

# 46

## Одноранговые сети

### **В ЭТОЙ ГЛАВЕ...**

- Общие сведения о технологии одноранговых сетей (peer-to-peer – P2P)
- Использование платформы одноранговых сетей Microsoft Windows 8
- Регистрация и преобразование имен равноправных участников
- Отправка и прием сообщений между равноправными участниками
- Построение приложений P2P с помощью .NET Framework

### ***Загружаемый код для этой главы***

Загружаемый код для этой главы содержит следующие основные примеры:

- P2PSample

## Обзор технологии P2P

Технология организации одноранговых сетей (peer-to-peer networking — P2P) является одной из самых полезных, но при этом часто неправильно понимаемых технологий, появившихся в последние годы. Когда люди думают о P2P, им на ум обычно приходит только возможность обмена музыкальными, видео, программными и другими файлами, зачастую незаконным образом. Это связано с тем, что приложения для обмена файлами наподобие BitTorrent стали очень популярными, а в них для работы используется именно технология P2P.

Однако, хотя технология P2P применяется в приложениях для обмена файлами, это вовсе не означает, что она не может использоваться в других приложениях. На самом деле, как будет показано в главе, эта технология может применяться во множестве других приложений, и она становится все более и более важной в современном мире повсеместных коммуникаций. В этом можно будет убедиться в первой части настоящей главы при кратком рассмотрении технологии P2P.

В Microsoft тоже не обошли стороной появление технологии P2P и стали разрабатывать собственные инструменты и технологии для ее применения. Так появилась платформа Microsoft Windows Peer-to-Peer Networking, исполняющая роль инфраструктуры для коммуникаций в приложениях P2P. В состав этой платформы входит такой важный компонент, как PNRP (Peer Name Resolution Protocol — протокол преобразования имен равноправных участников). Платформа .NET Framework содержит пространство имен `System.Net.PeerToPeer` и несколько типов и функциональных средств, которые можно использовать для построения приложений P2P с минимальными затратами.

Технология P2P представляет собой альтернативный подход к организации сетевых коммуникаций. Для того чтобы понять, чем P2P отличается от “стандартного” подхода к обеспечению коммуникаций, не помешает сделать шаг назад и вспомнить, что собой представляет связь типа “клиент-сервер”. Коммуникации такого типа повсеместно применяются в современных сетевых приложениях.

### Архитектура типа “клиент-сервер”

Традиционно взаимодействие с приложениями по сети (в том числе через Интернет) организуется с использованием архитектуры типа “клиент-сервер”. Прекрасным примером могут служить веб-сайты. При просмотре веб-сайта происходит отправка по Интернету соответствующего запроса веб-серверу, который затем возвращает требуемую информацию. Если необходимо загрузить какой-то файл, это делается напрямую с веб-сервера.

Аналогично, настольные приложения, имеющие возможность подключения к локальной или глобальной сети, обычно устанавливают соединение с каким-то одним сервером, например, сервером баз данных или сервером, предоставляющим набор служб.

На рис. 46.1 показана простая форма архитектуры типа “клиент-сервер”.

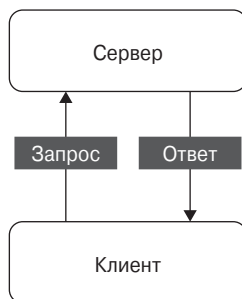
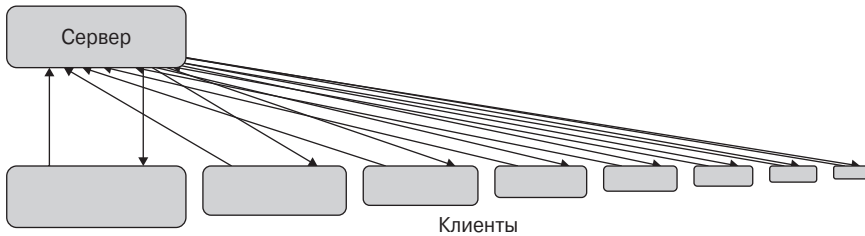


Рис. 46.1. Простая форма архитектуры типа “клиент-сервер”

Ничего принципиально плохого в такой архитектуре нет, и действительно во многих случаях она будет именно тем, что нужно. Однако ей присуща проблема, связанная с масштабированием. На рис. 46.2 показано масштабирование архитектуры типа “клиент-сервер” при добавлении дополнительных клиентов.



*Рис. 46.2. Масштабирование архитектуры типа “клиент-сервер” при добавлении дополнительных клиентов*

С добавлением клиентов нагрузка на сервер, который должен взаимодействовать с каждым клиентом, будет увеличиваться. Если снова вернуться к примеру веб-сайта, то такое увеличение нагрузки может стать причиной отказа функционирования веб-сайта. При слишком большом трафике сервер просто перестает реагировать на запросы.

Конечно, существуют варианты масштабирования, с помощью которых можно смягчить подобную ситуацию. Один из них предусматривает масштабирование “вверх” за счет увеличения мощности и ресурсов сервера, а другой — масштабирование “вширь” путем добавления дополнительных серверов. Первый способ, естественно, ограничивается доступными технологиями и стоимостью более мощного оборудования. Второй способ потенциально более гибок, но требует добавления в инфраструктуру дополнительного уровня для предоставления клиентам возможности либо взаимодействовать с отдельными серверами, либо поддерживать состояние сеанса независимо от сервера, с которым осуществляется взаимодействие. Для этого доступна масса решений, таких как средства, позволяющие создавать веб-фермы или фермы серверов.

## Архитектура P2P

Одноранговый подход полностью отличается от подхода с масштабированием “вверх” или “вширь”. В случае применения P2P вместо того, чтобы сосредоточить усилия на попытках улучшить коммуникации между сервером и его клиентами, все внимание уделяется поиску способов, которыми клиенты могут взаимодействовать между собой.

