Internet Engineering Task Force (IETF) M. Stenberg
Request for Comments: 7787 S. Barth
Category: Standards Track Independent
ISSN: 2070-1721 April 2016

Distributed Node Consensus Protocol

Распределенный протокол согласования узлов

Аннотация

Этот документ описывает распределенный протокол согласования узлов Distributed Node Consensus Protocol или DNCP), обеспечивающий синхронизацию состояний с помощью алгоритма Trickle и хэш-деревьев. DNCP является абстрактным протоколом и должен применяться с конкретным профилем для реализации.

Статус документа

Документ содержит проект стандарта Internet (Standards Track).

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошёл открытое обсуждение и был одобрен для публикации IESG[2]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 5741.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке http://www.rfc-editor.org/info/rfc7787.

Оглавление

[1]Internet Engineering Task Force - комиссия по решению инженерных задач Internet.
[2]Internet Engineering Steering Group - комиссия по инженерным разработкам Internet.

1. Введение

Протокол DNCP предназначен для обеспечения каждому участвующему узлу способа опубликовать небольшой набор триплетов TLV[1] (не более 64 Кбайт) и разделяемого общего представления о данных, опубликованных каждым узлом DNCP с двухсторонней доступностью.

Для синхронизации состояний применяется хэш-дерево. Дерево формируется расчётом хэша сначала для набора опубликованных каждым узлом данных (данные узла), затем хеша для хэш-значений узлов. Полученное в результате значение называется хэшем состояния сети и передаётся с использованием алгоритма Trickle [RFC6206], чтобы все узлы имели общее представления текущего состояния опубликованных данных. Использование Trickle лишь с короткими хэшами состояния сети, передаваемыми редко (в установившемся состоянии 1 раз за максимальный интервал Trickle на канал или unicast-соединение), делает DNCP очень экономным при нечастых обновлениях.

Для поддержки живучести топологии и данных в ней применяется комбинация состояния сети (Trickle), keep-alive и «другие» меры. Основная идея состоит в том, что при гарантии каждым узлом присутствия его партнёров транзитивность обеспечивает актуальность состояния всей сети.

1.1. Применимость

Протокол DNCP полезен для таких случаев, как начальная загрузка, обнаружение и согласование встраиваемых сетевых устройств (например, маршрутизаторов). Кроме того, он может служить основой для работы распределенных алгоритмов, подобных [RFC7596] или сценариев, описанных в Приложении C. DNCP является абстракцией, которую можно с помощью профилей настроить для разных приложений. Профили включают выбор:

- индивидуального транспорта - ориентированный на дейтаграммы или потоки протокол (например, TCP, UDP или SCTP) для базовых операций;
- необязательной защиты транспорта - условия и места применения защиты TLS (Transport Layer Security) или DTLS (Datagram Transport Layer Security), если выбранный транспорт её поддерживает;
- необязательного группового транспорта - протокола с поддержкой групповой адресации, такого как UDP, позволяющего обнаруживать автономные узлы и более эффективно использовать каналы с общим доступом;
- области действия адресов - использование лишь поэтапной (hop by hop) трансляции с адресацией link-local (например, для ЛВС), адресов с более широкой областью действия (например, WAN или Internet) на основе имеющейся инфраструктуры маршрутизации или комбинации обоих вариантов (например, для обмена состояниями между несколькими ЛВС через WAN или Internet);
- полезной нагрузки (payload) - дополнительные специфические данные (например, стандартизованные IANA, фирменные или приватные);
- расширений - возможные расширения протокола, заданные этим документом или разработанные специально для определённого приложения.

Однако в некоторых случаях определённый в этом документе протокол будет наименее подходящим. Ниже приведён обзорный список, а в последующих параграфах даны более подробные рекомендации.

- Объем данных - узлы ограничены публикацией не более 64 Кбайт.
- Высокая плотность сетей без групповой адресации. Узлы включают информацию обо всех прямых соседях в публикуемые данные.

[1] Type-Length-Value - тип-размер-значение.

- Преимущественно минимальный объем данных. Данные узлов всегда передаются без преобразования, что ведёт к сравнительно высоким издержкам при передаче изменений, влияющих лишь на малую часть данных.
- Частое изменение данных. DNCP с использованием алгоритма Trickle оптимизирован для установившихся состояний и менее эффективен в иных случаях.
- Большое число очень ограниченных устройств. DNCP требует от каждого узла хранить все данные, опубликованные всеми другими узлами.

Топология соединения устройств не ограничивается и распознаётся автоматически. При использовании лишь локальных каналов (link-local) все каналы с узлами DNCP должны быть транзитивно соединены с маршрутизаторами, поддерживающими протокол на нескольких конечных точках, для формирования сети соединений. Однако не требуется поддерживать протокол на каждом устройстве в физической сети. В частности, при использовании глобальных адресов от партнёров DNCP не требуется подключение к одному или даже соседним физическим каналам. Обычно в локальных сетях применяются функции автоматического обнаружения, а при включённой защите можно применять DNCP и в незащищённых сетях общего доступа. Размер сети ограничивается лишь возможностями устройств, т. е. каждый узел DNCP должен быть способен хранить все данные, опубликованные другими узлами. Данные, связанные с каждым идентификатором отдельного узла, этот документ ограничивает размером 64 Кбайта, однако могут быть определены расширения протокола для смягчения этого и других ограничений протокола.

DNCP больше всего подходит для редко меняющихся данных, чтобы по максимуму воспользоваться преимуществами использования Trickle. По мере роста числа узлов или частоты изменения данных алгоритм Trickle применяется все реже и применение DNCP уже не даёт преимуществ. В этих случаях применение Trickle лишь усложняет спецификацию. Пригодность DNCP для конкретной задачи можно оценить, рассматривая предполагаемый средний интервал возникновения изменений в сети A_NC_I, рассчитываемый путём деления среднего интервала между созданием узлом новых TLV на число участвующих узлов. При использовании сообщений keep-alive значение A_NC_I будет меньшим из рассчитанного A_NC_I и интервала keep-alive. Если A_NC_I меньше зависящего от применения минимального интервала Trickle, DNCP скорей всего не подойдёт, поскольку Trickle большую часть времени не будет применяться.

Если требуются постоянные быстрые смены состояния, предпочтительным вариантом будет использование дополнительного канала «точка-точка», для которого адрес или локатор публикуется с помощью DNCP. Но даже в этом случае, если рассчитанное выше значение A_NC_I не превышает (разумно выбранный интервал) Trickle для конкретного приложения, протокол DNCP может не подойти для приложения.

Другая проблема связана с размером публикуемого узлом набора TLV по сравнению с размером изменений в наборе TLV. Если публикуемый узлом набор TLV очень велик и в нем часто происходят мелкие изменения, описанный в этом документе протокол DNCP может оказаться неподходящим, поскольку ему не хватает схемы синхронизации изменений для сохранения простоты реализации.

DNCP можно использовать в сетях, где доступен лишь индивидуальный (unicast) транспорт. Хотя DNCP при работе с групповой адресацией расходует меньшую пропускную способность, даже при работе в режиме unicast использование Trickle (в идеале с $k < 2$) обеспечивает значительно меньше передач по сравнению с протоколом без Trickle.

2. Терминология

DNCP profile - профиль DNCP
Значения набора параметров, представленных в разделе 9. В этом документе для них используется префикс DNCP_. Профиль также задаёт набор используемых необязательных расширений DNCP. Пример простого профиля DNCP приведён в Приложении C.

DNCP-based protocol - протокол, основанный на DNCP
Протокол, предоставляющий профиль DNCP в соответствии с разделом 9 с возможными назначениями TLV из реестра TLV на уровне профиля DNCP,а также правила обработки.

DNCP node - узел DNCP
Отдельный узел, на котором работает основанный на DNCP протокол.

Link - канал
Среда канального уровня, через которую могут взаимодействовать соединённые напрямую узлы.

DNCP network - сеть DNCP
Множество узлов DNCP, на которых работает основанный на DNCP протокол(ы) с одним профилем DNCP. Множество включает узлы, обнаруживающие друг друга с использованием транспорта, заданного профилем DNCP, с помощью групповой рассылки на локальном канале и/или индивидуальных коммуникаций.

Node identifier - идентификатор узла
Необрабатываемый (opaque) идентификатор фиксированного размера DNCP_NODE_IDENTIFIER_LENGTH байтов, однозначно указывающих узел DNCP в сети DNCP.

Interface - интерфейс
Подключение узла к конкретному каналу.

Address - адрес
Идентификатор, указывающий источник или получателя потока сообщений DNCP, например, адрес IPv6 и порт UDP для транспорта IPv6 UDP.

Endpoint - конечная точка
Локально настроенная точка завершения потока сообщений DNCP (возможная или имеющаяся). Конечная точка является источником и получателем для отдельного индивидуального (unicast) потока сообщений отдельным узлам и (необязательно) для групповых сообщений доступным узлам (например, для обнаружения узлов). Конечная точка обычно находится в одном из транспортных режимов, заданных в параграфе 4.2.

Endpoint identifier - идентификатор конечной точки
32-битовое неанализируемое (opaque) значение, уникальное в локальном масштабе, которое указывает отдельную конечную точку на конкретном узле DNCP. Значение 0 зарезервировано для DNCP и протоколов на основе DNCP и не применяется для реальных конечных точек. Это определение согласуется с определением индекса интерфейса в [RFC3493], как отличного от 0 небольшого положительного числа, которому следует помещаться в 32 бита.

Peer - партнёр
Узел DNCP, с которым данный узел DNCP взаимодействует, используя хотя бы 1 пару конечных точек (локальная и удалённая).

