TCP Hole Punching Approach to Address Devices in Mobile Networks

Satish Narayana Srirama
Mobile & Cloud Lab, Institute of Computer Science
University of Tartu
J. Liivi 2, Tartu, Estonia
Email: srirama@ut.ee

Mohan Liyanage
Mobile & Cloud Lab, Institute of Computer Science
University of Tartu
J. Liivi 2, Tartu, Estonia
Email: liyanage@ut.ee

***Abstract*—The emergence of mobile terminals with enhanced features like high processing power, more memory, inbuilt sensors, low power consumption, etc. have lead to their extensive usage in different domains like mobile social networking, mobile cloud and Internet of Things (IoT). However, to successfully utilize these devices as information providing/processing entities, we need proper means of identification and addressing, so that the devices and their offered data/services are accessible also from outside the mobile network. Addressing devices in mobile networks has been studied extensively over the years and there are several solutions one can consider. However, the most popular and widely used addressing mechanism for internet, IP address, is also being extensively used in mobile data networks (3G/4G). Typically, the IP address assigned to mobile devices ends up with barriers like their temporarily availability, known only within the mobile operator's network, Network Address Translation (NAT) etc., which makes it difficult to use the addresses in IoT scenarios. To address these problems we propose to use UDP sessions with a Rendezvous server to send and receive peer information and TCP sessions established in a hole punching process over NAT, so that information can be exchanged over these devices. The approach is explained in detail along with its prototype implementation and conceptual limitations.**

Keywords*—*Hole Punching, NAT, 3G Mobile Network, Addressing devices.

I. Introduction

In recent years, smart phones have entered the market with enhanced features like high processing power, more memory, HD video, inbuilt sensors, touchscreen, low power consumption, etc. There are many new applications and services developed for mobile users and the mobile phone can even be used to provide services to other mobile users [1], [2]. Similarly, now there are several mobile Peer to Peer (P2P) and mobile social networking applications such as playing online games, sharing photos and videos with friends, teacher sharing blackboard with a group of students etc. While the applications are interesting, for successful adoption of these mobile P2P and social applications, the mobile terminal, that is registered and connected to the mobile operator network, requires some means of identification and addressing. The problem is also relevant in the Internet of Things (IoT) domain where every device is connected and can act as an information providing/processing entity.

Addressing devices in mobile networks has been studied extensively over the years and there are several solutions one can consider. Technologies such as OpenID [3], Session Initiation Protocol (SIP) [4], Extensible Messaging and Presence Protocol (XMPP) [5], Zero-configuration networking (zeroconf) [6] and Universal Plug and Play (UPnP) [7] have offered solutions for addressing devices in different P2P setups. Although there are different addressing mechanisms proposed, the most popular and widely used addressing mechanism for internet is Internet Protocol Address (IP address). IP addressing provides a globally unique 32 bit address for each and every device connected to the internetwork. Network operators always have provided different means to address the devices with IP addresses, so that they could be accessed from the Internet [8]. However, IP address space is limited and have mostly exhausted due to drastically growing number of devices attached to the internet. Internet Protocol version 6 (IPv6) [9] is the latest revision of the IP, developed by the Internet Engineering Task Force (IETF), to deal with the long-anticipated problem of IPv4 address exhaustion. However its adaption is still in its infancy.

As a temporary solution for the insufficient number of usable public IP addresses, the IETF introduced a new private IP address space [10] for the internal usage of any organization. Normally many organizations use this private IP address space to assign IP addresses for their intranet devices. However, when these devices need to communicate with a device outside the network, private IP address should be translated to the public IP address. This process is called Network Address Translation (NAT). Although private IP address space is a good solution for the problem under consideration, NAT causes several difficulties for mobile P2P communication. NAT boxes mostly drop the unsolicited inbound traffic and only allow if the traffic is a reply for the session initiated by the device within the network.

To tackle the problem we propose to use UDP (User Datagram Protocol) sessions with a Rendezvous server to send and receive peer information and TCP (Transmission Control Protocol) sessions established in a hole punching process over NAT. There are several NAT traversal techniques that have been proposed to establish a connection when hosts are behind the NAT Box. Most of these solutions were designed for the hosts in Wi-Fi or on Fixed LAN environment. But the solution proposed here is mainly targeted at addressing Android devices in the mobile 3G network. The paper is organized as follows:

Section 2 introduces the background details needed for TCP hole punching. Section 3 describes our proposed solution for addressing devices in mobile networks. Section 4 later provides the observed results and section 5 discusses the results along with the limitations of the approach. Section 6 provides the related work while section 7 concludes the paper with some future research directions.

II. Addressing Devices in Mobile Networks: Taxonomy and Current Technologies

A. *Mobile 3G network*

The Third Generation (3G) of the mobile networks is becoming very popular because of the ability to access the internet over mobile network. 3G networks operate under the standards provided by the International Telecommunications Union's (ITU) - International Mobile Telecommunications-2000 (IMT-2000) [11]. The Universal Mobile Telecommunications System (UMTS) is one of the common 3G mobile communication systems widely being using in many countries. According to this standard minimum data rate for the stationary or walking users is 2Mbit/s and for a moving vehicle is 348 Kbit/s. Mobile 3G also has high data rate up to 10Mbps with the HSDPA technology (High Speed Downlink Packet Access) [12]. With these high rates, the 3G network enables voice and video calling, file transmission, internet surfing, online TV, viewing high definition videos, playing games and much more for their users.

