Архитектура и принципы работы системы

Руководство по эксплуатации

Содержание

- Регистрация по доменному имени, где IP первый адрес ресолвится в SMG, второй и третий в SSW

Состав и архитектура комплекса

В состав комплекса ECSS-10 входят:

- *Гибкий коммутатор SSW* — выполняет управление обслуживанием вызовов (функции контроллера медиашлюзов MGC).
- *Ethernet-коммутаторы* — обеспечивают организацию надежной внутренней сети передачи данных.
  Используются стекируемые коммутаторы операторского уровня производства компании Элтекс или оборудование других производителей с требуемыми характеристиками.
- *Промышленные серверы* разных производителей на платформах Intel и AMD, конфигурация которых выбирается в зависимости от требуемой производительности системы.
- *Шлюзы доступа (access gateway)* — аппаратно-программные комплексы производства Элтекс или других производителей, через которые осуществляется подключение абонентских устройств по двухпроводным линиям.
- *Платформа доступа MSAN* — универсальная платформа доступа, выполняющая функцию шлюза доступа с высокой плотностью портов.
- *Цифровые сигнальные и медиашлюзы* — аппаратно-программные комплексы производства Элтекс, через которые осуществляется подключение к соединительным линиям ТфОП, а также подключение абонентских выносов.
- *Пограничные шлюзы* — аппаратно-программные комплексы производства Элтекс, выполняющие функцию пограничного контроллера сессий (SBC) и реализующие уровень согласования IP-подсетей (SIP-ALG).
- *Медиасерверы* — аппаратно-программные комплексы производства Элтекс, обеспечивающие функционал обработки медиапотоков с возможностью детектирования и генерации тональных сигналов, записи и проигрывания медиапотока, транскодирования.
- *Посредник СОРМ* — аппаратно-программный комплекс производства Элтекс, обеспечивающий подключение пульта управления СОРМ к комплексу ECSS-10.

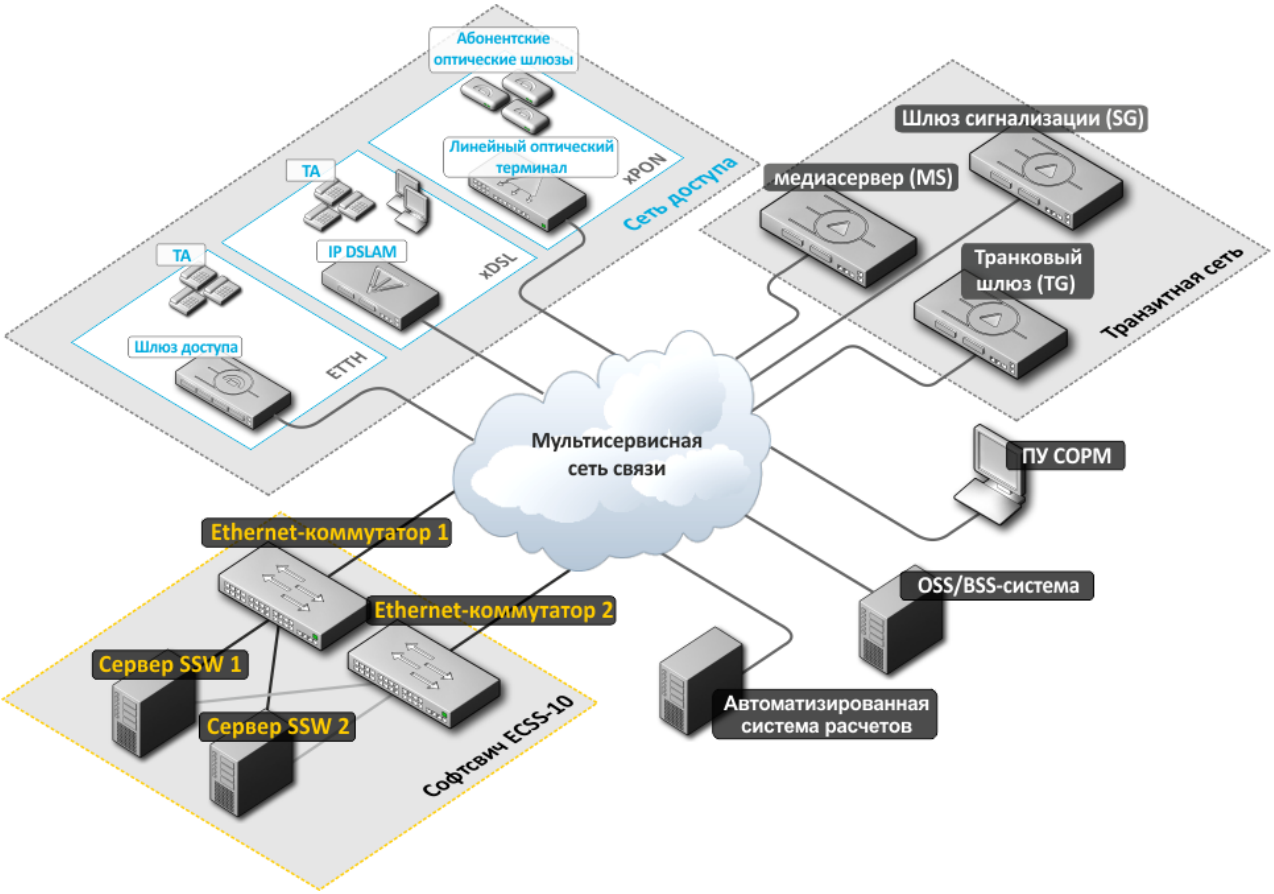


Рисунок 1 — Пример схемы включения комплекса ECSS-10

Гибкий коммутатор SSW является системой, состоящей из набора промышленных серверов и запускаемого на них специализированного программного обеспечения.

В комплексе используются промышленные серверы на платформах Intel и AMD от ведущих производителей.
Конфигурация выбирается в зависимости от требуемых показателей системы (цена, производительность, надежность).
Система ECSS-10 в минимальной версии может быть развернута на одном сервере (без поддержки аппаратного резервирования).
Минимальная конфигурация с поддержкой аппаратного резервирования строится на двух серверах, рисунок 2.
Высоконагруженные решения разворачиваются на большем количестве серверов, число которых определяется параметрами определенного узла связи.

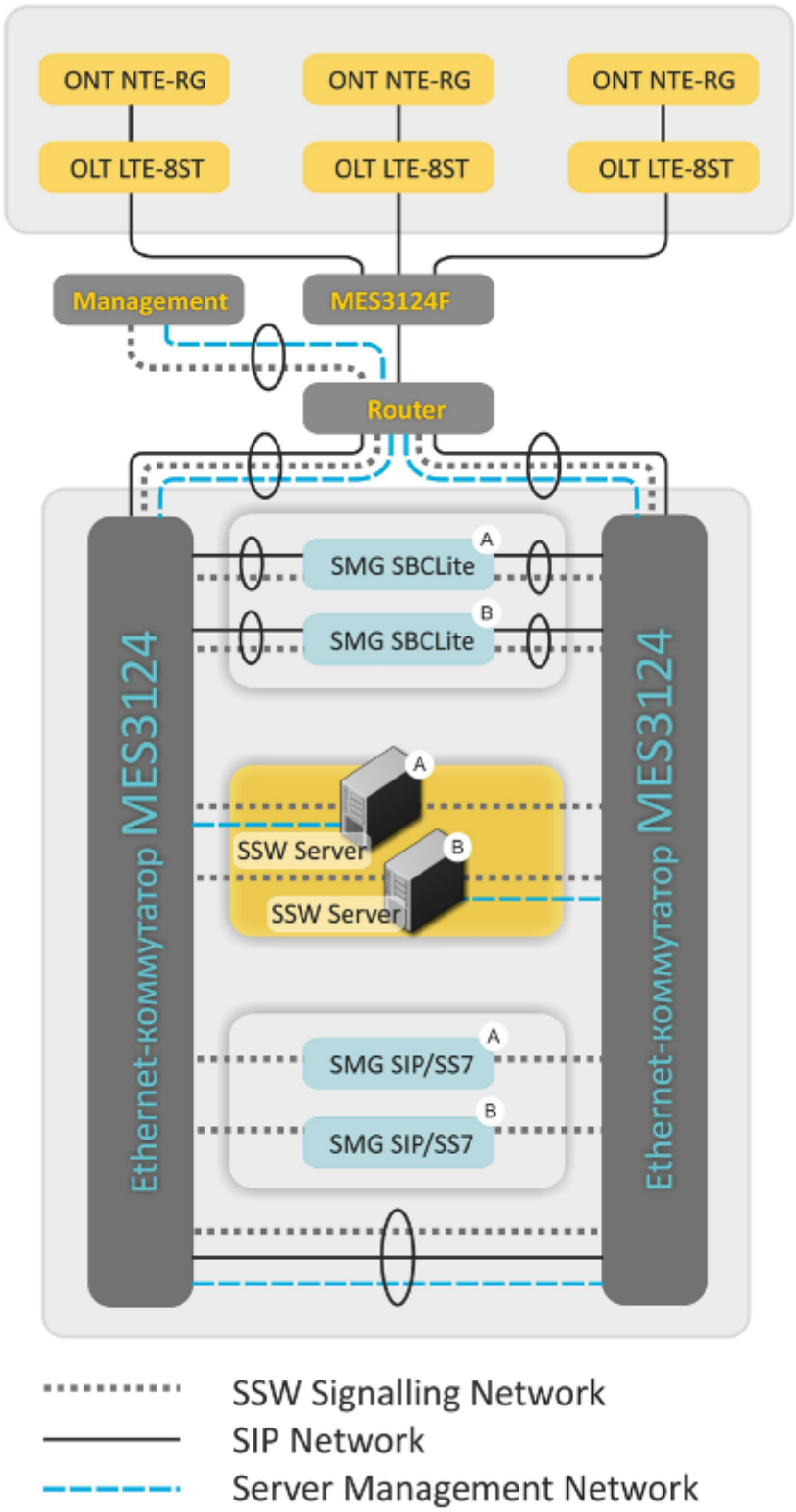


Рисунок 2 — Схема включения оборудования ECSS-10:

- ONT NTE-RG — абонентский оптический терминал производства компании Элтекс.

- OLT LTE-8ST — станционный оптический терминал производства компании Элтекс.
- MES3124/MES3124F — стекируемый Ethernet-коммутатор L2+ операторского уровня производства компании Элтекс.
- Management — терминалы управления.
- Router — маршрутизатор.
- SMG SBCLite — пограничный контроллер сессий на базе SMG-1016M.
- SSW Server — промышленные серверы SSW.
- SMG SIP/SS7 — шлюз сигнализации на базе SMG-1016M.

Типовые варианты построения систем

ECSS-10 является универсальной системой, на базе которой могут быть построены как типовые узлы связи, так и различные их комбинации.

Узел связи корпоративного уровня

Использование ECSS-10 в качестве коммуникационной платформы корпоративного уровня позволяет:

- объединить офисы компании вне зависимости от их географического положения в единую корпоративную сеть;
- организовать короткую нумерацию между сотрудниками компании в различных офисах;
- предоставить всем сотрудникам единообразные телефонные сервисы и средства для совместной работы;
- обеспечить постоянную доступность сотрудников с помощью мобильных приложений и соответствующих сервисов;
- использовать различные виды терминального оборудования: программные SIP-клиенты, аналоговые телефоны, IP-телефоны;
- контролировать качество обслуживания клиентов с помощью записи разговоров;
- снизить расходы на междугородние переговоры между сотрудниками различных офисов;
- снизить расходы на командировки при использовании сервиса видео-конференц-связи.

На рисунке 3 представлена распределённая корпоративная инфокоммуникационная сеть на основе оборудования производства компании Элтекс. В центральном офисе устанавливается платформа ECSS-10, которая является ключевым элементом данной сети. Подключение абонентской ёмкости к платформе осуществляется с помощью программных SIP-клиентов и IP-телефонов напрямую, с помощью аналоговых телефонов — через абонентские шлюзы TAU-72.IP. Стыковка с телефонной сетью общего пользования (ТфОП) выполнена через цифровой шлюз SMG-1016M. Дополнительные офисы и мобильные клиенты подключаются через пограничный контроллер сессий (SBC), который обеспечивает функции безопасности от внешних угроз. В крупных дополнительных офисах возможна установка отдельных платформ ECSS-10 для концентрации внутреннего трафика. Подключение платформ в дополнительных офисах к центральной системе осуществляется с помощью соответствующих SIP-транков.

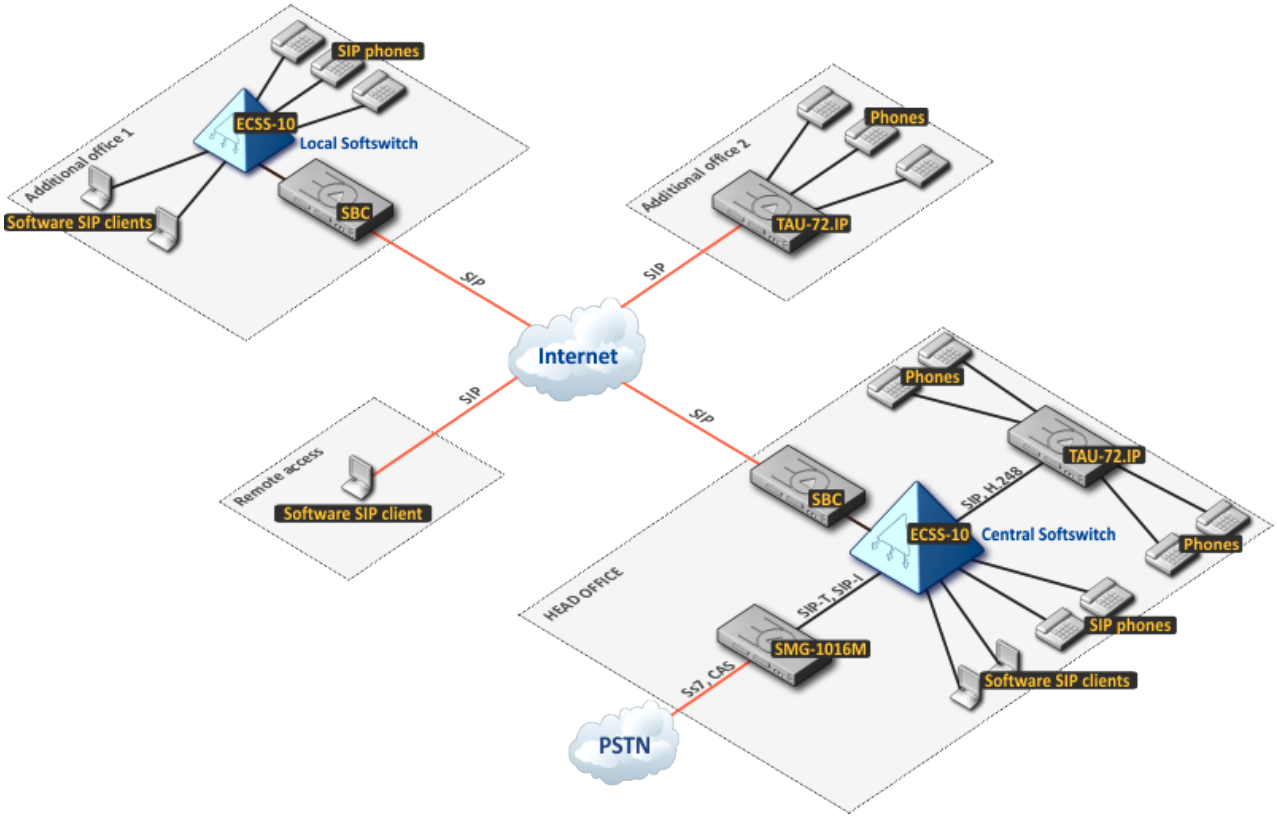


Рисунок 3 — Пример построения распределённой корпоративной сети на основе ECSS-10

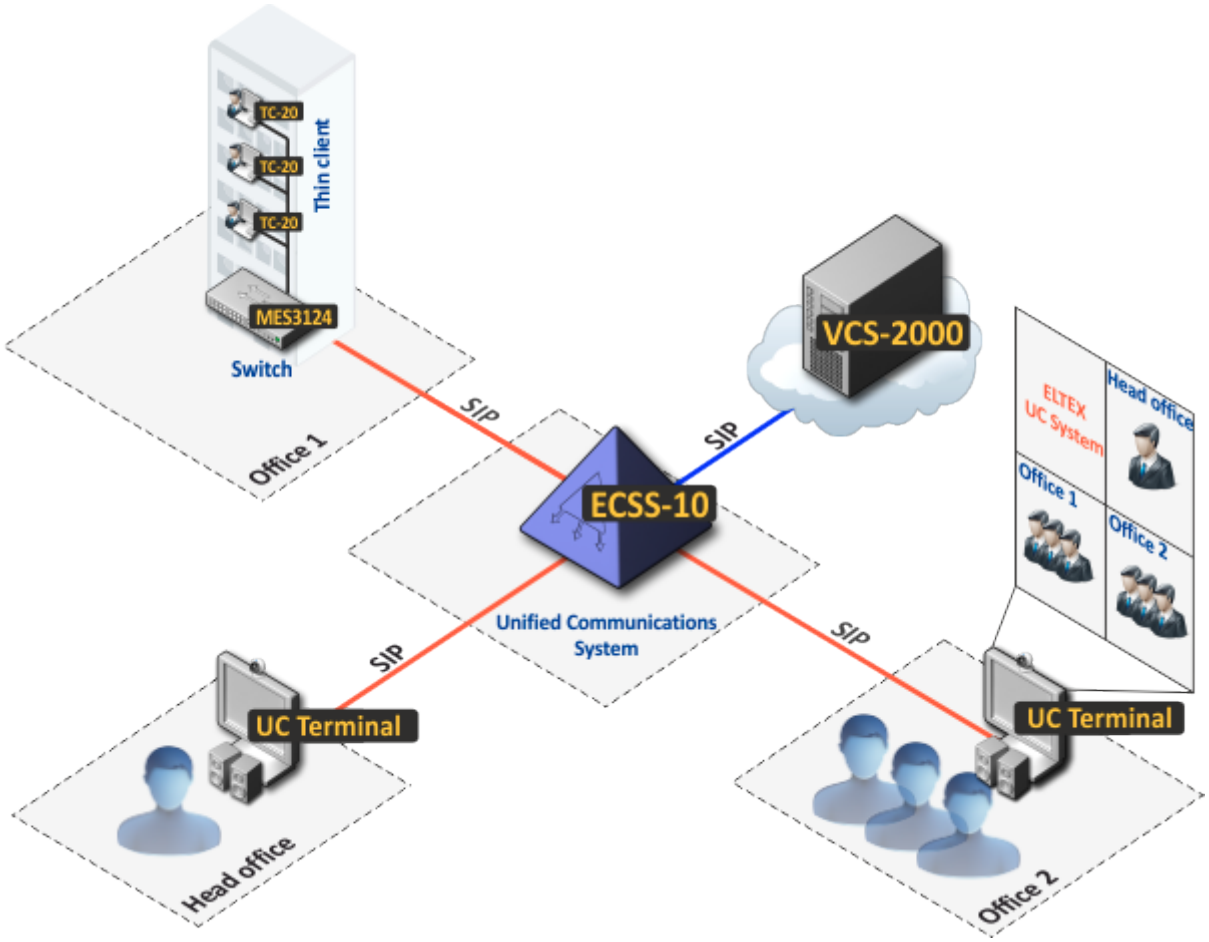


Рисунок 4 — Модель реализации сервиса видео-конференц-связи на основе платформы ECSS-10

Узел связи операторского уровня

Модульная архитектура комплекса ECSS-10 позволяет строить на его основе высоконагруженные отказоустойчивые узлы связи различного уровня: местный, зоновый, оконечно-транзитный, междугородний, международный. Платформа ECSS-10 удовлетворяет всем требованиям действующих российских нормативных документов для Системы оперативно-розыскных мероприятий (СОРМ) и имеет все виды российских сертификатов*.

*Сертификация платформы ECSS-10 на соответствие требованиям для междугородних, международных узлов возможна по дополнительному запросу.

Местный узел связи

Система ECSS-10 может использоваться как для построения новых узлов местной связи, так и для модернизации существующих, в том числе узлов связи на основе технологии коммутации каналов. Поддержка широкого набора интерфейсов обеспечивает возможность подключения различных оконечных устройств:

- абонентских VoIP-шлюзов (например, TAU-72.IP, TAU-8.IP), поддерживающих стандартные протоколы SIP, H.248 и H.323;
- мультисервисных платформ абонентского доступа MSAN с платами для подключения аналоговых телефонов (например, MC-1000PX);
- цифровых АТС через встроенные или внешние IP-шлюзы (например, MC240 с встроенным IP-шлюзом TM.IP);
- абонентских PON-терминалов при построении сетей доступа FTTx (например, NTE-RG-XX, NTP-RG-XX).

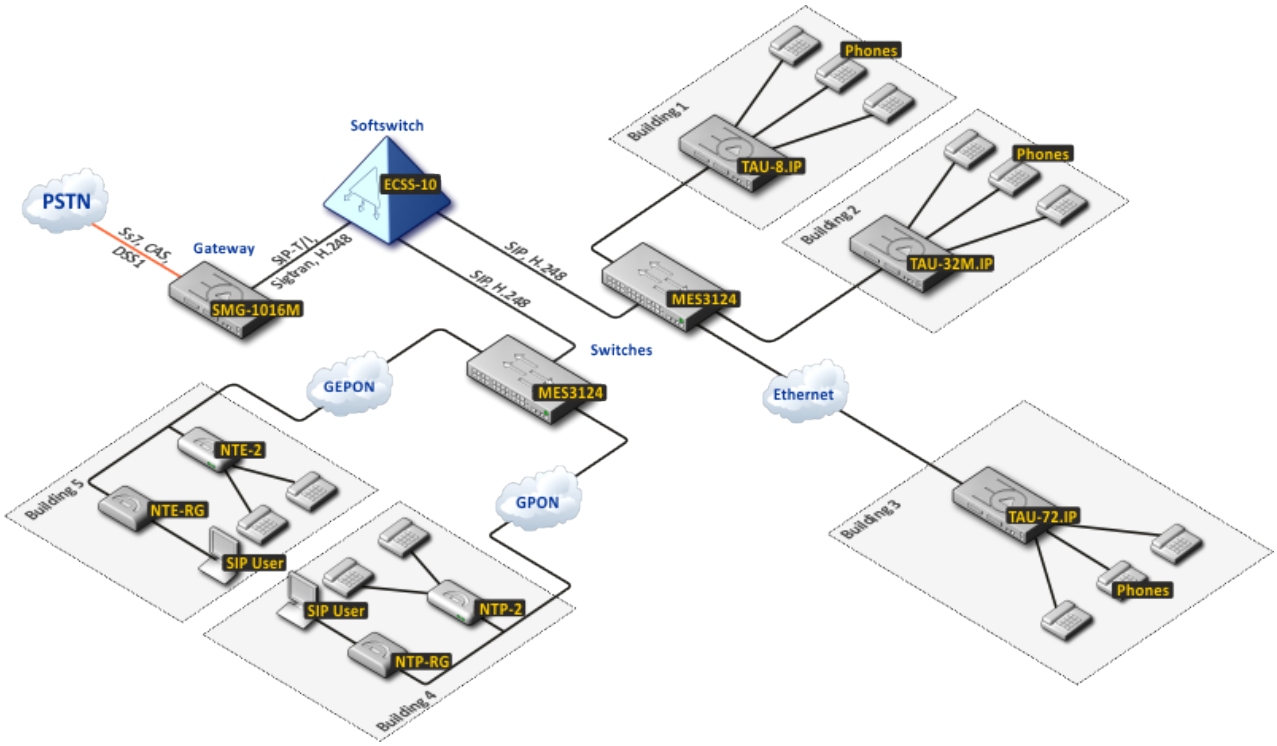


Рисунок 5 — Пример построения городского узла связи на основе ECSS-10

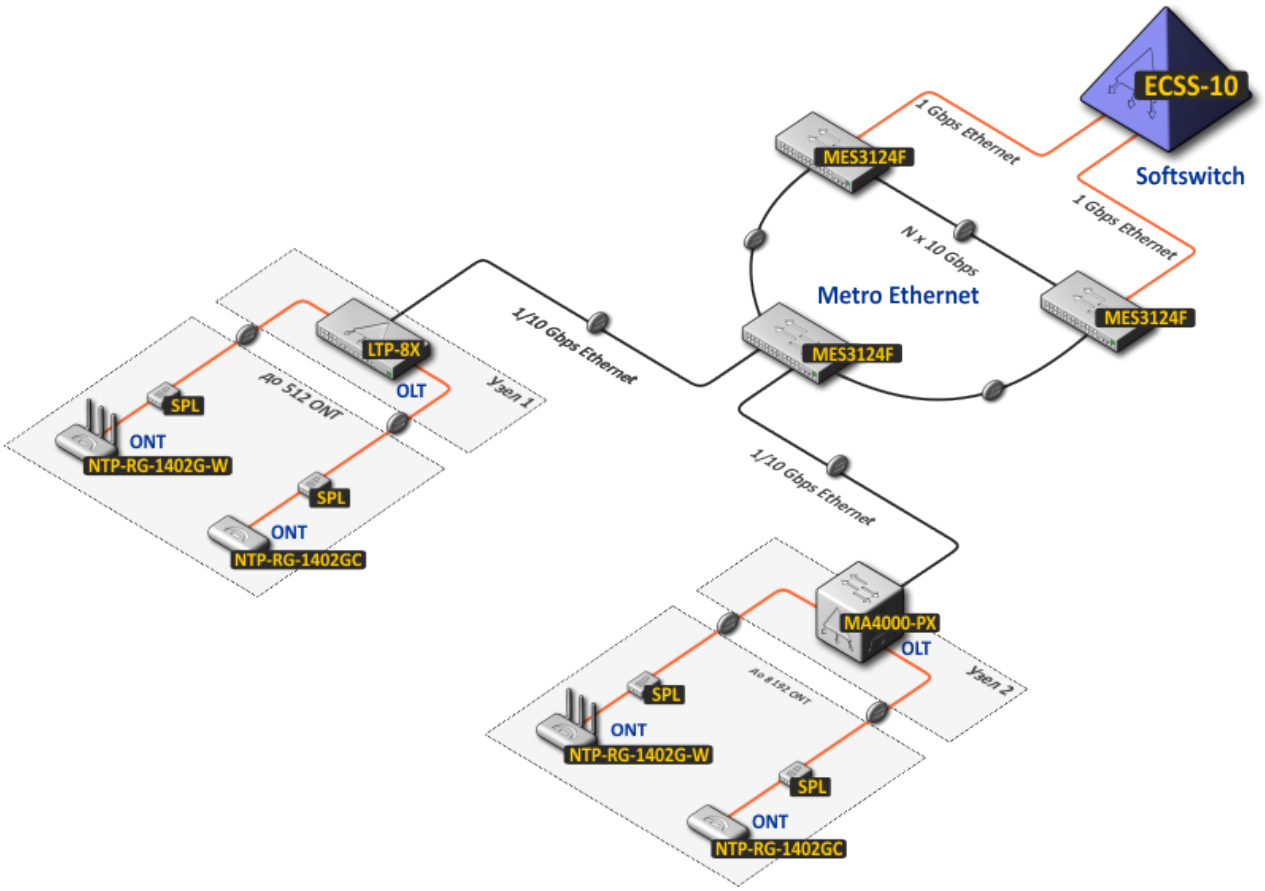


Рисунок 6 — Пример построения современной IP-сети с использованием на "последней миле" технологии PON

