Reading: 6.2, 6.3, 6.7
Next: 6.4, 6.6, Ch. 7 (s/kw)

Last time:
Finished synchronous network alg.
Started consensus, first link failures, with 2 general impossibility result.

Today: Stopping failures
Agreement with stopping failures: alg w an impor. result
Start Byzantine

Recall stopping problem def:
- m nodes connected, undigraph
- inputs, outputs in V
- Synch model
- f failures

Agreement: No 2 decode
Validity: If all start with v then v is only possible decision
Termination: All non faulty proc eventually decide
Note on stopping vs. Byzantine versions of problem

Byz. agreement algorithm does not necessarily solve stopping agreement.

To stopping, all processes that decide, even the faulty ones that later fail, must agree (uniformity).

But such a condition wouldn't make sense for the Byzantine case.

(in some special cases, e.g. when all decisions have to happen at the same round, the implication holds).

Complexity measures:

- **time**: rounds till all non-faulty decide
- **comm**: # of msgs. (stopping case: count all sent
  # of bits) (Byz. case: count those sent by non-faulty processes)

Algorithms for Stopping Failures

Complete n-mode graph

**Basic alg.**

Procs keep sending all the values in V they've ever seen.

*use simple decision rule at the end.*

Maintain set \( W \subseteq V \), init only i's initial value

Repeatedly: Broadcast \( W \), add all received elements to \( W \)

After \( k \) rounds (for a certain fixed number \( k \)), use rule:
- if \( W \) is singleton, then decide on the unique value.
- else decide on a known default \( v \).
(Alternatively: Decide on min \( W \) - assumes \( V \) is t.o. set.)

Q: How many rounds to do this for?
A: Depends on \# of failures we want to tolerate.

Ex:
- 0 failures: \( k = 1 \) enough, all get same \( W \)
- 1 failure: \( k = 1 \) doesn't work: one process with different \( W \) from others could fail in middle of first round, send to some & not others, so they get \( \neq W_s \).

But \( k = 2 \) does work: if someone fails in 1st round, then no one does in 2nd, so OK.

More precisely (generally): If the algorithm is to tolerate \( f \) failures, go \( f + 1 \) rounds.

Theorem: Correctness of this alg.

pf:

Claim 1: For any \( n \geq 1 \leq n \leq f + 1 \), if no process fails during round \( n \), then \( W_i = W_j \) right after round \( n \), for all \( i, j \) that are active after \( n \) rounds.

pf: Each has union of all values held by active processes at beginning of round.

Claim 2: Suppose \( W_i \) are all \( = \) just after round \( n \), for all processes that are active after round \( n \), \( n \geq n' \).

Then \( W_i \) are all \( = \) just after round \( n' \), for all processes active after round \( n' \). (Once =, always =)
Check properties:

Agreement: Consider \( i,j \), active after \( f+1 \)

\[ \exists \text{ some } r, 1 \leq r \leq f+1, \text{ where no process failed (i.e., failures)} \]

Claim 1 says all agree just after \( r \).
Then Claim 2 implies all agree after \( f+1 \). (so pick same value)

Validity: Initially all \( v \rightarrow v \) only value anywhere.

(Strong algorithm (min) - quiescence)

Validity condition.

Termination: Obvious from decision rule.

\[ \text{Complexity:} \]

\[ f+1 \text{ rounds} \]
\[ \text{msgs: } O((f+1)^2) \]
\[ \text{msg bits: multiply by mb} \]
\[ \text{a fix mb on # of bits} \]
\[ \text{number of values per message} \]

Optimization

Can drastically reduce communication:

\[ O(m^2) \text{ msgs} \]
\[ O(m^2) \text{ bits} \]

by interesting observation that it's only necessary to know the first 2 values a process sees.

(On the end, process only needs to know exact elements of its WV set if \( |W| = 1 \); otherwise, just needs to know \( |W| \geq 2 \).)
(Works only for the weaker validity condition - using default value.)

**Alg:** Process firsts at rd 1, as before.
May first one more tune - at first round after it has heard of
some value + its own.
Then choose one such to hear.

**after f+1 rounds:** As before, if \( W \) is singleton, then decide on
unique value, else decide on known
default \( v_0 \).

**Correctness:** Can argue directly, as before.
On try to relate \( \tilde{O} \), the optimized alg., to the original unpkt.
algorith \( \bar{v} \).

