University of Helsinki
Department of Computer Science
Series of Publications C, No. C-2012-4

Study of Middle-box Behavior on Transport Layer Protocols

Seppo Hätönen, Yonghao Li, Markku Kojo

Helsinki, June 13, 2012

Technical Report C-2012-4

University of Helsinki
Department of Computer Science
P. O. Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, FINLAND

Study of Middle-box Behavior on Transport Layer Protocols

Seppo Hätönen, Yonghao Li, Markku Kojo
Department of Computer Science, University of Helsinki
Technical Report C-2012-4
June 13, 2012
18 pages

Abstract. Today, the home gateways that act as middle-boxes between the internal network of a residential user or a Small Office/Home Office (SOHO) and the public network or Internet are very common. These devices include wireless access points and cable and DSL modems. The devices perform various higher-layer functions such as traffic filtering, network address translation (NAT) and can act as a dynamic host configuration protocol (DHCP) server. While some of these functions such as DHCP are well standardized, some functions such as NAT have only been defined on a more abstract level and the exact operation have not been standardized. These more loosely defined functions are known to have undesired effects on normal protocol operation and hinder the development of new protocols and applications. Therefore, it is important to understand what different middle-boxes that have been deployed all around the world actually do and what are their characteristics. This experimental study concentrates on the transport layer functionality of home gateways. The transport layer is responsible for the transport services such as provided by Transmission Control Protocol (TCP) and User Datagram Protocol (UDP.) While some of the protocols are widely used and well-known, some newer protocols and extensions to the earlier protocols might have difficulties with the middle-boxes. The NAT function is particularly problematic since many newer protocols and extensions might not have been deployed when some of the middle-boxes were built and deployed. Due to this, the middle-boxes may not be able to properly forward packets carrying these protocols and applications. We test new transport protocols such as Lightweight User Datagram Protocol (UDP-Lite) and Stream Control Transmission Protocol (SCTP), extensions to regular operation such as inserting payload to the initial TCP SYN packet, and the UDP throughput performance of the middle-boxes.

Contents

1 Introduction

Nowadays, home gateways such as wireless access points and Cable or DSL modems, are widely deployed for residential and Small Office/Home Office (SOHO) customers to access Internet services. The home gateways typically act as middle-boxes performing various higher-layer functions, such as network address translation (NAT) [11], traffic filtering or advanced application layer operations.

The NAT was first proposed by the Internet Engineering Task Force in 1994 to help with the foreseen IPv4 address shortage before IPv6 was was designed. Since then, while the last available IPv4 address pool was assigned by Internet Assigned Numbers Authority (IANA) in February 2011, IPv6 is still not widely deployed, at least not in the small companies and residential networks. Only a small percent of Internet Service Providers (ISP) offer IPv6 routing to normal home users and the ISPs see the NAT properties as a kind of firewall which masks the internal side of the NAT device from the Internet or public side of the NAT, the ISPs are not in a hurry to move forward with the change.

Unfortunately, the IETF only defined the basic properties of the NAT and left the implementation open. This has led to many different NAT implementations over the years and many of them cause considerable problems with different Internet protocols. The main goal of this study is to dig deeper into the characteristics of different home gateway devices that implement NAT functionality. We develop a number of tests to explore how the current NAT devices behave with the IP traffic that traverses through the devices.

The experiments extend the earlier study [4] that focused more on the NAT binding timeouts, Transmission Control Protocol (TCP) [9] throughput etc., while in this study we focus on investigating more specific characteristics of the home gateway devices in the presence of various TCP options and header fields. We also extend some of the earlier experiments with TCP to also cover User Datagram Protocol (UDP) [8] and test different NAT characteristics such as mapping and filtering behavior and try to classify the devices with taxonomy provided by RFC 5789 [7]. The TCP tests try to provide information on how the NATs might hinder adding new extensions to TCP and the NAT characteristics tests will give helpful information what to expect when creating NAT traversal methods. In addition, we include tests with Stream Control Transmission Protocol (SCTP) [13], Datagram Congestion Control Protocol (DCCP) [5] and Light-weight User Datagram Protocol (UDP-Lite) [6] traffic to determine whether the NAT devices support these new transport protocols.

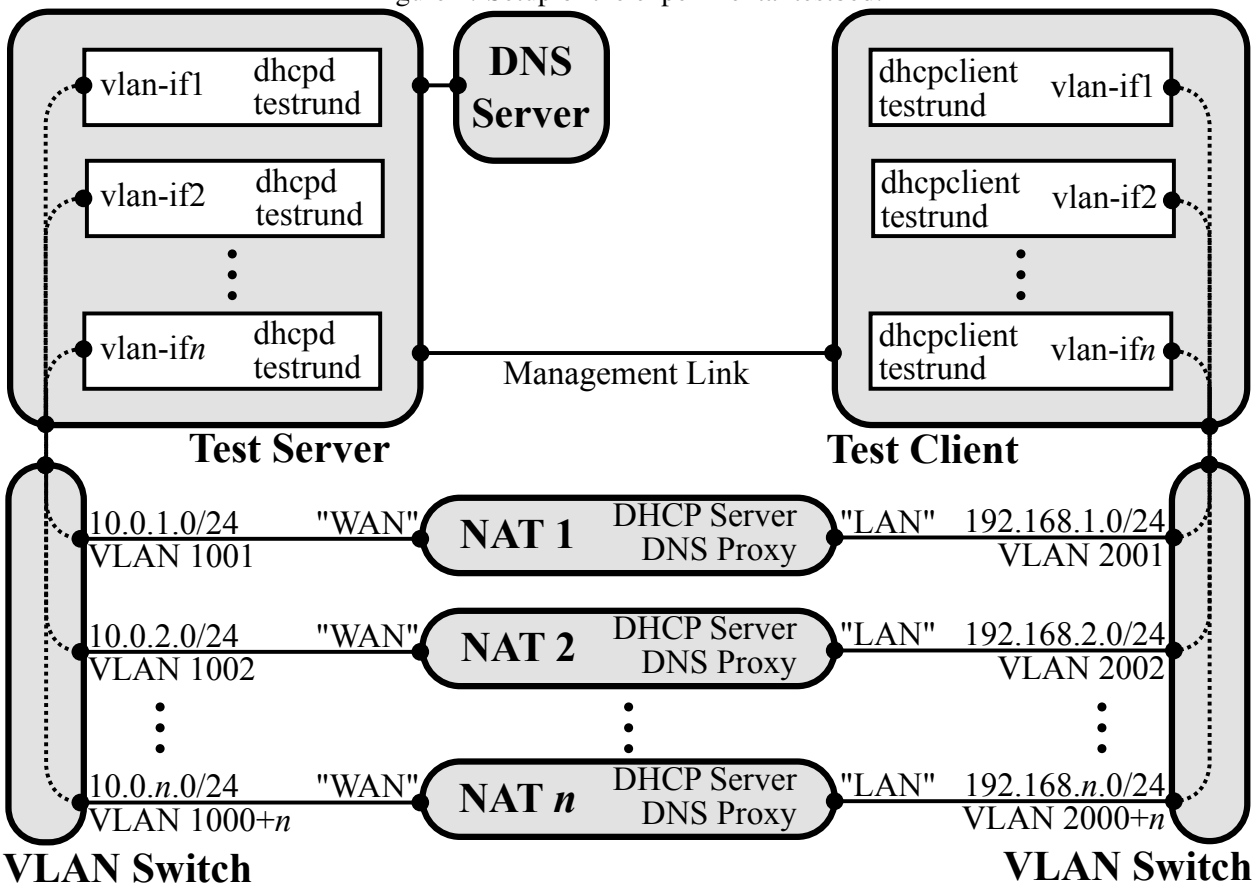
The rest of the report is organized as follows. In Section 2 we describe the testbed we use to run the tests. Section 3 describes the tests and presents the results of the tests and Section 4 concludes the findings.

