person) байтовая формула и длина панхэша равны:

FLDCFM, 3+6+3+4+2+2 (+2 на тип и язык) = 22 байта

в 16-ричной кодировке панхэш выглядит так (пробелы вставлены для наглядности):

[0105: a78e3c 0d38fe3da470 0063c9 d59bab8c f573 435a]

Приведенный панхэш содержит следующие компоненты:

01 – тип записи «персона»

05 – русский язык

a78e3c, 0d38fe3da470, f573 и 435a – усеченный sha1 от «Линус» и «Торвальдс», «Нильс» и «Анна»

0063c9 – закодированная в 3 байта дата

d59bab8c – катушечная координата города «Хельсинки».

Некоторые поля могут быть пропущены (не заполнены), в этом случае участки панхэша заполняются нулями. Например, если заданы только имя, фамилия и дата рождения (FLD=12 байт), то панхэш будет выглядеть примерно так:

[0105: a78e3c 0d38fe3da470 0063c9 00000000 0000 0000]

концевые нули можно не указывать:

[0105: a78e3c 0d38fe3da470 0063c9]

Сокращенный панхэш в человеко читаемом виде может выглядеть так:

[персона: Линус/Торвальдс]

или в виде гиперссылки:

pandora://person.ru/Линус/Торвальдс

pandora://0105a78e3c0d38fe3da470

Если в сети одна и та же запись (т. е. запись с одинаковыми полями) была введена дважды (трижды и т.д.), её панхэш будет абсолютно одинаковым, и такая запись будет идентифицироваться как одна и та же.

Неполные записи (панхэши которых имеют нулевые пропуски) могут соотносится с более полными (у которых панхэши имеют меньше пропусков). Такое соотношение называется подобие. Поиск подобия может быть полезен в запросах и настраивается дополнительно.

В Пандоре базовые типы записей (как правило, это мировые записи, такие как «человек», «сообщество» и т.п.) описываются с нуля. Остальные типы записей порождаются от базовых (в основном это деловые и региональные записи, например «партнер», «компания»).

Дочерние записи, порожденные от других типов (например, «сотрудник» от «персона») имеют удлиненный панхэш относительно базового типа, удлиненный на добавленные поля. Кроме того, порожденные записи не содержат в себе данные родительский полей. Родительские данные хранятся в записи базового типа, а в потомке хранится только ссылка на панхэш родителя, плюс хэши дополнительных полей. Например, если вы заводите сотрудника «Иван Иванов, менеджер», то создаются две записи: персона Иван Иванов и сотрудник с панхэшем Ивана Иванова и дополнительное поле – должность. Панхэш сотрудника будет представлять собой панхэш персоны, плюс хэш поля «должность». Панхэш сотрудника может выглядеть так:

[партнер: Иван/Иванов/////менеджер]

Записи представлены в Пандоре в виде объектов. К дочерним объектам применимы все методы родителя. Например, если ищется человек «Иван Иванов», то он будет найден, даже если вводился только на деловом уровне в виде сотрудника. С другой стороны, при начале трудовой деятельности не придется вводить данные человека, если они существуют на мировом уровне.

Такая иерархия позволяет: 1) вводить данные только один раз, 2) экономить дисковое пространство, 3) иметь сквозную единую идентификацию объектов, 4) применять методы родительских классов.

Пандора вычисляет бинарные (байтовые) панхэши, и в своей работе оперирует ими при идентификации и поиске объектов.

8. Слушание

Обычно Пандора при запуске сама входит в режим слушания. Ручное включение и выключение режима слушания доступно командой **Пандора-Слушать**. В режиме слушания внизу в статусной строке высветится ваш **порт** и, если удалось подключиться к другим узлам, **IP-адрес**.

Теперь вы можете сообщить друзьям свой IP-адрес, чтобы они добавили вас в свой список.

Когда ваша Пандора находится в режиме слушания, другие пользователи могут подключаться к вам, считывать и записывать данные на которые у них есть разрешения.

Сразу после установки Пандоры заданы минимальные разрешения. Как изменить разрешения описано ниже в главе «Доверие, разрешения и подписки».

9. Узел

В списке **Пандора-Узлы** добавьте IP-адрес узла, который вам удалось узнать. Достаточно указать только **IP адрес** (или **Домен**) и **TCP порт** – другие поля будут заполнены автоматически при обмене данными.

После добавления узла нажмите Узел-Охотиться (выкл/вкл), чтобы соединиться с новым узлом.

Ещё не реализовано:

Не беспокойтесь, что вам постоянно придется вручную вводить адреса. В основном таблица адресов будет заполняться автоматически при обмене данными с другими узлами.

10. Сообщения и звонки

Пандора поддерживает обмен сообщениями, голосовой разговор, видео звонок и игры. Есть несколько способов начать живой разговор.

В списке **Мир-Люди** выберите нужного человека и откройте диалог командой «**Диалог**» [Ctrl+D]. Аналогичная команда есть в списке **Пандора-Узлы**.

При открытии диалога и включении галочки «онлайн» Пандора подключается к узлу собеседника, если не была подключена до этого. Если соединение установлено, то сообщения сразу будут уходить собеседнику. Иначе сообщения будут копиться и будут отправлены, когда появится связь с узлом, например при очередной охоте (см. главу «Охота»).

Помимо шифрования трафика на сессионном ключе (BT) при отправке сообщения можно включить дополнительное шифрование на ключе получателя (RSA). Также можно шифровать локальную историю на активном ключе отправителя (RSA). Пандора запоминает ваш выбор при закрытии диалога.

Каждое сообщение в истории можно хранить шифрованным или расшифрованным, меняя галку «зашифровано» напротив сообщения в окне диалога. Еще у каждого сообщения есть галки «отправлено», «получено» и «прочитано», которые также можно менять, вновь запуская механизмы Пандоры.

Если соединение установлено, то можно начать голосовой или видео звонок, включив галочки «**звук**» и «**видео**».

Звук и видео работают только на Ubuntu 12.04 с Gstreamer 0.1 – в новых версиях API гстримера сломали, и работать перестало.

Ещё не реализовано:

Кроме того можно поиграть, нажав «**игра**» и выбрав «Морской бой», «Шахматы» и другие игры.

11. Охота

Режим охоты включается командой **Пандора-Охотиться**.

Узел Пандоры перебирает известные узлы, и как бы выходит на охоту. При этом охотник ищет слушающие узлы. Если «охотник» нашел «слушателя», он подключатся к нему и начинает

обмен данными.

Хотя подключение всегда инициирует «охотник», обмен данными между «охотником» и «слушателем» идёт в обе стороны.

Нормальная ситуация, когда Пандора находится и в режиме слушания и в режиме охоты одновременно. Роль определяется только тем, кто первый подключился. Сразу после подключения «охотник» и «слушатель» находятся в равных условиях.

Соединение или разрывается сразу после обмена данными, или остаётся подключенным, если было запрошено живое общение (чат, звонок или игра).

Пандора в режиме охоты циклически опрашивает только те узлы, на которые вы подписаны. О подписках рассказано ниже в главе «Доверие, разрешения и подписки».

12. Доверие

Доверие – это подписание анкеты или другой записи своей цифровой подписью с указанием коэффициента доверия от -1.0 до +1.0. Для разных типов записей доверие имеет разный смысл:

Тип записи	Доверие от +1 до 0	Доверие от -1 до 0
Человек, Сообщество, Страна	Конструктивный и ответственный, представляет пользу для общества	Деструктивен и безответственен, бесполезен или даже вреден для общества
Ключ	Хранится в надёжном месте и не может быть украден	Хранится в ненадёжном месте, может быть украден и использован во вред
Статья, Мнение	Поучительна, хорошо оформлена	Идея не прослеживается, безобразное оформление
Файл, Картинка	Полезен к применению	Бесполезен
Город	Удобный для жизни, красивый, перспективный	Неудобный, безобразный, нет перспектив
Проект	Детально и ясно прописан, готов к реализации, полезен для общества	Прописан смутно, к реализации не готов, бесполезен для общества
остальные	Удостоверяю, принимаю, ручаюсь что это полезно	Отвергаю, ручаюсь что это вредно

Доверие равное 0.0 означает, что вы видели эту запись и считаете её нейтральной.

Когда вы меняете доверие к записи, старая подпись остаётся, и рядом заводится новая подпись. Таким образом подписей может быть много. История подписей служит показателем эволюции вашего мнения. Но при расчёте рейтингов используется только самая последняя по времени создания подпись.

Перед тем как подписывать ключи реальных людей, обязательно убедитесь в истинности ключа, позвонив человеку и сопоставив хэши ключа. Этим вы обезопасите себя от атаки «человек посередине». Косвенно, реальность ключа может быть подтверждена высоким рейтингом в вашей системе доверия.

Полной защитой от подделки ключа может служить только непосредственная передача ключа на флешке при личной встрече с человеком, и последущая загрузка ключа в вашу Пандору.

При этом всегда передаётся только открытая часть ключа. Никогда не показывайте никому

закрытую часть ключей. По умолчанию Пандора не выгружает закрытые ключи. Также опасайтесь вирусов и троянов, ворующих закрытые ключи. Злоумышленник, получивший ваш закрытый ключ, выдавая себя за вас, может сильно подмочить вашу репутацию.

Если вскрылось, что ваш закрытый ключ украден, срочно оповестите людей об этом и сгененируйте новый ключ. После чего укажите недоверие к старому ключу со стороны нового, при этом дату создания доверия задайте равной дате потери ключа.

С другой стороны, если вы узнали, что чей-то ключ, которому вы раньше доверяли, украден, как можно быстрее отразите это в Пандоре понижением доверия до -1, т. е. созданием нового отрицательного доверия.

После первого подключения «охотник» и «слушатель» обмениваются ключами пользователей и их анкетами. После чего обмен записями приостанавливается до тех пор, пока пользователи не установят доверие, не настроят разрешения и подписки. Тем не менее, вы можете пообщаться с человеком, которому вы не доверяете, если это разрешено вашими настройками.

Разрешения

Разрешения задают доступ к записям и складывается на пересечении условий:

1) доверие человеку или сообществу

2) указание маски типов записей, котрые могут быть доступны

Подписки

Подписки определяют данные какого типа необходимо запрашивать у заданных сообществ, людей или узлов.

Подписки задаются связями типа «следит за».

Примеры подписок Ивана Иванова:

[связь: [персона:Иван Иванов]/«следит за»/[персона:Пётр Петров]]

[связь: [персона:Иван Иванов]/«следит за»/[сообщество:Клуб любителей кошек]]

13. Обмен записями

Обмен записями (Людьми, Городами, Статьями и т. д.) возможен по одному из четырёх алгоритмов.

1. Обмен по подпискам.

Для этого создаётся Связь: Человек (т.е. вы) «следит за» Город (допустим, Тверь).

2. Обмен по публикациям доверенных.

Ваш доверенный создаёт Связь: Человек (ну например, я) «публикует» Статью (например, «Сеть доверия»).

3. Обмен по подписям от доверенных.

Например, ваш доверенный подписывает Статью с доверием выше 0 (создаёт запись Подпись).

При сеансе связи узлы в первую очередь обмениваются записями Связь вида «публикую» и «слежу», а также Подписями ваших доверенных. Также узлы сообщают друг другу какие типы записей они собирают (задаётся в настройках, например Город, Статья, Товар, Услуга). После чего каждый узел смотрит у себя в базе, какие записи из запрошенных типов были затронуты по этим связям и подписям. Затем узлы составляют списки панхэшей, которые они могут предложить, а также какие им нужны, и посылают друг другу эти списки. Ну и в финале по панхэшам запрашиваются превьюшки/анонсы записей или сами записи целиком — это тоже задаётся в настройках. Если юзер поглядел превьюшку/анонс и хочет полную запись, то нажимает «смотреть всю запись», запись запрашивается и подгружается.

Далее юзер может оставить на хранение запись, поручиться за неё, опубликовать или поставить слежение за ней:

4. Запрос записи по параметрам.

Например, вы ищете Ивана Петрова. Открываете Узел-Поиск,

указываете тип записи — Человек, Имя — Иван, Фамилия — Петров, и нажимаете «Начать поиск». После чего ваша Пандора создаёт «Запрос» и рассылает его в рамках охоты всем доверенным узлам (они могут дальше по цепочкам доверия разослать). Каждый узел, получив запрос, глядит в свою базу. Когда какой-то узел в цепочке находит запись по Запросу, он высылает запись (или анонс) обратно по цепочке (или напрямую, если может).

Через какое-то время к вам приходит нужна запись.

Допустим пришла запись «Человек: Иван Петров». Теперь вы хотите с ним поговорить. Нажимаете «диалог» на этом человеке и Пандора формирует новый Запрос — найти Узел, принадлежащий человеку с таким-то панхэшем. Запрос на узел также расходится по веткам доверия. Через какое-то время к вам приходит запись «Узел: владелец — Иван Петров». Пандора соединяется с узлом (напрямую или через «рыбалку») и вы начинаете общаться.

14. Рейтинг

В Пандоре люди оказывают положительное или отрицательное доверие записям.

Рейтинг записи вычисляется на основе суммы доверия от разных людей, при вычислении рейтинга могут использоваться разные алгоритмы, которые делятся на две группы:

1) простая сумма доверия валидных подписей
2) сумма с весовыми коэффициентами, расчитанными с учётом личных веток доверия (или веток доверия заданного пользователя).

Простая сумма изначально кажется самой объективной. Но на самом деле такой рейтинг по-хорошему нигде не может быть учтён, так как без учёта системы доверия нет никакого способа

отличить подписи реальных людей от подписей подставных лиц. Но даже если бы все подписи были реальными, в «простом» подсчёте деструктивные члены общества получают одинаковые права с конструктивными, что в итоге будет вредно для общества.

Простой рейтинг (без учёта схем доверия) рассчитывается в Пандоре только как отвлеченная статистическая величина. Что-то наподобе средней по больнице температуры.

Релевантный рейтинг, рассчитанный с учётом системы доверия конкретного человека, применяется, во-первых, самим человеком, во-вторых, людьми, которые ему доверяют и хотят учесть его оценку в своих расчётах рейтинга записи.

Таким образом, простановка доверия к записям в Пандоре имеет важнейшее значение для конструктивного развития системы, по крайней мере в той области, в которой вы принимаете участие. Относитесь ответственнно к своим ключам и к своим подписям.

15. Мнение

Каждая запись может быть прокомментирована и оценена участниками Пандоры. Для этого создается специальная запись — мнение. Мнение содержит панхэш комментируемого объекта, панхэш создателя мнения, текст комментария и время создания мнения.

Мнение может комментировать другое мнение. Так образуются ветки дискуссии.
Вы можете выключить отображение мнений тех, кто имеет низкий рейтинг в вашей системе доверия. Исключение из вида мнений низкорейтинговых людей позволит сэкономить время, нервы и дисковое пространство, настраивая вас на позитивный и конструктивный лад.

16. Задача

Пандора может служить в качестве планировщика заданий и надпоминателя.
Укажите «Время» в которое должно начаться задание.
Задайте «Режим» выполнения:
1 – один раз вывести сообщение на экран.
2 – повторять сообщение с интервалом 5 минут.
3 – выполнить код, указанный в сообщении.
По умолчанию «Исполнителем» являетесь вы, но можно изменить исполнителя, если вы даёте задание другому человеку или группе.
Также исполнителем может быть ваш узел. В этом случае сообщение содержит правила выполнения задания, а «Режим» определяет, для какого блока дано задание:
1 – задание для «почемучки» (управляет сбором данных)
2 – задание для «ответчика» (управляет рассылкой данных)
3 – последовательность действий на узле в виде заданных команд меню.

17. Связь

Связь – это запись, показывающее отношение между двумя записями; содержит следующие поля:
1) панхэш первой записи
2) панхэш второй записи
3) тип связи (1 байт)

На текущий момент зарезервированы следующие типы связей, код означает «первая запись имеет следующее отношение ко второй»:
0 – неопределенная связь
1 – равно
2 – подобие (синоним)
3 – антипод (антоним)
4 – входит в состав
5 – породил
6 – следит за
7 – игнорирует
8 – пришёл от
235..255 – публикация (21 уровень) от всем (255) до только своим (235).

Примеры связей:
[персона:Линус/Торвальдс] 1 [персона:Линус/Торвальдс/28.12.1969]
означает, что любой Линус Торвальдс скорее всего тот самый.
[слово:классное] 2 [слово:клёвое]
слова «классное» и «клёвое» очень похожи
[слово:горячее] 3 [слово:холодное]
«горячее» антоним слову «холодное»
[персона:Линус/Торвальдс] 4 [сообщество:Разработчики ядра Linux]
Линус входит в состав сообщества
[персона:Линус/Торвальдс] 5 [персона:Патриция/Торвальдс]
Линус родитель Патриции
[сообщество:Жильцы дома №98] 6 [проект:Строительство детсада №7]
Жильцы дома следят за разработкой проекта садика в их дворе
[персона:Геннадий/Редискин] 7 [статья:P2P социальная сеть Pandora]
Геннадий равнодушен к некоторой статье, и не хочет больше получать эту запись при обмене с любыми узлами.
[персона:Линус/Торвальдс] 255 [файл:linux.zip]
Линус опубликовал файл для всех

Связи могут использоваться в различных случаях, таких как: определение состава сообщества, выявление синонимов, причинно-следственные связи, работа «охотника» и т.д. Связям, как и другим записям, также может быть оказана поддержка и доверие.

