

ЖХТА решения для P2P

Новая сетевая вычислительная платформа от Sun устанавливает основы инфраструктуры для разработки peer-to-peer систем.

P2P Tutorial

Наванис Кришнан

Peer-to-peer (P2P) технологии, несомненно, являются одной из наиболее популярных тем на сегодняшний день. Популярность, достигнутая с помощью таких систем как Napster и Gnutella, подтверждает потенциал peer-to-peer систем. Среди множества проектов/попыток в этой области, Sun начинает свой проект ЖХТА -сетевой вычислительной платформы для разработки P2P систем. Эта статья рассказывает о P2P в целом и технологии ЖХТА в частности. Она может послужить введением в тему для всех новых сторонников, разработчиков, и просто любителей, желающих заняться разработкой P2P приложений. *(4100 слов)*

Сегодня Internet переживает революционные изменения, которые направлены на изменение одной из самых базовых особенностей. Речь идет о конечных пользователях Inet, таких как desktop PC, mobile phones, PDA, и тому подобное — требующих улучшения своего статуса в сети. Революция, источником энергии для которой стали такие системы, как Napster & Gnutella, которые приведут к концу эры клиент-серверного Inet, революция, имя которой — P2P.

P2P(or peer-to-peer) — это топология сети, где в зависимости от контекста операций, каждый узел, может работать либо сервером, либо клиентом. P2P обеспечивает некоторые интересные возможности, отсутствующие в традиционной архитектуре Client/Server, где заранее четко определены функции клиента либо сервера для каждого узла.

В этой статье я представлю вам P2P технологию и сравню ее с традиционными

Client/Server технологиями. Я также представлю Вам ЖХТА (произносится как jux-ta), вычислительную платформу P2P, изобретенную Bill Joy, руководителем по исследованиям в Sun Microsystem и членом совета директоров. ЖХТА является open source продуктом, и над ним в данный момент работают сотни разработчиков. ЖХТА является многообещающей технологией для P2P. Он определяет набор протоколов, которые пользователь может использовать для разработки почти любых P2P приложений. С другой стороны, эти протоколы достаточно гибкие, чтобы быть легко приспособленными для специфических требований конкретных приложений. Пока ЖХТА не навязывает никаких определенных языков программирования или среды исполнения, Java потенциально может стать одним из языком для имплементации ЖХТА по ряду понятных причин: переносимость, простота разработки на java, и богатый набор библиотек.

1. P2P: Описание

На сегодняшний день, наиболее распространенной моделью является Client/Server. Ниже представлена типичная Client/Server архитектура:

В Client/Server архитектуре, клиенты опрашивают сервер, и сервер возвращает необходимые данные и производит нужные операции над ними. На сегодняшний день существуют разные сервера в Inet: Web сервера, Mail сервера, FTP и т.д. Архитектура Client/Server — это пример централизованной архитектуры, где вся сеть зависит от центральных узлов, называемых серверами, предназначенных для обеспечения необходимых сервисов. Без серверов такая архитектура не имеет никакого смысла. Независимо от наличия в сети клиентов, сеть будет существовать исключительно при условии существования серверов.

Подобно архитектуре Client/Server, P2P также распределенная вычислительная модель(!!!), но существует очень важная отличительная черта. Архитектура P2P — это децентрализованная архитектура (смотрите рис 2), где в сети не существует понятия клиента или сервера. Каждый объект в сети, назовем его peer (англ. равный, такой же), имеет тот же статус, это означает, что этот объект может выполнять как функции клиента (отсылать запросы) так и сервера (получать ответы).

И хотя все peers имеют одинаковый статус, это не значит, что они должны иметь

JXTA решения для P2P

одинаковые физические возможности. P2P сеть может состоять из peers с разными возможностями, начиная от мобильных устройств и заканчивая mainframes. Некоторые мобильные peer могут и не поддерживать всех функциональных возможностей серверов, в силу ограничения их ресурсов (слабый процессор/небольшой объем памяти), однако сеть никак не ограничивает их.

