A Queueing Theoretic Analysis of Source IP NAT

Cedric Westphal* , Charles E. Perkins†

*DOCOMO Labs USA, †WiChorus Inc
Email: *cwestphal@docomolabs-usa.com, †charliep@computer.org

***Abstract*—The number of devices connected to the Internet has outstripped the number of effectively assignable IPv4 addresses. In order to be globally reachable, many devices must share the same IPv4 address; current mechanisms only provide reachability when the device sharing the IPv4 address itself initiates communication. We describe a mechanism to make nodes behind a NAT globally reachable, even when communications are initiated from the global Internet. The intended application of the mechanism, denoted SIPNAT [1], is to allow for the first time bidirectional global reachability of IPv6 addresses by nodes in the global IPv4 Internet, in a scalable manner, thus resolving the major issue associated with IPv4-IPv6 translation.**

SIPNAT involves filtering flows at the gateway between the IPv4 and IPv6 domain through a combination of DNS request and timing information for the IPv4 initiated connection (the IPv6 to IPv4 connections are performed using typical NAT mechanisms, where the IPv6 domain takes the role of the private address space). We study the performance of the SIPNAT mechanism using queuing theoretic analysis, and show that our SIPNAT model is accurate on actual data traces.

I. Introduction

In order for network nodes with IPv6 addresses to maintain communications with the existing IPv4-based Internet, the IPv6 devices need the assistance of a Network Address Translator (NAT) [2], [3]. NATs have up until now successfully resolved the increasing pressures for address space allocation as more and more devices attach to the Internet. These NATs offer connectivity to the global Internet for devices in customer premises (i.e. customer premises equipment, CPE) that only have private IP addresses, and thus are not themselves individually addressable from the Internet. Typically, the NAT box sets up flow parameters so that communications from the premises device with the private address can be translated for proper delivery to a destination in the global Internet.

From the point of view of the external Internet, the CPE devices all appear to have an address belonging to the NAT box itself. This translation is thus one-to-many, and the demultiplexing almost always makes use of the port numbers in the UDP or TCP headers of the packets[1].

Since the port number is established by packets arriving from the CPE, traffic across the NAT can exclusively be initiated by the CPE device. For this reason, servers and websites in today's Internet are typically not hosted on nodes relying on NAT for their connectivity to the Internet. Such servers and hosts require allocation of static, globally routable IP addresses, which are becoming increasingly scarce and expensive in today's Internet.

[1]Almost all Internet traffic uses UDP or TCP, but there are protocols that do not use port numbers and thus are unable to connect network node endpoints on opposite sides of the NAT box.

The situation for IPv6-addressable CPE is very similar. Domains of IPv6-only devices can use the same methods for address translation as are already in use today between the global IPv4 Internet and CPE with private IPv6 addresses. IPv6 applications using existing NAT interface routers are, typically, similarly constrained by the need for the IPv6 CPE to initiate all communications with existing IPv4 network nodes.

All-the-time global reachability is a must for businesses, and it is infeasible to rely on a communications paradigm wherein the business service node initiates contact. Up until now there have not been any scalable solutions that allow for IPv4→IPv6 translation, which is what would be needed to allow the IPv6 destination (the CPE belonging to the business) to receive incoming communications from the legacy IPv4 devices of its existing customers.

This paper describes Source IP NAT (SIPNAT [1]), which has been proposed to provide this scalable IPv4→IPv6 translation. Using SIPNAT, the NAT demultiplexes flows based on a combination of source IP address and NAT interface. It is a mechanism which allows the NAT to create a mapping for a connection request initiated by an IPv4 external address to an IPv6 CPE server, without any changes to either device.

In this paper, we focus on SIPNAT in the context of IPv4→IPv6 translation; but it can also provide global IP reachability to nodes behind a NAT box using private IPv4 addresses as well. The mechanism and its analysis are applicable in the broader context. The key advantages of using IPv6 rather than a private address are the inherent global reachability over the growing IPv6 Internet (in addition to the global reachability with the IPv4 Internet through SIPNAT), the huge address space, and the other associated benefits of the improved IPv6 design.