Node data - данные узла
Набор TLV, принадлежащих узлу в сети DNCP и опубликованных им. Другие узлы передают данные без изменения, даже если не могут их полностью интерпретировать.

Origination time - время создания
Оценённое время, когда был опубликован набор данных узла с текущим порядковым номером.

Node state - состояние узла
Набор атрибутов метаданных для данных узла, включающий порядковый номер (версия), хэш-значение для сравнения с сохранёнными данными узла и метку, указывающее время, прошедшее с момента последней публикации (т. е. после момента создания). Хэш-функцию и размер хэша определяет профиль DNCP.

Network state hash - хэш состояния сети
Хэш-значение, которое представляет текущее состояние сети. Хэш-функцию и размер хэша определяет профиль DNCP. Значение меняется всякий раз при добавлении или удалении узла, а также при обновлении узлом опубликованных данных. Расчёт описан в параграфе 4.1.

Trust verdict - вердикт доверия
Заявление о надёжности сертификата от узла, участвующего в согласовании доверия на основе сертификатов.

Effective trust verdict - эффективный (действующий) вердикт доверия
Вердикт доверия с наивысшим приоритетом в наборе вердиктов, анонсированных для сертификата в сети DNCP.

Topology graph - граф топологии
Ненаправленный граф узлов DNCP, созданный на основе лишь двухсторонних партнерских связей между узлами.

Bidirectionally reachable - двухсторонняя достижимость
Партнёр является локально достижимым в одном направлении, если локальный узел получил от него корректный групповое сообщение или любое индивидуальное сообщение DNCP (параграф 4.5). Если этот партнёр также считает локальный узел односторонне достижимым, это говорит о двухсторонней достижимости. Поскольку процесс основан на публикации партнерских отношений и оценке результирующего графа топологии (параграф 4.6), эта информация доступна во всей сети DNCP.

Trickle instance - экземпляр Trickle
Отдельное состояние алгоритма [RFC6206], сохраняемое узлом (раздел 5) и относящееся к конечной точке или конкретному кортежу (партнёр-конечная точка) с переменными Trickle I, t, c (параграф 4.3).

2.1. Уровни требований

Ключевые слова **должно** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с RFC 2119 [RFC2119].

3. Обзор

DNCP работает обычно на основе индивидуального (unicast) обмена между узлами и может использовать групповую передачу (multicast) для основанного на алгоритме Trickle распространения общего состояния и определения топологии. При использовании лишь unicast-режима алгоритм Trickle применяется также между партнёрами.

Протокол DNCP основан на обмене TLV (раздел 7) и определяет часть их как обязательные для работы. TLV классифицируются как предназначенные для запроса информации (параграф 7.1), передачи данных (параграф 7.2) и публикуемые как данные (параграф 7.3). Протоколы на основе DNCP обычно задают дополнительные TLV для расширения возможностей.

DNCP определяет топологию узлов в сети DNCP и поддерживает жизненность опубликованных данных узлов, обеспечивая двухстороннюю доступность публикующих узлов. Новые потенциальные партнёры могут обнаруживаться автономно на каналах с групповой адресацией, их адреса могут быть настроены вручную или определены заданным в профиле DNCP способом. Профиль DNCP может задавать, например, общеизвестный anycast-адрес или предоставлять удалённый адрес для взаимодействия по иному протоколу, такому как DHCPv6 [RFC3315].

Хэш-дерево высотой 1 с корнем в себе поддерживается каждым узлом для представления состояния всех доступных в данный момент узлов (см. параграф 4.1), а алгоритм Trickle служит для запуска синхронизации (см. параграф 4.3). необходимость проверки узлов на предмет смены состояния определяется сравнением текущего корня соответствующих хэш-деревьев, т. е. рассчитанных индивидуально хэшей состояния сети.

Перед подключением к сети DNCP узел начинается с хэш-дерева, имеющего лишь один лист, если узел публикует TLV, и без листьев в ином случае. Затем узел анонсирует хэш состояния сети, рассчитанные из хэш-дерева с помощью алгоритма Trickle на всех настроенных конечных точках.

При обнаружении узлом обновления (например, получение от партнёра иного состояния сети) у инициатора события запрашивается список состояний всех узлов (т. е. вся информация, использованная им для расчёта своего хэш-дерева). Узел использует список для определения актуальности своих данных и при необходимости запрашивает изменившиеся данные. При обновлении локальной копии любого узла данных и его хэш-дерева (например, в результате смены состояния у него или другого узла, а также при добавлении или удалении партнёра), его экземпляры Trickle сбрасываются, что в конечном итоге вызывает распространение каждого обновления всем партнёрам.

4. Работа протокола

4.1. Хэш-дерево

Каждый узел DNCP поддерживает хэш-дерево произвольной ширины с высотой 1. Корень дерева представляет хэш общего состояния сети и служит для проверки согласованности и общности представления сети двумя или более узлами. Каждый лист представляет один узел DNCP с двухсторонней доступностью. При добавлении или удалении узла из графа топологии (параграф 4.6) лист добавляется или удаляется. В каждый момент листья дерева упорядочены по росту идентификаторов представляемых ими узлов.

4.1.1. Расчёт хэшей состояния сети и данных узлов

Хэш состояния сети и хэш-значения узлов данных рассчитываются с помощью хэш-функции, заданной профилем DNCP (раздел 9), и отсекаются до указанного в профиле числа битов.

Хэши отдельных узлов данных рассчитываются путём применения функции и отсечки для данных соответствующего узла, опубликованных в Node State TLV. Наборы данных узлов всегда упорядочиваются, как указано в параграфе 7.2.3.

Хэш состояния сети рассчитывается путём применения функции и отсечки для объединённого состояния сети. Это состояние формируется сначала конкатенацией порядкового номера узла (сетевой порядок байтов) с хэшем данных узла для формирования блока данных узла (datum). Затем выполняется конкатенация этих блоков по порядку роста номеров (т. е. в порядке размещения узлов в хэш-дереве).

4.1.2. Обновление хэшей состояния сети и данных узлов

Хэши состояния сети и узлов обновляются по запросу, а также при изменении любого локально сохранённого состояния узла. Это включает локальную одностороннюю достижимость в опубликованных Peer TLV (параграф 7.3.1) и (в комбинации с удалёнными данными) ведёт к осведомлённости об изменениях двухсторонней достижимости.

4.2. Транспортировка данных

DNCP предъявляет немного требований в базовому транспорту - нужен какой-либо способ передачи партнёру индивидуальных дейтаграмм или потока данных, а при использовании группового режима - способ передачи групповых дейтаграмм. Поскольку групповая адресация применяется лишь для идентификации потенциальных новых узлов DNCP и передачи статусных сообщений, которые просто указывают, что следует перейти в индивидуальный (unicast) режим, групповой транспорт не требует защиты. Если нужна защита индивидуального трафика и применение одного из встроенных механизмов защиты, потребуется одна из транспортных на основе TLS (например, TLS [RFC5246] для TCP или DTLS [RFC6347] для UDP). Это обеспечивает защиту целостности, а также проверку подлинности и полномочий с помощью схемы, заданной в разделе 8. Конкретное указание используемого транспорта и его параметров **должен** включать профиль DNCP.

TLV (раздел 7) транспорт передаёт без изменений и их **следует** передавать вместе, если какие-либо условия (например, MTU) не требуют передавать их в отдельных пакетах. DNCP не поддерживает фрагментации и сборки TLV, поэтому **должно** гарантироваться выполнение этих операций транспортным уровнем (при необходимости). Если этот документ указывает отправку одного или нескольких TLV, передающему узлу не требуется отслеживать отправленные пакеты после их передачи транспортному уровню, т. е. надёжность операций DNCP обеспечивается лишь явно заданными таймерами и конечными автоматами, такими как Trickle (параграф 4.3). TLV обычно обрабатываются индивидуально и без сохранения состояний (поэтому для них не требуется упорядоченной отправки) с одним исключением - для формирования отношений двухсторонней достижимости DNCP нужна идентификация участвующих конечных точек. Отношения двухсторонней достижимости требуются для проверки живучести опубликованных данных узла, как указано в параграфе 4.6, и узел DNCP **должен** передавать Node Endpoint TLV (параграф 7.2.1). Время отправки зависит от базового транспорта, но концептуально Node Endpoint TLV следует быть доступным при обработке Network State TLV, как указано ниже.

- При использовании потокового транспорта Node Endpoint TLV **должен** передаваться хотя бы один раз на каждое соединение и **не следует** передавать его неоднократно.
- При использовании дейтаграмм Node Endpoint TLV **должен** включаться в каждую дейтаграмму, содержащую Network State TLV (параграф 7.2.2) и **должен** размещаться перед любым таким TLV. Его **следует** включать также во все прочие дейтаграммы для ускорения обнаружения соседей.

С учётом различных вариантов транспорта, а также возможных конфигураций конечных точек, конечная точка DNCP может использоваться в указанных ниже транспортных режимах.

Unicast - индивидуальная адресация

- При использовании лишь гарантированного индивидуального транспорта алгоритм Trickle не применяется совсем. При каждом изменении рассчитанного локально хэша передаётся Network State TLV (параграф 7.2.2) каждому unicast-партнёру. **Могут** включаться также недавно изменённые Node State TLV (параграф 7.2.3).
- При использовании лишь индивидуального транспорта без гарантий состояние Trickle сохраняется на уровне партнёра и применяется для незамедлительной отправки Network State TLV, как указано в параграфе 4.3.

Multicast+Unicast - индивидуальная и групповая адресация

При доступности на конечной точке транспорта групповых дейтаграмм состояние Trickle поддерживается лишь для конечной точки в целом. Оно применяется для периодической отправки Network State TLV, как указано в параграфе 4.3. В дополнение профиль DNCP **может** определять keep-alive на уровне конечной точки.

