# An Experimental Study of Home Gateway Characteristics

Seppo Hätönen
University of Helsinki
shatonen@cs.helsinki.fi

Aki Nyrhinen
University of Helsinki
anyrhine@cs.helsinki.fi

Lars Eggert
Nokia Research Center
lars.eggert@nokia.com

Stephen Strowes
University of Glasgow
sds@dcs.gla.ac.uk

Pasi Sarolahti
HIIT / Aalto University
pasi.sarolahti@iki.fi

Markku Kojo
University of Helsinki
kojo@cs.helsinki.fi

## ABSTRACT

Many residential and small business users connect to the Internet via home gateways, such as DSL and cable modems. The characteristics of these devices heavily influence the quality and performance of the Internet service that these users receive. Anecdotal evidence suggests that an extremely diverse set of behaviors exists in the deployed base, forcing application developers to design for the lowest common denominator. This paper experimentally analyzes some characteristics of a substantial number of different home gateways: binding timeouts, queuing delays, throughput, protocol support and others.

## Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Performance Attributes; C.2.6 [**Computer Communication Networks**]: Internetworking

## General Terms

Experimentation, Measurement, Performance

## Keywords

Home Gateways, Behavior, Characteristics, Measurements

## 1. INTRODUCTION

Many residential and small business users connect to the Internet through "home gateways" – a colloquial term for customer-premises equipment (CPE) that includes DSL and cable modems, WLAN access points and even some kinds of (wired) Ethernet switches. The common defining characteristic of home gateways is that they do *not* just perform Ethernet switching or basic IP forwarding over various link-layer technologies. They also perform higher-layer operations, such as network address translation (NAT) or, more often, network address and port translation (NAPT). Usually, they include traffic filtering "firewall" functions, act as DHCP servers, and proxy DNS traffic. Many models offer other advanced features, including traffic prioritization, shaping, web and email virus and "phishing" protection, etc.


*IMC'10,* November 1–3, 2010, Melbourne, Australia.


Despite their wide deployment, few standards exist that regulate what functions home gateways should perform and how they should perform them. The relevant standards bodies have ignored home gateways in the past, and only recently begun to document best current practices for some of the functions they perform, such as translating various protocols [3, 11, 29] or DNS proxying [4].

The result is that the performance and the behavior of the network service that applications in residential and small-business deployments experience depends significantly on the specific home gateway model that provides Internet connectivity. Different home gateway models differ in many application-observable aspect, including NAT schemes [28], NAT binding timeouts, ICMP [23] handling, traffic filtering, queuing, buffer sizes, etc. Incomplete support – or buggy implementation – of common functions, *e.g.*, DNS or DHCP, is another significant source of behavioral variability.

The experimental study in this paper measures and analyzes the behavior of a substantial number of home gateways, including NAT binding timeouts, queuing delay, throughput, and protocol support, to characterize typical behaviors found in many home gateway models. This allows applications to understand the kinds of practices they can expect to encounter in the deployed base.

## 2. RELATED WORK

A few previous studies have focused on measuring NAT binding timeouts. In a study of UDP binding timeouts in a peer-to-peer network of ca. 3,500 peers, the majority (62%) of bindings were found to time out between 2 and 2.5 min [7]. In the context of investigating energy efficiency of mobile handsets, timeout behavior is important to throttle keepalives and minimize battery usage. The default connection timeout values documented by vendors [13] are approximately in line with these observations.

Other studies have looked at the inbound packet filtering behavior of NATs in the presence of valid bindings created by outbound traffic. An experimental analysis of this behavior on a small set of NATs is documented in [14] using the terminology defined in [28]; other tests focus on hair-pinning and ICMP forwarding. The memo highlights that NAT behavior depends on whether a NAT attempts to preserve port numbers for external mappings: a NAT can exhibit different behaviors for different mappings. Also highlighted are techniques to systematically investigate binding timeout behavior, but no binding timeout results are presented.

Peer-to-peer NAT traversal for UDP and TCP is covered in [10], with a focus on "hole-punching" techniques to make peer-to-peer applications work through NATs. The paper gives a useful characterization of different types of NATs, and describes the properties of a "well-behaving" NAT that supports hole-punching.

The success rates of various NAT traversal techniques for establishing direct TCP connections between two hosts located be-

hind NATs are not as good as those for UDP [12]. The study performed in [12] looks at only a small set of NATs, but none of the techniques presented guarantee the successful creation of an operational TCP connection, the best was a variant of STUNT [21] with a success rate of 89%. Some of the traversal techniques would benefit from a better understanding of how NATs handle unusual packet sequences (for TCP connection negotiation) or ICMP packets.

Another measurement study on NAT behavior [19] determines the applicability of primarily TCP-based NAT traversal techniques to cellular networks. The study treats the network as a "black box" and does not identify NAT manufacturers and models, or if routing policies for different classes of subscribers were in place, which leaves it open whether the reported behaviors are only due to NATs.

