МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ
УНИВЕРСИТЕТ «МИФИ»

А.Ю. Никифоров, В.А. Русаков

ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ
(основы теории и практики)

Учебное пособие

Москва 2010

УДК 004.7 (075)
ББК 32.973.202я7
Н 62

Никифоров А.Ю., Русаков В.А. **Взаимосвязь открытых систем (основы теории и практики)**: Учебное пособие. М.: МИФИ, 2010. – 108 с.

Данное учебное пособие соответствует практической части предмета «Взаимосвязь открытых систем» и представляет собой курс семинарских и лабораторных занятий по названному предмету.

Предназначено для студентов пятого курса факультета кибернетики, изучающих курс «Взаимосвязь открытых систем».

Рецензент Е.Ф. Березкин

Рекомендовано редсоветом МИФИ в качестве учебного пособия

ISBN 978-5-7262-1324-8

Редактор Е.Г. Станкевич

Подписано в печать 06.09.10. Формат 60х84 1/16
Печ. л. 7,0. Уч.-изд.л. 7,0. Изд.№ 084-1.
Тираж 150 экз. Заказ 274.

Национальный исследовательский ядерный университет «МИФИ».
Типография МИФИ.
115409, Москва, Каширское ш., 31

СОДЕРЖАНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ СОКРАЩЕНИЙ

Сокращение	Расшифровка
АД	атомарное действие
АПС	агент пользователя справочника
АСС	агент справочной службы
БДП	блок данных протокола
БДС	блок данных службы
БСИ	база справочной информации
БУС	блок управления событиями
БфДП	буфер данных пользователя
ВС	вычислительная сеть
ВТ	виртуальный терминал
ИВС	информационно-вычислительная сеть
ИДС	информационное дерево справочника
ИСУ	информационная служба управления
КА	конечный автомат
КТС	конечная точка соединения
ЛВС	локальная вычислительная сеть
МККТТ	международный комитет по телеграфии и телефонии
МОС (ISO)	международная организация по стандартизации
МФО	метод формального описания
НАС	нотация абстрактного синтаксиса
ОРИ	относительное различаемое имя
ОЭПС	общий элемент прикладных служб
ПДУФ	передача, доступ и управление файлами
ПОУС	прикладной объект управления системой
ПП	прикладной процесс
ППМС	прикладной процесс менеджера сети
ППУС	прикладной процесс управления системой
САС	системный агент справочника
СВОС	среда взаимосвязи открытых систем

СЕВ	системные единицы времени
СОС	система обработки сообщений
СРС	среда реальных систем
СС	сетевое соединение
СпС	справочная служба
СЭПС	специальный элемент прикладных служб
ТДС	точка доступа к службе
ТС	транспортное соединение
УБ	управление безопасностью
УЗПВ	управление завершением, параллельностью и восстановлением
УИП	управляющая информация протокола
УКИ	управление конфигурацией и именами
УО	управление при отказах
УУ	управление учетом
УЭФ	управление эффективностью функционирования
ЭАУС	элемент агента управления системой
ЭМВОС	эталонная модель взаимосвязи открытых систем
ЭМУС	элемент менеджера управления системой
ЭП	элемент пользователя
ЭСДС	элемент службы доступа к справочнику
ЭССС	элемент системной службы справочника
ЭСУА	элемент службы управления ассоциацией
ЭСУО	элемент службы удаленных операций

ПРЕДИСЛОВИЕ

В пособии рассматриваются вопросы организации взаимосвязи открытых систем (ВОС). В целом оно служат дополнением к теоретической части курса (В. А. Русаков. Взаимосвязь открытых систем: Уч. пособие. М.: МИФИ, 2001), которое описывает практическую сторону курса ВОС.

Основу такой практической стороны составляет домашнее задание (ДЗ), разделенное на несколько частей. Результаты выполнения предшествующих частей используются в последующих в рамках одного и того же варианта.

Суть каждой части ДЗ – это разработка, реализация и оценка протокола соответствующего уровня в функциональной иерархии организации ВОС.

В каждую часть ДЗ входят описания следующих сервисов. Это, во-первых, тот сервис, который поставляется нижележащим уровнем. Для первой части ДЗ такой сервис обеспечивается эмулятором, который предоставляется каждому студенту наряду с прочей программной поддержкой выполнения ДЗ и ЛР. Для последующих частей ДЗ этот сервис должен быть обеспечен самим студентом на основе ранее проделанных работ. Во-вторых, это тот сервис, который должен быть предоставлен своим пользователям уровнем.

Другими словами, в каждой части ДЗ студенту необходимо разработать и реализовать свой, специфический для каждого варианта механизм, использующий (или, иначе, опирающийся на) предоставляемый сервис нижележащего уровня и обеспечивающий требуемый сервис уровню вышележащему.

Несмотря на внесенные разработчиками значительные упрощения, адекватные учебному характеру ДЗ, такие механизмы довольно сложны. Так, типичная протокольная реализация одной части ДЗ может содержать 200 и более операторов встроенного языка. Рекомендуется поэтому поддерживать процесс создания протоколов (хотя бы на фрагментарном уровне) подходящими средствами их формального описания, такими, как конечные автоматы.

В рамках одной системы механизмы, соответствующие смежным уровням, т.е. создаваемые в смежных частях ДЗ, взаимодействуют, используя параметризованные примитивы. Удобными и ши-

роко применяемыми средствами описания временных и логических взаимоотношений таких примитивов являются диаграммы последовательностей примитивов, таблицы соответствия их параметров, таблицы следования и диаграммы состояний-переходов.

Качество создаваемых студентом протоколов, их соответствие варианту задания оцениваются с помощью специальных программных средств.

В начале работ над ДЗ студент получает, помимо соответствующего программного обеспечения, документацию по встроенному языку, интерпретатору встроенного языка, эмулятору сетевого уровня и первую часть своего варианта задания. В описание первой части варианта задания входят показатели качества сервиса сетевого уровня и показатели качества сервиса транспортного уровня, которые должны обеспечить протокол транспортного уровня, создаваемый студентом. Дома студенты загружают свой вариант и приступают к написанию, отладке и оценке протокола транспортного уровня. *Внимание!* Во время регистрации перед первой загрузкой варианта необходимо правильно указать свои фамилию, номер группы и вариант. Несоблюдение этого условия приведет к трудностям при сдаче лабораторной работы. После отладки протокола на упрощенном анализаторе статистики (упрощение состоит в значительном уменьшении числа тестирующих взаимодействий, что обеспечивает быстрое получение огрубленных оценок параметров) студент переходит к полномасштабной оценке протокола. Если показатели качества транспортного уровня попадают в заданные интервалы и оценка протокола устраивает студента, то подготовка к лабораторной работе заканчивается. В противном случае студент продолжает модификацию и отладку протокола. В ходе лабораторной работы студент демонстрирует свой вариант преподавателю и если все действительно работает, преподаватель оценивает работу студента и выдает ему следующую часть домашнего задания. В противном случае студент продолжает работу над текущей частью своего домашнего задания.

В работе над последующими частями ДЗ в рамках своего варианта студент должен пользоваться протоколами, которые он уже создал при работе над предыдущими частями ДЗ. Отсутствие таких протоколов попросту не даст возможности оценить качество протокола, создаваемого в соответствии с текущей частью ДЗ. Это об-

стоятельство с учетом жесткого графика ЛР, т.е. сдачи частей ДЗ, должно стимулировать систематическую работу над ДЗ в течение всего семестра.

Такая работа должна включать в себя изучение как материалов к тематическим семинарам, ориентированных на конкретные части ДЗ, так и к теоретической части курса ВОС. При сопоставлении учебных протоколов с реальными следует, конечно, отдавать себе отчет в учебном характере лабораторного практикума. Соответствующие упрощения обусловлены, помимо прочего, относительно малой мощностью установки, эмулирующей все процессы по ходу организации взаимосвязи двух систем. Это проявляется, например, в вынужденно элементарном характере сценария взаимосвязи прикладных процессов – как говорится, на каждый малый чих на самом верху иерархии в основании всей пирамиды процессов происходит движение необыкновенное.

Также с целью упрощения подсистемы контроля и оценки создаваемых протоколов описания сервисов в вариантах частей ДЗ представлены через фиксированные соответствующие совокупности параметризованных примитивов.

Хотя подобное описание сервисов, вероятно, может способствовать применению определенных стилистических приемов в ходе создания протоколов, авторы пособия, тем не менее, полагают это обстоятельство не слишком ограничительным.

1. ЭМУЛЯТОР СЕТЕВОГО УРОВНЯ

Программное обеспечение лабораторного практикума по курсу “Взаимосвязь открытых систем” имеет в своем составе средство, обеспечивающее сервис сетевого уровня семиуровневой модели ВОС. Этот уровень расположен между вышележащем транспортным и нижележащим канальным уровнем. Далее это средство называется “эмулятор сетевого уровня” или просто “эмулятор”.

Эмулятор способен менять свое поведение в зависимости от задаваемых показателей качества, отражающих показатели качества реальных сетевых сред, и предоставляет для использования студентом сетевой сервис, содержащий набор услуг, достаточный для целей обучения в рамках ДЗ.

1.1. Показатели качества сетевого уровня и их интерпретация эмулятором

е характеристики даны в системных единицах времени эмулятора и не имеют никакого отношения к реальным единицам времени. Вероятностные характеристики показателей качества задаются в десятитысячных долях. Например, на рис. 1.1 вероятность ошибки при установлении соединения равна 0,1. Если показатель качества описан в виде среднего значения с допустимым разбросом, то имеется в виду, что значение такого показателя представляется всякий раз как значение равномерно распределенной случайной величины из интервала (среднее значение минус величина допустимого разброса, среднее значение плюс величина допустимого разброса).

В качестве основных показателей качества были выбраны и реализованы в эмуляторе следующие:

- максимальный размер (длина) передаваемого блока данных. Эмулятор может передавать блоки данных как неограниченного (если не принимать во внимание объем доступной памяти, диапазон значений 32-битовых чисел и прочие очевидные ограничения) размера, так и размера, чья величина ограничена значением этого параметра. Для реальных сетевых сред это значение обычно лежит в диапазоне от 128 байт (X.25, ATM) до 1000–2000 байт (PPP, SLIP, и т.д.). При введенном ограничении на допустимый размер пакета от студента, возможно, потребуется в процессе выполнения лабораторной работы реализовать функцию разбиения (сегментации) блоков данных на меньшие и обратной сборки блоков данных из получаемых частей;
- скорость передачи данных в байтах за 1000 условных единиц времени. Этот параметр задается своим средним значением и допустимым разбросом и используется при вычислении времени передачи конкретного пакета данных между пользователями сетевой службы;
- в я задержка при установлении соединения. Как и ос-

  требует указания среднего значения и допустимого разброса. й задержкой при установлении соединения считается время между выдачей на инициирующей стороне примитива

N_CONNECT.REQ и получением на отвечающей стороне примитива N_CONNECT.IND. Этот же параметр используется для вычисления времени между выдачей на отвечающей стороне примитива N_CONNECT.RESP и получением примитива N_CONNECT.CONF на стороне инициатора. В случае возникновения ошибки при установлении соединения время между выдачей запроса N_CONNECT.REQ и получением примитива N_DISCONNECT.IND вычисляется также на основе этого параметра;

- вероятность ошибки при установлении соединения. Представляется как оценка в виде отношения числа неудачных попыток соединения к общему числу выданных запросов N_CONNECT.REQ;
- транзитная задержка. Входят затраты на гипотетические служебные расходы протоколов сетевого уровня. Исходя из значения транзитной задержки, задаваемого средней величиной и допустимым разбросом, и значения скорости передачи, для каждого передаваемого пакета вычисляется время между выдачей примитива N_DATA.REQ на стороне отправителя и получением примитива N_DATA.IND на принимающей стороне;
- вероятность искажения пакета. Представляется как оценка в виде отношения числа пакетов, искаженных в ходе передачи по сети, к общему числу переданных пакетов. В эмуляторе возможное искажение переданного пакета производится заменой содержимого байтов пакета на случайное значение;
- вероятность потери пакета данных. Ее оценка – это вычтенное из единицы отношение числа полученных примитивов N_DATA.IND к числу переданных примитивов N_DATA.REQ.
- вероятность случайного разрыва соединения в течении каждых 1000 системных единиц времени;
- временная задержка при разрыве соединения. Задаваемое средним значением и допустимым разбросом время между выдачей запроса N_DISCONNECT.REQ и получением на другой стороне соединения N_DISCONNECT.IND;
- вероятность появления дубликата. Представляется как оценка в виде отношения числа пакетов, у которых появились дубликаты в ходе передачи по сети, к общему числу переданных пакетов. В

эмуляторе возможное искажение переданного пакета производится заменой содержимого байтов пакета на случайное значение.

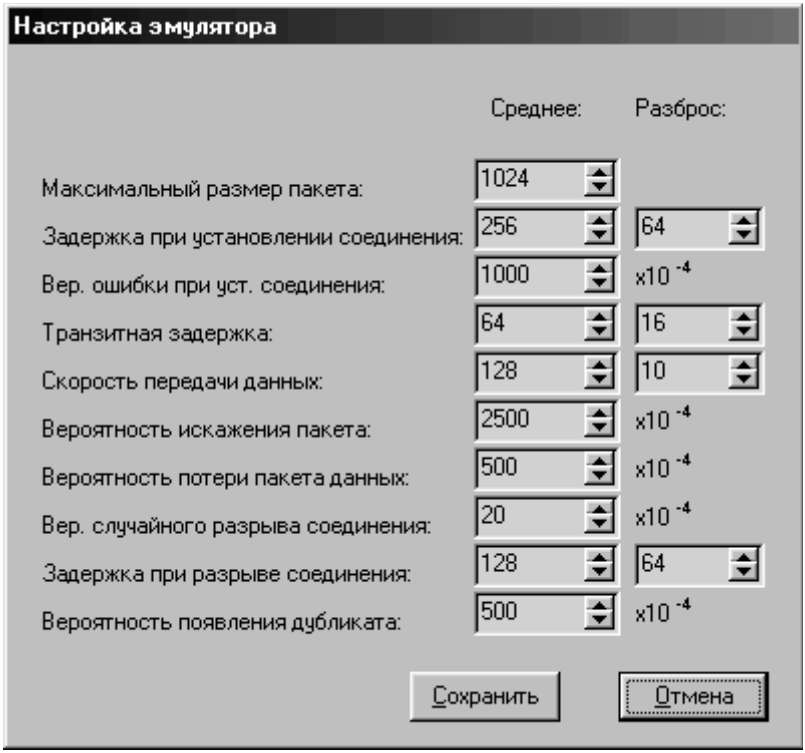


Рис. 1.1. Панель настройки показателей качества сетевого уровня

1.2. Сервис сетевого уровня

Эмулятор представляет услуги сетевого уровня для использования их на вышележащих уровнях. Реализованный эмулятор предоставляет транспортному уровню сервис, ориентированный на соединение, а также поддерживает передачу дейтаграмм. Эмулятор не поддерживает функции управления, синхронизации, а также передачу данных с подтверждением. Эмулятор сетевого уровня использует примитивы, описанные в табл. 1.1.

Таблица 1.1

Примитивы, используемые эмулятором

Примитив	Тип	Параметры	Описание
N_CONNECT .REQ	Запрос	address	Генерируется на транспортном уровне и отсылается вниз на сетевой уровень в начале попытки установления сетевого соединения. (address – адрес вызываемой системы)
N_CONNECT .IND	Индикация	address	Генерируется на сетевом уровне и отсылается вверх на транспортный уровень в целях индикации поступившего запроса на установление сетевого соединения. (address – адрес вызывающей системы)
N_CONNECT .RESP	Ответ	address	Генерируется на транспортном уровне и отсылается вниз на сетевой уровень в ответ на ранее полученный N_CONNECT.IND. (address – адрес вызываемой системы)
N_CONNECT .CONF	Подтверждение	address	Генерируется на сетевом уровне и отсылается вверх на транспортный уровень в целях индикации поступившего N_CONNECT.RESP. (address – адрес вызывающей системы)
N_DISCONNECT .REQ	Запрос	address	Генерируется на транспортном уровне и отсылается вниз на сетевой уровень для разрыва сетевого соединения

Продолжение табл. 1.1

Примитив	Тип	Параметры	Описание
N_DISCONNECT .IND	Индикация	-	Генерируется на сетевом уровне и отсылается вверх на транспортный уровень в целях индикации разрыва сетевого соединения или при возникновении ошибки при попытке его установления
N_DATA.REQ	Запрос	address userdata	Генерируется на транспортном уровне и отсылается вниз на сетевой уровень для передачи блока данных
N_DATA.IND	Индикация	userdata	Генерируется на сетевом уровне и отсылается вверх на транспортный уровень в целях индикации поступившего блока данных
N_DATAGRAM .REQ	Запрос	address userdata	Генерируется на транспортном уровне и отсылается вниз на сетевой уровень для передачи дейтаграммы. (address – адрес принимающей системы)
N_DATAGRAM .IND	Индикация	address userdata	Генерируется на сетевом уровне и отсылается вверх на транспортный уровень в целях индикации поступившей дейтаграммы. (address – адрес передающей системы)

1.3. Процесс эмуляции

Процесс эмуляции при использовании режима с установлением соединения заключается в следующем. Пусть одна из систем (например, система А) пытается установить сетевое соединение с другой системой (например, системой В), выдавая на сетевой уровень примитив N_CONNECT.REQ. Если системы не находились в состоянии “соединены” или “ожидание сетевого соединения”, то через некоторое время (зависит от параметра “временн я задержка

при установлении соединения") и с некоторой вероятностью (зависит от параметра "вероятность ошибки при установлении соединения") системы переходят в состояние "ожидание сетевого соединения" и на транспортный уровень системы В выдается примитив N_CONNECT.IND. После этого система В выдает на сетевой уровень примитив N_CONNECT.RESP. Так как системы находятся в состоянии "ожидание сетевого соединения", то через некоторое время (зависит от параметра "временн я задержка при установлении соединения") и с некоторой вероятностью (зависит от параметра "вероятность ошибки при установлении соединения") системы переходят в состояние "соединены" и на транспортный уровень системы А выдается примитив N_CONNECT.CONF.

Когда две системы находятся в состоянии "соединены", между ними возможна передача данных. Передающая система выдает на сетевой уровень примитив N_DATA.REQ. С некоторой вероятностью данные искажаются (зависит от параметра "вероятность искажения пакета") и теряются (зависит от параметра "вероятность потери пакета данных"). Если данные не потеряны, то через некоторое время (зависит от параметра "транзитная задержка" и "скорость передачи данных") на транспортный уровень принимающей системы выдается примитив N_DATA.IND.

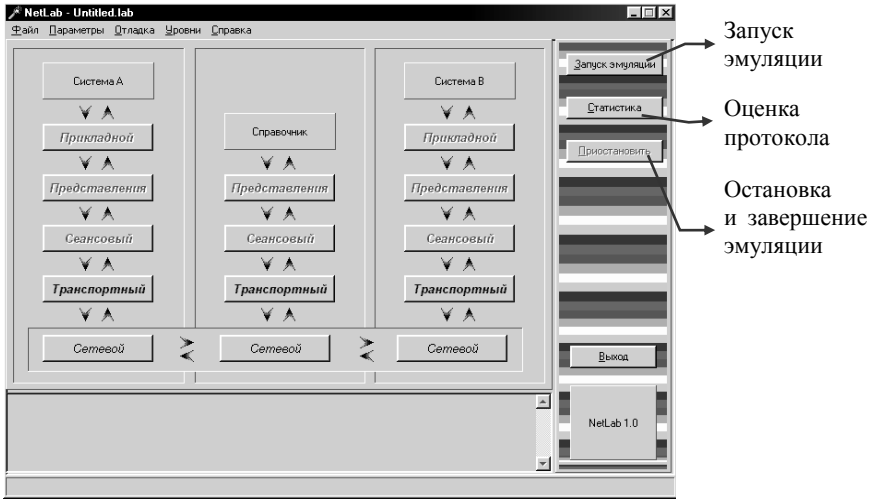
Разрыв соединения между двумя системами может произойти самопроизвольно (зависит от параметра "вероятность случайного разрыва") или в результате выдачи одной из систем на сетевой уровень примитива N_DISCONNECT.REQ. После выдачи примитива N_DISCONNECT.REQ сетевое соединение разрывается и через я задержка при разрыве соединения") на транспортный уровень второй системы выдается примитив N_DISCONNECT.IND.

Случай с использованием дейтаграммного режима значительно проще. Для передачи данных нет необходимости устанавливать соединение. Для передачи данных передающая система выдает на сетевой уровень примитив N_DATAGRAM.REQ. С некоторой вероятностью данные искажаются (зависит от параметра "вероятность искажения пакета") и теряются (зависит от параметра "вероятность потери пакета данных" и "скорость передачи данных"). Если данные не потеряны, то через некоторое время (зависит от

параметра “транзитная задержка”) на транспортный уровень принимающей системы выдается примитив N_DATAGRAM.IND.

2. ИНТЕРФЕЙС ПРОГРАММНОГО КОМПЛЕКСА

2.1. Основная панель лабораторного практикума



2.2. Главное меню

Пункт “Файл” (работа с файлами) содержит следующие подпункты:

“Создать” – команда открытия нового файла.

“Открыть” – команда открытия существующего файла.

“Открыть вновь” – команда открытия файлов, которые были открыты или сохранены ранее.

“Сохранить” – команда сохранения текущего файла.

“Сохранить как” – команда сохранения текущего файла с новым именем.

“Выход” – команда выхода.

Пункт “Параметры” (установка параметров) содержит следующие подпункты:

“Параметры эмулятора” – вызывает панель настройки показателей качества сетевого уровня.

“Пароль” – вызывает панель ввода административного пароля.

“Параметры” – вызывает панель настройки лабораторного комплекса.

Пункт “Отладка” (установка отладчика) содержит следующие подпункты:

“Трассировка” – включает (и выключает) режим вывода имен выполняемого события (определение события см. в п. 5).

“Отладчик” – вызывает отладчик.

“Просмотреть журнал” – просмотр журнала.

Пункт “Уровни” (установка доступа к уровням) содержит следующие подпункты (доступные только после ввода пароля):

“Прикладной процесс” – включает (и выключает) доступ к прикладным процессам.