Давайте для примера представим, что веб-сайтом, с которым взаимодействуют клиенты, является `www.wrox.com`, а издательство Wrox объявило о выходе новой версии данной книги на этом сайте и предоставлении его для бесплатной загрузки всем желающим, но лишь на протяжении одного дня. Нетрудно догадаться, что при таком положении дел накануне появления книги веб-сайт начнет просматривать масса людей, которые будут постоянно обновлять его содержимое в своих браузерах в ожидании появления файла. Как только файл станет доступным, все они одновременно начнут пытаться загрузить его и, скорее всего, веб-сервер, который обслуживает веб-сайт `wrox.com`, не выдержит такого натиска и выйдет из строя.

Чтобы предотвратить выход веб-сервера из строя, можно воспользоваться технологией P2P. Вместо отправки файла прямо с сервера сразу всем клиентам он может быть отправлен только определенному числу клиентов. Несколько остальных клиентов могут далее загрузить его у тех клиентов, у которых он уже есть. После этого еще несколько клиентов могут загрузить его у клиентов, получивших его вторыми, и т.д. Этот процесс может происходить даже быстрее благодаря разбиению файла на части и распределению этих частей

среди клиентов, одни из которых будут загружать их прямо с сервера, а другие — из других клиентов. Именно так и работают технологии файлообменных систем вроде BitTorrent, как показано на рис. 46.3.

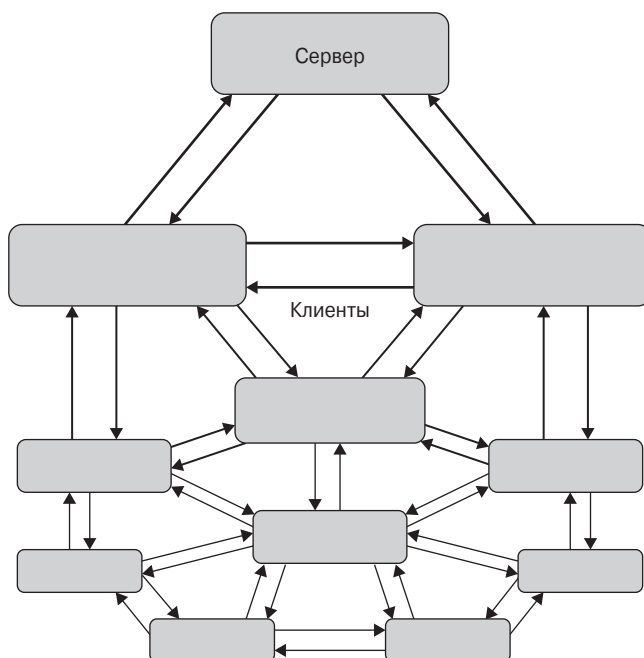


Рис. 46.3. Пример реализации архитектуры P2P

## Проблемы архитектуры P2P

Тем не менее, в описанной здесь архитектуре обмена файлами все равно остались кое-какие проблемы, которые должны быть решены. Для начала, каким образом клиенты узнают о том, что существуют другие клиенты, и как они будут обнаруживать фрагменты файла, которые, возможно, имеются у других клиентов? Кроме того, каким образом гарантировать оптимальное взаимодействие между клиентами, если их могут отделять друг от друга континенты?

Каждый клиент, участвующий в работе сетевого приложения P2P, для преодоления этих проблем должен быть способен выполнять следующие операции:

- обнаруживать других клиентов;
- подключаться к другим клиентам;
- обмениваться данными с другими клиентами.

В том, что касается способности обнаруживать других клиентов, возможны два очевидных решения: поддержка списка клиентов на сервере, чтобы клиенты могли получать его и связываться с другими клиентами (называемыми *равноправными участниками* (peer)), либо использование инфраструктуры (например, PNRP, о которой речь пойдет в следующем разделе), которая позволяет клиентам обнаруживать друг друга напрямую. В большинстве файлообменных систем применяется решение с поддержкой списка на сервере, и используются серверы, называемые *трекерами* (tracker). В файлообменных системах в роли сервера может также выступать и любой клиент, как показано на рис. 46.3, объявляя, что

у него имеется доступный файл, и регистрируя его на сервере-трекере. Фактически, в чистой сети P2P вообще не нужны никакие серверы, а лишь равноправные участники.

Проблема подключения к другим клиентам является более тонкой и распространяется на всю структуру сети, которая используется приложением P2P. При наличии одной группы клиентов, в которой все должны иметь возможность взаимодействовать друг с другом, топология соединений между этими клиентами может приобретать чрезвычайно сложный вид. Зачастую производительность удается улучшить за счет создания нескольких групп клиентов с возможностью установки подключения между клиентами в каждой из них, но не с клиентами в других группах. В случае создания этих групп по принципу локальности можно добиться дополнительного повышения производительности, поскольку в таком случае клиенты получают возможность взаимодействовать друг с другом по более коротким (с меньшим числом прыжков) сетевым путям между подключенными к сети компьютерами.

Способность обмена данными с другими клиентами, пожалуй, не так важна, поскольку существуют хорошо зарекомендовавшие себя протоколы вроде TCP/IP, которые вполне могут применяться и здесь. Однако существует возможность улучшения, как в высокоуровневых технологиях (например, можно использовать службы WCF, получая в свое распоряжение все предлагаемые ими функциональные возможности), так и в низкоуровневых протоколах (например, применяя протоколы многоадресной рассылки и тем самым обеспечивая отправку данных во множество конечных точек одновременно).

Обеспечение клиентов возможностью обнаруживать, подключаться и взаимодействовать друг с другом играет центральную роль в любой реализации P2P. Реализация, рассматриваемая в настоящей главе, предполагает применение типов `System.Net.PeerToPeer` вместе с PNM для обеспечения возможности обнаружения других клиентов и PNRP для обеспечения возможности подключения к другим клиентам. Как будет показано в последующих разделах, эти технологии обеспечивают возможность выполнения всех трех необходимых операций.

