DISTRIBUTED CONSENSUS-
PART 6: PAXOS IMPLEMENTATION
AND EXECUTIONS

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)
Basic (Non Centralized) Paxos
(Central) Synchronous
**Consensus Mechanism Hierarchy**

- **An Asynchronous Consensus Mechanism**
  - learn();
  - propose();
  - addConsensusListener();
  - accept();
  - accepted();

- **A Centralizable Consensus Mechanism**
  - remotePropose();
  - prepare();
  - prepared();

- **A Paxos Consensus Mechanism**
  - prepare();
  - prepared();

**Local application operations**

**Selected application operations shown**

**Code outline for Paxos**
When to Not Use Paxos

```java
protected boolean isNotPaxos() {
    return isNonAtomic() || isCentralizedPropose();
}
```
**Propose First Phase**

```java
protected void localPropose(float aProposalNumber,
StateType aProposal) {
    if (isNotPaxos()) {
        super.localPropose(aProposalNumber, aProposal);
    } else {
        startPreparePhase(aProposalNumber, aProposal);
    }
}

protected void startPreparePhase(float aProposalNumber,
StateType aProposal) {
    recordAndSendPrepareRequest(aProposalNumber, aProposal);
}
```
Each acceptor calculates and sends back last accepted value and its proposal number (or optionally last seen proposal number if no acceptance so far) and updates last seen proposal number.
**State-based Rejection**

State-based rejection should be done as early as possible, so in prepare rather than accept phase and makes the safety condition in Lamport’s algorithm stronger.

```java
protected synchronized ProposalFeedbackKind checkPrepareRequest(float aProposalNumber, StateType aProposal) {
    return isPrepareConcurrencyConflict(aProposalNumber, aProposal)?
        ProposalFeedbackKind.CONCURRENCY_CONFLICT:
        checkWithVetoer(aProposalNumber, aProposal);
}
```

Single site cannot veto if majority used, rejection equal to not responding, which means majority wins if its response not considered.
**Prepare Concurrency-based Rejection**

```java
protected boolean isPrepareConcurrencyConflict (float aProposalNumber, 
                                             StateType aState ) {

    return 
        maxProposalNumberReceivedInPrepareOrAcceptRequest > 
        aProposalNumber;
}
```

Preparer abandons proposal if it learns about a higher number proposal
Public void prepare(float aProposalNumber, StateType aProposal) {
    float aPreparedOrAcceptedProposalNumber = 
        maxProposalNumberSentInSuccessfulAcceptedNotification ;
    StateType anAcceptedState = null;
    if (aPreparedOrAcceptedProposalNumber != -1) {
        anAcceptedState = proposal(aPreparedOrAcceptedProposalNumber);
    } else if (sendPrepardNumberIfNoAccept()) {
        PreparedOrAcceptedProposalNumber = 
            maxProposalNumberReceivedInPrepareOrAcceptRequest;
    }
    prepare(aPreparedOrAcceptedProposalNumber,  
            anAcceptedState, aProposalNumber, aProposal,  
            checkPrepareRequest(aProposalNumber, aProposal));
}

Each acceptor calculates and sends back last accepted value and its proposal number (or optionally last seen proposal number if no acceptance so far) and updates last seen proposal number
protected void prepare(float aLastPreparedOrAcceptedProposalNumber, StateType aLastAcceptedProposal, float aPreparedProposalNumber, StateType aProposal, ProposalFeedbackKind aFeedbackKind) {
    recordReceivedPrepareRequest(aPreparedProposalNumber, aProposal);
    if (false)
    {
        // we accepted this proposal before preparing for it
        aPreparedProposalNumber == aLastPreparedOrAcceptedProposalNumber
        // preparer has started the accept phase
        || !isPending(aPreparedProposalNumber))
        { return;
        }
    if (!isSuccess(aFeedbackKind)) {
        processPrepareRejection(aLastPreparedOrAcceptedProposalNumber, aLastAcceptedProposal, aPreparedProposalNumber, aFeedbackKind);
    } else {
        recordAndSendPrepareResponse(aLastPreparedOrAcceptedProposalNumber, aLastAcceptedProposal, aPreparedProposalNumber, aFeedbackKind);
    }
}

Each acceptor calculates and sends back last accepted value and its proposal number (or optionally last seen proposal number if no acceptance so far) and updates last seen proposal number.
public void prepared(float aPreparedOrAcceptedProposalNumber, StateType anAcceptedProposal, float aPreparedProposalNumber, ProposalFeedbackKind aFeedbackKind) {
    recordReceivedPreparedNotification(aPreparedOrAcceptedProposalNumber, anAcceptedProposal, aPreparedProposalNumber, aFeedbackKind);
    if (!isPending(aPreparedProposalNumber) || isPreparePhaseOver(aPreparedProposalNumber)) {
        return;
    }
    if (aFeedbackKind == ProposalFeedbackKind.CONCURRENCY_CONFLICT) {
        newProposalState(aPreparedProposalNumber, proposal(aPreparedProposalNumber), toProposalState(aPreparedProposalNumber, anAcceptedProposal, aFeedbackKind));
        return;
    }
    aggregatePreparedNotification(aPreparedOrAcceptedProposalNumber, anAcceptedProposal, aPreparedProposalNumber, aFeedbackKind);
}

