Internet Engineering Task Force (IETF)
Request for Comments: 8345
Category: Standards Track
ISSN: 2070-1721

A. Clemm
Huawei
J. Medved
Cisco
R. Varga
Pantheon Technologies SRO
N. Bahadur
Bracket Computing
H. Ananthakrishnan
Packet Design
X. Liu
Jabil

March 2018

Модель данных YANG для сетевой топологии

A YANG Data Model for Network Topologies

Тезисы

Этот документ определяет абстрактную (общую или базовую) модель данных YANG для топологии и кадастров (учета) сетей и/или служб. Модель данных служит базой, которая может дополняться деталями топологии в моделях для более конкретных вариантов топологии или кадастра.

Статус документа

Документ относится к категории Internet Standards Track.

Документ является результатом работы IETF[1] и представляет согласованный взгляд сообщества IETF. Документ прошел открытое обсуждение и был одобрен для публикации IESG[2]. Дополнительную информацию о стандартах Internet можно найти в разделе 2 в RFC 7841.

Информацию о текущем статусе документа, ошибках и способах обратной связи можно найти по ссылке https://www.rfc-editor.org/info/rfc8345.

Оглавление

[1]Internet Engineering Task Force.
[2]Internet Engineering Steering Group.

1. Введение

В этом документе вводится абстрактная (базовая) модель данных [RFC3444] YANG [RFC7950] для представления сетей и топологии. Модель делится на две части. Первая часть определяет модель данных сети, которая позволяет определить сетевые иерархии или стеки сетей (т. е. наложенные сети) и поддержку кадастра узлов, содержащихся в сети. Вторая часть дополняет базовую модель сети информацией, описывающей сетевую топологию. В частности, она добавляет концепцию каналов (соединений) и точек завершения для описания соединений узлов в сети. Кроме того, модель данных задает вертикальные отношения между сетями, которые могут быть дополнены для охвата как кадастров, так и топологии сети и/или служб.

Хотя возможно объединение обеих частей в одну модель, разделение упрощает интеграцию моделей данных кадастра и сетевой топологии, поскольку позволяет добавлять кадастровую информацию в модель данных сети без учета топологии.

Модель данных может быть дополнена путем задания отдельных типов сетей и топологии. Например, дополнение модели может включать информацию об атрибутах узлов для конкретного типа сети. Примеры дополненных моделей включают модели данных для топологии канального уровня (L2), варианты топологии сетевого уровня (L3) типа IGP, IS-IS [RFC1195] и OSPF [RFC2328], данные организации трафика (TE[1]) [RFC3209] и разные варианты топологии транспорта и служб. Информация для конкретного типа сети будут собираться в отдельные модели, зависящие от сетевой технологии.

Базовая модель данных, описанная в этом документе, подходит для множества вариантов топологии сетей и служб, а также для кадастров. Модель данных позволяет приложениям оперировать с кадастром или топологией любой сети на базовом уровне, где специфика конкретных вариантов кадастра или топологии не требуется. В то же время при рассмотрении конкретного типа сети и дополнении модели данных созданные ранее данные сохраняют свою структуру и представляются в согласованном виде. Это также упрощает представление сетевых иерархий и зависимостей между разными сетевыми компонентами и типами сетей.

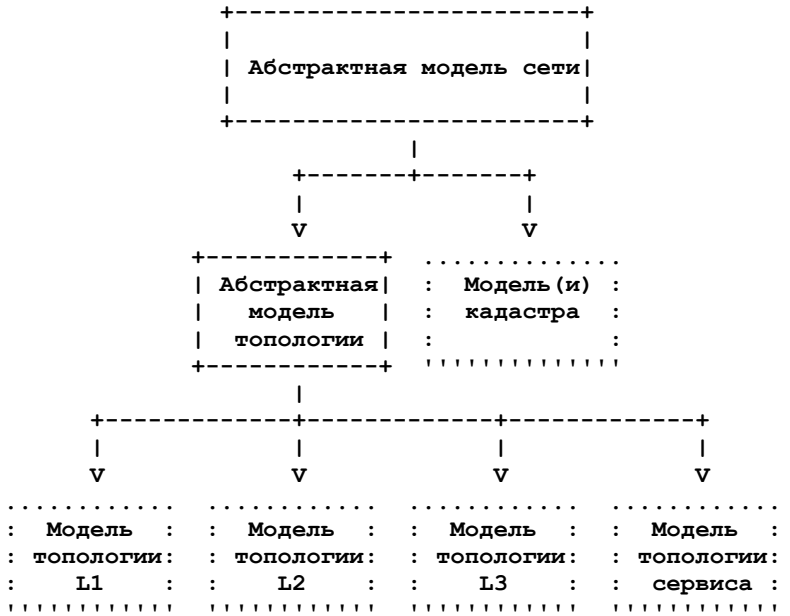

Рисунок 1. Структура модели данных сети.

Модуль YANG для абстрактной (базовой) сети, определенный в этом документе, называется ietf-network (параграф 6.1) и содержит список узлов абстрактной сети, а также определяет концепцию «иерахии сетей» (стека). Узел абстрактной сети может быть дополнен в моделях данных кадастра или топологии зависящими от этого кадастра или топологии атрибутами. Иерархия (стек) сетей позволяет данной сети иметь одну или множество «поддерживающих сетей». Отношения между моделью данных базовой сети, моделями данных кадастра и топологическими моделями данных показаны на рисунке 1 (линии из точек указывают возможность дополнения моделей, определенных в этом документе).

[1]Traffic engineering.

Модуль YANG для топологии сети, определенный в этом документе, называется ietf-network-topology (параграф 6.2) и определяет базовую топологическую модель с максимальным уровнем абстракции. Модуль определяет граф и компоненты топологии - узлы (node), ребра (edge) и точки завершения (termination point). Узлы (из модуля ietf-network) представляют вершины графа, а соединения - его ребра. Узлы также включают точки завершения для привязки соединений. Сеть может содержать множество топологий, например, топология разных уровней и топология перекрытий. Поэтому модель данных поддерживает связь между топологиями, а также зависимости между узлами и точками завершения разных топологий. Пример топологического стека приведен на рисунке 2.

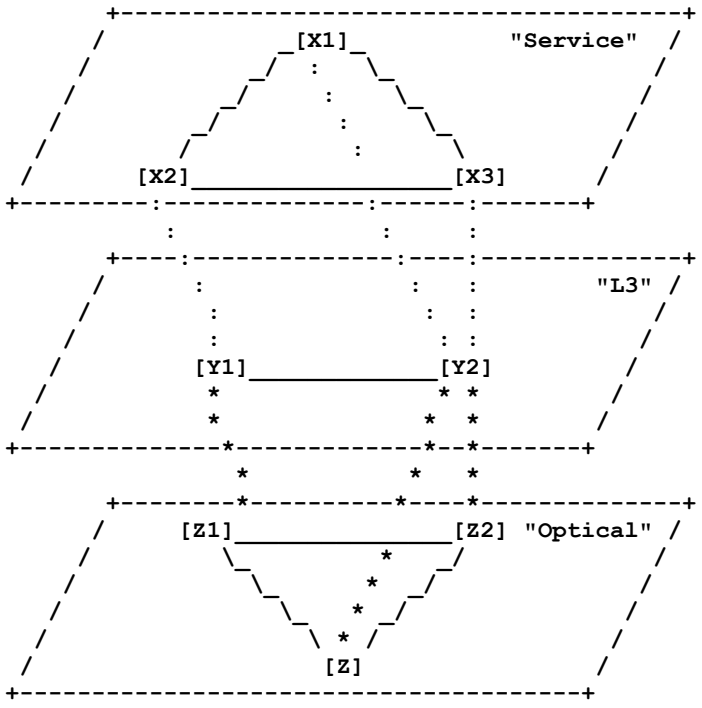

Рисунок 2. Пример иерархии (стека) топологий.

На рисунке 2 показаны три уровня топологии. На верхнем уровне Service показаны отношения между элементами служб типа сервисных функций и цепочек услуг. Топология сетевого уровня L3 показывает элементы уровня L3 (IP), а уровень Optical показывает сетевые элементы физического уровня L1. Сервисные функции в топологии Service отображаются на сетевые элементы топологии L3, которые в свою очередь отображаются на элементы физического уровня топологии Optical. Две сервисных функции (X1 и X3) отображаются на один элемент L3 (Y2) - это может происходить, например, при реализации двух функций на одной виртуальной машине VM[1] (или сервере) - и сообща используют сетевые интерфейсы. Один сетевой элемент L3 (Y2) отображается на два элемента уровня Optical (Z2 и Z). Это может быть, например, при подключении одного маршрутизатора IP к разным мультиплексорам ROADM[2] в оптическом домене.

Другой пример стека топологий служб показан на рисунке 3.

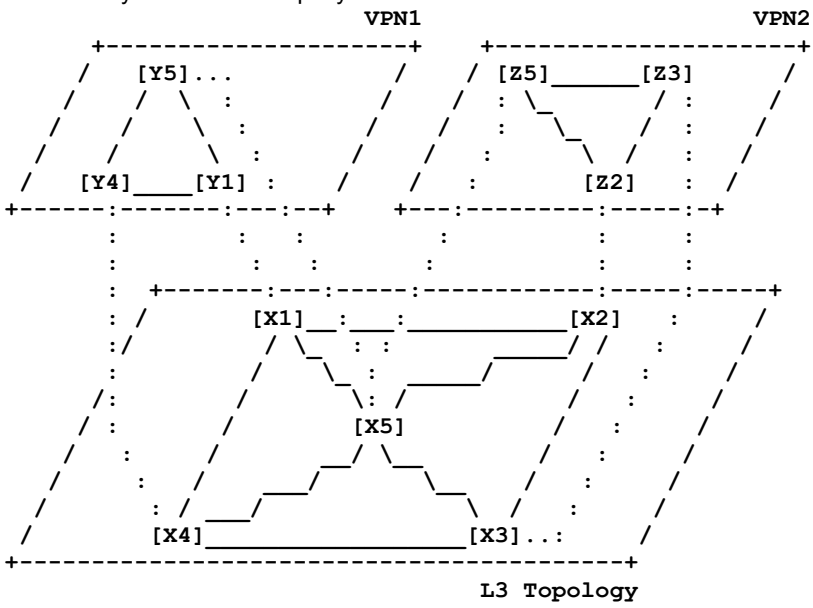

Рисунок 3. Пример топологической иерархии (стека).