## Терминология P2P

В предыдущих разделах уже было представлено понятие *равноправного участника* — именно так называют клиентов в сети P2P. В сети P2P слово *клиент* не имеет никакого смысла, потому что здесь нет обязательного сервера, клиентом которого нужно быть.

Группы равноправных участников, которые соединены друг с другом, называются *ячейками* (mesh), *облаками* (cloud) или *графами* (graph). Каждая отдельная группа считается *хорошо соединенной*, если соблюдено хотя бы одно из перечисленных далее условий.

- Между каждой парой равноправных участников существует путь соединения, позволяющий каждому участнику подключаться к другому равноправному участнику требуемым образом.
- Между каждой парой равноправных участников существует относительно небольшое количество соединений, по которым они могут связываться.
- Удаление одного равноправного участника из группы не лишает остальных равноправных участников возможности соединения друг с другом.

Это вовсе не означает, что каждый равноправный участник должен обязательно иметь возможность подключаться к каждому другому равноправному участнику напрямую. Если проанализировать сеть с математической точки зрения, то можно обнаружить, что для соблюдения упомянутых выше условий равноправным участникам необходимо иметь возможность подключаться к относительно небольшому количеству других равноправных участников.

Еще одним понятием в технологии P2P, о котором следует знать, является *волновое распространение* (flooding). Под волновым распространением подразумевается способ, кото-

рым один фрагмент данных может передаваться по сети всем равноправным участникам и которым может производиться опрос других узлов в сети для обнаружения конкретного фрагмента данных. В неструктурированных сетях P2P этот процесс является достаточно произвольным; при этом сначала устанавливается связь с ближайшими соседними равноправными участниками, которые затем, в свою очередь, связываются со своими ближайшими соседями, и т.д. до тех пор, пока не будет охвачен каждый равноправный участник в сети. Также допускается создавать и структурированные сети P2P с четко определенными путями, по которым должно происходить распространение запросов и данных среди равноправных участников.

## Решения P2P

При наличии подходящей инфраструктуры для P2P можно начинать разрабатывать не просто улучшенные версии клиент-серверных приложений, но и совершенно новые приложения. Технология P2P особенно подходит для приложений следующих классов:

- приложения, предназначенные для распространения содержимого, в том числе упомянутые ранее приложения обмена файлами;
- приложения, предназначенные для совместной работы, такие как приложения, которые позволяют открывать общий доступ к рабочему столу и “белой доске” (whiteboard);
- приложения, предназначенные для обеспечения многопользовательской связи и позволяющие пользователям общаться и обмениваться данными напрямую, а не через сервер;
- приложения, предназначенные для распределения обработки и служащие альтернативой приложениям для суперкомпьютеров, которые обрабатывают огромные объемы данных;
- приложения Web 2.0, объединяющие в себе некоторые или все перечисленные выше приложения и превращающие их в динамические веб-приложения следующего поколения.

## Протокол PNRP

Платформа для создания одноранговых сетей Windows производства Microsoft (Microsoft Windows Peer-to-Peer Networking) представляет собой реализацию технологии P2P от Microsoft. Она является составной частью Windows, начиная с Windows XP SP2. Протокол PNRP можно применять для публикации и преобразования адресов равноправных участников сети. Этот протокол более подробно рассматривается далее в разделе.

Для реализации приложения P2P можно использовать любой из имеющихся в распоряжении протоколов, но при работе в среде Microsoft Windows 8 имеет смысл, по меньшей мере, подумать о применении протокола PNRP. Сам по себе протокол PNRP не предоставляет всего, что необходимо для создания приложения P2P. Скорее, он является лишь одной из основополагающих технологий, которые применяются для преобразования адресов равноправных участников сети P2P. Протокол PNRP позволяет клиенту регистрировать конечную точку (называемую *именем равноправного участника* (peer name)), которая автоматически распространяется между остальными равноправными участниками в облаке (группе). Это имя инкапсулируется в идентификатор PNRP (PNRP ID). Любой равноправный участник, который обнаруживает идентификатор PNRP, может использовать PNRP для его преобразования в фактическое имя равноправного участника и затем начать взаимодействие с соответствующим клиентом напрямую.

Например, можно определить имя равноправного участника, представляющее конечную точку службы WCF, и воспользоваться PNRP для его регистрации в группе (облаке)

в виде идентификатора PNRP ID. После этого равноправный участник, выполняющий клиентское приложение, которое применяет механизм обнаружения, способный распознавать имена равноправных участников для предоставляемой службы, сможет обнаружить этот идентификатор PNRP ID. После обнаружения равноправный участник может использовать PNRP для обнаружения конечной точки службы WCF и последующей работы с этой службой.

**На заметку!** В протоколе PNRP не делается никаких допущений о том, что конкретно представляет имя равноправного участника. Равноправные участники должны сами решить, как использовать эти имена, после того как они обнаружены. Информация, которую равноправный участник получает от PNRP при преобразовании идентификатора PNRP, включает в себя адрес IPv6, а также обычно и адрес IPv4 участника, который опубликовал этот идентификатор, вместе с номером порта и (необязательно) небольшим количеством дополнительных данных. Без знания того, что означает имя равноправного участника, эта информация, скорее всего, окажется бесполезной.

## Идентификаторы PNRP

Идентификаторы PNRP ID имеют длину 256 битов. Младшие 128 битов используются для уникальной идентификации отдельного равноправного участника сети, а старшие 128 битов — для обозначения его имени. Старшие 128 битов представляют собой хеш-комбинацию, которая состоит из хешированного значения открытого ключа, полученного от публикуемого равноправного участника, и строки длиной до 149 символов, которая обозначает имя этого участника. Хешированный открытый ключ (называемый *авторитетным источником*) в сочетании с этой строкой (называемой *классификатором*) образует идентификатор P2P. Вместо хешированного открытого ключа также может использоваться значение 0, в случае чего имя равноправного участника считается *незащищенным* (если применяется открытый ключ, то имя считается *защищенным*).