Обе сетевые модели имеют свои преимущества и недостатки. Визуально вы можете видеть, что рост Client/Server системы (которая тем больше чем клиентов в нее добавлено) приводит к росту нагрузок на сервер. С каждым новым клиентом центральный узел слабеет. Таким образом, сеть может становиться перегруженной.

P2P сеть работает по другому сценарию. Каждый объект в сети (peer), является активным в сети, peer предоставляет некоторые ресурсы в сети, такие как пространство для хранения данных и дополнительные такты CPU. Чем больше peer в сети, тем больше производительность самой сети. Следовательно, по мере того как растет сеть, она становится мощнее.

Также P2P отличается от Client/Server модели тем, что P2P система считается рабочей, если в ней есть хотя бы один активный peer. Система будет считаться неактивной, если ни один peer не активен.

Однако существуют и недостатки у P2P систем. Во-первых, управление такой сетью намного сложнее, чем управление Client/Server системами, где администрирования требует только центральный узел — Server. Таким образом, нужно затратить намного больше усилий на поддержку security, backup, и т.п. Во-вторых, P2P протокол намного более "разговорчивый" — peer может присоединиться к сети или выйти из нее в любой момент, и это может отрицательно сказаться на производительности (Рассмотрите "Bandwidth Barriers to Gnutella Network Scalability" для дополнительной информации.)

2. Проект JXTA

Разные протоколы, разные архитектуры, разная реализация. Это точно соответствует описанию текущих P2P систем. На сегодняшний день, разработчики используют разные методологии и подходы при создании P2P систем. Стандарты, которых предостаточно в Client/Servers системах, заметно отсутствуют для P2P. Для решения этого дефицита, Sun разработал JXTA.

Цитата с ЖХТА:

Project ЖХТА занимается разработкой базы сетевой вычислительной технологии для обеспечения набора несложных, небольших и гибких механизмов, позволяющих реализовать P2P систем для любой платформы и в любое время. Этот проект сначала обобщает функциональность P2P систем, а потом строит основы технологии, направленные на преодоление сегодняшних ограничений для P2P протоколов. Цель проекта — создание базовых механизмов, выбор способов их использования предоставлен разработчикам приложений.

ЖХТА стремится к обеспечению основной инфраструктуры P2P, на основе которой могут создаваться другие P2P приложения. Эта база состоит из набора протоколов, которые не зависят ни от конкретного языка программирования, ни от конкретной платформы, ни от конкретного типа сетевого соединения или транспортного протокола. Эти протоколы решают только самые необходимые задачи для построения обобщенных P2P систем. Простые протоколы с низким объемом служебной информации нацелены, цитируя заявление авторов ЖХТА, на "любое устройство с цифровым пульсом".

ЖХТА определяет шесть протоколов, но от реегс не требуется реализовывать все из них. Количество протоколов, которые реализует реег, зависит от его возможностей, возможно, кому-то будет достаточно и одного. Кроме того, реегс могут расширять или заменять протоколы в зависимости от своих потребностей.

Важно заметить, что протоколы ЖХТА не гарантируют взаимодействия между собой. Здесь, вы можете провести параллель между ЖХТА и TCP/IP. Оба протокола FTP и HTTP созданы поверх TCP/IP, но вы не можете получить доступ к WEB странице через FTP. Точно та же картина с ЖХТА. То, что две системы созданы поверх ЖХТА, еще не означает, что они могут взаимодействовать между собой. Разработчики должны сами спроектировать приложения таким образом, чтобы они могли взаимодействовать. Однако, разработчики могут использовать ЖХТА, обеспечивающий базовый уровень для взаимодействия, с целью упрощения решения проблем взаимодействия.

3. XML в ЖХТА

Бесспорно, первый шаг к обеспечению универсального уровня протокола — это

применение подходящей для большинства доступных на сегодняшний день платформ формата представления данных. XML — это идеальный кандидат для такой задачи. Разработчики на JXTA осознают, что XML становится общепринятым стандартом для обмена данными. XML обеспечивает универсальный, независимый от языка и платформы формат передачи данных. XML может быть легко трансформирован в другие кодировки. Следовательно, формат XML используется для всех JXTA протоколов.

Хотя сообщения в JXTA определено в XML формате, JXTA не зависит от XML. Фактически, клиенты JXTA не требуют XML парсеров; это дополнительный компонент. Просто думайте об XML как о согласованной форме представления данных в JXTA. PDA устройства, такие как мобильные тел. должны использовать прекомпилированные XML сообщения.

4. Глоссарий JXTA

Перед тем как идти дальше давайте быстро рассмотрим разные концепции в JXTA.

Peers (пользователь) Любая сущность в сети, реализующая один или несколько протоколов JXTA. Peer — он может быть любое устройство от mainframes до мобильных телефонов или даже просто сенсорных мониторов. Peer существует независимо и общаются между собой асинхронно.

Peer Groups (Группа пользователей) Peers с общими интересами могут образовывать группы. Границы групп peers не ограничены границами физического сетевого домена.

Messages (Сообщение) Все коммуникации в сети JXTA происходят путем отсылки и получения сообщений. Эти сообщения, называемые JXTA сообщениями, придерживаются стандартного формата, который является ключом для способности к взаимодействию.

Pipes (Канал) Pipes организует виртуальный коммуникационный канал в JXTA окружении. Peer-ы используют его для отправки и получения сообщений. Pipes считается виртуальным, потому что peers не должны знать их настоящих адресов для того, что бы их использовать. Это очень важная абстракция.

Services (Услуги) Как peers так и peer groups могут предлагать сервисы. Сервис,

предложенный отдельным peer, на личном уровне, является Peer Services, и концептуально является эквивалентом централизации. Никакой другой peer не обязан предлагать такой же сервис; если peer в данный момент не доступен, то и сервис становится недоступным.

Сервис, предложенный Peer group, называется peer group services. В отличие от peer services, этот сервис может быть предоставлен несколькими peer в сети. Доступ к такому сервису осуществить проще, так как сервис будет доступен, пока доступен хотя бы один peer в группе.

Codats Codats (**Code/Data**), в ЖХТА, означает, что его содержимое будет одновременно и кодом и данными. Codats может быть опубликован, просмотрен, и реплицирован в случае надобности.

Advertisement (Объявление) Объявления печатает и просматривает любые ЖХТА ресурсы такие как peer, peer group, pipe, codat. Объявление имеет форму XML документа.

Identifiers (Идентификатор) Идентификатор играет ключевую роль в среде ЖХТА. Идентификатор специфицирует ресурсы, а не физические адреса в сети. Идентификатор в ЖХТА определен как URN (Уникальный имя ресурса). URN это не что иное как URI (Уникальный идентификатор ресурса), который должен оставаться глобально уникальным и постоянным, даже если ресурс, на который он ссылался, исчез.

World peer group (Всемирная группа пользователь) Каждый ЖХТА peer по умолчанию принадлежит к world peer group. Каждый ЖХТА peer знает о world peer group, даже если он не может найти никаких других peers в сети. Более того, даже peers без доступа в сеть принадлежат world peer group.

Net peer group (Сетевая peer группа) В локальной сети, сетевой администратор обычно конфигурирует таким образом что каждый peer в сети может присоединиться к net peer group. Эта группа похожа на DHCP (dynamic host configuration protocol) сервис. Net peer group обеспечивают peers глобальное соединение согласно ограничениям, установленным администратором.

Rendezvous peers (рандеву peer) Рандеву peers — это специальный peer который

ЖХТА решения для P2P

сохраняет информацию о других peers. Таким образом, рандеву peer может помочь peers находить другие peers в сети. Rendezvous peer может перенаправить поиск к другому rendezvous peer.

Endpoints (адресаты) Endpoints — это адресаты в сети которые могут быть представлены в сети через сетевой адрес. Peers не используют endpoints напрямую, они используют их через **pipes**, которые созданы поверх endpoints.

Routers (Роутер) Все, что двигает пакеты внутри ЖХТА протокола, называется ЖХТА router. Не все peer должны быть routers. Peers которые не являются routers должны найти routers в сети для роутинга их сообщения.

