

UDPTUN – Direct TCP Connection Between 'NAT behind' Hosts

Béla Almási

Faculty of Informatics, University of Debrecen, Hungary

Abstract

NAT boxes connect private LANs to the public Internetwork. Hosts located in the private network are able to connect to the servers located in the public world, but they are unable to establish a connection to a host located in a different private network. Special configuration of the NAT boxes (e.g. configuring 'port forwarding') can help to reach private hosts behind the NAT boxes. There are cases, where direct TCP connection between hosts (located in two different private networks) is necessary without NAT box configuration.

In this paper we would like to introduce a software based solution for the mentioned situation: The solution (named UDPTUN) establishes a direct tunnel connection between two hosts located in different private networks (without changing or touching the configuration of the NAT boxes). We would like to discuss the theoretical background of UDPTUN, and an implementation example will be given to illustrate the usage of the system.

Keywords: Address translation, TCP, NAT, tunnel, p2p connection

1. Introduction

Investigating communication networks at different layers is a very popular research area today (see e.g. [1], [2]). The topic of this paper is the IP Network Layer communication technology. Network Address Translation (NAT, or sometimes referred as Network and Port Address Translation) is a widely used technology to spare the globally used IP numbers. The basic idea of NAT was introduced in [3]: The IP address reusing idea allows organizations using the same private address range inside the organization, so giving a short-term solution for the IP address depletion problem. The private address ranges 192.168.0.0/16, 172.16.0.0/12, and 10.0.0.0/8 can be used for addressing the hosts inside an organization without registration (see [4]).

A host using a private IP address (called as 'NAT behind host') should be able to communicate with servers located in the global internetwork. A so called NAT

box must be installed at the border of the private and global address realms. The NAT box changes the private addresses into global ones when the packets traverse through it. To spare the global IP addresses, the NAT box may change not only the IP addresses, but the port numbers too (overloaded NAT).

Basically, the NAT behind hosts can not be reached with a connection establishment request from the global address realm: Concerning the situation represented on Figure 1., both 'Host A' and 'Host B' is able to establish connection to servers located in the Global Internet, but it is not possible in the opposite direction, and so 'Host A' and 'Host B' can not communicate directly to each other. In some

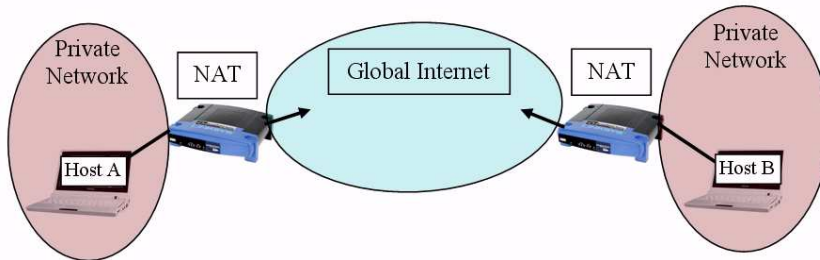


Figure 1: NAT behind hosts.

cases it would be necessary for NAT behind hosts to establish a direct TCP communication session to each other. In this paper we investigate this problem. We will give a short review on a widely known existing solution (port forwarding), and we introduce a new solution method (udptun). Also, the software tool that implements udptun will be discussed.

2. Port forwarding

The problem mentioned in Section 1. is not new. The NAT box producers usually build the so called 'port forwarding' feature into the equipment, which is able to handle the introduced problem. The different vendors use different names referring this technology; e.g. port forwarding, static NAT, static PAT (Port Address Translation), virtual server are different names of the same solution idea.

In this solution a static NAT entry is created by the system administrator of the NAT box. The static NAT table entry ensures that the incoming packets will be driven to a given host in the private address realm. The static entry contains four important data fields: the global IP number (if the NAT box has got more IP addresses, it identifies the actually used one), the global TCP port number, the private IP number (it identifies the host in the private address realm), and the private port number (which identifies the application on the private host).

