РАСПРЕДЕЛЕННАЯ ОБРАБОТКА ИНФОРМАЦИИ
(Вычислительные системы)
1984 год Выпуск 105

УДК 681.322.06:681.3.323

ОБ ОРГАНИЗАЦИИ КОММУНИКАЦИЙ МЕЖДУ ПРОЦЕССАМИ В ВЫЧИСЛИТЕЛЬНОЙ СИСТЕМЕ МИКРОС

А.Ф.Задорожный, В.В.Корнеев, М.С.Тарков

Вычислительная система (ВС) с программируемой структурой [1,2] представляет собой совокупность элементарных машин (ЭМ), объединенных программно-управляемой из этих машин сетью линий связи. Каждая ЭМ включает: вычислительный модуль (процессор с памятью) и системное устройство, подключенное к ней. Последнее функционирует под программным управлением вычислительного модуля и имеет v входных и v выходных полюсов, служащих входными и выходными полюсами ЭМ, $v \in \{1,2,\ldots\}$. Оно обеспечивает прием команд управления и данных с v входных полюсов, подключенных посредством линий связи к выходным полюсам v соседних ЭМ, и выдачу команд управления и данных, выработанных в вычислительном модуле или поступивших с входных полюсов на выходные полюса для передачи в одну или несколько соседних ЭМ.

Построение ВС с программируемой структурой базируется на идее удовлетворения комплекса требований, предъявляемых к современным универсальным средствам обработки информации [3-5], путем программной специализации ВС на решаемые задачи. В рамках этого подхода ВС должны обеспечивать пользователю, как минимум, высокоэффективные параллельные вычисления наряду со всеми возможностями, предоставляемыми локальными сетями ЭВМ. Иными словами, эти ВС должны функционировать в режиме, предусматривающем решение потока или набора заданий, каждое из которых представлено последовательной или параллельной программой. В первом случае программа исполняется на одной машине ВС, во втором - на подсистеме с требуемым программой числом машин (вплоть до всех машин ВС).

Одним из перспективных подходов к обеспечению указанного функционирования ВС, по нашему мнению, может быть следующий. Все

машины ВС должны работать в мультипрограммном режиме, что позволит им выступать в роли узла коммутации межмашинных обменов, осуществляющего доступ к удаленным файлам и другие функции распределенной обработки данных, и собственно вычислителя, исполняющего одну или несколько программ, каждая из которых либо последовательная программа, либо ветвь параллельной [1]. Для эффективного исполнения параллельных программ предлагается способ выделения подсистем, каждая из которых работает в монопрограммном режиме. Эффективность предлагаемого подхода демонстрируется на примере его реализации в ВС МИКРОС [6,7].

I. Виртуальные подсистемы - средство реализации межпроцессовых коммуникаций в ВС с программируемой структурой

Коммуникации между процессами в локальных сетях ЭВМ осуществляются посредством логических (виртуальных) каналов, обеспечивающих связь между парой процессов на то время, когда им необходимо обменяться информацией. Если взаимодействующие процессы протекают в одной и той же машине, то коммуникации между ними осуществляются через виртуальный канал на базе разделяемой памяти, контролируемой семафорами или мониторами [8]. Если же процессы порождены в разных машинах, то виртуальный канал, задающий коммуникацию, представляет собой путь, образованный последовательностью физических межмашинных линий связи и транзитных машин в совокупности с программными средствами, обеспечивающими функционирование линий связи и маршрутизацию сообщений в транзитных ЭМ.

В ВС с развитыми возможностями по организации параллельных вычислений используется более общий аппарат коммуникаций между процессами - аппарат виртуальных подсистем [2,6]. Каждая виртуальная подсистема создается на время исполнения одной параллельной программы и состоит из машин, в которых размещены ветви программы, и объединяющих эти машины линий связи. Последние используются для образования логических каналов. Неотъемлемой частью подсистемы служат путевые процедуры, размещаемые в каждой машине и обеспечивающие маршрутизацию информации в подсистеме.

Наряду с виртуальными подсистемами, исполняющими параллельные программы пользователей, в ВС с программируемой структурой присутствуют подсистемы, исполняющие служебные функции распределенной обработки данных, присущие локальным сетям ЭВМ, в частности: реализацию доступа к удаленным файлам, диагностику неисправ-

ностей, передачу программ и массивов данных по межмашинной сети связи. Такие подсистемы мы будем далее называть служебными. Для реализации межпроцессовых коммуникаций в этих подсистемах используются универсальные протоколы межмашинных взаимодействий и путевые процедуры, обеспечивающие бездедлоковое функционирование межмашинной сети связи [9].