На рисунке 3 показаны две топологии сервиса VPN (VPN1 и VPN2) организованные на базе общей топологии L3. Каждая топология сервиса VPN отображается на подмножество узлов общей топологии L3.

Существует множество применений для такой модели данных. Например, в контексте интерфейса в систему маршрутизации I2RS[3] узлы сети могут использовать модель данных для понимания общей топологии сети и ее

[1]Virtual Machine.
[2]Reconfigurable Optical Add/Drop Multiplexer - перенастраиваемый оптический мультиплексор ввода-вывода.
[3]Interface to the Routing System.

представления сетевому контроллеру. Контроллер может использовать данные подтвержденной топологии для сравнения и согласования своего представления топологии сети с ее представлением управляемыми им элементами. В дополнение к этому узлы сети могут распространять свое понимание для сравнения и согласования между собой или с помощью контроллера. Помимо сетевых элементов и непосредственного контекста самого I2RS, сетевой контроллер может использовать модель данных для представления своего взгляда на топологию, которую он контролирует, приложениям на своем северном интерфейсе. Другие случаи, где может быть применена описанная модель данных, рассмотрены в [USECASE-REQS].

В этой модели данных сеть классифицируется как управляемая системой или нет. Если сеть управляется системой, она автоматически заполняется сервером и представляет динамически определяемую (learned) информацию, которая может быть прочитана из хранилища данных операционного состояния. Модель данных может также служить для создания или изменения сетевой топологии, что может быть связано с моделью кадастра или наложенной (overlay) сетью. Такая сеть не контролируется системой, а вместо этого настраивается клиентом.

Модель данных позволяет сети ссылаться на поддерживающую сеть, поддерживающие узлы, каналы и т. п. Модель также позволяет делить на уровни сеть, расположенную «поверх» сети, контролируемой системой. Этот позволяет настраивать наложенные сети поверх тех сетей, которые были обнаружены. В частности, эта модель данных структурирована для поддержки реализации в форме части эфемерного хранилища данных [RFC8342], требования к которому определены в разделе 3 [RFC8242]. Это позволяет записывать данные сетевой топологии, т. е. обеспечивает возможность настройки клиентом без контроля системы для ссылки на динамически получаемые данные, которые контролируются системой, а не настраиваются клиентом. Простой пример использования может включать наложенную сеть, которая поддерживается динамически определяемой сетевой топологией с маршрутизацией IP. Когда реализация помещает записанные данные для этой модели в эфемерное хранилище, такая сеть **может** указывать на другую сеть, контролируемую системой.

2. 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 интерпретируются в соответствии с BCP 14 [RFC2119] [RFC8174] тогда и только тогда, когда они выделены шрифтом, как показано здесь.

3. Определения и сокращения

Datastore – хранилище данных
: Концептуальное место для хранения и доступа к информации. Хранилище может быть реализовано, например, в виде файла, базы данных, flash-памяти или их комбинации. Хранилище отображается на экземпляр дерева данных YANG (определение из [RFC8342]).

Data subtree – ветвь (субдерево) данных
: Конкретный узел данных и узлы, иерархические входящие в него.

IGP
: Interior Gateway Protocol - протокол внутреннего шлюза.

IS-IS
: Intermediate System to Intermediate System - протокол взаимодействия промежуточных систем.

OSPF
: Open Shortest Path First - сначала кратчайший путь (протокол маршрутизации по состоянию канала).

SDN
: Software-Defined Networking - программно определяемая сеть.

URI
: Uniform Resource Identifier - унифицированный идентификатор ресурса.

VM
: Virtual Machine - виртуальная машина.

4. Структура модели

4.1. Базовая модель сети

Модель данных абстрактной (базовой) сети определена в модуле ietf-network. Ее структура показана на рисунке 4. Обозначения на рисунке соответствуют синтаксису, используемому в [RFC8340].

```
module: ietf-network
  +--rw networks
     +--rw network* [network-id]
        +--rw network-id            network-id
        +--rw network-types
        +--rw supporting-network* [network-ref]
        |  +--rw network-ref    -> /networks/network/network-id
        +--rw node* [node-id]
           +--rw node-id            node-id
           +--rw supporting-node* [network-ref node-ref]
              +--rw network-ref
              |       -> ../../../supporting-network/network-ref
              +--rw node-ref       -> /networks/network/node/node-id
```

Рисунок 4. Структура модели данных абстрактной (базовой) сети.

Модель данных содержит контейнер со списком сетей. Каждая сеть фиксируется в своей записи, указанной network-id.

Сеть имеет тот или иной тип (например L2, L3, OSPF, IS-IS) или относится сразу к нескольким типам. Тип или типы указываются под контейнером network-types. В этой модели он служит лишь дополнением цели. Модули для конкретного типа сети будут позднее добавлять новые узлы данных для представления типа ниже этой цели (т. е. ниже network-types) с помощью дополнения YANG.

Когда сеть имеет определенный тип, она будет содержать соответствующий узел данных. Узлы всегда **следует** представлять с использованием контейнеров присутствия, а не листьев типа empty. Это обеспечивает представление иерархий подтипов сетей в информации экземпляра. Например, экземпляр сети OSPF (который в то же время является сетью L3 unicast IGP) будет содержать под network-types другой контейнер присутствия l3-unicast-igp-network, который в свою очередь будет содержать контейнер присутствия ospf-network. Фактические примеры можно найти в [RFC8346].

Сеть может быть частью иерархии сетей, построенной поверх других сетей. Все такие сети фиксируются в списке supporting-network. Поддерживающей сетью является базовая (underlay) сеть.

Кроме того, сеть содержит кадастр узлов, являющихся ее частями. Узлы фиксируются в своих списках. Каждый узел указывается относительно содержащей его сети идентификатором node-id.

Следует отметить, что узел не существует отдельно от сети - он является частью содержащей его сети. В тех случаях, когда устройство или элемент является частью нескольких сетей или нескольких уровней стека сетей, такое устройство или объект будет представлено множеством узлов (по одному для каждой сети). Иными словами, узел представляет абстракцию устройства для отдельной сети, в которую это устройство входит. Для индикации включения одного устройства или элемента в разные топологии или сети можно создать одну «физическую» сеть со списком узлов для каждого из устройств или элементов. Эту (физическую) сеть (узлы и элементы сети) можно называть базовой (underlay) сетью, а ее узлы - узлами других (логических) сетей и узлов. Отметим, что модель данных позволяет определить более одной базовой сети (и узла), что дает возможность одновременно представлять многоуровневые топологии и топологии служб, а также физическое размещение.

Подобно сети, узел может поддерживаться другими узлами и отображаться на один или множество других узлов в базовой сети. Это фиксируется в списке supporting-node (поддерживающий узел). Результирующая иерархия узлов позволяет также представлять стеки устройств, где узлы одного уровня поддерживаются набором узлов нижележащего уровня. Например,

- узел router может поддерживаться узлом, представляющим процессор маршрутизатора, и отдельными узлами для разных интерфейсных плат и сервисных модулей;
- виртуальный маршрутизатор может поддерживаться или размещаться на физическом устройстве, представленном отдельным узлом

и т. д.

Данные о сети на определенном уровне могут происходить один из двух источников: (1) данные настраиваются клиентским приложением (например, в случае наложенных сетей эти данные могут задаваться контроллером SDN) или (2) данные автоматически контролируются системой в случае сети, которая может быть обнаружена. Настраиваемая (наложенная) сеть может ссылаться на обнаруживаемую (базовую) сеть.

Для учета этих возможностей используется пересмотренная архитектура хранилищ данных [RFC8342]. В частности, для каждой сети источник данных указывается аннотацией метаданных origin [RFC7952] (в соответствии с определением [RFC8342]) - intended для данных, настраиваемых клиентским приложением и learned для раскрываемых данных. Раскрываемые данные автоматически заполняются как часть рабочего хранилища данных. Настраиваемые данные являются частью конфигурации и предусмотренных хранилищ. Настраиваемые данные, которые действительно применяются, дополнительно отражаются в рабочем хранилище. Данные в этом хранилище всегда будут иметь полную целостность ссылок. Если сконфигурированный элемент данных (например, узел) имеет «оборванную» ссылку на отсутствующий элемент (например, поддерживающий узел), этот сконфигурированный элемент будет автоматически удален из рабочего хранилища и останется лишь в хранилище предполагаемой конфигурации. Для устранения конфликта клиентское приложение (например, контроллер SDN) должно указать корректные ссылки на поддерживающие ресурсы.

4.2. Базовая модель данных сетевой топологии

Модель данных абстрактной (базовой) сетевой топологии определена в модуле ietf-network-topology. Она построена на основе модели данных сети, определенной в модуле ietf-network и дополненном каналами (определяют соединения между узлами) и точками завершения (содержащиеся в узлах привязки к каналам). Структура модели топологии сети представлена на рисунке 5. Синтаксис обозначений соответствует [RFC8340].

Узел имеет список точек завершения, которые служат для завершения (привязки) каналов. Примером точки завершения может служить физический или логический порт, более обобщенно - интерфейс.

Подобно узлу, точка завершения может поддерживаться базовой (нижележащей) точкой завершения, содержащейся в узле поддержки базовой сети.