This tries corresponds states of the \( \bar{z} \) algorithms with,
- same inputs (as before)
- same failure pattern.

**Sim relation definition:**
Write \( W \cup U \) for the \( W \) sets in the \( V \) alg.
\( W \cup \) \( V \cup \) \( U \cup \) \( \bar{v} \) alg.

**Sim reln:**
\[
\begin{align*}
\overline{W}_0 & \leq W U_c \\
\text{if} \ |W U_c| = 1 \ & \text{then} \ |\overline{W}_0| = |W U_c| \\
\text{if} \ |W U_c| \geq 2 \ & \text{then} \ |\overline{W}_0| \geq 2
\end{align*}
\]

This can be proved by induction on \( \# \) of rounds.

**Key argument:** Consider \( i \) receiving some maps at rd \( r \), in both
algorithms.
Messages could be different in the 2 alg., but each 0 msg's set = consup. V msg's set

Cases:
If i receives, in V, only from proc that have \( l_{W_0} = 1 \), then the 0 alg. sends all the same info.

Otherwise, if i receives, in V, from some proc. with \( l_{W_0} \geq 2 \), then i needs up with \( l_{W_0} \geq 2 \).

Details in book.

Sim. relation after \( f+1 \) rounds implies that the same decisions are made in both algorithms (using default).
Exponential Info Gathering

I'll introduce one more algorithm for stopping agreement. Very common, inefficient one.

Why? Introduces a data structure that extends to Byz. case (the ones used so far don't).
(That problem seems to be much harder.)

EIS tree: $T = T^n_m, 1$

Paths from root to leaf correspond to chains of processes, all ≠.

$f+2$ levels (nodes) numbered $0, \ldots, f+1$.

Level $k$ node has $m - k$ children.

Label each node with string of proc. ids.

- Root: $(\lambda)$
- Node with $(i_1, \ldots, i_k)$ label has children labelled with $(i_{j1}, \ldots, i_{jk})$, $j \in [1, \ldots, m]$, $i_{kj} \neq i_j$

**Ex**: For 4 proc. $T^4_2$

$\lambda$ is root

$f = 2$

Each process maintains same tree structure.

During alg., decorates tree with values $e \in V$, level $k$ at end of $rd$ $k$.

Don't: root decorated with with own input value.
Let $n \geq 1$.
Send all level $n-1$ decoration values from nodes with labels not containing your own index to everyone.

Recipients use it to decorate level $n$ (sender's id appended at end).

Thus, the node with label $i_1, \ldots, i_k$ gets decorated with $v$ provided that $i_k$ told you that $i_{k-1}$ told him that ... that $i_1$ told him that $i_1$'s initial value was $v$.

If common chain broken by a failure, label with null.

Decision rule: Easy (for stopping case - will be harder for Byz.)

Let $W$ be set of all values in the tree.
If $|W| = 1$, rule that value, else default.

(similar to the first alg.)

Ex: $\exists x T_3 \rightarrow 1$ failure

\begin{align*}
&\begin{array}{c}
1 \\
2 \\
3
\end{array} \\
&\begin{array}{c}
12 \\
13 \\
21 \\
23 \\
31 \\
32
\end{array}
\end{align*}

\begin{align*}
&\begin{array}{c}
1 \\
0
\end{array} \\
&\begin{array}{c}
10 \\
11
\end{array} \\
&\begin{array}{c}
11 \\
01
\end{array}
\end{align*}

Start with 1, 0, 1.

$p_2$ faulty.

Fail after sending to $p_1$ at rd 1, not $p_3$.

$p_2$ fails in nothing.

Note $p_3$ does not discover that $p_2$'s value is 0 until after rd 2. (fails from $p_1$.)
Correctness: Similar to the simpler alg.

Complexity:

- time: ok, $f+1$ rounds, as before
- msg: bad, $O((f+1)n^2)$, but blow up exponentially in size.
- bits: $O(n^{f+1}b)$

exp. a number of failures

For the stopping case, can optimize by pruning to the first 2 msg, again.

Remark: Authentication. This EIG alg. + its optimized version can be used to tolerate worse types of failures than just stopping - not full Byz., but a restricted version that adds power of authentication.

(removes ability of process to make false claims about what others have said)

But for this to make sense, must first consider Byz. failures.
One more kind of result before considering Byzantine case:
- lower bound on number of rounds.
  Note $f + 1$ rounds used in all the algo we've seen so far.
  Turns out that is inherent.