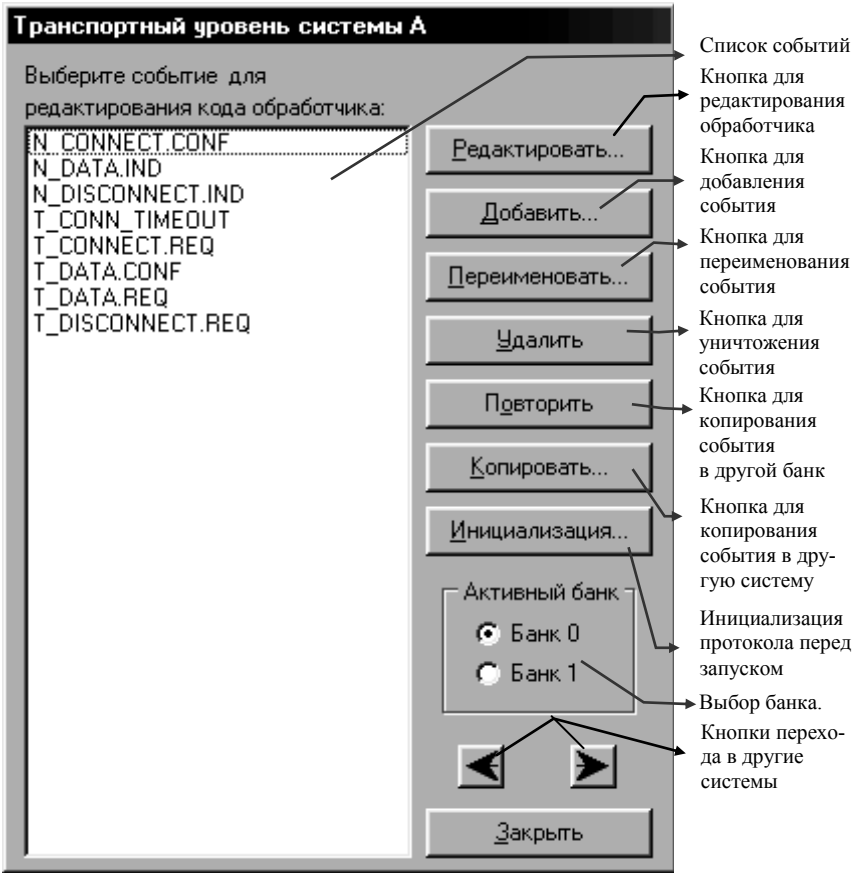
“Прикладной” – включает (и выключает) доступ к прикладному уровню.

“Представления” – включает (и выключает) доступ к уровню представления.

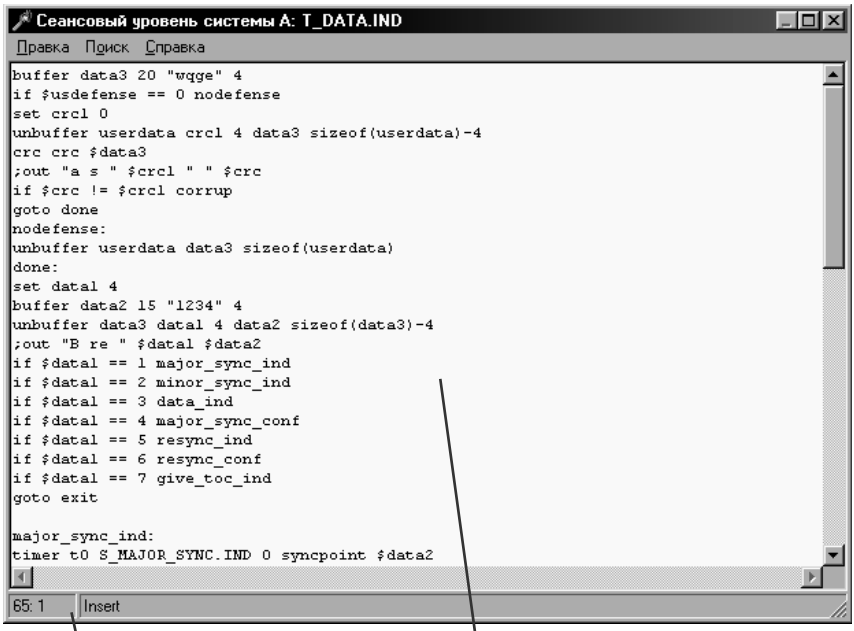
“Сеансовый” – включает (и выключает) доступ к сеансовому уроню.

“Транспортный” – включает (и выключает) доступ к транспортному уровню.

2.3. Панель редактора событий

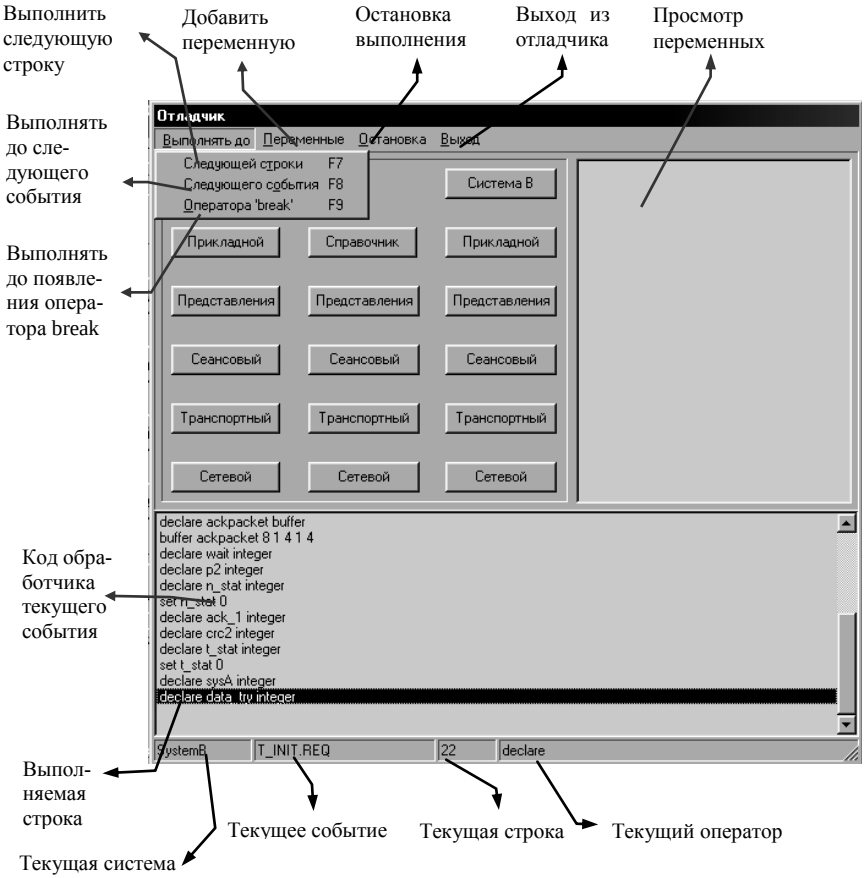


2.4. Панель редактирования обработчика события



2.5. Отладчик

С помощью отладчика можно выполнить построчную обработку протокола. При необходимости можно выполнить обработчик события без остановок после выполнения очередной строки, а также продолжить обработку всех событий без остановок до появления специального оператора break. В любом случае в панели отладчика отображается код обработчика события с выделением текущей строки. Также можно просмотреть значения переменных (только числовых) на текущем уровне.



2.6. Замечания к программному комплексу

- На нижележащий уровень могут посылаться события с именами, оканчивающимися только на ‘.REQ’ и ’.RESP’, а на вышележащий уровень – события с именами, оканчивающимися только на ‘.IND’ и ‘.CONF’.
- Имена примитивов не должны содержать символы логических и арифметических операций, т.е. символы /,*,-,+,!,&,|,%,<,>,=.
- Эмулятор может поддерживать только одно сетевое соединение на систему (так как систем три, то фактически в данный момент только одно сетевое соединение). Поэтому не забывайте разрывать установленное соединение перед установлением нового.

3. СИНТАКСИС ВСТРОЕННОГО ЯЗЫКА НАПИСАНИЯ ПРОТОКОЛОВ

3.1. Выражения

3.1.1. Переменные и константы

Имена переменных в выражении всегда начинаются с символа '$'. Например: $a, $n, $opyat_povislo и т.п. Длина имени переменной ограничена 254 символами.

Переменные всегда считаются глобальными в пределах уровня данной системы, за исключением тех случаев, когда они передаются в качестве параметров события.

Переменные типизируются в ходе выполнения программ по факту первого присваивания (т.е. тип первого присвоенного переменной значения и является типом этой переменной). Переменные могут быть типа число (Integer), строка (PChar) и буфер (аналог BLOB – “Binary Large Object“ в SQL).

Значения типа «число» могут рассматриваться в качестве логических значений (0 – false, любое другое число – true) в выражениях, включающих логические операции, и в операторе условного перехода.

Строковые константы записываются в двойных кавычках.

Символьные константы можно записывать через их числовой код (код ASCII) с помощью символа “#“. Например, символ “1” можно записать следующими способами: “1” или #49.

Числовые константы записываются традиционным образом, как в языках Pascal, C и т.д.

Константы типа «буфер» в языке не определены.

3.1.2. Операции

Операции над числами

+	арифметическое сложение
–	арифметическое вычитание
/	целочисленное деление
*	умножение
%	вычисление остатка от деления
&	побитовое "И"
|	побитовое "ИЛИ"
>	больше
<	меньше
==	равно
!=	не равно
>=	больше или равно
<=	меньше или равно
&&	логическое "И"
||	логическое "ИЛИ"
!	логическое "НЕ"

Операции над строками

+ производит сцепление (конкатенацию) строк

pos(<строковое выражение>,<переменная-строка>)

Пример:

pos(“bc”, s) если $s="abcd", то результат 2,
если $s="abdc", то результат 0.

Функция pos ищет вхождение подстроки в строке и возвращает номер первого символа подстроки в строке или 0, если строка не содержит подстроки или в случае какой-либо ошибки. Например, если переменная имеет тип integer или вместо строкового выражения написано числовое выражение.

Другие операции

Над переменными любого типа определена функция sizeof(<variable>), возвращающая размер переданного ей значения в байтах. Например:

```
set me 2
sizeof(me)              результат: 4
set you_too "ya geniy"
sizeof(you_too)         результат: 9
```

Для переменных типа буфер функция sizeof возвращает длину буфера.

3.1.3. Приоритеты операций

Наивысшим и равным между собой приоритетом обладают операции умножения, деления и вычисления остатка от деления. Далее следует группа равных по приоритету операций сложения, вычитания и побитовые операций; затем группа логических операций "И", "НЕ", "ИЛИ"; наименьшим приоритетом обладает группа операций сравнения. Порядок выполнения нескольких операций равного приоритета слева направо.

Порядок следования операций может быть изменен при помощи круглых скобок.

3.2. Метки

Любая строка программы может быть предварена меткой, заканчивающейся двоеточием. Например:

```
cool_label: down T_CONNECT_REQ my_cool_data $cooldata
```

3.3. Комментарии

Строка программы, начинающаяся с символа ';', считается комментарием и игнорируется при исполнении программы.

3.4. Операторы

3.4.1. Оператор присваивания (set)

Синтаксис:

```
set <имя переменной> <выражение>
```

Пример:

```
set my_cool_variable $other_cool_variable+1
```

Оператор вычисляет значение указываемого выражения и присваивает переменной с заданным именем. Если переменная уже была создана и ее тип не совпадает с типом вычисленного выражения, возникает ошибка времени выполнения.

3.4.2. Операторы управления примитивами (up, down, timer, untimer)

Синтаксис:

up <имя_примитива> {<имя_параметра> <выражение>}

down <имя_примитива> {<имя_параметра> <выражение>}

timer <имя переменной> <имя_примитива> <задержка> {<имя_параметра> <выражение>}

untimer <выражение>

Пример:

```
up S_DISCONNECT_NOTIFY errorcode 0
down N_DISCONNECT_REQ
timer timer1 T_CONNECT_TIMEOUT 400+$lastctime nextctime $lastctime+100 userdata $udata
untimer $timer1
```

Операторы up и down производят локализированную пересылку примитива с данного уровня через соответствующие точки доступа объектам соответственно выше- и нижележащего уровней (генерируют примитив на уровне выше- и ниже текущего, соответственно, с заданным именем и списком параметров).

Оператор timer генерирует примитив на текущем уровне через указанный интервал времени в системных единицах времени, начиная с момента обработки данного оператора с заданными именем примитива и списком параметров. В переменную с указанным именем помещается уникальный числовой идентификатор, который затем может быть использован в качестве выражения, передаваемого оператору untimer. В указанном примере оператор timer вызовет генерацию события T_CONNECT_TIMEOUT с параметрами nextctime и userdata, равными, соответственно, $lastctime+100 и $udata через 400+$lastctime системных единиц времени.

Оператор untimer отменяет генерацию отложенного события с указанным числовым идентификатором.

3.4.3. Операторы управления буферами (buffer, unbuffer, crc)

Синтаксис:

buffer <имя переменной-буфера> <длина буфера>
{ <значение поля в буфере> <длина поля> }
unbuffer <имя переменной-буфера> { <имя переменной> <длина поля> }
crc < имя переменной-числа >< имя переменной-буфера >

Пример:

buffer ctrlbuf 1028 $userdata 1024 $dcrc 4
unbuffer ctrlbuf userdata 1024 dcrc 4
crc paccrc $databuffer

Оператор buffer создает переменную типа «буфер» из полей указанной длины. Если фактическая длина переданного значения оказывается меньше указанной длины поля, то остаток отведенного поля заполняется нулями. Если суммарная длина полей оказывается больше размера буфера, то происходит ошибка времени выполнения.

Оператор unbuffer разбивает переменную типа «буфер» на поля указанной длины. Извлекаемые поля считаются имеющими тип, совпадающий с типом переменной, в которую помещается поле.

Оператор crc подсчитывает контрольную сумму буфера и заносит ее в переменную с заданным именем. Если переменная уже была создана и ее тип не совпадает с типом «число», возникает ошибка времени выполнения.

3.4.4. Операторы управления очередями (queue, dequeue, qcount)

Синтаксис:

queue <имя очереди> <выражение>

Пример:

queue my_queue $fully_useless_data

Оператор queue добавляет элемент в очередь с указанным именем. Элемент может иметь любой тип; при извлечении элемента из очереди выражение будет иметь тот же тип, что и добавленный элемент (т.е. тип выражения будет сохранен в очереди). Одна и та же очередь может содержать элементы различных типов.

Над очередями определены операции dequeue и qcount. Операция dequeue(<имя очереди>) возвращает первый элемент очереди с

указанным именем; если очередь пуста, возникает ошибка времени выполения. Операция qcount(<имя очереди>) возвращает число элементов, находящихся в очереди.

3.4.5. Операторы управления (goto, if)

Синтаксис:

goto <имя метки>
if <выражение> <имя метки>

Пример:

```
if $chislo_paketov > 100 disconnect_ego
goto podozhdem_esche_naverno
```

Оператор goto вызывает безусловный переход на заданную метку.

Оператор if вызывает переход на заданную метку в случае, если переданное выражение является логически истинным (т.е. является выражением типа «число», отличным от нуля). В противном случае выполнение программы продолжается.

3.4.6. Операторы диагностики (out)

Синтаксис:

out <выражение>

Пример:

```
out "soedinenie razorvanno"
out ($chislo_oshibok_na_linii+$chislo_moih_oshibok)/2
```

Оператор out выводит значение вычисленного выражения (строкового или числового) в диагностическое окно. Если вычисленное выражение имеет тип «буфер», то в окне выводится длина этого буфера.

3.4.7. Операторы управления строками (copy, delete)

Синтаксис:

copy(<подстрока>,<исходная строка>,<начало>,<длина>)
delete(<строка>,<начало>,<длина>)

Пример:

copy(s1, s, 2, 3) если $s="abcdef", то $s1="bcd"
delete(s, 2, 3) если $s="abcdef", то $s="bcd"

Оператор copy возвращает подстроку указанной длины, начинающуюся с указанной позиции в строке.

Оператор delete удаляет из строки подстроку указанной длины, начиная с указанной позиции в строке.

3.5. Ограничения, накладываемые языком

Очереди не являются переменными, и с ними невозможны никакие операции, кроме queue и dequeue.

Невозможна передача управления посредством goto на метку, находящуюся в описании другого примитива (метки всегда локальны).

Нельзя создать локальные переменные иным способом, кроме как путем передачи их в виде параметров, соответствующих событию (определение события см. в п. 7). Кроме того, параметры, соответствующие событию, всегда доступны только для чтения (т.е. попытка выполнения присваивания в коде обработчика события переменной, имя которой совпадает с именем параметра, вызовет ошибку времени выполнения).

3.6. Сообщения об ошибках

Сообщения об ошибках периода исполнения имеют следующий вид: “<Система>, Событие <имя события>, строка <номер строки>: <сообщение об ошибке>.”, где

Система – название системы, в которой произошла ошибка.

Имя события – имя события, в котором произошла ошибка.

Номер строки – номер строки, на которой произошла ошибка.

сообщение об ошибке – сообщение об ошибке.

Таблица 3.1

Список сообщений об ошибке

Сообщение	Возможная причина
Параметр должен иметь значение	Не хватает параметра оператора или значения переменной события в операторах up, down, timer
Не хватает закрывающей скобки	Не хватает скобки или синтаксическая ошибка. Например, вместо “==” написано “=”
Оператор down: неверный тип события	В операторе down используются события с именами, оканчивающимися только на ‘.REQ’ и ’.RESP’
Оператор up: неверный тип события	В операторе up используются события с именами, оканчивающимися только на ‘.IND’ и ’.CONF’

Продолжение табл. 3.1

Сообщение	Возможная причина
Оператор text: неверный тип номера панели	В качестве номера панели должно быть выражение типа “integer”
Оператор set: переменная "имя переменной" не объявлена. Используйте оператор declare для объявления переменных	Переменная не объявлена
Оператор declare: неизвестный тип “строка”	В операторе declare могут использоваться в качестве типа только “integer”, “string”, “buffer”
Оператор declare: переменная уже объявлена	Переменную можно объявлять только один раз. Лучше всего делать это в блоке инициализации
Оператор delete: неверный тип переменной	В операторе delete могут использоваться только строки
Оператор copy: неверный тип переменной	В операторе copy могут использоваться только строки
Оператор image: неверный тип номера панели или имени файла	В качестве номера панели должно быть выражение типа “integer”, а имени файла – выражение типа “string”
Оператор avi: неверный тип номера панели или имени файла	В качестве номера панели должно быть выражение типа “integer”, а имени файла выражение типа “string”
Оператор crc: переменная “имя переменной” должна быть буфером	Второй параметр оператора crc должен быть буфером
Оператор getaddress: неизвестная система	В качестве имени системы могут быть только “SystemA” и ”SystemB”
Нельзя найти метку "имя метки"	Отсутствует метка
Обработчик события 'имя события' пуст	В обработчике события должен присутствовать как минимум один символ
Адрес не найден	Неизвестный адрес системы
Неизвестный оператор: “строка”	Синтаксическая ошибка
"Строка" не является числовым значением	На месте указанной строки должно быть числовое значение. Скорее всего опечатка. Возможно синтаксическая ошибка, особенно если строка пустая

4. КОНЕЧНЫЕ АВТОМАТЫ

Спецификации протоколов на основе моделей конечных автоматов в настоящее время используются наиболее широко. Фор-

мально конечный автомат (КА) определяется шестеркой объектов КА={S, I, O, N, M, S_0 }, где S – конечное множество состояний; I – конечное множество входных событий; O – конечное множество выходных событий; N: $I \times S \rightarrow S$ – функция переходов; M: $I \times S \rightarrow O$ – функция выходов; S_0 – начальное состояние.

Функции N и M описывают поведение автомата, т.е. если в некотором текущем состоянии $s_i \in S$ на входе появляется сообщение (событие) $i \in I$, то функция переходов определяет новое состояние автомата $s_k \in S$, а функция выходов – выходное сообщение (событие) $o_i \in O$.

Для описания КА используются различные способы, однако наиболее широко распространены диаграммы состояний-переходов и таблицы решений (состояний-событий).

Диаграмма состояний-переходов представляет собой разновидность ориентированного графа, множество вершин которого соответствует множеству состояний, а множество дуг – множеству переходов. Дуги помечаются соответствующими входными и выходными событиями. Для различия входных и выходных событий последние подчеркиваются.

Таблица состояний-событий образована элементами – пересечениями столбцов, соответствующих входами $i \in I$, и строк, соответствующих состояниям $s \in S$. Элементам таблицы соответствуют пары вида (s, o), где s – новое состояние, а $o \in O$ – выход. В представлениях таблиц смысл строк и столбцов можно, очевидно, поменять местами.

5. РАЗРАБОТКА И РЕАЛИЗАЦИЯ ПРОТОКОЛА

Под событием на данном уровне здесь и далее понимается факт получения на нем либо сообщения от таймера, либо – сверху или снизу – примитива. Если примитив или сообщение от таймера параметризованы, то событию ставится в соответствие список параметров. Эти параметры передаются обработчику события в качестве локальных переменных. Обработчик события – это программная реализация части протокола данного уровня, которая начинает выполняться при возникновении данного события.

Для поддержки разработки протоколов настоятельно рекомендуется воспользоваться, хотя бы фрагментарно, автоматными мо-

делями в диаграммном или табличном виде (в диаграммном нагляднее, но, возможно, более громоздко). В терминах автоматных моделей, события – это множество **входных** событий соответствующего автомата. Его **выходные** события будут генерироваться обработчиками события. Обработчику события будет соответствовать в диаграммном представлении автомата дуга (несколько дуг). Например, на рис. 7.3 обработчик события на транспортном уровне N_CONNECT.CONF будет генерировать событие (примитив) T_CONNECT.CONF. В целом пример соответствия автоматной модели фазы установления транспортного соединения и реализации соответствующей части протокола на встроенном языке представлен в п. 8.2.

Для начала написания протокола необходимо выбрать нужные уровень и систему путем нажатия соответствующей кнопки на основной панели комплекса. После этого раскроется панель редактора событий данного уровня данной системы. В этой панели можно начать создавать новое событие путем нажатия кнопки “Добавить”. После ввода имени события можно приступить к написанию обработчика данного события на встроенном языке с помощью встроенного редактора, вызываемого путем двойного нажатия мышью на имени события. Во время работы со встроенным редактором можно пользоваться стандартными функциями редактирования (вставить из буфера, копировать в буфер, перенести в буфер, удалить, отменить последнее действие), доступными через меню “Правка” (“вставить”, “копировать”, “вырезать”, “удалить” соответственно), а также функциями поиска и замены, доступными через меню “Поиск” (“Найти” и “Заменить” соответственно).

После написания обработчика события окно редактора закрывается путем нажатия клавиши “Esc” и переходят к следующему событию. В процессе написания протокола можно переименовывать события, копировать в другой банк, копировать в другую систему и удалять с помощью редактора событий (см. п. 2.3).

Два банка (0 и 1) поваляют проводить две различных операции по сборке статистики. В банке 0 производится полномасштабное тестирование протокола, в банке 1 – упрощенное. Эти два банка независимы друг от друга, и в них возможно хранение двух различных вариантов протокола.

Для инициализации переменных, установки их начальных значений и т.п. перед началом работы протокола необходимо воспользоваться процедурой инициализации, доступной через кнопку “инициализация”. Эта процедура пишется на том же встроенном языке.

После написания протокола он запускается на выполнение путем нажатия кнопки “Запуск эмуляции” на основной панели комплекса. Работу протокола можно приостановить путем нажатия кнопки “Приостановить”, после чего можно продолжить работу протокола нажатием кнопки “Вернуться” или прекратить его выполнение нажатием кнопки “Завершить”. В случае приостановления работы протокола не рекомендуется производить какие-либо изменения в обработчики событий во избежание получения неверных результатов.

Для отладки протоколов можно воспользоваться встроенным отладчиком.

