NAT/Firewall Traversal Cost Model for Publish-Subscribe Systems

Debmalya Biswas
Nokia Research
Lausanne, Switzerland
debmalya.biswas@nokia.com

Kathryn Bean
SAP Business Objects
1012 Kingswood Avenue
Dublin, Ireland
kathryn.bean@sap.com

Florian Kerschbaum
SAP Research
Karlsruhe, Germany
florian.kerschbaum@sap.com

ABSTRACT

We consider large scale Publish/Subscribe systems deployed across multiple organizations. However, such cross organizational deployment is often hindered by firewalls and Network Address Translators (NATs). Several workarounds have been proposed to allow firewall and NAT traversal, e.g. VPN, connection reversal, relay routers. However, each traversal mechanism in turn leads to trade-offs with respect to implementation complexity, infrastructure overhead, latency, etc. We focus on the latency aspect in this work. We propose a cost-performance model that allows quantitative evaluation of the performance latency induced by the different firewall traversal mechanisms. The utility of the model is that for a given network configuration, it is able to provide a (close) approximation of the performance latencies based on simulation results, without actually having to deploy them in practice. This also allows selecting the best traversal mechanism for a given configuration. Finally, experimental results are given to show the validity of the proposed model.

Categories and Subject Descriptors

D.2.8 [**Software Engineering**]: Metrics—*performance measures*

General Terms

Performance, Measurement

Keywords

Cost performance model, NAT/Firewall traversal, Network latency, Publish-Subscribe systems

1. INTRODUCTION

We consider large scale Publish/Subscribe (Pub/Sub) systems capable of aggregating, propagating, and publishing business events in a consistent and timely fashion across multiple servers, locations, and organizations. All events are grouped by type. The system consists of the following main participants (Fig. 1):

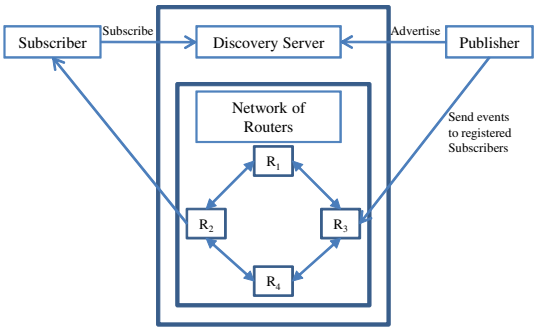

Figure 1: Pub/Sub Architecture

- Publishers: publish events and may charge subscription fees.
- Subscribers: are interested in receiving events of a specific type at specific intervals.
- Discovery server: is responsible for maintaining a mapping of which publishers can generate events of a specific type (and which subscribers are interested in events of which type). Publishers and subscribers basically send their advertisement and subscription messages to the discovery server.
- Network of routers: provides the underlying communication infrastructure to propagate events from publishers to subscribers. The publishers and subscribers establish logical point-to-point connections over the underlying network of routers, forming an overlay network. Each publisher/subscriber is connected to a dedicated router as shown in Fig. 1.

We envision cross organizational application scenarios for our Pub/Sub systems where not only the publishers and subscribers, but the network of routers may also be hosted across organizations. Our ultimate goal is thus to be able to seamlessly deploy the Pub/Sub system in a P2P fashion across multiple organizations. However such cross organizational deployment is often hindered by firewalls and Network Address Translators (NATs).

ICPE'11, March 14–16, 2011, Karlsruhe, Germany.

Firewalls and Network Address Translators (NATs) often make it impossible to establish a direct TCP connection between nodes across organization boundaries. TCP connections cannot be established between nodes protected by their respective firewalls as the node initiating the connection is outside the other nodeÂŠs firewall. The usual organizational security policy allows an internal node to initiate connections with external nodes, but not accept them.

Several workarounds have been proposed to allow firewall traversal, e.g.