$f + 1$ is needed, even just for stopping failures.
Assume $f$-rd algorithm is got contradiction.
Restrictions on alg, WLOG:
- $n$-node complete graph.
- decisions at end of $rd f$.
- $V = \{0, 1\}$
- all send msgs to each $rd \leq f$

To give the intuition, I'll start by showing for $f = 1 + f = 2$

\[ f = 1: \text{3lin}: \text{Suppose } n \geq 3. \]
- for any process stopping agreement alg. for 1 fault, in which nonfaulty proc always decide at end of rd 1.

If: Suppose A exists.
Construct chain of executions, s.t.
- first has 0 as (unique) decision value
- last has 1 as (unique) decision value
any 2 conseq executions are indistinguishable to some process i
  who is nonfaulty in both
This means i must decide the same in both, so the two have the same (unique) decision values.

Contrad.
\[ x_0: \text{All processes input 0, no failures} \]

\[ x_k: \text{Same, for } k. \]

\[ \text{Last} \]

To start, start from \( x_0 \).

Next set \( x_1 \), remove msg from \( 1 \rightarrow 2 \).

2 errors \( x_0, x_1 \), look same to all but 1 and 2.

(Since \( n \geq 3 \), there is some other process.)

Then, faces may fail in both cases.

Next remove msg \( 1 \rightarrow 3 \).

Then replace missing messages, 1 at a time. Etc. further limit values

\( f = 2 \): will illustrate how to extend this to more failures.

Elim: Suppose \( n \geq 4 \).

Then there is no \( n \) process stopping, agreement algorithm, for 2 faults,

in which all non-faulty processes always decide at end of \( n \) set 2.

If: Another chain, but this time longer, and more delicate to construct.

\[ x_0, x_1, \ldots, x_k \]

\[ \text{All 0's, no failures} \]

\[ \text{(2 sets)} \]

Each consecutive pair starts 2 faults between them, indistinguishable to some non-faulty process.
In between have intermediate excess \( a_1, \ldots, a_m \) (big steps) \[ a_i \] 

Suppose \( i, \ldots, i \) have 1

\( i+1, \ldots, n \) have 0

(1 no failures)

All these "big steps" are essentially the same—change one guy's initial value.

I'll just show how to connect \( a_0 \) to \( x_i \) involves changing \( p_i \)'s initial value from 0 to 1.

Start with \( a_0 \), work toward killing \( p_i \) at beginning to change initial value.

Start by removing not 2 maps one by one.

Then have:

\[ p_i \]

Can't just remove not 1 maps, because would no longer be true that the two excess look the same to some map process.

E.g., removing \( 1 \to i \) leaves \( i \) a chance to tell all other processes about the failure, at \( a_0 \).

So must be smarter...

Can take advantage of \( f = 2 \), allowing 2 processes to fail in same excess.

So, we may steps to remove not 1 map \( 1 \to i \).

In these steps, both \( 1 + i \) are faulty.

Start where \( a_0 \) sees \( i \) at \( a_0 \). \( i \) nonfaulty, (seem all not 2 maps).

Remove not 2 maps sent by \( i \); one by one, till we reach execution where \( 1 \text{ and } i \); but \( i \) sends no msg at \( a_0 \).
Then remove msg \( i \rightarrow 1 \).

Now, before \(+\) after this last step look the same to all by \( 1 + i \). (nonfaulty)

Then replace \( \text{rd 2} \) msgs \( 1 \) by \( 1 \), till \( i \) no longer faulty.

(All this has just removed one \( \text{rd 1} \) msg from \( 1 \).)

Now repeat for all \( \text{rd 1} \) msgs from \( i \), till all removed.

Then can change \( i \)'s initial value, as before.

---

**General Theorem**

\[
\text{Theorem: Suppose } m \geq f + 2. \\
\text{Then there is no } m \text{-process stopping alg., alg. for } f \text{ faults, } + \text{ in which all nonfaulty proc.'s decide at end of rd } f.
\]

**Proof:** Similar ideas, longer chain.

Must kill \( f \) processes along the way.

Recursive construction, technical, I'll sketch - you read carefully.

(Some new papers may provide simpler proofs, but really the same ideas.)

---

**Regular execution:** For all \( k \), \( 0 \leq k \leq f \), at most \( k \) proc's fail by end of rd \( k \).

All the execs appearing in the chains are regular.

Interesting general lemma says everything is related!

