Internet Research Task Force (IRTF)
Request for Comments: 8569
Category: Experimental
ISSN: 2070-1721

M. Mosko
PARC, Inc.
I. Solis
LinkedIn
C. Wood
University of California Irvine

July 2019

Content-Centric Networking (CCNx) Semantics

Семантика ориентированных на содержимое сетей (CCNx)

Аннотация

Этот документ описывает базовые концепции архитектуры ориентированных на содержимое сетей (CCNx[1]) и представляет сетевой протокол, основанный на двух сообщениях с объектами Interests и Content. Документ задаёт набор обязательных и опциональных полей в этих сообщениях, а также описывает их поведение и интерпретацию. Эта архитектура и протокол не зависят от представления сигналов в линиях передачи.

Протокол также применяет управляющие сообщения Interest Return, посредством которых одна система может возвращать сообщения Interest на предыдущий интервал пересылки (hop) при возникновении ошибки. Это указывает предыдущему интервалу, что текущая система не будет отвечать на Interest.

Документ является результатом работы исследовательской группы ICNRG[2]. Документ был рассмотрен множеством участников ICNRG. Две реализации активно применяются и показывают техническую зрелость спецификации протокола.

Статус документа

Документ не является спецификацией стандартного протокола Internet (Standards Track) и публикуется для проверки, экспериментальной реализации и оценки.

Документ определяет экспериментальный протокол для сообщества Internet. Документ является результатом работы IRTF[3]. IRTF публикует результаты связанных с Internet исследований и разработок. Эти результаты могут оказаться не подходящими для развёртывания. Данный RFC представляет согласованное мнение группы ICNRG в составе IRTF. Документ одобрен для публикации IRSG и не претендует на роль стандарта Internet (см. раздел 2 в RFC 7841).

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc8569.

Оглавление

[1]Content-Centric Networking.
[2]Information-Centric Networking Research Group - группа по исследованию ориентированных на информацию сетей.
[3]Internet Research Task Force.

1. Введение

Этот документ описывает принципы архитектуры CCNx. Описан протокол, применяющие иерархические имена для пересылки запросов и сопоставления откликов с запросами. Протокол не использует адресов конечных точек (таких, как IP). Ограничения в запросе могут лимитировать отклик открытым ключом подписавшей стороны или криптографическим хэшем отклика. Каждый узел пересылки CCNx в пути проверяет соответствие имён и ограничения. Протокол CCNx вписывается в более широкую модель протоколов ориентированных на информацию сетей (ICN[1]) [RFC7927]. Этот документ посвящён семантике протокола и не зависит от представления данных в линии. В документе, посвящённом сообщениям CCNx [RFC8609], описано представление TLV[2] для сигналов в линии. В этом параграфе рассмотрены основные концепции CCNx, которые более подробно описаны в оставшейся части документа.

Протокол CCNx основан на ранних работах ICN, выполненных Jacobson и др. [nnc]. Версия Jacobson для CCNx известна как 0.x (CCNx 0.x), а настоящая работа имеет версию 1.0 (CCNx 1.0). Имеются две активные реализации CCNx 1.0. Наиболее полной является Community ICN (CICN) [cicn] - проект Linux Foundation, размещённый на fd.io. Другой активной реализацией является CCN-lite [ccn-lite], поддерживающая системы IoT[3] и операционную систему RIOT. CCNx 0.x стал основой для университетского проекта NDN[4] [ndn].

Текущая спецификация CCNx 1.0 отличается от CCNx 0.x в нескольких значимых областях. Наиболее явное поведенческое различие между CCNx 0.x и CCNx 1.0 состоит в том, что CCNx 1.0 имеет более простую обработку откликов. В обоих версиях пересылающие узлы используют иерархическое сопоставление самого длинного совпадающего префикса с информационной базой пересылки (FIB[5]) для отправки запроса через сеть в систему, которая может выдать ответ. Пересылающий узел должен затем сопоставить имя отклика с именем запроса для определения обратного пути и доставить отклик запрашивающему. В CCNx 0.x имя Interest может быть иерархическим префиксом имени отклика, который позволяет обнаруживать содержимое на уровне 3 (L3). В CCNx 1.0 имя отклика должно совпадать с именем запроса. Обнаружение содержимого выполняет протокол вышележащего уровня.

Протокол селекторов CCNx Selectors [selectors] является примером использования протокола вышележащего уровня над CCNx 1.0 L3 для обнаружения содержимого. Протокол селекторов использует метод, похожий на исходные методы CCNx 0.x, не требуя частичного сопоставления имён откликов и запросов в пересылающем узле.

Этот документ представляет согласованную точку зрения группы ICNRG. Это первый протокол ICN исследовательской группы, созданный на основе раннего протокола CCNx [nnc] с существенным пересмотром и вкладом сообщества ICN и членов исследовательской группы. Документ прошёл критический обзор нескольких членов сообщества ICN и исследовательской группы. Авторы и руководитель исследовательской группы одобрили документ. Документ поддержан IRTF, выпущен без участия IETF и не является стандартом IETF. Это экспериментальный протокол, который может не подойти для конкретных применений. Спецификация в будущем может измениться.

1.1. Уровни требований

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

1.2. Архитектура

Опишем архитектуру сети, в которой работает CCNx и введём некоторые термины из [terminology]. Подробное описание всех компонент и грамматика сообщений рассмотрены в разделе 2.

[1]Information-Centric Networking.
[2]Type-length-value - тип, размер, значение.
[3]Internet of Things - Internet вещей.
[4]Named Data Networking - сеть именованных данных.
[5]Forwarding information base.

Производителем (producer) или издателем (publisher) является конечная точка, которая инкапсулирует содержимое в объекты Content для транспортировки в сеть CCNx. Издатель имеет открытый и секретный ключ, а также подписывает (напрямую или опосредованно) объекты Content. Обычно идентификатор открытого ключа издателя KeyId (хэш открытого ключа) широко известен или может быть выведен из пространства имён издателя стандартным путём.

Издатель оперирует в одном или нескольких пространствах имён, которые являются префиксами имён, представленными в информационных базах пересылки (FIB). Это позволяет запросу достичь издателя и получить отклик (если он существует).

Таблица FIB говорит пересылающему узлу, куда направить запрос. Она может указывать локальное приложение, локальный кэш или хранилище содержимого (Content Store), а также удалённую систему. Если в FIB не найдено соответствующей записи, узел пересылки не может обработать запрос. Детали правил сопоставления имён с FIB приведены в параграфе 2.4.4. Конечная точка имеет FIB, хотя это может быть простой маршрут, принятый по умолчанию. Промежуточные системы (т. е. маршрутизаторы) обычно имеют более крупные базы FIB. Центральный узел пересылки CCNx, например, будет знать все глобальные маршруты.

Потребителем является конечная точка, запрашивающая имя. Этот документ не описывает способов выяснения имён или KeyId издателей потребителями - протокол вышележащего уровня, работающий на основе CCNx, выполняет функции машины поиска и служб поиска или общеизвестных имён. Потребитель создаёт запрос, называемый Interest (интерес), и пересылает его через FIB конечной точки. Потребитель должен получить ответ на запрос (объект Content), соответствующий Interest, или управляющее сообщение (Interest Return), указывающее невыполнимость запроса.

Имеется 3 способа обнаружения ошибок при обработке Interest. Interest Return представляет собой управляющее сообщение, которое указывает ошибку низкого уровня, такую как отсутствие маршрута (no route) или нехватку ресурсов (out of resources). Если Interest приходит к издателю, но у того нет запрошенного содержимого, издателю следует передать зависящее от приложения сообщение об ошибке (например, not found - не найдено). Наконец, потребитель может не получить ничего (тайм-аут) и, в зависимости от приложения, ему следует повторить запрос или вернуть ошибку приложению.

1.3. Обзор протокола

Целью CCNx является именование содержимого и его извлечение из сети без привязки к конкретной конечной точке. Система маршрутизации (задаётся отдельно) заполняет таблицы FIB на каждом маршрутизаторе CCNx иерархическими префиксами имён, которые указывают издателей содержимого для данного префикса. Запрос находит соответствующее содержимое по этим путям и в этом случае отклик возвращает данные. Если совпадения не найдено, управляющее сообщение говорит об отказе. Запрос может дополнительно уточнять подходящие отклики путём. ограничений для подписывающей отклик стороны и криптографического хэширования отклика. Детали таких ограничений рассмотрены ниже.

Имя CCNx представляет собой иерархическую последовательность сегментов имён, каждый из которых имеет тип и может иметь значение. Сопоставление двух имён выполняется путём. посегментного двоичного сравнения типа и значения. Удобная для человека форма определена по схеме URI "ccnx:" [ccnx-uri], хотя каноническое кодирование имён представляет собой последовательность пар (тип, строка октетов). К сегментам имён не предъявляется требований удобочитаемости или кодировки UTF-8. Первые несколько сегментов имени будут сопоставляться с FIB и протокол маршрутизации может вносить свои ограничения для компонент маршрутизируемого имени (например, максимальный размер или правила кодирования символов). В принципе сегменты имён и имена могут иметь неограниченный размер, хотя на практике они ограничиваются кодированием в линии и практическими соображениями протоколов маршрутизации. Отметим, что в CCNx для сегментов имён применяется двоичное сравнение, тогда как в URI - имена хостов сравниваются без учёта регистра символов (обусловлено DNS).

