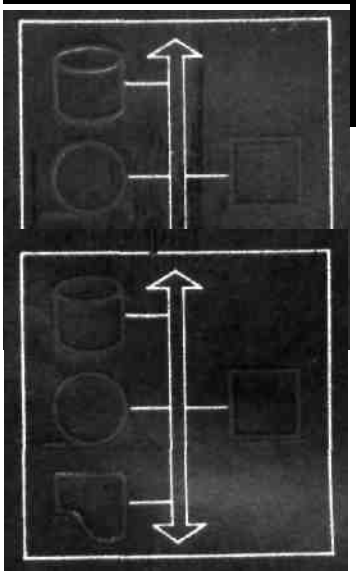
Программное обеспечение сетей СМ ЭВМ

Г. П. ВАСИЛЬЕВ
ГА. ЕГОРОВ
Н.Н. ЩЕРБИНА



Г.П.ВАСИЛЬЕВ
Г. А. ЕГОРОВ
Н. Н. ЩЕРБИНА

Программное обеспечение сетей СМ ЭВМ

Москва
Финансы и статистика
1983

ББК 32.973
В19

Васильев Г. П. и др.

В19 Программное обеспечение сетей СМ ЭВМ/Г. П. Васильев, Г. А. Егоров, Н. Н. Щербина. — М.: Финансы и статистика, **1983.—87 с., ил.** 30 к.

Описан конкретный пакет программ сетевой телеобработки, функционирующий под управлением операционной системы ОС РВ. Изложены структура и функции пакета, связи между процессами в сети, приведены описания соответствующих программных средств.

Книга предназначена для инженерно-технических и научных работников, занимающихся проектированием и эксплуатацией систем распределенной обработки данных, а также для аспирантов и студентов вузов соответствующих специальностей.

$$В\frac{2405000000-008}{010(01)-83}105-83$$

ББК 32.973

6Ф7.3

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

На сегодня сети ЭВМ представляют собой одну из наиболее прогрессивных форм организации средств вычислительной техники. Их появление, широкое и быстрое внедрение во все сферы человеческой деятельности обусловлено в первую очередь исключительной экономической эффективностью.

В настоящее время можно привести достаточно большое число примеров целесообразности создания и внедрения крупных систем автоматизации и сетей вычислительных центров на базе сетей ЭВМ. При этом наблюдается значительное снижение затрат на трудовые ресурсы в конкретном ВЦ, в некоторых случаях — в 2— 5 раз, при существенном улучшении предоставляемого вычислительного сервиса.

Структура и сложность сетей меняются в очень широком диапазоне: от специализированных, предназначенных для выполнения узкого, заранее определенного круга задач, до общих, решающих самые разные задачи для разнообразных групп пользователей. Поскольку сети ЭВМ являются большими системами и находятся на начальном этапе своего развития, то велики и число, и сложность проблем, возникающих при их построении. Сервис, предоставляемый сетями ЭВМ при доступе к информации и в ее обработке, в значительной степени определяется тем, что в сетях объединяются две отрасли техники: техника связи и вычислительная техника. При этом, как указывают некоторые специалисты, если технику связи можно охарактеризовать как отрасль довольно консервативную, развивающуюся с начала нашего века, имеющую хорошо поставленные проблемы, решаемые на основании продуманной и разработанной теории, то вычислительная техника очень нова, быстро меняется, ее фундаментальные проблемы плохо разработаны и она пока еще не стала наукой. Проектирование структур сетей, измерение различных их характеристик, анализ сложных системных моделей и создание принципов реализации сетевого программного обеспечения — все это складывается в растущее здание новой области знаний — теории сетей вычислительных машин.

Изложению некоторых решений одной из перечисленных выше проблем — проблемы создания сетевого программного обеспечения для ЭВМ серии СМ — и посвящена настоящая книга. Тематика рассмотренных в ней вопросов определяется прежде всего тем, что ЭВМ СМ-3 и СМ-4 уже в настоящее время являются наиболее распространенными минимашинами в стране и их выпуск продолжает расти. В связи с этим возрастает необходимость создания однородных сетей СМ ЭВМ на базе стандартного сетевого программного обеспечения. Таким программным обеспечением может служить пакет программ сетевой телеобработки ПП СТО/РВ, описываемый в данной книге.

Поскольку СМ ЭВМ поставляются с разными операционными системами, ориентированными на различные применения, то очевидно, что для создания единой сети из таких ЭВМ требуется наличие единой сетевой архитектуры. Рассмотрению такой сетевой архитектуры СМ ЭВМ (САСМ) посвящена первая глава. При этом значительное внимание уделяется* отображению логической и физической структур сети на ее программную структуру. В этой же главе приводится описание протоколов САСМ, их функций и возможностей.

Следующие две главы представляют наибольший интерес для пользователей-программистов, так как в них рассматриваются вопросы построения пользовательских программ, использующих услуги сети, а также возможности сервисных программ пакета и средства взаимодействия с ними.

Управление пакетом, выполняемое операторами узлов сети при помощи соответствующих программ, рассматривается в последней главе. В частности, описываются средства тестирования сети, использование которых позволяет получить статистику, необходимую для генерации наиболее эффективной для каждого конкретного случая версии пакета.

В заключение дается краткое описание других средств сетевой телеобработки СМ ЭВМ.

Авторы выражают надежду, что излагаемый в книге материал окажется полезным пользователям СМ ЭВМ в их практической работе.

ГЛАВА I

СЕТЕВАЯ ТЕЛЕОБРАБОТКА В СМ ЭВМ

1.1. АРХИТЕКТУРА И ФУНКЦИИ СЕТЕЙ ЭВМ

Практика использования отечественных и зарубежных систем распределенной обработки данных, базирующихся на сетях ЭВМ, позволяет выделить основные применения сетей и определить круг решаемых ими задач.

В области связи сети ЭВМ используются для реализации:

электронной почты, электронных систем передачи сообщений, предусматривающих средства редактирования, средства одновременной передачи одного сообщения нескольким абонентам и т. д.;

телеконференций — взаимодействия географически удаленных пользователей и групп пользователей с помощью сети ЭВМ.

Банковские и связанные с ними системы. Уже сейчас используется сетевое обслуживание ряда банков, страховых компаний и т. д. В первую очередь это относится 'к информационным системам кредитования, системам биржевой котировки, системам страхования и социального обеспечения.

В автоматизации работ учреждений широкое применение найдут как локальные, так и географически распределенные сети. Здесь в первую очередь стоит отметить их использование при резервировании авиа- и железнодорожных билетов, бронировании мест в гостиницах и т. д.

В сфере управления вычислительная техника используется уже весьма давно. Однако применение сетей, особенно локальных, базирующихся на мини- и микро-ЭВМ, позволяет значительно повысить качество управления.

В области торговли. Одной из наиболее перспективных областей применения сетей ЭВМ, по мнению большинства зарубежных специалистов, является сфера торговли. Здесь стоит выделить электронные рынки, торговые справочные системы, системы заказа товаров, системы сбора информации о торговых точек и др.

Материально-техническое снабжение. В настоящее время только в США насчитывается более 20 крупных автоматизированных систем учета материальных средств, использующих сетевое обслуживание.

Наука и образование. Большинство крупных научных центров как у нас в стране, так и за рубежом оснащены мощными вычислительными центрами. Однако необходимость в обмене информацией, планирование и проведение совместных экспериментов и др. требуют объединения таких автономных ВЦ в рамках развитых сетей. В СССР для этих целей создается Академсеть, многие научные центры США и Западной Европы имеют выходы в такие сети, как ARPANET, EURONET и TYMNET.

В области медицины. По мнению большинства специалистов, использование сетей ЭВМ в области медицины позволит в ближайшем будущем создавать архивы историй болезней, архивы диагностической информации и т. п.

Издательские системы редактирования представляют собой область возможного широкого применения локальных сетей на базе мини - и микро-ЭВМ. Эксперименты по внедрению и использованию такой локальной сети на базе нескольких ЭВМ PDP-11 проводятся в одном из западногерманских издательств.

Как следует из [1—3], для любой из перечисленных сфер применения в той или иной степени необходима реализация следующих основных функций:

обеспечение надежности хранения и защиты информации от несанкционированного доступа;

интерактивное взаимодействие пользователей с их программами;

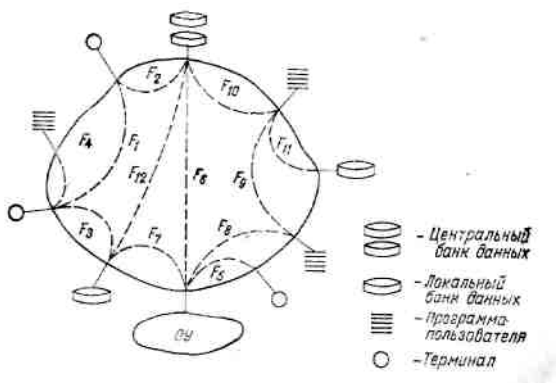


Рис. 1. Сеть ЭВМ с точки зрения пользователя: F_1 — взаимодействие различных компонентов системы: F_1 — терминал—терминал; F_2 — терминал — центральный банк данных: F_3 — терминал — локальный банк данных; F_4 — терминал — программа пользователя; F_5 — терминал — объект управления; F_6 — объект управления — центральный банк данных; F_7 — объект управления — локальный банк данных; F_8 — объект управления — программа пользователя; F_9 — взаимодействие двух пользовательских программ; F_{10}—программа пользователя — центральный банк данных; F_{11} — пользовательская программа — локальный банк данных; F_{12} — центральный банк — локальный банк; ОУ—объект управления

выполнение заданий в режиме дистанционной пакетной обработки;

выполнение программ в реальном масштабе времени;

обеспечение надежности и живучести системы в целом.

Выполнение перечисленных функций в сети можно представить схематически (рис. 1).

Естественно, что в зависимости от типа и назначения сети в ней могут отсутствовать те или иные взаимодействия. Так, F_2, F_6 и F_{10} существуют только при централизации данных, F_3, F_7 и F_{11} — при их децентрализации, а полный набор — лишь при комбинированном подходе к- организации размещения данных.

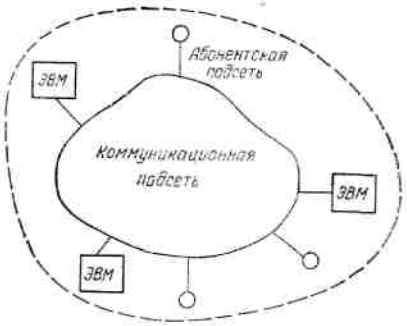


Рис. 2. Общий вид сети ЭВМ

Указанные взаимодействия могут реализовываться в одном из трех режимов или комбинаций некоторых из них: дистанционная пакетная обработка, диалоговый (интерактивный) режим и режим реального времени.

В соответствии с существующими в настоящее время представлениями в сети ЭВМ, призванной организовать реализацию рассмотренных взаимодействий, можно выделить две подсети (рис. 2): коммуникационную и абонентскую. При этом коммуникационная подсеть образуется узлами коммутации, связывающими их каналами связи и средствами передачи данных. Основной задачей коммуникационной подсети является осуществление безошибочной передачи данных. В свою очередь абонентская подсеть состоит из абонентских ЭВМ, предоставляющих вычислительный и информационный сервис сети, и из терминалов, обеспечивающих взаимодействие пользователей с сетью.

Исходя из рассмотренных требований сфер применения и основных задач абонентской и коммуникационной подсетей можно следующим образом определить основные функции сетей ЭВМ:

управление взаимодействующими пользовательскими программами (инициация, блокировка, прекращение выполнения и т. д.);

управление программами, входящими в состав программного обеспечения сети и реализующими различные виды информационных услуг;

адресацию требуемых элементов сети и определение маршрута для передачи им информации;

установление необходимых физических соединений между взаимодействующими ЭВМ, терминалами;

минимизацию ошибок передачи данных от источника к приемнику по физическим каналам;

обеспечение возможности изменения конфигурации сети и состава используемых в ней технических средств без нарушения функционирования системы в целом.

Очевидно, что создание таких сложных систем, какими являются сети ЭВМ, требует решения большого количества задач и в первую очередь задачи организации взаимодействия элементов сети между собой. В настоящее время эта задача решается с использованием концепции архитектуры сети.

В соответствии с введенными в [4] определениями под архитектурой сети ЭВМ в дальнейшем будем понимать взаимосвязь ее логической, физической и программной структур. Если считать, что рассмотренные функции сети реализуются сетевыми службами (логическими модулями в терминологии [4]), то логическая структура сети отражает состав сетевых служб и связи между ними. Исходя из функций абонентской и коммуникационной подсетей такими службами можно считать:

информационное и вычислительное обслуживание, обеспечивающее решение задач пользователей сети (см. рис. 1);

терминальную службу, обеспечивающую взаимодействие терминалов с сетью. Сюда входят преобразование форматов и кодов, управление разнотипными терминалами, отработка процедур обмена информацией между терминалами и сетью и т. д.;

коммуникационную службу, призванную решать все задачи, связанные с передачей сообщений по сети. В коммуникационной службе реализуются управление маршрутами, потоками и передачей данных, декомпозиция сообщений на пакеты и ряд других функций;

управление интерфейсами. В функции этой службы входит обеспечение взаимодействия разнотипных ЭВМ, функционирующих под управлением различных операционных систем, имеющих разную архитектуру, длину слова, форматы представления данных и др. Кроме того, служба управления интерфейсами призвана реализовывать взаимодействия ЭВМ, входящих в состав различных сетей;

административную службу сети, которая осуществляет управление сетью, реализует процедуры

реконфигурации и восстановления, собирает статистику о функционировании сети и проводит ее тестирование.

Разумеется, приведенный полный состав элементов логической структуры не является обязательным для всех реальных сетей. Так, в однородных сетях отпадает необходимость в службе управления интерфейсами, в простейших сетях может отсутствовать административная служба и т. д.

С точки зрения ранее рассмотренного разделения сети на абонентскую и коммуникационную подсети (см. рис. 2) можно говорить о том, что информационно-вычислительная и терминальная службы образуют абонентскую, а интерфейсная и коммуникационная службы — коммуникационную подсети. Административная служба не реализует непосредственно каких-либо функций связанных с сетевым обслуживанием пользователей, и должна рассматриваться как механизм обслуживания самой сети.

Распределение элементов логической структуры по различным ЭВМ задает **физическую структуру сети.** Элементами такой структуры являются вычислительные машины, связанные между собой и с терминалами. В зависимости от реализации в ЭВМ той или иной сетевой службы в физической структуре можно выделить главные ЭВМ (ГВМ), коммуникационные, интерфейсные, терминальные и, наконец, административные машины. При реализации в одной ЭВМ нескольких служб можно говорить о терминально-интерфейсных, терминально-коммуникационных и других «смешанных» элементах физической структуры. Как и в логической структуре, в физической структуре каждой конкретной сети могут отсутствовать те или иные элементы.

Программная структура сети описывает состав компонентов сетевого программного обеспечения (ПО) и связи между ними. Очевидно, что состав сетевого ПО определяется логической структурой сети, т. е. функциями, реализуемыми службами сети. В это же время связи между компонентами ПО во многом зависят от физической структуры, так как именно размещение сетевых служб на различных ЭВМ определяет необходимость реализации тех или иных межмашинных взаимодействий.

Большинство существующих и практически все создаваемые сети имеют многоуровневую иерархическую организацию сетевого ПО. Это обусловлено двумя основными причинами: необходимостью минимизации затрат на модификацию сетевого ПО при изменении состава используемого оборудования; любые осуществляемые в сети изменения не должны отражаться на пользовательских программах, использующих сетевые возможности. В свою очередь такая иерархическая организация потребовала определения правил взаимодействия программ, выполняемых в одной ЭВМ и находящихся на различных уровнях, и программ, находящихся на одном уровне, но расположенных в различных ЭВМ. Иными словами, потребовалось четкое определение интерфейсов и протоколов. **Интерфейсы** (макрокоманды, подпрограммы) зависят только от возможностей используемой операционной системы, и их стандартизация важна прежде всего с точки зрения замены ПО какого-либо уровня в каждой конкретной ЭВМ. Что же касается **протоколов**, то их роль в сети значительно важнее, и это объясняется тем, что протоколы определяют порядок обмена информацией между сетевыми объектами, т. е. их характеристики влияют на внешнюю по отношению к любой конкретной машине сети среду.

Стремление создать универсальную, открытую к изменениям физической и логической структур сетевую архитектуру привело к стандартизации числа уровней иерархии ПО и частично соответствующих протоколов [4—8].

1.2. СЕТЕВАЯ АРХИТЕКТУРА СМ ЭВМ

Сетевая архитектура однородных сетей СМ ЭВМ (САСМ) должна удовлетворять следующим основным требованиям:

быть реализуемой на стандартных операционных системах, поскольку СМ ЭВМ являются массовым серийным продуктом, поставляемым широкому кругу различных пользователей;

обеспечивать совместимость сетевого ПО для объединения в сети СМ ЭВМ, функционирующих под управлением разных операционных систем (ОС), используемых на СМ ЭВМ (ОС РВ, РАФОС, ДОС КП и др. [9]) и ориентированных на различные применения;

реализовывать в САСМ режим реального времени для использования сетей СМ ЭВМ при управлении объектами;

единая архитектура различных моделей СМ ЭВМ и их аппаратная совместимость позволяют исключить необходимость реализации интерфейсных функций, обусловленных техническими средствами (длиной слова, форматами представления данных и др.).

В результате логическая структура сети СМ. ЭВМ, учитывающая перечисленные требования, может быть представлена следующими службами (рис. 3).

Информационное и вычислительное обслуживание сети обеспечивается как пользовательскими, так и системными средствами. С точки зрения пользователя, входящие в состав данной службы системные средства определяют фактический уровень сетевого сервиса. В САСМ такие средства должны реализовывать следующие функции:

обмен файлами между удаленными носителями сети. При этом осуществляются необходимые преобразования, обусловленные различиями в файловых системах, используемых на разных ОС и в

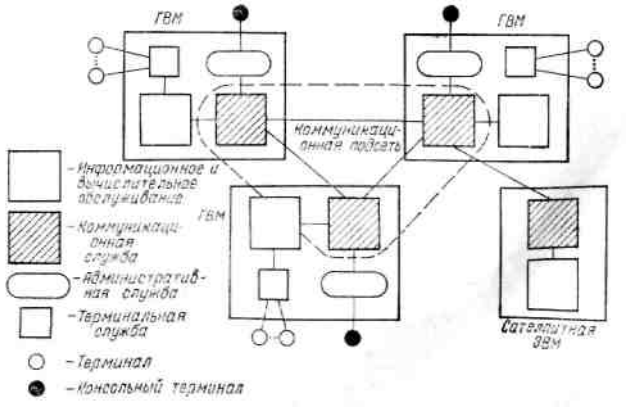


Рис. 3. Логическая и физическая структуры однородной сети СМ ЭВМ

разных системах управления базами данных даже в рамках одной и той же операционной системы (индексно-последовательные файлы, файлы прямого доступа и т. д.);

доступ к удаленным файлам с терминалов и из программ пользователей также с учетом различий в файловых организациях;

диалоговый режим взаимодействия удаленных терминалов сети;

выполнение пользовательских программ в режиме дистанционной пакетной обработки;

запуск и выполнение удаленных программ в реальном масштабе времени.

Терминальная служба однородной сети СМ ЭВМ обеспечивает управление локальными и удаленными терминалами, включая управление модемами, коммутацию требуемых каналов связи и т.д.

Коммуникационная служба реализует следующие основные функции сети:

связь элементов информационно-вычислительной службы с сетью передачи информации;

транспортировку сообщений по сети от источника к адресату;

управление маршрутами и потоками данных в сети СМ ЭВМ;

адресацию сетевых объектов;

безошибочную передачу информации;

обслуживание разнотипных коммуникационных устройств (адаптеров, мультиплексоров, модемов и др.).

Наконец, административная служба реализует тестирование как технического, так и программного обеспечения однородной сети СМ ЭВМ и осуществляет сбор статистики ее функционирования.

Основной особенностью однородных сетей СМ ЭВМ с точки зрения их физической структуры является то, что в них можно выделить лишь два типа ЭВМ — главные и сателлиты. При этом на главных машинах реализуются все рассмотренные элементы логической структуры. Это объясняется следующими основными факторами.

Комплексы СМ ЭВМ, имеющие различную конфигурацию и состав технических средств, весьма близки по цене, и, следовательно, использование каких-либо из машин в качестве коммуникационных процессоров в значительной степени повышает стоимость коммуникационной подсети по отношению к стоимости абонентской подсети. Иными словами, использование в однородных сетях СМ ЭВМ коммуникационных процессоров делает такие сети экономически нецелесообразными.

Как правило, однородные сети СМ ЭВМ имеют малую размерность, что исключает необходимость решения ряда сложнейших задач маршрутизации и управления потоками, что в свою очередь приводит к значительному уменьшению объемов памяти, требуемых для выполнения соответствующих программ, и обеспечивает возможность их размещения только в главной ЭВМ.

Сети СМ ЭВМ не имеют сетевых терминалов, все взаимодействия терминалов с сетью осуществляются через соответствующие операционные системы и пользовательские программы,

взаимодействующие с сетевым ПО, т.е. терминальная служба в этом случае также расположена в ГВМ.

Итак, в физической структуре (см. рис. 3) однородных сетей СМ ЭВМ можно выделить главные машины, реализующие функции всех сетевых служб, и сателлитные ЭВМ, на которые возлагается реализация ряда задач информационно-вычислительной службы. Кроме того, на сателлитной ЭВМ должны реализоваться некоторые функции коммуникационной службы.

Представляется целесообразным в данном разделе рассмотреть также краткие характеристики основных коммуникационных устройств из номенклатуры СМ ЭВМ, поскольку они в определенном смысле задают программную и физическую структуры САСМ.

В сетевой телеобработке СМ ЭВМ могут использоваться синхронные и асинхронные мультиплексоры и соответствующие модемы. При этом возможна работа на выделенных или коммутируемых телефонных или телеграфных каналах и через физические линии. Аппаратура передачи данных СМ ЭВМ обеспечивает возможность обмена данными в дуплексном или полудуплексном режиме.

Основные характеристики существующих и разрабатываемых коммуникационных устройств СМ ЭВМ приведены в табл. 1.

Как следует из этой таблицы, в зависимости от использования того или иного коммуникационного устройства возможны аппаратная реализация некоторых функций коммуникационной службы и, следовательно, уменьшение объема соответствующего компонента ПО сети.

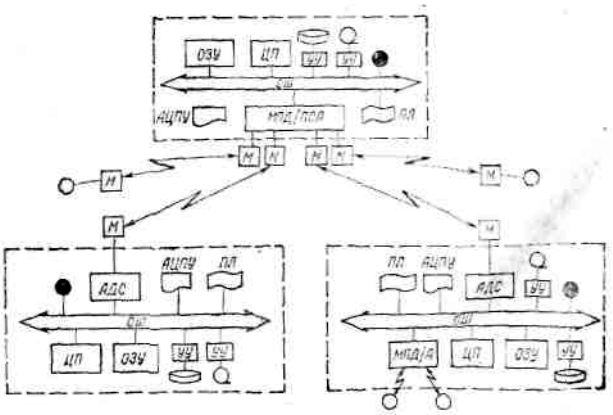


Рис. 4. Пример комплекса технических средств однородной сети СМ ЭВМ

Таблица 1

Наименование устройства	Характеристики					Примечания
	число каналов	скорость обмена	режим обмена	метод передачи	тип сопряжения	
Адаптер дистанционной связи асинхронный (АДС)	2	До 9600 бит/с	Дуплекс, полудуплекс	Асинхронный (стартстопный)	С2, ИРПС	
Адаптер дистанционной связи синхронный (АДС/С)	1	До 9600 бит/с	Симплекс, полудуплекс, дуплекс	Синхронный	С2	Реализует проверку данных по СКС-16 ПУИК
Мультиплексор асинхронный (МПД/А)	8	До 9600 бит/с	Полудуплекс, дуплекс	Асинхронный	С2, ИРПС	
Мультиплексор СМ-8514	16	До 19200 бит/с	Полудуплекс, дуплекс	Асинхронный	С2, ИРПС	
Мультиплексор синхронно-асинхронный (МПД/ПСА)	8—16	До 9600 бит/с	Полудуплекс, дуплекс	Синхронный	С2	Реализует проверку данных по CRC-16 ПУИК
Скоростной адаптер связи (АДС/Б)	1	1 Мбит/с (до 2 км), 56 Кбит/с (до 6 км), 19200 бит/с (более 6 км)	Дуплекс, полудуплекс	Синхронный	С2, коаксиальный кабель	Реализует протокол ПУИК

В целом комплекс технических средств сети показан на рис. 4. Заметим, что поскольку все перечисленные в табл. 1 устройства выходят на стандартный стык С2, то здесь предполагается использование соответствующих модемов из номенклатуры ЕС ЭВМ.

