**10.12.2020 ИС2К Операционные системы**

**«Основы операционных систем» К.А. Коньков** <https://yadi.sk/i/SojomTcdoQ2v_g>

 **Тема:** Лекция 14: **Сети и сетевые операционные системы**

 (Учебник «Основы операционных систем» К.А.Коньков стр. 64-71)

Заданий не будет –завтра экзамен. Готовтесь.

**Для тех у кого проблемы с Яндекс диском сегодняшнюю тему**

 **(**Лекция 14:

**Сети и сетевые операционные системы**)

**выкладываю ниже**

**Лекция: Сети и сетевые операционные системы**

В лекции рассматриваются особенности взаимодействия процессов, выполняющихся на разных операци-
онных системах, и вытекающие из этих особенностей функции сетевых частей операционных систем.

До сих пор в лекциях данного курса мы ограничивались рамками классических операционных систем, т.
е. операционных систем, функционирующих на автономных однопроцессорных вычислительных маши-
нах, которые к середине 80-х годов прошлого века составляли основу мирового парка вычислительной
техники. Подчиняясь критериям повышения эффективности и удобства использования, вычислительные
системы с этого времени, о чем мы уже упоминали в самой первой лекции, начинают бурно развиваться в
двух направлениях: создание многопроцессорных компьютеров и объединение автономных систем в вы-
числительные сети.

**Основы операционных систем**

**145**

Появление многопроцессорных компьютеров не оказывает существенного влияния на работу операцион-
ных систем. В многопроцессорной вычислительной системе изменяется содержание состояния исполне-
ние. В этом состоянии может находиться не один процесс, а несколько - по числу процессоров. Соответ-
ственно изменяются и алгоритмы планирования. Наличие нескольких исполняющихся процессов требует
более аккуратной реализации взаимоисключений при работе ядра. Но все эти изменения не являются из-
менениями идеологическими, не носят принципиального характера. Принципиальные изменения в мно-
гопроцессорных вычислительных комплексах затрагивают алгоритмический уровень, требуя разработки
алгоритмов распараллеливания решения задач. Поскольку с точки зрения нашего курса многопроцессор-
ные системы не внесли в развитие операционных систем что-либо принципиально новое, мы их рассмат-
ривать далее не будем.

По-другому обстоит дело с вычислительными сетями.

**Для чего компьютеры объединяют в сети**

Для чего вообще потребовалось объединять компьютеры в сети? Что привело к появлению сетей?

• Одной из главных причин стала необходимость совместного использования ресурсов (как физиче-
ских, так и информационных). Если в организации имеется несколько компьютеров и эпизодиче-
ски возникает потребность в печати какого-нибудь текста, то не имеет смысла покупать принтер
для каждого компьютера. Гораздо выгоднее иметь один сетевой принтер для всех вычислитель-
ных машин. Аналогичная ситуация может возникать и с файлами данных. Зачем держать одина-
ковые файлы данных на всех компьютерах, поддерживая их когерентность, если можно хранить
файл на одной машине, обеспечив к нему сетевой доступ со всех остальных?

• Второй причиной следует считать возможность ускорения вычислений. Здесь сетевые объедине-
ния машин успешно конкурируют с многопроцессорными вычислительными комплексами. Мно-
гопроцессорные системы, не затрагивая по существу строение операционных систем, требуют
достаточно серьезных изменений на уровне hardware, что очень сильно повышает их стоимость.

Во многих случаях можно добиться требуемой скорости вычислений параллельного алгоритма,
используя не несколько процессоров внутри одного вычислительного комплекса, а несколько от-
дельных компьютеров, объединенных в сеть. Такие сетевые вычислительные кластеры часто
имеют преимущество перед многопроцессорными комплексами в соотношении эффектив-
ность/ стоимость.

• Следующая причина связана с повышением надежности работы вычислительной техники. В сис-
темах, где отказ может вызвать катастрофические последствия (атомная энергетика, космонавти-
ка, авиация и т. д.), несколько вычислительных комплексов устанавливаются в связи, дублируя
друг друга. При выходе из строя основного комплекса его работу немедленно продолжает дубли-
рующий.

• Наконец, последней по времени появления причиной (но для многих основной по важности) стала
возможность применения вычислительных сетей для общения пользователей. Электронные пись-
ма практически заменили письма обычные, а использование вычислительной техники для органи-
зации электронных или телефонных разговоров уверенно вытесняет обычную телефонную связь.

**Сетевые и распределенные операционные системы**

В первой лекции мы говорили, что существует два основных подхода к организации операционных сис-
тем для вычислительных комплексов, связанных в сеть, - это сетевые и распределенные операционные
системы. Необходимо отметить, что терминология в этой области еще не устоялась. В одних работах все
операционные системы, обеспечивающие функционирование компьютеров в сети, называются распреде-
ленными, а в других, наоборот, сетевыми. Мы придерживаемся той точки зрения, что сетевые и распре-
деленные системы являются принципиально различными.

В сетевых операционных системах для того, чтобы задействовать ресурсы другого сетевого компьютера,
пользователи должны знать о его наличии и уметь это сделать. Каждая машина в сети работает под
управлением своей локальной операционной системы, отличающейся от операционной системы авто-
номного компьютера наличием дополнительных сетевых средств (программной поддержкой для сетевых

**Основы операционных систем**

**146**

интерфейсных устройств и доступа к удаленным ресурсам), но эти дополнения существенно не меняют
структуру операционной системы.

Распределенная система, напротив, внешне выглядит как обычная автономная система. Пользователь не
знает и не должен знать, где его файлы хранятся, на локальной или удаленной машине, и где его про-
граммы выполняются. Он может вообще не знать, подключен ли его компьютер к сети. Внутреннее
строение распределенной операционной системы имеет существенные отличия от автономных систем.

Изучение строения распределенных операционных систем не входит в задачи нашего курса. Этому во-
просу посвящены другие учебные курсы - Advanced operating systems, как называют их в англоязычных
странах, или "Современные операционные системы", как принято называть их в России.

В этой лекции мы затронем вопросы, связанные с сетевыми операционными системами, а именно - какие
изменения необходимо внести в классическую операционную систему для объединения компьютеров в
сеть.

**Взаимодействие удаленных процессов как основа работы вычислительных сетей**

Все перечисленные выше цели объединения компьютеров в вычислительные сети не могут быть достиг-
нуты без организации взаимодействия процессов на различных вычислительных системах. Будь то дос-
туп к разделяемым ресурсам или общение пользователей через сеть - в основе всего этого лежит взаимо-
действие удаленных процессов, т. е. процессов, которые находятся под управлением физически разных
операционных систем. Поэтому мы в своей работе сосредоточимся именно на вопросах кооперации та-
ких процессов, в первую очередь выделив ее отличия от кооперации процессов в одной автономной вы-
числительной системе (кооперации локальных процессов), о которой мы говорили в лекциях 4, 5 и 6.

