Сетевая Рабочая Группа
RFC: 3018
Категория: Экспериментальный

А. Богданов
НКО "ОРС"

Декабрь 2000

Спецификация Протокола Объединенного Пространства Памяти.

Статус этого документа

Данный документ определяет Экспериментальный Протокол для сообщества Internet. Он не специфицирует стандарт Internet любого вида. Запрашивается обсуждение и предложения для усовершенствования. Распространение данного документа неограниченно.

Авторские права

Резюме

Документ содержит спецификацию протокола объединенного пространства памяти(Unified Memory Space Protocol - UMSP), который предоставляет возможность прямого доступа к памяти удаленных узлов.

Соглашения, принятые в документе

Последующая спецификация синтаксиса использует расширенную Форму Бакуса-Наура (ABNF), как описано в RFC-2234 [3].

Оглавление

1 Введение

UMSP является сетевым протоколом с установлением соединения. Он соответствует сеансовому и представительному уровням модели OSI. Протокол предназначен для реализации в широком классе систем, от простых устройств на базе специализированных процессоров, до универсальных компьютеров и кластеров.

Для обмена данными протокол использует сервис транспортного уровня с гарантированной доставкой. Для передачи данных, не требующих подтверждения, возможно использование протокола, не обеспечивающего гарантированную доставку. Настоящий документ описывает использование TCP и UDP.

Основное назначение протокола UMSP – создание сетевой среды для организации 128-разрядного адресного пространства памяти, распределенного между узлами Internet. Протокол регламентирует механизм установления соединения, и формат передаваемых инструкций. Он не осуществляет непосредственное управление локальной памятью на узле.

В отличие от традиционных сетевых протоколов, приложения пользователя на разных узлах осуществляют взаимодействие не путем обмена сетевыми примитивами или работой с потоками данных, а непосредственным чтением/записью данных или передачей управления на код в виртуальной памяти удаленного узла. Приложение пользователя может ничего не знать о существовании протокола и сети, а просто пользоваться инструкциями с 128-разрядными адресами.

UMSP в первую очередь предполагается использовать в системах на базе виртуальных машин (VM), исполняющих псевдокод. Однако его можно применять в системах, непосредственно исполняющих код процессора. Например, в кластерах или в универсальных операционных системах, для организации распределенного виртуального адресного пространства. Кроме этого минимальный профиль протокола можно использовать в простых устройствах, не имеющих операционной системы.

Протокол предоставляет гибкие средства для установки параметров соединений и позволяет строить системы с высоким уровнем защиты без ущерба для функциональных возможностей приложений.

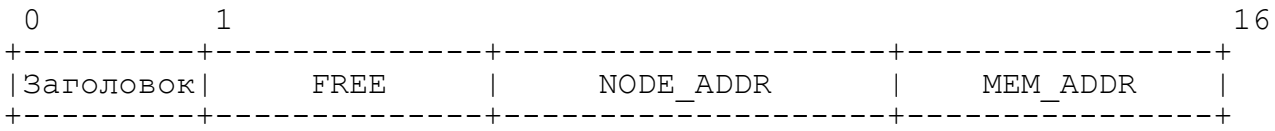
UMSP может существенно упростить процесс разработки распределенных систем. Он предоставляет возможность объединять не только информационные, но и вычислительные ресурсы большого числа разнотипных компьютеров без значительных затрат на стандартизацию и разработку ПО.

2 Модель UMSP

2.1 128-разрядное адресное пространство

UMSP опирается на модель 128-разрядного распределенного адресного пространства памяти. 128-разрядный адрес содержит информацию о типе сети, сетевом адресе узла и адресе локальной памяти. Он имеет следующий формат:

октеты

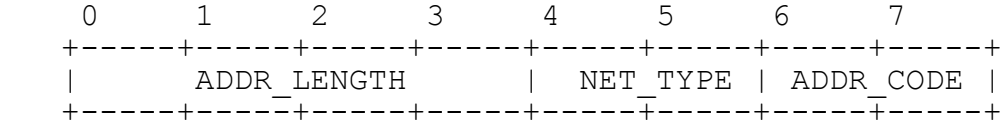


Полная длина адреса фиксирована и составляет 16 октетов.

Заголовок

1 октет. Заголовок адреса. Это поле полностью определяет формат адреса. Заголовок имеет следующий формат:

биты:



ADDR_LENGTH

4 бита. Длина сетевого адреса узла. Это поле содержит число октет в поле NODE_ADDR адреса. Значение 0 не допускается.

NET_TYPE

2 бита. Тип сети. Это поле определяет тип сети, в которой находится узел. Для определения типа сети значение этого поля используется в комбинации с остальными полями заголовка (одно значение NET_TYPE может определять разные сети, если значения полей ADDR_LENGTH и/или ADDR_CODE различны).

ADDR_CODE

2 бита. Код длины адреса локальной памяти. Значение этого поля задает длину адреса локальной памяти. Определены следующие значения кода и соответствующая им длина поля MEM_ADDR адреса:

%b00 - 16 бит
%b01 - 24 бит
%b10 - 32 бит
%b11 - 64 бит

Комбинация значений трех полей заголовка называется номером формата адреса и однозначно определяет сеть, в которой расположен узел. Номер формата записывается следующим образом:

№ <ADDR_LENGTH> - <NET_TYPE> - <ADDR_CODE>

Например, № 4-0-2 определяет адрес с длиной сетевого адреса узла 4 октета и адресом памяти длиной 32 бита. Тип сети 0 для такого формата адреса в настоящем документе определен для сети IPv4. Если тип сети равен нулю, при записи формата адреса он может быть пропущен. Например, формат № 4-0-2 и 4-2 совпадают. Если оба поля NET_TYPE и ADDR_CODE равны нулю, они могут быть опущены. При этом номер формата записывается, как одна цифра.

Каждой глобальной сети, включенной в объединенную систему, должен быть выделен один или несколько номеров формата адреса.

FREE

0 - 12 октет. Это поле не используется протоколом. Оно может содержать любую дополнительную информацию, необходимую системе управления памяти на узле, к которому относится 128 - битный адрес. Если это поле не используется, во все октеты должно быть записано нулевое значение. Использование этого поля приводит к тому, что в сетевых инструкциях должен содержаться только полный 16 - октетный адрес и не может использоваться короткий адрес локальной памяти.

NODE_ADDR

1 - 13 октет. Адрес узла. Формат этого поля определяется отдельно для каждого номера формата адреса. Поле адреса узла не обязательно должно точно соответствовать реальному сетевому адресу. Если реальный сетевой адрес длиннее этого поля, в сети необходимо выделить подмножество адресов, поддерживающих протокол UMSP.

MEM_ADDR

16/24/32/64 бит. Адрес локальной памяти. Это поле представляют собой адрес памяти в системе, которая задается полем NODE_ADDR. Узел полностью отвечает за управление своей памятью. Протокол не определяет порядок использования и формат этого поля.

Для приложений пользователя 128-битный адрес представляет собой единое поле. Код пользователя не обязательно должен знать о физическом расположении адресуемой памяти.

128-разрядный адрес памяти может передаваться между узлами, как данные. Например, в буфере параметров функции, или в инструкции копирования данных. Поэтому он должен однозначно идентифицировать заданный узел с любого другого узла.

Не существует какого либо определенного алгоритма, связывающего реальный сетевой и 128-разрядный адрес. Все используемые форматы адреса должны быть заранее известны.

Так как UMSP имеет свою систему адресации, он может объединять несколько глобальных сетей, а узлы могут иметь внутренние локальные сети или подчиненные адресуемые устройства, связанные с узлом несетевыми коммуникациями. Любой узел по номеру формата адреса должен иметь возможность определить шлюз, отвечающий за маршрутизацию этого адреса.

2.2 Вычислительная модель

Вычислительная модель является трехуровневой:

(1) Задание
(2) Задача
(3) Поток управления

Задание соответствует выполняемому приложению пользователя. Задание является распределенным и может одновременно выполняться на многих узлах. Управление заданием осуществляется централизованно с узла, называемого пунктом управления задания (Job Control Point - JCP). Один JCP может управлять несколькими заданиями. JCP может быть расположен на том же узле, на котором создается задание, или на любом другом адресуемом узле сети.

Задача является представлением задания на отдельном узле. Задача включает один или несколько вычислительных потоков управления. Задание имеет только одну задачу на каждом узле.

Задание завершается, когда завершается соответствующее ему приложение пользователя. При завершении задания завершаются все задачи этого задания на всех узлах.

Задание имеет свое изолированное 128-разрядное адресное пространство. Адресное пространство является сегментированным. Сегментом является локальная память одного узла. Кроме этого протокол позволяет работать с объектами. Объекты являются отдельной ассоциативной памятью узла.

Поток задачи представляет собой конкретный поток управления, выполняемый VM в определенном узле. Поток может писать и читать по любому адресу 128-разрядного адресного пространства задания. Передача управления на адрес из другого (удаленного) узла приводит к созданию нового потока на удаленном узле. Непрерывный сегмент кода не может быть распределен по нескольким узлам. Также нельзя получить непрерывную область памяти, распределенную по нескольким узлам.

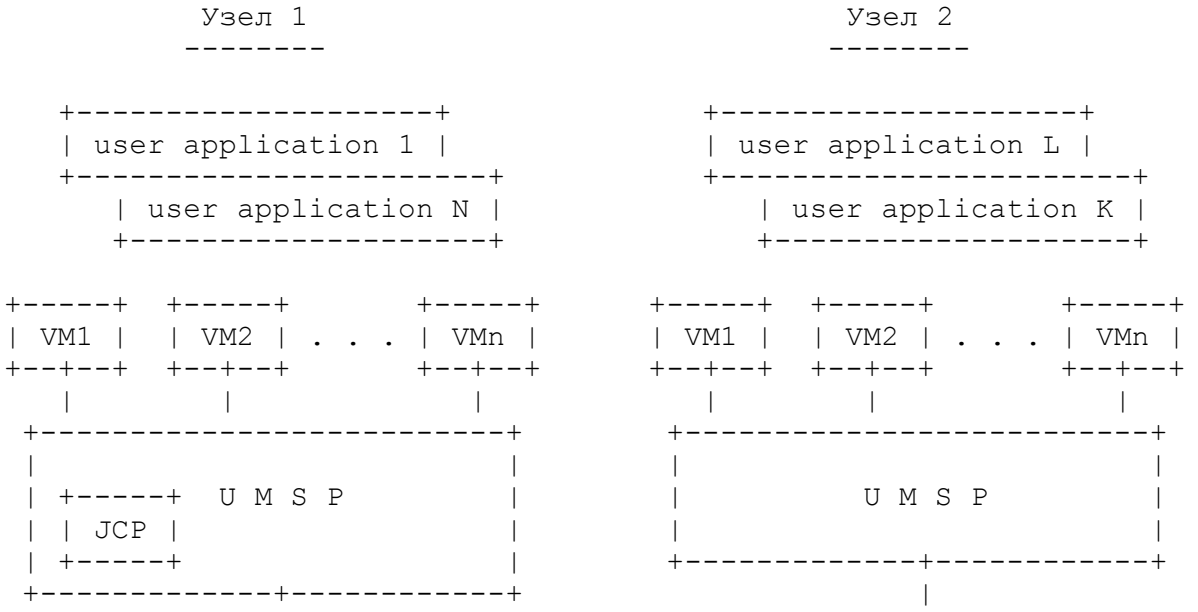
Протокол не требует от VM отдельного узла поддерживать для разных задач непересекающееся пространство памяти. Поддержка многопоточности также не является обязательным требованием.

Имеется глобальный 128-разрядный идентификатор задания (Global Job Identifier - GJID). Он устанавливается на JCP, который управляет заданием. В объединенной системе в каждый момент времени все активные GJID имеют уникальное значения.

Задание может содержать код VM разных типов. VM разных типов могут находится на разных узлах, или они могут быть подключены к протоколу на одном узле. Для того чтобы неоднородный код мог выполняться на одном узле в контексте одного задания, предусмотрен механизм объединения VM разного типа на одном узле в группы. Группы подробно описаны в секции 9. VM, объединенные в группы, должны работать в общем пространстве памяти (иметь общую подсистему управления памятью).

2.3 Архитектура системы

Общая структура системы на базе виртуальных машин и место протокола UMSP в ней приведена на следующем рисунке:



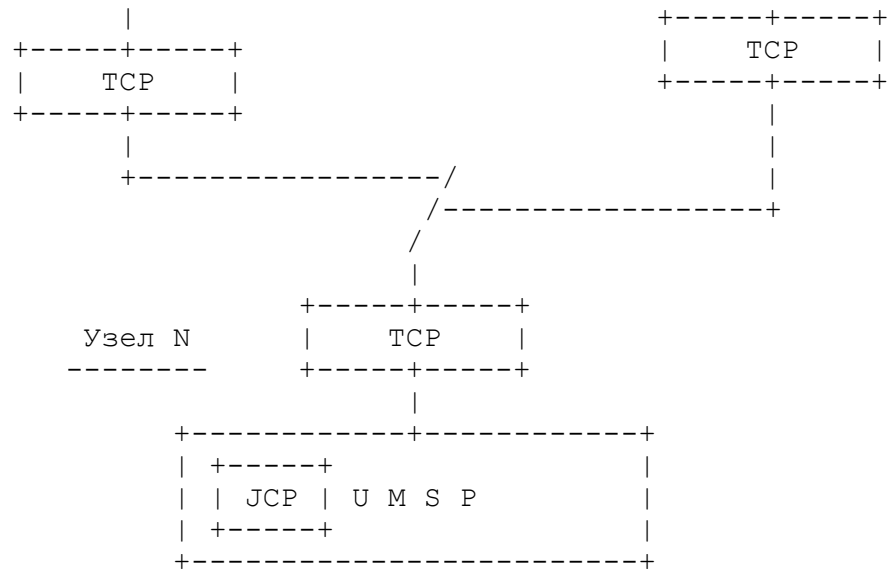


Рис. 1. Структура системы на базе VM.

На верхнем для UMSP уровне работает одна или несколько VM. Уровень VM не является сетевым уровнем. Последним сетевым уровнем является UMSP. Поэтому уровень VM не имеет собственных сетевых примитивов и использует с UMSP одно поле кода операции.

Конечным потребителем услуг протокола является код пользователя, выполняющейся на виртуальной машине. Он имеет инструкции с 128-битным адресом. VM преобразует эти инструкции в сетевые команды, которые через протокол UMSP передаются для выполнения на удаленной машине. Внутренняя организация VM, система команд и API могут быть любыми. Протокол определяет только формат примитивов, которыми виртуальные машины обмениваются через сеть.

Протокол не управляет памятью заданий. Управление памятью должны выполнять VM. Если на одном узле работает несколько VM. Они могут иметь общее пространство памяти или быть полностью изолированными.

Для обмена данными UMSP использует транспортный уровень с гарантированной доставкой. Настоящий документ определяет использование TCP. Для передачи данных, не требующих подтверждения, может использоваться UDP. При этом наличие соединения через TCP является обязательным. Допускается использование множественных соединений TCP с мультиплексированием. Управление транспортными соединениями не является частью протокола UMSP.

Инструкции UMSP не содержат сетевых адресов получателя и отправителя. Протокол требует, чтобы одному адресу транспортного уровня соответствовал один адрес UMSP. Соответственно по 128-разрядному адресу памяти необходимо однозначно определять адрес узла на транспортном уровне.

Кроме TCP возможно применение других транспортных протоколов или несетевых коммуникаций. К ним предъявляются следующие требования:

- гарантированная доставка. Транспортный уровень должен сообщать о доставке или об ее невозможности;
- допускается нарушение последовательности передаваемых сегментов;
- дублирование сегментов не допускается;

- при аварийной перезагрузке узлов необходимо гарантировать идентификацию сегментов, относящихся к сеансовым соединениям, установленным до перезагрузки;
- установление соединения не является обязательным.

VM является самостоятельной программой и взаимодействие с протоколом ей необходимо только при выполнении инструкций с 128-разрядным адресом, относящимся к другому узлу. VM может выполнять несколько пользовательских задач. Каждая задача может содержать несколько потоков управления. VM должна уметь интерпретировать инструкции приложения с 128-разрядным адресом в одну или несколько инструкций протокола UMSP.

Для обмена данными между узлами устанавливается сеансовое соединение. С одним соединением связано только одно задание. Одновременно между двумя узлами может существовать несколько сеансовых соединений для разных заданий. Протокол также предусматривает обмен данными без установления соединения.

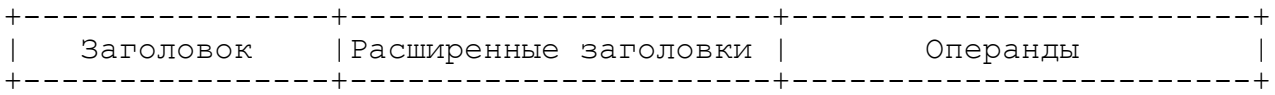
Обмен между узлами UMSP может включать инструкции следующего типа:
- прямое чтение/запись в память
- запросы выделения/освобождения памяти
- инструкции сравнения
- инструкции вызова подпрограммы и безусловного перехода
- инструкции синхронизации
- инструкции работы с объектами – чтение/запись в память объекта по смещению и вызов методов объекта по номеру.

UMSP не отслеживает пользовательские потоки управления. Поэтому VM должна сама обеспечивать необходимый порядок выполнения инструкций.

Длина инструкций UMSP не зависит от длины сегмента транспортного уровня. Для передачи длинных инструкций предусмотрена сегментация, а для передачи коротких инструкций используется их упаковка в один сегмент с возможностью сжатия заголовков. Минимальный размер сегмента, необходимый для работы 6 октет. Для реализации всех функций требуется 56 октет.

3 Формат инструкций

Инструкции UMSP состоят из основного заголовка, расширенных заголовков и операндов. Все поля имеют переменную длину.



Заголовок содержит код операции и несет информацию, необходимую для интерпретации инструкции.

Расширенные заголовки содержат дополнительную информацию, не определенную в основном заголовке.

Операнды содержат данные инструкции.