Preparer abandons proposal if it learns about a higher number proposal
protected void aggregatePreparedNotification(
float anAcceptedProposalNumber, StateType anAcceptedProposal,
float aPreparedProposalNumber, ProposalFeedbackKind aFeedbackKind) {
    Boolean isSufficientPreparers = sufficientPeparers(
        getPrepareSynchrony(), aPreparedProposalNumber);
    if (isSufficientPreparers == null)
        return;
    setPreparePhaseOver(aPreparedProposalNumber);
    if (isSufficientPreparers) {
        startAcceptPhase(aPreparedProposalNumber,
            preparedProposal(aPreparedProposalNumber));
    } else {
        newProposalState( aPreparedProposalNumber,
            proposal(aPreparedProposalNumber),
            ProposalState.PROPOSAL_AGGREGATE_DENIAL);
        return;
    }
}
**Prepared Proposal**

```java
protected StateType preparedProposal(float aPreparedProposalNumber) {
    float aChosenProposalNumber =
        maxAcceptedProposalNumberReceivedInPreparedNotification <= 0 ?
            aPreparedProposalNumber :
            maxAcceptedProposalNumberReceivedInPreparedNotification;
    return proposal(aChosenProposalNumber);
}
```

Proposer sends its proposal and value if majority acceptors have not yet accepted any value.

Proposer (re) proposes with highest accept proposal number as its own value (which may also be a majority value in majority acceptors).
protected void startAcceptPhase(float aProposalNumber, StateType aProposal) {
    super.startAcceptPhase(aProposalNumber, aProposal);
}
**Accept Check**

```java
protected synchronized ProposalFeedbackKind checkAcceptRequest(float aProposalNumber, StateType aProposal) {
    if (!isNotPaxos()) {
        return super.checkAcceptRequest(aProposalNumber, aProposal);
    }
    return (isAcceptConcurrencyConflict(aProposalNumber, aProposal))?
        ProposalFeedbackKind.CONCURRENCY_CONFLICT:
        ProposalFeedbackKind.SUCCESS;
}
```

No application-specific check in this phase under Paxos
protected boolean isAcceptConcurrencyConflict (float aProposalNumber, StateType aState) {
    return isPrepareConcurrencyConflict(aProposalNumber, aState);
}

protected void recordReceivedAcceptRequest(float aProposalNumber, StateType aProposal) {
    super.recordReceivedAcceptRequest(aProposalNumber, aProposal);
    maxProposalNumberReceivedInPrepareOrAcceptRequest = Math.max(
        maxProposalNumberReceivedInPrepareOrAcceptRequest, aProposalNumber);
}

Rejection reasons same for prepare and accept

Last seen proposal number updated by both
SYNCHRONOUS ACCEPT PHASE
Basic (Non Centralized) Paxos Accept Phase
DO NOT BROADCAST LEARNED INFORMATION

Super class code

```java
protected void sendLearnNotification(float aProposalNumber, 
StateType aProposal, ProposalFeedbackKind anAgreement) {
    localLearn(aProposalNumber, aProposal, anAgreement);
    sendLearnNotificationToOthers(aProposalNumber, aProposal, anAgreement);
}
```

Paxos-specific code

```java
protected void sendLearnNotificationToOthers(float aProposalNumber, 
StateType aProposal, ProposalFeedbackKind anAgreement) {
    if (!isNotPaxos()) {
        super.sendLearnNotificationToOthers(aProposalNumber, 
                                               aProposal, anAgreement);
    }
}
```

Learned values are not broadcast in Basic Paxos
protected void sendAcceptedNotification(float aProposalNumber, StateType aProposal, ProposalFeedbackKind aFeedbackKind) {
    if (isNotPaxos()) {
        super.sendAcceptedNotification(aProposalNumber, aProposal, aFeedbackKind);
        return;
    }
    sendAcceptedNotificationToLearners(aProposalNumber, aProposal, aFeedbackKind);
}
**EXAMPLE SCENARIOS**

![Diagram showing three scenarios]

1. `proposeMeaning(MEANING_1);`

3. `proposeMeaning(MEANING_2);`
Paxos Algorithm Properties

Proposer sends its proposal and value if majority acceptors have not yet accepted any value.