Various aspects of TCP interactions with the network are measured in [21]. The results suggest that different TCP options do not appear to cause problems for a TCP connection, except in certain rare cases. One example is middleboxes that shift TCP sequence numbers in the header without considering that certain TCP options also contain them (*e.g.*, SACK [20].) Slightly more problematic are the use of ECN [24] or PMTU discovery [22], which may cause loss of TCP SYN segments and therefore failed connection attempts. The results also indicate that the use of IP options leads to failure in most cases. Because the experiments were conducted end-to-end, the results do not identify whether buggy host implementations or intermediaries along the path caused these issues. Our study does not currently replicate these measurements, but we plan to expand it in the future to confirm their results.

Several studies evaluate the support of DNSSEC [2] in home gateways [1, 5, 9]. This includes testing for support of DNS over TCP, which our study also measures, but we do not currently perform exhaustive tests for DNSSEC support.

# 3. EXPERIMENTAL METHODOLOGY

This section describes the testbed used for the experiments presented in this paper and describes how each measurement result presented in Section 4 was obtained.

## 3.1 Testbed Setup

Figure 1 illustrates the experimental testbed used for this study, which consists of a test server and a test client, both running Linux 2.6.26 kernels, the *hiit.fi* DNS server, several HP-2524 VLAN switches and the different home gateway models listed in Table 1. Table 1 also defines a unique "tag" for each home gateway, which is used in the remainder of this paper as a shorthand to identify each specific device. All network links are using 100Mb/sec Ethernet.

The "WAN" uplink port of each home gateway connects to the test server through a switch on a separate VLAN. The test server runs a DHCP service that provides information about the global DNS server and leases a different private address block [25] on each VLAN, which the home gateways use to configure their uplink "WAN" interfaces and DNS proxies.

The "LAN" port of each home gateway connects to the test client through a second switch, again on a separate VLAN. The test client runs a separate DHCP client to set up each VLAN interface with the information that each home gateway provides via its DHCP server. The DHCP client is modified to configure only interface-specific routes.

The test server and test client are also directly connected through a management link, which is used to coordinate the measurements. Client and server run an instance of the *testrund* daemon, which is responsible for setting up and performing each measurement, as well as exporting all captured measurement data afterwards. A given measurement is run in parallel across all home gateways in the testbed, except for the throughput test, which measures each home gateway separately to avoid overloading the test network.

| 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* |
| Belkin | Wireless N Router | F5D8236-4_WW_3.00.02 | *be1* |
| | Enhanced N150 | F6D4230-4_WW_1.00.03 | *be2* |
| 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* |
| | DI-604 | v3.09 | *dl9* |
| | DI-713P | 2.60 build 6a | *dl10* |
| Edimax | 6104WG | 2.63 | *ed* |
| Jensen | Air:Link 59300 | 1.15 | *je* |
| Linksys | BEFSR41c2 | 1.45.11 | *ls1* |
| | WR54G | 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 v4 | V5.0_07 | *ng5* |
| Netwjork | 54M | Ver 1.2.6 | *nw1* |
| SMC Barricade | SMC7004VBR | R1.07 | *smc* |
| Telewell | TW-3G | V7.04b3 | *te* |
| Webee | Wireless N Router | e2.0.9D | *we* |
| ZyXel | P-335U | V3.60(AMB.2)C0 | *zy1* |

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

## 3.2 Measurement Methodology

Using the testbed described above, several different measurements were performed across the set of home gateways listed in Table 1, including determining the timeout values used for UDP and TCP bindings under various conditions, testing TCP throughput, and determining whether UDP- and TCP-related ICMP messages are translated correctly, among other measurements.

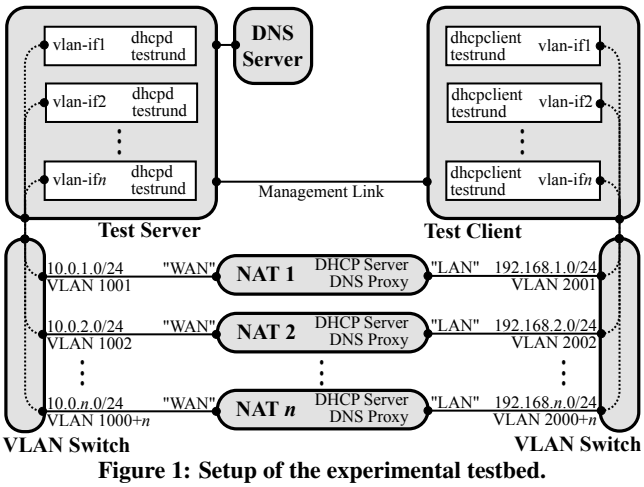


**Figure 1: Setup of the experimental testbed.**

### 3.2.1 UDP Binding Timeouts

UDP is a connectionless protocol without explicit connection startup or teardown handshakes. NATs therefore create UDP bindings when they observe a packet exchange, and remove bindings some period of time after the last observed packet on a flow – the *binding timeout*. A few critical services such as DNS use UDP, and it is becoming increasingly common to encapsulate other traffic inside UDP. Consequently, understanding the binding timeouts home gateways apply to UDP traffic is important, *e.g.*, to determine at which rate keepalives need to be sent.