B. *IP Address Assignment in the Mobile Networks*

Hierarchical architecture is used to design the core network of the mobile communication systems. In UMTS the data traffic generated by each mobile traverses through the UMTS Terrestrial Radio Access Network (UTRAN) and moves to the packet-switched domain. From the packet-switched domain the data moves to the internet or other mobile networks. In such a scenario, each mobile device gets IP address from the Gateway GPRS Support Node (GGSN) located at the edge of the packet switched domain. GGSN works as an interface for the mobile network and the internet. It is impossible to provide a public IP address for each and every mobile device because there are thousands of subscribers registered under one mobile network operator. As a solution for this, GGSN assigns private unroutable IP address for the mobiles and translates this private IP address to the public IP address when the user needs to access the internet.

C. *Network Address Translation*

Network Address Translation (NAT) is being used by many service providers and organizations as a solution for the insufficient number of usable public IP addresses. Originally NAT was introduced as a short-term solution for the IP address shortage, but is prone to some problems in networking. There are few IP address blocks were defined as Private IP addresses. These IP addresses are also called non routable as data packets with these addresses can't travel over internet. Since the scope is private many organizations can use same private IP addresses within their networks. But when the device needs to communicate with the devices on public network this private IP address should be converted to the public IP address. At this point NAT process is involved and maps internal private addresses to the external public addresses. An internal IP

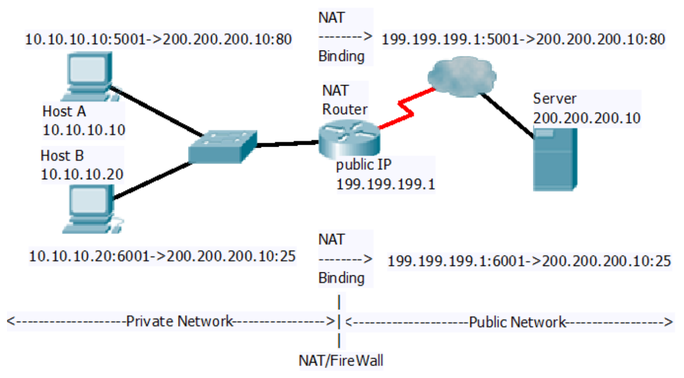


Fig. 1. NAT process

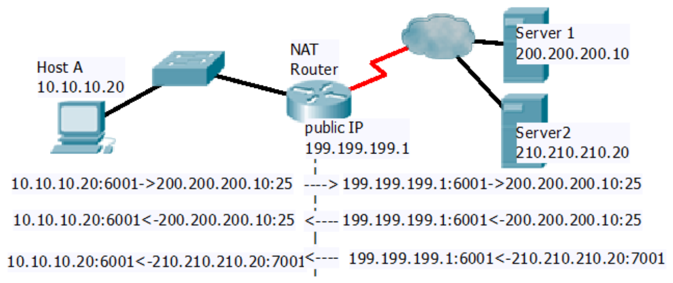


Fig. 2. Full Cone NAT

address and the port (Internal Socket) is mapped to an external IP:port pair (External Socket). Whenever the NAT receives a packet with the external IP:port, it uses the map to reroute the packet back to the internal device which is using matched internal socket (illustrated in figure 1). Using this method several private IP addresses can be mapped to one public IP address. Depending on the method of the translation it is called Network Address Translation (NAT) or Network Address Port Translation (NAPT) [13] .

D. *Commonly used NAT Translation Techniques*

Although there is no particular standard defined for the NAT process there are four commonly used translation techniques [14] in the industry.

1) Full Cone NAT: In full cone NAT, the internal IP address and port are mapped to an external IP and port. In this method, whenever inbound traffic comes to external port it is just forwarded to the matching internal device without doing any filtering. This is prone to some security risks as anybody from outside can send data to the internal device if he knows the external port of the mapping (illustrated in figure 2).

2) Address Restricted Cone NAT: In Address Restricted Cone NAT, internal IP and port are mapped to the external IP and port. But for inbound traffic NAT box checks the source IP address of the packet. It allows the traffic coming from the source only if it is already addressed by the internal host. Any unsolicited packets from outside to external port are blocked. This provides additional security but causes difficulty in connecting P2P applications (illustrated in figure 3).

