Advanced Distributed Algorithms and Data Structures

Chapter 7: Clock Synchronization

Christian Scheideler
Institut für Informatik
Universität Paderborn

Overview

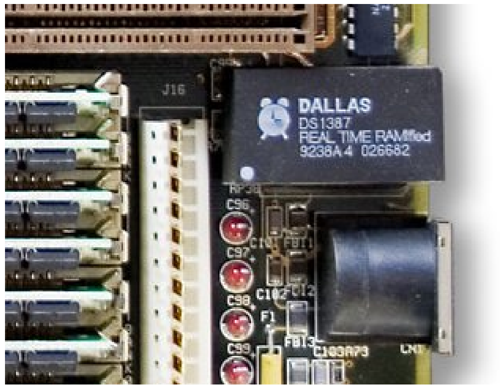
- Problem
- Physical Clock Synchronization
- Median rule

Clock Synchronization

Clock devices in computers

- RTC (Real Time Clock)
  - Battery backed up
  - 32.768 kHz oscillator + counter
  - Get value via interrupt

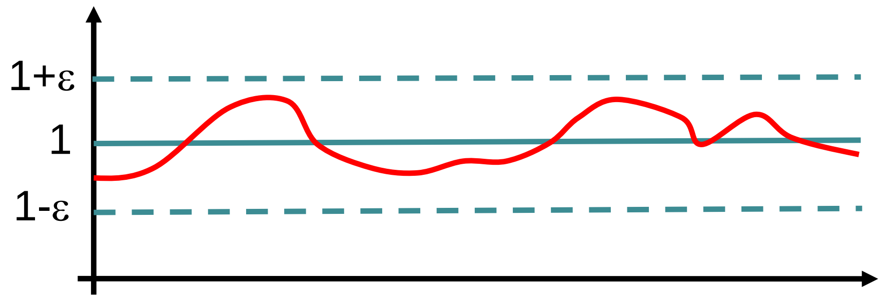
- HPET (High Precision Event Timer)
  - Oscillator: 10 Mhz … 100 Mhz
  - Up to 10 ns resolution
  - Schedules threads
  - Smooth media playback
  - Usually inside processors



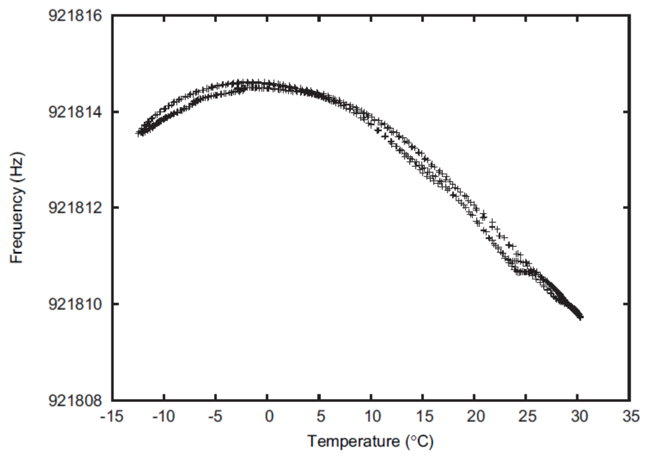
By Zac Luzader, Codeczero

Clock Drift

- Clock drift: random deviation from the nominal rate dependent on power supply, temperature, etc.



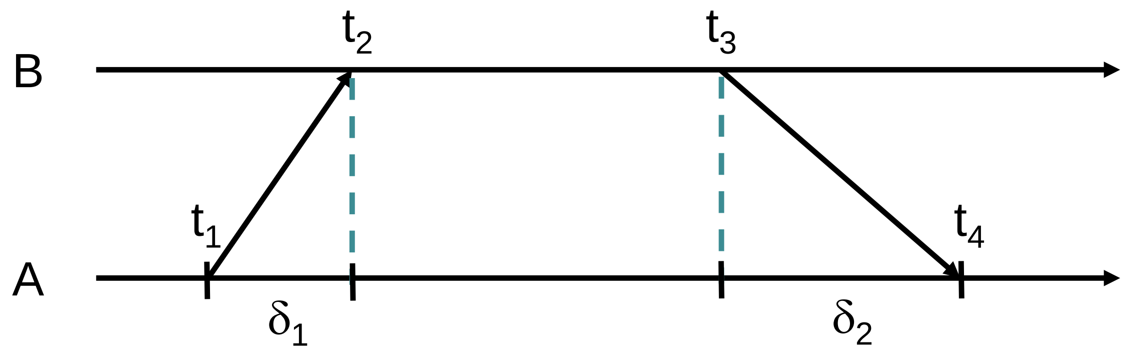
- E.g., TinyNodes have a maximum drift of 30-50 ppm (parts per million), which corresponds to up to 0.18s per hour



Clock Synchronization in Networks

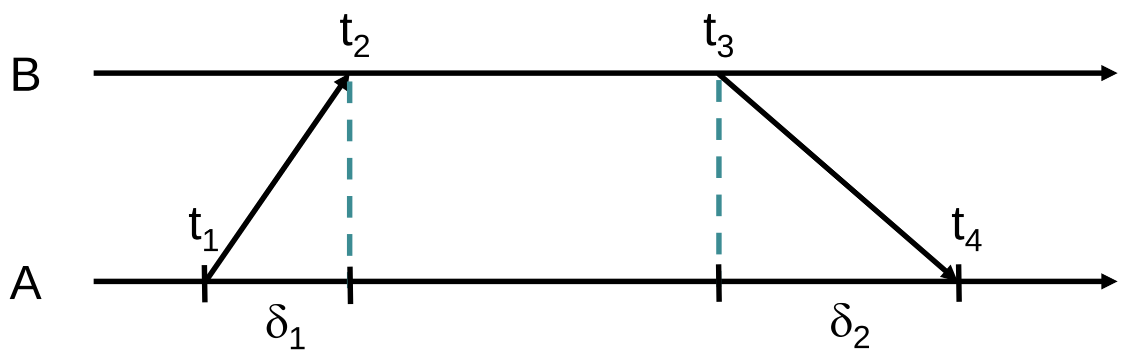
- Network Time Protocol (NTP)
- Precision Time Protocol (PTP)
- Publicly available NTP servers (UTC)

Main problem: estimate propagation delays δ_1 and δ_2



- t_1, t_4: time according to A
- t_2, t_3: time according to B

NTP Protocol



In NTP, average propagation delay δ and clock skew Θ are calculated as follows:

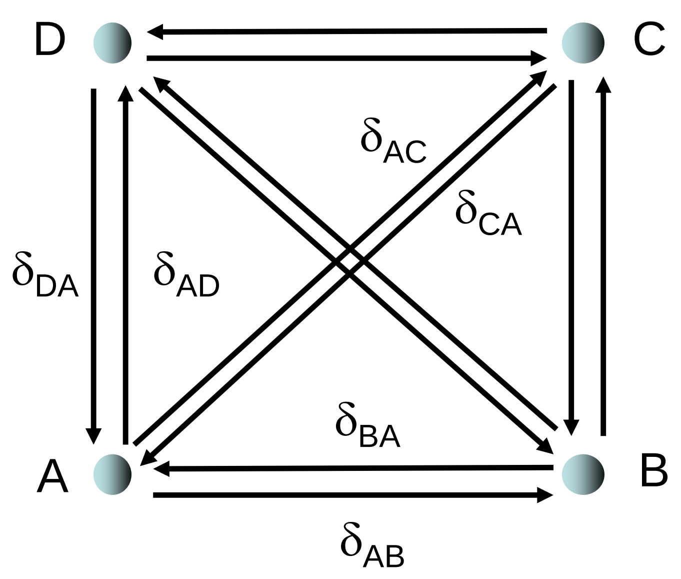
$$\delta = \frac{(t_4-t_1)-(t_3-t_2)}{2} = \frac{(t_2-t_1)-(t_3-t_4)}{2} = \frac{\delta_1+\delta_2}{2}$$

$$\Theta = \frac{t_2-(t_1+\delta)}{2} + \frac{(t_3+\delta)-t_4}{2} = \frac{(t_2-t_1)+(t_3-t_4)}{2} = \frac{\delta_1-\delta_2}{2}$$

Propagation Delay

Problem: δ_1 and δ_2 may differ. How to determine them?

