Interface:

Requests:

subscribe (topic**)**

unsubscribe (topic**)**

publish (topic, m**)**

Indications:

psDelivery (topic, m**)**

State:

diameter // diameter of the overlay
radiusSubs // includes own subscriptions Map[<process>] = {(<topic>, <TTL>)}
radiusSubs // includes own subscriptions Map[<topic>] = {(<process>, <TTL>)}

subHops // max hops of gossip when renewing subscriptions
pubHops // max hops of gossip when publishing a message of a given topic
neighbors // partial view of the overlay
delivered // set of Ids of messages
pendingSub // subscribe messages to be forwarded
pendingUnsub // unsubscribe messages to be forwarded
pendingPub // publish messages to be forwarded

Upon init () do:

diameter ← ln (#π * 10)
radiusSubs ← {}
subHops ← (diameter + 1) / 2
pubHops ← (diameter + 1) / 2
neighbors ← ⊥
delivered ← {}
pendingSub ← {}
pendingUnsub ← {}
pendingPub ← {}
Setup Periodic Timer RenewSub (T) // T < ttl

Upon subscribe (topic**) do:**

radiusSubs[myself] ← radiusSubs[myself] U {(topic, TTL)}
mid ← generateUID({myself, topic, getTimeOfSystem() })
delivered ← delivered U {mid}
pending ← pendingSub U {(**SUB**, myself, topic, TTL, subHops - 1, mid)}
Trigger GetNeighbors ()

Upon unsubscribe (topic**) do:**

radiusSubs[myself] ← radiusSubs[myself] \ {(topic, -)}
mid ← generateUID({**UNSUB**, myself, topic, getTimeOfSystem()})
delivered ← delivered U {mid}
pending ← pendingUnsub U {(**UNSUB**, myself, topic, subHops - 1, mid)}
Trigger GetNeighbors ()

Upon publish (topic, m**) do:**

mid ← generateUID({**PUB**, myself, topic, m})
delivered ← delivered U {mid}
pending ← pendingPub U {(**PUB**, myself, topic, pubHops, m, mid)}
Trigger GetNeighbors()

Upon RenewSub () do:

For each p ∈ radiusSubs **do:**
if {topic, ttl} ∈ radiusSubs[myself] ∧ ttl < TTL * 0.2 **then**
Trigger subscribe (topic**)**

Upon Neighbors (N**) do:**

neighbors ← N

For each (**SUB**, s, topic, TTL, subHops, mid) ∈ pendingSub **do**:

```
                    Trigger Gossip (SUB, myself, topic, TTL, subHops, mid)

                For each (UNSUB, s, topic, subHops, mid) ∈ pendingUnsub  do:
                    Trigger Gossip(UNSUB, s, topic, subHops, mid)

                For each (PUB, s, topic, m) ∈ pendingPub  do:
                    Trigger Gossip(PUB, s, topic, pubHops, m, mid)

                pendingSub ← {}
                pendingUnsub ← {}
                pendingPub ← {}


        Upon Receive (SUB, s, topic, TTL, subHops, mid) do:
            if mid ∉ delivered then
                radiusSubs[s] ← radiusSubs[s] U {(topic, TTL)}
                delivered ← delivered U {mid}

                if subHops > 0 then
                    pending ← pending U {(SUB, s, topic, TTL, subHops - 1, mid)}
                    Trigger GetNeighbors()


        Upon Receive (UNSUB, s, topic, subHops, mid) do:
            if mid ∉ delivered then
                radiusSubs[s] ← radiusSubs[s] \ {(topic, -)}
                delivered ← delivered U {mid}

                if subHops > 0 then
                    pending ← pending U {(UNSUB, s, topic, TTL, subHops - 1, mid)}
                    Trigger GetNeighbors()

        Upon Receive (PUB, s, topic, pubHops, m, mid) do:
            if mid ∉ delivered then
                delivered ← delivered U {mid}

                if {myself, TTL} ∈ radiusSubs[topic] ∧ TTL > 0 then
                    Trigger psDelivery(topic, m)

                For Each {p, TTL} ∈ radiusSubs[topic]  do:
                    if TTL > 0 then
                        Trigger Send (topic, m, mid)

                if pubHops > 0 then
                    pending ← pending U {(PUB, s, topic, pubHops - 1, m)}
                    Trigger GetNeighbors()


        Upon Receive (DIRECTMSG, topic, m, mid) do:
            if mid ∉ delivered then
                delivered ← delivered U {mid}
                if {topic, TTL} ∈ radiusSubs[myself] ∧ TTL > 0 then
                    Trigger psDelivery(topic, m)


//timeout para apagar radiusSubs, delivered
```