recursive algorithm

algorithm new_name(x, first, dir):

% x (n \geq x \geq 1) is the recursion parameter %
(01) \quad SM[x, first, dir][i] \leftarrow old_name_i;
(02) \quad competing_i \leftarrow SM[x, first, dir].collect();
(03) \quad if |competing_i| = x
(04) \quad \quad \text{then last} \leftarrow \text{first} + \text{dir}(2x - 2);
(05) \quad \quad \text{if old_name}_i = \max(\text{competing}_i)
(06) \quad \quad \quad \quad \text{then res}_i \leftarrow \text{last}
(07) \quad \quad \text{else res}_i \leftarrow \text{new_name}(x - 1, \text{last + dir, dir})
(08) \quad \quad \text{end if}
(09) \quad \text{else res}_i \leftarrow \text{new_name}(x - 1, \text{first, dir})
(10) \text{end if;}
(11) \text{return(res}_i\).

new_name(n,1,1)
• For a pair (first, dir); the Algo ensures at most x processes invoke new_name(x, first, dir)

• These processes compete for new names in space name of size 2x-1

• Initial space New_name(n,1,1) -> 1, 2n-1
• if k processes participate in the renaming, their main call new_name(n, 1, 1) will systematically entail the call new_name(n – 1, 1, 1), etc., until the call new_name(k, 1, 1).

• If k processes participate to new_name(k, 1, 1) at least one (the last process that writes) find k processes in competing

• In space 1..X, (dir=1) if k processes compete they get a name in 1..2k-1

• In space 1..X, (dir=-1) if k processes compete they get a name in X-(2k-2)..X
• **Splitter behavior** associated with the array of atomic registers SM \([x, \text{first}, \text{dir}]\) is defined by the following properties: (let \(x' = x - 1\)):

• At most \(x' = x - 1\) processes invoke new_name\((x-1, \text{first}, \text{dir})\) (line 9). Hence, these processes will obtain new names in an interval of size \((2x' - 1)\) as follows:
  • If \(\text{dir} = 1\), the new names will be in the “going up” interval \([\text{first}..\text{first} + (2x' - 2)]\),
  • if \(\text{dir} = -1\), the new names will be in the “going down” interval \([\text{first} - (2x' - 2)..\text{first}]\).

• At most \(x' = x - 1\) processes invoke new_name\((x - 1, \text{last} + \overline{\text{dir}}, \overline{\text{dir}})\) (line 7), where \(\text{last} = \text{first} + \text{dir}(2x - 2)\) (line 7). Hence, these \(x' = x - 1\) processes will obtain their new names in a renaming space of size \((2x' - 1)\) starting at last + 1 and going from left to right if \(\overline{\text{dir}} = 1\), or starting at last - 1 and going from right to left if \(\overline{\text{dir}} = -1\).
At most one process “stops”, i.e., defines its new name as $\text{last} = \text{first} + \text{dir}(2x - 2)$ (lines 4, 5, 6). Let us observe that the only process that can stop is the one such that $\text{id}$ has the greatest value in the array $\text{SM}[x, \text{first}, \text{dir}][1..n]$ which contains then exactly $x$ identities.
• With objects cons for 2 processes we can achieve strong adaptive renaming

  • (strong = space of new name are in 1..n)

• Adaptive: if k processes compete they gets a name in 1..k

• With strong adaptive renaming we can achieve consensus for 2 processes
Byzantine failures
• A correct process is a process that follow its code

• A byzantine process might fail by exhibiting arbitrary behavior (e.g. send bogus message)

• (but it can’t pretend to be another process)(or no message)
• With at most $t$ crash failures, in synchronous system we can solve consensus.
• This is not true with Byzantine failure.
• First define «byzantine agreement»: the previous definition of validity «the decided value is a value proposed by some process» doesn’t work.
Byzantine agreement

• **Agreement**: No two correct processes decide on different values

• **Validity**: if all correct processes propose the same value $v$ then $v$ is the only possible decision for correct processes

• **Termination**: All correct processes eventually decide
Attack

NO
Attack

Re: NO
Impossibility result

• Theorem: If $n \leq 3t$ then it is impossible to solve byzantine agreement

• proof $n=3$ $t=1$
Decide
• The proof may be easily extends to any value of $n$
\[ n \geq 3t \]
\[ n \geq 3t \]