«Синхроком»

Стек протоколов взаимодействия по сети Ethernet

Описание протокола

Синхроком-Адрес (SCA)

Версия: 18-10-2015

Аннотация

Настоящий документ описывает сетевой протокол стека протоколов «Синхроком» под названием Синхроком-Адрес (SCA).

СОДЕРЖАНИЕ

Введение

1.1 Мотивация

Протокол SCA предназначен для использования в соединенных между собой компьютерных сетях обмена данными на основе коммутации пакетов.

Протокол обеспечивает ТОЛЬКО передачу блоков данных, называемых датаграммами между отправителем и получателем, хосты которых идентифицируются адресами фиксированной длины, в то время как сам стек протоколов(посредством протоколов более высокого чем SCA уровня) обеспечивает передачу данных в режиме жесткого реального времени во время синхронных фаз сетевого обмена и туннелирование других сетевых протоколов нестрогого/мягкого реального времени во время асинхронных фаз сетевого обмена.

Протокол также обеспечивает фрагментацию и сборку для датаграмм большого размера, если сеть не позволяет передать датаграмму целиком.

Протокол обеспечивает синхронизацию по времени участников сетевого взаимодействия и мониторинг работоспособности сети.

1.2. Сфера действия протокола

Протокол SCA ограничивается доставкой битовых пакетов (датаграмм) от отправителя к получателю через систему соединенных между собой сетей. Протокол не поддерживает механизмов повышения надежности сквозной доставки, управления потоком данных, сохранения порядка и других функций, общепринятых для протоколов прямого взаимодействия между хостами.

Стек протоколов «Синхроком» относится к L3-L6 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»

Для обеспечения работы протокола в режиме жесткого реального времени программное окружение должно отвечать требованиям, возлагаемым к этому режиму работы. Так, например, протокол «Синхроком» может быть реализован на архитектуре микроядра семейства L4 с реализацией клиента SCS на ОС Linux, запущенной в качестве гостевой ОС в гипервизоре. Другими вариантами окружения реального времени могут быть Xenomai - патчи для ядра Linux, QNX - операционные системы реального времени, но все они имеют ряд недостатков, рассмотрение которых выходит за рамки этого документа, поэтому в дальнейшем рассматривается вариант микроядра.

Сетевой протокол SCA и транспортный SCTP реализуются 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.

Микроядро можно представить как уровень L2 модели OSI/ISO в стеке протоколов «Синхроком».

1.3. Интерфейсы

Этот протокол вызывается протоколами взаимодействия “хост-хост” и сам вызывает функции локальных сетевых протоколов для передачи датаграмм хосту-получателю.

Например, модуль SCTP будет вызывать модуль SCA для размещения сегмента SCTP (заголовок SCTPS и пользовательские данные) как объекта данных датаграммы SCA. Модуль SCTP будет указывать адреса и другие параметры заголовка SCA в качестве аргументов при вызове функции SCA. Модуль SCA будет создавать датаграмму SCA и обращаться к локальному сетевому интерфейсу для передачи датаграммы.

1.4. Работа протокола

Протокол SCA выполняет две основных функции – адресацию (в том числе межсетевую - блоком разделения сетей) и фрагментацию/сборку датаграмм, а также, посредством протокола SCMP синхронизацию по времени сетевых узлов, мониторинг работоспособности бортовой сети, работа с ошибками.

Модули SCA используют адреса из заголовков SCA для передачи датаграмм в направлении получателя.

Модули SCA используют поля заголовков SCA для фрагментации и сборки датаграмм SCA при необходимости передачи через сети с малым размером пакетов, например CAN.

Модули SCA используются на каждом хосте, участвующем в сети. Эти модули используют общие правила интерпретации полей адреса и фрагментации/сборки датаграмм SCA.

2. Обзор

2.1. Связь с другими протоколами

На Рисунок 3 показаны связи стека протоколов «Синхроком» с другими протоколами и приложениями.

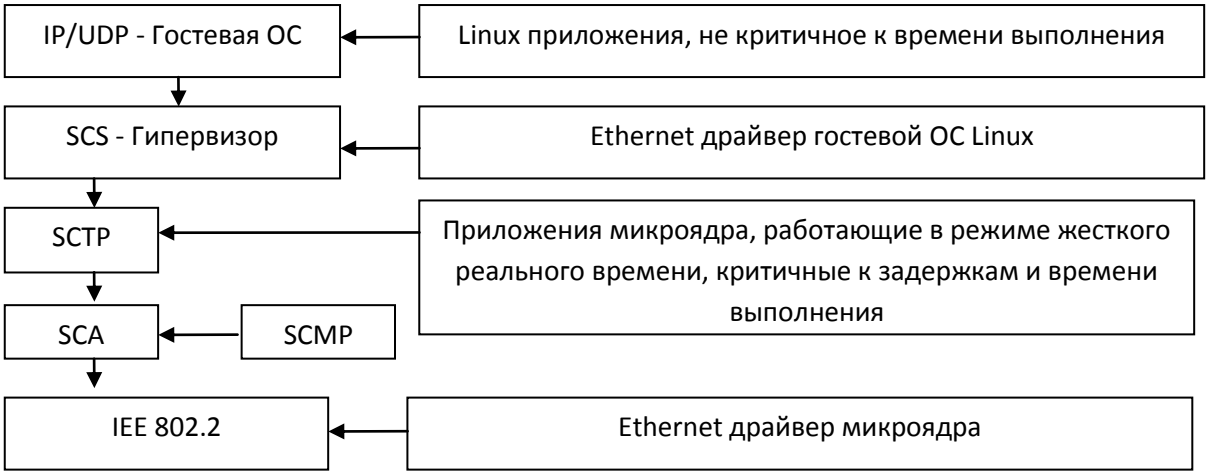

Рисунок 4 - Стек протоколов «Синхроком»

Протокол SCA взаимодействует с протоколом вышележащего уровня (протоколы взаимодействия между хостами – host-to-host) и с нижележащим протоколом локальной бортовой сети.

