Design of Anonymous Publish-Subscribe Messaging System in a P2P Network Architecture

Sushil Paneru[1], Dr. R. PADMAVATHY[2]

[1,2]*Department of Computer Science and Engineering, NIT Warangal*

---***---

Abstract - *Publish-subscribe is a popular asynchronous message exchange pattern that is widely used in distributed systems. Apache Kafka, RabbitMQ are some of the widely used centralized implementation of publish-subscribe message exchange system, which provides scalable and robust messaging service to connect various distributed services. Decoupling of publisher and subscriber is the nuance of publish-subscribe messaging pattern, but the aforementioned centralized implementations does not provide anonymity to publishers and subscribers i.e. an intruder having access to broker – the central entity — can find out the identity of publishers and subscribers. Thus, we present a design that masks the identity (IP address) of actors with virtual addressing — just as systems like Bitcoin and decentralizes the broker by relying on P2P network architecture to avoid single point of failure along with provision of load distribution for message routing. Our proposed design trades off low latency with anonymity, which is conducive to building an anonymous publish subscribe messaging system that ensures concealed identity of users along with data confidentiality and data integrity. Application of such anonymous system would be to construct a communication channel, wherein user can contact each other and exchange data while protecting their identities from each other and from outside parties, also such model would further allow governments to take part in covert operations or news agencies to obtain sensitive and ground breaking information secretly from a source.*

***Key Words*: Peer To Peer, publish subscribe, anonymity, decentralization, virtual address routing, bloom filter**

1. INTRODUCTION

With the shift in paradigm from centralized to decentralized systems in recent years, there arose a need for robust, simple and asynchronous way to exchange messages between the distributed services. Point to point messaging between services using HTTP/UPD/TCP turned out to be difficult with increase in the number of services as tremendous amount of effort was required to maintain an effective and simple communication between systems. Thus, in the process of easing out hurdles involved during message exchange in a distributed system, system designers opted for publish-subscribe pattern where an external agent simplifies the complexity in communication and makes the task of building a distributed application easier. Publish-subscribe is an asynchronous message exchange pattern where a sender, also called as publisher, does not send the message directly to the recipient, also termed as subscriber, but instead the message is sent to an entity called broker which then forwards the message to all the interested subscribers. Thus, such message exchange pattern introduces broker, an external agent, which decouples sender from receiver and enables many to many communication with minimal effort.

Along with benefit offered by pub-sub pattern like easing out the process of message dissemination, it has its own share of security issues like authentication, anonymity of agents, availability and confidentiality. A pub-sub network without any encryption scheme can leak out the details of published data to other peers (no confidentiality); without any message or peer authentication, the system lacks data and system integrity. Without anonymity to the peers, the role of an agent, publisher or subscriber, can also be leaked out. This paper intends to present a solution to provide anonymity to the agents along data confidentiality and integrity.

1.1 Related Works

Most of the existing work in the field of pub-sub network have focused on scalability, performance; some of them have provided approach for data confidentiality and integrity and very few have talked about anonymity of the peers. One of the most popular pub-sub message exchange system is Scribe [3], which introduced the use of spanning tree consisting of subscribers for efficient data dissemination. Scribe depends on the features of Pastry P2P network [4] which is considered to be scalable but it cannot provide anonymity to the users. One of the popular work in the field of pub-sub is by Mudhakar Srivatsa, Ling Liu [1] which builds a modular framework for providing confidentiality and integrity but not anonymity. One of the publish-subscribe systems that aims to provide anonymity is by Datta et al [5]. They propose a routing system based on maintaining multiple layers of weakly connected directed acyclic graphs. In this system, one or more sink nodes, which may change over time, become dissemination points receiving all publications and forwarding them to subscribers. However, anonymity is provided by assuming that the node from which a receiver gets a message may not be the original publisher. However, an adversary would still know that that node could possibly be an original publisher. Without probabilistic analysis of this possibility, it is difficult to say how well protected the publishers actually are. Also, no mention is made as to how difficult it is to identify subscribers in the system, and this has not been implemented. The other publish subscribe system is the work by Binh Vo and Steven M. Bellovin [6]. They propose the use of Tor hidden service [7] as an anonymous network which will handle the work of hiding the true identity of users and route messages by constructing spanning tree for each topic among the peers. The approach to use Tor hidden service is quite slow and expensive because it includes maintaining relay nodes for tor and multi-layered encryption and decryption whereas there is single layered encryption and no involvement of relay nodes in our system and also it is to be noted we use virtual address routing for hiding peers identity instead of Tor hidden service. Also, many attacks have been successfully made on Tor network and it has been broken by many systems in the past

