Algorithm 1 Code for process p_i

```
$id_i = i$
$neighbors_i = \{$ Conjunto de vecinos de $p_i\}$
$identifiers_i = \{\emptyset\}$

$main()$
begin:
    for each $j \in neighbors_i$ do
        send MYNAME($neighbors_i$) to $j$
    end for
end

when  MYNAME($identifiers_j$) is received from neighbor $p_j$
begin:
    $identifiers_i = identifiers_i \cup identifiers_j$
end
```

Algorithm 2 Code for process p_i

```
$vecinos_i$, $proc\_conocidos_i = \{i\}$
$canales\_conocidos_i = \{< i, j > | j \in vecinos_i\}$
start()
begin:
    for each $j \in vecinos_i$ do
        send POSITION$(i, vecinos_i)$ to $j$
    end for
end

when  POSITION$(k, vecinos)$ is received from neighbor $p_j$
begin:
    if $k \notin proc\_conocidos_i$ then
        $proc\_conocidos_i = proc\_conocidos_i \cup \{k\}$
        $canales\_conocidos_i = canales\_conocidos_i \cup \{< k, \ell > | \ell \in vecinos\}$
        for $\ell \in vecinos_i \setminus \{j\}$ do
            send POSITION$(k, vecinos)$ to $\ell$
        end for
        if $\forall < \ell, m > \in canales\_conocidos_i : \{\ell, m\} \subset proc\_conocidos_i$ then
            $p_i$ conoce la gráfica de comunicación return
        end if
    end if
end
```

Algorithm 3 Broadcast ingenuo para el proceso p_i

```
$neighbors_i = \{$conjunto de vecinos$\}$
$seen\_message = false$
**if** $p_i = p_s$ **then**
    $seen\_message = true$
    send $M$ to all neighbors
**end if**

**when** $M$ is received from neighbor $p_j$
**begin:**
    **if** $seen\_message = false$ **then**
        $seen\_message = true$
        send M to all neighbors
    **end if**
**end**
```

Algorithm 4 Algoritmo para construir el árbol generador

```
Initially do
begin:
    if $p_s = p_i$ then                                    ▷ Si soy el nodo distinguido
        $parent_i = i$; $expected\_msg_i = |neighbors_i|$
        for each $j \in neighbors_i$ do send $GO()$ to $p_j$
        end for
    else $parent_i = \emptyset$                             ▷ Si no, solo inicializo mis variables
    end if
    $children_i = \emptyset$
end

when GO() is received from $p_j$ do
begin:
    if $parent_i = \emptyset$ then
        $parent_i = j; expected\_msg_i = |neighbors_i| - 1$
        if $expected\_msg_i = 0$ then send BACK($i$) to $p_j$
        else
            for each $k \in neighbors_i \setminus \{j\}$ do send GO() to $p_k$
            end for
        end if
    else send BACK($\emptyset$) to $p_j$
    end if
end

when BACK($val\_set$) is received from $p_j$ do
begin:
    $expected\_msg_i = expected\_msg_i - 1$
    if $val\_set \neq \emptyset$ then $children_i = children_i \cup \{j\}$
    end if
    if $expected\_msg_i = 0$ then
        if $parent_i \neq i$ then
            send BACK($i$) to $parent_i$
        end if
    end if
end
```

Algorithm 5 Broadcast

```
Initially do
begin:
    if $p_s = p_i$ then                                   ▷ Si soy el nodo distinguido
        $data$ = mensaje que se quiere difundir
        for each $j \in children_i$ do send $GO(data)$ to $p_j$
        end for
    else $data = \emptyset$                                ▷ Si no, solo inicializo mis variables
    end if
end

when GO($data$) is received from $p_j$ do
begin:
    for each $k \in children_i$ do send $GO(data)$ to $p_k$
    end for
end
```

Algorithm 6 Convergecast

```
Initially do
begin:
    $v_i$                                                  ▷ Los valores que se enviarán
    if $children_i = \emptyset$ then                       ▷ Las hojas empiezan la ejecución
        send BACK($(i, v_i)$) to $parent_i$
    end if
end

when BACK($data$) is received from each $p_j$ such that $j \in children_i$ do
begin:
    $val\_set_i = \bigcup_{j \in children_i} val\_set_j \cup \{(i, v_i)\}$
    if $parent_i \neq i$ then
        send BACK($val\_set_i$) to $p_k$
    else
        the root $p_s$ can compute $f(val\_set_i)$
    end if
end
```

Algorithm 7 Broadcast y Convergecast sobre un árbol generador

```
Initially do
begin:
    if $p_s = p_i$ then                                   ▷ Si soy el nodo distinguido
        $parent_i = i$; $expected\_msg_i = |neighbors_i|$
        for each $j \in neighbors_i$ do send GO($data$) to $p_j$
        end for
    else $parent_i = \emptyset$                           ▷ Si no, solo inicializo mis variables
    end if
    $children_i = \emptyset$
end

when GO() is received from $p_j$ do
begin:
    if $parent_i = \emptyset$ then
        $parent_i = j; expected\_msg_i = |neighbors_i| - 1$
        if $expected\_msg_i = 0$ then send BACK($(i, v_i)$) to $p_j$
        else
            for each $k \in neighbors_i \setminus \{j\}$ do send GO($data$) to $p_k$
            end for
        end if
    else send BACK($\emptyset$) to $p_j$
    end if
end

when BACK($val\_set$) is received from $p_j$ do
begin:
    $expected\_msg_i = expected\_msg_i - 1$
    if $val\_set \neq \emptyset$ then $children_i = children_i \cup \{j\}$
    end if
    if $expected\_msg_i = 0$ then
        $val\_set_i = \bigcup_{x \in children_i} val\_set_x \cup \{(i, v_i)\}$
        if $parent_i \neq i$ then
            send BACK($(v_i, i)$) to $parent_i$
        else
            $p_s$ puede calcular la función $f(val\_set)$
        end if
    end if
end
```