To explore this question, we define several experiments to measure UDP binding timeouts under different conditions. In each case, the client sends UDP packets on a specific source/destination port pair to the server to create a binding, and has a modifiable "sleep timer". When the sleep timer expires, the client uses the management link to instruct the server to send a response packet back via the home gateway. Depending on whether the client receives this response packet, it knows whether the NAT binding is still active or not. Binding timeouts are determined through a modified binary search: The client retains the longest observed binding lifetime and shortest binding expiration, and on the next iteration sets the sleep timer to be their midpoint. The modification to the binary search allows the start of each search iteration to be identical to the first search. The test stops when it has converged to within one second. We define the following five UDP tests:

**UDP-1**: *Solitary outbound packet.* This test measures how long a NAT maintains a UDP binding after the test client sends a single UDP packet to the server. The server does not send traffic to the client, apart from the packet triggered by the sleep timer, and the client does not send any further traffic.

**UDP-2**: *Solitary outbound packet, multiple inbound packets.* The intent of this test is to determine if inbound traffic refreshes a binding, compared to *UDP-1*. The test client sends a solitary UDP packet to the test server and then remains silent. The server sends a stream of responses across the binding, and increases the delay between each response packet until the binding times out.

**UDP-3**: *Multiple outbound and inbound packets.* The intent is to determine whether outbound traffic refreshes a binding. This test is similar to *UDP-2*, except that the client sends another packet to the server whenever it receives a response packet from it.

**UDP-4**: *Binding and port-pair reuse behavior.* This test determines if a home gateway prefers to use the original source port as the external port for a binding, and if it waits before it reuses an expired binding for the same flow (*i.e.*, the same 5-tuple). This is observed from the *UDP-1* test. The port-pair reuse behavior is determined from the behavior of the binding created in the binary search iteration that follows immediately after a previous binding expires. If the device creates a new binding, as indicated by a changed source port, it is likely that the device prevents immediate binding reuse for the same flow.

**UDP-5**: *Binding timeout variations for different services.* A NAT might use different binding timeouts for different services, *i.e.*, different well-known destination port numbers. This test is identical to *UDP-2*, but tests different well-known server ports.

### 3.2.2 TCP Handling

TCP connections involve an explicit creation and tear-down phase, and NATs can observe the tear-down handshake to remove bindings immediately. However, TCP connections may idle for long periods of time (if no TCP or application keepalives are used) and endpoints may silently fail. A NAT therefore cannot assume that it will always be able to observe a tear-down handshake. Also, the number of bindings on a NAT is limited by the available ports.

The following TCP tests are carried out using Linux 2.6.26 on both the client and server, with the congestion control algorithm set to *Reno* and the Linux TCP options – SACK, timestamps, window scaling, F-RTO, D-SACK and control block interdependence (CBI) – disabled.

**TCP-1**: *TCP binding timeouts.* Although a NAT does not necessarily need to remove TCP bindings before it becomes overloaded, anecdotal evidence suggests that many devices time out idle bindings after a certain amount of time. This test determines the existence of a static timeout similar to *UDP-1*, except that the client opens a TCP connection with the server. The connection is left idle with no TCP keepalives in use. TCP binding timeouts are often much longer than UDP ones. To speed up the test, the binary search technique therefore uses multiple parallel connections, and stops if binding timeouts are longer than 24 hours.

**TCP-2**: *TCP throughput.* To test if home gateways can limit TCP throughput, we measure the performance of a 100 MB bulk transfer. A first test measures client-to-server upload throughput, a second measures server-to-client download throughput, and a third measures the throughputs of simultaneous up- and downloads.

**TCP-3**: *Queuing and processing delay.* Anecdotal evidence suggests that many home gateways have over-dimensioned transmission buffers, which can add significantly to the end-to-end delay experienced by TCP. We measure this delay by embedding evenly spaced timestamps (every 2 KB) into the payload of the throughput tests in *TCP-2*. Delay is determined by the difference between the received timestamps and the local system clock. Clocks are synchronized using NTP and the tests are short enough that any error introduced by clock drift stays well below 1 msec. The output is normalized, so that the minimum difference is zero. The maximum delay is the median of the normalized differences, to prevent TCP retransmissions from skewing the results. Since the measured round-trip delay across each device is below 2 msec, it is safe to assume that the results are accurate to about 1 msec.

**TCP-4**: *Maximum number of TCP bindings.* The recommended behavior for a NAT is to retain a TCP binding for 124 min [11] in the absence of an explicit connection tear-down. When serving many connections, it is likely that a NAT reaches its maximum number of bindings. This test measures the maximum number of TCP bindings a NAT supports, by systematically creating connections to the same server port and periodically passing messages over each, to prevent binding timeouts. When a new connection fails to be created or messages can no longer be passed on an existing one, the maximum number of bindings has been reached.

