РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 405 260** (13) **C2**

(51) МПК
H04H 60/25 (2008.01)
G06F 15/173 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21), (22) Заявка: **2005138840/08, 13.12.2005**

(24) Дата начала отсчета срока действия патента:
13.12.2005

(30) Конвенционный приоритет:
14.12.2004 US 11/012,367

(43) Дата публикации заявки: **20.06.2007**

(45) Опубликовано: **27.11.2010** Бюл. № **33**

(56) Список документов, цитированных в отчете о поиске: **US 2003/0233455 A1, 18.12.2003. JP 2004118482 A1, 15.04.2004. EP 0811942 A2, 10.12.1997. RU 0097118661, 20.09.1999.**

Адрес для переписки:
129090, Москва, ул. Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. Ю.Д.Кузнецову, рег.№ 595

(72) Автор(ы):
МАККЕЙ Джейсон Ф. (US)

(73) Патентообладатель(и):
МАЙКРОСОФТ КОРПОРЕЙШН (US)

(54) СИСТЕМА И СПОСОБ ДЛЯ ЗАГРУЗКИ ОБНОВЛЕНИЙ

(57) Реферат:

Изобретение относится к системам для получения и установки обновлений, используя среду одноранговой сети. Техническим результатом является минимизирование времени загрузки для каждого однорангового узла в среде одноранговой связи с добавленной защитой, а также уменьшение расхода полосы пропускания при получении обновления программного обеспечения. Определяют в вычислительном устройстве необходимое обновление из списка доступных обновлений посредством сканирования вычислительного устройства, являющегося одним из одноранговых узлов, чтобы определить его текущее состояние. Сравнивают текущее состояние со списком доступных обновлений, причем необходимое обновление может быть сегментировано на множество частей. Принимают в вычислительном устройстве от хоста список вычислительных устройств в сети, которые идентифицируют загрузку обновления. Загружают в вычислительное устройство упомянутое множество частей обновления. 3 н. и 26 з.п ф-лы, 10 ил.

RU 2405260 C2

RU 2405260 C2

Фиг.2А

RU 2405260 C2

RU 2405260 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 405 260** (13) **C2**

(51) Int. Cl.
H04H 60/25 (2008.01)
G06F 15/173 (2006.01)

(12) ABSTRACT OF INVENTION

(21), (22) Application: **2005138840/08, 13.12.2005**

(24) Effective date for property rights:
13.12.2005

(30) Priority:
14.12.2004 US 11/012,367

(43) Application published: **20.06.2007**

(45) Date of publication: **27.11.2010 Bull. 33**

Mail address:
129090, Moskva, ul. B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. Ju.D.Kuznetsovu, reg.№ 595

(72) Inventor(s):
MAKKEJ Dzhejson F. (US)

(73) Proprietor(s):
MAJKROSOFT KORPOREJShN (US)

(54) SYSTEM AND METHOD FOR DOWNLOADING UPDATES

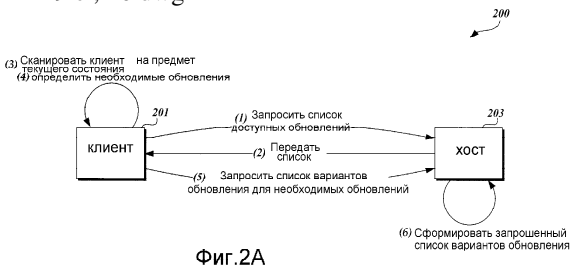
(57) Abstract:

FIELD: information technologies.

SUBSTANCE: in computing unit the necessary update is determined from a list of available updates by scanning of computing device, being one of peer-to-peer units, to determine its current condition. Current condition is compared to list of available updates, besides the necessary update may be segmented into multiple parts. List of online computing units, which identify downloading of update, is received in computing unit from host. Update is downloaded into computing device as specified multitude of parts.

EFFECT: minimisation of downloading time for each peer-to-peer unit in medium of peer-to-peer communication with added protection, reduced expenditure of pass band when obtaining update of software.

29 cl, 10 dwg



Фиг.2А

RU 2405260 C2

RU 2405260 C2

ОБЛАСТЬ ТЕХНИКИ, К КОТОРОЙ ОТНОСИТСЯ ИЗОБРЕТЕНИЕ

Вообще, настоящее изобретение относится к компьютерным обновлениям и, в частности, к системе и способу для получения и установки обновлений, используя среду одноранговой сети.

ПРЕДШЕСТВУЮЩИЙ УРОВЕНЬ ТЕХНИКИ

Компьютеры и, в частности, программное обеспечение для компьютеров часто обновляются пользователем, загружающим программное исправление с сетевого узла (хоста), например, от поставщика программного обеспечения, через сеть, такую как Интернет. В настоящее время исправления загружаются полностью из единого источника и устанавливаются на загружающей машине для обновления программного обеспечения до текущего состояния. В то время как эта методика является действенной, она имеет несколько недостатков.

Во-первых, поставщик программного обеспечения, который делает исправление доступным для загрузки, должен обеспечить достаточную полосу пропускания на выходы, чтобы позволить нескольким потребителям загружать обновления в одно и то же время. Для поставщиков программного обеспечения с большим количеством потребителей такая полоса пропускания на выходе может стать очень дорогостоящей.

Во-вторых, потребитель должен загрузить полное исправление из одного источника до установки исправления и до обновления программного обеспечения. В то время как это может быть приемлемо для исправлений небольшого размера, для более объемного исправления время, необходимое для загрузки исправления полностью из одного источника, может быть неприемлемым.

В-третьих, если обеспеченное программное обеспечение не сделало достаточной для загрузок полосу пропускания (что является особенно трудным и дорогостоящим, когда выпускается высокоприоритетное исправление, такое как обновление защиты), потребителю может быть временно запрещена загрузка исправления.

Недавние усовершенствования в работе с одноранговыми сетями обеспечили возможность загружать различные части большого файла с множества одноранговых узлов. Например, фиг.1 иллюстрирует блок-схему типичной среды одноранговой связи для загрузки заданного файла. Для загрузки заданного файла клиент 101 запрашивает этот заданный файл у хоста 103. В ответ хост передает список одноранговых узлов, идентифицирующий множество одноранговых узлов, которые делают доступными для загрузки различные части этих запрошенных файлов. Например, видеофайл может быть подразделен на множество частей, эти части разделены на подчасти, и эти подчасти разделены на подподчасти. Каждый одноранговый узел, который предварительно загрузил видеофайл, либо из одного источника, либо из множества других одноранговых узлов, может делать доступными для загрузки другими одноранговыми узлами, такими как клиент 101, одну или большее количество из этих подподчастей полного файла. Список одноранговых узлов, переданный хостом 103 клиенту 101, идентифицирует каждую из подподчастей запрошенного заданного файла и ассоциированный одноранговый узел, который делает доступными для загрузки эти подподчасти.

Клиент 101, после приема списка одноранговых узлов от хоста 103, выбирает одноранговый узел для каждой из подподчастей, из которого он будет загружать эту подподчасть. После идентификации одноранговых узлов клиент 101 загружает каждую из подподчастей из выбранных одноранговых узлов, пока все подподчасти заданного файла не будут загружены. По мере загрузки частей из разных одноранговых узлов множество загрузок может быть проведено одновременно. По

мере того как каждая подчасть загружается, клиент 101 может принять решение в отношении того, желает ли он стать одноранговым узлом, который делает эту подподчасть доступной для загрузки (то есть совместно использовать эту подподчасть). Если клиент решает, что он желает совместно использовать эту подподчасть, клиент 101 идентифицирует для хоста 103 подподчасть, которую он желает использовать совместно.

В то время как среды одноранговой связи, типа описанной со ссылкой на фиг.1, обеспечивают возможность клиенту загрузить заданный запрашиваемый им файл, такой как видеофайл, в настоящее время нет никакой методики для определения того, какие файлы могут понадобиться клиенту. Например, вновь обращаясь к загрузке программных исправлений, перед запрашиванием исправлений, клиент должен знать не только, какие обновления доступны, но и в каких обновлениях клиент нуждается. Помимо этого, в настоящее время нет никаких функциональных возможностей для защиты загружающего компьютера от компрометации идентификацией того, что он совместно использует, перед тем как этот материал будет полностью интегрирован в загружающий компьютер. Кроме того, не существует никакого способа, посредством которого множеству исправлений в локальном кэше клиента могут быть назначены приоритеты поставщиком программного обеспечения, когда клиенту нужно очистить пространство в локальном кэше.

Соответственно, есть потребность в системе и способе, которые позволяют клиенту идентифицировать необходимые обновления программного обеспечения и защищенным образом загружать эти обновления, используя среду одноранговой связи.

СУЩНОСТЬ ИЗОБРЕТЕНИЯ

Варианты осуществления данного изобретения дают возможность поставщику программного обеспечения распространять обновления программного обеспечения среди нескольких различных потребителей, используя среду одноранговой связи. Изобретение, описанное здесь, может использоваться для обновления программного обеспечения любого типа, включая, но не в ограничительном смысле, операционное программное обеспечение, программное обеспечение для программирования, антивирусное программное обеспечение, программное обеспечение баз данных и т.д. Использование среды одноранговой связи с добавленной защитой дает возможность минимизировать время загрузки для каждого однорангового узла и также уменьшить расход полосы пропускания на выходе, которая должна быть обеспечена поставщиком программного обеспечения, чтобы дать возможность его потребителям (одноранговым узлам) получить обновление.

В соответствии с первым аспектом данного изобретения предоставляется способ обновления первого вычислительного устройства. Согласно способу принимают список доступных обновлений и определяют, необходимы ли какие-либо из обновлений из списка. Если какие-либо из обновлений необходимы, принимают список вычислительных устройств, который идентифицирует части упомянутого обновления и вычислительные устройства, с которых эти части могут быть получены. Эти части затем загружают с вычислительных устройств, причем по меньшей мере две из множества частей загружают с разных вычислительных устройств. После того как части загружены, обновляют программное обеспечение первого вычислительного устройства, используя загруженные части.

В соответствии с другим аспектом данного изобретения предоставляется способ предоставления списка одноранговых узлов клиенту для загрузки обновления

программного обеспечения. Согласно способу передают список, идентифицирующий доступное обновление, и принимают запрос списка одноранговых узлов для доступного обновления. В ответ генерируют список одноранговых узлов, идентифицирующий множество одноранговых узлов, причем каждый идентифицированный одноранговый узел делает доступной для загрузки по меньшей мере часть доступного обновления. Затем этот список передают одноранговым узлам.