5. Протоколы ЖХТА

В основе ЖХТА лежит набор протоколов, созданных разработчиками ЖХТА. На основе этих протоколов создаются прикладные системы. Созданные с небольшим количеством служебной информации, эти протоколы не знают ничего о той сетевой топологии, поверх которой работает прикладное приложение, которое использует эти протоколы.

5.1. Peer Discovery Protocol (PDP)

Peers используют этот протокол для поиска всех открытых ЖХТА ресурсов. Так как Advertisement представляет открытые ресурсы, PDP попросту помогает peer находить advertisement других ресурсов. Будучи низкоуровневым протоколом, PDP обеспечивает базовые механизмы поиска. Прикладные системы могут выбирать разные высокоуровневые механизмы для поиска, которые реализованы поверх PDP.

5.2. Peer Resolver Protocol (PRP)

Часто в сети, peer отправляет запрос на доступ к какому-то ресурсу или сервису. PRP — это попытка стандартизации формата таких запросов. С помощью этого протокола peer может послать запрос и получить на него ответ.

5.3. Peer Information Protocol (PIP)

PIP используется для определения состояния (ping) peer в сети ЖХТА. Peer, получающий ping сообщение, может:

- отослать простое признание, состоящее только из времени работы в сети.
- может послать полный ответ, включающий его advertisement
- просто проигнорировать это сообщение. Таким образом, этот peer может быть доступен для получения сообщений, но не отвечать на сообщения.

5.4. Peer Membership Protocol (PMP)

Peer используют этот протокол для подключения и выхода из peer group. Этот протокол проходит четыре дискретных шага, которые осуществляются peer-ами и таким образом определяют ЖХТА сообщения для каждого из видов действий:

- Обращение (Apply). Peer, заинтересованный во вступлении в группу должен запросить у аутентификатора право на вступление в группу. Аутентификатор после проверки отправляет уведомление peer.
- Присоединение (Join). После получения уведомления peer может присоединиться к группе.
- Восстановление (Renew). Для обновления своего членства в группе peer используют восстановительное сообщение (renew message)
- Отказ (Cancel). Peers также может отказаться от членства в группе.

5.5. Pipe Binding Protocol (PBP)

В ЖХТА, peer получает доступ к сервису через pipe. Peer может создать новый pipe для доступа к сервису, или работать через уже существующий. Peer так же может отсоединиться от канала. Все эти действия peer выполняет с помощью PBP.

5.6. Endpoint Routing Protocol (ERP)

С помощью этого протокола peer может доставить сообщение адресату. ERP помогает peer пересылать запросы к роутерам других peer для определения маршрутов при отправке сообщений.

ЖХТА определяет базовый набор Peer Group сервисов, необходимых для создания и

функционирования систем на основе JXTA протокола. Каждый базовый сервис реализует один из базовых JXTA протоколов. Например, базовый Discovery Services создан поверх Peer Discovery Protocol. Кроме Discovery service, так же существуют: Membership service, Access Service, Pipe Service, Access Service, Pipe Service, Resolver Service, and Monitoring Service.

6. Связка JXTA и Java

Лучший способ увидеть эти протоколы в действии — рассмотреть связку JXTA и Java, стандартную реализацию JXTA на языке Java. Разработчики могут использовать существующую реализацию или реализовать собственную версию протоколов на языках и платформах по своему выбору. Хотя стандартная реализация использует протоколы HTTP и TCP/IP из-за их простоты и популярности, вы можете реализовать протоколы JXTA на основе любых транспортных протоколов, в зависимости от топологии сети.

Рассмотрим реализацию JXTA на Java в ее сегодняшнем состоянии.

7. Организация классов

Реализация состоит из двух главных иерархий классов:

- Классы net.jxta
- Классы net.jxta.impl

Первый пакет содержит все интерфейсы JXTA, которые представляют собой Java-интерфейсы протоколов и основных строительных блоков JXTA. Второй пакет содержит реализации этих интерфейсов. Интерфейсы и их реализации должны быть разделены явным образом. Давайте углубимся в эти пакеты.

