Internet Engineering Task Force (IETF)
Request for Comments: 8966
Obsoletes: 6126, 7557
Category: Standards Track
ISSN: 2070-1721

J. Chroboczek
IRIF, University of Paris-Diderot
D. Schinazi
Google LLC

January 2021

The Babel Routing Protocol

Протокол маршрутизации Babel

Аннотация

Babel представляет собой протокол маршрутизации на основе вектора удаленности (distance-vector) с предотвращением петель, который устойчив к отказам и эффективен как в кабельных, так и в беспроводных сетях. Документ описывает протокол маршрутизации Babel и отменяет RFC 6126 и RFC 7557.

Статус документа

Документ относится к категории 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 7841.

Информация о текущем статусе документа, найденных ошибках и способах обратной связи доступна по ссылке https://www.rfc-editor.org/info/rfc8966.

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 маршрутизации Babel на основе вектора удаленности с предотвращением петель предназначен для отказоустойчивой и эффективной маршрутизации для сетей, использующих префиксы и плоскую маршрутизацию (многосвязные сети - mesh), как в сравнительно стабильных проводных системах, так и в очень динамичных беспроводных сетях. Этот документ описывает протокол маршрутизации Babel и отменяет [RFC6126] и [RFC7557].

1.1. Свойства

Основным свойством, которое делает протокол Babel подходящим для нестабильных сетей, является то, что он, в отличие от простых протоколов на основе вектора удаленности [RIP], строго ограничивает частоту и продолжительность аномалий маршрутизации, таких как петли и «черные дыры» в процессе схождения маршрутов. Даже после обнаружения перемещения элемента (mobility event) сеть Babel обычно остается свободной от петель. После такого события Babel быстро пересчитывает конфигурацию, которая не содержит петель и сохраняет связность, но не обязательно является оптимальной. Эта операция во многих случаях даже не требует обмена пакетами. Затем Babel выполняет медленную процедуру схождения (может занимать минуты) для получения оптимальной конфигурации. Это достигается за счет использования упорядоченных маршрутов - метода, впервые примененного в маршрутизации Destination-Sequenced Distance-Vector (упорядоченные по адресатам векторы удаленности) [DSDV].

Более точно, протокол Babel имеет перечисленные ниже свойства:

- когда каждый префикс исходит лишь от одного маршрутизатора, в протоколе Babel не возникает петель;
- когда один префикс исходит от нескольких маршрутизаторов, Babel может иногда создавать временные маршрутные петли, которые исчезают за время, пропорциональное диаметру петли, и больше никогда (до произвольного момента сбора мусора (garbage-collection или GC)) вовлеченные маршрутизаторы не будут попадать в петлю для того же префикса;
- в предположении ограниченной частоты потери пакетов, любые черные дыры, которые могут возникать в результате перемещения элемента, будут устраняться в срок, пропорциональный диаметру сети.

Babel обеспечивает оценку качества каналов и поддерживает достаточно произвольную метрику. При соответствующей настройке Babel может реализовать маршрутизацию по кратчайшему пути (shortest-path) или использовать метрику, например, на основе числа теряемых пакетов.

Узлы Babel устанавливают связи между собой даже при настройке с разными параметрами. Например, мобильный узел с небольшой батареей, может использовать большие временные интервалы (сообщения hello, обновления и т. п.), нежели узел с питанием от электросети. И наоборот, узел с высоким уровнем мобильности может сократить временные интервалы. Возможность создавать такие неоднородные сети делают протокол Babel адаптируемым к работе в неуправляемых и беспроводных средах.

Babel является гибридным протоколом маршрутизации в том смысле, что он может передавать маршруты для разных протоколов сетевого уровня (IPv4 и IPv6), независимо от протокола, используемого для передачи пакетов Babel.

1.2. Ограничения

Протоколу Babel присущи два ограничения, делающие его непригодным в некоторых средах. Во-первых, протокол Babel основан на периодическом обновлении таблиц маршрутизации вместо использования надежного транспорта, что ведет к росту служебного трафика в больших стабильных сетях по сравнению с протоколами, которые передают обновления лишь при наличии изменений. Для таких сетей больше подходят такие протоколы как OSPF [OSPF], IS-IS [IS-IS] или EIGRP[1] [EIGRP].

Во-вторых, если не реализован второй алгоритм, описанный в параграфе 3.5.4, Babel устанавливает время удержания при удалении префикса. Хотя это время не применяется к удаляемому префиксу и поэтому не препятствует быстрому схождению после восстановления префикса, оно применимо для более коротких префиксов, которые охватывает удаленный. Это может сделать реализации Babel, не поддерживающие необязательный алгоритм, описанный в параграфе 3.5.4, не подходящими для сетей с автоматическим агрегированием префиксов.

1.3. Уровни требований

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **не рекомендуется** (NOT RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе должны интерпретироваться в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они набраны заглавными буквами, как показано здесь.

2. Концептуальное описание протокола

Babel является протоколом на основе вектора удаленности с предотвращением петель. Он основан на алгоритме Беллмана-Форда (Bellman-Ford), как известный протокол RIP [RIP], но включает множество усовершенствований, предотвращающих возникновение петель или быстро устраняющих петли без их повторного появления.

Концептуально алгоритма Bellman-Ford выполняется параллельно для каждого источника маршрутной информации (получателя трафика данных). Далее источник обозначается S с напоминанием, что алгоритм выполняется параллельно для всех источников.

2.1. Стоимость, метрика, соседство

Для каждой пары смежных узлов A и B протокол Babel рассчитывает абстрактное значение, называемое стоимостью канала от A к B и обозначаемое C(A, B). Для данного маршрута между любыми двумя (не обязательно смежными) узлами метрикой маршрута будет сумма стоимости всех каналов на пути. Цель алгоритма маршрутизации заключается в расчете для каждого S дерева маршрутов в S с минимальной метрикой.

Стоимость и метрика не обязаны быть целыми числами. В общем случае это могут быть любые числовые значения, удовлетворяющие двум достаточно общим условиям параграфа 3.5.2.

Узел Babel периодически передает сообщения Hello всем своим соседям, а также периодически передает сообщения IHU (I Heard You - я вас слышу) каждому соседу, от которого недавно получено сообщение Hello. На основе информации из сообщений Hello и IHU от соседа B узел A рассчитывает стоимость C(A, B) для канала от A до B.

2.2. Алгоритм Беллмана-Форда

Каждый узел A поддерживает две части данных - оценку расстояния до S - D(A) и следующий маршрутизатор в направлении S - NH(A). Исходно D(S) = 0, D(A) бесконечная, а NH(A) не определен.

Периодически каждый узел B передает всем своим соседям обновление маршрутов в сообщении, содержащем D(B). Когда сосед A узла B получает обновление маршрутов, он проверяет, выбран ли B его следующим интервалом. Если это так, в качестве NH(A) указывается B, а для D(A) устанавливается C(A, B) + D(B). В противном случае A сравнивает C(A, B) + D(B) с текущим значением D(A). Если это значение меньше, полученное обновление анонсирует лучший маршрут по сравнению с выбранным и в качестве NH(A) устанавливается B, а D(A) получает значение C(A, B) + D(B).

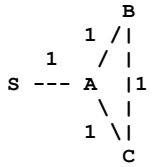
Возможен ряд усовершенствований этого алгоритма и они применяются в протоколе Babel. В частности скорость схождения может быть увеличена за счет отправки в дополнение к обычным плановым обновлениям не

[1] Enhanced Interior Gateway Routing Protocol - расширенный протокол внутренней маршрутизации.

запланированных «триггерных обновлений» при обнаружении существенных изменений в топологии. Кроме того, узел может поддерживать множество дополнительных маршрутов, которые были анонсированы соседями, отличными от выбранного им соседа, и которые могут применяться сразу же, если произойдет отказ выбранного соседа.

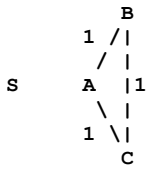
2.3. Временные петли в Bellman-Ford

Хорошо известно, что естественное применение алгоритма Bellman-Ford к распределенной маршрутизации может вызывать временные петли после изменения топологии. Рассмотрим пример, показанный на рисунке.



После схождения будет D(B) = D(C) = 2 и NH(B) = NH(C) = A.

Предположим, что на канале между S возник отказ, как показано на рисунке.



При обнаружении этого отказа узел A сменит следующий интервал на B (который продолжает анонсировать маршрут к S с метрикой 2), анонсирует метрику 3, а затем - новый маршрут с метрикой 3. Этот процесс смены выбранных соседей и увеличения их метрики продолжается, пока метрика не станет «бесконечной» (значение превысит все другие метрики, которые протокол маршрутизации способен передавать).

2.4. Условия осуществимости

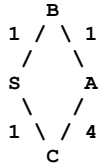
Алгоритм Bellman-Ford очень устойчив к отказам и его свойства сходимости сохраняются при задержке маршрутизаторами получения маршрутов или отбрасывании некоторых обновлений. Маршрутизаторы Babel отбрасывают полученные анонсы маршрутов, пока они не уверены, что восприятие маршрута не создаст петли. Более формально, определяется условие, когда анонсы маршрутов, известные как «условие осуществимости» (feasibility condition), гарантируют отсутствие маршрутных петель, где все маршрутизаторы игнорируют обновления, не соответствующие условию осуществимости. Фактически, это возвращает алгоритм Bellman-Ford в семейство алгоритмов маршрутизации, параметризуемых условием выполнимости.

Возможно много разных условий выполнимости. Например, BGP можно представить как протокол на основе вектора удаленности с (достаточно радикальным) условием осуществимости - обновления воспринимаются лишь при условии, что номер автономной системы принимающего узла не включен в атрибут AS_PATH этого обновления (отметим, что условие выполнимости в BGP не гарантирует отсутствие временных микропетель в процессе схождения).

Другой пример условия выполнимости применяется в протоколе маршрутизации DSDV [DSDV], а также в протоколе AODV[1] [RFC3561] и основан на наблюдении, что маршрутные петли могут возникать лишь после переключения маршрутизатора на маршрут, метрика которого больше выбранного ранее маршрута. Поэтому можно считать маршрута выполнимым, когда его метрика на локальном узле будет не больше метрики выбранного с маршрута, т. е. анонс с метрикой D(B) воспринимается узлом A, когда C(A, B) + D(B) <= D(A). Если все маршрутизаторы следуют этому ограничению, метрика на каждом маршрутизаторе не возрастает и всегда сохраняется следующий инвариант: если A выбрал B как следующий интервал, то D(B) < D(A), что означает отсутствие петель в графе пересылки.

Babel использует более тонкое условие выполнимости, выведенное из EIGRP [DUAL]. Для маршрутизатора A, определяется дистанция выполнимости (достижимости) A FD(A), как меньшая из метрик, анонсируемая A для S любому из своих соседей. Обновление переданное соседом B считается выполнимым, если анонсируемая узлом B метрика D(B) строго меньше дистанции достижимости A, т. е. D(B) < FD(A).

Легко видеть, что это условие не более ограничительно по сравнению с DSDV. Предположим, что узел A выполняет условие достижимости DSDV. Тогда D(A) не возрастает и в любой момент D(A) <= FD(A). Далее предположим, что A получает соответствующее условию DSDV обновление с анонсом метрики D(B). Поскольку обновление соответствует DSDV, C(A, B) + D(B) <= D(A), следовательно D(B) < D(A), а, поскольку D(A) <= FD(A), то и D(B) < FD(A). Чтобы увидеть, что это меньшее ограничение, рассмотрим приведенный ниже рисунок, где A выбрал маршрут через B и D(A) = FD(A) = 2. Поскольку D(C) = 1 < FD(A), маршрут через C достижим для A, хотя его метрика C(A, C) + D(C) = 5 больше чем для выбранного в данный момент маршрута.

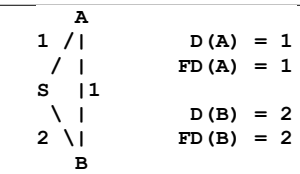


Чтобы показать, что условие выполнимости сохраняет гарантию отсутствия петель, напомним, что с тот момент, когда A воспринимает обновление от B, анонсируемая B метрика D(B) не меньше FD(B), поскольку она меньше FD(A), в этот момент FD(B) < FD(A). Поскольку это свойство сохраняется, когда A передает обновления и выбирает другой следующий интервал, это будет верно всегда и гарантирует отсутствие петель в графе пересылки.

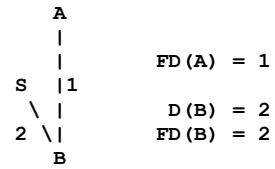
2.5. Решение проблемы истощения - нумерованные маршруты

Обычно определенное выше условие осуществимости вызывает истощение (starvation), когда у маршрутизатора не остается доступных маршрутов. Рассмотрим показанную на рисунке ситуацию, где A и B выбрали прямой маршрут к S.

[1] Ad hoc On-Demand Distance Vector.



Предположим, что канал между A и S оказался разорван.

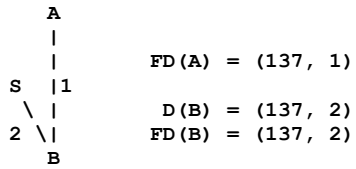


Единственный маршрут от A до S, проходящий через B, не является осуществимым и страдает от ложного истощения. В этот момент все субдерево, страдающее от истощения, должно быть сброшено, что, по сути, и делает EIGRP при выполнении глобальной синхронизации всех маршрутов в истощенном субдереве («активная» фаза EIGRP).

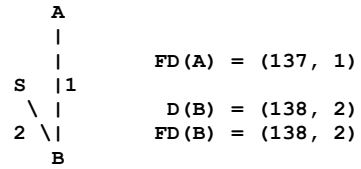
Babel менее резко реагирует на истощение, используя нумерованные маршруты, введенные в DSDV и адаптированные AODV. В дополнение к метрике каждый маршрут имеет порядковый номер, который является неубывающим целым числом, распространяемым через сеть без изменения и инкрементируемым лишь источником. Пара (s, m), где s - порядковый номер, а m - метрика, называется дистанцией (расстоянием).

Полученное обновление выполнимо если оно новее дистанции достижимости, поддерживаемой принимающим узлом или столь же свежее и его метрика строго меньше. Более формально, если FD(A) = (s, m), обновление с дистанцией (s', m') выполнимо, если s' > s или s = s' и m' < m.

Предположим, что S имеет порядковый номер 137. Тогда приведенный выше рисунок примет вид



После того, как S увеличит свой порядковый номер и новый номер будет распространен узлу B, мы получим



И маршрут через B становится выполнимым.

Отметим, что порядковые номера используются для определения достижимости, но они не применяются при выборе маршрутов. Узел игнорирует порядковые номера в процессе определения лучшего маршрута к данному адресату (параграф 3.6). Иное поведение вызвало бы осцилляции маршрутов при распространении порядкового номера через сеть и могло бы даже создать постоянные «черные дыры» с экзотической метрикой.

2.6. Запросы

В DSDV порядковый номер источника увеличивается периодически. Маршрут становится выполнимым после того, как источник увеличит свой порядковый номер и распространит новое значение через сеть, что в общем случае может потребовать значительного времени.