В соответствии с еще одним аспектом данного изобретения предоставляется компьютерная система, имеющая машиночитаемый носитель, включающий в себя машиноисполняемую программу для выполнения способа получения обновления. Компьютерная система принимает список доступных обновлений и идентифицирует необходимое обновление из списка доступных обновлений. Затем список одноранговых узлов запрашивается и принимается для необходимого обновления. Принятый список одноранговых узлов идентифицирует множество частей необходимого обновления и идентифицирует для каждой части одно или более вычислительных устройств, которые делают эту часть доступной для загрузки. Затем компьютерная система загружает каждую из частей с идентифицированных вычислительных устройств.

ПЕРЕЧЕНЬ ЧЕРТЕЖЕЙ

Предшествующие аспекты и многие из сопутствующих преимуществ этого изобретения станут с большей готовностью оцененными, поскольку они станут более понятными при обращении к нижеследующему подробному описанию, совместно с сопроводительными фигурами, на которых:

Фиг.1 - блок-схема типичной среды одноранговой связи для загрузки заданного файла;

Фиг.2A-2C - диаграмма состояний для загрузки обновления программного обеспечения с использованием среды одноранговой связи в соответствии с вариантом осуществления настоящего изобретения;

Фиг.3 - блок-схема списка вариантов обновления, идентифицирующего различные варианты для необходимых обновлений, которые могут быть сгенерированы хостом, в соответствии с вариантом осуществления настоящего изобретения;

Фиг.4 - блок-схема списка одноранговых узлов, идентифицирующая одноранговые узлы для необходимого обновления в ответ на запрос списка одноранговых узлов от клиента, в соответствии с вариантом осуществления настоящего изобретения;

Фиг.5 - блок-схема последовательности операций, иллюстрирующая процедуру обновления программного обеспечения клиента для обновления программного обеспечения клиента, в соответствии с вариантом осуществления настоящего изобретения;

Фиг.6 - блок-схема последовательности операций подпрограммы загрузки, которая может исполняться как часть процедуры обновления программного обеспечения клиента для загрузки частей необходимого обновления с различных одноранговых узлов, в соответствии с вариантом осуществления настоящего изобретения;

Фиг.7 - блок-схема последовательности операций подпрограммы приоритизации для увеличения размера кэша в системе клиента, чтобы обеспечить возможность загрузки исправления, в соответствии с вариантом осуществления настоящего изобретения; и

Фиг.8 - выполняемая хостом процедура, предназначенная для того, чтобы делать доступными обновления, используя среду одноранговой связи, в соответствии с вариантом осуществления настоящего изобретения.

ПОДРОБНОЕ ОПИСАНИЕ ПРЕДПОЧТИТЕЛЬНОГО ВАРИАНТА ОСУЩЕСТВЛЕНИЯ

Варианты осуществления настоящего изобретения обеспечивают возможность поставщику программного обеспечения, упоминаемому здесь как хост, распространять части обновлений программного обеспечения для нескольких различных получателей (упоминаемых здесь как одноранговые узлы). Одноранговый узел, в том смысле как он упоминается здесь, включает в себя любой тип вычислительного устройства, которое делает доступной для загрузки порцию (часть) обновления программного обеспечения. Например, одноранговый узел может быть, но не в ограничительном смысле, отдельным вычислительным устройством, сервером, вычислительным устройством-хостом, клиентским вычислительным устройством и т.д. Одноранговые узлы с частями обновления программного обеспечения делают эти части доступными другим одноранговым узлам для загрузки. Поскольку дополнительные одноранговые узлы получают те же самые части программного обеспечения, они также делают эти части доступными для загрузки другими одноранговыми узлами. Таким образом, поскольку число одноранговых узлов, содержащих некую часть, растет, растет число сайтов, доступных для загрузки этой части.

Фиг.2A-2C иллюстрируют диаграмму состояний для загрузки обновления программного обеспечения с использованием среды одноранговой связи в соответствии с вариантом осуществления настоящего изобретения. Среда 200 одноранговой связи обеспечивает клиенту 201 возможность идентифицировать необходимые обновления и получать список одноранговых узлов от хоста 203, с которого он может получить части этих обновлений.

Обращаясь сначала к фиг.2A, в начальном состоянии клиент 201 запрашивает от хоста список обновлений, которые являются доступными с хоста 203. В альтернативном варианте осуществления хост 203 периодически может публиковать список доступных обновлений, который клиент 201 получает. Хост 203, после приема запроса на список доступных обновлений, посылает список доступных обновлений клиенту 201. Клиент 201 выполняет сканирование в отношении самого себя для определения своего текущего состояния и сравнивает это состояние со списком доступных обновлений для определения того, какие обновления необходимы. После определения необходимых обновлений клиент 201 запрашивает список вариантов обновления для необходимых обновлений от хоста 203. Хост 203, после приема запроса на варианты обновления для необходимых обновлений, формирует список вариантов обновления.

Для учета аспектов конфиденциальности клиента в варианте осуществления настоящего изобретения минимальный объем информации клиента передается от клиента к хосту. Например, как только что было описано, хост предоставляет список доступных обновлений, и клиент решает, которые ему необходимы. В альтернативном варианте осуществления клиент мог бы обеспечить хост идентификацией его текущего состояния или информацией, соответствующей различным состояниям функционирования, и хост мог бы сообщить клиенту о необходимых обновлениях.

Со ссылкой на фиг.3 описывается список вариантов обновления, идентифицирующий различные варианты необходимых обновлений, который может быть сгенерированы хостом, в соответствии с вариантом осуществления настоящего изобретения. Список 300 вариантов обновления идентифицирует, для каждого запрашиваемого клиентом 201 обновления, такого как UPDATE1 301, различные

варианты, доступные для получения этой модификации. Дополнительно, список 300 вариантов обновления может включать в себя варианты обновления для нескольких требуемых обновлений. Например, предполагая, что клиент 201 запросил варианты обновления для UPDATE1 301 и UPDATE7 303, список 300 вариантов обновления включил бы в себя идентификацию каждого из запрошенных обновлений 301, 303 и идентификацию различных вариантов, доступных для получения этих обновлений. Например, UPDATE1 301 может включать в себя идентификацию полного исправления Full Patch1 305, дифференциального исправления Delta1.1 307, дифференциального исправления Delta1.2 309 и дифференциального исправления Delta1.3 311. Аналогично, UPDATE7 303 может включать в себя идентификацию полного исправления Full Patch7 313, дифференциального исправления Delta7.1 315, дифференциального исправления Delta7.2 317 и дифференциального исправления Delta7.3 319.

Полное исправление Full Patch1 305 является полным обновлением, которое может быть установлено для приведения программного обеспечения от существующего состояния до текущего состояния. Каждое дифференциальное исправление может использоваться для приведения программного обеспечения от известного существующего состояния до текущего состояния без необходимости использовать полное исправление. Программное обеспечение, которое должно быть обновлено, находится часто в одном из нескольких различных известных состояний, которые могут быть идентифицированы. Таким образом, для каждого из тех известных состояний дифференциальное исправление может использоваться для изменения этого программного обеспечения для приведения его к текущему состоянию. Например, дифференциальное исправление Delta1.1 307 может использоваться для приведения версии программного обеспечения в известном существующем состоянии к текущему состоянию.

Чтобы позволить клиенту 201 определять, может ли он использовать одно из дифференциальных исправлений для обновления программного обеспечения, каждое дифференциальное исправление включает в себя список файлов и ожидаемое значение хеш-функции для тех файлов. Ожидаемое значение хеш-функции может использоваться для определения того, является ли дифференциальное исправление соответствующим существующему программному обеспечению. Например, дифференциальное исправление Delta1.1 307 включает в себя файлы File1.1 320 с ожидаемым значением 321 хеш-функции, File1.2 322 с ожидаемым значением 323 хеш-функции и File1.3 324 с ожидаемым значением 325 хеш-функции. Клиент, после приема списка 300 вариантов обновления, может выполнять хеширование (применение хэш-функции) в отношении текущего состояния файлов, которые должны быть обновлены в системе клиента 201, и определять, совпадают ли значения хеш-функции с ожидаемыми значениями хеш-функции, содержащимися в списке обновлений. Если значения хеш-функции совпадают, то соответствующее дифференциальное исправление может использоваться для обновления системы клиента. Использование дифференциального исправления, такого как Delta1.1 307, вместо того, чтобы использовать полное исправление, такое как полное исправление Full Patch1 305, приводит к меньшей загрузке, потому что это является частичным обновлением вместо полного обновления.

Обращаясь теперь к фиг.2B, после того, как хост 203 сгенерировал требуемый список вариантов обновления, данный список вариантов обновления передается клиенту 201. Клиент 201 выбирает из списка вариантов исправление для каждого

обновления, которое приведет состояние клиента 201 к текущему состоянию. Как описано выше, клиент может выбрать полное исправление или соответствующее дифференциальное исправление для необходимых обновлений. Как будет оценено специалистом в данной области техники, одно или более необходимых обновлений могут быть выбраны из списка вариантов обновления. После выбора каждого из соответствующих исправлений клиент 201 запрашивает список одноранговых узлов для необходимых обновлений. Хост 203, в ответ на прием запроса на список одноранговых узлов для необходимых обновлений, формирует требуемый список одноранговых узлов и передает этот список одноранговых узлов клиенту 201.

Хост поддерживает главный список одноранговых узлов, который идентифицирует каждый одноранговый узел, который делает части обновления доступными для загрузки. Если обновление никогда не загружалось с хоста, начальный список одноранговых узлов может только идентифицировать хост как являющийся доступным одноранговым узлом. В такой ситуации клиент будет первым, кто загрузил обновление и выберет хост в качестве однорангового узла и загрузит полный файл с хоста. Как описано ниже, после того как клиент загрузил и установил обновление, он идентифицирует для хоста, что он делает доступными для загрузки (совместного использования) одну или более частей этого обновления. Таким образом, хост добавит этого клиента в главный список одноранговых узлов для частей обновления, которые клиент будет использовать совместно. По мере того как дополнительные клиенты загружают обновление с хоста, однорангового узла или их обоих и идентифицируют себя как теперь делающие части этого обновления доступными для загрузки, количество доступных одноранговых узлов растет.

Фиг.4 иллюстрирует блок-схему списка одноранговых узлов, идентифицирующего одноранговые узлы для необходимого обновления в ответ на запрос на список одноранговых узлов от клиента, в соответствии с вариантом осуществления настоящего изобретения. Продолжая вышеупомянутый пример, в предположении, что клиент 201 запросил дифференциальное исправление 401 Delta1.1 и полное исправление Full Patch7 403, список 400 одноранговых узлов включил бы в себя детали каждого из этих исправлений и идентификацию одноранговых узлов для этих исправлений. Значения хеш-функции также включены для верификации подлинности загруженных исправлений.