6. ОЦЕНКА СЕРВИСА, ПРЕДОСТАВЛЯЕМОГО ПРОТОКОЛОМ, И САМОГО ПРОТОКОЛА

Сервис, предоставляемый протоколом, оценивается исходя из значений пяти основных параметров, перечисленных ниже, для всех уровней и нескольких специальных – для конкретных уровней:

временная задержка при установлении соединения. Ею считается среднее время между выдачей примитива ?_CONNECT.REQ и получением примитива ?_CONNECT.IND;

вероятность ошибки при установлении соединения, представляется в виде отношения числа неудачных попыток соединения к общему числу выданных запросов ?_CONNECT.REQ;

транзитная задержка, т.е. время передачи блока данных протокола (БДП). Задается средней величиной;

вероятность искажения БДП, представленная в виде отношения числа неправильно переданных БДП к общему числу переданных БДП;

вероятность потери БДП данных, т.е. вычтенное из единицы отношение числа полученных примитивов ?_DATA.IND к числу переданных примитивов ?_DATA.REQ.

Сервис, предоставляемый протоколом, считается удовлетворяющим заданию, если его параметры, перечисленные выше, попадают в заданные интервалы. Вариант оценки сервиса, предоставляемого протоколом транспортного уровня, изображен на рис. 6.1.

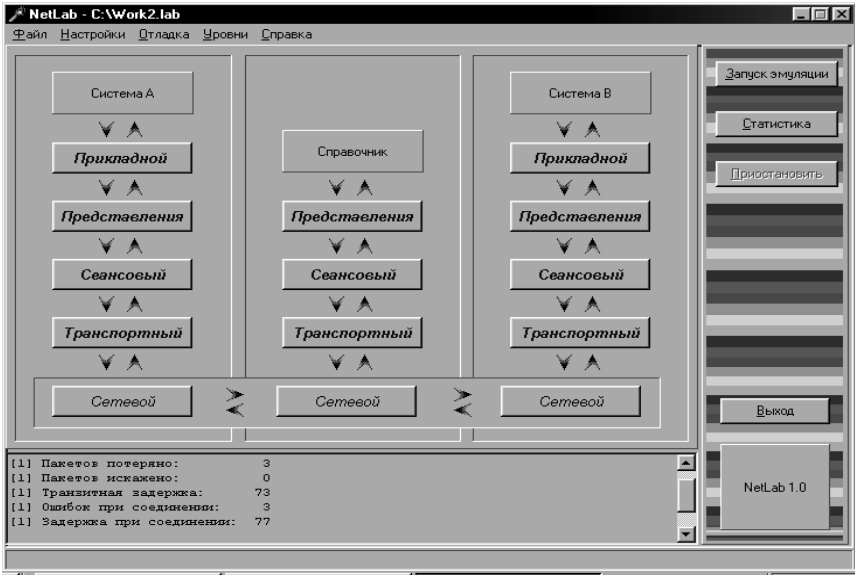


Рис. 6.1. Вариант оценки сервиса, предоставляемого протоколом транспортного уровня

Протокол оценивается двумя способами путем сбора статистики.

Первый способ, фактически оценивая “взвешенный” размер протокола, является статическим. Он определяет количество использованных в протоколе операторов встроенного языка. Веса, соответствующие каждому оператору, представлены в табл. 6.1. Таким образом, оценка считается по формуле: $M = \sum_{i \in O} W_i * N_i$,

где M – оценка протокола; W_i – вес оператора данного типа; N_i – количество операторов данного типа, используемых в протоколе; O – множество типов операторов встроенного языка.

Второй способ динамической оценки заключается в подсчете суммарного веса операторов встроенного языка, используемых во время работы протокола. Значения весов операторов получены эмпирическим путем, на основе анализа написанных протоколов.

Формула подсчета такая же, как и в первом случае. Веса, соответствующие каждому оператору, представлены в табл. 6.2.

Таблица 6.1

Веса, соответствующие каждому оператору, в статической оценке

Оператор	Вес	Оператор	Вес
up	10	queue	10
down	10	buffer	10
if	10	unbuffer	10
goto	10	set	10
timer	10	out	10
crc	10	untimer	10

Таблица 6.2

Веса, соответствующие каждому оператору, в динамической оценке

Оператор	Вес	Оператор	Вес
up	10·длина буфера	queue	5
down	10·длина буфера	buffer	10
if	15	unbuffer	10
goto	15	set	8
timer	5	out	25
crc	8	untimer	5

По результатам сбора статистики выставляется оценка (см. рис. 6.2).

Рис. 6.2. Пример оценки транспортного протокола

7. ТРАНСПОРТНЫЙ УРОВЕНЬ

7.1. Общие сведения

В соответствии с ЭМВОС транспортный уровень выполняет все необходимые процедуры для обеспечения надежной и эффективной прозрачной передачи данных из конца в конец от одного пользователя (сеансового объекта) до другого. Таким образом, все протоколы, определенные на транспортном уровне, функционируют в среде ВОС только между оконечными открытыми системами.

На транспортном уровне не выполняются функции маршрутизации и ретрансляции, так как сетевая служба обеспечивает прозрачную передачу данных между транспортными объектами даже при использовании нескольких промежуточных подсетей. Транспортный уровень скрывает от пользователей особенности сетевого сервиса.

В фазе установления соединения могут выполняться следующие функции:

- выбор сетевого соединения, наиболее полно удовлетворяющего требованиям сеансового объекта с учетом стоимости и качества обслуживания;
- решение о целесообразности мультиплексирования или расщепления транспортного соединения с целью оптимизации использования сетевых соединений;
- выбор оптимального размера транспортного БДП;
- выбор функций, которые будут задействованы в фазе передачи данных;
- отображение транспортных адресов в сетевые;
- обеспечение идентификации различных транспортных соединений между одной и той же парой транспортных ТДС;
- передача данных.

В фазе передачи данных осуществляется доведение транспортных БДС до сеансовых объектов-получателей по транспортному соединению передачей транспортных БДП. При этом могут быть задействованы следующие функции, использование каждой из которых согласуется в фазе установления соединения:

- упорядочение, сегментирование, блокирование и сцепление;
- мультиплексирование или расщепление;

– управление потоком;
– обнаружение ошибок; исправление ошибок;
– передача срочных данных; разграничение транспортных БДС;
– идентификация транспортных соединений.
– В фазе разъединения соединения могут выполняться функции:
– оповещения о причине разъединения;
– идентификации разъединяемого транспортного соединения;
– передачи данных.

При установлении транспортного соединения используются следующие примитивы:
– T-CONNECT-request;
– T-CONNECT-indication;
– T-CONNECT-response;
– T-CONNECT-confirmation.

Для разъединения уже установленного соединения, а также при отказе установить соединение используются пара примитивов с соответствующими параметрами:
– T-DISCONNECT-request;
– T-DISCONNECT-indication.

Такая же пара (T-DATA-) с единственным типом параметра – “данные пользователя” – используется при передаче данных по соединению. Длина Т-БДС не ограничена, так как на транспортном уровне есть функция разбиения Т-БДС на последовательность Т-БДП.

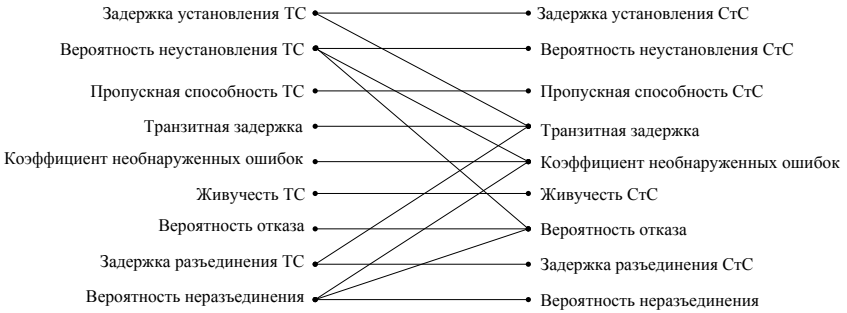
Для иллюстрации использования в целях описания соотношений примитивов на одном конце соединения диаграмм состояний-переходов приведен рис. 7.1 для рассматриваемого транспортного сервиса.

Связь параметров качества сервиса транспортного и сетевого уровней иллюстрирована на рис. 7.2.

Далее воспользуемся конечными автоматами для иллюстрации протокольной спецификации фрагмента фазы установления транспортного соединения. В табл. 7.1–7.4. приведены части списков имен элементов множеств входных и выходных событий, состояний автомата, разрешающих условий (предикатов).

Сокращение	Расшифровка
Т-СОЕДИНЕНИЕзапрос	Запрос ТС
Т-СОЕДИНЕНИЕиндикация	Индикация ТС
Т-СОЕДИНЕНИЕответ	Ответ на запрос ТС
Т-СОЕДИНЕНИЕподтвержение	Подтверждение ТС
Т-РАЗЪЕДИНЕНИЕзапрос	Запрос разрыва ТС
Т-РАЗЪЕДИНЕНИЕиндикация	Индикация разрыва ТС
Т-ДАННЫЕ	Передаваемые данные

Рис. 7.1. Диаграмма состояний переходов



Сокращение	Расшифровка
ТС	Транспортное соединение
СтС	Сетевое соединение

Рис. 7.2. Связь параметров качества сервиса транспортного и сетевого уровней

Таблица 7.1

Фрагменты списков имен элементов множеств входных событий

Имя	Интерфейс автомата	Смысл/значение
ТСДНзпр	ТСл-пользователь	Получен Т-СОЕДИНЕНИЕзапрос
ТСДНотв	Тсл-пользователь	Получен Т-СОЕДИНЕНИЕответ
СтСДНпдтв	Ст-поставщик	Получен Ст-СОЕДИНЕНИЕподтверждение
ЗС	Ст-поставщик	Получен Т-БДП “запрос на соединение”
ПС	Ст-поставщик	Получен Т-БДП “подтверждение соединение”
…	…	…

Таблица 7.2

Фрагменты списков имен элементов множеств состояний автомата

Имя	Смысл/значение
ЗАКРЫТО	Транспорное соединение закрыто
ОЖСтС	Ожидание установления сетевого соединения
ОЖПС	Ожидание Т-БДП “подтверждение соединения”
ОТКРЫТО	Транспортное соединение открыто и готово к передаче данных
ОЖТОТВ	Ожидание поступления Т-СОЕДИНЕНИЕответ от ТСл-пользователя
…	…

Таблица 7.3

Фрагменты списков имен элементов множеств выходных событий

Имя	Интерфейс автомата	Смысл/значение
ТСДНинд	ТСл-пользователь	Послан Т-СОЕДИНЕНИЕиндикация
ТСДНпдтв	Тсл-пользователь	Послан Т-СОЕДИНЕНИЕподтверждение
ТРЗДинд	Тсл-пользователь	Послан Т-РАЗЪЕДИНЕНИЕиндикация
СтСДНзпр	Ст-поставщик	Послан Т-СОЕДИНЕНИЕзапрос
ПС	Ст-поставщик	Послан Т-БДП “подтверждение соединение”
ЗС	Ст-поставщик	Послан Т-БДП “запрос на соединение”
ЗР	Ст-поставщик	Послан Т-БДП “запрос на разъединение”
СтРЗДзпр	Ст-поставщик	Послан Т-РАЗЪЕДИНЕНИЕзапрос
…	…	…

Таблица 7.4

Фрагменты списков имен элементов множеств предикатов

Имя	Смысл/значение
P0	Т-СОЕДИНЕНИЕзапрос, поступивший от ТСл-пользователя, неприемлем
P1	Получен неприемлемый Т-БДП “подтверждение соединения”
P2	Нет доступного сетевого соединения
P3	Сетевое соединение доступно и открыто
P4	Сетевое соединение доступно, открыто и находится в процессе использования
P5	Получен неприемлемый Т-БДП “запрос на соединение”
…	…

Предикату соответствует булевская переменная, зависящая от комбинации значений параметров, связанных с входным событием, и текущего состояния одной или нескольких автоматных переменных. Если предикатное условие не выполнено, а альтернатива не определена, то говорят, что имела место ошибка протокола. В этом случае выдается заранее определенное выходное событие, и автомат переходит в соответствующее состояние.

Таблица 7.5

Фрагмент таблицы состояний-переходов автомата отработки фазы установления транспортного соединения

Событие	Состояние					
	ЗАКРЫТО	ОЖТОТВ	P4: ОЖСтС	ОЖПС	ОТКРЫТО	…
ТСДНзпр	1	0	0	0	0	
ТСДНотв	0	2	0	0	0	
СтСДНпдтв	0	0	3	0	0	
ЗС	4	0	0	0	0	
ПС	0	0	0	5	0	
…						

0 = ТСДНзпр, СтРЗДзпр, ЗАКРЫТО (ошибочное условие)
1 = P0: ТРЗДинд, ЗАКРЫТО; P2: СтСДНзпр, ОЖСтС, P3: ЗС, ОЖПС
2 = ПС, ОТКРЫТО
3 = ЗС, ОЖПС
4 = P1: ЗР, ЗАКРЫТО; NOT P1: ТСДНинд, ОЖТОТВ
5 = NOT P5: ТСДНпдтв, ОТКРЫТО; P5: ТРЗДинд, СтРЗДзпр, ЗАКРЫТО

В табл. 7.5 приведен фрагмент соответствующей таблицы состояний-переходов автомата. Здесь он выписан в виде таблицы ссылок на список комбинаций состояний-событий. Некоторые из этих комбинаций обусловлены предикатами. В имеющемся фраг-

менте ни в одной из ситуаций никаких специальных действий, например запуска или остановки таймера, не производится, так что ссылочный список таких действий здесь пуст.

7.2. Фрагменты реализации протокола транспортного уровня на лабораторном комплексе

Простейший протокол транспортного уровня фактически является заглушкой и не выполняет основной задачи транспортного уровня – исправление и снижение количества ошибок.

Событие **T_CONNECT.REQ**:
down N_CONNECT.REQ address $address
Событие **N_CONNECT.IND**:
up T_CONNECT.IND
Событие **T_CONNECT.RESP**:
down N_CONNECT.RESP address $address
Событие **N_CONNECT.CONF**:
up T_CONNECT.CONF

Событие **T_DATA.REQ**:
down N_DATA.REQ userdata $userdata address $address
Событие **N_DATA.IND**:
up T_DATA.IND userdata $userdata
Событие **T_DISCONNECT.REQ**:
down N_DISCONNECT.REQ address $address
Событие **N_DISCONNECT.IND**:
up T_DISCONNECT.IND

В рамках лабораторного практикума основная задача транспортного уровня – снижение количества ошибок при установлении соединения и передаче данных при сохранении приемлемой скорости работы системы.

Основным методом борьбы с ошибками при установлении соединения является повторная попытка установлении соединения, а при передаче данных – повторная пересылка данных. Кроме того, при передаче данных необходимо вести контроль за искажением и появлением дубликатов БДП (блок данных протокола). Это делает-

ся с помощью подсчета контрольной суммы и нумерации БДП соответственно.

Простой метод борьбы с ошибками при установлении соединения (соединяется система А с системой В) представлен в табл. 7.6.

Таблица 7.6

Реализация метода борьбы с ошибками при установлении соединения

Система А		**Система В**	
Событие	**Код**	**Событие**	**Код**
T_CONNECT.REQ	set addrB $address	N_CONNECT.IND	up T_CONNECT.IND
	; послать запрос на соединение down N_CONNECT.REQ address $addrB		
	; поставить пять попыток соединения set connect_try 5		
	; запустить таймер на проверку установления соединения timer con_try CONNECT_TRY 200		
N_CONNECT.CONF	; соединение установлено – выключить таймер untimer $con_try	T_CONNECT.RESP	set addrA $address
	; подтверждение уже приходило? if $first_con == 1 exit		down N_CONNECT.RESP address $addrA
	; поставить флаг подтверждения установления соединения set first_con 1 ; послать подтверждение установления соединения up T_CONNECT.CONF		
RECONNECT	; послать запрос на соединение down N_CONNECT.REQ address $addrB		
	; запустить таймер на проверку установления соединения timer con_try RECONNECT 10 goto exit		

Продолжение табл. 7.6

Система А		Система В
Событие	**Код**	
CONNECT_TRY	; уменьшить количество попыток соединения set connect_try $connect_try-1	
	; попытки соединения кончились? if $connect_try == 0 terminate	
	; попытка соединения не удалась – разрыв соединения down N_DISCONNECT.REQ address $addrB	
	; запустить таймер на установление соединения timer con_try RECONNECT 10 goto exit	
	terminate: ; послать индикацию разрыва соединения up T_DISCONNECT.IND	
	exit:	

В этом примере разбирается простейший случай, исправляется только потеря запроса на соединение путем повторных попыток установления соединения при возникновении ошибки. Всего система пытается соединиться пять раз. Обратите внимание на то, что после каждой неудачной попытки соединения система посылает запрос на разрыв сетевого соединения, и только после этого производит следующую попытку. Это необходимо, так как потеря запроса могла произойти во время ответа системы В системе А, после чего система В будет находится в состоянии “соединено” и игнорировать все попытки соединения. Также стоит проверять является ли пришедшее подтверждение установления соединения повторным. Это необходимо, если таймер сработал до прихода подтверждения установления соединения, а потеря запроса не произошла.

На рис. 7.4 представлена диаграмма состояний конечного автомата соответствующего примеру, в табл. 7.7 – соответствующая им таблица состояний автомата.

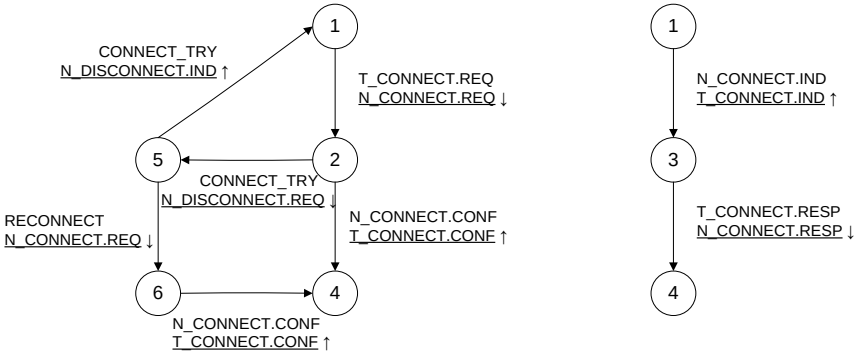


Рис.7.3. Диаграмма состояний конечного автомата фазы установления транспортного соединения

Таблица 7.7

Список состояний автомата

Состояние	Смысл / значение
1	Начальное состояние
2	Пришел запрос на установление транспортного соединения, отправлен запрос на установление сетевого соединения
3	Пришла индикация установления сетевого соединения, отправляем индикацию транспортного соединения
4	Транспортное соединение установлено
5	Произошла ошибка при установлении соединения, сработал таймер
6	Запрос на повторное установление соединения

При передаче данных потеря БДП обнаруживается и исправляется аналогичным механизмом подтверждения и передачи данных при их потере. Кроме того, для обнаружения искажения БДП необходимо вводить контрольную сумму, и при обнаружении несоответствия БДП контрольной сумме сбрасывать БДП. В случае правильно организованного механизма обнаружения потери БДП это приведет к повторной передаче БДП. Для обнаружения дубликатов БДП необходимо ввести нумерацию БДП и сбрасывать все просроченные БДП.

7.3. Формирование и распаковка БДП

Допустим нужен БДП: [номер БДП][данные].

Формирование

Оператор buffer создает переменную типа buffer из полей указанной длины. Если фактическая длина переданного значения ока-

зывается меньше указанной длины поля, то остаток отведенного поля заполняется нулями. Если суммарная длина полей оказывается больше размера буфера, то происходит ошибка времени выполнения.

Синтаксис:

buffer <имя переменной-буфера> <длина буфера> { <значение поля в буфере> <длина поля> }

В рамках рассматриваемой задачи:

buffer buf sizeof(data)+4 $num_pac 4 $data sizeof(data),

где

- num_pac – переменная с номером БДП;
- data – буфер с данными;
- buf – искомый БДП.

Распаковка (в другой системе и событии)

Оператор unbuffer разбивает переменную типа buffer на поля указанной длины. Если при этом из буфера необходимо извлечь строковую переменную, то ее тип должен быть установлен до выполнения оператора unbuffer.

Синтаксис:

unbuffer <имя переменной-буфера> { <имя переменной> <длина поля> }

В рамках рассматриваемой задачи:

```
; инициализация номера БДП
set num_pac 0
; инициализация буфера
buffer 20 data 0 4
unbuffer buf num_pac 4 data sizeof(buf)-4,
```

где

- num_pac – переменная, в которой будет номер принятого БДП;
- data – буфер, в который распакуют данные;
- buf – БДП, который распаковывают.

Теперь рассмотрим случай, когда вероятность искажения пакета (см. показатели качества сервиса сетевого уровня) не равна 0, т.е. необходимо проверять контрольную сумму БДП:

БДП: [crc(4байта)][номер БДП (4байта)][данные]

Формирование

Оператор crc подсчитывает контрольную сумму буфера и заносит ее в переменную с заданным именем. Если тип переменной не совпадает с типом «число», возникает ошибка времени выполнения.

Синтаксис:

crc < имя переменной-числа > < имя переменной-буфера >

В рамках рассматриваемой задачи:

```
buffer buf1 sizeof(data)+4 $num_pac 4 $data sizeof(data)
crc crc $buf1
buffer buf sizeof(data)+8 $crc 4 $buf1 sizeof(data)+4
```

где

- num_pac – переменная с номером БДП;
- data – буфер с данными;
- buf – искомый БДП;
- buf1 – промежуточный буфер;
- crc – контрольная сумма.

Распаковка (в другой системе и событии)

{num_pac – переменная в которой будет номер принятого БДП; data – буфер в который распакуют данные; buf – БДП, который распаковывают; crc – контрольная сумма в БДП; crc1 – контрольная сумма заново подсчитанная; buf1 – промежуточный буфер}

```
set num_pac 0
; инициализация контрольной суммы
set crc1 0
buffer 20 data 0 4
buffer 20 buf1 0 4
; распаковка БДП в промежуточный буфер
unbuffer buf crc1 4 buf1 sizeof(buf)-4
; подсчет контрольной суммы
crc crc $buf1
; проверка на искажение
if $crc != $crc1 corrupted
; пакет не искажен, распаковка БДП
unbuffer buf1 num_pac 4 data sizeof(data)-8
...
goto exit
```

; БДП искажен
:corrupted:
...
goto exit
...
exit:

На рис. 7.4–7.5 приведены диаграммы состояний конечного автомата, иллюстрирующая работу транспортного уровня, а в табл. 7.8–7.10 – списки входных, выходных событий и состояний автомата соответственно.