Babel использует другой подход. Когда узел обнаруживает, что он страдает от потенциально ложного истощения, этот узел передает явный запрос источнику для получения нового порядкового номера. Этот запрос пересылается поэтапно (hop by hop) к источнику, независимо от условия выполнимости. При получении запроса источник увеличивает свой порядковый номер и в широковещательном режиме передает обновление, которое пересылается запросившему узлу.

Отметим, что после изменения топологии сети не все такие запросы в общем случае будут попадать к источнику, поскольку некоторые будут отправлены через каналы, которые в данный момент разорваны. Однако, если сеть все еще подключена, хотя бы один из страдающих от ложного истощения узлов имеет (невыполнимый) маршрут к источнику. Поэтому при отсутствии потери пакетов хотя бы один запрос попадет к источнику. Для компенсации потери пакетов запросы могут повторяться несколько раз.

Поскольку запросы пересылаются независимо от условий выполнимости, они могут в общем случае попасть в петлю пересылки, которой можно избежать, детектируя дубликаты пересылаемых запросов.

2.7. Префикс от нескольких маршрутизаторов

В приведенном выше обсуждении предполагается, что каждый префикс исходит от одного маршрутизатора. Однако в реальных сетях префикс зачастую исходит от множества маршрутизаторов, например, принятый по умолчанию маршрут будет исходить от всех граничных маршрутизаторов домена маршрутизации.

Поскольку синхронизация порядковых номеров между маршрутизаторами проблематична, Babel считает маршруты к одному префиксу разными сущностями, когда они приходят от разных маршрутизаторов. Каждый анонс маршрутов содержит router-id создавшего его маршрутизатора, а дистанции выполнимости поддерживаются не на уровне префикса, а на уровне источника, который представляет собой пару (router-id - префикс). Фактически Babel гарантирует отсутствие петель в графе пересылки для каждого источника. Поскольку объединение множества ациклических графов не всегда является ациклическим, Babel не гарантирует отсутствия петель в случае происхождения префикса от

нескольких маршрутизаторов, но любые петли устраняются на время, пропорциональное диаметру петли, когда обновление «обойдет петлю».

Рассмотрим показанную ниже топологию, где узел A выбрал маршрут по умолчанию через S, а B - через S'.

```
                    1     1     1
::/0 -- S --- A --- B --- S' -- ::/0
```

Предположим, что одновременно отказали оба принятых по умолчанию маршрута. В этом случае ничто не мешает узлу A переключиться на B, а B - одновременно переключиться на A. Однако, как только A успешно анонсирует новый маршрут узлу B, маршрут через A станет невыполнимым для B. И наоборот, как только B анонсирует свой маршрут через A, маршрут через B станет невыполнимым для A.

Фактически петля исчезает не позднее прохождения маршрутной информации через нее. Поскольку этот процесс может быть задержан потерей пакетов, Babel прилагает некоторые усилия для обеспечения гарантированной доставки обновлений после смены router-id (параграф 3.7.2).

Кроме того, после анонсирования маршрутизаторами двух маршрутов оба источника будут в их таблицах источников, что предотвратит их повторное попадание в маршрутную петлю, включающую маршруты от S и S' (вплоть до времени источника GC, для которого, если позволяет память, можно устанавливать произвольно большое значение).

2.8. Перекрывающиеся префиксы

В предыдущем обсуждении предполагалось, что префиксы разъединены (не перекрываются) как в случае плоской (многосвязной - mesh) маршрутизации. Однако на практике префиксы могут перекрываться, например, принятый по умолчанию маршрут перекрывается со всеми маршрутами в сети.

После отказа маршрута в общем случае некорректно переключаться на маршрут, включающий отказавший. Рассмотрим показанный на рисунке пример.

```
                      1     1
::/0 -- A --- B --- C
```

Предположим, что узел C отказал. Если B пересылает пакеты для C по принятому по умолчанию маршруту, возникнет петля, которая будет сохраняться, пока A не узнает об отсутствии у B прямого маршрута к C. Узел B избегает этой ловушки, устанавливая «невыполнимый» маршрут после вызванного отказом пропадания маршрута. Этот маршрут поддерживается, пока не будет гарантии, что утраченный маршрут отменен всеми соседями B (параграф 3.5.4).

3. Работа протокола

Каждый узел Babel (speaker) имеет свой идентификатор (router-id) - произвольную стороку размером 8 октетов, которая предполагается уникальной в домене маршрутизации. Идентификаторы могут выделяться, например, случайным образом или выводиться из адресов канального уровня. Кодирование протокола становиться чуть более компактным при выделении router-id в стиле присвоения идентификаторов узлов IPv6 (см. описание флага R в параграфе 4.6.9.).

3.1. Передача и прием сообщений

Пакеты протокола Babel передаются в теле дейтаграмм UDP (раздел 4). Каждый пакет Babel может содержать множество (возможно пустое) TLV, большинство из которых могут включать суб-TLV.

Трафик управления Babel может передаваться по протоколу IPv6 или IPv4, а префиксы любого семейства адресов могут анонсироваться через любой протокол. Таким образом, имеется по меньшей мере две модели естественного развртывания - использование исключительно IPv6 для всего трафика управления или работа двух разных экземпляров протокола, по одному для каждого семейства адресов. **Рекомендуется** применять исключительное использование IPv6 для всего управляющего трафика, поскольку одновременное применение двух протоколов удваивает трафик, предназначенный для обнаружения соседей и оценки качества каналов.

Адресом отправителя пакетов Babel всегда является индивидуальный адрес (link-local в случае IPv6). Пакеты Babel могут передаваться по стандартному (link-local) групповому или индивидуальному (link-local) адресу. При обычной работе узел Babel передает своим соседям групповые и индивидуальные пакеты.

За исключением подтверждений, все Babel TLV могут передаваться по групповому или индивидуальному адресу и семантика TLV от этого не зависит. Поэтому узлу Babel не требуется определять адрес получателя в принятом пакете для его интерпретации.

К передаваемым узлом Babel пакетам могут применяться произвольные вариации задержки (jitter). Исходящие TLV буферизуются и их **следует** передавать со случайной задержкой. Это преследует две цели - избежать синхронизации узлов Babel в сети [JITTER] и обеспечить возможность объединения множества TLV в один пакет.

Максимальная задержка TLV может зависеть от TLV. Когда описание протокола указывает срочность TLV (параграфы 3.7.2 и 3.8.1), такие TLV **должны** передаваться в течение короткого времени, называемого тайм-аутом срочности (urgent timeout), рекомендуемые значения тайм-аута приведены в Приложении B. Когда TLV регулируется тайм-аутом, заданным в предшествующем TLV, как в случае подтверждений (параграф 4.6.4), обновлений (параграф 3.7) и IHU (параграф 3.4.2), TLV **должны** передаваться достаточно быстро, чтобы уложиться в указанное время (с небольшим запасом на задержку распространения). В остальных случаях TLV **следует** отправлять в течение половины интервала Multicast Hello.

Для предотвращения отбрасывания пакетов (отправителем или получателем) **следует** вносить задержку между последовательными пакетами с одного интерфейса с учетом ограничений, указанных выше. Однако следует отметить, что задержка может препятствовать возможности агрегирования пакетов для некоторых канальных технологий (например, IEEE 802.11 [IEEE802.11]).

3.2. Структуры данных

В этом параграфе описаны структуры данных, поддерживаемые каждым узлом Babel. Это концептуальное описание и узлы Babel могут применять иные структуры данных, обеспечивающие соответствие результирующего протокола данному документу. Например, вместо поддержки одной таблицы, содержащей выбранные и резервные (fallback)

маршруты, как описано в параграфе 3.2.6, реализация может использовать две разных таблицы для выбранных и резервных маршрутов.

3.2.1. Арифметика порядковых номеров
Порядковые номера (seqno) включаются во многие структуры данных Babel и интерпретируются как целые числа с модулем 2^{16}. Используемая в этом документе арифметика порядковых номеров описана ниже.

Для данного порядкового номера s и неотрицательного целого числа n сумма s и n определяется выражением

```
s + n (по модулю 216) = (s + n) MOD 2^(16)
```

или его эквивалентом

```
s + n (по модулю 216) = (s + n) AND 65535
```

где MOD операция деления по модулю, дающая неотрицательное целое число, а AND - побитовое пересечение (И). Для двух порядковых номеров s и s' отношение s меньше s' (s < s') определяется выражением

```
s < s' (по модулю 216), когда 0 < ((s' - s) MOD 2^(16)) < 32768
```

или его эквивалентом

```
s < s' (по модулю 216), когда s /= s' и ((s' - s) AND 32768) = 0.
```

3.2.2. Порядковый номер узла
Порядковый номер узла - это 16-битовое целое число, включаемое в обновления, передаваемые для маршрутов, исходящих из этого узла.

Узлы инкрементируют свой порядковый номер (модуль 2^{16}) при получении запроса на новый номер (параграф 3.8.1.2). Узлам **не следует** инкрементировать свой порядковый номер спонтанно, поскольку увеличение seqno снижает вероятность того, что другие узлы будут иметь выполнимые альтернативные маршруты при отказе выбранных ими маршрутов.

3.2.3. Таблица интерфейсов
Таблица интерфейсов содержит список всех интерфейсов, на которых узел использует протокол Babel. Каждая запись таблицы включает порядковый номер исходящего Multicast Hello (16-битовое целое число, которое передается в каждом Multicast Hello TLV на этом интерфейсе и инкрементируется с модулем 2^{16} при передаче Multicast Hello). Отметим, что порядковый номер Multicast Hello не связан с порядковым номером узла.

С каждой записью в таблице интерфейсов связано два таймера. Таймер периодических сообщений Multicast Hello управляет плановой отправкой пакетов Multicast Hello и IHU (параграф 3.4). Таймер периодических обновлений (Update) управляет периодической отправкой маршрутных обновлений (параграф 3.7.1). Рекомендуемые значения для таймеров приведены в Приложении B.

3.2.4. Таблица соседей
Таблица соседей содержит список всех смежных интерфейсов, от которых недавно были получены пакеты Babel. Таблица индексируется по парам (интерфейс — адрес) и каждая запись содержит:

- интерфейс локального узла, через который доступен этот сосед;
- адрес интерфейса соседа;
- историю недавно принятых пакетов Multicast Hello от этого соседа (это может быть, например, последовательность из n битов с небольшим n, указывающая какие из последних n сообщений hello от данного соседа были приняты локальным узлом);
- история недавно полученных сообщений Unicast Hello от этого соседа;
- значение «стоимости передачи» из последнего пакета IHU, полученного от этого соседа, или шестнадцатеричное значение FFFF (бесконечность), если истек таймер IHU для данного соседа;
- ожидаемый порядковый номер Multicast Hello от этого соседа (целое число с модулем 2^{16});
- ожидаемый порядковый номер Unicast Hello от этого соседа (целое число с модулем 2^{16});
- порядковый номер исходящего Unicast Hello для этого соседа (целое число с модулем 2^{16}), передаваемый в каждом Unicast Hello TLV и инкрементируемый (модуль 2^{16}) при отправке Unicast Hello (отметим, что номер исходящего Unicast Hello для соседа отличается от номера исходящего Multicast Hello для интерфейса).

С каждой записью таблицы связано три таймера - Multicast Hello, задающий интервал, передаваемый в плановых Multicast Hello TLV от этого соседа, Unicast Hello с интервалом из плановых Unicast Hello TLV и IHU со значением, кратным (в несколько раз) интервалу из IHU TLV (см. Приложение B).

Отметим, что таблица соседей индексируется по адресам IP, а не router-id, поскольку смежность существует между интерфейсами, а не узлами. Поэтому два узла с несколькими интерфейсами могут иметь несколько отношений смежности, что часто возникает в беспроводных сетях, где участвуют узлы с несколькими радио-модулями.

3.2.5. Таблица источников
Таблица источников служит для записи дистанций выполнимости (feasibility distance) и индексируется триплетами (prefix, plen, router-id). Каждая запись таблицы включает:

- префикс (prefix, plen), где plen указывает размер префикса в битах, применимый к этой записи;
- router-id маршрутизатора, от которого исходит этот префикс;
- пару (seqno, metric), указывающую дистанцию выполнимости для этого источника.

С каждой записью связан таймер «сборки мусора» (garbage-collection), имеющий значение порядка минут и сбрасываемый в соответствии с параграфом 3.7.3.

3.2.6. Таблица маршрутов

Таблица маршрутов содержит известные узлу маршруты и индексируется триплетом (prefix, plen, neighbour). Каждая запись таблицы включает:

- источник (prefix, plen, router-id) для которого анонсирован этот маршрут;
- сосед neighbour (запись в таблице соседей), анонсировавший этот маршрут;
- метрика, с которой маршрут анонсирован соседом, или шестнадцатеричное значение FFFF (бесконечность) для недавно утраченного маршрута;
- порядковый номер, с которым маршрут анонсирован;
- адрес next-hop для этого маршрута;
- флаг, указывающий выбор маршрута, т. е. его использование для пересылки и анонсирование.

С каждой записью связан таймер срока действия маршрута. Инициализация и сброс таймера описаны в параграфе 3.5.3.

Отметим, что с каждым маршрутом связаны две разных пары (seqno, metric), одна из которых указывает дистанцию (протяженность) маршрута в таблице, другая - дистанцию выполнимости, которая хранится в таблице источников и используется для всех маршрутов с данным источником.

3.2.7. Таблица ожидающих запросов порядковых номеров

Таблица ожидающих запросов seqno содержит список запросов порядкового номера, которые передал локальный узел (порождены локально или пересланы), но ответы еще не были получены. Таблица индексируется триплетом (prefix, plen, router-id) и каждая запись содержит:

- prefix, plen, router-id и seqno из запроса;
- сосед, для которого пересылался запрос (при наличии);
- небольшое целое число, указывающее повторов, если запрос не был выполнен.

С каждой записбю связан таймер, управляющий повтором запроса и сроком его действия.

3.3. Подтверждения и запросы подтверждений

Узел Babel может запросить у соседа явного подтверждения приема отправленного тому пакета. Хотя использование подтверждений не обязательно, каждый узел Babel **должен** поддерживать возможность ответа на такие запросы.

Подтверждения **должны** передаваться по индивидуальному адресу, но запросы могут отправляться как на индивидуальные, так и на групповые адреса. В последнем случае отвечают все получившие запрос узлы.

Передача запросов на подтверждение определяется локальной политикой. Простейшая стратегия - никогда не запрашивать подтверждений и полагаться на периодические обновления в качестве гарантии того, что все доступные маршруты в конечном итоге распространяются по всему домену маршрутизации. Для повышения скорости схождения и снижения объема трафика управления запросы подтверждения **могут** служить для надежной отправки срочных обновлений (параграф 3.7.2) и отзывов (параграф 3.5.4), особенно при небольшом числе соседей на данном интерфейсе. Поскольку протокол Babel разработан для корректной работы при потере пакетов в ненадежных средах, отправка всех пакетов с запросом подтверждения не обязательна и **не рекомендуется**, поскольку подтверждения создают дополнительный трафик и могут вызывать дополнительный обмен ARP (Address Resolution Protocol) или ND (Neighbour Discovery).

