

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
Государственное образовательное учреждение высшего профессионального образования
«ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Е.С. Чердынцев

МУЛЬТИМЕДИЙНЫЕ СЕТИ

Учебное пособие

Издательство
Томского политехнического университета
2008

УДК 681.327.2:378.64/69(075.8)
ББК Ч481.23Я73
Ч 459

Чердынцев Е.С.

Ч 459 Мультимедийные сети: учебное пособие / Е.С. Чердынцев. – Томск: Изд-во Томского политехнического университета, 2008. – 103 с.

В пособии рассмотрены вопросы, связанные с передачей мультимедийной информации по глобальным сетям на основе протокола RTP. Предназначено для магистров направления 230100 «Сети ЭВМ и телекоммуникации».

УДК 681.327.2:378.64/69(075.8)
ББК Ч481.23Я73

Рекомендовано к печати Редакционно-издательским советом
Томского политехнического университета

Рецензенты

Доктор технических наук, профессор ТУСУРа
Ю.П. Ехлаков

Кандидат технических наук, доцент кафедры КСУП ТУСУРа
Н.Ю. Хабибулина

© Томский политехнический университет,
2008
© Оформление. Издательство ТПУ, 2008

ОГЛАВЛЕНИЕ

ПРЕДИСЛОВИЕ	6
1. ВВЕДЕНИЕ В RTP	8
1.1. Отсылка пакетов RTP	8
1.2. Получение пакетов RTP	9
2. ПАКЕТНАЯ ПЕРЕДАЧА ЗВУКА И ИЗОБРАЖЕНИЯ ПО СЕТИ.....	11
2.1. TCP/IP и сетевая модель OSI.....	11
2.2. Характеристики производительности сетей IP	15
2.3. Измерение производительности сетей IP.....	16
2.4. Специфика использования транспортных протоколов.....	17
2.5. Требования по передаче звука и изображения в пакетных сетях.....	18
3. ПЕРЕДАЧА ЗВУКА И ИЗОБРАЖЕНИЯ ПО ПРОТОКОЛУ RTP	20
3.1. Базовые принципы RTP	20
3.2. Стандартные элементы RTP.....	20
3.3. Связанные с RTP стандарты.....	21
4. ОПИСАНИЕ ПРОТОКОЛА RTP	23
4.1. RTP сессии.....	23
4.2. Структура пакета RTP	23
4.3. Проверка качества пакета	28
4.4. Трансляторы и миксеры	28
5. ОПИСАНИЕ ПРОТОКОЛА RTCP	30
5.1. Компоненты RTCP	30
5.2. Передача пакетов RTCP	30
5.3. Формат пакетов RTCP	30
5.3.1. Формат RTCP RR.....	32
5.3.2. Интерпретация RR.....	33
5.3.3. Формат RTCP SR.....	34
5.3.4. Интерпретация SR	35
5.3.5. Описание источника RTCP SDES.....	36
5.3.6. Управление участниками сессии RTCP BYE	38

5.3.7. Пакеты, управляемые приложениями: RTCP APP.....	39
5.3.8. Компоновка пакетов.....	39
5.4. Безопасность и конфиденциальность.....	40
5.5. Проверка корректности данных.....	41
5.6. База данных участников сессии.....	42
5.7. Управление характеристиками времени.....	44
5.7.1. Отчетные интервалы.....	45
5.7.2. Базовые правила передачи.....	46
5.7.3. Процедура пересмотра вперед.....	48
5.7.4. Процедура пересмотра назад.....	50
5.7.5. Пересмотр пакетов ВУЕ.....	51
5.7.6. Общие проблемы реализации.....	51
6. ЗАХВАТ МУЛЬТИМЕДИА, ПРОИГРЫВАНИЕ И УПРАВЛЕНИЕ ХАРАКТЕРИСТИКАМИ ВРЕМЕНИ.....	52
6.1. Поведение отправителя.....	52
6.2. Захват мультимедиа и сжатие.....	52
6.2.1. Захват и сжатие звука.....	53
6.2.2. Захват и сжатие изображения.....	54
6.2.3. Использование предварительно записанной информации.....	55
6.3. Генерация пакетов RTP.....	55
6.3.1. Метки времени и модель времени RTP.....	56
6.3.2. Фрагментация.....	56
6.3.3. Заголовки, зависящие от формата данных.....	57
6.4. Поведение получателя.....	57
6.5. Получение пакетов.....	58
6.5.1. Получение пакетов данных.....	59
6.5.2. Получение управляющих пакетов.....	61
6.6. Буфер проигрывания.....	62
6.6.1. Базовые операции.....	63
6.7. Вычисление времени проигрывания.....	65
6.7.1. Преобразование к локальному времени.....	66
6.7.2. Компенсация изменений времени.....	67
6.7.3. Компенсация поведения отправителя.....	70
6.7.4. Компенсация неустойчивости работы сети.....	72
6.7.5. Компенсация изменений в маршрутизации.....	75
6.7.6. Компенсация изменений порядка пакетов.....	75

6.8.	Адаптация точки проигрывания	75
6.8.1.	Адаптация точки проигрывания для звука с подавлением тишины	76
6.8.2.	Адаптация точки проигрывания для звука без подавления тишины	77
6.8.3.	Адаптация точки проигрывания для видео.....	78
6.9.	Декодирование, смешивание и проигрывание	79
6.9.1.	Декодирование	79
6.9.2.	Смешивание звука	79
6.9.3.	Проигрывание звука.....	81
6.9.4.	Проигрывание изображения.....	82
7.	СИНХРОНИЗАЦИЯ ЗВУКА И ИЗОБРАЖЕНИЯ	83
7.1.	Поведение отправителя.....	83
7.2.	Поведение получателя.....	85
7.3.	Точность синхронизации	87
8.	КОМПЕНСАЦИЯ ОШИБОК	88
8.1.	Компенсация потерь звука.....	88
8.1.1.	Измерение качества звука.....	88
8.1.2.	Замещение периодами тишины.....	89
8.1.3.	Замещение шума	89
8.1.4.	Повторение	91
8.1.5.	Другие технологии восстановления речевого сигнала.....	92
8.2.	Компенсация потерь изображения.....	94
8.2.1.	Компенсация передвижения.....	94
8.3.	Чередование (Interleaving)	95
9.	ИСПРАВЛЕНИЕ ОШИБОК	96
9.1.	Прямое исправление ошибок	96
9.1.1.	Контроль четности.....	96
9.1.2.	НЕРАВНОМЕРНАЯ ЗАЩИТА ОТ ОШИБОК	98
9.1.3.	Коды Рида-Соломона	98
9.1.4.	Избыточное кодирование звука	98
9.2.	Кодирование канала	99
9.3.	Повторная передача.....	99
10.	КОНТРОЛЬ ПЕРЕГРУЗОК	101
	СПИСОК ЛИТЕРАТУРЫ	102

ПРЕДИСЛОВИЕ

Идея использования сети Интернет для передачи звука и изображения не нова. Первые попытки относятся к началу 70-х годов, а первый стандарт в этой области появился в 1977 году (Network Voice Protocol – NVP) и касался только передачи звука. Стандарты на передачу видео появились позже, но именно эту дату можно считать началом истории сетей мультимедиа.

Создатели стандарта NVP работали в сети ARPANET, предшественнице Интернета. В этих экспериментах число каналов передачи звука не превышало двух из-за низкой пропускной способности данной сети. Появление в 80-х годах спутниковой сети Wideband Satellite Network с пропускной способностью 3 mbps позволило не только увеличить число каналов передачи звука, но и провести первые эксперименты по передаче видео. Для организации вещания через данную сеть был разработан межсетевой протокол ST (Stream Protocol), который использовался совместно со второй версией протокола NVP (NVP-II).

В 1989–90 гг. на смену данной спутниковой сети пришла сеть Terrestrial Wideband Network и исследовательская сеть DARTnet. Протокол ST был переработан в протокол ST-II. В этих сетях уже широко применялись технологии проведения сетевых видеоконференций для обеспечения совместной работы групп исследователей.

Протоколы ST и ST-II на межсетевом уровне использовались параллельно с протоколом IP (Internet Protocol) преимущественно государственными и исследовательскими организациями. В марте 1992 была проведена аудиоконференция IETF (Internet Engineering Task Force, основная организация по разработке стандартов Интернет), в которой звук передавался всем 20 ее участникам на три континента по каналам мультивещания. С этой конференции началось развитие RTP (Real-Time Protocol) – транспортного протокола передачи информации в реальном времени через Интернет.

В результате данных экспериментов в начале 90-х годов возник интерес к проведению аудио- и видеоконференций. К этому времени развитие персональных компьютеров стало позволять захватывать, сжимать и воспроизводить потоки аудио и видео. В то же время развитие IP мультивещания дало возможность передавать эти потоки любому числу потребителей.

Видеоконференции и потоковое вещание явились хорошими примерами широковещательных приложений. В данном случае мультиве-

щение использовалось как для передачи данных в сети, так и в качестве механизма передачи информации между приложениями, работающими на одном компьютере (для синхронизации аудио и видео потоков). Развитие и использование средств проведения широковещательных конференций послужило толчком для понимания основных проблем передачи мультимедийной информации в реальном времени, а также повлияло на процесс разработки ключевых протоколов и стандартов в этой области.

Протокол RTP был разработан IETF в период с 1992 по 1996 год на основе протокола NVP-II. Средства проведения широковещательных конференций используют протокол RTP как для передачи данных, так и для их контроля, следовательно, RTP обеспечивает не только доставку мультимедийной информации, но и поддерживает управление участниками конференции, синхронизацию звука и изображения, а также формирование отчетов о качестве приема.

В дополнение к протоколу RTP были разработаны несколько вспомогательных протоколов. Протокол SAP (Session Announcement Protocol) обеспечил получение информации о существующих широковещательных потоках данных. Протокол SDP (Session Description Protocol) описал транспортную адресацию, сжатие и схемы формирования пакетов, используемые отправителями и получателями широковещательных сессий.

Завершением данного процесса явилось разработка протокола SIP (Session Initiation Protocol), как достаточно простого способа нахождения участников и инициации самой широковещательной сессии.

Параллельно процессу разработки протоколов шел и процесс создания сетей ISDN (Integrated Services Digital Network) и соответствующего множества стандартов видеоконференций. Эти стандарты, основанные на рекомендациях ITU H.320, не имеют прямого отношения к развитию мультимедийных сетей, однако в их рамках были разработаны достаточно эффективные алгоритмы сжатия мультимедийной информации.

Стремительное развитие самой сети Интернет дало возможность реального использования потокового аудио и видео (например, в форматах RealAudio и QuickTime). Растущий рынок таких приложений потребовал стандартного механизма контроля содержимого потоковых данных, что привело к разработке протокола RTSP (Real-Time Streaming Protocol), который был стандартизован в 1998 году.

1. ВВЕДЕНИЕ В RTP

Вместе с основным протоколом RTP обычно используются дополнительные протоколы и стандарты, совместно обеспечивающие информирование о начале сессии, старт сессии и управление ей, сжатие данных и передачу их по сети.

На рис. 1.1 показана общая схема наборов протоколов IETF и ITU для передачи аудио и видеoinформации через Интернет [1].

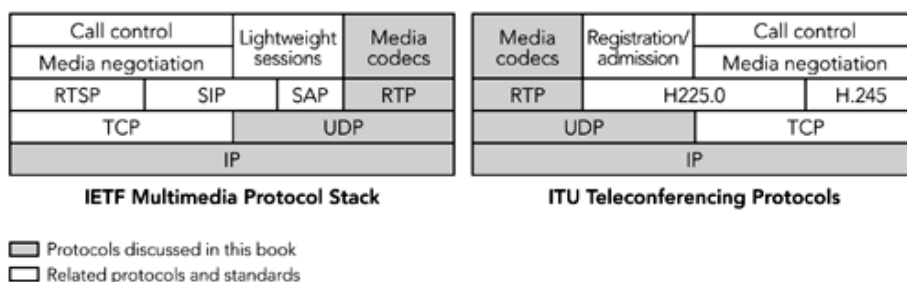


Рис. 1.1. Общая схема наборов протоколов IETF и ITU

1.1. Отсылка пакетов RTP

Отправитель информации по протоколу RTP (*RTP sender*) отвечает за захват (*capture*) и преобразование аудиовизуальных данных для их передачи в процессе генерации пакетов RTP. Он может также принимать участие в процессе контроля ошибок (*error correction*) и загрузки канала (*congestion control*). На рис. 1.2 изображена блок-схема процесса отправки информации по протоколу RTP [1].

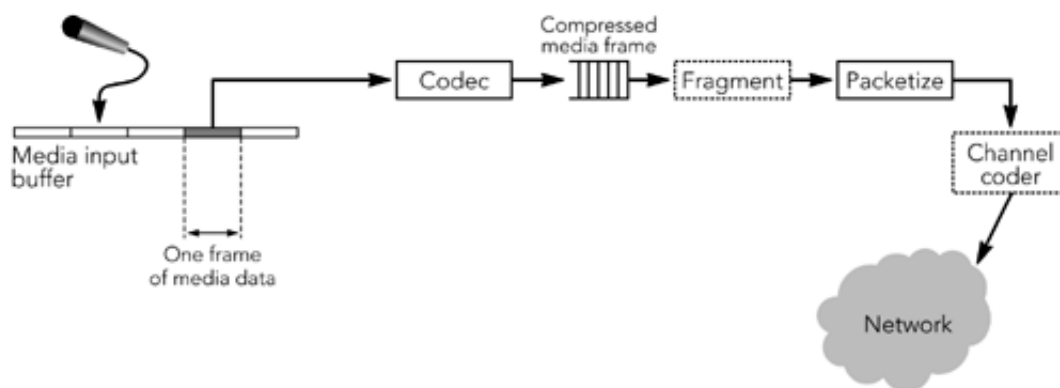


Рис. 1.2. Блок-схема процесса отправки информации по протоколу RTP

Несжатые звук или видео захватываются в буфер, из которого уже впоследствии создаются фреймы сжатой информации. Фреймы могут

быть закодированы различными способами в зависимости от типа используемого алгоритма сжатия и зависят как от предыдущих, так и от последующих порций данных.

Сжатые фреймы загружаются в пакеты RTP, готовые к отправке. Если фрейм слишком большой, он может быть разделен на несколько пакетов, а если фреймы небольшие, они могут быть объединены в один пакет. В зависимости от используемой схемы коррекции ошибок кодировщик канала (*channel coder*) может использоваться либо для создания дополнительных пакетов коррекции ошибок, либо для преобразования самих пакетов до их передачи.

После передачи пакетов буфер очищается. Отправитель не может отсечь данные, необходимые для контроля ошибок или кодирования пакетов. Отправитель также отвечает за периодическую генерацию отчетов о состоянии передаваемых данных, в том числе и для обеспечения последующей синхронизации звука и изображения. Кроме того, он получает информацию о качестве приема данных, которая может использоваться им для адаптации процесса пересылки следующих порций данных.

1.2. Получение пакетов RTP

Получатель пакетов RTP отвечает за собственно приемку пакетов по сети, компенсацию потерь информации, восстановление временной шкалы и сжатой информации и, наконец, представление результата пользователю. Он также пересылает сведения о качестве приема информации ее отправителю и поддерживает базу участников сессии.

Блок-схема процесса получения пакетов RTP показана на рис. 1.3.

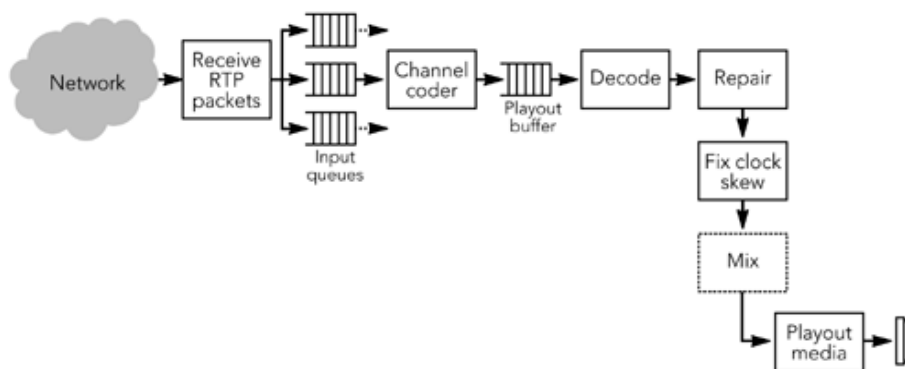


Рис. 1.3. Блок-схема процесса получения информации по протоколу RTP

Первым шагом процесса получения информации по протоколу RTP является сборка полученных пакетов, проверка их корректности и пересылка в зависящую от пользователя входную очередь. В соответствии с алгоритмом кодирования канала передачи пакеты вставляются в буфер

воспроизведения требуемого типа. Там они упорядочиваются в соответствии с временной шкалой и дополняются корректирующей информацией. Пакеты остаются в буфере до получения полного фрейма, после чего подвергаются дополнительной буферизации для восстановления временной шкалы воспроизведения. Вычисление количества добавляемых задержек является одной из самых важных задач данного процесса.

При достижении времени проигрывания пакеты группируются в виде полных фреймов, а поврежденные или отсутствующие фреймы восстанавливаются. На этой стадии может быть обнаружена разница в скорости проигрывания отправителя и получателя RTP пакетов. Получатель должен компенсировать эту разницу для предотвращения нежелательных пауз в процессе воспроизведения.

При воспроизведении в зависимости от формата мультимедийной информации и типа используемых устройств может возникнуть возможность воспроизведения каждого потока отдельно. Например, при получении нескольких потоков видео их можно проигрывать каждое в отдельном окне. Но может возникнуть и необходимость слияния нескольких потоков в один, например, при воспроизведении звука от нескольких источников на общем устройстве.

Из всего этого описания можно сделать вывод, что на уровне получения RTP пакетов используются более сложные операции, чем на уровне их отправителя. В большей степени это обуславливается необходимостью обнаружения и исправления ошибок передачи, а также устойчивой синхронизацией передаваемых данных.

2. ПАКЕТНАЯ ПЕРЕДАЧА ЗВУКА И ИЗОБРАЖЕНИЯ ПО СЕТИ

Перед тем, как приступить к детальному рассмотрению протокола RTP, необходимо ознакомиться со спецификой сети Интернет при передаче звука и изображения. В данной главе будут рассмотрены основные понятия, связанные с этим процессом.

2.1. TCP/IP и сетевая модель OSI

При работе с компьютерными сетями важно понимать концепцию уровней протоколов. Сетевая модель OSI, представленная на рис. 2.1, дает такую возможность.

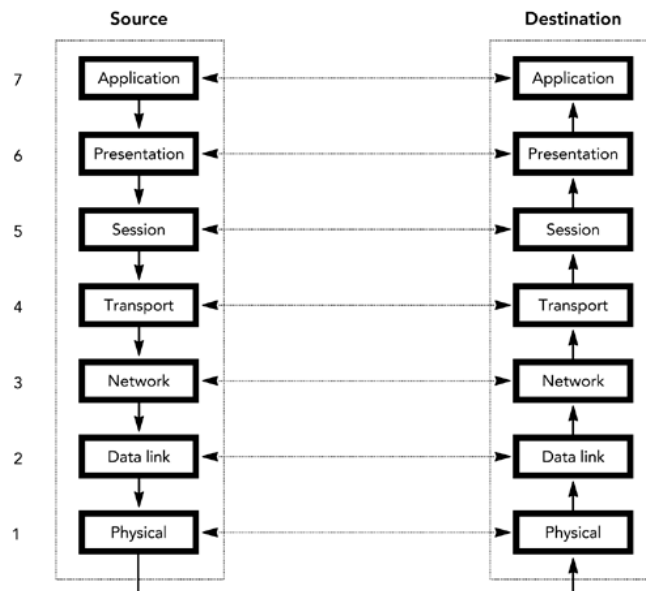


Рис. 2.1. Сетевая модель OSI

Функции уровней модели OSI:

1. **Физический уровень (Physical layer)**. Нижний уровень, включающий сетевые устройства и протоколы (кабели, коммутаторы и т. п.).
2. **Уровень управления передачей данных (Data link layer)**. Определяет правила совместного использования физического уровня узлами сети.
3. **Сетевой уровень (Network layer)**. Реализует функции маршрутизации пакетов, обработки ошибок, мультиплексирования пакетов и управления потоками данных. Если пакет адресуется рабочей станции в той же ЛВС, то он пересылается непосредственно, а если

он должен быть передан другой сети, то пересылается на маршрутизирующее устройство.

4. **Транспортный уровень (Transport layer).** Четвертый из семи уровней модели OSI/ISO, отвечающий за адресацию и прохождение данных в сети. Обеспечивает получение всех пакетов в нужном порядке и контроль правильности передачи данных между сетевыми устройствами. Во время сеанса обе системы сами поддерживают передачу данных, так как между ними устанавливается виртуальная связь, аналогичная гарантированной телефонной коммутации.
5. **Сеансовый уровень (Session layer).** Отвечает за поддержание сеанса связи, позволяя приложениям взаимодействовать между собой длительное время. Уровень управляет созданием/завершением сеанса, обменом информацией, синхронизацией задач, определением права на передачу данных и поддержанием сеанса в периоды неактивности приложений. Синхронизация передачи обеспечивается помещением в поток данных контрольных точек, начиная с которых возобновляется процесс при нарушении взаимодействия.
6. **Уровень представления (Presentation layer).** Отвечает за преобразование протоколов и кодирование/декодирование данных. Запросы приложений, полученные с уровня приложений, он преобразует в формат для передачи по сети, а полученные из сети данные преобразует в формат, понятный приложениям. На этом уровне может осуществляться сжатие/распаковка или кодирование/декодирование данных, а также перенаправление запросов другому сетевому ресурсу, если они не могут быть обработаны локально.
7. **Прикладной уровень (Application layer).** Описывает взаимодействие прикладных программ с сетевой операционной системой. Организует санкционированный доступ к запрашиваемым ресурсам и определяет их достаточность. Использует услуги нижележащих уровней, но полностью изолирован от деталей применяемого сетевого оборудования.

На каждом уровне модели существует логическая связь между этим уровнем на одном хосте (от англ. host – хозяин, принимающий гостей – любое устройство, предоставляющее сервисы формата «клиент-сервер» в режиме сервера по каким-либо интерфейсам и уникально определенное на этих интерфейсах) и соответствующем уровне на другом. Когда приложение в одной системе устанавливает связь с приложением в другой системе, связь инициируется на всех уровнях источника и через физические каналы передается на стек протоколов приемника.

Например, *Web-приложение* в *HTML-представлении* доставляется с использованием *http-сессии*, *транспортируется* через TCP соединение по *IP-сети* и *уровню управления данными Ethernet* с использованием

физического кабеля с витой парой. Каждый шаг может рассматриваться как работа очередного уровня модели на стеке протоколов. Результатом будет передача Web-страницы от приложения на стороне Web-сервера приложению на стороне Web-клиента (браузера).

Этот процесс не всегда так прост. Может не быть прямого физического соединения между источником и получателем информации, вследствие чего она частично может оставаться на промежуточном хранении на gateway-системах.

Два нижних уровня сетевой модели OSI могут быть напрямую отнесены к глобальной сети Интернет, которая работает с широким спектром устройств (модемы, DSL, Ethernet, оптоволоконные каналы, беспроводные сети и спутники).

На сетевом уровне разрозненные сети объединяются в единую глобальную сеть на основе протокола IP. Сервис, обеспечиваемый этим протоколом для верхних уровней, достаточно прост – доставка пакетов датаграмм по указанному назначению. Но за простоту надо платить – Интернет не гарантирует время доставки пакетов. Пакеты, содержащие датаграммы IP, могут быть утеряны, переформированы, задержаны или повреждены на нижних уровнях. IP не решает эти проблемы, однако предоставляет ряд сервисов:

- **фрагментацию**, если размер датаграммы превышает максимально возможный;
- **поле времени жизни пакета (time-to-live)**, с помощью которого предотвращаются бесконечные циклы;
- **метку типа сервиса (type-of-service label)**, для управления приоритетами обработки пакетов;
- **идентификатор протокола верхнего уровня (upper-layer protocol identifier)**, для направления пакета на корректный транспортный уровень;
- **адресацию пункта доставки**, включая мультивещание для группы получателей и маршрутизацию пакетов до заданных пунктов.

Формат заголовка пакета IP, показывающий расположение отдельных информационных элементов, можно видеть на рис. 2.2.

Использование протокола IP позволяет пользователю воспринимать Интернет как единую сеть. Но в то же время Интернет остается объединением разнородных сетей, как проиллюстрировано на рис. 2.3.

Провайдеры сети Интернет разного уровня управляют своей частью глобальной сети, причем пропускные способности отдельных фрагментов сети могут существенно различаться.

В этой путанице отдельных сетей для каждого пакета, содержащий IP датаграмму, индивидуально определяется маршрут его доставки в

пункт назначения. Маршрутизаторы не отправляют пакеты немедленно, они могут накапливать их в очередях. Они могут даже отказаться от их передачи во время перегрузки линий. Маршрут доставки пакета может также изменяться при изменении самой сети.

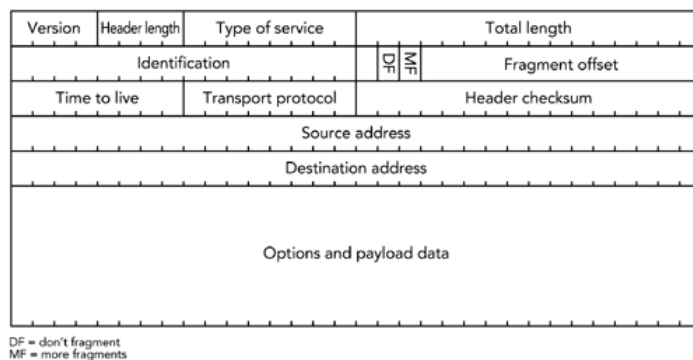


Рис. 2.2. Формат заголовка пакета IP

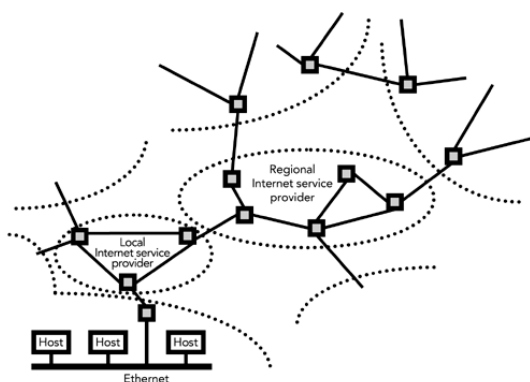


Рис. 2.3. Объединение сетей по протоколу IP

На верхнем уровне в архитектуре Интернет располагаются два общих транспортных протокола: TCP (Transmission Control Protocol) и UDP (User Datagram Protocol). TCP обеспечивает надежность доставки пакетов между хостами, варьируя скорость их доставки в соответствии с характеристиками сети. UDP, с другой стороны, обеспечивает тот же сервис, что и IP нижнего уровня, но с добавлением портов обслуживания.

Порты обеспечивают абстракцию, на основе которой различаются сервисы, запущенные на одном хосте. Многие сервисы имеют единственный и широко известный порт. Так в общем случае Web-сервисы работают на порту 80.

Оба перечисленных транспортных протокола используют обычные сеансовые протоколы, такие как HTTP при Web доступе или SMTP при передаче электронной почты. Стек протоколов завершается различными уровнями представления (HTML, MIME) и самими приложениями.

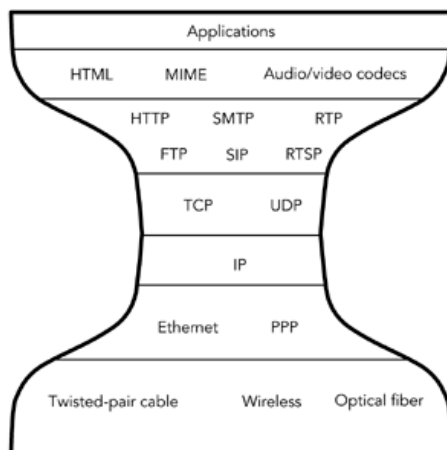


Рис. 2.4. «Модель песочных часов» архитектуры Интернет

Протокол IP играет ключевую роль в системе, обеспечивая уровень абстракции, скрывающий детали и топологию используемых сетей от приложений и изолируя нижние уровни от требований приложений. Эта модель известна как «модель песочных часов» (рис. 2.4).

2.2. Характеристики производительности сетей IP

Как видно из «модели песочных часов» архитектуры Интернет, от приложения скрыты детали нижних уровней за счет абстракций IP. Это означает, что невозможно определить типы сетей, через которые проходят пакеты и уровни их загруженности. Единственными средствами определения производительности сетей в этих условиях остаются наблюдение и измерение.

Что необходимо для измерения производительности сети и как это измерение выполнить? К счастью, на уровне IP число параметров ограничено и может быть подстроено под требования приложения. На этом этапе самыми важными вопросами являются:

- Какова вероятность потери пакета?
- Какова вероятность повреждения пакета?
- Каково время прохождения пакета по сети? Это время постоянно или меняется?
- Каков максимальный размер пакета?
- С какой скоростью надо отправлять пакеты?

Что влияет на измерения? Очевидным фактором этого рода является местоположения пункта измерений. Измерение по локальной сети даст существенно другие результаты, чем измерение по трансатлантическому каналу. Но географический фактор не является единственным. Важно, сколько промежуточных узлов глобальной сети проходит пакет на своем пути.

