«Синхроком»

Система протоколов взаимодействия по сети Ethernet

Описание протокола

Синхроком-Транспорт (SCTP)

Аннотация

Настоящий документ описывает транспортный протокол стека протоколов «Синхроком» под названием Синхроком-Транспорт (SCTP).

Синхроком-Транспорт (SCTP) - двухфазный протокол передачи данных по сети SCA, где:

- синхронная фаза SCTPS - фаза передачи данных в режиме жесткого реального времени;
- асинхронная фаза SCTPA - фаза передачи данных в режиме мягкого реального времени.

Оглавление

Введение

Протокол управления передачей Синхроком-Транспорт (SCTP) - предназначен для использования в качестве надежного протокола общения между хост-компьютерами в коммуникационных компьютерных сетях SCA. Данный документ описывает функции, которые должны выполняться протоколом управления передачей, программу, которая реализует протокол, а также ее интерфейс с программами или пользователями, нуждающимися в ее услугах. Синхроком-Транспорт (SCTP) - двухфазный протокол передачи данных по сети SCA, где:

- синхронная фаза SCTPS - фаза передачи данных в режиме жесткого реального времени;
- асинхронная фаза SCTPA - фаза передачи данных в режиме мягкого реального времени

1.1 Мотивация

SCTP - это протокол обеспечения надежности прямых соединений, созданный для многоуровневой иерархии протоколов, поддерживающих межсетевые приложения. Протокол SCTP обеспечивает надежность коммуникаций между парами процессов на хост-компьютерах, включенных в SCA компьютерные коммуникационные сети, которые объединены в единую систему.

В отношении надежности протоколов более низкого, чем SCTP, уровня сделаны весьма скромные запросы. SCTP предполагает, что он может получить простой, потенциально ненадежный сервис для своих датаграмм со стороны протоколов нижнего уровня.

1.2. Сфера действия протокола

Протокол SCTP занимает в многоуровневой архитектуре протоколов нишу непосредственно над протоколом SCA, который позволяет протоколу SCTP отправлять и получать сегменты информации переменной длины, заключенные в оболочку SCA датаграмм. SCA датаграмма предоставляет средства для адресации отправителя и получателя сегментов SCTP. Протокол SCA также осуществляет любую фрагментацию и сборку сегментов SCTP. Протокол SCA также осуществляет разграничение SCTP сегментов. Так что данная информация может быть передана напрямую через блок разделения сетей в CAN сеть.

Стек протоколов «Синхроком» относится к L3-L5 слоям модели OSI/ISO.

№	Уровни модели OSI/ISO	Уровни стека протоколов «Синхроком»
7	Прикладной	Прикладные службы
6	Представления	Синхроком-Дата (SCD) - протокол представления данных
5	Сеансовый	Синхроком-Сеанс (SCS)- протокол туннелирования. Приложения, неограниченные требованием жесткого реального времени, могут использовать различные существующие сетевые протоколы на базе IP, которые туннелируются в сети «Синхроком», например: UDP, RTP и

		т.д.
4	Транспортный	Синхроком-Транспорт (SCTP) - двухфазный протокол передачи данных по сети SCA, где: синхронная фаза SCTPS - фаза передачи данных в режиме жесткого реального времени; асинхронная фаза SCTPA - фаза передачи данных в режиме мягкого реального времени.
3	Сетевой	Синхроком-Адрес (SCA)
2	Канальный	IEEE 802.2 на платформе микроядра L4
1	Физический	

Рисунок 1 - Стек протоколов «Синхроком»

Большая часть этого документа написана в связи с реализациями SCTP протокола, которые вместе с протоколами более высокого уровня присутствуют на хост-компьютере.

Транспортный протокол SCTP, так же как и сетевой протокол SCA, реализуется только на уровне микроядра, тогда как сеансовый SCS - имеет архитектуру клиент сервер, где сервер реализован на уровне микроядра, а клиент - в качестве драйвера сетевого моста гипервизора гостевой ОС.

1.3 Цель

Протокол SCTP обязан обеспечить надежный сервис для коммуникаций между процессами для обеспечения взаимодействия устройств в строгом и нестрогом режимах реального времени. Протокол SCTP должен быть общим протоколом для коммуникаций между хост-компьютерами в бортовой сети.

1.4 Интерфейсы

Протокол SCTP взаимодействует с одной стороны с пользователем или прикладной программой, а с другой - с протоколом более низкого уровня, таким как протокол SCA.

Интерфейс между прикладным процессом и протоколом SCTP мы поясняем с приемлемой детализацией. Этот интерфейс состоит из набора вызовов, которые похожи на вызовы операционной системы, предоставляемые прикладному процессу для управления файлами. Например, в этом случае имеются вызовы для открытия и закрытия соединений, для отправки и получения данных на установленных соединениях в синхронных и асинхронных фазах. Предполагается также, что протокол SCTP сможет асинхронно взаимодействовать с прикладными программами. Хотя разработчикам SCTP протокола и предоставляется значительная свобода в создании интерфейсов, которые соответствуют свойствам конкретной операционной системы, все же от любой приемлемой реализации требуются некие обязательные минимальные функции интерфейса между протоколом SCTP и пользователем.

Интерфейс между протоколом SCTP и протоколами более низкого уровня задан в значительно меньшей степени, за исключением того, что должен существовать некий механизм, с помощью которого эти два уровня могут синхронно и асинхронно обмениваться информацией друг с другом. Обычно полагают, что протокол нижнего уровня задает данный интерфейс. Протокол SCTP спроектирован так, чтобы работать с весьма разнообразной средой объединенных компьютерных сетей. В данном документе предполагается, что протокол более низкого уровня - это SCA [2].

1.5 Действие

Как указывалось ранее, главной целью протокола SCTP является обеспечение надежного, безопасного сервиса для логических цепей или соединений между парами процессов. Чтобы обеспечить такой сервис, основываясь на менее надежных коммуникациях SCA, система должна иметь возможности для работы в следующих областях:

- базовая передача данных - протокол SCTP способен передавать непрерывные потоки октетов между своими клиентами в обоих направлениях, пакуя некое количество октетов в сегменты для передачи через системы SCA.
- достоверность - протокол SCTP должен иметь защиту от разрушения данных, потери, дублирования и нарушения очередности получения, вызываемых коммуникационной системой SCA.
- управление потоком
- разделение каналов
- работа с соединениями
- приоритет и безопасность

Основные действия протокола SCTP в каждой из этих областей описаны в следующих параграфах. Протокол SCTP для синхронной фазы имеет собственное название SCTPS и для асинхронной фазы SCTPA. Протоколы взаимодействуют друг с другом лишь в момент использования сетевого ресурса. SCTPS всегда безусловно прерывает работу SCTPA для выполнения требований работы в режиме жесткого реального времени. SCTPA имеет доступ «на чтение» к расписанию синхронных фаз SCTPS и соответственно подстраивает свои действия. В дальнейшем специально не будет акцентироваться внимание на факте, что SCTPS работает в синхронную фазу и прерывает асинхронную по своему расписанию, а SCTPA работает только в асинхронную фазу и может быть безусловно прервана синхронной фазой протокола SCTP - SCTPS.

1.6 Протокол SCTPA

1.6.1 Базовая передача данных в SCTPA

В общем случае протоколы SCTPA решают по своему усмотрению, когда производить блокировку и передачу данных, за исключением безусловного прерывания протоколом SCTPS.

Иногда пользователям бывает необходимо убедиться в том, что все данные, переданные ими протоколу SCTPA, уже отправлены. Для этой цели определена функция проталкивания (push).

Чтобы убедиться в том, что данные, отправленные протоколу SCTPA, действительно переданы, отправитель указывает, что их следует протолкнуть к получателю.

Проталкивание приводит к тому, что программы протокола SCTPA сразу осуществляют отправление и, соответственно, получение остающихся данных. Правильно осуществленное проталкивание может быть невидимо для получателя, а сама функция проталкивания может не иметь маркера границы записи.

1.6.2 Достоверность в SCTPA

Достоверность в протоколе STPA достигается присвоением очередного номера каждому передаваемому октету, а также требованием подтверждения (ACK) от программы SCTPA, принимающей данные. Если подтверждения не получено в течении контрольного интервала времени, то данные посылаются повторно. Со стороны получателя номера очереди используются для восстановления очередности сегментов, которые могут быть получены в неправильном порядке, а также для ограничения возможности появления дубликатов.

Повреждения фиксируются посредством добавления к каждому передаваемому сегменту контрольной суммы, проверки ее при получении и последующей ликвидации дефектных сегментов.

До тех пор, пока программы протокола SCTPA продолжают функционировать корректно, а система SCA не развалилась полностью на составные части (например, в моменты неопределенности СУ), ошибки пересылки не будут влиять на правильное получение данных. Протокол SCTPA защищает от ошибок коммуникационной системы SCA.

1.6.3 Управление потоком в SCTPA

Протокол STPA дает средства получателю управлять количеством данных, посылаемых ему отправителем. Это достигается возвратом так называемого "окна" (window) вместе с каждым подтверждением, которое указывает диапазон приемлемых номеров, следующих за номером последнего успешно принятого сегмента. Окно определяет количество октетов, которое отправитель может послать до получения дальнейших указаний. Окно выделяется с учетом времени работы протокола SCTPS.

1.6.4 Разделение каналов в SCTPA

Чтобы позволить на отдельно взятом компьютере многим процессам одновременно использовать коммуникационные возможности уровня SCTPA, протокол SCTPA предоставляет на каждом узле набор портов, которые, вместе с адресами узлов на коммуникационном уровне SCA, образуют сокет («SCA адрес»:«SCTPA порт»).

Каждое асинхронное соединение SCTPA уникальным образом идентифицируется парой сокетов. Таким образом, любой сокет может одновременно использоваться во многих соединениях.

Соотнесение портов и процессов осуществляется каждым узлом самостоятельно. Однако оказывается полезным связывать часто используемые процессы (процессы систем и подсистем) с фиксированными документированными сокетами.

Этот сервис можно впоследствии использовать через известные адреса. Установка и настройка адресов портов для других процессов может включать более динамичные механизмы.

1.6.5 Работа с соединениями в SCTPA

Механизмы управления потоком и обеспечения достоверности, описанные выше, требуют, чтобы программы протокола SCTPA инициализировали и поддерживали определенную информацию о состоянии каждого потока данных. Набор такой информации, включающий сокеты, номера очереди, размеры окон, называется соединением. Каждое соединение уникальным образом идентифицируется парой сокетов на двух концах.

Если два процесса желают обмениваться информацией, соответствующие программы протокола SCTPA должны сначала установить соединение (на каждой стороне инициализировать информацию о статусе). По завершении обмена информацией соединение должно быть расторгнуто или закрыто, чтобы освободить ресурсы для предоставления другим пользователям.

Поскольку соединения должны устанавливаться между ненадежными узлами и через ненадежную коммуникационную систему SCA, то во избежание ошибочной инициализации соединений используется механизм подтверждения связи с хронометрированными номерами очереди.

1.6.6 Приоритет и безопасность в SCTPA

Пользователи протокола SCTPA могут затребовать для своего соединения приоритет и безопасность. Предусмотрены принимаемые по умолчанию характеристики соединений, когда такие параметры не требуются.

1.7 Протокол SCTPS

1.7.1 Базовая передача данных в SCTPS

Протокол SCTPA способен передавать непрерывные потоки октетов между своими клиентами в обоих направлениях, пакуя некое количество октетов в сегменты для передачи через системы SCA. Передача по протокол SCTPS управляется одним на сеть синхронизирующим узлом СУ.

В начале каждой изохронной фазы, СУ передает широковещательное сообщение Старт цикла для синхронизации всех узлов в сети. После завершения изохронной передачи данных, СУ передает широковещательное сообщение старт асинхронной фазы.

Во время изохронной фазы цикла, СУ производит опрос всех узлов сообщением ИЗ. В каждом цикле (или в каждом N-ом цикле для мультиплексных ПУ) пересылается одна пара сообщений на каждый узел. Дополнительно в цикле может быть одна широковещательная Изохронная посылка от СУ.

1.7.2 Достоверность в SCTPS

Вся передача данных в изохронной фазе SCTP происходит без подтверждения, однако не стоит забывать, что протокол SCA контролирует доставку пакетов в сети, поэтому особые случаи контроля работы SCTPS могут быть реализованы на уровне SCA.

1.7.3 Управление потоком в SCTPS

В начале цикла, в SCTPS СУ посылает сообщение «Начало цикла» широковещательно всем узлам. Время посылки и получения этого сообщения является основой для синхронизации всех узлов, остановки передачи данных по SCTPA. Сообщение «Начало цикла» (SoC - Start of Cycle) порождается на периодической основе и используется для синхронизации узлов.

Обмен данными между подчиненными узлами ПУ и СУ происходит после SOC и по завершению в сеть отправляется широковещательное сообщение «Начало асинхронной фазы» (SoA - Start of Asynchronous).

Сообщение «Изохронный запрос» посылается каждому сконфигурированному и активном ПУ. Опрашиваемый ПУ должен ответить сообщением «Изохронный ответ». Изохронный запрос является адресным сообщением и получается только узлом, которому адресован. SCTPS посылает в одном сообщении «Изохронный запрос» данные, предназначенные только одному ПУ. «Изохронный запрос» предназначен для передачи данных только тому узлу, которому он адресован. Наоборот, «Изохронный ответ» может быть получен всеми узлами. Процедура запрос-ответ должна повторяться для каждого сконфигурированного и активного изохронного ПУ. Порядок опроса ПУ доступен для конфигурации. Приложение каждого узла должно подготовить необходимые для передачи «Изохронного ответа» данные сразу после получения SoC.

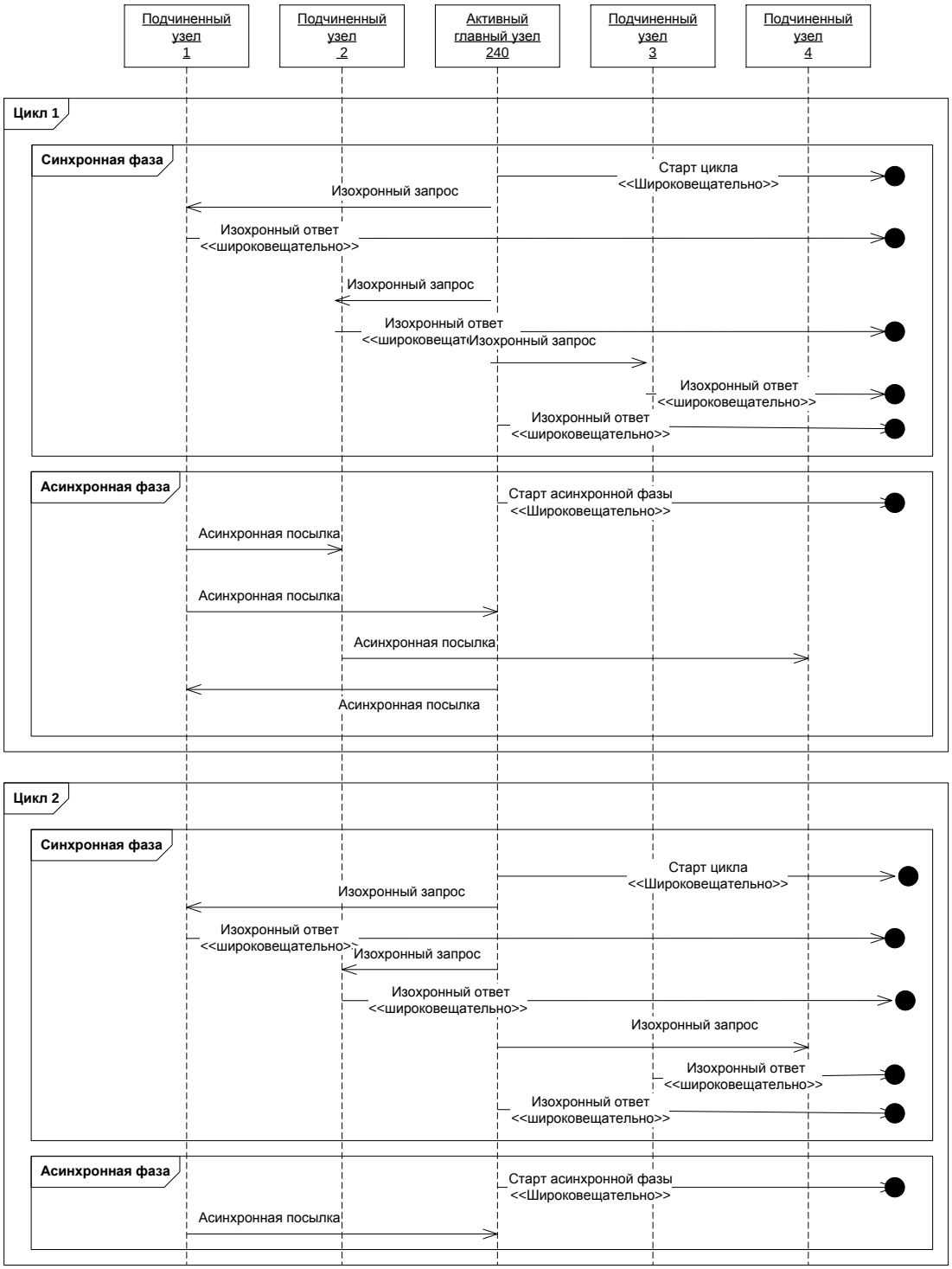

Рисунок 2. Общая структура цикла.