Имя CCNx в используемой узлом пересылки форме намеренно сохранено как базовый тип с октетным кодированием типа и значения без каких-либо требований к удобочитаемости для человека и кодировке символов. Причина этого заключается в том, что нас интересует способ обработки имён узлом пересылки. Предполагается, что приложения, протоколы маршрутизации и вышележащие уровни будут использовать свои соглашения и ограничения для разрешённых типов и значений сегментов имён.

CCNx является протоколом запросов и откликов, выбирающим блоки данных (chunk) по именам. Целостность каждого блока может быть подтверждена напрямую с помощью цифровой подписи или кода MAC[1], а также опосредованно с помощью хэш-цепочек. Блоки также могут включать более слабые коды MIC[2] или совсем не использовать защиты целостности. Поскольку информация об источнике передаётся в каждом блоке (или более крупном блоке с защитой целостности), больше не нужно полагаться на идентификацию хоста, такую как сертификат TLS, для подтверждения легитимности блока. Таким образом, целостность данных является основным свойством CCNx и не зависит от канала передачи данных. Имеется несколько вариантов защиты конфиденциальности, рассмотренных ниже.

Этот документ определяет лишь базовые свойства имён CCNx. В некоторых изолированных средах пользователи CCNx могут применять любые имена по своему выбору и даже внедрять такие имена (или префиксы) в протокол маршрутизации или использовать иные методы отыскания информации. В среде Internet будут применяться правила для формата имён и назначения имён издателями, хотя здесь эти правила не указаны.

Важная концепция CCNx заключается в том, что субъективное имя криптографически связывается с фиксированными данными (payload). Эти создаваемые издателями привязки можно проверить криптографическими методами. Именованные данные в результате являются кортежами {{Name, ExtraFields, Payload, ValidationAlgorithm}, ValidationPayload}, где все поля внутреннего кортежа учитываются в проверочных полях (например, в подписи). Потребители данных могут проверить целостность этих блоков путём. выполнения расчета тех же криптографических хэш-значений и сравнения с цифровыми подписями в ValidationPayload.

В дополнение к цифровым подписям (например, RSA) CCNx поддерживает коды проверки подлинности (например, HMAC[3]) и целостности (например, CRC[4]). Для поддержки криптографических привязок следует применять по меньшей

[1]Message Authentication Code - код аутентификации сообщения.
[2]Message Integrity Code - код целостности сообщения.
[3]Hashed Message Authentication Code - хешированный код аутентификации сообщения.
[4]Cyclic Redundancy Check - циклическая контрольная сумма с избыточностью.

мере одну подпись или код аутентификации, но это требуется не для всех объектов. Например, первый объект с подписью может охватывать другие объекты с помощью хэш-цепочки, дерева Merkle или подписанного манифеста. Последующие объекты могут не иметь каких-либо проверок и полагаться лишь на ссылки. Код целостности (например, CRC) предназначен для обнаружения случайных повреждений в Interest.

CCNx задаёт сетевой протокол для объектов Interest (запросные сообщения) и Content (отклики) для переноса именованных данных. Interest включает поле Name, которое указывает желаемый отклик, и необязательные ограничения по соответствию, определяющие подходящие объекты Content. Имеется два ограничения - для идентификатора ключа (KeyIdRestr) и хэш-значения объекта Content (ContentObjectHashRestr). Первое ограничение для KeyId позволяет отобрать отклики с полем ValidationAlgorithm KeyId, соответствующим заданному значению. Второе ограничение позволяет выбрать отклики, в которых криптографическое хэш-значение совпадает с заданным ограничением. В разделе 9 описано применение этих ограничений при сопоставлении объектов Content с Interest.

Иерархия имён CCNx служит для маршрутизации по максимальному соответствию префиксов в узлах пересылки. Максимальным совпадением префиксов является рассчитанный сегмент имени за сегментом в иерархическом имени, где все сегменты совпадают. Глобальной маршрутизации префиксов не требуется. В рамках развёртывания. может применяться любая локальная маршрутизация, даже если она использует один плоский (без иерархии) сегмент имени.

Другая концепция CCNx заключается в наличии баланса между потоками сообщений Interest и Content Object. На уровне сети сообщение Interest, проходящее по одному пути, должно вызывать не более одного отклика Content Object. Если некоторые узлы передают Interest по разным путям, таким узлам следует объединять отклики, чтобы к запрашивающему приходил лишь один поток Content Object. При передаче Interest в широковещательном или групповом режиме, отправителю следует быть готовым к множеству откликов, если не применяется зависимый от среды механизм, такой как подавление «болтовни» или выбор ведущего.

По мере перемещения Interest в направлении следующей таблицы FIB устанавливается состояние на каждом узле пересылки, так что ответ Content Object может возвращаться к исходному запрашивающему (запрашивающим) без необходимости включения запрашивающей стороной маршрутизируемого адреса возврата. Применяется смысловая таблица PIT[1] в качестве метода хранения состояний, упрощающих возврат Content Object.

Таблица PIT хранит последний интервал (hop) для Interest, а также поле Name и необязательные ограничения. Эти данные нужны для сопоставления Content Object с Interest (см. раздел 9). При поступлении Content Object он сопоставляется с PIT для определения подходящей записи. Для каждой такой записи передаётся не более одного Content Object в каждый указанный в таблице PIT последний интервал.

Реальные таблицы PIT не задаются данной спецификацией. Реализация может применять любой метод, обеспечивающий такое же внешнее поведение. Имеются исследовательские работы, где применялись методы, подобные коммутации по меткам в некоторых частях сети для снижения на узлах числа состояний, создаваемых PIT [dart]. Некоторые реализации хранят состояния PIT в FIB, не используя второй таблицы.

Если на узел приходит множество Interest с одинаковыми {Name, [KeyIdRestr], [ContentObjectHashRestr]} до прибытия Content Object, соответствующего первому Interest, они группируются в одну запись PIT и их последние интервалы (hop) агрегируются (см. параграф 2.4.2). Таким образом, один объект Content может соответствовать множеству ожидающих Interest в таблице PIT.

В CCNx протоколы вышележащего уровня часто называют протоколами на основе имён (name-based), поскольку они оперируют именами CCNx. Например, протокол управления версиями может добавлять сегменты имени для передачи состояния, связанного с версией содержимого. Протокол обнаружения содержимого может добавлять свои сегменты имени к префиксу для поиска содержимого с таким префиксом. Таких протоколов может быть много и они могут применять к именам свои правила. Протоколы могут образовывать несколько уровней с инкапсуляцией (слева) префикса имени верхнего уровня.

Этот документ описывает также управляющее сообщение Interest Return. Элемент сети может вернуть сообщение Interest Return на предыдущий интервал, если при обработке Interest возникла ошибка. Возвращённое сообщение может быть дополнительно обработано предыдущим интервалом или возвращено в направлении источника Interest. Когда узел возвращает Interest Return, он указывает, что предыдущему интервалу не следует ожидать от него отклика для Interest, т. е. на возвращающем узле нет записи PIT для Content Object.

Есть много способов описания более крупных объектов в CCNx. Агрегирование L3 Content Object в более крупные объекты выходит за рамки документа. Один из предложенных методов FLIC[2] [flic] использует манифест для перечисления фрагментов большого объекта. Сами манифесты являются объектами Content. Другим вариантом является применение соглашений в именах Content Object как в протоколе CCNx Chunking [chunking], где большой объект дробится на мелкие блоки и каждый такой блок получает специальную компоненту имени, указывающую его порядковый номер. Один из экспериментальных протоколов фрагментирования BeginEnd Fragments [befrags] использует метод в стиле многоточечного PPP для работы через интерфейсы L2 с указанием сообщений CCNx [RFC8609] в форме TLV для кодирования в линии.

На основе этих концепций в оставшейся части документа определяется поведение узлов пересылки при обработке сообщений Interest, Content Object и Interest Return.

2. Протокол

В этом разделе определена грамматика сообщений CCNx (Interest, Content Object, Interest Return). Затем описано типичное поведение потребителей, издателей и узлов пересылки. В параграфе, посвящённом пересылке, подробно описана обработка относящихся к пересылке вопросов, таких как HopLimit и Content Store, а также конвейеры обработки сообщений Interest и Content Object .

[1]Pending Interest Table - таблица ожидающих Interest.
[2]File-Like ICN Collection - файлоподобная коллекция ICN.

2.1. Грамматика сообщений

ABNF-грамматика [RFC5234] для сообщений CCNx показана на рисунке 1. Грамматика не включает разграничителей кодирования, таких как TLV. Кодирование в линиях представлено в отдельном документе. При наличии раздела Validation поле Validation Algorithm охватывает данные от Body (BodyName или BodyOptName) до конца раздела ValidationAlg. Поля InterestLifetime, CacheTime и Return Code не учитываются для проверки и могут быть изменены.

HashType, PayloadType и Private Enterprise Number (PEN) должны соответствовать значениям IANA в регистрах CCNx Hash Function Types, CCNx Payload Types [ccnx-registry] и Private Enterprise Numbers [eprise-numbers], соответственно.

Ниже представлены определения полей в алфавитном порядке.

AbsTime
Абсолютное время в виде 64-битового значения UTC в миллисекундах с начала эпохи (стандартное время POSIX).

CacheTime
Абсолютное время, при котором издатель будет считать кэшированный Content Object малоценным. Этот параметр рекомендуется для систем кэширования (см. раздел 4).

Cert
Некоторые приложения могут захотеть встраивать сертификат X.509 для проверки и обеспечения доверенной привязки. Cert представляет собой сертификат X.509 с кодированием DER.

ConObjField
Необязательные поля, которые могут присутствовать в объектах Content.