Solution: we need 4 nodes



- #cycles of length 2: 6
  (only one possible direction)
- #cycles of length 3: 4
  (two possible directions)

Hence, $6+2\cdot4=14$ measurements possible of which any 12 are linearly independent, giving 12 linearly independent equations with 12 unkown variables δ_{XY}

Hence, we can compute the different propagation delays using, for example, Gaussian elimination.

Propagation Delay

Problem: messages may experience jitter (variance) in the propagation delay

- Deterministic as well as non-deterministic sources of jitter
- Mostly responsible: the computers

Solution 1: timestamp packets at the MAC layer (some hardware like 1G Intel cards (82580) can timestamp any packet at the MAC layer)

$\rightarrow$ Jitter is usually reduced to about a microsecond

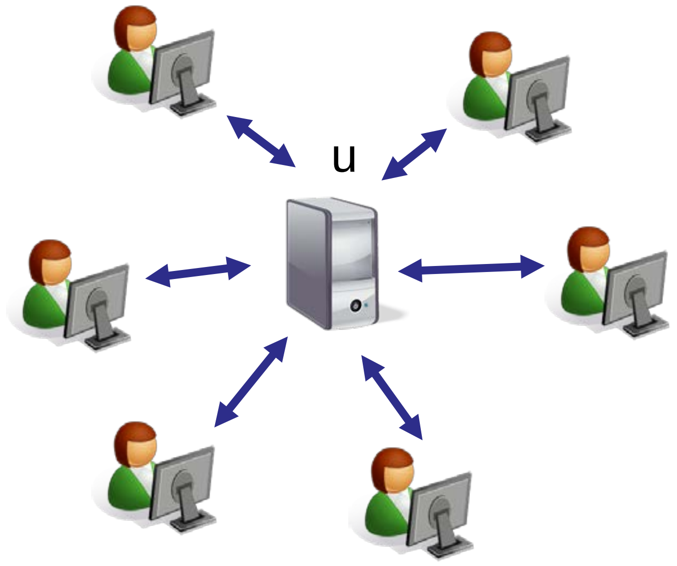
Solution 2: use the estimation strategy in TCP

- Initially, δ is set to the first computed value and $V[\delta]$ to $\delta/2$
- For each additional computed δ' of δ,
  $V[\delta]:=(1-\beta)V[\delta] + \beta\cdot|\delta-\delta'|$ (usually, $\beta=1/4$)
  $\delta:=(1-\alpha)\cdot\delta+\alpha\cdot\delta'$ (usually, $\alpha=1/8$)
- A reasonable upper bound on the next value of δ is then usually computed by $\delta_{max}:=\delta+4\cdot V[\delta]$ (which can be used, for example, to set an upper bound for waiting for an ACK message)

Clock Synchronization in Networks

Simple strategy:

the nodes select a master node u, and everybody then synchronizes with u (using the NTP strategy together with the MIMD protocol for handling contention as described in Chapter 6)



Problem: master node might be faulty or even adversarial.

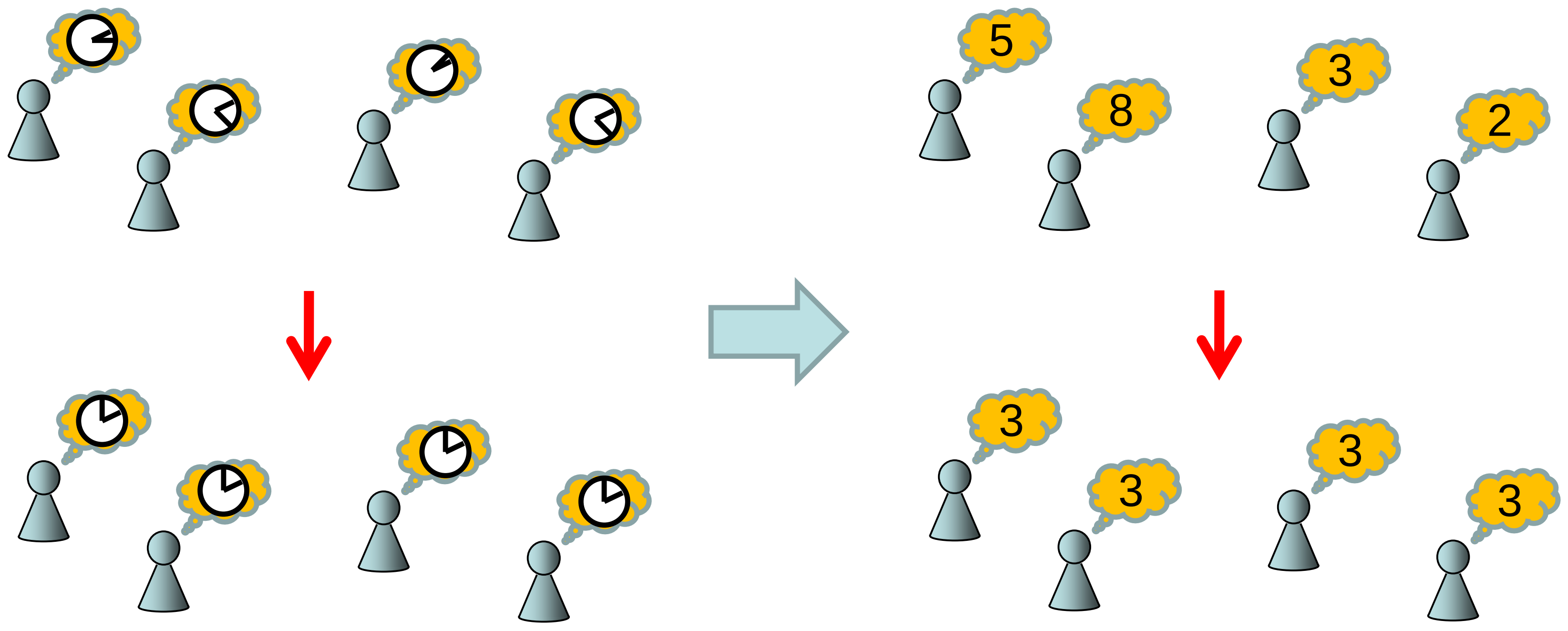
→ Clock synchronization will fail!

Is there a more robust strategy?

Clock Synchronization in Networks

Assumption: message delays are 0.