1. Изучая взаимодействие локальных процессов, мы разделили средства обмена информацией по
объему передаваемых между ними данных и возможности влияния на поведение другого процесса
на три категории: сигнальные, канальные и разделяемая память. На самом деле во всей этой сис-
тематизации присутствовала некоторая доля лукавства. Мы фактически классифицировали сред-
ства связи по виду интерфейса обращения к ним, в то время как реальной физической основой для
всех средств связи в том или ином виде являлось разделение памяти. Семафоры представляют со-
бой просто целочисленные переменные, лежащие в разделяемой памяти, к которым посредством
системных вызовов, определяющих состав и содержание допустимых операций над ними, могут
обращаться различные процессы. Очереди сообщений и pip'bi базируются на буферах ядра опера-
ционной системы, которые опять-таки с помощью системных вызовов доступны различным про-
цессам. Иного способа реально передать информацию от процесса к процессу в автономной вы-
числительной системе просто не существует. Взаимодействие удаленных процессов принципи-
ально отличается от ранее рассмотренных случаев. Общей памяти у различных компьютеров фи-
зически нет. Удаленные процессы могут обмениваться информацией, только передавая друг другу
пакеты данных определенного формата (в виде последовательностей электрических или электро-
магнитных сигналов, включая световые) через некоторый физический канал связи или несколько
таких каналов, соединяющих компьютеры. Поэтому в основе всех средств взаимодействия уда-
ленных процессов лежит передача структурированных пакетов информации или сообщений.

2. При взаимодействии локальных процессов и процесс-отправитель информации, и процесс-
получатель функционируют под управлением одной и той же операционной системы. Эта же опе-
рационная система поддерживает и функционирование промежуточных накопителей данных при
использовании непрямой адресации. Для организации взаимодействия процессы пользуются од-
ними и теми же системными вызовами, присущими данной операционной системе, с одинаковыми
интерфейсами. Более того, в автономной операционной системе передача информации от одного
процесса к другому, независимо от используемого способа адресации, как правило (за исключени-
ем микроядерных операционных систем), происходит напрямую - без участия других процессов-
посредников. Но даже и при наличии процессов-посредников все участники передачи информа-
ции находятся под управлением одной и той же операционной системы. При организации сети,
конечно, можно обеспечить прямую связь между всеми вычислительными комплексами, соединив
каждый из них со всеми остальными посредством прямых физических линий связи или подклю-
чив все комплексы к общей шине (по примеру шин данных и адреса в компьютере). Однако такая

**Основы операционных систем**

**147**

сетевая топология не всегда возможна по ряду физических и финансовых причин. Поэтому во
многих случаях информация между удаленными процессами в сети передается не напрямую, а че-
рез ряд процессов-посредников, "обитающих" на вычислительных комплексах, не являющихся
компьютерами отправителя и получателя и работающих под управлением собственных операци-
онных систем. Однако и при отсутствии процессов-посредников удаленные процесс-отправитель
и процесс-получатель функционируют под управлением различных операционных систем, часто
имеющих принципиально разное строение.

3. Вопросы надежности средств связи и способы ее реализации, рассмотренные нами в лекции 4, но-
сили для случая локальных процессов скорее теоретический характер. Мы выяснили, что физиче-
ской основой " общения" процессов на автономной вычислительной машине является разделяемая
память. Поэтому для локальных процессов надежность передачи информации определяется на-
дежностью ее передачи по шине данных и хранения в памяти машины, а также корректностью ра-
боты операционной системы. Для хороших вычислительных комплексов и операционных систем
мы могли забыть про возможную ненадежность средств связи. Для удаленных процессов вопросы,
связанные с надежностью передачи данных, становятся куда более значимыми. Протяженные се-
тевые линии связи подвержены разнообразным физическим воздействиям, приводящим к искаже-
нию передаваемых по ним физических сигналов (помехи в эфире) или к полному отказу линий
(мыши съели кабель). Даже при отсутствии внешних помех передаваемый сигнал затухает по мере
удаления от точки отправления, приближаясь по интенсивности к внутренним шумам линий свя-
зи. Промежуточные вычислительные комплексы сети, участвующие в доставке информации, не
застрахованы от повреждений или внезапной перезагрузки операционной системы. Поэтому вы-
числительные сети должны организовываться исходя из предпосылок ненадежности доставки фи-
зических пакетов информации.

4. При организации взаимодействия локальных процессов каждый процесс (в случае прямой адреса-
ции) и каждый промежуточный объект для накопления данных (в случае непрямой адресации)
должны были иметь уникальные идентификаторы - адреса - в рамках одной операционной систе-
мы. При организации взаимодействия удаленных процессов участники этого взаимодействия
должны иметь уникальные адреса уже в рамках всей сети.

5. Физическая линия связи, соединяющая несколько вычислительных комплексов, является разде-
ляемым ресурсом для всех процессов комплексов, которые хотят ее использовать. Если два про-
цесса попытаются одновременно передать пакеты информации по одной и той же линии, то в ре-
зультате интерференции физических сигналов, представляющих эти пакеты, произойдет взаимное
искажение передаваемых данных. Для того чтобы избежать возникновения такой ситуации (race
condition!) и обеспечить эффективную совместную работу вычислительных систем, должны вы-
полняться условия взаимоисключения, прогресса и ограниченного ожидания при использовании
общей линии связи, но уже не на уровне отдельных процессов операционных систем, а на уровне
различных вычислительных комплексов в целом.

Давайте теперь, абстрагировавшись от физического уровня организации связи и не обращая внимания на
то, какие именно физические средства - оптическое волокно, коаксиальный кабель, спутниковая связь и
т. д. - лежат в основе объединения компьютеров в сеть, обсудим влияние перечисленных отличий на ло-
гические принципы организации взаимодействия удаленных процессов.

**Основные вопросы логической организации передачи информации между удаленными
процессами**

К числу наиболее фундаментальных вопросов, связанных с логической организацией взаимодействия
удаленных процессов, можно отнести следующие:

1. Как нужно соединять между собой различные вычислительные системы физическими линиями
связи для организации взаимодействия удаленных процессов? Какими критериями при этом сле-
дует пользоваться?

2. Как избежать возникновения race condition при передаче информации различными вычислитель-
ными системами после их подключения к общей линии связи? Какие алгоритмы могут при этом
применяться?

3. Какие виды интерфейсов могут быть предоставлены пользователю операционными системами для
передачи информации по сети? Какие существуют модели взаимодействия удаленных процессов?

**Основы операционных систем**

**148**

Как процессы, работающие под управлением различных по своему строению операционных сис-
тем, могут общаться друг с другом?

4. Какие существуют подходы к организации адресации удаленных процессов? Насколько они эф-
фективны?

5. Как организуется доставка информации от компьютера-отправителя к компьютеру-получателю
через компьютеры-посредники? Как выбирается маршрут для передачи данных в случае разветв-
ленной сетевой структуры, когда существует не один вариант следования пакетов данных через
компьютеры-посредники?

Разумеется, степень важности этих вопросов во многом зависит от того, с какой точки зрения мы рас-
сматриваем взаимодействие удаленных процессов. Системного программиста, в первую очередь, интере-
суют интерфейсы, предоставляемые операционными системами. Сетевого администратора больше будут
занимать вопросы адресации процессов и выбора маршрута доставки данных. Проектировщика сетей в
организации - способы соединения компьютеров и обеспечения корректного использования разделяемых
сетевых ресурсов. Мы изучаем особенности строения и функционирования частей операционных систем,
ответственных за взаимодействие удаленных процессов, а поэтому рассматриваемый перечень вопросов
существенно сокращается.

Выбор способа соединения участников сетевого взаимодействия физическими линиями связи (количест-
во и тип прокладываемых коммуникаций, какие именно устройства и как они будут соединять,т. е. топо-
логия сети) определяется проектировщиками сетей исходя из имеющихся средств, требуемой производи-
тельности и надежности взаимодействия. Все это не имеет отношения к функционированию операцион-
ных систем, является внешним по отношению к ним и в нашем курсе рассматриваться не будет.

