ISSN 0002-306X. Изв. НАН РА и ГИУА. Сер. ТН. 2014. Т. LXVII, № 3.

УДК 681.3

ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ИНФОРМАТИКА

А.Х. ПАЛЯН, А.А. ПАЛЯН

МЕТОД ВЗАИМОБЛОКИРОВКИ ПРИКЛАДНЫХ ПРОГРАММ В РАСПРЕДЕЛЕННЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМАХ И ЕГО ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

Представлен метод взаимоблокировки прикладных программ в локальных вычислительных сетях, реализация которого облегчает разработку распределенных приложений. Описана реализация метода в виде программного пакета. Отличительной особенностью серверной компоненты пакета является ее функционирование в многопоточном режиме, что обеспечивает параллельный доступ к сетевым мьютексам со стороны прикладных программ клиентов.

Ключевые слова: сетевой мьютекс, механизм сокетов, многопоточная обработка, реентерабельная программа, интерфейсная функция.

В современных локальных вычислительных сетях прикладные программы, функционирующие на различных узлах сети, могут использовать общие сетевые ресурсы (файл, принтер и т.д.). В зависимости от режима доступа к ресурсам (запись, чтение) может возникнуть необходимость захвата ресурса прикладными программами для монопольного использования. Подобная проблема, возникающая в режиме мультипрограммирования на одном узле сети, решается путем использования известных механизмов современных операционных систем (ОС). В одной из наиболее популярных ОС Windows 7 основным является механизм “мьютексов” (mutex – mutual exclusion). Также используются разновидности этого механизма под названием “критическая секция” и “семафор” [1]. Однако упомянутые мьютексы являются локальными в данном узле и не могут быть использованы для взаимоблокировки доступа к сетевым ресурсам.

Исходя из вышесказанного, возникает необходимость разработки специальных механизмов взаимоблокировки на сетевом уровне, поддерживаемых сетевыми ОС. В настоящее время широко распространены сетевые ОС фирмы Microsoft. К их числу принадлежат ОС Windows 7 для клиентских узлов сети и Windows Server для серверных узлов. Подробное рассмотрение этих ОС показывает, что в них отсутствуют какие-либо системные средства взаимоблокировки на сетевом уровне, предоставляемые прикладным программам.

В литературе [2] представлен ряд методов взаимоблокировки доступа к сетевым ресурсам в распределенных системах. Они делятся на две группы:

- централизованные методы (программа-координатор);

- распределенные методы (циркуляция маркера доступа).

Методы обеих групп реализуются на прикладном уровне, что усложняет программирование приложений и приводит к нестандартным решениям в разных группах взаимодействующих прикладных программ. Несмотря на указанные недостатки, реализация этих методов является, по-видимому, единственным решением проблемы взаимоблокировки на сетевом уровне при разработке распределенных прикладных программных систем на платформе сетевых ОС фирмы Microsoft.

Однако более перспективным представляется альтернативный **метод централизованной реализации сетевой взаимоблокировки на системном уровне в виде компоненты ОС**. В этом случае прикладные программы освобождаются от необходимости реализации сложных методов взаимоблокировки. Кроме того, такое решение повысит надежность программной системы в целом, особенно при использовании отказоустойчивых серверов.

Настоящая статья посвящена изложению данного метода и его реализации в виде программного пакета.

Центральной компонентой пакета является программа, функционирующая в среде Windows Server в статусе системного процесса управления сетевыми мьютексами (ПУСМ). Запросы на обладание сетевыми мьютексами от прикладных программ клиентских узлов обрабатываются интерфейсными функциями, взаимодействующими с ПУСМ посредством **механизма сокетов** [2].

В терминах этого механизма **ПУСМ является многоканальным серверным процессом, функционирующим в многопоточном режиме.** Основной поток процесса (main thread) прослушивает сеть и при появлении запроса с клиентского узла устанавливает с ним логическое соединение. Для непосредственного взаимодействия с клиентом по данному соединению создается отдельный поток (thread), выполняемый функцией потока. Такая организация ПУСМ позволяет параллельно обслуживать запросы с нескольких клиентских узлов.