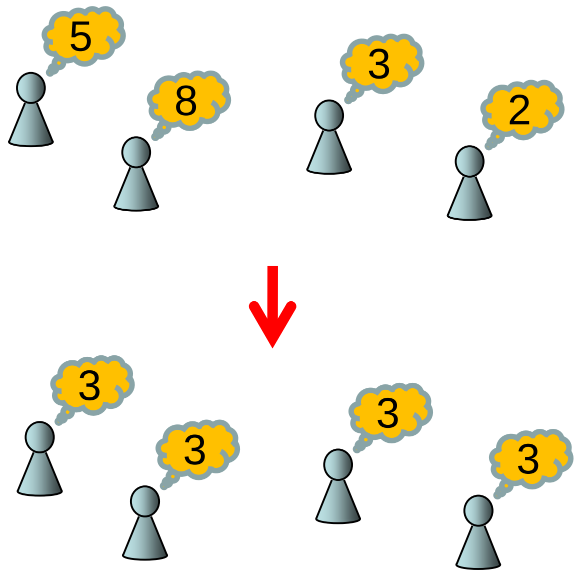
→ Clock synchronization reduces to distributed consensus



Distributed Consensus

Classical form of distributed consensus problem:

Processes must make a decision.



Formal requirements:

- Agreement: All correct processes must agree on the same value.
- Validity: For each correct process, its output must be a valid input of some process.
- Termination: All processes must eventually decide on an output value.

Distributed Consensus

- Fischer, Lynch, Paterson ´83:
  Impossible for deterministic protocol to decide in asynchronous message passing environment if single Byzantine crash.
- Fischer, Lynch, Merritt ´86:
  Impossible for deterministic protocol to decide in synchronous environment if Byzantine failures by >1/3 of processes.
- Ben-Or ´83,….:
  Randomized algorithms can solve consensus with prob $\rightarrow 1$ in asynchronous environment for constant fraction of Byzantine failures.
  But: overhead is large

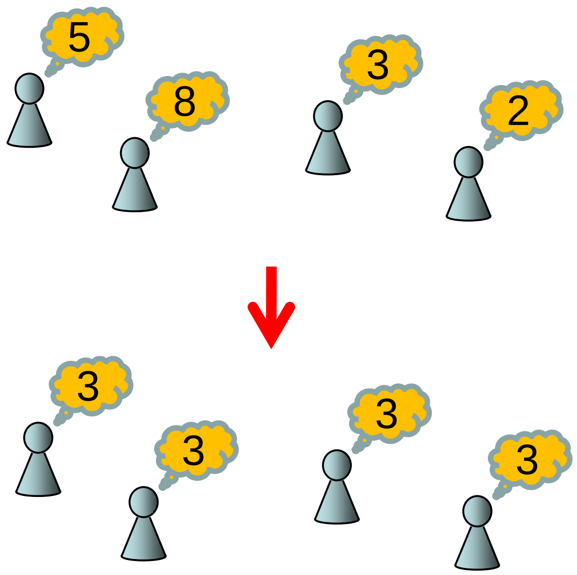
In our case, decisions are not needed. It is sufficient to know that eventually a protocol reaches a consensus.

Distributed Consensus

Stabilizing consensus problem:
Processes do not have to make a decision any more.

Formal requirements:

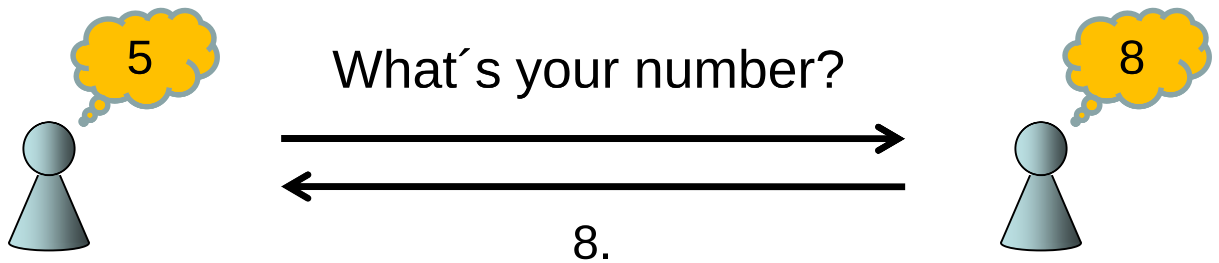
- Stabilization: All correct processes eventually reach a stable state (resp. a state remaining stable for poly(n) many communication rounds).
- Validity: For each correct process, its value must have been proposed by a process in some previous round.
- Agreement: for every stable state, all correct processes have the same value.



Stabilizing Consensus

Model:

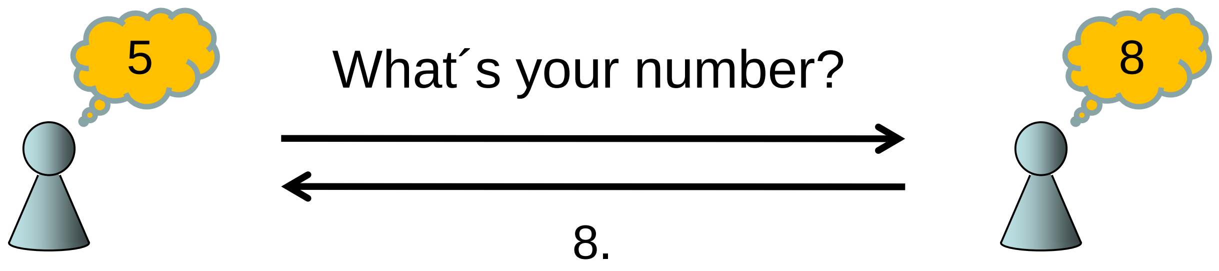
- n players
- Every player knows all others
- Time proceeds in synchronous rounds
- In each round, each player can contact any set of other players



Stabilizing Consensus

Complexity measures:

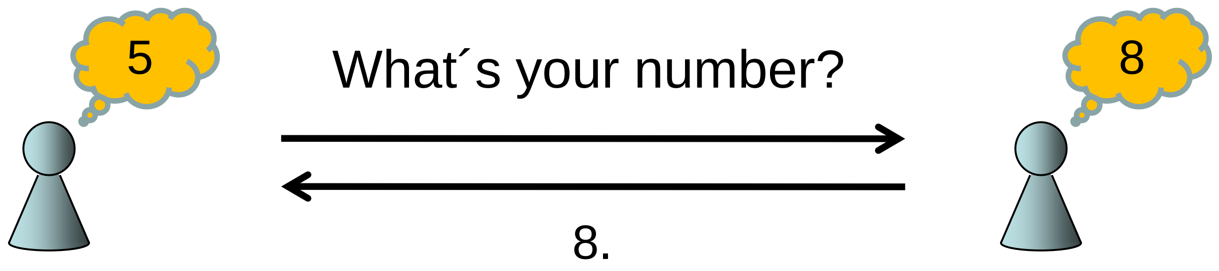
- Time: minimize number of communication rounds till stable consensus is reached
- Work: minimize maximum total work (i.e. number of messages) needed by a player for this



Distributed Consensus

Basic questions:

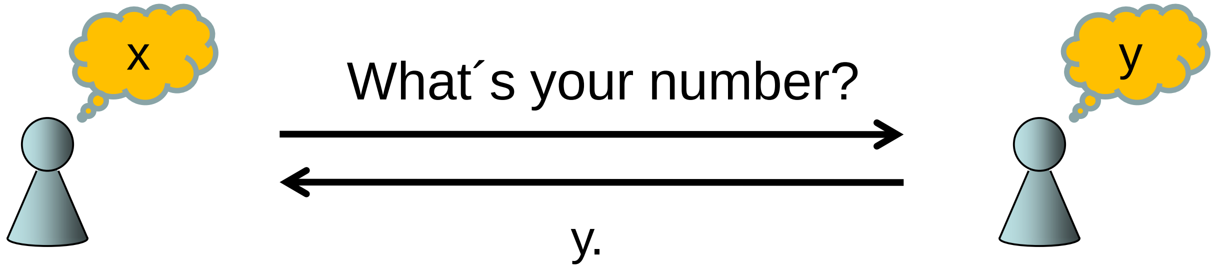
- Is it possible to reach consensus with logarithmic time and work from any state?
- If so, how many adversarial players can be tolerated?



