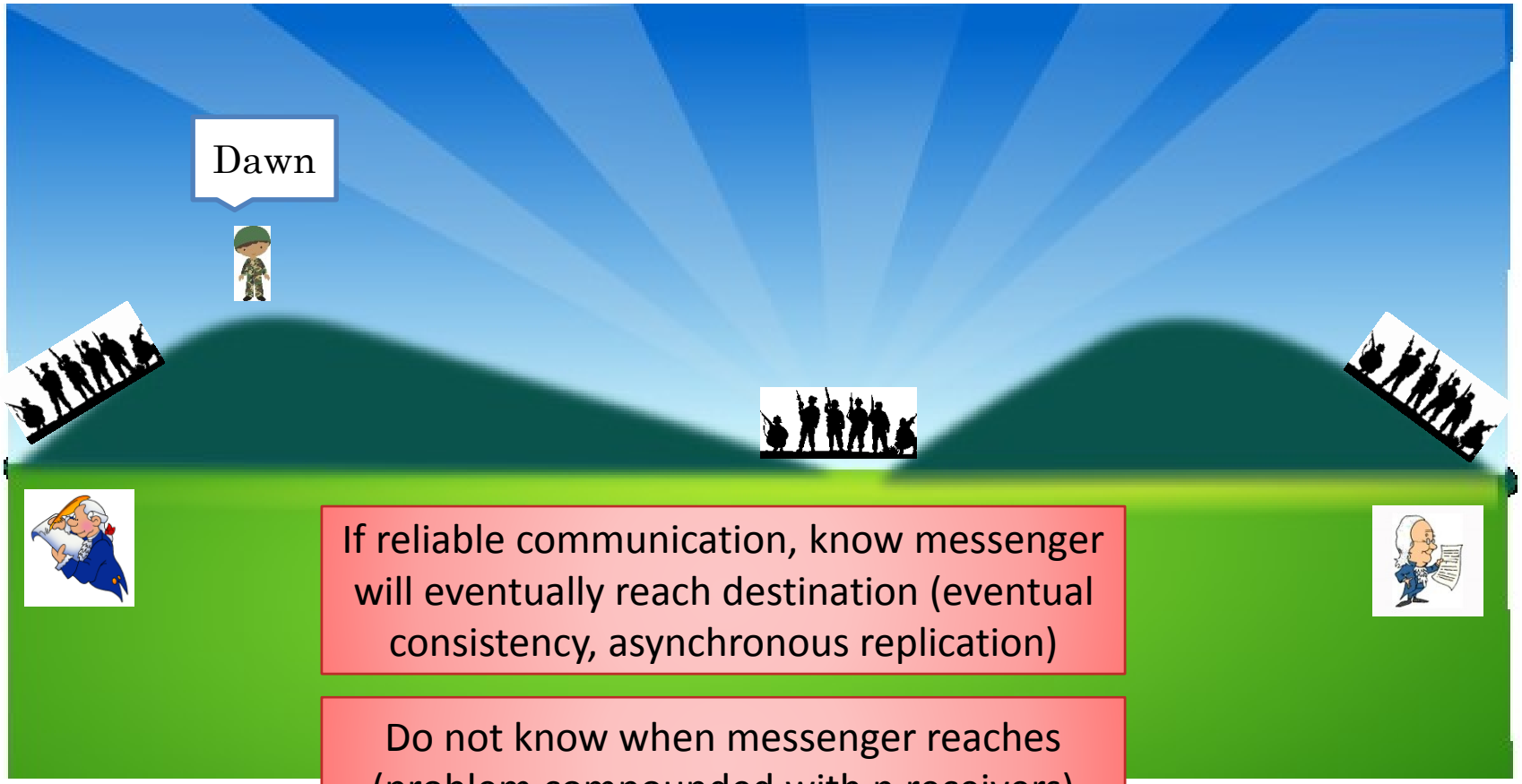
Distributed Consensus-Part 3: (Semi) Synchronous Replication and Acceptors

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)

Asynchronous Consistency Scenarios (Last Slide of Part 2)

- Multiple client UIs commit to single server ✓
  - Browser-Sakai
- Nested transaction involving multiple logical servers ✗
  - Travelocity (non replicated)
- Physical replication with multiple changers ✗
  - Diff-based with divergence (Git) ✗
  - Snapshot-based (Google Drive, OneDrive) ✗
  - Command-based: replicated state machines (Google Docs, LiveMeeting) ✗
- Lock and other meta/configuration state ✗
  - Live Meeting
- Physical mirroring ✓
  - Akamai
- Master (primary)-slave(backup) replication ✓ ✗
- Master-master replication ✗
  - Disjoint writes ✗
  - Overlapping writes ✗

Leading-Following General Problem: Immediate Consistency



If reliable communication, know messenger will eventually reach destination (eventual consistency, asynchronous replication)

Do not know when messenger reaches (problem compounded with n receivers)

Do not know if general (process) has looked at message

Asynchronous Algorithm

```
protected void localPropose(float aProposalNumber, StateType aProposal) {
  ProposalFeedbackKind aFeedbackKind = ProposalFeedbackKind.SUCCESS;
  recordSentLearnNotification(aProposalNumber, aProposal, aFeedbackKind);
  sendLearnNotification(aProposalNumber, aProposal, aFeedbackKind);
}
```

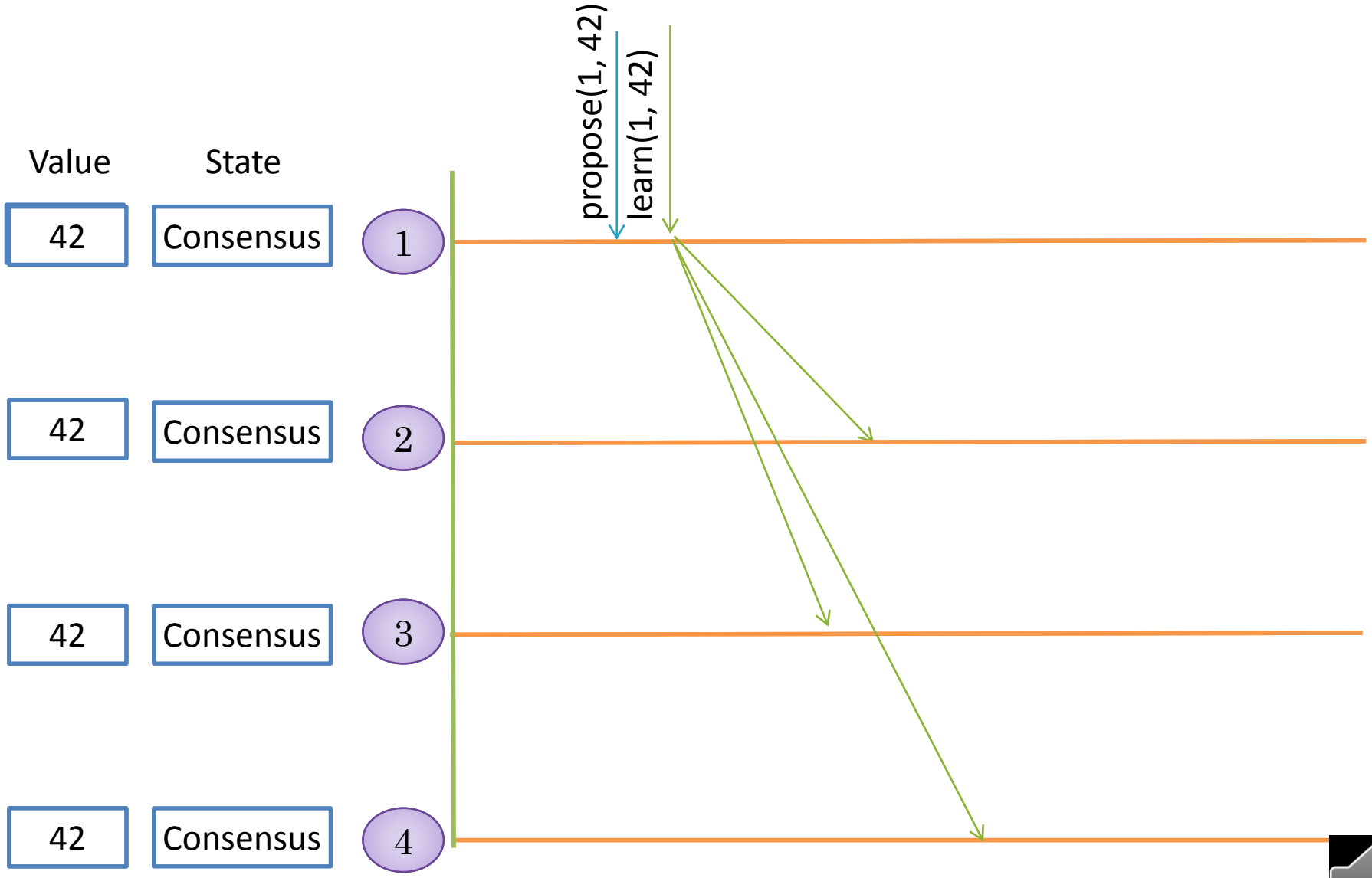
```
public synchronized void learn(float aProposalNumber, StateType aProposal,
ProposalFeedbackKind aFeedbackKind) {
  recordReceivedLearnNotification (aProposalNumber, aProposal,
aFeedbackKind);
}
```

Modify how to allow rejecting and better than asynchronous consistency?

Replication Synchrony Extremes

- Terminology:
  - D:set of distributed processes/agents that need some level of replication synchrony
  - O: object to be replicated
  - replica(d): version of O kept by process d.
  - value(p): value proposed by proposal p
  - commit(p, d): time when replica(d) is assigned value(p).
  - submit(p): time when p is issued.
  - state(d, t): Value of replica(d) at time d
  - proposer(p): process that issues p.
  - T: set of all times after some replica starts
- Asynchronous value replication:
  - $\text{commit}(p, d) - \text{submit}(p))|$ is finite $\forall\, d \in D$
- Pure synchronous (value) replication:
  - $\forall\, t \in T, \forall\, i, j \in D, \text{state}(i, t) = \text{state}(j, t)$
- Changing D?
  - Assume process will eventually start and has checkpointed, so D can be a static set

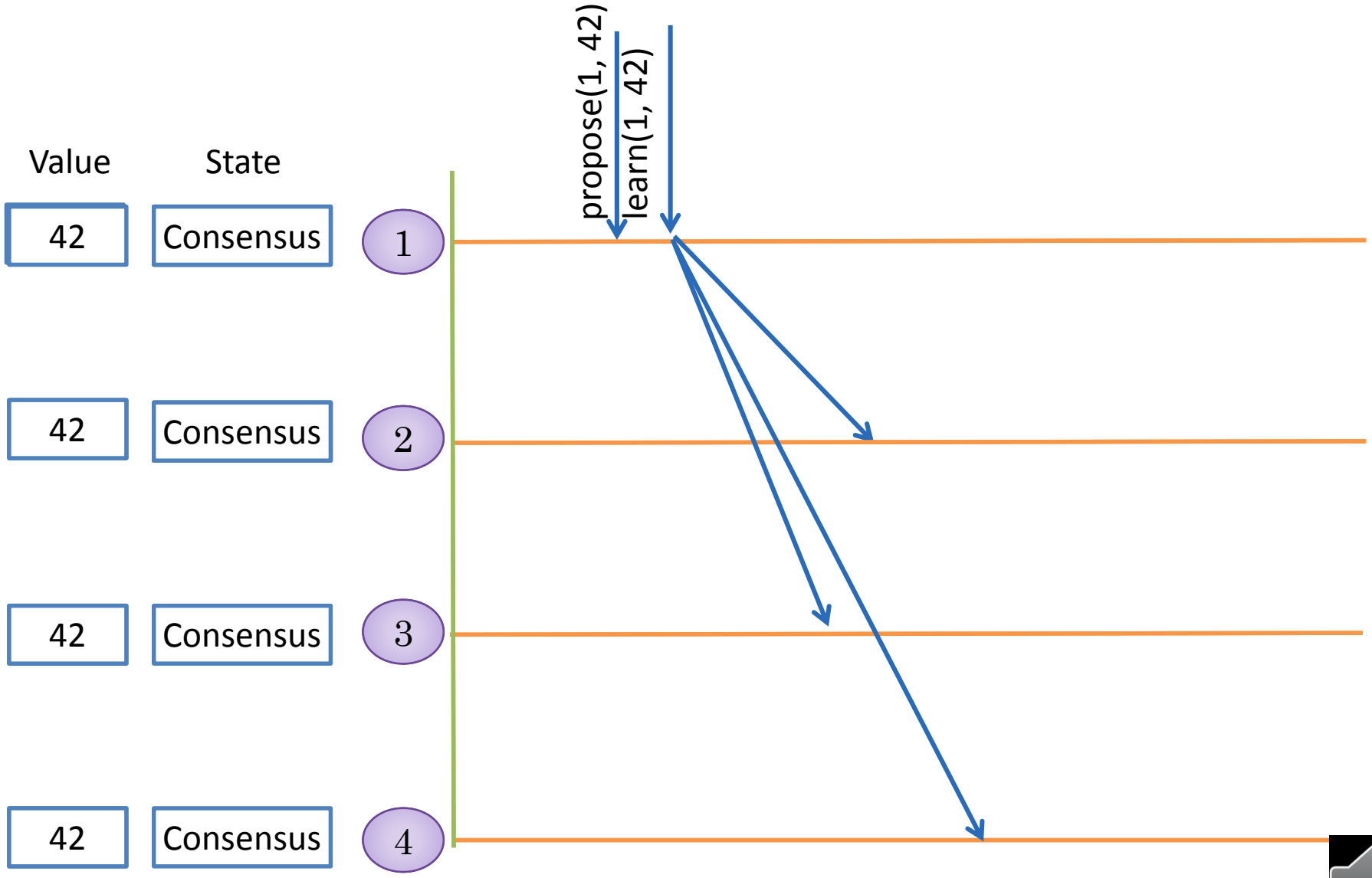
Asynchronous Value Replication



Replication Synchrony Extremes (Review)

- Terminology:
  - D:set of distributed processes/agents that need some level of replication synchrony
  - O: object to be replicated
  - replica(d): version of O kept by process d.
  - value(p): value proposed by proposal p
  - commit(p, d): time when replica(d) is assigned value(p).
  - submit(p): time when p is issued.
  - state(d, t): Value of replica(d) at time d
  - proposer(p): process that issues p.
  - T: set of all times after some replica starts
- Asynchronous value replication:
  - commit(p, d) – submit(p)) | is finite $\forall$ d $\in$ D
- Pure synchronous (value) replication:
  - $\forall$ t $\in$ T, $\forall$ i, j $\in$ D, state(i, t) = state (j, t)
- Changing D?
  - Assume process will eventually start and has checkpointed, so D can be a static set

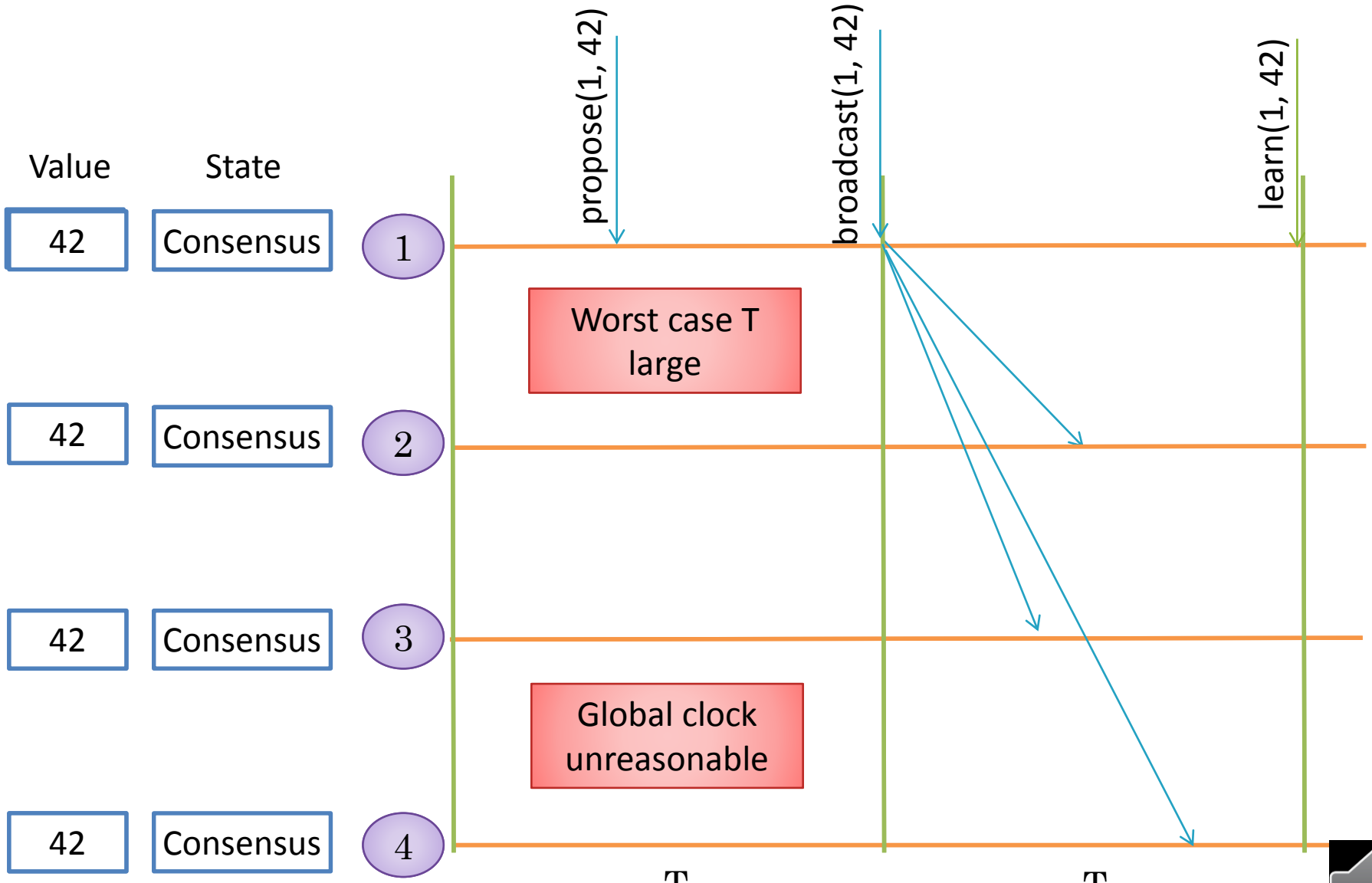
Asynchronous Value Replication



Synchronous Replication Assumptions

- Synchronous Distributed Systems
  - Global clock
  - Upper bound on successful message transmission time (T^N)
- Upper bound on when process receives message after arrival (T^S).
- $T = T^N + T^S$

Synchronous Value Replication



Alternative Replication Synchrony

Protocol Change

- Terminology:
  - D:set of distributed processes/agents that need some level of replication synchrony
  - O: object to be replicated
  - Replica(d): version of O kept by process d.
  - Value(p): value proposed by proposal p
  - Commit(p, d): time when replica(d) is assigned value(p).
  - Submit(p): time when p is issued.
  - State(d, t): Value of Replica(d) at time d
  - Proposer(p): process that issues p.
  - T: set of all times when system is up
- Asynchronous value replication:
  - $|\text{Commit}(p, \text{Proposer}(p) - \text{Commit}(p, d))|$ is finite $\forall\, d \in D$
- Pure Synchronous value replication:
  - $\forall\, t \in T, \forall\, i, j \in D, \text{State}(i, t) = \text{State}(j, t)$
- *Synchronous (invalidation) replication*
  - $\forall\, t \in T, \forall\, i, j \in D, \text{if State}(i, t) \mathrel{!=} \text{null}, \text{State}(j, t) = \text{null} \mid \text{State}(j, t) = \text{State}(i, t)$

Requirement: Sending and receiving protocol be same

Asynchronous: While message in transit, sending and receiving protocol at proposer not same

Idea: Invalidate protocol rather than pure synchronous

Synchronous Replication Assumptions

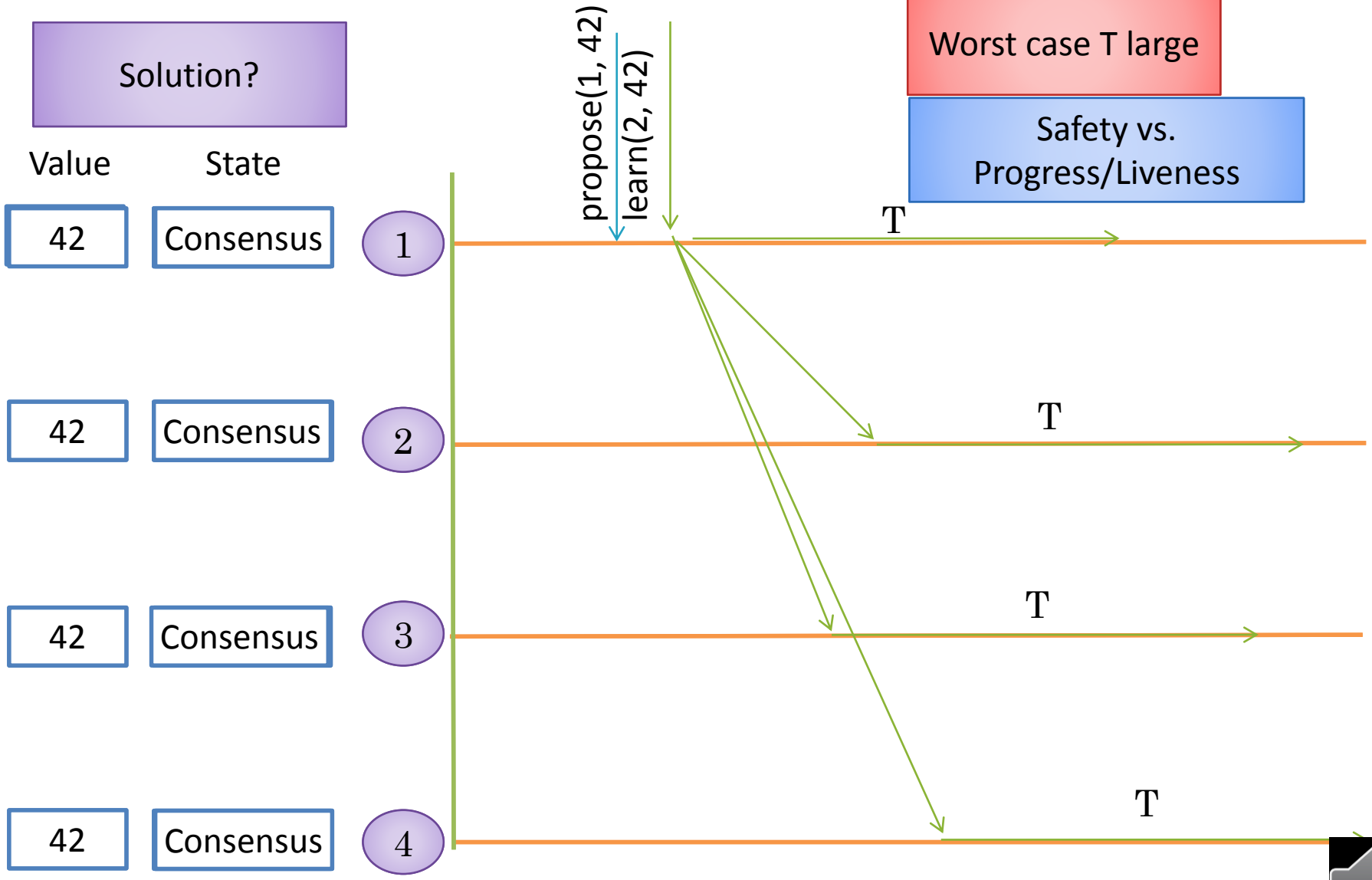
- Synchronous Distributed Systems
  - Global clock
  - Upper bound on successful message transmission time
- Upper bound on when process receives message after arrival.

One-Phase Synchronous Replication

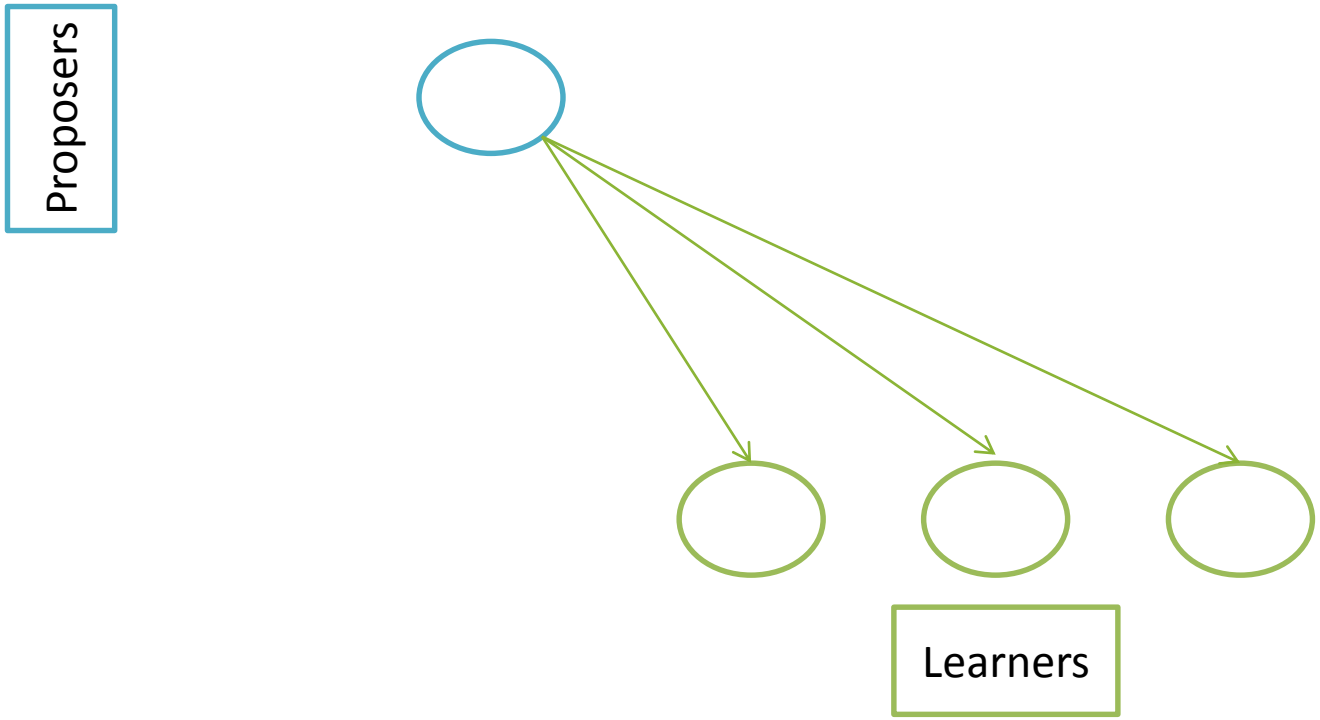
Solution?

Worst case T large

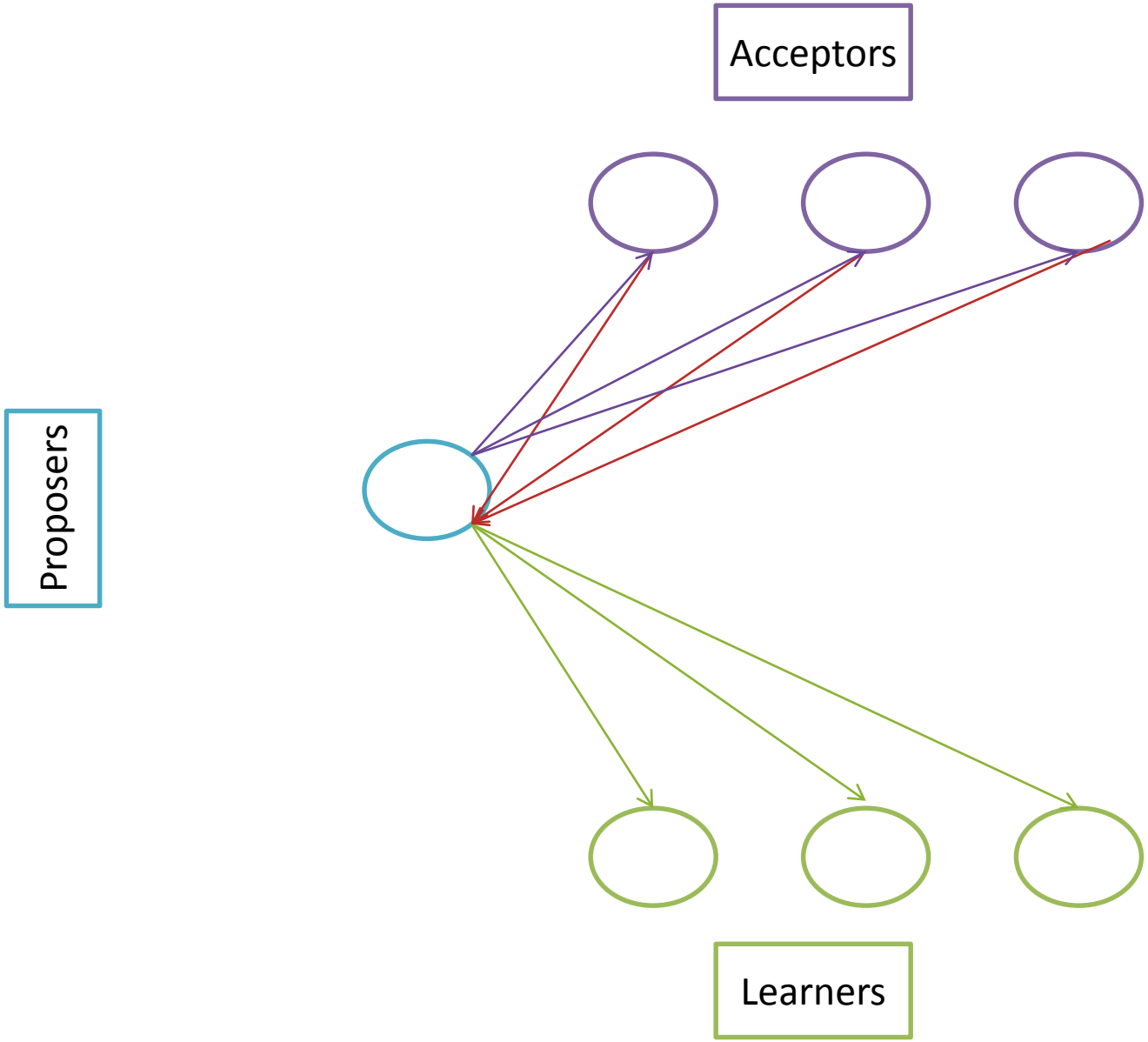
Safety vs. Progress/Liveness



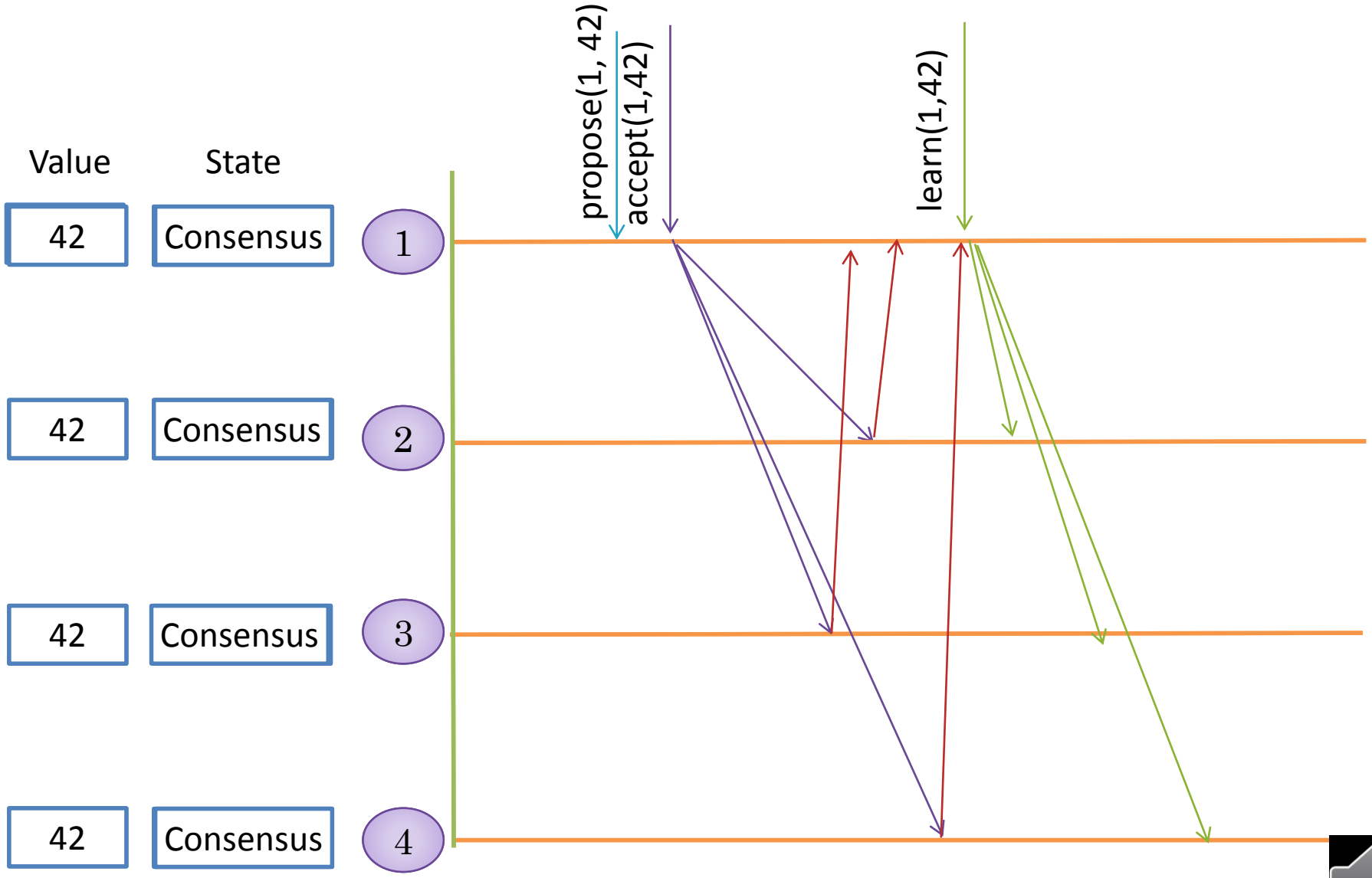
Asynchronous



Synchronous



Two-Phase Synchronous Replication



Propose

Send Accept Request to all Acceptors (instead of notification to all learners)

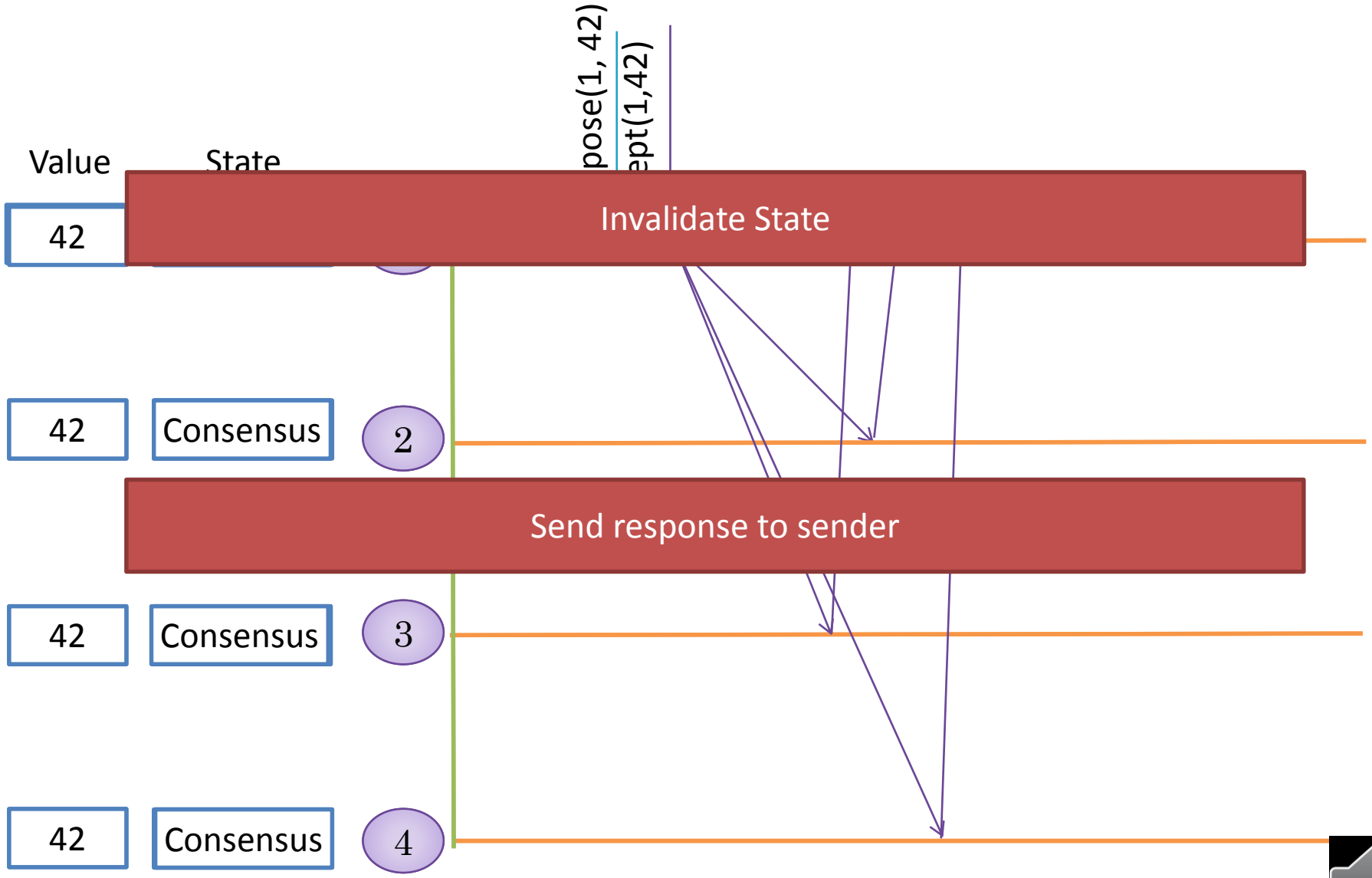
Value	State	
42	Consensus	1
42	Consensu	2
42	Consensus	3
42	Consensus	4

Acceptors will usually include self

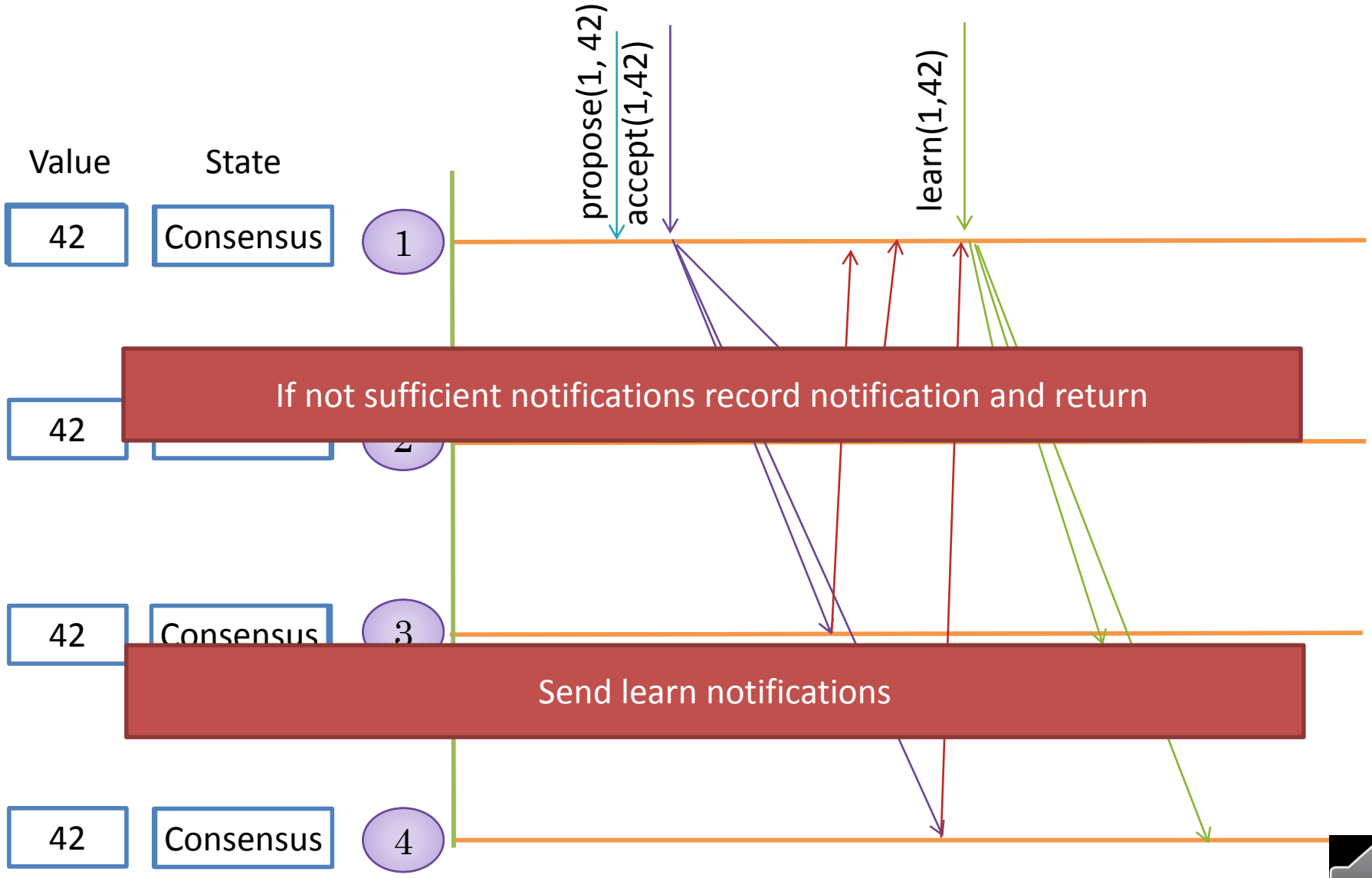
Acceptors != learners?

if learners can be asynchronous (slaves vs mirrors)

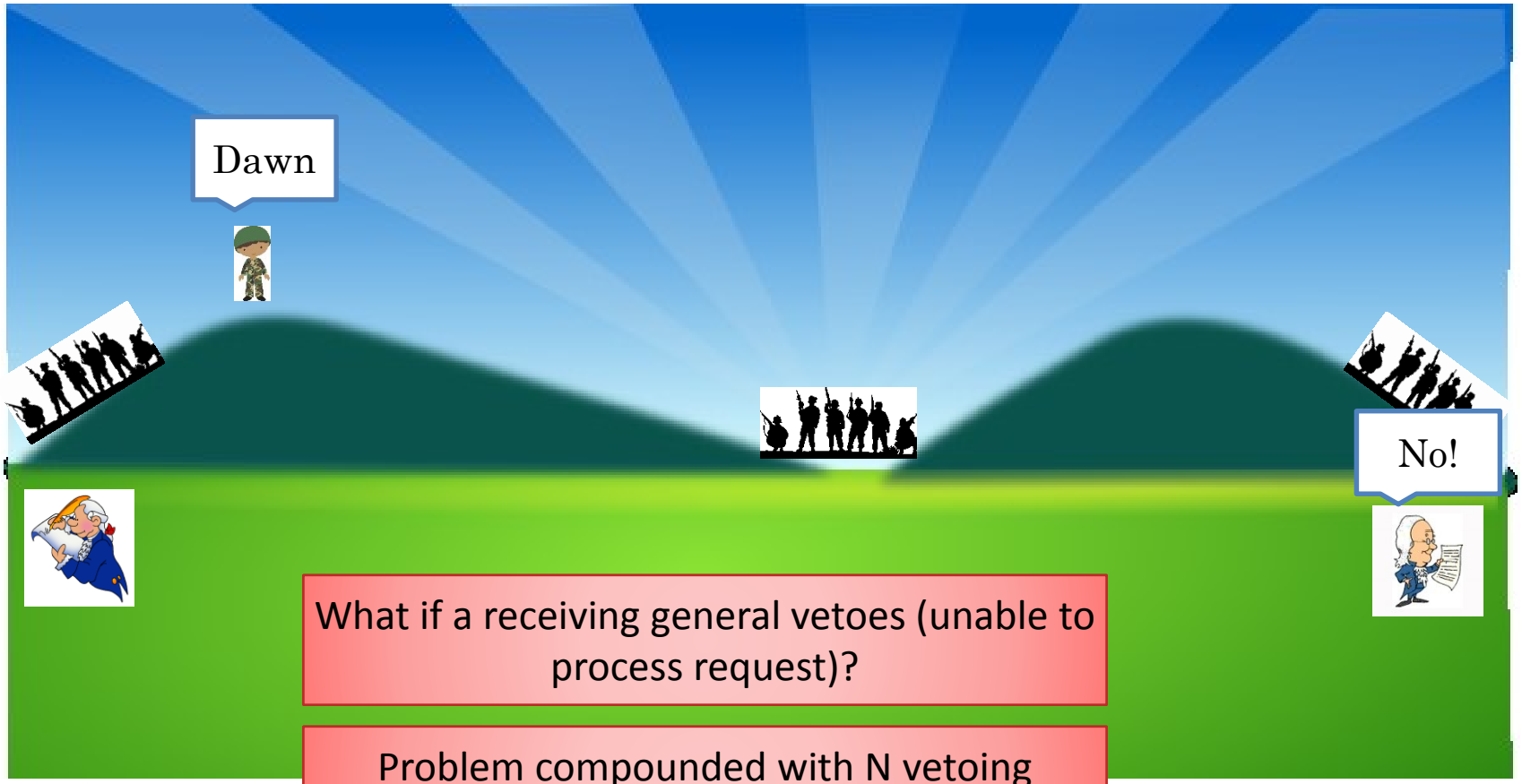
Accept


pose(1, 42)
ept(1,42)
Value
State
42
Invalidate State
42
Consensus
2
Send response to sender
42
Consensus
3
42
Consensus
4

Accepted


propose(1, 42)
accept(1,42)
learn(1,42)
Value
State
42
Consensus
1
42
2
42
Consensus
3
42
Consensus
4
If not sufficient notifications record notification and return
Send learn notifications

Leading-Thinking General Problem

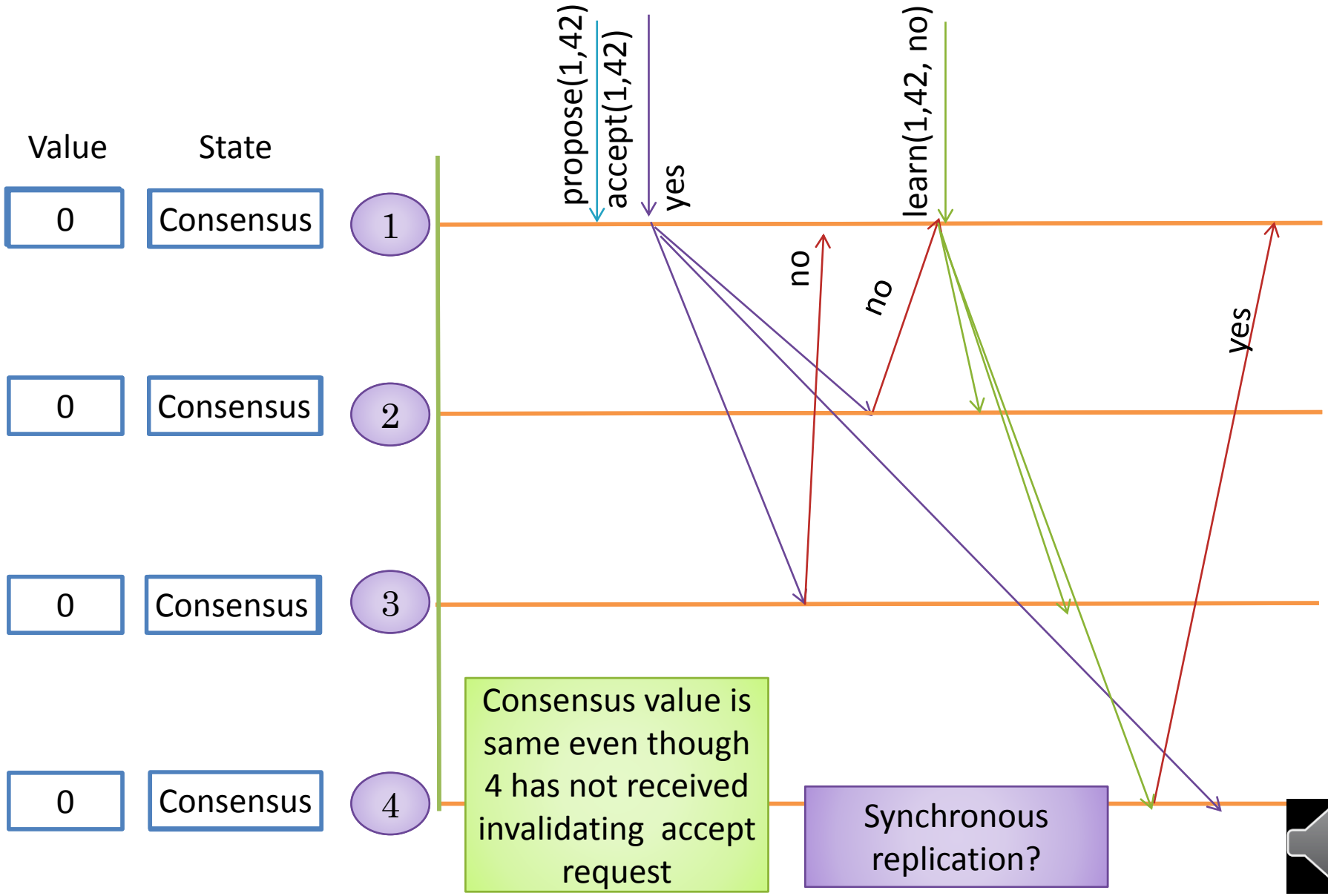


What if a receiving general vetoes (unable to process request)?

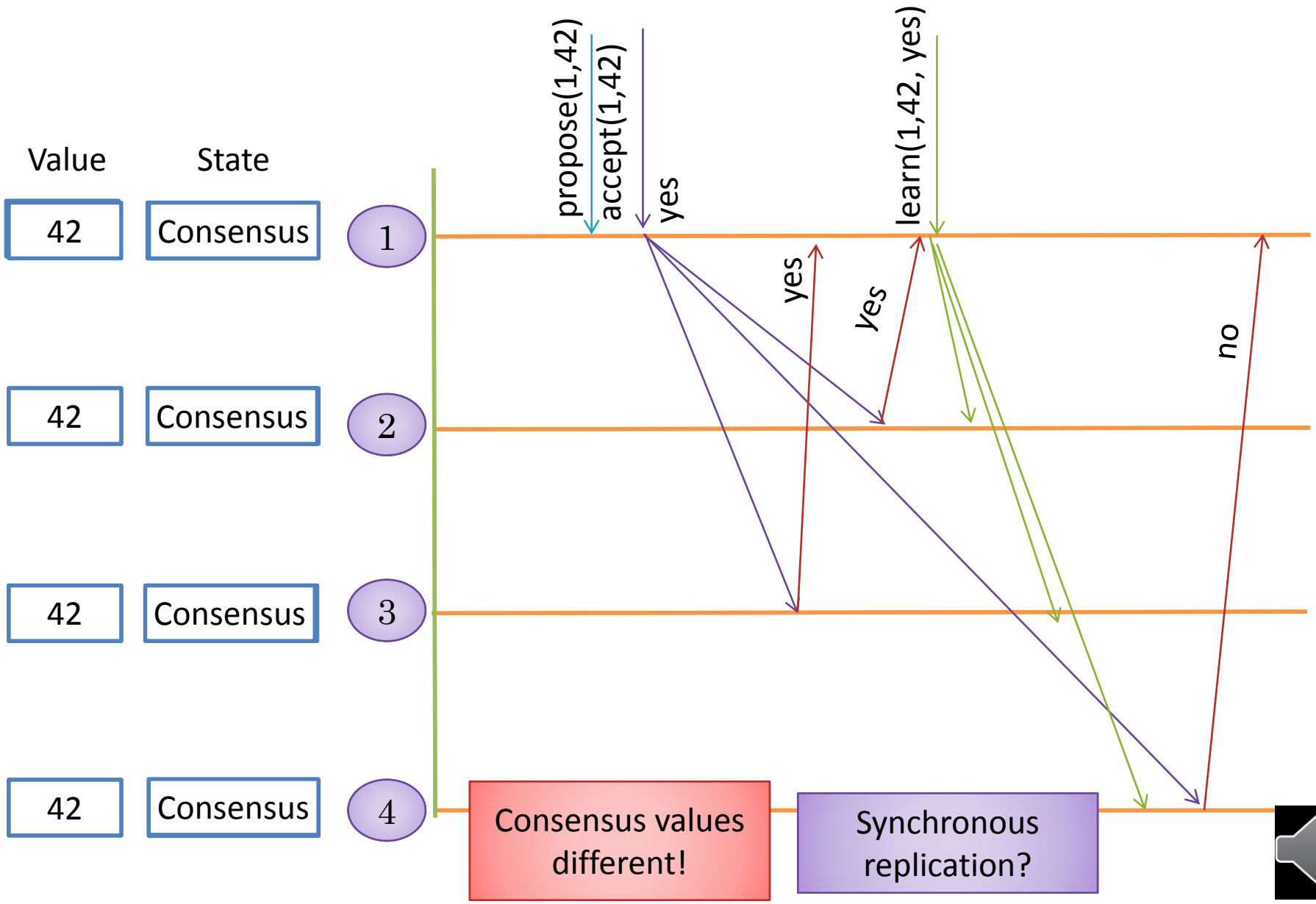
Problem compounded with N vetoing generals

Can take majority or allow veto power

Feedback & Negative Learn



Semi-Synchronous Replication



Semi-Synchronous Replication

- Terminology:
  - D:set of distributed processes/agents that need some level of replication synchrony
  - O: object to be replicated
  - Replica(d): version of O kept by process d.
  - Value(p): value proposed by proposal p
  - Commit(p, d): time when replica(d) is assigned value(p).
  - Submit(p): time when p is issued.
  - State(d, t): Value of Replica(d) at time d
  - Proposer(p): process that issues p.
  - T: set of all times when system is up
- Asynchronous value replication:
  - $|\text{Commit}(p, \text{Proposer}(p) - \text{Commit}(p, d))|$ is finite $\forall\, d \in D$
- Pure synchronous value replication:
  - $\forall\, t \in T, \forall\, i, j \in D, \text{State}(i, t) = \text{State}(j, t)$
- Synchronous (invalidation) replication
  - $\forall\, t \in T, \forall\, i, j \in D$, if State(i, t) != null, State (j, t) = null | State(j, t) = State(i, t)
- Semi-Synchronous (invalidation) replication
  - $\forall\, t \in T, \forall\, i, j \in E$, if State(i, t) != null, State (j, t) = null | State(j, t) = State(i, t)
  - $E \subseteq D$, E != Ø

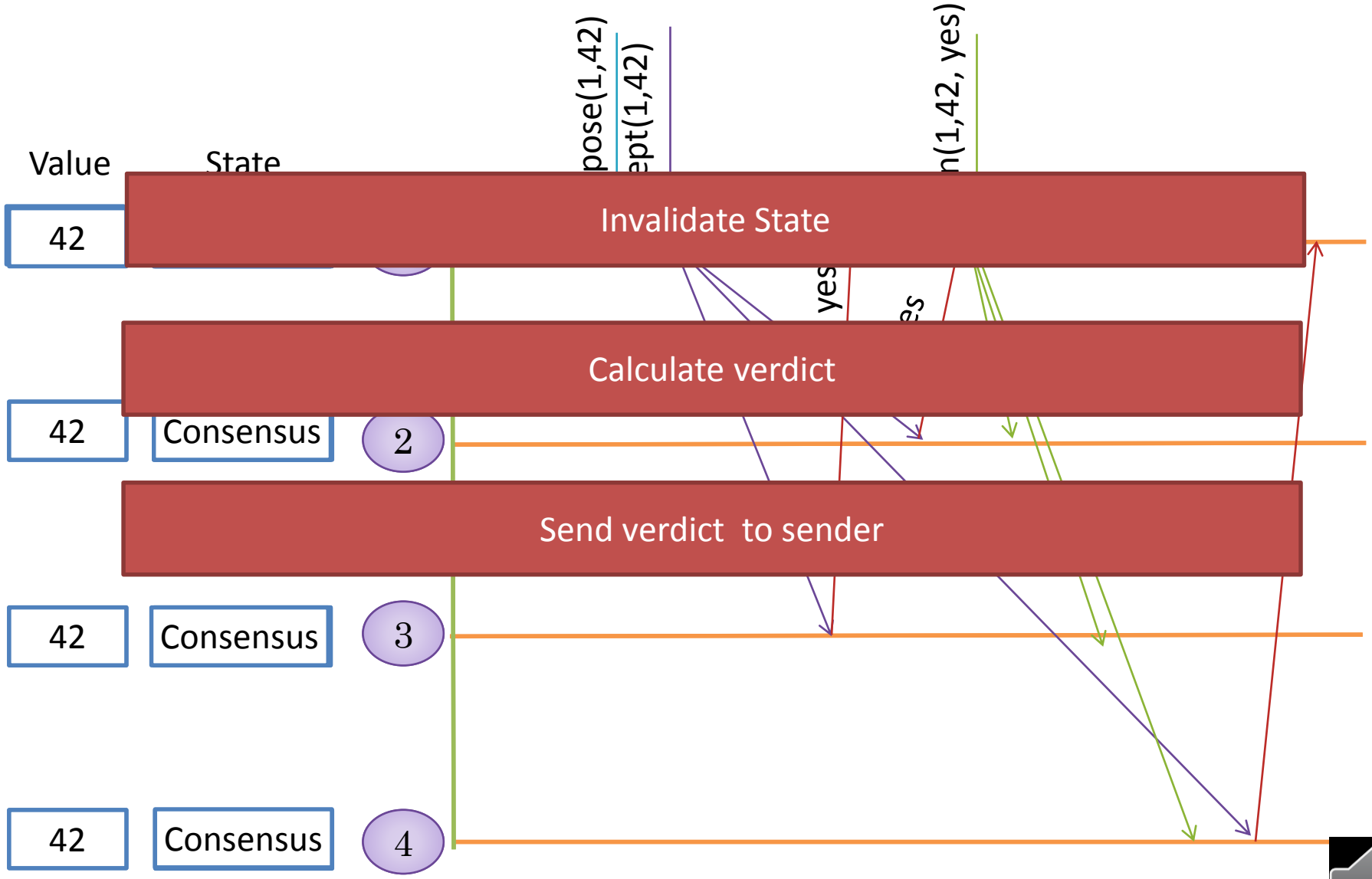
Replication Synchrony

```
public enum ReplicationSynchrony {
  ASYNCHRONOUS,
  TWO_SYNCHRONOUS,
  MAJORITY_SYNCHRONOUS,
  ALL_SYNCHRONOUS,
  CUSTOM_SYNCHRONOUS
}
```

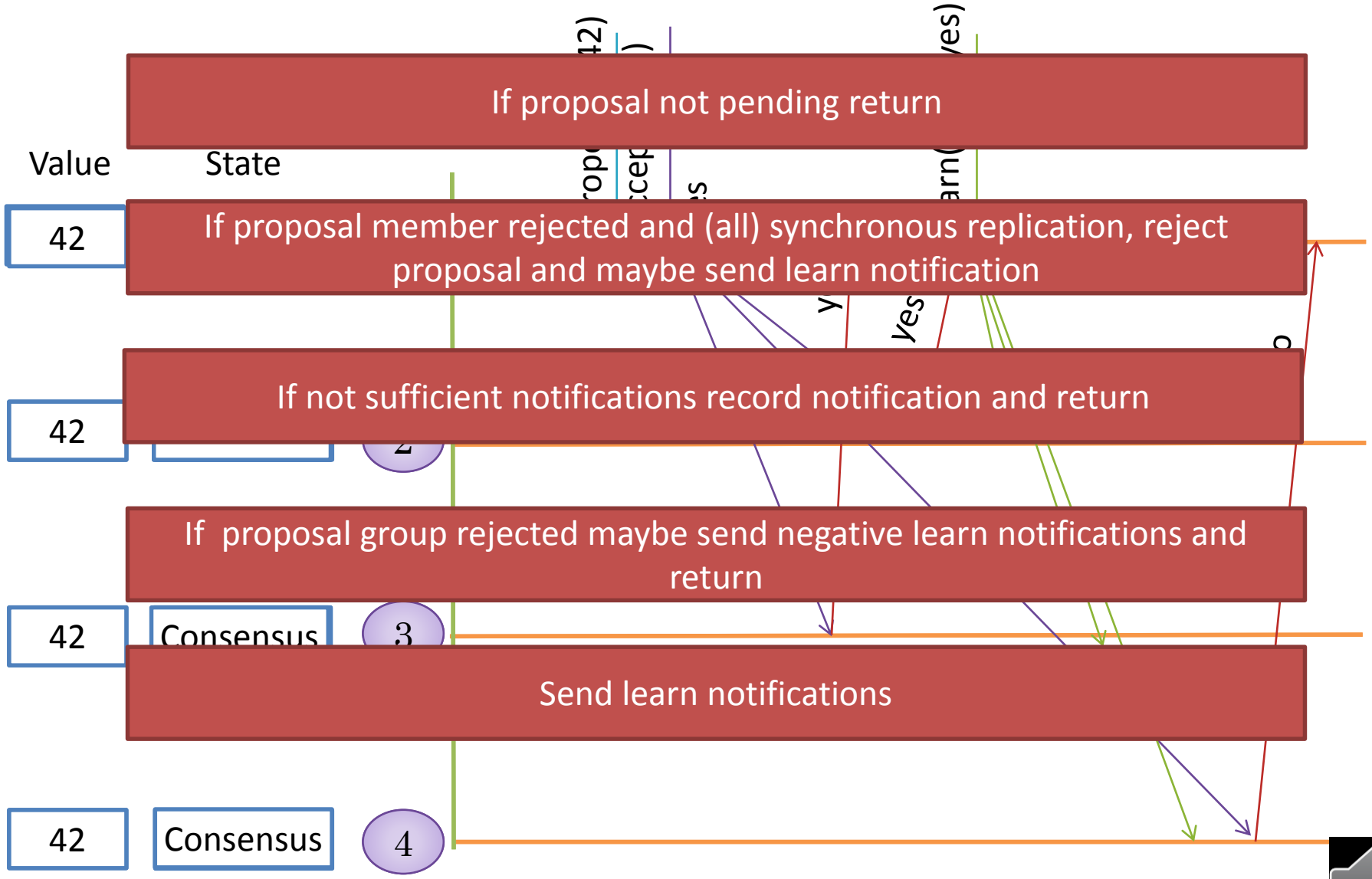
Majority in a dynamic set of acceptors?

```
public enum ConsensusMemberSetKind {
  CURRENT_MEMBERS,
  INITIAL_MEMBERS,
  MAXIMUM_MEMBERS
}
```

Accept


Value
State
pose(1,42)
ept(1,42)
n(1,42, yes)
yes
es
Invalidate State
Calculate verdict
Send verdict to sender
42
42
Consensus
2
42
Consensus
3
42
Consensus
4

Accepted


If proposal not pending return
Value
State
42
If proposal member rejected and (all) synchronous replication, reject proposal and maybe send learn notification
yes
If not sufficient notifications record notification and return
42
2
If proposal group rejected maybe send negative learn notifications and return
42
Consensus
3
Send learn notifications
42
Consensus
4

Accept

```
public void accept(float aProposalNumber, StateType aProposal) {
  recordReceivedAcceptRequest(aProposalNumber, aProposal);
  if (!isPending (aProposalNumber) &&
        !isSendAcceptReplyForResolvedProposal()) {
    return;
  }
  ProposalFeedbackKind aFeedbackKind =
        checkAcceptRequest(aProposalNumber, aProposal);
  if (!isSuccess(aFeedbackKind) && !isSendRejectionNotification()) {
    return;
  }
  recordAndSendAcceptedNotifica                                  al,
        aFeedbackKind);
}
```

Why not return value?

Do not want to tie to RPC

Caller may not wait for response

Response may be sent to others than proposal sender

Generalized Propose

```
protected void localPropose(float aProposalNumber, StateType aProposal) {
  if (isAsynchronousConsistency()) {
     sendLearnNotification(aProposalNumber, aProposal,
          ProposalFeedbackKind.SUCCESS);
  } else {
     sendAcceptRequest(aProposalNumber, aProposal);
  }
}
```

Accepted

```
public void accepted(float aProposalNumber, StateType aProposal,
        ProposalFeedbackKind aFeedbackKind) {
  ProposalAcceptResponseReceived.newCase(this, getObjectName(),
  ProposalNumber, aProposal, aFeedbackKind);
  recordReceivedAcceptedNotification(aProposalNumber, aProposal,
        aFeedbackKind);
  if (!isPending(aProposalNumber)) {
    return;
  }
  if (!isSuccess(aFeedbackKind) && isAllSynchronous()) {
    processAcceptRejection(aProposalNumber, aProposal, aFeedbackKind);
    return;
  }
  aggregateAcceptedNotification(
        aProposalNumber, aProposal, aFeedbackKind);
}
```

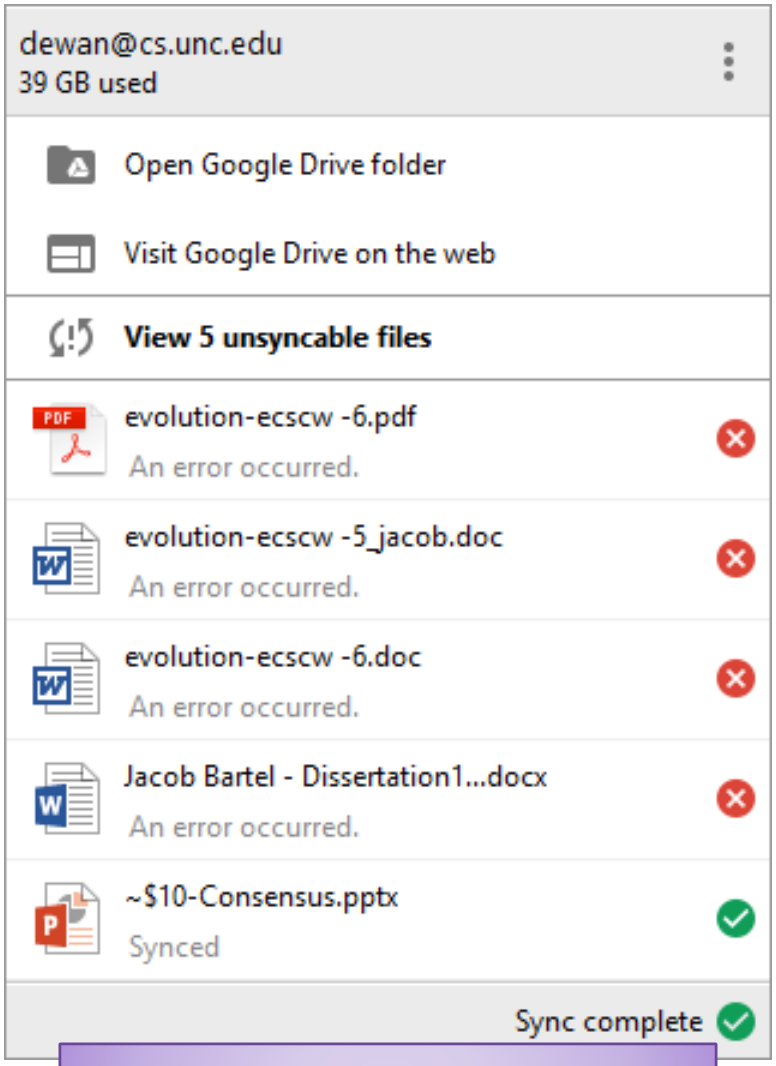
Aggregation

```
protected void aggregateAcceptedNotification(
        float aProposalNumber, StateType aProposal,
        ProposalFeedbackKind aFeedbackKind) {
  Boolean isSufficientAcceptors = sufficientAcceptors(
        aProposalNumber, aProposal);
  if (isSufficientAcceptors == null)
    return;
  if (isSufficientAcceptors) {
    recordAndSendLearnNotification(aProposalNumber,
       aProposal, ProposalFeedbackKind.SUCCESS);

  } else {
    processAcceptRejection(aProposalNumber, aProposal,
      ProposalFeedbackKind.AGGREGATE_DENIAL);

  }
}
```

Practical Application with Rejection



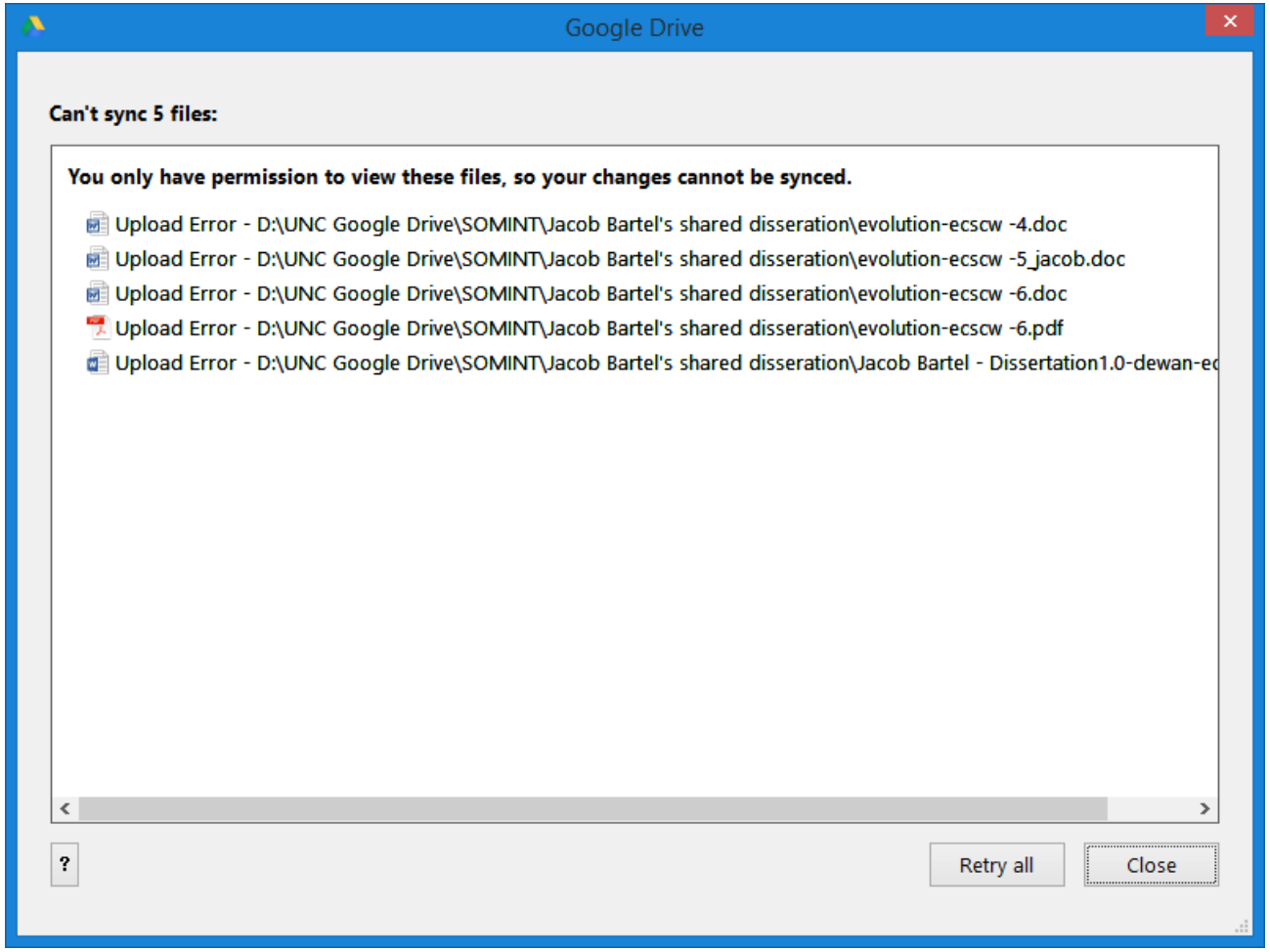
Want conflicts to not cause inconsistent files

Want to know if file actually savable by server

Want to know if file has reached server before shutting down laptop

Reason for not accepting?

Practical Reason for Rejecting



Feedback Kind and Proposal State

```
public enum ProposalFeedbackKind {
   SUCCESS,
   SERVICE_FAULT,
   ACCESS_DENIAL,
   SERVICE_DENIAL,
   CONCURRENT_OPERATION,
   AGGREGATE_DENIAL // consolidated response
}
```

Checks made by application vs consensus abstraction?

Abstraction

```
public enum ProposalState {
   PROPOSAL_CONSENSUS,
   PROPOSAL_PENDING,
   PROPOSAL_SERVICE_FAULT,
   PROPOSAL_ACCESS_DENIAL,
   PROPOSAL_SERVICE_DENIAL,
   PROPOSAL_CONCURRENT_OPERATION,
   PROPOSAL_AGGREGATE_DENIAL
}
```

How does application get involved?

Vetoer Design Pattern

Consensus Mechanism <StateType>

```
void addConsensusListener(ConsensusListener<StateType> aConsensusListener);
```

```
void addConsensusVetoer(ProposalVetoer<StateType> aConsensusVetoer);
```

Feedback Kind and Proposal State

```
public class AMeaningOfLifeVetoer implements ProposalVetoer<Integer>{
public ProposalFeedbackKind acceptProposal(float aProposalNumber, Integer
aState) {
  return aState % 2 != 0?
         ProposalFeedbackKind.SERVICE_DENIAL:ProposalFeedbackKind.SUCCESS;
}
}
```

```
 protected synchronized ProposalFeedbackKind checkProposalForAccept(
         float aProposalNumber, StateType aProposal ) {
   return checkWithVetoer(aProposalNumber, aProposal);
 }
 protected synchronized ProposalFeedbackKind checkWithVetoer(
         float aProposalNumber, StateType aProposal ) {
   for (ProposalVetoer<StateType> aConsensusVetoer:consensusVetoers){
     ProposalFeedbackKind aRejectionKind =
       aConsensusVetoer.acceptProposal(aProposalNumber, aProposal);
     if (aRejectionKind != ProposalFeedbackKind.SUCCESS) {
       return aRejectionKind;
     }
   }
   return ProposalFeedbackKind.SUCCESS;
}
```

Following Acceptor 3 (Synchronous)

3

```
I***(ProposalAcceptRequestReceived)  Meaning,1.0002=29
I***(ProposalAcceptedNotificationSent)  Meaning,1.0002=29-->SUCCESS
I***(ProposalAcceptRequestReceived)  Meaning,1.0001=42
I***(ProposalAcceptedNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I***(ProposalLearnNotificationReceived)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(ProposalStateChanged)  Meaning,1.0002=29-->PROPOSAL_SERVICE_DENIAL
```

Thinking Acceptor 4 (Synchronous)

4

```
I***(ProposalAcceptRequestReceived)  Meaning,1.0001=42
I***(ProposalAcceptedNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalAcceptRequestReceived)  Meaning,1.0002=29
I***(ProposalAcceptedNotificationSent)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I***(ProposalLearnNotificationReceived)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(ProposalStateChanged)  Meaning,1.0002=29-->PROPOSAL_SERVICE_DENIAL
```

Accepted Proposer 1 (Synchronous)

1

```
I***(ProposalMade)  Meaning,1.0001=42
I***(ProposalAcceptRequestSent)  Meaning,1.0001=42
I***(ProposalWaitStarted)  Meaning,1.0001=42
I***(ProposalAcceptRequestReceived)  Meaning,1.000
I***(ProposalAcceptedNotificationSent)  Meaning,1.
I***(ProposalAcceptRequestReceived)  Meaning,1.000
I***(ProposalAcceptedNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:1|1|4|4?4.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:2|2|4|4?4.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:3|3|4|4?4.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:4|4|4|4?4.0-->true
I***(ProposalLearnNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(ProposalStateChanged)  Meaning,1.0002=29-->PROPOSAL_SERVICE_DENIAL
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I***(ProposalWaitEnded)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
```

#agreements, #notifications, #current members, #total members considered

Rejected Proposer 2 (Synchronous)

2

```
I***(ProposalMade)  Meaning,1.0002=29
I***(ProposalAcceptRequestSent)  Meaning,1.0002=29
I***(ProposalWaitStarted)  Meaning,1.0002=29
I***(ProposalAcceptRequestReceived)  Meaning,1.0001=42
I***(ProposalAcceptedNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0002,29:1|1|4|4?4.0-->null
I***(ProposalAcceptRequestReceived)  Meaning,1.0002=29
I***(ProposalAcceptedNotificationSent)  Meaning,1.0002=29-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0002,29:2|2|4|4?4.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0002,29:3|3|4|4?4.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(ProposalLearnNotificationSent)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I***(ProposalLearnNotificationReceived)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(ProposalStateChanged)  Meaning,1.0002=29-->PROPOSAL_SERVICE_DENIAL
I***(ProposalWaitEnded)  Meaning,1.0002=29-->PROPOSAL_SERVICE_DENIAL
```

Accepted Proposer 2 (Two-Synchronous)

```
I***(ProposalMade)  Meaning,1.0002=29
I***(ProposalAcceptRequestSent)  Meaning,1.0002=29
I***(ProposalWaitStarted)  Meaning,1.0002=29
I***(ProposalAcceptRequestReceived)  Meaning,1.0001=42
I***(ProposalAcceptedNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0002,29:1|1|4|4?2.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SERVICE_DENIAL
I***(SufficientAgreementsChecked)  Meaning,1.0002,29:1|2|4|4?2.0-->null
I***(ProposalAcceptRequestReceived)  Meaning,1.0002=29
I***(ProposalAcceptedNotificationSent)  Meaning,1.0002=29-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0002,29:2|3|4|4?2.0-->true
I***(ProposalLearnNotificationSent)  Meaning,1.0002=29-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0002=29-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0002=29-->PROPOSAL_CONSENSUS
Meaning of Life:29
I***(ProposalWaitEnded)  Meaning,1.0002=29-->PROPOSAL_CONSENSUS
I***(ProposalAcceptResponseReceived)  Meaning,1.0002=29-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
```

Accepted Proposer 1 (Two-Synchronous)

```
I***(ProposalMade)  Meaning,1.0001=42
I***(ProposalAcceptRequestSent)  Meanin
I***(ProposalWaitStarted)  Meaning,1.00
I***(ProposalAcceptRequestReceived)  Me
I***(ProposalAcceptedNotificationSent)
I***(ProposalAcceptRequestReceived)  Meaning,1.0001=42
I***(ProposalAcceptedNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:1|1|4|4?2.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:2|2|4|4?2.0-->true
I***(ProposalLearnNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:3|3|4|4?2.0-->true
I***(ProposalLearnNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0002=29-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0002=29-->PROPOSAL_CONSENSUS
Meaning of Life:29
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:4|4|4|4?2.0-->true
I***(ProposalLearnNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I***(ProposalWaitEnded)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
```

Proposal state changed when learn invocation is later scheduled

Accepted Proposer 1 (Two-Synchronous, Immediate State Change)

```
I***(ProposalMade)  Meaning,1.0001=42
I***(ProposalAcceptRequestSent)  Meaning,1.0001=42
I***(ProposalWaitStarted)  Meaning,1.0001=42
I***(ProposalAcceptRequestReceived)  Meaning,1.0002=29
I***(ProposalAcceptedNotificationSent)  Meaning,1.0002=29-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:1|1|4|4?2.0-->null
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(SufficientAgreementsChecked)  Meaning,1.0001,42:2|2|4|4?2.0-->true
I***(ProposalLearnNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I***(ProposalWaitEnded)  Meaning,1.0001=42-->PROPOSAL_CONSENSUS
I***(ProposalAcceptRequestReceived)  Meaning,1.0001=42
I***(ProposalAcceptedNotificationSent)  Meaning,1.0001=42-->SUCCESS
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
I***(ProposalLearnNotificationReceived)  Meaning,1.0002=29-->SUCCESS
I***(ProposalStateChanged)  Meaning,1.0002=29-->PROPOSAL_CONSENSUS
Meaning of Life:29
I***(ProposalAcceptResponseReceived)  Meaning,1.0001=42-->SUCCESS
```

(Semi) Synchronous Consistency Scenarios

- Multiple client UIs commit to single server ✓
  - Browser-Sakai
- Nested transaction involving multiple logical servers ✓ ✗
  - Travelocity (non replicated)
- Physical replication with multiple changers ✗
  - Diff-based with divergence (Git) ✗
  - Snapshot-based (Google Drive, OneDrive) ✗
  - Command-based: replicated state machines (Google Docs, LiveMeeting) ✗
- Lock and other meta/configuration state ✗
  - Live Meeting
- Physical mirroring ✓
  - Akamai
- Master (primary)-slave(backup) replication ✓ ✗ ✓
- Master-master replication ✗
  - Disjoint writes ✗
  - Overlapping writes ✗