7.1. Где мой реер?

Peer — независимый асинхронный объект в сети, характеризуемый значением Peer ID. Вы можете рассматривать экземпляр исполняемого кода, как реер. В данной реализации реер запускается загрузочным классом net.jxta.impl.peergroup.Boot, который содержит метод main().

Возможности реер зависят от группы, к которой он принадлежит. Но у каждого реер есть какие-то минимальные возможности по определению — например, наличие Peer ID. Это означает, что должна существовать хотя бы одна группа, членами которой являются все реер — Мировая группа реер, также называемая платформенной группой реер. Мировая группа представлена классом `net.jxta.impl.peergroup.Platform`, реализацией интерфейса `net.jxta.peergroup.PeerGroup`.

7.2. Вложенность групп реер.

Представьте, что реер P1 является членом группы PG1, которая предоставляет базовые услуги поиска и обнаружения. В сети также имеется группа PG2, которая предлагает более продвинутое услуги поиска. P1 может присоединиться к PG2, чтобы пользоваться продвинутым поиском, и одновременно использовать базовые услуги PG 1. Чтобы обеспечить такую функциональность, мы можем использовать *вложенность групп реер*. При этом услуги, доступные в одной группе, перекрывают (overload) услуги, доступные в другой. Это дает отношения наследования — первая группа (в нашем случае PG1) является родителем, а вторая (PG2) — потомком. В этом случае услуги потомка перекрывают услуги родителя. Поэтому, когда P 1 входит в PG2, группы будут вложены следующим образом: Мировая группа/ PG1/PG2. Обратите внимание, что на вершине иерархии всегда находится Мировая группа, поскольку все реер по определению являются членами этой группы.

7.3. Группы реер как приложения.

Важная абстракция в реализации — приложение (`net.jxta.platform.Application`). Это то, что группа реер может инициализировать, запустить или остановить. Интересно заметить, что обычно одна группа (`net.jxta.peergroup.PeerGroup`) запускает другую (как в случае со вложенностью групп), поэтому группа реер тоже является приложением. Исключение — мировая группа, которая не запускается какой-либо другой группой. Приложение определяет три метода: `init()`, `startApp()` и `stopApp()`.

Так эти методы определяются в классе `Application`:

```
public void init(PeerGroup group, Advertisement adv);
public int startApp(String[] args);
public void stopApp();
```

7.4. Каналы и адресаты.

Как указывалось ранее, каналы (pipes) — это виртуальные коммуникационные каналы, используемые для передачи сообщений в среде ЖХТА. Каналы представлены интерфейсом Pipe (net. jxta. pipe. Pipe), который считается сервисом и поэтому наследуется от интерфейса Service (net. jxta. service. Service). Далее мы будем классифицировать каналы как входные (net. jxta. pipe. InputPipe) и выходные (net.jxta.pipe.OutputPipe). Класс PipeService (net. jxta. impl. pipe. PipeService) содержит реализацию каналов, используемую в нашей реализации ЖХТА.

Каналы реализуются поверх адресатов (endpoints). Класс Endpoint (net. jxta. impl. endpoint. Endpoint) — коллекция транспортных адресатов. Адресаты могут либо передавать широковещательные сообщения, используя метод propagate(), либо посылать сообщения конкретному адресату, используя реализацию Endpoint Messenger (net. jxta. impl. endpoint. EndpointMessenger). Реализация Endpoint Messenger зависит от транспортного протокола, для которого она реализована. Например, класс net.jxta.impl.endpoint.http.HttpNonBlockingMessenger — реализация Endpoint Messenger для протокола HTTP.

7.5. Объявления.

Все объявления в реализации расширяют абстрактный базовый суперкласс Advertisement (net. jxta. document. Advertisement). В зависимости от типа ресурса, который представляет объявление, их можно подразделить на объявления групп (net.jxta.protocol.PeerGroupAdvertisement), объявления каналов (net.jxta.protocol.PipeAdvertisement) и т.д. Класс AdvertisementFactory (net.jxta.document.AdvertisementFactory) создает объявления. Это позволяет скрыть их реализацию.