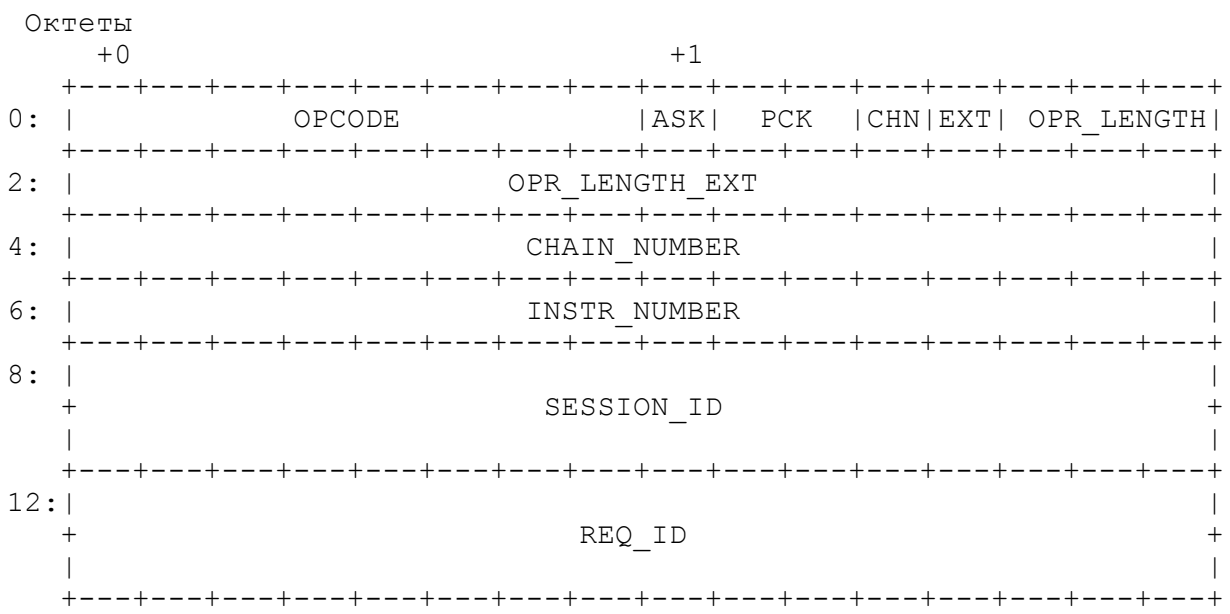
Формат инструкции спроектирован таким образом, что позволяет вычислить общую длину инструкции, не зная определение отдельного кода операции.

Для обеспечения эффективной работы в широком диапазоне сетевых скоростей заголовки инструкций предусматривают короткий и расширенный формат. Кроме этого имеется простой алгоритм сжатия заголовков.

Всем инструкциям и расширенным заголовкам присвоены идентификаторы, которые записываются заглавными буквами. Идентификаторы инструкций начинаются с буквы. Идентификаторы расширенных заголовков начинаются с символа подчеркивания.

3.1 Заголовок инструкции

Заголовок состоит из следующих полей:



OPCODE

1 октет. Код операции. Значение этого поля определяет инструкцию. Значения кодов операций делятся на следующие интервалы:

1 - 112	инструкции управления
113 - 127	зарезервированы
128 - 253	для инструкций обмена между VM
0, 254, 255	зарезервированы

ASK

1 бит. Флаг необходимости подтверждения. Это поле указывает на наличие поля REQ_ID. Если ASK = 1, поле REQ_ID присутствует в инструкции. Если ASK = 0, поле REQ_ID отсутствует.

PCK

2 бита. Признак сжатия заголовка. Эти биты используются для упаковки заголовков инструкций, передаваемых по одному соединению TCP или при передаче нескольких инструкций в одном сегменте UDP. Использование данных бит, основано на предположении, что две следующие подряд инструкции с большой вероятностью относятся к одному сеансовому соединению, или одной цепочке. Биты PCK имеют одно из следующих значений:

%b00 - инструкция не относится, к какой либо сессии. Поля CHAIN_NUMBER, INSTR_NUMBER и SESSION_ID в заголовке такой инструкции отсутствуют.
%b01 - данная инструкция относится к тому же сеансовому соединению, что и предыдущая. Поле SESSION_ID в заголовке инструкции отсутствует.

%b10 - данная инструкция относится к тому же соединению и той же цепочке, что и предыдущая. Поля CHAIN_NUMBER, INSTR_NUMBER и SESSION_ID в заголовке такой инструкции отсутствуют. Значение INSTR_NUMBER текущей инструкции получают прибавлением единицы к значению INSTR_NUMBER предыдущей инструкции.
%b11 - данная инструкция может не относится к той же сессии, что и предыдущая. Поле SESSION_ID в ней присутствует. Наличие полей CHAIN_NUMBER, INSTR_NUMBER определяется флагом CHN.

CHN

1 бит. Флаг цепочки. Относящиеся к одному заданию и передаваемые по одному сеансовому соединению инструкции могут быть объединены в цепочку. Подробно цепочки рассмотрены в секции 9. Если бит CHN установлен в 1, инструкция связана с цепочкой и в ней присутствуют поля CHAIN_NUMBER и INSTR_NUMBER (если PCK не равно %b10). Если бит CHN установлен в 0, инструкция не связана с цепочками и в ней отсутствуют поля CHAIN_NUMBER и INSTR_NUMBER.

EXT

1 бит. Флаг наличия в инструкции расширенных заголовков. Если он установлен в 1 – в инструкции имеется один или более расширенных заголовков. Если он установлен в 0 – расширенные заголовки в инструкции отсутствуют.

OPR_LENGTH

3 бита. Число 32 – битных слов в поле операндов. Значение 0 определяет отсутствие поля операндов. Значение %b111 указывает на использование расширенного формата заголовка. В расширенном формате длина операндов определяется полем OPR_LENGTH_EXT, а поле OPR_LENGTH не используется.

OPR_LENGTH_EXT

2 октета. Число 32 – битных слов в поле операндов. Поле OPR_LENGTH_EXT имеется в заголовке только, если OPR_LENGTH = %b111. Если OPR_LENGTH <> %b111, поле отсутствует. Если OPR_LENGTH_EXT = 0, то поле операндов отсутствует. Существуют следующие причины, по которым нужно использовать поле OPR_LENGTH_EXT вместо OPR_LENGTH:

(1) если длина операндов должна быть больше 24 октет;
(2) если выравнивание следующих полей на границу четырех октет более эффективно, чем сжатие заголовка на 2 октета.

CHAIN_NUMBER

2 октета. Номер цепочки. Указывает номер цепочки, к которой относится данная инструкция. Значения %x0000 и %xFFFF зарезервированы.

INSTR_NUMBER

2 октета. Номер инструкции. Определяет порядковый номер инструкции в цепочке. Нумерация начинается с нуля. Значения %xFFFF зарезервировано.

SESSION_ID

4 октета. Идентификатор сеансового соединения, установленный получателем инструкции. При установлении сеансового соединения каждая из сторон присваивает соединению свой идентификатор и сообщает его другой стороне. Нулевое значение этого поля указывает на то, что инструкция не относится, к какой либо сессии. Значение %xFFFFFFFF зарезервировано.

REQ_ID

4 октета. Идентификатор запроса. Используется для установления соответствия между запросом и ответом на него.

Далее в тексте при описании формата инструкций используются идентификатор OPR_LENGTH. Это подразумевает использование поля OPR_LENGTH_EXT, если OPR_LENGTH = %b111. Инструкция с длиной операндов, не превышающих 24 октета, может передаваться с заголовком в коротком формате (OPR_LENGTH <> %b111) или в расширенном формате (OPR_LENGTH = %b111). Обе формы являются эквивалентными.

Минимальная длина заголовка в коротком формате составляет 2 октета, в расширенном формате – 4 октета. Максимальная длина заголовка 16 октет.

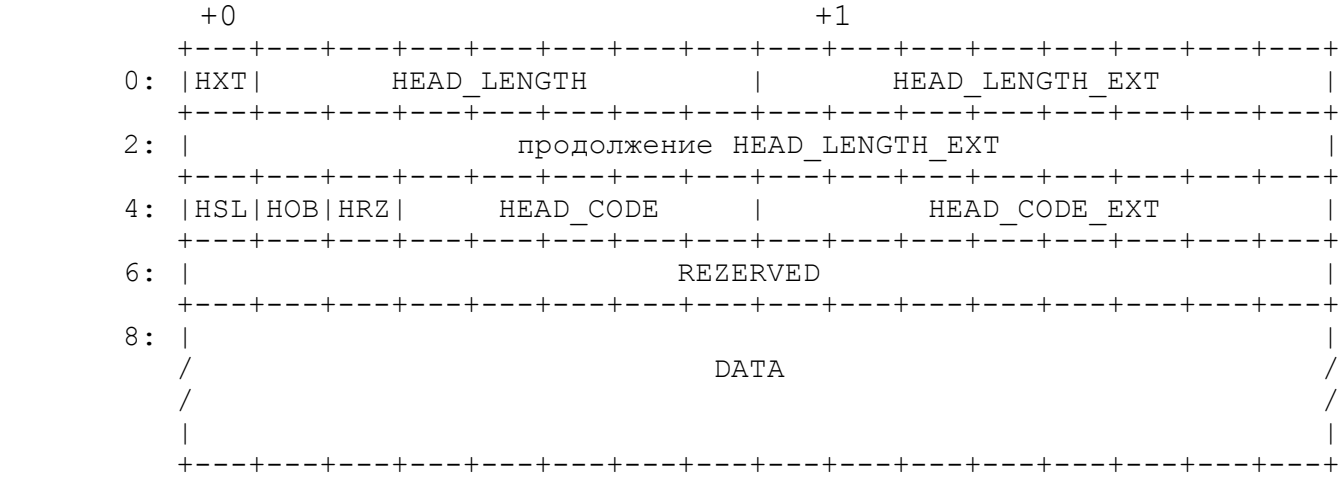
3.2 Расширенные заголовки

Если флаг EXT в заголовке инструкции установлен в 1, инструкция содержит от одного до 30 расширенных заголовков. Расширенные заголовки используются для следующих целей:

- Для передачи служебной информации, не предусмотренной в основном заголовке.
- Для передачи в одной инструкции данных длиной более 262240 октет.

Расширенные заголовки имеют следующий формат:

Октеты:



HXT

1 бит. Определяет длину поля длины данных. Если HXT = 0, длина расширенного заголовка определяется полем HEAD_LENGTH. Поле HEAD_LENGTH_EXT в этом случае отсутствует. Если HXT = 1, длина заголовка определяется объединением полей HEAD_LENGTH и HEAD_LENGTH_EXT.

HEAD_LENGTH

7 бит. Число 16 – битных слов в поле DATA. Если HXT = 0, это самостоятельное поле. Если HXT = 1, это старшие разряды полного поля длины.

HEAD_LENGTH_EXT

3 октета. Число 16 – битных слов в поле DATA. Если HXT = 0, это поле отсутствует. Если HXT = 1, это младшие разряды полного поля длины.

HSL

1 бит. Флаг последнего заголовка. Он устанавливается в 1 для последнего расширенного заголовка в инструкции. В остальных расширенных заголовках этот флаг устанавливается в 0.

HOB

1 бит. Флаг обязательной обработки. Он определяет порядок обработки инструкции, если принимающий узел не знает назначение этого расширенного заголовка или по какой либо причине не может его обработать. Если HOB = 1, то в этом случае инструкция не должна выполняться. Если HOB = 0, это не влияет на обработку инструкции. Протокол должен обрабатывать все расширенные заголовки, независимо от наличия ошибок.

HRZ

1 бит. Поле зарезервировано для будущих расширений. Это поле не должно анализироваться протоколом на приеме. При передаче оно должно устанавливаться в 0.

HEAD_CODE

5 бит. Если HXT = 0, в этом поле содержится код расширенного заголовка. Если HXT = 1, это поле присоединяется к полю HEAD_CODE_EXT – это старшие разряды кода заголовка.

HEAD_LENGTH_EXT

1 октет. Если HXT = 0, это поле отсутствует. Если HXT = 1, это младшие разряды кода заголовка.

REZERVED

2 октета. Если HXT = 0, это поле отсутствует. Если HXT = 1, это поле зарезервировано для дальнейшего использования. В текущей реализации протокола поле REZERVED не должно анализироваться протоколом на приеме. При передаче оно должно устанавливаться в 0.

DATA

Поле данных расширенного заголовка. Если HXT = 0, длина поля 0 – 254 октет, если HXT = 1, то 0 - 4 * 10^9 октет. Формат этого поля определяется отдельно для каждого значения кода заголовка.

На приемной стороне расширенные заголовки должны обрабатываться в том порядке, в каком они следуют в инструкции. Если инструкция содержит более 30 расширенных заголовков, то она считается ошибочной и нужно разорвать сеансовое соединение, по которому эта инструкция была передана.

Далее в тексте при описании формата расширенных заголовков используются идентификаторы HEAD_LENGTH и HEAD_CODE. Это

подразумевает использование полей HEAD_LENGTH + HEAD_LENGTH_EXT и HEAD_CODE + HEAD_CODE_EXT, если HXT = 1. Заголовки с кодом 0 - 30 могут передаваться и в коротком (HXT = 0) и в расширенном (HXT = 1) формате.

3.3 Операнды инструкции

Поле операндов содержит данные инструкции. Длина поля операндов указывается в OPR_LENGTH или OPR_LENGTH_EXT и кратна четырем октетам. Для выравнивания границы в конце поля записываются нулевые октеты. Максимальная длина операндов - 262140 октет. Если инструкция должна содержать более длинные данные, используются расширенные заголовки.

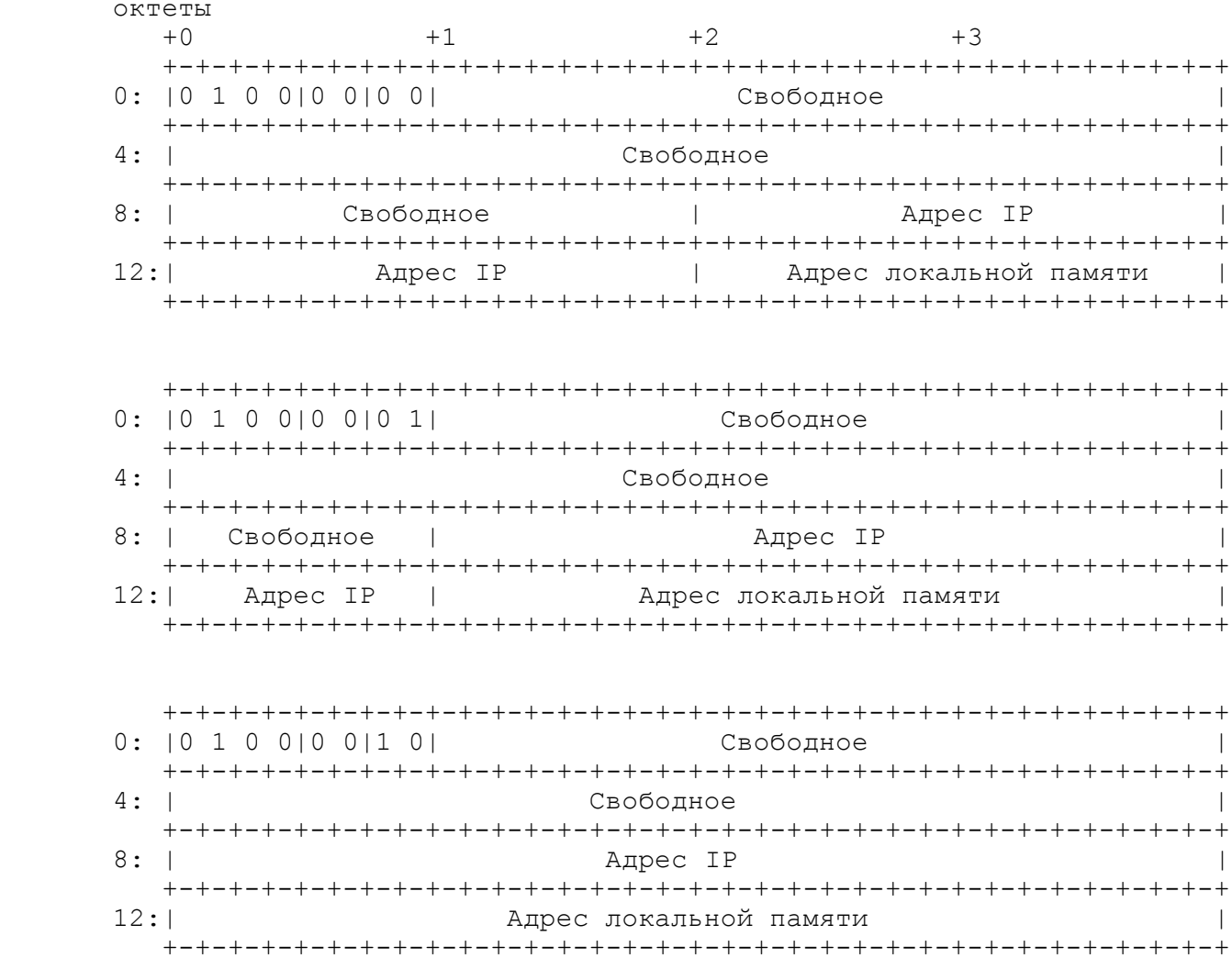
Формат поля операндов определяется отдельно для каждой инструкции.

3.4 Форматы адреса

Для непосредственно подключенных к глобальной сети IPv4 узлов определены следующие номера формата адреса:

№ 4-0-0 (4)
№ 4-0-1 (4-1)
№ 4-0-2 (4-2)

Соответствующие форматы 128-разрядных адресов:



Свободное

Не используемые протоколом биты.

Адрес IP

Задает адрес узла в глобальной сети IPv4.

Адрес локальной памяти

Описан в пункте 2.1.

Узлы данного типа однозначно определяются по IP-адресу. Для взаимодействия с адресами данного типа данный документ определяет TCP. Для передачи инструкций, не требующих подтверждения, допускается использование UDP. Выделенные IANA порты TCP и UDP - 2110. Этот порт должен быть открыт для приема. Инициирующий установление соединения TCP или осуществляющий передачу пакета UDP узел может использовать любой порт. Допускается использование множественных соединений TCP с мультиплексированием.