This paper also presents a queueing theoretical analysis of the SIPNAT mechanism. We highlight its fundamental properties and demonstrate its scalability.

Related work: There are two main strands of work related to the mechanism described in this paper: one focuses on the transition mechanisms for IPv6, while the other deals with providing global connectivity to nodes behind a NAT.

For the first strand, the main mechanisms are either embedding the IPv4 address into the IPv6 address (where for instance the IPv4 address 1.1.1.1 maps to the IPv6 2002:0101:0101::/48) [4], [5], of using tunneling mechanisms (for instance, [6]). These mechanisms have been standardized for several years, yet their deployment remains relatively limited. IVI[7] is a proposal which, similar to SIPNAT, enables IPv4 reachability to IPv6 nodes by way of network address translation. IVI requires the assignment of a static IPv4 address on the NAT box for any particular desired IPv6 destination, and each such IPv4 address can only be used by that single IPv6 destination. This approach is thus inherently unscalable. SIPNAT may be viewed as a method for dynamically sharing each such IVI address among many potential IPv6 destinations.

As for reaching nodes behind a NAT, there are standardized solutions, such as STUN [8] and proprietary solutions, such as peer-to-peer mechanisms or Skype [9]. However, these mechanisms require the use of globally reachable servers to act as rendez-vous points, and require the utilization of specific applications and configurations to set up the connection. Unlike SIPNAT, these are not IP level mechanisms to provide global reachability.

Organization: The paper is organized as follows. First, in the next section, we briefly highlight the existing IPv4-v6 transition mechanisms and the related work to our proposal. In section II, we describe the basic SIPNAT principles. Section III depicts our queuing theoretic model, and section IV presents the analytical results. Section V shows the performance evaluation, and section VI concludes the paper.

II. SIPNAT

In this section, we recall the basic SIPNAT mechanism. For further details, we refer the reader to [1].

We assume an external, global IPv4 network, and an IPv6 domain, connected to the IPv4 Internet through a gateway, denoted the *SIPNAT box*. We use the terms external for the global IPv4 Internet, and internal for the IPv6 domain. An (external) IPv4 client wishing to connect to an (internal) IPv6 server only needs to know the destination's fully qualified domain name (FQDN). It attempts to resolve the name into an address by sending a request to its DNS server. This request is relayed to the DNS serving the domain of the FQDN. This DNS server for the internal domain is controlled by the administrator of the domain and configured in a way to prevent other DNS servers to cache the mapping corresponding to the FQDN. The DNS reply also instructs the IPv4 client to cache this entry for the minimum amount of time (typically one second, even though this varies for different implementations). Practically, this means that all DNS requests for hosts in the IPv6 domain will be redirected and eventually handled by the same DNS server, which in typical operation will be collocated with the SIPNAT gateway.

Upon receiving a DNS request, the SIPNAT gateway will resolve the FQDN into an IPv6 address $Hv6$. The gateway then supplies a DNS Reply containing an A record with an IPv4 address, $GWv4_i$, which is assigned to its IPv4 interface to the global Internet. The gateway has K such IPv4 interface addresses, and i, $1 \leq i \leq K$, denotes the index for one of these interfaces. The SIPNAT gateway associates the requested IPv6 host $Hv6$ to this IPv4 address $GWv4_i$, so that, for some

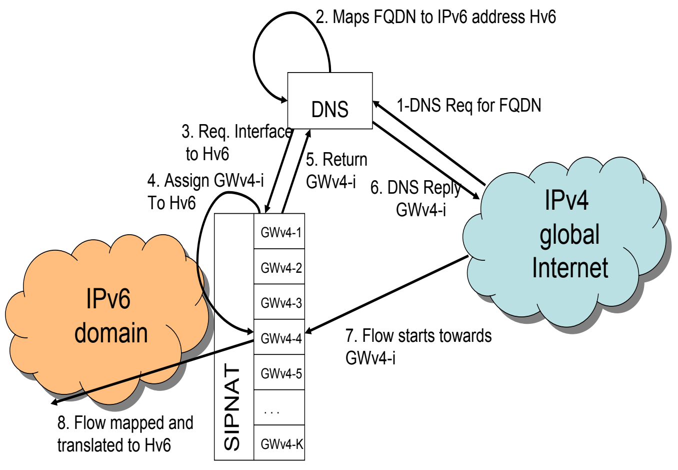

Fig. 1. SIPNAT gateway interfaces and signaling.