### 3.2.3 Other Tests

**ICMP**: *TCP- and UDP-related ICMP forwarding.* Many NATs attempt to translate ICMP messages related to TCP and UDP bindings, because this can improve application performance. The most important ICMP messages to translate are *Destination Unreachable* messages indicating that fragmentation is needed, which TCP uses for PMTU discovery [22]. If they are not translated properly, PMTU "black hole" issues can occur [17]. For UDP, even detection of port reachability depends on ICMP messages. We test whether ICMP messages are correctly translated by hijacking packets coming from the NAT, generating ICMP messages of the desired kind that are sent back to the NAT, and inspecting packet traces to determine what the NAT actually did.

**SCTP** and **DCCP**: *Support for SCTP and DCCP.* Deployment of these transport protocols is said to be hindered by middleboxes that do not support them. We therefore determine the level of support for SCTP [30] and DCCP [15,16] among the home gateways in our testbed. For each of these transport protocols, we attempt to create

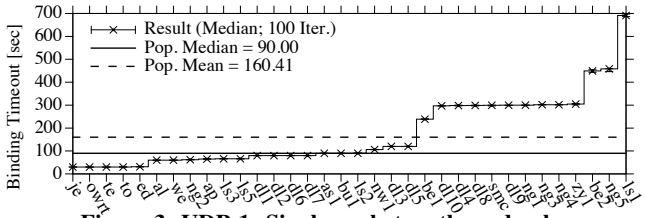


**Figure 3: UDP-1: Single packet, outbound only.**

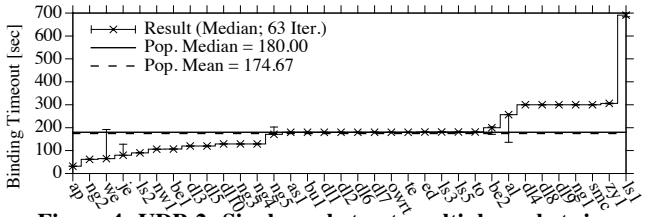


**Figure 4: UDP-2: Single packet out, multiple packets in.**

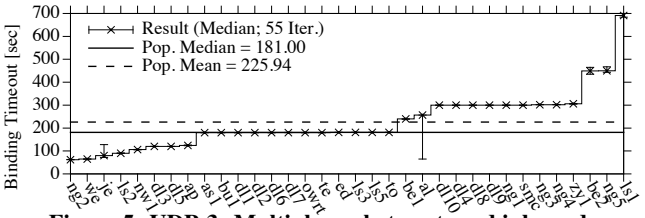


**Figure 5: UDP-3: Multiple packets out- and inbound.**

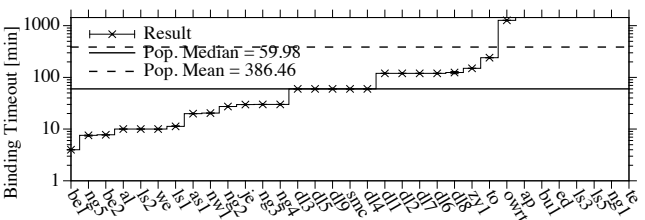


**Figure 7: TCP-1: TCP binding timeouts.**

a single connection and exchange data. If this succeeds, a home gateway supports the respective transport.

**DNS**: *DNS over TCP.* The DNS proxy on each NAT is tested for support of DNS-over-TCP by querying it using *dig* from the Berkeley Internet Name Daemon (BIND) suite.

# 4. EXPERIMENTAL RESULTS

This section presents the measurement results for the experiments described in Section 3. All plots show the measured results across the entire population of studied home gateways, arranged on the *x*-axis by increasing value. Each data point shown is the median of many repetitions of a measurement, as indicated in the plot legend. Quartiles for each data point are plotted as error bars, but the inter-quartile gap is usually too narrow for them to become visible.

## 4.1 UDP Timeout Results

Section 3.2.1 described a measurement method for determining UDP binding timeouts as well as several binding usage scenarios. This section presents the experimental results obtained from applying this method in these different scenarios. From Figure 2, it is apparent that home gateways do not behave consistently in the different UDP tests. While it is clear that many devices do not vary their timeout behavior between tests *UDP-2* and *UDP-3*, the timeouts do vary with manufacturers and firmware versions. Most devices retain UDP bindings for the 120 sec required in [3], at least while there is inbound traffic over the binding. *UDP-1* presents a more unusual case, where the binding is often removed much sooner. The next paragraphs discuss the detailed results.

**UDP-1**: Figure 3 plots the medians of measured binding timeouts for the *UDP-1* case, where the client sends only a single packet to the server. Quartiles are plotted as error bars but the inter-quartile range is too narrow to become visible, indicating stable results for all devices. One obvious result is that UDP binding timeouts vary by an order of magnitude across the measured set of home gateways. The *je* device is among those with the shortest timeout (30 sec), whereas *ls1* has a timeout that is more than twenty times longer (691 sec). The median timeout across the entire set is 90 sec, the mean is 160 sec. Note that more than half of the tested devices do not conform to the IETF specification [3] that requires timeouts of more than 120 sec; and only a single device (*ls1*) complies with the longer 600 sec timeout that the IETF recommends.