Программная структура рассматриваемых однородных сетей, как уже говорилось, определяется их

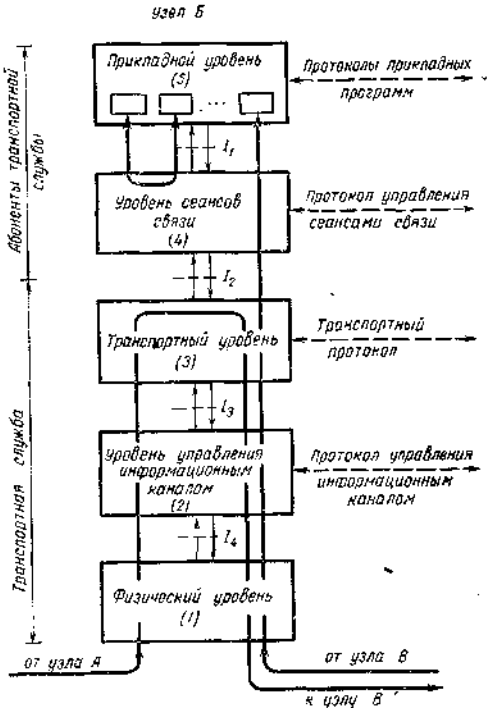


Рис. 5. Многоуровневая организация ПО сети СМ ЭВМ: I_i — межуровневый интерфейс

логической и физической структурами и характеристиками используемых технических средств и операционных систем СМ ЭВМ. В соответствии с этим была разработана пятиуровневая иерархическая организация ПО (рис. 5). Рассмотрим функции каждого из приведенных на рис. 5 уровней и соответствующие протоколы.

Физический уровень определяет электрические характеристики и сигнализацию, необходимые для установления, поддержания и разрушения физического соединения. На этом уровне обмен данными реализуется поблочно в соответствии с процедурами, определяемыми стыками С2, ИРПС и т.д.

Уровень управления информационным каналом обеспечивает безошибочную передачу информации между смежными узлами сети. На этом уровне в САСМ используется знакоориентированный протокол управления информационным каналом (ПУИК), благодаря которому достигаются независимость передачи от характеристик физического канала и используемого оборудования, оптимизация процесса передачи, проверка каналов связи, сбор статистики и т. д. Взаимодействия осуществляются за счет обмена сообщениями между соответствующими программами смежных узлов. Для выполнения такого обмена ПУИК включает три функциональные компоненты: форматирование сообщений, управление каналом, передачу сообщений.

Форматирование сообщений проводится в соответствии со схемой, допускающей использование сообщений фиксированной длины (управляющих) и сообщений переменной длины (информационных). В этом смысле ПУИК отличается от большинства применяемых знакоориентированных протоколов управления каналом тем, что не использует каких-либо специальных символов для обозначения конца сообщения. Для этих целей в заголовках соответствующих сообщений ПУИК фиксируются длины сообщений. Это дает возможность повысить достоверность передаваемых данных и организовать транспарентный режим передачи.

В протоколе применяются три типа сообщений: нумерованные информационные, ненумерованные управляющие, ненумерованные служебные. Пользовательские данные, поступающие в виде пакетов из транспортного уровня сети, передаются как информационные сообщения. Сообщения подтверждения и инициации образуют группу управляющих сообщений. Наконец, служебные сообщения используются для тестирования сети и телезагрузки операционных систем или пользовательских программ в удаленные сателлиты.

Информационное сообщение (пользовательские данные или данные другого уровня ПО сети, поступившие по интерфейсу I_3, см. рис. 5) состоит из заголовка, блока циклического контроля заголовка, поля данных и блока циклического контроля поля данных. Длина передаваемых данных задается в 14 разрядах специального поля заголовка. Следовательно, ПУИК допускает формирование сообщений с максимальной длиной 16383 байта. Пример заголовка информационного сообщения приведен на рис. 6,*а*.

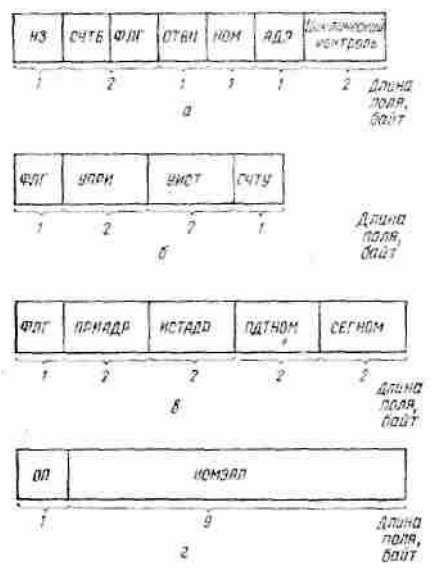


Рис 6. Типичные форматы заголовков протоколов САСМ:

НЗ— признак начала заголовка; СЧТБ - счетчик передаваемых в текущем сообщении байт; ФЛГ — флаг выбора и завершения (рис. *а*); ОТВН-но мер последнего правильно принятого донным узлом сообщения; НОМ-номер текущего сообщения, АДР-адрес узла-приемника; ФЛГ-флаг типа заголовка определяет также, что должна сеть делать с сообщением, которое по каким-либо причинам не может быть доставлено адресату (рис б); УПРИ-физический адрес узла-приемника; УИСТ - физический адрес узла-источника; СЧТУ — счетчик числа пройденных транзитных узлов; ФЛГ - идентификатор типа сообщения (рис. *в*); ПРИАДР-адрес приемника, т.е. сетевого объекта, которому передается информация по логическому каналу; ИСТАДР — адрес источника передачи- ПДТНОМ — номер последнего принятого сегмента данных и признак АСК (успешно) или NACK (с ошибкой); СЕГНОМ- номер данного сегмента данных; ОП — тип сообщения; НОМЗАП— номер записи пли аналогичный идентификатор, зависящий от типа записи

ПУИК включает пять типов управляющих сообщений: положительное подтверждение на получение информационного сообщения, отрицательное подтверждение на получение данных, запрос о состоянии сообщения по его номеру, начальную инициацию обмена, подтверждение о старте. Все управляющие сообщения имеют постоянную длину и содержат двухбайтовое поле циклического контроля. Сообщения этой группы принимаются и обрабатываются ПО уровня управления информационным каналом в соответствии с фиксированными значениями их приоритетов.

Правильность принимаемых данных в ПУИК обеспечивается за счет сравнения сумм циклического контроля (CRC), подсчитанных на передающем и принимающем узлах. При несовпадении осуществляется автоматическая повторная передача неправильно переданного сообщения, с использованием процедуры ARQ-GB(N) [10]. Суть этой процедуры состоит в том, что из одного узла может быть отправлено некоторое число сообщений без ожидания подтверждения. Если получено отрицательное подтверждение на *i*-е переданное сообщение, осуществляется повторная передача всех переданных сообщений начиная с *i* –го. ПУИК допускает посылку максимум 255 не подтвержденных сообщений. Реальное число таких сообщений определяется интенсивностью потоков в сети и значением таймаута, т.е. временем, в течение которого узел-источник ожидает подтверждения на ранее отправленное сообщение.

Очевидно, что в зависимости от тайм-аута будет передано то или иное число неподтвержденных сообщений ($\leq$255).

В целом по своим характеристикам и возможностям ПУИК близок к таким широкоизвестным протоколам канального уровня, как BSC, SLC, DDCMP и др. [11].

Транспортный уровень предназначен для транспортировки сообщений по сети от узла-источника к узлу-приемнику. При этом сообщения передаются в виде отдельных пакетов, снабженных заголовками транспортного протокола. Основой для формирования пакетов служат сегменты пользовательских данных или данных уровня сеансов связи, передаваемые по интерфейсу I_2 из программ уровня управления сеансами связи.

В целом транспортный уровень реализует следующие основные функции: маршрутизацию пакетов, управление потоками, слежение за временем пребывания пакетов в сети.

В САСМ используется принцип коммутации пакетов, реализуемой в соответствии с методом виртуального канала. При этом маршрут пакета формируется последовательно от узла к узлу, с анализом в каждом транзитном узле транспортного заголовка и определением кратчайшего пути до приемника на основании узловых маршрутных таблиц. Реализованный в каждом узле сети транспортный алгоритм маршрутизации пользуется данными двух таких таблиц (рис. 7): таблицы длин путей и таблицы стоимостей. Под длиной пути в данном случае понимается число каналов связи между смежными узлами, которые пакет должен пройти для достижения узла-приемника. Эти таблицы имеют форму матриц $L(i, j)$ — минимальная длина пути из данного узла через канал j в узел-приемник i и $C(i, j)$ — суммарная стоимость пути до узла-приемника i через канал j. При этом считается, что между двумя смежными узлами всегда существует только один соединяющий их физический канал (в том случае, если физических каналов более одного, они рассматриваются как один с соответствующими характеристиками).

Матрица стоимости задается, как правило, на основании пропускной способности, частоты появления ошибок, стоимости аренды и т. д. Однако при необходимости значения элементов этой матрицы могут отражать и такие характеристики узла, как объем памяти, доступной для буферизации данных, быстродействие процессора узла и др. При инициации узла его программы транспортного уровня определяют минимальную стоимость пути до любого узла сети. Длины и стоимости кратчайших путей

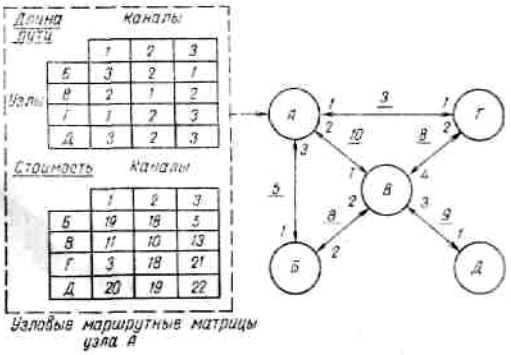


Рис. 7. Определение наилучшего маршрута пакета

сообщаются всем узлам, смежным с данным. Это делается для того, чтобы ПО транспортного уровня смежных узлов информировало соответствующие программы данного узла об изменениях в сети, требующих пересчета выбранного маршрута (выход из строя узла или канала на пути, изменение стоимости канала, ввод нового канала и т. д.).

Все элементы матриц $L(i, j)$ и $C(i, j)$ могут меняться автоматически при изменении топологии сети или могут быть изменены оператором сети.

Для управления потоками на транспортном уровне САСМ предусмотрен алгоритм, основанный на оценке состояния очередей: если число элементов очереди к каналу на передачу достигает некоторой заданной величины, то передача всех порождаемых данным узлом пакетов задерживается, а все транзитные пакеты не принимаются.

Кроме длин и стоимости путей в транспортном алгоритме используется значение задаваемой оператором максимальной длины пути. Эта величина устанавливается для каждого узла и получает значение, равное или большее максимальной длины пути от любого узла сети до данного. Так, в примере на рис. 7 в узел В можно попасть (без образования циклов) за два элементарных перехода. Максимальная длина пути используется для слежения за временем пребывания пакета в сети и имеет целью предотвращение зацикливания пакетов. Механизм функционирует следующим образом: в каждом транзитном узле для пакета на 1 увеличивается длина пройденного пути (число пройденных узлов) и сравнивается с максимальной для этого узла. Если текущая длина превышает заданную максимальную, факт зацикливания пакета очевиден.

Используемый на рассматриваемом уровне транспортный протокол (ТП) оперирует пакетами четырех форматов: информационные, содержащие данные более высоких уровней архитектуры; маршрутные, применяемые для передачи смежным узлам информации о минимальных путях из данного узла; тестовые, служащие для проверки адресуемых канала и узла; инициации, передаваемые на этапе инициации узла его соседям.

Формат типичного заголовка информационного пакета приведен на рис. 6,*б*.

С точки зрения описанной выше логической структуры сети физический уровень, уровень управления информационным каналом и транспортный образуют коммуникационную службу САСМ, реализующую коммутацию пакетов с использованием метода виртуального канала.

Уровень сеансов связи ответствен за установление, поддержание и уничтожение логических каналов между различными программами пользователей, функционирующими в сети. Данный уровень является связующим между прикладным уровнем и компонентами ПО сети, образующими коммуникационную службу, и поэтому целесообразно рассмотреть его несколько подробнее.

Основные функции, выполняемые компонентами ПО сети на уровне управления сеансами связи, заключаются в обслуживании логических каналов (ЛК):

создание ЛК;

прием и передача данных по ЛК;

прием и передача прерывающих сообщений по ЛК;

обеспечение правильной последовательности передаваемых сообщений;

уничтожение логических каналов.

Установление ЛК осуществляется на основе обмена четко определенного набора управляющих сообщений, которыми обмениваются программы уровня сеансов связи. По установленному ЛК могут передаваться три типа сообщений: информационные, содержащие пользовательские данные; прерывающие, обеспечивающие возможность изменения нормального порядка следования сообщений; управляющие. Если передаваемое информационное сообщение слишком велико для работы с ним в сети (требуются большие объемы буферной памяти, при ненадежных каналах связи велико число

повторений из-за искажений и т. д.), то оно может быть разбито на сегменты. Кроме того, небольшие объемы пользовательских данных могут передаваться и в прерывающих сообщениях.

Уровень управления сеансами связи использует два первичных интерфейса: для взаимодействия с прикладными программами (I_1 на рис. 5) и для взаимодействия с транспортным уровнем (I_2 на рис. 5). При этом через интерфейс I_1 передаются пользовательские данные в виде сообщений, а через I_2 осуществляется обмен сегментами, содержащими информацию пользователя либо сообщения ПО уровня управления сеансами связи.

На рассматриваемом уровне в САСМ используется протокол управления сеансами связи (ПУСС). Это балансный протокол с реализацией процедуры ARQ-GB(N), но в отличие от ПУИК в ПУСС максимально допустимое число переданных неподтвержденных сообщений равно 4096. На рис. *6,в* показан типичный формат заголовка сообщения ПУСС.

Прикладной уровень. На этом уровне программы принимают и интерпретируют данные из сообщений, обрабатываемых на других функциональных уровнях программной структуры САСМ. Взаимодействия таких программ осуществляются в соответствии с нестандартизованными пользовательскими протоколами. Исключение составляет протокол доступа к данным (ПДД), обеспечивающий возможность доступа к удаленным файлам. Типичный формат заголовка сообщения протокола ПДД показан на рис. 6,г. ПДД предусматривает следующие типы сообщений: конфигурационное, определяющее тип ПО узла-отправителя; сообщение-атрибут, определяющее тип и формат адресуемого файла; сообщение о режиме доступа; управляющее сообщение; продолжение

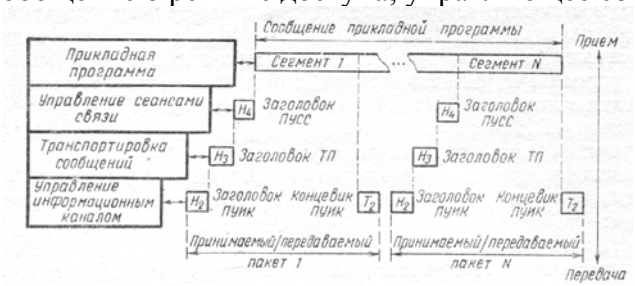


Рис. 8. Декомпозиция и сборка сообщений

передачи; подтверждение получения; завершение доступа; сообщение о состоянии. Подчеркнем тот факт, что за счет использования конфигурационных сообщений ПДД предоставляет возможность преобразования файловых форматов различных операционных систем, обеспечивая информационную совместимость в сети.

Работа различных уровней программной структуры сети СМ показана на рис. 8, где представлено форматирование передаваемого/принимаемого сообщения.

1.3. НАЗНАЧЕНИЕ, ФУНКЦИИ И СОСТАВ ПАКЕТА ПРОГРАММ СЕТЕВОЙ ТЕЛЕОБРАБОТКИ ПП СТО РВ

Как уже отмечалось, одним из основных требований, предъявляемых к САСМ, является реализуемость этой сетевой архитектуры на базе различных операционных систем СМ ЭВМ и в первую очередь на ОС РВ, ДОС КП и РАФОС. Поскольку функции сети наиболее полно реализуются в пакете программ сетевой телеобработки, функционирующем под управлением операционной системы ОС РВ (ПП СТО/РВ), в дальнейшем программное обеспечение однородной сети СМ ЭВМ рассматривается на примере именно этого пакета.

Таким образом, средства ОС РВ и ПП СТО/РВ должны обеспечивать реализацию рассмотренных возможностей САСМ. Это в свою очередь потребовало анализа характеристик ОС РВ для определения функций собственно пакета.

Основные результаты такого анализа можно сформулировать следующим образом.

1.Для одновременного взаимодействия нескольких пользовательских программ одного узла с несколькими удаленными программами в других узлах от операционной системы требуется реализация мультипрограммного режима работы. ОС РВ обеспечивает мультипрограммирование с переменным числом задач и, следовательно, удовлетворяет этому требованию.

2.Одновременная работа многих пользователей с ресурсами сети (файлами, программами) возможна при наличии средств защиты информации от несанкционированного доступа. В рамках ОС РВ существуют такие средства, обеспечивающие возможность использования кодов защиты и

идентификации пользователей. Следовательно, на каждой ЭВМ, входящей в состав рассматриваемой однородной сети, защита данных реализована.

3. Поскольку в сети, работающей под управлением ПП СТО/РВ, отсутствуют сетевые терминалы, то необходима реализация многотерминальной поддержки на каждом из узлов для предоставления пользователям возможностей взаимодействия с сетью. Базовая операционная система сети ОС РВ обеспечивает многотерминальную работу с локальными и удаленными терминалами.

4. Требуемая в сетях реального времени диспетчеризация программ в ОС РВ реализована как в виде приоритетной диспетчеризации, так и в виде диспетчеризации с квантованием времени. При этом предоставляется возможность выгрузки низкоприоритетных программ с последующим их восстановлением (свопинг).

5. Необходимый для реализации режима дистанционной пакетной обработки язык управления заданиями в ОС РВ представлен так называемыми косвенными командными файлами, предоставляющими существенно большие возможности, чем широкоизвестные языки управления заданиями ОС ЕС и ДОС ЕС.

6. Важной с точки зрения организации сетевой телеобработки характеристикой операционной системы является также ее архитектура ввода-вывода. При этом интерес представляют в первую очередь возможности включения программ обслуживания новых устройств и работа с такими устройствами как на физическом, так и на логическом уровнях без перегенерации системы. В этом смысле можно говорить об ОС РВ как об «открытой» системе, поскольку физический и логический уровни ввода-вывода в ней реализуются программами-драйверами и вспомогательными управляющими процессорами (АСР), которые могут быть написаны и легко включены в систему без ее перегенерации.

Рассмотренные ранее задачи логических сетевых служб САСМ с учетом приведенных характеристик позволяют следующим образом сформулировать назначение ПП СТО/РВ: пакет предназначен для организации однородных сетей на базе ЭВМ СМ-3, СМ-4 и вновь разрабатываемых моделей, функционирующих под управлением дисковой операционной системы реального времени ОС РВ. Пакет может использоваться также для организации взаимодействий главных ЭВМ с сателлитами, реализуемыми на микро-ЭВМ серии СМ или на ЭВМ «Электроника-60». На сателлитах ПП СТО/РВ предусматривает применение бездисковой версии ОС РВ — малой операционной системы реального времени (МОС РВ).

При этом ПП СТО/РВ реализует следующие основные функции:

а). для решения задач коммуникационной службы сети программы пакета осуществляют: управление коммуникационными устройствами (адаптерами, мультиплексорами, модемами и т. д.); обработку сообщений в соответствии с процедурами протокола ПУИК и обеспечение безошибочной передачи данных; одновременное обслуживание нескольких каналов связи; маршрутизацию и управление потоками сегментов сообщений в соответствии с используемым транспортным протоколом;

б). для связи между прикладными программами и коммуникационной службой реализуются: взаимодействия «коммуникационная сеть — программа пользователя»; одновременное обслуживание многих логических каналов с отработкой протокола ПУСС;

в). решение задач информационно-вычислительной службы включает: организацию взаимодействия различных терминалов сети; доступ с терминалов и из программ к удаленным файлам в соответствии с протоколом ПДД; пересылку файлов между удаленными узлами; передачу пакетов заданий на удаленные узлы для выполнения; запуск, в том числе в реальном времени, программ в удаленных узлах и управление этими программами как с терминалов, так и из других программ;

г). с точки зрения административной службы сети ПП СТО/РВ реализует: инициацию узлов и линий; управление узлами и линиями при функционировании сети; отображение состояния узлов и линий; накопление статистики использования линий и возникающих в них искажений; тестирование технических средств и программного обеспечения сети;

д). для связи главных ЭВМ с сателлитами пакет включает средства, осуществляющие: загрузку МОС РВ с носителя ГВМ в сателлит; загрузку программ пользователей из ГВМ в сателлит и управление ими; доступ к ресурсам ГВМ из программ, выполняемых в сателлитных ЭВМ.

Кроме перечисленных функций, связанных непосредственно с функционированием сети, средства пакета обеспечивают дополнительно генерацию ПП СТО/РВ под конкретную конфигурацию узла и используемую версию операционной системы.

В зависимости от реализуемых функций компоненты по сети СМ ЭВМ могут быть разделены на 4

группы (рис. 9). Рассмотрим состав таких групп и основные функции входящих в них программ, используя понятия процесс, сетевая программа, обслуживающая программа и прикладная программа. Здесь под процессом понимается элемент пакета, оформленный с точки зрения взаимодействия с ОС РВ как резидентное в памяти перекрытие супервизора операционной системы. Иными словами, процесс реализует некоторые дополнительные функции супервизора, являясь в определенном смысле его расширением. Под сетевыми программами понимаются компоненты ПП СТО/РВ, используемые на любых уровнях САСМ, не являющиеся процессами и не принимающими участия в реализации функций информационно-вычислительной службы. Обслуживающей программой будем называть

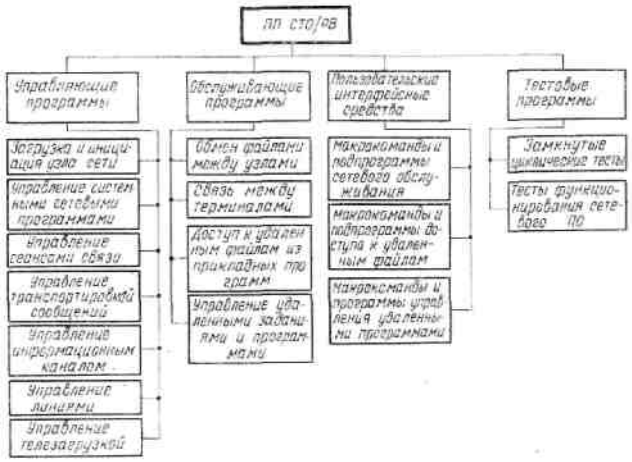


Рис. 9. Состав и структура ПП СТО/РВ

программу, входящую в состав пакета, выполняемую на прикладном уровне и реализующую какие-либо функции информационно-вычислительной службы. Наконец, прикладная программа выполняется на прикладном уровне и не входит в состав пакета.

Генерация ПП СТО/РВ осуществляется в режиме диалога с оператором и состоит из двух этапов: планирования и построения. На первом из них в соответствии с ответами оператора на вопросы системы планируется собственно генерация, т.е. определяются характеристики требуемых компонент пакета, его состав, вычисляются объемы оперативной памяти, занимаемой программами пакета, и др. Результатом выполнения этапа планирования являются таблицы, используемые на втором этапе генерации.

На этапе построения осуществляются трансляция некоторых элементов пакета и последующая компоновка всех программ, входящих в создаваемую версию ПП СТО/РВ в виде процессов, сетевых и обслуживающих программ. На этом этапе создается специальная таблица СЕТАВ, содержащая информацию о начальном состоянии узла, его линий и процессов, адреса и векторы коммуникационных устройств, режимы обмена и др.

Основные компоненты ПП СТО/РВ, их место в пакете и выполняемые функции приведены в табл. 2. Рассмотрим основные из этих компонент подробнее.

Изменение параметров пакета может быть осуществлено программой виртуальной связи с оператором (VNP) или программой изменения файла конфигурации (CFE). При этом VNP реализует изменения начального состояния узла и его процессов в дисковом файле, содержащем образ резидентной части пакета, загружаемой при установке узла. С помощью программы CFE можно изменить имя узла, адреса и векторы устройств, размеры буферов ввода-вывода, определяющих длину передаваемых в сети сегментов, и т.д. В отличие от VNP программа CFE вносит изменения в динамически используемую таблицу СЕТАВ.