2 Testbed Description

The testbed used in the experiments is shown in the Figure 1. The testbed consists of several servers, a HP 5412 zl switch and 42 NAT devices as listed in the Table 1. Over half of the devices were donated to the University of Helsinki to give a broader view of different home gateway devices abroad. Rest of the devices were bought in spring 2010 to get a picture of current devices that were available at that time and to get a picture what the consumers buy. The test servers are running Linux 2.6.32 kernels

Unfortunately, due to the age of the devices and the fact that the devices are consumer grade hardware, some of the devices failed during the testing and are not reported here. This lowered the number of devices from 48 to 42. The failed hardware included both system hardware and power supplies.

Figure 1: Setup of the experimental testbed.



The test servers are divided into two categories, the ”Internet” servers outside of the NAT devices and internal client servers inside the NAT. Both the test servers and the client servers run their instances of *runtestd* which is responsible of setting up test runs, capturing the traffic for analysis using *tcpdump* in both hosts and logging. All tests can either be run in parallel or serially depending on what kind of load the tests generate on the testbed. For example a test that explores the treatment of various header fields can be run in parallel on all devices but a throughput or similar test that requires notable amount of resources must be run serially as the traffic may overload the switches, servers or network interfaces and affect the test results.

Each of the NAT devices are connected to the ”Internet” servers using its “WAN” uplink port through a managed switch where each switch port has been configured to use its own separate virtual LAN (VLAN). The VLAN’s are used to keep the different NAT devices separate from each other. Each of the ”Internet” servers have two IP addresses per VLAN, which enables us to simulate multiple destinations. For some of the tests we enable a second network interface on the client servers to provide second connection to the NAT devices.

The management server is running a DHCP service [2] that provides each VLAN a separate private address block 10.0.x.0, where x is the unique number assigned for each NAT device in the testbed. The NAT devices use these to configure their ”WAN” interface and DNS proxies. The DHCP servers of the NAT devices are

Table 1: Home gateway models included in the study, with the shorthand "tags" used throughout this report

Vendor	Model	Firmware	Tag
A-Link	WNAP	e2.0.9A	*al*
Apple	Airport Express	7.4.2	*ap*
Asus	RT-N15	2.0.1.1	*as1*
	WL-500G Premium V2	3.0.3.5	*as2*
Belkin	Wireless N Router	F5D8236-4_WW_3.00.02	*be1*
	Enhanced N150	F6D4230-4_WW_1.00.03	*be2*
	Wireless G Router	F:3.00.03 H: F5D7234-4 v3 (01)	*be3*
	Wireless G Plus MIMO Router F5D9230-4 ver. 3000	3.02.76	*be4*
Buffalo	WZR-AGL300NH	R1.06/B1.05	*bu1*
D-Link	DIR-300	1.03	*dl1*
	DIR-300	1.04	*dl2*
	DI-524up	v1.06	*dl3*
	DI-524	v2.0.4	*dl4*
	DIR-100	v1.12	*dl5*
	DIR-600	v2.01	*dl6*
	DIR-615	v4.00	*dl7*
	DIR-635	v2.33EU	*dl8*
	WBR-1310	1.04	*dl11*
Edimax	6104WG	2.63	*ed*
Jensen	Air:Link 59300	1.15	*je*
Linksys	BEFSR41c2	1.45.11	*ls1*
	W54G	v7.00.1	*ls2*
	WRT54GL v1.1	v4.30.7	*ls3*
	WRT54GL-EU	v4.30.7	*ls5*
	WRT54G	OpenWRT RC5	*owrt*
	WRT54GL v1.1	tomato 1.27	*to*
Netgear	RP614 v4	V1.0.2_06.29	*ng1*
	WGR614 v7	(1.0.13_1.0.13)	*ng2*
	WGR614 v9	V1.2.6_18.0.17	*ng3*
	WNR2000-100PES	v.1.0.0.34_29.0.45	*ng4*
	WGR614 v6	V1.0.11_1.0.7	*ng6*
	WGR614	V1.40 Feb 18 2004	*ng7*
	WGT624 v4	V2.0.6_2.0.6NA	*ng8*
	WGT624 v3	v2.0.25_1.0.1NA	*ng9*
	MR314	V3.30(CF.0)	*ng10*
	RP114	V3.26(cd.0)	*ng11*
Netwjork	54M	Ver 1.2.6	*nw1*
SMC Barricade	SMC7004VBR	R1.07	*smc*
Telewell	TW-3G	V7.04b3	*te*
Unicom	WEP-72104G rev. 2	v4.2.3.18.1e	*un1*
Webee	Wireless N Router	e2.0.9D	*we*
ZyXel	P-335U	V3.60(AMB.2)C0	*zy1*

configured to distribute private address to clients from 192.168.x.0 blocks. The management server is also running a NTP server that provides synchronised time to both test and client servers.

The majority of the NAT devices in the testbed provide a switch capability and one device, *ap*, has only a wireless interface in addition to its "WAN" interface. Each of the NAT devices is connected to the test client through one of the "LAN" interfaces. (The *ap* is connected to the client host through a separate USB WLAN dongle.) The client hosts have a separate DHCP client listening on each separate VLAN and sets the interface with the information that the NAT device provides via its DHCP server. The DHCP client was modified to configure only the interface-specific routes and to not set up the default route.