MulticastListen+Unicast - прослушивание групповых пакетов и индивидуальная адресация

Отличается от использования индивидуального транспорта лишь прослушиванием групповых передач для обнаружения изменений старшего (наибольшего) идентификатора устройства. Этот режим применяется ли в случае поддержки профилем DNCP оптимизации каналов с групповой адресацией и высокой плотностью (параграф 6.2).

4.3. Управляемые Trickle обновления состояния

Алгоритм Trickle [RFC6206] применяется для обеспечения надёжной работы индивидуального и группового транспорта без гарантий. Для индивидуального (unicast) транспорта с гарантиями алгоритм фактически не нужен и опускается (параграф 4.2). DNCP поддерживает несколько состояний Trickle, как указано в разделе 5. Каждое из таких состояний может быть основано на разных параметрах (см. ниже) и отвечает за обеспечение конкретному или всем партнёрам на определённой конечной точке регулярное представление текущего локально рассчитанного хэша состояния сети для сравнения состояний, т. е. обнаружения возможного расхождения воспринимаемого состояния сети.

Trickle определяет 3 параметра: Imin, Imax, k. Параметры Imin и Imax представляют минимальное значение I и максимальное число удвоений Imin, где I - интервал, в течение которого конечная точка должна увидеть не менее k обновлений Trickle для предотвращения передачи локального состояния. Фактически предлагаемые параметры алгоритма Trickle зависят от профиля DNCP, как указано в разделе 9.

Состояние для всех экземпляров Trickle, определённых в разделе 5, считается несогласованным и сбрасывается, когда меняется локально рассчитанный хэш сети (и только в этом случае). Это происходит в результате изменения данных самого локального узла или получения более свежих данных от другого узла, как описано в параграфе 4.1. Узлу **недопустимо** сбрасывать своё состояние Trickle на основе лишь получения Network State TLV (параграф 7.2.2) с отличным от локально рассчитанного хэшем состояния сет.

Каждый раз, когда конкретный экземпляр Trickle указывает, что нужно отправить обновление, локальный узел должен передать Network State TLV (параграф 7.2.2), если выполняются указанные ниже условия (и только в этом случае).

- Конечная точка работает в режиме Multicast+Unicast. TLV в этом случая **должен** передаваться как групповой.
- Конечная точка работает в режиме, отличном от Multicast+Unicast и индивидуальный транспорт не обеспечивает гарантий. В этом случае TLV **должен** передаваться как unicast.

Можно также включать все или часть Node State TLV (параграф 7.2.3), если в профиле DNCP не указана нежелательность этого или не требуется предотвратить раскрытие TLV при групповой передаче путём отправки по защищённому индивидуальному транспорту.

4.4. Обработка принятых TLV

В этом параграфе описана обработка принятых TLV. Профиль DNCP может задавать игнорирование отдельных TLV, например, для изменения свойств защиты (в разделе 9 указано, для чего профиль может задавать безопасное игнорирование). Любой отклик (reply), упомянутый ниже, означает передачу конкретных TLV инициатору обрабатываемого TLV. Все такие отклики **должны** передаваться по индивидуальным адресам. Если TLV, для которого передаётся отклик, принят как групповой и в несколько каналов доступа, отклик **должен** быть задержан на случайное время из интервала [0, Imin/2] для предотвращения возможности одновременной отправки, которая может вызывать проблемы на некоторых каналах, если в профиле DNCP не указано иное. Для откликов **можно** также ограничивать скорость передачи на короткое время, определяемое реализацией. Однако, если TLV не запрещён профилем DNCP, реализация **должна** отвечать на повторные TLV с отличной от 0 вероятностью, чтобы не возникло «истощения» (starvation), нарушающего синхронизацию состояний.

Узел DNCP **должен** обрабатывать TLV, полученные с любого действительного (например, с корректной областью действия) адреса, как задано профилем DNCP и настройкой конкретной конечной точки, независимо от того, принадлежит ли адрес партнёру. Это условие удовлетворяет потребности мониторинга и других программ хоста, которым требуется определять топологию DNCP без добавления состояний в сеть.

Ниже описаны действия при получении разных TLV.

- Request Network State TLV (параграф 7.1.1). Получатель **должен** ответить Network State TLV (параграф 7.2.2) и Node State TLV (параграф 7.2.3) для каждого узла данных, использованного при расчёте хэша состояния сети. В Node State TLV **не следует** включать необязательную часть данных узла для предотвращения избыточной передачи данных узла, если в профиле DNCP явно не указано иное.
- Request Node State TLV (параграф 7.1.2). Если получатель имеет данные для соответствующего узла, он **должен** ответить Node State TLV (параграф 7.2.3) для соответствующего узла. Необязательная часть данных узла **должна** включаться в TLV.
- Network State TLV (параграф 7.2.2). Если хэш состояния сети отличается от рассчитанного локально и получатель не знает о каких-либо конкретных различиях в состоянии узла с отправителем, он **должен** ответить Request Network State TLV (параграф 7.1.1). Эти отклики **должны** ограничиваться по частоте передачи (не более 1 на канал для уникального хэша состояния сети в интервале Imin). Простейший способ обеспечить это ограничение заключается в использовании временных меток, указывающих запросы, и передачи не более одного Request Network State TLV (параграф 7.1.1) в интервале Imin. Для обеспечения более быстрой синхронизации состояний при передаче в ответ Request Network State TLV **можно** отправлять также текущий локальный Network State TLV MAY.
- Node State TLV (параграф 7.2.3).
  - Если идентификатор узла совпадает с идентификатором локального узла, а TLV имеет больший порядковый номер чем у текущего или номера совпадают, но различаются хэши, узлу **следует** заново опубликовать свои данные с порядковым номером существенно больше полученного (например, на 1000), чтобы повторно заявить идентификатор узла. Такое различие нужно для того, чтобы номер гарантированно превышал номера всех возможно существующих в сети копий состояния узла. Обычно такая ситуация возникают однократно при перезагрузке локального узла без сохранения последнего использованного номера. Для случаев, когда это происходит несколько раз или для узлов, не публикующих повторно свои данные, профиль DNCP **должен** включать руководство по обработке таких ситуаций, поскольку они указывают наличие другого активного узла с тем же идентификатором.
  - Если идентификатор узла не совпадает с локальным и выполняется любое из условий:
    - локальная информация для соответствующего узла устарела (локальный номер меньше чем в TLV);
    - локальная информация может быть некорректна (номера совпадают, но данные различаются);
    - для этого узла нет данных,

    выполняются указанные ниже действия.
    - Если TLV содержит поле Node Data, его следует проверить, убедившись, что локально рассчитанный хэш совпадает с полем H(Node Data) в TLV. При несовпадении **следует** игнорировать TLV.
    - Если TLV не содержит поле Node Data, а H(Node Data) в TLV отличается от локального хэша данных узла (или его нет), получатель **должен** ответить Request Node State TLV (параграф 7.1.2) для соответствующего узла.

- В остальных случаях получатель **должен** обновить сохранённое локально состояние для этого узла (данные узла на основе поля Node Data, если оно есть, порядковый номер и относительное время) в соответствии с полученным TLV.

Для сравнения порядковых номеров **должна** применяться функция циклического сравнения, чтобы избежать проблем при достижении максимума. **Рекомендуется** функция сравнения a < b <=> ((a - b) % (2^{32})) & (2^{31}) != 0 где (a % b) представляет остаток при делении по модулю a на b, a (a & b) - побитовое объединение (конъюнкцию) a и b, если профиль DNCP не задаёт иную.

- Прочие TLV. TLV, не распознанные получателем, **должны** просто игнорироваться, если они не были переданы в другом TLV (например, TLV в поле Node Data внутри Node State TLV). Нераспознанные TLV в поле Node Data внутри Node State TLV **должны** сохраняться для распространения другим узлам и расчёта хэша данных узла, как описано в параграфе 7.2.3, но игнорироваться в остальных случаях.

Если для конечной точки настроен индивидуальный транспорт с гарантиями, все Node State TLV, полученные в незащищённых групповых пакетах, **должны** просто игнорироваться.

4.5. Обнаружение, добавление и удаление партнёров

Партнерские отношения организуются между соседями с помощью одной или нескольких пар соединённых между собой конечных точек. Такие соседи напрямую обмениваются информацией о состоянии сети и опубликованных данных, в затем эта информация транзитивно распространяется по сети.

Поиск новых партнёров выполняется с использованием обычного индивидуального или группового транспорта, заданного в профиле DNCP (раздел 9). Этот процесс не отличается от добавления партнёра, т. е. неизвестный партнёр просто обнаруживается при получении от него обычных TLV протокола DNCP и специальных сообщений или TLV для обнаружения не применяется. Для транспорта с поддержкой лишь unicast транспортные адреса отдельных устройств настраиваются заранее или получаются с использованием внешнего протокола обнаружения служб. При наличии группового транспорта сообщения от неизвестных партнёров обрабатываются так же, как групповые сообщения от известных, т. е. новые партнёры обнаруживаются при передаче ими обычных TLV протокола DNCP по групповому адресу.

При получении конечной точкой Node Endpoint TLV (параграф 7.2.1) от неизвестного партнёра выполняются указанные ниже действия.

- При получении индивидуально удалённый узел **должен** быть добавлен конечной точкой в число партнёров и для него **должен** быть создан Peer TLV (параграф 7.3.1).
- При получении группового TLV узел **может** передать (возможно с ограничением частоты) индивидуальный Request Network State TLV (параграф 7.1.1).