Также необходимо принять во внимание используемый тип сети, ее существующую загрузку. В настоящее время наиболее широко используется протокол TCP, поэтому можно сделать следующие выводы:

- Так как инфраструктура таких сетей достаточно надежна, то потери в них являются в основном следствием их перегрузки.
- В TCP потеря пакетов является сигналом о наступлении перегрузки сети, для предотвращения которой снижается скорость отсылки пакетов. Поток TCP увеличивает скорость отсылки пакетов до первой потери, после чего снова ее снижает. Таким образом, загрузка сети поддерживается на уровне, близком к максимальному.

Если структура сети или ее загрузка меняется, то становятся важными другие источники потерь. Таким источником может быть резкий рост числа пользователей сети или увеличение доли передаваемой мультимедийной информации.

2.3. Измерение производительности сетей IP

Приводимые в этом разделе числовые характеристики основаны на широком спектре исследований, опубликованных в Интернет.

Ранние исследования на эту тему давали разброс *среднего числа потерянных пакетов* в пределах 3...17 % (1994–1995) [2], 1.38–11.03 (1997–1998) [3]. Естественно, что техническое развитие сетей должно уменьшать риск потери пакетов, но до сих пор он далек от нулевого значения. В настоящее время по США среднее число потерянных пакетов составляет около 2 %, а по всему Интернету – около 3 % [1].

Если же рассматривать *вероятность потери пакетов*, то исследования показали, что она существенно зависит от уровня потерь предыдущих пакетов [2], то есть частота потерь не является независимым процессом.

При обнаружении потери пакета он вновь передается в глобальную сеть. Слишком частое *дублирование пакетов* свидетельствует о наличии серьезных проблем в транспортной сети.

Кроме потери пакетов, может быть обнаружено и их повреждение. Для этой цели пакеты хранят контрольные суммы, по которым отслеживается целостность пакетов. *Вероятность повреждения пакетов* существенно зависит от типа используемой сети. Так для беспроводной сети она будет существенно выше, чем для обычной сети вследствие повышения уровня помех. Поврежденные пакеты уничтожаются, а приложение воспринимает их как потерянные пакеты.

Относительно *времени прохождения пакетов* по глобальной сети следует отметить, что она существенно зависит от выбранного маршрута передачи. На это время влияет несколько факторов: скорость передачи между отдельными узлами, количество промежуточных узлов и сте-

пень загруженности маршрутизаторов. Более короткий физически маршрут может содержать большое количество загруженных промежуточных узлов. Поэтому чаще всего в сетях под более коротким путем подразумевается маршрут с меньшим количеством промежуточных узлов независимо от его физической длины.

Само вычисление времени прохождения пакетов не всегда просто вычитание времени отправления пакета от времени его получения, так как следует учитывать разницу в синхронизации часов отправителя и получателя, что не всегда технически возможно.

Еще один из отрицательных результатов передачи пакетов по глобальной сети – нарушение их порядка. Но проблема эта легко разрешима в связи с наличием нумерации самих пакетов.

Уровень IP может допускать работу с пакетами разных размеров, вплоть до 65535 байт, или быть ограниченным максимальным размером передаваемого блока (MTU – maximum transmission unit). Для сети Ethernet MTU равен 1500 байт, для модема 576 байт, но для большинства современных сетей он равен 1500 байт или выше. Протокол IP поддерживает сегментацию, позволяющую разбить большой пакет на части в соответствии с размером MTU используемой сети.

Групповая передача (*multicast*) по протоколу IP позволяет передавать информацию нескольким получателям одновременно, причем по каждой сети передается только одна копия пакета. В результате групповая передача ведется очень эффективно, а ее скорость не зависит от числа участников группы. Проблемы, возникающие при измерении производительности групповой передачи, будут рассмотрены в следующих главах этого учебного пособия.

На измерение производительности IP сетей существенно влияет и тип самих сетей. Сети Интранет, являющиеся частью глобальной сети Интернет, могут быть специально сконструированы для передачи мультимедийной информации, что практически сводит к нулю задержки и вероятность перегрузки такой сети.

2.4. Специфика использования транспортных протоколов

Рассмотренные ранее характеристики производительности сетей рассматривались на уровне протокола IP. Однако реальные приложения работают на более высоких уровнях, например, на уровне протоколов TCP или UDP. Какую специфику это вносит в измерение характеристик производительности?

Протокол **UDP** (User Datagram Protocol) содержит небольшое количество расширений протокола IP. На рис. 2.5 показан заголовок пакета UDP общей длиной 64 бита.

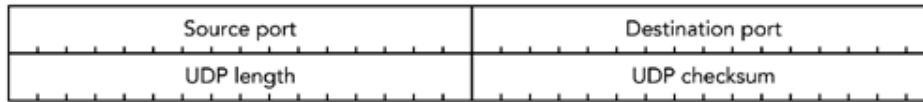


Рис. 2.5. Заголовок пакета UDP

Элементы *Source port* и *Destination port* определяют конечные точки маршрута пакета, разрешая мультиплексирование различных сервисов на разных портах. Значение *UDP length* добавляется к значению соответствующего параметра заголовка пакета IP. Контрольная сумма *UDP checksum* обеспечивает контроль качества передачи пакета и опциональна для приложений.

В целом протокол UDP не вносит существенной специфики в измерение характеристик передачи данных.

Протокол *TCP* является одним из самых распространенных протоколов при передаче данных в глобальной сети. Его характерные особенности:

1. **Надежность передачи данных.** Если канал передачи данных остается доступным, то весь поток данных будет передан получателю. Если же канал выходит из строя, то получатель будет оповещен о проблемах передачи, чего не делается в протоколе UDP.
2. **Возможность вычисления производительности.** Отправитель имеет возможность сообщить о завершении пересылки данных, поэтому есть возможность вычисления значения производительности сети.
3. **Возможность перегрузки сети.** При перегрузке сети (потере пакетов) в соответствии с протоколом TCP снижается скорость передачи информации в данном потоке, однако это может привести к потерям в другом потоке.

В целом следует отметить, что протокол TCP обеспечивает надежность передачи информации по сети, а протокол UDP – минимизацию временных задержек.

2.5. Требования по передаче звука и изображения в пакетных сетях

Под термином *передача данных в реальном времени (real-time)* будем в дальнейшем понимать тот факт, что полученная мультимедийная информация проигрывается сразу после получения, а не сохраняется полностью в файле до его проигрывания. В идеальном случае проигрывание происходит немедленно и синхронно, но на практике часто появляются задержки.

Первое требование в данном случае – предсказуемость времени передачи мультимедийных данных. Второе (менее жесткое) – надежность передачи данных. Меньшая жесткость обуславливается требованиями приложения, которое может допускать минимальный уровень потерь.

Эти требования диктуют выбор транспортного протокола. TCP/IP не подходит из-за частых временных задержек. UDP/IP в этом смысле более приемлем. Стандартный протокол RTP (Real-time Transport Protocol) основан на стеке протоколов UDP/IP и обеспечивает восстановление временных характеристик и обнаружение потерь при передаче мультимедийных данных. Этот протокол детально обсуждается в следующих главах учебного пособия.

Несмотря на недостатки протокола TCP/IP при передаче звука и изображения, некоторые приложения его все же используют. Они вычисляют значение возможных задержек по каналу и адаптируют к ней скорость передачи данных. В первую очередь такой выбор обусловлен легкостью прохождения потока TCP/IP через различные файрволлы, в отличие от потока UDP/IP. Но ситуация со временем меняется, так как файрволлы становятся все более интеллектуальными и все чаще пропускают потоки UDP/IP.

Следует также отметить несомненные преимущества передачи звука и изображения по пакетным сетям. Первым из них является то, что передача пакетов идет единообразно для всех видов информации. В результате существенное снижение затрат на обслуживание таких сетей. Кроме того, появляется возможность мультиплексирования каналов, что повышает степень их использования. Вторым преимуществом выступает экономичная система группового вещания, не приводящая к резкому увеличению нагрузки на каналы. Третьим преимуществом можно отметить развитие новых сервисов в пакетных сетях, направленных на повышение интерактивности работающих в них приложений.

3. ПЕРЕДАЧА ЗВУКА И ИЗОБРАЖЕНИЯ ПО ПРОТОКОЛУ RTP

3.1. Базовые принципы RTP

Перед создателями протокола RTP стояла задача разработки механизма надежной доставки мультимедийной информации по ненадежным сетям. Они достигли этой цели на основе двух принципов: *фрагментирования информации на уровне приложений и принципа конечных точек*.

Фрагментирования информации на уровне приложений позволяет учитывать специфику передаваемой информации и, как следствие, уменьшать число ошибок передачи. Только приложение может оценить существенность возникшей ошибки и принять решение о необходимости повторной пересылки фрагмента или другого способа устранения обнаруженной ошибки.

Что касается принципа конечных точек, то он означает, что ответственность за надежность передачи информации возлагается на ее отправителя и получателя, но не на промежуточные пункты маршрута передачи. Такой подход позволяет значительно упростить механизм передачи данных для промежуточных пунктов, исключая операции обнаружения и исправления ошибок передачи.

3.2. Стандартные элементы RTP

Стандарт RTP был опубликован в январе 1996 года (RFC 1889). Протокол состоит из двух частей – *протокола передачи данных* и связанного с ним *протокола управления*. Первый отвечает за доставку данных реального времени между конечными пунктами маршрута. Он включает порядковый номер пакета для предотвращения потерь пакетов, метку времени для восстановления временных характеристик, SSRC- и CSRS-идентификаторы, маркер существенных событий в передаваемом потоке. В деталях протокол будет рассмотрен позже.

Протокол управления RTP (RTP control protocol – RTCP) обеспечивает получение информации о качестве принимаемых данных и идентификаторе получателя, а также синхронизацию потоков. Более детально протокол RTCP также будет обсуждаться в дальнейших главах.

Отметим ограничения, вводимые протоколом RTP. Во-первых, стандарт не определяет алгоритмов проигрывания аудио или видео данных, восстановления временных характеристик, синхронизации между мультимедийными потоками.

тимедиа потоками, обнаружения и устранения ошибок, контроля перегрузок сети. Все это остается в сфере деятельности разработчика приложения, а так как разные приложения предъявляют разные требования в этой области, то стандартизировать их было бы ошибкой.

Во-вторых, некоторые характеристики передачи остаются открытыми и могут отражать специфику формата передаваемых данных (единица временной шкалы, отметки различных событий в рамках передаваемого потока и т. д.).

Протокол RTP также определяет формат передаваемых данных (*payload format*), ссылка на который есть в профиле RTP (*RTP profile*). Профиль может задавать некоторые общие свойства используемых форматов данных. В целом данная структура позволяет работать с различными кодеками, используемыми для сжатия передаваемой информации.

Кроме всего перечисленного, существуют форматы передаваемых данных, содержащие схемы коррекции данных. Эта тема также будет детально рассмотрена в следующих главах учебного пособия.

В заключение отметим, что элементами формата RTP могут быть два опциональных элемента: сжатие заголовка (*header compression*) и мультиплексирование (*multiplexing*).

3.3. Связанные с RTP стандарты

Протокол RTP связан еще с рядом стандартов. Полный стек протоколов показан на рис. 3.1.

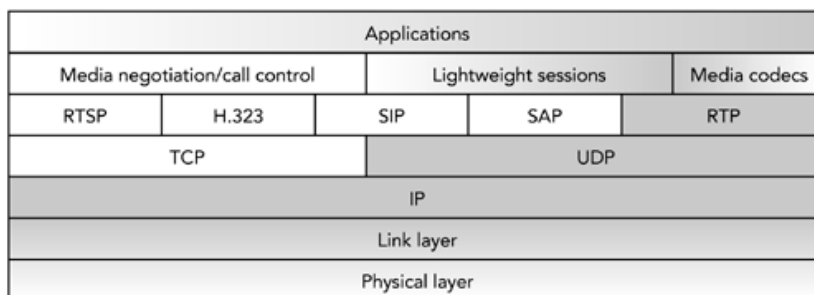


Рис. 3.1. Полный стек протоколов, связанных с RTP

При старте RTP сессии в зависимости от сценария приложения могут быть использованы дополнительные протоколы.

При старте *интерактивной сессии* (голосовом телефонном вызове или телеконференции) используются два протокола. Основными являются рекомендации ITU **H.323**, а в последнее время к ним добавляют протокол **SIP** (Session Initiation Protocol).

При старте *не интерактивной сессии* (просмотре видео по сети) используется протокол **RTSP** (Real-Time Streaming Protocol).

При старте *упрощенной модели конференции* (одностороннее вещание) используется в чистом виде протокол **RTP** с мультивещанием по протоколу **IP**. Для анонсирования конференции добавляется протокол **SAP** (Session Announcement Protocol).

Для настройки сессии часто используется протокол **SDP** (Session Description Protocol). Независимо от формата описания сессии, есть ряд параметров, которые обязательно должны быть в него включены. Необходимо передавать транспортные адреса потоков, формат и профиль данных, время активности сессии и ее цель.

SDP объединяет все эти параметры в файле текстового формата, сформированном в простой и понятной для анализа форме. В некоторых случаях этот файл направляется непосредственно в приложение **RTP**, давая достаточную информацию для прямого подключения к сессии. В других случаях протокол служит базой для переговоров о включении в проходящую сессию с более жестким контролем участников. Более детально протокол **SDP** будет рассмотрен в дальнейших главах пособия.

Хотя протокол **RTP** предназначен для работы с высокоэффективными сервисами, обеспечиваемыми протоколом **IP**, иногда бывает полезным резервирование сетевых ресурсов для обеспечения повышенного качества сервисов для потоков **RTP**. Для этих целей в настоящее время применяются две основные структуры — интегрирующие и дифференцирующие сервисы.

Интегрирующие сервисы используют протокол **RSVP** (Resource ReSerVation Protocol). Перед началом сеанса хост передает маршрутизатору информацию, необходимую для резервирования, что повышает качество передачи информации.

Дифференцирующие сервисы добавляют в сами пакеты информацию о требуемом сервисе. Это позволяет маршрутизатору расставить приоритеты для сервисов, что тоже снижает вероятность потери передаваемой информации.

4. ОПИСАНИЕ ПРОТОКОЛА RTP

4.1. RTP сессии

Сессия состоит из группы участников, обменивающимися данными по протоколу RTP. Каждый участник может присутствовать в нескольких RTP сессиях одновременно. Сессия идентифицируется сетевым адресом и парами портов отправления и получения информации, причем эти пары могут быть одинаковыми.

Каждая пара состоит из двух смежных портов: порта с четным номером для пакетов данных RTP и порта с нечетным номером (на единицу больше) для управляющих пакетов RTCP. По умолчанию пара портов 5004 и 5005 определена для UDP/IP, но многие приложения в процессе настройки сессии переопределяют эти стандартные номера. RTP сессия разработана для передачи только одного типа мультимедийной информации, поэтому для каждого такого типа организуется своя сессия.

Сессия может быть односторонней (либо напрямую между двумя участниками, либо в виде центрального сервера, транслирующего информацию) или широковещательной (группа участников).

4.2. Структура пакета RTP

Формат пакета RTP показан на рис. 4.1.

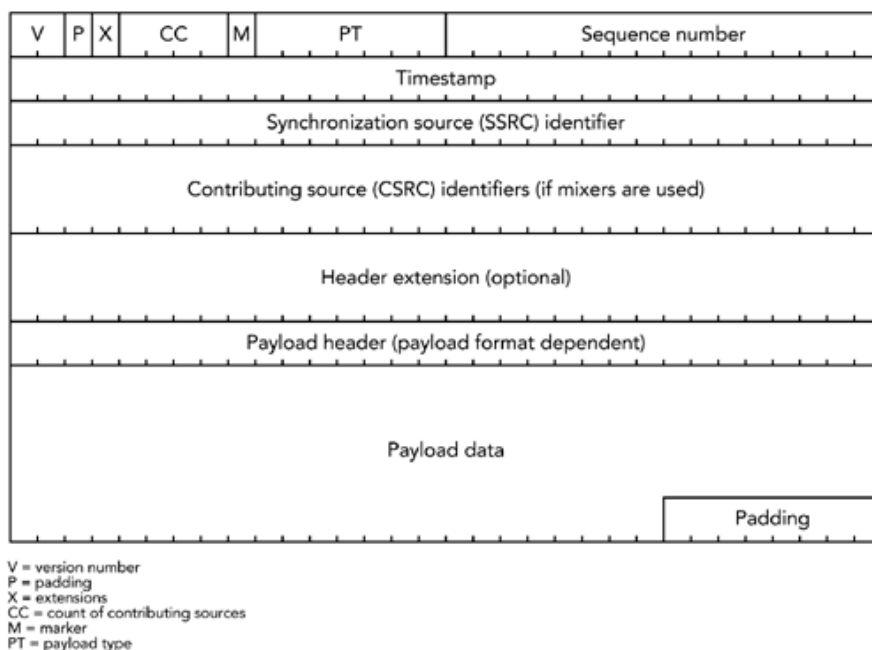


Рис. 4.1. Формат пакета RTP

Он состоит из четырех основных частей:

- обязательного RTP заголовка;
- опционального расширения заголовка;
- опционального заголовка данных (зависящего от формата данных);
- самих данных.

Обязательный заголовок пакета RTP имеет обычно 12 байт длины, хотя он может еще содержать список источников потока (**contributing source list**) несколькими порциями по 4 байта (всего до 60 байт). Полями заголовка являются: тип данных, порядковый номер пакета, метка времени (**time-stamp**) и идентификатор синхронизации источника (**synchronization source identifier**). Дополнительными элементами могут быть счетчик источников потока (**count of contributing sources**), маркер интересных событий, поддержка заполнения свободных мест (**support for padding**) и поле расширения заголовка (**header extension**), а также номер версии.

Тип передаваемых данных (**payload type** или **PT**) идентифицирует один из мультимедийных типов. Принимающее приложение по значению этого типа определяет способ обработки данных, например, их распаковку. Точная интерпретация поля типа данных осуществляется с помощью RTP профиля, содержащего все характеристики типа.

Многие приложения используют для проведения аудио и видео конференций RTP профиль с минимальным контролем [4]. Этот профиль определяет таблицу соответствия между номером типа данных и детальным описанием его формата. Примеры соответствия приведены в табл. 4.1.

Таблица 4.1

Примеры соответствия номера типа данных и его формата

Payload Type Number	Payload Format	Specification	Description
0	AUDIO/PCMU	RFC 1890	ITU G.711 μ -law audio
3	AUDIO/GSM	RFC 1890	GSM full-rate audio
8	AUDIO/PCMA	RFC 1890	ITU G.711 A-law audio
12	AUDIO/QCELP	RFC 2658	PureVoice QCELP audio
14	AUDIO/MPA	RFC 2250	MPEG audio (e.g., MP3)
26	VIDEO/JPEG	RFC 2435	Motion JPEG video
31	VIDEO/H261	RFC 2032	ITU H.261 video
32	VIDEO/MPV	RFC 2250	MPEG I/II video

Типы данных с 96 по 127 зарезервированы под динамические описания. Независимо от того, статическое или динамическое описание ти-

па данных используется, для работы приложения необходимо описать сессию. Для этих целей обычно используется протокол SDP (**Session Description Protocol**). Пример описания сессии:

```
v=0
o=bloggs 2890844526 2890842807 IN IP4 10.45.1.82
s=-
e=j.bloggs@example.com(Joe Bloggs)
c=IN IP4 224.2.17.12/127
t=2873397496 2873404696
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 98
a=rtpmap:98 H263-1998/90000
```

Жирным шрифтом в примере выделены параметры, имеющие отношение к обсуждаемым в этом разделе вопросам. В примере описываются две RTP сессии. Аудио посылается по адресу **224.2.17.12** на порт **49170** с параметром **time-to-live=127**. Видео посылается по тому же адресу на порт **51372**. Как видео, так и аудио используют транспортные протоколы **RTP/AVP**, которые основаны на RTP профиле с минимальным контролем. Для аудио используется тип данных с номером **0** (статический формат данных AUDIO/PCMU), а для видео – с номером **98** (динамический формат, определяемый значением параметра **a=rtpmap:98 H263-1998/90000**).

Выбор формата данных имеет и другие последствия. Он определяет скорость часов для RTP потока. Для статического случая эта скорость описана в профиле, а для динамического – в самом описании. В приведенном примере задана скорость **90000** килогерц.

RTP сессия не требует использования единственного формата данных, их может быть несколько. Более того, формат данных может меняться в течение самой сессии.

Порядковый номер пакета используется для идентификации пакетов. Это беззнаковое шестнадцатитбитное целое, которое увеличивается на единицу для каждого следующего пакета. При достижении максимально возможного значения оно сбрасывается в ноль. Для обычной передачи голоса 20-миллисекундными пакетами сбрасывание происходит примерно через каждые 20 минут.

Это означает, что приложение не может использовать порядковый номер пакета как его уникальный идентификатор. Для этих целей рекомендуется пользоваться расширенным порядковым номером пакета (32 бита или больше), где младшие 16 бит представляют собственно порядковый номер пакета. Расширенный порядковый номер вычисляется в приложении немедленно после получения пакета.

Метка времени отмечает время для первого байта пересылаемых данных и используется для их воспроизведения. Это 32-битное беззна-

ковое целое, которое увеличивается в соответствии с заданной скоростью и сбрасывается в ноль при достижении максимального значения. При работе с типичными видео-кодеками при скорости 90 кГц очередной сброс происходит примерно каждые 13 часов. При скорости 8 кГц интервал сброса значения составляет примерно 6 дней.

Если в одном потоке последовательно передаются два клипа, то метки времени образуют возрастающую последовательность и не сбрасываются с началом нового клипа.

Идентификатор синхронизации источника (SSRC) идентифицирует участников RTP сессии. Это 32-битное целое, выбираемое случайным образом участниками при подключении к сессии. Так как выбор происходит локально, возможно существование одинаковых SSRC в одной сессии. Эта ситуация может быть обнаружена при пересылке данных участниками сессии между собой. В этом случае один из участников сессии должен сменить свое значение SSRC. Механизм такой смены будет описан далее в главе 5. Для снижения вероятности таких коллизий следует использовать датчики случайных чисел, не основанные на датчике времени.

В нормальных условиях RTP данные генерируются одним источником, но когда несколько RTP потоков проходят через миксер или транслятор, в RTP пакете появляется несколько источников. **Список источников потоков (CSRCs)** идентифицирует участников сессии, вносящих свой вклад в общий поток, но не отвечающих за его временные характеристики и синхронизацию. Длина списка задается параметром **СС** в заголовке пакета. Работа с такими пакетами будет обсуждаться далее в этой главе в разделе «Миксеры».

Бит **маркера (M)** в заголовке пакета используется для выделения интересных событий в передаваемом потоке. Его точное значение определяется RTP профилем. Для звуковых потоков с минимальным контролем этот бит устанавливается в единицу для первого пакета, посылаемого после периода молчания. Для потоков видео с минимальным контролем бит устанавливается в единицу для пакетов, содержащих последнюю часть очередного фрейма.

Бит **заполнения (P)** в заголовке пакета служит индикатором дополнения исходных данных перед их пересылкой. В этом случае последний байт передаваемых данных содержит количество добавленных для заполнения байтов (рис. 4.2).

Каждый RTP пакет содержит **номер версии** в поле **V**. Он используется для проверки правильности передачи данных.

Протокол RTP допускает наличие **полей расширения заголовка**, наличие которых отмечается значением единицы в бите **X**. Эти расши-

рения заголовка размещаются после обычного заголовка пакета RTP, но перед заголовком передаваемых данных и самими данными. Данная опция на практике используется достаточно редко.

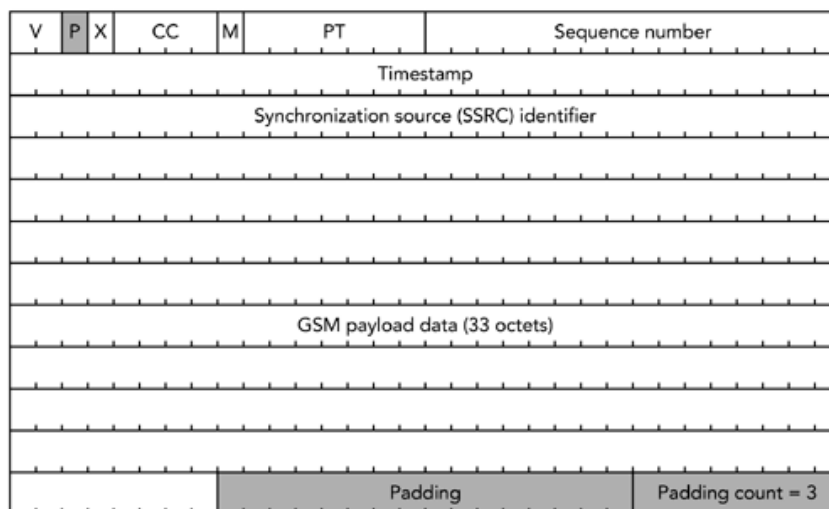


Рис. 4.2. Структура пакета RTP с тремя добавленными байтами

Обязательная часть **заголовка данных** содержит общую информацию для всех типов данных. Информация о конкретном типе данных размещается в дополнительном заголовке данных. Общая информация может быть статической и динамической. Динамические части конфигурируются по протоколу SDP, определяющего параметры формата, которые могут относиться к одной из трех следующих категорий:

- сигнализирующие о присутствии или отсутствии полей заголовка данных, их размере и формате;
- определяющие использование отдельных полей;
- используемые вместо параметров заголовка данных.

Непосредственно после заголовка данных следуют один или несколько фреймов **передаваемых данных**. Многие форматы данных разрешают передачу нескольких фреймов данных в одном пакете. Существует два способа определения получателем количества фреймов в полученном пакете. Если фреймы фиксированной длины, то их количество можно вычислить по размеру данных. Если фреймы переменной длины, то в самих фреймах предусматривается индикатор, содержащий информацию о размере фрейма. Так как обычно ограничения на количество фреймов не накладываеся, то следует учитывать ограничение на размер передаваемого пакета (**MTU, maximum transmission unit**). Пакеты, превышающие это значение, либо фрагментируются, либо отбрасываются. Сама фрагментация относится к нежелательным действиям, так как при потере любого фрагмента приходится повторять их все.

4.3. Проверка качества пакета

Так как RTP сессии обычно используют динамические пары портов, то очень важно определять, что полученный пакет действительно является пакетом RTP. На первый взгляд, эта задача не тривиальна, так как пакет не содержит обозначения протокола. Однако наблюдая последовательность полей заголовка в нескольких пакетах, можно быстро определить условие принадлежности пакета к RTP потоку.

Существует два типа таких тестов:

- **пакетная проверка**, основанная на знании полей заголовка;
- **потокковая проверка**, основанная на знании структуры полей заголовка.

Первый тип проверки больше подходит для проверки дефектных пакетов, но подразумевает дополнительную нагрузку на получателя данных. Выгоднее всего сначала провести проверку второго типа, и только потом для оставшихся пакетов проводить проверку первого типа.

Также возможно проводить проверку качества пакетов с помощью контрольных пакетов RTCP, но такая проверка влечет за собой задержки в передаче потока, что часто недопустимо.

4.4. Трансляторы и миксеры

В дополнение к нормальным конечным системам протокол RTP промежуточные обработчики, которые работают с потоком в процессе его передачи. Определены два класса таких обработчиков – **трансляторы** и **миксеры**.

Транслятором называется промежуточная система, работающая с RTP данными во время создания источника синхронизации и временной схемы потока данных. Примером являются системы, осуществляющие перекодировку между форматами без их смешивания, являясь мостом между разными транспортными протоколами или фильтрами этого потока. Трансляторы невидимы для конечных RTP систем, если у них нет предварительной информации о кодировке данных. Известные типы трансляторов:

- **Мосты (bridges)**. Этот тип трансляторов не меняет кодировки данных, поэтому чаще всего не изменяет передаваемого потока. Примером являются различные межсетевые интерфейсы (**gateways**).
- **Транскодеры (Transcoders)**. Основная функция транскодеров – преобразование кодировки данных в соответствии со спецификой используемой сети.
- **Эксплодеры (Exploders)**. Эксплодеры (множители) из одного пакета делают несколько.
- **Мерджеры (Mergers)**. Из нескольких пакетов делают один.

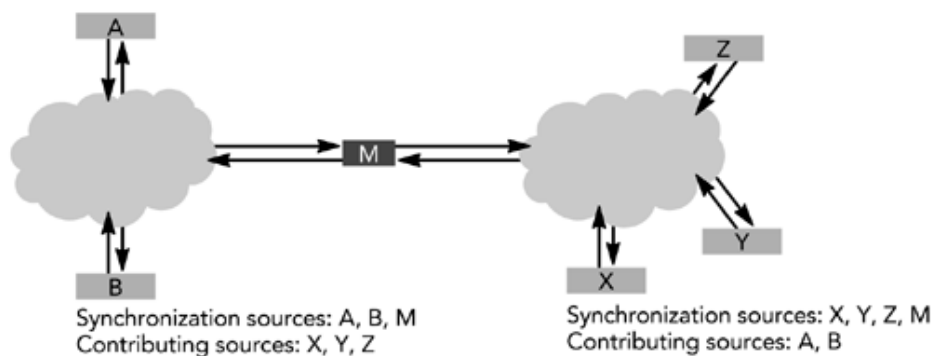


Рис. 4.3. Система с миксером (M) и пятью участниками сессии (A,B,X,Y,Z)

Миксером называется промежуточная система, собирающая RTP пакеты из нескольких источников и генерирующая на их основе общий пакет, возможно с изменением кодировки данных. Так как временные характеристики входящих потоков обычно не синхронизированы, задача синхронизации ложится на миксер. Миксеры могут использовать буферы проигрывания для временного хранения данных потоков. Миксер имеет свой собственный SSRC, который он вставляет в формируемый пакет данных. Идентификаторы SSRC из входящих пакетов копируются в CSRC список выходного пакета. Пример системы с миксером приведен на рис. 4.3.