3) Port Restricted Cone NAT: This case is almost similar to Address Restricted Cone NAT. In addition; it also checks

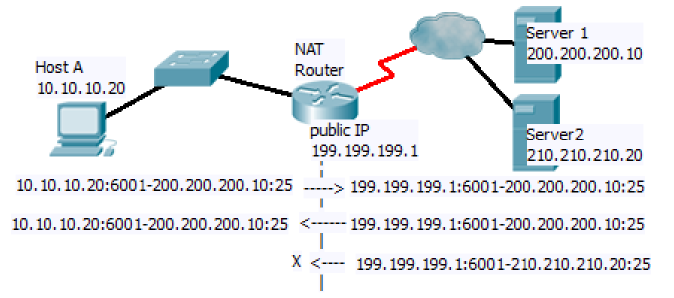


Fig. 3. Address Restricted Cone NAT

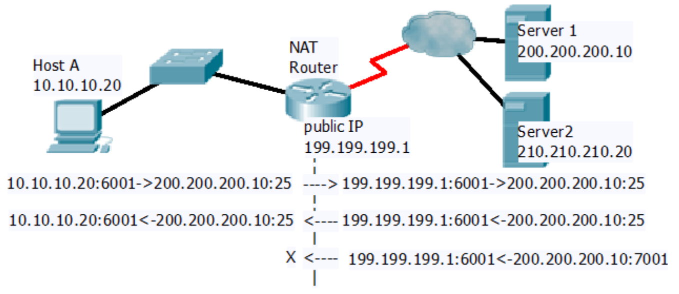


Fig. 4. Port Restricted Cone NAT

the port number of the inbound traffic. It allows the packet only if the source IP and port of the packet match with the mapped public socket (illustrated in figure 4).

4) Symmetric NAT: This is the most restricted mapping method in comparison with other three methods. In all the above cases the internal socket's port number is directly mapped to the public socket's same port number. But in symmetric NAT the internal socket is mapped to a completely different new external socket. If the internal device uses same socket to send data to a different external host, the NAT device maps it with another new external socket. This prevents any type of unsolicited inbound traffic. But it is very hard to predict the port mappings in this NAT process even though we know the internal socket details. In Symmetric NAT only the external device that receives a packet from internal device can send a packet back to the internal host. The scenario is illustrated in figure 5.

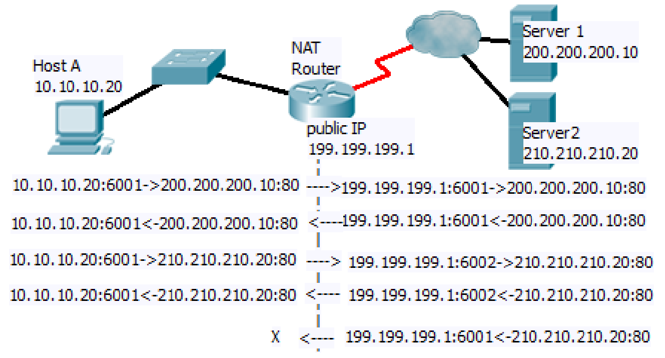


Fig. 5. Symmetric NAT

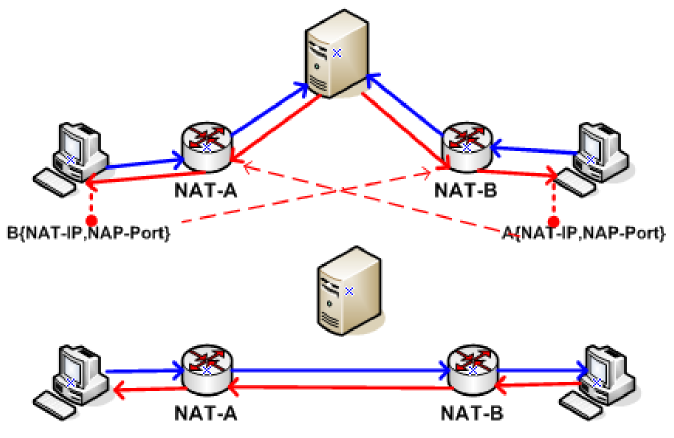


Fig. 6. Hole punching with Rendezvous server

5) TCP Hole Punching: When hosts are behind the NAT boxes it is impossible to establish connection from the outside network because NAT boxes drop the incoming requests. Hole Punching is a technique that allows a host located behind a firewall/NAT to send traffic to another host without collaboration of the NAT itself [15], [16]. With the help of the well-known Rendezvous server (RS), clients can establish these direct sessions. First hosts establish initial UDP session with the RS and the server later exchanges the connection details with both the hosts. Since each device knows the other peer's details they can establish a connection by sending an initial request. Figure 6 illustrates the hole punching process.

Transmission Control Protocol (TCP) is connection oriented and reliable protocol mainly used to transport data in a reliable manner. The size of the TCP header is 20 Bytes. Some important fields of the TCP header which are interesting for the hole punching and the rest of the discussion of the paper are described below.

Source port: The source port is the number that is assigned by the local host when it transmits data to the remote host. This is a random number which is normally higher than 1023 according to the Internet Assigned Numbers Authority (IANA) port numbering procedure. Source device can distinguish the sessions by referring the source port.

Destination port: The destination port is normally a well-known port number which is used by the remote device to identify the service requested by the source device.

Sequence number: It is a 32 bit number, which indicates the reference number of the transmission data byte. This number helps to maintain the ordered delivery of data byres.

Acknowledgment number: This is a 32 bit number which is used to maintain the reliability of the data transmission. The receiver should acknowledge the transmitter about the received data.

