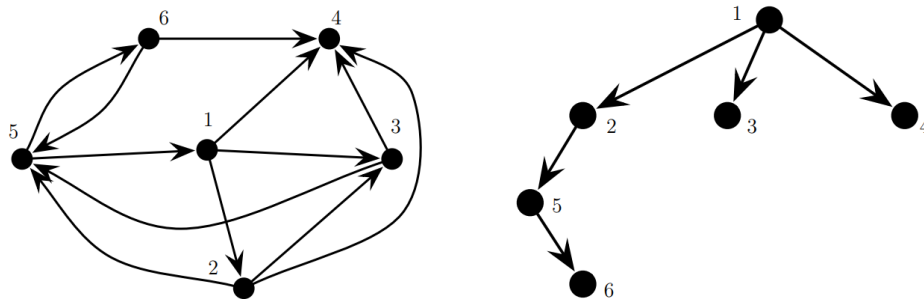


## Flooding con fifo

Studiare il funzionamento delle fifo 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 **Fifo** (bloccante o non bloccante?) per ogni canale di comunicazione? Si implementino con precisione le funzioni `msg` ed `stf`.



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

Distributed Algorithm: FLOODING

Alphabet:  $\mathbb{A} = \{\alpha, \dots, \omega\} \cup \text{null}$

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

<code>parent</code>	$\in \{0, \dots, n\}$ ,	initially: <code>parent</code> <sup>[1]</sup> = 1,
		<code>parent</code> <sup>[j]</sup> = 0 for all $j \neq 1$
<code>data</code>	$\in \mathbb{A}$ ,	initially: <code>data</code> <sup>[1]</sup> = $\mu$ ,
		<code>data</code> <sup>[j]</sup> = null for all $j \neq 1$
<code>snd-flag</code>	$\in \{\text{false}, \text{true}\}$ ,	initially: <code>snd-flag</code> <sup>[1]</sup> = true,
		<code>snd-flag</code> <sup>[j]</sup> = false for $j \neq 1$

function `msg(w, i)`

```

1: if (parent  $\neq i$ ) 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)

```

---