В современных сетевых вычислительных комплексах решение вопросов организации взаимоисключений
при использовании общей линии связи, как правило, также находится вне компетенции операционных
систем и вынесено на физический уровень организации взаимодействия. Ответственность за корректное
использование коммуникаций возлагается на сетевые адаптеры, поэтому подобные проблемы мы тоже
рассматривать не будем.

Из приведенного перечня мы с вами подробнее остановимся на решении вопросов, приведенных в пунк-
тах 3-5.

**Понятие протокола**

Для описания происходящего в автономной операционной системе в лекции 2 было введено основопола-
гающее понятие "процесс", на котором, по сути дела, базируется весь наш курс. Для того чтобы описать
взаимодействие удаленных процессов и понять, какие функции и как должны выполнять дополнитель-
ные части сетевых операционных систем, отвечающих за такое взаимодействие, нам понадобится не ме-
нее фундаментальное понятие - протокол.

"Общение" локальных процессов напоминает общение людей, проживающих в одном городе. Если взаи-
модействующие процессы находятся под управлением различных операционных систем, то эта ситуация
подобна общению людей, проживающих в разных городах и, возможно, в разных странах.

Каким образом два человека, находящиеся в разных городах, а тем более странах, могут обмениваться
информацией? Для этого им обычно приходится прибегать к услугам соответствующих служб связи. При
этом между службами связи различных городов (государств) должны быть заключены определенные со-
глашения, позволяющие корректно организовывать такой обмен. Если речь идет, например, о почтовых
отправлениях, то в первую очередь необходимо договориться о том, что может представлять собой поч-
товое отправление, какой вид оно может иметь. Некоторые племена индейцев для передачи информации
пользовались узелковыми письмами - поясами, к которым крепились веревочки с различным числом и
формой узелков. Если бы такое письмо попало в современный почтовый ящик, то, пожалуй, ни одно от-
деление связи не догадалось бы, что это - письмо, и пояс был бы выброшен как ненужный хлам. Помимо
формы представления информации необходима договоренность о том, какой служебной информацией
должно снабжаться почтовое отправление (адрес получателя, срочность доставки, способ пересылки: по-
ездом, авиацией, с помощью курьера и т. п.) и в каком формате она должна быть представлена. Адреса,

**Основы операционных систем**

**149**

например, в России и в США принято записывать по-разному. В России мы начинаем адрес со страны,
далее указывается город, улица и квартира. В США все наоборот: сначала указывается квартира, затем
улица и т. д. Конечно, даже при неправильной записи адреса письмо, скорее всего, дойдет до получателя,
но можно себе представить растерянность почтальона, пытающегося разгадать, что это за страна или го-
род - "кв.162"? Как видим, доставка почтового отправления из одного города (страны) в другой требует
целого ряда соглашений между почтовыми ведомствами этих городов (стран).

Аналогичная ситуация возникает и при общении удаленных процессов, работающих под управлением
разных операционных систем. Здесь процессы играют роль людей, проживающих в разных городах, а
сетевые части операционных систем - роль соответствующих служб связи. Для того чтобы удаленные
процессы могли обмениваться данными, необходимо, чтобы сетевые части операционных систем руко-
водствовались определенными соглашениями, или, как принято говорить, поддерживали определенные
протоколы. Термин "протокол" уже встречался нам в лекции 13, посвященной организации ввода-вывода
в операционных системах, при обсуждении понятия "шина". Мы говорили, что понятие шины подразу-
мевает не только набор проводников, но и набор жестко заданных протоколов, определяющий перечень
сообщений, который может быть передан с помощью электрических сигналов по этим проводникам, т. е.
в "протокол" мы вкладывали практически тот же смысл. В следующем разделе мы попытаемся дать более
формализованное определение этого термина.

Необходимо отметить, что и локальные процессы при общении также должны руководствоваться опре-
деленными соглашениями или поддерживать определенные протоколы. Только в автономных операци-
онных системах они несколько завуалированы. В роли таких протоколов выступают специальная после-
довательность системных вызовов при организации взаимодействия процессов и соглашения о парамет-
рах системных вызовов.

Различные способы решения проблем 3-5, поднятых в предыдущем разделе, по существу, представляют
собой различные соглашения, которых должны придерживаться сетевые части операционных систем, т.
е. различные сетевые протоколы. Именно наличие сетевых протоколов позволяет организовать взаимо-
действие удаленных процессов.

При рассмотрении перечисленных выше проблем необходимо учитывать, с какими сетями мы имеем де-
ло.

В литературе принято говорить о локальных вычислительных сетях (LAN - Local Area Network) и
глобальных вычислительных сетях (WAN - Wide Area Network). Строгого определения этим поня-
тиям обычно не дается, а принадлежность сети к тому или иному типу часто определяется взаимным
расположением вычислительных комплексов, объединенных в сеть. Так, например, в большинстве работ
к локальным сетям относят сети, состоящие из компьютеров одной организации, размещенные в преде-
лах одного или нескольких зданий, а к глобальным сетям - сети, охватывающие все компьютеры в не-
скольких городах и более. Зачастую вводится дополнительный термин для описания сетей промежуточ-
ного масштаба - муниципальных или городских вычислительных сетей (MAN - Metropolitan Area
Network) - сетей, объединяющих компьютеры различных организаций в пределах одного города или од-
ного городского района. Таким образом, упрощенно можно рассматривать глобальные сети как сети, со-
стоящие из локальных и муниципальных сетей. А муниципальные сети, в свою очередь, могут состоять
из нескольких локальных сетей. На самом деле деление сетей на локальные, глобальные и муниципаль-
ные обычно связано не столько с местоположением и принадлежностью вычислительных систем, соеди-
ненных сетью, сколько с различными подходами, применяемыми для решения поставленных вопросов в
рамках той или иной сети, - с различными используемыми протоколами.

**Многоуровневая модель построения сетевых вычислительных систем**

Даже беглого взгляда на перечень проблем, связанных с логической организацией взаимодействия уда-
ленных процессов, достаточно, чтобы понять, что построение сетевых средств связи - задача более
сложная, чем реализация локальных средств связи. Поэтому обычно задачу создания таких средств ре-
шают по частям, применяя уже неоднократно упоминавшийся нами "слоеный", или многоуровневый,
подход.

**Основы операционных систем**

**150**

Как уже отмечалось при обсуждении "слоеного" строения операционных систем на первой лекции, при
таком подходе уровень N системы предоставляет сервисы уровню N+1, пользуясь в свою очередь только
сервисами уровня N-1. Следовательно, каждый уровень может взаимодействовать непосредственно толь-
ко со своими соседями, руководствуясь четко закрепленными соглашениями - вертикальными протоко-
лами, которые принято называть интерфейсами.

Самым нижним уровнем в слоеных сетевых вычислительных системах является уровень, на котором реа-
лизуется реальная физическая связь между двумя узлами сети. Из предыдущего раздела следует, что для
обеспечения обмена физическими сигналами между двумя различными вычислительными системами не-
обходимо, чтобы эти системы поддерживали определенный протокол физического взаимодействия - го-
ризонтальный протокол.