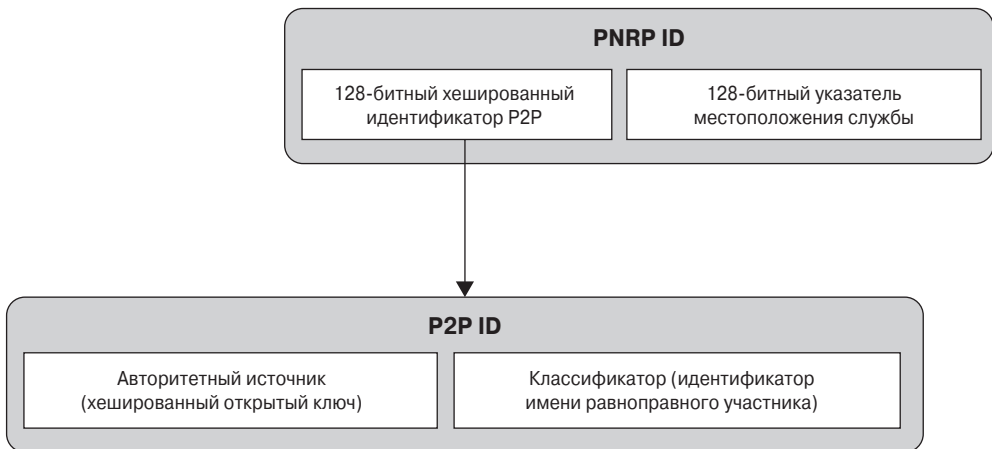


Рис. 46.4. Структура идентификатора PNRP

Служба PNRP на стороне равноправного участника отвечает за поддержание списка идентификаторов PNRP, как тех, которые публикует сама, так и тех, которые получает в виде кешированного списка от экземпляров служб PNRP, находящихся в других местах облака. Когда равноправный участник пытается выполнить преобразование идентификатора

PNRP, служба PNRP либо использует кешированную копию конечной точки для выяснения адреса того равноправного узла, который опубликовал данный идентификатор PNRP, либо спрашивает его соседей, не могут ли они сделать это. Рано или поздно соединение с публикующим равноправным участником устанавливается, и служба PNRP получает возможность выполнить преобразование идентификатора PNRP.

Все это не требует никакого внешнего вмешательства. Понадобится только позаботиться о том, чтобы равноправные участники знали, что следует делать с именами после их преобразования с помощью своей локальной службы PNRP.

Равноправные участники сети могут применять PNRP для нахождения идентификаторов PNRP, совпадающих с определенным идентификатором P2P. Эту возможность можно использовать для реализации простейшего механизма, позволяющего обнаруживать незащищенные имена равноправных участников. Дело в том, что в случае отображения несколькими равноправными участниками незащищенного имени с одинаковым классификатором, идентификатор P2P ID у них будет совпадать. Разумеется, из-за того, что любой равноправный участник может применять незащищенное имя, нет никакой гарантии, что конечная точка, с которой устанавливается соединение, будет именно той, которая ожидается, поэтому такое решение подходит лишь для реализации механизма обнаружения по локальной сети.

## Облака PNRP

Выше было рассказано, как PNRP осуществляет регистрацию и преобразование имен равноправных участников в облаке (группе). Облако поддерживается *порождающим сервером*, или сид-сервером (*seed server*), которым может быть любой сервер с запущенной службой PNRP, поддерживающей запись хотя бы об одном равноправном участнике. Для службы PNRP доступны облака двух типов.

- **Облака локальных соединений (link local).** В такие облака входят компьютеры с подключением к локальной сети. Каждый ПК может подключаться более чем к одному облаку такого типа при условии наличия у него нескольких сетевых адаптеров.
- **Глобальные облака (global).** В такие облака по умолчанию входят компьютеры с подключением к Интернету, хотя также можно определять частное глобальное облако. Различие между ними в том, что для глобального облака с подключением к Интернету Microsoft поддерживает сид-сервер, а для определяемого самостоятельно частного глобального облака должен использоваться собственный сид-сервер. В последнем случае необходимо позаботиться о том, чтобы все равноправные участники могли подключаться к нему, настроив соответствующим образом параметры политики.

**На заметку!** В предшествующих версиях PNRP существовали облака еще и третьего типа, которые назывались облаками локальных сайтов (*site local*). Они больше не применяются и поэтому здесь не рассматриваются.

Для выяснения того, в какие облака входит данный компьютер, служит следующая команда:

```
netsh p2p pnrp cloud show list
```

Типичный результат ее выполнения выглядит следующим образом:

| Scope | Id   | Addr | State   | Name                  |
|-------|------|------|---------|-----------------------|
| ----  | ---- | ---- | -----   | -----                 |
| 1     | 0    | 1    | Virtual | Global_               |
| 3     | 13   | 1    | Virtual | LinkLocal_ff00::%13/8 |
| 3     | 19   | 1    | Virtual | LinkLocal_ff00::%19/8 |

Как видите, в данном случае доступны три облака: одно глобальное и два облака локальных соединений. Понять это можно как по значению в столбце Name (Имя), так и по значению в столбце Scope (Область), в котором для облаков локальных соединений всегда отображается значение 3, а для глобальных облаков — значение 1. Для подключения к глобальному облаку необходимо иметь глобальный адрес IPv6.

Облака могут находиться в одном из перечисленных ниже состояний.