Загрузка и инициация узла сети осуществляются сетевыми программами NCP (сетевая программа связи с оператором), NTINIT (сетевой инициализатор) и NTL (сетевой загрузчик). NCP служит для ввода и обработки соответствующих команд оператора и загрузки в память резидентной части пакета. В свою очередь NTINIT инициирует начальное состояние линий и коммуникационного оборудования узла, а NTL реализует загрузку в память требуемых процессов, С помощью NCP может быть получена справочная информация об узле, и, кроме того, по запросу оператора с терминала может быть изменено начальное состояние узла, какой-либо из его линий или выгружен указанный процесс.

Таблица 2

Имя компонента	Статус	Уровень в САСМ	Служба сети	Функции
1	2	3	4	5
DDM	Процесс	Физический	Коммуникационная	Управление адаптерами связи и мультиплексорами
AUX	»	»	»	Управление модемами и буферами
DLX	»	»	»	Дополнительные функции управления линиями и управление телезагрузкой
DCP	»	»	»	Реализация управления информационным каналом в соответствии с протоколом ПУИК
ROUT	»	Транспортный	»	Маршрутизация и управление потоками
NSP	»	Управление сеансами связи	»	Управление логическими каналами в соответствии с протоколом ПУСС
CEX	»		»	Управление сетевыми процессами
NTL	Сетевая программа	Прикладной	Административная	Начальная загрузка процессов сети
NTINIT	То же	»	»	Инициация узла сети
NCP	»	»	»	Управление сетью
NICE	»	»	»	Тестирование сети и сбор статистики
DSR DTR	»	»	»	Тестирование программного обеспечения сети
NETACP	Процесс	Управление сеансами связи	Коммуникационная	Обеспечение взаимодействий между прикладными и обслуживающими программами и сетью
HLD SLD	Сетевая программа	Прикладной	Коммуникационная	Управление сателлитными ЭВМ
NFT	Обслуживающая программа	»	Информационно-вычислительная	Обмен файлами между носителями удаленных узлов, передача косвенных командных файлов
FAL	То же	»	То же	Доступ к удаленным файлам из программ
MCM	Сетевая программа	»	»	Организация дистанционной пакетной обработки
ANT RNT CNT	Сетевые программы	»	»	Управление удаленными программами
TLK LSN	Обслуживающие программы	»	»	Организация терминальных взаимодействий
TCL	Сетевая программа	»	»	Управление удаленными программами

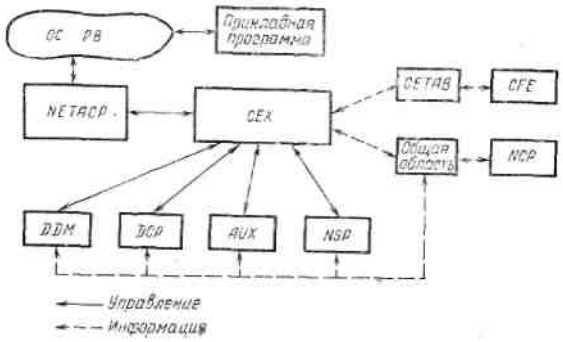


Рис. 10. Упрощенная схема взаимодействий сетевых процессов ПП СТО/РВ

Управление процессами и сетевыми программами осуществляет супервизор сети (CEX). Он планирует выполнение этих программ и процессов, организует их взаимодействия и управляет сетевым таймером и буферным пулом. С точки зрения ОС РВ CEX состоит из разделяемой системной библиотеки и общей области данных, содержащей всю необходимую для функционирования процессов сети информацию (рис. 10).

Присутствующий в каждом узле сети процесс AUX осуществляет **управление линиями передачи.** AUX является вспомогательным процессом для CEX и предоставляет средства для управления

сетевыми буферами и модемами линий передачи. Дополнительное управление линиями, необходимое для реализации телезагрузки в сателлитные машины как МОС РВ, так и прикладных программ, осуществляется процессом DLX.

Процесс DCP осуществляет отработку протокола ПУИК для передаваемых и принимаемых сообщений. Пакет ПП СТО/РВ предусматривает возможность использования телекоммуникационных устройств, микропрограммно или аппаратно реализующих ПУИК, и в этом случае DCP не будет включен в состав пакета. В рассматриваемой версии ПП СТО/РВ максимальная длина сообщения ПУИК составляет 1024 байта.

Управление функционированием телекоммуникационных устройств узла осуществляется соответствующими драйверами (DDM). При определении конфигурации узла на этапе генерации пакета для каждого из используемых типов устройств включается соответствующий драйвер.

Управление транспортировкой сообщений и управление потоками реализует процесс ROUT, отрабатывающий транспортный протокол CACM. Кроме того, ROUT следит за состоянием узловых маршрутных таблиц.

Основной функцией процесса NSP является управление сеансами связи. При этом создаются и обслуживаются логические каналы, очереди сетевых данных и т. д.

Реализация взаимодействий программ прикладного уровня с коммуникационной подсетью возложена на процессор NETACP, оформленный как вспомогательный управляющий процессор (см. рис. 10). NETACP осуществляет обработку всех макрокоманд интерфейса I_1, которые являются единственным средством организации сетевого обслуживания для прикладных и обслуживающих программ.

Управление телезагрузкой МОС РВ и пользовательских программ реализуется сетевыми программами: ГВМ-загрузчик (HLD) и сателлитный загрузчик (SLD) при участии процесса DLX.

На основе взаимодействия обслуживающих программ ПП СТО/РВ, выполняющихся в различных узлах сети, осуществляются следующие основные **функции информационно-вычислительной службы:**

обмен файлами между удаленными узлами по запросу оператора с помощью программы NFT. Эта программа обеспечивает возможность передачи файлов с любого носителя одного узла на любой носитель другого узла, выполняя при этом необходимые преобразования, обусловленные различиями в файловых организациях. Кроме того, программа NFT реализует защиту данных от несанкционированного доступа;

доступ к удаленным файлам из прикладных программ с использованием специальных интерфейсных средств, в основе которых лежат подпрограммы, реализующие обмен с удаленной сетевой программой FAL в рамках протокола ПДД.

Дистанционная пакетная обработка в ПП СТО/РВ реализуется за счет передачи в удаленные узлы косвенных командных файлов, представляющих собой программы, написанные на специальном языке управления заданиями. Переданный в удаленный узел косвенный командный файл может быть запущен на выполнение в любое время. Первая из указанных функций выполняется программой NFT, вторая — MCM.

Запуск и управление одиночными программами в удаленных узлах осуществляются за счет взаимодействия пользовательских программ или сетевых программ ANT, RNT и CNT с программой управления удаленными программами TCL. При этом используется специальный нестандартизованный в CACM протокол взаимодействия.

Терминальные взаимодействия в ПП СТО/РВ реализуются обслуживающими программами TLK и LSN, выполняемыми в различных узлах сети. Программы TLK. и LSN обеспечивают два режима взаимодействий: посылка одиночных сообщений и диалоговый режим обмена информацией с терминалом удаленного узла.

Тестирование сети. ПП СТО/РВ выполняет ряд функций по проверке функционирования сети и накоплению соответствующей статистики. Так, средствами сетевой программы «Администратор сети» (NICE) реализуются замкнутые циклические тесты трех видов: тест узла, тест узла и линии, тест двух смежных узлов.

Специальные сетевые программы DTR и DSR выполняют тестирование основных программных компонент узла сети, организованной на базе ПП СТО/РВ. Накапливаемая при этом информация позволяет оценить временные задержки, вносимые ПО сети, использование буферной памяти узлов, вероятность искажения информации в линии в зависимости от различных характеристик сети и т. д.

Такие данные дают возможность создания версий ПП СТО/РВ, наиболее эффективных для конкретных характеристик каналов связи, используемого коммуникационного оборудования, интенсивностей потоков сообщений и пр.

Как следует из рассмотренных функций и возможностей компонентов пакета, ПП СТО/РВ в построенных с его применением сетях СМ ЭВМ реализуются следующие из приведенных на рис. 1 взаимодействия: обмен информацией между терминалами различных узлов сети (F_1) обеспечивается обслуживающими программами пакета; доступ к центральному и локальным банкам данных сети с терминалов (F_2 и F_3), осуществляемый либо специальной обслуживающей программой, либо пользовательскими программами, содержащими средства интерфейса I_1 (см. рис. 5); взаимодействия типа терминал — программа пользователя (F_4) допускаются при использовании макрокоманд (подпрограмм) интерфейса I_1; аналогичным образом реализуется обмен данными между прикладными программами, выполняемыми в различных узлах сети (F_9); на базе функции F_9 могут быть осуществлены связи банков данных с программами пользователей (F_{10}, F_{11}), банков между собой (F_{12}) и с объектом управления (F_6 и F_7). При реализации взаимодействий F_{l0}—F_{12} возможно использование средств интерфейса с обслуживающей программой FAL, отрабатывающей протокол ПДД; наконец, автоматическое (F_8) и ручное (F_5) управление объектом должно быть реализовано пользовательскими программами.

ГЛАВА II

ВЗАИМОДЕЙСТВИЯ УДАЛЕННЫХ ПРОГРАММ

2.1. КОНЦЕПЦИИ СВЯЗИ МЕЖДУ ПРОГРАММАМИ

Как уже отмечалось в предыдущей главе, прикладные и обслуживающие программы для организации взаимодействия с сетью на базе ПП СТО/РВ используют единый интерфейс I_1 (см. рис. 5), представляющий собой набор макрокоманд для программ на макроассемблере ОС РВ и набор подпрограмм для программ на Фортране. В соответствии с изложенными концепциями CACM две программы, выполняемые на прикладном уровне в различных узлах сети, связываются друг с другом посредством ПУСС, отрабатываемого процессом NSP.

Рассмотрим организацию взаимодействий двух прикладных или обслуживающих программ, уделяя особое внимание основным принципам таких взаимодействий, реализуемых посредством ПУСС с использованием средств интерфейса I_1.

Процесс NSP, получая от прикладной или обслуживающей программы соответствующую информацию (через макрокоманду или подпрограмму), обеспечивает:

объявление данной программы как активной в сети, способной связываться с другими активными программами;

обработку запроса на установку логического канала между данной и удаленной программами;

принятие или отвержение запроса на установление логического канала от удаленной программы;

организацию отправления и получения информационных сообщений между программами;

отсоединение или уничтожение логических каналов и завершение активности программ в сети.

Поскольку основными функциями уровня управления сеансами связи являются установление и поддержание логических каналов, представляется целесообразным подробно определить это понятие.

Логический канал (ЛК) представляет собой последовательный " дуплексный коммуникационный путь, предназначенный для соединения сетевых объектов и обеспечивающий безошибочную передачу данных между ними. После того как NSP создаст логический канал, соответствующие программы могут отправлять и получать данные по нему. Отметим, что в ПП СТО/РВ используется механизм логических номеров, реализованный в базовой операционной системе ОС РВ. Суть этого механизма состоит в том, что все операции ввода-вывода, осуществляемые программами, адресуются не на физические адреса устройств, а на назначенные этим устройством логические номера. С точки зрения обмена информации по логическим каналам программы, являющиеся активными в сети, также адресуют не логический канал, а назначенный ему логический номер. Применение механизма логических назначений обеспечивает большую инвариантность программ по отношению к используемым в сети устройствам.

Любая программа может иметь один или более установленных логических каналов с одной или несколькими удаленными программами в одном и том же или в разных узлах сети. При этом ПП

СТО/РВ обеспечивает, возможность одновременного приема и передачи информации по разным логическим каналам для одной и той же программы.

Рассмотрим этапы установления связи между программой *L* в местном узле и программой *R* в удаленном узле (в скобках приведена мнемоника соответствующих макрокоманд интерфейса I_1,).

1. Программа *L* объявляет себя активной в сети и для нее строится очередь сетевых данных (OPN¤).
2. Программа *L* выдает запрос на установление ЛК (инициацию сеанса связи) с удаленной программой *R* (CON¤)
3. Программа *R* получает запрос на связь из своей очереди сетевых данных (GND¤) и либо принимает (ACC¤), либо отвергает (REJ¤) его.
4. Если *R* принимает запрос на установление ЛК, то открывается сеанс связи. После этого любая из взаимодействующих программ может отправлять (SND¤) или получать (REC¤) данные по логическому каналу.
5. Сеанс связи может быть окончен по инициативе любой из программ (макровызовами DSC ¤ или ABT ¤).
6. Прекратить сетевую активность программы (CLSP¤).

Подробно макрокоманды сетевого обслуживания и соответствующие подпрограммы, образующие интерфейс I_1, рассматриваются в последующих разделах данной главы. Здесь отметим только, что макрокомандам, заканчивающимся символом ¤, соответствуют подпрограммы, оканчивающиеся символами NT. В последующем рассмотрении будем использовать мнемонику средств интерфейса I_1 с окончанием x, подразумевая при этом как макрокоманды, так и соответствующие подпрограммы (например; OPNx эквивалентно макрокоманде OPN ¤ или подпрограмме OPNNT). Эти средства будем называть вызовами.

В несколько упрощенном виде организация взаимодействия удаленных программ показана на рис. 11.

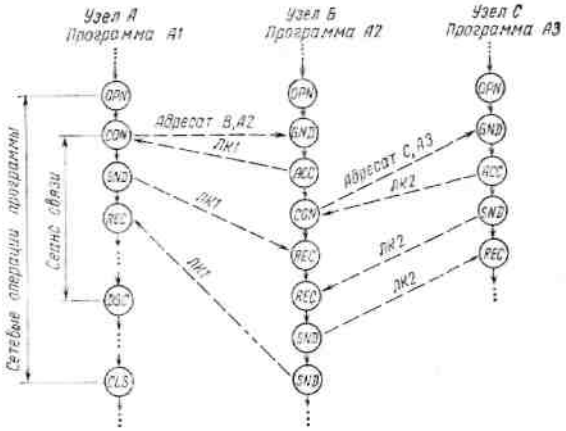


Рис. 11. Организация взаимодействий удаленных программ

Обращение к сетевым средствам связи между программами.

Первым должен быть введен сетевой вызов OPNx. При этом NSP определяет программу как активную в сети и создает для нее очередь сетевых данных (ОСД). Очереди ОСД назначается логический номер, заданный в вызове OPNx. В последующем все действия с ОСД осуществляются с использованием этого логического номера.

В ОСД содержатся незатребованные сообщения, поступающие к программе в то время, когда она является активной в сети. Такие сообщения автоматически заносятся в ОСД программы и должны быть явно извлечены ею с помощью вызова GNDx. К незатребованным относятся следующие сообщения (в скобках указаны порождающие их вызовы): запрос на установление логического канала (CONx); прерывающее сообщение, позволяющее изменить порядок передачи данных (XMIx); отсоединение «ли уничтожение ЛК пользовательской программой удаленного узла (DSCx или ABTx); уничтожение ЛК по инициативе сети, т. е. сообщение о том, что NSP уничтожила логический канал.

Следующим сетевым действием программы является установление ЛК с теми удаленными программами, обмен данными с которыми необходим.

Для описания того, как создается логический канал, назовем взаимодействующие программы «отправителем» и «получателем». Отправителем называется программа, запрашивающая соединение, а получателем — программа, принимающая или отвергающая этот запрос. Это разделение является

временным и относится только к процессу установления связи. После установления логического канала взаимодействующие программы становятся равноправными, каждая из них может отправлять и получать данные.

Программа-отправитель выдает CONx-запрос к программе-получателю на установление логического канала. Вызов CONx задается для каждого логического канала, который надо создать. Указанный в этом вызове логический номер устройства назначается логическому каналу. Данный логический номер будет использоваться в программе-отправителе для посылки (SNDx) и получения (RECx) данных, для отправки сообщений прерываний (XMIx) и для отсоединения (DSCx) или уничтожения (ABTx) логического канала. Необходимая для NSP отправителя и получателя информация при этом задается в специальном блоке связи, адресуемом в вызове CONx. Такой блок должен быть построен до выдачи запроса на установление ЛК.

NSP узла-получателя принимает запрос на установку логического канала от NSP узла-отправителя. При этом если программа-получатель активна, то NSP ее узла помещает запрос в очередь сетевых данных этой программы. В противном случае инициируется запуск требуемой программы и после выдачи в ней вызова OPNx создается ее ОСД, куда помещается запрос.

С помощью вызова GNDx запрос на установку связи извлекается из ОСД программой-получателем. В GNDx используется логический номер устройства, заданный в OPNx программы-получателя. После получения из ОСД запроса на установление ЛК программа-получатель в зависимости от различных условий может принять (ACCx) или отвергнуть (REJx) этот запрос.

ACCx выдается, если программа-получатель принимает запрос на установку связи. В качестве одного из параметров вызова ACCx задается логический номер устройства, сопоставляемый устанавливаемому логическому каналу. Этот номер будет в дальнейшем использоваться программой-получателем для посылки (SNDx) и получения (RECx) данных, для отправления сообщений прерывания (XMIx) и отсоединения (DSCx) или уничтожения (ABTx) ЛК.

REJx выдается если запрос на установку связи отвергается. В этом случае используется логический номер устройства, назначенный в вызове OPNx программы-получателя.

Обмен данными реализуется по установленному логическому каналу. При этом взаимодействующие программы становятся равноправными, т. е. разделение их на программу-отправитель и программу-получатель после установления ЛК не имеет смысла. Передача информации по ЛК обеспечивается за счет использования вызовов отправления/получения данных и посылки прерывающих сообщений.

Отсоединение или уничтожение логического канала осуществляется путем следующих вызовов:

DSCx — этот вызов отсоединяет канал. Отсоединение происходит как часть нормального потока данных по каналу (т. е. после завершения незаконченных передач);

ABTx — этот вызов отсоединяет канал и немедленно прекращает все незаконченные передачи.

Любая из взаимодействующих программ может выдать DSCx или ABTx. Если DSCx или ABTx выдается в программе-отправителе, то задается логический номер устройства, указанный в вызове CONx. Если вызов выдается в программе-получателе, то указывается логический номер устройства, определенный в вызове ACCx. После отсоединения или уничтожения логического канала назначенный ему логический номер устройства освобождается и может использоваться в последующих вызовах CONx.

Отказ от сетевого обслуживания, приводящий к окончанию сетевой активности программы, реализуется вызовом CLSx. В нем задается номер того же логического устройства, что и в OPNx. CLSx уничтожает все относящиеся к программе установленные логические каналы и освобождает все связанные с ними номера логических устройств. CLSx также уничтожает любые сообщения, находящиеся в ОСД.

Выбор незатребованных сообщений (GNDx). NSP последовательно помещает поступающие незатребованные сообщения в очередь сетевых данных программы. При извлечении сообщения из очереди оно заносится в буфер вызова GNDx, а затем вычеркивается из очереди. Существуют два способа выборки сообщений из ОСД при выполнении GNDx.

1. Первым пришло — первым получено (принцип FIFO). В GNDx при этом указываются только адрес и длина буфера для сообщения. Вызов в этом случае извлекает сообщение, стоящее в очереди первым.
2. Выбор определенного сообщения. Указываются адрес и длина буфера, тип сообщения и логический номер устройства, определяющий, по какому логическому каналу было получено сообщение. В этом случае извлекается сообщение указанного типа и с указанным логическим номером

устройства. Из таких сообщений выбирается стоящее первым в очереди сетевых данных.

Поскольку к программе может быть присоединено несколько одновременно функционирующих активных логических каналов и все поступающие незатребованные сообщения помещаются в общую ОСД, второй способ позволяет не только выбирать из очереди сообщения по их типам (т. е. сообщения прерывания, запросы на установку связи и т. д.), но и полученные по заданным логическим каналам.

Диагностика ошибок. Каждый макровызов или подпрограмма сетевого обслуживания могут включать в качестве необязательных параметров адрес двухсловного блока состояния, содержимое которого идентифицирует правильность или ошибку завершения соответствующей сетевой операции. Поскольку в ПП СТО/РВ не существует других средств диагностики ошибок выполнения вызовов сетевого обслуживания, рекомендуется применять этот необязательный параметр.

Ожидание завершения. В ПП СТО/РВ допускается использование как синхронных, так и асинхронных форм обмена информацией в сети. Для реализации синхронных сетевых операций используется формат макровызовов с ожиданием завершения соответствующих действий. В этом случае выполнение программы приостанавливается до момента успешного или аварийного завершения затребованной сетевой операции.

Средства AST. В операционной системе ОС РВ имеется ряд средств для синхронизации вычислительных процессов. В программе, написанной на макроассемблере, могут быть использованы системные средства, обеспечивающие прерывание ее выполнения и переход на специальную подпрограмму при возникновении определенных ситуаций. Если в вызове сетевого обслуживания задана подпрограмма обработки прерывания по завершению затребованной операции (AST-подпрограмма), то она получает управление в момент окончательного завершения всех процессов, связанных с передачей данных. Такой способ синхронизации может быть заменен или дополнен анализом флага события (системная переменная, принимающая значения 0 или 1), который выставляется системой при завершении выполнения сетевого вызова.

2.2. ОБЩЕЕ ОПИСАНИЕ СРЕДСТВ СЕТЕВОГО ОБСЛУЖИВАНИЯ

Прежде чем перейти к рассмотрению функций и форматов отдельных макрокоманд и подпрограмм интерфейса I_1, представляется целесообразным привести общие соглашения по их использованию.

Форматы макровызовов. При написании программ, предусматривающих использование сетевых взаимодействий, допускаются три типа форматов макровызовов: «Построить блок параметров DPB», «Выполнить», стековая форма.

Макровызов типа «Построить блок параметров DPB» используется для статического построения блока параметров макрокоманды во время трансляции соответствующей программы. Этот тип макровызова невыполняемый. Определяющий требуемую сетевую функцию блок параметров может быть указан в качестве аргумента в макровызове типа «Выполнить» и тем самым необходимое сетевое действие будет выполняться.

Формат макровызовов типа «Построить блок параметров DPB»:

метка: xxx[W] ¤ список-параметров

где метка — символическое имя, присваиваемое адресу блока параметров DPB;

xxx — имя сетевой макрокоманды;

W —определяет ожидание завершения сетевой операции;

список-параметров — список фактических параметров макрокоманды.

Макрокоманда типа «Выполнить» ссылается на блок параметров, созданный по макрокоманде типа «Построить блок параметров DPB». При этом перед выполнением возможна модификация любых ранее заданных в блоке DPB параметров.

Формат макровызовов типа «Выполнить»:

xxx[W] ¤ E метка[, список-параметров-замены]

где метка — метка области памяти, представляющей собой DPB. Блок параметров DPB может быть построен макрокомандой типа «Построить блок параметров DPB» и модифицироваться макровызовом типа «Выполнять» либо целиком им создаваться;

список-параметров-замены — один или несколько параметров, определяющих новые значения соответствующих полей DPB.

Остальные элементы формата аналогичны элементам макровызова типа «Построить блок параметров DPB».

Макровызовы стековой формы создают в стеке программы блоки параметров DPB и, используя содержащуюся в них информацию, выполняют соответствующие сетевые операции.

Формат макровызовов стековой формы:

xxx[W] ¤ S список-параметров

Элементы формата макровызова стековой формы совпадают с элементами формата макровызовов типа «Построить DPB». Пример:

```
LABEL1:   xxx ¤          —,—,—,<—,—> ;создать блок  DPB для макро xxx ¤
LABEL2:   .BLKB          N.xxL       ;зарезервировать место для блока DPB
          xxx[W] ¤E      LABEL1      ;выполнить (возможно, с  ожиданием)
          xxx[W] ¤E      LABEL1, —,— ;выполнить с модификацией
          xxx[W] ¤E      LABEL2, —,— ;построить блок DPB и выполнить
          xxx[W] ¤S      —,—,, —     ;выполнить из стека
```

Сводный перечень используемых в ПП СТО/РВ макрокоманд приведен в табл. 3.

Форматы обращений к подпрограммам. Как уже говорилось, наряду с макрокомандами средства сетевого взаимодействия в ПП СТО/РВ представлены также набором соответствующих подпрограмм, используемых при организации взаимодействия программ, написанных на языке Фортран. Естественно, что в этих подпрограммах в итоге используются макрокоманды сетевого обслуживания и, следовательно, в такие подпрограммы передаются в качестве параметров значения, используемые как аргументы макровызовов.

Обращения к подпрограммам имеют следующий формат: CALL xxxNT[W] (список-параметров)

где xxxNT — имя подпрограммы;

W — признак ожидания завершения сетевой операции; список-параметров — список фактических параметров. Список имен подпрограмм приведен в табл. 4.