1.2 Objective

This paper aims to provide a set of designs that anonymizes the network along ensuring with data confidentiality and integrity. When we mention anonymity, we mean about hiding the IP address of the peer since the sole identity of a user on internet is its IP address. Just like in bitcoin, an anonymous system, **virtual addressing** is introduced to mask the real identity of the user; all communication is performed by addressing the packets to virtual address of the receiver peer. With the use of virtual address, there comes a need to route the packets on the basis of virtual address; therefore, we propose a routing protocol, which is a variant of **distance vector routing** algorithm that trades off speed of packet transmission with anonymity. We also use a centralized service coordinator that helps bootstrap the network and intelligibly implements key management without jeopardizing the anonymity property of the network. A pub-sub network can be content based or topic based, but in this paper we focus on topic based network for simplicity.

2. PRELIMINARIES

2.1 Characteristics of P2P Network

The underlying P2P network on which we pass messages is unstructured, very much like Bittorrent and Naspster. A central service coordinator is used to track the global view of the P2P network and helps new peers join the network. The central service coordinator ensures the degree of connectivity of peers i.e. a peer is connected to how many other peers; the reason behind this strategy is to nullify the effects of peer churning, which can debilitate the network. Even though the central service is compromised, the anonymity of the network won't be at stake since the coordinator does not contain any data about the mapping of virtual address and the IP address of the peers. The service coordinator also tracks the availability of the peers via sending heartbeat — a basic UDP packet — to the peers and waits for the response to check if the peer is alive or not. Also, the underlying message transport protocol is UDP which offers the advantage of spoofing the sender field of the packet.

2.1 Context of Anonymity

The main crux of this work is introducing anonymity. To start, anonymity in our context should be defined. In this paper, anonymity refers to the following three situations:

- An eavesdropper (i.e. a middle man) should not be able to know the identity (IP address) and role of peers during message exchange (i.e. whether they are a publisher or a subscriber).
- A forwarder peer (a peer who helps in message dissemination) should not be able to know about the content of the messages that it is forwarding to other peers.
- A publisher should not know the identity of subscribers. Similarly, subscribers should not know about the identity of the publisher.

2.3 Degree of Anonymity

We define two kinds of system, open and closed, based on the how the peers are allowed to communicate and to what level of anonymity and other securities are provided.

For an **open system**, anyone can join the network and a person can read any other person's messages. An example of this would be a civil rights activist wanting to communicate to the masses in order to protest about Civil Rights Violation during a state of emergency (e.g. Arab uprising in Lebanon, Egypt, Syria, etc.), but is afraid that his/her identity will get leaked. This system will have anonymity and message authentication but no publisher authentication as without external authentication any peer can act as publisher

On the other hand in a **closed system**, only those who have already obtained a secret key by some external means can join the network and communicate. For example, a source wants to leak sensitive information to a group of newspapers but want their identity to be concealed during this exchange of information, and the interested parties should have the secret key to participate during message exchange. Here, a closed network is going to be used. Unlike open system, introduction of an external secret key ensures the authentication of publisher i.e. the subscribers are ensured they will receive the message from the publisher who has the secret key. Thus, closed system is supposed to have message authentication, publisher authentication and message confidentiality.

3. METHODOLOGY

3.1 The Role of Service Coordinator

The service coordinator maintains the global state of the network; for example, it tracks of the alive nodes (via receiving heartbeat from nodes) on the network and also information about to how many other peers a node is connected to so that it can minimize the risk caused by peer churning — for example, prevention of formation of articulation point which can bipartition the network. So, when a node wishes to join the network, it has to contact service coordinator which, in response, sends a list of peers that the new node can connect to. The list of peers is selected in such a way that after connection, each node in the network is connected to some minimum number peers so that effects of peer churning is minimized. Thus, service coordinator helps maintain the network to be in a healthy state.

