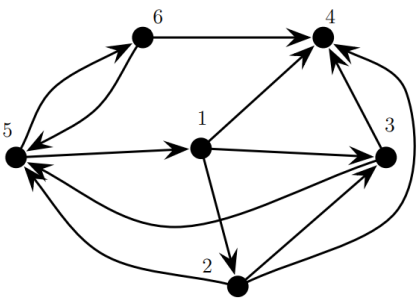
Student Name:
Student ID:

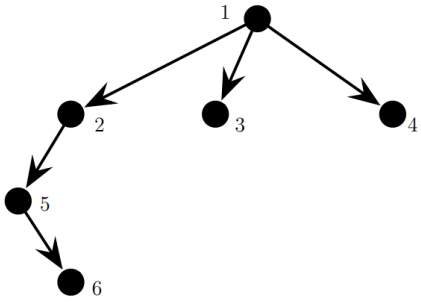
LabReSiD25
Hands-On 15

Università degli Studi di Messina

Flooding con pipe

Studiare il funzionamento delle pipe in linux. E' possibile realizzare per il grafo in figura l'algoritmo Flooding in C utilizzando un processo (o thread) per ogni agente ed una `Pipe` (bloccante o non bloccante?) per ogni canale di comunicazione? Si considerino due pipe per ogni agente (in-pipe, out-pipe) e si implementino con precisione le funzioni `msg` ed `stf`.





Synchronous Network: $\mathcal{S} = (\{1, \ldots, n\}, E_{\text{cmm}})$

Distributed Algorithm: FLOODING

Alphabet: $\mathbb{A} = \{\alpha, \ldots, \omega\} \cup \texttt{null}$

Processor State: $w = (\texttt{parent}, \texttt{data}, \texttt{snd-flag})$, where

`parent`	$\in \{0, \ldots, n\}$,	initially:	$\texttt{parent}^{[1]} = 1$,
			$\texttt{parent}^{[j]} = 0$ for all $j \neq 1$
`data`	$\in \mathbb{A}$,	initially:	$\texttt{data}^{[1]} = \mu$,
			$\texttt{data}^{[j]} = \texttt{null}$ for all $j \neq 1$
`snd-flag`	$\in \{\texttt{false}, \texttt{true}\}$,	initially:	$\texttt{snd-flag}^{[1]} = \texttt{true}$,
			$\texttt{snd-flag}^{[j]} = \texttt{false}$ for $j \neq 1$

```
function msg(w, i)
1: if (parent ≠ i) AND (snd-flag = true) then
2:     return data
3: else
4:     return null
```

```
function stf(w, y)
```

```
1:  case
2:    (data = null) AND (y contains only null messages):
       % The node has not yet received the token
3:       new-parent := null
4:       new-data := null
5:       new-snd-flag := false
6:    (data = null) AND (y contains a non-null message):
       % The node has just received the token
7:       new-parent := smallest UID among transmitting in-neighbors
8:       new-data := a non-null message
9:       new-snd-flag := true
10:   (data ≠ null):
       % If the node already has the token, then do not re-broadcast it
11:      new-parent := parent
12:      new-data := data
13:      new-snd-flag := false
14: return (new-parent, new-data, new-snd-flag)
```