5. ОПИСАНИЕ ПРОТОКОЛА RTCP

Протокол RTP состоит из двух частей: протокола передачи данных, изложенного в предыдущей главе, и соответствующего ему управляющего протокола, который и рассматривается ниже. Управляющий протокол RTCP обеспечивает периодическую отчетность о качестве принятых данных, идентификацию получателей, извещение об изменении состава участников сессии, а также получение информации, необходимой для синхронизации потоков.

5.1. Компоненты RTCP

Реализация RTCP включает три части: форматы пакетов, временные правила и базу данных участников.

Пять стандартных форматов пакетов и правила управления частотой их пересылки будут описаны позже.

Каждое изменение отображается в базе данных участников сессии на основе информации, получаемой из пакетов RTCP. Эта база используется для составления периодических отчетов о качестве принятой информации и для синхронизации звука и изображения. Более детально эта тема будет описана в дальнейших разделах учебного пособия.

5.2. Передача пакетов RTCP

Каждая RTP сессия идентифицируется сетевым адресом и парой портов (для передачи RTP и RTCP пакетов). RTP порт имеет четный номер, а номер RTCP порта на единицу больше. Например, если данные передаются по протоколу на порт 5004, контрольный канал будет работать по порту 5005.

Все участники сессии посылают свои пакеты RTCP и получают их от других участников. Отметим, что информация обратной связи посылается *всем* участникам сессии, либо индивидуально с использованием транслятора, либо сразу всем в режиме мультивещания. В результате каждый участник сессии имеет полную информацию о других участниках: их присутствии, качестве приема и опционально – персональные данные, такие как имя, адрес электронной почты, местоположение и телефон.

5.3. Формат пакетов RTCP

Спецификацией RTCP определены пять стандартных форматов:

- отчет получателя **RR (Receiver Report)**;
- отчет отправителя **SR (Sender Report)**;
- описание источника **SDES (Source DEscription)**;

- управление участниками **BYE**;
- определяемый приложением **APP**.

Все эти форматы имеют общую структуру, показанную на рис. 5.1, хотя заносимая в поля информация отличается для каждого типа формата.

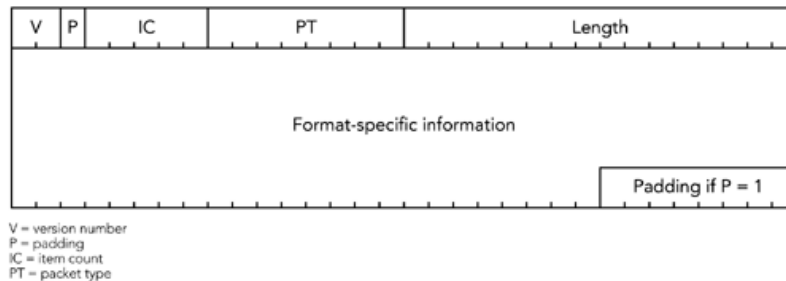


Рис. 5.1. Структура стандартного формата пакета RTCP

Заголовок пакета длиной 4 байта содержит 5 полей:

- номер версии **V (Version number)**, обычно равен 2;
- дополнение **P (Padding)**. Этот бит служит индикатором сделанной операции дополнения. Если он равен 1, то к концу пакета добавлен один или несколько дополнительных байтов, в последнем из которых записано их количество;
- количество элементов **IC (Item count)**. В каждый пакет может быть включено до 31 элемента. Если число элементов больше, они делятся на несколько пакетов. Те типы пакетов, которым не нужна данная информация, могут использовать это поле для других целей;
- тип пакета **PT (Packet type)**. Один из пяти стандартных типов;
- длина пакета (**Length**), следующего за заголовком (в 32-битных единицах).

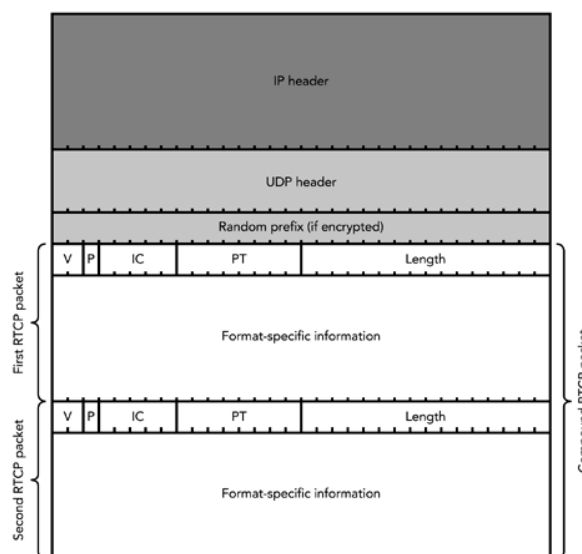


Рис. 5.2. Структура составного пакета RTCP

Пакеты RTCP никогда не передаются поодиночке, они всегда группируются перед передачей, формируя составные пакеты. Каждый составной пакет включает один пакет нижнего уровня (чаще всего UDP/IP) для транспортировки. Если составной пакет должен быть зашифрован, группа RTCP пакетов предваряется случайным 32-битным целым. Структура составного пакета изображена на рис. 5.2. Правила формирования групп пакетов RTCP будут изложены в соответствующем разделе.

5.3.1. Формат RTCP RR

Этому типу пакетов RTCP соответствует номер 201 в поле типа пакета. Структура формата показана на рис. 5.3.

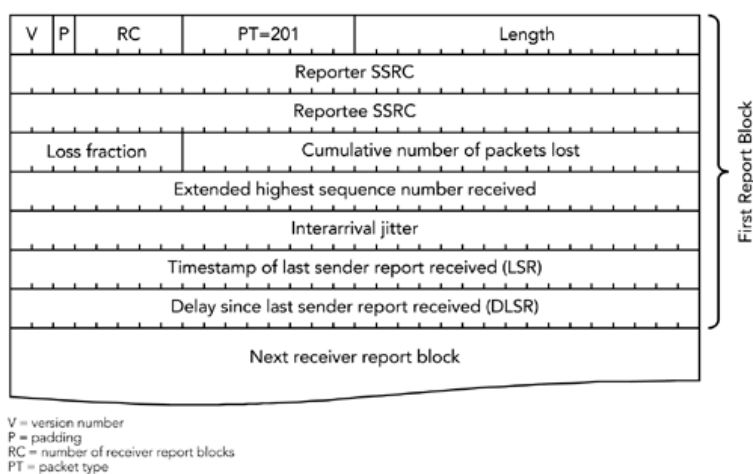


Рис. 5.3. Структура пакета RTCP RR

Пакет содержит идентификатор источника синхронизации (**reporter SSRC**), сразу за несколькими блоками отчетов, указанных в блоке. Каждый блок отчетов содержит информацию о качестве приема от одного источника синхронизации за текущий отчетный интервал. Таких блоков в пакете RTCP RR может быть не более 31. Если активных отправителей более 31, формируется составной пакет. Каждый отчет состоит из семи блоков общей длины 24 байта.

SSRC отправителя (**reportee SSRC**) идентифицирует участника, отправившего отчет.

Суммарное число потерянных пакетов (**cumulative number of packets lost**) является 24-битовым целым со знаком. Значение поля определяется вычитанием числа полученных пакетов из числа ожидаемых. Число ожидаемых пакетов определяется разницей порядковых номеров пакетов. Так как число полученных пакетов включает все опоздавшие и продублированные пакеты, то значение поля может быть и

отрицательным. Данное число подсчитывается за всю сессию, а не за отдельный ее интервал.

Определение значения параметра **extended highest sequence number received** будет обсуждаться в соответствующем разделе следующей главы.

Относительные потери (**loss fraction**) определяются делением числа потерянных за интервал пакетов на число ожидаемых пакетов.

Неустойчивость прибытия пакетов (**interarrival jitter**) вычисляется как дисперсия времени передачи пакетов по сети. Она измеряется в единицах метки времени, поэтому записывается в виде 32-битного беззнакового целого, как и метка времени RTP пакета.

Для вычисления дисперсии надо измерять времена передачи пакетов по сети. Так как отправитель и получатель не имеют синхронизированных часов, абсолютное время передачи замерить невозможно. Вместо него подсчитывается относительное время передачи пакетов как разница между значением метки времени пакета RTP и временем прибытия к получателю пакета, измеренном в тех же единицах. Для этого получателю приходится создавать таймер для каждого источника и запускать его с той же скоростью. Из-за отсутствия синхронизации между таймерами отправителя и получателя относительное время передачи по сети включает неизвестную константу сдвига по времени. Но этот факт не влияет на подсчет дисперсии, так как данная константа вычитается в процессе ее вычисления.

Если S_i – метка времени пакета i , а R_i – метка времени этого же пакета, то относительное время передачи пакета по сети вычисляется как $(R_i - S_i)$. Для двух пакетов i и j разница во времени передачи по сети составит

$$D(i,j) = (R_j - S_j) - (R_i - S_i)$$

Метка последнего отчета (**last sender report (LSR) timestamp**) записывается в средние 32 бита в 64-битном параметре NTP (**Network Time Protocol**). При отсутствии отчетов значение равно нулю.

Задержка после последнего полученного отчета (**delay since last sender report – DLSR**) вычисляется как разница между временем получения и отсылки отчета.

5.3.2. Интерпретация RR

Информация о качестве полученных пакетов полезна не только для их отправителя. Она позволяет скорректировать пересылку пакетов для более качественного приема. Другие же участники сессии могут определить, локальные или глобальные проблемы мешают передаче данных.

Отправитель может использовать поля LSR и DLSR для вычисления времени обмена информацией с каждым ее получателем. Пример такого вычисления показан на рис. 5.4 (пример взят из спецификации RTP).

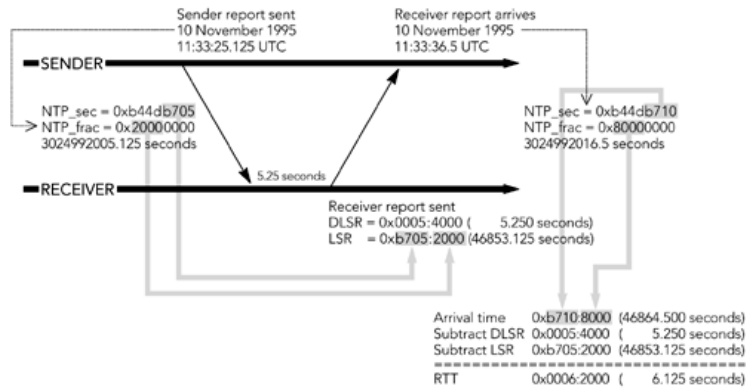


Рис. 5.4. Пример вычисления времени обмена информацией

Отметим, что это значение учитывает только время передачи по сети и не включает обработку данных в конечных точках. Получатель, например, может буферизовать данные перед их проигрыванием, чтобы исключить возможные нежелательные паузы.

Отправитель может использовать информацию о времени передачи в сети для оптимизации кодирования данных.

Наблюдая за динамикой относительных потерь, отправитель может определить, являются ли проблемы передачи временными или постоянными. Уровень потерь повлияет на выбор способа кодирования данных и включение схем корректировки ошибок. Неустойчивость прибытия пакетов может свидетельствовать о начале перегрузки сети, хотя выводы существенно зависят от топологии сети.

5.3.3. Формат RTCP SR

Этому типу пакетов RTCP соответствует номер 200 в поле типа пакета. Структура формата показана на рис. 5.5.

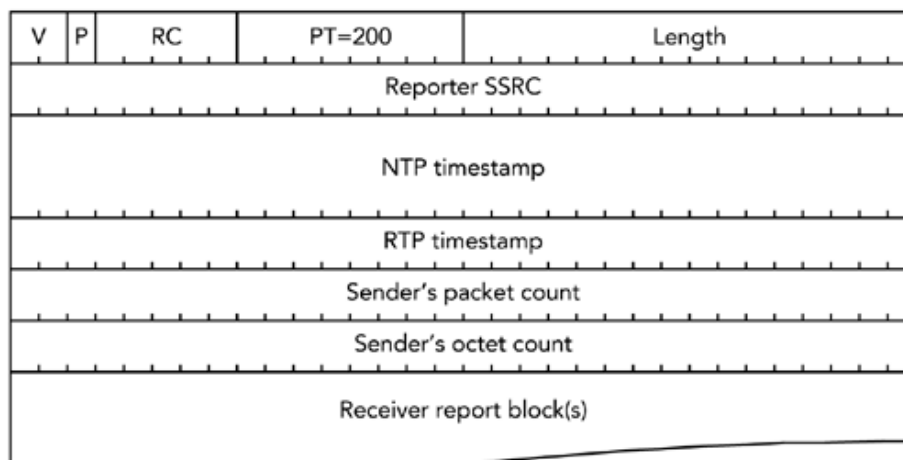


Рис. 5.5. Структура пакета RTCP SR

Пакет содержит 24 байта блока информации об отправителе, следующего за блоками отчетов получателя, количество которых задается полем RC. Блоки отчета получателя присутствуют, если отправитель одновременно является и получателем данных.

Поле метки времени NTP (**NTP timestamp**) содержит 64-битное целое без знака, соответствующее времени отсылки данного пакета. Верхние 64 бита содержат количество секунд от даты 1 января 1900 года, а нижние 32 бита – дробную часть значения секунд. Для преобразования значения метки времени в формате UNIX (где отсчет начинается с начала 1970 года) в этот формат следует добавить 2,208,988,800 секунд.

Метка времени RTP соответствует тому же моменту времени, что и метка времени NTP, но измеряется в медиачасах RTP. На рис. 5.6 показан пример метки времени NTP.

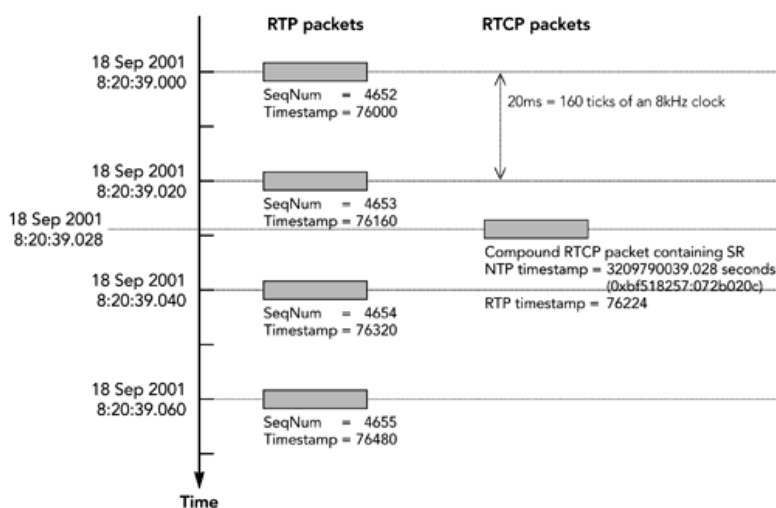


Рис. 5.6. Пример метки времени NTP

Количество пакетов отправителя (**sender's packet count**) равно количеству пакетов, которые данный источник синхронизации отправил с начала сессии. Счетчик байтов отправителя (**sender's octet count**) содержит число байтов данных, без учета заголовка и байтов дополнения. Значения обоих полей сбрасывается в нуль, если отправитель изменяет SSRC.

5.3.4. Интерпретация SR

С использованием информации SR приложение может вычислять среднюю скорость передачи данных в целом и по заданным интервалам без собственно передачи данных. Если предположить, что потери пакетов не зависят от их размера, то число пакетов, получаемых отдельным получателем, умножается на средний размер блока данных или самого пакета и показывает производительность передачи для данного получателя.

Метка времени чаще всего используется для установления соответствия таймера данных с внешними часами, например, для синхронизации изображения и звука, как будет более подробно рассмотрено далее в соответствующей главе.

5.3.5. Описание источника RTCP SDES

Описание источника **SDES (Source DEscription)** предназначено для передачи данных об источнике конечным потребителям. Эти данные включают идентификатор источника и некоторую его детализацию (местоположение, адрес электронной почты, номер телефона). Информация в SDES пакетах, введенная пользователем, обычно отображается в графических интерфейсах приложений, хотя это существенно зависит от типа приложения.

Формат пакета SDES показан на рис. 5.7.

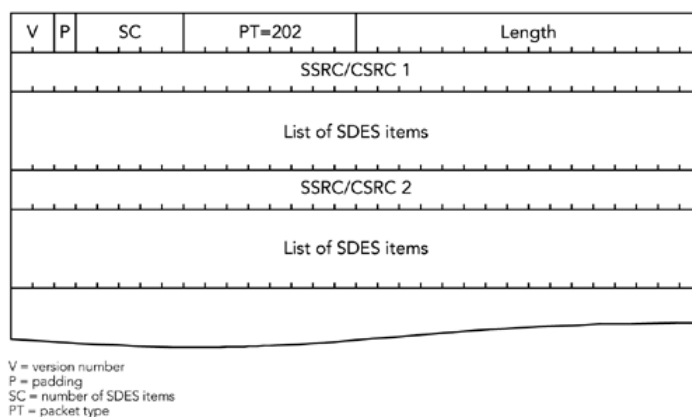


Рис. 5.7. Формат пакета SDES

Пакеты SDES содержат от нуля до нескольких списков SDES элементов. Их количество задано в поле SC заголовка. Для приложений возможна генерация пакетов с пустым списком SDES элементов, но обычно значение поля SC равно единице.

Каждый список SDES элементов начинается с SSRC описываемого источника. Формат SDES элемента показан на рис. 5.8.

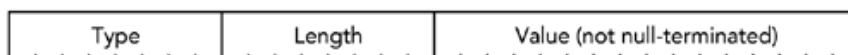


Рис. 5.8. Формат SDES элемента

Поле **Length** показывает, сколько байтов текста содержит описание источника. Само описание помещается в кодировке **UTF-8**. Все элементы пакета идут последовательно без разделителей и заполнителей. Спи-

сок элементов заканчивается одним или несколькими нулевыми байтами, первый из которых интерпретируется как элемент нулевого типа, обозначающий конец списка.

В RTP спецификации может быть описано несколько допустимых SDES элементов. Упомянутый нулевой тип обозначает конец списка, а другими могут быть CNAME, NAME, EMAIL, PHONE, LOC, TOOL, NOTE и PRIV.

Элемент **CNAME** (type=1) содержит каноническое имя каждого участника. Этот идентификатор стабилен и не зависит от источника синхронизации. Этот элемент может быть использован для ассоциации нескольких потоков мультимедиа от одного участника по разным RTP сессиям (например, для синхронизации звука и изображения) и является обязательным элементом SDES пакета.

CNAME назначается алгоритмически с использованием имени пользователя и IP адреса участника сессии (например, **csp@10.7.42.16** для IPv4). Если приложение работает в системе, не использующей имен пользователей, в данном элементе будет только IP адрес.

Элемент **NAME** (type=2) содержит имя участника и используется для изображения списка участников в графических интерфейсах. Это имя обычно вводится самим участником, поэтому может не быть уникальным.

Элемент **EMAIL** (type=3) содержит адрес электронной почты участника сессии. Посылающее его приложение не пытается проверить его корректность, оставляя эту задачу принимающему приложению.

Элемент **PHONE** (type=4) содержит номер телефона участника сессии. В спецификации RTP рекомендуется задавать полный международный номер (например, +1 918 555 1212 для участника из США).

Элемент **LOC** (type=5) указывает на местоположение участника сессии. Форма ввода данной информации зависит от используемого приложения.

Элемент **TOOL** (type=6) показывает инструментарий, используемый участником сессии. Поле используется для отладки и маркетинга и состоит из названия и номера версии инструментального средства. Пользователь обычно не может редактировать содержимое этого поля.

Элемент **NOTE** (type=7) дает возможность участнику занести короткую информацию произвольной тематики.

Элемент **PRIV** (type=8) используется для экспериментальных или зависимых от приложения SDES расширений. Текст элемента начинается с дополнительного байта и префиксной строки. Этот тип используется достаточно редко.

На рис. 5.9 приведен пример SDES пакета.

V	P	SC=1	PT=202	Length=10	
SSRC					
Type=1 (CNAME)		Length=15		c	s
p		@		1	0
.		7		.	4
2		.		1	6
9		Type=2 (NAME)	Length=13		C
o		l		i	n
		P		e	r
k		i		n	s
Type=0		0		0	0

Рис. 5.9. Пример SDES пакета

5.3.6. Управление участниками сессии RTCP BYE

RTCP обеспечивает возможность некоторого управления участниками сессии с помощью пакетов RTCP BYE, которые показывают, какие участники хотят покинуть сессию. Пакет BYE генерируется, когда участник либо покидает сессию, либо меняет свой SSRC. Этот пакет может быть потерян при передаче, а некоторые приложения его вообще не генерируют. Поэтому получатель должен быть готов к отключению по времени долго молчащих участников сессии, даже если от них не поступало пакетов BYE.

Пакеты BYE идентифицируются номером пакета 203 и имеют формат, показанный на рис. 5.10.

V	P	RC	PT=203	Length	
SSRC 1					
SSRC 2					
...					
SSRC n					
Optional length			Optional reason for leaving		

V = version number
P = padding
RC = number of SSRC headers
PT = packet type

Рис. 5.10. Пример пакета RTCP BYE

Поле RC в общем заголовке RTCP показывает число SSRC идентификаторов в пакете. Значение, равное нулю, допустимо, но бесполезно. При получении пакетов BYE приложение предполагает, что перечисленные в нем источники покидают сессию и должно игнорировать в дальнейшем все RTP и RTCP пакеты от них. Поэтому важно некоторое

время хранить список покинувших сессию участников, чтобы удалять пакеты от них.

Пакет BYE может также содержать текстовую информацию, объясняющую причины покидания сессии. Этот текст выводится в графическом интерфейсе для информирования остальных участников.

5.3.7. Пакеты, управляемые приложениями: RTCP APP

Последний класс RTCP пакетов разрешает расширения, определяемые приложениями. Это пакеты типа 204, с форматом, показанным на рис. 5.11.

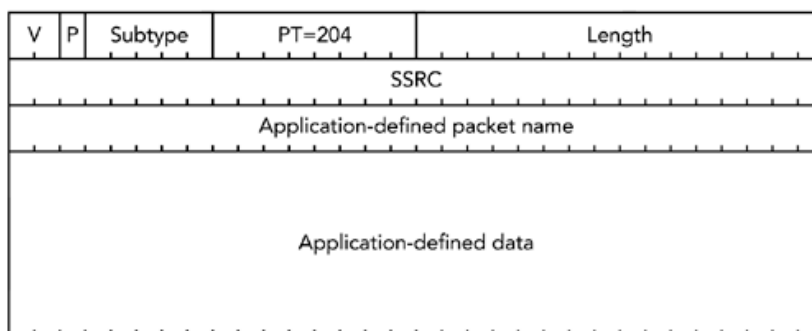


Рис. 5.11. Формат пакета RTCP APP

Определяемое приложением имя пакета (**application-defined packet name**) содержит четыре символа таблицы ASCII, идентифицирующих это расширение. Структура остальной части пакета определяется самим приложением.

Такой тип пакетов используется для отработки новых возможностей передачи данных по протоколу RTP.

5.3.8. Компоновка пакетов

Как уже отмечалось ранее, пакеты RTCP никогда не пересылаются поодиночке, они всегда компонуются в составные пакеты. При этом используются некоторые правила.

Если участник, создающий составной RTCP пакет, является активным источником данных, его составной пакет должен начинаться с пакета RTCP SR. В противном случае он начинается с пакета RTCP RR. Это должно быть даже при отсутствии самих отчетов.

Следующим пакетом в составном пакете должен быть пакет SDES. Как минимум, он должен содержать элемент CNAME.

Если пересылается пакет BYE, он должен быть последним в составном пакете. Промежуточные же пакеты RTCP могут помещаться в составной пакет в любом порядке.

5.4. Безопасность и конфиденциальность

При работе по протоколу RTCP есть несколько моментов, влияющих на сохранения конфиденциальности информации, например, использование пакетов описания источника (**source description packets**). Хотя эти пакеты опциональны, их использование может раскрыть важные персональные данные, поэтому приложения не пересылают пакеты SDES без предварительного информирования пользователя о возможных последствиях.

Использование же пакетов SDES CNAME является исключением, так как они обязательно должны присутствовать. Включение в CNAME IP адреса может при этом иметь некоторые последствия, но эта информация уже присутствует в заголовке самого IP пакета.

Доступ к имени пользователя может иметь более существенные последствия, но в этом случае приложение может либо не включать имя пользователя в пакеты, либо заменить его на другое.

Некоторые участники сессии могут не желать, чтобы их присутствие было видимо остальным участникам сессии. Для этого случая необходима возможность пересылки пакетов RTCP избранному кругу участников сессии.

Random prefix			
V	P=0	RC=1	PT=201 (RR) Length=7
SSRC			
Reportee SSRC			
Loss fraction		Cumulative number of packets lost	
Extended highest sequence number received			
Interarrival jitter			
Timestamp of last sender report received (LSR)			
Delay since last sender report received (DLSR)			
V	P=1	SC=1	PT=202 (SDES) Length=9
SSRC			
Type=1 (CNAME)	Length=15	d	o
e	@	1	0
.	5	1	.
2	.	2	2
3	Type=2 (NAME)	Length=9	J
o	n	n	y
	D	o	e
Type=0	0	0	0
0 (padding)	0 (padding)	0 (padding)	4 (padding count)

Рис. 5.12. Пример зашифрованного пакета RTCP, показывающий правильное использование заполнителей

Для достижения конфиденциальности потока информации может быть использовано ее шифрование. Шифрованные составные пакеты содержат случайную 32-битную добавку, как показано на рис. 5.12.

Более детально вопросы безопасности и конфиденциальности будут рассмотрены далее в соответствующей главе.

5.5. Проверка корректности данных

В первую очередь следует распознавать RTP и RTCP пакеты. Правила составления пакетов, рассмотренные ранее, позволяют проводить их строгую проверку. Успешность этой проверки дает существенную гарантию корректности соответствующего RTP потока, хотя это не исключает и проверку самого потока.

В приводимом ниже примере показан фрагмент псевдокода, проверяющий корректность пакетов.

```
validate_rtcp(rtcp_t *packet, int length)

    rtcp_t    *end = (rtcp_t *) (((char *) packet) + length);
    rtcp_t    *r = packet;
    int       l     = 0;
    int       p     = 0;
    // All RTCP packets must be compound packets
    if ((packet->length+ 1) * 4) == length) {
        ... error: not a compound packet
    }
    // Check the RTCP version, packet type, and padding of the first
    // in the compound RTCP packet...
    if (packet->version != 2) {
        ...error: version number != 2 in the first subpacket
    }
    if (packet-> p != 0) {
        ...error: padding bit is set on first packet in compound
    }
    if ((packet->pt != RTCP_SR) && (packet->pt != RTCP_RR)) {
        ...error: compound packet does not start with SR or RR
    }
    // Check all following parts of the compound RTCP packet. The RTP
    // version number must be 2, and the padding bit must be zero on
    // all except the last packet.
    do {
        if (p == 1) {
            ...error: padding before last packet in compound
        }
        if (r-> p) {
            p = 1;
        }
        if (r-> version != 2) {
            ...error: version number != 2 in subpacket
        }
        l += (r->length + 1) * 4;
        r = (rtcp_t *) (((uint32_t *) r) + r->length + 1);
    } while (r < end);
```

```

// Check that the length of the packets matches the length of the
// UDP packet in which they were received...
if ((l != length) || (r != end)) {
...error: length does not match UDP packet length
}
...packet is valid
}

```

В этом примере необходимо отметить несколько ключевых моментов.

- Все пакеты должны быть составными пакетами RTCP.
- Поле версии всех пакетов имеет значение 2.
- Тип поля в первом RTCP пакете составного пакета должно быть со значением SR или RR.
- Если заполнение необходимо, то дополняется только последний пакет в составном пакете. Значение бита поля заполнения для остальных пакетов должно быть равно нулю.
- Сумма значений полей длины отдельных пакетов должна соответствовать длине составного пакета.

5.6. База данных участников сессии

Каждое приложение в RTP сессии создает базу данных о самой сессии и ее участниках. Информация о сессии, на основе которой происходит настройка характеристик сессии по времени, может храниться в виде набора переменных.

- **Пропускная способность (RTP bandwidth)**, рассчитываемая при старте сессии.
- **Фрагмент пропускной способности (RTCP bandwidth fraction)**, задаваемый как процент от общей пропускной способности для использования при пересылке отчетов RTCP. Обычно это значение равно 5 %, но профиль может его изменять, вплоть до значения 0, означающего отказ от использования пакетов RTCP.
- **Средний размер RTCP пакетов**, получаемых и передаваемых участником.
- **Количество участников сессии**, количество участников, которым был послан последний пакет RTCP, и части участников, переславших эти пакеты в предыдущем отчетном интервале.
- **Время последней отсылки пакетов RTCP** и время следующей передачи.
- **Флаг**, указывающий на пересылку RTCP пакетов всем участникам.