TCP is a connection oriented protocol. That means, first the transmitter and the receiver should establish the logical connection before the data transmission. Then data can be sent through this connection and the connection should properly terminate at the end. The connection establishment process of

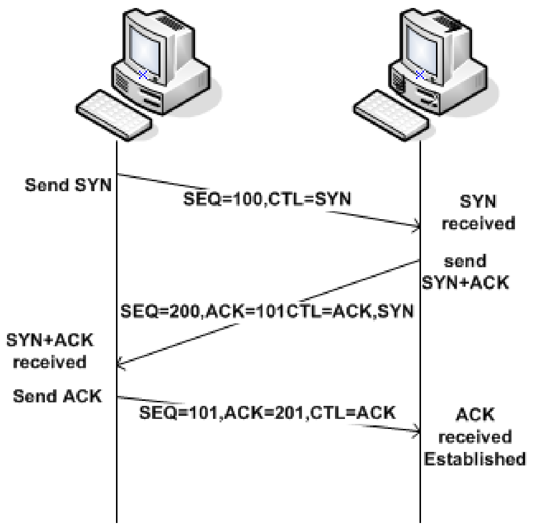


Fig. 7. TCP 3-way handshaking

the TCP is called as the Three-Way Handshaking and is shown in figure 7.

III. Proposed Solution

There are several solutions [16], [17], [18] already implemented for the TCP hole punching. But most of these solutions target establishing sessions between peers connected to the fixed networks. When considering a fixed network, most of the times the NAT device is also owned by the same organization and it can be configured (probably by admin) based on need so that some inconvenient NAT configurations can be avoided. But when using a mobile network, the user is unable to modify any NAT policy and must use whatever the NAT configurations are deployed by the mobile operator. In addition, when considering fixed network, number of the connected devices are generally limited. But in mobile networks every day there are thousands of devices at any particular time that are connected and using it. After an extensive search we could not find any workable solution which can be used for the Android devices in mobile networks.

To address these problems we propose to use UDP sessions with the Rendezvous server to send and receive peer information and TCP sessions in the hole punching process.

A. The Rendezvous Server

To demonstrate the proof of concept, the Rendezvous server is developed and deployed on an Ubuntu server installed on Amazon public cloud as an Elastic Compute Cloud (EC2) micro instance. Micro instances are minimum capacity servers offered by Amazon EC2 with 615 MB memory. However, when the services become popular, the rendezvous server can be deployed on a much powerful machine, exploiting the different types of servers offered by public clouds and their vertical scalability. The server is also assigned a public IP address (e.g. 107.20.232.29). Since the server is assigned with a public IP address, any host can easily access the server even though they are located behind the NAT. For the registration process we used UDP datagrams. Since UDP is a connectionless, we can reduce the overhead of maintaining

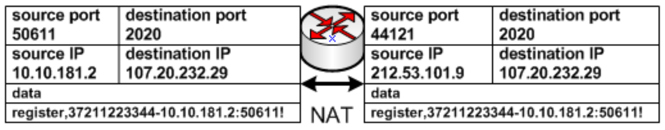


Fig. 8. Details of the datagram in registration

too many sessions for registration requests. When the server is running there is a UDP socket listening on local port 2020 to accept client's registration requests.

The first step in the hole punching procedure is that the client devices need to register with the server. We have developed an Android application to help the user in performing the procedure. The client should first send the UDP packet to the server with the client's registration details. As per the proposed solution, clients should register with the server by providing their mobile number in the IDD format in order to maintain a unique identification for the clients. The server stores clients' details (Mobile no, Public IP Address assigned at NAT, Public Port, Local IP Address and Local Port) in its internal data structure for future matching. When the client device is registered for the first time, the server program creates a client object and adds it into the client list. If the same client registers next time with the same number, the existing client object's details are updated with the current details.

B. Establishing peer connections

The proposed solution is capable to address mobile devices in different situations. As an example sometimes both peer devices can be located in the same cell under one mobile operator or sometimes the peers are in two different locations under two mobile operators. The server will identify clients' configurations and handles them accordingly. The possible scenarios and solutions are explained here in detail.

1) Both peers are located in the same 3G cell - same Mobile Operator: In this situation most of the times both host devices are getting their IP configurations from the local interface of the same GGSN. So this GGSN device is working as the NAT interface for both hosts. In other words both clients are behind the common NAT device and belong to the same private network. In this case, the connection will be established as follows.

When the host A needs to establish a connection with the host B to send some information, the first step is that both devices should register with the rendezvous server. For this they should start the installed client application and send the UDP request as "register" to the server endpoint 107.20.232.29:2020, as already explained, and shown in figure 8.

If the registration is successful, the server creates an object with A's registration details and adds it to the clients' list for later matching. If the recipient (host B) is not registered, host A can send the message to the host B by using existing communication methods (SMS, SIP message) for asking to register with the server. Extensions can also be introduced to the current approach, when it is widely adopted, such as the mobile networks themselves can register the devices