1.7.4 Разделение каналов в SCTPS

Протокол SCTPS работает на одном канале.

1.7.5 Работа с соединениями в SCTPS

Для того чтобы между СУ и ПУ было установлено SCTPS соединение адрес ПУ должен быть зарегистрирован в СУ. Способ реализации не оговаривается SCTPS. Очевидно, есть два способа:

- начальная конфигурация СУ;
- динамическая регистрация ПУ при инициализации сети.

Узел (как ПУ, так и АСУ) в SCTPS считается потерянным после неполучения от него данных заданного количества синхронных циклов. На ответ активному синхронизирующему узлу подчиненным узлом отводится строго определенный (по началу и продолжительности) интервал времени. Если обмен не произошел в заданный интервал, то фаза синхронного обмена АСУ с ПУ считается пропущенной.

1.7.6 Приоритет и безопасность в SCTPS

Синхронизирующие узлы в протоколе SCTPS являются подчиненными узлами ровно до того времени, когда активный синхронизирующий узел АСУ не выходит из строя. Тогда роль АСУ назначается следующему по адресации SCA синхронизирующему узлу. Вопрос уменьшения адреса АСУ не специфицируется SCTPS(случай, когда АСУ имел адрес 240, затем соединение с ним было потеряно, АСУ был присвоен 241, а затем в сети появился 240).

2. Идеология протокола

2.1 Элементы системы объединенных сетей

Бортовая сеть состоит из узлов, включенных в сеть, которые в свою очередь соединяются друг с другом и блоков разделения сетей - шлюзов, соединяющих сети «Синхроком» с другими сетями, например CAN.

Реальными агентами, создающими и потребляющими сообщения, циркулирующие в сети, являются процессы. Протоколы различных уровней в сети, на БРС и на узлах поддерживают систему коммуникаций между процессами (IPC - Inter Process Communication), которая обеспечивает двунаправленный поток данных по логическим соединениям между портами процессов SCA, CAN...

Термин пакет используется здесь в общем случае для обозначения порции данных, участвующей в отдельном элементарном акте взаимодействия между сетью и соединенным с ней узлом.

С точки зрения коммуникационных сетей, узлы - это компьютеры, связанные с сетью и являющиеся отправителями и получателями пакетов. Процессы рассматриваются как активные элементы на узлах (согласно наиболее общему определению процессов как исполняющихся программ). Предполагается, что даже терминалы, файлы и другие устройства ввода-вывода взаимодействуют друг с другом посредством процессов. Таким образом, любые коммуникации рассматриваются как коммуникации между процессами.

Поскольку процесс может контролировать несколько коммуникационных потоков, ведущих от него к другому процессу (или другим процессам), то мы постулируем, что каждый процесс может иметь набор портов, через которые он общается с портами других процессов.

2.2 Модель действия

Процесс пересылает данные, вызывая программу протокола SCTP и передавая ей в качестве аргументов буферы с данными. Протокол SCTP пакует данные из этих буферов в сегменты, а затем вызывает модуль SCA для передачи каждого сегмента на программу протокола SCTP, являющуюся адресатом. Этот адресат в свою очередь помещает данные из сегмента в буферы получателя и затем оповещает своего клиента о прибытии предназначенных ему данных. Программы протокола SCTP помещают в сегменты контрольную информацию, которая затем используется ими для проверки очередности передачи данных.

Модель коммуникаций бортовой сети состоит в том, что с каждой программой протокола SCTP связан модуль протокола SCA, обеспечивающий ей интерфейс с локальной сетью. Данный модуль SCA помещает сегменты SCTP в SCA датаграммы, а затем направляет их на другой SCA модуль или же БРС. Для передачи датаграммы по локальной сети она в свою очередь помещается в пакет соответствующего типа: SCA, SCTPA, SCTPS.

БРС являются коммутатором пакетов, которые могут осуществлять упаковку, фрагментацию или другие операции с тем, чтобы в CAN сети осуществить передачу пакетов по назначению с/на модуль SCA.

На БРС между сетями датаграмма освобождается от пакета сети и исследуется с тем, чтобы определить, по какой сети она должна идти, затем датаграмма упаковывается в пакет, соответствующий выбранной сети, и посылается конечному получателю.

БРС является шлюзом и имеет возможность разбивать датаграмму на более мелкие датаграммы фрагменты, если это необходимо для передачи в конкретной сети. Чтобы осуществить это, шлюз сам создает набор датаграмм, помещая в каждую по одному фрагменту. В дальнейшем фрагменты могут быть снова разбиты следующими шлюзами на еще более мелкие части. Формат фрагмента SCA датаграммы спроектирован так, чтобы адресат - модуль SCA смог собрать фрагменты снова в исходные SCA датаграммы.

SCA модуль, являющийся адресатом, выделяет сегмент из датаграммы (после ее сборки в случае необходимости) и затем передает его по назначению на программу протокола SCTP.

2.3 Программное обеспечение узла

Программа протокола SCTP является модулем микроядра L4. Клиенты обращаются к протоколу SCTP в значительной степени так же, как если бы они обращались к файловой системе. Сам протокол SCTP может обращаться к другим функциям операционной системы, к примеру, для управления структурами данных. Предполагается, что собственно интерфейс с локальной сетью осуществляется драйвером устройства. Протокол SCTP не обращается непосредственно к драйверам сетевых устройств, а вместо этого делает вызов для модуля SCA протокола, который в свою очередь и обращается к драйверу устройства.

2.4 Интерфейсы

Для запросов со стороны пользователя к протоколу SCTP интерфейс SCTP /пользователь обеспечивает:

- открытие и закрытие соединения;
- посылку и получение данных;
- получение статуса соединения;
- конфигурация узла.

Эти запросы похожи на другие запросы программы пользователя к операционной системе, например, на запросы открытия, чтения и закрытия файла.

Интерфейс между протоколами SCTP и SCA поддерживает запросы на посылку и получение датаграмм, адресованных на модули SCTP в узлах в любом месте сети SCA. Рассматриваемые запросы имеют аргументы для указания адреса, типа сообщения, приоритета, безопасности, а также передачи другой управляющей информации.

2.5 Связь с другими протоколами

Нижеприведенная диаграмма иллюстрирует место протокола SCTP в иерархии протоколов.

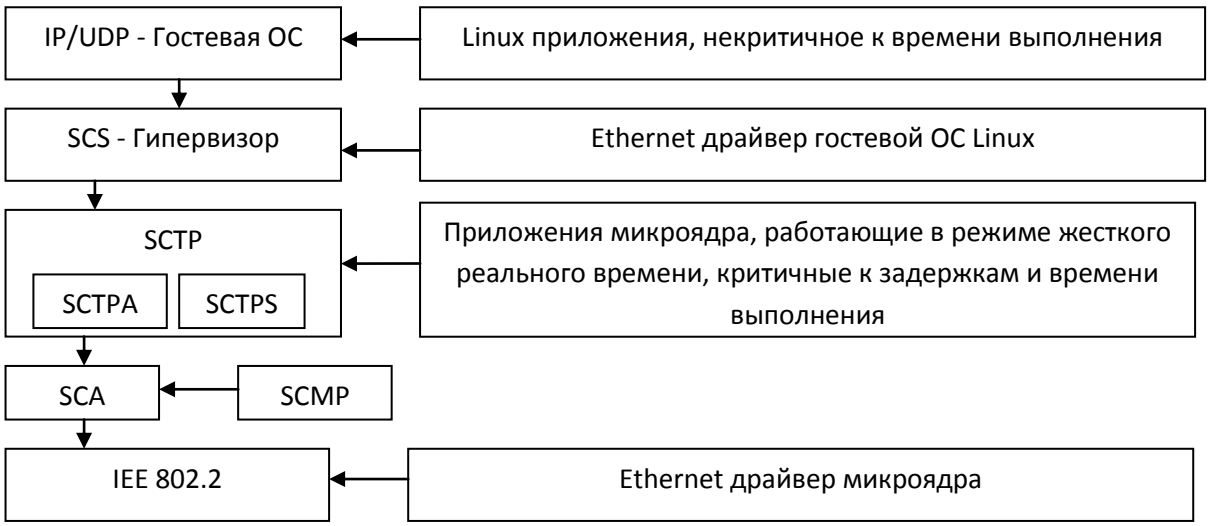

Рисунок 3 - Стек протоколов «Синхроком»

2.6 Надежные коммуникации

Поток данных, посылаемый на SCTP соединение, принимается получателем надежно и в соответствующей фазе и очередности.

Для SCTPS передача данных осуществляется без разбиения (на уровне SCTP, в то время как фрагментация на уровне SCA может иметь место). Надежность SCTPS гарантируется надежностью SCA и дополнительные мероприятия по надежности не предпринимаются.

Для SCTPA надежность передачи может основываться как и в SCTPS на SCA, а также могут дополнительно использоваться подтверждения и номера очередей.

Концептуально каждому октету данных присваивается номер очереди. Номер очереди для первого октета данных в сегменте передается вместе с этим сегментом и называется номером

очереди для сегмента. Сегменты также несут номер подтверждения, который является номером для следующего ожидаемого октета данных, передаваемого в обратном направлении. Когда протокол SCTPA передает сегмент с данными, он помещает его копию в очередь повторной передачи и запускает таймер. Когда приходит подтверждение для этих данных, соответствующий сегмент удаляется из очереди. Если подтверждение не приходит до истечения срока, то сегмент посылается повторно.

Подтверждение протокола SCTPA не гарантирует, что данные достигли конечного получателя, а только то, что программа протокола SCTPA на компьютере у получателя берет на себя ответственность за это.

Для направления потока данных между программами протоколов SCTPA используется механизм управления потоками. Получающая программа протокола SCTPA сообщает "окно" посылающей программе. Данное окно указывает количество октетов (начиная с номера подтверждения), которое принимающая программа SCTPA готова в настоящий момент принять. Окно выделяется с учетом интервалов синхронных фаз SCTPS и пропускной возможностью сети.

2.7 Установка соединения и его отмена для SCTP

Диаграмма состояний и переходов подчиненного узла приведена на рис. 3 и 4.

Описание состояний:

Рестарт узла - Временное состояние, в которое приходит узел после включения питания. После выполнения действия инициализации (загрузка микроядра), узел самостоятельно переходит в состояние **Рестарт приложения**.

Рестарт приложения - узел выполняет инициализацию модуля приложения и/или, при наличии, рестарт гостевой ОС, задавая параметры по-умолчанию для всех служб и переходит в состояние **Рестарт обмена**.

Рестарт обмена - выполняется конфигурация модуля обмена с параметрами по-умолчанию.

Готов - узел выполняет обмен по протоколу SCA. Это состояние применяется для настройки и технического обслуживания узла.

Работа - состояние нормальной работы узла, с сетевым обменом данными по протоколам SCA, SCTP.

Асинхронный обмен - узел выполняет обмен по протоколу SCTPA.

Изохронный обмен - узел выполняет обмен по протоколу SCTPS.

Останов - узел работает в пассивном режиме. В этом состоянии служба SCA узла работает, но не участвует в обмене, за исключением Запроса состояния и команд Запустить узел, Перейти в готовность, Сброс связи, Сброс приложения и Сброс узла, по которым переходит в состояния Работа, Готов, Рестарт связи, Рестарт приложения и Рестарт узла соответственно.