ConObjHash
Значение Content Object Hash, рассчитываемое с помощью SHA256-32 для сообщения от начала тела до конца сообщения. Отметим, что область охвата отличается от ValidationAlg. Значению **не следует** доверять за пределами домена (см. раздел 5).

ContentObjectHashRestr
Ограничение Content Object Hash. Объект Content должен хэшироваться до того же значения, что и ограничение, с использованием того же HashType. В ContentObjectHashRestr **должно** использоваться хэширование SHA256-32.

ExpiryTime
Абсолютное время, при котором Content Object следует считать устаревшим (см. раздел 4).

Hash
Хэш-значения в сообщении содержат HashType для указания алгоритма, использованного при генерации хэш-значения, за которым следует это значение. Такой подход позволяет обеспечить быстрое хэширование. Некоторые поля могут требовать определённого HashType.

HashType
Алгоритм, используемый для расчета свёртки (hash), который должен соответствовать одному из перечисленных в реестре IANA CCNx Hash Function Types [ccnx-registry].

HopLimit
Сообщения Interest могут попадать в петли, если они имеются на уровне пересылки. Для устранения влияния таких петель в каждом Interest содержится поле HopLimit, декрементируемое на каждом интервале. При достижении 0 сообщение не будет пересылаться. См. параграф 2.4.

InterestField
Необязательные поля, которые могут присутствовать в сообщении Interest.

KeyId
Идентификатор ключа, используемого в ValidationAlg. См описание Validation (раздел 8), где рассматривается использование ключей для кодов MAC и подписей.

KeyIdRestr
Ограничения KeyId. Объект Content должен иметь KeyId, значение которого совпадает с ограничением.

KeyLink
Канал (Link, см. параграф 6), который указывает, как получить ключ для проверки ValidationPayload (см. раздел 8).

Lifetime
Приблизительное время, в течение которого запрашивающий готов ждать отклик, обычно задаваемое в секундах. Оно не связано жёстко с временем кругового обхода в сети, но должно быть больше.

Name
Имя состоит из непустого первого сегмента, за которым могут следовать дополнительные сегменты возможно с нулевым размером. Сегменты имени являются не анализируемыми строками октетов и при использовании кодировки UTF-8 учитывается регистр символов. Объект Interest **должен** иметь имя (Name), Content Object **может** иметь Name (см. раздел 9). Сегменты имени называют полными, если они однозначно указывают один объект Content. Имя является точным, если его сегменты полны. Interest с полным именем - это объект, который указывает точное имя и ограничение Content Object Hash соответствующего Content Object.

Payload
Данные сообщения в соответствии с PayloadType.

PayloadType
Формат поля Payload. При отсутствии типа предполагается тип Data (T_PAYLOADTYPE_DATA) [ccnx-registry], означающий не анализируемые байты данных приложения. Тип Key (T_PAYLOADTYPE_KEY [ccnx-registry]) указывает DER-кодированный открытый ключ или сертификат X.509. Тип Link (T_PAYLOADTYPE_LINK [ccnx-registry]) указывает один или множество каналов (Link, см. раздел 6).

PublicKey
Некоторые приложения могут захотеть встраивать в сообщение открытый ключ, используемый для проверки подписи. PublickKey использует кодирование DER.

RelTime
Относительное время в миллисекундах.

ReturnCode
Указывает причины возврата сообщения Interest на предыдущий интервал (см. параграф 10.2).

SigTime
Абсолютное время создания подписи (UTC, в миллисекундах). Время подписи относится только к алгоритму проверки и не обязательно указывает момент создания проверяемого сообщения.

Vendor

Зависящие от производителя не анализируемые (opaque) данные. Данные Vendor включают IANA Private Enterprise Number [eprise-numbers], за которым следует заданная производителем информация. CCNx разрешает зависящие от производителя данные во многих местах.

```
Message       = Interest / ContentObject / InterestReturn
Interest      = IntHdr BodyName [Validation]
IntHdr        = HopLimit [Lifetime] *Vendor
ContentObject = ConObjHdr BodyOptName [Validation]
ConObjHdr     = [CacheTime / ConObjHash] *Vendor
InterestReturn= ReturnCode Interest
BodyName      = Name Common
BodyOptName   = [Name] Common
Common        = *Field [Payload]
Validation    = ValidationAlg ValidationPayload

Name          = FirstSegment *Segment
FirstSegment  = 1*OCTET / Vendor
Segment       = *OCTET / Vendor

ValidationAlg = (RSA-SHA256 / EC-SECP-256K1 / EC-SECP-384R1 /
                 HMAC-SHA256 / CRC32C) *Vendor
ValidationPayload = 1*OCTET
PublicAlg     = KeyId [SigTime] [KeyLink] [PublicKey] [Cert]
RSA-SHA256    = PublicAlg
EC-SECP-256K1 = PublicAlg
EC-SECP-384R1 = PublicAlg
HMAC-SHA256   = KeyId [SigTime] [KeyLink]
CRC32C        = [SigTime]

AbsTime       = 8OCTET ; 64-bit UTC msec since epoch
CacheTime     = AbsTime
ConObjField   = ExpiryTime / PayloadType
ConObjHash    = Hash
ExpiryTime    = AbsTime
Field         = InterestField / ConObjField / Vendor
Hash          = HashType 1*OCTET
HashType      = 2OCTET ; IANA "CCNx Hash Function Types"
HopLimit      = OCTET
InterestField = KeyIdRestr / ContentObjectHashRestr
KeyId         = Hash
KeyIdRestr    = Hash
KeyLink       = Link
Lifetime      = RelTime
Link          = Name [KeyIdRestr] [ContentObjectHashRestr]
ContentObjectHashRestr  = Hash
Payload       = *OCTET
PayloadType   = OCTET ; IANA "CCNx Payload Types"
PublicKey     = *OCTET ; DER-encoded public key
Cert          = *OCTET ; DER-encoded X.509 Certificate
RelTime       = 1*OCTET ; msec
ReturnCode    = OCTET ; см. параграф 10.2
SigTime       = AbsTime
Vendor        = PEN *OCTET
PEN           = 1*OCTET ; IANA "Private Enterprise Number"
```

Рисунок 1. ABNF-грамматика сообщений CCNx.

2.2. Поведение потребителя

Для запроса части содержимого данного кортежа {Name, [KeyIdRest], [ContentObjectHashRestr]} потребитель создаёт сообщение Interest с указанными значениями. Он **может** добавить раздел validation (обычно только CRC32C). Потребитель **может** включить поле Payload в сообщение Interest для отправки издателю в дополнение к имени. Имя служит для маршрутизации и может запоминаться на каждом интервале в смысловой таблице PIT для облегчения возврата Content Object. Сохранение большого числа состояний в имени ведёт к высокому расходу памяти. Поскольку данные (payload) не рассматриваются при пересылке Interest или сопоставлении Content Object с Interest, потребителю **следует** включать Interest Payload ID (см. параграф 3.2) как часть имени, чтобы позволить пересылающему узлу сопоставить Interest с Content Object и предотвратить объединение Interest с разными данными. Точно также, если потребитель применяет MAC или подпись, ему **следует** включать уникальный сегмент как часть имени для предотвращения агрегирования Interest с другими сообщениями Interest или его удовлетворение объектом Content, не связанным с проверкой пригодности.

Потребителю **следует** указать время InterestLifetime, в течение которого он готов ждать отклика. Значение InterestLifetime определяется приложением, а не временем кругового обхода в сети (см. параграф 2.4.2). По умолчанию применяется InterestLifetime = 2 сек.

Потребителю **следует** установить подходящее значение Interest HopLimit или использовать принятое по умолчанию ограничение (255). Если потребителю известно число интервалов маршрутизации до издателя, **следует** указать его.

Потребитель передаёт Interest своему первому узлу пересылки, который затем пересылает Interest через сеть к издателю (или реплике), где запрос может быть выполнен по имени (см. параграф 2.4).

Сообщения Interest не обеспечивают надёжной доставки. Потребителю **следует** применять транспортный протокол, который будет повторять безответные сообщения Interest, пока не пройдёт время InterestLifetime. Транспортный протокол этим документом не задаётся.

Сеть **может** передать потребителю сообщение Interest Return, указывающее невозможность выполнить Interest. Значение ReturnCode указывает причину отказа, такую как отсутствие маршрута или перегрузка. В зависимости от ReturnCode потребитель **может** повторить Interest или вернуть ошибку запрашивающему приложению.

Если содержимое найдено и возвращено первому узлу пересылки, потребитель получит Content Object. Для проверки принятого Content Object потребитель использует приведённый ниже набор тестов.

- Потребитель **должен** убедиться в корректности формата Content Object.
- Потребитель **должен** проверить, что полученный Content Object соответствует одному или нескольким Interest, как описано в разделе 9.
- Если Content Object подписан, потребителю **следует** криптографически проверить подпись в соответствии с разделом 8. При отсутствии нужного ключа потребителю **следует** получить его (например, от службы распространения ключей или через KeyLink).
- Если подпись содержит SigTime, потребитель **может** использовать это значение для проверки действия подписи. Например, если потребитель запрашивает динамически создаваемое содержимое, ему следует ожидать, что значение SigTime будет не раньше времени создания Interest.
- Если Content Object подписан, потребителю **следует** подтвердить доверие к ключу подписи для пространства имён. Эта процедура выходит за рамки документа, хотя можно использовать традиционные методы PKI, службы распространения доверенных ключей или методы, подобные [trust].
- Потребитель **может** кэшировать Content Object для последующего использования вплоть до ExpiryTime, если это значение присутствует.
- Потребитель **может** воспринять просроченный Content Object из линии. Срок действия пакета Content Object может завершиться в процессе передачи, если узлы пересылки не отбрасывают просроченные пакеты «на лету». Единственным требованием для просроченных объектов является **недопустимость** их размещения в Content Store или кэше, а издателям **недопустимо** отправлять просроченные Content Object.