2.2. Модель работы протокола

Модель передачи датаграмм от одной прикладной программы к другой можно проиллюстрировать описанным ниже сценарием.

Передающая программа готовит свои данные и вызывает локальный модуль SCA для передачи этих данных как датаграммы, указывая адрес получателя и другие параметры в качестве аргументов.

Модуль SCA готовит заголовок датаграммы и присоединяет к нему данные. После этого модуль SCA определяет локальный сетевой MAC адрес для указанного получателя.

Модуль передает датаграмму и локальный адрес локальному сетевому интерфейсу.

Интерфейс канального уровня создает заголовок и присоединяет к нему датаграмму SCA, после чего пакет передается в бортовую сеть. На хосте получателя датаграмма выделяется из пакета интерфейсом канального уровня и передается модулю SCA.

Модуль SCA определяет по заголовку, что датаграмма адресована приложению на данном хосте и передает прикладной программе данные из датаграммы вместе с адресом отправителя и другими параметрами в ответ на системный вызов.

2.3. Функциональное описание

Модули SCA размещаются на хостах бортовой сети. Протокол SCA обеспечивает механизмы фрагментации и сборки датаграмм.

2.3.1 Адресация

Следует различать имена, адреса и маршруты. Имя указывает объект, который мы видим. Адрес показывает местонахождение, а маршрут говорит, как до него добраться. Протокол SCA имеет дело преимущественно с адресами. Отображение адресов на имена и обратно (преобразование) является задачей протоколов более высоких уровней (т. е., транспортного SCTP и сеансового SCS).

Адреса имеют фиксированную длину – 1 байт (8 бита).

2.3.2 Фрагментация

Фрагментация датаграмм SCA требуется в тех случаях, когда датаграмма происходит из сети, которая поддерживает больший размер пакетов, нежели промежуточные сети на пути к адресату.

Датаграмма SCA может быть помечена как *“не фрагментировать”*. Такие датаграммы не будут фрагментироваться ни при каких обстоятельствах. Если нефрагментируемая датаграмма SCA не может быть доставлена адресату без фрагментации, то она просто отбрасывается.

Процедуры фрагментации и сборки датаграмм должны обеспечивать возможность разбиения датаграмм на почти произвольное число частей, которые впоследствии могут быть собраны воедино. Получатель фрагментов использует поле идентификации для того, чтобы фрагменты разных датаграмм не перемешивались. Поле смещения датаграммы говорит получателю о положении фрагмента в исходной датаграмме. Поля смещения и размера фрагмента определяют часть исходной датаграммы, содержащуюся в отдельном фрагменте. Флаг наличия последующих фрагментов (если он сброшен) говорит о том, что фрагмент является последним в датаграмме.

Поле идентификации позволяет различать фрагменты разных датаграмм. Отправляющий датаграмму модуль протокола устанавливает значение поля идентификации в каждой датаграмме так, чтобы оно было уникальным для данной пары отправитель-получатель и протокола в течение времени присутствия датаграммы в бортовой сети. Этот модуль также устанавливает нулевые значения смещения фрагмента и флага наличия других фрагментов.

Для фрагментирования длинной датаграммы SCA модуль SCA создает две новых датаграммы SCA и копирует содержимое полей заголовка из длинной датаграммы в заголовки обеих новых датаграмм. Данные исходной датаграммы делятся на две части по 64 битовой (8 октетов) границе. Вторая часть датаграммы может иметь размер, не кратный 8 октетам (64 битам), но первая часть должна содержать целое число 8-октетных блоков. Назовем число 8-октетных блоков в первой части датаграммы NFB (Number of Fragment Blocks – число блоков фрагментации). Первая часть датаграммы помещается в первую из новых датаграмм SCA и поле длины устанавливается в соответствии длиной первой датаграммы. Для первой датаграммы устанавливается флаг наличия дополнительных фрагментов. Вторая часть данных помещается во вторую из созданных заново датаграмм и поле размера устанавливается в соответствии с длиной новой датаграммы. Значение поля смещения увеличивается на величину NFB. Значение флага наличия дополнительных фрагментов сохраняется в соответствии с флагом исходной нефрагментированной датаграммы.

Эту процедуру легко обобщить на случай разбиения датаграммы на n фрагментов, где n > 2.

Для сборки фрагментов датаграммы SCA, модуль SCA на хосте адресата объединяет датаграммы SCA с совпадающими значениями полей идентификации, адресов отправителя и получателя, а также протокола. Объединение осуществляется путем размещения данных из каждой

датаграммы в позицию буфера, указанную полем смещения фрагмента в заголовке SCA. Первый фрагмент будет иметь нулевое смещение, а для последнего фрагмента флаг *more-fragments* будет иметь нулевое значение.