Таблица 3

Макрокоманды	Назначение
АВТГ5	Уничтожить логический канал
ACC¤	Установить логический канал
CLS¤	Закончить сетевые операции
CON¤	Запросить установление логического канала
CONB¤¤	Построить блок связи для установления логического канала
DSC¤	Отсоединить логический канал
GND¤	Получить сетевые данные
DPN¤	Получить доступ к сети
REC¤	Принять данные по логическому каналу
REJ¤	Отвергнуть запрос на установление логического канала
SND¤	Послать данные по логическому каналу
SPA¤	Назначить подпрограмму обработки прерываний по получению сетевых данных
XM1¤	Послать прерывающее сообщение

Таблица 4

Подпрограммы	Назначение
ABTNT	Уничтожить логический канал
ACCNT	Установить логический канал
BACC	Построить область управления доступом
BFMT0	Построить описатель назначения формата 0
BFMT1	Построить описатель назначения формата 1
BFMT2	Построить описатель назначения формата 2
CLSNT	Закончить сетевые операции
CONNT	Запросить установление логического канала
DSCNT	Отсоединить логический канал
GLNNT	Получить информацию о местном узле
GNDNT	Получить сетевые данные
OPNNT	Получить доступ к сети
RECNT	Принять данные по логическому каналу
REJNT	Отвергнуть запрос на установление логического канала
SNDNT	Послать данные по логическому каналу
WAITNT	Приостановить выполнение программы
XMINT	Послать прерывающее сообщение

Общие соглашения по описанию форматов макровызовов **и вызовов подпрограмм.** В целях унификации описания форматов средств сетевого обслуживания сформулируем перечень соглашений,

используемых при описании их синтаксиса.

1. Заглавные буквы означают, что данное поле формата должно быть заполнено именно так, как показано. Строчные буквы указывают обозначения аргументов, а не их значения.

2. В квадратные скобки [ ] заключаются необязательные аргументы. Аргумент, не заключенный в квадратные скобки, должен обязательно присутствовать в операторе. Квадратные скобки не входят в оператор (макровызов, вызов подпрограммы).

Пример:

```
CALL   ABTNT[W] (lun,[status],[outsize],[outmessage])
ABT[W] ¤        lun,[efn],[status],[ast]
```

Здесь обязательным аргументом является только аргумент lun.

3. Фигурные скобки указывают на то, что должен быть выбран только один из заключенных в них аргументов. Сами скобки не включаются в текст вызова.

Пример:

```
GND[W] ¤ S   [lun],[efn],[status],[ast],
             { <mail, mailen>                 }
             { <mail, mailen, mask>,NT.TYP    }
             { , ,NT.LON                      }
```

При использовании данного макровызова должен быть обязательно задан один из трех заключенных в фигурные скобки аргументов.

4. Запятые и угловые скобки включаются в текст, как показано в синтаксисе. Пропущенные аргументы должны быть указаны последовательными запятыми.

Пример:

```
ABT ¤   5, ,STATUS,AST, <OUT,OUTLEN>
```

5. Учитывая аналогичность параметров макрокоманд различных форматов, приводится только стековая форма.

6. При описании форматов используются следующие единые обозначения аргументов:

lun — логический номер устройства, адресующий логический канал или ОСД;

efn — необязательный номер флага события, устанавливаемого при завершении операции;

status — адрес необязательного блока состояния, занимающего два слова и содержащего признаки завершения при возврате из макрокоманды или подпрограммы;

ast — точка входа в написанную пользователем подпрограмму AST, которая должна быть выполнена после завершения сетевой операции;

out, outlen или outmessage, outsize — начальный адрес и длина (байт) необязательного дополнительного пользовательского сообщения (длиной 1—16 байт), посылаемого удаленной программе одновременно с выполнением сетевой операции, не связанной непосредственно с передачей данных (эти аргументы являются парными и могут быть либо оба заданы, либо опущены);

mail, mailen или mailbox, mailsz — начальный адрес и длина (байт) буфера приема данных из ОСД (почтовый ящик). Начало буфера должно находиться на границе слова;

mail — начальный адрес блока связи, требуемого для установки связи АСС¤ или выдачи отказа установить связь. Данные блока связи содержатся в буфере приема сетевых данных после выполнения макрокоманды GND ¤ или подпрограммы GNDNT;

mailen — длина блока связи. Если она не задана, то используется некоторое стандартное значение;

conbl или tgtblk — начальный адрес блока запроса на установку связи, передаваемого программе-получателю по CON ¤ или CONNT. Этот блок должен быть построен с выравниванием на границу четного байта (слова);

conblen — длина блока запроса на установку связи. Если она не указана, то используется стандартное значение;

in, inlen или inmessage, insize — начальный адрес и длина (байт) необязательного дополнительного пользовательского сообщения (длиной 1 —16 байт), которое может быть получено от удаленной программы одновременно с выполнением основной сетевой операции. Эти аргументы являются парными;

buf, buflen — начальный адрес и длина (байт) буфера данных пользователя, участвующих в сетевой операции. Аргументы являются парными.

Ряд макрокоманд и подпрограмм имеют специфические аргументы, которые приводятся в соответствующих описаниях.

2.3. ФОРМАТЫ МАКРОВЫЗОВОВ И ОБРАЩЕНИЙ К ПОДПРОГРАММАМ

1. Уничтожить логический канал (ABTx). Выдача ABTx приводит к разрыву ЛК, немедленно прекращает все незавершенные передачи, отсоединяет канал и освобождает номер соответствующего логического устройства. Дополнительно при выполнении ABTx можно передать сообщение длиной 1—16 байт программе, с которой разрывается связь.

Форматы вызова:

```
ABT[W]¤S      lun,[efn], [status], [ast],[<out,outlen>]
CALL  ABTNT [W] (lun, [status], [outsize], [outmessage])
```

Если ABTx выдается из программы, бывшей инициатором установления ЛК, используется lun, введенный в CONx, если нет, то введенный в ACCx.

2. Установить логический канал (ACCx). По вызову ACCx устанавливается логический канал между программой, выполнившей ACCx, и программой, ранее запросившей связь по макровызову CONx. Иными словами, с момента завершения выполнения ACCx начинается сеанс связи между этими программами. В ACCx можно запросить передачу удаленной программе от 1 до 16 байт пользовательской информации.

Форматы вызова:

```
ACC[W] ¤ S    lun,[efn],[status],[ast],<mail,[mailen][,out,outlen]>
CALL  ACCNT[W](lun,[status],  mailbox [,outsize,  outmessage])
```

Логический номер устройства (lun), указываемый в вызове подпрограммы, назначается логическому каналу. Этот логический номер устройства должен использоваться в последующих вызовах RECx, SNDx, XMIx, ABTx и DSCx для обращения к логическому каналу.

Данные, необходимые для установки логического канала, передаются ACCx через аргумент mailbox (или mail), в котором после выполнения GNDx содержится полученный с запросом на установку блок связи.

3. Построить область управления доступом (BACC). Подпрограмма BACC вызывается для создания области управления доступом в блоке связи, используемом в подпрограмме CONNT. Содержимое этой области определяет право на доступ в удаленном узле или процессе.

Формат вызова:

```
CALL  BACC   ([status],tgtbik, [usersz],[user], [passwdsz],
                [passwd], [accnosz], [accno])
```

Для данной подпрограммы аргумент status —это необязательная логическая переменная. При возврате из BACC ей присваивается значение «истина», если завершение успешное, и «ложь», если один из аргументов BACC не верен.

Необязательная целая переменная или константа usersz указывает длину идентификатора (описателя) пользователя. Массив байт user (от 1 до 16 элементов) содержит идентификатор пользователя. Аргументы user и usersz — парные и могут быть либо оба указаны, либо оба опущены.

Аргумент passwdsz — это целая переменная или константа, указывающая длину идентификатора. Массив байт passwd из 1—8 элементов содержит идентификатор, определяющий привилегии доступа. Аргументы passwdsz и passwd — парные.

Целая переменная или константа accnosz указывает длину учетного номера. Массив байт accno из 1—16 элементов содержит учетный номер, необходимый для контроля использованного пользователем машинного времени. Эти аргументы парные и либо оба задаются, либо оба опускаются.

4. Построить описатель назначения формата 0 (BFMT0). Подпрограмма BFMT0 вызывается для построения описателя назначения формата 0 в блоке связи, используемом подпрограммой CONNT. Описатель формата 0 используется в блоке связи, если для идентификации программы в удаленном узле требуется указание только ее типа.

Формат вызова:

```
CALL  BFMT0([status],tgtblk,[ndsz],[ndname],[objtype])
```

Аргументы status и tgtblk здесь аналогичны соответствующим аргументам подпрограммы BACC.

Необязательная целая переменная или константа ndsz указывает длину имени узла. Массив байт ndname из 1—6 элементов содержит имя адресуемого узла. Аргументы ndsz и ndname являются парными.

Целая переменная или константа objtype указывает тип программы, которой будет направлен запрос на установку связи. Тип программы — число от 0 до 255 — определяется системными соглашениями.

5. Построить описатель назначения формата 1 (BFMT1). С целью создания описателя назначения формата 1 в блоке связи вызывается подпрограмма BFMT1. Блок связи с таким описателем назначения используется для установки ЛК с такой программой в удаленном узле, которая требует указания только имени.

Формат вызова:

CALL BFMT1 ([status] .tgtblk, [ndsz], [ndname], [objtype], [namesz], [name])

Аргументы подпрограммы BFMT1 аналогичны аргументам подпрограммы BFMT0 за исключением:

namesz — необязательная целая переменная или константа, указывающая длину имени программы;

name—необязательный массив байт, длиной от 1 до 16 элементов, содержащий имя тай программы, с которой устанавливается связь. Аргументы namesz и name являются парными.

6. Построить описатель назначения формата 2 (BFMT2). Подпрограмма BFMT2 вызывается с целью создания описателя назначения формата 2 в блоке связи. Описатель назначения формата 2 используется для установки ЛК с такими программами в удаленных узлах, которые требуют указания имени, кодов группы и пользователя.

Формат вызова:

CALL BFMT2 ([status],tgtblk, [ndsz], [ndname], [objtype], [group], [user], [namesz], [name])

Подпрограмма BFMT2 имеет аргументы, аналогичные подпрограмме BFMT1, за следующим исключением:

group — необязательная целая переменная или константа, указывающая, код группы;

user — необязательная целая переменная или константа, указывающая код пользователя.

7. Закончить сетевые операции (CLSx). Выдача вызова CLSx приводит к завершению сетевых действий программы, уничтожению всех установленных в ней логических каналов и освобождению всех номеров ее сетевых логических устройств. Если выдать CLSx в то время, как данные находятся в ОСД, то произойдет следующее:

если ОСД содержит невыполненные запросы на установку связи, поступившие за время функционирования программы в сети, то она заново планируется к выполнению;

если в очереди имеются сообщения прерывания, то они отвергаются;

если ОСД содержит запрос на уничтожение ЛК, то он не обрабатывается.

Вызов CLSx имеет следующие форматы:

CLS[W] ¤ S lun,[efn],[status] [,ast]
CALL CLSNT[W] [(status)]

8. Запросить установление логического канала (CONx). Вызов CONx, выданный в программе, инициирующей сеанс связи, приводит к порождению соответствующего обмена между местной и удаленной NSP. При этом если удаленная программа не запущена на выполнение, то она запускается процессом NSP своего узла. В качестве параметров вызова CONx могут быть заданы пользовательские данные (до 16 байт) для передачи адресуемой удаленной программе.

Форматы:

CON[W] ¤ S lun,[efn],[status],[ast],<conbl,[conblen],
[out],[outlen] [,in,inlen]>
CALL CONNT[W] (lun,[status].tgtblk,[outsize],[outmessage],
[insize],[inmessage])

При успешном завершении операции установления логического канала указанный в вызове логический номер устройства (lun) следует использовать при указании логического канала в любых последующих вызовах RECx, SNDx, XMIx, ABTx и DSCx.

9. Построить блок связи для установления логического канала (CONB¤¤). Макрокоманда CONB ¤¤ используется для создания блока запроса на установку связи для макрокоманды CON¤. Действие этой макрокоманды аналогично функциям, реализуемым в подпрограммах BACC, BFMT0—BFMT2. В создаваемом блоке запроса содержатся имя узла, описатель назначения и необходимая информация для

управления доступом.

Макровызов имеет следующий формат:

CONB ¤¤ [node],[obj],[fmt,<aescrip>],[rqici],
[<pass>],[accno]

где node — имя узла, которому направляется запрос на установку связи (1—6 символов);

obj—тип программы, к которой направляется запрос на установку связи. Значение должно быть в пределах от 0 до 255;

fmt— номер формата описателя (см. аргумент descrip). Допустимые значения 0, 1 или 2;

descrip —описатель программы, с которой устанавливается связь. Этот аргумент создает поле описателя одного из трех типов:

описатель нулевого формата, требующий указания только типа программы (см. выше аргумент obj). Для описателей формата 0 аргумент descrip опускается;

описатель формата 1, требующий указания имени программы. Аргумент содержит имя удаленной программы (до 16 символов ASCII);

описатель формата 2, требующий указания группы, пользователя и имени программы;

группа — двоичный номер длиной до 2 байт, представляющий код группы. Вместе с полем пользователя это поле задает определитель, используемый для защиты во время выполнения, указания каталога для поиска и т. д. Значение кода должно быть в пределах от 0 до 65535 включительно;

пользователь — двоичное число длиной 2 байта, представляющее код пользователя. Это поле является следующим уровнем определения при использовании его совместно с полем группы. Значение кода должно быть в пределах от 0 до 65535 включительно;

имя — от 1 до 12 символов ASCII, указывающих имя требуемой программы;

данные управления доступом позволяют получить доступ к удаленному узлу или процессу;

rqid — идентификатор пользователя, от 1 до 16 символов ASCII;

pass — пароль удаленного узла или процесса, от 1 до 8 байт;

accno — учетный номер удаленного узла или процесса, от 1 до 16 символов ASCII.

10. Отсоединить логический канал (DSCx). Вызов DSCx предназначен для отсоединения логического канала, существующего между двумя взаимодействующими программами (для прекращения сеанса связи между ними). Он может быть выдан любой из таких программ. При прекращении сеанса связи передаются все ранее затребованные программой, воспользовавшейся макрокомандой DSCx данные. После выполнения всех таких запросов на передачу уничтожаются вновь поступившие запросы на прием и для каждого из них формируется код аварийного завершения. Отметим, что до окончательного завершения передач программа продолжает прием сообщений. Далее канал отсоединяется и освобождается соответствующий логический номер устройства.

В вызове DSCx можно задать отправку 1—16 байт пользовательских данных, с которой прекращается сеанс связи (аргументы out и outlen). Форматы:

DSC[W] ¤ S lun,[efn],[status],[ast] [,<out,outlen>]
CALL DSCNT [W] (lun,[status],[outsize],[outmessage])

Если DSCx выдается программой, инициировавшей сеанс связи, то в качестве lun используется логический номер устройства, назначенный в вызове CONx, в противном случае используется логический номер из вызова ACCx.

11. Получить информацию о местном узле GLNx. Вызов GLNx используется для получения информация о местном узле. Можно получить имя местного узла (6 байт); размер максимального сегмента передаваемых/принимаемых по протоколу ПУСС данных (2 байта). Если операция выполнена успешно, данные помещаются в указанный буфер, длина которого в зависимости от требуемой информации может быть равна 6 или 8 байт.

Форматы:

GLN[W]¤S lun,[efn],[status],[ast],<buf,buflen>
CALL GLNNT[W] ([status], buflen,buf)

12. Получить сетевые данные (GNDx). Данный вызов предназначен для получения информации из очереди сетевых данных программы. В такой очереди могут находиться незатребованные сообщения следующих типов: запрос на установку логического канала; сообщение прерывания; уведомление об отсоединении или уничтожении логического канала, произведенном пользовательской программой; уведомление об уничтожении логического канала сетевыми процессами.

GNDx проверяет очередь сетевых данных программы и если она пуста, то завершается с ошибкой. Формат макровызова:

GND[W] ¤S [lun],[efn],[status],[ast],
{<mail,mlen>
<mail,mlen,mask>,NT.TYP
, ,NT.LON }

Макровызов GND¤ имеет ряд специфических аргументов: mask — необязательный аргумент, используемый совместно с флагом NT.TYP. Обычно GND ¤ выбирает первое сообщение из очереди сетевых данных (по принципу первым пришел — первым обслужен). Аргумент mask дает возможность выбора того сообщения, тип которого и/или номер логического канала первым совпадает с указанным в этом аргументе значением. Если аргумент mask задан, то необходимо также задать аргументы mail, mlen и флаг NT.TYP;

NT.LON — указывает на то, что тип сообщения будет занесен в слово 0, байта 1 блока состояния, а длина — в слово 1, байта 0. Данные, однако, не заносятся в буфер mail, и сообщение не удаляется из очереди программы. Этот флаг удобно использовать для динамического распределения пространства буфера. Если NT.LON указан, аргументы mail, mlen, mask и флаг NT.TYP опускаются;

NT.TYP — флаг, указывающий на то, что запрашивается сообщение заданного типа и/или логического номера канала.

Когда макрокоманда завершается успешно, то первое сообщение из очереди помещается в буфер mail программы (см. описание аргументов mail, mlen). Первый байт нулевого слова блока состояния содержит тип переданных из очереди сетевых данных,

Вызов подпрограммы:

CALL GNDNT[W] ([status],type,[mailsz],[mailbox],
[ltonly], [immed], [typmsk])

Данная подпрограмма имеет ряд не описанных ранее аргументов:

type—целочисленная переменная, которой будет присвоен числовой код, определяющий тип сообщения в mailbox после возврата из подпрограммы GNDNT:

Код	Описание
1	Запрос на установку связи
2	Прерывающее сообщение
3	Отсоединение по инициативе пользователя
4	Уничтожение логического канала по инициативе пользователя
5	Уничтожение логического канала по инициативе сети

ltonly — логическая переменная. Если ее значение истинно, то тип сообщения указывается в аргументе type, а длина сообщения— в младшем байте status (2). Данные из очереди не удаляются и не заносятся в массив mailbox, и порядок очередности не изменяется. Таким образом, этот флаг следует использовать при желании динамически распределять буферное пространство. По умолчанию значение аргумента считается ложным;

immed — логическая переменная. Если ее значение истинно и в очереди есть данные, то подпрограмма завершится нормально. Если значение истинно, но очередь пуста, то подпрограмма завершится с кодом ошибки, указывающим на то, что в очереди нет данных;

typmsk — обычно GNDNT выбирает первое сообщение из очереди сетевых данных. Задание этого аргумента обеспечивает селективный выбор сообщения по его типу, указанному в typmsk.

13. Получить доступ к сети (OPNx). Вызов OPNx служит для активизации программы в сети и создания ее очереди сетевых данных. OPNx должен быть выдан раньше любого другого вызова для взаимодействия программ.

Формат:

OPN[W] ¤S [lun],[efn],[status],[ast] [,<links>]

Логический номер устройства lun следует применять во всех последующих вызовах GNDx, SPAx, GLNx, REJx и CLSx. Если логический номер устройства не указан, то будет использоваться стандартный номер.

Необязательный аргумент links определяет максимальное число одновременно активных в программе логических каналов. Если число активных ЛК равно значению этого аргумента, то сеть будет отвергать любой новый запрос на установку связи (максимальное значение равно 255). Этот аргумент

действителен только для приходящих запросов на установку связи и не распространяется на запросы типа CONx.

CALL OPNNT[W] ([lun],[status],[mstat],[count])

Аргумент mstat — необязательный целочисленный массив из трех элементов, в котором будет храниться текущее состояние ОСД программы. Если этот аргумент задан, то он модифицируется при каждом поступлении или извлечении данных из ОСД. Массив не должен использоваться для других целей, пока программа является активной в сети. В массиве хранятся следующие значения:

mstat (1) —число сообщений в очереди;

mstat (2) —тип данных первого сообщения;

mstat (3) —длина первого сообщения.

Максимальное число одновременно активных логических каналов указывается в аргументе count.

14. Принять данные по логическому каналу (RECx). Для получения информационных сообщений (пользовательских данных) по логическому каналу служит вызов RECx. Он резервирует буфер для принимаемых данных и ждет их поступления из сети. Форматы:

REC[W]¤S lun,[efn], [status], [ast],<buf,buflen>
CALL RECNT[W] (lun,[status] ,insize,indata)

В качестве аргумента lun задается логический номер, сопоставленный ЛК либо в вызове CONx, либо в ACCx.

15. Отвергнуть запрос на установление логического канала (REJx). В случае невозможности установить ЛК программа-получатель должна использовать вызов REJx для отказа от установки логического канала. При выдаче REJx программа-получатель может передать до 16 байт данных программе, запросившей установку связи.

Форматы:

REJ[W] ¤S lun, [efn], [status], [ast],< mail,
[mailen] [,out, outlen] >
CALL REJNT[W]([status] ,mailbox, [outsize]
[,outmessage])

Используемый для отказа блок связи задается аргументами mail или mailbox. Требуемая информация попадает в буферы, адресуемые этими аргументами при выполнении вызова GNDx.

16. Послать данные по логическому каналу (SNDx). Пользовательские сообщения по установленному ЛК посылаются за счет использования вызовов SNDx. Вызов имеет следующий формат:

SND[W]¤S lun, [efn], [status], [ast],<buf,buflen>
CALL SNDNT[W] (lun,[status], outsize,outdata)

Применяемый в SNDx логический номер устройства сопоставляется ЛК, определенному либо в CONx, либо в ACCx.

17. Назначить подпрограмму обработки прерываний по поступлению сетевых данных (SPA¤). Порождение асинхронных прерываний по поступлению данных в ОСД программы допустимо только при программировании на макроассемблере ОС РВ. Использование этого механизма обеспечивает возможность задания адреса AST-подпрограммы, на которую передается управление в момент поступления незатребованных запросов в очередь сетевых данных программы. Сообщения, поступившие в очередь сетевых данных до выдачи SPA¤, не вызывают выполнение указанной пользователем AST-подпрограммы. Счетчик числа сетевых сообщений в ОСД передается в первом слове блока состояния SPA ¤ при каждой выдаче макрокоманды.

Формат макровызова:

SPA[W]¤S lun, [efn], [status], [ast],<addr>

Логический номер устройства тот же, который был использован в макровызове OPN¤. Аргумент addr указывает начальный адрес пользовательской подпрограммы AST-обработки сетевых данных. Одно из ограничений макрокоманды SPA¤ заключается
в том, что для сообщений, поступивших в очередь сетевых данных до ввода макрокоманды, в блоке состояния передается счетчик таких сообщений. Асинхронные прерывания при этом не возникают.

