P2P Interim Report

Group Members:
Hamza Zafar
Muhammad Bhatti
Course: Peer to Peer Systems and Security
Project Module: Gossip
Group Number: 35

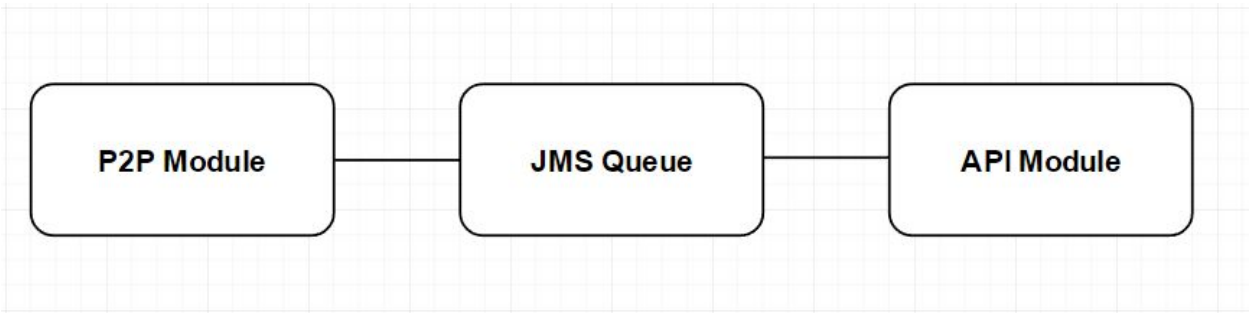
Process architecture

Since we are using Java NIO for network communication, our process architecture is inherently based on event loops.

Inter-module protocol

Firstly, one of the most important role in our gossip module is that of the bootstrap server. In short, a bootstrap server allows new peers joining a swarm to get information about some peers in the network. The bootstrap server maintains a list of all online peers, and assigns them an ID. Whenever a new peer joins the network, it registers itself with the bootstrap server. The bootstrap server assigns this peer a unique ID, and sends it a list of some peer, depending on the DEGREE it specified. This new peer now has knowledge of some of the peers in the network, and connects to them. Since all peers participate in sharing information about their neighbors, the new peer gets known across the network/swarm, and learns about new peers.

The Gossip module, running on any peer, wishes to know who its neighbors are connected to. There are periodic heartbeat messages, in which each peer shares information about their neighboring peers to the peers they themselves are connected to. In this way, a new peer who joins the swarm, is easily able to find out the peers that exist in the given network, and will be able to find a network on its own, depending on the value of its DEGREE.



The above diagram shows how the P2P and API modules communicate with each other, which is through Java Messaging service queues. The JMS queue signifies the messages that have been sent, by either module, and are waiting to be read. As the name queue suggests, the messages are delivered in the order sent. A message is removed from the queue once it has been read, and every message successfully processed is acknowledged by the consumer.

Message Formats

Module Message Type	**Message Code**
GOSSIP ANNOUNCE	500
GOSSIP NOTIFY	501
GOSSIP NOTIFICATION	502
GOSSIP VALIDATION	503
Gossip internal Messages	**Message Code**
GOSSIP INIT	510
GOSSIP POPULATE	511
GOSSIP CONNECT	512
GOSSIP DISCONNECT	513
GOSSIP NEIGHBORS	514