3. Спецификация

3.1. Формат заголовка SCA

Заголовок датаграмм имеет следующий формат:

<table>
<tr><td colspan="10">0</td><td colspan="10">1</td><td colspan="10">2</td><td colspan="2">3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="10">Source Address</td><td colspan="10">Destination Address</td><td colspan="12">Total Length</td></tr>
<tr><td colspan="20" rowspan="2">Identification</td><td colspan="10">Flags</td><td colspan="2" rowspan="2">TTL</td></tr>
<tr><td>DF</td><td>MF</td><td>Protocol</td><td colspan="7">Ррезерв</td></tr>
<tr><td colspan="22">Checksum</td></tr>
</table>

Рисунок 5 Формат заголовка датаграммы SCA.

3.1.1 Source Address -8 бит

Адрес отправителя.

3.1.2 Destination Address -8 бит

Адрес получателя.

3.1.3 Total Length - 16 бит

Это поле указывает общий размер (в октетах) датаграммы с учетом заголовка и данных. Размер этого поля позволяет создавать датаграммы длиной до 65 535 октетов.

3.1.4 Identification - 16 бит

Значение поля идентификации присваивается отправителем для обеспечения корректной сборки фрагментов датаграммы.

3.1.5 Флаги - 8 бит

Набор флагов управления.

- Бит 0: (DF) 0 = фрагментация возможна, 1 = фрагментация недопустима.
- Бит 1: (MF) 0 = последний фрагмент, 1 = фрагмент не является последним.
- Бит 2: (Protocol) указывает протокол следующего уровня, содержащийся в поле данных датаграммы SCA:

- 0 -SCMP
- 1 - SCTP

- Биты 3-7- в резерве

3.1.6 Fragment Offset - 8 бит

Это поле показывает положение данного фрагмента в исходной датаграмме. Смещение измеряется в единицах, кратных 8 октетам (64 бита). Смещение первого фрагмента равно нулю.

3.1.7 TTL - 8 бит

Это поле определяет максимальный срок существования датаграммы в системе internet. Датаграммы с нулевым значением времени жизни должны уничтожаться. Значение этого поля изменяется при обработке заголовков SCA. Время измеряется в секундах, но, поскольку каждый обрабатывающий датаграмму модуль должен уменьшать значение TTL, по крайней мере, на 1 (даже если обработка длилась меньше секунды), значение TTL следует рассматривать как верхний предел срока жизни датаграммы в систем. Это поле введено для того, чтобы можно было избавиться от недоставленных датаграмм.

3.1.8 Protocol - 1 бит

Это поле указывает протокол следующего уровня, содержащийся в поле данных датаграммы SCA.

- 0 -SCMP
- 1 - SCTP

3.1.9 Header Checksum - 16 бит

Контрольная сумма полей заголовка. Поскольку некоторые поля заголовка (например, TTL) изменяются в процессе доставки, значение контрольной суммы проверяется и вычисляется заново в каждой точке обработки заголовков SCA.

Контрольная сумма заголовка представляет собой 16-битовое поразрядное дополнение (one's complement) суммы поразрядных дополнений всех 16-битовых слов заголовка. При вычислении контрольной суммы значение самого поля принимается нулевым.

3.2. Обсуждение

Реализация протокола должна быть гибкой и разумной. Каждая реализация должна предполагать интероперабельность с продукцией других разработчиков. Хотя целью данной спецификации является четкое и строгое описание протокола, существует вероятность различных интерпретаций стандарта. При передаче датаграмм следует строго следовать спецификации, сохраняя в то же время готовность к восприятию любых датаграмм, которые можно интерпретировать (например, не содержащих технических ошибок).

Базовый сервис SCA ориентирован на обработку датаграмм и обеспечивает фрагментацию датаграмм и сборку фрагментов модулем SCA хоста-получателя.

Адреса SCA позволяют различить отправителя и получателя на уровне хоста. Дополнительные сведения содержатся в поле протокола. Предполагается, что каждый протокол обеспечивает мультиплексирование (если оно требуется) на хосте.

3.2.1 Адресация

Для обеспечения достаточного для бортовой сети распределения адресов используется 8 битная адресация.

Назначение адресов приведено в таблице

Адрес	Описание
0	Зарезервировано (неиспользуется)
SLAVE_HOST_ADDRESS_MIN = 0 SLAVE_HOST_ADDRESS_MAX = 239	Интервал адресов подчиненных узлов: • SLAVE_HOST_ADDRESS_MIN - минимально возможное значение адреса подчиненного узла; • SLAVE_HOST_ADDRESS_MAX - максимально возможное значение адреса подчиненного узла.
MASTER_SYNC_HOST_ADDRESS = 240	Активный синхронизирующий узел
SYNC_HOST_ADDRESS_MIN =241 SYNC_HOST_ADDRESS_MAX = 250	Интервал адресов синхронизирующих узлов: • SYNC_HOST_ADDRESS_MIN - минимально возможное значение адреса синхронизирующего узла; • SYNC_HOST_ADDRESS_MAX - максимально возможное значение адреса синхронизирующего узла.
PSEUDO_HOST_ADDRESS = 251	Псевдоузел (используется для передачи сообщений самому себе)
EMPTY_HOST_ADDRESS = 252	Пустой узел (сообщения не отправляются в сеть)
TECH_HOST_ADDRESS = 253	Технологический узел (используется для настройки и диагностики остальных узлов)
GW_HOST_ADDRESS = 254	Роутер для выхода в сеть Ethernet для протоколов стека IP
BROADCAST_HOST_ADDRESS = 255	Широковещательное сообщение

Рисунок 6 Назначение адресов в бортовой сети

MAC-адрес узла имеет 00-00-00-00-00-XX, где XX - УИД в виде шестнадцатеричного числа.