**Lemma 1:** If \( \alpha \), \( \alpha' \) are regular executions, then \( \alpha \cong \alpha' \).

Here, \( \cong \) represents the existence of a chain of regular execs, specifically:

\( \alpha \cong \alpha' \) looks the same to \( i \)

\( \alpha \cong \alpha' \) looks the same to some \( i \) that is nonfaulty in both

\( \alpha \cong \alpha' \) trans. closure of \( \alpha \)
To prove Lemma 1, break it down according to input assignments:

Lemma 2: If \( x, x' \) are reg. execs with same input assignment, then \( x \cong x' \).

If we have Lemma 2, we get Lemma 1 as follows:
Suppose \( x, x' \) differ in 1 input value, say \( p \).
Let \( \beta, \beta' \) be same input as \( x \) (resp. \( x' \)), \( p \) fails at \( \beta \) +
all other nonfaulty.
Then \( x \cong \beta + x' \cong \beta' \) by Lemma 2.
And \( \beta \cong \beta' \) because just change input (\( p \) fails at \( \beta' \)).
So \( x \cong x' \).

Extend this for more process values.

So, prove Lemma 2:

Recursive decomposition, expressed by using:

Claim: Let \( 0 \leq k \leq f \).
If \( x, x' \) are regular execs, same inputs, identical failure patterns
then \( k \) rounds, then \( x \cong x' \).

Claim \( \Rightarrow \) Lemma 2 easily, using \( k = 0 \).

\( \blacksquare \)

Proof: Reverse induction on \( k \), from

Base: \( k = f \)
Easy - \( x \) and \( x' \) are identical if same thru \( f \).

Inductive: \( 0 < k < f-1 \), assume true for \( k+1 \), show for \( k \)

Define \( \text{FF}(x, n) = \text{Exec same as } x \text{ thru first } n \text{ ids, then has no new failures afterwards.} \)

We'll show \( x \cong \text{FF}(x, k) \)

This is enough: Given arbitrary \( x, x' \) can use this twice

\( \text{FF}(x, k) = \text{FF}(x', k) \)
So, fix regular excess \( x \).

By induction, \( x \approx FF(x, k+1) \)

So, it suffices to show \( FF(x, k) \approx FF(x, k+1) \).

The interesting case is where some process fails in \( x \) in \( rd \ k+1 \).
So, assume some do; let \( I \) = set of such processes

Let \( x_0 \) be same as \( FF(x, k) \) except all processes in \( I \) fail
(completely), right at the end of round \( k+1 \).

\( x_0 \) and \( FF(x, k) \) identical through \( k+1 \) rounds, so by induction,

\( x_0 \approx FF(x, k) \)

So, it suffices to show

\( x_0 \approx FF(x, k+1) \).

So, it all boils down to constructing a chain of regular excess from \( x_0 \) to \( FF(x, k+1) \).

The only difference between \( x_0 \) and \( FF(x, k+1) \) is that some morgs sent by processes in \( I \) at \( rd \ k+1 \) in \( x_0 \) are missing in \( FF(x, k+1) \).

So, we remove these morgs 1 at a time; this is as in the previous proofs.

E.g., consider removal of a rd \( k+1 \) morg from \( i \) to \( j \), when \( i \in I \), let \( \beta \) be the excess including the morg.

Must argue \( \beta \approx \delta \)
If \( k + 1 = \sigma \), this is the final round.
Then \( \beta \approx \gamma \) indist. to all except \( i + j \), so \( \beta \approx \gamma \).

On the other hand, if \( k + 1 \leq \sigma - 1 \):
Then it's not the final round, use inductive hypothesis:
Define \( \beta' = \) same as \( \beta \) but with \( j \) failing right after \( i \) or \( i + k \) (if not previously failed).

\( \gamma' = \) same as \( \gamma \) but \( \cdots \) \( \cdots \) \( \gamma' \)
\( \beta \approx \beta' \) since same till \( k + 1 \) rounds (uses ind. hyp.)
\( \gamma \approx \gamma' \)
But \( \beta' \approx \gamma' \) because indist. to all processes exc. \( i + j \).
So \( \beta \approx \gamma \)

This gives the chain.
So \( x \approx \epsilon \), as needed.
A new proof of this lower box may be easier to see.
I'll just sketch the ideas.

Ref: Reidel-rajeban, preliminary note. 2001

Extend def of regular area to any length $\Sigma$.
Define slightly different $\sim$ relation which applies to any length $\Sigma$.