3.4. Нахождение соседей

Нахождение соседей (neighbour acquisition) - процесс, с помощью которого узел Babel определяет набор соседей, слышимых через каждый из его интерфейсов и устанавливает двухстороннюю достижимость (доступность). В ненадежных средах обнаружение соседей дополнительно обеспечивает статистику, которая может быть полезна при расчете качества каналов.

До того как узел Babel сможет обмениваться маршрутными данными с соседом, он **должен** создать для этого соседа запись в своей таблице соседей. Когда это следует делать, определяет реализация и приемлемые стратегии включают создание записи при получении любого пакета Babel или при анализе Hello TLV. Для экономии системных ресурсов реализации **следует** отбрасывать записи, которые не используются слишком долго. Приемлемой стратегией является отбрасывание соседей по тайм-ауту или при пустой истории соответствующих сообщений Hello (см. Приложение A.2).

3.4.1. Определение обратной достижимости

Каждый узел Babel регулярно или не регулярно передает своим соседям Hello TLV для индикации свое активности. В каждом Hello TLV содержится увеличивающийся (модуль 2^{16}) порядковый номер и верхняя граница интервала отправки следующего Hello того же типа (см. ниже). Если установлен интервал 0, Hello TLV не указывает нового «обещания». При этом интервал из предыдущего Hello того же типа остается в силе для следующего Hello (если недавнее сообщение Hello нужного типа было получено в момент t0 и включало интервал i, прежнее обещание передать Hello до момента t0 + i остается в силе). Сообщения Hello считаются «плановыми», если они включают отличный от 0 интервал.

Имеется два типа Hello - Multicast Hello, использующие счетчик Hello на уровне интерфейса (Multicast Hello seqno), и Unicast Hello со счетчиком для соседа (Unicast Hello seqno). Сообщения Multicast Hello с данным seqno **должны** передаваться всем соседям на данном интерфейсе путем отправки по групповому адресу или по индивидуальным адресам каждого соседа (т. е. название Multicast Hello является не совсем точным). Unicast Hello с данным given

обычно следует передавать лишь одному соседу (по индивидуальному адресу), поскольку порядковые номера для разных соседей в общем случае не синхронизированы.

Multicast Hello, переданные по групповому адресу могут служить для обнаружения соседей. Узлу **следует** передавать периодические (плановые) Multicast Hellos, если обнаружение соседей не выполняется вне протокола Babel. Узел **может** передавать Unicast Hello или неплановые Hello любого типа по любой причине, например для снижения группового трафика или повышения надежности на каналах со слабой поддержкой групповой адресации.

Узел **может** передать плановое сообщение Hello заранее, а также **может** изменить запланированный интервал Hello. Интервал Hello **можно** сократить в любой момент, а также **можно** увеличить непосредственно перед отправкой Hello TLV, но **не следует** увеличивать в другие моменты (эквивалентно, узлу **следует** передать плановые Hello сразу после увеличения интервала Hello).

Что делать с полученными Hello TLV и какую статистику для них поддерживать, определяет локальная реализация. Обычно узлы поддерживают тот или иной тип истории недавно принятых Hello. Пример подходящего алгоритма представлен в Приложении A.1.

После приема Hello или определения его пропуска узел пересчитывает стоимость ассоциации (параграф 3.4.3) и запускает процедуру выбора маршрута (параграф 3.6).

3.4.2. Двухстороннее определение достижимости

Для организации двухсторонней доступности каждый узел периодически передает IHU TLV (я вас слышу) каждому из своих соседей. Поскольку IHU содержат явный интервал, их **можно** передавать реже, чем Hello для снижения уровня маршрутного трафика в плотных сетях. В частности, их **следует** передавать реже Hello на каналах с незначительными потерями. Хотя IHU концептуально являются индивидуальными, их **можно** передавать по групповому адресу, чтобы избежать лишних обменой ARP или Neighbour Discovery и агрегировать множество IHU в один пакет.

В дополнение к периодическим IHU узел **может** в любой момент передать неплановый пакет IHU. Узел также **может** в любой момент снизить интервал IHU и **может** увеличить этот интервал непосредственно перед отправкой IHU, но **не** следует увеличивать интервал в другие моменты (эквивалентно, узлу **следует** передавать дополнительные IHU сразу после увеличения интервала Hello).

В каждом IHU TLV содержатся две части данных - rxcost для канала (стоимость приема) с точки зрения отправителя, используемое соседом для расчета стоимости канала (параграф 3.4.3), и интервал между периодическими пакетами IHU. Принявший IHU узел устанавливает значение txcost (стоимость передачи), поддерживаемое в таблице соседей, в соответствии со значением в IHU и сбрасывает таймер IHU, связанный с данным соседом, до значения кратного (в несколько раз) интервалу, полученному в IHU (см. Приложение B). По истечении таймера IHU для соседского txcost устанавливается бесконечное значение.

После обновления соседского txcost принимающий узел пересчитывает стоимость для соседа (параграф 3.4.3) и запускает процедуру выбора маршрута (параграф 3.6).

3.4.3. Расчет стоимости

Стоимость канала к соседу (neighbourship association) рассчитывается по значениям из таблицы соседей, где хранится статистика приема Hello и значения txcost, рассчитанные из пакетов IHU.

Для каждого соседа узел Babel рассчитывает значения, называемое rxcost. Оно обычно выводится из статистики Hello, которая может комбинироваться с другими данными, такими как статистика канального уровня. Значение rxcost передается соседу в каждом IHU.

Поскольку узлы не обязательно передают периодические Unicast Hello, но обычно передают Multicast Hello (параграф 3.4.1), узлу **следует** использовать алгоритм, который дает конечное значение rxcost, когда имеются лишь Multicast Hello, если не требуется взаимодействие лишь с узлами, которые передают только Multicast Hello.

Использование txcost и rxcost при расчете стоимости канала определяется локальными правилами. Для корректности работы Babel **должны** выполняться приведенные ниже условия:

- значение стоимости всегда положительно;
- если недавно не было принято Hello TLV любого типа, стоимость будет бесконечной;
- если txcost имеет бесконечное значение, стоимость будет бесконечной.

Рекомендуемая стратегия расчета стоимости канал приведена в Приложении A.2.

3.5. Поддержка таблицы маршрутизации

Концептуально обновления Babel задает квинтет (prefix, plen, router-id, seqno, metric), где (prefix, plen) - префикс для маршрута, router-id - идентификатор маршрутизатора, создавшего обновление, seqno - не уменьшающееся целое число (модуль 2^{16}) - порядковый номер маршрутизатора-источника, metric - анонсируемая метрика.

Перед восприятием обновления оно проверяется на предмет выполнимости (параграф 3.5.1), что обеспечивает предотвращение маршрутных петель. Если условие осуществимости не выполняется, обновление игнорируется или маршрут не выбирается, как описано в параграфе 3.5.3. Если условие выполнимости обеспечено, обновление не может создать маршрутную петлю.

3.5.1. Условие осуществимости

Условие выполнимости применяется ко всем полученным обновлениям. Сравнивается метрика в обновлении с метрикой, переданной ранее в обновлениях принявшего данное обновление узла. Обновления, которые не прошли проверку и, следовательно, имеют достаточно большую метрику, способную вызвать петлю, игнорируются или полученный в результате маршрут не выбирается.

Дистанция выполнимости задается парой (seqno, metric), где seqno - целое число с модулем 2^{16}, а метрика - положительное целое число. Дистанции сравниваются лексикографически с инвертированием номера - дистанция (seqno, metric) считается строго лучше (seqno', metric'), что записывается в форме

```
(seqno, metric) < (seqno', metric')
```

когда

```
seqno > seqno' или (seqno = seqno' и metric < metric')
```

где порядковые номера сравниваются по модулю 2^{16}.

С данным источником (prefix, plen, router-id) дистанция выполнимости узла для этого источника будет минимальной (в соответствии с определенным выше порядком) среди дистанций из всех конечных обновлений, когда-либо переданных этим узлом для префикса (prefix, plen) и данного router-id. Дистанции выполнимости хранятся в таблице источников, как описано в параграфе 3.7.3.

Принятое обновление выполнимо, если оно является отзывом (метрика имеет шестнадцатеричное значение FFFF) или анонсируемая дистанция строго лучше (в соответствии с приведенным выше определением) дистанции выполнимости для соответствующего источника. Точнее говоря, анонс маршрута в (prefix, plen, router-id, seqno, metric) выполним, при выполнении одного из приведенных ниже условий:

- метрика конечна;
- нет записи с таблице источников с индексом (prefix, plen, router-id);
- в таблице источников имеется запись (prefix, plen, router-id, seqno', metric') и выполняется одно из условий:
  - seqno' < seqno;
  - seqno = seqno' и metric < metric'.

Отметим, что условие выполнимости учитывает анонсированную соседом метрику, а не метрику маршрута, поэтому флуктуации стоимости для соседа не могут сделать выбранный маршрут невыполнимым. Также следует отметить, что отзывы (обновления с бесконечной метрикой) выполнимы всегда, поскольку не могут вызвать петлю.

3.5.2. Расчет метрики

Метрика маршрута рассчитывается из метрики, анонсированной соседом, и стоимости канала к соседу. Подобно расчету стоимости, вычисление метрики определяется локальной политикой. Используемая Babel функция расчета метрики M(c, m) по локально рассчитанной стоимости канала c и анонсированной соседом метрике m **должна** лишь соответствовать двум условиям:

- если стоимость c бесконечная, M(c, m) также бесконечна;
- функция M является строго монотонной, M(c, m) > m.

Кроме того, для метрики **следует** выполнять условие

```
если m <= m', то M(c, m) <= M(c, m')
```

Хотя строгая монотонность функции важна для целостности сети (иначе могут возникать сохраняющиеся петли), второе условие (дистрибутивность слева) не так важно и при его нарушении Babel все равно обеспечивает схождение беспетлевой конфигурации, но может не обеспечиваться глобальной оптимизации (фактически такой оптимизации может просто не существовать).

Приведенные выше условия легко выполняются при использовании аддитивной метрики, т. е. при определении M(c, m) = c + m. Поскольку аддитивная метрика полезна для многих вариантов расчета стоимости, **рекомендуется** применять ее по умолчанию. В Приложении C описаны методы, позволяющие менять значения той или иной метрики без риска возникновения маршрутных петель.

3.5.3. Получение маршрутов

Когда узел Babel получает обновление (prefix, plen, router-id, seqno, metric) от соседа neigh, он проверяет наличие в таблице маршрутов записи с индексом (prefix, plen, neigh). Если такой записи нет, выполняются следующие операции:

- если обновление невыполнимо, его **можно** игнорировать;
- если метрика бесконечна (обновление служит отзывом неизвестного маршрута), обновление игнорируется;
- в остальных случаях в таблицу маршрутов включается новая запись с индексом (prefix, plen, neigh), источником (prefix, plen, router-id), порядковым номером seqno и анонсируемой метрикой из обновления.

Если запись имеется в таблице, выполняются следующие операции:

- если запись выбрана, обновление невыполнимо и router-id в обновлении совпадает с идентификатором маршрутизатора в записи, эту запись **можно** игнорировать;
- в остальных случаях порядковый номер записи, анонсируемая метрика, метрика и router-id одновляются, даже когда для таймера срока действия маршрута установлено значение кратное (в несколько раз) интервалу, включенному в обновление (см. Приложение B). Если обновление невыполнимо, запись (невыполнимая) **должна** незамедлительно стать не выбранной. Если обновление вызвало смену router-id в записи, **должно** быть передано своевременное уведомление (возможно отзыв), как указано в параграфе 3.7.2.

Отметим, что в таблице могут присутствовать невыполнимые маршруты, ставшие таковыми в результате получения невыполнимого обновления или флуктуаций метрики. Такие маршруты никогда не выбираются, поскольку для них нет информации об отсутствии петель. Однако, если выс выполнимые маршруты перестанут быть таковыми, невыполнимые маршруты можно сделать выполнимыми и возможными для выбора путем отправки запросов по ним (параграф 3.8.2).

При срабатывании таймера срока действия маршрута поведение за висит от конечности метрики маршрута. Если метрика конечна, для нее устанавливается бесконечное значение и таймер сбрасывается. Если же метрика уже является бесконечной, маршрут удаляется из таблицы.

После обновления таблицы маршрутов запускается процедура выбора (параграф 3.6).

3.5.4. Время удержания

Когда префикс P отзывается (поскольку все маршруты стали невыполнимыми или имеют бесконечную метрику в результате завершения срока действия или по иной причине) и доступен более короткий префикс P', охватывающий P, в общем случае P' не может применяться для маршрутизации пакетов в P без риска появления петли (параграф 2.8).

Для предотвращения этой проблемы при отзыве префикса P поддерживается запись таблицы маршрутов с бесконечной метрикой, как описано в параграфе 3.5.3. Пока запись поддерживается, пакеты с адресом из P **недопустимо** пересылать по маршруту с более коротким префиксом. Запись удаляется из таблицы, как только будут получено обновление для P с конечной метрикой и выбран соответствующий маршрут. Если такого обновления не ожидается, бесконечную метрику **следует** поддерживать по меньшей мере до того момента, пока не будет гарантии, что ни один сосед не выбрал текущий узел в качестве следующего интервала (next-hop) для префикса P. Это можно сделать любым из двух способов:

- дождаться завершения срока действия маршрута по таймеру (параграф 3.5.3);
- передать отзыв с запросом подтверждения (параграф 3.3) каждому доступному соседу, который явно не отозвал префикс P, и дождаться всех подтверждений.

Первый вариант проще и гарантирует, что в этот момент для всех маршрутов к префиксу P, указывающих на текущий узел, срок действия закончился. Однако, поскольку срок действия может быть достаточно большим (несколько минут), такое решение препятствует автоматическому агрегированию, создавая ложные «черные дыры» для агрегированных маршрутов. **Рекомендуется** применять второй вариант, поскольку он значительно сокращает время недоступности префикса при наличии агрегированных маршрутов.

3.6. Выбор маршрута

В процессе выбора один из маршрутов для данного префикса назначается для использования при пересылке пакетов и реанонсирования соседям узла.

Протокол Babel поддерживает гибкие правила выбора маршрутов, при этом **должны** обеспечиваться два свойства:

- маршруты с бесконечной метрикой (отозванные маршруты) никогда не выбираются;
- невыполнимые маршруты никогда на выбираются.

Узлы Babel с разными правилами выбора могут взаимодействовать и не будут создавать петель при выполнении этих условий.

При выборе маршрутов **недопустим** учет порядковых номеров и **недопустимо** отдавать предпочтение маршруту с большим (более новым) seqno. Отказ от этого будет приводить к осцилляциям маршрутов при распространении нового номера по сети и может вызывать сохраняющиеся «черные дыры», если используемая метрика не дистрибутивна слева (параграф 3.5.2).