Узлы, использующие «стуннелированные» протоколы стека IP, должны иметь следующие настройки:

- Шлюз узла гостевой ОС: 10.0.0.0/8

- Маска подсети 255.255.255.0
- Шлюз гипервизора 192.168.YYY.1, где YYY - УИД в виде десятичного числа

3.2.2 Формат адресов

Нулевое значение поля номера сети означает, что пакет адресован для данной сети.

3.2.3 Фрагментация и сборка

Поле идентификации (ID) используется вместе с адресами отправителя/получателя и полем протокола для идентификации фрагментов датаграммы при сборке.

Флаг наличия других фрагментов (More Fragments или MF) устанавливается для всех фрагментов датаграммы, кроме последнего.

Поле Fragment Offset показывает положение фрагмента относительно начала исходной (нефрагментированной) датаграммы.

Смещение учитывается в блоках размером 8 октетов. Стратегия фрагментации разработана таким образом, чтобы в нефрагментированной датаграмме вся поля, связанные с фрагментацией, имели нулевые значения (MF = 0, fragment offset = 0).

Если датаграмма SCA фрагментируется, ее поле данных должно делиться на части по 8-октетным границам.

Таким образом, используемый формат поддерживает до 2^{13} = 8192 фрагментов по 8 октетов (т. е., до 65 536 октетов). Отметим, что это значение соответствует возможным значениям поля размера датаграммы в ее заголовке (естественно, заголовок показывает общую длину датаграммы, а не ее фрагментов).

При фрагментации датаграммы некоторые опции копируются из оригинальной датаграммы в каждый фрагмент, а часть опций сохраняется только в первом фрагменте.

Каждый модуль SCA должен поддерживать пересылку без фрагментации датаграмм размером 68 октетов. Это значение выбрано потому, что заголовок датаграммы может достигать 60 октетов и поле данных должно содержать, по крайней мере, 8 октетов.

Каждый получатель должен поддерживать прием датаграмм размером, по крайней мере, 576 октетов целиком или с использованием фрагментации и сборки.

При фрагментации могут изменяться следующие поля:

1. смещение фрагмента
2. размер заголовка датаграммы
3. общий размер
4. контрольная сумма заголовка.

Если установлен флаг запрета фрагментации (DF), датаграммы, которые невозможно передать целиком, отбрасываются. Этот вариант используется в тех случаях, когда принимающий хост не

может собирать фрагменты датаграммы по причине нехватки ресурсов. Примером запрета фрагментации может служить ситуация с линией к небольшому хосту. Такой хост может иметь

программу самозагрузки (boot strap), которая воспринимает датаграмму, хранящуюся в памяти и потом выполняет содержащийся в ней код.

3.2.3.1 Пример процедуры фрагментации

Размер максимальной датаграммы, которая может быть передана через следующую сеть, называется MTU.

Если общий размер датаграммы не превышает MTU, эта датаграмма обрабатывается дальше. В противном случае датаграмма делится на два фрагмента - первый фрагмент имеет максимальный размер, а во втором содержится оставшаяся часть датаграммы. Первый фрагмент передается на следующий этап обработки, а для второго заново выполняется процедура проверки размера и при необходимости выполняется дополнительная фрагментация (это продолжается до тех пор, пока размер всех фрагментов перестанет превышать максимальное значение).

3.2.3.2.1 Обозначения

- FO - смещение фрагмента
- IHL - длина заголовка SCA
- DF - флаг запрета фрагментирования
- MF - флаг наличия других фрагментов
- TL - общий размер
- OFO - смещение старого фрагмента
- OIHL - размер заголовка старого фрагмента
- OMF - старый флаг наличия других фрагментов
- OTL - старое значение общей длины
- NFB - число фрагментов
- MTU - максимальный передаваемый блок

3.2.3.3.2 Процедура

Если TL =< MTU,

то датаграмма передается на следующий этап обработки,

иначе проверяется флаг DF.

Если DF = 1,

то датаграмма отбрасывается;

иначе выполняются следующие операции:

1: копируется исходный заголовок SCA;

2: OIHL <- IHL; OTL <- TL; OFO <- FO; OMF <- MF;

3: NFB <- (MTU-IHL*4)/8;

4: присоединяются первые NFB*8 октетов данных;

5: корректируется заголовок:

MF <- 1; TL <- (IHL*4)+(NFB*8);

заново вычисляется контрольная сумма;

6: фрагмент передается на следующий этап обработки;

переход ко второму фрагменту:

7: частичное копирование заголовка SCA (некоторые опции не копируются)

8: добавляется оставшаяся часть данных;

9: корректируется заголовок:

IHL <- (((OIHL*4)-(размер опций не копируется))+3)/4;

TL <- OTL - NFB*8 - (OIHL-IHL)*4);

FO <- OFO + NFB; MF <- OMF;

заново вычисляется контрольная сумма;

10: фрагмент передается процедуре проверки размера.

После выполнения п. 10 процедура завершается (если размер фрагмента не превышает допустимое значение) или повторяется.

Эта процедура создает фрагменты одинакового (максимального) размера (за исключением последнего). Могут использоваться и другие процедуры, которые создают фрагменты с размером меньше максимального. Например, процедура фрагментации может использовать повторяющиеся операции деления данных датаграммы пополам, пока оно не достигнет приемлемого для передачи размера.

