Class 7
6.852
Fall, 01

Eric Chalk
HW2b

6.3.3, 6.5, 6.6,

Reading: 7, 8.1

Next: Rest of 8

Last time introduced Byz agr. problem.

EIG alg: $f+1$ rds $n > 3f$ procs, exp. comm (bits)

Proof that $3f+1$ necessary.

Today go back + relate BA for binary values vs. arb. value set

Then consider connectivity issues + other problems related to consensus.

(and on to asynch systems.)

<u>Agreeing on a bit vs. agreeing on a more general value</u> (in V)

Essentially the same problem!

It may seem that agreeing on a value from a big set, e.g. $\{0, \ldots, 99\}$, might introduce its own complications.

But not really: Can use a <u>binary BA</u> alg as a <u>subroutine</u> for general BA, with <u>not much overhead</u>: 2 extra rounds, a few msgs ($2n^2$, $O(n^2 b)$ bits)

Says that the core problem is simply agreeing – regardless of what we're agreeing on.

Again, need $n > 3f$.

<u>Proc has</u>
- x, init value
- y, z } intermediate values
- vote – input for the binary "subroutine"

<u>rd 1</u>: <u>Send x to all</u> (incl self – can simulate locally, nice for uniformity)
Actually, need to do this in EIG alg. too

If in set received, there are $\geq n-f$ copies of some $v \in V$,
then $y := v$
else $y :=$ null

(only retain clear preferences)

<u>rd 2</u>: <u>Send y to all</u>

If in set received, there are $\geq n-f$ copies of some $v \in V$,
then vote $:= 1$
else vote $:= 0$

Also, <u>set</u> $z :=$ the non-null value that occurs most often among received msgs. (ties arbitrary; if no such value, z undefined)

Then run BA on the vote values.

If decide 1 & z defined, decision := z

else decision := default v_0

Key Lemma 1: At most 1 value $v \in V$ is sent in rd 2 msgs by nonfaulty procs.

Pf: Counting argument.

Suppose there are two, i & j (nonfaulty) send v & w, resp.

Then i receives $\geq n-f$ rd 1 msgs cont. v

j receives $\geq n-f$ rd 1 " " w

But since $\leq f$ faulty, j receives $\geq n-2f \geq f+1$ rd 1 msgs cont. v.

But $(f+1)+(n-f) > n$, too many received msgs.

$\boxtimes$

Theorem: Solves BA.

Pf: Termination: ~~Easy~~ Obvious.

Validity: Easy. If all nonf. start with v, all bcast v at rd 1,

all set y := v

all set vote := 1, z := v

Then validity for subroutine implies decide v.

Agreement:

If subr. decides 0, then v_0 decided by all.

If subr. decides 1, then

By validity for subroutine, some nonfaulty i must begin subroutine with vote = 1.

So proc. i receives $\geq n-f$ rd 2 msgs cont. some particular v.
So " " " $\geq n-2f$ rd 2 msgs cont. v from nonfaulty procs.

So any other nonfaulty j also receives $\geq n-2f$ msgs with v (from nonf. procs).

Now by Lemma 1, only faulty procs can send value $\neq v$ in rd 2.
So j receives $\leq f$ msgs cont. values $\neq v$, in rd 2.
$n-2f > f$, so v is the most frequent value.

So j sets $z := v$.
This means j decides v.
So, all nonf. decide v.

$\boxtimes$

This result can be thought of as showing a kind of "reducibility" between problems.

V-BA $\leq$ binary BA

But I don't have a formal def for "reducibility".

Will see the "reducibilities" later in course - usually, transformations of algorithms that run in one model so they can run under a weaker set of assumptions.

General Graphs

$n > 3f$ isn't the whole story.

Consider

4, but can't tolerate 1 fault
Because center can disconnect.

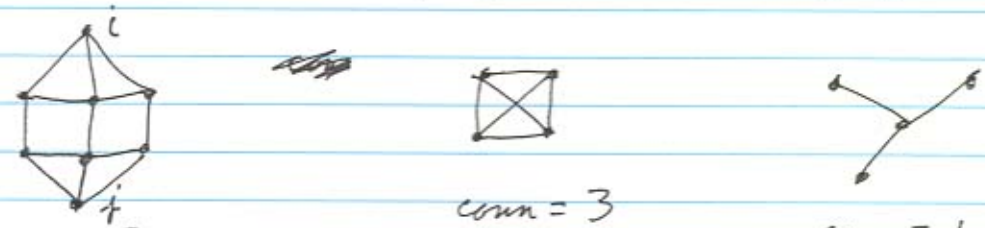
The whole story is:

Theorem: BA solvable in n-node graph, G, tolerating f faults iff both:

1. $n > 3f$
2. $\underbrace{\text{conn}(G)}_{} > 2f$

defined to be min # of nodes whose removal results in either a disconnected graph or 1-node graph

Ex:



Pf: if Suppose $n > 3f$ and $\text{conn}(G) > 2f$

Then can simulate total connectivity alg.

