Algorithms for Irrevocable Consensus

1 The *OneThirdRule* algorithm

Algorithm 1 The *OneThirdRule* algorithm

Initialization:
  $x_i := \nu_i$ { *ν_i is the initial value of i* }
  $y_i := \perp$

$\mathbf{S_i}$:
  send $\langle\, x_i \,\rangle$ to all

$\mathbf{T_i}$:
  if more than $2n/3$ values have been received in the current round **then**
    $x_i :=$ the smallest value amongst the most frequent received values
  if more than $2n/3$ values received in the current round are equal to $\overline{\nu}$ **then**
    $y_i := \overline{\nu}$

Theorem 1.1. *The* OneThirdRule *algorithm achieves irrevocable consensus in any network* $\mathbb{G}$ *such that*

1. $\exists t_0 \in \mathbb{N}^*,\ \exists S \subseteq V,\ |S| > 2n/3$ *and* $\mathbb{G}(t_0)$ *is* S*-uniform;*
2. $\forall i \in V,\ \forall t \in \mathbb{N}^*,\ \exists t_i \geqslant t,\ |\mathrm{In}_i(t_i)| > 2n/3.$

2 The *LastVoting* algorithm

We assume that the network is equipped with an oracle Leader, which may be questioned at each round $t \equiv 1 \mod 4$. The response provided to the agent i by the oracle is denoted by Leader$[i]$.

Theorem 2.1. *The* LastVoting *algorithm achieves irrevocable consensus in any network* $\mathbb{G}$ *for which there exists an index* $\Phi_0 \in \mathbb{N}^*$ *such that*

1. $\forall i \in V,\quad |\mathrm{In}_i(4\Phi_0 - 3)| > n/2$ *and* $|\mathrm{In}_i(4\Phi_0 - 1)| > n/2$*;*
2. $\forall i, j \in V^2,\ \mathrm{Leader}[i](\Phi_0) = \mathrm{Leader}[j](\Phi_0)$*;*
3. $\forall i \in V,\ \mathrm{Leader}[i](\Phi_0) \in \mathrm{In}_i(4\Phi_0 - 3) \cap \mathrm{In}_i(4\Phi_0 - 2) \cap \mathrm{In}_i(4\Phi_0 - 1) \cap \mathrm{In}_i(4\Phi_0)$.

Algorithm 2 The LastVoting algorithm

```
Initialization :
    x_i ∈ 𝒱, initially ν_i                                {ν_i is the initial value of i}
    vote_i ∈ 𝒱 ∪ {?}, initially ?
    commit_i a Boolean, initially false
    ready_i a Boolean, initially false
    ts_i ∈ ℕ, initially 0
    numround_i ∈ ℕ, initially 1

S_i :
    if numround_i ≡ 1 mod 4 then
        send ⟨x_i , ts_i⟩ to Leader[i]
    if numround_i ≡ 2 mod 4 then
        if i = Leader[i] and commit_i then
            send ⟨vote_i⟩ to all
    if numround_i ≡ 3 mod 4 then
        if ts_i = (numround_i + 1)/4 then
            send ⟨ack⟩ to Leader[i]
    if numround_i ≡ 0 mod 4 then
        if i = Leader[i] and ready_i then
            send ⟨vote_i⟩ to all

T_i :
    if numround_i ≡ 1 mod 4 then
        if i = Leader[i] and
        number of ⟨ν , θ⟩ received > n/2 then
            let θ̄ be the largest θ from ⟨ν , θ⟩ received
            vote_i := one ν such that ⟨ν , θ̄⟩ is received
            commit_i := true
    if numround_i ≡ 2 mod 4 then
        if received ⟨v⟩ from Leader[i] then
            x_i := v ; ts_i := (numround_i + 2)/4
    if numround_i ≡ 3 mod 4 then
        if i = Leader[i] and number of ⟨ack⟩ received > n/2 then
            ready_i := true
    if numround_i ≡ 0 mod 4 then
        if received ⟨v⟩ from Leader[i] then
            y_i := v
        if i = Leader[i] then
            ready_i := false
            commit_i := false
    numround_i := numround_i + 1
```