2.3. Поведение издателя

Этот документ не задаёт методов заполнения именами таблиц FIB на узлах пересылки (см. параграф 2.4). Издатель настраивает один или множество префиксов имён, с которыми он может создавать содержимое, или выбирает префиксы имён и информирует уровень маршрутизации для анонсирования этих префиксов.

При получении издателем сообщения Interest ему **следует** выполнить указанные ниже действия.

- Убедиться, что Interest является частью пространства имён издателя.
- Если в Interest есть раздел Validation, нужно проверить его в соответствии с разделом 8. Обычно Interest включают лишь CRC32C, если приложение издателя явно не указывает восприятие других вариантов. Издатель **может** отбрасывать Interest с разделом Validation, если приложение издателя не ожидает таких подписей, поскольку это может быть формой атаки на отказ в обслуживании. Если подпись требует ключа, которого нет у издателя, ему **не рекомендуется** получать ключ из сети, когда это не является частью ожидаемого поведения приложения.
- Нахождение или создание Content Object и возврат его на предыдущий интервал Interest. Если запрошенное содержимое не может быть возвращено, издателю **следует** ответить сообщением Interest Return или Content Object с данными, говорящими о недоступности содержимого. В такой объект Content следует включать малое значение ExpiryTime (в будущем) или 0 (чтобы его не кэшировали).

2.4. Поведение пересылающего узла

Пересылающий узел маршрутизирует сообщения Interest на основе таблицы пересылки (FIB), возвращает Content Object, соответствующие Interest, на предыдущий интервал Interest и обрабатывает управляющие сообщения Interest Return. Он может также сохранять кэш сообщений Content Object в смысловой таблице Content Store. Этот документ не задаёт внутреннего поведения узла пересылки, определяя лишь внешние характеристики.

В этом документе применяется два конвейера обработки - один для Interest, другой для Content Object. Обработка Interest состоит в проверке наличия дубликатов Interest в PIT (см. параграф 2.4.2), проверке наличия кэшированного Content Object в Content Store (см. параграф 2.4.3) и пересылке Interest на основе FIB. Обработка Content Store заключается в сопоставлении Interest с таблицей PIT и пересылке на предыдущий интервал.

2.4.1. Interest HopLimit

CCNx не предотвращает петель Interest. Попавшие в петлю сообщения Interest в конечном итоге отбрасываются с использованием поля HopLimit, которое декрементируется на каждом этапе пересылки Interest.

Петли могут также разрываться при агрегировании Interest с предыдущей записью PIT внутри петли. В этом случае Content Object будет отправлен назад по петле и может вернуться на узел, который уже переслал содержимое, где он скорей всего не будет иметь записи в PIT и будет отброшен. Может оказаться так, что новое или другое попавшее в петлю сообщение Interest вернётся на тот же узел и в этом случае узел будет возвращать кэшированный отклик для создания новой записи PIT, как описано ниже.

HopLimit является последним средством устранения петель Interest, когда Content Object проходит в петле Interest, где промежуточные узлы по каким-то причинам больше не имеют записи в PIT и кэшированного Content Object.

Каждое сообщение Interest **должно** включать HopLimit. Сообщение Interest от локального приложения **может** иметь HopLimit = 0, что ограничивает Interest локальными источниками.

При получении Interest от другого узла пересылки значение HopLimit **должно** быть положительным - в противном случае пересылающий узел отбрасывает Interest. Пересылающий узел **должен** декрементировать HopLimit в Interest перед пересылкой по меньшей мере на 1.

Если после декрементирования HopLimit приобретает значение 0, сообщение Interest **недопустимо** пересылать другому узлу пересылки, но его **можно** передать локальному приложению издателя или обслужить из локального Content Store.

Рекомендуемый конвейер обработки HopLimit представлен ниже.

- Если сообщение Interest получено от удалённой системы:
  - при HopLimit = 0 сообщение Interest отбрасывается и можно передать Interest Return (HopLimit Exceeded);
  - в остальных случаях HopLimit уменьшается на 1.
- Выполняется обработка по правилам Content Store и Aggregation.
- Если сообщение Interest будет пересылаться:
  - при (возможно декрементированном) HopLimit = 0, пересылка ограничивается локальной системой;
  - в остальных случаях сообщение пересылается локальной или удалённой системе.

2.4.2. Агрегирование Interest

Агрегирование Interest происходит, когда пересылающий узел получает сообщение Interest, которое может быть удовлетворено ответом на другое сообщение Interest, уже пересланное этим узлом. Поэтому узел не пересылает новое сообщение Interest и лишь записывает дополнительный предыдущий интервал, чтобы Content Object, полученный в ответ на первое сообщение Interest, удовлетворил оба Interest.

CCNx использует правило агрегирования Interest, которое считает, что время InterestLifetime сродни времени подписки, а не кругового обхода в сети. В некоторых прежних правилах агрегирования срок действия считался временем кругового обхода и это создавало проблемы завершения срока действия Interest до прихода отклика, если значение RTT было недооценено или возникали конфликты со схемой автоматического повтора запросов (ARQ[1]), которая хотела повторить Interest, но для предыдущего Interest значение RTT было переоценено.

Узел пересылки **может** реализовать схему агрегирования Interest. Если это не делается, узел будет пересылать все сообщения Interest. Это не означает возврата множества (возможно идентичных) Content Object. Пересылающий узел **должен** по-прежнему выполнять все ожидающие запросы Interest, поэтому один Content Object может удовлетворять множеству похожих Interest, даже если узел пересылки не подавляет дубликаты Interest.

Рекомендуется приведённая ниже схема агрегирования Interest.

- Два сообщения Interest считаются «похожими», если у них совпадают Name, KeyIdRestr и ContentObjectHashRestr, а отсутствующие в одном сообщении поля отсутствуют и в другом.
- Разрешать совпадение двух смысловых значений InterestExpiry (локальные для пересылающего узла) со временем приёма + InterestLifetime (или зависимое от платформы значение по умолчанию, если поле не задано).
- Запись Interest (в таблице PIT) считается недействительной, если её время InterestExpiry в прошлом.
- Полученное первым сообщение Interest **должно** пересылаться.
- Второе и последующие сообщения Interest, похожие на действительное ожидающее сообщение Interest, от того же предыдущего интервала **должны** пересылаться (это считается повтором передачи запроса).
- Второе и последующие сообщения Interest, похожие на действительное ожидающее сообщение Interest, от другого предыдущего интервала **можно** агрегировать (не пересылать). Если это сообщение Interest имеет большее значение HopLimit, нежели ожидающее Interest, оно **должно** пересылаться.
- Агрегирование Interest **должно** продлевать время InterestExpiry для записи Interest. Реализация **может** сохранять одно значение InterestExpiry для всех предыдущих интервалов или сохранять InterestExpiry независимо для каждого интервала. В первом случае пересылающий узел может передать Content Object по пути, где объект уже не ждут и в этом случае предыдущий интервал (следующий для Content Object) будет отбрасывать его.

2.4.3. Поведение хранилища содержимого

Content Store представляет собой специальный кэш, являющийся частью узла пересылки CCNx. Это хранилище не является обязательным и служит для исправления повреждённых пакетов и обработки множественных (flash) запросов популярного содержимого. Хранилище может заполняться заранее или содержать кэшированные данные. Поскольку Content Store можно использовать для усиления атак с помощью «отправления кэша», для Content Store применяются специальные правила, указанные ниже.

1. Пересылающий узел **может** реализовать Content Store. В этом случае содержимое Content Store сопоставляется с Content Object для Interest по обычным правилам (см. раздел 9).
2. Если Interest имеет ограничение KeyId, хранилище Content Store **недопустимо** использовать для ответа при отсутствии уверенности в корректности подписи совпадающего Content Object. Это можно проверить с помощью внешнего источника (например, в управляемой сети или системе с заранее заполненным кэшем) или путём. получения открытого ключа для криптографической проверки подписи. От Content Store **не требуется** проверять подписи и отказ от предоставления непроверенного объекта просто считается его отсутствием.
3. Если Content Store выбирает проверку подписей, это **можно** сделать, как описано далее. Если открытый ключ указан в самом Content Object (в поле PublicKey field) или в сообщении Interest, хранилище Content Store **должно** проверить совпадение хэш-значения открытого ключа с KeyId, а затем проверить подпись (см. параграф 8.4). Content Store **может** проверять цифровую подпись Content Object перед его кэшированием, но

[1]Automatic Repeat reQuest.

это не требуется. Хранилищу Content Store **не следует** получать ключи через сеть. Если подпись ещё не проверена или её невозможно проверить, следует считать Interest не кэшированным.

4. Если Interest имеет ограничение Content Object Hash, хранилищу Content Store **недопустимо** отвечать на этот запрос, пока нет уверенности в наличии у Content Object корректного хэш-значения. Если проверка значения невозможна, Interest считается не кэшированным.

5. Хранилище должно выполнять директивы управления кэшем (см. раздел 4).

2.4.4. Конвейер Interest

1. Выполняется проверка HopLimit (см. параграф 2.4.1).