Distributed Consensus

All n players are honest:

- Minimum rule:

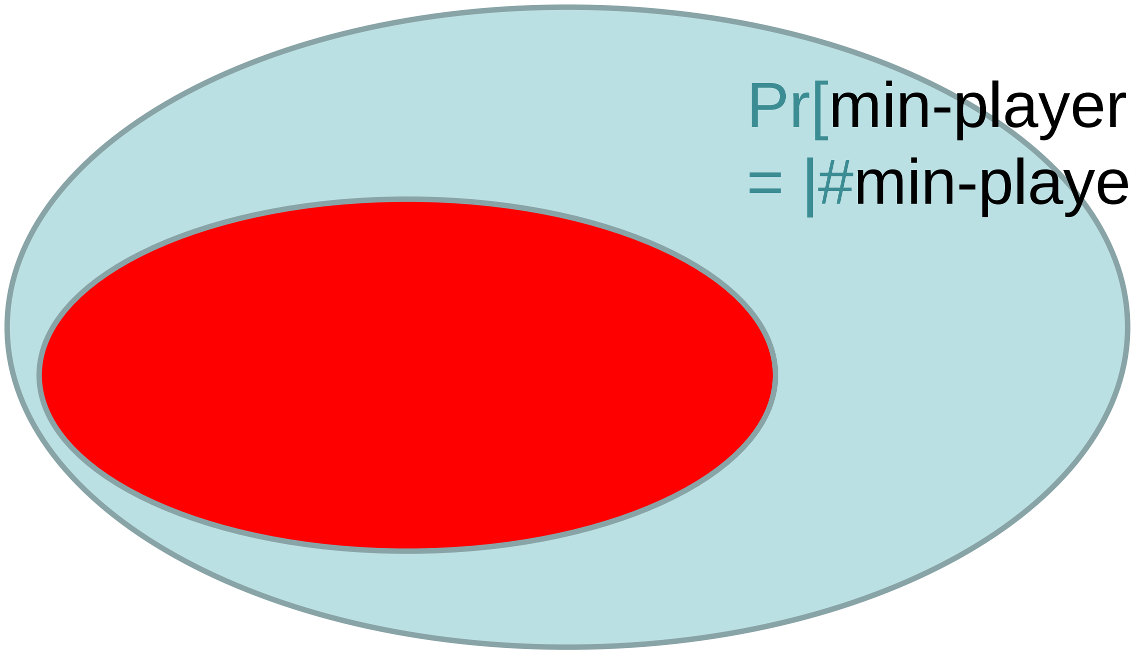


x:=min{x,y} Random player

- After O(log n) rounds (and therefore with O(log n) work), all players have the same number with high probability (i.e., with probability $1-1/n^{\Omega(1)}$)

Distributed Consensus

Minimum rule needs O(log n) rounds, w.h.p.:



Pr[min-player contacted]
= |#min-players|/n

🟥: players with minimum value

Distributed Consensus

Minimum rule needs $O(\log n)$ rounds, w.h.p.

Proof sketch:

- X_t: number of min-players at round t
- Pr[non-min-player becomes min-player]$=X_t/n$
- Hence,

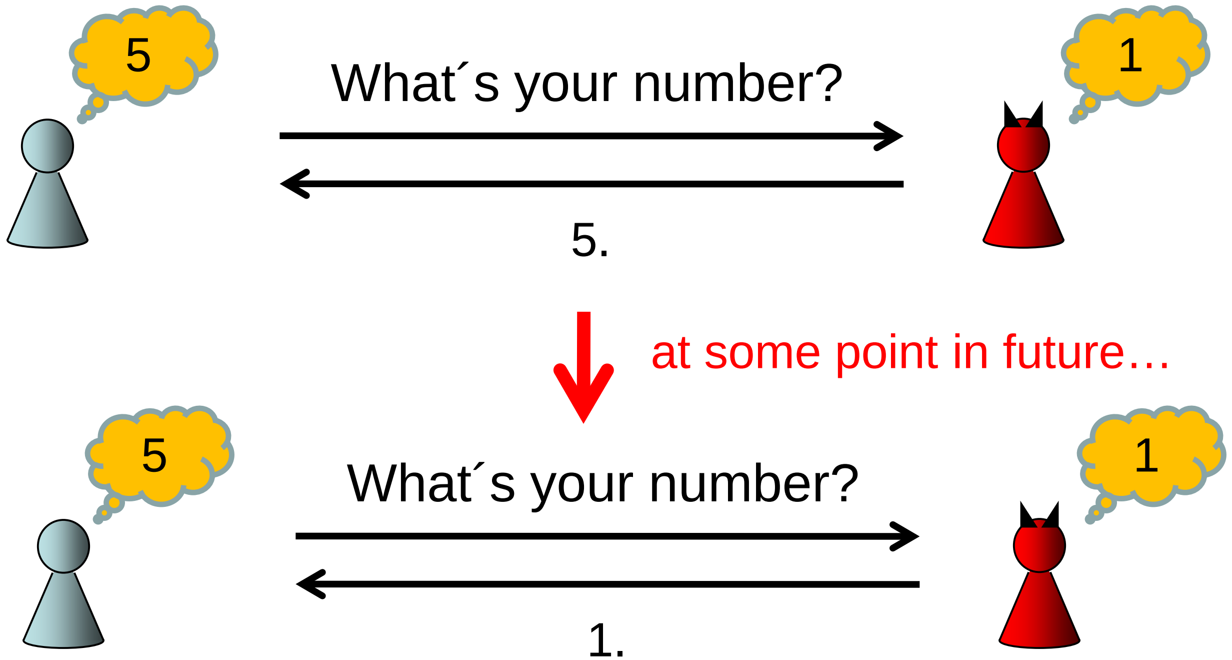
$$\begin{aligned} E[X_{t+1} \mid X_t] &= X_t + (n-X_t)\cdot X_t/n \\ &= 2X_t - X_t^2/n \geq (3/2)X_t \end{aligned}$$

  as long as $X_t \leq n/2$.
- Suppose for simplicity that $X_{t+1} \geq (3/2)X_t$. Then it takes at most $\log_{3/2} n \leq 2 \log n$ rounds till $X_t \geq n/2$.
- Exercise: show that afterwards the number of non-min-players shrinks exponentially on expectation.
- Hence, on expectation, all players store the minimum value after $O(\log n)$ rounds. This can also be shown to hold w.h.p. with the help of the Chernoff bounds.