При функционировании ЭМ в мультипрограммном режиме команды одной параллельной программы перемежаются командами операционной системы и других параллельных программ, что может привести к резкому замедлению счета параллельных программ. Для эффективного исполнения параллельной программы пользователя при создании виртуальных подсистем должно быть использовано "замораживание" в них всех процессов, кроме тех, что исполняют параллельную программу. Замораживание процесса состоит в удалении его из рассмотрения операционной системы на время существования данной виртуальной подсистемы. Машины подсистемы работают под управлением специализированной для параллельных вычислений операционной системы, каковой является ядро операционной системы [10,11], а межмашинные обмены выполняются специализированными драйверами, обеспечивающими максимальную скорость передачи данных по ее линиям связи. Тем самым как бы организуется монопрограммный режим работы машин виртуальной подсистемы над одной параллельной программой. По истечении времени существования подсистемы или по прерыванию от таймера возобновляется исполнение всех загруженных в машины процессов. Машины, входившие в подсистему, снова начинают функционировать в мультипрограммном режиме.

2. Межпроцессовые коммуникации в ЭМ

В работах [10,11] предлагается ядро операционной системы ЭМ ВС с программируемой структурой. Ядро операционной системы ЭМ осуществляет согласование работы программ в ЭМ посредством моделирования на одном процессоре ЭМ множества виртуальных процессоров, каждый из которых выполняет одну породившую его программу. Исполнение программы виртуальным процессором образует вычислительный процесс.

Взаимодействие процессов осуществляется посредством синхронизирующего примитива **when** $\Phi(X)$ **do** $X \leftarrow G(X)$ **od**. Переключение процессора с одной программы на другую определяется значением преди-

ката $\mathfrak{S}(X)$, который определен на множестве X общих (глобальных) для всех исполняемых программ объектов, называемых семафорами в отличие от индивидуальных (локальных) для каждой программы переменных. Семафор S представляет собой тройку $\langle sem(S), m(S), i(S)\rangle$. Здесь sem(S) - значение семафора; m(S) - строго упорядоченное множество, состоящее из sem(S) элементов; i(S) - множество допустимых операций над семафором S. Если не требуется различать и использовать элементы множества m(S), а важно лишь знать значение мощности этого множества, то само множество может отсутствовать. Семафоры этого класса называют неструктурированными. Операции над неструктурированными семафорами изменяют их значение. В случае, когда элементы множества m(S) рассматриваются как буфера, посредством которых осуществляются коммуникации между процессами, семафор S называется структурированным. Операции над структурированным семафором S исключают/включают буфера в множество m(S) и изменяют значение семафора S.

Коммуникации между процессами X и Y, исполняющимися в одной физической ЭМ, описываются с помощью структурированных семафоров xF, yF и memory. Коммуникации происходят следующим образом. Процесс (например, X), пересылающий данные, берет посредством выполнения действия

k: when L_X=k $\wedge$ memory > 0 do memory ← memory - 1(α) od

буфер α. Далее в процессе X массив α заполняется данными, которые нужно передать процессу Y. По окончании заполнения выполняется действие

l: when L_X = l do yF ← yF + tail, 1(α) od,

которое внесет буфер α в очередь yF.

Выбор тем же процессом X данных, пришедших из процесса Y ,совершается следующим образом:

p: when L_X = p $\wedge$ xF > 0 do xF ← xF - 0,1(α) od.

Извлеченный из очереди xF буфер α содержит пришедшие из процесса Y данные. По окончании обработки этих данных выполняется действие:

q: when L_X = q do memory ← memory + 1(α) od.

Тем самым использованный буфер возвращается в список свободных массивов memory .

Управление мультипрограммной ситуацией в ЭМ описывается с помощью структурированных семафоров ready и wait, представляющих очереди информационных векторов [10,11], готовых к выполнению и

ждущих процессов соответственно. Если значение предиката текущего процесса равно false, то загруженный в процессор информационный вектор посылается в очередь ждущих процессов wait, из очереди готовых процессов ready выбирается информационный вектор, который загружается в процессор. Ядро операционной системы ЭМ ВС МИКРОС реализовано на базе языка программирования MACRO-11, объем ядра составляет около 7 К слов. Средний объем одной подпрограммы ядра, реализующей операцию над семафором, составляет около 80 команд языка MACRO-11.

В общем случае взаимодействующие процессы могут быть порождены в различных ЭМ. Коммуникация между такими процессами предполагает обязательное использование линий межмашинного обмена.