period of time τ, any packet from a "new" flow arriving at $GWv4_i$ will be forwarded to $Hv6$. A packet is designated to belong to a "new" flow if it does not conform to the flow parameters for a previously assigned flow between an IPv4 source and IPv6 destination via $GWv4_i$. The period τ denotes the *bind time* for the interface $GWv4_i$ and the $Hv6$ address. The bind time should be thought of as the length of time that a packet is enabled to complete a new binding between the source and destination.

Upon receiving the DNS reply, the IPv4 client $Sv4$ starts its transaction with the IPv4 address $GWv4_i$. When the gateway receives a packet at $GWv4_i$, it looks up the IP address of the source, $Sv4$. If found, then the packet is considered to be part of an existing flow. Otherwise, SIPNAT looks again to find if there is a pending flow assignment; there can be at most one such pending assignment. If it exists, SIPNET uses $Sv4$ to establish the pending assignment, along with the destination and associated translation parameters (possibly including QoS) for the flow. $GWv4_i$ is translated to $Hv6$. $Sv4$ is translated to an IPv6 address by prepending a well-known prefix (e.g. ffff:0:0/96) to $Sv4$ – for instance, as indicated, to create the *IPv4-mapped IPv6 address*. Depending on configuration, another 96-bit IPv6 prefix can be used instead, in a similar fashion. In general, SIPNAT can include port information as part of the mapping to $Hv6$, and even add some further protocol identification. This depends on the hardware capability of the gateway. To keep this exposition and analysis generic, we just denote the mapping as between a source and a host, and understand that the definition of *source* for flow management purposes can include more than just the IP address.

Once the mapping is assigned, further packets from this source $Sv4$ to the gateway address $GWv4_i$ are translated and forwarded to $Hv6$ just as with traditional NAT, except that the translation parameters for packets received by the NAT now depend on the IP address and not the port number.

Any IPv4 source can thus be mapped to the corresponding

IPv6 with which it wishes to communicate, and multiple IPv6 addresses can be mapped to the same IPv4 address on the SIPNAT box. However, the rate at which such mappings happen is limited by the DNS mechanism and the requirement that there may be only one pending flow assignment per SIPNAT IPv4 address. Upon receiving the DNS request, the gateway associates an IPv4 address $GWv4_i$ with $Hv6$ until the expiration of the bind time-out value τ. Since it possesses only a limited range of addresses $GWv4_i, 1 \leq i \leq K$, the rate of new flows that can be supported is limited by this allocation mechanism.

It is bounded in two manners: the first constraint is how fast the system can allocate such mappings, since any new flow will block an interface from any other new flow, for a period of time τ. The second constraint is how many flows from the same source can be handled by the SIPNAT gateway. For a given interface, if source $Sv4$ has an existing flow going through $GWv4_i$, the SIPNAT filters would match any new flow from $Sv4$ as identical to $Sv4$'s existing flow. Packets of $Sv4$'s new flow will be forwarded to the IPv6 address assigned to the existing flow, and the new flow would fail.

These two constraints need to be analyzed in detail to assess the viability of the SIPNAT proposal, and to properly dimension a SIPNAT box (for instance, with respect to the number K of interfaces) to function as intended.

We only described the setting up of flows from the external IPv4 network to the IPv6 network, as we are interested in global reachability. The other direction, from IPv6 to the external Internet, is handled according to any traditional NAT mechanism. Security issues are handled in [1].

In the next section, we define a mathematical model to compute the fundamental properties of SIPNAT, which we use to estimate the number of IPv6 addresses that can be used, namely how many IPv6 addresses can be mapped to a single IPv4 address. The higher this multiplicative factor, the more scalable the scheme.