7.6. Сервисы.

Все сервисы реализуют интерфейс Service (net. jxta. service. Service). Service — это приложение (application), поэтому он расширяет интерфейс Application. Поскольку объектами Service нельзя манипулировать непосредственно, доступ к ним производится через интерфейс сервиса. Например, интерфейс Discovery (net. jxta.

discovery. Discovery) представляет сервис обнаружения. Класс DiscoveryService (net.jxta.impl.discovery.DiscoveryService) представляет реализацию этого сервиса. Доступ к этому классу производится не напрямую, а через интерфейс DiscoveryInterface (net.jxta.impl.discovery.DiscoveryInterface).

8. Проекты сообщества ЖХТА

Как упоминалось ранее, проект ЖХТА — открытое сообщество, в котором участвуют разработчики open- source, технологические энтузиасты и студенты. Любой может принять участие в развитии технологии. На Jxta.org есть несколько интересных проектов. Рассмотрим некоторые из них:

8.1. ЖХТА Shell

ЖХТА Shell — пример приложения, основанного на реализации ЖХТА. оболочка предоставляет интерактивный доступ к среде ЖХТА через интерфейс командной строки. Она напоминает оболочку UNIX и предоставляет похожие команды для доступа к примитивам ЖХТА. оболочку можно загрузить с ее страницы, она легко устанавливается и конфигурируется. Команды оболочки напоминают команды UNIX. Поддерживается также оператор конвейера |.

Оболочка сконструирована так, что большинство команд отделены от оболочки и загружаются динамически при вызове. Это позволяет разработчикам легко добавлять новые команды. Вот несколько встроенных команд оболочки:

man -- система документации оболочки. Эта команда выдает список всех доступных команд оболочки. Подробную информацию о любой команде можно получить, набрав `man <имя команды>`.

clear -- очищает экран.

env — показывает значения переменных окружения оболочки. По умолчанию определены семь переменных:

1. `consin` — входной канал консоли по умолчанию
2. `consout` — выходной канал консоли по умолчанию
3. `stdin` — входной канал по умолчанию

JXTA решения для P2P

4. `stdout` — выходной канал по умолчанию
5. `stdgroup` — группа `peer` по умолчанию
6. `rootgroup` — сетевая группа `peer` по умолчанию
7. `shell` — оболочка `root`

setenv -- устанавливает переменную окружения оболочки

cat -- печатает содержимое объекта JXTA

whoami — выдает информацию о `peer` и группе `peer`. Без параметров выдает информацию о локальном `peer`.

rdvstatus -- выдает информацию обо всех рандеву `peer`, к которым подключен `peer` в данный момент. Также показывает, является ли сам `peer` рандеву `peer`.

peers -- используйте эту команду для поиска других `peer` в той же группе. Запуск команды без параметров выдает информацию о всех `peers`, уже известных локальному `peer`. Это возможно, поскольку оболочка кэширует объявления `peer`, обнаруженных ранее. Команда с параметром `g` посылает запрос на обнаружение, чтобы найти удаленных `peer`. Команда с параметром `f` очищает кэш объявлений.

groups -- то же, что и предыдущая команда, но для поиска групп.

importfile -- импортирует внешний файл в переменную окружения оболочки

exportfile -- экспортирует переменную окружения в файл.

mkadv -- создает объявление из переменной окружения. Это может быть объявление группы или канала.

mkgrp — создает новую группу, используя объявление `peer`. Если объявление не указано, оболочка создает клон сетевой группы.

join — команда для подключения к группе.

leave - команда для выхода из группы.

mkpipe -- создает входной или выходной канал.

talk -- команда для простого обмена сообщениями между двумя `peers`. Она состоит из трех шагов. Сначала пользователь регистрируется, благодаря чему создается `talk-`

объявление для данного пользователя — это нужно сделать только один раз. Регистрация происходит с помощью команды `talk — register < nick>`. После регистрации пользователь должен войти в систему с помощью команды `talk — login < nick>`. Теперь он может искать других пользователей командой `talk — search`. Когда он найдет другого пользователя, он может послать ему сообщение командой `talk -u <nick> <destinationUserName>`.

exit -- выход из оболочки.