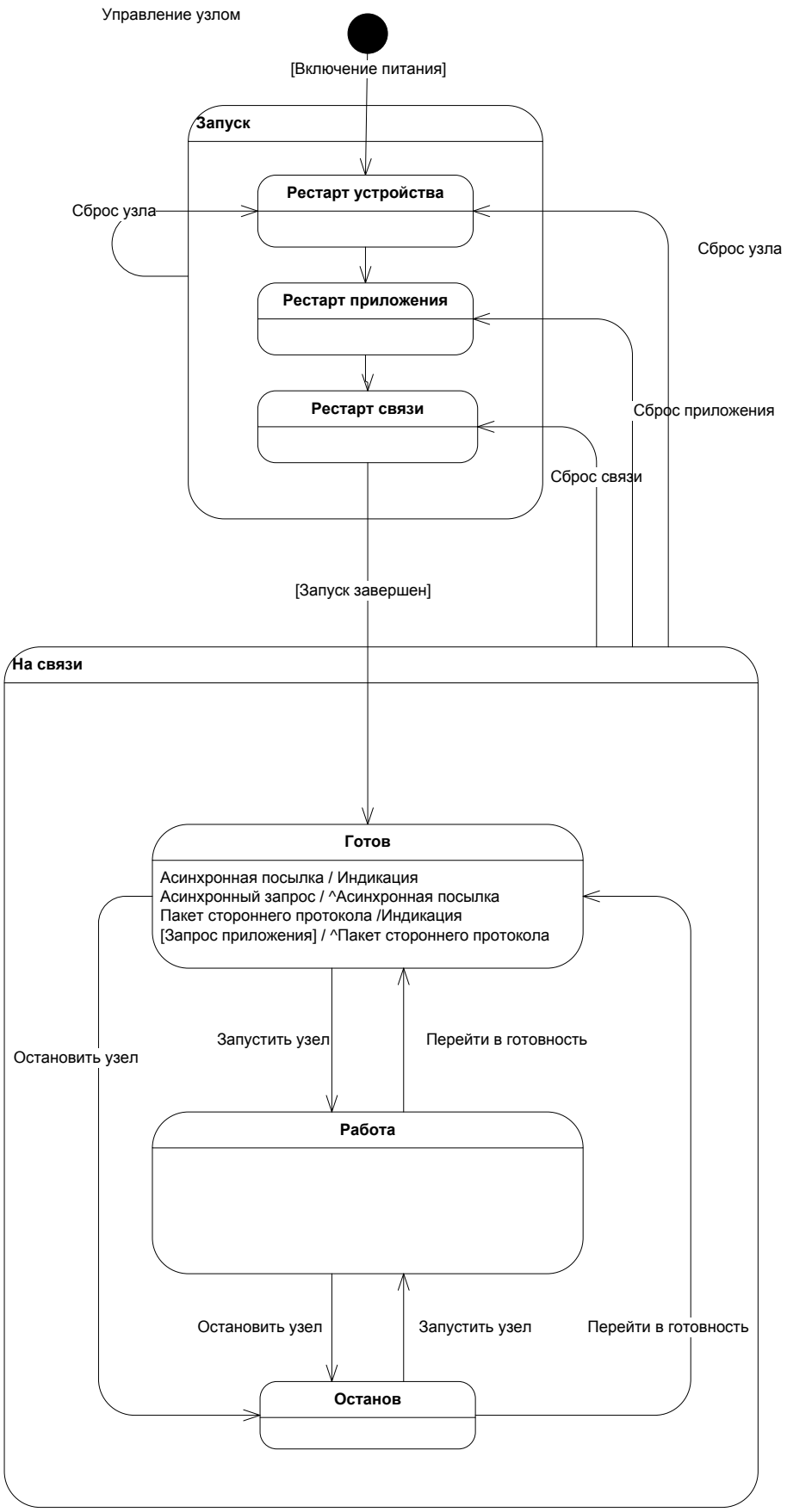

Рисунок 4. Диаграмма состояний управления циклом подчиненного узла

Работа

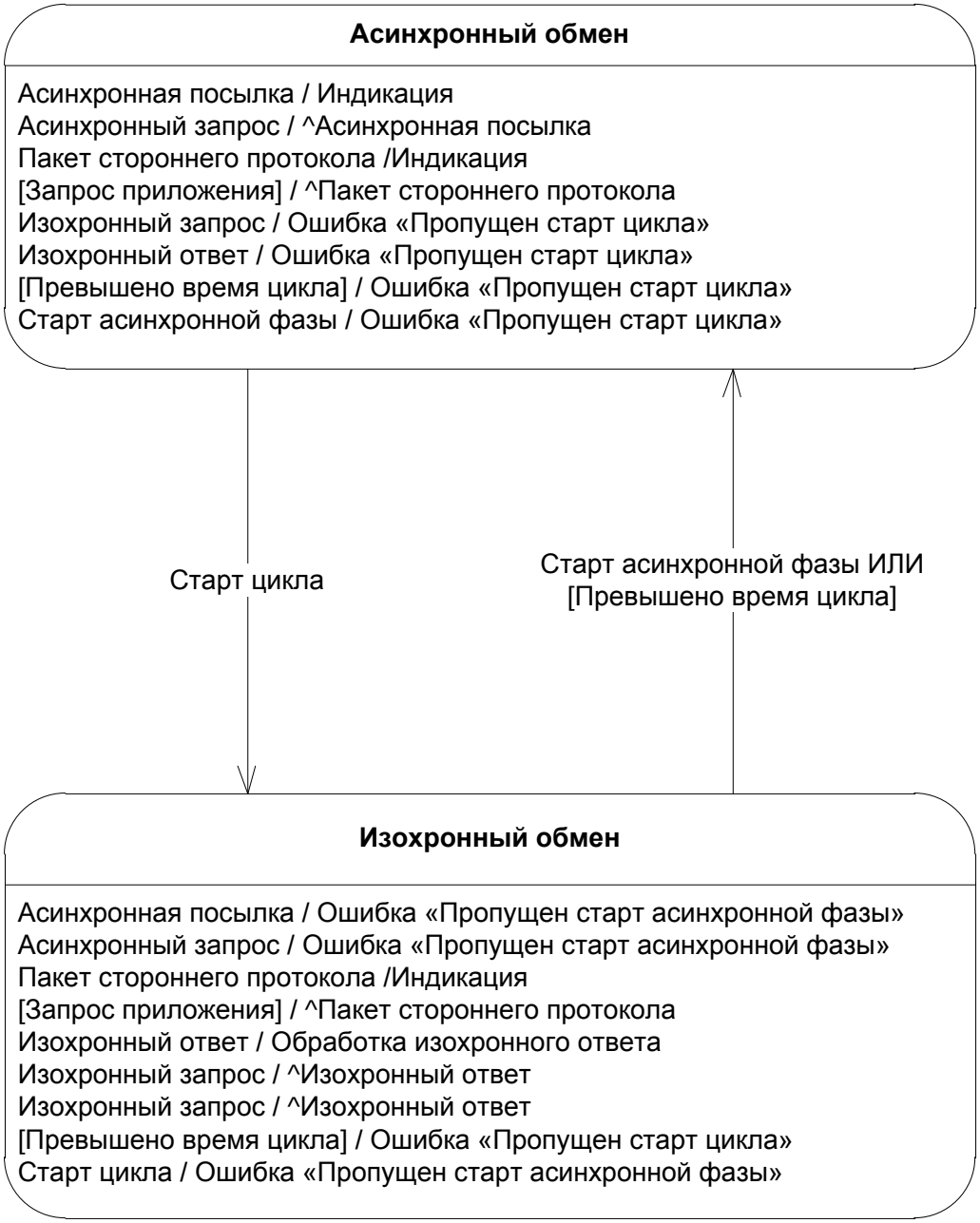

Рисунок 5. Диаграмма состояний управления сетью подчиненного узла.

2.7.1 Особенности установки соединения и его отмены для SCTPA

Чтобы идентифицировать отдельные потоки данных, поддерживаемые протоколом SCTPA, последний определяет идентификаторы портов. Поскольку идентификаторы портов выбираются каждой программой протокола SCTPA независимо, то они не будут уникальны. Чтобы обеспечить уникальность адресов для каждой программы протокола SCTPA, протокол объединяет

идентифицирующий эту программу SCA адрес и идентификатор порта - сокет, который будет уникален в бортовой сети.

Соединение полностью определяется парой сокетов на своих концах. Локальный сокет может принимать участие во многих соединениях с различными чужими сокетами. Соединение можно использовать для передачи данных в обоих направлениях, иными словами, оно является "полностью дуплексным".

Протокол SCTPA волен произвольным образом связывать порты с процессами. Однако при любой реализации протокола необходимо придерживаться нескольких основополагающих концепций. Должны присутствовать общеизвестные сокеты, которые протокол SCTPA ассоциирует исключительно с "соответствующими им" процессами. Мы представляем себе, как будто процессы могут "владеть" портами и что процессы могут инициировать соединения только с тех портов, которыми они владеют. (С точки зрения реализации протокола "владение" ограничивается узлом, однако можно представить себе команду пользователя по запросу порта (Request Port) или же метод выделения группы уникальных портов данному процессу, например посредством ассоциирования старших байтов в имени порта с данным процессом).

Соединение задается командой OPEN (открыть), сделанной с локального порта и имеющей аргументом чужой сокет. В ответ на такой запрос программа протокола SCTPA предоставляет имя локального (короткого) соединения. По этому имени пользователь адресуется к данному соединению при последующих вызовах.

Предполагается, что имеется некая структура данных, называемая блоком управления асинхронной передачей (Asynchronous Transmission Control Block -ATCB), предназначенная для сохранения описанной выше информации. Можно было бы реализовать протокол таким образом, чтобы локальное имя для соединения было бы указателем на структуру ATCB последнего. Запрос OPEN указывает также, осуществляется ли соединение активным образом, или же происходит пассивное ожидание соединения извне.

Запрос на пассивное открытие соединения означает, что процесс ждет получения извне запросов на соединение, вместо того, чтобы пытаться самому установить его. Часто процесс, сделавший запрос на пассивное открытие, будет принимать запросы на соединение от любого другого процесса. В этом случае чужой сокет указывается как состоящий целиком из нулей, что означает неопределенность. Неопределенные чужие сокеты могут присутствовать лишь в командах пассивного открытия.

Сервисный процесс, желающий обслужить другие, неизвестные ему процессы, мог бы осуществить запрос на пассивное открытие с указанием неопределенного сокета. В этом случае соединение может быть установлено с любым процессом, запросившим соединения с этим локальным сокетом. Такая процедура будет полезна, если известно, что выбранный локальный сокет ассоциирован с определенным сервисом.

Общеизвестные сокеты представляют собой удобный механизм априорного привязывания адреса сокета с каким-либо стандартным сервисом. Концепция общеизвестного сокета является частью спецификаций протоколов информационного обмена конкретных систем и подсистем и,

соответственно асоциирование сокетов с услугами выходит за рамки данного описания протокола.

Процессы могут осуществлять пассивные открытия соединений и ждать, пока от других процессов придут соответствующие запросы на активное открытие, а протокол SCTPA проинформирует их об установлении соединения. Два процесса, сделавшие друг другу одновременно запросы на активное открытие, получат корректное соединение. Гибкость такого подхода становится критичной при поддержке распределенных вычислений, когда компоненты системы взаимодействуют друг с другом асинхронным образом.

Когда осуществляется подбор сокетов для локального запроса пассивного открытия и чужого запроса на активное открытие, то принципиальное значение имеют два случая. В первом случае местное пассивное открытие полностью определяет чужой сокет. При этом подбор должен осуществляться очень аккуратно. Во втором случае во время местного пассивного открытия чужой сокет не указывается. Тогда в принципе может быть установлено соединение с любых чужих сокетов. Во всех остальных случаях подбор сокетов имеет частичные ограничения.

Если на один и тот же местный сокет осуществлено несколько ждущих пассивных запросов на открытие (записанных в блоки TCB), и осуществляется извне активный запрос на открытие, то чужой активный сокет будет связываться с тем блоком TCB, где было указание именно на этот запросивший соединения сокет. И только если такого блока TCB не существует, выбор партнера осуществляется среди блоков TCB с неопределенным чужим сокетом.

Процедура установки соединения использует флаг управления синхронизацией (SYN).

Соединение инициируется при встрече пришедшего сегмента, несущего флаг синхронизации (SYN), и ждущей его записи в блоке TCB. И сегмент и запись создаются пришедшими от пользователей запросами на открытие. Соответствие местного и чужого сокетов устанавливается при инициализации соединения. Соединение признается установленным, когда номера очередей синхронизированы в обоих направлениях между сокетами.

Отмена соединения также включает обмен сегментами, несущими на этот раз управляющий флаг FIN.

Диаграмма состояний для SCTPA требует введения дополнительной терминологии, что будет сделано в п.3.4 и там же приведена на Рисунок 11.

В п.3.6 рассматривается более подробно процедура установления соединения.

2.7.2 Особенности установки соединения и его отмена для SCTPS

Протокол SCTPS также содержит блок управления синхронной передачей (Synchronous Transmission Control Block - STCB). Все узлы, участвующие в синхронном обмене данными в режиме строгого реального времени бортовой сети заранее известны, поэтому могут быть сгененрированы, быть статичными, а значит количество записей в STCB также будет константным. STCB отслеживает доступность, работоспособность ПУ по заданному таймауту. В SCTPS соединение всегда остается открытым и не может быть закрыто по своей структуре, т.е. запись адреса ПУ ни при каких условиях не может быть удалена или вновь добавлена в STCB в режиме работы протокола. При потере связи АСУ с ПУ возможны различные реализации сценариев

повторного обращения к утерянному узлу, что не определяется протоколом и отдается на реализацию пользователю.

2.8 Коммуникация данных

Набор данных, передаваемых по соединению, можно рассматривать как поток октетов. Пользователь, отправляющий данные по SCTPA, указывает при запросе по посылку, следует ли данные, отправляемые при этом запросе, немедленно проталкивать через сеть к получателю. Указание осуществляется установкой флага PUSH (проталкивание). Для SCTPS понятие проталкивания отсутствует.

Программа протокола SCTPA может собирать данные, принимаемые от пользователя, а затем передавать их в сеть по своему усмотрению в виде сегментов. Если же выставлен запрос на проталкивание, то протокол должен передать все не отправленные ранее данные. Когда программа протокола SCTPA, принимающая данные, сталкивается с флагом проталкивания, ей не следует ожидать получения новых данных по сети до тех пор, пока уже имеющиеся данные не будут переданы ждущему их местному процессу.

Нет нужды привязывать функции проталкивания к границам сегмента. Данные, содержащиеся в каком-либо сегменте, могут быть результатом одного или нескольких запросов на посылку. Или же один запрос может породить несколько сегментов.

Целью функции проталкивания и флага PUSH является проталкивание данных через сеть от отправителя к получателю. Функция не осуществляет обработки самих данных.

Существует связь между функцией проталкивания и использованием буферов данных в интерфейсе между пользователем и протоколом SCTPA. Каждый раз, когда в буфер получателя приходят данные с флагом PUSH, содержимое этого буфера передается пользователю на обработку, даже если буфер и не был заполнен. Если приходящие данные заполняют буфер пользователя до того, как получена команда проталкивания, пользователю отправляется блок данных, соответствующий размеру буфера. Протокол SCTPA имеет также средства для сообщения получателю, что с некоторого момента он имеет дело со срочными данными. Протокол SCTPA не пытается определить, что именно пользователь делает со ждущими обработки срочными данными. Однако предполагается, что получающий данные процесс будет предпринимать усилия для быстрой обработки срочных данных, конкретная реализация данного процесса протоколом SCTPA не определяется.

3 Спецификация для функций протокола

Передача SCTP сегментов осуществляется в виде SCA датаграмм. Заголовок датаграммы SCTP следует за SCA заголовком и дополняет его информацией, специфической для SCTP протокола. Такое деление не ограничивает узел в расширении набора поддерживаемых протоколов на базе SCA.

3.1 Формат SCTP заголовка

0	1	2	3

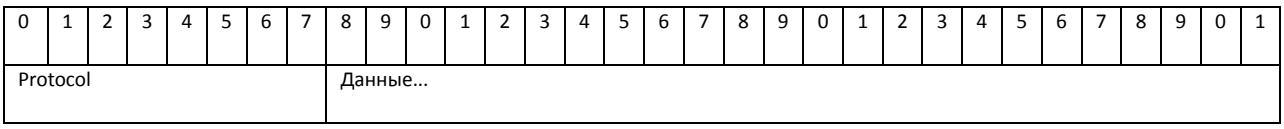

Рисунок 6. Формат SCTP заголовка

Protocol - тип протокола SCTP:

- 0 - SCTPS;
- 1 - SCTPA.

3.2 Формат SCTPA заголовка

Рисунок 7. Формат SCTPA заголовка

3.2.1 Контрольные биты

Размер поля: 8 бит.

- URG - задействовано поле срочного указателя
- ACK - задействовано поле подтверждения
- PSH - функция проталкивания
- RST - перезагрузка данного соединения
- SYN - синхронизация номеров очереди
- FIN - нет больше данных для передачи
- Два бита в резерве

3.2.2 Окно

Размер: 16 бит.

Количество октетов данных, начиная с октета, чей номер указан в поле подтверждения.
Количество октетов, получения которых ждет отправитель настоящего сегмента.

3.2.3 Порт отправителя

Размер: 16 бит.

Номер порта отправителя

3.2.4 Порт получателя

Размер: 16 бит.

Номер порта получателя.

3.2.5 Номер очереди - 32 бит.

Размер: 32 бит.

Номер очереди для первого октета данных в данном сегменте (за исключением тех случаев, когда присутствует флаг синхронизации SYN). Если же флаг SYN присутствует, то номер очереди является инициализационным (ISN), а номер первого октета данных - ISN+1.

3.2.6 Номер подтверждения

Размер: 32 бит.

Если установлен контрольный бит ACK, то это поле содержит следующий номер очереди, который отправитель данной датаграммы желает получить в обратном направлении. Номера подтверждения посылаются постоянно, как только соединение будет установлено.

3.2.7 Контрольная сумма

Размер: 16 бит.

Поле контрольной суммы - это 16-битное дополнение суммы всех 16- битных слов заголовка и текста. Если сегмент содержит в заголовке и тексте нечетное количество октетов, подлежащих учету в контрольной сумме, последний октет будет дополнен нулями справа с тем, чтобы образовать для предоставления контрольной сумме 16-битное слово. Возникший при таком выравнивании октет не передается вместе с сегментом по сети. Перед вычислением контрольной суммы поле этой суммы заполняется нулями.

3.2.8 Срочный указатель

Размер: 16 бит.

Это поле сообщает текущее значение срочного указателя. Последний является положительной величиной - смещением относительно номера очереди данного сегмента. Срочный указатель сообщает номер очереди для октета, следующего за срочными данными. Это поле интерпретируется только в том случае, когда в сегменте выставлен контрольный бит URG.

3.3 Формат SCTPS заголовка

0										1										2										3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1

Заголовок протокола SCTP	Тип	Резервные биты	Данные

Рисунок 8. Формат SCTPS заголовка

3.3.1 Тип

Размер: 2 бит.

Протокол SCTPS поддерживает 4 типа сообщений

1. SoC - Start of Cycle - Старт цикла
2. Req - Request - Запрос
3. Resp - Response - Ответ
4. SoA - Start of Asynchronous - Старт асинхронной фазы

3.3.2 Резервные биты

Размер: 6 бит.

Основное назначение - выравнивание. Возможно дальнейшее использование в качестве флагов, расширения типов сообщений, контрольной суммы, поля длины пакета в байтах и тому подобной служебной информации. Так или иначе в размере не ограничено и поэтому заголовок протокола может быть расширен.

3.4 Терминология

Рассмотренная в данном разделе терминология будет касаться SCTPA протокола, т.к. его реализация не такая тривиальная как SCTPS, концептуально представляющий собой удаленный синхронный вызов функции/метода и не требующий введения дополнительной терминологии.

Для поддержания SCTPA соединения необходимо иметь несколько переменных, которые помещены в соответствующую запись блока ATCB (см.2.7.1). Среди переменных блока ATCB имеются номера местного и чужого сокетов, флаги безопасности и приоритета для данного соединения, указатели буферов посылки и получения, указатели текущего сегмента и очереди повторной посылки. Кроме всего этого в ATCB имеются несколько переменных, имеющих отношение к номерам очередей отправителя и получателя.