III. Model

We model the gateway/DNS mechanism described in the previous section using a queueing theoretic approach.

We assume that there are N_6 IPv6 addresses recognized by the DNS server[2], and that the gateway possesses K interfaces (with the corresponding IPv4 addresses) $GWv4_1, \dots, GWv4_K$.

We assume that each IPv6 address is requested according to a Poisson process with rate λ. The Poisson arrival time is consistent with the actual flow arrival process being the superposition of a large number of independent individual flows, as the combined inter-arrival times for such processes converges to Poisson (see for instance Chapter 2.6 in [10]). The total rate of request arriving at the gateway is thus λN_6. This Poisson process could be modulated according to time-of-day or day-of-week variation patterns, but since the performance is monotonous with the arrival rate, we can consider the maximum Poisson arrival rate that the system will encounter to dimension it properly.

We assume furthermore that the DNS session times out after a time θ at the source, namely if the source does not receive a reply to its DNS request within time θ, then the DNS session is lost. We also assume that θ is computed so as to exclude the RTT as a first approximation, so that θ corresponds to the amount of time the gateway has to return an IPv4 address $GWv4_j$ to the source. θ is a constant for now, but could be modeled as a random variable with a specific distribution if such level of detail was necessary.

After processing the DNS request, the gateway opens an address $GWv4_i$ for a binding time-out period of τ seconds, during which it will map the packets arriving at $GWv4_i$ to the address $Hv6$. This is a one-to-one mapping, meaning that all (non-already recognized) packets to $GWv4_i$ can only go to $Hv6$. Thus during one time window τ, only one new flow can be established through the interface $GWv4_i$.

If no interface is available to open a binding window (for instance, all interfaces are already waiting for a new flow packet to arrive), the mapping request is buffered, and a timer is started to assess how long the request has been in the system. The assignment of an address is made only if the request has been in the system within the time θ, as beyond this limit, the DNS response will be too late and ignored, and allocating resources (namely an interface $GW4_i$ reserved for a period of time τ) would lead to a waste.

We define by W the waiting time of the arrival packet before it is assigned an interface. New flows are processed only if, when they are ready to be assigned an interface, $W < \theta$, so that the DNS request has not timed out at the original client when it eventually receives its DNS reply.

We model our system as a queueing system where customers (here, requests for IPv6 hosts) arrive at rate $N_6\lambda$ and are served for a deterministic period of time τ. K customers can be served in parallel, and customers are impatient: if they wait more than θ before their service starts, then they leave the system. In queueing theoretic terms, this is an M/D/K system with impatient customers. Such M/D/K system with impatience was analyzed, for instance in [11].

IV. Analytical Results

We are interested in knowing the fundamental properties of the system described in the previous section. One quantity of interest, for instance, is the relationship between K and N_6.

One obvious bound is the rate of allocation: since each new flow will reserve one of the K interfaces for τ units of time, the system can serve new requests at a rate of at most K/τ per unit of time. This can be stated as:

Theorem 4.1: A necessary condition on the arrival rate of customers in the system is:

$$N_6\lambda < \frac{K}{\tau} \tag{1}$$

Technically, for any rate under this upper bound, the queueing system would be stable: it would process all the arrivals

[2]We assume it to be co-located to the SIPNAT gateway, even though the two functions could reside on separate hosts.

and reach a steady state distribution with finite waiting time. However, since we are dealing with impatient customers, we can in practice increase the rate of arrivals only such that the probability of customers leaving because their DNS timer has expired stays small, that is, bounded above by some (tunable) parameter $\epsilon > 0$.

We assume now that $\lambda < K/N_6\tau$. We attempt to see how close from this bound we can operate the system, while keeping the probability of dropping a request below some fixed ϵ.

We recall the formula from [11], which is a close approximation of the system we consider. Assume that W follows the steady-state distribution of the waiting time in the queue for the M/D/K customer without impatient customers. Denote by $\rho = K/\tau N_6\lambda$. We assume $\rho < 1$.