INIT

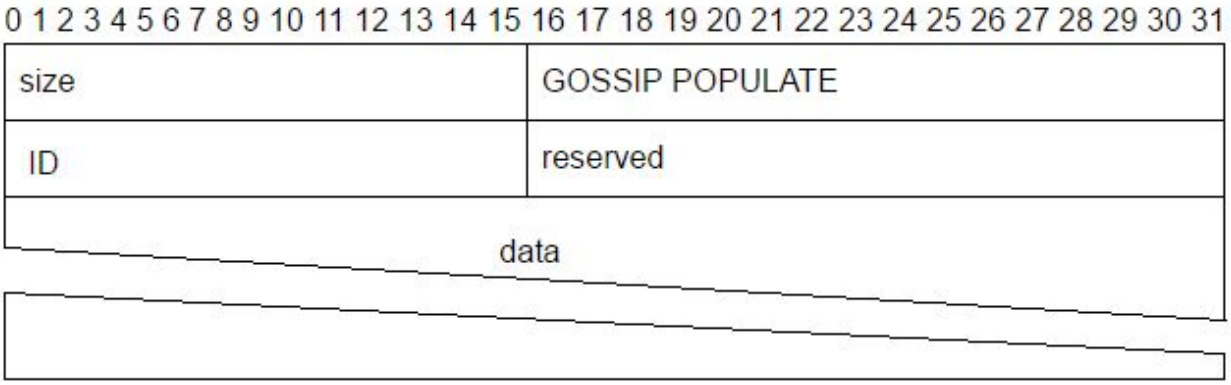
The INIT message is the first message a peer, running the gossip module, sends when it starts up. It is sent to a bootstrap server, using UDP, to get some knowledge of available peers. In the message, the peer specifies the degree, which is the maximum number of peers it is interested in connecting with.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

size	GOSSIP INIT
degree	reserved

POPULATE

The POPULATE message is the reply sent by the bootstrap server to a newly started peer, which has sent an INIT message. The ID is a unique identifier assigned to this peer by the bootstrap server. The data consists of the list of online peers and their IDs, so that the new peer may connect with them.



CONNECT

The CONNECT message is the first message a peer sends to another peer, to establish a connection with it. The ID specifies the identification of the peer who is sending this message, which was assigned to it by the bootstrap server on starting up. Upon receiving this message, the destination peer initiates a TCP connection with the source peer.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

size	GOSSIP CONNECT
ID	reserved

DISCONNECT

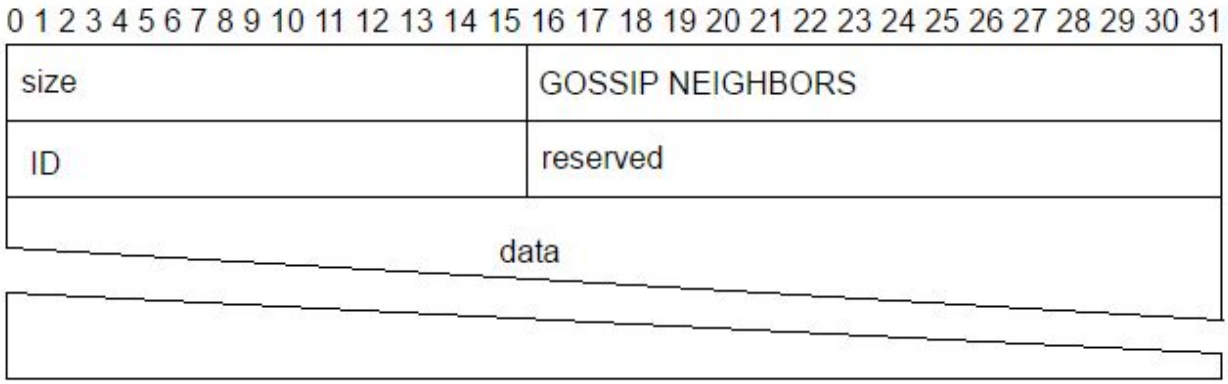
The DISCONNECT message is the last message a peer sends to another peer, to close the connection between them. The message format is similar to CONNECT.

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

size	GOSSIP DISCONNECT
ID	reserved

NEIGHBORS

Periodically, every peer gossips information to other peers it's connected to about its neighbors. This allows for extended peer discovery along the network.



Authentication

Basic authentication follows from the previously defined protocol. If the message types are valid, then the peer is likely to be talking to another gossip module.

Exception handling

After authentication, each peer forms a TCP connection with a few other peers. The peer can go down by closing the connection through a DISCONNECT message or timing out, in either case, an event is generated by the SocketChannel API. A peer can simply listen for the occurrence of this event and be notified to take appropriate measures in case of a peer disconnecting.

Since we are using TCP for communication within gossip peers, the data we receive on the application layer cannot be corrupted (since any corrupted data would be discarded at the

network layer). Gossip modules only verify the size of a message, along with the message type. To check the validity of the message contents, it is the job of the API modules. Gossip simply forwards the message to these modules, given they have subscribed for this message type.