- Relay nodes: If both nodes A and B are behind firewalls, then have them route all communication through a relay node C that is visible to both (forms the basis of the Traversal Using Relay NAT (TURN) [5] specification).
- Support multiple protocols: If a direct TCP connection cannot be established, try over HTTP, and finally over HTTPS. Most firewalls are configured to allow selected outgoing TCP traffic to port 80 (HTTP port) and port 443 (HTTPS port).
- Virtual Private Networks (VPN): VPNs enable communication between untrusted nodes by establishing secure encrypted channels between the nodes.

However, all the proposed firewall traversal mechanisms lead to trade-offs with respect to:

- implementation complexity, e.g. the code needs to be modified to accommodate communication over HTTP;
- infrastructure overhead, e.g. need for additional publicly addressable relay nodes;
- network latency, e.g. HTTP is slower than TCP, routing via relay nodes adds an additional hop, encryption increases the data size.

We focus on the latency aspect in this work. Our goal is to perform quantitative evaluation of the performance latency induced by the different firewall traversal mechanisms for a given deployment scenario. For a given network topology, we would like to provide a (close) approximation of the performance guarantees that can be provided by the Pub/Sub system. Previous works [3, 4] have proposed performance models for Pub/Sub systems, however accommodating NAT/Firewall traversal aspects is clearly a novel contribution of our model.

The paper is organized as follows: In Section 2, we present our performance model. NAT/Firewall traversal aspects are integrated in the model in Section 3. Experimental results to validate the performance model are given in Section 4. Section 5 concludes the paper and provides directions for future work.

2. PUBLISH/SUBSCRIBE COST MODEL

We are mainly interested in studying the latency of a given Pub/Sub network. We fist consider the latency of a point-to-point connection.

2.1 Point-to-Point Latency

For a point-to-point connection between routers r_1 and r_2, latency $l(R_1 \rightarrow R_2, e)$ refers to the time taken by an event to traverse from R_1 to R_2. In our experiments (Section 4), we have seen that for a specific transport mechanism (e.g. TCP/HTTP), the latency does not increase linearly with an increase in the event size. For instance, for a pair of events e and f having sizes s and $10s$ respectively, the latency $l(R_1 \rightarrow R_2, f) < 10 \times l(R_1 \rightarrow R_2, e)$. The same decrease in latency is also noticed as the frequency of events increases [7]. We thus define the latency $l_T(R_1 \rightarrow R_2, E_{s,n})$ to send n events of size sKB over a specific transport mechanism T as follows:

$$l_T(R_1 \rightarrow R_2, E_{s,n}) = \begin{cases} t & \text{if } n = 1,\ s = 1\text{KB} \\ \dfrac{n \times s \times t}{\alpha_T(s,n)} & \text{if } n > 1 \text{ or } s > 1 \end{cases}$$

where $\alpha_T(s,n)$ is the cost saving factor for transport T, specified as a function of s and n. The above latency also holds between a publisher/subscriber and its dedicated router. We next consider the latency between a publisher and subscriber.

2.2 Publisher-to-Subscriber Latency

Recall that the publishers and subscribers form an overlay network over the router infrastructure. Thus to compute the latency between a publisher P and subscriber S, we need to recursively combine the latencies of each pair of routers in the path connecting P to S. We assume that such router paths are static (pre-determined) for each P-S pair. Let there be m routers in the path connecting P to S, denoted $P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S$. The set of routers in a path p connecting P to S is given by $\mathcal{R}_p$. Given this,

$$l_T(P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S, e) = l_T(P \rightarrow R_1, e) + l_T(R_1 \rightarrow R_2, e) \cdots + l_T(R_m \rightarrow S, e) \quad (1)$$

Note that the path latency from P to S also needs to consider the latency between P and S and their dedicated routers (R_1 and R_m respectively). By slight abuse of notation, we refer to P and S as R_1 and R_m respectively in the sequel. In the absence of any ambiguity, we also denote $l_T(R_1 \overset{\{R_2,\cdots,R_{m-1}\}}{\rightarrow} R_m, e)$ in short as $l_T(R_1 \rightarrow R_m, e)$.