Distributed Consensus

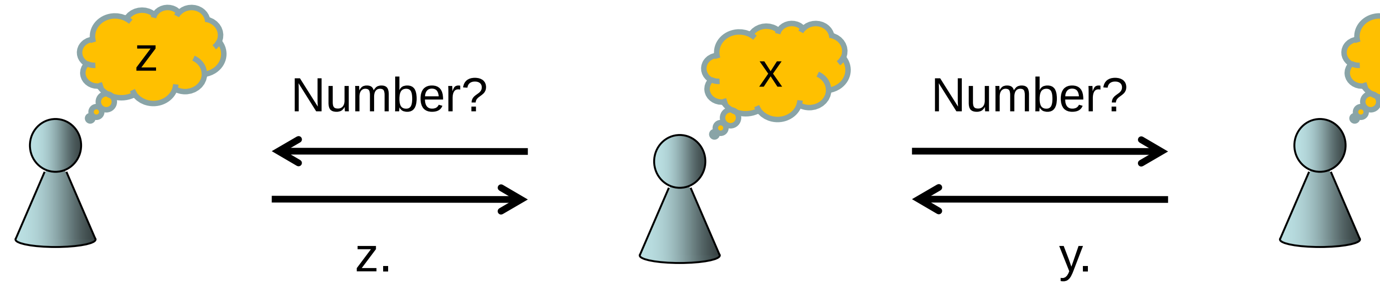
One player adversarial:

- Minimum rule: unlimited runtime.



Distributed Consensus

Better: median rule



Random player x:=median{x,y,z} Random player

- O(log n) rounds w.h.p.: all honest
- O(log n loglog n) w.h.p.: $<\sqrt{n}$ Byzantine

All Players Honest

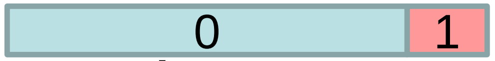
Theorem 7.1: If all players are honest, then the median rule solves the stabilizing consensus problem in $O(\log n)$ rounds w.h.p.

Proof:

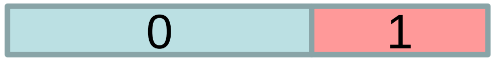
- First, assume that all initial values in $\{0,1\}$
- Median rule = majority rule:
  out of 3 values (own + two from random players) choose value of majority

All Players Honest

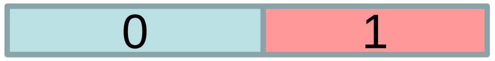
- Δ=|#players with value 0 - #players with value 1|
- Case 1: $\Delta \geq n/3$:
  O(log log n) rounds till consensus w.h.p.



- Case 2: $c\sqrt{n \ln n} \leq \Delta < n/3$
  O(log n) rounds till $\Delta \geq n/3$ w.h.p.



- Case 3: $\Delta < c\sqrt{n \ln n}$:
  - $\Delta < c\sqrt{n}$: Central Limit Theorem: constant probability for $\Delta \geq c\sqrt{n}$
  - $\Delta \geq c\sqrt{n}$: Chernoff : constant probability for $\Delta_{new} \geq (4/3)\Delta$
    Random walk analysis: O(log n) rounds till $\Delta \geq c\sqrt{n \ln n}$ w.h.p.



All Players Honest

Analysis of case 1:

- Suppose that $\Delta_{t_0} \geq n/3$ at the beginning of some time step t_0
- W.l.o.g. assume that there are more 1s then 0s
- X_t: random variable denoting the number of 0s at the beginning of round t
- Note that $X_t=n/2-\Delta_t/2$.
- Pr[value change $0 \rightarrow 0$ in round $t]=1-(1-X_t/n)^2$
- Pr[value change $1 \rightarrow 0$ in round $t]=(X_t/n)^2$
- Hence,

$$E[X_{t+1} \mid X_t] = X_t \cdot (1-(1-X_t/n)^2) + (n-X_t) \cdot (X_t/n)^2$$
$$= (X_t^2/n) \cdot (3-2X_t/n) \leq 3X_t^2/n$$

- Using the Chernoff bounds (set $\mu=3X_t^2/n$ and $\delta=1$), we get

$$\Pr[X_{t+1} \geq 6X_t^2/n] \leq \exp(-3(X_t^2/n)/3) \leq \exp(-X_t^2/n)$$

All Players Honest

Analysis of case 1 (continued):

- Hence, $X_{t+1} \leq 6X_t^2/n$ with high probability, as long as $X_t = \Omega(\sqrt{n \log n})$, i.e., the fraction of players with value 0 goes down from X_t/n to $6(X_t/n)^2$.
- Thus, when starting with $\Delta_{t_0} \geq n/3$ and therefore $X_{t_0} \leq n/3$, it takes just $O(\log \log n)$ rounds till $X_t = \Omega(\sqrt{n \log n})$.
- If $X_t = O(\sqrt{n \log n})$, then $3X_t^2/n = O(\log n)$. Hence, using again the Chernoff bounds,

$$\Pr[X_{t+1} \geq (\log n)^2] = \exp(-\Omega(\log^2 n))$$

- Once $X_t = O((\log n)^2)$, $3X_t^2/n = O((\log n)^2 / n)$. Hence, the Markov inequality yields that

$$\Pr[X_{t+1} \geq 1] = O((\log n)^2 / n)$$

- Therefore, in $O(\log \log n)$ rounds all players have value 1 with high probability.

All Players Honest

Analysis of case 2:

- Suppose that $\Delta_{t_0} \geq c\sqrt{n \log n}$ at the beginning of some time step t_0
- W.l.o.g. assume again that there are more 1s then 0s
- X_t: random variable denoting the number of 0s at the beginning of round t
- Note that $X_t=n/2-\Delta_t/2=n(1/2-\delta_t)$, where $\delta_t=\Delta_t/(2n)$.
- $\Pr[0\rightarrow0$ in round $t]=1-(1/2+\delta_t)^2$
- $\Pr[1\rightarrow0$ in round $t]=(1/2-\delta_t)^2$
- Hence,

$$\begin{aligned}
E[X_{t+1} \mid X_t] &= X_t\cdot(1-(1/2+\delta_t)^2) + (n-X_t)\cdot(1/2-\delta_t)^2 \\
&= (1/2-\delta_t)n\cdot(1-(1/2+\delta_t)^2) + (1/2+\delta_t)n\cdot(1/2-\delta_t)^2 \\
&= (1/2-(3/2)\,\delta_t+2\delta_t^3)n \\
&= n/2 - \Delta_t/2 - ((1/2)\delta_t-2\delta_t^3)n \\
&\leq n/2 - \Delta_t/2 - (1/4)\delta_t n && \text{(using } \delta_t\leq 1/3\text{)} \\
&\leq X_t - (\delta_t/2)X_t && \text{(using } X_t\leq n/2\text{)} \\
&= (1-\delta_t/2)X_t
\end{aligned}$$

All Players Honest

Analysis of case 2 (continued):

- Thus, it follows from the Chernoff bounds (choose $\delta=\delta_t/4$ and $\mu=(1-\delta_t/2)X_t$) that

$$\begin{aligned} \Pr[X_{t+1}\geq(1-\delta_t/4)X_t] &\leq \Pr[X_{t+1}\geq(1+\delta)\mu] \\ &\leq \exp(-\delta^2\mu/3) = n^{-\Omega(c)} \end{aligned}$$

- Hence, w.h.p.,

$$n/2 - \Delta_{t+1}/2 \leq (1-\delta_t/4)(n/2-\Delta_t/2) = n/2 - \Delta_t/2 - (\delta_t/4)(n/2-\Delta_t/2)$$

$$\Leftrightarrow \Delta_{t+1}/2 \geq \Delta_t/2 + (\delta_t n)/8 - \delta_t\Delta_t/8 = \Delta_t/2 + (\Delta_t/2)/8 - \delta_t(\Delta_t/2)/4$$

$$\Leftrightarrow \Delta_{t+1} \geq (1+1/8 - \delta_t/4)\Delta_t \geq (1+1/16)\Delta_t$$

- Hence, after k rounds, Δ has increased by a factor of at least $(1+1/16)^k$ w.h.p.
- Therefore, at most $O(\log n)$ rounds are needed w.h.p. till $\Delta_t\geq n/3$.

