РОССИЙСКАЯ ФЕДЕРАЦИЯ

(19) **RU** (11) **2 432 608** (13) **C2**

(51) МПК
G06F 17/00 (2006.01)

ФЕДЕРАЛЬНАЯ СЛУЖБА
ПО ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ,
ПАТЕНТАМ И ТОВАРНЫМ ЗНАКАМ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ

(21)(22) Заявка: **2008146060/08, 22.05.2007**

(24) Дата начала отсчета срока действия патента:
22.05.2007

Приоритет(ы):
(30) Конвенционный приоритет:
22.05.2006 US 11/438,176

(43) Дата публикации заявки: **27.05.2010** Бюл. № 15

(45) Опубликовано: **27.10.2011** Бюл. № **30**

(56) Список документов, цитированных в отчете о поиске: **RU 2250490 C2, 20.04.2005. US 2003/063134 A1, 03.04.2003. KR 2001/0094095 A, 31.10.2001. US 2002/059465 A1, 16.05.2002. JP 2004094928 A, 25.03.2004.**

(85) Дата начала рассмотрения заявки PCT на национальной фазе: **21.11.2008**

(86) Заявка PCT:
US 2007/012283 (22.05.2007)

(87) Публикация заявки PCT:
WO 2007/139824 (06.12.2007)

Адрес для переписки:
129090, Москва, ул.Б.Спасская, 25, стр.3, ООО "Юридическая фирма Городисский и Партнеры", пат.пов. А.В.Мицу, рег.№ 364

(72) Автор(ы):
УИТРИОЛ Дэниел Б. (US),
ФЕРРЕЙРА Джордж (US)

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

RU 2432608 C2

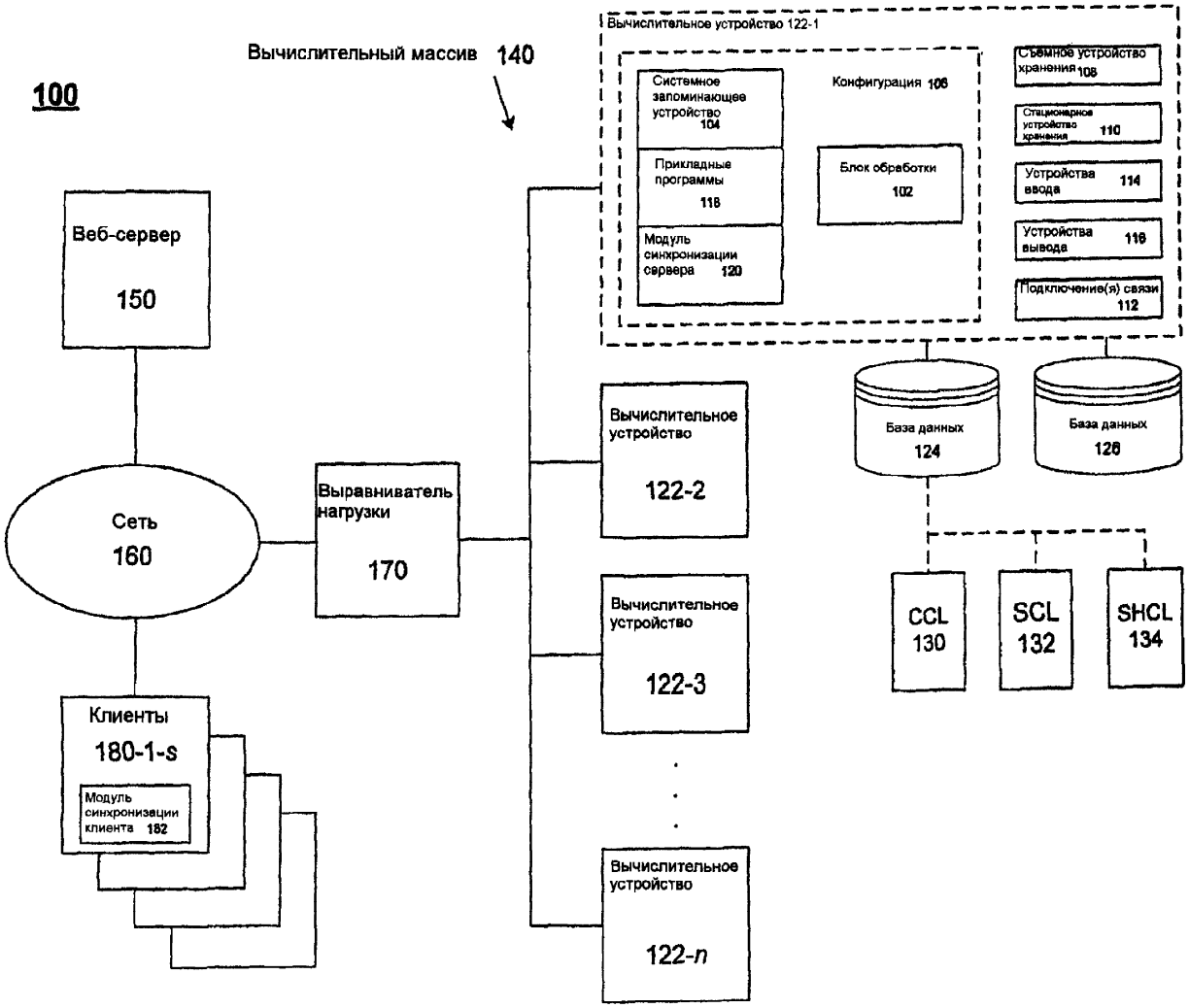
(54) СИНХРОНИЗАЦИЯ СТРУКТУРИРОВАННОГО СОДЕРЖИМОГО ВЕБ-УЗЛОВ

(57) Реферат:

Изобретение относится к средствам для синхронизации структурированного содержимого веб-узлов. Техническим результатом является сокращение времени при синхронизации данных. Устройство включает в себя сервер, имеющий модуль синхронизации сервера для идентификации типов структурированного содержимого, совместно используемого сервером и клиентом, и синхронизирует структурированное содержимое, соответствующее типам структурированного содержимого. Структурированное содержимое, соответствующее типам структурированного содержимого, может быть синхронизировано после идентификации. Таким образом, синхронизация может быть сфокусирована на совместно используемом структурированном содержимом, идентифицированном по типам структурированного содержимого, а не по полному набору данных, сохраненных клиентом и/или сервером. Соответственно, время и ресурсы синхронизации могут быть

уменьшены, тем самым улучшая общую работу устройства и сетевые услуги для пользователя.

3 н. и 13 з.п. ф-лы, 4 ил.



ФИГ.1

RU 2432608 C2

RU 2432608 C2

RUSSIAN FEDERATION

FEDERAL SERVICE
FOR INTELLECTUAL PROPERTY,
PATENTS AND TRADEMARKS

(19) **RU** (11) **2 432 608** (13) **C2**

(51) Int. Cl.
G06F 17/00 (2006.01)

(12) ABSTRACT OF INVENTION

(21)(22) Application: **2008146060/08, 22.05.2007**

(24) Effective date for property rights:
22.05.2007

Priority:
(30) Priority:
22.05.2006 US 11/438,176

(43) Application published: **27.05.2010** Bull. 15

(45) Date of publication: **27.10.2011 Bull. 30**

(85) Commencement of national phase: **21.11.2008**

(86) PCT application:
US 2007/012283 (22.05.2007)

(87) PCT publication:
WO 2007/139824 (06.12.2007)

Mail address:
129090, Moskva, ul.B.Spasskaja, 25, str.3, OOO "Juridicheskaja firma Gorodisskij i Partnery", pat.pov. A.V.Mitsu, reg.№ 364

(72) Inventor(s):
UITRIOL Dehniel B. (US),
FERREJRA Dzhordzh (US)

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

RU 2432608 C2

(54) SYNCHRONISING STRUCTURED WEBSITE CONTENT

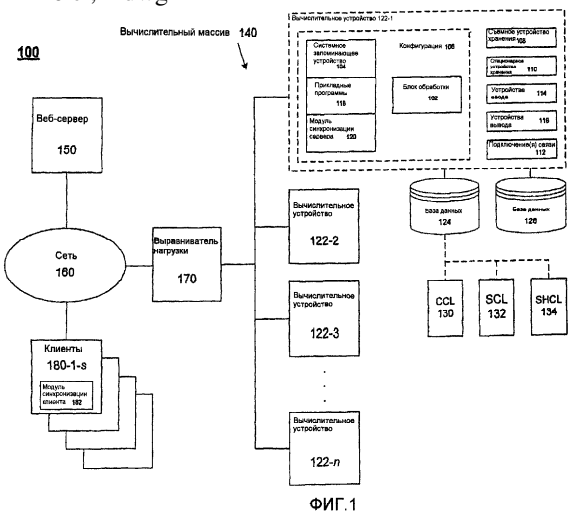
(57) Abstract:

FIELD: information technology.

SUBSTANCE: device includes a server having a server synchronisation module to identify structured content types shared by the server and a client, and synchronise structured content corresponding to the structured content types. Structured content corresponding to structured content types can be synchronised after identification. That way, synchronisation can be focused on shared structured content as identified by the structured content types, rather than the entire set of data stored by the client and/or server. Accordingly, synchronisation time and resources may be reduced, thereby enhancing overall device operations and network services for a user.

EFFECT: faster data synchronisation.

16 cl, 4 dwg



ФИГ.1

Уровень техники

Сетевые прикладные программы в типичном варианте хранят централизованную сетевую базу данных для приложения и/или пользовательских данных. В некоторых случаях устройство может загружать или реплицировать поднабор центральной базы данных из сетевой базы данных и затем отсоединяться от сети. Например, беспроводное портативное устройство может загружать календарь и контактную информацию из центральной базы данных, такой как веб-узел. Если информация, хранимая посредством сетевой базы данных, модифицируется, либо сам реплицированный набор информации реплицируется, может требоваться событие синхронизации для того, чтобы обновлять такие изменения в обоих размещениях данных. Например, предположим, что пользователь добавляет новую встречу в приложение календаря. Когда портативное устройство устанавливает соединение с сетевой базой данных, информация календаря, сохраненная посредством сетевой базы данных, возможно, должна быть обновлена с тем, чтобы отразить модифицированные данные из портативного устройства, и наоборот. По мере того как объем прикладных данных возрастает вместе с числом устройств, пытающихся синхронизироваться с сетевой базой данных, события синхронизации могут стать более затратными в отношении времени и полосы пропускания. Следовательно, улучшенные методики синхронизации могут потребоваться для того, чтобы разрешать эти и другие проблемы.

Сущность изобретения

Данная сущность предусмотрена для того, чтобы в упрощенной форме представить набор идей, которые дополнительно описываются ниже в подробном описании. Эта сущность не предназначена для того, чтобы идентифицировать ключевые признаки или важнейшие признаки заявляемого предмета изобретения, а также не предназначена для того, чтобы быть использованной так, чтобы ограничивать область применения заявляемого предмета изобретения.

Различные варианты осуществления, в общем, могут быть направлены на методики синхронизации структурированного содержимого между физическими и логическими объектами. Более конкретно, различные варианты осуществления могут быть направлены на методики синхронизации между клиентом и сервером. В некоторых вариантах осуществления, например, определенные типы структурированного содержимого, совместно используемые между сервером и клиентом, могут быть идентифицированы до операций синхронизации. Структурированное содержимое, соответствующее типам структурированного содержимого, может быть синхронизировано после идентификации. Таким образом, синхронизация может быть сфокусирована на совместно используемом структурированном содержимом, идентифицированном по типам структурированного содержимого, а не по полному набору данных, сохраненных клиентом и/или сервером. Соответственно, время и ресурсы синхронизации могут быть уменьшены, тем самым улучшая общую работу устройства и сетевые услуги для пользователя.

В одном варианте осуществления клиент может выполнять операции обнаружения или идентификации. Например, сервер может принимать запрос списка содержимого сервера. Сервер может извлекать и отправлять список содержимого сервера, представляющий типы структурированного содержимого, поддерживаемые сервером. Клиент может принимать список содержимого сервера и сравнивать список содержимого сервера со списком содержимого клиента, имеющим значения структурированного содержимого, представляющие типы структурированного

содержимого, поддерживаемые посредством клиента. Клиент может отправлять запрос на синхронизацию и/или список совместно используемого содержимого, чтобы синхронизировать типы структурированного содержимого, поддерживаемые посредством сервера и клиента. Сервер может принимать запрос синхронизации и/или список совместно используемого содержимого и выполнять операции синхронизации соответствующим образом.