3 Experiments

We mainly focus on exploring various NAT behavior in the transport layer (layer 4). As we know, the NAT devices in many cases cause problems with the normal protocol and application operation and make developing new protocols difficult. Thus, the first tests (UDPLite1, DCCP1 and SCTP1,2) try to explore the compatibility of the NAT devices with newer transport protocols, including Stream Control Transmission Protocol (SCTP), Light-weight User Datagram Protocol (UDP-Lite) and Datagram Congestion Control Protocol (DCCP). The following of tests focus on TCP specific NAT behavior, namely, what occurs to the TCP header fields and TCP options. The possible behavior of the NAT devices include dropping the packets which the NAT device does not understand, modifying the packet or just forwarding the packet to the receiver. In addition, the instantaneous NAT behavior is difficult to observe. Thus, we include several tests that emulate the specific functions of the Session Traversal Utilities for NAT (STUN) protocol [10] to map the diverse real-time NAT behavior with the UDP and TCP traffic. Especially, this allows the internal test client to emulate specific STUN properties for the purpose of probing the behavior of the NAT devices between the test client and the test server [7].

3.1 UDPLite1,2: UDP-Lite Support

3.1.1 Test Description

UDP-Lite is one of the new transport layer protocols and it has many similar properties with UDP. UDP-Lite is able to tolerate corrupted payload in the UDP-Lite packet. Instead of discarding all packets with partly corrupted payload like UDP does when checksum is enabled, UDP-Lite can set the checksum to cover only a part of the packet and let a receiver to compute the partial checksum. If the corrupted part of the payload does not reside within the checksum coverage the receiver accepts the partially corrupted packet. The RFC 3828 [6] presents that UDP-Lite would be very useful for applications such as Voice over IP (VoIP) and many video applications for which even partially corrupted packets would be usable. Therefore, it is necessary to explore if the NAT devices support UDP-Lite. In regard to the specification on the RFC 3828 [6], the checksum must be configured to cover either all octets of the UDP message or *8+* (equal or more than *8*) octets. Setting the checksum coverage to value *0* represents the checksum calculation covering the entire message, whereas value *8* means the checksum calculation only includes the 8 bytes of the UDP-Lite header.

In order to understand the UDP-Lite support for various NAT devices, the UDP-Lite tests have two scenarios to understand how the NAT devices handle the UDP-Lite packet with different checksum coverage. In the first case(UDPLite1), the checksum will cover the whole UDP-Lite packet and assign different ports to the server to determine whether the NAT devices successfully forward or discard UDP-Lite packets in the test. Different server ports are assigned to be well-known ports (picked from all ports in the well-known range), some ports in the registered range and random ports (randomly picked from the dynamic range) in order to discover whether the various ports will influence NAT to accept the UDP-Lite packet or not.

In second scenario of the test (UDPLite2), 8 byte checksum coverage (checksum coverage only includes the UDP-Lite header) is set and UDP-lite messages are sent to the external server using various destination ports similarly to the UDPLite1 test. The server port selection approach is the same as for the UDPLite1 test.

Table 2: Summary of the UDP-Lite and DCCP support test. °: Unmodified, ∗: ChksumError, •: Dropped

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2
udplite-chk-header-unkn	∗	∗	∗	∗	•	•	∗	∗	∗	∗	°	∗	•	°	•	∗	∗	∗	∗	∗	°	•
udplite-unkn	∗	∗	∗	∗	•	•	∗	∗	∗	∗	°	∗	•	°	•	∗	∗	∗	∗	∗	°	•
dccp-con-rdsp-unkn	∗	∗	∗	∗	•	•	∗	∗	∗	∗	°	∗	•	°	•	∗	∗	∗	∗	∗	°	•

	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
udplite-chk-header-unkn	∗	∗	•	∗	∗	•	•	•	•	•	•	•	•	∗	°	∗	∗	∗	∗	∗
udplite-unkn	∗	∗	•	∗	∗	•	•	•	•	•	•	•	•	∗	°	∗	∗	∗	∗	∗
dccp-con-rdsp-unkn	∗	∗	•	∗	∗	•	•	•	•	•	•	•	•	∗	°	∗	∗	∗	∗	∗

3.1.2 Results

The results on the row ”udplite-unkn” (udplite test with entire checksum coverage) in Table 2 and the row ”udplite-chk-header-unkn” (udplite test with 8 bytes checksum coverage) show that the behavior for each of the NAT devices is exactly the same for UDPLite1 and UDPLite2. However, the behavior of the NAT devices differ from each other. 28 NAT devices (out of total 42 devices) forwarded UDP-Lite packets to the external server and the rest of 14 NAT devices dropped the UDP-Lite packets. However, although those 28 NAT devices forward the UDP-Lite packets to the external server, the external server cannot successfully receive them. The reason for this is that 24 of the 28 NAT boxes have incorrect UDP-Lite checksum. The remaining four NAT devices (*dl4, dl11, ls1* and *smc*) forward the packets without translating the source IP address and port.

As a result, none of the NAT devices in the testbed support UDP-Lite and different NAT devices deal with the unknown protocol packets in various ways.

3.2 DCCP1: DCCP Support

3.2.1 Test Description

Deploying new transport protocols such as SCTP and DCCP might be hard since various middle-boxes and NAT devices may not understand the new protocols. If the protocols require more than just the basic NAT functions, i.e. translating source and destination address, the new protocol will not be able to work properly unless the NAT device implements protocol-specific functionality for these protocols.

The earlier DCCP and SCTP tests in [4] provided some evidence that the SCTP might work with some of the NAT devices and that none of the NAT devices supported DCCP. However, those tests were not thorough enough to prove that the NAT devices supported SCTP or DCCP. Since the previous DCCP and SCTP tests only used a single connection with a fixed IP address and port, and that the SCTP test showed that while a NAT device was able to translate at least the IP address, the tests did not fully prove that the protocol actually worked. To determine if the DCCP and SCTP actually work with any of the devices, we attempt to test SCTP and DCCP under various circumstances.

The previous DCCP test result showed that none of the NAT boxes supported DCCP even with a single DCCP connection. Based on that, creating more than one connection is not useful as the result would be the same. However, setting different ports for the external server might have an influence on the DCCP protocol support for some of the NAT boxes. Thus, based on the previous test, in DCCP1 test, we use the same list of server ports as in the UDPLite1 and UDPLite2 tests to explore if the destination port has any influence on the behavior of the NAT devices.