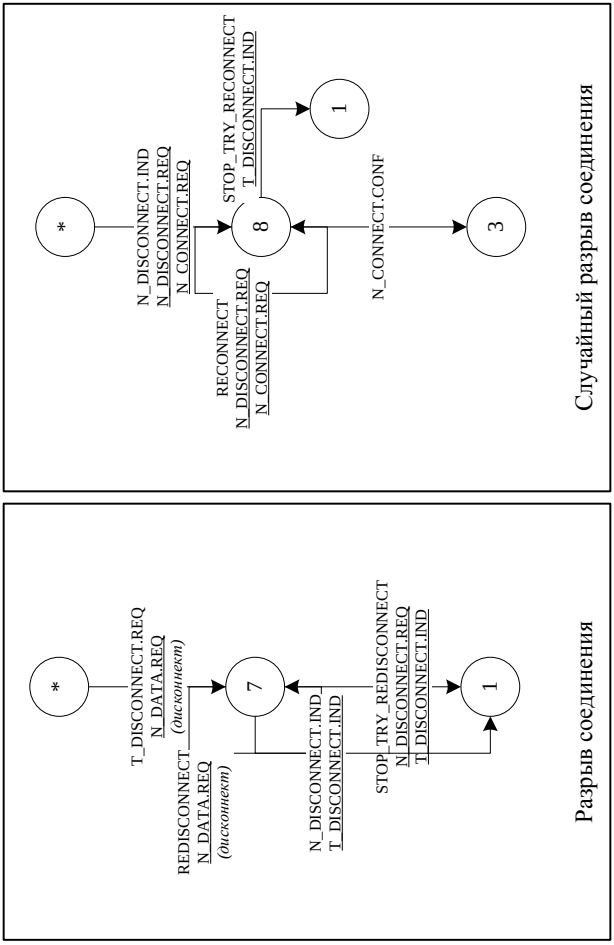


Рис. 7.4. Диаграмма состояний конечного автомата

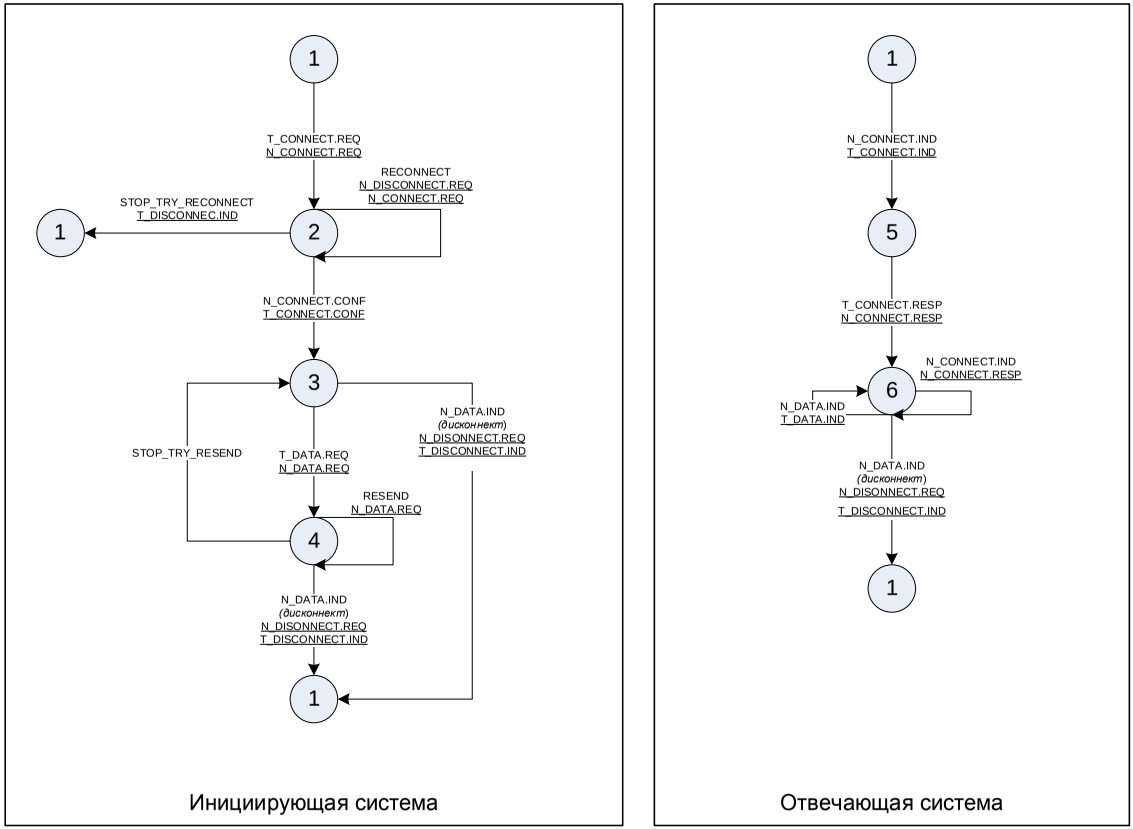


Рис. 7.5. Диаграмма состояний конечного автомата

Таблица 7.8

Список входных событий

Входное событие	Смысл/значение
T_CONNECT.REQ	Запрос на установление транспортного соединения
RECONNECT	Попытка повторного соединения транспортного уровня
STOP_TRY_RECONNECT	Слишком много попыток соединения. Остановка
N_CONNECT.CONF	Подтверждение установления сетевого соединения
T_DATA.REQ	Запрос на передачу транспортных данных
RESEND	Повторная отправка данных
STOP_TRY_RESEND	Слишком много попыток отправки данных
N_DATA.IND *(дисконнект)*	Данные для запроса на разрыв соединения
N_CONNECT.IND	Индикация установления сетевого соединения
T_CONNECT.RESP	Ответ установления транспортного соединения
N_DATA.IND	Получены данные
T_DISCONNECT.REQ	Запрос на разрыв транспортного соединения
REDISCONNECT	Попытка повторного разрыва транспортного соединения
N_DISCONNECT.IND	Индикация разрыва сетевого соединения
STOP_TRY_REDISCONNECT	Слишком много попыток разрыва соединения

Таблица 7.9

Список выходных событий

Входное событие	Смысл/значение
N_CONNECT.REQ	Запрос соединения сетевого уровня
N_DISCONNECT.REQ	Запрос разрыва сетевого соединения
T_DISCONNECT.IND	Индикация разрыва транспортного соединения
T_CONNECT.CONF	Подтверждение установления сетевого соединения
N_DATA.REQ	Передача сетевых данных
T_CONNECT.IND	Индикация установления транспортного соединения
N_CONNECT.RESP	Ответ установления сетевого соединения
T_DATA.IND	Индикация получения транспортных данных
N_DATA.REQ *(дисконнект)*	Передача сетевых данных, для запроса на разрыв

Таблица 7.10

Список состояний автомата

Состояние	Смысл / значение
1	Начальное состояние
2	Пришел запрос на установление транспортного соединения
3	Пришло подтверждение установления сетевого соединения
4	Пришел запрос на пересылку данных транспортного уровня
5	Пришла индикация установления сетевого соединения
6	Получен ответ на установление транспортного соединения
7	Пришел запрос на разрыв соединения
8	Получен случайный разрыв связи

Все вышесказанное относится к случаю, когда применяются сетевые службы с установлением соединения. При использовании сетевой службы без установления соединения отличия сводятся к следующему:

– нет необходимости устанавливать и разрывать сетевое соединение;
– в БДП транспортного уровня вводится поле их назначения (данные пользователя, системная информация), т. е. формат БДП будет следующий:

Контрольная сумма	Номер БДП	Назначение	Данные

Методы исправления ошибок передачи данных остаются теми же, что и в случае использования сетевой службы с установлением соединения (нумерация и подсчет контрольной суммы БДП). Кроме того, необходимо ввести подтверждение приема БДП в целях обнаружения потери БДП. Такое подтверждение необходимо и для случая при использовании сетевой службы с установлением соединения.

Рассмотрим пример реализации протокола транспортного уровня без установления соединения.

Таблица 7.11

Реализация протокола транспортного уровня без установления соединения

Система А		**Система В**	
Событие	Код	Событие	Код
T_CONNECT.REQ	;Формируем пакет для соединения (длина=1) buffervar Outpac 1	N_DATAGRAM.IND	;Делим по длине if sizeof(userdata) == 1 ConInd if sizeof(userdata) == 2 ConConf if sizeof(userdata) == 3 PacCor if sizeof(userdata) == 4 DConReq if sizeof(userdata) >= 8 DataInd goto corrupted
	;Шлем его senddown address $address userdata $Outpac N_DATAGRAM.REQ		ConInd: up T_CONNECT.IND set Addr $address goto exit
	;Сохраняем адрес и устанавливаем таймер		ConConf: up T_CONNECT.CONF

	$address var Addr timeevent Timer0 CONTIMER 100 address $address		untimer $Timer0 goto exit
N_CONNECT.CONF	;соединение установлено – выключить таймер untimer $con_try		PacCor: down address $Addr userdata $Datapac N_DATAGRAM.REQ goto EXIT
	; подтверждение уже приходило? if $first_con == 1 exit		DConReq: up T_DISCONNECT.IND goto EXIT
	; поставить флаг подтверждения установления соединения set first_con 1 ; послать подтверждение установления соединения up T_CONNECT.CONF		DataInd: ;Распаковываем и проверяем CRC unbuffer userdata Incrc 4 Buf sizeof(userdata)-4 crc Calccrc $Buf if $Calccrc != $Incrc corrupted ;Вычленяем номер, идентификатор и данные unbuffervar Buf Num 4 Data sizeof(Buf)-4 ;Проверяем номер принятого пакета if $Num <= $Incnum dublicate set Incnum $Num up T_DATA.IND userdata $Data goto exit
CONTIMER	; увеличиваем количество попыток соединения set Contry $Contry + 1		dublicate: goto exit
	; попытки соединения кончились? if $Contry > 3 over		
	; попытка соединения не удалась – разрыв соединения down N_DISCONNECT.REQ address $addrB		corrupted: ;Формируем пакет для сигнала об искажении(длина=3) buffer Outpac 3 ;Посылаем down N_DATAGRAM.REQ address $Addr userdata $Outpac goto exit
	;Формируем пакет для коннекта (id=1) buffer Outpac 1		
	;Шлем его down N_DATAGRAM.REQ address $address userdata $Outpac		

	;Устанавливаем таймер timer Timer0 CONTIMER 100 goto exit		exit:
	over: ;Формируем пакет для разрыва соединения (длина=4) buffer Outpac 4 ;Шлем его down N_DATAGRAM.REQ address $Addr userdata $Outpac up T_DISCONNECT.IND	T_CONNECT.RESP	set Addr $address
			;Формируем пакет для подтверждения (длина=2) buffer Outpac 2
			;Посылаем down N_DATAGRAM.REQ address $Addr userdata $Outpac
	exit:		
T_DATA.REQ	; увеличиваем номер исходящего пакета set Outnum $Outnum+1		
	; формируем исходящий БДП buffer Buf 4+sizeof(userdata) $Outnum 4 $userdata sizeof(userdata) crc Outcrc $Buf buffer Datapac 8+sizeof(userdata) $Outcrc 4 $Buf 4+sizeof(userdata)		
	;Шлем его down N_DATAGRAM.REQ address $address userdata $Datapac		
T_DISCONNECT.REQ	;Формируем пакет для разрыва соединения (длина=4) buffer Outpac 4		
	;Шлем его down N_DATAGRAM.REQ address $address userdata $Outpac		
	up T_DISCONNECT.IND		

Контрольные вопросы

1. Верно ли, что транспортный сервис с соединением нуждается в поддержке аналогичного сетевого сервиса? Ответ обоснуйте. Приведите примеры сочетания режимов с установлением и без установления в рамках от канального уровня до уровня представления ЭМВОС.

2. Опишите услуги, предоставляемые сетевым уровнем (для сетевых служб с установлением и без установления соединения).
3. Приведите последовательность действий протокола во время выполнения следующих услуг:
   а) установления соединения транспортного уровня;
   б) разрыва соединения транспортного уровня;
   в) передачи данных транспортного уровня.
4. Охарактеризуйте связь параметров качества сервиса транспортного и сетевого уровней с раскрытием смысла этих параметров.
5. Приведите в виде диаграммы состояний-переходов автоматную модель следующих фаз:
   а) установления соединения транспортного уровня;
   б) передачи данных транспортного уровня;
   в) разрыва соединения транспортного уровня.
6. За счет чего в контексте эталонной модели ВОС обеспечивается прозрачность передачи данных на транспортном уровне?

8. СЕАНСОВЫЙ УРОВЕНЬ

8.1. Общие положения

Сеансовый уровень с установлением соединения обеспечивает средства организации и синхронизации обмена данными между пользователями. Функции сеансового уровня сильно связаны с его сервисом, так как собственные, т.е. не инициированные с верхнего уровня, действия на сеансовом уровне практически отсутствуют. В целом сеансовый уровень (с помощью служб, обеспечиваемых уровнем представления) предоставляет прикладным объектам следующие средства равноправного, синхронизированного, структурированного взаимодействия:

- установления сеансового соединения, синхронизированного обмена данными, упорядоченного и безусловного завершения сеансового соединения;
- согласования использования маркеров обмена данными, синхронизации и завершения взаимодействия, а также фиксации маркеров на одной из взаимодействующих сторон;
- установления точек синхронизации внутри диалога;

– выполнения ресинхронизации сеансового соединения, т.е. возврата к согласованной прикладными объектами точке синхронизации;
– прерывания диалога и его возобновления с заранее организованной точки синхронизации.

Сеансовая служба содержит три фазы:
– установления сеансового соединения;
– передачи данных;
– освобождения сеансового соединения.

В фазе установления соединения предусмотрена одна услуга – S-CONNECT, которая позволяет согласовать его параметры, распределить маркеры, выбрать начальный номер точки синхронизации.

В фазе передачи данных осуществляется синхронизированный обмен данными между двумя пользователями сеансовой службы. Для передачи данных используется услуга S-DATA, для управления расположением маркеров услуги – S-PLEASE-TOKENS, S-GIVE-TOKENS и для фиксации точек синхронизации и ресинхронизации услуги – S-SYNC-MINOR, S-SYNC-MAJOR, S-RESYNCHRONIZE.

Фаза завершения сеансового соединения характеризуется тремя другими услугами:

S-RELEASE – упорядоченное завершение (может быть использован маркер завершения TR);

S-P-ABORT и S-U-ABORT – безусловное завершение.

Некоторые услуги могут инициироваться и поставщиком (Пс, provider, P), и пользователем (Пл, user, U), например, услуги безусловного завершения сеансового соединения – соответственно S-P-ABORT и S-U-ABORT.

Качество сеансового сервиса определяет параметры сеансового соединения, которые касаются исключительно поставщика сеансовой службы. Параметры делятся на два типа:
– первого (согласуются в ходе установления сеансового соединения), к ним относится защита сеансового соединения;
– второго (не согласуются в фазе установления сеансового соединения, но их значение известно либо изначально, либо в результате проведения предварительных измерений). К ним относятся:

– задержка установления сеансового соединения;
– вероятность отказа в установлении сеансового соединения;
– вероятность ошибки передачи;
– задержка завершения сеансового соединения;
– вероятность ошибки завершения сеансового соединения.

После установления соединения СнСл-пользователи не могут модифицировать выбранные параметры качества сеансового сервиса в период существования этого соединения. Поставщик сеансовой службы не информирует СнСл-пользователей о каких-либо изменениях параметров качества сеансового уровня.

Надо заметить, что формат БДП меняется в зависимости от параметров, согласуемых в ходе установления сеансового соединения. Так, при отключении защиты исчезает поле ответственное за защиту.

Поставщик сеансовой службы для обеспечения сеансового соединения использует транспортное соединение. В каждый момент времени существует однозначное соответствие между сеансовым и транспортным соединениями, каждому сеансовому соединению соответствует единственное транспортное соединение. Однако время их жизни может отличаться: одно транспортное соединение может поддерживать несколько последовательных сеансовых соединений, несколько последовательных транспортных соединений могут поддерживать одно сеансовое.

Транспортное соединение, об использовании которого шла речь, должно быть установлено путем инициации поставщиком сеансовой службы услуги T-CONNECT. После этого при благополучном развитии ситуации корреспондирующие сеансовые объекты, пользуясь услугами T-DATA, обмениваются по этому транспортному соединению данными, которые необходимы для установления сеансового соединения.

Воспользуемся здесь формализмом конечных автоматов для протокольной спецификации, как и в случае транспортного уровня, фрагмента фазы установления сеансового соединения. В иллюстративных целях была избрана фаза установления соединения для основного комбинированного подмножества. Ему соответствует набор из 11 типов Сн-БДП, формируемых Сн-объектом.

В табл. 8.1–8.6 приведены части списков имен элементов множеств входных и выходных событий, состояний автомата, предикатов и специальных действий.

Таблица 8.1

Фрагменты списков имен элементов множеств входных событий

Имя	Интерфейс автомата	Смысл/значение
СнСДННзпр	СнСл-пользователь	Получен Сн-СОЕДИНЕНИЕзапрос
СнСДННотв(+)	СнСл-пользователь	Получен Сн-СОЕДИНЕНИЕответ (принято)
СнСДННотв(-)	СнСл-пользователь	Получен Сн-СОЕДИНЕНИЕответ (отвергнуто)
ТСДНинд	ТСл-поставщик	Получен Т-СОЕДИНЕНИЕиндикация
ТСДНпдтв	ТСл-поставщик	Получен Т-СОЕДИНЕНИЕподтверждение
СД	ТСл-поставщик	Получен Сн-БДП “соединение”
ПН	ТСл-поставщик	Получен Сн-БДП “принято”
ОТ	ТСл-поставщик	Получен Сн-БДП “отказано”
…	…	…
СнТМР	Таймер	Истек таймер СнТМР

Таблица 8.2

Фрагменты списков имен элементов множеств состояний автомата

Имя	Смысл/значение
Состояние 01	Бездействие; транспортное соединение не установлено
Состояние 01 (B)	Ожидание Т-СОЕДИНЕНИЕподтверждение
Состояние 01 (C)	Бездействие; транспортное соединение установлено
Состояние 02	Ожидание Сн-БДП “принято”
Состояние 08	Ожидание Сн-СОЕДИНЕНИЕответ
Состояние 016	Ожидание Сн-СОЕДИНЕНИЕиндикация
…	…
Состояние 713	Передача данных

Таблица 8.3

Фрагменты списков имен элементов множеств предикатов

Имя	Смысл/значение
Состояние P1	T-CONNECT инициирована данным Сн-объектом
…	…

Таблица 8.4

Фрагменты списков имен элементов множеств специальных действий

Имя	Смысл/значение
[1]	Положить P1 равным “ложь”
[2]	Положить P1 равным “истина”
[3]	Остановить таймер СнТМР
[4]	Запустить таймер СнТМР
…	…

Таблица 8.5

Фрагменты списков имен элементов множеств выходных событий

Имя	Интерфейс автомата	Смысл/значение
СнСДННинд	СнСл-пользователь	Послан Сн-СОЕДИНЕНИЕиндикация
СнСДННпдтв(+)	СнСл-пользователь	Послан Сн-СОЕДИНЕНИЕ подтверждение (принято)
СнСДННпдтв(-)	СнСл-пользователь	Послан Сн-СОЕДИНЕНИЕ подтверждение (отвергнуто)
СнПсПКРинд	СнСл-пользователь	Послан Сн-Пс-ПРЕКРАЩЕНИЕ индикация
ТСДНзпр	ТСл-поставщик	Послан Т-СОЕДИНЕНИЕзапрос
ТСДНотв	ТСл-поставщик	Послан Т-СОЕДИНЕНИЕответ
Сд	ТСл-поставщик	Был послан Сн-БДП “соединение”
ПН	ТСл-поставщик	Был послан Сн-БДП “принято”
ОТ	ТСл-поставщик	Был послан Сн-БДП “отказано”
ПКР	ТСл-поставщик	Был послан Сн-БДП “прекращение”
…	…	…

Таблица 8.6

Фрагмент таблицы состояний-переходов автомата отработки фазы установления сеансового соединения

Событие	Состояние							
	Сост 01	Сост 01 В	Сост 01 С	Сост 02	Сост 08	Сост 016	…	Сост 713
СнСДНзпр	1	0	2	0	0	0		
СнСДНотв(+)	0	0	0	0	3	0		
СнСДНотв(+)	0	0	0	0	4	0		
ТСДНинд	5	0	0	0	0	0		
ТСДНпдтв	0	6	0	0	0	0		
СД	0	0	8	0	0	7		
ПН	0	0	7	9	0	12		
ОТ	0	0	7	10	0	12		
…								
СнТМР	0	0	0	0	0	11		

0 = СнПсПКРинд, ПКРинд, Сост 01
1 = ТСДНзпр, [2], Сост 01В
2 = Р1: СД, Сост 02
3 = ПН, Сост 713
4 = ОТ, [4], Сост 16
5 = ТСДНотв,[1], Сост 01С
6 = СД, Сост 02
7 = ТРЗДзпр, [3], Сос 01
8 = Р1: ТРЗДзпр, Сост 01
NOT Р1: СдСДНинд, Сост 08
9 = СнСДНпдтв (+), Сост 713
10 = СнСДНпдтв (-), ТРЗДзпр, Сост01
11 = ОТ, [4], Сост 16
12 = Сост 16

8.2. Службы сеансового уровня

Так как транспортный и сеансовый уровни совместно образуют функцию, осуществляемую в интересах уровня представления, то результатом действия большинства из изображенных примитивов служб сеансового уровня оказываются соответствующие блоки данных сеансового протокола (СнБДП), формируемых на основании параметров, связанных с примитивами (рис. 8.1). На рис. 8.2 й последовательности взаимодействия основных служб обеспечиваемых сеансовым уровнем.

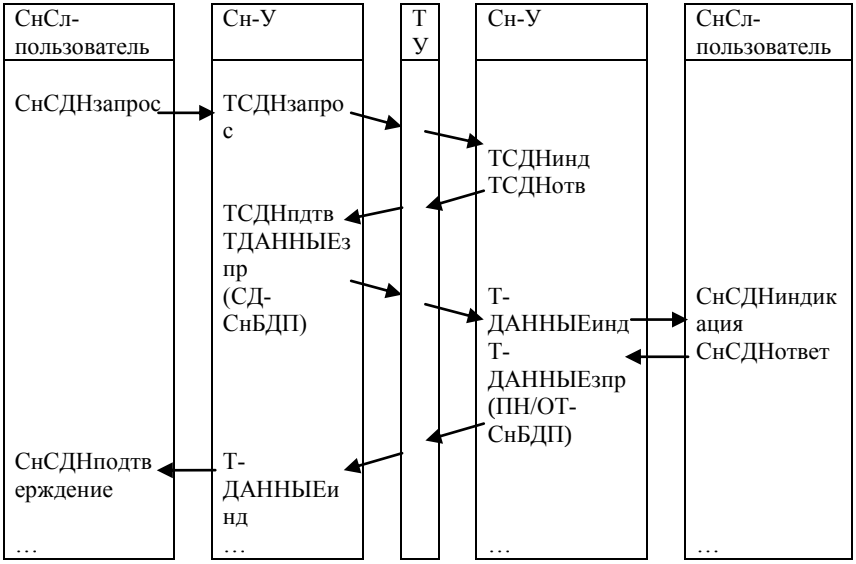


Рис. 8.1. Основные службы сеансового уровня

СнБДП	Посланы в ответ на:
СОЕДИНЕНИЕ, СД	Сн.СОЕДИНЕНИЕ.запрос
ПРИНЯТО, ПН	Сн.СОЕДИНЕНИЕ.ответ(успешный)
ОТКАЗАНО, ОТ	Сн.СОЕДИНЕНИЕ.ответ(безуспешный)
ДАННЫЕ, ДН	Сн.ДАННЫЕ.запрос
ЗАПРОС МАРКЕРА, ЗМ	Сн.ЗАПРОС_МАРКЕРА. запрос
ПЕРЕДАЧА МАРКЕРАБ, ПМ	Сн.ПЕРЕДАЧА_МАРКЕРА. запрос
КОНЕЦ, КН	Сн.ОСВОБОЖДЕНИЕ.запрос
РАЗЪЕДИНЕНИЕ, РЗ	Сн.ОСВОБОЖДЕНИЕ.ответ
ПРЕКРАЩЕНИЕ, ПКР	Сн.Пл_ПРЕКРАЩЕНИЕ.запрос
ПРЕКРАЩЕНИЕ ПРИНЯТО, ППН	Получение ПКР

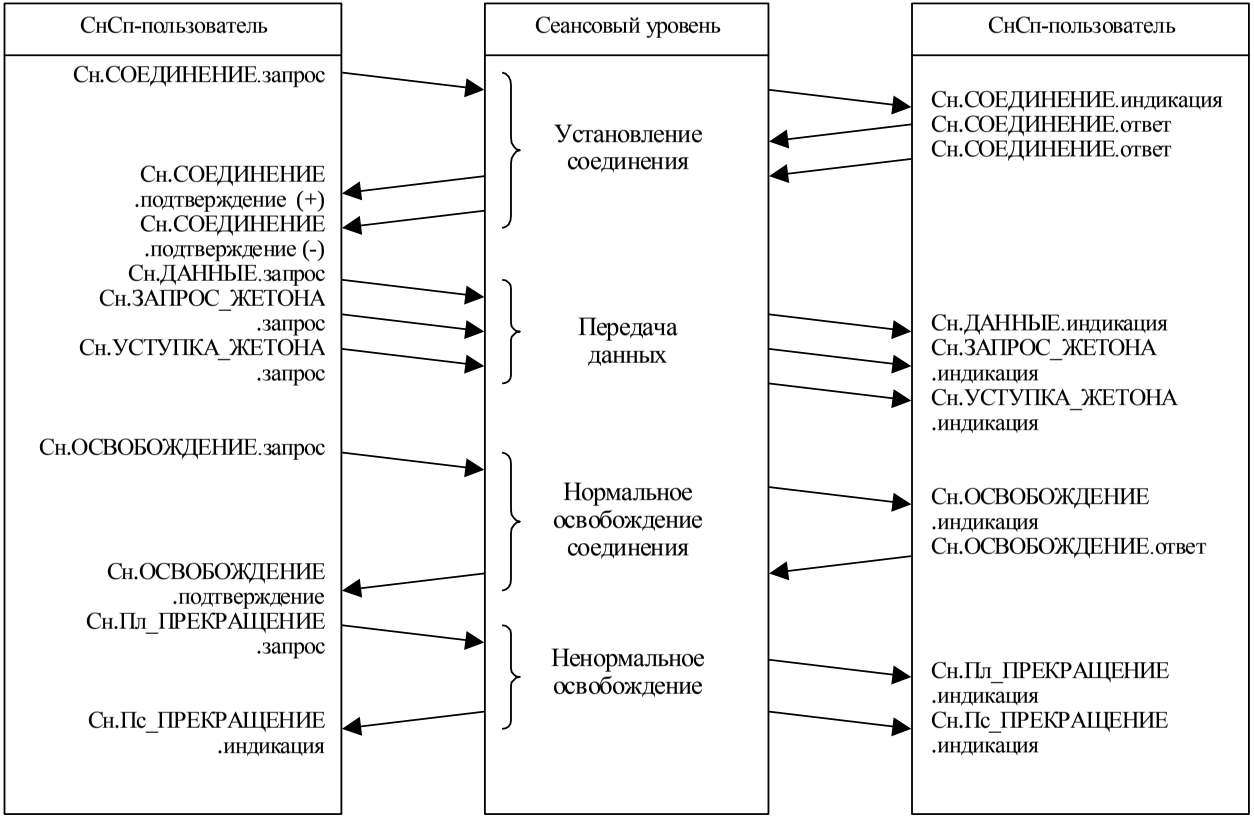


Рис. 8.2. Транспортные службы, связанные с сеансовым уровнем

8.3. Фрагменты реализации протокола сеансового уровня на лабораторном комплексе

Основной задачей сеансового уровня (в контексте данной части ДЗ) является предоставление прикладным объектам (с помощью служб, обеспечиваемых уровнем представления) следующих средств взаимодействия:

- установления сеансового соединения, синхронизированного обмена данными, упорядоченного и безусловного завершения сеансового соединения;
- согласования использования маркеров обмена данными, синхронизации и завершения взаимодействия, а также фиксации маркеров на одной из взаимодействующих сторон;
- установления точек синхронизации внутри диалога;
- выполнения ресинхронизации сеансового соединения, т.е. возврата к согласованной прикладными объектами точке синхронизации.

Синхронизация необходима для проверки правильности передачи данных – передачи в нужном порядке, без потери и дублирования БДП. Услуга ресинхронизации вызывается в случае нарушения синхронизации или проблем с маркерами (потеря или дублирование). В случае отсутствия услуги ресинхронизации сеансовый протокол фактически может работать до первой ошибки. Так, при потере маркера протокол фактически зависнет, при дублировании маркеров произойдет потеря синхронизации.

Сеансовая служба содержит три фазы:

- установления сеансового соединения;
- передачи данных;
- освобождения сеансового соединения.

В фазе установления соединения предусмотрена одна услуга, S-CONNECT.

Пример простейшей организации услуги S-CONNECT без каких-либо параметров:

Событие S_CONNECT.REQ:
down T_CONNECT.REQ address $sysB
Событие T_CONNECT.IND:
up S_CONNECT.IND

Событие S_CONNECT.RESP:

down T_CONNECT.RESP address $sysA

Событие T_CONNECT.CONF:

up S_CONNECT.CONF

где:

- sysB - адрес вызываемой системы,
- sysA – адрес вызывающей системы.

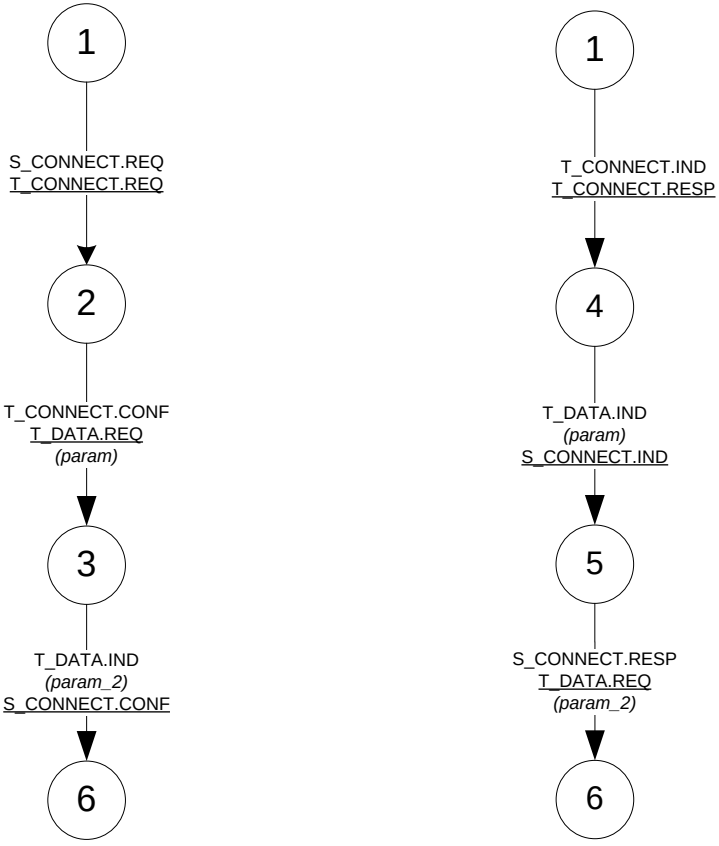


Рис. 8.3. Диаграмма состояний конечного автомата фазы установления сеансового соединения

При установлении сеансового соединения с некоторыми параметрами (например, использование защиты от ошибок или начальная расстановка маркеров) после установления транспортного соединения необходимо согласовать (передать) эти параметры, ис-

пользуя примитивы передачи данных, и только после этого индицировать установление соединения на вышележащий уровень. Диаграмма состояний соответствующего конечного автомата представлена на рис. 8.3, а в табл. 8.7–8.9 – списки входных, выходных событий и состояний автомата соответственно. В целях увеличения надежности работы протоколов рекомендуется дополнить данную схему средствами исправления ошибок (при установлении соединения), аналогичными средствам транспортного уровня. В данном примере система А является инициирующей соединение, система В отвечает.

Таблица 8.7

Список входных событий

Входное событие	Смысл / значение
S_CON.REQ	Получен С_СОЕДИНЕНИЕзапрос
T_CON.CONF	Получен Т_СОЕДИНЕНИЕподтверждение
T_DATA.IND2	Получен Т_ДАННЫЕиндикация – параметры согласованы
T_DATA.IND1	Получен Т_ДАННЫЕиндикация – список параметров
T_CON.IND	Получен Т_СОЕДИНЕНИЕиндикация

Таблица 8.8

Список выходных событий

Выходное событие	Смысл / значение
T_CON.REQ	Послан Т_СОЕДИНЕНИЕзапрос
T_DATA.REQ1	Послан Т_ДАТАзапрос – список параметров
S_CON.CONF	Послан С_СОЕДИНЕНИЕподтверждение
T_CON.RESP	Послан Т_СОЕДИНЕНИЕответ
T_DATA.REQ2	Послан Т_ДАТАзапрос– параметры согласованы
S_CON.IND	Послан С_СОЕДИНЕНИЕиндикация

Таблица 8.9

Список состояний автомата

Состояние	Смысл / значение
1	Начальное состояние
2	Пришел запрос на установление сеансового соединения, послан запрос на установление транспортного соединения
3	Пришло подтверждение установления транспортного соединения, начинаем переговоры
4	Пришла индикация установления транспортного соединения, отправляем ответ транспортного соединения
5	Пришли параметры, отправляем их наверх
6	Сеансовое соединение установлено

В фазе передачи данных осуществляется синхронизированный обмен данными между двумя пользователями сеансовой службы.

Для передачи данных используется услуга S-DATA, для управления расположением маркеров – услуги S-GIVE-TOKENS, S-PLEASE-TOKENS и для фиксации точек синхронизации и ресинхронизации – услуги S-SYNC-MINOR, S-SYNC-MAJOR, S-RESYNCHRONIZE.

В простейшем случае во время фазы передачи данных для собственно передачи данных, а также для синхронизации и управления маркерами, достаточно пользоваться БДП следующего типа:

Идентификатор	Данные

где идентификатор определяет тип БДПа, а данные – соответствующие данные. В этом случае услуга S-DATA выглядит следующим образом:

Обработка события S_DATA.REQ:

```
;если маркера DK нет, то перейти к gettoken
if $DK!=1 gettoken
;если очередь посылаемых БДП пуста, то послать, иначе поместить в очередь
if qcount(outgoingq)==0 send queue outgoingq $userdata
goto exit

send:
;составить  и отослать БДП с идентификатором
3buffer buf1 8 3 4 $userdata 4
down T_DATA.REQ userdata $buf1
;увеличить число прошедших БДП
set SP $SP+1
; поместить данные в очередь неподтвержденных данных
queue senddata $userdata
goto exit

gettoken:
;запустить запрос на получение маркера
timer S_PLEASE_TOKENS.REQ 0 token 1
;поместить данные в очередь        исходящих данных
queue outgoingq $userdata
goto exit

exit:
```

На удаленном конце сеансового соединения соответствующее событие (на сеансовом уровне) происходит при получении с транспортного уровня примитива T_DATA.IND. Его обработчик может выглядеть в данном случае так:

Обработка события T_DATA.IND:

```
set ID 4
buffer data 4 4 4
unbuffer userdata ID 4 data 4
;определить тип БДП по идентификатору и перейти к соответствующему обработчику
if $ID == 1 major_sync_ind
if $ID == 2 minor_sync_ind
if $ID == 3 data_ind
if $ID == 4 major_sync_conf
if $ID == 5 resync_ind
if $ID == 6 resync_conf
if $ID == 7 give_toc_ind
if $ID == 8 please_toc_ind
if $ID == 9 release_ind
if $ID == 10 u_abort_ind
goto exit

major_sync_ind:
timer t0 S_SYNC_MAJOR.IND 0 syncpoint $data
goto exit

minor_sync_ind:
timer t0 S_SYNC_MINOR.IND 0 syncpoint $data
goto exit

data_ind:
;увеличить число пришедших БДП
set SP $SP+1
;поместить данные в очередь неподтвержденных данных
queue ingoingq $data
;если принято SN БДП, послать запрос на основную синхронизацию
if $SP % $SN == 0 major_sync
```

```
    goto exit

    major_sync:
    timer t0 S_SYNC_MAJOR.REQ 0
    goto exit

    major_sync_conf:
    timer t0 S_SYNC_MAJOR.CONF 0 syncpoint $data
    goto exit

    resync_ind:
    timer t0 S_RESYNC.IND 0 syncpoint $data
    goto exit

    resync_conf:
    timer t0 S_RESYNC.CONF 0 syncpoint $data
    goto exit

    give_toc_ind:
    timer t0 S_GIVE_TOKEN.IND 0 syncpoint $data
    goto exit

    please_toc_ind:
    timer t0 S_PLEASE_TOKEN.IND 0 syncpoint $data
    goto exit

    release_ind:
    timer t0 S_RELEASE.IND 0 syncpoint $data
    goto exit

    u_abort_ind:
    timer t0 S_U_ABORT.IND 0 syncpoint $data
    goto exit

    exit:
```

где

DK – маркер данных (1 – маркер назначен, 0 – маркер не назначен);

ID – идентификатор БДП (значения идентификатора указаны в табл. 8.9);
SN – количество данных, после которого необходимо проводить синхронизацию (определяется в ходе установления соединения);
data – данные БДП;
senddata – очередь с посланными, но не подтвержденными БДП;
outgoingq – очередь с посылаемыми БДП;
ingoingq – очередь с принятыми, но не подтвержденными БДП.

В случае введения защиты в условиях лабораторного программного комплекса можно ограничится средствами защиты, аналогичными средствам защиты транспортного уровня, т.е. необходимы введение дополнительного поля, содержащего контрольную сумму данных, в БДП данных, и его соответствующая обработка. Это поле и его обработка аналогичны средствам защиты транспортного уровня, описанным в материале, посвященном транспортному уровню. При этом стоит заметить, что если протокол поддерживает защиту, но при установлении соединения защита была отключена, то поле защиты исключается и его обработка, разумеется, не производится.

Таблица 8. 9

Значения идентификатора

Идентификатор	Значение
1	Индикация главной синхронизации
2	Индикация вспомогательной синхронизации
3	Индикация передачи данных
4	Подтверждение главной синхронизации
5	Индикация ресинхронизации
6	Подтверждение ресинхронизации
7	Индикация передачи маркера
8	Индикация запроса маркера
9	Индикация упорядоченного завершения
10	Индикация безусловного завершения пользователя

Для управления расположением маркеров достаточно передать соответствующий БДП, как это указано выше, переустановить значения переменных, отвечающих за расположение маркеров, и убедиться, что не произошло потери или дублирования маркера.

Синхронизация необходима для проверки правильности передачи данных в нужном порядке, без потери и дублирования БДП. Для

этого БДП нумеруются, и во время вызова услуг малой и основной синхронизации проверяется правильность передачи.

Услуга ресинхронизации вызывается в случае нарушения синхронизации или проблем с маркерами (потерей или дублированием). Во время выполнения этой услуги происходит согласование маркеров, а также повторная передача всех данных, не подтвержденных услугой основной синхронизации.

Диаграммы состояний конечного автомата фазы передачи данных представлены на рис. 8.4–8.5.

Фаза завершения сеансового соединения характеризуется тремя услугами:

– S-RELEASE – упорядоченное завершение;
– S-P-ABORT и S-U-ABORT – безусловное завершение.

Основным отличием этих услуг является то, что при упорядоченном завершении не происходит потери данных, а при безусловном эта потеря может произойти. Фактически услуги S-P-ABORT и S-U-ABORT являются аналогом услуги транспортного уровня T_DISCONNECT. Услуга S_RELEASE отличается от них тем, что после прихода ее запроса участники сеансового соединения перед разрывом соединения пытаются переслать все не посланные данные. Пример простейшей организации услуги:

Обработка события **S_P_ABORT.REQ**:

```
buffer buf 8 10 4 $SP 4              создать БДП данных
down T_DATA.REQ userdata $buf        отослать их другой системе
down T_DISCONNECT.REQ                разорвать соединение
```

Контрольные вопросы

1. Определите понятия «маркер», «точка синхронизации», «диалоговый элемент», «активность», «рецензия», «переговорный», «функциональная группа».
2. Приведите последовательность действий протокола во время выполнения следующих услуг:
   а) ресинхронизации сеансового уровня;
   б) установления соединения сеансового уровня;
   в) разрыва соединения сеансового уровня;
   г) передачи данных сеансового уровня.
2. Опишите услуги, предоставляемые сеансовым уровнем.
3. Приведите в виде диаграммы состояний-переходов конечный автомат, моделирующий работу системы

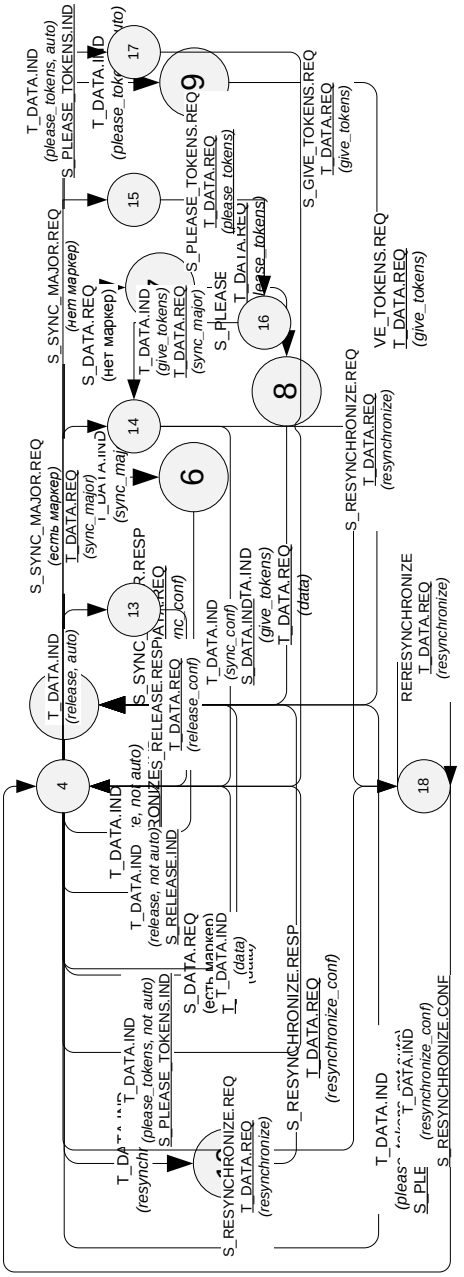


Рис. 8

дующих фаз:

а) установления соединения сеансового уровня;

б) передачи данных сеансового уровня;

в) разрыва соединения сеансового уровня.

4. Охарактеризуйте параметры качества сервиса сеансового уровня.

9. УРОВЕНЬ ПРЕДСТАВЛЕНИЯ

9.1. Общие положения

Оконечные системы (абоненты) вычислительных сетей весьма разнообразны и представлены устройствами различных типов – от простых символьно-ориентированных дисплеев до универсальных ЭВМ и систем, ориентированных на базы данных. В разных устройствах используется различное внутренне представление хранимой информации. Для обеспечения их взаимодействия модель ВОС содержит шестой уровень, называемый уровнем представления (представительным уровнем).

В процессе своего функционирования объекты уровня представления используют протокол, который позволяет согласовать различия в синтаксисе данных с помощью преобразования местного представления (локальный синтаксис) в синтаксис передачи. Синтаксис передачи вместе с правилами преобразования составляет образ конкретного соединения уровня представления. Согласование синтаксических различий должно выполняться таким образом, чтобы сохранить семантику передаваемой информации.

Представительный уровень в ходе согласования синтаксических различий имеет дело с двумя аспектами. Первый относится к описанию (заданию) синтаксиса со стороны прикладных объектов, второй – к тому, как описанные заданным синтаксисом данные выражаются в терминах представления их значений в среде ВОС. Первый аспект обозначается термином «абстрактный синтаксис» второй – «синтаксис передачи».

Конкретная семантика прикладных систем может быть выражена с помощью различных абстрактных синтаксисов $АС_1$, …, $АС_n$ (рис. 9.1), каждый из которых использует одинаковые или различные синтаксисы передачи $СП_1$, …, $СП_m$.

В общем случае не предполагается однозначного соответствия между конкретными абстрактными синтаксисами и синтаксисами

передачи. Сервис представительного уровня позволяет устанавливать такие соотношения динамически. Например, при использовании одного и того же абстрактного синтаксиса передача может происходить с шифрацией или без шифрации, со сжатием или без сжатия передаваемой информации.

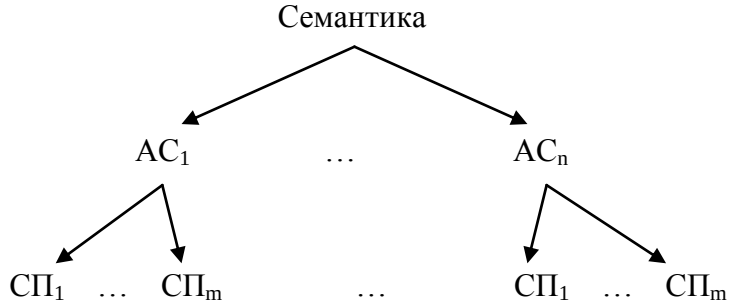


Рис. 9.1. Связь семантики, абстрактных синтаксисов и синтаксисов передачи

Функция согласования синтаксисов передачи поддерживается механизмом переговоров, выполняемым в соответствии с представительным протоколом.

Таким образом, двумя основными функциями уровня представления являются:

- согласование синтаксиса передачи;
- преобразование (в обе стороны) между абстрактным синтаксисом и синтаксисом передачи.

Это преобразование выполняется в рамках представительного объекта невидимым со стороны представительного протокола способом, чем обеспечивается отсутствие влияния таких преобразований на протокол – он имеет дело лишь с использованием системы идентификации синтаксисов передачи.

К другим функциям уровня представления относятся:

- запрос на установление сеанса;
- запрос на прекращение сеанса;
- передача данных.

При каждом случае передачи без установления соединения уровень представления добавляет к сеансовой службе вызываемые услуги преобразования синтаксиса и выбора синтаксиса. Для их обеспечения уровень представления выполняет функции передачи данных, идентификации синтаксиса и его преобразования.

Возможности, предоставляемые уровнем представления своим пользователям, отнесены к следующим категориям, каждая из которых объединяет ряд функционально схожих услуг:

– установление и завершение соединения;
– управление контекстами;
– передача информации;
– управление диалогом.

Работа с заданным множеством контекстов зависит от выбираемых на этапе установления соединения функциональных групп.

Функциональные группы (блоки) – логические объединения связанных между собой услуг. Такие объединения используются для указания требований пользователя представительной службы во время установления соединения.

Выделяются три функциональные группы:

– ядра (базовая);
– управления контекстами;
– восстановления контекстов.

Функциональная группа ядра доступна всегда, она обеспечивает услуги установления соединения, передачи информации и завершения соединения. Передача информации поддерживается при использовании выбранных сеансовых функциональных групп. Если выбрана только эта группа, то множество заданных на этапе установления представительного соединения контекстов в дальнейшем во время существования этого соединения не может быть изменено. Эта функциональная группа устанавливается по умолчанию.

Функциональная группа управления контекстами должна явно заказываться и согласовываться при установлении представительного соединения. Если она согласована, то множество заданных контекстов может быть изменено во время существования соединения. Услуга, с использованием которой это достигается, может конфликтовать с другими услугами уровня представления, обладающими так называемыми разрушающими воздействиями. Такие конфликты могут привести к рассогласованию множества заданных контекстов на разных концах соединения. В этом случае возможно получение данных в контексте, неизвестном представительному объекту, о чем он будет сообщать пользователю: “нечи-

таемые данные". В то же время пользователи представительного сервиса могут избежать таких конфликтов корректным использованием маркеров.

Функциональная группа восстановления контекстов явно заказывается и согласовывается при установлении представительного соединения, причем в этом случае также должна быть заказана функциональная группа управления контекстами. Выбор функциональной группы восстановления дает возможность запоминать множества заданных контекстов в специфицированных точках во время существования представительного соединения. В дальнейшем пользователь представительного сервиса может затребовать возврат к одной из запомненных точек, а сервис представительного уровня обеспечивает корректное восстановление контекстов, существовавших в этой точке. Точками запоминания могут служить точки главной или вспомогательной синхронизации, прерывания активности, установления соединения.

Ход процессов восстановления и управления контекстами во избежание рассогласования контекстных множеств на концах соединения должен подчиняться ряду правил, опирающихся, в частности, на использование маркеров и номеров точек запоминания.

9.2. Службы уровня представления

На рис. 9 й последовательности взаимодействия основных служб обеспечиваемых уровнем представления. Как можно увидеть, с каждым примитивом связаны некоторые параметры. Так как уровень представления и сеансовый уровень совместно образуют функцию, осуществляемую в интересах прикладного уровня, то результатом действия большинства из изображенных примитивов служб уровня представления оказываются соответствующие блоки данных протокола представления (ПдБДП), формируемых на основании параметров, связанных с примитивами (рис. 9.3).

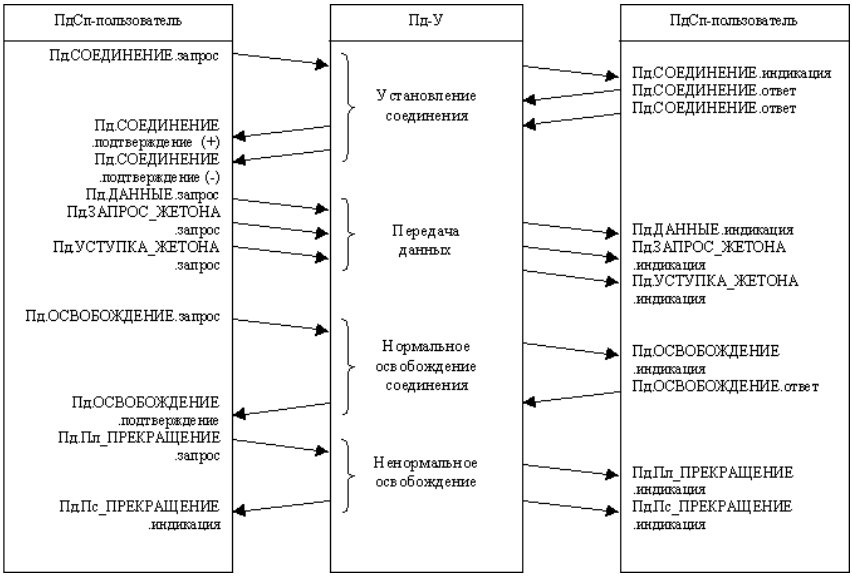


Рис. 9.2. Основные службы уровня представления

Наконец, имеются несколько примитивов службы представления, которые без какой-либо модификации отображаются на соответствующие примитивы СнСл без формирования ПдБДП. Обычно параметры, связанные такими примитивами, передаются в поле данных пользователя соответствующего примитива СнСл. В число таких примитивов входят примитивы службы ПдОСВОБОЖДЕНИЕ, а также примитивы, обеспечивающие функции управления синхронизацией, маркерами и др.

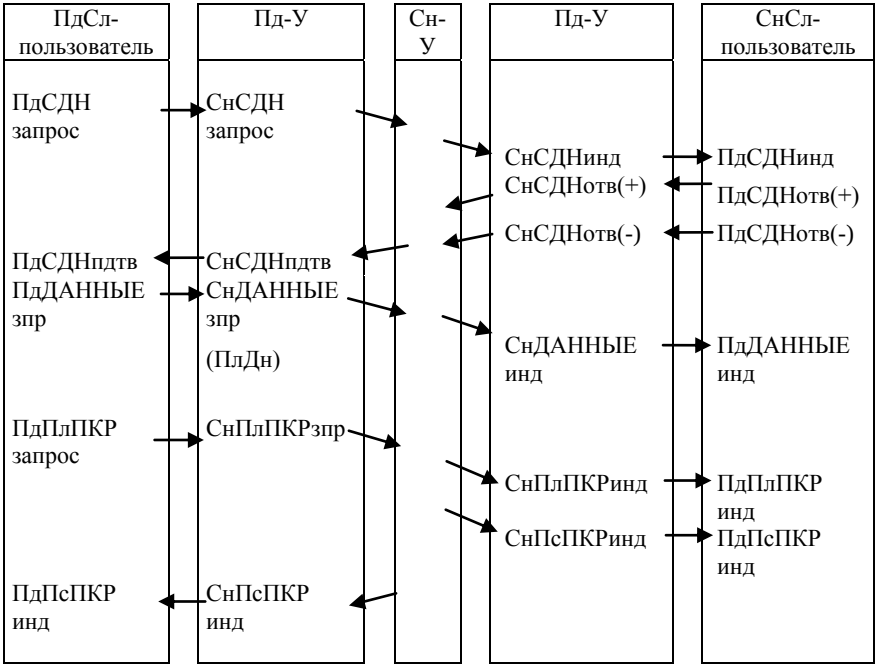


Рис. 9.3. Связанные с уровнем представления сеансовые службы

Таким образом,

Примитив службы	формирует ПдБДП
ПдСДНзапрос	Соединение уровня представления ПдСд
ПдСДНответ(+)	Соединение уровня представления принято (ПдСдПН)
ПдСДНответ(-)	Соединение уровня представления отказано (ПдСдОТ)
ПдДАННЫЕзапрос	Передача данных уровня представления ПдДн
ПдПлПКРзапрос	Ненормальное освобождение(пользователь) ПлПКР
Пд.Пс-ПКРиндикация	Ненормальное освобождение(поставщик) ПсПКР

9.3. Автоматная модель протокола уровня представления

На рис. 9.4 и 9.5 изображены диаграммы состояний конечного автомата взаимодействующих систем, в табл. 9.1–9.3 выписаны списки имен элементов множеств входных и выходных событий и состояний автомата.

Таблица 9.1

Список входных событий

Событие	Смысл/значение
P_CONNECT.REQ	Запрос установления соединения уровня представления
S_CONNECT.CONF	Подтверждение установления соединения сеансового уровня
S_DATA.IND (param_2)	Индикация получения данных, содержащих выбранные параметры соединения
P_U_ABORT.IND	Индикация разрыва соединения уровня представления пользователем
P_U_ABORT.REQ	Запрос разрыва соединения уровня представления пользователем
P_RELEASE.REQ	Запрос релиза соединения уровня представления
S_RELEASE.IND	Индикация релиза сеансового соединения
P_RELEASE.RESP	Ответ релиза соединения уровня представления
S_RELEASE.CONF	Подтверждение релиза сеансового соединения
S_GIVE_TOKENS.IND	Индикация получения токена сеансового уровня
P_SYNC_MAJOR.REQ	Запрос на синхронизацию уровня представления
P_SYNC_MAJOR.RESP	Ответ на синхронизацию уровня представления
S_SYNC_MAJOR.IND	Индикация синхронизации сеансового уровня
S_SYNC_MAJOR.CONF	Подтверждение синхронизации сеансового уровня
S_RESYNCHRONIZE.CONF	Подтверждение синхронизации сеансового уровня
P_DATA.REQ	Запрос на передачу данных уровня представления
P_GIVE_TOKENS.REQ	Запрос получения токена уровня представления
P_PLEASE_TOKENS.REQ	Запрос на запрос получения токена уровня представления
P_RESYNCHRONIZE.REQ	Запрос ресинхронизации уровня представления
P_RESYNCHRONIZE.RESP	Ответ ресинхронизации уровня представления
S_RESYNCHRONIZE.IND	Индикация ресинхронизации сеансового уровня
S_PLEASE_TOKENS.IND	Индикация запроса получения токена сеансового уровня
S_CONNECT.IND	Индикация установления сеансового соединения
P_CONNECT.RESP	Ответ на установление соединения уровня представления
S_DATA.IND (param)	Индикация получения данных с возможными параметрами

S_DATA.IND (data)	Индикация получения сеансовых данных

Таблица 9.2

Список выходных событий

Событие	Смысл/значение
S_CONNECT.REQ	Запрос установления сеансового соединения
S_DATA.REQ (param/2)	Запрос передачи параметров соединения уровня представления
P_CONNECT.CONF	Подтверждение установления представительного соединения
S_U_ABORT.IND	Индикация разрыва сеансового соединения пользователем
S_U_ABORT.REQ	Запрос разрыва сеансового соединения пользователем
S_RELEASE.REQ	Запрос релиза сеансового соединения
P_RELEASE.IND	Индикация релиза соединения уровня представления
S_RELEASE.RESP	Ответ на релиз сеансового соединения
P_RELEASE.CONF	Подтверждение релиза соединения уровня представления
P_GIVE_TOKENS.IND	Индикация получения токена уровня представления
S_SYNC_MAJOR.REQ	Запрос синхронизации сеансового уровня
S_SYNC_MAJOR.RESP	Ответ на синхронизацию сеансового уровня
P_SYNC_MAJOR.IND	Индикация синхронизации уровня представления
P_SYNC_MAJOR.CONF	Подтверждение синхронизации уровня представления
P_RESYNCHRONIZE.CONF	Подтверждение ресинхронизации уровня представления
S_DATA.REQ (data)	Запрос передачи сеансовых данных
S_GIVE_TOKENS.REQ	Запрос получения токена сеансового уровня
S_PLEASE_TOKENS.REQ	Запрос запроса токена сеансового уровня
S_RESYNCHRONIZE.REQ	Запрос ресинхронизации сеансового уровня
S_RESYNCHRONIZE.RESP	Подтверждение ресинхронизации сеансового уровня
P_RESYNCHRONIZE.IND	Индикация ресинхронизации уровня представления
P_PLEASE_TOKENS.IND	Индикация запроса токена уровня представления
P_CONNECT.IND	Индикация установления соединения уровня представления
S_CONNECT.RESP	Ответ на установление сеансового соединения
P_DATA.IND	Индикация получения данных

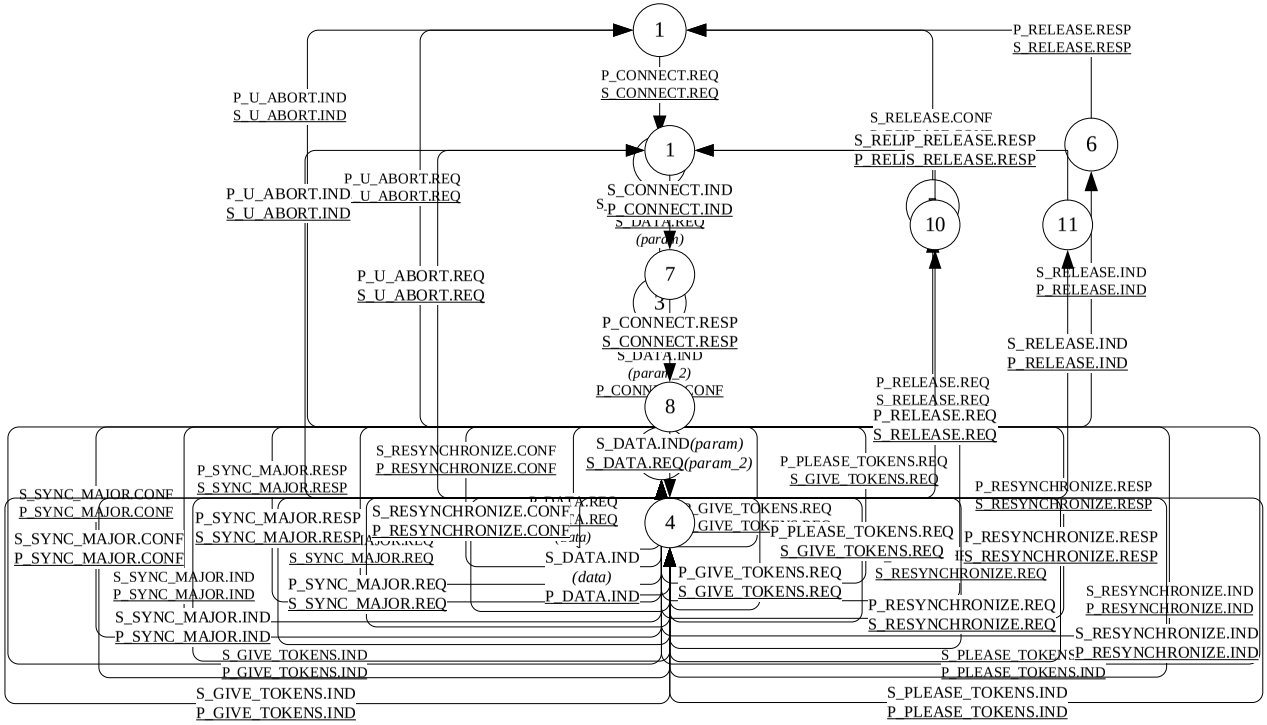

1
P_CONNECT.REQ
S_CONNECT.REQ
P_U_ABORT.IND
S_U_ABORT.IND
P_RELEASE.RESP
S_RELEASE.RESP
1
S_RELEASE.CONF
S_RELIP_RELEASE.RESP
P_RELIS_RELEASE.RESP
6
P_U_ABORT.REQ
S_U_ABORT.REQ
P_U_ABORT.IND
S_U_ABORT.IND
S_CONNECT.IND
P_CONNECT.IND
S_DATA.REQ
(param)
10
11
7
P_U_ABORT.REQ
S_U_ABORT.REQ
S_RELEASE.IND
P_RELEASE.IND
3
P_CONNECT.RESP
S_CONNECT.RESP
S_DATA.IND
(param_2)
P_CONNECT.CONF
S_RELEASE.IND
P_RELEASE.IND
8
P_RELEASE.REQ
S_RELEASE.REQ
P_RELEASE.REQ
S_RELEASE.REQ
S_DATA.IND(param)
S_DATA.REQ(param_2)
S_RESYNCHRONIZE.CONF
P_RESYNCHRONIZE.CONF
P_PLEASE_TOKENS.REQ
S_GIVE_TOKENS.REQ
P_RESYNCHRONIZE.RESP
S_RESYNCHRONIZE.RESP
P_SYNC_MAJOR.RESP
S_SYNC_MAJOR.RESP
S_SYNC_MAJOR.CONF
P_SYNC_MAJOR.CONF
4
P_PLEASE_TOKENS.REQ
S_GIVE_TOKENS.REQ
P_RESYNCHRONIZE.RESP
S_RESYNCHRONIZE.RESP
S_SYNC_MAJOR.REQ
S_DATA.IND
(data)
P_DATA.IND
P_GIVE_TOKENS.REQ
S_GIVE_TOKENS.REQ
S_RESYNCHRONIZE.REQ
S_SYNC_MAJOR.IND
P_SYNC_MAJOR.IND
P_SYNC_MAJOR.REQ
S_SYNC_MAJOR.REQ
P_RESYNCHRONIZE.REQ
S_RESYNCHRONIZE.REQ
S_RESYNCHRONIZE.IND
P_RESYNCHRONIZE.IND
S_GIVE_TOKENS.IND
P_GIVE_TOKENS.IND
S_PLEASE_TOKENS.IND
P_PLEASE_TOKENS.IND
S_GIVE_TOKENS.IND
P_GIVE_TOKENS.IND
S_PLEASE_TOKENS.IND
P_PLEASE_TOKENS.IND

Таблица 9.3

Список состояний автомата

Событие	Смысл/значение
1	Начальное состояние
2	Получен запрос на установление соединения уровня представления
3	Получено подтверждение на установление соединения уровня представления
4	Представительное соединение установлено
5	Получен запрос на релиз соединения в инициирующей системе
6	Получена индикация релиза соединения в инициирующей системе
7	Получена индикация установления соединения уровня представления
8	Получен ответ установления соединения уровня представления
10	Получен запрос на релиз соединения в отвечающей системе
11	Получена индикация релиза соединения в отвечающей системе

9.4. Абстрактные синтаксисы и синтаксисы передачи

Нотация абстрактного синтаксиса должна обеспечивать определение сложных типов данных и задание значений этих типов без указания способа представления (в виде последовательности октетов) отдельных экземпляров этих типов во время передачи.

В текущей версии лабораторного комплекса потребности приложений разработчики ограничили стремлением представить взаимосвязь прикладных процессов в виде элементарных взаимодействий простых моделей двух животных. В итоге в настоящее время, помимо таких типов, как число, строка и буфер, – они стандартны для встроенного языка комплекса: необходимыми представляются следующие основные типы – звук, движение, цвет, запах и структура.

Из рис. 9.1 видно, что возможны различные способы выражения семантики абстрактным синтаксисом и абстрактных синтаксисов – синтаксисами передачи. Так, различная семантика может использовать различный (см. рис. 9.1а) или одинаковый (см. рис. 9.1б) абстрактный синтаксис с одинаковым синтаксисом передачи. Для указания приемлемой комбинации абстрактного синтаксиса и синтаксиса передачи используется термин “**контекст представления**”.

Как и ASN. 1, абстрактные синтаксисы лабораторного комплекса – это средства со строгой типизацией, с явным описанием используемых типов. В текущей версии комплекса в целях упрощения во всех версиях применяются структуры фиксированного вида

– звук, движение, цвет, запах – различаться будут лишь разделители. Пример такой структуры: {{мяу}{прыгнул}{серый}{}} в одной системе и [[мяу][прыгнул][серый][]] в другой (в приведенном примере разделители – это скобки {} и []).

Уже шла речь о том, что каждому абстрактному синтаксису может соответствовать множество синтаксисов передачи, см. рис. 9.1. Синтаксисы передачи задают правила кодирования, определяющие значения потока октетов между объектами уровня представления. В текущей версии комплекса из возможного множества наборов таких правил используется так называемый базовый набор правил кодирования ASN. 1, принятый в настоящее время в качестве стандарта МОС.

Таким образом, в лабораторном практикуме используются два способа указания длины содержимого, которые рекомендованы МОС. Это кодирование с явным указанием длины и кодирование по признаку конца содержимого.

Кодирование с явным указанием длины использует следующее преобразование некоторого типа данных:

Идентификатор	Длина	Содержимое

Идентификатор представляет собой число, которое задает тип содержимого, а длина содержит в себе длину содержимого. В табл. 9.4 представлено соответствие типов данных и идентификаторов.

Таблица 9.4

Соответствие типов данных и идентификаторов

Идентификатор	Тип
1	Структура
2	Число
3	Строка
4	Буфер
10	Звук
20	Движение
30	Цвет
40	Запах

Так звук “мяу” будет представлен следующим образом:

Идентификатор	Длина	Содержимое
10	3	Мяу

А структура вида {{мяу}{прыгнул}{серый}{}} будет выглядеть следующим образом:

Общая структура													
		Звук “мяу”			Движение “прыгнул”			Цвет “серый”			Запах “”		
1	47	10	3	Мяу	20	7	Прыгнул	30	5	Серый	40	0	

Декодирование:

1. Считываются два первых числа. Первое обозначает тип следующего содержимого, второе – его длину.
2. Если тип простой (не структура), переходим к следующему пункту, иначе, п. 1.
3. Считываем в переменную соответствующего типа буфер указанного размера.
4. Если исходный буфер не пуст, переходим к п. 1.
5. Собираем целевую структуру с расстановкой необходимых разделителей.

В данном примере:

1. Считываются два первых числа. Получаем тип – структура, длина – 47.
2. Считываются два первых числа. Получаем тип – звук, длина – 3.
3. Считываем в переменную sound 3 байта. В результате переменная sound = “мяу”.
4. Считываются два первых числа. Получаем тип – движение, длина – 7.
5. Считываем в переменную move 7 байт. В результате переменная move = “прыгнул”.
6. Считываются два первых числа. Получаем тип – цвет, длина – 5.
7. Считываем в переменную color 5 байт. В результате переменная color = “серый”.
8. Считываются два первых числа. Получаем тип – запах, длина – 0.
9. Считываем в переменную smell 0 байт. В результате переменная smell = “”.
10. Исходный буфер пуст, переходим к составлению целевой структуры.
11. Пусть в целевой структуре разделитель квадратная скобка ([]), тогда она равна [[+$sound+][+$move][+$color][+$smell]] или [[мяу][прыгнул][серый][]].

12. Таким образом структура {{мяу}{прыгнул}{серый}{}} преобразована в [[мяу][прыгнул][серый][]].

Кодирование по признаку конца содержимого имеет следующий вид:

Идентификатор	Содержимое	Признак конца содержимого

В силу ограничений интерпретатора в качестве признака конца содержимого использовать два нулевых байта, как это рекомендуется МОС, нельзя, поэтому используйте любые другие комбинации, например “255”.

Таким образом звук ‘мяу’ будет представлен в виде:

Идентификатор	Содержимое	Признак конца содержимого
10	Мяу	255

А структура вида {{мяу}{прыгнул}{серый}{}} будет выглядеть так:

Общая структура													
	Звук “мяу”			Движение “прыгнул”			Цвет “серый”			Запах “”		
1	10	Мяу	255	20	Прыгнул	255	30	Серый	255	40	255	255

Таким образом, благодаря механизмам уровня представления (т.е. согласованию синтаксисов, кодированию и т.д.) на приемном конце принятые данные всегда будут трактоваться однозначно и верно, несмотря на различие абстрактных синтаксисов систем. Это происходит из-за перекодировки данных в синтаксис передачи, который одинаков (в течение одного сеанса связи) для обоих систем.

9.5. Фрагменты реализации протокола уровня представления на лабораторном комплексе

Основная задача уровня представления в рамках данной части ДЗ – согласование контекста представления и преобразование (в обе стороны) между абстрактным синтаксисом и синтаксисом передачи. Служба представления содержит три фазы: установления соединения, передачи данных, освобождения соединения. В целях упрощения задачи из службы представления изъяты услуги управления контекстами.

В фазе установления соединения предусмотрена одна услуга, P-CONNECT.

Пример простейшей организации услуги P-CONNECT без каких-либо параметров:

Обработка события P_CONNECT.REQ:
down S_CONNECT.REQ address $sysB
Обработка события S_CONNECT.IND:
up P_CONNECT.IND
Обработка события P_CONNECT.RESP:
down S_CONNECT.RESP address $sysA
Обработка события S_CONNECT.CONF:
up P_CONNECT.CONF

где

sysB – адрес вызываемой системы;
sysA – адрес вызывающей системы.

При установлении соединения системы должны договориться об используемом контексте. Это необходимо для того, чтобы системы "говорили" на одном языке. В случае отсутствия такой договоренности на приемном конце вместо полезной информации получится информационный "мусор". Как, например, при открытии текстового файла в 866-й кодировке с помощью 1251-й. Механизм, используемый при этом, может быть похож на используемый сеансовым уровнем при установлении защиты с небольшими изменениями. Такой механизм описан в предыдущей части. Он обеспечивает передачу служебных данных после установления сеансового соединения перед индикацией соединения уровня представления. Изменения заключаются в том, что инициирующая система посылает ведомой все возможные варианты синтаксиса передачи, а выбирает среди них тот, который возможен также с её стороны и отсылает его обратно.

Фаза передачи данных включает в себя средства передачи информации и управления диалогом.

Средства передачи информации позволяют пользователям обмениваться информацией и включают в себя услугу P-DATA, которая и производит преобразование между абстрактным синтаксисом и синтаксисом передачи. Правила кодирования описаны в предыдущем разделе. В примере ниже используется кодирование с явным указанием длины, при этом структура имеет укороченный вид (только два поля), на передающем конце использует в качестве разделителя знак фигурные скобки ({}), а на принимающем – знак квадратные скобки ([]) и имеет вид {{звук}{движение}}. Все различие между абстрактными синтаксисами на обоих концах сводит-

ся к различиям в разделителях при описании структур. Используемый способ кодировки и разделители согласовываются ранее в процессе установления соединения при согласовании контекстов.

Обработка события P_DATA.REQ:

```
;datatype – тип данных (число)
; clbrack – закрывающий символ-разделитель (строка длиной 1) (в примере “}”)
; определить тип данных
unbuffer userdata datatype 4 s sizeof(userdata)-4
;структура
if $datatype == 1 struct
;число
if $datatype == 2 number
;строка
if $datatype == 3 string
;буфер
if $datatype == 4 buff
goto send1

number:
buffer buf 12 2 4 4 4 $s 4
goto send1

string:
buffer buf sizeof(s)+8 3 4 sizeof(s) 4 $s 4
goto send1

buff:
buffer buf sizeof(s)+8 4 4 sizeof(s) 4 $s 4
goto send1

struct:
;пусть userdata содержит структуру {{мяу}{прыгнул}{серый}{}}
;подготовка буфера (перевод строки в буфер и удаление двух первых символов, т.е. символов “{{”)
;s содержит “{{мяу}{прыгнул}}”; t содержит “мяу}{прыгнул}}”
unbuffer s c 2 t sizeof(s)-2
;извлечение переменных соответствующих  типов
```

```
; s содержит “прыгнул}}” ; sound содержит “мяу”
unbuffer t sound pos($clbrack,t)-1 c 2 s sizeof(t)-2-$q
; t содержит “” move содержит “прыгнул”
unbuffer s move pos($clbrack,s)-1 c 2 t sizeof(s)-2-$q
;кодирование переменных соответствующих типов
; bufsound содержит “10 3 мяу”
buffer bufsound sizeof(sound)+8 10 4 sizeof(sound) 4 $sound sizeof(sound)
; bufmove содержит “20 7 прыгнул”
buffer bufmove sizeof(move)+8 20 4 sizeof(move) 4 $move sizeof(move)
;расчет размера окончательного буфера
set q sizeof(bufsound)+sizeof(bufmove)
;приготовление окончательного буфера
;buf содержит ”1 26 10 3 мяу 20 7 прыгнул”
buffer buf $q+8 1 4 $q 4 $bufsound sizeof(bufsound) $bufmove sizeof(bufmove)
goto send1

send1:
down S_DATA.REQ userdata $buf
goto exit
```

Событие T_DATA.IND:

```
; clbrack – закрывающий символ-разделитель (строка длиной 1) (в примере “]”)
; opbrack – открывающий символ-разделитель (строка длиной 1) (в примере “[”)
;определение типа и длины буфера (type – тип; len – длина)
unbuffer userdata type 4 len 4 buf sizeof(userdata)-8
if $type == 1 struct1
if $type == 2 number1
if $type == 3 string1
if $type == 4 buff1
goto exit

number1:
unbuffer buf num $len
```

```
buffer data 8 2 4 $num 4
out $num
goto send

string1:
unbuffer buf str $len
buffer data sizeof(str)+4 3 4 $str sizeof(str)
out $str
goto send

buff1:
unbuffer buf buff $len
buffer data sizeof(buff)+4 4 4 $buff sizeof(buff)
goto send

struct1:
;userdata -”1 26 10 3 мяу 20 7 прыгнул”
; type – 1    len – 26 buf -”10 3 мяу 20 7 прыгнул”

next:
if sizeof(buf)==0 done
;buf содержит ”1 26 10 3 мяу 20 7 прыгнул”
unbuffer buf type 4 len 4 buf1 sizeof(buf)-8
if $type==10 sound
if $type==20 move
goto error

sound:
; ssound - ”мяу”  ;  buf - ”20 7 прыгнул”
unbuffer buf1 ssound $len buf sizeof(buf1)-$len
goto next
move:
; smove - ”прыгнул”    buf - ”  ”
unbuffer buf1 smove $len buf sizeof(buf1)-$len
goto next

error:
out ”неизвестный тип”
```

```
goto exit

;приготовление окончательного буфера
done:
set str $opscob+$opscob+$ssound+$clscob+$opscob+$smove+
$clscob+$clscob ;str – “[[мяу][прыгнул]]”
buffer data sizeof(str)+4 1 4 $str sizeof(str)
goto send

send:
up P_DATA.IND userdata $data

exit:
```

Средство управления диалогом дает возможность распределять маркеры, управлять синхронизацией и ресинхронизацией. Используются следующие услуги:

- для управления расположением маркеров – P-GIVE-TOKENS, P-PLEASE-TOKENS, P-GIVE-CONTROL;
- для фиксации точек синхронизации и ресинхронизации – P-SYNC-MINOR, P-SYNC-MAJOR, P-RESYNCHRONIZE.

Эти услуги отображаются на соответствующие сервисы сеансового уровня, и поэтому их реализация проста. Например:

Обработка события P_SYNC_MAJOR.REQ:

```
down S_SYNC_MAJOR.REQ
```

Фаза завершения сеансового соединения характеризуется тремя услугами:

S-RELEASE – упорядоченное завершение;

S-P-ABORT и S-U-ABORT – безусловное завершение.

Эти услуги отображаются на соответствующие сервисы сеансового уровня так же, как и услуги средств управления диалогом.

Контрольные вопросы

1. В контексте эталонной модели ВОС кратко охарактеризуйте значения и взаимоотношения терминов «абстрактный синтаксис» и «синтаксис передачи».
2. Приведите последовательность действий протокола во время выполнения следующих услуг:

а) установления соединения уровня представления;
б) разрыва соединения уровня представления;
в) передачи данных уровня представления.

3. Опишите услуги, предоставляемые уровнем представления.
4. Приведите в виде диаграммы состояний-переходов автоматную модель следующих фаз:
   а) установления соединения уровня представления;
   б) передачи данных уровня представления;
   в) разрыва соединения уровня представления.
5. Охарактеризуйте параметры качества сервиса уровня представления.

10. ПРИКЛАДНОЙ УРОВЕНЬ

10.1. Общие положения

Прикладной уровень является последним (седьмым) уровнем ЭМВОС. С этим связан ряд особенностей, отличающих его от других уровней эталонной модели. На прикладном уровне завершается наращивание сервисной мощности модели ВОС по архитектурной вертикали и начинается своего рода горизонтальное наращивание. Сервис, поставляемый различными протокольными прикладными объектами с учетом определенных ограничений, можно объединять, формируя различные профили сервиса прикладного уровня в интересах конкретных специальных прикладных систем.

На прикладном уровне осуществляется окончательное и естественное погружение механизмов взаимосвязи, объявленных в модели, в вычислительную среду с ее понятийным построением. Поэтому очевидна потребность в уточненном описании понятия прикладного процесса (application process, AP).

Прикладной процесс – это идентифицируемый объект в рамках реальной открытой системы, ведущий обработку информации и ответственный за согласование правил среды своего существования с законами модели ВОС. Прикладной процесс представляется в рамках модели ВОС одним или более прикладными объектами (Application Entity, AE).

Прикладной объект, далее, представляется совокупностью элементов прикладных служб (Application Service Elements, ASE). Такая совокупность, в свою очередь, может быть разделена на набор

общих элементов прикладных служб (ОЭПС, ACSE), некоторый набор **специальных элементов прикладных служб** (СЭПС, SASE), а также элемент, создаваемый разработчиком прикладной системы - **элемент пользователя** (ЭП, UE)[1], рис. 10.1.

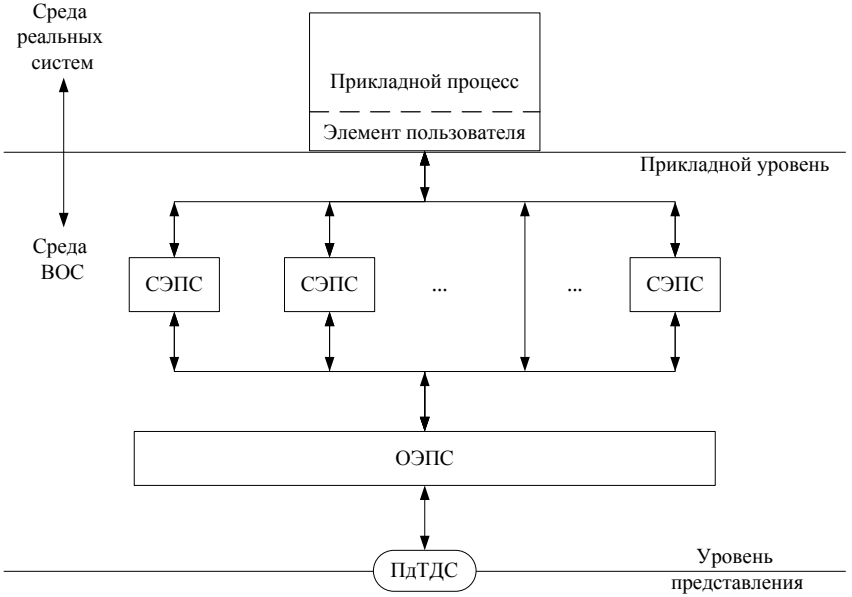


Рис. 10.1. Элементы прикладных служб

Целью выделения общего прикладного сервиса в отдельный стандарт является обеспечение разработчиков прикладных распределенных систем обработки информации наиболее общими унифицированными услугами, которые в совокупности с услугами представительного уровня составляют понятийную основу построения и работы приложений.

В наборе общих элементов прикладных служб (ОЭПС) имеется элемент службы управления ассоциацией (ЭСУА) – множество функций, обеспечивающих минимальные требуемые услуги служ-

[1]В ряде источников используется аббревиатура CASE – common application service elements, термину ASE может ставиться в соответствие термин множество прикладных услуг, термину UE – термин услуги пользователя, интерфейс пользователя, агент пользователя. Элемент пользователя также может не определяться, что подразумевает выполнение его функций прикладным процессом.

бы прикладного уровня[1]. Прикладная ассоциация устанавливается между двумя вызовами[2] прикладных объектов при необходимости обмена информацией между прикладными объектами с целью выполнения задания по распределенной обработке информации.

Свойства и цель информационного обмена по прикладной ассоциации определяются в зависимости от устанавливаемого для нее контекста обработки информации и прикладного контекста. Первый определяется конкретным заданием по распределенной обработке информации, для которого установлена ассоциация. Прикладной контекст формируется явно определенным набором элементов прикладной службы, которые могут быть вызваны пользователем ассоциации.

Каждый элемент прикладной службы требует описания предоставляемых им услуг и поддерживающего их выполнение протокола.

Спецификация протокола задает правила информационного обмена между равноправными элементами прикладной службы. Она также может содержать описание используемой службы представительного уровня или услуг, предоставляемых другими элементами прикладной службы.

Определение услуг аналогично определению служб уровней ЭМВОС, оно описывает функции, выполняемые данным элементом.

Прикладное взаимодействие базируется на представительном соединении, которое создается и завершается поставщиком общего прикладного сервиса. Поэтому у пользователя сервиса нет необходимости устанавливать представительное соединение прямым выходом на представительный уровень с помощью примитива P-CONNECT. Доступ к ЭСУА, обеспечивающему услугу образования ассоциации, может быть осуществлен опосредованно – вызовом специального элемента прикладной службы (СЭПС) или элемента пользователя, что иллюстрируется структурой связей элементов на рис. 10.1.

[1] Используется также термин «базовое ядро ACSE».

[2] Вызов прикладного объекта выполняет функции прикладного объекта для конкретного случая обмена информацией.

В параметрическом отношении общая функциональная ориентация всех трех верхних уровней ЭМВОС (на приложения) проявляется в том, что многие параметры, связанные с примитивами, отображаются непосредственно с одного уровня на другой. Параметрическая комплектация сеансовой услуги S-CONNECT и представительной услуги P-CONNECT осуществляется на основе прикладной услуги A-ASSOCIATE.

Для того чтобы некоторый вызов прикладного объекта установил ассоциацию с другим вызовом прикладного объекта, ему должно быть известно наименование этого прикладного объекта. Используя такие наименования, пользователь прикладной службы сначала с помощью СЭПС справочной службы (СС) должен получить адреса прикладных объектов. Все последующие его обращения к другим СЭПС уже, как правило, содержат в качестве параметров как имена, так и соответствующие адреса обоих прикладных объектов.

10.2. Услуги, предоставляемые ЭСУА

Для управления одной ассоциацией используются следующие услуги: A-ASSOCIATE (Пк-АССОЦИИРОВАНИЕ), A-RELEASE (Пк-ОСВОБОЖДЕНИЕ), A-U-ABORT (Пк-Пл-ПРЕКРАЩЕНИЕ (Пк-РАЗРЫВ)), A-P-ABORT (Пк-Пс-ПРЕКРАЩЕНИЕ (Пк-Пс-РАЗРЫВ)).

Подтверждаемая услуга A-ASSOCIATE позволяет прикладному объекту установить прикладное соединение (ассоциацию) с другим прикладным объектом. В ходе установления ассоциации прикладные объекты обмениваются параметрами прикладного соединения и согласовывают их. Подтверждаемая услуга A-RELEASE позволяет прикладному объекту произвести упорядоченное завершение существующего прикладного соединения без потери передаваемой информации. Две оставшиеся неподтверждаемые услуги используются для безусловного завершения ассоциации с возможной потерей информации.

В число параметров, обрабатываемых в ходе установления ассоциации, входят такие, как имена принимающего (ведомого) и инициирующего (ведущего) прикладных объектов, вызывающий и вызываемые адреса объектов, имя прикладного контекста, данные пользователя, результат, а также целый ряд параметров, семантика

которых определяется сервисом сеансового и представительного уровня. Среди последних – такой, как список представительных контекстов. В общем случае параметр определяет набор абстрактных синтаксисов, для которых требуется создание представительных контекстов.

Установив ассоциацию, ЭСУА не вмешивается в дальнейший диалог, ведущийся СЭПС, пока те (либо поставщик прикладного сервиса) не запросят (не сообщат о) завершении ассоциации.

10.3. Автоматная модель протокола прикладного уровня

На рис. 10.2 и 10.3 изображены диаграммы состояний конечного автомата взаимодействующих систем, в табл. 10.1–10.3 выписаны списки имен элементов множеств входных и выходных событий и состояний автомата.

Таблица 10.1

Список выходных событий

Имя	Смысл/значение
P_CONNECT.REQ (help)	Запрос на соединения со справочником
P_DATA.REQ (get_address)	Запрос на передачу данных для справочника
P_U_ABORT.REQ	Запрос на безусловных разрыв соединения уровня представления пользователем
P_CONNECT.REQ	Запрос на соединения с отвечающей системой
A_TRANSFER_INIT.CONF	Подтверждение установления прикладной ассоциации
A_TRANSFER_ABORT.IND	Индикация безусловного разрыва прикладного соединения
P_RELEASE.REQ	Запрос упорядоченного разрыва разъединения соединения уровня представления
P_RELEASE.RESP	Ответ упорядоченного разрыва разъединения соединения уровня представления
P_DATA.REQ (data)	Запрос на передачу данных
A_TERMINATE.CONF	Подтверждение упорядоченного разрыва прикладного соединения
A_TERMINATE.IND	Индикация на упорядоченный разрыв прикладного соединения
A_TRANSFER_INIT.IND	Индикация установления прикладной ассоциации
P_CONNECT.RESP	Ответ на установление соединения уровня представления
A_DATA.IND	Индикация получения данных

Таблица 10.2

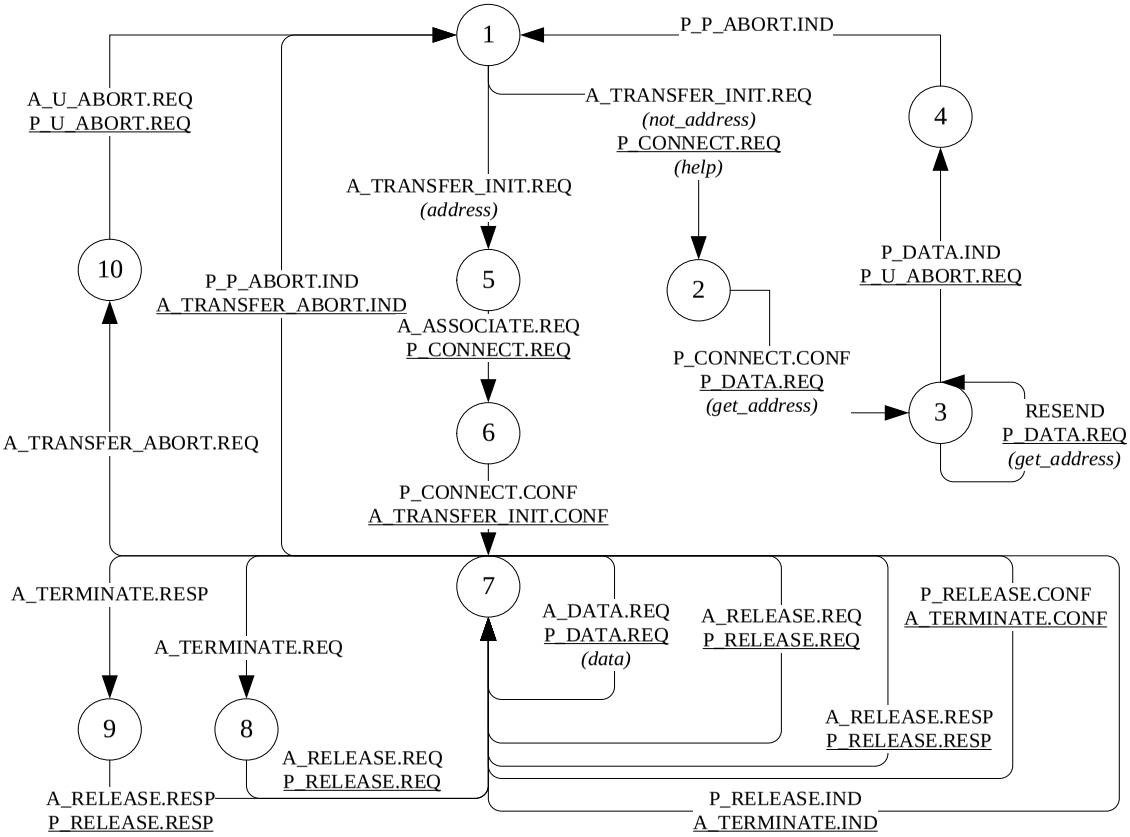
Список имен входных событий

Имя	Смысл/значение
A_TRANSFER_INIT.REQ (not_address)	Запрос на установление прикладной ассоциации, при этом не известен адрес системы
P_CONNECT.CONF	Подтверждение установления соединения уровня представления
RESEND	Повторная посылка данных
P_DATA.IND	Индикация получения искомого адреса системы
P_P_ABORT.IND	Индикация разрыва соединения уровня представления
A_TRANSFER_INIT.REQ (address)	Запрос на установление прикладной ассоциации, при этом известен адрес системы
A_ASSOCIATE.REQ	Запрос на начало установления ассоциации
P_CONNECT.CONF	Подтверждение установления соединения уровня представления
A_U_ABORT.REQ	Запрос разрыва прикладного соединения пользователем
A_TRANSFER_ABORT.REQ	Запрос безусловного разрыва прикладного соединения
A_TERMINATE.REQ	Запрос упорядоченного разрыва прикладного соединения
A_TERMINATE.RESP	Ответ на упорядоченный разрыв прикладного соединения
A_RELEASE.REQ	Запрос упорядоченного разрыва разъединения
A_RELEASE.RESP	Ответ упорядоченного разрыва разъединения
A_DATA.REQ	Запрос на передачу прикладных данных
P_RELEASE.CONF	Подтверждение упорядоченного разрыва разъединения
P_RELEASE.IND	Индикация упорядоченного разрыва разъединения
P_CONNECT.IND	Индикация установления соединения уровня представления
A_TRANSFER_INIT.RESP	Ответ на установления прикладной ассоциации
P_DATA.IND (data)	Индикация получения данных уровня представления

Таблица 10.3

Список состояний автомата

Имя	Смысл/значение
1	Начальное состояние
2	Пришел запрос на установление ассоциации, адрес отвечающей системы не известен
3	Поступило подтверждение на установление соединения уровня представления со справочником
4	Получен данные, содержащие нужный адрес системы
5	Получен запрос на установление ассоциации, адрес отвечающей системы известен
6	Получен запрос на начало установления прикладной ассоциации
7	Получено подтверждение установление соединения уровня представления
8	Получен запрос на упорядоченное разъединение в инициирующей системе
9	Получен ответ на упорядоченное разъединение в инициирующей системе
10	Получен запрос на безусловное разъединение в инициирующей системе
11	Получена индикация установления соединения уровня представления
12	Получен ответ на запрос установления ассоциации
13	Получен запрос на упорядоченное разъединение в отвечающей системе
14	Получен ответ на упорядоченное разъединение в отвечающей системе
15	Получен запрос на безусловное разъединение в инициирующей системе


1
P_P_ABORT.IND
A_U_ABORT.REQ
P_U_ABORT.REQ
A_TRANSFER_INIT.REQ
(not_address)
P_CONNECT.REQ
(help)
4
A_TRANSFER_INIT.REQ
(address)
10
P_P_ABORT.IND
A_TRANSFER_ABORT.IND
5
2
P_DATA.IND
P_U_ABORT.REQ
A_ASSOCIATE.REQ
P_CONNECT.REQ
P_CONNECT.CONF
P_DATA.REQ
(get_address)
3
RESEND
P_DATA.REQ
(get_address)
A_TRANSFER_ABORT.REQ
6
P_CONNECT.CONF
A_TRANSFER_INIT.CONF
7
A_TERMINATE.RESP
A_DATA.REQ
P_DATA.REQ
(data)
A_RELEASE.REQ
P_RELEASE.REQ
P_RELEASE.CONF
A_TERMINATE.CONF
A_TERMINATE.REQ
A_RELEASE.RESP
P_RELEASE.RESP
9
8
A_RELEASE.REQ
P_RELEASE.REQ
A_RELEASE.RESP
P_RELEASE.RESP
P_RELEASE.IND
A_TERMINATE.IND

10.4. Служба справочника

Справочная служба (служба справочника) ВОС

играет значительную роль во взаимодействии открытых систем. Справочник обеспечивает получение справочной информации, необходимой прикладным процессам ВОС, процессам управления ВОС, другим уровневым объектам и телекоммуникационным средствам. К обеспечиваемым службой справочника возможностям относятся, например, использование ориентированных на пользователя имен для обращения к объектам, поддержание соответствия между именем и адресом.

Одна из причин использования имени – стремление освободить пользователя от необходимость знать конфигурацию используемой сети, ее изменений в ходе подключений-отключений подсетей, например локальных сетей и их компонентов, перемещений прикладных процессов из одного пункта сети в другой. Имена объектов, очевидно, должны быть уникальными, так что в больших межнациональных сетях “имя” может состоять из нескольких компонентов (атрибутов), образующих формат “Страна.Подсеть.Система.Имя”. В итоге, если Имя уникально в системе, то “имя” с гарантией будет уникальным и для всей СВОС.

Возможны различные варианты размещения справочной информации в сети – начиная вариантом организации централизованного единственного справочника на всю сеть и кончая вариантом полной его децентрализации, когда каждая система хранит и поддерживает свою копию такого справочника. Оба крайних варианта имеют свои достоинства и недостатки. Обычно на практике применяется некоторый компромисс. Принятая МОС регламентация логической структуры базы справочной информации позволяет не учитывать тот факт, что справочник является скорее распределенным, нежели централизованным.

Каждый пользователь при доступе к справочнику представлен агентом пользователя справочника (АПС). Сам справочник представлен совокупностью системных агентов справочника (САС). Опуская детализацию описания, оба типа агентов называют также просто агентами справочной службы (АСС).

В состав прикладного объекта АПС входят СЭПС – элемент службы доступа к справочнику (ЭСДС) и ОЭПС: ЭСУА – элемент службы управления ассоциацией и ЭСУО – элемент службы удаленных операций. В состав прикладного объекта САС входят

СЭПС – элемент системной службы справочника (ЭССС) и, возможно, ЭСДС, а также ОЭПС, ЭСУА и ЭСУО.

Доступ к справочнику обеспечивается парой ЭСДС, один из которых – часть прикладного объекта АПС, а другой – прикладного объекта САС. Вместе они образуют поставщика услуг по доступу к справочнику. Системная служба справочника обеспечивается парой ЭССС, которые являются частями прикладных объектов САС двух открытых систем.

Предоставляемые пользователям возможности собраны в функциональные группы.

Услуги функционального ядра всегда доступны. Остальные возможности в любой комбинации могут обеспечиваться поставщиком услуг дополнительно. Если доступна какая-либо функциональная группа, то доступны все входящие в нее возможности.

Доступность возможностей определяется уровнем сложности конкретного АСС. В типичном варианте множества подтверждаемых услуг справочной службы в набор примитивов входят примитивы определения адреса и добавления/удаления/изменения элемента содержимого БСИ (база справочной информации). В ряду параметров соответствующих примитивов запросов и ответов, помимо специфицированных имен и адресов, могут использоваться:

- идентификатор запроса, позволяющий прикладному процессу (элементу пользователя) различать запросы, ожидающие ответа справочника;
- идентификатор подлинности, удостоверяющий в том, что пользователь имеет право инициировать внесение изменений в БСИ;
- код ошибки.

Принята следующая типизация ошибочных ситуаций: ошибка контроля доступа, ошибка в атрибуте, аутентификации, в имени, ошибка службы, при обновлении, ошибка дублирования, соединения, неверный САС.

В реализуемой на практике компромиссной схеме размещения справочной информации в сети АСС, получив запрос на определение адреса по специфицированному символическому имени, обращается к своей локальной БСИ (ЛБСИ). Кроме того, обычно в каждой подсети имеется еще и АСС, ответственный за хранение и поддержание экземпляра полной общесетевой БСИ. К этому АСС об-

ращаются прочие агенты подсети в случае, когда они оказываются не в состоянии самостоятельно ответить на запрос (рис. 10.4).

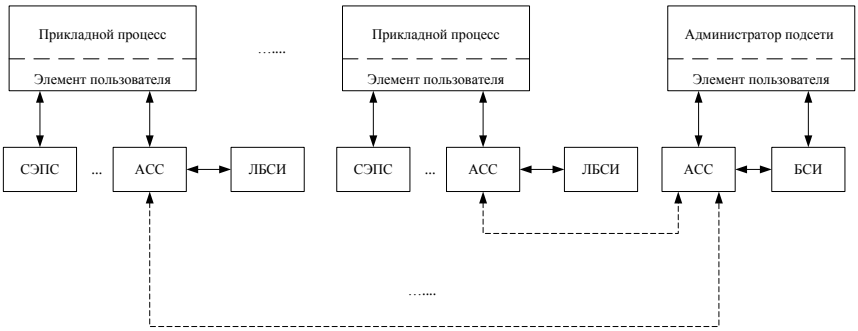


Рис. 10.4. Схема справочной службы

Для обеспечения взаимосвязи между АСС (между АПС и САС или между двумя САС) должна быть установлена ассоциация между соответствующими прикладными объектами. Общий прикладной сервис (ЭСУА) обсуждался в п. 10.2.

Так как каждый локальный АСС поддерживает свою версию БСИ, то центральный АСС подсети, произведя по запросу изменение БСИ и послав подтверждение об этом агенту-инициатору, должен информировать о выполненном изменении и все прочие локальные АСС.

Для обеспечения взаимодействия АПС с САС (и САС с другим САС) протокол доступа к справочнику (и, соответственно, системный протокол справочника) использует службу удаленных операций (ЭСУО). В свою очередь, служба УО обеспечивается протоколом УО в сочетании со службой и протоколом управления ассоциацией, представительной службой и, возможно, при использовании службы и протокола надежной передачи.

10.5. О реализации протокола прикладного уровня на лабораторном комплексе

По необходимости, предельно упростив архитектуру прикладного уровня, создатели лабораторного комплекса оставили разработчикам протоколов этого уровня из ОЭПС лишь поддержку служб ЭСУА, а из СЭПС – СЭПС справочной службы и СЭПС, ответственный за передачу данных.

Другими словами, вследствие ограниченности моделирующих ресурсов у создателей протоколов прикладного уровня в этой версии комплекса нет мощной поддержки таких компонентов ОЭПС, как:

- элемент службы управления завершением, параллельностью и восстановлением (УЗПВ);
- элемент службы удаленных операций (УО);
- элемент службы надежной передачи (НП),

таких СЭПС, как:

- передача, доступ и управление файлами (ПДУФ);
- передача и манипулирование заданиями (ПМЗ);
- виртуальный терминал (ВТ);
- служба обработки сообщений (СОС);
- служба сообщений производственного предприятия (ССПП) и др.

Зато и прикладные процессы (их модели), с которыми здесь приходится иметь дело, исключительно просты (п. 11).

ЭСУА предназначен для управления прикладным взаимодействием.

Услуги, предоставляемые ЭСУА:

- A-ASSOCIATE;
- A-RELEASE;
- A-U-ABORT;
- A-P-ABORT.

Подтверждаемая услуга A-ASSOCIATE позволяет прикладному объекту установить прикладное соединение (ассоциацию) с другим прикладным объектом. На рис. 10.5 приведена схема, в соответствии с которой формируются параметры услуг A-ASSOCIATE, P-CONNECT и S-CONNECT. О таком формировании (параметрической комплектации) шла речь в конце п. 10.1.

Обеспечение услуги A-ASSOCIATE прикладного уровня на лабораторном комплексе в целом аналогично выполнению услуг установления соединения на нижележащих уровнях.

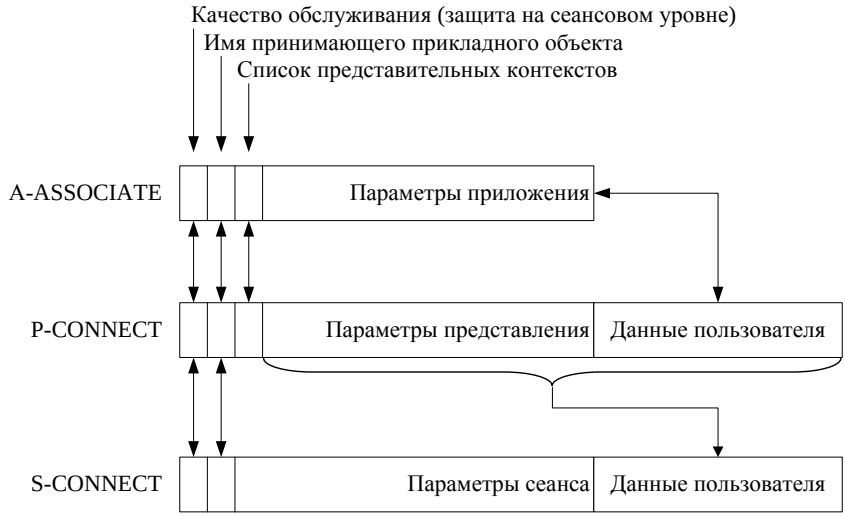


Рис. 10.5. Схема формирования параметров услуг A-ASSOCIATE, P-CONNECT и S-CONNECT

Установив ассоциацию, ЭСУА не вмешивается в дальнейший диалог, ведущийся СЭПС или ЭП, пока последние не запросят об освобождении (разъединении) ассоциации.

Подтверждаемая услуга A-RELEASE позволяет прикладному объекту произвести упорядоченное завершение существующей прикладной ассоциации без потери передаваемой информации, основана на соответствующей услуге уровня представления.

Неподтверждаемая услуга A-U-ABORT позволяет любому пользователю прикладного сервиса выполнить безусловное завершение ассоциации с возможной потерей информации. Основана на соответствующей услуге уровня представления.

Неподтверждаемая услуга A-P-ABORT предоставляет сервис, в соответствии с которым участники ассоциации информируются о безусловном завершение ассоциации с возможной потерей информации. Основана на соответствующей услуге уровня представления.

Поддерживающий передачу данных СЭПС DT (DATA TRANSFER) введен здесь как элемент, чья услуга передачи данных обобщенно заменяет аналогичные по сути услуги отсутствующих

здесь СЭПСов. СЭПС DATA TRANSFER поддерживает три вида услуг:

– установления ассоциации;
– передачи данных;
– завершения ассоциации.

Установление ассоциации включает в себя одну услугу A_TRANSFER_INIT. Эта услуга является подтверждаемой. Её основные задачи состоят в следующем:

– определить адрес вызываемой системы по её имени;
– установить с ней ассоциацию.

Первая задача решается путем использования СЭПС справочной службы, а вторая – с помощью ЭСУА. При использовании СЭПС справочной службы (СпС) возможен отказ в получении адреса. В этом случае услуга должна проанализировать причину отказа: если имя неизвестно, то отправляется соответствующее уведомление ЭП, если же СпС сообщает о возможной ошибке в имени и способе ее исправления, то услуга должна попробовать произвести ее исправление и произвести повторный запрос без извещения ПП. В реальной ситуации это может означать смену формата имен. Например, в именах формата “Страна.Подсеь.Система.Имя” замена точки на запятую.

Передача данных включает в себя одну неподтверждаемую услугу A_DATA. Она основана на соответствующей услуге уровня представления.

Завершение ассоциации включает в себя услуги A_TERMINATE, A_TRANSFER_ABORT. Первая является неразрушающей, вторая – разрушающей. Они основаны на соответствующих услугах ЭСУА.

В связи с отсутствием таких элементов, как удаленные операции и надежная передача, которыми пользуется СЭПС справочной службы, этот СЭПС также использует прямой выход на уровень представления.

Потребность в СЭПС СпС вызвана тем, что пользователь ПП использует для идентификации корреспондирующего ПП его символическое имя (или идентификатор). Поэтому до инициализации запроса на, например, установление ассоциации СЭПС, связанный с ПП, должен сначала выяснить адрес названного корреспонди-

рующего ПП. Следовательно, минимальной функцией СпС является обеспечение пользователя так называемой службой предоставления адреса, которая заключается в получении точного адреса, соответствующего специфицированному символическому имени. Именно эту функцию необходимо реализовать в рамках данного задания.

Таким образом, СпС будет представлена единственной подтверждаемой услугой A-RESOLVE.

Схема функционирования СпС заключается в следующем: после получения запроса на адрес СпС запрашивает в локализованной компоненте справочника необходимый адрес. В случае присутствия необходимого адреса в локальном справочнике СпС возвращает адрес и завершает работу. В противном случае СпС повторно обращается к локализованной компоненте справочника и определяет адрес централизованной компоненты справочника.

После этого СпС устанавливает ассоциацию с централизованной компонентой справочника с помощью ЭСУА, посылает запрос на адрес, получает ответ и производит анализ ответа. Если в ответе присутствует необходимый адрес, то СпС завершает ассоциацию, возвращает ответ и завершает работу.

Если неполучение адреса связано с отсутствием адреса в данный момент и известно время появления адреса (идентификатор ответа равен 1 или 2 (см. ниже описание функции locguide)), то СпС организует повторный запрос адреса в нужный момент времени (при этом необходимо учитывать транзитную задержку), получает адрес, завершает ассоциацию, возвращает ответ и завершает работу.

Если же неполучение адреса связано с именем, то идентификатор типа ответа равен 3 или 4. В последнем случае справочник подыскал к запрошенному имени имя, которое, по его, справочника, мнению, самое близкое. Решение в этой связи, например, коррекцию имен в запросе к справочнику по рекомендации последнего, может принять только тот, кто к справочнику обращался. Пример такого взаимодействия со справочником дан выше. В обоих случаях СпС по возвращению ответа типа 3 или 4 завершает ассоциацию, возвращает ответ и завершает работу.

Обращение к локализованной компоненте справочника производится с помощью функции `locguide`:

Синтаксис:

locguide(<идентификатор>)

где <идентификатор> – строковое выражение, представляющее имя системы, адрес которой необходимо найти или числовое выражение, представляющее адрес системы, имя которой надо найти.

В первом случае функция возвращает буфер следующего типа:

Идентификатор	Адрес	Время1	Время2	Символ1	Символ2

где

А*дрес (число)* – адрес запрашиваемой системы,

Идентификатор (число) – определяет тип ответа:

0 – адрес получен;

1 – адрес будет после *Время1 (число)*;

2 – адрес будет между *Время1* и *Время2 (число)*;

3 – адреса не будет (неизвестное имя);

4 – неизвестное имя, возможно необходимо заменить Символ1 (строка длиной 1) на Символ2(строка длиной 1).

Именем справочника является “Guide”.

Во втором случае функция возвращает строку с именем системы.

Пример:

set addr locguide(“Guide ”)

Из описания схемы функционирования СпС и архитектурной локализации ее СЭПСа (см. рис. 10.4) ясно, что разрабатываемый здесь протокол опирается на ранее созданные и уже отлаженные протоколы уровня представления, сеансового и транспортного уровней.

Контрольные вопросы

1. Составьте схему элементов прикладных служб.
2. Приведите последовательность действий протокола во время выполнения следующих услуг:
   а) установления ассоциации прикладного уровня;
   б) разрыва ассоциации прикладного уровня;
   в) передачи данных прикладного уровня;
2. Опишите услуги, предоставляемые элементом службы управления ассоциацией.
3. Приведите в виде диаграммы состояний-переходов автоматную модель следующих фаз:

а) установления ассоциации прикладного уровня;
б) передачи данных прикладного уровня;
в) разрыва ассоциации прикладного уровня.

4. Опишите службу управления завершением, параллельностью и восстановлением.
5. Можно ли ограничиться лишь адресацией или лишь именованием объектов? Ответ обоснуйте.
6. Дайте общее описание службы справочника.

11. ПРИКЛАДНЫЕ ПРОЦЕССЫ И ЭЛЕМЕНТ ПОЛЬЗОВАТЕЛЯ

11.1. Общие положения

Реализация прикладных процессов и элемента пользователя завершает домашнее задание. Хотя в реальной жизни функции и структуры таких процессов могут быть исключительно сложными, но здесь, в этой части ДЗ, логика работы прикладных процессов проста: в основном это вызовы событий по таймеру, посылающих некоторые данные, и некоторая реакция на приходящие данные. По-видимому, в рамках лабораторного практикума написание прикладных процессов несколько проще, чем написание протоколов.

Элемент пользователя обеспечивает прикладному процессу интерфейс со службами обработки распределенной информации, предоставляемыми прикладным уровнем. Обычно эти элементы реализованы в виде комплекта библиотечных процедур и функций, которые подключаются к прикладному процессу, желающему получить доступ к имеющимся службам распределенной информации. Примитивы интерфейса между ПП и ЭП не обязаны совпадать со стандартными примитивами службы, обеспечиваемыми конкретным прикладным элементом. Отображение между этими двумя комплектами примитивов реализует ЭП; иными словами, ЭП осуществляет взаимное преобразование формы общения между средами реальных и открытых систем. Благодаря этому могут применяться примитивы интерфейса существующих производителей программного обеспечения.

В рамках лабораторного практикума ЭП отвечает за подобное отображение между интерфейсами ПП и прикладного уровня.

11.2. Дополнительные операторы встроенного языка

Для написания прикладных процессов понадобятся дополнительные операторы встроенного языка, перечисленные в п. 11.2.1.– 11.2.4

11.2.1. Операторы вывода текста (say, text)

Синтаксис:
say <строка> <панель>
text <строка> <панель>
Пример:
say "говорит 'мяу'"+#13+"движение 'нет'"+#13+"цвет 'серый'"+#13+"запах 'нет'" 0
text "слышит '"+$sound+"'"+#13+"видит движение '"+$move+"'"+#13+"видит цвет '"+$color+"'"+#13+"чувствует запах '"+$smell+"'" 1

Операторы выводят текстовые сообщения на панели окна прикладных процессов. Оператор “say” выводит сообщения на верхние панели, оператор “text” на нижние. Параметр “панель” определяет правую или левую панели: 0 – левая, 1 – правая. Для разделения строки на несколько строк необходимо использовать символ “Возврат каретки”, обозначаемый “#13”.

11.2.2. Оператор вывода изображения

Синтаксис:
image <имя файла> <панель>
Пример:
image “tomcat.bmp” 1

Оператор выводит изображение в формате “bmp” на центральные панели окна прикладных процессов. Параметр “панель” определяет правую или левую панели: 0 – левая, 1 – правая. Параметр “имя файла” задает имя файла с изображением и, если файл не находится в текущем каталоге, путь к нему.

11.2.3. Оператор вывода видеозаписи

Синтаксис:
video <имя файла> <панель>
Пример:

```
video "tomcat.avi" 1
```

Оператор выводит видеозапись в формате “avi” на центральные панели окна прикладных процессов. Параметр “панель” определяет правую или левую панели: 0 – левая, 1 – правая. Параметр “имя файла” задает имя файла с видеозаписью и, если файл не находится в текущем каталоге, – путь к нему.

11.2.4. Оператор получения текущего системного времени

Синтаксис:
systime <переменная-число>
Пример:
systime cursystime
Оператор присваивает переменной значение текущего системного времени.

11.3. Реализация элемента пользователя

Как уже отмечалось, ЭП в рамках данной части ДЗ отвечает за отображение между интерфейсами ПП и прикладного уровня. ПП оперирует только двумя услугами: пересылки данных и завершения ассоциации. Таким образом, ЭП при запросе на пересылку данных должен установить ассоциацию при помощи услуги A_TRANSFER_INIT, после чего переслать данные при помощи услуги A_DATA. В случае прихода запроса на завершение ассоциации ЭП завершает ассоциацию при помощи услуги A_TERMINATE. В случае прихода запроса на пересылку данных поименованному прикладному объекту, с которым уже установлена ассоциация, ЭП просто передает данные. Если же приходит запрос объекту с другим именем – завершает ассоциацию, устанавливает новую и передает данные.

11.4. Реализации прикладного процесса на лабораторном комплексе

Разберем следующий пример. Система А посылает пять раз некоторые данные системе В с промежутком каждый раз в 500 системных единиц времени (СЕВ), после чего она ожидает ответа системы В в течение 1500 СЕВ. Если это происходит, то процесс повторяется сначала, в противном случае процесс заканчивается. Система В ожидает пятикратного прихода данных и отвечает на

них, пока таких пятерок данных не больше десяти. Реализуется это следующим образом: во время инициализации система А устанавливает необходимые переменные и посылает запрос на установление ассоциации. После прихода подтверждения установления ассоциации система А начинает посылать данные с заданным промежутком времени. После завершения пересылки пяти данных система А ожидает ответа и, если он не приходит в заданный промежуток времени, завершает работу. Фрагменты реализации на лабораторном комплексе представлены в табл. 11.1.

С точки зрения элементарного взаимодействия простых моделей двух животных данный сценарий можно проинтерпретировать как весеннюю серенаду кота кошке. Кот поёт, пока кошка его периодически подбадривает. Если такого одобрения хотя бы раз не будет, то замолкает и серенада.

Таблица 11.1

Фрагмент реализации прикладного процесса

Система А		**Система В**	
Событие	**Код**	Событие	**Код**
Инициализация	; объявить счетчик посланных данных declare data_send integer	Инициализация	; объявить счетчик принятых данных declare data_receive integer
	; поставить на таймер запрос на передачу данных timer t1 AP_DATA 500		; объявить счетчик ответов declare answers integer
AP_DATA	; послать данные $data down UE_DATA.REQ userdata $data name “SystemB”	UE_DATA.IND	;осуществление вывода на окно прикладных процессов сообщений о принятых данных …
	; осуществление вывода на окно прикладных процессов сообщений о пересылаемых данных say "Мяу-у-у-у-у-у-у" 0 out “Кот поет серенаду” image “tomcat.bmp” 0		; увеличить счетчик принятых данных set data_receive $data_receive+1
	; увеличить счетчик посланных данных set data_send $data_send+1		; если данных принято меньше 5, то продолжать принимать if ($data_receive<5) && ($answers>10) exit

Продолжение табл. 11.1

	; если данных передано меньше 5, то продолжать передачу if $data_send<5 next		; увеличить счетчик ответов set answers $ answers +1
	;включить таймер ожидания ответа timer t2 AP_WAIT 1000		;послать данные $data down UE_DATA.REQ userdata $data name “SystemA”
	; обнулить счетчик посланных данных set data_send 0 timer t1 AP_DATA 1500 goto exit		; осуществление вывода на окно прикладных процессов сообщений о пересылаемых данных say "Мур-р-р-р-р-р-р-р-р-р-р" 1 out “Кошка одобряет” image “kitty.bmp” 1
	; продолжать передачу данных next: timer t1 AP_DATA 500		; обнулить счетчик принятых данных set data_receive 0
	exit:		exit:
AP_WAIT	; отключить передачу данных untimer $t1		
UE_DATA.IND	; отключить ожидание ответа untimer $t2		
	;осуществление вывода на окно прикладных процессов сообщений о принятых данных …		
	exit:		

А.Ю. Никифоров, В.А. Русаков

ВЗАИМОСВЯЗЬ ОТКРЫТЫХ СИСТЕМ
(основы теории и практики)

Учебное пособие

Москва 2010