3.4.1 Отправление

- SND.UNA посылка неподтверждена
- SND.NXT послать следующий сегмент
- SND.WND отправить окно
- SND.UP отправить срочный указатель
- SND.WL1 номер очереди сегмента, использованный для обновления последнего окна

- SND.WL2 номер подтверждения в сегменте, используемый для обновления последнего окна
- ISS первоначальный номер очереди отправления

3.4.2 Получение:

- RCV.NXT - получить следующий сегмент
- RCV.WND - получить окно
- RCV.UP - получить срочный указатель
- IRS - первоначальный номер очереди получения

3.4.3 Очередь отправления

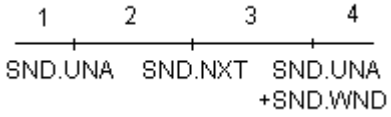

Рисунок 9. Очередь отправления

где:

1. старые номера очереди, которые получили подтверждение
2. номера очереди для данных, не получивших подтверждения
3. номера очереди, допущенные к новой передаче
4. следующие номера очереди, чья передача еще не разрешена

3.4.4 Очередь получения

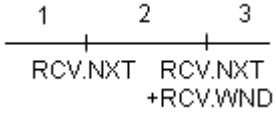

Рисунок 10. Очередь получения

, где:

1. старые номера очереди, которые получили подтверждение
2. номера очереди, допущенные к очередному этапу получения
3. следующие номера очереди, еще не получившие разрешения

3.4.5 Очередной сегмент

Переменные для очередного сегмента:

- SEG.SEQ номер очереди для сегмента
- SEG.ACK номер подтверждения для сегмента
- SEG.LEN длина сегмента
- SEG.WND окно для сегмента
- SEG.UP срочный указатель для сегмента
- SEG.PRC приоритет для сегмента

3.4.6 Диаграмма состояний SCTPA

Соединение во время функционирования проходит через серии промежуточных состояний. Это состояния LISTEN, SYN-SENT, SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT, а также фиктивное состояние CLOSED. Состояние CLOSED является фиктивным, поскольку оно представляет состояние, когда не существует блока SCTPA, а потому и нет соединения. Краткое описание состояний:

- **LISTEN** Ожидание запроса на соединение со стороны чужих портов и программ SCTPA
- **SYN-SENT** Ожидание парного запроса на установление соединения. С нашей стороны запрос уже сделан.
- **SYN-RECEIVED** Ожидание подтверждения после того, как запрос соединения уже принят и отправлен.
- **ESTABLISHED** Состояние открытого соединения, принимаемые данные можно представить пользователю. Это нормальное состояние соединения в фазе передачи данных.
- **FIN-WAIT-1** Ожидание запроса от чужой программы SCTPA, или подтверждения ранее отправленного запроса на закрытие соединения.
- **FIN-WAIT-2** Ожидание запроса на закрытие соединения со стороны чужой программы SCTPA.
- **CLOSE-WAIT** Ожидание запроса на закрытие соединения со стороны своего клиента.
- **CLOSING** Ожидание подтверждения со стороны чужой программы SCTPA запроса о закрытии соединения.
- **LAST-ACK** Ожидание запроса на закрытие соединения, ранее отправленного чужой программе SCTPA (запрос включал также подтверждение получения чужого запроса на закрытие соединения).
- **TIME-WAIT** Ожидание когда истечет достаточное количество времени и можно быть уверенным, что чужая программа SCTPA получила подтверждение своего запроса на закрытие соединения.

- **CLOSED** Состояние полного отсутствия соединения.

Соединение SCTPA переходит с одного состояния на другое в ответ на события. Событие - это запросы клиента (открытие, посылка, получение, закрытие, отказ, получение состояния соединения), приход сегментов, и особенно тех, которые содержат флаги SYN, ACK, RST и FIN, а также истечение выделенного времени.

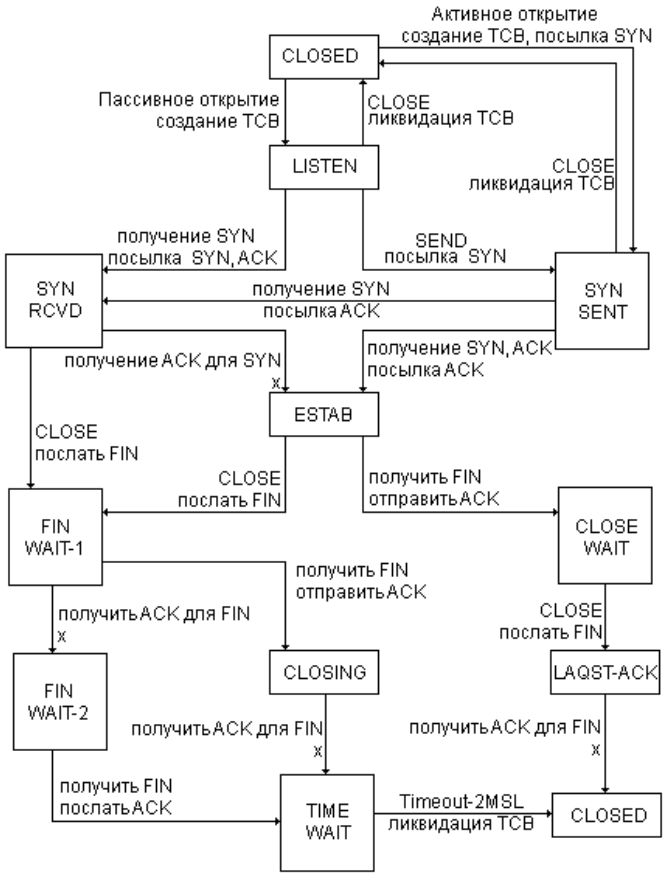

Рисунок 11. Диаграмма состояний SCTPA

3.5 Номер очереди

Каждый октет данных, посылаемый на SCTPA соединение, имеет номер очереди. Поскольку каждый октет пронумерован, то каждый из них может быть опознан. Приемлемый механизм опознания является накопительным, так что опознание номера X означает, что все октеты с предыдущими номерами уже получены. Этот механизм позволяет регистрировать появление дубликатов в условиях повторной передачи. Нумерация октетов в пределах сегмента осуществляется так, чтобы первый октет данных сразу вслед за заголовком имел наименьший номер, а следующие за ним октеты имели номера по возрастающей.

Протокол SCTPA должен осуществлять следующие типы сравнения для номеров очереди:

- является ли номер в подтверждении номером очереди для октетов, уже отправленных, но еще не получивших подтверждения;

- получили ли все октеты в сегменте подтверждение своих номеров (т.е. следует ли удалить данный сегмент из очереди на повторную посылку);
- содержит ли пришедший сегмент ожидаемые нами номера (т.е. "перекрывает" ли этот сегмент окно получателя).

В ответ на посылку данных протокол SCTPA будет получать их подтверждение. Для работы с полученным подтверждением необходимо уметь делать сравнение для:

- SND.UNA самого старого из номеров, не имевших подтверждения,
- SND.NXT следующего номера очереди, ждущего посылки,
- SEG.ACK номера подтверждения, полученного от чужой принимающей программы SCTPA (следующего номера очереди, ожидаемого чужой программой SCTPA),
- SEG.SEQ номера очереди первого октета в сегменте,
- SEG.LEN количества октетов в поле данных сегмента (учитывая SYN и FIN),
- SEG.SEQ+SEG.LEN-1 номера очереди последнего октета из сегмента.

Новое подтверждение (называемое "подтверждением приемлемости") - это подтверждение выполнимости неравенств

SND.UNA < SEG.ACK =< SND.NXT

Сегмент из очереди повторной посылки получает полное подтверждение, если сумма его номера в очереди и длины поля данных меньше или равна номеру подтверждения из пришедшего сегмента.

При получении данных необходимо производить операции сравнения для следующих величин:

- RCV.NXT следующий номер из очереди приходящих сегментов, а также левая или нижняя граница окна получения,
- RCV.NXT+RCV.WND-1 номер очереди последнего сегмента, ожидаемого в приходящем сегменте, а также правая или верхняя граница окна получения,
- SEG.SEQ первый номер в очереди, принесенный пришедшим сегментом,
- SEG.SEQ+SEG.LEN-1 последний номер в очереди, принесенный пришедшим сегментом.

Считается, что сегмент перекрывает часть разрешенных номеров в очереди получения, если

RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND

или

RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Первая часть этого текста смотрит, попадает ли начало сегмента на окно, а вторая часть - попадает ли в окно задняя часть сегмента. Если выполняется какая-либо часть теста, то сегмент попадает в окно.

Выбирая окно нулевой длины или сегмент нулевой длины, мы получаем четыре варианта проверки на приемлемость для приходящих сегментов:

ЕСЛИ SEG.LEN=0 И RCV.WND=0 ТО ПРОВЕРИТЬ SEG.SEQ = RCV.NXT

ЕСЛИ SEG.LEN=0 И RCV.WND>0 ТО ПРОВЕРИТЬ RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND

ЕСЛИ SEG.LEN>0 И RCV.WND=0 ТО ОШИБКА

ЕСЛИ SEG.LEN>0 И RCV.WND>0 ТО ПРОВЕРИТЬ

RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND

ИЛИ RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Для целей поддержания очередности, сигнал SYN рассматривается как стоящий перед первым действительным октетом данных в сегменте, куда оба они были помещены. В то же время FIN считается стоящим после последнего реального октета данных в сегменте. Длина сегмента (SEG.LEN) учитывает как данные, так и номера очереди, отведенные под управление. В случае, когда присутствует SYN, значение SEG.SEQ соответствует номеру в очереди для сигнала SYN.

3.5.1 Выбор начального значения очереди

Протокол не накладывает ограничения на многократное повторное использование конкретного соединения. Соединение задается подбором пары сегментов. Новые запросы на установление какого-либо соединения будут рассматриваться как повторные реализации этого соединения. Эта проблема становится явной, если соединение быстро открывается и закрывается несколько раз подряд, или же если соединение прерывает свою работу с потерей информации, хранившейся в оперативной памяти компьютера, и затем устанавливается повторно. Такие варианты в протоколе SCTP возможны, хотя и необходимо уделить должное внимание в реализации SCTPS протокола корректное прерывание SCTPA.

Чтобы избежать сбоя, необходимо избегать использования сегментов данной реализации соединения, когда в сети еще присутствуют те же самые номера очереди, оставшиеся от предыдущей реализации соединения. При создании новых соединений применяется генератор первоначальных номеров очереди (ISN), который выбирает новые 32 битные значения ISN. Генератор привязан к 32-битным часам. Необходимо, чтобы полный цикл часов ISN былне более максимального времени жизни сегмента (Maximum Segment Lifetime - MSL), тогда с основанием можно будет полагать, что номера ISN будут уникальны.

Для каждого соединения существует номер в очереди отправления и номер в очереди получения. Первоначальный номер в очереди отправления (ISS) выбирается программой SCTPA, посылающей данные в этой очереди, а первоначальный номер в очереди получения (IRS) выясняется во время установления соединения.

3.5.2 Синхронизация начальных значений очередей

Во время установления или инициализации какого-либо соединения обе программы протокола SCTPA должны синхронизировать друг с другом первоначальные номера очередей. Это осуществляется посредством обмена сегментами, устанавливающими соединения, несущими контрольный бит, называемый "SYN" (for synchronize - для синхронизации), несущими исходные номера для очередей. Для краткости, сегменты, несущие бит SYN, также называются SYN сегментами. Следовательно, решение проблемы требует приемлемого механизма для подбора первоначального номера очереди и немногочисленных сигналов подтверждения при обмене номерами ISN.

Синхронизация требует, чтобы каждая сторона, участвующая в соединении, посылала свой собственный первоначальный номер очереди, а также получала подтверждение на это от напарника. Каждая сторона должна также получить первоначальный номер очереди от напарника и послать подтверждение.

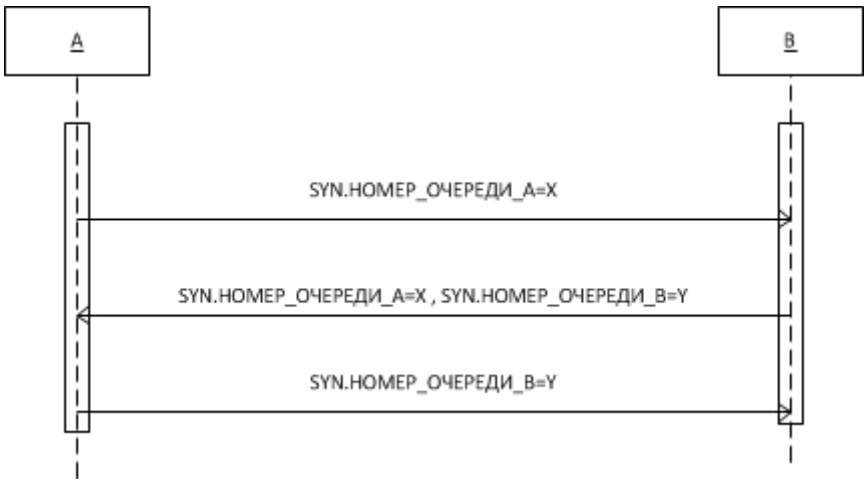

Рисунок 12. Диаграмма последовательности синхронизации начальных значений очередей двух SCTPA программ A и B.

Номера очереди не привязываются к неким глобальным часам данной компьютерной сети (SCTPA не является протоколом жесткого реального времени), и программы SCTPA могут иметь различные механизмы для подбора номеров ISN. Получатель первого сигнала SYN не может знать, задержался ли этот сигнал и уже устарел, или это не так, даже если получатель не помнит последний номер очереди, использованный этим соединением (что тоже не всегда возможно). Так что он должен попросить отправителя проверить этот сигнал SYN.

3.5.3 Период молчания

Чтобы быть уверенным в том, что программа SCTPA не создает сегмента, несущего номер очереди, который уже используется старым сегментом, все еще "ходящим" по сети, программа SCTPA должна сохранять молчание по крайней мере в течении времени жизни сегмента (MSL) до тех пор, пока она не назначит какие-либо номера очереди при запуске или восстановлении после сбоя, когда записи в памяти для прежних номеров из очереди были потеряны.

3.6 Установление соединения

Процедура синхронизации начальных значений очередей используется при установлении соединения.

Эта процедура обычно инициируется программой протокола SCTPA в ответ на запрос другой программы SCTPA. Данная процедура также работает, если две программы SCTPA инициируют ее одновременно. Когда попытка инициализации осуществляется с обоих концов одновременно, каждая программа протокола SCTPA получает сегмент "SYN", который не несет подтверждения для уже отправленного ею "SYN". Конечно, прибытие старых дубликатов сегмента "SYN" может произвести впечатление на получателя, будто осуществляется одновременное открытие соединения. Корректное применение сегментов "перезагрузки" может предотвратить двусмысленность таких ситуаций.

Ниже приведены несколько примеров инициализации соединений. Хотя эти примеры не показывают синхронизации соединения с помощью сегментов, несущих данные, это совершенно правомерно, поскольку программа SCTPA, получающая сегменты, не передаст данные своему клиенту, пока не станет очевидным корректность данных (т.е. данные должны "складироваться" пользователем до тех пор, пока соединение не перейдет в состояние ESTABLISHED).

3.6.1 Базовый случай

	SCTPA A				**SCTPA B**
1.	CLOSED				LISTEN
2.	SYN-SENT	-->	<SEQ=100><CTL=SYN>	-->	SYN-RECEIVED
3.	ESTABLISHED	<--	<SEQ=300><ACK=101><CTL=SYN,ACK>	<--	SYN-RECEIVED
4.	ESTABLISHED	-->	<SEQ=101><ACK=301><CTL=ACK>	-->	ESTABLISHED
5.	ESTABLISHED	-->	<SEQ=101><ACK=301><CTL=ACK><DATA>	-->	ESTABLISHED

Рисунок 13. Основная процедура синхронизации начальных значений очередей двух SCTPA программ A и B.

3.6.2 Одновременная синхронизация

На следующем рисунке 14 показана та же инициализация с незначительными усложнениями. Каждая программа SCTPA проходит по очереди состояния CLOSED, SYN-SENT, SYN-RECIEVED и наконец, ESTABLISHED.

	SCTPA A				**SCTPA B**
1.	CLOSED				CLOSED
2.	SYN-SENT	-->	<SEQ=100><CTL=SYN>	....	
3.	SYN-RECEIVED	<--	<SEQ=300><CTL=SYN>	<--	SYN-SENT

4.		....	<SEQ=101><ACK=301><CTL=ACK>	-->	SYN-RECEIVED
5.	SYN-RECEIVED	-->	<SEQ=100><ACK=301><CTL=SYN,ACK>	....	ESTABLISHED
6.	ESTABLISHED	<--	<SEQ=300><ACK=101><CTL=SYN,ACK>	<--	SYN-RECEIVED
7.		....	<SEQ=101><ACK=301><CTL=ACK>	-->	ESTABLISHED

Рисунок 14. Одновременная синхронизация соединения

3.6.3 Получение старого дубликата

Главной причиной для применения подтверждений является попытка предотвратить возникновение сбоев при получении старых дубликатов, инициирующих соединение. Для работы с подтверждениями в SCTPA протоколе существует специальное контрольное сообщение - перезагрузка (reset).