В одном варианте осуществления сервер может выполнять операции обнаружения или идентификации. Например, сервер может принимать список содержимого клиента. Сервер может извлекать список содержимого сервера и сравнивать оба списка содержимого. Сервер может генерировать список совместно используемого содержимого, имеющий значения структурированного содержимого, представляющие типы структурированного содержимого, поддерживаемые клиентом и сервером в соответствии со сравнением. Сервер может отправлять список совместно используемого содержимого клиенту. Клиент может принимать список совместно используемого содержимого и отправлять запрос синхронизации, чтобы синхронизировать типы структурированного содержимого в соответствии со списком совместно используемого содержимого. Сервер может принимать запрос на синхронизацию и выполнять операции синхронизации в соответствии с запросом синхронизации. Другие варианты осуществления описываются и приводятся в формуле изобретения.

Краткое описание чертежей

Фиг.1 иллюстрирует примерный вариант осуществления сети.

Фиг.2 иллюстрирует примерный вариант осуществления логической последовательности операций.

Фиг.3 иллюстрирует примерный вариант осуществления первой последовательности сообщений.

Фиг.4 иллюстрирует примерный вариант осуществления второй последовательности сообщений.

Подробное описание изобретения

Фиг.1 иллюстрирует один примерный вариант осуществления сети. Фиг.1 иллюстрирует блок-схему сети 100. Сеть 100 может представлять общую архитектуру сети, подходящую для реализации различных вариантов осуществления. Сеть 100 может содержать несколько элементов. Элемент может содержать любую физическую и логическую структуру, выполненную с возможностью выполнять определенные операции. Каждый элемент может быть реализован как аппаратные средства, программное обеспечение или любая комбинация вышеозначенного, как требуется для данного набора проектных параметров или ограничений производительности. Примеры аппаратных элементов могут включать в себя процессоры, микропроцессоры, схемы, элементы схем (к примеру, транзисторы, резисторы, конденсаторы, индукторы и т.п.), интегральные схемы, специализированные интегральные схемы (ASIC), программируемые логические устройства (PLD), процессоры цифровых сигналов (DSP), программируемая пользователем матричная БИС (FPGA), запоминающие устройства, логические вентили, регистры, полупроводниковые устройства, кристаллы, микросхемы, наборы микросхем и т.п. Примеры программного обеспечения могут включать в себя любые программные компоненты, программы, приложения, вычислительные программы, прикладные программы, системные программы, машинные программы, программное обеспечение операционной системы, промежуточное программное обеспечение,

микропрограммное обеспечение, программно-реализованные модули, процедуры, подпрограммы, функции, способы, программные интерфейсы, интерфейсы прикладного программирования (API), наборы инструкций, вычислительный код, компьютерный код, сегменты кода, сегменты компьютерного кода, слова, значения, символы или любую комбинацию вышеозначенного. Хотя сеть 100, как показано на фиг.1, имеет ограниченное число элементов в определенной топологии, можно принимать во внимание, что сеть 100 может включать в себя больше или меньше элементов в альтернативных топологиях, как требуется для данной реализации. Варианты осуществления не ограничены этим контекстом.

Как показано на фиг.1, например, сеть 100 может содержать различные элементы, такие как вычислительный массив 140, сервер 150, сеть 160, выравниватель 170 нагрузки и клиентские устройства 180-1-s. В одном варианте осуществления, например, сервер 150 может быть реализован как веб-сервер. Веб-сервер может содержать вычислительное устройство, допускающее прием запросов по протоколу передачи гипертекста (HTTP) от клиентских устройств (к примеру, клиентов 180-1-s и/или вычислительных устройств 122-1-n), чтобы обслуживать веб-страницы. Веб-страницы - это в типичном варианте документы, сгенерированные с помощью определенной формы языка разметки, такого как язык гипертекстовой разметки (HTML), расширяемый язык разметки (XML), расширяемый язык гипертекстовой разметки (XHTML), язык разметки Microsoft Word® (WordML) и т.п. Сеть 160 может включать в себя пакетную сеть, использующую один или более Интернет-протоколов, такой как протокол управления передачей и Интернет-протокол (TCP/IP). Выравниватель 170 нагрузки может содержать устройство для того, чтобы назначать рабочие нагрузки набору сетевых вычислительных серверов (к примеру, вычислительному массиву 140) таким образом, что вычислительные ресурсы используются эффективным способом. Выравниватель 170 нагрузки может быть реализован с помощью, например, компьютера, сервера, виртуального сервера, сетевого устройства и т.п.

В различных вариантах осуществления сеть 100 может включать в себя различные вычислительные устройства. В одном варианте осуществления, например, вычислительный массив 140 может включать в себя несколько вычислительных устройств 122-1-n. Аналогично, клиентские устройства 180-1-s также могут быть реализованы как различные типы вычислительных устройств. Примеры вычислительных устройств могут включать в себя, но не обязательно ограничены этим, компьютер, вычислительную систему, вычислительную подсистему, рабочую станцию, сервер, веб-сервер, виртуальный сервер, персональный компьютер (PC), настольный компьютер, дорожный компьютер, компактный дорожный компьютер, портативный компьютер, карманный компьютер, персональное цифровое устройство (PDA), мобильное вычислительное устройство, сотовый телефон, комбинацию сотовый телефон/PDA, цифровое видеоустройство (к примеру, цифровую камеру, видеокамеру или портативную видеокамеру), цифровое аудиоустройство (к примеру, MP3-проигрыватель), пейджер односторонней связи, пейджер двусторонней связи, виртуальные экземпляры реализации любого из предыдущих примеров, а также другое электронное, электромеханическое или электрическое устройство. Варианты осуществления не ограничены этим контекстом.

В одном варианте осуществления, например, вычислительный массив 140 может быть реализован как пул серверов, при этом вычислительные устройства 122-1-n каждое представляют сервер, виртуальный сервер, виртуальную машину,

одноплатный компьютер (SBC), блейд-сервер, веб-сервер и т.п. Пул серверов - это в типичном варианте набор вычислительных серверов, обычно поддерживаемых организацией для того, чтобы удовлетворять серверные потребности за пределами возможностей одной машины. Зачастую пулы серверов имеют первичный и теневой сервер, назначенные одной задаче, с тем чтобы в случае сбоя первичного сервера теневой сервер подхватывал функции первичного сервера. Пулы серверов, как правило, используются для того, чтобы предоставлять серверы веб-хостинга. Серверы веб-хостинга - это тип серверов Интернет-хостинга, которые предоставляют частным лицам и организациям онлайновые системы для хранения информации, изображения, видео, аудио, текста, анимаций, фильмов, изображений или любой другой формы веб-содержимого, доступного по всемирной паутине (WWW, или "веб"). Веб-хосты - это компании, которые предоставляют место на сервере, которым они владеют, для использования своими клиентами, а также предоставления возможностей подключения к Интернету, в типичном варианте в центре обработки данных. Веб-хосты также могут предоставлять место для центра обработки данных и возможности подключения к Интернету для серверов, которыми они не владеют, чтобы размещать их в центре обработки данных.

Фиг.1 дополнительно иллюстрирует более подробную блок-схему вычислительного устройства 122-1. Вычислительное устройство 122-1 может быть примером любого из вычислительных устройств 122-1-n. Более того, вычислительное устройство 122-1 также может быть примером любого из клиентских устройств 180-1-s. В своей самой базовой конфигурации 106 вычислительное устройство 122-1 в типичном варианте включает в себя, по меньшей мере, один блок 102 обработки и запоминающее устройство 104. Запоминающее устройство 104 может быть реализовано с помощью любых машиночитаемых или компьютерночитаемых носителей, допускающих хранение данных, включая энергозависимое и энергонезависимое запоминающее устройство. Например, запоминающее устройство 104 может включать в себя постоянное запоминающее устройство (ROM), оперативное запоминающее устройство (RAM), динамическое RAM (DRAM), DRAM с двойной скоростью передачи данных (DDRAM), синхронное DRAM (SDRAM), статическое RAM (SRAM), программируемое ROM (PROM), стираемое программируемое ROM (EPROM), электрически стираемое программируемое ROM (EEPROM), флэш-память, полимерное запоминающее устройство, такое как запоминающее устройство на сегнетоэлектрических полимерах, запоминающее устройство на аморфных полупроводниках, фазосдвигающее или сегнетоэлектрическое запоминающее устройство, запоминающее устройство по технологии кремний-окись-нитрид-окись-кремний (SONOS), магнитные или оптические карты либо любой другой тип носителей, подходящий для хранения информации. Как показано на фиг.1, запоминающее устройство 104 может сохранять различные программно-реализованные программы, такие как одна или более прикладных программ 118, модуль 120 синхронизации сервера и сопутствующие данные.

Вычислительное устройство 122-1 также может иметь дополнительные признаки и/или функциональность помимо конфигурации 106. Например, вычислительное устройство 122-1 может включать в себя съемное устройство 108 хранения и стационарное устройство 110 хранения, которые также могут содержать различные типы машиночитаемых или компьютерночитаемых носителей, как описано выше. Вычислительное устройство 122-1 также может иметь одно или более устройств 114 ввода, например клавиатуру, мышь, перо, устройство речевого ввода, устройство

сенсорного ввода и т.п. Одно или более устройств 116 вывода, например дисплей, громкоговорители, принтер и т.п., также могут быть включены в вычислительное устройство 122-1.

Вычислительное устройство 122-1 дополнительно может включать в себя одно или более подключений 112 связи, которые позволяют вычислительному устройству 122 обмениваться данными с другими устройствами. Подключения 112 связи могут включать в себя различные типы стандартных элементов связи, такие как один или более интерфейсов связи, сетевых интерфейсов, сетевых интерфейсных карт (NIC), радиостанций, беспроводных передающих устройств/приемных устройств (приемо-передающих устройств), проводных и/или беспроводных носителей связи, физических разъемов и т.п. Среда связи в типичном варианте содержит машиночитаемые инструкции, структуры данных, программные модули или другие данные в модулированном сигнале данных, таком как несущее колебание или другой механизм распространения, и включает в себя любой носитель для доставки информации. Термин "модулированный сигнал данных" означает сигнал, который имеет одну или более из его характеристик, установленных или изменяемых таким образом, чтобы кодировать информацию в сигнале. В качестве примера, а не ограничения, среда связи включает в себя проводную среду связи и беспроводную среду связи. Примеры проводных сред связи могут включать в себя провод, кабель, металлические выводы, схемные печатные платы (PCB), системные платы, многовходовые системы коммутации, полупроводниковый материал, витую пару, коаксиальный кабель, оптоволокно, распространяемый сигнал и т.п. Примеры беспроводных сред связи могут включать в себя акустические, в радиочастотном (RF) спектре, инфракрасные и другие беспроводные среды. Термины машиночитаемые носители и компьютерночитаемые носители при использовании в данном документе предназначены для того, чтобы включать в себя носители хранения и среду связи.

Каждое из вычислительных устройств 122-1-n может включать в себя несколько баз данных. Как подробно показано относительно вычислительного устройства 122-1, вычислительное устройство 122-1 может быть подключено к базам 124, 126 данных. Каждая база данных может быть выполнена с возможностью сохранять различные типы данных для вычислительного устройства 122-1, прикладных программ 118, модуля 120 синхронизации сервера и т.п. В одном варианте осуществления, например, база 124 данных может сохранять список 130 содержимого клиента, список 132 содержимого сервера, список 134 совместно используемого содержимого и т.п. Список 130 содержимого клиента может включать в себя значения структурированного содержимого, представляющие типы структурированного содержимого, поддерживаемые клиентом, например одним или более клиентами 180-1-s. Список 132 содержимого сервера может включать в себя значения структурированного содержимого, представляющие типы структурированного содержимого, поддерживаемые сервером, например одним или более серверами 122-1-n. Список 134 совместно используемого содержимого может включать в себя значения структурированного содержимого, представляющие типы структурированного содержимого, совместно используемые одним или более клиентами 180-1-s и одним или более серверами 122-1-n. Другие типы данных и баз данных могут быть реализованы в вычислительном устройстве 122-1, и варианты осуществления не ограничены в этом контексте.