3.2.3.2 Пример процедуры сборки

Для каждой датаграммы идентификатор буфера определяется путем объединения (конкатенации) полей адресов отправителя и получателя, протокола и идентификации. Если датаграмма не является фрагментом (поля смещения фрагмента и флага дополнительных фрагментов имеют нулевые значения), все ресурсы сборки, связанные с этим идентификатором буфера, освобождаются и датаграмма передается на следующий этап обработки.

Если больше нет фрагментом с таким же идентификатором буфера, происходит выделение ресурсов для сборки. Эти ресурсы включают буфер данных, буфер заголовков, таблицу фрагментов (блоков), поле общего размера и таймер. Данные из фрагментов помещаются в буфер данных в соответствии с размером и смещением фрагментов; флаги в таблице фрагментов устанавливается при получении соответствующего фрагмента.

Заголовок первого фрагмента (fragment offset=0) помещается в буфер заголовков. Если фрагмент является последним (MF=0), определяется общий размер датаграммы. Если фрагмент завершает прием датаграммы (проверяется по таблице блоков), датаграмма передается на следующий этап обработки; в противном случае для таймера устанавливается наибольшее из двух значений – текущие показания таймера и значение поля TTL из данного фрагмента; управление передается процедуре сборки фрагментов.

Если заданное таймером время истекло, освобождаются все ресурсы, выделенные для данного идентификатора буфера. Начальная установка таймера задает нижнюю границу времени ожидания при сборке. При получении фрагментов время ожидания может быть увеличено в соответствии с TTL, а уменьшение времени ожидания не предусмотрено. Максимальное значение таймера совпадает с максимальным значением TTL (255 сек.). Рекомендуется устанавливать начальное значение таймера равным в секундах равным количеству адресатов бортовой сети. Эта рекомендация может быть изменена с учетом реального опыт использования протокола. Отметим, что выбор этого значения связан с доступным размером буфера и скоростью среды передачи, т. е., при скорости 10 гбит/с и времени ожидания 15 сек. может потребоваться буфер размером 150 гбит.

3.2.3.2.1 Обозначения

- FO - смещение фрагмента
- IHL - длина заголовка Internet
- DF - флаг запрета фрагментирования
- MF - флаг наличия других фрагментов
- TTL - время жизни
- NFB - число фрагментов
- TL - общий размер
- TDL - общий размер данных
- BUFID - идентификатор буфера
- RCVBT - таблица полученных фрагментов
- TLB - нижняя граница таймера.

3.2.3.2.2 Процедура

1: BUFID <- source|destination|protocol|identification;

2: если FO = 0 и MF = 0

3: то если выделен буфер с BUFID

4: то удалить все ресурсы сборки для этого BUFID;

5: Передать датаграмму на следующий этап обработки; DONE.

6: иначе

если буфер с BUFID не выделен

7: то

выделить ресурсы сборки для BUFID;

TIMER <- TLB; TDL <- 0;

8: Поместить данные из фрагмента в буфер BUFID, начиная с октета FO*8 и до октета (TL-(IHL*4))+FO*8;

9: Установить биты RCVBT для октетов с FO до FO+((TL-(IHL*4)+7)/8);

10: если MF = 0 то TDL <- TL-(IHL*4)+(FO*8)

11: если FO = 0 то поместить заголовок в буфер заголовков

12: если TDL # 0

13: и все биты RCVBT от 0 до (TDL+7)/8 установлены

14: то TL <- TDL+(IHL*4)

15: Передать датаграмму на следующий этап обработки;

16: Освободить все ресурсы сборки, выделенные для BUFID; DONE.

17: TIMER <- MAX(TIMER,TTL);

18: повторять, пока не будут получены все фрагменты или не истечет время, заданное таймером;

19: истекло время ожидания: освобождаются все ресурсы, выделенные для BUFID; DONE.

В случаях, когда два фрагмента идентичны или перекрываются, эта процедура будет использовать более позднюю копию.

3.2.4 Идентификация

Выбор идентификатора для датаграммы базируется на обеспечении уникальности обозначения фрагментов отдельно взятой датаграммы. Модуль протокола, собирающий датаграмму из фрагментов, считает фрагменты относящимися к одной датаграмме, если у них совпадают адреса отправителя/получателя, тип протокола и поле идентификации. Таким образом, отправитель должен выбирать значение идентификатора так, чтобы оно было уникальным для комбинации

адресов отправителя/получателя и типа протокола в течение срока жизни датаграммы (или любого ее фрагмента).

Представляется целесообразным для передающего хоста сохранение таблицы использованных идентификаторов – по одной записи для каждого получателя, с которым хост взаимодействовал в течение последнего промежутка времени, равного максимальному сроку существования пакета в бортовой сети.

Однако поле идентификации позволяет выбрать любое из 65536 значений идентификатора, поэтому некоторые хосты могут использовать уникальные идентификаторы, не связанные с получателем.

Для некоторых ситуаций разумно выбирать значения идентификаторов с помощью протокола вышележащего уровня.

3.2.5 TTL

Значение TTL устанавливается отправителем и задает максимальное время существования датаграммы в бортовой сети. Если время, заданное полем TTL, истекло, датаграмма уничтожается.