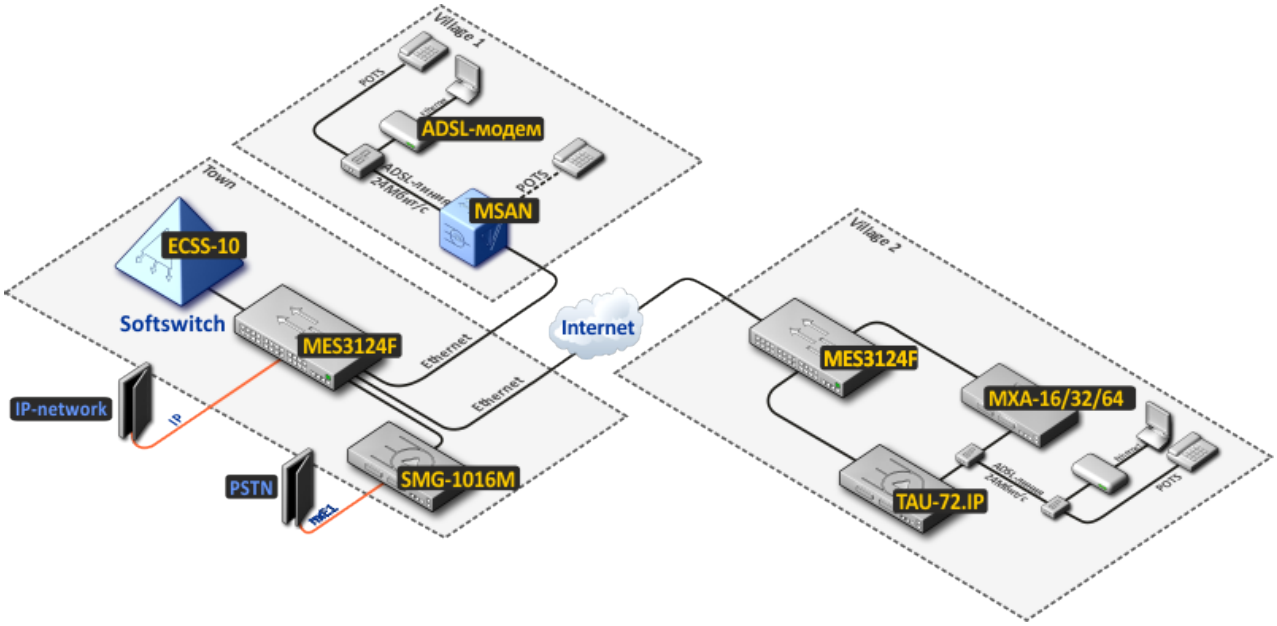


Рисунок 7 — Пример построения сельского узла связи на основе ECSS-10

Зоновый узел связи

Высокая надёжность платформы ECSS-10, которая является основным требованием к узлам связи подобного уровня, обеспечивается модульностью, кластеризацией на наборе доступных вычислительных ресурсов и динамическим распределением нагрузки между компонентами системы. Строгое соответствие существующим рекомендациям и стандартам в области IP-телефонии обеспечивает совместимость комплекса ECSS-10 с большинством платформ других производителей, обеспечивая тем самым простоту инсталляции и минимальные сроки запуска системы в коммерческую эксплуатацию. Поддержка различных механизмов маршрутизации вызовов позволяет гибко создавать индивидуальные сценарии обработки соединений.

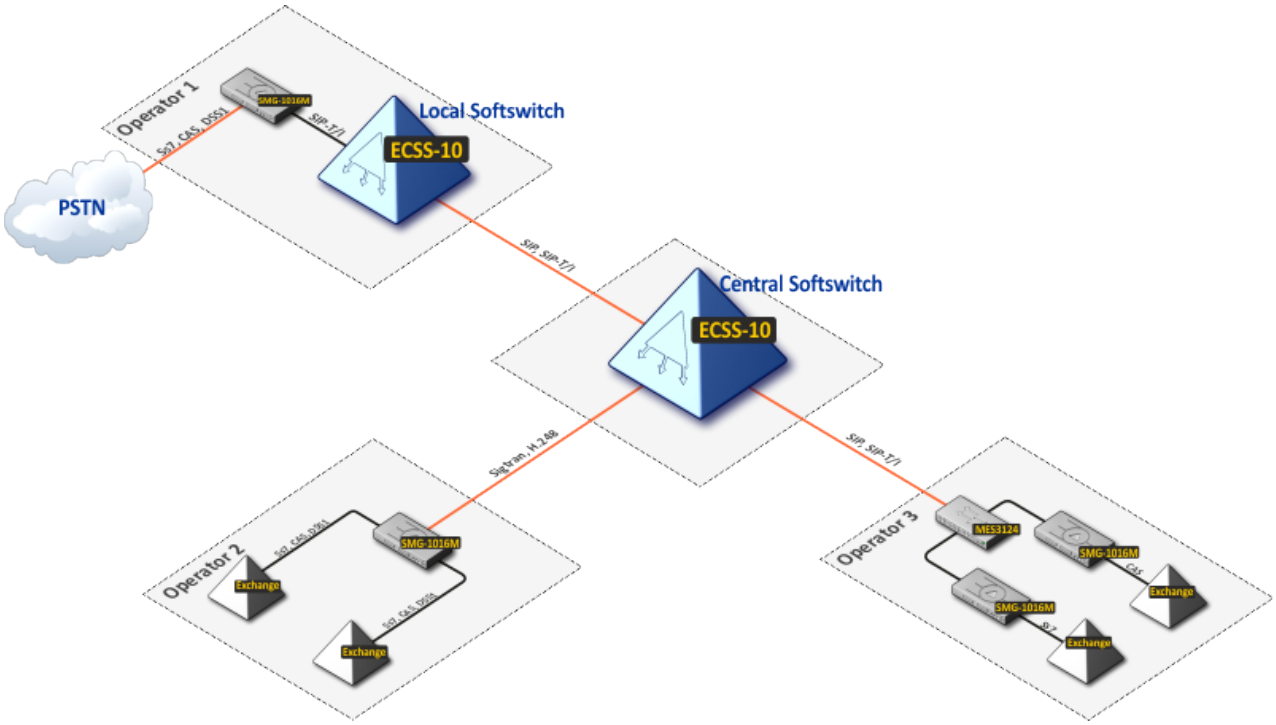


Рисунок 8 — Пример построения зонового узла связи на основе ECSS-10

Оконечно-транзитный узел связи

Программно-аппаратный комплекс ECSS-10 может быть использован одновременно как для организации единой точки подключения к внешней сети нескольких цифровых АТС, так и для подключения абонентов с помощью абонентских VoIP-шлюзов с единой системой маршрутизации соединений

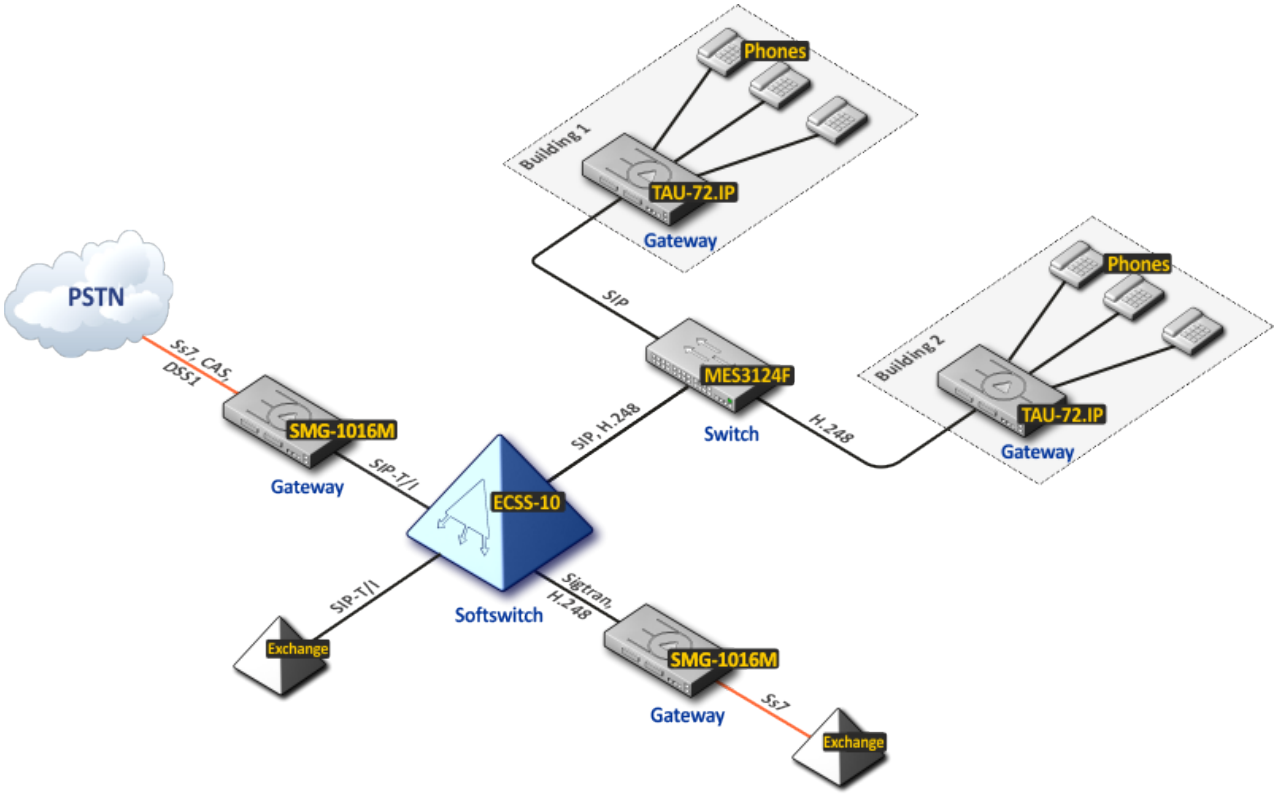


Рисунок 9 — Пример построения оконечно-транзитного узла связи на основе ECSS-10

SaaS платформа

Возможность логического разделения ресурсов комплекса ECSS-10 позволяет использовать его в качестве облачной платформы для предоставления виртуальных АТС с поддержкой широкого набора сервисов (в том числе видеосвязи и видео-конференц-связи). Оператору/сервис-провайдеру предоставляется гибкий инструментарий для разработки индивидуальных сервисов.

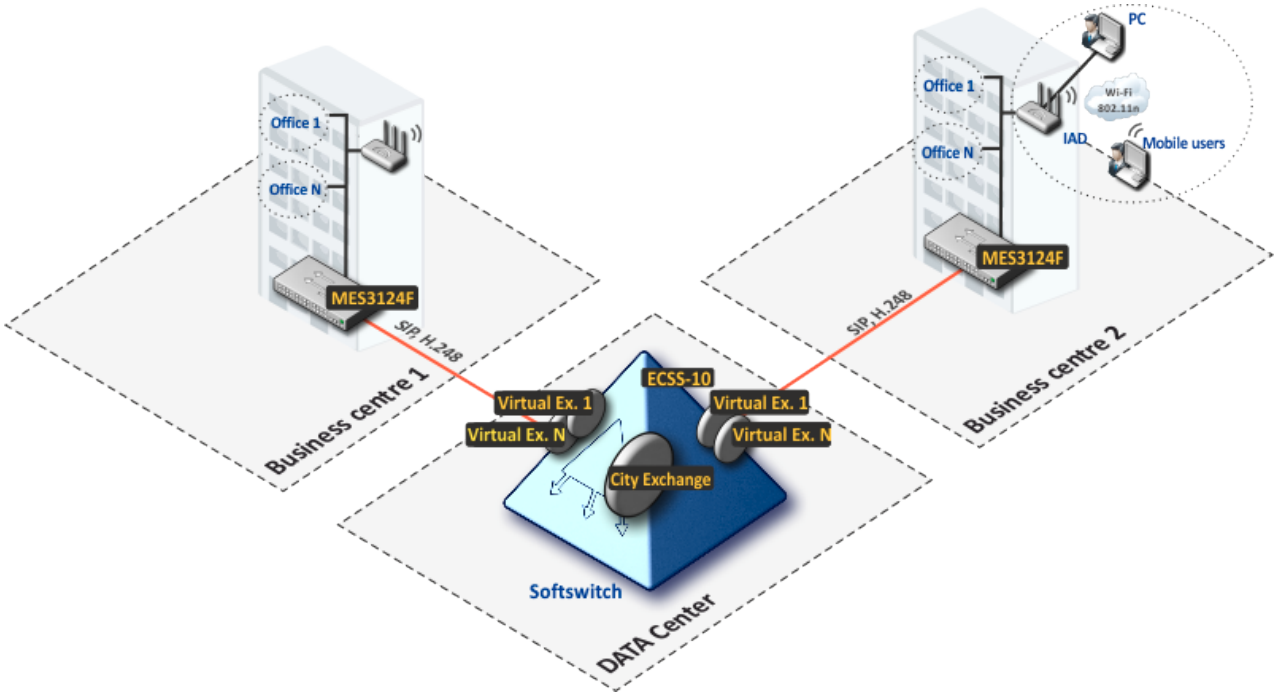


Рисунок 10 — Пример построения SaaS платформы на основе ECSS-10

Применение ECSS-10 позволяет:

- осуществлять постепенную модернизацию оборудования существующей сети;
- расширить спектр предоставляемых пользователям сервисов;
- гибко формировать набор сервисов для каждого пользователя;
- поэтапно увеличивать емкость узла связи без больших капитальных затрат;
- гибко модифицировать структуру узла связи по мере возникновения новых задач и потребностей.

Типовые варианты распределения подсистем на серверах

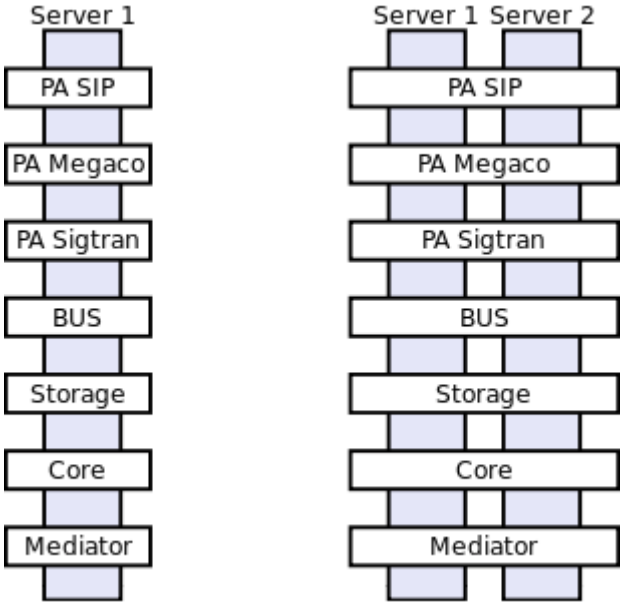


Рисунок 11 — Системы малой емкости

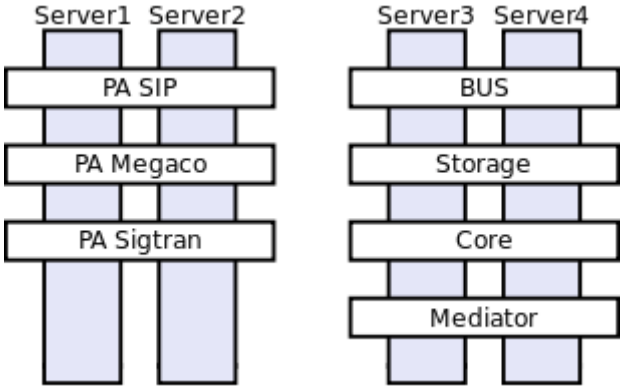


Рисунок 12 — Системы средней емкости

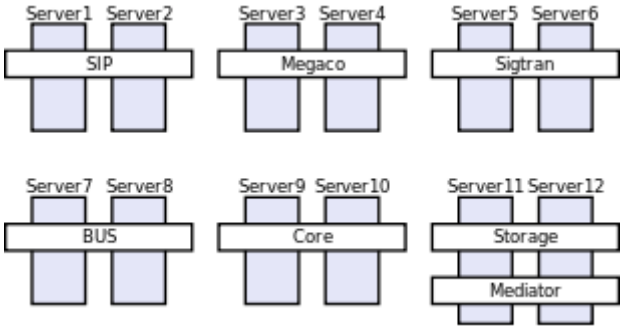


Рисунок 13 — Системы high-end класса

Программные компоненты

Главным элементом системы ECSS-10 является программное обеспечение (далее ПО). ПО состоит из кластеров, выполняющих различные функции. Каждый кластер включает в себя одну или несколько нод. Прикладное программное обеспечение является полностью переносимым, что позволяет использовать любую операционную систему, как семейства Unix/Linux, так и серверные ОС от Microsoft, а также использовать широкий спектр аппаратных архитектур, не ограничиваясь серверными платформами Intel/AMD, например, ОС Эльбрус/МЦСТ.

В настоящий момент в системе ECSS-10 используется операционная система с открытым исходным кодом Ubuntu Server. Переносимость, высокая надежность и эффективность ПО обеспечиваются:

- использованием при разработке ПО открытого специализированного функционального языка программирования Erlang и большого набора программных библиотек OTP (Open Telecom Platform);
- модульностью ПО;
- функциональной изолированностью модулей;
- согласованностью межмодульных интерфейсов;
- использованием модульного и системного тестирования как ПО, так и всей системы ECSS-10, встроенного в процесс разработки;
- высоким уровнем квалификации и инженерной культуры разработки.

В состав ПО системы ECSS-10 входят следующие типы кластеров, рисунок 14:

- *Storage* — обеспечивает хранение долговременных данных (конфигурации);
- *BUS* — интеграционная шина, обеспечивает надежную передачу сообщений между подсистемами;
- *Core* — осуществляет управление вызовом, маршрутизацию телефонных вызовов и управление предоставлением услуг, собирает тарификационные данные об обслуженных вызовах и взаимодействует с посредником СОРМ;
- *Adapter (Protocol adapter – PA)* — адаптирует определенный сигнальный протокол к внутреннему протоколу сигнализации ECSS-10;
- *Mediato*r — обеспечивает функции управления системой ECSS-10, предоставление статистической информации и аварийной сигнализации;

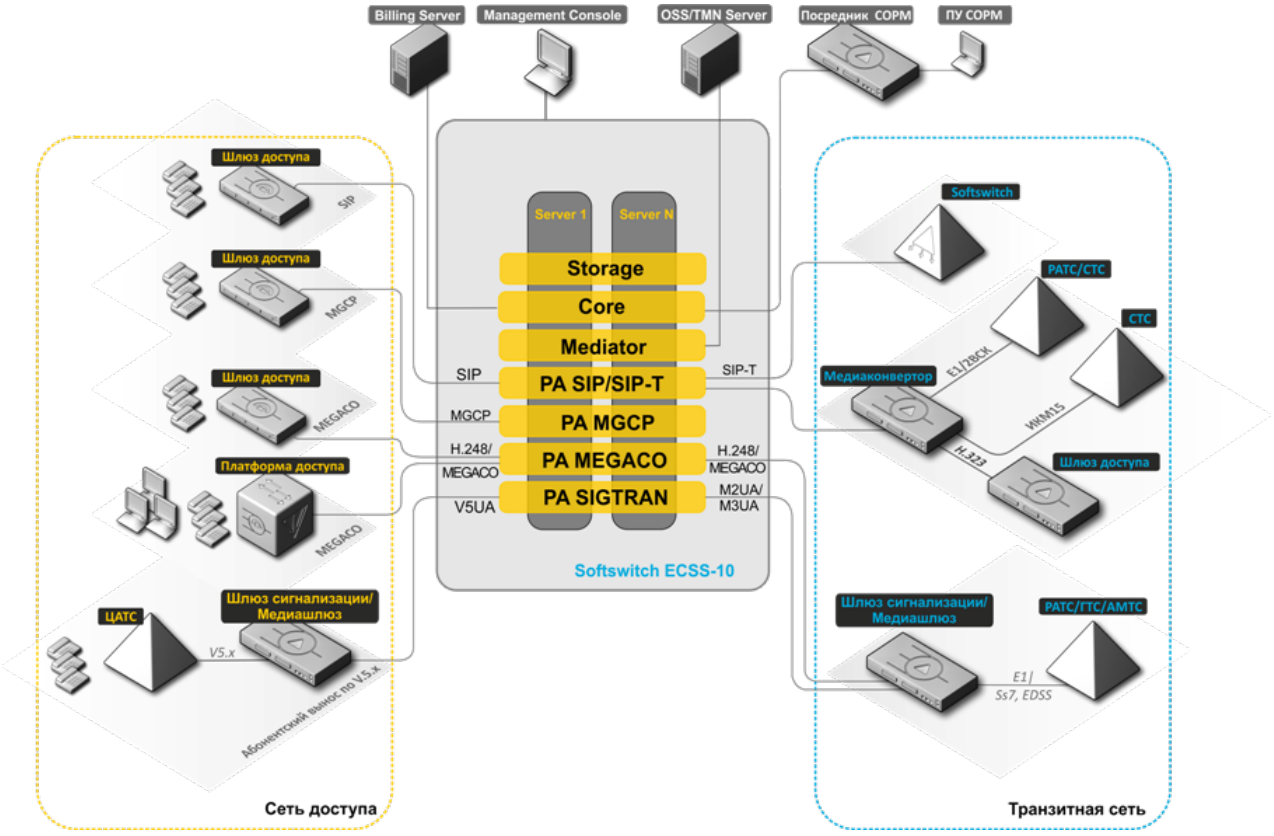


Рисунок 14 — Функциональный состав ПО системы ECSS-10

Кластер BUS

Одной из важных особенностей ECSS-10 является осуществление взаимодействия между программными компонентами комплекса через *интеграционную шину — BUS*.

Кластер BUS построен на базе сервера обмена сообщениями (брокера) собственной разработки Mycelium, который поддерживает функционал очередей сообщений и транзакционный механизм обмена. Взаимодействие с клиентами осуществляется по протоколу AMQP (Advanced Message Queuing Protocol) версии 0-10.

При отправке клиентом сообщения на брокер в качестве пункта назначения указывается адрес точки обмена (exchange) на брокере, который представляет собой строку определенной структуры. Точка обмена представляет собой механизм маршрутизации сообщений, который позволяет направить поступающие в точку обмена сообщения в одну или несколько очередей по правилам, регламентированным протоколом AMQP. Далее из точки обмена сообщение попадает в очередь, из которой отправляется клиентам, подписанным на сообщения очереди.
Транзакционность отправки, поддерживаемая брокером, позволяет гарантировать то, что сообщение будет отправлено на брокер, пройдет механизм маршрутизации и поступит в одну или несколько очередей, а значит, не будет потеряно. Если в процессе отправки сообщения происходит ошибка, то клиент об этом незамедлительно информируется.

Для получения сообщения от брокера клиент должен подписаться на определенную очередь. Когда на сообщения одной и той же очереди подписаны несколько клиентов, доставка их осуществляется в режиме равномерного распределения сообщений между клиентами (round-robin). Эта возможность позволяет реализовать в системе режим разделения нагрузки между нодами кластера.
Транзакционность доставки, поддерживаемая брокером, позволяет гарантировать то, что сообщение будет доставлено клиенту. При отказе клиента или ошибке доставки производится повторная доставка сообщения другому клиенту, подписанному на данную очередь.
Взаимодействие через интеграционную шину обеспечивает унификацию механизмов взаимодействия, слабую связанность между подсистемами и позволяет повысить надежность доставки.

Режим разделения нагрузки при доставке сообщения клиенту используется для реализации в ECSS-10 резервирования «active-active». При такой схеме резерва все компоненты находятся в активном состоянии и обслуживают поступающую нагрузку в режиме разделения нагрузки. Конфигурационная информация и операционные данные системы синхронизируются между зарезервированными элементами. Если из строя выйдет какой-либо программный компонент, то за счет механизмов BUS вся поступившая на программный компонент нагрузка (все сообщения) будет считаться не обслуженной, и произойдет её повторная доставка на однотипный резервирующий программный компонент (другую ноду того же кластера), который осуществит обработку данной нагрузки. В случае выхода из строя аппаратной части (хоста) произойдет полное переключение обработки текущей и вновь поступающей нагрузки на оставшиеся в работе компоненты системы (ноды находящиеся на другом хосте). При восстановлении работоспособности вышедшего из строя компонента происходит его регистрация в системе и подписка на требуемые для его работы потоки информации (очереди), что приводит к началу поступления на него сообщений — компонент встает в работу.

