Ubiquitous overlay

Universal connectivity using imperfect hardware

Matouš Skála

Ubiquitous overlay

Universal connectivity using imperfect hardware

by

Matouš Skála

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday January 1, 2013 at 10:00 AM.

Student number:	4893964	
Project duration:	November 15, 2019 – June 30, 2020	
Thesis committee:	Dr.ir. J.A. Pouwelse,	TU Delft, supervisor
	?,	TU Delft
	?,	TU Delft

An electronic version of this thesis is available at
http://repository.tudelft.nl/.

TU Delft

Preface

Preface…

Matouš Skála
Delft, January 2013

Contents

1 Introduction **1**

2 Problem Description **3**
2.1 Ubiquitous Overlay Network . . . 3
2.2 Network Address Translation . . . 3
2.2.1 NAT Classification . . . 3
2.2.2 Carrier Grade NAT . . . 3
2.2.3 Port Forwarding . . . 3
2.3 Nearby Communication . . . 3
2.4 Peer Discovery . . . 4

3 State of the Art **5**
3.1 NAT Traversal . . . 5
3.1.1 Session Traversal Utilities for NAT (STUN) . . . 5
3.1.2 Traversal Using Relays around NAT (TURN) . . . 5
3.1.3 Interactive Connectivity Establishment (ICE) . . . 5
3.1.4 ICMP Hole Punching . . . 5
3.1.5 Symmetric NAT Traversal . . . 5
3.2 P2P Communication Libraries . . . 5
3.2.1 libp2p . . . 5
3.2.2 IPv8 . . . 5
3.2.3 Nearby Connections API . . . 5
3.2.4 Bridgefy SDK . . . 5

4 Design **7**
4.1 NAT Traversal with Peer Introductions . . . 7
4.2 Relay Protocol with Bandwidth Accounting . . . 7
4.3 Nearby Communication over Bluetooth . . . 7
4.3.1 BLE Advertising . . . 7
4.3.2 BLE Scanning . . . 7
4.3.3 GATT Server . . . 7
4.3.4 Discovery Strategy . . . 7

5 Implementation **9**
5.1 Project Structure . . . 9
5.2 System Architecture . . . 9
5.2.1 Communities . . . 9
5.2.2 Discovery Strategies . . . 9
5.2.3 Endpoints . . . 9

5.3 Bootstrap Server . . . 9
5.4 Maintaining Backward Compatibility . . . 9
5.5 TrustChain Explorer . . . 9
5.6 Binary Transfer over UDP . . . 9
5.7 PeerChat: Distributed Messenger . . . 9
5.8 Testbed for Distributed Android Applications . . . 9

6 Experiment **11**
6.1 Analysis and Puncturing of Carrier Grade NAT . . . 11
6.2 Performance Evaluation . . . 12
6.2.1 Bootstrap Performance . . . 12
6.2.2 Stress Test . . . 12

7 Conclusion **13**
7.1 Future Work . . . 13

Bibliography **15**

1

Introduction

The Internet was created with the idea that any two computers connected to the common network should be able to communicate with each other. In *Internet Protocol version 4 (IPv4)* which routes most traffic today, each computer gets assigned an address which is subsequently used for packet routing. As IPv4 uses a 32-byte address space, it is not feasible to assign a unique address to every device on the planet. To deal with IPv4 address exhaustion, internet providers were forced to deploy *Network Address Translation (NAT)*, which allows a single address to be shared across multiple devices behind a NAT device.

Another issue appeared with the rise of portable computers and smartphones. IPv4 addresses are dependent on the physical location and can not be in any way considered as stable user identifiers. There has been several proposals including Mobile IP, IPSec, and IPv6 to improve the usability and security of the Internet Protocol. However, none of them have been widely deployed yet or address all known issues.

This thesis proposes and implements a decentralized protocol for peer to peer communication. The protocol allows any two devices to establish a direct connection by taking advantage of NAT traversal techniques to connect to peers behind NATs. When the Internet connection is not available and peers are located in proximity, the connection can be established using Bluetooth. Peers are addressed by their public keys and their physical addresses on lower layers are abstracted away.

The protocol makes best effort to connect peers behind NATs. In case the connection is not possible, it resorts to a relay protocol. Bandwidth accounting prevents misusing the relay servers and provides incentive for relay operators. The protocol is completely decentralized and does not rely on any central entity.

To show one of many practical use cases of the protocol, a simple chat messaging application is implemented on top of it. It allows to send not only text messages, but also images and videos to demonstrate binary file transfer.