$\sim \sim \sim$ for $a \neq a'$, regular areas of same length
$\sim \sim \sim$ for all but one.

Then let $\sim$ be reflexive trans closure, as before.

Use essentially the same construction we already did to show that
all regular areas of the same length are $\sim$ related,
using this stronger relationship $\sim$. For any length $\Sigma$.

Nice thing is that this statement can be proved by (ordinary)
induction on length $\Sigma$.

Case: $k = 0$
Remains only.
Must show $a \sim a$.

Inductive Step: Assume for $k < j-1$, show for $k$.
Suffices to show:

Suppose \( x, x' \) are \( k+1 \)-round regular executions.

(a) If \( x, x' \) extend a common \( k \)-rd exec \( \beta \), then \( x \sim x' \).

(b) If \( x + x' \) extend \( \beta + \beta' \) respectively, then \( x \not\sim x' \).

Each of \( x, x' \) may involve some failures.

Relate each to \( x'' \), the extension of \( \beta \) with no new failures.

Do this by constructing chain, at each step doing only one of:

- remove one msg
- don't let a newly-failed proc perform its state transition
- remove one msg

At each step, look save to all except proc that doesn't perform the transition a proc that doesn't receive the msg.

Know \( \beta \sim \beta' \) to all but one, say \( j \).

Construct \( \gamma \) (resp. \( \gamma' \)) by failing \( j \) right after not \( k \) + then having no more failures

These are still regular.

Moreover, \( \gamma \sim \gamma' \) since still save to all but \( j \).

Also know \( \gamma \sim x, \gamma' \sim x' \), by part (a).

So by transitivity, \( x \sim x' \).
Now the connectivity yields the f+1-round lower bound similarly to before.
Assume \( m \geq f+2 \). (needed?)

Assume first, after f steps.

Define \( \text{val}(x) = \) the unique decision value in \( x \), for every execution \( x \) with \( \leq f \) failures.

Then \( \exists \) regular actions \( \left\{ \text{val} = 0 \right\} \left\{ \text{val} = 1 \right\} \)

So by connectivity, \( \exists x \) and \( x' \), the regular actions of length \( f \)
with \( \text{val}(x) + \text{val}(x') \neq \text{val}(x) \).

But these look the same to all but 1 process.
Since total number of failures in both is \( \leq f+1 \)

\( + m \geq f+2 \) \hspace{1cm} \text{(can get f?)} \hspace{1cm} \text{if steps are small enough}

\( \exists \) some nonfaulty process to which both look the same.

Contrad - must decide same in both \( x \) and \( x' \).

Can also use this to get a kind of lower \( f+1 \) - limitation on early decision:

For any \( k \leq f-1 \), (Assume \( m \geq f+1 \))

\( \exists k\)-rd execution with \( \leq f \) failures, s.t. most every nonfaulty process decides \( x \), in the same \( f \) extension in which no one else fails, not everyone \( A \) nonf.

Process that doesn't decide by the end of our round.

Proof: By connectivity, get \( x \), a \( k \)-regular action.
Can also use this to get a kind of lower bound: limitation on "early decision".

For any $k \leq f-2$:

\[ \exists \text{ 3rd round (with } \leq f \text{ failures) } \text{ s.t. in extension in which no one else fails, } \exists \text{ nonfaulty process that doesn't decide by the end of one more rd.} \]

Proof sketch:

For $k$-rd rd $x$, define $\text{val}(x)$ to be the unique decision value any with $\leq f$ failures

in nonfaulty extension.

Connectivity yields $x, x'$ of length $k$, with $x \neq x'$ look same to all but one, say $j$.

Assume, for sake of contrast, that in next round both $x, x'$ nonf. act, must decide.

Consider extension $\beta$ of $x$ in which $j$ fails $x$ sends to one monf.

process $p_j$ only, in next round $(k+1)$.

Likewise, $x'$ of $x'$ ... was $p_j$.

Then $p_j$ must decide after rd $k+1$, since it looks to $p_k$

like a nonfaulty round.

Decides $\text{val}(x)$ in $\beta$, $\text{val}(x')$ in $\beta'$, different.

But $\beta \equiv \beta'$ for all $m \in \{j, k\}$.

They know something is wrong, so aren't required to decide after rd $k+1$. 
But now if we fail I in \( \beta + \beta' \), the others can’t tell the difference. So, have to decide eventually on something + will disagree in one of the cases.

Control.