Theorem 4.2: The probability that a customer leaves the system before being served is:

$$P_{loss} = \frac{(1-\rho)P(W>\theta)}{1-\rho P(W>\theta)} \quad (2)$$

The distribution W is given, for instance, in [12] or [13]. From this, we can deduce the rate λN_6 which satisfies the bound $P_{loss} < \epsilon$.

While the distribution of W is given, it is difficult to numerically compute for large values of K. It also depends on the specific allocation of the interfaces to the flow arrival. We resort thus to some special cases and some approximations.

A. One interface SIPNAT

The simplest implementation of a SIPNAT box would have one IPv6 and one external interfaces. Since the cost of a SIPNAT platform would grow with the number of interfaces, it is interesting to consider the minimal configuration as a baseline. (Note that, since several IPv4 addresses can be multiplexed over the same interface, the cost might not consist of additional line-cards, but of additional global, public IPv4 addresses.)

Theorem 4.3: For a system with one interface, $P(W > \theta)$ can be computed as:

$$P(W>\theta) = 1-(1-\rho)\sum_{j=0}^{\Theta}\frac{(\rho(j-\theta))^j}{j!}e^{-\rho(j-\theta)} \quad (3)$$

where $\rho = N_6\lambda\tau$ and $\Theta = \lfloor\theta\rfloor$, i.e. the largest integer less than or equal to θ.

Proof: This is a straightforward derivation of classical queuing theory. A numerical method to solve the equation can be found in [14] ∎

Combined with Equation 2, we can thus obtain the probability of packet loss as a function of N_6, τ, and λ.

B. Multiple interfaces

We now consider that $K > 1$ and study the allocation policy, namely how the SIPNAT box selects the interface to map to the requested IPv6 address. In the first considered policy, the assignment is made randomly, using for instance a hash of the FQDN requested by the IPv4 source. In the second policy, the assignment is made according to a round robin policy, where the interfaces are selected sequentially, modulo K.

Hash-based interface assignment

We assume that the SIPNAT box picks the interface $GWv4_i$ randomly according to a uniform distribution over $\{1,\ldots,K\}$. This models a system were the interface is assigned based upon a hash of the IPv6 destination (where the IPv6 population is large enough and the requests evenly distributed over the whole population).

Theorem 4.4: P_{loss} and $P(W > \theta)$ can be computed as in Theorem 4.3 with rho replaced by $\rho' = N_6\lambda\tau/K$.

Proof: Hashing functions as a random thinning of the Poisson process arrival. With probability i/K, the next flow is assigned to interface $GWv4_i$. Thus $GWv4_i$ receives incoming flows with a Poisson process inter-arrival with rate $N_6\lambda/K$. ∎

A random assignment of the interface to the new flow requests ensures that the K interfaces act as K independent single interface SIPNAT with the incoming rate scaled down by a factor K. One needs to set the parameters so that the probability of a loss for each one of these SIPNAT is less than ϵ.

Round-robin interface assignment

Another popular policy is to allocate the interfaces to the new flow sequentially. Note that, since the service time is deterministic, this means that the allocation policy is *work preserving*, namely that if the chosen interface is non-empty, then there exist no empty interface which is empty which could have processed the request faster. Since the random allocation does not necessarily have this property, the round-robin interface must be more efficient.

In this set-up, the inter-arrival process at interface $GWv4_i$ is not Poisson anymore, but two flow requests are separated by K exponential random inter-arrival times, and the inter-arrival process at interface $GWv4_i$ is thus Erlang-K distributed. There is no expression for the waiting time distribution of an $Er(K)/D/1$ system (using Kendall's notation). However, we can consider the convergence as K grows large.

Theorem 4.5: For a round-robin SIPNAT system with K interfaces, if the arrival rate $\lambda(K)$ increases proportionally to K so that $\rho = N_6\lambda(K)\tau/K$ stays constant, then the waiting time in the system decreases to 0, and the probability of losing a new customers decays to zero as well as $K \to \infty$.