Например, дифференциальное исправление 401 Delta1.1 включает в себя результат хеширования различия и значение приоритета для этого исправления. Как будет описано более подробно ниже, значение хеш-функции для исправления может использоваться для подтверждения действительности результата хеширования до установки. Дополнительно, для каждого файла исправления, такого как File1.1 405 и File1.2 407, включен результат хеширования различия. Подобно результату хеширования различия для исправления, результат хеширования различия для файла может использоваться для верификации подлинности файла. Значение приоритета исправления может использоваться для определения того, должны ли части исправления быть сохранены в кэше компьютера-клиента или удалены. В альтернативном варианте осуществления значение приоритета может быть не включено в исправление. В таком варианте осуществления части сохраняются в кэше, и по мере того, как дополнительное пространство памяти становится необходимым, компьютер-клиент может запросить список приоритетов для частей, которые в настоящее время расположены в кэше (см. фиг.7). Это обеспечивает возможность хосту динамически управлять приоритетами частей во времени.

Дополнительно, каждый файл, такой как File1.1 405, может быть разделен на части, такие как Piece1 409 и Piece2 411. Каждая часть файла может далее быть разделена на подчасти, а эти подчасти могут быть разделены на подподчасти. Например, Piece1 409 может быть разделена на подчасти, такие как Sub-Piece1.1 413 и Sub-Piece1.2 415. Аналогично, эти подчасти могут быть далее разделены на подподчасти, такие как Sub-Sub-Piece1.1.1 417 и Sub-Sub-Piece1.1.2 419. Как будет оценено специалистами в данной области техники, деление на части, подчасти и подподчасти может быть предпринято до любого уровня разделения, и степень детализации, соответствующая подподчасти, как описано здесь, используется только для целей объяснения.

Наименьший уровень деления, такой как подподчасть, включает в себя идентификацию одного или более одноранговых узлов, которые делают ту часть доступной для загрузки. Например, Sub-Sub-Piece1 417 включает в себя идентификацию двух одноранговых узлов, которые делают ту часть доступной для загрузки. В частности Sub-Sub-Piece1.1.1 417 может быть загружен с однорангового узла Peer1 421 или однорангового узла Peer4 425. Каждый одноранговый узел, который делает подподчасть доступной для загрузки, идентифицируется адресом, например адресом протокола Internet. Список 400 одноранговых узлов может включать в себя идентификацию всех одноранговых узлов, которые делают подподчасть доступной для загрузки, части одноранговых узлов, которые делают подподчасть доступной для загрузки, или одного однорангового узла для каждой подподчасти. Дополнительно, для каждого однорангового узла, перечисленного в списке одноранговых узлов, может быть обеспечена дополнительная информация. Например, дополнительная информация об одноранговом узле может включать в себя, но не в ограничительном смысле, полосу пропускания однорангового узла, рейтинг однорангового узла (насколько надежен одноранговый узел), связь одноранговых узлов с компаниями или организациями, предысторию использования одноранговых узлов в одноранговой системе загрузки и т.д. Список одноранговых узлов может также включать в себя набор 427 команд обновления, который обеспечивает инструкции по установке непосредственно дифференциального исправления.

Обращаясь вновь к фиг.2B, клиент 201 после приема списка одноранговых узлов выбирает одноранговые узлы для подподчастей, идентифицированные в списке одноранговых узлов. Выбор части и однорангового узла будет описан более подробно ниже со ссылкой на фиг.6. После выбора подподчастей и одноранговых узлов, с которых эти части должны быть загружены, клиент 201 открывает несколько каналов связи с различными одноранговыми узлами и начинает загружать различные части обновления одновременно. Методика загрузки различных частей обновления с различных одноранговых узлов увеличивает скорость, с которой обновление может быть получено, и таким образом уменьшает полное время загрузки. Дополнительно, обеспечение обновлений через среду одноранговой связи уменьшает расход полосы пропускания на выходе, необходимый для хоста 203. Используя варианты осуществления настоящего изобретения, по мере того как клиенты обращаются к хосту, загружают обновления и начинают совместно использовать части этих обновлений с другими клиентами (то есть с другими одноранговыми узлами), расход полосы пропускания на выходе хоста 203 уменьшается, поскольку клиенты получают части обновления от других одноранговых узлов вместо этого хоста. Методика справляется с пиками запросов на определенный контент намного более эффективно, чем традиционное обслуживание файлов, которое может в лучшем случае стать очень

дорогостоящим, а в худшем случае потерпеть полную неудачу при загрузке.

Обращаясь теперь к фиг.2C, после того как загрузка каждой подподчасти завершается, эту подчасть хешируют и сравнивают со значением хеш-функции, включенным в список одноранговых узлов, для подтверждения того, что эта подподчасть не была изменена. После того как подподчасти конкретного исправления загружены и аутентифицированы, эти части собирают для воссоздания исправления. Дополнительно, копию одной или более из загруженных подподчастей сохраняют в кэше. Эти копии могут быть сделаны доступными для загрузки другими одноранговыми узлами.

Собранное исправление устанавливают для обновления клиента до текущего состояния. После завершения установки компьютерная система клиента может быть перезапущена в случае необходимости. После того как установка исправлений и перезапуск компьютера-клиента 201 (если необходимо) завершится, клиент 201 передает хосту 203 идентификацию тех подподчастей, которые хранятся в кэше. Такая идентификация может быть значением хеш-функции каждой сохраненной подподчасти. Хост 203, после приема идентификации кэшированных подподчастей, обновляет главный список одноранговых узлов, добавляя клиента в качестве одноранговых узлов, который делает эти части доступными для загрузки. Главный список одноранговых узлов включает в себя список каждого одноранговых узла, который поддерживает часть исправления, которая сделана доступной для загрузки другими клиентами.

Фиг.5 - блок-схема последовательности операций, иллюстрирующая процедуру обновления программного обеспечения клиента для обновления программного обеспечения клиента, в соответствии с вариантом осуществления данного изобретения. Процедура 500 обновления программного обеспечения клиента начинается на этапе 501, и на этапе 503 клиент запрашивает и получает список обновлений. Как описано выше, список обновлений включает в себя идентификацию обновлений, доступных для загрузки. Используя этот список, на этапе 505 клиент сравнивает состояние его существующей системы с доступными обновлениями. Базируясь на этом сравнении, на этапе ветвления 507 определяют, необходимы ли обновления, чтобы привести существующую систему клиента к текущему состоянию.

Если на этапе ветвления 507 определено, что обновления необходимы, на этапе ветвления 509 определяют, доступно ли дифференциальное исправление для необходимого обновления. Как описано выше, определение того, доступно ли дифференциальное исправление для необходимого обновления, может быть установлено путем запрашивания вариантов обновления для необходимого обновления и приема списка различных вариантов обновления. Список различных вариантов обновления включает в себя идентификацию дифференциальных исправлений и ожидаемых значений хеш-функции для файлов, которые будут обновлены дифференциальным исправлением. Сравнивая ожидаемые значения хеш-функции дифференциального исправления со значениями хеш-функции файлов на клиенте, клиент может определить, доступно ли дифференциальное исправление. В альтернативном варианте осуществления список доступных обновлений может также включать в себя варианты обновления для каждого доступного обновления, тем самым устраняя потребность впоследствии получать варианты обновления.

Разрешение клиенту определять, доступно ли дифференциальное исправление, в противоположность предоставлению клиентом значений хеш-функции своей существующей системы хосту, позволяет клиенту поддерживать свою

конфиденциальность. Если на этапе 509 ветвления определено, что дифференциальное исправление доступно, то данное дифференциальное исправление добавляется к списку загрузки, как иллюстрировано этапом 511. Однако, если на этапе 509 ветвления определено, что дифференциальное исправление недоступно, на этапе 513 идентификация полного исправления для обновления добавляется к списку загрузки.

На этапе 515 ветвления определяют, необходимы ли дополнительные обновления для приведения клиента к текущему состоянию. Если на этапе 515 ветвления определено, что дополнительные обновления необходимы, процедура 500 обновления программного обеспечения клиента возвращается к этапу 509 ветвления и процедура продолжает выполняться. Однако, если на этапе 515 ветвления определено, что дополнительные обновления не нужны, на этапе 517 клиент запрашивает список одноранговых узлов для обновлений, идентифицированных в списке загрузки. В ответ на запрос списка одноранговых узлов на этапе 517, на этапе 519 клиент принимает список одноранговых узлов, который включает в себя идентификацию одноранговых узлов, которые делают доступным для загрузки каждую подподчасть каждого исправления, идентифицированного в списке загрузки. После приема списка одноранговых узлов выполняется подпрограмма загрузки, иллюстрируемая этапом 521. Подпрограмма загрузки будет описана более подробно со ссылкой на фиг.6. После завершения подпрограммы загрузки на этапе 523 загруженные подподчасти собирают воедино для воссоздания требуемых исправлений, и эти исправления устанавливаются в системе клиента для приведения системы клиента к текущему состоянию.

После завершения установки загруженных исправлений и перезапуска системы клиента, в случае необходимости, процедура 500 обновления программного обеспечения клиента передает результат хеширования кэшированных подподчастей, как иллюстрировано этапом 525. Кэшированные подчасти - это те части, которые хранятся в кэше клиента и которые клиент будет делать доступными для загрузки другими одноранговыми узлами. В варианте осуществления настоящего изобретения, клиент запрашивается в отношении совместного использования частей, которые он загрузил с других одноранговых узлов, чтобы гарантировать, что среда одноранговой связи поддерживается.

Например, если клиент загружает 50 Мбайт обновлений через среду одноранговой связи, клиент может быть запрошен сделать доступными для загрузки тот же самый объем данных, который он сам загрузил. В фактическом варианте осуществления настоящего изобретения, результат хеширования кэшированных подподчастей может не быть передан, таким образом идентифицируя, что совместное использование не доступно, пока каждая из частей не будет верифицирована и установлена. Верификация и установка исправлений до их совместного использования могут быть необходимы, потому что, когда список одноранговых узлов предоставляется другим одноранговым узлам с хоста, адрес однорангового узла может использоваться со знанием об отсутствии у однорангового узла конкретного исправления, такого как загружаемый с целью компрометации машины однорангового узла. После того как результат хеширования кэшированных подподчастей передан на этапе 525, или если на этапе 507 ветвления определено, что никакие обновления не нужны, процедура 500 обновления программного обеспечения клиента завершается на этапе 527.

Процедура 500 обновления программного обеспечения клиента может быть инициирована вручную, или в альтернативном варианте намечена на периодическое выполнение, чтобы гарантировать, что клиент поддерживается в текущем состоянии.

Например, процедура 500 обновления программного обеспечения клиента может быть намечена на работу вечером, или когда вычислительное устройство клиента не используется. Дополнительно, в качестве добавленной функциональной возможности защиты, доступный список обновлений, список вариантов обновления и список одноранговых узлов могут каждый быть подписанными в цифровой форме хостом так, чтобы они не могли быть изменены. Подписание документов в цифровой форме известно в технике, и, таким образом, не будет описано здесь подробно.

Фиг.6 иллюстрирует блок-схему подпрограммы загрузки, которая может исполняться как часть процедуры обновления программного обеспечения клиента для загрузки частей исправления с различных одноранговых узлов, в соответствии с вариантом осуществления настоящего изобретения. Хотя подпрограмма 600 загрузки описана относительно загрузки одного исправления, следует понимать, что любое количество исправлений может быть загружено или одновременно, или в разное время, используя подпрограмму 600 загрузки.

Подпрограмма 600 загрузки начинается на этапе 601, а на этапе 603 принимают список одноранговых узлов для каждой из подподчастей одного или более исправлений. Как описано выше, список одноранговых узлов идентифицирует адреса одноранговых узлов, которые делают доступными для загрузки различные подподчасти исправления. В течение загрузки подподчастей список одноранговых узлов может периодически обновляться (например, каждые две минуты), чтобы гарантировать, что клиент запрашивает загрузки от доступных одноранговых узлов.

Используя этот список одноранговых узлов, на этапе 605 ветвления, определяют, будет ли необходим дополнительный кэш для хранения подподчастей исправления до установки. Если на этапе 605 ветвления определено, что дополнительный кэш будет необходим, выполняется подпрограмма приоритизации, как иллюстрировано этапом 607. Подпрограмма приоритизации для получения дополнительного кэша будет описана более подробно со ссылкой на фиг.7. Однако, если на этапе 605 ветвления определено, что дополнительный кэш не нужен, на этапе 609 выбирается файл загружаемого исправления. На этапе 611 выбирается часть выбранного файла, для которого загрузка должна начаться. Используя эту выбранную часть, на этапе 613 ветвления, определяют, закончена ли загрузка одной из подчастей этой части. Если определено, что загрузка одной из подчастей этой части не закончена, на этапе 615 ветвления определяют, была ли загружена одна из подподчастей данной подчасти. Если на этапе 615 ветвления определено, что одна из подподчастей подчасти не была загружена, на этапе 617 подподчасть подчасти выбирается в случайном порядке. На этапе 619 одноранговый узел, который делает доступным для загрузки случайным образом выбранную подподчасть, также выбирается в случайном порядке. После выбора подподчасти и одноранговового узла на этапе 621 начинается загрузка подподчасти с выбранного одноранговового узла. После того как загрузка подподчасти начата, подпрограмма 600 загрузки возвращается на этап 613 и продолжается.

Обращаясь вновь к этапу 615 ветвления, если определено, что загрузка одной из подподчастей завершена, на этапе 623 выбирается подподчасть подчасти, которая имеет наименьшее количество одноранговых узлов, с которых эта подчасть является доступной для загрузки. Дополнительно, на этапе 625 один из одноранговых узлов для выбранной подподчасти выбирается в случайном порядке. На этапе 627 инициируется загрузка выбранной подподчасти с выбранного в случайном порядке одноранговового узла. После того как загрузка выбранной подподчасти начата, на этапе 627, подпрограмма 600 загрузки возвращается на этап 613 и продолжается.

Поскольку множество подподчастей может быть загружено в одно и то же время, таким образом уменьшая полное время, необходимое для загрузки исправления или множества исправлений, после того как загрузка начата на этапе 621 или этапе 627, подпрограмма 600 загрузки продолжается, выбирая другую часть и другой одноранговый узел и инициируя эту загрузку. Таким образом, полное насыщение полосы пропускания для клиента, который загружает обновление, может быть достигнуто, и загрузка обновления может быть завершена быстро. Дополнительно, поскольку загрузка каждой подподчасти завершается, подлинность этих частей может быть верифицирована хешированием подчастей и сравнением этого значения хеш-функции со значением хеш-функции, включенным в список одноранговых узлов.

Возвращаясь теперь к этапу 613 ветвления, если определено, что загрузка подчасти части закончена, на этапе 629 ветвления определяют, закончена ли загрузка всех частей файла. Если на этапе 629 ветвления определено, что загрузка всех частей файла не закончена, подпрограмма 600 загрузки возвращается к этапу 611 и выбирает другую часть, и процесс продолжается. Однако, если на этапе 629 ветвления определено, что загрузка всех частей файла закончена, на этапе 631 ветвления определяют, закончена ли загрузка всех файлов для исправления. Если на этапе 631 ветвления определено, что загрузка всех файлов для исправления не закончена, подпрограмма 600 загрузки возвращается на этап 609 и выбирает другой файл, который должен быть загружен. Однако, если на этапе 631 ветвления определено, что все файлы исправления были загружены и таким образом все части, подчасти и подподчасти исправления были также загружены, подпрограмма завершается на этапе 633 и возвращает управление процедуре 500 обновления программного обеспечения клиента (фиг.5).

Фиг.7 иллюстрирует блок-схему подпрограммы приоритизации для увеличения размера кэша в системе клиента, чтобы обеспечить возможность загрузки исправления, в соответствии с вариантом осуществления настоящего изобретения. Подпрограмма 700 приоритизации начинается на этапе 701 и на этапе 703 хосту передается список всех в настоящее время кэшированных подподчастей, которые используются совместно. В ответ на передачу списка кэшированных подподчастей, которые используются совместно, на этапе 705 принимаются приоритеты для каждой из этих подподчастей. Подподчастям могут быть назначены приоритеты на основе частоты, с которой запрашивается их загрузка. Например, если новое обновление было опубликовано для обеспечения защиты против нового компьютерного вируса, исправлений и, таким образом, подподчасти для этого обновления могут загружаться с очень высокой частотой. Таким образом, подподчасти данного обновления могут получить высокий приоритет.

После приема приоритетов для кэшированных подчастей на этапе 707 кэшированная подчасть с самым низким приоритетом удаляется из кэша. На этапе 709 ветвления определяют доступно ли достаточное пространство в кэше для обновления, после того как совместно используемая подподчасть была удалена. Если на этапе 709 ветвления определено, что достаточное место доступно, процедура возвращает управление подпрограмме 600 загрузки (фиг.6), как иллюстрировано этапом 723. Однако, если на этапе 709 ветвления определено, что достаточное место все еще не доступно, на этапе 711 ветвления определяют, есть ли другие совместно используемые кэшированные подподчасти, которые еще не были удалены.

Если на этапе 711 ветвления определено, что все еще имеются другие совместно используемые кэшированные подподчасти, подпрограмма 700 приоритизации

возвращается на этап 707 и удаляет подподчасть со следующим самым низким приоритетом, и процесс продолжается. В фактическом варианте осуществления настоящего изобретения, некоторые кэшированные подподчасти, ассоциированные с исправлениями для конкретных обновлений, могут быть идентифицированы как неудаляемые. Например, если подподчасть ассоциирована с обновлением, которое загружается очень часто, она может быть идентифицирована как неудаляемая, чтобы гарантировать, что есть достаточно одноранговых узлов, с которых эта подподчасть может быть загружена.

Если на этапе 711 ветвления определено, что нет никаких остающихся кэшированных подподчастей, которые могут быть удалены, на этапе 713 ветвления определяют, было ли какое-либо исправление полностью загружено. Как описано выше, множество исправлений может быть загружено одновременно, в соответствии с вариантом осуществления настоящего изобретения. Если загрузка одного из этих исправлений завершена, на этапе 713 ветвления это идентифицируется, и на этапе 717 исправление устанавливают. Такая установка может привести к тому, что оставшиеся из загрузок будут приостановлены, в то время как обновление устанавливается. Однако существующая технология допускает возобновление приостановленных загрузок без необходимости перезапуска. Таким образом, нет никакой существенной потери загруженных данных из-за приостановки/перезапуска загрузки. После того как загруженное исправление установлено, на этапе 717 подчасти этого исправления удаляются из кэша, как иллюстрировано этапом 719, если они не идентифицированы как неудаляемые. После удаления загрузок на этапе 721 ветвления определяют, есть ли достаточно места в кэше, чтобы загрузить обновление. Если на этапе 721 ветвления определено, что нет достаточного места в кэше, подпрограмма 700 приоритизации возвращается на этап 713 ветвления и определяет, есть ли другие исправления, загрузка которых завершена, и процесс продолжается.

Возвращаясь назад к этапу 713 ветвления, если определено, что нет исправлений, загрузка которых завершена, на этапе 715 размер кэша увеличивают. Размер кэша может быть увеличен путем запроса в отношении того, чтобы пользователь клиентской системы увеличил размер кэша. Альтернативно, размер кэша может быть увеличен автоматически. После того как достаточное пространство в кэше готово к загрузке обновления, либо посредством удаления существующих частей из кэша, либо посредством увеличения размера кэша подпрограмма 700 приоритизации завершается на этапе 723, возвращая управление подпрограмме 600 загрузки.

Фиг.8 иллюстрирует выполняемую хостом процедуру для того, чтобы делать доступными обновления с использованием среды одноранговой связи, в соответствии с вариантом осуществления настоящего изобретения. Выполняемая хостом процедура 800 начинается на этапе 801, и на этапе 803 хост принимает запрос на список обновлений. Как описывалось выше, список обновлений является списком доступных обновлений, которые предоставляются этим хостом для загрузки. На этапе 805 хост передает список обновлений, идентифицирующий доступные загрузки.

После того как список обновлений передан, в некоторый более поздний момент времени, от клиента принимается запрос на варианты обновления для одного или большего количества доступных обновлений, идентифицированных в списке обновлений, как иллюстрированно этапом 807. На этапе 809 список вариантов обновления генерируется для обновлений, идентифицированных в запросе, принятом на этапе 807. Как описывалось выше, список вариантов обновления включает в себя идентификацию исправлений, которые могут использоваться для получения

обновления. Этот список затем передается клиенту, который запрашивал варианты обновления. На этапе 811 принимается запрос на список одноранговых узлов, в котором запрашивается список одноранговых узлов для одного или более исправлений, идентифицированных в списке вариантов, переданном на этапе 809. На этапе 813 хост генерирует и передает список одноранговых узлов для запрошенных исправлений. Как описывалось выше, список одноранговых узлов идентифицирует каждое исправление части, подчасти и подподчасти. Дополнительно, для каждой из этих подподчастей включаются один или более одноранговых узлов и адреса этих одноранговых узлов для идентификации того, откуда эти подподчасти могут быть получены. Список одноранговых узлов затем передается клиенту, и на этапе 815 принимается идентификация тех подподчастей, которые были загружены и сохранены данным клиентом и которые являются теперь доступными для совместного использования с другими одноранговыми узлами. Принятый список используется для обновления главного списка одноранговых узлов 817, который включает в себя идентификацию всех подподчастей для доступных обновлений и одноранговых узлов, которые имеют эти подподчасти доступными для загрузки. Выполняемая хостом процедура 800 завершается на этапе 819.

Хотя варианты осуществления изобретения были иллюстрированы и описаны, следует понимать, что в них могут быть сделаны различные изменения, не выходя за рамки объема изобретения.

Формула изобретения

1. Способ обновления первого вычислительного устройства в сети, содержащей это первое вычислительное устройство и хост, причем в первом вычислительном устройстве имеется кэш для хранения по меньшей мере одного обновления, при этом способ содержит этапы, на которых

(A) принимают от хоста в первом вычислительном устройстве список доступных обновлений;

(B) определяют в первом вычислительном устройстве, в качестве реакции на этап (A), необходимое обновление из списка доступных обновлений посредством сканирования первого вычислительного устройства, чтобы определить его текущего состояние, и сравнения этого текущего состояния со списком доступных обновлений, причем необходимое обновление сегментировано на множество частей;

(C) принимают в первом вычислительном устройстве от хоста список вычислительных устройств в сети, который идентифицирует для каждой части из упомянутого множества частей по меньшей мере одно вычислительное устройство, которое делает эту часть доступной для загрузки;

(D) загружают в первое вычислительное устройство упомянутое множество частей с по меньшей мере некоторых вычислительных устройств из упомянутого списка вычислительных устройств, причем по меньшей мере две части из упомянутого множества частей загружают с разных вычислительных устройств; и

(E) обновляют программное обеспечение первого вычислительного устройства, используя загруженное множество частей.

2. Способ по п.1, дополнительно содержащий этапы, на которых

принимают список вариантов обновления для необходимого обновления, причем каждый вариант обновления сегментирован на множество частей;

выбирают вариант обновления из списка вариантов обновления, при этом при приеме от хоста в первом вычислительном устройстве списка вычислительных

устройств принимают список вычислительных устройств, идентифицирующий для каждой части выбранного варианта обновления по меньшей мере одно вычислительное устройство.

3. Способ по п.2, в котором список вариантов обновления включает в себя для необходимого обновления полное исправление и по меньшей мере одно дифференциальное исправление.

4. Способ по п.3, в котором дифференциальное исправление включает в себя имя файла и ожидаемое значение хеш-функции для файла.

5. Способ по п.1, дополнительно содержащий этап, на котором удаляют посредством первого вычислительного устройства компонент в кэше, если сделано определение в отношении необходимости сформировать доступное пространство в кэше, причем данное удаление включает в себя этапы, на которых

генерируют список приоритетов для обновлений, сохраненных в кэше, посредством назначения приоритета каждому обновлению, сохраненному в кэше,

определяют, имеется ли в кэше доступное пространство для загрузки упомянутого множества частей, и

если доступного пространства в кэше нет, выбирают для удаления компонент, который соответствует обновлению, имеющему низкий приоритет в списке приоритетов, причем назначение приоритета для каждого обновления основывается на частоте запросов по каждому обновлению.

6. Способ по п.1, в котором список вычислительных устройств включает в себя для по меньшей мере одной из упомянутого множества частей идентификацию множества вычислительных устройств.

7. Способ по п.1, в котором вычислительные устройства идентифицируются адресом протокола Internet.

8. Способ по п.1, в котором при обновлении программного обеспечения собирают загруженные части в исправление и устанавливают это исправление.

9. Способ по п.1, в котором каждую загруженную часть подтверждают на основе результата ее хеширования.

10. Способ по п.1, дополнительно содержащий этапы, на которых обеспечивают идентификацию загруженного множества частей и совместно используют идентифицированное загруженное множество частей с другими вычислительными устройствами.

11. Способ по п.10, в котором идентификация загруженных частей включает в себя значение хеш-функции каждой загруженной части.

12. Способ по п.1, в котором с предопределенного момента идентифицированные загруженные части не используются совместно.

13. Способ предоставления списка одноранговых узлов клиенту, имеющему кэш, для загрузки обновлений программного обеспечения в сети, содержащей упомянутый клиент и хост, при этом способ содержит этапы, на которых

принимают посредством хоста от клиента идентификацию текущего состояния хоста или информацию, соответствующую состояниям функционирования хоста;

передают посредством хоста на клиент список, идентифицирующий варианты для доступного обновления, определенные хостом для клиента на основе принятой идентификации или информации;

принимают на хосте от клиента запрос списка одноранговых узлов, где список одноранговых узлов идентифицирует одноранговые узлы, из которых следует запрашивать вариант обновления из упомянутого списка вариантов обновления;

генерируют посредством хоста список одноранговых узлов, идентифицирующий множество одноранговых узлов и множество частей, каковое множество частей в совокупности образует запрашиваемый вариант обновления; и

передают посредством хоста на клиент список одноранговых узлов, где список одноранговых узлов идентифицирует по меньшей мере один одноранговый узел для каждой из упомянутого множества частей.

14. Способ по п.13, в котором список, идентифицирующий варианты для доступного обновления, идентифицирует множество доступных вариантов обновления.

15. Способ по п.13, в котором упомянутое множество вариантов обновления включает в себя полное исправление и по меньшей мере одно дифференциальное исправление.

16. Способ по п.13, в котором при приеме запроса списка одноранговых узлов на хосте от клиента принимают запрос списка одноранговых узлов для одного из упомянутого множества вариантов обновления.

17. Способ по п.13, дополнительно содержащий этапы, на которых принимают идентификацию кэшированной части доступного обновления и обновляют главный список одноранговых узлов для доступного обновления.

18. Способ по п.13, в котором один из идентифицированных одноранговых узлов является компьютером-хостом.

19. Компьютерная система, имеющая машиночитаемый носитель, включающий в себя машиноисполняемую программу для выполнения способа получения обновления в сети, содержащей данную компьютерную систему и хост, причем в упомянутой компьютерной системе имеется кэш для хранения обновления, при этом способ содержит

(A) прием в компьютерной системе от хоста списка доступных обновлений;

(B) идентификацию в компьютерной системе в качестве реакции на этап (A) необходимого обновления из списка доступных обновлений посредством сканирования компьютерной системы, чтобы определить ее текущее состояние, и сравнения этого текущего состояния со списком доступных обновлений;

(C) запрос компьютерной системой у хоста списка одноранговых узлов для необходимого обновления;

(D) прием в компьютерной системе от хоста списка одноранговых узлов для необходимого обновления, при этом список одноранговых узлов идентифицирует множество частей необходимого обновления и идентифицирует для каждой части вычислительное устройство в сети, которое делает эту часть доступной для загрузки;

(E) загрузку в компьютерную систему каждой из упомянутых частей с идентифицированных вычислительных устройств.

20. Компьютерная система по п.19, в которой список одноранговых узлов идентифицирует для каждой из упомянутого множества частей множество подчастей и идентифицирует для каждой подчасти вычислительное устройство, которое делает эту подчасть доступной для загрузки.

21. Компьютерная система по п.20, в которой список одноранговых узлов идентифицирует для каждой из упомянутого множества подчастей множество подподчастей и идентифицирует для каждой подподчасти вычислительное устройство, которое делает эту подподчасть доступной для загрузки.

22. Компьютерная система по п.19, в которой список одноранговых узлов идентифицирует значение хеш-функции для каждой части.

23. Компьютерная система по п.22, в которой этап (Е) включает в себя хеширование загруженной части для получения второго значения хеш-функции и

сравнение первого значения хеш-функции и второго значения хеш-функции для подтверждения действительности загруженной части,

при этом способ дополнительно содержит удаление компьютерной системой компонента в кэше, если сделано определение в отношении необходимости сформировать доступное пространство в кэше, причем данное удаление включает в себя

генерирование списка приоритетов для обновлений, сохраненных в кэше, посредством назначения приоритета каждому обновлению, сохраненному в кэше,

определение того, имеется ли в кэше доступное пространство для загрузки упомянутого множества частей, и

выбор, если доступного пространства в кэше нет, для удаления компонента, который соответствует обновлению, имеющему низкий приоритет в списке приоритетов, причем назначение приоритета для каждого обновления основывается на частоте запросов по каждому обновлению.

24. Компьютерная система по п.19, в которой способ дополнительно содержит сборку каждой из загруженных частей и установку обновления.

25. Компьютерная система по п.24, в которой список одноранговых узлов включает в себя первое значение хеш-функции для обновления, при этом способ дополнительно включает в себя

хеширование собранных частей для получения второго значения хеш-функции и

сравнение первого значения хеш-функции и второго значения хеш-функции для подтверждения действительности обновления.

26. Компьютерная система по п.19, дополнительно содержащая сохранение по меньшей мере одной из загруженных частей в кэше;

передачу идентификации кэшированной части и обеспечение доступности кэшированной части для загрузки.

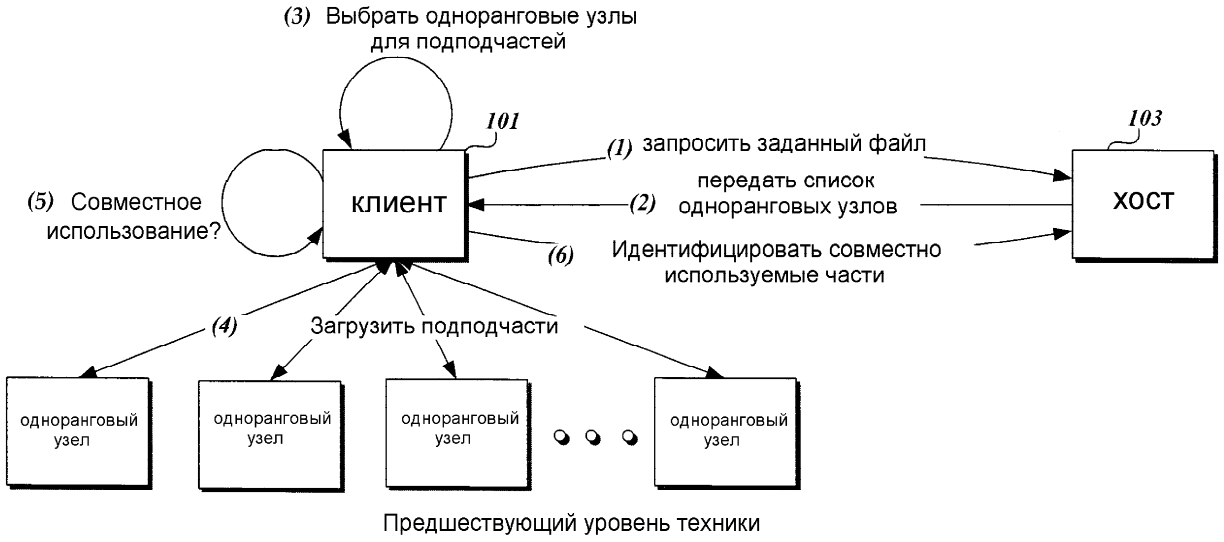
27. Компьютерная система по п.19, в которой загрузка в компьютерную систему каждой из частей с идентифицированных вычислительных устройств включает в себя подтверждение подлинности загруженной части.

