Gossip Peer Sampling in Real World

Amir H. Payberah (amir@sics.se)
Gossip Peer Sampling
Peer Sampling Service

- The peer sampling service provides each node with a list of nodes in the system.

- We would like that nodes are selected following a uniform random sample of all nodes in the system.
Gossip Peer Sampling Service

• One solution to achieve the uniform random selection is that every node knows all other nodes of the system.
  ▪ Not scalable

• Use a gossip-based dissemination of membership information to build an unstructured overlay.
  ▪ There are many variants of the basic gossip-based membership dissemination idea, but it is not clear whether any of these variants actually lead to uniform sampling.
Generic Framework

• First, a node Q is selected to exchange membership information with by node P.

• Node P pushes its view to Q.

• If a reply is expected, the view is pulled from Q.

• They merge their current view and the received one, and select a new view.
Gossip Protocol (1/4)
Gossip Protocol (2/4)
Gossip Protocol (3/4)
Gossip Protocol (4/4)
Design Space

- Peer Selection
  - Rand
  - Tail

- View Propagation
  - Push
  - Push-Pull

- View Selection
  - Blind
  - Healer
  - Swapper
Impact of NAT on Gossip Peer Sampling Protocols
Natted Gossip Protocol (1/4)
Natted Gossip Protocol (2/4)
Update State

Natted Gossip Protocol (3/4)

Update State
Natted Gossip Protocol (4/4)
Network Partition

View size: 15

View size: 27
Stale References
Randomness

![Graph showing the relationship between the percentage of NATs and the average percentage of non-self-named references. The graph includes two lines, one for a view size of 15 and another for a view size of 27, with data points representing the expected behavior.]
Classic NAT Types

- **Full Cone (FC):** The most permissive type of NAT.
- **Restricted Cone (RC):** Imposes restrictions on the IP addresses of external peers that can send messages to natted peers.
- **Port Restricted Cone (PRC):** Imposes restrictions on the IP addresses and ports of external peers that can send messages to natted peers.
- **Symmetric (SYM):** The most restrictive type of NAT.
NAT Types

- NATs differ in:
  - Way they assign public IP addresses (IP)
  - Way assign ports (Port)
  - Filtering rules (Filtering)
Classic NAT Types – FC

- **IP**: Same public IP to all sessions started from a given natted IP address and port.

- **Port**: Same port to all sessions started from a given natted IP address and port.

- **Filtering**: These sessions all share the same filtering rule, which states that the **NAT must forward all incoming messages**.
Classic NAT Types – RC

• **IP:** The same as FC.

• **Port:** The same as FC.

• **Filtering:** The sessions started from a given natted peer’s IP address and port towards a **target IP address**, share the same filtering rule: the **NAT device only forwards messages coming from this IP address.**
Classic NAT Types – PRC

- **IP**: The same as FC.

- **Port**: The same as FC.

- **Filtering**: The sessions started from a given natted peer’s IP address and port towards a target IP address and port, share the same filtering rule: the NAT device only forwards messages coming from this IP address and port.
Classic NAT Types – Symmetric

- **IP**: The same as FC.
- **Port**: Different port for each session started from a given natted IP address and port.
- **Filtering**: The same as PRC.
NATCracker Perspective

- **Mapping policy:** Decides *when* to bind a new port.
  - Endpoint Independent (EI)
  - Host Dependent (HD)
  - Port Dependent (PD)

- **Allocation policy:** Decides *which* port should be bound.
  - Port Preservation (PP)
  - Port Contiguity (PC)
  - Random (RD)

- **Filtering policy:** Decides whether a packet from the outside world to a public endpoint of a NAT gateway should be forwarded to the corresponding private endpoint.
  - Endpoint Independent (EI)
  - Host Dependent (HD)
  - Port Dependent (PD)
NAT Traversal Techniques

- Hole punching (UDP)

- Relaying
  - When the destination node is behind a SYM NAT and the source node is either behind a PRC NAT or a SYM NAT.
  - When the destination node is behind a PRC NAT and the source node is behind a SYM NAT.
NAT Traversal Techniques – Hole Punching (UDP)
NAT Traversal Techniques – Relaying
Three Proposed Solutions
ARRG: Real-World Gossiping

Niels Drost, Elth Ogston, Rob V. van Nieuwoort and Henri E. Bal
Vrije Universiteit Amsterdam

(HPDC'07)
Design Space

- Peer Selection
  - Rand
  - Blind

- View Propagation
  - Push
  - Push-Pull

- View Selection
  - Blind
  - Healer
  - Swapper
The ARRG Protocol

- **Actualized Robust Random Gossiping (ARRG).**

- It uses **Fallback Cache** to solve the network connectivity problem.

- The Fallback Cache acts as a **backup** for the normal membership cache present in the gossiping algorithm.