Сетевые мьютексы реализуются в рамках ПУСМ в виде глобального массива 32 целых переменных. Мьютекс с номером i – это элемент массива m[i] со следующими значениями:

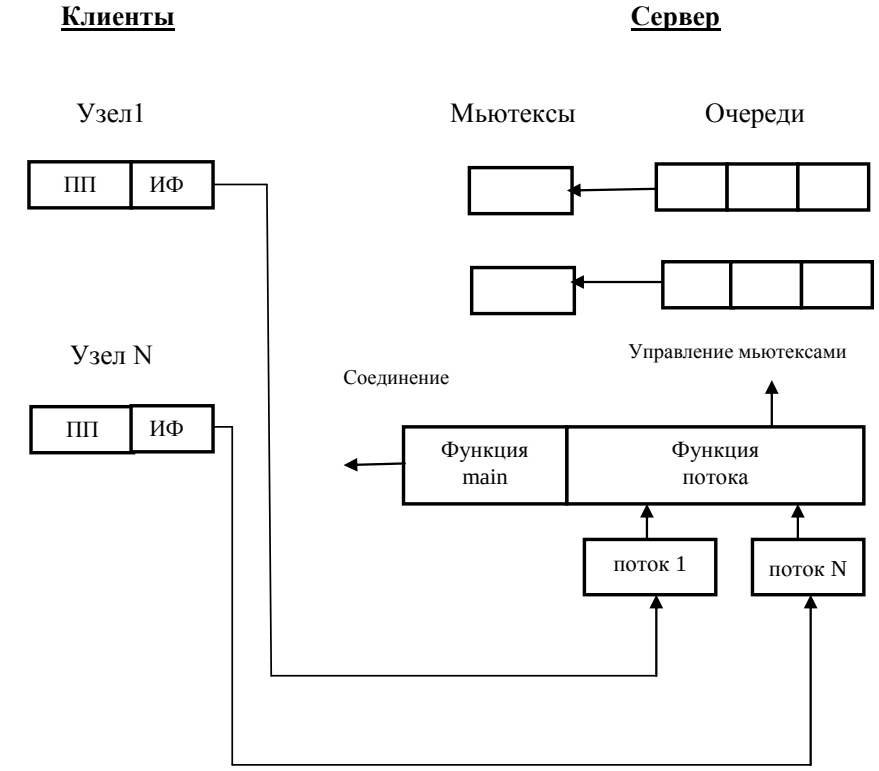
0 – мьютекс свободен;

1 – мьютекс занят;

>1 – мьютекс занят, имеется очередь к мьютексу.

С каждым мьютексом связана **очередь запросов** на обладание им в виде глобального массива структур (в терминах языка C++). Очередь функционирует в режиме “первый пришел – первым обслужен” по циклическому принципу. Элементом очереди является заявка от прикладной программы клиентского узла на обладание сетевым мьютексом.

Работа программного пакета схематически представлена на рисунке.



ПП – прикладная программа
ИФ – интерфейсная функция

Рис. Схема взаимодействия программ в сети

Каждый поток осуществляет один и тот же алгоритм взаимодействия с клиентом, поэтому все потоки могут выполняться одной программой – функцией потока. Такая программа должна обладать свойством **реентерабельности**. Это свойство программы означает, что она может быть прервана в рамках одного потока и запущена с начальной точки в рамках другого потока. Для этого программа должна хранить локальные переменные в стеке того потока, в рамках которого она работает, что осуществляется автоматически и поэтому не вызывает проблем.

Сложнее дело обстоит с глобальными переменными - сетевыми мьютексами и очередями запросов. При одновременном обращении двух и более клиентов для захвата одного сетевого мьютекса m[i] и, как следствие, параллельной работе программы в рамках разных потоков с одним сетевым мьютексом m[i] и соответствующей очередью может возникнуть логическая ошибка. Для исключения этой ситуации программа должна работать с указанными данными строго последовательно. С этой целью с каждым сетевым мьютексом m[i] и соответствующей

очередью связывается **локальный мьютекс Lm[i]**, поддерживаемый ОС Windows [1]. В ПУСМ создается 32 таких локальных мьютекса. Программа может работать с сетевым мьютексом m[i] и соответствующей очередью только после предварительного захвата Lm[i]. При занятости Lm[i] соответствующий поток перейдет в состояние ожидания. Работа с разными сетевыми мьютексами m[i] и соответствующими очередями может осуществляться программой параллельно в рамках разных потоков, поскольку пересечение по данным отсутствует.

В пакете имеются две интерфейсные функции. Захват сетевого мьютекса производится **функцией GetNetworkMutex(p1, p2, p3)**, а освобождение – функцией **ReleaseNetworkMutex(p1)**. Обе функции оформляются вместе в виде динамически связываемой библиотеки, что позволяет использовать их копии одновременно различными прикладными программами клиентского узла.

Параметры функции GetNetworkMutex:

p1 - номер запрашиваемого сетевого мьютекса. Это целое число в пределах 1-32;

p2 - предельное время ожидания предоставления мьютекса (в миллисекундах) либо указание о неограниченном ожидании;

p3 – номер порта, выделенного данной прикладной программе.