We now extend the above formula to n event sends from P to S. We consider *synchronous* communication where a router R_i starts processing the event e_j only after it has finished sending the previous e_{j-1} to R_{i+1}. Let (R_h, R_{h+1}) denote the pair of intermediate routers having the highest latency, i.e. $1 \leq h \neq j < m, l_T(R_h \rightarrow R_{h+1}, E_{s,n}) > l_T(R_j \rightarrow R_{j+1}, E_{s,n})$. Further, let $G(P)$ denote the publish rate of publisher P in terms of events/ms. We consider a highly active system where

$$\frac{1}{G(P)} < \frac{l_T(R_h \rightarrow R_{h+1}, E_{s,n})}{n} \quad (2)$$

THEOREM 1. *The latency to send n events of size s from publisher P to subscriber S is:*

$$l_T(P \equiv R_1 \overset{\{R_2,\cdots,R_{m-1}\}}{\rightarrow} S \equiv R_m, E_{s,n}) = \frac{l_T(P \rightarrow S, E_{s,n})}{n} + \frac{(n-1) l_T(R_h \rightarrow R_{h+1}, E_{s,n})}{n}$$

where (R_h, R_{h+1}) denotes the pair of intermediate routers having the highest latency. □

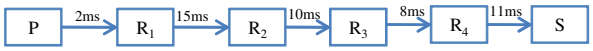

Figure 2: Sample scenario

The proof omitted here due to space constraints, can be found in the extended version available at `https://sites.google.com/site/debmalyabiswas/research/NF_CostModel.pdf`.

Note that by Theorem 1,

$$l_T(P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S, E_{s,n}) < \sum_{1\leq i<m} l_T(R_i \rightarrow R_{i+1}, E_{s,n})$$

This is because of the pipelining effect. For instance, if the latencies of intermediate router pairs in the path $P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S$ were strictly increasing, i.e. $1 < i < m$, $l_T(R_{i-1} \rightarrow R_i, E_{s,n}) < l_T(R_i \rightarrow R_{i+1}, E_{s,n})$. Then, $h = m-1$ in Theorem (1).

EXAMPLE 1. *We consider the latency for the scenarios depicted in Fig. 2. The times on the arrows denote the average latencies, i.e.* $R_1 \overset{x\ ms}{\rightarrow} R_2$ *implies* $\frac{1}{n} \times l_T(R_1 \rightarrow R_2, E_{s,n}) = x$. *Given this, the first event will reach S after* $46ms$. *The second event after* $61ms$, *the third after* $76ms$, *and so on. Note that subsequent events reach after an interval of* $15ms$, *which is basically the highest average latency among the intermediate router pairs (between* R_1 *and* R_2*).* □

Theorem 1 is very useful in practice as it allows one to compute the latency between a publisher-subscriber pair based on the latency of an intermediate router pair. This allows approximating the latencies without having to deploy the whole network.

In a heterogeneous system, the different publishers may generate events at different rates. If

$$\frac{1}{G(P)} > \frac{1}{n} \times \sum_{1\leq i<m} l_T(R_i \rightarrow R_{i+1}, E_{s,n})$$

then

$$l_T(P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S, E_{s,n}) = \sum_{1\leq i<m} l_T(R_i \rightarrow R_{i+1}, E_{s,n})$$

(There is no cost savings due to pipelining.) else if

$$\frac{l_T(R_h \rightarrow R_{h+1}, E_{s,n})}{n} < \frac{1}{G(P)} < \sum_{1\leq i<m} \frac{l_T(R_i \rightarrow R_{i+1}, E_{s,n})}{n}$$

then

$$l_T(P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S, E_{s,n})$$

$$\approx \frac{1}{n} \sum_{1\leq i<m} l_T(R_i \rightarrow R_{i+1}, E_{s,n}) + \frac{(n-1)}{G(P)}$$

else the latency is given by Equation (1).

In the sequel, we implicitly assume that Equation (2) holds, i.e.

$$\frac{1}{G(P)} < \frac{l_T(R_h \rightarrow R_{h+1}, E_{s,n})}{n}$$

2.3 Failure Model

We consider a dynamic Pub/Sub system where the routers may fail arbitrarily at any given point of time. We do not consider router path reconfiguration in the event of a failure. As such, in the event of a router failure, the transmission of events via that router is delayed till the router recovers.