Значение этого поля должно уменьшаться в каждой точке обработки заголовка датаграммы для того, чтобы учесть затраты времени на такую обработку. Даже в тех случаях, когда нет информации о затратах времени на обработку, значение поля должно уменьшаться на 1. Время жизни измеряется в секундах и максимальное значение TTL (255) соответствует 255 секундам или 4,25 мин. Поскольку каждый модуль, обрабатывающий датаграмму, должен уменьшать значение TTL, по крайней мере, на 1, значение TTL должно трактоваться как верхний предел срока существования датаграммы. Назначение поля TTL состоит в том, чтобы недоставленные датаграммы уничтожались и ограничить срок существования датаграмм в системе.

Некоторые протоколы вышележащих уровней, обеспечивающие гарантированную доставку, базируются на предположении, что старые копии датаграмм не могут приходить по истечении некоторого времени. Поле TTL дает таким протоколам гарантию истинности этого предположения.

3.2.6 Протокол

0 - значение по умолчанию.

В соответствии с 3.2.8 необходимо отдельно отмечать SCMP пакеты, для них значение поля Protocol равно 1.

3.2.7 Контрольная сумма

Контрольная сумма заголовка заново вычисляется каждый раз при изменении заголовка (например, при уменьшении TTL, добавлении или изменении опций, фрагментации датаграммы). Контрольная сумма на уровне SCA предназначена для защиты полей заголовка от ошибок при передаче.

Для некоторых приложений допустимо небольшое число ошибок, но недопустима задержка передачи. Если протокол SCA будет заниматься исправлением ошибок, такие приложения не смогут работать.

3.2.8 Работа с ошибками - SCMP

Протокол Synchrocom Control Message Protocol (SCMP) - протокол межсетевых управляющих сообщений, является неотъемлемой частью SCA и предназначен для передачи сообщений об ошибках. SCMP пакеты инкапсулируются в SCA пакеты.

Вариант использования SCMP можно проиллюстрировать следующим примером.

Каждая машина, которая перенаправляет SCA пакет, уменьшает TTL заголовка на единицу, если TTL достигает 0, SCMP сообщение о превышении TTL отправляется на источник пакета.

Также, SCMP сообщения генерируются при нахождении ошибок в заголовке SCA пакета (за исключением самих SCMP пакетов, дабы не привести к бесконечно растущему потоку SCMP сообщений об SCMP сообщениях).

Формат SCMP пакета приведен на рисунке 5.

<table>
<tr><td colspan="10">0</td><td colspan="10">1</td><td colspan="10">2</td><td colspan="2">3</td></tr>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>0</td><td>1</td></tr>
<tr><td colspan="8">Тип</td><td colspan="8">Код</td><td colspan="16">Контрольная сумма</td></tr>
<tr><td colspan="32">Данные(формат зависит от значения полей «Код» и «Тип»)</td></tr>
</table>

Рисунок 7 Формат пакета SCMP

Типы пакетов SCMP

<table>
<tr><td>Тип</td><td>Код</td><td>Сообщение</td><td>Данные</td></tr>
<tr><td>0</td><td>0</td><td>Эхо-ответ</td><td><table><tr><td>Идентификатор(16)</td><td>Номер последовательности(16)</td></tr><tr><td colspan="2">Данные (переменная)</td></tr></table></td></tr>
<tr><td>1,2</td><td></td><td>Зарезервировано</td><td></td></tr>
<tr><td rowspan="6">3</td><td colspan="2">Адресат недоступен</td><td rowspan="6"><table><tr><td>Исходное SCA сообщение</td></tr></table></td></tr>
<tr><td>0</td><td>Сеть недостижима</td></tr>
<tr><td>1</td><td>Узел недостижим</td></tr>
<tr><td>2</td><td>Протокол недостижим</td></tr>
<tr><td>3</td><td>Порт недостижим</td></tr>
<tr><td>4</td><td>Необходима фрагментация, но установлен флаг ее</td></tr>
</table>