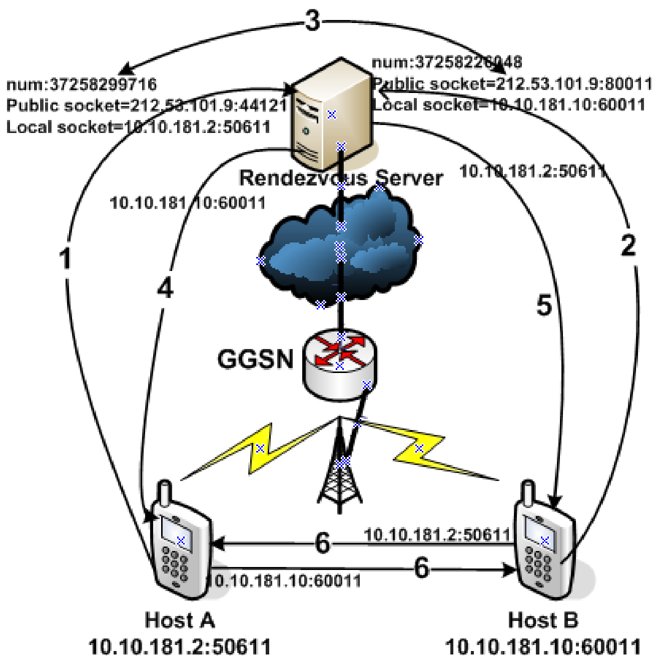


Fig. 9. Establishing connection through hole punching when peers are behind the same NAT

to the public rendezvous servers once they show interest in exchanging packet data.

Registration details of the host A are as shown in table I. Once host B gets the invitation from the host A, it also registers with the server by sending UDP ”register” request. B’s registrations details are also mentioned in table I.

	Host A	Host B
Mobile No	37211223344	37211223355
Local IP	10.10.181.2	10.10.181.10
Local Port	50611	60011
Public IP(at NAT box)	212.53.101.9	212.53.101.9
Public Port(at NAT box)	44121	80011

TABLE I. Registration details of hosts A and B

Now host A can send the request as ”connect” with the host B’s mobile number as the destination. At this point server processes this request and stores the information in a temporary object. Then the server gets the destination number of this temporary client object (37211223344) and looks for a matching client in the client list. Since host B is already registered in the client list the server can find the connection details of the B. Steps 1 and 2 in figure 9 represent the registration process.

To identify the scenario whether both hosts are behind the same NAT, the server does further matching with public IP addresses of both hosts A and B. If the both public IP addresses of A and B are the same then they should be behind the same NAT box. In this situation hosts A and B only need to know their local endpoint details to initiate a connection.

Then the server crafts a Datagram and inserts host A’s local endpoint details (10.10.181.2: 50611) and sends it to the host B. Also server crafts another Datagram and inserts host B’s local socket details (10.10.181.10: 60011) and sends it to the host A. The procedure is illustrated in figure 9, steps 3, 4 and 5.

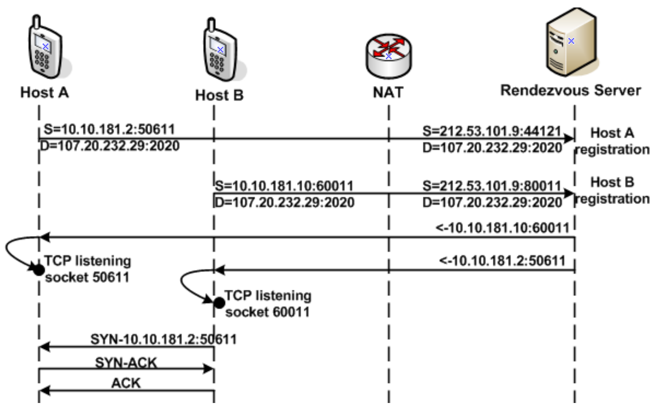


Fig. 10. Sequence diagram showing the timeline of activities performed during the connection establishment between the devices

Once the datagrams are received, both host A and B should create a TCP listening socket according to their local port number which was sent to the other host. As an example host A and B should create listening TCP sockets with the ports 50611 and 60011, respectively. Host A then sends TCP SYN message to initiates the session. At the host B there is a TCP listening socket running on port 60011 to accept TCP SYN message from the host A. Then host B sends a SYN-ACK to Host A. Host A finishes the establishment of TCP connection by confirming with ACK. The complete scenario is shown in sequence diagram figure 10. Notice that, since the TCP listening sockets are running at the both host devices any device can initiate the connection procedure, after the registration.

2) Peers are located behind different NAT boxes: When the peers are in different mobile operators they are located behind different NAT boxes and hence are in different private networks. In order to initiate a connection, both clients should learn the public socket details about each other.

The registration phase is almost the same as the previous scenario and table II shows the registration details of hosts A and B.

	Host A	Host B
Mobile No	37211223344	37211223355
Local IP	10.10.181.2	10.100.100.2
Local Port	50611	60011
Public IP(at NAT box)	212.53.101.9	89.18.102.10
Public Port(at NAT box)	44121	80011

TABLE II. Registration details of hosts A and B

The server uses the same algorithm to match the destination device as described earlier. But in this case peers public IP addresses are not same because they are coming from different NAT boxes. Because of this reason, the server should exchange hosts’ public socket as well as private socket details between peers to initiate the sessions. At this point, server crafts a datagram and inserts host B’s public and private socket details and sends it to the host A. Similarly, server sends host A’s public and private socket details to host B by using another datagram (figure 11, steps 3, 4 and 5).