2. Если Interest включает проверку пригодности, такую как MIC или подпись со встроенным открытым ключом или сертификатом, пересылающий узел **может** проверить Interest, как указано в разделе 8. Пересылающему узлу **не следует** получать ключ через KeyLink. Если пересылающий узел отбрасывает Interest в результате проверки, он **может** передать Interest Return (параграф 10.3.9).

3. Определяется возможность агрегирования Interest, как указано в параграфе 2.4.2. Если агрегирование возможно, оно выполняется и Interest не пересылается.

4. При пересылке Interest проверяется его наличие в Content Store, как описано в параграфе 2.4.3. Если соответствующий Content Object найден, он возвращается на предыдущий интервал Interest и объект помещается в Content Store, как описано в параграфе 2.4.5.

5. Выполняется просмотр Interest в таблице FIB с поиском максимально длинного префикса (LPM[1]) по сегментам имени (а не по байтам или битам). Предыдущий интервал Interest **следует** исключать. Если найдено соответствие, сообщение Interest пересылается. Если соответствия не найдено или узел решил не пересылать сообщение по локальным условиям (например, перегрузка), ему **следует** передать сообщение Interest Return, как указано в разделе 10.

2.4.5. Конвейер объектов Content

1. Пересылающему узлу, получившему Content Object, **рекомендуется** проверить, что Content Object принят от ожидаемого предыдущего узла. На этот узел указывает FIB или запись в PIT, которая имеет совпадающее сообщение Interest, отправленное по этому пути.

2. Объект Content **должен** соответствовать всем ожидающим запросам Interests, которые удовлетворяют правилам сопоставления (см. раздел 9). Каждый удовлетворяющий запрос Interest **должен** удаляться из числа ожидающих Interest.

3. Пересылающему узлу **не следует** передавать более одной копии полученного Content Object на один предыдущий интервал Interest. Например, может случиться так, что два Interest запрашивают один Content Object по-разному (например, по имени или KeyId и имени) и оба объекта придут от одного предыдущего интервала. Нормально передавать несколько раз один Content Object через один интерфейс (например, Ethernet), если он передаётся на разные предыдущие интервалы.

4. Content Object **следует** помещать в Content Store лишь в том случае, когда он удовлетворяет Interest (см. п. 1 выше). Это снизит вероятность отравления кэша.

3. Имена

Имя CCNx состоит из нескольких сегментов, каждый из которых содержит метку, указывающую цель сегмента и его значение. Например, некоторые сегменты являются именами общего назначения, а другие служат для определённых целей, таких как передача информации о версии или упорядочение множества блоков большого объекта в мелкие подписанные Content Object.

В CCNx имеется три разных типа имён - префиксы, точные и полные имена. Префикс является просто именем, которое однозначно указывает не один Content Object, а скорее пространство имён или префикс существующего имени Content Object. Точное имя однозначно указывает Content Object. Полное имя включает точное имя и сопровождающее его явное или неявное значение ConObjHash. Явные ConObjHash используются в Interest, неявные - в Content Object.

Отметим, что узлам пересылки не нужно знать семантику имён. Они должны лишь быть способны сопоставлять префиксы для пересылки Interest и точное или полное имя - для пересылки Content Object. Узлы пересылки не чувствительны к типам сегментов.

Метки сегментов имён, заданных в этом документе, представлены в таблице 1. Name Segment является сегментом общего назначения, Interest Payload ID указывает данные Interest, а Application Components представляет набор типов сегментов имён, зарезервированных для приложений.

Таблица 1. Типы сегментов имён CCNx.

Тип	***Описание***
Name Segment	Базовый сегмент имени, включающий произвольные октеты.
Interest Payload ID	Строка октетов, указывающая данные, переносимые в Interest. Например, Payload ID может быть хэш-значением Interest Payload. Это позволяет различать Interest через сегмент имени без включения самих байтов данных.
Application Components	Специфичные для приложения данные в сегменте имени. Приложение может использовать свою семантику компонент. Хорошим тоном является указание приложения в сегменте имени перед сегментами компонент приложения.

На нижнем уровне пересылающему узлу не требуется знать семантику сегментов имени, достаточно лишь идентифицировать границы сегментов и обеспечить возможность проверки совпадения двух сегментов (их меток и

[1]Longest Prefix Match.

значений). Пересылающий узел сопоставляет имена посегментно со своей таблицей пересылки для поиска следующего интервала.

3.1. Примеры имён

В этом параграфе используется представление CCNx URI [ccnx-uri] для имён CCNx. Отметим, что эта грамматика применяется на уровне сообщения и у Interest должно быть поле Name хотя бы с одним сегментом имени, который должен иметь не менее 1 октета в значении. У Content Object должно быть похожее имя или совсем не быть имени. FIB может иметь имена размером 0 (принятый по умолчанию маршрут), первый сегмент без значения или обычное имя.

Таблица 2. Примеры имён CCNx.

Имя	***Описание***
ccnx:/	Пустое (0-length) имя, соответствующее принятому по умолчанию маршруту.
ccnx:/NAME=	Имя с 1 сегментом нулевого размера. Отличается от ccnx:/.
ccnx:/NAME=foo/APP:0=bar	2-сегментное имя, где первый сегмент имеет тип NAME, а второй - APP:0.

3.2. Идентификатор данных Interest

Interest может включать поле Payload, в котором передаётся состояние Interest, не используемое для сопоставления с Content Object. Если Interest содержит данные, в имя Interest следует включать Interest Payload ID (IPID). Значение IPID позволяет записи PIT корректно мультиплексировать Content Object в ответ на конкретные Interest с конкретным идентификатором данных. IPID можно выводить из хэш-значения данных, а также применять уникальный идентификатор GUID[1] или одноразовое значение (nonce). Необязательное поле Metadata определяет поле IPID так, чтобы другие системы могли проверить его (например, при выводе из хэш-значения). Проверка IPID не требуется.

4. Управление кэшем

CCNx поддерживает два поля для воздействия на управление кэшем, управляющие обработкой Content Object в кэше или Content Store. Они не применяются в быстром пути и служат лишь для определения возможности внедрения Content Object в быстрый путь при отклике на Interest.

Поле ExpiryTime используется в «конверте» подписи Validation Algorithm. Оно указывает время UTC в миллисекундах, после которого Content Object считается устаревшим и **должен** больше не применяться в откликах на Interest из кэша. Устаревшее содержимое **может** исключаться из кэша.

Поле RCT[2] размещается за пределами конверта подписи и указывает время UTC в миллисекундах, по достижении которого издатель будет считать Content Object малоценным для кэша. Системе кэширования **следует** отбрасывать объекты после RCT, хотя их **можно** хранить и применять в откликах. Кэш **может** отбросить Content Object до RCT - нет никаких обязательств по хранению объектов.

Эта формулировка позволяет издателю создавать объекты Content с большим ExpiryTime, но коротким RCT, снова и снова переиздавая один и тот же подписанный Content Object с расширением RCT. Это позволяет издателю периодически видеть, что содержимое продолжает использоваться.

5. Хэш Content Object

CCNx позволяет сообщениям Interest ограничивать отклик определенным хэшем, который учитывает тело сообщения Content Object и разделы проверки, если они имеются. Таким образом, для подписанного Content Object хэш будет учитывать значение подписи. Хэш не включает фиксированные и поэтапно добавляемые заголовки Content Object. Поскольку это является частью правил сопоставления (см. раздел 9), хэш применяется на каждом интервале.

Имеется два варианта сопоставления с ограничением Content Object Hash в Interest. Во-первых, узел пересылки может рассчитывать для себя хэш-значение и сравнивать его с ограничением, однако это ресурсоёмкая операция. Вторым вариантом является расчёт хэш-значения на граничном устройстве и его размещение в заголовке (ConObjHash), передаваемом через сеть. Этот вариант не обеспечивает защиты сопоставления, поэтому его **следует** применять лишь в доверенных средах. Заголовок **следует** удалять на границе доверенной области.

6. Канал

Link представляет собой кортеж {Name, [KeyIdRestr], [ContentObjectHashRestr]}, информация которого состоит из полей Interest, определяющих цель Link. Content Object с PayloadType = Link является объектом, данными которого является один или несколько каналов (Link). Этот кортеж может применяться в качестве KeyLink для указания конкретного объекта с «обёрнутым» сертификатом ключом. **Рекомендуется** включать хотя бы одно ограничение KeyId или Content Object Hash. Если таких ограничений нет, от издателя может быть возвращён любой Content Object с совпадающим именем.

7. Хэш-значения

Несколько полей протокола используют криптографические хэш-функции, которые должны быть защищены от атак и конфликтов. Поскольку такие функции с течением времени меняются - появляются лучшие, а старые становятся уязвимыми для атак, важно, чтобы реализации протокола CCNx сохраняли гибкость выбора хэш-функций.

В этом документе предложены несколько функций хэширования (например, SHA-256), но конкретная реализация может применять функцию, которая кажется лучшей. Нормативную спецификацию сообщений CCNx [RFC8609] следует считать определением подходящих функций и их использования.

[1]Globally Unique Identifier - уникальный в глобальном масштабе идентификатор.
[2]Recommended Cache Time - рекомендуемое время кэширования.

8. Проверка пригодности

8.1. Алгоритм проверки

Validator включает раздел ValidationAlgorithm, задающий способ проверки сообщения, и ValidationPayload с выводом проверки (например, цифровая подпись или MAC). Раздел ValidationAlgorithm определяет тип используемого алгоритма и включает требуемую дополнительную информацию. Проверка выполняется от начала сообщения CCNx до конца раздела ValidationAlgorithm (т. е. до данных проверки, но без их включения). Это называется областью проверки. ValidationPayload содержит байты значения контроля целостности, такие как MAC или подпись.

Грамматика сообщения CCNx (параграф 2.1) показывает разрешённые алгоритмы и их структуру. Для случая алгоритма Vendor производитель может использовать любую желаемую структуру. Validator может применяться лишь к Interest и Content Object, но не к Interest Return. Interest внутри Interest Return будет иметь исходное поле проверки, если оно есть.

Грамматика разрешает множество расширений Vendor для алгоритма проверки. Производитель должен описать область проверки для каждого расширения. Например, производитель может использовать обычную подпись в алгоритме проверки, затем добавить фирменный код MIC для обнаружения ошибок в сети без применения дорогостоящей проверки подписи. Данная спецификация не разрешает использовать несколько блоков Validation Algorithm за пределами фирменных методов.

8.2. Коды целостности сообщений

Если алгоритмом проверки служит CRC32C, данные проверки являются выводом CRC для проверяемой области. Этот алгоритм разрешает необязательную временную метку момента подписи (SigTime) при проверке сообщения (не совсем корректно называется «временем подписи», но здесь для этих целей применяется одно поле в MIC, MAC и подписях).

Коды MIC обычно применяются с Interest для предотвращения непреднамеренного повреждения в сети. Они обычно не используются с Content Object, поскольку объекты как правило подписаны или связаны с хэш-цепочкой, поэтому значение CRC32C избыточно.

8.3. Коды аутентификации сообщений

Если для проверки применяется алгоритм HMAC-SHA256, проверочными данными служит вывод HMAC для области проверки. Алгоритм проверки требует KeyId и разрешает SigTime и KeyLink.

Поле KeyId указывает общий секрет, используемый двумя сторонами для аутентификации сообщений. Эти секреты **следует** выводить из протокола обмена ключами, такого как [ccnx-ke]. KeyId для общего секрета **следует** быть неанализируемым идентификатором, который не выводится из реального ключа (хорошим выбором, например, будет целочисленный счётчик).

Временем подписи является временная метка момента, когда код аутентификации был рассчитан и добавлен в сообщение.

Поле KeyLink в MAC определяет способ согласования ключей и ему следует указывать в направлении конечной точки обмена ключами. Применение алгоритма согласования ключей выходит за рамки спецификации и такие алгоритмы не требуются для использования этого поля.

8.4. Подпись

Для проверки подписи применяются криптографические алгоритмы с открытым ключом, такие как RSA и ECDSA[1]. Данная спецификация и документ по кодированию [RFC8609] поддерживают лишь 3 конкретных алгоритма - RSA-SHA256, EC-SECP-256K1 и EC-SECP-384R1. Дополнительные алгоритмы могут применяться на основе других документов или с использованием экспериментальных и фирменных типов.

Для подписи на основе открытого ключа требуется поле KeyId и она может включать время подписи, встроенный открытый ключ и KeyLink. Время подписи является обычной временной меткой момента создания подписи. Применение этого поля и его взаимодействие с другими полями описано ниже.

Встроенные сертификаты не распространены, поскольку они могут быть очень большими, а периоды их действия могут отличаться от сроков действия проверяемых данных. Предпочтительным методом проверки является KeyLink.

Поле KeyId в ValidationAlgorithm указывает открытый ключ, применяемый для проверки подписи. Он похож на Subject Key Identifier в X.509 (параграф 4.2.1.2 [RFC5280]). KeyId определён как криптографическое хэш-значение открытого ключа в форме DER. Все реализации **должны** поддерживать SHA-256 для хэширования KeyId.

Использование других алгоритмов для KeyId разрешено и не будет вызывать проблем на узлах пересылки, поскольку эти узлы сопоставляют лишь алгоритм и его результат, не выполняя самих вычислений (см. раздел 9). Могут возникать проблемы с Content Store, поскольку хранилищу нужно проверять соответствие KeyId и PublicKey (см. параграф 2.4.3), хотя в этом случае может происходить лишь потеря кэша и сообщение Interest будет пересылаться издателю. Пока издатель и потребитель поддерживают хэш, данные будут проверяемы.

В соответствии с разделом 9, пересылающий узел сопоставляет лишь KeyId с ограничениями для него. Ему не нужно смотреть такие поля, как открытый ключ, сертификат или KeyLink.

Если в сообщении имеется несколько KeyId, открытых ключей, сертификатов или KeyLink, конечная точка (потребитель, кэш или Content Store) **должна** обеспечить согласованность этих полей. Поле KeyId **должно** быть соответствующим хэш-значением встроенного открытого ключа или открытого ключа сертификата. Хэш-функцией является HashType из KeyId. Если есть встроенный открытый ключ и сертификат, открытые ключи **должны** совпадать.

Сообщению **не следует** иметь PublicKey и KeyLink для открытого ключа, поскольку это избыточно. Оно **может** иметь PublicKey и KeyLink для сертификата.

[1]Elliptic Curve Digital Signature Algorithm - алгоритм цифровой подписи на основе эллиптической кривой.

KeyLink в первом Content Object может указывать на второй Content Object с DER-представлением открытого ключа в поле PublicKey и необязательным DER-представлением сертификата X.509 в поле данных (payload). KeyId второго Content Object **должен** совпадать с KeyId первого объекта. Поле PublicKey второго объекта **должно** быть открытым ключом, соответствующим KeyId. Этот ключ должен проверять пригодность подписей первого и второго объекта. Сертификат X.509 в форме DER может быть включён в данные второго объекта, а его встроенный открытый ключ **должен** соответствовать PublicKey. Он должен быть выдан доверенным центром, предпочтительно задающим действительное пространство имён ключа в отличительном имени. Второму объекту **недопустимо** иметь KeyLink, поскольку рекурсивный поиск ключа не разрешён.

9. Сопоставление Interest и Content Object

Content Object соответствует запросу Interest тогда и только тогда, когда (a) имя Content Object при его наличии совпадает с именем Interest, (b) ValidationAlgorithm KeyId в Content Object совпадает с ограничением Interest KeyId при наличии и (c) рассчитанное значение Content Object Hash совпадает с ограничением Interest Content Object при наличии.

KeyId и KeyIdRestr используют формат Hash (см. параграф 2.1), имеющий встроенное поле HashType, за которым следует хэш-значение. При сравнении KeyId и KeyIdRestr сравниваются HashType и значение.

Правила сопоставления задаются утверждением, значение true в котором означает, что Content Object соответствует Interest, Ni = Name в Interest (не может быть пустым), Ki = KeyIdRestr в Interest (может быть пустым), Hi = ContentObjectHashRestr в Interest (может быть пустым). No, Ko и Ho являются свойствами Content Object, где No и Ko могут быть пустыми, а Ho существует всегда (это внутреннее свойство Content Object). Для двоичных выражений используются символы & (AND) и | (OR), E означает оператор существования (не пусто), а ! - оператор отсутствия.

В особом случае, когда ограничение Content Object Hash в Interest задаёт не поддерживаемый алгоритм хэширования, Content Object не может соответствовать Interest, поэтому системе следует отбросить Interest и **можно** передать Interest Return на предыдущий интервал. В этом случае приведённое ниже утверждение никогда не выполняется, поскольку Interest не пересылается. Если система использует необязательное поведение, заключающееся в том, что другая система рассчитывает хэш для неё, эта система может предполагать поддержку всех хэш-функций и оставлять другой системе решение вопроса о восприятии или отклонении Interest.

```
(!No | (Ni=No)) & (!Ki | (Ki=Ko)) & (!Hi | (Hi=Ho)) & (E No | E Hi)
```

Как можно видеть, имеется два типа атрибутов, которые могут совпадать. Первое совпадение зависит от наличия атрибута в Content Object, а два следующих - от наличия атрибута в Interest. Последним совпадением является ограничение Nameless Object (безымянный объект), которое говорит, что Content Object, не имеющий Name, должен соответствовать Interest хотя бы по ограничению Hash.

Если Content Object не содержит Content Object Hash в явном виде, хэш-значение должно рассчитываться в сети для сопоставления. В автономной системе достаточно рассчитать Content Object Hash на граничном маршрутизаторе и передавать его доверенным способом внутри автономной системы. Если ValidationAlgorithm в Content Object не имеет KeyId, этот объект Content не может соответствовать Interest с ограничением KeyId.

10. Возврат Interest

В этом разделе описывается процесс, посредством которого элемент сети может возвращать сообщение Interest на предыдущий интервал, если при обработке Interest возникла ошибка. Возвращённое сообщение Interest может обрабатываться на предыдущем интервале или возвращаться в направлении его источника. Когда узел возвращает Interest, это указывает, что предыдущий интервал не ожидает отклика для данного Interest, т. е. на возвратившем сообщение узле нет записи PIT.

Возвращаемое сообщение совместимо с имеющимся форматом пакетов TLV (фиксированный заголовок, необязательные поэтапные заголовки и тело сообщения CCNx). Возвращаемый пакет Interest имеет лишь 2 отличия:

- в PacketType устанавливается Interest Return для индикации сообщения Feedback;
- в ReturnCode указывается подходящее значение для обозначения причины возврата.

Детали кодирования Interest Return указаны в [RFC8609].

Пересылающий узел не обязан передавать какие-либо сообщения Interest Return.

Пересылающий узел не обязан обрабатывать полученные сообщения Interest Return. Если он не может обработать Interest Return, **следует** просто отбросить сообщение.

Сообщения Interest Return не применяются к Content Object и другим типам сообщений.

Сообщение Interest Return передаётся между партнёрами одного интервала (1-hop) и не распространяется через множество интервалов с применением FIB. Промежуточный узел, получивший Interest Return, может выполнить корректирующие действия или передать своё сообщение Interest Return на предыдущий интервал, указанный в пути возврата записи PIT.

10.1. Формат сообщения

Сообщение Interest Return выглядит как исходное сообщение Interest, отличаясь лишь двумя отмеченными выше изменениями. Поле PacketType устанавливается для индикации типа сообщения Interest Return, а резервный байт в заголовке Interest служит кодом возврата. Значения PacketType и ReturnCode приведены в [RFC8609].

10.2. Типы ReturnCode

В этом параграфе описаны коды возврата Interest Return (ReturnCode), определённые в данном RFC. Численные значения кодов, используемые в пакетах, определены в [RFC8609].

Таблица 3. Коды причин Interest Return.

Имя	*Описание*
No Route (параграф 10.3.1)	Возвращающий узел пересылки не имеет пути к имени в Interest.
HopLimit Exceeded (параграф 10.3.2)	Значение HopLimit уменьшено до 0, но требуется пересылка пакета.
Interest MTU too large (параграф 10.3.3)	MTU для Interest не соответствует требуемому минимуму и нужна фрагментация.
No Resources (параграф 10.3.4)	Узел не имеет ресурсов для обработки Interest.
Path error (параграф 10.3.5)	Ошибка передачи при пересылке Interest по маршруту (временная ошибка).
Prohibited (параграф 10.3.6)	Административные настройки запрещают обработку Interest.
Congestion (параграф 10.3.7)	Сообщение Interest отброшено по причине перегрузки (временная ошибка).
Unsupported Content Object Hash Algorithm (параграф 10.3.8)	Сообщение Interest отброшено по причине запроса в нем ограничения Content Object Hash, использующего алгоритм хэширования, который не доступен.
Malformed Interest (параграф 10.3.9)	Сообщение Interest отброшено по причине невозможности корректного разбора.

10.3. Протокол возврата Interest

В этом параграфе описано поведение узла пересылки для разных кодов причины возврата Interest. От узла пересылки не требуется генерация каких-либо кодов, но если узел делает это, он **должен** следовать данной спецификации.

Если узел пересылки получает Interest Return, ему **следует** выполнить стандартные корректировочные действия. Узлу разрешено игнорировать сообщения Interest Return, при этом для записи PIT действует обычная процедура тайм-аута.

- Проверка прибытия Interest Return со следующего интервала, на который было реально передано Interest.
- Если записи PIT для соответствующего Interest нет, узлу пересылки следует игнорировать Interest Return.
- Если запись PIT для соответствующего Interest имеется, узел **может** выбрать один из двух вариантов:
  - попробовать другой путь пересылки при его наличии и отбросить Interest Return;
  - сбросить состояние PIT и передать Interest Return по пути возврата.

Если узел пересылки пытается использовать дополнительные маршруты, он **должен** гарантировать, что один и тот же путь не будет проверяться многократно. Например, он может отслеживать проверяемый следующий интервал и не применять его более одного раза.

При использовании узлом пересылки дополнительных маршрутов он может получить второе сообщение Interest Return возможно другого типа, нежели первое сообщение Interest Return. Например, узел A передал Interest для узла B, который вернул сообщение No Route. Затем узел A пытается передать сообщение через узел C, который возвращает Prohibited Interest Return. Узлу A следует выбрать подходящий код для отправки своему предыдущему интервалу.

При использовании узлом пересылки дополнительных маршрутов ему следует декрементировать Interest Lifetime с учётом времени, уже затраченного на обработку Interest.

10.3.1. No Route

Если узел пересылки получает Interest, для которого нет маршрута или существует лишь маршрут назад к передавшей Interest системе, узлу пересылки **следует** генерировать сообщение Interest Return No Route.

Способ управления узлом пересылки таблицей FIB при получении сообщения No Route зависит от реализации. В общем случае при получении No Route узлу пересылки не следует удалять маршрут из таблицы. Протоколу динамической маршрутизации, установившему маршрут, следует его исправить или установившему статический маршрут администратору следует внести коррективы в конфигурацию. Узел пересылки может подавить использование следующего интервала, который не работает, на некоторое время.

10.3.2. HopLimit Exceeded

Узел пересылки **может** передавать сообщения HopLimit Exceeded при получении сообщений Interest с HopLimit = 0, которые нужно пересылать.

10.3.3. Interest MTU Too Large

Если узел пересылки получает сообщение Interest, в котором MTU превышает предписанный минимум, узел **может** передать сообщение Interest MTU Too Large или просто отбросить Interest.

Если узел пересылки получает отклик Interest MTU Too Large, ему **не следует** пробовать другие пути. **Следует** передать это сообщение Interest Return на предыдущий интервал.

10.3.4. No Resources

Если узел пересылки получает сообщение Interest и не может его обработать по причине нехватки ресурсов, он **может** передать сообщение No Resources. Нехватка ресурсов может быть связана с размером PIT или иными ограничениями.

10.3.5. Path Error

Если узел пересылки обнаруживает ошибку при пересылке Interest (например, через надёжный канал), он **может** передать сообщение Path-Error, указывающее неспособность устранить ошибку при пересылке.

10.3.6. Prohibited

Узел пересылки может иметь административные правила, такие как списки контроля доступа (ACL[1]), которые препятствуют приёму или пересылке Interest. Если узел отбрасывает Interest на основании правил, он **может** передать сообщение Prohibited на предыдущий интервал. Например, при наличии ACL, разрешающего получать /example/private только на интерфейсе e0, но полученном на e1, узел должен иметь способ вернуть Interest с указанием причины.

10.3.7. Congestion

Узел пересылки, отбрасывающий Interest по причине перегрузки, **может** передать Congestion на предыдущий интервал.

10.3.8. Unsupported Content Object Hash Algorithm

Если ограничение Content Object Hash задаёт алгоритм хэширования, который узел пересылки не может проверить, Interest не следует воспринимать и узел **может** передать Interest Return на предыдущий интервал.

10.3.9. Malformed Interest

Если узел пересылки обнаруживает структурную или синтаксическую ошибку в Interest, ему **следует** отбросить Interest и **можно** передать Interest Return на предыдущий интервал. Это не означает, что любой маршрутизатор должен проверять структуру Interest целиком.

11. Взаимодействие с IANA

Этот документ не задаёт действий IANA.

12. Вопросы безопасности

CCNx является протоколом сетевого уровня (L3), который может работать в режиме наложения с использованием такого транспорта, как UDP или туннели. Протокол имеет встроенную поддержку аутентификации сообщений с помощью подписи (например, RSA или эллиптическая кривая) или кода аутентификации (например, HMAC). Взамен аутентификатора может применяться проверка целостности сообщения (например, SHA или CRC). CCNx не определяет «конверт» шифрования, оставляя это вышележащим протоколам (например, [esic]).

Формат сообщений CCNx позволяет присоединить MIC (например, CRC32C), MAC (например, HMAC) и подписи (например, RSA или ECDSA) к пакетам любого типа. Это не означает, что хорошей идеей будет использовать произвольный ValidationAlgorithm или включать ресурсоёмкие алгоритмы в пакеты Interest, поскольку это может приводить к DoS-атакам за счёт вычислений. Приложениям следует использовать явный протокол для руководства применением подписей пакетов. В качестве общего руководства приложения могут использовать MIC в сообщениях Interest для обнаружения непреднамеренно повреждённых пакетов. Если нужна защита Interest, следует рассмотреть возможность шифрования и протокол, предотвращающий атаки с повторным использованием, особенно при использовании Interest в качестве исполнительного механизма (actuator). Простое применение кода аутентификации или подписи не обеспечивает защиты Interest. В литературе имеется несколько примеров защиты обмена сообщениями в стиле ICN [mobile] [ace].

Поскольку документ относится к протоколу L3, он не описывает способов доставки ключей и механизмов доверия к ним. Content Object в CCNx может включать открытый ключ или сертификат, а также может использовать поле KeyLink для указания открытого ключа или сертификата для проверки подлинности сообщения. Одной из спецификаций обмена ключами является CCNxKE [ccnx-ke] [mobile], где обмен похож на процедуру TLS 1.3, отличаясь тем, что происходит через сообщения CCNx L3. Вопросы доверия выходят за рамки протокола L3 и решаются приложениями.

Комбинация обмена эфемерными ключами (например, CCNxKE [ccnx-ke]) с инкапсулирующим шифрованием (например, [esic]) обеспечивает эквивалент туннеля TLS. Промежуточные узлы могут пересылать сообщения Interest и Content Object, но не будут видеть их содержимого. Это также полностью скрывает внутренние имена, заменяя их именами, используемыми уровнем шифрования. Этот тип туннельного шифрования полезен для передачи содержимого, которое мало или совсем не подходит для кэширования, поскольку оно может применяться лишь теми, кто знает эфемерный ключ. Краткосрочное кэширование может помочь на каналах с потерями, но длительное кэширование обычно не представляет интереса.

Шифрование широковещательных пакетов и перешифрование на прокси могут быть полезны для содержимого, используемого многократно в течение времени или множеством потребителей. В настоящее время нет рекомендаций по этим вариантам шифрования.

