TUM

Peer-to-Peer Systems and Security

IN2194, SoSe 2023

Gossip Module

for Anonymous and Unobservable VoIP Application

Midterm Report

Team #97

July 2024

1- Logical Structure:

- We are planning to have a class called “Gossip”, which will handle the overall setup of the module. In its initialization, it will initialize the Config class, it will connect to the bootstrapping node and it will open a separate socket for both api and p2p connections. It will also store all the necessary data needed by the module while it’s running, including data structures of current open connections, messages cached, etc.

- We plan to have two classes “ApiConnection” and “P2PConnection”. A new instance will be created for every connection and will be stored in the corresponding data structure of connections in the “Gossip” class. Each instance will store all the info we need about each connection. These classes will have methods that will handle receiving new messages depending on the message type. These classes will sometimes have to communicate with the “Gossip” class; for example when the API layer receives a GOSSIP ANNOUNCE message, it will have to call a specific method in the “Gossip” class which will in turn call a specific method in the connected “P2PConnections” instances it has.

- We will have other helper classes, one for parsing, validating and storing the data from the config file, we will name this class “Config”.
  We will also have one class per each message type that the module should handle. These classes will be used to properly parse, validate and store the messages’ data.

2- Process Architecture & Networking:

We will use the “asyncio” package, which utilizes a single-threaded design with an event loop. It excels at handling I/O-bound operations, like reading from or writing to a network or file, without blocking the entire program. We decided to go with this approach because it will be easier to handle and less error prone.

3- Security Measures:

Before interacting with peers that connect to us, we will first verify them by performing a Proof of Work puzzle within a specific time limit, to prevent Sybill attacks.
We will also drop any connection that sends us any unknown or malformed messages.

4- Specification of the P2P Protocol:

- **GOSSIP P2P INIT**: For peers to connect to each other, the initiator node will connect to the socket on the receiver and will immediately receive this message from the receiver, which will contain a unique 64 bit challenge.

Size (16 bits)	**GOSSIP P2P INIT** (16 bits)
Challenge (64 bits)	

- **GOSSIP P2P VERIFY**: After receiving the challenge from GOSSIP P2P INIT, the initiator will have to compute a 64 bit random nonce that when concatenated to the challenge and then hashed using SHA256, produces a value that has the first 24 bits as zeros. The initiator will then send this nonce alone in this message.

Size (16 bits)	**GOSSIP P2P VERIFY** (16 bits)
Nonce (64 bits)	

- **GOSSIP P2P OK**: This message will be sent by the receiver upon receiving the nonce in the GOSSIP P2P VERIFY if the nonce is valid. The connection is now considered valid and other messages can be communicated between the peers. The connection will be dropped if the nonce was not valid.

Size (16 bits)	**GOSSIP P2P OK** (16 bits)

- **GOSSIP P2P ANNOUNCE**: This message will be used to spread information between peers that is received via the GOSSIP ANNOUNCE from the API protocol. If a peer receives this message it will notify the API connections that are subscribed to the same datatype.

Size (16 bits)	**GOSSIP P2P ANNOUNCE** (16 bits)
Message ID (16 bits)	Data Type (16 bits)
TTL (8 bits)	Reserved (24 bits)
Data	

5- Workload Distribution:

We are both always working together on the same PC.