Кластер Storage

Кластер Storage выполняет функцию распределенного хранилища конфигурационных данных всей системы. Также в рамках этой подсистемы реализован модуль маршрутизации телефонных вызовов, обладающий высокой производительностью.
Storage использует для хранения конфигурационных данных документо-ориентированной базы данных Mnesia, которая является частью комплекта библиотек OTP (Open Telecom Platform), поставляемых вместе с Erlang. Кластер обеспечивает зеркалирование хранимой в БД информации между всеми нодами кластера. Зеркалирование обеспечивается транзакционными механизмами Mnesia — внесение изменений в данные считается выполненным, если подтверждение о применении этих изменений приходит со всех нод кластера.
В системе должен быть один кластер *Storage*.

Storage хранит в себе следующую информацию о конфигурации и текущем состоянии системы:

- топология кластеров системы ECSS-10, описывающая имена, роли и ноды кластеров;
- индивидуальные параметры каждого кластера;

- конфигурационная информация по виртуальным АТС, их абонентам и телефонной маршрутизации;
- конфигурационная информация по интерфейсам (описание транковых и абонентских портов, подключенных к системе);
- оперативная информация о состоянии интерфейсов;
- скрипты IVR.

В процессе работы системы кластер Storage взаимодействует со всеми остальными кластерами системы ECSS-10 для выдачи им конфигурационных и оперативных данных, а также сохранения изменений в этих данных.

Кластер Core

Кластер Core реализует логику управления обработкой телефонных вызовов (функции Call Control) и предоставления услуг. Алгоритм обслуживания вызовов реализован с использованием эталонных конечных автоматов O-BCSM и T-BCSM модели реализации интеллектуальных сетей связи (IN) с набором возможностей CS-3 согласно Рекомендации ITU-T Q.1238. Такой подход позволяет регулировать стандартный процесс обслуживания вызова и реализовывать различные услуги.
Реализация услуг выполнена в виде отдельных загружаемых модулей, что дает возможность расширять набор поддерживаемых сервисов и формировать различные пакеты дополнительных услуг для каждого оператора.
В правильно настроенной системе должен быть минимум один кластер Core (в высоконагруженных системах может быть несколько кластеров). Ноды кластера Core работают в режиме разделения нагрузки, обслуживая поступающие вызовы. Данные об обслуживаемых в каждый момент вызовах синхронизируются между нодами кластера, что позволяет незаметно для абонента перевести обслуживание вызова с одной ноды кластера на другую. Это может потребоваться в ситуациях отказа ноды в обслуживании, выходе из строя хоста или при выводе хоста или ноды из работы для проведения работ по их обслуживанию.

Сервис TTS реализует функционал сбора данных о вызовах и формирование файлов с записями о разговорах CDR (Call Detail Record). TTS поддерживает различные режимы сохранения CDR с возможностью адаптации информации в CDR и формата файла под определенного заказчика. Сервис TTS обеспечивает кластерное сохранение CDR, когда файлы с данными о вызовах записываются одновременно на всех нодах кластера.

В процессе работы системы кластер Core взаимодействует со всеми остальными кластерами системы ECSS-10 следующим образом:

- BUS — посредством интеграционной шины передаются сообщения для других подсистем;
- Storage — запросы на получение конфигурационных данных, отправка запросов на выполнение телефонной маршрутизации;
- Adapter — обмен сообщения по внутрисистемному сигнальному протоколу (ACP), обеспечивающему обработку вызова и услуг;
- Mediator — передача информации об обслуженных вызовах для формирования статистики, отправка уведомления об обнаруженных авариях.

Кластер Adapter

Кластер Adapter осуществляет адаптацию сетевого сигнального протокола, по которому подключаются внешние по отношению к гибкому коммутатору системы и оборудование, к внутреннему сигнальному протоколу системы (ACP).
В настоящее время реализованы следующие адаптеры протоколов:

- *PA Megaco* — кластер взаимодействия со шлюзами, работающими по протоколу H.248/Megaco;
- *PA SIP/SIP-T/SIP-I* — кластер взаимодействия со шлюзами, периферийным оборудованием и другими гибкими коммутаторами по протоколам SIP и SIP-T/SIP-I;
- *PA Sigtran* — кластер взаимодействия со шлюзами, работающими по стеку протоколов Sigtran (M2UA, M3UA, IUA).

Кластер Adapter состоит из одной или нескольких нод, работающих в режиме разделения нагрузки и синхронизирующих оперативные данные процесса обслуживания вызовов между собой. Такая структура позволяет обрабатывать ситуации принудительного или произвольного отказа в обслуживании ноды с автоматическим переводом обслуживания вызовов на другие ноды кластера.
Также Adapter отрабатывает ситуации отказа сети передачи данных (выход из строя сетевого адаптера, коммутатора, патч-корда и т.п.), приводящие к невозможности обмена пакетами.
В Adapter'е реализована виртуализация сетевого адреса, по которому осуществляется обмен сигнальными пакетами целевого сигнального протокола. Использование виртуализации позволяет резервировать IP-адрес системы ECSS-10, который используется при обмене с шлюзами и абонентским оборудованием. Для виртуализации сетевого адреса используется реализация протокола VRRP, выполненная в сервисе keepalived ОС Linux.
В правильно настроенной системе должно быть как минимум по одному кластеру Adapter'а для каждого протокола, используемого при подключении шлюзов и абонентов к системе.

В процессе работы системы кластер Adapter взаимодействует с другими кластерами системы ECSS-10 следующим образом:

- BUS — используется для обмена сообщениями с другими подсистемами;
- Storage — запросы на получение конфигурационных данных, обновление оперативных данных о состояниях интерфейсов;
- Core — обмен сигнальными сообщениями протокола ACP в ходе обслуживания вызовов и предоставления услуг;
- Mediator — передача информации о попытках занятия и релизах для формирования статистики, а также информация об обнаруженных авариях.

Кластер Mediator

Кластер Mediator реализует:

- функции накопления статистической информации об обслуженной нагрузке, которая отправляется PA, и формирования периодических статистических отчетов согласно требованиям Рекомендаций ITU-T E.502 и Q.752;
- сбор информации об авариях, обнаруженных в системе, и выдачу этой информации по протоколу SNMP на внешние системы мониторинга и управления сетью;

- интерфейс взаимодействия на web-сервисах, обеспечивающий возможность подключения таких внешних, по отношения к системе ECSS-10, как web-конфигуратор SSW, биллинговые системы, "Портал абонента", OSS/BSS.

Как и другие подсистемы ECSS-10 кластер Mediator обеспечивает резервирование своих данных (данные о статистике и обнаруженных авариях). Резервирование данных осуществляется посредством механизмов распределённых транзакций БД Mnesia, в которой они хранятся. Поэтому отказ в обслуживании ноды или хоста, на котором запущена нода, не приводит к их потере.
В правильно настроенной системе должен быть один кластер Mediator.

В процессе работы системы кластер Mediator взаимодействует с другими кластерами ECSS-10 следующим образом:

- BUS — используется для обмена сообщениями с другими подсистемами;
- Storage — запросы на получение конфигурационных данных;
- Core — получение данных об обслуженных вызовах для формирования статистики, получение уведомлений об авариях;
- Adapter — получение данных о попытках занятия и разъединениях для формирования статистики, получение уведомлений об авариях.

Процесс обслуживания вызовов. Внутренний протокол сигнализации

В процессе обслуживания вызова участвуют все компоненты системы. Основными компонентами являются Adapter и Core, которые непосредственно работают с сигнализацией и отслеживают фазы вызова.
Обмен сообщениями осуществляется через интеграционную шину, которую обеспечивает кластер BUS. Кластера Adapter и Core обмениваются сообщениями внутреннего протокола ACP (Adapter Core Protocol). Этот протокол является модификацией протокола DSS и основан на примитивах обмена, описанных в стандарте ITU-T Q.1238 для интеллектуальных сетей.

В протоколе ACP выделяют следующие типы сообщений:

- Setup — сообщение для установления соединения (запрос, подтверждение, ответ);
- SubsequentAddress — сообщение с информацией о дополнительных цифрах;
- AddressEnd — сообщение с информацией о завершении набора номера и набранными цифрами;
- ServiceFeature — сообщение с индикацией об активации сервисной логики;
- CallProgress — сообщение для передачи информации без смены фазы обслуживания вызова
- Release — сообщения о разъединении вызова.

Для этих сообщений применяются модификаторы, которые описывают фазу передачи сообщения.

Различают следующие модификаторы:

- Ind — индикация события;
- Req — запрос;
- Resp — ответ;
- Conf — подтверждение;
- Ack — подтверждение в приеме сообщения.

Полный набор сообщений, которые присутствуют в протоколе ACP с учетом модификаторов, имеет вид:

- SetupInd — сообщение передается, когда адаптер определил факт входящего занятия;
- SetupIndAck — сообщение передается на адаптер как подтверждение начала обработки SetupInd;
- SetupReq — сообщение передается на адаптер, когда ядро инициирует исходящее занятие;
- SetupReqAck — сообщение передается на ядро как подтверждение начала обработки SetupReq
- SetupConf — сообщение передается на ядро, когда адаптер получил индикацию об ответе (подтверждение об установлении соединения);
- SetupResp — сообщение передается на адаптер, когда ядро подтверждает установление соединения (ответ);
- SubsequentAddressInd — сообщение передается на ядро, когда адаптер получил донабор от абонента (ситуация dig-by-dig набора);
- AddressEndInd — сообщение передается, когда определено завершение набора номера;
- CallProgressInd — сообщение передается, когда адаптер информирует ядро о событии;

- CallProgressReq — сообщение передается, когда ядро отправляет на адаптер запрос на передачу сообщения;
- ServiceFeatureInd — сообщение передается на ядро, когда адаптер определил событие активации сервисной логики;
- ReleaseInd — сообщение передается на ядро, когда адаптер определил факт разъединения;
- ReleaseReq — сообщение передается на адаптер, когда ядро отправляет запрос на разъединение.

Основные параметры сигнальных сообщений:

SetupInd/Req

- CallRef — идентификатор вызова;
- CallingPartyNumber — номер вызывающего абонента (номер А);
- CalledPartyNumber — номер вызываемого абонента (номер Б);
- CallingPartysCategory — категория вызывающего абонента;
- CalledPartysCategory — категория вызываемого абонента;
- CallingPartyInfo — параметры вызывающего абонента (домен, интерфейс, информация о подключенных услугах);
- CalledPartyInfo — параметры вызываемого абонента (домен, интерфейс, информация о подключенных услугах);
- TrunkGroupId — идентификатор входящей транковой группы для вызова, пришедшего с транка;
- sdp — информация о медиапотоке.

SetupResp/Conf

- CallRef — идентификатор вызова;
- ConnectedNumber — номер вызываемого абонента, ответившего на вызов (если он отличается от изначально набранного номера);
- sdp — информация о медиапотоке.

SubsequentAddressInd

- CallRef — идентификатор вызова;
- Digits — цифры номера.

AddressEndInd

- CallRef — идентификатор вызова;
- Digits — цифры номера.

CallProgressInd/Req

- CallRef — идентификатор вызова;
- Cause — внутренний индикатор причины отправки сообщения;
- CauseInitiator — инициатор события (система, пользователь или событие обнаружено на сети);
- CauseIsup — индикатор причины в формате ISUP;
- EventInformation — описание события;
- gNotification — информационные флаги;
- sdp — информация о медиапотоке;
- TrunkGroupId — идентификатор исходящей транковой группы для исходящего вызова.

ReleaseInd/Req

- CallRef — идентификатор вызова;
- Cause — внутренний индикатор причины разъединения;
- CauseInitiator — инициатор события (система или события обнаружено на сети);
- CauseIsup — индикатор причины разъединения в формате ISUP;
- TrunkGroupId — идентификатор исходящей транковой группы для исходящего вызова.

Далее будут рассмотрены процессы обслуживания вызова на sequence-диаграммах, из которых будет виден процесс обмена сообщениями.

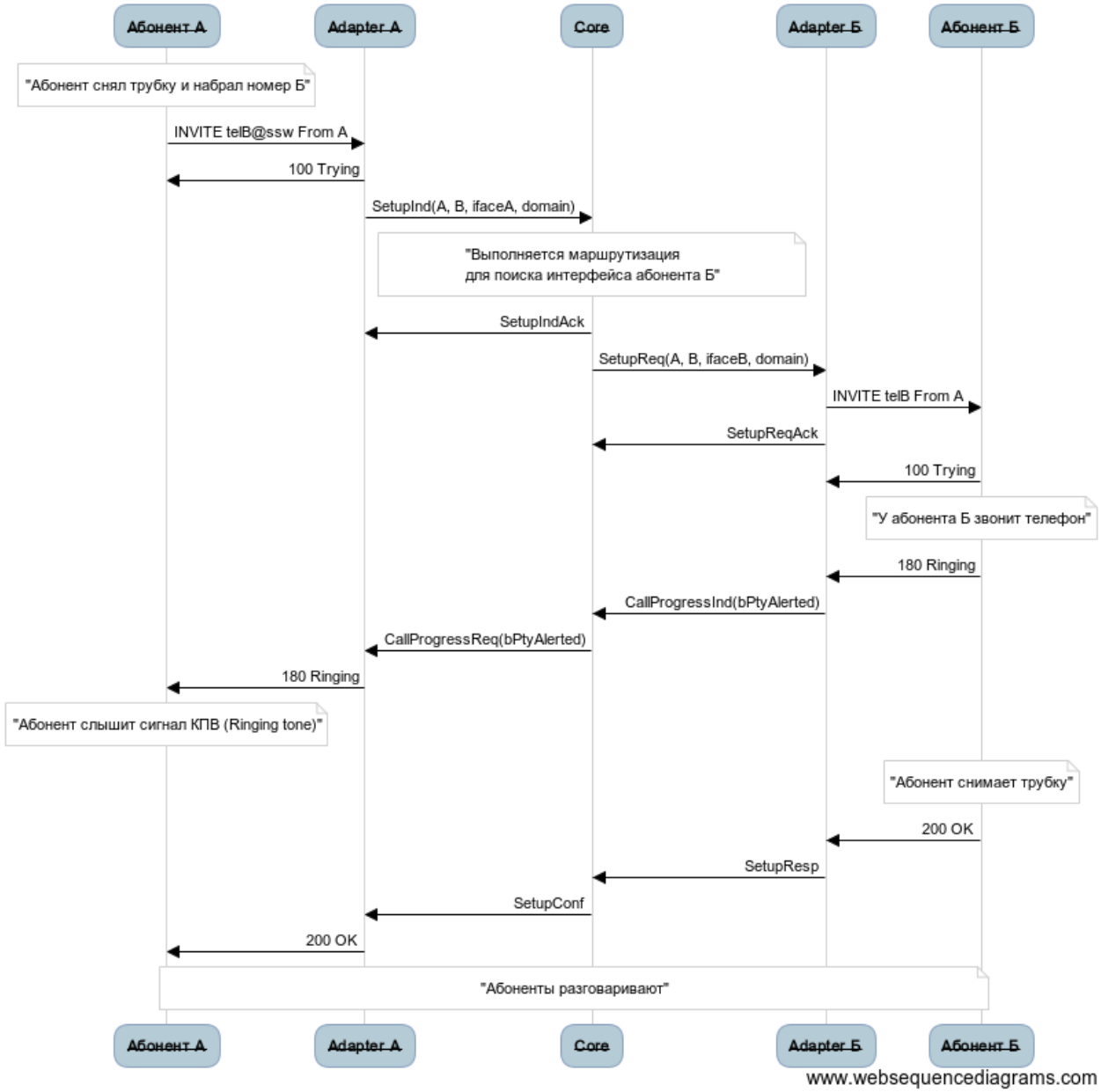


Рисунок 15 — Базовый вызов по протоколу SIP в режиме "enblock", упрощенный вид

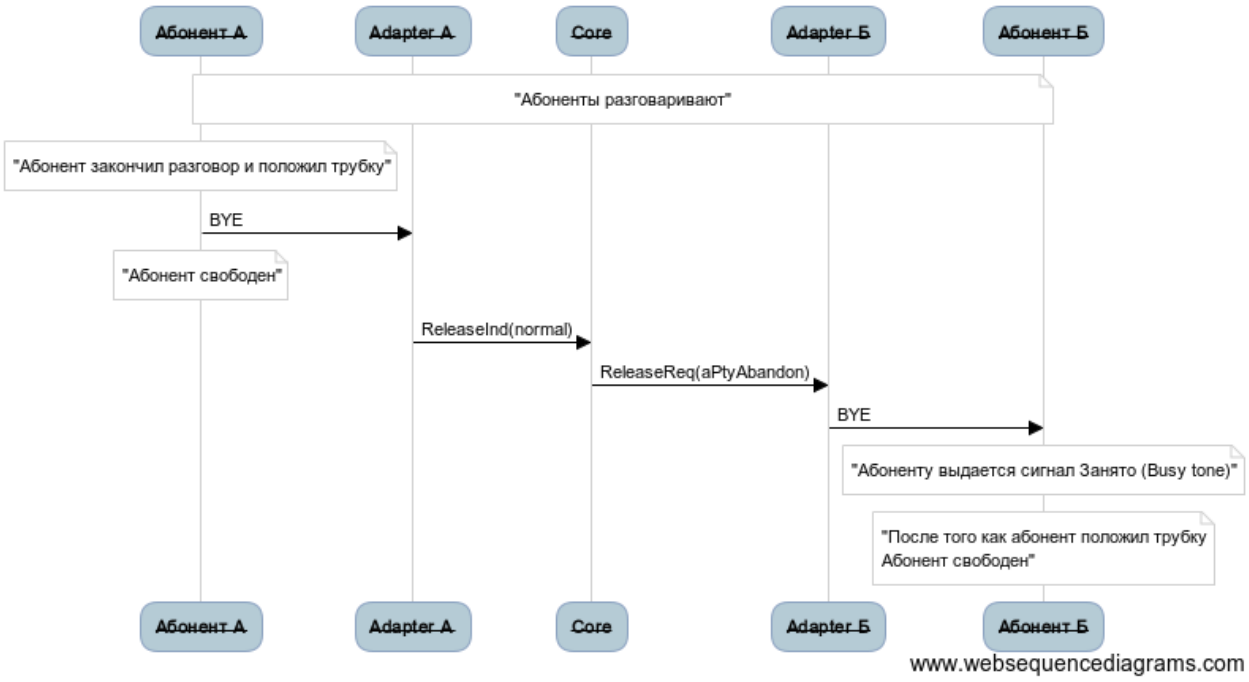


Рисунок 16 — Разъединение вызова по инициативе абонента А, упрощенный вид

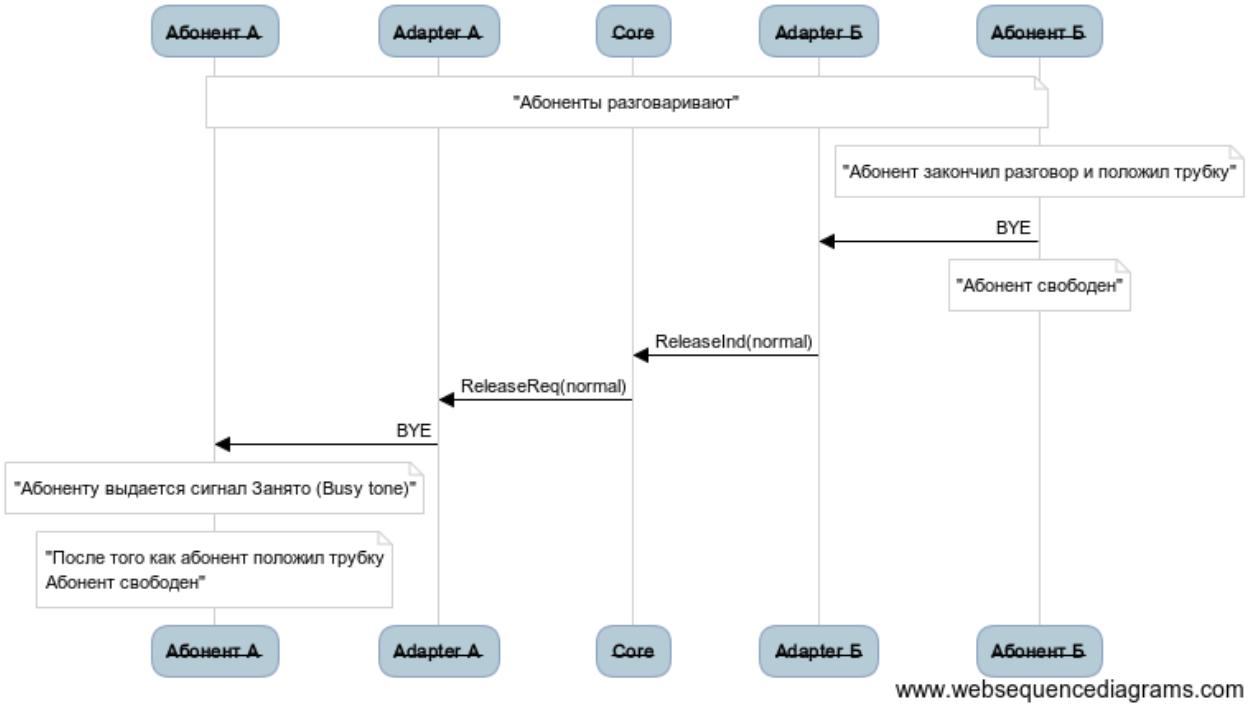


Рисунок 17 — Разъединение вызова по инициативе абонента Б, упрощенный вид

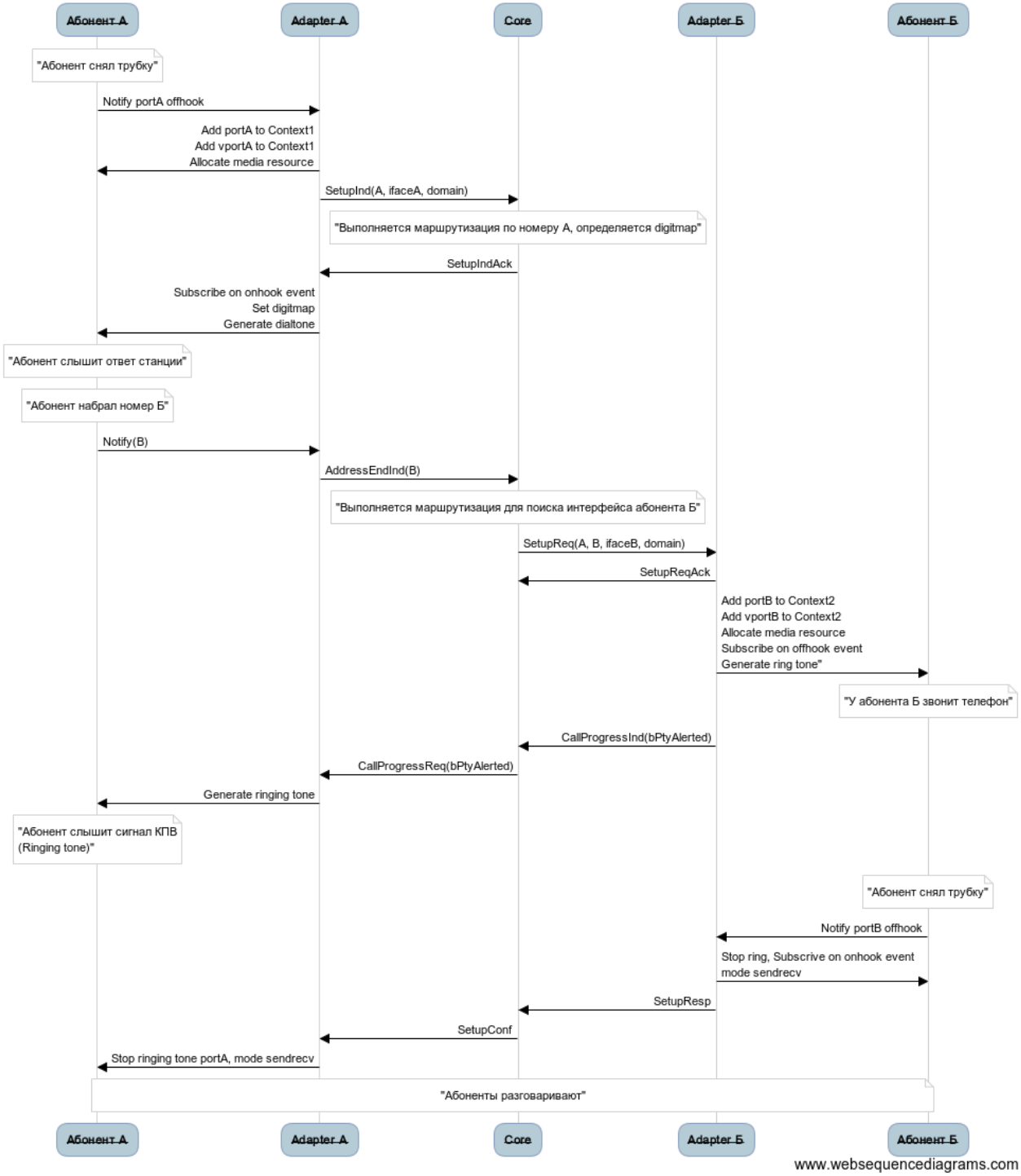


Рисунок 18 — Базовый вызов по протоколу H.248/Megaco в режиме "enblock", упрощенный вид

