РОССИЙСКАЯ АКАДЕМИЯ НАУК
СИБИРСКОЕ ОТДЕЛЕНИЕ
А В Т О М Е Т Р И Я
№ 1 1993

УДК 621.391

П. М. Песляк
(Новосибирск)

АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ЦЕНТРА КОММУТАЦИИ ПАКЕТОВ

Рассматриваются вопросы построения архитектуры программного обеспечения узла коммутации пакетов для интегральных цифровых сетей. Описана реализация супервизора (исполняющей системы) для мультипроцессорного варианта узла коммутации. Приводится описание архитектуры прикладного обеспечения — пакета, реализующего второй и третий уровни протокола X.25.

Введение. Институт автоматики и электрометрии СО РАН в течение ряда лет ведет исследования и разработки в области создания базовых сетей передачи данных, соответствующих рекомендации X.25 МККТТ.

В процессе исследования были разработаны различные типы аппаратных средств. Этот факт потребовал от создателей программных средств спроектировать такую архитектуру программного обеспечения, которая позволила бы максимально просто адаптироваться к различным аппаратным решениям. Для достижения этой цели разработаны архитектуры прикладного обеспечения (реализация протоколов X.25) и исполняющей программы-супервизора. Проведены реализация протокольной части и несколько реализаций супервизора: для мультипроцессорной конфигурации и однопроцессорных конфигураций трех типов. При разработке макетов систем использованы различные типы процессоров и микропроцессоров семейства «Электроника 60», что и определило некоторые подходы при реализации программного обеспечения.

1. Архитектура программного обеспечения мультипроцессорного центра коммутации пакетов. Выбор архитектуры программного обеспечения был обусловлен следующими особенностями задачи и протоколов X.25:

— независимость потоков данных по линиям связи;

— независимость процедур обработки по разным линиям связи;

— «естественное» распараллеливание по уровням протокола X.25;

— многовариантность архитектуры аппаратных средств (от однопроцессорной конфигурации до многопроцессорной системы);

— многовариантность линейных адаптеров (от «интеллигентных» до простейших).

Для удовлетворения всем этим требованиям была выбрана архитектура программного обеспечения на основе понятий абстрактных типов данных (процессов, мониторов, семафоров и т. п.). Реализация системного и прикладного обеспечения проводилась на языке высокого уровня PL-11, который можно отнести к классу «структурированных» ассемблеров.

1.1. Процесс — это программа, которая может исполняться на любом процессоре системы параллельно с другими процессами. Каждый процесс резервирует собственную область данных для хранения своих переменных и стека.

В мультипроцессорных системах эта область данных расположена в той части памяти, которая в равной степени доступна каждому процессору. Каждый процесс имеет секцию инициации, при запуске которой выдаются запросы на резервирование области данных под переменные процесса и стек. В этой же

секции производится инициация всех переменных процесса. Вызов секции инициации осуществляется процессом-«родителем», и после завершения исполнения этой секции в системе начинает самостоятельно «существовать» еще один процесс. Естественно, что процесс-«родитель» может передать все необходимые параметры при инициации.

Как правило, процесс — это программа, имеющая безусловный цикл в своем теле, хотя и предусмотрен системный вызов для уничтожения процесса (фактически уничтожается блок управления процессом и его область данных).

Такой подход к возникновению и уничтожению процессов предполагает, что в системе при ее запуске всегда есть один процесс, в функцию которого и входит, как правило, инициация всех своих потомков, может быть и косвенных, через поколения. В нашей реализации данной архитектуры для ЭВМ типа PDP-11 указатель на локальные данные процесса хранится в регистре общего назначения.

1.2. Одним из основных вопросов при параллельном программировании является вопрос о выборе средств синхронизации параллельно исполняющих программ-процессов. Нами было использовано понятие «монитор».

Монитор — это набор специальным образом запрограммированных процедур для работы со структурами данных, которые совместно используются несколькими процессорами. Для хранения этих структур, а также собственных переменных каждый монитор использует соответствующую по размерам область данных в общей памяти. Эта область в равной степени доступна каждому процессу системы.

Каждый монитор, как и процесс, имеет процедуру инициации. При вызове этой процедуры резервируется область системных данных и производится инициация всех внутренних переменных монитора, а также и всех структур данных, служащих для взаимодействия процессов.

Секция инициации любого монитора вызывается из процесса-«родителя», который может передать в монитор параметры, а обратно (обязательно) получает указатель на область данных монитора. Этот указатель процесс-«родитель» передает в качестве параметра тем процессам-«сыновьям», которые должны взаимодействовать друг с другом и используют общие структуры данных, принадлежащие этому монитору.

