Admin:

→ Final next Monday!

→ Will have practice exams & review sessions. Stay tuned!
make sure to work through them

Today: Distributed computing

→ Leader election

→ Maximal independent set
(not "maximum")

So far: Thinking of computation as done on one processor/machine

But: That's not exactly how computation is done today

Computing paradigms:

→ parallel computing (think: multiple processor cores)

(not covered today ⇒ see 6.816)

⇒ how to parallelize a task? (Can it be parallelized?)
E.g., digging a hole vs. digging a ditch.
Can we benefit from having more (identical) workers?

→ distributed computing (think: computer networks/internet)

(today Also: 6.852)

⇒ how to cooperate to solve a joint task? even if some are NOT cooperating

Setup:
- → n processors/players
- → each has input x_i
- → want to compute, for each processor i,

$$y_i = f_i(x_1, \dots, x_n)$$

E.g. x_i = numbers
y_i = max of these n numbers

Crucial: Each f_i might depend on <u>all</u> inputs

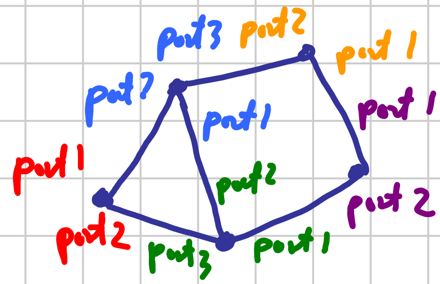
⇒ Cooperation essential

(today will ignore situation when some processors are faulty or adversarial)

How to talk?

(today also: 6.852)

① **Message passing model:**
- → processors connected in an undirected graph
- → in each round: can send/receive messages along edges

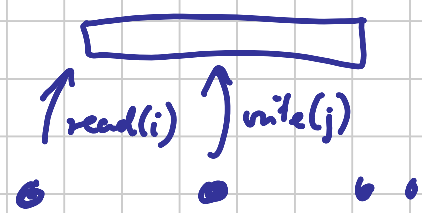


→ each proc. know their I/O ports by local name
↑ arbitrary

② **Shared memory model:**

(not today but see 6.852)

- → Processors communicate by reading/writing to shared memory in each round



Think: Msg. board

Important: We assume synchronous model here, i.e., things happen in rounds

Leader election:

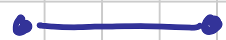
"protocol" emphasizes the focus on communication!

Goal: Run a protocol (algorithm) so as at the end exactly one processor outputs "I am the leader"

Sounds simple but how to do that?

Warning: Impossible if the protocol is deterministic & processors are truly identical

Simplest counterexample: $n = 2$



Observe:

Round 1: Same starting states
$\Rightarrow$ Same outgoing message
$\Rightarrow$ Same incoming message

Round 2: Same state
$\Rightarrow$ Same outgoing message
$\Rightarrow$ Same incoming message

Round 3: ...

$\Rightarrow$ Either goes on forever or both processors declare "I am the leader"

$\Rightarrow$ Protocol fails $\boxed{\xi}$

Fundamental problem: Lack of "symmetry breaking" mechanism

Solution I: Make processors non-identical

⇒ each processor has a unique ID
(email, IP address, MAC, ...)

Simple protocol: (Assume graph is connected)

→ Each processor i has a local variable max_i

→ In each round:

→ Send max_i (to all neighbors)

→ Update $max_i := \max(max_i, \{\text{incoming msgs}\})$

→ After Δ rounds, (Δ = upper bound on diameter of the graph)
if $max_i := ID_i$, output
"I am the leader" (otherwise, do nothing)

Solution II: No unique IDs but have randomness

⇒ can use randomness to "manufacture" UIDs!

Protocol:

→ Choose ID_i at random from the set $\{1, 2, \ldots, K\}$ for some $K \geq 1$

→ Run the protocol for unique ID setting

Clearly: If all ID_is end up being unique ⇒ protocol correct

What is the probability of a collision?

Note: $\forall_{i \neq j}$ $\Pr[ID_i = ID_j] = \dfrac{1}{K}$

$\Rightarrow$ By union bound over all $\binom{n}{2}$ possible pairs

$$\boxed{\Pr[\text{collision}] \leq \sum_{i \neq j} \Pr[ID_i = ID_j] = \binom{n}{2} \frac{1}{k} \leq \varepsilon}$$

if $\boxed{k \geq \varepsilon^{-1} \binom{n}{2}}$ for some $\varepsilon > 0$

$\Rightarrow$ The protocol succeeds with prob. $\geq 1 - \varepsilon$

BUT: Processors do not know if they succeeded!

$\Rightarrow$ Monte Carlo algorithm

Fix: To get a Las Vegas algorithm:

Detect & Repeat (if needed)

$\Rightarrow$ Before "officially" annoucing the leader, run a check if there is only one leader if not, repeat the protocol

checking that can be done deterministically without UIDs, by simple broadcast for Δ rounds

$\Rightarrow$ In expectation, need $\leq \frac{1}{(1-\varepsilon)}$ repeats

$\Rightarrow$ Expected # of rounds $O\left(\frac{\Delta}{(1-\varepsilon)}\right)$
but it is always correct, as needed

Note: We ignore here time needed for local computations, as we are focused on complexity of reaching a consensus, and this is measured via # of rounds of communication.