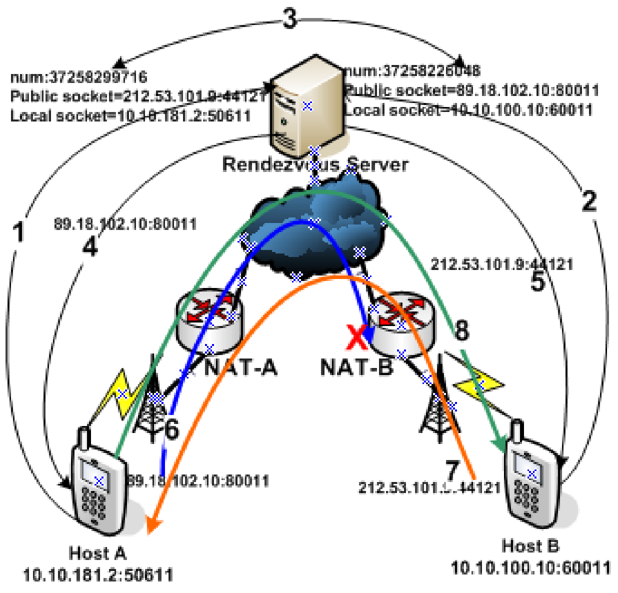


Fig. 11. Establishing connection through hole punching when peers are behind different NATs

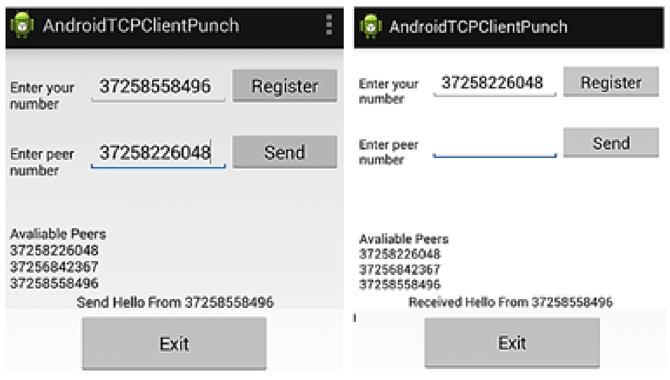
Once the datagrams are received, the next step is starting a TCP session between the peers. Here both peers initiate sessions simultaneously. Host A starts the TCP listening socket on port 50611 and sends the TCP-SYN message to the host B. At the NAT-A it creates a map for this session and forwards it to the host B's public end point, here it is 89.18.102.10:80011. But at the NAT-B this packet is dropped because it is unknown inbound packet for the NAT-B. But still NAT-A is waiting for the reply from B (hole is punched).

Same time host B also starts a TCP listen socket on the port 60011 and sends the TCP-SYN packet to the host A's public socket, here it is 212.53.101.9:44121. NAT-B creates a map and forwards this packet to the A. Now this packet can reach the host A because NAT-A is waiting for a reply from the B. In other words, this packet can travel through the punched hole in NAT-A and reaches the host A. Host A accepts this TCP-SYN message as there is a TCP listen socket on port 50611 to accept SYN messages. Then host A sends SYN-ACK message to the B and NAT-B allows this packet because there is a hole punched during initial step. Now B accepts SYN-ACK and confirms to A using ACK message. Once the ACK received by the host A then both devices have established TCP session which can be used to exchange data between them (2 way communication).

It is interesting to note that, in the above scenario, host A wanted to initiate the session, however if one thinks from a pure TCP perspective, the scenario ended up as though host B initiated the TCP session

IV. Test Scenarios and the Results

The proposed solution is implemented and tested with few Estonian mobile networks, TELE 2 [19], Elisa [20], EMT [21]. The considered test scenarios and the results are described below.



37258558496

37258226048

Avaliable Peers
37258226048
37256842367
37258558496

Send Hello From 37258558496

37258226048

Avaliable Peers
37258226048
37256842367
37258558496

Received Hello From 37258558496

Fig. 12. Android application with the messages

A. Test Scenario 1 - Both hosts are located in same mobile network

The scenario is demonstrated without any troubles in TELE2 network. Here both mobile devices use TELE2 3G internet connection. Host A is a Samsung Galaxy SIII phone with the mobile number 37258299716. Host B is a Nexus-5 phone with the number 37258226048.

Both hosts A and B are registered with the server. After that host A again sends "connection" request to the server to establish a connection with host B. The server checks the hosts' registration number to locate the destination. The details are later exchanged, as already explained. After this step, both devices were able to send and receive messages from each other. Figure 12 displays the output captured from two hosts when the host A sent the message and the host received the message on our developed client application. As already explained once the connection is established, host B can also send a message.

B. Test Scenario 2 - Hosts are located in the different mobile operators

In this scenario the mobiles are connected to deferent mobile networks and tried to exchange messages. The results of this analysis are mixed and the reasons are discussed in detail in the coming sections. Consider the case where host A was in the TELE2 3G/4G mobile network and the host B was in the EMT 3G/4G mobile network. As the first step both devices register with the server. Now host A sends a message to the host B. Server locates the host B (3725855496) and exchanges connection details. Then the devices tried to exchange messages.

Here we observed interesting results. When sending messages from host A (A initiates connection) to host B our proposed solution failed. However, when sending messages from host B (B initiates connection) to host A it was successful. Similar results we observed when host A is in TELE2 and host B in Elisa.

Table III summarizes the results of different test scenarios with the three common Estonian mobile networks. So we observed, when one of the devices is in the TELE2 mobile network, connection can be established form any other mobile network, by making the second device to initiate connection.