Каналы обозначаются идентификаторами link-id, которые однозначно указывают канал в рамках данной топологии. Каналы относятся к типу «точка-точка» и являются односторонними. Следовательно, каждый канал имеет источник и цель. И то и другое указывает соответствующий узел, а также точку завершения на этом узле. Подобно узлу, канал может отображаться на один или множество каналов (которые завершаются в соответствующих базовых точках привязки) базовой топологии. Это отображение фиксируется в списке supporting-link.

```
module: ietf-network-topology
  augment /nw:networks/nw:network:
    +--rw link* [link-id]
       +--rw link-id            link-id
       +--rw source
       |  +--rw source-node?   -> ../../../nw:node/node-id
       |  +--rw source-tp?     leafref
       +--rw destination
       |  +--rw dest-node?   -> ../../../nw:node/node-id
       |  +--rw dest-tp?     leafref
       +--rw supporting-link* [network-ref link-ref]
          +--rw network-ref
          |       -> ../../../nw:supporting-network/network-ref
          +--rw link-ref       leafref
  augment /nw:networks/nw:network/nw:node:
    +--rw termination-point* [tp-id]
       +--rw tp-id                            tp-id
       +--rw supporting-termination-point*
                [network-ref node-ref tp-ref]
          +--rw network-ref
          |       -> ../../../nw:supporting-node/network-ref
          +--rw node-ref
          |       -> ../../../nw:supporting-node/node-ref
          +--rw tp-ref         leafref
```

Рисунок 5. Структура модели данных топологии абстрактной (базовой) сети.

4.3. Расширение модели данных

Чтобы создать производную модель данных для конкретного типа сети, базовая модель может быть расширена. Это можно сделать примерно так - создается новый модуль YANG для нового типа сети и в этом модуле определяются дополнения к модулям ietf-network и ietf-network-topology.

Начнем дополнение с модуля ietf-network. Сначала требуется определить новый тип сети - это делается путем определения контейнера присутствия, который представляет новый тип сети. Новый тип добавляется под контейнером базовых типов. Далее определяются узлы данных для всех параметров узла, относящиеся к конкретному типу сети, и добавляются в список узлов. Новые узлы данных могут быть определены как условные (when) по наличию соответствующего типа в содержащей сети. При наличии каких-либо требований или ограничений применительно к иерархии сетей типа потребности нового типа сети в определенной базовой сети, можно определить соответствующие ограничения, а также дополнения к списку поддерживающих сетей. Однако следует соблюдать осторожность и избегать слишком большого числа ограничений.

Далее определяются дополнения для модуля ietf-network-topology. Определяются узлы данных для параметров каналов и точек завершения, которые относятся к новому типу сети. Эти узлы добавляются в списки каналов и точек завершения, соответственно. Эти узлы данных также могут определяться по условию присутствия соответствующего типа сети в содержащей сети с помощью оператора when.

Возможна (но не требуется) группировка узлов данных для типа сети под выделенным контейнером. Это обеспечивает дополнительную структуризацию, до удлиняет имена путей к узлам данных.

В тех случаях, где определена иерархия типов сетей, дополнения могут применяться к дополняющим модулям с использованием более специфического типа сети.

4.4. Обсуждение и выбор некоторых решений

4.4.1. Структура контейнера

Вместо поддержки списков в отдельных контейнерах модель данных остается сравнительно плоской с части структуры контейнеров. Списки узлов, каналов и узлов поддержки, каналов поддержки и поддерживающих точек завершения не хранятся в отдельных контейнерах. Поэтому идентификаторы путей, используемые для указания конкретных узлов (в операциях управления или в спецификациях ограничений) могут оставаться сравнительно компактными. Конечно это означает, что в информации экземпляра нет отдельной структуры, которая отделяет элементы одного списка от элементов другого. Семантически такая структура не требуется, но она может предоставляться в некоторых случаях для удобочитаемости.

4.4.2. Базовые иерархии и отображения

Для минимизации допущения о привязках конкретных элементов отображения между сетями, узлами, каналами и точками завершения являются обобщенными. Например, не принимается каких-либо допущений о связи точек завершения с интерфейсами или связи узлов с конкретными «сисетмами» или устройствами - модель данных на этом общем уровне не предусматривает таких привязок.

Когда требуются более конкретные отображения между верхним и нижним уровнем, эта информация может фиксироваться в модулях дополнения. Например, для указания того, что точка завершения определенного типа сети отображается на интерфейс, может быть добавлен модуль дополнения для точки завершения. Дополнение создает лист типа interface-ref, который указывает на соответствующий интерфейс [RFC8343]. Аналогично, если узел отображается на конкретное устройство или элемент сети, модуль дополнения может дополнять данные узла листом, указывающим на элемент сети.

Канал одного уровня иерархии может отображаться на множество каналов другого уровня. Например, топология VPN может моделировать туннели VPN как каналы. Когда туннель VPN отображается на путь, состоящий из цепочки отдельных соединений, канал будет содержать список поддерживающих соединений. Аналогично каналы одного уровня иерархии могут объединяться (агрегироваться) в связку каналов другого уровня.

4.4.3. Изменения в базовых сетях

В сети возможны разные флуктуации, даже если эта сеть является базовой для других сетей. При удалении поддерживающего узла, канала или точки завершения поддерживающие leafref в наложении становятся «подвешенными». Для таких случаев в модели данных используется конструкция require-instance языка YANG 1.1 [RFC7950].

Обособленный лист leafref настроенного объекта оставляет соответствующий экземпляр в состоянии с утерянной целостностью ссылок, что фактически делает его неработоспособным. Поэтому все соответствующие экземпляры объекта удаляются из рабочего хранилища данных пока ситуация не будет исправлена (1) путем добавления поддерживающего объекта в рабочее хранилище или изменения экземпляра с указанием ссылки на другой объект, который реально отражен в рабочем хранилище. В хранилище предполагаемой конфигурации объект будет сохранен.

Приложение, поддерживающее наложение, отвечает за возможность оттока в базовую сеть. Когда сервер получает запрос на настройку наложенной сети, ему **следует** проверить, указывают ли поддерживающие узлы/каналы/точки завершения на реально существующие узлы базовой сети, т. е. убедиться в том, что узлы отражены в хранилище рабочего состояния. Запросы конфигурации, в которых узлы/каналы/точки завершения указывают на несуществующие узлы, **следует** отвергать. Приложение отвечает за обновление наложений при последующем удалении узла/канала/точки завершения. По этой причине приложение может подписаться на обновления в базовой сети, например, с помощью механизмов, определенных в [YANG-Push].

4.4.4. Использование группировок

Модель данных использует группировки вместо простого определения узлов данных как встроенных (inline). Это упрощает включение соответствующих узлов данных в уведомления, не требуя заново определять каждый включаемый узел данных. Однако это усложняет задание ограничений, поскольку ограничения, вовлекающие узлы данных вне группировки требуется задавать вместе с оператором uses при использовании группировки. Это также означает, что ограничения и операторы языка описания путей (XPath[1]) требуется указывать так, чтобы они сначала перемещались «вниз» и выбирали весь набор узлов, а не просто задавать их для отдельных узлов данных.

4.4.5. Тип и направленность соединений

Модель топологических данных включает каналы, которые относятся к типу «точка-точка» и являются односторонними. Модель не поддерживает напрямую многоточечные и двухсторонние каналы. Это может показаться ограничением, однако такое решение делает модель простой и универсальной, позволяя использовать ее в приложениях, применяющих алгоритмы на основе графов. Двухстороннее соединение можно представить парой однонаправленных. Многоточечные сети можно представить псевдоузлами (например, как в IS-IS). Внедрение иерархии, в которой узлы одного уровня отображаются на множество узлов другого уровня и внедрение соединений на этом уровне позволяет представить топологии, не являющиеся соединениями «точка-точка».

4.4.6. Многодомность и агрегирование каналов

Канал завершается одной точкой, а не множеством их. Соединения, включающие многодомность и агрегирование, необходимо представлять множеством соединений «точка-точка», определяя на более высоком уровне канал, образуемых этими отдельными соединениями.

4.4.7. Избыточность отображения

В иерархии сетей имеются узлы, отображенные на узлы, канала, отображенные на каналы, и точки завершения, отображенные на точки завершения. Часть этой информации является избыточной. В частности, если известно отображение канала на один или несколько других каналов и известны точки завершения каждого канала, информация об отображении точек завершения может быть выведена путем транзитивного замыкания и не требует явной настройки. Тем не менее, чтобы не ограничивать приложения в части настройки и вывода отображений, модель предоставляет опцию явной настройки этой информации. Модель данных включает ограничения целостности, позволяющие проверить согласованность.

4.4.8. Типизация

Типы сетей представляются с использованием контейнера, который содержит узел данных для каждого из имеющихся в сети типа. Сеть может включать одновременно несколько типов сетей, поэтому применяется контейнер вместо конструкции case, причем каждый тип сети в свою очередь представляется выделенным контейнером присутствия. Причина того, чтобы не использовать просто пустой лист или (что еще проще) отказаться от контейнера network и просто использовать взамен лист-список (leaf-list) network-type, заключается в возможности представления иерархий классов для типов сетей, где один тип уточняет другой. Контейнеры для определенного типа сети определяются в связанных с сетью модулях, дополняющий контейнер network-types.

4.4.9. Представление одного устройства в нескольких сетях

Одно общее требование связано с возможностью указать, что одно и то же устройство может входить в разные сети и топологии. Однако модель данных определяет узел относительно сети, в которой он расположен. Один уже не может быть частью разных топологий. Во многих случаях узел будет абстракцией отдельного устройства в сети. Для отражения принадлежности одного устройства множеству топологий можно выбрать специальную модель - вводится новый тип сети для представления физической сети («устройства») с узлами, представляющими устройства. Эта сеть формирует базовую сеть для расположенной выше логической сети с отображением узлов логической сети на узлы физической сети.