Очевидной стратегией служит выбор (для каждого получателя) выполнимого маршрута с минимальной метрикой. При стабильной метрике такой подход гарантирует сходимость дерева кратчайших путей (в предположении дистрибутивности метрики слева, см. параграф 3.5.2). Однако имеются две причины, по которым эта стратегия может вести к нестабильности при постоянно меняющейся метрике. Во-первых, если два параллельных маршрута осциллируют вокруг одного значения метрики, стратегия выбора меньшей метрики ведет к постоянным переключениям с одного маршрута на другой. Во-вторых, выбор маршрута увеличвает нагрузку для него, что может вызвать потери пакетов, а они, в свою очередь, увеличат метрику и возникшая петля обратной связи может вызвать стойкие переключения маршрутов.

Для предотвращения такой нестабильности в стратегию выбора маршрута **следует** включать гистерезис, т. е. учитывать историю маршрута и менять выбранный маршрут на другой, если тот был лучше в течение некоторого времени. **Рекомендуемый** алгоритм гистерезиса представлен в Приложении A.3.

После выбора маршрута передаются триггерные уведомления (параграф 3.7.2) и запросы (параграф 3.8.2).

3.7. Передача обновлений

Узел Babel анонсирует соседям свой набор выбранных маршрутов. Обычно это выполняется путем отправки одного или множества групповых пакетов с Update TLV через все подключенные интерфейсы. Однако для канальных технологий, где групповая передача менее эффективна по сравнению с индивидуальной, **можно** передавать множество копий индивидуальных пакетов, особенно при небольшом числе соседей.

Кроме того, для надежного и своевременного устранения «черных дыр» узел Babel может передавать отзывы (обновления с бесконечной метрикой) для недавно отозванных префиксов. Если обновление предназначено для маршрута, внесенного в домен Babel локальным узлом (например, оно содержит адрес локального интерфейса, префикс подключенной напрямую сети или префикс из другого протокола маршрутизации), в качестве router-id указывается идентификатор локального маршрутизатора, для метрики указывается произвольное конечное значение (обычно 0) и включается порядковый номер локального маршрутизатора.

Если обновление предназначено для маршрута от другого узла Babel, router-id и порядковый номер берутся из записи таблицы маршрутов, а метрика рассчитывается в соответствии с параграфом 3.5.2.

3.7.1. Периодические обновления

Каждый узел Babel **должен** анонсировать каждый из выбранных маршрутов на каждом интерфейсе хотя бы один раз в каждый интервал Update. Поскольку в Babel не возникает проблем от маршрутных петель (не требуется «счет до бесконечности») и протокол основан на триггерных обновлениях (параграф 3.7.2), полная передача маршрутов происходит достаточно редко (интервалы предложены в Приложении B).

3.7.2. Триггерные обновления

В дополнение к периодическим обновлениям узлы Babel передают неплановые (триггерные) обновления для информирования своих соседей о существенных изменениях в топологии сети.

Изменение router-id в выбранном маршруте для данного префикса может служить индикацией образующейся петли, поэтому при каждой смене router-id для данного адресата узел **должен** передать обновление как срочный блок TLV (см. параграф 3.1). Кроме того, **следует** предпринять разумные попытки обеспечить получение этого обновления всеми соседями.

Для этого есть две стратегии. Если число соседей невелико, разумно передать обновление с запросом подтверждения, которое будет повторяться несколько раз, пока все соседи не подтвердят прием пакета. При большом числе соседей такой запрос подтверждений может вызвать ощутимый трафик и более предпочтительным вариантом может оказаться простое повторение обновления несколько раз (скажем, 3 для беспроводных сетей или 2 для кабельных). **Недопустимо** передавать больше 5 копий и их **следует** разделять небольшим интервалом (параграф 3.1).

Отзыв маршрута вызывает меньше беспокойства - если он придет не всем соседям, может возникнуть «черная дыра», которая, в отличие от петли, не представляет опасности для целостности сети. При отзыве маршрута узлу **следует** передать триггерное обновление, а также **следует** предпринять разумные попытки доставить его всем соседям.

Узел **может** передавать триггерное уведомление при существенном изменении метрики для данного префикса в результате получения обновления, изменения стоимости канала или выбора другого следующего узла (next-hop). Узлу **не следует** передавать триггерные обновления по иным причинам, таким как незначительные флуктуации метрики маршрутов, смене next-hop без существенного изменения метрики маршрута или распространении нового порядкового номера (за исключением выполнения запроса, как указано в параграфе 3.8).

3.7.3. Поддержка дистанций достижимости

Перед отправкой обновления (prefix, plen, router-id, seqno, metric) с конечной метрикой (т. е. не отзыва) узел Babel обновляет дистанцию выполнимости в таблице источников, как описано ниже.

Если в таблице источников нет записи с индексом (prefix, plen, router-id), создается запись со значениями(prefix, plen, router-id, seqno, metric). Если имеется запись (prefix, plen, router-id, seqno', metric') она обновляется:

- если seqno > seqno', устанавливается seqno' := seqno, metric' := metric;
- если seqno = seqno' и metric' > metric, устанавливается metric' := metric;
- в остальных случаях ничего не меняется.

Для записи сбрасывается таймер сборки мусора. Отметим, что дистанция выполнимости не обновляется и таймер сборки мусора не сбрасывается при отправке отзыва маршрута (обновления с бесконечной метрикой).

По завершении отсчета таймера сборки мусора запись удаляется из таблицы источников.

3.7.4. Расщепление горизонта

При работе в транзитивной топологии с симметричными каналами (например, соединения «точка-точка» или проводная ЛВС, такая как Ethernet) узлу Babel **следует** использовать оптимизацию, называемую «расщеплением горизонта» (split horizon). При использовании ее на данном интерфейсе маршрутные обновления для префикса P не передаются на интерфейс, через который был получен выбранных маршрут в направлении префикса P.

Расщепление горизонта **не следует** применять на интерфейсе, пока нет уверенности в симметрии и транзитивности. В частности, этот метод не применим к децентрализованным беспроводным канальным технологиям (например, IEEE 802.11 в режиме ad hoc), когда маршрутные обновления передаются по групповым адресам.

3.8. Явные запросы

При обычной работе таблица маршрутов узла заполняется из плановых и триггерных обновлений от соседей. Однако в некоторых обстоятельствах узел может передавать явные запросы для ресинхронизации с источником после перемещения (mobility event) или для предотвращения ненужного завершения срока действия маршрута.

Протокол Babel поддерживает два типа явных запросов - запрос маршрута, который возвращает обновление для данного префикса, и запрос порядкового номера для обновления номера у данного префикса. Первый тип запросов не пересылается, запросы второго типа пересылаются, если получатель не может выполнить запрос.

3.8.1. Обработка запросов

При получении запроса узел пересылает его или возвращает запрошенное обновление, как описано ниже. Обновления передаются по групповому адресу для принявшего запрос интерфейса или по индивидуальному адресу отправителя. Запроса. Детали поведения различаются для запроса маршрутов и порядковых номеров.

3.8.1.1. Запросы маршрутов

Когда узел получает запрос маршрута для данного префикса, он ищет в своей таблице выбранный маршрут с совпадающим в точности префиксом. При наличии такого маршрута узел **должен** передать обновление (групповое или индивидуальное), а при отсутствии **должен** отправить отзыв для данного префикса.

При получении шаблонного запроса маршрута узлу **следует** возвращать полный дамп таблицы маршрутов. При этом **следует** ограничивать скорость отправки, особенно для групповой передачи.

3.8.1.2. Запросы порядковых номеров

При получении запроса seqno для данного router-id и порядкового номера узел ищет в своей таблице маршрутов выбранную запись для данного префикса. Если такая запись присутствует и имеет конечную метрику, а значения router-id различаются или эти значения совпадают, но порядковый номер записи не меньше (по модулю 2^{16}) запрошенного порядкового номера, узел **должен** передать обновление для данного префикса. Если router-id совпадают, но запрошенный порядковый номер больше (по модулю 2^{16}) чем в записи для маршрута, узел сравнивает router-id со своим идентификатором. При совпадении узел увеличивает свой порядковый номер на 1 (по модулю 2^{16}) и передает обновления. Узлу **недопустимо** увеличивать своей номер больше чем на 1 в ответ на один запрос seqno.

Если router-id в запросе не совпадает с идентификатором узла, принявший запрос узел проверяет поле Hop Count в запросе. Если поле имеет значение не меньше 2 и узел анонсирует префикс своим соседям, он выбирает соседа для пересылки тому запроса, как указано ниже.

- Если узел имеет один или несколько выполнимых маршрутов в направлении запрошенного префикса, где next-hop не является запросившим узлом, он **должен** переслать запрос слудующему маршрутизатору (next-hop).
- В остальных случаях, если узел имеет один или несколько (невыполнимых) маршрутов к запрошенному префиксу, где next-hop не является запросившим узлом, ему **следует** переслать этот запрос следующему маршрутизатору на одном из таких маршрутов.

Для фактической пересылки запроса узел декрементирует счетчик интервалов (hop count) и передает запрос по индивидуальному адресу выбранного соседа. Запрос **следует** пересылать как срочный TLV (параграф 3.1).

Узлу **следует** поддерживать список недавно пересланных запросов seqno и пересылать отклики на них (обновление с достаточно большим для удовлетворения запроса seqno) как срочные TLV (параграф 3.1). Узлу **следует** сравнивать каждый входящий запрос seqno со списком недавно пересылавшихся запросов seqno во избежание избыточной пересылки (т. е. он недавно пересылал запрос с тем же префиксом, prefix, router-id и не меньшим seqno по модулю 2^{16}).

Поскольку механизм пересылки запросов не обязан соответствовать условию выполнимости, запрос может попасть в петлю, поэтому в него включается счетчик интервалов для ограничения времени существования запроса в сети. Поскольку запросы всегда пересылаются в индивидуальных пакетах, начальное значение счетчика интервалов не требуется делать очень низким и не нужно выполнять поиск с расширением горизонта. Запрос **недопустимо** дублировать, **недопустимо** пересылать по групповым адресам и **недопустимо** пересылать нескольким соседям. Однако запрос, повторенный его инициатором, можно переслать другому соседу, нежели исходный запрос.

3.8.2. Передача запросов

Узел Babel **может** отправлять запросы маршрутов и порядковых номеров по групповым и индивидуальным адресам. Отправка запросов требуется лишь в одном случае (параграф 3.8.2.1).

3.8.2.1. Предотвращение истощения

Когда маршрут отозван или закончился срок его действия, узел Babel обычно переключается на другой выполнимый маршрут для того же префикса. Однако в некоторых случаях таких маршрутов может не оказаться. Узел, потерявший все выполнимые маршруты к данному адресату, но имеющий невыполнимые маршруты к нему с не истекшим сроком действия, **должен** отправить запрос seqno. Если нет даже таких маршрутов, узел все равно **может** передать запрос seqno. В поле router-id такого запроса указывается router-id из потерянного маршрута, а запрашиваемым номером является значение из таблицы источников, увеличенное на 1. Запрос включает счетчик интервалов, который служит «механизмом последней надежды» (last-resort) для гарантии сохранения узла в сети и может включать любое значение, превышающее диаметр сети (значение 64 приемлемо по умолчанию).

Если узел имет какие-либо (невыполнимые) маршруты к нужному адресату, он **должен** передать запрос хотя бы одному из соседей next-hop, анонсировавших эти маршруты, и **следует** передать его всем. В любом случае узел **может** передать запрос любому другому соседу независимо от анонсирования им нужного маршрута. Простой стратегией реализации будет безусловная отправка группового запроса через все интерфейсы.

Похожие запросы могут быть переданы другими узлами, на которые повлияла потеря маршрута. Если сеть остается соединенной и предполагается отсутствие потерь, хотя бы один из таких запросов будет переслан источнику и маршрут будет анонсирован с новым порядковым номером (в результате подавления дубликатов лишь малое число таких запросов достигнет источника). Для компенсации потери пакетов узлу **следует** повторить запрос несколько раз, если не появилось выполнимого маршрута в течение короткого времени (см. Приложение B) Однако при значительной потере пакетов могут пропасть все такие запросы. В этом случае получение нового порядкового номера обеспечит описанный в следующем параграфе механизм.

3.8.2.2. Работа с невыполнимыми обновлениями

Когда метрика маршрута увеличивается, узел может получить невыполнимое обновление для маршрута, который выбран. Как указано в параграфе 3.5.1, узел может игнорировать обновление или отменить выбор маршрута.

Для предотвращения ложного завершения срока действия маршрута, ставшего невыполнимым, узлу **следует** передать индивидуальный запрос seqno при получении невыполнимого обновления для выбранного маршрута. Порядковый номер для запроса рассчитывается по таблице источников, как указано в параграфе 3.8.2.1.

Поскольку расчет метрики не обязательно совпадает (по времени) с задержкой распространения обновлений, узел может получить невыполнимое обновление от не выбранного в данный момент соседа, который предпочтительней выбранного маршрута (т. е. имеет меньшую метрику). В таком случае узлу **следует** передать индивидуальный запрос seqno соседу, анонсировавшему предпочтительное обновление.

3.8.2.3. Предотвращение завершения срока действия маршрутов

При нормальной работе таймер срока действия маршрута не срабатывает никогда, поскольку время удержания маршрута рассчитывается на основе явного интервала из Update TLV и обновления (возможно отзывы) должны поступать своевременно, чтобы избежать завершения срока действия маршрута.

Однако при наличии потерь пакетов может случиться так, что обновление не будет получено в течение срока ожидания и срок действия маршрута закончится. Для предотвращения таких ситуаций узлу Babel, незадолго до завершения срока действия маршрута, **следует** передать индивидуальный запрос маршрута анонсировавшему его соседу. Поскольку узлы всегда передают обновление или отзыв в ответ на запрос точного (не шаблонного) маршрута (параграф 3.8.1.1), это обычно ведет к обновлению или отзыву маршрута.

4. Кодирование протокола

Пакет Babel **должен** передаваться как тело дейтаграммы UDP со значением счетчика интервалов сетевого уровня (hop count) 1 по общеизвестному групповому адресу протокола IPv4 или IPv6 (для IPv6 это адреса link-local). Для потов UDP отправителя и получателя должен указываться общеизвестный номер. Пакеты Babel **должны** игнорироваться, если адресом отправителя не является адрес IPv6 link-local или адрес IPv4, относящийся к локальной сети, а в качестве порта не указан общеизвестный порт Babel. Пакеты **могут** игнорироваться, если для получателя указан глобальный (маршрутизируемый) адрес IPv6.

Для минимизации числа передаваемых пакетов и предотвращения фрагментации на нижележащих уровнях узлу Babel **следует** использовать пакеты максимального размера, вплоть для MTU выходного интерфейса с учетом заголовков нижележащего уровня (28 октетов для UDP по протоколу IPv4 и 48 октетов для UDP по протоколу IPv6). **Недопустимо** передавать пакеты, размер которых превышает большее из значений MTU выходного интерфейса (с учетом нижележащих заголовков) или 512 октетов. В любом случае размер пакета не может быть больше 2^{16} - 1 с учетом заголовков нижележащего уровня. Каждый узел Babel **должен** поддерживать прием пакетов размером до большего из значений MTU приемного интерфейса с учетом заголовков нижележащего уровня или 512 октетов. **Недопустимо** передавать пакеты Babel в джамбограммах IPv6 [RFC2675].