Конкретное кодирование сообщений будет влиять на безопасность. В [RFC8609] применяется кодирование TLV. Был выбран компромисс между расширяемостью и однозначностью кодирования типа и размера. Некоторые TLV используют поля T и L переменного размера, чтобы вместить более широкий диапазон значений с сохранением эффективного использования байтов. Здесь TLV кодируются с 2-байтовыми полями T и L, что решает 2 проблемы. Первая проблема связана с псевдонимами. Если можно кодировать одно значение в полях разного размера (например, %x02 и %x0002), кто-то может счесть их одним, а другой - разными. Если есть различие, оно должно определяться в буферах, а не числовом представлении. Если от этого отказаться, придётся проверять кодирование TLV в каждом поле каждого пакета на всех узлах пересылки. Если они одинаковы, возникает другая проблема - как задавать фильтры. Например, если имя включает 6 компонент, имеется 7 полей T и 7 полей L, каждое из которых может иметь до 4 представлений одного значения. В результате имеется 14 полей с 4 представлениями для каждого или «1001 комбинация». Это также означает невозможность сравнения, например, имени через функции памяти, поскольку нужно учитывать разные форматы T и L.

Сообщение Interest Return не имеет аутентификатора от предыдущего интервала. Поэтому данные из Interest Return следует применять лишь локально для сопоставления с Interest. Узлу не следует пересылать эти Interest Payload как Interest. Он должен также убедиться, что передал Interest в Interest Return нужному узлу и не позволять кому-либо отменять сообщения Interest.

[1]Access control list.

Кэширующие узлы должны соблюдать осторожность при обработке Content Object. Важно, чтобы хранилище Content Store следовало правилам параграфа 2.4.3 во избежание некоторых типов атак. CCNx 1.0 не имеет механизмов обхода нежелательных результатов из сети (нет «исключений»), поэтому при отравлении кэша непригодным содержимым это может вызвать проблемы при поиске. Существует 3 типа доступа к содержимому из Content Store - неограниченный, а также ограниченные подписью и хэш-значением. Если Interest не имеет ограничений, запрашивающая сторона не заботится о том, что она получит в ответ и подойдёт любой кэшированный объект. При ограничении по хэш-значению запрашивающая сторона очень точно определяет желаемый результат и Content Store (и каждый пересылающий узел) может легко проверить соответствие содержимого запросу. При ограничении подписью (часто служит для начального обнаружения манифеста) запрашивающий знает лишь KeyId для подписи содержимого. Этот случай требует пристального внимания в Content Store, чтобы избежать усиления неверных данных. Хранилище Content Store должно отвечать лишь объектами Content с проверенной подписью. Это значит, что Content Object содержит в себе открытый ключ или Interest передаёт открытый ключ в дополнение к KeyId. Если это не выполняется, хранилищу Content Store следует рассматривать Interest как отсутствие кэша и позволять конечной точке ответить.

Кэш пользовательского уровня может выполнять полную проверку подписи путём. извлечения открытого ключа или сертификата по KeyLink. Однако это не та задача, которую можно отдать узлу пересылки. Пользовательский кэш может также полагаться на внешнюю (out-of-band) аттестацию, например, если оператор кэша включает лишь информацию, для которой у него есть корректная подпись.

Грамматика CCNx обеспечивает гибкость алгоритма хэширования с помощью HashType, указывая список приемлемых алгоритмов, которые следует реализовать на каждом узле пересылки. Некоторые типы пригодны лишь для конечных систем и обновление их не влияет на узлы пересылки, которые будут просто сопоставлять буферы, включающие type-length-hash. Некоторые поля (например, ConObjHash) должны проверяться на каждом узле, поэтому узел пересылки (или связанная с ним система) должен знать алгоритм хэширования. Это может вызвать проблемы совместимости при смене типа хэша. [RFC8609] является официальным источником данных о разрешённых типах хэширования.

Для имён CCNx применяется двоичное сопоставление, тогда как для URI сравниваются имена хостов без учёта регистра. Некоторые системы могут применять независимое от регистра сопоставление путей URI к ресурсу. В результате введённые человеком имена CCNx будут скорей всего сталкиваться с несоответствием символов разных регистров и символов не-ASCII, если не применять нормализацию URI для имён CCNx. Это означает, что объект, регистрирующий маршрутизируемый префикс CCNx, скажем ccnx:/example.com, должен зарегистрировать варианты типа ccnx:/Example.com. Если это не учтено в нормализации URI и соглашениях протокола маршрутизации, становятся возможными фишинговые атаки.

Более общее рассмотрение вопросов безопасности ICN приведено в [RFC7927] и [RFC7945].

13. Литература

13.1. 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>.

[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>.

13.2. Дополнительная литература

[ace] Shang, W., Yu, Y., Liang, T., Zhang, B., and L. Zhang, "NDN-ACE: Access Control for Constrained Environments over Named Data Networking", NDN Technical Report NDN-0036, December 2015, <http://new.named-data.net/wp-content/uploads/2015/12/ndn-0036-1-ndn-ace.pdf>.

[befrags] Mosko, M. and C. Tschudin, "ICN "Begin-End" Hop by Hop Fragmentation", Work in Progress, draft-mosko-icnrg-beginendfragment-02, December 2016.

[ccn-lite] Tschudin, C., et al., "CCN-lite", University of Basel, 2011-2019, <http://ccn-lite.net>.

[ccnx-ke] Mosko, M., Uzun, E., and C. Wood, "CCNx Key Exchange Protocol Version 1.0", Work in Progress, draft-wood-icnrg-ccnxkeyexchange-02, March 2017.

[ccnx-registry] IANA, "Content-Centric Networking (CCNx)", <https://www.iana.org/assignments/ccnx>.

[ccnx-uri] Mosko, M. and C. Wood, "The CCNx URI Scheme", Work in Progress, draft-mosko-icnrg-ccnxurischeme-01, April 2016.

[chunking] Mosko, M., "CCNx Content Object Chunking", Work in Progress, draft-mosko-icnrg-ccnxchunking-02, June 2016.

[cicn] FD.io, "Community ICN (CICN)", February 2017, <https://wiki.fd.io/index.php?title=Cicn&oldid=7191>.

[dart] Garcia-Luna-Aceves, J. and M. Mirzazad-Barijough, "A Light-Weight Forwarding Plane for Content-Centric Networks", International Conference on Computing, Networking, and Communications (ICNC), DOI 10.1109/ICCNC.2016.7440637, February 2016, <https://arxiv.org/pdf/1603.06044.pdf>.

[eprise-numbers] IANA, "IANA Private Enterprise Numbers", <https://www.iana.org/assignments/enterprise-numbers>.

[esic] Mosko, M. and C. Wood, "Encrypted Sessions In CCNx (ESIC)", Work in Progress, draft-wood-icnrg-esic-01, September 2017.

[flic] Tschudin, C. and C. Wood, "File-Like ICN Collection (FLIC)", Work in Progress, draft-tschudin-icnrg-flic-03, March 2017.

[mobile] Mosko, M., Uzun, E., and C. Wood, "Mobile Sessions in Content-Centric Networks", IFIP Networking Conference (IFIP Networking) and Workshops, DOI 10.23919/IFIPNetworking.2017.8264861, June 2017, <https://dl.ifip.org/db/conf/networking/networking2017/1570334964.pdf>.

[ndn] UCLA, "Named Data Networking", 2019, <https://www.named-data.net>.

[nnc] Jacobson, V., Smetters, D., Thornton, J., Plass, M., Briggs, N., and R. Braynard, "Networking Named Content", Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, DOI 10.1145/1658939.1658941, December 2009, <https://dx.doi.org/10.1145/1658939.1658941>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC7927] Kutscher, D., Ed., Eum, S., Pentikousis, K., Psaras, I., Corujo, D., Saucez, D., Schmidt, T., and M. Waehlisch, "Information-Centric Networking (ICN) Research Challenges", RFC 7927, DOI 10.17487/RFC7927, July 2016, <https://www.rfc-editor.org/info/rfc7927>.

[RFC7945] Pentikousis, K., Ed., Ohlman, B., Davies, E., Spirou, S., and G. Boggia, "Information-Centric Networking: Evaluation and Security Considerations", RFC 7945, DOI 10.17487/RFC7945, September 2016, <https://www.rfc-editor.org/info/rfc7945>.

[RFC8609] Mosko, M., Solis, I., and C. Wood, "Content-Centric Networking (CCNx) Messages in TLV Format", RFC 8609, DOI 10.17487/RFC8609, July 2019, <https://www.rfc-editor.org/info/rfc8609>.

[selectors] Mosko, M., "CCNx Selector Based Discovery", Work in Progress, draft-mosko-icnrg-selectors-01, May 2019.

[terminology] Wissingh, B., Wood, C., Afanasyev, A., Zhang, L., Oran, D., and C. Tschudin, "Information-Centric Networking (ICN): CCN and NDN Terminology", Work in Progress[1], draft-irtf-icnrg-terminology-04, June 2019.

[trust] Tschudin, C., Uzun, E., and C. Wood, "Trust in Information-Centric Networking: From Theory to Practice", 25th International Conference on Computer Communication and Networks (ICCCN), DOI 10.1109/ICCCN.2016.7568589, August 2016, <https://doi.org/10.1109/ICCCN.2016.7568589>.

Адреса авторов

Marc Mosko

PARC, Inc.

Palo Alto, California 94304

United States of America

Phone: +01 650-812-4405

Email: marc.mosko@parc.com

Ignacio Solis

LinkedIn

Mountain View, California 94043

United States of America

Email: nsolis@linkedin.com

Christopher A. Wood

University of California Irvine

Irvine, California 92697

United States of America

Phone: +01 315-806-5939

Email: woodc1@uci.edu

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

[1]Опубликовано в RFC 8793. *Прим. перев.*