Each acceptor accepts a proposal if its proposal number is higher than what it has seen so far.
CASE 1 PROPERTIES

1 and 3 finds no previous acceptance and 1 finds no previous prepare

1’s accept is rejected by majority, 3’s accept goes through
**CASE 1: IN-ORDER PREPARES BEFORE ACCEPTS**

Breaks in `startPreparePhase()` and `startAcceptPhase()`

1-Prepare

3-Prepare

1-Accept

3-Accept

Resume 1.startPreparePhase()

Resume 3.startPreparePhase()

Resume 1.startAcceptPhase()

Resume 3.startAcceptPhase()
Making proposal of: 42
I*** (ProposalMade) Meaning,1.0001=42
I*** (ProposalWaitStarted) Meaning,1.0001=42
I*** (ProposalPrepareRequestReceived) 1--> Meaning,1.0001=42
I*** (ProposalPreparedNotificationSent) Meaning,1.0001<--(-1.0,null) == SUCCESS
I*** (ProposalPreparedNotificationReceived) 1--> Meaning,1.0001<--(-1.0,null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning,1.0001,42:1|1|3|3?1.5-->null
I*** (ProposalPreparedNotificationReceived) 2--> Meaning,1.0001<--(-1.0,null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning,1.0001,42:2|2|3|3?1.5-->true
I*** (ProposalAcceptRequestSent) Meaning,1.0001=42
I*** (ProposalPreparedNotificationReceived) 3--> Meaning,1.0001<--(-1.0,null) == SUCCESS
I*** (ProposalPreparedRequestReceived) 3--> Meaning,1.0003=29
I*** (ProposalPreparedNotificationSent) Meaning,1.0003<--(1.0001,null) == SUCCESS
I*** (ProposalAcceptedNotificationReceived) 2--> Meaning,1.0001=42:CONCURRENCY_CONFLICT
I*** (SufficientAgreementsChecked) Meaning,1.0001,42:0|1|3|3?1.5-->null
I*** (ProposalAcceptRequestReceived) 1--> Meaning,1.0001=42
I*** (ProposalAcceptedNotificationSent) Meaning,1.0001=42-->CONCURRENCY_CONFLICT
I*** (ProposalAcceptedNotificationReceived) 1--> Meaning,1.0001=42:CONCURRENCY_CONFLICT
I*** (SufficientAgreementsChecked) Meaning,1.0001,42:0|2|3|3?1.5-->false
I*** (ProposalStateChanged) Meaning,1.0001=42-->PROPOSAL AGGREGATE_DENIAL
I*** (ProposalWaitEnded) Meaning,1.0001=42-->PROPOSAL AGGREGATE_DENIAL
I*** (ProposalAcceptRequestReceived) 3--> Meaning,1.0003=29
I*** (ProposalAcceptedNotificationReceived) Meaning,1.0003=29-->SUCCESS
I*** (ProposalAcceptedNotificationReceived) 2--> Meaning,1.0003=29:SUCCESS
I*** (SufficientAgreementsChecked) Meaning,1.0003,29:1|1|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 1--> Meaning,1.0003=29:SUCCESS
I*** (SufficientAgreementsChecked) Meaning,1.0003,29:2|2|3|3?1.5-->true
I*** (ProposalStateChanged) Meaning,1.0003=29-->PROPOSAL_CONSENSUS

Meaning of Life: 29
I*** (ProposalAcceptedNotificationReceived) 3--> Meaning,1.0001=42:CONCURRENCY_CONFLICT
I*** (ProposalAcceptedNotificationReceived) 3--> Meaning,1.0003=29:SUCCESS
Paxos Algorithm Properties

Preparer abandons proposal if it learns about a higher number proposal
Case 2 Properties

1 finds a higher proposal in prepare phase
CASE 2: REVERSE-ORDER Prepares Before Accepts

Breaks in startPreparePhase() and startAcceptPhase() in 1 and 3

<table>
<thead>
<tr>
<th>3-Prepare</th>
<th>Resume 3.startPreparePhase()</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-Prepare</td>
<td>Resume 1.startPreparePhase()</td>
</tr>
<tr>
<td>1-Accept</td>
<td>Resume 1.startAcceptPhase()</td>
</tr>
<tr>
<td>3-Accept</td>
<td>Resume 3.startAcceptPhase()</td>
</tr>
</tbody>
</table>
Case 2: Abandoning Lower Proposal in Prepare Phase

1-Prepare → 3-Prepare → 3-Accept

Proposer abandons proposal if it learns about a higher number proposal.
Paxos Algorithm Property

Proposer (re) proposes majority value learned from the prepare phase as its own value.
CASE 3 PROPERTY

3 finds majority acceptances from 1 in prepare phase
Case 3: 1 Before 3

Breaks in startPreparePhase() and startAcceptPhase() in 1 and 3

1-Prepare
1-Accept
3-Prepare
3-Accept

Resume 1.startPreparePhase()
Resume 1.startAcceptPhase()
Resume 3.startPreparePhase()
Resume 3.startAcceptPhase()
CASE 3