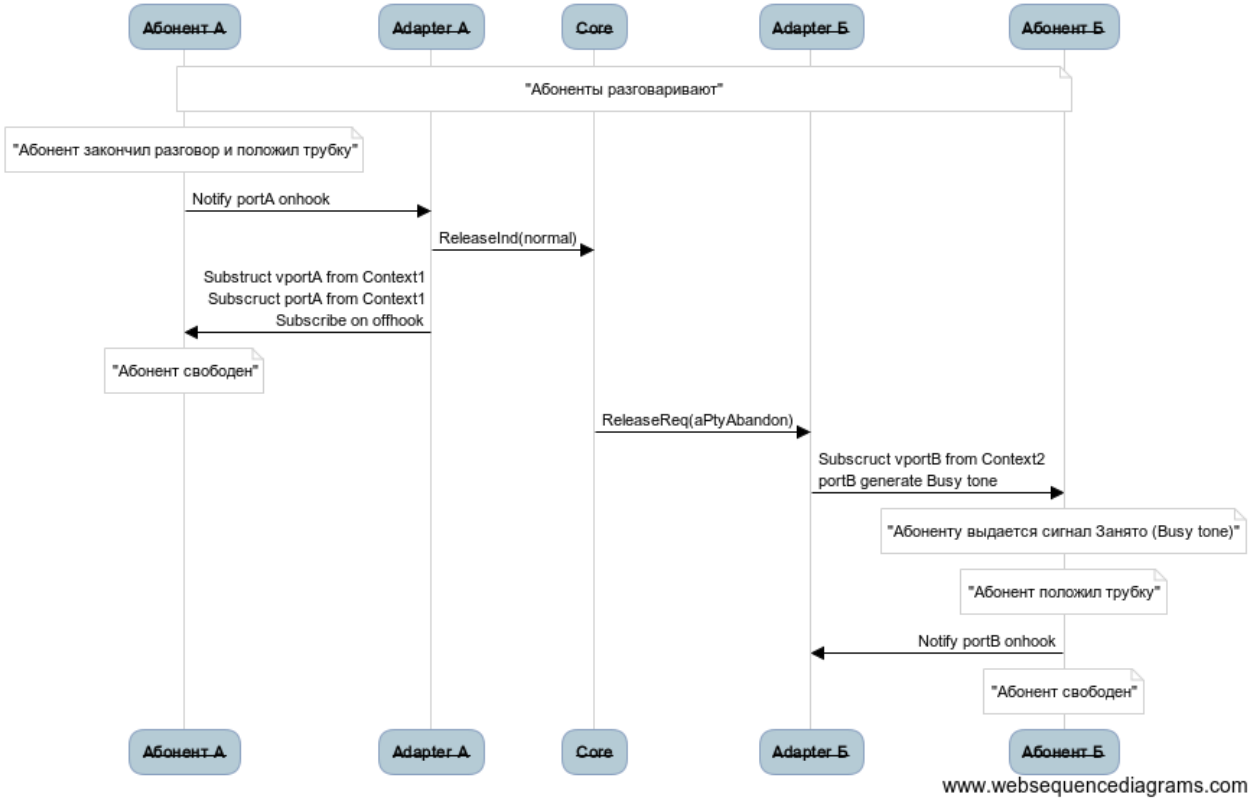


Рисунок 19 — Разъединение вызова по инициативе абонента А по протоколу H.248/Megaco, упрощенный вид

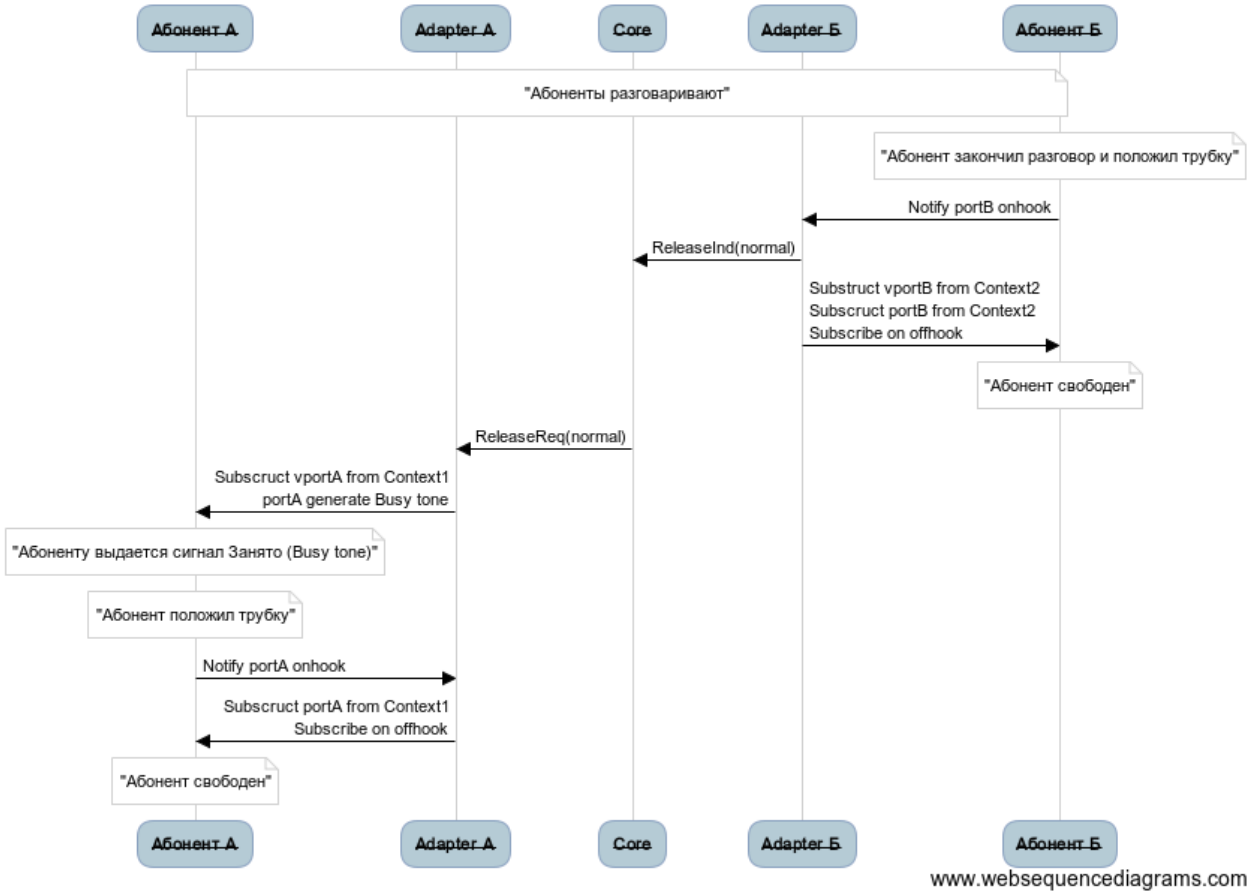


Рисунок 20 — Разъединение вызова по инициативе абонента Б по протоколу H.248/Megaco, упрощенный вид

Примеры неуспешных вызовов

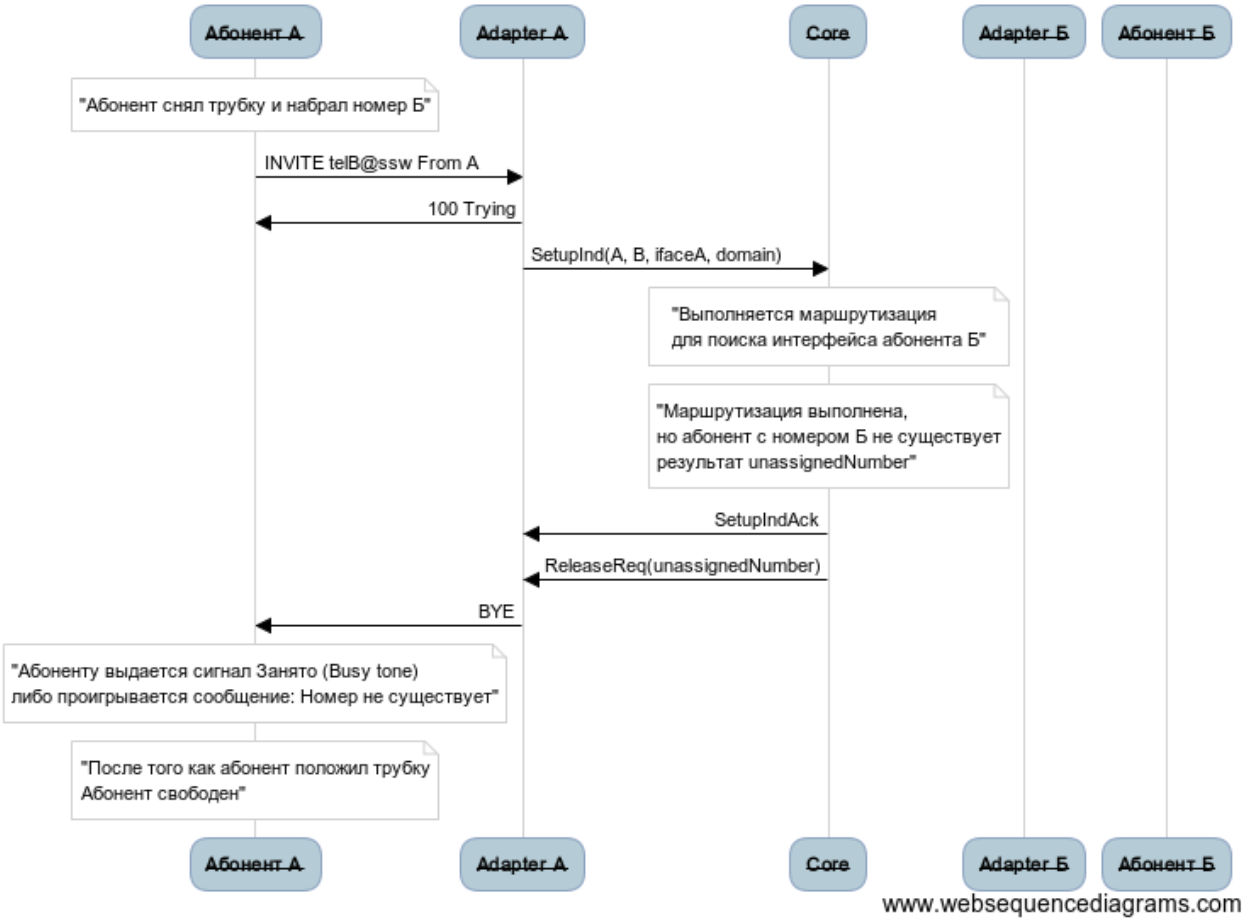


Рисунок 21 — Неуспешный вызов по причине того, что номер Б не существует

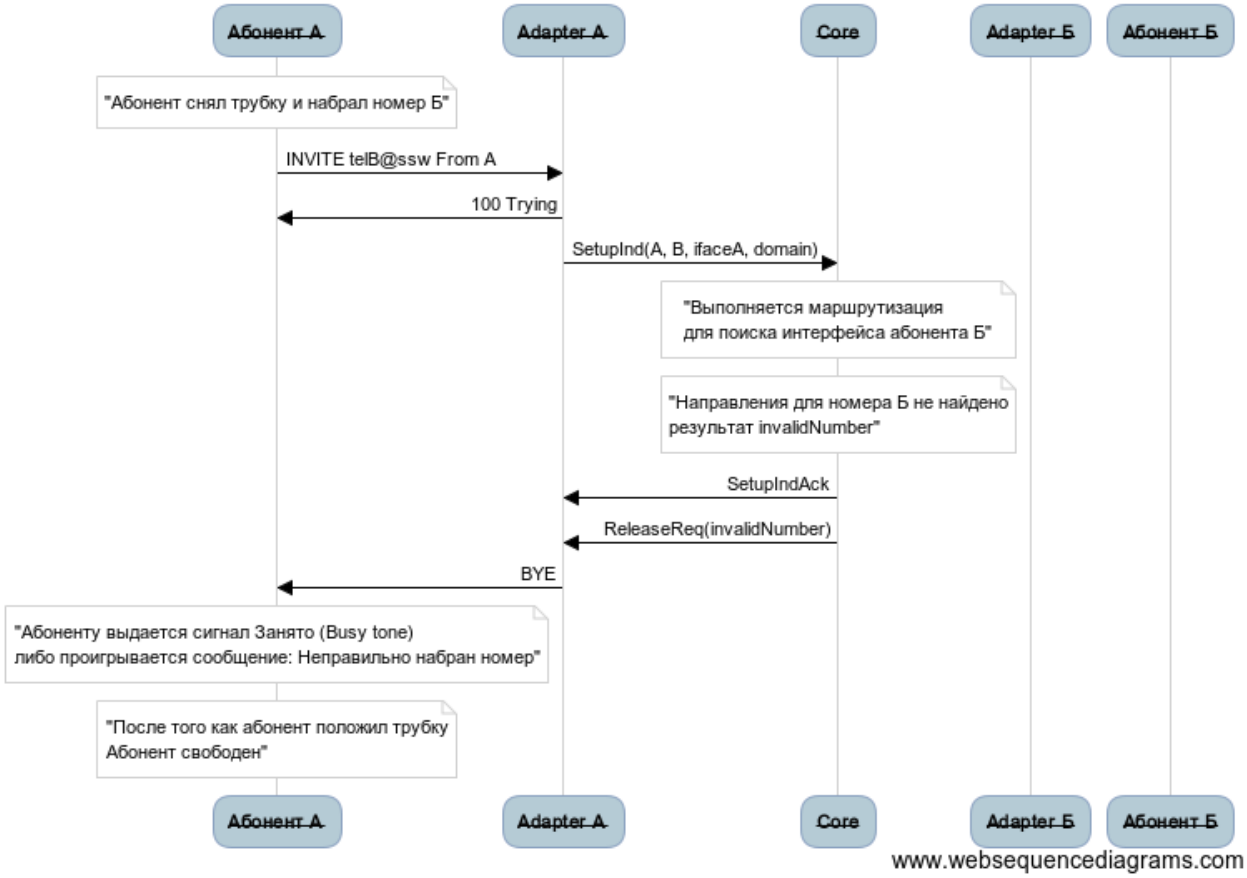


Рисунок 22 — Неуспешный вызов по причине того, что номер Б не существует

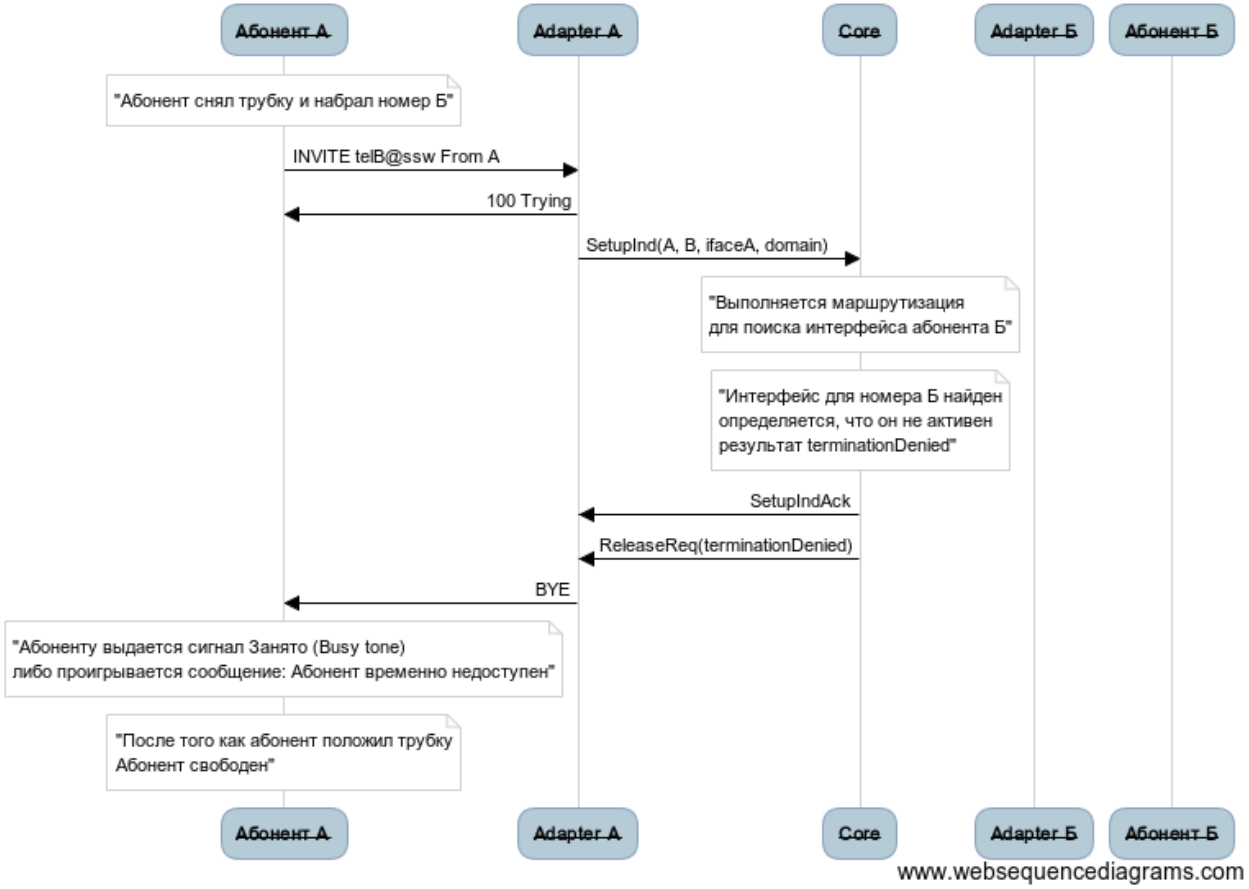


Рисунок 23 — Неуспешный вызов по причине того, что интерфейс абонента Б не активен

Процесс обслуживания вызовов. Схема обмена данными

В данном разделе описан общий порядок обмена информацией в процессе обслуживания вызова.

Adapter A

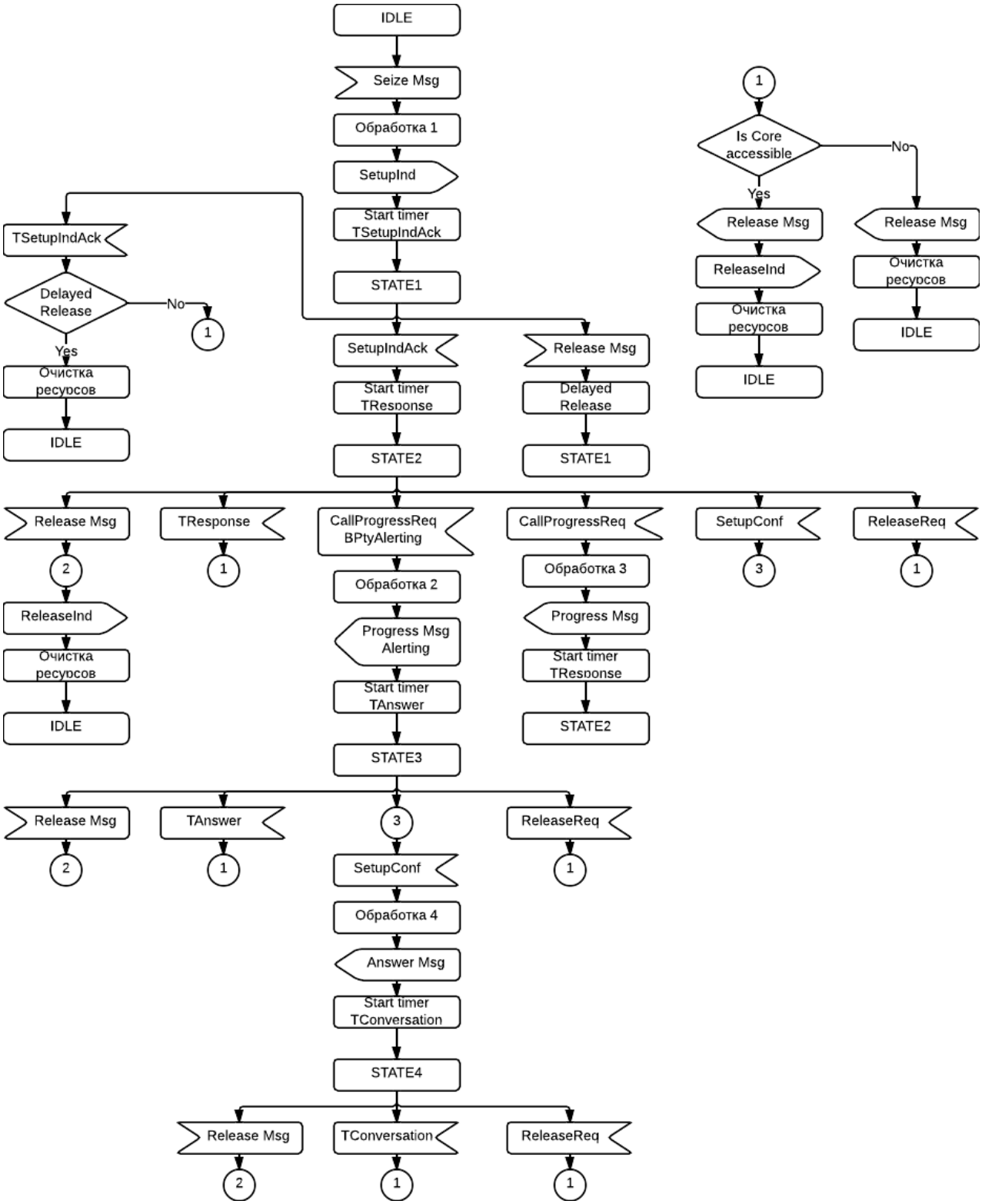

IDLE
Seize Msg
Обработка 1
SetupInd
Start timer TSetupIndAck
STATE1
TSetupIndAck
Delayed Release
No
1
Yes
Очистка ресурсов
IDLE
SetupIndAck
Start timer TResponse
STATE2
Release Msg
Delayed Release
STATE1
1
Is Core accessible
No
Yes
Release Msg
ReleaseInd
Очистка ресурсов
IDLE
Release Msg
Очистка ресурсов
IDLE
Release Msg
2
ReleaseInd
Очистка ресурсов
IDLE
TResponse
1
CallProgressReq BPtyAlerting
Обработка 2
Progress Msg Alerting
Start timer TAnswer
STATE3
CallProgressReq
Обработка 3
Progress Msg
Start timer TResponse
STATE2
SetupConf
3
ReleaseReq
1
Release Msg
2
TAnswer
1
3
SetupConf
Обработка 4
Answer Msg
Start timer TConversation
STATE4
ReleaseReq
1
Release Msg
2
TConversation
1
ReleaseReq
1

Рисунок 24 — Диаграмма работы originating кластера Adapter (Adapter A)

Описание действий, выполняемых в функциональных блоках на диаграмме работы originating кластера Adapter (Adapter A):

Обработка 1 — входящий вызов по протоколу сигнализации поступает в ноду originating кластера Adapter (Adapter A), в которой:

- производится декодирование информационного пакета декодером соответствующего протокола;
- осуществляется проверка корректности и целостности полученного сигнального сообщения;
- анализируются параметры полученного сигнального сообщения и определяются идентификаторы отправителя и получателя;
- определяется, должен ли сигнальный пакет обрабатываться системой;
- на основании параметров сообщения определяется, к какому originating интерфейсу (интерфейс A) оно относится;
- выполняется запрос в кластер Storage о конфигурационных данных для интерфейса A;
- формируется и подготавливается к отправке в кластер Core сообщение "SetupInd" в кластер Core;
- процесс обслуживания вызова переводится в логическое состояние ожидания подтверждения начала обработки от кластера Core (State1);
- формируется и отправляется в кластер Mediator пакет статистической информации о поступлении занятия по интерфейсу A.

Обработка 2 — от ядра получено сообщение о выдаче абоненту Б сигнала "Посылка вызова", выполняются следующие действия:

- анализ корректности и целостности полученного сообщения;
- процесс обслуживания вызова переводится в логическое состояние Alerting (State3);
- срабатывают триггерные точки для состояния Alerting, приводящие к выполнению логики активированных сервисов;
- формируется и подготавливается к отправке абоненту А сообщение "Progress" с индикатором "Alerting" целевого протокола адаптера;
- обновляется информация, накапливаемая модулем СОРМ и модулем тарификации.

Обработка 3 — от ядра получено промежуточное сообщение о статусе процесса установления соединения, выполняются следующие действия:

- анализ корректности и целостности полученного сообщения;
- процесса обслуживания вызова остается в том же логическом состоянии ожидания ответного сообщения от абонента Б (State2);
- формируется и подготавливается к отправке абоненту А сообщение "Progress" с индикаторами, построенными на базе исходного сообщения по правилам целевого протокола адаптера.

Обработка 4 — от ядра получено сообщение об ответе абонента Б (SetupConf), выполняются следующие действия:

- анализ корректности и целостности полученного сообщения;
- процесс обслуживания вызова переводится в логическое состояние "Conversation" (State4);

- срабатывают триггерные точки для состояния "Converstaion", приводящие к выполнению логики активированных сервисов;
- формируется и подготавливается к отправке абоненту А сообщение "Answer" целевого протокола адаптера;
- обновляется информация, накапливаемая модулем СОРМ и модулем тарификации.