- Each time a **successful gossip exchange** is done, the target of this gossip is added to the Fallback Cache.

- Whenever a gossip attempt **fails**, the **Fallback Cache is used** to select an entry to gossip with instead of the one selected by the original algorithm.
Example (1/4)

\[ n_1 \xrightarrow{\text{gossip}} n_2 \xrightarrow{} n_3 \xrightarrow{\text{Fallback Cache}} n_4 \]
Example (2/4)

Diagram showing network topology with nodes n1, n2, n3, and n4. The diagram illustrates a reply path and a fallback cache.
Example (3/4)

n1 — gossip — n2

n2 — gossip — n3

n3 — gossip — n4

Fallback Cache
Example (4/4)

The diagram illustrates a network topology with nodes n1, n2, n3, and n4. Node n2 is highlighted and connected to other nodes through gossip communication. The diagram also includes a Fallback Cache, indicated by a red arrow pointing towards n2.
NAT-resilient Gossip Peer Sampling

Anne-Marie Kermarrec, Alessio Pace, Vivien Quema, Valerio Schiavoni
INRIA - CNRS

(ICDCS'09)
Design Space

- Peer Selection
  - Rand
  - Blind

- View Propagation
  - Push
  - Push-Pull

- View Selection
  - Blind
  - Healer
  - Swapper
The Nylon Protocol

- The main idea of Nylon is to implement reactive hole punching.
- A peer only performs hole punching towards peers it gossip with.
- Hole punching is implemented using a chain of RVPs that forward the OPEN HOLE message until it reaches the gossip target.
The Nylon Protocol

• Each node maintains a **routing table** that maintains the mapping between a natted node from its view and its associated RVP.

• For each node $P$ in the routing table, the RVP is the node it shuffled with to obtain the reference to $P$.

• RVPs do **not proactively** refresh holes.
  - Therefore, a time to live (**TTL**) is associated to each RVP entries in routing tables.
Example (1/3)

Hole punching

n1 and n2 become RVP for each other.

<table>
<thead>
<tr>
<th>rule</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n2: allow</td>
<td>120</td>
</tr>
<tr>
<td>Others: deny</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dest</th>
<th>RVP</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n2</td>
<td>-</td>
<td>120</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rule</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1: allow</td>
<td>120</td>
</tr>
<tr>
<td>Others: deny</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dest</th>
<th>RVP</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>-</td>
<td>120</td>
</tr>
<tr>
<td>n2</td>
<td>-</td>
<td>120</td>
</tr>
</tbody>
</table>
### Example (2/3)

n2 and n3 become RVP for each other.

<table>
<thead>
<tr>
<th>dest</th>
<th>RVP</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n2</td>
<td>-</td>
<td>120</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dest</th>
<th>RVP</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>-</td>
<td>120</td>
</tr>
<tr>
<td>n3</td>
<td>-</td>
<td>120</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rule</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n2: allow</td>
<td>120</td>
</tr>
<tr>
<td>Others: deny</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>rule</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1: allow</td>
<td>120</td>
</tr>
<tr>
<td>n3: allow</td>
<td>140</td>
</tr>
<tr>
<td>Others: deny</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dest</th>
<th>RVP</th>
<th>TTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>n2</td>
<td>120</td>
</tr>
<tr>
<td>n2</td>
<td>-</td>
<td>120</td>
</tr>
<tr>
<td>n3</td>
<td>-</td>
<td>120</td>
</tr>
</tbody>
</table>
Example (3/3)

Through this chain n3 can shuffle with n1. n3 performs hole punching toward n1 by sending an OPEN_HOLE message to n2 that will forward it to n1.
Balancing Gossip Exchanges in Networks with Firewalls

Joao Leitao, Robbert van Renesse, Luís Rodrigues
INESC-ID/IST - Cornell University

(IPTPS'10)
Design Space

- Peer Selection
  - Rand
  - Blind

- View Propagation
  - Push
  - Push-Pull

- View Selection
  - Blind
  - Healer
  - Swapper
  - ?
The Protocol

• Each node maintains:
  ▪ A quota value (initially with a value of 1).
    • Nodes increase their quota when they initiate a gossip exchange.
  ▪ A single-entry cache for connections created by other nodes.
    • The connection cache keeps alive the last connection used by another peer to initiate a gossip exchange.

• When a node receives a gossip request, engages in gossip exchange if:
  ▪ Has a quota value above zero.
  ▪ Has an empty connection cache.
  ▪ The gossip message has been already forwarded TTL times.
Example (1/8)
Example (2/8)
Example (3/8)
Example (4/8)
Example (5/8)
Example (6/8)

{n1} → gossip → {n2} → gossip → {n3}

n1
2

n2
0

n3
2

gossip

Amir H. Payberah – 22 June 2010
Example (7/8)
Example (8/8)
Question?