ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

А.Ю. ПРИВАЛОВ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ТЕЛЕКОММУНИКАЦИЙ

Самара 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САМАРСКИЙ ГОСУДАРСТВЕННЫЙ АЭРОКОСМИЧЕСКИЙ
УНИВЕРСИТЕТ имени академика С.П. КОРОЛЕВА
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

А.Ю. ПРИВАЛОВ

ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ТЕЛЕКОММУНИКАЦИЙ

Самара 2017

УДК 621.391.2(075)

Рецензенты: к.т.н., доц. Кудрявцев И.А.,
к.т.н., доц. Лафицкий И.В.

Привалов, Александр Юрьевич
Теоретические основы телекоммуникаций. [Электронный учебный ресурс]: электрон. учеб. пособие/ А.Ю. Привалов. – Самара: Изд-во Самарского ун-та, 2017. – Электрон. текст. и граф. дан. (2143 Кбайт) – 113 с.

Пособие содержит изложение основ таких теоретических разделов телекоммуникаций, как теория передачи дискретных сообщений, теория кодирования, теория массового обслуживания в приложении к разделению канала, теория случайного множественного доступа, маршрутизация и управление потоком.

Пособие предназначено для магистрантов, обучающихся по направлению 01.04.02 Прикладная математика и информатика.

Электронное учебное пособие разработано на кафедре прикладных математики и физики.

Стр. 115, Ил. 33, Табл. 8, Библ. 13 наимен.

УДК 621.391.2(075)

СОДЕРЖАНИЕ

Введение

Данное пособие содержит материал, излагаемый в лекциях по курсу “Теоретические основы телекоммуникаций”. Курс обзорный, предназначен для физико-математических и компьютерных специальностей и направлений, которые не относятся напрямую к связи и телекоммуникациям, но при этом, как правило, постоянно имеют дело с сетями передачи данных. Курс сильно математизирован и предполагает у обучающегося предварительное освоение следующих дисциплин на уровне стандартного университетского курса для физико-математических или технических специальностей и направлений**:** “Линейная алгебра и аналитическая геометрия”, “Теория вероятностей и математическая статистика”, “Теория случайных процессов”, раздел “Электричество и магнетизм” курса общей физики, “Теория сигналов” в части, касающейся преобразований Фурье и линейных систем. Также полезно предварительное знакомство с основами радиотехники, основами теории алгоритмов, а также с конечными алгебрами, особенно с алгебрами полиномов.

Курс содержит обзор теоретических основ и некоторые иллюстрирующие их примеры из практики, касающихся передачи данных, в особенности главной задачи, решаемой при организации передачи данных – **обеспечению надёжной передачи** (здесь и далее наиболее важные понятия и определения выделены жирным шрифтом). Под передачей данных принято понимать передачу какой-либо информации в цифровой форме, а под надёжностью передачи – обеспечение того, что несовпадение между переданной и принятой информацией происходит с вероятностью меньшей, чем заданный, очень малый, как правило, порог. То, что проблема надёжной передачи – это центральная проблема передачи данных легко понять из следующего примера**:** если передаваемые данные – это компьютерная программа, то искажение хотя бы одного бита может привести к непредсказуемым последствиям при её запуске.

На практике наряду с требованием надёжности передачи данных, также приходится учитывать и другие требования, в частности, эффективное использование ресурсов системы связи, простота её реализации и т.д. Этим задачам в курсе также уделено некоторое внимание.

Вопросы, рассматриваемые в курсе, обычно относят к таким разделам теории связи, как теория передачи дискретных сообщений, теория кодирования, теория массового обслуживания, теория случайного множественного доступа и, собственно, теория сетей связи. Для того, чтобы изложение теоретических основ не было слишком абстрактным, некоторое внимание уделено примерам практического применения рассматриваемых теоретических результатов. Оно, естественно, весьма поверхностно, и часто касается не самых современных систем связи, так как служит только для иллюстрации идей, и в более простых системах предыдущих поколений они лучше заметны. Также для связи с практикой, мы будем иногда указывать, к какому уровню широко известной семиуровневой модели взаимодействия открытых систем относится практическое применение рассматриваемых теоретических концепций.

Поскольку изложенный в пособии теоретический материал относится к основам теории связи, в списке литературы, кроме более-менее современных учебников, приведены классические учебники и монографии, где рассматриваемые в курсе вопросы изложены (по мнению автора) проще и понятней всего. Они довольно старые, и за прошедшее с момента их издания время появилось, разумеется, множество других замечательных книг по данным темам. При этом, излагая современное состояние теории связи, современные учебники и монографии меньше внимания уделяют основам, а больше - последним достижениям, что совершенно правильно, если речь идёт о подготовке профессионалов-связистов. Но для знакомства с проблематикой данной области тех, кто напрямую с ней не связан, это может создать дополнительные трудности. Поэтому, по мнению автора, в данном случае справедливо известное высказывание о том, что “классика не стареет”.

Автор хотел бы выразить глубокую благодарность рецензентам И.А. Кудрявцеву и И.В. Лофицкому за ценные замечания, учёт которых, безусловно, повысил уровень данного учебного пособия.

1. Некоторые необходимые сведения из физики, теории сигналов и радиотехники

В современных сетях телекоммуникаций, в том числе и в сетях передачи данных, для переноса информации используются, как правило, электромагнитные сигналы, то есть колебания электромагнитного поля. Существующие в природе электромагнитные колебания различной частоты обладают очень разнообразными свойствами относительно их генерации, распространения, взаимодействия с веществом и т.д. Достаточно вспомнить такие явления, как электрический ток, радиоволны, оптические явления, рентгеновское и гамма-излучение. В современной технике связи используется нижняя часть электромагнитного спектра вплоть до оптического диапазона включительно. Но и здесь свойства электромагнитных колебаний весьма разнообразны, и также разнообразны технические решения, использующие тот или иной вид электромагнитных колебаний для переноса информации. Примерами являются проводные и кабельные линии связи, оптоволокно, радиосвязь, как наземная, так и с космической компонентой, инфракрасные беспроводные сети. Однако есть наиболее общие принципы применения электромагнитных сигналов для передачи информации, которые используются практически во всех современных системах связи. В основном они и понадобятся нам в данном курсе, и потому мы их кратко напомним.

Прежде всего, отметим, что в процессе приёма и обработки, электромагнитные сигналы преобразуются в сигналы электрические. Поэтому на самом деле, нам понадобятся наиболее общие свойства электрических сигналов, и некоторые сведения о наиболее общих методах их обработки.

С самой общей точки зрения способность сигнала совершать какие-либо изменения в окружающем мире (и потому, в частности, являться переносчиком информации), определяется **энергией сигнала**. Из физики известно, что мгновенная мощность электрического сигнала $s(t)$ пропорциональна квадрату его амплитуды, а полная его энергия, таким образом, пропорциональна интегралу от квадрата сигнала по всему интервалу времени, когда сигнал отличен от нуля. Коэффициент пропорциональности для нас не важен, и для простоты мы будем считать, что энергия E_s сигнала $s(t)$ просто равна величине этого интеграла, то есть в общем случае

$$E_s = \int_{-\infty}^{+\infty} s^2(t)dt \qquad (1)$$

Нам также понадобятся некоторые результаты из теории сигналов, касающиеся прохождения сигнала через **линейные фильтры,** то есть устройства с одним входом и одним выходом, обладающие свойством линейности. Если на вход фильтра подаётся некоторый сигнал, на выходе фильтра появляется другой сигнал, называемый **откликом**. Линейность фильтра заключается в наличии следующих свойств:

1). Если $v(t)$ – отклик на $s(t)$ и a- константа, то откликом на $as(t)$ будет $av(t)$.

2). Если $v(t)$ – отклик на $s(t)$ и $w(t)$ – отклик на $g(t)$, то откликом на сигнал $s(t)+ g(t)$ будет $v(t)+ w(t)$.

Линейный фильтр является очень удобной начальной моделью для многих природных явлений и технических систем, в частности, для каналов передачи сигналов. Сигнал, прошедший через канал, считается откликом моделирующего канал фильтра на сигнал, который в канал был передан. Таким образом, к задаче описания прохождения сигнала по каналу связи, может быть применён хорошо развитый математический аппарат линейной фильтрации. Свойство линейности позволяет действовать следующим образом: представив входной сигнал линейной комбинацией сигналов некоторого удобного базиса, и зная отклик фильтра на каждую из базисных функций, отклик на исходный сигнал получим как линейную комбинацию откликов на базисные функции с теми же коэффициентами, которые были у этих базисных функций во входном сигнале.

Очень удобным базисом для таких целей являются мнимые экспоненты, разложение сигнала по которым эквивалентно его **прямому преобразованию Фурье:**

$$S(f) = \int_{-\infty}^{+\infty} s(t) e^{-i2\pi f t} dt \qquad (2)$$

Комплексная функция $S(f)$, называемая **спектром сигнала** – это коэффициенты разложения сигнала по функциям гармонических колебаний $e^{i2\pi ft}$:

$$s(t) = \int_{-\infty}^{+\infty} S(f) e^{i2\pi f t} df \qquad (3)$$

Если у фильтра известна его **частотная характеристика** $H(f)$, которая для каждой частоты f характеризует отклик фильтра на вход $e^{i2\pi ft}$ (в том смысле, что на выходе будет $H(f)\, e^{i2\pi ft}$), то спектр выходного сигнала $V(f)$ легко найти из формулы

$$V(f) = H(f) S(f), \qquad (4)$$

а форму сигнала на выходе можно найти по формуле **обратного преобразования Фурье** (3), подставив в неё спектр выходного сигнала $V(f)$ из (4). Другие нужные нам результаты из теории линейной фильтрации мы будем вспоминать по ходу нашего рассмотрения.

В технике связи, как правило, желательно, чтобы при передаче сигнала по каналу его форма сохранялась. Это можно описать следующим образом: если на входе в канал был сигнал $s(t)$, то на выходе желательно иметь $v(t)=as(t-\tau)$, то есть ослабленный в a раз и задержанный на τ, но сохранивший форму исходный сигнал. Нетрудно доказать, что для этого частотная характеристика канала должна иметь вид

$$H(f) = ae^{-i2\pi f\tau}, \qquad (5)$$

то есть модуль частотной характеристики (как комплексного числа) должен быть постоянным, а фаза линейно зависеть от частоты. Формулу (5) будем называть **условием неискажающей передачи**. Для реальных каналов связи условие неискажающей передачи приближённо выполняется только в некотором интервале оси частот, который принято называть **полосой частот канала**. Понятно, что передавать без искажений такие каналы могут лишь сигналы, спектр которых отличен от нуля только внутри полосы частот канала. На практике это может выполняться лишь приблизительно, но для простоты мы будем считать, что это имеет место.

На оси частот полоса частот канала может быть расположена по-разному. Заметим, что формально при преобразовании Фурье частоты могут быть как положительными, так и отрицательными, но на практике считают, что частоты принимают только неотрицательные значения. Пусть нижняя граница полосы частот канала обозначается f_b, верхняя - f_u, тогда $f_0=(f_b+f_u)/2$ - середина полосы частот канала. Принято говорить, что канал является **узкополосным**, если

$$f_u - f_b << f_0$$

Если же f_b-f_u того же порядка, что и f_0, то канал называется **широкополосным**.

По узкополосным каналам могут передаваться сигналы, спектр которых весь сосредоточен вблизи частоты f_0, и потому они сами “похожи” на синусоиду этой частоты. По широкополосным каналам могут передаваться сигналы, в спектре которых присутствуют очень разные частоты, и потому эти сигналы могут иметь гораздо более разнообразную форму.

Для описания методов обработки сигналов, которые могут использоваться при приёме, нам понадобятся также и другие, кроме линейной фильтрации, операции, которые радиотехнические устройства могут производить с сигналами, как с функциями времени. К числу таких операций можно отнести интегрирование и дифференцирование сигнала, сложение и перемножение двух сигналов, а также сравнение двух сигналов между собой или сигнала с заданным порогом. Остальные операции (например, дискретизация или квантование) строятся, как комбинации вышеперечисленных.

2. Предмет теории передачи дискретных сообщений

Простейшая система связи, с которой мы начнём наше рассмотрение проблем организации надёжной связи (то есть связи, при которой вероятность значимого несовпадения между переданной и принятой информацией может быть сделана достаточно малой), включает источник информации, передатчик, канал связи, приёмник и получателя информации (см. рис. 2.1.)

Прежде всего заметим, что в основе проблем связи вообще, и надёжной связи в частности, лежит наличие в реальных системах всевозможных случайностей. Например, если бы получатель всегда заранее точно знал, что именно и в какой момент ему должен передать источник сообщений, то вообще отсутствовала бы необходимость в системе связи. Поэтому предполагается, что выход источника сообщений является случайным и его нельзя с определённостью предсказать на приемном конце.

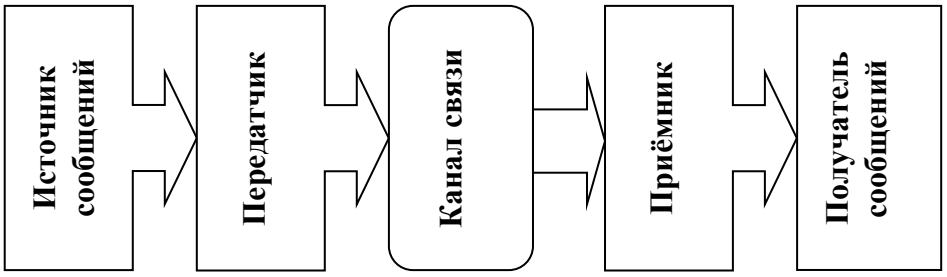

Рисунок 2.1. Простейшая система связи.

Менее очевидно то, что проблемы надёжной связи не существовало бы, если бы передаваемый сигнал не искажался случайным образом во время генерации, распространения по каналу связи и приема. Для иллюстрации этого факта представим себе, что наш передатчик может генерировать любое значение напряжения между 0 и 1 В, и оно без искажений передается на приемник. Предположим, что передаваемое сообщение представляет собой текст из латинских букв, цифр, знаков препинания и некоторых специальных знаков, всего 128 возможных символов. Каждому из них можно сопоставить семизначное двоичное число, например, согласно известному всем коду ASCII (или восьмизначное для расширенного кода ASCII с 256 возможными символами). Тогда все содержимое этого текста, независимо от длины, представляется в виде двоичной последовательности, где первые семь двоичных символов - это первая буква, следующие семь - вторая и т.д. Получившуюся последовательность можно интерпретировать как двоичное число между нулем и единицей, если перед ней поставить ноль и запятую. Тогда, передав это число одним значением сигнала, мы передадим и весь текст, независимо от его длины.

Такой способ, очевидно, практически непригоден, так как малые искажения сделают либо прием, либо передачу сигнала с такой невероятной точностью невозможными. Нетрудно также прийти из общих соображений к выводу, что чем больше сигнал подвержен неконтролируемым случайным искажениям, тем хуже условия для возможности передавать информацию с помощью этого сигнала. Поэтому одна из центральных проблем теории связи - борьба с влиянием случайных искажений на передаваемый сигнал.

Борьба со случайными искажениями может пониматься по-разному, в зависимости от того, что мы знаем о сигналах или о сообщениях, для передачи и приёма которых создана система связи. Например, если мы можем ожидать любой сигнал (это означает, что мы ничего или почти ничего о нем не знаем), то необходимо воспроизвести его на приемном конце как можно точнее. Эта задача возникает, в частности, при передаче и

приеме аналоговых сигналов (в телевидении, радиовещании, аналоговой телефонии и т.д.).

Другая ситуация - когда мы почти всё о сигнале знаем, кроме нескольких его параметров, например, времени прихода и амплитуды. В этом случае нам надо понять, пришел сигнал или нет, и если пришёл, то определить эти параметры. Это задача называется задачей обнаружения сигнала, и возникает, в частности, в радиолокации.

Третья ситуация - когда передатчик может генерировать конечное, заранее известное множество сигналов, и приёмник должен сделать выбор относительно того, какой именно из этих сигналов передавался. Это и есть задача передачи и приема дискретных сообщений, возникающая, в частности, в цифровых сетях телекоммуникаций.

Для возможности содержательного анализа, определим математические модели отдельных частей системы связи, изображённой на рис.2.1. Относительно источника дискретных сообщений будем предполагать, что на его выходе может появляться сообщение m – элемент множества $\{m_i\}_{i=1,\dots,M}$ возможных сообщений. С достаточной степенью общности можно считать, что при появлении на выходе источника сообщения $m=m_i$, передатчик передаёт в канал сигнал $s(t)=s_i(t)$, т.е. передаваемое сообщение однозначно соответствует сигналу. Далее при передаче по каналу сигнал случайным образом искажается и вместо $s(t)$ в приёмник приходит сигнал $r(t)$. Приемник по принятому сигналу $r(t)$ должен построить оценку $\hat{m}$ переданного сообщения. Если при этом $\hat{m} \neq m$, то произошла ошибка, если же $\hat{m} = m$, то передача прошла без ошибок.

Далее мы рассмотрим задачу нахождения алгоритма работы **оптимального приемника**, минимизирующего вероятность ошибки приёма, то есть алгоритма определения $\hat{m}$, минимизирующего $\Pr\{\hat{m} \neq m\}$ (или, что тоже самое, максимизирующего вероятность правильного приема $\Pr\{\hat{m} = m\} = 1 - \Pr\{\hat{m} \neq m\}$).

Понятно, что в общем случае оптимальный приёмник должен использовать для принятия решения информацию о множестве возможных сигналов $\{m_i\}$, принятый сигнал $r(t)$ и искажающие свойства канала, которые можно описать в общем случае совместным распределением случайных процессов $r(t)$ и $s(t)$, которое определяется бесконечным семейством конечномерных совместных распределений. Такая постановка задачи слишком сложна для содержательного анализа, и потому нам необходимо упростить нашу модель канала связи.

3. Переход от непрерывного канала к векторному

Предположим, что сигнал на выходе передатчика может быть полностью описан N-мерным вектором $\vec{s} = (s_1, \dots, s_N)$, и сигнал на выходе канала также будем описывать N-мерным вектором $\vec{r} = (r_1, \dots, r_N)$. Тогда искажающие свойства канала связи можно описать с помощью условного распределения с плотностью вероятности $f(\vec{r} \mid \vec{s})$ - что существенно проще для использования в нашем анализе.

Сразу же обсудим вопрос о возможности (в принципе) рассмотрения реальных каналов связи с непрерывными сигналами как каналов, передающих векторные сигналы. Прежде всего, заметим, что любой сигнал из конечного набора сигналов $\{s_i(t)\}_{i=1,\dots,M}$, может быть представлен в виде

$$s_i(t) = \sum_{k=1}^{N} s_{ik} \phi_k(t), \quad \text{где} \quad N \leq M \tag{6}$$

для некоторого фиксированного набора функций $\{\varphi_i(t)\}_{i=1,\dots,N}$, который будем называть базисом для системы сигналов $\{s_i(t)\}_{i=1,\dots,M}$, а сами $\varphi_i(t)$ - базисными функциями. Очевидно, что такое представление всегда возможно, по крайней мере, для $N=M$ (на практике же часто $N<M$ и даже $N<<M$).

Более того, для любого конечного набора сигналов $\{s_i(t)\}_{i=1,\dots,M}$ с конечной энергией (то есть для которых $\int_{-\infty}^{+\infty} s_i^2(t)dt < \infty \quad \forall i$) всегда можно найти **ортонормальный базис**, то есть набор функций $\{\varphi_i(t)\}_{i=1,\dots,N}$, таких что

$$\int_{-\infty}^{+\infty} \phi_i(t)\phi_j(t)dt = \delta_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases} \tag{7}$$

Ортонормальные базисы обладают рядом свойств, которые понадобятся нам в дальнейшем, поэтому далее везде, где не оговорено обратное, будем считать, что базис является ортонормальным. Процедура построения такого базиса для заданного набора сигналов называется **процедурой ортогонализации Грама - Шмидта** и в данном случае выглядит следующим образом: пусть задан набор $\{s_i(t)\}_{i=1,\dots,M}$ сигналов с ограниченной энергией.

1) Возьмём сигнал $s_1(t)$. Если $s_1(t) \equiv 0$, перенумеруем сигналы, чтобы этого не было. Положим

$$\phi_1(t) = \frac{s_1(t)}{\sqrt{E_1}}, \quad \text{где} \quad E_1 = \int_{-\infty}^{+\infty} s_1^2(t)dt \tag{8}$$

Так как E_1 - энергия сигнала $s_1(t)$, то $\varphi_1(t)$ имеет единичную энергию, и $s_1(t)=s_{11}\varphi_1(t)$, где $s_{11} = \sqrt{E_1}$, то есть $\vec{s}_1 = (\sqrt{E_1}, 0, \dots 0)$.

2) Далее пусть $s_2(t) \neq 0$. Определим

$$\theta_2(t) = s_2(t) - s_{21}\phi_1(t), \quad \text{где} \quad s_{21} = \int_{-\infty}^{+\infty} s_2(t)\phi_1(t)dt \tag{9}$$

Если $\theta_2(t) \neq 0$, то положим

$$\phi_2(t) = \frac{\theta_2(t)}{\sqrt{E_2}}, \quad \text{где} \quad E_2 = \int_{-\infty}^{+\infty} \theta_2^2(t)dt, \tag{10}$$

тогда $\varphi_2(t)$ также имеет единичную энергию и $s_{22} = \sqrt{E_2}$. При этом $\varphi_1(t)$ и $\varphi_2(t)$ ортогональны, так как

$$\int_{-\infty}^{+\infty} \phi_1(t)\phi_2(t)dt = \frac{1}{\sqrt{E_2}} \int_{-\infty}^{+\infty} \theta_2(t)\phi_1(t)dt = \frac{1}{\sqrt{E_2}} \left(\int_{-\infty}^{+\infty} s_2(t)\phi_1(t)dt - s_{21} \int_{-\infty}^{+\infty} \phi_1^2(t)dt \right) = \frac{1}{\sqrt{E_2}} (s_{21} - s_{21}) = 0$$

При этом $\vec{s}_2 = (s_{21}, \sqrt{E_2}, 0, \dots 0)$. Если же $\theta_2(t)=0$, то сигнал $s_2(t)$ не порождает новой базисной функции и выражается через $\varphi_1(t)$ как $s_2(t)=s_{21}\varphi_1(t)$, а его векторное представление в данном базисе будет $\vec{s}_2 = (s_{21}, 0, \dots 0)$.

3) Пусть проведено n-1 шагов процедуры ортогонализации; n-й шаг состоит в следующем: пусть по сигналам $s_1(t), \dots, s_{k-1}(t)$, мы определили n-1 ортонормальных функций $\varphi_1(t), \dots, \varphi_{n-1}(t)$. Понятно, что $n \leq k$, так как каждый новый сигнал порождает не более одной ортонормальной функции. Пусть $s_k(t) \neq 0$, тогда

$$\theta_k(t) = s_k(t) - \sum_{j=1}^{n-1} s_{kj}\phi_j(t), \quad \text{где} \quad s_{kj} = \int_{-\infty}^{+\infty} s_k(t)\phi_j(t)dt \tag{11}$$

Если $\theta_k(t) \neq 0$, то положим

$$\phi_n(t) = \frac{\theta_k(t)}{\sqrt{E_k}}, \quad \text{где} \quad E_k = \int_{-\infty}^{+\infty} \theta_k^2(t)dt \tag{12}$$

Такая $\varphi_n(t)$ имеет единичную энергию, и $s_{kn} = \sqrt{E_k}$.

Кроме того,

$$\int_{-\infty}^{+\infty} \phi_n(t)\phi_j(t)dt = 0 \quad \forall j = 1, \dots, n-1 \tag{13}$$

так как

$$\int_{-\infty}^{+\infty}\phi_n(t)\phi_j(t)dt=\frac{1}{\sqrt{E_k}}\int_{-\infty}^{+\infty}\theta_k(t)\phi_j(t)dt=\frac{1}{\sqrt{E_k}}\left(\int_{-\infty}^{+\infty}s_k(t)\phi_j(t)dt-\sum_{i=1}^{n-1}s_{ki}\int_{-\infty}^{+\infty}\phi_i(t)\phi_j(t)dt\right)=$$

$$=\frac{1}{\sqrt{E_k}}\left(s_{kj}-\sum_{i=1}^{n-1}s_{ki}\delta_{ij}\right)=0$$

Если $\theta_k(t)=0$, то $s_k(t)$ полностью раскладывается по n-1 имеющимся функциям ортонормального базиса и не порождает новой базисной функции.

Пункт 3 продолжается до тех пор, пока не будут исчерпаны все $s_i(t)$. В результате будут найдены $N \le M$ ортонормальных функций, причем равенство будет иметь место тогда и только тогда, когда все M сигналов линейно независимы, то есть ни один из них не может быть представлен линейной комбинацией других. N называется **размерностью пространства сигналов**, образованного сигналами $\{s_i(t)\}_{i=1,\dots,M}$.

Итак, любой сигнал из произвольного набора из M сигналов с конечной энергией $\{s_i(t)\}_{i=1,\dots,M}$., может быть представлен в виде вектора $\vec{s}_i=(s_{i1},s_{i2},\dots,s_{iN})$ в некотором ортономальном базисе. Что касается принципиальной возможности практической реализации векторной передачи, то с помощью линейных фильтров несложно построить принципиальную схему передатчика (рис. 3.1), осуществляющего синтез сигнала, представленного в виде вектора в некотором базисе $\{\varphi_i(t)\}_{i=1,\dots,N}$.

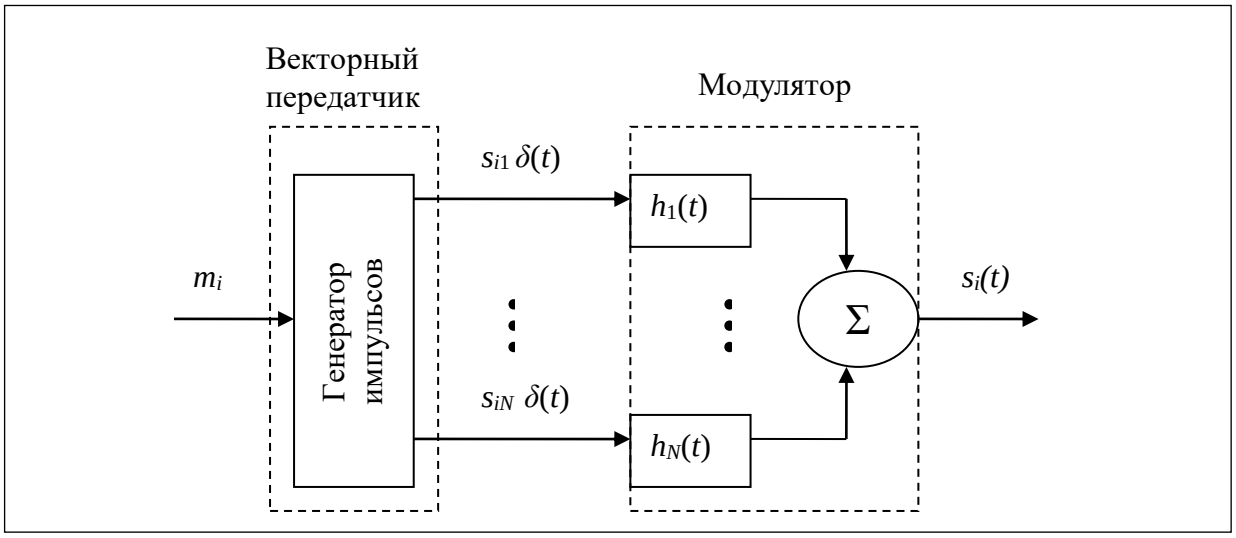

Рисунок 3.1. Принципиальная схема передатчика для модели векторного канала

Работает схема следующим образом: сообщение m_i, которому соответствует вектор $\vec{s}_i=(s_{i1},s_{i2},\dots,s_{iN})$, поступает от источника на генератор импульсов, который имеет N выходов. На каждый из своих выходов этот генератор выдаёт короткий импульс, являющийся практическим приближением к дельта-функции, при этом на выход k подаётся сигнал, приближённо равный $s_{ik}\delta(t)$. Далее этот сигнал поступает на соответствующий линейный фильтр, имеющий импульсную характеристику $h_i(t)=\varphi_i(t)$ (напомним, что **импульсная характеристика** фильтра – это его отклик на дельта-функцию). На выходе фильтра получается сигнал $s_{ik}\varphi_k(t)$, а после суммирования сигналов со всех N фильтров - сигнал $s_i(t)$, который и передаётся в канал.

Для простоты изложения принципов векторного приёма, примем следующее ограничение: здесь и далее все базисные функции $\{\varphi_i(t)\}_{i=1,\dots,N}$. отличны от нуля только в промежутке времени $[0,T]$ для некоторого конечного T. Тогда и все сигналы, которые можно получить в данном базисе, тоже могут быть отличны от нуля только в интервале

[0,*T*]. На практике такое ограничение всегда выполняется, так как в реальных системах связи все сигналы ограничены по времени.

Пусть в нулевой момент времени из канала на вход приёмника, принципиальная схема которого изображена на рис. 3.2, начинает поступать из канала сигнал *r*(*t*). При этом в тот же самый момент времени на каждый из присутствующих в схеме *N* умножителей начинает поступать соответствующая функция $\varphi_k(t)$. Тогда на выходе соответствующего интегратора в момент времени *T* получится выражение

$$r_k = \int_0^T r(t)\phi_k(t)dt, \quad k = 1,...N \tag{14}$$

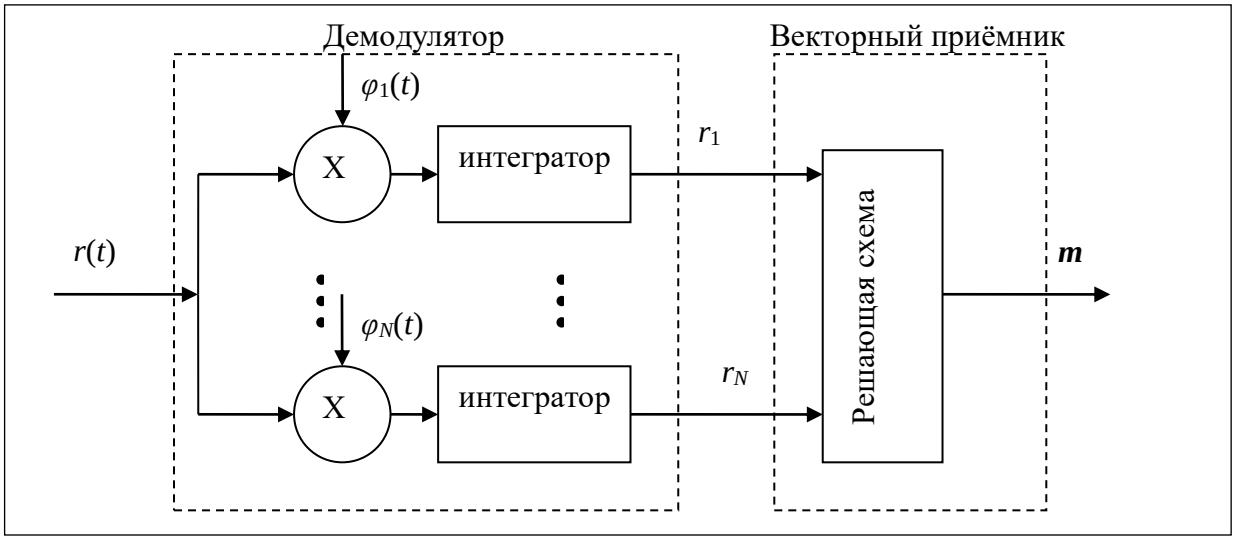

Рисунок 3.2. Принципиальная схема приёмника для модели векторного канала

Нетрудно видеть, что если бы $r(t)$ совпадал с каким-либо $s_i(t)$, то получилось бы соответствующее векторное представление $\vec{s}_i = (s_{i1}, s_{i2}, ..., s_{iN})$. Однако в общем случае

$$r(t) \neq \sum_{k=1}^{N} r_k \phi_k(t),$$

так как из-за искажений в канале связи, сигнал *s*(*t*), первоначально представимый в базисе $\{\varphi_i(t)\}_{i=1,...,N}$., мог превратиться в сигнал, в этом базисе непредставимый. Тем не менее, если кроме $\vec{r}$ решающая схема приёмника не имеет другой информации о *r*(*t*), то мы можем считать, что из канала принимается вектор $\vec{r}$. Эти значения поступают на решающую схему, которая по ним должна построить оценку $\hat{m}$ и передать её получателю.

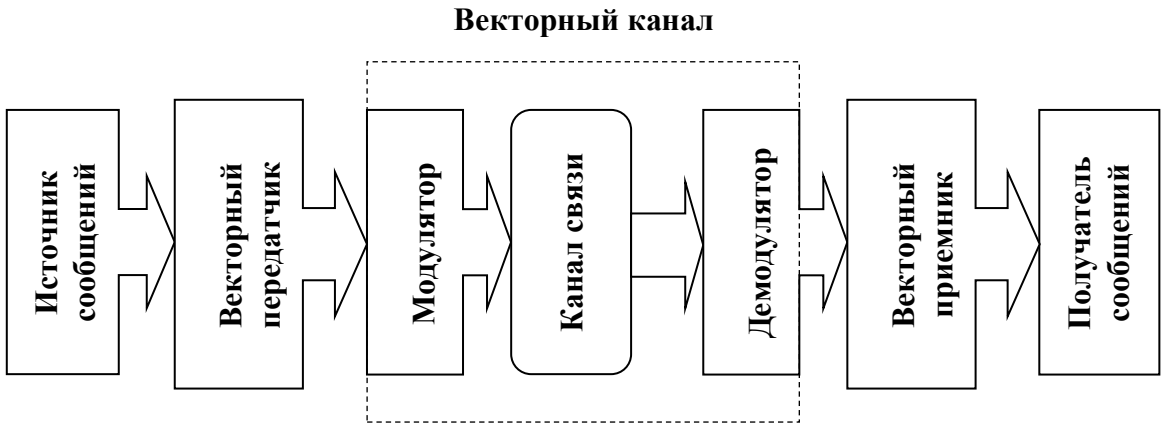

Рисунок 3.3. Модель системы связи с векторным каналом

Если теперь перерисовать схему системы связи, изображённую на рис. 2.1 так, как изображено на рис 3.3, разделив передатчик на **векторный передатчик** и **модулятор**, а приёмник на **демодулятор** и **векторный приёмник**, то, считая модулятор, канал связи и демодулятор в совокупности **векторным каналом связи**, получим систему связи с векторным каналом.

4. Оптимальный приём в векторном канале

Найдём теперь оптимальный алгоритм работы векторного приёмника (то есть, алгоритм работы решающей схемы на рис. 3.2). Искажающие свойства векторного канала полностью определяются плотностью условного распределения $f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s})$, которая при $\vec{s} = \vec{s}_i$ есть функция N переменных, и может быть записана как $f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s} = \vec{s}_i) = f_{\vec{r}|m}(\vec{r} \mid m = m_i)$. Отсюда по формуле Байеса (для условных плотностей) получаем величину

$$\Pr\{m = m_k \mid \vec{r}\} = \frac{\Pr\{m = m_k\} f_{\vec{r}|m_k}(\vec{r} \mid m = m_k)}{f_{\vec{r}}(\vec{r})}, \tag{15}$$

о которой пойдёт речь в следующей важной теореме:

Теорема (*об алгоритме оптимального приема в векторном канале)***:**
Оптимальный приемник в системе связи с векторным каналом для каждого принятого вектора $\vec{r}$ полагает, что $\hat{m} = m_k$ всегда, когда

$$\Pr\{m = m_k \mid \vec{r}\} > \Pr\{\hat{m} = m_i \mid \vec{r}\} \quad \forall\, i \neq k,\ i = 1,\ldots,N \tag{16}$$

Если максимум достигается при нескольких k, т.е. для некоторых k и j имеет место $\Pr\{m = m_k \mid \vec{r}\} = \Pr\{\hat{m} = m_j \mid \vec{r}\}$, то приемник может положить $\hat{m} = m_k$ или $\hat{m} = m_j$ произвольным образом, и это не повлияет на вероятность ошибки приёма.

Такой приемник называется **приемником максимальной апостериорной вероятности**.

Доказательство: Всякий раз, когда какой-либо приемник принимает вектор $\vec{r}$ и делает вывод, что $\hat{m} = m_k$ (то есть, что передавалось сообщение m_k), он делает правильный выбор с вероятностью $\Pr\{\hat{m} = m \mid \vec{r}\} = \Pr\{m = m_k \mid \vec{r}\}$. Полная вероятность правильного приема при этом есть

$$\Pr\{\hat{m} = m\} = \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} \Pr\{\hat{m} = m \mid \vec{r}\} f_{\vec{r}}(\vec{r}) d\vec{r} \tag{17}$$

Для приемника максимальной апостериорной вероятности, согласно алгоритму его работы

$$\Pr\{\hat{m} = m\} = \int_{-\infty}^{+\infty} \ldots \int_{-\infty}^{+\infty} \max_{1 \le i \le M} (\Pr\{m = m_i \mid \vec{r}\}) f_{\vec{r}}(\vec{r}) d\vec{r}\,. \tag{18}$$

Пусть теперь другой приемник при некоторых значениях $\vec{r}$ принимает другое решение. $f_{\vec{r}}(\vec{r})$ от этого не зависит, и поэтому подынтегральное выражение в (18) может только уменьшиться, а это не может увеличить значение интеграла, что и требовалось доказать. ■

Заметив, что $f_{\vec{r}}(\vec{r})$ не зависит от решающего правила, по которому работает оптимальный приемник, и, принимая во внимание взаимную однозначность m_i и $\vec{s}_i$, а также формулу (15), заключаем, что вместо максимума апостериорной вероятности оптимальный приёмник может пользоваться максимумом величины

$$D_k(\vec{r}) = \Pr\{m = m_k\} f_{\vec{r}|m}(\vec{r} \mid m = m_k) = \Pr\{m = m_k\} f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s} = \vec{s}_k) \tag{19}$$

Эту величину принято называть **решающей функцией** оптимального приемника.

В случае априорно равновероятных сообщений, т.е. когда все $\Pr\{m = m_i\}$ равны, в качестве решающей функции оптимального приёма можно использовать просто $f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s} = \vec{s}_i)$. Такой приём называют **приёмом по методу максимального правдоподобия.**

Проиллюстрируем работу оптимального приемника с помощью графических представлений. Пусть источник сообщений может порождать три возможных сообщения, вероятности которых есть $\Pr\{m_1\}$, $\Pr\{m_2\}$ и $\Pr\{m_3\}$, а векторное представление соответствующих им сигналов в некотором двумерном базисе есть $\vec{s}_1 = (1,2)$, $\vec{s}_2 = (2,1)$, $\vec{s}_3 = (1,-2)$. Изобразим их на координатной плоскости (рис. 4.1), каждая точка которой может быть интерпретирована как некоторый возможный вектор $\vec{r}$.

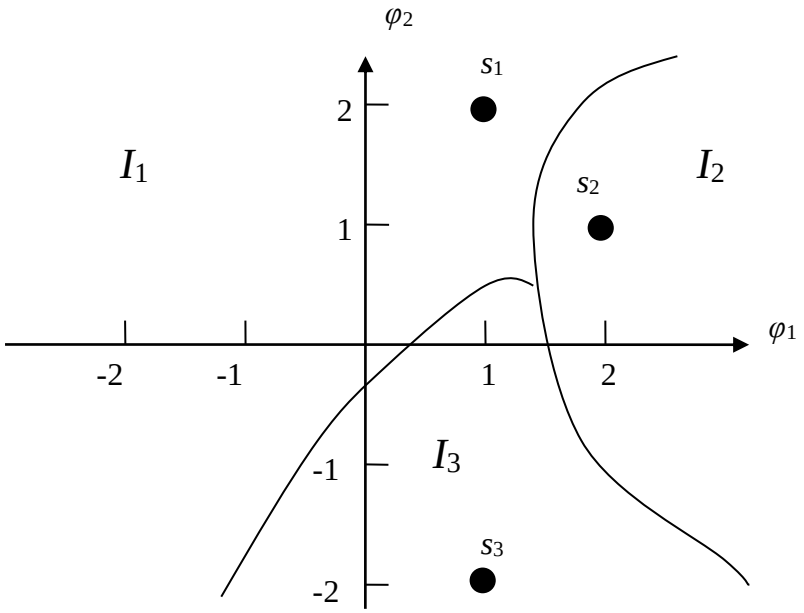

Рисунок 4.1. Графическое представление оптимального приёма в векторном канале

Для оптимального приемника алгоритм работы сводится к разделению всей плоскости на некоторые области I_i, i=1,2,3, такие, что каждая содержит только те точки $\vec{r}$, для которых $D_i > D_j \quad \forall\ j \neq i$. Поэтому оптимальный приемник принимает решение о том, что $\hat{m} = m_i$ тогда и только тогда, когда $\vec{r} \in I_i$ (точки, для которых D_i=D_j, образуют границу между I_i, и I_j,). Области I_i, называются **областями решений оптимального приёмника**. Оптимальный приемник совершает ошибку, когда из-за искажений в канале $\vec{r}$ оказывается в другой области, чем был переданный $\vec{s}_i$. Понятно, что подобное представление легко обобщается на случай произвольной размерности базиса и любого числа сигналов.

В соответствии с видом решающей функции D_i конкретный вид областей решения зависит, во-первых, от априорных вероятностей $\Pr\{m = m_k\}$, во-вторых, от расположения векторов сигналов $\vec{s}_i$ в пространстве сигналов, и, в-третьих, от свойств канала, описываемых условным распределением $f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s})$. В общем случае вычисление границ таких областей - весьма сложная задача, однако для некоторых частных случаев она упрощается. Рассмотрим один из них, важный для дальнейшего изложения.

5. Канал с аддитивным белым гауссовским шумом

Вернёмся к системе связи, изображённой на рис. 2.1. Предположим, что искажающее действие канала связи на передаваемый сигнал $s(t)$ сводится к прибавлению некоторого случайного сигнала $n(t)$, так что

$$r(t) = s(t) + n(t) \tag{20}$$

Сигнал $n(t)$ в таком случае принято называть **аддитивной помехой** или **аддитивным шумом**. Пусть передатчик, изображённый на рис. 3.1, передаёт в канал сигнал $s_i(t)$. После прохождения канала с аддитивным шумом сигнал поступает на вход приёмника, изображённого на рис. 3.2. На вход его решающей схемы с k-того интегратора, согласно (14), поступит

$$r_k = \int_{-\infty}^{+\infty} r(t)\phi_k(t)dt = \int_{-\infty}^{+\infty} s_i(t)\phi_k(t)dt + \int_{-\infty}^{+\infty} n(t)\phi_k(t)dt = s_{ik} + n_k \tag{21}$$

где

$$n_k = \int_{-\infty}^{+\infty} n(t)\phi_k(t)dt \tag{22}$$

то есть

$$\vec{r} = \vec{s}_i + \vec{n}\,. \tag{23}$$

Наличие аддитивного шума в канале связи привело к тому, что в полученном из него векторном канале появился векторный аддитивный шум – случайный вектор, прибавляющийся к вектору полезного сигнала. Далее будем считать, что $n(t)$ статистически независим от $s(t)$. Тогда и $\vec{n}$ статистически независим от $\vec{s}$, и в частности, $f_{\vec{n}}(\vec{n} \mid \vec{s}) = f_{\vec{n}}(\vec{n})$. Отсюда и из уравнения (23) следует, что

$$f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s} = \vec{s}_i) = f_{\vec{n}|\vec{s}}(\vec{n} = \vec{r} - \vec{s}_i \mid \vec{s} = \vec{s}_i) = f_{\vec{n}}(\vec{n} = \vec{r} - \vec{s}_i) \tag{24}$$

Решающая функция оптимального приёма, следовательно, равна

$$D_i(\vec{r}) = \Pr\{m = m_i\} f_{\vec{n}}(\vec{n} = \vec{r} - \vec{s}_i) \tag{25}$$

то есть для её вычисления необходимо знать $f_{\vec{n}}(\vec{n})$ – совместное распределение компонент вектора шума (которое, разумеется, зависит от $n(t)$).

Рассмотрим следующий частный случай: пусть $n(t)=n_W(t)$, где $n_W(t)$ – это стационарный в узком смысле гауссовский случайный процесс с нулевым средним и спектральной плотностью $S_W(f)=N_0/2$, $-\infty<f<\infty$. Такой шум называется **белым гауссовским шумом**.

Из свойств белого шума как случайного процесса отметим, что его корреляционная функция есть

$$R_W(\tau) = \mathrm{M}\big(n_W(t+\tau)n_W(t)\big) = \frac{N_0}{2}\delta(\tau) \tag{26}$$

где $\delta(t)$ – дельта-функция. Это следует из того, что

$$S_W(f) = \frac{N_0}{2} = \frac{N_0}{2}\int_{-\infty}^{+\infty} \delta(t)e^{-i2\pi f t}dt \tag{27}$$

и взаимной однозначности преобразования Фурье.

Заметим, что с физической точки зрения белый гауссовский шум невозможен, так как его общая средняя мощность была бы равна

$$\overline{n_W^2}(t) = \int_{-\infty}^{+\infty} |S_W(f)|^2 df = \infty$$

Полезность введения понятия белого шума заключается в том, что это удобное для анализа приближение к ситуации, когда спектр шума гораздо шире спектра сигнала и на

ширине спектра сигнала имеет приблизительно постоянное значение спектральной плотности.

Так как все значения $n_W(t)$ являются некоррелированными гауссовскими случайными величинами (это следует из вида корреляционной функции), то любая их взвешенная сумма или интеграл от произведения на детерминированную функцию есть тоже величина гауссовская. Поэтому все n_k, получающиеся при подстановке в (22) процесса $n_W(t)$, – это гауссовские случайные величины. Математическое ожидание любой из них есть:

$$\mathrm{M}(n_k) = \mathrm{M}(\int_{-\infty}^{+\infty} n_W(t)\phi_k(t)dt) = \int_{-\infty}^{+\infty} \mathrm{M}(n_W(t)\phi_k(t))dt = \int_{-\infty}^{+\infty} \phi_k(t)\mathrm{M}(n_W(t))dt = 0 \qquad (28)$$

Найдём корреляцию этих величин. Учитывая равенство нулю их математических ожиданий, имеем

$$R(n_k n_j) = \mathrm{M}(n_k n_j) = \mathrm{M}\left(\int_{-\infty}^{+\infty} n_W(u)\phi_k(u)du \int_{-\infty}^{+\infty} n_W(v)\phi_j(v)dv\right) = \int_{-\infty-\infty}^{+\infty+\infty} \mathrm{M}(n_W(u)n_W(v))\phi_k(u)\phi_j(v)dudv =$$

$$= \frac{N_0}{2}\int_{-\infty-\infty}^{+\infty+\infty} \delta(u-v)\phi_k(u)\phi_j(v)dudv = \frac{N_0}{2}\int_{-\infty}^{+\infty} \phi_k(v)\phi_j(v)dv = \frac{N_0}{2}\delta_{kj} = \begin{cases} N_0/2, & \text{если } k = j \\ 0, & \text{если } k \neq j \end{cases} \qquad (29)$$

где мы воспользовались видом корреляционной функции белого шума, **фильтрующим свойством** дельта-функции, состоящим в том, что

$$\int_{-\infty}^{+\infty} \delta(u-v)\phi_k(u)du = \phi_k(v) \qquad (30)$$

и тем обстоятельством, что $\{\varphi_i(t)\}_{i=1,\ldots,N}$ образуют ортонормальный базис.

Получилось, что n_k, k=1,…,N, - гауссовские некоррелированные, а значит и независимые случайные величины с нулевым средним и дисперсией $\sigma^2=N_0/2$. Следовательно, их совместная плотность распределения, то есть плотность распределения вектора шума $\vec{n} = (n_1,...,n_N)$, есть

$$f_{\vec{n}}(\vec{n}) = f_{\vec{n}}(\vec{n} = (n_1,...,n_N)) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp(-\frac{1}{2\sigma^2}\sum_{j=1}^{N} n_j^2) = \frac{1}{(\pi N_0)^{N/2}}\exp(-\frac{|\vec{n}|^2}{N_0}) \qquad (31)$$

где мы использовали тот факт, что в случае ортонормального базиса

$$|\vec{n}|^2 = (\vec{n},\vec{n}) = \sum_{j=1}^{N} n_j^2 \qquad (32)$$

Итак, наличие в исходном канале связи аддитивного белого гауссовского шума, спектральная плотность которого равна $N_0/2$, приводит к появлению в векторном канале аддитивного шума, представляющего собой гауссовский случайный вектор с независимыми одинаково распределёнными компонентами, имеющими нулевое математическое ожидание и дисперсию, равную $N_0/2$.

Подставив плотность распределения этого векторного шума в выражение (25) для решающей функции оптимального приёма, получим, что

$$D_k(\vec{r}) = \Pr\{m_k\}\frac{1}{(\pi N_0)^{N/2}}\exp\left(-\frac{|\vec{r}-\vec{s}_k|^2}{N_0}\right) \qquad (33)$$

Нетрудно видеть, что максимизация этого выражения по k равносильна минимизации по k выражения

$$|r - s_k|^2 - N_0 \ln(\Pr\{m_k\}) \qquad (34)$$

которому можно дать простое геометрическое толкование: если все m_k имеют одинаковые вероятности (то есть, приём ведётся по методу максимального правдоподобия), то оптимальное правило приема состоит в том, чтобы определить, какой $\vec{s}_i$ ближе всего к принятому из канала $\vec{r}$, и положить, что $\hat{m} = m_i$.

Легко в этом случае строятся границы решающих областей. Для равновероятных сигналов они проходят по серединным перпендикулярам к отрезкам, соединяющим две сигнальные точки. Так, для примера, рассматривавшегося ранее в общем случае на рис. 4.1, в случае векторного аддитивного гауссовского шума с независимыми компонентами, имеющими одинаковую дисперсию, решающие области имеют вид, изображённый на рис. 5.1 сплошными линиями. Если же априорные вероятности сообщений не равны (например $\Pr\{m_1\} > \Pr\{m_2\} > \Pr\{m_3\}$), то границы областей будут иметь вид, изображённый на рис. 5.1 штриховыми линиями.

После того, как области решений I_k определены, можно записать выражение для условной вероятности правильного приема при условии, что передавалось определенное сообщение:

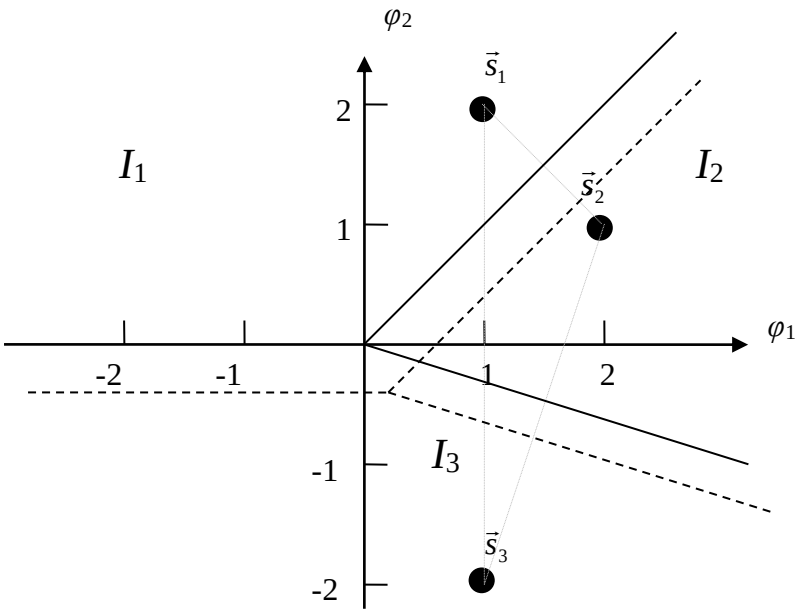

Рисунок 5.1. Решающие области оптимального приёма в канале с аддитивным белым гауссовским шумом.

$$\Pr\{\hat{m} = m \mid m = m_k\} = \Pr\{\vec{r} \in I_k \mid m = m_k\} = \int_{I_k} f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s} = \vec{s}_k)d\vec{r} \qquad (35)$$

Для случая аддитивного гауссовского шума с независимыми компонентами, имеющими одинаковую дисперсию, имеем

$$\Pr\{\hat{m} = m \mid m = m_k\} = \int_{I_k} \frac{1}{(\pi N_0)^{N/2}} \exp\left(-\frac{|\vec{r} - \vec{s}_k|^2}{N_0}\right)d\vec{r} \qquad (36)$$

Полная вероятность правильного приема

$$\Pr\{\hat{m} = m\} = \sum_{k=1}^{M} \Pr\{m_k\} \int_{I_k} \frac{1}{(\pi N_0)^{N/2}} \exp\left(-\frac{|\vec{r} - \vec{s}_k|^2}{N_0}\right)d\vec{r} \qquad (37)$$

В дальнейшем мы рассмотрим случаи, когда интегралы по решающим областям могут быть легко вычислены или аппроксимированы. А сейчас рассмотрим варианты построения решающей схемы оптимального приёмника.