The service coordinator also act as certificate authority (CA). A peer can register its public key by providing its virtual address (further explained in section 3.2) and its public key. The CA accepts such key registration requests (or can also be called as certificate generation) only when the hash of public key matches with the virtual address of the node. Non-collision property of hash functions like SHA-256 ensures that a bad peer can't fool service coordinator to tamper with the certificate directory. Similarly, whenever a peer wants to get the public key of another peer, it has to query the service coordinator with the virtual address of the peer whose public key it wants. The queried certificate will be signed by the service coordinator whose certificate (signed by some root CA like Godaddy) will be made available to a peer at the time of joining the network. Thus, service coordinator also acts as a trustful entity in the network who can be relied upon for distribution of public key.

It can be argued that when service coordinator gets compromised, data can be leak out which can benefit a attacker to map the virtual address of a peer to its IP address. Except during peer's certificate registration, a peer does not have to provide its virtual address to service coordinator. The solution to this scenario is that, at the time of certificate registration, a peer spoofs the origin field of the UPD packet — unlike TCP, no handshake is required — which it sends to the CA. The reason that UDP is used at this point of contact with service coordinator is to spoof the real identity to service coordinator, which cannot be done with TCP since handshake is required. Thus, even if the service coordinator gets compromised at the time of certificate registration, the IP address that the attacker observes will be a spoofed one. Hence, the real identity of a peer remains secret.

But a problem with service coordinator is that it is also a single point of failure. This can be mitigated by keeping another service coordinator in stand-by mode for a master slave configuration, where the slave service coordinator's state will continuously be updated by the master.

3.2 Virtual Addressing

Usage of IP address in the standard pub-sub model makes it non-anonymous because messages can be traced easily and identities of peers can be revealed, unless some proxy or VPN is used. In fact, even the usage of VPN and proxy can reveal the identity of the user by accessing the central service provider, and this is standard procedure followed by government agencies. So, just like in systems like bitcoin, we introduce the usage of virtual address for addressing peers instead of IP addresses. Virtual address or **nodeId** is basically a hash of public key which the peer chooses for making its communication with other peers confidential and authenticated. Rather than fixing the usage of a particular hash function, we let the user, as per their choice, choose the hash function. This allows the system to evolve with the any kind of development of hash function in the community of cryptography. Hence, a nodeId will also contain a short header, along with the hash digest, that specifies the type of hash function used and the length of hash digest. An example is shown below:

[Hash function code][Length of hash digest][Hash digest]

Since we are using virtual address to identify a node, we will need a way to route the packets in the network. Therefore, we have come up with a variant of distance vector routing algorithm, which is the algorithm used by routers to route the packets over internet. Virtual address routing basically help determine the sub-optimal path to route the packet to a destination without revealing identity of the peers involved. Similar to distance vector routing, each node maintains a table, as shown below in table 3.1, which has a row for each destination peer and a column for each of its directly attached neighbors; with the help of routing table, a node can only know the direction to destination, but not the whole path. In other words, a node will only know to which neighbor to send the packet for effective routing. Unlike distance vector routing, instead of distance, a node stores timestamp of the packet received from the neighbor which was sent to that node from sender node; timestamp will help decide the neighbor (direction) that allows sub-optimal routing in future. Until the routing table does not have entry for a destination peer, it can't determine to which neighbor forward the packets to. So, packets are sent to all the neighbors i.e. broadcast until the packet reaches a node which has the entry for destination, and it can happen that the packet reaches all the way to destination during the broadcast. Also, to preclude the flooding of network, a unique message id (hash of the message packet) is generated for each message; so, that the already seen packet's circulation can be stopped. This is further explained in section 3.5. An example is given below to explain how a node learns to construct routing table from the message packets being circulated in the network.