В общем, при работе сеть 100 может упрощать операции синхронизации между одной или более прикладными программами, сохраняемыми или приводимыми в

исполнение посредством одного или более вычислительными устройствами 122-1-n вычислительного массива 140, и одной или более прикладными программами, сохраняемыми или приводимыми в исполнение посредством одного или более клиентскими устройствами 180-1-s. Допустим, каждое вычислительное устройство 122-1-n реализует одну или более прикладных программ 118, к примеру, серверное веб-приложение. Примеры серверных веб-приложений могут включать в себя Windows® SharePoint® Services (WSS) версия 2.0, версия 3.0 и варианты, созданные Microsoft® Corporation (вместе упоминаемые в данном документе как "приложение WSS" или "приложение SharePoint"). WSS версия 3.0 - это комплект интегрированных прикладных программ, который предоставляет базовые услуги групповой работы, функциональность веб-порталов и сетей интранет для различных клиентов, таких как, к примеру, клиенты 180-1-s. Пользовательские данные или данные содержимого для WSS версия 3.0 могут быть сохранены посредством баз 124, 126 данных. Более того, одно или более вычислительных устройств 122-1-n могут реализовывать стороннее приложение, встроенное поверх WSS версия 3.0, такое как, к примеру, SharePoint Portal Server (SPS) 2003 или Office SharePoint Services (OSS) 2006. Хотя некоторые варианты осуществления могут быть описаны со ссылкой на прикладную программу WSS и прилагаемые сторонние программы в качестве примера, следует принимать во внимание, что любая прикладная программа, приводимая в исполнение посредством одного или нескольких вычислительных устройств, может быть синхронизирована с помощью методик синхронизации, описанных в данном документе. Варианты осуществления не ограничены в этом контексте.

WSS версия 3.0 предоставляет платформу для нескольких пользователей, чтобы эффективно совместно использовать данные. Например, пользователи могут быть организованы в группы, причем каждая группа имеет совместно используемый календарь, список контактов, список задач, электронную почту, форумы, документы и т.п. В попытке повысить удобство использования таких совместно используемых данных различные элементы сети 100 могут быть реализованы как периодически синхронизируемая система баз данных (ISDB). ISDB позволяет различным физическим или логическим клиентским объектам загружать или реплицировать части информации из баз 124, 126 данных на клиентское устройство, такое как, к примеру, клиентское устройство 180-1. Допустим, что клиентское устройство 180-1 включает в себя такое приложение, как Microsoft Outlook®, созданное Microsoft Corporation. Microsoft Outlook может импортировать совместно используемые данные с сервера 122-1, поддерживаемого с помощью WSS версия 3.0. В результате, пользователь может иметь портативное устройство, такое как карманное вычислительное устройство, приводящее в исполнение локальную версию Microsoft Outlook, и загружать части совместно используемых данных с сервера 122-1. Клиентское устройство 180-1 может модифицировать локально реплицированные данные без постоянного подключения к серверу 122-1 (к примеру, оффлайн), например, посредством добавления встреч, удаления контакта, модификации задачи, создания примечания, обновления временной шкалы проекта и т.п. Между тем, другие пользователи также могут модифицировать локально реплицированные данные из баз 124, 126 данных. Чтобы поддерживать целостность данных, клиентское устройство 180-1, возможно, должно периодически или время от времени синхронизироваться с сервером 122-1, чтобы отразить изменения в любом структурированном содержимом или данных, совместно используемых обоими устройствами.

Чтобы выполнять операции синхронизации, каждое клиентское устройство 180-1-s может включать модуль 182 синхронизации клиента, а каждый сервер 122-1-n может включать модуль 120 синхронизации сервера. Модули 120, 182 синхронизации могут иметь варьирующиеся уровни интеграции, чтобы позволить серверу 122 и клиентскому устройству 180, соответственно, синхронизировать структурированное содержимое или наборы данных, поддерживаемые обоими устройствами. В одном варианте осуществления, например, клиентское устройство 180 может включать в себя клиентское приложение Microsoft Outlook, интегрированное на некотором уровне с приложением WSS, приводимым в исполнение посредством сервера 122. Эта интеграция может включать в себя синхронизацию данных Microsoft Outlook и приложением WSS. Например, приложение WSS может включать в себя совместно используемый календарь, контакты, задачи, электронную почту и форумы, сохраненные в приложении WSS. Различные варианты осуществления позволяют задавать пользовательскую модель, драйверы, технические требования, схемы данных и инфраструктуру синхронизации для пользователей Microsoft Outlook, чтобы взаимодействовать с данными приложения WSS. Хотя некоторые варианты осуществления могут быть описаны как синхронизация структурированного содержимого между Microsoft Outlook и приложением WSS только в качестве примера, можно принимать во внимание, что синхронизация структурированного содержимого может быть реализована для любых прикладных программ и вычислительных платформ, требуемых для данного набора проектных параметров и ограничений производительности. Варианты осуществления не ограничены в этом контексте.

Имеется несколько проектных соображений при попытке синхронизировать два или более приложений и/или устройство, таких как сервер 122-1 и клиентское устройство 180-1. Например, чтобы использовать веб-узел в качестве центрального репозитория данных, данные должны быть синхронизированы с другими приложениями быстро при создании минимальной нагрузки на сервер веб-узла. Дополнительно, пользователи зачастую заинтересованы только в небольшой области всех данных, и, следовательно, пользователям требуются легко обнаруживаемые места, чтобы начать операции синхронизации для важных интересующих данных. Помимо этого, различные клиенты синхронизации могут допускать синхронизацию только с конкретными типами данных с сервера. Аналогично, некоторые клиенты синхронизации могут поддерживать типы или значения данных, не поддерживаемые сервером веб-узла, тогда как сервер веб-узла может поддерживать типы или значения данных, не поддерживаемые клиентами синхронизации. В другом примере веб-узел и клиент синхронизации могут выполняться на разных языках. В еще одном другом примере несколько клиентов синхронизации потенциально могут обновлять одни и те же данные, не будучи подключенными к серверу, и поэтому должны быть предприняты меры для того, чтобы обеспечить то, что данные не теряются в ходе событий синхронизации. В еще одном другом примере пользователи, возможно, должны синхронизироваться с одним веб-узлом внутри как частных сетей, так и сетей общего пользования, таких как Интернет. В еще одном примере клиенты, синхронизирующие содержимое от сервера, не могут точно прогнозировать то, как часто изменяется содержимое, что приводит к дополнительной нагрузке на сервер или устаревшему содержимому на синхронизированном клиенте. Наконец, администраторы серверов хотят иметь возможность контролировать нагрузку, создаваемую на сервер через синхронизирующихся клиентов.

В попытке разрешить эти и другие проблемы сервер 122 и/или клиентское

устройство 180 может реализовать улучшенные методики синхронизации с помощью структурированного содержимого. Структурированное содержимое может означать любой тип мультимедийного содержимого (к примеру, текст, аудио, видео, изображения, рисунки, анимацию, символы, знаки, числа, значки и т.п.), имеющий один или более заданных структурных параметров, таких как ширина, длина, размер в битах, размер в байтах, синтаксис, поля, значения, коды, флаги и т.п. Структурированное содержимое может включать в себя прикладные данные и/или пользовательские данные, чтобы поддерживать любое число прикладных программ, таких как Microsoft Outlook, Microsoft Access®, Microsoft FrontPage®, Microsoft OneNote®, Microsoft PowerPoint®, Microsoft Word, Microsoft Visio®, приложение WSS, приложение SPS и т.п. В одном варианте осуществления, например, структурированное содержимое может включать в себя прикладные данные и/или пользовательские данные для Microsoft Outlook, такие как один или более элементов из списка календаря, списка задач, списка контактов, списка примечаний или памяток, список почтовых сообщений и т.п. Модуль 120 синхронизации сервера и модуль 182 синхронизации клиента могут быть интегрированы для того, чтобы идентифицировать типы структурированного содержимого, совместно используемые между сервером 122 и клиентом 180. Информация структурированного содержимого, соответствующая типам структурированного содержимого, совместно используемым между сервером 122 и клиентом 180, затем может быть синхронизирована.

Выполнение операций синхронизации с помощью структурированного содержимого позволяет предоставлять несколько преимуществ в сравнении с традиционными методиками синхронизации. В различных вариантах осуществления, например, инициирование операций синхронизации между устройствами может быть усовершенствовано рядом способов. Например, сетевой адрес и/или гиперссылка может быть создана для клиента синхронизации, чтобы инициировать операции синхронизации на основе определенных типов содержимого, поддерживаемых сервером и/или клиентом. Различные гиперссылки могут быть созданы для того, чтобы инициировать операции синхронизации для различных областей содержимого. В другом примере операции синхронизации могут быть инициированы по-другому на основе объема синхронизируемого содержимого (к примеру, синхронизация только заголовков в сравнении с синхронизацией полных наборов содержимого). В еще одном примере блок управления протоколами может быть использован для того, чтобы инициировать операции синхронизации из веб-обозревателя.

После того как операции синхронизации инициированы, различные варианты осуществления позволяют улучшить то, как содержимое фактически синхронизируется. Например, серверный журнал изменений может быть использован для того, чтобы синхронизировать только измененное содержимое. Маркер может передаваться обратно и вперед с тем, чтобы получить только новые изменения, сделанные с момента предыдущего события синхронизации. В другом примере операции синхронизации могут поддерживать только отдельно запрошенные или заданные поля в структурированной схеме. В еще одном другом примере операции синхронизации могут поддерживать различные иерархические области, такие как созданные в иерархиях папок. В еще одном другом примере представления синхронизации могут быть сгенерированы так, чтобы отображать фильтрованный набор содержимого, определенный посредством различных значений полей. В еще одном другом примере некоторые варианты осуществления могут поддерживать очень большие наборы данных с помощью методик страничной организации. В

другом примере некоторые варианты осуществления могут выполнять обнаружение и автоматическое разрешение конфликтов с помощью архива версий. В еще одном примере некоторые варианты осуществления могут поддерживать синхронизацию вложений с помощью E-Tags для разрешения конфликтов. В другом примере некоторые варианты осуществления могут поддерживать несовпадения схемы между синхронизацией веб-узла и клиентом с помощью контейнера свойств. В дополнительном примере некоторые варианты осуществления могут поддерживать несовпадения языка сервера/клиента с помощью преобразования значений полей. В еще одном другом примере некоторые варианты осуществления могут поддерживать считывания и обновления с помощью шаблона считывания/отправки/обновления. Наконец, некоторые варианты осуществления могут предоставлять клиенту альтернативные унифицированные указатели ресурса (URL-адреса), с которыми он может синхронизировать для того, чтобы осуществлять доступ к веб-узлу из внутренних частных сетей или внешних сетей общего пользования.

Чтобы дополнительно улучшить операции синхронизации, различные варианты осуществления позволяют разрешать вопросы синхронизации, касающиеся того, когда синхронизировать клиентское устройство с веб-узлом. Например, некоторые варианты осуществления позволяют администрировать клиентскую нагрузку и/или серверную нагрузку за счет предоставления рекомендованных интервалов синхронизации клиенту синхронизации (к примеру, клиент синхронизируется каждые P минут). Рекомендованные интервалы синхронизации могут представлять минимальный интервал синхронизации, максимальный интервал синхронизации или некоторое значение между ними. Рекомендованные интервалы синхронизации могут быть основаны на данных изменения архивного содержимого, сохраненных на сервере (к примеру, сетчатая синхронизация), или на основе результатов предыдущих запросов синхронизации от клиента (к примеру, экспоненциального спада).

Чтобы синхронизировать структурированное содержимое с сервером 122, клиентское устройство 180 может использовать элемент управления ActiveX *SharePoint.StssyncHandler*. Этот элемент управления устанавливается на клиентском устройстве 180 вместе с Microsoft Outlook и указывает то, установлено ли на клиентском устройстве 180 и может ли Microsoft Outlook синхронизировать конкретный тип списка. В WSS версия 3.0, например, типы списка, которые могут быть синхронизированы, включают в себя список календаря, список контактов, список документов, список задач, список тем форума, а также другие.

Операции в вышеописанных вариантах осуществления дополнительно могут быть описаны со ссылкой на последующие чертежи и прилагаемые чертежи. Некоторые из чертежей могут включать в себя логическую последовательность операций. Хотя эти чертежи, представленные в данном документе, могут включать в себя конкретную логическую последовательность операций, можно принимать во внимание, что логическая последовательность операций просто предоставляет пример того, как общая функциональность, описанная в данном документе, может быть реализована. Дополнительно, данная логическая последовательность операций не обязательно должна приводиться в исполнение в представленном порядке, если не указано иное. Помимо этого, данная логическая последовательность операций может быть реализована посредством аппаратного элемента, программного элемента, приводимого в исполнение посредством процессора, либо любой комбинации вышеозначенного. Варианты осуществления не ограничены этим контекстом.