4 Подтверждение инструкций

Инструкции протокола делятся на два типа:

(1) Управляющие инструкции, передаваемые на уровне UMSP (OPCODE = 1 - 112).
(2) Инструкции обмена между VM (OPCODE = 128 – 253).

Обработка двух типов инструкций различается следующим образом:

- Поле идентификатора запроса REQ_ID в инструкциях первого типа формируется протоколом, для инструкций второго типа – VM.
- Протокол должен анализировать поле REQ_ID и сопоставлять его с переданными ранее инструкциями при приеме ответа на инструкции первого типа
- Протокол не должен анализировать поле REQ_ID при приеме ответов на инструкции второго типа. Ответная инструкция просто передается VM.

Инструкции подтверждения имеют поле ASK равное 1. Это означает, что в заголовке присутствует поле REQ_ID. В поле REQ_ID записывается значение, взятое из подтверждаемой инструкции. Инструкция подтверждения не требует подтверждения.

К протоколу на узле могут быть подключены несколько VM. Каждая VM может работать в своем адресном пространстве. Идентификаторы запросов для разных VM могут совпадать. Поэтому инструкция идентифицируется по двум полям:

- идентификатор сессии SESSION_ID, который связан с определенной VM;
- идентификатор запроса REQ_ID.

4.1 RSP, RSP_P

Инструкции «Подтверждение» (RSP) и «Подтверждение протокола» (RSP_P) имеют одинаковый формат. Отличие только в коде операции:

```
OPCODE = 129/1  ; соответственно для RSP/RSP_P
ASK = 1
PCK = %01/11
EXT = 0/1
CHN = 0
OPR_LENGTH = 0/1
SESSION_ID и REQ_ID - значения берутся из подтверждаемой
                      инструкции.
```

Операнды:
2 октета: основной код возврата.
2 октета: дополнительный код возврата.
Необязательный расширенный заголовок:
_MSG – содержит произвольное описание ошибки.

Инструкция без операндов используется для положительного подтверждения. Она эквивалентна нулевым значениям поля основного и дополнительного кодов возврата

Нулевой основной код возврата всегда говорит о том, что инструкция успешно выполнена. Дополнительный код возврата может иметь ненулевое значение.

Инструкция с ненулевым основным кодом возврата используется для отрицательного подтверждения. Основной код возврата определяет категорию ошибки. Дополнительный код возврата идентифицирует ошибку.

Инструкция RSP формируется по запросу VM. Коды возврата также должны быть получены от VM. Если протокол не может доставить до VM инструкцию, требующую подтверждения, он самостоятельно формирует отрицательное подтверждение RSP.

Инструкция RSP_P всегда формируется на уровне UMSP. Если протокол не может определить на какую инструкцию передана RSP_P, он просто ее отбрасывает.

4.2 SND_CANCEL

При передаче длинных фрагментированных инструкций или транзакций может возникнуть необходимость отмены передачи после того, как часть данных уже передана и занимает буфер на приемной стороне. Протокол предусматривает для этого инструкцию «Передача отменена» (SND_CANCEL). Эта инструкция имеет следующие значения полей:

OPCODE = 2
ASK = 0
PCK = %b01/10/11
EXT = 0/1
CHN = 1
OPR_LENGTH = 1
SESSION_ID - значения берется из отменяемой цепочки.
CHAIN_NUMBER - номер отменяемой цепочки
INSTR_NUMBER - всегда имеет нулевое значение
Операнды:
2 октета: основной код возврата.
2 октета: дополнительный код возврата.
Необязательный расширенный заголовок:
_MSG – содержит произвольное описание ошибки.

Инструкция используется для отмены частично переданной транзакции или фрагментированной инструкции. При приеме инструкции SND_CANCEL все ранее принятые данные из этой цепочки отбрасываются.

5 Управление заданиями

Управление заданиями включает в себя следующие функции:

- инициирование и завершение заданий;
- инициирование и завершение задач;
- открытие и закрытие сеансовых соединений;
- контроль активности узлов.

Для управления заданиями используются инструкции с OPCODE 1 - 112. Эти инструкции должны передаваться через TCP. Использование UDP не допускается, даже если инструкция не требуют подтверждения.

UMSP опирается на модель с централизованным управлением отдельного задания. Это связано с тем, что управление ссылками в децентрализованной системе не представляется возможным. Любая задача может в любой момент завершиться или узел может перезагрузиться. В децентрализованной системе не существует способа, гарантирующего оповещение об этом всех остальных узлов, на которых работает задание. Так как задание продолжает существовать - на том же узле может быть вновь инициирована относящаяся к заданию задача. Эта задача может выделить новые динамические ресурсы. Адреса для вновь выделенных ресурсов могут пересекаться с адресами ресурсов, которые существовали на узле до рестарта задачи. На других узлах могут сохраниться старые указатели. Формально это могут быть корректные указатели. В действительности они будут указывать на другие объекты. Следствием такой ситуации может быть неуправляемая работа приложения.

UMSP решает эту задачу следующим образом:

- он позволяет точно определить узел, на котором задача была завершена,
- если задача на узле завершается до завершения задания, об этом оповещают все узлы, на которых выполняется заданию,
- не допускается повторная инициализация задачи на узле до тех пор, пока все остальные узлы не получат сообщение о завершении первой задачи.

Протокол не управляет указателями. Корректность указателей контролирует VM. Для этого VM должны иметь архитектуру, в которой 128 - разрядные указатели и указатели на динамические объекты хранятся в специальных областях памяти. Протокол сообщает VM об узлах, на которых задачи завершили свою работу. VM должна сделать все указатели, относящиеся к таким задачам, недействительными. Это приводит к возникновению исключительных ситуаций при обращении по этим указателям. Если приложение предусматривает обработку исключительных ситуаций, оно сохраняет свою работоспособность. Иначе оно аварийно завершается. Такое решение позволяет исключить неуправляемую работу приложений.

Для решения указанных вопросов на уровне UMSP для каждого задания определяется узел, который будет управлять заданием. Такой узел называется Job Control Point (JCP). Это может быть тот же узел, на котором инициируется задание, либо это может быть другой, специально выделенный узел. Основная функция JCP - отслеживать инициализацию и завершения задач задания. Кроме этого выделенный узел JCP может использоваться для централизованной идентификации пользователей и защиты от атак.

Для управления заданиями и задачами определены следующие идентификаторы:

- Каждой активной задаче на узле присваивается Локальный Идентификатор Задачи (Locally Task Identifier - LTID). Длина LTID равна длине адреса памяти, установленной для узла. В каждый момент времени все LTID на узле должна иметь уникальные значения. Для вновь инициируемых задач допускается устанавливать LTID, использованные ранее в уже завершенных задачах.
- Каждой задаче задания JCP присваивает Управляющий Идентификатор Задачи (Control Task Identifier - CTID). Его длина равна длине адреса локальной памяти на узле JCP. В каждый момент времени все CTID на JCP должна иметь уникальные значения. В отличие от

LTID на выбор значения CTID накладываются некоторые ограничения.

- Каждой задаче присваивается Глобальный Идентификатор Задачи (Globally Task Identifier - GTID). GTID имеет такой же формат, как и 128 - разрядный адрес памяти для узла. В нем адрес локальной памяти заменен на LTID.
- Для задания устанавливается Глобальный Идентификатор Задания (Globally Job Identifier - GJID). GJID устанавливается на узле JCP. GTID имеет такой же формат, как и 128 - разрядный адрес памяти для JCP. В нем адрес локальной памяти заменен на CTID первой (инициирующей) задачи задания. GJID используется в процедуре установления сеансового соединения для определения JCP, который управляет заданием.

LTID и CTID записываются в инструкциях в поле длиной 2/4/8 октет. Если в инструкции выделенное для идентификатора поле длиннее идентификатора, то LTID (CTID) записывается в последние октеты. В начальные октеты должно записываться значение 0. Если принятый идентификатор LTID (CTID) короче адреса памяти, он должен быть дополнен спереди необходимым числом нулевых октет.

GTID и GJID записываются в инструкциях в поле длиной 4-16 октет. Поле FREE в этих идентификаторах не присутствует (см. секцию 2.1). Считается, что оно состоит из нулевых октет. Длина идентификатора определяется по заголовку адреса.

При передаче инструкций CONTROL_REQ, TASK_REG and SESSION_OPEN протокол использует тайм-ауты. Величина тайм-аута устанавливается самим узлом и должна быть больше трех интервалов максимального времени передачи пакета на транспортном уровне. На тайм-аут не влияет время ожидания в очереди к транспортному уровню.

5.1 Инициация задания

Задание относится к выполняемому на VM приложению пользователя. Инициация задания UMSP может производиться или одновременно со стартом приложения пользователя или в процессе его работы.

Вместе с заданием на узле инициируется соответствующая ему задача. Этой задаче присваивается LTID.

Если для JCP выбран узел, на котором загружено приложение пользователя, вопрос инициализации задания лежит вне границ сетевого протокола.

Другой узел может быть выбран в качестве JCP по следующим причинам:

- узел инициации задания подключен к сети по низкоскоростному или перегруженному каналу. По такому каналу нежелательно передавать управляющий трафик.
- Узел не имеет вычислительных возможностей для ведения управляющих таблиц.
- Требуется идентификация на специально выделенном узле.

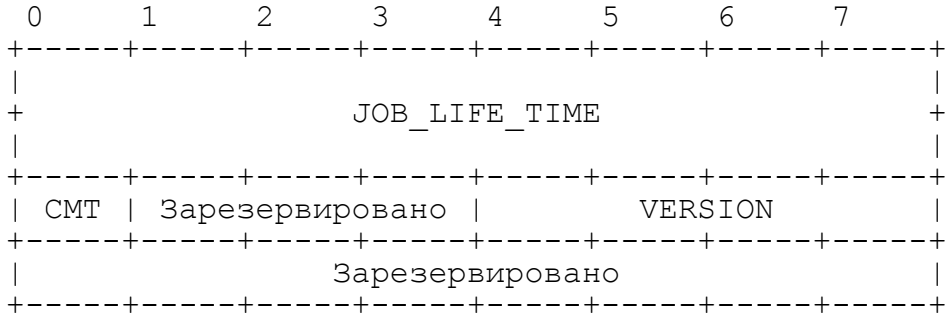
Если для JCP выбран другой узел, то инициирующий задание узел должен зарегистрировать свое задание у JCP.

5.1.1 CONTROL_REQ

Инструкция «Запрос управления» (CONTROL_REQ) передается от инициирующего задание узла к JCP другого узла. Инструкция имеет следующие значения полей:

OPCODE = 3

PCK = %b00
CHN = 0
ASK = 1
EXT = 0/1
OPR_LENGTH = 2/3 ; Зависит от длины LTID.
REQ_ID - значение устанавливается протоколом узла отправителя и затем передается в ответе.
Операнды:
4 октета: профиль параметров управления. Это поле имеет следующий формат:



JOB_LIFE_TIME

2 октета. Время жизни задания в секундах. Значение 0 указывает, что ограничение времени жизни не будет использоваться.

CMT

1 бит. Флаг использования нескольких JCP. Это поле зарезервировано для будущего расширения протокола.

VERSION

1 октет. Номер версии протокола UMSP. Должно иметь значение 1.

Зарезервировано

3 + 8 бит. Все биты должны быть установлены в 0.

4/8 октет: Локальный идентификатор задачи задания LTID, установленный на инициирующем задание узле (посылающим эту инструкцию).
Необязательные расширенные заголовки:
_NAME - Имя задания. Имя задания устанавливается один раз и не должно изменяться в дальнейшем.
_INACT_TIME – Время неактивности узла (см. секцию 5.7).

На приеме инструкции CONTROL_REQ JCP проверяет значение LTID из полученной инструкции и делает следующее:

(1) Если узел, который послал CONTROL_REQ, уже зарегистрировал на JCP активное задание с таким LTID, посылается уведомление об аварийном завершении зарегистрированного задания, как описан в разделе 5.5.2. После этого посылается разрешение на создание нового задания.
(2) Если узел не имеет зарегистрированного задания с полученным LTID, разрешение на создание нового задания передается сразу.

Если JCP подтверждает управление, то он предает инструкцию CONTROL_CONFIRM, иначе CONTROL_REJECT.

5.1.2 CONTROL_CONFIRM

Инструкция «Подтвердить управление» (CONTROL_CONFIRM) передается от JCP для положительного ответа на CONTROL_REQ. CONTROL_CONFIRM имеет следующие значения полей:

```
OPCODE = 4
PCK = %b00
CHN = 0
ASK = 1 ; Инструкцию не нужно подтверждать. Этот флаг указывает
          на наличие поля REQ_ID.
EXT = 0/1
OPR_LENGTH = 1-4 ; Зависит от длины GTID.
REQ_ID – записывается значение из инструкции CONTROL_REQ
Операнды:
   4-16 октет: GJID, присвоенный заданию на JCP.
```

Передача инструкции CONTROL_REQ подразумевает запрос на управление и запрос на инициацию задачи. Присвоенный задаче CTID является частью GJID (поле адреса локальной памяти).

5.1.3 CONTROL_REJECT

Инструкция «Отклонить управление» (CONTROL_ REJECT) передается от JCP для отрицательного ответа на CONTROL_REQ. CONTROL_CONFIRM имеет следующие значения полей:

```
OPCODE = 5
PCK = %b00
CHN = 0
ASK = 1. Инструкцию не нужно подтверждать. Этот флаг указывает
         на наличие поля REQ_ID.
EXT = 0/1
OPR_LENGTH = 1/2  ; Зависит от наличия в инструкции поля
                    параметров управления.
REQ_ID – записывается значение из инструкции CONTROL_REQ
Операнды:
   2 октета: Основной код возврата. Значение 0 не допускается.
   2 октета: Дополнительный код возврата.
   4 октета: Профиль параметров управления (см. секцию 5.1.1),
             которые могут быть предоставлены JCP. Данное поле
             является необязательным.
Необязательные расширенные заголовки:
   _INACT_TIME – Время неактивности узла (см. секцию 5.7).
   _MSG – содержит произвольное описание ошибки.
```

5.2 Инициализация задачи

Задание одновременно выполняется на нескольких узла. На каждом узле должна быть инициализирована соответствующая ему задача. Одному заданию на узле соответствует только одна задача. Каждая задача может быть связана только с одним заданием.

На создавшем задание узле задача инициируется вместе с заданием. На остальных узлах задача инициируется при получении первого запроса на установление сеансового соединения, соответствующего заданию. Запрос на установления сеансового соединения содержит GJID. GJID содержит адрес JCP. Для старта задачи нужно получить разрешение от JCP. Если запрос на установление сеанса поступил от узла JCP, запрашивать разрешение не нужно.

5.2.1 TASK_REG

Инструкция «Зарегистрировать задачу» (TASK_REG) передается от инициирующего задачу узла к JCP удаленного узла. Инструкция имеет следующие значения полей:

```
OPCODE = 6/7/8 ; Для длины поля CTID 2/4/8 октета
соответственно.
PCK = %b00
CHN = 0
ASK = 1
EXT = 0/1
OPR_LENGTH = 2-8 ; Зависит от длины GJID, GTID и LTID.
REQ_ID – значение устанавливается протоколом узла отправителя и
         затем передается в ответе.
Операнды:
   2/4/8 octets: CTID задачи, которая инициировала задание.
                 CTID является частью GJID из инструкции
                 SESSION_OPEN.
   4-16 октет: GTID, присвоенный на инициирующем сеансовое
               соединение узле. GTID формируется из адреса
               отправителя (на транспортном уровне) и поля LTID
               инструкции SESSION_OPEN.
   2/4/8 октет:  LTID, установленный на инициирующем задачу
                 узле (посылающим эту инструкцию).
Необязательные расширенные заголовки:
   _INACT_TIME – Время неактивности узла (см. секцию 5.7).
```

Инструкция TASK_REG должна посылаться, только если задача с заданным GJID на узле не инициирована.

JCP подтверждает инициацию задачи при соблюдении следующих условий:

(1) Задача с полученным GTID уже была зарегистрирована на JCP
(2) Задача с LTID для узла, инициирующего задачу, не была зарегистрирована.

Во всех остальных случаях JCP не подтверждает задачу.

Если JCP подтверждает задачу, он передает инструкцию TASK_CONFIRM, иначе TASK_REJECT.

5.2.2 TASK_CONFIRM

Инструкция «Подтвердить задачу» (TASK_CONFIRM) передается от JCP в качестве положительного ответа на TASK_REG. TASK_CONFIRM имеет следующие значения полей:

```
OPCODE = 9
PCK = %b00
CHN = 0
ASK = 1. Инструкцию не нужно подтверждать. Этот флаг указывает
         на наличие поля REQ_ID.
EXT = 0/1
OPR_LENGTH = 1/2 ; Зависит от длины CTID.
REQ_ID – записывается значение из инструкции TASK_REG
Операнды:
   4/8 октет: CTID присвоенный задаче на JCP.
Необязательные расширенные заголовки:
   _JOB_NAME - Этот заголовок содержит имя задания.
```

5.2.3 TASK_REJECT

Инструкция “Отклонить задачу” (TASK_REJECT) передается от JCP для отрицательного ответа на инструкцию TASK_REG. TASK_REJECT имеет следующие значения полей:

OPCODE = 10
PCK = %b00
CHN = 0
ASK = 1. Инструкция не требует подтверждения. Этот флаг указывает на наличие поля REQ_ID.
EXT = 0/1
OPR_LENGTH = 1
REQ_ID – Значение берется из инструкции TASK_REG
Операнды:
2 октета: Основной код возврата. Нулевое значение не допускается.
2 октета: Дополнительный код возврата.
Необязательные расширенные заголовки:
_INACT_TIME – Время неактивности узла (см. секцию 5.7).
_MSG – содержит произвольное описание ошибки.

5.2.4 TASK_CHK

В целях обеспечения безопасности получивший запрос на установление сеансового соединения узел, может проверить у JCP инициирующий соединение узел, даже если задача уже была инициирована.

Инструкция «Проверить задачу» (TASK_CHK) передается к JCP от узла, получившего инструкцию установления сеансового соединения. Задача с заданным GJID уже должна существовать на узле. Формат инструкции TASK_CHK совпадает с TASK_REG. OPCODE = 11. Ответ на инструкцию TASK_CHK JCP формирует аналогично инструкции TASK_REG.