Пример использования макрокоманды SPA¤ приведен на рис. 12.

```
                .TITLE  ...                     ; ТЕКСТ ОСНОВНОЙ ПРОГРАММЫ
                .IDENT  ...
                    .
                    .
                    .
                OPN¤S   ...
                    .
                    .
                    .
                SPA¤S   ... , #CMPAST,<#SPAAST>
                    .
                    .
                    .
; ТЕКСТЫ         AST-ПОДПРОГРАММ
CMPAST:         MOV     (SP)+,IOSB      ; СОХРАНИТЬ АДРЕС БЛОКА
                                        ; СОСТОЯНИЯ SPA¤
                MOV     RO,—(SP)        ; СОХРАНИТЬ СОДЕРЖИМОЕ R0
                                        ; В СТЕКЕ
                MOV     IOSB,R0         ; АДРЕС БЛОКА СОСТОЯНИЯ
                                        ; SPA¤ В R0
                CMPB    #IS.SUC,(R0)    ; НОРМАЛЬНОЕ
                                        ; ЗАВЕРШЕНИЕ SPA¤
                BNE     20¤             ; НЕТ,ВЫХОД ИЗ
                                        ; AST-ПОДПРОГРАММЫ
                MOV     2(R0),R0        : СОДЕРЖИМОЕ СЧЕТЧИКА
                                        ; ОЧЕРЕДИ В R0
                BEQ     20¤             ; ЕСЛИ СЧЕТЧИК = 0, ВЫХОД
                                        ; ИЗ AST-ПОДПРОГРАММЫ
                BR      10¤             ; ЕСЛИ СЧЕТЧИК НЕ 0,
                                        ; ПРОДОЛЖИТЬ ОБРАБОТКУ
SPAAST:         MOV     R0,—(SP)        ; СОХРАНИТЬ СОДЕРЖИМОЕ R0
                                        ; В СТЕКЕ
                MOV     #l,R0           ; УСТАНОВИТЬ ЗНАЧЕНИЕ
                                        ; СЧЕТЧИКА
10¤:            GND¤S   ... ,#GNDSB     ; ПОЛУЧИТЬ ДАННЫЕ ИЗ
                                        ; СЕТЕВОЙ
                                        ; ОЧЕРЕДИ ДАННЫХ
                BCS     20¤             ; ПЕРЕХОД ПРИ ОШИБКЕ
                                        ; (ЕСЛИ «С» = 1)
                CMPB    #IS.SUC.GNDSB   ; УСПЕШНОЕ ЗАВЕРШЕНИЕ ?
                BNE     20¤             ; НЕТ, ВЫХОД ИЗ
                                        ; AST-ПОДПРОГРАММЫ
                                        ; ОБРАБОТКА СЕТЕВЫХ ДАННЫХ
                    .
                    .
                    .
                SOB     R0, 10¤         ; ЦИКЛ ДЛЯ ВЫБОРКИ
                                        ; ВСЕХ ЭЛЕМЕНТОВ ОЧЕРЕДИ
20¤:            MOV     (SP) + ,R0      ; ВОССТАНОВЛЕНИЕ R0
                ASTX¤S                  ; ВЫХОД ИЗ
                                        ; AST-ПОДПРОГРАММЫ
                .DSABL  LSB
                    .
                    .
                    .
```

Рис. 12. Пример использования макрокоманды SPA¤ 46

```
;               ПРИМЕР ИСПОЛЬЗОВАНИЯ ДИРЕКТИВ СЕТЕВОГО
;               ОБСЛУЖИВАНИЯ В ПРОГРАММЕ НА ЯЗЫКЕ
;               МАКРОАССЕМБЛЕРА ****ПРОГРАММА SENDR ****
;               *****
;               УСТАНОВЛЕНИЕ СВЯЗИ С УДАЛЕННЫМ УЗЛОМ
;               *****
                .GLOBL MAUX,RECORD
;               НАЗНАЧИТЬ LUN 2 И 3 НА СЕТЕВОЕ УСТРОЙСТВО
COMM::          ALUN¤C 2,NS,0
                ALUN¤C 3,NS,0
;               ПОЛУЧИТЬ ДОСТУП К СЕТИ
                OPNW¤E NETBLK
;               НАЗНАЧИТЬ ПОДПРОГРАММУ ОБРАБОТКИ ПРЕРЫВАНИИ ПО
;               ПОЛУЧЕНИЮ СЕТЕВЫХ ДАННЫХ
                SPAW¤E SPABLK
;               ЗАПРОСИТЬ УСТАНОВЛЕНИЕ ЛОГИЧЕСКОГО КАНАЛА
                CONW¤E CNCBLK
;               ПЕРЕДАТЬ УЧЕТНУЮ ИНФОРМАЦИЮ
                SNDW¤E SNDBLK
;               ПОЛУЧЕНИЕ СОДЕРЖИМОГО ЗАПИСИ
                RECW¤E RECBLK
;               ВОЗВРАТ
                RETURN
```

Продолжение

```
;                   *****
;                   УПРАВЛЯЮЩИЕ БЛОКИ И ДАННЫЕ
;                   *****
;                   БЛОКИ СОСТОЯНИЯ
IOSTN:              .BLKW 2
NTSTAT:             .BLKW 2
;                   СЕТЕВЫЕ ИНФОРМАЦИОННЫЕ БЛОКИ
NETBLK:             OPN¤   2,,NTSTAT
SPABLK:             SPA¤    2,1,NTSTAT,,<AST>
CNCBLK:             CON¤    3,l,IOSTN,,<CRBLK>
SNDBLK:             SND¤    3,l,IOSTN,,<MAUX,16.>
RECBLK:             REC¤    3,1,IOSTN,,<RECORD,200.>
;                   БЛОК СВЯЗИ ДЛЯ УСТАНОВЛЕНИЯ ЛОГИЧЕСКОГО КАНАЛА
CRBLK:              CONB¤¤LOCAL,0,2,<200,200,GETR>
;                   *****
;                   ОБРАБОТКА ОШИБОЧНЫХ СИТУАЦИЙ
;                   *****
AST                         .
                            .
                            .
                    ASTX¤S
                    .END
;                   ПРОГРАММА-ПРИЕМНИК GETR
START:              ALUN¤C 2,NS,0
                    ALUN¤C 3,NS,0
;                   ПОЛУЧИТЬ ДОСТУП К СЕТИ
                    OPNW¤E NETBLK
;                    ПОЛУЧИТЬ ЗАПРОС НА УСТАНОВЛЕНИЕ ЛОГИЧЕСКОГО
;                    КАНАЛА
;                    GNDW¤E GNDBLK
                    НАЗНАЧИТЬ ПОДПРОГРАММУ ОБРАБОТКИ ПРЕРЫВАНИЙ
;                    ПО ПОЛУЧЕНИЮ СЕТЕВЫХ ДАННЫХ
                    SPAW¤E SPABLK
;                    УСТАНОВИТЬ ЛОГИЧЕСКИЙ КАНАЛ
                    ACCW¤E ACCBLK
;                    ПОЛУЧИТЬ УЧЕТНУЮ ИНФОРМАЦИЮ
                    RECW¤E RECBLK
;                    ПОСЛАТЬ ПРОГРАММЕ SENDR ЗАПИСЬ С ИНФОРМАЦИЕЙ
OUTPUT:             SNDW¤E SNDBLK
;                   ОТСОЕДИНИТЬ ЛОГИЧЕСКИЙ КАНАЛ
                    DSCW¤E SNDBLK, ,,,,<#0,#0>
;                   ЗАКОНЧИТЬ СЕТЕВЫЕ ОПЕРАЦИИ
                    CLSW¤E NETBLK
;                   ВЫХОД ИЗ ПРОГРАММЫ
                    EXIT¤S
;                   *****
;                   УПРАВЛЯЮЩИЕ БЛОКИ И ДАННЫЕ .
;                   *****
;                   БЛОКИ СОСТОЯНИЯ
IOSTN:              .BLKW   2
NTSTAT:             .BLKW   2
;                   СЕТЕВЫЕ ИНФОРМАЦИОННЫЕ БЛОКИ
NETBLK:             OPN¤    2,,NTSTAT, ,<1>
GNDBLK:             GND¤    2,1,NTSTAT, ,<DATBUF,50.>
SPABLK:             SPA¤    2, ,NTSTAT, ,<AST>
ACCBLK:             ACC¤    3,1,IOSTN, ,<DATBUF>
RECBLK:             REC¤    3,l,IOSTN, ,<MAUX,16.>
SNDBLK:             SND¤    3,l,IOSTN, ,<RECORD,200.>
;                   УЧЕТНАЯ ИНФОРМАЦИЯ
MAUX:               .BLKB 16.
;                   БУФЕР ДЛЯ ПОЛУЧЕНИЯ СЕТЕВЫХ ДАННЫХ ПО GND¤
DATBUF:             .BLKB 50.
;                   БУФЕР, СОДЕРЖАЩИЙ ПЕРЕДАВАЕМУЮ ЗАПИСЬ
RECORD:             .BLKB 200.
;                   *****
;                   ОБРАБОТКА ОШИБОЧНЫХ СИТУАЦИИ
;                   *****
AST:


                    ASTX¤S
                    .END
```

Рис. 13. Взаимодействие программ, написанных на макроассемблере

18. Приостановить выполнение программы (WAITNT). Подпрограмма WAITNT вызывается для приостановки выполнения программы до тех пор, пока не закончатся запрошенные ею сетевые операции. Выполнение программы продолжается при получении сигнала о том, что указанное в соответствующем блоке состояния событие произошло.

Формат вызова:

CALL WAITNT ([index], $status_1$,..., $status_n$)

При обращении к подпрограмме WAITNT для вызывающей программы вводится состояние ожидания. Вызывающая программа перейдет в активное состояние как только какая-либо из сетевых операций, использующих блок состояния $status_1$, внесенный в список, завершится. При этом целочисленная переменная index будет содержать номер позиции соответствующего блока состояния в списке. Если аргумент index опущен, то для выявления того, какое из событий, связанных с завершением сетевых операций, произошло, необходимо проверять отдельные блоки состояния.

```
C   EXMPLX
            INTEGER*2 MLTYP,RECSIZ,SNDSIZ,OPNLUN,CONLUN,
            MESNUM,XMITS
            INTEGER*2 IOST(2),MSTAT(3)
            BYTE ERRMES(2),TSKNAM(6),CONBLK(72),NDNAM(5)
            BYTE SNDBUF(50),RECBUF(10)
            LOGICAL*1  STAT,IMMED
C
C   ЗАДАНИЕ НАЧАЛЬНЫХ ЗНАЧЕНИИ
C
            DATA NDNAM/'R','M','O','T','E'/
            DATA TSKNAM/'R',"E','C','V','E','R'/
            IMMED = .TRUE.                  !*   УСТАНОВИТЬ IMMED В
C                                           !*   TRUE    ДЛЯ GNDNTW
                                            !*
            OPNLUN=l                        !*   LUN ДЛЯ OPNNT
            CONLUN=2                        !*   CONNT LUN ДЛЯ
C                                           !*   ЛОГИЧЕСКИХ СВЯЗЕЙ
            XMITS=20                        !*  ЧИСЛО ЗАПРОСОВ, ПОСЫ-
                                            !*   ЛАЕМЫХ В УДАЛЕННЫЙ
C                                           !*   УЗЕЛ
            SNDSIZ = 50                     !*   РАЗМЕР СООБЩЕНИИ ДЛЯ
C                                           !*   ОТПРАВКИ
            RECSIZ=10                       !*   РАЗМЕР СООБЩЕНИИ ДЛЯ
C                                           !*   ПОЛУЧЕНИЯ
C   ДОСТУП К СЕТИ
           CALL OPNNTW (OPNLUN,IOST,MSTAT)
            IF (IOST(l).NE.l)GOTO 100       !*   ЕСЛИ ОШИБКА, ВЫХОД
C
C   ПОСТРОИТЬ ОБЛАСТЬ ДАННЫХ УПРАВЛЕНИЯ ДОСТУПОМ
C   2-ГО ФОРМАТА
C
            CALL BFMT2(STAT,CONBLK,5,NDNAM, ,"200,"200.6,TSKMAM)
            IF (STAT)GOTO 10                !*   ЕСЛИ УСПЕШНО,
C                                           !*   ПРОДОЛЖИТЬ
            TYPE 200                        !*   В ПРОТИВНОМ СЛУЧАЕ
C                                           !*   ВЫДАТЬ СООБЩЕНИЕ ОБ
C                                           !*   ОШИБКЕ И ВЫЙТИ
            GOTO 90 C
C   СВЯЗАТЬ ЗАДАЧУ С УДАЛЕННЫМ УЗЛОМ
C
10           CALL CONNTW(CONLUN,IOST,CONBLK)
            IF (IOST(l).NE.l)GOTO 90        !*   ЕСЛИ ОШИБКА,
C                                           !*   ОТСОЕДИНИТЬ ОТ СЕТИ И
C                                           !*   ВЫЙТИ
C   ПРИЕМ И ПЕРЕДАЧА СООБЩЕНИЙ ОТ УДАЛЕННОГО УЗЛА
C
            DO 40 MESNUM=1,XMITS
C
C   ПОЛУЧИТЬ СООБЩЕНИЕ ОБ ОШИБКАХ ОТ УДАЛЕННОГО УЗЛА
C   С ПОМОЩЬЮ ПРЕРЫВАЮЩИХ СООБЩЕНИЙ
           IF (MSTAT(1).EQ.0)GOTO 20        !*   ЕСЛИ 0, НЕТ СООБЩЕНИЙ
            CALL GNDNTW(IOST,MLTYP,2,ERRMES,,IMMED,2)
C                                           !*   ПОЛУЧИТЬ СООБЩЕНИЕ
            IF (IOST(l).NE.l)GOTO 20        !*   ЕСЛИ НЕТ ВОЗМОЖНОСТИ
C                                           !*   ПОЛУЧИТЬ
C                                           !*   СООБЩЕНИЕ.
C                                           !*   ИГНОРИРОВАТЬ ЕГО
C           TYPE 210,ERRMES(1)              !*   ПЕЧАТЬ СООБЩЕНИЯ
```

Продолжение

```
C
C  ПЕРЕДАТЬ УВЕДОМЛЕНИЕ
C
20        CALL SNDNTW(CONLUN,IOST,SNDSIZ,SNDBUF)
          IF (IOST(l).EQ.l)GOTO 30          !*   ЕСЛИ УСПЕШНО,
                                            !*   ПРОДОЛЖИТЬ
          TYPE 210,MESNUM                   !*   В ПРОТИВНОМ СЛУЧАЕ
C                                           !*   НАПЕЧАТАТЬ СООБЩЕНИЕ
C                                           !*   ОБ ОШИБКЕ И НАЧАТЬ
          GOTO 40                           !*  ОБРАБОТКУ НОВОГО
                                            !*   СООБЩЕНИЯ
C
C  ПОЛУЧИТЬ ОТВЕТ ОТ УДАЛЕННОГО УЗЛА
C
30        CALL RECNTW(CONLUN,IOST,RECSIZ,RECBUF)
C                                           !*  ЕСЛИ УСПЕШНО,
          IF (IOST(l),EQ.l)GOTO 40          !*   ПРОДОЛЖИТЬ
C                                           !*   В ПРОТИВНОМ СЛУЧАЕ
          TYPE 210,MESNUM                   !*   НАПЕЧАТАТЬ СООБЩЕНИЕ
C                                           !*   ОБ ОШИБКЕ И
C                                           !*   ПРОДОЛЖИТЬ
40        CONTINUE                          !*   ПОСЫЛКУ СООБЩЕНИЙ
C
C  РАЗОРВАТЬ ЛОГИЧЕСКУЮ СВЯЗЬ
C         CALL DSCNTW(CONLUN,IOST)
C
C  ОКОНЧИТЬ СЕТЕВЫЕ ОПЕРАЦИИ И ВЫЙТИ
C
90        CALL CLSNTW
100       STOP 'КОНЕЦ ПРОГРАММЫ'
C
C  ОПЕРАТОРЫ FORMAT
C
200       FORMAT ('ОШИБКА ПОСТРОЕНИЯ БЛОКА СВЯЗИ')
210       FORMAT ('ОШИБКА В ЗАПРОСЕ',18)
          END
```

Рис. 14. Фортран-программа, инициирующая сеанс связи

19. Послать прерывающее сообщение (XMIx). Для отправки по логическому каналу прерывающего сообщения, поступающего в сетевую очередь данных удаленной программы, используется вызов XMIx, который можно выдать в любой момент после установки логического канала. Только одно сообщение прерывания может обрабатываться в логическом канале; в противном случае вызов завершается с ошибкой. Длина пользовательских данных, передаваемых в сообщении, не должна превышать 16 байт.

Форматы вызовов:

XMI [W]¤ S lun, [efn], [status], [ast] ,< int.intlen >
CALL XMINT[W] (lun, [status] ,intsize,intmsg)

Аргументы intlen или intsize указывают длину, а int или intmsg — адрес массива байт, содержащего передаваемое сообщение прерывания. Аргументы int и intlen , а также intsize и intmsg являются парными.

```
C  EXMPLR
          INTEGER*2   OPNLUN,MLTYP,INDEX,ACCLUN,NUMERR,NUMMES
          INTEGER*2 RECSIZ,SNDSIZ,INTSIZ
          INTEGER*2 MSTAT(3),IOST(2),IOSTl(2),IOST2(2)
    BYTE RECBUF(50),SNDDAT(10),MLBX(98),INTMES(2)
C
C  ЗАДАНИЕ НАЧАЛЬНЫХ ЗНАЧЕНИЙ
C
C         OPNLUN=l
          ACCLUN=2                          !*   ACCNT LUN ДЛЯ
C                                           !*   ЛОГИЧЕСКОЙ СВЯЗИ
          RECSIZ=50                         !*   РАЗМЕР БУФЕРА ДАННЫХ
C                                           !*   ДЛЯ ПОЛУЧЕНИЯ
          INTSIZ=2                          !*   РАЗМЕР БУФЕРА ДАННЫХ
C                                           !*   ПРЕРЫВАНИЯ ДЛЯ
C                                           !*   ПОСЫЛКИ
```

```
      NUMMES=0                    !*  ЧИСЛО ПОЛУЧЕННЫХ
C                                 !*  СООБЩЕНИИ
      SNDSIZ=10                   !*  КОЛИЧЕСТВО БАЙТ,
                                  !*  КОТОРЫЕ ДОЛЖНЫ БЫТЬ
C                                 !*  ВОЗВРАЩЕНЫ
C
C  ДОСТУП К СЕТИ
C
      CALL OPNNTW(OPNLUN,IOST,MSTAT)
      IF (IOST(l).NE.l)GOTO 100   !*  ЕСЛИ ОШИБКА, ВЫЙТИ
      IF (MSTAT(l).EQ.0)GOTO 40   !*  ЕСЛИ НЕТ
C                                 !*
10    CALL GNDNT(IOST1,MLTYP,98, MLBX)
C                                 !*  ЧТЕНИЕ ДАННЫХ СЕТИ
20    CALL WAITNT(INDEX,IOSTl,IOST2)
C                                 !*  ОЖИДАНИЕ ЗАВЕРШЕНИЯ
      IF (INDEX.EQ.2)GOTO 50      !*  ЕСЛИ INDEX = 2 ПРИЕМ
C                                 !*  ЗАКОНЧЕН
C
C  ДАННЫЕ СЕТИ ПОЛУЧЕНЫ
      IF (IOST(l).NE.l)GOTO 40    !*  ЕСЛИ GNDNT С ОШИБКОЙ,
C                                 !*  ЗАКОНЧИТЬ И ВЫЙТИ
      IF (MLTYP.GE.3)GOTO 40      !*  ЕСЛИ MLTYP> = 3,
C                                 !*  СВЯЗЬ РАЗОРВАНА
      IF (MLTYP.EQ.2)GOTO 10      !*  ЕСЛИ MLTYP=2,
C                                 !*  ПОЛУЧИЛИ
C                                 !*  ПРЕРЫВАЮЩЕЕ
C                                 !*  СООБЩЕНИЕ
C                                 !*  ВЫДАТЬ НОВЫЙ GNDNT
C
C  ПОЛУЧИЛИ ЗАПРОС НА СВЯЗЬ, ВВЕСТИ КОМАНДУ ПРИЕМА
C
      CALL ACCNTW(ACCLUN,IOST, MLBX)
      IF (IOST(l).NE.l)GOTO 1     !*  ЕСЛИ ОШИБКА, ВЫДАТЬ
C                                 !*  НОВЫЙ GNDNT
C
C  ВВЕСТИ КОМАНДУ RECNT ДЛЯ ПОЛУЧЕНИЯ ДАННЫХ
C
30    CALL RECNT (ACCLUN,LOST,RECSIZ,RECBUF)
 GOTO 10                          !*  ВЫДАТЬ НОВЫЙ GNDNT
C                                 !*  И ЖДАТЬ ЗАВЕРШЕНИЯ
C  ПОЛУЧАЕМ УПРАВЛЕНИЕ В ЭТУ ТОЧКУ, ЕСЛИ ЗАПРОС НА
C  РАЗЪЕДИНЕНИЕ ИЛИ ПРЕРЫВАНИЕ
40    CALL CLSNTW                 !*  ПРЕКРАТИТЬ ДОСТУП К
C                                 !*  СЕТИ
C                                 !*  И ВЫЙТИ
      GOTO 100
C
C  ПРИХОДИМ СЮДА, ЕСЛИ ПОЛУЧИЛИ ЗАПРОС
C
50    NUMMES = NUMMES+1           !*  УВЕЛИЧИТЬ СЧЕТЧИК
C                                 !*  ЧИСЛА СООБЩЕНИЙ
      IF (IOST2(l).EQ.l)GOTO 60   !*  ЕСЛИ IOST(l) = l, ВСЕ
C                                 !*  В ПОРЯДКЕ
C
C  ЕСЛИ БЫЛА ОШИБКА В ПРЕРЫВАЮЩЕМ СООБЩЕНИИ,
C  ПЕРЕДАЕТСЯ НОМЕР ОШИБОЧНОГО ИНФОРМАЦИОННОГО
C  СООБЩЕНИЯ
C
      1NTMES(1)=NUMMES            !*  ПОСЛАТЬ НОМЕР
C                                 !*  СООБЩЕНИЯ
      CALL XMINT(ACCLUN,IOST, INTSIZ,INTMES)
      GOTO 70                     !*  ВЫДАТЬ НОВУЮ КОМАНДУ
C                                 !*  RECNT
C
C  В ЭТОМ МЕСТЕ ПОЛЬЗОВАТЕЛЬ МОЖЕТ ОБРАБОТАТЬ ДАННЫЕ
C  В БУФЕРЕ RECBUF И ОТВЕТИТЬ, ПОМЕЩАЯ ТРЕБУЕМУЮ
C  ИНФОРМАЦИЮ В SNDDAT
C
C  ПОСЛАТЬ ДАННЫЕ И ВЫДАТЬ НОВЫЙ RECNT
C
60    CALL SNDNTW(ACCLUN,IOST,SNDSIZ,SNDDAT)
```

Продолжение

```
70      CALL RECNT(ACCLUN,IOST2,RECSIZ,RECBUF)
        GOTO 20                         !*  ОЖИДАНИЕ ЗАВЕРШЕНИЯ
C
C  КОНЕЦ ПРОГРАММЫ
C
100     STOP                            !*  ОСТАНОВИТЬ ПРОГРАММУ
C                                       !*  И ВЫЙТИ
        END
```

Рис. 15. Фортран-программа, запускаемая по запросу из удаленного узла

На рис. 13 показаны фрагменты текстов программ SENDR (источник сообщений) и GETR (приемник сообщений), а на рис. 14 и 15 — примеры двух взаимодействующих Фортран-программ. Программа EXMPLX получает доступ к сети, устанавливает связь с программой EXMPLR в удаленном узле. После завершения обмена данными с программой EXMPLR программа EXMPLX отсоединяет логический канал, закрывает доступ к сети и кончает работу.

ГЛАВА III

СЕРВИСНЫЕ СРЕДСТВА ПАКЕТА

3.1. ДОСТУП К УДАЛЕННЫМ ФАЙЛАМ

На основе взаимодействия соответствующих компонентов в ПП СТО/РВ обеспечиваются две основные возможности работы с удаленными файлами: обмен файлами между удаленными узлами сети в интерактивном режиме и доступ к удаленным файлам из пользовательских программ.

Обмен файлами. Обмен файлами между удаленными узлами в интерактивном режиме реализуется на основе взаимодействия сетевых программ передачи файлов NFT, выполняемых в разных узлах сети. Программы NFT выполняются на прикладном уровне и взаимодействуют между собой в соответствии с протоколом доступа к данным. Организация обмена файлами с использованием программ NFT показана на рис. 16.

В целом передача файлов реализуется следующим образом: в соответствии с описанной в предыдущих главах методикой, NFT узла-источника устанавливает ЛК с аналогичной программой узла, на котором расположен требуемый файл. После установления ЛК две программы NFT обмениваются сообщениями в форматах протокола ПДД. В зависимости от затребованных действий над файлом NFT удаленного узла проводит операции с ним. При этом используется стандартный механизм логического метода доступа ОС РВ. Так, при передаче файла с одного узла на другой NFT узла, на котором расположен файл, последовательно читает записи файла и передает их в сообщениях протокола ПДД удаленной NFT. Естественно, при этом соблюдаются требования файловой структуры ОС РВ и учитываются архитектурные особенности СМ ЭВМ.

Информация, необходимая для организации обмена через NFT, задается с терминала. Запуск программы NFT осуществляется в соответствии с системными соглашениями ОС РВ [9].

Средства защиты удаленного файла. Для доступа к удаленному файлу необходимо задать предварительно контрольную информацию. Это требование обеспечивает защиту файлов от не-

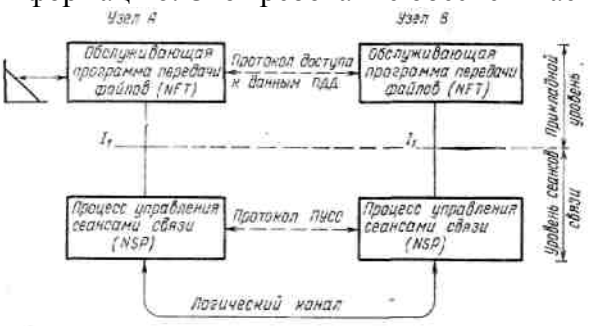


Рис. 16. Организация обмена файлами:
I_1 — интерфейс сетевого обслуживания

санкционированного доступа. Для доступа к удаленному файлу необходимо выполнить два условия:

зарегистрироваться в удаленном узле;

после ввода команды задать следующую контрольную информацию: идентификатор пользователя,

идентификатор защиты (ИЗ) и учетный номер.

Программа NFT обеспечивает передачу файлов в двух режимах:

передача одного или нескольких входных файлов из одного «ли разных узлов в один выходной файл. При указании двух или более входных файлов они сливаются в один выходной;

передача одного или нескольких входных файлов в один или несколько выходных файлов. Каждому входному файлу должен быть при этом назначен свой выходной файл. Входные и выходные файлы могут находиться в одном или разных узлах.