Кроме того, необходимо создавать переменные для формирования пакетов RTCP SR.

- **Количество переданных RTCP пакетов и байтов данных.**
- **Последний номер** пакета.

- **Соответствие** между временем RTP пакетов и меткой времени формата NTP.

Структура данных сессии, содержащая эти переменные, является хорошим местом для хранения используемых SSRC, SDES приложения и дескрипторов RTP и RTCP каналов.

Для правильного формирования RTCP пакетов каждому участнику сессии необходимо отслеживать состояние остальных участников. хорошим путем решения этой задачи является хранение такой информации в базе данных. Она может включать следующие данные:

- идентификатор SSRC;
- описание источника (CNAME и другая информация);
- информация, полученная из отчетов отправителя для обеспечения синхронизации изображения и звука;
- последнее время получения информации от неактивных участников;
- флаг, показывающий, пересылались ли участником данные в текущий отчетный интервал;
- буфер накопления мультимедийной информации перед проигрыванием;
- любая информация, необходимая при кодировании канала и обнаружения ошибок.

В RTP сессии участники идентифицируются своими идентификаторами SSRC. Так как участников может быть много и они могут обращаться к базе данных в любом порядке, то наиболее приемлемым видом базы данных участников является хэш-таблица, индексированная по идентификатору SSRC. Для приложений, работающих с единственным медиа форматом, этого вполне достаточно. Однако для синхронизации звука и изображения требуется доступ по параметру CNAME, поэтому хэш-таблица должна быть проиндексирована и по этому параметру.

Участники должны добавляться в базу данных при получении от них проверенного пакета. Этап проверки здесь весьма важен, так как приложение не должно заносить в базу информацию о непроверенных участниках. На эту тему можно сформулировать два совета:

- если пакет RTCP получен и проверен, соответствующий участник обязательно должен быть добавлен в базу. Проверка пакета RTCP достаточно строга, поэтому вероятность прохождения через нее неправильного пакета незначительна;
- заполнение базы должно происходить не только на основе RTP пакетов, за исключением случаев прибытия пакетов строго в порядке их номеров. Проверка пакета RTP недостаточно строга, поэтому вероятность прохождения через нее неправильного пакета высока.

Эти советы предполагают создание дополнительной упрощенной базы «подозрительных» участников сессии, в которую включаются участники, отправляющие только пакеты RTP. Для предотвращения использования некорректными пакетами слишком большого объема памяти в этой таблице должно быть жесткое ограничение по времени и максимальному размеру.

Каждый идентификатор CSRC, имеющийся в проверенном пакете, тоже должен был добавлен в базу.

Когда новый участник добавляется в базу, необходимо увеличивать количество участников на уровне сессии. Такое добавление может быть причиной повторного просмотра RTCP пакетов.

Участники удаляются из базы при получении пакета BYE или в результате превышения установленного периода неактивности. Это звучит просто, но скрывает некоторые проблемы.

Нет гарантии, что пакеты будут получены в той же последовательности, как они были отправлены, поэтому пакет BYE может быть получен раньше последнего пакета с данными от этого отправителя. Обычно приложения после получения пакета BYE делают задержку в 2 секунды для получения запоздавших пакетов с данными и лишь потом удаляют этого участника сессии.

Источники могут быть удалены, если они не присылают отчетов в течении пяти отчетных периодов. Если отчетные интервалы менее 5 секунд, то используется минимальное значение 5 секунд.

Процедура повторного просмотра RTCP пакетов будет рассмотрена более подробно в соответствующем разделе пособия.

5.7. Управление характеристиками времени

Частота, с которой участники сессии посылают RTCP пакеты, не фиксирована и изменяется в зависимости от количества участников и формата передаваемых в потоке данных. Целью является ограничить объем передачи пакетов RTCP выделенной для этих целей частью пропускной способности канала (обычно эта доля составляет 5 %). Эта цель достигается снижением частоты отправления пакетов RTCP при увеличении числа участников сессии. При телефонном разговоре двух участников с использованием протокола RTP отчеты RTCP посылаются с интервалом в несколько секунд, а при вещании для тысяч участников – с интервалом в несколько минут.

Каждый участник сам решает, когда посылать пакеты RTCP на основе набора правил, которые будут описаны далее. Очень важно строго выполнять эти правила, особенно в случае участия в больших сессиях. В противном случае может произойти перегрузка сети.

5.7.1. Отчетные интервалы

Составные пакеты RTCP пересылаются периодически, в соответствии со случайным таймером. Среднее время ожидания очередного пакета RTCP участником называется *отчетным интервалом*. На его значение влияют следующие факторы:

- **пропускная способность канала, выделенная под RTCP.** Обычное значение – 5 % от скорости передачи данных в сессии. Последняя устанавливается до сессии и служит параметром ее конфигурирования;
- **средний размер передаваемых и получаемых RTCP пакетов.** В этот размер входят не только передаваемые данные, но и UDP и IP заголовки;
- **общее количество участников и процент отправителей среди них.** Эти данные необходимы для создания базы данных участников.

Если количество отправителей больше нуля, но меньше четверти от общего количества участников сессии, отчетный интервал зависит от пересылки информации. Если пересылка идет, то отчетный интервал вычисляется умножением числа отправителей на средний размер пакета и делением на 25 % пропускной способности канала. При отсутствии передачи отчетный интервал вычисляется умножением количества получателей на средний размер RTCP пакета и умножением результата на 75 % пропускной способности канала:

```
If ((senders > 0) and (senders < (25 % of total number of participants))
{
  If (we are sending) {
    Interval = average RTCP size * senders / (25 % of RTCP bandwidth)
  } else {
    Interval = average RTCP size * receivers / (75 % of RTCP bandwidth)
  }
}
```

Если отправителей нет, или их количество больше четверти от общего числа участников сессии, то отчетный интервал вычисляется умножением числа общего числа участников на средний размер пакета, деленное на требуемую пропускную способность канала:

```
if ((senders = 0) or (senders > (25 % of total number of participants)) {
  Interval = average RTCP size * total number of members / RTCP
  bandwidth
}
```

Эти правила обеспечивают использование выделенной части канала. Пакеты RTCP требуются для синхронизации звука и изображения и идентификации отправителей, поэтому должны посылаться достаточно часто.

Полученный интервал всегда сравнивается с абсолютным минимальным значением, которое по умолчанию равно пяти секундам:

```
If (Interval < minimum interval) {  
    Interval = minimum interval  
}
```

В некоторых случаях требуется посылать RTCP пакеты чаще, чем через минимальный интервал, например, при очень высокой скорости передачи данных. Последние варианты спецификации RTP позволяют это делать:

```
Minimum interval = 360 / (session bandwidth in Kbps)
```

Отчетный интервал будет меньше 5 секунд при скорости передачи более 72 Kbps.

5.7.2. Базовые правила передачи

Когда приложение стартует, первый RTCP пакет отправляется на основе вычисленного отчетного интервала. Реальный интервал между пакетами определяется случайным образом от половинного значения отчетного интервала до полуторного, для предотвращения одновременного прибытия всех отчетов. Для первого пакета берется половина отчетного интервала, для обеспечения быстрой обратной связи с новым пользователем.

```
I = (Interval * random[0.5, 1.5])  
  
if (this is the first RTCP packet we are sending) {  
    I *= 0.5  
}  
next_rtcp_send_time = current_time + I
```

Подпрограмма `random[0.5, 1.5]` генерирует случайное число в интервале от 0.5 до 1.5. На некоторых платформах она называется `rand()` или `drand48()` при большей точности вычислений.

В качестве примера применения базовых правил передачи рассмотрим работу Интернет-радиостанции, передающей на скорости 128 Kbps звук в формате **mp3** с использованием RTP/IP вещания с числом слушателей 1000 человек. Используется минимальный отчетный интервал 5 секунд и 5 % пропускной способности канала отводится под RTCP, при среднем размере RTCP пакетов 90 байт, включая UDP/IP заголовки. Когда подключается новый слушатель, он не уведомляет других участников сессии. Число участников, которые являются отправителями, превышает четверть общего числа участников сессии. В этом случае отчетный интервал может быть вычислен следующим образом:

$$\begin{aligned}
\text{Interval} &= \text{average RTCP size} * \text{total number of members} / \text{RTCP bandwidth} \\
&= 90 \text{ octets} * 2 / (5 \% \text{ of } 128 \text{ Kbps}) \\
&= 180 \text{ octets} / 800 \text{ octets per second} \\
&= 0.225 \text{ seconds}
\end{aligned}$$

Так как значение 0.225 меньше минимального отчетного интервала, то используется интервал в 5 секунд. Это значение затем пропускается через датчик случайных чисел и делится пополам, так как речь сначала идет о первом пакете. Таким образом первый пакет будет послан в интервале от 1.25 до 3.75 секунд после старта приложения.

В течение времени от старта приложения до отправки первого пакета RTCP будет получено несколько отчетов от слушателей, что позволяет приложению дополнить список участников сессии. Это новое значение будет использовано при составлении второго пакета RTCP.

Как будет показано далее, 1000 слушателей достаточно для превышения средним интервалом минимального значения, поэтому частота получения пакетов RTCP от всех пользователей будет равна

$$75 \% \times 800 \text{ bytes per second} \div 90 \text{ bytes per packet} = 6.66 \text{ packets per second}$$

Если приложение посылает первый пакет, скажем, через 2.86 секунды, то число пользователей будет

$$2.86 \text{ seconds} \times 6.66 \text{ per second} = 19$$

Так как число отправителей в этом случае не превышает 25 % от общего числа участников сессии, то отчетный интервал для второго пакета вычисляется следующим образом:

$$\begin{aligned}
\text{Interval} &= \text{receivers} * \text{average RTCP size} / (75 \% \text{ of RTCP bandwidth}) \\
&= 19 * 90 / (75 \% \text{ of } (5 \% \text{ of } 128 \text{ Kbps})) \\
&= 1710 / (0.75 * (0.05 * 16000 \text{ octets/second})) \\
&= 1710 / 600 \\
&= 2.85 \text{ seconds}
\end{aligned}$$

Полученное значение вновь увеличивается до минимального интервала и рандомизируется. Вторым RTCP пакетом будет послан в интервале от 2.5 до 7.5 секунд после первого.

Процесс повторяется, причем будет подключено еще 33 слушателя в интервале от первого до второго пакета RTCP и их общее число возрастет до 52. В результате следующий отчетный интервал будет равен 7.8 секунд и будет использоваться напрямую, так как он превышает минимальное значение. Третий пакет будет послан в интервале от 3.9 до 11.7 секунд после второго. Средний интервал будет возрастать до тех

пор, пока не подключится вся аудитория радиостанции и он может быть вычислен следующим образом:

$$\begin{aligned} \text{Interval} &= \text{receivers} * \text{average RTCP size} / (75 \% \text{ of RTCP bandwidth}) \\ &= 1000 * 90 / (75 \% \text{ of } (5 \% \text{ of } 128 \text{ Kbps})) \\ &= 90000 / (0.75 * (0.05 * 16000 \text{ octets/second})) \\ &= 90000 / 600 \\ &= 150 \text{ seconds} \end{aligned}$$

Интервал в 150 секунд эквивалентен частоте $1/150 = 0.0066$ пакетов в секунду, что для 1000 слушателей составит 6.66 пакетов в секунду.

5.7.3. Процедура пересмотра вперед

Как уже упоминалось ранее, проходит несколько отчетных интервалов, прежде чем новый участник узнает общий размер сессии. На протяжении этого периода изучения новый участник посылает пакеты RTCP чаще, чем это установлено, так как ему не хватает информации. Этот факт становится существенным, когда много новых участников одновременно присоединяются к сессии. Такая ситуация типична для начала сессии.

В этом случае при использовании только базовых правил передачи данных каждый участник будет исходить из отсутствия на данный момент других участников сессии и соответствующим образом структурировать первый пакет RTCP. Он перешлет этот пакет в среднем через половину минимального отчетного интервала и будет строить следующий пакет на основе количества присоединившихся на данный момент участников, которых может быть несколько сотен или даже тысяч. Из-за низкого начального вычисленного количества участников сессии будет наблюдаться резкий рост нагрузки на сеть, что часто приводит к ее перегрузке.

Этот феномен изучался Розенбергом [5], рассматривавшего случай подсоединения 10 000 участников к сессии. Его эксперимент показал, что в первые 2.5 секунды все 10 000 участников пытаются послать первый RTCP отчет, в 3 000 раз превышая установленную частоту. Это естественно сразу приводит к перегрузке сети.

Эту проблему можно решить постоянным вычислением числа участников и доли отправителей в нем. При наступлении времени очередной передачи интервал пересчитывается на базе обновленного размера группы и это значение используется для установления нового времени передачи. Если оно еще не наступило, пакет не пересылается, а время передачи переносится на вновь рассчитанный момент.

Эта процедура звучит сложно, но ее реализация крайне проста. При передаче по базовым правилам фрагмент псевдокода выглядит следующим образом:


```

if (current_time >= next_rtcp_send_time) {
    send RTCP packet
    next_rtcp_send_time = rtcp_interval() + current_time
}

```

С процессом пересмотра он изменяется следующим образом:

```

if (current_time >= next_rtcp_check_time) {
    new_rtcp_send_time = (rtcp_interval() / 1.21828) + last_rtcp_send_time
    if (current_time >= new_rtcp_send_time) {
        send RTCP packet
        next_rtcp_check_time = (rtcp_interval() / 1.21828) + current_time
    } else {
        next_rtcp_check_time = new_send_time
    }
}
}

```

Здесь функция `rtcp_interval()` возвращает рандомизированное значение отчетного интервала, базирующееся на последнем вычислении размера сессии. Отметим деление этого интервала на число 1.21828 (Эйлера константа e минус 1.5). Этот фактор компенсирует эффект процедуры пересмотра вперед, обеспечивая суммарную нагрузку на сеть в рамках заданных пяти процентов от общей нагрузки.

Эффектом процедуры пересмотра вперед является задержка пере-сылки RTCP пакетов в момент увеличения числа пользователей. Это показано на рис. 5.13, где видно существенное сокращение нагрузки на сеть при использовании процедуры пересмотра вперед.

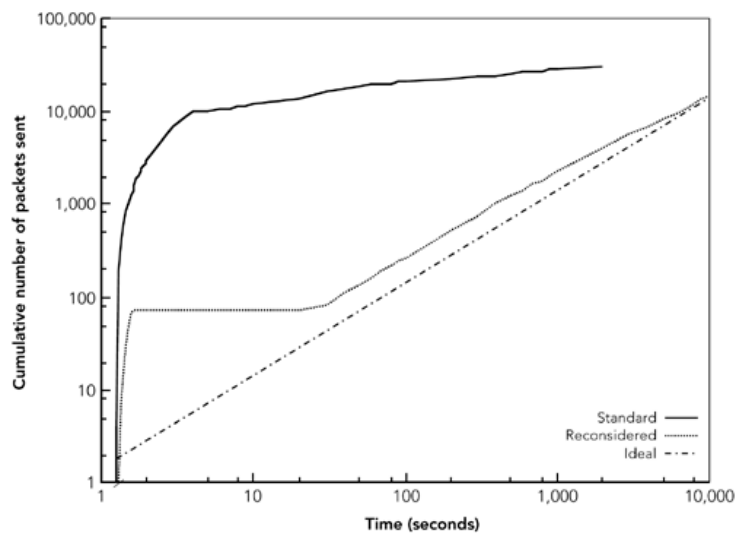


Рис. 5.13. Эффект процедуры пересмотра вперед

В качестве второго примера рассмотрим сценарий, изложенный в предыдущем разделе, когда новый пользователь подключается к радиостанции, ведущей Интернет-вещание. Когда слушатель подклю-

ется к сессии, первый RTCP пакет, как было рассмотрено ранее, должен быть отослан между 1.25 и 3.75 секундами после старта приложения. Предположим, что значение первого интервала получилось 2.86 секунды. Перед отсылкой обрабатывает процедура пересмотра вперед, обнаруживающая 19 новых участников трансляции и вычисляется новый интервал 2.85 секунды:

```
Interval = number of receivers * average RTCP size / (75 % of RTCP
bandwidth)
          = 19 * 90 / (0.75 * (0.05 * 16000 octets/second))
          = 1710 / 600
          = 2.85 seconds
```

Результат меньше минимально допустимого значения отчетного интервала, поэтому берется значение 5 секунд, которое рандомизируется и делится на масштабный фактор. Если полученное значение меньше установленного ранее (2.85 секунды), то пакет посылается, если же нет – устанавливается новое время отсылки отчета.

Процедура просмотра вперед проста для применения и рекомендуется для всех приложений, работающих с сессиями передачи данных.

5.7.4. Процедура пересмотра назад

Как и при одновременном подключении пользователей, проблемы возникают и при одновременном их отключении (этап завершения сессии). Проблема возникает не с перегрузкой сети, а со слишком большими задержками в отчетах оставшихся участников сессии. Эти задержки для оставшихся участников сессии могут восприниматься как длительное молчание и пользователи в этом случае отмечаются как неактивные.

Проблема решается путем, аналогичным способу, примененному в предыдущем разделе. При получении каждого пакета ВУЕ пересчитывается число участников сессии и время отсылки следующего пакета. Разница с процедурой пересмотра вперед заключается в направлении изменения времени отсылки пакетов, которое в этом случае уменьшается, а не увеличивается.

Процедура вычисления выглядит следующим образом:

```
if (BYE packet received) {
    member_fraction = num_members_after_BYE / num_members_before_BYE
    time_remaining = next_rtcp_send_time - current_time
    next_rtcp_send_time = current_time + member_fraction * time_remaining
}
```

Процедура пересмотра назад применяется при числе пользователей в несколько сотен, когда есть вероятность быстрого уменьшения числа участников сессии.

5.7.5. Пересмотр пакетов BYE

Участники сессии, решившие ее покинуть, посылают пакет BYE и затем выходят из сессии. Если это происходит для многих участников одновременно, может произойти перегрузка сети.

Для предотвращения этой нежелательной ситуации текущая версия RTP разрешает посылать пакет BYE немедленно только в том случае, если число участников сессии менее 50. При числе участников более 50 пакет BYE отсылается с задержкой, если в это время получен такой же пакет от другого участника. Этот процесс называется пересмотром пакетов BYE (**BYE reconsideration**).

Этот процесс аналогичен процессу просмотра вперед, но базируется на числе полученных пакетов BYE, а не на числе участников сессии. Перед отправкой пакета BYE учитывается число принятых пакетов этого типа от других участников и пересчитывается время отправления этого пакета. Эта процедура повторяется до тех пор, пока пакет не будет отослан.

Данная операция полезна только в случае многочисленных сессий, в которых возможен одновременный выход большого числа участников.

5.7.6. Общие проблемы реализации

Наиболее общими проблемами реализации являются:

- Некорректное масштабирование при увеличении числа участников сессии. Наблюдается при использовании фиксированного отчетного интервала.
- Отсутствие рандомизации отчетного интервала. Приводит к перегрузкам в сети.
- Не включение заголовков протоколов нижнего уровня при различных вычислениях размеров пакетов.
- Некорректное использование заполнителей. Если заполнение необходимо, оно проводится только для последнего пакета RTCP в составном пакете.

При тестировании поведения приложений, работающих с RTP сессиями, необходимо использовать не один сценарий, а их спектр, меняя в них характеристики сессий (число участников, доля отправителей и т. д.).

6. ЗАХВАТ МУЛЬТИМЕДИА, ПРОИГРЫВАНИЕ И УПРАВЛЕНИЕ ХАРАКТЕРИСТИКАМИ ВРЕМЕНИ

В этой главе мы отойдем от обсуждения протоколов и рассмотрим создание систем, работающих по протоколу RTP. Реализация RTP имеет много аспектов, часть из которых касается всех систем, а другая – используется опционально при необходимости.

6.1. Поведение отправителя

Как отмечалось ранее, отправитель отвечает за захват мультимедиа («живого» потока или файла), сжатие данных для их передачи, генерацию RTP пакетов. Он может также участвовать в коррекции ошибок и контроле перегрузок сети путем адаптации передаваемого потока в ответ на отчеты его получателей.

Работа отправителя начинается с чтения несжатых данных мультимедиа (аудио потока или видео фрагментов) в буфер, из которого потом создаются сжатые фреймы. Фреймы могут быть сжаты несколькими способами, в зависимости от выбранного алгоритма сжатия, а сами фреймы могут зависеть от предыдущих или последующих данных.

Сжатым фреймам назначается метка времени и номер в последовательности, после чего они загружаются в RTP пакеты и готовы к передаче. Если фрейм не помещается в один пакет, он может быть фрагментирован на несколько пакетов. Если фреймы достаточно малы, в один пакет могут быть помещено несколько фреймов. В зависимости от используемой схемы коррекции ошибок кодировщик канала может использоваться для генерации пакетов коррекции ошибок или перестановки пакетов. Затем отправитель периодически отправляет отчеты в виде RTCP пакетов для генерируемых потоков и получает отчеты о качестве полученной информации от других участников сессии, имея возможность на их основе адаптировать параметры потока. Все эти моменты будут рассмотрены далее.

6.2. Захват мультимедиа и сжатие

Процесс захвата мультимедиа практически одинаков как для звука, так и для изображения. Несжатый фрейм захватывается, при необходимости преобразуется в подходящий для сжатия формат, а затем с помощью кодировщика создается сжатый фрейм. Полученный фрейм пропускается через программу пакетизации, в результате чего получается один или несколько пакетов RTP.

6.2.1. Захват и сжатие звука

Рис. 6.1 иллюстрирует процесс захвата звука рабочей станцией с его оцифровкой и сохранением во входном звуковом буфере.

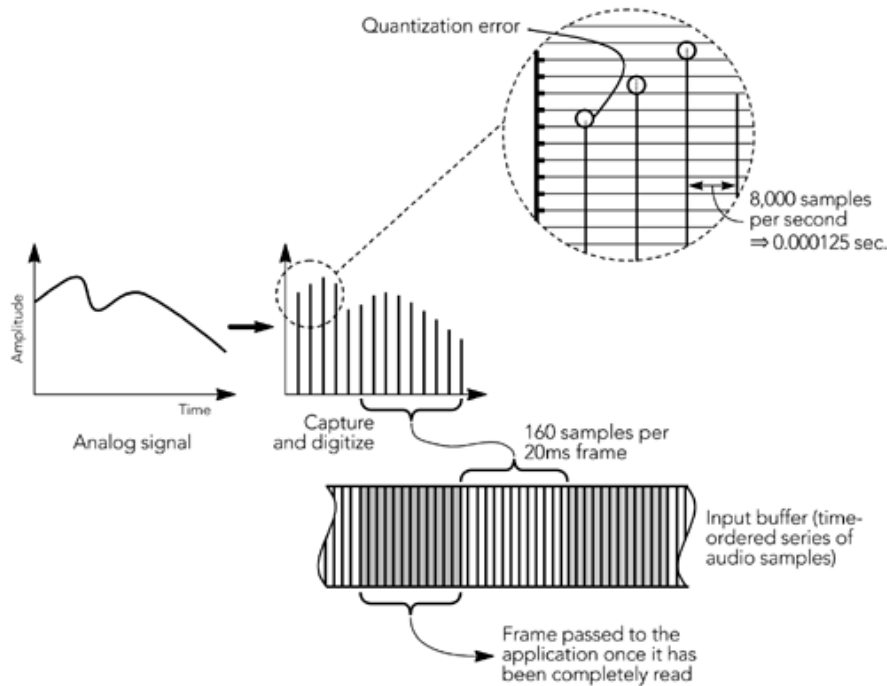


Рис. 6.1. Захват звука

Входной звуковой буфер обычно делают доступным для приложения, когда в нем накопится фиксированное количество сэмплов (сэмпл – амплитудная выборка, т. е. выходной уровень звука в данный момент). Большинство приложений для захвата звука забирают данные из входного звукового буфера фреймами фиксированной продолжительности, каждый раз ожидая, пока в буфере накопится полный фрейм. Это приводит к появлению задержки в получении очередной порции звука. Для уменьшения влияния этой задержки размер фрейма устанавливается в соответствии с размером буфера используемого кодека (обычно 20 или 30 миллисекунд).

Несжатый фрейм, полученный от устройства захвата звука, может содержать не один, а несколько типов сэмплов. Обычно эти устройства возвращают 8-, 16- или 24-битные сэмплы со скоростью от 8000 до 96000 сэмплов в секунду (моно или стерео).

Многие звуковые кодеки обнаруживают паузы в передаче звука, выбрасывая фреймы без звука или содержащие только фоновый шум. Это дает более эффективное использование канала передачи информации.

6.2.2. Захват и сжатие изображения

Устройства захвата изображения чаще работают с полными фреймами, чем с полосами сканирования или областями изображения. Многие предлагают работу с пониженным разрешением и разным размером фреймов, их форматом и глубиной цвета.

В зависимости от используемого кодека, бывает необходимо преобразовать фрейм из формата устройства захвата изображения в другой формат. Алгоритмы такого преобразования находятся за пределами вопросов, рассматриваемых в данном учебном пособии. Наиболее часто встречаемое преобразование состоит в переходе между цветовой схемой RGB и YUV. Такое преобразование ускоряет обработку фреймов на основе использования специальных команд процессора. На рис. 6.1 показан процесс захвата изображения (NTSC сигнал со схемой YUV) и преобразование его в формат RGB.

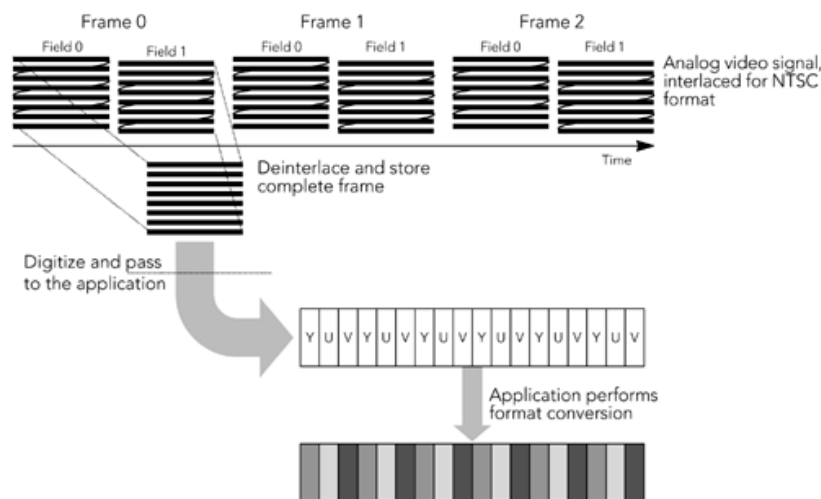


Рис. 6.2. Захват изображения

Захваченные фреймы перед их сжатием кодировщиком помещаются в буфер. Размер буфера зависит от используемой схемы сжатия.

Устройства захвата и звука, и изображения могут сразу осуществлять их сжатие. Эта функция реализована в специализированных устройствах, хотя встречается и в продвинутых рабочих станциях. Такие фреймы можно напрямую передавать в поток RTP, хотя могут быть наложены некоторые ограничения на его характеристики.

Вне зависимости от типа мультимедиа и использованного способа сжатия результатом данного этапа является последовательность сжатых фреймов с указанным временем захвата. Эти фреймы поступают в модуль RTP для пакетизации и передачи.

6.2.3. Использование предварительно записанной информации

Подготовленные фреймы обычно поступают на этап пакетизации в том же порядке, в каком они были захвачены. Протокол RTP не делает различий между предварительно записанной информацией и «живым» потоком.

В частности, в начале предварительно записанной информации отправитель должен сгенерировать новый идентификатор SSRC и выбрать случайные значения для метки времени и порядкового номера. В процессе передачи он должен управлять SSRC коллизиями, получать и отправлять пакеты RTCP.

Предположим, что отправитель не может просто брать поток из файла и помещать его содержимое тоже в файлы. Вместо этого он использует процесс запоминания и пакетизации «на лету» (рис. 6.3).