Если получающая сигнал программа SCTPA находится не в синхронизированном состоянии (т.е. в SYN-SENT, SYN-RECEIVED), то она возвращает сигнал LISTEN, показывая, что она получила приемлемый сигнал перезагрузки. Если же программа SCTPA находится в одном из синхронизированных состояний (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT), то она ликвидирует соединение и проинформирует об этом своего клиента. Мы обсудим ниже такую ситуацию при рассмотрении "наполовину открытых" соединений п.3.6.4.

В качестве простого примера рассмотрим ситуацию с получением старых дубликатов на рисунке 15.

	SCTPA A				**SCTPA B**
1.	CLOSED				LISTEN
2.	SYN-SENT	-->	<SEQ=100><CTL=SYN>	....	
3.	(дубликат)	....	<SEQ=90><CTL=SYN>	-->	SYN-RECEIVED
4.	SYN-SENT	<--	<SEQ=300><ACK=91><CTL=SYN,ACK>	<--	SYN-RECEIVED
5.	SYN-SENT	-->	<SEQ=91><CTL=RST>	-->	LISTEN
6.		....	<SEQ=100><CTL=SYN >	-->	SYN-RECEIVED
7.	SYN-SENT	<--	<SEQ=400><ACK=101><CTL=SYN,ACK>	<--	SYN-RECEIVED
8.	ESTABLISHED	-->	<SEQ=101><ACK=401><CTL=ACK>	-->	ESTABLISHED

Рисунок 15. Получение старого дубликата сигнала SYN

3.6.4 Наполовину открытое соединение

Уже установившееся соединение называется "наполовину открытым", если одна из программ SCTPA закрыла соединение, или отказалась от него. Причем сделала это на своем конце, не предупредив своего партнера. Также такая ситуация может возникнуть, если нарушена

синхронизация на концах линии вследствие сбоя, приведшего к потере информации в памяти. Если на таких соединениях делается попытка отправить данные в каком-либо направлении, то автоматически производится перезагрузка соединения.

Если на конце А соединение считается уже несуществующим, а клиент на конце В пытается послать данные, то это приведет к тому, что программа SCTPA на конце В получит контрольное сообщение о перезагрузке. Такое сообщение показывает программе SCTPA на конце В, что что-то неправильно и ей предлагается ликвидировать это соединение.

Предположим, что два клиента в точках А и В общаются друг с другом, и в этот момент происходит крах(или перезагрузка узла, или неудачное прерывание протоколом SCTPS), приводящий к неактуальности информации в памяти у программы SCTPA на конце А. В зависимости от операционной системы, обслуживающей программу SCTPA А, вероятно, будет задействован некий механизм исправления ошибки. Когда программа SCTPA А будет запущена вновь, она, вероятно, вновь начнет свою работу с самого начала или же с инструкции преодоления сбоя. В результате, программа А, вероятно, попытается открыть (OPEN) соединение или послать информацию (SEND) через соединение, которое, как она полагает, является открытым. В последнем случае от местной программы SCTPA (на конце А) будет получено сообщение "соединение не открыто". При попытке установить соединение программа SCTPA А будет посылать сегмент, содержащий сигнал SYN. Такой сценарий приводит к ситуации, показанной на рисунке 16. После того, как программа SCTPA А потерпит крах, пользователь попытается повторно открыть соединение. Программа SCTPA В тем временем продолжает полагать, будто соединение остается открытым.

	SCTPA A				**SCTPA B**
1.	сбой		(номер посылки - 300, получения - 100)		
2.	CLOSED				ESTABLISHED
3.	SYN-SENT	-->	<SEQ=400><CTL=SYN>	-->	(??)
4.	(!!)	<--	<SEQ=300><ACK=100><CTL=SYN>	<--	ESTABLISHED
5.	SYN-SENT	-->	<SEQ=100><CTL=RST>	-->	(ликвидация)
6.	SYN-SENT				CLOSED
7.	SYN-SENT	-->	<SEQ=400><CTL=SYN>	-->	

Рисунок 16. Обнаружение наполовину открытого соединения.

3.6.5 Активная сторона приводит к обнаружению наполовину открытого соединения

Возможен вариант, когда программа SCTPA А терпит крах, а программа SCTPA В, полагая, что находится в состоянии синхронизации, пытается послать данные. Эта ситуация показана на рисунке 17.

В этом случае данные, отправленные программой SCTPA В, и пришедшие на программу SCTPA А (строка 2). будут отвергнуты, поскольку используемого ими соединения не существует. На

основании этого программа SCTPA A посылает сигнал RST. Как только сигнал RST принят программой SCTPA B, он будет рассмотрен, а использованное прежде соединение будет ликвидировано.

	SCTPA A				**SCTPA B**
1.	сбой		(номер посылки - 300, получения - 100)		
2.	(??)	<--	<SEQ=300><ACK=100><DATA=10><CTL=SYN>	<--	ESTABLISHED
3.		-->	<SEQ=100><CTL=RST>	-->	(ликвидация)

Рисунок 17. Активная сторона приводит к обнаружению наполовину открытого соединения

3.6.6 Старый дубликат сигнала SYN

На рисунке 18 показано, что две программы SCTPA - A и B - ,имея пассивное состояние, ждут сигнала SYN. Старый дубликат сигнала, достигает программу SCTPA B (строка 2), запускает ее. Возвращается сигнал SYN-ACK (строка 3) и заставляет программу SCTPA A генерировать сигнал RST (на строке 3 сигнал ACK неприемлем). Программа SCTPA B принимает команду перезагрузки и возвращается в пассивное состояние LISTEN.

	SCTPA A				**SCTPA B**
1.	LISTEN				LISTEN
2.		...	<SEQ=Z><CTL=SYN>	-->	SYN-RECEIVED
3.	(??)	<--	<SEQ=X><ACK=Z+1><CTL=SYN,ACK>	<--	SYN-RECEIVED
4.		-->	<SEQ=Z+1><CTL=RST>	-->	возврат в LISTEN
5.	LISTEN				LISTEN

Рисунок 18. Старый дубликат сигнала SYN инициирует перезагрузку на двух пассивных сокетах

3.7 Сигнал перезагрузки

Сигнал перезагрузки (RST) должен посылаться всякий раз, когда приходит сегмент, который очевидным образом не предназначен для данного соединения, либо конфликтует с текущим состоянием. Если непонятно, имеет ли место данный случай, следует воздержаться от перезагрузки.

3.7.1 Группирование состояний по отношению к RST

Можно выделить три группы состояний для соединения:

1. Если соединения не существует (CLOSED), то сигнал перезагрузки посылается в ответ на любой пришедший сегмент, за исключением встречного сигнала перезагрузки. В частности, сигналы SYN, адресованные на несуществующее соединение, отвергаются именно таким образом. Если приходящий сегмент имеет флаг в поле ACK, то сегмент с

сигналом перезагрузки получает номер для очереди из поля ACK первого сегмента. В противном случае сегмент с сигналом перезагрузки имеет нулевой номер очереди и значение в поле ACK, равным сумме номера очереди пришедшего сегмента и его же длины. Соединение остается в состоянии CLOSED.

2. Если соединение находится в каком-либо не синхронизированном состоянии (LISTEN, SYN-SENT, SYN-RECEIVED), если какие-либо подтверждения пришедшего сегмента еще не отправлены (сегмент несет неприемлемое значение в поле ACK) или пришедший сегмент имеет уровень безопасности/закрытости не соответствующий уровню и защите данного соединения, то отправляется сигнал перезагрузки.
   Если наш сигнал SYN не был подтвержден, а уровень приоритета пришедшего сегмента больше запрошенного уровня, то либо будет увеличен местный уровень приоритета (если это приемлемо для пользователя и системы), либо будет послан сигнал перезагрузки. Или же если уровень приоритета пришедшего сегмента меньше запрошенного, то обработка будет продолжена далее, как если бы уровень был таким же (если чужая программа SCTPA не может повысить уровень приоритета до нашего, то это будет отмечено в следующем отправляемом ею сегменте, тогда и будет закрыто соединение). Если наш сигнал SYN получил подтверждение (возможно в пришедшем к нам сегменте), то уровень приоритета пришедшего сегмента должен точно соответствовать мест ному уровню. Если последнее условие не выполняется, посылается сигнал перезагрузки.
   Если приходящий сегмент несет сигнал ACK, то сигнал перезагрузки будет иметь номер в очереди, соответствующий номеру сигнала ACK в пришедшем сегменте. В противном случае сигнал перезагрузки будет иметь нулевой номер очереди, а сигнал ACK - номер, равный сумме номера пришедшего сегмента и его же длины. Соединение не меняет своего состояния.

3. Если соединение находится в синхронизированном состоянии (ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST ACK, TIME-WAIT), то любой неприемлемый сегмент (не попадающий в окно номеров очереди, несущий неправильный номер подтверждения) должен приводить к появлению сегмента с пустым полем подтверждения, содержащего текущий номер в очереди на посылку, а также подтверждение, указывающее на следующий ожидаемый с этого соединения номер. Соединение остается в своем прежнем состоянии.
   Если пришедший сегмент имеет уровень защиты, изоляции или приоритета, не соответствующий местному уровню соединения, то отправляется сигнал перезагрузки, а соединение переходит в состояние CLOSED. Сигнал перезагрузки имеет номер очереди, соответствующий номеру сигнала ACK в пришедшем сегменте.

3.7.2 Обработка сигнала RST

Для всех состояний, кроме SYN-SENT, все сегменты с сигналом перезагрузки (RST) проходят проверку полей SEQ. Сигнал перезагрузки признается, если его номер очереди попадает в окно. В состоянии же SYN SENT (сигнал RST получен в ответ на посылку инициирующего сигнала SYN), сигнал RST признается, если поле ACK подтверждает ранее сделанную посылку сигнала SYN.

Получатель сигнала RST проверяет его, и затем меняет свое состояние. Если получатель находился в состоянии LISTEN, то он игнорирует сигнал. Если получатель находился в состоянии SYN-

RECEIVED, то он возвращается вновь в состояние LISTEN. В иных случаях получатель ликвидирует соединение и переходит в состояние CLOSED. Если получатель находится в каком-либо ином состоянии, то он ликвидирует соединение и прежде чем перейти в состояние CLOSED, оповещает об этом своего клиента.

3.8 Закрытие соединения

Состояние CLOSED означает "я не имею данных для передачи". Конечно, закрытие полнодуплексного соединения является предметом множества интерпретаций, поскольку не очевидно, как интерпретировать в соединении сторону, получающую информацию. Мы решили интерпретировать CLOSE в упрощенной манере. Клиент, находящийся в состоянии CLOSE, может все еще получать информацию (RECEIVE) до тех пор, пока партнер тоже не сообщит, что переходит в состояние CLOSE. Таким образом, клиент может изящно завершить работу на своем конце соединения. Программа протокола SCTPA гарантированно получит все буферы с информацией, отправленные до того, как соединение было закрыто. Поэтому клиенту, не ждущему информации с соединения, следует лишь ждать сообщения об успешном закрытии этого соединения, что означает, что все данные получены программой SCTPA, принимающей информацию. Клиенты должны сохранять уже закрытые ими для чтения информации соединения до тех пор, пока программа протокола SCTPA не сообщит им, что такой информации больше нет.

3.8.1 Местный клиент инициирует закрытие

В этом случае создается сегмент с сигналом FIN и помещается в очередь сегментов, ждущих отправления. После этого программа SCTPA уже не будет принимать от этого клиента каких-либо команд на отправление данных по закрытому соединению, а сама переходит в со стояние FIN-WAIT-1. Тем не менее, в этом состоянии еще возможно получение клиентом данных с этого соединения. Все сегменты, стоящие в очереди, и сам сегмент с сигналом FIN будут в случае необходимости посылаться напарнику вновь и вновь, пока не получат своего подтверждения.

Когда программа SCTPA партнера подтвердит получение сигнала FIN, и сама отправит сюда свой сигнал FIN, местная программа может подтвердить получение последнего. Заметим, что программа SCTPA, получающая сигнал FIN, будет подтверждать его, но не будет посылать своего собственного сигнала FIN до тех пор, пока ее клиент тоже не закроет соединения.

	SCTPA A				**SCTPA B**
1.	ESTABLISHED				ESTABLISHED
2.	(Close)	-->	<SEQ=100><ACK=300><CTL=FIN,ACK>	-->	CLOSE-WAIT
	FIN-WAIT-1				
3.	FIN-WAIT-2	<--	<SEQ=300><ACK=101><CTL=ACK>	<--	CLOSE-WAIT
4.	(Close)	<--	<SEQ=300><ACK=101><CTL=FIN,ACK>	<--	LAST-ACK
	TIME-WAIT				
5.	TIME-WAIT	-->	<SEQ=101><ACK=301><CTL=ACK>	-->	CLOSED

6. (2 MSL)

CLOSED

Рисунок 19. Нормальная процедура закрытия

3.8.2 Программа SCTPA получает из сети сигнал FIN.

Если из сети приходит невостребованный сигнал FIN, то принимающая его программа SCTPA может подтвердить получение такого сигнала и оповестить своего клиента о том, что соединение закрыто. Клиент ответит командой CLOSE, по которой программа SCTPA может после пересылки оставшихся данных послать партнеру сигнал FIN. После этого программа SCTPA ждет, пока не придет подтверждение на отправленный ею сигнал FIN, после чего она ликвидирует соединение. Если подтверждения не было по истечении отведенного времени, то соединение ликвидируется в принудительном порядке, о чем дается сообщение клиенту.

3.8.3 Одновременное закрытие соединения

Одновременное закрытие соединения клиентами на обоих концах приводит к обмену сегментами с сигналом FIN. Когда все сегменты, стоящие в очереди перед сегментом с FIN, будут переданы и получат подтверждение, каждая программа SCTPA может послать подтверждение на полученный ею сигнал FIN. Обе программы по получении этих подтверждений будут ликвидировать соединение.

	SCTPA A				**SCTPA B**
1.	ESTABLISHED				ESTABLISHED
2.	(Close)			...	
	FIN-WAIT-1	-->	<SEQ=100><ACK=300><CTL=FIN,ACK>	...	FIN-WAIT-1
		<--	<SEQ=300><ACK=100><CTL=FIN,ACK>	<--	
		...	<SEQ=100><ACK=300><CTL=FIN,ACK>	-->	
3.	CLOSING	-->	<SEQ=101><ACK=301><CTL=ACK>	...	CLOSING
		<--	<SEQ=301><ACK=101><CTL=ACK>	<--	
		...	<SEQ=101><ACK=301><CTL=ACK>	-->	
4.	TIME-WAIT				TIME-WAIT
	(2 MSL)				(2 MSL)
	CLOSED				CLOSED

Рисунок 20. Процедура одновременного закрытия соединения с обоих концов

3.9 Передача данных

Передача данных осуществляется с помощью обмена сегментами. Т.к. сегменты могут быть потеряны в результате ошибок (например, ошибки в контрольной сумме), неудачными для SCTPA прерываниями синхронной фазой SCTPS или перегрузки сети, то программа протокола SCTPA использует механизм повторной посылки (по истечении определенного времени) с тем, чтобы убедиться в получении каждого сегмента. В главе, посвященной номерам очередей, обсуждалось, как программа SCTPA в сегментах осуществляет проверку номеров очередей и номеров подтверждения на предмет их корректности.

Отправитель данных с помощью значения переменной SND.NXT отслеживает следующий номер в очереди, подлежащий отправке. Получатель данных с помощью переменной RCV.NXT отслеживает следующий номер, прибытие которого он ожидает. В переменную SND.UNA отправитель данных помещает значение самого старого номера, который был отправлен, но еще не получил подтверждения. Если бы поток данных моментально иссяк, а все отправленные данные получили подтверждение, то тогда бы все эти при переменные содержали одинаковое значение.

Когда отправитель информации создает и посылает некий сегмент, он увеличивает значение переменной SND.NXT. Адресат по получении сегмента увеличивает значение переменной RCV.NXT и отправляет подтверждение. Когда программа SCTPA, пославшая данные, получает подтверждение, она увеличивает значение SND.UNA. Разность в значениях этих переменных является мерой, характеризующей задержку сегментов в сети. Величина, на которую надо всякий раз осуществлять приращение значения этих переменных, является длиной поля данных в сегменте. Заметим, что поскольку соединения находятся в состоянии ESTABLISHED, все сегменты, в дополнение к собственно данным, должны нести некую информацию о подтверждении ранее отправленных сегментов.

Запрос пользователя о закрытии соединения (CLOSE) подразумевает использование функции проталкивания, что осуществляется с помощью контрольного флага FIN приходящем сегменте.

3.9.1 Контрольное время повторной посылки

Пример процедуры измерения контрольного времени для повторной посылки сегментов

Измеряется время, прошедшее между посылкой октета данных, имеющего некий определенный номер в очереди, и получение подтверждения, указывающего наряду с другими и на этот номер (посылаемые сегменты не обязаны соответствовать полученным сегментам),и змеренная временная задержка - это время обращения (Round Trip Time - RTT).