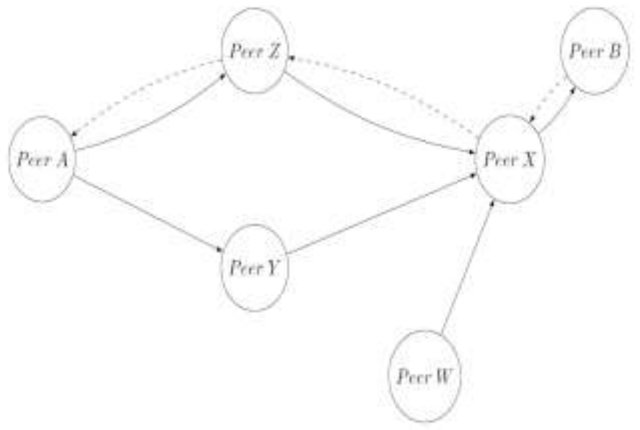

Figure –1: Routing among peers, solid line represents connection between the peers, dotted line represents path a message was routed through

Table –1: Routing table of peer X

Destination	Neighbor peers (timestamp of message received)		
	Peer Z	Peer Y	Peer W
Peer A (virtual address)	5	8	-1

In Figure 3.1, let peer X received messages originated from Peer A through two of its neighbors (Peer Y and Peer Z). Peer X would then populate its routing table as shown in Table 3.1. For populating the routing table, timestamp of the event when the message was received is stored, and -1 if the node hasn't ever received a message from a neighbor. Later, at some point of time, if Peer X receives a message from Peer B, it checks the destination's Virtual Address inside the message header. If the destination address happens to be Peer A's virtual address, Peer X then knows that it could forward the message to either Peer Z or Peer Y. Peer X then chooses the one from which it received the message earliest i.e. entry with the least timestamp value, in this case peer Z. So, in Virtual Address Routing, each peer maintains its routing table by learning from the packets being routed in the network.

3.2.1 Publish Subscribe Events

When a peer wants to subscribe to a topic, it broadcasts a subscription request query all over the network and the corresponding publisher and other subscribers of the same topic will store the virtual address and in response, publisher will send a dummy message along with its signature. Such broadcast is supposed to flood the network, for which we have come up with a solution, which is explained in section 3.5. Similarly, when the publisher intends to publish a message, it disseminates the message via gossip protocol (explained in section 3.4) along with its signature; the signature ensures message has not been tampered. The underlying routing happens as explained in section 3.2. Also, since the underlying transfer protocol is UDP, there is no need for exposing the source IP address unlike TCP, and thus, anonymity is attained by trading off reliability.

Thus, with virtual address routing, we construct a sub-optimal path — which no peers knows about — to replace the need for a centralized broker, and also since every peer can act as a broker, the task of routing packets is distributed amongst all the nodes in the network.

.3 Encryption and Authentication

Along with anonymity, we aim to provide message confidentiality and message authentication. Public Key Encryption and symmetric key encryption is used for maintaining message confidentiality and message authentication so that a forwarder peer (P2P broker), who happens to be an attacker, won't be able to know content of the message and the message receiver will be assured of its source.

In a **Closed Network**, the secret key, which is obtained from external sources, will act as the key for symmetric encryption. Since, authenticated subscribers in a closed network also have the secret key from an earlier point in time, only they can only decrypt the message. The secret key is hashed with SHA-256; first 128 bits of the key is used for symmetric encryption (AES) and second 128 bits are used for message authentication (HMAC).

$$K_{0\text{-}255} = SHA_{256}(K_s)$$
$$m' = AES(m, K_{0\text{-}127})$$
$$c = HMAC(m, K_{128\text{-}255})$$

In an **Open Network**, where there is no restriction on users while joining the network, message confidentiality cannot be provided because any peer can join the network and view the message no matter how we encrypt it. Message authentication can be provided in open network by signing the message with the private key of the publisher. But this does not ensure publisher authenticity because in a system without central authority, any node can act as publisher.

3.4 Message Dissemination

A efficient way to disseminate messages to the subscribers is to use Gossip[13] protocol shown in Fig 3.4. The protocol is inspired by the form of gossip seen in social networks. A gossip spreads in a social network when it starts as a conversation between two parties. These two parties then tell two more parties each and this message spreads across the entire network. When one party hears the message for a second time, it simply identifies the message and ignores it because it has already received the message from someone else.