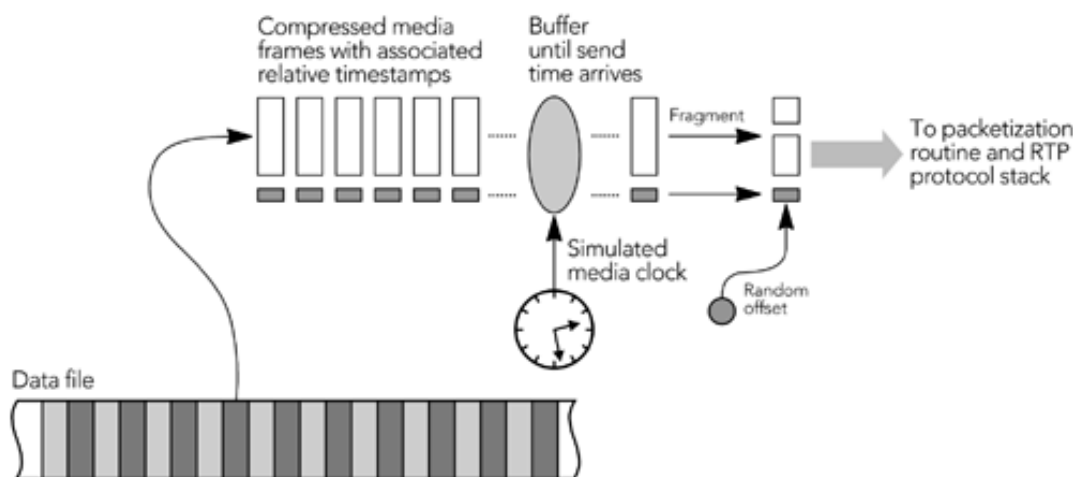


Рис. 6.3. Использование предварительно записанной информации

6.3. Генерация пакетов RTP

Сгенерированные сжатые фреймы передаются процедуре пакетизации. Каждому фрейму соответствует метка времени, из которой получается метка времени RTP. Если формат данных поддерживает фрагментацию, большие фреймы разделяются на максимально допустимые для передачи фрагменты (обычно это необходимо только для видео). Наконец, один или несколько RTP пакетов, содержащих заголовки и собственно данные, генерируются для каждого фрейма. Формат пакета и его заголовка определяется используемым кодеком. Критической частью процесса генерации пакетов является назначение фреймам меток времени, фрагментирование больших фреймов и генерация заголовков данных. Эти моменты мы и обсудим более подробно.

6.3.1. Метки времени и модель времени RTP

Метка времени представляет момент получения первого байта данных. Значение начинается со случайно сгенерированного числа и возрастает со скоростью, определяемой передаваемыми данными.

В процессе захвата «живого» потока мультимедиа метка времени обозначает момент получения первых сэмплов с устройства захвата. Если речь идет о синхронизированных звуковых и видео потоках, то надо быть уверенным, что время обработки потоков разного типа учтено при вычислении метки времени. Для большинства звуковых форматов метка времени RTP для каждого очередного пакета возрастает пропорционально числу входящих в него сэмплов, а не байтов данных его длины. Исключением является форматы звука MPEG, включая формат MP3, который использует частоту 90 КГц в данных для совместимости с другими форматами MPEG. Большинство форматов видео также используют часы 90 КГц, что дает им возможность увеличивать значение метки времени целыми значениями шага. Например, при использовании стандартной для NTSC скорости 29.97 фреймов в секунду и использовании в данных частоты 90 КГц метка времени RTP будет увеличиваться на 3.003 на каждый пакет.

Для потока, предварительно записанного в файле, метка времени дает время фрейма в последовательности фреймов плюс постоянное случайное смещение. Метки времени назначаются для фреймов, поэтому, если большой фрейм фрагментируется, каждый фрагмент будет иметь одинаковую метку времени.

Спецификация RTP не гарантирует детальность, точность и стабильность часов данных. Каждое приложение само решает эту задачу.

Метка времени в RTP данных и отсылаемых отправителем RTCP пакетах отражает параметры времени у отправителя. Отметим, что модель времени RTP ничего не говорит о времени проигрывания передаваемых данных. Метки времени данных дают относительные параметры времени и RTCP отчеты отправителя помогают синхронизировать разные потоки. Но RTP ничего не говорит о количестве буферов, используемых для проигрывания данных или о времени их раскодирования.

Хотя модель времени детально описана в спецификации RTP, в ней нет упоминания об алгоритмах, используемых при реконструкции параметров времени на стороне получателей пакетов. Это сделано преднамеренно, так как такие алгоритмы зависят от требований самого приложения.

6.3.2. Фрагментация

Процесс фрагментации большого фрейма проиллюстрирован на рис. 6.4. Каждый фрагмент имеет метку времени всего фрейма и дополнительный заголовок данных, описывающий этот фрагмент.

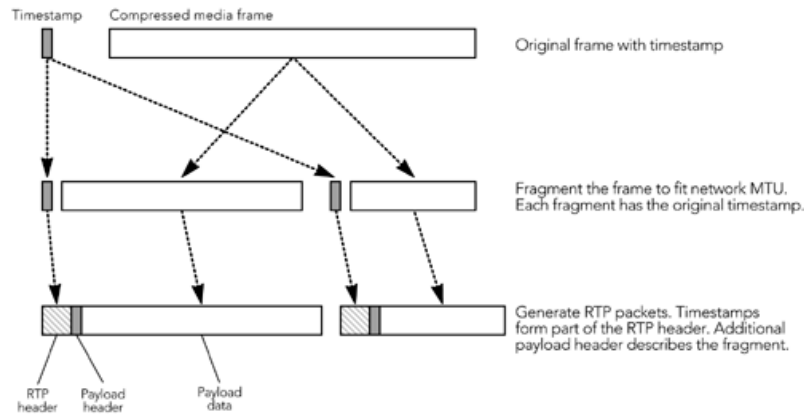


Рис. 6.4. Фрагментация большого фрейма

Процесс фрагментации критичен к потере пакетов. Желательна возможность декодирования отдельных фрагментов, в противном случае потеря одного фрагмента приводит к потере всего фрейма.

После фрагментации отправитель должен решить, будет ли он посылать все фрагменты одновременно, или последовательно через определенный интервал. Первый вариант может привести к перегрузке сети, поэтому рекомендуется выбирать второй.

6.3.3. Заголовки, зависящие от формата данных

В дополнение к заголовку RTP и самим передаваемым данным мультимедиа пакеты часто содержат заголовки, зависящие от формата данных. Этот заголовок определяет спецификацию используемого формата данных и обеспечивает переход от RTP к выходу кодека.

Типичное использование дополнительного заголовка происходит при адаптации форматов данных, не предназначенных для передачи в сетях с потерями. Правильно разработанные заголовки данных существенно расширяют сферу применения таких форматов. Более детально эта тема будет рассмотрена позднее.

6.4. Поведение получателя

Как уже отмечалось ранее, получатель отвечает за прием пакетов из сети, восстановление и корректировку потерянных и поврежденных пакетов, восстановление характеристик времени, декодирование данных и представление результата пользователю. Кроме того, от получателя ожидают присылку пакетов RTCP для обеспечения адаптации процесса передачи данных к характеристикам используемых сетей. Получатель также обычно создает базу данных об участниках сессии.

Первым шагом в процессе получения данных сессии является собственно сбор пакетов из сети, проверка их корректности и выстраивание

в последовательности, заданной отправителем. Это достаточно простые операции, не зависящие от формата данных. Далее этот процесс будет описан подробнее.

Остальные операции выполняются в стиле отправителя и зачастую зависят от формата принимаемых данных. Пакеты из входной очереди передаются программе кодирования и проверяются на потери. Кодировщик помещает пакеты в очереди по типу данных мультимедиа, где они ожидают получения полного фрейма и достижения времени проигрывания. Процедура вычисления размера задержек будет рассмотрена далее в соответствующем разделе.

Иногда до достижения времени проигрывания пакеты группируются в полные фреймы с восстановлением потерянных или поврежденных фреймов, после чего они декодируются. В завершение полученные данные подготавливаются для пользователя. В зависимости от формата данных и используемого проигрывающего устройства каждый поток может проигрываться отдельно (например, два потока видео в разных окнах). В других случаях бывает необходимо смешать данные из разных потоков в один. Каждая из этих операций также будет рассмотрена далее.

Операции, выполняемые получателем, сложнее операций, выполняемых отправителем. Многие из этих осложнений связаны с необходимостью восстановления потерянной при передаче информации.

6.5. Получение пакетов

RTP сессия включает в себя как потоки данных, так и потоки управления, работающие с назначенными портами (обычно четным для потока данных и нечетным для потока управления). Это означает, что принимающее потоки приложение должно открывать два *сокета* (*socket* – объект, являющийся конечным элементом соединения, обеспечивающего взаимодействие между процессами транспортного уровня сети), один – для данных, второй – для управления. Так как RTP работает в стеке протоколов UDP/IP, он использует стандартные сокеты SOCK_DGRAM.

После создания сокетов приложение подготавливается к получению пакетов по сети и хранению их для последующей обработки. Многие приложения реализуют это в виде цикла, повторяющегося при получении очередных пакетов.

```
fd_data = create_socket(...);
fd_ctrl = create_socket(...);
while (not_done) {
    FD_ZERO(&rfd);
    FD_SET(fd_data, &rfd);
    FD_SET(fd_ctrl, &rfd);
    timeout = ...;
```

```

if (select(max_fd, &rfd, NULL, NULL, timeout) > 0) {
    if (FD_ISSET(fd_data, &rfd)) {
        ...validate data packet
        ...process data packet
    }
    if (FD_ISSET(fd_ctrl, &rfd)) {
        ...validate control packet
        ...process control packet
    }
}
...do other processing
}

```

Пакеты данных и управления проверяются на корректность, как описывалось в предыдущих разделах. Интервал повторения цикла получения пакетов выбирается по интервалу между фреймами.

Для организации процесса получения пакетов может использоваться и событийный подход: пакеты в этом случае обрабатываются по мере их поступления. Обычно такой подход применяется при условии работы в реальном времени.

6.5.1. Получение пакетов данных

Первой стадией проигрывания данных мультимедиа по сети является получение пакетов и их буферизация для дальнейшей обработки. Так как сеть часто изменяет интервал времени между пакетами, как показано на рис. 6.5, то несколько пакетов могут появиться в короткий интервал времени или могут возникнуть большие паузы между поступлением пакетов. Получатель не знает заранее, когда придут очередные пакеты, поэтому он должен быть готов к получению нескольких пакетов сразу, причем в любом порядке.

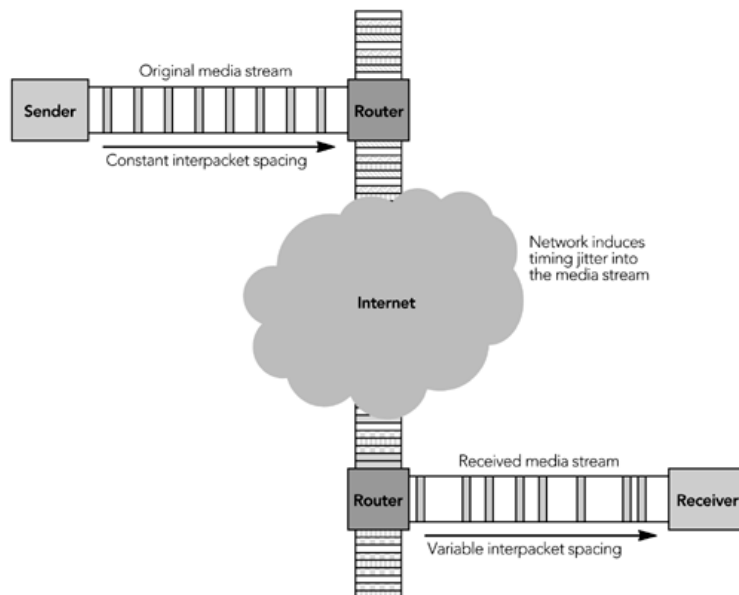


Рис. 6.5. Нарушение интервала времени между пакетами при их транспортировке по сети

После получения пакетов они проверяются на корректность и добавляются в очередь для дальнейшей обработки в порядке возрастания их меток времени. Эти шаги разрывают связь между временем прибытия пакетов и временем их обработки и проигрывания для пользователя. На рис. 6.6 показано разделение между получением пакетов и программами проигрывания, которые имеют доступ только к входным очередям.

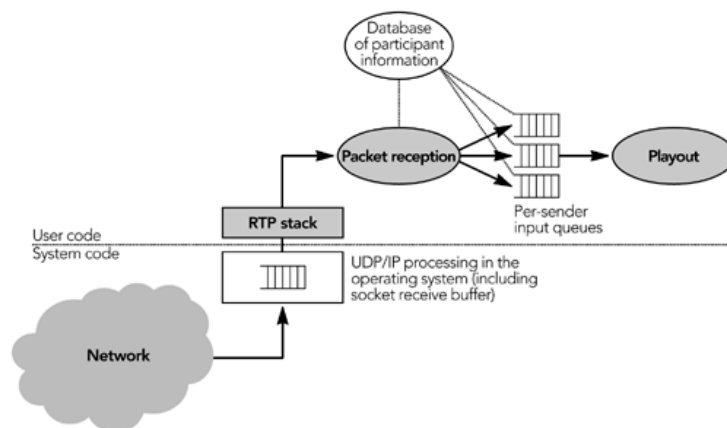


Рис. 6.6. Отделение процесса получения пакетов от их проигрывания

Важно запоминать точное время прибытия пакетов M , чтобы иметь возможность рассчитать соответствующие статистические характеристики. Неточное измерение может привести к появлению задержек в передаче пакетов. Время прибытия измеряется по реальному времени получателя T и преобразуется в потоковое время R :

$$M = T \times R + \text{смещение}$$

Здесь *смещение* используется для перехода от относительного времени к потоковому времени, корректируя разницу между ними.

Как отмечалось ранее, процесс приемки пакетов в приложении может чередоваться с их обработкой, или эти действия могут быть разделены по разным ветвям логики работы приложения. В первом случае одна ветвь осуществляет и прием пакетов, и их обработку перед проигрыванием. За один цикл работы все пришедшие пакеты считываются из сокета и вставляются в правильном порядке в очередь проигрывания. Во втором случае одна ветвь приложения занимается приемкой пакетов и формированием из них очереди. Другие ветви занимаются декодированием данных и подготовкой их к проигрыванию.

Независимо от выбранного варианта приложение не может принимать и обрабатывать пакеты непрерывно. Входные очереди сглаживают флуктуации для процесса проигрывания в приложении, но что делать с задержками в поступлении пакетов? К счастью, большинство операци-

онных систем общего назначения управляют приемкой пакетов UDP/IP на основе управления прерываниями и могут буферизовать пакеты на уровне сокета, даже если в это время принимающее приложение занято. Принятый по умолчанию буфер сокета подходит для большинства приложений, но приложения, работающие с высокоскоростными сетями, могут увеличить размер буфера до необходимого. Правда, это может привести к увеличению задержек в получении пакетов, но в приложении можно соответственно увеличить значение задержки проигрывания, что компенсирует данный отрицательный эффект.

6.5.2. Получение управляющих пакетов

Параллельно процессу приемки пакетов данных приложение должно быть готово к получению, проверке, обработке и отсылке пакетов RTCP. Информация пакетов RTCP используется при создании базы данных об участниках сессии, как было описано в предыдущей главе, а также для проверки и идентификации участников, адаптации параметров сети и синхронизации звука и изображения.

Одноканальные приложения обычно обрабатывают и пакеты данных, и пакеты RTCP в одном цикле. Многоканальные приложения выделяют одну ветвь для приема и обработки управляющих пакетов. Так как частота появления пакетов RTCP намного меньше, чем для пакетов данных, загрузка их обработчика невелика и не критична по времени. Однако следует запоминать точное время прибытия пакетов SR с отчетами отправителя, так как это время используется при генерации пакета RR получателя и участвует в вычислении времени передачи по сети.

После получения пакета RTCP с отчетом отправителя или получателя содержащаяся в нем информация сохраняется. Синтаксический разбор блоков отчетов в пакетах SR и RR достаточно несложен. Поле `count` в заголовке пакета RTCP показывает количество блоков отчета, причем значение 0 допустимо и показывает, что отправитель пакета RTCP не получал пакетов данных RTP.

Главное назначение пакетов RTCP является отслеживание качества приемки отправленных пакетов с данными. Если отчеты показывают плохое качество приема, то можно либо добавить коды предупреждения ошибок, либо уменьшить скорость отправки пакетов. В сессии с несколькими отправителями можно отслеживать качество других отправителей. Приложения обычно сохраняют информацию из отчетов для последующего ее использования при подстройке процесса передачи данных.

Отчеты отправителей содержат также информацию о соответствии часов данных RTP и относительных часов отправителя, используемую для синхронизации звука и изображения.

Когда приходит пакет SR, его информация сохраняется и может быть показана пользователю. Каноническое имя CNAME в пакете SDES обеспечивает связь между потоками звука и изображения. Другое направление его использования – группировка потоков, идущих от одного отправителя (например, при передаче изображения с нескольких камер).

После проверки качества пакетов RTCP их информация добавляется в базу данных об участниках сессии. Так как проверка качества пакетов RTCP достаточно строга, то наличие участника в базе указывает, что он действительно существует. Это полезный параметр проверки пакетов RTP.

Когда приходит пакет RTCP BYE, соответствующий вход в базу данных помечается для последующего удаления. Как отмечалось в предыдущей главе, удаление не выполняется немедленно, давая возможность получить задержанные пакеты данного отправителя. Еще одна операция, выполняемая на основе этой информации – удаление неактивных пользователей.

6.6. Буфер проигрывания

Пакеты данных извлекаются из входной очереди и вставляются в определенный источник буфер проигрывания в соответствии с их метками времени. Фреймы задерживаются в очереди проигрывания, чтобы сгладить неравномерность поступления данных по сети. Эта задержка позволяет также получить и собрать вместе фрагменты больших фреймов. В завершении процесса фреймы подвергаются декомпрессии, а полученная мультимедийная информация подготавливается для пользователя. Рис. 6.7 иллюстрирует этот процесс.

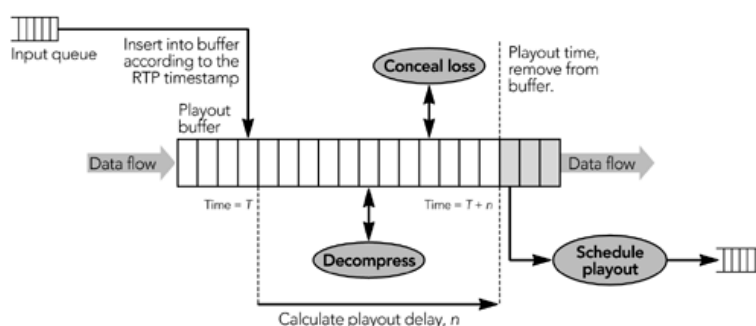


Рис. 6.7. Буфер проигрывания

Один и тот же буфер может быть использован и для компенсации изменений в передающей сети, и для работы соответствующего кодека. Но возможно и разделение этих функций, однако это не является жестким требованием для RTP.

6.6.1. Базовые операции

Буфер проигрывания содержит упорядоченный по времени связанный список вершин. Каждая вершина представляет фрейм данных мультимедиа с соответствующей информацией об их параметрах по времени. Структура данных в вершине содержит указатель на смежную вершину, время получения, метку времени RTP, желаемое время проигрывания и указатели как на фрагменты фрейма, так и на готовые для проигрывания данные. Рис. 6.8 изображает эту структуру.

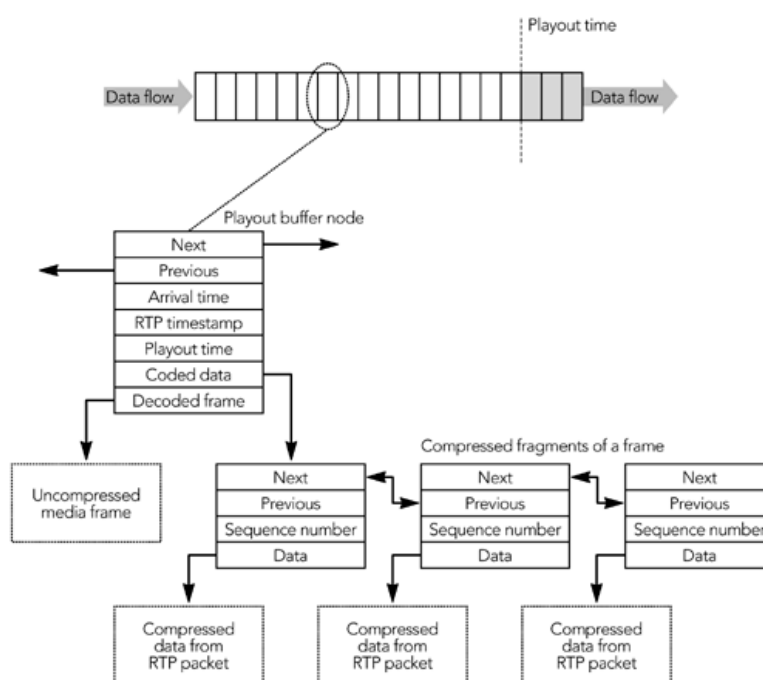


Рис. 6.8. Структура данных буфера проигрывания

При получении первого пакета RTP из фрейма он удаляется из входной очереди и помещается в буфер проигрывания в соответствии со значением его метки времени. Это приводит к созданию новой вершины в структуре данных буфера проигрывания, которая вставляется в связанный список. Сжатые данные из последнего полученного пакета связываются с этой вершиной для дальнейшего декодирования. Затем вычисляется время проигрывания фрагмента по схеме, которая будет описана позже.

Вновь созданная вершина находится в буфере проигрывания до момента наступления времени проигрывания данного фрейма. В течение периода ожидания могут быть получены и связаны с данной вершиной остальные фрагменты фрейма. Как только будет обнаружено поступление последнего фрагмента фрейма, проводится его декодирование

и результат связывается с вершиной. Определение завершенности фрейма зависит от используемых кодеков. Звуковые кодеки обычно не фрагментируют фреймы и пересылают в пакете фрейм целиком (исключением является формат MP3). Видео кодеки практически всегда фрагментируют фреймы, отмечая маркером последний фрагмент фрейма в структуре пакета RTP.

Решение о времени применения декодера принимается самим получателем пакетов и не определяется форматом пакетов RTP. Пакеты могут декодироваться сразу после получения или оставаться сжатыми до последнего момента. Выбор зависит от загрузки приложения обработкой и наличия достаточного места для хранения декодированной информации.

При достижении момента проигрывания фрейма и наличии фрейма в очереди проигрывания выполняются действия, которые будут описаны далее в соответствующей главе. Если фрейм до этого момента еще не декодирован, то необходимо немедленно выполнить данную операцию, даже при отсутствии некоторых его фрагментов. Также в это время может быть выполнено исправление обнаруженных ошибок.

При проигрывании фрейма соответствующая вершина в структуре буфера проигрывания должна быть удалена или очищена и использована заново. При использовании процедуры исправления ошибок может потребоваться ввод некоторой задержки в удалении вершины, пока фрейм не будет полностью проигран.

Пакеты RTP, прибывшие после удаления соответствующей им вершины в структуре буфера проигрывания, должны быть отброшены. Своевременность пришедшего пакета определяется путем сравнения его метки времени с меткой времени самого «старого» пакета в буфере проигрывания. Понятно, что следует выбирать такую задержку проигрывания фреймов, чтобы случаи прихода опоздавших пакетов были достаточно редки, поэтому приложение должно подстраивать значение этой задержки на основе анализа процесса получения пакетов.

При операциях с буфером проигрывания приходится выбирать между точностью проигрывания и задержками. Системы, предназначенные для интерактивного использования, такие как Интернет-конференции и IP-телефония, должны пытаться уменьшить значение задержки до минимально возможного. Исследования показывают, что значение времени отклика в 300 миллисекунд является максимально допустимым в таких системах, что ограничивает максимальную задержку половиной этого времени. Однако в не интерактивных системах, таких как потоковое вещание, задержка может достигать значения в несколько секунд, обеспечивая высокое качество воспроизведения потока.

6.7. Вычисление времени проигрывания

Одна из главных проблем при создании буфера проигрывания – определение значения задержки проигрывания. На выбор влияют следующие факторы:

- задержка между получением первого и последнего фрагмента фрейма;
- задержка до получения любого пакета коррекции ошибок;
- изменения интервала поступления пакетов из-за перегрузок в сети и изменения маршрутов;
- относительное смещение по времени между отправителем и получателем;
- ограничение по задержкам в приложении и важность качества приема и величины задержек.

Факторы, на которые приложение может влиять, включают интервал между пакетами во фрейме и задержку между данными и пакетами корректировки ошибок. Воздействие данных факторов будет рассмотрено позже в соответствующем разделе.

За пределами зоны контроля приложения оказывается поведение сети, аккуратность и стабильность часов у отправителя и получателя. На рис. 6.9 показано взаимоотношение между временем передачи пакетов и временем получения пакетов звука по RTP. Если часы отправителя и получателя работают с одинаковой скоростью, то наклон линии на графике должен быть около 45 градусов. На практике часы отправителя и получателя не синхронизированы и имеют разные скорости. На рисунке видно, что часы отправителя идут быстрее, чем часы получателя, так как наклон кривой меньше 45 градусов. В одном из следующих разделов мы рассмотрим процедуру синхронизации часов.

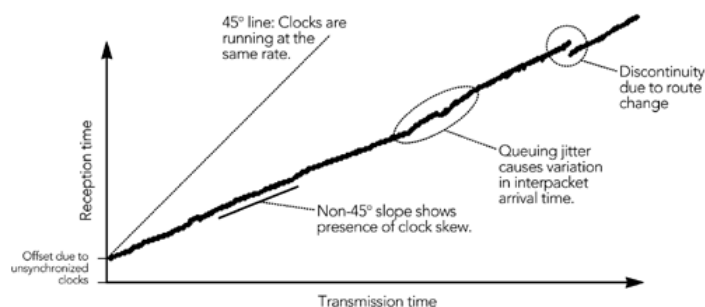


Рис. 6.9. Смещение времени отправки и получения пакетов

Если пакеты имеют постоянную скорость передачи по сети, линия на рисунке будет прямой. Однако обычно сеть обладает некоторой не-

равномерностью интервала между пакетами из-за неравномерности задержек в очередях и это видно на рисунке как отклонение от прямой. На рисунке также виден разрыв, являющийся следствием изменения скорости передачи по сети (чаще всего – из-за изменений в маршрутизации пакетов). Учет и компенсация этих процессов будут рассмотрены позже.

Что касается вычисления полной задержки, то она зависит от ограничений приложения. В основном принимается во внимание человеческий фактор – сколько времени пользователь может ожидать данные.

Получатель должен учитывать все эти факторы при вычислении времени проигрывания. Эти вычисления выполняются в следующей последовательности:

- время отправителя сравнивается со временем локального проигрывания для компенсации сдвига между часами отправителя и получателя, давая базовое время для дальнейших вычислений;
- при необходимости получатель компенсирует сдвиг по времени путем добавления величины сдвига к базовому времени;
- задержка проигрывания для локального времени вычисляется с использованием зависящей от отправителя задержки и неустойчивости передачи в сети;
- задержка проигрывания пересчитывается, если происходит изменение маршрута передачи пакетов по сети, изменение порядка пакетов или при других изменениях в данных;
- наконец, задержка проигрывания добавляется к базовому времени, формируя время проигрывания.

На рис. 6.10 изображена последовательность этих шагов.

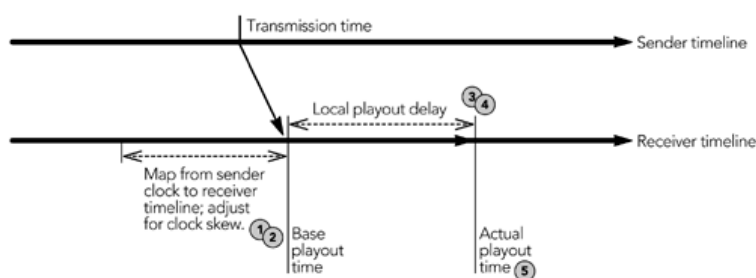


Рис. 6.10. Вычисление времени проигрывания

6.7.1. Преобразование к локальному времени

Первой стадией вычисления времени проигрывания является установление соответствия схемы времени отправителя (по метке времени RTP) со схемой времени получателя путем добавления относительного смещения их часов к метке времени.

Для вычисления относительного смещения получатель вычисляет разницу $d(n)$ между меткой времени $T_{R(n)}$ RTP пакета с порядковым номером n и временем прибытия $T_{L(n)}$ этого пакета, измеренными в одинаковых единицах:

$$d(n) = T_{L(n)} - T_{R(n)}$$

Эта разница включает постоянный фактор, потому что часы отправителя и получателя инициализированы разными случайными начальными значениями с переменными задержками на подготовку данных на стороне отправителя. Второй постоянный фактор связан с минимальным временем передачи пакетов по сети. Кроме того, в нее входят переменная задержка из-за неустойчивости времени передачи по сети и разница по времени из-за использования разных часов отправителем и получателем. Значение разницы записывается в 32-битное беззнаковое целое, как и метка времени, на основе которой она была получена, и, так как часы отправителя и получателя не синхронизированы, разница может иметь любое 32-битное значение.

Разница вычисляется для каждого проходящего пакета и получатель отслеживает ее минимальное значение для получения относительного смещения:

$$offset = \min(d(n-w) \dots d(n))$$

Из-за существования разницы скоростей между $T_{L(n)}$ и $T_{R(n)}$ разница $d(n)$ будет иметь тенденцию дрейфа значения в меньшую или большую сторону. Для предотвращения этого дрейфа минимальное смещение вычисляется через окно w , начинающегося с момента последней компенсации изменения часов. Так как значение хранится как беззнаковое, то необходимы следующие операции:

$$a < b \text{ if } (a-b) \ \& \ 0x80000000 \neq 0$$

Значение относительного смещения (*offset*) используется для вычисления базовой точки проигрывания в соответствии со схемой времени получателя:

$$base_playout_time(n) = T_{R(n)} + offset$$

Это начальное значение времени проигрывания, которое потом корректируется.

6.7.2. Компенсация изменений времени

Формат данных RTP определяет номинальную скорость часов для передаваемого потока, но не предъявляет требований к стабильности и

точности часов. Часы отправителя и получателя в общем идут с различными скоростями, заставляя последнего компенсировать эту разницу. График времени передачи пакетов, приведенный ранее на рис. 6.9, иллюстрирует этот факт. Если наклон графика точно 45 градусов, то часы отправителя и получателя идут с одинаковой скоростью, а отклонение от этого угла указывает на различие в скоростях.

Получатель должен обнаружить изменение характеристик времени, оценить их величину и на этой основе скорректировать точку проигрывания. Возможны два варианта корректировки: подстройка часов получателя или периодическая перенастройка буфера проигрывания.

В последнем варианте, если скорость часов отправителя больше скорости часов получателя, из буфера проигрывания исключаются некоторые данные. В противном случае синтезируются дополнительные данные и помещаются в буфер. В зависимости от величины изменений по времени определяется частота пересмотра буфера проигрывания.

При первом же варианте компенсации (если часы получателя могут быть точно отрегулированы) подстраивается их скорость по времени отправителя. Этот подход обеспечивает лучшее качество, так как никакие данные не отбрасываются и не добавляются, но он может требовать некоторую техническую поддержку.

Вычисление величины рассогласования времен может вначале показаться простой задачей: посмотреть скорость часов отправителя по метке времени в данных и сравнить с показаниями локальных часов. Если $T_{R(n)}$ – метка времени полученного пакета с номером n , а $T_{L(n)}$ – показание локальных часов в этот момент, то рассогласование времени может быть вычислено следующим образом:

$$\text{skew} = \frac{T_{R(n)} - T_{R(n-1)}}{T_{L(n)} - T_{L(n-1)}}$$

Значение *skew* меньше единицы означает, что часы отправителя медленнее часов получателя и наоборот. Однако наличие нестабильности времени передачи по сети сводит полезность такого простого расчета к нулю. Получателю приходится рассматривать большой период времени получения пакетов, чтобы исключить влияние данного фактора нестабильности.

Существует множество алгоритмов управления рассогласованием времени в зависимости от требований к точности и чувствительности к нестабильности работы сети. Далее обсуждается один из простых подходов, применимый к передаче голоса по сети Интернет (*voice-over-IP*).

При этом подходе постоянно отслеживается среднее значение задержки в сети и сравнивается с текущей вычисленной задержкой. Уве-

личение разницы между ними свидетельствует о наличии рассогласования времен. При прибытии каждого пакета получатель вычисляет мгновенную задержку d_n пакета с номером n на базе времени приема и метки времени пакета:

$$d(n) = T_{L(n)} - T_{R(n)}$$

При получении первого пакета получатель устанавливает значение текущей задержки $E = d_0$ и средней задержки $D_0 = d_0$. С каждым очередным пакетом средняя задержка пересчитывается по формуле экспоненциально взвешенного перемещающегося среднего:

$$D_n = \frac{1}{32} D_{n-1} + \frac{31}{32} d_n = (D_{n-1} + d_n) \frac{31}{32}$$

Фактор $\frac{1}{32}$ контролирует процесс усреднения, делая среднее нечувствительным к кратковременным флуктуациям значения времени передачи. Отметим, что это вычисление аналогично вычислению показателя неустойчивости работы сети, но возвращает только знак изменения и использует долговременную константу, полученную при длительном по времени изменении, что снижает влияние кратковременной неустойчивости работы сети.

Средняя задержка D_n сравнивается с текущей задержкой E для получения величины их дивергенции:

$$divergence = E - D_n$$

Если часы отправителя и получателя синхронизированы, значение дивергенции будет близко к нулю с небольшими вариациями из-за неравномерности работы сети. Существенно отличное от нуля значение показывает необходимость запускать процедуру компенсации. Порог зависит от неустойчивости работы сети, используемого кодека и набора доступных вариантов адаптации.

Компенсация увеличивает или уменьшает буфер проигрывания, как будет рассмотрено далее в этой главе. Точка проигрывания смещается на величину дивергенции, измеренную в единицах метки времени. После компенсации рассогласования по времени получатель пересчитывает текущую задержку E , приравнивая ее средней задержке D_n . Это приводит к нулевому значению дивергенции на данный момент времени.

На Си-подобном псевдокоде алгоритм выглядит следующим образом:

```
adjustment_due_to_skew(rtp_packet p, uint32_t curr_time) {
    static int      first_time = 1;
    static uint32_t delay_estimate;
    static uint32_t active_delay;
    uint32_t        adjustment = 0;
    uint32_t        d_n = p->ts - curr_time;
```

```

if (first_time) {
    first_time = 0;
    delay_estimate = d_n;
    active_delay = d_n;
} else {
    delay_estimate = (31 * delay_estimate + d_n)/32;
}
if (active_delay - delay_estimate > SKEW_THRESHOLD) {
    // Sender is slow compared to receiver
    adjustment = SKEW_THRESHOLD;
    active_delay = delay_estimate;
}
if (active_delay - delay_estimate < -SKEW_THRESHOLD) {
    // Sender is fast compared to receiver
    adjustment = -SKEW_THRESHOLD;
    active_delay = delay_estimate;
}
// Adjustment will be 0, SKEW_THRESHOLD, or -SKEW_THRESHOLD. It is
// appropriate that SKEW_THRESHOLD equals the framing interval.
return adjustment;
}

```

В этом алгоритме предполагается, что неустойчивость работы сети симметрична и любое систематическое смещение обуславливается расогласованием часов. Если это не так, то алгоритм даст неверную адаптацию. В этом случае можно использовать более сложный алгоритм Муна [6], основанный на линейном программировании.

6.7.3. Компенсация поведения отправителя

Природа процесса генерации пакетов отправителем может влиять на вычисление параметров проигрывания несколькими путями, приводя к увеличению задержки в буфере проигрывания.

Если отправитель отправляет фреймы через установленный интервал, как обычно бывает для видео, то это приведет к появлению задержки между первым и последним пакетом фрейма. В этом случае получатель должен буферизовать пакеты до получения полного фрейма. На рис. 6.11 показан этот вариант.

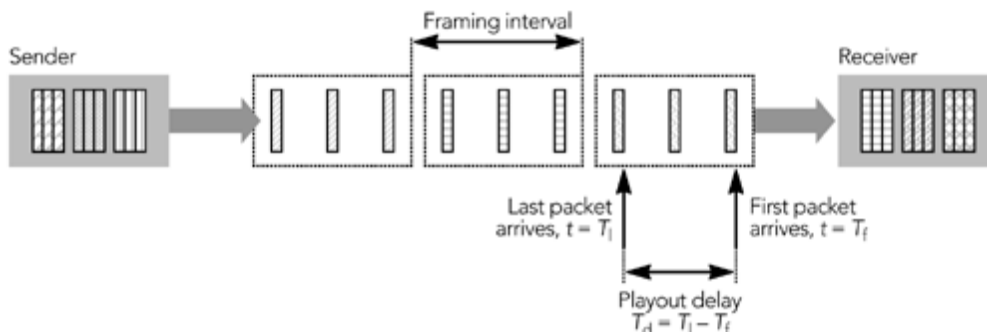


Рис. 6.11. Задержка из-за группировки пакетов во фрейм

Если интервал между фреймами и количество пакетов во фрейме известны, дополнительная задержка вычисляется достаточно просто (однако в реальных ситуациях это происходит достаточно редко):

$$\text{adjustment_due_to_fragmentation} = (\text{packets_per_frame} - 1) \times (\text{interframe_time} / \text{packets_per_frame})$$

Аналогичная ситуация возникает, если отправитель использует процедуру корректировки ошибок, которая будет описана в главе 9. Из-за наличия дополнительных пакетов корректировки ошибок значение времени проигрывания увеличивается. Присутствие пакетов корректировки ошибок отмечается в процессе организации сессии и дает достаточно информации для правильного расчета размера буфера проигрывания. Компенсационная задержка зависит от типа схемы корректировки ошибок. Тремя наиболее распространенными схемами являются FEC (Forward Error Correction), звуковая избыточность (audio redundancy) и повторная пересылка (retransmission).

Схема FEC, обсуждаемая в главе 9, оставляет пакеты данных без изменений, посылая пакеты корректировки ошибок в отдельном потоке RTP. Эти пакеты содержат битовую маску в заголовке FEC для идентификации номеров пакетов данных, которые они защищают. По этой маске получатель может определить размер задержки. Если интервал между пакетами постоянен, задержка добавляется ко времени проигрывания. В противном случае получатель должен суммировать интервалы между пакетами для получения смещения точки проигрывания.

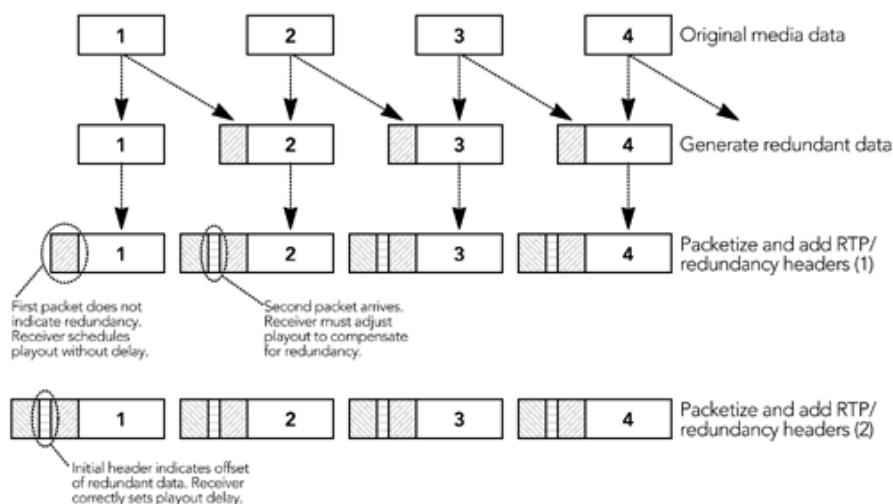


Рис. 6.12. Эффект избыточного звукового кодирования в буфере проигрывания

Схема звуковой избыточности также обсуждается в главе 9. Она вводит смещение по времени в избыточных пакетах, которое может

быть использовано при вычислении размера буфера проигрывания. Избыточный звук может быть представлен в двух режимах: начальные пакеты могут посылаться без избыточного блока, или с избыточным блоком нулевой длины. Как показано на рис. 6.12, проще рассчитать размер буфера проигрывания во втором случае.

6.7.4. Компенсация неустойчивости работы сети

Когда пакеты RTP проходят по реальным сетям, колебания межпакетного интервала неизбежны. При существенной неустойчивости передачи в сети получатель должен компенсировать ее добавлением задержки в буфер проигрывания. На рис. 6.13 показан процесс компенсации неустойчивости работы сети.

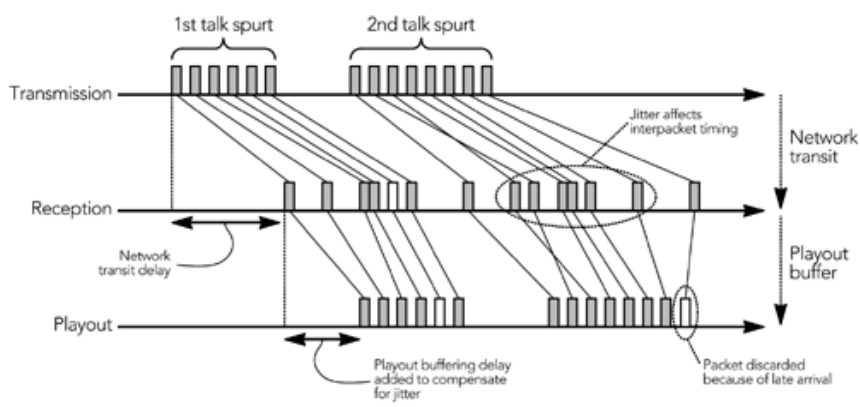


Рис. 6.13. Компенсация неустойчивости работы сети

Стандартных алгоритмов вычисления размера такой задержки не существует. Приложения используют в основном различные адаптивные алгоритмы в зависимости от типа приложения и состояния сети.

Во многих случаях неустойчивость работы сети носит случайный характер. На рис. 6.14 показан график изменения интервала между пакетами в зависимости от частоты, который по виду близок к распределению Гаусса.

Такая аппроксимация подходит для большинства случаев, хотя реальная практика передачи данных по сетям сдвигает распределение в сторону больших интервалов. Это допущение позволяет достаточно просто рассчитать задержку проигрывания. Вычисляется стандартное отклонение нестабильности сети, а из теории вероятности мы знаем, что 99 % нормального распределения лежит в пределах интервала в три стандартных отклонения от среднего значения.

Как измерить стандартное отклонение? Значение нестабильности сети вычисляется для помещения в отчет получателя для отслеживания среднего изменения во времени доставки пакетов, которое в свою оче-

редь может быть использовано для аппроксимации значения стандартного отклонения. На основе этой аппроксимации задержка проигрывания, требуемая для компенсации нестабильности сети, может быть принята равной утроенной величине нестабильности сети, вычисленной для отдельного источника потока.

$$T_{\text{playout}} = T_{\text{current}} + 3J$$

Здесь J является текущим вычисленным значением нестабильности сети, как было описано в главе 5.

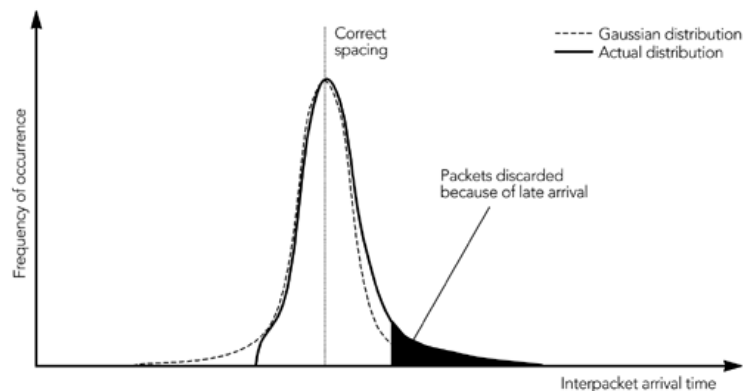


Рис. 6.14. График изменения интервала между пакетами

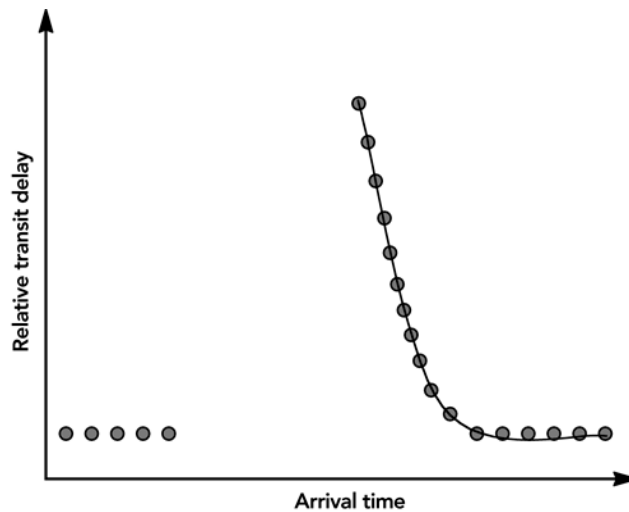


Рис. 6.15. Увеличение интервала времени между пакетами

Насколько часто следует пересчитывать в этих случаях размер буфера проигрывания? На решение этого вопроса влияют несколько факторов.

- Значимое увеличение доли пакетов, отброшенных из-за позднего прибытия.
- Получение нескольких пакетов подряд, которые следует отбросить по той же причине (достаточно трех таких пакетов).

- Получение пакетов от источника, который неактивен длительное время (обычно в течение 10 секунд).
- Резкое увеличение времени передачи по сети.

Определить момент резкого увеличения времени передачи по сети достаточно просто. Это просто момент неожиданного увеличения времени доставки между очередными пакетами. Такая ситуация показана на рис. 6.15.

Соответствующий псевдокод выглядит следующим образом:

```
int
adjustment_due_to_jitter(...)
{
    delta_transit = abs(transit - last_transit);
    if (delta_transit > SPIKE_THRESHOLD) {
        // A new "delay spike" has started
        playout_mode = SPIKE;
        spike_var = 0;
        adapt = FALSE;
    } else {
        if (playout_mode == SPIKE) {
            // We're within a delay spike; maintain slope estimate
            spike_var = spike_var / 2;
            delta_var = (abs(transit - last_transit) + abs(transit
                last_last_transit))/8;

            spike_var = spike_var + delta_var;
            if (spike_var < spike_end) {
                // Slope is flat; return to normal operation
                playout_mode = NORMAL;
            }
            adapt = FALSE;
        } else {
            // Normal operation; significant events can cause us to
            //adapt the playout
            if (consecutive_dropped > DROP_THRESHOLD) {
                // Dropped too many consecutive packets
                adapt = TRUE;
            }
            if ((current_time - last_header_time) >
                INACTIVE_THRESHOLD) {
                // Silent source restarted; network conditions have
                //probably changed
                adapt = TRUE;
            }
        }
    }
    desired_playout_offset = 3 * jitter
    if (adapt) {
        playout_offset = desired_playout_offset;
    } else {
        playout_offset = last_playout_offset;
    }
    return playout_offset;
}
```

6.7.5. Компенсация изменений в маршрутизации

Хотя ситуация изменения маршрутизации передаваемых пакетов вследствие нарушения связи между отдельными пунктами сети или других топологических изменений встречается не так часто, она вполне возможна. Если происходит изменение маршрутизации потока RTP, это проявляется в неожиданном изменении времени передачи по сети. Такое изменение будет разрушать буфер проигрывания, так как либо пакеты будут прибывать за пределами времени проигрывания, либо слишком рано и перекрываться с уже пришедшими пакетами.

Алгоритмы компенсации нестабильности сети обнаруживают изменения в задержке доставки пакетов и могут адаптировать буфер проигрывания, но этот подход не оптимален. Более быстрая адаптация возможна, если получатель напрямую наблюдает за временем передачи пакетов по сети и подстраивает размер буфера проигрывания при больших его изменениях. Например, приложение может обнаружить, что изменение времени передачи по сети в пять раз превышает вычисленное значение нестабильности сети.

6.7.6. Компенсация изменений порядка пакетов

В экстремальных случаях нестабильность работы сети и изменение в маршрутизации пакетов может нарушить последовательность их передачи. Как уже отмечалось в главе 2, это достаточно редкая ситуация, но приложение должно быть готово к ней, умея воспроизводить данные при нарушении порядка пакетов.

При правильной организации получения изменение порядка пакетов не является проблемой, так как в буфер проигрывания они вставляются в соответствии с меткой времени, независимо от порядка, в котором они поступали. Если задержка проигрывания достаточно велика, такие пакеты проигрываются в правильном порядке. В противном случае они отбрасываются, как и любые опоздавшие пакеты. Если таких пакетов окажется слишком много, стандартный алгоритм компенсации нестабильности работы сети пересчитает размер буфера проигрывания.

6.8. Адаптация точки проигрывания

Существует два основных подхода к процессу адаптации точки проигрывания: либо немного изменить время проигрывания каждого фрейма и передвинуть тем самым точку проигрывания, либо удалить или добавить лишнее или недостающее количество фреймов.

6.8.1. Адаптация точки проигрывания для звука с подавлением тишины

Звук является непрерывным форматом мультимедиа, то есть каждый фрейм проигрывается определенное время и следующий фрейм проигрывается немедленно после предыдущего. Если не используется подавление тишины, то между фреймами нет времени на их адаптацию, поэтому соответствующие алгоритмы адаптации существенно образом учитывают наличие такого подавления.

При передаче речи говорящий генерирует периоды звуковых колебаний продолжительностью несколько сот миллисекунд, разделяемые периодами тишины. На рис. 6.16 показана структура такого сигнала.

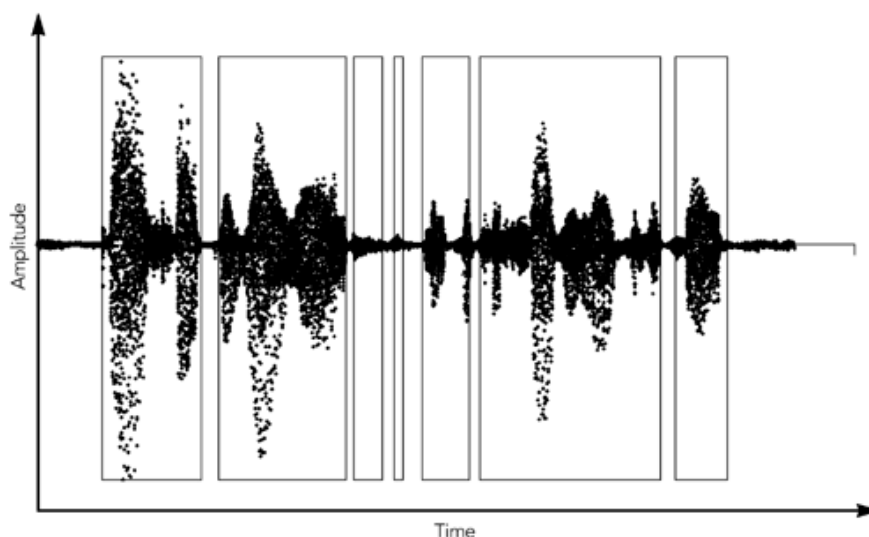


Рис. 6.16. Структура речевого сигнала

Отправитель обнаруживает фреймы, содержащие периоды тишины и подавляет их. В результате последовательность пакетов RTP содержит метки времени, учитывающие эти периоды тишины, то есть появляются некоторые разрывы времени между воспроизведением фреймов. В таких случаях получатель должен стараться адаптировать точку проигрывания только в период тишины.

Получателю достаточно несложно обнаружить начало очередного периода звучания, так как отправитель обязан установить в единицу соответствующий бит для первого пакета после периода тишины. Иногда, правда, этот первый пакет теряется. Однако возможность обнаружить этот период остается, так как в этом случае нарушается заданная последовательность номеров и значений меток времени, как показано на рис. 6.17.

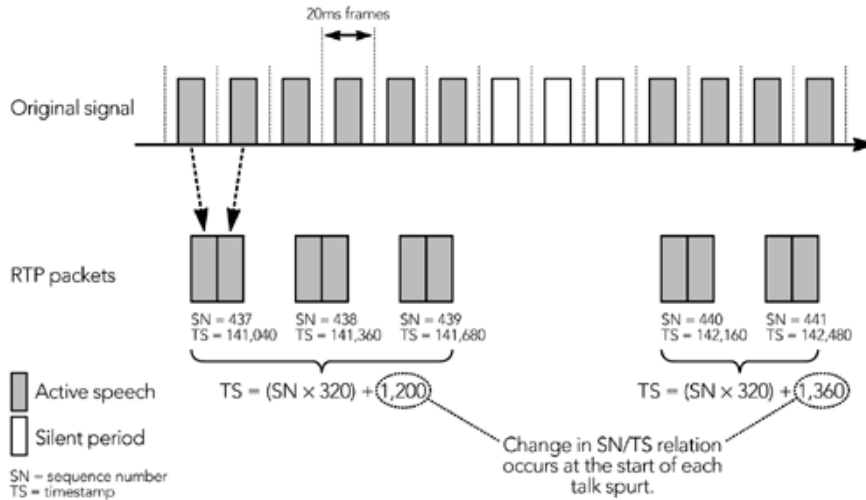


Рис. 6.17. Обнаружение начала периода звучания

Когда начало периода звучания обнаружено, можно адаптировать точку проигрывания небольшим изменением интервала тишины. Ниже приведен псевдокод, реализующий данные действия.

```

int
should_adjust_playout(rtp_packet curr, rtp_packet prev, int contdrop) {
    if (curr->marker) {
        return TRUE; // Explicit indication of new talk spurt
    }
    delta_seq = curr->seq - prev->seq;
    delta_ts = curr->ts - prev->ts;
    if (delta_seq * inter_packet_gap != delta_ts) {
        return TRUE; // Implicit indication of new talk spurt
    }
    if (curr->pt == COMFORT_NOISE_PT) || is_comfort_noise(curr) {
        return TRUE; // Between talk spurts
    }
    if (contdrop > CONSECUTIVE_DROP_THRESHOLD) {
        contdrop = 0;
        return TRUE; // Something has gone badly wrong, so adjust
    }
    return FALSE;
}

```

6.8.2. Адаптация точки проигрывания для звука без подавления тишины

Если не используются алгоритмы подавления тишины, получатель должен успевать адаптировать точку проигрывания следующего фрейма во время проигрывания предыдущего. Существует несколько алгоритмов такой адаптации, зависящих от природы устройства вывода и ресурсов получателя.

- Звук может быть переработан программными средствами для приведения его в соответствие с характеристиками устройства вывода.

- Выравнивание по сэмплам на основе знания воспроизводимого содержимого.
- Добавление или удаление целых фреймов, как если бы пакеты были потеряны или повторялись.

6.8.3. Адаптация точки проигрывания для видео

Видео является дискретным форматом мультимедиа в виде набора отдельных последовательных фреймов. Дискретная природа видео обеспечивает гибкость используемых алгоритмов адаптации, позволяя варьировать интервалы времени между фреймами. Правда, устройства воспроизведения обычно работают с фиксированной скоростью и ограничивают возможность изменения этих интервалов.

Например, рассмотрим воспроизведение клипа со скоростью 50 фреймов в секунду на мониторе с частотой обновления 85 герц. В этом случае частота дисплея не совпадает с частотой проигрывания фреймов, что порождает флуктуации времени показа очередного фрейма (рис. 6.18).

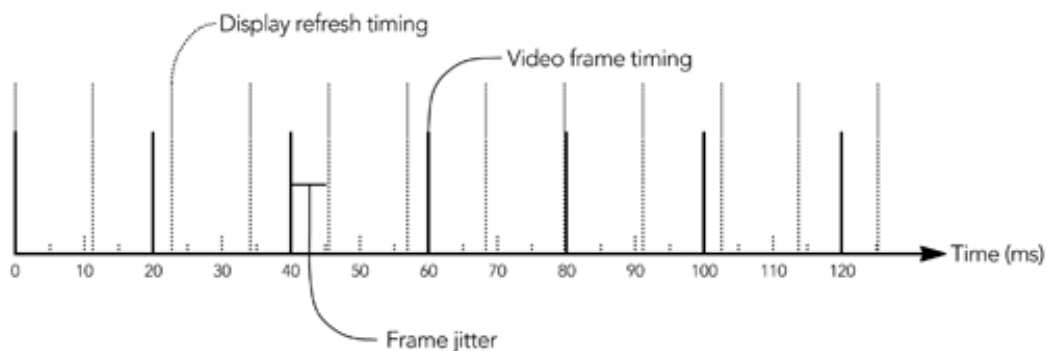


Рис. 6.18. Несовпадение частот воспроизведения в данных и устройстве вывода

Возможны три варианта адаптации, в зависимости от взаимного соотношения частот в данных и устройстве проигрывания.

Если частота дисплея выше, чем скорость захвата фреймов, то возможные точки воспроизведения будут окружать желаемое время воспроизведения и каждый фрейм можно сопоставить соответствующему интервалу обновления дисплея. Простейшим вариантом будет выбор ближайшего к точке воспроизведения интервала обновления. Промежуточные интервалы заполняются копиями предыдущего фрейма.

Если частота дисплея ниже, чем скорость захвата фреймов, то показ всех фреймов невозможен и получатель должен отбрасывать часть данных.

Если устройство захвата изображения и дисплей работают с одинаковой частотой, то размер буфера проигрывания должен учитывать и нестабильность работы сети. Данный случай на практике встречается редко, так как обычно существует рассогласование часов отправителя и получателя.

6.9. Декодирование, смешивание и проигрывание

Завершающей стадией процесса проигрывания является декодирование сжатых данных, смешивание отдельных потоков и собственно проигрывание данных для пользователя.

6.9.1. Декодирование

Для каждого активного источника приложение должно использовать декодер. Декодер может быть реальным физическим устройством или программным средством. Он преобразует сжатый фрейм в несжатые данные, как показано на рис. 6.19.

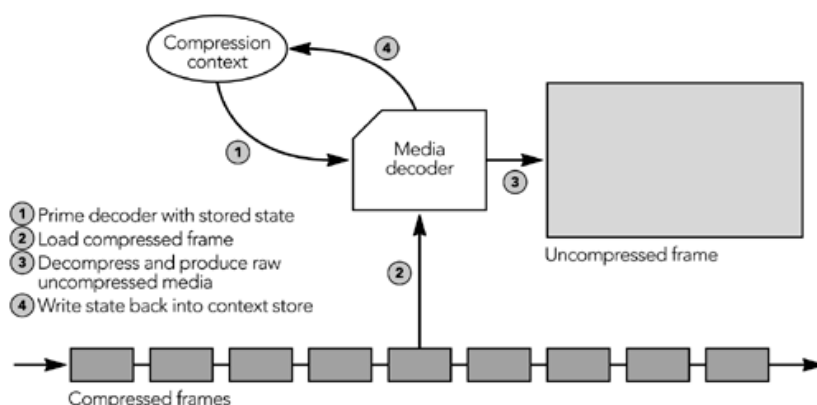


Рис. 6.19. Декомпрессия фреймов декодером

Если часть фреймов потеряна, декодер может выдать некорректный результат. В этом случае в процессе проигрывания может возникнуть пауза.

6.9.2. Смешивание звука

Смешивание является процессом объединения нескольких потоков в один. Этот случай более всего характерен для звуковых приложений, которые получают звук из нескольких источников, а проигрывают его на одном комплекте воспроизведения, например, при проведении телеконференции. Процесс смешивания показан на рис. 6.20.

Соответствующая программа на псевдокоде выглядит следующим образом:

```

audio_mix(sample *mix_buffer, sample *src, int len)
{
    int i, tmp;
    for(i = 0; i < len; i++) {
        tmp = mix_buffer[i] + src[i];
        if (tmp > 32767) {
            tmp = 32767;
        } else if (tmp < -32768) {
            tmp = -32768;
        }
        mix_buffer[i] = tmp;
    }
}

```

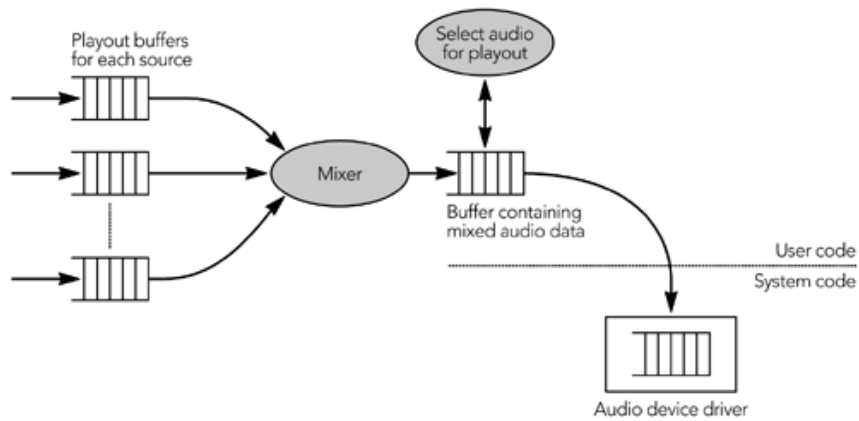


Рис. 6.20. Смешивание звука

Реальный буфер смешивания реализуется как циркулярный буфер, как показано на рис. 6.21.

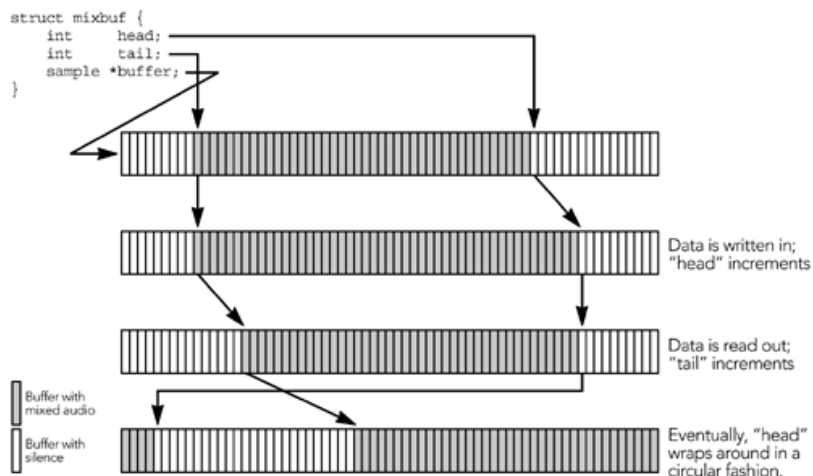


Рис. 6.21. Циркулярный буфер

На рис. 6.22 показан циркулярный буфер с дополнительным буфером копирования.

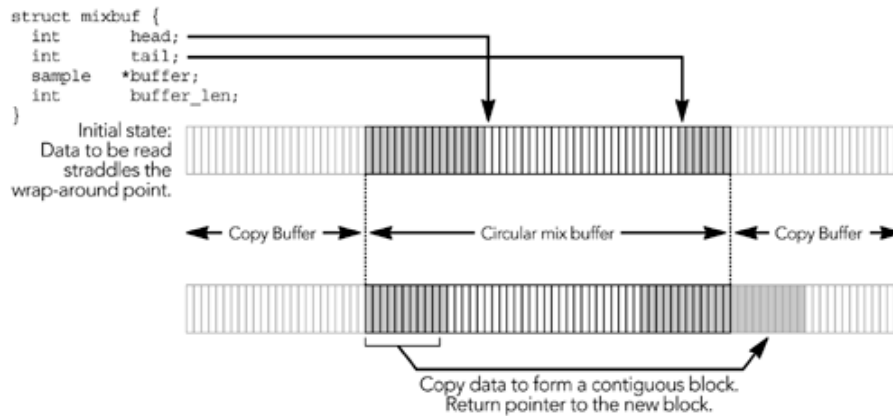


Рис. 6.22. Циркулярный буфер с дополнительным буфером копирования

6.9.3. Проигрывание звука

Процесс проигрывания звука для пользователя обычно является асинхронным, позволяя обрабатывать следующий фрейм во время воспроизведения текущего фрейма. Такая технология защищает приложение от последствий изменений в системе передачи данных по сети.

Асинхронное воспроизведение особенно важно в операционных системах общего назначения с ограниченной поддержкой мультимедийных приложений. В таких системах нельзя гарантировать качество работы приложений реального времени. В таких случаях приложения обычно для асинхронного проигрывания используют устройства с DMA (Directory Memory Access), как показано на рис. 6.23.

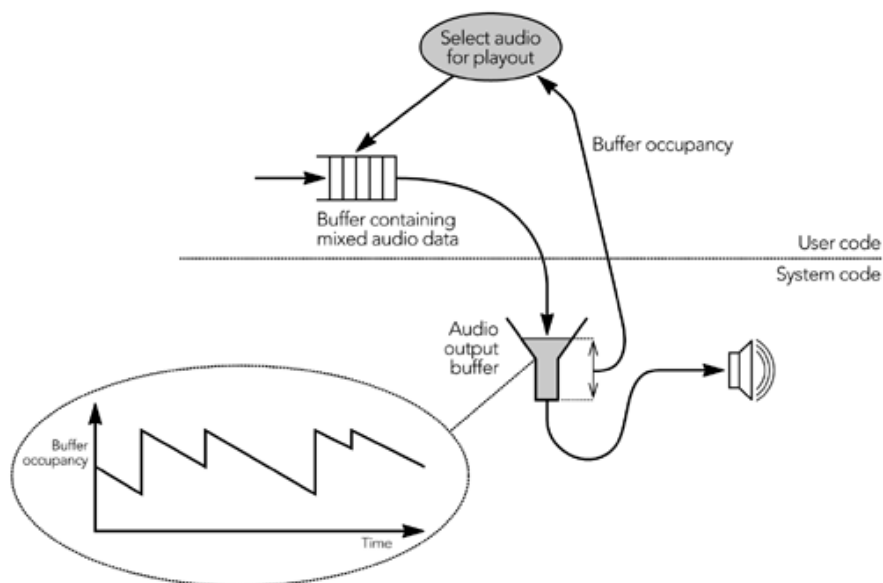


Рис. 6.23. Использование DMA

6.9.4. Проигрывание изображения

Вывод изображения в основном диктуется частотой обновления экрана монитора, которая определяет задержку вывода на него. Плавность вывода зависит от двух основных факторов: одинаковости скорости показа фреймов и отсутствия изменений в уже обрабатываемом фрейме. Первый фактор зависит от размера буфера проигрывания, который был описан в предыдущих разделах. Второй фактор зависит от дисплея. Фреймы не выводятся мгновенно, они появляются в виде серии сканирующих линий: слева направо и сверху вниз. Это оставляет возможность изменения фрейма в процессе его вывода, что может привести к искажению изображения. Эту проблему может решить двойная буферизация, когда в первом буфере хранится фрейм, а во втором происходит его вывод на экран.

7. СИНХРОНИЗАЦИЯ ЗВУКА И ИЗОБРАЖЕНИЯ

Сессия мультимедиа состоит из нескольких потоков, каждый из которых передается в отдельной сессии RTP. Так как задержки, связанные с форматами кодирования, существенно различаются, то в разных потоках будет разное время проигрывания, как показано на рис. 7.1.

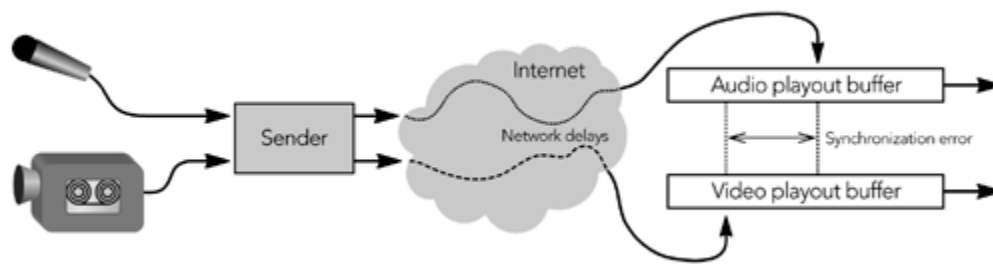


Рис. 7.1. Синхронизация потоков мультимедиа

Обычно синхронизация применяется для выравнивания потоков звука и изображения, но данная технология может применяться для любых типов потоков.

Часто разделение звука и изображения на отдельные потоки диктуют предпочтения участников сессии. Некоторые в видеоконференциях предпочитают получать только звук. Это особенно важно для конференций с большим количеством участников.

7.1. Поведение отправителя

Отправитель помогает в процессе синхронизации потоков мультимедиа путем запуска общих часов и периодического уведомления через пакеты RTCP о рассогласовании общего и потокового времени. Общие часы идут с постоянной скоростью, а получателю дается информация, помогающая выравнивать скорости в потоках, как показано на рис. 7.2.

Соответствие между общим временем и потоковым определяется в момент формирования пакета RTCP. Общее время $T_{reference}$ отражается в метке времени пакета RTP:

$$T_{RTP} = T_{reference} \times R_{audio} + O_{audio}$$

Отсюда получаем смещение часов:

$$O_{audio} = T_{audio} - (T_{available} - D_{audio_capture}) \times R_{audio}$$

Здесь $T_{available}$ определяется задержками операционной системы, а $D_{audio_capture}$ – длительностью процесса захвата данных.

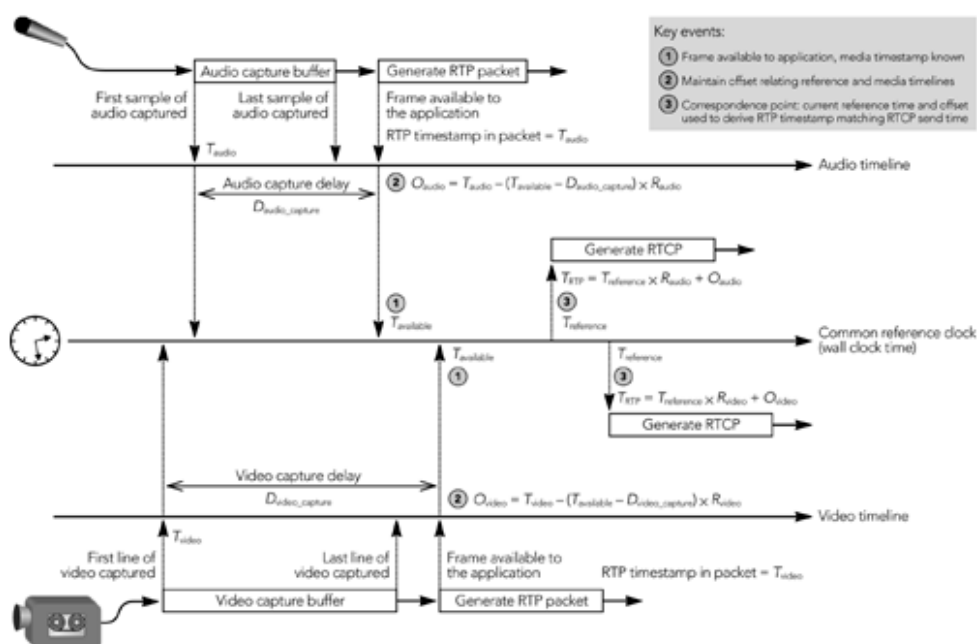


Рис. 7.2. Выравнивание времени по общим часам

На рис. 7.3 демонстрируется необходимость синхронизации часов, когда потоки мультимедиа получаются из разных источников.

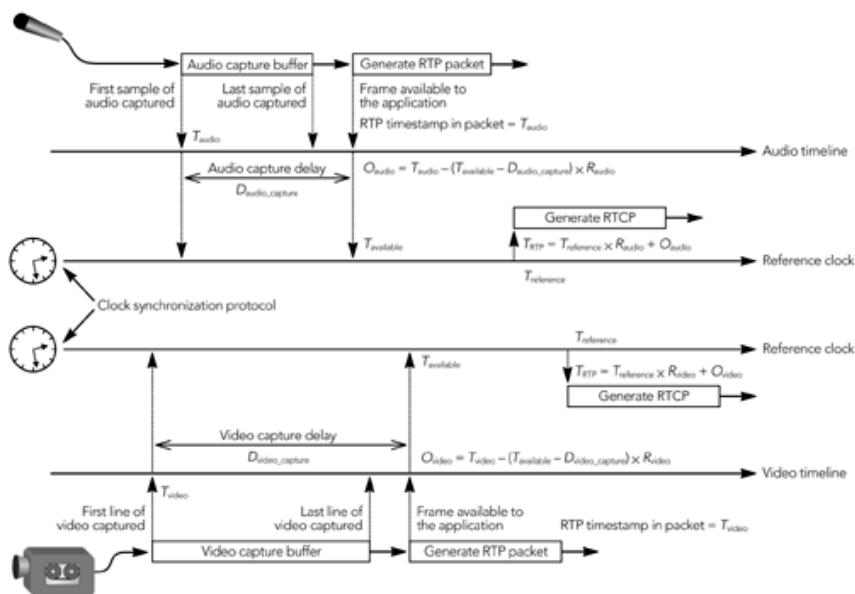


Рис. 7.3. Синхронизация потоков от разных источников

Еще одна проблема синхронизации – идентификация потоков, к которой она должна быть применена. RTP решает ее путем выдачи свя-

занным источникам общих имен (CNAME), поэтому получатель различает зависимые и независимые потоки.

7.2. Поведение получателя

Получатель должен выделить синхронизируемые потоки и выровнять их перед проигрыванием. Первое достигается достаточно просто использованием одинаковых имен CNAME в разных потоках. Сложнее сама процедура синхронизации (рис. 7.4 и 7.5).

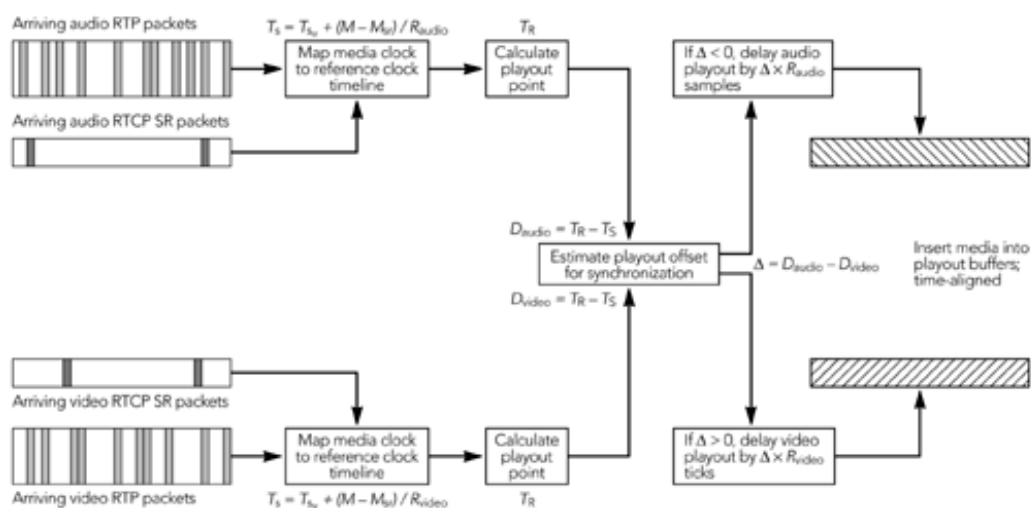


Рис. 7.4. Синхронизация звука и изображения на стороне пользователя

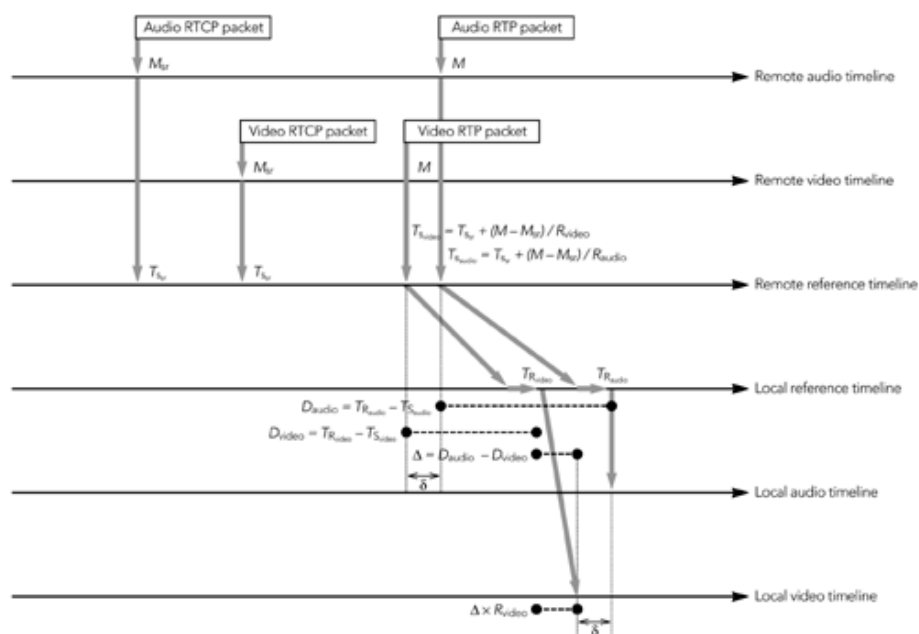


Рис. 7.5. Установление соответствия времен на стороне пользователя

Получатель вначале определяет соответствие между общим временем, установленным отправителем, и временем синхронизируемых потоков путем сравнения данных RTP и RTCP пакетов. При получении пакета данных RTP с меткой времени M может быть вычислено время захвата:

$$T_S = \frac{T_{Ssr} + (M - M_{sr})}{R}.$$

Здесь M_{sr} – метка времени RTP в последнем полученном пакете RTCP, T_{Ssr} – общее время в секундах, R – номинальная скорость часов данных в герцах. Получатель также вычисляет время вывода синхронизированных данных T_R согласно локальным часам. Оно равно метке времени пакета, согласованной с общим временем отправителя, плюс задержка в буфере проигрывания для декодирования, смешивания и обработки.

Когда время захвата и проигрывания известны, получатель может вычислить относительную задержку между захватом данных и их проигрыванием для каждого потока. Если данные захвачены во время T_S по общему времени отправителя и выводятся в момент времени T_R по часам получателя, то разница между ними $D = T_R - T_S$ даст величину задержки между захватом изображения и его выводом. Из-за того, что часы отправителя и получателя не синхронизированы, это значение включает в себя неизвестное смещение между ними, но его можно игнорировать, так как оно одинаково для всех синхронизируемых потоков, а нас интересует только относительное смещение между потоками.

После вычисления этой задержки как для потока звука, так и для потока изображения можно вычислить собственно задержку синхронизации потоков $D = D_{audio} - D_{video}$. Если это значение получится равным нулю, то потоки синхронизированы. В противном случае оно дает смещение между потоками в секундах.

Для опережающего потока данных задержка синхронизации умножается на номинальную скорость потока данных для преобразования значения в формат метки времени данных и затем используется как значение постоянного смещения во всех вычислениях времени.

Пользователь в соответствии со своими приоритетами может выбирать, по какому потоку проводить синхронизацию. Для большинства кодеков кодирование и декодирование видео является доминирующим фактором, но звук более чувствителен к происходящим изменениям.

Задержку синхронизации необходимо пересчитывать при изменении задержки проигрывания любого из потоков. Это также необхо-

димо при изменении соотношения между общим временем и временем потока.

7.3. Точность синхронизации

Может возникнуть вопрос, каким значением задержки между потоками можно пренебречь? Ответ на этот вопрос зависит от ряда факторов, в том числе от того, что синхронизируется и с какой целью. Например, синхронизация звука и изображения может быть достаточно нестрогой и изменяться в зависимости от качества видео и скорости фреймов, в то время как синхронизация звуковых потоков должна быть очень точной.

При синхронизации звука и изображения считается достаточной точность синхронизации в рамках нескольких десятком миллисекунд. Эксперименты с проведением видеоконференций дают граничное значение порядка 80...100 миллисекунд, превышение которого не следует допускать. При повышении качества передаваемого изображения этот порог уменьшается.

8. КОМПЕНСАЦИЯ ОШИБОК

Предыдущие главы рассматривали прохождение потока RTP по протоколам UDP/IP в ненадежных сетях и соответствующее поведение приложений. Возникающие в процессе передачи данных ошибки можно либо компенсировать, либо корректировать. Рассмотрим оба этих варианта.

8.1. Компенсация потерь звука

Когда пакет RTP, содержащий звук, независимо от того, речь это или музыка, потерян, то получатель должен генерировать его заменитель, чтобы обеспечить непрерывность воспроизведения. Это можно сделать разными способами в зависимости от степени чувствительности системы к ошибкам.

8.1.1. Измерение качества звука

Восприятие звука человеком – комплексный процесс, зависящий от многих факторов. Это делает достаточно трудным оценку качества звука. Недостаточно просто сравнить отправленный и принятый звук, потому, что восприятие различий не соответствует этой разнице напрямую. Простые измерители, такие как отношение уровня сигнала к уровню шума, здесь не подходят. Более сложные схемы дают результат, но не со 100 % надежности.

Из-за отсутствия объективных мер измерения приходится применять субъективные тесты. Такие тесты включают проигрывание разных типов музыки, набора слов, фраз и предложений с оценкой качества реальным слушателем по заданной шкале.

Шкала выбирается в зависимости от типа оцениваемых данных. Если это речь, то можно применять шкалу MOS (Mean Opinion Score). Это пятибалльная шкала (excellent = 5, good = 4, fair = 3, poor = 2, bad = 1). Для обеспечения статистической значимости оценки ее надо проводить на большом количестве тестов и разных примерах. Типичное значение качества передачи звука для обычных телефонных переговоров имеет оценку около 4.2, а для мобильной сети – от 3.5 до 4.

Шкала MOS дает достаточно стабильные результаты, но не различает специфику передаваемых данных. Даже при высокой оценке нельзя гарантировать понятность принятой речи. Поэтому данная оценка обычно дополняется серией синтаксических тестов, на основе которых подсчитывается статистическая оценка качества передачи данных.

8.1.2. Замещение периодами тишины

Простейшая техника замещения периодами тишины потерянных пакетов показана на рис. 8.1.

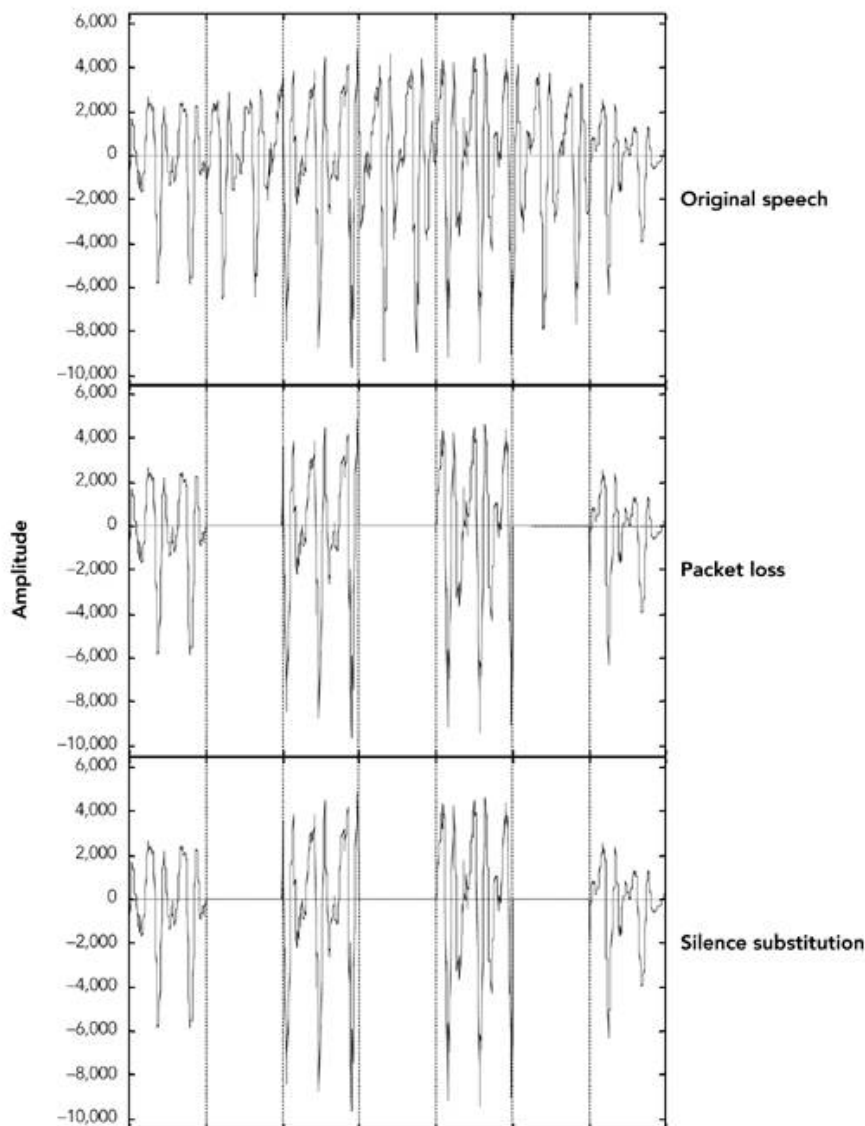


Рис. 8.1. Замещение потерянных пакетов периодами тишины

Данная техника применима только к очень коротким пакетам (менее 16 миллисекунд) при низкой доле потерянных пакетов (менее 2 процентов), иначе она приводит к существенному падению качества передачи данных.

8.1.3. Замещение шума

Процесс замещения шума показан на рис. 8.2.

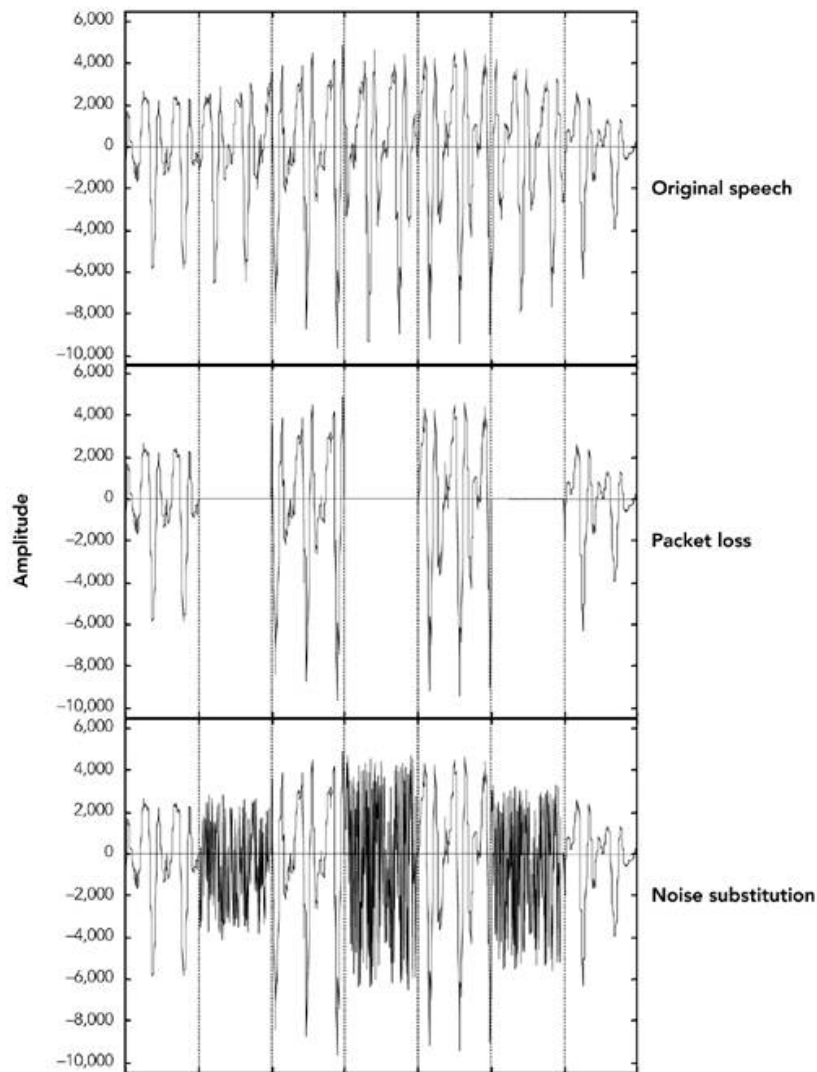


Рис. 8.2. Замещение шума

Ниже приведена программа на псевдокоде, реализующая этот процесс.

```
void
substitute_noise(sample previous_frame[samples_per_frame],
                 sample missing_frame[samples_per_frame])
{
    double energy;

    // Calculate energy (amplitude) of the previous frame
    energy = 0.0;
    for(j = 0; j < samples_per_frame; j++) {
        energy += previous_frame[j] * previous_frame[j];
    }
    energy = sqrt(energy);
    // Fill in the noise
    for(j = 0; j < samples_per_frame; j++) {
        missing_frame[j] = energy * random(-1,1);
    }
}
```

8.1.4. Повторение

На рис. 8.3 показан типичный речевой сигнал.

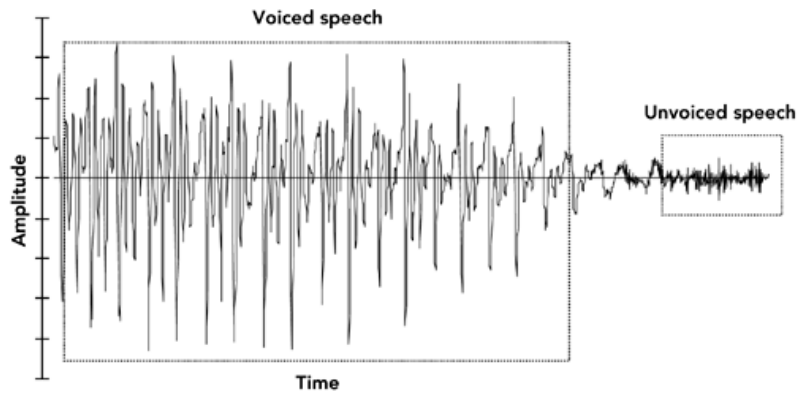


Рис. 8.3. Типичный речевой сигнал

На рис. 8.4 показано восстановление речевого сигнала путем повторения.

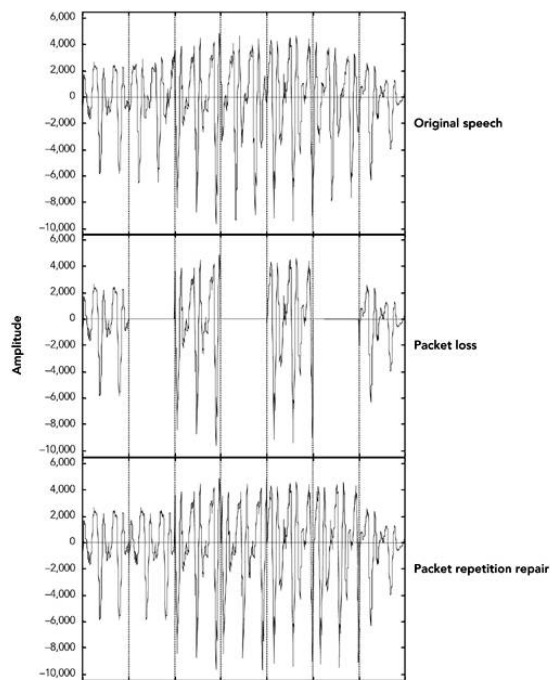


Рис. 8.4. Восстановление речевого сигнала

Соответствующий алгоритм выглядит следующим образом:

```
void  
repeat_and_fade_frame(sample previous_frame[samples_per_frame],  
                    sample missing_frame[samples_per_frame],  
                    int consecutive_lost)
```

```

{
    // Repeat previous frame
    for (j = 0; j < samples_per_frame; j++) {
        missing_frame[j] = previous_frame[j];
    }
    // Fade, if we've lost multiple consecutive frames
    if (consecutive_frames_lost > 0) {
        fade_per_sample = 1 / (samples_per_frame *
            fade_duration_in_frames);
        scale_factor = 1.0 - (consecutive_frames_lost *
            samples_per_frame * fade_per_sample);
        if (scale_factor <= 0.0) {
            // In case consecutive_frames_lost >
            // fade_duration_in_frames
            scale_factor = fade_per_sample = 0.0;
        }
        for (j = 0; j < samples_per_frame; j++) {
            missing_frame[j] *= scale_factor
            scale_factor -= fade_per_sample
        }
    }
}

```

8.1.5. Другие технологии восстановления речевого сигнала

Три приведенных выше подхода образуют базу для восстановления речевого сигнала. Кроме того, существуют технологии, использующие специфику передаваемого сигнала, например, форму волны. При этом вместо потерянного пакета генерируется пакет, содержащий похожую форму волны на базе информации о ее характеристиках. Ниже приведен пример такой генерации на псевдокоде.

```

void pattern_match_repair(sample previous_frame[samples_per_frame],
                        sample missing_frame[samples_per_frame],
                        int consecutive_frames_lost)
{
    // Find best match for the window of the last few samples
    // in the packet
    window_start = samples_per_frame - window_length;
    target = infinity;
    for(i = 0; i < window_start; i++) {
        score = 0;
        for(j = i, k = 0; k < window_length; j++, k++) {
            score += previous_frame[j] -
                previous_frame[window_start + k];
        }
        if (score < target) {
            target = score;
            best_match = i; // The start of the best match for the
                // window
        }
    }
    pattern = best_match + window_length;
    pattern_length = samples_per_frame - pattern;
    // "pattern" now points to the start of the region to repeat.
    // Copy the region into the missing packet
}

```

```

dest = 0;
for (remain = samples_per_frame; remain > 0;
     remain -= pattern_length) {
    for (j = 0; j < min(remain, pattern_length); j++) {
        missing_frame[dest++] = previous_frame[pattern + j];
    }
}
// Fade, if we've lost multiple consecutive frames
if (consecutive_frames_lost > 0) {
    fade_buffer(missing_frame, consecutive_frames_lost);
}
}

```

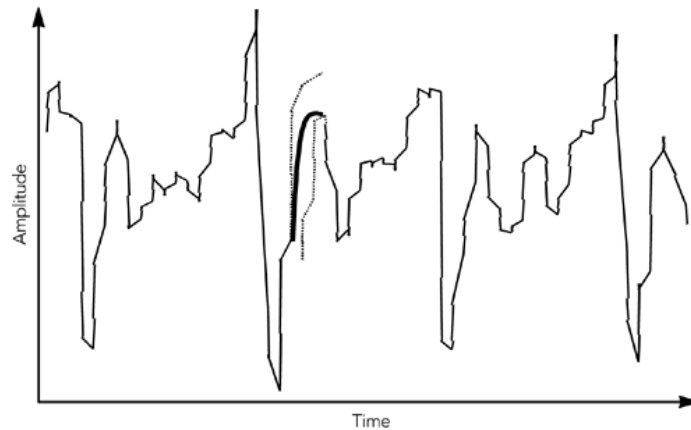


Рис. 8.5. Выравнивание речевого сигнала

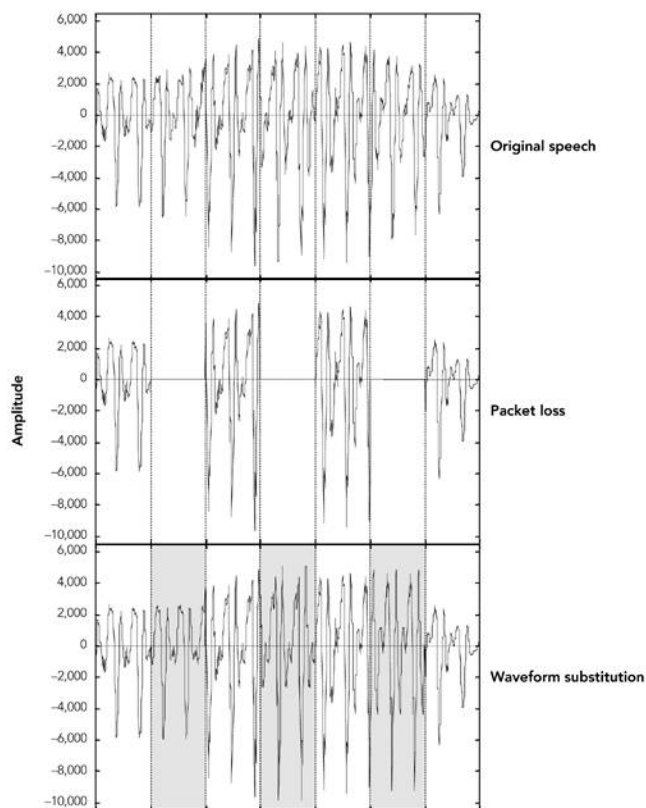


Рис. 8.6. Полный процесс выравнивания речевого сигнала

На границах сгенерированного пакета необходимо добиваться плавности перехода, как показано на рис. 8.5.

Весь процесс восстановления показан на рис. 8.6.

8.2. Компенсация потерь изображения

Многие кодеки видео используют межфреймовую компрессию, посылаю не следующий фрейм, а его отличия от предыдущего, как показано на рис. 8.7.

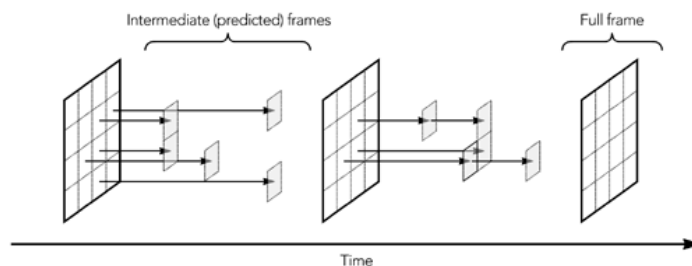


Рис. 8.7. Межфреймовая компрессия

Потеря пакета с промежуточной информацией приведет в этом случае к незначительному искажению части передаваемого кадра. С другой стороны, потеря основного кадра может влиять на ряд следующих пакетов.

8.2.1. Компенсация передвижения

Часто при потере пакета вместо него используют содержимое предыдущего пакета. Но это напрямую допустимо только при отсутствии существенного передвижения в кадре. В противном случае приходится восстанавливать вектор движения, анализируя сохранившиеся кадры. На рис. 8.8 показано восстановление положения единственного потерянного блока.

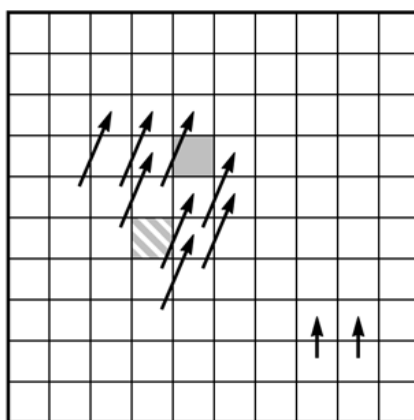


Рис. 8.8. Восстановление потерянного блока

8.3. Чередование (Interleaving)

При применении технологии чередования меняется порядок данных перед их передачей, в результате чего соседние данные передаются на достаточно большом расстоянии между ними. На рис. 8.9 потеря четырех пакетов в потоке с чередованием приводит к потере четырех изолированных пакетов после восстановления исходного порядка данных.

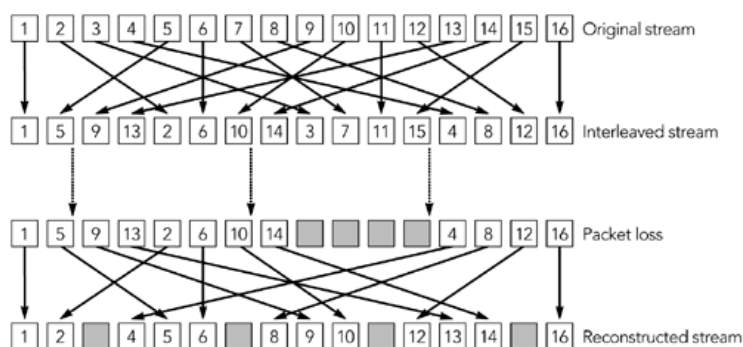


Рис. 8.9. Передача с чередованием

Чередование может применяться на уровне пакетов RTP, но чаще всего происходит чередование фреймов в передаваемых пакетах. В последнем случае в пакетах нарушается последовательность меток времени. Примером применения такой схемы является известный формат для передачи звука MP3.

9. ИСПРАВЛЕНИЕ ОШИБОК

Хотя сглаживание ошибок достаточно важно, лучше их избегать или исправлять. Рассмотрим две базовые категории технологий исправления ошибок.

9.1. Прямое исправление ошибок

Алгоритмы системы прямого исправления ошибок FEC (Forward error correction) преобразуют битовый поток, делая его устойчивым к передаче. Дополнительная информация, содержащаяся в преобразованном битовом потоке, позволяет получателю точно реконструировать исходный битовый поток при наличии ошибок передачи.

Если отправитель пакетов RTP использует FEC, он должен на основе анализа ошибок сети определить долю дополнительной информации корректировки ошибок, помещаемой в каждый пакет. Одним из способов решения данной задачи является наблюдение за пакетами RTCP, приходящими от получателей.

Теоретически путем выбора схемы FEC можно гарантировать отсутствие ошибок при приеме данных. Практически же приходится учитывать, что дополнительная информация, передаваемая в потоке, еще больше нагружает сеть и увеличивает количество ее ошибок. Поэтому речь идет о корректировке ошибок с определенным уровнем вероятности.

Ключевое преимущество схемы FEC заключается в возможности ее применения к большим сессиям, в которых некоторые получатели вообще не отсылают пакетов RTCP, а главный недостаток – в прямой зависимости объемов дополнительной информации от уровня ошибок в сети.

9.1.1. Контроль четности

Контроль четности является простейшим способом обнаружения и корректировки ошибок. Передаваемые данные преобразуются с помощью логической операции XOR:

```
0 XOR 0 = 0
1 XOR 0 = 1
0 XOR 1 = 1
1 XOR 1 = 0
```

На рис. 9.1 показан процесс восстановления данных.

Пакеты FEC могут передаваться и как отдельные потоки (рис. 9.2).

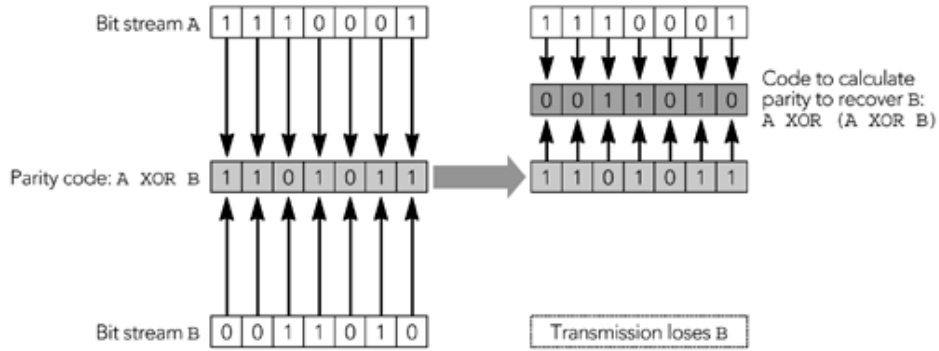


Рис. 9.1. Контроль четности

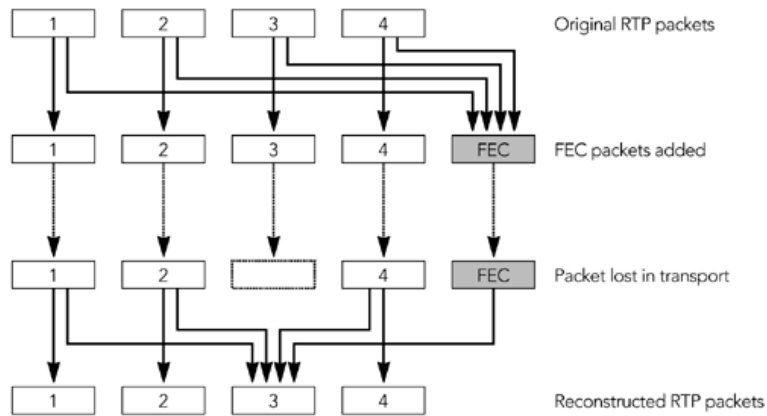


Рис. 9.2. Пакеты FEC в отдельном потоке

Формат пакетов FEC определен в соответствующих стандартах. Некоторые распространенные схемы проверки на четность представлены на рис. 9.3.

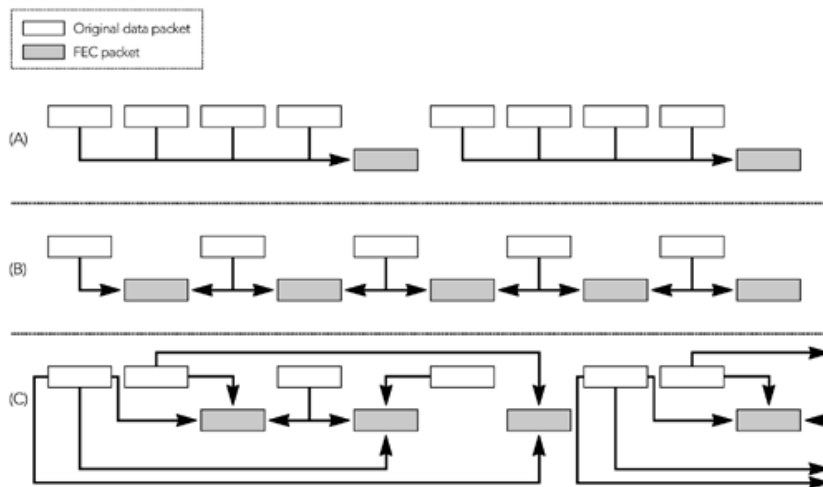


Рис. 9.3. Возможные схемы проверки на четность

Пример сценария проверки на четность:

```
v=0
o=hamming 2890844526 2890842807 IN IP4 128.16.64.32
s=FEC Seminar
c=IN IP4 10.1.76.48/127
t=0 0
m=audio 49170 RTP/AVP 0 122
a=rtpmap:122 parityfec/8000
a=fmtp:122 49172 IN IP4 10.1.76.48/127
```

9.1.2. Неравномерная защита от ошибок

Если данные содержат очень важную часть, то ее можно защитить с помощью неравномерной защиты от ошибок ULP (Unequal Layered Protection). Формат пакета RTP с такой защитой показан на рис. 9.4.

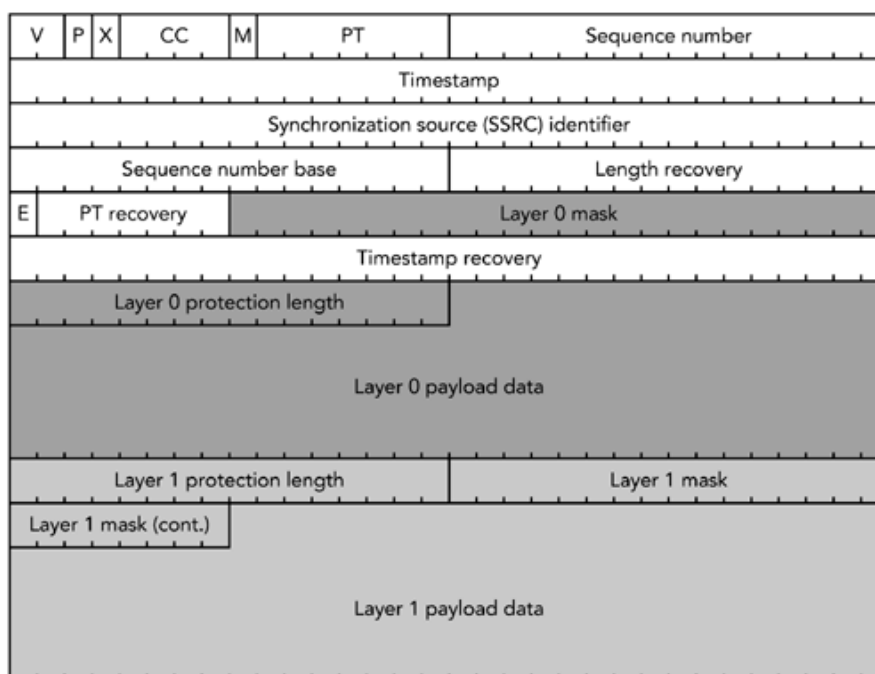


Рис. 9.4. Защита по схеме ULP

9.1.3. Коды Рида-Соломона

Коды Рида-Соломона преобразуют каждый блок данных с помощью коэффициентов полиномиального уравнения. Они меньше загружают каналы передачи, но требуют более сложной обработки.

9.1.4. Избыточное кодирование звука

Схема данного кодирования приведена на рис. 9.5.

Формат соответствующего пакета показан на рис. 9.6.

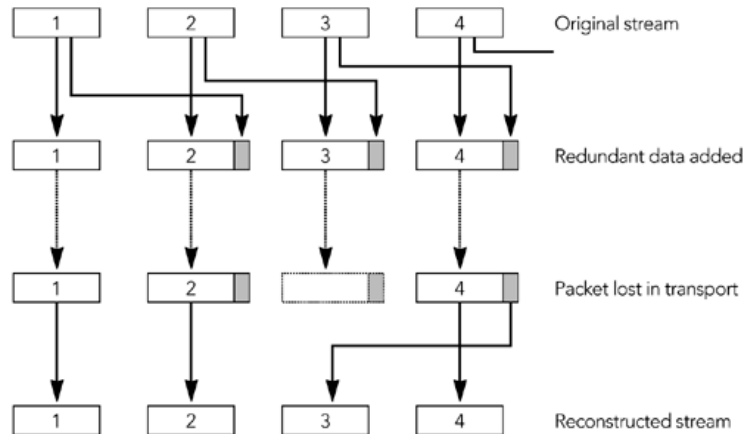


Рис. 9.5. Избыточное кодирование звука

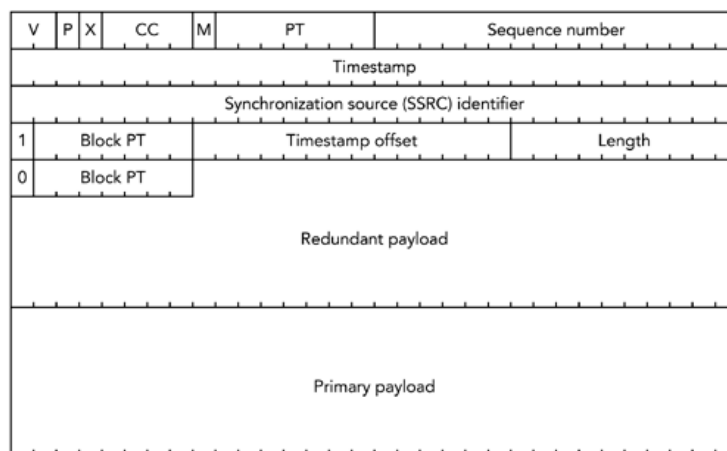


Рис. 9.6. Формат пакета с избыточным кодированием звука

9.2. Кодирование канала

В этом случае для контроля качества передачи используется либо контрольная сумма пакетов UDP, либо частичная контрольная сумма. На рис. 9.7 приведен пример использования частичной контрольной суммы для пакетов данных в формате AMR.

9.3. Повторная передача

Повторная передача является естественным вариантом корректировки ошибок передачи. Повторную передачу можно проводить с использованием канала RTCP, так как именно по нему приходит информация об ошибках передачи.

Стандартное описание RTCP имеет строгие правила управления характеристиками времени. Профиль повторной передачи модифицирует эти правила, что приводит к кратковременным превышениям лимита передачи, хотя в целом скорость передачи остается прежней.

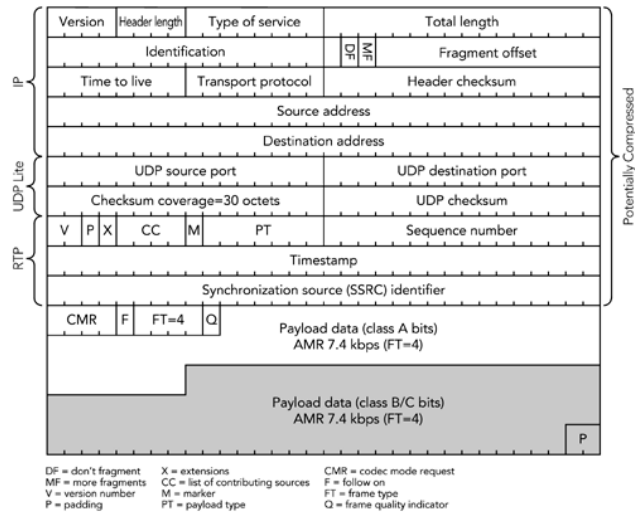


Рис. 9.7. Пакет с частичной контрольной суммой

На рис. 9.8 представлены режимы повторной передачи.

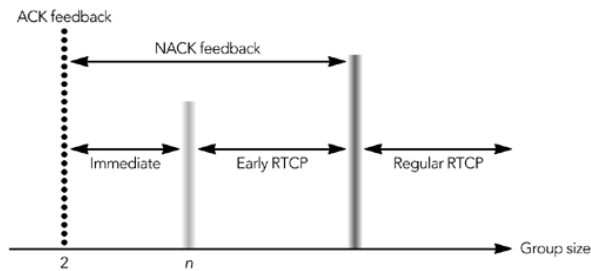


Рис. 9.8. Режимы повторной передачи

Следует отметить, что использование различных схем коррекции ошибок приводит к повышению сложности алгоритмов работы как приложений-отправителей, так и приложений-получателей.

10. КОНТРОЛЬ ПЕРЕГРУЗОК

Результаты перегрузки сети наглядно показаны на рис. 10.1.

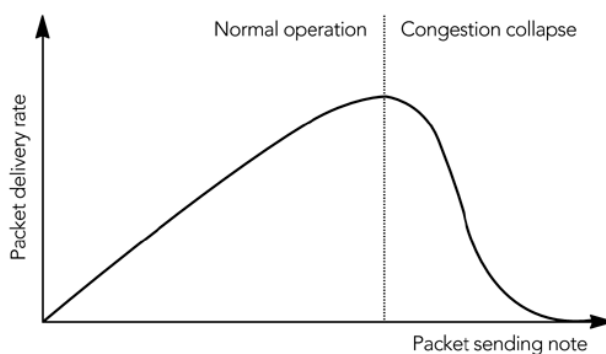


Рис. 10.1. Перегрузка сети

Как видим, в результате перегрузки сети пересылка пакетов может упасть до нуля. Широкое вещание еще более усложняет проблему контроля перегрузки сети, так как отправителям приходится адаптировать передачу данных для большого количества пользователей, требования которых могут быть противоречивы.

Решение проблемы лежит в уровневом кодировании, когда отправитель разделяет передачу данных на несколько групп (уровней) широкого вещания, а получатели подсоединяются только к части доступных групп. Естественно, что такой подход требует и кодеков, которые могут работать с разными уровнями сети передачи. Пример такого кодирования приведен на рис. 10.2.

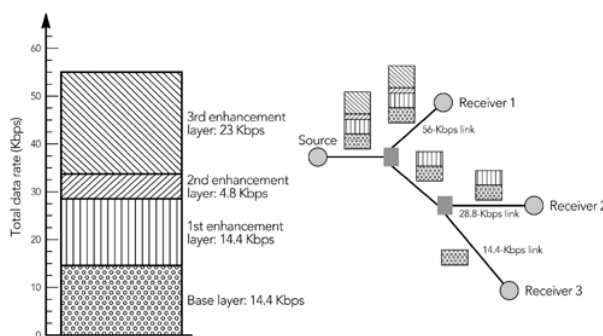


Рис. 10.2. Уровневое кодирование

СПИСОК ЛИТЕРАТУРЫ

1. Perkins C. RTP: Audio and Video for the Internet. Addison Wesley, 2003. – 414 p.
2. V. Paxson. «End-to-End Internet Packet Dynamics,» IEEE/ACM Transactions on Networking, Volume 7, Number 3, June 1999. An earlier version appeared in the Proceedings of ACM SIGCOMM '97, Cannes, France, September 1997.
3. M. Yajnik, S. Moon, J. Kurose, and D. Towsley, «Measurement and Modeling of the Temporal Dependence in Packet Loss,» Technical Report 98–78, Department of Computer Science, University of Massachusetts, Amherst, 1998. © 1999 IEEE.
4. H. Schulzrinne. «RTP Profile for Audio and Video Conferences with Minimal Control,» Internet Engineering Task Force, RFC 1890, January 1996.
5. J. Rosenberg and H. Schulzrinne. «Timer Reconsideration for Enhanced RTP Scalability,» Proceedings of IEEE Infocom '98, San Francisco, CA, March 1998.
6. S. Moon, P. Skelly, and D. Towsley. «Estimation and Removal of Clock Skew from Network Delay Measurements,» Proceedings of IEEE Infocom '99, New York, March 1999.

Научное издание

ЧЕРДЫНЦЕВ Евгений Сергеевич

МУЛЬТИМЕДИЙНЫЕ СЕТИ

Учебное пособие

Научный редактор
доктор технических наук,
профессор *В.А. Силич*

Издано в авторской редакции

Компьютерная верстка *К.С. Чечельницкая*
Дизайн обложки *О.Ю. Аршинова*

Подписано к печати 10.08.2011. Формат 60×84/16. Бумага «Классика».


Печать RISO. Усл.печ.л. 5,99. Уч.-изд.л. 5,42.

Заказ ___-11. Тираж 75 экз.



Томский политехнический университет
Система менеджмента качества
Томского политехнического университета сертифицирована
NATIONAL QUALITY ASSURANCE по стандарту ISO 9001:2000



ИЗДАТЕЛЬСТВО  ТПУ. 634050, г. Томск, пр. Ленина, 30.