18. Правка и удаление

Обычное изменение записи происходит по-разному, в случаях если запись:

не была отправлена

– при удалении ставится пометка «удалена», исчезает из видимости пользователя, через неделю сборщик мусора удалит запись окончательно;

– при редактированиии изменяется непосредственно сама запись.

была отправлена

– при удалении ставится пометка «удалена», исчезает из видимости пользователя, через месяц сборщик мусора удалит запись окончательно;

– при редактированиии создаётся новая копия, которая и редактируется; если запись ваша, то старая запись помечается как «удалена», если запись чужая, то она остаётся в неизменном виде; создаётся связь типа «порождён от».

Если вы удалили запись, и она скрылась из видимости, получив пометку «удалена», то при обмене с другими узлами эта запись больше не будет загружаться. Но если сборщик мусора удалил запись, то при охоте запись загрузится к вам в базу.

Если вы насегда хотите избавится от записи, то при удалении выберайте пометку «игнорировать в дальнейшем», в этом случае будет создана связь «вы игнорируете запись», и после того, как сборщик мусора её удалит, запись не будет загружаться, так как существует связь «игнорирует».

Если вы хотите удалить запись минуя кэширование, то устанавливайте галочку «физически».

Если при редактировании вы гарантированно хотите сохранить старую запись, то выбирайте команду «Копировать», вместо «Редактировать».

Если вы хотите восстановить удаленную запись, то включите в списке «показывать удаленные» и выберите команду «Восстановить». При этом запись «игнорирует» будет удалена согласно общему механизму удаления.

Если вы не поставили пометку «храню» или «ручаюсь», то запись считается временной. Временные записи автоматически удаляются сборщиком мусора спустя две недели.

Сборщик мусора работает сам по себе, без участия пользователя.

19. Файл (статья, картинка)

Файлы, в том числе статьи и фотографии, публикуются в таблице **Мир-Файлы**.

Пример:

1) название файла: «Мой кот Барсик»

2) тип файла: «JPG» (обычно определяется автоматически)

3) путь к файлу: /home/user/Pictures/cat_barsik.jpg

Статьи можно создавать и редактировать прямо в Пандоре на вкладке «Тело».

Поддерживается четыре формата: bbcode, html, wiki, org-mode, plain.
Формат plain — это простой текст в кодировке UTF-8.
Форматы org-mode поддерживает следующие теги:
выделенный
/наклонный/
подчёркнутый
-зачёркнутый-
  моноширинный
+тоже моноширинный+
`текст *без* /форматирования/`
> цитата
Читать далее ->
* Список
- Список
Список нумерованный
=== Глава
== Заголовок
= Подзаголовок
===# Глава с номером
[Выравнивание
по центру]<>
[Выравнивание по ширине]<=>
[Выравнивание вправо]>
[Выравнивание влево]<
[]>
С этой строки -- выравнивание вправо
После этой строки --- влево (начиная с таблицы)
[]<
=№	=Фамилия	=Рост
1	Иванов	175
2	Петров	180

[Развернуть или свернуть ссылки]->
http://robux.biz/pandora.html
mailto://ironsoft@mail.ru

pandora://person/Linus/Torvalds
pandora://персона/Линус/Торвальдс
pandora://article.ru/Народная сеть Pandora/0xfd94e3a95c12
[http:robux.biz/pandora.html|Страница с реквизитами]
[mailto:ironsoft@mail.ru|Ironsoft Lab]
[pandora:person.ru/Линус/Торвальдс]
[person.ru/Линус/Торвальдс|Клёвый чувак]
[персона/Линус/Торвальдс]
[статья/Народная сеть Pandora/[Михаил/Галюк]]
[персона/0x1d71e3a95c45]:the_mark1:
[0x01051d71e3a95c45]
[file:/etc/crontab.conf@size]<>
[file:pandora/pandora.rb|Главный файл@ruby]->
[file:files/download_by_pandora.png@image]
[file:home/facepalm.jpg@image/640x480|Лицо ладонь]
[]
[@ruby]
puts 'Hello World'
Форматирование завершится пустыми квадратными скобками
[]
[#the_mark1|Перейти к метке]
{+red,black}красный текст на черном фоне{-}
{:Сноска внизу страницы}
{:[http:google.ru/search?&q=p2p+сеть|Сноска с ссылкой на источник]}

Формат BBCODE и Html поддерживают теги:
B, I, U, S, EM, STRIKE, STRONG, D, BR,
FONT, SIZE, COLOR, COLOUR, STYLE, BACK, BACKGROUND, BG,
FORE, FOREGROUND, FG, SPAN, DIV, P,
RED, GREEN, BLUE, NAVY, YELLOW, MAGENTA, CYAN,
LIME, AQUA, MAROON, OLIVE, PURPLE, TEAL, GRAY, SILVER,
URL, A, HREF, LINK, ANCHOR, QUOTE, BLOCKQUOTE, LIST,
CUT, SPOILER, CODE, INLINE,
BOX, PROPERTY, EDIT, ENTRY, INPUT,
BUTTON, SPIN, INTEGER, HEX, REAL, FLOAT, DATE,

TIME, DATETIME, COORD, FILENAME, BASE64, PANHASH, BYTELIST,

PRE, SOURCE, MONO, MONOSPACE,

IMG, IMAGE, VIDEO, AUDIO, FILE, SUB, SUP,

ABBR, ACRONYM, HR, H1, H2, H3, H4, H5, H6,

LEFT, CENTER, RIGHT, FILL, IMAGES, SLIDE, SLIDESHOW,

TABLE, TR, TD, TH,

SMALL, LITTLE, X-SMALL, XX-SMALL, LARGE, BIG, X-LARGE, XX-LARGE.

Разница заключается в использовании квадратных или угловых скобок:

```
[b]это жирный текст в bbcode[/b]
<b>это жирный текст в html</b>
```

Текст редактируется в режиме «Правка», а отображается в режиме «Просмотр»,

быстрое переключение возможно клавишей F5.

20. Параметр

Параметры задают работу Пандоры и доступны в списке **Пандора-Параметры**.

Параметры привязываются к узлу и к пользователю. Параметры могут перемещаться между узлами, согласно веткам доверия. Это позволяет восстанавливать свои настройки при переходе с одного места на другое.

При запросе по имени параметра значение берётся у параметра с максимальным доверием для текущего пользователя и текущего узла.

Когда пользователь не задан, считается, что параметр предназначен для всех людей.

Когда узел не задан, считается, что это параметр по умолчанию для всех узлов.

21. Объявление

Добавьте своё объявление в списке **Дело-Объявления**.

Укажите «Название», например «Продаётся ноутбук Acer PN-2000» и «Суть» объявления, например «Монитор: 15", процессор AMD, жёсткий диск 500Гб. Цена 10 000 руб.».

Если простой «Формат» объявления не устраивает, выберите «Табличный». Здесь вы сможете указать список товаров и их цены.

22. Заказ

Щелкнув по нужному объявлению правой кнопкой мыши, выберите «Заказать».

В закладке «Блага» укажите что именно вы хотите заказать и в каком количестве.

Здесь также поддерживается «Простой» формат и «Таблица».

23. Сделка

Щелкнув по нужному заказу правой кнопкой мыши, выберите «Оформить».

В открывшейся «Сделке» выберите «Договор».

На базе этой сделки вы можете распечатать «Счет для оплаты».

24. Накладная

Щелкните правой кнопкой мыши на Сделке и выберите «Отгрузить».

25. Расписка и гаранты

В основе платежей в Пандоре заложена «Распределённая расписочная платёжная система» (Distributed debenture payment system, или DDPS).

Оплата в Пандоре совершается передачей расписок. О Передаче будет сказано ниже.

Расписка – это обязательство кого-либо предоставить услуги или товары в указанном объёме.

Вам предстоит создать Расписку от своего имени, когда у вас нет накопленных платежных средств в виде расписок других эммитентов, или когда получатель желает получить оплату именно в вашей расписке.

Укажите «Договор», в котором прописаны условия предоставления вами товаров или условия выполнения услуг. В договоре должны быть прописаны виды товаров и услуг, максимальные объемы и сроки исполнения. В дальнейшем вам предстоит выполнить условия договора предъявителю расписки. Подробнее о договорах, регулирующих реализацию расписок, будет рассказано ниже.

Укажите «Валюту», например «МэДж» или «руб.2013.07» или «BTC.2013.07». Если сумма указана в виде энергии или материальной ценности (например «МэДж» или «золото.585»), то стоимость расписки постоянна на любую дату. Если сумма указана в материально необеспеченных валютах (рубли, доллары, евро, биткоины и так далее), ценность которых определяется торгами на валютных биржах, то дата расписки (день, месяц и год) имеет важное значение: в дальнейшем такая расписка будет переоцениваться на дату операции. В договоре могут быть указаны дополнительные условия, по которым производится пересчет «дрейфующих» валют.

Если вы выпускаете расписку, то должны понимать, что обязаны её в дальнейшем выполнить. Если вы не выполните свои обязательства, это скажется на доверии к вам, снизит ваш рейтинг в системе, уменьшит вашу привлекательность и, в крайних случаях, может служить причиной наказания.

Для повышения ликвидности, а значит и привлекательности расписок, существуют ещё три документа, являющимися гарантами, работающими для возмещения необеспеченных расписок:

1) Залог – выставление некоторой материальной ценности (товара, недвижимости, авто и так далее) для возмещения необеспеченной расписки;

2) Поручительство – заявление одного человека, что он берет на поруки другого. В случае срыва расписок у подопечного, поручитель должен будет покрыть его расписки;

3) Страховое общество – сообщество, которое берет на поруки своих членов. В случае срыва расписок члена общества, другие члены должны будут покрыть его расписки.

Расписка может выпускаться одним документом на всю сумму, или пачкой более мелких для удобства их использования получателем. Если расписка выпущена крупной суммой, то эммитент обязан разменивать (т. е. обменивать на несколько расписок меньшего номинала) при запросе любого держателя расписки. Обычно разменом занимается биржевой робот эммитента, который работает на его узле в автоматическом режиме. О биржах рассказывается ниже.

26. Передача (платёж)

Передача – это документ, по которому имеющаяся у вас расписка передаётся другому лицу. В Пандоре Передача используется как способ оплаты, а также как способ обмена расписками. Обмен расписками может производиться в ручном или автоматическом (через биржи) режиме. О биржах будет рассказано ниже.

Щекните правой кнопкой мыши по Сделке и выберите «Оплатить».

Выберите «Расписку», которую вы хотите передать в качестве оплаты.

Сумма передачи берётся из Сделки. Если сумма сделки больше суммы расписки, то расписка передается всей суммой, а остаток передаётся другой распиской.

Если сумма сделки меньше суммы расписки, то от расписки «откалывается» часть суммы. Крупная расписка разделяется на части при обращение к эммитенту этой расписки с запросом размена на две части: на сумму сделки и на остаток. Располовиниванием обязан заниматься биржевой робот эммитента в автоматическом режиме.

Если узел эммитента временно недоступен (или недоступен вообще), то получатель платежа принимает всю расписку, а на сдачу выдаёт другую расписку, которая по уровню доверия устроит исходного плательщика. При этом расписка на сдачу также может быть располовинена.

В крайнем случае сдача может быть выдана распиской самого получателя платежа. Механизм проведения оплаты определяется настройками узла, а также настройками и доступностью бирж эммитентов расписок, учавствующих в операции.

Передача должна быть подписана как плательщиком (покупателем), так и получателем (продавцом). Если получатель (продавец) не оказал вашей (как покупателя) Передаче положительного доверия, значит он не принял оплату в виде этой расписки и ему нужна другая расписка. А значит, вы должны отменить передачу и сделать новую передачу с указанием другой имеющейся у вас расписки. Подбор расписок, устраивающих получателя, а также нахождение и обмен других расписок (в случае отсутствия у вас надёжных с точки зрения

получателя) может выполнять биржевой робот в полуавтоматическом или автоматическом режиме.

На стороне получателя Передачу может принять также биржевой робот получателя, если он расценил предлагаему расписку как ликвидную. В таком случае приём оплаты может происходить в полностью автоматическом режиме.

27. Биржа

Биржа работает в автоматическом или полуавтоматическом режиме. В автоматическом режиме биржевой робот по заданным правилам находит выгодные операции, создаёт Передачи и Расписки (или только что-то одно) и подписывает их заданным ключом (для автоматических биржевых операций рекомендуется создать отдельный комплект ключей). Это самый быстрый режим работы биржи. В полуавтоматическом режиме робот только подготавливает Передачи и Расписки, а человек после личной проверки подписывает (или не подписывает) их. Такой режим биржи более медленный, но более надёжный. Возможен комбинированный вариант: робот подписывает биржевым ключом, а человек после проверки, дополнительно подписывает личным ключом.

Биржевой робот в процессе охоты постоянно ищет на других узлах более ликвидные расписки, чем те, которыми располагает, и пытается сделать обмен. Ликвидность расписок зависит от уровня доверия к их эммитентам, а также от текущей заданной потребности в некоторых товарах или услугах. Всё это задаётся настройками биржи. Получается, что две биржи на разных узлах оценивают расписки друг друга и заключают взаимовыгодный обмен. Обоюдность обеспечивается тем, что одни расписки могут быть выгодны для одного, а другие для другого.

Если ваш биржевой робот нашел более ликвидную расписку на другом узле, он начинает с ним торговаться, предлагая ваши расписки. В итоге, если оба робота нашли друг у друга более ликвидные для себя расписки, они могут: а) выставить уведомления владельцу ключа; б) провести обмен в автоматическом режиме, выписав друг другу Передачи.

Также функцией биржи является размен (на сдачу) своих расписок, т. е. биржевой робот: а) принимает (через Передачу) свою расписку, б) выписывает две (или более) новые расписки на такую же общую сумму и на тот же договор, в) передает новые расписки тому, кто запрашивал размен.

Режим работы биржи задают владельцы узла и соответствующих ключей.

28. Загрузка данных

Вы можете загружать записи из внешних файлов (например из xls-таблиц) или баз данных (например из баз CRM-систем или интернет-магазинов).

Щелкните правой кнопкой по любому списку и выберите «Загрузить».

29. Выгрузка данных

Вы можете выгружать записи во внешние файлы (например в xls-таблицы) или в базы данных

(например базы CRM-систем или интернет-магазинов).

Щелкните правой кнопкой по любому списку и выберите «Выгрузить».

30. Проект

Вы можете разработать новый проект в списке **Регион-Проекты**.

Это может быть проект детского сада, электромобиля, бытового прибора, нового моста через реку и так далее.

31. Постановление

Щёлкните правой кнопкой по Проекту и выберите «Реализовать», чтобы создать новое постановление.

В постановлении кроме проекта указываются скроки выполнения, ответственный реализатор, способ финансирования (краудфандинг, донейт или бюджет). Также могуть быть указаны и подрядчики.

Постановление может касаться вашего дома, города, района, области или страны. Естественно, вы должны понимать, что постановление должно быть адекватным, чтобы большинство людей приняло его.

Процедура утверждения постановлений может различаться для разных регионов.

32. Закон

Вы можете предложить новый закон в списке **Регион-Законы**.

Закон может касаться вашего города, района, области, страны или международный закон. Естественно, вы должны понимать, что закон должен быть адекватным, чтобы большинство людей приняло его.

Процедура принятия законов может различаться для разных регионов.

33. Преступление

Преступление – это фиксация некоего факта, который нарушает принятые законы или моральные нормы. Во втором случае не обязательно будет следовать Наказание.

Преступлением может также являться невыполнение эммитентом обязательств по Расписке.

34. Наказание

Наказание – это реакция на Преступление, согласно принятым Законам.

Щелкните правой кнопкой на Преступлении и выберите «Наказать».

Выберите «Исполнителя» и срок исполнения. Также укажите ссылки на законы.

Процедура запуска наказаний может различаться для разных регионов.

35. Делегирование

Делегирование – это передача права голосовать по какому-либо Постановлению, Закону или Наказанию другому лицу (делегату).

Укажите срок, задав в поле «Истекает» конечную дату. Начальной датой служит время создания Делегирования.

Учёт делегирования при голосовании может учитываться по-разному для разных регионов.

36. Регистр

Регистры используются для регистрации номеров каких-либо объектов: автомобилей, вагонов и другое.

Созданием регистров обычно занимается узел общественной организации, которую утвердили соответсвующим Постановлением.

37. Обмен с внешними базами

Чтобы автоматически загружать данные из файлов, каталогов или внешних таблиц добавьте задание в таблице **Пандора-Загрузка/Выгрузка**.

Укажите «Ресурс», задайте параметры отслеживания: «Время правки», «Размер», «Число записей» (для таблицы), «Изменение записей» (для выгрузки); также задайте интервал работы

и частоту отслеживания.

Пандора отследит изменения и запустит загрузчик или выгрузчик данных, после чего зарегистрирует состояние загрузки/выгрузки в таблице **Пандора-Загружено/Выгружено**.

Автоматическая загрузка может быть полезна при загрузке списка «Города», «Слова», «Товары», «Заказы», а выгрузка при выгрузке таблиц «Люди», «Товары», «Законы», например в базу данных интернет-магазина, и так далее.

38. Настройка сети

Pandora не имеет серверов, весь трафик идёт напрямую между клиентами сети. Чтобы к вам могли подключаться ваши друзья необходимо настроить на своём роутере проброс TCP-порта 5577 снаружи вовнутрь. Для «белого» и статического IP-адреса этого достаточно.

Если ваш IP-адрес «белый», но динамический, то можно воспользоваться любым сервисом DDNS для получения постоянного доменного имени.

Если ваш IP-адрес «серый» (находитесь за NAT), но при этом имеете хостинг с питоном, то можно запустить на нём отдельную утилиту pangate.py (находится в подкаталоге util).

Если ваш IP-адрес «серый» и вы не имеете никаких хостов с белыми IP, то ваш клиент всё равно сможет подключаться к другим клиентам Пандоры. Также, даже будучи за NAT'ом, почти у каждого провайдера имеется возможность получить «белый» IPv6 адрес, используя сервис Teredo. Для его запуска в линуксе достаточно поставить пакет «miredo».

39. Библиотеки

Библиотеки, которые использует Пандора, удовлетворяют следующим критериям:

1) свободная и чистая лицензия;

2) независимость от вендора;

3) кросс платформенность;

4) популярность и актуальность;

5) ясность;

6) документированность;

7) простота развертывания;

8) малый размер.

Текущий список используемых библиотек: ruby, gtk, sqlite, openssl, gstreamer.

Форматы: txt, xml, odf.

Протокол бинарный на порту tcp/udp — 5577.

Ниже описаны дополнительные критерии, которые стали ключевыми при выборе:

1. Ruby.

Код на Ruby эстетичен и лаконичен. Написано множество библиотек: пользовательские интерфейсы (gtk, ncursed, web, qt, fox, wx).

2. GTK

Хорошая документированность по отношению к другим графическим библиотекам. По умолчанию присутствует в любом GNU/Linux.

3. SQLite

Простота использования. Фактически не нужно устанавливать.

4. OpenSSL

По умолчанию присутствует в любом GNU/Linux.

5. GStreamer

Гибкость. Простота использования.

6. XML

Удобство редактирования.

7. ODF

Свободный стандарт.

8. GPL

Гарантия свободы и открытости. Защита от притязаний.

9. Бинарный протокол

Компактный трафик. Высокая скорость обмена.

10. Порт 5577

Красивые цифры 5 и 7. По статистике активности, порт 5577 в интернете не используется никакими другими службами.

40. Языковые файлы

Сообщения в Пандоре введены на английском. При запуске, если в системе установлен другой язык, из подкаталога /lang загружается соответсвующий языковой текстовый файл. Например, чтобы сообщения выводились на русском языке необходимы два условия:

1) задать переменную среды LANG=ru_RU (или запуск с параметром --lang=ru)

2) создать языковой файл **/Pandora/lang/lang.ru.txt** в таком виде:

```
Pandora=>Пандора
Folk network of trust=>Народная сеть доверия
"Warning! \"Evil Empire\" wants
```

```
to own the World!"=>"Внимание! \"Империя Зла\" хочет
владеть Миром!"
```

Здесь левая часть отделена от правой символами «=>». Многострочный текст заключается в двойные кавычки, при этом переводы строк воспринимаются как есть. Внутренние кавычки экранируются слэшем \".

После создания текстового файла никаких дополнительных действий не требуется. Пандора автоматически подгружает соответствующий языковой файл при запуске и согласно ему переводит все сообщения. Если перевод не найден, сообщение выводится на английском языке.

Если вы создали свой языковой файл, то пожалуйста поделитесь им с другими. Например, вышлите файл автору Пандоры для включения в следующий выпуск. В будущем планируется распростанять языковые файлы средствами самой Пандоры, в этом случае языковой файл будет «подтягиваться» с максимальным рейтингом согласно вашей схемы доверия.

41. Модель Пандоры

Модель описана в файлах: **/Pandora/model/*.xml**. Модель задаёт типы объектов, описание их полей, вид отображения в талице и способ редактирования в форме ввода. Объекты разнесены по четырём группам:

1. Мир

Человек, Сообщество, Город, Страна и др.

2. Дело

Товары, Услуги, Сделки и др.

3. Регион

Проекты, Законы, Ресурсы и др.

4. Пандора

Узлы, Ключи, Подписи и др.

Вы можете дополнять базовые типы, или создавать новые типы, унаследованные от базовых.

Когда вы поменяли описание модели или структуру базы, Пандора обнаруживает несоответствие и предлагает преобразовать таблицу в соответствие с описанием. При этом появляется диалог преобразования, в котором вы можете управлять процессом преобразования, например, новые поля можно проинициализировать каким-то постоянным значением или комбинацией других столбцов.

42. Средства разработки

Компоненты Пандоры не требуют линковки и компиляции. Вы открываете файлы (*.rb, ,*.xml, /lang/*.txt) в простом текстовом редакторе, изменяете текст, сохраняете — и запускаете программу.

В качестве текстового редактора рекомендую Geany: лёгкий, эстетичный, кроссплатформенный с подстветкой синтаксиса ruby, xml и многих других.

В Ubuntu он устанавливается командой:

apt-get -y install geany

В Windows можно скачать и установить отсюда:

http://www.geany.org/Download/Releases

43. Адаптер баз данных

Адаптер является посредником между базой данных и бизнес-логикой, он позволяет абстрагироваться от особенностей базы.

44. Формат данных PSON

PSON (Pandora Simple Object Notation) – бинарный формат упаковки данных, позволяющий упаковывать (то есть переводить в последовательность байт, в простую строку) значение одного из 9 типов:

1. Целое число (Integer)
2. Дробное число (Float)
3. Строка (String)
4. Логическое (Boolean)
5. Дата и время (Time в Ruby или Datetime в Python)
6. Массив (Array в Ruby или List в Python)
7. Словарь (Hash в Ruby или Dict в Python)
8. Символьное (Symbol в Ruby)
9. Пустое значение (Nil).

Поддерживаются вложенные значения, например:

value = ['Hello', 1500, 3.14, true, {:name=>'Michael', :family=>'Jackson'}]

будет упаковано в строку длиной 57 байт, а при обратной распаковке выдаст такой же объект.

В настоящий момент PSON используется в Пандоре для:

1) упаковки записи перед созданием и проверкой подписи;

2) упаковки данных перед передачей по сети;

3) сохранения нескольких значений в одно поле таблицы базы данных.

В упакованном виде каждое значение содержит одну обязательную (*) и две необязательных компоненты:

	Имя	**Длина, байт**	**Описание**
1	Type*	1	Задаёт тип данных (4 бита), знак (1 бит), длину целого (3 бита)
2	Length	0..7	Содержит целое значение или длину данных
3	Data	0..7*10^16	Содержит данные. Для типов PT_Int, PT_Bool, PT_Time и PT_Nil эта компонента отсутствует, т.к. данные содержатся в компоненте Length

Первые 4 бита в Type задают тип данных: целое (PT_Int=0), строка (PT_Str=1), логическое (PT_Bool=2), время (PT_Time=3), массив (PT_Array=4), хэш (PT_Hash=5), символьное (PT_Sym=6), дробное (PT_Real=7), зарезервировано для других типов (8-14), пустое (PT_Nil=15). 5й бит используется как признак «отрицательный» для численных или логических значений. Оставшиеся 3 бита показывают блину поля Length в байтах. «0» означает, что поля «Length» и «Data» пропущены.

В поле Length задано целое число. Поле Length может быть в пределах (1..7) байт, 1 байт означает что длина в пределах 255, 2 – в пределах 65535, а 7 – в пределах 256^7=7*10^16. Если тип данных указан как Int или Time, то поле Length содержит значение, а поле Data опускается. Если же тип данных задан как Str, Sym, Array, Hash или Real, то поле Length указывает длину данных, а сами данные содержатся в поле Data.

Для упаковки нескольких значений желательно помещать их в массив (как в примере выше).

Также для упаковки записей создан дополнительный формат – Name-PSON. Такой формат удобен для представления записей из таблицы базы данных перед подписыванием или передачей по сети. При упаковке входной параметр задаётся как Hash, в котором название поля задано в виде строки или символа, например:

{:name=>'Michael', 'family'=>'Jackson', 'birthday'=>'29.08.1958'.to_time, :sex=>1}

Перед упаковкой имена полей будут переведены в текст, поля будут отсортированы по алфавиту, а затем упакованы в виде последовательности:

Len1:Name1:Pson1	Len2:Name2:Pson2	Len2:Name2:Pson3	Len1:Name1:Pson4

В нём каждое поле кроме типа и данных имеет имя:

	Имя	**Длина, байт**	**Описание**
1	Len	1	Длина названия (0-255)
2	Name	0..255	Название в виде строки
3	PSON	1..2*10^16	Значение поля, упакованное PSON

Для примера выше распакованное значение будет выглядеть так:

{"birthday"=>1958-08-29 00:00:00 +0500, "family"=>"Jackson", "name"=>"Michael", "sex"=>1}

Обратите внимание, все имена полей стали строковыми (хотя изначально были и строковые, и символьные), а поля отсортированы по алфавиту. Такая строгость обеспечивает формирование определенной структуры перед созданием подписи, а в дальнейшем, на других компьютерах – создание идентичной структуры и проверку имеющихся подписей, что исключает разночтение структуры и ошибки при проверке подписей.

В целом формат PSON прост, в отличие от BSON, и компактен, в отличие от JSON или XML, так как хранит данные в сыром (бинарном) виде и не требует преобразования в Base64. PSON строго структурирован, легко компонуется и разбирается. За счёт отсутствия преобразования и парсинга, PSON работает мгновенно и экономит вычислительные ресурсы и электроэнергию. Недостатком является только отсутствие человекочитаемости, что не имеет значения при машинной обработке. При подписывании, проверке подписи и передаче данных по сети PSON экономит дисковое пространство, процессорные мощности и сетевой трафик.

45. Устройство узла

При каждом подключении создается сокет. Сокеты создаются как слушателем, так и охотником. На каждое соединение запускается свой обработчик. Все установленные соединения регистрируются в списке активных узлов.

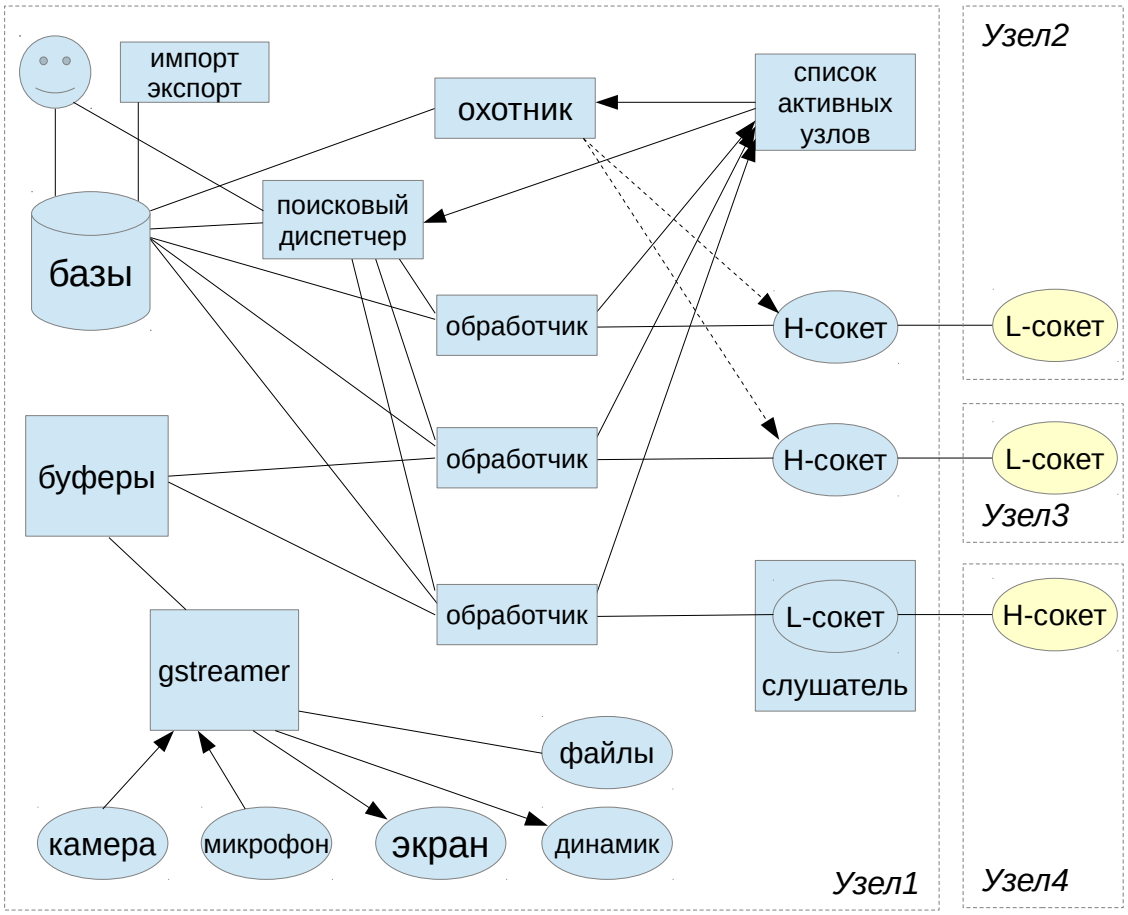


Рисунок 2: Устройство узла

46. Протокол Пандоры

Суть протокола

Обмен между двумя узлами (охотник и слушатель) выглядит примерно так:

O1: Привет! Требую протокол 1. Прошу сжатие, но не требую. [Отмычка a2c5df]

C1: Привет! Работаем на протоколе 1, без сжатия. [Работаем на отмычке]

O2:+ Дай мне статьи Михаила Михайлова?

C2:+ А ты кто?

C3:+ У тебя есть вакансии "Программист"?

O3:-C2 Я Иван Иванов.

O4:+ А ты кто?

C4:-O4 Я Петя Петров.

O2:+ Повтор1. Дай мне статьи Михаила Михайлова?

C5:-O2 Статей нет.

O5:-C3 Лови две вакансии.

O6:+ Примешь канал1 звука и канал2 видео?

C6:-O6 Давай канал1 и канал2.

O-1:= Канал1. Лови кусок звука!

C7:-O5 Ловлю вакансии.

O-2:= Канал2. Лови кусок видео!

O-3:= Канал2. Лови кусок видео!

O7:+ Вакансия 1.

C8:+ У тебя есть файлы Федора Федорова?

C9:-O7 Поймал вакансию.

O8:+ Вакансия 2.

O-4:= Канал2. Лови кусок видео!

O9:-C7 Лови список файлов.

O-5:= Канал2. Лови кусок видео!

Сеанс — это весь разговор от момента подключения до разъединения.

Протокол Пандоры является байтовым (бинарным). Обмен происходит небольшими порциями — сегментами.

Сегменты

Структура сегмента:
длина сегмента (Segsize) – 2 байта
номер сегмента (Index) – 2 байта,
код команды обмена (Comm) – 1 байт,
код расширения команды (Code) – 1 байт,
данные (Data) – 0..65000 байт.
Каждый сегмент-запрос получает номер. Сегмент-ответ кроме своего номера также содержит номер сегмента-запроса, на который она отвечает.

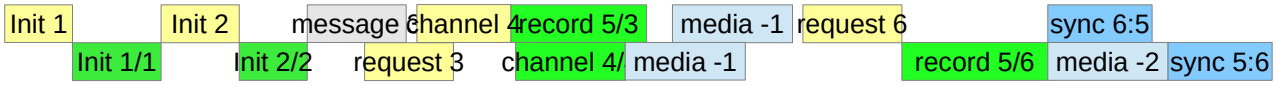


Рисунок 3: Диаграмма обмена сегментами

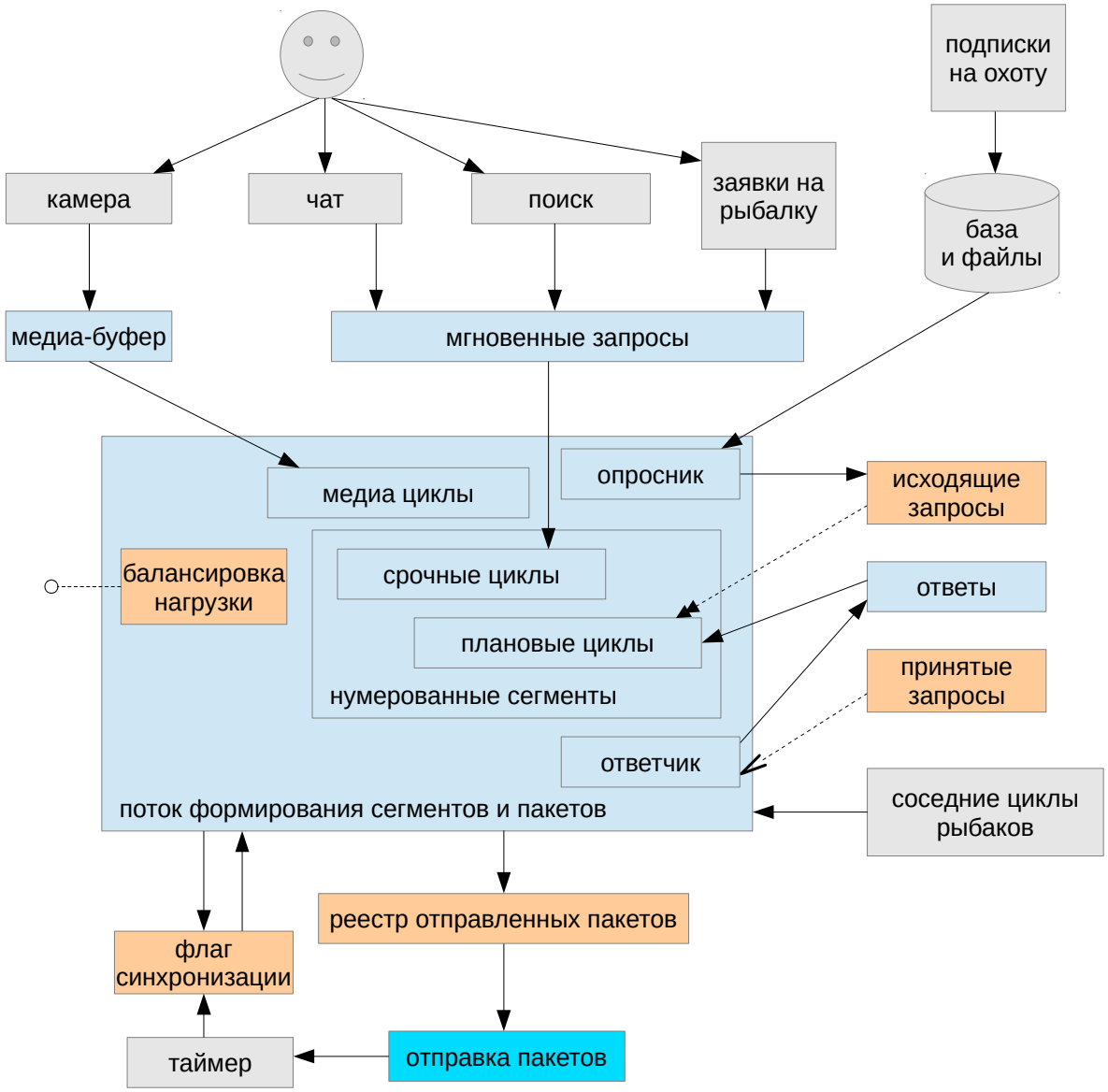


Рисунок 4: Формирование сегментов на отправку

Сегмент содержит минимум 4 байта:

Поле	Длина сегмента (2 байта)	Номер [и канал] (2 байта)	[Код команды] (1 байт)	[Параметр команды] (1 байт)	[Данные] (0..65535 байт)
Пример	23	7	2	0	Привет! Как дела?

Длина сегмента служит для определения конца сегмента в потоковых протоколах (таких как TCP), а также для дополнительного контроля в порционных протоколах (UDP).

Номер сегмента интерпретируется по-разному. Если верхний бит номера сброшен, то это **запросный сегмент**, а оставшиеся 15 бит содержат его номер (0..32767). Если верхний бит двухбайтового номера установлен, то это **мультимедиа сегмент**. В этом случае, следующие 5

бит означают номер канала (0..31), а оставшиеся 10 бит — номер мультимедиа сегмента (0..1023).

Код и параметр присутствуют только в запросных сегментах и отсутствуют в мультимедиа сегментах.

Код команды (0..255) определяет суть запросного сегмента, например «инициализация».

Параметр команды (0..255) расширяет код команды, например «приветствие».

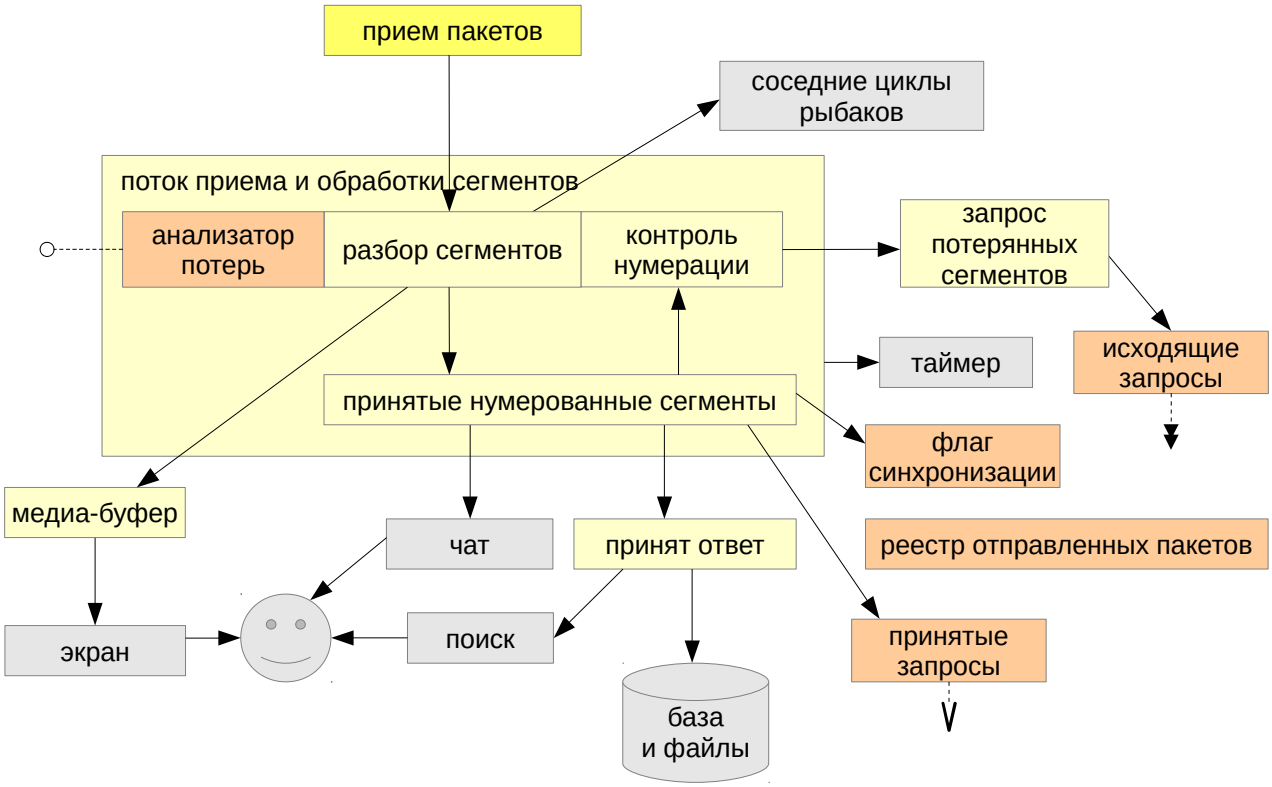


Рисунок 5: Прием и обработка сегментов

Данные (если они есть) интерпретируются по разному для разных типов сегментов. Мультимедиа сегменты переносят живой звук или видео поток. Запросные сегменты переносят все остальные виды данных.

Запросные сегменты при приёме добавляются в очередь и обрабатываются последовательно по номерам. Если в очереди потерян какой-то номер, запрашивается повтор сегмента. Если сегмент так и не был получен после нескольких запросов, сеанс разрывается с ошибкой.

Потерянные мультимедиа сегменты повторно не запрашиваются. Нумеруются же только для отслеживания процента потерь и балансировки нагрузки соединения.

Таблица. Зарезервированные коды команд для протокола 1 (альфа-версия).

Константа	**Код**	**Назначение**
EC_Repeat	0	Запрос потерянного сегмента
EC_Init	1	Инициализация сеанса

EC_Message	2	Мгновенное текстовое сообщение
EC_Channel	3	Управление медиа каналом
EC_Query	4	Запрос пачки сортов или пачки панхэшей
EC_News	5	Пачка сортов или пачка панхэшей
EC_Request	6	Запрос записи/патча/миниатюры
EC_Record	7	Выдача записи
EC_Patch	8	Выдача патча
EC_Preview	9	Выдача миниатюры
EC_Fishing	10	Управление рыбалкой
EC_Pipe	11	Данные канала двух рыбаков
EC_Sync	12	Последняя команда в серии, или индикация "живости"
EC_Wait	253	Временно недоступен
EC_More	254	Давай дальше
EC_Bye	255	Рассоединение

Узлы высылают друг другу сегменты. При этом запросные сегменты и мультимедиа перемежаются. Если канал перегружен (много потерь), то обмен запросными сегментами приостанавливается, до тех пор пока не закроются мультимедиа каналы и не освободят соединение.

Обмен запросными сегментами происходит сериями (мультимедиа сегменты не влияют на обработку серий). Серия заканчивается, когда заканчиваются данные для отправки, или когда пауза в отправке запросных сегментов затянулась. Каждая серия заканчивается сегментом синхронизации (EC_Sync), который показывает другой стороне, что его серия закончилась.

Так как протокол байтовый, то никакое кодирование не требуется.
Но зачастую для передачи структурированных данных используется формат PSON.
PSON (Pandora Simple Object Notation) — это байтовый формат, в котором закодированы: тип данных, длинна данных, сами данные.

Пакетирование

Размер сегмента как правило подбирается таким образом, чтобы он вписывался в IP-пакет. Если отправляется сразу нескольких небольших сегментов, то для уменьшения доли служебных данных и снижения нагрузки на канал Пандора пытается уместить несколько сегментов в один IP-пакет.

В протоколе не предусмотрены механизмы подтверждения о получении.
Уверенность о полноте полученных данных контролируется индексом сегментов.
Если по дороге пакет с сегментами потерялся (а такое может случиться для UDP), то получатель, отследив это по индексу полученных сегментов запрашивает пакет повторно.

Когда в обмене возникает пауза, то отправитель посылает синхронизирующий сегмент.
Таким образом, обмен идёт идёт сериями сегментов с оконечным sync-сегмент, который

говорит, что все сегменты в данной серии отправлены.

Пауза между сериями возникает по естественным причинам, например, узел ждёт, пока человек наберет сообщение, или например, идёт формирование ответа на сложный запрос. Величина паузы задана в настройках и по умолчанию равна 2 секундам. Паузы могут быть разными, поэтому узлы сообщают их друг другу при согласовании протокола.

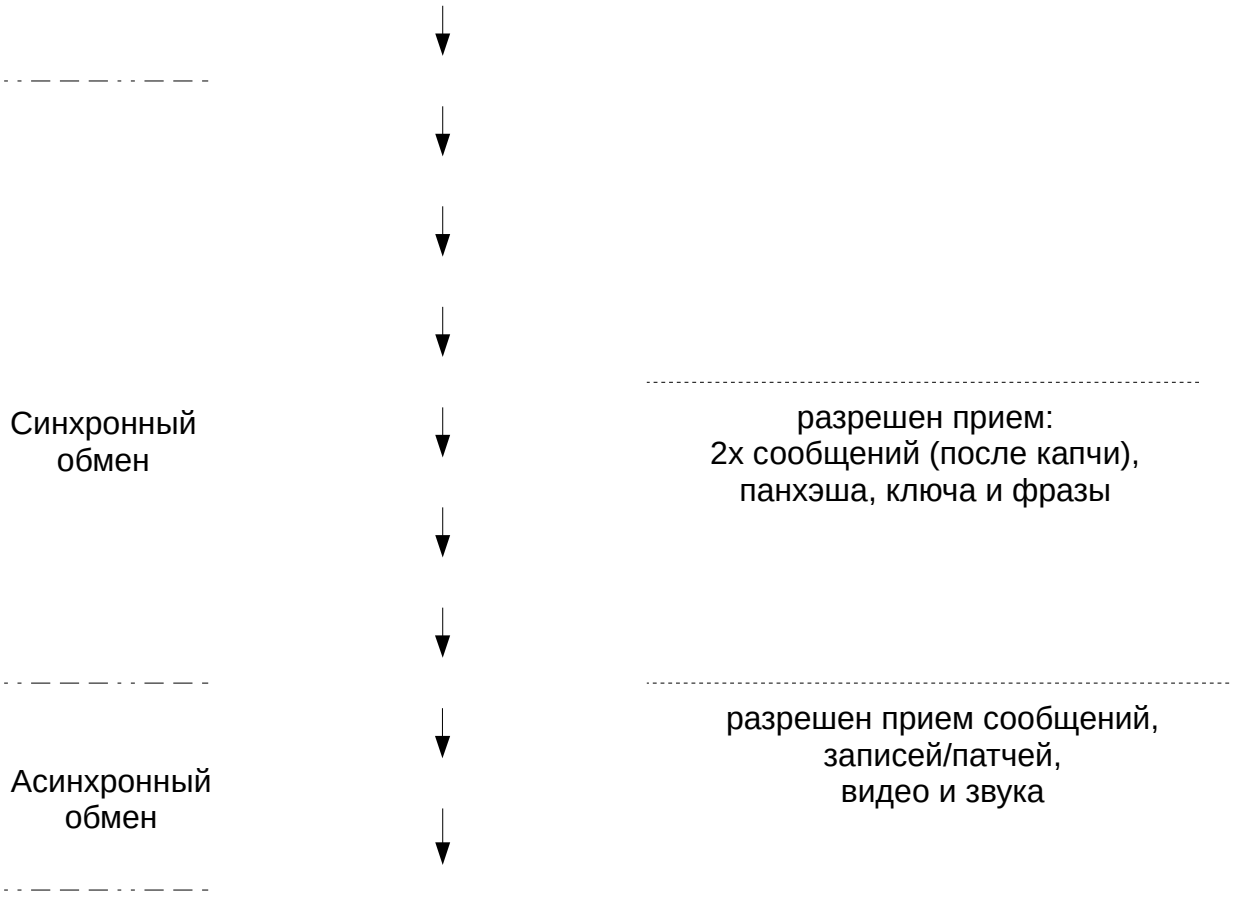
Если по истечении паузы не пришел синхронизирующий пакет, то узел сам отправляет синхронизирующий пакет, на который отправитель должен отреагировать.

Второе условие, при котором высылаются синхронизирующие пакеты — длинная пауза между сериями (по умолчанию равна 20 сек). В этом случае sync-пакет высылается, чтобы убедиться, что другая сторона еще на связи.

47. Авторизация

После соединения узлы начинают обмен сегментами проходя три этапа:
- согласование протокола
- авторизация
- обмен данными.

Любой узел в любой момент на любой стадии может прервать сеанс, если ему что-то не понравилось, или если возникла необходимо завершить работу узла.



**Желтым цветом указанны стадии, которые [могут быть пропущены]*

Рисунок 6: Этапы обмена

Каждый этап состоит из нескольких стадий.

Этап авторизации — самый сложный на сегодня этап. Авторизация выливается в прохождение от 4х до 12 стадий в зависимости от настроек безопасности узлов:
приветствие, хэш-загадка, подпись, капча, бан-лист.

На каждой стадии порция данных должна умещаться в заданный лимит.
Приветствие должно соответствовать заданному формату.
После приветсвия слушатель посылает охотнику случайную фразу и говорит: нужно решить хэш-загадку и создать электронную подпись.

Хэш-загадка обязывает охотника затратить аппаратные ресурсы, чтобы перейти к следующему шагу. Если требуется решить хэш-загадку, то последним байтом в фразе задается количество

первых бит фразы (по умолчанию 14 бит), которые считаются хэш-загадкой. Охотник должен подобрать к фразе любую добавку так, чтобы первая часть хэша SHA1 от блока «фраза+добавка» совпала с хэш-загадкой. Так как аналитического решения у этой задачи нет, а составить радужные таблицы для всех фраз длинной 256 байт не представляется возможным, то охотнику приходится искать отгадку методом перебора.