На самом верхнем уровне находятся пользовательские процессы, которые инициируют обмен данными.
Количество и функции промежуточных уровней варьируются от одной системы к другой. Вернемся к
нашей аналогии с пересылкой почтовых отправлений между людьми, проживающими в разных городах,
правда, с порядком их пересылки несколько отличным от привычного житейского порядка. Рассмотрим в
качестве пользовательских процессов руководителей различных организаций, желающих обменяться
письмами. Руководитель (пользовательский процесс) готовит текст письма (данные) для пересылки в
другую организацию. Затем он отдает его своему секретарю (совершает системный вызов - обращение к
нижележащему уровню), сообщая, кому и куда должно быть направлено письмо. Секретарь снимает с
него копию и выясняет адрес организации. Далее идет обращение к нижестоящему уровню, допустим,
письмо направляется в канцелярию. Здесь оно регистрируется (ему присваивается порядковый номер),
один экземпляр запечатывается в конверт, на котором указывается, кому и куда адресовано письмо, впе-
чатывается адрес отправителя. Копия остается в канцелярии, а конверт отправляется на почту (переход
на следующий уровень). На почте наклеиваются марки и делаются другие служебные пометки, определя-
ется способ доставки корреспонденции и т. д. Наконец, поездом, самолетом или курьером (физический
уровень!) письмо следует в другой город, в котором все уровни проходятся в обратном порядке. Пользо-
вательский уровень (руководитель) после подготовки исходных данных и инициации процесса взаимо-
действия далее судьбой почтового отправления не занимается. Все остальное (включая, быть может, про-
верку его доставки и посылку копии в случае утери) выполняют нижележащие уровни.

Заметим, что на каждом уровне взаимодействия в городе отправителя исходные данные (текст письма)
обрастают дополнительной служебной информацией. Соответствующие уровни почтовой службы в го-
роде получателя должны уметь понимать эту служебную информацию. Таким образом, для одинаковых
уровней в различных городах необходимо следование специальным соглашениям - поддержка опреде-
ленных горизонтальных протоколов.

Точно так же в сетевых вычислительных системах все их одинаковые уровни, лежащие выше физическо-
го, виртуально обмениваются данными посредством горизонтальных протоколов. Наличие такой вирту-
альной связи означает, что уровень N компьютера 2 должен получить ту же самую информацию, которая
была отправлена уровнем N компьютера 1. Хотя в реальности эта информация должна была сначала дой-
ти сверху вниз до уровня 1 компьютера 1, затем передана уровню 1 компьютера 2 и только после этого
доставлена снизу вверх уровню N этого компьютера.

Формальный перечень правил, определяющих последовательность и формат сообщений, которыми об-
мениваются сетевые компоненты различных вычислительных систем, лежащие на одном уровне, мы и
будем называть сетевым протоколом.

Всю совокупность вертикальных и горизонтальных протоколов (интерфейсов и сетевых протоколов) в
сетевых системах, построенных по " слоеному" принципу, достаточную для организации взаимодействия
удаленных процессов, принято называть семейством протоколов или стеком протоколов. Сети, по-
строенные на основе разных стеков протоколов, могут быть объединены между собой с использованием
вычислительных устройств, осуществляющих трансляцию из одного стека протоколов в другой, причем
на различных уровнях слоеной модели

Эталоном многоуровневой схемы построения сетевых средств связи считается семиуровневая модель от-
крытого взаимодействия систем (Open System Interconnection - OSI), предложенная Международной ор-

**Основы операционных систем**

**151**

ганизацией Стандартов (International Standard Organization - ISO) и получившая сокращенное наименова-
ние OSI/ISO (см. рис. 14.1).

Давайте очень кратко опишем, какие функции выполняют различные уровни модели OSI/ISO ГОлифер,
2001]:

• Уровень 1 - физический. Этот уровень связан с работой hardware. На нем определяются физиче-
ские аспекты передачи информации по линиям связи, такие как: напряжения, частоты, природа
передающей среды, способ передачи двоичной информации по физическому носителю, вплоть до
размеров и формы используемых разъемов. В компьютерах за поддержку физического уровня
обычно отвечает сетевой адаптер.

• Уровень 2 - канальный. Этот уровень отвечает за передачу данных по физическому уровню без
искажений между непосредственно связанными узлами сети. На нем формируются физические
пакеты данных для реальной доставки по физическому уровню. Протоколы канального уровня
реализуются совместно сетевыми адаптерами и их драйверами (понятие драйвера рассматрива-
лось в лекции 13).

• Уровень 3 - сетевой. Сетевой уровень несет ответственность за доставку информации от узла-
отправителя к узлу-получателю. На этом уровне частично решаются вопросы адресации, осущест-
вляется выбор маршрутов следования пакетов данных, решаются вопросы стыковки сетей, а также
управление скоростью передачи информации для предотвращения перегрузок в сети.

• Уровень 4 - транспортный. Регламентирует передачу данных между удаленными процессами.
Обеспечивает доставку информации вышележащим уровнем с необходимой степенью надежно-
сти, компенсируя, быть может, ненадежность нижележащих уровней, связанную с искажением и
потерей данных или доставкой пакетов в неправильном порядке. Наряду с сетевым уровнем мо-
жет управлять скоростью передачи данных и частично решать проблемы адресации.

**Уровень приложений**

**Компьютер**

**Компьютер\_**

*у* **ровень приложений**

*у* **ровень пре (ставления
данных**

*У* **ровень пре [ставления
данных**

**('еансовый уровень**

**1 'еансовый уровень**

**1 ранспорп и nil уровень**

**ГранспорлмNil уровень**

**('етевой уровень**

**< 'ел свой уровень**

**Канальный уровень**

**Канальный уровень**

**Фи (ический уровень**

**Фи шческий уровень**

Рис. 14.1. Семиуровневая эталонная модель OSI/ISO



**Основы операционных систем**

**152**

• Уровень 5 - сеансовый. Координирует взаимодействие связывающихся процессов. Основная зада-
ча - предоставление средств синхронизации взаимодействующих процессов. Такие средства син-
хронизации позволяют создавать контрольные точки при передаче больших объемов информации.
В случае сбоя в работе сети передачу данных можно возобновить с последней контрольной точки,
а не начинать заново.

• Уровень 6 - уровень представления данных. Отвечает за форму представления данных, перекоди-
рует текстовую и графическую информацию из одного формата в другой, обеспечивает ее сжатие
и распаковку, шифрование и декодирование.

• Уровень 7 - прикладной. Служит для организации интерфейса между пользователем и сетью. На
этом уровне реализуются такие сервисы, как удаленная передача данных, удаленный терминаль-
ный доступ, почтовая служба и работа во Всемирной паутине (Web-браузеры).

Надо отметить, что к приведенной эталонной модели большинство практиков относится без излишнего
пиетета. Эта модель не предвосхитила появления различных семейств протоколов, таких как, например,
семейство протоколов TCP/IP, а наоборот, была создана под их влиянием. Ее не следует рассматривать
как готовый оптимальный чертеж для создания любого сетевого средства связи. Наличие некоторой
функции на определенном уровне не гарантирует, что это ее наилучшее место, некоторые функции (на-
пример, коррекция ошибок) дублируются на нескольких уровнях, да и само деление на 7 уровней носит
отчасти произвольный характер. Хотя в конце концов были созданы работающие реализации этой моде-
ли, но наиболее распространенные семейства протоколов лишь до некоторой степени согласуются с ней.
Как отмечено в книге ГТаненбаум, 20021, она больше подходит для реализации телефонных, а не вычис-
лительных сетей. Ценность предложенной эталонной модели заключается в том, что она показывает на-
правление, в котором должны двигаться разработчики новых вычислительных сетей.