All Players Honest

Analysis of Case 3 (sketch):

- An escape attempt is successful if starting with some $\Delta_0 < c\sqrt{n \ln n}$, some $\Delta \geq c\sqrt{n \ln n}$ is reached while satisfying $\Delta_{t+1} \geq (4/3)\Delta_t$ along the way.
- At most $O(\log n)$ rounds are consumed by $O(\log n)$ failed attempts to reach $\Delta = c\sqrt{n \ln n}$ w.h.p.
- Pr[attempt successful] at least a constant >0
- So at most $O(\log n)$ escape attempts fail w.h.p. (proof: exercise)

All Players Honest

- General case: show that after $O(\log n)$ rounds we are left with two different values w.h.p.
- Gravity of process: factor by which presence of its value increases
- We may pretend that all values are distinct (players with same value: append player ID)
- Median: gravity $3/2$
- H_v: $c\sqrt{n \ln n}$ players of largest gravity with value v

Lemma 7.2: If at step t a value v has a player in H_v with gravity $<4/3$, then at step $t+1$ either v is gone or it has a process in H_v of gravity $<4/3$ w.h.p.

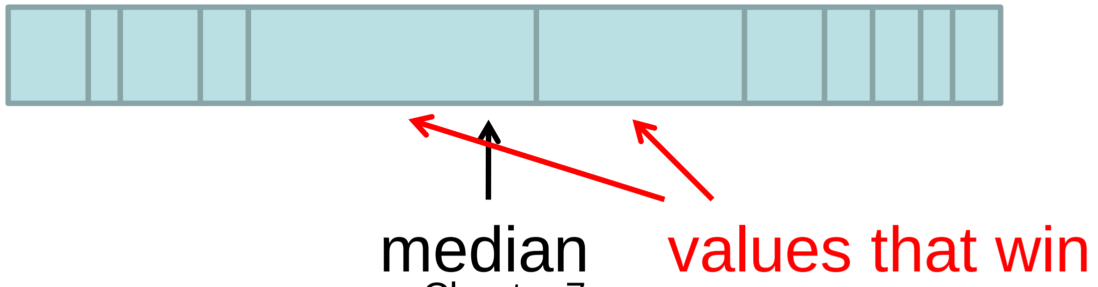
Proof: by reduction to 2-value case

All Players Honest

Lemma 7.3: For any initial value v there is a round $t=O(\log n)$ s.t. one of the cases holds w.h.p.:

- At least one process in H_v has gravity $<4/3$ or v is gone
- $|H_v| = c\sqrt{n \ln n}$

Consequence: after $O(\log n)$ further rounds at most 2 values left w.h.p.

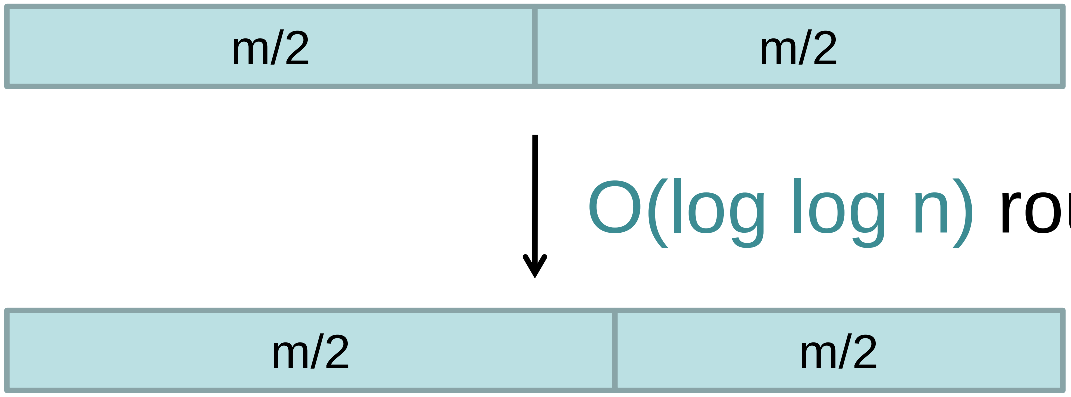


Some Players Adversarial

2-value case (base case): analysis holds despite up to $\sqrt{n}$ Byzantine players

m-value case: (sketch)

- Phase 1: Cut values into two halves

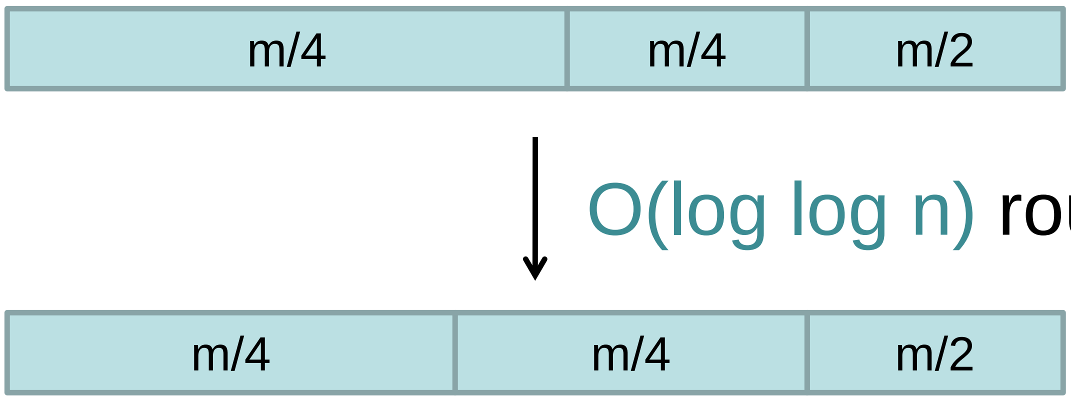


Some Players Adversarial

2-value case: analysis holds despite up to $\sqrt{n}$ Byzantine players

m-value case: (sketch)

- Phase 2: Cut larger half into two quarters

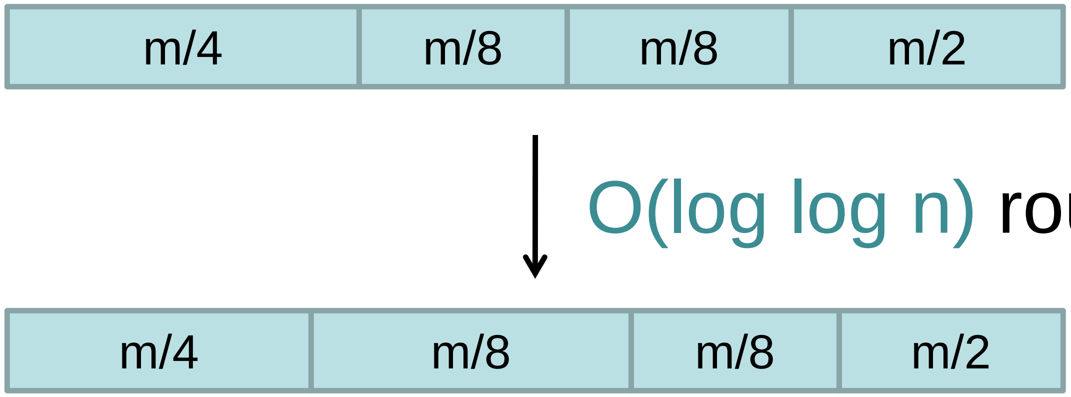


Some Players Adversarial

2-value case: analysis holds despite up to $\sqrt{n}$ Byzantine players

m-value case: (sketch)

- Phase 3: Larger quarter into two eights



Adaptive Adversary

2-value case: analysis holds despite up to $\sqrt{n}$ Byzantine players

m-value case:

- After $\log m$ phases: down to 2 values
- $O(\log m \log \log n + \log n)$ rounds in total w.h.p.

Proofs: see Benjamin Doerr, Leslie Ann Goldberg, Lorenz Minder, Thomas Sauerwald, and Christian Scheideler. Stabilizing consensus with the power of two choices. In *SPAA 2011*, pp. 149-158.

Adaptive Adversary

Remarks:

1. It is known that if the adaptive adversary has $\Omega(\sqrt{n \log n})$ players under its control, then the median rule takes at least poly(n) many rounds to solve the stabilizing consensus problem, w.h.p.
2. This holds even if the adversary just knows the number of players storing a certain value (but not where the values are stored) and can only reset the values to values of its choice in a random set of $\Omega(\sqrt{n \log n})$ players in each round. Why?

Adaptive adversary

Problem: consensus fragile when having many adversarial players.

Example:

- Suppose that $\sqrt{c \cdot n}$ out of n players are adversarial for some constant $c>1$.
- Then Pr[honest player switches to adversarial value] $\geq (\sqrt{c \cdot n}/n)^2 = c/n$
- Hence, E[number of honest nodes switching to adv. value] $\geq (n - \sqrt{c \cdot n}) \cdot c/n \approx c$.
- Thus, on average we can expect a constant number of honest nodes having an adversarial value, so the consensus is not stable for poly(n) rounds.

Solution:

- Each player memorizes the values of the last $\Theta(\log n)$ rounds.
- The output of the player is defined as the median of these values.

Lemma 7.4: If there are at most $\sqrt{n}$ adversarial players, then the correct players reach a consensus that is stable for at least poly(n) many steps w.h.p.

Proof: exercise

Message Delays

Problem: So far we assumed that all message delays are 0. What if we have arbitrary bounded message delays, at least for most of the nodes?

Assumption: (ε,Δ)-bounded DoS-adversary

- For a $(1-\varepsilon)$-fraction of the nodes selected by the adversary, the adversary may pick arbitrary delays up to some Δ, where $\varepsilon>0$ is a small constant. Both Δ and ε are not known to the protocol.
- For the remaining ε-fraction, the adversary may choose not to deliver the messages at all (or choose arbitrarily large delays).

Pseudo Code

Idea: use MIMD-approach

- Start with $T:=T_{min}$.
- Pick two nodes uniformly and independently at random.
- If both answers received within T time, update value according to the median rule and set $T:=\max\{T/(1+\gamma),T_{min}\}$.
- Otherwise, keep value as it is and set $T:=(1+\gamma)^c T$.

In the balanced case for T, there are on average c successful updates vs. one failure.

Pseudo Code

```
Subject Consensus:
    in, v1, v2: Relay
    N: Array of Relay  {neighbors}
    counter, T: Integer
    val, val1, val2: Integer

init(input) $\rightarrow$
    in:=new Relay
    val:=input; T:=$T_{min}$
    counter:=1
    enable(timeout, T)

ack(value) $\rightarrow$
    if counter=1 then val1:=value
    if counter=2 then val2:=value
    counter:=counter+1

timeout: true $\rightarrow$
    if counter>2 then
        val:=median(val,val1,val2)
        T:=max(T/(1+$\gamma$),$T_{min}$)
    else
        T:=$(1+\gamma)^c$T
    counter:=1
    v1:=random(N)
    v2:=random(N)
    v1$\leftarrow$get_val(in)
    v2$\leftarrow$get_val(in)
    enable(timeout, T)

get_val(out) $\rightarrow$
    out$\leftarrow$ack(value)
    delete out
```

Message Delays

Problem: if the (ε,Δ)-bounded DoS-adversary is adaptive, i.e., it can base its decisions where to set large delays or drop messages on the values of the nodes, then it can prevent a consensus to be reached for any constant $\varepsilon>0$.

Message Delays

Attack strategy of an adaptive (ε,Δ)-bounded DoS-adversary:

- Suppose that we have case 2 of the median rule, i.e., for the given round t, $\sqrt{c\ n \ln n} \leq \Delta_t < \varepsilon n/5$ for a (sufficiently large) constant c and $\varepsilon>0$ as specified for the adversary, and w.l.o.g. there are fewer nodes with 0 than with 1.
- If the adversary blocks $5\Delta_t$ nodes with value 1, then the difference Δ'_t for the remaining nodes is $4\Delta_t$, and now there are more nodes left with 0 than with 1.
- Let $\delta_t=\Delta'_t/(2n)$ and X_t be the number of non-blocked nodes with value 1 in round t.
- From case 2 we know that $E[X_{t+1} \mid X_t] \leq (1-\delta_t/2)X_t$ (which still holds if $\varepsilon>0$ is sufficiently small). For simplicity, suppose that $E[X_{t+1} \mid X_t] = (1-\delta_t/2)X_t$ and that this also holds w.h.p.
- Then there will be $\delta_t X_t/2$ fewer non-blocked nodes with 1 than before, which means that there are $\delta_t X_t/2$ more non-blocked nodes with 0 than before.
- If $\varepsilon>0$ is small, it follows that $\delta_t X_t/2 \approx \delta_t n/4 = \Delta'_t/8 = 4\Delta_t/8 = \Delta_t/2$.
- Adding the blocked nodes back, we obtain a difference of $\Delta_{t+1}=0$, so the adversary can get the system back to a balanced state.

Oblivious Adversary

How about an oblivious (ε,Δ)-bounded DoS-adversary?

- An adversary is called oblivious if it does not know the current state of the system.

Example:

- If the median protocol is run in the TCL, then the adversary has no means of finding out which TCL stores which value.
- So the adversary just degrades to an oblivious adversary. In fact, it cannot even change the values. All it can do is to shut down or block the TCL or delay messages. (Replay attacks can be prevented by sending a random value with a request, which must be part of the reply.) Hence, it is just an oblivious DoS-adversary.

Conjecture: There is a constant $\varepsilon>0$ so that the median protocol is robust against any oblivious (ε,Δ)-bounded DoS-adversary.

Physical Clock Synchronization

Problem: once message delays are >0, physical clock synchronization cannot simply be reduced to the standard consensus problem since message delays cause errors in the received physical clock values.

Solution:

- Let δ_1 and δ_2 be the average message delays of the two nodes that were randomly chosen in the given round (see slide 6), and let $\delta=\max\{\delta_1,\delta_2\}$.
- Let Θ_1 and Θ_2 be the clock skews of these nodes.
- If $\theta:=|\mathrm{median}(0,\Theta_1,\Theta_2)|\leq\delta$, keep the physical clock value PH_v of v as it is, otherwise change it to $PH_v+\theta$.

Physical Clock Synchronization

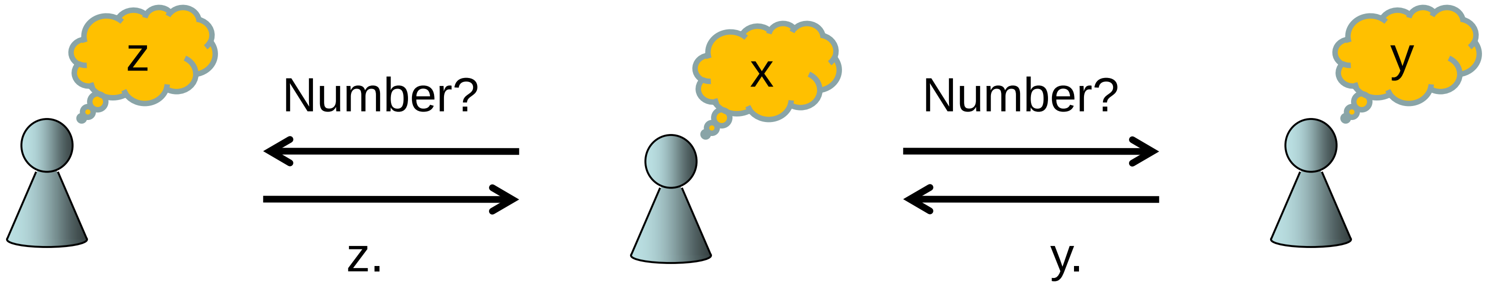
Solution:

- Let δ_1 and δ_2 be the average message delays of the two nodes that were randomly chosen in the given round (see slide 6), and let $\delta=\max\{\delta_1,\delta_2\}$.
- Let Θ_1 and Θ_2 be the clock skews of these nodes.
- If $\theta:=|\text{median}(0,\Theta_1,\Theta_2)|\le\delta$, keep the physical clock value PH_v of v as it is, otherwise change it to $PH_v+\theta$.

Conjecture: For any oblivious (ε,Δ)-bounded DoS-adversary with $\varepsilon>0$ being a sufficiently small constant, the median rule together with a MIMD approach and the solution above ensures that in $O(\Delta \log n)$ time all clocks are synchronized up to an additive $O(\Delta)$.

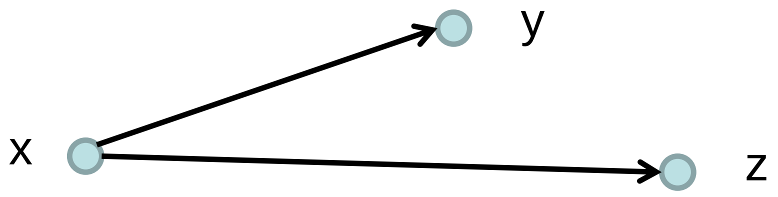
Other Rules

Nearest neighbor rule:



Random player x:=nearest{y,z} Random player

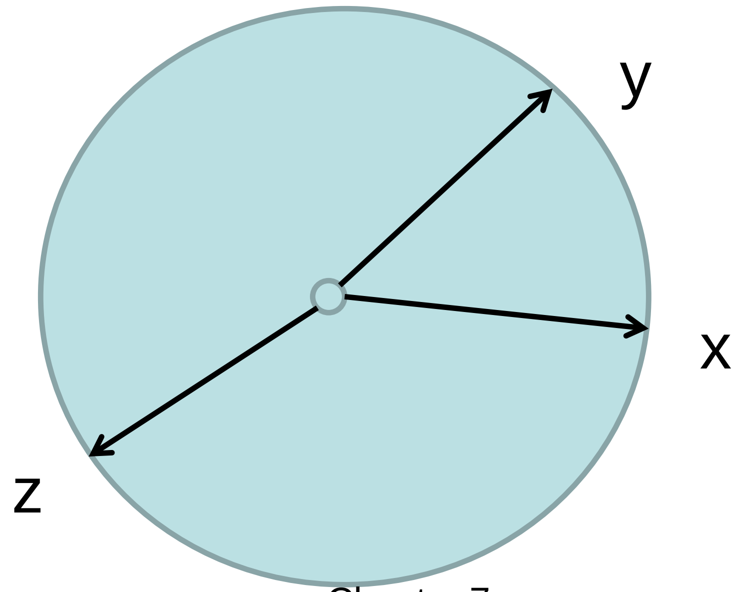
nearest{y,z}: the one with min distance to x



Other Rules

Advantages of nearest neighbor rule:

- Time synchronization for periodic actions: no absolute clock value needed any more!

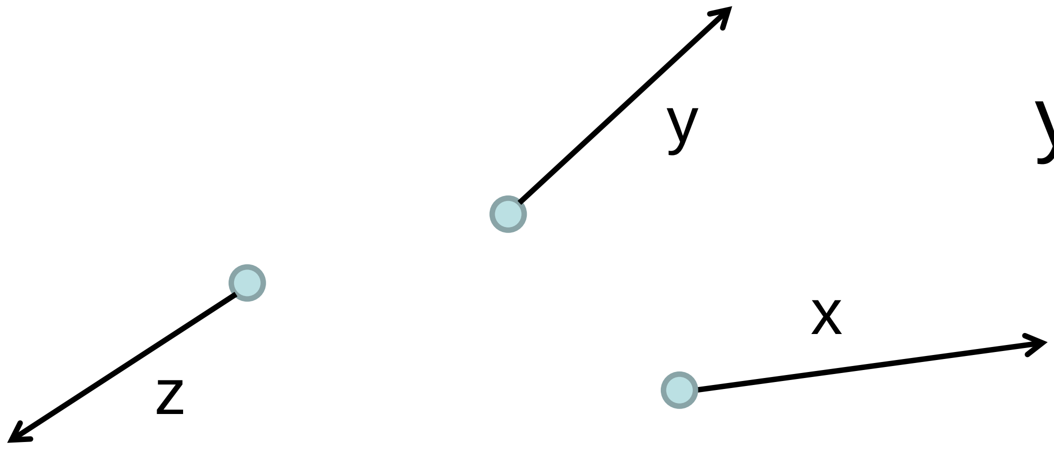


y wins

Other Rules

Advantages of nearest neighbor rule:

- Robot alignment:
  no compass needed!

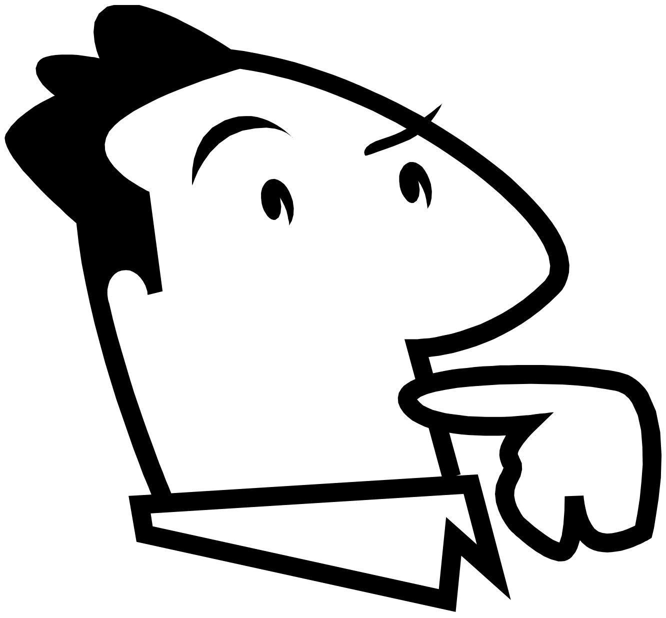


y wins

Nearest Neighbor Rule

Problem:

- Best bound shown so far on convergence time for line: $O(n^2 \log n)$
- No reasonable bound known for cycle or higher dimensions…



Questions?