**UDP-2**: Figure 4 shows the measurements for the case where the server sends a stream of response packets to the client. The intent of this measurement was to determine whether inbound traffic affects the binding timeout, and for many devices, it does. Most devices here tend to use longer timeouts; the minimum is now 54 sec, the median is 180 sec and the mean 175 sec. For example, *ed, owrt, to,* and *te*, which shared the shortest timeout (30 sec) in the *UDP-1* case, now use a median timeout of 180 sec. Other devices, however, shorten their timeouts now, *e.g.*, *be2*, which had a timeout of ca. 450 sec previously, now reduces its timeout to ca. 202 sec. The inter-quartile range for *we* and *al* as well as, to a lesser degree, *je* and *ng5* is substantial, because these boxes seem to use very coarse-grained binding timers.

**UDP-3**: Figure 5 plots the measurements for the case where a received server response triggers the client to send another packet. The intent of this test is to check whether outbound traffic on a binding affects the timeout. The difference to *UDP-2* is less pronounced; the median timeout remains almost unchanged, although the mean increases to 226 sec. This is mostly due to a few devices (*be1*, *dl10*, *ng3 ng4*, and esp. *be2* and *ng5*) lengthening their timeouts as outbound packets are now also present, reaching the same level as in the *UDP-1* test; no devices shorten them.

**UDP-4**: The results show that different behaviors exist for how NATs choose external port numbers and how they reuse port pairs. Most of the devices (27 out of 34) prefer to use the original source port as the external port for a binding. 23 of these devices seem to also reuse an expired binding, while 4 devices create a new binding. 7 devices do not attempt to use the original source port and seem to always create a new binding after an old one expired.

**UDP-5**: Figure 6 shows the median measured binding timeouts to different well-known server ports. The results indicate that most devices use a timeout scheme that is independent of the server port. Notable exception is *dl8*, which uses a shorter timeout for the DNS port.

## 4.2 TCP Results

This section discusses the results of the TCP tests described in Section 3.2.2.

**TCP-1**: Figure 7 shows the measured TCP binding timeouts. Because the measured timeouts are highly variable, the plot uses a log scale to highlight the differences. *be1* has the shortest timeout; it consistently times out TCP bindings after 239 sec – less than 4 min. More than half the devices fail to meet the IETF recommended timeout of 124 min [11]. Some of the NATs retain TCP bindings for considerably longer – the seven devices on the right in Figure 7 still had not timed out their bindings after 24 h (1,440 min), which was the cutoff for this test.