Проблемы 3-5, перечисленные в разделе "Основные вопросы логической организации передачи инфор-
мации" между удаленными процессами, относятся в основном к сетевому и транспортному уровням эта-
лонной модели и, соответственно, решаются на уровне сетевых и транспортных протоколов. Давайте
приступим, наконец, к их рассмотрению.

**Проблемы адресации в сети**

Любой пакет информации, передаваемый по сети, должен быть снабжен адресом получателя. Если взаи-
модействие подразумевает двустороннее общение, то в пакет следует также включить и адрес отправите-
ля. В лекции 4 мы описали один из протоколов организации надежной связи с использованием контроль-
ных сумм, нумерации пакетов и подтверждения получения неискаженного пакета в правильном порядке.
Для отправки подтверждений обратный адрес также следует включать в пересылаемый пакет. Таким об-
разом, практически каждый сетевой пакет информации должен быть снабжен адресом получателя и ад-
ресом отправителя. Как могут выглядеть такие адреса?

Несколько раньше, обсуждая отличия взаимодействия удаленных процессов от взаимодействия локаль-
ных процессов, мы говорили, что удаленные адресаты должны обладать уникальными адресами уже не в
пределах одного компьютера, а в рамках всей сети. Существует два подхода к наделению объектов таки-
ми сетевыми адресами: одноуровневый и двухуровневый.

Одноуровневые адреса

В небольших компьютерных сетях можно построить одноуровневую систему адресации. При таком под-
ходе каждый процесс, желающий стать участником удаленного взаимодействия (при прямой адресации),
и каждый объект, для такого взаимодействия предназначенный (при непрямой адресации), получают по
мере необходимости собственные адреса (символьные или числовые), а сами вычислительные комплек-
сы, объединенные в сеть, никаких самостоятельных адресов не имеют. Подобный метод требует доволь-
но сложного протокола обеспечения уникальности адресов. Вычислительный комплекс, на котором за-
пускается взаимодействующий процесс, должен запросить все компьютеры сети о возможности присвое-
ния процессу некоторого адреса. Только после получения от них согласия процессу может быть назначен
адрес. Поскольку процесс, посылающий данные другому процессу, не может знать, на каком компоненте
сети находится процесс-адресат, передаваемая информация должна быть направлена всем компонентам
сети (так называемое широковещательное сообщение - broadcast message), проанализирована ими и

**Основы операционных систем**

**153**

либо отброшена (если процесса-адресата на данном компьютере нет), либо доставлена по назначению.
Так как все данные постоянно передаются от одного комплекса ко всем остальным, такую одноуровне-
вую схему обычно применяют только в локальных сетях с прямой физической связью всех компьютеров
между собой (например, в сети NetBIOS на базе Ethernet), но она является существенно менее эффектив-
ной, чем двухуровневая схема адресации.

Двухуровневые адреса

При двухуровневой адресации полный сетевой адрес процесса или промежуточного объекта для хране-
ния данных складывается из двух частей - адреса вычислительного комплекса, на котором находится
процесс или объект в сети (удаленного адреса), и адреса самого процесса или объекта на этом вычисли-
тельном комплексе (локального адреса). Уникальность полного адреса будет обеспечиваться уникально-
стью удаленного адреса для каждого компьютера в сети и уникальностью локальных адресов объектов на
компьютере. Давайте подробнее рассмотрим проблемы, возникающие для каждого из компонентов пол-
ного адреса.

Удаленная адресация и разрешение адресов

Инициатором связи процессов друг с другом всегда является человек, будь то программист или обычный
пользователь. Как мы неоднократно отмечали в лекциях, человеку свойственно думать словами, он легче
воспринимает символьную информацию. Поэтому очевидно, что каждая машина в сети получает сим-
вольное, часто даже содержательное имя. Компьютер не разбирается в смысловом содержании символов,
ему проще оперировать числами, желательно одного и того же формата, которые помещаются, например,
в 4 байта или в 16 байт. Поэтому каждый компьютер в сети для удобства работы вычислительных систем
получает числовой адрес. Возникает проблема отображения пространства символьных имен (или адре-
сов) вычислительных комплексов в пространство их числовых адресов. Эта проблема получила наимено-
вание проблемы разрешения адресов.

С подобными задачами мы уже сталкивались, обсуждая организацию памяти в вычислительных системах
(отображение имен переменных в их адреса в процессе компиляции и редактирования связей) и органи-
зацию файловых систем (отображение имен файлов в их расположении на диске). Посмотрим, как она
может быть решена в сетевом варианте.

Первый способ решения заключается в том, что на каждом сетевом компьютере создается файл, содер-
жащий имена всех машин, доступных по сети, и их числовые эквиваленты. Обращаясь к этому файлу,
операционная система легко может перевести символьный удаленный адрес в числовую форму. Такой
подход использовался на заре эпохи глобальных сетей и применяется в изолированных локальных сетях
в настоящее время. Действительно, легко поддерживать файл соответствий в корректном виде, внося в
него необходимые изменения, когда общее число сетевых машин не превышает нескольких десятков.

Как правило, изменения вносятся на некотором выделенном административном вычислительном ком-
плексе, откуда затем обновленный файл рассылается по всем компонентам сети.

В современной сетевой паутине этот подход является неприемлемым. Дело даже не в размерах подобно-
го файла, а в частоте требуемых обновлений и в огромном количестве рассылок, что может полностью
подорвать производительность сети. Проблема состоит в том, что добавление или удаление компонента
сети требует внесения изменений в файлы на всех сетевых машинах. Второй метод разрешения адресов
заключается в частичном распределении информации о соответствии символьных и числовых адресов по
многим комплексам сети, так что каждый из этих комплексов содержит лишь часть полных данных. Он
же определяет и правила построения символических имен компьютеров.

Один из таких способов, используемый в Internet, получил английское наименование Domain Name
Service или сокращенно DNS. Эта аббревиатура широко используется и в русскоязычной литературе. Да-
вайте рассмотрим данный метод подробнее.

Организуем логически все компьютеры сети в некоторую древовидную структуру, напоминающую
структуру директорий файловых систем, в которых отсутствует возможность организации жестких и
мягких связей и нет пустых директорий. Будем рассматривать все компьютеры, входящие во Всемирную

**Основы операционных систем**

**154**

сеть, как область самого низкого ранга (аналог корневой директории в файловой системе) - ранга 0. Ра-
зобьем все множество компьютеров области на какое-то количество подобластей (domains). При этом не-
которые подобласти будут состоять из одного компьютера (аналоги регулярных файлов в файловых сис-
темах), а некоторые - более чем из одного компьютера (аналоги директорий в файловых системах). Каж-
дую подобласть будем рассматривать как область более высокого ранга. Присвоим подобластям собст-
венные имена таким образом, чтобы в рамках разбиваемой области все они были уникальны. Повторим
такое разбиение рекурсивно для каждой области более высокого ранга, которая состоит более чем из од-
ного компьютера, несколько раз, пока при последнем разбиении в каждой подобласти не окажется ровно
по одному компьютеру. Глубина рекурсии для различных областей одного ранга может быть разной, но
обычно в целом ограничиваются 3 - 5 разбиениями, начиная от ранга 0.