Фиг.2 иллюстрирует один вариант осуществления логической последовательности

операций. Фиг.2 иллюстрирует логическую последовательность 200 операций. Логическая последовательность 200 операций может быть примером операций, приводимых в исполнение посредством одного или более вариантов осуществлений, описанных в данном документе, таких как сеть 100, вычислительные устройства 122-1-n и/или клиентские устройства 180-1-s. В некоторых вариантах осуществления, например, определенные структурированные типы содержимого, совместно используемые сервером и клиентом, могут быть идентифицированы перед операциями синхронизации на этапе 202. Структурированное содержимое, соответствующее типам структурированного содержимого, далее может быть синхронизировано после того, как идентифицировано, на этапе 204. Таким образом, синхронизация может быть сфокусирована на совместно используемом структурированном содержимом, идентифицированном посредством типов структурированного содержимого, а не на полном наборе данных, хранимых клиентом и/или сервером. Соответственно, время и ресурсы синхронизации могут быть уменьшены, тем самым улучшая общую работу устройства и сетевые услуги для пользователя.

Фиг.3 иллюстрирует один вариант осуществления первой последовательности сообщений. Фиг.3 иллюстрирует последовательность 300 сообщений, в которой клиент выполняет операции идентификации. Как показано на фиг.3, клиентское устройство 180 может отправить сообщение 302 на сервер 122, при этом сообщение 302 содержит запрос списка содержимого сервера. Сервер 122 может принимать сообщение 302 и извлекать список 132 содержимого сервера из базы 124 данных. Сервер 122 может отправлять сообщение 304 в клиентское устройство 180, при этом сообщение 304 содержит список 132 содержимого сервера со значениями структурированного содержимого, представляющими типы структурированного содержимого, поддерживаемые посредством сервера 122. Клиентское устройство 180 может принимать список 132 содержимого сервера и извлекать список 130 содержимого клиента. Список 130 содержимого клиента может включать в себя значения структурированного содержимого, представляющие типы структурированного содержимого, поддерживаемые клиентом. Клиентское устройство 180 может сравнивать список 132 содержимого сервера со списком 130 содержимого клиента, чтобы находить совпадающие типы содержимого. Клиентское устройство 180 может генерировать список 134 совместно используемого содержимого и отправлять сообщение 306 на сервер 122. Сообщение 306 может содержать запрос синхронизации и/или список 134 совместно используемого содержимого. Сервер 122 может принимать сообщение 306, и сервер 122 и клиентское устройство 180 могут начинать операции синхронизации, чтобы синхронизировать типы структурированного содержимого, поддерживаемые посредством сервера 122 и клиента 180, как указано посредством стрелки 308.

Фиг.4 иллюстрирует один вариант осуществления второй последовательности сообщений. Фиг.4 иллюстрирует последовательность 400 сообщений, в которой сервер выполняет операции идентификации. Как показано на фиг.4, сервер 122 может принимать список 130 содержимого клиента через сообщение 402. Сервер 122 может извлекать список 132 содержимого сервера и сравнивать оба списка содержимого. Сервер 122 может генерировать список 134 совместно используемого содержимого, имеющий значения структурированного содержимого, представляющие типы структурированного содержимого, поддерживаемые клиентским устройством 180 и сервером 122 в соответствии со сравнением. Сервер 122 может отправлять список 134 совместно используемого содержимого в клиентское устройство 180 через

сообщение 404. Клиентское устройство 180 может принимать список 134 совместно используемого содержимого и отправлять запрос синхронизации, чтобы синхронизировать типы структурированного содержимого в соответствии со списком 134 совместно используемого содержимого через сообщение 406. Сервер 122 может принимать запрос синхронизации, и сервер 122 и клиентское устройство 180 могут выполнять операции синхронизации в соответствии с запросом синхронизации, указанным посредством стрелки 408.

В различных вариантах осуществления инициирование операций синхронизации между устройствами может быть улучшено рядом способов. Например, сетевой адрес и/или гиперссылка может быть создана для клиента синхронизации, чтобы инициировать операции синхронизации на основе определенных типов содержимого, поддерживаемых сервером и/или клиентом. Различные гиперссылки могут быть созданы для того, чтобы инициировать операции синхронизации для различных областей содержимого. В одном варианте осуществления модуль 120 синхронизации сервера может генерировать гиперссылку, чтобы отправлять запрос синхронизации, по меньшей мере, с одним типом структурированного содержимого. Как результат, пользователь может выборочно синхронизировать определенные типы структурированного содержимого с сервером 122 прозрачным образом.

В различных вариантах осуществления операции синхронизации могут быть инициированы по-разному на основе объема синхронизированного содержимого. В одном варианте осуществления, например, модуль 120 синхронизации сервера может определять объем структурированного содержимого для синхронизации. Модуль 120 синхронизации сервера может задавать параметры синхронизации на основе определенного объема. Сервер 122-1 может отправлять информацию синхронизации клиенту 180-1 в соответствии с параметрами синхронизации. Например, предположим, что типом структурированного содержимого являются почтовые сообщения. Если определенный объем - это большой объем почтовых сообщений, которые должны быть синхронизированы между сервером 122-1 и клиентом 180-1, то сервер 122-1 может задать параметр синхронизации так, чтобы указать, что только заголовки почтовых сообщений должны быть первоначально синхронизированы, а не все содержимое почтовых сообщений. Наоборот, если определенный объем - это небольшой объем почтовых сообщений, то сервер 122-1 может задать параметр синхронизации так, чтобы указать то, что все содержимое почтовых сообщений должно быть синхронизировано с клиентом 180-1. В другом примере допустим, что типом структурированного содержимого являются документы. Если определенный объем - это большой объем документов, которые должны быть синхронизированы между сервером 122-1 и клиентом 180-1, то сервер 122-1 может задать параметр синхронизации так, чтобы указать, что только определенные части каждого документа должны быть первоначально синхронизированы, а не вся совокупность каждого документа. Наоборот, если определенный объем - это небольшой объем документов, то сервер 122-1 может задать параметр синхронизации так, чтобы указать, что все содержимое каждого документа должно быть синхронизировано с клиентом 180-1. Эта методика также может быть применена посредством анализа общего размера документа, причем больший документ имеет только часть, отправляемую за один раз, а меньший документ полностью отправляется в одной транзакции. Это всего несколько примеров, и другие характеристики типа структурированного содержимого могут быть использованы для того, чтобы задавать параметр синхронизации.

В различных вариантах осуществления блок управления протоколами может быть использован для того, чтобы инициировать операции синхронизации из веб-обозревателя. После того как типы совместно используемого структурированного содержимого идентифицированы, веб-обозреватель на клиенте 180-1 может предоставить один или более URL-адресов для того, чтобы инициировать операции синхронизации, как описано ранее. Чтобы добиться этого, модуль 182 синхронизации клиента может использовать блок управления протоколами для того, чтобы отправлять различные запросы синхронизации на сервер 122-1.

После того как операции синхронизации инициированы, различные варианты осуществления позволяют улучшить то, как содержимое фактически синхронизируется. Например, серверный журнал изменений может быть использован для того, чтобы синхронизировать только измененное содержимое. В одном варианте осуществления, например, модуль 120 синхронизации сервера может принимать запрос синхронизации, чтобы синхронизировать определенный тип структурированного содержимого. Модуль 120 синхронизации сервера может определять то, модифицировано ли структурированное содержимое, указанное посредством типа структурированного содержимого, с помощью журнала изменений. Сервер 122-1 может отправлять информацию синхронизации клиенту на основе этого определения.

Для этой синхронизации чтения/записи заданная методика требуется для Microsoft Outlook, чтобы синхронизировать изменения из приложения WSS. Это может быть достигнуто с помощью журнала изменений, прилагаемого к API веб-служб. Например, WSS версия 3.0 поддерживает журнал изменений по виртуальным серверам, узлам и спискам. Это предоставляет возможность Microsoft Outlook опрашивать конкретно на предмет добавленных, отредактированных, удаленных, переименованных и перемещенных элементов. Например, веб-служба может быть реализована, чтобы дать возможность Microsoft Outlook выполнять пакетно операцию "get" элементов журнала изменений.

Различные варианты осуществления могут определять то, были ли несколько версий структурированного содержимого модифицированы независимо, с помощью архива версий. Некоторые варианты осуществления могут выполнять обнаружение и автоматическое разрешение конфликтов с помощью архива версий. Архив версий - это механизм, который используется для того, чтобы автоматически разрешать конкретный класс конфликтов синхронизации в равноправных окружениях. В частности, он может автоматически разрешать конфликты, которые являются результатом согласованных последовательных изменений элемента.

В качестве примера, предположим, что первый пользователь синхронизирует элемент с первым клиентским устройством 180-1, таким как настольный компьютер, используемый в офисе. Первый пользователь выполняет несколько изменений в элемент, но не копирует эти изменения обратно на сервер 122-1. Первый пользователь синхронизирует первое клиентское устройство 180-1 со вторым клиентским устройством 180-2, таким как PDA, и уходит из офиса домой. Когда первый пользователь приходит домой, первый пользователь синхронизирует второе клиентское устройство 180-2 с третьим клиентским устройством 180-3, таким как домашний PC. Первый пользователь вносит дополнительные изменения в элемент перед его синхронизацией с третьего клиентского устройства 180-3 на сервер 122-1. На следующий день первый пользователь приходит на работу и синхронизирует первое клиентское устройство 180-1 с сервером 122-1. Первое клиентское устройство 180-1

пытается отправить обновленный элемент, который оно имеет, на сервер 122-1, но сервер 122-1 обнаруживает изменение в элементе с момента предыдущей синхронизации с первым клиентским устройством 180-1 и, следовательно, провоцирует конфликт. Тем не менее, конфликт не является реальным конфликтом, поскольку обновленная версия элемента, размещающаяся на сервере 122-1, уже включает в себя изменение, сделанное на первом клиентском устройстве 180-1, когда первый пользователь синхронизировал третье клиентское устройство 180-3 с сервером 122-1. Некоторые варианты осуществления используют архив версий для того, чтобы обнаруживать такие сценарии, как этот, и автоматически разрешать конфликт.

В вышеописанном сценарии одного номера версии для элемента может быть недостаточно для того, чтобы разрешить конфликт. Например, что, если первый пользователь не синхронизировался с сервером 122-1 из третьего клиентского устройства 180-3. Наоборот, предположим, что второй пользователь внес несвязанное изменение в элемент. В этом случае реально имеется конфликт. Одного номера версии недостаточно для того, чтобы различать два сценария. Методика архивов версий сохраняет дополнительную информацию для того, чтобы помочь разрешить эту проблему.

В частности, методика архива версий может быть использована для того, чтобы определять то, есть ли несколько версий элемента (к примеру, версия A и версия B), и включает ли одна версия изменения, сделанные в другой версии (к примеру, включает ли версия B изменения, которые сделаны в версии A). В этом случае анализ текста рассматриваемого документа может становиться трудным и неточным. Например, рассмотрим сценарий, в котором версия B фактически включала в себя изменения от версии A, тем не менее, одно из изменений в версии B заключалось в том, чтобы удалить строку, добавленную в версию A.

Методика архива версий также может быть использована для того, чтобы записать список авторов, и увеличения номера изменения для каждого из элементов, которое указывает то, какое изменение в последовательности они сделали. "Авторами" в данном случае являются компьютеры, представляемые посредством глобальных пользовательских идентификаторов (GUID). Свойство архива версий, как результат, может содержать конкатенированный список GUID и номера изменений следующим образом:

{199A4AEA-A573-40CB-BB3C-7A66C0375104:1,201B4AEA-A573-40CB-BB3C-7A66C0375104:2,185D4AEA-A573-40CB-BB3C-7A66C0375104:3}

Каждый раз, когда конкретное вычислительное устройство редактирует данный элемент, соответствующий модуль синхронизации (к примеру, модули 120 и/или 182 синхронизации) берет номер последнего изменения, увеличивает его на один и далее записывает его рядом с GUID. Конфликт автоматически разрешается между версией A и версией B посредством сравнения наивысшего набора изменения для каждой версии. "Наивысший набор изменения" может содержать пару GUID и номера с наивысшим номером. Если наивысший набор изменения для версии A включен в версию B, а наивысший набор изменения для версии B не включен в версию A, то версия B является более новой, и конфликт разрешается таким образом, что версия B является победителем. Тем не менее, если наивысший набор изменения для версии A не включен в версию B и наивысший набор изменения для версии B не включен в версию A, то версия B не включает в себя все редактирования, сделанные в версии A. В этом случае пользователь должен разрешать конфликт вручную.