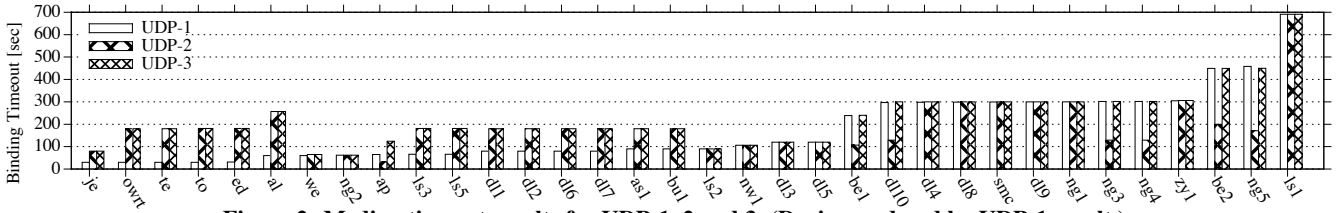


**Figure 2: Median timeout results for UDP-1, 2 and 3. (Devices ordered by UDP-1 result.)**

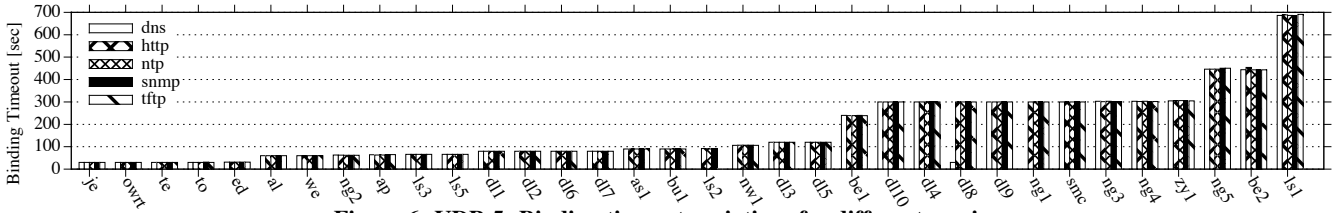


**Figure 6: UDP-5: Binding timeout variations for different services.**

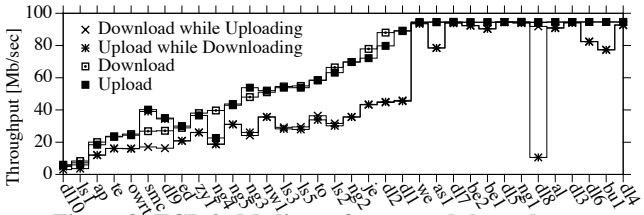


**Figure 8: TCP-2: Medians of measured throughputs.**

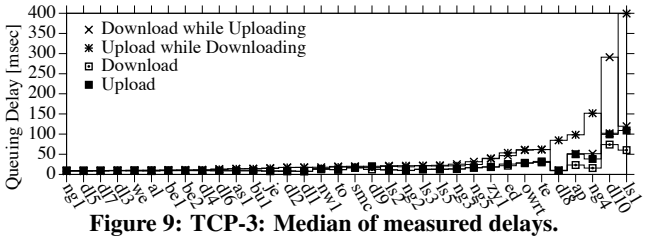


**Figure 9: TCP-3: Median of measured delays.**

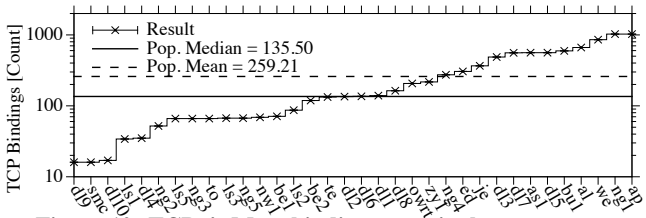


**Figure 10: TCP-4: Max. bindings to a single server port.**

**TCP-2**: Figure 8 shows the medians of the measured TCP throughputs for each device. Thirteen devices can sustain the maximum possible throughput of 100 Mb/sec for uploads and downloads, but other devices fail to do so: The median throughput across the device set is roughly 59 Mb/sec for unidirectional uploads and downloads.

For the bidirectional tests, the devices that achieve the best performance in the unidirectional tests continue to demonstrate higher throughputs, though not all reach 100 Mb/sec in both directions. There is a marked distinction in the performance of some devices: many are incapable of handling more than 50 Mb/sec of traffic in either direction. Further, the median in the bidirectional case is ca. 35 Mb/sec, which is much lower than the 68 Mb/sec in the unidirectional case. There are extremely poor performers in this test, *dl10* and *ls1* being the worst performers *dl10* and *ls1* can only sustain unidirectional throughput of around 6 Mb/sec and 8 Mb/sec respectively for download and 6 Mb/sec each for upload. Some other devices also demonstrate a distinct difference in upload and download throughputs: *smc*, for example, can sustain 41 Mb/sec for upload, but only 27 Mb/sec for download.

The poor throughput of many devices may normally go unnoticed: the WAN port will generally connect to the user's ISP over a link with a much lower capacity than 100 Mb/sec.

**TCP-3**: Figure 9 shows the results for queuing and processing delays introduced by the devices. Perhaps not surprisingly, the devices that perform well in the throughput tests (*TCP-2*) also perform well in the latency test. Bidirectional traffic increases latencies slightly for most of the devices, more significantly so for the poorest performers, which are *ls1* and *dl10*. The median latency for *dl10* is 74 msec when downloading only but jumps to 291 msec when uploading at the same time, whereas the median latency for *ls1* is 110 msec when uploading only but reaches 400 msec when downloading at the same time. Even the best-performing boxes see minor delay increases of ca. 2 msec.

**TCP-4**: Figure 10 shows the maximum number of bindings a NAT allows to a single server port. *dl9* and *smc* support only 16 bindings, whereas *ng1* and *ap* allow ca. 1024 bindings; the median is 135.

## 4.3 Other Results

Table 2 shows pass/fail results for the tests described in Section 3.2.3 related to SCTP and DCCP support, DNS-over-TCP support, and ICMP handling.

**ICMP**: The "TCP" and "UDP" columns in Table 2 indicate which home gateways correctly translate various ICMP messages related to flows of the respective transport protocol. *nw1* does not translate *any* transport-related ICMP messages; all others translate at least "Port Unreachable" and "TTL Exceeded". *ls2* translates all TCP-related ICMP messages into (invalid) TCP resets. About half of the devices (16 out of 34) do not correctly translate transport headers contained in ICMP payloads, and *zy1* and *ls1* do not correctly translate IP checksums in ICMP payloads.