The key is how to implement reliable comm. from a node i to another node j.

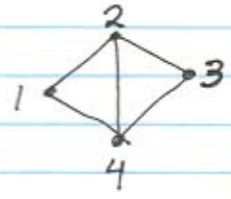
Rely on Menger's Theorem, which says that a graph is c-connected (that is, has conn $\geq c$) iff each pair of nodes is connected by at least c node-disjoint paths.

(see i & j in exaple 1 above)

So have $\geq 2f+1$ disjoint paths.
So send msg, send on all these paths.
Majority must be correct.
So take majority value.

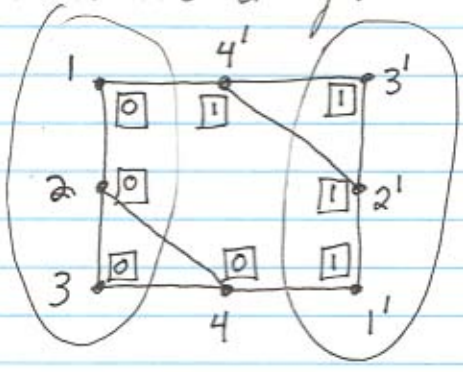
only if:
Already showed need $n > 3f$.
Remains to show $\text{conn}(G) > 2f$
Just do simple case that gives the key idea:
$\text{conn} \leq 2$ + $f=1$ impossible

Canonical example:

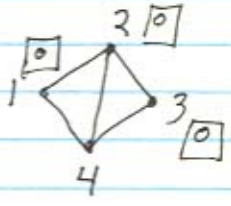


1 + 3 can be disconnected by removing 2 + 4 (2-connected)

2 copies:



Consider 1, 2, 3. Looks to them like



So decide 0.

Consider 1', 2', 3': Decide 1

Consider 1, 4', 3': Must agree, contradiction.

As before, can generalize pf to $\text{conn} > 2f$, or use reduction.

Weak BA

It turns out that the bounds $n > 3f$ + conn $> 2f$ are fundamental for consensus-style problems with Byz. failures.

E.g. in timed (partially synch) case, to do clock synch in presence of some number of Byz. faulty procs, same bounds hold.

Byz. firing squad (HW)

Proofs generally similar

E.g. consider weak agreement - same agreement, term as BA

But weaken validity to say:

If there are no faulty procs + all procs start with v, then v is the only possible decision value.

A similar condition was used for the 1 (commit) case for the two Generals problem.

Constrains the situations where the decision has to go a certain way.

However, still impossible.

Though now harder to prove impossibility

Pf: If: follows from ordinary BA

Only if: Just do 3 vs 1 as ex, $\left.\begin{matrix} 3f \text{ vs } f \\ \text{connectivity} \end{matrix}\right\}$ LTTR

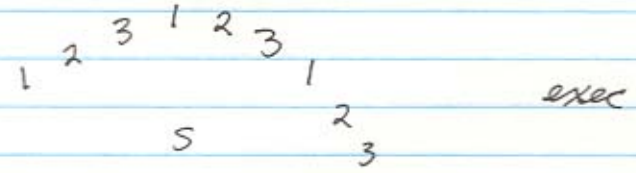
Assume $\exists A$, procs 1, 2, 3

$\left.\begin{matrix} \alpha_0 \\ \alpha_1 \end{matrix}\right\}$ execs in which all start with $\left\{\begin{matrix} 0 \\ 1 \end{matrix}\right\}$ no failures

Both result in decisions.

Let r = u.b. on rounds for all procs to decide, in both.

S: Paste $2r$ copies of A into $6r$-process ring



Start top half with $\boxed{0}$
Bottom half with $\boxed{1}$

Can show any 2 adjacent procs in S must agree, in S exec
(because they must agree in A)

So all agree.
WLOG say decide 1.
Now consider block of $2r+1$ consecutive 0s.
Claim all but ends indist. from all-0 exec α_0 for 1st rd
" " 2 at ends " " " α_0 " 2nd rd
⋮
Mid indistinguishable from α_0 for r rds.
But that's enough for it to decide 0, contradicts the fact that it decided 1.