Proof: The inter-arrival rate to interface $GWv4_i$ is equal to the sum of K independent random variables with identical exponential distribution with rate $N_6\lambda K$ (writing $\lambda(K) = \lambda K$ since the relation is of proportionality). Denote by $X_j, 1 \le j \le K$ a sequence of i.i.d. exponentially distributed random variable with parameter $N_6\lambda(K)$. Then the inter-arrival time $A(K)$ at $GWv4_i$ is equal in distribution to $\sum_{j=1}^{K} X_j$. A straight-forward application of the central limit theorem states that $A(K)$ converges in distribution to:

$$KE(X) + \sqrt{K}\sigma_X N(0,1) \quad (4)$$

where $E(X)$ is the mean of all X_j random variables, and σ_X its standard deviation. Since $E(X) = 1/(N_6\lambda(K))$ and $\sigma_X = 1/(N_6\lambda(K))$, $A(K) = \frac{K}{N_6\lambda(K)}(1 + 1/\sqrt{K}N(0,1))$.

Taking, $\lambda(K) = \lambda K$, $A(K)$ converges to $\frac{1}{N_6 \lambda}(1 + N(0,1)/\sqrt{K})$.

As $K \rightarrow \infty$, this means that the inter-arrival times converge to a deterministic distribution, and the system is asymptotically equivalent to a D/D/1 system with rate $\rho < 1$. For such system, the waiting time is zero. This leads to a probability of flow loss equal to zero as well. ∎

Scaling up both the arrival rate and the number of interfaces proportionally improves the performance (providing that $\rho < 1$). This means that a system dimensioned to work with a specific arrival rate and number of interfaces could be simply modified, if the arrival rate of flow request increases, by adding a proportional amount of interfaces to the rate increase.

Similarly, multiplexing an increasing number of IPv6 addresses (i.e. increasing N_6) should be compensated by a corresponding increase in K, so as to keep $\rho < 1$. If the increase of N_6 and K is exactly proportional, then the performance of the system will improve, as the packet loss decreases to zero. One could increase K slightly less if one wants to only keep the flow loss probability constant, but it should be linear to N_6 to satisfy the constraint $\rho < 1$. Thus a proportional relationship between K and N_6 is both necessary (Equation 1) and sufficient (Theorem 4.5). Also, please note that, while a sublinear growth of the number K of interfaces with N_6 would scale better, a linear growth is sufficient: there is currently a constant proportional factor for NAT translation, where a typical NAT box could multiplex up to 65,536 private addresses onto a single public address.

These results provide guideline to minimize the loss of new flows when K is the limiting factor. However, after the interface has been allocated to the new flow within the θ-time window, there is a risk that the new flow will come from a source which already has an on-going filtering rule set-up with the interface at the SIPNAT box. We discuss this issue in the next subsection.

C. Probability of Source Address Collision

SIPNAT establishes filtering rule based on the interface address $GWv4_i$ and the source address $Sv4$, which translates packets which match the filtering rule to the IPv6 address $Hv6$. If a new flow is assigned the same interface, the interface waits for the new flow, and creates the filtering rule corresponding to this flow towards the IPv6 host specified by the DNS request.

If the filtering is based strictly upon the IP header of the packets, then a collision occurs when a new flow is assigned the same interface as an existing flow from the same IP address. However, if the source is defined to include more protocol information (filtering on transport protocol, or using TCP/UDP ports, or using the TCP sequence number) then multiple flows from the same source IP address can be easily multiplexed through the same SIPNAT interface. We define a source to be the filtering information that the SIPNAT interface uses to forward the packet.

Then we ask: how many interfaces are required to reduce the probability of collision from the same (generalized) source below a given ϵ probability?

The probability of collision depends on the length of time a filtering rule is set at the SIPNAT, which is equal to the time length of the flow plus the time required for the interface to notice that the flow is over[3]. We denote by B the random variable describing the lifetime of a filtering rule at the interface.