Этот сценарий представлен на рисунке 6, где приведены 3 сети с двумя узлами в каждой. Физическая сеть (P на рисунке) состоит из двух узлов (D1 и D2), являющихся устройствами. Вторая сеть (X) имеет третью (Y) в качестве базовой и для обеих сетей X и Y физическая сеть P служит базовой. X1 имеет базовые узлы Y1 и D1, а для Y1 узел D1 является базовым. Аналогично, X2 имеет базовые узлы Y2 и D2, а Y2 имеет базовый узел D2. Размещение X1 и Y1 на обном физическом узле (D1) легко увидеть на рисунке.

[1]XML Path Language.

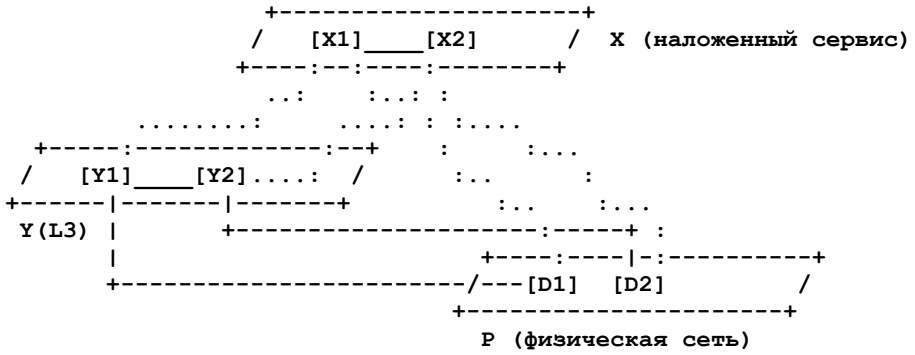

Рисунок 6. Пример иерархии топологий - множественное наложение.

В случае физической сети узлы представляют устройства, а точки завершения - физические порты. Следует отметить, можно дополнить модель данных для физического типа сети, определив дополнения, которые имеют узлы, указывающие системную информацию, и точки завершения, указывающие физические интерфейсы, чтобы обеспечить «мост» между моделями сети и устройств.

4.4.10. Поддержка настроенной клиентом и управляемой системой топологии

YANG требует обозначать узлы данных как конфигурационные (config true) или операционные (config false), но не оба сразу и важно, чтобы вся информация о сети, включая вертикальные зависимости между сетями, была записана в одну согласованную модель данных. В большинстве случаев топологическая информация о сети получается пуем обнаружения и топология является свойством сети, которое отражается в модели данных. Тем не менее, некоторые типы топологии должны быть также настраиваемыми приложением (например, в случае наложенной топологии).

В модели данных YANG для сетевой топологии все данные обозначаются как config true. Различие между данными, которые реально настраиваются, и данными которые действуют, включая обнаруженные данные, обеспечивается через хранилища, представленные как часть архитектуры NMDA[1] [RFC8342]. Обнаруженные данные сетевой топологии автоматически становятся частью хранилища рабочего состояния (<operational>). Они «управляются системой». Настраиваемая топология сети создается как часть хранилища конфигурации (<intended>) и только после ее вступления в силу эта топология становится частью рабочего хранилища (<operational>.

В общем случае настроенная сетевая топология будет указывать базовую топологию и включать информацию об уровнях, такую, как поддерживающие узлы, каналы и точки завершения. Поддерживающие объекты должны быть созданы в рабочем хранилище. Если настроенный элемент данных (такой как узел) имеет «подвешенную» ссылку на несуществующий элемент данных (например, поддерживающий узел), настроенный элемент будет автоматически удален из хранилища <operational> и отобразиться лишь в <intended>. Клиентское приложение может разрешить такую проблему и убедиться в корректности настройки ссылок на поддерживающие ресурсы.

Для каждой сети источник данных указывается в аннотации метаданных origin [RFC7952], определенной в [RFC8342]. В общем случае источником обнаруженных сетевых данных является обучение (learned), а настроенных - intended.

4.4.11. Идентификаторы в виде строк и URI

Текущая модель данных определяет идентификаторы узлов, сетей, каналов и точек завершения как URI. Дополнительно их можно определить в форме строк.

Дело в том, что строки проще реализовать. Причина выбора URI состоит в том, что топология, узел, точка завершения существует в более широком контексте, поэтому полезна возможность сопоставлять идентификаторы в разных системах. Хотя строки являются универсальным типом данных и проще для понимания человеком, с ними связано много путаницы. Обычно строки имеют некую структуру, которая задается «магически» и эту «магию» нужно передать каждой системе, работающей с данными. URI деляют структуру явной и обеспечивают дополнительную семантику, поскольку, в отличие от строк в свободной форме, могут быть переданы анализатору URI (resolver), который может указать дополнительные ресурсы, связанные с URI. Это свойство важно при интеграции топологических данных в более крупную и сложную систему.

5. Взаимодействие с другими модулями YANG

Эта модель данный использует типы данных, определенные в [RFC6991].

Это независимая от протоколов модель данных YANG с топологической информацией. Она отделена и не связана с моделями данных, служащими для настройки протоколов маршрутизации или маршрутной информации (например, модуль YANG ietf-routing [RFC8022]).

Модель данных соответствует требованиям к эфемерному состоянию, заданным в [RFC8242]. Для контролируемых системой эфемерных данных топологии процесс, которому поручено поддерживать топологическую информацию, будет загружать данные из процесса маршрутизации (например, OSPF) в рабочее хранилище данных, не полагаясь на конфигурационное хранилище.

6. Модули YANG

6.1. Определение абстрактной сети - ietf-network

```
<CODE BEGINS> file "ietf-network@2018-02-26.yang"

module ietf-network {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network";
```

[1]Network Management Datastore Architecture - архитектура хранилища данных сетевого управления.

```
  prefix nw;

  import ietf-inet-types {
    prefix inet;
    reference
      "RFC 6991: Common YANG Data Types";
  }

  organization
    "Рабочая группа IETF I2RS (интерфейс в систему маршрутизации)";

  contact
    "WG Web:    <https://datatracker.ietf.org/wg/i2rs/>
     WG List:   <mailto:i2rs@ietf.org>

     Editor:    Alexander Clemm
                <mailto:ludwig@clemm.org>

     Editor:    Jan Medved
                <mailto:jmedved@cisco.com>

     Editor:    Robert Varga
                <mailto:robert.varga@pantheon.tech>

     Editor:    Nitin Bahadur
                <mailto:nitin_bahadur@yahoo.com>

     Editor:    Hariharan Ananthakrishnan
                <mailto:hari@packetdesign.com>

     Editor:    Xufeng Liu
                <mailto:xufeng.liu.ietf@gmail.com>";

  description
    "Этот модуль определяет базовую модель данных общего назначения
     для набора узлов в сети. Определения узлов потом применяются в
     топологии и кадастре сети.

     Авторские права (Copyright (c) 2018) принадлежат IETF Trust
     и указанным авторам. Все права защищены.

     Распространение и использование с исходной или двоичной форме
     с внесением изменений или без них определяется лицензией
     Simplified BSD License, изложенной в разделе 4.c IETF Trust
     Legal Provisions для документов IETF
     (https://trustee.ietf.org/license-info).

     Данная версия модуля YANG является частью RFC 8345, где указаны
     дополнительные правовые аспекты.";

  revision 2018-02-26 {
    description
      "Первоначальный вариант.";
    reference
      "RFC 8345: A YANG Data Model for Network Topologies";
  }

  typedef node-id {
    type inet:uri;
    description
      "Идентификатор узла. Точная структура node-id будет
       определяться реализацией. Например, некоторые
       реализации МОГУТ выбрать URI с включением network-id
       как части пути. Идентификаторы СЛЕДУЕТ выбирать так,
       чтобы один и тот же узел в реальной сетевой топологии
       всегда указывался одним и тем же идентификатором, даже при
       размещении модели данных в раздельных хранилищах. Реализация
       МОЖЕТ фиксировать в идентификаторах некую семантику, например,
       для указания типа узла.";
  }

  typedef network-id {
    type inet:uri;
    description
      "Идентификатор сети. Точная структура network-id будет зависеть от
       реализации. Идентификаторы СЛЕДУЕТ выбирать так, чтобы одна и та же
```

```
        сеть всегда указывалась одним идентификатором даже при создании
        экземпляров модели в разных хранилищах данных. Реализация МОЖЕТ
        зафиксировать семантику идентификаторов (например, для указания
        типа сети).";
    }

    grouping network-ref {
      description
        "Содержит информацию, требуемую для ссылки на сеть (например, базовая сеть).";
      leaf network-ref {
        type leafref {
          path "/nw:networks/nw:network/nw:network-id";
        require-instance false;
        }
        description
          "Служит для ссылки на сеть (например, базовая сеть).";
      }
    }

    grouping node-ref {
      description
        "Содержит информацию, требуемую для ссылки на узел.";
      leaf node-ref {
        type leafref {
          path "/nw:networks/nw:network[nw:network-id=current()/../"+
            "network-ref]/nw:node/nw:node-id";
          require-instance false;
        }
        description
          "Служит для ссылки на узел. Узлы идентифицируются относительно сети,
           которая их содержит.";
      }
      uses network-ref;
    }

    container networks {
      description
        "Служит контейнером верхнего уровня для списка сетей.";
      list network {
        key "network-id";
        description
          "Описывает сеть. Сеть обычно содержит опись узлов, топологическую
           информацию (дополненную моделью данных топологии сети) и сведения
           об уровнях.";
        leaf network-id {
          type network-id;
          description
            "Идентифицирует сеть.";
        }
        container network-types {
          description
            "Служит в качестве цели дополнения. Тип сети указывается
             соответствующими контейнерами присутствия, добавленными
             в этот контейнер.";
        }
        list supporting-network {
          key "network-ref";
          description
            "Базовая сеть, испольуемая для представления многоуровневой топологии.";
          leaf network-ref {
            type leafref {
              path "/nw:networks/nw:network/nw:network-id";
            require-instance false;
            }
            description
              "Ссылка на базлвую сеть.";
          }
        }
        list node {
          key "node-id";
          description
            "Опись узлов сети.";
          leaf node-id {
            type node-id;
            description
              "Уникальный идентификатор узла в сети.";
```

```
          }
          list supporting-node {
            key "network-ref node-ref";
            description
              "Представляет другой узел, находящийся в базовой сети и
               поддерживающий данный узел. Служит для представления
               многоуровневой структуры.";
            leaf network-ref {
              type leafref {
                path "../../../nw:supporting-network/nw:network-ref";
              require-instance false;
              }
              description
                "Указывает базовую сеть, к которой относится базовый узел.";
            }
            leaf node-ref {
              type leafref {
                path "/nw:networks/nw:network/nw:node/nw:node-id";
              require-instance false;
              }
              description
                "Указывает сам базовый узел.";
            }
          }
        }
      }
    }
  }

  <CODE ENDS>
```