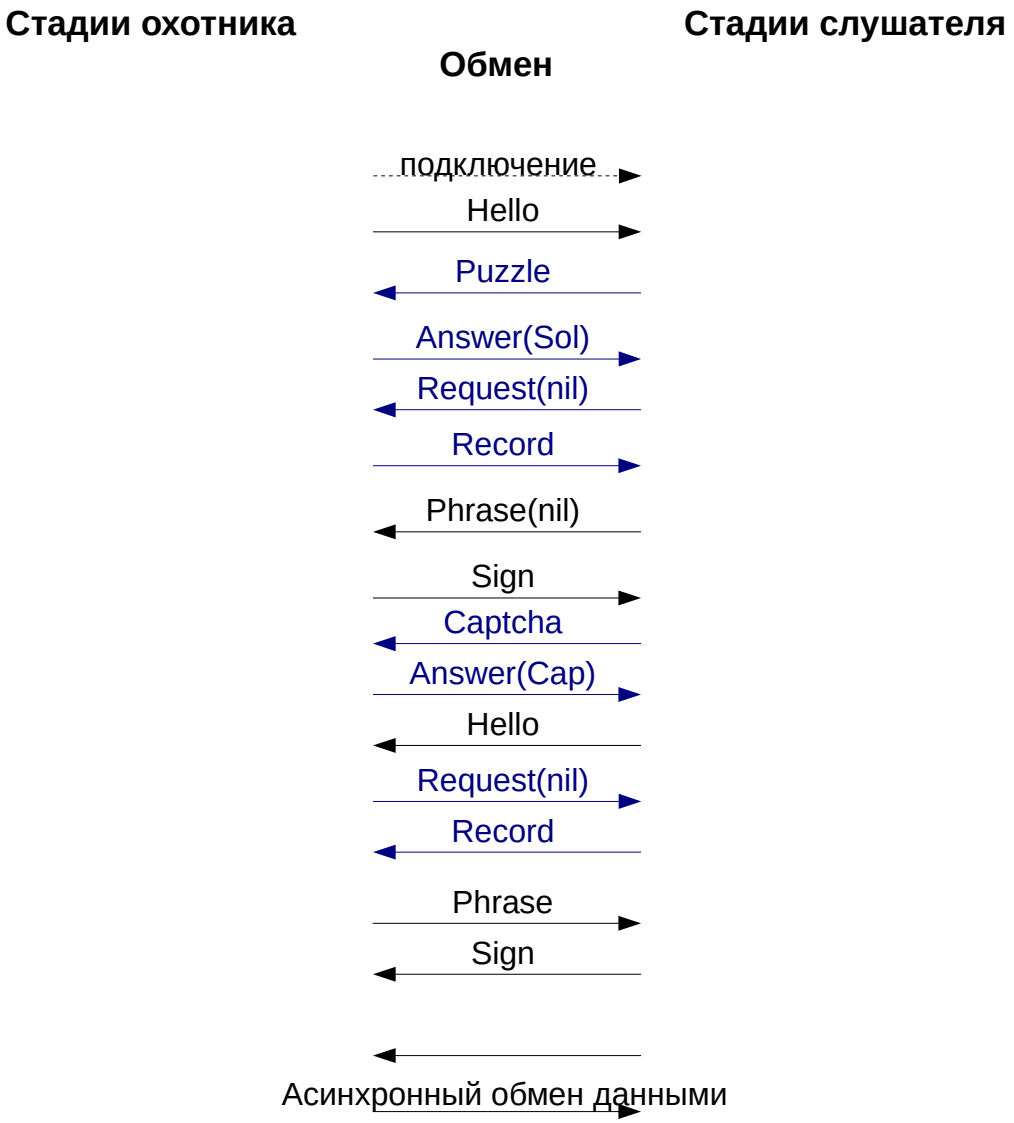


Рисунок 7: Стадии авторизации

Слушатель получает отгадку и проверяет хэш, затем запрашивает подпись.
Охотник создает подпись и высылает слушателю. Слушатель проверяет подпись.

Зловредный охотник может посылать случайные блоки, имитирующие ключ или подпись, и этим самым заставлять слушателя тратить аппаратные ресурсы на запись, инициализацию ключа и проверку подписи. Так может быть организована DDoS-атака. Чтобы уровнять положение и вводится хэш-загадка. В случае «мусорной» отгадки, слушатель тратит ресурсы только на вычисление хэша, что несопоставимо с затратами на криптографию.

Во всех случаях, когда охотник прислал «мусорные» отгадку, ключ или подпись, слушатель заносит IP атакующего в бан-лист с штрафным временем зависящим от тяжести нарушения.

Если охотник прислал корректные отгадку, ключ и подпись, но доверие к этому ключу не найдено, или найдено, но ниже заданного, то слушатель отправляет картинку с капчой. В этом случае человек, находящийся на стороне охотника, должен разгадать капчу.

После того, как охотник прошел все шаги (загадка, подпись, капча) и подтвердил себя, слушатель добавляет его в свой белый список. Затем, охотник высылает слушателю свою случайную фразу и ждёт электронную подпись в ответ. После того как охотник проверил подпись слушателя, он добавляет слушателя в свой белый список. Обоюдная авторизация считается пройденной и сеанс связи переходит в стадию обмена данными.

Если задано шифрование сеанса связи, то первым делом слушатель высылает охотнику сеансовый ключ, на котором будет шифроваться сессия.

Если IP в белом списке, то хэш-загадка и капча могут не запрашиваться. Также может использоваться упрощенная авторизация по сеансовому (симметричному) ключу. Оба приёма позволяют сильно экономить ресурсы на повторную авторизацию для восстановления недавно разорванных соединений.

При нарушениях накапливается штрафной бал от которого зависит время блокировки. Абсолютной блокировки нет, но если узел не «унимается», то суммарный период блокировки может существенно возрасти.

Блокировка выражается в «отбрасывании» соединения при подключении. В unix-подобных система в дальнейшем планируется согласование с системным фаерволом, таким как iptables.

48. Сессия

Сессия – это обработчик соединения с другим узлом, с другим ключом и другим владельцем. Соответственно, сессия характеризуется комбинацией: BaseID, Key и Person.

Сессия может быть инициализированна тремя способами:

1) подключением по адресу и авторизацией

2) поиском по Person, Key, BaseID или их комбинацией, рыбалкой и авторизацией

Процесс инициализации сессии может состоять из следующих цепочек:

«Подключение» → «Авторизация» → «Инициализация» → «Обмен»

«Запрос» → «Рыбалка» → «Ответ» → «Авторизация» → «Инициализация» → «Обмен»

«Инициализация» → «Подключение»|(«Рыбалка» → «Ответ») → «Авторизация» → «Обмен»

Инициализация может быть только по трём параметрам: BaseID, Key и Person. Первоначальная инициализация возможна только при исходящей охоте или рыбалке. При входящей охоте или рыбалке инициализация возможна только после авторизации (после шага Sign).

Исходящий запрос может быть по одному или двум параметрам из: BaseID, Key и Person,

что недостаточно для изначальной инициализации сессии в этом случае.

В этом случае исходящие соединения могут идентифицироваться по адресу, Person или Key.

При инициализации ранее не инициализированной сессии, необходима проверка – существует ли другая сессия с таким набором Person/Key/BaseID. Если существует, то:

1) при прямом подключении передать сокет на существующую сессию (там скорее всего рыбалка). Авторизация возможна через леску: туда шлётся фраза, а по соединению должна придти её подпись.

2) при рыбацком соединении смотреть, через кого идёт леска, если через другого посредника, то присоединять к существующей. Авторизация возможна через сокет-подключение или другую леску

При необходимости обмена с определенным узлом создаётся сессия. Сессия инициирует исходящий сокет на требуемый адрес. Если соединение удалось, слушающий узел создаёт сессию на своей стороне. Если сессия уже существует, то сокет прикрепляется к ней. Не подключенные напрямую к сокету сессии возможны, либо после потери связи, при этом сессия пытается сама подключиться, либо подключенные через промежуточные узлы, это называется «рыбалка».

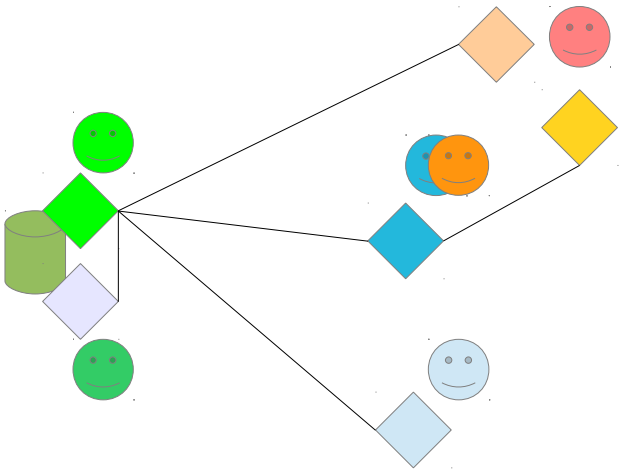


Рисунок 8: Соединение узлов, баз и ключей

Рыбаки используют любую возможность подключиться напрямую, или найти более короткую и производительную цепочку посредников.

Все сессии регистрируются в пуле. Диспетчер пула управляет соединениями:

а) регистрирует входящие соединения

б) инициирует новые исходящие сокеты

в) рассылает заявки на рыбалку, если не удалось создать сокет

г) перебрасывает соединение сессии на другие соединения

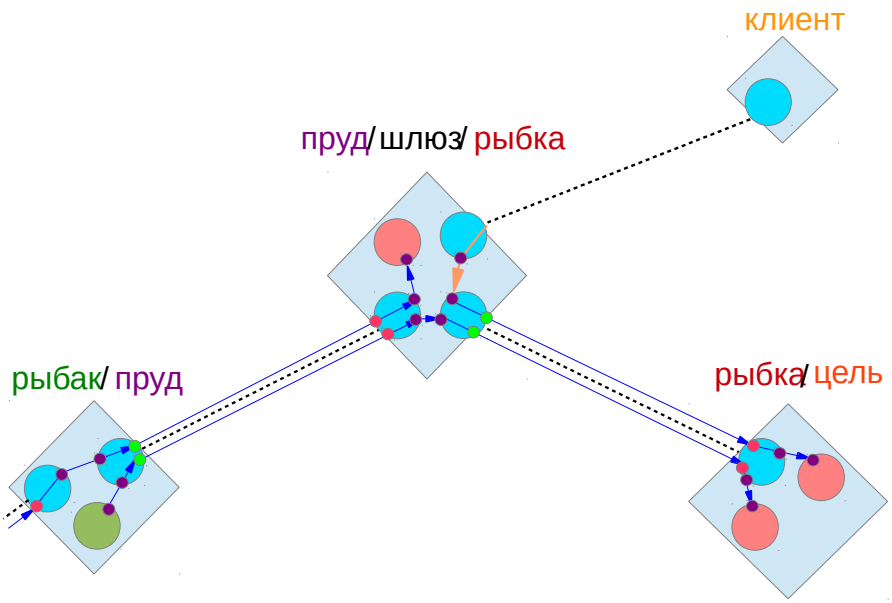


Рисунок 9: Разновидности рыбалки

Сессия инициируется в случаях:
1) открыт диалог и нажата галочка «онлайн»
2) охота по запросу (например, поиск или рыбалка)
3) плановая охота по шедулеру

Сессия продолжает переподключаться или рыбачить если:
1) активен хотя бы один диалог на этот узел
2) почемучка не закончил свою работу

Сессия держит соединение, если установлен флаг «держать» хотя бы на одной из сторон.
Флаг «держать» ставится, если есть хотя бы один диалог с галочкой «онлайн».
Флаг «держать» снимается, если диалоги закрыты или в них сняты галочки «онлайн».
Когда снимается флаг «держать», сессия оповещает об этом другую сторону.
Сессия следит за своим флагом «держать» и за таким же флагом на другой стороне.
Когда оба флага сброшены, а почемучка завершил свою работу, запускается таймер 1 мин, после чего, если необходимость в обмене заново не возникла и флаг «держать» не был заново установлен, соединение разрывается.

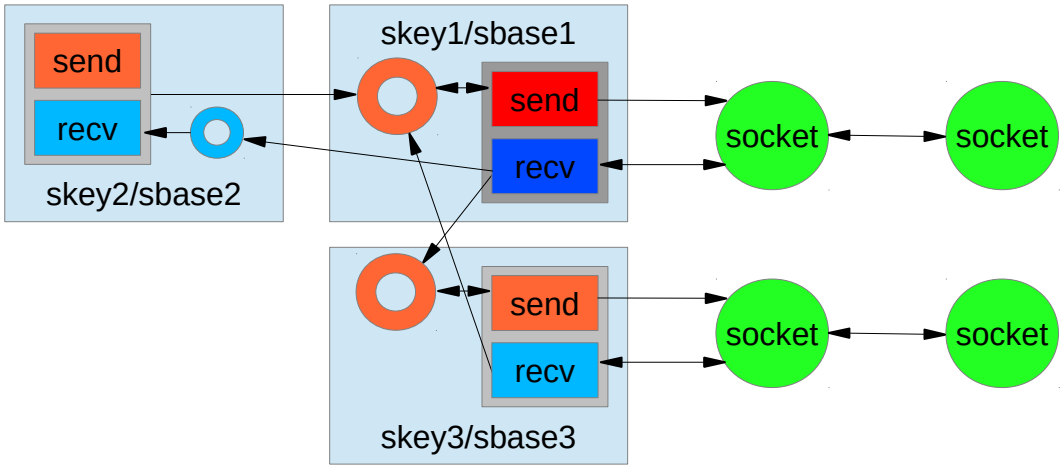


Рисунок 10: Буферы отправки и приёма

Если соединение разорвалось, а флаг «держать» установлен, сессия пытается переподключиться. Если напрямую подключиться не удалось, то сессия формирует запрос на рыбалку и передаёт диспетчеру пула. Диспетчер пула рассылает заявку по активным сессиям, и если искомый узел найден, то устанавливается рыбацкое соединие. Если в течение 20 сек активные сессии не нашли рыбку, то диспетчер начинает охоту, подключаясь к ранее не подключенным узлам.

Если рыбаки установили соединение через промежуточный узел, то они первым делом обмениваются срочными данными (сообщениями, уведомлениями), а после этого, не останавливая обмен данными, пробуют пробиться друг к другу напрямую. Для этого используется несколько методик:
1) встречный прострел на заданные порты tcp и udp
2) прострел с помощью SYN-пакета tcp
3) протокол NAT-PMP
4) перебор udp-портов в попытке «поймать» нужный
5) выход «наружу» роутера используя UPnP

Обмен данными происходит по приоритету:
1) отсылка сообщений
2) запросы почемучки
3) приём медиа потоков
4) отсылка медиа потоков

Рассмотрим сессии, сокеты, циклы обработки, циклические буферы.

49. Почемучка и ответчик

Почемучка формирует запросы и отправляет их узлу собеседника.

На другой стороне ответчик формирует ответы в два этапа. Сначала высылается список панхэшей, которые может предложить ответчик. А затем уже на конкретные запросы по панхэшам высылаются сами записи.

Почемучка рассылает панхэш требуемой записи по всем известным узлам и получает ответы тех, кто имеет эти записи или хотя бы их части. Ну и на последнем этапе, если запись больше 5 Мб, почемучка организует распределённую закачку, запрашивая разные куски у разных узлов. Размер куска согласуется вначале обмена и равен 1200 байт. При запросе куска указывается панхэш, смещение и число кусков от смещения. Схема запроса кусков задаётся в настройках. Изначально, выбрана схема «от начала», при которой почемучка в первую очередь запрашивает куски в начале записи и далее к концу.

При этом нагрузка распределяется с огладкой на пропускную способность каналов.

Если какой-то узел долго не отдаёт кусок, то кусок запрашивается у другого узла. Если приходит кусок, который уже есть, то он игнорируется.

Чтобы не перегружать базу данных запросами, на узле составляется матрица всех доступных панхэшей, которая формируется по ходу поступления запросов и пополняется, если поступили новые записи или пользователь ввёл новую запись.

Матрица панхэшей содержит:

- панхэш;
- публичность записи;
- доверие к записи;
- дату создания/модификации записи;
- связь с другими записями (для связей).

Также кэшируются последние запросы к базе данных. Размер кэша задан 100 Мб.

50. Виды коммуникаций

Полноценная работа сети обеспечивается сразу несколькими видами коммуникаций:

1) прямое соединение между узлами
2) массовые всеобщие сообщения: присутствие, чаты, заявка на рыбалку, поиск
3) массовые направленные (на узел, на ключ, на персону, на base_id): уведомление об обнаружении рыбки, запрос на создание рыбацкого канала
4) обмен между узлами по маршруту: рыбацкий канал (через хуки), анонимный канал (через cifer)

51. Массовые сообщения

Массовые сообщения – это служебные сообщения, которые узел рассылает соседним узлам, а те дальше своим соседям. Глубина рассылки задаётся узлом-инициатором и настройками на узлах.

Несколько типов массовых сообщений с разным назначением и глубиной:

1. Присутствие (MK_Presence)

Оповещение окружающих о появлении в сети и уходе из неё. Глубина 2го уровня.

2. Чат (MK_Chat)

Оповещение о входе в комнату чата и уходе из неё, а также о новом сообщении. Глубина 3го уровня.

3. Поиск (MK_Search)

Запрос у окружающих какой-либо записи (файлов в том числе) по панхэшу или полям. Глубина 3го уровня.

4. Рыбалка (MK_Fishing)

Запрос соединения с заданным человеком, ключом или узлом. Глубина 3го уровня.

5. Каскад (MK_Cascade)

Управление медиа трансляциями: открытие канала, закрытие канала, запрос трафика, отказ от трафика, поиск прерванного каскада. Глубина 3го уровня.

6. Шифроящик (MK_CiferBox)

Анонимное шифрованное сообщение. Обязательно указывается тип ключа (AES256, RSA2048 и другое). Также может быть указан нюанс (noncence) панхэша ключа получателя длиной от 1 до 6 байт, заданного типа (MD5, SHA1, CRC32 и другое). Все получатели сопоставляют нюанс шифроящика с нюансами своих ключей и, при совпадении, пытаются расшифровать содержимое. Тот, кто расшифровал – и есть получатель сообщения. Шифроящики хранятся в кэше 2 дня и рассылаются с периодом 3 часа. Особенностью является, что в поле MR_Node задаётся MD5 хэш шифроблока, а не узел-источник. Шифроящики скрывают отправителя и особенно получателя в общей массе. Глубина 4го уровня.

В будущем возможно добавление других типов массовых сообщений, например, уведомления об активности пользователя (опубликовал запись, сменил режим и так далее).

Массовые сообщения рассылаются одним механизмом и подчиняются одним правилам рассылки.

Каждый узел ведёт список массовых сообщений (mass_records) созданных собою и соседями. Сообщения рассылаются по соседям, а спустя заданное время жизни (по умолчанию, 5 минут), удаляются мусорщиками на узлах и, соответственно, перестают рассылаться. Расползаясь по узлам, массовое сообщение вызывает в них нужную реакцию.