Core

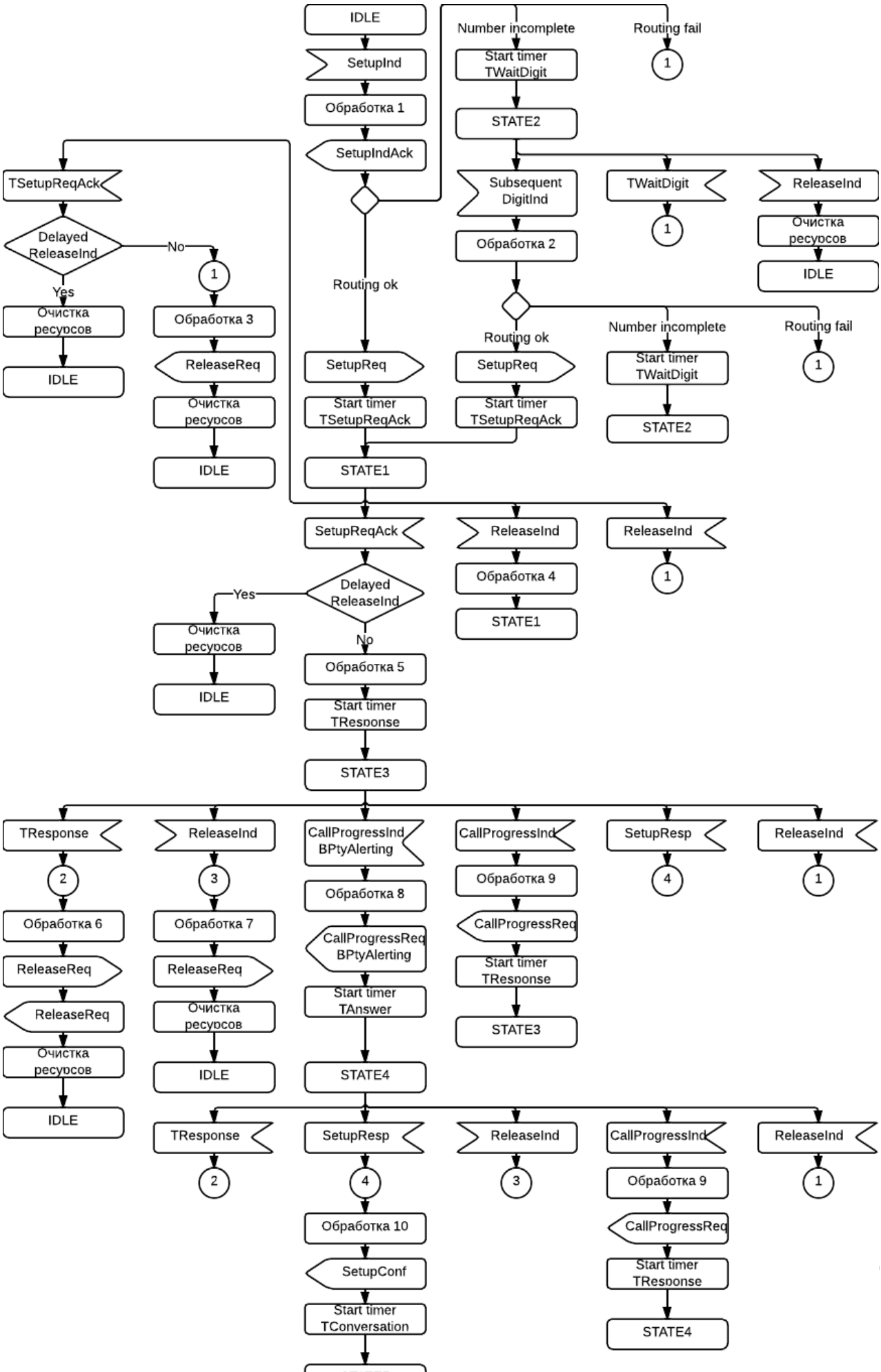

IDLE
SetupInd
Обработка 1
SetupIndAck
Routing ok
Number incomplete
Routing fail
Start timer TWaitDigit
1
STATE2
Subsequent DigitInd
TWaitDigit
ReleaseInd
Обработка 2
Очистка ресурсов
IDLE
Routing ok
Number incomplete
Routing fail
SetupReq
Start timer TSetupReqAck
Start timer TWaitDigit
STATE2
STATE1
TSetupReqAck
Delayed ReleaseInd
No
Yes
Очистка ресурсов
IDLE
Обработка 3
ReleaseReq
SetupReqAck
ReleaseInd
Обработка 4
Delayed ReleaseInd
Yes
No
Обработка 5
Start timer TResponse
STATE3
TResponse
ReleaseInd
CallProgressInd BPtyAlerting
CallProgressInd
SetupResp
2
3
4
Обработка 6
Обработка 7
Обработка 8
Обработка 9
CallProgressReq BPtyAlerting
CallProgressReq
Start timer TAnswer
Start timer TResponse
STATE4
STATE3
TResponse
SetupResp
ReleaseInd
CallProgressInd
2
4
3
Обработка 10
SetupConf
Start timer TConversation
STATE4

Рисунок 25 — Диаграмма работы кластера Core

Описание действий, выполняемых в функциональных блоках на диаграмме работы кластера Core:

Обработка 1

В кластер Core поступает сообщение "SetupInd", информирующее о начале обработки нового вызова. Выполняются следующие операции:

- анализируются параметры сообщения "SetupInd", проверяется их полнота и целостность;
- извлекается и сохраняется информация, необходимая для выполнения функций СОРМ и тарификации;
- подготавливается к отправке в Adapter A сообщение "SetupIndAck", подтверждающее начало обработки вызова кластером Core;
- выполняется запрос в кластер Storage на выполнение телефонной маршрутизации, целью которой является поиск "terminating" интерфейса (интерфейса Б), а также применение различного вида модификаций номеров А и Б;
- анализируются результаты маршрутизации, и в зависимости от результатов маршрутизации различается поведение.

В результате маршрутизации был успешно найден интерфейс локального абонента Б:

- в случае успешного нахождения маршрута с кластера Storage будут получены:
  - номер абонента А (возможно модифицированный);
  - абонентские параметры абонента А;
  - идентификатор интерфейса Б и его параметры;
  - номер абонента Б (возможно модифицированный);
  - абонентские параметры абонента Б.
- запускаются функции обработки услуг, активированных у абонента А и абонента Б;
- дополняются данные, собираемые модулем СОРМ и модулем тарификации;
- формируется и подготавливается к отправке в "terminating" кластер Adapter (Adapter Б) сообщение "SetupReq" для абонентского интерфейса Б.

В результате маршрутизации был успешно найден интерфейс/интерфейсы исходящего транка абонента Б:

- в случае успешного нахождения маршрута с кластера Storage будут получены:
  - номер абонента А (возможно модифицированный);
  - абонентские параметры абонента А;
  - идентификатор интерфейса Б и его параметры;
  - номер абонента Б (возможно модифицированный);
  - абонентские параметры транкового интерфейса абонента Б.
- запускаются функции обработки услуг, активированных у абонента А;
- дополняются данные, собираемые модулем СОРМ и модулем тарификации;
- формируется и подготавливается к отправке в "terminating" кластер Adapter (Adapter Б) сообщение "SetupReq" для транкового интерфейса Б.

В результате маршрутизации был определен неполный номер:

⚠ Неполный номер может быть определен только, если Adapter A является адаптером протокола H.248/Megaco. В рамках протокола H.248/Megaco поддерживается передача номера в режиме overlap. В случае использования протокола SIP номер, как правило, передается методом enblock, а метод overlap обычно не используется.

- в случае неполного номера в результате маршрутизации с кластера Storage будут получены:
  - код ошибки "неполный номер";
- процесс обработки вызова, переход в фазу ожидания последующих цифр номера (STATE2);
- запуск таймера ожидания следующей цифры номера.

В результате маршрутизации была возвращена ошибка:
Обслуживание вызова продолжается по процедуре "Обработка 3".

Обработка 2

Получили одну или несколько цифр номера в случае посимвольного набора, выполняется следующая обработка:

- выполняется запрос в кластер Storage для запуска процесса маршрутизации с использованием всех накопленных цифр номера Б;
- анализируются результаты маршрутизации, и в зависимости от результатов маршрутизации различается поведение.

Дальнейшие действия по обработке вызова аналогичны действиям из процедуры "Обработка 1" после получения результатов маршрутизации.

Обработка 3

Производятся действия по обработке ошибки, которая была выявлена в процессе обслуживания вызова:

В результате маршрутизации был определен неактивный интерфейс абонента Б:

Неактивный интерфейс абонента Б — это когда по номеру Б была найдена соответствующая ему локальная абонентская запись либо транковый интерфейс, но оперативные данные показывают, что интерфейс, соответствующий этому абоненту, не активен (абонент не был зарегистрирован либо регистрация истекла, для транка не прошла регистрация либо не отработал механизм OPTIONS-контроля).

- в случае неактивного интерфейса в результате маршрутизации с кластера Storage будут получены: ** код ошибки "неактивный интерфейс";
- если в системе активирован механизм CFC (Call Forwarding by Cause), который обеспечивает фразы автоинформатора для определенных типов ошибок, то он запускается на выполнение по логике для ошибки "неактивный интерфейс";
- после проигрывания фразы на Adapter A подготавливается к отправке запрос на разъединение "ReleaseReq" с соответствующей причиной;
- обновляется информация для модулей СОРМ и тарификации, итоговый информационный пакет отправляется в кластер CORE.

В результате маршрутизации было определено, что у абонента А закрыт доступ к номеру абонента Б:

Запрет доступа абонента А к номеру абонента Б может быть введен на уровне групп доступа или режима обслуживания, и служит для ограничения видов доступа в зависимости от соглашении об уровне сервиса между абонентом и оператором (например использование междугородней связи запрещено или действует временное ограничение исходящей связи по причине неоплаты услуг связи).

- в случае запрета доступа на номер Б в результате маршрутизации с кластера Storage будут получены: ** код ошибки "запрет доступа";
- если в системе активирован механизм CFC (Call Forwarding by Cause), который обеспечивает фразы автоинформатора для определенных типов ошибок, то он запускается на выполнение по логике для ошибки "запрет доступа";
- после проигрывания фразы на Adapter A подготавливается к отправке запрос на разъединение "ReleaseReq" с соответствующей причиной;
- обновляется информация для модулей COPM и тарификации, итоговый информационный пакет отправляется в кластер CORE.

В результате маршрутизации был определен несуществующий номер:

> ***Несуществующий номер*** *— это когда формат номера соответствует формату, заданному планом нумерации, номер относится к локальным абонентским номерам, но абонента с таким номером не создано.*

- в случае несуществующего номера в результате маршрутизации с кластера Storage будут получены: ** код ошибки "номер не существует";
- если в системе активирован механизм CFC (Call Forwarding by Cause), который обеспечивает фразы автоинформатора для определенных типов ошибок, то он запускается на выполнение по логике для ошибки "номер не существует";
- после проигрывания фразы на Adapter A подготавливается к отправке запрос на разъединение "ReleaseReq" с соответствующей причиной;
- обновляется информация для модулей COPM и тарификации, итоговый информационный пакет отправляется в кластер CORE.

В результате маршрутизации был определен неправильный номер:

> ***Неправильный номер*** *— это когда формат номера не соответствует форматам, заданным планом нумерации.*

- в случае неправильного номера в результате маршрутизации с кластера Storage будут получены: ** код ошибки "неправильный номер";
- если в системе активирован механизм CFC (Call Forwarding by Cause), который обеспечивает фразы автоинформатора для определенных типов ошибок, то он запускается на выполнение по логике для ошибки "неправильный номер";
- после проигрывания фразы на Adapter A подготавливается к отправке запрос на разъединение ReleaseReq с соответствующей причиной;
- обновляется информация для модулей COPM и тарификации, итоговый информационный пакет отправляется в кластер CORE.

Другие ошибки:

- анализируется код ошибки обслуживания вызова и преобразуется в код ошибки целевого протокола Adapter A;
- если в системе активирован механизм CFC (Call Forwarding by Cause), который обеспечивает фразы автоинформатора для определенных типов ошибок, то он запускается на выполнение по логике для ошибки "неправильный номер";

- после проигрывания фразы на Adapter A подготавливается к отправке запрос на разъединение ReleaseReq с соответствующей причиной;
- обновляется информация для модулей COPM и тарификации, итоговый информационный пакет отправляется в кластер CORE.

Обработка 4

Процедура выполняется при получении сообщения о разъединении "ReleaseInd" из кластера Adapter A до получения из кластера Adapter Б:

- анализируются параметры сообщения "ReleaseInd", проверяется их полнота и целостность;
- сообщение запоминается во внутреннем буфере процесса обработки вызова (Delayed ReleaseId);
- извлекается и сохраняется информация, необходимая модулям COPM и тарификации;
- процесс обработки вызова остается в состоянии ожидания подтверждения обработки от кластера Adapter Б (STATE1).

Обработка 5

Кластер Adapter Б подтвердил обработку вызова, получено сообщение "SetupReqInd", отложенных "ReleaseInd" в буфере процесса обработки вызова нет, идет нормальная обработка вызова:

- транспортные параметры процесса обслуживания вызова в кластере Adapter Б извлекаются из сообщения и сохраняются в области данных процесса обработки вызова;
- запускается таймер ожидания сообщения из кластера Adapter Б (TResponse);
- процесс обработки вызова переходит в состояние ожидания ответного сообщения (STATE3).

Обработка 6

В состоянии ожидания ответного сообщения из кластера Adapter Б (STATE3) сработал таймер TResponse, что означает, что Adapter Б не прислал никаких сообщений за заданный интервал времени, ситуация аварийная для вызова:

- активизируется процедура разъединения вызова;
- если в системе активирован механизм CFC (Call Forwarding by Cause), который обеспечивает фразы автоинформатора для определенных типов ошибок, то он запускается на выполнение по логике для ошибки "номер временно недоступен";
- формируется и подготавливаются к отправке в кластеры Adapter А и Adapter Б сообщения разъединения ReleaseReq;
- обновляется информация для модулей COPM и тарификации, итоговый информационный пакет отправляется в кластер CORE.

Обработка 7

В состоянии ожидания ответного сообщения из кластера Adapter Б (STATE3) получено сообщение разъединения ReleaseInd от кластера Adapter A, что означает, что абонент А положил трубку:

- анализируются параметры сообщения "ReleaseInd", проверяется их полнота и целостность;

- формируется и подготавливается к отправке в кластер Adapter Б сообщение разъединения "ReleaseReq";
- обновляется информация для модулей СОРМ и тарификации, итоговый информационный пакет отправляется в кластер CORE.

Обработка 8

В состоянии ожидания ответного сообщения из кластера Adapter Б (STATE3) получено сообщение "CallProgressInd" с индикатором выдачи абоненту Б сигнала "Посылка вызова" (BPtyAlerting):

- анализируются параметры сообщения "CallProgressInd", проверяется их полнота и целостность;
- формируется и подготавливается к отправке в кластер Adapter А сообщение "CallProgressReq" с индикатором "BPtyAlerting";
- активируются триггеры и запускаются процессы обработки активных сервисов;
- процесс обработки вызова переводится в логическое состояние ожидания ответа абонента Б (STATE4);
- обновляется информация для модулей СОРМ и тарификации;
- запускается таймер ожидания ответа абонента Б (TAnswer).

Обработка 9

В состоянии ожидания ответного сообщения из кластера Adapter Б (STATE3) получено сообщение "CallProgressInd" (промежуточное сообщение процесса установления соединения):

- анализируются параметры сообщения "CallProgressInd", проверяется их полнота и целостность;
- формируется и подготавливается к отправке в кластер Adapter А сообщение "CallProgressReq";
- процесс обработки вызова остается в том же состоянии (STATE3);
- запускается таймер ожидания сообщения из кластера Adapter Б (TResponse).

Обработка 10

В состоянии ожидания ответа абонента Б (STATE3 или STATE4) получено сообщение об ответе абонента Б (SetupResp):

- анализируются параметры сообщения "SetupResp", проверяется их полнота и целостность;
- формируется и подготавливается к отправке в кластер Adapter А сообщение "SetupConf";
- активируются триггеры и запускаются процессы обработки активных сервисов;
- процесс обработки вызова переводится в логическое состояние разговора (STATE5);
- обновляется информация для модулей СОРМ и тарификации;
- запускается таймер ограничения максимальной длительности разговора (TConversation).

Adapter Б

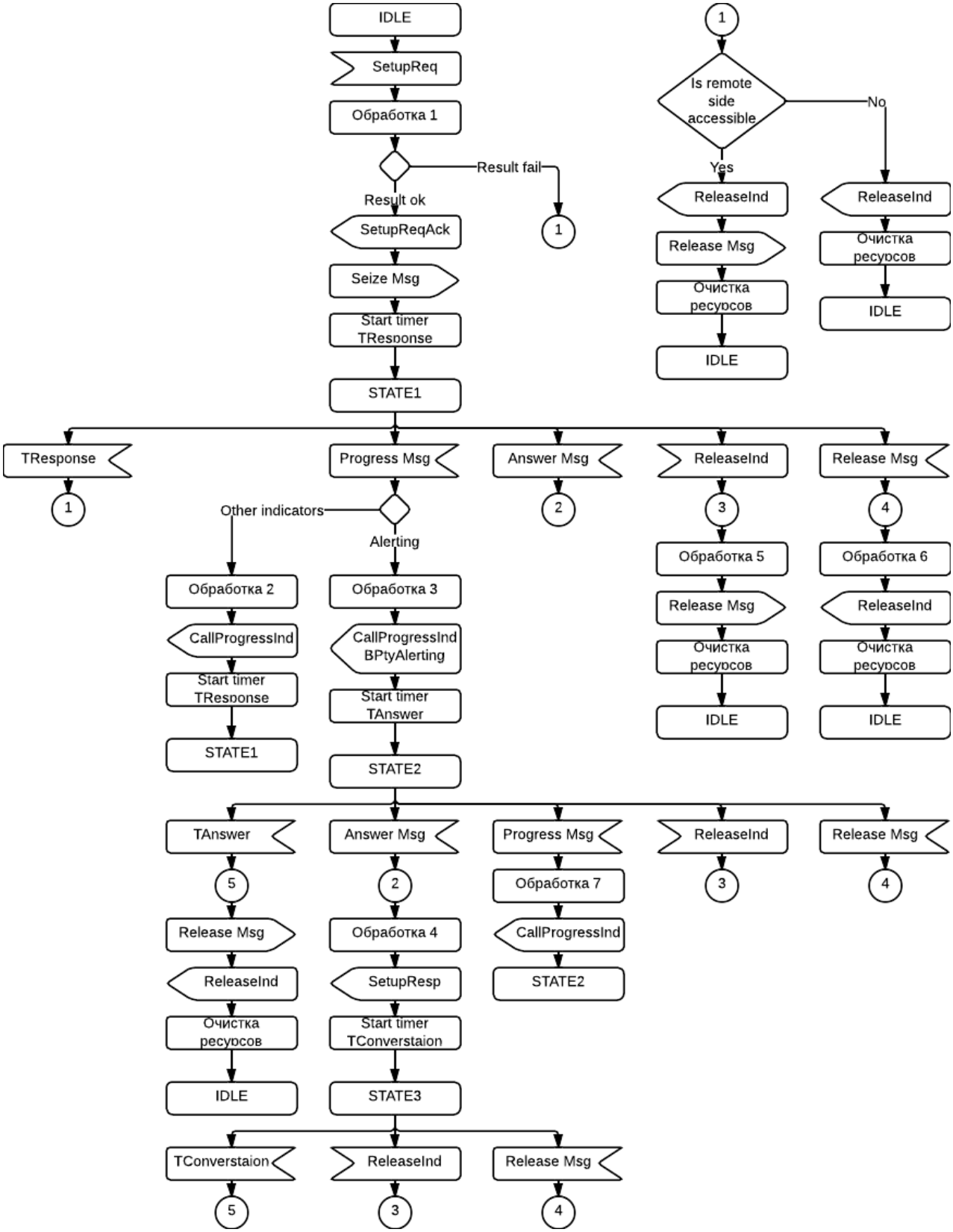


Рисунок 26 — Диаграмма работы terminating кластера Adapter (Adapter Б)

Описание действий, выполняемых в функциональных блоках диаграммы:

Обработка 1

В кластере Adapter Б получено сообщение о инициации вызова SetupReq.Выполняются следующие операции:

- анализ корректности и целостности полученного сообщения;
- проверка того, существует или нет активный вызов для интерфейса абонента Б: ** если вызов существует, то проверяется наличие ограничений на количество одновременных вызовов и если количество одновременных вызовов достигло установленного максимального уровня, производится отказ в обслуживании вызова, формируется и подготавливается к отправке в кластер Core сообщение о разъединении "ReleaseInd", обслуживание вызова прекращается;
- если ограничений на обработку вызова нет, то формируется и подготавливается к отправке в кластер Core сообщение "SetupReqAck";
- согласно правил целевого протокола, на основании данных полученных в "SetupReq", формируется и отправляется абоненту Б сообщение "Seize";
- процесс на ноде кластера Adapter Б переходит в состояние ожидания ответного сообщения от абонента/транка Б (STATE1);
- активируется таймер ожидания ответного сообщения от абонента Б (TResponse).

Обработка 2

В состоянии ожидания ответного сообщения от абонента/транка Б получено промежуточное сообщение "Progress":

- анализ корректности и целостности полученного сообщения, извлечение параметров;
- формируется и подготавливается к отправке в кластер Core сообщение "CallProgressInd" с данными заполненными на базе сообщения "Progress";
- активируется таймер ожидания ответного сообщения от абонента Б (TResponse);
- процесс обработки вызова остается в том же логическом состоянии (STATE1).

Обработка 3

В состоянии ожидания ответного сообщения от абонента/транка Б получено промежуточное сообщение "Progress" с индикатором "Alerting".

- анализ корректности и целостности полученного сообщения, извлечение параметров;
- формируется и подготавливается к отправке в кластер Core сообщение "CallProgressInd" с данными заполненными на базе сообщения "Progress", устанавливается индикатор "BPtyAlerting";
- активируется таймер ожидания ответа абонента Б (TAnswer);
- процесс обработки вызова переводится в состояние ожидания ответа абонента Б (STATE2).

Обработка 4

В состоянии ожидания ответного сообщения от абонента/транка Б (STATE1) или в состоянии ожидания ответа абонента Б (STATE2) получено сообщение об ответе абонента Б (Answer).Выполняется:

- анализ корректности и целостности полученного сообщения, извлечение параметров;
- формируется и подготавливается к отправке в кластер Core сообщение "SetupResp" с данными, заполненными на базе сообщения "Answer";

- активируется защитный таймер ограничения максимальной продолжительности разговора (TConverstaion);
- процесс обработки вызова переводится в состояние активного вызова (STATE3).

Обработка 5

Получено сообщение о разъединении с кластера Adapter A:

- анализ корректности и целостности полученного сообщения;
- формируется и подготавливается к отправке абоненту/транк Б сообщение о разъединении (Release);
- очищаются ресурсы, занятые процессом обработки вызова.

Обработка 6

Получено сообщение о разъединении от абонента/транка Б:

- анализ корректности и целостности полученного сообщения;
- формируется и подготавливается к отправке в кластер Core сообщение о разъединении (ReleaseInd);
- очищаются ресурсы, занятые процессом обработки вызова.

Обработка 7

В состоянии ожидания ответа абонента Б получено промежуточное сообщение "Progress":

- анализ корректности и целостности полученного сообщения, извлечение параметров;
- формируется и подготавливается к отправке в кластер Core сообщение "CallProgressInd" с данными, заполненными на базе сообщения "Progress";
- процесс обработки остается в логическом состоянии ожидания ответа абонента Б (STATE2).

Контейнеры параметров

В ECSS реализована иерархическая система контейнирования свойств различных сущностей.

На уровне системы определяются следующие виды сущностей:

1. **Кластер** — является совокупностью вычислительных узлов одного типа, выполняющих, с точки зрения системы, единую функцию. С их помощью описывается вычислительная топология системы.
2. **Виртуальная АТС (домен)** — группировка информации, относящейся к одной виртуальной АТС (домена). Позволяет задавать права доступа к просмотру/изменению информации.
3. **Алиас** — совокупность информации об абоненте.
4. **Интерфейс** — совокупность информации о коммуникационном порте, физическом или виртуальном (бридж).

Каждый вид сущностей обладает набором существенных характеристик:

1. **Кластер**
   Роль, Имя кластера
2. **Виртуальная АТС (домен)**
   Имя домена
3. **Алиас**
   Адрес, Имя домена, Имя интерфейса
4. **Интерфейс**
   Адаптер владелец, Группа, Имя интерфейса, Тип интерфейса

Каждый экземпляр сущности обладает своим набором параметров.
Например, определенный алиас характеризуется определенным адресом, определенным именем домена и определенным именем интерфейса, а также содержит в себе набор параметров, специфичных именно для этого алиаса.

Набор параметров определенного экземпляра сущности является совокупностью параметров взятой из иерархии профилей этой сущности.

Существуют следующие иерархии профилей/контейнеров параметров:

- **Для Кластера**:

1. **Параметры Кластера** — действуют на определенный кластер (задано имя кластера и роль кластера).

- **Для Интерфейса**:

1. **Параметры Интерфейса** — действуют на определенный интерфейс (задано имя интерфейса, группа, имя адаптера владельца, профиль интерфейса(опционально)).
2. **Профиль Интерфейса** — действует на все интерфейсы с заданным именем профиля.
3. **Профиль адаптера владельца и группы** — действует на все интерфейсы с заданным именем адаптера владельца и группой.
4. **Профиль адаптера владельца** — действует на все интерфейсы с заданным именем адаптера владельца.
5. **Профиль группы** — действует на все интерфейсы с заданной группой.
6. **Профиль глобальных параметров** — действует на все интерфейсы системы.

Параметры интерфейсов являются внутрисистемными, и не должны задаваться пользователями ECSS-10 непосредственно через CoCon.

- **Для Алиаса**:

1. **Параметры Алиаса** — действуют на определенный алиас (заданы имя домена, имя интерфейса, адрес, профиль алиаса (опционально)).
2. **Профиль Алиаса** — действует на все алиасы с заданными именем профиля.
3. **Профиль интерфейса** — действует на все алиасы с заданными именем домена и именем интерфейса (на данном уровне задаются свойства, которые необходимо задать для вызовов с/на определенных транков).
4. **Профиль адреса** — действует на все алиасы с заданным именем домена и адресом.
5. **Профиль домена** — действует на все алиасы с заданным доменом.

- **Для Домена**:

1. **Параметры Домена** — действуют на определенный домен (задано имя домена).
2. **Профиль глобальных параметров** — действует на все домены системы.