6.2. Определение топологии абстрактной сети - ietf-network-topology

```
  <CODE BEGINS> file "ietf-network-topology@2018-02-26.yang"

  module ietf-network-topology {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-network-topology";
    prefix nt;

    import ietf-inet-types {
      prefix inet;
      reference
        "RFC 6991: Common YANG Data Types";
    }
    import ietf-network {
      prefix nw;
      reference
        "RFC 8345: A YANG Data Model for Network Topologies";
    }

    organization
      "Рабочая группа IETF I2RS (интерфейс в систему маршрутизации)";

    contact
      "WG Web:    <https://datatracker.ietf.org/wg/i2rs/>
       WG List:   <mailto:i2rs@ietf.org>

       Editor:    Alexander Clemm
                  <mailto:ludwig@clemm.org>

       Editor:    Jan Medved
                  <mailto:jmedved@cisco.com>

       Editor:    Robert Varga
                  <mailto:robert.varga@pantheon.tech>

       Editor:    Nitin Bahadur
                  <mailto:nitin_bahadur@yahoo.com>

       Editor:    Hariharan Ananthakrishnan
                  <mailto:hari@packetdesign.com>

       Editor:    Xufeng Liu
                  <mailto:xufeng.liu.ietf@gmail.com>";

    description
      "Этот модуль определяет общую базовую модель сетевой топологии,
```

```
     дополняющую базовую модель данных сети каналами для подключения
     узлов, а также точками завершения каналов на узлах сети.

     Авторские права (Copyright (c) 2018) принадлежат IETF Trust
     и указанным авторам. Все права защищены.

     Распространение и использование с исходной или двоичной форме
     с внесением изменений или без них определяется лицензией
     Simplified BSD License, изложенной в разделе 4.c IETF Trust
     Legal Provisions для документов IETF
     (https://trustee.ietf.org/license-info).

     Данная версия модуля YANG является частью RFC 8345, где указаны
     дополнительные правовые аспекты.";

   revision 2018-02-26 {
     description
       "Первый выпуск.";
     reference
       "RFC 8345: A YANG Data Model for Network Topologies";
   }

   typedef link-id {
     type inet:uri;
     description
       "Идентификатор канала в топологии. Точная структура link-id
        зависит от реализации. Идентификаторы СЛЕДУЕТ выбирать так,
        что данный канал в реальной сетевой топологии всегда будет
        указываться одним идентификатором даже при размещении модели
        данных в раздельных хранилищах. Реализация МОЖЕТ выбрать
        семантику идентификаторов, например, указывать тип канала
        и/или тип топологии, к которой относится канал.";
   }

   typedef tp-id {
     type inet:uri;
     description
       "Идентификатор точки завершения на узле. Точная структура tp-id
        зависит от реализации. Идентификаторы СЛЕДУЕТ выбирать так,
        данная точка завершения всегда будет указываться одним
        идентификатором даже при размещении модели данных в
        раздельных хранилищах. Реализация МОЖЕТ выбрать семантику
        идентификаторов, например, указывать тип точки завершения
        и/или тип узла, к которому относится точка завершения.";
   }

   grouping link-ref {
     description
       "Эта группировка может служить для указания канала в
        конкретной сети. Хотя этот модуль не применяет группировку,
        она определена для удобства дополнения модулей.";
     leaf link-ref {
       type leafref {
         path "/nw:networks/nw:network[nw:network-id=current()/../"+
           "network-ref]/nt:link/nt:link-id";
         require-instance false;
       }
       description
         "Тип для абсолютной ссылки на экземпляр канала (этот тип не
          следует применять для относительных ссылок, используя для
          них относительный путь.)";
     }
     uses nw:network-ref;
   }

   grouping tp-ref {
     description
       "Эта группировка может служить для указания точки завершения
        на конкретном узле. Хотя этот модуль не применяет группировку,
        она определена для удобства дополнения модулей.";
     leaf tp-ref {
       type leafref {
         path "/nw:networks/nw:network[nw:network-id=current()/../"+
           "network-ref]/nw:node[nw:node-id=current()/../"+
           "node-ref]/nt:termination-point/nt:tp-id";
         require-instance false;
```

```
        }
        description
          "Тип для абсолютной ссылки на точку завершения
           (этот тип не следует применять для относительных ссылок,
           используя для них относительный путь.)";
      }
      uses nw:node-ref;
    }

    augment "/nw:networks/nw:network" {
      description
        "Add links to the network data model.";
      list link {
        key "link-id";
        description
          "Сетевой канал соединяет локальный (источник) и удаленный
           (получатель) узел через набор точек завершения
           соответствующих узлов. Можно иметь несколько каналов между
           парой узлов. Канал можно перемещать между точками
           завершения. Поэтому для однозначного различения каналов
           каждый из них указывается выделенным идентификатором.
           Отметим, что канал моделирует соединение «точка-точка», а
           не многоточечное.";
        leaf link-id {
          type link-id;
          description
            "Идентификатор канала в топологии. Канал связан с
             топологией, к которой он относится.";
        }
        container source {
          description
            "Контейнер, содержащий логический источник конкретного канала.";
          leaf source-node {
            type leafref {
              path "../../../nw:node/nw:node-id";
              require-instance false;
            }
            description
              "Идентификатор узла-источника. Должен быть в той же топологии.";
          }
          leaf source-tp {
            type leafref {
              path "../../../nw:node[nw:node-id=current()/../"+
                "source-node]/termination-point/tp-id";
              require-instance false;
            }
            description
              "Эта точка завершения размещается внутри узла-источника и
               завершает канал.";
          }
        }

        container destination {
          description
            "Контейнер, содержащий логического получателя конкретного канала.";
          leaf dest-node {
            type leafref {
              path "../../../nw:node/nw:node-id";
            require-instance false;
            }
            description
              "Идентификатор узла-получателя. Должен быть в той же топологии.";
          }
          leaf dest-tp {
            type leafref {
              path "../../../nw:node[nw:node-id=current()/../"+
                "dest-node]/termination-point/tp-id";
              require-instance false;
            }
            description
              "Эта точка завершения размещается внутри узла-получателя и
               завершает канал.";
          }
        }
        list supporting-link {
          key "network-ref link-ref";
```

```
        description
          "Указывает канал или каналы, от которых данный канал зависит.";
        leaf network-ref {
          type leafref {
            path "../../../nw:supporting-network/nw:network-ref";
          require-instance false;
          }
          description
            "Этот лист указывает топологию, в которой присутствует
             поддерживающий канал.";
        }

        leaf link-ref {
          type leafref {
            path "/nw:networks/nw:network[nw:network-id=current()/"+
              "../network-ref]/link/link-id";
            require-instance false;
          }
          description
            "Этот лист указывает канал, который является базы поддержки
             данного канала. Петли ссылок, где канал указывает себя
             как свою же базу напрямую или опосредованно, не разрешены.";
        }
      }
    }
  }
  augment "/nw:networks/nw:network/nw:node" {
    description
      "Дополняет точки завершения каналов. Эти точки могут в конце
       концов отображаться на интерфейсы.";
    list termination-point {
      key "tp-id";
      description
        "Точка завершения может быть окончанием канала. В зависимости
         от топологии такая точка может указывать, например, на порт
         или интерфейс.";
      leaf tp-id {
        type tp-id;
        description
          "Идентификатор точки завершения.";
      }
      list supporting-termination-point {
        key "network-ref node-ref tp-ref";
        description
          "Этот список указывает все точки завершения, от которых
           данная точка завершения зависит или куда отображается.
           Эти точки будут сами по себе содержаться в поддерживающем
           узле. Эта информация о зависимостях может быть выведена из
           зависимостей между каналами. Поэтому не требуется отдельно
           настраивать этот элемент и нет необходимости формулировать
           соответствующее ограничение. Нужная информация просто
           обеспечивается реализующей системой.";
        leaf network-ref {
          type leafref {
            path "../../../nw:supporting-node/nw:network-ref";
          require-instance false;
          }
          description
            "Этот лист указывает, в какой топологии присутствует
             поддерживающая точка завершения.";
        }
        leaf node-ref {
          type leafref {
            path "../../../nw:supporting-node/nw:node-ref";
          require-instance false;
          }
          description
            "Этот лист указывает, в каком узле присутствует
             поддерживающая точка завершения.";
        }
        leaf tp-ref {
          type leafref {
            path "/nw:networks/nw:network[nw:network-id=current()/"+
              "../network-ref]/nw:node[nw:node-id=current()/../"+
              "node-ref]/termination-point/tp-id";
            require-instance false;
```

```
            }
            description
              "Ссылка на базовый узел (этот узел должен находиться
               в другой топологии).";
          }
        }
      }
    }
  }

  <CODE ENDS>
```