More on consensus in synch model

Some variations on this problem that have been studied

- k-agreement (can have k values, not just 1)
- approx. agreement
- commit

I'll just cover quickly in class.

k-agreement

Generalizes ordinary stopping agreement by allowing $k > 1$ different decisions instead of just one

Practical motivation? Sort ~~resource alloc~~ allocating shareable resources.
E.g. agree on small # of bcast frequencies for receiving bcasts.

Really motivated by math considerations:
Elegant theory + interesting, tight bounds.

Prob. statement

n-node complete undirected, stopping failures
inputs, decisions in V

agreement: $\exists$ subset $W \subseteq V$, $|W| = k$, all dec. values in W

validity: Any decision value is initial value of some process.
(That corresponds to stronger validity I mentioned for 1-agreement.)

termination: All nonf. eventually decide

Alg: Flood-Min (like Flood-Set for ordinary agreement)

Each maintains min value seen, initially own value.
Decide on final min value.
If do this for <u>f+1 rounds</u>, solves ordinary stopping agreement (1-agreement)

For <u>k-agreement</u>, just do for $\lfloor \frac{f}{k} \rfloor + 1$ rounds. (similar to Flood Set)

Not obvious – allowing k values <u>divides the runtime by k.</u> !

<u>Correctness:</u>
<u>M(r)</u> = set of min-vals of active procs after r rounds
This set can only <u>decrease</u> over time.
(subset sense)

<u>Lemma 1:</u> $M(r) \subseteq M(r-1)$ for all r, $1 \le r \le \lfloor \frac{f}{k} \rfloor + 1$.
<u>Pf:</u> Any min value after r is <u>someone's</u> after r-1

<u>Lemma 2:</u> If at most d-1 procs fail during rd r, then $|M(r)| \le d$.

(at most d different values left, if d-1 fail.)
Generalizes result for d=1 : if none fail then all same.

<u>Pf:</u> Suppose for contrad. that $\begin{cases} \le d-1 \text{ failures} \\ |M(r)| > d. \end{cases}$

Let $\begin{cases} \underline{m} = \underline{\max}(M(r)) \\ \underline{m'} = \text{any other}, \ m' < m \end{cases}$

element of M(r)

Then $m' \in M(r-1)$ by Lemma 1; let i be a process active after $r-1$ with m' as its min.

If i doesn't fail in rd r then every process receives m'.

Contrad. the fact that some active process has $m > m'$ as its min.

So, i must fail.

But this is true for every ~~this~~ value of $M(r) \neq m$: ~~the~~ There ~~So there must~~ must be some process active after rd $r-1$ with that initial value, who fails in rd r.

So, there must be $\geq d$ failures.

Contrad.

$\boxtimes$

Thm: Correct

Pf: Term., validity easy

Agreement: By contradiction. Suppose in an exec with $\leq f$ failures, have $> k$ decision values.

Then # of min-vals for active procs after $\lfloor \frac{f}{k} \rfloor + 1$ rounds is $\geq k+1$.

That is, $\left| M\left(\lfloor \frac{f}{k} \rfloor + 1 \right) \right| \geq k+1$.

Then by Lemma 1; $|M(r)| \geq k+1$ for all r, $0 \leq r \leq \lfloor \frac{f}{k} \rfloor + 1$

(can only shrink)

Then by Lemma 2, at least k procs fail in each round.

Total of $\geq \underbrace{\left(\left\lfloor \frac{f}{k} \right\rfloor + 1\right)}_{\#\text{ of rds}} \underbrace{k}_{\text{failures in each rd}}$ failures

$> f$, contrad

$\boxtimes$

Lower bd

Quick sketch.

Interesting result statement: $\left\lfloor \frac{f}{k} \right\rfloor + 1$ is exact lower bd.

Interesting techniques: Though ~~ff~~ the general theory for these + further applications of them are well beyond the scope of this class.

Assume $|V| \geq k+1$ (else problem is trivial).

Recall pf of ~~lower~~ $f+1$-round lower bd for 1-agreement.

Chain of execs, for assumed alg.

For more allowed decisions, a simple chain doesn't work:

Can't say unique decision value in each exec., same in 2 consecutive:

$$\alpha_i \overset{p}{\text{———}} \alpha_{i+1}$$

$$\text{dec} = v \qquad \text{dec} = v$$

E.g., for $k=2$, could have 2 different values here without violating agreement.

However, a more complex, k-dimensional structure will work.

