Механизм работы DHT  
Реализация распределённой сети в BitTorrent-клиентах основана на варианте DHT, называемом Kademlia. А вообще говоря, DHT (Distributed hash table) означает децентрализованную распределённую систему для объединения большого количества постоянно исчезающих и появляющихся узлов и эффективной передачи сообщений между ними. На основе структур DHT строят разные более сложные системы, такие как файлообмен P2P, кооперативное веб-кэширование, службы DNS и т. п.  
DHT использует протокол UDP. Клиенты BitTorrent «слушают» тот же номер порта UDP, который они используют для входящих TCP-соединений. Если вы активно используете DHT, то открытие этого UDP-порта для доступа снаружи желательно, но не обязательно — DHT будет работать и так.  
Каждый подключённый клиент является в сети DHT отдельным узлом. У него есть свой уникальный ID (идентификатор), случайно выбираемый из того же 160-битного пространства, что и infohash’и торрентов.  
Каждый узел хранит таблицу маршрутизации, содержащую контактную информацию о многих «ближайших» к нему узлах, и о нескольких более далёких. «Близость» двух узлов вычисляется из «сходства» их ID, и не имеет никакого отношения к их географической близости.  
Когда узел хочет найти пиров для раздачи, он сравнивает infohash этой раздачи с ID известных ему узлов, и затем посылает запрос тому узлу, чей ID наиболее похож на этот infohash. Тот узел возвращает ему адрес узла, чей ID ещё ближе к infohash торрента.  
Тогда наш узел посылает запрос тому новому узлу, и получает от него адрес следующего узла, чей ID ещё более похож на infohash торрента.  
Таким образом, запросы от клиентов, участвующих в раздаче торрента с определённым infohash, постепенно стекаются к узлам, чьи ID наиболее похожи на этот infohash. Эти узлы помнят предыдущие запросы, и всем следующим запрашивающим узлам вернут адреса предыдущих пиров с той же раздачи.  
Протокол DHT  
Введение  
Каждый узел имеет уникальный идентификатор, называемый «node ID». Этот идентификатор случайным образом выбирается из того же 160-битного диапазона значений, что и infohash используемый в BitTorrent протоколе. «Distance metric» (расстояние) используется для сравнения «node ID» двух узлов или «node ID» узла с «infohash» для определения их «близости». Узлам необходимо хранить таблицу маршрутизации, содержащую контактную информацию о небольшом количестве других узлов. Таблица маршрутизации постепенно наполняется идентификаторами, близкими к идентификатору данного узла. Узлам известно о множестве других узлов в DHT, идентификаторы которых «близки» к данному узлу, и лишь о нескольких «далёких».  
В Kademlia, метрикой (функцией, определяющей дистанцию между элементами упорядоченного множества) является XOR и результат интерпретируется как беззнаковое целое (unsigned integer). distance(A, B) = |A xor B| . Меньшие значения считаются более близкими.  
Когда узлу необходимо найти пиров для торрента, он использует «расстояния», сравнивая хеш торрента (infohash) с идентификаторами узлов («node ID») из своей таблицы маршрутизации. Затем он соединяется с известными ему узлами, идентификаторы которых более близки к хешу торрента, и запрашивает у них контактную информацию о пирах, участвующих в данный момент в файлообмене этого торрента. Если узлу, с которым установлена связь, известно о пирах для этого торрента, то в его ответе будет возвращена контактная информация о них. В противном случае, в своем ответе, узел должен вернуть контактную информацию о наиболее «близких» к хешу торрента узлах, из своей таблицы маршрутизации. Далее наш узел циклично опрашивает все более «близкие» к хешу торрента узлы, до тех пор пока не будут найдены все самые «близкие» узлы. После того, как ресурсы поиска исчерпаны, клиент рассылает свою контактную информацию всем ответившим пирам, идентификаторы которых наиболее «близки» к хешу торрента.  
Когда узел запрашивает информацию о пирах для торрента, в ответ ему также приходит секретное значение «token». Если наш узел впоследствии решит проинформировать узел, к которому обращался ранее, о том что «мы» участвуем в файлообмене по запрашиваемому торренту, наш узел должен будет вернуть обратно этот «token». В свою очередь узел который предоставил «token», проверяет его используя IP нашего узла. Это позволяет предотвратит регистрацию торрентов «на имя» других узлов. Время, в течении которого возвращенный «token» всё ещё будет считаться «валидным», настоящим руководством точно не регламентируется. Token'ы должны приниматься обратно лишь в течении некоторого приемлемого количества времени. Реализация в BitTorrent использует хеш SHA1 от IP адреса и некоторого секретного значения, которое изменяется каждые 5 минут; принимаются token’ы, которым не более 10 минут.  
Таблица Маршрутизации  
Каждый узел поддерживает у себя таблицу маршрутизации, состоящую из других известных «актуальных» узлов. Узлы этой таблицы используются при первых запросах в DHT. Также, узлы из таблицы возвращаются в ответ на поисковые запросы от других узлов.  
Не все узлы, которые есть в таблице маршрутизации, одинаковы. Некоторые из них «актуальные», другие же нет. Многие узлы в DHT способны посылать и принимать запросы, но не все способны отвечать на запросы других узлов. Очень важно, чтобы таблицы маршрутизации всех узлов содержали только известные «актуальные» узлы. «Актуальным» узлом считается тот, который ответил хотя бы на один из наших запросов за последние 15 минут. Так же к «актуальным» относятся узлы, которые хотя бы раз отвечали на наши запросы, и при этом посылали нам хотя бы один запрос в течении последних 15-ти минут. После 15 минут отсутствия активности, узел переходит в разряд «сомнительных». Узел становится «не актуальным» в случае если он не отвечает на несколько запросов подряд. Узлы со статусом «актуальный» имеют более высокий приоритет.  
Таблица маршрутизации содержит ID (идентификаторы) узлов, которые принадлежат диапазону значений от 0 до 2^160 (2 в степени 160). Таблица маршрутизации поделена на «блоки», каждый из которых покрывает часть из исходного диапазона значений. Для каждого блока определены min и max, определяющие минимальное и максимальное значение ID, которое может в нем храниться. Пустая таблица содержит всего один блок, включающий все возможные значения ID, где min=0 и max =2^160. Когда узел с ID равным N вставляется в таблицу маршрутизации, он помещается в блок для которого выполняется условие min <= N < max. Так как пустая таблица содержит только один блок, то в нее может быть помещен и любой ID. Каждый блок может содержать максимум К узлов, в текущей реализации К=8. Блок с К узлами считается «заполненным». Если блок «заполнен» и содержит только «актуальные» узлы, то ни один узел больше не может быть в него добавлен. Исключение составляет наш собственный ID, если он попадает в диапазон значений этого блока. В случае необходимости текущий блок заменяется двумя новыми, каждый из которых делит диапазон значений исходного блока пополам. Соответственно ID из старого блока распределяются между этими двумя новыми блоками. Например для таблицы с одним блоком, «заполненный» блок будет разделен на два новых с диапазонами значений: от 0 до 2^159 и от 2^159 до 2^160.  
Когда блок заполнен «актуальными» узлами, то новый узел просто отбрасывается. Если какие-либо узлы в блоке становятся «не актуальны», то один из них заменяется на новый узел. Если в блоке есть «сомнительные» узлы, с которыми не было связи в течении последних 15-ти минут, то производиться попытка пропинговать наименее активный из них (связь с которым была наиболее давно). Если такой узел отвечает, то пингуется следующий «сомнительный» и наименее активный узел. И так до тех пор, пока один из них не ответить или все узлы в блоке не станут «актуальными». Если узел в блоке не отвечает на пинг, то допускается сделать ещё одну попытку перед исключением этого узла и заменой его на новый «актуальный» узел. Таким образом, таблица заполняется стабильно работающими узлами.  
Каждый блок должен содержать свойство «last changed», показывающее, насколько устарело содержимое. Это свойство должно обновляться: когда узел в блоке пингуется и отвечает, происходит добавление нового узла в блок, узел в блоке заменяется другим узлом. Блоки, которые не изменялись в течении последних 15-ти минут, должны быть обновлены. Это делается посредством выбора случайного ID в пределах блока и применением к нему функции поиска find\_nodes. Узлам, которые способны принимать запросы от других узлов, обычно не требуется часто обновлять блоки. Узлам же, которые не способны принимать запросы от других узлов (например находящимся за NAT), обычно требуется периодически обновлять все блоки, чтобы при использовании DHT, гарантировать наличие в их таблице «актуальных» узлов.  
После помещения первого узла в его же таблицу маршрутизации и затем во время запуска узел должен попытаться найти в DHT ближайшие к себе узлы. Это делается путем выдачи сообщений find\_node к все более и более близким узлам до тех пор, пока он будет не в состоянии найти какой-либо более близкий. Между запусками клиентской программы таблица маршрутизации должна сохраняться.  
Дополнение к протоколу BitTorrent  
Протокол BitTorrent был дополнен для того, чтобы производить обмен номерами UDP-портов узлов между теми пирами, которые предоставил трекер. Таким образом клиенты могут автоматически осуществить сидирование своих таблиц маршрутизации посредством периодической загрузки торрентов. Недавно установленные клиенты, которые пытаются загрузить бестрекерный торрент в первый раз, не будут иметь каких-либо узлов в своей таблице маршрутизации и будут нуждаться в контактах, включённых в торрент-файл.  
В BitTorrent используется «рукопожатие» после которого идут 8 зарезервированных байт для расширений. Пиры, поддерживающие DHT, выставляют последний бит этих 8-ми байт. Пир, получающий при установлении связи подтверждение, что удалённый пир поддерживает DHT, должен отправить сообщение PORT. Оно начинается с байта 0x09 и последующих двух байт, содержащих UDP-порт DHT узла, в сетевом порядке байтов (big-endian). Пиры, которые получают это сообщение, должны попытаться осуществить пингование узла на полученный порт и IP-адрес удалённого пира. Если получен ответ на пинг, то узел должен попытаться вставить новую контактную информацию в свою таблицу маршрутизации по обычным правилам.  
Расширения торрент-файлов  
В словаре торрентов, не использующих трекер, нет ключа «announce». Вместо этого, у таких торрентов имеется ключ «nodes» (узлы). Этот ключ должен быть установлен на K ближайших узлов из таблицы маршрутизации клиента, создающего торрент. Также ключ может быть установлен на известный действующий узел. Например, на узел управляемый человеком, создавшим торрент. Пожалуйста, не добавляйте автоматически "router.bittorrent.com" в торрент-файлы или таблицы маршрутизации клиентов.  
nodes = [["<адрес хоста>", <номер порта>], ["<адрес порта>", <номер порта>], ...]  
nodes = [["127.0.0.1", 6881], ["узел.вашего.роутера", 4804]]  
Протокол KRPC  
Протокол KRPC представляет собой простой RPC-механизм, который состоит из закодированных словарей, пересылаемых через UDP. В запросе и в ответе на запрос производится отсылка одного пакета. Повторных попыток нет. Имеется три типа сообщений: query, response и error. Для протокола DHT существуют четыре типа запросов: ping, find\_node, get\_peers и announce\_peer.  
KRPC-сообщение – это один словарь с двумя ключами, общими для каждого сообщения, и дополнительными ключами, зависящими от типа сообщения. Каждое сообщение содержит ключ «t» — строка, представляющая собой ID операции (транзакции). Данный ID операции генерируется запрашивающим узлом и возвращается в виде эхо-ответа. Таким образом, ответы могут быть сопоставлены нескольким разным запросам к одному и тому же узлу. В каждом KRPC сообщении также содержится ключ «y» — строка, описывающая тип сообщения. Ключа «y» может принимать следующие значения: «q» – для запроса, «r» – для ответа, «e» – для ошибки.  
Кодирование связи  
Контактная информация пиров кодируется в виде 6-байтной строки. Она также известна как «Информация в компактном виде о IP-адресе/порте» и представляет собой IP-адрес в виде 4 байтов (big-endian) вместе с 2-мя байтами порта (big-endian), расположенными в конце.  
Контактная информация узлов кодируется в виде 26-байтной строки. Она также известна как «Информация в компактном виде об узле» и представляет собой ID узла в виде 20 байтов (big-endian) вместе с «информацией в компактном виде о IP-адресе/порте», расположенной в конце.  
Запросы  
Словари KRPC-сообщений со значением «q» ключа «y». Содержат два дополнительных ключа: «q» и «a». Ключ «q» — строка, содержащая название метода этого запроса. Ключ «a» — словарь, содержащий именованные аргументы к запросу.  
Ответы  
Словари KRPC-сообщений со значением «r» ключа «y». Содержат один дополнительный ключ «r». Ключ «r» — словарь, содержащий именованные возвращаемые значения. Сообщения-ответы отсылаются после успешного завершения запроса.  
Ошибки  
Словари KRPC-сообщений со значением «e» ключа «y». Содержат один дополнительный ключ «e». Ключ «e» — список. Первым элементом является целое число, представляющее собой код ошибки. Вторым элементом является строка, содержащая сообщение ошибки. Ошибки отсылаются если запрос не может быть выполнен. Следующая таблица описывает возможные коды ошибок:  
Код Описание  
201 Общая ошибка  
202 Ошибка сервера  
203 Ошибка протокола, к примеру, неправильно сформированный пакет, неверные аргументы или недопустимый token  
204 Метод неизвестен  
Пакеты с примерами ошибок  
общая ошибка = {"t":"aa", "y":"e", "e":[201, "A Generic Error Ocurred"]}  
в кодированном виде = d1:eli201e23:A Generic Error Ocurrede1:t2:aa1:y1:ee  
Запросы DHT  
Во всех запросах имеется ключ «id» и значение, содержащее ID запрашивающего узла. Во всех ответах имеется ключ «id» и значение, содержащее ID отвечающего узла.  
ping  
Наиболее элементарным запросом является пинг. «q» = «ping». В пинг-запросе имеется один аргумент: «id» — 20-байтная строка (big-endian), содержащая ID узла отправителя. В правильном ответе на пинг содержится лишь один ключ «id», содержащий ID отвечающего узла.  
аргументы: {"id" : "<id запрашивающих узлов>"}  
ответ: {"id" : "<id ответивших узлов>"}  
Примеры пакетов  
Запрос ping = {"t":"aa", "y":"q", "q":"ping", "a":{"id":"abcdefghij0123456789"}}  
в кодированном виде = d1:ad2:id20:abcdefghij0123456789e1:q4:ping1:t2:aa1:y1:qe  
Ответ = {"t":"aa", "y":"r", "r": {"id":"mnopqrstuvwxyz123456"}}  
в кодированном виде = d1:rd2:id20:mnopqrstuvwxyz123456e1:t2:aa1:y1:re  
find\_node  
«Find node» используется для поиска контактной информации об узле, давшего свой ID. «q» = «find\_node». В запросе find\_node имеется два аргумента: «id» — содержит ID запрашивающего узла, и «target» — содержит ID узла, который собственно и ищет запрашивающий узел. Когда узел получает запрос find\_node, он должен ответить ключом «nodes» и строкой, содержащей «информацию в компактном виде об узле» для искомого узла или для K (как правило равно 8-ми) ближайших к искомому «актуальных» узлов из своей таблицы маршрутизации.  
аргументы: {"id" : "<ID запрашивающего узла>", "target" : "<ID искомого узла>"}  
ответ: {"id" : "<ID отвечающего узла>", "nodes" : "<compact node info>"}  
Примеры пакетов:  
запрос find\_node = {"t" : "aa", "y" : "q", "q" : "find\_node", "a" : {"id" : "abcdefghij0123456789", "target" : "mnopqrstuvwxyz123456"}}  
в кодированном виде = d1:ad2:id20:abcdefghij01234567896:target20:mnopqrstuvwxyz123456e1:q9:find\_node1:t2:aa1:y1:qe  
ответ = {"t" : "aa", "y" : "r", "r" : {"id":"0123456789abcdefghij", "nodes" : "def456..."}}  
в кодированном виде = d1:rd2:id20:0123456789abcdefghij5:nodes9:def456...e1:t2:aa1:y1:re  
get\_peers  
«Get peers» связано с полем «infohash» торрента. «q» = «get\_peers». В запросе get\_peers имеется два аргумента: «id» — содержит ID запрашивающего узла, и «info\_hash» — содержит значение «infohash» торрента. Если у опрошенного узла имеются пиры для данного «infohash», то они возвращаются в ключе «values» в виде списка строк, содержащих «информацию в компактном формате об узлах». Если же у опрошенного узла нет пиров для данного «infohash», то он возвращается ключ «nodes», содержащий K ближайших, к предоставленному в запросе «infohash», узлов из своей таблицы маршрутизации. В обоих случаях ключ «token» также включается в возвращаемое значение. Значение «token» является необходимым аргументом для будущего запроса «announce\_peer».  
аргументы: {"id" : "<ID запрашивающего узла>", "info\_hash": "<20-ти байтный infohash искомого торрента>"}  
ответ: {"id" : "<ID отвечающего узла>", "token" : "<Секретный token>", "values" : ["<Информация о пире №1>", "<Информация о пире №2>"]}  
или: {"id" : "<ID отвечающего узла>", "token" :"<Секретный token>", "nodes" : "<информация в компактном формате об узле>"}  
Примеры пакетов:  
запрос get\_peers = {"t" : "aa", "y" : "q", "q" : "get\_peers", "a" : {"id" : "abcdefghij0123456789", "info\_hash" : "mnopqrstuvwxyz123456"}}  
в кодированном виде = d1:ad2:id20:abcdefghij01234567899:info\_hash20:mnopqrstuvwxyz123456e1:q9:get\_peers1:t2:aa1:y1:qe  
ответ, содержащий пиров = {"t" : "aa", "y" : "r", "r" : {"id" : "abcdefghij0123456789", "token" : "aoeusnth", "values" : ["axje.u", "idhtnm"]}}  
в кодированном виде = d1:rd2:id20:abcdefghij01234567895:token8:aoeusnth6:valuesl15:axje.uidhtnmbrlee1:ti0e1:y1:re  
ответ, содержащий ближайшие узлы = {'t' : 0, 'y' : 'r', 'r' : {'id' : 'abcdefghij0123456789', 'token' : 'aoeusnth', 'nodes' : 'def456...'}}  
в кодированном виде = d1:rd2:id20:abcdefghij01234567895:nodes9:def456...5:token8:aoeusnthe1:ti0e1:y1:re  
announce\_peer  
Этим сообщением узел анонсирует что он участвует в файлообмене по указанному торренту. Сообщение announce\_peer имеет 4 аргумента:«id» — содержит ID запрашивающего узла, «info\_hash» — содержит «infohash» торрента, «port» — содержит номер порта в виде целого числа и «token», полученный в ответе на предыдущий запрос get\_peers. Опрошенный узел обязан проверить, что «token» был ранее отправлен на тот же самый IP-адрес, что и принадлежащий запрашивающему узлу. Затем опрошенный узел должен сохранить предоставленные IP-адрес и номер порта в своём хранилище контактной информации пиров, связав их с соответствующим «infohash».  
аргументы: {"id" : "<ID запрашивающего узла>", "info\_hash" : "<20-ти байтный infohash>", "port" : <номер порта>, "token" : "<Секретный token>"}  
ответ: {"id" : "<ID отвечающего узла>"}  
Примеры пакетов:  
запрос announce\_peers = {"t" : "aa", "y" : "q", "q" : "announce\_peer", "a" : {"id" : "abcdefghij0123456789", "info\_hash" : "mnopqrstuvwxyz123456", "port" : 6881, "token" : "aoeusnth"}}  
в кодированном виде = d1:ad2:id20:abcdefghij01234567899:info\_hash20:  
mnopqrstuvwxyz1234564:porti6881e5:token8:aoeusnthe1:q13:announce\_peer1:t2:aa1:y1:qe  
ответ = {"t" : "aa", "y" : "r", "r" : {"id" : "mnopqrstuvwxyz123456"}}  
в кодированном виде = d1:rd2:id20:mnopqrstuvwxyz123456e1:t2:aa1:y1:re  
  
Сноски  
1. "Kademlia: A Peer-to-peer Information System Based on the XOR Metric",  
Petar Maymounkov и David Mazieres, IPTPS 2002. <http://www.cs.rice.edu/Conferences/IPTPS02/109.pdf>  
2. Чтобы получить уникальный ID — пользуйтесь SHA1 и обеспечивайте приемлемый уровень энтропии.

DHT  
DHT (англ. Distributed hash table распределенная хэш-таблица)— это протокол, позволяющий битторрент-клиентам находить друг друга без использования трекера.  
  
Клиенты с поддержкой DHT образуют общую DHT-сеть и помогают друг другу найти участников одних и тех же раздач.  
Поддержка DHT есть в клиентах Mainline, µTorrent, KTorrent, Deluge, BitSpirit, Transmission и BitComet. В Azureus есть собственная реализация DHT, то есть клиенты Azureus образуют свою собственную отдельную DHT-сеть, а с помощью плагина mainlineDHT присоединяются к стандартной сети DHT, участвуя одновременно в обеих сетях.  
PEX  
PEX (Peer exchange) — это расширение протокола для обмена списками участников.  
PEX реализуется как дополнительные сообщения между клиентами, уже соединёнными между собой для обмена сегментами файла по обычному протоколу BitTorrent.  
В отличие от трекера и DHT, PEX может быть только вспомогательным средством получения пиров, так как он не может помочь новым пирам подключиться к раздаче .  
PEX есть в клиентах Transmission, Azureus, BitComet, KTorrent, µTorrent и BitTornado, причем в каждом клиенте он реализован по-своему, поэтому PEX между собой могут пользоваться только одинаковые клиенты. Начиная с 3 версии Azureus (Vuze) может обмениваться PEX с uTorrent и BitTorrent.  
Функции  
И DHT и PEX фактически выполняют основную функцию трекера — помогают участникам файлообмена узнать друг о друге. Они могут:  
Помочь участникам быстрее друг друга найти  
Например, на раздаче есть пир X с недоступным портом. К раздаче подключается пир Z, который сам начать соединение с X не может, и вынужден ждать, пока Х о нём узнает сам. Х только что обращался к трекеру, и в следующий раз собирается это сделать через час. Но вот пир Y в очередной раз обращается к трекеру и узнаёт про нового пира Z. При этом Y сам давно уже соединен и занимается файлообменом с X, поэтому он через PEX сообщает X адрес этого нового пира. Теперь X может начать соединение к Z.  
Снизить нагрузку на трекер  
Некоторые клиенты, например Azureus, получая адреса пиров через DHT или PEX, реже обращаются за списком пиров на трекер.  
Поддержать участников вместе в периоды недоступности трекера  
Известно, что если трекер является единственным источником информации о пирах, то при его неработоспособности раздача постепенно останавливается. Клиенты помнят уже известные списки адресов других пиров, но постепенно эти списки устаревают — некоторые пользователи отключаются от раздачи, у некоторых меняется IP адрес, а новые пользователи не могут подключиться к раздаче вообще.  
PEX позволяет несколько замедлить процесс распадения роя участников, а DHT позволяет полностью заменить трекер, то есть даже подключаться к раздаче новым участникам.  
DHT позволяет раздавать вообще без трекера  
Такая раздача называется trackerless. Торрент для неё создается без адреса трекера, и клиенты друг друга находят через DHT сеть. При участии в trackerless раздачах БТ клиенты приобретают определённое сходствой с eMule, использующим сеть KAD.  
Закрытый ключ (англ. Private key)  
На открытых трекерах (англ. public), где каждый желающий может скачать торрент и участвовать в раздаче, DHT и PEX служат на благо всех участников.  
Закрытым (англ. private) трекерам в первую очередь важно, чтобы в раздачах могли участвовать только зарегистрированные пользователи, и чтобы они соблюдали определённые правила. При первом обращении клиента частный трекер имеет возможность не допустить его к раздаче, просто не сообщая ему адреса других клиентов-участников. Поэтому для закрытого трекера важно, чтобы клиенты не получали эти адреса через DHT/PEX.  
DHT и PEX появились в клиентах Azureus и BitComet примерно летом 2005 года. Администраторы многих частных трекеров были недовольны такой новой функциональностью, и поэтому стали запрещать на трекере эти новые версии клиентов.  
Тогда разработчики клиентов предложили новый ключ внутри торрент файла: private. Если его значение равно единице, то клиент обязан для этого торрента автоматически отключать DHT/PEX независимо от желания пользователя. Такой торрент называют защищенным (англ. Secure Torrent).  
Практически все современные частные трекеры сами принудительно вставляют private:1 во все торренты, выкладываемые на трекере, а также запрещают несколько устаревших версий клиентов, поддерживающих DHT или PEX, но ещё не знающих про private key. Пользователи трекера просто не могут на раздачах использовать DHT/PEX, и проблемы нет.  
Отметим, что присутствие private key изменяет infohash торрента, поэтому выреза́ть его из торрент файла бесполезно — другие клиенты изменённый торрент всё равно не признают.  
Включение в клиенте  
Рассмотрим, как включается DHT в популярных клиентах Bitcomet и µTorrent.  
DHT можно включить глобально в настройках клиента. При этом клиент подключается к DHT сети, статус подключения показан в самом низу окна клиента.  
Затем в свойствах каждой отдельной задачи вы можете указать, использовать ли для этой задачи DHT сеть для нахождения новых пиров.  
Если в свойствах задачи включить DHT нельзя, то это значит, что либо у вас выключено DHT глобально в клиенте, либо этот торрент — частный, то есть с private key внутри, и клиент не позволит использовать ни DHT ни PEX.  
µTorrent на вкладке «Пиры» показывает, получен ли каждый пир с трекера, через PEX или через DHT, с помощью флагов X (для PEX) и H (для DHT) в колонке «Флаги». Важно понимать, что обычно это разделение условное, один и тот же пир вполне может быть получен сначала одним, а потом другим способом.  
Пользоваться ли  
Все ваши торренты — с частных трекеров  
Если при этом в клиенте разрешить DHT, то получится, что клиент подключается к DHT сети, тратит на это трафик, помогает другим клиентам найти нужных им пиров, но ни на одной раздаче DHT для себя не использует. Если вы не хотите тратить лишний трафик, то видимо лучше DHT в клиенте отключить.  
Вы качаете раздачу с публичного трекера  
Если трекер возвращает вам много пиров и их достаточно для достижения хорошей скорости скачивания, то DHT/PEX вам вероятно не нужно. Если нет, то стоит попробовать их включить (и в клиенте и в свойствах раздачи), это может помочь найти больше источников.  
Вы качаете раздачу с частного трекера без принудительного private key  
Из крупных русскоязычных трекеров на конец 2006 года это торрентс.ру. Возможность использования на раздачах DHT/PEX на этих трекерах отдана на откуп раздающему (создателю торрента).  
Вообще говоря, такая ситуация не может быть признана нормальной, особенно на трекерах с системой passkey. Дело в том, что в клиентах BitComet, BitSpirit и Azureus через DHT пользователи могут узнать passkey других пользователей, и нечестные пользователи могут использовать чужие passkey для скачивания под чужой учетной записью. Поэтому по крайней мере в этих клиентах на таких трекерах рекомендуется DHT выключить.  
DHT и статистика  
Этот раздел касается только частных трекеров, на которых private key в торренты принудительно не вставляется, и на некоторых раздачах (в зависимости от того, вставил ли раздающий сам в торрент private key) можно использовать DHT и PEX.  
Часто встречается мнение, что включенный в клиенте DHT влияет на учет статистики клиента трекером, например «раздавал через DHT, значит статистика шла мимо трекера». Это неверно.  
Во-первых, DHT/PEX используется только для получения адресов пиров. Ни файлообмена, ни какого-либо учета статистики по ним не ведётся. Клиент рапортует статистику скачанного и отданного только на трекер.  
То есть «раздавал через DHT» фактически означает «о некоторых (или о всех) пирах получил информацию по DHT, и вероятно некоторые пиры тоже нашли меня через DHT»  
Во-вторых, хотя клиенты обычно и знают, откуда ими получены адреса пиров, ни один клиент не разделяет трафик на «полученный/отданный DHT пирам» и «полученный/отданный пирам, полученным от трекера». Даже при желании это было бы клиенту сделать затруднительно — некоторые пиры могут быть получены и от трекера и через DHT или PEX, и часто клиент не знает, как его адрес получил пир, сам начинающий к нему соединение.  
Клиент рапортует трекеру суммарные данные об объёмах им скачанного и отданного всем пирам, с которыми он общался, независимо от того, узнал клиент об отдельных пирах через трекер, DHT или PEX, или тот пир вообще начал соединение сам. То есть даже если из-за DHT/PEX на раздаче появятся «левые» пользователи (не обращающиеся к трекеру), клиент все равно сообщит на трекер все, что у них скачал и отдал.  
Правильный учет статистики зависит только от состояния трекера: работает трекер — статистика учитывается, не работает — не учитывается. Только в случае длительно неработающего трекера DHT/PEX может играть косвенную роль, не давая постепенно затухнуть файлообмену на такой «раздаче без учета статистики».  
Механизм работы DHT  
Реализация распределеной сети в БТ клиентах основана на варианте DHT, называемом Kademlia. А вообще говоря, DHT (Distributed hash table) означает децентрализованную распределенную систему для объединения большого количества постоянно исчезающих и появляющихся узлов и эффективной передачи сообщений между ними. На основе DHT структур строят разные более сложные системы, такие как P2P файлообмен, кооперативное веб кеширование, DNS сервисы и т. п.  
DHT использует UDP протокол. БТ клиенты слушают тот же UDP номер порта, который они используют для входящих TCP соединений. Если вы активно используете DHT, то открытие этого UDP порта для доступа снаружи желательнo, но не обязательно — DHT будет работать и так.  
Каждый подключенный БТ клиент является в DHT сети отдельным узлом. У него есть свой уникальный ID (идентификатор), случайно выбираемый из того же 160-битного пространства, что и infohash’ы торрентов.  
Каждый узел хранит таблицу маршрутизации, содержащую контактную информацию о многих «ближайших» к нему узлах, и о нескольких более далеких. «Близость» двух узлов вычисляется из «сходства» их ID, и не имеет никакого отношения к их географической близости.  
Когда узел хочет найти пиров для какой-то раздачи, он сравнивает infohash этой раздачи с ID известных ему узлов, и затем посылает запрос тому узлу, чей ID наиболее похож на этот infohash. Тот узел возвращает ему адрес узла, чей ID ещё ближе к infohash торрента.  
Тогда наш узел посылает запрос тому новому узлу, и получает от него адрес следующего узла, чей ID ещё более похож на infohash торрента.  
Таким образом, запросы от клиентов, участвующих в раздаче торрента с определённым infohash, постепенно стекаются к узлам, чьи ID наиболее похожи на этот infohash. Эти узлы помнят предыдущие запросы, и всем следующим запрашивающим узлам вернут адреса предыдущих пиров с той же раздачи.