Если сообщения keep-alive, заданные в параграфе 6.1, **не** передаются партнёром (профиль DNCP не задаёт их использование или конкретный партнёр отказался от передачи keep-alive), **должны** применяться другие меры, предоставляемые локальным транспортом (такие как обнаружение несущей Ethernet или TCP keep-alive), для обозначения присутствия. Если партнёр не передаёт keep-alive и нет средств проверки его присутствия, он **должен** считаться отсутствующим и его **не следует** добавлять снова, пока он не возобновит передачу keep-alive. Когда партнёр больше не присутствует, Peer TLV и локальное состояние DNCP для партнёра **должны** удаляться. DNCP не определяет явных сообщений или TLV, указывающих прерывание операции DNCP прерывающим узлом, однако производные протоколы при необходимости могут задавать такие расширения.

Если локальная конечная точка находится в транспортном режиме Multicast-Listen+Unicast, Peer TLV (параграф 7.3.1) **недопустимо** публиковать для партнёров, не имеющих более высокого значения идентификатора узла.

4.6. Проверка живучести данных

Поддержка хэш-дерева (параграф 4.1) и обновление хэшей состояния сети зависят от актуальности информации о двухсторонней достижимости узла, выводимой из содержимого графа топологии. Граф меняется при добавлении и удалении узлов сети, а также при организации или потере двухсторонней связности между имеющимися узлами, поэтому он **должен** обновляться сразу же или с задержкой меньше заданного профилем DNCP значения Trickle Imin после любого из указанных ниже событий.

- Добавление или удаление Peer TLV или узла в целом.
- Время создания данных того или иного узла (в миллисекундах) раньше, чем (текущее время - (2^{32} + 2^{15})).

Искусственный верхний предел времени создания служит для аккуратного предотвращения переполнения и возможности повторной публикации данных узлом, как указано в параграфе 7.2.3.

Обновление графа топологии начинается с пометки локального узла достижимым, а всех остальных - недоступными. Затем остальные узлы итеративно помечаются как доступные по специальному алгоритму. Ещё не достижимый узел-кандидат N с конечной точкой NE помечается как доступный, если имеется узел R с конечной точкой RE, для которого выполняются все указанные ниже условия.

- Время создания данных узла R (в миллисекундах) больше чем (текущее время - (2^{32} + 2^{15})).
- R публикует Peer TLV с:
  - Peer Node Identifier = идентификатор узла N;
  - Peer Endpoint Identifier = идентификатор конечной точки NE;
  - Endpoint Identifier = идентификатор конечной точки RE.
- N публикует Peer TLV с:
  - Peer Node Identifier = идентификатор узла R;

- Peer Endpoint Identifier = идентификатор конечной точки RE;
- Endpoint Identifier = идентификатор конечной точки NE.

Алгоритм завершается, когда больше нет кандидатов, соответствующих приведённым выше требованиям.

Узлы DNCP, которые были недоступны при последнем обходе графа топологии, **недопустимо** использовать при расчёте хэша состояния сети, предоставлять приложениям, которым нужно использовать весь граф TLV, или предоставлять удаленным узлам. О них **можно** забыть сразу после обхода дерева топологии, однако **рекомендуется** сохранять их хотя бы ненадолго для повышения скорости схождения состояния сети DNCP. Это снижает число запросов при начальном схождении сети, а также в случаях потери и восстановления двухсторонней связности в течение этого времени.

5. Модель данных

В этом разделе описаны локальные структуры данных, которые может использовать минимальная реализация. Раздел предназначен лишь для удобства разработчиков. Некоторые необязательные расширения (раздел 6) описывают свои требования к данным и некоторые необязательные части ядра протокола могут вносить свои требования.

Узел DNCP имеет структуру данных, содержащую сведения о недавно переданных Request Network State TLV (параграф 7.1.1). Простейшим вариантом является сохранение временных меток последних запросов (нужны для ограничения скорости передачи откликов, как описано в параграфе 4.4).

Для каждого узла в сети DNCP узел DNCP имеет указанные ниже сведения.

- Идентификатор узла, однозначно указывающий узел. Размер, способ создания и обработка конфликтов определяются в профиле DNCP.
- Данные узла, включающие набор TLV, опубликованных этим узлом. Поскольку данные передаются в определённом порядке (см. параграф 7.2.3), может быть разумным сохранение этого порядка в структуре.
- Последний порядковый номер - 32-битовое значение, увеличиваемое при публикации каждого набора TLV. Функция сравнения описана в параграфе 4.4.
- Время создания - (оценочное) время публикации текущего набора TLV с текущим порядковым номером. Это время используется в поле Milliseconds Since Origination блока Node State TLV (параграф 7.2.3). В идеальном случае обеспечивает миллисекундную точность.

Узел DNCP имеет набор конечных точек, для которых настроено применение DNCP, с указанными ниже параметрами.

- Идентификатор конечной точки - 32-битовое неанализируемое (opaque) уникальное в локальном масштабе значение, указывающее конечную точку на узле. **Не следует** повторно использовать идентификатор сразу после отключения конечной точки.
- Экземпляр Trickle с параметрами I, T, c (только для точек в транспортном режиме Multicast+Unicast).

Для конечной точки также поддерживается один или несколько перечисленных ниже параметров.

- Интерфейс - локально назначенный сетевой интерфейс.
- Индивидуальный адрес, указывающий узел DNCP, с которым следует соединяться.
- Набор адресов узлов DNCP, от которых принимаются соединения.

Для каждой удалённой пары (партнёр, конечная точка), обнаруженной на локальной конечной точке, узел DNCP имеет:

- идентификатор узла, однозначной указывающий партнёра;
- идентификатор конечной точки, однозначно указывающий её у партнёра;
- адрес партнёра - последний использованный партнёром адрес (аутентифицированный и уполномоченный при включённой защите);
- экземпляр Trickle с параметрами I, T, c (только для конечных точек в режиме Unicast при использовании индивидуального транспорта без гарантий).

6. Дополнительные расширения

В этом разделе указаны расширения ядра протокола, которые могут быть заданы профилем DNCP.

6.1. Keep-Alive

Хотя DNCP предоставляет механизмы для обнаружения и добавления новых партнёров на конечной точке (параграф 4.5), а также уведомления о смене состояний, могут потребоваться иные механизмы для избавления от устаревших и недействительных партнёров, если транспортный или нижележащие уровни не обеспечивают их, как указано в параграфе 4.6.

Если в профиле DNCP не заданы сообщения keep-alive, остальная часть этого параграфа **должна** игнорироваться.

Профиль DNCP **может** задавать поддержку keep-alive на уровне конечных точек (передаются по групповому адресу всем узлам DNCP, подключённым к multicast-каналу) или партнёров (передаются индивидуально).

Для каждой конечной точки, для которой в профиле DNCP заданы сообщения keep-alive, **должен** поддерживаться свой интервал keep-alive (по умолчанию DNCP_KEEPALIVE_INTERVAL). Если предпочтительно заданное локально значение по той или иной причине (настройка, энергосбережение, тип среды и т. п.), используется это значение. Если какая-либо конечная точка применяет отличное от принятого по умолчанию значение, узел DNCP **должен** публиковать соответствующие Keep-Alive Interval TLV (параграф 7.3.2) в данных узла.

6.1.1. Дополнения модели данных

Для поддержки keep-alive нужны указанные ниже дополнения к модели данных (раздел 5).

Для каждой настроенной конечной точки, где включены keep-alive на уровне конечных точек:

- время последней передачи, если метка времени, указывающая последний блок Network State TLV (параграф 7.2.2), была передана через этот интерфейс.

Для каждой удалённой пары (партнёр, конечная точка), обнаруженной на локальной конечной точке, узел DNCP имеет указанные ниже значения.

- Метка времени последнего контакта, указывающая момент получения последнего согласованного Network State TLV (параграф 7.2.2) от партнёра в групповом режиме или любой информации в индивидуальном режиме (unicast). Если метка не обновлена в течение некоторого времени, как указано в параграфе 6.1.5, партнёр удаляется. При добавлении партнёра для метки устанавливается текущее время.
- Время последней передачи. При включённых keep-alive для партнёров это будет временная метка, указывающая момент последней передачи Network State TLV (параграф 7.2.2) данному партнёру point-to-point. При добавлении партнёра для метки устанавливается текущее время.

6.1.2. Периодические Keep-Alive для конечной точки

Если включены keep-alive для конечных точек в транспортном режиме Multicast+Unicast и трафик с Network State TLV (параграф 7.2.2) не передавался конкретной конечной точке в течение заданного для неё интервала keep-alive, этой конечной точке **должен** передаваться Network State TLV (параграф 7.2.2) с запуском нового интервала Trickle, как указано в п. 2 параграфа 4.2 [RFC6206]. Фактическую передачу **следует** задерживать на случайное время из интервала [0, Imin/2].

6.1.3. Периодические Keep-Alive для партнёра

Если включены keep-alive для партнёров на конечной точке, поддерживающей лишь индивидуальный транспорт и не было передано трафика с Network State TLV (параграф 7.2.2) в заданном для конечной точки интервале keep-alive, партнёру. **должен** передаваться Network State TLV (параграф 7.2.2) с запуском нового интервала Trickle, как указано в п. 2 параграфа 4.2 [RFC6206].

6.1.4. Дополнения к обработке принятых TLV

При получении TLV от партнёра индивидуально для этого партнёра **должна** обновляться метка времени контакта.

При получении Network State TLV (параграф 7.2.2), соответствующего локально рассчитанному кэшу состояния сети, для этого **должна** обновляться метка времени контакта, чтобы сохранить его как партнёра.