В результате мы получим дерево, неименованной вершиной которого является область, объединяющая
все компьютеры, входящие во Всемирную сеть, именованными терминальными узлами - отдельные
компьютеры (точнее - подобласти, состоящие из отдельных компьютеров), а именованными нетерми-
нальными узлами - области различных рангов. Используем полученную структуру для построения имен
компьютеров, подобно тому как мы поступали при построении полных имен файлов в структуре дирек-
торий файловой системы. Только теперь, двигаясь от корневой вершины к терминальному узлу - отдель-
ному компьютеру, будем вести запись имен подобластей справа налево и отделять имена друг от друга с
помощью символа ".".

Допустим, некоторая подобласть, состоящая из одного компьютера, получила имя serv, она входит в по-
добласть, объединяющую все компьютеры некоторой лаборатории, с именем crec. Та, в свою очередь,
входит в подобласть всех компьютеров Московского физико-технического института с именем mipt, ко-
торая включается в область ранга 1 всех компьютеров России с именем ru. Тогда имя рассматриваемого
компьютера во Всемирной сети будет serv.crec.mipt.ru. Аналогичным образом можно именовать и подоб-
ласти, состоящие более чем из одного компьютера.

В каждой полученной именованной области, состоящей более чем из одного узла, выберем один из ком-
пьютеров и назначим его ответственным за эту область - сервером DNS. Сервер DNS знает числовые ад-
реса серверов DNS для подобластей, входящих в его зону ответственности, или числовые адреса отдель-
ных компьютеров, если такая подобласть включает в себя только один компьютер. Кроме того, он также
знает числовой адрес сервера DNS, в зону ответственности которого входит рассматриваемая область
(если это не область ранга 1), или числовые адреса всех серверов DNS ранга 1 (в противном случае). От-
дельные компьютеры всегда знают числовые адреса серверов DNS, которые непосредственно за них от-
вечают.

Рассмотрим теперь, как процесс на компьютере serv.crec.mipt.ru может узнать числовой адрес компьюте-
ра ssp.brown.edu. Для этого он обращается к своему DNS-серверу, отвечающему за область crec.mipt.ru, и
передает ему нужный адрес в символьном виде. Если этот DNS-сервер не может сразу представить необ-
ходимый числовой адрес, он передает запрос DNS-серверу, отвечающему за область mipt.ru. Если и тот
не в силах самостоятельно справиться с проблемой, он перенаправляет запрос серверу DNS, отвечающе-
му за область 1-го ранга ru. Этот сервер может обратиться к серверу DNS, обслуживающему область 1-го
ранга edu, который, наконец, затребует информацию от сервера DNS области brown.edu, где должен быть
нужный числовой адрес. Полученный числовой адрес по всей цепи серверов DNS в обратном порядке
будет передан процессу, направившему запрос (см. рис. 14.2).

**Основы операционных систем**

**155**

Рис. 14.2. Пример разрешения имен с использованием DNS-серверов

В действительности, каждый сервер DNS имеет достаточно большой кэш, содержащий адреса серверов
DNS для всех последних запросов. Поэтому реальная схема обычно существенно проще, из приведенной
цепочки общения DNS-серверов выпадают многие звенья за счет обращения напрямую.

Рассмотренный способ разрешения адресов позволяет легко добавлять компьютеры в сеть и исключать
их из сети, так как для этого необходимо внести изменения только на DNS-сервере соответствующей об-
ласти.

Если DNS-сервер, отвечающий за какую-либо область, выйдет из строя, то может оказаться невозмож-
ным разрешение адресов для всех компьютеров этой области. Поэтому обычно назначается не один сер-
вер DNS, а два - основной и запасной. В случае выхода из строя основного сервера его функции немед-
ленно начинает выполнять запасной.

В реальных сетевых вычислительных системах обычно используется комбинация рассмотренных подхо-
дов. Для компьютеров, с которыми чаще всего приходится устанавливать связь, в специальном файле
хранится таблица соответствий символьных и числовых адресов. Все остальные адреса разрешаются с
использованием служб, аналогичных службе DNS. Способ построения удаленных адресов и методы раз-
решения адресов обычно определяются протоколами сетевого уровня эталонной модели.

Мы разобрались с проблемой удаленных адресов и знаем, как получить числовой удаленный адрес нуж-
ного нам компьютера. Давайте рассмотрим теперь проблему адресов локальных: как нам задать адрес
процесса или объекта для хранения данных на удаленном компьютере, который в конечном итоге и дол-
жен получить переданную информацию.

Локальная адресация. Понятие порта

Во второй лекции мы говорили, что каждый процесс, существующий в данный момент в вычислительной
системе, уже имеет собственный уникальный номер - PID. Но этот номер неудобно использовать в каче-
стве локального адреса процесса при организации удаленной связи. Номер, который получает процесс
при рождении, определяется моментом его запуска, предысторией работы вычислительного комплекса и
является в значительной степени случайным числом, изменяющимся от запуска к запуску. Представьте
себе, что адресат, с которым вы часто переписываетесь, постоянно переезжает с место на место, меняя
адреса, так что, посылая очередное письмо, вы не можете с уверенностью сказать, где он сейчас прожи-
вает, и поймете все неудобство использования идентификатора процесса в качестве его локального адре-
са. Все сказанное выше справедливо и для идентификаторов промежуточных объектов, использующихся
при локальном взаимодействии процессов в схемах с непрямой адресацией.

**Основы операционных систем**

**156**

Для локальной адресации процессов и промежуточных объектов при удаленной связи обычно организу-
ется новое специальное адресное пространство, например представляющее собой ограниченный набор
положительных целочисленных значений или множество символических имен, аналогичных полным
именам файлов в файловых системах. Каждый процесс, желающий принять участие в сетевом взаимо-
действии, после рождения закрепляет за собой один или несколько адресов в этом адресном пространст-
ве. Каждому промежуточному объекту при его создании присваивается свой адрес из этого адресного
пространства. При этом удаленные пользователи могут заранее договориться о том, какие именно адреса
будут зарезервированы для данного процесса, независимо от времени его старта, или для данного объек-
та, независимо от момента его создания. Подобные адреса получили название портов, по аналогии с пор-
тами ввода-вывода.

Необходимо отметить, что в системе может существовать несколько таких адресных пространств для
различных способов связи. При получении данных от удаленного процесса операционная система смот-
рит, на какой порт и для какого способа связи они были отправлены, определяет процесс, который заявил
этот порт в качестве своего адреса, или объект, которому присвоен данный адрес, и доставляет получен-
ную информацию адресату. Виды адресного пространства портов (т. е. способы построения локальных
адресов) определяются, как правило, протоколами транспортного уровня эталонной модели.

Полные адреса. Понятие сокета (socket)

Таким образом, полный адрес удаленного процесса или промежуточного объекта для конкретного спосо-
ба связи с точки зрения операционных систем определяется парой адресов: <числовой адрес компьютера
в сети, порт>. Подобная пара получила наименование socket (в переводе - "гнездо" или, как стали писать
в последнее время, сокет), а сам способ их использования - организация связи с помощью сокетов. В
случае непрямой адресации с использованием промежуточных объектов сами эти объекты также принято
называть сокетами. Поскольку разные протоколы транспортного уровня требуют разных адресных про-
странств портов, то для каждой пары надо указывать, какой транспортный протокол она использует, -
говорят о разных типах сокетов.