JCP подтверждает инструкцию TASK_CHK, если задачи с полученным GTID и LTID уже зарегистрировались на JCP.

Инструкция TASK_CHK является опциональной для узла, не являющегося JCP.

5.3 Установление сеансового соединения

Сеансовое соединение устанавливается между двумя задачами одного задания. Соединение устанавливается по инициативе VM и используется для обмена инструкциями между VM.

С одним сеансовым соединением может быть связано только одна задача на узле. Задача может иметь несколько соединений с разными узлами. Между двумя узлами может существовать только одно сеансовое соединение с одним GJID.

Запрос на установление сеансового соединения содержит глобальный идентификатор задания GJID. Если узел получает запрос на установление соединения с GJID, которого нет на данном узле, VM должна создать новую задачу. Если задача уже имеется, новая задача не создается.

Сеансовое соединение нужно устанавливать по TCP. После того, как соединение установлено, через UDP возможна передача инструкций, не требующих подтверждение выполнения. Одно соединение TCP могут использовать несколько сеансовых соединений. Одно сеансовое соединение может использовать несколько соединений TCP.

Протокол допускает работу без установления сеансового соединения. Узел должен иметь VM по умолчанию, которая обеспечивает работу без установления соединения.

При установлении сеансового соединения стороны договариваются об используемом типе VM и подмножестве функций протокола. Сеансовое соединение UMSP может быть несимметричным. Это означает, что две стороны одного соединения могут быть связаны с VM разного типа и поддерживать разное подмножество функций протокола.

Если при установлении соединения используется нулевой тип VM, это указывает на группу VM (см. секцию 9). Версия VM в этом случае должна иметь ненулевое значение.

Процедура установления сеансового соединения является многоходовой и может содержать от двух до восьми шагов.

5.3.1 SESSION_OPEN

Инструкция «Открыть соединение» (SESSION_OPEN) используется для инициирования сеансового соединения, и при уточнении параметров соединения в процессе диалога. Она имеет следующие значения полей:

```
OPCODE = 12
PCK = %b00/%b11. В первой инструкции (инициирующей) значение
                 этого поля установлено в %b00. Во всех
                 последующих – %b11.
CHN = 0
ASK = 1
EXT = 0/1
OPR_LENGTH = 6 – 11 ; Зависит от длины GJID и GTID.
SESSION_ID - В первой инструкции это поле отсутствует. Во всех
             последующих оно содержит идентификатор сессии,
             установленный получателем инструкции.
REQ_ID – Это поле содержит идентификатор сессии, установленный
         отправителем инструкции.
Операнды:
   2 октета: Тип VM требуемый от получателя.
   2 октета: Версия VM требуемая от получателя.
   4 октета: Профиль соединения, требуемый от получателя
             инструкции.
   2 октета: Тип VM отправителя.
   2 октета: Версия VM отправителя.
   4 октета: Профиль соединения, предоставляемый отправителем
             инструкции.
   2 октета: Число 256-октетных блоков в буфере ("окне"),
             выделенном для сеанса на стороне отправителя этой
             инструкции (см. раздел 7.4). Нулевое значение
             определяет отсутствие буфера.
   4-16 октет: GJID.
   4/8 октет: LTID задачи, установленный на узле – отправителе
              инструкции. Он используется в инструкции
              TASK_REG.
```

Если тип и версия VM, требуемые от получателя, имеют значение 0, принимающий узел самостоятельно выбирает тип VM и сообщает его в ответе. Установление соединения без привязки к VM или группе VM не допускается.

Всего при установлении соединения может быть передано до 7 инструкций SESSION_OPEN. Для подтверждения установления соединения используется инструкция SESSION_ACCEPT. Для отказа от соединения используется инструкция SESSION_REJECT.

Отказаться от соединения можно на любом шаге. На восьмом шаге нужно либо подтвердить соединения, либо отказаться от него.

В процессе установления соединения могут быть изменены следующие параметры:

- тип VM и версия VM;
- профили соединения.

Если от определенного узла поступил повторный запрос на открытие сеансового соединения, в то время как одно соединение с полученным GJID уже установлено, возможны следующие варианты:

(1) Если запрос поступил от узла JCP, то нужно:
   - Аварийно завершить существующую задачу и освободить все принадлежащие ей динамические ресурсы.
   - Инициировать новую задачу, как описано в п. 5.2.
   - Подтвердить установление соединения

(2) Если запрос поступил не от узла JCP, то нужно отказаться от установления нового сеансового соединения. Существующую задачу изменять не нужно.

5.3.2 SESSION_ACCEPT

Инструкция «Согласиться с соединением» (SESSION_ACCEPT) используется для положительного подтверждения на установления сеансового соединения. Она имеет следующие значения полей:

```
OPCODE = 13
ASK = 1
PCK = %b11
EXT = 0/1
CHN = 0
OPR_LENGTH = 0
SESSION_ID – Это поле содержит идентификатор сессии,
             установленный узлом получателем инструкции.
REQ_ID – Это поле содержит идентификатор сессии, установленный
         отправителем инструкции.
```

5.3.3 SESSION_REJECT

Инструкция «Отклонить сессию» (SESSION_REJECT) используется для отрицательного ответа на установления сеансового соединения. Она имеет следующие значения полей:

```
OPCODE = 14
ASK = 0
PCK = %b11
EXT = 0/1
CHN = 0
OPR_LENGTH = 1
SESSION_ID – Это поле содержит идентификатор сессии,
             установленный узлом получателем инструкции.
Операнды:
   2 октета: Основной код возврата. Нулевое значение не
             допускается.
   2 октета: Дополнительный код возврата.
Необязательные расширенные заголовки:
   _MSG – содержит произвольное описание ошибки.
```

5.3.4 Профиль соединения

Профиль соединения определяется в 4 – октетном поле флагов. Флаги имеют идентификаторы S0 – S31. Число в идентификаторе определяет порядковый номер бита. Если флаг имеет значение 1, то связанная с ним функция поддерживается (требуется). Если флаг имеет значение 0, то связанная с ним функция не поддерживается (не требуется). Далее приводится список функций, определяемых при установлении сеансового соединения.

Работа с цепочками:

S0 - использование фрагментированных инструкций
S1 - использование последовательностей
S2 - использование транзакций

Установление соединения:

S3 - обмен данными без установления соединения
S4 - обмен данными с установлением соединения

Формат инструкций:

S5 - Зарезервирован. Должно быть установлено в 0.
S6 - Поддержка 16-октетного адреса в инструкциях обмена между VM.
S7 - Разрешено использование сжатой формы заголовка инструкции (OPR_LENGTH <> %b111)
S8 - Разрешено использование расширенной формы заголовка инструкции (OPR_LENGTH = %b111)
S9 - поддержка расширенных заголовков с полем данных длиной до 254 октет
S10 - поддержка расширенных заголовков с полем данных длиной до 4 * 10^9 октет
S11-S15. Максимальная длина поля данных в операндах в 4 - октетных словах. Эти биты являются единым полем. Максимальная длина в октетах вычисляется по формуле <max длина> = (<значение этого поля> + 1) * 4. Если поле равно %b1111, максимальная длина данных определяется форматом инструкции.
S16-S19. Эти биты являются одним полем. В профиле, запрашиваемом от получателя инструкции, это поле содержит версию протокола. Оно должно иметь значение %b0001. В профиле, предоставляемым отправителем инструкции, это поле содержит приоритет задания. Чем больше значение поля, тем больше приоритет. Приоритет задания используется:
- В очередях на передачу к транспортному уровню для инструкций задания.
- Для задания параметров транспортного уровня.
- Для установки вычислительного приоритета задачи.

S16 - выравнивание на границу, кратную четырем октетам. Если S16=1, то:
(1) OPR_LENGTH = %b111
(2) С границы кратной четырем октетам начинается каждый расширенный заголовок и поле операндов.
(3) В конец каждого заголовка добавляется необходимое число нулевых октет.

S21 - Использование имен процедур объектов.
S22 - Использование имен объектов.

Разрешенные инструкции:

S23 - поддерживается подтверждение выполнения на VM(инструкция RSP)
S24 - Использование инструкций чтения и сравнения данных.
S25 - Использование инструкций записи данных.
S26 - Использование инструкций передачи управления.
S27 - Использование инструкции синхронизации.
S28 - Использование инструкций работы с объектами.
S29 - Использование прямого доступа к памяти объектов. Если этот флаг установлен в 0, доступ к объекту разрешен только через его процедуры. Если S28=0, этот флаг должен быть установлен в 0.
S30 - Использование инструкции MVRUN в 0-сессии.

S31 - Зарезервирован. Должен быть установлен в 0.

5.4 Закрытие сессии

Инициировать закрытие сеансового соединения может только узел, который инициировал его установление. Для этого он использует инструкцию SESSION_CLOSE. Процедура разрыва соединения является 3-ходовой. Предусмотрена процедура аварийного безусловного завершения соединения, которая может быть инициирована любым узлом.

Пусть узел А являлся инициатором установления сеанса, а узел Б является второй стороной соединения. Для инициации закрытия сессии узел А должен послать инструкцию SESSION_CLOSE. После посылки этой инструкции узел А может возобновить передачу инструкций. Это означает, что он отказался от закрытия соединения. Инструкции подтверждения (см. секцию 6) не влияет на закрытие соединения. Узел, пославший SESSION_CLOSE, не использует тайм-аута и может ожидать подтверждение неограниченно долго.

Узел Б после получения инструкции SESSION_CLOSE, посылает в ответ инструкцию RSP_P. Нулевой основной код возврата подтверждает закрытие сессии. Ненулевой основной код возврата отменяет закрытие сессии. После передачи положительного подтверждения узел Б не должен использовать соединение в течение 30 секундного тайм-аута. Если по истечении этого времени от узла А не поступила инструкция SESSION_ABEND или любая другая инструкция, кроме инструкций подтверждения, узел Б посылает инструкцию SESSION_ABEND и считает сеансовое соединение закрытым.

Узел А после получения положительного подтверждения на инструкцию SESSION_CLOSE посылает инструкцию SESSION_ABEND. После этого соединения считается закрытым. Узел А может не подтверждать закрытие соединения. Для этого посылается любая инструкция, включая NOP. В этом случае процедура завершения прерывается, и сеансовое соединение переводится в работающее состояние.

5.4.1 SESSION_CLOSE

Инструкция «Закрыть сессию» (SESSION_CLOSE) инициирует завершение сеансового соединения. Она имеет следующие значения полей:

```
OPCODE = 15
PCK = %b01/%b11
CHN = 0
ASK = 0
EXT = 0/1
OPR_LENGTH = 0/1
SESSION_ID - содержит идентификатор сессии, установленный
             получателем
Операнды:
   2 октета: Основной код завершения.
   2 октета: Дополнительный код завершения.
Расширенные заголовки:
   _MSG - необязательный заголовок с произвольным сообщением.
```

Операнды могут отсутствовать. Это эквивалентно нулевому коду завершения.

5.4.2 SESSION_ABEND

Инструкция «Аварийное завершение сессии» (SESSION_ABEND) применяется для безусловного завершения сессии. Узел, пославший эту инструкцию, завершает обмен данными по соединению в обе

стороны, не дожидаясь подтверждения от другого узла. Инструкция имеет следующие значения полей:

```
OPCODE = 16
PCK = %b01/%b11
CHN = 0
ASK = 0
EXT = 0/1
OPR_LENGTH = 0/1
SESSION_ID - содержит идентификатор сессии, установленный
             получателем
Операнды:
   2 октета: Основной код завершения.
   2 октета: Дополнительный код завершения.
Расширенные заголовки:
   _MSG - необязательный заголовок с произвольным сообщением.
```

Операнды могут отсутствовать. Это эквивалентно нулевому коду завершения.

5.5 Завершение задачи

При нормальном завершения работы приложения пользователя задача завершается в процессе завершения задания. Эта процедура описана в следующем пункте. Следующие ситуации требуют завершить задачу независимо от задания:

- Для поддержания задачи на узле недостаточно вычислительных ресурсов;
- Узел завершает свою работу;
- Если VM по своим внутренним причинам приняла такое решение.

Ссылки на ресурсы, выделенные задачей, могут находиться на любом узле, на котором выполняется задание. Поэтому все узлы должны быть оповещены о завершении задачи.

Завершающий задачу узел должен аварийно завершить все сеансовые соединения, связанные с задачей (послать инструкции SESSION_ABEND).

5.5.1 TASK_TERMINATE

Инструкция «Завершение задачи» (TASK_TERMINATE) передается от узла, на котором завершается задача, к JCP. Инструкция имеет следующие значения полей:

```
OPCODE = 17
PCK = %b00
CHN = 0
ASK = 0
EXT = 0/1
OPR_LENGTH = 2/3  ; в зависимости от длины CTID
Операнды:
   2 октета: Основной код завершения.
   2 октета: Дополнительный код завершения.
   4/8 октет: CTID.
Расширенные заголовки:
   _MSG - необязательный заголовок с произвольным сообщением.
```

Сразу после передачи к JCP инструкции TASK_TERMINATE узел передает инструкцию безусловного завершения соединения ABEND_SESSION по всем сеансовым соединениям, связанным с задачей. После этого задача считается завершенной.

Если основной код возврата в инструкции TASK_TERMINATE равен 0, то оповещать о завершении задачи другие узлы не требуется. Такая ситуация возникает, если задача не выделяла динамических ресурсов. Если основной код возврата неравен 0, то после получения инструкции TASK_TERMINATE JCP должен оповестить о завершении задачи другие узлы, на которых выполняется задание. JCP отвечает за оповещение всех узлов задания о завершении задачи.

5.5.2 TASK_TERMINATE_INFO

Инструкция «Информация о завершении задачи» (TASK_TERMINATE_INFO) используется для оповещения о завершении задачи. Она передается от JCP к другим узлам, на которых выполняется задание. Инструкция имеет следующие значения полей:

```
OPCODE = 18
PCK = %b00
CHN = 0
ASK = 0
EXT = 0/1
OPR_LENGTH = 2-5  ; Зависит от длины GTID
Операнды:
   2 октета: Основной код завершения.
   2 октета: Дополнительный код завершения.
   4-16 октет: GTID завершенной задачи. JCP формирует GTID из
             LTID (из инструкции TASK_REG) и адреса
             транспортного уровня задачи.
Расширенные заголовки:
   _MSG - необязательный заголовок с произвольным сообщением.
```

Коды завершения переписываются из инструкции TASK_TERMINATE. При получении инструкции TASK_TERMINATE_INFO VM должен удалить (сделать недействительными) все ссылки на ресурсы, относящиеся к узлу, на котором работала завершенная задача.

5.6 Завершение задания

Задание завершается, когда завершается соответствующее ему приложение пользователя на узле, на котором оно было инициировано. Завершение задания происходит по инициативе VM. Кроме этого оно может быть завершено по инициативе JCP при окончании времени жизни задания или при завершении работы узла JCP.

5.6.1 JOB_COPLETED

Инструкция «Задание завершено» (JOB_COPLETED) передается от узла инициировавшего задание в сторону JCP. Она имеет следующие значения полей:

```
OPCODE = 19
PCK = %b00
CHN = 0
ASK = 0
EXT = 0/1
OPR_LENGTH = 2/3 ; Зависит от длины CTID.
Операнды:
   2 октета: основной код завершения
   2 октета: дополнительный код завершения.
   4/8 октета: CTID завершенной задачи задания. CTID является
               частью GJID задания.
Необязательные расширенные заголовки:
   _MSG - произвольное сообщение об ошибке.
```

Сразу после передачи к JCP инструкции JOB_COPLETED узел передает по всем сеансовым соединениям, связанным с заданием, инструкцию

безусловного завершения соединения ABEND_SESSION. После этого задание считается завершенным.

После получения инструкции JOB_COPLETED JCP должен оповестить о завершении задания узлы, на которых выполняется задание. JCP отвечает за оповещение всех узлов задания о завершении задания.

5.6.2 JOB_COPLETED_INFO

JOB_COPLETED_INFO – «Информация о завершении задания». Инструкция используется для оповещения о завершении задания. Она передается от JCP к другим узлам, на которых выполняется задание. Инструкция имеет следующие значения полей:

```
OPCODE = 20
PCK = %b00
CHN = 0
ASK = 0
EXT = 1
OPR_LENGTH = 2-5   ; Зависит от длины GJID
Операнды:
   2 октета: Основной код завершения.
   2 октета: Дополнительный код завершения.
   4-16 октет: GJID завершенного задания
Расширенные заголовки:
   _MSG - необязательный заголовок с произвольным сообщением.
```

Коды завершения переписываются из инструкции JOB_COPLETED. При приеме инструкции JOB_COPLETED_INFO узел должен сделать следующие:

(1) Удалить все связанные с заданием сеансовые соединения. При этом не нужно посылать сетевые примитивы.
(2) Аварийно завершить задачу задания и освободить все динамические ресурсы задачи.

Инструкция JOB_COPLETED_INFO также используется для завершения задания по инициативе JCP при завершении времени жизни задания или при завершении работы JCP. В этих случаях первым получателем инструкции является узел, инициировавший задание.

После передачи всех инструкций JOB_COPLETED_INFO JCP считает задание завершенным.

5.7 Контроль активности узлов

UMSP объединяет узлы, имеющие произвольное расположение в сети и не имеющие единого управления. Каждый из узлов в любой момент может быть выключен или перезагружен. При этом остальные узлы могут быть об этом не оповещены. Факт разрыва или повторного установления транспортного соединения не может служить индикатором отключения или рестарта узла. Управление транспортными соединениями не является частью протокола UMSP и наличие транспортного соединения не является обязательным.

Кроме того, может аварийно завершиться отдельная задача на узле. В этом случае должна быть проведена процедура, описанная в секции 5.5.1. Если эта процедура не может быть проведена, нужно завершить работу узла.

Функции контроля работоспособности узлов выполняет JCP. Для этого между задачами на узлах и JCP периодически передается инструкция запроса состояния TASK_REG.

Возможны следующие действия JCP при обнаружении отключения узла:

(1) Если завершилась задача, инициировавшая задание, считается, что завершено все задание. JCP всем остальным узлам, на которых выполнялось задание, передает инструкцию JOB_COPLETED_INFO.
(2) Если завершилась задача, не инициировавшая задание, JCP всем остальным узлам задания передает инструкция TASK_TERMINATE_INFO.