6.1.5. Удаление партнёра

Для каждого партнёра на каждой конечной точке интервал keep-alive для конечной точки должен рассчитываться с использованием Keep-Alive Interval TLV (параграф 7.3.2), опубликованных узлом, а при их отсутствии применяется принятое по умолчанию значение DNCP_KEEPALIVE_INTERVAL. Если метка времени последнего контакта с партнёром не обновлялась в течение по меньшей мере потенциально зависящего от конечной точки локального числа интервалов keep-alive (по умолчанию DNCP_KEEPALIVE_MULTIPLIER) для конечной точки партнёра, для этого партнёра **должны** удаляться Peer TLV и локальное состояние партнёра DNCP.

6.2. Поддержка каналов Multicast с высокой плотностью узлов

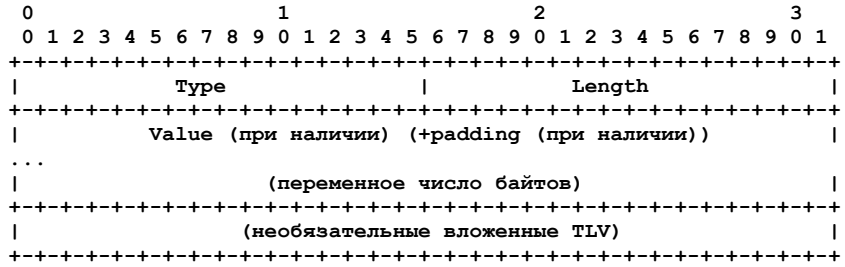
Эта оптимизация нужна для предотвращения взрывного роста пространства состояний. С учётом большого числа узлов DNCP, публикующих данные на конечной точке, использующей групповую адресацию на канале, каждый узел будет добавлять Peer TLV (параграф 7.3.1) для каждого партнёра. Хотя Trickle до некоторой степени ограничивает объем трафика на канале в стабильном состоянии, общий объем данных, добавляемых и поддерживаемых в сети DNCP с числом узлов N, на канале с поддержкой групповой адресации составляет O(N2). Кроме того, при использовании keep-alive на уровне партнёров на канале будет O(N2) сообщений keep-alive, если жизненность партнёров не подтверждается иным путём (например, срок действия соединения TCP, уведомление L2 или keep-alive на уровне конечной точки).

Верхнюю границу числа партнёров, разрешённого для отдельного типа каналов, которую использует конечная точка в транспортном режиме Multicast+Unicast, **следует** задавать в профиле DNCP, но она **может** быть выбрана и в процессе работы. Основное внимание при выборе границы для конкретного типа канала (если она задаётся) следует уделять вопросу поддержки группового трафика и большого числа партнёров при использовании данного профиля DNCP на канале. Если то и другое маловероятно, нет смысла задавать поддержку расширения для этого типа канала.

Если профиль DNCP совсем не поддерживает это расширение, остальная часть параграфа **должна** игнорироваться. Это обусловлено тем, что при использовании расширения состояние в сети DNCP включает лишь часть полной топологии сети. Поэтому каждый узел должен знать о возможности его использования в конкретном профиле DNCP.

Если заданная верхняя граница превышена для какой-либо конечной точки с режимом транспорта Multicast+Unicast и узел не имеет наивысшего идентификатора на канале, ему **следует** считать конечную точку индивидуально (unicast) подключённой к узлу, имеющему наивысший идентификатор на канале, и переходить в режим Multicast-listen+Unicast. Влияние на поведение конкретной конечной точки рассмотрено в параграфе 4.2. Узлы в режиме Multicast-listen+Unicast **должны** сохранять прослушивание группового трафика для получения сообщений от узлов, остающихся в режиме Multicast+Unicast, и реакции на появлении узла с большим идентификатором. Если наибольший идентификатор на канале меняется, **должен** измениться удалённый unicast-адрес конечных точек в режиме Multicast-Listen+Unicast. Если идентификатор локального узла является высшим на канале, узел **должен** переключиться обратно или остаться в режиме Multicast+Unicast и организовать партнерские отношения со всеми партнёрами, как указано в параграфе 4.5.

7. Объекты TLV



Каждый блок TLV содержит:

- 2-байтовое поле Type;
- 2-байтовое поле Length, указывающее размер Value в байтах, 0 говорит об отсутствии значения;
- значение, если оно имеется;
- необязательные байты заполнения (0) до следующей 4-байтовой границы, если Length не кратно 4.

Хотя байты заполнения **недопустимо** учитывать в поле Length блока TLV, при наличии вложенных TLV их заполнение включается в поле Length внешнего (охватывающего) TLV.

Каждый TLV, не определяющий необязательных полей и содержимого переменного размера, **может** передаваться с дополнительными суб-TLV, добавленными после него для поддержки расширяемости. При обработке таких типов TLV каждый узел **должен** воспринимать полученные TLV, размер которых превышает размер фиксированных полей для данного типа, и игнорировать суб-TLV неизвестного или неподдерживаемого в данном TLV типа. При наличии суб-TLV поле Length в основном TLV указывает число байтов от первого байта поля Value (при наличии) до последнего байта заполнения в последнем суб-TLV, включительно. Например, TLV с type=123 (0x7b) и значением x (120 = 0x78) представляется как 007B 0001 7800 0000. Если в него включён суб-TLV типа 124 (0x7c) со значением y, кодирование будет иметь вид 007B 000C 7800 0000 007C 0001 7900 0000.

В этом разделе используются указанные ниже специальные обозначения.

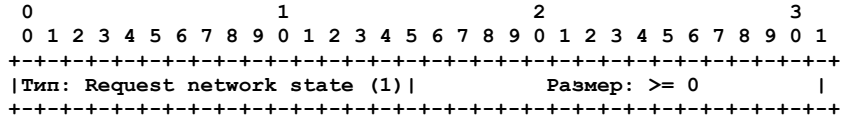
.. - конкатенация (объединение) строк.

H(x) - некриптографическая хэш-функция, заданная профилем DNCP.

В дополнение к определенным здесь типам TLV значения типов 11-31 и 512-767 зарезервированы и могут быть зарегистрированы позднее (начиная с 11) по процедуре Standards Action [RFC5226] расширениями DNCP, которые могут применяться в разных профилях DNCP.

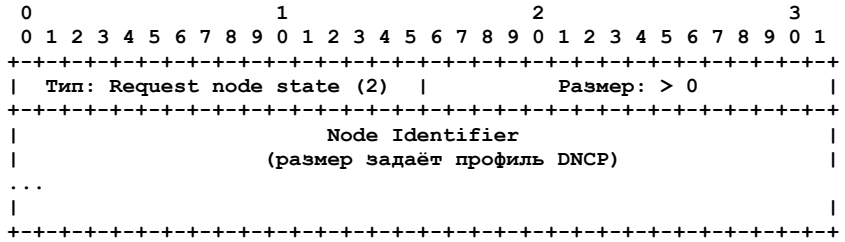
7.1. TLV запросов

7.1.1. Request Network State TLV



Запрашивает отклик с Network State TLV (параграф 7.2.2) и всеми Node State TLV (параграф 7.2.3) без данных узла.

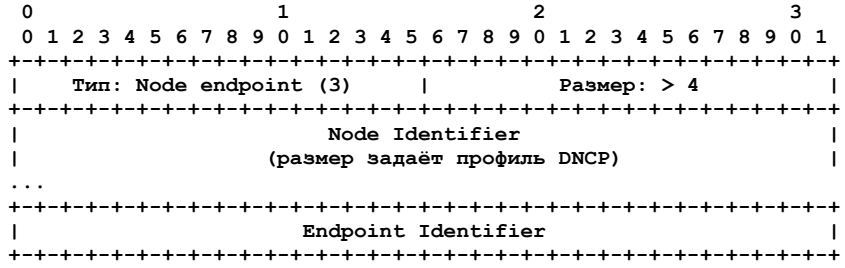
7.1.2. Request Node State TLV



Запрашивает Node State TLV (параграф 7.2.3) с данными узла, соответствующего идентификатору.

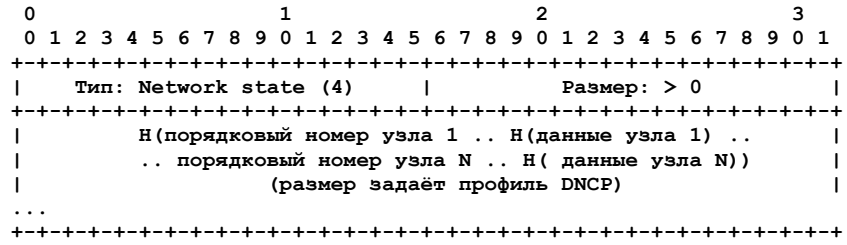
7.2. TLV с данными

7.2.1. Node Endpoint TLV



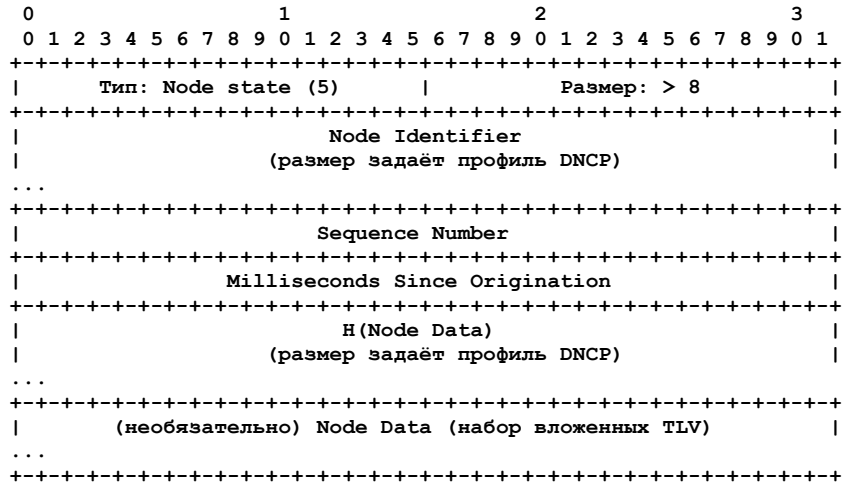
Указывает идентификаторы локального узла и конкретной конечной точки. Передача этого TLV описана в параграфе 4.2.