Параметры профиля с меньшей областью действия переопределяют параметры профиля с большей областью действия.
В приведенных выше иерархиях параметры уровня 1 переопределяют параметры уровня 2 и больших, и т.д.

На практике рекомендуется задавать значения параметров в наиболее общих профилях, т.е. на больших уровнях иерархии. Это позволяет хранить меньше данных, вносить групповые изменения в одном месте.

Схема информационных потоков

Система ECSS-10 поддерживает следующие способы обмена информацией (помимо VoIP):

- Внутрисистемный обмен через интеграционную шину (BUS) по протоколу AMQP (обработка вызовов, конфигурация, COPM, аварийные сообщения и т.д.).
- Внутрисистемный обмен контроля состояния доступности интерфейсов по протоколу VRRP.
- Взаимодействие с Автоматизированной системой расчетов по протоколу FTP.
- Взаимодействие с системой резервного копирования конфигураций по протоколу rsync.
- Взаимодействие с системой мониторинга и управления EMS либо другой системой OAM по протоколу SNMP.
- Взаимодействие с внешним почтовым сервером по протоколу SMTP (отправка нотификаций об авариях и пропущенных вызовах).
- Взаимодействие с внешним сервером обмена сообщениями по протоколу XMPP (отправка нотификаций о пропущенных вызовах).
- Взаимодействие с web-конфигуратором посредством web-сервисов и web-сокетов.
- Взаимодействие с внешним Radius-сервером по протоколу Radius (аутентификация и аккаунтинг).
- Взаимодействие с внешним LDAP-сервером по протоколу LDAP (централизованная аутентификация пользователей CoCon и SIP-абонентов).
- Взаимодействие с OSS посредством web-сервисов.
- Взаимодействие с порталом абонента посредством web-сервисов.

Интеграционная шина

Для унификации обмена сообщениями между кластерами системы ECSS-10 применяется обмен через интеграционную шину, которую обеспечивает кластер Bus.
Обмен сообщениями осуществляется по протоколу AMQP версии 0-10. Со спецификациями протокола AMQP версии 0-10 можно ознакомиться по ссылке: http://www.amqp.org/sites/amqp.org/files/amqp0-10.zip.

Далее будут рассмотрены механизмы, которые используются в ECSS-10 при взаимодействии через Bus.

AMQP является протоколом, ориентированным на передачу сообщений через очередь с поддержкой механизмов транзакционности и гарантии доставки.
Брокером AMQP называется сервер, который выполняет функции приема, маршрутизации, хранения сообщения в очереди и доставки сообщения получателю.
В AMQP различают клиента отправителя (Publisher) и клиента получателя

я (Consumer) сообщения.
Точка обмена (Exchange) выполняет функции маршрутизации сообщения при получении его брокером от отправителя. Целью "exchange" является определение того, в какую очередь должно быть доставлено каждое сообщение на основании транспортных параметров сообщения.

Свойство транзакционности и гарантии доставки, поддерживаемое протоколом AMQP, позволяет гарантировать как доставку сообщения от отправителя до брокера с одной стороны, так и доставку от брокера до получателя с другой. При этом поддерживаются

механизмы подтверждения доставки сообщения до всех нод кластера в случае схем с резервированием.

Типовой процесс доставки сообщения по протоколу AMQP можно отобразить следующим образом:

```
Отправитель == отправка ==> Точка обмена == маршрутизация ==> Очередь ==
доставка ==> Получатель
(Publisher)    (publish)    (Exchange)      (route)          (Queue)
(consumes)   (Consumer)
```

В процессе отправки сообщения отправитель устанавливает для сообщения различные параметры, которые влияют на процесс обработки и доставки сообщения.
Ключевыми параметрами, которые используются в ECSS-10, являются:

- exchange — точка обмена, которая должна использоваться для маршрутизации сообщения на брокере;
- routing.key — метка маршрутизации, которая используется при маршрутизации;
- reply_to.exchange — точка обмена, которая должна использоваться при отправке ответного сообщения;
- reply_to.routing_key — метка маршрутизации, которая должна использоваться при отправке ответного сообщения;
- time_to_live — время жизни сообщения.

Точка обмена (Exchange)

Протокол AMQP регламентирует несколько типов точек обмена, которые отличаются принципами маршрутизации поступающих в них сообщений.Поддерживаются следующие типа точек обмена:

- Direct exchange
- Fanout exchange
- Topic exchange
- Headers exchange

В системе ECSS-10 используется Direct Exchange.

В Direct Exchange маршрутизация входящих сообщений осуществляется в очереди на основании значения параметра "routing key" сообщения. Этот тип точки обмена идеален для организации unicast-потоков обмена сообщениями (можно организовать и multicast обмен).Direct Exchange работает следующим образом:

- очередь Q связывается с точкой обмена с меткой маршрутизации routing.key=K;
- когда новое сообщение поступает с меткой маршрутизации routing.key=R, то точка обмена маршрутизирует это сообщение в очередь Q, если R=K.

Direct exhange часто используется в ситуациях, когда нужно распределять задачи между несколькими однотипными обработчиками в режиме распределения нагрузки (round robin).

Схематически работа Direct exhcnage:

```
                                                    |==Msg1 → Consumer1
== Msgs → Direct exhcnage == Msgs.rk1 → Queue1 |==Msg2 → Consumer2
                                                    |==Msg3 → Consumer3
```

Очередь (Queue)

Очереди в AMQP очень похожи на очереди в других системах доставки сообщений или обработки задач и представляют собой накопитель сообщений, работающий по принципу FIFO (First In First Out).Часть параметров очередь наследует от точки обмена, а также обладает следующими индивидуальными параметрами:

- Name — имя;
- Durable — определяет, сохраняются ли сообщения в очереди при рестарте брокера;
- Exclusive — используется только одной коннекцией (получателем) и очередь удаляется, если коннекция закрывается;
- Auto-delete — очередь удаляется, когда последний получатель отписывается от очереди.

Для того чтобы очередь можно было использовать её необходимо задекларировать (declare). Декларация очереди ведет к её созданию, если очереди с таким именем и тем же набором атрибутов не существует. Декларация очереди не происходит, если уже существует очередь с именно таким именем и набором атрибутов (очередь уже существует, декларация считается успешно выполненной). Если же происходит декларация очереди и определяется, что очередь с таким именем, но отличающимся набором параметров, уже существует — возвращается ошибка "PRECONDITIONS_FAILED".

AMQP 0-10 определяет следующие общие правила именования очередей:

- имя очереди может содержать от 1 до 255 символов;
- первый символ имеет следующие ограничения: буквы a-z или A-Z, цифры 0-9 и символ подчеркивания '_';
- второй и последующие символы могут быть любыми UTF-8 символами.

В системе ECSS-10 применяются следующие дополнительные правила именования очередей:

- имя очереди состоит из набора смысловых частей;
- смысловые части разделяются символом точка ('.');
- имя заканчивается суффиксом 'q' (допускается использовать суффикс 'qe').

В настоящий момент используются следующие общие правила формирования имен очередей:

- **core.cp.'core_cluster_name'.'number_of_call_process_group'.'number_of_call_process'.qe** — очередь используется для отправки сообщений из кластера Adapter в кластер Core;
- **ecss.pa.sip-t.'pa_sip_cluster_name'.init.q** — очередь для приема инициирующих сообщений, поступающих в кластер адаптера SIP из кластера Core;
- **ecss.pa.sip-t.'pa_sip_cluster_name'.'session_identifier'.q** — очередь для приема сообщений кластером адаптера SIP из кластера Core в рамках вызова;
- **ecss.pa.megaco.'pa_megaco_cluster_name'::'gateway_name'.init.q** — очередь для приема инициирующих сообщений, поступающих в кластер адаптера Megaco из кластера Core;

- **ecss.pa.megaco.'pa_megaco_cluster_name'::'gateway_name'.'session_identifier'.q** — очередь для приема сообщений, поступающих в кластер адаптера Megaco из кластера Core в рамках вызова;
- **tring_client'client_version'....q** — очереди подсистемы контроля целостности tring системы ECSS-10;
- **ccn....q** — очереди подсистемы распределенной командной консоли CoCon системы ECSS-10;
- **ecss.cm....** — очереди подсистемы синхронизации конфигурации Configuration Manager системы ECSS-10;
- **ecss.rps....q** — очереди подсистемы обмена событиями, нотификациями, авариями RPS системы ECSS-10;
- **dds.bus....q** — очереди распределенной подсистемы доступа к хранилищу конфигурационной информации DDS кластера DS системы ECSS-10;
- **mycelium.mgmt....** — очереди, используемые брокером для синхронизации в кластере Bus.

Правила связывания (Bindings)

Bindings — это правила связывания/маршрутизации, которые используются в точке обмена для маршрутизации сообщений в очереди.
Для того чтобы точка обмена E могла направлять сообщения в очередь Q необходимо, чтобы Q была связана с E.
Правило связывания может иметь дополнительный "routing key" атрибут, который используется в некоторых типах точек обмена.
Назначение "routing key" в том, чтобы выбрать определенные сообщения, размещенные в "exchange", и направить их в соответствующую очередь. Таким образом, "routing key" работает как фильтр.

Если сообщение не может быть смаршрутизировано в точке обмена (например, нет соответствующего правила связывания), то оно отбрасывается либо возвращается отправителю (зависит от настроек атрибутов сообщения).

Получатели (Consumers)

Получатель подписывается на сообщения определенной очереди. На сообщения очереди может быть подписано несколько получателей.
Также предусмотрен механизм эксклюзивной подписки на очередь, тогда получатель сообщений у очереди может быть только один.

Процесс доставки сообщения получателю

Получатели представляют собой приложения, которые обрабатывают получаемые сообщения, производят определенные манипуляции, могут отправить какой-то ответ. В процессе своей работы у получателя может произойти ошибка (в том числе в процессе обработки полученного сообщения), которая приведет к рестарту получателя. Также к подобным последствиям потенциально могут привести ошибки на сети.
Эти ситуации вызывают вопрос о том, когда брокеру можно удалить сообщение из очереди.

Спецификации протокола AMQP определяют 2 возможных алгоритма доставки сообщения:

- basic.delivery/basic.get — в данном режиме сообщение удаляется из очереди сразу после отправки получателю;
- использование механизма basic.ack — в данном режиме функционирует надежная доставка. Сообщение удаляется, когда брокер получил подтверждение об успешной обработке сообщения получателем.

Первый вариант доставки сообщения называется режимом "автоматического подтверждения". Второй вариант можно назвать режимом "явного подтверждения". Особенностью второго варианта является то, что получатель может подтвердить доставку сообщения как сразу после получения сообщения, так и через какое-то время, после выполнения заложенной в нем бизнес-логики (например, после завершения транзакции по записи изменений в базу данных).

Если процесс получателя падает (падает коннекция к брокеру) либо истекает таймер ожидания подтверждения, брокер производит повторную доставку сообщения другому получателю. Если других получателей нет, то брокер будет ожидать появления получателя для последующей доставки ему сообщения.

В процессе обработки сообщения получателем может сложиться ситуация когда получатель не может обработать сообщение (некорректная информация, какая-то ошибка в логике, недоступность подсистем и т.п.). Получатель может просигнализировать брокеру об ошибке обработки сообщения путем отправки ему сообщения "reject". В процессе отправки "reject" получатель может сигнализировать брокеру о том, что сообщение должно быть сброшено либо предоставлено другому получателю (зависит от прикладной логики и характера ошибки).

Механизм управления потоком сообщений к получателю

В AMQP введен очень полезный механизм ограничения количества сообщений, которые брокер отправляет одному получателю. Обычно такие механизмы называются "flow control" (механизм управления потоком сообщений).

Механизм управления потоком AMQP позволяет ограничивать поток информации, отправляемой брокером получателю по:

- количеству переданной информации (потолок в байтах);
- количеству переданных сообщений.

В системе ECSS-10 этот механизм используется практически во всех кластерах и служит способом ограничения входящей нагрузки на процессы обрабатывающие сообщения.

Внутрисистемный обмен

С точки зрения транспорта взаимодействие компонент системы можно схематично изобразить следующим образом:

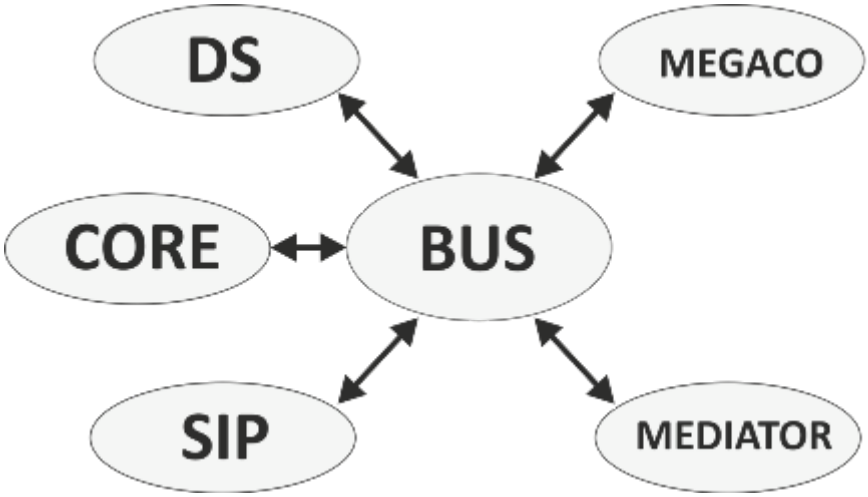


Рисунок 27 — Взаимодействие компонентов системы

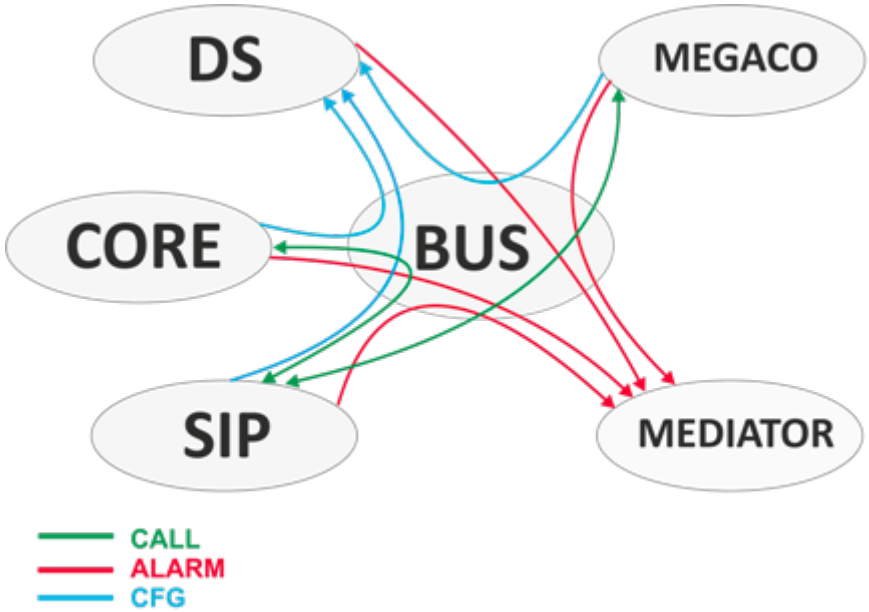


Рисунок 28 — Схема путей обмена сообщениями между кластерами через информационную шину

> *Для упрощения схемы взаимодействия далее в описании будет опущен кластер Bus. Но будет подразумеваться, что при обмене через AMQP все сообщения передаются через кластер Bus.*

В процессе работы системы существуют следующие виды связей между кластерами:

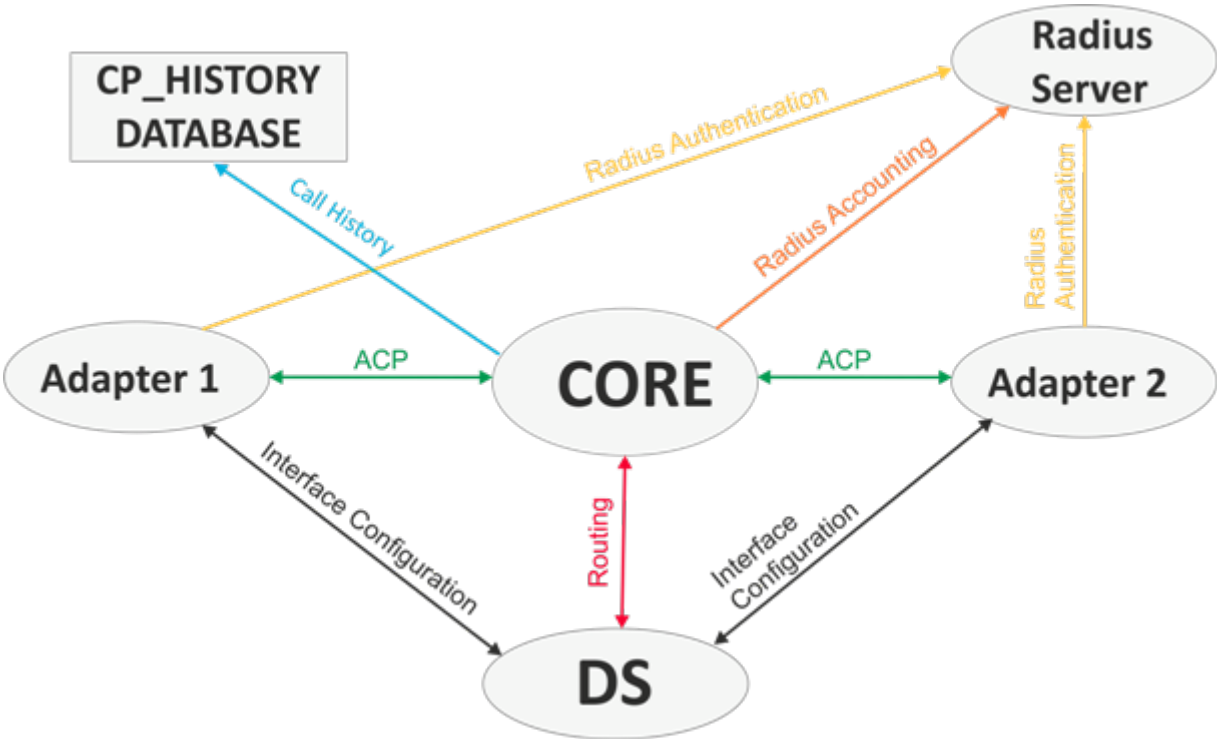


Рисунок 29 — Обмен сообщениями в системе ECSS-10 в рамках обслуживания вызова

На рисунке 29 приведена схема обмена сообщениями в системе ECSS-10 в рамках обслуживания вызова, где

- ACP (adapter core protocol) — внутренний сигнальный протокол, используемый системой ECSS-10 для обслуживания вызовов;
- Tollticket — детальная информация о вызове для последующего формирования CDR.

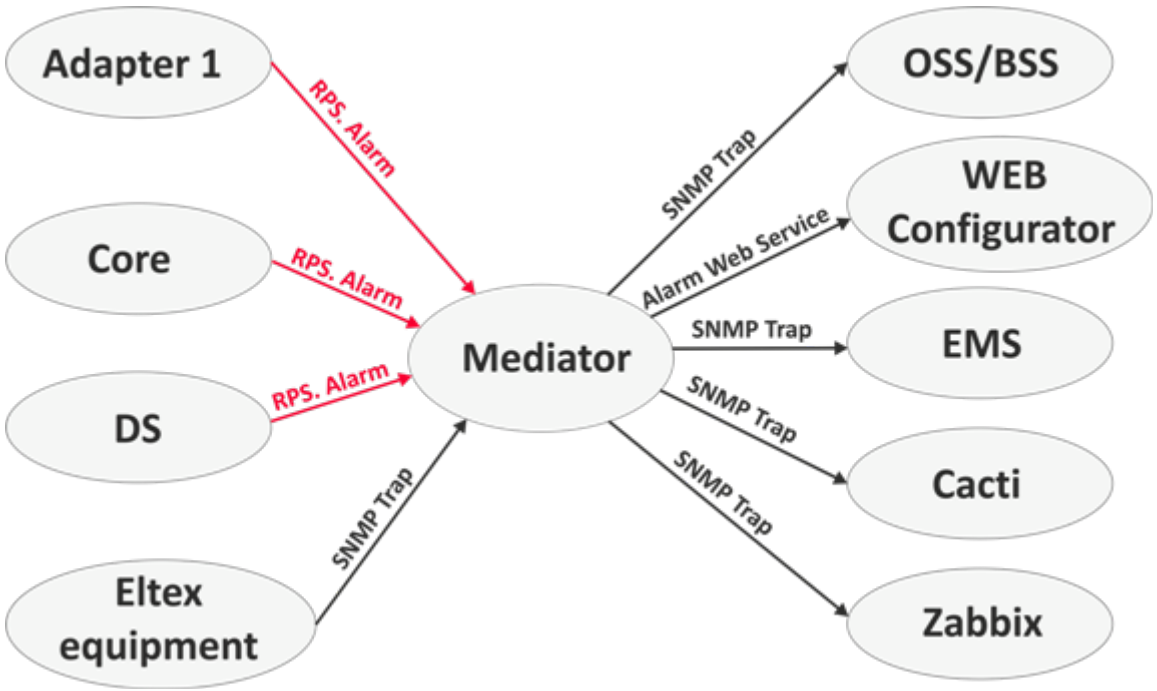


Рисунок 30 — Пути отправки сообщения об аварии системе ECSS-10

Информационные потоки сообщений распределенной консоли CoCon.
CoCon представляет собой распределенную облачную среду, к которой подключены все компоненты системы ECSS-10. Каждый подключенный компонент несет в себе функционал SSH-сервера, обеспечивающий возможность подключения клиентских терминалов по протоколу SSH. Каждый функциональный компонент содержит реализацию базовых команд CoCon (переходы по каталогам, команды управления нодой) и реализацию команд управления специфичных для этого компонента. При запуске компонента CoCon обеспечивает автоматическое связывание всех компонент в единую среду, за счет чего предоставляется возможность подключения к любому хосту системы ECSS-10 по протоколу SSH с возможностью выполнения любой команды вне зависимости от того где физически расположен компонент её реализующий. При выходе из строя какого-то компонента системы ECSS-10 он автоматически пропадает из CoCon (пропадает нода), а также пропадают команды, которые реализованы в этом компоненте. Когда компонент восстанавливает свою работу, он автоматически появляется в CoCon.

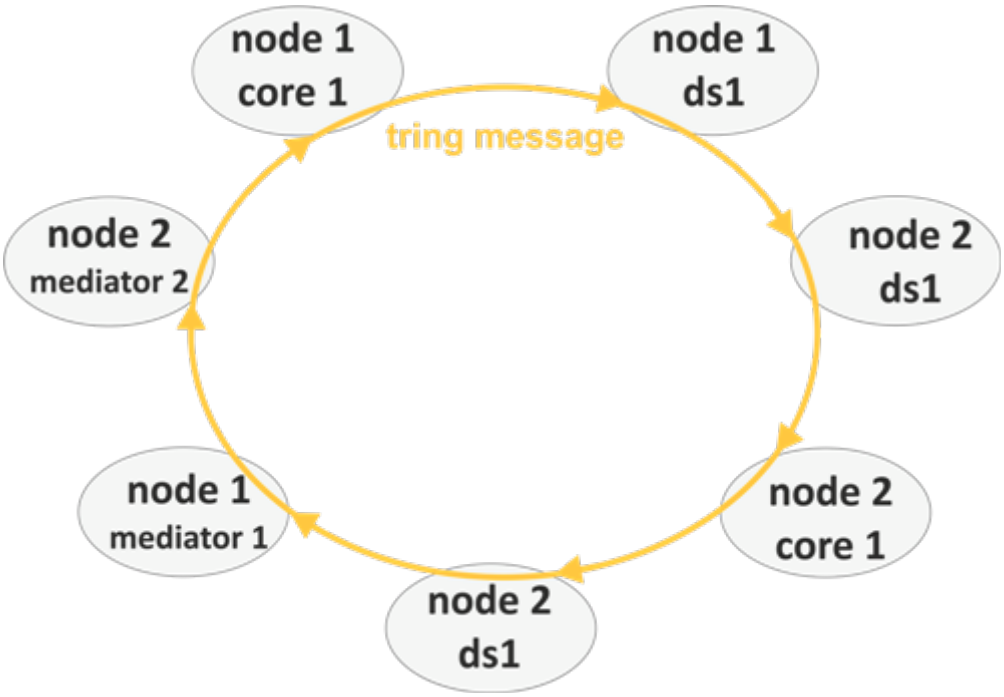


Рисунок 31 — Информационные потоки механизма мониторинга компонентов системы tring

Внутрисистемный обмен контроля состояния доступности интерфейсов

Система ECSS-10 при работе с подключаемыми к ней абонентами и шлюзами выглядит как единая система с одним IP-адресом для подключения, например, по протоколу SIP, несмотря на то, что в схеме с резервированием система разворачивается минимум на двух хостах. Такое функционирование системы обеспечивается организацией виртуального "расшаренного" между хостами интерфейса, который в каждый момент времени присутствует только на одном хосте. Этот функционал предоставляется службой keepalived и используется в протокольных адаптерах, которые непосредственно взаимодействуют с абонентами и шлюзами по VoIP-протоколам.

Механизм организации виртуального IP-адреса в keepalived реализован за счет использования протокола VRRP, который поддерживается на коммутаторах.
При конфигурировании системы каждому хосту устанавливается индивидуальный приоритет. В процессе работы keepalived посылает на коммутатор широковещательные запросы протокола VRRP, анонсируя информацию о хосте и приоритете. Остальные хосты получают эту информацию, сравнивают со своей. Если их приоритет ниже указанного в пакете, то не выполняют какие-либо действия. Если за заданный интервал времени хост не получает сообщения о том, что в сети доступен хост с более высоким приоритетом, то хост запускает механизм активации виртуального интерфейса у себя (становится мастером).
Keepalived осуществляет отправку контрольных пакетов с заданной в конфигурации периодичностью, при этом перед отправкой проверяет корректность функционирования ПО ноды кластера адаптера. Если ПО функционирует не корректно — пакет не отправляется. Это позволяет отрабатывать ситуации, когда ПО ноды выходит из строя, и переводить виртуальный интерфейс на другую ноду кластера.

