**Interface:**

**Requests:**

**subscribe (**topic**)**

**unsubscribe (**topic**)**

**publish** **(**topic, m**)**

**Indications:**

**psDelivery** **(**topic, m**)**

**State:**

diameter // diameter of the overlay

radiusSubsByTopic // includes own subscriptions Map[<process>] = {(<topic>, <TTL>)}

radiusSubsByProcess // 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)

radiusSubsByTopic ← {}

radiusSubsByProcess ← {}

subHops ← (diameter + 1) / 2

pubHops ← (diameter + 1) / 2

neighbors ← ⊥

delivered ← {}

pendingSub ← {}

pendingUnsub ← {}

pendingPub ← {}

**Setup Periodic Timer RenewSub (T)** // T < ttl

**Setup Periodic Timer CleanExperiedSubs(CLEAN\_FREQUENCY)**

**Upon subscribe (**topic**) do:**

radiusSubsByProcess [myself] ← radiusSubsByProcess [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:**

radiusSubsByProcess [myself] ← radiusSubsByProcess [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} ∈ radiusSubsByProcess [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**

radiusSubsByProcess [s] ← radiusSubsByProcess [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**

radiusSubsByProcess [s] ← radiusSubsByProcess [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} ∈ radiusSubsByTopic [topic] ∧ ttl > 0 **then**

**Trigger psDelivery(**topic, m**)**

**For Each** {p, ttl} ∈ radiusSubsByTopic [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} ∈ radiusSubsByProcess [myself] ∧ ttl > 0 **then**

**Trigger psDelivery(**topic, m**)**

**Upon CleanOldSubs() do:**

**For each** p,{(topic,ttl)} ∈ radiusSubsByProcess ∧ ttl <= 0 **do:**

removeFromRadiusSubs(p, topic)

//timeout para apagar radiusSubs, delivered