Командная строка для передачи файлов имеет следующий формат:

OFILEl [,OFILE2,...,OFILEN] = IFILE1 [/SW] [,IFILE2,..., IFILEN[/SW]]

где OFILE — описатели выходных файлов. Если указаны один выходной файл и несколько входных файлов, то осуществляется слияние их в один выходной; IFILE —описатели входных файлов; /SW — ключ, определяющий вид передачи файлов: /AS — передача в символьном коде КОИ-7; /IM — передача в двоичном коде. Описатель файлов в командной строке имеет следующий формат:

[имя узла_] спецификация файла,

где имя узла содержит от одного до шести символов, указывающих узел, на который адресована операция NFT. Если имя узла не задается, то введенная команда действует на файл местного узла.

Спецификация файла соответствует стандартным соглашениям ОС РВ. Имя узла разделяется от спецификации файла знаком подчеркивания.

Таблица 5

Мнемоника макровызова	Имя NFAR-подпруграмы	Функции
NFOR¤W	OPRNFW	Открыть файл для чтения
NFOW¤W	OPWNFW	Открыть файл для записи
NFOA¤W	OPANFW	Открыть файл для расширения
NFGT¤W	GETNFW	Читать запись
NFPT¤W	PUTNFW	Занести запись в файл
NFCL¤W	CLSNFW	Закрыть файл
NFDL¤W	DELNFW	Удалять файл
NPARM¤		Построить блок параметров

Доступ к удаленным файлам из программ пользователя. На прикладном уровне программам пользователя предоставляются средства доступа к удаленным файлам. Для программ на Фортране доступ обеспечивается через вызовы NFAR-подпрограмм, реализующих в рамках ПДД требуемые взаимодействия с обслуживающей программой доступа к файлам FAL в удаленном узле. Для программ на макроассемблере доступ к удаленным файлам осуществляется через вызовы макрокоманд, объединенных под общим названием NFAMAC, в которых используются обращения к соответствующим NFAR-подпрограммам.

Организация доступа к удаленным файлам из пользовательских программ показана на рис. 17. В табл. 5 приведены перечень макровызовов доступа к удаленным файлам и имен соответствующих NFAR-подпрограмм, а также их функции. В рассматриваемой версии программного обеспечения ПП СТО/РВ обеспечивается доступ только к файлам с последовательной организацией. Открытие файлов для чтения и расширения относится к существующим файлам, а открытие для записи обеспечивает создание новых файлов.

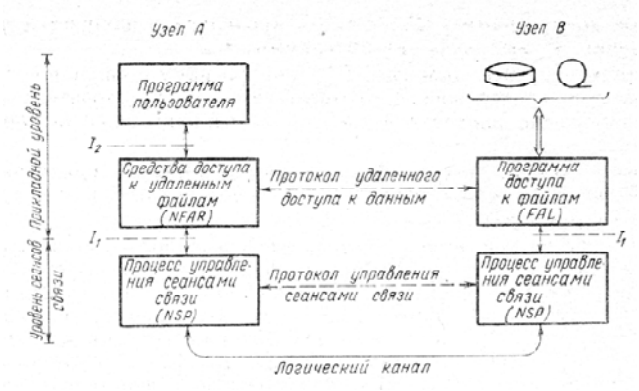


Рис. 17. Организация доступа к удаленным файлам:

I_1 — интерфейс прикладного уровня с уровнем сеансов связи;

I_2 — интерфейс доступа к файлам, реализуемый через вызовы NFAR-подпрограмм или макрокоманд

Отметим основные особенности доступа к удаленным файлам из программ на макроассемблере и Фортране.

1. Доступ к удаленным файлам на макроассемблере.

Перед использованием макрокоманд доступа к удаленным файлам необходимо с помощью макровызова NPARM¤ создать блок параметров. Блок параметров создается один раз для всех последующих использований макрокоманд группы NFAMAC и заполняется соответствующими аргументами.

Отсутствующий необязательный параметр в блоке параметров заменяется на 1.

Блок параметров может содержать до 10 однословных элементов, являющихся адресами аргументов макрокоманд.

Для доступа к удаленным файлам пользовательская программа с помощью макрокоманды сетевого обслуживания — OPN ¤ должна объявить себя активной в сети программой. При этом обязательным является задание в макровызове OPN¤ в качестве аргумента адреса AST-подпрограммы.

После завершения макрокоманды OPN ¤ во втором слове блока состояния ввода-вывода указывается число запросов в очереди сетевых данных программы.

Запрещается в пользовательской программе при работе с удаленными файлами использовать макрокоманду сетевого обслуживания SPA ¤.

2. Доступ к удаленным файлам на Фортране. Последовательность процедур для доступа к удаленным файлам из Фортран-программ включает:

вызов подпрограммы OPNNT. Эта процедура должна быть выполнена первой как для организации взаимодействия между программами, так и для сетевого доступа к удаленным файлам;

открыть файл. Файл может быть открыт для чтения, расширения и записи. Каждая NFAR-подпрограмма, соответствующая указанным режимам работы, устанавливает логический канал с программой FAL удаленного узла, в котором находится файл. Логический номер устройства, назначенный ЛК, используется в дальнейшем для всех операций с удаленным файлом;

выполнить операции чтения или записи в файле;

закрыть файл после завершения его обработки. Выполнение этой операции завершает все незаконченные действия с файлом, отсоединяет ЛК и освобождает соответствующий логический номер устройства;

вызов подпрограммы CLSNT. Эта процедура выполняется для завершения сетевых операций программы.

Примечание. Все вызовы NFAR-подпрограмм являются синхронными, т. е. управление возвращается в пользовательскую программу после завершения требуемой операции над удаленным файлом.

Прежде чем описывать макровызовы и вызовы подпрограмм на Фортране, приведем обозначения используемых в них аргументов (параметров):

lun— логический номер устройства, связанного с удаленным файлом, назначается при открытии файла и используется для всех последующих операций над файлом до его закрытая; status — двухсловный блок состояния завершения операции; node — строка ASC1Z до 6 символов, содержащая имя удаленного узла; ident — массив символов ASCIZ до 143 символов, содержащий идентификатор пользователя, учетную информацию, идентификатор защиты; ifile — строка ASCIZ, содержащая спецификацию файла в удаленном узле; inbytes—длина в байтах вводного буфера для чтения записи из удаленного файла; outbytes — длина в байтах выводного буфера записи для записи в удаленный файл; inarray (outarray) — адрес вводного (выводного) буфера записей; ichar — трехбайтовое поле, определяющее: режим обмена—ichar(1), тип записи — ichar(2), управление кареткой — ichar (3). Эти поля могут принимать соответственно следующие значения: *A* — файл содержит символьные данные; *B* — файл содержит двоичные данные; *F* — записи фиксированной длины; *V* — записи переменной длины; *T* — первый байт записи содержит символ управления кареткой; *N* — отсутствие символа управления кареткой; 1en—длина записи; iblock — поле длиной в одно слово. При открытии файла для чтения iblock — необязательный параметр, указывающий при возврате в пользовательскую программу размер файла в блоках и принимающий значения: +*n* — число несмежных блоков, распределенное для файла; —*n* — число смежных блоков, распределенное для файла. При открытии файла для записи в параметре iblock указывается требуемый размер файла в блоках, а при возврате в пользовательскую программу содержится число распределенных блоков для файла: о — динамическое распределение пространства для файла; +*n* или —*n* — соответственно число несмежных или смежных блоков, распределенных для файла. При открытии файла для расширения iblock является указателем расширения файла, принимающий одно из следующих значений: о — динамическое расширение файла, +*n*— требуемое число несмежных блоков для расширения файла:—*n* — требуемое число смежных

блоков для расширения файла.

Указанные обозначения параметров будут использоваться в дальнейшем без их пояснения. В квадратные скобки заключаются необязательные параметры.

Описание вызовов доступа к удаленным файлам.

1. Построить блок параметров. Вызов используется только в программах на макроассемблере, для выделения памяти для блока параметров. Блок параметров создается в программной секции с именем ¤ARGLT, получает глобальное имя. ARGLT и обеспечивает необходимой информацией NFAR-подпрограммы. Макровызов:

NPARM¤

2. Открыть файл для чтения. Предназначается для открытия существующего файла в режиме чтения. Чтение файла начинается с первой записи в файле. Макровызов:

NFOR¤W prmblk

Аргумент prmblk является адресом блока параметров макрокоманды, который содержит в указанной последовательности адреса 8 следующих параметров: lun, status, node, ident, ifile, ichar, len, iblock. Вызов на Фортране:

CALL OPRNFW (lun,status,node,ident, ifile.ichar.len [,iolock])

Необязательный параметр iblock (если задается) после возврата из подпрограммы указывает размер файла.

3. Открыть файл для расширения. Используется для открытия существующего файла с целью добавления записей в конец файла. Макровызов:

NFOA¤W prmblk

Аргумент prmblk полностью идентичен аргументу макровызова NFOR¤W.
Вызов на Фортране:

CALL OPANFW (lun,status,node,ident, ifile, ichar,len[,iolock])

4. Открыть файл для записи. Используется для создания последовательного файла и открытия его для записи. Макровызов:

NFOW¤W prmblk

Аргумент prmblk является адресом блока параметров макрокоманды, который содержит в указанной последовательности адреса 8 следующих параметров: lun, status, node, ident, ifile, ichar, len, iblock.

Вызов на Фортране:

CALL OPWNFW (lun.status.node.ident, ifile,ichar,len [,iblock])

5. Читать запись. Предназначается для чтения записи из последовательного файла, начиная с первой записи в файле. Файл должен предварительно быть открыт. Макровызов:

NFGT¤W prmblk

Аргумент prmblk является адресом блока параметров макрокоманды, который содержит в указанной последовательности адреса 4 следующих параметров: lun, status, inbytes, inarray. Если фактическая длина записи в удаленном файле превышает значение параметра inbytes, то часть записи (остаток) теряется.

Вызов на Фортране:

CALL GETNFW(lun,status,inbytes,inarray)

6. Занести запись в файл. Используется для занесения записи в файл, который должен быть предварительно открыт. Если файл открыт для записи, то записи заносятся последовательно с начала вновь созданного и открытого файла. При открытии файла для расширения записи добавляются в конец файла. Макровызов:

NFPT¤W prmblk

Аргумент prmblk является адресом блока параметров макрокоманды, который содержит в указанной последовательности адреса 4 следующих параметров: lun, status, outbytes, outarray. Значение параметра outbytes не должно превышать значения параметра 1еп (длина записи), задаваемого в вызовах для открытия файла, в противном случае часть записи (остаток) теряется.

Вызов на Фортране:

CALL PUTNFW(lun,status,outbytes,outarray)

7. Закрыть файл. Используется для закрытия удаленного файла после завершения работы с ним. После закрытия файла отсоединяется ЛК, используемый для доступа к удаленному файлу. Макровызов:

NFCL¤W prmblk

Аргумент prmblk является адресом блока параметров макрокоманды, который содержит в указанной последовательности адреса 2 параметров: lun, status. Вызов на Фортране:

CALL CLSNFW(lun,status)

8. Удалить файл. Используется для удаления файла. Удаляемый файл должен быть закрыт. Если удаление файла является единственной сетевой операцией в программе, то до ее выполнения необходимо предварительно выполнить OPNx, а после удаления файла — CLSx. Макровызов:

NFDL ¤ W prmblk

Аргумент prmblk служит адресом блока параметров макрокоманды, который содержит в указанной последовательности адреса 5 параметров: lun, status, node, ident, ifile.

Вызов на Фортране:

CALL DELNFW (lun,status,node,ident,ifile)

3.2. УПРАВЛЕНИЕ УДАЛЕННЫМИ ПРОГРАММАМИ

Управление удаленными программами в ПП СТО/РВ включает дистанционную пакетную обработку, управление удаленными программами с терминалов и управление удаленными программами из пользовательских программ.

Реализация режима дистанционной пакетной обработки показана на рис. 18. Пакеты заданий оформляются в виде специально оформленных текстовых файлов, именуемых косвенными командными файлами. Передача косвенных командных файлов между узлами сети осуществляется с помощью программы передачи файлов NFT, а запуск командного файла на выполнение в удаленном узле — программой управления заданиями МСМ.

В режиме дистанционной пакетной обработки обеспечиваются следующие возможности.

1. Передача командного файла в удаленный узел, выполнение его и удаление после выполнения. Команда NFT, используемая в этом режиме, имеет формат:

OFILE = IFILE/SB

где OFILE — описатель временного командного файла в удаленном узле;

IFILE — описатель передаваемого косвенного командного файла, содержащего последовательность команд для выполнения в удаленном узле;

/SB — ключ, задающий режим выполнения командного файла. После выполнения файл удаляется.

2. Выполнение командного файла, находящегося в удаленном узле. Команда NFT в этом режиме имеет формат:

NFT>OFILE/EX

где OFILE — описатель косвенного командного файла, хранящегося в удаленном узле;

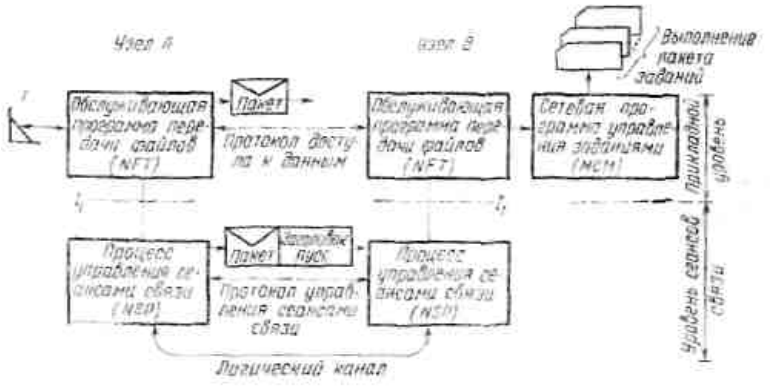


Рис. 18. Реализация режима дистанционной пакетной обработки:

I_1 — интерфейс сетевого обслуживания

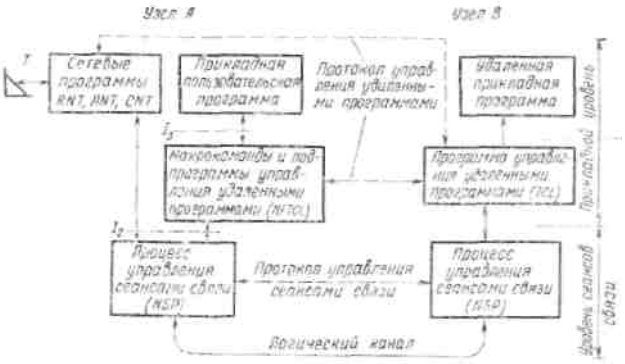


Рис. 19. Механизм управления удаленными программами

/EX — ключ режима выполнения. После выполнения файл не удаляется.

Механизм управления удаленными программами с терминалов и из программ пользователя показан на рис. 19. В этом режиме выполняются следующие функции: запуск требуемой программы в указанный момент времени; запуск программы с указанным интервалом ее перезапуска; прекращение текущего выполнения программы; прекращение периодического перезапуска. Как видно из рис. 19, эти функции могут быть затребованы пользовательскими программами через интерфейс управления удаленными программами I_3 либо с помощью команд оператора, выходящих через интерфейс сетевого обслуживания I_2 непосредственно на уровень управления сеансами связи. Управление удаленными программами в указанном узле осуществляется программой TCL.

Управление удаленными программами с терминалов обеспечивается использованием трех вспомогательных программ RNT, ANT и CNT. Эти программы реализуются в местном узле как встроенные функции программы связи с оператором ОС РВ, т. е. выполняются как дополнительные команды MCR. Эти команды могут использоваться только при наличии в удаленном узле установленной программы TCL. Рассматриваемые команды обеспечивают следующие функции по управлению удаленными программами:

RNT —запуск и планирование выполнения удаленной программы;

ANT — завершение выполнения программы;

CNT — отмена запроса на выполнение программы по таймеру.

Форматы команд:

RNT ndname psswd tsknam [ugc] [umc] [smg] [snt] [rmg] rnt]

ANT ndname psswd tsknam [ident]

CNT ndname psswd tsknam [ident]

Все аргументы разделяются пробелами или знаками табуляции. ndname — имя удаленного узла; psswd — ИЗ пользователя, определяющий его права доступа в удаленном узле. Непривилегированный ИЗ дает возможность пользователю запустить (прекратить выполнение, отменить) программу с кодом идентификации пользователя UIC, соответствующим этому ИЗ. Привилегированный ИЗ позволяет управлять программами с любым UIC; tsknam —имя программы, которая должна быть запущена; отменена и т. д.; ugc —код группы в UIC; umc— код пользователя в UIC; smg —интервал времени до запуска программы (интервал диспетчеризации); snt—,единицы измерения интервала диспетчеризации; rmg — интервал времени передиспетчеризации программы; rnt — единица измерения интервала передиспетчеризации; ident — идентификационный номер, присваиваемый программе, запушенной в удаленном узле.

Управление удаленными программами из программ на макроассемблере включает возможность запуска (одноразового и многократного по таймеру), отмену запуска по таймеру и отмену запуска с прекращением текущего выполнения удаленных программ. Соответствующая программа в удаленном узле должна быть установлена. Перечисленные функции в удаленном узле выполняются программой управления удаленными задачами TCL. Пользовательская программа на макроассемблере в местном узле взаимодействует с TCL с использованием следующих процедур.

Шаг 1. Пользовательская программа с помощью макрокоманды OPN¤ получает доступ к сети. Шаг 2. Пользовательская программа пытается установить логический канал с удаленной TCL посредством макрокоманды CON¤. При этом в макрокоманде CON ¤ в качестве необязательной информации передается запрос на запуск требуемой программы в формате (запрос RUN), приведенном в табл. 6.

Таблица 6

Поле	Длина поля, байт	Содержимое
tsknam	8	Имя удаленной программы, подлежащей запуску, при необходимости дополненное справа пробелами
ugc	1	Групповой код
umc	1	Код пользователя
smg	2	Интервал времени до запуска программы (интервал диспетчеризации)
rmg	2	Интервал времени передиспетчеризации
snt	1	Единицы измерения интервала диспетчеризации
rnt	1	Единицы измерения интервала передиспет-черизации

Таблица 7

Поле	Длина поля, байт	Содержимое
ident	2	Идентификатор, присвоенный запущенной в удаленном узле программе
ec	2	Код завершения обработки запроса на запуск удаленной программы

Шаг 3. В ответ на запрос CON¤ удаленная TCL отвечает пользовательской программе либо макрокомандой ACC¤ (запрос на запуск принят), либо REJ ¤(запрос отвергнут). В обоих случаях TCL в качестве необязательной информации передает сообщение, приведенное в табл. 7.

Шаг 4. Реакцией пользовательской программы на REJ ¤ из удаленной TCL должно быть отсоединение логического канала с TCL (макрокоманда ABT ¤).

Шаг 5. При необходимости пользовательская программа может отменить передиспетчеризацию удаленной программы, прекратить ее текущее выполнение или отменить передиспетчеризацию с одновременным прекращением текущего выполнения. Во всех случаях должен быть вновь установлен логический канал связи с TCL (макрокоманда CON¤) и в качестве необязательных данных переданы следующие сообщения:

для прекращения текущего выполнения удаленной программы запрос ABORT;

для отмены последующих запусков удаленной программы по таймеру — запрос CANCEL;

для одновременного прекращения выполнения удаленной программы и отмены ее последующих запусков по таймеру — запрос
ABORT-CANCEL.

Форматы запросов ABORT, CANCEL и CANCEL совместно с ABORT даны в табл. 8.

Шаг 6. Обработать полученную информацию от TCL. Эта информация передается в необязательных данных макрокоманд ACC¤ или REJ¤ в формате, приведенном в табл. 9.

Шаг 7. Уничтожить логический канал с TCL (макрокомандой ABT¤) и при необходимости отсоединиться от сети (макрокомандой CLS¤).

Таблица 8

Поле	Длина поля, байт	Содержимое
tsknam	8	Имя удаленной программы
—	2	Не используется
pr	2	Код запроса: 100000—ABORT 100001—CANCEL 100002—ABORT, CANCEL
ident	2	Идентификатор удаленной программы
—	2	Не используется

Таблица 9

Поле	Длина поля, байт	Содержимое
ident	2	0
ec	2	Код завершения обработки запроса на отмену передиспетчеризации и/или прекращение выполнения удаленной программы

Замечания по программированию

1. Идентификатор ident присваивается удаленной программе при ее запуске. Этот идентификатор используется для процедур отмены запуска и прекращения выполнения.

2. Единицы измерения интервалов диспетчеризации snt и передиспетчеризации rnt задаются в следующем виде: 1 — тики*, 2— секунды, 3 — минуты, 4 — часы.

3. Если в качестве значений интервалов smg и rmg задается. О, то инициация выполнения удаленной программы осуществляется немедленно.

Управление удаленными программами из программ на Фортране обеспечивается с помощью подпрограмм RUNNCW и ABONCW. До вызова этих подпрограмм программа пользователя должна

* Тик — интервал времени, определяемый частотой сети.

получить доступ к сети через вызов OPNNT. Вызовы подпрограмм RUNNCW и ABONCW являются синхронными, т. е. управление возвращается в пользовательскую программу только после завершения операций, выполняемых этими подпрограммами. Для успешного выполнения подпрограмм в удаленном узле должна быть установлена программа управления удаленными программами TCL.

Подпрограмма RUNNCW (выполнить установленную в удаленном узле программу) обеспечивает выполнение установленной в удаленном узле программы в следующих режимах:

немедленное выполнение программы в удаленном узле;

планирование выполнения программы в заданный момент времени;

планирование периодического выполнения программы на основе заранее заданных временных интервалов.

Формат:

```
CALL RUNNCW(lun,[status,]ndsiz,ndnam,
psswdsiz,psswd,tsksiz,tsknam,
ident,uic[,smg] [,snt] [,rmg] [rnt])
```

где lun — целочисленная переменная или константа одинарной точности, представляющая собой свободный номер логического устройства; status — необязательный целочисленный массив из двух элементов одинарной точности, в который будет занесено состояние завершения при возврате из RUNNCW: status (1) — код ошибки/завершения, 1 слово; status (2)—содержит код причины отсоединения или отказа, если код ошибки в status (1) указывает на отказ сети; ndsiz — целочисленная переменная или константа, указывающая длину имени узла (в байтах); ndnam — массив длиной от 1 до 6 байт, содержащий имя того узла, которому направляется этот запрос; psswdsiz—целочисленная переменная или константа, указывающая длину ИЗ (в байтах); ppsswd — массив, длиной от 1 до 8 байт, содержащий ИЗ для получения доступа к требуемому узлу. Привилегированный ИЗ позволяет запускать на выполнение программы в удаленном узле с любым uic. Непривилегированный ИЗ позволяет запускать программы только с назначенным uic; tsksiz — целочисленная переменная или константа, указывающая длину имени удаленной программы (в байтах); tsknam — массив длиной от 1 до 8 байт, содержащий имя той удаленной программы, которая должна быть выполнена; ident — адрес необязательной целочисленной переменной, назначенной программе, которая должна быть выполнена. Этот параметр используется, если планируется отмена или прекращение выполнения программы в дальнейшем; uic — массив из двух байт, содержащий коды группы и пользователя, с которыми программа будет выполняться в удаленном узле. В старшем байте задается код члена группы, в младшем байте—код группы. Этот аргумент не обязателен при задании привилегированного ИЗ. Если привилегированный пользователь опустит этот аргумент, то программа будет выполняться с назначенным ей в удаленном узле UIC по умолчанию; smg— интервал диспетчеризации, определяется интервалом времени от момента вызова подпрограммы до запуска программы; snt— единица измерения интервала диспетчеризации: 1—тики, 2 —секунды, 3 — минуты, 4 — часы; rmg — интервал времени передиспетчеризации программы определяется интервалом времени от момента запуска программы до ее перезапуска; rnt — единица измерения интервала передиспетчеризации (аналогично snt). Примечания:

1. Если аргументы smg, snt, rmg и rnt не задаются, то программа запускается на выполнение немедленно.
2. Если заданы аргументы smg и snt, но не заданы rmg и rnt, то программа запускается на выполнение только один раз через заданный интервал диспетчеризации.
3. Если заданы аргументы rmg и rnt, но не заданы smg и snt, то программа запускается на выполнение немедленно и затем каждый раз через заданный интервал передиспетчеризации.

Подпрограмма ABONCW (прекратить выполнение задачи или отменить запланированный запуск задачи) используется для прекращения выполнения программы или отмены запланированного запуска программы. Вызов подпрограммы имеет формат:

```
CALL ABONCW (lun [,status] ,ndsiz,ndnam, psswdsiz,psswd, tsksiz, tsknam, [,ident] [,mask]
```

где все параметры, кроме mask, аналогичны описанным параметрам подпрограммы RUNNCW.

mask — необязательный параметр, указывающий режим отмены или прекращения программы. Если этот параметр равен 0 или не задается, прекращается текущее выполнение программы; если его значение равно 1, отменяется только передиспетчеризация программы; если значение параметра

больше 1, прекращаются текущее выполнение программы и ее передиспетчеризация.