It should be remembered that once connection is established, both the devices can send the messages to each other as long as the connection persists (TCP connection is bidirectional).

Connection initiator	Receiving mobile	Result
TELE2	EMT	Fail
	Elisa	Fail
EMT	TELE2	Success
	Elisa	Fail
Elisa	TELE2	Success
	EMT	Fail

TABLE III. TEST RESULTS OF THE HOLEPUNCHING APPROACH ACROSS DIFFERENT MOBILE NETWORKS

V. DISCUSSION OF THE RESULTS AND LIMITATIONS OF THE APPROACH

According to the test results we observed that some hosts cannot establish connections properly. There are few internal and external factors that will limit the functionality of our implementation. Main external factor is the differences in behavior of the NAT techniques used by the mobile operators. To get a better idea about the behavior of the NAT box applied, we implemented a simple application which can retrieve devices' own public socket details. This application sends two messages using same local socket to two different servers. These servers replied to the host with the public socket details assigned by the respective NAT for that communication. Table IV summarizes the result of the NAT translation of three common mobile operators in Estonia.

Mobile network	Local socket	Public socket (from server1)	Public socket (from server2)
TELE2 (subscription 1)	83.176.3.195 :35049	83.176.3.195 :35049	83.176.3.195 :35049
TELE2 (subscription 2)	83.176.2.178 :40756	83.176.2.178 :40756	83.176.2.178 :40756
EMT (subscription 1)	10.145.20.13 :52142	217.71.46.13 :52142	217.71.46.13 :52142
EMT (subscription 2)	10.128.194.59 :58296	217.71.44.59 :58296	217.71.44.59 :58296
Elisa (subscription 1)	10.27.201.251 :51605	194.150.65.155 :55251	194.150.65.155 :55252
Elisa (subscription 2)	10.26.164.121 :33045	194.150.65.24 :32322	194.150.65.24 :32323

TABLE IV. NAT ACROSS DIFFERENT MOBILE NETWORKS IN ESTONIA

Our approach is basically working properly on the TELE2 mobile network. When we analyze the IP address assignment here we can see that TELE2 always assigns a public IP address (observed actually a huge pool of them) to a mobile device. Since, probably there is no NAT translation applied in TELE2, devices from any mobile network (even when having NAT applied within) can directly initiate a session with a device that is attached to the TELE2 mobile network.

When considering EMT mobile, it assigns a private IP address to the clients and translates it into a public IP address when needed. During NAT it keeps the local port number as the same and only translates the private IP address into the public IP address. Our implementation currently supports only full-cone type NAT and EMT might be using a technique other than full-cone NAT and connection can't be initiated to the EMT network.

Address translation of the ELISA mobile network is more likely similar to the symmetric NAT translation. We found that ELISA translates a local socket into a completely new public socket. Even the request coming from same local socket to the different destination, NAT assigns a new socket for that session. It is very difficult to implement hole-punching because of the port prediction which will be involved in this case.

When implementing our solution we only considered about how the NAT maps local socket and the public socket. But in TCP session, sequence numbers also play a major role. Some NAT boxes are also checking the sequence numbers of the incoming TCP packets. If this sequence number is invalid, NAT just closes the public socket. To overcome this problem we should also learn the sequence numbers of end points TCP process. But to capture the sequence numbers we need the help of the raw sockets. Creating a raw socket is not difficult for a PC but is very difficult for android devices, which most often needs rooting the device. We have a solution developed based on this approach, but we think this approach is too invasive for the mobile device or IoT [22]. However, one can consider the case as one of the limitations of our implemented solution rather than the proposed solution.

VI. RELATED WORK

Establishing a TCP session through NAT has been studied extensively. There are few prominent techniques which are summarized here.

NUTSS is an approach which proposed using an external STUNT server and simultaneous NAT traversal technique that is applicable to devices connected to the fixed network [23]. According to this approach both end points send an initial SYN with a little TTL value that is just enough to cross their own NATs. This SYN is dropped in the middle of the network before reaching to its destination. End points can learn their initial sequence number by listening to their outbound SYN using PCAP or a RAW socket. Then both endpoints inform this sequence number to the STUN server hosted on the public network. The STUN server spoofs a SYNACK to each host with the properly settled sequence numbers. After that both hosts send ACK to complete the handshaking process. There are some potential problems which can be seen when implementing this approach. We need to set the lower TTL value in the SYN packet that should be more enough to cross the own NAT but less enough to cross the destination NAT. But it is impossible to set that kind of the TTL value if both devices are located behind the same NAT. To set the TTL value we need to access the OS TCP process using a raw socket. But when considering android devices creating a raw socket is still complicated and needs a rooted device.

Similarly, in NATBLASTER approach both devices use SYN packet with lower TTL value to initiate the session [18]. This SYN packet is dropped in the middle of the network before reaching its destination. Then hosts learn the initial sequence numbers of the SYN packet and send it to the publicly accessible third party server. This server exchanges initial sequence number with the hosts. Instead of using server to forge SYNACK packets here each host forges SYNACK to the other host. After this step each host can establish the TCP session by sending ACK to the peer. Again, The approach ends up with the problem of setting a TTL value and making raw socket which needs root privileges.