В нашей реализации данной архитектуры для ЭВМ типа PDP-11 указатель на локальные данные монитора должен находиться в регистре общего назначения процессора.

Таким образом, процесс обращается к общим данным в два этапа: 1) загружает указатель на локальные данные монитора в регистр общего назначения; 2) вызывает соответствующую процедуру монитора, внутри которой записаны действия по модификации общих данных монитора.

Механизм реализации монитора гарантирует, что доступ к его разделяемым структурам получает лишь один процесс, остальные ставятся в очередь ожидания и получают права доступа по освобождении монитора.

1.3. При анализе прикладного уровня программного обеспечения выделены структуры специального типа — очереди с дисциплиной обслуживания: «первый пришел — первый обслуживается». Конечно, эти структуры можно было реализовать с помощью специального монитора — набора соответствующих мониторных процедур. Но так как в нашем случае операции с очередями весьма критичны по времени, эти структуры и операции с ними выделены в специальные библиотеки. Предложены и реализованы два типа очередей: очередь с блокировкой ожидающего процесса, очередь без блокировки.

Последний тип очереди представлен набором простых процедур работы со списком, которые оптимизированы по времени исполнения. Реализованы операции создания очереди, добавления буфера в очередь, извлечения буфера из очереди, проверки состояния очереди.

Очередь с блокировкой имеет более сложную структуру, и операции по извлечению и добавлению буферов имеют побочные эффекты — изменения состояния процессов.

Операция извлечения буфера из очереди с блокировкой приводит к блокировке процесса в случае, если очередь пуста. Данный процесс разблокируется в тот момент времени, когда в очередь поступит хотя бы один буфер.

Операция добавления буфера к очереди с блокировкой может вызвать разблокировку ожидающего процесса.

1.4. Архитектура второго и третьего уровней пакета X.25. При проектировании архитектуры пакета X.25 учтены особенности, присущие задаче коммутации пакетов: естественное распараллеливание, независимость потоков передачи и приема. Разработке архитектуры пакета предшествовало создание формального описания протоколов второго и третьего уровней X.25 в виде диаграммы переходов конечного автомата с описанием событий, действий и переходов. Конечный автомат, реализующий протокол второго уровня, представляет собой монитор с шестью точками входа: две — для приема и передачи кадров, поступающих из линии; две — для приема и передачи кадров на третий уровень; пятая точка входа — для приема запросов с третьего уровня, шестая — вход таймера. Событиями для данного монитора служат кадры, принятые на линии, и запросы, приходящие с третьего уровня.

Каждое состояние — это таблица адресов процедур для обработки всех событий, а смена состояния сводится к смене указателя на данную таблицу. Взаимодействие монитора второго уровня с нижним и верхним уровнями происходит с помощью передачи указателей на буфера, содержащие информацию.

По такому же принципу построена архитектура третьего уровня: существует монитор, в котором реализован автомат третьего уровня. Обмен информацией производится с помощью передачи указателей на буфера.

Для взаимодействия между уровнями выделены пары процессов, которые получают буфера из соответствующих очередей, обрабатывают их и вызывают соответствующие процедуры в мониторах второго и третьего уровней.

На уровне взаимодействия с оборудованием передачи данных (линейными адаптерами) находятся процессы специального вида — драйверы. Их особенность в том, что они имеют специальные секции, которые запускаются по прерываниям от аппаратуры и по завершении обработки прерываний ставят процессы-драйверы в очередь на исполнение.

2. Программная модель мультипроцессорного центра коммутации пакетов [1]. Мультипроцессорный центр коммутации пакетов (ЦКП) состоит из нескольких (от 1 до 4) микроЭВМ типа «Электроника 60», оснащенных диспетчером памяти, который позволяет обращаться к общим ресурсам ЦКП — системам ввода-вывода и памяти. Каждый процессор имеет локальную память 32 Кбайта, модуль ПЗУ 4 Кбайта, интерфейс консоли.

2.1. Основой аппаратной архитектуры ЦКП является модуль управления памятью (диспетчер памяти активный — ДПА). Этот модуль с программной точки зрения позволяет отобразить страницу локальной памяти размером 4 Кбайта на любую страницу такого же размера в системной области. Модуль ДПА имеет 16 регистров и возможность отключения любой страницы и всего диспетчера.

2.2. Специальное пространство размером 8 Кбайт в старших адресах системной области занимает система ввода-вывода. В ее адресном пространстве расположены диспетчер очереди задач и линейные адаптеры. Диспетчер очереди задач перехватывает все прерывания от линейных адаптеров и выдает в своем регистре вектор самого приоритетного из них.

2.3. Общая память. Все процессоры через ДПА могут обратиться в любую область системной памяти. Ее размер может достигать 2 Мбайт (22-разрядный адрес). В этой области системной памяти расположены все буфера и области данных мониторов и процессоров, а также управляющий блок системы.

3. Супервизор (исполняющая система) для мультипроцессорного варианта ЦКП. При реализации супервизора учитывались аппаратные особенности ЦКП. Все процессоры при запуске системы программного обеспечения загружают копию супервизора и прикладную задачу в свою локальную память. После запуска системы один из процессоров инициирует системную память, записывает в нее блок управления системой и ставит в очередь на

исполнение основной процесс системы. Функция этого процесса состоит в том, чтобы инициировать все мониторы и процессы системы. После этого процессы начинают самостоятельную жизнь и при помощи мониторов взаимодействуют между собой.

3.1. Управление памятью. В описываемом варианте мультипроцессорного супервизора вся общая память системы разбита на две части: системную область и область буферов.

Под системную область отведено 32 Кбайта. В этой области находятся все общие переменные процессов и мониторов, структуры, управляющие очередями, структуры, управляющие буферным пространством, и т. д.

Под буферное пространство отведена оставшаяся часть системной памяти; пространство разбито на блоки по 128 байт для приема и передачи кадров. Механизм выделения буферного пространства описан в [2].

В момент запуска супервизор настраивает диспетчер памяти таким образом, что через верхние страницы локальной памяти процессор получает доступ к системной области (общим данным системы).

С помощью последней страницы диспетчера памяти производится доступ к области ввода/вывода, а с помощью предпоследней страницы — к данным в текущем буфере.

В процессе работы такая настройка сохраняется, что позволяет любому процессору обращаться к общим данным без потери времени. Доступ к данным, находящимся в буфере кадра, происходит достаточно редко — только в момент обработки запроса на установление соединения. Поэтому обращение к разным буферам кадров происходит через одну страницу доступа.

Супервизор имеет точки входа для обслуживания процессов:

— зарезервировать область системной памяти;
— освободить область системной памяти;
— зарезервировать буфер;
— освободить буфер;
— зарезервировать заголовок буфера;
— освободить заголовок буфера.

3.2. Управление процессами. Управление процессами производит специальный модуль супервизора, имеющий ряд точек входа: создания структур, необходимых для функционирования процесса, создания управляющих структур монитора, очереди. Другие процедуры этого модуля служат для синхронизации процессов. Пара процедур (ENTR и LEAVE) применяется для взаимоисключения процессов, обрабатывающих общие структуры данных.

Некоторые процедуры предназначены для приостановки процессов на определенный промежуток времени для ожидания некоторого события.

3.3. Управление очередями. В описываемом варианте мультипроцессорного супервизора практически все взаимодействие между процессами производится с помощью специальных структур — очередей. Реализовано два типа очередей: простая и очередь с ожиданием.

Дисциплина обслуживания в любом типе очередей: «первый пришел — первый вышел».

Очередь с ожиданием отличается от простой тем, что при обращении к ней может возникнуть побочный эффект: процесс может быть приостановлен. Данное свойство таких очередей используется как дополнительный механизм синхронизации процессов. Например, процесс «потребитель» при обращении к пустой очереди буферов будет приостановлен и вновь запущен только в тот момент, когда процесс «производитель» запишет в очередь хотя бы один буфер.

Управление очередями реализовано в виде набора процедур, в которые встроены механизмы взаимоисключения.

Заключение. Вышеописанная архитектура программного обеспечения мультипроцессорного центра коммутации пакетов позволяет соблюдать принципы модульного программирования, отлаживать процессы и мониторы по мере реализации и тестировать отлаживаемый модуль в различных конфигурациях.

Хорошо определенное управляющее ядро (супервизор) дает возможность эффективно использовать особенности аппаратной части и «закрывать» от

прикладных уровней детали конкретной аппаратной реализации. В частности, опробованы: однопроцессорный вариант без диспетчера памяти, однопроцессорный вариант с различными вариантами диспетчера памяти, мультипроцессорный вариант с общим полем памяти и внешними устройствами. Переход с одной конфигурации аппаратных средств на другую потребовал лишь перепрограммирования ряда процедур в супервизоре, при этом прикладная часть оставалась неизменной.

СПИСОК ЛИТЕРАТУРЫ

1. **Бобко В. Д., Головин В. Ф., Золотухин Ю. Н.** Высокопроизводительный узел коммутации для интегральной цифровой сети // Автометрия.—1993.—№ 1.
2. **Бредихин С. В., Иванченко А. Я., Песляк П. М., Щербакова Н. Г.** Механизм распределения буферной памяти в протоколе канального уровня // Локальные вычислительные сети.