Массовые сообщения **на 1-м уровне** (т. е. созданные самим узлом) рассылаются сразу в полном виде (размер от 97 до 275 байт и более), за исключением MK_CiferBox:

№	**Поле**	**Тип (длина)**	**Описание**
Общие поля для всех типов массовых сообщений (75 байт)			
1	MR_Kind	Byte (1)	Тип: присутствие (P), чат (C), поиск (S), рыбалка (F)
2	MR_Index	Integer (4)	Индекс в реестре отправителя
3	MR_Person*	Panhash (22)	Панхэш персоны отправителя
4	MR_Key*	Panhash (22)	Панхэш ключа отправителя
5	MR_BaseId*	String (16)	ID базы отправителя

6	MR_Time**	Word (4)	Время смерти сообщения (секунд от начала часа)
7	MR_Trust	Byte (1)	Минимальное доверие для рассылки
8	MR_Depth	Byte (1)	Максимальная глубина рассылки
0	MR_Session	Integer (4)	Указатель на сессию, с которой пришло (заполняется только в реестре и не рассылается другим)
Специальные поля для типов P, C, S, F +(22-200)			
9	MRP_Nick	String (30)	Ник на радаре и в чатах
9	MRC_Comm	Byte (1)	Команда чата (открыл, закрыл, сообщение)
10	MRC_Body	Panhash (22)	Панхэш комнаты (и тело сообщения)
9	MRS_Kind	Byte (1)	Тип запроса
10	MRS_Request	String (60)	Тело запроса
11	MRA_Answer	String (22)	Ответ на запрос (заполняется при ответе)
9	MRF_Fish	Panhash (22)	Панхэш человека для рыбалки
10	MRF_Fish_key	Panhash (22)	Панхэш ключа для рыбаки
11	MRL_Fish_Baseid	String (16)	ID базы для рыбалки

* Для 1-го уровня поля могут не указываться, а для 2-го и глубже заменяться одним полем MR_Node, длинною в 22 байта.

** Для синхронного учёта времени жизни узлы могут запрашивать друг у друга абсолютное время и вычислять коррекцию.

Массовые сообщения типа MK_CiferBox, а также **на 2-м уровне и глубже** (т. е. сообщения соседей) рассылаются через уведомления в три шага: 1) сначала высылается уведомление о массовом сообщении (tweet, твит), 2) затем, если у узла, получившего твит, нет сообщения с присланным идентификатором, он высылает запрос (ECC_Mass_Req), 3) по запросу высылается уже само массовое сообщение в полном виде.

Использование уведомлений позволяет предотвратить многократную рассылку одних и тех же массовых сообщений, что было бы неизбежно в децентрализованной сети, так как между двумя узлами может существовать несколько маршрутов.

Уведомление и запрос содержат только идентификатор сообщения (26 байт):

№	**Поле**	**Тип (длина)**	**Описание**
1	MR_Node	Panhash (22)	Идентификатор инициирующего узла
2	MR_Index	Integer (4)	Индекс в реестре массовых сообщений инициатора

При поступлении уведомлений, уведомления регистрируются, а узлы-хранители добавляются в список источников (MR_KeepNodes). Шедулер пула перебирает список, находит непринятое массовое сообщение (было одно только уведомление), отправляет на первый узел из списка KeepNodes запрос, добавляет индекс узла в поле MR_ReqIndexes и записывает время запроса в MR_ReceiveState. Если узел прислал массовое сообщение, то все поля заполняются, а в узел MR_ReceiveState записывается true. Если же ответ не пришёл 5 секунд, шедулер берёт второй узел хранитель, высылает ему запрос, добавляет индекс в поле MR_ReqIndexes и обновляет время MR_ReqTime. Если очередной опрашиваемый узел по порядку из KeepNodes ушёл оффлайн, то он пропускается, а при достижении конца списка снова опрашивается. Если

список хранителей кончился, а ответы так и не поступили, то уведомление помечается как мёртвое (MR_ReqTime=false) и по таймауту будет удалено мусорщиком.

Нечаянные или злонамеренные коллизии, когда разные узлы выдают себя за узел с одним MR_Node (MR_Person, MR_Key, MR_BaseId), разрешаются с помощью системы доверия – у какого человека и ключа выше рейтинг, тот и считается источником массового сообщения.

В этом случае ответы на массовые сообщения могут приходить не тем, кто их рассылал. При получении такого ответа, узел должен оповестить отвечающего о фейке. В этом случае отвечающий узел запускает процедуру обнаружения фейка и его наказание.

В начале сеанса связи узлы сообщают друг другу параметры массовой рассылки:

Параметр	**Диапазон**	**Умолчание**	**Значение**
Mass_Kinds	0-15	15	Каждый бит задаёт тип массового сообщения (P, C, S, F), которое будет принимать узел, остальные игнорируются (по умолчанию все 4 бита включены)
Mass_Trust	-1.0..+1.0	0	Определяет минимальное доверие к отправителю массового сообщения, остальные отбрасываются
Mass_Depth	0-255	3	Глубина, с которой принимаются массовые сообщения
Mass_Full	0-255	0	Глубина, с которой принимаются сразу полные сообщения без уведомлений

Все эти параметры могут быть изменены владельцем узла.

При подключении узлы сохраняют твит последнего массового сообщения. После разрыва и повторного подключения, узлы сообщают последние твиты друг друга и по ним делают досылку новых твитов друг другу в заданных лимитах объема (16kb) и количества (500). Если последний твит не сохранился, или объём или количество получились больше лимита, то высылаются только последние твиты с конца списка.

52. Рыбалка

Рыбалка – это механизм запроса одним узлом у другого соединения с третьим узлом в случае, когда узел-инициатор не может соединиться с искомым напрямую, например, в случае нахождения третьего узла за NAT.

В зависимости от того, находятся третий узел на связи, или нет, является сам инициатор публичным слушателем или нет, рыбалка выглядит следующим образом:

- A запрашивает у B рыбалку узла C

- если C подключен к B, узел B сообщает узлу C что его ищет A

- если C отключен, но слушает, к нему подключается узел B

- если C отключен и не слушает, то B запрашивает A: будет ждать (10 мин макс) или периодически подключаться (каждую 1 мин)?

- если A ждет - он висячий рыбак

- если A периодически опрашивает - он прыгающий рыбак

- если C подключился к B, то B сообщает C, что на него идет рыбалка от A, далее одно из трёх:

- C подключается к узлу A напрямую

- начинается разговор между A и C через B, если еще A висит на B

- узел B даёт узлу C время на встречную висячую рыбалку узла A (10 минут), если A прыгающий

- если прыгающий A появился за это время, их соединяют, если нет - оба рыбака снимаются с рыбалки

- рыбаки снимаются с рыбалки также по таймауту (10 мин)

Дополнительно:

- на узле (B) задается максимальное число заявок на рыбалку, например 500

- кроме того задается максимальное число подключенных рыбаков, например 20

- в заявке на рыбалку может фигурировать не только узел, но и панхеш человека или ключа

- узел B может задавать ограничения на объем трафика в ед. времени или запретить медиа-обмен.

№	Hook	NeighborSession	NeighborHook	SocketHook	FromKeyBase/ToKeyBase
1	1	0x0000005	2	4	
2	3	0x0000007	8	5	

Стадии охоты и рыбалки при запросе соединения с искомым узлом:
1) просмотр активных сессий по: узлу, ключу, человеку, стоп, если есть сессия
2) попытка подключения к узлу – 5 сек, стоп – есть сокет
3) регистрация заявки на рыбалку (bob, поплавок) в пуле: искомый узел (ключ, человек, baseid)
4) рассылка активным сессиям заявки – 10 сек, стоп – пришёл крючок, леска натянулась
5) короткая охота по связям «пришёл с» – 7 сек, стоп – есть сокет
6) раунд полной охоты (с рассылкой поплавков) – до прохода всех, но не более 5 мин, стоп – есть крючок и леска
7) как только есть леска – начать попытки перфорации NAT: встречный прострел, прострел по предпоследним, и далее схождение в середину.

Конкретно, рыбалка начинается с генерации поплавка (bob) на узле, который не смог установить соединение напрямую. Поплавок рассылается по окружным узлам через механизм массовых сообщений.

53. Перфорация NAT

NATы преодолеваются с использованием технологий : UPnP (IGD?) - for local, NAT-PMP, STUN, SOCKS, NAT-T (in IKE), TURN, RSIP, MIDCOM, ICE, Teredo(miredo), SBC, SYN-TCP,

UDP-pounching, ALG.

54. Мультимедиа и GStreamer

конвейеры, кодеки, захват звука и видео, отображение и звучание.

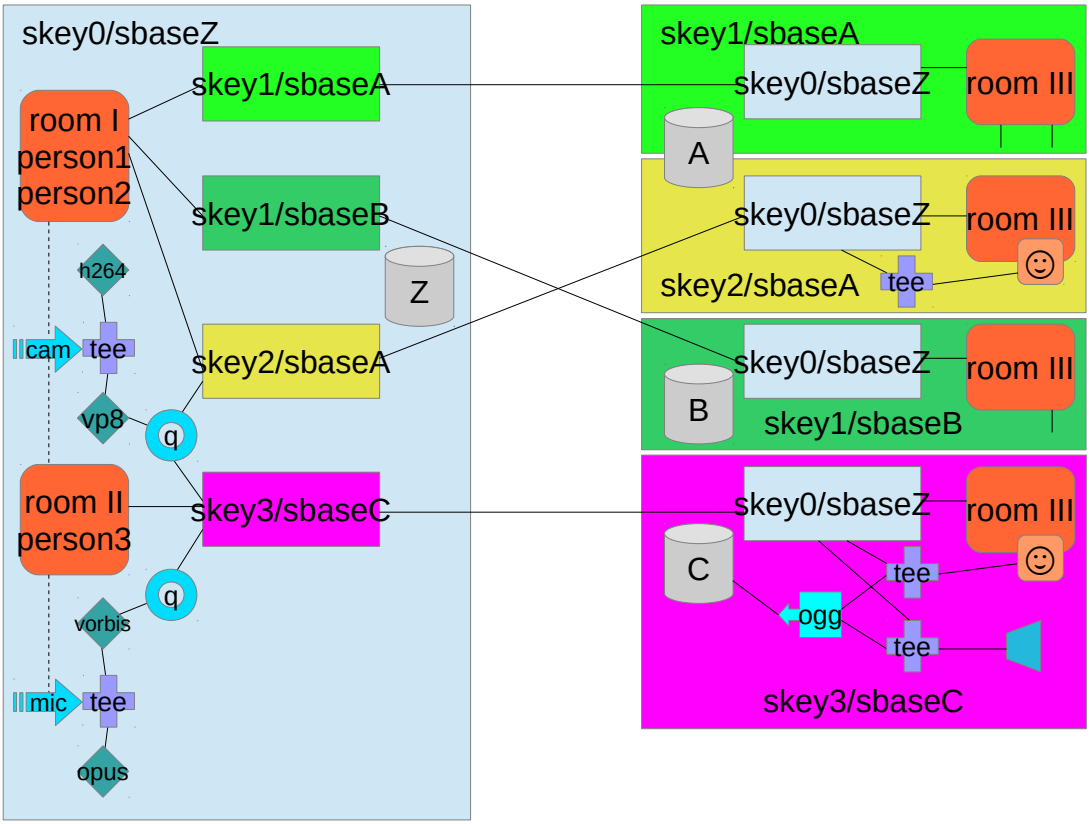


Рисунок 11: Обмен между сеансами, комнатам, буферами и базами

GStreamer позволяет:

- захватывать видео и аудио потоки с камеры и микрофона,
- выводить их в окно или на звуковое устройство,
- запаковывать медиа поток нужными кодеками в нужный контейнер,
- сохранять медиа поток в файл.

Хотя возможностей у GStreamer больше, для Пандоры перечисленных достаточно.

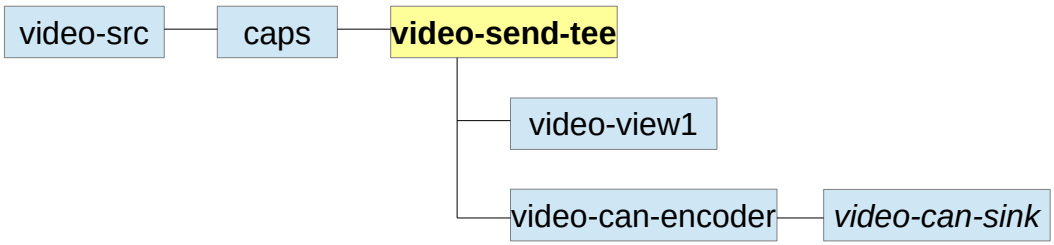


Рисунок 12: Видео-конвейеры

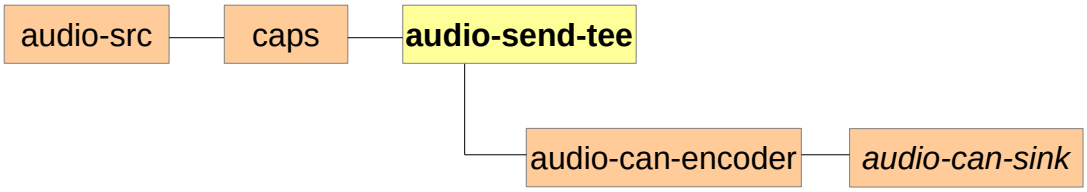


Рисунок 13: Аудио-конвейеры

55. Передача файлов/блобов

Если блоб меньше 20Кб, то он передаётся напрямую от одного узла.

Если размер больше, то организуется закачка с нескольких узлов наподобе торрента.

56. Доверие и рейтинг

Алгоритмы расчёта, настройка доступа.
Расчёт рейтинга записи возможен после авторизации. При каждом запросе, рейтинг высчитывается и сохраняется в таблице в памяти. При поступлении очередной подписи: а) человека – рейтинги пересчитываются по веткам доверия, б) другой записи – рейтинг только самой записи.

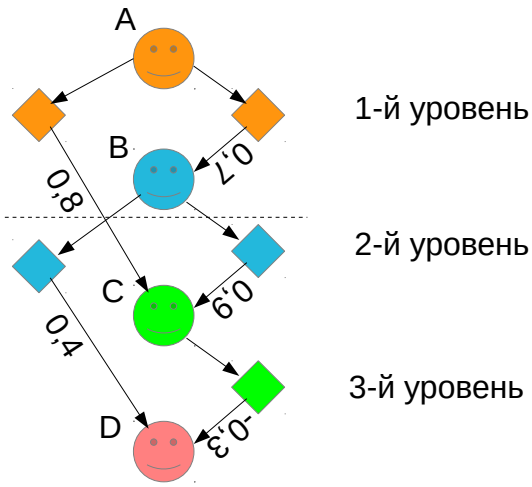


Рисунок 14: Ветки доверия

57. Графический интерфейс Gtk2

Форма ввода записи FieldsDialog заполняется полями разного вида, в зависимости от типа полей. Размер формы и разбиение полей на строки происходит автоматически. Для этого создаётся матрица field_matrix, которая при событии on_resize_window перестраивается таким образом, чтобы на экране уместилось максимальное число полей.

баги и хаки.
к чему надо стремиться.

58. Формы отчётов

Формы отчетов составляются в формате *.odt или *.ods.

59. Криптогрфия через OpenSSL

Генерация ключей

Ключ — это случайная последовательность байт определенной длины.

В зависимости от того, в каком механизме будет использоваться ключ, бывают одиночные ключи (при симметричном шифровании) и пара ключей (при асимметричном шифровании).

Симметричное шифрование — это преобразование данных с помощью заданного алгоритма и заданного ключа. Для расшифровки используется тот же ключ, что и для шифрования. Допустим, общаются два узла. Чтобы обменяться данными один узел генерирует ключ и передаёт его партнеру. Теперь оба узла используют одинаковый ключ для шифрования и расшифровки. Один и тот же ключ для симметричного шифрования может использоваться в разных алгоритмах шифрования. Безопасность обеспечивается передачей ключа незаметно ни для кого другого. Если кто-то перехватил ключ, то он сможет как расшифровывать данные и смотреть их, так и зашифровывать данные, изменяя их.

Асимметричное шифрование — это преобразование данных с помощью заданного алгоритма и двух пар ключей (т. е. четырёх ключей вместо одного). Каждый участник генерирует пару: открытый и закрытый ключ. Второй узел генерирует свою пару. Затем узлы обмениваются открытыми ключами. При этом каждый узел оставляет у себя свой закрытый ключ.

Пара ключей связанны между собой математической зависимостью, заложенной в алгоритм шифрования. Поэтому пара ключей сгенерированная для определенного алгоритма шифрования не может использоваться в другом асимметричном алгоритме. Вторая особенность ассимитричного алгоритма заключается в двух видах шифрования.

В первом случае шифрование происходит на открытом ключе получателя, и расшифровать сообщение может только владелец закрытого ключа. Это шифрование широко используется, в том числе для скрытой передачи симметричных ключей.

Во втором случае шифрование происходит на закрытом ключе отправителя, а расшифровать сообщение может любой обладатель открытого ключа. Такое шифрование используется для создания электронной подписи (ЭЦП), и последующей её проверки любым обладателем открытого ключа.

Безопасность асимметричного шифрования выше симметричного за счет того, что если кто-то перехватывает ключи, то он не может расшифровать данные или подделать подпись, так как у него нет закрытых половинок. Тем не менее, существует способ атаки на асимметричный обмен, называемый «человек посередине»: при первом обмене ключами, злоумышленник перехватывает открытые ключи и подменяет их своими поддельными. Далее обмен идет на фиктивных ключах злоумышленника.