Следующий шаг - вычисление усредненного времени обращения (Smoothed Round Trip Time - SRTT):

SRTT = (ALPHA * RSTT) + ((1-ALPHA) * RTT)

Затем с помощью найденного значения определяется контрольное время повторной посылки (Retransmission Timeout - RTO):

RTO = min[UBOUND, max[LBOUND, (BETA * SRTT)]]

ЕСЛИ CTIME +SRTT+RTO > SOC, ТО RTO += SOA

,где:

- UBOUND - верхний предел контрольного времени (например, 1 сек),
- LBOUND - нижний предел (например, 0,1 сек).
- ALPHA - фактор сглаживания (например, от 0.8 до 0.9),
- BETA - фактор изменения задержки (например, от 1.3 до 2.0).
- CTIME - текущее время
- SOC - время начала синхронной фазы
- SOA - время начала асинхронной фазы

3.9.2 Передача срочной информации

Механизм срочной передачи протокола SCTPA предназначен для того, чтобы клиент, отправляющий данные, мог побудить получателя принять некую срочную информацию, а также позволить программе SCTPA, принимающей данные, информировать своего клиента, когда вся имеющаяся на настоящий момент информация будет получена. В бортовой сети к срочной информации может относиться, например, потоковое видео.

Данный механизм позволяет помечать некую точку в потоке данных как конец блока срочной информации. Когда в программе SCTPA, принимающей данные, данная точка окажется впереди индикатора номера в очереди получения (RCV.NXT), эта программа SCTPA должна дать команду своему клиенту перейти в "срочный режим". Когда номер в очереди получения догонит срочный указатель в потоке данных, программа SCTPA должна дать команду клиенту прийти в "нормальный режим". Если срочный указатель сменит свое положение, когда клиент находится в "срочном режиме", последний не узнает об этом.

Данный метод использует поле флага срочности, который присутствует во всех передаваемых сегментах. Единица в поле контрольного флага URG означает, что задействовано поле срочности . Чтобы получить указатель этого поля в потоке данных, необходимо дополнить его номером рассматриваемого сегмента в очереди. Отсутствие флага URG означает отсутствие у отправителя не посланных срочных данных.

При указании срочности клиент должен также послать по крайней мере один октет данных. Если клиент, помещающий данные, дополнительно закажет функцию проталкивания, то передача срочной информации ждущему ее процессу должна произойти незамедлительно.

3.9.3 Управление окном

Окно, посылаемое с каждым сегментом, указывает диапазон номеров очереди, которые отправитель окна (он же получатель данных) готов принять в настоящее время. Предполагается, что такой механизм связан с наличием в данный момент места в буфере данных.

Указание окна большого размера стимулирует передачу. Но если пришло большее количество данных, чем может быть принято программой SCTPA, либо данные были переданы неполностью из-за прерывания SCTPS, то данные будут отброшены. Это приведет к излишним пересылкам информации и ненужному увеличению нагрузки на сеть и программу SCTPA. Указание окна малого размера может ограничить передачу данных скоростью.

Такие механизмы протокола позволяют программе SCTPA заявлять большое окно, но впоследствии объявлять окна намного меньшего размера и не принимать такое большое количество данных. Такое, так называемое, сокращение окна выглядит довольно обескураживающе. Принцип устойчивости диктует, чтобы программа протокола SCTPA не сокращала сама окно, но была бы готова к таким действиям со стороны другой программы SCTPA.

Программа SCTPA, посылающая данные, должна быть готова принять от клиента и передать по сети по крайней мере один октет новых данных, даже если окно отправления равно нулю.

Программа SCTPA пакует предназначенные к в пересылке данные в сегменты, заполняющие текущее окно. Однако она не может перепаковать уже имеющиеся сегменты в очереди на повторную посылку.

В соединении, имеющем односторонний поток данных, информация об окне будет передаваться с сегментами подтверждения, а они будут все иметь одинаковый номер очереди. Поэтому не будет способа восстановить их очередность при получении. Это не является серьезной проблемой, но может случайно привести к получению информации об окне из какого-нибудь устаревшего сообщения. Во избежание такой проблемы должен осуществляться отсев и информация об окне должна браться из сегментов, имеющих самый большой номер в очереди (это сегменты, чей номер подтверждения больше или равен наибольшему из ранее полученных номеров).

Процедура управления окном оказывает значительное влияние на характеристики коммуникаций. Чтобы избежать применения малых окон, получателю данных предлагается откладывать изменение окна до тех пор, пока свободное место не составит от 20 до 40 процентов от максимально возможного в памяти для этого соединения и при этом также не стоит забывать об интервалах изохронных циклов SCTPS.

4. Интерфейсы

Данный раздел рассматривает три интерфейса:

- Клиент - SCTP и в частности:
    - Клиент - SCTPS;
    - Клиент - SCTPA;
- SCTP - SCA.

4.1 Интерфейс Клиент-SCTP

Нотация вызова подобна нотации большинства процедур или нотации вызова функции в языках высокого уровня, однако это не означает неправомерность вызовов на обслуживание в виде ловушек(сторожей), например SVC, UUO, EMT.

Описанные ниже команды клиента определяют основные операции, которые должна выполнять программа протокола SCTP для поддержки коммуникаций между процессами. Отдельные реализации протокола должны определять свой собственный конкретный формат и могут обеспечить комбинации или наборы базовых функций для одиночных вызовов. В частности, некоторые реализации могут автоматически открывать соединение (OPEN), как только по нему клиент дает первую команду посылки (SEND) или получения (RECEIVE).

Для того чтобы поддерживать интерфейс между процессами, программа SCTP должна не только принимать команды, но и возвращать некую информацию обслуживаемым процессам. Эта информация состоит из:

- общей информации о соединении (т.е. прерываний, закрытия соединения партнером, управление связью с не предопределенным чужим сокетом).
- ответа на конкретные команды клиента, указывающего на успешность действий или различные типы ошибок.

Команды приведены в трех следующих форматах:

SCTP.Команда(SCTPObject) - интерфейс протокола SCTP, где SCTPObject - интерфейс, который реализуют два класса SCTPSObject для SCTPS протокола и SCTPAObject - для SCTPA протокола и инкапсулирующие методы SCTPS.Команда и SCTPA.Команда соответственно.

SCTPS.Команда(обязательные, параметры[,необязательные] [,параметры]) - метод интерфейса протокола SCTPS

SCTPA.Команда(обязательные, параметры[,необязательные] [,параметры]) - метод интерфейса протокола SCTPA

Способ обработки ошибок выполнения строго не регламентируется и может быть построен как на возвращении методами кода ошибок, так и на использовании исключений.

4.1.1 Открыть соединение

SCTP.OPEN(SCTPObject)

SCTPS.OPEN([конфигурационные опции]) -> ссылка на Singleton сокет.

SCTPA.OPEN(свой порт, чужой порт, активный/пассивный [,контрольное время] [,приоритет] [,опции]) -> ссылка на сокет

Команда открытия соединения. При повторном открытии SCTPA сокета будет возвращена ошибка «сокет уже открыт».

4.1.2 Послать данные

SCTP.SEND(SCTPObject)

SCTPS.SEND(адрес буфера, количество байтов с данными)

SCTPA.SEND (ссылка на сокет, адрес буфера, количество байтов с данными [, флаг проталкивания] [, флаг срочности] [,контрольное время])

Данная команда приводит к тому, что данные, содержащиеся в указанном клиентом буфере, передаются на указанное соединение. Если соединение не было к этому времени открытым, команда SEND является ошибочной.

4.1.3 Получить данные

SCTP.RECEIVE(SCTPObject)

SCTPS.RECEIVE(адрес буфера, счетчик байт) -> счетчик байт

SCTPA.RECEIVE(ссылка на сокет, адрес буфера, счетчик байт) -> счетчик байт, флаг срочности, флаг проталкивания.

Данная команда размещает получаемую информацию в буфере, связанном с конкретным соединением. Если команде не предшествует команда OPEN или если процесс, осуществляющий вызов, не уполномочен на использование данного соединения, то возвращается ошибка.

4.1.4 Закрыть соединение

SCTP.CLOSE(SCTPObject)

SCTPS.CLOSE()

SCTPA.CLOSE(ссылка на сокет)

Данная команда приводит к закрытию указанного соединения. Если соединение не открыто или отдавший команду процесс не уполномочен использовать данное соединение, то возвращается со общение об ошибке. Предполагается, что закрытие SCTPA соединения будет медлительной операцией в том смысле, что оставшиеся команды посылки SCTPA.SEND будут еще некоторое время передавать данные (и даже, в случае необходимости, делать это повторно), насколько это позволит управление потоком, и не будет выполнена заказанная работа. Таким образом, можно будет сделать несколько команд посылки SCTPA.SEND, а затем закрыть соединение командой SCTPA.CLOSE, будучи уверенным, что отправленные данные достигнут адресата. Очевидно, что клиенты должны продолжать давать команды получения данных с уже закрытых соединений, поскольку чужая программа будет еще пытаться переслать оставшиеся у нее данные.

4.1.4 Статус соединения

SCTP.STATUS(SCTPObject)

SCTPS.STATUS()->информация о статусе

SCTPA.STATUS(ссылка на сокет)->информация о статусе

Это команда клиента, зависящая от конкретной реализации. Она должна выполняться без опасных последствий для системы. Возвращаемая клиенту информация обычно получается из блоков TCB (ATCB, STCB), связанного с данным соединением.

Данная команда может возвращать блок данных с разнообразной информацией о:

- текущей фазе;
- текущем состоянии
- местном сокете;
- чужом сокете;
- ссылках на сокеты;
- окне получения;
- окне отправления;
- статусе соединения;
- количестве буферов, ждущих подтверждения;
- количестве буферов, ожидающих получения данных;
- статусе срочности;
- приоритете;
- безопасности/закрытости;
- контрольном времени пересылки.

В зависимости от состояния соединения или от особенностей реализации протокола, часть указанной информации может быть недоступна или не имеет смысла. Если процесс, осуществивший вы зов, не имеет прав на использование данного соединения, то воз вращается сообщение об ошибке. Такой подход не позволяет не имеющим полномочий процессам получать информацию о соединении.

4.1.5 Ликвидация соединения

SCTP.ABORT(SCTPObject)

SCTPS.ABORT()

SCTPA.ABORT(ссылка на сокет)

Выполнение данной команды приводит к ликвидации всех незаконченных операций посылки и получения данных. Блок TCB ликвидируется, а также должно быть послано специальное сообщение RESET программе SCTP на другом конце соединения. В зависимости от реализации протокола, клиенты могут получать сообщение о ликвидации в ответ на каждый оставшийся

невыполненным запрос о посылке или получении данных. Или же клиенты вместо этого могут просто получить подтверждение команды ABORT.

4.1.6 Сообщения клиенту от программы SCTP

Когда программа SCTP посылает сигнал программе клиента, то определенная часть информации передается также самому клиенту. Часто это осуществляется в виде сообщений об ошибках. В других случаях наряду с этим будет предоставляться информация, связанная с завершением посылки и получения данных, а также выполнением других команд клиента.

Предоставляется следующая информация:

- местное имя соединения — всегда
- строка отчета — всегда
- адрес буфера — посылка и получение данных
- количество байт (счетчик полученных байт) — получение данных
- флаг проталкивания — получение данных
- флаг срочности — получение данных

4.2 Интерфейс SCTP - SCA

Программа протокола SCTP для реальной посылки информации по сети, а также для ее получения делает запросы к модулю протокола нижнего уровня SCA. В качестве аргументов вызова запрашивается время жизни данных в сети. Подробно данный интерфейс рассмотрен в [2].

4.3 Обработка событий

Обработка, описанная в данной главе, является лишь примером одной из возможных реализаций протокола. Иные реализации могут иметь несколько иные процедуры обработки, однако они должны отличаться от описанных в данной главе лишь в деталях, но никак не по существу.

Деятельность программы протокола SCTP можно рассматривать как реагирование на события. Эти происходящие события можно разбивать на три категории:

- запросы клиентов,
- прибытие сегментов,
- истечение контрольного времени.

Данная глава описывает деятельность в протоколе SCTPA в ответ на каждое из этих событий. Во многих случаях необходимая обработка зависит от состояния соединения. События, которые могут произойти:

- Команды клиента
    - на открытие соединения

    - на посылку данных
    - на получение данных
    - на закрытие соединения
    - на ликвидацию соединения
    - на определение статуса соединения
- Получения сегментов
- Истечение контрольного времени
    - для действий клиента
    - для повторной посылки
    - в состоянии ожидания

Модель интерфейса SCTPA и клиента состоит в том, что команды клиента выполняются немедленно, а вероятный отложенный отчет предоставляется через механизм событий или псевдопрерываний. В дальнейшем описании понятие "сигнал" может обозначать некое основание для посылки отложенного отчета.

Сообщение об ошибках предоставляется в виде текстовых строк. Например, команды клиента, адресованные к несуществующим соединениям, получат сообщение "Ошибка: соединение не было открыто".

4.3.1 Запрос OPEN

4.3.1.1 Состояние CLOSED (т.е. блок TCB отсутствует)

Создать новый блок управления передачей (TCB) для хранения информации о состоянии соединения. Заполнить поля идентификатора местного сокета, чужого сокета, приоритета, закрытости/безопасности, а также контрольного времени для клиента. Заметим, что некоторые параметры чужого сокета могут остаться не конкретизированными при пассивном открытии и соответствующие им поля должны быть заданы исходя из параметров пришедшего SYN сигнала. В случае пассивного открытия следует перейти в состояние LISTEN и вернуть управление давшему команду OPEN процессу. Если открытие является активным, а чужой сокет не конкретизирован, то вернуть сообщение " ошибка: внешний сокет не определен ".

Если открытие является активным и указан чужой сокет, то послать сегмент с сигналом SYN. Выбирается начальный номер для очереди отправления. Посылаемый сегмент и сигналом SYN имеет форму <SEQ=ISS><CTL=SYN>. Установить переменную SND.UNA в ISS, а SND.NXT в ISS+1. Перейти в состояние SYN-SENT. Вернуть управление процессу, вызвавшему рассматриваемую команду.

Если сделавший запрос клиент не получил доступа к указанному в запросе сокету, то вернуть сообщение "ошибка: соединение невозможно для этого процесса". Если для создания нового

соединения нет места в памяти компьютера, то вернуть сообщение "ошибка: недостаточно ресурсов".

4.3.1.2 Состояние LISTEN

Если происходит активизация и указан чужой сокет, то сменить состояние соединения с пассивного на активный, выбрать ISS. Послать сегмент с сигналом SYN, занести в SND.UNA значение ISS, а в SND.NXT ISS+1. Перейти в SYN-SEND состояние.

Данные, указанные в команде SEND, могут быть посланы в том же сегменте с сигналом SYN, или же могут быть помещены в очередь на передачу, которая может быть осуществлена после перехода в ESTABLISHED состояние. Если в команде сделан запрос на применение бита срочности, то в результате ее выполнения должны быть посланы сегменты данных. Если в очереди заказов на пересылку нет места, то в результате будет получен ответ "ошибка: недостаточно ресурсов". Если чужой сокет не указан, то вернуть сообщение "ошибка: внешний сокет не определен".

4.3.1.3 Прочие состояния

- SYN-SENT
- SYN-RECEIVED
- ESTABLISHED
- FIN-WAIT-1
- FIN-WAIT-2
- CLOSE-WAIT
- CLOSING
- LAST-ACK
- TIME-WAIT

возвращают в ответ на команду открытия сообщение "ошибка: соединение уже существует"

4.3.2 Запрос SEND

4.3.2.1 Состояние CLOSED (например, нет блока TCB)

Если клиент не имеет доступа к такому соединению, то вернуть сообщение " ошибка: соединение невозможно для этого процесса ". В противном случае вернуть "ошибка: соединение не существует".

4.3.2.2 Состояние LISTEN

Если указан чужой сокет, то сменить состояние соединения с пассивного на активный, выбрать номер ISS. Послать сегмент с сигналом SYN, установить SND.UNA в ISS, а SND.NXT в ISS+1. Установить новое состояние SYN-SENT. Данные из вызова SEND могут быть посланы вместе с

сигналом SYN, а могут быть помещены в очередь и отправлены уже после установления ESTABLISHED состояния.

Если в команде дан запрос на применение бита срочности, то он должен быть передан вместе с сегментом данных, возникающим при выполнении этой команды. Если в очереди нет места для запроса, то вернуть сообщение "ошибка: недостаточно ресурсов". Если чужой сокет не указан, то вернуть "ошибка: внешний сокет не определен".