Отключение узла JCP накладывает ограничение на присваиваемые им после перезагрузки GJID. Возможны следующие варианты:

(1) Отключение узла JCP прошло нормальным образом. Он передал всем узлам, которыми он управлял, инструкцию JOB_COPLETED_INFO. В этом случае после перезагрузки он может присваивать любые GJID.
(2) Произошло аварийное отключение узла JCP. Он не сообщил всем узлам о своем выключении. В этом случае после перезагрузки он должен гарантировать, что в течение двух максимальных интервалов времени бездействия (которые устанавливает этот JCP) новые GJID не будут совпадать с GJID, существовавшем до перезагрузки.

Перезагрузка узлов, не являющихся JCP, не накладывает ограничений на устанавливаемые на этих узлах LTID.

5.7.1 _INACTION_TIME

Расширенный заголовок «Время неактивности» (_INACTION_TIME) используется для указания времени бездействия узла. Он имеет следующий формат:

```
HEAD_CODE = 2
HEAD_LENGTH = 1;
HOB = 1
DATA contains:
   2 октета. Период неактивности. Число 0,5 секундных
   интервалов, через которые будет проверяться активность узла.
```

Период бездействия должен быть больше трех интервалов максимального времени передачи пакета на транспортном уровне. Время ожидания в очереди к транспортному уровню не учитывается.

Заголовок _INACTION_TIME может быть присоединен к следующим инструкциям:

(1) К инструкции TASK_REG. В этом случае должно выполняться условие – на узле не должно существовать других активных задач, которыми управляет JCP получатель (хотя заголовок присоединяется к инструкции, которая относится к определенной задаче, он определяет период неактивности всего узла). Нулевое значение указывает, что проверка активности не будет использоваться. Отсутствие заголовка указывает, что период бездействия устанавливается узлом JCP.
(2) К инструкции TASK_REJECT, если JCP не подходит время из инструкции TASK_REG.
(3) В инструкции TASK_CONFIRM, если TASK_REG не содержала этого заголовка.

Если JCP получает инструкцию TASK_REG с присоединенным заголовком _INACTION_TIME (это может означать, что узел перезагрузился), то он должен проверить наличие активных задач с узлом отправителем. Если такие задачи существуют, для каждой из них необходимо выполнить процедуру завершения задачи, описанную в секции 5.6.2. Инструкция TASK_CONFIRM может быть послана только после этого.

5.7.2 STATE_REQ

Инструкция «Запрос состояния» (STATE_REQ) передается от JCP к определенной задаче другого узла. Инструкция имеет следующие значения полей:

```
OPCODE = 21
PCK = %b00
CHN = 0
ASK = 0
EXT = 0
OPR_LENGTH = 1/2 ; Зависит от длины LTID.
Операнды:
   4/8 октета: Локальный идентификатор задачи LTID,
               установленный на узле получателе инструкции.
```

Инструкция STATE_REQ передается в определенной задаче, но относится ко всему узлу. Она посылается, если между узлом и JCP не было передачи инструкции в течение времени неактивности. Задачи, активизированные после передачи последней инструкции STATE_REQ, не влияют на контроль активности.

В ответ на инструкцию STATE_REQ передается инструкция TASK_STATE. При ожидании ответа используется тайм-аут равный одному периоду неактивности. По истечении тайм-аута узел считается отключенным.

Если узел не получает никаких инструкций от JCP в течении двух интервалов времени неактивности, считается, что JCP аварийно завершил свою работу. Действия узла в этом случае описаны в секции 5.6.2 при приеме инструкции JOB_COMPLITED_INFO. Проверка этого условия является опциональной для узла.

Если у JCP нет активных задач, связанных с определенным узлом, контроль активности этого узла не проводится.

5.7.3 TASK_STATE

Инструкция "Состояние задачи" (TASK_STATE) передается от определенной задачи к JCP. Она служит для подтверждения инструкции STATE_REQ. Инструкция имеет следующие значения полей:

```
OPCODE = 22
PCK = %b00
CHN = 0
ASK = 0
EXT = 0
OPR_LENGTH = 1/2/3 ; Зависит от длины CTID.
Операнды:
   1 октет: код состояния задачи. Определены следующие значения
            этого поля:
      %x01 - задача активна и имеет активные сеансовые
             соединения
      %x02 - задача активна и не имеет сеансовых соединений
      %x03 - задача активна, не имеет сеансовых соединений и не
             имеет выделенных на узле ресурсов
      %x04 - задача завершена
   1/3 октета: Зарезервировано. Если OPR_LENGTH = 1, то поле имеет
               длину 1 октет, иначе 3 октета. JCP не должен
               проверять значение этого поля. При передаче оно
               должно устанавливаться в 0.
   2/4/8 октета: CTID связанный с LTID из инструкции STATE_REQ.
```

Если OPR_LENGTH = 1 то длина зарезервированного поля равна одному октету и длина CTID составляет два октета. Во всех остальных

случаях длина зарезервированного поля – 3 октета и длина CTID – не менее 4 октет.

5.7.4 RELOAD_NODE

Инструкция «Узел перезапустился» (NODE_RELOAD) посылается к JCP в качестве отрицательного ответа на инструкцию STATE_REQ. NODE_RELOAD имеет следующие значения полей:

```
OPCODE = 23
PCK = %b00
CHN = 0
ASK = 0
EXT = 0
OPR_LENGTH = 1/2 ; Зависит от длины LTID.
Операнд:
   4/8 octets: LTID. Значение берется из инструкции STATE_REQ.
```

Инструкция RELOAD_NODE означает, что задача с заданным LTID для данного JCP на узле отсутствует. При получении этой инструкции JCP должен сделать следующее:

(1) Послать инструкции STATE_REQ всем задачам узла, которые были инициированы перед посылкой предпоследней инструкции STATE_REQ.
(2) Подождать окончание одного интервала бездействия после посылки последней инструкции STATE_REQ (на которую был получен отрицательный ответ).
(3) Послать инструкции STATE_REQ всем задачам узла, которые были инициированы между последней и предпоследней инструкциями STATE_REQ (не считая инструкции из пункта 1).

На все инструкции STATE_REQ должен быть передан положительный ответ (TASK_STATE) или отрицательный ответ (RELOAD_NODE).

5.8 Работа без сеансового соединения

Протокол предусматривает обмен данными между узлами без установления сеансового соединения. В этом случае инициализация задания и задач не производиться и JCP не используется.

Формат инструкции, передаваемые без установления соединения, полностью соответствует инструкциям, передаваемым по сеансовым соединениям. Отличие заключено в том, что поле SESSION_ID имеет нулевое значение или PCK = %b00.

Узел, поддерживающий работу без установления сеансового соединения, должен иметь VM, которая по умолчанию выполняет инструкции, передаваемые без установления соединения. Фактически эти инструкции выполняются в рамках так называемой 0–задачи (или 0-сессии) этой VM. Адресное пространство памяти этой VM доступно без установления соединения.

Инструкция SESSION_INIT с SESSION_ID = 0 и REQ_ID = 0 позволяет указать параметры своей 0-задачи и запросить параметры 0-задачи узла получателя. Если узел, получивший такую инструкцию, поддерживает запрашиваемый профиль, он посылает инструкцию SESSION_ACCEPT с нулевым кодом возврата. Если профиль не поддерживается, посылается ответная инструкция SESSION_INIT, в которой поле SESSION_ID и REQ_ID также имеют значение 0. Фактически такие инструкции инициализации сессии не устанавливают соединение, а носят информационный характер. Обмен данными по 0-сессии может происходить независимо от них.

При работе без соединения существуют следующие ограничения:

- цепочка может передаваться, только если она полностью помещается в один сегмент транспортного уровня
- Нельзя запрашивать выделение памяти и создавать объекты. Это объекты не будут привязаны к определенному заданию и не будут автоматически освобождать ресурсы при завершении задания, которое их создало.
- Параметры функций и возвращаемые значения не должны содержать указатели. Это связано с тем, что узел в любой момент может перезагрузиться. Это приведет к тому, что указатели станут недействительны или будут адресовать другие объекты.

Протокол не может проконтролировать указанные условия. Их реализация лежит целиком на VM.

Работа без установления сеансового соединения может использоваться в следующих системах:

- в устройствах, не имеющих операционной системы;
- на серверах, обслуживающих большое количество запросов (при работе без установления соединения меньше потребление ресурсов);
- В системах, требующих быстрого ответа на редкие запросы (если удержание соединения является нецелесообразным).

6 Инструкции обмена между VM

Инструкции, предназначенные для обмена между VM, имеют OPCODE в диапазоне 128 - 253. В зависимости от длины поля операндов, для одного OPCODE может быть задано несколько форматов инструкции. Полный формат инструкции определяется совокупностью значений поля OPCODE и OPR_LENGTH.

Если в заголовке инструкции флаг ASK установлен в 1, инструкция имеет поле REQ_ID, используемое для идентификации ответа. Значение этого поля устанавливает VM. Ответ формируется так же VM. Для инструкций обмена между VM протокол не контролирует ответы и не анализирует значение поля REQ_ID. Для передачи ответа используется одна из инструкций RSP, DATA, RETURN, ADDRESS, OBJECT или PROC_NUM. Инструкции подтверждения имеют ASK = 1 и в REQ_ID установлено значение, взятое из подтверждаемой инструкции. Инструкции подтверждения не требуют подтверждения.

Инструкции обмена между VM можно передавать по UDP при соблюдении всех следующих условий:

- ASK = 0;
- инструкция помещается в один сегмент UDP;

Для инструкций обмена между VM тайм-ауты и повторные передачи на уровне UMSP не используются. Считается, что время передачи инструкций с низким приоритетом из-за наличия выходных очередей может быть очень большим. Поэтому принимать решение о тайм-ауте должна VM, так как только VM имеет полную информацию о характере передаваемых данных. Кроме этого тайм-ауты должен использовать протокол транспортного уровня.

К протоколу на узле может быть подключено несколько VM. VM может одновременно выполнять несколько заданий. Каждое задание может работать в своем адресном пространстве. Протокол определяет VM и задание, которому следует передать поступившую инструкцию, на основании значения поля SESSION_ID.

Адрес локальной памяти записывается в инструкции в поле длиной 2/4/8 октет. Если длина адреса памяти в инструкции не равна длине

адреса памяти, установленной для узла, возможны следующие варианты:

- если для узла установлена длина адреса памяти 24 бита, адрес записывается в конец 4 – октетного поля. В начальный (нулевой) октет при этом должно записываться значение 0.
- Если формат инструкции предполагает длину адреса памяти не меньше 4 – октет, 2 – октетный адрес записывается в последние октеты. Первые 2 октета должны иметь нулевое значение.
- если инструкция относится к цепочке и имеет длину адреса памяти меньше, чем установлено для узла – считается, что используется базовая адресация. Если для цепочки не установлено значение базы памяти – инструкция является ошибочной.
- если инструкция не относится к цепочке и имеет длину адреса памяти меньше, чем установлено для узла – считается, что использован укороченный адрес. Полная длина адреса может быть получена путем присоединение спереди необходимого числа нулевых октет.
- во всех остальных случаях инструкцию следует считать ошибочной.

Полный 128 – битный адрес памяти записывается в операндах в 16 – октетное поле. Необходимость использования полного адреса связана с тем, что в его поле FREE (см. секцию 2.1) может содержаться служебная информация, используемая подсистемой управления памяти на узле, к которому относится адрес. Если свободные поля полного адреса установлены в 0, в операндах рекомендуется использовать локальный адрес.

Длина поля операндов инструкции должна быть кратна 4 октетам. При необходимости, выравнивание производят добавлением нулевых октетов в конце за всеми значащими операндами, если иное не определено в описании формата инструкции.

Поля заголовка инструкций, не приводимые в описании форматов, используются в соответствии с описанием из секции 3.

Инструкция передачи управления JUMP, CALL, CALL_BNUM и CALL_BNAME могут содержать информацию о VM отправителя. Если тип и версия VM отправителя включена в инструкцию, параметры вызова формируются в формате VM отправителя. Иначе параметры вызова имеют формат, определяемый VM получателя. Код всегда связан с определенной VM.

Все инструкции протокола работают с двоичными данными и не поддерживают операций преобразования форматов.

6.1 Инструкции чтения/записи данных

6.1.1 REQ_DATA

Инструкция «Запросить данные» (REQ_DATA) используется для запроса данных с удаленного узла. Определены две инструкции с длиной поля длины 2 и 4 октета. Эти инструкции имеют следующее значение полей:

```
OPCODE = 130/131      ; для длины поля длины 2/4 октета
OPR_LENGTH = 1/2/3/5  ; Зависит от длины адреса.
Операнды:
   2/4 октета: Поле длины. Длина запрашиваемых данных в октетах
   2/4/8/16 октета: Адрес памяти запрашиваемых данных
```

В ответ на инструкцию REQ_DATA передается инструкция DATA, содержащая запрашиваемые данные. Если данные не могут быть переданы, возвращается инструкция RSP с ненулевым основным кода возврата.

6.1.2 DATA

Инструкция «Данные» (DATA) передается в ответ на инструкции REQ_DATA и OBJ_REQ_DATA. Эта инструкция имеет следующие значения полей:

```
OPCODE = 132
OPR_LENGTH = 0 - 65535  ; Зависит от длины данных операнда.
Операнды:
   4 - 262140 октет: Непосредственные данные. Если OPR_LENGTH =
                     0 то поле отсутствует.
Расширенные заголовки:
   _DATA - Содержит непосредственные данные. Этот заголовок
           присутствует, только если OPR_LENGTH = 0.
```

Расширенный заголовок используется, если данные не помещаются в поле операндов. Данные не должны одновременно передаваться в операндах и в расширенном заголовке. Если запрошенная длина данных не кратна 4 октетам, в конец поля данных для выравнивания границы записываются 1 – 3 нулевых октет.

6.1.3 WRITE

Инструкция «Записать данные» (WRITE) используется для записи данных на удаленном узле. Эта инструкция имеет следующие значения полей:

```
OPCODE = 133/134/135/136  ; для длины адреса 2/4/8/16 октет.
OPR_LENGTH = 4 - 65535    ; Зависит от длины данных
Операнды:
   2/4/8/16 октет: Адрес памяти для записи данных.
   0 - 262138 октет: Непосредственные данные для записи.
Расширенные заголовки:
   _DATA - Содержит непосредственные данные. Этот заголовок
           присутствует, только если данных нет в операндах.
```

При длине адреса 2 октета длина данных должна быть 2 октета. Во всех остальных случаях длина адреса должна быть не меньше 4 октет и длина данных кратна четырем. Данные не должны одновременно передаваться в операндах и в расширенном заголовке.

В ответ на инструкцию WRITE передается инструкция RSP. Нулевой основной код возврата определяет нормальное завершение.

6.1.4 WRITE_EXT

Инструкция «Расширенная запись данных» (WRITE_EXT) используется для записи данных на удаленном узле. Длина данных может быть 1 – 262132 октета с шагом 1 октет. Эта инструкция имеет следующие значения полей:

```
OPCODE = 137
   OPR_LENGTH = 3 - 65535  ; Зависит от длины данных и длины
                             адреса
Операнды:
   1 октет: Всегда имеет значение 0.
   3 октета: Длина записываемых данных в октетах. Значение 0
             недопустимо.
   4 - 262132 октет: Непосредственные данные для записи. Длина
                     данных кратна 4 октетам.
   4/8/16 октета: Адрес памяти для записи данных.
```

Если длина записываемых данных не кратна четырем октетам, в конец поля данных для выравнивания границы записываются 1 – 3 нулевых октета.

В ответ на инструкцию WRITE_EXT передается инструкция RSP. Нулевой основной код возврата определяет нормальное завершение.

6.2 Инструкции сравнения

6.2.1 CMP

Инструкция «Сравнение» (CMP) используется для двоичного сравнения данных. Эта инструкция имеет следующие значения полей:

```
OPCODE = 138/139/140/141  ; для длины адреса 2/4/8/16 октет
OPR_LENGTH = 1 - 65535    ; Зависит от длины данных
Операнды:
   2/4/8/16 октет: адрес памяти сравниваемых данных.
   2 - 262138 октет: непосредственные данные для сравнения.
```

При длине адреса 2 октета длина данных должна быть 2 октета. Во всех остальных случаях длина адреса должна быть не меньше 4 октет и длина данных кратна четырем октетам.

6.2.2 CMP_EXT

Инструкция «Расширенное сравнение» (CMP_EXT) используется для двоичного сравнения данных. Длина данных может быть 1 - 262132 октета с шагом 1 октет. Эта инструкция имеет следующие значения полей:

```
OPCODE = 142
OPR_LENGTH = 3 - 65535  ; Зависит от длины данных и длины
                          адреса
Операнды:
   1 октет: всегда имеет значение 0.
   3 октета: Длина сравниваемых данных в октетах. Значение 0
             недопустимо.
   4 - 262132 октет: непосредственные данные для сравнения.
                     Длина поля кратна 4 октетам.
   4/8/16 октета: адрес памяти сравниваемых данных.
```

Для выравнивания адреса памяти на границу четырех октет в конец непосредственных данных записываются 1 - 3 нулевых октет.

6.2.3 Ответ на инструкции сравнения

В ответ на инструкции CMP, CMP_EXT и OBJ_CMP (смотри ниже) передается инструкция RSP. Если сравнение было выполнено, основной код возврата равен нулю. Дополнительный код возврата равен –1, если данные по адресу памяти меньше данных из операнда, 0, если они равны и 1, если они больше. Если сравнение не может быть выполнено, основной код возврата инструкции RSP должен быть ненулевым.