Table 3: Summary of the SCTP tests. •: Dropped, ‡: NoInitAck, †: FirstConSuccess, ^{o}: SecondConFailed, ∗: InitAckDropped, –: NoResult

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2
sctp-con-multi-11-rdsp	†^{o}	†	†^{o}	†^{o}	•	•	∗	†^{o}	†^{o}	†^{o}	‡‡	†^{o}	•	‡‡	•	†^{o}	†^{o}	†^{o}	†^{o}	†^{o}	‡‡	•
sctp-con-multi-stream		–			•	•	∗						•		•						‡‡	•

	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
sctp-con-multi-11-rdsp	†^{o}	†^{o}	•	∗	∗	•	•	•	•	•	•	•	•	†^{o}	‡‡	†^{o}	†^{o}	†^{o}	†^{o}	∗
sctp-con-multi-stream			•	∗	∗	•	•	•	•	•	•	•	•		•					∗

3.2.2 Results

In terms of the DCCP1 test, the results show that none of the NAT boxes on the row "dccp-con-rdsp-unkn" in Table 2 are able to establish a DCCP connection to the external server with the given destination ports. The final outcome is the same as with the earlier DCCP test in [4]. The results show that 28 out of 42 devices forward the DCCP packets but none do it correctly. Out of the 28 devices, 24 devices fail to calculate the DCCP checksum correctly and four devices, *dl4, ls1, smc* and *dl11*, do not translate the headers of the DCCP packets at all. The remaining 14 devices discard the DCCP packets. Therefore, the results outcome is that none of the NAT boxes support DCCP and that the NAT devices handle unknown protocol packets the same way as with the UDP-Lite tests.

3.3 SCTP1,2: SCTP Support

3.3.1 Test Description

The SCTP result in the earlier study [4] indicates that some of NAT boxes may successfully establish a single SCTP connection to exchange packets. But this result was not sufficient enough to prove that those NAT boxes properly support the SCTP. This is due to the fact that the NAT devices might only translate the IP header and correctly compute IP checksum (the SCTP checksum does not include the network-layer pseudo header). Thus, to determine if the devices support SCTP, two SCTP support scenarios are created.

In the first test, SCTP1, an internal client tries to establish a SCTP connection to the same external server using the same destination IP address but several different destination ports in order to determine if the different server ports will affect the NAT device. Also, to determine NAT boxes properly support SCTP, the next test, SCTP2, attempts to establish two SCTP connections from two different clients with different source address and port to the same external server. If the client is able to successfully establish the SCTP connection, the NAT device should handle the SCTP packets. The server port (port3) picking strategy is the same as with the UDPLite1 and UDPLite2 tests (well-known ports and random ports) in order to observe possible alternate behavior with different destination ports.

3.3.2 Results

From the result of SCTP1 test, various and complicated behavior (Dropped, NoInitAck, InitAckDropped and FirstConSuccess) have been observed and are shown on the row "sctp-con-multi-11-rdsp" in Table 3. There are 14 NAT devices that drop SCTP packets (dropped). Comparing with DCCP1 results, these 14 NAT

devices (total 42 NATs) also have the same behavior to drop DCCP packets. In addition, 4 NAT boxes (*dl4, ls1, smc* and *dl11*) can not translate the source address and pass first SCTP INIT Chunk through the NAT device without generating INIT-ACK Chunk back to the internal endpoints (NoInitAck). This might be due to the source address not being correctly translated by those NAT boxes and the external server is not able to respond with the SCTP INIT-ACK Chunk. Moreover, four NATs (*zy1, be3, ng10* and *ng11*) can simply translate the IP address and pass through the first SCTP INIT Chunk, but the following SCTP INIT-ACK Chunk will be dropped (InitAckDropped). Furthermore, the remaining 20 NAT devices can translate the destination and source addresses and successfully exchange messages via them (FirstConSuccess). However, as mentioned in [4], the SCTP checksum does not cover the network-layer pseudo header and the NAT box might only simply translate the IP header and this does not yet indicate that the NAT devices properly support SCTP.

The result of SCTP2 on the row "sctp-con-multi-stream" in Table 3 show that 19 out of those 20 NAT boxes mentioned above (the client has only one interface towards the *ap*, which is the only device with only WLAN connection. The testbed client server hardware does not currently support connections with two WLAN interfaces and cannot perform SCTP2 test), which were able to successfully establish one SCTP connection in the SCTP1 test, are not able to establish the second SCTP connection (SecondConFailed). This is due to the new internal endpoint's address, which the NATs cannot translate properly. In other words, the NAT boxes in the testbed act as pure NAT without any SCTP support code and do not properly support SCTP [12].

3.4 UDP1-Sub1,2: TCP&UDP NAT mapping classification test

3.4.1 Test Description

When an internal client needs to initialise an outgoing session with an external server, the NAT device will allocate an external IP address and port pair for this session. The external server can then use this address and port pair as the destination for packets to the internal client. Due to the loose definition of the NAT by the IETF, many different behaviors have been observed with the NAT devices. To classify each of the NAT devices in the testbed, the following UDP1-Sub1,2 and UDP2 tests have been carried out to discover the specific NAT Mapping and Filtering behavior of the devices. These mapping and filtering behaviors define how the NAT devices allocate external address and port pairs each of the packet flows through the device.

To achieve this goal, the tests are based on the RFC 5780 [7] and the test software emulates the necessary STUN properties to determine particular NAT behavior. For the TCP NAT behavior test, the tests can only detect the NAT Mapping behavior, whereas the UDP behavior test supports both the NAT Mapping and Filtering behavior detection.

The RFC4787 [1] specifies the NAT behavioral requirements for Unicast UDP, while the RFC5382 [3] specifies the NAT behavior requirements for TCP. Both of them require that the NAT must have an "Endpoint-Independent Mapping" behavior so that for example P2P applications are able to know and advertise the external address assigned to the internal peer and allow external peers to make contact with the internal peers. In this case, UDP1 test creates series of tests to discover if the NAT device uses the Endpoint Independent Mapping (EIM), the Address-Dependent Mapping (ADM) or Address and Port Dependent Mapping (APM) behavior. In order to understand these different NAT Mapping behaviors, we suppose that the client A with IP1 & Port1 has created a NAT binding to external server with IP2 & Port2. If another external server

Table 4: UDP NAT mapping and filtering classification. •: Yes, †: Exception

box	ng2	ls5	owrt	nw1	dl4	ls1	ng1	ls2	dl1	zy1	ed	ng3	dl3	al	as1	ls3	to	bu1	smc	dl2	dl5
EIM	•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•
ADM																					
APM							•														
IEF				•	•														•		
ADF						•				•		•			•						
APF	•	•	•				•	•	•		•		•	•		•	•	•		•	•

box	we	dl7	je	dl6	dl8	ap	ng4	be1	be2	te	as2	ng6	ng7	ng10	be3	dl11	ng8	un1	be4	ng11	ng9
EIM	•	•	•	•	•	•	•	•	•	•	•		•	•	†	•	•	•	•	•	•
ADM																					
APM												•			•	†					
IEF						•			•					•	•	•				•	
ADF			•		•		•	•					•			†	•				•
APF	•	•		•						•	•	•						•	•		

with IP3 & Port3 can still use that existing binding to communicate with the client IP1 & Port1, this NAT uses EIM. If the existing NAT binding is only able to be used to destination with same IP address (IP2) and different Port3, that NAT mapping behavior is ADM. If and only if the external server's address is exactly same with IP2 & Port2 in order to reuse existing binding, the NAT device uses APM.