Для защиты от «человека посередине» узлам необходимо убедиться, что они получили именно те открытые ключи, которые посылали друг другу. Для этого существует три способа: 1) дублирование отправки ключа по другому каналу; 2) сверка хэшей ключей, например, по телефону; 3) проверка цифровых подписей открытых ключей, сделанных третьими узлами, которым доверяет получатель.

Важно заметить, что при одинаковых уровнях криптостойкости, скорость работы асимметричных алгоритмов в среднем в 1000 раз медленее, чем у симметричных, и требует ключи в 10 раз длинее, чем у симметричных. Поэтому в Пандоре, как и в других системах, используется комбинированное шифрование: в начале соединения асимметричное шифрование используется для авторизации и передачи симметричного ключа (так называемого «сеансового»), а далее обмен проходит на этом симметричном ключе.

Для генерации служит метод:

generate_key(type_klen = KT_Rsa | KL_bit2048, cipher_hash=nil, cipherkey=nil)

Результат: один или пара ключей, в зависимости от заданного алгоритма.

Хранение ключей

Пандора хранит пары ключей разрознено: открытые отдельно от закрытых. Каждая пара ключей (или один симметричный ключ) в Пандоре имеет свой панхэш.

Открытые ключи хранятся в базе в открытом виде.

Закрытые ключи (и для симметричного и для асимметричного шифрования) хранятся в зашифрованном виде. При этом шифрование ключа происходит на симметричном алгоритме, ключом которому служит некоторый пароль, обработанный функциями хеширования и «засолки». Симметричные ключи, как правило, хранятся зашифрованными на ассиметричных ключах.

Активация ключей

Активация ключа заключается в получении пароля от пользователя, соленого хеширования этого пароля, расшифровки ключа, вычисления параметров ключа (например, для RSA требуется вычислить 7 параметров) и создания криптографического объекта. Используется метод:

init_key(key_vec)

Результат: криптографический объект или Integer при ошибке с кодом ошибки.

Шифрование/дешифрование

Генерация подписей

Проверка подписей

4. Асимметричное (rsa) и симметричное (aes, bf) шифрование, эл. подпись, хэши

60. Мусорщик

Сборщик мусора работает в фоне шагами по таймеру с заданным временным интервалом.

Функции мусорщика:

1) сокрытие, а затем удаление, старых записей без поддержки и доверия
2) рассылка синхропакетов в активных сессиях, отключение замерших соединений
3) очистка списков в памяти от устаревших записей
4) активация заданий, у которых подошло время

61. Катушечная координата

Под географическую координату в Пандоре отводится 4 байта.

Широта имеет диапазон 180 градусов, а долгота – 360. Если отводить по 2 байта на каждую координату, то погрешности распределяются ассиметрично: 0,0014 (150 м) и 0,0027 (300 м) градусов, соответственно.

Чтобы выровнять погрешность, в Пандоре используется «катушечная координата», при которой 4-байтовое число показывает порядное смещение точки в матрице от левого верхнего угла (Северного полюса), равного 0, до правого нижнего (Южного полюса), равного чуть меньше 255^4 (размер матрицы ограничен 4 байтами).

Матрице в градусах 360*180=64800 соответствует матрица значений 92681*46340=4294837540.

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	10	11	12	13	14	15	16	17	18
3	19	20	21	22	23				
4									
5									
6									

Рисунок 15: Катушечная координата

Если широта равна W, а долгота L, то

катушечная координата вычисляется по формуле:

K = 92681*(ROUND(W/180*46340)-1)+ROUND(L/360*92681)

Обратное преобразование происходит по формулам:

L = (K-TRUNK(K/92681))*92681)/92681*360

W = (TRUNK(K/92681) + 1)/46340*180

при этом погрешность для каждой координаты становится одинаковой и составляет 0,0019 градуса (211 м).

Хотя обеспечивается достаточно большая точность, Пандора округляет координату до сотой доли градуса 0,01 (погрешность 556 м), а при вычислении признака координаты в панхэше координата округляется до десятой доли градуса 0,1 (погрешность 5,5 км), это позволяет нивелировать разброс незначительно различающихся координат для одного и того же насёленного пункта, если пользователи берут координаты из разных справочников.

При вычислении признака координаты в панхэше населённые пункты, удалённые на 5,5 км и ближе, «сливаются». Это с одной стороны компенсирует разброс при вводе координат, но с другой даёт достаточную географическую идентификацию объекта в его панхэше.

62. Параметры

Параметр служит для долгосрочного хранения некоего значения в базе. Значение параметра пользователь может в любое время изменять. Параметр создаётся в таблице «parameters» при первом обращении к нему функцией get_param(). Параметры описываются в файле /model/01-base.xls в разделе «Defaults». Описание выглядит примерно так:

```
<base_id desc="Base ID" type="String" section="common" setting="[create_base_id],hex" />
```

1) имя параметра (в примере выше «base_id»)

2) desc – описание

3) type – тип (String, Integer, Word, Byte, Boolean, Time)

4) section – секция (common, view, update, network, crypto, media), может задаваться числом

5) setting – может содержать два значения, разделенных запятой: значение по умолчанию (или функция инициализации в квадратных скобках), вид отображения (hex, base64 и т.д.).

Секция «Главное» (Common)

Главные параметры определяют общую работу Пандоры.

base_id – Идентификатор базы [вдобавок к ключу] служит для идентификации узла. Генерируется функцией create_base_id при первом запросе его значения, содержит 16 байт. Первые 4 байта – время генерации, остальные 12 – случайные байты. Таким образом идентификатор уникален для каждого узла. Если вы переустановили Пандору после сбоя, то можете восстановить старый идентификатор вручную. В остальных случаях параметр генерируется автоматически.

Секция «Вид» (View)

Параметры определяют визуальное поведение Пандоры.

do_on_show – Делать при показе главного окна (Do on show main window). 0 – ничего, 1 – инициализировать ключ, 2 – запустить слушателя, 4 – запустить охотника, 7 – всё сразу.

hide_on_minimize – Скрывать главное окно при минимизации, иначе сворачивать.

status_update_win_icon – Менять иконку главного окна при поступлении сообщения.

status_flash_on_new – Моргать иконкой в трее при поступлении нового сообщения.

status_flash_interval – Интервал моргания иконкой в секундах.

play_sounds – Проигрывать звуки при событиях (например, получении сообщения).

linux_mp3_player – Задаёт утилиту для GNU/Linux, с помощью которой будут проигрываться звуки.

win_mp3_player – Задаёт утилиту для Windows, с помощью которой будут проигрываться звуки.

load_history_count – Число сообщений из истории подгружаемых в окно диалога при его открытии.

sort_history_mode – Сортировать сообщения по: 0 – времени получения получателем, 1 – времени создания отправителем.

Секция «Обновление» (Update)

Параметры регулируют обновление Пандоры.

check_update – Проверять обновления.

check_interval – Интервал проверки в днях, если истёк update_period.

update_period – Интервал в днях, меньше которого обновления не проверяются.

last_check – Время последней успешной проверки.

last_update – Время последнего успешного обновления.

Секция «Сеть» (Net)

Параметры определяют сетевое поведение Пандоры.

tcp_port, udp_port – Порты слушателя. По умолчанию 5577.

listen_host – Хост слушателя. Обычно это локальный адрес.

callback_addr – Домен и порт, который сообщается удаленному узлу для обратной связи. Может использоваться в случаях, когда входящий и исходящий адреса различаются. Или когда используется шлюз для сбора внешних соединений.

puzzle_bit_length – Длина хэш-загадки в битах (0 – отключено).

puzzle_sec_delay – Требовать ожидание на шаге хэш-загадки (в секундах).

captcha_length – Длина капчи в символах (0 – отключено).

captcha_attempts – Число допустимых попыток ввода капчи.

trust_for_captchaed – Оказывать малое (равное 0.01) доверие узлам, прошедшим капчу.

trust_for_listener – Доверять (с доверием 0.01) слушателю (т. е. к кому подключаешься)

low_conn_trust – Минимально допустимый уровень доверия для подключаемых. По умолчанию 0.

Секция «Крипто» (Crypto)

Параметры задают работу с ключами.

last_auth_key – Последний авторизованный ключ.

Секция «Медиа» (Media)

Параметры задают работу с мультимедия через библиотеку GStreamer.

video_src_v4l2 – источник изображения в GNU/Linux, как привило устройство для веб-камеры.

video_src_win – источник изображения в Windows.

video_src_auto – Источник звука для автоматического режима.

video_src – Ссылается на параметр с источником звука (по умолчанию на video_src_auto).

video_send_caps_raw_320x240 – Задаёт фильтр для видео потока raw 320x240.

video_send_caps_raw_640x480

video_send_caps_yuv_320x240

video_send_caps_yuv_640x480

video_send_caps – Ссылается на параметр с нужным фильтром (по умолчанию на raw 320x240)

video_send_tee_def – Разветвитель для отправки видео.

video_view1_auto – Элемент авто захвата видео потока .

video_view1_x – Элемент для нерастягивающегося экрана.

video_view1_xv – Элемент для растягивающегося экрана.

video_view1_win – Элемент для вывода в Windows.

video_view1 – Ссылка на параметр для экрана с видео на отправку.

video_can_encoder_vp8 – Цепочка для кодирования vp8.
video_can_encoder_jpeg – ** jpeg
video_can_encoder_smoke – **smoke
video_can_encoder_theora – ** theora
video_can_encoder_x264 – ** x264
video_can_encoder_h264 – ** h264
video_can_encoder – Ссылка на параметр кодировщика (по умолчанию vp8).
video_can_sink_app – Цепочка для извлечения сырого видео на отправку.
video_can_src_app – Цепочка для заполнения сырого видео при получении.
video_can_decoder_vp8 – Цепочка декодирования vp8
video_can_decoder_jpeg – **
video_can_decoder_smoke – **
video_can_decoder_theora – **
video_can_decoder_x264 – **
video_can_decoder_h263 – **
video_can_decoder – Ссылка на параметр декодера (по умолчанию vp8)
video_recv_tee_def – Разветвитель на получении видео.
video_view2_auto – Авто
video_view2_x
video_view2_xv
video_view2_win
video_view2
audio_src_alsa – Цепочка захвата звука с ALSA
audio_src_pulse
audio_src_auto
audio_src_win
audio_src_test
audio_src
audio_send_caps_8000 –
audio_send_caps –
audio_send_tee_def –
audio_can_encoder_vorbis –
audio_can_encoder_speex –
audio_can_encoder_opus –
audio_can_encoder_a52 –
audio_can_encoder_flac –
audio_can_encoder_mulaw –
audio_can_encoder_mp3 –

audio_can_encoder_voaac –
audio_can_encoder_faac –
audio_can_encoder_voamrwb –
audio_can_encoder_adpcm –
audio_can_encoder_amrnb –
audio_can_encoder_nelly –
audio_can_encoder –
audio_can_sink_app –
audio_can_src_app –
audio_can_decoder_vorbis –
audio_can_decoder_speex –
audio_can_decoder_mulaw –
audio_can_decoder_mp3 –
audio_can_decoder_amrwb –
audio_can_decoder_adpcm –
audio_can_decoder_amrnb –
audio_can_decoder_voaac –
audio_can_decoder_a52 –
audio_can_decoder_faad –
audio_can_decoder_nelly –
audio_can_decoder_flac –
audio_can_decoder_opus –
audio_can_decoder –
audio_recv_tee_def –
audio_phones_auto –
audio_phones_alsa –
audio_phones_pulse –
audio_phones_win –
audio_phones –

63. Функции и их описание

I. Модуль PandoraUtils – служебные функции

1. Общие функции

os_family – возвращает тип операционной системы: unix, windows или other
level_to_str(level) – преобразует уровень логирования в строку: Error, Warning, Info, Trace
log_message(level, mes) – добавляет сообщение в список событий внизу главного окна [и в лог-

файл]

2. Языковые функции

load_language(lang='ru') – загружает переведенные фразы из языковового файла в память

unslash_quotes(str) – разэкранирует кавычки

addline(str, line) – добавляет строку к другим

spaces_after?(line, pos) – есть ли пробелы после позиции?

save_as_language(lang='ru') – сохраняет переводы из памяти в файл

slash_quotes(str) – экранирует кавычки слэшем

there_are_end_space?(str) – есть конечный пробел или табуляция?

get_name_or_names(mes, plural=false) – из перевода берет слово в нужном числе

3. Операции с панхэшем

panhash_nil?(panhash) – панхэш не нулевой?

kind_from_panhash(panhash) – берет код типа из панхэша

lang_from_panhash(panhash) – берет код языка из панхэша

4. Преобразования типов

bytes_to_hex(bytes) – преобразует блок данных в 16-ричную строку

hex_to_bytes(hexstr) – преобразует 16-ричную строку в блок

bytes_to_bigint(bytes) – преобразует строку байт в большое целое

bytes_to_int(bytes) – преобрзует строку байт в целое

date_to_str(date) – дату в строку

str_to_date(str) – стоку в дату

fill_zeros_from_left(data, size) – заполняет строку нулями слева до нужного размера

fill_by_zeros(str) – перебивает строку нулями

string_to_pantype(type) – преобразует тип в код

pantype_to_view(type) – код типа в метод отображение

any_value_to_boolean(val) – любое значение в логическое

calc_midnight(time) – округлить время к полуночи

time_to_str(val, time_now=nil) – время в человеческий вид

time_to_dialog_str(time, time_now) – преобразует время в строку для диалога

val_to_view(val, type, view, can_edit=true) – значение для отображения

view_to_val(val, type, view) – преобразовать значение в согласованный вид

text_coord_to_float(text) – координата как текст в число

coord_to_int(y, x) – координату в 4-байтовое целое

int_to_coord(int) – целое в координату

simplify_coord(val) – упростить координату

encode_pson_type(basetype, int) – кодирует тип данных и размер в тип PSON и число байт размера

decode_pson_type(type) – раскодирует тип PSON в тип данных и число байт размера

rubyobj_to_pson_elem(rubyobj) – конвертирует объект руби в PSON ("простая нотация объектов в Пандоре")

pson_elem_to_rubyobj(data) – конвертирует PSON в объект руби

value_is_empty?(val) – значение пустое?

namehash_to_pson(fldvalues, pack_empty=false) – пакует поля панобъекта в бинарный формат PSON

pson_to_namehash(pson) – преобразует PSON блок в поля панобъекта

1) Класс DatabaseSession – абстрактный класс адаптера

2) Класс SQLiteDbSession – адаптер для работы с SQLite

pan_type_to_sqlite_type(rt, size) – преобразует пандорский тип в тип записи sqlite

ruby_val_to_sqlite_val(v) – преобразует значения ruby в значения sqlite

panobj_fld_to_sqlite_tab(panobj_flds) – по матрице описания полей строит описание вновь создаваемой таблицы sqlite

connect – подключается к базе

create_table(table_name, recreate=false, arch_table=nil, arch_fields=nil, new_fields=nil) – создаёт таблицу в базе

fields_table(table_name) – возвращает список полей таблицы в виде [имя, тип]

escape_like_mask(mask) – экранирует спецсимволы маски для LIKE символом $

select_table(table_name, filter=nil, fields=nil, sort=nil, limit=nil, like_filter=nil) – делает выборку из таблицы

update_table(table_name, values, names=nil, filter=nil) – записывает данные в таблицу

3) Класс RepositoryManager – менеджер хранилищ

get_adapter(panobj, table_ptr, recreate=false) – инициировать адаптер к базе

def get_tab_update(panobj, table_ptr, values, names, filter='') – записать данные в таблицу

get_tab_fields(panobj, table_ptr) – взять список полей

4) Класс BasePanobject – базовый объект Пандоры

ider – идентификатор класса (название типа записи, например «Person»)

kind – номер класса (для Person равно 1)

sort – сортировка таблицы по умолчанию, передаётся в sql-запрос после «ORDER BY»

def_fields – массив с описанием полей таблицы

get_parent – находит родительский класс объекта Пандоры

field_des(fld_name) – возвращает описание поля в виде строки из массива def_fields

field_title(fd) – название поля на текущем языке, заголовок колонки

set_if_nil(f, fi, pfd) – установливает описание поля из родителя, если своё пустое

decode_pos(pos=nil) – разгадывает позицию метки по отношению к полю ввода

set_view_and_len(fd) – определяет способ отображения и размер ввода для поля

tab_fields(reinit=false) – берет описание полей из sql-таблицы

expand_def_fields_to_parent(reinit=false) – расширяет описание полей, беря недостающие значения от родителя

def_hash(fd) – формула для компонента панхэша по его типу

panhash_pattern(auto_calc=true) – шаблон для вычиисления панхэша

select(afilter=nil, set_namesvalues=false, fields=nil, sort=nil, limit=nil, like_filter=nil) – делает выборку из таблицы

update(values, names=nil, filter='', set_namesvalues=false) – записывает данные в таблицу

field_val(fld_name, values) – выбирает значение по имени поля

field_des(fld_name) – возвращает описание поля в виде строки из массива def_fields

lang_to_str(lang) – краткое обозначение языка по его коду

panhash_formula – формула панхэша для показа

calc_hash(hfor, hlen, fval) – рассчитывает компоненту панхэша

show_panhash(val, prefix=true) – панхэш для показа

panhash(values, lang=0, prefix=true, hexview=false) – рассчитывает панхэш

matter_fields – сущностные поля (входящие в панхэш)

clear_excess_fields(row) – удалить избыточные поля

5. Работа с параметрами

create_base_id – создаёт новый идентификатор базы

decode_param_setting(setting) – распознаёт атрибуты параметра

normalize_param_value(val, type) – нормализует значение параметра

calc_default_param_val(type, setting) – вычисляет значение по умолчанию параметра

get_model(ider, models=nil) – возвращает экземпляр модели

get_param(name, get_id=false) –возвращает значение параметра

set_param(name, value, definition=nil) – задаёт значение параметра

5) Класс RoundQueue – циклический буфер

add_block_to_queue(block, max=MaxQueue) – добавить блок в очередь

single_read_state(max=MaxQueue) – состояние одиночной очереди

get_block_from_queue(max=MaxQueue, ptrind=nil) – взять блок из очереди

6. Работа с капчой

generate_captcha(drawing=nil, length=6, height=70, circles=5, curves=0) – сгенерировать капчу