28. Компьютерная система по п.19, в которой загрузка в компьютерную систему каждой из частей включает в себя случайный выбор части для загрузки и случайный выбор вычислительного устройства для загрузки этой части.

29. Компьютерная система по п.19, в которой загрузка в компьютерную систему каждой из частей включает в себя

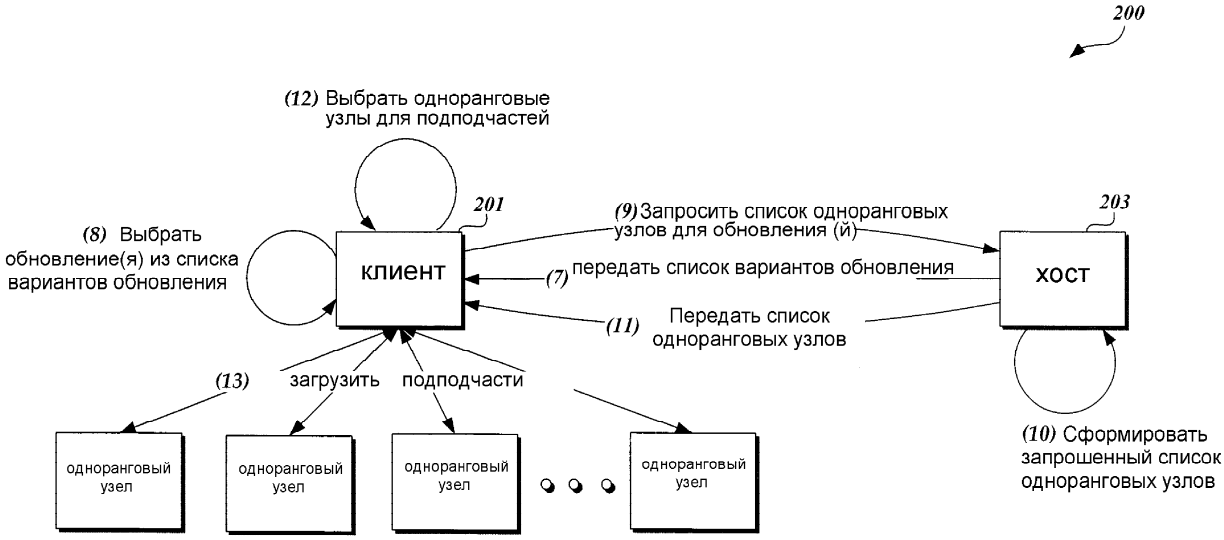
идентификацию части с наименьшим количеством вычислительных устройств, которые делают эту часть доступной для загрузки; и

загрузку части с наименьшим количеством вычислительных устройств, которые делают эту часть доступной для загрузки.



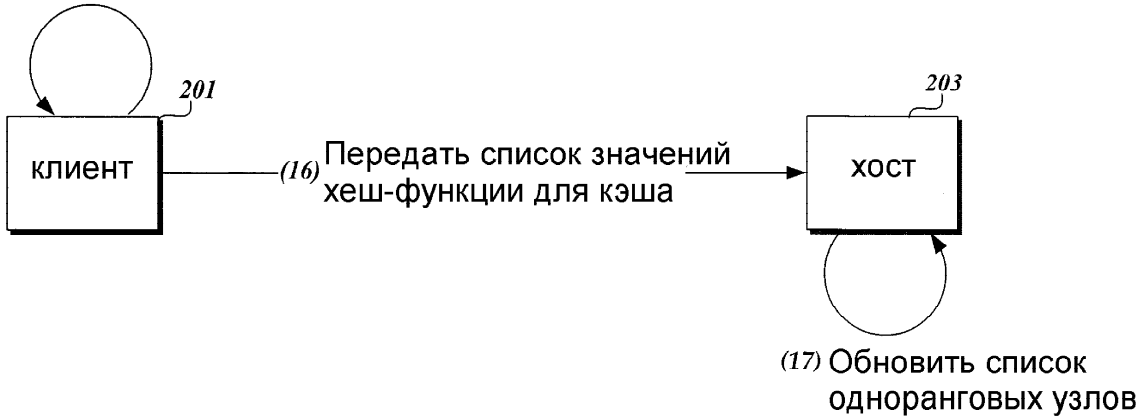
Предшествующий уровень техники

Фиг.1

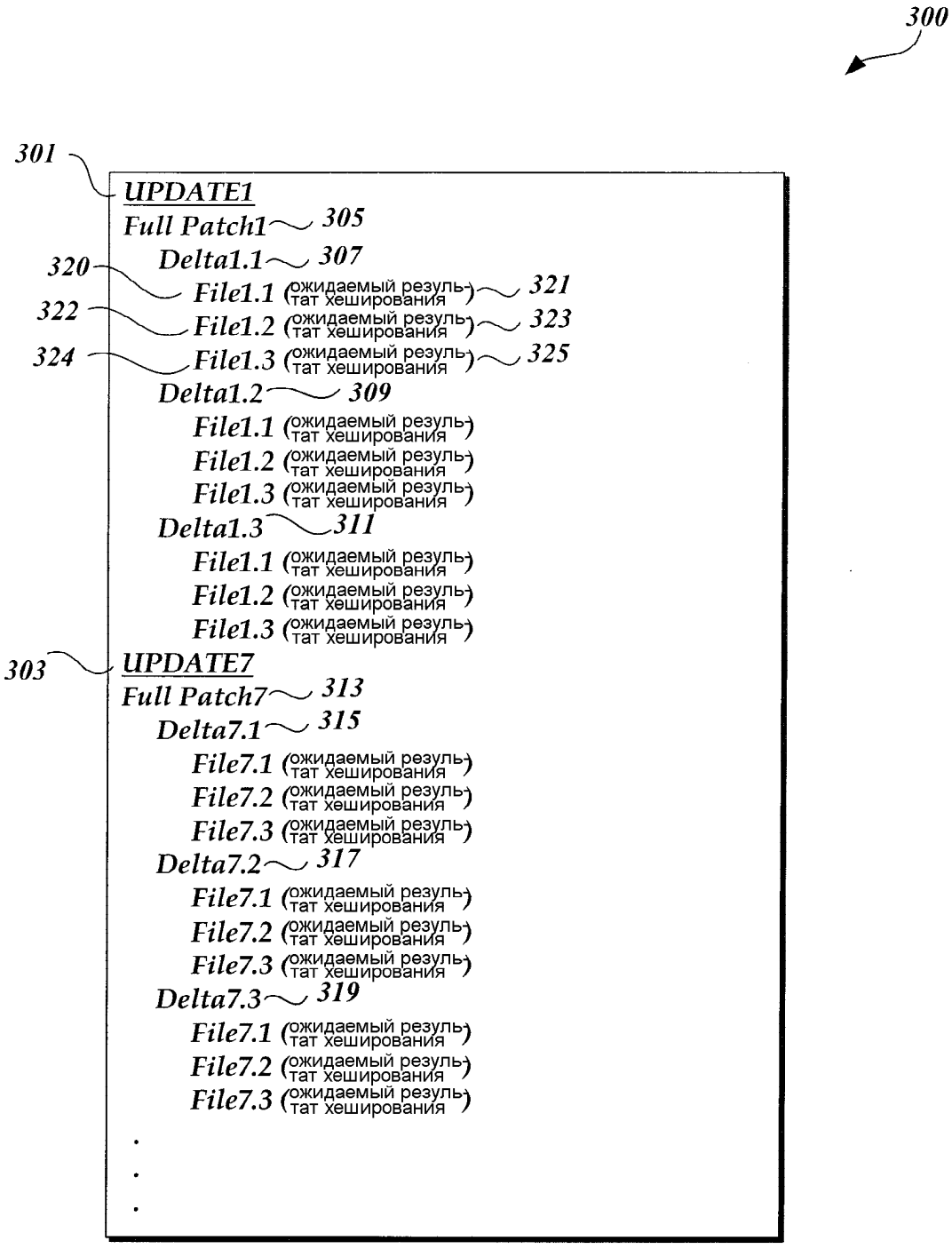


Фиг.2B

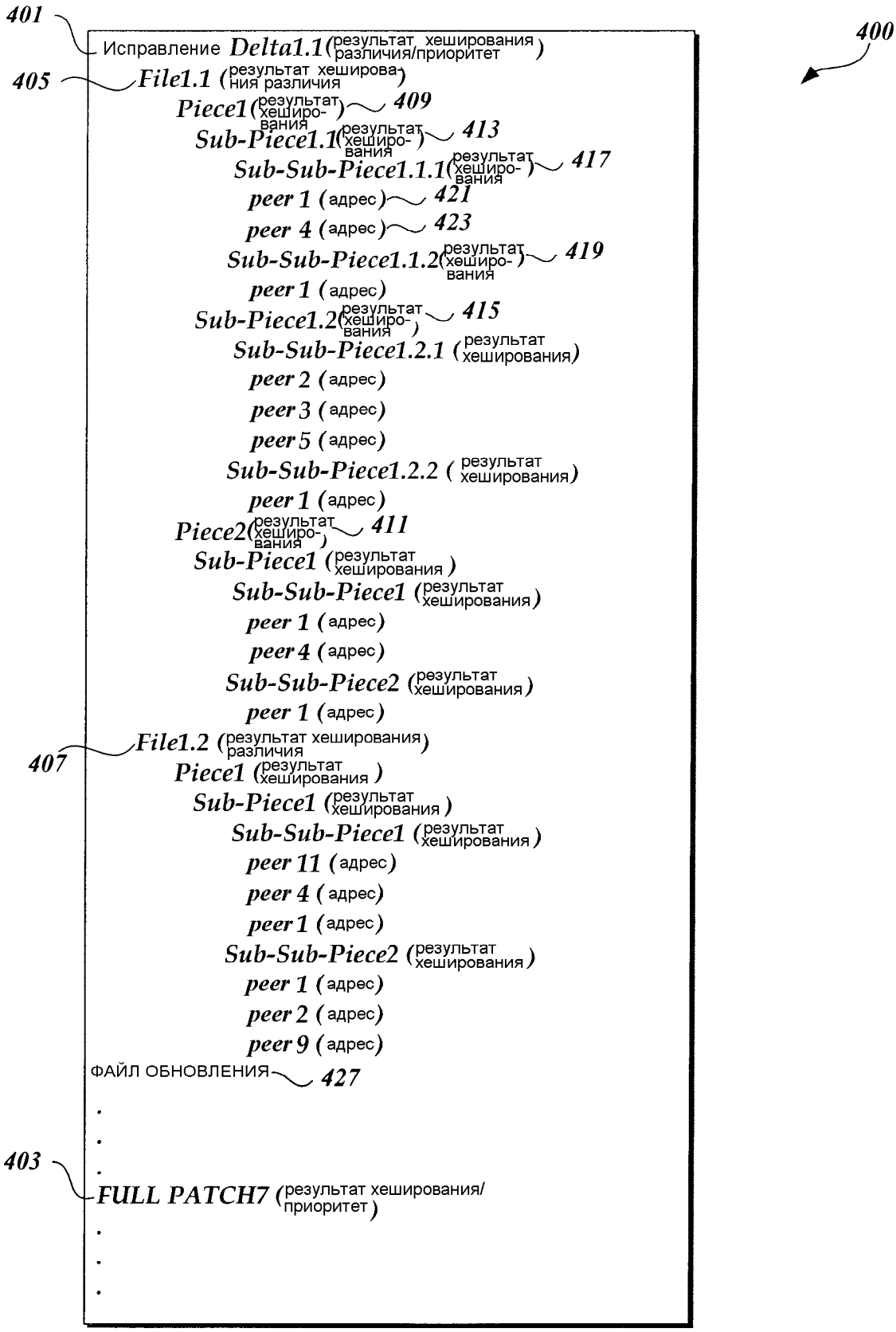
(14) Собрать и установить обновления
(15) Выполнить перезапуск при необходимости



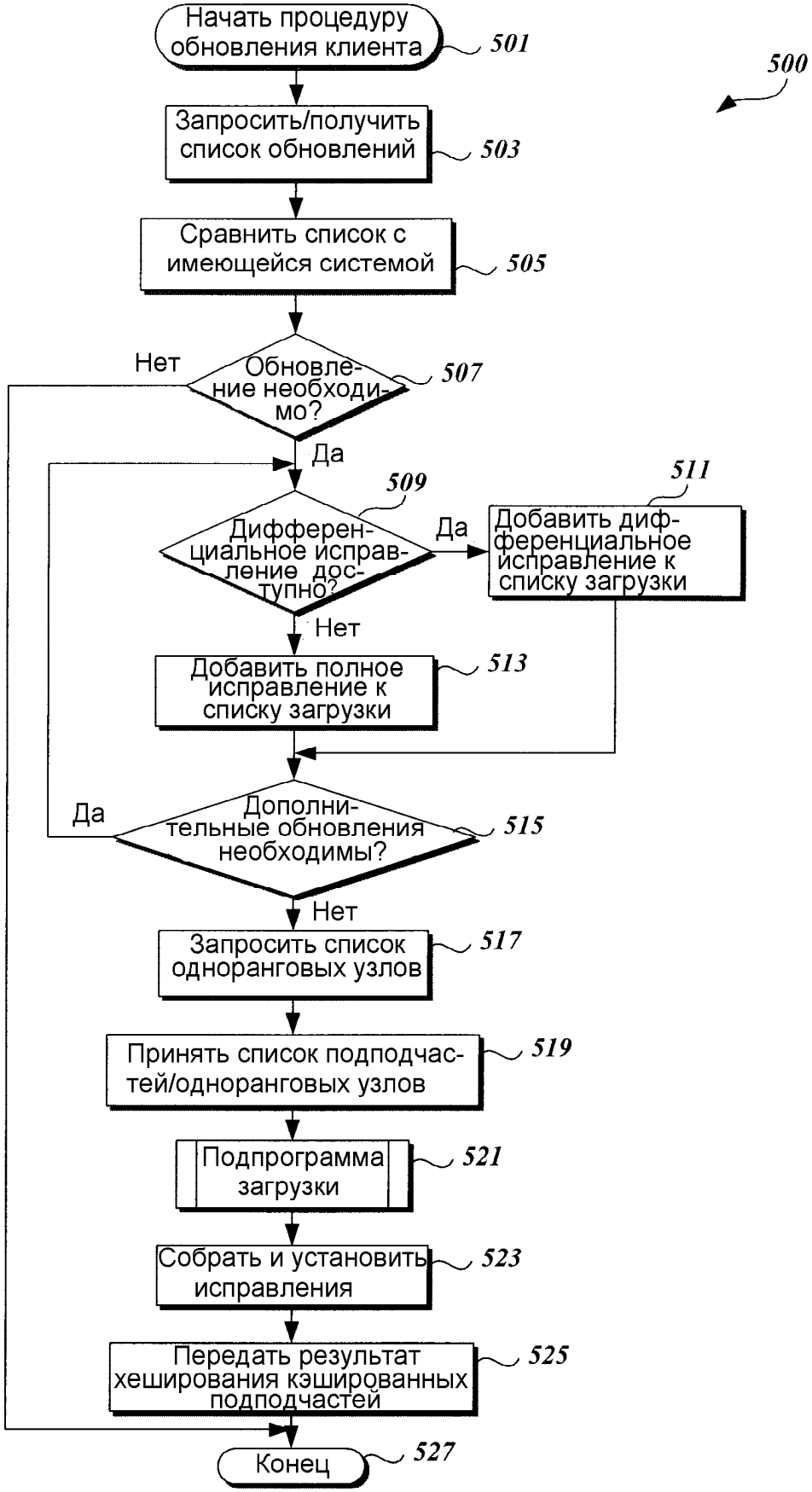
Фиг.2C


300
301
UPDATE1
Full Patch1
305
Delta1.1
307
320
File1.1 (ожидаемый результат хеширования)
321
322
File1.2 (ожидаемый результат хеширования)
323
324
File1.3 (ожидаемый результат хеширования)
325
Delta1.2
309
File1.1 (ожидаемый результат хеширования)
File1.2 (ожидаемый результат хеширования)
File1.3 (ожидаемый результат хеширования)
Delta1.3
311
File1.1 (ожидаемый результат хеширования)
File1.2 (ожидаемый результат хеширования)
File1.3 (ожидаемый результат хеширования)
303
UPDATE7
Full Patch7
313
Delta7.1
315
File7.1 (ожидаемый результат хеширования)
File7.2 (ожидаемый результат хеширования)
File7.3 (ожидаемый результат хеширования)
Delta7.2
317
File7.1 (ожидаемый результат хеширования)
File7.2 (ожидаемый результат хеширования)
File7.3 (ожидаемый результат хеширования)
Delta7.3
319
File7.1 (ожидаемый результат хеширования)
File7.2 (ожидаемый результат хеширования)
File7.3 (ожидаемый результат хеширования)

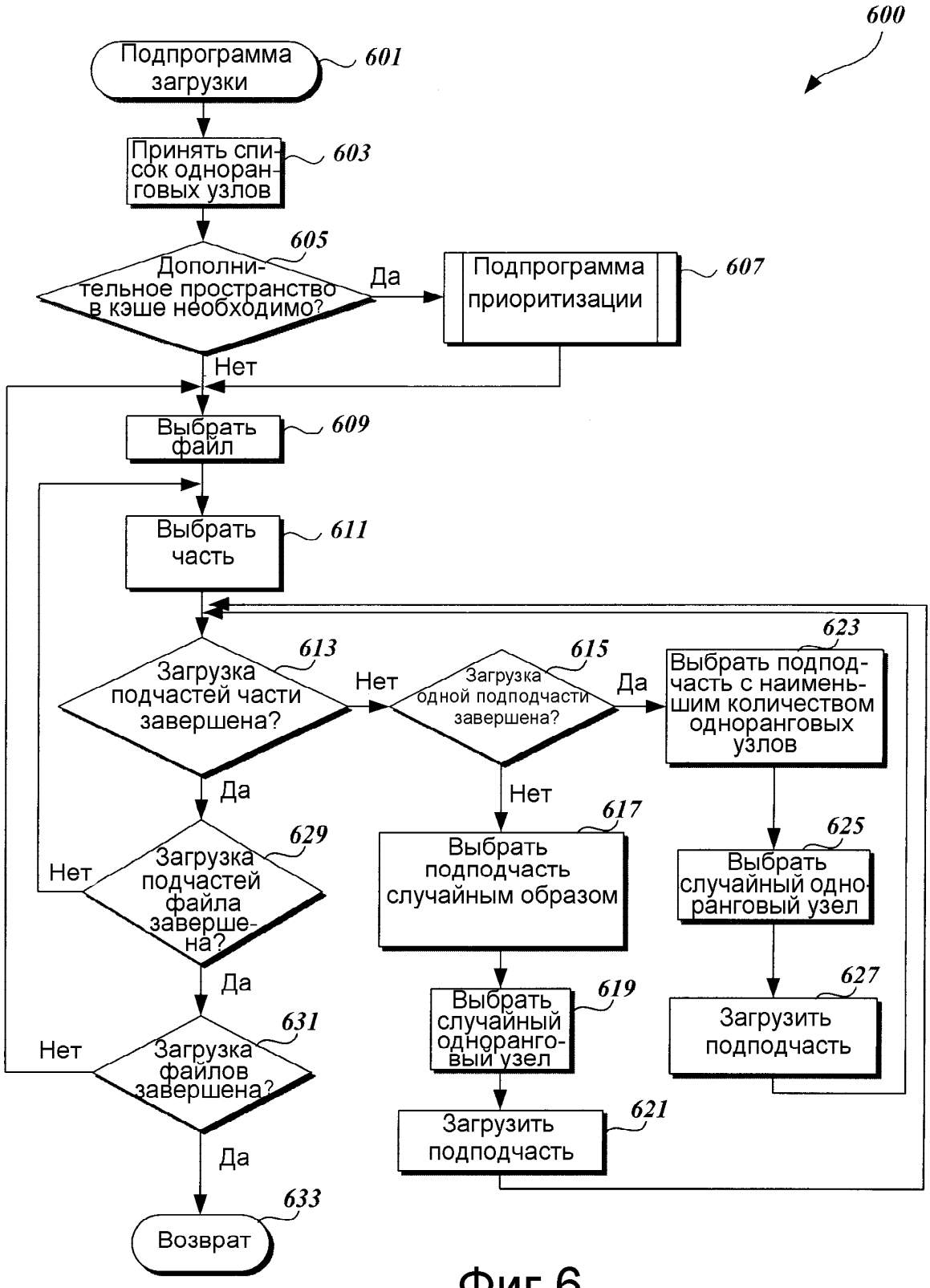
Фиг.3


401
400
Исправление Delta1.1 (результат хеширования различия/приоритет)
405
File1.1 (результат хеширования различия)
Piece1 (результат хеширования) 409
Sub-Piece1.1 (результат хеширования) 413
Sub-Sub-Piece1.1.1 (результат хеширования) 417
peer 1 (адрес) 421
peer 4 (адрес) 423
Sub-Sub-Piece1.1.2 (результат хеширования) 419
peer 1 (адрес)
Sub-Piece1.2 (результат хеширования) 415
Sub-Sub-Piece1.2.1 (результат хеширования)
peer 2 (адрес)
peer 3 (адрес)
peer 5 (адрес)
Sub-Sub-Piece1.2.2 (результат хеширования)
peer 1 (адрес)
Piece2 (результат хеширования) 411
Sub-Piece1 (результат хеширования)
Sub-Sub-Piece1 (результат хеширования)
peer 1 (адрес)
peer 4 (адрес)
Sub-Sub-Piece2 (результат хеширования)
peer 1 (адрес)
File1.2 (результат хеширования различия)
407
Piece1 (результат хеширования)
Sub-Piece1 (результат хеширования)
Sub-Sub-Piece1 (результат хеширования)
peer 11 (адрес)
peer 4 (адрес)
peer 1 (адрес)
Sub-Sub-Piece2 (результат хеширования)
peer 1 (адрес)
peer 2 (адрес)
peer 9 (адрес)
ФАЙЛ ОБНОВЛЕНИЯ 427
403
FULL PATCH7 (результат хеширования/приоритет)

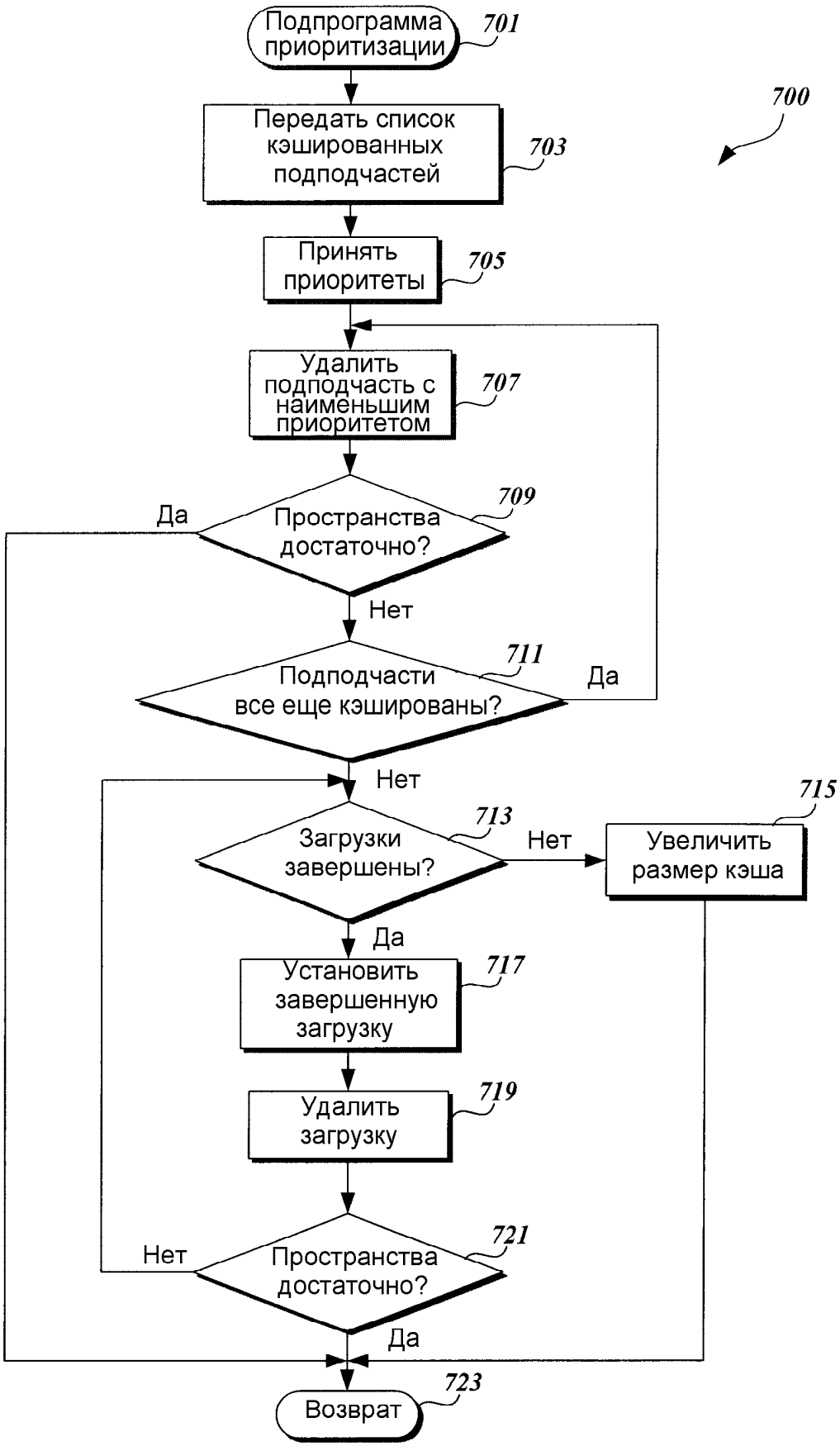
Фиг.4


Начать процедуру обновления клиента
501
500
Запросить/получить список обновлений
503
Сравнить список с имеющейся системой
505
Нет
Обновление необходимо?
507
Да
509
Дифференциальное исправление доступно?
Да
511
Добавить дифференциальное исправление к списку загрузки
Нет
Добавить полное исправление к списку загрузки
513
Да
Дополнительные обновления необходимы?
515
Нет
Запросить список одноранговых узлов
517
Принять список подподчастей/одноранговых узлов
519
Подпрограмма загрузки
521
Собрать и установить исправления
523
Передать результат хеширования кэшированных подподчастей
525
Конец
527

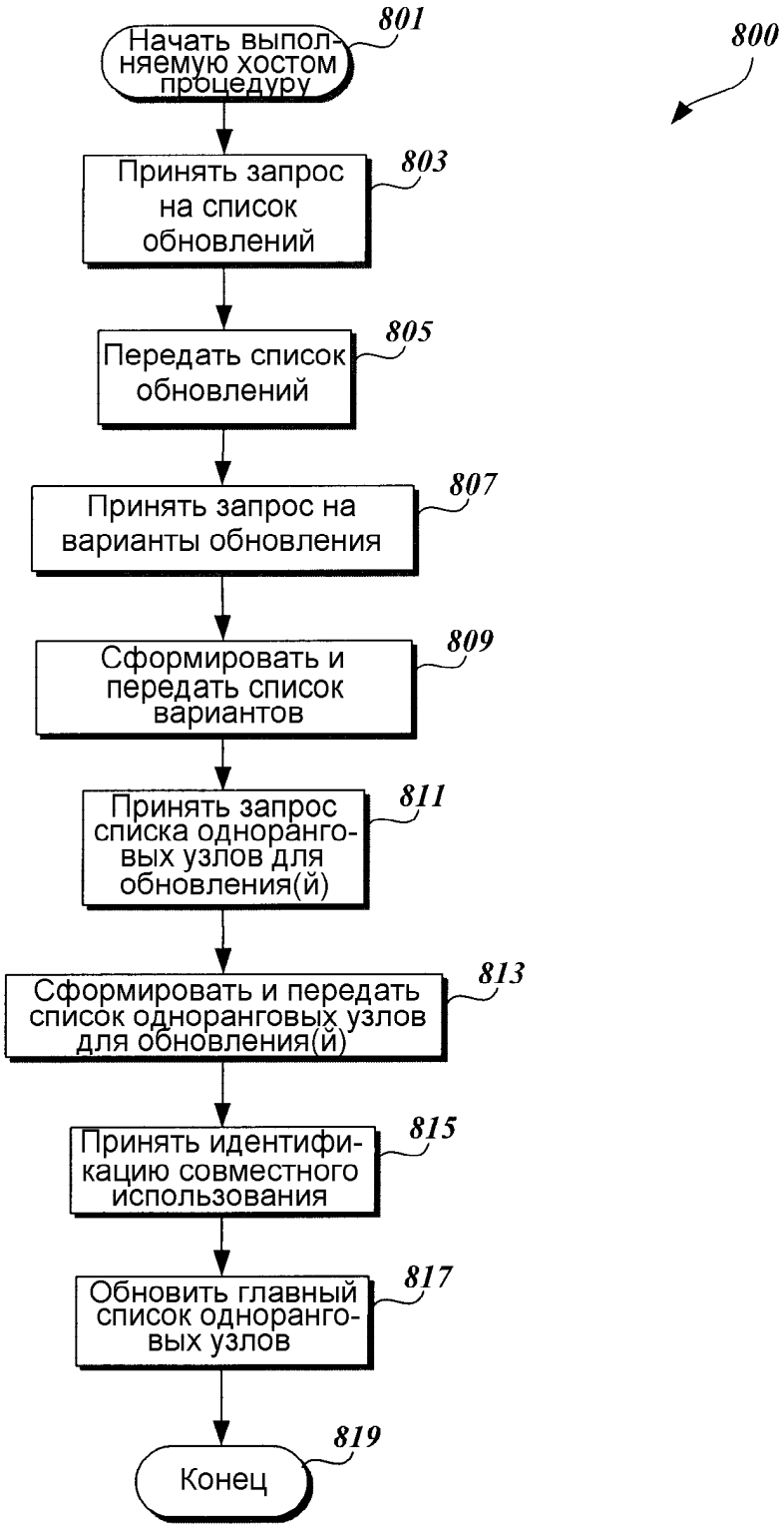
Фиг.5


600
Подпрограмма загрузки
601
Принять список одноранговых узлов
603
Дополнительное пространство в кэше необходимо?
605
Да
Подпрограмма приоритизации
607
Нет
Выбрать файл
609
Выбрать часть
611
Загрузка подчастей части завершена?
613
Нет
Загрузка одной подподчасти завершена?
615
Да
Выбрать подподчасть с наименьшим количеством одноранговых узлов
623
Да
Нет
Загрузка подчастей файла завершена?
629
Нет
Да
Выбрать подподчасть случайным образом
617
Выбрать случайный одноранговый узел
625
Выбрать случайный одноранговый узел
619
Загрузить подподчасть
627
Загрузить подподчасть
621
Загрузка файлов завершена?
631
Нет
Да
Возврат
633

Фиг.6


Подпрограмма приоритизации
701
700
Передать список кэшированных подподчастей
703
Принять приоритеты
705
Удалить подподчасть с наименьшим приоритетом
707
Пространства достаточно?
709
Да
Нет
Подподчасти все еще кэшированы?
711
Да
Нет
Загрузки завершены?
713
Нет
Увеличить размер кэша
715
Да
Установить завершенную загрузку
717
Удалить загрузку
719
Пространства достаточно?
721
Нет
Да
Возврат
723

Фиг.7


800
801
Начать выполняемую хостом процедуру
803
Принять запрос на список обновлений
805
Передать список обновлений
807
Принять запрос на варианты обновления
809
Сформировать и передать список вариантов
811
Принять запрос списка одноранговых узлов для обновления(й)
813
Сформировать и передать список одноранговых узлов для обновления(й)
815
Принять идентификацию совместного использования
817
Обновить главный список одноранговых узлов
819
Конец

Фиг.8