Connected to all members
Making proposal of: 42
I*** (ProposalMade)  Meaning, 1.0001=42
I*** (ProposalWaitStarted)  Meaning, 1.0001=42
I*** (ProposalPrepareRequestReceived)  1 --> Meaning, 1.0001=42
I*** (ProposalPreparedNotificationSent)  Meaning, 1.0001<--(-1.0,null) == SUCCESS
I*** (ProposalPreparedNotificationReceived)  1 --> Meaning, 1.0001<--(-1.0,null) == SUCCESS
I*** (SufficientAgreementsChecked)  Meaning, 1.0001, 42:1|1|3|3?1.5 -->null
I*** (ProposalPreparedNotificationReceived)  2 --> Meaning, 1.0001<--(-1.0,null) == SUCCESS
I*** (SufficientAgreementsChecked)  Meaning, 1.0001, 42:2|2|3|3?1.5 -->true
I*** (ProposalAcceptRequestSent)  Meaning, 1.0001=42
I*** (ProposalPreparedNotificationReceived)  3 --> Meaning, 1.0001<--(-1.0,null) == SUCCESS
I*** (ProposalAcceptedNotificationReceived)  2 --> Meaning, 1.0001=42:SUCCESS
I*** (SufficientAgreementsChecked)  Meaning, 1.0001, 42:1|1|3|3?1.5 -->null
I*** (ProposalAcceptRequestReceived)  1 --> Meaning, 1.0001=42
I*** (ProposalAcceptedNotificationSent)  Meaning, 1.0001=42 -->SUCCESS
I*** (ProposalAcceptedNotificationReceived)  3 --> Meaning, 1.0001=42:SUCCESS
I*** (SufficientAgreementsChecked)  Meaning, 1.0001, 42:2|2|3|3?1.5 -->true
I*** (ProposalStateChanged)  Meaning, 1.0001=42 --> PROPOSAL_CONSENSUS
Meaning of Life: 42
I*** (ProposalWaitEnded)  Meaning, 1.0001=42 --> PROPOSAL_CONSENSUS
I*** (ProposalAcceptedNotificationReceived)  1 --> Meaning, 1.0001=42:SUCCESS
I*** (ProposalPrepareRequestReceived)  3 --> Meaning, 1.0003=29
LATE MESSAGES AND RE-PROPOSALS

--------- --- -------- ------------- ---
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0001=42: SUCCESS
I*** (ProposalPrepareRequestReceived) 3 --> Meaning, 1.0003=29
I*** (ProposalPreparedNotificationSent) Meaning, 1.0003<--(1.0001, 42) == SUCCESS
I*** (ProposalAcceptRequestReceived) 3 --> Meaning, 1.0003=42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0003=42-->SUCCESS
I*** (ProposalAcceptedNotificationReceived) 2 --> Meaning, 1.0003=42: SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 42:1 | 1 | 3 | 3?1.5 --> null
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0003=42: SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 42:2 | 2 | 3 | 3?1.5 --> true
I*** (ProposalStateChanged) Meaning, 1.0003=42 --> PROPOSAL_CONSENSUS
Meaning of Life: 42
I*** (ProposalAcceptedNotificationReceived) 3 --> Meaning, 1.0003=42: SUCCESS

3-Prepare-3-Accept
Why Re-propose with Majority?

- Proposer (re) proposes majority value learned from the prepare phase as its own value.

- A prepare phase can prevent some nodes from accepting the previous (to be) majority value.

- If some of the nodes in current majority die, some learner nodes may not get consensus value even though majority of nodes are alive and can converge to a value.

- Want consensus value to propagate to all acceptors for fault tolerance.
**Case 4: Properties**

A prepare phase prevents node 3 from accepting the previous majority value of 1’s proposal accepted by 1 and 2.

If node 2 dies, node 3 will not get consensus value even though majority of nodes (1 and 3) are alive and can converge to a value.

Want consensus value to propagate to all acceptors for fault tolerance.
Break in startAcceptPhase() at start, sendAcceptedFrom2() before sending to 3, sendPrepareFrom3() before sending to 1,2

1-Prepare-*

3-Prepare-3

1-Accept-*

1-Accepted-*

2-Accepted-1,2

3-Prepare-1,2

Kill 2

3-Accept-*

Resume 1.startAcceptPhase

Resume 3.sendPrepareFrom3()