3. Организация линий межмашинного обмена

При отождествлении входного полюса q $ЭМ_j$ с выходным полюсом p $ЭМ_i$ образуется линия межмашинного обмена (i_p, j_q), которая обеспечивает передачу информации с выходного полюса p $ЭМ_i$ на входной полюс q $ЭМ_j$, $p,q \in \{1,2,\ldots,v\}$, где v - степень вершины графа межмашинных связей. В передающей $ЭМ_i$ имеются структурированные семафоры $memory_i$ и $ВЫХ^i_p$, играющие роль цепочек пустых и полных буферов, традиционно используемых при работе с внешними устройствами. Принимающая $ЭМ_j$ содержит структурированные семафоры $memory_j$ и $ВХ^j_q$, являющиеся цепочками пустых и полных буферов на приемном конце линии (i_p, j_q).

Передача данных по линии (i_p, j_q) инициируется асинхронно при наличии хотя бы одного полного буфера в $ВЫХ^i_p$ и пустого буфера в $ВХ^j_q$. Такая организация линий сводит передачу данных между процессами X и Y, протекающими в $ЭМ_i$ и $ЭМ_j$, соответственно к вышеописанной с той лишь разницей, что в процессе X данные заносятся не в семафор yF, а в $ВЫХ^i_p$. Аналогично выборка процессом Y данных, пришедших из процесса X, осуществляется не из yF, а из $ВХ^j_q$.

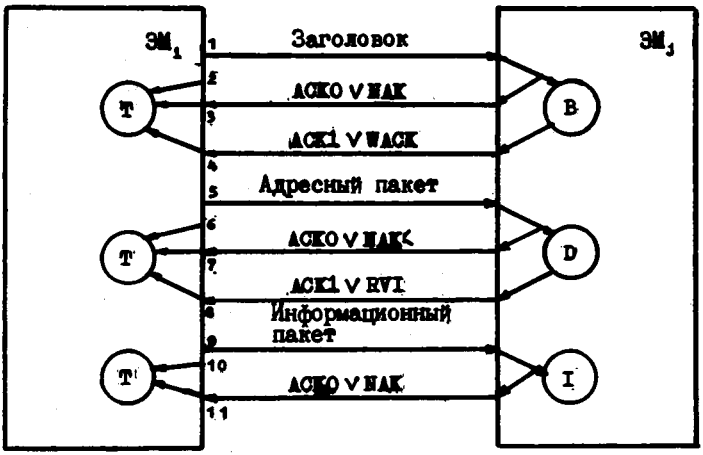
Преобразования программ, обусловленные перераспределением процессов между машинами подсистемы, заключаются в следующем. Пусть два процесса p_1 и p_2 запрограммированы так, что должны выполняться в одной ЭМ, и для передачи данных из p_1 в p_2 используется структурированный семафор S. Тогда если процессы p_1 и p_2 должны протекать соответственно в $ЭМ_i$ и $ЭМ_j$, связанных линией (i_p, j_q), то операция занесения данных в семафор S в p_1 заменяется на опе-

рацию занесения в $ВЫХ_p^i$, а операция выборки данных в p_2 – на операцию выборки из $ВХ_q^j$. Если p_1 и p_2 должны протекать в непосредственно не связанных линией машинах, то коммуникация между ними выполняется аналогично вышеописанной с той разницей, что вместо одной линии связи используется цепочка линий, передача данных по которым идет под управлением путевой процедуры. Наоборот, пусть p_1 и p_2 запрограммированы для протекания в $ЭМ_i$ и $ЭМ_j$ соответственно, и обмен между ними выполняется через линию (i_p, j_q). Если эти процессы размещаются в одной ЭМ, то вводится структурированный семафор S, и операции по занесению в $ВЫХ_p^i$ и выборки из $ВХ_q^j$ заменяются на операции с семафором S. Таким образом, программы, записанные на входном языке ядра операционной системы ЭМ, могут легко перестраиваться на разное число машин, вплоть до выполнения их на одной ЭМ.

4. Реализация межмашинных обменов в ВС МИКРОС

Детальное рассмотрение работы линий связи, требующееся при изложении способа организации монопрограммного режима линии подсистемы, проведем на примере ВС МИКРОС. Это не снижает общности рассмотрения, а просто вводит вместо гипотетических управляющих символов протокола межмашинных обменов те символы, которые используются в ВС МИКРОС.