In order to classify the NAT Mapping behavior, the first test UDP1-Sub1 is performed by the internal client sending request to the external server and the server will simulate the STUN server's functionality to return the client's mapped external address back to client. When this returned address is compared to client's local address, we are able to determine the NAT mapping behavior.

Moreover, according to the RFC 5780 [7], we need at most three tests to classify NAT Mapping behavior. The first test is used to check the UDP connectivity and the second one detects whether Mapping behavior is the EIM or not. If it is not, the third test will be used to verify whether it is ADM or APM. Also, this UDP1 test will test various UDP well-known ports and random ports (port selection in this test is same as the UDPLite1 test except numerous well-known TCP ports) as the external server port. The reason why this test picks numerous ports as the external server's port is to determine whether the different external server port will change the NAT Mapping behavior or not.

Regarding with the RFC 5780 [7], the UDP1-Sub2 test is quite similar to the UDP1-Sub1 test. The Sub2 test uses at maximum three tests to classify the TCP NAT mapping behavior. We then compare the results of the Sub1 and Sub2 tests to explore if there are differences between the UDP NAT mapping behavior and TCP NAT mapping behavior.

In addition, this test is also performed over numerous well-known TCP ports and random ports (we use the same list as with the UDPLite1 test except various well-known UDP ports) to determine if the devices change their mapping behavior depending on the external port.

3.4.2 Results

The UDP1-Sub1 experiment result shows that only *ng1, ng6* and *be3* out of 42 NAT devices use Address and Port Dependent Mapping and the rest of the NAT devices use Endpoint Independent Mapping as shown in Table 4. This means that almost none of the NAT devices do not care of the external server port and most of the NAT boxes follow the RFC 4787 [1] requirement. The results also show that different server ports have no influence on the UDP NAT Mapping behavior for most of the NAT devices. The NAT box *be3* with port

Table 5: TCP NAT mapping classification. •: Yes, †: Exception

box	ng2	ls5	owrt	nw1	dl4	ls1	ng1	ls2	dl1	zy1	ed	ng3	dl3	al	as1	ls3	to	bu1	smc	dl2	dl5
EIM		•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•
ADM																					
APM	•						•														

box	we	dl7	je	dl6	dl8	ap	ng4	be1	be2	te	as2	ng6	ng7	ng10	be3	dl11	ng8	un1	be4	ng11	ng9
EIM	•	•	•	•	•		•	•	•	•	•		•		•		•	•	•		•
ADM																•					
APM						•						•		•		†				•	

1701 changes its the NAT mapping behavior from APM to EIM. Moreover, the NAT device *dl11* with the port *500* changes from EIM to APM. Thus, while some of the devices do change their behavior according to the destination port, most of the devices do not behave differently over different ports.

After the UDP1-Sub2 experiment, the result shows that 6 of the 42 NAT devices in Table 3.4.2 use the APM and 35 NAT boxes follow the RFC 5382 [3] and use the EIM. The NAT *dl11* uses the ADM behavior except when the port 33434 is assigned for the external server. It will then change its NAT mapping behavior from ADM to APM. Comparing with the UDP1-Sub 1 tests results, UDP1-Sub2 demonstrates that these NAT devices (*ng2, ap, ng10, be3, dl11* and *ng11*) have different NAT Mapping behavior between TCP and UDP. Therefore, the NAT Mapping behavior for TCP and UDP is not always same for some of devices and most of NAT devices use the same mapping behavior. In addition, no matter what port is assigned to the external endpoint, the TCP NAT Mapping behavior will not change apart from the device *dl11*. To our surprise, the NAT device *dl11* has different mapping behavior in few situations.

3.5 UDP2: UDP NAT Filtering Classification

3.5.1 Test Description

Since diverse NAT filtering behavior affects the normal operation of VoIP and P2P applications, the RFC780 [7] and RFC 5382 [3] respectively recommend to use Endpoint-Independent Filtering in the transparent situation and Address-Dependent Filtering in the strict Filtering situation, it is necessary to take UDP NAT Filtering behavior into consideration. The UDP2 test also follows the RFC 5780 [7] and uses at most three tests to determine the filtering behavior of the NAT devices. The test simulates certain STUN server functions to determine UDP NAT Filtering behavior. The first test is to verify the UDP connectivity, and then, for the second test, the client sends ”change port and address” request to the external server. This request asks the server to emulate STUN server to use an alternative address to reply to the client. If the client can receive the reply message from the different source IP address and port, the NAT device uses thee Endpoint-Independent Filtering (EIF); otherwise, a third test is needed to determine the filtering behavior. The client sends ”change port only” request to the external server. The server then sends a reply to the client from an another port than the port used initially. If the client gets the reply from the server, regardless of external server's port, the NAT Filtering behavior is Address-Dependent Filtering (ADF); Otherwise, it is the Address and Port-Dependent Filtering (APF). The UDP2 experiment uses the same external ports as with the previous UDP1-Sub1 test in order to verify whether the different external servers' port has an influence on the UDP NAT Filtering behavior.

3.5.2 Results

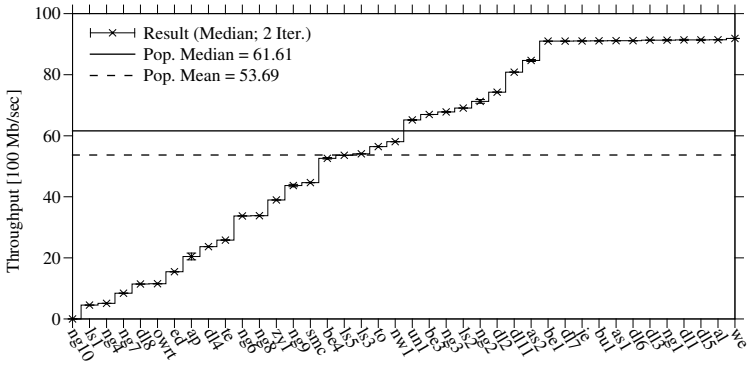
After tests, 9 NAT devices (total 42 NATs) use EIF, 11 NAT devices use ADF, and 22 NAT devices use APF in Table 4. In addition, the common UDP NAT Filtering behavior of the NAT *dl11* is EIF, but it will change to ADF when the external server uses port *500*. The results also show that setting the external ports to either well-known port range or to some of the widely used ports does not affect the filtering behavior.