7. Взаимодействие с IANA

Этот документ регистрирует перечисленные ниже URI пространств имен в реестре IETF XML Registry [RFC3688].

```
URI: urn:ietf:params:xml:ns:yang:ietf-network
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-network-topology
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-network-state
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-network-topology-state
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document registers the following YANG modules in the "YANG
Module Names" registry [RFC6020]:

Name:      ietf-network
Namespace: urn:ietf:params:xml:ns:yang:ietf-network
Prefix:    nw
Reference: RFC 8345

Name:      ietf-network-topology
Namespace: urn:ietf:params:xml:ns:yang:ietf-network-topology
Prefix:    nt
Reference: RFC 8345

Name:      ietf-network-state
Namespace: urn:ietf:params:xml:ns:yang:ietf-network-state
Prefix:    nw-s
Reference: RFC 8345

Name:      ietf-network-topology-state
Namespace: urn:ietf:params:xml:ns:yang:ietf-network-topology-state
Prefix:    nt-s
Reference: RFC 8345
```

8. Вопросы безопасности

Заданные в этом документе модули YANG определяют схему для данных, которые предназначены для доступа через сеть с помощью протоколов управления, таких как NETCONF [RFC6241] или RESTCONF [RFC8040]. Нижним уровнем NETCONF является защищенный сетевой транспорт с обязательной реализацией протокола SSH[1] [RFC6242], а нижним уровнем RESTCONF - протокол HTTPS с обязательной реализацией TLS [RFC5246].

Модель контроля доступа NETCONF [RFC8341] позволяет разрешать доступ к заданному подмножеству доступных операций и содержимого NETCONF или RESTCONF лишь уполномоченным пользователям NETCONF или RESTCONF.

Топология и опись сети, создаваемые этими модулями, раскрывают информацию о структуре сети, которая может помочь атакующим. Хотя в модулях и не определяется персональная информация, некоторые идентификаторы узлов могут быть связаны с конкретными устройствами, а те - с конечными пользователями, поэтому конфиденциальность также имеет значение.

Модули YANG определяют информацию, которая в некоторых экземплярах может быть настраиваемой, например для наложенных топологий, создаваемых клиентскими приложениями. В таких случаях вредоносный клиент может внести нежелательную топологию. В частности, он может попытаться удалить или добавить узел, канал или точку завершения путем создания нужных ему элементов в узле, канале или точке завершения, соответственно. В случае изучения топологии сервер будет автоматически предотвращать такие попытки. В случае настраиваемой топологии, т. е. когда источником данных является хранилище intended, нежелательная конфигурация может быть приведена в действие и отражена в рабочем хранилище, что приведет к нарушению обслуживания. Например, топология может быть «обрезана» (cut) или сделана неоптимальной, что ведет к росту потребления ресурсов базовой сети в результате неэффективности маршрутизации и использования пропускной способности. Кроме того, этом может вести к снижению

[1]Secure Shell - защищенная оболочка.

качества обслуживания и даже полному его прекращению. По этим причинам важно активно применять модель контроля доступа NETCONF для предотвращения изменений топологии несанкционированными клиентами.

В этих модулях данных YANG имеется множество узлов, для которых возможна запись, создание и/или удаление (значение config true, которое устанавливается по умолчанию). Такие узлы могут считаться деликатными или уязвимыми в некоторых сетевых средах. Операции записи (например, edit-config) в такие узлы без подобающей защиты могут оказать негативное влияние на работу сети. Ниже перечислены узлы и субдеревья, которые могут быть уязвимы.

В модуле ietf-network:

- network - вредоносный клиент может пытаться удалить или добавить сеть в целью удаления наложенной топологии или добавления ненужного наложения;
- supporting network - вредоносный клиент может пытаться нарушить логическую структуру модели, что приведет к отсутствию общей целостности и затруднит, например, поиск неполадок в многоуровневой топологии;
- node - вредоносный клиент может пытаться удалить или добавить узел, например, для нарушения работы наложенной топологии;
- supporting node - вредоносный клиент может пытаться сменить поддерживающий узел для нарушения уровней наложения.

В модуле ietf-network-topology:

- link - вредоносный клиент может пытаться удалить канал из топологии, добавить новый канал, манипулировать уровнем канала в поддерживающих каналах, а также менять источник или получателя канала; в любом случае структура топологии может быть нарушена и это может приводить к неоптимальной топологии наложения;
- termination point - вредоносный клиент может пытаться удалить точки завершения с узла, добавить «фантомные» точки на узел или изменить зависимости уровней для точек завершения с целью нарушения целостности топологии и возможного нарушения порядка операций наложения.

9. Литература

9.1. 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>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/info/rfc3688>.

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/info/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/info/rfc6991>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[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>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/info/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/info/rfc8342>.

9.2. Дополнительная литература

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

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

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG", RFC 7951, DOI 10.17487/RFC7951, August 2016, <https://www.rfc-editor.org/info/rfc7951>.

[RFC7952] Lhotka, L., "Defining and Using Metadata with YANG", RFC 7952, DOI 10.17487/RFC7952, August 2016, <https://www.rfc-editor.org/info/rfc7952>.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, DOI 10.17487/RFC8022, November 2016, <https://www.rfc-editor.org/info/rfc8022>.

[RFC8242] Haas, J. and S. Hares, "Interface to the Routing System (I2RS) Ephemeral State Requirements", RFC 8242, DOI 10.17487/RFC8242, September 2017, <https://www.rfc-editor.org/info/rfc8242>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/info/rfc8340>.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, DOI 10.17487/RFC8343, March 2018, <https://www.rfc-editor.org/info/rfc8343>.

[RFC8346] Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", RFC 8346, DOI 10.17487/RFC8346, March 2018, <https://www.rfc-editor.org/info/rfc8346>.

[USECASE-REQS] Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", Work in Progress, draft-ietf-i2rs-usecase-reqs-summary-03, November 2016.

[YANG-Push] Clemm, A., Voit, E., Gonzalez Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", Work in Progress, draft-ietf-netconf-yang-push-15[1], February 2018.

Приложение A. Примеры использования модели

A.1. Излечение топологии из элемента сети

В простейшей форме топологию изучает элемент сети (например, маршрутизатор) за счет участия в партнерских протоколах (IS-IS, BGP и т. п.). Определенная таким путем топология экспортируется (например в систему управления сетью NMS[2]) для внешних применений. Обычно у элемента сети в домене можно запросить его топологию и ожидать получения того или иного результата.

В несколько более сложной форме сетевой элемент может быть контроллером. Это может быть элемент с подключенными к нему или вспомогательными устройствами или контроллер в более распространенном смысле - специализированное устройство, предназначенное для координации действий множества других устройств (например, оптический контроллер). В таком случае контроллер является единым логическим элементом и должен запрашиваться определенным образом.

Следует отметить, что контроллеры могут создаваться на основе других контроллеров для создания топологии, включающей все домены сети.

Во всех перечисленных выше случаях топология изучается сетевым элементом, считается данными рабочего состояния. Т. е. данные извлекаются целиком в результате взаимодействия элемента сети с другими системами и могут динамически меняться без ввода и подтверждения.

A.2. Изменение топологии TE, импортированной из оптического контроллера

Рассмотрим случай, где Optical Controller представляет свою топологию клиентскому контроллеру пакетов в абстрактных терминах TE. Эта настраиваемая топология (которая объединяется с собственной топологией клиента) содержит достаточно данных, чтобы рассчитывающий пути клиент мог выбрать путь через оптический домен в соответствии с его политикой. Если клиент считает, что в данный момент) импортированная топология не соответствует в точности его требованиям, он может запросить изменение топологии. Такой запрос настройки может включать добавление или удаление элементов топологии или изменение атрибутов имеющихся элементов. Для оптического контроллера эти запросы транслируются в изменение настроек экспортируемой абстрактной топологии.

A.3. Аннотирование топологии для локальных расчетов

В некоторых вариантах определенную контроллером топологию нужно дополнить атрибутами до начала ее использования алгоритмами расчета. Рассмотрим случай, где рассчитывающему путь приложению на контроллере нужны географические координаты узлов при расчете пути. Если в определенной топологии нет этих координат, эти дополнительные атрибуты должны быть настроены в соответствующих элементах топологии.

A.4. Основанная на контроллере SDN конфигурация наложенной сети

В этом варианте контроллер SDN (например, Open Daylight) поддерживает представление топологии, которой он управляет, на основе информации, обнаруженной в сети. Кроме того, он обеспечивает приложение, в котором настраивается и поддерживается топология наложения.

Таким образом, контроллер SDN должен поддерживать две роли:

- клиент сети;
- сервер для своих северных приложений и клиентов, например системы поддержки операций OSS[3].

Т. е. клиент одной системы (в данном случае контроллер) может быть сервером (системой управления) другой.

В этом варианте контроллер SDN поддерживает модель консолидированных данных нескольких уровней топологии. Это включает нижние уровни, построенные на базе информации, полученной из сети, а также верхние уровни

[1]Доступен более свежий вариант документа. *Прим. перев.*
[2]Network Management System.
[3]Operations Support System.

топологии наложения, настраиваемые клиентом в контроллер, т. е. OSS. Для OSS нижние уровни предоставляют информацию, доступную лишь для чтения, а верхние должны разрешать чтение и запись.

Приложение B. Модели YANG для реализаций без поддержки NMDA

Определенные в этом документе модули YANG предназначены для применения с реализациями, поддерживающими архитектуру NMDA, определенную в [RFC8342]. Чтобы реализации могли использовать модель данных даже без поддержки NMDA, определены два используемых совместно модуля ietf-network-state и ietf-network-topology-state, представляющие рабочее состояние и топологию сети, соответственно. Эти модули «отражают» модули ietf-network и ietf-network-topology (параграфы 6.1 и 6.2), однако все их узлы не поддерживают настройку. Они представляют состояние, которое возникает при (1) получении топологической информации из сети или применения конфигурации из «отраженных» модулей.