We assume a specific flow is assigned to an interface randomly and uniformly over the set of interfaces: this is trivially true if the interface assignment is random. For round-robin assignments, two successive flows from the same source will be assigned to two interfaces separated by the number of new flows from other sources requesting an interface in between. Since this number is random, and if there are many such flow requests (with respect to K) between two successive flows from the same source, then one can assume the two successive assignments for the same flow are uniformly distributed.

One key point regarding the assignment is that the SIPNAT does not know the source of the flow when it assigns the interface, since it only receives the DNS request (it can assign flows requests issued by a given DNS server to different groups of interfaces, but the treatment is similar on the subset of interfaces).

A flow from source S will stay (as a filter rule) at an interface for a time period B and flows will arrive to this interface with rate λ_S/K where λ_S is the inter-arrival rate of flows from source S into the SIPNAT system.

A collision will occur any time that B is greater than the inter-arrival of subsequent flows from S to the same interface. Denoting by F_B the cumulative distribution function of B, this can be computed as:

$$P(collision) = \int_0^\infty (1 - F_B(x)) \frac{\lambda_S}{K} e^{-\frac{\lambda_S}{K}x} dx \tag{5}$$

Theorem 4.6: If B is exponentially distributed with rate μ (respectively deterministic with duration D), then the probability of a collision is:

$$\frac{\lambda_S}{\lambda_S + K\mu} \left(\text{respectively } 1 - e^{-\frac{\lambda_S D}{K}} \right) \tag{6}$$

Proof: This is a consequence of Equation 5 with F_B exponential or deterministic. ∎

Again, one can compute from Equation 5 the number K of interfaces required to set the probability of a collision below a given threshold ϵ.

V. Evaluation

We first consider the relationship between the number of servers and the time in the system. Figure 2 depicts the waiting time in the system (without impatience) for the assignment of an interface. Each curve corresponds to the same load in the system (plotted on the x-axis), but split along a varying number of interfaces (equal to 1,8,32,64,128 and 256; the latter corresponds to a /24 prefix assigned to the SIPNAT box on its IPv4 side). One can check that, despite the constant load,

[3]The interface notices by having a timer reach a time-out value; explicit notification, while possible, would require changes on the part of the client.

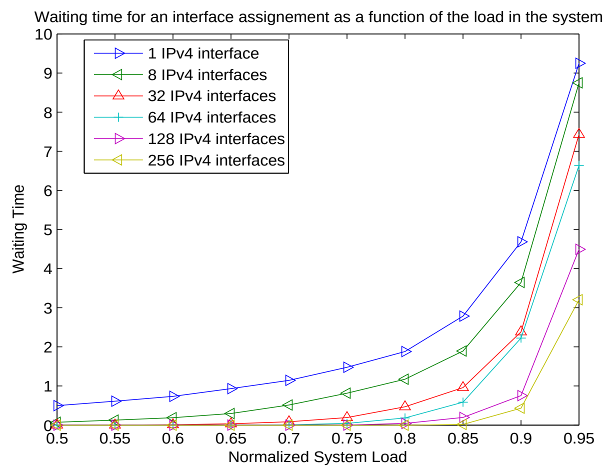

Fig. 2. Waiting Time as a function of the load for various numbers of interface.

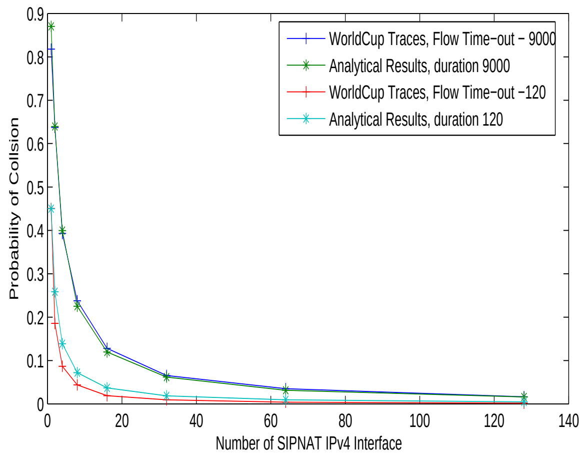

Fig. 3. Probability of a collision for two flows with the same source at a given interface

the increasing number of interfaces monotonously reduces the wait time, and thus the probability of losing a flow through the interface assignment process.

Figure 3 describes the probability of a source collision at a given interface derived in Equation 6 (deterministic) compared with the actual connection attempts to a server[4]. The data is slightly dated, but the distribution of the source address space has not *qualitatively* changed since then. The flow duration is composed of two parts: the actual transfer of data packets, and the time-out after the last packets for the interface to notice that the flow has become quiescent. The longer the interval, the more the flow duration looks deterministic, and the closer the approximation. In this case, the queueing theoretic model very accurately captures the properties of the system when run on actual traces.

[4]WorldCup'98 data from http://ita.ee.lbl.gov/html/contrib/WorldCup.html.

VI. Concluding Remarks

We have described a mechanism to provide global reachability to nodes behind a NAT/firewall, with application to private address spaces, and most importantly to IPv6 domains. The described mechanism is envisioned as an enabler for the transition from IPv4 to IPv6, and is thus of extreme importance to the Internet as a whole. We have modeled the mechanism using queueing theoretic analysis, and have assessed the asymptotic behavior as the number of IPv4 interfaces of the SIPNAT box is increasing. We have conducted a numerical performance evaluation as well which supports our analysis.

Future system designs must address mobility support: since the forwarding to the $Hv6$ destination address at the SIPNAT gateway is tied up to the source of the packet, any change in the source information due to a mobility event (for instance, change in the Care-of-Address [15]) would have to trigger an update in the filtering rules, and instances of triangular routing need to be handled specifically.

Deep packet inspection (DPI) of the payload for certain protocols (notably HTTP) allows often to unambiguously determine the destination. SIPNAT can dynamically reclassify such destinations so that new allocations may be overloaded much more readily even at interfaces where there are other existing pending flow allocations.

References

[1] C. E. Perkins, "Translating IPv4 to IPv6 based on source IPv4 address," Internet-Draft, behave Working Group, draft-perkins-sourceipnat-00b, June 2009.

[2] K. Egevand and P. Francis, "The IP network address translator (NAT)," IETF RFC 1631, Network Working Group, May 1994.

[3] P. Srisuresh and K. Egevand, "Traditional IP network address translator (traditional NAT)," IETF RFC 3022, Network Working Group, January 2001.

[4] B. Carpenter and K. Moore, "Connection of ipv6 domains via ipv4 clouds," IETF RFC 3056, Network Working Group, February 2001.

[5] P. Savola, "Observations of IPv6 traffic on a 6to4 relay," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 1, pp. 23–28, 2005.

[6] C. Huitema, "Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)," IETF RFC 4380, Network Working Group, February 2006.

[7] X. Li, C. Bao, M. Chen, H. Zhang, and J. Wu, "The CERNET IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition," IETF Network Working Group, draft-xli-behave-ivi-02.

[8] P. M. D. W. J. Rosenberg, R. Mahy, "Session Traversal Utilities for NAT (STUN)," IETF RFC 5389, Network Working Group, October 2008.

[9] "Skype," http://www.skype.com.

[10] R. Durrett, *Probability: theories and examples*, Wadsworth, Ed. Brooks/Cole Publishing, 1991.

[11] N. K. Boots and H. Tijms, "A multiserver queueing system with impatient customers," *MANAGEMENT SCIENCE*, vol. 45, no. 3, pp. 444–448, March 1999.

[12] G. Franx, "A simple solution for the m/d/c waiting time distribution," *Operations Research Letters*, vol. 29, no. 5, pp. 221–229, December 2001.

[13] A. Janssen and J. van Leeuwaarden, "Back to the roots of the M/D/s queue and the works of Erlang, Crommelin and Pollaczek," *Statistica Neerlandica*, vol. 62, no. 3, pp. 299–313, 2008.

[14] V. Iversen and L. Staalhagen, "Waiting time distribution in M/D/1 queueing systems," *Electronics Letter*, vol. 35, no. 25, pp. 2184–2185, 1999.

[15] C. E. Perkins, *Mobile IP: Design Principles and Practices*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.