3.6 UDP3: UDP Throughput

3.6.1 Test Description

In order to explore how NAT boxes impact the UDP throughput performance, the throughput test series use Jugi's Traffic Generator (jtg) tool to measure the client-to-server upload UDP throughput, the server-to-client download UDP throughput and simultaneous upload and download UDP throughput. The Constant Bit Rate (CBR) of jtg is set to be 100Mb/sec (which includes IP and UDP headers) for each direction and each throughput test is set to run for 60 seconds. The UDP3-Sub1 and UDP-3-Sub2 experiments focus on the unidirectional UDP bandwidth performance for upstream and downstream directions, respectively. The UDP3-Sub3 test measures the throughput performance when data is transferred simultaneously to both directions.

Figure 2: UDP upstream throughput.

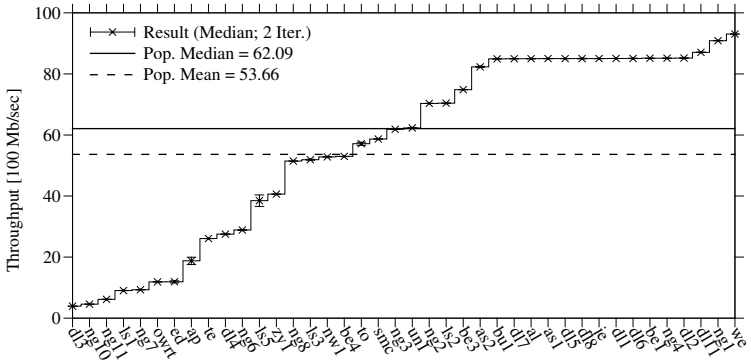


3.6.2 Results

Based on the result of the UDP throughput in upstream direction (UDP3-Sub1) shown in Figure 2, only 12 devices (out of 40 NAT devices) are able to sustain over 90Mb/sec throughput, but the rest of NAT devices are not able to reach that high throughput. The worst throughput performance in the upstream case is the NAT device *ng10* which is able to deliver traffic roughly at rate 0.01Mb/sec only and thereby experiences extremely high packet loss rate. Furthermore, the throughput of 16 NAT devices is less than the mean throughput of 53.69Mb/sec.

Our tests also show that devices *be2* and *ng11* do not cope with the 100Mb/sec CBR. The device *be2* stops forwarding the UDP packets for the test flow after one second and the *ng11* stops forwarding the test

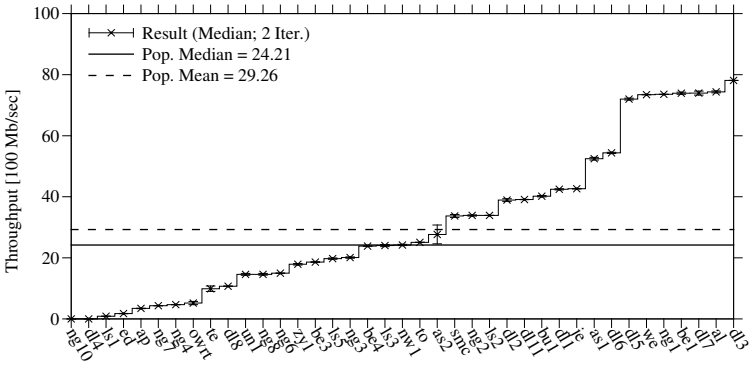
Figure 3: UDP downstream throughput.



packets immediately. While the exact cause is unknown, we suspect that the devices might have some builtin function that causes the devices to stop forwarding the packets at the high rate used in this test.

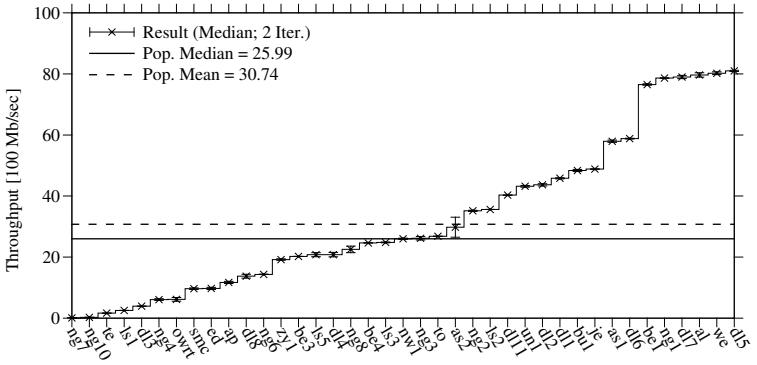
The results in Figure 3 show that in the downstream direction only two devices, *ng1* and *we*, out of 40 NAT devices (*be2 and ng9* failed to sustain the UDP packet flow of 100 Mb/sec and stopped forwarding the packets after a several seconds) can sustain over 90Mb/sec throughput. In addition, 17 NAT boxes do not even reach the UDP throughput of the mean value of 53.66Mb/sec. Comparing the upload and download throughput performance, some devices illustrate a diverse difference between the Upstream and downstream. For example, the devices *dl8* and *ng4* have much better UDP throughput performance for the downstream direction compared to the upstream direction. In contrast to those two devices, the NAT device *dl3* has significantly higher UDP throughput on the upstream direction than on the downstream direction. Moreover, the median value of these two tests are 61.61Mb/sec and 62.09Mb/sec in the upstream and downstream direction, respectively. Also, these two unidirectional tests present that none of NATs can reach to 100Mb/sec throughput.

Figure 4: Summary of the UDP upload throughput while downloading.



The results of the simultaneous upstream and downstream tests UDP3-Sub3 are a lot worse than unidirectional UDP throughput test results. None of NAT boxes (total 39 NATs *be2, ng9, ng11* behaved the same way as in the previous throughput tests), can reach 100Mb/sec in the bidirectional measurement, and none of NAT boxes can achieve above 80 Mb/sec throughput in both upstream and downstream direction as can be observed from Figure 4 and Figure 5. Moreover, the median throughput of the bidirectional test is

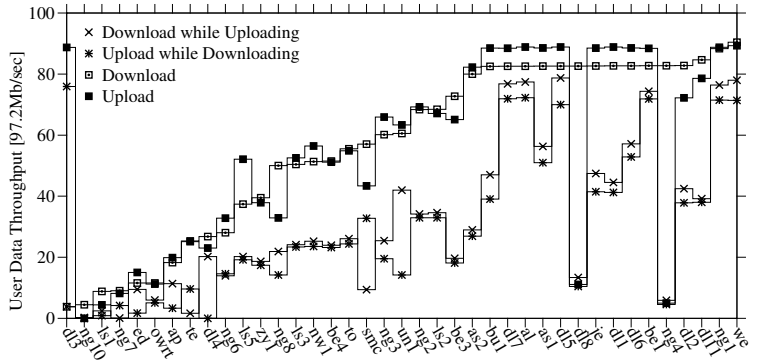
Figure 5: Summary of the UDP download throughput while uploading.