4.1. Типы данных

4.1.1. Представление целых чисел

Многооктетные поля, представляющие целые числа начинаются со старшего октета (формат Big-Endian [IEN137], называемый также сетевым порядком байтов). Базовый протокол использует лишь значения без знака. Если расширению потребуются числа со знаком, оно должно будет задать кодирование (например, дополнение до 2).

4.1.2. Интервалы

Относительные значения времени указываются 16-битовыми числами в сотых долях секунды (centisecond). Это позволяет задать интервалы приблизительно до 11 минут с шагом 10 мсек., что должно быть достаточно для всех применений Babel (см. Прилжение B).

4.1.3. Идентификаторы маршрутизаторов

Идентификатор router-id представляет собой произвольной значение размером 8 октетов. **Недопустимо** использовать для router-id значения, содержащие только нули (шестнадцатеричное 0000000000000000) или только единицы (шестнадцатеричное FFFFFFFFFFFFFFFF).

4.1.4. Адреса

Поскольку основная работа протокола выполняется с адресами, поддерживается несколько типов их кодирования. Кроме того, в Update TLV может опускаться маска подсети при передаче множества адресов в одном пакете, это называется сжатием адресов (параграф 4.6.9). Варианты кодирования адресов приведены ниже.

AE 0
: Шаблонный адрес (строка октетов размером 0).

AE 1
: Адрес IPv4 (до 4 октетов).

AE 2
: Адрес IPv6 с поддержкой сжатия (до 16 октетов).

AE 3
: Адрес IPv6 link-local без сжатия (8 октетов, предполагается префикс fe80::/64).

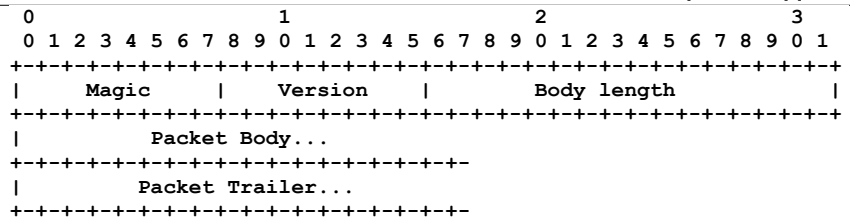
Семейством адресов, связанным с форматом представления является IPv4 или IPv6. Семейство не определено для AE 0, IPv4 для AE 1 и IPv6 для AE 2 и AE 3.

4.1.5. Префиксы

Сетевые префиксы кодируются как адреса, но хранятся с минимальным числом октетов, достаточным для записи значимых битов (до размера префикса).

4.2. Формат пакетов

Пакет Babel состоит из 4-октетного заголовка, за которым следует набор TLV (тело пакета) и может следовать еще один набор TLV (трейлер).Формат является расширяемым (см. Приложение D).



Magic

Тщательно подобранное произвольное значение 42 (десятичное). Пакеты, в которых первый октет имеет другое значение **должны** игнорироваться.

Version

Этот документ задает версию 2 протокола Babel. Пакеты, в которых второй октет имеет другое значение **должны** игнорироваться.

Body length

Размер следующего за заголовком тела пакета (без учета полей Magic, Version, Body length, а также трейлера) в октетах.

Packet Body

Тело пакета, представляющее собой последовательность TLV.

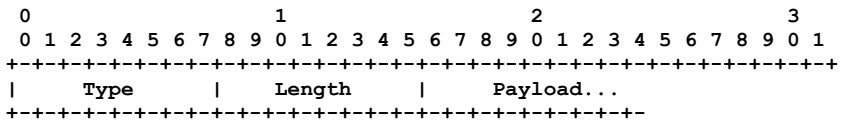
Packet Trailer

Трейлер пакета, содержащий последовательность TLV.

Телом и трейлером пакета служат цепочки TLV. Обычно TLV помещаются в тело пакета, а трейлеры применяются лишь для особых TLV, которые не требуют криптографической защиты. При анализе трейлера получатель **должен** игнорировать все TLV, в определениях которых явно не указана возможность размещения в трейлере. Из числа определенных в документе TLV таковыми являются лишь Pad1 и PadN. Поскольку эти TLV в любом случае игнорируются, реализация **может** полностью игнорировать трейлер, даже не анализируя его, пока не используется расширений, определяющих другие TLV для включения в трейлер.

4.3. Формат TLV

Все TLV, за исключением Pad1, имеют показанную на рисунке структуру.



Type

Тип TLV.

Length

Размер тела в октетах без учета полей Type и Length.

Payload

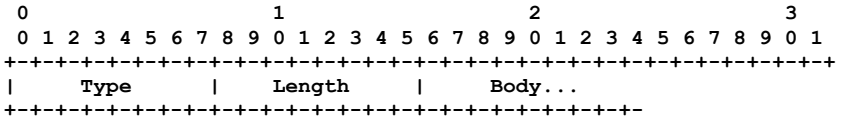
Данные TLV (payload), которые представляют собой тело, а в некоторых случаях - набор суб-TLV.

TLV неизвестного типа **должны** игнорироваться.

4.4. Формат суб-TLV

В заголовке каждого TLV явно указывается размер, однако большинство TLV являются самозавершающими в том смысле, что можно определить размер без явного обращения к полю Length. Если TLV имеет самозавершающий формат, пространство между естественным размером TLV и значением в поле Length может применяться для включения sub-TLV.

Структура суб-TLV не отличается от структры TLV и, кроме Pad1, все суб-TLV имеют показанный ниже формат.



Type

Тип суб-TLV.

Length

Размер тела в октетах без учета полей Type и Length.

Body

Тело суб-TLV, интерпретация которого зависит от типа суб-TLV и типа содержащего его TLV.

Старший бит типа суб-TLV (шестнадцатеричное значение 80) называется битом обязательности (mandatory). Иными словами, типы суб-TLV от 128 до 255 включают этот бит, который определяет обработку неизвестных суб-TLV. При сброшенном бите неизвестный суб-TLV целиком **должен** игнорироваться, а остальная часть TLV обрабатывается нормально. Если бит установлен, **должен** целиком игнорироваться блок TLV (за исключением обновления состояния анализатора с помощью Router-Id, Next Hop, Update TLV, как указано в следующем параграфе).

4.5. Состояние анализатора и кодирование обновлений

В больших сетях основную часть трафика Babel составляют обновления, поэтому были применены способы их эффективного кодирования. Включаемые в обновление данные концептуально разделены на 3 части (параграф 3.5):

- префикс, порядковый номер и метрика содержатся в самих Update TLV (параграф 4.6.9);
- router-id берется из Router-Id TLV, предшествующего Update TLV и может использоваться несколькими Update TLV (параграф 4.6.7);

- следующий интервал (next-hop) берется из адреса отправителя пакета сетевого уровня, содержащего пакет Babel, или из явного Next Hop TLV (параграф 4.6.8).

В дополнение к этому в Update TLV можно опускать префикс анонсируемого префикса, который извлекается из предшествующего Update TLV с тем же семейством адресов (IPv4 или IPv6). Кроме того, в особом случае совпадения router-id с идентификатором интерфейса в адресе IPv6 можно опустить Router-Id TLV и вывести значение router-id из младших битов анонсируемого префикса (параграф 4.6.9).

Для реализации этих методов сокращения в Babel применяется анализатор с учетом состояния, позволяющий TLV ссылаться на данные предыдущего TLV. Состояние анализатора включает указанные ниже данные.

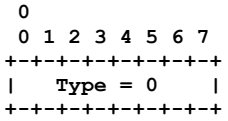
- Принятый по умолчанию префикс для каждой кодировки адреса с возможностью сжатия. Префикс не определен в начале пакета и обновляется каждым Update TLV с флагом Prefix (параграф 4.6.9).
- Текущий следующий интервал пересылки (next-hop) для каждого семейства адресов (IPv4 или IPv6). Это адрес отправителя в содержащем обновление пакете для совпадающего семейства адресов в начале пакета, обновляемый каждым Next Hop TLV (параграф 4.6.8);
- Текущее значение router-id. Значение не определено в начале пакета и обновляется каждым Router-Id TLV (параграф 4.6.7) и Update TLV с флагом Router-Id.

Поскольку состояние анализатора должно быть идентичным в реализациях, оно обновляется до проверки обязательности суб-TLV. Анализ TLV **должен** обновлять состояние анализатора, даже если в ином случае TLV игнорируется из-за наличия неизвестного обязательного суб-TLV или по иной причине.

Ни один из меняющих состояние анализатора TLV не разрешается включать в трейлер пакета, поэтому для анализа трейлеров реализация может использовать отдельный анализатор без учета состояний.

4.6. Специальные TLV

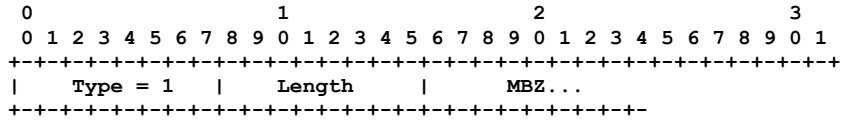
4.6.1. Заполнение Pad1



Type
Значение 0 указывает Pad1 TLV.

Такие TLV игнорируются при получении и могут включаться в трейлер пакета.

4.6.2. Заполнение PadN



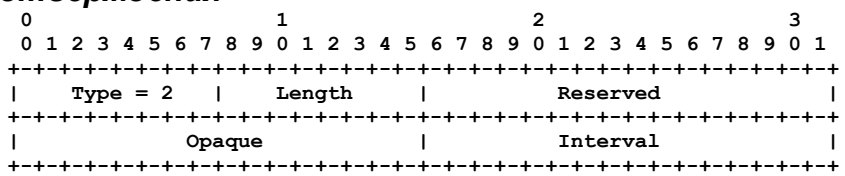
Type
Значение 1 указывает PadN TLV.

Length
Размер тела в октетах без учета полей Type и Length.

MBZ
Должно иметь значение 0, устанавливаемое при передаче.

Такие TLV игнорируются при получении и могут включаться в трейлер пакета.

4.6.3. Запрос подтверждения



TLV запрашивает у получателя отправку Acknowledgment TLV в течение заданного полем Interval времени.

Type
Значение 2 указывает Acknowledgment Request TLV.

Length
Размер тела в октетах без учета полей Type и Length.

Reserved
Устанавливается 0 при передаче и **должно** игнорироваться при получении.

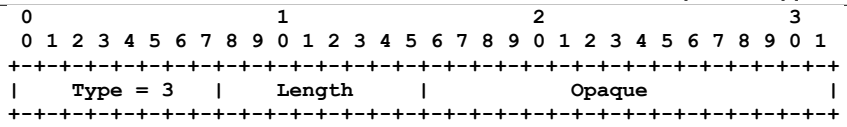
Opaque
Произвольное значение, возвращаемое в Acknowledgment TLV.

Interval
Интервал времени в сотых долях секунды (centisecond), по истечении которого отправитель сочтет пакет потерянным. Установка значения 0 **недопустима**. Получатель **должен** передать Acknowledgment TLV до истечения заданного интервала (с запасом на время доставки).

TLV относится к самозавершающим и может включать суб-TLV.

4.6.4. Подтверждение

TLV передается узлом в ответ на получение Acknowledgment Request TLV.



Type
Значение 3 указывает Acknowledgment TLV.

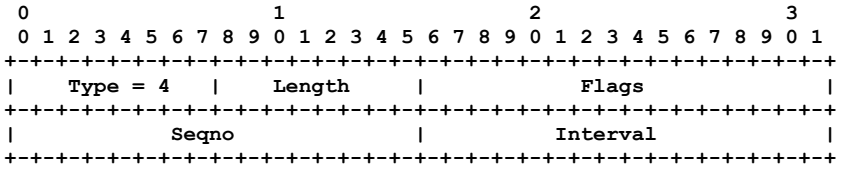
Length
Размер тела в октетах без учета полей Type и Length.

Opaque
Значение поля Opaque из Acknowledgment Request, вызвавшего это подтверждение.

Поскольку значения Opaque не уникальны в глобальном масштабе, этот блок TLV **должен** передаваться по индивидуальному адресу.

TLV относится к самозавершающим и может включать суб-TLV.

4.6.5. Hello



TLV для обнаружения соседей и определения стоимости приема от соседа.

Type
Значение 4 указывает Hello TLV.

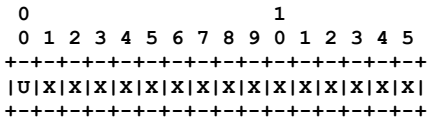
Length
Размер тела в октетах без учета полей Type и Length.

Flags
Биты флагов, задающие обработку данного TLV (см. ниже).

Seqno
При установленном флаге Unicast это будет значение порядкового номера Unicast Hello передающего узла для этого соседа. Иначе это будет исходящим порядковым номером Multicast Hello для данного интерфейса.

Interval
Ненулевое значение задает верхнюю границу (в сотых долях секунды) времени, по истечение которого передающих узел будет повторять плановое сообщение Hello TLV с таким же флагом Unicast. Нулевое значение указывает неплановое сообщение Hello, которое не включает данных о времени передачи других Hello.

Формат поля Flags показан на рисунке.



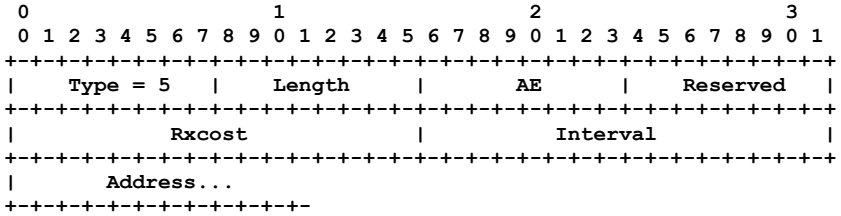
U (Unicast)
Флаг (шестнадцатеричное значение 8000), установка которого указывает Unicast Hello, сброс - Multicast Hello.

X
Все остальные биты **должны** иметь значение 0 при отправке и игнорируются при получении.

При каждой отправке сообщения Hello **должно** инкрементироваться значение соответствующего счетчика seqno. Поскольку все Multicast Hello используют общий счетчик seqno для интерфейса на узле, если флаг Unicast сброшен, TLV **должен** передаваться всем соседям на этом канале, что можно обеспечить отправкой по групповому адресу или передачей групповых пакетов по индивидуальным адресам всех доступных соседей. И наоборот, при установленном флаге Unicast этот блок TLV **должен** передаваться одному соседу по индивидуальному адресу. Во избежание существенного снижения снижения качества канала **не следует** передавать несколько Hello TLV в одном пакете.

TLV относится к самозавершающим и может включать суб-TLV.

4.6.6. IHU



IHU TLV (я слышу вас) служит для подтверждения двухсторонней доступности и передачи стоимости отправки в канал.

Type
Значение 5 указывает IHU TLV.

Length
Размер тела в октетах без учета полей Type и Length.