4.3.2.3 Состояния SYN-SENT, SYN-RECEIVED

Поместить данные в очередь с тем, чтобы отправить после установления ESTABLISHED состояния. Если в очереди нет места, то вернуть сообщение " ошибка: недостаточно ресурсов".

4.3.2.4 Состояния ESTABLISHED, CLOSE-WAIT

Сегментировать буфер данных и переслать его с ответным подтверждением (значение подтверждения = RCV.NXT). Если для размещения этого буфера недостаточно места в памяти, то просто вернуть сообщение " ошибка: недостаточно ресурсов".

Если установлен флаг срочности, то занести в SND.UP значение SND.NXT-1 и установить указатель срочности на уходящие сегменты.

4.3.2.5 Прочие состояния

- FIN-WAIT-1
- FIN-WAIT-2
- CLOSING
- LAST-ACK
- TIME-WAIT

Вернуть сообщение "ошибка: закрытие соединения" и не выполнять запрос клиента.

4.3.3 Запрос RECEIVE

4.3.3.1 Состояние CLOSED (например, отсутствует блок TCB)

Если клиент не имеет доступа к такому соединению, вернуть сообщение "ошибка: соединение невозможно для этого процесса". В противном случае вернуть сообщение "ошибка: соединение не существует".

4.3.3.2 Состояния LISTEN, SYN-SENT, SYN-RECEIVED

Поместить запрос в очередь на обслуживание после установления ESTABISHED состояния. Если в очереди для этого нет места, вернуть сообщение "ошибка: недостаточно ресурсов".

4.3.3.3 Состояния ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2

Если в пришедших сегментах недостаточно данных для выполнения данного запроса, поместить последний в очередь на обслуживание. Если же в очереди нет места для размещения запроса RECEIVE, вернуть сообщение " ошибка: недостаточно ресурсов".

Собрать данные из приходящих сегментов в буфере получения, а затем передать их клиенту. Установить флаг "обнаружено проталкивание" (PUSH), если это имеет место.

Если данным, передаваемым в настоящий момент клиенту, предшествовал RCV.UP, то оповестить клиента о присутствии срочных данных. Когда протокол SCTPA берет на себя ответственность(опционально) за получение клиентом данных, то это фактически означает обмен информацией с отправителем в виде подтверждений. Формирование такого подтверждения обсуждается ниже при рассмотрении алгоритма обработки приходящего сегмента.

4.3.3.4 Состояние CLOSE-WAIT

Поскольку партнер на другом конце соединения уже послал сигнал FIN, то команды RECEIVE должны получать данные, уже имеющиеся в системе, а не только те, которые уже переданы клиенту. Если в системе больше нет текста, ждущего своего запроса RECIVE, то передать клиенту сообщение "ошибка: закрытие соединения". В противном случае использовать для удовлетворения запроса RECEIVE любую имеющуюся информацию.

4.3.3.5 Состояния CLOSING,LAST-ACK,TIME-WAIT

Вернуть сообщение "ошибка: закрытие соединения".

4.3.4 Запрос CLOSE

4.3.4.1 Состояние CLOSED (например, нет блока TCB)

Если клиент не имеет доступа к такому соединению, вернуть сообщение "ошибка: соединение невозможно для этого процесса". В противном случае вернуть сообщение "ошибка: соединение не существует".

4.3.4.2 Состояние LISTEN

Любые остающиеся неудовлетворенными запросы RECEIVE будут завершены с сообщением " ошибка: закрытие". Стереть блок TCB, перейти в CLOSED состояние и вернуть управление клиенту.

4.3.4.3 Состояние SYN-SENT

Стереть блок TCB и вернуть сообщение "ошибка: закрытие соединения" для любых еще остающихся в очередях запросов SEND или RECEIVE. Состояние SYN-RECEIVED

Если не сделано каких-либо запросов SEND и нет данных, ожидающих отправки, то сформировать FIN сегмент и послать его, а затем перейти в FIN-WAIT-1 состояние. В противном случае поместить данные в очередь для рассмотрения после установления ESTABLISHED состояния.

4.3.4.4 Состояние ESTABLISHED

Поместить запрос в очередь в ожидании, когда все данные предшествующих команд будут сегментированы. Тогда сформировать FIN сегмент и отправить его партнеру. В любом случае перейти в FIN-WAIT-1 состояние.

4.3.4.5 Состояния FIN-WAIT-1, FIN-WAIT-2

Строго говоря, такая ситуация является ошибочной и должна привести к получению клиентом сообщения "ошибка: закрытие соединения", однако может быть приемлемым также ответ "Ok", пока не отправлен второй FIN (хотя первый FIN может быть отправлен повторно).

4.3.4.6 Состояние CLOSE-WAIT

Поместить этот запрос в очередь, пока все предшествующие запросы SEND не будут помещены в сегменты. Затем послать сегмент с сигналом FIN, перейти в CLOSING состояние.

4.3.4.7 Состояния CLOSING, LAST-ACK, TIME-WAIT

Возвратить сообщение "ошибка: закрытие соединения".

4.3.5 Запрос ABORT

4.3.5.1 Состояние CLOSED (например, нет блока TCB)

Если клиент не имеет доступа к такому соединению, вернуть сообщение "ошибка: соединение невозможно для этого процесса". В противном случае вернуть сообщение "ошибка: соединение не существует".

4.3.5.2 Состояние LISTEN

Любые остающиеся запросы RECEIVED должны завершиться с возвратом сообщения "ошибка: сброс соединения ". Стереть блок TCB, перейти в состояние CLOSED, вернуть управление программе клиента.

4.3.5.3 Состояние SYN-SENT

Все находящиеся в очереди запросы SEND и RECEIVE должны получить сообщение " ошибка: сброс соединения ", стереть блок TCB, перейти в состояние CLOSED, вернуть управление клиенту.

4.3.5.4 Состояния SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT

Послать сегмент перезагрузки:

<SEQ=SND.NXT><CTL=RST>

Все находящиеся в очереди запросы SEND и RECEIVED должны получить сообщение "ошибка: сброс соединения ". Все сегменты, находящиеся в очереди на передачу (за исключением только что сформированного сигнала RST) и в очереди на повторную пересылку должны быть ликвидированы. Стереть блок TCB, перейти в CLOSED состояние, вернуть управление клиенту.

4.3.5.5 Состояния CLOSING, LAST-ACK, TIME-WAIT

Вернуть сообщение "ok" и стереть блок TCB, перейти в состояние CLOSED, вернуть управление клиенту.

4.3.6 Запрос STATUS

4.3.6.1 Состояние CLOSED (например, нет блока TCB)

Если клиент не имеет доступа у такому соединению, то возвратить сообщение " ошибка: соединение невозможно для этого процесса ". В противном случае вернуть " ошибка: соединение не существует ".

4.3.6.2 Состояние LISTEN

Вернуть сообщение "state=LISTEN" и указатель на блок TCB.

4.3.6.3 Состояние SYN-SENT

Вернуть сообщение "state=SYN-SENT" и указатель на блок TCB.

4.3.6.4 Состояние SYN-RECEIVED

Вернуть сообщение "state=SYNRECEIVED" и указатель на блок TCB.

4.3.6.5 Состояние ESTABLISHED

Вернуть сообщение "state=ESTABLISHED" и указатель на блок TCB.

4.3.6.6 Состояние FIN-WAIT-1

Вернуть сообщение "state=FIN-WAIT-1" и указатель на блок TCB.

4.3.6.7 Состояние FIN-WAIT-2

Вернуть сообщение "state=FIN-WAIT-2" и указатель на блок TCB.

4.3.6.8 Состояние CLOSE-WAIT

Вернуть сообщение "state=CLOSE-WAIT" и указатель на блок TCB.

4.3.6.9 Состояние CLOSING

Вернуть сообщение "state=CLOSING" и указатель на блок TCB.

4.3.6.10 Состояние LAST-ACK

Вернуть сообщение "state=LAST-ACK" и указатель на блок TCB.

4.3.6.11 Состояние TIME-WAIT

Вернуть сообщение "state=TIME-WAIT" и указатель на блок TCB.

4.3.7 Приход сигментов

4.3.7.1 Состояние CLOSED (например, нет блока TCB)

Все данные из указанного сегмента будут выброшены. Сегмент, пришедший с сигналом RST, будет ликвидирован. Сегмент же, не содержащий сигнала RST, вызовет посылку сигнала RST в ответ. Подтверждение и номер очереди будут выбраны таким образом, чтобы сделать последовательность перезагрузки приемлемой для программы SCTPA, отправившей сегмент, который и вызвал такую реакцию.

1: Если бит ACK сброшен, то используется номер очереди нуль:

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>

2: Если же ACK установлен, то

<SEQ=SEG.ACK><CTL=RST>

3; Вернуть управление прерванной программе.

4.3.7.2 Состояние LISTEN

1. Проверить присутствие сигнала RST. Сигнал RST, пришедший вместе с сегментом, должен игнорироваться, а управление должно быть возвращено прерванной программе.

2. Проверить на присутствие ACK.
   Любое подтверждение является ошибкой, если оно пришло на конец соединения, все еще находящийся в состоянии LISTEN. В ответ на любой сегмент, пришедший с ACK, должен быть сформирован приемлемый сегмент с сигналом перезагрузки.
   Сигнал RST должен быть сформирован следующим образом:
   <SEQ=SEG.ACK><CTL=RST>
   Вернуть управление прерванной программе.

3. Проверить на присутствие сигнала SYN.
   Если установлен бит SYN, то проверить безопасность. Если значение параметра безопасность/закрытость в пришедшем сегменте не совпадает в точности со значением безопасность/ закрытость в блоке TCB, то послать сигнал перезагрузки и вернуть управление прерванной программе:
   <SEQ=SEG.ACK><CTL=RST>
   Если значение SEQ.PRC меньше, чем TCB.PRC, то перейти к следующему пункту 4.
   Установить RCV.NXT в SEG.SEQ+1, IRS установить в SEG.SEQ, а остальные тексты и функции управления поместить в очередь для последующей обработки. Выбрать значение для ISS и отправить сегмент подтверждения в форме
   <SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>
   Переменную SND.NXT установить в ISS+1, а SND.UNA в ISS. Установить для соединения новое состояние SYN-RECEIVED. Заметим, что в состоянии SYN-RECEIVED будут обрабатываться все приходящие данные и команды управления (вместе с SYN), но уже не будет как прежде осуществляться обработка сигналов SYN и ACK. Если состояние LISTEN не

сформулировано полностью (например, не указан исчерпывающе чужой сокет, то именно в этот момент должны быть доопределены поля блока TCB, оставшиеся незаполненными.

4. Искать в пришедшем сегменте остальные команды управления, а также собственно данные. Любые сегменты с иными командами управления или заполненные текстом (но не содержащие сигнала SYN) должны получить от местной программы SCTPA подтверждение, и, таким образом, будут отброшены во время работы с подтверждением. Приходящий сегмент с сигналом RST не может быть правильным, поскольку он не может являться ответом на информацию, переданную данной реализацией соединения. Так что Вы вряд ли получите это сигнал, но если это произойдет, выбросьте пришедший сегмент и верните управление прерванной программе.

4.3.7.3 Состояние SYN-SENT

1. Проверить бит ACK. Если бит ACK выставлен, то
   В случае, если SEG.ACK = <ISS или SEG.ACK > SND.NXT,
   послать сигнал перезагрузки
   <SEQ=SEG.ACK><CTL=RST>
   (Если не выставлен бит RST. Если он все же выставлен, то, ничего не делая выкинуть пришедший сегмент и вернуть управление прерванной программе.) Ликвидировать сегмент, вернуть управление.
   Если SND.UNA =< SEG.ACK =< SND.NXT, то полученное в сегменте подтверждение становится приемлемым.

2. Проверить бит RST.
   В случае, если бит RST выставлен
   Если ожидалось получение сегмента с подтверждением, то дать клиенту объявление "ошибка: сброс соединения", вернуть сегмент, перейти в состояние CLOSED, убрать блок TCB, и, наконец, вернуть управление прерванной программе. В противном случае (нет подтверждения) ликвидировать сегмент и вернуть управление.

3. Проверить уровни безопасности и приоритета
   Если в пришедшем сегменте значения полей безопасность/ закрытость не совпадают в точности со значениями в блоке TCB, то послать сигнал перезагрузки, а точнее если имеется ACK, то послать
   <SEQ=SEG.ACK><CTL=RST>
   в противном случае послать
   <SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>
   Если имеется значение ACK, то
   приоритет в пришедшем сегменте должен совпадать с приоритетом, указанным в блоке TCB. Если это не так, то послать сигнал перезагрузки
   <SEQ=SEG.ACK><CTL=RST>
   Если же ACK отсутствует, то выполнить следующее
   Если в сегменте указан приоритет выше, чем приоритет в блоке TCB, то, если это позволяют клиент и система, увеличить значение приоритета в блоке TCB до значения, указанного в сегменте.
   Если же увеличивать приоритет не разрешается, послать сигнал перезагрузки:

<SEQ=0><ACK=SEG.SEQ+SEG.LEN><CTL=RST,ACK>
Если в сегменте указан приоритет, меньший, чем в блоке TCB, то просто перейти к следующему пункту анализа.

4. Проверить установку бита SYN. Данный этап должен осуществляться только если бит ACK не вызывает проблем, или если он не установлен, сегмент также не содержит сигнала RST. Если бит SYN установлен и параметры безопасности/ закрытости приоритета являются приемлемыми, то в переменную SEG.NXT записать значение SEG.SEQ+1, а IRS установить равным SEG.SEQ.
   SND.UNA должно быть повышено до SEG.ACK (если имеется ACK), а любые сегменты в очереди на повторную посылку, получившие таким образом подтверждение, должны быть удалены.
   Если SND.UNA > ISS (наш сигнал SYN получит подтверждение), то установить для соединения состояние ECTABLISHED и сформировать ACK сегмент
   <SEQ-SND.NXT><ACK=RCV.NXT><CTL=ACK>
   и отправить его. В этот сегмент могут быть включены данные или команды из очереди на отправление. Если в пришедшем сегменте есть иные команды или даже некий текст в поле данных, то продолжить обработку далее, начиная с шестого этапа ниже, где осуществляется проверка бита URG. Если таких команд и данных нет, передать управление прерванной программе. Если же бит SYN не установлен, то перейти в состояние SYN-RECEIVED, сформировать сегмент SYN, ACK:
   <SEQ=ISS><ACK=RCV.NXT><CTL=SYN,ACK>
   и послать его. Если в пришедшем сегменте имеются команды или текст в поле данных, то поместить их в очередь для обработки после установления ESTABLISH состояния. Вернуть управление прерванной программе.

5. Если установлены биты SYN или RST, то выкинуть пришедший сегмент и вернуть управление прерванной программе. Если во время прихода сегмента соединение находилось в состоянии, не описанном выше, то

    a. Проверить номер очереди

        i. Состояния SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT
           Сегменты обрабатываются по очереди. По получении сегмента сперва осуществляется тест для удаления старых дубликатов, но дальнейшая обработка осуществляется в порядке номеров SEG.SEQ. Если содержимое сегмента перекрывает границу между старой и пока новой информацией, то должны обрабатываться только новые данные.
           ЕСЛИ SEG.LEN=0 И RCV.WND=0 ТО
               ПРОВЕРИТЬ SEG.SEQ = RCV.NXT
           ЕСЛИ SEG.LEN=0 И RCV.WND>0 ТО
               ПРОВЕРИТЬ RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND
           ЕСЛИ SEG.LEN>0 И RCV.WND=0 ТО сегмент неприемлем
           ЕСЛИ SEG.LEN>0 И RCV.WND>0 ТО
               ПРОВЕРИТЬ RCV.NXT =< SEG.SEQ < RCV.NXT+RCV.WND

или RCV.NXT =< SEG.SEQ+SEG.LEN-1 < RCV.NXT+RCV.WND

Если приходящий сегмент неприемлем, то в ответ послать его подтверждение

<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>

Если бит RST не установлен (если он все же выставлен, то выкинуть пришедший сегмент и вернуть управление прерванной программе.)

После посылки подтверждения ликвидировать не принятый сегмент и вернуть управление прерванной программе.

b. Проверить бит RST.

   i. Состояние SYN-RECEIVED.

   Если бит RST установлен, то выполнить следующие действия:

   Если данное соединение было инициировано командой пассивного открытия OPEN (например, был осуществлен переход из состояния LISTEN), то возвратить данное соединение в состояние LISTEN, а управление вернуть прерванной программе. Нет нужды информировать об этом пользователя.