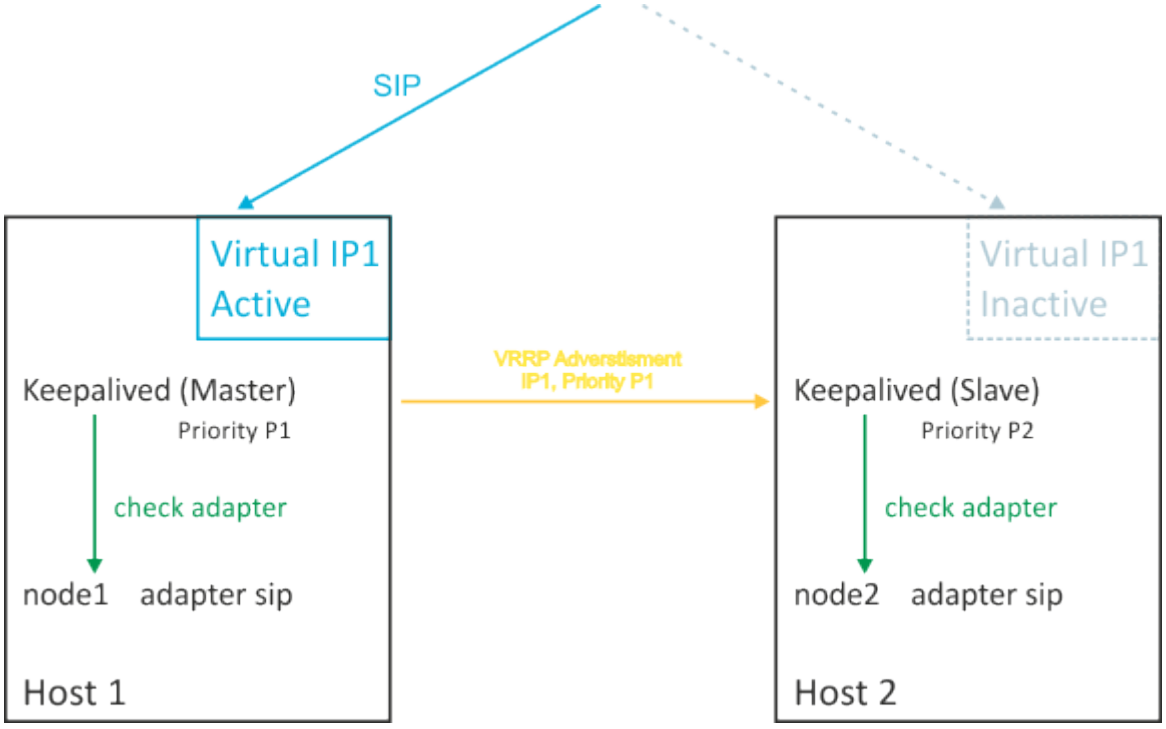


Рисунок 32 — Работа службы Keepalived

Транки и бриджи

Транки

Транк представляет собой интерфейс, соответствующий выходу из виртуальной АТС (домена). Соответствует транку на бридже (см. Менеджер бриджей (Bridge manager)). Представляет собой совокупность ресурсов для обслуживания телефонных вызовов в заданном направлении. Используется в маршрутизации, а так же и для ограничения входящих и исходящих линий.

Транки создаются либо с помощью команды declare, выполненной на соответствующем протокольном адаптере, например:

```
/domain/<Domain>/trunk/sip/declare
```

либо при создании бриджа:

```
/bridge/declare
```

Для транка могут быть настроены как на уровне системы, так и на уровне домена следующие ограничения:

- `bandwidth/in` — максимальное количество входящих линий;
- `bandwidth/out` — максимальное количество исходящих линий;
- `bandwidth/total` — максимальное суммарное количество линий.

Значениями `bandwidth/in`, `out` и `total` могут быть целые неотрицательные числа, либо специальное значение `unbounded`, которое говорит о том, что каких-либо специальных ограничений на уровне SSW не накладывается. Это не отменяет ограничений конкретного протокола, специальных настроек конкретных адаптеров, физических ограничений, а также настроек других станций.

Кроме ограничений по количеству линий можно настроить ограничения по CPS.

В примере показаны значения по умолчанию для транка:

```
admin@[mycelium1@ecss1]:/$ domain/biysk.local/trunk/info sbc.gr to_aster

Trunk: to_aster, Active: true, Type: sip
```

Property	In	Out	Total
System bandwidth	unbounded	unbounded	unbounded
Domain bandwidth	unbounded	unbounded	256
Actual bandwidth	unbounded	unbounded	256
Active calls	0	0	0
CPS	0	0	0
CPS Limit	256	256	256
WhiteList	undefined (false)	undefined (false)	-
BlackList	undefined (false)	undefined (false)	-
Stat/max_cps	0	0	0
Stat/calls	0	0	0
Stat/rejected	0	0	0

Логика работы системных и доменных ограничений для транков такая, как и для бриджей.

Транки используются в маршрутизации вызовов (подробнее Виртуальная АТС. Маршрутизация телефонных вызовов).

Подробнее об управлении SIP-транками — см. раздел Управление SIP-транками.

Бриджи

Бридж (Bridge) — виртуальный транк, позволяющий соединять между собой две виртуальные АТС в рамках одной системы ECSS-10 (см. Менеджер бриджей (Bridge manager)).

При создании бриджа в каждом из двух доменов автоматически декларируется по одному транку, связанному с данным бриджем.

Помимо имеющихся доменных ограничений транка `bandwidth/in`, `bandwidth/out` и `bandwidth/total`, для бриджей могут быть установлены соответствующие им ограничения с префиксом `system/`, например `system/bandwidth/total`. Данные ограничения могут устанавливать и изменять только администраторы системы, а администраторам домена они доступны только для чтения.
Системные настройки используются только для бриджей. Для sip транков значения `system/bandwidth/*` будут равны unbounded.

Администратор домена может изменять значения `bandwidth/in`, `bandwidth/out` и `bandwidth/total`. При этом они могут превышать установленные администратором системы значения, однако актуальные значения ограничений — то есть те, которые фактически будут применяться к вызовам, проходящим через транк, не будут превышать системные ограничения. При изменении системных ограничений, доменные ограничения остаются неизменными — меняются только актуальные значения.

Пример:

Системное значение	Доменное значение	Актуальное значение
10	5	5
10	15	10
10	unbounded	10
unbounded	15	15
unbounded	unbounded	unbounded

Как видно из данной таблицы, при вычислении актуального значения, будет использовано минимальное из двух значений — системного и доменного.

Как для ограничений уровня системы, так и для ограничений уровня домена, специальное значение unbounded, говорит, что на данном уровне никаких искусственных ограничений на пропускную способность не накладывается. Мы можем вообще искусственно не ограничивать пропускную способность ни на одном из уровней, как это видно из последней колонки. Разумеется, последнее требует некоторой осторожности от администратора домена и особенно администратора системы, так как это может негативно сказаться на общей производительности в случае, если транк будет использоваться чрезмерно интенсивно (например, при неправильно настроенном скрипте автообзвона). Однако отсутствие ограничения может быть удобно, например, если компания-владелец системы сама является её пользователем, а не предоставляет функционал ВАТС третьим сторонам.

В версиях ECSS-10 до 3.10.0 значение unbounded не поддерживалось.

Бридж может работать в симплексном и дуплексном режимах. Симплексный бридж из домена А в домен Б пропускает вызовы только из домена А в домен Б. Дуплексный бридж пропускает вызовы в обоих направлениях. Подробнее о создании дуплексных и симплексных бриджей см. раздел /bridge/ — команды управления bridge-интерфейсами.

Описание системы СОРМ

Для выполнения требований по поддержке СОРМ в системе ECSS-10 устанавливается специализированный модуль Посредник СОРМ, который обеспечивает подключение ПУ СОРМ по стандартизированному интерфейсу к системе. Посредник СОРМ выполнен в конструктиве SMG1016M.

Программный модуль SORM кластера CORE системы ECSS-10 получает команды от Посредника СОРМ по шифрованному SSH-каналу и отправляет оперативную информацию о вызовах на Посредник СОРМ.

В процессе работы модуль SORM накапливает всю необходимую для взаимодействия с Посредником СОРМ информацию об активности наблюдаемых вызовов:

1. CallRef — идентификатор вызова;
2. номер абонента А с параметрами;
3. номер абонента Б с параметрами;
4. транк, с которого вызывает абонент А;
5. транк, через который вызов проключен к абоненту Б;
6. флаг контроля вызова.

Все эти данные хранятся в надежном, резервированном хранилище кластера CORE и предоставляются в Посредник СОРМ, а далее в ПУ СОРМ.

Для обеспечения требований стандарта на СОРМ оперативная информация о вызовах предоставляется модулю SORM кластером Core в режиме реального времени.
Режим прослушивания телефонных разговоров обеспечивается принудительным проключением вызовов абонентов, за которыми идет наблюдение через Посредник СОРМ, который осуществляет передачу медиапотока вызова в канал к ПУ СОРМ.

Организация распределенной отказоустойчивой сети

Исходные данные

Исходные данные — требуется организовать распределенную отказоустойчивую сеть со следующими элементами:

- АТС — в качестве главной АТС используется Софтсвич ECSS-10 с георезервированием (использование георезервирования опционально);
- SBC — SBC на базе SMG-2016/3016 (использование опционально);
- СОРМ — в качестве СОРМ посредника используется SMG-2016/3016 (использование опционально);
- Вынесенные узлы, которые в случае потери связи с главной АТС, смогут обеспечить непрерывную работу связи. Каждый из узлов может содержать в себе:
    - Сигнальные медиа шлюзы SMG различной серии. В том числе и с резервом потоков E1;
    - Абонентские шлюзы ТАУ;
    - Маршрутизаторы с функцией телефонии — ESR-XXVF;
    - Вынесенные медиа сервера.

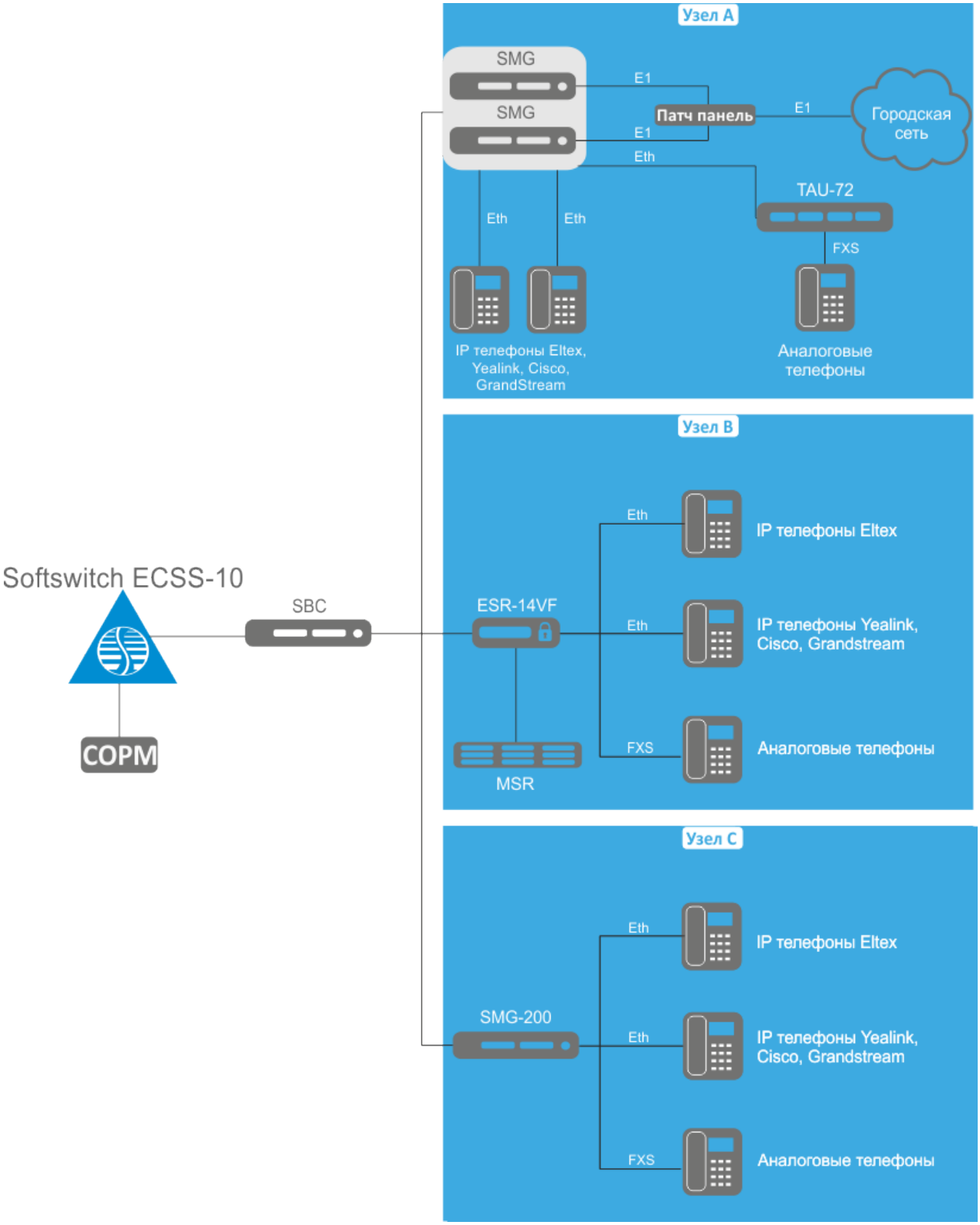


Рисунок 33 — Типовая схема организации распределенной отказоустойчивой сети

Описание вынесенных узлов связи

Вынесенные узлы связи могут в себе содержать различный набор оборудования. Ниже приведено описание узлов связи их типовой схемы, расположенной сверху.

Узел A

В данном узле содержится две SMG-2016 в стеке, которые обеспечивают резервирование потоков E1.
Данная SMG имеет выход в городскую сеть по потоку E1. Также на SMG-2016 зарегистрированы Smart терминалы (Eltex VP-12/15, телефоны вендров Yealink, Cisco, GrandStream).
ТАУ-72 используется для подключения аналоговых абонентских терминалов через FXS порты. Сама ТАУ-72 регистрирует своих абонентов на SMG-2016.

Узел B

В данном узле вместо SMG-2016 содержится маршрутизатор с функциями VoIP — ESR-14VF. Данный маршрутизатор имеет в себе встроенный VoIP-сервер, который способен обслуживать как Smart терминалы, так и аналоговые терминалы, подключенные к нему через встроенные FXS порты.
Выход в сеть IP осуществляется за счет отдельного линка, уходящего в сеть Интернет.
Для резервирования медиа трафика в данном узле стоит вынесенный медиа сервер MSR.

Узел C

В данном узле содержится SMG-200. На данной SMG включен режим транзитной регистрации. К данной SMG подключены Smart терминалы (Eltex VP-12/15, телефоны вендров Yealink, Cisco, GrandStream).
Также к ней подключены аналоговые телефонные аппараты через встроенные FXS порты.

Таких узлов может быть гораздо больше, схема является масштабируемой. В самих узлах также имеется возможность использовать различное оборудование, набор которого не ограничен приведенным выше примером.

Описание работы резервирования

Резервирование Софтсвича ECSS-10

Резервирование программного коммутатора ECSS-10 может обеспечиваться за счет собранного кластера. Описание настройки Софтсвича ECSS-10 в качестве кластера содержится в разделе Конфигурирование кластеров.
Кроме того, система Софтсвича ECSS-10 позволяет обеспечить географическое резервирование в случае сильной территориальной разрозненности сети. Данный пункт описан в разделе Настройка георезерва.

Также требуется обеспечивать резервирование DNS серверов, который резолвят dns-имена для всех шлюзов в рамках данной сети.

Резервирование вынесенных узлов связи

Для того чтобы обеспечить резервирование в вынесенных узлах сети, требуется настроить на шлюзах SMG транзитную регистрацию.
Полная настройка транзитной регистрации на SMG описана в разделе Сервер выживания (транзитная регистрация).

Софтсвич ECSS-10 имеет в своей базе всех абонентов и индивидуальные настройки ДВО, которых обслуживает в рамках сети. В свою очередь, в вынесенных узлах обслуживается только часть абонентов из всех сети. Поэтому на SMG требуется сконфигурировать только тех абонентов, которые обслуживаются в рамках данного узла. Также, им требуется назначить тот набор ДВО, который позволит абонентам комфортно пользоваться услугами связи даже в аварийной ситуации на сети. Тем самым, обеспечится резервирование сигнализации.
Изначально, телефонные аппараты будут совершать регистрацию и звонки через вышестоящий шлюз SMG, который в свою очередь будет делать вызов на Софтсвич ECSS-10.
В случае падения линка до Софтсвича ECSS-10, вся локальная связь, а также связь с внешним миром, будет осуществляться через SMG. Это осуществляется за счет транзитной регистрации.
Также, имеется возможность обеспечить резервирование медиа трафика за счет вынесенных медиа серверов.

За счет такого подхода к построению сети имеется возможность масштабировать сеть, добавляя вынесенные узлы с включенной функцией транзитной регистрации.

Пример настройки

Софтсвитч ECSS-10 обслуживает в рамках сети 3000 абонентов, чьи номера начинаются с 1000 по 4000. Каждый такой абонент может пользоваться услугами HOLD, 3way conference, DND, CFU
На сети имеется три вынесенных узла связи, каждый из которых обслуживает 1000 внутренних абонентов. Узел А обслуживает номера с 1000 по 1999, Узел В обслуживает номера с 2000 по 2999 и т.д.

Настройка Софтсвитч ECSS-10:

1. Создание абонентов. Описание создания абонентов описано в пункте Добавление абонента.

```
admin@core1@ecss1:/$ domain/ecss10/sip/user/declare default_routing
sip.user {1000-4000}@ecss10 alias-as-user no_qop_authentication
common_login 123 123
Executed on the sip1@ecss1
Intermediate (incomplete) result:
Declaration for range: 1000@ecss10..4000@ecss10 (3000)
...
3000 interfaces check for existing ...
[**********************************************************************]
55mks
3000 users interfaces declaration ...
[**********************************************************************]
10ms
3000 users aliases declaration ...
[**********************************************************************]
58ms
3000 interfaces recall to base
[**********************************************************************]
14ms
Executed on the sip1@ecss1
┌──────────────────────────────┐
│ declared 3000 subscribers │
└──────────────────────────────┘
```

2. Включение и активация услуг на абонентах. Описание работы с услугами описано в пункте Инсталляция и управление услугами.

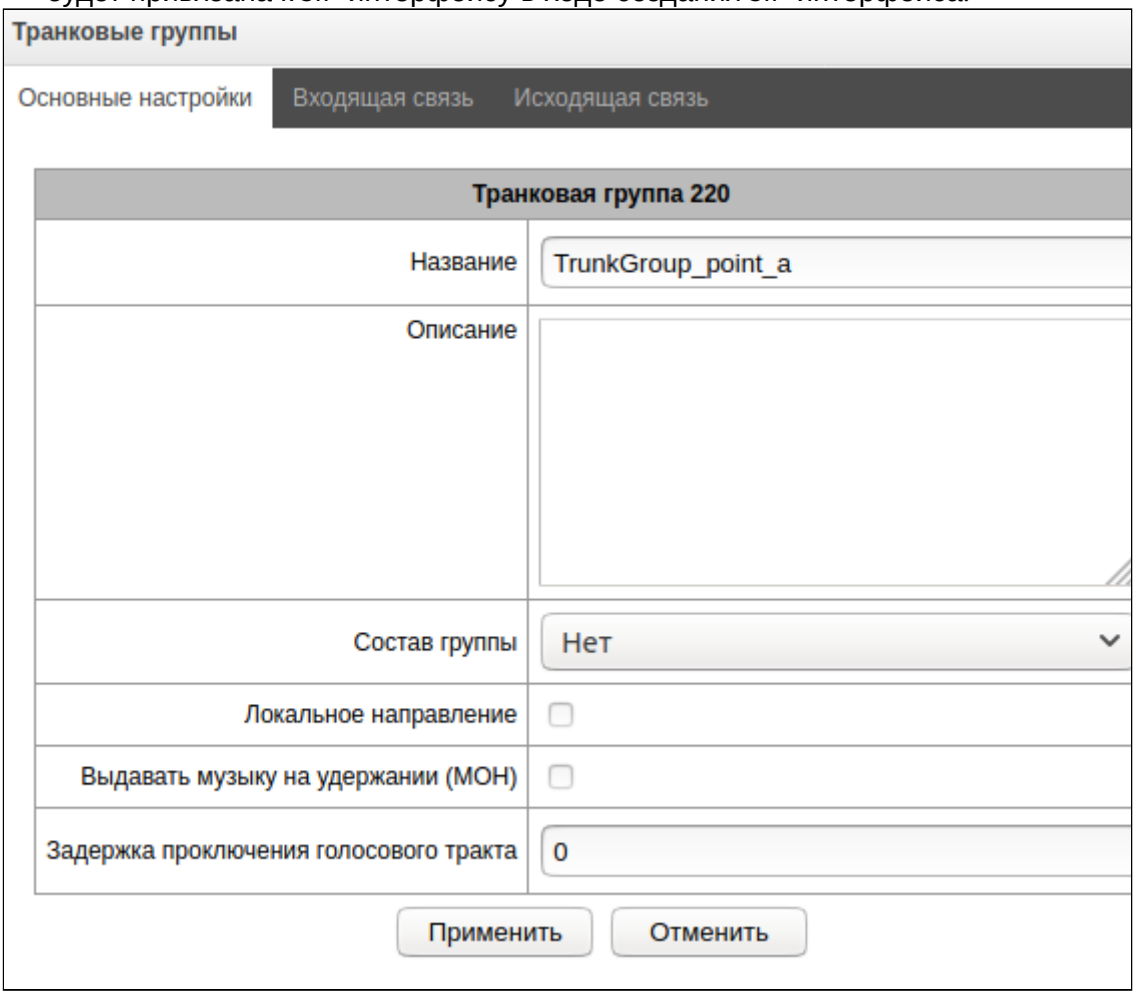
```
admin@core1@ecss1:/$ domain/ecss10/ss/enable {1000-4000} chold 3way dnd
cfu
Success: Supplementary service cfu enabled for domain "refactor",
address "{1000-4000}".
Success: Supplementary service dnd enabled for domain "refactor",
address "{1000-4000}".
Success: Supplementary service 3way enabled for domain "refactor",
address "{1000-4000}".
Success: Supplementary service chold enabled for domain "refactor",
address "{1000-4000}".

admin@core1@ecss1:/$ domain/ecss10/ss/activate {1000-4000} chold
Success: Supplementary service chold activated for domain "ecss10"
address "{1000-4000}"

admin@core1@ecss1:/$ domain/ecss10/ss/activate {1000-4000} 3way
Success: Supplementary service dnd activated for domain "ecss10" address
"{1000-4000}"
```

3. Создание транкового направления в сторону вынесенного Узла А. Описание создания транка описано в пункте Команды управления транками SIP.

```
admin@core1@ecss1:/$ domain/ecss10/trunk/sip/declare default_routing
sip.trunk poin_a ecss10 static 192.168.116.161 5060 sip-proxy 5099
Executed on the sip1@ecss1
declared
```

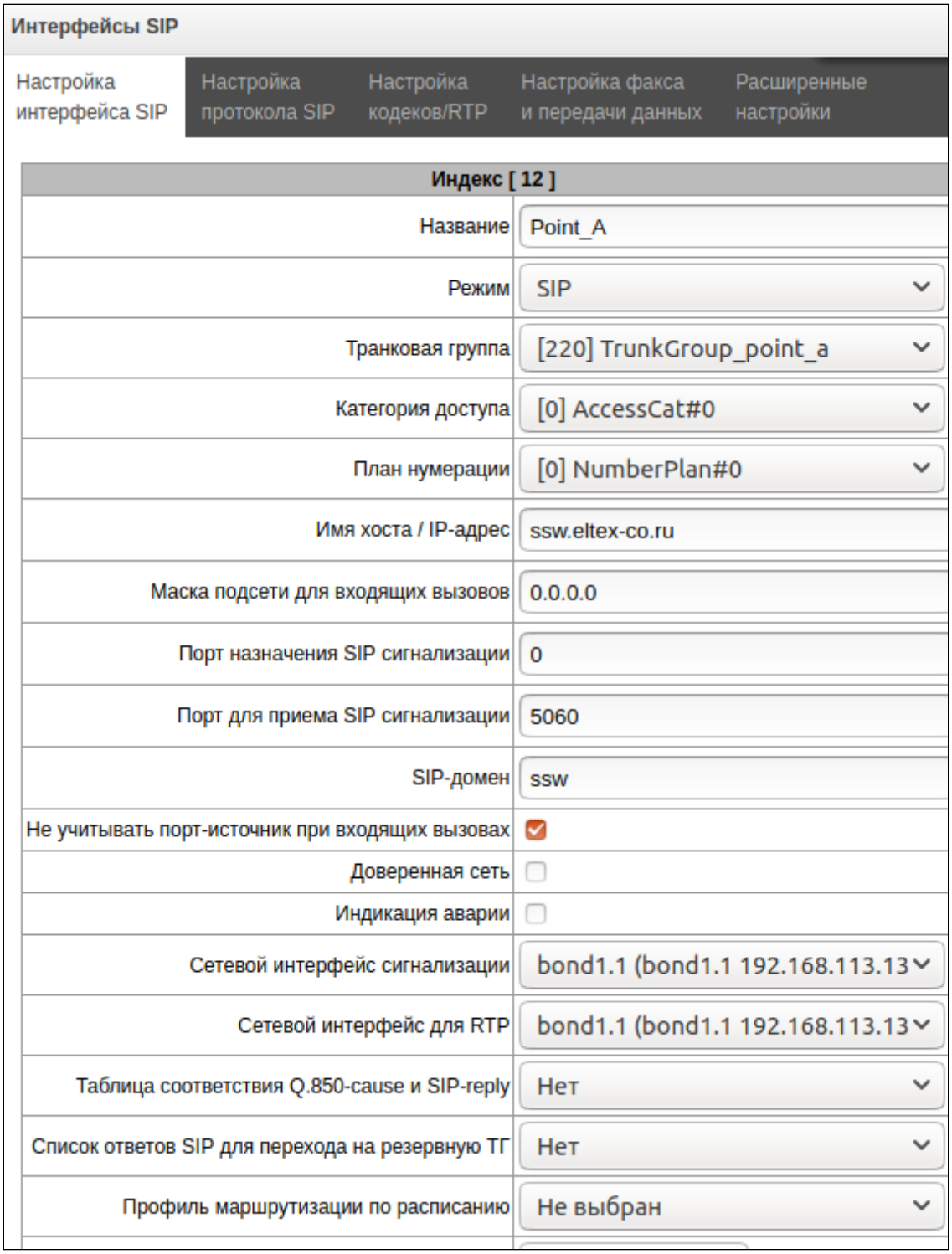
Настройка SMG в узле А:

1. Создать транковую группу для центральной АТС:
   - Перейти в настройки транковых групп (*Маршрутизация — Транковые группы*);
   - Добавить новую транковую группу;
   - Задать название, например, "TrunkGroup_point_a";
   - В выпадающем меню "Состав группы" ничего выбирать не надо, транковая группа будет привязана к SIP-интерфейсу в ходе создания SIP-интерфейса.