**SCTP** and **DCCP**: It is possible to establish an SCTP connection through 18 of the 34 devices – an astounding result, since the general belief has been that even single-homed SCTP connections do not usually work across NATs. None of the devices allowed establishing a DCCP connection. *dl4*, *dl9*, *dl10* and *ls1* pass SCTP and DCCP packets entirely untranslated, 20 others attempt to simply translate the IP source address. Among those 20 devices are all those that work with SCTP – which raises the question if they actually fully support SCTP, or whether a single SCTP connection

| Tag | DCCP: Comm. | DNS over TCP | DNS over UDP | ICMP: Host Unreach. | SCTP: Comm. | TCP: Reass. Time. Ex. | TCP: Frag. Needed | TCP: Param. Prob. | TCP: Src. Route Fail. | TCP: Source Quench | TCP: TTL Exceeded | TCP: Host Unreach. | TCP: Net Unreach. | TCP: Port Unreach. | TCP: Proto. Unreach. | UDP: Reass. Time Ex. | UDP: Frag. Needed | UDP: Param. Prob. | UDP: Src. Route Fail | UDP: Source Quench | UDP: TTL Exceeded | UDP: Host Unreach. | UDP: Net Unreach. | UDP: Port Unreach. | UDP: Proto. Unreach. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *al* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ap* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *as1* | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *be1* | | | • | | | | | | | | | | | | | • | • | | • | | • | • | • | • | • |
| *be2* | | | • | | | | | | | | | | | | | • | • | | • | | • | • | • | • | • |
| *bu1* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl1* | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl10* | | | • | | | | | | | | • | | | • | | | | | | | • | | | • | |
| *dl2* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl3* | | | • | • | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl4* | | | • | | | | | | | | • | | | • | | | | | | | • | | | • | |
| *dl5* | | | • | • | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl6* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl7* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl8* | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *dl9* | | | • | | | | | | | | • | | | • | | | | | | | • | | | • | |
| *ed* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *je* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ls1* | | | • | • | | | • | | | | | | | | | • | • | • | • | • | • | • | • | • | • |
| *ls2* | | | • | • | | | | | | | | | | | | • | • | | • | • | • | • | • | • | • |
| *ls3* | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ls5* | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ng1* | | | • | • | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ng2* | | | • | • | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ng3* | | • | • | • | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ng4* | | • | • | • | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *ng5* | | | • | • | | | | | | | | | | | | • | | | • | | • | • | • | • | • |
| *nw1* | | | • | | | | | | | | | | | | | | | | | | | | | | |
| *owrt* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *smc* | | | • | | | | | | | | • | | | • | | | | | | | • | | | • | |
| *te* | | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *to* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *we* | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |
| *zy1* | | | • | • | | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • | • |

**Table 2: Summary of the results of other tests.**

works because only IP headers are translated (the SCTP checksum does not cover the network-layer pseudo header). Further experiments are needed to better characterize the nature of SCTP support in these devices, for example, whether SCTP multihoming works over those devices that do support a single-homed connection.

**DNS**: 14 of the tested devices accept connections on TCP port 53 (DNS), and 10 of them accept and respond to DNS queries on that TCP port; this roughly agrees with [9]. *ap* forwards DNS queries arriving via TCP over UDP, the others forward over TCP.

## 4.4 Observations and Discussion

The results in Section 4.1 indicate that although UDP binding timeouts are relatively low for bindings that see little use (*UDP-1*), bindings that see some bidirectional traffic (*UDP-2*) and especially bindings that see repeated bidirectional traffic (*UDP-3*) are being granted longer timeouts. (Although no device uses the IETF recommended 600 sec.) In this light, UDP keepalive intervals as short as 15 sec, which are used by some applications, are perhaps overly aggressive: the lowest measured timeout when a binding has seen bidirectional traffic is 54 sec. It is clear that there are a variety of different behaviors among the tested devices, and that the timeout applied to a binding depends more on what traffic pattern is seen than on what port numbers are used.

In Section 4.2, the lowest measured number of parallel TCP bindings to a single server port is 16, and even the "best" devices imposed a limit of ca. 1024, which is much smaller than the 16-bit port number space. A low number of permitted parallel bindings can interfere with important applications, such as web browsing: modern browsers impose a limit on the number of concurrent connections to a DNS name, which has caused content providers to use multiple DNS names for a web server with one IP address. The intent of this practice is to circumvent this browser limit in an attempt to improve performance.

TCP binding timeouts vary much more widely than UDP timeouts. Half the devices time out TCP bindings after less than 1 h, which is much less than the IETF-recommended minimum of 124 min. This also means that TCP stacks that implement the standardized minimum TCP keepalive interval of 2 h [6] will not be able to reliably refresh TCP connections in many cases.

In addition to the specific results reported in this paper, our experiments also exposed a few other interesting behaviors: First, some devices use the same Ethernet MAC address for both their WAN and LAN ports. In a typical configuration, this should make no difference, but it required us to connect the WAN and LAN ports to physically different VLAN switches. Second, some devices do not decrement the IP time-to-live (TTL) field and few honor a "Record Route" IP option, which can interfere with network diagnostics and other uses of the TTL field. Third, it was unexpected to find that some boxes fall back to only translating the IP header when they encounter an unknown transport protocol.

Discovering that this "fallback" behavior is not uncommon raises an interesting question for future transport protocol design. Would it be possible to leverage this behavior, by avoiding dependencies on the IP layer (such as the pseudo-headers for checksum calculation) in order to improve the chances for un-encapsulated and un-relayed NAT traversal?