Обмен по линии (i_p, j_q) осуществляется под управлением процесса $output_p^i$ на передающем конце и процесса $input_q^j$ на приемном конце, которые взаимодействуют с драйверами системных устройств $СУ_i$ и $СУ_j$ передающей и принимающей ЭМ (протокол межмашинного обмена показан на рис.I). Драйверы в совокупности с аппаратными компонентами системного устройства реализуют протокол BSC физического уровня управления линиями связи [I2]. Выбор этого протокола, уступающего по быстродействию протоколу HDLC, обусловлен бо́льшим разнообразием в BSC управляющих символов, чем в управляющей части кадра HDLC. Наличие большого числа управляющих символов, значения которых заранее не фиксированы, позволяет иметь требуемую гибкость, необходимую при создании ВС МИКРОС как экспериментальной [6]. По завершению отработки механизма функционирования линий связи в ВС МИКРОС возможен переход к более эффективному протоколу HDLC.



1 - начало вывода заголовка, 2 - конец вывода заголовка, 3 - прием АСКО о правильном приеме заголовка, 4 - прием АСК1 о выделении свободного буфера, 5 - начало вывода адресного пакета, 6 - конец вывода адресного пакета, 7 - прием АСКО о правильном приеме адресного пакета, 8 - прием АСК1 о невозможности дедлока, 9 - начало вывода информационного пакета, 10 - конец вывода информационного пакета, 11 - прием АСКО об окончании связи, T - таймер, B - выделение свободного буфера, D - анализ возможности дедлока, I - обработка принятого сообщения.

Рис. I

Передача данных по линии (i_p, j_q) инициируется процессом $output^i_p$ при наличии буферов в очереди $ВЫХ^i_p$, готовности драйвера к выводу (семафор "режим вывода" $RGOUT^i_p > 0$) и разрешении вывода (семафор $ALLOW^i_p > 0$):

$output^i_p$ process

M1: when $L = M1 \wedge ВЫХ^i_p > 0 \wedge RGOUT^i_p > 0 \wedge ALLOW^i_p > 0$ do

$ВЫХ^i_p \leftarrow ВЫХ^i_p - 0,1(\alpha)$; setout($\alpha$);

$RGOUT^i_p \leftarrow 0$; $L \leftarrow M2$ od

M2: when L = M2 $\wedge$ RGOUT^i_p < 0 do

memory^i ← memory^i + 1(α); Δ_i:=Δ_i + 1;

RGOUT^i_p ← 1; L ← M1 od

Процесс output^i_p посредством операции ВЫХ^i_p ← ВЫХ^i_p - 0,I (α) извлекает буфер α из очереди ВЫХ^i_p и передает этот буфер в подпрограмму постановки на вывод setout драйвера вывода СУ$_i$. Программа setout инициирует драйвер вывода. Семафор RGOUT^i_p устанавливается равным нулю и управление передается на проверку предиката второго действия процесса output^i_p. Второе действие инициируется при выполнении условия RGOUT^i_p<0, означающего окончание передачи содержимого буфера α по линии (i_p,j_q). Посредством операции memory^i ← ← memory^i+1(α) опустошенный буфер α помещается в список массивов свободной памяти memory^i. Число пустых буферов Δ_i увеличивается на I и семафор RGOUT^i_p устанавливается равным I, что означает готовность драйвера вывода к передаче по линии (i_p,j_q) следующего сообщения.

Передача буфера α по линии (i_p,j_q) состоит из трех этапов*): I) передачи заголовка сообщения; 2) передачи адресного пакета; 3) передачи информационного пакета.

Предварительная передача по линии (i_p,j_q) заголовка, содержащего суммарную длину адресного и информационного пакетов, имеет целью выделение на приемном конце линии (i_p,j_q) свободного буфера требуемой длины для приема адресного и информационного пакетов. На основе использования информации, содержащейся в адресном пакете, на приемном конце линии связи проверяется возможность возникновения дедлока [9] в случае завершения передачи сообщения по линии (i_p,j_q). Если возможность возникновения дедлока исключена, то по линии (i_p,j_q) передается информационный пакет. В противном случае информационный пакет не передается, а память, выделенная в ЭМ$_j$ для приема сообщения, утилизуется.

*) Информационным пакетом здесь называется собственно передаваемое сообщение, а адресным пакетом - дескриптор этого сообщения, который содержит, в частности, и информацию для определения маршрута передачи сообщения по сети связи.

```
process input_q^j
M1: when L = M1 ∧ HEAD_q^j > 0 do HEAD_q^j ← 0;
    memory^j ← memory^j - 1(γ);
    if γ = 0 then begin setsym (WACK); L ← M1 end
    else begin setsym (ACK1); Δ_j ← Δ_j - 1; L ← M2 end
    od
M2: when L = M2 ∧ ADDRESS_q^j > 0 do ADDRESS_q^j ← 0;
    if deadlock then begin setsym (RVI);
    memory^j ← memory^j + 1(γ); Δ_j ← Δ_j + 1;
    L ← M1 end else begin setsym (ACK1); L ← M3 end
    od
M3: when L = M3 ∧ INFORM_q^j > 0 do INFORM_q^j ← 0;
    BX_q^j ← BX_q^j + tail, 1(γ); L ← M1 od
```

Процесс $input_q^j$ на приемном конце линии связи (i_p, j_q) исполняет программу, состоящую из трех действий, каждое из которых соответствует одному из этапов передачи буфера по линии связи.

