Snowflake – интернет протокол с нуля

А. Калмацкий
Google LLC, Нью-Йорк, США
andrewkoder82@gmail.com

Н. А. Жукова
Санкт-Петербургский институт информатики и автоматизации Российской академии наук
Санкт-Петербург, Россия
nazhukova@mail.ru

И. А. Куликов
Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
Санкт-Петербург, Россия
i.a.kulikov@gmail.com

***Аннотация*. Статья посвящена разработке нового, полностью децентрализованного Интернет протокола. Данный протокол отличает простой дизайн, возможность естественного роста сети, добавлять внешние сети, новые узлы и сервисы без переконфигурирования других частей сети. Протокол не имеет логических ограничений. Переменная длина адреса позволяет использовать короткие для локальных соединений и длинные для более отдаленных.**

Ключевые слова: Snowflake protocol; Decentralized Internet protocol

I. Введение

Взаимодействие между различными локальными объектами (приложения, сети и пр.) является одной из основных задач децентрализованных вычислений. Разработка компактного децентрализованного протокола является актуальной задачей для многоагентных и других распределенных вычислительных систем. Snowflake протокол обеспечивает выигрыш за счет компактного дизайна, низкого потребления сетевого трафика и гибкого подхода к организации адресации.

II. Характеристики протокола

Snowflake протокол обладает следующими характеристиками:

- Snowflake протокол полностью децентрализованный.
- Протокол имеет простой дизайн: сетевые адреса, адреса подсетей / маски, порты, документированные url – все рассматривается как одно и то же.
- Это позволяет сетям расти естественно, присоединять существующие сети, добавлять новые хосты и сервисы без переконфигурирования остальных частей сети.
- Протокол не имеет логических ограничений. Адреса различной длины позволяют использовать короткие для локальной адресации и более длинные для адресации удаленных объектов. В сравнении с (4+2) * 2=12 адресных байт в IPv4 датаграммах [1].

Предлагаемый протокол функционирует на Сетевом уровне 3 модели OSI [2].

- Протокол доставляет датаграммы между клиентами, представляющими более высокие уровни OSI модели.
- Протокол спроектирован как множество роутеров, соединенных друг с другом каналами, представляющими более низкие уровни OSI модели.
- Здесь клиенты обозначаются одними и теми же именами для клиентов, серверов и Clients here are common name for clients, servers and сетевые узлы.
- С точки зрения алгоритма маршрутизации, клиенты является листьями роутеров.

Snowflake протокол базируется на двух идеях:

- связанная адресация,
- абсолютно полная децентрализация.

A. Сценарий 1. Один хост содержит несколько приложений, отправляющих сообщения друг другу

Шаги сценария:

- Клиенты соединены с роутером по выделенным каналам.
- Каналы идентифицированы числами.
- Каждый раз, когда клиент хочет отправить датаграмму другому клиенту, он использует список каналов как связанный адрес.

Диаграмма примера сценария (Клиент1 отправляет слово «hi» Клиенту2) представлена на рис. 1.

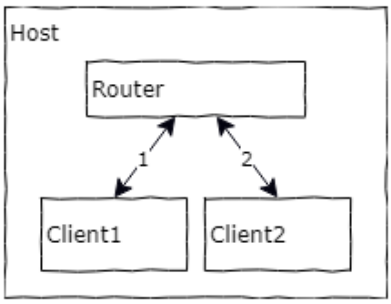

Рис. 1. Диаграмма примера сценария: Клиент1 отправляет слово «hi» Клиенту2

Например, Клиент1 отправляет слово «hi» Клиенту2:

- Клиент1 отправляет на роутер сообщение следующего формата {from: []; to: [2]; payload: “Hi”}, или в двоичном виде: 0x00_0102_024869.
- Роутер получает сообщение по Каналу1, он видит, что адрес получателя сообщения начинается с байта 2, и отправляет по Каналу 2 модифицированное сообщение {from: [1]; to: []; payload: “Hi”} в двоичном виде: 0x0101_00_024869.
- Клиент2 принимает сообщение имея не только содержание сообщения, но и хорошо отформатированный обратный адрес.

Таким образом, такая связанная схема адресации использует номера каналов как в традиционном IP протоколы используются номера портов.

B. Сценарий 2. Два хоста

Роутеры на различных хостах также соединены каналами связи. Эти каналы пронумерованы таким же образом, как каналы между роутером и клиентами.

Каналы между роутерами должны иметь уникальные номера на обеих сторонах. В рассматриваемом примере Канал 3 уникален на роутере на хостах 1 и 2.

Диаграмма примера сценария (Два хоста) представлена на рис. 2.

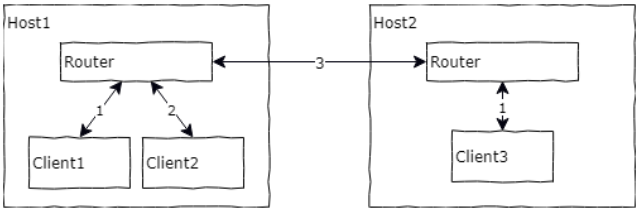

Рис. 2. Диаграмма примера сценария: Два хоста

Теперь Клиент2 на Хост1 передает слово «hi» Клиенту3 на Хост2:

- Клиент2 на свой роутер сообщение вида {from: []; to: [3,1]; payload: “Hi”}, или в двоичном виде: 0x00_020301_024869.
- Роутер Хоста1 принимает сообщение по Каналу 2, и так как адрес получателя начинается с байта 3, он отправляет по Каналу2 сообщение { from: [2]; to: [1]; payload: “Hi” }
- Роутер Хоста 2 принимает сообщение по Каналу 3. Адрес получателя начинается с байта 1, поэтому он направляет по Каналу 1 сообщение { from: [3, 2]; to: []; payload: “Hi” }
- Клиент3 получает сообщение с содержанием и с хорошо форматированным обратным адресом.

Этот пример показывает, что одна и та же схема адресации может универсально адресовать приложения внутри одного хоста, сетевые ресурсы, также при необходимости сервисы и документы внутри клиента могут адресоваться как URI.

C. Сценарий 3. Межсетевые соединения и сети с коммутацией

В этом сценарии имеются две сети. В каждой есть роутер. Они могут быть как отдельными устройствами, так и в виде приложения в одном из сетевых компьютеров.

Диаграмма примера сценария представлена на рис. 3.

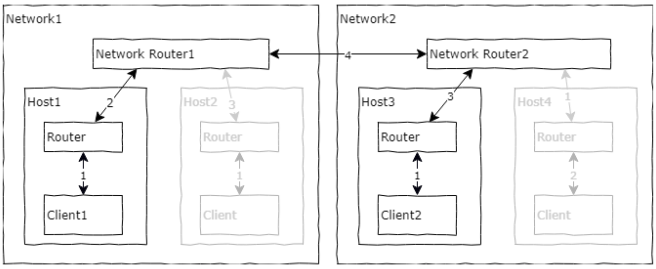

Рис. 3. Диаграмма примера сценария: Межсетевые соединения и сети с коммутацией

Как и в предыдущем примере, Клиент может:

- отправлять сообщения внутри своего хоста, используя 1-байтовую адресацию;
- или взаимодействовать с любыми роутерами внутри сети с помощью 2-байтовой адресации;
- или взаимодействовать с любыми сервисами в домашней сети при помощи 3-байтового адреса: например, Клиент1 может соединяться с Клиентом на Хосте2 используя адрес [2,3,1].

Не имеет значения, какую топологию и архитектуру имеет сеть 1 и сеть 2, мы можем просто и прозрачно соединить их. Все что нам надо, это выбрать любые два подходящих роутера в обеих сетях и построить канал между ними. Как и в предыдущем примере, этот канал должен иметь уникальный номер для обоих роутеров.

Теперь Клиент1 может соединиться с Клиентом2 из другой сети, используя адрес [2,4,3,1], и обратный адрес будет таким: [3,4,2,1].

D. Сценарий 4. Сервер A передает адрес Ресурса B Клиенту C

Диаграмма примера сценария приведена на рис. 4.

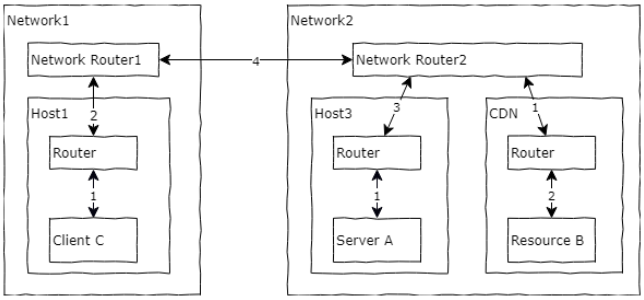

Рис. 4. Диаграмма примера сценария: Сервер A передает адрес Ресурса B Клиенту C

Например:

- Клиент C передает запрос на Сервер A: {to: [2,4,3,1]; payload: “get resource b”}.
- Сервер A принимает запрос в виде: {from: [3,4,2,1]: payload: “get resource b”}. Он знает, что Ресурс B расположен по адресу [3,1,2], и он отвечает с сообщением {to: {3,4,2,1}; payload: [3,1,2]}
- Клиент получает ответ: {from:[2,4,3,1]; payload: [3,1,2]}.
- Теперь клиенту необходимо вычислить адрес Ресурса B относительно Клиента C (CB) из предоставленного относительного адреса AB [312] и относительного адреса CA [2431]. Это может быть выполнено в три шага:
  - Удалить последний шаг из CA: CA’ = [243], AB=[312],
  - пока CA.tail == AB.head, удалять начало и конец: CA”=[24], AB’=[12]
  - склеить CA” and AB’: result=[2412].

С момента, когда появилась возможность передавать адресную информацию между сетевыми узлами, мы можем создать сервера имен, которые будут преобразовывать человеко-читаемые сетевые имена во внутренние адреса.

E. Дополнительные возможности

- Если все 128 каналов на роутере определены, дополнительные каналы кодируются 2-байтовыми числами – 128.0, 128.1...
- Если оба роутера с обеих сторон канала связи не имеют свободного одинакового номера, также используются 2-байтовые номера (от пары свободных номеров).
- Если вторичный маршрут создан между сетями, этот канал получает номер, как адрес одного конца канала относительно второго. Также сервисный пакет на добавление маршрута в таблицу маршрутов отправляется на оба роутера.
- Если канал перестал функционировать, сервисный пакет на удаление маршрута отправляется на оба роутера. Маршрут по умолчанию также должен быть удален.

III. Заключение

В статье был представлен новый Интернет протокол – Snowflake. Описаны четыре работоспособных сценария и дополнительные возможности протокола. В качестве дальнейших исследований, следующие аспекты должны быть изучены подробно:

- Если на роутере закончатся каналы (полиндромная адресация).
- Создание вторичного канала между сетями (таблица маршрутизации).
- Падение основного канала связи.
- Каналы «многие ко многим».

Snowflake Интернет протокол способен обеспечить оптимальные протокол взаимодействия между агентами, объектами, локальными сетями, устройствами и пр. Минимизация Интернет трафика способна помочь в использовании современных децентрализованных технологий в ситуации, когда высокоскоростной Интернет не доступен. Кроме того, мы можем говорить, что рассмотренные дополнительные возможности позволяют обойти лимит в 128 каналов для роутера, что позволяет использовать протокол Snowflake для соединения большего числа клиентов (устройств, локальных сетей, агентов и пр.)

Список литературы

[1] https://tools.ietf.org/html/rfc760
[2] https://www.iso.org/ics/35.100/x/
[3] https://tools.ietf.org/html/rfc8200