6.3 Инструкции передачи управления

6.3.1 JUMP, CALL

Инструкции «Безусловный переход» (JUMP) и «Вызов подпрограммы» (CALL) имеют одинаковый формат и отличаются только OPCODE. Эти инструкции имеют следующие значения полей:

```
OPCODE = 143/144/ ; Соответственно для JUMP не включающей и
                    включающей информацию о VM.
         145/146  ; Соответственно для CALL не включающей и
                    включающей информацию о VM.
```

```
OPR_LENGTH = 2 - 65535 ; Зависит от включения информации о VM,
                         длины адреса и длины параметров.
Операнды:
   2 октета: Тип VM отправителя. Если OPCODE=143/145 это поле
             отсутствует.
   2 октета: Версия VM отправителя. Если OPCODE=143/145 это
             поле отсутствует.
   2 октета: Число 32 – битных слов в поле параметров вызова.
   4 - 262136 октета: непосредственные данные являющиеся
                      параметрами вызова.
   4/8/16 октета: адрес памяти, куда следует передавать
                  управление.
```

На приемной стоне обработка инструкций передачи управления происходит следующим образом:

- Проверяется адрес перехода. Если он имеет ошибочное значение, передается отрицательное подтверждение RSP. На этом этапе может быть так же проверена корректность параметров вызова.
- Если адрес перехода и параметры вызова имеют корректное значение, для инструкции JUMP передается положительное подтверждение RSP. После приема подтверждения передающая сторона считает инструкцию JUMP выполненной.
- Для подтверждения выполнения инструкции CALL передается инструкция RETURN. Инструкция RETURN может содержать возвращаемые значения. Если в потоке управления, созданном по инструкции CALL, возникает исключительная ситуация, вместо RETURN передается инструкция RSP с ненулевым основным кодом возврата.

6.3.2 RETURN

Инструкция «Возврат управления» RETURN используется при возврате управления из инструкций CALL, CALL_BNUM и CALL_BNAME (см. ниже). Эта инструкция имеет следующие значения полей:

```
OPCODE = 147
OPR_LENGTH = 0 - 65535  ; Зависит от длины непосредственных
                          данных операнда.
Операнды:
   0 – 262140 октет: непосредственные данные, возвращаемые из
                     подпрограммы.
```

Инструкция RETURN используется при возврате управления из инструкций CALL и RUN. Если не требуется передавать возвращаемое значение, инструкция RETURN не содержит операндов. Формат возвращаемых данных совпадает с инструкцией, на которую передается ответ (формат VM отправителя или получателя).

6.4 Инструкции управления памятью

UMSP предоставляет средства для разделения памяти для кода и для данных. Протокол не производит проверок корректности операций с памятью. Код и данные используют общее адресное пространство. Управление памятью полностью осуществляется VM.

6.4.1 MEM_ALLOC

Инструкция «Запросить память для данных» (MEM_ALLOC) используется для запроса выделения памяти под данные. Эта инструкция имеет следующие значения полей:

```
OPCODE = 148
OPR_LENGTH = 1
```

```
Операнды:
   4 октета: размер выделяемой памяти в байтах.
```

Для положительного ответа на инструкцию MEM_ALLOC передается инструкция ADDRESS, для отрицательного - RSP с ненулевым основным кодом возврата. Полученный адрес может быть использован протоколом в инструкциях чтения/записи, сравнения и синхронизации.

6.4.2 MVCODE

Инструкция «Перенести код» используется для переноса исполняемого кода с одного узла на другой. Эта инструкция имеет следующие значения полей:

```
OPCODE = 149
OPR_LENGTH = 1 - 65535  ; Зависит от длины поля кода
Операнды:
   2 октета: тип VM получателя
   2 октета: версия VM получателя
   0 - 262136 октет: содержит исполняемый код
Расширенные заголовки:
   _DATA - содержит исполняемый код. Этот заголовок
           присутствует только, если код не содержится в
           операндах.
```

Код всегда связан с VM определенного типа. Поле кода прозрачно для протокола. Оно формируется VM отправителя и должно содержать всю информацию, необходимую VM получателя. Код не должен одновременно передаваться в операндах и в расширенном заголовке.

Для положительного ответа на инструкции MVCODE используется инструкция ADDRESS, для отрицательного - RSP с ненулевым основным кодом возврата. Код, перенесенный по инструкции MVCODE, может быть выполнен по инструкции JUMP или CALL.

6.4.3 ADDRESS

Инструкция «Адрес памяти» (ADDRESS) используется для положительного ответа на инструкции MEM_ALLOC и MVCODE. ADDRESS имеет следующие значения полей:

```
OPCODE = 150
OPR_LENGTH = 1/2/4  ; Зависит от длины адреса
Операнды:
   4/8/16 октет: адрес выделенной памяти.
```

Для инструкции MEM_ALLOC адрес указывает на первый байт выделенной области данных. Для инструкции MVCODE содержание адреса определяется VM, с которой связан код.

6.4.4 FREE

Память, выделенная инструкциями MEM_ALLOC и MVCODE, должна быть явно освобождена. Для этого используется инструкция «Освободить память» (FREE). Она имеет следующие значения полей:

```
OPCODE = 151
OPR_LENGTH = 1/2/4 ; Зависит от длины адреса.
Операнды:
   4/8/16 октета: адрес освобождаемой памяти.
```

VM должна освободить память автоматически при завершении задачи на узле.

6.4.5 MVRUN

Инструкция «Перенести и выполнить» (MVRUN) используется для одновременного переноса кода и его исполнения. Инструкция имеет следующий формат:

```
OPCODE = 152
OPR_LENGTH = 1 - 65535  ; Зависит от длины поля кода
Операнды:
   2 октета: Тип VM получателя.
   2 октета: Версия VM получателя.
   0 - 262136 октет: содержит исполняемый код.
Расширенные заголовки:
   _DATA - Содержит исполняемый код. Этот заголовок
           присутствует только, если код не содержится в
           операндах.
```

С точки зрения протокола исполняемый код это просто буфер с двоичными данными. Формат этого поля определяется VM и должен содержать всю информацию, необходимую для загрузчика VM получателя, включая параметры вызова.

Код не должен одновременно передаваться в операндах и в расширенном заголовке.

Ответ на инструкцию MVRUN формируется аналогично инструкции CALL. Память, выделенную для кода по этой инструкции, не нужно освобождать. За освобождение памяти отвечает VM.

6.5 Другие инструкции

6.5.1 SYN

Инструкция «Синхронизировать» (SYN) используется для однократного сообщения об изменении данных. Она имеет следующий формат:

```
OPCODE = 153/154/155    ; для длины адреса 4/8/16 октет.
OPR_LENGTH = 2 - 65535  ; зависит от длины данных
Операнды:
   4/8/16 октета: адрес памяти проверяемых данных.
   2 - 131068 октет: Непосредственные исходные данные. Длина
                     данных должна быть кратна двум октетам.
   2 - 131068 октет: Непосредственная маска для сравнения.
                     Длина этого поля равна длине поля исходных
                     данных.
```

Отслеживаемые данные заданы адресом памяти в первом операнде. Эти данные первоначально сравниваются со значением исходных данных из второго операнда. Если значения не совпадают, считается, что данные изменились. Третий операнд позволяет задать маску для сравнения. Установленные в единицу биты в маске указывают на биты в данных, изменение которых следует отслеживать.

На инструкцию возможны следующие варианты ответа:

- Если адрес локальной памяти некорректен, в ответ передается инструкция RSP с ненулевым основным кодом возврата.
- Если данные не изменяются, в ответ ничего не передается.
- Если данные изменились, передается инструкция DATA с новым значением отслеживаемых данных.

6.5.2 NOP

Инструкция «Пустая операция» (NOP) имеет следующий формат:

```
OPCODE = 156
OPR_LENGTH = 0 - 65535
Операнды:
   0 - 262140 октет: инкапсулированные данные.
Расширенные заголовки:
   Любые расширенные заголовки.
```

Инструкция NOP предназначена для решения следующих задач:

- Передача управляющих расширенных заголовков, когда в сеансе нет других инструкций для передачи.
- Инкапсуляция фрагментированных инструкций и транзакций с установленным флагом специальной обработки (см. секцию 7).

6.6 Работа с объектами

Протокол имеет набор инструкций, являющихся расширением протокола RPC [6]. В отличие от RPC UMSP позволяет напрямую адресовать память на удаленных узлах и передавать ссылки в качестве параметров и возвращаемых значений.

Объект UMSP идентифицируется по 4-октетному номеру. Значения разделены на следующие диапазоны:

```
   I -> %x00000000 - 1FFFFFFF   назначены для стандартных объектов
  II -> %x20000000 - 3FFFFFFF   назначены для пользователей
 III -> %x30000000 - 4FFFFFFF   свободные
  IV -> %x50000000 - DFFFFFFF   временные
   V -> %xE0000000 - FFFFFFFF   зарезервированы
```

Стандартные объекты из диапазона I должны быть зарегистрированы и спецификации их интерфейсов должны быть опубликованы. Протокол не допускает частных или не описанных интерфейсов стандартных объектов.

Объекты из диапазона II должны быть зарегистрированы, но спецификации их интерфейсов могут быть не опубликованы. Эти номера применяются в случаях, когда требуется исключить возможный конфликт систем разных производителей.

Диапазон III может использоваться свободно. Объекты, доступные по этим номерам, могут создаваться статически или динамически. Эти объекты могут иметь любые интерфейсы.

Все объекты, относящиеся к диапазонам I, II и III, общие для всех заданий на узле, включая 0 - задание. Их интерфейсы доступны всем задачам на узле, возможно с учетом параметров идентификации.

Диапазон IV предназначен для объектов, создаваемых динамически в рамках одного задания. Это объекты являются изолированной ассоциативной памятью задания. Доступ к этим объектам может быть получен только из кода задания, которое их создало. 0 - задание не имеет доступа к объектам из диапазона IV.

Протокол предоставляет доступ к данным объекта, как к непрерывному сегменту памяти. Память объектов может пересекаться или не пересекаться с плоской локальной памятью узла. В инструкциях работы с данными объекта используется поле смещения от начала объекта в байтах. Правила записи смещения аналогичны правилам записи локального адреса, определенные вначале данной секции.

Разрядность памяти узла, определенная для протокола UMSP, ограничивает максимальный размер данных одного объекта. Инструкции, определенные в данном пункте, позволяют работать с

ассоциативной памятью с теоретическим предельным размером на один узел – 2^96 (7,9 * 10^28).

Кроме номера объект имеет версию, длиной 2 октета, и реализацию, длиной 2 октета. Протокол требует обязательной совместимости снизу вверх для всех реализаций одной версии объекта. Публикация новой реализации стандартного объекта может содержать только добавляемые интерфейсы.

Если для отправителя инструкции версия и/или реализация объекта не играет роли или неизвестна, то инструкция может содержать нулевые поля версии и реализации объекта или только нулевое поле реализации. Нулевое поле версии и ненулевое поле реализации не допускается.

6.6.1 Чтение/запись данных объекта

6.6.1.1 OBJ_REQ_DATA

Инструкция «Запросить данные объекта» (OBJ_REQ_DATA) используется для запроса данных с удаленного узла. Она имеет следующий формат:

```
OPCODE = 192/193       ; Для длины поля длины 2/4 октета
OPR_LENGTH = 3/4/5     ; Зависит от длины поля смещения
Операнды:
   4 октета: Номер объекта.
   2 октета: Версия объекта.
   2 октета: Реализация объекта.
   2/4 октета: Длина запрашиваемых данных в октетах.
   2/4/8 октета: Смещение запрашиваемых данных от начала
                 объекта в байтах.
```

При длине поля длины в 2 октета длина смещения должна быть два октета. Во всех остальных случаях длина поля длины и смещения должны быть не меньше 4 октет.

В ответ на инструкцию OBJ_REQ_DATA передается инструкция DATA, содержащая запрашиваемые данные. Если данные не могут быть переданы, возвращается инструкция RSP с ненулевым основным кода возврата.

6.6.1.2 OBJ_WRITE

Инструкция «Записать данные в объект» (OBJ_WRITE) используется для записи данных в объект. Инструкция имеет следующий формат:

```
OPCODE = 194/195/196    ; Для длины поля смешения 2/4/8 октет.
OPR_LENGTH = 3 – 65535  ; Зависит от длины данных.
Операнды:
   4 октета: номер объекта
   2 октета: версия объекта
   2 октета: реализация объекта
   2/4/8 октетов: смещение от начала объекта для записи данных.
   0 - 262130 октет: непосредственные данные для записи.
Расширенные заголовки:
   _DATA – содержит непосредственные данные для записи. Этот
           заголовок присутствует, только если данных нет в
           операндах.
```

При длине поля смещения 2 октета длина данных должна быть 2 октета. Во всех остальных случаях длина смещения должна быть не меньше 4 октет и длина данных кратна четырем. Данные не должны одновременно передаваться в операндах и в расширенном заголовке.

6.6.1.3 OBJ_WRITE_EXT

Инструкция «Расширенная запись данных в объект» (OBJ_WRITE_EXT) используется для записи данных в объект. Длина данных может быть 1 – 262132 октета с шагом 1 октет. Инструкция имеет следующие значения полей:

```
OPCODE = 197
OPR_LENGTH = 3 – 65535  ; Зависит от длины данных и длины адреса
Операнды:
   4 октета: номер объекта
   2 октета: версия объекта
   2 октета: реализация объекта
   1 октет: всегда имеет значение 0.
   3 октета: Длина записываемых данных в октетах. Значение 0 недопустимо.
   4 – 262128 октет: непосредственные данные для записи. Длина данных кратна 4 октетам.
   2/4/8 октета: смещение от начала объекта для записи данных.
```

Если длина записываемых данных не кратна четырем октетам, в конец поля данных для выравнивания границы записываются 1 – 3 нулевых октет.

В ответ на эту инструкции OBJ_WRITE, OBJ_WRITE_EXT передается инструкция RSP. Нулевой основной код возврата определяет нормальное завершение.

6.6.2 Инструкции сравнения данных объекта

6.6.2.1 OBJ_DATA_CMP

Инструкция «Сравнить данные объекта» (OBJ_DATA_CMP) используется для двоичного сравнения данных объекта с непосредственными данными из операндов. Инструкция имеет следующие значения полей:

```
OPCODE = 198/199/200     ; для длины поля смешения 2/4/8 октет
OPR_LENGTH = 3 – 65535   ; Зависит от длины данных
Операнды:
   4 октета: Номер объекта.
   2 октета: Версия объекта.
   2 октета: Реализация объекта.
   2/4/8 октета: Смещение от начала объекта для сравниваемых
                 данных.
   2 - 262128 октет: Непосредственные данные для сравнения.
```

При длине поля смещения 2 октета длина данных должна быть 2 октета. Во всех остальных случаях длина смещения должна быть не меньше 4 октет и длина данных кратна четырем.

Ответ на инструкцию OBJ_DATA_CMP описан в секции 6.2.3.

6.6.2.2 OBJ_DATA_CMP_EXT

Инструкция «Расширенное сравнение данных объекта» (OBJ_DATA_CMP_EXT) используется для двоичного сравнения данных объекта с непосредственными данными из операндов. Длина данных может быть 1 – 262132 октета с шагом 1 октет. Эта инструкция имеет следующие значения полей:

```
OPCODE = 201
OPR_LENGTH = 5 – 65535  ; Зависит от длины данных и длины
                          адреса
Операнды:
```

4 октета: Номер объекта.
2 октета: Версия объекта.
2 октета: Реализация объекта.
1 октет: всегда имеет значение 0.
3 октета: Длина сравниваемых данных в октетах. Значение 0 недопустимо.
4 - 262132 октет: непосредственные данные для сравнения. Длина поля кратна 4 октетам.
4/8 октета: Смещение от начала объекта для сравниваемых данных.

Для выравнивания смещения на границу четырех октет в конец непосредственных данных записываются 1 - 3 нулевых октет.

Ответ на инструкцию OBJ_DATA_CMP_EXT описан в секции 6.2.3.

6.6.3 Вызов процедур объекта

6.6.3.1 CALL_BNUM

Инструкция «Выполнить процедуру объекта по номеру» (CALL_BNUM) передает управление процедуре объекта по ее номеру. Эта инструкция имеет следующие значения полей:

OPCODE = 202/203 ; Соответственно для инструкций не включающей и включающей информацию о VM.
OPR_LENGTH = 4 - 65535 ; Зависит от включения информации о VM и длины параметров.
Операнды:
2 октета: Тип VM отправителя. Если OPCODE=202 это поле отсутствует.
2 октета: Версия VM отправителя. Если OPCODE=202 это поле отсутствует.
4 октета: номер объекта.
2 октета: версия объекта.
2 октета: реализация объекта.
4 октета: номер вызываемой процедуры
4 - 262128 октет: параметры вызова.

Обработка на приемной стороне производится аналогично инструкции CALL (см. секцию 6.3.1).

6.6.3.2 CALL_BNAME

Инструкция «Выполнить процедуру объекта по имени» (CALL_BNAME) передает управление процедуре объекта по ее имени. Эта инструкция имеет следующие значения полей:

OPCODE = 204/205 ; Соответственно для инструкций не включающей и включающей информацию о VM.
OPR_LENGTH = 3 - 65535 ; Зависит от включения информации о VM и длины параметров.
Операнды:
2 октета: Тип VM отправителя. Если OPCODE=204 это поле отсутствует.
2 октета: Версия VM отправителя. Если OPCODE=204 это поле отсутствует.
4 октета: номер объекта.
2 октета: версия объекта.
2 октета: реализация объекта.
4 - 262128 октет: параметры вызова.
Расширенный заголовок:
_NAME - содержит имя вызываемой процедуры.

Обработка на приемной стороне производится аналогично инструкции CALL (см. секцию 6.3.1).

Имена могут иметь процедуры объектов из диапазонов III и IV. Процедуры объектов из диапазонов I и II не должны иметь имен на уровне UMSP. Они должны иметь только номер.

6.6.3.3 GET_NUM_PROC

Инструкция «Получить номер процедуры» (GET_NUM_PROC) позволяет получить номер процедуры объекта из диапазона III и IV по имени процедуры. Эта инструкция имеет следующие значения полей:

```
OPCODE = 206
OPR_LENGTH = 2
Операнды:
   4 октета: номер объекта.
   2 октета: версия объекта.
   2 октета: реализация объекта.
Расширенный заголовок:
   _NAME - содержит имя процедуры.
```

Для положительного ответа на инструкцию GET_NUM_PROC передается инструкция PROC_NUM, для отрицательного – RSP с ненулевым основным кодом возврата.