В современных сетевых системах числовой адрес обычно получает не сам вычислительный комплекс, а
его сетевой адаптер, с помощью которого комплекс подключается к линии связи. При наличии несколь-
ких сетевых адаптеров для разных линий связи один и тот же вычислительный комплекс может иметь
несколько числовых адресов. В таких системах полные адреса удаленного адресата (процесса или про-
межуточного объекта) задаются парами <числовой адрес сетевого адаптера, порт> и требуют доставки
информации через указанный сетевой адаптер.

**Проблемы маршрутизации в сетях**

При наличии прямой линии связи между двумя компьютерами обычно не возникает вопросов о том, ка-
ким именно путем должна быть доставлена информация. Но, как уже упоминалось, одно из отличий
взаимодействия удаленных процессов от взаимодействия процессов локальных состоит в использовании
в большинстве случаев процессов-посредников, расположенных на вычислительных комплексах, не яв-
ляющихся комплексами отправителя и получателя. В сложных топологических схемах организации сетей
информация между двумя компьютерами может передаваться по различным путям. Возникает вопрос:
как организовать работу операционных систем на комплексах -участниках связи (это могут быть конеч-
ные или промежуточные комплексы) для определения маршрута передачи данных? По какой из несколь-
ких линий связи (или через какой сетевой адаптер) нужно отправить пакет информации? Какие протоко-
лы маршрутизации возможны? Существует два принципиально разных подхода к решению этой пробле-
мы: маршрутизация от источника передачи данных и одношаговая маршрутизация.

• Маршрутизация от источника передачи данных. При маршрутизации от источника данных пол-
ный маршрут передачи пакета по сети формируется на компьютере-отправителе в виде последова-
тельности числовых адресов сетевых адаптеров, через которые должен пройти пакет, чтобы доб-
раться до компьютера-получателя, и целиком включается в состав этого пакета. В этом случае
промежуточные компоненты сети при определении дальнейшего направления движения пакета не
принимают самостоятельно никаких решений, а следуют указаниям, содержащимся в пакете.

**Основы операционных систем**

**157**

• Одношаговая маршрутизация. При одношаговой маршрутизации каждый компонент сети, прини-
мающий участие в передаче информации, самостоятельно определяет, какому следующему ком-
поненту, находящемуся в зоне прямого доступа, она должна быть отправлена. Решение принима-
ется на основании анализа содержащегося в пакете адреса получателя. Полный маршрут передачи
данных складывается из одношаговых решений, принятых компонентами сети.

Маршрутизация от источника передачи данных легко реализуется на промежуточных компонентах сети,
но требует полного знания маршрутов на конечных компонентах. Она достаточно редко используется в
современных сетевых системах, и далее мы ее рассматривать не будем.

Для работы алгоритмов одношаговой маршрутизации, которые являются основой соответствующих
протоколов, на каждом компоненте сети, имеющем возможность передавать информацию более чем од-
ному компоненту, обычно строится специальная таблица маршрутов (см. рис. 14.3). В простейшем слу-
чае каждая запись такой таблицы содержит: адрес вычислительного комплекса получателя; адрес компо-
нента сети, напрямую подсоединенного к данному, которому следует отправить пакет, предназначенный
для этого получателя; указание о том, по какой линии связи (через какой сетевой адаптер) должен быть
отправлен пакет. Поскольку получателей в сети существует огромное количество, для сокращения числа
записей в таблице маршрутизации обычно прибегают к двум специальным приемам.

Во-первых, числовые адреса топологически близко расположенных комплексов (например, комплексов,
принадлежащих одной локальной вычислительной сети) стараются выбирать из последовательного диа-
пазона адресов. В этом случае запись в таблице маршрутизации может содержать не адрес конкретного
получателя, а диапазон адресов для некоторой сети (номер сети).

Во-вторых, если для очень многих получателей в качестве очередного узла маршрута используется один
и тот же компонент сети, а остальные маршруты выбираются для ограниченного числа получателей, то в
таблицу явно заносятся только записи для этого небольшого количества получателей, а для маршрута,
ведущего к большей части всей сети, делается одна запись - маршрутизация по умолчанию (default).
Пример простой таблицы маршрутизации для некоторого комплекса некой абстрактной сети приведен
ниже:

|  |  |  |
| --- | --- | --- |
| **Адресат на значения** | **Адрес очередногокомпонента** | **Адрес исходящей**ЛИНИИ СВЯ 'II |
| 5-12 | 20 | 21 |
| 1-4 | 15 | 2: |
| **default** | 28 | 24 |

Рис. 14.3. Простая таблица маршрутизации

По способам формирования и использования таблиц маршрутизации алгоритмы одношаговой маршру-
тизации можно разделить на три класса:

• алгоритмы фиксированной маршрутизации;

• алгоритмы простой маршрутизации;

• алгоритмы динамической маршрутизации.

При фиксированной маршрутизации таблица, как правило, создается в процессе загрузки операционной
системы. Все записи в ней являются статическими. Линия связи, которая будет использоваться для дос-
тавки информации от данного узла к некоторому узлу A в сети, выбирается раз и навсегда. Обычно ли-
нии выбирают так, чтобы минимизировать полное время доставки данных. Преимуществом этой страте-
гии является простота реализации. Основной же недостаток заключается в том, что при отказе выбран-
ной линии связи данные не будут доставлены, даже если существует другой физический путь для их пе-
редачи.

**Основы операционных систем**

**158**

В алгоритмах простой маршрутизации таблица либо не используется совсем, либо строится на основе
анализа адресов отправителей приходящих пакетов информации. Различают несколько видов простой
маршрутизации - случайную, лавинную и маршрутизацию по прецедентам. При случайной маршру-
тизации прибывший пакет отсылается в первом попавшемся направлении, кроме исходного. При лавин-
ной маршрутизации один и тот же пакет рассылается по всем направлениям, кроме исходного. Случайная
и лавинная маршрутизации, естественно, не используют таблиц маршрутов. При маршрутизации по пре-
цедентам таблица маршрутизации строится по предыдущему опыту, исходя из анализа адресов отправи-
телей приходящих пакетов. Если прибывший пакет адресован компоненту сети, от которого когда-либо
приходили данные, то соответствующая запись об этом содержится в таблице маршрутов, и для даль-
нейшей передачи пакета выбирается линия связи, указанная в таблице. Если такой записи нет, то пакет
может быть отослан случайным или лавинным способом. Алгоритмы простой маршрутизации действи-
тельно просты в реализации, но отнюдь не гарантируют доставку пакета указанному адресату за прием-
лемое время и по рациональному маршруту без перегрузки сети.

Наиболее гибкими являются алгоритмы динамической или адаптивной маршрутизации, которые
умеют обновлять содержимое таблиц маршрутов на основе обработки специальных сообщений, прихо-
дящих от других компонентов сети, занимающихся маршрутизацией, удовлетворяющих определенному
протоколу. Такие алгоритмы принято делить на два подкласса: алгоритмы дистанционно-векторные и
алгоритмы состояния связей.