- **active** (активное). Если облако находится в состоянии **active**, оно может применяться для публикации и преобразования имен равноправных участников.
- **alone** (одинокое). Если к облаку, из которого запрашивается равноправный участник, больше никаких равноправных участников не подключено, оно будет находиться в состоянии **alone**.
- **no net** (без сети). Если равноправный участник потерял соединение с сетью, состояние облака может измениться с **active** на **no net**.
- **synchronizing** (синхронизация). Облака находятся в состоянии **synchronizing** во время подключения к ним какого-то равноправного участника. Это состояние будет сменяться другим очень быстро, поскольку подключение не занимает много времени. Поэтому увидеть облако в таком состоянии удастся редко.
- **virtual** (виртуальное). Служба PNRP подключается к облакам только при возникновении потребности в регистрации или преобразовании имени какого-то равноправного участника. Если соединение с облаком находится в неактивном состоянии дольше 15 минут, облако может перейти в состояние **virtual**.

**На заметку!** При наличии проблем с сетевой связностью следует проверить, не препятствует ли брандмауэр передаче локального сетевого трафика через UDP-порты 3540 или 1900. UDP-порт 3540 используется протоколом PNRP, а UDP-порт 1900 — протоколом SSDP (Simple Service Discovery Protocol — простой протокол обнаружения служб), который, в свою очередь, применяется службой PNRP (а также устройствами UPnP).

## Протокол PNRP, начиная с Windows 7

Начиная с версии Windows 7, в протоколе PNRP используется компонент, называемый *DRT* (*Distributed Routing Table* — таблица распределенной маршрутизации). Этот компонент отвечает за определение структуры применяемых протоколом PNRP ключей, роль которой в реализации по умолчанию исполняет описанный выше идентификатор PNRP. С помощью API-интерфейса DRT можно определить альтернативную схему ключей при условии, что они представляют собой 256-битные целочисленные значения (подобно идентификаторам PNRP ID). Это означает возможность использования любой схемы, но при этом, разумеется, нужно самостоятельно заботиться о генерации и защите ключей. За счет применения этого компонента можно создавать совершенно новые топологии облаков, выходящие за рамки действия PNRP и, следовательно, за рамки контекста настоящей главы, поскольку этот прием относится к числу сложных.

В Windows 7 также появилась новая опция для подключения к другим пользователям в приложении Remote Assistance (Удаленный помощник), которая называется Easy Connect (Легкое подключение). Эта опция использует PNRP для обнаружения пользователей, к которым необходимо подключиться. После создания сеанса с помощью Easy Connect или других средств (например, отправки приглашения по электронной почте) пользователи могут открыть общий доступ к своим рабочим столам и оказывать помощь друг другу через интерфейс Remote Assistance.

## Создание приложений P2P

Теперь, когда известно, что собой представляет технология P2P, и какие технологии доступны разработчикам приложений .NET для реализации приложений P2P, пришло время посмотреть, как создавать такие приложения. Из приведенного выше материала понятно, что в любом приложении PNRP должен применяться протокол PNRP для публикации, распространения и преобразования имен равноправных участников. Поэтому вначале нужно разобраться, как достичь этого, используя .NET. Затем будет продемонстрировано применение PNM в качестве инфраструктуры для приложения P2P. Последнее может обеспечить преимущество в том, что в случае применения PNM реализовать собственные механизмы обнаружения не понадобится.

Перед изучением этих тем сначала необходимо ознакомиться с классами, которые предлагаются в пространстве имен `System.Net.PeerToPeer`. Чтобы работать с этими классами, нужно обязательно ссылаться на сборку `System.Net`.

Классы в пространстве имен `System.Net.PeerToPeer` инкапсулируют API-интерфейс для PNRP и позволяют взаимодействовать со службой PNRP. Их можно применять для решения двух основных задач:

- регистрации имен равноправных участников;
- преобразования имен равноправных участников.

В последующих разделах все упоминаемые типы относятся к пространству имен `System.Net.PeerToPeer`, если только явно не указано другое.

### Регистрация имен равноправных участников

Для регистрации имени равноправного участника выполните следующие шаги.

1. Создайте защищенное или незащищенное имя равноправного участника с указанным классификатором.
2. Настройте процесс регистрации этого имени, предоставив следующие необязательные сведения по своему выбору.
  - Номер порта TCP.
  - Облако или облака, в которых нужно зарегистрировать имя равноправного участника. (Если они не указаны, PNRP регистрирует имя равноправного участника во всех доступных облаках.)
  - Комментарий длиной до 39 символов.
  - Дополнительные данные объемом до 4096 байт.
  - Информация о том, должны ли для имени равноправного участника автоматически генерироваться конечные точки (стандартное поведение, при котором конечные точки автоматически генерируются на основе IP-адреса или адресов равноправного участника и номера порта, если таковой указан).
  - Коллекция конечных точек.
3. Зарегистрируйте имя равноправного участника в локальной службе PNRP.

После выполнения третьего шага имя равноправного участника становится доступным для всех соседей в выбранном облаке (или облаках). Регистрация равноправного участника продолжается до тех пор, пока не будет прекращена явным образом или не будет завершен процесс, зарегистрировавший имя равноправного участника.

Для создания имени равноправного участника применяется класс `PeerName`. Экземпляр этого класса создается из строкового представления идентификатора P2P ID в форме `авторитетный_источник.классификатор` либо из строки классификатора и типа `PeerNameType`.

Можно использовать как тип `PeerNameType.Secured`, так и тип `PeerNameType.Unsecured`. Например:

```
var pn = new PeerName("Peer classifier", PeerNameType.Secured);
```

Поскольку в незащищенном имени равноправного участника значением авторитетного источника является 0, следующие строки кода эквивалентны:

```
var pn = new PeerName("Peer classifier", PeerNameType.Unsecured);
var pn = new PeerName("0.Peer classifier");
```

После создания экземпляр `PeerName` можно использовать вместе с номером порта для инициализации объекта `PeerNameRegistration`:

```
var pnr = new PeerNameRegistration(pn, 8080);
```

В качестве альтернативного варианта для объекта `PeerNameRegistration`, создаваемого с использованием его стандартного параметра, можно установить свойство `PeerName` и (необязательно) свойство `Port`. Кроме того, желаемый экземпляр `Cloud` можно передать либо в третьем параметре конструктору `PeerNameRegistration`, либо указать с помощью свойства `Cloud`. Экземпляр `Cloud` можно получить либо из имени облака, либо с применением одного из следующих статических членов класса `Cloud`.

- `Cloud.Global`. Это статическое свойство позволяет получить ссылку на глобальное облако, которое в зависимости от конфигурации политики равноправных участников может быть частным глобальным облаком.
- `Cloud.AllLinkLocal`. Это статическое поле позволяет получить облако, которое содержит все облака локальных соединений, доступные для данного равноправного участника.
- `Cloud.Available`. Это статическое поле позволяет получить облако, которое содержит все облака, доступные для данного равноправного участника, т.е. локальные и глобальные (если таковые имеются).

После создания экземпляра `PeerNameRegistration` можно при желании установить его свойства `Comment` и `Data`. При этом следует помнить об ограничениях этих свойств. Если попытаться установить для свойства `Comment` строку длиной более 39 символов `Unicode`, будет сгенерировано исключение `PeerToPeerException`, а при попытке установить для свойства `Data` байтовый массив размером более 4096 байт — исключение `ArgumentOutOfRangeException`. Можно также с помощью свойства `EndPointCollection` добавить конечные точки. Это свойство представляет собой коллекцию типа `System.Net.IPEndPointCollection` с объектами `System.Net.IPEndPoint`. В случае применения свойства `EndPointCollection` может понадобиться установить свойство `UseAutoEndPointSelection` в `false`, чтобы предотвратить автоматическую генерацию конечных точек.

После того как все готово к регистрации имени равноправного участника, можно вызвать метод `PeerNameRegistration.Start`. Для удаления записи о регистрации имени равноправного участника из службы PNRP служит метод `PeerNameRegistration.Stop`.

Ниже приведен пример регистрации защищенного имени равноправного участника вместе с комментарием:

```
var pn = new PeerName("Peer classifier", PeerNameType.Unsecured);
var pnr = new PeerNameRegistration(pn, 8080);
pnr.Comment = "Get pizza here";
pnr.Start();
```

## Преобразование имен равноправных участников

Для преобразования имени равноправного участника необходимо выполнить следующие шаги.

1. Сгенерируйте имя равноправного участника на основе известного идентификатора P2P либо идентификатора P2P ID, полученного посредством операции обнаружения.
2. Воспользуйтесь распознавателем для преобразования имени равноправного участника и получения коллекции записей с именами равноправных участников. Распознаватель можно ограничить отдельным облаком и/или максимальным количеством возвращаемых результатов.
3. Из любой полученной записи с именем равноправного участника извлеките само имя, а также необходимую информацию о конечной точке, комментарии и дополнительные данные, если таковые присутствуют.

Подобно процессу регистрации имени равноправного участника, этот процесс начинается с создания объекта `PeerName`. Отличие состоит в том, что здесь используется имя равноправного участника, которое уже было зарегистрировано одним или более удаленными участниками. Простейшим способом для получения списка активных равноправных участников в локальном облаке является регистрация незащищенного имени для каждого равноправного участника с одним и тем же классификатором и применение этого имени на этапе преобразования. Однако для глобальных облаков пользоваться такой стратегией не рекомендуется, поскольку незащищенные имена могут быть легко подделаны.

Для преобразования имен равноправных участников служит класс `PeerNameResolver`. Имея экземпляр этого класса, можно выбрать, как должно выполняться преобразование — синхронно с применением метода `Resolve()` или же асинхронно с помощью метода `ResolveAsync()`.

Метод `Resolve()` можно вызывать с указанием как лишь одного параметра `PeerName`, так и дополнительно экземпляра `Cloud`, в котором должно выполняться преобразование, максимального количества возвращаемых записей с именами равноправных участников в виде целого числа, или того и другого вместе. Этот метод возвращает экземпляр `PeerNameRecordCollection`, представляющий собой коллекцию объектов `PeerNameRecord`. Например, ниже показан пример преобразования незащищенного имени равноправного участника во всех локальных облаках с возвратом максимум 5 результатов:

```
var pn = new PeerName("0.Peer classifier");
var pnres = new PeerNameResolver();
PeerNameRecordCollection pnrc = pnres.Resolve(pn, Cloud.AllLinkLocal, 5);
```

Метод `ResolveAsync()` использует стандартную схему вызова асинхронного метода. Ему передается уникальный объект `userState` и организуется прослушивание на предмет поступления событий `ResolveProgressChanged`, свидетельствующих об обнаружении равноправных участников и выполнении преобразования, и события `ResolveCompleted`, сигнализирующего о прекращении работы метода. Метод `ResolveAsyncCancel()` позволяет отменить любой ожидающий выполнения асинхронный запрос.

В обработчиках события `ResolveProgressChanged` используется параметр аргументов события `ResolveProgressChangedEventArgs`, унаследованный от стандартного класса `System.ComponentModel.ProgressChangedEventArgs`. Свойство `PeerNameRecord` объекта аргумента события, получаемого в обработчике событий, можно применять для получения ссылки на запись с именем равноправного участника, которая была обнаружена.

Аналогичным образом для события `ResolveCompleted` требуется обработчик, принимающий параметр типа `ResolveCompletedEventArgs`, который является производным от `AsyncCompletedEventArgs`. Этот тип имеет параметр `PeerNameRecordCollection`, который можно использовать для получения полного списка записей с именами обнаруженных равноправных участников.

В следующем коде показан пример реализации обработчиков для этих событий:

```
private pnres_ResolveProgressChanged(object sender,
ResolveProgressChangedEventArgs e)
{
    // Использование e.ProgressPercentage (унаследованного
    // от базовых классов аргументов события).
    // Обработка PeerNameRecord из e.PeerNameRecord.
}

private pnres_ResolveCompleted(object sender,
ResolveCompletedEventArgs e)
{
    // Проверка наличия e.IsCancelled и e.Error
    // (унаследованных от базовых классов аргументов события).
    // Обработка PeerNameRecordCollection из e.PeerNameRecordCollection.
}
```

После получения одного или более объектов `PeerNameRecord` можно переходить к их обработке. Этот класс `PeerNameRecord` предоставляет в наше распоряжение свойства `Comment` и `Data` для изучения комментариев и дополнительных данных, которые были добавлены при регистрации имени равноправного участника (если были), свойство `PeerName` для извлечения объекта `PeerName` и записи с именем участника и, что наиболее важно, свойство `EndPointCollection`. Как и в случае с `PeerNameRegistration`, здесь это свойство тоже представляет собой коллекцию типа `System.Net.IPEndPointCollection`, содержащую объекты `System.Net.IPEndPoint`. Эти объекты можно применять для подключения к предоставляемым равноправным участником конечным точкам любым желаемым образом.

## Безопасность доступа кода в `System.Net.PeerToPeer`

Пространство имен `System.Net.PeerToPeer` также включает два следующих класса, которые можно использовать вместе с моделью CAS (Code Access Security — безопасность доступа кода), рассматриваемой в главе 22:

- `PnrpPermission`, унаследованный от класса `CodeAccessPermission`;
- `PnrpPermissionAttribute`, унаследованный от класса `CodeAccessSecurityAttribute`.

Эти классы удобно применять для обеспечения возможности настройки доступа PNRP с помощью полномочий обычным для CAS образом.

## Пример приложения

Загружаемый код для этой главы содержит пример приложения P2P, которое называется `P2PSample` и иллюстрирует применение описанного в настоящем разделе пространства имен и связанных с ним концепций. Это приложение WPF, в котором для конечной точки равноправного участника используется служба WCF.

Конфигурационные настройки этого приложения содержатся в конфигурационном файле и указывают, как должно выглядеть имя равноправного участника и какой порт должен прослушиваться (файл `App.config`):

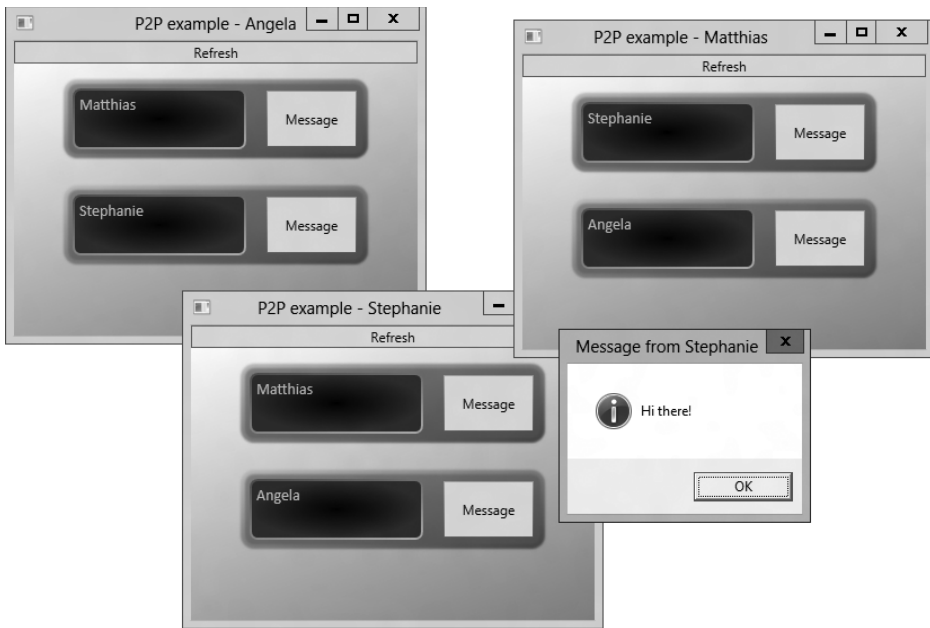
```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="username" value="Christian" />
    <add key="port" value="8731" />
  </appSettings>
</configuration>
```

После построения этого приложения можно приступить к его тестированию, либо скопировав его на другие компьютеры в локальной сети и запустив все его экземпляры, либо

запустив множество его экземпляров на одном компьютере. В последнем варианте нужно будет поменять используемый для каждого экземпляра порт, внося соответствующие изменения в отдельные конфигурационные файлы (понадобится скопировать содержимое каталога Debug на свой локальный компьютер и по очереди отредактировать каждый конфигурационный файл). В обоих вариантах результаты будут выглядеть яснее, если изменить также имя пользователя в каждом экземпляре.

После запуска всех экземпляров приложения можно щелкнуть на кнопке Refresh (Обновить), чтобы получить список доступных равноправных участников асинхронным образом. Обнаружив в этом списке нужного участника, можно отправить ему стандартное сообщение, щелкнув на кнопке Message (Сообщение).

На рис. 46.5 показан пример того, как приложение выглядит в действии при запуске трех его экземпляров на одном компьютере. На рисунке видно, что один равноправный участник только что отправил сообщение другому равноправному участнику, и оно отобразилось в диалоговом окне.



*Рис. 46.5. Приложение P2PSample в действии*

Теперь обратимся к коду. Поля-члены класса MainWindow (файл MainWindow.xaml.cs) позволяют определить доступную для просмотра коллекцию, содержащую всех равноправных участников. В конструкторе класса в коллекцию добавлен один элемент PeerEntry, предоставляющий информацию пользователю, которому достаточно щелкнуть на кнопке Refresh, чтобы получить список всех равноправных участников.

```
public partial class MainWindow : Window
{
    private P2PService localService;
    private ServiceHost host;
    private PeerName peerName;
    private PeerNameRegistration peerNameRegistration;
    private ObservableCollection<PeerEntry> peerList =
        new ObservableCollection<PeerEntry>();
    private object peersLock = new object();
```

```

public MainWindow()
{
    InitializeComponent();
    this.DataContext = peerList;
    peerList.Add(
        new PeerEntry
        {
            DisplayString = "Refresh to look for peers.",
            ButtonsEnabled = false
        });
    BindingOperations.EnableCollectionSynchronization(peerList, peersLock);
}

```

В этом приложении большая часть работы выполняется в обработчике события `Window_Loaded` окна `MainWindow`. В этом методе сначала производится загрузка конфигурационной информации и установка в заголовке окна имени пользователя:

```

private void Window_Loaded(object sender, RoutedEventArgs e)
{
    // Получение конфигурационной информации из App.config.
    string port = ConfigurationManager.AppSettings["port"];
    string username = ConfigurationManager.AppSettings["username"];
    string machineName = Environment.MachineName;
    string serviceUrl = null;

    // Установка заголовка окна.
    this.Title = string.Format("P2P example - {0}", username);
}

```

Далее адрес хоста равноправного участника и сконфигурированный порт применяются для определения конечной точки, в которой должна предоставляться служба WCF. Служба использует привязку `NetTcpBinding`, поэтому в URL-адресе конечной точки применяется протокол `net.tcp`:

```

// Получение URL-адреса службы с использованием адреса IPv4
// и порта из конфигурационного файла.
serviceUrl = Dns.GetHostAddresses(Dns.GetHostName())
    .Where(address => address.AddressFamily == AddressFamily.InterNetwork)
    .Select(address =>
        string.Format("net.tcp://{0}:{1}/P2PService", address, port))
    .FirstOrDefault();

```

Затем производится проверка достоверности полученного URL-адреса конечной точки, после чего выполняется регистрация и запуск службы WCF:

```

// Выполнение проверки, не является ли адрес нулевым.
if (serviceUrl == null)
{
    // Отображение ошибки и завершение работы приложения.
    MessageBox.Show(this, "Unable to determine WCF endpoint.",
        "Networking Error", MessageBoxButton.OK, MessageBoxImage.Stop);
    Application.Current.Shutdown();
}

// Регистрация и запуск службы WCF.
localService = new P2PService(this, username);
host = new ServiceHost(localService, new Uri(serviceUrl));
var binding = new NetTcpBinding();
binding.Security.Mode = SecurityMode.None;
host.AddServiceEndpoint(typeof(IP2PService), binding, serviceUrl);
try
{
    host.Open();
}
}

```

```

catch (AddressAlreadyInUseException)
{
    // Отображение ошибки и завершение работы приложения
    MessageBox.Show(this, "Cannot start listening, port in use.",
        "WCF Error", MessageBoxButton.OK, MessageBoxImage.Stop);
    Application.Current.Shutdown();
}

```

Единственный экземпляр класса службы позволяет легко реализовать взаимодействие между приложением-хостом и службой (для отправки и получения сообщений). Кроме того, для простоты в конфигурации привязки обеспечение безопасности отключено.

Затем производится регистрация имени равноправного участника с помощью классов из пространства имен `System.Net.PeerToPeer`:

```

// Создание имени равноправного участника.
peerName = new PeerName("P2P Sample", PeerNameType.Unsecured);

// Подготовка регистрации имени равноправного участника
// в облаке локального соединения.
peerNameRegistration = new PeerNameRegistration(peerName, int.Parse(port));
peerNameRegistration.Cloud = Cloud.AllLinkLocal;

// Начало регистрации.
peerNameRegistration.Start();
}

```

После щелчка на кнопке **Refresh** в обработчике события `RefreshButton_Click()` вызывается метод `PeerNameResolver.ResolveAsync()` для обнаружения соседних равноправных участников асинхронным образом:

```

private async void RefreshButton_Click(object sender, RoutedEventArgs e)
{
    // Создание распознавателя и добавление обработчиков событий.
    var resolver = new PeerNameResolver();
    resolver.ResolveProgressChanged +=
        new EventHandler<ResolveProgressChangedEventArgs>(
            resolver_ResolveProgressChanged);
    resolver.ResolveCompleted +=
        new EventHandler<ResolveCompletedEventArgs>(
            resolver_ResolveCompleted);

    // Подготовка новых равноправных участников.
    peerList.Clear();
    RefreshButton.IsEnabled = false;

    // Распознавание незащищенных равноправных участников асинхронным образом.
    resolver.ResolveAsync(new PeerName("0.P2P Sample"), 1);
}

```

При получении информации о равноправных участниках генерируются события `ResolveProgressChanged` и `ResolveCompleted`. На тот случай, если равноправные участники еще не активны, определен таймаут для прекращения процесса распознавания и генерации события `ResolveCompleted`. Обработка таймаута выполняется методом `Task.Delay()`, и по его истечении `ResolveAsyncCancel()` вызывается с тем же значением состояния пользователя, которое было передано методу `ResolveAsync()`. В результате эта же задача преобразования имени выбирается для отмены:

```

await Task.Delay(5000);
resolver.ResolveAsyncCancel(1);
}

```

Остальная часть кода отвечает за отображение и взаимодействие с равноправными участниками; ее можно изучить самостоятельно.

Предоставление конечных точек WCF через облака P2P является замечательным способом для обеспечения возможности установки местонахождения служб внутри предприятия, а также налаживания связи между равноправными участниками сети подобно тому, как было сделано в рассмотренном примере.

## Резюме

В этой главе было показано, как реализовать в приложениях функциональность одноранговых сетей (P2P) с применением классов P2P.

В частности, здесь рассматривались различные типы решений, которые позволяет создавать технология P2P, и то, как структурированы эти решения. Кроме того, демонстрировалось применение протокола PNRP с типами из пространств имен `System.Net.PeerToPeer`. Наконец, вы узнали об исключительно полезной методике предоставления служб WCF в качестве конечных точек P2P.

В следующей главе будет рассказано о создании очередей сообщений, как с помощью классов из пространства имен `System.Messaging`, так и посредством WCF.