Algoritmos e Sistemas Distribuídos
Algorithms for the Protocols in the Babel Example

João Carlos Antunes Leitão
NOVA Laboratory for Computer Science and Informatics (NOVA LINCS)
and
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade NOVA de Lisboa

October 9, 2022

Algorithm 1: Flood Broadcast

Interface:
Requests:
BroadcastRequest (s**,** mid**,** m **)** `//s is the sender, m the message`
Indications:
BroadcastDeliver (s**,** mid m **)** `//s is the sender, m the message`

State:
neighbors `set with all neighbors`
received `set with message identifiers already received`

Upon Init () do:
neighbors $\longleftarrow \{\}$
received $\longleftarrow \{\}$

Upon BroadcastRequest (s**,** mid**,** m **) do:**
Call processFloodMessage(FLOODMSG, $\{s, mid, m\}$)

Upon Receive (**FLOODMSG,** $\{s, mid, m\}$ **) do:**
Call processFloodMessage($\{s, mid, m\}$)

Procedure processFloodMessage ($\{s, mid, m\}$)
If $mid \exists$ received **Then**
received $\longleftarrow$ received $\cup$ mid
Trigger BroadcastDeliver (s, mid m)
Foreach $p \in$ neighbors **do**
Trigger Send (p, FLOODMSG, $\{s, mid, m\}$)

Upon NeighborUP (p) **do**
neighbors $\longleftarrow$ neighbors $\cup \{p\}$

Upon NeighborDown (p) **do**
neighbors $\longleftarrow$ neighbors $\setminus \{p\}$

Algorithm 2: Simple Full Membership

Interface:
Requests:

Indications:
neighborUP (p **)**
neighborDOWN (p **)**

State:
self `//identifier of self`
membership `//Set with all neighbors`
subsetSize `//number of neighbors to send to other`
$\mathcal{T}$ `//period between anouncements`

Upon Init (myself, ssSize, t, contact) do:
self $\longleftarrow$ myself
membershio $\longleftarrow$ {}
If contact $\neq \perp$ **Then**
membershio $\longleftarrow$ membership $\cup$ { contact }
subsetSize $\longleftarrow$ ssSize
$\mathcal{T} \longleftarrow$ t
Setup Periodic Timer SAMPLETIMER ($\mathcal{T}$)

Upon Timer SAMPLETIMER **do:**
If #membership $>= 1$ **Then**
target $\longleftarrow$ random(membership)
sample $\longleftarrow$ { self }
sample $\longleftarrow$ random(subsetSize, membershop $\setminus$ { target }
Trigger Send (target, SAMPLEMSG, sample

Upon Receive (s, SAMPLEMSG, sample) **do:**
Foreach $p \in$ sample **do**
If $p \notin$ membership **do**
membership $\longleftarrow$ membership $\cup \{p\}$
Trigger NeighborUP (p)

Upon ChannelClosed (p **) do:**
If $p \in$ membership **do**
membership $\longleftarrow$ membership $\setminus \{p\}$
Trigger NeighborDOWN (p)