A port forwarding example is shown on Figure 2. The working mechanism is the following: The administrator of the NAT box create a static NAT entry (see

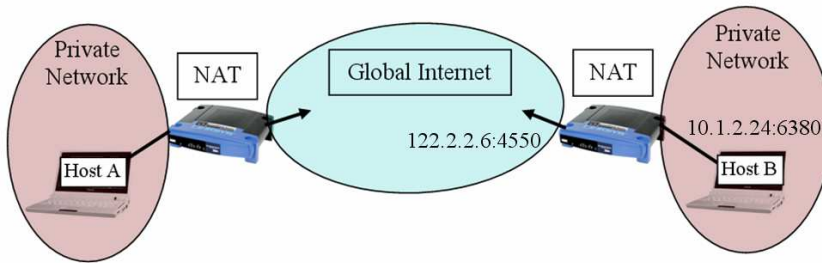


Figure 2: Port forwarding.

Table 1). When a packet arrives to the NAT box from the internet addressed to 122.2.2.6 with destination TCP port number 4550, then the packet will be sent to the private host 10.1.2.4 with destination TCP port number 6380. On the other hand, the dynamic NAT entries are applied as usually, i.e. all of the private hosts are able to communicate to the servers located in the global address realms.

Global IP	Global TCP Port	Private IP	Private TCP Port
122.2.2.6	4550	10.1.2.24	6380

Table 1: Static NAT table entry.

The mentioned static NAT entry also opens the communication way for 'Host A' to reach 'Host B' (where the two hosts are located in different private address realms): Host A is able to open a communication session to any host in the global internet. It means, that Host A can reach the IP address of 122.2.2.6 with the destination TCP port of 4550. When this packet arrives to the NAT box (shown at the right side of the picture), then the NAT box will use the static NAT entry of Table 1 to directed this packet to 'Host B'. Port forwarding gives a good solution in many cases, but we may not forget, that it requires special configuration of the NAT box. There are situations, where the NAT box configuration can not be performed (e.g. we do not have administration right over the Nat box). The following section gives a solution example for these cases.

3. UDPTUN

3.1. UDP NAT Traversal

In [5] we can read a solution to the NAT traversal problem using UDP. The theoretical idea of STUN is the following: Both of the NAT behind hosts get the peer's global IP number and port number. Both hosts start flooding UDP packets to the peer's global address. These packets will create a dynamic NAT box entry,

when they leave the sender's local address realm. When the peer's packet arrives, it will be directed to the right private address, according to the NAT box entry created before. Getting the peer's global address and port number is an interesting and sometimes difficult question (see [6]).

In this paper we do not work on the global address determining problem, we assume, that we could finish it, and the traversal of UDP packet works.

3.2. The theory of UDPTUN

The basic idea of the UDPTUN solution is the following: We create a NAT traversal UDP connection between the hosts (assuming, that we could get the global address of the peer). We create a tunnel interface on the NAT behind hosts. The tunnelling mechanism will be set up according to the followings: When an application sends an IP packet to the tunnel interface, then the whole IP packet will be encapsulated into an UDP segment according to the created NAT traversal connection, and will be sent to the peer. When a UDP packet arrives on the NAT traversal connection, then the data part of the UDP segment (i.e. the IP packet which was encapsulated on the sender's side) will be sent to the tunnel interface as an incoming packet. The assumption of the UDPTUN solution is, that the UDP NAT traversal works and is active at both sides. The UDP NAT traversal connection use dynamic NAT box entries, which are usually deleted from the NAT table, if no packet transfer occurs for a longer time. In order to keep the UDP NAT traversal connection active (i.e. not to lose the dynamic NAT entry), we must send "empty" UDP packets to the peer regularly when there is no application activity.

3.3. The UDPTUN application

We implemented the UDPTUN theory in Linux and in Microsoft Windows XP. The Linux kernel 2.6 already contained the tunnel interface implementation, so Linux supports the tunnel interface natively. In Microsoft Windows XP (and also in Windows 7) basically there is no tunnel interface support. Fortunately, a third party tunnel interface implementation can be found in the OpenVPN software package (see [7]). The tunnel interface can be installed alone, without the full installation of the OpenVPN package, or some older version of the WinTAP32 tunnel interface can be found independently of the OpenVPN package too.