24.21Mb/sec and 25.99Mb/sec for upstream and downstream direction, respectively. That is much less than the throughput in the unidirectional tests above. In addition, typically either of the directions tend to dominate the NAT boxes For example, *dl4* device only can maximally support about 0.03Mb/sec in upstream direction and 22.21Mb/sec downstream direction in the bidirectional throughput test, while in the unidirectional throughput tests it was able to sustain 23.72Mb/sec in the upstream direction and 27.57Mb/sec in the downstream direction. Furthermore, in the bidirectional throughput test the downstream throughput of the NAT box *ng7* was much worse than its throughput in the unidirectional test.

Overall, many NAT boxes demonstrate the poor UDP throughput performance and none of NATs can reach roughly 100Mb/sec.

Figure 6: Summary of the medians of UDP data throughput (data only) for UDP3-Sub1,2,3 tests.



The following Figure 6 and Figure 7 respectively illustrate the summary of UDP throughput medians in each test above without or with headers (UDP and IP headers).

3.7 TCP1: Does NAT Devices Rewrite TCP Header Fields

3.7.1 Test Description

While some fields in the TCP header need to be rewritten when a NAT device translated the packet, such as source and destination ports and checksums fields, some other fields might also be changed. In this test we

Figure 7: Summary of the medians of UDP link throughput (including UDP+IP headers) for UDP3-Sub1,2,3 tests.

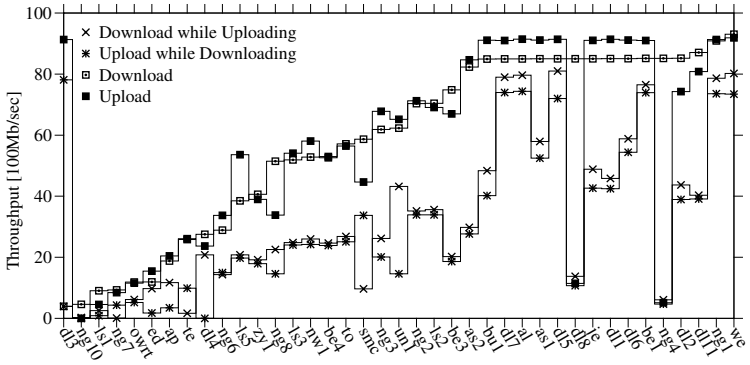


Table 6: Summary of the TCP header test.

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
window-1024	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
window-2048	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
window-4096	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
window-65535	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
seq	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

will check if the NAT devices change the field values of different window sizes and sequence numbers. The sequence numbers are gathered from other tests while the window size is set 1024, 2048, 4096 and 65535. We check if the SYN packet comes through and if the field is changed.

3.7.2 Results

The tests indicate that the NAT devices are not very prone to changing TCP headers, except the fields the NAT needs to change depending on its classification, i.e. destination and source IP address and port pairs and cheksum field.

3.8 TCP2: Reserved Bits in the TCP Header

3.8.1 Test Description

The purpose of this test is to find out how the different home gateway devices treat the six reserved bits of the TCP header. This is important information since the different reserved bits or bit combinations might be used in the future for some purposes.

The test is performed by creating a TCP SYN packet, setting the reserved bits in the TCP header to all possible combinations and then checking if the packets with different bit combinations go through the NAT device, if the packet are dropped or if the fields are cleared.

Table 7: Summary of the TCP reserved bits. •: No change, °: changed, †: dropped

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
1	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
2	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
3	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
4	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
5	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
6	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
7	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
8	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
9	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
10	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
11	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
12	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
13	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
14	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•
15	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	†	•	•	•	•	•	•	•	•	•

Table 8: Summary of the TCP RST test. •: No change, °: changed, †: dropped

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
tcp-rst-WtoL	°	°	°	°	°	°	•	°	°	°	°	°	°	°	°	°	°	°	°	°	°	•	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°
tcp-rst	°	°	°	°	°	°	•	°	°	°	°	°	°	°	°	°	°	°	°	°	°	•	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°	°

3.8.2 Results

The results are shown in Table 7, where the number on each of the rows indicates the reserved bit combination as 6-bit integer. The results indicate that most of the NAT devices do not modify the reserved bits in the SYN packets from LAN to WAN. There were only two devices, *ap* and *ng7*, that modified or dropped the packets. The *ng7* passes bit combinations indicating numeric values from one to three but drops the packets containing rest of the bit combinations. The most surprising results was *ap*, which changed all reserved bits to zero.

3.9 TCP3: Does TCP RST Tear Down NAT Bindings

3.9.1 Test Description

The goal of this test is to see what happens to the NAT bindings if a TCP RST is sent either from the client or from the server.

The test is performed by first creating a TCP connection through a NAT device. The client inside the NAT then starts sending data to the server on the other side of the NAT device. The server closes the corresponding TCP socket as soon as there is readable data in the socket. This should trigger TCP RST on arrival of any later packet. We explore tcpdump packet captures to determine if the RST packet has removed the binding. This should be visible from comparing the order of the packets going through the NAT device.

3.9.2 Results

The results in Table 8 seem to indicate that most NAT devices do not remove the NAT binding immediately after a TCP RST has been seen by the device. The packet captures show that the test client receives multiple

Table 9: Summary of the TCP SYN with payload test. •: No change, °: changed, †: dropped

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
payload	°	•	•	•	•	°	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	•	•	•	•	°	•	•	•	°	•	°	•	°	°	•
no-payload	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

TCP packets with ”R” flag set. Only one device, *be3*, seems to remove the binding as only the first RST packet is received.

3.10 TCP4: Does TCP SYN with Payload Go Through NAT Devices

3.10.1 Test Description

The purpose of this test is to see if TCP SYN segments can be used to transfer application data in the payload of the segment. If the devices do not drop the SYN packets or strip the payload, the data in the payload could be used for transferring some application specific data that the application could act early on, for example. The test is done by sending a TCP SYN segment with a short string attached as payload to the segment and checking whether the packet carrying the segment arrived and whether the payload was still present the SYN segment.

3.10.2 Results

This test indented to find out how feasible it is to send data in the first TCP SYN packet of the TCP three-way handshake. The results in Table 9 show that nine of the NAT devices drop the SYN packet with payload.

3.11 TCP6: TCP Options in SYN Packets Handling

3.11.1 Test Description

The goal is to see what happens to the packets which contain TCP options. These include both known and unknown or proposed TCP options. As with many other tests, the purpose is to determine how feasible deploying new options is. We also try to determine if the NAT devices remove or change the option values or if they add options to the packets.