3.3. ВЗАИМОДЕЙСТВИЯ ТЕРМИНАЛОВ

Терминальные взаимодействия в сети ПП СТО/РВ реализуются за счет обслуживающих программ TLK и LSN, выполняющихся в различных узлах сети. Обмен информацией между терминалами в сети осуществляется по следующей схеме: терминал узла А — программа TLK узла А — сеть — программа LSN узла В — терминал узла В. Существующие в каждом из узлов пары программ TLK — LSN обеспечивают два режима терминальных взаимодействий — построчный и диалоговый.

Построчный режим обеспечивает обмен сообщениями, которые превышают длину одной строки ввода на терминале. При этом после передачи сообщения в одном направлении логический капал, используемый для взаимодействия TLK — LSN, отсоединяется.

Диалоговый режим обеспечивает интерактивное взаимодействие двух терминалов. Логический канал остается открытым на весь интервал терминального взаимодействия, а передаваемые сообщения могут превышать длину одной строки терминала. Диалоговый режим, задаваемый при генерации, является дополнительной возможностью пакета.

Завершение диалогового режима может осуществляться по инициативе любого терминала.

3.4. ТЕЛЕЗАГРУЗКА ОПЕРАЦИОННОЙ СИСТЕМЫ МОС РВ

Одной из дополнительных возможностей, предоставляемых ПП СТО/РВ, является телезагрузка операционной системы МОС РВ в удаленный узел. Образ систем МОС РВ, который телезагружается в удаленный узел, должен храниться в виде файла в главном узле. Процедура телезагрузки МОС РВ может осуществляться тремя способами: аппаратным загрузчиком (А3), А3 и командой LOAD, по команде LOAD.

В табл. 10 приведены характеристики всех способов телезагрузки.

Таблица 10

Способы	Выполняемые функции		
	главной ЭВМ	оператором удаленного узла	оператором главного узла
С помощью А3	Установить линию в ON*	Активизировать А3	Никаких действий не требуется
С помощью А3 и команды LOAD	Установить линию в MAINTENANCE	Активизировать А3	Ввести команду LOAD
По команде LOAD	Установить линию в MAINTENANCE	Никаких действий не требуется	Ввести команду LOAD

Для использования каждого из способов удаленная машина должна иметь аппаратный загрузчик. После активизации А3 удаленного узла загружает из ГВМ программу вторичной загрузки, которая и загружает файл образа МОС РВ.

Необходимо отметить, что загрузка по команде LOAD может осуществляться, только если в качестве коммуникационного устройства используется быстродействующий адаптер дистанционной связи. В этом случае для вызова А3 в удаленном узле должен быть указан ИЗ, который устанавливается в быстродействующем адаптере дистанционной связи при генерации.

Преимуществом использования для телезагрузки команды LOAD является возможность модификации параметров удаленного узла, определенных при генерации и находящихся в базе данных конфигурации. Перед телезагрузкой с помощью команд программы VNP необходимо установить в файле образа системы МОС РВ рабочее состояние удаленного узла.

Команды VNP также должны быть выполнены, если будут загружаться какие-либо дополнительные драйверы или активизироваться линии. Перед телезагрузкой линии должны быть установлены в состояние ON.

Команда LOAD имеет формат:

$$\text{LOAD}\left\{\begin{array}{l}\text{REMOTE remnod}\left[\text{VIA locnod lineid}\right]\\ \text{VIA locnod lineid}\end{array}\right\}\left\{\begin{array}{l}\left[\text{FROM lfile}\right]\left[\text{PASSWORD trig}\right]\\ \left[\text{PARAMETES}\left\{\begin{array}{l}\text{NONE}\\ \text{NAME remnod}\end{array}\right\}\right]\end{array}\right\}$$

* Описание состояний линий приведено в гл. 4

где в квадратные скобки заключены необязательные аргументы, в фигурные скобки заключены аргументы, из которых должен быть выбран один; REMOTE remnod—1—6 — символьное имя телезагружаемого удаленного узла. Если аргумент VIA не задается, то его значения locnod lineid берутся из базы данных конфигурации. Если же аргумент VIA задан, то его значения используются вместо содержащихся в базе данных конфигурации.

Если аргумент VIA задан, а REMOTE remnod не задан, то значения locnod lineid определяют использование программой NCP имени удаленного узла из базы данных конфигурации;

locnod — символьное имя местного узла (до 6 символов).

lineid — идентификация той линии, которая будет использоваться для телезагрузки. Эта линия должна быть установлена в состояние MAINTENANCE до ввода команды LOAD.

FROM lfile — спецификация загружаемого файла, содержащего образ памяти для телезагрузки. По умолчанию используется загрузочный файл, определенный в базе данных конфигурации. Если же этот аргумент задан, то он заменяет спецификацию файла (если таковая имеется) из базы данных конфигурации.

PARAMETRS NONE —означает, что параметры из базы данных конфигурации использоваться не должны.

PARAMETERS NAME remnod — задает имя удаленного узла, включаемого в сеть.

PASSWORD trig — указывает ИЗ для вызова А3 в удаленном узле. Этот аргумент применим только при использовании быстродействующего адаптера дистанционной связи. Аргументу trig должен быть присвоен четырехсимвольный пароль, по умолчанию используется значение, заданное в базе данных конфигурации. Если же аргумент задан, то он применяется вместо определенного в базе данных конфигурации.

Если аргументы PARAMETERS или FROM lfile заданы в команде LOAD, то после выполнения процесса телезагрузки следует установить линию в ON-состояние. При этом выполняется инициализация с удаленным узлом.

Если аргументы' PARAMETERS или FROM lfile не заданы в команде LOAD, то выдается оповещение о том, что завершена только команда LOAD, телезагрузка при этом не начиналась и произошел только вызов А3 в удаленном узле с АДС-Б. Затем, если установить линию в ON-состояние, начинается загрузка с помощью аппаратного загрузчика (если А3 в удаленном узле предназначен для загрузки ПП СТО/РВ) или инициализация удаленного узла.

Телезагрузка и прерывание программ в контрольных точках в системе МОС РВ. ПП СТО/РВ обеспечивает телезагрузку программ из главного узла в узел с МОС РВ и организацию прерываний в контрольных точках программ в МОС РВ.

Телезагрузка программ осуществляется с помощью сателлитного загрузчика SLD в узле МОС РВ и загрузчика HLD в главном узле. HLD и SLD связываются по логическим каналам.

Если в узле МОС РВ поступает запрос на запуск программы, находящейся в главном узле, то SLD посылает в HLD главного узла запрос на телезагрузку этой программы с диска главной машины. При генерации системы в таблицах распределений HLD можно указать программы как общие, т. е. телезагружаемые в любой сателлитный узел, или привязанные к определенному сателлитному узлу.

Программа SLD строится во время процедуры генерации ПП СТО/РВ для МОС РВ, а программа HLD и ее таблицы распределения— при генерации ПП СТО/РВ для главного узла ОС РВ.

Программа SLD должна быть установлена и фиксирована в файле образа системы МОС РВ. Программы, которые будут телезагружаться в сателлитный узел, должны быть установлены в файл образа системы МОС РВ перед его телезагрузкой.

Прерывание в контрольных точках позволяет приостановить выполнение программы, когда становится активной программа с более высоким приоритетом, установленная в том же разделе. Приостановленная программа выгружается из оперативной памяти в файл на главной машине (запись информации в контрольной точке), а программа с более высоким приоритетом загружается в раздел и запускается. Когда обработка более высокоприоритетной программы заканчивается, то приостановленная программа вновь загружается в оперативную память (чтение информации о контрольной точке) и продолжает выполняться. Примечания.

1. При организации прерывания в контрольной точке предполагается, что программа уже выполняется в разделе. Место для информации о контрольной точке должно быть внутри образа телезагружаемой программы на главной машине.

2. Таблица распределения памяти главного загрузчика HLD может иметь два явно выделенных набора таблиц распределения памяти:

внутренняя таблица распределения памяти, являющаяся частью самой программы HLD, создается пользователем во время генерации сети;

внешняя таблица распределения памяти, составляющая специальный файл (HLDTAB.TSK), имеющий сетевой UIC, создается пользователем после генерации сети.

Главный загрузчик HLD использует информацию, содержащуюся во внутренней или внешней таблице распределения памяти для поиска на диске образов телезагружаемых программ. Порядок работы с таблицами распределения памяти задается при генерации. HLD имеет более быстрый доступ к внутренним таблицам распределения памяти, чем к внешним, но первые более трудно изменить. В тех случаях, когда не предполагается часто вносить изменения в таблицы, следует использовать только внутренние таблицы распределения памяти.

3. Задачи общего назначения не могут быть прерваны в контрольной точке.

ГЛАВА IV

УПРАВЛЕНИЕ ПАКЕТОМ

Функции загрузки, управления, контроля и тестирования программного и технического обеспечения в ПП СТО/РВ реализуются с помощью команд оператора. Команды оператора обслуживаются сетевой программой связи с оператором NCP и разделяются на команды управления контроля сетью и тесты макрокоманд сетевого обслуживания. Имеются две версии NCP, обеспечивающие соответственно полный и сокращенный набор команд. Выбор конкретной версии задается при генерации пакета. Вызов программы NCP выполняется в соответствии с системными соглашениями по запуску обслуживающих программ в ОС РВ.

4.1. УПРАВЛЕНИЕ СЕТЬЮ

Команды NCP для управления сетью выполняют следующие функции:

загрузку и активизацию сетевых процессов и управление их работой;

управление рабочим состоянием узла сети;

управление рабочим состоянием линии;

изменение пароля в местном узле для приема или передачи информации.

Команды NCP для управления сетью описываются ниже.

1. Команда SET CEX — загрузить сетевой супервизор. Используется для загрузки сетевого супервизора CEX и всех сетевых процессов и линии, установленных в ON-состояние при генерации пакета. Сетевой супервизор остается загруженным до его выгрузки или до перезагрузки системы. Команда имеет формат:

SET CEX

2. Команда CLEAR CEX —выгрузить сетевой супервизор. Используется для выгрузки супервизора CEX и освобождения используемых сетью системных ресурсов. Перед вводом команды необходимо перевести узел в SHUT-состояние и разъединить все логические связи. Формат команды:

CLE CEX

3. Команда SET STATE LOCAL — изменить состояние узла. Команда используется для изменения состояния местного узла. Узел может иметь два рабочих состояния — ON и SHUT и автономное состояние — OFF.

ON — состояние узла определяет нормальное рабочее состояние узла в сети. В этом состоянии узел готов для сетевых передач данных и к инициации связи с другими узлами по линиям, установленным в ON-состояние. После загрузки пакета узел находится в автономном OFF-состоянии и до начала сетевых передач его необходимо перевести в ON-состояние.

SHUT — состояние определяет доступность узла для сетевых операций по существующим логическим связям, но создание новых логических связей запрещается. После завершения всех сетевых операций и разрыва всех логических связей узел переходит в OFF-состояние.

OFF — состояние узла определяет, что узел находится в автономном состоянии, т. е. не доступен для сетевых операций. Команда имеет формат:

$$\text{SET STA LOC}\left\{\begin{matrix}\text{ON}\\\text{SHUT}\end{matrix}\right\}$$

4. Команда SET LINE — установить параметры линии. Предназначается для загрузки соответствующих баз данных и связанных с ней процессов, если они еще не загружены. Дополнительные параметры команды могут использоваться для изменения состояния соответствующего канала (дуплексного и полудуплексного), а также параметров контроллера, установленных при генерации: адрес вектора прерывания, адрес регистра управления и приоритет устройства.

Линия остается активной до ее отсоединения или до перезагрузки системы. Все линии, подключаемые к мультиплексору передачи данных, должны быть отсоединены до изменения параметров контроллера. Команда имеет формат:

$$\text{SET LINE lineid [VECTOR vaddr] [PRIORYTET pnum]}$$
$$\text{[CSR addr]}\left[\left\{\begin{matrix}\text{FDX}\\\text{HDX}\end{matrix}\right\}\right]$$

где lineid — идентификация линии; vaddr — адрес вектора прерывания контроллера, по умолчанию используется адрес, определенный в файле конфигурации CFE; pnum — аппаратный приоритет контроллера, по умолчанию используется приоритет контроллера, определенный в CFE; addr — адрес регистра управления контроллером во внешней странице памяти, по умолчанию принимается адрес CSR, определенный в CFE; FDX — дуплексный, HDX — полудуплексный режимы работы. Для идентификации линии используется формат:

dev _ ctl _line

где dev — мнемоническое обозначение устройства передачи данных из поддерживаемых в ПП СТО/РВ; ctl — номер контроллера,
перед которым стоит знак подчеркивания; line — номер линии, перед которым — знак подчеркивания, задается только при использовании мультиплексора передачи данных.

5. Команда CLEAR LINE — отсоединить линию. Используется для отсоединения линии связи. Перед отсоединением линия должна быть переведена в OFF-состояние. Команда имеет формат:

CLE LINE lineid

где lineid — идентификация линии.

6. Команда SET PROCESS—загрузить процесс. Используется для загрузки сетевого процесса ПП СТО/РВ. Загруженный сетевой процесс остается в памяти до его удаления или до перезагрузки системы. Формат команды:

$$\text{SET PROCESS}\left\{\begin{matrix}\text{DCP}\\\text{NSP}\\\text{DLX}\\\text{DR}\end{matrix}\right\}$$

где DCP — процесс управления информационным каналом; NSP— процесс управления сеансами связи; DLX — телекоммуникационный процесс; DR — мнемоническое обозначение драйвера, устройства.

7. Команда CLEAR PROCESS — удалить сетевой процесс. Предназначается для деактивирования сетевого процесса. Команда имеет формат:

$$\text{CLE PROCESS}\left\{\begin{matrix}\text{DCP}\\\text{NSP}\\\text{DLX}\\\text{DR}\end{matrix}\right\}$$

8. Команда SET STATE LINE —установить состояние линии. Используется для изменения состояния указанной линии. Формат:

$$\text{SET STATE LINE lineid}\left\{\begin{matrix}\text{ON}\\\text{OFF}\\\text{MAINTENANCE}\end{matrix}\right\}$$

где lineid — идентификация линии; ON-состояние определяет линию в рабочем состоянии для работы с сетью; OFF-состояние определяет автономное состояние линии, т. е. недоступность линии для сетевых

операций; MAINTENANCE-состояние определяет доступность линии для проведения замкнутых тестов или телезагрузки удаленного узла.

9. Команда SET STATE KNOWN LINES—установить состояние линий местного узла. Используется для изменения состояния всех линий, присоединенных к местному узлу. Формат:

$$\text{SET STATE KNOWN LINES} \left\{ \begin{array}{l} \text{ON} \\ \text{OFF} \\ \text{MAINTENANCE} \end{array} \right\}$$

10. Команда SET PASSWORD—установить пароль. Предназначается для назначения паролей узлам сети на прием и передачу. Пароль задается при начальной установке связи с удаленным узлом. Пароль на прием определяет возможность доступа к местному узлу от удаленного узла. Пароль на передачу задается в удостоверяющем сообщении, посланном удаленным узлом. Для установления связи между местным и удаленным узлом пароль на прием в местном узле должен совпадать с паролем на передачу, получаемым в удостоверяющем сообщении от удаленного узла. Если в местном узле необходимо установить пароль на прием, то команда SET PASSWORD должна быть введена до установления линий этого узла в ON-состояние. Формат:

$$\text{SET PASSWORD} \left\{ \begin{array}{l} \text{RECEIVE} \\ \text{TRANSMIT} \end{array} \right\} \text{ pswd}$$

где RECEIVE — определяет присвоение пароля на прием;

TRANSMIT — определяет присвоение пароля на передачу; pswd— пароль от 1 до 8 символов.

Назначенный пароль сохраняется до выдачи другой команды SET PASSWORD, отмены пароля командой CLEAR PASSWORD или перезагрузки системы.

11. Команда CLEAR PASSWORD —отменить пароль. Используется для отмены пароля на прием или передачу в местном узле. Формат:

$$\text{CLEAR PASSWORD} \left\{ \begin{array}{l} \text{RECEIVE} \\ \text{TRANSMIT} \end{array} \right\}$$

12. Команда SET LOCAL LOOPBACK ENABLED— замкнуть местный узел. Предназначается для разрешения создания логических связей от местного узла к самому узлу по указанной линии. Узел остается в этом состоянии до установки линии в состояние OFF или MAINTENANCE. Формат:

SET LOCAL LOOPBACK ENABLED lineid

где lineid — спецификация линии.

13. Команда SET LOCAL LOOPBACK DISABLED— разомкнуть местный узел. Используется для разъединения логических связей на линиях, замкнутых на местный узел. Формат:

SET LOCAL LOOPBACK DISABLED

4.2. КОНТРОЛЬ СЕТИ

Команды NCP для контроля сети обеспечивают контроль в сети состояний местного узла, удаленных узлов, а также позволяют получить статистическую информацию о функционировании сети. Команды NCP для контроля сети приведены ниже.

1. Команда SHOW STATUS LOCAL —отобразить состояние местного узла. Используется для вывода на терминал, с которого выдана команда, или в выводной файл информации о состоянии местного узла. Отображаемая информация включает имя узла, версии операционной системы и протоколов и состояние местного узла (ON, OFF). Формат:

SHOW STATUS LOCAL [TO outfile]

где TO outfile — ключевое слово (TO) с спецификацией выводного файла является необязательным параметром команды, указывающим устройство или файл местного узла для вывода информации о состоянии местного узла. По умолчанию этого параметра информация выводится на терминал, с которого введена команда.

3. Команда SHOW STATUS REMOTE — отобразить состояние, удаленного узла. Используется для отображения состояния одного или всех известных удаленных узлов. Отображаемая информация включает имя узла и его состояние (ON или OFF). Формат:

$$\text{SHOW STATUS} \left\{ \begin{matrix} \text{REMOTE node} \\ \text{KNOWN REMOTES} \end{matrix} \right\} [\text{TO outfile}]$$

где REMOTE node — определяет удаленный узел с заданным именем node, состояние которого должно отображаться; KNOWN REMOTES определяет вывод информации о всех известных удаленных узлах.

3. Команда SHOW STATUS LINE — отобразить состояние линии. По команде отображается состояние одной или всех линий, связанных с местным узлом. Информация о состоянии включает идентификацию линии и ее состояние (ON, OFF или MAINTENANCE). Формат:

$$\text{SHOW STATUS} \left\{ \begin{matrix} \text{LINE lineid} \\ \text{KNOWN LINES} \end{matrix} \right\} [\text{TO outfile}]$$

где LINE lineid — определяет вывод состояния линии, заданной ее идентификацией (lineid); KNOWN LINES определяет отображение состояний всех линий местного узла.

4. Команда SHOW COUNTS LOCAL — отобразить счетчики местного узла. Предназначена для отображения значений счетчиков местного узла, являющихся статистической информацией о функционировании сети. Формат:

$$\text{SHOW COUNTS LOCAL } [\text{TO outfile}]$$

5. Команда SHOW COUNTS LINE — отобразить счетчики линий. Используется для отображения значений счетчиков одной или всех линий местного узла. Формат:

$$\text{SHOW COUNTS} \left\{ \begin{matrix} \text{LINE lineid} \\ \text{KNOWN LINES} \end{matrix} \right\} [\text{TO outfile}]$$

где LINE lineid — определяет отображение значений счетчиков линии, заданной ее идентификацией — lineid, KNOWN LINES — определяет отображение значений счетчиков всех известных линий местного узла сети. Линия становится известной в местном узле, если для нее была выполнена команда SET LINE.

6. Команда ZERO — очистить счетчики местного узла для очистки всех используемых счетчиков: местного узла, известных в местном узле линий или только указанной линии. Формат:

$$\text{ZERO} \left\{ \begin{matrix} \text{LOCAL} \\ \text{LINE lineid} \\ \text{KNOWN LINES} \end{matrix} \right\}$$

где LOCAL — определяет очистку счетчиков местного узла; LINE lineid — определяет очистку счетчиков линии, заданной ее идентификацией lineid; KNOWN LINES — определяет очистку счетчиков всех известных линий местного узла.

4.3. ПРОВЕРКА ФУНКЦИОНИРОВАНИЯ ПП СТО/РВ

Для проверки функционирования аппаратного и программного обеспечения сети в состав ПП СТО/РВ входит набор циклических тестов, работающих под управлением программы управления сетью NCP. С помощью тестов осуществляется передача сообщений в любой узел сети, допускающий замкнутое тестирование. Передаваемые при тестировании сообщения состоят из двоичных нулей и единиц или их чередующихся сочетаний. При запуске теста можно указать длину и число циклических сообщений. При обнаружении ошибки тестирование прекращается и выдается диагностическое сообщение с указанием причины сбоя и счетчика числа непереданных сообщений.

Замкнутые тесты. Тестирование сети может проходить на уровне линии или на уровне узла. Тесты линии получают прямой доступ к проверяемой линии и передают сообщения протокола управления информационным каналом через устройство возврата сообщений (LBD), установленное на линии. Для тестирования линия должна быть установлена в MAINTENANCE-состояние.

На уровне узла сообщения передаются по логическим каналам связи, установленным в пределах местного узла или между удаленными узлами.

В общем виде формат команды узла запуска тестов имеет вид:

$$\text{LOOP} \left\{ \begin{matrix} \text{LINE lineid} \\ \text{REMOTE node} \end{matrix} \right\} [\text{ COUNT n}] \quad [\text{ WITH} \left\{ \begin{matrix} \text{ONES} \\ \text{ZEROS} \\ \text{MIXED} \end{matrix} \right\}] \quad [\text{ LEN m}]$$

где LINE lineid — задается для запуска теста линии; REMOTE node — для запуска теста на уровне узла; lineid — идентификатор линии; node — имя узла; n — число сообщений, используемых для

тестирования в пределах от 1 до 32767, по умолчанию 1; WITH — определяет вид тестовых сообщений; ONES—двоичные единицы; ZEROS—двоичные нули; MIXED (по умолчанию)—комбинация единиц и нулей; m — определяет длину тестовых сообщений от 1 до 255 байт.

Замкнутый тест линии. Используется для проверки линии, замкнутой через устройство возврата сообщений (LBD) или модем с аналогичными возможностями (рис. 20).

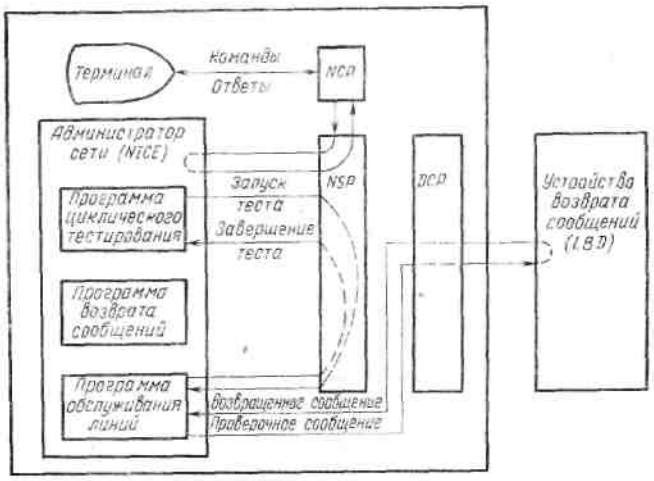


Рис. 20. Замкнутый тест линии

Циклические сообщения, передаваемые на уровне протокола управления информационным каналом, через устройство LBD возвращаются в программу обслуживания линии. Эта программа непосредственно управляет линией. Управление тестированием осуществляется программой циклического тестирования, где сравнивается возвращаемое сообщение с передаваемым. При обнаружении ошибки тест прекращается и на терминал выдается сообщение об ошибке. Для прогона замкнутого теста линии местный узел необходимо установить в ON-состояние, а проверяемую линию — в MAINTENANCE-состояние. Команда для запуска теста имеет формат:

$$\text{LOOP LINE lineid [COUNT n] [WITH } \left\{ \begin{matrix} \text{ONES} \\ \text{ZEROS} \\ \text{MIXED} \end{matrix} \right\} \text{] [LEN m]}$$

2. Внутренний замкнутый тест через NSP. Является внутриузловым циклическим тестом для проверки программного обеспечения ПП СТО/РВ на уровне узла. Схема работы теста показана на рис. 21. Тестовые сообщения, передаваемые по логическому каналу через NSP, адресуются программе возврата сообщений. Сообщение, возвращаемое в программу циклического тестирования, сравнивается с исходным сообщением для выявления ошибок. При обнаружении ошибки выполнение теста прекращается, логический канал разрывается и на терминал выдается сообщение об ошибке.