Продолжая наш предыдущий пример, допустим, что первый пользователь

синхронизирует элемент ниже первого клиентского устройства 180-1. Модуль 120 синхронизации сервера может хранить архив версий в базе 124 данных. Модуль 182 синхронизации клиента также может содержать архив версий на клиентском устройстве 180-1-s. В этой точке архив версий может выглядеть следующим образом:

Version History: {Server_GUID:1}

Первый пользователь затем редактирует элемент на первом клиентском устройстве 180-1. Архив версий может теперь отображаться следующим образом:

Version History: {Server_GUID:1, Work_GUID:2}

Первый пользователь синхронизирует элемент со вторым клиентским устройством 180-2, а затем из второго клиентского устройства 180-2 с третьим клиентским устройством 180-3. Поскольку отсутствуют редактирования в элемент, нет соответствующего изменения в архиве версий. Первый пользователь редактирует элемент на третьем клиентском устройстве 180-3. Архив версий обновляется следующим образом:

Version History: {Server_GUID:1, Work_GUID:2, Home_GUID:3}

Первый пользователь синхронизируется из третьего клиентского устройства 180-3 с сервером 122-1.

Первый пользователь возвращается на работу на следующий день и синхронизирует первое клиентское устройство 180-1 с сервером 122-1. Модуль 182 синхронизации клиента на первом клиентском устройстве 180-1 может иметь следующий архив версий:

{Server_GUID:1, Work_GUID:2}

Модуль 120 синхронизации сервера на сервере 122-1 может иметь следующий архив версий:

{Server_GUID:1, Work_GUID:2, Home_GUID:3}

Сравнение архивов версий для версии A и версии B демонстрирует, что наивысший набор изменения в A (Work_GUID:2) включен в версию B, но наивысший набор изменения для версии B (Home_GUID:3) не включен в версию A. Модуль 120 синхронизации сервера, следовательно, знает, что версия B включает в себя все обновления, которые сделаны в версии A, но версия A не включает в себя все изменения, которые сделаны в версии B. Версия B - это расширенный набор версии A, и, следовательно, он побеждает в конфликте.

Допустим, тем не менее, что первый пользователь никогда не синхронизировался с сервером 122-1 с третьего клиентского устройства 180-3, а вместо этого второй пользователь сделал изменение в элементе, размещающемся на сервере 122-1. Архив версий может быть использован для того, чтобы обнаруживать конфликт из этого сценария. В этом случае модуль 182 синхронизации клиента четвертого клиентского устройства 180-4 может иметь следующий архив версий:

{Server_GUID:1, Work_GUID:2}

Между тем, модуль 120 синхронизации сервера сервера 122-1 может иметь следующий архив версий:

{Server_GUID:1, User2_Work_GUID:2}

Сравнение архива версий для версии A и версии B теперь показывает, что наивысший набор изменения в версии A (Work_GUID:2) не включен в версию B, а наивысший набор изменения в версии B (User2_Work_GUID:2) не включен в версию A. Следовательно, версия B не является расширенным набором версии A, версия A не является расширенным набором версии B, и модуль 120 синхронизации сервера обнаружил реальный конфликт, который должен быть разрешен.

Различные варианты осуществления позволяют обнаруживать конфликты схем данных с помощью контейнера свойств. Некоторые варианты осуществления могут поддерживать несовпадения схемы между сервером 122 и клиентом 180 с помощью контейнера свойств. Контейнер свойств может означать набор определений полей для различных схем данных, используемых посредством прикладных программ и/или устройств. Контейнер свойств может быть использован для того, чтобы задавать поля на основе необходимости вместо фиксированного или жестко запрограммированного набора полей. Например, схема Task для Microsoft Outlook, приводимая в исполнение посредством клиентского устройства 180, возможно, должна иметь определенные поля, недоступные посредством приложения WSS сервера 122, и наоборот. Контейнер свойств может быть использован для того, чтобы задавать поля для различных списков или приложений, которые могут быть использованы и Microsoft Outlook, и приложением WSS на самоорганизующейся основе. Таким образом, несовпадения схемы между сервером 122 и клиентом 180 могут быть скорректированы без необходимости обязательно обновлять или модифицировать фактические прикладные программы, приводимые в исполнение посредством обоих устройств.

Различные варианты осуществления могут обнаруживать конфликты языка с помощью преобразования значений полей. Некоторые варианты осуществления могут поддерживать несовпадения языка между сервером 122-1 и клиентом с помощью 180-1 преобразования значений полей. Этот признак может быть желательным для того, чтобы поддерживать международные сценарии, где клиентское устройство 180-1 и сервер 122-1 не имеют одинакового языка. Например, Microsoft Outlook имеет логику, которая приводится в исполнение для задачи на основе поля *Status* или поля *Priority*. Сервер 122-1 может записывать различные значения в эти поля на основе языка сервера. Например, в английском языке поле *Status* может включать в себя такие выражения, как Not Started, Deferred, Completed и т.п. Microsoft Outlook должен знать, как достоверно преобразовать эти значения в значения, которые он понимает. В одном варианте осуществления секция преобразования может быть добавлена в схему для списка, который Microsoft Outlook может использовать для того, чтобы преобразовывать записанные значения в значения, которые он понимает.

Различные варианты осуществления могут синхронизировать типы структурированного содержимого с помощью одной или более методик страничной организации. Некоторые варианты осуществления могут поддерживать синхронизацию очень крупных наборов данных между сервером 122-1 и клиентом 180-1 с помощью различных методик страничной организации. Чтобы повысить производительность сервера 122-1 и клиента 180-1, запросы страниц могут быть использованы для того, чтобы получить набор элементов, которые изменены в списке. Вместо возврата всех элементов, которые изменились, сервер 122-1 может вернуть только последние Q элементов (к примеру, последние 100 элементов). Этот признак особенно полезен, когда клиент 180-1 может иметь более медленные подключения связи к серверу 122-1 или когда сервер 122-1 имеет тысячи или миллионы элементов в списке. Большой объем элементов может приводить к узким местам по производительности, в частности, при синхронизации типа структурированного содержимого Address Book (Адресная книга) между Microsoft Outlook на клиентском устройстве 180-1 и сервере 122-1, например.

Данная методика страничной организации может быть реализована рядом способов. В одном варианте осуществления, например, модуль 120 синхронизации сервера может быть выполнен с возможностью поддерживать свойство с названием

rowLimit в веб-службе *GetListChangesSinceToken*, которое задает максимальное число элементов для возврата. Например, *rowLimit=100* возвращает первые 100 элементов, которые обновлены с момента значения маркера изменения. Если свойство не задано, все элементы, измененные с момента маркера изменения, возвращаются.

Когда приложение WSS сервера 122-1 принимает запрос с пределом элементов, операции удаления могут быть исключены из пределов элементов. Параметр может быть передан в запрос списка, чтобы ограничивать число элементов, возвращаемых в *rowLimit*. Если изменено меньше, чем это число элементов, обновленный маркер изменений возвращается вместе с элементами. Тем не менее, если изменено больше числа элементов, маркер изменений не обновляется, и тот же маркер изменения может быть возвращен, который первоначально передан. Значение *ListItemCollectionPositionNext* также может быть возвращено. После приема значения *ListItemCollectionPositionNext* клиентское устройство 180-1 должно выполнить повторный запрос к серверу 122-1 при отправке и маркера изменений, и этого значения. Сервер 122-1 должен выполнить новый запрос, чтобы вернуть следующее число *rowLimit* элементов сначала *ListItemCollectionPositionNext*. Блок *If* затем может быть приведен в исполнение как требуется.

Следует отметить, что клиент 180-1 должен обязательно обработать удаления перед продолжением обновлений. В случае если клиент 180-1 принимает удаление для элемента, который он уже удалил, он должен подавить ошибку и переходить дальше. В случае если клиент 180-1 пробует обновление элемента, который удален на сервере 122-1, клиент 180-1 должен подавить ошибку и продолжить работу.

Различные варианты осуществления могут предоставлять клиенту 180-1 альтернативные URL-адреса, которые он может использовать для того, чтобы синхронизироваться с сервером 122-1 по внутренней частной сети или внешней сети общего пользования (к примеру, общедоступному Интернету). В одном варианте осуществления, например, модуль 120 синхронизации сервера может генерировать первый сетевой адрес, чтобы отправлять запрос синхронизации, по меньшей мере, с одним типом структурированного содержимого из внутренней частной сети, и второй сетевой адрес, чтобы отправлять запрос синхронизации, по меньшей мере, с одним типом структурированного содержимого из внешней сети общего пользователя. Первый и второй адреса могут содержать URL-адреса, реализованные, например, с помощью гиперссылки.

В качестве примера, модуль 120 синхронизации сервера может быть выполнен с возможностью передавать альтернативные привязки домена с возвращенными данными в ответ на запрос синхронизации. Это предоставляет возможность сценария, где пользователь переходит к списку в частной сети (к примеру, корпоративной сети интранет) и синхронизирует список с локальным приложением, таким как Microsoft Outlook. Пользователь затем идет домой, подключается к почте с помощью удаленного вызова процедуры (RPC) по протоколу передачи гипертекста (HTTP) и ожидает свои списки SharePoint, чтобы также синхронизировать. Если сервер 122-1 (к примеру, узел SharePoint) также открыт по сети общего пользователя (к примеру, сети экстранет, такой как Интернет), то пользователь может синхронизировать списки, даже если домен для URL-адреса другой.

Альтернативные привязки домена могут быть реализованы любым числом способов. В одном варианте осуществления, например, альтернативные домены могут быть возвращены в следующем порядке:

[сеть интранет], [по умолчанию], [сеть экстранет], [Интернет], [собственный]

Эти привязки могут быть возвращены в атрибуте *AlternateUrls* или теге *listitems*. Если привязки не существует, тот же порядок может быть возвращен с запятой, вставленной для пропущенного домена следующим образом:

(http://intranet, https://default, https://extranet.com, http://custom)

Эти пять альтернативных привязок доменов могут быть заданы как централизованное администрирование. Домен может включать в себя только первую часть URL-адреса, к примеру, http://www.microsoft.com или http://msw. Клиентское приложение (к примеру, Microsoft Outlook) должно быть ответственным за синтаксический анализ этих доменов и определение того, какие использовать, на основе состояния клиентского устройства 180-1-s. В общем, клиент 180-1 должен начинать с первого домена и итеративно проходить через оставшиеся домены по порядку.

Чтобы дополнительно улучшить операции синхронизации, различные варианты осуществления позволяют разрешать вопросы синхронизации, касающиеся того, когда синхронизировать клиентское устройство с веб-узлом. Например, некоторые варианты осуществления позволяют администрировать клиентскую нагрузку за счет предоставления рекомендованных интервалов синхронизации клиенту синхронизации (к примеру, клиент синхронизируется каждые *P* минут). Рекомендованные интервалы синхронизации могут представлять минимальный интервал синхронизации, максимальный интервал синхронизации или некоторое значение между ними. В одном варианте осуществления, например, модуль 120 синхронизации сервера может генерировать параметр интервала синхронизации. Сервер 122-1 может отправлять параметр интервала синхронизации клиенту 180-1. Сервер 122-1 может принимать запросы синхронизации от клиента 180-1 в соответствии с параметром интервала синхронизации.

Использование рекомендованных интервалов синхронизации позволяет предоставлять несколько преимуществ. Использование рекомендованных интервалов синхронизации позволяет повышать масштабирование так, что значительное число клиентов Microsoft Outlook (к примеру, 100000) не должно привести к тому, что пул серверов SharePoint (к примеру, вычислительный массив 140) станет непригодным для использования. Например, предположим, что Microsoft Outlook имеет интервал синхронизации, который приводит к тому, что клиентское устройство 180-1 синхронизирует каждый список структурированного содержимого каждые 60 минут. Допустим, каждый пользователь в итоге сводится к 10 синхронизированным спискам, общее число посещений сервера 122-1 может достигать 10*24*100000 ежедневно, что выражается примерно в 278 посещениях в секунду круглосуточно. Потоки по технологии очень простой синхронизации (RSS), которые синхронизируются примерно каждые 60 минут, могут дополнительно осложнять эту проблему.