AE
Вариант кодирования поля Address (в большинстве случаев 1 или 3). В качестве оптимизации **можно** применять 0, если TLV передается по индивидуальному адресу, ассоциация организована по каналу «точка-точка» или двухсторонняя доступность устанавливается средствами другого протокола (вне Babel).

Reserved
Устанавливается 0 при передаче и **должно** игнорироваться при получении.

Rxcost
Значение rxcost, соответствующее передающему узлу интерфейса, адрес которого указан в поле Address. Шестнадцатеричное значени FFFF (бесконечность) указывает недоступность интерфейса.

Interval
Верхняя граница времени по истечении которого предающий узел будет повторять IHU (в сотых долях секнды). Значение 0 **недопустимо**. Принимающий узел использует это значение для расчета времени удержания данной симметричной ассоциации (связи).

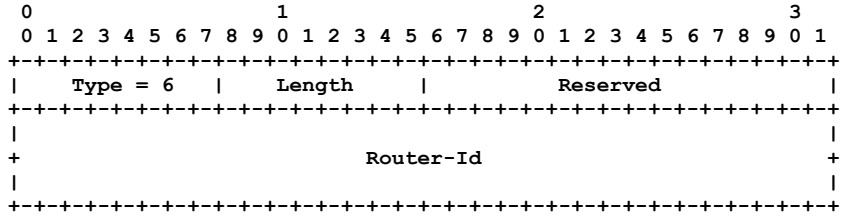
Address
Адрес получателя в формате, заданном полем AE. Сжатие адреса не допускается.

Концептуально IHU адресуется одному соседу. Однако IHU TLV включают явный адрес получателя и **могут** передаваться по групповому адресу, что позволяет объединить IHU для нескольких соседей в один пакет, избавляя от необходимости использовать обмен ARP или Neighbour Discovery, когда сосед не участвует в трафике данных.

IHU TLV с неизвестным значением в поле AE **должны** игнорироваться.

TLV относится к самозавершающим и может включать суб-TLV.

4.6.7. Router-Id



Router-Id TLV устанавливает значение router-id, которое предполагается следующими Update TLV, как указано в параграфе 4.5. TLV задает router-id даже при наличии в нем неизвестного обязательного суб-TLV.

Type
Значение 6 указывает Router-Id TLV.

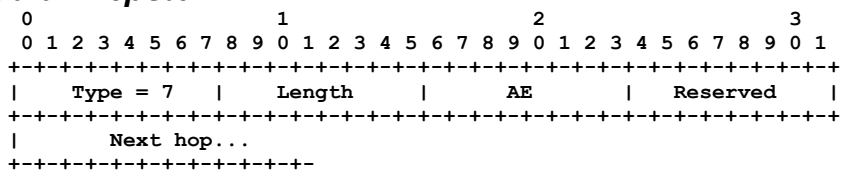
Length
Размер тела в октетах без учета полей Type и Length.

Reserved
Устанавливается 0 при передаче и **должно** игнорироваться при получении.

Router-Id
Значение router-id для маршрутов, анонсируемых в последующих Update TLV. **Недопустимы** значения, содержащие только 0 или только 1.

TLV относится к самозавершающим и может включать суб-TLV.

4.6.8. Следующий интервал



Next Hop TLV задает адрес следующего интервала (next-hop) для данного семейтсва адресов (IPv4 или IPv6), которые предполагается следующими Update TLV, как указано в параграфе 4.5. TLV задает следующий интервал для последующих Update TLV даже при наличии в нем неизвестного обязательного суб-TLV.

Type
Значение 7 указывает Next Hop TLV.

Length
Размер тела в октетах без учета полей Type и Length.

AE
Вариант кодирования поля Address. **Следует** задавать 1 (IPv4) или 3 (IPv6 link-local), **недопустимо** указывать 0.

Reserved
Устанавливается 0 при передаче и **должно** игнорироваться при получении.

Next hop
Адрес next-hop, анонсируемый последующими Update TLV для этого семейства адресов.

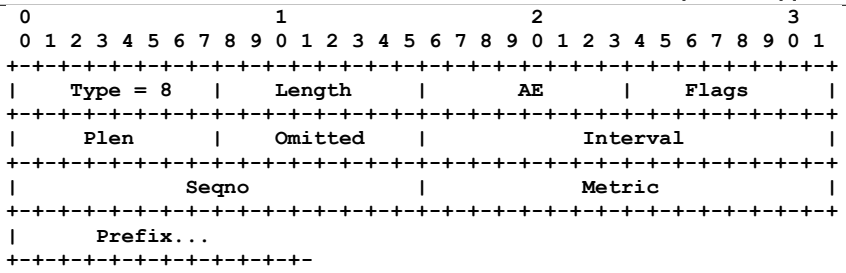
Когда семейство адресов соответствует протоколу сетевого уровня, который передает пакет, Next Hop TLV не требуется и адрес следующего интервала при отсутствии Next Hop TLV берется из адреса отправителя пакета.

Next Hop TLV с неизвестным значением в поле AE **должны** игнорироваться.

TLV относится к самозавершающим и может включать суб-TLV.

4.6.9. Обновление

Update TLV анонсирует или отзывает маршрут. В качестве оптимизации может дополнительно устанавливать новое подразумеваемое значение router-id и принятого по умолчанию префикса, как указано в параграфе 4.5.



Type
Значение 8 указывает Update TLV.

Length
Размер тела в октетах без учета полей Type и Length.

AE
Вариант кодирования поля Prefix.

Flags
Биты флагов, задающие обработку данного TLV (см. ниже).

Plen
Размер анонсируемого префикса в битах. В случае AE 3 (IPv6 link-local) поле Omitted **должно** иметь значение 0.

Omitted
Число октетов, опускаемых в начале анонсируемого префикса, которые следует брать из предшествующего Update TLV для того же семейства адресов при установленном флаге Prefix.

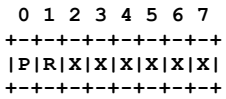
Interval
Верхняя граница интервала, по истечение которого передающий узел будет отправлять новое обновление для данного префикса (в сотых долях секунды). Значение 0 **недопустимо**. Принимающий узел использует это значение для расчета времени удержания записи в таблице маршрутов. Шестнадцатеричное значение FFFF (бесконечность) говорит, что это обновление не будет повторяться без запроса (параграф 3.8.2.3).

Seqno
Порядковый номер инициатора для данного обновления.

Metric
Метрика маршрута. Шестнадцатеричное значение FFFF (бесконечность) указывает отзыв маршрута.

Prefix
Анонсируемый префикс размером (Plen/8 - Omitted) с округлением в большую сторону.

Формат поля Flags показан на рисунке.



P (Prefix)
Флаг (шестнадцатеричное значение 8000), установка которого указывает, что данный Update TLV задает новый префикс для последующих Update TLV в пакете с тем же кодированием адресов. Значение устанавливается даже при наличии в данном TLV неизвестного обязательного суб-TLV.

R (Router-Id)
Флаг (шестнадцатеричное значение 4000), установка которого указывает, что данный TLV задает новое значение принятого по умолчанию router-id для этого TLV и Update TLV в пакете с тем же кодированием адресов. Значение устанавливается даже при наличии в данном TLV неизвестного обязательного суб-TLV и рассчитывается из первого адреса анонсируемого префикса, как показано ниже.

- Если размер адреса не меньше 8 октетов из последних 8 октетов адреса извлекается значение router-id.
- Если размер адреса меньше 8 октетов, новое значение router-id состоит из нужного числа нулевых октетов и поля адреса, записываемого в правую часть router-id. Например, для IPv4 значение router-id будет включать 4 октета нулей, за которыми следует адрес IPv4.

X
Все остальные биты **должны** иметь значение 0 при отправке и игнорируются при получении.

Расчет префикса, анонсируемого данным Update TLV, выполняется в соответствии с приведенным ниже описанием.

- Первые Omitted октетов префикса берутся из предыдущего Update TLV с флагом Prefix и тем же кодированием адресов даже при наличии так неизвестного обязательного суб-TLV. Если поле Omitted отлично от 0, но такого TLV нет, данное обновление **должно** игнорироваться.
- Следующие (Plen/8 - Omitted) октетов (с округлением вверх) берутся из поля Prefix.
- Если Plen не кратно 8, все биты вне Plen (младшие (8 - Plen MOD 8) битов последнего октета) сбрасываются.
- В оставшихся октетах устанавливается 0.

Если поле Metric имеет конечное значение, router-id узла-инициатора для данного анонса берется из префикса, анонсируемого этим обновлением, при наличии флага Router-Id, рассчитанного, как указано выше. В остальных случаях значение берется из предшествующего Router-Id TLV или Update TLV с флагом Router-Id, в зависимости от того, что указано последним, даже при наличии в TLV неизвестного обязательного суб-TLV. При отсутствии подходящего TLV обновление игнорируется.

Адрес следующего интервала (next-hop) для обновления берется из последнего предшествующего Next Hop TLV с тем же семейством адресов (IPv4 или IPv6) в том же пакете даже при наличии неизвестного обязательного суб-TLV. Если такого TLV нет, значение берется из адреса отправителя в пакете сетевого уровня, содержащем данное обновление, если он относится к тому же семейству адресов. В остальных случаях это обновление **должно** игнорироваться.

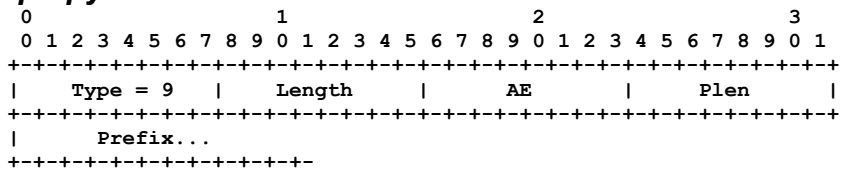
Шестнадцатеричное значение FFFF в поле метри указывает отзыв маршрута. В этом случае router-id, next-hop и seqno не используются. AE **может** иметь значение 0 и в этом случае Update TLV отзывает все маршруты, анонсированные ранее передавшим интерфейсом. Если метрика конечна, установка AE 0 **недопустима**. Update TLV с конечной

метрикой и AE 0 **должны** игнорироваться. Если метрика бесконечная и AE имеет значение 0, в полях Plen и Omitted **должно** быть значение 0. Update TLV, не соответствующие этому требованию, **должны** игнорироваться.

Update TLV с неизвестным значением в поле AE **должны** игнорироваться.

TLV относится к самозавершающим и может включать суб-TLV.

4.6.10. Запрос маршрута



Route Request TLV приглашает получателя передать обновление для заданного префикса или всю таблицу маршрутов.

Type
Значение 9 указывает Route Request TLV.

Length
Размер тела в октетах без учета полей Type и Length.

AE
Вариант кодирования поля Prefix. Значение 0 указывает запрос полного дампа таблицы маршрутов (шаблонный запрос).

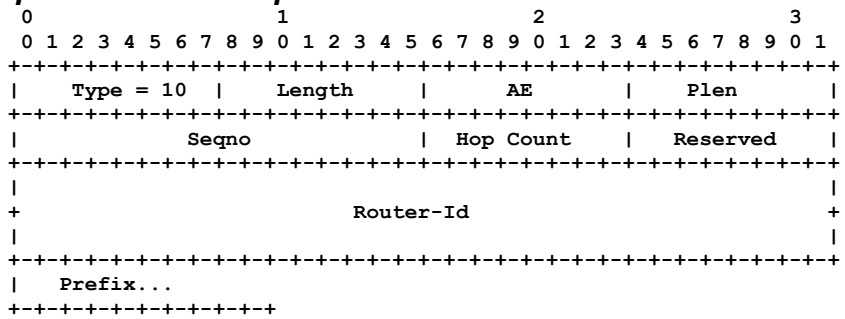
Plen
Размер запрашиваемого префикса в битах.

Prefix
Запрашиваемый префикс, размер которого составляет Plen/8 с округлением в большую сторону.

Route Request TLV приглашает получателя передать обновление (возможно отзыв) для префикса, указанного полями AE, Plen и Prefix или полный дамп его таблицы маршрутов, если AE имеет значение 0 (в этом случае поле Plen должно иметь значение 0 и размер Prefix также должен быть 0). Request TLV с AE 0 и ненулевым Plen **должны** игнорироваться.

TLV относится к самозавершающим и может включать суб-TLV.

4.6.11. Запрос порядкового номера



Seqno Request TLV приглашает получателя передать Update для данного префикса с данным порядковым номером или переслать запрос, если его нельзя выполнить локально.

Type
Значение 10 указывает Seqno Request TLV.

Length
Размер тела в октетах без учета полей Type и Length.

AE
Формат кодирования поля Prefix. Значение 0 **недопустимо**.

Plen
Размер запрашиваемого префикса в битах.

Seqno
Запрашиваемый порядковый номер.

Hop Count
Максимальное число пересылок данного TLV плюс 1. Значение 0 **недопустимо**.

Reserved
Устанавливается 0 при передаче и **должно** игнорироваться при получении.

Router-Id
Запрашиваемый идентификатор Router-Id. **Недопустимы** значения, содержащие только 0 или только 1.

Prefix
Запрашиваемый префикс, размер которого составляет Plen/8 с округлением в большую сторону.

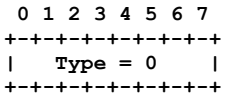
Seqno Request TLV приглашает получателя передать Update с конечной метрикой для префикса, указанного полями AE, Plen и Prefix, с router-id, отличным от заданного полем Router-Id, или Seqno не меньше (по модулю 2^{16}) указанного в поле Seqno. Если запрос нельзя выполнить локально, он пересылается в соответствии с правилами параграфа 3.8.1.2.

Хотя Seqno Request **можно** передавать по групповому адресу, его **недопустимо** пересылать по групповому адресу, а также **недопустима** пересылка более чем одному соседу. Запрос **недопустимо** пересылать при Hop Count = 1.

TLV относится к самозавершающим и может включать суб-TLV.

4.7. Специальные суб-TLV

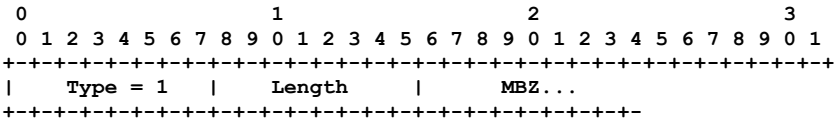
4.7.1. Pad1



Type
Значение 0 указывает суб-TLV Pad1.

Суб-TLV игнорируется при получении и может включаться в TLV, поддерживающие суб-TLV.

4.7.2. PadN



Type
Значение 1 указывает суб-TLV PadN.

Length
Размер тела в октетах без учета полей Type и Length.

MBZ
Должно иметь значение 0, устанавливаемое при передаче.

Суб-TLV игнорируется при получении и может включаться в TLV, поддерживающие суб-TLV.

5. Взаимодействие с IANA

Агентство IANA зарегистрировало порт UDP с номером 6696, названный babel, для использования протоколом Babel.

Агентство IANA зарегистрировало multicast-группы IPv6 ff02::1:6 и IPv4 224.0.0.111 для протокола Babel.

В IANA создан реестр Babel TLV Types с политикой распределения Specification Required [RFC8126] для типов 0-223 и Experimental Use для типов 224-254. Значения реестра приведены в таблице 1.