2’s accepted notification will not reach 3 but will reach 1 and 2

So 3 should re-propose
CASE 4: 2 LEARNS BEFORE DYING

Connected to all members
I*** (ProposalPrepareRequestReceived) 1--> Meaning, 1.0001=42
I*** (ProposalPreparedNotificationSent) Meaning, 1.0001<--(-1.0, null) == SUCCESS
I*** (ProposalAcceptRequestReceived) 1--> Meaning, 1.0001=42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0001=42-->SUCCESS
I*** (ProposalAcceptedNotificationReceived) 1--> Meaning, 1.0001=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001,42:1|1|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 3--> Meaning, 1.0001=42:CONCURRENCY_CONFLICT
I*** (SufficientAgreementsChecked) Meaning, 1.0001,42:1|2|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 2--> Meaning, 1.0001=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001,42:2|3|3|3?1.5-->true
I*** (ProposalStateChanged) Meaning, 1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life: 42
I*** (ProposalPrepareRequestReceived) 3--> Meaning, 1.0003=29
I*** (ProposalPreparedNotificationSent) Meaning, 1.0003<--(1.0001,42) == SUCCESS
Making proposal of: 42
I*** (ProposalMade) Meaning, 1.0001=42
I*** (ProposalWaitStarted) Meaning, 1.0001=42
I*** (ProposalPrepareRequestReceived) 1 --> Meaning, 1.0001=42
I*** (ProposalPreparedNotificationSent) Meaning, 1.0001<--(-1.0, null) == SUCCESS
I*** (ProposalPreparedNotificationReceived) 1 --> Meaning, 1.0001<--(-1.0, null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42:1|1|3|3?1.5-->null
I*** (ProposalPreparedNotificationReceived) 2 --> Meaning, 1.0001<--(-1.0, null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42:2|2|3|3?1.5-->true
I*** (ProposalAcceptRequestSent) Meaning, 1.0001=42
I*** (ProposalPreparedNotificationReceived) 3 --> Meaning, 1.0001<--(-1.0, null) == SUCCESS
I*** (ProposalAcceptRequestReceived) 1 --> Meaning, 1.0001=42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0001=42 --> SUCCESS
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0001=42: SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42:1|1|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 3 -->
I*** (SufficientAgreementsChecked) Meaning, 1.0001=42: SUCCESS
I*** (ProposalStateChanged) Meaning, 1.0001=42 --> PROPOSAL_CONSENSUS
Meaning of Life: 42
I*** (ProposalWaitEnded) Meaning, 1.0001=42 --> PROPOSAL
I*** (ProposalPrepareRequestReceived) 3 --> Meaning, 1.0003=29
I*** (ProposalPreparedNotificationSent) Meaning, 1.0003<--(1.0001, 42) == SUCCESS
AReadCommand for java.nio.channels.SocketChannel[connected local=/152.2.130.185:60079 rem 2 has left the session
AReadCommand for java.nio.channels.SocketChannel[connected local=/152.2.130.185:7001 remo
I*** Received left message : Host: DEWAN1 Name: 2 ID: 7002
I*** (ProposalAcceptRequestReceived) 3 --> Meaning, 1.0003=42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0003=42 --> SUCCESS
I*** (ProposalAcceptedNotificationReceived) 3 --> Meaning, 1.0003=42: SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 42:1|1|2|2?1.0-->true
I*** (ProposalStateChanged) Meaning, 1.0003=42 --> PROPOSAL_CONSENSUS
Meaning of Life: 42
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0003=42: SUCCESS

Two read failures as each process creates a connection to the other
CASE 4: 3 LEARNS ONCE

Making proposal of: 29
I*** (ProposalMade) Meaning, 1.0003 = 29
I*** (ProposalWaitStarted) Meaning, 1.0003 = 29
I*** (ProposalPrepareRequestReceived) 1 --> Meaning, 1.0001 = 42
I*** (ProposalPreparedNotificationSent) Meaning, 1.0001 <= (-1.0, null) == SUCCESS
I*** (ProposalPrepareRequestReceived) 3 --> Meaning, 1.0003 = 29
I*** (ProposalPreparedNotificationSent) Meaning, 1.0003 <= (-1.0, null) == SUCCESS
I*** (ProposalPreparedNotificationReceived) 3 --> Meaning, 1.0003 <= (1.0001, null) == SUCCESS

I*** (SufficientAgreementsChecked) Meaning, 1.0003, 29: 1 | 2 | 3 | 3?1.5 --> null
I*** (ProposalAcceptRequestReceived) 1 --> Meaning, 1.0001 = 42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0001 = 42 --> CONCURRENCY_CONFLICT
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0001 = 42: SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42: 1 | 2 | 3 | 3?1.5 --> null
I*** (ProposalAcceptedNotificationReceived) 3 --> Meaning, 1.0001 = 42: CONCURRENCY_CONFLICT
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42: 1 | 2 | 3 | 3?1.5 --> null
I*** (ProposalPreparedNotificationReceived) 1 --> Meaning, 1.0003 <= (1.0001, 42) == SUCCESS

I*** (SufficientAgreementsChecked) Meaning, 1.0003, 29: 2 | 2 | 3 | 3?1.5 --> true
AReadCommand for java.nio.channels.SocketChannel[connected local=/152.2.130.185:60089 remote
2 has left the session
AReadCommand for java.nio.channels.SocketChannel[connected local=/152.2.130.185:7003 remote
I*** (ProposalAcceptRequestSent) Meaning, 1.0003 = 42

I*** (ProposalPreparedNotificationReceived) 2 --> Meaning, 1.0003 <= (1.0001, 42) == SUCCESS
I*** Received left message: Host: DEWAN1 Name: 2 ID: 7002
I*** (ProposalAcceptRequestReceived) 3 --> Meaning, 1.0003 = 42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0003 = 42 --> SUCCESS
I*** (ProposalAcceptedNotificationReceived) 3 --> Meaning, 1.0003 = 42: SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 42: 1 | 1 | 2 | 2?1.0 --> true
I*** (ProposalStateChanged) Meaning, 1.0003 = 42 --> PROPOSAL_CONSENSUS
Meaning of Life: 42
I*** (ProposalWaitEnded) Meaning, 1.0003 = 29 --> PROPOSAL_CONSENSUS
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0003 = 42: SUCCESS
Why Re-propose with Minority

Proposer (re) proposes value of highest accept proposal number as its own value.

The value with highest proposal number may or may not be or become majority value.

If it does become majority value then prepare phase may have locked some nodes from accepting it.
CASE 5: PROPERTIES

Node 1’s minority proposal will become majority value

3’s prepare phase has locked itself from accepting it
CASE 5: CONSTRUCTION

Break in sendPrepareFrom3() before sending to 1,2, start of startAcceptPhase()

1-Prepare-*
3-Prepare-3
1-Accept-1,2,3
3-Prepare-1

3’s prepare sees acceptance from 1 and rejection from 3 though 2 has accepted to create majority value

Resume 1.startAcceptPhase()
Step Over sendPrepareFrom3()

3-Accept-*
3-Prepare-2

So 3 should re propose 1’s value

Resume 3.sendAcceptFrom3() and sendPrepareFrom3()
CASE 5: EXECUTION

Making proposal of:29
I*** (ProposalMade) Meaning, 1.0003=29
I*** (ProposalWaitStarted) Meaning, 1.0003=29
I*** (ProposalPrepareRequestReceived) 1--> Meaning, 1.0001=42
I*** (ProposalPreparedNotificationSent) Meaning, 1.0001<--(-1.0, null) == SUCCESS
I*** (ProposalPrepareRequestReceived) 3--> Meaning, 1.0003=29
I*** (ProposalPreparedNotificationSent) Meaning, 1.0003<--(1.0001, null) == SUCCESS
I*** (ProposalPreparedNotificationReceived) 3--> Meaning, 1.0003<--(1.0001, null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 29:1|1|3|3?1.5-->null
I*** (ProposalAcceptRequestReceived) 1--> Meaning, 1.0001=42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0001=42-->CONCURRENCY_CONFLICT
I*** (ProposalAcceptedNotificationReceived) 1--> Meaning, 1.0001=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42:1|1|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 3--> Meaning, 1.0001=42:CONCURRENCY_CONFLICT
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42:1|2|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 2--> Meaning, 1.0001=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001, 42:2|3|3?1.5-->true
I*** (ProposalStateChanged) Meaning, 1.0001=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I*** (ProposalPreparedNotificationReceived) 1--> Meaning, 1.0003<--(1.0001, 42) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 29:2|2|3|3?1.5-->true
I*** (ProposalAcceptRequestSent) Meaning, 1.0003=42
I*** (ProposalAcceptedNotificationSent) 2--> Meaning, 1.0003=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 42:1|1|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 2--> Meaning, 1.0003=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 42:2|2|3|3?1.5-->true
I*** (ProposalStateChanged) Meaning, 1.0003=42-->PROPOSAL_CONSENSUS
Meaning of Life:42
I*** (ProposalWaitEnded) Meaning, 1.0003=29-->PROPOSAL_CONSENSUS
I*** (ProposalAcceptedNotificationReceived) 1--> Meaning, 1.0003=42:SUCCESS
I*** (ProposalAcceptedNotificationReceived) 1--> Meaning, 1.0003=42:SUCCESS
CASE 6: PROPERTIES

Node 1’s minority proposal will not become majority value

But 3 does not know that at start of accept phase, so re-proposes
CONSTRUCTION

Break in sendPrepareFrom3() before sending to 1, 2, and start of sendAcceptFrom1() and sendAcceptFrom3()

1-Prepare-*
3-Prepare-3
1-Accept-1
3-Prepare-*
3-Accept-*
1-Accept-2, 3

Step Over
1.sendAcceptFrom1()

Resume
sendPrepareFrom3()

Resume
3.sendAcceptFrom3()

Resume sendAcceptFrom1()

3’s prepare sees acceptance from 1 and rejection from 3

3 re propose 1’s value,

Though 2 will reject 1’s value as 3 does not know in what order the acceptances will reach 2
Making proposal of:29
I*** (ProposalMade) Meaning, 1.0003=29
I*** (ProposalWaitStarted) Meaning, 1.0003=29
I*** (ProposalPrepareRequestReceived) 1 --> Meaning, 1.0001=42
I*** (ProposalPrepareNotificationSent) Meaning, 1.0001<=(-1.0,null) == SUCCESS
I*** (ProposalPrepareRequestReceived) 3 --> Meaning, 1.0003=29
I*** (ProposalPrepareNotificationSent) Meaning, 1.0003<=(-1.0001,null) == SUCCESS
I*** (ProposalPrepareNotificationReceived) 3 --> Meaning, 1.0003<=(-1.0001,null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003,29:1|1|3<3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0001=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0001,42:1|1|3<3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0003<=(-1.0001,42) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003,29:2|2|3<3?1.5-->true
I*** (ProposalAcceptRequestSent) Meaning, 1.0003=42
I*** (ProposalAcceptedNotificationReceived) 2 --> Meaning, 1.0003=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003,42:1|1|3<3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0003=42:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003,42:2|2|3<3?1.5-->true
I*** (ProposalStateChanged) Meaning, 1.0003=42 --> PROPOSAL_CONSENSUS

Meaning of Life:42
I*** (ProposalWaitEnded) Meaning, 1.0003=29 --> PROPOSAL_CONSENSUS
I*** (ProposalAcceptedNotificationReceived) 2 --> Meaning, 1.0003=42:SUCCESS
I*** (ProposalAcceptedNotificationReceived) 1 --> Meaning, 1.0003=42:SUCCESS
I*** (ProposalAcceptRequestReceived) 1 --> Meaning, 1.0001=42
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0001=42 --> CONCURRENCY_CONFLICT
I*** (ProposalAcceptedNotificationReceived) 3 --> Meaning, 1.0001=42:CONCURRENCY_CONFLICT
I*** (SufficientAgreementsChecked) Meaning, 1.0001,42:1|2|3<3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 2 --> Meaning, 1.0001=42:CONCURRENCY_CONFLICT
I*** (SufficientAgreementsChecked) Meaning, 1.0001,42:1|3<3|3?1.5-->false
I*** (ProposalStateChanged) Meaning, 1.0001=42 --> PROPOSAL_AGGREGATE_DENIAL
Paxos Algorithm Property

Re-proposals can prevent any successful acceptance
1’s first acceptance prevented from 3’s first prepare

3’s first acceptance is prevented from 1’s second prepare

1’s second acceptance is prevented from 3’s second prepare

3’s second acceptance is prevented from 1’s third prepare
## Case 7: Paxos Livelock with Retries

<table>
<thead>
<tr>
<th>(resume 1.startPrepare()) 1^1-Prepare (Succeeds)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1^1-Accept Blocks</td>
<td>(resume 3.startAccept()) 3^1-Accept Unblocks and Fails (wait for retry)</td>
</tr>
<tr>
<td>(resume 3.startPrepare) 3^1-Prepare Succeeds</td>
<td>(resume 3.startPrepare) 3^2-Reprepare Succeeds</td>
</tr>
<tr>
<td>3^1-Accept Blocks</td>
<td>3^2-Accept Blocks</td>
</tr>
<tr>
<td>(resume 1.startAccept()) 1^1-Accept-Unblocks and Fails (wait for retry)</td>
<td>(resume 1.startAccept()) 1^2-Reprepare-Unblocks and Fails (wait for retry)</td>
</tr>
<tr>
<td>(resume 1.startPrepare) 1^2-Reprepare (Succeeds)</td>
<td>1^2-Reprepare Succeeds</td>
</tr>
<tr>
<td>1^2-Accept Blocks</td>
<td></td>
</tr>
</tbody>
</table>

Breaks in `startPreparePhase()` and `startAcceptPhase()` in 1 and 3
Case 7: 3-Prepare 1-Accept

1-Prepare Succeeds

1-Accept- Unblocks and Fails and 1 retries
CASE 7: 1-REPREPARE SUCCEEDS

1

I*** (ProposalMade) Meaning, 2.0001=42
I*** (ProposalWaitStarted) Meaning, 2.0001=42
I*** (ProposalPrepareRequestReceived) 1 --> Meaning, 2.0001=42
I*** (ProposalPreparedNotificationSent) Meaning, 2.0001<--(1.0003,null) == SUCCESS
I*** (ProposalPreparedNotificationReceived) 2 --> Meaning, 2.0001<--(1.0003,null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 2.0001,42:1|1|3|3?1.5-->null
I*** (ProposalPreparedNotificationReceived) 1 --> Meaning, 2.0001<--(1.0003,null) == SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 2.0001,42:2|2|3|3?1.5-->true

Timed out waiting for proposal 1.0001.
CASE 7: 3-(Re) ACCEPT AFTER 1-(Re-Re) PREPARE

3-Accept Fails
3-Reproposes
1-Reaccept Fails
CASE 8: UNCONTROLLED EXECUTION-1 (CONFLICT IN PREPARE PHASE)

Connected to all members
Making proposal of: 42
I*** (ProposalMade) Meaning, 1.0001=42
I*** (ProposalWaitStarted) Meaning, 1.0001=42
I*** (ProposalPrepareRequestReceived) 3--> Meaning, 1.0003=29
I*** (ProposalPreparedNotificationSent) Meaning, 1.0003<--(-1.0, null) == SUCCESS
I*** (ProposalPreparedNotificationReceived) 1--> Meaning, 1.0001=42
I*** (ProposalPreparedNotificationSent) Meaning, 1.0001<--(1.0003, null) == CONCURRENCY_CONFLICT
I*** (ProposalPreparedNotificationReceived) 3--> Meaning, 1.0001<--(1.0003, null) == CONCURRENCY_CONFLICT
I*** (ProposalStateChanged) Meaning, 1.0001=42-->PROPOSAL_CONCURRENT_OPERATION
I*** (ProposalWaitEnded) Meaning, 1.0001=42-->PROPOSAL_CONCURRENT_OPERATION
I*** (ProposalAcceptedRequestReceived) 3--> Meaning, 1.0003=29
I*** (ProposalAcceptedNotificationSent) Meaning, 1.0003=29-->SUCCESS
I*** (ProposalPreparedNotificationReceived) 1--> Meaning, 1.0001<--(1.0003, null) == CONCURRENCY_CONFLICT
I*** (ProposalPreparedNotificationReceived) 2--> Meaning, 1.0001<--(1.0003, null) == CONCURRENCY_CONFLICT
I*** (ProposalAcceptedNotificationReceived) 1--> Meaning, 1.0003=29:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 29: 1|1|3|3?1.5-->null
I*** (ProposalAcceptedNotificationReceived) 3--> Meaning, 1.0003=29:SUCCESS
I*** (SufficientAgreementsChecked) Meaning, 1.0003, 29: 2|2|3|3?1.5-->true
I*** (ProposalStateChanged) Meaning, 1.0003=29-->PROPOSAL_CONSENSUS
Meaning of Life: 29
I*** (ProposalAcceptedNotificationReceived) 2--> Meaning, 1.0003=29:SUCCESS
**Paxos vs. Centralized Synchronous**

- 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
A Consensus Mechanism Hierarchy

- **An Asynchronous Consensus Mechanism**
  - `learn();`
  - `addConsensusListener();`
  - `propose();`
  - `setAcceptSynchrony();`

- **A Centralizable Consensus Mechanism**
  - `remotePropose();`
  - `prepare();`
  - `prepared();`
  - `accept();`
  - `accepted();`

- **A Paxos Consensus Mechanism**
  - `preprepare();`
  - `prepare();`
  - `prepared();`

Local application operations

Selected application operations shown

Use most efficient scheme for needs

Internal distributed operations
public interface ConsensusCustomization {
public ConcurrencyKind getConcurrencyKind();
public void setConcurrencyKind(ConcurrencyKind consistencyStrength);
public ProposalFeedbackKind getProposalVetoKind();
public void setProposalVetoKind(ProposalFeedbackKind proposalRejectionKind);
public ReplicationSynchrony getAcceptSynchrony();
public void setAcceptSynchrony(ReplicationSynchrony consensusSynchrony);
public void setSendRejectionInformation(boolean newVal);
public boolean isSendRejectionNotification();
public boolean isAllSynchronous();
public void setAllowVeto(boolean newVal);

public ConsensusMemberSetKind getConsensusMemberSetKind();
public void setConsensusMemberSetKind(ConsensusMemberSetKind consensusMemberSet);
public boolean isValueSynchrony();
public void setValueSynchrony(boolean newVal);
public boolean isSendAcceptReplyForResolvedProposal();
public void setSendAcceptReplyForResolvedProposal(boolean newVal);
public boolean isClient();
public boolean isServer();
public boolean isCentralizedPropose();
}
SUMMARY

- Distributed → Replicated Systems
- RPC → Replicated Object
- Pure replication synchrony → Global clock, time bounds
- Replication Synchrony → Propose, Invalid State, Listeners, rather than pure get and set
- Replication Synchrony → Two phase algorithm
- Shared abstraction and algorithms for replication synchrony and distributed proposal rejection
- Degree of replication synchrony and vetoing based on set synchronized with/consulted by proposer
- Safety vs Progress
- Concurrent proposes in asynchronous and synchronous case can be handled with centralized coordinator
  - Centralized Synchronous is Two Phase Commit if invalidation step involves transaction set up and checks
- Coordinator switch-offs can be done using consensus protocol or zero phase id choice
- Central solution can lead to inconsistency even if majority alive
- 3-Phase Paxos supports consistency if majority alive
  - Centralization and timeouts for practical reasons
  - Atomic setting of proposal number and getting of state in acceptor
  - Convergence towards proposals with higher proposal numbers and re-proposing of acceptances
  - To update a value multiple times, multiple paxos mechanisms instantiated, once for each update
  - Centralization to achieve consensus about update sequence number