6.6.3.4 PROC_NUM

Инструкция «Номер процедуры» (PROC_NUM) передается в ответ на инструкцию GET_NUM_PROC. Инструкция PROC_NUM имеет следующие значения полей:

```
OPCODE = 207
OPR_LENGTH = 3
Operands:
   4 октета: номер объекта.
   2 октета: версия объекта.
   2 октета: реализация объекта.
   4 октета: номер процедуры.
```

6.6.4 Создание объектов

Объекты из диапазонов I и II (стандартные и выделенные для пользователя) не могут быть созданы на удаленном узле через интерфейс UMSP. Эти объекты могут быть созданы только через API VM. Объекты из диапазона III и IV могут быть созданы на удаленном узле при помощи инструкций протокола.

Реализация объектов из диапазона I – II (не связанных с определенным заданием) и III (созданных определенным заданием, но доступных из любого задания) достаточно сложна. Это связано с тем, что разные задания могут иметь разные адресные пространства памяти. Указатели должны обрабатываться в контексте задания, от которого они получены. Кроме этого общие объекты должны отслеживать завершение заданий для освобождения динамических ресурсов. При этом указатели на объекты из III диапазона могут находиться на узлах, на которых задание не выполняется и протокол не предоставляет средств для оповещения этих узлов о завершении задания. Указанные требования накладывают существенные ограничения на эти объекты. На объекты из IV диапазона протокол не накладывает никаких ограничений.

Уникальным ключом, идентифицирующим объект на узле, является номер объекта. Объектам из диапазона III и IV может быть присвоено имя. Объекты из диапазона I и II не могут иметь имен на уровне UMSP. В

рамках одной задачи не может быть двух объектов, имеющих один номер или одно имя.

6.6.4.1 NEW, SYS_NEW

Формат обеих инструкций «Создать объект» (NEW) и «Создать системный объект» (SYS_NEW) аналогичен. Первая создает объект в диапазоне IV, вторая - в диапазоне III. Эти инструкции имеют следующий формат:

```
OPCODE = 208/209  ; соответственно для NEW/SYS_NEW
OPR_LENGTH = 3
Операнды:
   2 октета: Тип VM.
   2 октета: Версия VM.
   2 октета: версия объекта
   2 октета: реализация объекта
   4 - 262136 октет: данные, необходимые для создания объекта.
Расширенные заголовки:
   _DATA - содержит данные, необходимые для создания объекта.
           Этот заголовок присутствует, только если данных нет
           в операндах.
   _NAME - содержит имя объекта. Этот заголовок является
           необязательным.
```

Инструкция SYS_NEW используется для создания объекта, доступного из любого задания, NEW - для создания объекта, доступного только из «своего» задания. Если объект создан, для подтверждения передается инструкция OBJECT. Если объект не удалось создать, передается инструкция RSP с ненулевым основным кодом возврата.

Поле данных прозрачно для протокола. Оно формируется VM отправителя и должно содержать всю информацию, необходимую VM получателя для создания объекта. Данные не должен одновременно передаваться в операндах и в расширенном заголовке.

Поле SESSION_ID инструкции не может иметь нулевое значение. Динамический объект может быть создан только в контексте определенного задания. Объект всегда создается на VM, с которой связан сеанс.

Нулевые значения версии и реализации объекта говорят о том, что объект их не имеет.

Одновременно с созданием объекта можно зарегистрировать его имя. Имя содержится в расширенном заголовке _NAME.

Все объекты, созданные по инструкциям NEW и SYS_NEW должны быть явно удалены. При завершении задания VM должна автоматически удалять все динамические объекты, созданные и не удаленные заданием.

6.6.4.2 OBJECT

Инструкция «Объект» (OBJECT) используется для положительного ответа на инструкции NEW и SYS_NEW. Инструкция OBJECT имеет следующие значения полей:

```
OPCODE = 210
OPR_LENGTH = 2
Операнды:
   4 октета: номер объекта
   2 октета: версия объекта
   2 октета: реализация объекта
```

6.6.4.3 DELETE

Инструкция «Удалить объект» (DELETE) используется для удаления объекта, созданного по инструкции NEW или SYS_NEW. Она имеет следующие значения полей:

```
OPCODE = 211
OPR_LENGTH = 1
Операнды:
   4 октета: номер объекта
```

Объект может быть удален только из задания, которое его создало. В ответ на эту инструкцию передается инструкция RSP.

6.6.5 Идентификация объектов

При регистрации объекта на узле, он может быть связан с именем, длиной 4 - 254 октет. Имя состоит из символов ASCII. Следующие версии протокола могут определить другие типы имени.

Имя связывается с номером объекта и является его синонимом. Имена всех активных объектов в одной задачи на узле должны быть уникальны. При этом все активные объекты из диапазона номеров III должны иметь уникальные имена для всех задач на узле. Протокол позволяет получить номер объекта по имени и имя объекта по номеру.

6.6.5.1 OBJ_SEEK

Инструкция «Найти объект» OBJ_SEEK используется для получения номера объекта по имени. Она имеет следующие значения полей:

```
OPCODE = 212
OPR_LENGTH = 0
Обязательный расширенный заголовок:
   _NAME - содержит имя объекта.
```

Если объект найден - в ответ передается инструкция OBJECT. Если объект не найден - в ответ передается инструкция RSP с ненулевым основным кодом возврата.

Инструкция OBJ_SEEK может передаваться широковещательно через UDP. В этом случае она относится к 0-сессии и должна содержать поле REQ_ID для идентификации ответов. В ответ на широковещательную рассылку OBJ_SEEK должны передаваться только положительные ответы. Ответ может быть направлен через UDP.

6.6.5.2 OBJ_GET_NAME

Инструкция «Получить имя объекта» (OBJ_GET_NAME) используется для получения имени объекта по номеру. Она имеет следующие значения полей:

```
OPCODE = 213
OPR_LENGTH = 1
Операнды:
   4 октета: номер объекта
```

Если объект существует - в ответ передается инструкция OBJECT с присоединенным расширенным заголовком _NAME. Если объект не найден - в ответ передается инструкция RSP с ненулевым основным кодом возврата.

7 Цепочки

Инструкции, передаваемые по одному сеансовому соединению, могут быть объединены в цепочку. Цепочка - это группа связанных друг с другом инструкций. В одном сеансе одновременно может передаваться несколько цепочек. Цепочки бывают следующих типов:

- последовательность
- транзакция
- фрагментированная инструкция.

Если инструкция включена в цепочку, флаг CHN должен быть равен 1. Поле CHAIN_NUMBER заголовка содержит номер цепочки, INSTR_NUMBER – порядковый номер инструкции в цепочке начиная с 0. Нумерацию цепочек ведет протокол. Всего в одном сеансе одновременно может передаваться до 65533 цепочки. Значения номеров цепочек %x0000 и %xFFFF зарезервированы протоколом. Одна цепочка может содержать до 65535 инструкций.

Инструкция с нулевым порядковым номером INSTR_NUMBER должна содержать расширенный заголовок, описывающий цепочку. Каждый тип цепочки имеет свой инициирующий расширенный заголовок.

Расширенный заголовок “Конец цепочки” (_END_CHAIN) передается в последней инструкции цепочки, независимо от ее типа. Он имеет следующие значения полей:

```
HEAD_CODE = 6
HEAD_LENGTH = 0
HOB = 1
```

Номер завершаемой цепочки содержится в поле CHAIN_NUMBER заголовка инструкции, к которой присоединен заголовок.

Инструкции, включенные в цепочки, могут передаваться через UDP, только если вся цепочка помещаются в один сегмент.

7.1 Последовательность

Последовательность это тип цепочки, который объединяет зависящие друг от друга инструкции. Следующая инструкция последовательности может быть выполнена на VM, только если была выполнена предыдущая. Если текущая инструкция не может быть выполнена, все остальные инструкции данной последовательности (уже переданные или ожидающие передачи) просто отбрасываются. Благодаря этому для одного вычислительного потока управления можно не ждать положительного завершения текущей инструкции и сразу передавать следующую, если она относится к тому же узлу.

_BEGIN_FLOW. Расширенный заголовок “Начать последовательность” передается в первой инструкции последовательности. Он имеет следующие значения полей:

```
HEAD_CODE = 3
HEAD_LENGTH = 0
HOB = 1
```

Номер создаваемой цепочки содержится в поле CHAIN_NUMBER заголовка инструкции. Поле INSTR_NUMBER должно иметь значение 0.

Инициатором создания последовательности является VM. Последовательность не обязательно должна иметь заранее известную длину. Он может быть завершена в любой момент. Если нужно завершить последовательность и нет инструкций для передачи, может быть сформирована инструкция NOP.

7.2 Транзакции

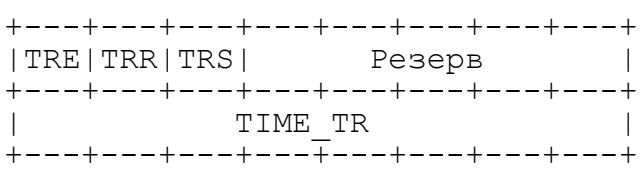
Транзакция это тип цепочки, объединяющей несколько, возможно не связанных друг с другом инструкций. Все инструкции транзакции должны быть выполнены все сразу либо не должны выполняться. Имеется возможность отменить или подтвердить выполнение транзакции. Отмена транзакции после выполнения не предусмотрена. Если необходимо, такой механизм должен быть реализован на уровне VM. Это объясняется тем, в транзакции могут быть инструкции, которые невозможно откатить, например передача управления.

Инициатором создания транзакции является VM. Длина транзакции должна быть заранее известна. Длина определяет способ, которым транзакция будет передаваться. Это связано с буферизацией, которая описана в секции 7.4.

7.2.1 _BEGIN_TR

Расширенный заголовок “Начать транзакцию” (_BEGIN_TR) передается в первой инструкции транзакции. Он имеет следующие значения полей:

HEAD_CODE = 4
HEAD_LENGTH = 1
HOB = 1
DATA - имеет следующий формат:



TRE

1 бит. Флаг обязательного выполнения. Этот флаг имеет отношение только к полностью переданной, но еще не выполненной транзакции. Если TRE = 1, то транзакция должна быть выполнена при истечении времени существования, установленным полем TIME_TR, или при аварийном завершении сеанса. Если TRE = 0, то при истечении времени существования транзакция должна быть отменена и передано отрицательное подтверждение, а при аварийном завершении сеанса - просто отменена.

TRR

1 бит. Флаг выполнения после передачи. Если TRR = 1, то транзакция должна быть выполнена сразу после передачи всех инструкций, из которых она состоит. Такая транзакция выполняется после получения инструкции с расширенным заголовком _END_CHAIN. Если TRR = 0, то для выполнения транзакции следует передать специальную инструкцию EXEC_TR подтверждения транзакции.

TRT

1 бит. Флаг специальной обработки. Он введен для возможности дальнейшего расширения протокола. Если TRT = 1, то перед выполнением транзакции над инструкциями, из которых она состоит, необходимо произвести некоторые дополнительные действия, например расшифровать. Эти действия могут быть определены в дополнительных расширенных заголовках, передаваемых в инструкциях транзакции. Данный документ не определяет случаев использования этого флага. Значение TRT должно быть нулевым.

Резерв

Должен быть установлен в 0.

TIME_TR

1 октет. Время жизни транзакции в 2 – секундных интервалах (максимальное время жизни – 8 минут). Принимающая сторона начинает отсчет этого времени после того, как получены все инструкции транзакции. Значение %x00 задает транзакцию без ограничения времени жизни.

В последней инструкции транзакции всегда передается заголовок _END_CHAIN.

7.2.2 EXEC_TR

Инструкция “Выполнить транзакцию” EXEC_TR указывает, что переданная раннее транзакция должна быть выполнена. Она имеет следующие значения полей:

```
OPCODE = 158
ASK = 1
PCK = b01/10/11
CHN = 1
EXT = 0/1
CHAIN_NUMBER - Содержит номер цепочки транзакции, которую
               необходимо выполнить.
INSTR_NUMBER = 0
OPR_LENGTH = 0
```

7.2.3 CANCEL_TR

Инструкция “Отменить транзакцию” (CANCEL_TR) указывает, что переданная раннее транзакция должна быть отменена. Она имеет следующие значения полей:

```
OPCODE = 159
ASK = 0
PCK = b01/10/11
CHN = 1
EXT = 0/1
CHAIN_NUMBER - Содержит номер цепочки транзакции, которую
               необходимо выполнить.
INSTR_NUMBER = 0
OPR_LENGTH = 0
```

Инструкции, из которых состоит отменяемая транзакция, удаляются без возможности восстановления.

7.3 Фрагментированные инструкции

UMSP рассчитан на работу с транспортным протоколом с ограниченным размером передаваемого сегмента данных. Фрагментация инструкций производится в следующих двух случаях:

(1) Если инструкция длиннее максимального размера сегмента транспортного уровня.
(2) Если сегмент формируется из нескольких инструкций и последняя инструкция полностью в него не помещается.

Решение о фрагментации принимается на уровне UMSP.

Фрагментируемая инструкция инкапсулируется в несколько инструкций NOP. Затем все инструкции NOP передаются, как одна цепочка специального типа. При инкапсуляции используется следующий алгоритм:

(1) Поля SESSION_ID и REQ_ID из фрагментируемой инструкции переписываются в первую инструкцию NOP. Если поля REQ_ID нет в исходной инструкции, его не должно быть в инструкции NOP. Поле SESSION_ID всегда присутствует в фрагментируемых инструкциях.
(2) Затем эти поля удаляются из исходной инструкции. При этом значения всех остальных полей заголовка не изменяется.
(3) После этого исходная инструкция делится на фрагменты необходимой длины. Каждый фрагмент помещается в поле операндов инструкции NOP. В поле операндов не должны записываться другие данные.

_BEGIN_FRG. Расширенный заголовок «Первый фрагмент» передается в инструкции NOP, которая содержит первый фрагмент. Он имеет следующие значения полей:

```
HEAD_CODE = 5
HEAD_LENGTH = 0/2  ; Зависит от подчиненности цепочки.
HOB = 1
Данные:
   2 октета: Номер родительской цепочки. Фрагментированная
             инструкция может являться частью
             последовательности или транзакции.
   2 октета: Номер инструкции в родительской цепочки.
```

В инструкции NOP, которая содержит последний фрагмент, передается заголовок _END_CHAIN.

7.4 Буферизация

В данном пункте описана буферизация, применяемая протоколом на приеме данных. Вопрос буферизации на передаче лежит вне сетевого протокола.

Если инструкция не включена в цепочку – она сразу передается к VM для выполнения и не требует буферизации на уровне протокола. Интерфейс UMSP – VM должен предусматривать асинхронную передачу инструкций. Рекомендуется, чтобы производительность систем, реализующих UMSP, позволяла обрабатывать инструкции, принимаемые из сети, с той скоростью, с какой они поступают. Все инструкции спроектированы таким образом, что несут известную и ограниченную вычислительную нагрузку. Исключением является инструкции передачи управления, которые должны обрабатываться в два этапа. Сначала проверяется корректность инструкции и производится ее диспетчеризация. Затем инструкция выполняется. При этом должно гарантироваться, что протокол может получить такую часть процессорного времени, которая позволяла бы ему работать в стационарном режиме. Таким образом, вопросы перегрузки узла выводятся на уровень VM и приложений пользователя, где ими можно осмысленно управлять.

Для цепочек протокол предусматривает две схемы буферизации на приеме:

(1) При установлении сеансового соединения стороны договариваются о размере выделенного буфера («окна»). Выделенный буфер всегда больше максимального сегмента транспортного уровня. Передающая сторона может рассчитывать на этот буфер без предварительного согласования с принимающей стороной. Размер выделенного буфера устанавливается однократно, для каждого

сеансового соединения, и не может изменяться в последующем. UMSP рассчитан на использование транспортного уровня, который сообщает о доставке данных. Поэтому передающая сторона самостоятельно отслеживает текущий свободный размер выделенного буфера на приемной стороне для каждого соединения. Если приемная сторона обнаруживает, что поступили данные, которые не могут быть помещены в выделенный буфер, соединение разрывается.

(2) Для транзакций и фрагментированных инструкций, размер которых превышает выделенный буфер, нужно запрашивать разрешения на передачу у приемного узла. Теоретический предельный размер цепочки, которая может быть передана таким образом – 4 Гбайт.

REQ_BUF. Инструкция «Запрос буфера» запрашивает у VM выделение буфера для передачи транзакции или большой фрагментированной инструкции. Она имеет следующие значения полей:

```
OPCODE = 24
ASK = 1
PCK = b01/11
CHN = 0
EXT = 0/1
OPR_LENGTH = 1
Операнды:
   4 октета: Размер запрашиваемого буфера в октетах. Значение
             равно суммарному размеру всех инструкций цепочки,
             включая размер подчиненных цепочек.
```

Инструкция формируется по инициативе протокола и в качестве подтверждения использует инструкции RSP_P. При этом на приемной стороне буфер выделяется на уровне VM, так как VM имеет наиболее полную информацию о задаче. Интерфейс между UMSP и VM должен предоставлять возможность асинхронного запроса такого буфера.

Инструкция REQ_BUF может быть использована независимо от того, помещается цепочка в выделенный для сессии буфер или нет. Следует учитывать, что на эту инструкцию может быть передано отрицательное подтверждение, а использование «окна» гарантирует передачу.

Подчиненная цепочка на приеме использует буфер родительской цепочки.

Передача последовательности в отличие от транзакции или фрагментированной инструкции не требует запроса на выделение буфера. Если для передачи используется одиночное соединение TCP, буферизация последовательности вообще не нужна. Если используется множественное соединение TCP с мультиплексированием, последовательность требует буферизации для неупорядоченных инструкций. В этом случае нужно использовать выделенный для сессии буфер.

Транзакции, у которых флаг TRR = 0, для передачи всегда должны запрашивать разрешение, даже если они могут поместиться в один сегмент транспортного уровня.

Буферизация фрагментированных инструкций и транзакций, у которых флаг TRR = 1, зависит от их размера:

- Если транзакция помещается в один сегмент транспортного уровня, она передается без буферизации.
- Если длина цепочки не больше «окна», она может быть передана без запроса выделения буфера. При этом место в буфере окна должно резервироваться до начала передачи. Если места в буфере недостаточно, передача не может быть начата. В этом случае

можно дождаться освобождения окна или использовать инструкцию запроса выделения буфера у VM REQ_BUF.
- Если длина превышает размер выделенного буфера необходимо использовать инструкцию REQ_BUF.