Могут быть использованы различные методики для того, чтобы уменьшать рабочую нагрузку на данный сервер 122-1-n. В одном варианте осуществления, например, модифицированное время для списка кэша и/или узла может быть реализовано на клиентской стороне. Если на сервер 122-1 поступает запрос, запрашивающий изменения с момента последнего времени модификации кэшированного списка или узла, сервер 122-1 должен знать, что нет изменений, и может вообще исключать опрашивание баз 124, 126 данных. Кэш, возможно, должен утрачивать силу с некоторым регулярным интервалом, таким как, к примеру, примерно каждые 5 минут. Число списков или узлов, которые кэшируются, возможно, должно быть ограничено так, чтобы соответствовать ресурсам запоминающего

устройства, таким как, к примеру, примерно 100 списков и/или узлов.

Другая методика для того, чтобы уменьшить рабочую нагрузку на данный сервер 122-1-n, может состоять в том, чтобы реализовать методику адаптивной синхронизации даты/времени. Модуль 120 синхронизации сервера может уменьшать число запросов синхронизации, выполняемых данным клиентом Microsoft Outlook, посредством отправки клиентским устройством 180-1 параметра интервала синхронизации, который изменяется на основе оценки вероятности следующего времени изменения списка с помощью архивных данных изменения списка в показателях на информацию даты/времени, такой как рабочее время, вечернее время и выходные.

В различных вариантах осуществления рекомендуемые интервалы синхронизации могут быть основаны на архивных данных изменений содержимого, хранимых на сервере. В одном варианте осуществления, например, модуль 120 синхронизации сервера может генерировать параметр интервала синхронизации для клиента 180-1 на основе архивных данных изменений содержимого, хранимых на сервере 122-1. Модуль 120 синхронизации сервера может отправлять параметр интервала синхронизации в клиент 180-1.

Один пример методики на основе архивных данных изменения содержимого может упоминаться как сетчатая адаптивная синхронизация ("сетчатая синхронизация"). Каждый список на сервере SharePoint имеет различные характеристики применения. Число факторов может влиять на использование списка и, следовательно, частоту обновления. Примеры этих факторов могут включать в себя тип списка, тип узла, размер списка, объем пользовательской настройки списка, число пользователей узла, географическое местоположение и распределение пользователей, общее число посещений узла, время дня, текущее место в цикле проекта узла (к примеру, начало, середина, конец или архив) и т.п. Число этих факторов базируется на конфигурации, и на основе этих данных могут быть сделаны оценки шаблонов использования. Число этих факторов, тем не менее, также базируется на времени. Базирующиеся на времени факторы могут оказывать значительное влияние на шаблоны использования. Узлы в типичном варианте являются цикличными по характеру, и веб-узел редко используется на одинаковой скорости всегда. Более того, люди стремятся работать по расписаниям. Следовательно, объем работы, которая выполняется с 22:00 до 6:00, обычно меньше, чем объем, который выполняется от 9:00 до 17:00, несмотря на то что каждый период содержит 8 часов. Люди также стараются не работать по выходным. Следует отметить, что характер работы и географическое распределение работы может оказывать влияние на эти факторы, и поэтому может быть важным адаптировать шаблоны использования вместо попытки жестко запрограммировать их заранее. Предложения, которые должны оказывать наиболее существенное влияние на число запросов к серверу при удовлетворении пользовательских потребностей, должны включать в себя базирующиеся на времени алгоритмы и должны быть чувствительны к способу, которым пользователи работают и посредством этого используют данный узел.

С этой точки зрения одно проектное соображение заключается в том, чтобы предоставить в клиентское устройство 180-1 рекомендуемое время, когда оно должно в следующий раз выполнить операции синхронизации. Клиентское устройство 180-1 должно синхронизироваться с сервером 122-1, только когда произошло изменение либо в клиентском устройстве 180-1, либо на сервере 122-1. Следовательно, анализ вероятностей может быть использован для того, чтобы определять идеальный

интервал синхронизации. Администратор может интеллектуально регулировать режим работы сети 100 через регулирования порога вероятности, при котором клиентские устройства 180-1-*s* захотят повторно синхронизироваться. Например, администратор сервера может отрегулировать порог вероятности, чтобы инструктировать клиентскому устройству 180-1 синхронизироваться, когда есть 10% вероятность обновления содержимого. Если это создает слишком большую нагрузку, администратор может отрегулировать порог до 50% порога вероятности. Администратор может продолжать уточнять этот процесс до тех пор, пока характеристики сети не будут в пределах допустимых параметров.

В некоторых случаях порог вероятности для будущего использования списка может быть основан на статистических данных использования. Список может содержать, например, набор данных или элементов одного или более типов структурированного содержимого. Использование зачастую следует шаблонам. Люди, разумеется, являются продуктами привычек, и структурированные рабочие окружения зачастую дополнительно способствуют этому, поскольку это создает прогнозируемость. Чтобы представить предшествующее использование, время может быть разделено на сетку. При условии, что периодом времени является неделя, у-ось может иметь дни недели, тогда как х-ось может иметь часы в день. В каждой ячейке сетки может быть задано булево значение, указывающее то, обновлялся или нет список в течение этого периода времени, где значение 1 представляет то, что список модифицировался в течение этого периода, а значение 0 представляет, что список не модифицировался в течение этого периода. Например, посмотрим на строку понедельника и ячейку 9:30-9:40, если значение в ячейке равно 1, то список модифицировался в понедельник в некоторое время между 9:30 и 9:40. Следующей единицы может не быть до понедельника 15:10-15:20. Перемещаясь дальше, отслеживание использования узла предоставляет законченную сетку. Сетка должна иметь больше значений 1, если список обновлялся более часто, и больше значений 0, если список обновлялся менее часто. Допустим, что клиентское устройство 180-1 синхронизируется с сервером 122-1 в понедельник в 9:05. После выполнения запроса, чтобы проверить то, было ли что-либо изменено, сервер 122-1 далее должен сделать рекомендацию клиентскому устройству 180-1 в отношении того, когда оно должно осуществить свое следующее событие синхронизации. Анализируя архивные данные, очевидно, что обновление произошло в понедельник между 9:30-9:40. На основе этих архивных данных сервер 122-1 может инструктировать клиентскому устройству 180-1 выполнить свою следующую синхронизацию между 9:30-9:40. Алгоритмически это может быть осуществлено посредством прохождения по сетке, пока не достигнуто значение 1. Для каждой пройденной ячейки размер времени ячейки может быть прибавлен к рекомендуемому времени между синхронизациями. В этом примере ячейка 9:00-9:10, ячейка 9:10-9:20 и ячейка 9:20-9:30 пройдены. Следовательно, сервер 122-1 может возвратить рекомендуемый интервал синхронизации в 30 минут.

В этой точке рекомендация 9:30-9:40 является просто начальной точкой. Более длительные периоды времени уточняют данную рекомендацию. Продолжая предыдущий пример, допустим, что список обновлен в 9:22 на вторую неделю. В течение третьей недели клиентское устройство 180-1 еще раз выполняет синхронизацию с сервером 122-1 в 9:05, и сервер 122-1 должен сделать обновленную рекомендацию. Проходя по сетке, есть значение 1 в ячейке 9:20-9:30, и значение 1 в ячейке 9:30-9:40. Сетчатый алгоритм начинается в текущей ячейке и добавляет 10 минут с каждым проходом ячейки до тех пор, пока не достигнута ячейка, которая

имеет значение 1. Ячейка 9:00-9:10 пройдена, и ячейка 9:10-9:20 пройдена. Сервер 122-1 теперь может возвратить значение в 20 минут клиенту 180-1. Тем не менее, на основе архивных данных может быть создана вероятность, чтобы указать то, что список должен обновляться через 20 минут. В двух возможных понедельниках с начального времени в 9:05 до конечного времени в 9:30 список обновлялся только в один понедельник. На основе этих статистических данных вероятность может быть вычислена как 1/2, или 50%. Десять минут может быть добавлено к рекомендуемому интервалу синхронизации, и анализ вероятности может быть выполнен повторно. С двумя понедельниками в данных с начального времени в 9:05 до конечного времени в 9:40 список обновлялся два раза. Вероятность в 30 минут, следовательно, составляет 1/2+1/2=2/2, или 100%. Использование дополнительных данных приводит к уточненным рекомендациям. Это теперь становится обоснованным решением в отношении того, как часто администратор сервера удовлетворен клиентскими устройствами 180-1-s, запрашивающими информацию от серверов 122-1-n, в сравнении с тем, как важно то, пользователи сразу имеют самые актуальные данные. Если администратор сервера задает порог, например, в 50% и меньше, то сервер 122-1 должен возвратить интервал синхронизации в 20 минут. Тем не менее, если порог задан между 50-100%, то сервер должен возвратить интервал синхронизации в 30 минут.

Обобщенный алгоритм для того, чтобы генерировать интервал синхронизации, может быть проиллюстрирован следующим образом:

P - это вероятность

T - это заданный пользователем порог

S - это временной интервал каждой ячейки, в примере выше он составляет 10 минут

C - это текущее число ячеек,

G[] - это сетка, допустим, что G[0] представляет значение в текущее время

M - это максимальное время для ячейки в сетке, в примере выше это число недель

P=0;

Для (C=0; P<T; C++)

P=P+G[C]/M;

*возвращаемое значение C*S*

Следует отметить, что возвращаемое значение не должно пересчитываться при каждом клиентском запросе. Наоборот, оно может кэшироваться на период времени и считаться допустимым до тех пор, пока следующий временной интервал не начнется. В вышеприведенном примере это должно составлять до 10 минут.

Чтобы построить фактическую сетку, каждый раз, когда элемент модифицируется, время последней модификации списка проверяется. Если время последней модификации списка происходит в том же временном интервале, что и текущее время, то изменения не должны записываться. Например, два изменения, которые происходят в один период времени между 9:00 и 9:10, требуют только 1 обновления. Это поддерживает число записей на минимуме, помогая при конфликтах при блокировках и масштабировании. Если время последней модификации списка происходит в отличном временном интервале от текущего времени, то значение в текущей ячейке временного интервала может быть увеличено на 1. В некоторых случаях время модификации списка также должно быть обновлено, хотя WSS версия 3.0 уже обрабатывает это.

Решетчатый алгоритм является масштабированным во многих измерениях на основе того, сколько данных может быть сохранено, и какие тренды являются

общими при использовании конкретного приложения. В направлении *x* временной интервал каждой ячейки может быть уменьшен до минуты или увеличен до часа и более. В размерности y сетка может быть расширена так, чтобы включать в себя другую строку для каждого дня месяца либо сетка может быть свернута только до одной строки, представляющей один день. Фактически, она может быть свернута немного более интеллектуальным способом за счет наличия двух дней, где один представляет рабочий день, а другой представляет выходной день. Если сетка представляет 12-часовую посменную работу, сетка может иметь y строк, каждая из которых представляет 12-часовую смену с начала до конца.

Как упоминалось ранее, использование узла в типичном варианте следует прогнозируемым циклам. Эти циклические периоды зачастую изменяются медленно в течение нескольких месяцев. Чтобы уточнить сетчатый алгоритм, свежие данные могут быть лучшим индикатором, чем более старые данные. Например, обновление списка, которое произошло в понедельник 6 месяцев назад, в типичном варианте имеет меньшее значение, чем обновление, которое произошло вчера. Следовательно, сетчатый алгоритм может быть уточнен посредством поддержания больших наборов данных и применения функций, чтобы взвешивать данные в рамках большего набора по-разному на основе времени. Альтернативно, ресурсы запоминающего устройства могут быть использованы более эффективно посредством утрачивания силы устаревших данных. Например, предположим, что сетка сбрасывается каждые 8 недель. В сетке, заданной в предыдущем примере выше, максимальное значение в каждой ячейке должно быть 8. Следовательно, ячейка может быть представлена всего с помощью 3 битов.

Сброс ячейки, тем не менее, также удаляет данные, которые могут влиять на результаты сетчатого алгоритма. Эта проблема может быть смягчена за счет создания сеток, которые смещены друг от друга на определенный период времени. Например, первая сетка может начать запись данных на неделе 1, тогда как вторая сетка может начать запись данных на неделе 4. В этой точке модуль 120 синхронизации сервера может использовать первую сетку для вычислений вероятности. На неделе 8 первая сетка может быть сброшена, и модуль 120 синхронизации сервера может переключиться, чтобы использовать сетку для того, чтобы выполнять вычисления вероятности. Модуль 120 синхронизации сервера может начать запись в первой сетке повторно. Таким образом, модуль 120 синхронизации сервера может постоянно циклически сдвигаться от одной сетки к следующей, никогда не будучи должным начинать со стирания. В этом случае разрывность в потоке данных возникает с интервалами в 4 недели. В зависимости от важности плавности это может быть уменьшено за счет добавления дополнительных сеток. Например, 8 сеток должно приводить к менее значительным разрывностям по 1 неделе каждое. Отметим, что эта система также может быть использована для того, чтобы предоставлять большие возможности более свежим событиям посредством использования нескольких сеток для того, чтобы генерировать вероятность.