Let λ_p denote the path failure rate of sending an event via path p. Then $\mathcal{F}_{(p,e)}$ denotes the set of failed routers while routing an event e via path p. Note that if $|\mathcal{R}_p| = m$, then $|\mathcal{F}_{(p,e)}| \approx \lambda_p \times m$. Further let $t_r(R)$ denote the recovery time of a router R. Then for send of an event e via path p, the latency of each router $R_i \in \mathcal{F}_{(p,e)}$ increases by an additional $t_r(R_i)$. We can now compute the latency $l_T(P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S, E_{s,n})$ given failure rate $\lambda_{\mathcal{R}=\{R_1,\cdots,R_m\}}$ as follows:

$$l_T(P \overset{\{R_1,\cdots,R_m\}}{\rightarrow} S, E_{s,n})$$

$$\approx \sum_{R_i\in\mathcal{R}'} \frac{l_T(R_i \rightarrow R_{i+1}, E_{s,n})}{n} + \sum_{R_j\in\mathcal{F}_{(p,e_1)}} t_r(R_j) + \sum_{k=2\cdots n} t_k$$

where $\mathcal{R}' = \mathcal{R} \setminus \mathcal{F}_{(p,e_1)}$ and $k = 2\cdots n, t_k$ is defined as follows: For an event e_k, let R_{e_k} be the router having maximum latency plus recovery time among the routers in $\mathcal{F}_{p,e_k}$. If

$$\frac{l_T(R_h \rightarrow R_{h+1}, E_{s,n})}{n} > \frac{l_T(R_{e_k} \rightarrow R_{e_k+1}, E_{s,n})}{n} + t_r(R_{e_k})$$

then

$$t_k = \frac{l_T(R_h \rightarrow R_{h+1}, E_{s,n})}{n}$$

otherwise

$$t_k = \frac{l_T(R_{e_k} \rightarrow R_{e_k+1}, E_{s,n})}{n} + t_r(R_{e_k})$$

EXAMPLE 2. *We again consider the sample configuration in Fig. 2, this time with failures. Let routers* R_1, R_2 *and* R_3 *fail while sending the first, second and third events respectively. Further let the recovery times of* R_1,R_2 *and* R_3 *be* $t_r(R_1) = 25ms$, $t_r(R_2) = 35ms$ *and* $t_r(R_2) = 5ms$. *Then the first event reaches S after* $(46+25)ms$. *The second event reaches S after a delay of* $(10+35)ms$ *as the latency plus recovery time of* R_2 *is greater than* 15, *the highest latency among intermediate router pairs (between* R_1 *and* R_2*). The third event reaches S after a delay of* $15ms$. *Note that although there was a failure (of* R_3*) while sending the third event, the latency plus recovery time* $(8+5)ms$ *of* R_3 *is less than* $15ms$. □

3. NAT/FIREWALL TRAVERSAL COSTS

In this section, we outline how the different NAT/Firewall traversal mechanisms can be integrated into the performance model presented in Section 2.

3.1 HTTP

HTTP can be considered as the simplest firewall traversal mechanism as incoming HTTP traffic on port 80 is usually allowed by default, even in the most restrictive enterprise scenarios nowadays. It can be easily integrated into our performance model by performing latency measurements for a new transport mechanism, i.e. computing

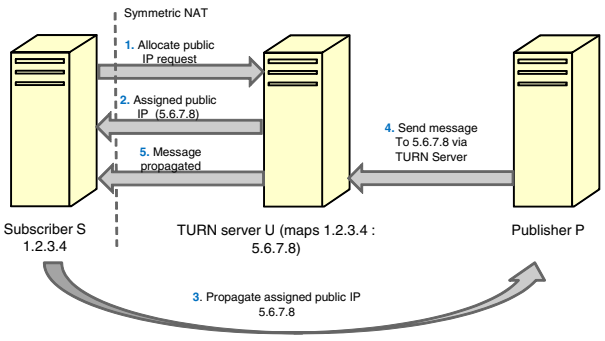

Figure 3: Traversal via Relays (TURN)

$$l_{HTTP}(R_1 \rightarrow R_2, E_{s,n}) \tag{3}$$

A technical comparison of the HTTP vs. TCP protocols is beyond the scope of this paper, however we suffice to say that HTTP is slower than TCP. The precise difference in latency in quantified later in Section 4. We also discuss alternatives to bridge this gap in latency in Section 4.

3.2 Traversal via Relays

We first give a simplified description of the TURN protocol, focusing on the details important from a performance perspective. (For a more detailed description, the interested reader is referred to [5].) We consider the scenario illustrated in Fig. 3. Publisher P would like to send events to subscriber S. However, S is behind a symmetric NAT, as such any attempts to initiate communication by P (or an intermediate router R) would be rejected by S. To overcome this:

1. S sends an "Assign public IP address" request to the TURN server U.
2. Let 1.2.3.4 be the source IP address of the request message (as exposed by S's NAT). U maps 1.2.3.4 to say 5.6.7.8 and notifies S. As U had previously received a message from S, U's messages will be allowed to reach S. U also maintains a record of this mapping to be able to route any future messages to S.
3. S informs P about it mapped public IP, in this case 5.6.7.8. This can be performed by something as simple as exchanging email messages or specific "rendezvous" protocols. How this is performed is even outside the scope of TURN.
4. Given this pre-processing, to send an event to S, P sends an event to U with instructions to route it to S's public IP 5.6.7.8.
5. U reverse maps 5.6.7.8 to 1.2.3.4, and then forwards the event to S.

Routing via relay peers is more-or-less guaranteed to succeed even under the most restrictive NAT settings. However, it clearly has a latency overhead, that of an additional send for each event send. The TURN latency for sending n events from router R_1 to R_2 via relay U is given as:

$$l_T(R_1 \xrightarrow{U} R_2, E_{s,n}) = L_1 + L_2 \tag{4}$$

where $L_1 = l_{T_1}(R_1 \rightarrow U, E_{s,n})$ and $L_2 = l_{T_2}(U \rightarrow R_2, E_{s,n})$. Technically, it is possible that different transport mechanisms are used while communicating between R_1 to U (T_1) and U to R_2 (T_2).

The other aspects we need to consider for TURN like traversal mechanisms are the initial allocation times and additional keep-alive messages. Note that the public IP allocation times are relevant only when the router joins for the first time or rejoins after a failure. For our performance model, it suffices to assume that this initial allocation time for a router R is included in its recovery time $t_r(R)$. Keep-alive messages need to be sent regularly to the TURN server to keep the allocation alive. For instance, the default lifetime of an allocation as specified by TURN is 10 minutes, i.e. a keep-alive message would need to be sent every 10 minutes. In a general scenario, let f messages/ms and s_k denote the frequency and size of keep-alive messages. Then the TURN latency needs to be extended as follows:

$$l_T(R_1 \xrightarrow{U} R_2, E_{s,n}) = L_1 + l_{T_1}(R_1 \rightarrow U, E_{s_k, f \times L_1}) + L_2$$

3.3 Virtual Private Networks

VPNs can be broadly categorized into two types:

- Remote client VPNs: connecting a single PC using VPN software to the host network on demand.
- Site-to-site VPNs: is generally a permanent connection between two sites using dedicated channels.

In this work, we will mainly focus on the latter type, i.e. site-to-site VPNs. To understand the effect of VPNs on latency, we give a brief overview of the Layer 2 Tunneling Protocol (L2TP) [6]. We chose L2TP over the alternative Point-to-Point Tunneling Protocol (PPTP) [2] as L2TP is "more" secure. L2TP provides additional levels of security via the use of both authentication and data encryption. Authentication computes an Integrity Check Value (ICV) over the packet's contents, and it is usually built on top of hash algorithms such as MD5 or SHA-1. It incorporates a secret key known to both routers, and this allows the recipient to perform an integrity check on the received packet. Encryption uses a secret key to encrypt the data before transmission, with DES, 3DES, Blowfish and AES being the common choices for IPSec encryption. As such, data packets under L2TP are encapsulated on an average five times, depending on the IPSec policy being used. Each level of encapsulation further increases the message size. L2TP also relies on UDP which is a different transport mechanism than the assumed default TCP.

The VPN latency between routers R_1 and R_2 to send n events is thus given by

$$l_{UDP}(R_1 \xrightarrow{V} R_2, E_{s,n}) = l_{UDP}(R_1 \rightarrow R_2, E_{s+s_e,n}) \tag{5}$$

where s_e is the increase in message size due to encryption/authentication headers.

4. EXPERIMENTAL EVALUATION

This section presents experimental results that allow one to evaluate the performance in terms of latency of the Publish/Subscribe system as well as that of the different traversal mechanisms. We first describe the simulation setting

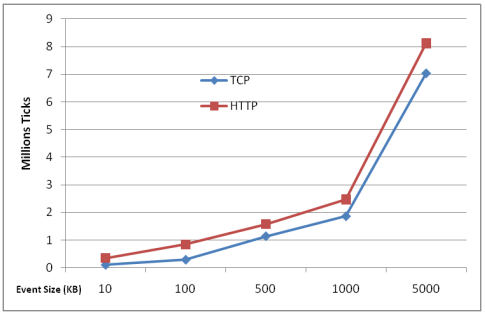

Figure 4: HTTP vs. TCP latencies to transmit 10 events between a pair of routers

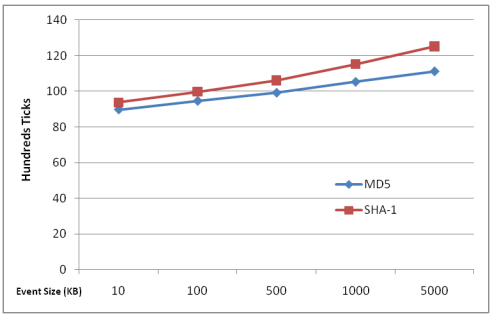

Figure 5: MD5 vs. SHA-1 latencies

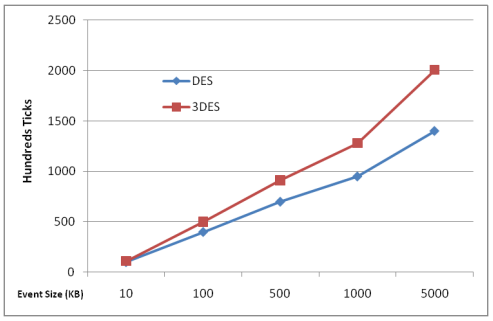

Figure 6: DES vs. 3DES latencies

and then our results. We finally compare our empirical results with those as predicted by our theoretical performance model, verifying correctness and usefulness of our performance model.

4.1 HTTP versus TCP

In this section, we compare the latency of sending events via HTTP vs. TCP between a pair of routers. In theory, TCP is expected to be faster than HTTP as TCP transmits raw binary data over TCP sockets, and there is no encoding (decoding) involved.

For our experiments, the HTTP and TCP clients/servers were programmed in C++ using the .NET framework. The HTTP messages were sent as POST messages. The client-server programs were run on a pair of machines in the same intranet. The latencies are given in Fig. 4. As expected, HTTP is slower than TCP with a latency of 349352 ticks to send 10 events of size 10KB over HTTP as as compared to 99676 ticks over TCP. We consider the latency of 10 event sends to reduce the chances of any "one-off" discrepancies affecting the performance study. The more interesting part is the rate at which the latency increases as the event size increases. As can be observed, the increase in latency is not proportional to the increase in event size. For instance, the latency to transmit 10 events of size 5000KB over HTTP is 8123961 ticks as compared to 7034366 ticks over TCP, i.e only 1.15 times more time to send 5000KB events as compared to 3.5 times more time to send 10KB events. This is because the encoding time in case of HTTP does not increase proportional to the latency, as the event data size increases. *Indirectly, this implies that the latencies of sending events over HTTP and TCP will be comparable as long as the events are sufficiently large.*

The above pattern is captured by the cost saving factor $\alpha_T(s, n)$ (Section 2.1) in our theoretical model.

4.2 VPN Authentication and Encryption

In this section, we study the increase in latency as a result of VPN authentication and encryption. We compare the latencies of IPSec authentication hash algorithms: MD5 vs. SHA-1 and encryption algorithms: DES vs. 3DES. The algorithms were implemented using the System Cryptography class of the .NET framework. The transport mechanism used in this case is UDP.

Fig. 5 compares the latencies of hashing algorithms MD5 vs. SHA-1 as the event data size increases. For small event sizes, their latencies are comparable with SHA-1 slightly slower than MD5. This is because MD5 produces a 128 bits hash as compared to 160 bit hash produced by SHA-1. As the event size increases, MD5 clearly leads but then MD5 is also "less" secure than SHA-1. The latency to send a 5000KB event is 12.5% more when SHA-1 is used as compared to using MD5.

Fig. 6 compares the encryption choices: DES vs. 3DES. The (key, block) sizes used for DES and 3DES are $(64, 64)$ and $(192, 64)$ respectively. The key and block sizes can be interpreted as follows: 3DES encrypts (and decrypts data) in 64-bit blocks using a 192-bit key. 3DES is basically an extended version of DES where the DES algorithm is run 3 times with 3 keys of 64 bits (so the need for 192 bit key sizes by 3DES) with encryption by the first 64 bit key, followed by decryption by the second 64 bit key, followed by final encryption by the third 64 bit key. As expected, DES is faster than 3DES but its shorter key length also makes it more vulnerable. We can also observe that the increase in latency as a result of encryption is much more than that due to authentication, leading us to conclude that it is mainly the encryption choice that needs to be made more carefully while considering the security vs. performance trade-off. In general, we found that the message size can increase up to 50% as a result of encryption. The above increase in message size as a result of authentication/encryption is of course accommodated in our theoretical model (See Equation (5)).

4.3 Publisher to Subscriber Latency

We first study latencies for the scenario where there are no Firewalls/NATs. For the same scenarios, we then study

the latencies with specific routers behind NAT and TURN traversal deployed to traverse them. TCP is the transport mechanism used for all experiments in this section.

We deployed a Pub/Sub system consisting of 3 publishers, 3 subscribers and 14 routers forming the router network. On an average, a publisher-subscriber pair was connected via 4 intermediate routers. Publishers maintained a stable publication rate, i.e. published events at regular intervals. The actual publication rate depended on the latencies of the intermediate router pairs. Recall that we assume the time between subsequent publication of events to be less than the average latency of the maximum latency intermediate router pair, refer to Equation (2).

Simulating NAT behavior in an intranet setting is a challenge in itself. We simulated NAT behavior by running specific routers as VMWare virtual machines. A VMWare virtual machine can be configured to run behind a NAT [1], with the host machine acting as the respective NAT. The VMWare NAT Windows Service running on the host machine provides the NAT functionality. The VMWare virtual machine network connection behavior can be controlled by choosing between "Bridged" and "NAT" connection types, with "Bridged" implying that the virtual machine has its own identity in the network (independent of the host).

The above complication also applies with respect to simulating a TURN server in an intranet setting. Ideally, the TURN server should be a publicly addressable machine in the internet. We simulated it as Linux machine having multiple IP addresses in the same intranet. To simulate the internet latency, set the latency between a router and TURN server to be more than that between any pair of routers.

Fig. 7 gives the publisher-subscriber latencies for transmitting 5 events of size 500KB over a path containing 4 intermediate routers (say, $P \rightarrow R_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_4 \rightarrow S$). The graph shows the increase in latency as the number of router pairs requiring NAT traversal increases. For the first event send, only router R_1 is behind NAT. For the second event send, routers R_1 and R_2 are behind NAT, and so on. It is interesting to note that the increase in latency is not proportional to the number of intermediate routers needing NAT traversal. This is because the gap between subsequent events reaching the subscriber depends only on the intermediate router pair requiring maximum TURN traversal time. For instance, in the experimental set-up, the router pair $P \rightarrow R_1$ has the highest average latency, i.e. $\frac{l_T(P \xrightarrow{U} R_1, E_{500,10})}{10} = 232212$ ticks. As a result, subsequent events reach the subscriber after approximately 232212 ticks. The average latency when none of the routers are behind NAT is equal to 512127 ticks.

The above observations are clearly in sync with the behavior predicted by Theorem 1 and Equation (4).

While performing the experiments, we realized that an alternative to achieve comparable performance would also be to embed some "intelligence" in the TURN server. Let us assume that the router path is static (e.g. the router path is decided at event publish time) and a a group of routers in sequence ($R_2 \rightarrow R_3 \rightarrow R_4$) in the path all required NAT traversal. Further assume that the same TURN server is used for NAT traversal of all the concerned router pairs. Given this, when the TURN server receives an event from R_2 to be propagated to R_3, it can propagate it directly to R_5 saving some traversal time ($R_3 \rightarrow R_4$ and $R_4 \rightarrow R_5$).

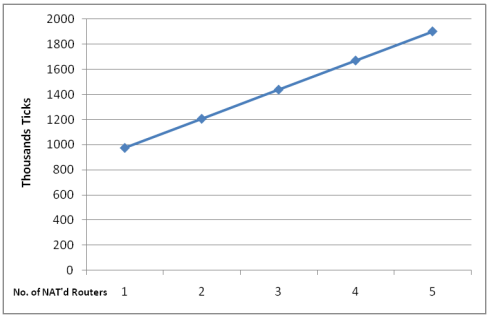

Figure 7: TURN traversal effect on network latency

5. CONCLUSION

In this work, we presented a performance model to capture the network latency of Pub/Sub systems. With firewalls and NATs becoming omnipresent in today's networks, it is important to be able capture the effect of their corresponding traversal mechanisms on the system performance. Towards this end, we showed how the performance model can be extended to accommodate NAT/Firewall traversal aspects. The model was validated via preliminary experimental results. Experiments w.r.t. the failure recovery part of the model are planned as future work. Future work also includes extending the theoretical model to accommodate less active systems where e.g. the publish rate is less than the average latency between a publisher-subscriber pair.

6. ACKNOWLEDGMENTS

This work is partly funded by the European Commission through the ICT program under Framework 7 grant 213531 to the SecureSCM project. We would like to thank the anonymous referees for their helpful suggestions.

7. REFERENCES

[1] VMware Workstation 4.5: Understanding NAT. `http://www.vmware.com/support/ws45/doc/index.html`, 2010.

[2] K. Hamzeh. Point-to-Point Tunneling Protocol (PPTP) Specification. `www.ietf.org/rfc/rfc2637.txt`, 1999.

[3] M. A. Mastouri and S. Hasnaoui. Performance of a Publish/Subscribe Middleware for the Real-Time Distributed Control Systems. *Computer Science and Network Security*, 7(1):313–319, 2007.

[4] S. Oh, J.-H. Kim, and G. Fox. Real-Time Performance Analysis for Publish/Subscribe Systems. *Future Generation Computer Systems*, 26(3):318–323, 2010.

[5] J. Rosenberg, R. Mahy, and P. Matthews. Traversal Using Relay NAT (TURN) Specification. `http://tools.ietf.org/html/draft-ietf-behave-turn-16`, 2009.

[6] W. Townsley. Layer Two Tunneling Protocol (L2TP) Specification. `http://www.ietf.org/rfc/rfc2661.txt`, 1999.

[7] L. Zhai, L. Sun, and Y. Liu. Modeling and Evaluation of High-performance Publish-Subscribe Systems. In *Computational Intelligence and Design Symposium Proceedings*, pages 457–460, 2008.