Таблица 1.

Тип	*Имя*	*Документ*
0	Pad1	RFC 8966
1	PadN	RFC 8966
2	Acknowledgment Request	RFC 8966
3	Acknowledgment	RFC 8966
4	Hello	RFC 8966
5	IHU	RFC 8966
6	Router-Id	RFC 8966
7	Next Hop	RFC 8966
8	Update	RFC 8966
9	Route Request	RFC 8966
10	Seqno Request	RFC 8966
11	TS/PC	[RFC7298]
12	HMAC	[RFC7298]
13	Reserved	
14	Reserved	
15	Reserved	
224-254	Reserved for Experimental Use	RFC 8966
255	Reserved for expansion of the type space	RFC 8966

В IANA создан реестр Babel Sub-TLV Types с политикой распределения Specification Required для типов 0-111 и 128-239 и политикой Experimental Use для типов 112-126 и 240-254. Значения реестра приведены в таблице 2.

Таблица 2.

Тип	*Имя*	*Документ*
0	Pad1	RFC 8966
1	PadN	RFC 8966
2	Diversity	[BABEL-DIVERSITY]
3	Timestamp	[BABEL-RTT]
4-111	Unassigned	
112-126	Reserved for Experimental Use	RFC 8966
127	Reserved for expansion of the type space	RFC 8966
128	Source Prefix	[BABEL-SS]
129-239	Unassigned	
240-254	Reserved for Experimental Use	RFC 8966
255	Reserved for expansion of the type space	RFC 8966

В IANA создан реестр Babel Address Encodings с политикой распределения Specification Required для кодирования адресов (AE) 0-223 и Experimental Use для 224-254. Значения реестра приведены в таблице 3.

Таблица 3.

AE	*Имя*	*Документ*
0	Wildcard address	RFC 8966

1	IPv4 address	RFC 8966
2	IPv6 address	RFC 8966
3	Link-local IPv6 address	RFC 8966
4-223	Unassigned	
224-254	Reserved for Experimental Use	RFC 8966
255	Reserved for expansion of the AE space	RFC 8966

Реестр Babel Flags Values переименован в Babel Update Flags Values. Для реестра применяется политика распределения Specification Required. Значения реестра приведены в таблице 4.

Таблица 4.

Бит	***Имя***	***Документ***
0	Default prefix	RFC 8966
1	Default router-id	RFC 8966
2-7	Unassigned	

В IANA создан реестр Babel Hello Flags Values с политикой распределения Specification Required. Значения реестра приведены в таблице 5.

Таблица 5.

Бит	***Имя***	***Документ***
0	Unicast	RFC 8966
1-15	Unassigned	

Агентство IANA заменило ссылки на RFC 6126 и RFC 7557 во всех перечисленных выше реестрах на данный документ.

6. Вопросы безопасности

Как указано в этом документе, протокол Babel является совершенно не защищенным. Без дополнительных механизмов защиты Babel доверяет любой информации, приходящей в открытых дейтаграммах UDP, и действует на основании ее. Злоумышленник в локальной сети может влиять на работу Babel разными способами. Например,

- подменять пакеты Babel и перенаправлять трафик, анонсируя анонсирования маршруты с меньшей метрикой, большим порядковым номером или более длинным префиксом;
- создавать некорректно сформированные пакеты, нарушающие работу недостаточно устойчивых реализаций или препятствующие остальной сети;
- повторно использовать перехваченные пакеты Babel, что может вести к перенаправлению трафика, возникновению «черных дыр» и даже нарушению работы всей сети.

При передаче по протоколу IPv6 пакеты Babel игнорируются, если они не переданы с адреса IPv6 link-local. Поскольку маршрутизаторы не пересылают пакеты IPv6 link-local, это смягчает атаки, указанные выше, необходимостью для злоумышленника иметь доступ к локальному каналу. Для пакетов Babel, переданных по протоколу IPv4 такой естественной защиты нет и это служит одной из причин рекомендуемого использования Babel на основе IPv6 (параграф 3.1).

Обычно сложно обеспечить доверие к пакетам, прибывающим на узел Babel, даже при защищенном локальном канале. По этой причине **рекомендуется** защищать весь трафик Babel криптографическим протоколом на уровне приложения. Имеется два подходящих механизма, выбор между которыми определяется копромиссом между простотой и безопасностью.

- Babel по протоколу DTLS [RFC8968] передает основную часть трафика Babel по протоколу DTLS и этот протокол применяется для аутентификации узлов, а также защиты целостности и конфиденциальности.
- MAC-аутентификация [RFC8967] добавляет код аутентификации сообщений (message authentication code или MAC) в каждый пакет Babel для подтверждения его отправки узлом, знающим общий секрет, и включения дополнительной информации для проверки свежести пакета (предотвращение повторного использования).

Оба механизма позволяют узлам игнорировать пакеты от злоумышленника без требуемых свидетельств. Обеспечивается также целостность сообщений и предотвращается их повторное использование. Babel-DTLS использует асимметричные ключи и обеспечивает конфиденциальность, а Babel-MAC имеет более ограниченную область применения (см. параграфы 1.1, 1.2, 7 в [RFC8967]). Поскольку Babel-MAC проще и более облегчен, его рекомендуется использовать в приложениях Babel, где ограничения протокола допустимы, например, достаточно симметричных ключей, а маршрутные данные не считаются конфиденциальными.

Каждой реализации Babel **следует** поддерживать BABEL-MAC.

Следует учитывать, что информации, которую мобильный узел Babel анонсирует всему домену маршрутизации, достаточно для определения физического местоположения мобильного узла с неплохой точностью, что может вызывать проблемы приватности даже при защите трафика управления от неаутентифицированных злоумышленников криптографическими механизмами, такими как Babel-DTLS. Эту проблему можно смягчить использованием случайных значений router-id и адресов IP, а также их достаточно частой сменой.

7. Литература

7.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <https://www.rfc-editor.org/info/rfc793>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8967] Dô, C., Kolodziejak, W., and J. Chroboczek, "MAC Authentication for the Babel Routing Protocol", RFC 8967, DOI 10.17487/RFC8967, January 2021, <https://www.rfc-editor.org/info/rfc8967>.

7.2. Дополнительная литература

[BABEL-DIVERSITY] Chroboczek, J., "Diversity Routing for the Babel Routing Protocol", Work in Progress, Internet-Draft, draft-chroboczek-babel-diversity-routing-01, 15 February 2016, <https://tools.ietf.org/html/draft-chroboczek-babel-diversity-routing-01>.

[BABEL-RTT] Jonglez, B. and J. Chroboczek, "Delay-based Metric Extension for the Babel Routing Protocol", Work in Progress, Internet-Draft, draft-ietf-babel-rtt-extension-00, 26 April 2019, <https://tools.ietf.org/html/draft-ietf-babel-rtt-extension-00>.

[BABEL-SS] Boutier, M. and J. Chroboczek, "Source-Specific Routing in Babel", Work in Progress[1], Internet-Draft, draft-ietf-babel-source-specific-07, 28 October 2020, <https://tools.ietf.org/html/draft-ietf-babel-source-specific-07>.

[DSDV] Perkins, C. and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers", ACM SIGCOMM '94: Proceedings of the conference on Communications architectures, protocols and applications, 234-244, DOI 10.1145/190314.190336, October 1994, <https://doi.org/10.1145/190314.190336>.

[DUAL] Garcia Luna Aceves, J. J., "Loop-free routing using diffusing computations", IEEE/ACM Transactions on Networking, 1:1, DOI 10.1109/90.222913, February 1993, <https://doi.org/10.1109/90.222913>.

[EIGRP] Albrightson, B., Garcia Luna Aceves, J. J., and J. Boyle, "EIGRP -- a Fast Routing Protocol Based on Distance Vectors", Proc. Networld/Interop 94, 1994.

[ETX] De Couto, D., Aguayo, D., Bicket, J., and R. Morris, "A high-throughput path metric for multi-hop wireless networks", MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking, 134-146, DOI 10.1145/938985.939000, September 2003, <https://doi.org/10.1145/938985.939000>.

[IEEE802.11] IEEE, "IEEE Standard for Information technology--Telecommunications and information exchange between systems Local and metropolitan area networks—Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE 802.11-2012, DOI 10.1109/ieeestd.2012.6178212, April 2012, <https://doi.org/10.1109/ieeestd.2012.6178212>.

[IEN137] Cohen, D., "On Holy Wars and a Plea for Peace", IEN 137, 1 April 1980.

[IS-IS] International Organization for Standardization, "Information technology -- Telecommunications and information exchange between systems — Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473)", ISO/IEC 10589:2002, 2002.

[JITTER] Floyd, S. and V. Jacobson, "The Synchronization of Periodic Routing Messages", IEEE/ACM Transactions on Networking, 2, 2, 122-136, DOI 10.1109/90.298431, April 1994, <https://doi.org/10.1109/90.298431>.

[OSPF] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[PACKETBB] Clausen, T., Dearlove, C., Dean, J., and C. Adjih, "Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", RFC 5444, DOI 10.17487/RFC5444, February 2009, <https://www.rfc-editor.org/info/rfc5444>.

[RFC2675] Borman, D., Deering, S., and R. Hinden, "IPv6 Jumbograms", RFC 2675, DOI 10.17487/RFC2675, August 1999, <https://www.rfc-editor.org/info/rfc2675>.

[RFC3561] Perkins, C., Belding-Royer, E., and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing", RFC 3561, DOI 10.17487/RFC3561, July 2003, <https://www.rfc-editor.org/info/rfc3561>.

[RFC6126] Chroboczek, J., "The Babel Routing Protocol", RFC 6126, DOI 10.17487/RFC6126, April 2011, <https://www.rfc-editor.org/info/rfc6126>.

[RFC7298] Ovsienko, D., "Babel Hashed Message Authentication Code (HMAC) Cryptographic Authentication", RFC 7298, DOI 10.17487/RFC7298, July 2014, <https://www.rfc-editor.org/info/rfc7298>.

[RFC7557] Chroboczek, J., "Extension Mechanism for the Babel Routing Protocol", RFC 7557, DOI 10.17487/RFC7557, May 2015, <https://www.rfc-editor.org/info/rfc7557>.

[RFC8968] Décimo, A., Schinazi, D., and J. Chroboczek, "Babel Routing Protocol over Datagram Transport Layer Security", RFC 8968, DOI 10.17487/RFC8968, January 2021, <https://www.rfc-editor.org/info/rfc8968>.

[RIP] Malkin, G., "RIP Version 2", STD 56, RFC 2453, DOI 10.17487/RFC2453, November 1998, <https://www.rfc-editor.org/info/rfc2453>.

[1] Опубликовано в RFC 9079. *Прим. перев.*

Приложение A. Расчет стоимости и метрики

Стратегия рачсета стоимости и метрики маршрутов определяется локально. Протокол Babel требует лишь соответствия условиям, указанным в параграфах 3.4.3 и 3.5.2. Разные узлы могут применять разную стратегию в одной сети и даже на разных типах интерфейсов. В это приложении описаны варианты, проверенные в реальных сетях.

Говоря кратко, узел поддерживает статистику на уровне соседа для 16 последних Hello TLV каждого типа (Приложение A.1) и рассчитывает стоимость, используя стратегию !два из трех» (Приложение A.2.1) на проводных каналах и ETX[1] (Приложение A.2.2) на беспроводных. Для расчета метрики применяется аддитивная алгебра (параграф 3.5.2).

A.1. Поддержка истории Hello

Для каждого соседа узел поддерживает два комплекта истории Hello (по одному для каждого типа) и ожидаемый порядковый номер (один для Multicast, другой для Unicast Hello). Каждая история Hello представляет собой 16-битовый вектор, где 1 указывает прием Hello, а 0 - пропуск. Для каждого типа Hello ожидаемым порядковым номером (ne) является номер, который предполагается получить в следующем Hello от этого соседа.

При каждом приеме пакета Hello данного типа от соседа узел сравнивает полученный порядковый номер (nr) с ожидаемым для этого типа Hello номером ne. В зависимости от результат выполняются указанные ниже действия.

- Если номера различаются больше чем на 16 (по модулю 2^{16}), это говорит о возможной перезагрузке передавшего пакет узла с потерей порядковых номеров. В этом случае запись таблицы соседей очищается и создается заново.
- В остальных случаях, если nr меньше (по модулю 2^{16}) ожидаемого номера ne, это говорит об увеличении передающим узлом интервала Hello без уведомления и принимающий узел удаляет последние (ne - nr) элементов из истории Hello для этого соседа (undo history).
- Если nr больше (по модулю 2^{16}) чем ne, это говорит об уменьшении передающим узлом интервала Hello и потере некоторых сообщения Hello. Принимающий узел добавляет (nr - ne) битов со значением 0 в историю Hello (fast-forward).

Принимающий узел добавляет бит 1 в историю Hello и устанавливает ne = (nr + 1). Если поле Interval в принятом Hello отлично от 0, для таймера Hello с соседом устанавливается значение в 1,5 анонсированного интервала (увеличение позволяет учесть вариации задержки). По завершении отсчета связанного с соседом таймера локальный узел добавляет 0 в историю Hello и инкрементирует номер ожидаемого Hello. Если обе истории Hello пусты (содержат лишь 0), запись соседа сбрасывается. В противном случае для соответствующего таймера Hello устанавливается значение, анонсированное в последнем Hello того же типа от данного соседа (без увеличения, поскольку вариации задержки уже учтены при расчете только что завершившегося тайм-аута).

A.2. Расчет стоимости

В этом параграфе описаны два алгоритма расчета стоимости (параграф 3.4.3) на основе истории Hello. Алгоритм из парарафа A.2.1 применим к проводным каналам, A.2.2 - к беспроводным. **Рекомендуемые** значения для установки по умолчанию для параметров этих алгоритмов указаны в Приложении B.

A.2.1. k-out-of-j

Метод k-out-of-j подходит для проводных каналов, которые могут принимать два состояния - рабочее (up), где пакеты могут иногда отбрасываться, и нерабочее (down), когда отбрасываются все пакеты.

Метод k-out-of-j параметризуется двумя небольшими целыми числами k и j, так что 0 < k <= j, и номинальной стоимостью канала C >= 1. Узел хранит историю последних j Hello и если из них не менее k были получены корректно, канал считается работающим (up) и устанавливается C. В противном случае канал считается неработающим (down) и устанавливается бесконечное значение rxcost.

Поскольку Babel поддерживает два типа Hello, узел Babel выполняет k-out-of-j дважды для каждого соседа с историей Unicast Hello и Multicast Hello. Если любой из вариантов k-out-of-j показывает рабочее (up) состояние канала, канал считается активным (up) и устанавливается rxcost = C. Если оба варианта указывают нерабочее (down) состояние канала, он считается неактивным и для rxcost устанавливается бесконечное значение. Иными словами, результирующим значением rxcost будет меньший из результатов двух экземпляров k-out-of-j link.

Стоимость канала с определением k-out-of-j будет:

- cost = FFFF (шестнадцатеричное), если rxcost = FFFF (шестнадцатеричное);
- cost = txcost в остальных случаях.