Полученный выше алгоритм оптимального приёма позволяет построить несложную решающую схему для оптимального приёмника (рис. 5.2). Константы

$$c_i = \frac{1}{2}(N_0 \ln(\Pr\{m_i\}) - |\vec{s}_i|^2) \qquad (38)$$

могут быть вычислены заранее, так как не зависят от $\vec{r}$.

Нетрудно видеть, что эта схема производит следующие действия: если в некоторый момент времени на соответствующие входы начинают подаваться $r(t)$ и все $\phi_k(t)$, то схема находит все r_k (согласно (21)), потом для всех i=1,…,M находит $(\vec{r},\vec{s}_i)$ и, наконец, по максимуму выражения $(\vec{r},\vec{s}_i) + \frac{1}{2}(N_0 \ln(\Pr\{m_i\}) - |\vec{s}_i|^2)$ определяет $\hat{m}$. Этот алгоритм

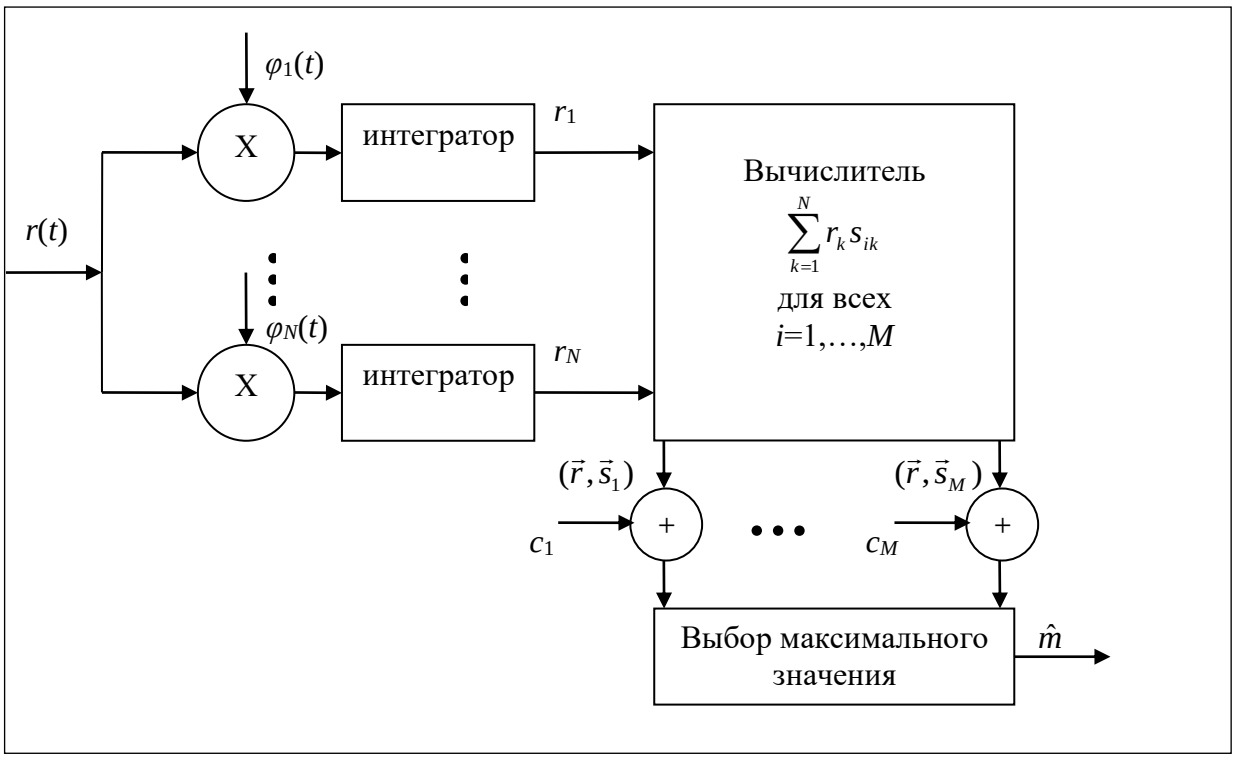

Рисунок 5.2. Принципиальная схема оптимального приёмника.

полностью равносилен алгоритму оптимального приёма, так как $|\vec{r}-\vec{s}_k|^2 - N_0 \ln(\Pr\{m_k\}) = |\vec{r}|^2 - 2(\vec{r},\vec{s}_k) + |\vec{s}_k|^2 - N_0 \ln(\Pr\{m_k\})$, где $|\vec{r}|^2$ одно и то же для всех m_i. Таким образом, можно минимизировать $-2(\vec{r},\vec{s}_k) + |\vec{s}_k|^2 - N_0 \ln(\Pr\{m_k\})$ или максимизировать $(\vec{r},\vec{s}_k) + \frac{1}{2}(N_0 \ln(\Pr\{m_k\} - |\vec{s}_k|^2))$ что наш приемник и делает.

Разберём ещё один способ построения оптимального приёмника для векторного канала с аддитивным гауссовским шумом (рис. 5.3).Здесь на выходе i-го интегратора имеем сразу $(\vec{r},\vec{s}_k)$. Поэтому решающая схема в приёмнике на рис. 5.3 проще, чем на рис. 5.2 - в ней отсутствует блок вычисления скалярного произведения. Однако это не означает, что сам такой приёмник проще. Дело в том, что в схеме на рис. 5.3 ветвей с перемножителями и интеграторами M штук, а в приёмнике на рис. 5.2 - только N штук, где $N \le M$, и на практике часто $M >> N$. Так что вопрос о том, какая схема лучше, должен решаться в каждом конкретном случае отдельно.

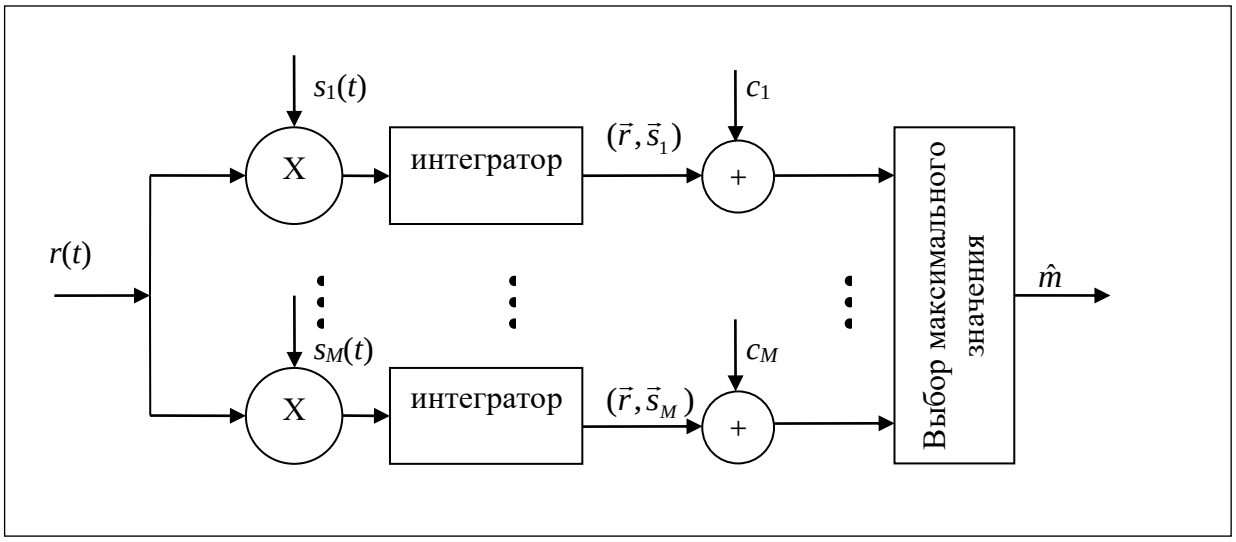

Рисунок 5.3. Второй вариант принципиальной схемы оптимального приёмника.

6. Мультивекторные каналы

Модель векторного канала с аддитивным гауссовским шумом оказалась весьма плодотворной. Нам удалось найти и процедуру оптимального приёма для такого канала, и составить принципиальные схемы устройств, её реализующих. Но остался невыясненным следующий очень важный вопрос: какое отношение имеет оптимальный приём в векторном канале к оптимальному приёму в исходном канале связи, не являющимся векторным? Ведь из того, что приёмник максимальной апостериорной вероятности является оптимальным для векторного канала, ещё не следует, что он же будет оптимальным для исходного канала. Может быть можно не преобразовывать исходный канал к векторному, а придумать что-то ещё, что даст лучшие результаты? Для того чтобы приступить к ответу на эти вопросы, нам понадобятся некоторые предварительные исследования, которые мы проведём в данном параграфе.

Итак, рассмотрим некоторое обобщение модели векторного канала. Пусть имеется система связи с несколькими одновременно действующими векторными каналами, по каждому из которых передаётся свой вектор, соответствующий передаваемому сообщению. Все эти векторы поступают в приёмник, и приёмник может принимать решение о том, какое именно сообщение передавалось на основании всех принятых векторов. Такую систему связи будем называть **системой связи с мультивекторным каналом**.

Пусть теперь способ получения передатчиком векторов, представляющих передаваемое сообщение в каждом из каналов, значения не имеет. Эти вектора могут быть наборами коэффициентов разложения непрерывного сигнала по различным базисам, наборами значений сигнала в какие-то моменты времени и т.д. - сейчас для нас это неважно. Вопрос, который нас интересует, состоит в следующем: какую именно информацию в системе связи с мультивекторным каналом необходимо учитывать, чтобы организовать оптимальный приём?

Для простоты будем считать, что векторных каналов в системе два, как на рис. 6.1, и векторы на выходе каждого из каналов обозначим $\vec{r}' = (r_1',...,r_K')$ и $\vec{r}'' = (r_1'',...,r_L'')$. Полный вход приемника естественно описывать вектором $\vec{r} = (\vec{r}',\vec{r}'')$. Алгоритм оптимального приема при этом запишется так: положить, что $\hat{m} = m_i$, если при k=i достигается максимум выражения

$$D_k(\vec{r}) = \Pr\{m = m_k\} f_{\vec{r}|\vec{s}}(\vec{r} \mid \vec{s} = \vec{s}_k) = \Pr\{m = m_k\} f_{\vec{r}|\vec{s}}(\vec{r}',\vec{r}'' \mid \vec{s} = \vec{s}_k) \qquad (39)$$

среди всех k=1,…,M.

Рассмотрим следующий вопрос: всегда ли знание обоих векторов $\vec{r}'$ и $\vec{r}''$ влияет на вероятность ошибки оптимального приема или есть ситуации, в которых оптимальный приемник может пренебречь, например, вектором $\vec{r}''$, не учитывая его в решающей функции? Условия, при которых это можно сделать, сформулируем в виде теоремы:

Теорема (*о несущественных данных*)**:** В системе связи с мультивекторным каналом оптимальный приемник может пренебречь вектором $\vec{r}''$ тогда и только тогда, когда

$$f_{\vec{r}''|\vec{r}',\vec{s}}(\vec{r}''\,|\,\vec{r}',\vec{s}) = f_{\vec{r}''|\vec{r}'}(\vec{r}''\,|\,\vec{r}'). \tag{40}$$

Доказательство: Итак, решающая функция имеет вид (39). По формуле для плотности условной вероятности имеем

$$f_{\vec{r}|\vec{s}}(\vec{r}',\vec{r}''\,|\,\vec{s}=\vec{s}_k) = f_{\vec{r}''|\vec{r}',\vec{s}}(\vec{r}''\,|\,\vec{r}',\vec{s}=\vec{s}_k)\, f_{\vec{r}'|\vec{s}}(\vec{r}'\,|\,\vec{s}=\vec{s}_k) = f_{\vec{r}''|\vec{r}'}(\vec{r}''\,|\,\vec{r}')\, f_{\vec{r}''|\vec{s}}(\vec{r}'\,|\,\vec{s}=\vec{s}_k) \tag{41}$$

Тогда решающая функция может быть записана как

$$D_k(\vec{r}) = \Pr\{m=m_k\}\, f_{\vec{r}''|\vec{r}'}(\vec{r}''\,|\,\vec{r}')\, f_{\vec{r}'|\vec{s}}(\vec{r}'\,|\,\vec{s}=\vec{s}_k) \tag{42}$$

где множитель $f_{\vec{r}''|\vec{r}'}(\vec{r}''\,|\,\vec{r}')$ не зависит от номера сообщения m_k. Он один и тот же для всех m_k, и при отыскании максимума решающей функции по k его можно не учитывать. Получается, что в решающей функции $\vec{r}''$ не присутствует, что и требовалось доказать. ■

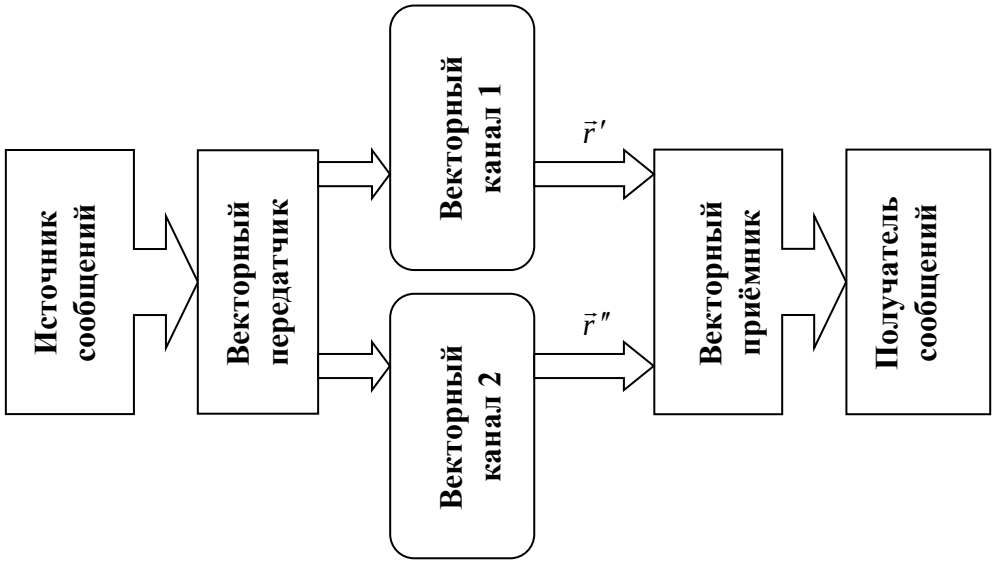

Рисунок 6.1. Модель системы связи с мультивекторным каналом.

Следствие (*достаточное условие несущественности данных*)**:**
Если для системы связи с мультивекторным каналом справедливо равенство

$$f_{\vec{r}''|\vec{r}',\vec{s}}(\vec{r}''\,|\,\vec{r}',\vec{s}) = f_{\vec{r}''}(\vec{r}'') \tag{43}$$

то $\vec{r}''$ - несущественные для оптимального приёма данные.

Проиллюстрируем смысл и применимость этой теоремы на следующих трех примерах:

1) Мультивекторный канал, изображённый на рис. 6.2а, в котором $\vec{n}'$ и $\vec{n}''$ независимы и не зависят от остальных векторов системы. Как нетрудно видеть из рисунка, здесь $\vec{r}' = \vec{s} + \vec{n}'$, $\vec{r}'' = \vec{n}''$. Для этого примера выполняется достаточное условие несущественности данных, а именно, поскольку $\vec{r}''$ не зависит ни от $\vec{s}$, ни от $\vec{n}'$, то $f_{\vec{r}''|\vec{r}',\vec{s}}(\vec{r}''\,|\,\vec{r}',\vec{s}) = f_{\vec{r}''}(\vec{r}'') = f_{\vec{n}''}(\vec{n}'')$, то есть вектором $\vec{r}''$ можно пренебречь.

2) Мультивекторный канал, изображённый на рис. 6.2б, в котором $\vec{n}'$ и $\vec{n}''$ независимы и не зависят от других векторов в системе. Из рисунка следует, что $\vec{r}' = \vec{s} + \vec{n}'$, $\vec{r}'' = \vec{r}' + \vec{n}''$. Здесь достаточное условие несущественности данных не выполняется, однако условия теоремы о несущественных данных выполняются.

Действительно, так как по условию

$$\vec{n}'' = \vec{r}'' - \vec{r}' \tag{44}$$

то если задано значение $\vec{r}'$, для любых значений вектора $\vec{r}''$ справедливо

$$f_{\vec{r}''|\vec{r}',\vec{s}}(\vec{r}''|\vec{r}',\vec{s}) = f_{\vec{n}''|\vec{r}',\vec{s}}(\vec{n}'' = \vec{r}'' - \vec{r}'|\vec{r}',\vec{s}). \tag{45}$$

Так как $\vec{n}''$ независим, в частности, от $\vec{s}$, то $\vec{s}$ из условия можно убрать (заметим, что $\vec{r}'$ из условия убирать нельзя, так как только при условии задания его конкретного значения мы можем записать равенство (45)). Таким образом, имеем

$$f_{\vec{n}''|\vec{r}',\vec{s}}(\vec{n}'' = \vec{r}'' - \vec{r}'|\vec{r}',\vec{s}) = f_{\vec{n}''|\vec{r}'}\{\vec{n}'' = \vec{r}'' - \vec{r}'|\vec{r}'\}$$

И ещё раз воспользуемся равенством (44), но теперь для обратного перехода от $\vec{n}''$ к $\vec{r}''$, который возможен опять-таки при условии задания конкретного значения $\vec{r}'$:

$$f_{\vec{n}''|\vec{r}'}(\vec{n}'' = \vec{r}'' - \vec{r}'|\vec{r}') = f_{\vec{r}''|\vec{r}'}\{\vec{r}''|\vec{r}'\}$$

Таким образом, здесь выполняется равенство (40), то есть для этого случая справедливы условия теоремы о несущественных данных, и оптимальный приёмник может пренебречь вектором $\vec{r}''$.

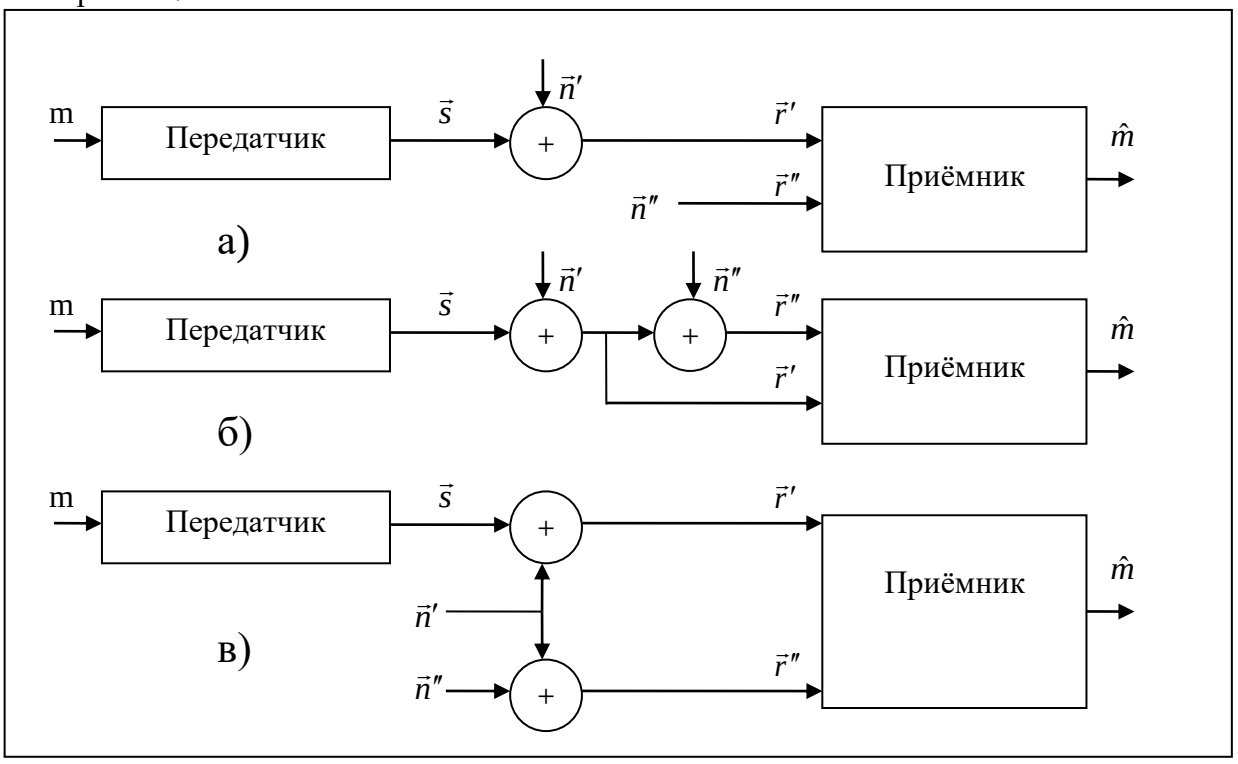

Рисунок 6.2. Примеры мультивекторных каналов

Этот результат понятен и из соображений здравого смысла: в рассматриваемой системе связи $\vec{r}''$ - это еще более искаженный $\vec{r}'$ и не может дать дополнительной информации об $\vec{s}$, поэтому им можно пренебречь.

3) Мультивекторный канал, изображённый на рис. 6.2в, в котором $\vec{n}'$ и $\vec{n}''$ независимы и не зависят от других векторов в системе. Из рисунка находим, что

$$\vec{r}' = \vec{s} + \vec{n}', \qquad \vec{r}'' = \vec{n}' + \vec{n}'' \tag{46}$$

Здесь оптимальный приемник не может игнорировать $\vec{r}''$, так как

$$f_{\vec{r}''|\vec{r}',\vec{s}}(\vec{r}''|\vec{r}',\vec{s}) = f_{\vec{n}''|\vec{n}'}\{\vec{n}'' = \vec{r}'' - \vec{r}' + \vec{s}\,|\,\vec{n}' = \vec{r}' - \vec{s}\} \tag{47}$$

то есть здесь есть явная зависимость от $\vec{s}$ и потому $\vec{r}''$ является существенным. Смысл этого понятен из того, что если по $\vec{r}''$ можно хорошо оценить $\vec{n}'$, то мы можем найти $\vec{s}$

точнее (например, если дисперсия $\vec{n}''$ много меньше дисперсии $\vec{n}'$, то с большой вероятностью $|\vec{n}'| >> |\vec{n}''|$, то есть $\vec{r}''$ близок к $\vec{n}'$ и является его хорошей оценкой).

7. Оптимальный приём в канале с аддитивным белым гауссовским шумом

Теперь мы готовы к доказательству теоремы о том, что при определённых условиях оптимальный приём в векторном канале является оптимальным и для исходного канала связи.

Рассмотрим канал связи с аддитивным белым гауссовским шумом. Как отмечалось ранее, в предположении о том, что передавался сигнал $\vec{s}_i$, после прохождения через демодулятор векторного приёмника получается вектор $\vec{r}$, для компонент которого r_k справедлива формула (21), в которой n_k задаются выражением (22). Так как $n_W(t)$ не зависит от $s(t)$, то и $\vec{n}$ не зависит от $\vec{s}$. Важно подчеркнуть, что в общем случае

$$n_W(t) \neq \sum_{j=1}^{N} n_j \phi_j(t) \tag{48}$$

и, следовательно, преобразуя $r(t)$ в векторном приемнике, мы потеряли часть информации об $r(t)$. Поэтому наша цель – доказать, что вся потерянная информация несущественна для оптимального приёма.

Теорема: Оптимальный приём в канале связи с аддитивным белым гауссовским шумом может быть осуществлён как оптимальный приём в векторном канале.

Доказательство: Пусть

$$r_1(t) = \sum_{j=1}^{N} r_j \phi_j(t) = s(t) + n^*(t), \quad \text{где} \quad n^*(t) = \sum_{j=1}^{N} n_j \phi_j(t) \tag{49}$$

Пусть также

$$r_2(t) = r(t) - r_1(t) = s(t) + n_W(t) - s(t) - n^*(t) = n_W(t) - n^*(t) \tag{50}$$

(используя равенства (20) и (49)). Сигнал $r_2(t)$ не зависит от передаваемого сигнала $s(t)$ и содержит всю информацию, которую мы потеряли при векторном приеме, поэтому нам надо показать, что любые данные, получаемые из $r_2(t)$, несущественны для оптимального приема.

Пусть $\vec{r}_2 = (r_2(t_1), r_2(t_2), ..., r_2(t_m))$ - любая конечная совокупность временных отсчетов сигнала $r_2(t)$. Очевидно, что $\vec{r}_2$, как и $\vec{n}$, зависит только от $n_w(t)$ и не зависит от $\vec{s}$. Из (49) имеем $\vec{r}_1 = \vec{s} + \vec{n}$. Заметим, что

$$f_{\vec{r}_2|\vec{r}_1,\vec{s}}(\vec{r}_2 \mid \vec{r}_1, \vec{s}) = f_{\vec{r}_2|\vec{r}_1,\vec{s}}(\vec{r}_2 \mid \vec{n}, \vec{s}) = \frac{f_{\vec{r}_2,\vec{n},\vec{s}}(\vec{r}_2, \vec{n}, \vec{s})}{f_{\vec{n},\vec{s}}(\vec{n}, \vec{s})} = \frac{f_{\vec{r}_2,\vec{n}}(\vec{r}_2, \vec{n}) f_{\vec{s}}(\vec{s})}{f_{\vec{n}}(\vec{n}) f_{\vec{s}}(\vec{s})} = f_{\vec{r}_2|\vec{n}}(\vec{r}_2 \mid \vec{n}) \tag{51}$$

По теореме о несущественных данных оптимальный приемник может игнорировать $\vec{r}_2$, если $\vec{r}_2$ не будет зависеть и от $\vec{n}$. Так как $\vec{r}_2$ - это произвольный набор значений $r_2(t)$, то мы должны доказать независимость случайных процессов $r_2(t)$ и $n^*(t)$. Тогда можно игнорировать весь $r_2(t)$.

Для доказательства этого факта прежде всего заметим, что процессы $r_2(t)$ и $n^*(t)$ являются результатами линейных операций (сложения, вычитания и интегрирования) над гауссовским процессом $n_w(t)$. Поэтому процессы $r_2(t)$ и $n^*(t)$ являются совместно гауссовскими процессами, то есть их любое совместное распределение - гауссовское.

По свойству гауссовских процессов они независимы тогда и только тогда, когда некоррелированы, то есть когда их взаимная корреляционная функция равна нулю:

$$\mathrm{M}(n^*(s) r_2(t)) - \mathrm{M}(n^*(s)) \mathrm{M}(r_2(t)) = 0 \quad \forall t, s \tag{52}$$

Так как $n_w(t)$ - процесс с нулевым средним, то $r_2(t)$ и $n^*(t)$ - тоже, поэтому нам достаточно показать, что

$$\mathrm{M}(n^*(s) r_2(t)) = 0 \quad \forall t, s \tag{53}$$

Пользуясь определением $n^*(t)$, имеем

$$\mathrm{M}(n^*(s)r_2(t)) = \mathrm{M}(r_2(t)\sum_{j=1}^{N} n_j\phi_j(s)) = \sum_{j=1}^{N}\phi_j(s)\mathrm{M}(n_j r_2(t)). \quad (54)$$

Учитывая, что функции базиса $\phi_j(t)$ линейно независимы, эта сумма может равняться нулю, только если

$$\mathrm{M}(n_j r_2(t)) = 0 \quad \forall j,t \quad (55)$$

Из определения $r_2(t)$ имеем:

$$\mathrm{M}(n_j r_2(t)) = \mathrm{M}(n_j(n_W(t) - n^*(t))) = \mathrm{M}(n_j n_W(t) - n_j n^*(t)) = \mathrm{M}(\int_{-\infty}^{+\infty} n_W(t)n_W(s)\phi_j(s)ds) -$$

$$-\mathrm{M}(n_j\sum_{k=1}^{N} n_k\phi_k(t)) = \int_{-\infty}^{+\infty}\mathrm{M}(n_W(s)n_W(t))\phi_j(s)ds - \sum_{k=1}^{N}\mathrm{M}(n_j n_k)\phi_k(t) \quad (56)$$

Вычислим первый член этого выражения:

$$\int_{-\infty}^{+\infty}\mathrm{M}(n_W(t)n_W(s))\phi_j(s)ds = \int_{-\infty}^{+\infty}R_W(t-s)\phi_j(s)ds = \frac{N_0}{2}\int_{-\infty}^{+\infty}\delta(t-s)\phi_j(s)ds = \frac{N_0}{2}\phi_j(t) \quad (57)$$

согласно свойству дельта - функции (здесь мы воспользовались видом корреляционной функции белого шума).

А для второго члена значения $\mathrm{M}(n_j n_k)$ мы уже вычисляли - см. (29), откуда:

$$\sum_{k=1}^{N}\mathrm{M}(n_j n_k)\phi_k(t) = \sum_{k=1}^{T}\frac{N_0}{2}\delta_{jk}\phi_k(t) = \frac{N_0}{2}\phi_j(t) \quad (58)$$

Подставив (57) и (58) в (56), получим (55).

Таким образом, $r_2(t)$ и $n^*(t)$ независимы, поэтому $\vec{r}_2$ и $\vec{n}$ независимы и, следовательно, по достаточному условию несущественности данных $\vec{r}_2$ можно игнорировать при оптимальном приеме, что и требовалось доказать. ■

Итак, теперь у нас есть алгоритм оптимального приёма для канала связи с аддитивным белым гауссовским шумом. Этот результат очень важен не только с теоретической, но и с практической точки зрения, так как многие реальные каналы связи могут быть с хорошей степенью точности описаны либо математической моделью канала с аддитивным белым гауссовским шумом, либо близкими к ней по свойствам, но более сложными математическими моделями. Это относится к значительной части проводных и оптоволоконных каналов связи, а также некоторым радиоканалам.

8. Вероятность ошибки оптимального приёма для системы из двух сигналов

Теперь перейдём к вычислению вероятности ошибки оптимального приёма в канале с аддитивным белым гауссовским шумом. Это важная задача, так как для практики нам надо знать не только то, что мы используем наилучший алгоритм приёма, но и то, какой результат от его использования мы получим в каждой конкретной ситуации.

Отметим одно важное свойство, которое мы получили при рассмотрении каналов с аддитивным белым гауссовским шумом: мы нигде не использовали ни конкретный вид сигналов $s_i(t)$, ни конкретный вид ортонормального базиса $\{\varphi_i(t)\}_{i=1,\ldots,N}$. в пространстве этих сигналов. Это говорит о том, что для характеристик оптимального приема (таких, например, как вероятность ошибки) важны только взаимное расположение векторов $\vec{s}_i$, представленных в каком-либо ортонормальном базисе, и дисперсия шума $N_0/2$, а вид самих сигналов не важен.

Для практики это очень важное обстоятельство. Оно позволяет при сохранении заданной помехоустойчивости за счёт выбора формы сигналов одновременно решать и другие задачи, например, обеспечить простоту генерации сигналов. Часто поступают так:

задаются некоторым базисом пространства сигналов, который легок в реализации, а потом уже в этом базисе выбирают сами сигналы.

При дальнейшем анализе различных систем сигналов мы будем, как правило, иллюстрировать наше рассмотрение с помощью сигналов из пространства самого простого базиса - прямоугольных импульсов. Но не надо забывать, что базис может быть любым - от этого зависит только форма сигналов, но не свойства системы сигналов относительно помехоустойчивости оптимального приёма.

В общем случае вероятность ошибки оптимального приёма в канале с аддитивным белым гауссовским шумом дают формулы (36) и (37). Заметим, что подинтегральные выражения в них зависят только от модуля разности $|\vec{r}-\vec{s}_k|$, но не от самих векторов $\vec{r}$ и $\vec{s}_k$, а это означает, что значение интеграла не изменяется при преобразованиях системы координат, не изменяющих значение $|\vec{r}-\vec{s}_k|$, а именно при сдвиге, повороте и инверсии осей. При вычислении вероятности ошибки мы будем активно пользоваться этими свойствами, так как преобразования системы координат могут значительно упростить нахождение соответствующих интегралов.

Простейший случай - когда система состоит из двух сигналов. При этом удобно преобразовать базис так, чтобы $\vec{s}_1-\vec{s}_2$ лежал вдоль оси $\varphi_1(t)$, а сами сигналы были противоположными (рис. 8.1).

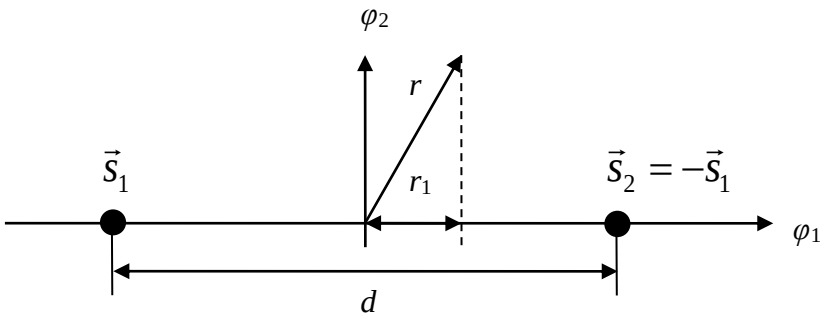

Рисунок 8.1. Система из двух сигналов

Решающие области оптимального приёма в этом случае определяются выражением:

$$\min_{k=1,2} |\vec{r}-\vec{s}_k|^2 - N_0 \ln(\Pr\{m_k\}).$$

При равных априорных вероятностях $\Pr\{m_1\}=\Pr\{m_2\}$ можно рассматривать $\min_{k=1,2} |\vec{r}-\vec{s}_k|^2$. В этом случае граница областей решения есть прямая, равноудаленная от $\vec{s}_1$ и $\vec{s}_2$, то есть ось φ_2. Нетрудно видеть, что в этом случае ошибки приема возникают тогда и только тогда, когда координата n_1 вектора шума $\vec{n}=(n_1,n_2)$ такова, что превышает по модулю $d/2$ (где d - расстояние между сигналами), и смещает $\vec{r}$ через ось φ_2 в другую решающую область. Так, для первого сигнала на рис. 8.1, имеем:

$$\Pr\{\hat{m}\neq m \mid m=m_1\} = \Pr\{\vec{r}\in I_2 \mid m=m_1\} = \Pr\{n_1 > d/2\} \qquad (59)$$

где

$$d^2 = |\vec{s}_1-\vec{s}_2|^2 = \int_{-\infty}^{+\infty} \left(s_1(t)-s_2(t)\right)^2 dt \qquad (60)$$

Так как n_1 является гауссовской величиной с нулевым средним и дисперсией $N_0/2$, то

$$\Pr\{n_1 > d/2\} = \int_{d/2}^{+\infty} \frac{1}{\sqrt{\pi N_0}} \exp\left(-\frac{x^2}{N_0}\right) dx \tag{61}$$

Введём определение:

$$Q(y) = \int_{y}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx \tag{62}$$

Функция $Q(y)$ называется **функцией ошибок**.

Далее используем подстановку $\gamma = x\sqrt{2/N_0}$, получим

$$\Pr\{\hat{m} \neq m \mid m = m_1\} = \int_{\frac{d/2}{\sqrt{N_0/2}}}^{+\infty} \frac{1}{\sqrt{2\pi}} \exp(-\frac{\gamma^2}{2}) d\gamma = Q\left(\frac{d}{\sqrt{2N_0}}\right). \tag{63}$$

И для безусловной вероятности ошибки

$$\Pr\{\hat{m} \neq m\} = \sum_{k=1}^{2} \Pr\{m_k\} \Pr\{\hat{m} \neq m \mid m = m_k\} = Q\left(\frac{d}{\sqrt{2N_0}}\right) \tag{64}$$

Это есть вероятность ошибки оптимального приема для любой пары равновероятных сигналов, удаленных друг от друга на расстояние *d*, независимо от их положения в пространстве сигналов.

Если это сигналы противоположные, то $|\vec{s}|_1 = |\vec{s}_2| = \sqrt{E_s}$, и $d = 2\sqrt{E_s}$, откуда

$$\Pr\{\hat{m} \neq m\} = Q\left(\sqrt{\frac{2E_s}{N_0}}\right) \tag{65}$$

Для двух ортогональных сигналов с амплитудой $\sqrt{E_s}$ расстояние между ними $d = \sqrt{2E_s}$ и

$$\Pr\{\hat{m} \neq m\} = Q\left(\sqrt{\frac{E_s}{N_0}}\right) \tag{66}$$

Так как $Q(x)$ - функция убывающая, то $Q\left(\sqrt{\frac{2E_s}{N_0}}\right) < Q\left(\sqrt{\frac{E_s}{N_0}}\right)$, то есть при той же энергии сигналов система связи с противоположными сигналами обеспечивает лучшую помехоустойчивость, чем система с ортогональными сигналами. Расчеты показывают, что при больших отношениях сигнал/шум разница примерно в 2 раза.

В случае, когда априорные вероятности сообщений m_1 и m_2 не равны ($\Pr\{m_1\} > \Pr\{m_2\}$), граница решающих областей выводится из уравнения

$$|\vec{r} - \vec{s}_1|^2 - N_0 \ln(\Pr\{m_1\}) = |\vec{r} - \vec{s}_2|^2 - N_0 \ln(\Pr\{m_2\}) \tag{67}$$

то есть (рис. 8.1):

$$(r_1 + \frac{d}{2})^2 - N_0 \ln(\Pr\{m_1\}) = (r_1 - \frac{d}{2})^2 - N_0 \ln(\Pr\{m_2\})$$

Откуда

$$r_1 d - N_0 \ln(\Pr\{m_1\}) = -r_1 d - N_0 \ln(\Pr\{m_2\})$$

и

$$r_1 = \frac{N_0}{2d} \ln\left(\frac{\Pr\{m_1\}}{\Pr\{m_2\}}\right) \tag{68}$$

Тогда, производя в интегралах по I_1 и I_2 подстановку $\gamma = x\sqrt{2/N_0}$, для вероятности ошибки несложно получить

$$\Pr\{\hat{m}\neq m\}=\sum_{k=1}^{2}\Pr\{m_k\}\Pr\{\hat{m}\neq m\,|\,m=m_k\}=\Pr\{m_1\}Q\left(\frac{d+2r_1}{\sqrt{2N_0}}\right)+\Pr\{m_2\}Q\left(\frac{d-2r_1}{\sqrt{2N_0}}\right). \quad (69)$$

9. Прямоугольные системы сигналов

Перейдем к разбору случаев M>2. **Прямоугольными системами сигналов** будем называть системы сигналов, у которых все сигналы расположены в вершинах прямоугольников (параллелограммов) (например, рис. 9.1).

Если при этом сигналы равновероятны, то все области решений имеют прямоугольные границы. Поворотом системы координат можно сделать все границы параллельными осям и тогда многомерное интегрирование по областям решений сведется к перемножению одномерных интегралов. Для правильной расстановки пределов интегрирования удобно сдвигать начало координат в соответствующую сигнальную точку. Для примера применим этот подход для системы, изображенной на рис. 9.1. Несложно получить, что

$$\Pr\{\hat{m}=m\,|\,m=m_1\}=\int\limits_{\vec{r}\in I_1}f_{\vec{r}|m_1}(\vec{r}\,|\,m=m_1)d\vec{r}=\int\limits_{-\infty}^{d/2}f_{n_i}(n_1)dn_1\int\limits_{-d/2}^{+\infty}f_{n_i}(n_2)dn_2=\left(1-Q(\frac{d}{\sqrt{2N_0}})\right)^2 \quad (70)$$

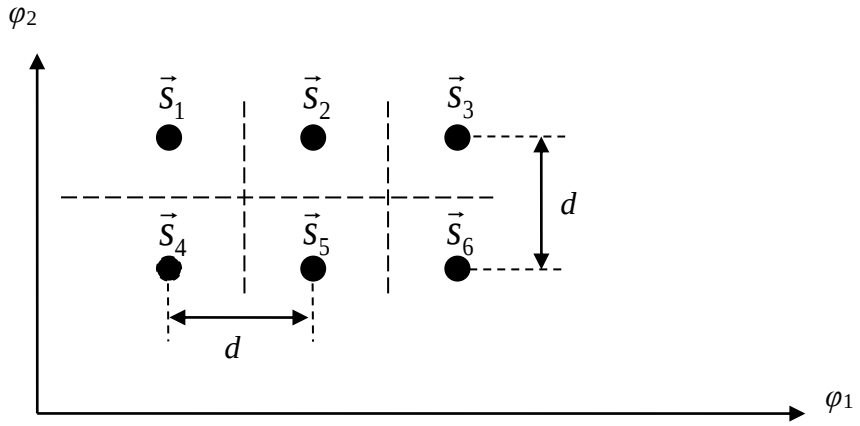

Рисунок 9.1. Пример прямоугольной системы сигналов.

Из соображений симметрии

$$\Pr\{\hat{m}=m\,|\,m=m_1\}=\Pr\{\hat{m}=m\,|\,m=m_3\}=\Pr\{\hat{m}=m\,|\,m=m_4\}=\Pr\{\hat{m}=m\,|\,m=m_6\}$$

Из тех же соображений

$$\Pr\{\hat{m}=m\,|\,m=m_2\}=\Pr\{\hat{m}=m\,|\,m=m_5\}=\int\limits_{-d/2}^{d/2}f_{n_i}(n_1)dn_1\int\limits_{-d/2}^{+\infty}f_{n_i}(n_2)dn_2=\left(1-2Q(\frac{d}{\sqrt{2N_0}})\right)\left(1-Q(\frac{d}{\sqrt{2N_0}})\right) \quad (71)$$

Обозначив $p=Q\left(\frac{d}{\sqrt{2N_0}}\right)$, получим

$$\Pr\{\hat{m}=m\}=\frac{4}{6}(1-p)^2+\frac{2}{6}(1-2p)(1-p)=(1-p)(1-\frac{4}{3}p) \quad (72)$$

Разберем один важный частный случай прямоугольной системы сигналов, когда векторы сигналов являются вершинами гиперкуба. Будем считать, что центр гиперкуба совпадает с началом координат (для N=2 такая система изображена на рис. 9.2). Будем называть такую систему сигналов **системой сигналов в вершинах гиперкуба**. При этом

сигналов $M=2^N$. Если все эти сигналы равновероятны, то решающие области совпадают с координатными квадрантами, и многомерный интеграл по какой – либо области решения есть произведение N одномерных интегралов:

$$\Pr\{\hat{m}=m \mid m=m_i\}=\int\ldots\int_{I_i} f_{\vec{r}|m_i}(\vec{r}\mid m=m_i)d\vec{r}=\left(\int_{-\infty}^{d/2} f_{n_i}(n_1)dn_1\right)^N=\left(1-Q(\frac{d}{\sqrt{2N_0}})\right)^N \qquad (73)$$

из соображений симметрии это верно для всех сигналов).
Из равновероятности сигналов имеем

$$\Pr\{\hat{m}=m\}=\left(1-Q(\frac{d}{\sqrt{2N_0}})\right)^N=(1-p)^N \qquad (74)$$

Выразим p через энергию сигнала**:**

$$|\vec{s}_i|^2=E_s=N\left(\frac{d}{2}\right)^2, \text{ то есть } d=2\sqrt{\frac{E_s}{N}}, \text{ откуда } p=Q\left(\sqrt{\frac{2E_s}{NN_0}}\right) \qquad (75)$$

Заметим, что выражение (74) можно получить и из следующих рассуждений: в данной системе по каждой из N координат мы имеем два возможных сигнала - на расстояниях $d/2$ и $-d/2$ от начала координат. Так что в каждом измерении $\Pr\{\hat{m}=m\}=1-p$. Чтобы не сделать ошибки, мы должны правильно принять все N координат, то есть $\Pr\{\hat{m}=m\}=(1-p)^N$, так как шум и, соответственно, прием по каждой координате независимы.

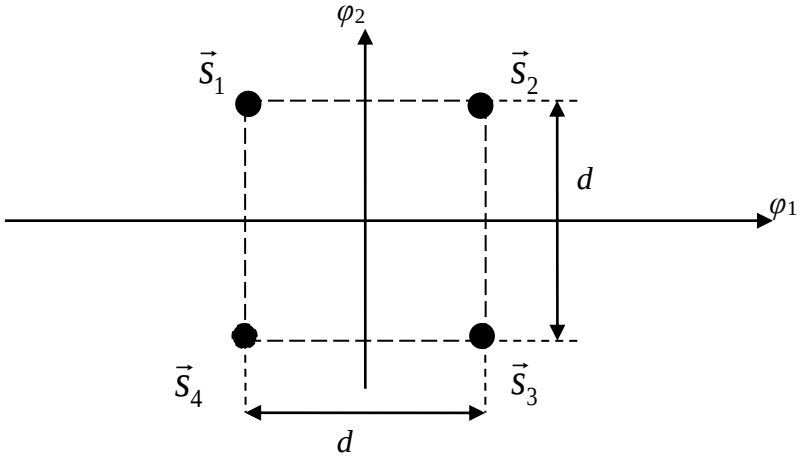

Рисунок 9.2. Двумерный случай системы сигналов в вершинах гиперкуба

10. Система ортогональных сигналов

Другим классом сигналов, для которых можно получить относительно простые формулы для минимально достижимой вероятности ошибки, является система M равновероятных ортогональных векторов равной энергии. При этом, очевидно, $M=N$ и

$$\int_{-\infty}^{+\infty} s_i(t)s_k(t)dt=(\vec{s}_i,\vec{s}_k)=E_s\delta_{ik}, \quad i,k=1,\ldots,M \qquad (76)$$

Для анализа системы ортогональных сигналов удобно с помощью сдвига и поворота осей сделать так, что $\vec{s}_i=\sqrt{E_s}\vec{\phi}_i$, где $\vec{\phi}_i$ - единичный вектор вдоль i-й координатной оси. При этом

$$|\vec{r}-\vec{s}_i|^2=|\vec{r}|^2+|\vec{s}_i|^2-2(\vec{r},\vec{\phi}_i)\sqrt{E_s}=|\vec{r}|^2+E_s-2r_i\sqrt{E_s}$$

где r_i - i-я составляющая вектора $\vec{r}$. Тогда нетрудно видеть, что в этом случае неравенство $|\vec{r}-\vec{s}_k|^2 < |\vec{r}-\vec{s}_i|^2$ равносильно неравенству $r_k > r_i$. То есть, если передавался k-тый сигнал, то оптимальный приёмник не сделает ошибки, если компонента r_k будет больше, чем другие компоненты $\vec{r}$. Поэтому можно записать, что

$$\Pr\{\hat{m}=m \mid m=m_1, r_1=\alpha\} = \Pr\{n_2<\alpha,...,n_M<\alpha\} = \left(\Pr\{n_2<\alpha\}\right)^{M-1}, \quad (77)$$

где учтено, что $\vec{s}_1=(\sqrt{E_s},0,...0)$, откуда $\vec{r}=(\alpha,n_2,...n_M)$, а также учтено, что n_i все независимы и имеют одно и то же распределение. Принимая во внимание, что

$$f_{r_1}(\alpha) = f_{n_1}(\alpha-\sqrt{E_s}) \quad (78)$$

и интегрируя по n_1, имеем

$$\Pr\{\hat{m}=m \mid m=m_1\} = \int_{-\infty}^{+\infty} f_{n_1}(\alpha-\sqrt{E_s})\left(\int_{-\infty}^{\alpha} f_{n_i}(\beta)d\beta\right)^{M-1} d\alpha \quad (79)$$

где под интегралами везде стоит гауссовское распределение $f_{n_i}(x)=\frac{1}{\sqrt{\pi N_0}}\exp\left(-\frac{x^2}{N_0}\right)$.

Вследствие симметрии, условные вероятности правильного приёма все одинаковые, так что правая часть (79) – это и безусловная вероятность правильного приема.

Итак, мы получили точную формулу вероятности правильного оптимального приёма для системы ортогональных сигналов, но из-за её сложности, на практике она мало полезна. Далее мы представим более простой метод оценки вероятности ошибки, применимый в том числе и к данной ситуации.

11. Аддитивная граница для вероятности ошибки

Данный метод оценки основан на том, что в ситуации равновероятных сигналов ошибка происходит тогда, когда при передаче $\vec{s}_k$, на приеме вектор $\vec{r}$ ближе к $\vec{s}_i$, $i\neq k$, чем к $\vec{s}_k$. Если обозначить как A_i событие, когда при передаче $\vec{s}_k$ принятый $\vec{r}$ ближе к $\vec{s}_i$, чем к $\vec{s}_k$, то можно записать, что

$$\Pr\{\hat{m}\neq m \mid m=m_k\} = \Pr\{A_1\cup A_2...A_{k-1}\cup A_{k+1}...\cup A_M\} \le \sum_{\substack{i=1\\ i\neq k}}^{M}\Pr\{A_i\} \quad (80)$$

Нетрудно видеть, что $\Pr\{A_i\}$ - это вероятность ошибки для системы из двух равновероятных сигналов $\vec{s}_k$ и $\vec{s}_i$, а остальные сигналы никак на нее не влияют (заметим также, что $\Pr\{A_i\}\neq\Pr\{\hat{m}=m_i \mid m=m_k\}$). Для каналов с белым гауссовским шумом

$$\Pr\{A_i\} = Q\left(\frac{|\vec{s}_i-\vec{s}_k|}{\sqrt{2N_0}}\right) \quad (81)$$

Для полностью симметричных систем сигналов $\Pr\{\hat{m}\neq m\}=\Pr\{\hat{m}\neq m \mid m=m_k\}$, и кроме того, большинство слагаемых в сумме в правой части (80) одинаковы, так что выражение еще упрощается. Так, для ортогональной системы имеем

$$\Pr\{\hat{m}\neq m\} \le (M-1)Q\left(\sqrt{\frac{E_s}{N_0}}\right) \quad (82)$$

Во многих случаях аддитивная граница есть полезное приближение к точному значению. Расчеты показывают, что при фиксированном M точность границы растет с ростом отношения E_s/N_0.

12. Примеры систем сигналов, используемых на практике

В этом параграфе мы рассмотрим приложения тех теоретических выводов, которые мы получили ранее. Мы рассмотрим несколько использующихся на практике систем сигналов, и обсудим их достоинства и недостатки.

Независимость помехоустойчивости оптимального приёма от формы сигналов для ситуаций, близких к каналу с аддитивным белым гауссовским шумом, даёт возможность выбором формы сигнала учитывать и другие практические требования к системам сигналов, кроме потенциальной помехоустойчивости. К таким требованиям относятся**:**

1) Простота генерации сигналов.
2) Согласование спектра сигналов с полосой частот канала.
3) Простота реализации оптимального или близкого к оптимальному приёма.

Первый пункт достаточно очевиден - чем проще сигнал генерируется, тем проще, дешевле и надёжней будет устройство передатчика.

Что касается пункта второго, то для узкополосных сигналов он сводится к тому, чтобы спектр сигнала был сосредоточен вблизи центральной частоты канала, и потому соответствующие сигналы строятся на основе синусоиды этой частоты. Для широкополосных каналов этот пункт сводится к требованию, чтобы спектр сигнала не выходил как за верхнюю, так и за нижнюю границы полосы частот канала.

Если форма сигнала выбрана, то верхняя граница полосы частот широкополосного канала накладывает ограничение на степень сжатия этой формы, так как при сжатии сигнала расширяется его спектр (см. свойства преобразования Фурье). Что касается нижней границы полосы частот широкополосного канала, то условие согласования с ним спектра используемой системы сигналов принято формулировать как **условие отсутствия в суммарном сигнале постоянной составляющей**. Дело в том что каналы связи на большие расстояния, например, длинные кабельные или проводные линии, весьма плохо пропускают самые низкие частоты, в том числе и "нулевую частоту" - постоянную составляющую. И чем длиннее линия связи, тем хуже она пропускает низкие частоты, тем важнее требование отсутствия постоянной составляющей.

Поясним пункт третий. Как мы видели, построение оптимального приёмника требует использования таких устройств, как перемножители и интеграторы (или эквивалентные им фильтры). Это достаточно сложные радиотехнические устройства. Поэтому на практике вместо оптимального приёмника часто применяют более простые схемы, использующие особенности конкретных систем сигналов и несущественно уступающие по надёжности приёма оптимальной. Так, например, если в качестве системы сигналов используется система на основе прямоугольных импульсов (положительной и отрицательной полярности), то близкий к оптимальному приём можно обеспечить, определяя уровень сигнала в середине каждого импульса, и сравнивая эту величину с нулём. Такие упрощённые методы приёма сигналов различной формы существуют и для других форм сигналов, но для всех них необходимо соблюдение одного очень важного требования, которое является необходимым и для оптимального приёма тоже - наличие **синхронизации** между приёмником и передатчиком.

Действительно, когда мы разбирали принципы построения оптимального приёмника, то в самом начале обсуждения неявно сделали предположение о том, что приёмник точно знает тот момент, когда на его вход начинает поступать сигнал из канала связи (см. параграф 5). На самом деле, организация синхронизации между приёмником и передатчиком - сложная техническая проблема, один из простых, но действенных практических подходов к которой – использовать такие сигналы, которые кроме переноса информации, помогали бы поддерживать синхронизацию. Такое свойство принято называть **самосинхронизацией сигнала**.

Рассмотрим несколько практических примеров и разберём их достоинства и недостатки с точки зрения введённых нами критериев. Сначала рассмотрим коды для широкополосных каналов связи.

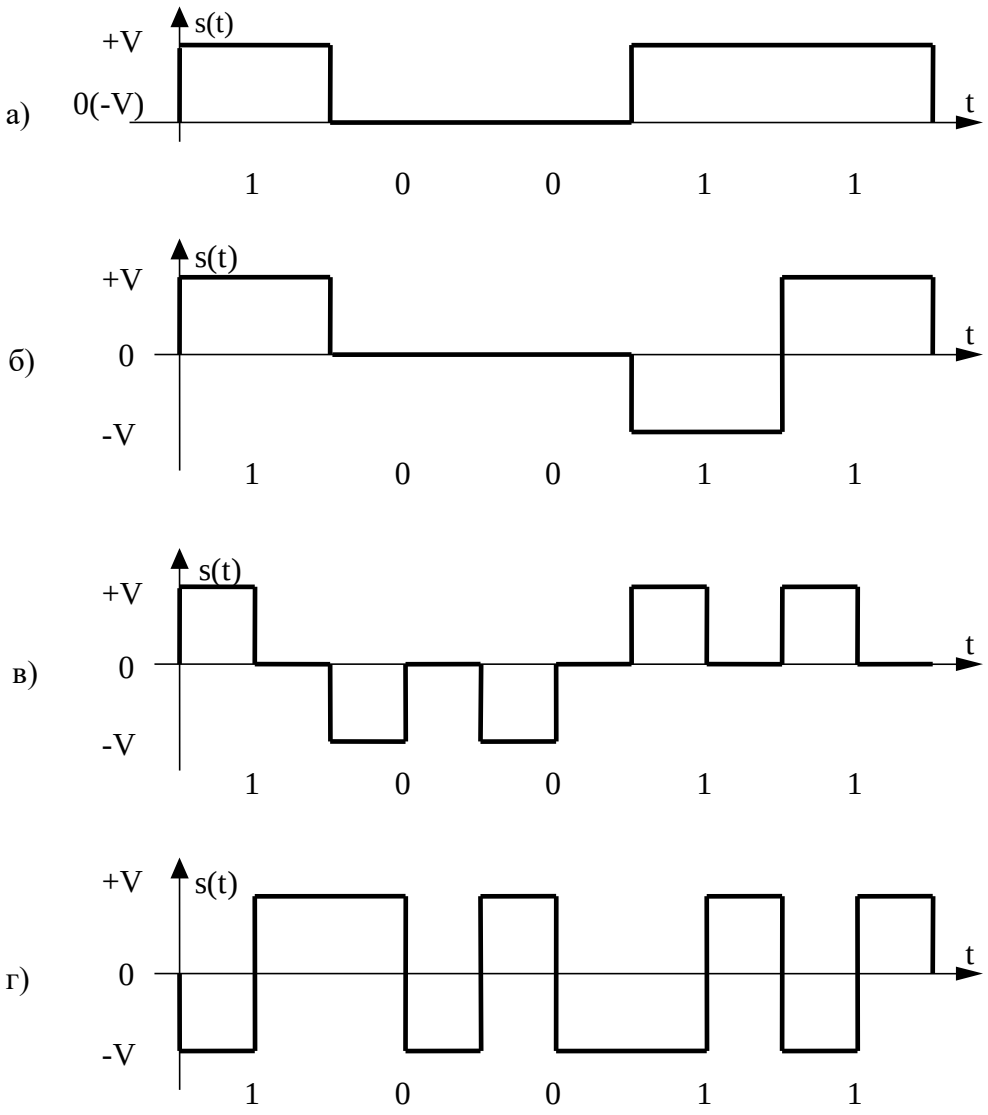

Рисунок 12.1. Примеры кодов для широкополосных каналов

1) **Код NRZ** (без возвращения к нулю), пример кодирования этим кодом изображён на рис. 12.1, а. Это не что иное, как уже хорошо нам знакомые прямоугольные импульсы.

Достоинства: очень прост в генерации; хорошо использует полосу частот (спектр сигнала имеет ширину порядка $1/\tau$, где τ - время передачи одного бита источника); потенциально помехоустойчив, так как использует противоположные сигналы.

Недостатки: плохо поддерживает синхронизацию - если идёт последовательность одних нулей или одних единиц, то у сигнала нет никаких признаков, помогающих различить, где кончается один импульс и начинается другой. Кроме того, если в передаваемой информации число единиц не равно числу нулей или если импульсы однополярные, то в суммарном сигнале будет присутствовать постоянная составляющая.

Используется в тех системах, где главное – простота генерации и приёма, и одновременно важна скорость передачи, а синхронизация может быть обеспечена извне, например, на внутренних шинах ЭВМ и различных цифровых устройств. Без модификаций для внешних телекоммуникаций используется только на небольшие

расстояния с небольшой скоростью (например, связь ЭВМ с периферийными устройствами).

Однако, существует несложный приём, с помощью которого можно улучшить способность к самосинхронизации у данной системы сигналов. Этот приём называется **битстаффинг** (bit stuffing), и заключается в том, что после определённого количества одинаковых бит в передаваемую последовательность вставляется служебный бит противоположного значения, который в данном случае вызывает переход сигнала на другой уровень, и по этим переходам можно осуществить синхронизацию. На приёмном конце такие служебные биты удаляются, так как не являются частью передаваемой информации. Так, в интерфейсе CAN (стандарт для промышленных сетей) используется NRZ с битстаффингом после каждых 5 одинаковых бит.

2) **AMI-код** (Alternate Mark Inversion) – рис. 12.1, б.

Достоинства: простота генерации и хорошее использование полосы частот, как у NRZ; гораздо лучшая, чем у NRZ, способность к самосинхронизации, так как при передаче единицы в сигнале всегда присутствует перепад, по которому приёмник может восстанавливать синхронизацию; отсутствие постоянной составляющей независимо от передаваемой информации.

Недостатки: теоретически обладает несколько худшей потенциальной помехоустойчивостью, чем NRZ, так как вместо противоположных использует систему с тремя уровнями сигналов; при длинной последовательности нулей возможна потеря синхронизации.

В целом, весьма хороший код, широко использовался на практике в магистральных кабельных системах связи во второй половине 20 века. При этом возможная потеря синхронизации при передаче длинной последовательности нулей исключалась битстаффингом, причём вставлялся импульс, нарушающий правила кодирования для AMI кода, т.е. имеющий ту же полярность, что и предыдущий – это облегчало распознавание и удаление этих служебных вставок на приёмном конце.

3) **Код Bipolar RZ** (биполярный с возвращением к нулю) - рис. 12.1, в.

Достоинства: прост в генерации; обладает очень хорошей способностью к самосинхронизации, так как имеет перепад сигнала в середине интервала передачи каждого бита; потенциальная помехоустойчивость высокая, так как использует противоположные сигналы.

Недостатки: хуже, чем NRZ и AMI, использует полосу частот, так как при той же τ (времени передачи одного бита) требует в два раза большую полосу частот (спектр сигнала имеет ширину порядка $2/\tau$); постоянная составляющая суммарного сигнала отлична от нуля, если единиц и нулей в передаваемой последовательности не поровну.

Используется в некоторых среднескоростных локальных сетях, где ширина полосы частот используемого канала не является ограничивающим фактором.

4) **Код "Манчестер - 2"** -- рис. 12.1, г.

Достоинства: обладает очень высокой способностью к самосинхронизации, так как имеет перепад сигнала в середине интервала передачи каждого бита; обладает большой потенциальной помехоустойчивостью, так как использует противоположные сигналы; суммарный сигнал не имеет постоянной составляющей.

Недостатки: по сравнению с NRZ и AMI требует в два раза большую полосу частот при том же времени передачи одного бита (спектр сигнала имеет ширину порядка $2/\tau$).

Широко используется в среднескоростных локальных сетях ЭВМ, когда ширина полосы частот используемого канала не является ограничивающим фактором – в частности, это стандартный код для сети "Ethernet" (до 10 Mbit/s).

Что касается высокоскоростных локальных сетей, таких как Fast Ethernet (100 Mbit/s) и Gigabit Ethernet (1 Gbit/s), то для них "Манчестер - 2" уже не подходит - слишком широкий у него становится на таких скоростях спектр для передачи сигнала по стандартизованному для этих сетей кабелю.

Для одного из вариантов Fast Ethernet (для 100Base-FX) используется т.н. **NRZI кодирование**, которое в некотором смысле объединяет в себе свойства NRZ и AMI**:** используются двухуровневые сигналы, при этом если на интервале передачи одного бита передавалась единица, то в начале этого интервала уровень сигнала меняется на противоположный, а если ноль - то уровень остаётся таким же, как и был. Таким образом, при передаче последовательности единиц перепады в сигнале есть в начале каждого интервала передачи бита, и получающиеся импульсы имеют длительность τ, т.е. спектр сигнала имеет ширину порядка $1/\tau$ и полоса частот канала используется эффективно. Проблемы с синхронизацией могут возникнуть только при передаче последовательности нулей. Для борьбы с этим применяется так называемое 4B/5B кодирование, когда каждые 4 бита источника, перед тем как поступить в передатчик, преобразуются в 5 бит согласно таблице 12.1**.**

Таблица 12.1. Код 4B/5B

Биты источника	Биты кода	Биты источника	Биты кода
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

Нетрудно видеть, что какова бы ни была последовательность бит источника, после кодирования больше трёх нулей подряд не будет, при этом необходимая ширина полосы частот канала возрастает только в 1.25 раза.

Заметим, что и для другого популярного варианта Fast Ethernet - 100Base-TX – также применяется 4B/5B кодирование, только код там другой – т.н. MLT-3 – он похож на NRZI, только трёхуровневый.

Для Gigabit Ethernet используемые сигналы также зависят от варианта стандарта. Для некоторых из них, например, стандартизован сигнальный код NRZI с кодированием 8B/10B для улучшения синхронизирующих свойств сигнала.

Вообще, NRZI весьма популярный сейчас код – в интерфейсе USB используется NRZI с битстаффингом после каждых 6 единиц (сменой уровней кодируется 0).

Всё это были сигналы, предназначенные для широкополосных каналов. Теперь рассмотрим системы сигналов, предназначенные для работы по узкополосным каналам, у которых полоса частот спектра сигналов должна быть сосредоточена вокруг некоторой частоты, называемой **несущей**.

Так как здесь очень важно требование того, чтобы спектр сигнала был узкий и весь сосредоточен вокруг заданной частоты, то базовыми функциями для системы таких сигналов являются отрезки синусоиды и косинусоиды этой частоты на длине одного периода. Будем изображать используемые для передачи информации сигналы для одного элемента суммарного сигнала точками на координатной плоскости. Главные методы передачи информации такими сигналами - это **фазовая** и **амплитудно-фазовая манипуляции**.

1) **Двухпозиционная фазовая манипуляция** - рис. 12.2, а. Самая простая в реализации, самая помехоустойчивая, так как использует противоположные сигналы. Но обеспечивает передачу не более одного бита за период несущей частоты.

2) **Четырёхпозиционная фазовая манипуляция** - рис. 12.2, б. Сложнее, требует более качественных (то есть с большим отношением сигнал/шум) каналов, так как менее помехоустойчива. Зато на качественных каналах обеспечивает передачу 2 бит на периоде несущей, то есть в два раза большую скорость передачи, чем двухпозиционная фазовая манипуляция.

Аналогично строится и восьмипозиционная фазовая манипуляция, но более чем 8-позиционная фазовая манипуляция на практике не используется, так как из-за уменьшающегося расстояния между сигнальными точками помехоустойчивость таких систем сигналов сильно падает.

3) **Шестнадцатеричная амплитудно-фазовая манипуляция** - рис. 12.2, в. Как видно из рисунка, здесь у сигнала три возможных амплитуды и 12 возможных значений фазы. Обеспечивает очень высокую скорость передачи – 4 бита на период несущей частоты, но может использоваться только для высококачественных каналов связи с большим отношением сигнал/шум.

Существует и множество других подобных систем сигналов. Фактически, каждая из них используется в каком-либо стандарте на передачу данных по телефонным линиям общего назначения - в основном такие сигналы применяются именно там.

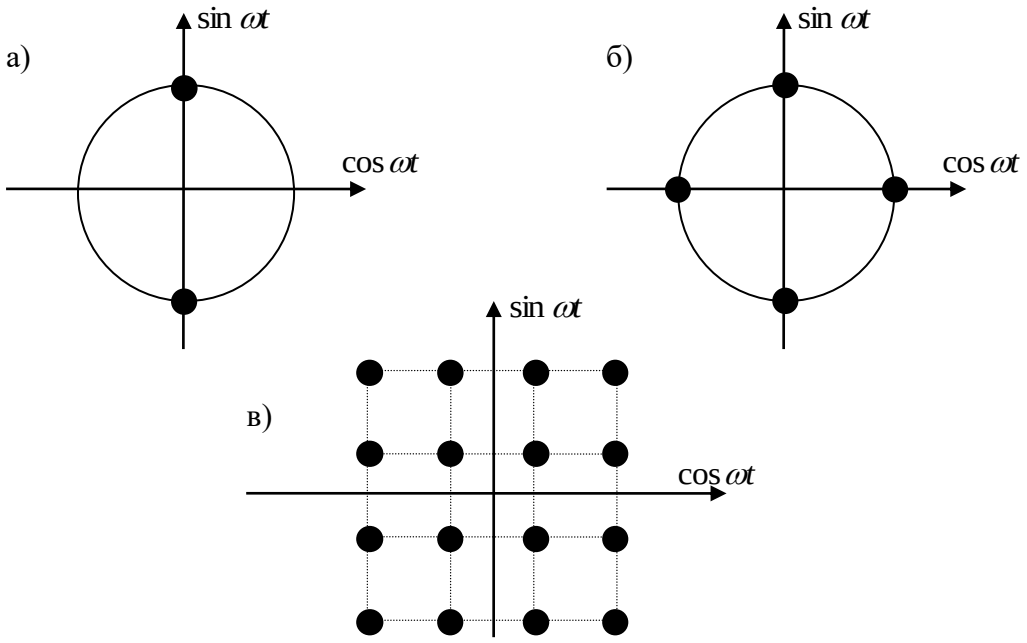

Рисунок 12.2. Примеры кодов для узкополосных каналов

13. Передача последовательностей сообщений

До сих пор в нашем анализе речь шла о передаче одного сообщения от источника к получателю, и единственное условие относительно поведения системы связи во времени, которое мы учитывали, - это условие о том, что передача одного сообщения осуществляется за конечное время. Теперь от передачи одного единственного сообщения мы переходим к передаче последовательности сообщений, когда, передав одно сообщение, система связи сразу же переходит к передаче следующего.

При этом мы будем использовать два различных подхода. Первый состоит в том, что мы будем рассматривать совокупность сигналов, переданных в канал за определённый промежуток времени, как один сложный сигнал. Например, если система связи за каждые T единиц времени передаёт одно сообщение, выбираемое из M возможных, то

последовательность из K таких сообщений можно представить как передачу одного из M^K возможных сообщений, осуществляемую за KT единиц времени. Мы будем широко использовать такой подход для доказательства теоретических результатов, в том числе и для асимптотического анализа (когда $T\to\infty$). Но наряду с этим, мы будем использовать и более практический подход, при котором явно учитывался бы последовательный характер передачи сообщений. Для этого нам необходимо ввести несколько новых понятий и пересмотреть некоторые предположения о нашей модели системы связи.

Прежде всего, определим понятие **скорости дискретного источника сообщений**, или как её ещё называют, **информационной скорости**. Пусть имеется источник сообщений, который производит последовательность дискретных двоичных символов (т.е. M=2). Предположим, что каждый из этих символов может быть либо 0, либо 1 с равной вероятностью и независимо от всех других событий. Пусть этот источник производит за единицу времени R таких символов. Тогда один символ производится за $1/R$, а за время T, кратное величине $1/R$, источник генерирует RT двоичных символов, причем каждая из 2^{RT} возможных последовательностей генерируется с равной вероятностью. То есть передатчик, связанный с этим источником, должен быть способен передать одно из 2^{RT} равновероятных сообщений в течение каждого из последовательных интервалов длины T.

В ситуациях, аналогичных рассмотренной выше, будем называть R информационной скоростью источника и измерять ее числом двоичных символов в единицу времени (бит/с - от английского сокращения bit - BInary digiT).

В общем случае для источника, который за единицу времени генерирует R сообщений, каждое из которых с равной вероятность и независимо от других выбирается из M возможных, определим информационную скорость источника как

$$R' = R\log_2 M, \quad (бит/с) \tag{83}$$

Разумность определения скорости источника таким образом можно понять, заметив, что элементы множества из M сообщений можно пронумеровать и записав их номера в двоичной форме, получить эквивалентное представление в двоичном виде. Если M=2^K, то в полученной таким образом двоичной последовательности каждый двоичный символ независим от остальных и с равной вероятностью принимает значения 0 и 1.

Ещё одно подтверждение полезности введённого нами определения информационной скорости можно продемонстрировать на следующем примере**:** пусть есть два независимых источника. Один из них за промежуток T генерирует одно из M_1 равновероятных сообщений, а другой - одно из M_2 равновероятных сообщений. Если каждый из источников связан с отдельным передатчиком, то первый передатчик должен передавать со скоростью $R_1 = \frac{1}{T}\log_2 M_1$, а второй - со скоростью $R_2 = \frac{1}{T}\log_2 M_2$. Если они оба подключены к одному передатчику, то за время T этот передатчик должен передавать одно из M=M_1 M_2 равновероятных сообщений, и его скорость

$$R = \frac{1}{T}\log_2 M_1M_2 = \frac{1}{T}\log_2 M_1 + \frac{1}{T}\log_2 M_2 = R_1 + R_2.$$

Для краткости и простоты мы в основном будем рассматривать передачу двоичных последовательностей, но все результаты могут быть обобщены на случай произвольного алфавита источника (произвольного M).

Следующее понятие, которое необходимо ввести - **мощность сигнала.** Энергию для процесса, который может длиться произвольно долго, ограничивать не имеет смысла, но если на передачу одного сообщения мы намерены тратить как можно меньше времени, то на практике придётся учесть ограничение на мощность передатчика. Для сигнала $s(t)$ длительности T **средняя мощность** P_s определяется как

$$P_s = \frac{1}{T}\int_0^T s^2(t)dt = \frac{E_s}{T} \tag{84}$$

Ограничение на среднюю мощность сигнала может иметь важное значение, так как в канале с аддитивным белым гауссовским шумом, как мы видели, помехоустойчивость оптимального приёма зависит от отношения сигнал/шум, то есть от E_s/N_0. Если средняя мощность ограничена, то за время T это отношение не может быть больше, чем P_sT/N_0.

Введём ещё понятие **средней энергии на бит**, затрачиваемой передатчиком. Если скорость источника равна R, то можно считать, что каждые $1/R$ единицы времени генерируется один двоичный символ. Если при этом средняя мощность ограничена величиной P_s, то средняя энергия, приходящаяся на один бит, которую мы обозначим E_b, есть

$$E_b = \frac{P_s}{R} \tag{85}$$

Средняя энергия на бит, требуемая различными системами связи для достижения заданной помехоустойчивости, является мерой их относительной эффективности.

Далее нам необходимо ввести ещё одно ограничение на систему сигналов, которое ранее во внимание не принималось – **ограничение на точность воспроизведения сигналов** приёмником и передатчиком. К ранее введённым нами предположениям:

– об ограничении сигналов по времени (то есть то, что на практике сигнал имеет конечную длительность);

– об ограничении сигнала по полосе частот (то есть то, что за пределами конечной полосы частот находится столь малая по энергии часть спектра сигнала, что ей можно пренебречь);

– об ограничении сигнала по энергии (то есть то, что энергия сигнала конечна);

теперь мы должны добавить ещё предположение о том, что для того, чтобы два сигнала различались, разница между ними должна быть больше некоторой заданной величины. Например, чтобы выполнялось условие

$$\int_{-\infty}^{+\infty} \left(s_1(t) - s_2(t)\right)^2 dt \geq \varepsilon > 0 \tag{86}$$

для некоторого заданного ε, определяемого возможностями аппаратуры. Учёт этого ограничения, наряду с ранее введёнными, приводит к следующему результату**:** количество измерений в пространстве сигналов (количество векторов в базисе пространства) становится конечным, и при этом не более чем линейно зависящим от длительности сигналов и ширины полосы частот их спектров.

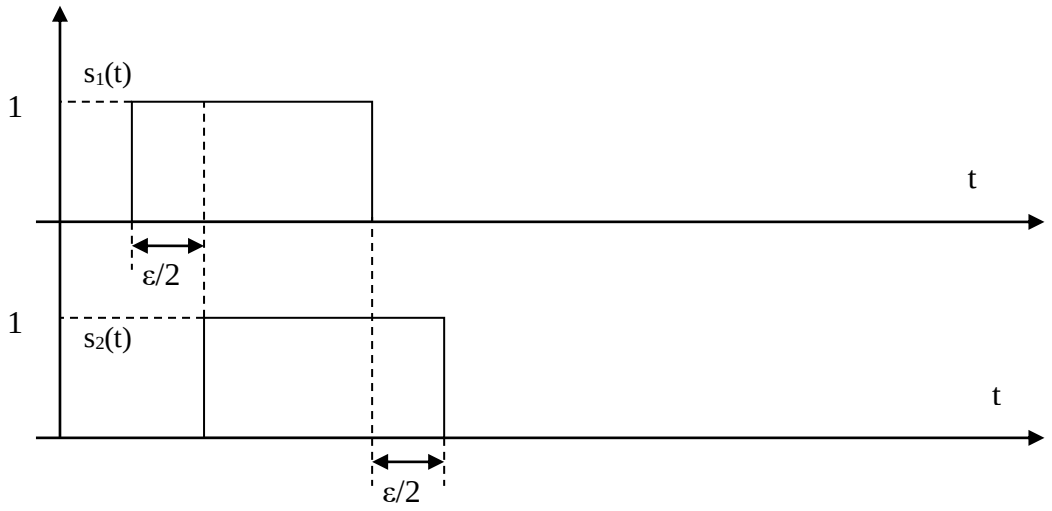

Рисунок 13.1. Различение двух импульсов

Строгое доказательство этого результата довольно сложно, оно формулируется в ряде теорем, доказанных впервые Ландау, Поллаком и Шенноном. Мы же продемонстрируем, почему это так, на предельно упрощённом примере. Рассмотрим систему прямоугольных импульсов одинаковой длительности и одинаковой единичной амплитуды, неточность в воспроизведении которых заключается только в их положении на оси времени. Тогда для того, чтобы два таких импульса различались, согласно условию (86) их положение на оси времени должно различаться не менее чем на $\varepsilon/2$ (см. рис. 13.1). Если потребовать, чтобы все такие импульсы лежали внутри временного интервала $[\ 0,T\ ]$, и были различимы друг от друга, то нетрудно видеть, что их не может быть больше, чем $2T/\varepsilon$. И в ортонормальном базисе такой системы сигналов не может быть больше, чем $2T/\varepsilon$ векторов.

В случае произвольной системы сигналов с учётом всех введённых ограничений, Ландау, Поллаком и Шенноном была получена оценка сверху на количество измерений в пространстве сигналов (то есть, на количество различимых друг от друга ортогональных сигналов):

$$N \leq 2TW+1, \tag{87}$$

где N-количество измерений в пространстве сигналов (функций в ортогональном базисе), T – длительность сигналов, а W – ширина полосы частот спектра сигналов (см. [5]).

Для полноты рассмотрения заметим, что линейная оценка сверху ещё не гарантирует, что зависимость между длительностью сигнала и количеством измерений пространства сигналов именно линейная. Но несложно привести примеры систем сигналов, у которых эта линейная зависимость имеет место, то есть

$$N=DT \tag{88}$$

Например, пусть $x(t)$ - некоторый сигнал, тождественно равный нулю вне интервала длительности τ и занимающий полосу частот W (простейший случай - прямоугольный импульс). Тогда за время T можно передать T/τ таких сигналов, чтобы они не перекрывались (и, значит, были бы ортогональными). То есть $D=1/\tau$ (ортогональных сигналов в единицу времени). Вспомнив, что при заданной форме ширина спектра сигнала обратно пропорциональна его длительности, увидим, что коэффициент D в этом случае прямо пропорционален W.

Другой пример ортогонального базиса с числом измерений, подчиняющихся (88) - отрезки синусоид с целым числом периодов на интервале $[0,T]$ и частотами, не превосходящими W (частоты кратны $1/T$).

Можно привести и другие примеры систем ортогональных сигналов длительности T, число которых линейно растет с ростом T. В реальных каналах такое поведение сохраняется, но D имеет меньшую величину, чем в (87). Обычно число практически реализуемых ортогональных сигналов в полосе W и длительностью T лежит между TW и $1{,}5\ TW$, в зависимости от определения W и сложности реализации.

И в заключение параграфа, обсудим, какие системы сигналов будут получаться, если передатчик будет последовательно передавать сообщения, используя для передачи каждого заданное множество сигналов.

Если возможных сигналов два, и длительность у них τ, то на интервале T у такой системы сигналов будет $N=T/\tau$ измерений, а сама система сигналов – это известная нам система сигналов в вершинах гиперкуба (примерами из практики могут служить коды NRZ, “Манчестер-2” и двухпозиционная фазовая манипуляция). В каждом измерении её можно закодировать максимум 1 бит источника. Если же возможных сигналов больше двух, то на интервале T у такой системы сигналов будет по-прежнему $N=T/\tau$ измерений, но в каждом измерении будет больше сигнальных точек, и максимальное количество бит источника, которые можно закодировать в одном измерении, также будет больше (как у четырёхпозиционной фазовой и 16-ричной амплитудно-фазовой манипуляции).

Заметим следующее свойство получающихся таким образом сигналов – они состоят из набора более простых сигналов. Такие сигналы принято называть **сложными**, а

составляющие их сигналы – **элементами**. Если в каждом элементе сигнала кодируется более 1 бита, то информационная скорость источника не совпадает с количеством элементов сигнала, предаваемых в единицу времени. Поэтому для характеристики передачи сложных сигналов введена специальная единица измерения количества элементов сигнала, передаваемых в секунду – **бод**.

14. Передача отдельных символов и передача блоков ортогональными сигналами

Чтобы увидеть, что различные системы связи, характеризуемые одной и той же величиной E_b, при передаче последовательных сообщений могут вести себя совершенно по-разному по отношению к надёжности передачи, рассмотрим передачу последовательности $K=RT$ равновероятных двоичных символов двумя способами.

Первый (весьма простой) способ - передать информацию последовательностью K неперекрывающихся сигналов в системе связи с бинарными противоположными сигналами (например, прямоугольными импульсами), где каждый сигнал соответствует передаче одного бита. Энергия каждого импульса E_b, общая расходуемая энергия $E_s=KE_b$. Найдём вероятность P_R принять все K бит без ошибки в предположении о наличии в канале аддитивного белого гауссовского шума (вероятность ошибки, очевидно P_E =1- P_R).

Как отмечалось в предыдущем параграфе, система сигналов, порождаемая при таком способе передачи – это K-мерный гиперкуб с 2^K вершинами, каждая из которых соответствует какой-либо K-битовой последовательности, полученной от источника. Из (74) имеем

$$P_E = 1-(1-p)^K = 1-(1-p)^{RT}, \tag{89}$$

где $p = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) = Q\left(\sqrt{\frac{2P_s}{RN_0}}\right)$ для канала с белым гауссовским шумом со спектральной мощностью $N_0/2$. Видно, что с ростом T, то есть с ростом количества бит K переданной последовательности, вероятность ошибки стремится к 1. При фиксированных K и $N_0/2$ вероятность ошибки можно уменьшить, только увеличивая E_b, что достигается либо увеличением средней мощности, либо уменьшением скорости передачи R.

Теперь рассмотрим другую схему передачи, где каждые T единиц времени передается один из 2^K ортогональных импульсов, причем длительность одного импульса есть $\tau=T/2^K$, а его положение внутри интервала T определяется передаваемой последовательностью, как двоичным K -разрядным числом**:**

$$s_i(t) = \sqrt{E_s}\phi_i(t - i\tau), \quad i = 0,1,...,2^K - 1$$

где $\phi_i(t)$ - импульс длительности τ и единичной энергии, а $E_s=KE_b=TP_s$ – энергия сигнала. Для анализа этой системы мы воспользуемся аддитивной границей вероятности ошибки передачи одного из M равновероятных ортогональных сигналов равной энергии (82):

$$P_E \le (M-1)Q\left(\sqrt{\frac{E_s}{N_0}}\right)$$

Чтобы преобразовать это выражение, нам потребуется следующая оценка для величины функции ошибок $Q(y)$, определённой выражением (62)**:**

$$Q(y) \le \frac{1}{2}\exp(-\frac{y^2}{2}) \tag{90}$$

Докажем сначала, что эта оценка справедлива. Для этого рассмотрим двумерный гауссовский вектор с независимыми компонентами, нулевым средним и единичными дисперсиями компонент (X,Y), и двумерной плотностью:

$$f(x,y) = \frac{1}{2\pi}\exp\left(-\frac{x^2+y^2}{2}\right)$$

Пусть D_1 - область, изображенная на рис.14.1, а. Вероятность для (X,Y) попасть в область D_1 есть, как нетрудно видеть:

$$\Pr\{(X,Y)\in D_1\}=\int_{\alpha}^{+\infty}\int_{\alpha}^{+\infty}\frac{1}{2\pi}\exp\left(-\frac{x^2+y^2}{2}\right)dx\,dy=\int_{\alpha}^{+\infty}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)dx\int_{\alpha}^{+\infty}\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{y^2}{2}\right)dy=(Q(\alpha))^2$$

Теперь рассмотрим область D_2 на рис. 14.1, б. Очевидно, что $\Pr\{(X,Y)\in D_2\ge\Pr\{(X,Y)\in D_1\}$, а из соображений симметрии нетрудно видеть, что $\Pr\{(X,Y)\in D_2=\frac{1}{4}\Pr\{(X,Y)\in D_3\}$ где D_3 - внешность круга на рис. 14.1, б. Для отыскания вероятности попадания в D_3 перейдем к полярным координатам:

$$\Pr\{(X,Y)\in D_3\}=\iint_{D_3}f(x,y)dxdy=\int_{\sqrt{2}\alpha}^{+\infty}\int_{0}^{2\pi}\frac{1}{2\pi}\exp\left(-\frac{\rho^2}{2}\right)\rho\,d\rho\,d\varphi=\int_{\sqrt{2}\alpha}^{+\infty}\exp\left(-\frac{\rho^2}{2}\right)\rho d\rho=\exp(-\alpha^2)$$

откуда следует, что $\Pr\{(X,Y)\in D_2=\frac{1}{4}\exp(-\alpha^2)$ и $Q(y)\le\sqrt{\frac{1}{4}\exp(-\alpha^2)}=\frac{1}{2}\exp(-\frac{\alpha^2}{2})$, то есть (90) доказано.

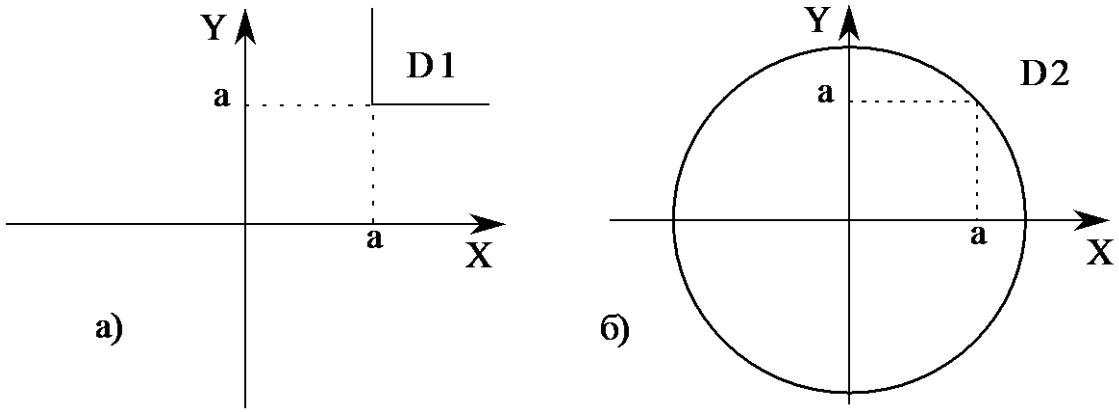

Рисунок 14.1. Построение оценки для функции ошибок

Вернёмся к оценке вероятности ошибки передачи для передачи блоков ортогональными сигналами:

$$P_E\le(M-1)Q\left(\sqrt{\frac{E_s}{N_0}}\right)<M\exp\left(-\frac{E_s}{2N_0}\right)$$

Подставив сюда $M=2^K=2^{RT}$ и $E_s=KE_b=TP_s$, получим

$$P_E<2^{RT}\exp\left(-\frac{TP_s}{2N_0}\right)=\exp\left(-T\left(\frac{P_s}{2N_0}-R\ln 2\right)\right)\qquad(91)$$

то есть с ростом T вероятность ошибки будет стремиться к нулю, если

$$R<\frac{P_s}{2N_0\ln 2}\approx 0.72\frac{P_s}{N_0}\qquad(92)$$

Если выразить всё через E_b и K, то

$$P_E<2^K\exp\left(-\frac{KE_b}{2N_0}\right)=\exp\left(-K\left(\frac{E_b}{2N_0}-\ln 2\right)\right)\qquad(93)$$

и условие стремления вероятности ошибки к нулю есть

$$\frac{E_b}{N_0}>2\ln 2\approx 1.39\qquad(94)$$

Как видно, две рассмотренные системы ведут себя относительно вероятности ошибки совершенно по-разному. В первом случае вероятность ошибки с ростом K стремится к 1, как бы велико ни было отношение сигнал/шум E_s/N_0. У второй системы, увеличивая K, мы можем сделать вероятность ошибки сколь угодно малой, если только E_b/N_0 больше чем 1,39. Иными словами, отношение сигнал/шум E_b/N_0 устанавливает некоторую границу для максимальной скорости передачи - если скорость меньше этого максимума, то вероятность P_E может быть сделана сколь угодно малой выбором достаточно больших T.

Геометрическая интерпретация векторов сигналов позволяет лучше понять причины этого различия: при передаче отдельных символов расстояние между сигналами не меняется, а число соседей и размерность пространства растут линейно с ростом K.

При передаче блоков ортогональными сигналами расстояние между сигналами растет пропорционально $\sqrt{K}$ за счет новых измерений и перенормировки амплитуды. И хотя число соседей тоже растет, именно увеличение расстояния между сигналами при больших E_b/N_0 определяет поведение ошибки. А вот при малых E_b/N_0 поведение определяется ростом числа соседей.

К сожалению, схема передачи блоков ортогональными сигналами на практике не может считаться решением проблемы надежной связи по гауссовскому каналу. При R, близких к 0,72 P_s/N_0 для надежной передачи необходимы очень большие значения T. В свою очередь, при больших T число ортогональных сигналов, равное 2^{RT}, - очень велико, а длительность импульса, равная $T/2^{RT}$, - очень мала. Такой импульс невозможно будет передать по реальным каналам с ограниченной полосой частот. Например, пусть скорость передачи в канале R=100 бит/с и отношение P_s/N_0 таково, что для достижения нужной помехоустойчивости необходимо T=1 с (то есть K=100), так что $M=2^{100}$. Если приблизительно принять, что для прямоугольного импульса его ширина спектра f и длительность τ связаны соотношением $\tau f\approx 1$, то в такой системе $\tau=T/2^{RT}\approx 10^{-30}$ с, и $f\approx 10^{30}$ Гц, что в несколько раз выше, чем граница рентгеновского диапазона электромагнитного излучения.

Кроме того, с уменьшением τ пиковая мощность, которую должен излучать передатчик, растет и тоже может принимать чрезмерно большие значения. Все это делает рассмотренную схему практически нереализуемой.

Чего же можно добиться в реальных системах связи, когда мы вынуждены учитывать все введённые выше условия? Один из главных результатов теории связи состоит в том, что надежная передача может быть организована и при таких условиях. Теперь мы готовы перейти к доказательству этого факта для канала с аддитивным белым гауссовским шумом. Мы повторим доказательство, принадлежащее Шеннону. Отметим следующее необычное обстоятельство: мы докажем существование системы надёжной передачи последовательных сообщений по каналу с ограниченной полосой частот и аддитивным белым гауссовским шумом без ее построения.

15. Надежная передача по каналу с аддитивным белым гауссовским шумом

В качестве элементов сложных сигналов, используемых для передачи битовой последовательности от источника, возьмем два противоположных сигнала, имеющих длительность τ . За базисный вектор, по которому раскладываются эти сигналы, возьмём сигнал той же формы и единичной энергии. Тогда один из пары сигналов образуется из базисного вектора путём умножения на $\sqrt{E_N}$, а другой – на $-\sqrt{E_N}$, где E_N - энергия элемента сигнала - эта величина имеет также смысл **энергии сложного сигнала, приходящейся на одно измерение пространства сигналов.** Как указывалось выше, все возможные последовательности таких сигналов на интервале T образуют систему сигналов в вершинах гиперкуба, количество измерений которого N связано с T равенством (88) с $D=1/\tau$. Число измерений пространства используемых сигналов N часто называют **длиной кодового блока**. Такую систему сигналов мы будем использовать для передачи

информации от источника двоичной информации со скоростью R, причем будем считать, что

$$D>R. \tag{95}$$

Число возможных двоичных последовательностей, порождаемых источником за время T, есть $M=2^{RT}$, а число сигналов, являющихся вершинами DT - мерного гиперкуба, - 2^{DT}. Вершин, которые надо использовать для передачи, меньше чем их общее число. При этом доля вершин, которые мы должны использовать, равна

$$\frac{2^{RT}}{2^{DT}} = 2^{-(D-R)T} \tag{96}$$

Вследствие (95) она стремится к нулю с ростом T . Поэтому можно надеяться, что вероятность ошибки не будет стремиться к 1 с ростом T, как это было для $D=R$ из-за слишком близкого "соседства" сигналов. Вопрос в том, как выбирать используемые вершины. Но мы обойдём этот вопрос следующим образом: рассмотрим не одну конкретную систему связи данного типа, а сразу все возможные такие системы связи, каждая из которых состоит из передатчика, канала и оптимального приёмника. Каждая из них использует свою систему сигналов $\{s_i(t)\}_{i=1,\dots,M}$, а в остальном эти системы совершенно идентичны.

Всего для $n=2^{DT}$ вершин гиперкуба и $M=2^{RT}$ сигналов $s_i(t)$ имеется 2^{nM} вариантов выбора $s_i(t)$ (включая, например, и такой, где M раз мы выбираем один и тот же вектор для всех сообщений - явно плохой выбор). Предположим, что каждый из этих 2^{nM} вариантов используется в одной и только одной системе связи из нашего набора, и приёмник соответствующей системы оптимален для этого набора.

Следуя установившейся терминологии, будем называть вектор, соответствующий каждому из выбранных сигналов - **кодовым словом**, а совокупность всех таких векторов - **кодом**.

Ясно, что каждая система в нашем наборе обладает определённой вероятностью ошибки. Обозначим вероятность ошибки для k-той системы P_k, $k=1,2\dots,2^{NM}$. Некоторые из этих систем, например, с кодами, в которых все M векторов $s_i(t)$ сопоставлены одной и той же вершине гиперкуба, обладают очень большой вероятностью ошибки. Для других систем она меньше, для некоторых из них она может быть очень мала. Мы докажем этот факт, вычислив верхнюю оценку для **средней** (по всему набору кодов) вероятности ошибки.

Итак, для каждого конкретного набора $\{s_i(t)\}^{(k)}_{i=1,\dots,M}$

$$\Pr\{\{s_i(t)\}^{(k)}_{i=1,\dots,M}\} = \frac{1}{2^{nM}}, \qquad \Pr\{\hat{m}\neq m \mid \{s_i(t)\}^{(k)}_{i=1,\dots,M}\} = P_k \tag{97}$$

Тогда

$$\overline{P}_E = \sum_{k=1}^{2^{NM}} \Pr\{\hat{m}\neq m \mid \{s_i(t)\}^{(k)}_{i=1,\dots,M}\} \Pr\{\{s_i(t)\}^{(k)}_{i=1,\dots,M}\}. \tag{98}$$

Аналогично, средняя по всему ансамблю кодов условная вероятность ошибки при условии, что передаётся сообщение m_j, есть

$$\overline{\Pr\{\hat{m}\neq m \mid m=m_j\}} = \sum_{k=1}^{2^{NM}} \Pr\{\hat{m}\neq m \mid m=m_j, \{s_i(t)\}^{(k)}_{i=1,\dots,M}\} \Pr\{\{s_i(t)\}^{(k)}_{i=1,\dots,M}\}. \tag{99}$$

Применяя к каждому конкретному коду аддитивную границу для вероятности ошибки, полученную в параграфе 11, имеем

$$\Pr\{\hat{m}\neq m \mid m=m_j, \{s_i(t)\}^{(k)}_{i=1,\dots,M}\} \le \sum_{\substack{i=1\\ i\neq j}}^{M} P_2[\vec{s}_j, \vec{s}_i], \tag{100}$$

где $P_2[\vec{s}_j, \vec{s}_i]$ - вероятность ошибки для системы из двух сигналов $\vec{s}_j$ и $\vec{s}_i$.

Подставляя (100) в (99) для средней условной вероятности ошибки получаем оценку

$$\overline{\Pr\{\hat{m} \neq m \mid m = m_j\}} \leq \sum_{k=1}^{2^{NM}} \Pr\{\{s_i(t)\}_{i=1,\dots,M}^{(k)}\} \sum_{\substack{i=1 \\ i \neq j}}^{M} P_2[\vec{s}_j, \vec{s}_i]. \tag{101}$$

Сменим порядок суммирования:

$$\overline{\Pr\{\hat{m} \neq m \mid m = m_j\}} \leq \sum_{\substack{i=1 \\ i \neq j}}^{M} \left(\sum_{k=1}^{2^{NM}} \Pr\{\{s_i(t)\}_{i=1,\dots M}^{(k)}\} P_2[\vec{s}_j, \vec{s}_i] \right) = \sum_{\substack{i=1 \\ i \neq j}}^{M} \overline{P_2[\vec{s}_j, \vec{s}_i]}. \tag{102}$$

Для канала с аддитивным белым гаусовским шумом $P_2[\vec{s}_j, \vec{s}_i]$ зависит, как мы знаем, только от евклидова расстояния между $\vec{s}_j$ и $\vec{s}_i$:

$$P_2[\vec{s}_j, \vec{s}_i] = Q\left(\frac{|\vec{s}_j - \vec{s}_i|}{\sqrt{2N_0}} \right). \tag{103}$$

Если у векторов $\vec{s}_j$ и $\vec{s}_i$ различны *h* координат, то квадрат расстояния между ними есть

$$|\vec{s}_j - \vec{s}_i|^2 = \sum_{n=1}^{N} (s_{jn} - s_{in})^2 = \left(2\sqrt{E_N}\right)^2 h = 4hE_N \tag{104}$$

По нашему предположению, $\vec{s}_i$ с равной вероятностью может быть любой вершиной гиперкуба независимо от $\vec{s}_j$. Поэтому вероятность того, что s_{in} совпадает с s_{jn} есть 1/2 для всех $n=1,\dots,N$. Следовательно, вероятность того, что у векторов $\vec{s}_j$ и $\vec{s}_i$ различны *h* координат – это вероятность *h* успехов в *N* испытаниях Бернулли с p=q=1/2:

$$\Pr\{h\} = \binom{N}{h} \left(\frac{1}{2}\right)^N \tag{105}$$

Тогда

$$\overline{P_2[\vec{s}_j, \vec{s}_i]} = \overline{Q\left(\frac{|\vec{s}_j - \vec{s}_i|}{\sqrt{2N_0}} \right)} = \sum_{h=0}^{N} \Pr\{h\} Q\left(\frac{\sqrt{4hE_N}}{\sqrt{2N_0}} \right) = \sum_{h=0}^{N} 2^{-N} \binom{N}{h} Q\left(\sqrt{\frac{2hE_N}{N_0}} \right). \tag{106}$$

Это выражение не зависит от индексов i и j, поэтому если ввести обозначение $\overline{P_{2E}} = \overline{P_2[\vec{s}_j, \vec{s}_i]}$, получим

$$\overline{\Pr\{\hat{m} \neq m \mid m = m_j\}} \leq \sum_{\substack{i=1 \\ i \neq j}}^{M} \overline{P_2[\vec{s}_j, \vec{s}_i]} = (M-1)\overline{P_{2E}} < M\,\overline{P_{2E}}$$

и

$$\overline{\Pr\{\hat{m} \neq m\}} = \sum_{i=1}^{M} \overline{\Pr\{m \neq m \mid m - m_i\}} \Pr\{m_i\} < M\,\overline{P_{2E}} \sum_{i=1}^{M} \Pr\{m_i\} = M\,\overline{P_{2E}}$$

Используем оценку (90), для простоты ещё загрубив её: $Q(y) \leq \frac{1}{2}\exp(-\frac{y^2}{2}) < \exp(-\frac{y^2}{2})$, получим

$$\overline{\Pr\{\hat{m} \neq m\}} < M\,\overline{P_{2E}} < M \sum_{h=0}^{N} 2^{-N} \binom{N}{h} e^{-\frac{hE_N}{N_0}} = M\,2^{-N} \sum_{h=0}^{N} \binom{N}{h} \left(e^{-\frac{E_N}{N_0}} \right)^h \tag{107}$$

Сумма в правой части (107) – это разложение бинома *N*-ной степени, т.е.

$$\overline{\Pr\{\hat{m} \neq m\}} < M\,2^{-N} \left(1 + e^{-\frac{E_N}{N_0}} \right)^N \tag{108}$$

Введём обозначение:

$$R_0 = \log_2\left(\frac{2}{1+e^{-\frac{E_N}{N_0}}}\right) = 1 - \log_2\left(1+e^{-\frac{E_N}{N_0}}\right). \qquad (109)$$

и определим величину

$$R_N = \frac{R}{D} \quad (\textit{бит / измерение}) \qquad (110)$$

равную числу бит, приходящихся на одну размерность гиперкуба или, в более общем случае, на одно измерение пространства сигналов, тогда $M = 2^{RT} = 2^{R_N N}$ и оценку (108) можно записать как

$$\overline{\Pr\{\hat{m} \neq m\}} < 2^{-N(R_0 - R_N)}. \qquad (111)$$

Согласно всем этим соотношениям, независимо от формы их записи, средняя вероятность ошибки, а значит и вероятность ошибки для, по крайней мере, одного кода в ансамбле, может быть сделана как угодно малой за счёт выбора больших *N*, если только R_N меньше R_0.

На самом деле, можно ожидать, что таких "хороших" кодов, у которых ошибка ведёт себя согласно (111), больше, чем один. И это на самом деле так, но доказательство этого сложнее, чем приведённые выше рассуждения, и находится за пределами нашего курса. Нам же достаточно существования хотя бы одного "хорошего" кода.

Как мы видели, R_0. зависит только от E_N/N_0, т.е. от отношения сигнал/шум. При $E_N \to \infty$ $R_0 \to 1$ - это максимальное значение R_0.. Поэтому при любом соотношении сигнал/шум должно выполняться $R_N \leq 1$ для возможности организации надёжной передачи. При этом R_N может быть равным единице, только если $E_N/N_0 = \infty$. Это хорошо согласуется с результатом о том, что при $R=D$, когда $R_N=1$, т.е. число возможных сигналов равно числу вершин гиперкуба, вероятность ошибки стремится к 1, если в канале есть хоть какой-нибудь шум.

Рассуждения, приведённые выше, могут быть распространены на более общий, чем система сигналов в вершинах гиперкуба - случай, когда по каждому измерению пространства сигналов соответствующая компонента может принимать более двух значений. Пусть на отрезке $[-\sqrt{E_N}, \sqrt{E_N}]$ существует A возможных значений, расставленных на этом промежутке через равные расстояния. Примером сигнала с такими элементами для $A=4$ может служить 16-ричная амплитудно-фазовая манипуляция (рис.12.2 в).

При этом возможное число сигналов есть $A^N = 2^{N \log_2 A}$, и энергия любого из них $|\vec{s}_i|^2 \leq NE_N$. Для данной системы могут быть проведены рассуждения, аналогичные двоичному случаю, с той только разницей, что можно рассмотреть ситуацию, когда разные значения амплитуды выбираются с разной вероятностью. Однако для каждого из этих случаев можно доказать неравенство (111), только величина R_0 будет, разумеется, иметь разный вид для разных случаев. Для случая, когда все возможные значения амплитуды элемента сигнала выбираются с равными вероятностями $p=1/A$, графики зависимости R_0. от E_N/N_0 изображены на рис. 15.1.

Видно, где выгоднее использовать различные *A*. При малых отношениях сигнал/шум более высокую скорость R_0 обеспечивает $A=2$, а по мере того как отношение сигнал/шум растёт, становится выгоднее использовать всё большие и большие значения *A*. Эти результаты хорошо согласуются со здравым смыслом: когда шум велик, то сигналы надо максимально разнести друг от друга и потому сделать $A=2$, а когда шум мал, можно уже не разносить их далеко и, таким образом, уместить в каждом измерении больше бит информации.

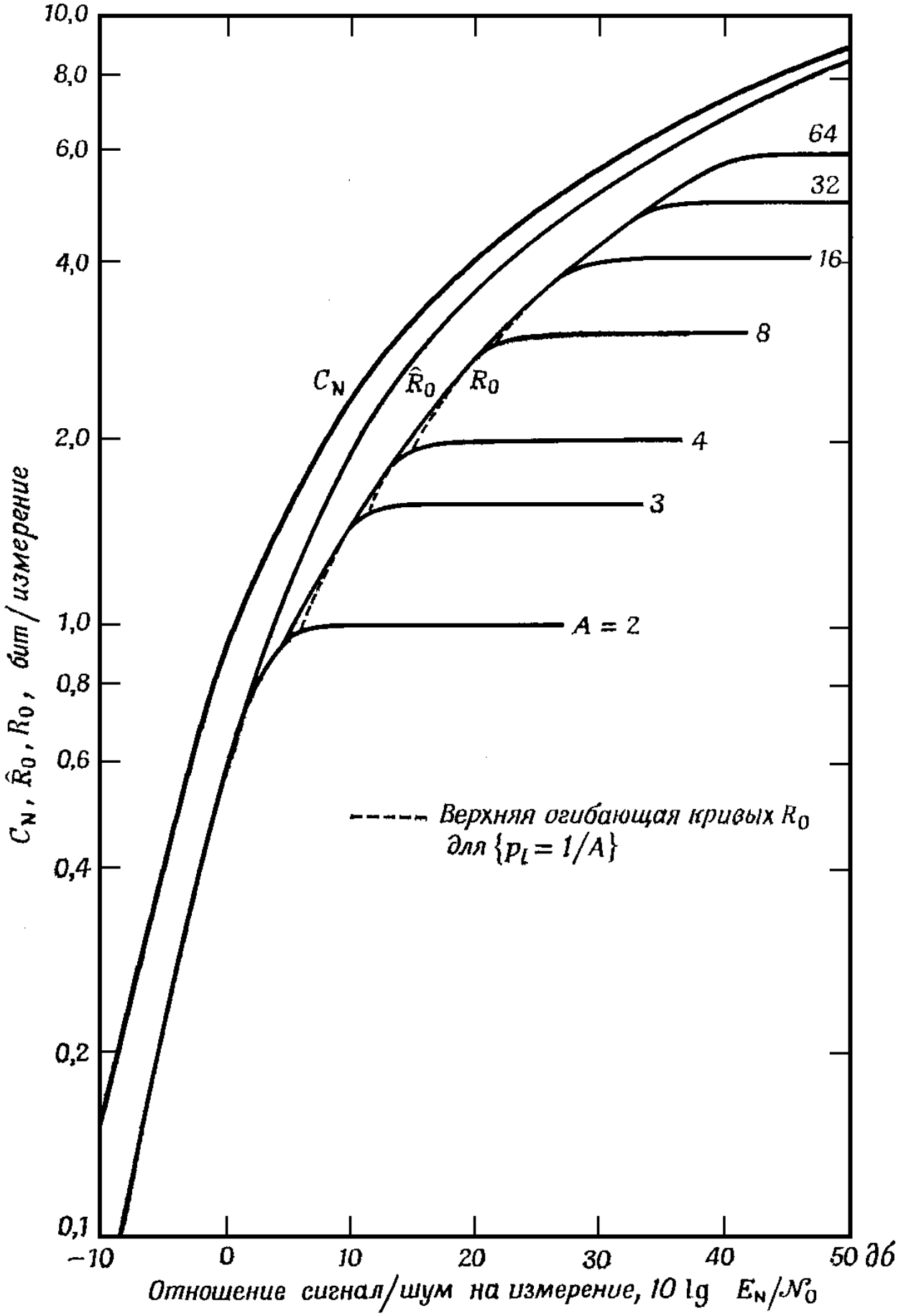

Рисунок 15.1. Граница надёжной блочной передачи и пропускная способность канала.

16. Пропускная способность канала

Мы показали, что существует совокупность сигналов для канала с аддитивным белым гауссовским шумом, такая, что при $R_N<R_0$ вероятность ошибки передачи может быть сделана сколь угодно малой.

Но из того что мы доказали, не следует, вообще говоря, что для скоростей $R_N>R_0$ не может быть предложен какой-либо другой способ построения совокупности сигналов, обеспечивающий произвольно малую вероятность ошибки. Окончательный ответ на этот вопрос даёт доказанная Шенноном теорема. Для сравнения с результатами предыдущего параграфа, приведём здесь формулировку теоремы, использующую параметры E_N и R_N (энергию сигнала на одно измерение и скорость поступления данных в битах на измерение пространства сигналов).

Теорема: Для канала с аддитивным белым гауссовским шумом существует постоянная

$$C_N = \frac{1}{2}\log_2\left(1+\frac{2E_N}{N_0}\right) \tag{112}$$

называемая **пропускной способностью канала**, которая обладает следующими свойствами:

1) Если $R_N>C_N$ и число равновероятных сообщений $M = 2^{R_N N}$ стремится к бесконечности, то для любой совокупности M передаваемых сигналов вероятность ошибки оптимального приёма стремится к 1.

2) Если $R_N<C_N$ и число равновероятных сообщений $M = 2^{R_N N}$ стремится к бесконечности, то существуют такие совокупности M передаваемых сигналов и соответствующие им оптимальные приёмники, для которых вероятность ошибки будет стремиться к нулю.

Доказательство данной теоремы весьма сложно, и здесь не приводится (см. [5]). Заметим только, что доказательство существования в пункте 2 проводится без построения соответствующей системы сигналов.

Если сравнить C_N с R_0 или с огибающей для семейства графиков R_0 при разных A (рис. 15.1), оптимизированных по вероятностям выбора различных амплитуд элементов сигнала (обозначим эту величину как $\hat{R}_0$), то мы увидим, что

$$\frac{1}{2}C_N \le \hat{R}_0 \le C_N \tag{113}$$

Это означает, что теоретически возможны более эффективные методы надёжной передачи, чем рассмотренная нами в предыдущем параграфе блоковая передача.

Приведём ещё выражение C_N через ширину полосы W, определённую каким-либо способом. Для этого заметим, что из неравенства $R_N<C_N$ следует, что $R<DC_N$ и если допустимое число измерений в секунду D оценить как $D\approx 2W$, то $E_N = \frac{P_s}{D} = \frac{P_s}{2W}$. Отсюда имеем другую известную формулу из теоремы Шеннона для пропускной способности:

$$R < 2WC_N = W\log_2\left(1+\frac{P_s}{WN_0}\right) = C \tag{114}$$

Здесь C - пропускная способность канала в битах на единицу времени (бит/с).

Понятие пропускной способности канала является основой современной теории связи. До работ Шеннона считалось, что при наличии шума в канале и фиксированной скорости источника надёжность передачи не может превзойти некоторый предел. Теорема о пропускной способности утверждает, что шумы, мощность сигнала и полоса частот ограничивают лишь скорость надёжной передачи, но возможность надёжной передачи существует всегда.

Модификации этой теоремы верны и для других математических моделей каналов, кроме канала с аддитивным белым гауссовским шумом. Ещё более важно то, что в реальных каналах также наблюдаются явления, согласующиеся с тем, что надёжная связь возможна, но её скорость не может превзойти некоторый предел. Добавим, что достигнутый на практике предел уже не так далёк от $\hat{R}_0$ (максимальной скорости блочной передачи).

17. Минимаксный приём для полностью симметричных систем сигналов и двоичный симметричный канал без памяти

Как мы видели выше, во многих случаях вычисление вероятности ошибки упрощается из-за полной симметрии геометрической конфигурации множества $\vec{s}_i$. Под полной симметрией мы понимаем свойство, состоящее в том, что любая перенумерация сигнальных точек может быть осуществлена поворотом, сдвигом или инверсией осей координат.

При наличии полной симметрии условие равновероятности сообщений $\Pr\{m_i\} = 1/M \quad \forall i$ приводит к конгруэнтности областей решения и, таким образом, к тому, что условная вероятность правильного приема не зависит от передаваемого сигнала: $\Pr\{\hat{m} = m \mid m = m_j\} = P^* \quad \forall i$. Если такой приемник с конгруэнтными областями решения используется для случая, когда $\Pr\{m_i\}$ не равны, то результирующая безусловная вероятность правильного приема есть

$$\Pr\{\hat{m} = m\} = \sum_{j=1}^{M} \Pr\{m_j\} \Pr\{\hat{m} = m \mid m = m_j\} = P^* \tag{115}$$

то есть такая же, как и в случае равновероятных сообщений. Это свойство называется **инвариантностью данного приемника относительно статистики источника**. Если статистика источника известна, то помехоустойчивость может быть еще увеличена за счет оптимального выбора областей. Но если она не известна, а система сигналов полностью симметрична, то построив инвариантный относительно статистики источника приемник, мы будем знать вероятность ошибки независимо от этой статистики.

В результате можно сделать вывод о том, что приемник, оптимальный для равновероятных сигналов, является **минимаксным** для полностью симметричных систем сигналов. При фиксированном передатчике и канале, для приемника вероятность ошибки зависит от статистики источника и при некотором выборе этой статистики достигает максимальной величины (наихудший случай). Это максимальное значение $\Pr\{\hat{m} \neq m\}$ является удобным критерием качества работы приемника при отсутствии априорных сведений о вероятностях $\Pr\{m_i\}$, так как это гарантированный минимум помехоустойчивости. При таком критерии приемник, для которого максимальная величина $\Pr\{\hat{m} \neq m\}$ оказывается наименьшей среди всех возможных приемников, называется минимаксным приемником.

Доказать, что приемник, оптимальный для равновероятных сигналов, - минимаксный для полностью симметричной системы сигналов, не трудно. Во-первых, вероятность ошибки такого приёмника одинакова для всех статистик источника, то есть она и есть наихудшая его вероятность ошибки. Во-вторых, поскольку он оптимален для равновероятных сообщений, то любой другой приемник даст на равновероятных сообщениях большую вероятность ошибки и, значит, имеет ещё большую наихудшую вероятность ошибки, что и требовалось доказать.

Давайте обсудим, как работает такой минимаксный приёмник для системы сигналов в вершинах гиперкуба. Он предполагает, что все вершины используемого N-мерного гиперкуба равновероятны. Именно такой случай мы рассматривали в параграфе

9, и нашли, что вероятность правильного приёма всего N-мерного сигнала есть произведение вероятностей правильного приёма по каждому измерению (см. формулу (74)), так что приём по каждому измерению независим от приёма по другим измерениям. Часть сигнала, которая соответствует одному измерению, мы назвали элементом сигнала, так что отмеченную выше особенность можно сформулировать так: минимаксный приёмник для данной ситуации может быть построен как **приёмник с поэлементным приёмом** (также говорят **с покомпонентным приёмом**).

В отличие от минимаксного покомпонентного приёма, оптимальный приём в общем случае может осуществляться только как **приём в целом,** то есть решение о том, какое сообщение передавалось, принимается после приёма всех компонентов сложного сигнала, как единого целого. С точки зрения теоретической помехоустойчивости приём в целом, конечно, лучше покомпонентного. Но у покомпонентных приёмников есть другое очень важное для практики преимущество – они гораздо проще в реализации, и потому в реальных системах телекоммуникаций общего пользования, где простота, а значит и стоимость, очень важны, они используются гораздо чаще настоящих оптимальных приёмников.

Так как минимаксный приём является в случае равновероятных сигналов оптимальным, для него верны все результаты относительно возможности надёжной передачи, то есть доказано, что коды, обеспечивающие надёжную передачу существуют, но их надо как-то найти среди всего ансамбля кодов. Этими вопросами занимается специальная дисциплина – **теория кодирования**, к начальному знакомству с которой мы далее и переходим. При этом, предположение об использовании минимаксного приёма в ситуации, когда элементами сложного сигнала являются бинарные сигналы, и потому множество возможных сложных сигналов является системой сигналов в вершинах гиперкуба, позволяет абстрагироваться не только от формы элементарных сигналов, но от понятия передачи сигналов вообще, перейдя к рассмотрению **канала передачи символов,** с которым, как правило, и имеет дело элементарная теория кодирования. Действительно, вместо передачи в интервале τ одного из пары противоположных сигналов, можно говорить о передаче за время τ одного двоичного символа (0 или 1) – такие каналы передачи символов принято называть **двоичными каналами**. Вероятность ошибки при оптимальном приёме элементарного сигнала можно назвать вероятностью искажения символа, при котором вместо 0 принимается 1, или вместо 1 принимается 0. Так как условные вероятности ошибки для обоих возможных сигналов в канале связи равны, то и вероятности искажений 0 в 1 и 1 в 0 в соответствующем ему канале передачи символов тоже будут одинаковыми – такие каналы принято называть **симметричными**. И, наконец, так как приём каждого элементарного сигнала после канала связи осуществляется минимаксным приёмником независимо от других элементарных сигналов, то в соответствующей модели канала передачи символов искажения при прохождении каждого символа будут независимыми в совокупности событиями – такие каналы принято называть **каналами без памяти**.

Таким образом, задача поиска кода (как множества сигналов), обеспечивающего надёжную передачу для канала с аддитивным белым гауссовским шумом и минимаксным приёмником, эквивалента задаче поиска кода (как множества двоичных слов), обеспечивающего надёжную передачу для **двоичного симметричного канала без памяти.** Такой канал – один из простейших, с которыми работает теория кодирования, и в нашем рассмотрении мы ограничимся именно им. Другие модели систем связи порождают другие модели каналов передачи символов, для которых теория кодирования тоже получила множество важных результатов, но они находятся за пределами нашего курса.

18. Задачи и основные понятия алгебраической теории кодирования

Итак, модель системы связи, с которой имеет дело теория кодирования, изображена на рис. 18.1. Будем считать, что от источника сообщений поступает поток двоичных символов, которые поделены на блоки по k бит. Кодер преобразует эти блоки в блоки по N бит, которые будем называть **кодовыми словами,** а их совокупность – **кодом**. Параметры N и k – важнейшие, поэтому код с такими параметрами называют (N,k)-кодом. После прохождения канала передачи символов, где некоторые биты передаваемого блока могут исказиться, слово поступает в декодер, который должен из него восстановить исходный блок из k бит и передать его получателю. Нетрудно видеть, что условие возможности надёжной передачи, полученное в параграфе 15, в данном случае имеет вид

$$\frac{k}{N} = R_N < R_0 \qquad (116)$$

Как мы помним, R_0, определённое в (109), всегда меньше единицы и стремиться к ней при стремлении отношения сигнал/шум к бесконечности. Отношение k/N, равное количеству бит источника, приходящихся на один бит кодового слова, в теории кодирования называют **скоростью кода.**

Теперь рассмотрим, как в системе связи с двоичным симметричным каналом без памяти выглядит работа минимаксного приёмника. Вместо точек в пространстве сигналов у нас теперь двоичные последовательности. Вспомним, что точки были вершинами гиперкуба, и квадрат расстояния между ними был пропорционален количеству различающихся координат (см. (104)). Очевидно, что количество позиций, в которых два двоичных слова отличаются друг от друга – это мера расстояния, в данном случае полностью эквивалентная расстоянию в пространстве сигналов. Такое расстояние называют **расстоянием Хэмминга**. В терминах системы связи с каналом передачи символов, алгоритм минимаксного (и оптимального для равновероятных кодовых слов) приёмника можно описать следующим образом: принимая из канала какое-либо слово, приёмник находит расстояние Хэмминга от него до каждого кодового слова, и принимает решение, что передавалось то кодовое слово, которое ближе к принятому.

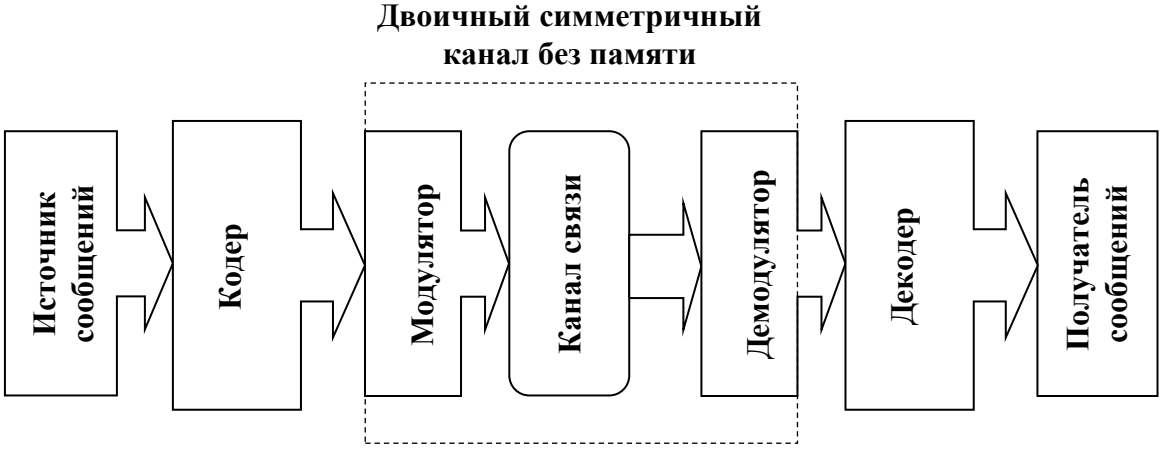

Рисунок 18.1. Система связи с каналом передачи символов

Введём важные для дальнейшего изложения понятия обнаружения и исправления ошибок. Если приёмник, декодирующий какой-либо код, может определить, что принятое из канала слово не является кодовым, это значит, что он обнаружил ошибку. При этом он может быть неспособен определить, какие именно символы исказились. Тогда говорят о **способности кода обнаруживать ошибки**. Если же приёмник может определить, на каких именно местах стоят искажённые символы, то такие ошибки легко исправляются заменой искаженных символов на противоположные. Тогда говорят о **способности кода**

исправлять ошибки. С точки зрения минимаксного подхода нас интересует такая характеристика кода, как кратность ошибки (количество искажённых символов), которую код всегда обнаруживает, и кратность ошибки, которую код всегда исправляет.

Если ввести понятие **кодового расстояния** d, как минимального расстояния Хэмминга между словами данного кода, то, принимая во внимание указанный выше алгоритм работы приёмника, нетрудно видеть, что код обнаруживает любую ошибку кратности $q_d<d$, и исправляет любую ошибку кратности $q_c<d/2$. Действительно, если кодовое расстояние d, то невозможно перевести одно кодовое слово в другое, изменив в нём $q<d$ символов. Аналогично, поменяв в слове $q<d/2$ символов, получим слово, по расстоянию Хэмминга всё ещё более близкое к исходному кодовому слову, чем к любому другому слову этого кода.

Тогда задачу поиска наилучшего в смысле обнаружения и исправления ошибок кода в самом простом виде можно сформулировать так**:** при заданных N и k найти код с наибольшим d. Но для практики такой формулировки недостаточно. Для практической полезности кода надо, чтобы наряду с принципиальной возможностью обнаруживать или исправлять ошибки, для него существовали бы эффективные алгоритмы кодирования и декодирования. Поясним необходимость этого на следующем примере**:** если единственный алгоритм кодирования, который известен для кода – это табличный, то есть поиск по кодируемому слову соответствующего ему кодового в таблице, то размер этой таблицы, скажем для k=100 и N=110, будет $2^kN\approx10^{30}$ бит. Это превышает не только доступные объёмы памяти даже для суперкомпьютеров, но и объёмы, которые могут быть достигнуты в обозримом будущем. Для декодирования поиском по таблице её размеры будут ещё больше.

Поэтому, основная задача теории кодирования – это не просто поиск кодов с хорошими корректирующими свойствами, а поиск кодов с хорошими корректирующими свойствами, имеющих эффективные алгоритмы кодирования и декодирования (как в смысле используемых ресурсов, так и в смысле временных затрат).

На каких же принципах можно строить такие коды? Рассмотрим один из самых простых , и в тоже время один из самых важных способов – так называемые проверки на чётность.

19. Блочные коды с проверкой на чётность

Приведём два простых примера для иллюстрации основной идеи таких кодов:

1). **Код контроля чётности.** Добавим к каждому слову от источника один бит, и пусть его значение будет равно сумме по модулю два значений остальных k бит. Тогда при отсутствии ошибок сумма по модулю два (k+1) бит каждого кодового слова равна 0. А если она равна единице, то значит, нечётное число символов искажено. То есть, такой код обнаруживает любую одиночную ошибку, и вообще, любую ошибку нечётной кратности. Но исправить он не может ни одной ошибки (его кодовое расстояние равно 2, и более того, любая ошибка чётной кратности переведёт одно кодовое слово данного кода в другое).

2). **Двойная проверка на чётность.** Пусть k - размер слова, получаемого от источника, таков, что $k=m*n$, где m,n – целые числа. Запишем эти k битов в виде матрицы m на n, и к каждой строке и к каждому столбцу добавим по одному биту, так что размер матрицы станет m+1 на n+1. В добавленные биты мы будем записывать результаты контроля чётности соответствующей строки или столбца.

Нетрудно понять, что данный код исправляет любую одиночную ошибку – ошибочный символ находится на пересечении той строки и того столбца, у которых не прошла проверка на чётность. Но двойную ошибку этот код исправить не может, только обнаруживает (его кодовое расстояние равно 3).

Итак, можно сформулировать основную идею построения кодов с проверкой на чётность следующим образом**:** к информационным битам, полученным от источника, надо добавить биты, в которых записывать результаты сложения по модулю два различных подмножеств множества информационных бит. Для математического исследования таких кодов удобно сформулировать их определение в матричном виде (и несколько более общем виде), к чему мы сейчас и перейдём.

Определение. Пусть $\vec{u} = (u_1,...,u_k)$ - последовательность (вектор-строка) k двоичных символов, полученных от источника, а $\vec{x} = (x_1,...,x_N)$ - соответствующее двоичное кодовое слово (вектор-строка). **Кодом с проверкой на чётность** называется код, у которого символы кодового слова получаются из символов источника следующим образом**:**

$$x_n = \sum_{i=1}^{k} g_{i,n} u_i \quad n = 1,..., N, \tag{117}$$

где $\{g_{i,n}\}_{i=1,...k,\ n=1,...N}$ - некоторое множество двоичных чисел, а суммирование понимается как суммирование по модулю два (здесь и далее в параграфах, посвящённых теории кодирования, суммирование, в том числе в матричных операциях, будет пониматься по модулю 2, если не указано обратное).

Удобно записывать (117) в матричном виде:

$$\vec{x} = \vec{u}G, \tag{118}$$

где $G = \begin{pmatrix} g_{1,1} & \cdots & g_{1,N} \\ \vdots & \ddots & \vdots \\ g_{k,1} & \cdots & g_{k,N} \end{pmatrix}$ называется **порождающей матрицей** данного кода.

Смысл элементов порождающей матрицы понятен – каждый её столбец соответствует одной проверке на чётность, если в *i*-той стоит единица, то *i*-тый символ источника в этой проверке участвует, если 0 – то нет.

Сформулируем два важных свойства кодов с проверкой на чётность, которые следуют прямо из их определения.

Свойство 1: Пусть $\vec{u}_1$ и $\vec{u}_2$ - две произвольные последовательности, полученные от источника, и пусть $\vec{u} = \vec{u}_1 \oplus \vec{u}_2$. Тогда $\vec{x} = \vec{u}G = (\vec{u}_1 \oplus \vec{u}_2)G = \vec{x}_1 \oplus \vec{x}_2$. То есть, кодовое слово, соответствующее сумме слов источника, есть сумма кодовых слов слагаемых. Это свойство является следствием того, что свойства операций матричной алгебры для двоичных чисел такие же, как у обычной матричной алгебры, в частности, умножение на двоичную матрицу является линейной операцией.

Свойство 2: Пусть вектор $\vec{g}_i = (g_{i,1},..., g_{i,N})$ - *i*-тая строка матрицы G (вектор-строка), тогда все кодовые слова $\vec{x} = \vec{u}G = \sum_{i=1}^{k} u_i \vec{g}_i$ образуют подпространство, натянутое на $\vec{g}_i$, $i = 1,...,k$.

Из этих свойств следуют два важных для нас следствия:

Следствие 1 (*способ поиска кодового расстояния*)**:** кодовое расстояние кода с проверкой на четность равно минимальному весу ненулевого кодового слова этого кода (**вес** двоичного слова – это количество единиц в нём).

Доказательство: если сложить два любых кодовых слова, то в слове-результате единицы будут стоять на тех местах, в которых кодовые слова – слагаемые различны, то есть расстояние Хэмминга между слагаемыми – это вес их суммы. Одновременно, по свойству 1, слово-результат – это всегда кодовое слово (соответствующее сумме слов источника). Поэтому минимальный вес ненулевого кодового слова – это кодовое расстояние. ■

Данное свойство позволяет более эффективно искать кодовое расстояние – мы можем перебирать кодовые слова, а не пары кодовых слов. Заметим, что из свойства 2 следует, что перебор кодовых слов можно осуществить, как перебор всех линейных комбинаций (с двоичными коэффициентами) строк порождающей матрицы.

Следствие 2 (*существование проверочной матрицы*)**:** если код таков, что каждому слову источника соответствует своё кодовое слово (т.е. нет слов источника, которые отображались бы в одно кодовое слово), то для такого кода существует двоичная матрица H размером N на N-k ранга N-k, такая что

$$GH=0 \tag{119}$$

Она называется **проверочной матрицей** кода.

Доказательство: Если все кодовые слова различны, то их 2^k штук. В пространстве точек с двоичными координатами в каждом измерении имеется 2 точки, то есть, пространство кодовых слов, натянутое на k строк порождающей матрицы, должно быть k-мерным, а это означает, что все строки порождающей матрицы должны быть линейно независимы. В N-мерном пространстве остаётся ещё N-k измерений, принадлежащих подпространству, ортогональному подпространству кодовых слов. Если выбрать в нём какой-либо базис, то каждый вектор этого базиса будет ортогонален любому вектору из пространства кодовых слов, в том числе и всем строкам порождающей матрицы. Составим из векторов (как векторов-столбцов) этого базиса матрицу H, тогда для неё будет выполняться (119). ■

Заметим, что требование взаимной однозначности слов источника и кодовых слов используемого кода есть, очевидно, обязательное требование к кодам, обеспечивающим надёжную передачу. Мы будем рассматривать только такие коды, и проверочная матрица у них будет всегда существовать. Это важное обстоятельство, так как проверочная матрица является полезным средством для декодирования (как при обнаружении ошибок, так и при их исправлении).

Свойство проверочной матрицы*:* слово $\vec{x}$ является кодовым тогда и только тогда, когда

$$\vec{x}H = 0 \tag{120}$$

Доказательство: Если $\vec{x}$ - кодовое слово, то по свойству 2 это линейная комбинация строк порождающей матрицы, и справедливость (120) следует из (119). Если (120) справедливо, то $\vec{x}$ принадлежит подпространству кодовых слов, в котором любая точка – это кодовое слово, то есть $\vec{x}$ - кодовое слово. ■

Данное свойство позволяет использовать H для обнаружения ошибок – принятое из канала слово надо умножить на H, и если получится ненулевой вектор, то значит, ошибки есть. Вектор, полученный умножением принятого из канала слова на проверочную матрицу называется **синдромом** данного слова. Для решения вопроса о том, как использовать H для исправления ошибок, сформулируем следующую теорему:

Теорема (*о синдромном декодировании кодов с проверкой на чётность*)**:**
Пусть код с проверкой на чётность используется в двоичном симметричном канале без памяти с параметром $p<1/2$. Пусть H – проверочная матрица этого кода. Тогда декодирование по методу максимального правдоподобия принятого из канала слова $\vec{y}$ можно произвести следующим образом:

1) Находим синдром принятого слова $\vec{s} = \vec{y}H$
2) Среди всех решений системы уравнений $\vec{s} = \vec{z}H$ относительно $\vec{z}$ находим решение с минимальным весом $\vec{z}_{\min}$.
3) Исправляем ошибки в принятом слове, полагая $\vec{x} = \vec{y} \oplus \vec{z}_{\min}$

Доказательство: искажение передаваемого по каналу кодового слова можно представить, как прибавление к нему вектора, в котором на искажённых позициях стоят единицы, а другие компоненты равны 0. Такой вектор называется **вектором ошибок.**

Пусть в канал передавалось слово $\vec{x}$, к нему добавился вектор ошибок $\vec{z}$, и из канала мы приняли вектор $\vec{y} = \vec{x} \oplus \vec{z}$. Тогда для синдрома имеем:

$$\vec{s} = \vec{y}H = (\vec{x} + \vec{z})H = \vec{x}H + \vec{z}H = \vec{z}H \quad (121)$$

То есть, в синдром вносит вклад только вектор ошибок. Система (121) недоопределённая, и имеет столько решений, сколько в коде есть кодовых слов. Это понятно, если принять во внимание тот факт, что любое кодовое слово можно исказить так, чтобы получилось пришедшее из канала $\vec{y}$. Какое же из этих решений должно быть выбрано? Если переданное слово $\vec{x}_i$ и полученное из канала слово $\vec{y}$ отличаются в n позициях, то для двоичного симметричного канала без памяти

$$\Pr\{\vec{y} \mid \vec{x}_i\} = p^n (1-p)^{N-n} \quad (122)$$

При $p<1/2$ это убывающая функция n, то есть расстояния Хэмминга. Поэтому декодирование по максимуму правдоподобия, как уже говорилось ранее, заключается в поиске ближайшего к $\vec{y}$ кодового слова $\vec{x}_i$, а так как это расстояние – это вес вектора ошибок, то мы должны найти решение (121) с минимальным весом. Замечая, что коррекция ошибок может быть произведена повторным прибавлением вектора ошибок к принятому слову (первое прибавление произошло в канале), заканчиваем доказательство.■

Доказанная теорема оставляет нерешённым вопрос о том, как искать решение с минимальным весом в процессе декодирования. Один из практических путей решения этой проблемы заключается в построении **таблицы синдромного декодирования**. Состоит он в том, что мы заранее составляем таблицу, куда в левый столбец записываем все возможные значения синдрома (то есть, все двоичные последовательности длины N-k), а затем начинаем перебирать все вектора ошибок в порядке возрастания их веса. То есть, сначала перебираем все вектора единичного веса, потом веса 2 и т.д. Взяв очередной вектор ошибок, домножаем его на H, и получаем синдром. По синдрому находим строку таблицы, и смотрим, заполнен ли правый столбец - если нет, заполняем его этим вектором ошибок. Если уже заполнен, но в таблице ещё остались незаполненные строки, переходим к следующему вектору ошибок.

При декодировании принимаемого слова, оно домножается на H, потом в таблице ищется строка, в которой получившийся синдром стоит в левом столбце, берётся из этой строки стоящий в правом столбце вектор ошибок и прибавляется к принятому слову.

Нетрудно видеть, что вектора ошибок, попавшие в таблицу – это те комбинации ошибок, которые код исправляет. Если случится комбинация ошибок, не вошедшая в таблицу, то на приёме она будет исправлена неверно (произойдёт ошибка приёма). Так как вероятность любого вектора ошибок, попавшего в таблицу, легко считается, то, если вычесть из единицы все эти вероятности (и вероятность того, что ошибок вообще не было), мы получим вероятность ошибки при синдромном декодировании данного кода с исправлением ошибок.

20. Систематические блочные коды с проверкой на чётность

В общем случае для кодов с проверкой на чётность остаются нерешёнными два практически важных вопроса: как найти проверочную матрицу и как после исправления ошибок получить закодированное в нём слово источника. Для ответа на первый вопрос приходится искать нужное число линейно-независимых решений системы (119), а для ответа на второй – решать систему (117) относительно u_i. Но для одного подкласса кодов с проверкой на чётность эти вопросы решаются чрезвычайно легко, и этот подкласс мы сейчас рассмотрим.

Определение. Систематическим кодом с проверкой на чётность называется код с проверкой на чётность, который задаётся следующеё системой уравнений**:**

$$\begin{cases} x_n = u_n \quad n = 1,\ldots,k \\ x_n = \sum_{i=1}^{k} g_{i,n} u_i \quad n = k+1,\ldots,N \end{cases} \tag{123}$$

Из определения видно, что порождающая матрица систематического кода имеет вид:

$$G = \begin{pmatrix} 1 & 0 & \cdots & 0 & g_{1,k+1} & \cdots & g_{1,N} \\ 0 & 1 & \cdots & 0 & & & \\ \vdots & & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & g_{k,k+1} & \cdots & g_{k,N} \end{pmatrix} \tag{124}$$

А кодовое слово состоит из исходного слова источника и приписанных к нему сзади проверочных символов. Поэтому при отсутствии ошибок выделение из кодового слова закодированного в нём слова источника тривиально.

Оказывается, что и проверочную матрицу систематического кода искать очень просто. Докажем, что матрица

$$H = \begin{pmatrix} g_{1,k+1} & & \cdots & g_{1,N} \\ \vdots & & & \vdots \\ g_{k,k+1} & & \cdots & g_{k,N} \\ 1 & 0 & \cdots & 0 \\ 0 & 1 & & \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}, \tag{125}$$

у которой вверху стоит задняя часть матрицы G (размером k на N-k), а внизу дописана единичная матрица размером N-k на N-k, является проверочной матрицей кода, задаваемого системой (123) и порождающей матрицей (124). Для этого в системе (123), задающей связь между символами кодового слова, подставим k первых уравнений в остальные N-k.

$$x_n = \sum_{i=1}^{k} g_{i,n} x_i \quad n = k+1,\ldots,N \tag{126}$$

Прибавим по модулю два к обоим частям каждого уравнения в (126) соответствующее x_n:

$$x_n \oplus x_n = 0 = \sum_{i=1}^{k} g_{i,n} x_i \oplus x_n \quad n = k+1,\ldots,N. \tag{127}$$

Сравнив эти выражения с результатом умножения $\vec{x} = (x_1,\ldots,x_N)$ на матрицу H из (125), обнаруживаем, что для каждого кодового слова $\vec{x}$ выполняется $\vec{x}H = 0$.

В обратную сторону, если какое-то $\vec{x} = (x_1,\ldots,x_N)$ при умножении на H из (125) даёт 0, то значит, его компоненты удовлетворяют системе (123), то есть, это $\vec{x}$ - кодовое слово данного кода. Согласно свойству проверочной матрицы, H – проверочная матрица. ■

21. Коды Хэмминга

С помощью введённого выше математического аппарата, рассмотрим одно из самых знаменитых семейств кодов с проверкой на чётность – кодов Хэмминга. Для иллюстрации основной идеи этого кода, заметим, что если при вычислении синдрома умножаемый на проверочную матрицу вектор ошибок имеет вес 1 (то есть содержит 1 только в одной позиции), то результатом этого умножения будет соответствующая строка проверочной матрицы (номер которой равен номеру позиции 1 в векторе ошибок).

Отсюда следует, что если у проверочной матрицы все строки различны, то все вектора ошибок единичного веса будут иметь разные синдромы и будут исправляться

таким кодом. Для (N,k) систематического кода в кодовом слове N-k проверочных символов, то есть у синдрома есть 2^{N-k}-1 ненулевых значений, которые могут быть выбраны в качестве строк проверочной матрицы. Поэтому если $N \le 2^{N-k}$-1, то все строки проверочной матрицы могут быть ненулевыми и различными.

Коды Хэмминга – это коды, у которых $N=2^{N-k}$-1, и проверочная матрица содержит все ненулевые строки длины N-k. Семейство кодов Хэмминга – бесконечное, вот примеры подходящих пар (N,k)**:** (3, 1), (7, 4), (15, 11), (31, 26), … .

Построение кода Хэмминга удобно начинать именно с проверочной матрицы. Сначала выписываются нижние строки, образующие единичную матрицу размером N-k на N-k, а потом сверху к ним приписываются остальные ненулевые строки в произвольном порядке (переставление местами этих строк ведёт только к переставлению местами проверочных символов в кодовом слове). Порождающая матрица кода Хэмминга строится по проверочной.

Так как таблица синдромного декодирования кода Хэмминга не содержит ничего, кроме вектора нулевого веса и N векторов единичного веса, то легко видеть, что любое N-значное двоичное слово – это либо кодовое слово, либо лежит на расстоянии 1 от какого-либо кодового. То есть, всё пространство N-значных двоичных слов код делит на N-мерные сферы радиуса 1, в центре каждой из которых находится кодовое слово. Вне этих сфер нет ни одной точки. Такой код называется **сферически упакованным**.

Понятно, что код исправляет любые единичные ошибки и больше никаких ошибок не исправляет и не обнаруживает. Нетрудно найти, что его кодовое расстояние равно 3. Несколько менее тривиально заметить, что если для данных N и k существует код Хэмминга, то это код с максимально возможным для данных параметров кодовым расстоянием. Это следует из сферической упакованности кода – попытка сдвинуть любое кодовое слово со своего места приведёт к “деформации” окружающей его сферы, и расстояние до центра какой-либо из соседних сфер уменьшится.

Также код Хэмминга является одним из трёх известных семейств **совершенных кодов**.

22. Циклические коды

Синдромное декодирование при помощи таблицы синдромов является простым, и при небольших значениях N и k эффективным методом декодирования с исправлением ошибок для кодов с проверкой на чётность. К сожалению, размер таблицы синдромного декодирования экспоненциально растёт (как 2^{N-k}) с ростом N- k. Есть, однако, подкласс кодов с проверкой на чётность, для которых существует другой способ декодирования, сложность которого линейно зависит от N- k. С этими очень популярными на практике кодами мы познакомимся в этом параграфе.

Определение. Код называется **циклическим,** если любая циклическая перестановка символов в любом его кодовом слове также является кодовым словом.

В теории циклических кодов принято представлять двоичные слова в виде полиномов с двоичными коэффициентами. Так, слову $\vec{v} = (v_0,...,v_{N-1})$ соответствует полином $v(X) = v_0 + v_1 X + v_2 X^2 + ... + v_{N-1} X^{N-1}$. Такое представление удобно, так как многие операции, которые производятся при кодировании – декодировании циклических кодов удобно представляются в виде действий над полиномами. Например, циклическая перестановка коэффициентов полинома, может быть представлена, как домножение его на X и взятие по модулю X^N+1. Действительно,

$$\begin{array}{l|l} v_{N-1}X^N + v_{N-2}X^{N-1} + ... + v_0 X & \\ \hline v_{N-1}X^N + \quad .... \quad + v_{N-1} & \dfrac{X^N+1}{v_{N-1}} \\ \hline v_{N-2}X^{N-1} + \quad ... \quad + v_0 X + v_{N-1} & \end{array}$$

где полином – остаток от деления соответствует последовательности $(v_{N-1}, v_0, ..., v_{N-2})$ с циклическим сдвигом относительно исходной (при получении этого результата использовалось то, что вычитание по модулю 2 равносильно сложению по модулю 2).

Циклический код с параметрами (*N*,*k*) задаётся **производящим полиномом** *g(X)*, степени *N*-*k*, который должен быть делителем полинома X^N+1. То есть

$$X^N + 1 = g(X)h(X), \tag{128}$$

где *h(X)* – полином степени *k*, называемый **проверочным полиномом** этого кода.

Строки порождающей матрицы циклического кода являются последовательностями коэффициентов полиномов *g(X)*, *Xg(X)*, $X^2g(X), ..., X^{k-1}g(X)$, дополненными нулями до нужного размера. То есть, обозначив g_i коэффициент при *i*-той степени многочлена *g(X)*, получим, что порождающая матрица выглядит следующим образом:

$$G = \begin{pmatrix} g_0 & g_1 & \cdots & g_{N-k} & 0 & \cdots & 0 \\ 0 & g_0 & \cdots & g_{N-k-1} & g_{N-k} & \cdots & 0 \\ \vdots & & \ddots & & & \ddots & \vdots \\ 0 & 0 & & \cdots & & g_{N-k-1} & g_{N-k} \end{pmatrix}$$

Если от источника получено слово $\vec{u} = (u_0, ..., u_{k-1})$, которое представляется в виде полинома $u(X) = u_0 + u_1X + u_2X^2 + ... + u_{k-1}X^{k-1}$, то нетрудно видеть, что результатом умножения $\vec{u}G$ будет вектор-строка $(u_0g_0, u_0g_1 + u_1g_0, ..., u_{k-1}g_{N-k})$, то есть последовательность коэффициентов полинома *u(X)g(X)*. Так что процесс кодирования циклическим кодом можно описать, как умножение полинома, полученного от источника, на производящий полином кода:

$$x(X) = u(X)g(X) \tag{129}$$

Проверочная матрица циклического кода строится с помощью проверочного полинома следующим образом:

$$H = \begin{pmatrix} 0 & 0 & \cdots & h_k \\ \vdots & \vdots & & h_{k-1} \\ 0 & 0 & & \vdots \\ 0 & h_k & & \\ h_k & h_{k-1} & \iddots & \\ \vdots & & & \\ h_1 & h_0 & & 0 \\ h_0 & 0 & \cdots & 0 \end{pmatrix} \tag{130}$$

В том, что для данных *G* и *H* выполняется (119), легко убедиться, если заметить, что элементы матрицы-результата – это коэффициенты полинома X^N+1 при степенях от 1 до *N*-1, а они все нули (например, в первой строке и первом столбце матрицы-результата будет стоять элемент, равный $g_{N-k-1}h_k + g_{N-k}h_{k-1}$, то есть, коэффициент при степени *N*-1) .

Таким образом, проверка на наличие ошибок тоже может быть представлена в виде действий с полиномами. Полученный из канала полином домножается на проверочный полином, и результат берётся по модулю X^N+1. Если остаток от деления на X^N+1 есть ноль, то ошибок нет, и частное от деления – декодированный полином, полученный от источника. Понятно, что вместо умножения на *h(X)* и деления на X^N+1 можно просто делить на *g(X)*.

Среди циклических кодов попадаются коды с самыми разными свойствами, от “очень хороших” до “очень плохих”. Например, рассмотрим циклический (7,4) код с

производящим полиномом $g(X)=1+X+X^3$, и проверочным полиномом, соответственно, $h(X)=1+X+X^2+X^4$. Соответствующие матрицы выглядят так:

$$G=\begin{pmatrix}1&1&0&1&0&0&0\\0&1&1&0&1&0&0\\0&0&1&1&0&1&0\\0&0&0&1&1&0&1\end{pmatrix}\quad H=\begin{pmatrix}0&0&1\\0&1&0\\1&0&1\\0&1&1\\1&1&1\\1&1&0\\1&0&0\end{pmatrix} \tag{131}$$

то есть, проверочная матрица содержит все ненулевые строки длины 3, а это значит, что данный циклический код эквивалентен коду Хэмминга (но не является систематическим).

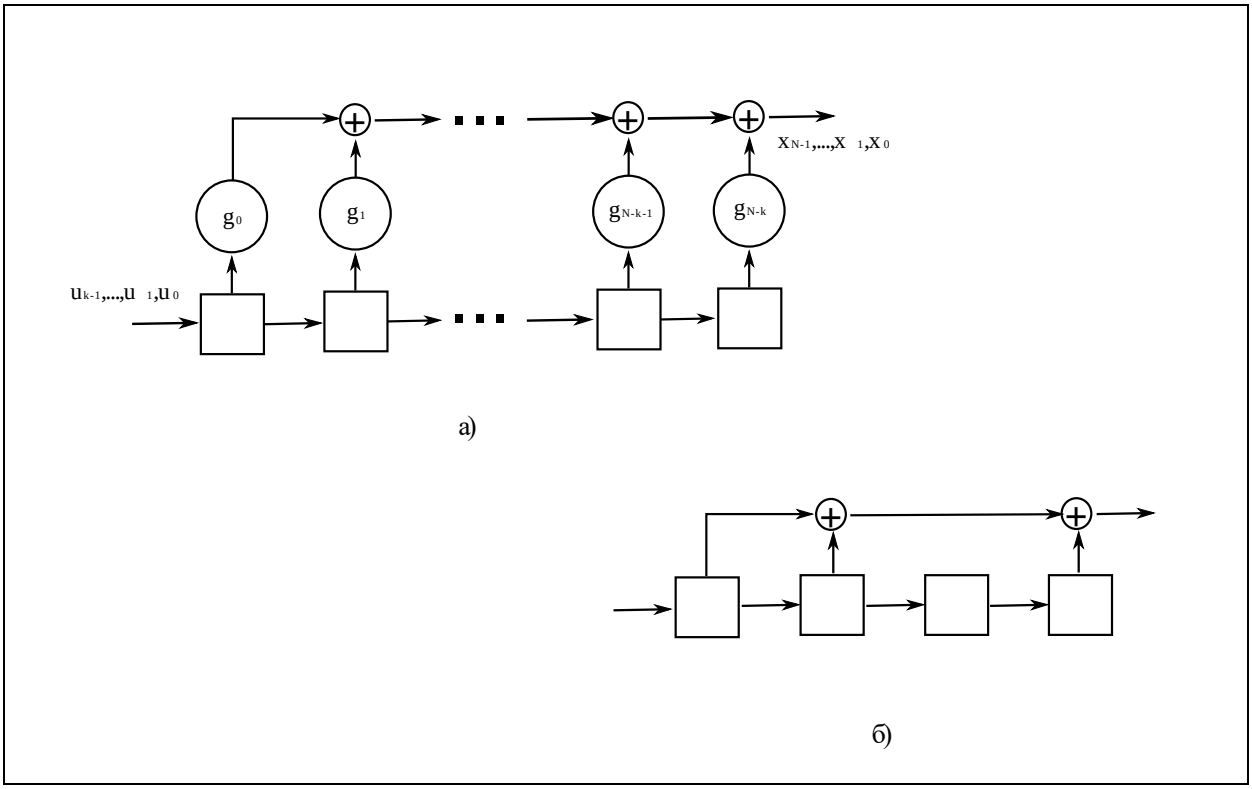

Рисунок 22.1. Реализация умножения полиномов с помощью регистров сдвига.

Для циклических кодов пригоден весь аппарат синдромного декодирования, рассмотренный ранее, но для них существуют и более эффективные методы, не требующие таблиц, размеры которых экспоненциально зависят от параметров кода. Они основаны на том, что действия над полиномами могут производить специальные устройства, называемые **регистрами сдвига**, сложность которых (количество элементов) линейно зависит от параметров кода. Так, умножение на производящий полином может быть осуществлено схемой, приведённой на рис. 22.1 а. В начальный момент в регистре сдвига все нули. На первом такте u_0 входит в первую ячейку регистра, и на выходе появляется младший коэффициент полинома $x(X)$. На N-том шаге появляется последний коэффициент (после входа в регистр u_{k-1} все остальные входные символы - нули). Для примера умножитель на производящий полином $g(X)=1+X+X^3$ изображён на рис. 22.1 б.

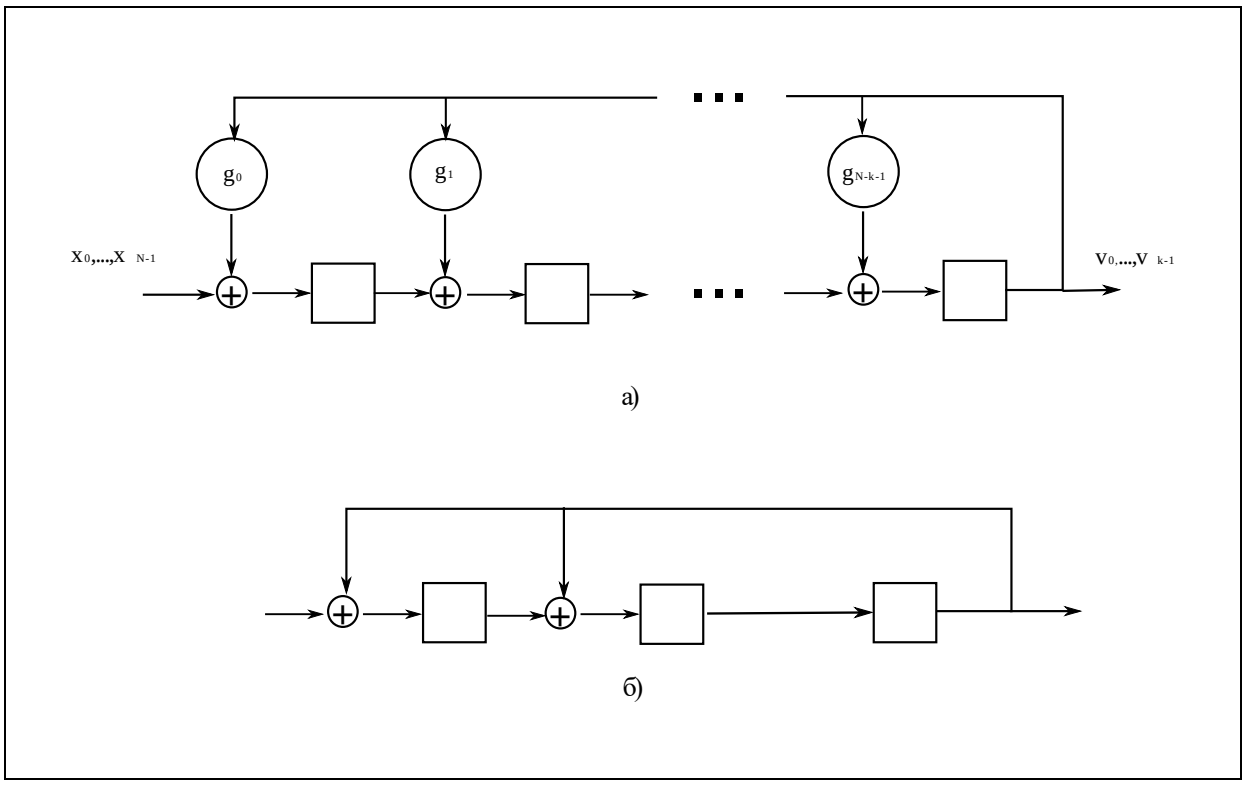

Рисунок 22.2. Реализация деления полиномов с помощью регистров сдвига.

Для деления многочленов может быть использована схема, изображённая на рис. 22.2 а. После N сдвигов на выходе регистра последовательно появятся все коэффициенты частного, а в регистре останется остаток от деления. Пример делителя на $g(X)=1+X+X^3$ изображён на рис. 22.2 б.

При использовании циклических кодов для исправления ошибок можно не хранить таблицу синдромного декодирования, а перебирать вектора ошибок в порядке возрастания их веса (в виде полиномов), и для каждого с помощью приведённой схемы быстро считать остаток от деления (вместо синдрома), пока он не совпадёт с остатком от деления принятого из канала слова.

23. Исправление нескольких ошибок, идея кодов БЧХ

До сих пор, когда мы рассматривали коды, исправляющие ошибки, мы делали это на примере кодов, исправляющих одиночные ошибки, как код Хэмминга. Теперь давайте познакомимся с основным подходом к построению кодов, исправляющих большее число ошибок.

Для этого нам понадобится вспомнить некоторые сведения из алгебры конечных полей, которая является основным математическим аппаратом, используемым для построения таких кодов.

Конечным полем (полем Галуа) называется конечное множество элементов, на котором определены две операции, которые принято называть "*сложением*" и "*умножением*". Под понятием "определены на множестве" подразумевается, что эти операции определены для любой пары элементов множества и результатом будет опять элемент этого множества (т.е. замкнутость операции на множестве). Обе эти операции должны быть ассоциативны, коммутативны, и в поле должны присутствовать нейтральные элементы обоих операций (нейтральный элемент операции – это элемент, результатом операции с которым любого другого элемента будет это же самый элемент). Для сложения нейтральный элемент принято обозначать **0**, для умножения **1**. Кроме того, обе операции должны быть обратимы, то есть для каждого элемента поля должен

существовать “обратный” элемент поля, операция с которым даст нейтральный элемент данной операции. Единственное исключение – у **0** нет обратного элемента по умножению (“деление на ноль запрещено”). И, наконец, должна быть дистрибутивность умножения относительно сложения, то есть выполняться равенство (a+b)*c=a*c+b*c для любых элементов поля (вместе с запретом деления на ноль, это второе отличие сложения от умножения).

Нетрудно видеть, что эти свойства совпадают со свойствами умножения и сложения у действительных чисел, то есть действительные числа тоже образуют поле, только бесконечное. Однако, по сравнению с полем действительных чисел, у конечных полей имеется множество совершенно особых свойств, и некоторые из них нам понадобятся.

Конечные поля существуют не для любого количества элементов в поле, а только для p^m, где p – простое число, m – натуральное. Обозначается такое поле GF(p^m). Для m=1 такое поле называется *простым*, а GF(p^m) при m>1 называется *расширением простого поля* GF(p).

Примеры:

1). GF(2) состоит из двух элементов: {**0**, **1**}. Таблицы сложения и умножения (которыми удобно задавать операции в поле), в данном случае тривиальны (это действия по модулю 2):

+	**0**	**1**
0	**0**	**1**
1	**1**	**0**

*	**0**	**1**
0	**0**	**0**
1	**0**	**1**

2). Элементы GF(4), то есть GF(2^2), будем обозначать, кроме **0** и **1,** греческими буквами β,γ. Тогда таблицы операций для такого поля могут выглядеть так:

+	0	1	β	γ
0	0	1	β	γ
1	1	0	γ	β
β	β	γ	0	1
γ	γ	β	1	0

*	0	1	β	γ
0	0	0	0	0
1	0	1	β	γ
β	0	β	γ	1
γ	0	γ	1	β

3). Элементы GF(5), будем обозначать, кроме **0** и **1,** греческими буквами β,γ,δ. Тогда таблицы операций для такого поля могут выглядеть так:

+	0	1	β	γ	δ
0	0	1	β	γ	δ
1	1	β	γ	δ	0
β	β	γ	δ	0	1
γ	γ	δ	0	1	β
δ	δ	0	1	β	γ

*	0	1	β	γ	δ
0	0	0	0	0	0
1	0	1	β	γ	δ
β	0	β	δ	1	γ
γ	0	γ	1	δ	β
δ	0	δ	γ	β	1

Пока для нас элементы конечного поля – это некие математические объекты, обладающие указанными выше свойствами. Возникает вопрос: а нельзя ли поставить в соответствие элементам конечных полей какие-то более привычные математические объекты? Или, говоря по-другому, образуют ли какие-нибудь более привычные для нас математические объекты, со своими, более привычными для нас операциями, такие конечные группы? Ответ на этот вопрос утвердительный. В случае простого поля GF(p) его элементы можно отождествить с целыми неотрицательными числами, операции с которыми проводятся по модулю p (то есть, с числами от 0 до p-1). В примере 1 это очевидно, а в примере 3 можно подставить β=2, γ=3, δ=4, и получить правильные таблицы

операций по модулю 5. А вот для расширений простых полей такое представление не имеет места. Подстановка β=2 и γ=3 в таблицы из примера 2 не сделает их таблицами операций по модулю 4. И вообще, подстановка никаких чисел не сделает их таблицами операций по модулю какого-то числа.

Однако, и для расширений простых полей представление их элементов более привычными математическими объектами существует. Только это не числа, а многочлены. Элементы поля $GF(p^m)$ могут быть отождествлены с многочленами над полем GF(p) (то есть, коэффициенты этих многочленов – это элементы GF(p), которые, как мы уже знаем, можно считать обычными числами от 0 до p-1), а степени этих многочленов должны быть меньше m. При действиях с этими многочленами, все действия с коэффициентами проводятся по правилам арифметики по модулю p. А если в результате умножения одного такого многочлена на другой в результате получится многочлен степени m или более, его нужно взять по модулю некоторого специального многочлена P(x) степени m, то есть, разделить на P(x) и взять остаток от деления. Как найти этот многочлен – вопрос особый, общих методов его поиска, кроме полного перебора, нет. Известно, однако, что он должен быть *неприводимым* в поле GF(p), то есть не раскладываться на произведение других многочленов над GF(p), в частности, не иметь корней среди элементов GF(p). Для некоторых размеров конечной группы таких многочленов может быть несколько, но хотя бы один есть всегда.

По другому то же самое можно сказать следующим образом: существует неприводимый многочлен степени m над GF(p) такой, что вычеты (остатки от деления) по модулю этого многочлена образуют конечное поле $GF(p^m)$. Если таких многочленов несколько, то существует несколько разных множеств многочленов-вычетов, каждое из которых есть $GF(p^m)$.

Особое свойство конечных полей – связь между всеми элементами поля через один элемент, называемый *примитивным* (и обычно обозначаемый α). Эта связь выражается в том, что любой ненулевой элемент $GF(p^m)$ есть какая-то степень примитивного элемента (то есть, соответствующее количество умножений этого элемента на себя). Таким образом, $1, \alpha, \alpha^2, ..., \alpha^{p^m-2}$ - это все ненулевые элементы $GF(p^m)$, а

$$\alpha^{p^m-1} = 1, \quad \alpha^{p^m} = \alpha . \qquad (132)$$

Какой именно элемент будет примитивным – зависит от P(x), вычеты которого образуют данное поле. Примитивных элементов может быть несколько, и даже все элементы поля могут быть примитивными, но хотя бы один есть всегда.

Примеры:

1). Для GF(4) поле вычетов образует многочлен $P(x)=x^2+x+1$, а соответствие ненулевых элементов поля степени примитивного элемента приведено в таблице (здесь это x, хотя и x+1 тоже примитивный):

1	α^0
x	α^1
x+1	α^2

Интересно, что здесь совершенно не важно, какой именно элемент из таблицы операций GF(8) в пункте 2 предыдущего примера отождествить с многочленами x и x+1. Можно положить β=x, γ=x+1, а можно наоборот – таблицы операций в обоих случаях будут истинными.

2). Для GF(8) поле вычетов образует, например, многочлен $P(x)=x^3+x+1$, а соответствие элементов поля степени примитивного элемента приведено в таблице (здесь это тоже x, хотя и другие элементы тоже примитивные, кроме 1):

1	α^0
x	α^1
x^2	α^2
x+1	α^3
x^2+x	α^4
x^2+x+1	α^5
x^2+1	α^6

Теперь вернёмся к задачам кодирования. Последовательности коэффициентов полиномов, являющихся элементами GF(2^m) – это двоичные вектора-строки длины m. Поэтому мы можем установить взаимно-однозначное соответствие между матрицами из элементов GF(2^m) и двоичными матрицами. Так, если в каждой строке приведённой ниже матрицы *H** заменить элемент поля GF(8) на последовательность коэффициентов полинома, соответствующего этому элементу, то получим матрицу H, являющуюся проверочной матрицей кода Хэмминга (7,4), причём даже систематического:

$$H^* = \begin{pmatrix} \alpha^6 \\ \alpha^5 \\ \alpha^4 \\ \alpha^3 \\ \alpha^2 \\ \alpha \\ 1 \end{pmatrix} \quad \Leftrightarrow \quad H = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Пусть теперь из канала связи принято двоичное слово $\vec{s} = (s_6, s_5, \ldots, s_0)$ (обратите внимание на нумерацию бит, она упростит дальнейшие выкладки). Его представление в виде многочлена запишем в виде $S(X) = s_6 X^6 + s_5 X^5 + \ldots + s_0$. Если это слово – кодовое, то есть ошибок в нём нет, то $\vec{s}H = (0,0,0)$. Обычно для краткости пишут просто ноль, но в данном случае вспомним, что это вектор-строка из трёх нулевых элементов, и каждый её элемент - это результат произведения $\vec{s}$ на соответствующий столбец *H*. А в каждом столбце *H* стоят коэффициенты при соответствующей степени многочленов, представляющих элементы GF(8). То есть, в результате этого умножения получается нулевой многочлен, который представляет нулевой элемент GF(8). Таким образом, можно перейти к описанию этой операции (умножения кодового слова на проверочную матрицу) сразу в GF(8), что даёт

$$\vec{s}H^* = s_6\alpha^6 + s_5\alpha^5 + \ldots + s_0 = 0,$$

где под нулём справа понимается нулевой элемент GF(8). А теперь посмотрим на среднюю часть этого равенства. Там стоит не что иное, как подстановка в многочлен *S(X)* вместо *X* элемента α. То есть, это равенство может быть записано, как *S*(α)=**0** в GF(8). Если же $\vec{s} = (s_6, s_5, \ldots, s_0)$ не кодовое слово и содержит одну ошибку, то уравнение будет иметь вид

$$S(\alpha) = \alpha^k,$$

где α^k – это некоторый элемент GF(8), коэффициенты представления которого в виде многочлена стоят в соответствующей строке *H*, а α^k – это его представление через степень примитивного элемента, стоящее в соответствующей строке в *H**. При этом *k* – это номер позиции, в которой произошла ошибка, если нумеровать биты $\vec{s} = (s_6, s_5, \ldots, s_0)$ справа налево, начиная с нуля (как мы и сделали).

Понятно, что так можно работать с любым GF(2^m), и кодами соответствующих параметров. То есть, для исправления одной ошибки, у каждого кодового слова, представленного в виде многочлена, должен быть один и тот же корень в GF(2^m). Этого можно добиться для циклического кода, если это будет корень порождающего многочлена, умножением на который информационных многочленов получаются кодовые многочлены. Возникает вопрос: а если у каждого кодового слова будут два общих корня (общие они опять будут потому, что эти корни порождающего многочлена), это позволит исправить две ошибки? Ответ: да, если брать "подходящие" корни. Дело в том, что корни в GF(2^m) могут существовать группами, подобно тому, как в поле комплексных чисел у многочлена x^2+1 корень i существует в паре с корнем $(-i)$. В GF(2^m) корень α всегда существует вместе с корнем α^2, и всеми чётными степенями – так что если мы используем α, то чётные степени не подойдут для борьбы с двойными ошибками, а вот корень α^3 подходит. Получим:

$$H^* = \begin{pmatrix} \alpha^6 & \alpha^4 \\ \alpha^5 & \alpha \\ \alpha^4 & \alpha^5 \\ \alpha^3 & \alpha^2 \\ \alpha^2 & \alpha^6 \\ \alpha & \alpha^3 \\ 1 & 1 \end{pmatrix} \quad \Leftrightarrow \quad H = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

где для заполнения второго столбца H^* мы использовали (132). Теперь умножение принятого из канала кодового слова $\vec{s} = (s_6, s_5, \ldots, s_0)$ на H^* можно записать как два уравнения: $S(\alpha)=0$, $S(\alpha^3)=0$. А если принятое из канала слово не кодовое, то система уравнений будет такой:

$$\begin{cases} S(\alpha) = \beta \\ S(\alpha^3) = \gamma \end{cases} \tag{133}$$

где β и γ – некоторые ненулевые элементы GF(8) (коэффициенты их представления в виде многочленов получатся при умножении принятого слова на H: первые три элемента результата будут коэффициенты β, последние три – коэффициенты γ).

Предположим, что в принятом слове произошли две ошибки. Как и ранее, можно представить это как прибавление к изначально кодовому многочлену многочлена ошибок вида X^k+X^j, где k и j – номера позиций, где произошли ошибки (как они нумеруются в $\vec{s} = (s_6, s_5, \ldots, s_0)$). Если умножить на H^* эту сумму многочленов $S(X)+ X^k+X^j$, где $S(X)$ – кодовый, и при умножении на H^* даст нули, то (133) примет вид

$$\begin{cases} \alpha^k + \alpha^j = \beta \\ \alpha^{3k} + \alpha^{3j} = \gamma \end{cases}$$

Эту систему нужно решить относительно α^k и α^j, откуда можно найти позиции ошибок и исправить их. Как и в случае систем уравнений для обычных (комплексных) чисел, у систем уравнений в конечных полях для единственного решения тоже нужно столько же уравнений, сколько неизвестных. Так что для систем приведённого выше вида алгоритмы решения существуют, но мы не будем их здесь разбирать.

Понятно, что этот подход можно распространить и на большее число ошибок. На такого рода идеях построены чрезвычайно популярные на практике коды Боуза-Чоудхури-Хоквингема (БЧХ), и их частный случай – коды Рида-Соломона (РС). Только в них приходящую из канала последовательность бит часто рассматривают как последовательность элементов $GF(2^m)$, то есть разбивают на части по m бит каждая, и считают, что это один элемент конечного поля (т.е. коэффициенты его представления в виде многочлена). Так что многочлены, с которыми работают, становятся многочленами над $GF(2^m)$ (а не над GF(2), как мы считали здесь), и блоки кода считают состоящими из элементов $GF(2^m)$. Это даёт кодам БЧХ и РС ряд дополнительных преимуществ, но их рассмотрение по сложности и необходимому времени лежит за пределами нашего курса.

24. Пачки ошибок и импульсные помехи

Итак, в циклическом коде всё зависит от порождающего полинома. Для поиска полиномов, порождающих "хорошие" коды, надо более глубоко исследовать их свойства с помощью математического аппарата теории конечных алгебр, но это находится за пределами нашего курса. В дополнение к вышеизложенному, мы приведём здесь лишь один пример свойства циклических кодов, доказать которое очень просто. А именно: к очень полезным для практики достоинствам циклических кодов относится то, что они обладают весьма хорошими свойствами в обнаружении (и исправлении) пачек ошибок.

Определение: Пачкой длины *t* называется двоичный вектор, все ненулевые компоненты которого расположены среди *t* последовательных компонент, первая и последняя из которых единичные.

Если этот вектор – вектор ошибок, то его называют **пачкой ошибок**. Особое внимание к такой конфигурации ошибок вызвано тем, что пачки ошибок являются хорошей моделью негауссовского шума. Во многих реальных каналах, особенно в современных многопользовательских системах связи, модель гауссовского шума уже не является удовлетворительной. В частности, независимость ошибок в соседних символах не имеет места, а появление ошибки в каком-то символе сильно увеличивает вероятность ошибки в следующем. Так и образуются пачки ошибок. Соответствующие модели шумов принято называть **импульсными помехами**.

Циклические коды очень хорошо зарекомендовали себя именно в условиях импульсных помех. Так, любой циклический код обнаруживает пачки ошибок длиной до *N-k*. Чтобы это доказать, заметим, что необнаруженной ошибка может пройти, только если полином, её представляющий, будет без остатка делиться на *g(X)*. Но у полинома, представляющего пачку ошибок длиной *N-k,* старшая степень на *N-k*-1 больше младшей, то есть он имеет вид $X^n e(X)$, где e(X) – полином степени *N-k*-1. Множитель X^n не может иметь общих делителей с *g(X),* так как у *g(X)* обязательно есть нулевая степень (иначе не могло бы выполняться (128)). Так что для того, чтобы $X^n e(X)$ делился на *g(X),* на *g(X)* должен делиться *e(X).* Но степень *e(X)* меньше, чем у *g(X),* и потому он на *g(X)* не делится. ■

Известны классы циклических кодов, обнаруживающие значительную долю пачек ошибок и большей, чем *N-k* длины. Есть классы циклических кодов, исправляющих пачки ошибок длиной до *(N-k)/2* (в частности, коды БЧХ и РС). А ещё, есть простые, но остроумные приёмы, широко используемые на практике, с помощью которых можно многократно увеличить длину пачки ошибок, с которой циклический код справляется (исправляет или обнаруживает). Приёмы эти сводятся к так называемому **перемежению бит**. Идея у него такая: пусть у нас есть циклический код, который справляется (обнаруживает или исправляет, в данном случае неважно) с пачкой ошибок длины до *m* включительно, а мы хотели бы его использовать в ситуации, когда длина пачки ошибок может быть до *n* включительно, где *n>m*. Для простоты, пусть *n/m=k,* где *k* – целое число. Тогда сделаем следующее – после кодера перед отправкой последовательности кодовых

слов (длины *N*) в канал, накопим в промежуточной памяти *k* таких слов, а в канал будем передавать их так – сначала все первые биты этих слов, потом подряд все вторые биты, и т.д. пока не дойдём до последних. На приёмном конце перед декодером приведём обратную операцию – *k* первых принятых из канала бит распределим по одному в каждое из *k* слов, следующие *k* бит поставим в эти слова на вторые места и т.д., пока не заполним все *N* позиций в каждом слове. И только после этого отправим слова в декодер. Тогда, как нетрудно понять, наложившаяся в канале на последовательность бит пачка ошибок, затронувшая соседние биты, после перемежения поделится на *k* частей, и в каждое кодовое слово попадёт часть не длиннее *m*, с которой наш код справляется.

25. Примеры кодов, использующихся на практике

В этом параграфе мы приведём пару примеров кодов, использующихся в реальных системах передачи информации. Один – код с обнаружением ошибок рассмотри подробнее, а другой – с исправлением ошибок – коротко, фактически просто назовём.

1). **Циклический избыточный код (CRC) сети Ethernet.** Этот код с обнаружением ошибок, предназначенный для работы в локальной сети Ethernet, с достаточно хорошими каналами связи, где вероятность ошибки мала, и в случае её обнаружения принимающая сторона имеет возможность запросить у передающей стороны повторно передать искажённое сообщение.

При его создании, постарались получить в одном коде преимущества, присущие циклическим кодам, и одновременно, преимущества, присущие систематическим кодам, и при этом решить ещё несколько специфических технических проблем. Для упрощения понимания, давайте сначала обозначим эти проблемы и подходы к их решению, а потом опишем сами процедуры кодирования и декодирования

Одна из технических проблем, которые надо было решить – это необходимость кодировать блоки переменной длины. При этом исчезает возможность использовать проверочный полином циклического кода (он свой для каждой длины кодового слова), а вместо него при декодировании используется деление на производящий полином. Производящий полином 32-ой степени стандартизован (для полноты картины, вот последовательность из 33 его коэффициентов, от 0 к 32 степени**:** 11101101101110001000001100100000 1).

Для того чтобы, как у систематических кодов, в кодовом слове начало совпадало со словом источника, но при этом, как в циклическом коде, каждое кодовое слово делилось на производящий полином, используется следующий приём: слово от источника делится на производящий полином, и находится остаток от деления. Если прибавить его к кодовому слову, то получится слово, которое уже без остатка делится на производящий полином.

Ещё одна техническая проблема, которую пришлось решать – это необходимость сделать так, чтобы в коде не было полностью нулевого кодового слова. Это обусловлено тем, что нулевое кодовое слово в этой системе получается на выходе декодера и тогда, когда из канала в него ничего не поступает. Чтобы эта ситуация не была перепутана с ситуацией, когда принято нулевое слово, надо сделать так, чтобы при кодировании нулевое слово никогда не возникало (в обычном циклическом коде нулевое слово всегда есть – оно получается из нулевого слова источника).

Процедуры кодирования и декодирования, которые были разработаны с учётом всех этих требований, мы опишем в виде действий над полиномами (с некоторыми пояснениями).

*Кодирование**:*** пусть *C(X)* – полином, полученный от источника. Он соответствует слову из *k*’бит, то есть, его степень не более чем *k*-1.

Шаг 1: Вычисление контрольной суммы. Припишем к полученной от источника последовательности справа 32 нулевых бита, а старшие 32 бита проинвертируем. После

этого разделим получившийся полином на производящий полином *g(X)* и получим остаток от деления *R(X)*. На языке действий над полиномами это выглядит так:

$$\frac{X^{32}C(X)+X^{k}\mathrm{I}^{(31)}(X)}{g(X)}=Q(X)+\frac{R(X)}{g(X)} \tag{134}$$

где $\mathrm{I}^{(31)}(X)$ – полином 31-ой степени со всеми единичными коэффициентами, X^{32} – сдвиг на 32 бита влево, X^{k} – сдвиг на длину кодируемого слова.

Шаг 2: Формирование кодового слова. Снова возьмём *C(X)* и припишем к нему справа тридцать два нулевых бита. Потом к получившемуся полиному прибавим *R(X)* и ещё прибавим $\mathrm{I}^{(31)}(X)$, что инвертирует *R(X)*. На языке полиномов это выглядит так:

$$X^{32}C(X)+R(X)+\mathrm{I}^{(31)}(X)=X^{(32)}C(X)+\overline{R(X)} \tag{135}$$

Этот полином передаётся в канал.

Декодирование: Пусть из канала принят полином *D(X)*. Опять допишем к нему справа 32 бита нулей, и ещё раз проинвертируем его старшие 32 бита. После этого разделим на *g(X)*, и получим остаток $R^{*}(X)$.

$$\frac{X^{32}D(X)+X^{k+32}\mathrm{I}^{(31)}(X)}{g(X)}=Q^{*}(X)+\frac{R^{*}(X)}{g(X)} \tag{136}$$

Если в канале искажений не было, то перед делением на *g(X)* у нас получится $X^{64}C(X)+X^{k+32}\mathrm{I}^{(31)}(X)+X^{32}R(X)+X^{32}\mathrm{I}^{(31)}(X)$. Нетрудно видеть, что три первых слагаемых в этой сумме делятся на *g(X)* без остатка, так как представляют собой сдвинутый влево на 32 бита числитель левой части (134) с прибавленным к нему *R(X)*, который и делает результат кратным *g(X)*. Поэтому остаток от деления при отсутствии ошибок будет равен остатку от деления $X^{32}\mathrm{I}^{(31)}(X)$ на *g(X)*, то есть, заранее известному полиному (для полноты картины, вот последовательность из 32 его коэффициентов, от 0 к 31 степени 11101111010111011001000001110001).

Как видно из приведённых процедур, в кодовом слове сначала стоит последовательность слов источника, но кодирование и декодирование производится с помощью производящего полинома. Такие коды называются **систематическими циклическими кодами**. Инвертирование различных частей слова, как перед операциями деления, так и перед передачей в канал, производится для того, чтобы в канал никогда не передавалось слово, целиком состоящее из нулей, и чтобы об отсутствии ошибок свидетельствовал не нулевой остаток, а заранее заданный (указан выше).

Данный код обнаруживает не только все пачки ошибок длиной до 32 бит, но и значительную часть более длинных пачек ошибок.

2). **Код для исправления ошибок в музыкальном CD.** В современных сетях передачи данных общего пользования коды с обнаружением ошибок используются гораздо чаще, чем коды с исправлением ошибок, так как каналы связи в таких сетях, как правило, двухсторонние. Коды с исправлением ошибок используются там, где нет возможности двухсторонней связи, или нецелесообразно использовать её для повторных передач. Это, в основном военные, дальние космические и другие специальные системы. Но есть ситуации, с которыми сталкивается и “обычный” пользователь, которые могут требовать использования кодов с исправлением ошибок для, например, восстановления информации. К числу таких ситуаций относится, в частности, воспроизведение музыки в CD проигрывателе. Записывающее CD устройство можно рассматривать как передатчик, CD плеер – как приёмник, а канал, причём многократной передачи – это сам CD. Искажения передаваемой информации возникают из-за дефектов поверхности CD, т.е. царапин. Понятно, что в данном случае повторная передача неприемлема, и надо использовать код для исправления ошибок.

Особенностью ситуации является как раз то, что код должен исправлять пачки ошибок очень большой длины, так как площадь записи одного бита на два порядка меньше, чем размеры обычной царапины. Поэтому в данной системе используется

двухкаскадный код с многоступенчатой системой перемежения бит. Сначала производится первый этап перемежения, потом кодирование кодом Рида-Соломона, работающий в $GF(2^8)$ с параметрами (28,24) символов $GF(2^8)$ (то есть, байт; в битах параметры этого кода (224,192)), потом второй этап перемежения, и второй этап кодирования кодом Рида-Соломона с параметрами (32,28) (в битах (256,224)), потом третий этап перемежения. Результирующая скорость кода получается (24/28)*(28/32)=3/4. Сам по себе такой двухкаскадный код может исправить две ошибки в каждом кодовом слове, но за счёт системы перемежения, результирующий код исправляет пачку ошибок длиной до 4000 бит (≈2,5 мм длины дорожки записи). Заметим, что кроме кодирования с исправлением ошибок, в системе воспроизведения CD приняты ещё меры для борьбы с искажениями сигнала, основанные уже на особенностях человеческого восприятия звука. В результате, за счёт интерполяции звукового сигнала возможно воспроизведение без потери качества при пачке ошибок длиной до 12300 бит (≈7,7 мм длины дорожки записи). Аппаратная реализация декодера представляет собой одну специальную БИС (большую интегральную схему) и 2 килобайта оперативной памяти.

26. Методы повторной передачи

Как уже упоминалось выше, использование кодов с обнаружением ошибок для обеспечения надёжной передачи имеет мало смысла без возможности повторно передать исказившуюся информацию. При практической реализации такой повторной передачи возникает ряд проблем, подходы к решению которых мы кратко рассмотрим ниже.

Модель системы связи с возможностью организовать повторную передачу приведена на рис. 26.1. Здесь, кроме прямого канала, по которому, как и раньше, передаётся информация от передатчика к приёмнику, появляется обратный канал, по которому приёмник может посылать запросы передатчику, и таким образом, управлять его работой. Заметим, что в реальных сетях передачи данных обратный канал – это такой же канал, как и прямой, он также передаёт информацию пользователей, только в противоположном направлении, а команды от рассматриваемого нами приёмника – это только малая часть его нагрузки.

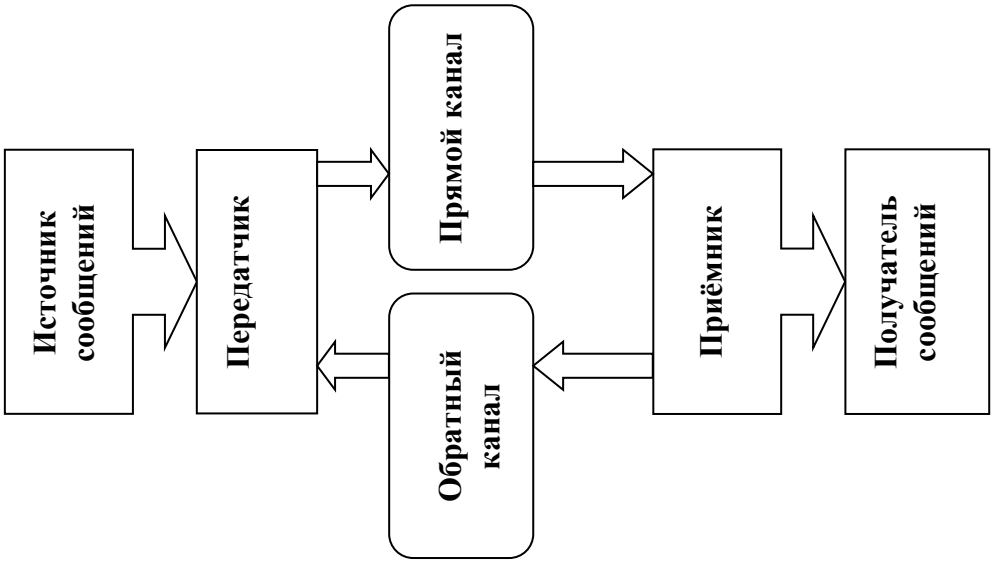

Рисунок 26.1. Система связи с двухсторонним каналом.

Блоки, которые кодируются кодом с обнаружением ошибок, вместе со служебной информацией, необходимой для управления процессом передачи информации, будем называть **кадрами**. Работу алгоритмов повторной передачи будем рассматривать, считая справедливыми следующие положения:

- Приёмник обнаруживает все кадры, содержащие ошибки;
- Каждый передаваемый кадр, как в прямом, так и в обратном канале может задерживаться на произвольное случайное время, прежде чем поступит в приёмник (передатчик);
- Некоторые кадры, как в прямом, так и в обратном канале, могут быть потеряны, и вообще не дойти до места назначения;
- Те кадры в обоих каналах, которые доходят до места назначения, доходят в том порядке, в котором вошли в канал.

Поясним смысл сделанных предположений. Предположение об абсолютной надёжности кода сделано для упрощения анализа. Случайное время задержки кадра надо рассматривать потому, что во-первых, кадр действительно может задержаться с выходом из передатчика (если источнику временно нечего передавать), а во-вторых, когда, согласно третьему предположению, какой-то кадр теряется и не доходит до приёмника, то его повторная передача будет воспринята приёмником как первая, но задержавшаяся. Потери в канале могут происходить, в частности, из-за сильных искажений (так, что приёмник вообще не поймёт, что что-то передавалось). И, наконец, предположение о том, что кадры в канале друг друга не обгоняют, отражает действительное положение вещей, и упрощает анализ.

К алгоритмам повторной передачи будут предъявляться требования во-первых, безошибочной работы в указанных выше условиях, то есть, каждый кадр, полученный от источника для передачи, должен быть передан получателю без ошибок один и только один раз; а во-вторых, эффективности, то есть, времени и ресурса системы на повторные передачи должно тратиться как можно меньше.

Попробуем построить простейший алгоритм повторной передачи, и в процессе этого построения разберёмся, с какими проблемами приходится сталкиваться. Пусть передатчик (будем называть его сторона A) передаёт в прямой канал некоторый кадр. Приёмник (сторона B) должен сообщить по обратному каналу, принят ли кадр без ошибок. Предположим, что приёмник посылает по обратному каналу специальное служебное сообщение ACK (acknowledgement), если ошибок не было, и NAK (negative acknowledgement), если ошибки были. Передатчик, передав кадр, ждёт, какое из этих служебных сообщений придёт по обратному каналу, и в зависимости от этого, либо повторяет передачу кадра, либо переходит к передаче следующего кадра.

Вспомнив о возможности потерь в обоих каналах, нетрудно понять, что и приёмнику и передатчику необходимо правило тайм-аута: для передатчика – если по обратному каналу в течение некоторого времени T ничего не приходит, то передатчик повторяет последний кадр; для приёмника – если от передатчика в течение некоторого времени T^* ничего не приходит, приёмник повторяет по обратному каналу последний запрос.

Однако, этого недостаточно, чтобы обеспечить корректную работу алгоритма. Действительно, предположим, что ACK задержался в пути, и пришёл в передатчик уже после того, как по истечении тайм-аута, передатчик повторно передал кадр. Тогда передатчик будет думать, что это подтверждение второй передачи, а приёмник на копию первого кадра будет думать, что это второй кадр, и примет его ещё раз – то есть, произойдёт незамеченное дублирование кадров. Чтобы этого избежать, надо добавить в качестве служебной информации номер передаваемого кадра, и при этом не только в сам кадр в прямом канале, но и в запрос от приёмника к передатчику в обратном канале. То есть, теперь передатчик нумерует кадры, начиная с 0, посылает очередной, и ждёт запроса

от приёмника, а приёмник, приняв без ошибок кадр с каким-то номером, посылает запрос на кадр, с номером, на единицу большим, а приняв кадр с ошибками, посылает запрос на кадр с тем же номером, что и ошибочный. Этот алгоритм (вместе с правилами тайм-аута) уже работает безошибочно. Действительно, из-за правила тайм-аута передатчик будет повторять передачу какого-то кадра через каждые T единиц времени, и если вероятность безошибочного приёма этого кадра больше нуля, то рано или поздно это произойдёт. После этого приёмник начнёт передавать запрос на следующий кадр через T^* единиц времени, и рано или поздно, передатчик его получит, и перейдёт к следующему кадру.

На практике ещё приходится обращать внимание на то, что поле номера кадра имеет конечный размер, и при долгой работе будет переполняться. Оказывается, для данного алгоритма, который называется **алгоритмом с остановкой и ожиданием**, достаточно нумеровать кадры по модулю 2, и его безошибочность сохраниться.

Однако, с точки зрения эффективности, этот алгоритм не слишком хорош. Пока передатчик ждёт ответа от приёмника, канал простаивает. Понятно, что для повышения эффективности надо разрешить передатчику во время ожидания подтверждения на какой-либо кадр передавать тем временем в канал следующие кадры в надежде, что ошибок не будет. Именно эта идея лежит в основе более эффективного алгоритма – **алгоритма с подтверждением на *n* шагов назад**, который мы сейчас рассмотрим.

Пусть в системе задан параметр n, который обозначает количество кадров, которые передатчик может передать в канал без подтверждения от приёмника правильного приёма первого из них. Получение запроса на кадр номер i означает, что приёмник подтверждает правильный приём на все кадры, с номерами, меньшими i. Сформулируем инструкции этого алгоритма более строго.

1) *Инструкции стороны A:* на стороне A хранятся переменные $y_{\min}$.и $y_{\max}$ Переменная $y_{\min}$ равна номеру кадра, на который получен запрос от приёмника последний раз (в начальный момент $y_{\min}=0$). Переменная $y_{\max}$ на один больше, чем максимальный номер кадра, который передатчик уже хотя бы раз передавал (в начальный момент $y_{\max}=0$). Передатчик хранит все полученные от источника кадры, которые ещё не передавал, а также все кадры, с номерами, большими, чем $y_{\min}$. Передатчик начинает передавать кадры с номера $y_{\min}$ по порядку, увеличивая (если надо) каждый раз $y_{\max}$, при этом проверяя, что $y_{\max}- y_{\min}\le n$. Как только это нарушается, передатчик возвращается к передаче кадра с номером $y_{\min}$ и следующих после него по порядку номеров до $y_{\max}$. Также сохраняется правило тайм-аута – не реже, чем каждые T единиц времени передатчик повторяет передачу кадра $y_{\min}$ и последующих (на случай, если кадров для передачи у передатчика новых не появляется).

2) *Инструкции стороны B:* на стороне B хранится переменная $y_{\text{пр}}$ – это номер кадра, который приёмник ожидает (в начальный момент $y_{\text{пр}}=0$). Когда из канала без ошибок принимается кадр, его номер сравнивается с $y_{\text{пр}}$. Если номер равен $y_{\text{пр}}$, кадр отправляется получателю, $y_{\text{пр}}$ увеличивается на 1 и отсылается в запросе к A. В противном случае кадр игнорируется. Также сохраняется правило тайм-аута – не реже, чем каждые T^* единиц времени приёмник повторяет запрос с текущим значением $y_{\text{пр}}$.

Аналогично предыдущему алгоритму, можно доказать безошибочность работы: если приёмник ожидает кадр $y_{\text{пр}}$, он будет повторять запрос на него до тех пор, пока запрос не дойдёт до передатчика. После этого передатчик будет повторять передачу кадра $y_{\min}= y_{\text{пр}}$ через каждые T единиц времени, пока приёмник не примет его без ошибки. После этого приёмник перейдет к запросу на следующий кадр.

Также здесь существует проблема переполнения номера кадра. Можно доказать, что алгоритм (с минимальными изменениями) сохраняет безошибочность, если кадры нумеровать по модулю $m>n$.

В смысле эффективности этот алгоритм гораздо лучше предыдущего. Нетрудно видеть, что с задержками и потерями в обратной линии он справляется очень хорошо. Но при ошибке в прямой линии ему приходится повторно передавать и те кадры, которые

шли после ошибочного, сами ошибок не содержали, но были проигнорированы приёмником. Чтобы таких кадров было не слишком много, надо аккуратно подбирать величину *n* и тайм-аутов под конкретные условия.

Можно также усовершенствовать работу приёмника, разрешив ему принимать кадры не в порядке очерёдности, и хранить их, пока он очерёдность не восстановит. При этом приёмник может запрашивать передатчик о каком-то конкретном кадре, после передачи которого можно сразу перепрыгнуть на несколько номеров вперёд. Такие алгоритмы называются **алгоритмами с выборочным повтором**. Но они сложнее в реализации, требуют хранения большого количества кадров не только в передатчике, но и в приёмнике, и в полном объёме на практике реализуются редко.

27. Пример протокола повторной передачи

В качестве примера практического применения изложенных выше алгоритмов, рассмотрим самый популярный из протоколов повторной передачи – протокол HDLC. Как обычно, при его реализации, кроме повторной передачи, пришлось решать ещё целый ряд технических проблем, некоторые из которых мы назовём.

Первая проблема – обозначение границ кадров. В информации, помещённой в кадр может встретиться любая последовательность бит, поэтому для того, чтобы выделять начало и конец кадра, надо принимать специальные меры. В HDLC эта проблема решена следующим образом: начало и конец кадра обозначаются специальным сочетанием бит 01111110, называемое **межкадровым флагом**, а внутри кадра после 5 единиц подряд обязательно вставляется 0, который потом будет удаляться при приёме (ещё один пример битстаффинга, но уже для кадрирования).

Другая проблема – необходимость согласования скорости работы передатчика и приёмника. Может сложиться ситуация, когда приёмник не успевает передавать принятые кадры получателю в том темпе, в каком они приходят из канала. Тогда ему надо “притормозить” передатчик на некоторое время. Её разработчики протокола решили, добавив дополнительные возможности в ту часть протокола, которая отвечает непосредственно за повторную передачу.

Итак, в протоколе HDLC предусмотрено три вида кадров – информационные, супервизорные и ненумерованные. Ненумерованные служат для инициализации или прекращения работы линии, и для обмена некоторой дополнительной служебной информацией, в организации повторной передачи они не участвуют. Информационные кадры – это основной тип кадров в системе. Именно они переносят информацию пользователя. Кроме неё, в состав кадра включается заголовок, состоящий из адреса и поля управления, а также трейлер, который представляет собой контрольную сумму кода CRC. Обрамляется кадр, как было сказано выше, межкадровыми флагами.

Информация, необходимая для организации повторной передачи, записывается в поле управления. Среди прочей служебной информации, там выделено три бита под номер кадра и ещё три бита под номер запроса. Это значит, что информационные кадры в этом протоколе обычно нумеруются по модулю 8. При этом в поле номера кадра, идущего, скажем, от *A* к *B*, пишется, очевидно, номер кадра в потоке от *A* к *B*, а поле номера запроса используется приёмником на стороне *A* для указания номера ожидаемого им кадра в потоке от *B* к *A*. Это позволяет при нормальной работе механизма повторной передачи не делать специальные кадры с запросами от приёмников к передатчикам, а вставлять их в информационные кадры обратного направления.

В качестве алгоритма повторной передачи в протоколе HDLC используется алгоритм с подтверждением на *n* шагов назад (где, как правило, *n*=7), несколько усовершенствованный элементами алгоритмов выборочного повтора, а также с возможностью управления скоростью передатчика. Для того чтобы использовать эти усовершенствования (а также для ситуации, когда нет возможности встраивать номера

запросов в информационные кадры по причине отсутствия информационных кадров в нужном направлении), служат супервизорные кадры, не содержащие информации пользователя, а только поле управления. В них есть 3-х битовое поле запроса, и ещё 2-х битовое поле типа запроса. Всего предусмотрено 4 типа запросов.

Первый – RR (Receiver ready) – это обычный запрос приёмника на очередной кадр, подтверждающий приём всех кадров с меньшими номерами. Он равносилен описанному выше получению запроса через соответствующее поле в информационном кадре.

Второй – RNR (Receiver not ready) – этот тип запроса реализует управление скоростью работы передатчика. Посылка его приёмником означает, что приёмник безошибочно принял все кадры с номерами, меньшими, чем номер в запросе, но он просит передатчик пока остановиться и ничего больше не передавать. Получив такой запрос, передатчик останавливается и ждёт получение запроса типа RR с таким же номером кадра.

Третий и четвёртый типы запросов - REJ (Reject) и SREJ (Selective Reject) реализуют ограниченную возможность выборочного приёма, а также реакцию на особые ситуации. Если приёмник способен, он может после обнаружения ошибки в кадре продолжать принимать кадры с большими номерами, а передатчику послать запрос типа SREJ, который означает, что передатчик может повторно передать только кадр с указанным номером, и вернуться потом к передаче новых кадров. Но SREJ можно посылать только на один кадр, и пока этот кадр успешно не принят, новый SREJ посылать нельзя. Если же за это время появились ещё кадры с ошибками, приёмник посылает REJ, по которому передатчик должен повторно передать всю последовательность кадров, начиная с указанного номера. Также REJ используется для сигнализации о некоторых особых ситуациях.

При начале работы протокола существует также возможность выделять под номера кадров и номера запросов не 3 бита, а 7, и в этом случае номера берутся по модулю 128, а n=127. Это предусмотрено для использования протокола в каналах с большим временем прохождения, например, в спутниковых.

Вообще, протокол HDLC очень распространён в системах связи. В частности, он используется для повторной передачи в сети Ethernet. Кроме того, существует ряд похожих стандартов, например, SDLC, ADCCP, LAPB, которые либо почти совпадают с HDLC, либо являются его урезанной по возможностям версией.

28. Задачи и основные понятия управления доступом (разделения канала)

Сети связи по своей природе являются системами коллективного пользования. Часто из экономических соображений невозможно предоставить каждому пользователю отдельный канал связи для выхода в сеть, и группа пользователей использует такой канал совместно. Однако, при наличии в канале двух и более полезных сигналов, оптимальный приём, организованный согласно изложенной выше теории, становится невозможным, и в общем случае, надёжная передача информации в такой ситуации сильно усложняется. Наиболее простой подход, снижающий сложность реализации надёжной передачи до приемлемого уровня заключается в разделении общего канала связи на ряд подканалов, по числу пользователей, таким образом, чтобы передача сигнала по одному из подканалов не влияла на передачи по другим подканалам.

С самой общей точки зрения это не сложно: рассматривая сигналы длительностью T, получаем, что число ортогональных друг другу измерений пространства сигналов $N=DT$. Если в системе k пользователей, то потребуем, чтобы сигналы первого пользователя лежали в подпространстве N_1 измерений, второго – в ортогональном первому подпространстве N_2 измерений, и т.д., так что $N_1+N_2+\ldots+N_k=N$. Таким образом, у каждого пользователя оказывается подканал, ортогональный всем другим подканалам, передача по которому не влияет на другие передачи, и потому изложенными выше

способами может быть сделана надёжной. Рассмотрим наиболее простые примеры применения подобного подхода.

Временное разделение. Разделим временную ось на промежутки длительностью T, называемые циклами, и каждый цикл разделим на k равных интервалов, называемых окнами. Сигналы, которые использует первый пользователь, должны быть отличны от нуля только в первом окне каждого цикла, у второго пользователя сигналы отличны от нуля во втором окне каждого цикла и т.д. Тогда сигналы разных пользователей будут отличны от нуля на непересекающихся интервалах времени, и потому ортогональны друг другу. С житейской точки зрения это очень понятный метод разделения канала – чтобы не мешать друг другу все говорят по очереди, а пока очередь не подошла, молчат и никому не мешают.

Частотное разделение. Разделим полосу частот общего канала связи на k равных частей, называемых частотными подканалами, и потребуем, чтобы спектр сигналов первого пользователя лежал в первом частотном подканале, второго – во втором, и так далее. То, что полосы частот спектров сигналов разных пользователей не перекрываются, делает сигналы приблизительно ортогональными друг другу, что решает задачу разделения канала. Частотное разделение широко используется на практике – все системы радио и телевещания используют этот метод разделения канала (в данном случае эфира). Обусловлено это простотой реализации этого метода с точки зрения радиотехники, а также тем, что при аналоговом теле- или радиовещании непрерывный сигнал, исходящий от передающей станции, согласован с полосой частот выделенного станции подканала, что обусловливает высокую эффективность частотного разделения в данной ситуации (ниже понятие эффективности будет определено). Однако в случае сети передачи данных общего пользования ситуация может быть совершенно иной. Поток данных от пользователя в этой ситуации может быть прерывистым, случайным, и эффективность частотного разделения в этой ситуации не очевидна.

Кодовое разделение. При этом методе разделения канала сигналы разных пользователей могут перекрываться и по времени и по частоте, а их ортогональность достигается за счёт выбора формы сигнала. Хотя это самый современный и перспективный метод разделения канала, из-за сложности его теоретического описания и анализа он не входит в наш курс.

Мы будем рассматривать временное и частотное разделение, называя эти методы базовыми. Кроме них мы будем рассматривать и другие методы разделения канала, и в отношении их всех нас будет интересовать вопрос об их эффективности в сетях передачи данных, так что нам необходимо ввести меру этой эффективности.

С точки зрения пользователя главной характеристикой метода разделения канала является **задержка передачи сообщения** (будем обозначать её d) – по определению, это время, которое проходит от момента, когда отправитель обращается к системе связи для передачи этого сообщения, и до момента, когда получатель заканчивает приём этого сообщения. Это определение подходит к случаю сетей передачи данных. В случае упомянутого выше аналогового радио- и телевещания, вместо сообщения можно использовать понятие сигнала. Легко видеть, что в этом случае задержка передачи сигнала равна просто времени распространения электромагнитной волны от передатчика к приёмнику. Сигнал передающей станции сразу же поступает в соответствующий подканал, имеющий согласованную со спектром сигнала полосу частот, никаких задержек при доступе к каналу не происходит, а передатчик сразу производит приём текущего значения сигнала.

В ситуации же случайного потока сообщений от одного пользователя в сети передачи данных, нетрудно видеть, что задержка передачи сообщения должна рассматриваться как случайная величина, так что нас будет интересовать её среднее значение. Понятно также, что статистические характеристики задержки передачи сообщения будут зависеть от величин, характеризующих текущее состояние сети.

Например, от количества информации, передаваемой по сети другими пользователями, так что нас будут интересовать зависимости средней задержки передачи сообщения от этих величин. Какие это именно величины – мы определим, когда будем формулировать основы математического аппарата, используемого для анализа методов разделения канала, а именно, **теории массового обслуживания**.

В заключение данного параграфа заметим, что задачи разделения канала, называемые также задачами управления доступом пользователей к каналу связи относятся к канальному уровню семиуровневой модели, где образуют отдельный подуровень MAC – media access control (другой подуровень канального уровня образуют рассмотренные выше методы организации повторной передачи – подуровень LLC – logical link control).

29. Основы теории массового обслуживания, теорема Литтла

Центральным понятием теории массового обслуживания является понятие **системы массового обслуживания** (СМО), включающей **обсуживающий прибор**, на который поступает **поток заявок на обслуживание**. Каждое требование должно обрабатываться прибором в течение некоторого времени, и после этого оно покидает систему. В простейшем случае, которым мы и ограничимся, в каждый момент времени в приборе может обрабатываться только одно требование. Остальные требования, если они пришли в то время, когда прибор занят, ждут, образуя **очередь** (поэтому теорию массового обслуживания называют также **теорией очередей**). Относительно очереди мы также ограничимся простейшим случаем, когда размер очереди бесконечен, требования могут прибывать в ней сколь угодно долго, и забираются они из очереди на обслуживание в порядке прибытия (т.н. дисциплина FIFO – first in first out).

Нетрудно понять, как описать ситуацию передачи сообщений по каналу связи в терминах теории массового обслуживания. Канал – это обслуживающий прибор, каждое сообщение – требование на обслуживание, время обслуживания – это время передачи по каналу сообщения, которое зависит как от размера сообщения, так и от скорости передачи канала, а задержка передачи сообщения – это время от появления соответствующей заявки в системе до окончания её обслуживания.

Перейдём к определению величин, характеризующих работу СМО. Пусть $N(t)$ – случайный процесс, значение которого равно числу требований в системе в момент времени t. Введём обозначение $p_n(t) = \Pr\{N(t) = n\}$. Тогда среднее значение $N(t)$ по ансамблю реализаций $\overline{N}(t) = \sum_{n=0}^{\infty} n p_n(t)$. В общем случае как $p_n(t)$, так и $\overline{N}(t)$ зависят от времени и начального распределения вероятностей, но мы будем рассматривать случаи, когда в системе достигается стационарный в статистическом смысле режим, то есть независимо от начального распределения

$$\lim_{t\to\infty} p_n(t) = p_n,\ n = 0,1,\ldots \quad \text{и} \quad \lim_{t\to\infty} \overline{N}(t) = N = \sum_{n=0}^{\infty} n p_n .$$

Кроме того, будем предполагать эргодичность случайного процесса $N(t)$, то есть для взятого по одной реализации среднего $N_t = \frac{1}{t}\int_0^t N(\tau)d\tau$ с вероятностью единица будет выполняться

$$\lim_{t\to\infty} N_t = \lim_{t\to\infty} \overline{N}(t) = N . \qquad (137)$$

Обозначим за d_k задержку k-того по порядку прихода в систему требования, то есть время от момента появления этого требования до окончания его обслуживания. В общем случае оно состоит из времени ожидания в очереди и времени обслуживания. Рассматривая d_k как последовательность случайных величин, обозначим через $\overline{d_k}$ среднее

по ансамблю реализаций *k*-того члена последовательности, и будем считать, что это среднее с вероятностью единица сходится к стационарному значению

$$d = \lim_{k\to\infty} \overline{d_k} .$$

Это предел будем называть **средней задержкой требования в системе**. Именно эту величину мы будем искать. Также будем считать, что во всех, интересующих нас случаях, с вероятностью единица выполняется

$$\lim_{k\to\infty} \frac{1}{k} \sum_{l=1}^{k} d_l = \lim_{k\to\infty} \overline{d_k} = d , \qquad (138)$$

где суммирование выполняется по одной реализации случайной последовательности.

Определим также случайный процесс $A(t)$, равный количеству заявок, пришедших в систему за промежуток времени $[0,t)$. Среднее по ансамблю реализаций обозначим $\overline{A}(t)$. Предположим, что для любой реализации $A(t)$ существует предел

$$\lambda = \lim_{t\to\infty} \frac{\overline{A}(t)}{t} = \lim_{t\to\infty} \frac{A(t)}{t} , \qquad (139)$$

который будем называть **интенсивностью поступления требований** (или **интенсивностью входного потока**).

Дальнейшие доказательства, касающиеся теории массового обслуживания, мы будем проводить нестрого, так как строгие доказательства этих результатов весьма громоздки и не очень наглядны. Первым мы докажем следующую важную теорему:

Теорема Литтла. Если для СМО выполняются сформулированные выше условия, то справедлива следующая формула (формула Литтла):

$$N = \lambda d \qquad (140)$$

Доказательство: определим случайный процесс $D(t)$ как количество требований, ушедших из системы за промежуток времени $[0,t)$. Если предположить, что в начальный момент времени СМО пуста, то очевидно $N(t)=A(t)-D(t)$. Изобразим графически некоторую реализацию процессов $A(t)$ и $D(t)$ (рис.29.1). Рассмотрим область между графиками $A(t)$ и $D(t)$ для некоторого момента времени t. С одной стороны, эта площадь есть $\int_0^t N(\tau)d\tau$. С другой – она может быть выражена как $\sum_{i=1}^{B(t)} d_i + \sum_{i=B(t)+1}^{A(t)} (t-t_i)$, где t_i – время прибытия i-того требования. Приравняем эти два выражения, разделим на t и перейдём к пределу:

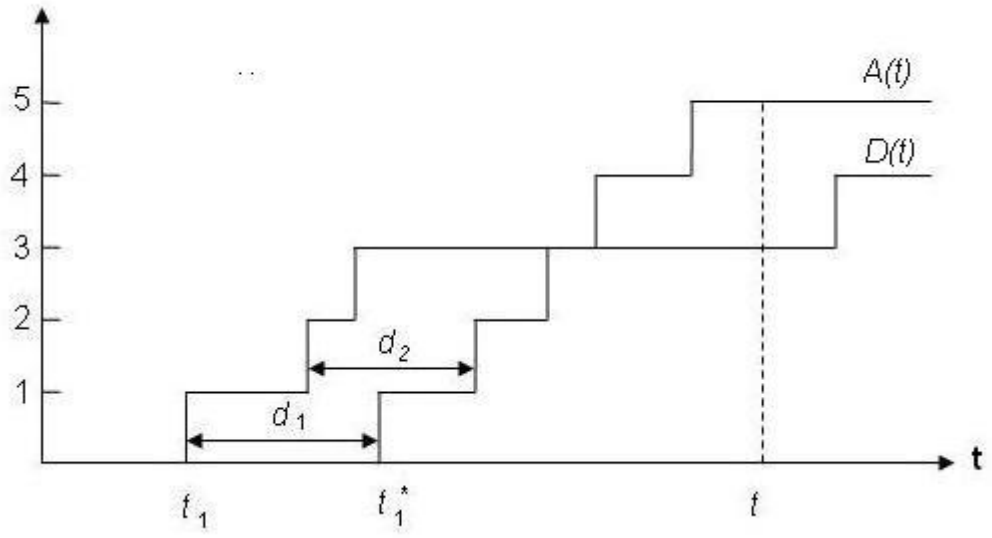

Рисунок 29.1. Реализации процессов прихода и ухода требований из СМО.

$$\lim_{t\to\infty}\frac{1}{t}\int_0^t N(\tau)d\tau=\lim_{t\to\infty}\frac{1}{t}\left(\sum_{i=1}^{D(t)}d_i+\sum_{i=D(t)+1}^{A(t)}(t-t_i)\right)=\lim_{t\to\infty}\frac{A(t)}{t}\lim_{t\to\infty}\frac{\sum_{i=1}^{D(t)}d_i+\sum_{D(t)+1}^{A(t)}(t-t_i)}{A(t)}.$$

Левая часть равна N согласно (137), первый предел справа равен λ согласно (139). Если рассматривать те моменты времени, когда $A(t)=B(t)$, то второй предел справа, очевидно, равен d, согласно (138). Если $A(t)\neq B(t)$, то в числителе присутствует вторая сумма, где вместо d_i стоят t-t_i – для тех заявок, которые находятся в системе, эта величина меньше, чем d_i. Нестрогость нашего доказательства заключается в частности, в том, что мы без доказательства примем, что при $t\to\infty$ в рассматриваемом нами случае величина второй суммы становится пренебрежимо малой по сравнению с первой и не влияет на величину предела. В заключение доказательства заметим, что оно сохраняет свою справедливость при любой дисциплине обслуживания – она нигде в доказательстве не фигурировала. ■

Значение формулы Литтла – в её большой общности. Она справедлива почти для всех СМО, которые в пределе достигают статистического равновесия. А проявления этой формулы часто встречаются в повседневной жизни. Так, например, в дождливую погоду в городе обычно больше автомобилей на улице, чем в сухую. Это не иллюзия, а факт – необходимость использовать машину в городе практически не зависит от погоды, то есть интенсивность входного потока одинакова, средняя скорость движения машин в дождь меньше, значит, среднее время пребывания d на улице больше, соответственно больше и N – среднее число машин на улице. Другой пример из повседневной жизни – заметная разница в величине помещения ресторанов быстрого питания и популярных дорогих ресторанов. При одном и том же потоке клиентов в ресторане быстрого питания посетители проводят заметно меньше времени, чем в дорогом. Потому и среднее число посетителей в ресторане быстрого питания меньше, для них достаточно меньшего помещения.

Рассмотрим теперь нужные для нас примеры применения этой формулы. В качестве системы, для которой справедлива формула Литтла, можно взять очередь заявок на обслуживания (без прибора). Входная интенсивность λ такая же, стационарное среднее время ожидания в очереди обозначим через w (аналогично d в (138)), а стационарную среднюю величину очереди через Q (аналогично N в (135)) , тогда

$$Q=\lambda w. \qquad (141)$$

Также формула Литтла справедлива и для прибора отдельно от очереди. Обозначим время обслуживания i-той заявки через x_i, а стационарное среднее – через x (аналогично d в (138)). Стационарное среднее число заявок в приборе обозначим как ρ (аналогично N в (137)), тогда

$$\rho=\lambda x. \qquad (142)$$

Учитывая, что в приборе может быть либо одна, либо ноль заявок, нетрудно видеть, что ρ имеет смысл вероятности того, что прибор в произвольный момент времени занят. В случае канала связи эта величина – **коэффициент использования канала**, то есть доля времени, когда канал занят передачей информации.

Для продвижения дальше нам придётся определить основные параметры СМО: тип входного потока и тип и параметры времени обслуживания заявки в приборе. В простейшем случае (который мы и рассматриваем) время обслуживания каждой заявки – независимая одинаково распределённая случайная величина, параметры распределения не зависят от входного потока и числа требований в системе, входной поток так же не зависит от состояния СМО.

Существует соглашение об обозначениях СМО указанием основных параметров в следующем виде: X/X/n, где первая буква – тип входного потока, вторая – тип распределения времени обслуживания, а на третьем месте стоит цифра – это число обслуживающих приборов (у нас здесь всегда будет 1). В качестве обозначения типов входного потока используют буквы;

M – пуассоновский поток,

D – детерминированный поток,

G – входной поток общего вида, в котором интервалы между последовательными заявками являются независимыми, одинаково распределёнными величинами.

В качестве обозначения типов распределения времени обслуживания используют:

M– экспоненциальное распределение,

D – детерминированное время обслуживания,

G – распределение общего вида.

Для сетей передачи данных входной поток часто моделируют пуассоновским потоком, чтобы отразить случайный, прерывистый характер поступления сообщений для передачи от пользователей. Поэтому СМО, которая будет интересовать нас в первую очередь – это M/G/1.

30. Система M/G/1, формула Поллачека-Хинчина, анализ частотного разделения

Напомним определение и некоторые свойства пуассоновского потока (случайного процесса).

Определение. Случайный процесс $A(t)$, $t \geq 0$, принимающий целые неотрицательные значения, называется пуассоновским потоком с интенсивностью λ, если

1). $A(0)=0$ и $\forall s < t$ величина $A(t)- A(s)$ равна числу заявок, поступивших на интервале (s,t].

2). $\forall t_1 < t_2 < \ldots < t_n$ величины $A(t_1), A(t_2) - A(t_1), \ldots, A(t_n) - A(t_{n-1})$ взаимно независимы (то есть, числа заявок, поступающих на непересекающихся интервалах времени, независимы). Другими словами, пуассоновский поток – это случайный процесс с независимыми приращениями.

3). Число заявок, поступивших в любом интервале длины τ, имеет пуассоновское распределение с параметром $\lambda\tau$, т.е.

$$\Pr\{A(t+\tau) - A(t) = n\} = e^{-\lambda\tau} \frac{(\lambda\tau)^n}{n!}, \quad n = 0,1,\ldots$$

Пуассоновский поток имеет следующие важные свойства:

1. Интервалы между последовательными поступлениями заявок независимы и распределены экспоненциально с параметром λ.

2. $\forall t \geq 0, \delta \geq 0$
$$\Pr\{A(t+\delta) - A(t) = 0\} = 1 - \lambda\delta + o(\delta),$$
$$\Pr\{A(t+\delta) - A(t) = 1\} = \lambda\delta + o(\delta),$$
$$\Pr\{A(t+\delta) - A(t) \geq 2\} = o(\delta).$$

Смысл этого свойства состоит в том, что поступления в пуассоновском потоке единичные с вероятностью единица, а вероятность одновременного поступления более чем одной заявки равна нулю.

3. Суммарное число поступлений заявок на совокупности непересекающихся временных интервалах с длинами $\tau_1, \tau_2, \ldots, \tau_n$ имеет пуассоновское распределение с параметром $\lambda(\tau_1 + \tau_2 + \ldots + \tau_n)$.

4. Пусть $A_1(t)$, $A_2(t)$,…, $A_n(t)$ – независимые пуассоновские потоки с интенсивностями $\lambda_1, \lambda_2, \ldots, \lambda_n$ соответственно. Тогда процесс $A(t)= A_1(t)+A_2(t)+\ldots+A_n(t)$ – это пуассоновский поток с интенсивностью $\lambda_1+\lambda_2+\ldots+\lambda_n$.

Эти свойства пригодятся нам в дальнейшем, а теперь перейдём к рассмотрению СМО M/G/1. Входной поток у неё пуассоновский, распределение времени обслуживания – общего вида, о котором мы будем предполагать только, что у него существуют конечные математическое ожидание $\bar{x}$ и математическое ожидание квадрата $\overline{x^2}$. Прибор в системе один, то есть в каждый момент времени может обслуживаться не более одной заявки.

Докажем одно важное свойство, которым обладает СМО с пуассоновским входным потоком. Для этого, кроме введённой выше стационарной вероятности

$$p_n = \lim_{t\to\infty} \Pr\{N(t) = n\}, \quad n = 0,1,\ldots,$$

рассмотрим также стационарную вероятность

$$a_n = \lim_{t\to\infty} \Pr\{N(t) = n \mid \textit{требование поступило сразу после момента } t\}.$$

Про такие вероятности говорят, что это вероятности для входящей заявки застать СМО в состоянии $N(t)=n$. В общем случае $a_n \neq p_n$. Например, если рассмотреть СМО, где входной поток таков, что интервалы между соседними поступлениями равномерно распределены между 2-мя и 4-мя единицами времени, а длительность обслуживания всегда равна 1 единице времени, то любое требование всегда застаёт систему пустой, то есть $a_0=1$. В то же время, $p_0=2/3$, $p_1=1/3$.

Свойство: Для СМО с пуассоновским потоком, независимым от состояния системы,

$$a_n = p_n \tag{143}$$

Другими словами, входящая заявка всегда застаёт систему в обычном состоянии.

Доказательство: То, что заявка пришла сразу после момента t можно описать как событие, состоящее в том, что $A(t+\delta)-A(t)\geq 1$ при $\delta\to 0$. Тогда

$$a_n(t) = \lim_{\delta\to 0} \Pr\{N(t) = n \mid A(t+\delta) - A(t) \geq 1\} = \lim_{\delta\to 0} \frac{\Pr\{N(t) = n, A(t+\delta) - A(t) \geq 1\}}{\Pr\{A(t+\delta) - A(t) \geq 1\}} =$$

$$= \lim_{\delta\to 0} \frac{\Pr\{A(t+\delta) - A(t) \geq 1 \mid N(t) = n\} \Pr\{N(t) = n\}}{\Pr\{A(t+\delta) - A(t) \geq 1\}},$$

где мы применили формулу Байеса. По предположению, событие входного потока $A(t+\delta)-A(t)\geq 1$ не зависит от количества требований в системе, то есть

$$\Pr\{A(t+\delta) - A(t) \geq 1 \mid N(t) = n\} = \Pr\{A(t+\delta) - A(t) \geq 1\}.$$

Отсюда получаем $a_n(t) = \Pr\{N(t) = n\} = p_n(t)$, и, устремляя $t\to\infty$, получаем (143). ■

Определение: Остаточное время обслуживания в момент поступления *i*-той заявки, которое мы обозначим как R_i, равно 0, если в момент поступления i-той заявки прибор пуст, и равно времени, оставшемуся до окончания текущего обслуживания, если прибор не пуст.

Теорема Поллачека-Хинчина: Для системы M/G/1

$$w = \frac{\lambda \overline{x^2}}{2(1-\rho)}, \qquad d = \bar{x} + \frac{\lambda \overline{x^2}}{2(1-\rho)} \tag{144}$$

Доказательство: Пусть w_i – это время ожидания i-той заявки, а Q_i – это величина очереди, которая была непосредственно перед приходом i-той заявки. Тогда нетрудно видеть, что

$$w_i = R_i + \sum\nolimits_{j=i-Q_i}^{i-1} x_j, \tag{145}$$

так как в очереди i-тая заявка должна во-первых, подождать окончания текущего в момент её появления обслуживания, и во-вторых, обслуживание всех Q_i заявок, стоящих в очереди перед ней. Чтобы получить выражение для стационарного среднего w, нам необходимо усреднить это выражение по ансамблю реализаций и устремить $i\to\infty$. Чтобы усреднить стоящую справа сумму случайного числа Q_i случайных величин x_j, заметим, что по предположению, все x_j независимы и одинаково распределены, и не зависят от Q_i. Кроме того, из свойства (143), распределение Q_i такое же, как распределение Q. Поэтому стационарное среднее значение суммы в (145) есть произведение среднего значения слагаемого на среднее значение числа слагаемых, то есть $\overline{xQ}$. Предполагая существование стационарного среднего у R_i и обозначая его через R, получим

$$w = R + \bar{x}Q = R + \lambda \bar{x} w = R + \rho w,$$

где мы использовали (141) и (142). Отсюда $w=R/(1-\rho)$. Величину R вычислим из графических соображений (см. рис. 30.1). На графике изображена некоторая реализация случайного процесса $r(t)$, который равен текущему остаточному времени обслуживания.

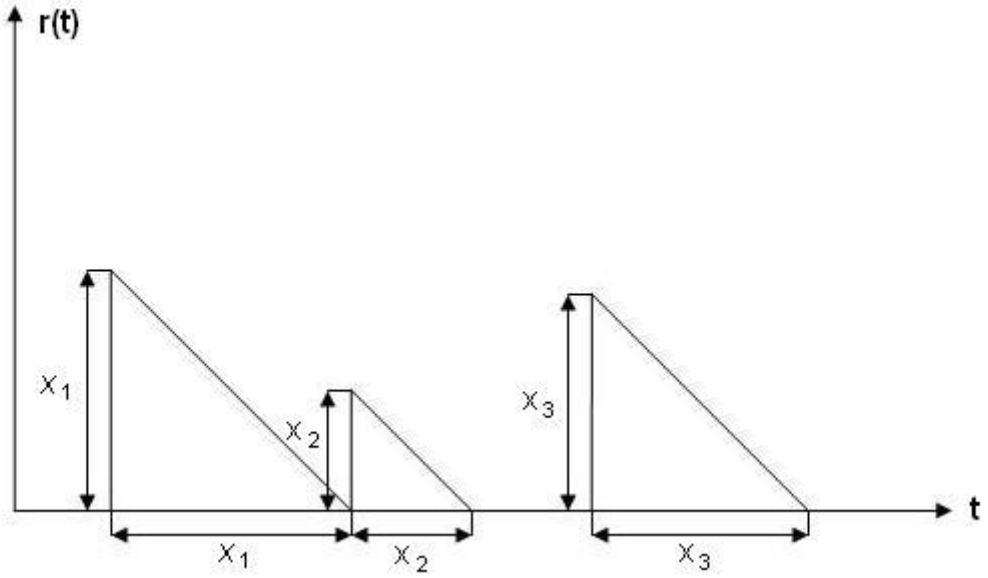

Рисунок 30.1. Реализация процесса *r*(*t*), равного остаточному времени обслуживания.

Заметим, что в момент начала нового обслуживания $r(t)$ принимает значение x_i – времени начавшегося обслуживания, а потом убывает линейно до 0 в течении x_i единиц времени. Рассмотрим некоторый момент времени t, для которого $r(t)=0$. Среднее по времени значение $r(t)$ на отрезке [0,t) есть

$$\frac{1}{t}\int_0^t r(\tau)d\tau = \frac{1}{t}\sum_{i=1}^{D(t)}\frac{x_i^2}{2} = \frac{1}{2}\frac{D(t)}{t}\frac{\sum_{i=1}^{D(t)}x_i^2}{D(t)},$$

где $D(t)$ – количество закончившихся обслуживаний (то есть, обслуженных заявок) на отрезке [0,t). Взяв предел при $t\to\infty$, получим, считая, что стационарное среднее у всех процессов существует, и, учитывая, что для системы в стационарном состоянии $D(t)\to A(t)$,

$$R = \frac{1}{2}\lambda\overline{x^2},$$

откуда следует (144). Для произвольного момента t, кроме суммы квадратов возникнет ещё один член, равный площади части треугольника, который полностью не вошел в интервал [0,t). Очевидно, что при $t\to\infty$ его вклад стремится к нулю, и (144) справедливо. ■

Применим полученный результат к системам M/M/1 и M/D/1. При этом будем использовать так называемую интенсивность обслуживания μ, которая по определению есть среднее количество обслуживаемых за единицу времени заявок при условии, что прибор постоянно занят, т.е. $\mu = 1/\bar{x}$. В этих терминах $\rho = \lambda\bar{x} = \lambda/\mu$.

Для M/M/1 (где μ – параметр экспоненциального распределения времени обслуживания), $\bar{x} = \frac{1}{\mu}$, $Dx = \frac{1}{\mu^2}$, и потому $\overline{x^2} = \frac{2}{\mu^2}$. Отсюда

$$w = \frac{\rho}{\mu(1-\rho)}, \qquad d = \frac{1}{\mu} + \frac{\rho}{\mu(1-\rho)} = \frac{1}{\mu(1-\rho)}. \tag{144}$$

Для M/D/1 $\bar{x} = \frac{1}{\mu}$, $Dx = 0$, и потому $\overline{x^2} = \frac{1}{\mu^2}$. Отсюда

$$w = \frac{\rho}{2\mu(1-\rho)}, \qquad d = \frac{1}{\mu} + \frac{\rho}{2\mu(1-\rho)} = \frac{2-\rho}{2\mu(1-\rho)}. \tag{145}$$

У M/D/1 среднее время ожидания в очереди w в два раза меньше, чем у M/M/1, только из-за разных распределений времени обслуживания. Соответственно, средняя задержка требования в системе d для малых (близких к 0) нагрузок ρ почти одинакова, так как в

этом случае очереди малы, и d близко к среднему времени обслуживания. А при больших (близких к 1) нагрузках d для M/D/1 в два раза меньше, так как решающий вклад в этом случае вносит время ожидания в очереди.

Нетрудно видеть, что у M/D/1 минимальная дисперсия времени обслуживания, и потому, при одинаковых λ и μ, самое малое d среди всех систем M/G/1. Мы будем использовать этот результат при нашем анализе эффективности методов разделения канала – будем считать, что все информационные пакеты в канале передачи данных общего пользования имеют одинаковую длину, при этом скорость канала такова, что за единицу времени он передаёт μ таких пакетов, а все пользователи канала одинаковы, поток от каждого пуассоновский, так что суммарный поток пуассоновский с интенсивностью λ. Если бы все пользователи такого канала образовывали СМО, которое при сделанных предположениях, было бы типа M/D/1, то средняя задержка передачи сообщения в этом случае была бы наименьшей из возможных. К сожалению, на практике организовать это, как правило, невозможно, так как каждый пользователь при этом должен точно знать моменты возникновения пакетов для передачи у всех других пользователей. Поэтому формулы (147) будут служить нам в качестве недостижимой нижней границы для средней задержки передачи сообщения.

Формула Поллачека-Хинчина позволяет нам исследовать эффективность частотного разделения канала. Действительно, в сделанных нами ранее предположениях, если разделить исходный канал на k одинаковых подканалов по числу пользователей в системе, то скорость передачи в одном подканале будет в k раз меньше, чем в исходном, то есть μ/k. Входным потоком в подканал будет поток пакетов только от одного пользователя, то есть, пуассоновский поток с интенсивностью λ/k. Подставив в (147) вместо эти величины вместо μ и λ, получим:

$$w_{ЧР} = \frac{k\rho}{2\mu(1-\rho)}, \qquad d_{ЧР} = \frac{k}{\mu} + \frac{k\rho}{2\mu(1-\rho)} = k\frac{2-\rho}{2\mu(1-\rho)}, \tag{148}$$

то есть, в условиях пуассоновского входного потока и одинаковых размеров пакетов средняя задержка передачи сообщения при частотном разделении в k раз больше, чем “идеальное” (т.е. у системы M/D/1), где k – число пользователей.

Что касается временного разделения, то имеющиеся у нас результаты пока не позволяют проанализировать его эффективность.

31. Система M/G/1 с перерывами, сравнение эффективности базовых методов разделения канала

Подробно разберём, как можно было проанализировать работу временного разделения канала, и что нам не хватает для этого. Пусть справедливы все те предположения, которые мы сделали в предыдущем параграфе относительно канала и пользователей, когда рассматривали частотное разделение. Рассмотрим работу временного разделения канала с точки зрения пакета, принадлежащего одному, скажем, первому пользователю. Пусть пакет пришёл в тот момент, когда у пользовтеля скопилась некоторая очередь на передачу. Каждый цикл длина очереди перед нашим пакетом уменьшается на 1, как если бы передача пакета из этой очереди занимала бы весь цикл, а не одно окно. А когда очередь доходит, наконец, до нашего пакета, он сам передаётся одно окно, то есть, в k раз быстрее. Таким образом, в формуле для d из (145) в первом члене, соответствующем передаче самого пакета, нужно оставить μ, а во втором, соответствующем ожиданию в очереди, подставить вместо μ и λ соответственно μ/k и λ/k (что даст в k раз большее время ожидания в очереди, как и у временного разделения).

Но, к сожалению, рассмотренная выше ситуация – не единственная, которая может случиться при работе системы. Пакет может прийти и в тот момент, когда у пользователя нет других пакетов для передачи, ни в очереди, ни в канале. И вот в этом случае обычная СМО работает не так, как система с временным разделением. У обычной СМО пакет бы

сразу начал передаваться, так как прибор (канал) свободен. А при временном разделении пакету приходится ждать своего окна в ближайшем кадре. Чтобы корректно проанализировать работу временного разделения, нам придётся рассмотреть новую математическую модель - **систему массового обслуживания с перерывами.**

Предположим, что теперь обслуживающий прибор будет работать следующим образом: каждый раз, когда он закончил обслуживание заявки, и очередь опустела, то есть больше обслуживать нечего, у прибора начинается перерыв. В течение перерыва прибор новых обслуживаний не начинает, даже если заявки появились. Только после окончания перерыва он начинает обслуживать первую заявку из скопившейся очереди. Если же за время перерыва ни одна заявка не пришла, то у прибора сразу же начинается следующий перерыв. Пусть продолжительности перерывов v_1, v_2, … - независимые одинаково распределённые случайные величины, не зависящие ни от входного потока, ни от состояния СМО, и имеющие конечное математическое ожидание $\bar{v}$ и конечное математическое ожидание квадрата $\overline{v^2}$. Модифицируем наше определение остаточного времени обслуживания.

Определение: Остаточное время обслуживания в момент поступления *i*-той заявки, которое мы обозначим как R_i^*, равно времени до окончания текущего перерыва, если в момент поступления *i*-той заявки прибор пуст, и равно времени, оставшемуся до окончания текущего обслуживания, если прибор не пуст.

Теорема Поллачека-Хинчина для системы с перерывами: Для системы M/G/1 с перерывами

$$w = \frac{\lambda \overline{x^2}}{2(1-\rho)} + \frac{\overline{v^2}}{2\bar{v}}, \qquad d = \bar{x} + \frac{\lambda \overline{x^2}}{2(1-\rho)} + \frac{\overline{v^2}}{2\bar{v}} \tag{149}$$

Доказательство: Аналогично предыдущему доказательству, нетрудно видеть, что

$$w_i = R_i^* + \sum_{j=i-Q_i}^{i-1} x_j ,$$

что для стационарных средних даст $w = R^* + \rho w$, откуда $w=R^*/(1-\rho)$.

Как и ранее, R^* вычислим из графических соображений (см. рис.31.1).

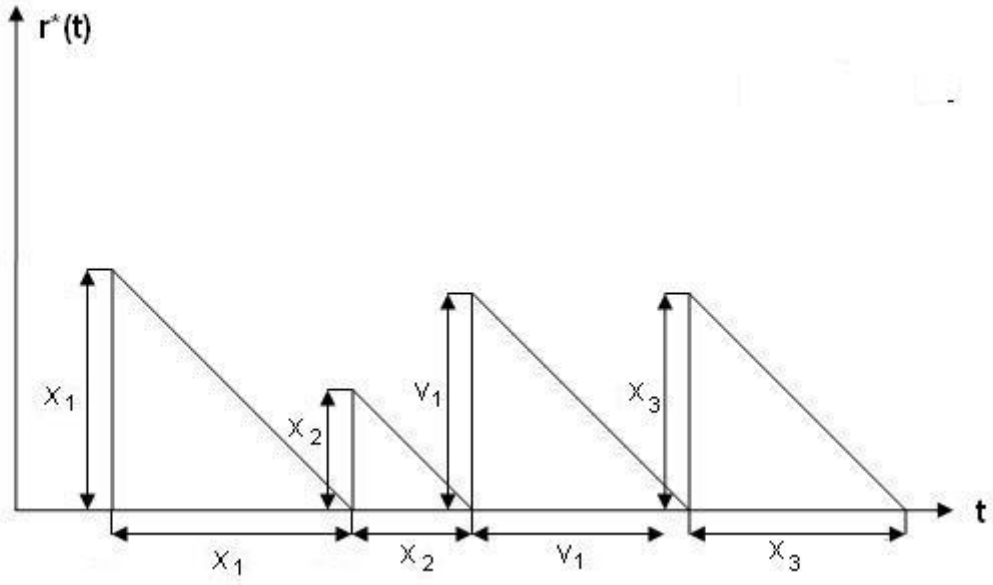

Рисунок 31.1. Реализация процесса $r^*(t)$, равного остаточному времени обслуживания

Записывая среднее по реализации на интервале [0,*t*), когда в момент t величина $r^*(t)$ равна 0, для данного случая получаем:

$$\frac{1}{t}\int_0^t r^*(\tau)d\tau = \frac{1}{t}\sum_{i=1}^{D(t)}\frac{x_i^2}{2} + \frac{1}{t}\sum_{i=1}^{L(t)}\frac{v_i^2}{2} = \frac{1}{2}\frac{D(t)}{t}\frac{\sum_{i=1}^{D(t)}x_i^2}{D(t)} + \frac{1}{2}\frac{L(t)}{t}\frac{\sum_{i=1}^{L(t)}v_i^2}{L(t)},$$

где $L(t)$ – количество перерывов в интервале $[0,t)$. Нам необходимо найти $\lim_{t\to\infty}\frac{L(t)}{t}$ (так как все остальные пределы вычисляются аналогично предыдущему доказательству). Для этого заметим, что доля времени, которая в пределе тратится на перерывы, равна $1-\rho$. Предполагая, что как и для всех остальных процессов, для последовательности $\{v_i\}$ среднее по времени равно среднему по ансамблю (эргодическое свойство), имеем, что $\lim_{t\to\infty}\frac{t(1-\rho)}{L(t)} = \bar{v}$, откуда $\lim_{t\to\infty}\frac{L(t)}{t} = \frac{1-\rho}{\bar{v}}$. Подставив всё в выражение для R^*, получим

$$R^* = \frac{\lambda\overline{x^2}}{2} + \frac{(1-\rho)\overline{v^2}}{2\bar{v}},$$

откуда следует (149). ■

Теперь мы можем получить задержку передачи сообщения для временного разделения. Перерыв – это, очевидно, длина кадра, то есть k окон для передачи. Длина одного окна $1/\mu$, поэтому $\bar{v} = k/\mu$, $\overline{v^2} = k^2/\mu^2$. Далее действуем, как сформулировали ранее – длительность обслуживания в выражении для d из (147), берём $1/\mu$, а в выражение для времени ожидания в очереди из (147) вместо μ и λ подставляем соответственно μ/k и λ/k:

$$d_{вр} = \frac{1}{\mu} + \frac{k\rho}{2\mu(1-\rho)} + \frac{k}{2\mu} = \frac{1}{\mu} + \frac{k}{2\mu(1-\rho)} \qquad (150)$$

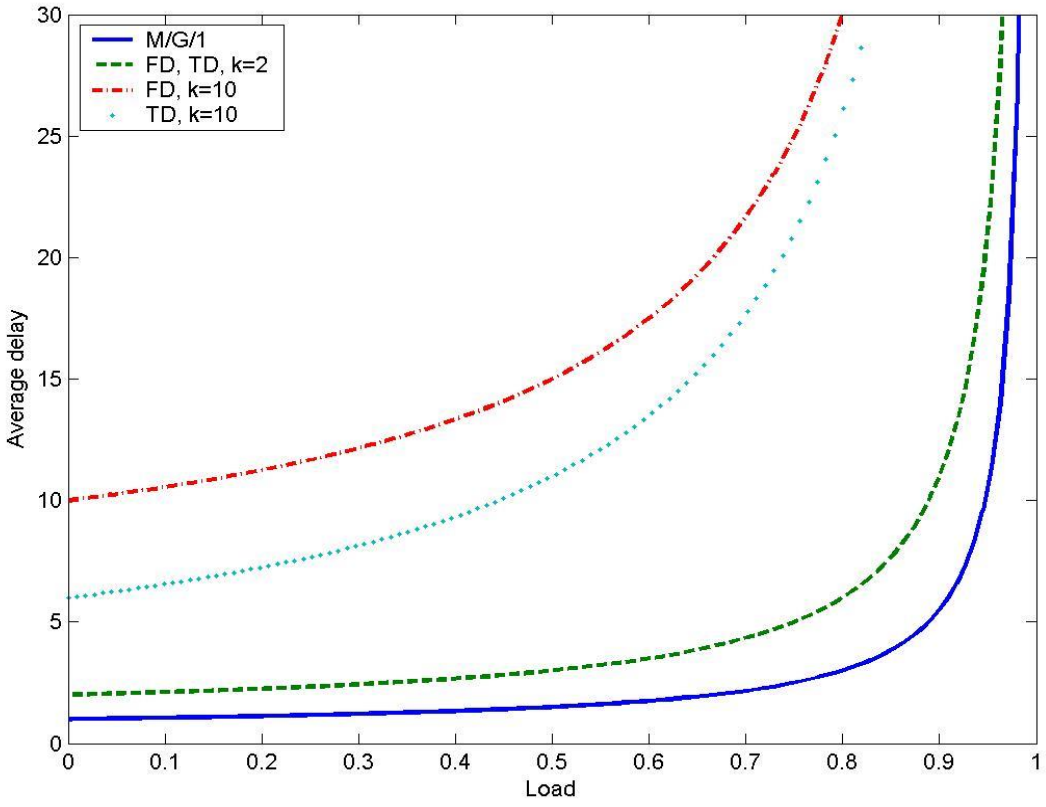

Рисунок 31.2. Задержки передачи сообщения для различных методов разделения канала.

Сравним задержки передачи базовых методов доступа:

$$d_{чр} - d_{вр} = \frac{k}{\mu} + \frac{k\rho}{2\mu(1-\rho)} - \frac{1}{\mu} - \frac{k}{2\mu(1-\rho)} = \frac{1}{\mu}\left(\frac{k}{2} - 1\right). \quad (151)$$

Следовательно, задержки частотного и временного разделения равны только при k=2, а при большем числе пользователей временное разделение эффективнее, причём разница растёт пропорционально k. Однако, временное разделение сложнее реализовать на практике, так как оно требует поддержания синхронизации между всеми пользователями.

Сравним задержки базовых методов доступа с “идеальным” разделением канала – системой M/D/1 (см. рис. 31.2). Видно, что проигрыш базовых методов доступа “идеальному” мал только для очень небольшого числа пользователей, а при большом числе пользователей базовые методы доступа очень неэффективны. Вопрос о том, можно ли предложить что-либо лучшее, хотя бы для каких-то частных случаев, будет рассмотрен в следующих параграфах.

32. Модели и алгоритмы случайного множественного доступа

Рассмотрим ситуацию, когда нагрузка в канале общего пользования мала. В этом случае причина малой эффективности базовых методов доступа очевидна – хотя информации для передачи у большей части пользователей нет, им выделяется большая доля канала, которая простаивает. Один из возможных более эффективных подходов к разделению канала в такой ситуации может быть следующим: пусть каждый пользователь, у которого возникла необходимость передать сообщение, ведёт себя так, как будто канал находится полностью в его распоряжении. Раз конкурентов у него мало, с большой вероятностью он никому не помешает (и ему соответственно, тоже), а в тех редких случаях, когда два (или более) пользователей одновременно попробуют занять канал, надо предусмотреть процедуру, которая бы разрешала возникший конфликт. Эта идея легла в основу целого класса методов разделения канала, которые получили название методов **случайного множественного доступа**.

Исторически первая система связи, использующая случайный множественный доступ, возникла ранее, чем теория такого рода систем, и необходимость понять особенности её работы как раз и послужили начальным толчком к развитию теории случайного множественного доступа.

Это была знаменитая радиосеть Гавайского университета “ALOHA net”, созданная в 1968 году для обеспечения связи главного компьютера университета с терминалами в кампусах, расположенных на разных островах Гавайского архипелага. Предполагалось, что пользователь сможет работать за терминалом, как будто терминал напрямую подключен к главному компьютеру, независимо от того, где на самом деле, терминал находится. Работала система следующим образом: было выделено два радиоканала: первый - для передач от терминалов к компьютеру, второй – широковещательный канал в обратном направлении, который слушали приёмники всех терминалов. Когда пользователь нажимал на клавиатуре терминала кнопку, по первому каналу передатчик терминала сразу же посылал пакет данных, содержащий номер терминала и код нажатой кнопки. Приёмник главного компьютера, приняв пакет, во-первых, передавал его компьютеру, а во-вторых, передатчику широковещательного канала для немедленной передачи “в обратном направлении”. Приёмник терминала принимал копию только что отосланного по первому каналу сообщения, и сравнивал его с хранящимся у него оригиналом. Если было совпадение, передача считалась успешной, и соответствующий символ отображался на экране терминала.

Если же совпадения не было, или пакет по широковещательному каналу не возвращался, то возникала необходимость в повторной передаче. При этом принималось во внимание, что искажения пакета могут возникать потому, что в первом канале наложились друг на друга пакеты двух (или более) пользователей (возник так называемый

конфликт). Если не принимать это во внимание, и производить повторную передачу сразу же, то с большой вероятностью конфликт возникнет снова – ведь другой попавший в конфликт пользователь делает то же самое. Поэтому в качестве **алгоритма разрешения конфликта** в данной системе использовалась случайная задержка перед началом повторной передачи. А именно, перед тем, как повторно передать попавший в конфликт пакет, каждый пользователь производил испытание случайной величины (в данном случае имевшей экспоненциальное распределение со средним много большим, чем длина пакета), и выпавшее значение использовал как задержку перед началом повторной передачи. Так как задержки у разных пользователей выпадали разные, то с большой вероятностью конфликтовавшие ранее пакеты снова друг на друга не накладывались.

Предварительные оценки загрузки первого канала от терминалов к компьютеру давали довольно низкие значения, поэтому предполагалось, что конфликты возникают редко, разрешаются быстро, и никаких проблем возникать при работе системы не должно. Однако, на практике всё оказалось не так. Система вела себя следующим странным образом: после включения она некоторое время работала хорошо, а потом вдруг количество конфликтов в канале резко возрастало, почти никакие успешные передачи не проходили, и терминалы “зависали”. Иногда через довольно продолжительное время система вновь начинала хорошо работать, а иногда приходилось её выключать и перезапускать, чтобы вывести из “почти неработающего” состояния. Анализ поведения с целью понять, что происходит, и улучшить работу системы, и были первыми работами, относящимися к теории случайного множественного доступа.

Ниже мы тоже займёмся этими проблемами, но начнём с анализа идеализированной системы случайного множественного доступа, более простой для анализа. Для этой идеализированной системы мы рассмотрим алгоритм случайного множественного доступа, который, хотя и не совпадает с исходным алгоритмом сети “ALOHA net”, но имеет с ним много общего и поможет разобраться в происходящем.

Сформулируем предположения, определяющие идеализированную систему множественного доступа:

1. **Система синхронная**. Пусть все передаваемые сообщения имеют одинаковую длину, и время передачи по каналу такого сообщения принято за единицу. Будем считать, что все передатчики в системе идеально синхронизированы друг с другом, и могут начинать передачу только в целочисленные моменты времени t=0,1, … . При этом передача заканчивается непосредственно перед следующим целочисленным моментом времени. Таким образом, можно считать, что время в системе разделено на окна единичной длины (интервал [t-1,t) будем называть окном t), и все события начинаются и заканчиваются на стыке окон.

2. **Пуассоновские входные потоки.** Поток новых сообщений для передачи у каждого пользователя пуассоновский, суммарный поток тоже пуассоновский с интенсивностью λ.

3. **Результат передачи – конфликт или успешный приём**. Будем считать, что если два или более пользователя передают пакеты в одном временном окне, то возникает конфликт, и приёмник не получает никакой информации о содержании или отправителях пакетов. Если передаёт только один пользователь, то пакет всегда принимается успешно (неисправимых ошибок в канале не бывает).

4. **Мгновенная обратная связь с сигналами Успех, Конфликт, Пусто.** Будем считать, что в конце каждого окна каждый пользователь получает сигнал обратной связи, который сообщает о том, что происходило в этом окне: успешная передача, конфликт или никто ничего не передавал (окно пустое).

5. **Обязательное разрешение конфликта.** Будем считать, что каждый пакет, попавший в конфликт, обязательно должен быть рано или поздно передан успешно. Будем говорить, что пользователь имеет **задолженность**, если у него есть пакеты для повторной

передачи, а сами пакеты, требующие повторной передачи, будем называть **задолженными**.

Следующее предположение имеет два варианта, и при анализе мы будем указывать, какой вариант используем:

6а. **Бесконечное число пользователей.** При этом мы будем считать, что каждый новый пользователь появляется вместе со своим единственным пакетом для передачи, и после передачи этого пакета исчезает. Это, как мы увидим, приводит к независимости интенсивности входного потока от количества задолженных пакетов в системе.

6б. **Конечное число пользователей, отсутствие буферизации.** Под отсутствием буферизации понимается то обстоятельство, что если у пользователя возник пакет для передачи, то пока он его не передаст, новых пакетов для передачи у него не возникает. Это ведёт к зависимости интенсивности входного потока от количества задолженных пакетов – чем больше задолженность, тем меньше входной поток. (Это предположение, вообще-то, противоречит, предположению 2, но для простоты изложения мы допустим подобную нестрогость).

Теперь сформулируем алгоритм работы пользователя в такой системе, называемый **“Синхронная Алоха”**:

1. Когда у пользователя возникает пакет для передачи, он передает его в ближайшем окне.

2. Если пакет попал в конфликт, он становится задолженным, и пользователь в начале каждого следующего окна производит испытание Бернулли с вероятностью успеха q_r. В случае успеха в испытании, он повторно передаёт в этом окне задолженный пакет, в случае неуспеха – молчит. Так он действует до тех пор, пока задолженный пакет не будет успешно передан. Величина q_r является известной всем пользователям константой.

Анализ работы алгоритма “Синхронная Алоха” мы начнём с вычисления его максимальной пропускной способности.

Определение. Максимальной пропускной способностью системы (алгоритма) называется максимальная интенсивность входного потока, при которой количество требующих передачи сообщений в системе при $t\to\infty$ с вероятностью 1 конечно.

Применительно к рассматриваемой ситуации, будем измерять эту величину в пакетах на временное окно (то есть, количеству пакетов единичной длительности, поступающих в единицу времени).

Данное определение применимо и к системам случайного множественного доступа, где количество требующих передачи сообщений – это задолженные пакеты плюс новые, возникшее в текущем окне, и к системам с очередями, где количество требующих передачи сообщений – это количество сообщений, стоящих в очереди.

Для систем случайного множественного доступа найти эту величину не просто. Сразу можно сказать только, что она в любом случае будет заметно меньше чем 1 пакет на окно, так как часть окон тратится на конфликты. Приведём приблизительную оценку этой величины для Синхронной Алохи.

Пусть выполняется положение 6а о бесконечном числе пользователей. Будем считать, что q_r достаточно мало, и повторные передачи можно приблизительно рассматривать, как пуассоновский поток. Тогда суммарный поток новых пакетов и повторных передач будет пуассоновским с интенсивностью $G>\lambda$. При этом предположении вероятность успешной передачи в произвольном окне есть Ge^{-G} – это вероятность того, что в течение предыдущего окна появился ровно 1 пакет для передачи (новый либо задолженный). Так как в одном окне может успешно передастся либо 1 пакет, либо 0, то эта вероятность есть также среднее число пакетов, уходящих из системы за одно окно. В состоянии статистического равновесия среднее число входящих за единицу времени пакетов, то есть λ, должно быть равно среднему числу уходящих, откуда получаем условие статистического равновесия:

$$\lambda = Ge^{-G} \qquad (152)$$

Изобразим эту зависимость на графике в координатах *(G, λ)* (см. рис. 30.1). Видно, что максимально возможная скорость ухода пакетов из системы достигается при G=1, и равна $1/e \approx 0.368$. Это верхняя граница для максимальной пропускной способности Синхронной Алохи. При $\lambda > 1/e$ статистическое равновесие вообще невозможно.

Если же $\lambda < 1/e$, то горизонтальная линия графика входной интенсивности для бесконечного числа пользователей пересекает кривую (150) в двух точках, то есть статистическое равновесие возможно при двух значениях G. Но тут надо принять во внимание, какие это положения равновесия – устойчивые, или неустойчивые, и соответственно, может ли система долгое время находится в них? В данном случае анализ легко провести с помощью того же графика. Сначала рассмотрим ближнюю к началу координат точку равновесия. Мысленно сдвинемся из неё по оси G, например, вправо. Это соответствует событию, когда вследствие статистической флуктуации возросла интенсивность потока передач. При этом скорость ухода из системы возрастёт, а прихода – остаётся постоянной, то есть, уходить будет больше, чем приходить, значит, уменьшится задолженность, и соответственно, уменьшится G, мы вернемся в точку равновесия. В предположении случайного малого уменьшения G, рассуждая аналогично, увидим, что и здесь мы вернёмся обратно, то есть эта точка равновесия устойчивая.

А вот дальняя от начала координат точка равновесия неустойчивая – любая флуктуация интенсивности потока передач вызовет дальнейшее изменение G в ту же сторону, и система продолжит отклоняться от этой точки равновесия всё дальше и дальше. Если она пойдёт от неустойчивой точки влево, то придёт в устойчивую точку равновесия, а если вправо – то будет двигаться по оси G в бесконечность, то есть задолженность будет бесконечно расти, а скорость передачи стремиться к нулю.

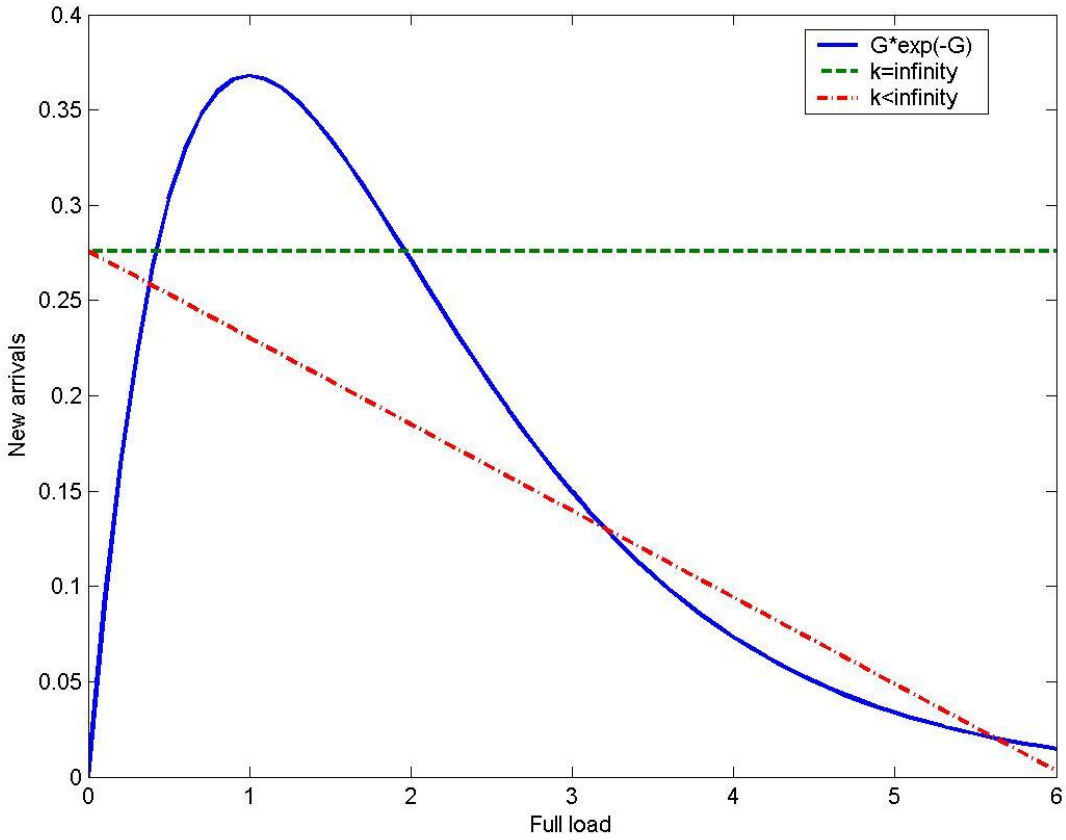

Рисунок 32.1. Скорость передачи алгоритма “Синхронная Алоха”.

Заметим, что при пуассоновском входном потоке существует ненулевая вероятность появления в одном окне любого числа пакетов, получим, что рано или поздно система

окажется справа от неустойчивой точки и “пойдёт в бесконечность”. Таким образом, в предположении бесконечного числа пользователей, максимальная пропускная способность Синхронной Алохи равна нулю.

Анализ максимальной пропускной способности в случае предположения 6б о конечном числе пользователей более сложен, и мы приведём здесь только его результат. На рис. 32.1 для конечного числа пользователей также изображена зависимость входной интенсивности λ от суммарной интенсивности передач G (это следствие зависимости λ от величины задолженности). Теперь это не горизонтальная прямая линия, а наклонная, и возможных точек равновесия получается три. Из них средняя неустойчивая, а две крайние – устойчивые. То есть в этом случае система ведёт себя следующим образом: после включения она попадёт в ближайшую к началу координат, “желательную” точку равновесия, где задолженность мала, а скорость передачи велика. Там система будет находиться до тех пор, пока из-за флуктуаций, присущих пуассоновскому потоку, не будет переброшена в окрестности “нежелательной особой точки”, где задолженность велика, а скорость передачи мала. Там она тоже может находиться долгое время, пока флуктуация не приведёт её обратно в “желательную” точку равновесия. Как видно, поведение этой модельной системы качественно похоже на то, которое наблюдалось в реальной системе.

33. Алгоритм “Адаптивная синхронная Алоха”, приближённый анализ задержки

Давайте познакомимся со способом улучшить синхронную Алоху таким образом, чтобы максимальная скорость передачи перестала быть нулевой. Основная идея этого способа связана с тем обстоятельством, что максимальная скорость ухода пакетов из системы достигается при G=1. Если обозначить количество задолженных пакетов в системе за n, то приблизительно можно оценить G как $q_r n$. То есть, если бы пользователям была бы известна общая задолженность в системе, то они могли бы менять своё q_r так, чтобы поддерживать скорость ухода максимально возможной. Один из алгоритмов для такого рода оценок называется **псевдобайессовским алгоритмом**, а алгоритм синхронного множественного доступа, созданный на основе синхронной Алохи и использующий псевдобайесовский алгоритм для модификации q_r называется адаптивной синхронной Алохой. Давайте сформулируем его правила.

1. **Правило передачи пакета**. Каждый пакет с момента своего возникновения считается задолженным. Это значит, что пользователь производит испытания Бернулли с вероятностью успеха q_r в начале каждого окна с момента появления пакета, и передаёт его, если в испытании выпал успех, до тех пор, пока пакет не будет успешно передан.

2. **Правило оценки q_r.** В конце каждого окна t каждый пользователь производит оценку задолженности в системе n_t по следующей формуле:

$$n_0 = 0, \quad n_t = \begin{cases} \max(\lambda, n_{t-1} + \lambda - 1), & \textit{если в окне t было пусто или успех} \\ n_{t-1} + \lambda + \dfrac{1}{e-2}, & \textit{если в окне t был конфликт} \end{cases} \qquad (153)$$

и если у пользователя есть задолженность, то он определяет величину q_r, которую будет использовать в испытании для окна t+1 по следующей формуле:

$$q_r(t+1) = \min(1, \frac{1}{n_t}) .$$

Можно доказать, что максимальная скорость передачи такой Алохи равна $1/e \approx 0.368$ пакета на окно. Мы не будем приводить здесь доказательство, так как по своей сложности оно выходит за пределы нашего курса, приведём лишь некоторые наглядные соображения относительно смысла данных формул. Если в каком-то окне был успех, то 1 пакет ушёл, а пришло в среднем λ новых. Если же был конфликт, то ничего не ушло, а добавилось λ новых. То, что менять оценку задолженности для пустого окна надо также, как и для

успешного, и то, что в случае конфликта, кроме λ, надо прибавлять ещё $(e\text{-}2)^{-1}$, выясняется в процессе доказательства, простых пояснений, к сожалению, нет.

Заметим, что если λ входного потока неизвестно, то в (153) можно вместо λ подставить $1/e$ – максимальная скорость передачи при этом не изменится.

Приведем теперь приблизительную оценку средней задержки передачи сообщения d для адаптивной синхронной Алохи. Предположим, что λ известно точно. Предположим также, что при текущей задолженности $n_t \geq 2$ вероятность успешной передачи $P_s=1/e$, а при $n_t=1$ пусть $P_s=1$. Это приближение приемлемо при очень малых λ, когда конфликты редки, а также при λ, близких к $1/e$, когда задолженность практически всегда велика.

Определим теперь величины, которые потребуются нам при выводе формулы для средней задержки передачи, так как определения, которыми мы пользовались в теории очередей здесь не вполне подходят.

Пусть w_i – время, прошедшее от момента прихода в систему i-того пакета, до начала i-той успешной передачи. Обратим внимание, что это в данном случае не время ожидания в очереди i-того пакета. Дело в том, что рассматриваемые нами алгоритмы случайного множественного доступа не сохраняют дисциплину “первый пришёл – первый ушёл”. Пришедший i-тым в систему пакет, и i-тым из системы вышедший в общем случае это пакеты разные. Для w_i, тем не менее, как и ранее, будем предполагать существование стационарного среднего, и выполнение эргодического свойства о равенстве среднего по ансамблю и среднего по реализации, то есть

$$\lim_{k\to\infty}\frac{1}{k}\sum_{l=1}^{k} w_l = \lim_{k\to\infty}\overline{w_k} = w.$$

Определим также величину m_i – количество задолженных пакетов в системе в момент, непосредственно предшествующий поступлению i-того пакета. В m_i не даёт вклад пакет, находящийся в процессе успешной передачи в окне, в котором пришёл i-тый пакет, но дают вклад пакеты, попавшие в этом окне в конфликт. Тогда можно записать, что

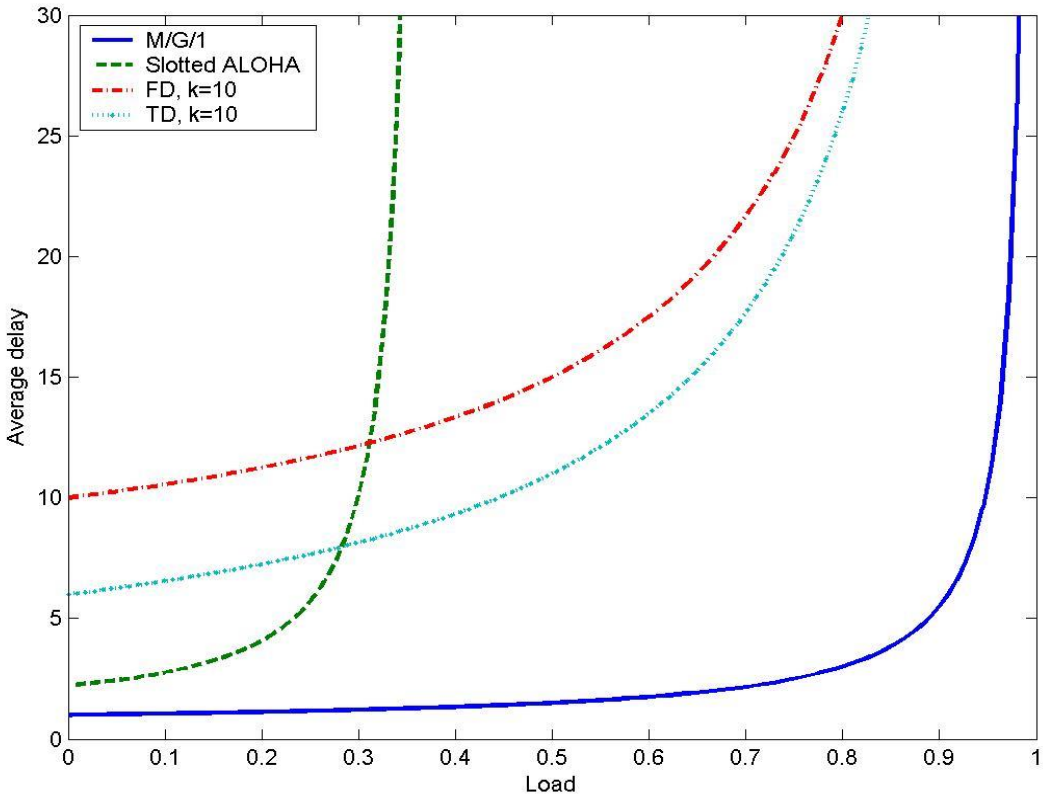

Рисунок 33.1. Задержки различных методов разделения канала.

$$w_i = r_i + \sum_{j=1}^{m_i} T_j + y_i \,,$$

где r_i – это время до начала ближайшего, после прихода i-того пакета, окна, а T_1 – последующий отрезок времени до окончания первой, после прихода i-того пакета, успешной передачи. Аналогично, T_j – это промежуток времени от конца (j-1)-ой до конца j-той, после прихода i-того пакета, успешной передачи. После того, как пройдут m_i успешных передач, начинается интервал y_i, который заканчивается в момент начала следующей успешной передачи, то есть, успешной передачи с номером i.

Заметим, что во время каждого интервала T_j задолженность составляет не менее 2-х пакетов, считая новый, i-тый по порядку поступления, пакет и уже имевшиеся m_i задолженных. Следовательно, по предположению, каждое окно является успешным с вероятностью 1/e и средняя длина каждого T_j равна e.

Усредняя выражение для w_i, чтобы найти стационарное среднее w, используем формулу Литтла $m=\lambda w$, где m – стационарное среднее m_i. Среднее значение r_i есть, очевидно, ½. Тогда

$$w = 1/2 + e\lambda w + M(y). \qquad (152)$$

Рассмотрим момент окончания (i-1)-ой успешной передачи (это последняя из m_i успешных передач, интервалы между которыми обозначены T_j). Если в этот момент в задолженность в системе будет 1, то y_i=0. Если задолженность в этот момент больше единицы, то $M(y)=e$-1. Таким образом, для безусловного среднего имеем:

$$M(y)=(e\text{-}1)\ \Pr\{\text{задолженность} > 1 \mid \text{происходит успешная передача}\}.$$

Обозначим стационарную вероятность иметь задолженность в системе равной n, через p_n. По предположению, если в начале окна задолженность равна 1, то в этом окне обязательно происходит успешная передача. Значит, p_1 – это доля окон, в которых происходит успешная передача при единичной задолженности. Всего доля окон с успешными передачами λ, значит вероятность единичной задолженности при успешной передаче есть p_1/λ, а вероятность иметь задолженность больше 1 при успешной передаче есть 1-p_1/λ, откуда

$$M(y) = (e-1)\left(1 - \frac{p_1}{\lambda}\right) \qquad (155)$$

Осталось найти p_1. Из приведённого выше рассуждения следует, что

$$\lambda = p_1 + (1 - p_1 - p_0)/e.$$

Заметим, что перейти в состояние с нулевой задолженностью в конце окна можно только если в начале окна задолженность была нулевой или единичной, и за это окно ни одного пакета не пришло. Поэтому $p_0=(p_1 + p_0)\ e^{-\lambda}$. Разрешая эти два уравнения относительно p_1, получаем

$$p_1 = \frac{(1-\lambda e)(e^{\lambda}-1)}{1-(e-1)(e^{\lambda}-1)}\,.$$

Подставляя это выражение в (153), а (153) в (152), получим

$$w = \frac{e-1/2}{1-\lambda e} - \frac{(e^{\lambda}-1)(e-1)}{\lambda\left(1-(e-1)(e^{\lambda}-1)\right)} \qquad (156)$$

График величины d=1+w, являющейся средней задержкой передачи для данной системы, вместе с графиками d для базовых методов доступа и идеального разделения канала, изображён на рис. 33.1.

34. Практические приёмы улучшения характеристик систем случайного множественного доступа

К сожалению, рассмотренная выше адаптивная синхронная Алоха, и вообще, та идеализированная система случайного множественного доступа, в которой она работает, являются лишь математическими моделями и на практике нереализуемы. В данном параграфе мы попробуем убрать некоторые нереализуемые предположения этих моделей, рассмотреть, к каким изменениям в результатах это приведёт, и предложить практически реализуемые улучшения реальных систем случайного множественного доступа.

Прежде всего, весьма сложно на практике реализовать самое первое предположение идеализированной модели - о синхронности системы. В реальной радиосети “ALOHAnet” никакой синхронизации между передатчиками не было, и тот вариант алгоритма, который там работал, получил название **“несинхронная (чистая Алоха)”**. Сформулируем правила этого алгоритма:

1. Как только у пользователя появился пакет для передачи, он сразу же начинает его передавать.

2. В случае конфликта, пользователь повторяет передачу пакета со случайной задержкой, имеющей экспоненциальное распределение со средним, много большим длины пакета.

Оценим максимальную пропускную способность этого алгоритма. Пусть, как и ранее, длина всех пакетов одинакова и равна 1 единице времени. Рассмотрим i-тую по порядку попытку передать в системе. Она будет успешной, если передаваемый пакет не перекроет пакет, передававшийся в $(i-1)$-ю попытку и его самого не перекроет пакет $(i+1)$-ой попытки. Считая поток по прежнему пуассоновским с интенсивностью G, получаем, что в течении 1 единицы времени до момента начала i-той передачи из этого потока не должно прийти ни одного пакета, и в течение 1 единицы времени после начала i-той передачи тоже. Вероятность этого события есть e^{-2G}. Соответственно, $1-e^{-2G}$ есть вероятность того, что рассматриваемая попытка будет неудачной. Учитывая, что G – среднее число попыток передать в единицу времени, среднее число неудачных попыток передать в единицу времени будет $G(1- e^{-2G})$. Для сохранения статистического равновесия среднее число успешных передач в единицу времени должно быть равно λ – среднему числу входящих пакетов, поэтому:

$$G=\lambda+G(1-e^{-2G})$$

откуда

$$\lambda=Ge^{-2G} \qquad (157)$$

В отличие от выражения (150), максимум этой зависимости достигается при $G=1/2$, и равен $1/(2e)$. Рассуждая аналогично рассмотрению синхронной Алохи, заключаем, что при $\lambda>1/(2e)\approx0.184$ статистическое равновесие при чистой Алохе невозможно. То есть, если и удастся как-то стабилизировать её работу, подобно адаптивному варианту в синхронном случае, то более 0.184 пакета на окно всё равно не получится.

Диапазон возможных нагрузок получается слишком малым, и хотелось бы как-нибудь его расширить. Рассмотрим приёмы для этого, которые можно использовать в тех сетях, где время распространения сигнала через всю сеть много меньше, чем время передачи пакета.

1. **Прослушивание несущей.** До сих пор пользователь, у которого возник пакет для передачи по каналу случайного множественного доступа, сразу пытался его передавать, тем самым, возможно, портя текущую передачу другого пользователя. Более разумным было бы сначала проверить, не идёт ли в данный момент передача в канале, и если идёт, то подождать, когда она закончится. Этот подход и получил название “прослушивание несущей”. Вопрос заключается в том, сколько именно надо прослушивать канал, чтобы убедиться, что он пуст. Оказывается, для надёжного

определения того, что канал пуст, пользователь должен прослушивать канал в течение времени, необходимого сигналу для распространения от одного конца сети до другого и обратно. Такой подход исключает почти все конфликты, кроме случаев, когда у двух пользователей пакеты для передачи возникли почти одновременно.

2. **Обнаружение конфликтов.** Во многих сетях, использующих канал со случайным множественным доступом, за время, много меньшее времени передачи сообщения, можно определить не только, занят ли канал в данный момент или нет. Пользователь, производящий передачу, так же быстро может определить, передаёт ли он один, или ещё кто-то в данный момент ведёт передачу, и значит, уже произошёл конфликт. В этом случае разумно не продолжать передачу, а прервать её и перейти к разрешению конфликта. Этот подход, называемый обнаружением конфликтов, может ещё улучшить характеристики системы случайного множественного доступа, особенно в сочетании с прослушиванием несущей. При этом пользователь, начавший передачу должен прослушивать канал также в течении времени распространения сигнала из конца в конец сети и обратно, и если за это время конфликт не обнаружен, то далее (при правильной работе системы) его и не будет, и передача пройдёт успешно.

Приближённо оценим, какую максимальную скорость передачи можно получить, если использовать оба этих приёма. Обозначим за β отношение времени распространения сигнала из конца в конец сети и обратно к времени передачи пакета, и предположим, что оно мало ($\beta<<1$). В исходной системе пустые и конфликтные окна занимали 1 окно (единицу времени), теперь они занимают β единиц времени. Пусть исходный алгоритм случайного множественного доступа имел максимальную скорость передачи S пакетов на окно. Это значит, что он обеспечивал в среднем 1 успешную передачу за $1/S$ окон. То есть, в среднем через $1/S$ окон один из пользователей получал в распоряжение канал. Теперь это происходит в среднем за β/S “миниокон”. После этого, пользователь беспрепятственно передаёт сообщение, длина которого равна единице, то есть, тратит на успешную передачу $\beta/S+1$ единиц времени. Более точная оценка учитывает то, что резервирование канала пользователь производил передачей в течение β единиц времени этого же сообщения, то есть к моменту, когда он убедился, что в конфликт не попал, он уже часть сообщения передал, поэтому на успешную передачу он тратит $\beta/S+1-\beta$ единиц времени. Тогда максимальная скорость передачи системы с прослушиванием несущей и обнаружением конфликтов будет

$$S^*=1/(1+\beta(1/S-1)) \qquad (158)$$

Для примера предположим, что $S=1/(2e)\approx0.184$, $\beta=0.01$. Тогда $S^*\approx0.95$, что уже вполне приемлемо.

Остался пока без обсуждения только вопрос о том, можно ли сделать величину максимальной пропускной способности чистой Алохи отличной от нуля и равной её верхней границе (как это было сделано для синхронной Алохи с помощью псевдобайесовского алгоритма). Сам псевдобайесовский алгоритм здесь не пригоден, так как предназначен для синхронной системы. Но кроме несинхронности, в реальных сетях присутствуют и другие черты, делающие такого рода алгоритмы практически непригодными. В частности, результат передачи пакета (успех или конфликт) в реальных сетях часто известен только тем, чьи пакеты пытались передаваться. В настоящее время неизвестны алгоритмы оценки задолженности, которые в таких условиях стабилизируют систему случайного множественного доступа (то есть делают максимальную скорость передачи отличной от нуля), но есть алгоритмы, которые заметно улучшают работу системы тем, что увеличивают время нахождения системы в благоприятной устойчивой точке.

Изучим один из них, который называется “**замедление по двоичной экспоненте**”. Основная идея его состоит в том, что если пользователь, который пытается что-то передавать, часто попадает в конфликт, то для него разумно предположить, что система сильно загружена, и ему надо снижать вероятность повторных передач. Один из

вариантов этого выглядит так: после первой попытки передачи, если пакет попал в конфликт, пользователь производит испытание двоичной случайной величины, где оба значения (0 и 1) выпадают с равной вероятностью. Если выпал 0, то начинает повторную передачу сразу после того, как в течении времени β после конфликта канал был свободен. Если выпал 1, то пропускает 2β, либо передачу одного пакета (предположительно, это пакет того, кому выпал 0) и потом передаёт. Можно сказать, что пользователь после первого попадания в конфликт с вероятностью ½ выбирает одно из двух следующих за конфликтом миниокон длины β для повторной передачи. Если опять получился конфликт, то для следующей повторной передачи с равной вероятностью выбирается одно из 4 следующих за конфликтом миниокон, и т.д. То есть после n-ного конфликта момент начала следующей попытки выбирается из 2^n миниокон. Однако, как указывалось выше, этот алгоритм лишь увеличивает время нахождения системы в каждом из устойчивых состояний.

Попадание системы в нежелательное устойчивое состояние определяется пользователем по тому, что его пакет попал в число конфликтов, превышающее максимально допустимое их количество. После этого пользователь посылает в канал специальный сигнал – сигнал “общего сброса”, и уничтожает пакет, который пытается передать. Каждый пользователь, который слышит сигнал общего сброса, делает тоже самое – повторяет этот сигнал и уничтожает свой пакет. Таким образом, система становится пустой и устанавливается в исходное состояние. Сброшенные при этом пакеты будут восстановлены подуровнем, отвечающим за повторные передачи (LLC подуровнем). И работа системы начнётся снова из окрестностей желательной точки равновесия.

35. Примеры управления доступом в реальных системах связи

В качестве примера применения изложенных выше подходов, рассмотрим протокол доступа к каналу в семействе локальных сетей Ethernet (включающий самый первый Ethernet со скоростью 10 Мб/с, Fast Ethernet со скоростью 100 Мб/с и Gigabit Ethernet со скоростью 1Гб/с).

Во всех этих сетях протокол доступа один и тот же – это алохоподобный алгоритм с прослушиванием несущей, обнаружением конфликтов и замедлением по двоичной экспоненте. Для облегчения обнаружения конфликта предусмотрена передача специального сигнала тем узлом, который обнаружил конфликт. Максимальное количество конфликтов, в которые может попасть пакет до наступления общего сброса, от 10 до 16 для разных Ethernet-ов.

В отличие от рассмотренных ранее модельных примеров, все Ethernet-ы предназначены для работы с пакетами переменной длины. Для корректной работы протокола доступа, а также для совместимости разных типов Ethernet-ов друг с другом, время прослушивания несущей и обнаружения конфликта равно времени передачи самого маленького пакета в 10 Мб/с Ethernet. Если же брать β – отношение этого времени к среднему времени передачи пакета, то оно получается не более 0.01.

Время прослушивания несущей и обнаружения конфликта определяет максимальный размер передающей среды, так как для корректной работы механизма прослушивания несущей и обнаружения конфликтов это время должно быть не меньше максимального времени распространения сигнала из конца в конец и обратно. Поэтому максимальный размер передающей среды различен для разных типов передающих сред – для медных кабелей это от ≈25 м у Gigabit Ethernet до ≈104 м у Fast Ethernet и 10 Мб/с Ethernet; для оптоволоконных кабелей разных типов – от ≈412 м до десятков километров у Fast Ethernet и Gigabit Ethernet.

При соблюдении всех требований стандарта, параметр β (отношение времени прослушивания несущей и обнаружения конфликтов к среднему времени передачи

пакета) получается весьма малым - не более 0.01. Это обуславливает высокую эффективность протокола доступа. При нарушении же ограничений стандарта на максимальную длину сегмента, появляется вероятность необнаруженных конфликтов, которая тем больше, чем больше превышение длины сегмента над максимально допустимой. Это резко снижает эффективность протокола доступа и, соответственно, всей сети. Так что если надо покрыть локальной сетью этого типа большее расстояние, то надо организовывать в ней несколько сегментов, связанных специальными сетевыми устройствами – т.н. “мостами”.

Другое условие хорошей работы Ethernet-ов – это условие сохранения небольшой средней нагрузки в канале. Считается, что максимальной загрузкой канала Ethernet-ов, при которой они ещё удовлетворительно работают, является 1/3, поэтому у системных администраторов это правило получило название “правило 1/3”. Если вследствие роста трафика локальной сети какой-то организации средняя загрузка канала используемого Ethernet превысила 1/3, пора принимать какие-то меры, например, переходить к более быстрым вариантам того же Ethernet-семейства.

В качестве ещё одного примера использования случайного множественного доступа к каналу, можно привести беспроводную сеть Wi-Fi (стандарт IEEE 802.11), в которой один из возможных режимов для доступа к радиоканалу – это т.н. метод доступа с контролем несущей и избеганием конфликтов. Прослушивание несущей в нём сделано по обычной схеме, а вот захват канала пользователем осуществляется не началом передачи пакета с данными, а посылкой специального сообщения, адресованного тому, кому отправитель хочет передать данные – т.н. RTS (ready to send). В этом сообщении есть служебное поле, где отправитель указывает, сколько времени ему потребуется на передачу данных. Если получатель принял RTS, он отвечает специальным сообщением CTS (clear to send), которое является разрешением к отправителю начинать передавать данные. В этом сообщении тоже есть поле длины ожидаемых данных, которое копируется из RTS. Другие пользователи, когда слышат чьи-нибудь сигналы CTS, читают это поле, и после этого считают канал занятым на указанное время. После окончания этого времени они начинают прослушивать канал, чтобы убедиться, что он пуст, и начать новые попытки его зарезервировать. Разрешение конфликтов, которые, как правило, возникают при передаче RTS, разрешаются с помощью задержки на случайное число миниокон, при этом для стабилизации используется замедление по двоичной экспоненте.

В качестве примера более сложной схемы управления доступом, рассмотрим основные принципы управления доступом абонентов в сети сотовой телефонии стандарта GSM. В сотовой телефонии каждый абонент непосредственно обменивается сообщениями с базовой станцией той “соты”, в которой в данный момент находится. А базовая станция через сеть оператора организовывает соединение с той базовой станцией, в “соте” которой находится второй абонент, с которым первый хочет поговорить.

При организации доступа в сотовую сеть приходится решать целый ряд проблем, связанных с особенностями трафика от абонентов, потому рассмотрим трафик одного телефонного соединения подробнее. Запросы на установление соединения (то есть, звонки) к базовой станции разумно представлять случайными событиями, образующими случайный процесс, например пуассоновский поток. Но после того как соединение установлено, поток данных в обе стороны носит уже детерминированный характер – по стандарту GSM пакеты данных одного абонента в таких потоках идут с интервалом в 120 миллисекунд. Учитывая, что разговор длится, как правило, несколько минут, в таких потоках проходят сотни и тысячи пакетов. То есть для уже установленных соединений базовые методы доступа (частотное и временное разделения) вполне подойдут, а вот для установления соединений их использование проблематично. Поэтому управление доступом в сети GSM устроено следующим образом: на каждой базовой станции имеется ряд частотных каналов. Часть их выделена для управления, в том числе один (т.н. RACH – random access channel) является каналом общего пользования для передачи запросов на

установление соединения. В нём реализован алохоподобный алгоритм случайного множественного доступа. По этому каналу каждый пользователь должен передать базовой станции запрос на установление соединения. После этого базовая станция посылает ему по другому служебному каналу информацию о том, какая именно пара пользовательских каналов (один от него, другой к нему) ему выделена. Для разделения пользовательских каналов применяется частотно-временное разделение, когда в каждом частотном подканале ещё организовано временное разделение на 8 пользователей. Частотных подканалов на базовой станции может быть разное число – в зависимости от размеров соты и количества абонентов в ней. Для организации синхронизации, необходимой для временного разделения, служит специальный служебный канал.

36. Системы с опросом и передачей маркера, кольцевые сети с передачей маркера

Как было отмечено выше, эффективность протокола доступа в сети Ethernet сильно зависит, во-первых, от размеров сети, а во-вторых, от загрузки канала, то есть, такого рода протоколы доступа могут использоваться в локальных сетях с небольшой средней загрузкой канала. На практике хотелось бы иметь методы доступа, которые могли бы сохранять эффективность в ситуациях, когда эти ограничения не выполняются. В этом параграфе мы рассмотрим один из таких подходов.

За исходную точку возьмём временное разделение и попробуем устранить некоторые из его недостатков. В частности, при обычном временном разделении окно каждому пользователю предоставляется независимо от того, есть у него пакеты для передачи, или нет.

Как усовершенствованный вариант, рассмотрим следующую систему: пусть есть центральная станция, управляющая доступом в канал и k пользователей. Центральная станция по очереди опрашивает всех пользователей, посылая им специальное сообщение, являющееся разрешением начинать передачу. Если у пользователя нечего передавать, он отвечает специальным коротким сообщением, означающим, что он закончил передачу. Если у пользователя есть, что передавать, он сначала всё это передаёт, а потом уже посылает сообщение об окончании передачи. Так как тем пользователям, у которых информации меньше, канал предоставляется на меньшее время, можно ожидать, что такая система (она называется **системой с опросом**) управляет доступом эффективней, чем простое временное разделение.

Однако, у неё есть большой недостаток – наличие центральной станции, от исправности которой зависит работа всей системы. Нетрудно впрочем, предложить похожую систему, в которой центральной станции нет: **систему с передачей маркера.**

Пусть между k пользователями заранее установлен некоторый порядок, то есть один из них первый, другой второй и т.д. Тогда в начальный момент канал считается принадлежащим первому пользователю. Он передаёт всю имеющуюся у него информацию, а потом посылает специальное сообщение – **маркер**, которое говорит о том, что он передаёт канал второму пользователю, тот действует аналогично и т.д. Последний пользователь передаёт маркер первому. В случае, когда маркер приходит к пользователю, которому нечего передавать, он сразу же передаёт маркер следующему.

Проведём приближённый анализ задержки передачи сообщения в такой системе. Для этого сделаем ряд упрощающих предположений:

1. Поток пакетов у каждого пользователя – пуассоновский, с одинаковой у всех интенсивностью, у суммарного входного потока интенсивность λ.

2. Среднее время передачи пакета $\bar{x} < \infty$, средний квадрат времени передачи $\overline{x^2} < \infty$.

3. Передача маркера занимает время a.

Рассмотрим момент прибытия маркера к некоторому пользователю. Среднее число пакетов, которое у него к этому времени имеется, обозначим N_1, тогда время,

необходимое для их передачи, и завершающей передачи маркера есть $N_1\bar{x}+a$, а среднее время цикла (время между приходами маркера к одному пользователю) есть

$$T_c = k(N_1\bar{x}+a)$$

Заметим, что N_1 пакетов накапливается в среднем именно за среднее время цикла, то есть они связаны формулой Литтла: $N_1 = \frac{\lambda}{k}T_c$, где λ/k – интенсивность входного потока одного пользователя. Отсюда, $T_c = k(\frac{\lambda}{k}T_c\bar{x}+a)$, и вспоминая (140),

$$T_c = \frac{ka}{1-\rho} \tag{159}$$

Ожидание в очереди некоторого пакета складывается из двух частей: w_1 – время от появления пакета до получения маркера его хозяином, и w_2 – время от получения маркера хозяином до начала передачи этого пакета. Относительно оценки w_1 заметим, что наш пакет может с равной вероятность прийти в любой момент времени между моментом ухода маркера от его хозяина и до момента возвращения маркера к хозяину, поэтому среднее значение w_1 – это половина среднего интервала между уходом маркера от пользователя и его следующим приходом. От момента получения маркера до момента его передачи следующему проходит в среднем, как указывалось выше, $\frac{\lambda}{k}T_c\bar{x}$ единиц времени, тогда от ухода маркера до возвращения, в среднем $T_c - \frac{\lambda}{k}T_c\bar{x} = T_c(1-\frac{\rho}{k})$, и

$$w_1 = \frac{1}{2}T_c(1-\frac{\rho}{k}) = \frac{ka(1-\frac{\rho}{k})}{2(1-\rho)} \tag{160}$$

Для того чтобы оценить w_2, предположим, что a настолько мало по сравнению с $\bar{x}$, что можно считать, что a=0. Тогда можно считать, что пакеты всех пользователей образуют одну общую очередь в системе M/G/1, но обслуживаются не в порядке поступления, а в другом. Но среднее время ожидания в очереди не зависит от порядка обслуживания, и потому,

$$w_2 = \frac{\lambda\overline{x^2}}{2(1-\rho)},$$

откуда

$$d = \bar{x} + \frac{ka(1-\frac{\rho}{k})}{2(1-\rho)} + \frac{\lambda\overline{x^2}}{2(1-\rho)}. \tag{161}$$

От задержки идеального разделения канала (161) отличается членом с ka, а при $a=\bar{x}$, превращается в формулу задержки для временного разделения. При больших нагрузках и малом a работает гораздо лучше, чем случайный множественный доступ.

В сети минимальная величина a определяется, как правило, временем распространения сигнала из конца в конец и обратно. Поэтому, реализация систем с передачей маркера на основе общего кабеля с отводами, как для Ethernet, обладала тем же недостатком – длина такого кабеля оказывалась ограниченной. Кроме того, сложные алгоритмы включения-выключения новых пользователей и обработки аварийных ситуаций типа потери маркера привели к тому, что появившиеся в начале 80-х годов 20-го века локальные сети с передачей маркера на основе общего кабеля были вытеснены Ethernet-ом. В настоящее время IEEE даже аннулировало стандарт для общего кабеля с передачей маркера. Но этот подход не исчез – в настоящее время шина с передачей маркера используется в одном из самых популярных в Европе стандартов промышленных

сетей Profibus. И, кроме того, есть особый класс сетей, где передача маркера зарекомендовала себя весьма хорошо. Это **кольцевые сети**.

В кольцевых сетях формально много каналов связи – между каждыми двумя соседними по кольцу пользователями. Однако в смысле передачи информации они не могут считаться самостоятельными каналами, так как различных маршрутов в кольце не существует – маршрут один (если кольцо однонаправленное) или два (если кольцо двунаправленное). А вот в смысле передачи маркера то, что в каждом канале один передатчик и один приёмник, приводит к тому, что передача маркера в таких сетях уже не зависит от времени распространения по каналу, и может быть сделана очень короткой. Фактически, это просто значение одного бита в служебном заголовке пакета.

Кроме того, раз время передачи маркера теперь не зависит от размера сети, кольцевые сети могут охватывать весьма большие расстояния. Специфика этих сетей, а именно то, что каждая станция должна всегда работать для обеспечения работоспособности всей сети, а также необходимость отключать сеть на время ввода новых и исключения старых станций, сделали их более дорогими, чем Ethernet, при использовании в качестве локальных сетей. Но в области так называемых городских сетей связи (MAN – metropolitan area network), занимающих промежуточное положение между локальными (LAN – local area network) и глобальными (WAN – wide area network) сетями, они зарекомендовали себя весьма хорошо.

Алгоритм доступа с передачей маркера в таких сетях довольно очевиден, обратим внимание разве что на то, что получатель, приняв пакет, передаёт его дальше по кольцу, пакет возвращается к отправителю, который сравнивает его с тем, что он отправил, и таким образом, дополнительно контролирует правильность передачи.

Из приёмов, которые можно применить для улучшения работы системы, отметим **полимаркерный режим**, который можно использовать в длинных кольцах, когда время передачи пакета становится много меньше времени распространения сигнала по кольцу. В этом режиме по кольцу одновременно движется много маркеров, соответственно, производительность системы резко повышается.

Также отметим доступный в двунаправленных кольцах режим превращения в однонаправленное кольцо, когда каналы между двумя соседними пользователями неработоспособны (например, когда между этими пользователями вставляют нового). Это увеличивает надёжность сети в случае отказов оборудования.

Практическими примерами использования маркерного доступа в кольцевых сетях являются сети типа Token Ring (однонаправленное кольцо) и FDDI (двунаправленное кольцо). Token Ring был локальной сетью, в 80-х годах 20-го века некоторое время составлял конкуренцию Ethernet, но потом практически исчез. FDDI предназначен для городских сетей, поддерживает полимаркерный режим, а также переход при необходимости в режим однонаправленного кольца. Ещё в первой декаде 21 века был весьма популярен в США как сеть масштаба кампуса, но сейчас также вытесняется более современными системами типа ATM и 10Gig Ethernet for WAN.

37. Основные понятия и задачи маршрутизации, подход к выбору маршрута, как кратчайшего пути

Маршрутизация в сети передачи данных относится к сетевому уровню семиуровневой модели ВОС. Очевидно, что здесь рассмотрение выходит за рамки одного канала связи, и касается уже сети, то есть, совокупности ряда каналов связи и сетевых узлов, которые эти каналы между собой связывают. Теперь, в общем случае, сообщение от отправителя до получателя должно пройти по нескольким каналам связи, посетив при этом несколько промежуточных узлов.

Двумя главными функциями, которые относят к задачам маршрутизации, является, во-первых, выбор маршрута для доставки сообщения через сеть адресату, и, во-вторых,

доставка по выбранному маршруту этого сообщения. Мы, в основном, будем интересоваться первой функцией, но, как будет видно ниже, обе они иногда так тесно связаны, что разделить их невозможно, и надо рассматривать как единый процесс.

Также, как и управление доступом в канал (разделение канала), маршрутизация оказывает влияние на такие характеристики сети, как пропускная способность и средняя задержка передачи. То, что маршрутизация влияет на задержку передачи, понятно – при прочих равных, если маршрут проложен по наименее загруженному пути, где меньше задержки передачи по составляющим маршрут каналам связи, то и суммарная задержка передачи скорее всего будет меньшей. То, что качество работы механизма маршрутизации может влиять на пропускную способность сети, менее очевидно, поэтому проиллюстрируем это на примере (и заодно, влияние маршрутизации на задержку тоже).

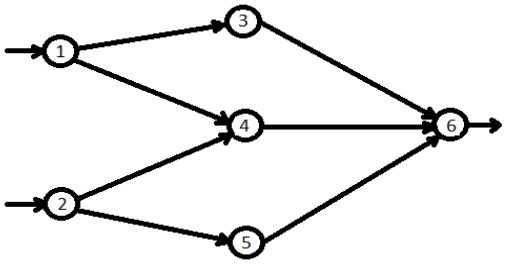

Рисунок 37.1. Пример сети для иллюстрации влияния маршрутизации.

Рассмотрим сеть, схематично изображённую на рис. 37.1 в виде ориентированного графа. Узлы сети – это вершины графа, линии связи – это дуги графа. Пусть все линии связи имеют одинаковую пропускную способность в 10 единиц информационного трафика в секунду (**информационный трафик** – совокупность сообщений, передаваемых с помощью рассматриваемой системы связи; единицы его измерения могут быть разными, и в данном случае не важны).

Пусть в этой сети только два отправителя – один посылает сообщения через узел 1, второй через узел 2, и один получатель, подключённый к узлу 6. Если и у первого, и у второго отправителя потоки сообщений в узел 6 имеют интенсивность по 5 единиц в секунду, то при любой маршрутизации весь суммарный поток пройдёт через сеть, и достигнет адресата. Но если маршрутизация будет такова, что весь поток от узла 1 пойдёт в узел 4, и от узла 2 тоже весь в узел 4, то линия связи 4-6 будет загружена на 100%, очередь в узле 4 (например, в предположении пуассоновских потоков от адресатов) будет большой, и задержка передачи тоже. Если же узел 1 будет посылать весь свой трафик в узел 3, а узел 2 в узел 5, то каждая из используемых линий будет загружена на 50% и задержка уменьшится. Самый же лучший результат будет достигнут, если узел 1 2/3 трафика будет отправлять узлу 3, а 1/3 узлу 4, а узел 2 – 2/3 узлу 5 и 1/3 узлу 4. Тогда каждая линия будет загружена на треть, и задержка будет минимальной.

Пусть теперь от пользователя 1 поступает 15 единиц трафика в секунду. Если узлы 1 и 2 будут посылать узлу 4 столько трафика, сколько позволяют линии связи, узел 4 будет перегружен – входной поток будет превышать пропускную способность исходящей линии, и часть трафика придётся сбросить. Наилучшая маршрутизация в данном случае – это когда узел 2 весь свой трафик посылает через узел 5, а узел один по 50% направляет узлам 3 и 4. Понятно, что в данном случае маршрутизация влияет уже не только на задержку, но и на пропускную способность. В рассматриваемой сети в зависимости от маршрутизации пропускная способность может меняться от 10 до 30 единиц трафика в секунду.

Ещё один вывод, который можно сделать из данного примера таков: для поиска наилучших маршрутов для каждого потока в сети, необходимо проводить глобальную оптимизацию в сети в целом, что практически невозможно.

Мы рассмотрим более простой, и практически осуществимый подход к поиску маршрута между двумя узлами в сети, по которому сообщение будет в большинстве случаев проходить с задержкой, близкой к минимальной. Этот подход называется "**поиск кратчайшего пути**". Будем считать, что для каждой линии связи известна текущая средняя задержка сообщения, и ситуация в сети за время прохождения сообщения через сеть изменяется мало, так что мы можем полагаться на это значение при поиске маршрута. Также мы будем считать, что поток, который мы отправим по рассчитанному маршруту, вносит малый вклад в общий трафик сети, и мало изменяет нагрузки линий, по которым проходит, и, соответственно, среднюю задержку в этих линиях. Тогда, если считать среднюю задержку в линиях связи длиной соответствующих дуг графа сети, то маршрут можно искать, как кратчайший путь по дугам графа между узлом-отправителем и узлом-получателем.

38. Алгоритмы поиска кратчайшего пути на графах

Кратко вспомним основные понятия теории графов, которые будем использовать при формулировке алгоритмов.

Ориентированный граф (дограф) G=(N, A) – это пара объектов, где N – конечное непустое множество вершин графа, а A – конечное множество упорядоченных пар элементов множества N. Каждая такая пара называется **дугой**.

Ориентированный переход по графу G – это конечная последовательность вершин графа, в которой каждые две последовательные вершины – это дуга этого графа.

Ориентированный путь – это ориентированный переход с неповторяющимися вершинами.

Ориентированный цикл – это ориентированный переход, в котором последняя вершина совпадает с первой и больше нет повторяющихся вершин.

Ориентированный граф называется **сильно связным,** если для любой упорядоченной пары вершин существует ориентированный путь из первой вершины во вторую.

Если каждой дуге графа поставлено в соответствие некоторое число, называемое длиной дуги, то **длина ориентированного перехода** – это сумма длин всех входящих в переход дуг. Отсюда следуют определения длины ориентированного пути и ориентированного цикла.

Задача поиска кратчайшего пути в сильно ориентированном графе – это задача отыскания пути между заданной упорядоченной парой вершин, имеющего наименьшую длину среди всех таких путей.

Теперь мы готовы сформулировать несколько наиболее популярных алгоритмов поиска кратчайшего пути на графе. Для упрощения формулировок будем считать, что граф содержит все возможные дуги (i, j), но если на самом деле, какой-то из этих дуг нет, то её длину d_{ij} будем считать равной бесконечности.

Алгоритм Беллмана-Форда. Для упрощения обозначений будем считать, что требуется найти кратчайшие пути от вершины 1 до всех остальных вершин графа. Длины дуг для корректной работы этого алгоритма могут быть как положительными, так и отрицательными, но в графе не должно быть циклов отрицательной длины. Основная идея алгоритма состоит в том, чтобы сначала найти длины кратчайших путей при условии, что в пути не более одной дуги, потом длины кратчайших путей при условии, что в пути не более двух дуг и т.д.

Путь при условии, что он содержит не более h дуг, будем называть h-путём. Длину кратчайшего h-пути из вершины 1 до вершины I будем обозначать $D_i^{(h)}$. Будем считать, что $D_1^{(h)} = 0$ для всех h и d_{ii}=0 для всех i. Действия алгоритма состоят в следующем:

1). Начальные условия $D_i^{(1)} = d_{1i} \quad \forall i \neq 1$, кратчайший 1-путь из 1 в i – это (1,i).

2). Повторить N-2 раза, где N – число вершин графа, следующую итерацию:

$$D_i^{(h+1)} = \min_k \left(D_k^{(h)} + d_{ki} \right) \quad \forall i \neq 1, \tag{162}$$

и если min в (162) достигается при $k \neq i$, то кратчайший (h+1)-путь из 1 в i – это кратчайший h-путь из 1 в k, и потом переход в i; если же min достигается при $k = i$, то кратчайший (h+1)-путь из 1 в i – это кратчайший h-путь из 1 в i.

Доказательство корректности алгоритма проведём по индукции**:**

1). $D_i^{(1)} = d_{1i} \quad \forall i \neq 1$ - и это действительно длины кратчайших путей, состоящих не более чем одной дуги.

2). Пусть для некоторого h величины $D_i^{(h)}$ являются длинами кратчайших h-путей. Докажем, что полученные согласно (162) $D_i^{(h+1)}$ являются длинами кратчайших (h+1)-путей. Сначала покажем, что если $D_i^{(h+1)}$ - длина кратчайшего (h+1)-пути, то

$$D_i^{(h+1)} \geq \min_k \left(D_k^{(h)} + d_{ki} \right) \quad \forall i \neq 1 \tag{163}$$

Действительно, если (1,…, k, i) – кратчайший (h+1)-путь, то его длина есть минимальная длина h-пути от 1 до k плюс d_{ki}, поэтому

$$D_i^{(h+1)} = D_k^{(h)} + d_{ki} \geq \min_j \left(D_j^{(h)} + d_{ji} \right) \quad \forall i \neq 1$$

по определению минимума, и (163) доказано.

Теперь докажем, что

$$D_i^{(h+1)} \leq \min_k \left(D_k^{(h)} + d_{ki} \right) \quad \forall i \neq 1. \tag{164}$$

Пусть минимум правой части достигается при некотором k. Тогда, если (1, …, k, i) – это некоторый (h+1)-путь из 1 в i, то длина этого (h+1)-пути, как мы уже замечали ранее, есть минимальная длина пути от 1 до k плюс d_{ki}, а длина минимального (h+1)-пути из 1 в i не может быть больше этой величины по определению.

Если же (1,…,k,i) не является путём, то это означает, что где-то между вершиной 1 и вершиной k вершина i уже встречалась (другие вершины два раза там встречаться не могут, потому что до добавления вершины i это был путь по предположению индукции). Так что (1,…,k,i) – это переход, где сначала идёт путь от 1 до первого появления i, а потом цикл до второго появления i. Но по условию корректности алгоритма, в графе нет циклов отрицательной длины, и потому (164) выполняется. Одновременное выполнение (163) и (164) возможно только при выполнении (162). В заключение доказательства добавим, что в графе из N вершин путь не может содержать больше N-1 дуги, и потому N-2 повторения итерации (162) дадут истинные длины кратчайших путей из 1 во все вершины графа.■

Количество вычислений, необходимых для расчётов кратчайших путей для всех возможных пар отправитель-получатель при использовании данного алгоритма можно оценить следующим образом: алгоритм должен быть исполнен для каждого из N отправителей, в каждом запуске выполняется в наихудшем случае N-2 итерации, и в каждой итерации происходит вычисление (162) для N-1 вершины, и прир каждом вычислении ищется минимум среди N-1 числа. Итого O(N^4). Более точный расчёт (см. [9]) даёт O(N*m*A), где A – число дуг в графе, а m – количество дуг в максимальном пути. В разреженных графах (где мало дуг по сравнению с их максимальным числом), такое количество операций может быть гораздо меньше наихудшего случая. И именно такие графы часто встречаются на практике.

Вообще нетрудно догадаться, что если при работе алгоритма оказалось, что $D_i^{(h+1)} = D_i^{(h)} \quad \forall i$, то последующие итерации не изменят длин кратчайших путей и алгоритм можно останавливать. Но если мы будем ориентироваться именно на данное событие, как момент остановки алгоритма, в графах с циклами отрицательной длины этот

момент никогда не наступит. Давайте разберём, как себя будет вести алгоритм, если в графе присутствуют циклы отрицательной длины, на конкретном примере.

Рисунок 38.1. Пример графа с циклом отрицательной длины.

Для графа с рисунка 37.2 длина перехода из вершины 1 в вершину 2, состоящего из одной дуги, равна 1, а длина перехода (1,2,3,2) из трёх дуг равна -1. То есть, с ростом h, длина перехода уменьшается, и алгоритм Беллмана-Форда никогда не остановится.

Существуют ли отрицательные циклы в конкретном графе, можно определить, сравнивая $D_i^{(N-1)}$ и $D_i^{(N)}$ для всех i, то есть результат последней итерации алгоритма, необходимой для отыскания кратчайших длин в наихудшем случае, и ещё одной дополнительной итерации. Если $D_i^{(N)} = D_i^{(N-1)} \quad \forall i$, то в графе нет отрицательных циклов, и результат получен верный. В противном случае, цикл отрицательной длины присутствует, и на данном графе работа алгоритма некорректна.

Одной из привлекательных для практики черт алгоритма Беллмана-Форда является то, что если в графе длины всех дуг положительны, то итерации алгоритма для разных узлов могут выполняться независимо друг от друга и параллельно. Это имеет большое значение для реализации так называемых **распределённых алгоритмов маршрутизации**, о которых мы поговорим позже. Именно тогда нам понадобится следующее важное свойство алгоритма Беллмана-Форда, которое мы примем без доказательства.

Свойство: Если в графе нет циклов отрицательной и нулевой длины, то существует единственный набор D_i, i=1,…,N, являющийся решением системы уравнений

$$\begin{cases} D_1 = 0 \\ D_i = \min\limits_k \left(D_k + d_{ki}\right) \quad \forall i \neq 1, \end{cases} \tag{165}$$

и этот набор есть длины кратчайших путей от вершины 1 во все другие вершины графа.

Алгоритм Дийкстра. Этот алгоритм для корректной работы требует, чтобы длины всех дуг были положительны. Для графов, изображающих сети телекоммуникаций, где в качестве длины дуги выбирается, например, задержка передачи по соответствующей линии связи, это выполняется. Объем вычислений для худшего случая у этого алгоритма гораздо меньше, чем у алгоритма Беллмана-Форда.

Основная идея алгоритма Дийкстра – отыскание кратчайших путей в порядке возрастания их длин. Так, кратчайшим путём из вершины 1 является путь, состоящий из самой короткой исходящей дуги, потому что если он ведёт в вершину k, то все остальные пути в k будут длиннее – дуга из 1 ещё куда-то (скажем, в j) уже длиннее, чем дуга из 1 в k, а ещё дуга из j в k добавит длины пути (ведь все длины дуг положительны). Следующим по длине из кратчайших путей из вершины 1, будет либо вторая по длине исходящая из 1 дуга, либо путь из двух дуг из 1 в k и из k ещё куда-то, и т.д.

Чтобы формально описать данный алгоритм, будем считать, что у каждой вершины i имеется метка D_i, которая равна оценке длины кратчайшего пути из 1 в i. Когда эта оценка перестаёт изменяться, будем считать, что вершина *окончательно помечена*, и множество окончательно помеченных вершин обозначим через $\boldsymbol{P}$. На каждом шаге алгоритма та из ещё не входящих в $\boldsymbol{P}$ вершин, которая ближе всего к 1, добавляется в $\boldsymbol{P}$.

Формальное описание:

1). Начальные условия: $\boldsymbol{P} = \{1\}$, $D_1 = 0$, $D_k = d_{1k}$, текущий кратчайший путь в вершину к – это дуга $(1,k)$ $\forall k \neq 1$.

2). Поиск следующей ближайшей вершины: найти вершину $i \notin P : D_i = \min\limits_{k \notin P} D_k$ и добавить её в $\boldsymbol{P}$. Текущий кратчайший путь в вершину i – это истинный кратчайший путь. Если в результате $\boldsymbol{P}$ содержит все вершины графа, закончить работу, в противном случае перейти к пункту 3.

3). Обновление меток: $\forall k \neq P \quad D_k = \min\left(D_k, D_i + d_{ik}\right)$, где i – вершина, добавленная в $\boldsymbol{P}$ в пункте 2. Если D_k при этом не меняется, то текущий кратчайший путь в вершину k не меняется, в противном случае текущим кратчайшим путём в вершину k становится кратчайший путь в вершину i и далее дуга (i, k). После обновления всех меток, перейти к пункту 2.

Доказательство корректности алгоритма. Заметим, что на каждой итерации перед началом выполнения пункта 2 для любой вершины, ещё не вошедшей в $\boldsymbol{P}$, его оценка D_i, - это длина кратчайшего пути среди тех путей, у которых все дуги, кроме последней, соединяют вершины из $\boldsymbol{P}$, и только последняя дуга выходит из $\boldsymbol{P}$ в эту вершину. Действительно, после пункта 1 это, очевидно, имеет место, и каждый раз после окончания очередного выполнения пункта 3 это свойство сохраняется (при пересчёте меток путь либо остаётся таким же, либо становится путём через вершину, которая только что была включена в $\boldsymbol{P}$).

Теперь заметим, что по построению алгоритма любая вершина из $\boldsymbol{P}$ ближе (не дальше) к узлу 1, чем вершина не из $\boldsymbol{P}$. На некоторой итерации алгоритма рассмотрим вершину i не из $\boldsymbol{P}$, которая, согласно значениям меток, ближе всех к вершине 1. Так как длины всех дуг по условию положительны, то кратчайший путь из 1 в i должен проходить по узлам из P, и только последняя дуга выходит из $\boldsymbol{P}$ в i. Действительно, если выйдя из $\boldsymbol{P}$, путь идёт сначала не в i, а ёщё куда-то, скажем, в j, и только потом в i, то он уже до j имеет большую длину D_j, чем D_i, а ещё туда добавится длина дуги d_{ji}. Именно эта вершина i и будет включена в $\boldsymbol{P}$ самой первой из оставшихся вне $\boldsymbol{P}$ согласно алгоритму. А когда в $\boldsymbol{P}$ войдут все вершины графа, тогда все текущие кратчайшие пути станут истинными кратчайшими путями. ■

Число итераций алгоритма N-1, на каждой итерации поиск минимума и пересчёт меток требуют количество операций, пропорционально N, то есть, $O(N^2)$ для путей из одной вершины во все остальные. Для всех пар вершин надо запустить алгоритм N раз, то есть, количество операций $O(N^3)$. Это гораздо меньше, чем наихудший случай у алгоритма Беллмана-Форда, но как мы замечали ранее, на практике алгоритм Беллмана-Форда часто требует гораздо меньше вычислений, и может оказаться даже быстрее алгоритма Дийкстра.

Алгоритм Флойда-Уоршела. Этот алгоритм, в отличие от двух предыдущих, находит кратчайшие пути сразу между всеми парами вершин графа. Как и в алгоритме Беллмана-Форда, длины дуг могут быть неположительными числами, но не должно быть циклов отрицательной длины. Основная идея алгоритма – итерации по множеству вершин, которые могут быть промежуточными на пути. Начинается алгоритм с путей, состоящих из одной дуги, т.е. без промежуточных вершин. Затем вычисляются кратчайшие пути, у которых промежуточным может быть только вершина 1, затем – только вершины 1 и 2, и т.д. Когда в множество возможных промежуточных вершин войдут все вершины графа, алгоритм закончит работу.

Для формального описания алгоритма обозначим через $D_{ik}^{(n)}$ длину кратчайшего пути от вершины i к вершине k, у которого промежуточными могут быть только вершины $1,2,\ldots,n$. Действия алгоритма:

1). Начальные условия: $D_{ik}^{(0)} = d_{ik} \quad \forall i,k : i \neq k$, и текущий кратчайший путь – это дуга (i,k).

2). *N* раз повторить следующую итерацию:

$$D_{ik}^{(n+1)} = \min\left(D_{ik}^{(n)},\, D_{i,n+1}^{(n+1)} + D_{n+1,k}^{(n+1)}\right) \quad \forall i,k : i \neq k\,, \tag{166}$$

при этом, если минимум достигается на первом выражении в скобках, то текущий кратчайший путь для данной пары узлов не меняется, если минимум достигается на втором выражении, текущий кратчайший путь от *i* до *k* составляется из двух кратчайших путей – сначала от *i* до *n*+1, потом от *n*+1 до *k*.

Доказательство корректности алгоритма проведём по индукции. При *n*=0 начальными условиями являются длины кратчайших путей, где нет промежуточных вершин, то есть множество возможных промежуточных вершин пусто. Предположим, что для некоторого *n* величины $D_{ik}^{(n)}$, полученные в результате работы алгоритма, это длины кратчайших путей, промежуточными вершинами на которых могут быть вершины с номерами не больше *n*. Кратчайший путь от *i* до *k*, использующий в качестве промежуточных, вершины с номерами не больше *n*+1, либо проходит через вершину *n*+1, либо нет. В первом случае такой путь идёт сначала от *i* до *n*+1, а потом от *i*+1 до *k* и длина его $D_{i,n+1}^{(n)} + D_{n+1,k}^{(n)}$. Во втором случае это путь, использующий в качестве промежуточных вершины с номерами не более *n*, и его длина $D_{ik}^{(n)}$. Выбирая из этих двух случаев минимальный, получаем $D_{ik}^{(n+1)}$. Через *N* итераций в множестве промежуточных вершин будут все вершины графа. ■

Пар вершин в графе $N(N\text{-}1)$, и для каждой пары проводится *N* итераций, так что объём вычислений для алгоритма Флойда-Уоршела равен $O(N^3)$, то есть, как у алгоритма Дийкстра, повторённого для всех вершин графа, выбранных за начало пути.

39. Особенности работы алгоритмов маршрутизации в реальных сетях

Рассмотренные выше алгоритмы – это математическая основа маршрутизации в ситуации, когда она может быть осуществлена как поиск кратчайшего пути. Но при использовании их в реальных сетях придётся учитывать целый ряд обстоятельств.

Во-первых, в реальных сетях нагрузка линий связи постоянно меняется. Во-вторых, линии могут вообще выходить из строя, и через некоторое время входить обратно в строй. Всё это означает, что маршрутизация в сети должна быть динамической (когда маршруты могут постоянно изменяться), и осуществляться непрерывно.

Другой важный аспект реальной маршрутизации состоит в том, что информация, необходимая для маршрутизации, распределена по всей сети. То есть, возникает проблема сбора этой информации и доведения её до тех, кому она нужна. Причём и этот процесс должен быть постоянным.

С данным аспектом связан и ещё один важный вопрос реальной маршрутизации: какая именно информация требуется, и кто именно рассчитывает маршруты для каждого узла сети? Некоторые алгоритмы для выполнения требуют информацию обо всей сети, а сбор и последующее доведение такой информации до всех узлов – это большие накладные расходы (так что кажется, что лучше сделать один управляющий центр, куда будет сходиться информация, и который будет считать всем маршруты). Выполнение других алгоритмов может быть распределено по узлам сети таким образом, чтобы каждому узлу нужна была бы только информация об окружающем его участке сети, а полной информации не требуется. Кажется, что при этом и надёжность выше, так как сеть может сохранять работоспособность при выходе из строя отдельных узлов, и накладные расходы на пересылку информации, необходимой для маршрутизации, меньше. А вот так ли это на самом деле, надо рассмотреть повнимательнее.

Мы рассмотрим некоторые подходы к решению этих проблем, и начнём с модельной ситуации для рассмотрения теоретической возможности распределённых вычислений при маршрутизации на примере распределённого алгоритма Беллмана-Форда.

40. Распределённый синхронный алгоритм Беллмана-Форда

Для того, чтобы подчеркнуть, что с этого момента и далее речь будет идти о маршрутизации в сети связи, а не просто об абстрактном графе, будем использовать терминологию сетей, то есть вместо “вершина графа” будем говорить “узел сети”, вместо “дуга графа“ – “линия связи” и т.д.

При рассмотрении данного алгоритма нам будет удобно представлять, что он ищет не кратчайший путь от узла 1 до какого-то другого узла сети, а наоборот, от произвольного узла сети к узлу 1. Мы покажем, что для корректного выполнения данного алгоритма каждый узел должен знать, во-первых, номера узлов, находящихся в сети, и во-вторых, длины (т.е. задержки передачи) исходящих из него линий связи. Тогда он может производить все необходимые согласно этому алгоритму расчёты, обмениваясь некоторой информацией только с непосредственными соседями по сети (то есть, с теми узлами, с которыми непосредственно связан линиями связи).

Для простоты будем, как и ранее, считать, что сеть всегда остаётся сильно связным графом, а также, что длины всех линий связи положительны и не изменяются после момента времени, который мы примем за нулевой.

Обозначая теперь через $D_i^{(h)}$ длину кратчайшего h-пути из узла i в узел 1, итерации алгоритма Беллмана-Форда можно записать, как

$$\begin{cases} D_1^{(h+1)} = 0 \\ D_i^{(h+1)} = \min\limits_k \left(d_{ik} + D_k^{(h)} \right) \quad \forall i \neq 1, \end{cases} \tag{166}$$

Будем считать, что каждый узел (включая и узел 1) абсолютно одновременно рассылает свою текущую оценку $D_i^{(h)}$ по всем соседям, одновременно получая от них их оценки, и после этого производит итерацию (166). Потом через определённый промежуток времени это повторяется. В качестве начальных условий давайте возьмём

$$D_1^{(0)} = 0, \quad D_i^{(0)} = \infty \ \forall i \neq 1. \tag{167}$$

Тогда при начале первой итерации все соседи узла 1 получат от него $D_1^{(0)} = 0$, и, вычислив (166), получат

$$D_i^{(1)} = d_{i1} \ \forall i \neq 1. \tag{168}$$

Все прочие узлы получат оценки, равные бесконечности, и в результате получат опять бесконечность. Заметим, что (167) – это те начальные условия, с которыми мы доказали корректность алгоритма Беллмана-Форда ранее, так что он остаётся корректным, просто становится длиннее на одну итерацию.

В результате первой итерации соседи узла 1 узнают истинные кратчайшие длины пути к узлу 1. При начале второй итерации соседи соседей узла 1 получат их и по ним вычислят свои истинные длины кратчайших путей, и т.д. Максимум через *N* итераций истинные длины до узла 1 рассчитают все узлы сети.

Обратите внимание, что каждый узел сети (кроме непосредственных соседей узла 1), не знает всего кратчайшего маршрута до узла 1. Он знает только длину этого маршрута, и того соседа, от которого получил оценку, которая и дала ему эту длину как минимум выражения (166). Но ему весь маршрут и не нужен: если ему надо переслать сообщение узлу 1, то всё, что ему надо знать – это исходящую от него линию связи, которая лежит на кратчайшем пути, и по которой это сообщение надо отправить. А её-то он как раз и знает – это линия к тому соседу, чьё присланное $D_k^{(h)}$ обеспечило минимум выражения (166).

К сожалению, в реальной сети реализовать такой синхронный алгоритм чрезвычайно трудно (практически невозможно). Во-первых, крайне трудно обеспечить идеальную синхронизацию между всеми узлами сети. Во-вторых, в реальной сети загрузки линий связи, а значит и задержки передачи, то есть “длины” линий могут

меняться, и все узлы должны опять-таки синхронно реагировать на это событие, перезапуская расчёт маршрутов. А ведь о том, что изменилась длина линии знают только узлы, которым она инцидентна, а до других эту информацию надо как-то доводить.

Далее мы рассмотрим практически реализуемую альтернативу синхронному распределённому алгоритму Беллмана-Форда.

41. Асинхронный распределённый алгоритм Беллмана-Форда

Рассмотрим вариант распределённого алгоритма Беллмана-Форда, в котором работа всех узлов не должна быть синхронизирована и которому не требуется начинать свою работу с начальных условий (167). Это делает ненужными и синхронизацию всех узлов, и повторные перезапуски при изменении длин линий связи.

Алгоритм полностью асинхронный, то есть в каждом узле он работает независимо от других узлов, время от времени выполняя итерацию $D_i = \min_k \left(d_{ik} + D_k\right)$, где оценки D_k он получает только от своих соседей, использует их для вычисления своей оценки, и рассылает их только по своим соседям. Причём, и в проведении итераций никакая синхронизация с соседями не нужна, нужно лишь, чтобы узел производил каждую такую итерацию и рассылку за конечное время, и никогда не прекращал этот процесс.

Наша цель состоит в том, чтобы показать, что если после некоторого момента времени (который мы примем за 0), никакие изменения длин линий не происходят, то за конечное время такой асинхронный распределённый алгоритм найдёт правильные кратчайшие расстояния до узла 1 от каждого узла сети.

Формально определим алгоритм, потом докажем его корректность. Будем считать, что сеть представлена сильно связным графом. Для каждого узла i обозначим через $N(i)$ множество его соседей, то есть узлов, непосредственно соединённых с ним линиями связи. В каждый момент времени t в каждом узле i хранятся:

1) $D_k^i(t) \quad \forall k \in N(i)$ - последние полученные от соседей оценки кратчайшего расстояния до узла 1;
2) $D_i(t)$ - своя оценка кратчайшего пути до узла 1, вычисленная во время последней итерации алгоритма в этом узле.

Для узла 1 $D_1(t) = 0 \; \forall t$ и $D_1^i(t) = 0 \; \forall i \in N(1), \forall t$. Также в каждом узле i имеются $d_{ik} \; \forall k \in N(i)$ - длины линий ко всем соседям. Они все положительны и не меняются при $t \geq 0$.

Существует бесконечная возрастающая последовательность моментов времени, которую мы обозначим $T^{[i]}$, когда узел i производит обновление своей оценки согласно формуле

$$D_i(t) = \min_{k \in N(i)} \left(d_{ik} + D_k^i(t)\right). \qquad (169)$$

В остальные моменты времени $D_i(t)$ не изменяется. Будем считать, что промежуток между соседними моментами времени в $T^{[i]}$ не может превышать некоторой константы. Также будем считать, что каждый узел после проведения итерации (169) обязательно рассылает результат по своим соседям, не обязательно всем одновременно, но каждому только один раз и до того, как сделает новую итерацию. Будем считать, что все такие оценки, рассылаемы узлами, всегда доходят до адресатов, и без искажений.

Обозначим через $T_{[k]}^{[i]}, k \in N(i)$ множество моментов времени, когда узел i получает от узла k очередное $D_k^i(t)$, и обновляет его у себя. В остальные моменты времени $D_k^i(t)$ не изменяется. Нетрудно видеть, что все $T_{[k]}^{[i]}$ также бесконечные возрастающие

последовательности моментов времени, причём промежуток времени между соседними моментами в этой последовательности ограничен некоторой константой.

В момент времени t=0, когда система начинает работу, все хранящиеся во всех узлах начальные значения $D_i(0)$ и $D_k^i(0)$ неотрицательны.

Утверждение: При сформулированных выше условиях существует конечный момент времени t_m, такой что $D_i(t) = D_i \ \forall i, \forall t \geq t_m$, где D_i - истинные длины кратчайших путей от узла i до узла 1.

Доказательство: Основная идея доказательства состоит в том, чтобы сначала для каждого узла i определить две последовательности $\{\underline{D}_i^n\}$ и $\{\overline{D}_i^n\}$, для которых будут справедливы следующие неравенства:

$$\underline{D}_i^n \leq \underline{D}_i^{n+1} \leq D_i \leq \overline{D}_i^{n+1} \leq \overline{D}_i^n \ , \tag{170}$$

и для достаточно большого n

$$\underline{D}_i^n = D_i = \overline{D}_i^n . \tag{171}$$

Эти последовательности получаются при итерациях синхронного распределённого алгоритма Беллмана-Форда, стартующего из двух разных начальных условий. Потом покажем, что для любого n можно найти конечное t, такое, что

$$\underline{D}_i^n \leq D_i(t) \leq \overline{D}_i^n \tag{172}$$

Для реализации этой идеи рассмотрим две совокупности начальных условий для синхронного алгоритма Беллмана-Форда:

1). $D_1^{(0)} = 0, \quad D_i^{(0)} = \infty \quad \forall i \neq 1;$

2). $D_i^{(0)} = 0 \quad \forall i = 1,\ldots N .$

Покажем, что $\overline{D}_i^n$ - это n-ная итерация синхронного алгоритма Беллмана-Форда, стартовавшего из начальных условий 1, а $\underline{D}_i^n$ - это n-ная итерация синхронного алгоритма Беллмана-Форда, стартовавшего из начальных условий 2, то есть докажем, что для этих последовательностей выполняются (170) и (171).

Для этого прежде всего заметим, что итерации алгоритма Беллмана-Форда обладают свойством монотонности, а именно, если имеются два набора $\{\underline{D}_i\}$ и $\{\overline{D}_i\}$, i=1,…,N, таких, что

$$\underline{D}_i \leq \overline{D_i} \quad \forall i = 1,..., N , \tag{173}$$

то знак этого неравенства сохранится и после проведения итерации алгоритма Беллмана-Форда с каждым из этих наборов, т.е.

$$\min_{k \in N(i)} \left(d_{ik} + \underline{D}_k\right) \leq \min_{k \in N(i)} \left(d_{ik} + \overline{D}_k\right). \tag{174}$$

Отсюда, в частности, следует, что если, стартуя из каких-то начальных условий $\{D_i^{(0)}\}$, i=1,…N, и получив после первой итерации набор $\{D_i^{(1)}\}$, i=1,…N, мы обнаружим, что для этих наборов справедливо $D_i^{(1)} \leq D_i^{(0)} \quad \forall i = 1,..., N$, то вся последовательность $\{D_i^{(n)}\}$, n=0,1,… будет невозрастающей для любого i, то есть $D_i^{(n+1)} \leq D_i^{(n)} \ \forall n$. Чтобы увидеть, что это так, возьмите сначала в качестве $\{\overline{D}_i\}$ набор $\{D_i^{(0)}\}$, а в качестве $\{\underline{D}_i\}$ набор $\{D_i^{(1)}\}$, и далее по индукции в качестве $\{\overline{D}_i\}$ набор $\{D_i^{(n)}\}$, а в качестве $\{\underline{D}_i\}$ набор $\{D_i^{(n+1)}\}$.

Отсюда, в свою очередь, следует, что $\overline{D}_i^n$, полученная как результат итераций, стартовавших из начальных условий 1, действительно неубывающая, так как в начале все оценки бесконечные (кроме i=1), а после первой итерации, как мы видели при рассмотрении синхронного алгоритма Беллмана-Форда, соседи узла 1 получают оценки, равные длинам линий до узла 1, которые меньше бесконечности. Таким образом, $\overline{D}_i^1 \leq \overline{D}_i^0$,

и доказано самое правое неравенство в (170). Также доказано и второе справа неравенство в (170), так как очевидно, что $D_i \le \overline{D}_i^0 \quad \forall i = 1,...,N$, а результатом применения итераций алгоритма к истинным длинам кратчайших путей D_i, являются, согласно (165), эти же D_i.

Аналогично, если при старте с некоторых начальных условий получилось, что $D_i^{(1)} \ge D_i^{(0)} \quad \forall i = 1,...,N$, то вся последовательность $\{D_i^{(n)}\}$, n=0,1,…,будет неубывающей для любого i, и это доказывает оставшиеся два неравенства в (170), так как с очевидностью, $\underline{D}_i^0 \le \underline{D}_i^1$, и вся последовательность $\{\underline{D}_i^n\}, i = 0,1,...$ будет неубывающей, а также $D_i \ge \underline{D}_i^0 \quad \forall i = 1,...,N$, и значит, $D_i \ge \underline{D}_i^n$ для любого n.

Перейдём теперь к доказательству двух равенств в (171). Правое равенство мы уже доказали, когда рассматривали синхронный алгоритм Беллмана-Форда. Осталось доказать, что итерации алгоритма, стартовавшие из нулевых начальных условий тоже сойдутся за конечное число шагов к истинным длинам кратчайших путей. Для этого заметим, что $\underline{D}_i^n$ для любого n является суммой не более, чем n длин линий, каждая из которых положительное число. Действительно, для $\underline{D}_i^0$=0 это выполняется, и далее на каждой итерации может прибавляться не более 1 слагаемого. При этом, мы имеем, что $\underline{D}_i^n \le D_i \ \forall n$, и поэтому в величину $\underline{D}_i^n$ могут внести вклад не более, чем $\frac{\max D_i}{\min d_{ik}}$ слагаемых, а возможные величины слагаемых – это длины линий связи, которых конечное число. Поэтому число возможных значений $\underline{D}_i^n$ конечно. Так как $\{\underline{D}_i^n\}$- монотонно неубывающая последовательность, то начиная с некоторого m будет $\underline{D}_i^m = \underline{D}_i^{m+1} \ \forall i = 1,...N$, то есть, итерации алгоритма Беллмана-Форда начнут выдавать одни и те же значения. Но, как мы знаем из (165), набор чисел, обладающий таким свойством, всего один – это истинные длины кратчайших путей D_i. То есть, начиная с этого m $\underline{D}_i^m = D_i \ \forall i = 1,...N$, и левое равенство в (171) доказано.

Осталось доказать (172). Сделаем это по индукции. В начальный момент времени выполняются неравенства

$$\underline{D}_i^0 \le D_i(0) \le \overline{D}_i^0,$$

$$\underline{D}_k^0 \le D_k^i(0) \le \overline{D}_k^0$$

по предположению о неотрицательности начальных значений $D_i(0)$ и $D_k^i(0)$. Теперь пусть для некоторого n и некоторого конечного t_n выполняются

$$\underline{D}_i^n \le D_i(t_n) \le \overline{D}_i^n,$$

$$\underline{D}_k^n \le D_k^i(t_n) \le \overline{D}_k^n.$$

Рассмотрим узел сети, который после момента t_n самым первым среди всех узлов проводит итерацию алгоритма в момент $t_{n+1}^{(i)} > t_n$ и потом рассылает результаты по соседям. Рассуждая аналогично предыдущим частям доказательства, нетрудно видеть, что для этого узла

$$\underline{D}_i^{n+1} \le D_i(t_{n+1}^{(i)}) \le \overline{D}_i^{n+1}, \qquad (175)$$

и по соседям он рассылает оценки, для которых справедливо

$$\underline{D}_i^{n+1} \le D_i^k(t^*) \le \overline{D}_i^{n+1}, \qquad (176)$$

где t^* - момент получения адресатом этой оценки.

Рассмотрим другой узел, который, перед тем, как сделать первую после момента t_n, итерацию, успевает получить от соседей одно или более обновлений их текущей оценки. Эти обновления удовлетворяют (176), но так как

$$\underline{D}_i^n \le \underline{D}_i^{n+1} \le \overline{D}_i^{n+1} \le \overline{D}_i^n ,$$

то, очевидно, что

$$\underline{D}_i^n \le D_i^k(t^*) \le \overline{D}_i^n , \qquad (176)$$

и итерация, которую будет проводить этот узел, даст результат, удовлетворяющий (175). Нетрудно видеть, что аналогичные рассуждения применимы и к случаю, когда один из узлов делает несколько итераций за то время, пока соседи ещё не сделали даже одну. Присланные “быстрым” узлом “медленным” соседям оценки будут удовлетворять (176), и итерации “медленных” соседей, когда они их всё-таки сделают, будут удовлетворять (175). Так как по условию, все узлы все-таки сделают одну итерацию и разошлют её результаты в течении конечного интервала времени, то в момент t_{n+1} получения последним адресатом последней такой оценки станет справедливо

$$\underline{D}_i^{n+1} \le D_i(t_{n+1}) \le \overline{D}_i^{n+1} ,$$

$$\underline{D}_k^{n+1} \le D_k^i(t_{n+1}) \le \overline{D}_k^{n+1} . \blacksquare$$

Заметим, что количество итераций и пересылок результатов асинхронным алгоритмом не равно числу итераций и пересылок “ограничивающих” синхронных алгоритмов и может быть много больше.

К главным недостаткам распределённого асинхронного алгоритма Беллмана-Форда именно то, что в некоторых ситуациях он может потребовать слишком много итераций и пересылок, для того, чтобы оценки в узлах сошлись к истинным значениям. Рассмотрим пример сети, изображённой на рис. 41.1. В начальный момент линия 4-1 имеет длину 1, другие длины линий указаны на рисунке. Во всех узлах сети в начальный момент оценки равны истинным кратчайшим расстояниям до узла 1 (D_2=3, D_3=2, D_4=1).

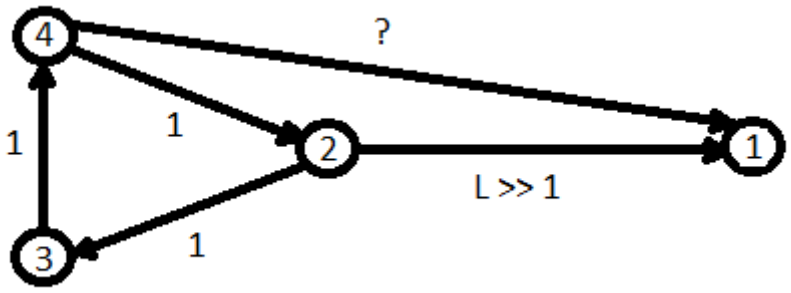

Рисунок 41.1. Пример сети для демонстрации недостатков асинхронного распределённого алгоритма Беллмана-Форда.

Предположим, что линия 4-1 внезапно выходит из строя (её длина становится равной бесконечности). Заметивший это узел 4 во время ближайшей итерации пересчитает свои оценки, и получит, что кратчайший путь в 1 теперь проходит через узел 2, и D_4=4. На следующей итерации узел 3 заметит, что длина пути до 1 изменилась, и вычислит, что D_3=5. На следующей итерации узел 2 вычислит, что теперь D_2=6, но маршрут, с его точки зрения, идёт всё также через узел 3, так как L, будем считать, много больше 6, и т.д. То есть, узлам 2,3,4 вместе потребуется порядка L итераций, чтобы найти новый правильный путь. До этого все сообщения, адресованные узлу 1, будут направляться по циклу 2-3-4.

Ещё хуже будет обстоять дело, если выход из строя линии связи разделит сеть на несвязанные части (например, если линии 2-1 на рис. 41.1 нет вообще). Нетрудно видеть, что асинхронный распределённый алгоритм Беллмана-Форда в том виде, в котором мы его рассматривали, будет распознавать такую ситуацию за бесконечное время, и все сообщения, адресованные из одной части в другую, будут направляться по циклам.

Вообще, медленная реакция на резкое изменение ситуации в сети – это общий недостаток распределённых асинхронных алгоритмов. Некоторые подходы по его преодолению мы рассмотрим ниже.

42. Распространение информации о сетевой топологии

Итак, информация о резком изменении ситуации на каком-либо участке сети имеет чрезвычайно важное значение для процесса маршрутизации. Прежде всего, к такого рода информации относятся сведения о выходе из строя или входе в строй сетевых узлов и линий связи. Эта информация называется **информацией о сетевой топологии**. К распространению по сети такой информации предъявляются особые требования. Мы рассмотрим некоторые подходы к решению этой задачи. Сразу заметим, что эти подходы применимы и к распространению по сети другой важной информации. В частности, если вместо распределенных алгоритмов маршрутизации, будут использоваться т.н. **локальные**, где каждый рассчитывающий маршрут узел должен иметь всю информацию о состоянии сети и рассчитывать весь маршрут (о них поговорим позже), тогда по сети необходимо распространять такую информацию, и к ней можно относиться также, как к топологической.

Как обычно, сначала сформулируем условия, в которых мы будем проводить рассмотрение. Они определят идеализированную модель для проверки работоспособности подхода.

1. По работающим линиям связи сообщения передаются без ошибок и с сохранением очерёдности. В узлах сети сообщения хранятся также без искажений.
2. Выход из строя линии обнаруживается на обоих её концах, хотя не обязательно одновременно. Под этим понимается то, что если узел на одном конце объявляет, что линия вышла из строя, то второй концевой узел также объявит, что она вышла из строя до того, как первый узел объявит, что эта линия снова вошла в строй.
3. Имеется некоторый механизм в концевых узлах для регистрации момента времени, когда вышедшая из строя линия снова входит в строй. Если один из концевых узлов объявляет, что данная линия вошла в строй, то в течение конечного интервала времени либо второй концевой узел объявит об этом же, либо первый узел объявит, что линия снова вышла из строя.
4. Узел может выйти из строя, и в этом случае все примыкающие к нему линии объявляются вышедшими из строя. Это делают узлы на других концах этих линий в течение конечного интервала времени.

Мы будем считать эти предположения выполненными, но на практике они справедливы не всегда. Например, в редких случаях пункт 1 может нарушаться из-за того, что искаженные кадры данных не были замечены при декодировании. Поэтому, в дополнение к анализу обычного случая, когда эти предположения выполняются, мы также уделим некоторое внимание исключительным ситуациям, когда они нарушаются.

Заметим, что корректный алгоритм распространения топологической информации должен в обычных условиях работать для всех возможных топологий, то есть, например, быть способным восстановить работоспособность сети из состояния, когда все линии связи вышли из строя, а потом начинают входить в строй.

Лавинный алгоритм. Его основная идея состоит в том, что как только какой-либо узел фиксирует значимое изменение статуса какой-либо смежной линии (например, с рабочей на нерабочую или обратно), он тут же посылает соответствующее сообщение всем соседям. Те, получив его, пересылают своим соседям, кроме того, от которого его получили, и т.д. Таким образом, предполагается, что сообщение максимально быстро расходится по сети. Однако, нетрудно понять, что в таком виде алгоритм имеет весьма существенный недостаток. Рассмотрим, например, сеть, изображённую на рис. 42.1.

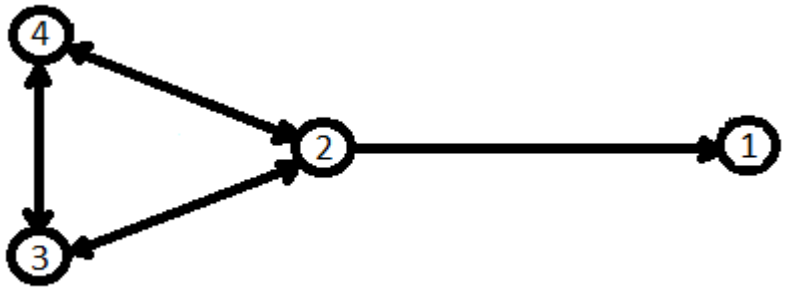

Рисунок 42.1. Пример сети для демонстрации некорректности простейшей версии лавинного алгоритма.

Пусть линия 1-2 сначала была рабочей, а потом вышла из строя. Узел 2 рассылает сообщение об этом узлам 3 и 4, они посылают его друг другу, потом узлу 2, и далее понятно, что в данной сети возникнет бесконечно долгое хождение сообщение по кругу, причём в обоих направлениях. То есть, простейший лавинный алгоритм может использоваться только в сети, где нет циклов. Это слишком сильное ограничение для практического использования.

Поэтому рассмотрим усовершенствованный лавинный алгоритм, в котором у каждого сообщения об изменении топологии будет порядковый номер. Каждый узел хранит порядковые номера таких сообщений, полученных от других узлов последний раз, и когда получает очередное сообщение от какого-либо узла, он сравнивает его номер с хранящимся номером сообщения от этого узла. Только если номер полученного сообщения больше, чем хранящийся, оно распространяется дальше, а его номер запоминается. В противном случае сообщение игнорируется. Данная схема действительно гарантирует, что при обычных условиях работы сети и при отсутствии переполнения поля номера каждое сообщение об изменении топологии будет передаваться каждым узлом своим соседям не более одного раза.

Отсутствие переполнения можно гарантировать, сделав поле порядкового номера достаточно большим. А вот исключительные ситуации, когда нарушаются условия 1-4 могут создать серьёзные проблемы. Рассмотрим несколько возможных сценариев.

Пусть часть сети, включающая по крайней мере один узел, выходит из строя, и все, попавшие в неё узлы теряют информацию о том, какой порядковый номер своих собственных сообщений они использовали последний раз. Если перед запуском установить его в ноль, то сообщения от этих узлов будут игнорироваться всей остальной сетью, до тех пор, пока их порядковые номера не превысят те, которые помнит остальная сеть.

Другая проблема может появиться тогда, когда либо в результате сбоя в памяти узла, либо в результате необнаруженных ошибок при передаче, порядковый номер сообщения изменяется на более высокий и распространяется по сети. Тогда последующие сообщения с корректными номерами будут игнорироваться, пока их порядковые номера не превысят ошибочный. Особенно тяжёлая ситуация возникнет, если ошибочный номер станет максимальным, который представим в выделенном ему поле.

Сформулируем правила работы лавинного алгоритма, свободные от некоторых из указанных выше недостатков. Для этого нам понадобится более строго определить порядок сообщений, исходящих от некоторого узла *i*. О двух сообщениях, A и B, порождённых узлом *i*, будем говорить, что A>B, если либо порядковый номер A больше порядкового номера B, либо, если у A и B совпадают порядковые номера, содержимое A больше содержимого B согласно некоторому лексикографическому правилу (например, если интерпретировать содержимое как двоичное число). Тогда для любых A и B

возможны только случаи, когда A>B, A=B, A<B, причём A=B только при полной идентичности A и B.

Пусть теперь каждый узел хранит не просто номера, а полностью сами сообщения, полученные последний раз от других узлов сети. Правила работы алгоритма для удобства разобьем на несколько групп:

Правило генерации новых сообщений: когда узел *i* фиксирует значимое изменение топологии сети, он немедленно рассылает всем соседям сообщение об этом с порядковым номером, на единицу больше, чем хранящийся у него номер предыдущего сообщения.

Правила обновления сообщений: пусть узел *k* получил обновляющее сообщение A, сгенерированное в узле *i*, а в его памяти хранится B - предыдущее сообщение от *i*. Тогда:

1. Если A>B, то
   1.1. Если $i \neq k$, то узел записывает себе в память A вместо B и рассылает копию A всем своим соседям, кроме того, от кого получил A.
   1.2. Если $i = k$, (т.е. узел вроде бы получил собственное сообщение, которое сам сгенерировал ранее) то узел *i* рассылает всем своим соседям обновляющее сообщение, содержащее текущие состояния всех смежных линий с порядковым номером, на единицу большим порядкового номера A. И это разосланное сообщение запоминает.
2. Если A=B, то узел игнорирует A.
3. Если A<B, то узел игнорирует A, но при этом посылает B обратно тому соседу, от которого получил A.

Правила восстановления узла после сбоев: **сбоем узла** считается ситуация, когда все смежные с ним линии связи вышли из строя и он некоторое время был изолирован от сети, а потом в некоторый момент времени одна или более линий вступили в строй.

1. Вступающий в строй узел посылает по всем вступившим в строй линиям связи обновляющее сообщение с порядковым номером 0.
2. Любой узел, получивший сообщение с порядковым номером 0, должен немедленно переслать по линии, откуда пришло сообщение номер 0, все сообщения о топологии, хранящиеся в его памяти (вместе с их порядковыми номерами, разумеется).

Нетрудно видеть, что правило обновления 1.2 в сочетании с правилом восстановления после сбоя 2 обеспечивают доведение текущих порядковых номеров и текущей топологии до тех частей сети, которые восстанавливаются после сбоя. Правила обновления 1.2 и 3 служат для коррекции ошибок в порядковых номерах сообщений.

Для того, чтобы понять, зачем надо сравнивать содержимое A и B при равенстве порядковых номеров, рассмотрим следующий пример: пусть в сети три узла и две линии связи, одна соединяет узлы 1 и 2, другая – узлы 2 и 3. Вначале обе линии исправны, и сообщения в памяти всех узлов содержат правильную информацию о статусе линий и имеют равные номера. Пусть вначале выходит из строя линия (2,3), потом (1,2), потом линия (2,3) восстанавливается. Узлы 2 и 3 начинают восстанавливаться после сбоя, посылают друг другу сообщения номер 0, и обмениваются после этого информацией о состоянии сети. При этом узел 2 считает, что линия 1-2 не исправна, а узел 3 – что она исправна, то есть, они посылают друг другу противоречивую информацию. Так как порядковые номера сообщений у обоих узлов одинаковы, то если бы сравнивались только они, каждый узел запомнил бы сообщение от другого узла, и противоречивая информация в разных узлах продолжала бы существовать. А в рассматриваемом алгоритме при равенстве номеров будет проведено сравнение содержимого, одно из двух сообщений будет больше, чем другое, и останется в обоих узлах именно оно. Оно может быть и неправильным – в данном случае, это информация о том, что линия 1-2 исправна, но смежный с этой линией узел 2 с помощью соответствующего механизма распознает эту ситуацию, и обновит информацию в сети, сделав её правильной. Главное, чтобы информация о состоянии сети во всех узлах была одинаковой.

К сожалению, для данного алгоритма остаётся нерешённой проблема переполнения поля данных в результате его искажения, а также проблема искажения самой топологической информации. В настоящее время неизвестны алгоритмы распространения топологической информации, использующие только порядковый номер сообщения и посылающие обновляющие сообщения только в момент значимых изменений состояния сети, свободные от всех перечисленных выше недостатков. Рассмотрим ниже один из существующих в настоящее время подходов.

Лавинный алгоритм с контролем времени и периодическим обновлением добавляет в обновляющее сообщение ещё одно служебное поле, кроме номера сообщения – **поле возраста**. В нём указывается время, в течение которого сообщение находится в сети. Каждый узел, куда приходит сообщение, запоминает время его прихода и перед отправкой добавляет к полю возраста время, проведённое сообщением в узле. Когда величина поля возраста превысит некоторый предел, сообщение считается устаревшим. К правилам работы алгоритма, приведены выше, добавляется ещё одна группа правил.

Правила обновления сообщений:

1. Устаревшие сообщения игнорируются независимо от их номера и содержания.
2. Неустаревшие сообщения обрабатываются согласно приведённым выше правилам.
3. Каждый узел периодически повторяет рассылку обновляющих сообщений, даже если состояние сети вокруг него не меняется.

Эта группа правил гарантирует, что на искажённую информацию не будут полагаться слишком долго и она за конечное время будет вытеснена из сети. Также периодические повторения облегчают корректное восстановление после сбоев.

Но это достигается ценой значительного увеличения передаваемой через сеть служебной информации из-за периодических обновлений. Поэтому к выбору интервала времени между периодическими обновлениями надо подходить очень серьёзно.

43. Примеры протоколов маршрутизации в Internet

Примеры реальных протоколов маршрутизации рассмотрим на примере сети Internet, но сначала необходимо отметить особенности этой сети, прямо влияющие на принципы маршрутизации в ней. Во-первых, Internet – это очень большая сеть. Количество узлов в ней точно неизвестно, но исчисляется сотнями миллионов и всё время растёт. При таком размере никакой, даже самый быстрый алгоритм не сможет за разумное время найти кратчайший путь. Но это и не требуется. Internet изначально создавался как “сеть сетей”, то есть, как объединение независимых друг от друга сетей самого разного размера. Независимость, в частности, означает, что внутри каждой сети маршрутизация производится независимо от маршрутизации в других частях Internet. Более того, соединение отдельной сети (называемой автономной системой) с остальной сетью происходит только через выделенные узлы (сетевые шлюзы), и если сообщение адресовано извне кому-то внутри сети, то оно доставляется сначала именно сетевому шлюзу, а он уже внутри сети пересылает сообщение получателю.

На самом верхнем уровне (между т.н. граничными шлюзами автономных систем) протокол маршрутизации один, в данный момент это BGP версии 4. А вот внутри каждой автономной системы и внутри каждой сети в автономной системе протоколы маршрутизации могут быть разными (выбираются владельцами автономной системы). Процедура выбора маршрута в BGP по сложности находится за пределами нашего курса. В качестве примеров применения сформулированных выше подходов и принципов маршрутизации мы рассмотрим два самых распространённых протокола, предназначенных как для работы внутри автономной системы, т.е. для поиска маршрутов между граничными шлюзами её сетей, так и внутри сети, т.е. для поиска маршрута между отдельными хостами.

Заметим, что сам процесс доставки сообщения по выбранному маршруту в Internet един на всех уровнях, и осуществляется протоколом IP (Internet Protocol).

RIP (Routing Internet Protocol) – это полностью распределённый протокол маршрутизации, работающий на основе асинхронного распределённого алгоритма Беллмана-Форда. Длины всех исправных линий связи в этом протоколе считаются равными 1, независимо от их нагрузки и текущей задержки передачи. Это означает, что длина пути принимается равной количеству линий связи на этом пути (т.н. hop count – счёт прыжков). При этом, чтобы избежать слишком долгого реагирования на резкое изменение топологии, длина пути может принимать значения от 1 до 15, а длина 16 считается бесконечностью. Поэтому данный протокол не может корректно работать в больших сетях, где на пути может быть более 15 линий связи.

Кроме того, чтобы избежать "зацикливаний", часто возникающих в распределённых алгоритмах при распознавании резких изменений топологии, в протоколе применяется особый приём, называемый split horizon (расщепление горизонта) – он заключается в том, что если узел сети *i* в результате итераций алгоритма Беллмана-Форда обнаруживает, что кратчайший путь от него к узлу *k* лежит через соседа – узел *j*, то в качестве оценки длины пути до *k*, он этому соседу *j* посылает не то, что у него получилось в результате итераций (и что он рассылает всем остальным соседям), а бесконечность. Это позволяет избежать ситуаций, когда у узла *j* резко изменяется длина пути до *k*, и он начинает считать, что теперь кратчайший путь в *k* лежит через *i*, а *i* по прежнему считает, что путь идёт через *j*.

Итерации алгоритма Беллмана-Форда и рассылку результатов соседям каждый узел, согласно протоколу, должен делать не реже, чем раз в 30 сек, даже если изменений в статусе смежных линий связи нет. Если статус какой-либо из смежных с ним линий изменится с рабочей на нерабочую или наоборот, то итерация и рассылка должны быть сделаны как можно скорее, но не чаще, чем некоторое случайное время, равномерно распределённое между 1 и 5 сек. Такое ограничение на максимальную частоту итераций сделано для того, чтобы устранить явление т.н. "самосинхронизации", которое наблюдалось в сетях с протоколом RIP, когда этого ограничения не было – через некоторое время после начала работы протокола все узлы начинали производить итерации и рассылки всё более и более одновременно. Это вызывало периодическую загрузку всех каналов служебной информацией, и, соответственно, падением в этот момент скорости передачи полезной информации. Точная причина этого явления осталась неизвестной, но после введения рандомизированной границы для максимальной частоты итераций оно исчезло.

Если по какой-либо линии дольше чем 180 сек в узел не приходят служебные сообщения протокола RIP (а, как указано выше, должны приходить не реже, чем раз в 30 сек.), то узел на другом конце линии считается вышедшим из строя, и линия связи считается нерабочей.

RIP – это самый первый протокол маршрутизации в Internet, что видно из его названия. И многие его особенности связаны с изначальными особенностями Internet. В частности, метрика (т.е. измерение) длины пути в количестве линий связи, которые в нём задействованы, связано с тем, что главным критерием для выбора маршрута должна была быть его надёжность. Internet (точнее, её предок ARPANET) первоначально создавался как сеть правительственной связи США на случай тотальной ядерной войны с СССР. С точки зрения сети передачи данных, последствия обмена массированными ядерными ударами проявляются как внезапный и, как правило, необратимый выход из строя узлов и целых участков сети. ARPANET должен был до последней возможности обеспечивать передачу сообщений в таких условиях. Потом он превратился в Internet, постепенно стал сетью общего пользования, а протокол маршрутизации остался прежним.

В настоящее время вместо первоначальной версии RIP-1, используется так называемый RIP-2, способный, в частности, работать с групповой адресацией (а не только с маршрутами от одного отправителя к одному получателю). Кроме того, в него введены элементы защиты (шифрования) служебных сообщений.

Преимуществом RIP являются его простота и надёжность, недостатками – неспособность работать в больших сетях, медленная реакция на быстрые изменения состояния сети и быстрый рост объёмов служебной информации при росте размеров сети.

OSPF (Open Shortest Path First) – локальный (т.е. нераспределённый) протокол маршрутизации, где в каждом узле имеется вся информация о состоянии сети, и каждый узел рассчитывает полный маршрут до узла-получателя сообщения.

В качестве алгоритма распространения информации о состоянии сети используется лавинный алгоритм с нумерацией сообщений, контролем времени и периодическим обновлением, близкий к рассмотренному нами ранее. В него введён ряд усовершенствований, сокращающий объём передаваемой служебной информации. В частности, возможность запрашивать и получать информацию о статусе определённой линии, а не всей базы данных соседнего узла. Рассылка сообщений о состоянии линий связи производится узлом при изменении этого состояния немедленно, а при отсутствии изменений не реже чем раз в 30 минут.

Кратчайшие маршруты могут рассчитываться любым корректным алгоритмом поиска кратчайшего пути, но рекомендуются к использованию алгоритмы Беллмана-Форда, Дийкстра и Флойда-Уоршелла (на практике чаще всего используется алгоритм Дийкстра). Метрики при этом могут использоваться разные, а именно:

- задержка пакета на линии связи;
- величина, обратная скорости линии связи;
- стоимость передачи по линии;
- величина, обратная надёжности линии связи, то есть вероятности её выхода из строя.

Понятно, что использование определённой метрики ведёт к выбору маршрута, минимизирующего соответствующую характеристику. Поддерживать все метрики, или только некоторые, решает администратор сети. При этом надо учитывать, что поддержание нескольких метрик – это весьма ресурсозатратное дело, так как для каждой нужно рассчитывать и хранить свою маршрутную таблицу, собирать и рассылать соответствующую информацию по сети. В каждом IP-пакете есть поле предпочтительной метрики для маршрутизации, и если в данной сети этот тип поддерживается, то используется соответствующая маршрутная таблица. Если же нет – используется основная метрика (тоже определяется администратором).

OSPF также поддерживает групповую адресацию и шифрование служебной информации. Достоинства этого протокола – большая оптимальность, возможность работать в больших сетях и меньшие накладные расходы на пересылку информации в больших сетях. Недостатки – сложность (описание стандарта RIP занимает около 100 стр., описание стандарта OSPF – более 900 стр.), и потому необходимость в грамотном администрировании, повышенные требования к аппаратуре маршрутизаторов в смысле быстродействия процессоров и объема памяти.

По-видимому, эти два протокола ещё долгое время будут сосуществовать в Internet, так как они подходят для использования в сетях с разными условиями, и этим дополняют друг друга. Другие протоколы маршрутизации весьма похожи либо на RIP, либо на OSPF, но их рассмотрение находится за пределами нашего курса.

44. Задачи транспортного уровня, управление потоком

После того, как на сетевом уровне (третий уровень модели OSI) из надёжных линий связи организуется сеть, она, с точки зрения теории связи, утрачивает свойство надёжности, которое после второго (канального) уровня было присуще логическим линиям связи. Это кажущийся парадокс, связанный на самом деле с тем, что у сети появляются дополнительные функциональные возможности, которыми отдельная линия связи не обладала.

Дело в том, что при рассмотрении работы одной линии связи мы не рассматривали ситуацию, когда из строя выходит либо сама линия, либо одна из сторон, участвующая в передаче данных. Не рассматривали потому, что в таких ситуациях система связи становится полностью неработоспособной, а ликвидация таких ситуаций в область рассмотрения теории связи не входит.

А вот сеть передачи данных вполне может сохранять работоспособность при выходе из строя части своих элементов. Их починка по-прежнему не входит в область теории связи, а вот организация работы сети, сохраняющая свойство надёжной передачи данных в таких ситуациях, вполне может быть (и является) предметом нашего теоретического рассмотрения.

С инженерной точки зрения для этого в модели OSI служит транспортный (четвёртый) уровень. Его главная задача – восстановление свойства надёжности передачи данных, когда они передаются от отправителя к получателю через сеть. Но для её решения, кроме борьбы с потерей и искажением информации при передаче, как это было на канальном уровне, приходится решать и другие, совершенно новые проблемы, которых раньше не было.

Мы рассмотрим некоторые подходы к одной из таких проблем, а именно к борьбе с перегрузками сети, которая осуществляется методами так называемого **управления потоками**.

Перегрузкой сети называется ситуация, когда поступающий в сеть трафик превышает возможности сети по его передаче, даже при самом оптимальном поведении. При этом, если не принять никаких мер по ограничению поступающего трафика, то размеры очередей в узлах, смежных с наиболее нагруженными линиями начнут неограниченно расти, и превысят размеры памяти, отведённой для хранения этих очередей. После этого поступающие в узел пакеты, которым не нашлось места в буфере узла, начнут сбрасываться. А механизмы второго уровня будут фиксировать потерю пакета, и передавать их повторно, ещё увеличивая и без того большую нагрузку. Сетевой уровень в попытке перенаправить трафик из перегруженных участков в менее нагруженные может довольно быстро распространить перегрузку по всей сети, и в результате скорость передачи по сети информации пользователей упадет практически до нуля.

В чём-то эта ситуация похожа на автомобильную пробку, когда слишком большое количество автомобилей приводит к полной остановке их движения по автостраде и прилегающим дорогам, но есть и серьёзные отличия: в пробке автомобили стоят, так что со стороны сразу видно, что система не работает. В случае с перегрузкой сети всё совершенно по-другому – сеть в это время работает, так сказать, на полную мощность, все линии связи загружены, пакеты передаются и принимаются с максимально возможной скоростью, а вот передачи полезной информации практически не происходит.

Такие эффекты возникли именно после широкого распространения современных сетей передачи данных. Первая перегрузка в Internet наблюдалась в 1986 году. До этого иногда происходили явления, которые можно считать родственными перегрузке сети передачи данных, но всё-таки многих аспектов современных перегрузок там не наблюдалось. Так, самой первой глобальной перегрузкой телефонной сети в масштабах всей страны считаются события, происшедшие в США непосредственно после убийства

президента Кеннеди. Так как в те времена снятие трубки на телефонном аппарате вызывало некоторое увеличение силы тока, проходящего через схемы коммутатора на телефонной станции, то, когда сразу слишком много людей попытались позвонить (после объявления по средствам массовой информации о происшедшем), не рассчитанные на такое количество одновременных вызовов коммутаторы просто вышли из строя, и большая часть телефонной сети всех США оказалась на много часов неработоспособной.

Заметим также, что попав в состояние перегрузки, сеть передачи данных может находиться в ней весьма долго, если не принимать специальных мер. Но лучше перегрузок не допускать, и именно это является одной из функций транспортного уровня.

Основными задачами при управлении потоком являются:

1) установление разумного компромисса между ограничением трафика, входящего в сеть и средней задержкой сообщения;
2) соблюдение разумно справедливости по отношению ко всем пользователям, когда часть трафика не допускается в сеть.

Разберём эти задачи более подробно. Малая средняя задержка сообщения – весьма желательная характеристика сети с точки зрения пользователя. Однако, механизм управления потоком далеко не всегда уменьшает задержку *для пользователя*. Это означает, что ограничивая входной трафик в сеть, механизм управления потоком заставляет сообщения пользователя ожидать *вне сети*, у пользователя, а не внутри сети, в очередях на передачу в сетевых узлах. Это приводит к тому, что с точки зрения пользователя после включения управления потоком сообщения будут передаваться медленнее, чем до этого. Но это позволит избежать перегрузки, когда падение эффективности работы сети станет катастрофическим.

Основная причина, по которой важно сохранять задержку малой внутри сети, даже ценой её увеличения вне её – это стремление избежать повторных передач во-первых, из-за сброса пакетов в переполненных очередях, и во-вторых, из-за того, что подтверждения возвращаются с задержкой, большей тайм-аута протокола повторной передачи (тоже из-за больших очередей в узлах). Эти повторные передачи впустую тратят ресурс сети и способствуют распространению перегрузок.

Что касается соблюдения справедливости при ограничении доступа части трафика в сеть, то это нетривиальная задача, так как максимизация пропускной способности сети часто оказывается несовместимой с соображениями справедливости. В качестве примера рассмотрим следующую ситуацию: пусть один из пользователей передаёт трафик по маршруту, проходящему через, скажем, n линий связи, а остальные n пользователей передают трафик только между смежными узлами сети, т.е. так, что маршрут у каждого проходит только по одной из линий связи, лежащей на маршруте первого пользователя. То есть, в каждой из n линий связи будет трафик первого пользователя и ещё одного другого пользователя. Если совместно они перегрузят линию, то должен включится механизм управления потоком. С точки зрения максимизации полной пропускной способности трафик первого пользователя надо не допускать в сеть весь, или почти весь, так как это уменьшит нагрузку сразу на n линий связи. Однако, это не справедливо по отношению к нему, если он имеет такой же статус при пользовании сетью, как и остальные. А если соблюдать полную справедливость, то полная пропускная способность сети будет далека от максимальной. Тут необходим разумный компромисс.

Оконное управление потоком. Главная идея этого самого распространённого метода управления потоком в сети передачи данных – это установление верхней границы на число единиц данных (для простоты предположим пакетов), которые были уже отосланы некоторым источником некоторому получателю, но ещё не попали к нему, то есть, находятся в сети. Эта верхняя граница – целое положительное число, называется **размером окна**, или просто **окном**.

Приёмник уведомляет передатчик о том, что к нему попала очередная единица данных отправлением специального сообщения, которое будем называть **разрешением**. После его получения, передатчик может отослать в сеть очередную единицу данных, предназначенную для этого приёмника. Разрешения могут иметь вид специальных сообщений, а могут прикрепляться к другим сообщениям в виде служебных полей.

При этом, если по каким либо причинам замедляется возвращение разрешений к передатчику, то и интенсивность потока от передатчика в сеть уменьшается. Следовательно, если в процессе движения трафика по сети он где-то попадает в состояние, близкое к перегрузке, то задержка его прохождения по сети растёт, время возвращения разрешений ко всем передатчикам, которые шлют трафик через близкий к состоянию перегрузки район сети тоже растёт, и скорость их засылки новых сообщений в сеть падает. Кроме того, приёмники при необходимости могут намеренно задерживать отправку разрешений для ограничения скорости передатчиков.

Очевидно сходство между оконным методом управления потоком и алгоритмом повторной передачи. Однако между ними есть принципиальная разница, обусловленная совершенно разными целями двух этих механизмов: при повторной передаче, если передатчик не получает подтверждения, он передаёт пакет снова, а при оконном управлении потоком передатчик будет молчать и ждать пока придёт разрешение. Поэтому нетрудно видеть, что без механизма повторной передачи механизм управления потоком работать в реальных условиях не будет, но совместить работу двух этих механизмов так, чтобы они не мешали друг другу и обеспечивали надёжную передачу с избеганием перегрузок, задача непростая.

Оконное управление от конца до конца. Рассмотрим простой метод оценки характеристик оконного управления потоком. Пусть размер окна для некоторого передатчика равен W пакетов. У каждого пакета есть, разумеется, порядковый номер, и в разрешении содержится номер пакета, который приёмник ожидает. Так что получение разрешения с номером k означает, во-первых, что все пакеты с номерами меньше k успешно приняты, и во-вторых, что можно передать в сеть пакеты с номерами от k до $k+W-1$ включительно. Пусть время передачи одного пакета равно x.

Для простоты мы будем считать, что у передатчика всегда есть пакет для передачи. Тогда, после начала работы он может, не ожидая разрешения, передать W пакетов, что займёт xW единиц времени. Если за это время ему придёт хотя бы одно разрешение, и дальше примерно через x единиц времени будут приходить следующие, то он сможет работать, не останавливаясь. Обозначив через d время, проходящее от начала передачи пакета до получения разрешения от приёмника, которое он отправил, приняв этот пакет, можем заключить, что если $d \le xW$ на протяжении некоторого времени, то передатчик может работать без остановки, и его скорость будет $1/x$ пакетов в единицу времени.

Если же $d > xW$, то передатчик после отправки W пакетов остановится и будет ждать разрешения, которое приёмник пошлёт ему после получения первого пакета. Если d будет для всех пакетов одинаково, то дождавшись первого разрешения, передатчик будет получать ещё $W-1$ разрешений через x единиц времени, и передаст, не останавливаясь, вторую порцию из W пакетов. Потом опять остановится, ожидая разрешение, которое ему пошлют на первый пакет второй группы, и т.д. То есть, в этом случае он будет передавать в среднем W пакетов за d единиц времени, и средняя скорость передачи будет W/d.

Суммируя всё вышесказанное, можно выразить среднюю скорость передатчика, как

$$r = \min\left(\frac{1}{x}, \frac{W}{d}\right). \qquad (177)$$

Из этой формулы видно, что управление потоком включается, когда $d=xW$, и при дальнейшем росте d уменьшает среднюю скорость передатчика обратно пропорционально d.

К достоинствам оконного управления потоком относится простота этого механизма, сравнительно быстрая реакция на увеличение нагрузки в сети (время реакции порядка xW), и малый объём служебных сообщений.

Однако, имеются и недостатки. Самый серьёзный из них – это то, что в том виде, в котором мы его рассмотрели, оконное управление потоком не во всех ситуациях способно предотвратить перегрузку. Коротко говоря, оконное управление с постоянным размером окна не спасёт, если за небольшое время в сети начнёт работу так много передатчиков, что W пакетов от каждого создадут перегрузку. В этом случае необходимо использовать оконное управление с переменным размером окна, но его анализ находится за пределами нашего курса, мы лишь приведём практический пример его использования.

45. Управление потоком в протоколе TCP

Главная задача протокола TCP – обеспечение надёжной передачи данных через сеть. Это, в частности, означает, что он должен организовывать повторную передачу, так как из-за возможности выхода из строя узлов и потери всех сообщений, которые в этот момент в этих узлах находятся, механизма повторной передачи на канальном уровне недостаточно. Кроме того, поскольку в сети Internet должно обеспечиваться взаимодействие между очень разными по своим возможностям сетевыми устройствами, протокол должен поддерживать согласование различных параметров (в частности, скорости передачи) а также, как мы упоминали выше, протокол должен бороться с перегрузками, используя для этого оконные методы. Есть ещё ряд задач, которые протоколу TCP приходится попутно решать, но мы не будем их рассматривать, а сосредоточимся только на указанных выше.

Протокол TCP – это протокол с установлением соединения. Это означает, что перед тем, как начать передачу от отправителя к получателю, между ними должно быть организовано “виртуальное соединение”, существующее всё время, пока передаются данные. При установлении соединения стороны обмениваются между собой несколькими сообщениями (т.н. “трёхстороннее рукопожатие”), в процессе которого согласуют необходимые параметры. Нас в этом процессе прежде всего интересует то, что одновременно стороны производят измерение времени прохождения сообщений от отправителя до получателя и обратно – на основе этих измерений они устанавливают, в частности, значение тайм-аутов повторной передачи, которые устанавливаются в несколько раз большими, чем измеренное время, считающееся временем нормального отклика.

Для совместной организации повторной передачи и оконного управления потоком, приёмник и передатчик определяют т.н. “окно потока”, а передатчик ещё и т.н. “окно перегрузки”.

Окно потока (или просто **окно**) – это количество байт, которое получатель готов принять. В протоколе TCP информация пользователя, передаваемая по установленному соединению, представляется просто как последовательность байт, называемая **потоком**. Поэтому подтверждение успешного приёма происходит в терминах этого потока, а именно: в подтверждении указывается номер в потоке ожидаемого байта, что означает безошибочный приём всех предыдущих байт, и ещё в специальном поле указывается размер окна в байтах, то есть ту порцию информации, которую получатель готов принять. Этим одновременно управляется и повторная передача, и скорость передатчика, чтобы он не “затопил” более медленный приёмник данными. Так, приёмник может отправить в подтверждении нулевой размер окна, что означает для передатчика указание временно прекратить передачу (как RNR в HDLC).

Окно перегрузки – это количество т.н. сегментов потока, то есть, информационных пакетов, которые передатчику разрешено передать в сеть до получения подтверждения о получении самого первого из них. В самом начале работы размер окна

перегрузки равен 1 сегменту. Если подтверждение на очередной сегмент приходит за время, меньшее тайм-аута, то размер окна увеличивается на два сегмента, что приводит к его очень быстрому росту. По историческим причинам этот механизм называется “медленный старт” и служит для того, чтобы передатчик в случае более-менее благоприятной ситуации в сети, мог быстро установить достаточно большую скорость работы. Но такой рост окна происходит только до некоторого порога (как правило, 8 сегментов), после чего он сменяется линейным ростом – каждый вовремя подтверждённый сегмент увеличивает размер окна перегрузки на 1 сегмент. А если подтверждение не пришло вовремя (или вообще не пришло), это считается показателем приближения перегрузки, и размер окна уменьшается в два раза – то есть, это оконное управление с переменным размером окна для лучшей борьбы с перегрузками.

В результате, передатчик, когда готовит к отправке приёмнику очередную порцию информации, руководствуется данными обоих окон – сначала он определяет, сколько информации получатель готов принять согласно размеру окна потока. А потом, из этой части информации он формирует не более чем разрешённое окном перегрузки число сегментов (если данных много, то старается, как правило, использовать сегменты максимального размера, обычно 1500 байт – максимальный размер пакета в Ethernet). Что не уместилось – остаётся ждать прихода очередного подтверждения.

Это сильно упрощённое описание работы механизма управления потоком в TCP, но более детальное рассмотрение находится за пределами нашего курса.

Список литературы

1. К.К. Васильев, В.А. Глушков, А.В. Дормидонтов, А.Г. Нестеренко. Теория электрической связи: учебное пособие. - Ульяновск: УлГТУ, 2008.
2. А.Г. Зюко, Д.Д. Кловский, В.И. Коржик, М.В. Назаров. Теория электрической связи: учебник для вузов. – М: Радио и связь, 1999.
3. В.Н. Васюков. Общая теория связи: Учебник– Новосибирск: Изд-во НГТУ, 2013.
4. В.А. Григорьев и др. Теория электрической связи. Конспект лекций: –СПб: НИУ ИТМО, 2012.
5. Д. Возенкрафт, И. Джекобс. Теоретические основы техники связи. – М: Мир, 1969.
6. Р. Блейхут. Теория и практика кодов, контролирующих ошибки.– М: Мир, 1986.
7. А.А. Харкевич. Борьба с помехами. -М: Наука, 1965.
8. У. Питерсен, Э. Уэлдон. Коды, исправляющие ошибки. – М: Мир, 1976.
9. Д. Бертсекас, Р. Галлагер. Сети передачи данных. – М: Мир, 1989.
10. И.С. Гоноровский. Радиотехнические цепи и сигналы. – М: Дрофа, 2006.
11. Дж. Прокис. Цифровая связь. – М: Радио и связь, 2000.
12. Б. Скляр. Цифровая связь. Теоретические основы и практическое применение. – М: Вильямс, 2003.
13. В.Г. Олифер, Н.А. Олифер. Компьютерные сети. Принципы, технологии, протоколы. – М: Питер, 2010.