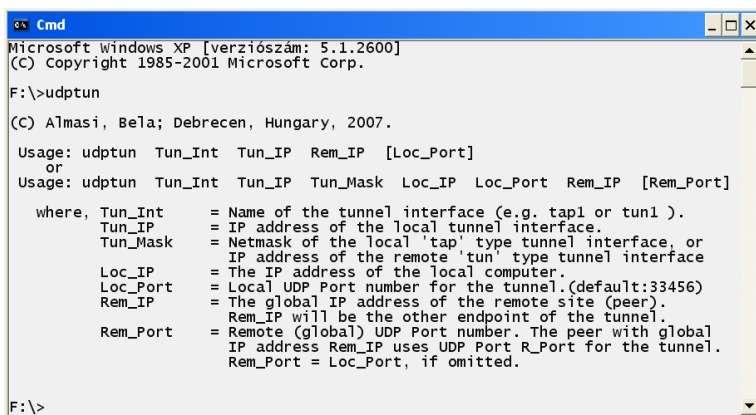
In both operating systems there are two types of tunnel interfaces: the 'tap' type and the 'tun' type. The 'tap' type tunnel interface represent an Ethernet-like interface, while the 'tun' type tunnel interface represent a Point-to-Point like interface (see e.g. in [8]). UDPTUN can be used with both types, but the same type must be used at the tunnel endpoints in order to get a successful connection.

The statically linked (Linux and Windows XP) binary executables of the udptun software can be downloaded from <http://irh.inf.unideb.hu/user/almasi/udptun>. The Windows XP executable can be used in Windows 7 too, but we must have full

administrator right to use it (root/administrator right is necessary in Linux/XP, too).

3.4. The usage of UDPTUN

Starting the `udptun` executable program without parameters will show the usage syntax possibilities (see Figure 3.). The short syntax version of `udptun` contains only the name of the tunnel interface (e.g. `tap1` or `tun1`), which specifies the type of the interface too (i.e. the tap type interface must have the name 'tapX', and the tun type interface must have the name 'tunX' (X=0,1,...,9)). The second parameter is the IP address of the tunnel interface. `Udptun` will configure the tunnel interface with this IP address and netmask of 255.255.255.0. The third parameter is the global IP address of the peer, i.e. the outgoing UDP packets will be sent to this destination address. Using the short syntax form we assume, that all the UDP port numbers are the same. The default value of this common UDP port number is 33456, or it can be specified as the 4th command-line parameter.



```
Microsoft Windows XP [verziószám: 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

F:\>udptun

(C) Almasi, Bela; Debrecen, Hungary, 2007.

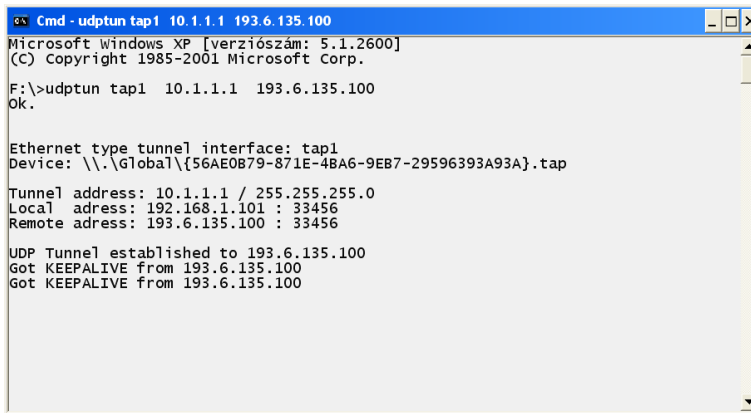
Usage: udptun Tun_Int Tun_IP Rem_IP [Loc_Port]
or
Usage: udptun Tun_Int Tun_IP Tun_Mask Loc_IP Loc_Port Rem_IP [Rem_Port]

where, Tun_Int      = Name of the tunnel interface (e.g. tap1 or tun1 ).
      Tun_IP        = IP address of the local tunnel interface.
      Tun_Mask      = Netmask of the local 'tap' type tunnel interface, or
                    = IP address of the remote 'tun' type tunnel interface
      Loc_IP        = The IP address of the local computer.
      Loc_Port      = Local UDP Port number for the tunnel.(default:33456)
      Rem_IP        = The global IP address of the remote site (peer).
                    = Rem_IP will be the other endpoint of the tunnel.
      Rem_Port      = Remote (global) UDP Port number. The peer with global
                    = IP address Rem_IP uses UDP Port R_Port for the tunnel.
                    = Rem_Port = Loc_Port, if omitted.
```