A.2.2. ETX

В отличии от проводных каналов, которые имеют лишь два состояния (up или down), у беспроводных соединений параметры качество могут изменяться непрерывно. Наивное применение маршрутизации по числу интервалов пересылки в сетях с беспроводными каналами для транзита имеет тенденцию к выбору длинных каналов с потерями вместо более коротких каналов без потерь, что может значительно снизить пропускную способность. По этой причине протоколы маршрутизации для беспроводных сетей должны выполнять ту или иную оценку качества соединений.

Алгоритм ETX [ETX] является простым способом оценки качества канала, предназначенным для работы с IEEE 802.11 MAC [IEEE802.11]. По умолчанию IEEE 802.11 MAC выполняет запрос автоматического повтора (Automatic Repeat Query или ARQ) и адаптацию скорости для индивидуальных кадров, но не делает этого для групповых, которые передаются с фиксированной скоростью без ARQ. Поэтому измерение частоты потерь групповых кадроа дает полезную оценку качества канала.

[1] Expected Transmission Cost - ожидаемая стоимость передачи.

Узел, выполняющий оценку качества канала ETX, использует историю Multicast Hello для расчета оценки (beta) вероятности успешного приема Hello TLV. Значение параметра рассчитывается как доля битов 1 в небольшой (скажем, 6) группе последних записей истории Multicast Hello, экспоненциальное среднее значения или комбинация этих вариантов. Пусть rxcost = 256/beta.

Пусть alpha = MIN(1, 256/txcost) будет оценкой вероятности успешной передачи Hello TLV. Тогда стоимость будет cost = 256/(alpha * beta) или, эквивалентно, cost = (MAX(txcost, 256) * rxcost)/256.

Поскольку IEEE 802.11 MAC выполняет ARQ для индивидуальных кадров, такие кадры не вносят полезного вклада в измерение качества канала и ETX игнорирует историю Unicast Hello. Таким образом, узел, выполняющий оценку качества канала ETX, не будет маршрутизировать пакеты через соседние узлы, от которых он не получает Multicast Hello (возможно в дополнение в Unicast Hello).

A.3. Выбор маршрутов и гистерезис

В процессе выбора маршрута (параграф 3.6) узел выбирает один из доступных для данного адресата маршрутов. В Babel любая процедура, выбирающая лишь выполнимые маршруты с конечной метрикой будет давать маршрутизацию без петель. Однако при наличии постоянно меняющейся метрики, такой как ETX (Приложение A.2.2) наивная процедура выбора может приводить к постоянным переключениям маршрутов. Такие переключения можно ограничить и даже предотвратит путем задания гистерезиса в алгоритме выбора маршрута, например, учитывая при выборе историю маршрутов. Хорошие результаты должны давать любые разумные алгоритмы гистерезиса и ниже приведен один из вариантов, успешно развернутый во многих реализациях.

Для каждого маршрута R в дополнение к его метрике m(R) поддерживается сглаженная версия метрики ms(R) (**рекомендуется** рассчитывает ее как экспоненциально сглаженное среднее значение метрики в соответствии с параграфом 3.7 в [RFC793] для интервала, равного интервалу Hello, умноженному на небольшое число, например , 3). Если маршрута к данному адресату не выбрано, выбирается маршрут с наименьшей метрикой без учета сглаживания. Если выбран маршрут R, переключение на R' выполняется лишь при условии m(R') < m(R) и ms(R') < ms(R).

Интуитивно сглаженная метрика представляется долгосрочной оценкой качества маршрута. Описанный выше алгоритм работает лишь путем переключения маршрута когда краткосрочная когда выгодность такого переключения показывают сразу и мгновенная и долгосрочная оценка качества маршрута.

Приложение B. Параметры протокола

Выбор временных констант является компромиссом между быстрым обнаружением перемещений (mobility event) и издержками протокола. Два экземпляра Babel с разными временными константами могут взаимодействовать, но при этом максимальное время схождения будет определять более медленная реализация. Интервал Hello является наиболее важной из временных констант, поскольку перебои или перемещения обнаруживаются в течение 1,5 - 3,5 интервалов Hello. В результате использования Babel таблицы маршрутов с избыточностью и работы протокола на основе триггерных обновлений интервал Update не оказывает существенного влияния на время схождения после отказа. На практике он сильно влияет лишь на время получения новых маршрутов после перемещений (mobility event). Хотя протокол разрешает интервалы меньше от 10, столь малые значения вызывают большой протокольный трафик при незначительной практической пользе.

Ниже даны проверенные на практике в разных средах значения, которые **рекомендуется** применять по умолчанию.

Интервал Multicast Hello
: 4 секунды.

Интервал Unicast Hello
: Бесконечный (Unicast Hello не передаются).

Стоимость канала
: Оценка ETX на беспроводных каналах и «два из трех» с C=96 - на проводных.

Интервал IHU
: Анонсируемый интервал IHU составляет 3 интервала Multicast Hello. IHU передаются с каждым Hello на каналах с потерями (в соответствии с историей Hello), но лишь с каждым третьим Multicast Hello на каналах без потерь.

Интервал Update
: 4 интервала Multicast Hello.

Время удержания IHU
: 3,5 анонсированных интервала IHU.

Срок действия маршрута
: 3,5 анонсированных интервала обновления.

Тайм-аут для запросов
: Изначально 2 секунды с удвоением при каждом повторе запроса (до 3 раз).

Тайм-аут срочности
: 0,2 секунды.

Время сборки мусора для источника
: 3 минуты.

Приложение C. Фильтрация маршрутов

Фильтрация маршрутов представляет собой процедуру, где экземпляр протокола маршрутизации отбрасывает некоторые из анонсированных соседями маршрутов или проверяет их метрику, сравнивая с ожидаемой. Подобно другим протоколам на основе вектора удаленности, Babel может применять к изучаемым маршрутам произвольные фильтры и реализации Babel с разными наборами фильтров и правилами фильтрации могут взаимодействовать, не порождая маршрутных петель. Способность протокола фильтровать маршруты обусловлена расчетом метрики, описанным в параграфе 3.5.2 - Babel может применять любую метрику со строгой монотонностью, включая метрику с бесконечным значением для выбранного подмножества маршрутов (см. также параграф 3.8.1, где запросы для несуществующих маршрутов рассматриваются как запросы маршрутов с бесконечной метрикой).

В общем случае некорректно изучать маршруты с метрикой меньше анонсированного значения или менять префикс адресатов маршрута на более длинный, поскольку это может создавать постоянные маршрутные петли.

Фильтрация маршрутов является полезным инструментом, поскольку она позволяет точно настроить маршрутные решения, принимаемые протоколом. Поэтому некоторые реализации Babel поддерживают обширный язык настройки конфигурации, позволяющий задать набор правил фильтрации, например, по входному интерфейсу и префиксу.

Для ограничения последствий ошибочной настройки реализации Babel поддерживают по умолчанию разумный набор правил фильтрации, даже если они не позволяют настраивать фильтры пользователю. Как минимум, реализации отбрасывают маршруты с префиксом адресатов fe80::/64, ff00::/8, 127.0.0.1/32, 0.0.0.0/32, 224.0.0.0/8.

Приложение D. Расширения протокола

Протокол Babel является расширяемым и этот документ определяет множество механизмов, которые могут служить для расширения протокола с обеспечением совместимости с имеющимися версиями:

- увеличение номера версии в заголовке пакетов;
- определение новых TLV;
- определение новых суб-TLV (в битом обязательности или без него);
- определение новых AE;
- использование трейлеров в пакетах.

В этом приложении содержатся рекомендации для разработчиков расширений в части выбора кодирования.

Номер версии в заголовке Babel следует увеличивать лишь в тех случаях, когда не обеспечивается совместимости с более ранней версией протокола. Во многих случаях расширения можно реализовать путем определения новых TLV или добавления суб-TLV к имеющимся TLV. Например, расширение для добавления данных в обновления маршрутов можно реализовать путем «обогащения» Update TLV за счет необязательных или обязательных суб-TLV в Update TLV.

Различные кодировки по-разному обрабатываются реализациями, которым непонятно данное расширение. В случае новых TLV или суб-TLV с флагом обязательности не понимающая расширение реализация будет игнорировать TLV целиком, тогда как при нопонятных необязательных суб-TLV будет выполняться анализ и обработка TLV в целом с игнорированием непонятных суб-TLV. Поэтому для совместимых расширений Update TLV следует применять необязательные суб-TLV (расширение можно игнорировать), а обязательные суб-TLV или новые TLV должны применяться в расширениях, где непонятные TLV должны отбрасываться целиком.

Опыт показывает, что необходимость добавления данных обычно возникает в самых неожиданных местах. Поэтому рекомендуется определять новые TLV как самозавершающие, чтобы к ним можно было добавлять суб-TLV.

Добавление AE, по сути, эквивалентно определению нового TLV, поскольку Update TLV с неизвестным AE игнорируются подобно неизвестным TLV. Однако для добавление AE потребуется больше усилий, поскольку это создает новый набор состояний сжатия. А поскольку Next Hop TLV создает состояние, связанное с данным семейством адресов, а не данным AE, новое кодирование адресов AE для ранее определенного семейства даресов недопустимо применять в Next Hop TLV, если нужна совместимость с прежними версиями. Похожая проблема возникает для Update TLV с неизвестными AE, задающими новое значение router-id (в результате установки флага Router-Id). Поэтому определять новые AE нужно очень осторожно, если нужна совместимость с имеющимися реализациями.

Трейлер пакетов предназначен для использования криптографических подписей, охватывающих лишь тело пакета. Сохранение криптографической подписи в трейлере предотвращает сброс подписи перед расчетом хэш-значения для тела пакета и позволяет проверять подпись до запуска полного анализатора TLV с учетом состояний. Поэтому в трейлере следует размещать лишь TLV, для которых не требуется криптографическая защита. Такие TLV должны быть просты для анализа и, в частности, не требовать учета состояний при анализе.

Приложение E. Реализации-заглушки

Протокол Babel достаточно экономичен. Обновление для одного адресата занимает от 12 до 40 октетов в зависимости от семейства адресов, а средний размер составляет менее 24 октетов. Запись в таблице маршрутов для IPv6 занимает около 35 октетов. Таким образом, в один стандартный кадр Ethernet можно поместить около 65 обновлений а мегабайт памяти позволяет записать таблицу маршрутов с 20000 записей и связанную с ней таблицы источников.

Babel также является довольно простым протоколом и его полная реализация включает меньше 12000 строк C, а размер двоичного файла - меньше 120 Кбайт для 32-битовой архитектуры CISC, из которых около половины занимает код расширений и пользовательского интерфейса. Тем не менее, в некоторых средах с очень большими ограничениями, таких как КПК (PDA), микроволновые печи и т. п., может оказаться желательным реализовать лишь часть протокола.

Есть много разных определений маршрутизатора-заглушки, но здесь таким устройством считается реализация Babel, анонсирующая один или несколько подключенных напрямую префиксов в сети Babel, не не реанонсирующая полученные от соседей маршруты и всегда предпочитающая прямой маршрут к подключенному напрямую префиксу маршруту, полученному от протокола Babel, даже при одинаковых префиксах. Заглушка может поддерживать полную таблицу маршрутов или просто выбрать принятый по умолчанию маршрут через одного из соседей, анонсирующих такой маршрут. Поскольку заглушки не пересылают пакеты, не связанные с подключенными напрямую каналами, они не могут участвовать в маршрутных петлях и поэтому для них не нужно проверять условие выполнимости или поддерживать таблицу источников.

Независимо от степени упрощения, реализация-заглушка должна анализировать суб-TLV во всех понятных TLV и проверять флаг обязательности. Она должна отвечать на запросы подтверждений и участвовать в обмене Hello и IHU, а также должна быть способна отвечать на запросы seqno для анонсируемых маршрутов и на запросы маршрутов. Опыт показывает, что реализация-заглушка с поддержкой лишь IPv6 может иметь менее 1000 строк кода C и компилироваться в 13 Кбайт исполняемого кода для 32-битовой архитектуры CISC.

Приложение F. Совместимость с предыдущими версиями

Определенный в этом документе протокол является преемником протокола, заданного [RFC6126] и [RFC7557]. Хотя эти протоколы не совместимы полностью, новый протокол можно развернуть в сетях RFC 6126 без остановки работы.

Протокол имеет 3 необязательный свойства, делающих его несовместимым с предшественником. Во-первых, RFC 6126 не требует Unicast Hello (параграф 3.4.1) и реализация RFC 6126 будет ошибочно воспринимать Unicast Hello как Multicast Hello. Поскольку для этих сообщений применяются разные порядковые номера, передача Unicast Hello реализации RFC 6126 будет приводить к сбоям при оценке качества канала. Во-вторых, RFC 6126 не определяет неплановых Hello и реализация RFC 6126 будет неверно трактовать Hello с интервалом 0. В-третьих, RFC 7557 не определяет обязательных суб-TLV (параграф 4.4), поэтому реализации RFC 6126 и RFC 7557 будут некорректно игнорировать TLV с неизвестными обязательными суб-TLV, что может нарушать корректность маршрутизации.

Реализацию данного документа, которая не будет передавать Unicast и неплановые Hello, а также поддерживать какие-либо расширения с обязательными суб-TLV, можно без опаски развертывать в сети, где некоторые узлы используют протокол, определенный в RFC 6126 и RFC7557.

При внесении двух изменений в реализации RFC 6126 и RFC 7557 они смогут безопасно взаимодействовать с реализацией данной спецификации. Во-первых, нужно игнорировать или обрабатывать незапланированные и Unicast Hello. Во-вторых, нужно анализировать суб-TLV во всех понятных TLV и игнорировать TLV с неизвестными обязательными суб-TLV. Разбор неизвестных TLV не требуется и они просто игнорируются.

Ниже другие изменения, которые не препятствуют взаимодействию:

- смягчены условия восприятия маршрутов (параграф 3.5.3);
- при выборе маршрута больше не требуется его порядковый номер (параграф 3.6);
- определен формат трейлера в пакетах (параграф 4.2);
- запрещены router-id, содержащие только 0 или только 1 (параграф 4.1.3);
- состояние сжатия зависит от семейства адресов, а не от формата кодирования (параграф 4.5);
- рекомендуется задавать темп отправки пакетов (параграф 3.1).

Благодарности

Много людей внесло свой вклад в текст и идеи для этой спецификации. Авторы особо признательны Matthieu Boutier, Gwendoline Chouasne, Margaret Cullen, Donald Eastlake, Toke Høiland-Jørgensen, Benjamin Kaduk, Joao Sobrinho и Martin Vigoureux. Предыдущая версия спецификации [RFC6126] в значительной мере основана на вкладе Joel Halpern. Метод сжатия адресов был заимствован из [PACKETBB].

Адреса авторов

Juliusz Chroboczek

IRIF, University of Paris-Diderot

Case 7014

75205 Paris CEDEX 13

France

Email: jch@irif.fr

David Schinazi

Google LLC

1600 Amphitheatre Parkway

Mountain View, California 94043

United States of America

Email: dschinazi.ietf@gmail.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru