6.852: Distributed Algorithms
Fall, 2009

Class 5
Today’s plan

- Review EIG algorithm for Byzantine agreement.
- Number-of-processors lower bound for Byzantine agreement.
- Connectivity bounds.
- Weak Byzantine agreement.
- Time lower bounds for stopping agreement and Byzantine agreement.
- Reading: Sections 6.3-6.7, [Aguilera, Toueg], [Keidar-Rajsbaum]
- Next:
  - Other distributed agreement problems
  - Reading: Chapter 7 (but skim 7.2)
Byzantine agreement

• Recall correctness conditions:
  – Agreement: No two nonfaulty processes decide on different values.
  – Validity: If all nonfaulty processes start with the same v, then v is the only allowable decision for nonfaulty processes.
  – Termination: All nonfaulty processes eventually decide.

• Presented EIG algorithm for Byzantine agreement, using:
  – Exponential communication (in f)
  – f+1 rounds
  – n > 3f
EIG algorithm for Byzantine agreement

• Use EIG tree.
• Relay messages for f+1 rounds.
• Decorate the EIG tree with values from V, replacing any garbage messages with default value $v_0$.
• Call the decorations $\text{val}(x)$, where $x$ is any node label.
• Decision rule:
  – Redecorate the tree bottom-up, defining $\text{newval}(x)$.
    • Leaf: $\text{newval}(x) = \text{val}(x)$
    • Non-leaf: $\text{newval}(x) =$
      – $\text{newval}$ of strict majority of children in the tree, if majority exists,
      – $v_0$ otherwise.
  – Final decision: $\text{newval}(\lambda)$ (newval at root)
Example: \( n = 4, f = 1 \)

- \( T_{4,1} \):
- Consider a possible execution in which \( p_3 \) is faulty.
- Initial values 1 1 0 0
- Round 1
- Round 2
Example: \( n = 4, f = 1 \)

- Now calculate newvals, bottom-up, choosing majority values, \( v_0 = 0 \) if no majority.

Corrected by taking majority
Correctness proof

• **Lemma 1:** If \( x \) ends with a nonfaulty process index then \( \text{val}(x)_i = \text{val}(x)_j \) for every nonfaulty \( i \) and \( j \).

• In example, **such nodes are:**

```
12 λ 43
14 41 32 34
13 23 21 24 42
```

• **Lemma 2:** If \( x \) ends with a nonfaulty process index then \( \exists v \) such that \( \text{val}(x)_i = \text{newval}(x)_i = v \) for every nonfaulty \( i \).

• **Proof:** Induction on level in the tree, bottom up.
Main correctness conditions

• Validity:
  – Uses Lemma 2.

• Termination:
  – Obvious.

• Agreement:
Agreement

• **Path covering:** Subset of nodes containing at least one node on each path from root to leaf:

\[ \lambda \]

- 12 13 14 21 23 24 31 32 34 41 42 43

• **Common node:** One for which all nonfaulty processes have the same newval.
  - All nodes whose labels end in nonfaulty process index are common.
Agreement

- **Lemma 3:** There exists a path covering all of whose nodes are common.
- **Proof:**
  - Let $C = \text{nodes with labels of the form } x_j, j \text{ nonfaulty}$.

- **Lemma 4:** If there’s a common path covering of the subtree rooted at any node $x$, then $x$ is common.

- **Lemma 5:** The root is common.
- **Yields Agreement.**
Complexity bounds

• As for EIG for stopping agreement:
  – Time: $f+1$
  – Communication: $O(n^{f+1})$

• But now, also requires $n > 3f$ processors.

• Q: Is $n > 3f$ necessary?
Lower bound on the number of processes for Byzantine Agreement
Number of processors for Byzantine agreement

• \( n > 3f \) is necessary!
  – Holds for any \( n \)-node (undirected) graph.
  – For graphs with low connectivity, may need even more processors.
  – Number of failures that can be tolerated for Byzantine agreement in an undirected graph \( G \) has been completely characterized, in terms of number of nodes and connectivity.

• **Theorem 1**: 3 processes cannot solve Byzantine Agreement with 1 possible failure.
Proof (3 vs. 1 BA)

- By contradiction. Suppose algorithm A, consisting of processes 1, 2, 3, solves BA with 1 possible failure.
- Construct new system S from 2 copies of A, with initial values as follows:
  - What is S?
    - A synchronous system of some kind.
    - Not required to satisfy any particular correctness conditions.
    - Not necessarily a correct BA algorithm for the 6-node ring.
    - Just some synchronous system, which runs and does something.
    - We’ll use it to get our contradiction.
Proof (3 vs 1 BA)

- Consider 2 and 3 in S:
- Looks to them like:
  - They’re in A, with a faulty process 1.
  - 1 emulates 1′-2′-3′-1 from S.
- In A, 2 and 3 must decide 0
- So by indistinguishability, they decide 0 in S also.
Proof (3 vs 1 BA)

• Now consider 1’ and 2’ in S.
• Looks to them like:
  – They’re in A with a faulty process 3.
  – 3 emulates 3’-1-2-3 from S.
• They must decide 1 in A, so they decide 1 in S also.
Proof (3 vs 1 BA)

- Finally, consider 3 and 1’ in S:
  - Looks to them like:
    - They’re in A, with a faulty process 2.
    - 2 emulates 2’-3’-1-2 from S.
  - In A, 3 and 1 must agree.
  - So by indistinguishability, 3 and 1’ agree in S also.

- But we already know that process 1’ decides 1 and process 3 decides 0, in S.
- Contradiction!
Discussion

• We get this contradiction even if the original algorithm A is assumed to “know n”.

• That simply means that:
  – The processes in A have the number 3 hard-wired into their state.
  – Their correctness properties are required to hold only when they are actually configured into a triangle.

• We are allowed to use these processes in a different configuration S---as long as we don’t claim any particular correctness properties for S.
Impossibility for $n = 3f$

- **Theorem 2:** $n$ processes can’t solve BA, if $n \leq 3f$.
- **Proof:**
  - Similar construction, with $f$ processes treated as a group.
  - Or, can use a reduction:
    - Show how to transform a solution for $n \leq 3f$ to a solution for $3$ vs. $1$.
    - Since $3$ vs. $1$ is impossible, we get a contradiction.

- Consider $n = 2$ as a special case:
  - $n = 2$, $f = 1$
  - Each could be faulty, requiring the other to decide on its own value.
  - Or both nonfaulty, which requires agreement, contradiction.

- So from now on, assume $3 \leq n \leq 3f$.
- Assume a Byzantine Agreement algorithm $A$ for $(n,f)$.
- Transform it into a BA algorithm $B$ for $(3,1)$. 
Transforming A to B

• Algorithm:
  – Partition A-processes into groups $I_1$, $I_2$, $I_3$, where $1 \leq |I_1|, |I_2|, |I_3| \leq f$.
  – Each $B_i$ process simulates the entire $I_i$ group.
    – $B_i$ initializes all processes in $I_i$ with $B_i$’s initial value.
    – At each round, $B_i$ simulates sending messages:
      • Local: Just simulate locally.
      • Remote: Package and send.
    – If any simulated process decides, $B_i$ decides the same (use any).

• Show $B$ satisfies correctness conditions:
  – Consider any execution of $B$ with at most 1 fault.
  – Simulates an execution of $A$ with at most $f$ faults.
  – Correctness conditions must hold in the simulated execution of $A$.
  – Show these all carry over to $B$’s execution.
B’s correctness

• **Termination:**
  - If $B_i$ is nonfaulty in $B$, then it simulates only nonfaulty processes of $A$ (at least one).
  - Those terminate, so $B_i$ does also.

• **Agreement:**
  - If $B_i$, $B_j$ are nonfaulty processes of $B$, they simulate only nonfaulty processes of $A$.
  - Agreement in $A$ implies all these agree.
  - So $B_i$, $B_j$ agree.

• **Validity:**
  - If all nonfaulty processes of $B$ start with $v$, then so do all nonfaulty processes of $A$.
  - Then validity of $A$ implies that all nonfaulty $A$ processes decide $v$, so the same holds for $B$. 
General graphs and connectivity bounds

• $n > 3f$ isn’t the whole story:
  – 4 processes, can’t tolerate 1 fault:

• **Theorem 3:** BA is solvable in an $n$-node graph $G$, tolerating $f$ faults, if and only if both of the following hold:
  – $n > 3f$, and
  – $\text{conn}(G) > 2f$.

• $\text{conn}(g)$ = minimum number of nodes whose removal results in either a disconnected graph or a 1-node graph.

• **Examples:**

  - $\text{conn} = 1$
  - $\text{conn} = 3$
  - $\text{conn} = 3$
Proof: “If” direction

• **Theorem 3:** BA is solvable in an n-node graph G, tolerating f faults, if and only if n > 3f and conn(G) > 2f.
• **Proof ("if"):**
  – Suppose both hold.
  – Then we can simulate a total-connectivity algorithm.
  – Key is to emulate reliable communication from any node i to any other node j.
  – Rely on **Menger’s Theorem**, which says that a graph is c-connected (that is, has conn ≥ c) if and only if each pair of nodes is connected by ≥ c node-disjoint paths.
  – Since conn(G) ≥ 2f + 1, we have ≥ 2f + 1 node-disjoint paths between i and j.
  – To send message, send on all these paths (assumes graph is known).
  – Majority must be correct, so take majority message.
Proof: “Only if” direction

• **Theorem 3**: BA is solvable in an n-node graph G, tolerating f faults, if and only if $n > 3f$ and $\text{conn}(G) > 2f$.

• **Proof (“only if”)**:
  – We already showed $n > 3f$; remains to show $\text{conn}(G) > 2f$.
  – Show key idea with simple case, $\text{conn} = 2$, $f = 1$.
  – Canonical example:
    • Disconnect 1 and 3 by removing 2 and 4:
  – Proof by contradiction.
  – Assume some algorithm A that solves BA in this canonical graph, tolerating 1 failure.
Proof (conn = 2, 1 failure)

• Now construct $S$ from two copies of $A$.

• Consider 1, 2, and 3 in $S$:
  – Looks to them like they’re in $A$, with a faulty process 4.
  – In $A$, 1, 2, and 3 must decide 0.
  – So they decide 0 in $S$ also.

• Similarly, 1’, 2’, and 3’ decide 1 in $S$. 
Proof (conn = 2, 1 failure)

- Finally, consider 3’, 4’, and 1 in S:
  - Looks to them like they’re in A, with a faulty process 2.
  - In A, they must agree, so they also agree in S.
  - But 3’ decides 0 and 1 decides 1 in S, contradiction.

- Therefore, we can’t solve BA in canonical graph, with 1 failure.

- As before, can generalize to conn(G) ≤ 2f, or use a reduction.
Byzantine processor bounds

• The bounds $n > 3f$ and $\text{conn} > 2f$ are fundamental for consensus-style problems with Byzantine failures.

• Same bounds hold, in synchronous settings with $f$ Byzantine faulty processes, for:
  – Byzantine Firing Squad synchronization problem
  – Weak Byzantine Agreement
  – Approximate agreement

• Also, in timed (partially synchronous settings), for maintaining clock synchronization.

• Proofs used similar methods.
Weak Byzantine Agreement

[Lamport]

• Correctness conditions for BA:
  – Agreement: No two nonfaulty processes decide on different values.
  – Validity: If all nonfaulty processes start with the same $v$, then $v$ is the only allowable decision for nonfaulty processes.
  – Termination: All nonfaulty processes eventually decide.

• Correctness conditions for Weak BA:
  – Agreement: Same as for BA.
  – Validity: If all processes are nonfaulty and start with the same $v$, then $v$ is the only allowed decision value.
  – Termination: Same as for BA.

• Limits the situations where the decision is forced to go a certain way.

• Similar style to validity condition for 2-Generals problem.
WBA Processor Bounds

• **Theorem 4**: Weak BA is solvable in an $n$-node graph $G$, tolerating $f$ faults, if and only if $n > 3f$ and $\text{conn}(G) > 2f$.

• Same bounds as for BA.

• **Proof**:
  – “If”: Follows from results for ordinary BA.
  – “Only if”:
    • By constructions like those for ordinary BA, but slightly more complicated.
    • Show 3 vs. 1 here, rest LTTR.
Proof (3 vs. 1 Weak BA)

• By contradiction. Suppose algorithm A, consisting of procs 1, 2, 3, solves WBA with 1 fault.
• Let $\alpha_0 =$ execution in which everyone starts with 0 and there are no failures; results in decision 0.
• Let $\alpha_1 =$ execution in which everyone starts with 1 and there are no failures; results in decision 1.
• Let $b =$ upper bound on number of rounds for all processes to decide, in both $\alpha_0$ and $\alpha_1$.
• Construct new system S from $2b$ copies of A:
Proof (3 vs. 1 Weak BA)

• Claim: Any two adjacent processes in S must decide the same thing.
  – Because it looks to them like they are in A, and they must agree in A.
• So everyone decides the same in S.
• WLOG, all decide 1.
Proof (3 vs. 1 Weak BA)

- Now consider a block of $2b + 1$ consecutive processes that begin with 0:

```
1 2 3 1 2 3 1 2 3
0 0 0 0 0 0 0 0 0
```

- **Claims:**
  - To all but the endpoints, the execution of $S$ is indistinguishable from $\alpha_0$, the failure-free execution in which everyone starts with 0, for 1 round.
  - To all but two at each end, indistinguishable from $\alpha_0$ for 2 rounds.
  - To all but three at each end, indistinguishable from $\alpha_0$ for 3 rounds.
  - ...
  - To midpoint, indistinguishable for $b$ rounds.

- But $b$ rounds are enough for the midpoint to decide 0, contradicting the fact that everyone decides 1 in $S$. 
Lower bound on the number of rounds for Byzantine agreement
Lower bound on number of rounds

- Notice that \( f+1 \) rounds are used in all the agreement algorithms we’ve seen so far---both stopping and Byzantine.
- That’s inherent: \( f+1 \) rounds are needed in the worst-case, even for simple stopping failures.
- Assume an \( f \)-round algorithm \( A \) tolerating \( f \) faults, and get a contradiction.
- Restrictions on \( A \) (WLOG):
  - \( n \)-node complete graph.
  - Decisions at end of round \( f \).
  - \( V = \{0,1\} \)
  - All-to-all communication at every round \( \leq f \).
Special case: \( f = 1 \)

- **Theorem 5:** Suppose \( n \geq 3 \). There is no \( n \)-process 1-fault stopping agreement algorithm in which nonfaulty processes always decide at the end of round 1.
  - **Proof:** Suppose A exists.
    - Construct a chain of executions, each with at most one failure, such that:
      - First has (unique) decision value 0.
      - Last has decision value 1.
      - Any two consecutive executions in the chain are indistinguishable to some process i that is nonfaulty in both. So i must decide the same in both executions, and the two must have the same decision values.
      - Decision values in first and last executions must be the same.
      - Contradiction.
Round lower bound, $f = 1$

- $\alpha_0$: All processes have input 0, no failures.
- ... 
- $\alpha_k$ (last one): All inputs 1, no failures.
- Start the chain from $\alpha_0$.
- Next execution, $\alpha_1$, removes message 1 → 2.
  - $\alpha_0$ and $\alpha_1$ indistinguishable to everyone except 1 and 2; since $n \geq 3$, there is some other process.
  - These processes are nonfaulty in both executions.
- Next execution, $\alpha_2$, removes message 1 → 3.
  - $\alpha_1$ and $\alpha_2$ indistinguishable to everyone except 1 and 3, hence to some nonfaulty process.
- Next, remove message 1 → 4.
  - Indistinguishable to some nonfaulty process.
Continuing…

- Having removed all of process 1’s messages, change 1’s input from 0 to 1.
  - Looks the same to everyone else.
- We can’t just keep removing messages, since we are allowed at most one failure in each execution.
- So, we continue by replacing missing messages, one at a time.
- Repeat with process 2, 3, and 4, eventually reach the last execution: all inputs 1, no failures.
Special case: \( f = 2 \)

- **Theorem 6:** Suppose \( n \geq 4 \). There is no \( n \)-process 2-fault stopping agreement algorithm in which nonfaulty processes always decide at the end of round 2.

- **Proof:** Suppose \( A \) exists.
  - Construct another chain of executions, each with at most 2 failures.
    - This time a bit longer and more complicated.
  - Start with \( \alpha_0 \): All processes have input 0, no failures, 2 rounds:
  - Work toward \( \alpha_n \), all 1’s, no failures.
  - Each consecutive pair is indistinguishable to some nonfaulty process.
  - Use intermediate execs \( \alpha_i \), in which:
    - Processes 1,\ldots,i have initial value 1.
    - Processes i+1,\ldots,n have initial value 0.
    - No failures.
Special case: \( f = 2 \)

- Show how to connect \( \alpha_0 \) and \( \alpha_1 \).
  - That is, change process 1’s initial value from 0 to 1.
  - Other intermediate steps essentially the same.
- Start with \( \alpha_0 \), work toward killing p1 at the beginning, to change its initial value, by removing messages.
- Then replace the messages, working back up to \( \alpha_1 \).
- Start by removing p1’s round 2 messages, one by one.
- Q: Continue by removing p1’s round 1 messages?
  - No, because consecutive executions would not look the same to anyone:
    - E.g., removing 1 \( \rightarrow \) 2 at round 1 allows p2 to tell everyone about the failure.
Special case: \( f = 2 \)

- Removing \( 1 \rightarrow 2 \) at round 1 allows \( p2 \) to tell all other processes about the failure:

\[
\begin{array}{cc}
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
\end{array}
\]

vs.

\[
\begin{array}{cc}
0 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 0 \\
\end{array}
\]

- Distinguishable to everyone.
- So we must do something more elaborate.
- Recall that we can allow 2 processes to fail in some executions.
- Use many steps to remove a single round 1 message \( 1 \rightarrow i \); in these steps, both 1 and \( i \) will be faulty.
Removing p1’s round 1 messages

- Start with execution where p1 sends to everyone at round 1, and only p1 is faulty.
- Remove round 1 message 1 → 2:
  - p2 starts out nonfaulty, so sends all its round 2 messages.
  - Now make p2 faulty.
  - Remove p2’s round 2 messages, one by one, until we reach an execution where 1 → 2 at round 1, but p2 sends no round 2 messages.
  - Now remove the round 1 message 1 → 2.
    - Executions look the same to all but 1 and 2 (and they’re nonfaulty).
    - Replace all the round 2 messages from p2, one by one, until p2 is no longer faulty.
- Repeat to remove p1’s round 1 messages to p3, p4,…
- After removing all of p1’s round 1 messages, change p1’s initial value from 0 to 1, as needed.
General case: Any $f$

- **Theorem 7**: Suppose $n \geq f + 2$. There is no $n$-process $f$-fault stopping agreement algorithm in which nonfaulty processes always decide at the end of round $f$.

- **Proof**: Suppose $A$ exists.
  - Same ideas, longer chain.
  - Must fail $f$ processes in some executions in the chain, in order to remove all the required messages, at all rounds.
  - Construction in book, LTTR.

- **Newer proof [Aguilera, Toueg]**:
  - Uses ideas from [FLP] impossibility of consensus.
  - They assume strong validity, but the proof works for our weaker validity condition also.
Proof:
- By contradiction. Assume A solves stopping agreement for f failures and everyone decides after exactly f rounds.
- Restrict attention to executions in which at most one process fails during each round.
- Recall failure at a round allows process to miss sending an arbitrary subset of the messages, or to send all but halt before changing state.
- Consider vector of initial values as a 0-round execution.
-Defs (adapted from [Fischer, Lynch, Paterson]): \( \alpha \), an execution that completes some finite number (possibly 0) of rounds, is:
  - 0-valent, if 0 is the only decision that can occur in any execution (of the kind we consider) that extends \( \alpha \).
  - 1-valent, if 1 is...
  - Univalent, if \( \alpha \) is either 0-valent or 1-valent (essentially decided).
  - Bivalent, if both decisions occur in some extensions (undecided).
Initial bivalence

• **Lemma 1**: There is some 0-round execution (vector of initial values) that is bivalent.
• **Proof (adapted from [FLP])**:  
  – Assume for contradiction that all 0-round executions are univalent.
  – 000…0 is 0-valent
  – 111…1 is 1-valent
  – So there must be two 0-round executions that differ in the value of just one process, say i, such that one is 0-valent and the other is 1-valent.
  – But this is impossible, because if process i fails at the start, no one else can distinguish the two 0-round executions.
**Bivalence through f-1 rounds**

- **Lemma 2:** For every $k$, $0 \leq k \leq f$-1, there is a bivalent $k$-round execution.

- **Proof:** By induction on $k$.
  - **Base (k=0):** Lemma 1.
  - **Inductive step:** Assume for $k$, show for $k+1$, where $k < f$ -1.
    - Assume bivalent $k$-round execution $\alpha$.
    - Assume for contradiction that every 1-round extension of $\alpha$ (with at most one new failure) is univalent.
    - Let $\alpha^*$ be the 1-round extension of $\alpha$ in which no new failures occur in round $k+1$.
    - By assumption, this is univalent, WLOG 1-valent.
    - Since $\alpha$ is bivalent, there must be another 1-round extension of $\alpha$, $\alpha^0$, that is 0-valent.
Bivalence through f-1 rounds

• In $\alpha^0$, some single process $i$ fails in round $k+1$, by not sending to some subset of the processes, say $J = \{j_1, j_2, \ldots j_m\}$.
• Define a chain of $(k+1)$-round executions, $\alpha^0, \alpha^1, \alpha^2, \ldots, \alpha^m$.
• Each $\alpha^l$ in this sequence is the same as $\alpha^0$ except that $i$ also sends messages to $j_1, j_2, \ldots j_l$.
  – Adding in messages from $i$, one at a time.
• Each $\alpha^l$ is univalent, by assumption.
• Since $\alpha^0$ is 0-valent, there are 2 possibilities:
  – At least one of these is 1-valent, or
  – All of these are 0-valent.
Case 1: At least one $\alpha^l$ is 1-valent

- Then there must be some $l$ such that $\alpha^{l-1}$ is 0-valent and $\alpha^l$ is 1-valent.
- But $\alpha^{l-1}$ and $\alpha^l$ differ after round $k+1$ only in the state of one process, $j_l$.
- We can extend both $\alpha^{l-1}$ and $\alpha^l$ by simply failing $j_l$ at beginning of round $k+2$.
  - There is actually a round $k+2$ because we've assumed $k < f-1$, so $k+2 \leq f$.
- And no one left alive can tell the difference!
- Contradiction for Case 1.
Case 2: Every $\alpha^l$ is 0-valent

• Then compare:
  – $\alpha^m$, in which i sends all its round k+1 messages and then fails, with
  – $\alpha^*$, in which i sends all its round k+1 messages and does not fail.
• No other differences, since only i fails at round k+1 in $\alpha^m$.
• $\alpha^m$ is 0-valent and $\alpha^*$ is 1-valent.
• Extend to full f-round executions:
  – $\alpha^m$, by allowing no further failures,
  – $\alpha^*$, by failing i right after round k+1 and then allowing no further failures.
• No one can tell the difference.
• Contradiction for Case 2.

• So we’ve proved:
• Lemma 2: For every k, $0 \leq k \leq f-1$, there is a bivalent k-round execution.
And now the final round…

- **Lemma 3:** There is an $f$-round execution in which two nonfaulty processes decide differently.
- **Contradicts the problem requirements.**
- **Proof:**
  - Use Lemma 2 to get a bivalent $(f-1)$-round execution $\alpha$ with $\leq f-1$ failures.
  - In every 1-round extension of $\alpha$, everyone who hasn’t failed must decide (and agree).
  - Let $\alpha^*$ be the 1-round extension of $\alpha$ in which no new failures occur in round $f$.
  - Everyone who is still alive decides after $\alpha^*$, and they must decide the same thing. WLOG, say they decide 1.
  - Since $\alpha$ is bivalent, there must be another 1-round extension of $\alpha$, say $\alpha^0$, in which some nonfaulty process decides 0 (and hence, all decide 0).
Disagreement after $f$ rounds

- In $\alpha^0$, some single process $i$ fails in round $f$.
- Let $j$, $k$ be two nonfaulty processes.
- Define a chain of three $f$-round executions, $\alpha^0, \alpha^1, \alpha^*$, where $\alpha^1$ is identical to $\alpha^0$ except that $i$ sends to $j$ in $\alpha^1$ (it might not in $\alpha^0$).

- Then $\alpha^1 \sim^k \alpha^0$.
- Since $k$ decides 0 in $\alpha^0$, $k$ also decides 0 in $\alpha^1$.
- Also, $\alpha^1 \sim^j \alpha^*$.
- Since $j$ decides 1 in $\alpha^*$, $j$ also decides 1 in $\alpha^1$.
- Yields disagreement in $\alpha^1$, contradiction!

- So we have proved:
- **Lemma 3**: There is an $f$-round execution in which two nonfaulty processes decide differently.
- Which immediately yields the impossibility result.
Early-stopping agreement algorithms

- Tolerate \( f \) failures in general, but in executions with \( f' < f \) failures, terminate faster.
- [Dolev, Reischuk, Strong 90] Stopping agreement algorithm in which all nonfaulty processes terminate in \( \leq \min(f' + 2, f+1) \) rounds.
  - If \( f' + 2 \leq f \), decide “early”, within \( f' + 2 \) rounds; in any case decide within \( f+1 \) rounds.
- [Keidar, Rajsbaum 02] Lower bound of \( f' + 2 \) for early-stopping agreement.
  - Not just \( f' + 1 \). Early stopping requires an extra round.

**Theorem 8:** Assume \( 0 \leq f' \leq f – 2 \) and \( f < n \). Every early-stopping agreement algorithm tolerating \( f \) failures has an execution with \( f' \) failures in which some nonfaulty process doesn’t decide by the end of round \( f' + 1 \).
Special case: \( f' = 0 \)

- **Theorem 9:** Assume \( 2 \leq f < n \). Every early-stopping agreement algorithm tolerating \( f \) failures has a failure-free execution in which some nonfaulty process does not decide by the end of round 1.

- **Definition:** Let \( \alpha \) be an execution that completes some finite number (possibly 0) of rounds. Then \( \text{val}(\alpha) \) is the unique decision value in the extension of \( \alpha \) with no new failures.
  - Different from bivalence defs---now consider value in just one extension.

- **Proof:**
  - Again, assume executions in which at most one process fails per round.
  - Identify 0-round executions with vectors of initial values.
  - Assume, for contradiction, that everyone decides by round 1, in all failure-free executions.
  - \( \text{val}(000\ldots0) = 0, \text{val}(111\ldots1) = 1 \).
  - So there must be two 0-round executions \( \alpha^0 \) and \( \alpha^1 \), that differ in the value of just one process \( i \), such that \( \text{val}(\alpha^0) = 0 \) and \( \text{val}(\alpha^1) = 1 \).
Special case: $f' = 0$

• 0-round executions $\alpha^0$ and $\alpha^1$, differing only in the initial value of process $i$, such that $\text{val}(\alpha^0) = 0$ and $\text{val}(\alpha^1) = 1$.
• In the ff extensions of $\alpha^0$ and $\alpha^1$, all nonfaulty processes decide in just one round.
• Define:
  – $\beta^0$, 1-round extension of $\alpha^0$, in which process $i$ fails, sends only to $j$.
  – $\beta^1$, 1-round extension of $\alpha^1$, in which process $i$ fails, sends only to $j$.
• Then:
  – $\beta^0$ looks to $j$ like ff extension of $\alpha^0$, so $j$ decides 0 in $\beta^0$ after 1 round.
  – $\beta^1$ looks to $j$ like ff extension of $\alpha^1$, so $j$ decides 1 in $\beta^1$ after 1 round.
• $\beta^0$ and $\beta^1$ are indistinguishable to all processes except $i$, $j$.
• Define:
  – $\gamma^0$, infinite extension of $\beta^0$, in which process $j$ fails right after round 1.
  – $\gamma^1$, infinite extension of $\beta^1$, in which process $j$ fails right after round 1.
• By agreement, all nonfaulty processes must decide 0 in $\gamma^0$, 1 in $\gamma^1$.
• But $\gamma^0$ and $\gamma^1$ are indistinguishable to all nonfaulty processes, so they can’t decide differently, contradiction.
Next time…

- Other kinds of consensus problems:
  - k-agreement
  - Approximate agreement (skim)
  - Distributed commit

- Reading: Chapter 7