3.11.2 Results

The test results shown in Table 10 indicate that for the most of the options, 36 devices do not change options. The devices *owrt, al, we, te, un1* and *be4* add the option *MSS* to reflect their individual MSS value. When the option *MSS* was set to 1024, none of the mentioned devices changed the value. When examining the assigned options, from one to 30, the devices either pass the options unchanged or with the changes described above. Only one device, *be4*, seems to have different behavior for certain options.

Table 10: TCP options in SYN packets test. •: No change, °: changed, †: dropped

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
1	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
2	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•
3-10	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
11	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
12-23	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
24	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
25-111	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
112	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
113-126	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
127	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
128-168	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
169	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
170-175	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
176	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
177-182	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
183	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
184-189	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
190	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
191-241	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
242	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•
243-255	°	•	•	•	•	•	•	°	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	°	•	°	•	°	°	•

Table 11: TCP options in data packets test.

	al	ap	as1	as2	be1	be2	be3	be4	bu1	dl1	dl11	dl2	dl3	dl4	dl5	dl6	dl7	dl8	ed	je	ls1	ls2	ls3	ls5	ng1	ng10	ng11	ng2	ng3	ng4	ng6	ng7	ng8	ng9	nw1	owrt	smc	te	to	un1	we	zy1
1-255	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

The results indicate that only a small portion of NAT devices change the content of the TCP options. The most prominent change is the TCP option Maximum Segment Size. The devices *owrt, al, we, te, un1* and *be4* either add it or change the value to lower size if the value is higher than their supported MSS. If the value of the MSS option was set to lower than the individual value of the devices, the devices did not alter the value. This result indicates that TCP options can be used in the SYN packets and new options can be added to the existing specification.

3.12 TCP7: TCP Options in Data Packets Handling

3.12.1 Test Description

This test is similar to the TCP6-test. While the test goal is basically the same, that is, are TCP options removed, changed or are the packets dropped, we now test TCP data packets with the PUSH-flag set. Also in the TCP6-test we only send the initial SYN packet to the server and checked its Options field. In this test we create a TCP connection by hand using *scapy*. The server creates a TCP listening socket and we send a crafted SYN packet to the server. We then capture the SYNACK from the server and send a proper crafted ACK packet back to the server to complete the TCP handshake. We then craft a TCP packet with the Options field set and send it to the server and check if the packet arrived and was the Options field changed or removed.

Since we used scapy to create the initial SYN packet, we used raw sockets. So when the SYNACk arrived from the server to the client, the kernel generated a TCP RST packet as it was not aware of the SYN packet. To take care of this RST packet, we used *iptables* to drop the RST packet.

3.12.2 Results

The results for TCP options in data packets shown in Table 11 look similar to results in Table 10, with only one exception. The NAT devices, which added TCP option MSS to the SYN packets did not add the option to the data packets as could be expected. Also compared to the TCP6, the *be4* now behaved consistently and did not remove or drop options from the TCP header. This result indicates that TCP Options can be used and new options can be deployed.

4 Conclusions

In this report we extended the earlier experimental study [4]. In this study, we focused on the transport layer (layer 4) behavior of home gateways. Several experiments were carried out to determine the support for new transport protocols such as SCTP, DCCP and UDPLite. Our tests show that slightly surprisingly none of the NAT devices in our testbed support these protocols properly and prevent the deployment of these protocols. Furthermore, throughout the UDP throughput test, the quality of many of the NAT devices seriously affect the UDP throughput performance, especially in simultaneous upstream and downstream test.

We also probed how the NAT devices behaved when encountering different TCP options and field values that have either been standardized, unknown or reserved. Many of the NAT devices behaved more or less properly by translating the packets and preserving the field values. Only in some cases some of the devices either changed the values or dropped the packets. This indicates that at least in some cases new options and field values can be used for new options. Hence, discovering how the various NAT devices behave is essential to provide information to developers on what can be done and what to take in consideration when developing new transport protocols and applications.

5 Acknowledgements

This work was supported by TEKES as part of the Future Internet programme of TIVIT (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT).

The authors would like to thank Lars Eggert and Pasi Sarolahti for their suggestions and fruitful discussions during the study. We would also like to thank all individuals who donated their home gateway hardware used as a part of the testbed. In addition, we thank CSC - IT Center for Science, for donating the testbed servers to the University of Helsinki. The servers were originally a part of the CSC Sepeli cluster.

References

[1] F. Audet and C. Jennings. Network Address Translation (NAT) Behavioral Requirements for Unicast UDP. RFC 4787 (Best Current Practice), Jan. 2007.

[2] M. Chatel. Classical versus Transparent IP Proxies. *Internet RFCs, ISSN 2070-1721*, RFC 1919, Mar. 1996.

[3] S. Guha, K. Biswas, B. Ford, S. Sivakumar, and P. Srisuresh. NAT Behavioral Requirements for TCP. RFC 5382 (Best Current Practice), Oct. 2008.

[4] S. Hätönen, A. Nyrhinen, L. Eggert, S. Strowes, P. Sarolahti, and M. Kojo. An Experimental Study of Home Gateway Characteristics. In *Proceedings of the 10th annual conference on Internet measurement*, IMC '10, pages 260–266. ACM, 2010.

[5] E. Kohler, M. Handley, and S. Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340 (Proposed Standard), Mar. 2006.

[6] L.-A. Larzon, M. Degermark, S. Pink, L.-E. Jonsson, Ed., G. Fairhurst, and Ed. The Lightweight User Datagram Protocol (UDP-Lite). *Internet RFCs, ISSN 2070-1721*, RFC 3828, July 2004.

[7] D. MacDonald and B. Lowekamp. NAT Behavior Discovery Using Session Traversal Utilities for NAT (STUN). *Internet RFCs, ISSN 2070-1721*, RFC 5780, May 2010.

[8] J. Postel. User Datagram Protocol. *Internet RFCs, ISSN 2070-1721*, RFC 0768, Aug. 1980.

[9] J. Postel. Transmission Control Protocol. *Internet RFCs, ISSN 2070-1721*, RFC 0793, Sept. 1981.

[10] J. Rosenberg, R. Mahy, P. Matthews, and D. Wing. Session Traversal Utilities for NAT (STUN). RFC 5389 (Proposed Standard), Oct. 2008.

[11] P. Srisuresh and M. Holdrege. IP Network Address Translator (NAT) Terminology and Considerations. *Internet RFCs, ISSN 2070-1721*, RFC 2663, Aug. 1999.

[12] R. Stewart and M. Tuexen. Stream Control Transmission Protocol (SCTP) Network Address Translation. Internet-Draft draft-stewart-behave-sctpnat-03, Internet Engineering Task Force, Nov. 2007. Work in progress.

[13] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. *Internet RFCs, ISSN 2070-1721*, RFC 2960, Oct. 2000.