   Если данное соединение инициируется командой активного открытия OEPN (например, был переход из состояние SYN-SENT), то происходит отказ от этого соединения, а клиенту посылается сообщение "connection refused". В любом случае должны быть удалены все сегменты из очереди на повторную посылку. Кроме того, в случае активного открытия перейти в состояние CLOSED, удалить блок TCB и вернуть управление прерванной программе.

   ii. Состояния ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT

   Если установлен бит RST, то все ждущие обработки запросы RECEIVE и SEND должны получить ответ "reset".

   Убрать все сегменты из очередей. Клиенты должны получить необязательное сообщение общего назначения "connection reset". Перейти в состояние CLOSED, стереть блок TCB и вернуть управление прерванной программе.

   iii. Состояния CLOSING, LAST-ACK, TIME-WAIT

   Если выставлен бит RST, то перейти в состояние CLOSED, удалить блок TCB и вернуть управление прерванной программе.

c. Проверить бит SYN

   i. Состояния SYN-RECEIVED, ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2, CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT

   Если SYN находится в пределах окна, то послать сигнал перезагрузки. Любые ждущие обработки команды RECEIVE и SEND должны получить ответ "reset", убрать из очередей все сегменты, а клиент должен получить необязательное общее сообщение "connection reset". Перейти в состояние CLOSED, убрать блок TCB, вернуть управление прерванной программе.

Если SYN находится за пределами окна, то до данного пункта дело не должно дойти. Еще на первом этапе (проверка номера очереди) должно было быть послано подтверждение. В-пятых, проверить поле ACK Если бит ACK не установлен, то сегмент ликвидировать, а управление передать прерванной программе.

Если бит ACK установлен

Состояние SYN-RECEIVED.

Если SND.UNA =< SEG.ACK =< SND.NXT, то перейти в состояние ESTABLISHED и продолжить обработку.

Если же подтверждение в сегменте оказалось неприемлемым, то сформировать сегмент с сигналом перезагрузки

<SEQ=SEG.ACK><CTL=RST>

и послать его.

ii. Состояние ESTABLISHED

Если SND.UNA < SEG.ACK =< SND.NXT, то установить в SND.UNA значение из SEG.ACK. Любые сегменты из очереди на повторную посылку, получившие при этом подтверждение, удаляются. Клиенты должны получить положительные отзывы на буферы, которые были посланы командой SEND, а ныне получили полное подтверждение (например, команда послать буфер с данными должна завершиться сообщением "ok").

Если подтверждение является дубликатом (SEG.ACK < SND.UNA), то его можно игнорировать.

Если сообщение ACK подтверждает что-либо, еще не отправленное (SEG.ACK > SND.NXT), то послать ACK, ликвидировать сегмент и вернуть управление прерванной программе.

Если SND.UNA < SEG.ACK =< SND.NXT, то следует обновить окно для посылки.

Если (SND.WL1 < SEG.SEQ или (SND.WL1 = SEG.SEQ и SND.WL2 =< SEG.ACK)), то установить SND.WND согласно значению SEG.WND, SND.WL1 в SEG.SEQ, SND.WL2 в SEG.ACK.

Заметим, что в SND.WND записано смещение относительно SND.UNA, xnj d SND.WL1 записан номер очереди для последнего сегмента, используемого для обновления SND.WND, а также, что в SND.WL2 записан номер подтверждения из последнего сегмента, используемого для обновления SND.WND. При этом проверка охраняет от использования устаревших сегментов для обновления окна.

iii. Состояние FIN-WAIT-1

Все так же как при обработки в случае состояния ESTABLISHED, но если наш сигнал FIN теперь получил подтверждение, то перейти к FIN-WAIT-2 и продолжить обработку в таком состоянии.

iv. Состояние FIN-WAIT-2

Все так же как при обработке для случая состояния ESTABLISHED, но если

очередь повторной посылки пуста, то команда клиента на закрытие соединения CLOSE может получить подтверждение ("ok"), но при этом не удаляет блока TCB.

v. Состояние CLOSE-WAIT
Делается та же обработка, что была для случая со стояния ESTABLISHED.

vi. Состояние CLOSING
Все так же, как при обработке в случае состояния ESTABLISHED, но если ACK в пришедшем сегменте подтверждает наш сигнал FIN, то перейти в состояние TIME-WAIT. В противном случае сегмент игнорируется.

vii. Состояние LAST-ACK
Единственная вещь, которая может произойти в этом состоянии получение подтверждения на сигнал FIN. Если наш сигнал FIN не был подтвержден, то удалить блок TCB, перейти в состояние CLOSED и вернуть управление прерванной программе.

viii. Состояние TIME-WAIT
Единственная вещь, которая может произойти в этом состоянии это повторная передача чужого сигнала FIN.

d. Подтвердить сигнал и повторно стартовать по истечении контрольного времени 2MSL.

6. Проверить бит URG

a. Состояния ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2
Если бит URG установлен, то в RCV.UP занести max(RCV.UP, SEG.UP), а также дать знать клиенту, что на другом конце соединения имеются срочные данные, если срочный указатель (RCV.UP) стоит перед данными. Если же клиент уже был оповещен о данной цепочке срочных данных (или если все еще находится в "режиме срочности"), не следует ему напоминать об этом снова.

b. Состояния CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT
Такого не должно произойти, поскольку был получен си гнал FIN с другого конца соединения. Игнорировать бит URG.

7. Обработать данные из сегмента

a. Состояния ESTABLISHED, FIN-WAIT-1, FIN-WAIT-2
Раз мы оказались в состоянии ESTABLISHED, то стало возможным принимать текст для размещения в буферах получения, указанных клиентом. Текст из сегментов может пере носиться в буферы до тех пор, пока либо не наполнится соответствующий буфер, либо не станет пустым сегмент. Если сегмент пуст и несет флаг проталкивания PUSH, то при возврате буфера оповестить клиента о том, что был получен сигнал PUSH.

Когда протокол SCTPA берет на себя ответственность за получение клиентом данных, то он должен также давать подтверждение факта получения этих данных. Как только программа протокола SCTPA принимает на себя управление потоком данных, она ставит значение RCV.NXT перед блоком принимаемых данных, а RCV.WND устанавливает соответственно емкости буфера в данный момент. В общем случае значения RCV.NXT и RCV.WND не должны корректироваться в меньшую сторону.

Послать подтверждение в виде
<SEQ=SND.NXT><ACK=RCV.NXT><CTL=ACK>
Данное подтверждение должно быть добавлено к уходящему на другой конец соединения сегменту и, по возможности, без излишних задержек.

b. Состояния CLOSE-WAIT, CLOSING, LAST-ACK, TIME-WAIT
Такого не должно случаться, поскольку был получен сигнал FIN с другого конца соединения. Игнорировать текст в сегменте.

8. Проверить бит FIN

a. Если состояния CLOSED, LISTEN или SYN-SENT
Не обрабатывать сигнала FIN. Поскольку нет возможности проверить SEG.SEQ, выкинуть пришедший сегмент и возвратить управление прерванной программе.

b. Иначе
Если бит FIN установлен, то дать клиенту сигнал "connection closing", с тем же сообщением завершить все ждущие решения запросы RECEIVED, установить RCV.NXT перед местом в очереди сигнала FIN, послать для последнего подтверждение. Заметим, что сигнал FIN подразумевает проталкивание (PUSH) текстов во всех сегментах, еще не полученных клиентом.

i. Состояния SYN-RECEIVED, ESTABLISHED
Перейти в состояние CLOSE-WAIT

ii. Состояние FIN-WAIT-1
Если наш сигнал FIN получил подтверждение (возможно в этом же сегменте), то перейти в состояние TIME-WAIT, за пустить контрольный таймер, отключить все иные таймеры. Если подтверждения не было, перейти в состояние CLOSING.

iii. Состояние FIN-WAIT-2
Перейти в состояние TIME-WAIT. Запустить контрольный таймер, отключить все контрольные таймеры.

iv. Состояние CLOSE-WAIT
Остаться в состоянии CLOSE-WAIT

v. Состояние CLOSING
Остаться в состоянии CLOSING

vi. Состояние LAST-ACK
Остаться в состоянии LAST-ACK

vii. Состояние TIME-WAIT
Остаться в состоянии TIME-WAIT. По истечении контрольного времени 2MSL стартовать повторно. Вернуть управление прерванной программе.

4.3.8 Истечение контрольного времени для клиента

Если истекло контрольное время, то в каком бы состоянии не находилась программа, убрать все очереди, дать клиенту общий сигнал "ошибка: соединение разорвано по таймауту пользователя ", такой же сигнал дать всем ждущим обработки запросам, ликвидировать блок TCB, перейти в состояние CLOSED, вернуть управление прерванной программе.

4.3.9 Истечение контрольного времени для повторной посылки

В каком бы состоянии не находилась программа, если для сегмента в очереди на повторную посылку истекло контрольное время, послать этот сегмент еще раз, но уже вне очереди, произвести вновь инициализацию таймера повторной посылки, вернуть управление прерванной программе.

4.3.10 Истечение контрольного времени для состояния TIME-WAIT

Если истекло контрольное время для состояния TIME-WAIT, то ликвидировать соединение и блок TCB, перейти в состояние CLOSED, вернуть управление прерванной программе.

Аббревиатуры

АСК

Контрольный бит (подтверждения), не занимающий какого-либо места в очереди. Бит информирует о том, что поле подтверждения в данном сегменте определяет номер очереди, который хочет получить программа протокола SCTPA, пославшая данный сегмент. Это означает подтверждение факта получения всех предшествующих сегментов в очереди.

Соединение

Логический путь для коммуникаций, определяемый парой сокетов.

Датаграмма

Сообщение, посылаемое через компьютерную коммуникационную систему с коммуникацией пакетов.

Адрес получателя

Адрес получателя это обычно идентификаторы сети и хост-компьютера

FIN

Контрольный бит (конечный), занимающий одно место в очереди и указывающий на то, что программа протокола SCTPA не будет более посылать данные или какие-либо команды, под которые следует в очереди отводить место.

Фрагмент

Часть логической единицы данных. В частности фрагмент SCA являются частью SCA датаграммы.

Заголовок

Контрольная информация в начале сообщения, сегмента, фрагмента, пакета или блока данных

Хост-компьютер, узел

Любой цифровой вычислитель: просто компьютер, микроконтроллер, вычислитель на базе ПЛИС и т.п. В частности, он является отправителем и получателем сообщений с точки зрения коммуникационной сети.

Идентификация

Поле SCA протокола. Значение этого поля назначает отправитель для идентификации с тем, чтобы осуществлять сборку фрагментов датаграммы.

SCA адрес

Адрес отправителя или получателя на уровне хоста в сети SCA.

SCA датаграмма

Блок данных, передаваемый между модулем протокола SCA и программой вышестоящего протокола, снабженное SCA заголовком.

SCA фрагменты

Часть данных из SCA датаграммы, которая обзавелась собственным SCA заголовком.

SCA

Протокол стека протоколов «Синхроком» под названием Синхроком-Адрес (SCA) уровня L3 в модели OSI/ISO.

IRS

Первоначальный номер в очереди получения. Первый номер очереди, который использует программа протокола SCTPA при посылке данных через соединение.

ISN

Первоначальный номер очереди. Первый номер, используемый соединением SCTPA (либо ISS либо IRS). Определяется процедурой выбора, использующей таймер.

ISS

Первоначальный номер в очереди посылки. Первый номер очереди, используемый программой протокола SCTPA при посылке данных через соединение.

Остающаяся очередь

Это следующий номер в очереди, который должен быть подтвержден программой SCTPA, получающей данные (или иначе наименьший номер в очереди, еще не получивший в данный момент своего подтверждения). Иногда на него ссылаются как на левый край окна посылки.

Модуль

Реализация, обычно программа, какого-либо протокола или иной процедуры.

MSL

Максимальное время жизни сегмента. Время, в течении которого SCTPA сегмент может существовать в системе бортовой сети.

Октет

Байт, состоящий из восьми битов.

Опции

Поле опций может содержать несколько опций, каждая опция может иметь длину в несколько октетов. В основном, опции используются для тестирования различных ситуаций. Например, опции могут нести временной штамп. Поля с опциями могут иметь оба протокола SCA и SCTP.

Пакет

Пакет данных, имеющий заголовок, который в свою очередь может быть логически завершенным, а может и не быть. Чаще это означает физическую упаковку данных, нежели логическую.

Порт

Часть сокета, указывающая логический канал ввода или вывода для процесса, имеющего дело с данными.

Процесс

Некая использующаяся программа. Отправитель или получатель данных с точки зрения протокола SCTP или иных фрагментов уровня хост-хост.

PUSH

Контрольный бит, который не требует места в очереди и указывает на то, что данный сегмент содержит данные, которые следует "протолкнуть" к клиенту-адресату.

RCV.NXT

Следующий номер в очереди получения

RCV.UP

Срочный указатель для получения

RCV.WND

Окно получения

Следующий номер в очереди получения

Это следующий номер в очереди, который хочет получить местная программа протокола SCTPA

Окно получения

Это понятие характеризует номера в очереди, которые должна получить местная программа протокола SCTPA. Таким образом, местная программа SCTPA считает, что сегменты, попадающие в диапазон от RCV.NXT до RCV.NXT+RCV.WND-1, несут данные и команды управления, которые следует принимать во внимание. Сегменты, чьи номера в очереди ни коим образом не попадают в этот диапазон, воспринимаются как дубликаты и ликвидируются.

RST

Контрольный бит (бит перезагрузки), который не занимает места в очереди и указывает, что получатель этого бита должен ликвидировать соединение без каких-либо дополнительных действий. Получатель может, основываясь на анализе номера очереди и поля подтверждения в сегменте, принесшем данный сегмент, решить, следует ли выполнять операцию перезагрузки или же следует проигнорировать эту команду. Ни в коем случае получатель сегмента с битом RST не должен давать в ответ ту же команду RST.

SEG.ACK

Подтверждение сегмента

SEG.LEN

Длина сегмента

SEG.PRC

Значение приоритета в сегменте

SEG.SEQ

Номер очереди для сегмента

SEG.UP

Поле срочного указателя для сегмента

SEG.WND

Поле окна в сегменте

Сегмент

Логический блок данных. В частности, сегмент SCTP является блоком данных, который передается между парой SCTP модулей.

Подтверждение сегмента

Номер для очереди в поле подтверждения в пришедшем сегменте

Длина сегмента

Место в очереди, которое занимают данные этого сегмента (с учетом также всех команд, под которые тоже отводится место в очереди).

Номер сегмента в очереди

Значение в поле номера у пришедшего сегмента

Номер в очереди отправления

Следующий номер очереди для местной программы протокола SCTP, отправляющей данные и использующей эти номера для управления соединением. Первоначальный номер очереди (ISN) выбирается процедурой инициализации, а затем увеличивается на единицу с передачей по сети каждого октета данных или некоторой команды.

Окно посылки

Окно представляет собой набор номеров из очереди, которые желает получить чужая программа протокола SCTPA. Информация о границах этого окна берется из сегментов, пришедших от чужой программы SCTPA, получающей данные. Программе протокола SCTPA дозволяется посылать данные с номерами от SND.NXT до SND.UNA+SND.WND-1 (конечно, это подразумевает повторную посылку тех данных, чьи номера лежат между SND.UNA и SND.NXT).

SND.NXT

Очередь на посылку

SND.UNA

Очередь еще не посланных данных

SND.UP

Срочный указатель в очереди на посылку

SND.WL1

Номер очереди сегмента в последнем обновленном окне

SND.WL2

Номер подтверждения сегмента в последнем обновленном окне

SND.WND

Окно посылки

Сокет

Адрес, который особым образом включает в себя идентификатор порта. А именно, он включает связь SCA адреса с SCTPA портом

Адрес отправителя

Адрес отправления, обычно состоящий из идентификаторов сети и хост-компьютера.

SYN

Контрольный бит в приходящем сегменте, который занимает одно место в очереди и используется для инициализации соединения, для указания, где начинается отсчет номеров очереди.

TCB, ATCB, STCB

Контрольный блок для передачи, некая структура данных, где записан статус соединения. ATCB - TCB в протоколе SCTPA, а STCB, соответственно, в SCTPS.

TCB.PRC

Приоритет данного соединения

SCTP

Протокол управления пересылкой, протокол для надежной передачи информации между хост-компьютерами в системе бортовых гетерогенных сетей.

URG

Контрольный бит (бит срочности), который не требует места в очереди. Этот бит требует, чтобы клиенту был послан приказ использовать ускоренную обработку до тех пор, пока имеются данные, чьи номера в очереди меньше, чем указано в срочном указателе.

Срочный указатель

Срочный указатель имеет значение лишь если установлен бит URG. В поле срочного указателя определяется значение, которое указывает на некий октет данных, последний был связан с запросом клиента на срочную пересылку