<table>
<tr><td></td><td></td><td>запрета (DF)</td><td rowspan="11"></td></tr>
<tr><td></td><td>5</td><td>Неверный маршрут от источника</td></tr>
<tr><td></td><td>6</td><td>Узел назначения неизвестен</td></tr>
<tr><td></td><td>7</td><td>Узел источник изолирован</td></tr>
<tr><td></td><td>8</td><td>Сеть административно запрещена</td></tr>
<tr><td></td><td>9</td><td>Узел административно запрещен</td></tr>
<tr><td></td><td>10</td><td>Коммуникации административно запрещены</td></tr>
<tr><td>4</td><td>0</td><td>Сдерживание источника(отключение источника при переполнении очереди)</td></tr>
<tr><td rowspan="3">5</td><td colspan="2">Перенаправление</td><td rowspan="3"><table><tr><td>Адрес блока разделения сетей (32)</td></tr><tr><td>Исходное SCA сообщение</td></tr></table></td></tr>
<tr><td>0</td><td>Перенаправление пакетов в сеть CAN</td></tr>
<tr><td>1</td><td>Перенаправление пакетов к узлу CAN</td></tr>
<tr><td>6</td><td>0</td><td>Альтернативный адрес узла</td><td></td></tr>
<tr><td>7</td><td></td><td>Зарезервировано</td><td></td></tr>
<tr><td>8</td><td>0</td><td>Эхо-запрос</td><td><table><tr><td>Идентификатор (16)</td><td>Номер последовательности (16)</td></tr><tr><td colspan="2">Данные (переменная)</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td></tr>
<tr><td>9</td><td>0</td><td>Объявление блока разделения сетей</td><td><table><tr><td>Количество адресов (8)</td><td>Размер элемента (8)</td><td>Срок действия (16)</td></tr><tr><td colspan="3">Адрес[1] (16)</td></tr><tr><td colspan="3">Предпочтительность[1] (16)</td></tr><tr><td colspan="3">...</td></tr><tr><td colspan="3">Адрес[N] (16)</td></tr><tr><td colspan="3">Предпочтительность[N] (16)</td></tr></table></td></tr>
<tr><td>10</td><td>0</td><td>Запрос маршрутизатора</td><td></td></tr>
<tr><td>11</td><td colspan="2">Время жизни датаграммы истекло</td><td></td></tr>
<tr><td></td><td>0</td><td>Время жизни пакета (TTL) истекло при транспортировке</td><td rowspan="2"><table><tr><td>Исходное SCA сообщение</td></tr></table></td></tr>
<tr><td></td><td>1</td><td>Время жизни пакета истекло при сборке фрагментов</td></tr>
<tr><td>12</td><td colspan="3">Неверный параметр (проблема с параметрами датаграммы: ошибка в SCA заголовке или отсутствует необходимая опция)</td></tr>
<tr><td></td><td>0</td><td>Указатель говорит об ошибке</td><td><table><tr><td>Указатель (8)</td><td>Исходное SCA сообщение</td></tr></table></td></tr>
<tr><td></td><td>1</td><td>Отсутствует требуемая опция</td><td rowspan="2"><table><tr><td>Исходное SCA сообщение</td></tr></table></td></tr>
<tr><td></td><td>2</td><td>Некорректная длина</td></tr>
<tr><td>13</td><td>0</td><td>Запрос метки времени</td><td></td></tr>
<tr><td>14</td><td>0</td><td>Ответ с меткой</td><td><table><tr><td>Идентификатор (16)</td><td>Номер</td></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>времени</td><td><table><tr><td></td><td>последовательности (16)</td></tr><tr><td colspan="2">Начальное время (32)</td></tr><tr><td colspan="2">Время приема получателем(32)</td></tr><tr><td colspan="2">Время приемки отправителя(32)</td></tr></table></td></tr>
<tr><td>15</td><td>0</td><td>Информационный запрос</td><td rowspan="2"><table><tr><td>Идентификатор (16)</td><td>Номер последовательности (16)</td></tr></table></td></tr>
<tr><td>16</td><td>0</td><td>Информационный ответ</td></tr>
<tr><td>19
-
29</td><td></td><td>Зарезервировано для обеспечения безопасности и экспериментов на устойчивость к ошибкам</td><td></td></tr>
<tr><td>30</td><td colspan="2">Трассировка маршрута в блоке разделения сетей</td><td rowspan="3"><table><tr><td>Идентификатор(16)</td><td>Не используется (16)</td></tr><tr><td>Количество хопов исходящего пакета (16)</td><td>Количество хопов возвращающегося пакета (16)</td></tr><tr><td colspan="2">Скорость линии связи (32)</td></tr><tr><td colspan="2">MTU линии связи (32)</td></tr></table></td></tr>
<tr><td></td><td>0</td><td>Исходящий пакет успешно отправлен</td></tr>
<tr><td></td><td>1</td><td>Путь для исходящего пакета не найден, пакет уничтожен</td></tr>
<tr><td>31</td><td colspan="2">Ошибка преобразования датаграммы</td><td rowspan="2"><table><tr><td>Указатель (32)</td></tr></table></td></tr>
<tr><td></td><td>0</td><td>Неизвестная или</td></tr>
</table>

		неуказанная ошибка	Заголовок SCA и транспортного протокола исходной датаграммы
	1	Невозможно конвертировать опцию	
	2	Неизвестная обязательная опция	
	3	Неподдерживаемая обязательная опция	
	4	Неподдерживаемый транспортный протокол	
	5	Превышена полная длина	
	6	Превышена длина заголовка SCA	
	7	Номер транспортного протокола больше 255	
	8	Номер порта вне допустимого диапазона	
	9	Превышена длина заголовка транспортного протокола	
	10	Переход через границу 32 бит и установлен бит ACK	
	11	Неизвестная обязательная опция транспортного протокола	
32 - 255	Зарезервировано		

Рисунок 8 Типы пакетов SCMP

3.3. Интерфейсы

Протокол SCA взаимодействует с одной стороны с локальной сетью, а с другой – с протоколом вышележащего уровня или прикладной программой. Далее протокол вышележащего уровня и прикладные программы (или даже программы шлюзов) будем называть для краткости «пользователь», поскольку они используют услуги модуля SCA. Поскольку SCA имеет дело с датаграммами между передачей отдельных датаграмм нет почти никакой связи и пользователь при каждом обращении к модулю SCA передает ему все требуемые параметры для выполнения запрашиваемой операции.