Maximal independent set (MIS) problem

(NOT "maximum")

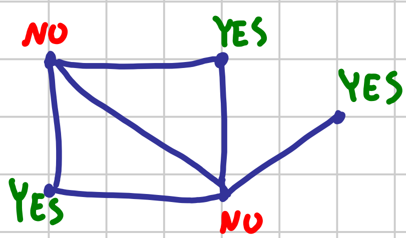
"maximal" = such that we can't add any new element to it without the need to "swap out" another element

Goal: A protocol such that at the end each processor outputs a Yes/No decision & the yes-decision processors form a maximal independent set

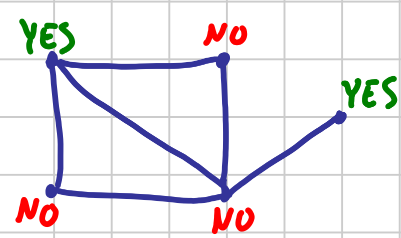
→ independent ⇒ no two yes-processors are neighbors

→ maximal ⇒ can't add any more yes-processors without violating independence

Note:



maximal & maximum independent set (size = 3)



maximal but NOT maximum independent set (size = 2)

Recall: Maximum independent set is NP-hard

But: Maximal independent set (MIS) is in P

(As in distributed computing we ignore local computation time, in principle, we can solve NP-hard problems too! But this would be an "abuse" of the model.)

Simple protocol for MIS:

→ Do leader election → add leader to MIS & make its neigh. inactive

→ Repeat $\boxed{\Rightarrow O(n \cdot \Delta) \text{ rounds}}$

Can we do better?

Luby's (randomized) MIS protocol

Setup: NO UIDs but we have randomness

Protocol:
- → All processors are "active" in the beginning
- → Protocol proceeds in phases. Each phase = 2 rounds
- → In each phase:

Round 1:
- → Choose random value $r_i \in \{1, 2, \ldots, k\}$ & send to all neighbors
- → Receive values from neighbors
- → If received values are all $< r$, then join the MIS (i.e., output YES)

Round 2:
- → If you joined MIS, announce to all neighbors
- → If you received such an announcement, decide to NOT join MIS (i.e., output NO)
- → If you decided YES/NO in this phase, become inactive

Observe:
- → Final set satisfies independence
  (since we join MIS only if value of r is uniquely maximal among neighbors;
  ⇒ when this happens, all neighbors output NO)
- → Final set is maximal
  (since only way to become inactive is if you or one of your neighbors joins MIS)

Remaining question: How many rounds until done?
(= all processors inactive)

Lemma: Need (only!) $\boxed{O(\log n)}$ rounds to terminate
(provided $K \gg n^3$, e.g., $K = n^4$ is fine)

In fact: Terminate in $4 \log n$ phases with prob. $\geq 1 - \frac{1}{n}$

Proof: (Will only argue for a line graph

But: the same ideas + more careful calculations give the claim for general graphs)

Note: In each phase, if $i \neq j$ then $\Pr[r_i = r_j] = \frac{1}{K} \ll \frac{1}{n^3}$

$\Rightarrow$ union bounding over all pairs & $4 \log n$ first phases

$$\Pr[r_j = r_i \text{, for some } i \neq j \text{ in phase } l \leq 4 \log n] \leq \frac{4 \log n \cdot \binom{n}{2}}{K} \leq \frac{2n^2 \log n}{K} \ll \frac{1}{n}$$

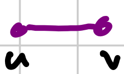
$\Rightarrow$ Wlog can assume all r_is are always distinct! if $K \geq n^4$

Now: Say an edge (u,v) is active iff both u & v are still active

Key claim: For any edge (u,v) & any phase in which (u,v) starts as active

$$\boxed{\Pr[(u,v) \text{ becomes inactive}] \geq \frac{1}{2}}$$

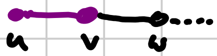
Proof: Consider cases:

① 

(no active edges incident to (u,v))

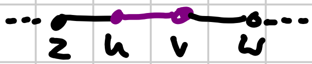
$\Rightarrow$ either $r_u > r_v$ or vice versa (with prob. $1 \geq \frac{1}{2}$)

$\Rightarrow$ either u or v output Yes $\Rightarrow$ (u,v) becomes inactive

②  (one active edge incident to (u,w))

$\Rightarrow$ with prob. $\frac{1}{2}$, $r_u > r_v$ $\Rightarrow$ u outputs Yes

$\Rightarrow$ (u,v) becomes inactive with prob $\geq \frac{1}{2}$

③  (two active edges incident to (u,v))

$$\Rightarrow \left.\begin{array}{l} Pr[r_u < r_v \ \& \ r_w < r_v] = \frac{1}{4} \\ Pr[r_u > r_v \ \& \ r_z < r_u] = \frac{1}{4} \end{array}\right\} \text{disjoint events}$$

$\Rightarrow$ $Pr[u \text{ or } v \text{ is a local max}] = \frac{1}{2}$ $\Rightarrow$ (u,v) becomes inactive w prob. $\geq \frac{1}{2}$

∎

Using the key claim:

$$\forall_{(u,v)} \quad Pr[(u,v) \text{ is still active after } t \text{ phases}] \leq \left(\frac{1}{2}\right)^t$$

$\Rightarrow$ union bounding over all $|E|$ edges:

$Pr[\text{protocol did not terminate after } t \text{ phases}] =$

$= Pr[\text{Some } (u,v) \text{ still active after } t \text{ phases}] \leq$

$$\leq |E| \left(\frac{1}{2}\right)^t \leq \frac{n}{2^t} \leq \frac{1}{n^3} \ll \frac{1}{n} \text{ if } t \geq 4 \log n$$

∎