Nattrav is a software approach to solve the NAT problems for P2P applications [16]. First peers should register with a connection broker which provides current network address for the recipients and facilitate the NAT traversal if peers are behind the NAT. In the registration, peers register with the connection broker by providing the peer's URI. Connection broker uses this URI to send subsequent connection requests from other peers. When a peer device needs to connect with other peer device it should send the lookup message to the connection broker. The connection broker eventually replies back with the IP address and the port number of the destination peer. Since peers know the other side endpoint details they can establish a connection by sending a SYN packet. If the peers are behind the NAT connection, broker helps the traversal by providing a public endpoint details of the destination peer. In the implementation they used a Java package called "nattrav" which provides classes for setting up TCP sockets. This solution is conceptually the most similar approach to the one proposed by us. However, our approach is specifically targeted for mobile devices, which makes it different from any proposed solutions.

VII. Conclusions and Future Work

In this paper, we analyzed addressing a mobile device in the mobile 3G network (also applicable for 4G) using TCP hole punching approach. We introduced the approach and developed a prototype solution as an android application that helps in making the TCP hole punching. The testing results showed that the connections can be established following the approach in different networks. However, there are some situations where the approach is unusable, in some networks. The main reason is that there is no standard for the NAT process. Different mobile operators use different NAT process and it is difficult to implement all in one solution. In this context, the limitations of the approach are discussed in detail.

To overcome these limitations we propose to implement a TCP server that can relay the messages in between hosts. Then the host should first establish a connection with the server. When the host needs to send a message to another host that message is forwarded to the relay server and the server forwards it to the destination host. However, this approach is applicable only when we consider sending one off messages. In scenarios explained by the paper like mobiles providing services and establishing the mobile Internet of things the solution may not be suitable.

Acknowledgment

This work is supported by European Regional Development Fund through EXCS, Estonian Science Foundation grant PUT360 and Target Funding theme SF0180008s12.

References

[1] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile web service provisioning," in *Int. Conf. on Internet and Web Applications and Services (ICIW'06)*. IEEE, 2006, pp. 120–125.

[2] S. N. Srirama and C. Paniagua, "Mobile web service provisioning and discovery in android days," in *Proceedings of the 2013 IEEE Second International Conference on Mobile Services*. IEEE Computer Society, 2013, pp. 15–22.

[3] D. Recordon and D. Reed, "Openid 2.0: a platform for user-centric identity management," in *Proceedings of the second ACM workshop on Digital identity management*. ACM, 2006, pp. 11–16.

[4] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, E. Schooler *et al.*, "Sip: session initiation protocol," RFC 3261, Internet Engineering Task Force, Tech. Rep., 2002.

[5] P. Saint-Andre, "Extensible messaging and presence protocol (xmpp): Core," 2011.

[6] D. Braun, S. Mukherjee, and C. Akinlar, "Zero configuration networking," Feb. 21 2006, uS Patent 7,002,924.

[7] U. Plug and P. Forum, "Universal plug and play forum." [Online]. Available: http://www.upnp.org/forum/default.htm

[8] S. N. Srirama, M. Jarke, and W. Prinz, "Mobile host: A feasibility analysis of mobile web service provisioning," in *4th International Workshop on Ubiquitous Mobile Information and Collaboration Systems, UMICS*. Citeseer, 2006, pp. 942–953.

[9] S. E. Deering, "Internet protocol, version 6 (ipv6) specification," 1998.

[10] G. J. d. Groot, Y. Rekhter, D. Karrenberg, and E. Lear, "Address allocation for private internets," 1996.

[11] I. global standard for international mobile telecommunications. [Online]. Available: www.imt-2000.org

[12] K. Richardson, "Umts overview," *Electronics & Communication Engineering Journal*, vol. 12, no. 3, pp. 93–100, 2000.

[13] P. Srisuresh and M. Holdrege, "Ip network address translator (nat) terminology and considerations," 1999.

[14] B. Sterman and D. Schwartz, "Nat traversal in sip," *IEC Annual Review of Communications*, vol. 56, 2002.

[15] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-peer communication across network address translators." in *USENIX Annual Technical Conference, General Track*, 2005, pp. 179–192.

[16] J. L. Eppinger, "Tcp connections for p2p apps: A software approach to solving the nat problem," *Institute for Software Research*, p. 16, 2005.

[17] S. Guha and P. Francis, "Characterization and measurement of tcp traversal through nats and firewalls," in *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*. USENIX Association, 2005, pp. 18–18.

[18] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig, "Natblaster: Establishing tcp connections between hosts behind nats," in *ACM SIGCOMM Asia Workshop*, vol. 5, 2005.

[19] Tele2. [Online]. Available: http://www.tele2.ee/index.html

[20] Elisa. [Online]. Available: https://www.elisa.ee/

[21] EMT. [Online]. Available: https://www.emt.ee/en/

[22] K. Reinloo, "Addressing smartphones located behind firewalls," 2013. [Online]. Available: http://dspace.utlib.ee/dspace/handle/10062/32957?show=full

[23] S. Guha, Y. Takeda, and P. Francis, "Nutss: A sip-based approach to udp and tcp network connectivity," in *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*. ACM, 2004, pp. 43–48.