7.2.2. Network State TLV



Передаёт хэш текущего состояния сети, рассчитанный отправителем (алгоритм описан в параграфе 4.1).

7.2.3. Node State TLV



Этот TLV представляет знание локального узла об опубликованном состоянии узла сети DNCP, указанного полем Node Identifier в TLV.

Каждый узел, включая публикующий свои данные, **должен** обновлять поле Milliseconds Since Origination всякий раз при передаче Node State TLV на основе оценки узлом времени исходной публикации данных. Это делается, например, для того, чтобы относительные метки времени в публикуемых данных узла можно было корректно интерпретировать. В конечном итоге предсказание является просто оценкой, поскольку не учитывает задержек при передаче.

Если исходный узел замечает, что 32-битовое поле Milliseconds Since Origination близко к переполнению (больше 232 - 216), он **должен** заново опубликовать TLV, даже при отсутствии в них изменений. Иными словами, при отсутствии изменений узел **должен** заново публиковать набор TLV примерно каждые 48 дней.

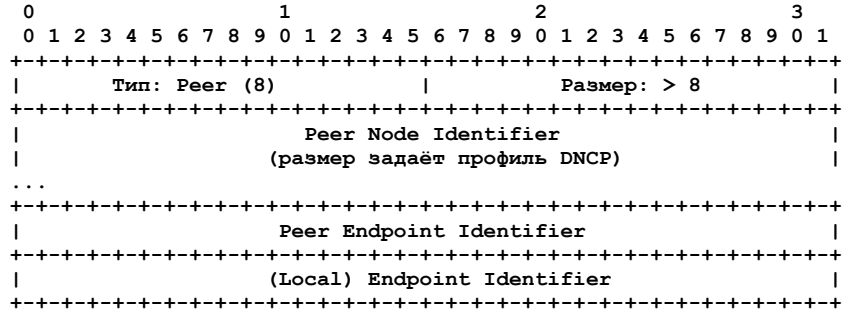
Фактические данные узла могут включаться в TLV, а также в необязательное поле Node Data. Набор TLV **должен** строго упорядочиваться по возрастанию двоичного содержимого (включая тип и размер TLV). Это позволяет получателю, например, эффективно обрабатывать изменения и индексировать по типу TLV без копирования. Содержимое данных узла **должно** передаваться в том виде, как оно было получено.

При получении **следует** также проверять совпадение локально рассчитанного значения H(Node Data) с содержимым поля в TLV и при наличии расхождений TLV **следует** игнорировать.

7.3. TLV данных в Node State TLV

Эти TLV публикуются узлами DNCP и применяются лишь в поле Node Data блоков Node State TLV. При появлении вне Node State TLV они **должны** игнорироваться.

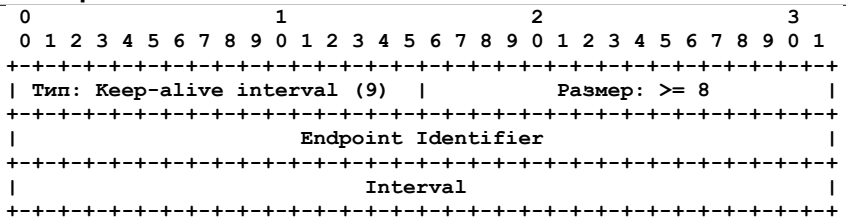
7.3.1. Peer TLV



Указывает, что рассматриваемый узел подтверждает доступность указанного партнёра с заданной локальной конечной точки. Наличие этого TLV по меньшей мере гарантирует, что публикующий его узел недавно получил трафик от партнёра. Для гарантии фактической двухсторонней достижимости требуется проверить наличие соответствующих Peer TLV обоих узлов.

7.3.2. Keep-Alive Interval TLV

Указывает использование иного, чем принятый по умолчанию, интервала keep-alive (параграф 6.1).



Идентификатор конечной точки служит для указания отдельной (локальной) конечной точки, для которой на передающем узле применяется данный интервал. Значение 0 указывает **все** конечные точки, где нет конкретного TLV.

Поле Interval указывает число миллисекунд между отправкой узлом keep-alive. Значение 0 отключает передачу keep-alive и в таких случаях **должен** быть доступен механизм нижележащего уровня, гарантирующий присутствие узлов.

8. Управление защитой и доверием

Если это задано в профиле DNCP, может применяться DTLS [RFC6347] или TLS [RFC5246] для аутентификации и шифрования части (если профиль не задаёт обязательность) или всего unicast-трафика. Ниже определены методы организации доверия, а возможность, желательность или обязательность их применения задаёт профиль DNCP.

8.1. Доверие на основе PSK

Модель доверия на основе заранее распределенных ключей (Pre-Shared Key или PSK) обеспечивает простой механизм управления защитой, позволяющий администратору развёртывать устройства в имеющейся сети, настраивая на них предопределённый ключ, подобно настройке пароля администратора или ключа защиты Wi-Fi (Wi-Fi Protected Access или WPA. Это ограниченный, но полезный и удобный для небольших сетей механизм защиты.

8.2. Доверие на основе PKI

Модель доверия на основе PKI обеспечивает расширенные возможности управления, но требует больших усилий при настройке и начальной загрузке. Однако модель обеспечивает централизованное управление, поэтому она полезна в больших сетях, где требуется полномочное управление доверием.

8.3. Согласование доверия на основе сертификатов

В некоторых случаях, таких как начальная загрузка в частично сети управляемой сети, описанные выше методы не могут обеспечить желаемый баланс защиты и удобства для пользователей. В этом параграфе приведено руководство по реализации метода гибкой защиты [RFC7435], на основе которого можно создать профили DNCP, приспособленные к конкретным требованиям. Модель согласования на основе сертификатов разработана как компромисс между усилиями по управлению доверием и гибкостью. Она основана на сертификатах X.509 и позволяет каждому узлу DNCP вынести вердикт доверия по любому другому сертификату и прийти к соглашению в части доверия к узлу использующему этот сертификат или другой сертификат, подписанный им.

Узлы DNCP, не использующие этот метод защиты, **должны** игнорировать все анонсируемые вердикты доверия и им **недопустимо** анонсировать такие вердикты, т. е. требования этого параграфа к нему не применяются.

Текущий действующий вердикт доверия для любого сертификата определяется как вердикт с наибольшим приоритетом из числа вердиктов доверия, объявленных на данный момент к для указанного сертификата.

8.3.1. Вердикты доверия

Вердикты доверия - это заявления узлов DNCP о доверию к сертификатам X.509. Существует 5 возможных вердиктов, указанных ниже в порядке роста значимости (приоритета).

0 (Neutral - нейтральный) - вердикта доверия нет и сети DNCP следует определить его.

1 (Cached Trust - кэшированное доверие) - последний известный вердикт - Trust (Configured или Cached).

2 (Cached Distrust - кэшированное недоверие) - последний известный вердикт - Distrust (Configured или Cached).

3 (Configured Trust - настроенное доверие) - доверие на основе внешней церемонии или настройки.

4 (Configured Distrust - настроенное недоверие) - недоверие на основе внешней церемонии или настройки.

Вердикты доверия собраны в 3 группы:

- настроенные (Configured) вердикты доверия применяются для анонсов узла явных вердиктов, основанных на внешней доверенной загрузки или предопределённых отношений, сформированных для данного сертификата.
- Кэшированные (Cached) вердикты доверия служат для анонсов сохранения последнего известного состояния в случае, когда все узлы с настроенными вердиктами были отсоединены или выключены.
- Нейтральный (Neutral) вердикт служит для анонсирования нового узла, намеревающегося подключиться к сети, когда окончательный вердикт ещё не определён.

В качестве текущего действующего вердикта доверия для сертификата применяется вердикт с наибольшим приоритетом из числа анонсированных для этого сертификата в сети DNCP. Узел **должен** считаться доверенным для участия в сети DNCP тогда и только тогда, когда текущим действующим вердиктом доверия к его сертификату или любому из сертификатов в его иерархии является Trust (Cached или Configured) и ни один из сертификатов в его иерархии не имеет текущего действующего вердикта Distrust (Cached или Configured). Если узел имеет настроенный вердикт, отличающийся от текущего действующего вердикта доверия для сертификата, последний имеет преимущество при решении вопроса о доверии. Несмотря на это, узел поддерживает и анонсирует настроенный вердикт.

8.3.2. Кэш доверия

Каждому узлу **следует** поддерживать кэш доверия, содержащий текущие действующие вердикты для всех сертификатов, анонсируемых в данный момент в сети DNCP. Этот кэш служит резервной копией последнего известного состояния, если нет узла, анонсирующего настроенный вердикт для известного сертификата. Кэш следует хранить в энергонезависимой памяти в течение разумного срока на случай перезагрузки или потери питания.

Каждый раз при соединении узла с сетью или обнаружении смены текущего действующего вердикта для любого сертификата, узел синхронизирует свой кэш, переписывая сохранённые ранее вердикты. Настроенные вердикты хранятся в кэше как соответствующие кэшированные дубликаты. Нейтральные вердикты не сохраняются и не переопределяют имеющиеся в кэше вердикты.

8.3.3. Анонсирование вердиктов

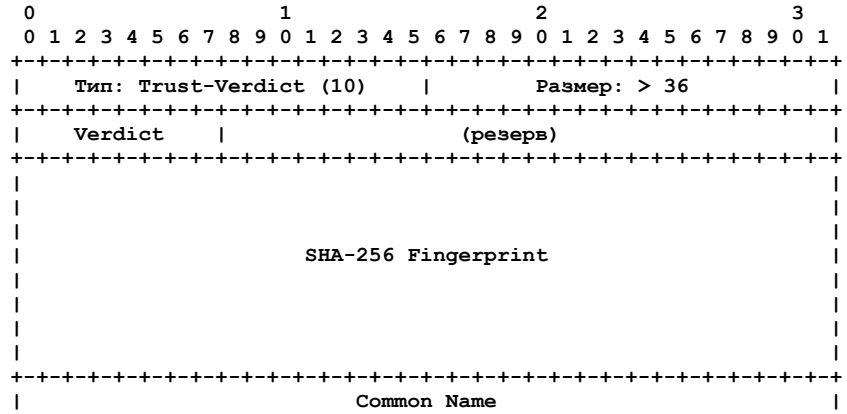
Узлу **следует** анонсировать все созданные им вердикты и он **должен** делать это, если анонс настроенного вердикта ведёт к смене текущего действующего вердикта для соответствующего сертификата. При отсутствии настроенных вердиктов узел **должен** анонсировать вердикты Cached Trust из своего кэша доверия, если выполняется 1 из условий:

- сохранён вердикт Cached Trust, а текущий действующий вердикт для сертификата - Neutral или его нет;
- сохранен вердикт Cached Distrust, а текущий действующий вердикт для сертификата - Cached Trust.

Узел проверяет эти условия при каждом обнаружении изменений анонсированных в сети вердиктов доверия.

При обнаружении узла с иерархией сертификатов, для которой нет действующего вердикта доверия, узел добавляет Neutral Trust-Verdict TLV в свои данные для всех сертификатов, найденный в иерархии, и публикует его, пока не будет найден отличный от Neutral вердикт доверия для любого из сертификатов или не пройдёт разумное время (предлагается 10 минут) без реакции на это и дальнейших попыток аутентификации. Такие вердикты доверия следует ограничивать по частоте и количеству для предотвращения атак с отказом в обслуживании (DoS).

Вердикты доверия анонсируются в Trust-Verdict TLV.



Verdict указывает численный индекс вердикта доверия.

Резервное поле оставлено на будущее и **должно** содержать в 0 при создании TLV и игнорироваться при анализе.

SHA-256 Fingerprint содержит хэш SHA-256 [RFC6234] для сертификата в формате DER.

Поле Common name имеет переменный размер (1-64 байта) и содержит базовое имя сертификата.

8.3.4. Церемонии начальной загрузки

Ниже описаны некоторые методы организации отношений доверия между узлами DNCP и сертификатами узлов. Доверие устанавливается двухсторонним процессом, где существующая сеть должна доверять новому узлу, а такой узел - хотя бы одному из своих партнёров. Поэтому от нового и уже доверенного узла требуется выполнение такой церемонии для успешного внедрения нового узла в сеть DNCP. В любом случае администратору **должны** быть предоставлены внешние средства для идентификации узла, связанного с сертификатом, на основе «отпечатка» (fingerprint) и значимого базового имени.

8.3.4.1. Доверие по идентификации

Узел с доверием на основе сертификатов **должен** предоставлять интерфейс для извлечения текущего набора действующих вердиктов доверия, отпечатков, имён всех известных в настоящий момент сертификатов и настроенных вердиктов доверия. Как вариант, он **может** предоставить сопутствующий узел DNCP или приложение с такими возможностями, с которым заранее установлены отношения доверия.

8.3.4.2. Заранее настроенное доверие

На узле **можно** заранее настроить доверие к набору сертификатов узла или удостоверяющего центра (УЦ или CA). Однако таким отношениям **недопустимо** приводить к нежелательному или не связанному с делом доверию к узлам, не предназначенным для работы в одной сети (например, к другим устройствам того же производителя).

8.3.4.3. Доверие по нажатию кнопки

Узел **может** предоставлять физический или виртуальный интерфейс для временного перевода одного или нескольких внутренних сетевых интерфейсов в режим доверия к сертификату первого узла DNCP с которым удалось соединиться.

8.3.4.4. Доверие при первом использовании

Узел, не связанный с другим узлом DNCP, **может** доверять сертификату первого узла DNCP, с которым удалось соединиться. Этот метод **недопустимо** применять, когда узел уже связан с каким-либо узлом DNCP.

9. Определения, связанные с профилем DNCP

Каждый профиль DNCP **должен** задавать перечисленные ниже аспекты.

- Используемый протокол с индивидуальной и (необязательно) групповой адресацией. Если желательная поддержка групповой адресации для узлов и обнаружения, должен быть доступен транспорт на основе дейтаграмм.
- Способ защиты выбранного транспорта. Без защиты, необязательная защита, использование определённой здесь схемы TLS с одним или несколькими методами или что-то иное. Если каналы узлов DNCP могут быть достаточно защищены или изолированы, можно запустить DNCP с защитой без использования дополнительной аутентификации и шифрования.
- Параметры транспортного протокола, такие как используемые номера портов или групповые адреса. Для индивидуальных, групповых и защищённых индивидуальных пакетов могут потребоваться свои параметры.
- TLV, игнорируемые при получении (например, по соображениям безопасности, в дополнение к указанным в параграфе 4.4. Хотя защита данных узла, опубликованных в Node State TLV, уже обеспечивается базовой спецификацией (при использовании защищённого индивидуального транспорта Node State TLV передаются только индивидуально, поскольку групповые при получении игнорируются), если профиль добавляет TLV, передаваемые вне данных узла, профилю следует указывать, следует ли игнорировать эти TLV при получении по групповому транспорту или индивидуальному транспорту без защиты. Профиль DNCP может указывать безопасное игнорирование следующих DNCP TLV:
  - принятые как групповые, кроме Node Endpoint TLV (параграф 7.2.1) и Network State TLV (параграф 7.2.2);
  - все групповые или незащищённые индивидуальные TLV, принятые со слишком высокой скоростью; Trickle обеспечит возможное схождение при условии снижения скорости в какой-то момент.
- Разрешение конфликтов идентификаторов узлов, как описано в параграфе 4.4. Основными вариантами являются назначение нового идентификатора одному или обоим узлам или подача сигнала о критической ошибке в сети DNCP.
- Диапазоны Imin, Imax и k, предлагаемые реализации для применения в алгоритме Trickle. Алгоритм не требует одинаковых значений во всех реализациях, но близкие по порядку значения помогут реализациям профиля DNCP вести себя более согласованно и облегчат оценку нижнего и верхнего предела для схождения сети.
- Используемая хэш-функция H(x) и реально используемый размер её вывода. Выбранная функция применяется для хэширования данных узлов и создания хэша состояния сети (хэширование хэш-значений узлов). Функция SHA-256 [RFC6234] рекомендуется для использования по умолчанию, но можно применять и некриптографические хэш-функции. При возникновении конфликта (коллизии) хэша состояния сети сеть будет фактически разделена на части, считающие, что в настоящий момент они больше не сходятся. Сеть сойдётся после того, как изменятся данные того или иного узла или конфликтующие Node State TLV будут переданы через разделение в результате обновления состояния под управлением Trickle (параграф 4.3) или как часть обработки принятых TLV (параграф 4.4). Если узел публикует данные, которые конфликтуют с ранее опубликованными данными зла, обновление может не распространяться (полностью) и будут использованы старые данные узла.
- DNCP_NODE_IDENTIFIER_LENGTH задаёт фиксированный размер идентификатора узла в байтах.
- Следует ли передавать keep-alive и режим передачи - для конечной точки (может потребоваться групповой транспорт) или партнёра. С сообщениями keep-alive связаны параметры, указанные ниже.
  - DNCP_KEEPALIVE_INTERVAL задаёт частоту передачи keep-alive по умолчанию (при использовании).
  - DNCP_KEEPALIVE_MULTIPLIER указывает допустимое число пропущенных DNCP_KEEPALIVE_INTERVAL (или указанного партнёром интервала keep-alive). Это значение применяется по умолчанию, если не задано других настроек.
- Будет ли поддерживаться оптимизация каналов с групповой адресацией и высокой плотностью узлов (параграф 6.2).

Некоторые рекомендации по выбору транспорта и защиты даны в Приложении B.

10. Вопросы безопасности

Протоколы на основе DNCP могут применять групповую передачу для указания смены состояний DNCP и сообщений keep-alive. Однако по таким каналам не передаются реально публикуемые TLV с данными. Поэтому атакующий может узнать лишь хэш-значения состояний в сети DNCP и попытаться инициировать unicast-синхронизацию между узлами на локальном канале. Поэтому узлы DNCP **должны** ограничивать скорость реакции на групповые пакеты.

При использовании DNCP для начальной загрузки сети в решениях на основе PKI могут возникать проблемы с проверкой сертификатов по причине недоступности точного времени или невозможности использовать сеть для проверки списков отозванных сертификатов (Certificate Revocation List) или выполнения online-проверки.

Механизм согласия на основе сертификатов, определённый в этом документе, позволяет отозвать согласие, однако в случае скомпрометированного устройства кэш доверия может быть «отравлен» до фактического отзыва, что позволит недоверенному устройству снова подключиться к сети, применяя другое отождествление. Для остановки такой атаки может потребоваться физическое вмешательство и сброс кэшей доверия.

11. Взаимодействие с IANA

Агентство IANA создало реестр DNCP TLV Types с 16-битовыми значениями в десятичном формате в разделе Distributed Node Consensus Protocol (DNCP). Значения регистрируются по процедуре Standards Action [RFC5226]. Начальное содержимое реестра приведено ниже.

0: Reserved (резерв);
1: Request network state (запрос состояния сети);
2: Request node state (запрос состояния узла);
3: Node endpoint (конечная точка узла);
4: Network state (состояние сети);
5: Node state (состояние узла);
6: Reserved for future use (ранее Custom, резерв на будущее);
7: Reserved for future use (ранее Fragment count, резерв на будущее);
8: Peer (партнёр);
9: Keep-alive interval (интервал keep-alive);
10: Trust-Verdict (вердикт доверия);
11-31: Unassigned (не распределены)
32-511: Reserved for per-DNCP profile use (резерв для использования в профилях DNCP);
512-767: Unassigned (не распределены);
768-1023: Reserved for Private Use [RFC5226] (резерв для приватного использования);
1024-65535: Reserved for future use (резерв на будущее).

12. Литература

12.1. Нормативные документы

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <http://www.rfc-editor.org/info/rfc6206>.

[RFC6234] Eastlake 3rd, D. and T. Hansen, "US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)", RFC 6234, DOI 10.17487/RFC6234, May 2011, <http://www.rfc-editor.org/info/rfc6234>.

12.2. Дополнительная литература

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <http://www.rfc-editor.org/info/rfc3493>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <http://www.rfc-editor.org/info/rfc6347>.

[RFC7435] Dukhovni, V., "Opportunistic Security: Some Protection Most of the Time", RFC 7435, DOI 10.17487/RFC7435, December 2014, <http://www.rfc-editor.org/info/rfc7435>.

[RFC7596] Cui, Y., Sun, Q., Boucadair, M., Tsou, T., Lee, Y., and I. Farrer, "Lightweight 4over6: An Extension to the Dual-Stack Lite Architecture", RFC 7596, DOI 10.17487/RFC7596, July 2015, <http://www.rfc-editor.org/info/rfc7596>.

Приложение A. Дополнительные режимы работы

Кроме описанных выше применения протокол поддерживает и другие варианты использования. Ниже приведено несколько примеров.

A.1. Работа в режиме «только чтение»

Если узел использует лишь одну конечную точку и ему не нужно публиковать TLV, полная функциональность DNCP не требуется. Такой узел с ограничениями может получить и поддерживать представление пространства TLV, реализуя логику обработки, описанную в параграфе 4.4. Такому узлу не нужен алгоритм Trickle, поддержка партнёров и даже keep-alive, поскольку их использование узлами DNCP гарантировало бы получение хэшей состояния сети и синхронизацию данных узла даже при использовании транспорта без гарантий.

A.2. Пересылка

Если узел имеет пару конечных точек, которым не нужно публиковать TLV, он может, например, обнаружить узлы с наибольшими идентификаторами на каждой из конечных точек (если они есть). Все TLV, полученные от одной из точек, узел будет передавать без изменения по индивидуальному адресу другого узла с наибольшим идентификатором.

Любые оплошности в TLV будут нарушать гарантии работы этой схемы, однако для пассивного мониторинга она подойдёт. Для такой пересылки нельзя создать цепочки, поскольку в ней нет проактивной пересылки.

Приложение B. Дополнительное руководство по профилям DNCP

В этом приложении разъяснены последствия выбора для профиля DNCP конкретного транспорта и опций защиты.

B.1. Индивидуальный транспорт - UDP или TCP?

Размер данных, публикуемых узлом DNCP, ограничен 64 Кбайт в результате использования 16-битового поля размера в TLV. Некоторые варианты транспорта могут вносить дополнительные ограничения. Например, при использовании индивидуальных дейтаграмм UDP верхняя граница размера данных узла определяется тем, что узлы и базовая сеть

могут передавать друг другу, поскольку DNCP не имеет механизма фрагментации. Профиль на основе UDP будет ограничен в размере данных узла (например, меньше принятого по умолчанию IPv6 MTU, если применяется IPv6) или будет задавать минимальный размер, который должны поддерживать все узлы. При использовании коммуникаций, включающих нелокальные каналы, приходится учитывать обработку фрагментов промежуточными устройствами. Поэтому применение потокового транспорта, такого как TCP, будет вероятно более оправданным, если нужна связь не только по локальным каналам или предполагается, что фрагментация может вызывать проблемы.

TCP также обеспечивает другие возможности, такие как сравнительно большой интервал встроенных сообщений keep-alive, которого в сочетании с закрытием соединений из-за отказов при повторной передаче может быть достаточно для блокировки использования протокольных сообщений keep-alive, описанных в параграфе 6.1. Кроме того, протокол обеспечивает гарантии, избавляющие от необходимости применять алгоритм Trickle на таких unicast-соединениях.

Основным недостатком использования TCP по сравнению с UDP в профилях на основе DNCP является утрата контроля над временем приёма TLV, хотя дейтаграммы UDP без гарантий доставки также имеют задержку. TLV в надёжном потоковом транспорте могут существенно задерживаться из-за повторов передачи. Это не вызывает проблем, если в TLV протокола на основе DNCP не хранится зависящих от времени сведений. В таких протоколах TCP является разумным выбором для индивидуального транспорта, если он доступен.

B.2. Необязательный групповой транспорт

Групповая передача требуется для динамического обнаружения партнёров или запуска индивидуального обмена, для этого в спецификации определён единственный вариант - транспорт дейтаграмм без гарантий доставки (обычно UDP), хотя протоколы на основе DNCP могут сами определять другой механизм доставки или обнаружения партнёров (например, на основе Multicast DNS - mDNS или DNS).

Для групповой передачи следует применять общеизвестный адрес (например, IPv6) для нужной области действия. В большинстве случаев подходят области действия link-local (локальный канал) или site-local (локальный сайт).

B.3. Необязательная защита транспорта

С точки зрения предоставляемой защиты протоколы DTLS и TLS эквивалентны и имеют одинаковое число состояний на устройствах. TLS работает на основе потокового протокола, но и при использовании DTLS также нужно достаточно длительное хэширование на уровне DTLS для предотвращения дорогостоящего повтора проверки подлинности и полномочий в случае смены состояния в сети DNCP или передачи keep-alive (если сообщения используются).

Реализации TLS (на момент написания документа) представляются более зрелыми и доступными (в открытом коде), чем DTLS. Это может быть связано с долгой историей использования HTTPS.

Некоторые библиотеки, похоже, не поддерживают мультиплексирование между защищёнными и незащищёнными каналами на одном порту, поэтому может быть полезно задать разные порты.

Приложение C. Пример профиля

Здесь рассматривается придуманный специально для этого документа профиль DNCP SHSP, предназначенный для домашней автоматизации. Сам протокол применяется для создания хранилища значений ключей, публикуемых каждым узлом и доступных всем другим узлам для распределенного мониторинга и управления домашней инфраструктурой. Протокол определяет 1 дополнительный тип TLV - key=value с одним назначением для публикации.

- Индивидуальный транспорт. IPv6 TCP на порту EXAMPLE-P1, поскольку в данных key=value используются лишь абсолютные метки времени и решение предназначено в основном для узлов Linux, которые поддерживают оба протокола. Соединения за пределами локального канала не поддерживаются, а не относящиеся к локальному каналу адреса игнорируются для предотвращения раскрытия протокола вовне.
- Групповой транспорт. IPv6 UDP на порту EXAMPLE-P2 для группового адреса ff02:EXAMPLE на локальном канале. Предполагается, что хотя бы один домашний узел на канал обеспечит обнаружение узлов без привязки к другой инфраструктуре.
- Защиты нет. Профиль следует применять лишь на доверенных каналах (беспроводные WPA2-x, физически защищённые кабели).
- Дополнительных TLV для игнорирования нет. Защита DNCP не задана и новые TLV сверх данных узла не определены.
- Размер идентификатора узла (DNCP_NODE_IDENTIFIER_LENGTH) - 32 бита (генерируются случайно).
- Обслуживание конфликтов между идентификаторами узлов путём выбора нового случайного значения.
- Параметры Trickle - Imin = 200 мсек, Imax = 7, k = 1. Это означает хотя бы один групповой пакет на канал каждые 25 секунд в стабильном состоянии ($0{,}2 * 2^7$).
- Хэш-функция H(x) и размер. SHA-256 с использованием 128 битов. Функция достаточно быстрая, а 128 битов должно быть достаточно для предотвращения конфликтов (скорей всего, достаточно будет 64 битов).
- Встроенных keep-alive (параграф 6.1) нет. Применяются TCP keep-alive. На практике TCP keep-alive встречается редко, поскольку изменение состояния сети вызывает передачу пакетов по индивидуальным соединениям, а отказы при достаточно большом числе повторов обусловлены отбрасыванием задолго до истечения интервала keep-alive.
- Нет поддержки оптимизации для групповых каналов с высокой плотностью (параграф 6.2). SHSP - простой протокол для нескольких узлов (во всей сети, а не на одном канале), поэтому такая поддержка не нужна.

Благодарности

Спасибо Ole Troan, Pierre Pfister, Mark Baugher, Mark Townsley, Juliusz Chroboczek, Jiazi Yi, Mikael Abrahamsson, Brian Carpenter, Thomas Clausen, DENG Hui, Margaret Cullen за вклад в подготовку документа.

Спасибо Kaiwen Jin и Xavier Bonnetain за их исследования, связанные с документом.

Адреса авторов

Markus Stenberg
Independent
Helsinki 00930
Finland
Email: markus.stenberg@iki.fi

Steven Barth
Independent
Halle 06114
Germany
Email: cyrus@openwrt.org

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru