Analytical Modelling of Localized P2P Streaming Systems under NAT Consideration

Mohammad Z. M. Masoud

Huazhong University of Science and Technology, Wuhan, China

ABSTRACT

NAT has been design to work with Internet client-server structure. The emerged of Peer-to-Peer (P2P) networks and applications revealed the incompatibility between P2P applications and NAT. Many methods has been developed and implemented to solve connectivity between peers behind NAT devices. Nevertheless, various NATing types can't communicate with one another. In this work, we are going to study the impact of NAT types on the start-up delay time of peers in P2P streaming systems. We will demonstrate the ability of NATing to expel peers in P2P live streaming systems. A new neighbour selecting algorithm will be proposed. This algorithm will utilize NAT-types configurations as a parameter. We have utilized NS2 simulator to show the performance of the new algorithm in increasing the connectivity, reducing the number of expelled peers and implementing of locality.

KEYWORDS

Network address translation, peer-to-peer; live streaming; Internet service provider; locality awareness

1. INTRODUCTION

Due to their outperforming, P2P streaming systems have proliferated in the past few years. The massive amount of traffic that P2P streaming systems generate and freely propagate across ISPs has revealed new concerns [1]. These concerns recap in the ability of ISP's networks to sustain the cost of P2P traffic on their transit network.

Locality awareness emerged as the niche to cure the transit traffic load issue [2]. Locality awareness algorithms attempt to replace random content recourses (neighbour peers) with localized ones. Due to the selection process of new localized resources (peers), locality awareness algorithms unveiled performance shortcoming [3], [4]. This shortcoming appears in P2P file sharing systems as an increasing in the download time. Researches Attempted to ameliorate and enhance the knot that locality awareness algorithms left behind. This enhancement emerged in modifying the selecting of the localized peers in different ways. Even by adding a percentage of random peers in the selecting process or by the addition of cache devices and amplifier peers in ISP's networks [5], [6], [7]. However, measurement studies revealed a new issue that may impact the performance of P2P streaming network and the implementation of locality awareness. Network Address Translation (NAT) and its popularity have been reported as a performance degrading factor in P2P networks [8], [9].

NATing is defined as the process of mapping a private IP address of a source peer in the packet header into a public IP address that can surf the Internet. This method has been proposed to solve the shorting in IPv4 address space [10]. NATing has been implemented to operate in the original client-server architecture of the Internet. Many methods have been proposed to fit NATing in P2P applications [11], [12], [13]. Nevertheless, many types of NATing are classified as unfriendly according to P2P applications. NATing has the ability to expel peers from P2P networks. This fact reduces the number of peers in P2P systems which on the other hand, may reduce the

DOI : 10.5121/ijcnc.2013.5306

contribution of upload bandwidth in these networks [14]. This issue may impact the performance of P2P file sharing systems by increasing the download time. However, in P2P streaming systems it may increase the start-up delay of peers or even worst by reducing the number of joining peers. This makes P2P streaming systems more prone to the performance declining. However, how NATing may affect locality awareness algorithms?

Measurement studies have revealed results of the percentages of different NAT types against their locations [15]. According to these measurements, NAT type's popularity varies in different geographical locations. With the implementation of locality awareness algorithms, the concentration of unfriendly NAT types will be higher in some places than others. This may lead to failure or collapsing of P2P networks in these areas. Even if P2P streaming networks survive, the start-up delay of these networks will be unsustainable.

In any live streaming system, the peer starts to join by sending a message to the source. The source answers with a peer list. The node then starts to contact these peers to obtain the full stream. If it cannot obtain the full stream, it attempts to connect to other peers in the system with a second connection round. The number of rounds depends on the P2P streaming system's design. With an increasing number of rounds, the probability of finding suitable parents increases including the playback start-up delay time. Zattoo [10] for example, uses three rounds before the peer stops trying. When the peer stops trying, we call it expulsion.

In this paper, we show that in any live streaming network there is a high probability that, peers will be expelled. We introduce theoretical upper and lower boundaries for the number of kicked-out peers. We also propose a new and easy to implement parent selection algorithm to build a non-random peer list that reduces transit traffic, increases peer connectivity and diminishes playback start time delay by reducing the number of rounds. This mechanism tries to keep the number of expelled peers as close as possible to the lower boundary. In addition, a start-up delay mathematical modelling of P2P live streaming system under NAT type configuration will be introduced. This mathematical model considers peers NAT type as important parameter in calculating the start-up delay time. We found that neighbor selection algorithms that depend only upon AS awareness are not enough for live P2P streaming system. We will also show that there is a trade-off between traffic locality and peer connectivity. Our contribution in this work can be summarized as following:

- Introducing a theoretical upper and lower boundary of the number of expelled peers in P2P streaming systems.
- Mathematical analysis of the start-up delay time with the presence of NAT types will be introduced.
- Proposing a NAT-aware neighbour selecting algorithm that takes into account the AS numbers and NAT type as parameters.
- Demonstrating that implementing AS aware algorithms without considering NAT types in P2P live streaming systems may lead to reduction in system performance.

The rest of this paper is organized as follows. We end this section by describing related works done in this area. Section 3 will introduce NATing and its relation with P2P applications. Section 4 introduces the mathematical analysis of the start-up time and the impact of NATing on it. The theoretical upper and lower boundaries will be demonstrated in section 5. We conducted a simulation study to validate our proposed P2P streaming architecture and then analyzed the simulation results in Section 8. Finally, we conclude this paper in Section 9.

2. RELATED WORK

NATing and its impact on P2P streaming systems have attracted network measurement researches. Many measurement studies have been conducted to measure NAT type percentage and their impact on performance.

NAT type percentages have a critical impact on P2P performance. In [16], a study of P2P VoD system has been conducted. This study reported the popularity of symmetric NATing. This type of NATing is defined as unfriendly for P2P applications. In addition, it has been reported by studying P2P file sharing system the correlation between geographical locations and NAT type distributions [15]. This correlation demonstrates the possibilities of harming P2P applications with the implementation of locality-aware algorithms.

NAT impact on P2P performance has many aspects. NAT may expels nodes from P2P applications [8], [9]. In addition, it may increase the start-up delay time. These aspects are classified as performance declining aspects. These two aspects harm P2P streaming applications.

NATing did not only inspire network measurement studies, mathematical modeling of P2P applications with the consideration of NAT has been studied heavily in the past few years. In [17], the authors created an analytical model of the impact of NATing on P2P file system. In addition, they extended their model to cover Bittorent based live steaming system [18]. In [19], the author studied the impact of un-connectable peers (peers behind firewalls and NAT devices) on the performance of swarms in P2P streaming systems. However, these work assumed that Nated peers can only connect with open peers. The authors neglected the fact that different NAT types can connect with each other. This assumption expels most of the peers in any live streaming system since the percentage of Nated peers is massive comparing with open peers. In our analytical model, we will consider the fact of connectivity between different kinds of NATing and the impact of this connectivity on the start-up time.

3. P2P AND NAT

As we have mentioned, NAT is based on mapping private IP addresses into public IP addresses that are allowed to surf the Internet (one-to-one mapping). However, to allow many-to-one mapping, NAT utilizes transport layer parameters (port numbers). This technique is known as Port address Translation (PAT).

PAT is the technique that maps the port number in the packet header with a new different number. This technique allows NAT devices to map multi-internal hosts (private IP addresses) into one external public IP address (many-to-one mapping). PAT is a special kind of NAT and sometimes has NAPT abbreviation. However, in this paper we used NAT to refer to NAPT.
Over the past years, NAT has evolved to enhance the security of its protocol. Four different types of NAT configuration were the breed of these enhancements. These types can be summarized in the following [20]:

- Full cone: In this process, NAT device maps the address of nodes in the internal zone to the same external address for every flow. This process allows any node from the external domains to send packets to the internal node through that external address.
- Restricted address cone: In this process, external nodes can only sends packets to the internal node if and only if this internal node contacted that external node before.

- Restricted Port cone: In this process, external nodes can only sends packets to the internal node if and only if this internal node contacted that external node before with the same port number.
- Symmetric: In this process, NAT device maps the internal address and port number to different pairs of external address and port for each flow. An external node can only respond to an internal node.

All types of NAT operate in a client-server manner. The client in an internal zone initiates a request to obtain resources from a server in the public Internet zone. The client has a private IP address with a private port number. On the contrary, the server owns a pubic IP address with a well-known port number.

To operate, NAT requires the client to initiate a request to the server. The server on the other hand responses to the same IP and port address of the received packet's source address. However, in P2P network the source and the destination of a request might be nodes in private zones with private IP addresses. In this case, NAT operation encounters connectivity issue since both of the IP addresses and ports are private. This issue makes it impossible for two nodes behind NAT devices to communicate with each other. Researchers proposed many methods to allow connectivity between two different nodes behind NAT devices, such as, Universal Plug and Play (UPnP) [21], [22], Simple Traversal UDP through Network Address Translators (STUN) [23] and Traversal Using Relays around NAT (TRUN) [11], [24]. However, we will emphasize on STUN since it can be utilized to predict the NAT type configuration of the NAT device that nodes lay behind.

STUN is a client-server protocol. In this protocol, the client pack its IP address in the data portion of its request. Subsequently, when the server receives the packet, the server compares the IP address in the data portion and the original IP address from the packet header to predict the type of NAT device that node lay behind. This protocol is an important part in constructing NAT-aware neighbor selecting algorithm as we will demonstrate in the following sections.

4. START-UP TIME MATHEMATICAL ANALYSIS

4.1-Sub-Streaming System Start-up Delay

When a node connects to the server, the node sends a join message. The node then starts to determine its NAT type by connects with a special server in the system by using a lot of ways. The most well known why is STUN. Subsequently, the node needs to exchange Y number of message with the server. Each one of these Y messages needs t1 time. The delay to determine the NAT type will be as in Eq. 1.

$$D_1 = Y * t1 \quad (1)$$

Subsequently, the server selects random number of peers L from the system and constructs a list with the addresses of these peers. The server sends this list to the new peer. The time to send the list is very small so we neglected it. When the peer receives the list, it chooses N peers from L to contact with. These peers' responds with a list of their highest sequence number of each sub-stream they have. Subsequently, the node starts to construct its neighbour list after time D_2 as in Eq. 2.

$$D_2 = \min\{\theta 1, \max\{R_t(e), e = 1, 2, 3....N, \}\} \quad (2)$$

where θ_1 is a threshold time that a peer will wait to receive all the answers and R_t is the round trip time

If the peer does not receive all the answers after θ_1, it will neglect the other answers. Depending on the responses that the peer has received, the peer will first check if these peers can upload the whole stream (a complete copy). To do this, assume that the peer must have at least u number of parents as in Eq. 3.

$$u = L/\theta_2 \quad (3)$$

If these u parents cannot upload a complete copy of the stream to the new peer, the peer will try to contact with other peers from the list L with a second round of trying. The delay of any round can be as in Eq. 4.

$$D_3 = W * D_2 \quad (4)$$

where W is the number of rounds

The total start-up delay time will be as in Eq. 5.

$$D_t = D_1 + D_3 + D_b \quad (5)$$

where D_b is the buffer delay time

TABLE I
VARIABLES DEFINITIONS

Variable	Definition
Hxy	Percentage of y NAT configuration that x NAT can connect with
L	number of nodes in the neighbor List
N	Nodes that the peer chose to contact with
M	Total number of Peers in the system
R	video rate
W	number of rounds
L_u	useful list
N_u	number of useful node

4.2-Sub-Streaming System Start-up Delay with NATing Type consideration

In our model, reduction of Playback delay depends on reducing the rounds that a peer needs to obtain a complete copy of the stream. Hence, this number depends on the number of rounds a peer needs to find peers with NAT configuration that allows it to connect with.

NAT configurations have six different options including: 4 types (Symmetric, Cone, Port and IP Restricted) and another 2 configuration open (No NATing) and UDP disabled. Not all these types can connect with each others. Lets denote the percentage of the peers with a NAT type that can connect with a peer with b NAT type as H_{ab}. Assume that the streaming server uses the normal distribution of the peers when it construct a new neighbour list. The probability that the server can construct a full list with all the peers in the list can connect with the new peer is in Eq. 6.

$$P_1 = \prod_{1<i<L} \frac{(H_{ab} * M) * i}{M - i} \quad (6)$$

where M is the total number of peers in the system

As example: Assume that UDP disabled percentage in the system is 20%. UDP disabled can only connect with open peers. Assume the percentage of open peers is also 20%. Assume the network has 2000 peers and the size of the list that the server will sends is 10. The probable that the server will construct a list with 100% useful peers is 8:35x10-8 which is a very small probability. To increase this probability let's consider the probability p2 that the server will send a list that has u number of useful addresses to build the parent list as in Eq. 7.

$$P_2 = \sum_{u<i<n} \binom{L}{i} * H_{xy}^{i} * \left(1 - H_{xy}\right)^{L-i} \quad (7)$$

This gives the number of useful peers from the list as Eq. 8.

$$Lu = L * P2 \quad (8)$$

After the peer receives the list, it will contact N peers from L as in Eq. 9.

$$N = L/\theta 2 \quad (9)$$

$$N \geq u \quad (10)$$

Let P3 denotes the probability that at least u peers will answer as in Eq. 11.

$$P_3 = \sum_{u<i<N} \binom{N}{i} * P_2^{i} * (1 - P_2)^{N-i} \quad (11)$$

To prevent another round, the number of peers Nu that will respond must be greater than u as in Eq. 13.

$$Nu = P_3 * N \quad (12)$$

$$Nu \geq u \quad (13)$$

$$\frac{\theta_1 * L * P_3}{\theta\_2} \geq L \quad (14)$$

$$\frac{\theta_1 * P_3}{\theta\_2} \geq 1 \quad (15)$$

The last equation shows that to reduce the start-up delay time, we have to:

- Increase the list that should be send to any peer (Increase θ_1)
- Increase the number of peers that the new peer should contact (Reduce θ_2)
- Increase the probability of finding useful peers (Peers with NAT type that the new peer can connect with)

To study P3, we have implemented our own simulator in Python. This simulator consists of six different NAT configuration options; type 1 (No NAT), type 2 (full cone), type 3 (IP restricted), type 4 (port restricted), type 5 (Symmetric) and type 6 (UDP disabled). Type 3 has been neglected since it obeys the same connectivity rules of type 4 as in table II. As we can observe from the figure, even with the increase in upload speed, the probability P3 still less than the threshold that is required for type 6. This figure demonstrates that peers upload speed should be high or equivalent to the video rate to reach an appropriate P3 value.

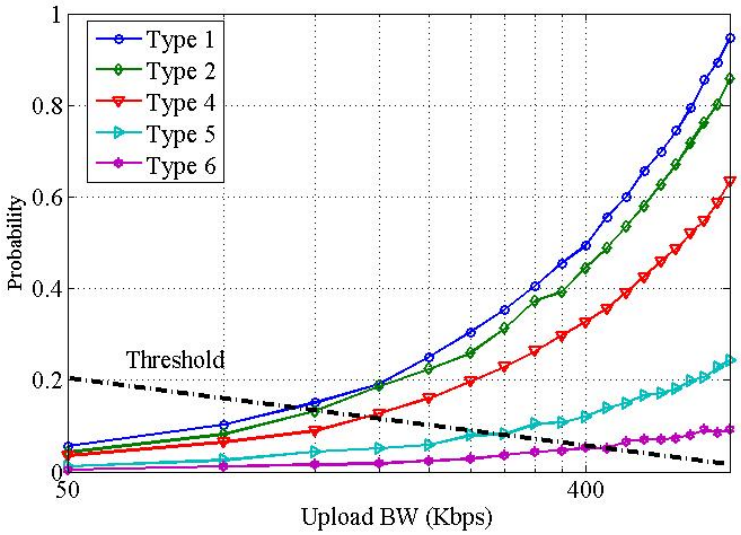

Fig. 1. The Relation between P3 and Upload BW

5- THE EXPELLED PEERS BOUNDARIES

To demonstrate that NATing has the ability to expel peers from P2P live streaming system, we need to understand how this can happen.

Four types of NATing can be found: Full Cone, Port Restricted, IP Restricted, and Symmetric. In P2P live streaming systems, peers can be NATed with one of these four types plus another two configuration options: Open (without NATing) and UDP disabled. Table I shows the ability to establish a connection among these types. From the table we can observe that even with NATing solutions, such as, NAT traversal, not all NAT types can connect to one another. What, however, is the probability of this scenario? To answer this question, we wrote all possible combinations of NAT configuration percentages that any network may have. We used the traces from [10], [11], [12], [13] as constrains to write only the possible percentages combinations. To anneal our prediction, we used five different average speed classes. 3,764 different possible combinations of NAT percentage scenarios were constructed. We used C language to write a program that construct random pairs of types. Each pair, simulates a connection between any two NATing types. The output of any pair, has one of two status outputs: true or false. False status means that peers from these NAT types in this pair cannot connect with each other and one of them will be expelled. We used 100 runs to create 100 random possible ways for each type to find its match. We done this in each one of the combination that we created to find the scenario with the highest connected peers (lower boundary) and the scenario with the lowest connected peers (highest boundary).

Within our theoretical boundaries, we used five NAT configurations (port restricted and IP restricted are the same in our model depending on table I). To predict the minimum theoretical number of peers expelled in a system, peers with open configurations should connect with UDP disabled peers in a higher priority and vice versa. Peers with symmetric NATing must connect with full cone peers as a first priority then with open peers. Full cone peers must connect with any type except open configuration peers and themselves. For the restricted configuration, (IP or port) peers can connect with themselves as a first priority then with other types. The percentage of IP

restricted is always high [10], [11], [12], [13]. If peers follow this schema, the number of expelled peers will be at a minimum. We name it the Lower Boundary.

To obtain the upper boundary, which is the worst case, we found that every type of NATed peers must connect only with itself. In such a case all UDP disabled and symmetric
NATed peers will be expelled with other peers depending on the average upload capacity. We found that from these 3764 different combinations of NAT percentage in any network with average upload capacity equal to half of the video rate, only 0.14% of these tries have
100% connected peers.

Fig. 2 shows the percentage of expelled peers with different video rates. This figure shows that the upload capacity can decrease the number of expelled peers but cannot eliminate it.

Fig. 3 shows number of tries where no peers were expelled. From this figure, we can conclude that for low average upload capacities the probability of having no expelled peers is approximately zero for random selection and high for priority selection.

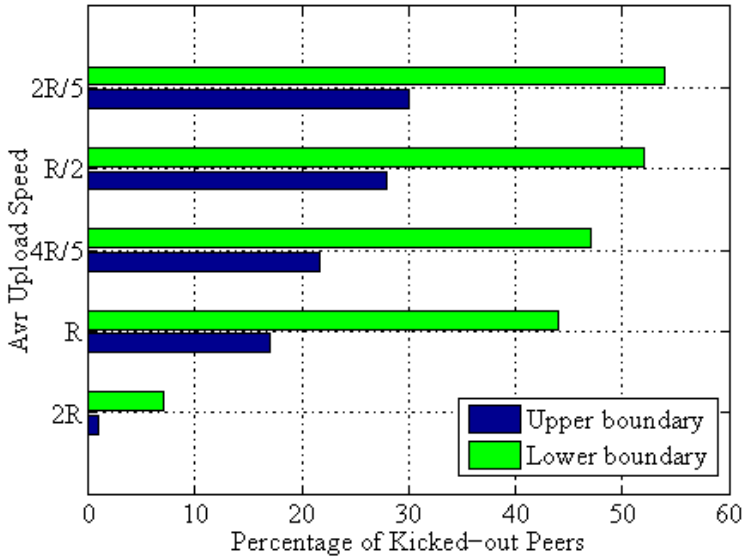

Fig. 2: Kicked-Out Peer Percentage to the Avg Upload Capacity

6. THE PROPOSED ALGORITHM

The proposed algorithm aims to increase *P*3 to reduce the startup delay. This algorithm replaces the returned random peer list that the server sends to the requested peer with a new list. This new list is constructed according to the NAT type of the requested peer. STUN in its simplest form allows the server in the streaming network to obtain the NAT type of any new connected peer. To construct this NAT-based list, peers behind disabled UDP configuration must receive a list that consists of type 1 peers. Peers with type 5 NATing will receive a list consists of type 2 as well as type 1. Type 3 and type 4 will be used to construct the list which will be send to peers with type 3 or type 4. Type 1 peers will receive lists consist of type 5 and type 6. Finally, if the peer is type 2, the list will consist of type 5, type 4 and type 3. Alg. 1 shows the pseudo code of how the server can construct these lists. This algorithm will diminish the playback start time. As we have mentioned, in [15], we can observe that there is a correlation between the location and the

NATing distribution. This can lead to reduction of peer's conductivities under AS-locality awareness in some locations or domains and to an increasing in others. Considering NATing as a parameter when construct the neighbor list, will allow the system to have a balanced peer's connectivity in all domains.

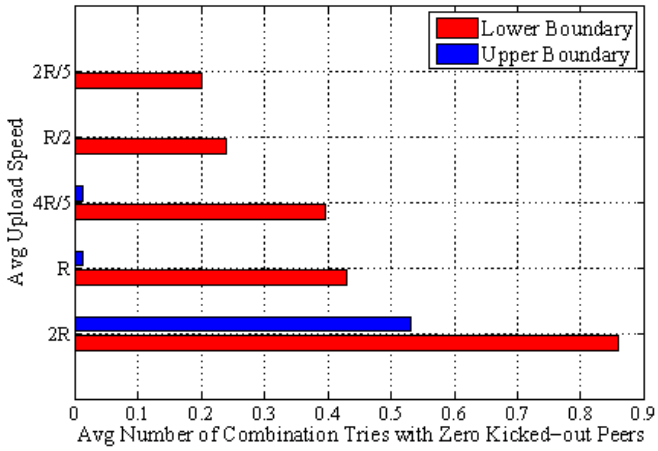

Fig.3: Zero Kicked-Out Peer Percentage combination tries to the Avg Upload Capacity

Algorithm 1 The Theocratical upper bound algorithm for Kicked Peers

$P(i)$: Peer with i NATing type
R: Video Rate in Kbps
$U_{p(i)}$: upload capacity of a Peer with NATing type i
i: NATing configuration 1=open, 2=port restricted, 3=symetric, 4=UDP disabled
K: Number of kicked out Peers
for all $Nodes\ in\ the\ sameAS$ **do**
  if $\Sigma U_{P(1)} >= (\Sigma P(4) * R)$ **then**
    $K = 0$
    $Available = \Sigma U_{p(1)} - (\Sigma P(4) * R)$
    if $(\Sigma U_{P(2)} + Avalible) >= (\Sigma P(3) * R)$ **then**
      if $\Sigma U_{P(2)} >= (\Sigma P(3) * R)$ **then**
        $K = 0$
        $Available = Available + (\Sigma U_{p(2)} - (\Sigma P(3) * R))$
      else
        $K = 0$
        $Available = (\Sigma U_{p(2)} + Available) - (\Sigma P(3) * R)$
      end if
    else
      $K = K + ((\Sigma P(3) * R) - \Sigma U_{p(2)})$
    end if
  else
    $K = K + ((\Sigma P(3) * R) - \Sigma U_{p(2)})$
  end if
end for

TABLE II
NAT CONNECTIVITY

TYPE NAME	TYPE	1	2	3	4	5	6
OPEN	1	Y	Y	Y	Y	Y	Y
FULL CONE	2	Y	Y	Y	Y	Y	N
IP RESTRICTED	3	Y	Y	Y	Y	N	N
PORT RESTRICTED	4	Y	Y	Y	Y	N	N
SYMMETRIC	5	Y	Y	N	N	N	N
UDP DISABLE	6	Y	N	N	N	N	N

7. THE EXPERIMENT

We used NS2 simulator [25] in our experiment. A pull-push streaming protocol was employed. We used a pull-push system for two reasons: first, tree based system with high streaming rate creates many nodes without children peers. Second, the mesh topology is good but it has a huge overhead and as we have recently experienced in the last few years the most well-known mesh

system (coolstreaming) has adopted a pull-push approach called (coolstreaming+). For our comparisons, we have implemented three algorithms to construct the neighbour list: the random, the AS-aware and the NAT types-aware algorithms.

In the AS-locality aware algorithm the list is constructed depending on the AS number. In NAT types-aware algorithm, the list's construction depends upon the algorithm that we have discussed in this paper. In the next section, we will discuss the streaming protocol that was implemented to be used in NS2 simulator.

7.1. The Streaming System

The streaming system consists of peers and a streaming server. When a new peer joins the system, it sends a request to the server which in turn answers with a list consisting of 20-50 peers. When the peer receives the list, it sends requests (to obtain the whole sub-streams lists) to the above number of peers. If the NATing type between the two peers allows them to connect, the peer then answers with a message that contains the latest sequence number for each sub-stream. If the NATing does not allow them to connect, the peer will not answer. The peer then selects the parents depending upon the sequence number. Subsequently, the peer sends a joining message to these nodes with the sub-stream number. If the peer is unable to obtain the whole stream from its connected peers, it will ask for a new list from the server (another round). When the peer receives the streams, it checks the quality of the system every five seconds. If the quality of any sub-stream is less compared to others, the peer disconnects the sub-stream and asks for the disconnected sub-stream from another parent. If the quality remains the same for three rounds, the peer disconnects the whole stream and asks for a new list from the server.

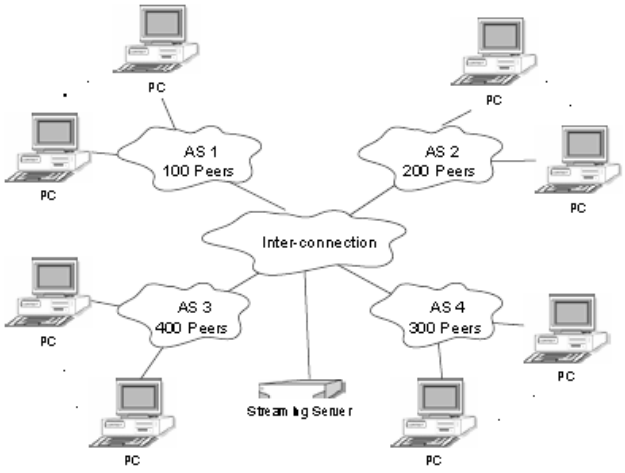

Fig 4: Simulation Topology

7.2. Topology and Parameters

Four ASes were located and connected to each other via a backbone as in figure 2.The number of peers in the first AS is 100, the second 200, the third 300 and the fourth 400, which gives 1000 peers in total. The effective upload capacity of these peers was chosen based upon a normal distribution from 120 to 650 Kbps. The NAT configuration we used in our simulation was similar to the one in [9]. Five hundred peers tried to join the system in the first second. After 1 minute the remaining peers connect. The rest of simulation parameters can be found in table III.

In the second configuration we changed the upper boundary of the upload speed from 650 to 1000 kbps. We kept the rest of the parameters as delay

TABLE III
SIMULATION PARAMETERS

Definition	Value
Routers	20
ASes	4
Video Rate (R)	500Kbps
Peers	1000
Video Duration	5 mins
Sub-Streams	10
Server BW	4500Kbps

8. RESULTS

The performance metric we used is divided into two categories. One category is for ISP Measurement while the second one pertains to the P2P system. Number of connected peers, playback start time including its Stability and the number of join messages are the metrics to measure the effects of P2P systems. Transit in, out and the number of exchanged packets in the same domain is the metrics to measure the load on the Internet Service Provider.

The number of connected peers at the theoretical upper and lower boundaries was counted. The connected peers were measured utilizing the simulation and the boundaries were computed using the algorithm as in Fig. 5. In these figure we can observe that the theoretical lower number of kicked out peers is 20% for both the random and the AS-aware algorithms and 22% for the NAT-aware algorithm. We can observe in the figure that the practical number of kicked out peers in the NAT aware algorithm is 26.5% which is close to the theoretical boundary. For the Random algorithm the expelled peer's percentage is 36.8% and for the upper boundary it is 46.7% which shows that 36.8% is in the middle between the two boundaries. In the AS-aware algorithm the number of expelled peers produced the highest value of 54.4% which is very close to the upper boundary of 56%.

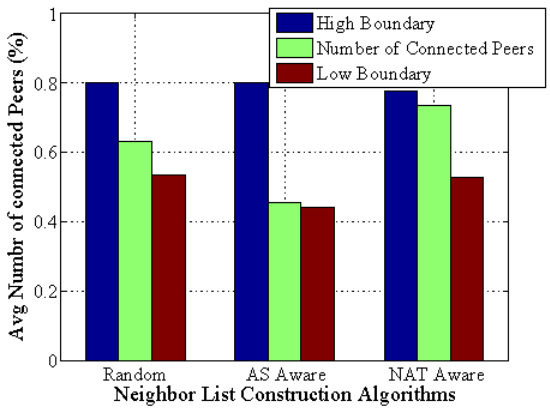

Fig.5: Avg Number of connected Peers

In Fig. 6, we measured the effects that pertain to the second AS which has 200 peers. From this figure we can conclude that the decreasing of transit traffic in, out and the maximum number of exchange packets in the domain is the optimum for the AS-aware algorithm. For the NAT-aware

algorithm, the reduction is higher than the Random Algorithm but it is higher than the Locality Aware one. The AS aware reduced of the transit traffic by 54% but for the NAT Aware algorithm the corresponding reduction was 35% and this represent a trade off between both algorithms.

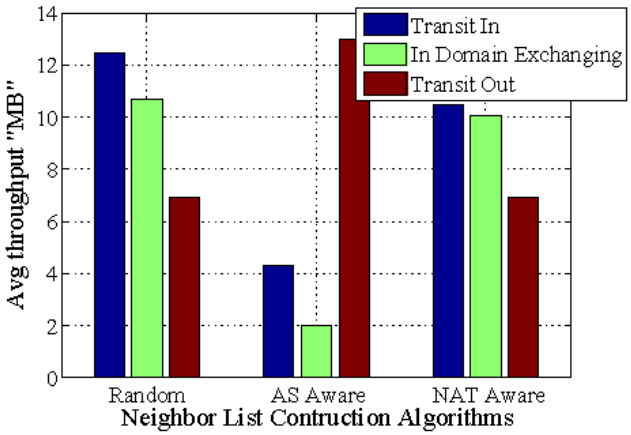

Fig. 6: Effects on ISPs

Fig. 7 shows the playback start time of the peers. From this figure we can observe the stability of the system with the NAT type-aware algorithm. Stability here means that all peers playback start time is around the average playback start time. In the figure the scattered points of both the random and the AS aware algorithms show that our algorithm is more stable than both of them. In the figure we used 100 as start time for nodes that tried all the simulation times to connect but they failed to do so.

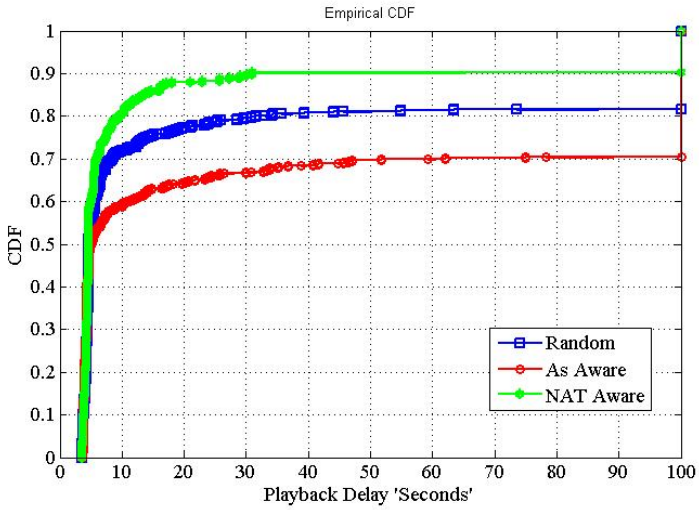

Fig. 7. Stability of the Algorithms

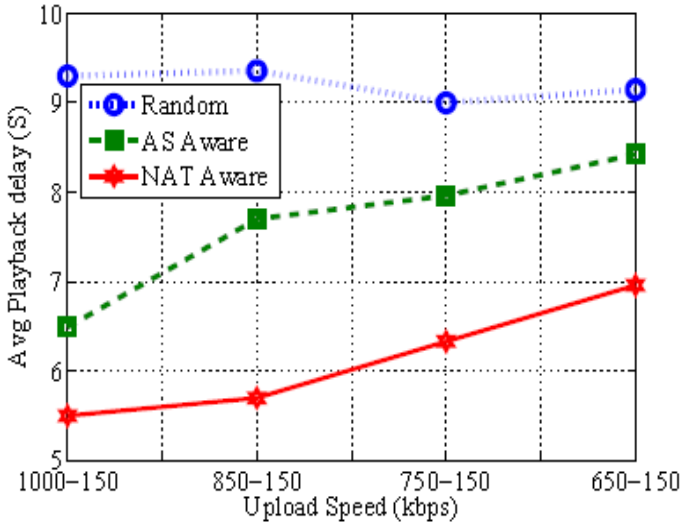

Fig. 8: Avg Start-up Time of the Whole System

Fig. 8 shows the average start up time of the whole system. Without doubt our protocol has the lowest average upload time for two reasons, first less number of rounds to connect as we will observe in Fig. 9. Secondly, NAT-aware algorithm build a full useful list not like the other algorithms which send a list with percentage of useful peers. Fig. 9 shows the average number of rounds that peers need to join the system. By reducing this number we are decreasing the playback start time and the overhead of the system. Our algorithm has decreased the average join messages but this number increased with the reduction of the upload speed. Most of the peers in our system use only one round. Peers that did not connect to the system have increased this number which in turn increased the average number.

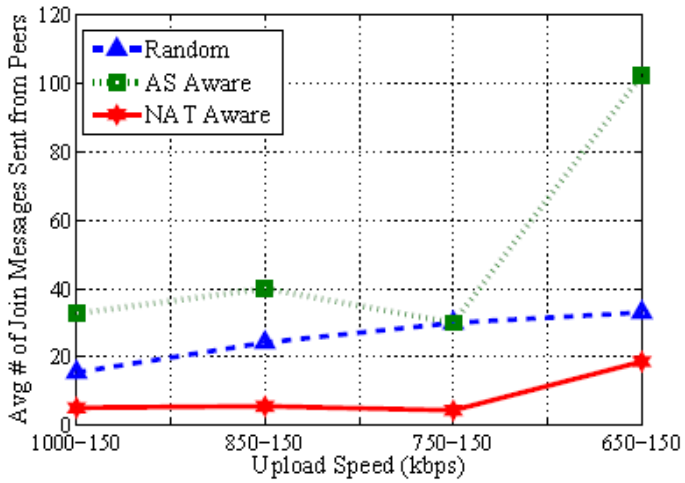

Fig. 9: Avg number of Join Messages

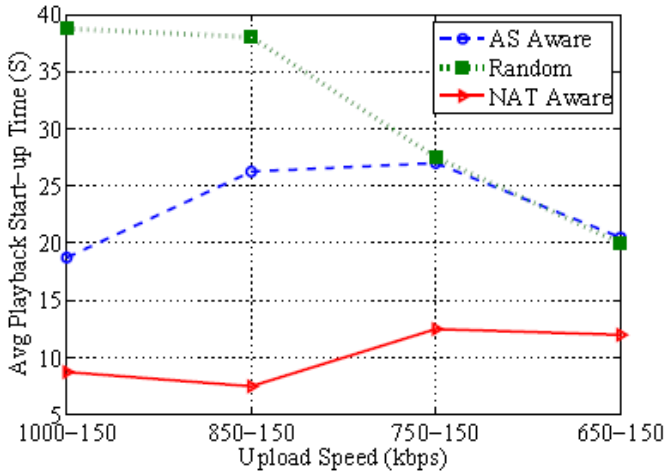

Fig. 10: Avg Playback start-up time of Peers with UDP-disabled NATing

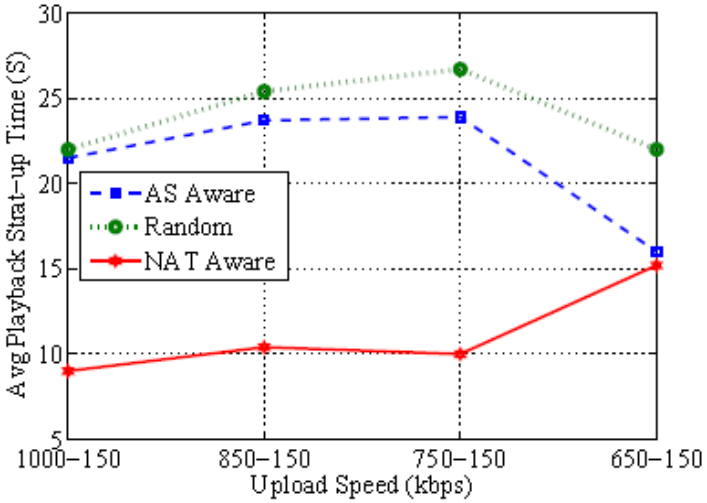

Fig. 11. Avg Playback start-up time of Peers with Symmetric NATing

Fig. 10 shows the average playback start time of nodes with UDP Disabled NATing. We can observe that the Random and the AS Aware algorithms did not reduce or even effect the playback start time of peers with these NATing configurations. Another misleading conclusion than can be obtaining from this figure is that the playback start-up time decreases when the average upload rate decreases. This reduction in playback start-up occurs because the numbers of connected peers that have been expelled from the system increases with the reduction of average upload speed. This lead to only few connected peers of this type that connected in the beginning

Fig.11 shows the average playback start-up time of symmetric NATing. This figure demonstrates that our algorithm can reduce this start-up time more efficient than AS aware and random algorithms. Another observation can be concluded from Fig.7 and 6 that the start-up time of symmetric NATing peers is less than UDP disabled NAting peers even when we use our algorithm. The reasons behind this are: first Symmetric NATing can connect with two types of NATing unlike UDP Disabled only one. Secondly the percentage of Port Restricted NATing is

higher than Open NATing. Our algorithm tried to keep the average playback start time close as possible to the average start up time of the network

9. CONCLUSION

NATing can make it difficult, if not impossible, for peers to connect to each other. In live streaming systems, peers can be expelled as a result. Locality aware solutions replace the random peer list with a selection mechanism that depends on the AS number of the peers when constructing neighbour lists. In this paper, we have shown that the AS-aware algorithm can satisfy ISPs need for reducing inter/intra-domain traffic with no benefits for P2P live streaming systems, when it is used with a high average upload capacity. We have demonstrated that the playback start-up delay time may increase by implementing locality awareness algorithms. We have analyzed the start-up delay in P2P streaming systems under NAT considerations. We proposed a novel and easy method to implement a selection algorithm to build a non-random peer list that increases peer connectivity and diminishes playback start-up time. We found that neighbour selection algorithms that are dependent upon AS numbers are not enough for P2P live streaming systems.

REFERENCES

[1] K. Kerpez, "Bandwidth Reduction via Localized Peer-to-Peer (P2P) Video," *OFC*, 2009.

[2] V. Hilt, I. Rimac, M. Tomsu, and E. Marocco, "A survey of research on the application-layer traffic optimization problem and the need for layer cooperation," *IEEE Communications Magazine*, august 2009.

[3] Lehrieder, F. Oechsner, S. Hossfeld, T. Despotovic, Z. Kellerer, and M. W. Michel, "Can P2P-users benefit from locality-awareness?" in *IEEE P2P*, Aug2010.

[4] IETF, "China telecom ALTO/P4P trial," 2009, http://www.ietf.org/proceedings/78/slides/alto-4.pdf.

[5] F. Picconi and L. Massoulie, "ISP friend or Foe? making P2P live streaming ISP-aware," in *Proceedings of the 2009 29th IEEE InternationalConference on Distributed Computing Systems*, ser. ICDCS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 413–422. [Online].Available: http://dx.doi.org/10.1109/ICDCS.2009.37

[6] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: provider portal for applications," in *ACM SIGCOMM*, 2008.

[7] M. Masoud, X. Hei, and W. Cheng, "Constructing a locality-aware isp-friendly peer-to-peer live streaming architecture," in *Information Science and Technology (ICIST), 2012 International Conference on*, March, pp. 368–376.

[8] H. Chang, S. Jamin, and W. Wang, "Live Streaming Performance of the Zattoo Network," in *ACM IMC*, 2009.

[9] K. Shami, D. Magoni, H. Chang, W. Wang, and J. S., "Impacts of Peer Characteristics on P2PTV Networks Scalability," in *IEEE INFOCOM*, 2009.

[10] I. Beijnum, "IPv4 Address Consumption," *The Internet Protocol Journal*, 2007.

[11] S. Ren, L. Guo, and X. Zhang, "ASAP: an AS-Aware Peer-Relay Protocol for High Quality VoIP," in *IEEE ICDCS*, 2006.

[12] B. Ford, P. Srisuresh, and D. Kegel, "Peer-to-Peer communication across network address translators," *ATEC*, 2005.

[13] X. WANG and Q. DENG, " VIP : A P2P communication platform for NAT traversal," in *Parallel and distributed processing and applications. Internationalsymposium No3*, Nov 2005.

[14] L. D'Acunto, M. Meulpolder, R. Rahman, J. A. Pouwelse, and H. J. Sips, "Modeling and analyzing the effects of firewalls and NATs in P2P swarmingsystems," in *IPDPS Workshops*, 2010, pp. 1–8.

[15] L. D'Acunto, J. Pouwelse, and H. Sips, "A measurement of NAT and firewall characteristics in peer-to-peer systems," in *Proc. 15-th ASCI Conference*,June 2009.

[16] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-VoD system," in *ACM SIGCOMM*,
August 2008.

[17] Y. Liu and J. Pan, "The impact of nat on bittorrent-like p2p systems," in *Peer-to-Peer Computing*, 2009, pp. 242–251.
[18] Z. Wei and J. Pan, "Modeling bittorrent-based p2p video streaming systems in the presence of nat devices," in *ICC*, 2011, pp. 1–5.
[19] L. D'Acunto, M. Meulpolder, R. Rahman, J. A. Pouwelse, and H. J. Sips, "Modeling and analyzing the effects of firewalls and nats in p2p swarming systems," in *IPDPS Workshops*, 2010, pp. 1–8.
[20] "Nat types." [Online]. Available: http://en.wikipedia.org/wiki/Network-address-translation
[21] "Understanding UPnP: a white paper," in *UPnP Forum*, June 2000.
[22] "UPnP device architecture v1.0.1 draft," in *UPnP Forum*, Dec. 2003.
[23] Internet Engineering Task Fource (IETF), "STUN - simple traversal of user datagram protocol (udp) through network address translators (nats),"RFC3489. [Online]. Available: http://www.ietf.org/rfc/rfc3489.txt
[24] W. Kho, S. Baset, and H. Schulzrinne, "Skype relay calls: Measurements and experiments," in *IEEE INFOCOM Workshops*, Apr 2008.
[25] "Network Simulator (NS2)," http://www.isi.edu/nsnam/ns/.