Для запуска теста необходимо перевести местный узел в ON-состояние и разомкнуть логические связи на линиях, замкнутых на местный узел. Команда для запуска теста имеет формат:

$$\text{LOOP REMOTE node [COUNT n] [WITH } \left\{ \begin{matrix} \text{ONES} \\ \text{ZEROS} \\ \text{MIXED} \end{matrix} \right\} \text{] [LEN m]}$$

3. Внутренний, замкнутый через NSP тест с использованием линии. Осуществляет тестирование сети на уровне узла (рис. 22). Логический канал для передачи сообщений создается между программой циклического тестирования и программой возврата сообщений через программу NSP и выбранную линию связи. На уровне управления информационным каналом логический канал замыкается через устройство LBD. Циклические сообщения, передаваемые по логическому каналу, возвращаются в программу циклического тестирования, где сравниваются с передаваемыми сообщениями. При обнаружении ошибки тестирование завершается, а на терминал выдается сообщение об ошибке.

Для прогона теста необходимо предварительно перевести местный узел и используемую линию в ON-состояние и разрешить создание логической связи по этой линии, замкнутой на местный узел. Команда для выбора параметров теста и его запуска имеет формат:

$$\text{LOOP REMOTE node [COUNT n] [WITH } \left\{ \begin{matrix} \text{ONES} \\ \text{ZEROS} \\ \text{MIXED} \end{matrix} \right\} \text{] [LEN m]}$$

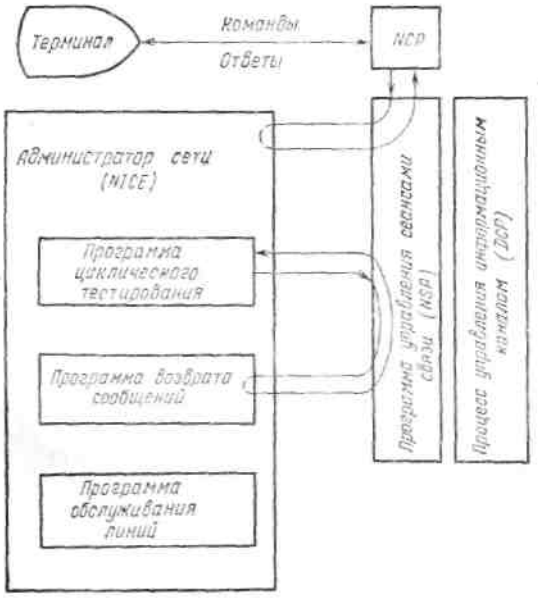


Рис. 21, Внутренний замкнутый тест

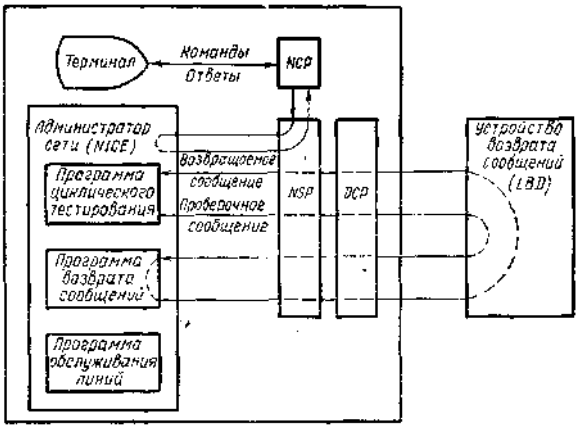


Рис. 22. Внутренний замкнутый тест с проверкой линии

4. Межузловой тест. Предназначен для проверки сети при передаче тестовых сообщений по логическим каналам связи, установленным между удаленными узлами сети (рис. 23). Логический канал создается между программой циклического тестирования местного узла и программой возврата сообщений в удаленном узле. Возвращаемое в программу циклического тестирования сообщение сравнивается с исходным передаваемым сообщением. При обнаружении ошибки тестирование завершается, логический канал уничтожается и на терминал выводится сообщение об ошибке. Для прогона межузлового теста необходимо предварительно перевести местный и удаленный узлы в ON-состояние. Формат команды:

$$\text{LOOP REMOTE node [COUNT n] [WITH } \left\{ \begin{matrix} \text{ONES} \\ \text{ZEROS} \\ \text{MIXED} \end{matrix} \right\} \text{] [LEN m]}$$

Тесты макрокоманд сетевого обслуживания. Для проверки макрокоманд сетевого обслуживания в состав ПП СТО/РВ включены две сетевые программы тестирования узлов передатчика и приемника — DTS и DTR. Эти программы обеспечивают четыре основных режима проверки уровня управления логическими каналами: тест соединения для проверки установления логических связей, тест данных для проверки передачи данных по логическому каналу, тест отсоединения для проверки уничтожения логических связей, тест прерываний для проверки передачи прерывающих сообщений по логическому каналу.

Тестовая программа DTR всегда выполняется в узле-приемнике под управлением программы DTS в узле-передатчике. Каждая программа DTR может обслуживать до трех логических каналов, т. е. обеспечивает одновременный запуск трех тестов. Программа DTS может управлять одновременно только одним логическим каналом. Для тестирования нескольких каналов можно использовать копии программы DTS.

Запуск одного из четырех возможных тестов обеспечивается по команде DTS, имеющей следующий формат:

$$\text{DTS>NODE/TEST=} \left\{ \begin{matrix} \text{DIS} \\ \text{CON} \\ \text{DATA} \\ \text{INT} \end{matrix} \right\} \text{/ RRINT} \left\{ \begin{matrix} \text{YES} \\ \text{NO} \end{matrix} \right\}$$

где NODE — имя узла-приемника, в котором выполняется программа DTR; значение ключа TEST определяет выбор одного из 4 тестов: CON—тест соединения, DATA — тест данных, DIS — тест отсоединения, INT — тест прерывания; ключ PRINT определяет возможность вывода сообщения программы DTR в узле-приемнике и может принимать одно из значений: YES — требуется вывод сообщения, NO — не требуется вывод сообщения.

По умолчанию ключи в команде принимают значения: NODE — имя местного узла, TEST=DATA, PRINT = YES.

Описание тестов. 1. Тест соединения. Тест предназначен для проверки обработки запроса на установление логического канала связи. Обеспечивается проверка как установления логического канала, так и отказа запроса на установление логического

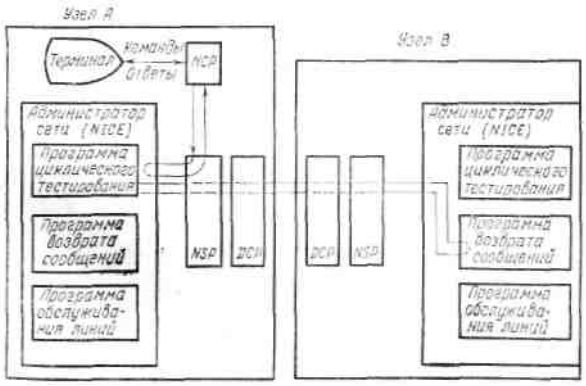


Рис. 23. Межузловой тест

канала в программе с передачей или без передачи пользовательских данных от DTR к DTS.

Команда для запуска теста имеет формат:

$$\text{CONNECT TEST} > \left\{ \begin{matrix} \text{ASS} \\ \text{REJ} \end{matrix} \right\} / \text{DATA} = \left\{ \begin{matrix} \text{NONE} \\ \text{STD} \\ \text{RCVD} \end{matrix} \right\}$$

где ACC и REJ — определяют режим теста:

ACC — проверка подтверждения программой DTR запроса на установление логического канала, REJ — проверка отвержения программой DTR запроса на установление логического канала; ключ DATA — задает один из трех возможных типов данных, передаваемых от DTR к DTS при обработке в DTR запроса на установление логического канала: NONE—данные не передаются, STD — передаются 16 байт стандартных данных, RCVD — передаются пользовательские данные, полученные в DTR из DTS по запросу на установление логического канала.

По умолчанию ключей в команде запускается тест для проверки отвержения запроса на установление логического канала без передачи данных от DTR к DTS.

2. Тест данных. Тест обеспечивает контроль правильности передачи данных по логическим каналам. Возможны следующие режимы использования теста:

тест потери, т. е. DTR игнорирует все принятые данные без проверки правильности их передачи;

тест последовательности обеспечивает передачу от DTS к DTR сообщений, содержащих последовательные четырехбайтовые номера. При получении в DTR сообщения с номером, не соответствующим порядковому номеру сообщения, тест завершается;

тест с шаблоном осуществляет передачу от DTS к DTR сообщений, содержащих как последовательный номер сообщения, так и стандартный код (шаблон). При несовпадении в DTR последовательного номера или стандартного кода тест прекращается;

эхо-тест обеспечивает передачу сообщения от DTS к DTR с возвратом сообщения в DTS без проверки последовательного номера и правильности передаваемых данных.

Команда для запуска теста имеет формат:

$$\text{DATA TEST} > \left\{ \begin{matrix} \text{SINK} \\ \text{SEQ} \\ \text{PAT} \\ \text{ECHO} \end{matrix} \right\} \text{/MSG} = \text{lll} \quad \text{BUFS} = \text{nnn} \quad \text{TIME} = \text{ttt} \left\{ \begin{matrix} \text{S} \\ \text{M} \\ \text{H} \end{matrix} \right\} \quad \text{M}$$

$$\text{BAUD} = \text{vvv}$$

где SINK, SEQ, PAT, ECHO—определяют соответственно режимы теста потери, теста последовательности, теста с шаблоном, эхо-теста; lll — определяет размер сообщения, т. е. размер передаваемых данных, nnn — задает уровень буферизации; ttt — длительность прогона теста в единицах времени; S—секунды, M — минуты, H — часы; vvv — быстродействие линии (бит/с).

По умолчанию ключи в команде принимают значения: SINK; MSG=длине сообщения в NSP; BUFS=1; TIME=1M; BAUD =0.

3. Тест отсоединения. Тест предназначен для проверки обработки запросов из программы DTR на уничтожение или отсоединение логического канала. Отсоединение или уничтожение логического канала может задаваться с передачей или без передачи данных от DTR к DTS. Команда для запуска

теста имеет формат:

$$\text{DISCONNECT TEST>} \left\{ \begin{matrix} \text{DSC} \\ \text{ABT} \end{matrix} \right\} \text{/DATA=} \left\{ \begin{matrix} \text{NONE} \\ \text{STD} \\ \text{RCVD} \end{matrix} \right\}$$

где DSC и ABT — определяют режим теста: DSC — проверка выполнения запроса на отсоединение логического канала; ABT — проверка выполнения запроса на уничтожение логического канала; ключ DATA — задает один из трех возможных типов данных, передаваемых от DTR к DTS в запросах на уничтожение или отсоединение логического канала: NONE — данные не передаются, STD — передаются 16 байт стандартных данных, RCVD — передаются пользовательские данные, полученные в DTR из DTS при установлении логического канала. По умолчанию ключей в команде запускается тест проверки выполнения запроса на отсоединение логического канала без передачи данных от DTR к DTS.

4. Тест прерывания. Тест обеспечивает проверку передачи прерывающих сообщений от DTS к DTR. В зависимости от данных, используемых для прерывающих сообщений, могут задаваться следующие режимы теста:

тест потери, т. е. DTR игнорирует все полученные данные прерывающего сообщения;

тест последовательности осуществляет передачу в прерывающих сообщениях от DTS к DTR последовательных четырехбайтовых номеров. При получении в DTR сообщения с номером, не соответствующим порядковому номеру сообщения, тест завершается;

тест с шаблоном обеспечивает передачу в прерывающих сообщениях от DTS к DTR как последовательного номера, так и стандартного кода. При несовпадении в DTR последовательного номера или стандартного кода тест завершается;

тест-эхо обеспечивает передачу прерывающего сообщения, полученного в DTR, обратно к DTS без проверки последовательного номера и правильности передаваемых данных прерывающего сообщения.

Команда для запуска теста имеет формат:

$$\text{INTERRUPT TEST>} \left[\left\{ \begin{matrix} \text{SINK} \\ \text{SEQ} \\ \text{PAT} \\ \text{ECHO} \end{matrix} \right\} \text{/MSG = lll} \quad \text{TIME = ttt} \left\{ \begin{matrix} \text{S} \\ \text{M} \\ \text{H} \end{matrix} \right\} \right]$$

где используемые ключи аналогичны ключам в команде для запуска теста данных. По умолчанию ключи в команде принимают значения: SINK, MSG=16, TIME = 1M.

В заключение сформулируем особенности использования тестов.

1. Длина сообщения MSG, задаваемая в командах для запуска тестов, должна удовлетворять следующим условиям: MSG> >0 — для теста потери и эхо-теста; MSG>4 — для теста последовательности; MSG>5 — для теста с шаблоном.
2. Уровень буферизации передачи BUFS должен задаваться в пределах от 1 до 16.
3. За единицу времени по умолчанию принимается секунда, наибольшая длительность теста 18 часов.
4. Программы DTR и DTS строятся как задачи ОС РВ.

Таким образом, широкое развитие и использование серии малых ЭВМ предоставляет, возможность создания однородных вычислительных сетей на базе этих ЭВМ. Рассмотренная структура программного обеспечения для построения сетей на базе СМ ЭВМ позволяет создавать однородные сети, имеющие различную топологию и сложность. Это обеспечивает возможность применения пакета ПП СТО/РВ в различных системах от автоматизации распределенного эксперимента до больших систем телеобработки данных.

Пакет ПП СТО/РВ предоставляет пользователю широкий набор услуг для распределенной обработки данных в сети: управление удаленными программами, доступ к удаленным файлам, терминальные взаимодействия, дистанционную пакетную обработку и т. д.

Надежность сетей, базирующихся на ПП СТО/РВ, достигается за счет возможности изменения конфигурации сети в процессе ее функционирования и предоставления широкого набора тестовых средств.

Все это определяет возможность широкого использования рассмотренного в книге программного обеспечения для построения однородных сетей на базе СМ ЭВМ.

* * *

В качестве заключения к настоящей книге представляется целесообразным кратко осветить современное состояние и перспективы развития других средств телеобработки ЭВМ серии СМ. В этом смысле наибольший интерес представляют следующие конфигурации: иерархические локальные системы, неоднородные сети СМ ЭВМ и ЕС ЭВМ и, наконец, локальные сети.

Иерархические локальные системы (ИЛС) предназначены для управления технологическими процессами и научными экспериментами и могут использоваться для автоматизации нескольких небольших или одного достаточно сложного распределенного объекта. Основной особенностью таких систем можно считать коллективное использование разделяемых относительно дорогих и сложных в обслуживании устройств (магнитные ленты, магнитные диски, АЦПУ и др.) несколькими ЭВМ, входящими в состав системы и удаленными на небольшие расстояния.

В ИЛС можно выделить ГВМ и сателлитные ЭВМ, в качестве которых используются ЭВМ СМ-1800, «Электроника-60» и др. Кроме сателлитных машин, к ГВМ в ИЛС могут подсоединяться локальные или удаленные терминалы. Из аппаратуры связи в таких системах поддерживаются адаптеры и мультиплексоры, приведенные в табл. 1.

ИЛС базируются на операционной системе РАФОС и пакете программ ПП СТО/Ф. Сателлитные машины оснащены эмулятором ОС РАФОС, который телезагружается из ГВМ. Архитектура ИЛС несимметрична и включает 5 уровней на главной и 4 на сателлитной ЭВМ. Наибольший интерес представляют два верхних уровня — прикладной и доступа к данным. На первом из них реализуется взаимодействие пользовательской программы, выполняемой в ГВМ с инициировавшим ее запуск терминалом сателлита. Уровень доступа к данным обеспечивает возможность обращения программы, выполняемой в сателлитной ЭВМ, к разделяемым ресурсам ГВМ. При этом используется стандартный набор интерфейсных средств, принятый в РАФОС для работы с файлами.

В целом ИЛС построены на базе уже известной читателю сетевой архитектуры САСМ и в них применяются подмножества рассмотренных ранее протоколов ПУИК, ПУСС и ПДД.

С точки зрения приведенных на рис. 1 взаимодействий ИЛС обеспечивают: организацию доступа к банку данных с терминалов сателлитных ЭВМ (F_3) и из их программ (F_{11}) за счет использования средств интерфейса доступа к данным. Такой доступ может осуществляться как в диалоговом режиме, так и в реальном времени; взаимодействия пользователя с запущенной по его требованию программой (F_4); управление объектом, выполняемое как оператором (F_5), так и автоматически (F_8). Для этого, естественно, требуются пользовательские программы, отрабатывающие алгоритмы управления; сбор данных с объекта и накопление их в банке данных системы (F_7) также должны реализовываться соответствующими пользовательскими программами.

Очевидно, что ПО иерархических локальных систем выполняет значительно меньший по сравнению с ПП СТО/РВ объем функций. Однако при выборе системы для автоматизации конкретного объекта должны также учитываться такие факторы, как временная реакция системы, занимаемый ею объем памяти и др. И в этом смысле для ряда случаев использование ИЛС представляется оправданным.

Предназначенные для автоматизации большого числа крупномасштабных объектов **неоднородные сети** ЕС ЭВМ и СМ ЭВМ. представляют другое важное направление работ по сетевой телеобработке СМ ЭВМ. Здесь стоит указать на использование минимашин как процессоров телепередачи данных и коммуникационных кластерных контроллеров в сети ЕС ЭВМ. Отметим, что при одновременном функционировании на одной СМ ЭВМ программного обеспечения эмулятора такого контроллера или процессора телепередачи и какого-либо из пакетов сетевой телеобработки СМ (например, ПП СТО/РВ) возможна организация межсетевых взаимодействий объектов однородных сетей ЕС ЭВМ и СМ ЭВМ.

Учитывая тот факт, что для пользователей рассматриваемых неоднородных сетей процессор телепередачи данных является недоступным элементом внутренней структуры системы, остановимся на кратком описании программного обеспечения коммуникационного кластерного контроллера на базе СМ ЭВМ для сетевой архитектуры ЕС ЭВМ.

В соответствии с сетевой архитектурой ЕС ЭВМ на СМ ЭВМ, эмулирующей коммуникационный кластерный контроллер, должны реализовываться функции следующих модулей сети ЕС: модуля физического управления (МФУ), модуля подключения к сети (МПС), модуля управления передачей (УП), модуля управления маршрутами (УМ), модуля управления звеном данных (УЗД). Выполнение функций перечисленных элементов сетевой телеобработки ЕС ЭВМ призван обеспечить пакет программ эмуляции протоколов (ПП ЭП/ЕС).

Этот пакет ориентирован на функционирование под управлением операционной системы ОС РВ. Пользовательские программы, выполняемые в СМ ЭВМ, связываются с программами пакета

посредством набора интерфейсных макрокоманд и подпрограмм (имеющих форматы, аналогичные рассмотренным форматам интерфейсных средств ПП СТО/РВ). Со стороны ЕС ЭВМ интерфейс пользовательских программ с комплексом также реализуется соответствующим системным ПО (виртуальный метод доступа, общий телекоммуникационный метод доступа и т. д.).

Пакет ПП ЭП/ЕС поддерживает дуплексные и полудуплексные линии с одноточечным или многоточечным соединением. При одноточечном соединении канал может быть коммутируемым, в противном случае — выделенным. Функционирование пакета допускается при использовании процессоров телепередачи данных в качестве связных процессоров главных ЕС ЭВМ сети.

Существуют три режима функционирования пакета: эмуляция, расширенная эмуляция и независимость. Эти режимы отличаются полнотой реализации соответствующих протоколов. В первом из них выполняется усеченное подмножество протоколов. В этом режиме невозможны взаимодействия программ в СМ ЭВМ с некоторыми пакетами ЕС ЭВМ. Расширенная эмуляция делает доступными дополнительные возможности протоколов, но требует специального программирования пользовательских программ, выполняемых в СМ ЭВМ. Наконец, последний режим обеспечивает передачу любых данных между ЕС ЭВМ и СМ ЭВМ и требует выполнения протоколов пользовательскими программами СМ.

Структурно пакет ПП ЭП/ЕС можно представить следующими компонентами: драйверы коммуникационных устройств, модули пользовательского интерфейса, модуль протоколов верхних уровней и модуль канального протокола.

Модуль протоколов верхних уровней в том или ином объеме (в зависимости от режима работы эмулятора) реализует функции сетевых объектов МФУ, МПС, УП и УМ. Сюда в первую очередь следует отнести следующие функции: организацию связи с модулем управления сетевым обслуживанием (МУСО) главной ЕС ЭВМ региона, установление и поддержание логических каналов (сессий в терминологии сетевой архитектуры ЕС ЭВМ) между МПС СМ ЭВМ и МПС главной ЭВМ, формирование и интерпретацию транспортных заголовков сообщений и передачу их соответствующим программам СМ ЭВМ, управление буферами и темпом передачи сообщений.

Модуль канального протокола в полном объеме реализует принятый в сети ЕС ЭВМ протокол управления каналом передачи данных и обеспечивает безошибочный обмен информацией между СМ ЭВМ и связным процессором ГВМ или с соседним процессором телепередачи данных (коммуникационным процессором сети ЕС).

Два других компонента пакета выполняют внутренние по отношению к сети функции. Драйвер коммуникационных устройств обеспечивает управление адаптером АДС/С (см. табл. 1) и при необходимости — модемом, в то время как модули пользовательского интерфейса осуществляют реализацию функций макрокоманд и требуемые преобразования данных.

Подобный подход, основанный на реализации на базе СМ ЭВМ одного из важнейших элементов сети ЕС ЭВМ, по-видимому, представляет собой одну из наиболее перспективных форм организации неоднородных сетей СМ ЭВМ и ЕС ЭВМ, поскольку при этом, не требуется доработок сложного и дорогостоящего программного обеспечения больших ЭВМ.

Еще одним, едва ли не самым важным и перспективным направлением работ по сетевой телеобработке СМ ЭВМ можно считать создание локальных сетей. Такие системы предназначены для автоматизации объектов, имеющих незначительное географическое распределение и требующих высокой надежности и скорости передачи данных. Так, в [10] приведены следующие типовые значения этих характеристик локальных сетей: удаленность — до 20 км, скорость передачи данных — до десятков Мбит/с, надежность—до 10^{-12}.

Такие высокие характеристики наряду с весьма низкой стоимостью локальных сетей в самом ближайшем будущем приведут, по-видимому, к их необычайно широкому применению в самых различных сферах человеческой деятельности. Так, по прогнозам фирмы Strategic Business Services Inc. (США), в 1985 г. общее число подключенных к локальным сетям терминалов составит 9,05 млн. шт., а к 1990 г. это число достигнет 21 млн.

В целом подобные системы представляют собой комплекс мини- и/или микро-ЭВМ и периферийных устройств, связанных витыми парами проводов, коаксиальным кабелем специальной или оптоволоконной шиной. Локальные сети допускают использование
самых различных топологий и строятся на базе самых разных архитектур. Общими в них являются разделение устройств между процессорами системы и замена маршрутизации, порождающей значительное число проблем в сетях, аналогичных рассмотренным в предыдущих главах книги, на

адресацию сетевых объектов. Это в свою очередь приводит к уменьшению объема программного обеспечения локальных сетей я, следовательно, к повышению их производительности. С другой стороны, столь высокие значения характеристик локальных сетей в значительной степени достигаются за счет создания большого числа специальных устройств.

Мы не будем здесь рассматривать концепции и реализацию локальных сетей на базе СМ ЭВМ, поскольку они пока находятся на начальном этапе своего развития. Отметим, однако, что это направление работ по сетевой телеобработке СМ ЭВМ, основанное на самой передовой технологии как микроэлектроники, так и программирования, будет развиваться самыми быстрыми темпами.

ЛИТЕРАТУРА

1. *Мартин Дж.* Системный анализ передачи данных. М., Мир, 1975, т, 1.
2. *Дэрниак А.* Новые вычислительные сети для распределенной обработки данных. — Электроника, 1978, № 5, с. 25—48.
3. *Какунин Л. А.* Проблемы проектирования и разработки вычислительных сетей. — Зарубежная радиоэлектроника, 1978, № 9, с. 65—100.
4. *Якубайтис Э. А.* Архитектура вычислительных сетей. М., Статистика, 1980.
5. *Клейнрок Л.* Вычислительные системы с очередями. М., Мир, 1981.
6. *Сипсер Р.* Архитектура связи в распределенных системах. М., Мир, 1981.
7. Сети ЭВМ. Под ред. акад. В. М. Глушкова. М., Связь, 1977.
8. *Бутрименко А. В.* Разработка и эксплуатация сетей ЭВМ. М., Финансы и статистика, 1981.
9. Малые ЭВМ и их применения. Под общ. ред. чл.-кор. АН СССР Б. Н. Наумова. М., Статистика, 1980.
10. *Weitzman C.* Distributed Micro/Minicomputer Systems. Prentice-Hall, Inc., Englewood Cliffs, N.Y., 1980. 403 p.
11. *Conord J. W.* Character-oriented data link control protocol. — IEEE Trans. Communs., 1980, v, 28, N 4, p. 445—454,