Compared to the state of the art solutions, it combines both nearby and Internet connectivity, does not require any central server, and is completely open source. Finally, the protocol performance is experimentally evaluated with multiple Android devices connected to different Wi-Fi and carrier networks and running a stress test over the period of 24 hours.

2

Problem Description

2.1. Ubiquitous Overlay Network

2.2. Network Address Translation

2.2.1. NAT Classification

2.2.2. Carrier Grade NAT

2.2.3. Port Forwarding

2.3. Nearby Communication

Modern smartphone devices come equipped with several wireless communication standards that can potentially be used for communication with other nearby devices. It is desirable to use such a technology when multiple devices in proximity want to communicate with each other when there is no reliable Internet infrastructure available. These technologies can be also preferred over the Internet in case of censorship and privacy concerns, as has been shown during numerious occasions such as Hong Kong protests. From the user experience perspective, it is desired that the device discovery and connection establishment does not require any user interaction besides the one required by the application use case. However, this is often opposed by the security model of smartphone operating systems which try to protect users by enforcing system UI for any sensitive operations.

Probably the oldest and the most battle-tested technology is Bluetooth, which has been in development for over more then 20 years already. The common flow for Bluetooth usage is to first force user to pair two devices. Only then, *Radio frequency communication (RFCOMM)* channel can be established between paired devices. This process requires user interaction which degrades the user experience in certain applications when authentication is performed on the application level. While it is possible to establish an *insecure* RFCOMM channel if the MAC address of the other device is known, the user of the other device still needs to manually set it to be *discoverable*.

In Bluetooth 4.0 standard, *Bluetooth Low Energy (BLE)* has been introduced. It is a completely different communication protocol incompatible with the classic Bluetooth, which we will further refer to as *Bluetooth Classic*. Compared to Bluetooth

Classic, BLE offers considerably decreased power consumption with a similar communication range. Its was originally intended to support an infrequent low-power communication with wearables, healthcare accessories, or smart home appliances. However, while it is not a primary use case, it could also be potentially used for low-bandwidth peer-to-peer communication between smartphone devices. It is notable that once user allows Bluetooth permission, the application can fully control BLE APIs without any user interaction, which opens up doors for a range of many different applications.

2.4. Peer Discovery

3

State of the Art

3.1. NAT Traversal

3.1.1. Session Traversal Utilities for NAT (STUN)

3.1.2. Traversal Using Relays around NAT (TURN)

3.1.3. Interactive Connectivity Establishment (ICE)

3.1.4. ICMP Hole Punching

3.1.5. Symmetric NAT Traversal

3.2. P2P Communication Libraries

3.2.1. libp2p

3.2.2. IPv8

3.2.3. Nearby Connections API

3.2.4. Bridgefy SDK

4

Design

4.1. NAT Traversal with Peer Introductions

4.2. Relay Protocol with Bandwidth Accounting

4.3. Nearby Communication over Bluetooth

4.3.1. BLE Advertising

4.3.2. BLE Scanning

4.3.3. GATT Server

4.3.4. Discovery Strategy

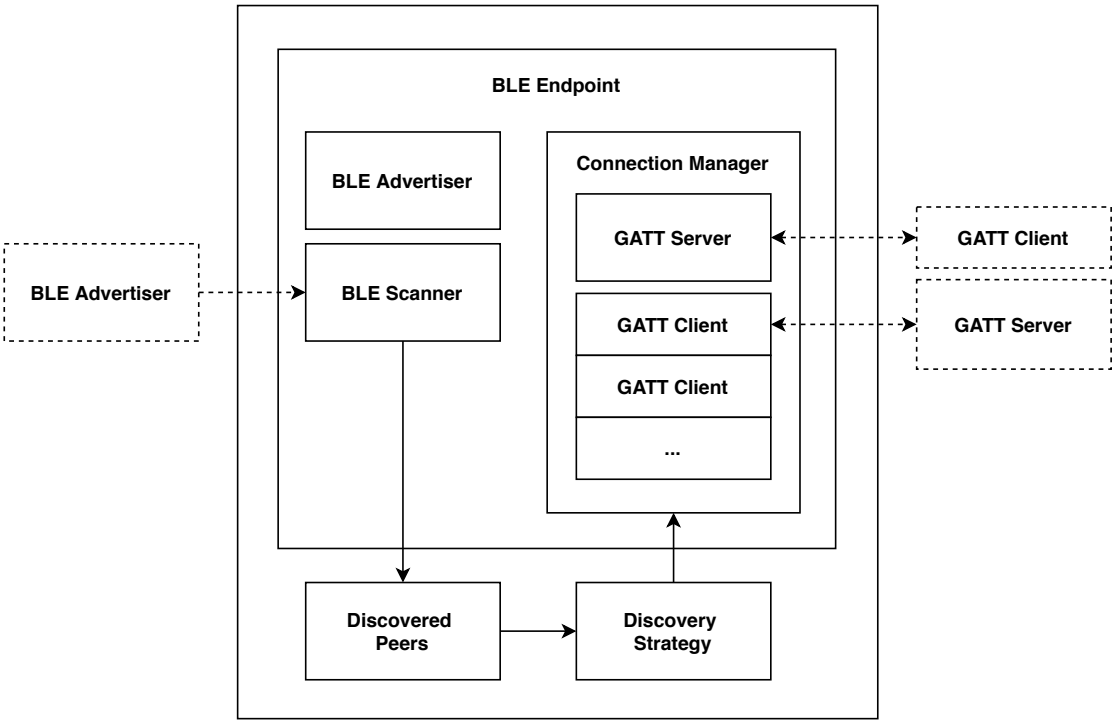

Figure 4.1: Bluetooth Low Energy Communication Architecture

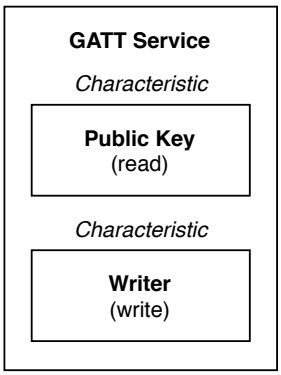

Figure 4.2: GATT Server Architecture

5

Implementation

5.1. Project Structure
5.2. System Architecture
5.2.1. Communities
5.2.2. Discovery Strategies
5.2.3. Endpoints
5.3. Bootstrap Server
5.4. Maintaining Backward Compatibility
5.5. TrustChain Explorer
5.6. Binary Transfer over UDP
5.7. PeerChat: Distributed Messenger
5.8. Testbed for Distributed Android Applications

6

Experiment

6.1. Analysis and Puncturing of Carrier Grade NAT

According to the report by Statista [2], there were three major mobile phone operators providing services in the Netherlands in Q4 2018. They are listed in Table 6.1. In total, these represent up to 85 % of the market share. The rest of the market is shared by Mobile Virtual Network Operators who sell services over existing networks of those three operators.

Operator	**Market share**
KPN	35%
Vodafone	25%
Mobile Virtual Network Operators	25%
T-Mobile	20%

Table 6.1: Market share of mobile network operators in the Netherlands in Q4 2018. The shares do not sum up to 100% as they are rounded up within five percent ranges in the original report. [2]

We have purchased pre-paid SIM cards for all three major mobile network operators to investigate whether they are suitable for peer-to-peer communication. First, we tried to infer the characteristics of their Carrier Grade NAT deployments.

We used the STUN protocol and NAT behavior discovery mechanisms described in [1]. They have shown that all networks appear to use *Endpoint-Independent Mapping (EIM)* and *Address and Port-Dependent Filtering* (also known as *port-restricted cone NAT*). EIM is a sufficient condition for our NAT traversal mechanism to be successful, so this would make all these NATs suitable for P2P communication.

However, as NAT behavior can change over time, we performed some more tests to verify that the behavior is consistent over time. We attempted to connect to 50 different peers over the interval of 5 minutes. We verified that KPN and T-Mobile networks are consistent with EIM behavior. However, the Vodafone network changes the mapped port for new connections approximately every 60 seconds, even when connecting to the same IP address and a different port. This behavior can be described as *Address and Port-Dependent Mapping*, which is characteristic for a *symmetric NAT*.

The mapped ports seem to be assigned at random from the range of 10,000 ports, which makes it infeasible to use any known symmetric NAT traversal techniques such as port prediction or multiple hole punching [4][3].

6.2. Performance Evaluation

6.2.1. Bootstrap Performance

6.2.2. Stress Test

7

Conclusion

7.1. Future Work

Bibliography

[1] D. MacDonald and B. Lowekamp. NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN). RFC 5780, May 2010. URL https://tools.ietf.org/html/rfc5780.

[2] Statista. Distribution of mobile network connections in the netherlands in the fourth quarter of 2018, by operator. Accessed: Mar. 11, 2020. URL https://www.statista.com/statistics/765491/distribution-mobile-network-connections-in-the-netherlands-by-operator/.

[3] Y. Takeda. Symmetric NAT traversal using STUN. Technical report, June 2003. URL https://tools.ietf.org/id/draft-takeda-symmetric-nat-traversal-00.txt.

[4] Yuan Wei, Daisuke Yamada, Suguru Yoshida, and Shigeki Goto. A new method for symmetric NAT traversal in UDP and TCP. 2008.