Механизм сокетов предполагает идентификацию каждой программы, взаимодействующей в сети, выделенным ей **номером порта**. Узел сети идентифицируется **сетевым адресом IP**.

Параметр функции ReleaseNetworkMutex:

p1 - номер освобождаемого сетевого мьютекса.

Рассмотрим процессы захвата и освобождения сетевых мьютексов. Для захвата сетевого мьютекса прикладная программа вызывает функцию *GetNetworkMutex(),* которая устанавливает логическое соединение с ПУСМ посредством механизма сокетов. ПУСМ создает отдельный поток для взаимодействия с функцией, которая передает потоку параметры запроса, а именно: номер запрашиваемого сетевого мьютекса и идентификационные данные о прикладной программе (IP, номер порта).

Далее поток определяет состояние мьютекса, и если он свободен, посылает положительный ответ функции *GetNetworkMutex(),* которая информирует прикладную программу о захвате мьютекса. Если мьютекс занят, то запрос устанавливается в соответствующую очередь, а функции посылается отрицательный ответ. Функция переводит себя и прикладную программу в состояние ожидания предоставления ей мьютекса. Логическое соединение с ПУСМ разрывается.

Для освобождения сетевого мьютекса прикладная программа вызывает функцию *ReleaseNetworkMutex(),* которая устанавливает логическое соединение с ПУСМ аналогичным способом. ПУСМ создает поток для взаимодействия с функцией, которая передает потоку номер освобождаемого сетевого мьютекса.

Поток выдает функции подтверждение об освобождении мьютекса, после чего логическое соединение функции с ПУСМ разрывается. Далее в случае наличия очереди к данному мьютексу поток выбирает первую заявку из очереди, в которой имеются идентификационные данные ожидающих прикладной программы и функции *GetNetworkMutex()*. Поток устанавливает логическое соединение с функцией и сообщает ей о выделении мьютекса. Функция информирует прикладную программу о захвате мьютекса. Логическое соединение разрывается.

Представленный программный пакет может использоваться разработчиками распределенных приложений в локальных вычислительных сетях под управлением сетевых ОС фирмы Microsoft.

СПИСОК ЛИТЕРАТУРЫ

1. **Рихтер Дж.** Windows для профессионалов. – СПб.: Питер, 2001.- 752 с.
2. **Танненбаум Э.** Распределенные системы. Принципы и парадигмы.- СПб.: Питер, 2003.- 877 с.

ГИУА (ПОЛИТЕХНИК). Материал поступил в редакцию 10.02.2014.

Ա.Խ. ՊԱԼՅԱՆ, Ա.Ա. ՊԱԼՅԱՆ

ԲԱՇԽՎԱԾ ՀԱՇՎՈՂԱԿԱՆ ՀԱՄԱԿԱՐԳԵՐՈՒՄ ԿԻՐԱՌԱԿԱՆ ԾՐԱԳՐԵՐԻ ՓՈԽԱՐԳԵԼԱՓԱԿՄԱՆ ՄԵԹՈԴԸ ԵՎ ՆՐԱ ԾՐԱԳՐԱՅԻՆ ԻՐԱԿԱՆԱՑՈՒՄԸ

Լոկալ հաշվողական ցանցերում կիրառական ծրագրերի փոխարգելափակման մեթոդի իրականացումը հեշտացնում է բաշխված կիրառությունների մշակումը: Ներկայացվում է այդ մեթոդն իրականացնող ծրագրային փաթեթը: Փաթեթի սերվերային մասի առանձնահատկությունն այն է, որ աշխատում է բազմահոսքային ռեժիմում, ինչը ապահովում է զուգահեռ հասանելիություն կլիենտների կիրառական ծրագրերի ցանցային մյութեքսներին:

Առանցքային բառեր. ցանցային մյութեքս, սոկետների մեխանիզմ, բազմահոսքային մշակում, կրկնակի գործարկվող ծրագիր, ինտերֆեյսային ֆունկցիա:

A.KH. PALYAN, A.A. PALYAN

A METHOD FOR THE APPLIED PROGRAM INTERLOCKING IN DISTRIBUTED COMPUTING SYSTEMS AND ITS SOFTWARE IMPLEMENTATION

A method of the applied program interlocking in the local area networks facilitating the design of distributed applications is presented. The software implementation of this method is described. The distinctive feature of the server part of the software is its operation in multithreaded mode that provides parallel access from the client applied program to network mutexes.

Keywords: network mutex, socket mechanism, multithreaded processing, reenterable program, interface function.