Первое действие исполняется по окончании приема заголовка, после того, как в ЭМ драйвер ввода q-го направления устанавливает семафор $HEAD_q^j = 1$. Действие обнуляет семафор $HEAD_q^j$ и посредством операции $memory^j \leftarrow memory^j - 1(\gamma)$ производит попытку выборки пустого буфера из списка массивов свободной памяти $memory^j$. Если в списке $memory^j$ нет буфера требуемой длины, то операция выборки завершается присвоением $\gamma := 0$. Если $\gamma = 0$, то подпрограмма setsym инициирует передачу по линии связи (j_q, i_p) кода WACK, запрещающего передачу по линии (i_p, j_q) адресного и информационного пакетов. Если же $\gamma \neq 0$ (буфер требуемой длины выделен), то по линии (j_q, i_p) передается код ACKI, сигнализирующий наличие выделенного пустого буфера на приемном конце линии (i_p, j_q). При этом число Δ_j пустых буферов в списке $memory^j$ уменьшается на единицу. Если буфер выделен, то по окончании первого действия управление передается на проверку предиката второго действия процесса $input_q^j$. В противном случае управление вновь передается на проверку предиката первого действия этого процесса.

Второе действие исполняется по окончании передачи по линии (i_p, j_q) адресного пакета, которая завершается установкой семафора $ADDRESS_q^j = 1$. Второе действие устанавливает $ADDRESS_q^j = 0$ и обращается к подпрограмме deadlock. Эта подпрограмма производит анализ состояния ресурсов $ЭМ_j$ с целью предотвращения дедлока, который может возникнуть, если передача сообщения по линии (i_p, j_q) будет завершена. Если дедлок возможен, то подпрограмма setsym инициирует передачу по линии (j_q, i_p) кода RVI, который запрещает дальнейшую передачу информации по линии (i_p, j_q). При этом выделенный ранее буфер γ возвращается в список $memory^j$, число свободных буферов Δ_j увеличивается на единицу и управление передается на проверку предиката первого действия процесса $input_q^j$. Если же дедлок невозможен, то по линии (j_q, i_p) передается код ACKI, разрешающий передачу по линии (i_p, j_q) информационного пакета. При этом управление передается на проверку предиката третьего действия процесса $input_q^j$.

Третье действие исполняется по окончании передачи по линии (i_p, j_q) информационного пакета, которая завершается установкой семафора $INFORM_q^j = 1$. Третье действие устанавливает $INFORM_q^j = 0$ и помещает заполненный принятым сообщением буфер γ в очередь полных буферов BX_q^j. Далее управление передается на проверку предиката первого действия процесса $input_q^j$ для приема следующего сообщения по линии (i_p, j_q).

Посредством установки семафора $ALLOW_p^i$ равным нулю можно запретить передачу сообщений по линии (i_p, j_q). Такая установка осуществляется драйвером ввода p-го направления $ЭМ_i$ при получении из $ЭМ_j$ по линии (j_q, i_p) управляющего кода EOT. Если в момент установки $ALLOW_p^i = 0$ передача сообщения по линии (i_p, j_q) начата, т.е. инициирован процесс $output_p^i$, то она доводится до конца. Дальнейшая передача сообщений по линии (i_p, j_q) остается запрещенной либо до истечения кванта задержки, установленного в таймере $ЭМ_i$ по получении кода EOT, либо до момента получения из $ЭМ_j$ кода ENQ. По получении кода ENQ драйвер ввода p-го направления $ЭМ_i$ устанавливает семафор ALLOW = 1, разрешая дальнейшую передачу сообщений по линии (i_p, i_q).

Запрет передачи сообщений по линии (i_p, i_q) приводит к изоляции машины $ЭМ_j$ от машины $ЭМ_i$. При этом $ЭМ_j$ переходит в монопрограммный режим и включается в виртуальную подсистему, предназначенную для эффективного выполнения параллельной программы. Переход

$ЭМ_j$ в монопрограммный режим ("замораживание") осуществляется путем передачи всех информационных векторов, готовых к исполнению и ждущих процессов из очередей ready и wait в очереди readyfreeze и waitfreeze соответственно. В процессор загружается информационный вектор процесса, исполняющего в $ЭМ_j$ одну из ветвей параллельной программы. "Размораживание", т.е. обратный переход $ЭМ_j$ в мультипрограммный режим, происходит либо по нормальному завершению исполнения параллельной программы, либо в случае аварийной ситуации (сбоя, переполнения и т.п.). В первом случае $ЭМ_j$ передает по линии (j_q, i_p) код ENQ. В аварийном случае машина $ЭМ_i$ возобновляет функционирование $ЭМ_j$ в мультипрограммном режиме по истечении кванта задержки, установленного в таймере $ЭМ_i$.

5. Организация поблочной передачи сообщений в ВС МИКРОС

В служебной виртуальной подсистеме передаваемое сообщение (например, текст программы) может иметь произвольную длину. Общеизвестным способом повышения надежности безошибочной передачи сообщения в сети ЭВМ является поблочная передача этого сообщения по сети связи. В ЭМ, являющейся абонентом-отправителем, производится разбиение сообщения на блоки заданной длины. Длина задана для подсистемы, в которой осуществляется передача. Образовавшиеся в результате разбиения блоки передаются по сети связи независимо. В ЭМ, являющейся абонентом-адресатом, эти блоки собираются вместе. Полученное в результате сборки сообщение поступает в обработку.

Передача каждого блока (информационного пакета) по межмашинной линии связи сопровождается передачей дескриптора этого блока (адресного пакета). Дескриптор блока (ДБ) содержит служебную информацию, необходимую для маршрутизации и идентификации блока. Дескриптор блока формируется на основе дескриптора передаваемого сообщения (см.рис.2). Дескриптор сообщения (ДС) содержит следующие составляющие: ДС[признаки]; ДС[число блоков]; ДС[номер блока]; ДС[длина сообщения]; ДС[номер машины]; ДС[адрес]. Слово признаков характеризует тип передаваемого сообщения (команда или данные) и тип путевой процедуры, используемой для передачи сообщения. ДС[число блоков] содержит число блоков, на которые разбивается сообщение. ДС[номер машины] задает номер машины, являющейся абонентом-отправителем, ДС[адрес] задает адрес машины, являющейся абонентом-адресатом, и используется для маршрутизации блока. ДС[номер блока] задает текущий номер блока, образуемого в процессе раз-