Prove impossibility:

<u>Assume:</u> n-process alg. for k-agreement, $\le f$ failures.
Halts in $r \le \lfloor \frac{f}{k} \rfloor$ rounds

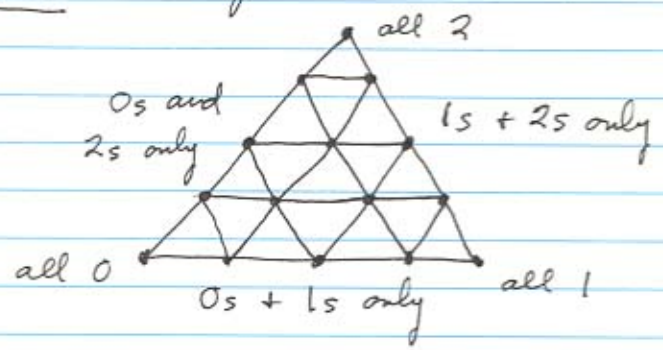
$n \ge f + k + 1$ (at least $k+1$ procs don't fail in any exec)
All nonf. procs decide at end of rd r.

All send to all at all rounds.
$V = \{0, \ldots, k\}$

Show contrad. by producing exec. with $\ge k+1$ different decisions.
<u>Pf sketch:</u> Use <u>k-dim collection of execs</u> rather than 1-dim.
adjacent execs indist. to some nonf. process.

<u>Bermuda $\triangle$:</u> (algorithm vanishes)

<u>$k = 2$</u>



Nodes corresp. to execs.
In general, for arbitrary k, use <u>"k-dim simplex"</u> (alg. topology)
1-dim simplex = line
2-dim = $\triangle$
3-dim = tetrahedron, etc

$\triangle$ Grid points, connect to give little $\triangle$s (triangulation)
Extends to k dims.

<u>Now assign exec to each vertex:</u>

Corners: No failures, all start with same value.

Edges (+ higher-dim "faces"): Init values chosen from those of the corners of the face. Can have failures.

Interior: Mixture of all inputs.

Assignment done in such a way that: The execs morph gradually as we move in each direction:

Just $\left\{\begin{array}{l}\text{remove or add a msg}\\ \text{fail or recover a process}\\ \text{change initial value}\end{array}\right\}$ as before.

Also, to each vertex:

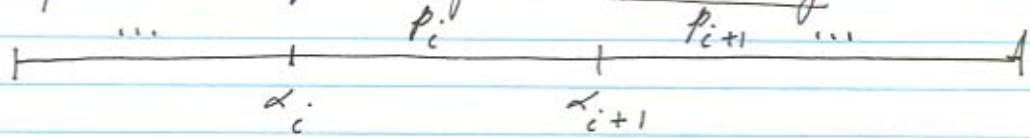
assign index of some process nonf. in execution of that vertex.

Do this "consistently".

So that for every tiny Δ, $\exists$ a single execution α with $\leq f$ faults, that is "compatible" with execs + procs assigned to its corners:

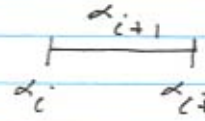
1. all these procs are nonfaulty in α
2. If (α', i) assigned to some corner, then $\alpha \overset{i}{\sim} \alpha'$.

Ex: Do this for our previous chain arg:



Set p_i nonf in α_i + α_{i+1}, p_{i+1} nonf in α_{i+1} + α_{i+2}, etc.

Assign p_i to α_i, p_{i+1} to α_{i+1} ...

Let the "new exec" assigned to  be $\beta = \alpha_{i+1}$

Then p_i + p_{i+1} both nonfaulty in α_{i+1} ✓

Also, $\beta \overset{p_i}{\sim} \alpha_i$ and $\beta \overset{p_{i+1}}{\sim} \alpha_{i+1}$.

This assignment requires the assumption $r \le \lfloor \frac{f}{k} \rfloor$, that is, $f \ge rk$.

Why: Construction requires us to fail r processes (1 per rd) to construct the chain in each dimension k dimensions, so rk failures.

More complicated, but basically same idea as chain constr.

Now color each vertex with a "color" in $\{0, \ldots, k\}$

$color(\alpha, i)$ = i's decision value in α

Properties:

1. Colors of the $k+1$ corners of ~~any little~~ big Δ all distinct (by validity)

2. Color of each pt on any edge (or higher-dim face) is the color of one of its corners. (by validity)

Now pull a math trick out of a hat:

Coloring of a triangulated simplex having these 2 properties has a special name in algebraic topology: Sperner Coloring.

Sperner's Lemma says any such coloring has some tiny simplex where $k+1$ corners are colored by all $k+1$ colors.

But here this yields an execution with $k+1$ values.

Contradicts k-agreement! $\boxtimes$