При дистанционно-векторной маршрутизации компоненты операционных систем на соседних вычисли-
тельных комплексах сети, занимающиеся выбором маршрута (их принято называть маршрутизатор или
router), периодически обмениваются векторами, которые представляют собой информацию о расстояни-
ях от данного компонента до всех известных ему адресатов в сети. Под расстоянием обычно понимается
количество переходов между компонентами сети (hops), которые необходимо сделать, чтобы достичь
адресата, хотя возможно существование и других метрик, включающих скорость и/или стоимость пере-
дачи пакета по линии связи. Каждый такой вектор формируется на основании таблицы маршрутов. При-
шедшие от других комплексов векторы модернизируются с учетом расстояния, которое они прошли при
последней передаче. Затем в таблицу маршрутизации вносятся изменения, так чтобы в ней содержались
только маршруты с кратчайшими расстояниями. При достаточно длительной работе каждый маршрути-
затор будет иметь таблицу маршрутизации с оптимальными маршрутами ко всем потенциальным адреса-
там.

Векторно-дистанционные протоколы обеспечивают достаточно разумную маршрутизацию пакетов, но не
способны предотвратить возможность возникновения маршрутных петель при сбоях в работе сети. По-
этому векторно-дистанционная маршрутизация может быть эффективна только в относительно неболь-
ших сетях. Для больших сетей применяются алгоритмы состояния связей, на каждом маршрутизаторе
строящие графы сети, в качестве узлов которого выступают ее компоненты, а в качестве ребер, обла-
дающих стоимостью, существующие между ними линии связи. Маршрутизаторы периодически обмени-
ваются графами и вносят в них изменения. Выбор маршрута связан с поиском оптимального по стоимо-
сти пути по такому графу.

Подробное описание протоколов динамической маршрутизации можно найти в ГОлифер, 20021,
ГТаненбаум, 20031.

Обычно вычислительные сети используют смесь различных стратегий маршрутизации. Для одних адре-
сов назначения может использоваться фиксированная маршрутизация, для других - простая, для третьих
- динамическая. В локальных вычислительных сетях обычно используются алгоритмы фиксированной
маршрутизации, в отличие от глобальных вычислительных сетей, в которых в основном применяют ал-
горитмы адаптивной маршрутизации. Протоколы маршрутизации относятся к сетевому уровню эталон-
ной модели.

**Связь с установлением логического соединения и передача данных с помощью сообще-
ний**

Рассказывая об отличиях взаимодействия локальных и удаленных процессов, мы упомянули, что в осно-
ве всех средств связи на автономном компьютере так или иначе лежит механизм совместного использо-

**Основы операционных систем**

**159**

вания памяти, в то время как в основе всех средств связи между удаленными процессами лежит передача
сообщений. Неудивительно, что количество категорий средств удаленной связи сокращается до одной -
канальных средств связи. Обеспечивать интерфейс для сигнальных средств связи и разделяемой памяти,
базируясь на передаче пакетов данных, становится слишком сложно и дорого.

Рассматривая канальные средства связи для локальных процессов в лекции 4, мы говорили о существо-
вании двух моделей передачи данных по каналам связи (теперь мы можем говорить о двух принципиаль-
но разных видах протоколов организации канальной связи): поток ввода-вывода и сообщения. Для обще-
ния удаленных процессов применяются обе модели, однако теперь уже более простой моделью становит-
ся передача информации с помощью сообщений. Реализация различных моделей происходит на основе
протоколов транспортного уровня OSI/ISO.

Транспортные протоколы связи удаленных процессов, которые предназначены для обмена сообщениями,
получили наименование протоколов без установления логического соединения (connectionless) или

протоколов обмена датаграммами, поскольку само сообщение здесь принято называть датаграммой
(datagramm) или дейтаграммой. Каждое сообщение адресуется и посылается процессом индивидуаль-
но. С точки зрения операционных систем все датаграммы - это независимые единицы, не имеющие ниче-
го общего с другими датаграммами, которыми обмениваются эти же процессы.

Необходимо отметить, что с точки зрения процессов, обменивающихся информацией, датаграммы, ко-
нечно, могут быть связаны по содержанию друг с другом, но ответственность за установление и поддер-
жание этой семантической связи лежит не на сетевых частях операционных систем, а на самих пользова-
тельских взаимодействующих процессах (вышележащие уровни эталонной модели).

По-другому обстоит дело с транспортными протоколами, которые поддерживают потоковую модель.

Они получили наименование протоколов, требующих установления логического соединения
(connection-oriented). И в их основе лежит передача данных с помощью пакетов информации. Но опера-
ционные системы сами нарезают эти пакеты из передаваемого потока данных, организовывают правиль-
ную последовательность их получения и снова объединяют полученные пакеты в поток, так что с точки
зрения взаимодействующих процессов после установления логического соединения они имеют дело с
потоковым средством связи, напоминающим pipe или FIFO. Эти протоколы должны обеспечивать на-
дежную связь.

**Синхронизация удаленных процессов**

Мы рассмотрели основные принципы логической организации сетевых средств связи, внешние по отно-
шению к взаимодействующим процессам. Однако, как отмечалось в лекции 5, для корректной работы та-
ких процессов необходимо обеспечить определенную их синхронизацию, которая устранила бы возник-
новение race condition на соответствующих критических участках. Вопросы синхронизации удаленных
процессов обычно рассматриваются в курсах, посвященных распределенным операционным системам.
Интересующиеся этими вопросами могут обратиться к книгам [Silberschatz, 2002], [Таненбаум II, 2003].

**Заключение**

Основными причинами объединения компьютеров в вычислительные сети являются потребности в раз-
делении ресурсов, ускорении вычислений, повышении надежности и облегчении общения пользователей.

Вычислительные комплексы в сети могут находиться под управлением сетевых или распределенных вы-
числительных систем. Основой для объединения компьютеров в сеть служит взаимодействие удаленных
процессов. При рассмотрении вопросов организации взаимодействия удаленных процессов нужно при-
нимать во внимание основные отличия их кооперации от кооперации локальных процессов.

Базой для взаимодействия локальных процессов служит организация общей памяти, в то время как для
удаленных процессов - это обмен физическими пакетами данных.

Организация взаимодействия удаленных процессов требует от сетевых частей операционных систем
поддержки определенных протоколов. Сетевые средства связи обычно строятся по "слоеному" принципу.

**Основы операционных систем**

**160**

Формальный перечень правил, определяющих последовательность и формат сообщений, которыми об-
мениваются сетевые компоненты различных вычислительных систем, лежащие на одном уровне, называ-
ется сетевым протоколом. Каждый уровень слоеной системы может взаимодействовать непосредственно
только со своими вертикальными соседями, руководствуясь четко закрепленными соглашениями - вер-
тикальными протоколами или интерфейсами. Вся совокупность интерфейсов и сетевых протоколов в се-
тевых системах, построенных по слоеному принципу, достаточная для организации взаимодействия уда-
ленных процессов, образует семейство протоколов или стек протоколов.

Удаленные процессы, в отличие от локальных, при взаимодействии обычно требуют двухуровневой ад-
ресации при своем общении. Полный адрес процесса состоит из двух частей: удаленной и локальной.

Для удаленной адресации используются символьные и числовые имена узлов сети. Перевод имен из од-
ной формы в другую (разрешение имен) может осуществляться с помощью централизованно обновляе-
мых таблиц соответствия полностью на каждом узле или с использованием выделения зон ответственно-
сти специальных серверов. Для локальной адресации процессов применяются порты. Упорядоченная па-
ра из адреса узла в сети и порта получила название socket.

Для доставки сообщения от одного узла к другому могут использоваться различные протоколы маршру-
тизации.

С точки зрения пользовательских процессов обмен информацией может осуществляться в виде дата-
грамм или потока данных.