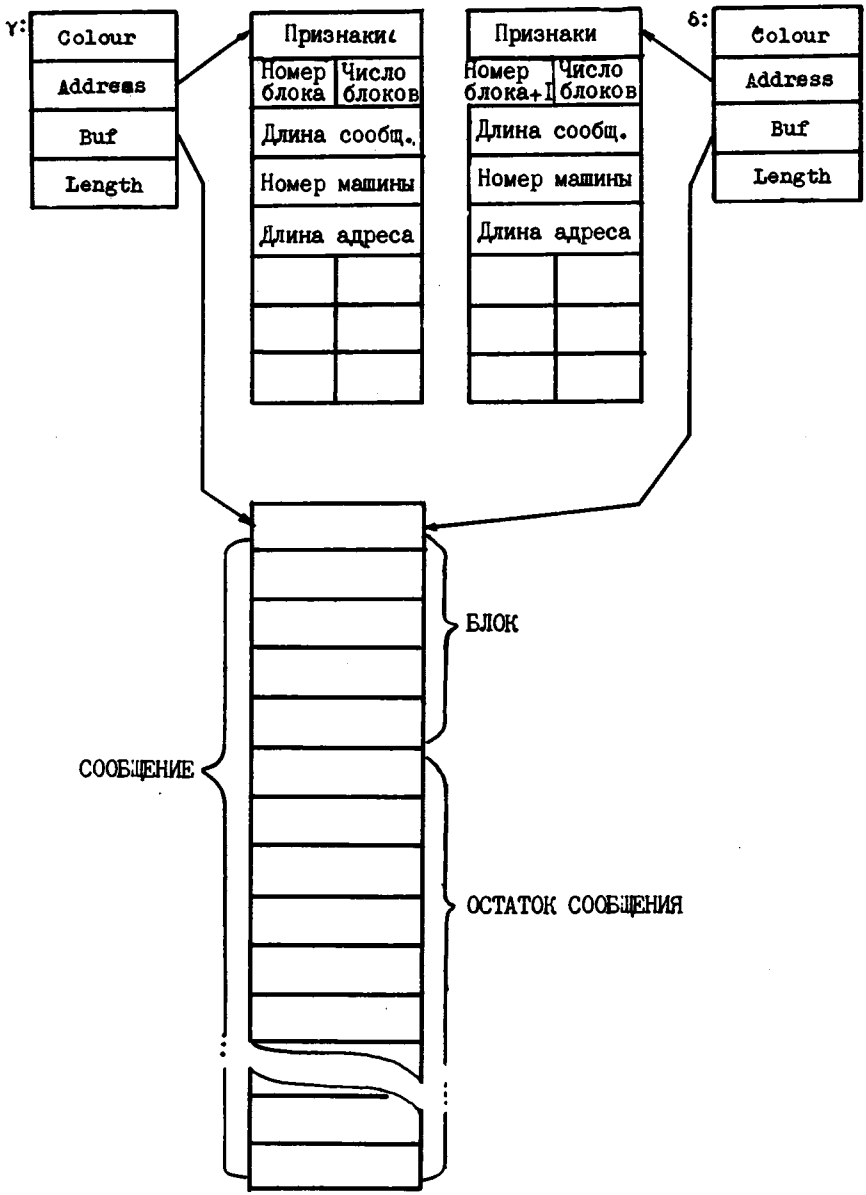


Рис. 2. Выделение блока в сообщении.

биения сообщения. Разбиение сообщения на блоки осуществляется процедурой insertoutput, которая интерпретирует операцию $ВЫХ_p \leftarrow ВЫХ_p +$ + tail,1(γ) занесения сообщения γ в конец очереди $ВЫХ_p$ p-го выходного полюса абонента-отправителя, $p \in \{1,2,\ldots,v\}$, v - число полюсов абонента-отправителя. Сообщение задается таблицей адресов и длин [γ], где buf(γ) - адрес сообщения, а address(γ) - адрес дескриптора сообщения. Компоненты дескриптора сообщения ДС [признаки] и ДС[адрес] определяются программой, породившей сообщение. Остальные компоненты формируются процедурой insertoutput.

В сети связи ВС постоянно возникают и вновь исчезают блокировки направлений кратчайших путей. Причинами блокировок могут быть неисправности в линиях связи, загруженность линий связи, установление монопрограммного режима в физической подсистеме (порождение виртуальной подсистемы для исполнения параллельной программы) и так далее. В связи с этим различные блоки одного и того же сообщения движутся по различным маршрутам и в результате поступают в машину-адресат в произвольном порядке и с различных направлений. Сборку сообщения из поступающих блоков осуществляет процедура insertinput, интерпретирующая операцию $ВХ_0 \leftarrow ВХ_0$+tail,1(γ) занесения буфера γ в очередь сообщений $ВХ_0$, обрабатываемых абонентом-получателем. При сборке используется список "компонуемых" сообщений linklist. Каждый элемент этого списка является массивом, длина которого на два слова больше длины сообщения. Этот массив извлекается из списка массивов свободной памяти memory при поступлении первого (по порядку поступления) блока сообщений. Из дескриптора принятого блока считываются: номер передающей машины, число блоков сообщения, номер данного блока. Номер передающей машины записывается в первое слово выделенного массива, число блоков без единицы - во второе. По номеру блока определяется адрес, начиная с которого содержимое принятого блока переписывается в выделенный массив. Если [β] - таблица адресов и длин, указывающая на элемент списка linklist, то address(β) указывает на первое слово этого элемента, а buf(β) - на третье слово. Таким образом, buf(β) является адресом "компонуемого" сообщения и адрес, начиная с которого записывается содержимое принятого блока, равен buf(β)+2L_b·(ДБ[номер блока]-I), L_b - длина блока.

После принятия и копирования каждого блока параметр ЧИСЛО БЛОКОВ в элементе списка linklist уменьшается на единицу. Когда этот параметр станет равным нулю, сформированное сообщение поме-

щается в конец очереди $ВХ_0$. Все принятые блоки по мере их копирования возвращаются в список свободных массивов memory.

Поблочная передача очередного сообщения из некоторой $ЭМ_i$ в $ЭМ_j$, $j \neq i$, возможна лишь после окончания поблочной передачи из $ЭМ_i$ в $ЭМ_j$ предыдущего сообщения. В противном случае может произойти "перемешивание" блоков этих сообщений, поскольку их дескрипторы одинаковы. В результате такого перемешивания в $ЭМ_j$ в элемент списка linklist, содержащий номер i передающей машины $ЭМ_i$, могут быть скопированы блоки различных сообщений. Во избежание такой ситуации необходимо из $ЭМ_j$ в $ЭМ_i$ передавать извещение об окончании поблочной передачи сообщения из $ЭМ_i$ в $ЭМ_j$. Только после принятия извещения в $ЭМ_i$ начинается поблочная передача следующего сообщения из $ЭМ_i$ в $ЭМ_j$.