Figure 3: Udptun usage syntax.

The long syntax version of `udptun` allows to specify all the data related to the tunnel establishment (see Figure 3).

After issuing the `udptun` command with the right parameters it configures the tunnel interface and begin to send UDP packets to the peer's global address (so creating the dynamic NAT table entry in the local NAT box). When the peer's UDP packet arrives, the tunnel is established, and the applications can use the tunnel interface for direct TCP connection to the peer. If there is no application activity, then an empty 'keepalive' UDP packet is sent to the peer in every 10 seconds, so ensuring, that the dynamic NAT entries will not be deleted (see Figure 4).



```

Microsoft Windows XP [verziószám: 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

F:\>udptun tap1 10.1.1.1 193.6.135.100
Ok.

Ethernet type tunnel interface: tap1
Device: \\.\Global\{56AE0B79-871E-4BA6-9EB7-29596393A93A}.tap

Tunnel address: 10.1.1.1 / 255.255.255.0
Local address: 192.168.1.101 : 33456
Remote address: 193.6.135.100 : 33456

UDP Tunnel established to 193.6.135.100
Got KEEPALIVE from 193.6.135.100
Got KEEPALIVE from 193.6.135.100

```

Figure 4: Udptun in work.

4. Summary

In this paper we considered the problem of direct TCP connection between two NAT behind hosts. Although static NAT table entries can solve the problem, there are cases, when direct TCP connection between hosts (located in two different private networks) is necessary without NAT box configuration.

We introduced a new software based solution UDPTUN. It establishes a direct tunnel connection between two hosts located in different private networks, based on UDP NAT traversal. The binary executables of UDPTUN can be downloaded from <http://irh.inf.unideb.hu/user/almasi/udptun>.

References

- [1] M. ARATÓ, A. KUKI, ZS. LENCSE, *Experimental news distribution system*, Proceedings of the 4th International Conference on Applied Informatics, pp 31-34, Eger, Hungary (1999).
- [2] P. OROSZ, J. SZTRIK, C. KIM, *Dynamics and Congestion Control of Alternative TCP Variants on Asymmetric Lines*, ISAST Transactions on Communications and Networking, No 1. Vol. 2., pp 71-74, (2008).
- [3] K. EGEVANG, P. FRANCIS, *The IP Network Address Translator*, RFC 1631, (1994).
- [4] Y. REKHTER, B. MOSKOWITZ, D. KARREBERG, G. J. DE GROOT, LEAR, *Address Allocation for Private Internets*, RFC 1918, (1996).
- [5] J. WEINBERGER, C. HUITEMA, R. MAHY, *STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs)*, RFC 3489, (2003).
- [6] J. ROSENBERG, R. MAHY, P. MATTHEWS, D. WING, *Session Traversal Utilities for NAT (STUN)*, RFC 5389, (2008).

- [7] OPENVPN TECHNOLOGIES INC., *The OpenVPN Project*: <http://openvpn.net/index.php/open-source.html>, checked on 06.06.2010.
- [8] <http://en.wikipedia.org/wiki/TUN/TAP> checked on 06.06.2010.

Béla Almási

H4010 Debrecen, Egyetem tér 1. P.O.Box 12. Hungary.

e-mail: almasi.bela@inf.unideb.hu