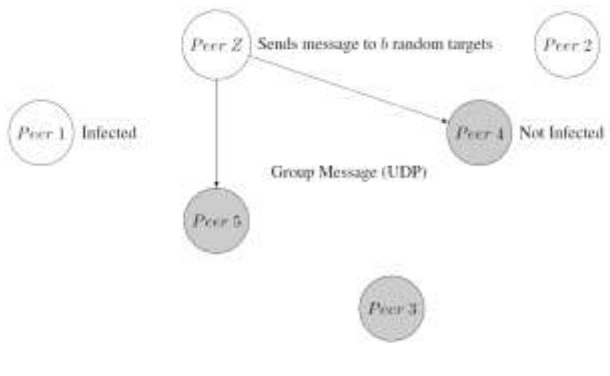

Figure 3.4 – Gossip protocol

When a new message is to be disseminated, the publisher selects b number of subscribers from its list and sends messages to all of them. Now these randomly selected subscribers send the message to randomly selected subscribers since each subscriber will also have list of subscribers for its topic. When all the subscribers have received the message, the dissemination process stops. On average, in $\log_b N$ steps is required to disseminate the message to all the nodes, where b is the number of subscribers randomly selected and N is the total number of nodes in the network.

3.5 Message Flooding Control

Message flooding is a very common problem in a P2P network. In our case, there are two scenarios where network might get flooded. First, whenever a subscriber needs to make subscription for a topic, it broadcast the subscribe event in the network. While request message is traveling across the network, the message can continuously keep coming to one or more than one peers. Such messages will keep circulating in the network for an infinite amount of time and unnecessarily waste resources. Second, whenever a publisher publishes a message, it sends it to a random set of forwarders (set of peers that routes). The forwarding peers might receive repeated messages, so the forwarding peer must be able to discard the repeated messages it receives so that network resources are not wasted.

The problem of message flooding in a P2P network can best be solved by Bloom filters [9]. Bloom filter is a simple and elegant data structure that allows us to find out if an element is present in a set or not, given the possibility of false positive result. Bloom filters can efficiently check if a message, identified by a message id, has been seen before or not. Every time a message is received by a peer, it checks if it has already seen the message by querying the bloom filter. If the node has not dealt with the message previously, then it adds the message's id into bloom filter and process the packet further. Bloom filter can do addition and lookup in O(k) time complexity, where k is number of hash functions and each hash function work in constant time. Similarly, the space consumption is also very low because instead of storing whole message id, we are storing the hashed value of a bit size.

In our network, thousands of messages might have to be tracked and false positive scenarios of duplicate message check can be ignored hence, usage of such efficient and lightweight data structure helps saving a lot of computation and memory for each node in the network.

3.6 UDP Packet Spoofing

Because of virtual addressing of the nodes, TCP connections cannot be made because without IP address, three way handshake protocol will not work. Therefore, we decided to use UDP as the transport protocol. For our context, UDP provides a helpful feature for ensuring anonymity i.e. spoofing the UDP packets with false source IP addresses. When a publisher publishes a message or a subscriber subscribes, it can change the source IP address of the packet and hence, it becomes even more difficult for an attacker to map the virtual address to IP address. The choice of UDP over TCP comes with a cost, and that is, message transfer is not reliable for a UDP packet.

4. IMPLEMENTATION AND RESULT

The distributed nature of the application adds difficulty for the implementation and testing. It is infeasible to code and continuously setup a real distributed environment to test. So, we needed a tool that allowed the simulation of a real like distributed environment. Virtual machines would have a solution to our problem but it consumes too much of the host machine's resources and slows down the performance of the system which is being used for development. Linux containers are a better option compared to virtual machines because they provide same set of features like isolation and resource control but are very lightweight compared to Virtual Machines. A Python library Pycrypto, which has been thoroughly tested for security, has also been used to for cryptographic and hashing functionalities.