При исполнении параллельной программы каждое передаваемое сообщение имеет фиксированный размер и содержит, как правило, одно или несколько данных. В связи с этим отпадает необходимость разбиения сообщения на блоки. Для организации передачи сообщения по линии связи пользователь, создавший программу, должен предусмотреть очереди пустых буферов фиксированной длины на приемных концах линий связи, а для маршрутизации сообщений в подсистеме разработать специализированные путевые процедуры. Примеры таких процедур приведены в работе [13]. При использовании специализированных процедур упрощается алгоритм функционирования межмашинной линии связи, поскольку исчезает необходимость в процедуре предотвращения дедлоковых ситуаций. В этом случае адресный пакет (дескриптор сообщения) и информационный пакет (текст сообщения) образуют единый массив, передача которого по линии связи осуществляется за один этап.

Заключение

Рассмотренные в данной работе программные средства в совокупности с ядром операционной системы ЭМ, драйверами системного устройства и путевыми процедурами позволяют организовать обмен сообщениями между произвольными ЭМ подсистемы ВС МИКРОС. Обмен основан на использовании сети связи с коммуникацией пакетов, что позволяет обеспечить надежную передачу сообщений. В работе выделены особенности организации межмашинных передач сообщений в подсистемах, предназначенных для исполнения параллельных программ. Для

эффективного исполнения таких программ машины выделенной подсистемы должны функционировать в монопрограммном режиме. Описанные алгоритмы реализованы на базе языка программирования MACRO-11 в составе математического обеспечения ВС МИКРОС. Объем каждой из рассмотренных процедур в среднем составляет около 100 команд языка MACRO-II.

Литература

I. ЕВРЕИНОВ Э.В., ХОРОШЕВСКИЙ В.Г. Однородные вычислительные системы. -Новосибирск: Наука, 1978. - 318 с.

2. КОРНЕЕВ В.В., ХОРОШЕВСКИЙ В.Г. Вычислительные системы с программируемой структурой. -Электронное моделирование, 1979, №1, с. 42-52.

3. ЭВМ пятого поколения. Концепции, проблемы, перспективы /Под редакцией Мото-ока Т. -М.: Финансы и статистика, 1984. - 110 с.

4. СОФРОНОВ И.Д. Оценка параметров вычислительной машины, предназначенной для решения задач механики сплошной среды. -В кн.: Численные методы механики сплошной среды. Том 6. Новосибирск, 1975, № 3, с. 98-147.

5. КОНОВАЛОВ А.Н., ЯНЕНКО Н.Н. Некоторые вопросы теории модульного анализа и параллельного программирования для задач математической физики и механики сплошной среды. -В кн.: Современные проблемы математической физики и вычислительной математики. - М.: 1982, с. 200-207.

6. ХОРОШЕВСКИЙ В.Г. Вычислительная система МИКРОС. - Новосибирск, 1983. - 52 с. (Препринт/ИМ СО АН СССР: ОВС-19).

7. ДИМИТРИЕВ Ю.К., ЗАДОРОЖНЫЙ А.Ф., КОРНЕЕВ В.В. Элементарная машина вычислительной системы с программируемой структурой МИКРОС. -В кн.: Вычислительные системы с программируемой структурой (Вычислительные системы, вып. 94). Новосибирск, 1982, с.3-15.

8. ВЕЙЦМАН К. Распределенные системы мини- и микро-ЭВМ. -М.: Финансы и статистика, 1983. - 302 с.

9. Wai Sum Lai. Protocol traps in Computer Networks - a Catalog.-IEEE Trans.on Comm.,1982,v.COM-30,№ 6,p.1434-1449.

10. КОРНЕЕВ В.В. Элементарная машина однородной вычислительной системы с программируемой структурой. -Кибернетика, 1980, №1, с. 75-81.

11. КОРНЕЕВ В.В., МОНАХОВ О.Г., ТАРКОВ М.С. Ядро операционной системы с программируемой структурой. -В кн.: Однородные вычислительные системы (Вычислительные системы, вып. 90). Новосибирск, 1981, с. 22-42.

12. ЯКУБАЙТИС Э.А. Архитектура вычислительных сетей. -М.: Статистика, 1980. - 279 с.

13. КОРНЕЕВ В.В. Способ организации межмашинных обменов в вычислительных системах с программируемой структурой. - Электронное моделирование, 1982, № 6, с. 17-26.

Поступила в ред.-изд.отд.
2 октября 1984 года