3.3.1 Пример интерфейса с вышележащим уровнем

Ниже приведены два примера вызовов, удовлетворяющих требованиям к пользовательским вызовам SCA ("=>" означает возврат):

SEND (src, dst, prot, TTL, BufPTR, len, Id, DF => result)

где:

- src = адрес отправителя
- dst = адрес получателя
- prot = протокол
- TTL = время жизни
- BufPTR = указатель на буфер
- len = размер буфера
- Id = идентификатор
- DF = не фрагментировать
- result = отклик
- OK = датаграмма успешно отправлена
- Error = ошибка в аргументах или сетевая ошибка (локальная сеть)

RECV (BufPTR, prot, => result, src, dst, len)

где:

- BufPTR = указатель на буфер
- prot = протокол
- result = отклик
- OK = датаграмма успешно принята

- Error = ошибка в аргументах
- len = размер буфера
- src = адрес отправителя
- dst = адрес получателя

Для передачи датаграммы пользователь применяет вызов SEND, передавая все требуемые аргументы. Модуль SCA, принявший вызов, проверяет аргументы, готовит и передает сообщение. Если параметры указаны корректно и датаграмма воспринята локальной сетью, модуль возвращает сообщение об успешной передаче. Если какой-то из параметров указан некорректно или датаграмма не принята локальной сетью, модуль возвращает сообщение об ошибке. В таких случаях модуль должен также возвращать соответствующий отклик, указывающий причину ошибки. Уровень детализации таких откликов зависит от реализации.

Получение модулем SCA датаграммы из локальной сети может быть связано с ожидающим пользовательским вызовом RECV. Если такой вызов имеется, информация из датаграммы передается пользователю. Если же вызова нет, пользователю передается уведомление о прибывшей датаграмме. Если пользователь с указанным адресом не существует, отправителю возвращается сообщение SCMP об ошибке и датаграмма уничтожается.

Для уведомления пользователя могут применяться псевдопрерывания или аналогичный механизм, приемлемый в используемой реализацией среде.

Пользовательский вызов RECV может быть исполнен незамедлительно при наличии ожидающей датаграммы или помещен в состояние ожидания прихода датаграммы.

Адрес отправителя включается в вызов SEND, если передающий хост имеет несколько адресов (для логических или физических устройств). Модуль SCA должен убедиться, что полученный адрес корректен и связан с данным хостом.

Реализация также может разрешать или требовать вызов модуля SCA для индикации заинтересованности в получении или предоставления исключительного использования для класса датаграмм (например, все датаграммы с определенным значением поля протокола).

В этом параграфе было дано функциональное описание интерфейса пользователь - SCA. Использованные обозначения похожи на нотацию большинства языков высокого уровня, однако, такое использование не требует использования “функций-ловушек» (например, SVC, UUO, EMT) или иных форм взаимодействия между процессами.

Приложение A: Примеры и сценарии

Пример 1

Ниже приведен пример минимальной датаграммы SCA, содержащей данные

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
Source Address										Destination Address										Total Length = 21											
Identification = 111																				Flag=0			Fragment Offset=0							
TTL=123										Protocol=0										Header Checksum											
Данные…																															

Рисунок 9 Пример датаграммы SCA

Показанная датаграмма соответствует протоколу SCA; заголовок содержит три 32-битовых слов, а полный размер датаграммы составляет 21 октет. Показанная выше датаграмма является полной (не фрагментом).

Пример 2

В этом примере показана датаграмма среднего размера (452 октета данных), которая разбивается на два фрагмента, поскольку сеть не поддерживает передачу датаграмм, размер которых превышает 280 октетов.

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
Source Address										Destination Address										Total Length = 472											
Identification = 111																				Flag=0			Fragment Offset=0							
TTL=123										Protocol=0										Header Checksum											
Данные																															
Данные ... Данные																															
Данные																															

Рисунок 10 Пример датаграммы SCA

Ниже показан первый фрагмент, после выделения из датаграммы первых 256 октетов данных.

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
Source Address										Destination Address										Total Length = 276											

Identification = 111		Flag=0	Fragment Offset=0
TTL=119	Protocol=0	Header Checksum	
Данные			
Данные ... Данные			
Данные			

Рисунок 11 Пример фрагмента датаграммы SCA

Второй фрагмент будет иметь форму.

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
Source Address										Destination Address						Total Length = 216															
Identification = 111																Flag=1				Fragment Offset=32											
TTL=119										Protocol=0						Header Checksum															
Данные																															
Данные ... Данные																															
Данные																															

Рисунок 12 Пример фрагмента датаграммы SCA

Приложение B: Порядок передачи данных

Порядок передачи заголовков и данных, описываемых в этой спецификации, задается на уровне октетов. Датаграмма представляет собой группу октетов, которые передаются в том же порядке, в котором мы читаем. Например, в показанной ниже датаграмме октеты передаются в порядке возрастания номеров на рисунке.

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
1								2								3								4							
5								6								7								8							
9								10								11								12							

Рисунок 13 Порядок передачи байтов.

Когда октет представляет числовое значение, показанный слева бит является старшим или наиболее значимым. На приведенных в спецификации рисунках это бит 0. Например, показанная ниже последовательность битов задает десятичное число 170.

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

Рисунок 14 Значимость битов

Подобно этому для многооктетных полей, представляющих числа, указанный слева бит является старшим. При передаче многооктетных значений старший октет передается первым.

Приложение C: Перечень SCA служб

SCA службы - это сетевые приложения реализованные в микроядре и использующие для соединения протокол SCA.

sca-core - реализация протокола SCA

- send - FIFO очередь отправки сообщений
- recv - FIFO очередь получения сообщений

sca-qos - сервис качества сети

- ping - проверка скорости связи до конкретного узла
- stat - сетевые настройки хоста
- sync - синхронизация времени на узлах
- scmp-core - реализация протокола SCMP
- host - соответствие SCA адресов MAC адресам
- trace - правила маршрутизации, например, для доставки SCA сообщения в сеть CAN используется Ethernet Gateway.
- neighbors - контроль структуры SCA сети, появление новых хостов, потеря(дисконект) существующих

sca-socket - интерфейс для SCA приложений микроядра