Finally, it is worth noting that no single home gateway consistently performs better than others across all tests, which makes it difficult for application designers to target a "better" subset of devices.

# 5. CONCLUSIONS AND FUTURE WORK

The results presented in this paper are a snapshot of an ongoing study, but already present some interesting variations in home gateway behavior. The results cover 34 different home gateway models, but do not necessarily reflect how frequent the observed behaviors are in the deployed device base. We are planning to expand the range of tests to investigate handling of TCP and IP options, to measure the rate at which NATs are capable of creating new bindings, more extensive DNSSEC, queuing and SCTP tests, measuring the success rates of STUN [27], TURN [18] and ICE [26], investigate the support for ECN [24], IPv6 [8] and various IP and TCP options, and more.

We are also continuing to expand the range of home gateways in the testbed. All home gateways measured in this paper use an Ethernet interface for their uplink. We plan to extend this study to include already-donated DSL and cable modems, once a DSLAM (DSL access multiplexer) and a CMTS (cable modem termination system), which also have been donated after preliminary versions of this study were presented, have been added to the testbed.

# 6. ACKNOWLEDGMENTS

Many of the home gateway devices were donated to the testbed, and we would like to sincerely thank all donors. The results presented in this paper are a snapshot of a study that is continuously being extended. Please contact the authors if you have suggestions for additional or improved tests, or if you can donate a spare, unused home gateway to the testbed – we will cover shipping.

L. Eggert is in part funded by *Trilogy*, a research project supported by the European Commission under its 7th Framework Program.

## 7. REFERENCES


[1] J. Åhlund and P. Wallström. DNSSEC Tests of Consumer Broadband Routers. Technical Report, .SE Internet Infrastructure Foundation, Feb. 2008.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033 (Proposed Standard), Mar. 2005.

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

[4] R. Bellis. DNS Proxy Implementation Guidelines. RFC 5625 (Best Current Practice), Aug. 2009.

[5] R. Bellis and L. Phifer. Test Report: DNSSEC Impact on Broadband Routers and Firewalls. Technical Report, Nominet, Sept. 2008.

[6] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), Oct. 1989.

[7] L. D'Acunto, J. Pouwelse, and H. Sips. A Measurement of NAT & Firewall Characteristics in Peer to Peer Systems. In *Proc. ASCI Conference*, 2009.

[8] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), Dec. 1998.

[9] T. Dietrich. DNSSEC Support by Home Routers in Germany. In *Proc. 60th Réseaux IP Européens (RIPE) Meeting*, May 2010.

[10] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-Peer Communication Across Network Address Translators. In *Proc. USENIX Annual Technical Conference*, pages 13–13, 2005.

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

[12] S. Guha and P. Francis. Characterization and Measurement of TCP Traversal through NATs and Firewalls. In *Proc. ACM SIGCOMM IMC*, pages 199–211, 2005.

[13] H. Haverinen, J. Siren, and P. Eronen. Energy Consumption of Always-On Applications in WCDMA Networks. In *Proc. IEEE Vehicular Technology Conference*, pages 964–968, Apr. 2007.

[14] C. Jennings. NAT Classification Test Results. Internet-Draft draft-jennings-behave-test-results-04, Internet Engineering Task Force, July 2007. Work in Progress.

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

[16] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion Control Without Reliability. *ACM SIGCOMM CCR*, 36(4):27–38, 2006.

[17] K. Lahey. TCP Problems with Path MTU Discovery. RFC 2923 (Informational), Sept. 2000.

[18] R. Mahy, P. Matthews, and J. Rosenberg. Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN). RFC 5766 (Proposed Standard), Apr. 2010.

[19] L. Mäkinen and J. Nurminen. Measurements on the Feasibility of TCP NAT Traversal in Cellular Networks. In *Proc. Conference on Next Generation Internet Networks*, pages 261–267, 2008.

[20] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP Selective Acknowledgment Options. RFC 2018 (Proposed Standard), Oct. 1996.

[21] A. Medina, M. Allman, and S. Floyd. Measuring the Evolution of Transport Protocols in the Internet. *ACM SIGCOMM CCR*, 35(2):37–52, 2005.

[22] J. Mogul and S. Deering. Path MTU Discovery. RFC 1191 (Draft Standard), Nov. 1990.

[23] J. Postel. Internet Control Message Protocol. RFC 792 (Standard), Sept. 1981.

[24] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168 (Proposed Standard), Sept. 2001.

[25] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), Feb. 1996.

[26] J. Rosenberg. Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols. RFC 5245 (Proposed Standard), Apr. 2010.

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

[28] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489 (Proposed Standard), Mar. 2003.

[29] P. Srisuresh, B. Ford, S. Sivakumar, and S. Guha. NAT Behavioral Requirements for ICMP. RFC 5508 (Best Current Practice), Apr. 2009.

[30] R. Stewart. Stream Control Transmission Protocol. RFC 4960 (Proposed Standard), Sept. 2007.