For initial testing, we set up 20 containers to simulate as peers and formed a network of nodes in different permutations and combinations. We used network monitoring tool Wireshark to analyze the packets in the network, and also we wrote MITM scripts for some nodes to act as attackers for tampering the packets. After completion of the implementation on real environment, we tested on 10 machines each having 8 processors (1.1 GHz Intel Pentium IV Xeon processors on Ubuntu 14) connected via a high speed LAN. Similar to initial testing, we tested on different topology of the network, and found that the communication between nodes was anonymous and confidential. In the packet analysis, we were unable to find the IP address of the node who sent it and who received it. Being a P2P network, latency of message transfer was high compared to centralized implementation (Kafka, RabbitMQ), which was expected as a tradeoff for anonymity.

5. FUTURE WORK

The central service coordinator is at high load given that fact that it has to maintain the global state of the network. Maintaining the alive status of all nodes can get hectic for a single central coordinator. As part of future work, we plan to shard the network and assign the task of maintaining the global state to multiple service coordinator, and these new instances will work together to maintain a global state.

This work has not been subjected to a stringent performance analysis. Being a P2P network and having a path of communication between nodes that constantly changes over time, it's difficult to come with statistics for latency and throughput; thus, we plan to devise strategies for performance analysis.

6. CONCLUSION

We have a presented a message exchange system that calculates a secret sub-optimal path of communication and allows entities to communicate without exposing their real identity i.e. IP address. The idea of virtual addressing enabled us to mask the identity of nodes and the idea of Gossip protocol and virtual address routing enabled us to disseminate the messages by concealing the identities. Also, the usage of central service coordinator facilitated the maintenance of global state of the network to prevent network partition and ensure stability of the network. Most of the research in the field of pub-sub network has focused on performance and security but very little effort has been made in introducing the dimension of anonymity. Thus, we extended the pub-sub network to achieve anonymity to a substantial extent.

REFERENCES

[1] Mudhakar Srivatsa, Ling Liu, “Securing publish-subscribe overlay services with eventguard” in CCS 05 Proceedings of the 12th ACM conference on Computer and communications security

[2] Joan Daemen and Vincent Rijmen, “The design of Rijndael: AES-the advanced encryption standard” in Springer Science & Business Media, 2013.

[3] Miguel Castro, Peter Druschel, A-M Kermarrec, and Antony IT Rowstron, “Scribe: A large-scale and decentralized application-level multicast infrastructure” IEEE Journal on Selected Areas in communications, 20(8):1489–1499, 2002.

[4] Antony Rowstron and Peter Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems” inn IFIP/ACM International Conference on Distributed Systems Platforms and Open Distributed Processing,Springer, 2001.

[5] Ajoy Kumar Datta, Maria Gradinariu, Michel Raynal, and Gwendal Simon, “Anonymous publish/ subscribe in p2p networks” in Parallel and Distributed Processing Symposium, 2003.

[6] Binh Vo and Steven Bellovin, “Anonymous publish-subscribe systems” in International Conference on Security and Privacy in Communication Systems,Springer, 2014.

[7] Roger Dingledine, Nick Mathewson, and Paul Syverson. “Tor: The second-generation onion router” in Technical report, DTIC Document, 2004.

[8] Mesut Gunes, Udo Sorges, and Imed Bouazizi, “Ara-the ant-colony based routing algorithm for Manets” in Parallel Processing Workshops, 2002. Proceedings. International Conference on, pages 79–85. IEEE, 2002.

[9] Burton H. Bloom “Space/time trade-offs in hash coding with allowable errors - BLOOM – 1970” in Communications of the ACM Volume 13 Issue 7, July 1970

[10] Andr´e Allavena, Alan Demers, and John E Hopcroft, “Correctness of a gossip based membership Protocol” in Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing, ACM, 2005

[11] Kevin Bauer, Damon McCoy, Dirk Grunwald, adayoshi Kohno, and Douglas Sicker, ”Lowresource routing attacks against tor” in Proceedings of the 2007 ACM workshop on Privacy in electronic society, pages 11–20. ACM, 2007.

[12] Timothy G Abbott, Katherine J Lai, Michael R Lieberman, and Eric C Price. Browser-based attacks on tor” in International Workshop on Privacy Enhancing Technologies, pages 184–199. Springer, 2007.

[13] Dirk Merkel,”Docker: lightweight linux containers for consistent development and deployment” in Linux Journal, 2014(239):2, 2014.