7.5 Подтверждение цепочек

Поле REQ_ID в цепочках любого типа устанавливается только в первой инструкции и относится ко всей цепочке. Все последующие инструкции, включая последнюю, не содержат REQ_ID.

Транспортный протокол, используемый для передачи цепочек, должен сообщать о завершении передачи данных. Это необходимо для того, чтобы передающая сторона знала свободный размер выделенного для сеанса окна на приемной стороне.

Если цепочка использует буфер, выделенный VM (запрашивалось разрешение на передачу REQ_BUF), или цепочка полностью помещается в сегмент транспортного уровня, протокол на передающей стороне не отслеживает подтверждений.

Если передается последовательность, передающая сторона получает информацию о свободном месте буфера на приемной стороне по подтверждениям доставки транспортного уровня. Это можно сделать, так как упорядоченные инструкции последовательности сразу после приема передаются VM и освобождают буфер.

Фрагментированные инструкции и транзакции не передаются VM до тех пор, пока не будут полностью приняты. Если используется окно сессии, занятие мест в буфере может быть вычислено по подтверждениям передачи транспортного уровня. Для того чтобы отслеживать освобождение мест нужно проверять подтверждения выполнения на VM. Для этого используется следующий алгоритм передачи:

- Значение поля REQ_ID, которое предоставила VM для передачи цепочки, сохраняется и вместо него записывается значение, установленное протоколом.
- Новое значение REQ_ID передается в первой инструкции цепочки.
- На приемной стороне цепочка полностью собирается в окне сессии. После сборки она передается для выполнения на VM. При этом цепочка может продолжать занимать место в буфере.
- После выполнения VM сообщает об этом протоколу приемной стороны. Протокол освобождает место в выделенном буфере.
- Затем протокол формирует и передает на цепочку подтверждение RSP_P, а не RSP, как в остальных случаях.
- Протокол на передающей стороне после получения подтверждения RSP_P корректирует величину свободного места в буфере приемной стороны.
- Затем восстанавливается старое значение REQ_ID и подтверждение передается VM.

7.6 Базовая адресация

Для инструкций из одной цепочки может быть установлен адрес базы памяти для относительной адресации. При этом в инструкциях цепочки можно использовать укороченные поля адреса памяти. Укороченные адреса будут использоваться, как смещение от базы.

_SET_MBASE. Расширенный заголовок “Установить базу памяти” устанавливает для цепочки значение базового адреса. Он имеет следующие значения полей:

```
HEAD_CODE = 7
HEAD_LENGTH = 2/4/8   ; Зависит от длины поля адреса.
```

```
HOB = 1
DATA содержит:
   4/8/16 октета: адрес базы памяти.
```

Адрес длиной 3 октета, записывается в последние октеты 4 октетного поля данных. Начальный октет устанавливается в 0. Для узлов с длиной адреса 2 октета базовая адресация не используется.

Значение базы памяти для последовательности может изменяться. Для всех инструкций транзакции база может устанавливаться один раз в любой инструкции. Повторное установление базы для транзакции является ошибкой, которая приводит к отказу выполнения транзакции.

8 Расширенные заголовки

Данная секция содержит описание расширенных заголовков, не связанных с определенной инструкцией. Описание специализированных расширенных заголовков приводится в соответствующих секциях данного документа.

8.1 _ALIGMENT

Расширенный заголовок “Выравнивание” (_ALIGMENT) позволяет выровнять любой расширенный заголовок или поле операндов на границу 4 - 16 октет с шагом в два октета. Протокол, не предоставляет каких либо правил использования данного расширенного заголовков. Он может использоваться произвольным образом. Заголовок имеет следующие значения полей:

```
HEAD_CODE = 8
HEAD_LENGTH = 0 - 7  ; Зависит от длины поля данных.
HOB = 0
DATA содержит:
   0 - 14 октет: Все октеты поля имеют значение 0.
```

Формат инструкций протокола подразумевает выравнивание на границу двух октет без дополнительных средств.

8.2 _MSG

Ррасширенный заголовок “Произвольное сообщение” (MSG) позволяет передавить текстовое сообщение в символах ASCII. Порядок обработки этого поля при приеме может быть любым. Оно может быть записано в log-файл, выдано на консоль или проигнорировано. Заголовок имеет следующие значения полей:

```
HEAD_CODE = 9
HEAD_LENGTH = 1 - 127  ; Зависит от длины поля данных.
HOB = 0
DATA содержит:
   2 - 254 октет: Произвольный текст сообщения.
```

Инструкция может содержать несколько заголовков _MSG.

8.3 _NAME

Расширенный заголовок “Имя” (_NAME) позволяет указать имя VM или группы VM, имя задания, имя объекта или процедуры объекта. Заголовок имеет следующие значения полей:

```
HEAD_CODE = 10
HEAD_LENGTH = 1 - 127  ; Зависит от длины поля данных.
HOB = 0
DATA содержит:
```

2 - 254 октет: Текст имени в символах ASCII.

8.4 _DATA

Расширенный заголовок “Данные” (_DATA) используется для передачи данных в инструкциях обмена между VM, если данные не могут быть размещены в операндах. Он позволяет в одной инструкции передать до 4 Гбайт данных. Заголовок имеет следующие значения полей:

```
HEAD_CODE = 11
HEAD_LENGTH = 1 - 2 147 483 647  ; Зависит от длины поля
данных.
HOB = 1
DATA содержит:
   2 - 4 294 967 294 октет: Двоичные данные в произвольном
                            формате.
```

8.5 _LIFE_TIME

Расширенный заголовок “Время жизни” (LIFE_TIME) содержит значение времени. Он имеет следующий формат:

```
HEAD_CODE = 12
HEAD_LENGTH = 1/2  ; В зависимости от длины данных.
HOB = 1
DATA содержит:
   2/4 октета: Время в 1,024 миллисекундных интервалах.
```

Заголовок _LIFE_TIME позволяет задать предельное время доставки инструкции к VM получателя.

Время жизни инструкции считается следующим образом:

- На передающей стороне учитывается время ожидания в очереди к транспортному уровню. В момент формирования пакета транспортного уровня значение времени уменьшается на время ожидания.
- На приемной стороне время жизни учитывается только для фрагментированных инструкций. Величина времени жизни уменьшается на время сборки инструкции. Для не фрагментированных инструкций этот заголовок при приеме игнорируется. При этом его значение должно передаваться VM.
- Время передачи на транспортном уровне не учитывается. Для фрагментированных инструкций не учитывается только время передачи первого фрагмента.

Завершение времени жизни у инструкции, относящейся к потоку, завершает передачу потока. При передаче транзакций заголовок _LIFE_TIME не должен использоваться.

Если инструкция фрагментируется, заголовок _LIFE_TIME передается только в инструкции NOP, содержащей первый фрагмент. Из исходной фрагментируемой инструкции этот заголовок удаляется. Если время истекает, когда часть фрагментируемой инструкции еще не передана, оставшееся часть инструкции отбрасывается.

Время жизни инструкции устанавливается VM отправителя и должно передаваться вместе с данными к VM получателя. Если время жизни истекает, инструкция отбрасывается и на нее передается отрицательное подтверждение (если ASK = 1). Если ASK = 0, подтверждение не передается.

Заголовок _LIFE_TIME может использоваться в системах мультимедиа для отказа от передачи устаревших данных и в системах реального времени. Протокол может повышать приоритет передачи данных с

заканчивающимся временем жизни. Это гарантирует передачу за заданный интервал.

9 Поиск ресурсов

Идентифицируемыми ресурсами протокола являются VM. Стандартизация VM не является функцией UMSP. Протокол предоставляет прозрачную среду для транспортировки кода и данных любого типа.

Для VM, подключенных к протоколу, устанавливаются следующие значения:

- Тип VM. Диапазон значений 1 – 65534.
- Версия VM. Диапазон значений 1 – 65534.

Протокол требует обязательной совместимости снизу вверх для VM одного типа и разными номерами версий (VM с большим номером версии должна уметь исполнять код VM с любым меньшим номером версии).

Номера типов VM разбиты на следующие диапазоны:

1 – 1023	Выделены для стандартных VM
1024 - 49151	Выделены для зарегистрированных VM пользователей
49152 – 65534	Свободные (определены для динамических и частных VM)

Номера типов и версий %x0000 и %xFFFF зарезервированы протоколом.

Несколько VM разного типа могут быть объединены в группу. Все VM, включенные в группу, должны работать в едином пространстве локальной памяти и иметь единую подсистему управления заданиями. Это означает, что если в коде любой VM для одной задачи пользователя встречается один и тот же 128-разрядный адрес, он должен указывать на одну физическую ячейку памяти. Выполнение указанных условий позволяет исполнять на одном узле неоднородный код пользователя (содержащий процедуры для разных VM). Все VM, входящие в группу, должны иметь разные типы. Группа может включать не более 65534 VM.

Каждой группе VM на узле присваивается код группы, длиной 2 октета. До тех пор, пока узел имеет хотя бы одно сеансовое соединение, коды групп не должны изменяться. Код группы рекомендуется изменять только при реконфигурации узла. Группа VM идентифицируется, также как одна VM. При этом тип VM устанавливается в 0, а версии VM присваивается номер группы. Один номер группы на разных узлах может идентифицировать группы с разным составом VM.

Поддержка объединения VM в группы не является обязательным требованием протокола. Неоднородный код пользователя может быть исполнен, даже если объединение в группы не поддерживается. Для этого процедуры, содержащие разный тип кода, должны выполняться на разных узлах.

UMSP предоставляет инструкции идентификации VM, которые позволяют определить, какие VM и группы VM подключены в данный момент к протоколу на определенном узле.

Инструкции идентификации VM могут передаваться по TCP или UDP. Может использоваться широковещательная рассылка. Узел может самостоятельно оповещать об имеющихся на нем VM, например, при включении, или отвечать на запросы других VM. Ответные инструкции должны передаваться по тому же протоколу, по которому был получен запрос.

VM из диапазоны номеров 49152 – 65534 или любая группа VM могут идентифицироваться по именам. VM с номерами 1 – 49151 не должны иметь имена на уровне инструкций UMSP.

9.1 VM_REQ

Инструкция «Запрос VM» (VM_REQ) позволяет узнать VM, подключенные на удаленном узле. Инструкция имеет следующие значения полей:

```
OPCODE = 25
PCK = %b00
CHN = 0
ASK = 0/1
EXT = 0/1
OPR_LENGTH = 0 – 65534  ; В зависимости от количества VM в
                          операндах.
Операнды:
   2 октета: Тип запрашиваемой VM. Нулевое значение не
             допускается.
   2 октета: Версия запрашиваемой VM. Нулевое значение не
             допускается. Значение %xFFFF запрашивает самую
             старшую версию
   .
   .
   .

   2 октета: Тип запрашиваемой VM.
   2 октета: Версия запрашиваемой VM.
Необязательный расширенный заголовок:
   _NAME – содержит имя запрашиваемой VM или группы VM.
```

Инструкция без операндов используется для запроса всех типов VM, подключенных на узле. Инструкция с одной VM в операндах запрашивает информацию об одной VM. Если в операндах содержится несколько VM, запрашивается группа VM, содержащая все указанные VM.

Чтобы запросить VM, используемую при работе без сеансового соединения, тип и версия VM должны быть установлены в %xFFFF.

Заголовок _NAME не связан с значениями в операндах. На него должен передаваться дополнительный ответ.

9.2 VM_NOTIF

Инструкция «Оповестить об VM» (VM_NOTIF) используется для оповещения о подключенной на узле одной VM или одной группы VM. Эта инструкция имеет следующие значения полей:

```
OPCODE = 26
PCK = %b00
CHN = 0
ASK = 0/1
EXT = 0/1
OPR_LENGTH = 1 – 65534  ; в зависимости от количества VM в
операндах
Операнды:
   2 октета: Используемый транспортный протокол. Определены
             следующие значения этого поля:
      %x0100 – Одиночное соединение TCP через порт 2110.
      %x0101 – Множественное соединение TCP через порт 2110.
      %x0102 – Одиночное соединение TCP через порт 2110 и UDP
               через порт на приеме 2110.
```

%x0103 - Множественное соединение TCP через порт 2110 и UDP через порт на приеме 2110.
В каждом соединении TCP порт 2110 должен быть хотя бы с одной стороны.
2 октета: Зарезервировано. Это поле не должно анализироваться на приеме и должно устанавливаться в 0 при передаче.
2 октета: Тип подключенной VM.
2 октета: Версия подключенной VM.

.
.
.

2 октета: Тип подключенной VM.
2 октета: Версия подключенной VM.
Необязательный расширенный заголовок:
_NAME - содержит имя отдельной VM или группы VM из операндов инструкции.

Если требуется сообщить о нескольких VM или группах, необходимо сформировать несколько инструкций. Если узел поддерживает несколько транспортных протоколов - необходимо для каждого протокола формировать отдельные инструкции.

Если инструкция используется для ответа на запрос VM_REQ, она может содержать ASK = 1 и REQ_ID, установленный в значение из инструкции запроса. Если запрашивалась группа VM, инструкция должна содержать одну VM с типом, равным 0, и версией, содержащий номер подходящей группы VM. Тип и версия VM в списке должны быть индексированы по возрастанию.

Указанные в инструкции VM_NOTIF протоколы может отличаться от протокола, через который передается эта инструкция.

10 Соображения безопасности

Настоящий документ содержит описание функций, минимально необходимых для реализации поставленной задачи - прямого доступа к памяти удаленного узла. Чтобы уменьшить начальную сложность протокола, решение вопросов безопасности не включено в документ. Все соображения данного раздела являются рекомендациями к дальнейшему расширению протокола.

Для описания использованы три узла - узел А и узел Б обмениваются данными. Узел Г является JCP.

Защита от прослушивания, подмены, переупорядочивания:

(1) Могут быть использованы средства, предусмотренные в TCP/IP.
(2) Имеется возможность создавать цепочки с особой обработкой. Чтобы создать такую цепочку в первой инструкции цепочки нужно передать расширенный заголовок, определяющий особую обработку. Инструкции цепочки могут быть инкапсулированы в инструкции NOP. Таким способом могут быть реализованы алгоритмы контроля целостности потока инструкций или шифрование.

Защита от человек-в-середине:

Защита основана на том, что маршруты между узлами А - Б, А - Г и Г - Б не пересекаются. Такая схема позволяет организовать дополнительный управляющий поток данных, позволяющий выявить

атаку данного типа. Если указанные маршруты проходят через один шлюз, эта защита менее эффективна.

Идентификация:

Работа протокола основана на принципе централизованного управления. Это позволяет использовать несколько схем идентификации. Параметры идентификации передаются в расширенных заголовках. Установление сеансового соединения может содержать до восьми шагов. Это также повышает гибкость при выборе алгоритма идентификации. Проведение идентификации возможно между тремя парами узлов А – Б, А – Г и Г – Б. Все пары могут использоваться в любой комбинации. Узел Г может быть специально выделен для проведения идентификации.

Защита от перегрузки:

Инструкции протокола имеют известную вычислительную нагрузку. Это позволяет проектировать узел таким образом, чтобы он мог обрабатывать инструкции с такой скоростью, с какой они поступают из сети. Возможным источником перегрузки является инструкции JUMP и CALL. Решением этой задачи должна заниматься VM. Она имеет полную информацию о задаче и может принимать решение о количестве выделяемых ресурсов. Решением проблемы является отказ в обслуживании низкоприоритетного трафика.

Защита на уровне архитектуры приложений:

Протокол позволяет создать приложения любой архитектуры. Это возможно благодаря асимметричному профилю соединения. Можно выделить три основные группы:

(1) Клиент, выполняющий функции терминала, и технологии клиент - сервер. Безопасность таких систем полностью определяется сервером. Такая архитектура представляется наиболее защищенной.
(2) Клиент, загружающий с сервера активный код. Это наименее защищенная архитектура с точки зрения клиента. На стороне сервера особых требований по защите не требуется.
(3) Клиент, выполняющий на сервере свой код. Эта архитектура безопасна для клиента. На сервере требуется повышенное внимание к защите. Функциональные возможности такой архитектуры не отличаются от архитектуры загрузки клиентом активного кода. Если учитывать, что сервер является специально выделенным компьютером, данная архитектура, вероятно, является оптимальной.

Все приведенные технологии могут использоваться одновременно в любом сочетании.

11 Использованные сокращения

API Application Programming Interface. Программный интерфейс приложения.

CTID Control Task IDentifier. Управляющий Идентификатор Задачи присваивает на узле JCP каждой задаче задания. Его длина равна длине адреса локальной памяти на узле JCP.

GJID Globally Job IDentifier. Глобальный Идентификатор Задания имеет каждое задания. GJID устанавливается на узле JCP. GJID имеет такой же формат, как и 128 – разрядный адрес памяти для JCP. В нем адрес локальной памяти заменен на CTID первой (инициирующей) задачи задания.

GTID Globally Task IDentifier. Глобальный Идентификатор Задачи присваивается каждой задаче на узле. GTID имеет такой же формат, как и 128 - разрядный адрес памяти для узла. В нем адрес локальной памяти заменен на LTID.

JCP Job Control Point. Узел, который управляет заданием.

LTID Locally Task IDentifier. Локальный Идентификатор Задачи присваивается каждой активной задаче на узле. Длина LTID равна длине адреса памяти, установленной для узла.

VM Virtual Machine. Виртуальная машина.

12 Ссылки

[1] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[2] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels". RFC 2119, March 1997.

[3] D. Crocker, Ed., P. Overell. "Augmented BNF for Syntax Specifications: ABNF." RFC 2234, November 1997.

[4] Postel, J., "Transmission Control Protocol - DARPA Internet Program Protocol Specification", STD 7, RFC 793, USC/Information Sciences Institute, September 1981.

[5] Postel, J., "User Datagram Protocol", STD 6, RFC 768, USC/Information Sciences Institute, August 1980.

[6] Srinivasan, R., "RPC: Remote Procedure Call Protocol Specification Version 2", RFC 1831, August 1995.

Адрес автора

Александр Богданов
EMail: a_bogdanov@iname.ru

Замечание автора

Если Ваши предложения подразумевают Ваши авторские права, то лучше публиковать их, как документы Internet-Draft http://www.ietf.org/ID.html (публикация этих документов не требует ничьего одобрения) или RFC http://www.rfc-editor.org/howtopub.html.