Парные модули ietf-network-state и ietf-network-topology-state являются избыточными и их **не следует** включать в реализации с поддержкой NMDA, поэтому они определены в приложениях B.1 и B.2 (ниже), а не в основном тексте.

Структура этих модулей отражает структуру базовых модулей, поэтому дерево YANG для них не приводится.

B.1. Модуль YANG для состояния сети

```
<CODE BEGINS> file "ietf-network-state@2018-02-26.yang"

module ietf-network-state {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-network-state";
  prefix nw-s;

  import ietf-network {
    prefix nw;
    reference
      "RFC 8345: A YANG Data Model for Network Topologies";
  }

  organization
    "Рабочая группа IETF I2RS (интерфейс в систему маршрутизации)";

  contact
    "WG Web:    <https://datatracker.ietf.org/wg/i2rs/>
     WG List:   <mailto:i2rs@ietf.org>

     Editor:    Alexander Clemm
                <mailto:ludwig@clemm.org>

     Editor:    Jan Medved
                <mailto:jmedved@cisco.com>

     Editor:    Robert Varga
                <mailto:robert.varga@pantheon.tech>

     Editor:    Nitin Bahadur
                <mailto:nitin_bahadur@yahoo.com>

     Editor:    Hariharan Ananthakrishnan
                <mailto:hari@packetdesign.com>

     Editor:    Xufeng Liu
                <mailto:xufeng.liu.ietf@gmail.com>";

  description
    "Этот модуль определяет общую базовую модель данных для
     набора узлов в сети. Определения узлов затем применяются
     в сетевой топологии и описях (inventorY). Модуль представляет
     информацию, которая (1) изучается автоматически и заполняется
     или (2) является результатом применения сетевой информации,
     заданной в соответствии с моделью данных ietf-network,
     отражающей соответствующие узлы этой модели.

     Модель данных отражает ietf-network, но содержит лишь данные
     состояния, доступные только для чтения. Модель данных не
     требуется, когда инфраструктура базовой реализации
     поддерживает архитектуру NMDA.

     Авторские права (Copyright (c) 2018) принадлежат IETF Trust
     и указанным авторам. Все права защищены.

     Распространение и использование с исходной или двоичной форме
     с внесением изменений или без них определяется лицензией
     Simplified BSD License, изложенной в разделе 4.c IETF Trust
     Legal Provisions для документов IETF
```

```
      (https://trustee.ietf.org/license-info).

      Данная версия модуля YANG является частью RFC 8345, где указаны
      дополнительные правовые аспекты.";

  revision 2018-02-26 {
    description
      "Первый выпуск.";
    reference
      "RFC 8345: A YANG Data Model for Network Topologies";
  }

  grouping network-ref {
    description
      "Содержит информацию, необходимую для указания сети, например,
       указывает базовую сеть.";
    leaf network-ref {
      type leafref {
        path "/nw-s:networks/nw-s:network/nw-s:network-id";
      require-instance false;
      }
      description
        "Служит для указания сети, например, базовой.";
    }
  }

  grouping node-ref {
    description
      " Содержит информацию, необходимую для указания узла.";
    leaf node-ref {
      type leafref {
        path "/nw-s:networks/nw-s:network[nw-s:network-id=current()"+
          "/../network-ref]/nw-s:node/nw-s:node-id";
        require-instance false;
      }
      description
        "Служит для указания узла относительно содержащей его сети.";
    }
    uses network-ref;
  }

  container networks {
    config false;
    description
      "Служит контейнером верхнего уровня для списка сетей.";
    list network {
      key "network-id";
      description
        "Описывает сеть. Обычно сеть содержит опись узлов,
         топологическую информацию (дополненную моделью данных
         сетевой топологии) и информацию об уровнях.";
      container network-types {
        description
          "Служит целью дополнения. Тип сети указывается через
           соответствующие контейнеры присутствия, добавленные
           в этот контейнер.";
      }
      leaf network-id {
        type nw:network-id;
        description
          "Identifies a network.";
      }
      list supporting-network {
        key "network-ref";
        description
          "Базовая сеть, используемая для представления
           многоуроневой сетевой топологии.";
        leaf network-ref {
          type leafref {
            path "/nw-s:networks/nw-s:network/nw-s:network-id";
          require-instance false;
          }
          description
            "Указывает базовую сеть.";
        }
      }
```

```
      list node {
        key "node-id";
        description
          "Опись узлов данной сети.";
        leaf node-id {
          type nw:node-id;
          description
            "Однозначно указывает узел в содержащей его сети.";
        }
        list supporting-node {
          key "network-ref node-ref";
          description
            "Представляет другой узел, находящийся в базовой сети и
             поддерживающий данный узел. Служит для представления
             многоуровневой структуры.";
          leaf network-ref {
            type leafref {
              path "../../../nw-s:supporting-network/nw-s:network-ref";
            require-instance false;
            }
            description
              "Указывает базовую сеть, в которой находится базовый узел.";
          }
          leaf node-ref {
            type leafref {
              path "/nw-s:networks/nw-s:network/nw-s:node/nw-s:node-id";
            require-instance false;
            }
            description
              "Указывает сам базовый узел.";
          }
        }
      }
    }
  }
}

<CODE ENDS>
```

B.2. Модуль YANG для состояния сетевой топологии

```
  <CODE BEGINS> file "ietf-network-topology-state@2018-02-26.yang"

  module ietf-network-topology-state {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-network-topology-state";
    prefix nt-s;

    import ietf-network-state {
      prefix nw-s;
      reference
        "RFC 8345: A YANG Data Model for Network Topologies";
    }
    import ietf-network-topology {
      prefix nt;
      reference
        "RFC 8345: A YANG Data Model for Network Topologies";
    }

    organization
       "Рабочая группа IETF I2RS (интерфейс в систему маршрутизации)";

    contact
      "WG Web:    <https://datatracker.ietf.org/wg/i2rs/>
       WG List:   <mailto:i2rs@ietf.org>

       Editor:    Alexander Clemm
                  <mailto:ludwig@clemm.org>

       Editor:    Jan Medved
                  <mailto:jmedved@cisco.com>

       Editor:    Robert Varga
                  <mailto:robert.varga@pantheon.tech>

       Editor:    Nitin Bahadur
                  <mailto:nitin_bahadur@yahoo.com>
```

```
      Editor:     Hariharan Ananthakrishnan
                  <mailto:hari@packetdesign.com>

      Editor:     Xufeng Liu
                  <mailto:xufeng.liu.ietf@gmail.com>";

    description
      "Эта модель определяет общую базовую модель данных для состояния
       сетевой топологии, представляющую топологию, полученную (1) путем
       обучения или (2) в результате применения топологии, настроенной в
       модели данных ietf-network-topology и отражающей соответствующие
       узлы данных в этой модели. Она дополняет базовую модель данных
       состояния сети каналами для подключения узлов и точками
       завершения каналов на узлах.

       Модель данных отражает ietf-network-topology, но содержит лишь
       данные состояния, доступные только для чтения. Модель данных не
       требуется, когда инфраструктура базовой реализации
       поддерживает архитектуру NMDA.

       Авторские права (Copyright (c) 2018) принадлежат IETF Trust
       и указанным авторам. Все права защищены.

       Распространение и использование с исходной или двоичной форме
       с внесением изменений или без них определяется лицензией
       Simplified BSD License, изложенной в разделе 4.c IETF Trust
       Legal Provisions для документов IETF
       (https://trustee.ietf.org/license-info).

       Данная версия модуля YANG является частью RFC 8345, где указаны
       дополнительные правовые аспекты.";

    revision 2018-02-26 {
      description
        "Первый выпуск.";
      reference
        "RFC 8345: A YANG Data Model for Network Topologies";
    }

    grouping link-ref {
      description
        "Указывает канал в конкретной сети. Эта группировка не
         применяется в модуле и определена для удобства его дополнения.";
      leaf link-ref {
        type leafref {
          path "/nw-s:networks/nw-s:network[nw-s:network-id=current()"+
            "/../network-ref]/nt-s:link/nt-s:link-id";
          require-instance false;
        }
        description
          "Тип для абсолютной ссылки на экземпляр канала
           (этот тип не следует применять для относительных ссылок,
           используя для них относительный путь.)";
      }
      uses nw-s:network-ref;
    }

    grouping tp-ref {
      description
        "Указывает точку завершения в конкретном узле. Эта группировка не
         применяется в модуле и определена для удобства его дополнения.";
      leaf tp-ref {
        type leafref {
          path "/nw-s:networks/nw-s:network[nw-s:network-id=current()"+
            "/../network-ref]/nw-s:node[nw-s:node-id=current()/../"+
            "node-ref]/nt-s:termination-point/nt-s:tp-id";
          require-instance false;
        }
        description
          "Тип для абсолютной ссылки на точку завершения
           (этот тип не следует применять для относительных ссылок,
           используя для них относительный путь.)";
      }
      uses nw-s:node-ref;
    }
```

```
  augment "/nw-s:networks/nw-s:network" {
    description
      "Добавляет каналы в модель данных сети.";
    list link {
      key "link-id";
      description
        "Сетевой канал соединяет локальный (источник) и удаленный
         (получатель) узел через набор точек завершения
         соответствующих узлов. Можно иметь несколько каналов между
         парой узлов. Канал можно перемещать между точками
         завершения. Поэтому для однозначного различения каналов
         каждый из них указывается выделенным идентификатором.
         Отметим, что канал моделирует соединение «точка-точка», а
         не многоточечное.";
      container source {
        description
          "Контейнер, содержащий логический источник конкретного канала.";
        leaf source-node {
          type leafref {
            path "../../../nw-s:node/nw-s:node-id";
            require-instance false;
          }
          description
            "Идентификатор узла-источника. Должен быть в той же топологии.";
        }
        leaf source-tp {
          type leafref {
            path "../../../nw-s:node[nw-s:node-id=current()/../"+
              "source-node]/termination-point/tp-id";
            require-instance false;
          }
          description
             "Эта точка завершения размещается внутри узла-источника и
              завершает канал.";
        }
      }
      container destination {
        description
           "Контейнер, содержащий логического получателя конкретного канала.";
        leaf dest-node {
          type leafref {
            path "../../../nw-s:node/nw-s:node-id";
          require-instance false;
          }
          description
             "Идентификатор узла-получателя. Должен быть в той же сети.";
        }
        leaf dest-tp {
          type leafref {
            path "../../../nw-s:node[nw-s:node-id=current()/../"+
              "dest-node]/termination-point/tp-id";
            require-instance false;
          }
          description
             "Эта точка завершения размещается внутри узла-получателя и
              завершает канал.";
        }
      }
      leaf link-id {
        type nt:link-id;
        description
          "Идентификатор канала в топологии, к которой канал относится.";
      }
      list supporting-link {
        key "network-ref link-ref";
        description
          "Идентификатор канала или каналов, от которых данный канал зависит.";
        leaf network-ref {
          type leafref {
            path "../../../nw-s:supporting-network/nw-s:network-ref";
          require-instance false;
          }
          description
            "Этот лист указывает, в какой базовой топологии присутствует
             поддерживающий канал.";
        }
```

```
        leaf link-ref {
          type leafref {
            path "/nw-s:networks/nw-s:network[nw-s:network-id="+
              "current()/../network-ref]/link/link-id";
            require-instance false;
          }
          description
            "Это лист указывает канал, который является частью базы для
             данного канала. Петли, в которых канал указывает себя в
             качестве своей основы, напрямую или косвенно, не разрешены.";
        }
      }
    }
  }

  augment "/nw-s:networks/nw-s:network/nw-s:node" {
    description
      "Дополняет точки завершения каналов. Точки завершения в
       конце концов указывают на интерфейсы.";
    list termination-point {
      key "tp-id";
      description
        "Точка завершения может заканчивать канал. В зависимости от типа
         топологии точка завершения может указывать, например, на порт
         или интерфейс.";
      leaf tp-id {
        type nt:tp-id;
        description
          "Идентификатор точки завершения.";
      }
      list supporting-termination-point {
        key "network-ref node-ref tp-ref";
        description
          "Этот список указывает все точки завершения, от которых
           данная точка завершения зависит или куда отображается.
           Эти точки будут сами по себе содержаться в поддерживающем
           узле. Эта информация о зависимостях может быть выведена из
           зависимостей между каналами. Поэтому не требуется отдельно
           настраивать этот элемент и нет необходимости формулировать
           соответствующее ограничение. Нужная информация просто
           обеспечивается реализующей системой.";
        leaf network-ref {
          type leafref {
            path "../../../nw-s:supporting-node/nw-s:network-ref";
          require-instance false;
          }
          description
            "Этот лист указывает, в какой топологии присутствует
             поддерживающая точка завершения.";
        }
        leaf node-ref {
          type leafref {
            path "../../../nw-s:supporting-node/nw-s:node-ref";
          require-instance false;
          }
          description
            "Этот лист указывает, в каком узле присутствует
             поддерживающая точка завершения.";
        }
        leaf tp-ref {
          type leafref {
            path "/nw-s:networks/nw-s:network[nw-s:network-id="+
              "current()/../network-ref]/nw-s:node[nw-s:node-id="+
              "current()/../node-ref]/termination-point/tp-id";
            require-instance false;
          }
          description
            "Ссылка на базовый узел (этот узел должен находиться
             в другой топологии).";
        }
      }
    }
  }
}
<CODE ENDS>
```

Приложение C. Пример

В этом приложении представлен пример экземпляра дерева данные в коде JSON [RFC7951]. Пример создает экземпляр ietf-network-topology (и дополняемого им ietf-network) для топологии, показанной на рисунке 7. Имеется три узла - D1, D2, и D3. У D1 имеется 3 точки завершения (1-0-1, 1-2-1, 1-3-1), у D2 - тоже три (2-1-1, 2-0-1, 2-3-1), а у D3 - две (3-1-1 и 3-2-1). Кроме того, имеется 6 односторонних каналов, по паре разнонаправленных между каждыми двумя узлами.

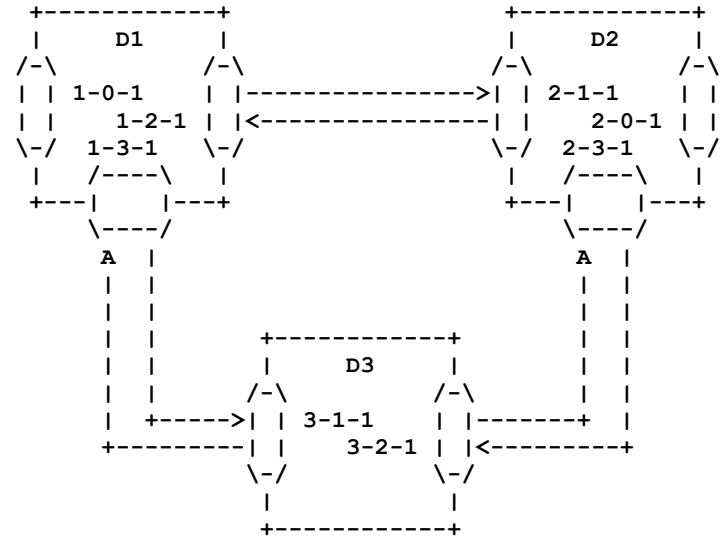

Рисунок 7. Пример топологии сети.

Соответствующий экземпляр дерева данных представлен на рисунке 8.

```
{
  "ietf-network:networks": {
    "network": [
      {
        "network-types": {
        },
        "network-id": "otn-hc",
        "node": [
          {
            "node-id": "D1",
            "termination-point": [
              {
                "tp-id": "1-0-1"
              },
              {
                "tp-id": "1-2-1"
              },
              {
                "tp-id": "1-3-1"
              }
            ]
          },
          {
            "node-id": "D2",
            "termination-point": [
              {
                "tp-id": "2-0-1"
              },
              {
                "tp-id": "2-1-1"
              },
              {
                "tp-id": "2-3-1"
              }
            ]
          },
          {
            "node-id": "D3",
            "termination-point": [
              {
                "tp-id": "3-1-1"
              },
              {
                "tp-id": "3-2-1"
              }
            ]
```

```
            }
          ],
          "ietf-network-topology:link": [
            {
              "link-id": "D1,1-2-1,D2,2-1-1",
              "source": {
                "source-node": "D1",
                "source-tp": "1-2-1"
              }
              "destination": {
                "dest-node": "D2",
                "dest-tp": "2-1-1"
              }
            },
            {
              "link-id": "D2,2-1-1,D1,1-2-1",
              "source": {
                "source-node": "D2",
                "source-tp": "2-1-1"
              }
              "destination": {
                "dest-node": "D1",
                "dest-tp": "1-2-1"
              }
            },
            {
              "link-id": "D1,1-3-1,D3,3-1-1",
              "source": {
                "source-node": "D1",
                "source-tp": "1-3-1"
              }
              "destination": {
                "dest-node": "D3",
                "dest-tp": "3-1-1"
              }
            },
            {
              "link-id": "D3,3-1-1,D1,1-3-1",
              "source": {
                "source-node": "D3",
                "source-tp": "3-1-1"
              }
              "destination": {
                "dest-node": "D1",
                "dest-tp": "1-3-1"
              }
            },
            {
              "link-id": "D2,2-3-1,D3,3-2-1",
              "source": {
                "source-node": "D2",
                "source-tp": "2-3-1"
              }
              "destination": {
                "dest-node": "D3",
                "dest-tp": "3-2-1"
              }
            },
            {
              "link-id": "D3,3-2-1,D2,2-3-1",
              "source": {
                "source-node": "D3",
                "source-tp": "3-2-1"
              }
              "destination": {
                "dest-node": "D2",
                "dest-tp": "2-3-1"
              }
            }
          ]
        }
      ]
    }
  }
```

Рисунок 8. Экземпляр дерева данных.

Благодарности

Мы хотели бы выразить признательность за полезный включа в работу, комментарии и предложения Alia Atlas, Andy Bierman, Martin Bjorklund, Igor Bryskin, Benoit Claise, Susan Hares, Ladislav Lhotka, Carlos Pignataro, Juergen Schoenwaelder, Robert Wilton, Qin Wu и Xian Zhang.

Участники работы

Многие люди в дополнение к перечисленным в разделе «Адреса авторов» внесли свой вклад в модель данных, представленную в этом документе. В число таких людей входят:

- Vishnu Pavan Beeram, Juniper;
- Ken Gray, Cisco;
- Tom Nadeau, Brocade;
- Tony Tkacik;
- Kent Watsen, Juniper;
- Aleksandr Zhdankin, Cisco.

Адреса авторов

Alexander Clemm

Huawei USA - Futurewei Technologies Inc.

Santa Clara, CA

United States of America

Email: ludwig@clemm.org, alexander.clemm@huawei.com

Jan Medved

Cisco

Email: jmedved@cisco.com

Robert Varga

Pantheon Technologies SRO

Email: robert.varga@pantheon.tech

Nitin Bahadur

Bracket Computing

Email: nitin_bahadur@yahoo.com

Hariharan Ananthakrishnan

Packet Design

Email: hari@packetdesign.com

Xufeng Liu

Jabil

Email: xufeng.liu.ietf@gmail.com

Перевод на русский язык

Николай Малых

nmalykh@gmail.com