Служба менеджера контента ЖХТА

Служба менеджера контента, сокращенно CMS, позволяет предоставление и получение контента, представленного уникальным идентификатором, внутри группы peer. CMS также использует объявления контента, которые предоставляют метаданные о контенте. Кроме того, сервис позволяет управлять контентом на локальном peer, а также просматривать и загружать контент с удаленных peers.

8.2. Instant P2P

Instant P2P — программа для мгновенной передачи сообщений, реализованная на ЖХТА. Она поддерживает разговор один на один, групповой чат, разделение файлов и т.д. Программа доступна для платформ Linux, Solaris и Win32. Устройства, поддерживающие Personal Java 3.1, также могут запускать программу.

Instant P2P может послужить хорошим примером для изучения ЖХТА. Пользователь может войти в систему, используя любой псевдоним по своему выбору. Он может искать группы, доступные в сети, и присоединиться к группам, к которым он захочет. Пользователи также могут видеть peers, являющиеся членами их группы. Опция `chat` позволяет говорить один на один, а `group chat` позволяет разговаривать в группе. Опция `share` позволяет разделять контент с другими членами группы. Пользователь может также проводить поиск контента в группе. Instant P2P использует CMS для разделения контента.

9. Перспективы ЖХТА

В этой статье я познакомил вас с системами peer- to- peer. Вы узнали, что ЖХТА

ЖХТА решения для P2P

предлагает для мира P2P. Я также описал несколько важных терминов и концепций, используемых программистами, создающими приложения для ЖХТА.

ЖХТА — перспективная низкоуровневая платформа для создания P2P приложений. Поскольку Java — предпочтительный язык для разработки приложений, рассчитанных на работу в неоднородных средах, это естественный выбор для P2P-приложений. Надеюсь, эта статья достаточно заинтересовала вас, чтобы начать исследовать мир P2P и Java.

10. Об авторе

Navaneeth Krishnan is senior product engineer at [Aztec Software and Technology Solutions](#), where he has designed and developed several e-commerce solutions and frameworks for clients. He is an active P2P enthusiast and currently deeply involved with Jxta. He owns iPeers, a Jxta community project that deals with the integration of artificial intelligence in P2P networks.

11. Ресурсы

- Visit the Project Jxta home page: <http://www.jxta.org>
- The Jxta Protocol Specification and other whitepapers are available here: http://www.jxta.org/white_papers.html
- The Jxta Shell project home page: <http://shell.jxta.org>
- The CMS home page: <http://cms.jxta.org>
- The InstantP2P home page: <http://instantp2p.jxta.org/>
- To learn more about scalability problems faced in the Gnutella network, read "Bandwidth Barriers to Gnutella Network Scalability" (*Clip2*, September 2000): http://www.clip2.com/dss_barrier.html
- Learn more about URIs, URLs, and URNs: <http://www.ietf.org/rfc/rfc2396.txt>
- "The Gnutella File-Sharing Network and Java," Ken McCrary (*JavaWorld*, October 2000): <http://www.javaworld.com/javaworld/jw-10-2000/jw-1006-fileshare.html>
- Mark Johnson reports from JavaOne 2001 on the motivation for a new P2P protocol in "Get Connected with Jxta" (*JavaWorld*, June 2001): <http://www.javaworld.com/javaworld/javaone01/j1-01-jxta.html>
- For more articles on **Enterprise Java**, browse the *JavaWorld* Topical Index:

http://www.javaworld.com/channel_content/jw-enterprise-index.shtml

- Sign up for *JavaWorld's* free weekly email newsletters: <http://www.idg.net/jw-subscribe>
- You'll find a wealth of IT-related articles from our sister publications at [IDG.net](http://www.idg.net)

Reprinted with permission from the October 2001 edition of JavaWorld magazine. Copyright © ITworld.com, Inc., an IDG Communications company.

View the original article at: <http://www.javaworld.com/javaworld/jw-10-2001/jw-1019-jxta.html>