2. Создать SIP-интерфейс для подключения к центральной АТС:
   - Перейти в настройки интерфейсов SIP (*Маршрутизация — Интерфейсы SIP*);
   - Добавить новый SIP-интерфейс;

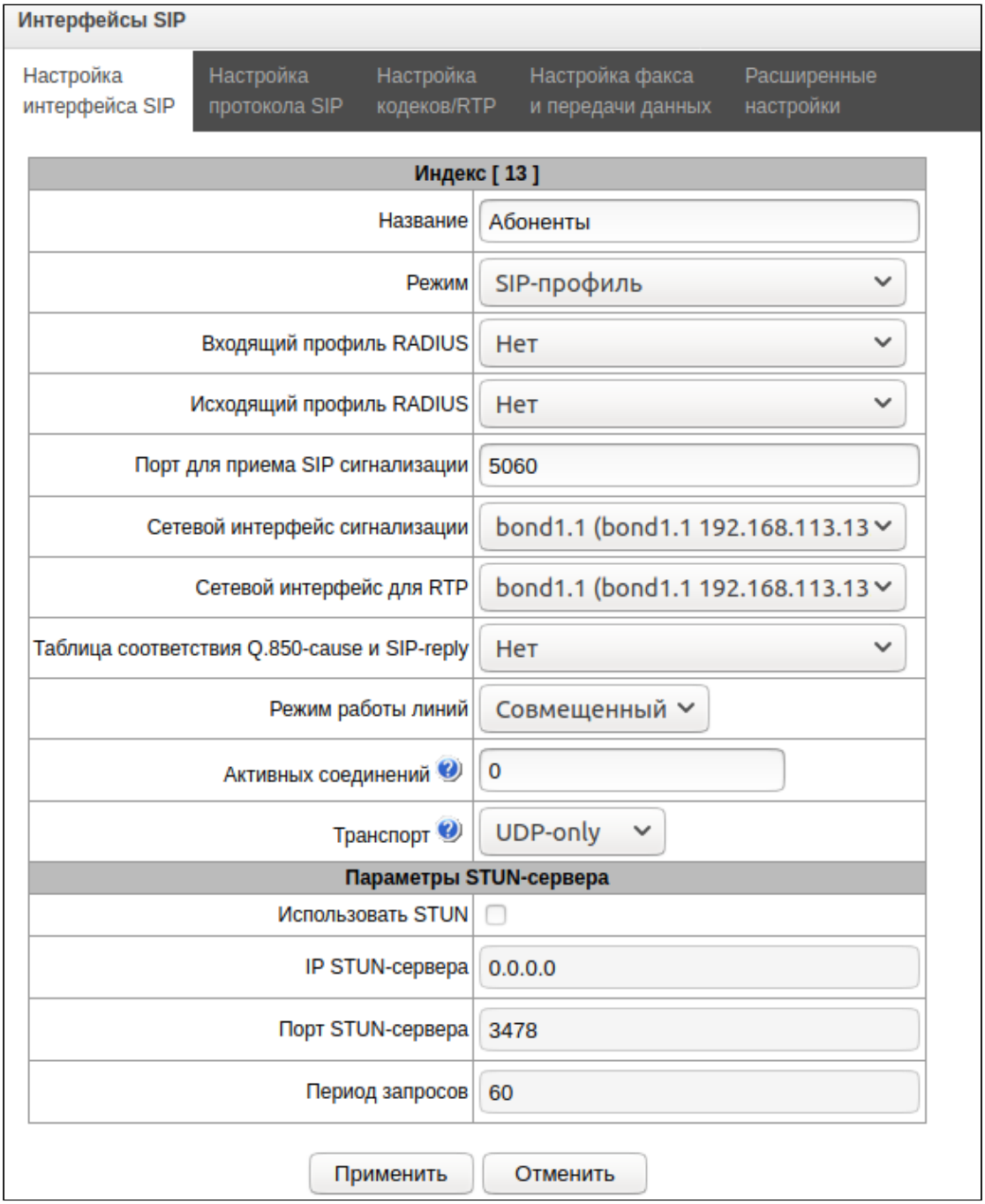
- Задать название, к примеру, "Point_A";
- Выбрать транковую группу "TrunkGroup_point_a";
- Задать IP-адрес встречного устройства (городская АТС) в поле "Имя хоста / IP-адрес";
- Если используются отличные от стандартных (5060) порты сигнализации, то их нужно задать в полях "Порт назначения SIP сигнализации" и "Порт для приема SIP сигнализации";
- Выбрать сетевые интерфейсы для сигнализации и RTP;



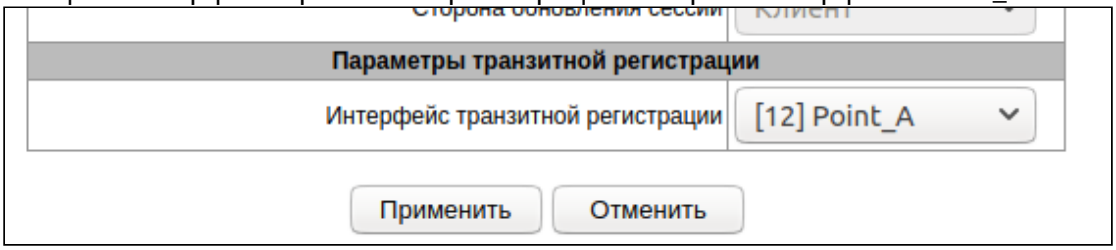
- На вкладке "Настройки протокола SIP":
  Включить "Контроль доступности встречной стороны"
  Выбрать "транзитная регистрация" в опции "Регистрация на вышестоящем сервере".

Параметры регистрации	
Регистрация на вышестоящем сервере	транзитная регис Для данного режима регистрации необходимо включить контроль доступности встречной сторо
Логин	
Пароль	
Имя пользователя/Номер	
CdPN по умолчанию	
Подмена CgPN при исходящем вызове	
Период регистрации (сек)	1800
Интервал запросов регистрации (мс)	1000

3. Создать SIP-профиль:
   - Перейти в настройки интерфейсов SIP (раздел *Маршрутизация — Интерфейсы SIP*);
   - Добавить новый SIP-интерфейс;
   - В открывшемся окне задать название интерфейса, к примеру, "Абоненты";
   - Выбрать режим "SIP-профиль";
   - Выбрать сетевые интерфейсы сигнализации и RTP для нужной подсети.

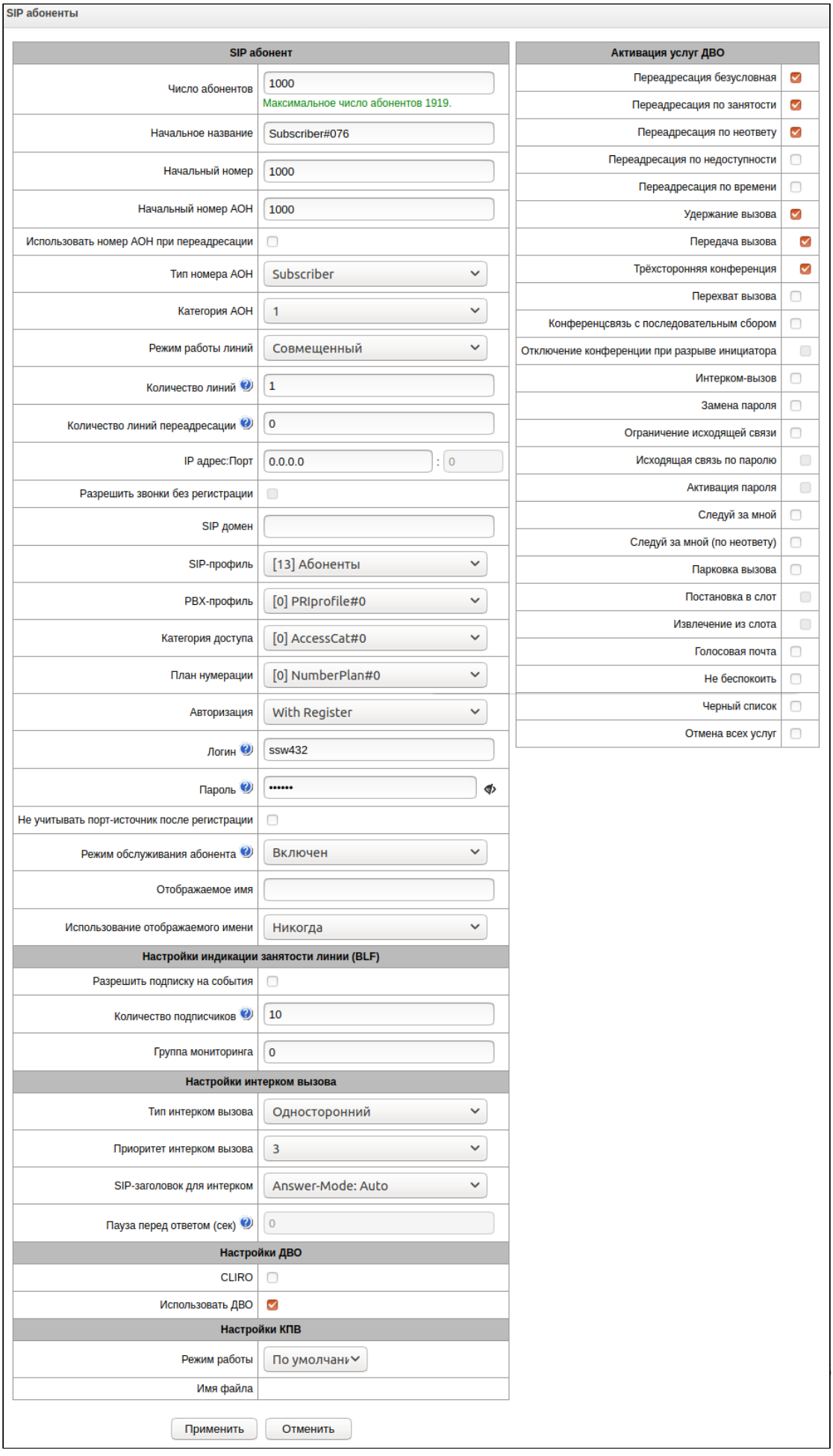


- На вкладке "Настройки протокола SIP":
  В опции "Интерфейс транзитной регистрации" выбрать интерфейс "Point_A".



4. Создать SIP-абонентов:

- Перейти в настройки Абонентов (раздел Абоненты — *SIP абоненты*);
- Создать новых SIP-абонентов;
- В открывшемся окне задать число абонентов, в нашем случае 1000;
- Задать начальный номер, в нашем случае 1000;
- Выбрать ранее созданный SIP-профиль — "Абоненты";
- Ввести логин и пароль (такой же, как и на Софтсвиче ECSS-10);
- Включить требуемые ДВО.


SIP абоненты
SIP абонент
Число абонентов
1000
Максимальное число абонентов 1919.
Начальное название
Subscriber#076
Начальный номер
1000
Начальный номер АОН
1000
Использовать номер АОН при переадресации
Тип номера АОН
Subscriber
Категория АОН
1
Режим работы линий
Совмещенный
Количество линий
1
Количество линий переадресации
0
IP адрес:Порт
0.0.0.0
0
Разрешить звонки без регистрации
SIP домен
SIP-профиль
[13] Абоненты
PBX-профиль
[0] PRIprofile#0
Категория доступа
[0] AccessCat#0
План нумерации
[0] NumberPlan#0
Авторизация
With Register
Логин
ssw432
Пароль
Не учитывать порт-источник после регистрации
Режим обслуживания абонента
Включен
Отображаемое имя
Использование отображаемого имени
Никогда
Настройки индикации занятости линии (BLF)
Разрешить подписку на события
Количество подписчиков
10
Группа мониторинга
0
Настройки интерком вызова
Тип интерком вызова
Односторонний
Приоритет интерком вызова
3
SIP-заголовок для интерком
Answer-Mode: Auto
Пауза перед ответом (сек)
0
Настройки ДВО
CLIRO
Использовать ДВО
Настройки КПВ
Режим работы
По умолчани
Имя файла
Применить
Отменить
Активация услуг ДВО
Переадресация безусловная
Переадресация по занятости
Переадресация по неответу
Переадресация по недоступности
Переадресация по времени
Удержание вызова
Передача вызова
Трёхсторонняя конференция
Перехват вызова
Конференцсвязь с последовательным сбором
Отключение конференции при разрыве инициатора
Интерком-вызов
Замена пароля
Ограничение исходящей связи
Исходящая связь по паролю
Активация пароля
Следуй за мной
Следуй за мной (по неответу)
Парковка вызова
Постановка в слот
Извлечение из слота
Голосовая почта
Не беспокоить
Черный список
Отмена всех услуг

После проделанных действий нужно зарегистрировать абонентов на SMG. В случае аварийной ситуации, при которой будет потеряна связь с Софтсвитч ECSS-10, абоненты будут обслуживаться локально через SMG.

Пример настройки телефонов VP-1x/2x для использования с сервером выживания

После успешной настройки SSW и SMG, которая описана в данном разделе Организация распределенной отказоустойчивой сети , требуется зарегистрировать телефоны VP-1x/2x в различных режимах.

Одновременная регистрация на SMG и SSW

Исходные данные:

Требуется зарегистрировать телефон VP-12 одновременно на SMG и SSW.

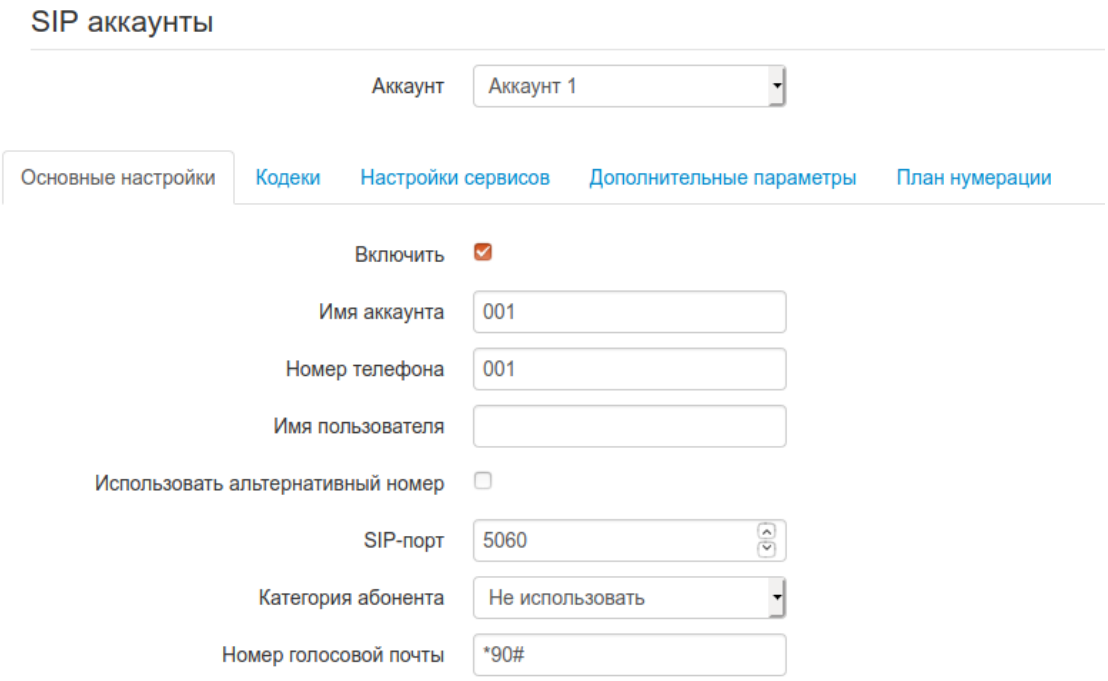
- Номер телефона — 001
- Логин — 001
- Пароль — 001
- SIP-домен — refactor
- IP SMG — 192.168.116.161
- IP SSW — 192.168.116.132
- Порт регистрации — 5099

Перед тем как начать настройку телефона VP нужно настроить SSW и SMG в соответствии с пунктом Организация распределенной отказоустойчивой сети.

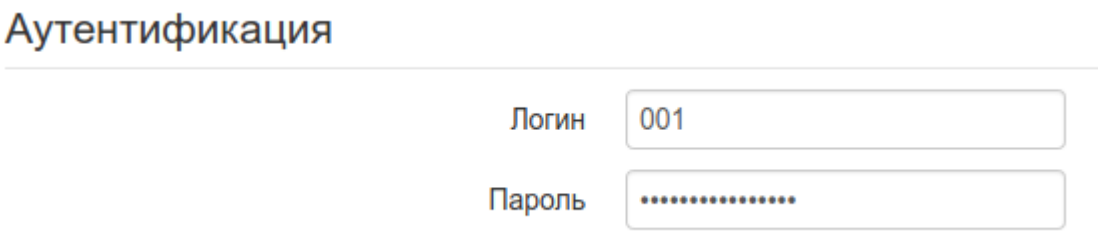
Настройка VP-1x/2x будет производиться через web-интерфейс.

1. Перейти во вкладку *IP-телефония — SIP-аккаунты — Аккаунт 1* и ввести требуемые значения в SIP-аккаунты
    - Включить SIP-аккаунт
    - Имя аккаунта — 001

- Номер телефона — 001

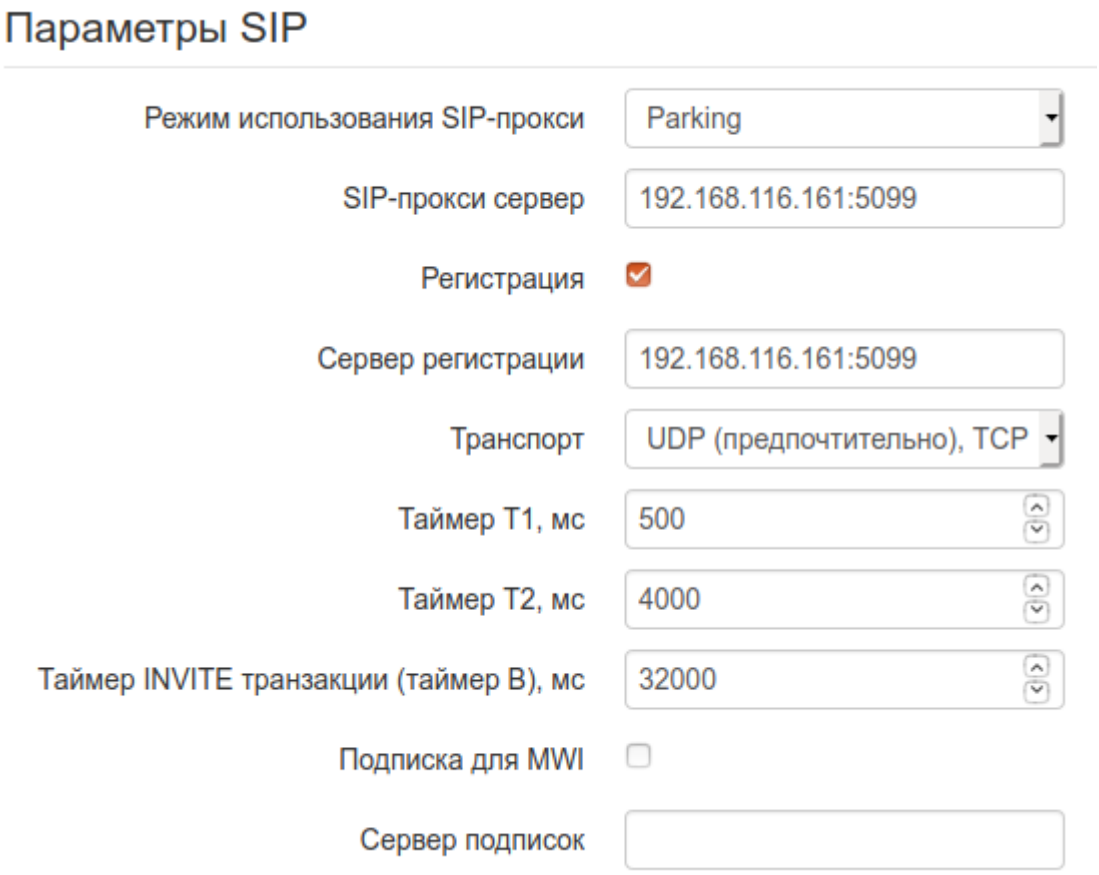


2. Аутентификация
   - Логин — 001
   - Пароль — 123

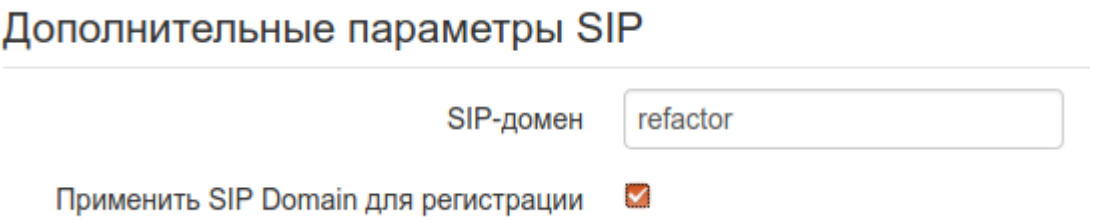


3. Параметры SIP
   - Режим использования SIP-прокси — Parking
   - SIP-прокси сервер — 192.168.116.161:5099
   - Включить регистрацию

- Сервер регистрации — 192.168.116.161:5099



4. Дополнительные параметры
   - SIP-домен — refactor
   - Применить SIP Domain для регистрации — включено

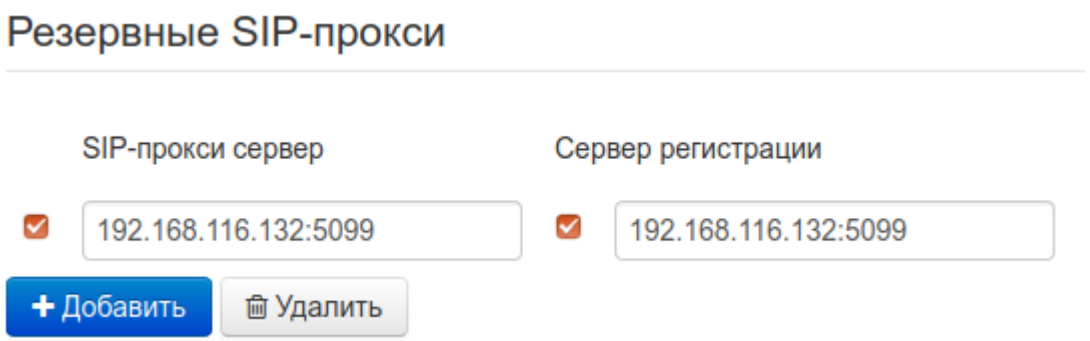


Основная регистрация на SMG, резервная на SSW

Для данного режима работы требуется добавить резервный SIP-прокси сервер.

Для этого во вкладке *IP-телефония — SIP-аккаунты — Аккаунт 1* добавляем резервный SIP-прокси сервер.

В поле *SIP-прокси сервер* и *Сервер регистрации* вводим IP и порт SSW — 192.168.116.132:5099.



При совершении исходящего вызова телефон отправляет сообщение INVITE на адрес основного SIP-прокси или при попытке регистрации — сообщение REGISTER. В случае если по истечении времени Invite total timeout от основного SIP-прокси не приходит ответ или приходит ответ 408 или 503 — телефон отправляет INVITE (либо REGISTER) на адрес первого резервного SIP-прокси.

Регистрация по доменному имени, где IP первый адрес ресолвится в SMG, второй и третий в SSW

Вместо IP адресов можно использовать DNS имя. Если DNS имя ресолвится в 3 различный IP адреса, то VP будет использовать первый в списке. В случае недоступности узла с данным IP адресом, будет использован следующий по списку.
В данном случае пример настройки будет выглядеть так:

Параметры SIP

Режим использования SIP-прокси	Homing
SIP-прокси сервер	ssw.eltex.loc
Регистрация	☑
Сервер регистрации	ssw.eltex.loc
Метод контроля основного сервера	Invite
Транспорт	UDP (предпочтительно), TCP
Таймер T1, мс	500
Таймер T2, мс	4000
Таймер INVITE транзакции (таймер B), мс	32000
Подписка для MWI	☐
Сервер подписок	ssw.eltex.loc