Чтобы дополнительно повысить производительность сетчатого алгоритма, максимальный или минимальный интервал синхронизации может быть задан. Минимальный интервал синхронизации, вероятно, по умолчанию должен составлять размер интервала ячейки, который в вышеприведенном примере равен 10 минутам. Размер максимального интервала зависит от уровня конфиденциальности системы. Оба должны быть конфигурируемыми посредством администратора наряду с порогом. Эти значения могут регулироваться для того, чтобы уточнять систему при

использовании, хотя может быть обременительным то, чтобы изменить структуру сетки (к примеру, представление времени на x-оси и y-оси) после того, как она уже используется. Время по умолчанию должно быть задано для начальных синхронизаций до тех пор, пока сетка не будет заполнена до минимального используемого состояния.

Данный подход применим к другим протоколам синхронизации, таким как RSS. Он также применим к таким областям, как поисковые серверы, индексаторы и прокси-серверы, которые кэшируют содержимое. Хотя алгоритм сетки может быть реализован на клиенте, точность, вероятнее всего, будет уменьшена, поскольку клиент не будет знать обо всех изменениях, сделанных на сервере. Сетчатый алгоритм также может быть использован при синхронизациях между клиентами. Он также может быть использован вместе с основанными на уведомлениях подходами к синхронизации, где уведомления обрабатывают редко изменяющееся содержимое, и эта система обрабатывает более часто изменяющееся содержимое.

Сетчатый подход может иметь варьирующийся эффект на ресурсы запоминающего устройства в зависимости от данной реализации. Оценки емкости хранения данных могут быть сделаны следующим образом:

интервалы по 20 минут
7 отдельных дней
8 недель на таблицу
2 таблицы
=378 байтов
интервалы по 10 минут
2 строки (рабочий день/выходной день)
6 недель на таблицу
2 таблицы
=288 байтов

Помимо генерирования рекомендованных интервалов синхронизации на основе статистических данных, рекомендованные интервалы синхронизации также могут быть основаны на результатах предыдущих запросов синхронизации для клиента. В одном варианте осуществления, например, модуль 120 синхронизации сервера может генерировать параметр интервала синхронизации для клиента на основе предыдущих результатов синхронизации из предыдущих запросов синхронизации посредством клиента 180-1. Модуль 120 синхронизации сервера может отправлять параметр интервала синхронизации в клиент 180-1.

В качестве примера, модуль 120 синхронизации сервера может выполнять функциональную адаптивную синхронизацию для того, чтобы генерировать параметр интервала синхронизации. Число запросов синхронизации, сделанных данным клиентом Microsoft Outlook, может быть сгенерировано с помощью определенной формы функции экспоненциального спада/роста. Например, возьмем последний интервал синхронизации. Если изменение произошло, уменьшим временной интервал посредством экспоненциальной функции. Тем не менее, если изменение не произошло, увеличим временной интервал посредством экспоненциальной функции.

Другие методики также могут быть использованы для того, чтобы генерировать параметр интервала синхронизации. Например, методика адаптивной синхронизации по последней модификации может быть использована. Эта методика уменьшает число запросов синхронизации, выполняемых клиентом Microsoft Outlook, посредством отправки клиентом 180-1 интервала синхронизации, который изменяется строго на

основе времени последней модификации для списка. Например, если список модифицирован в течение последнего часа, синхронизируемся снова через 20 минут. Если список модифицирован более часа назад, но меньше дня назад, синхронизируемся снова через 2 часа. Если список модифицирован более дня назад, синхронизируемся снова через один день. В другом примере методика одного запроса для нескольких списков также может быть использована. Эта методика уменьшает число запросов синхронизации, выполняемых клиентом Microsoft Outlook, посредством предоставления возможности клиенту 180-1 делать один запрос, который проходит все интересующие GUID списка, за раз. Времена последней модификации могут проверяться пакетно для всех этих списков, и затем модуль 120 синхронизации сервера может сообщить клиентскому устройству 180-1 о том, какие списки явно синхронизировать. В еще одном примере методика основанной на уведомлениях по электронной почте синхронизации может быть использована. Текущие методики синхронизации могут использовать различные методики опрашивания, где клиент опрашивает сервер на регулярной или полурегулярной основе. В одном варианте осуществления сервер 122-1 может уведомлять клиента 180-1 о том, что событие синхронизации требуется, посредством отправки "скрытого оповещения по электронной почте". Это должно усиливаться на текущей инфраструктуре оповещений. Оповещение не обязательно должно быть видимым пользователю и должно автоматически создаваться в SharePoint в то время, когда пользователь выбирает выполнить операции синхронизации. Это может быть особенно эффективным для часто изменяющихся списков. Тем не менее, клиентское устройство 180-1 должно поддерживать прием поступающих по почте запросов, чтобы реализовать эту конкретную методику.

Различные конкретные подробности изложены в данном документе для того, чтобы предоставлять полное понимание вариантов осуществления. Тем не менее, специалисты в данной области техники должны понимать, что примерные варианты осуществления могут использоваться на практике без этих конкретных подробностей. В других случаях хорошо известные операции, компоненты и схемы не описаны подробно, чтобы не затруднять понимание вариантов осуществления. Следует принимать во внимание, что конкретные структурные и функциональные подробности, раскрытые в данном документе, могут быть характерными и не обязательно ограничивают область применения вариантов осуществления.

Также следует отметить, что все ссылки на "один вариант осуществления" или "вариант осуществления" означают, что конкретный признак, конструкция или характеристика, описываемые в связи с вариантом осуществления, включены, по меньшей мере, в один вариант осуществления. Вхождения фразы "в одном варианте осуществления" в различных местах подробного описания не обязательно ссылаются на один и тот же вариант осуществления.

Некоторые варианты осуществления могут быть описаны с помощью выражения "соединены" или "подключены" наряду с его производными словами. Следует понимать, что эти термины не служат в качестве синонимов друг для друга. Например, некоторые варианты осуществления могут быть описаны с помощью термина "подключены", чтобы указывать, что два или более элементов находятся в прямом физическом или электрическом контакте друг с другом. В другом примере некоторые варианты осуществления могут быть описаны с помощью термина "соединены", чтобы указывать, что два или более элементов находятся в прямом физическом или электрическом контакте друг с другом. Тем не менее, термин

"соединены" также может означать, что два или более элементов не находятся в прямом контакте друг с другом, но при этом по-прежнему совместно работают или взаимодействуют друг с другом. Варианты осуществления не ограничены этим контекстом.

Некоторые варианты осуществления могут быть реализованы, например, с помощью машиночитаемого носителя или изделия, которое может сохранять инструкцию или набор инструкций, которые, если приводятся в исполнение посредством машины, могут инструктировать машине осуществлять способ и/или операции в соответствии с вариантами осуществления. Эта машина может включать в себя, например, любую надлежащую платформу обработки, вычислительную платформу, вычислительное устройство, вычислительное устройство, вычислительную систему, систему обработки, компьютер, процессор и т.п., и может быть реализована посредством любой надлежащей комбинации аппаратных средств и/или программного обеспечения. Машиночитаемый носитель или изделие может включать в себя, например, любой надлежащий тип блока запоминающего устройства, запоминающего устройства, изделия из запоминающего устройства, запоминающего носителя, устройства хранения, изделия хранения, носителя хранения и/или блока хранения, к примеру память, съемные и стационарные носители, стираемые или нестираемые носители, записываемые или перезаписываемые носители, цифровые или аналоговые носители, жесткий диск, гибкий диск, компакт-диск только для чтения (CD-ROM), записываемый компакт-диск (CD-R), перезаписываемый компакт-диск (CD-RW), оптический диск, магнитные носители, магнитооптические носители, съемные карты памяти или диски, различные типы универсальных цифровых дисков (DVD), ленты, кассеты и т.п.

Хотя предмет изобретения описан на языке, характерном для структурных признаков и/или методологических действий, необходимо понимать, что предмет изобретения, определенный в прилагаемой формуле изобретения, не обязательно ограничен описанными характерными признаками или действиями. Наоборот, вышеописанные характерные признаки и действия раскрываются как примерные формы реализации формулы изобретения.

Формула изобретения

1. Машиночитаемый носитель хранения, содержащий выполняемые компьютером инструкции, которые при исполнении компьютерным устройством выполняют способ для синхронизации данных между клиентским устройством и серверным устройством, причем способ содержит этапы, на которых:

идентифицируют один или более типов структурированного содержимого, поддерживаемых клиентским устройством;

принимают на клиентском устройстве один или более типов структурированного содержимого, поддерживаемых серверным устройством;

идентифицируют клиентским устройством один или более типов структурированного содержимого, поддерживаемых как клиентским устройством, так и серверным устройством;

модифицируют в клиентском устройстве, по меньшей мере, первый элемент данных одного или более типов структурированного содержимого, поддерживаемых как клиентским устройством, так и серверным устройством;

принимают от серверного устройства серверную версию, по меньшей мере, первого элемента данных, причем серверная версия первого элемента данных включает в себя

архив версий;

анализируют архив версий серверной версии первого элемента данных, причем архив версий для серверной версии первого элемента данных включает первое множество пар пользовательских идентификаторов и номеров изменений, и при этом пользовательский идентификатор идентифицирует компьютер, который редактировал первый элемент данных, а номер изменений указывает последовательность изменений, сделанных в первом элементе данных;

определяют из архива версий серверной версии первого элемента данных, был ли первый элемент данных отредактирован третьим устройством;

определяют из архива версий серверной версии первого элемента данных, что модификации, сделанные в клиентском устройстве, включены в серверную версию первого элемента данных; и

синхронизируют один или более типов структурированного содержимого, поддерживаемых как клиентским устройством, так и серверным устройством, причем синхронизация основана, по меньшей мере, на архиве версий.

2. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

генерируют различные гиперссылки, чтобы отправлять упомянутый запрос синхронизации для различных типов структурированного содержимого.

3. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

определяют объем структурированного содержимого для синхронизации;

задают параметры синхронизации на основе упомянутого объема; и

отправляют информацию синхронизации упомянутому клиентскому устройству в соответствии с упомянутыми параметрами синхронизации.

4. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

отправляют упомянутый запрос синхронизации из веб-обозревателя с использованием блока управления протоколами.

5. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

принимают упомянутый запрос синхронизации с упомянутым типом структурированного содержимого;

определяют, модифицировано ли структурированное содержимое, указанное посредством упомянутого типа структурированного содержимого, с помощью журнала изменений; и

отправляют информацию синхронизации упомянутому клиентскому устройству на основе упомянутого определения.

6. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

определяют, были ли несколько версий структурированного содержимого модифицированы независимо, с помощью архива версий.

7. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

обнаруживают конфликты схем данных с помощью контейнера свойств.

8. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

обнаруживают конфликты языка с помощью преобразования значений полей.

9. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

генерируют параметр интервала синхронизации;

отправляют параметр интервала синхронизации клиентскому устройству; и

принимают запросы синхронизации от клиентского устройства в соответствии с параметром интервала синхронизации.

10. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

генерируют параметр интервала синхронизации для клиентского устройства на основе архивных данных изменений содержимого, хранимых на серверном устройстве, или результатов предыдущих синхронизации из запросов предыдущих синхронизации; и

отправляют упомянутый параметр интервала синхронизации упомянутому клиентскому устройству.

11. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

синхронизируют типы структурированного содержимого с помощью операций страничной организации.

12. Машиночитаемый носитель по п.1, в котором способ дополнительно содержит этапы, на которых:

генерируют первый сетевой адрес, чтобы отправлять запрос синхронизации с, по меньшей мере, одним типом структурированного содержимого из внутренней частной сети, и второй сетевой адрес, чтобы отправлять запрос синхронизации с, по меньшей мере, одним типом структурированного содержимого из внешней сети общего пользователя.

13. Способ для синхронизации данных между клиентским устройством и серверным устройством, содержащий этапы, на которых:

передают, посредством клиентского устройства, запрос на список содержимого сервера, причем список содержимого сервера содержит типы структурированного содержимого, поддерживаемые серверным устройством;

принимают посредством клиентского устройства список содержимого сервера и сравнивают список содержимого сервера со списком содержимого клиента, причем список содержимого клиента содержит типы структурированного содержимого, поддерживаемые клиентским устройством;

идентифицируют, посредством клиентского устройства, один или более типов структурированного содержимого, поддерживаемых как клиентским устройством, так и серверным устройством;

передают посредством клиентского устройства запрос синхронизации, чтобы синхронизировать один или более типов структурированного содержимого, поддерживаемых как клиентским устройством, так и серверным устройством;

модифицируют в клиентском устройстве один или более элементов данных одного или более типов структурированного содержимого, поддерживаемых как клиентским устройством, так и серверным устройством;

принимают из серверного устройства серверную версию одного или более элементов данных, причем серверная версия одного или более элементов данных включает в себя архив версий;

анализируют архив версий серверной версии одного или более элементов данных, причем архив версий для серверной версии одного или более элементов данных

включает в себя первое множество пар пользовательских идентификаторов и номеров изменений и причем пользовательский идентификатор идентифицирует компьютер, который редактировал один или более элементов данных, и при этом номера изменений указывают последовательность изменений, сделанных в одном или более элементах данных;

определяют из архива версий серверной версии одного или более элементов данных, что один или более элементов данных были отредактированы третьим устройством;

определяют из архива версий серверной версии одного или более элементов данных, что модификации, сделанные на клиентском устройстве, включены в серверную версию одного или более элементов данных; и

автоматически сохраняют серверную версию одного или более элементов данных в клиентском устройстве.

14. Система для синхронизации структурированного содержимого между клиентским устройством и серверным устройством, содержащая:

процессор и

память, связанную с процессором, причем память содержит инструкции компьютерной программы, выполняемые процессором для:

идентификации одного или более типов структурированного содержимого, поддерживаемых серверным устройством;

приема списка содержимого клиента, идентифицирующего один или более типов структурированного содержимого, поддерживаемых клиентским устройством;

сравнения одного или более типов структурированного содержимого, поддерживаемых клиентским устройством, и одного или более типов содержимого, поддерживаемых серверным устройством для генерации списка типов совместно используемого содержимого, при этом список типов совместно используемого содержимого содержит один или более типов содержимого, поддерживаемых как клиентским устройством, так и серверным устройством; и

передачи посредством серверного устройства запроса синхронизации, чтобы синхронизировать один или более типов структурированного содержимого, поддерживаемых как клиентским устройством, так и серверным устройством, причем перед синхронизацией определяется, модифицирован ли до запроса синхронизации, посредством третьего устройства, архив версий одного или более типов содержимого, поддерживаемых как клиентским устройством, так и серверным устройством.

15. Система по п.14, дополнительно содержащая модуль синхронизации клиента, сконфигурированный для отправки запроса синхронизации, чтобы синхронизировать типы структурированного содержимого, поддерживаемые серверным устройством и клиентским устройством.

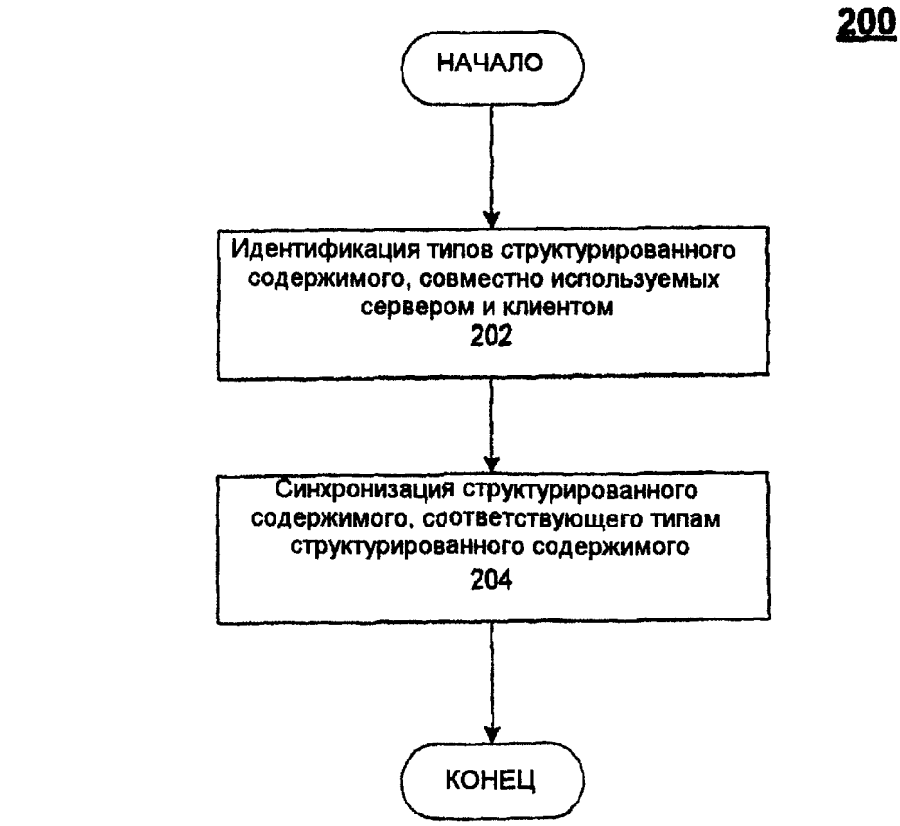
16. Система по п.14, дополнительно содержащая модуль синхронизации сервера, сконфигурированный для:

приема списка содержимого клиента;

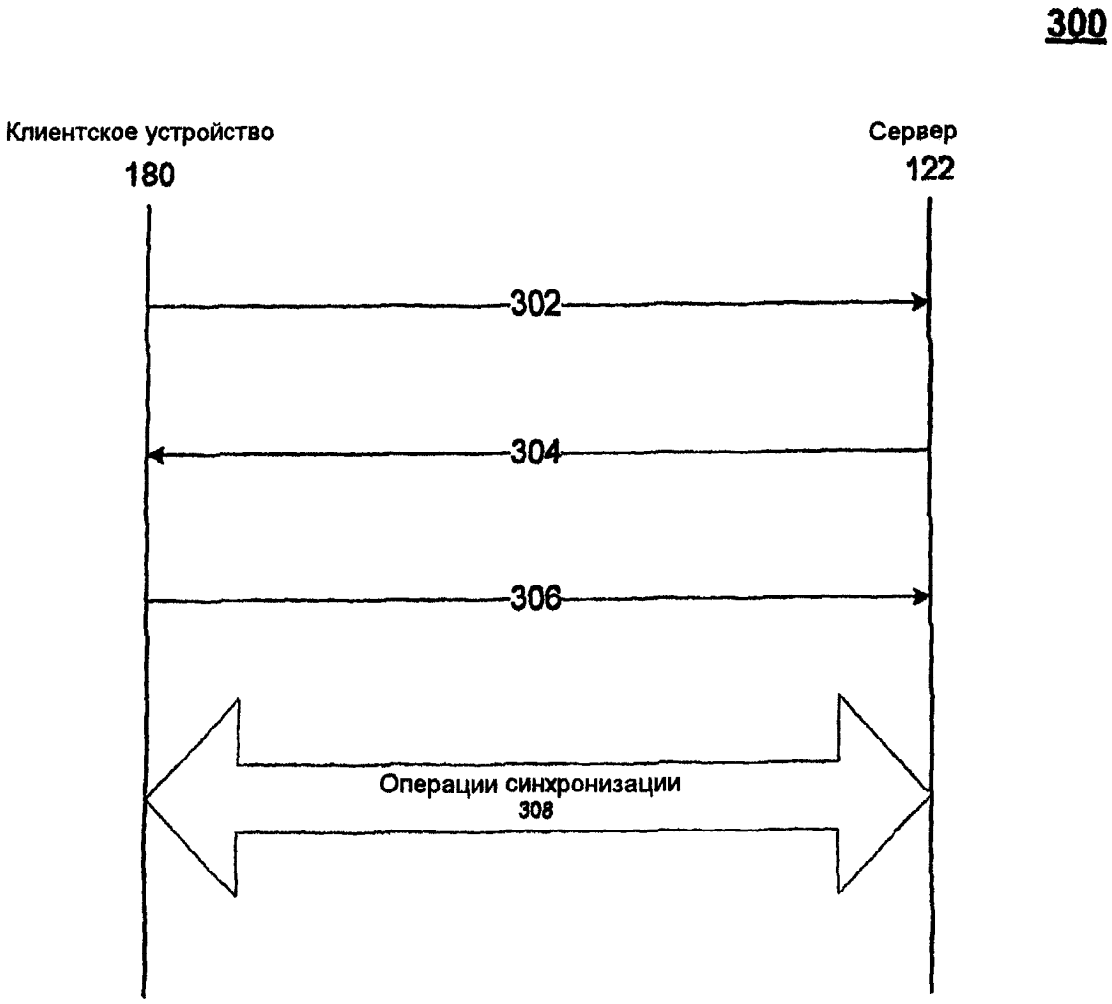
сравнения типов структурированного содержимого, поддерживаемых клиентским устройством с типами структурированного содержимого сервера, поддерживаемыми серверным устройством;

отправления списка типов совместно используемого содержимого с совпадающими типами структурированного содержимого; и

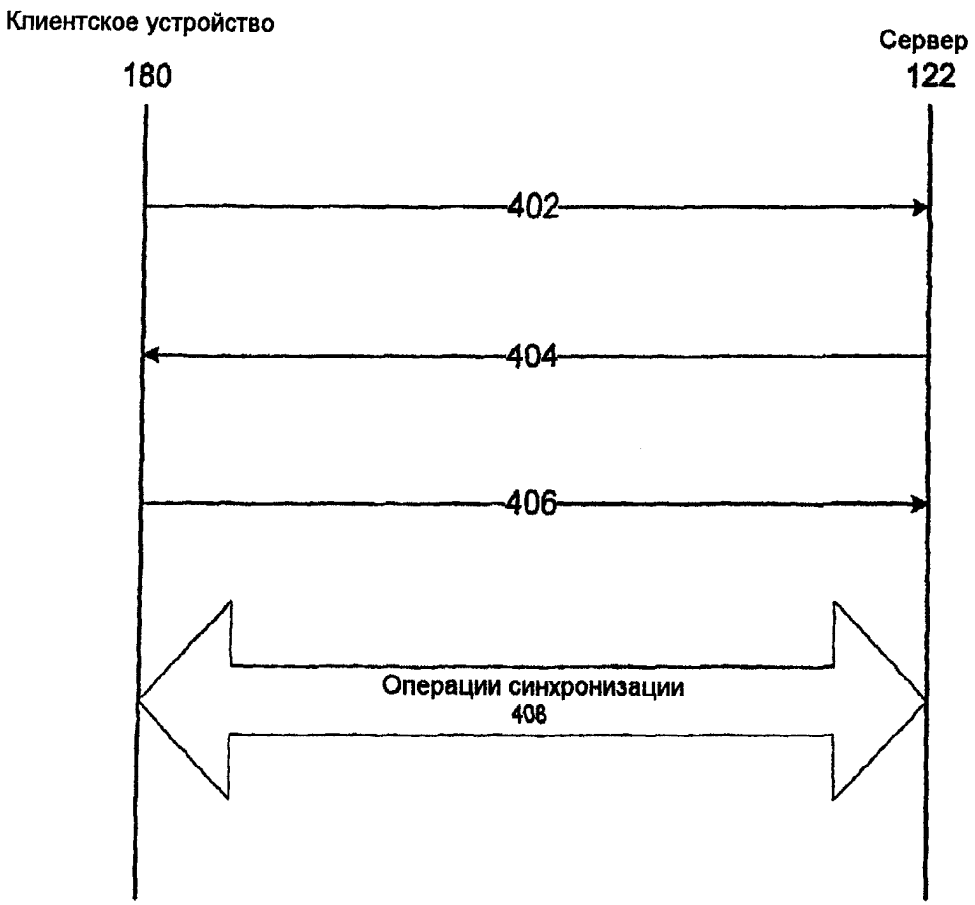
приема запроса синхронизации, чтобы синхронизировать типы структурированного содержимого в соответствии с упомянутым списком совместно используемого содержимого.


200
НАЧАЛО
Идентификация типов структурированного содержимого, совместно используемых сервером и клиентом
202
Синхронизация структурированного содержимого, соответствующего типам структурированного содержимого
204
КОНЕЦ

ФИГ.2


300
Клиентское устройство
180
Сервер
122
302
304
306
Операции синхронизации
308

ФИГ.3


400
Клиентское устройство
180
Сервер
122
402
404
406
Операции синхронизации
408

ФИГ.4