7. Мультимедиа функции

is_64bit_os? – ОС является 64-битной?

win_exec(cmd) – запустить в Винде

play_mp3(filename, path=nil) – проиграть mp3

elem_stopped?(elem) – элемент остановлен

elem_playing?(elem) – элемент проигрывается

II. Модуль PandoraModel – логическая модель

1) Класс Panobject – корневой класс объектов Пандоры

1. Управление моделью

load_model_from_xml(lang='ru') – сформировать описание модели по XML-файлу

panobjectclass_by_kind(kind) – класс панобъекта по коду типа

normalize_trust(trust, to_int=nil) – нормализовать и преобразовать доверие

get_record_by_panhash(kind, panhash, pson_with_kind=nil, models=nil, getfields=nil) – читает запись по панхэшу

save_record(kind, lang, values, models=nil, require_panhash=nil) – сохранить запись

relation_is_symmetric?(relation) – связь симметрична

act_relation(panhash1, panhash2, rel_kind=RK_Unknown, act=:check, creator=true, init=false, models=nil) – проверяет, создаёт или удаляет связь между двумя объектами

III. Модуль PandoraCrypto – криптографический модуль Пандоры

1. Битовые преобразование

klen_to_bitlen(len) – код длины ключа в байтовую длину ключа

bitlen_to_klen(len) – байтовая длина ключа в код длины

divide_type_and_klen(tnl) – разделить тип и код длины

2. Ключи, шифрование и подпись

encode_cipher_and_hash(cipher, hash) – упаковать коды методов шифровки и хэширования

decode_cipher_and_hash(cnh) – распаковать коды методов шифровки и хэширования

pan_kh_to_openssl_hash(hash_len) – получает объект OpenSSL по коду хэша Пандоры

pankt_to_openssl(type) – преобразует тип хэша Пандоры в имя OpenSSL

pankt_len_to_full_openssl(type, len) – преобразует тип хэша Пандоры в строку OpenSSL

key_recrypt(data, encode=true, cipher_hash=nil, cipherkey=nil) – зашифровать или расшифровать ключ

generate_key(type_klen = KT_Rsa | KL_bit2048, cipher_hash=nil, cipherkey=nil) – генерирует ключ или ключевую пару

init_key(key_vec) – инициализирует ключ или ключевую пару

make_sign(key, data, hash_len=KH_Sha2 | KL_bit256) – создает подпись

verify_sign(key, data, sign, hash_len=KH_Sha2 | KL_bit256) – проверяет подпись

recrypt(key_vec, data, encrypt=true, private=false) – зашифровывает или расшифровывает данные

3. Высокоуровневая криптография

deactivate_key(key_vec) – деактивирует текущий или указанный ключ

reset_current_key – деактивирует текущий ключ

current_key(switch_key=false, need_init=true) – возвращает текущий ключ или позволяет выбрать и активировать ключ

read_key(panhash, passwd, key_model) – считывает ключ из базы

recrypt_key(key_model, key_vec, cipher, panhash, passwd, newpasswd) – перекодирует ключ

current_user_or_key(user=true, init=true) – возвращает панхэш текущего пользователя или ключа

sign_panobject(panobject, trust=0, models=nil) – подписывает PSON ПанОбъекта и сохраняет подпись как запись

unsign_panobject(obj_hash, delete_all=false, models=nil) – удаляет подписи по заданному панхэшу

trust_of_panobject(panhash, models=nil) – возвращает доверие к панобъекту по его панхэшу

rate_of_panobj(panhash, depth=$query_depth, querist=nil, models=nil) – вычислить рейтинг панобъекта

open_key(panhash, models=nil, init=true) – активировать ключ по заданному панхэшу

name_and_family_of_person(key, person=nil) – возвращает имя и фамилию человека

short_name_of_person(key, person=nil, view_kind=0, othername=nil) – возвращает короткое имя человека

find_sha1_solution(phrase) – находит sha1-загадку

check_sha1_solution(phrase, add) – проверяет sha1-загадку

IV. Модуль PandoraNet – сетевые классы Пандоры

1) Класс Pool – пул

add_to_white(ip) –

is_white?(ip) – ip в белом списке?

is_black?(ip) – ip в черном списке?

add_session(conn) – добавляет сессию в список

del_session(conn) – удаляет сессию из списка

session_of_node(node) – возвращает сессию для узла

session_of_keybase(keybase) – возвращает сессию по ключу и идентификатору базы

session_of_key(key) – возвращает сессию по панхэшу ключа

session_of_person(person) – возвращает сессию по панхэшу человека

sessions_on_dialog(dialog) – возвращает сессию по диалогу

init_session(node=nil, keybase=nil, send_state_add=nil, dialog=nil, node_id=nil) – находит или создает соединение с нужным узлом

stop_session(node=nil, keybase=nil, disconnect=true) #, wait_disconnect=true) – останавливает соединение с заданным узлом

encode_node(host, port, proto) – формирует маркер узла

decode_node(node) – распаковывает маркер узла

check_callback_addr(addr, host_ip) – стукануться по обратному адресу

init_fish_for_fisher(fisher, in_lure, aim_keyhash=nil, baseid=nil) – инициализирует рыбку для заданного рыбака

2) Класс Session – сессия

set_keepalive(client) – установить опции сокета

pool – ссылка на пул

get_type – тип сессии

unpack_comm(comm) – распаковать команду

unpack_comm_ext(comm) – распаковать расширение команды

send_comm_and_data(index, cmd, code, data=nil) – отправляет команду, код и данные (если есть)

err_scmd(mes=nil, code=nil, buf=nil) – компонует команду ошибки и логирует сообщение

add_send_segment(ex_comm, last_seg=true, param=nil, ascode=nil) – добавляет сегмент в пакет и отправляет если пора

set_request(panhashes, send_now=false) – компонует команду запроса записи/записей

set_query(list, time, send_now=false) – компонует команду запроса сорта/сортов

accept_segment – принять полученный сегмент

recognize_params – распознает данные приветствия

set_max_pack_size(stage) – ставит лимит на допустимый размер пакета

init_skey_or_error(first=true) – отреагировать на приветствие

get_sphrase(init=false) – сгенерировать случайную фразу

send_captcha – компоновать команду с капчой

update_node(skey_panhash=nil, sbase_id=nil, trust=nil, session_key=nil) – обновить запись об узле

process_media_segment(cannel, mediabuf) – обработать медиа сегмент

get_simple_answer_to_node – взять пароль для упрощенной авторизации

take_out_lure_for_fisher(fisher, in_lure) – взять исходящую наживку по входящей наживке для заданного рыбака

get_out_lure_for_fisher(fisher, in_lure) – проверить исходящую наживку по входящей наживке и рыбаку

get_fisher_for_out_lure(out_lure) – определить рыбака по исходящей наживке

free_out_lure_of_fisher(fisher, in_lure) – очистить исходящие наживки для рыбака и входящей наживки

set_fish_of_in_lure(in_lure, fish) – поставить рыбку на входящую наживку

get_fish_for_in_lure(in_lure) – взять рыбку по входящей наживке

free_fish_of_in_lure(in_lure) – очистить рыбку для входящей наживки

send_segment_to_fish(in_lure, segment) – отправляет сегмент от текущей рыбацкой сессии к сессии рыбки

send_segment_to_fisher(out_lure, segment) – отправляет сегмент от текущей сессии к сессии рыбака

socket_recv(maxsize) – прочитать следующие данные из сокета, или вернуть nil, если сокет закрылся

initialize(asocket, ahost_name, ahost_ip, aport, aproto, aconn_state, anode_id, a_dialog, tokey, send_state_add) – запускает три цикла сессии: чтение из очереди, чтение из сокета, отправка (общий)

1. Функции слушания и охоты

get_listener_client_or_nil(server) – взять следующий сокет клиента со слушателя, или вернуть nil

get_exchage_params – взять параметры обмена

start_or_stop_listen – открывает серверный сокет и начинает слушать

hunt_nodes(round_count=1) – начать охоту

V. Модуль PandoraGtk – графический интерфейс Пандоры

1) Класс AdvancedDialog – продвинутое окно диалога

initialize(*args) – метод создания

run2 – показать диалог в модальном режиме

2) Класс SafeToggleToolButton – ToggleToolButton с безопасным переключением "active"

safe_signal_clicked – запомнить обработчик сигнала

safe_set_active(an_active) – безопасно установить свойство "active"

3) Класс SafeCheckButton – CheckButton с безопасным переключением "active"

safe_signal_clicked – запомнить обработчик сигнала

safe_set_active(an_active) – безопасно установить свойство "active"

4) Класс MaskEntry – поле ввода с допустимыми символами в маске

5) Класс IntegerEntry – поле ввода целых чисел

6) Класс FloatEntry – поле ввода дробных чисел

7) Класс HexEntry – поле ввода шестнадцатеричных чисел

8) Класс Base64Entry – поле ввода Base64

9) Класс DateEntrySimple – поле ввода даты (простое)

10) Класс TimeEntry – поле ввода даты и времени

11) Класс DateEntry – поле ввода даты

12) Класс PanhashBox – поле ввода панхэша

set_classes – определить допустимые классы Пандоры

13) Класс CoordEntry – поле ввода координаты

14) Класс CoordBox – поле ввода координат

15) Класс ExtTextView – расширенный TextView

before_addition(cur_time=nil, vadj_value=nil) – выполнить перед добавлением

after_addition(go_to_end=nil) – выполнить после добавления

16) Класс SubjTreeView – таблица для объектов Пандоры

17) Класс SubjTreeViewColumn – колонка для SubjTreeView

18) Класс PanobjScrollWin – ScrolledWindow для объектов Пандоры

19) Класс FieldsDialog – диалог с полями ввода

add_menu_item(label, menu, text) – добавляет пункт в меню

set_view_buffer(format, view_buffer, raw_buffer) – задает тестовый буфер для просмотра

set_raw_buffer(format, raw_buffer, view_buffer) – задает сырой тестовый буфер

set_buffers(init=false) – задать буферы

set_tag(tag) – задать тэг для выделенного

initialize(apanobject, afields=[], *args) – создать форму с полями

calc_field_size(field) – вычислить размер поля

calc_row_size(row) – вычислить размер ряда

on_resize_window(window, event) – событие при изменении размеров окна

20) Класс TabLabelBox – бокс закладки для блокнота с картинкой и кнопкой

21) Класс ViewDrawingArea – DrawingArea для вывода видео

set_expose_event(value) – устанавливает обработчик события expose

22) Класс DialogScrollWin – диалог разговора

initialize(known_node, a_room_id, a_targets) – показать диалог общения

add_mes_to_view(mes, key_or_panhash=nil, myname=nil, modified=nil, created=nil, to_end=nil) – добавляет сообщение в диалог

load_history(max_message=6, sort_mode=0) – подгрузить историю сообщений

get_name_and_family(i) – определить имя и фамилию

set_session(session, online=true) – задать сессию

add_and_send_mes(text) – отправляет сообщение на узел

update_state(received=true, curpage=nil) – обновляет цвет закладки при получении новых данных

parse_gst_string(text) – распознаёт строку Gstreamer

append_elems_to_pipe(elements, pipeline, prev_elem=nil, prev_pad=nil, name_suff=nil) – добавляет элементы в конвейер

add_elem_to_pipe(str, pipeline, prev_elem=nil, prev_pad=nil, name_suff=nil) – добавляет элемент в конвейер

link_sink_to_area(sink, area, pipeline=nil) – прицепляет сливной элемент к области виджета

set_xid(area, sink) – устанавливает дескриптор окна

get_video_sender_params(*) – берёт параметры отправителя видео

init_video_sender(start=true, just_upd_area=false) – инициализирует отправщика видео

get_video_receiver_params(*) – берёт параметры приёмщика видео

init_video_receiver(start=true, can_play=true, init=true) – инициализирует приёмщика видео

get_audio_sender_params(*) – берёт параметры отправителя аудио

init_audio_sender(start=true, just_upd_area=false) – инициализирует отправителя аудио

get_audio_receiver_params(*) – берёт параметры приёмщика аудио

init_audio_receiver(start=true, can_play=true, init=true) – инициализирует приёмщика аудио

23) Класс SearchScrollWin – панель поиска

search_in_bases(text, th, bases='auto') – поиск в базах

initialize(text=nil) – показать окно поиска

24) Класс ProfileScrollWin – панель профиля

initialize(a_person=nil) – показать окно профиля

25) Класс SessionScrollWin – список сеансов

initialize(session=nil) – показать окно сессий

1. Работа с элементами

set_readonly(widget, value=true, sensitive=true) – установить виджету режим только для чтения

hack_enter_bug(enterbox) – исправляет баг с исчезновением нажатия Enter

hack_grab_focus(widget_to_focus) – исправляет баг с неработающей постановкой фокуса

create_menu_item(mi, treeview=nil) – создание пункта меню по его описанию

set_statusbar_text(statusbar, text) – задает текст статусной строки

add_tool_btn(toolbar, stock, title, toggle=nil) – добавить кнопку на панель инструментов

2. Обновление

start_updating(all_step=true) – проверить обновления и скачать их

update_file(http, tail, pfn) – обновить файл

3. Работа с записями

act_panobject(tree_view, action) – выполнить действие над выделенной записью

get_panobject_icon(panobj) – взять иконку ассоциированную с панобъектом

show_panobject_list(panobject_class, widget=nil, sw=nil, auto_create=false) – показ списка панобъектов

4. Работа с комнатами

set_send_ptrind_by_room(room_id) – взять индекс указателя для отправки по id комнаты

get_send_ptrind_by_room(room_id) – проверить индекс указателя для отправки по id комнаты

nil_send_ptrind_by_room(room_id) – сбросить индекс указателя для отправки для комнаты

extract_targets_from_panhash(targets, panhashes) – получить панхэш персоны по произвольному панхэшу

extend_targets_by_relations(targets) – расширить списки персон, узлов и ключей пройдясь по связям

start_extending_targets_by_hunt(targets) – запуск потока, которые ищет дополнительные узлы и ключи

construct_room_id(persons) – создать идентификатор комнаты

find_active_sender(not_this=nil) – найти активного отправителя

5. Общие методы

get_view_params – взять параметры вида

get_main_params – взять основные параметры

show_about – показ окна "О программе"

show_talk_dialog(panhashes, known_node=nil) – показать диалог общения

show_search_panel(text=nil) – показать панель поиска

show_profile_panel(a_person=nil) – показать панель профиля

show_session_panel(session=nil) – показать список сеансов

26) Класс PandoraStatusIcon – иконка в трее

initialize(a_update_win_icon=false, a_flash_on_new=true, a_flash_interval=0, a_play_sounds=true, a_hide_on_minimize=true) – создает иконку в трее

create_menu – создает и показывает всплывающее меню

set_online(state=nil) – задаёт статус «онлайн»

set_message(message=nil) – задаёт статус "есть сообщение"
set_flash(flash=true) – задаёт мигание
update_icon – обновляет иконку
timeout_func – ставит таймер на шаг мигания
icon_activated(top_sens=true, force_show=false) – действия при нажатии на иконку

27) Класс CaptchaHPaned – панель с капчой

initialize(first_child) – показать панель
show_captcha(srckey, captcha_buf=nil, clue_text=nil, node=nil) – показать капчу

29) Класс MainWindow – главное окно

update_conn_status(conn, session_type, diff_count) – обновить состояние подключений
correct_lis_btn_state – изменить состояние кнопки слушателя
correct_hunt_btn_state – изменить состояние кнопки охотника
add_status_field(index, text) – добавляет поле в статусбар
set_status_field(index, text, enabled=nil, toggle=nil) – задаёт свойства поля в статусбаре
get_status_field(index) – возвращает поле статусбара
construct_room_title(dialog, check_all=true) – задаёт осмысленный заголовок окна
do_menu_act(command, treeview=nil) – обработчик события меню
fill_menubar(menubar) – заполнить главное меню
fill_toolbar(toolbar) – заполнить панель инструментов
init_scheduler(interval=nil) – инициировать планировщик
initialize(*args) – показать главное окно Gtk

Функции и классы вне модулей

delete_psocket – удаляет unix-сокет Пандоры
init_win32api – инициализирует модуль win32
AsciiString – класс строки в виде Ascii
Utf8String – класс строки в виде Utf8