CS251 Fall 2022

(cs251.stanford.edu)

Fundamentals of Consensus

Ertem Nusret Tas

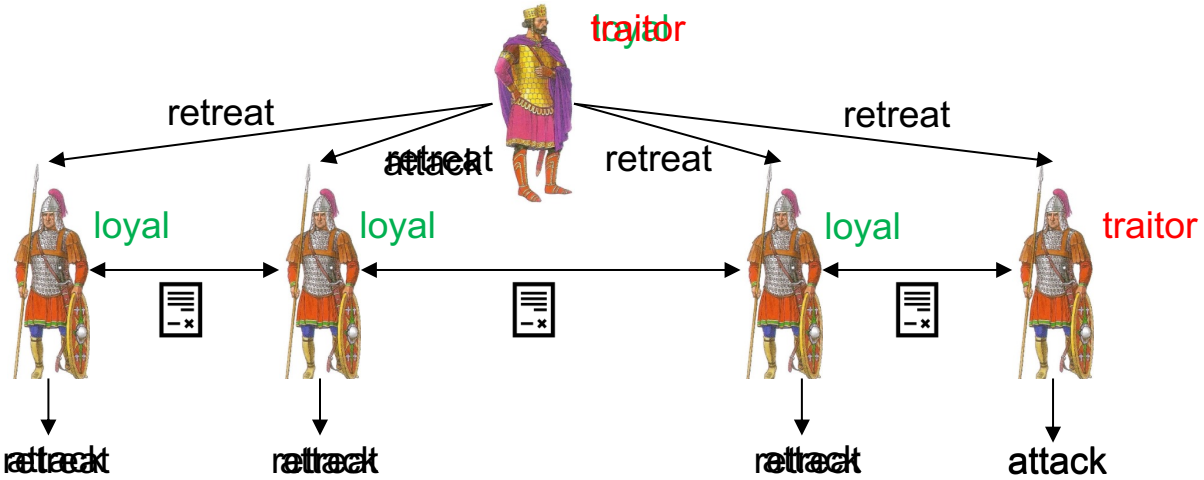
Byzantine Generals Problem

- Encapsulates the problem of reaching consensus.
- Introduced by Lamport et al. in 1982.
- Problem statement:
  - There are n generals (where n is fixed), one of which is the *commander*.
  - Some generals are *loyal*, and some of them can be *traitors* (including the commander).
  - The commander sends out an order that is either *attack* or *retreat* to each general.
  - If the commander is loyal, it sends the *same* order to all generals.
  - All generals take an action after some time.

The Byzantine Generals Problem (1982)

Byzantine Generals Problem

- Goal:
  - All loyal generals must take the **same** action.
  - If the commander is loyal, then all loyal generals must take the action **suggested by the commander**.



From Generals to Nodes

- Solution to the Byzantine Generals Problem is a *consensus protocol*.
- When modelling consensus protocols:
  - Generals → Nodes
  - Commander → Leader
  - Loyal → Honest, Traitor → Adversary
    - What can the adversarial nodes do?

Adversary

- *The* adversary can *corrupt* nodes, after which they are called **adversarial**.
- The adversary is said to induce:
  - **Crash faults if** the adversarial nodes do not send or receive any messages.
  - **Omission faults if** the adversarial nodes can selectively choose to drop or let through each messages sent or received.
  - **Byzantine faults (Byzantine adversary) if** the adversarial nodes can deviate from the protocol arbitrarily.

- Consensus protocols typically assume that the adversary cannot forge signatures. Why?

Adversary

- Adaptive vs. static adversary:
  - **Static adversary** corrupts the nodes of its choice before the protocol execution commences.
  - **Adaptive adversary** can dynamically corrupt the nodes during the protocol execution.
- We typically bound the adversary's power by assuming an upper bound on the number of nodes f that can ever be adversarial.
  - e.g., $f < n$, $f < \frac{n}{2}$, $f < \frac{n}{3}$, …

Communication 

- Nodes can send messages to each other within the protocol.
- Time proceeds in discrete *rounds*. We will hereafter use *seconds* as the minimal unit of time.
- We assume that the adversary *controls* the delivery of the messages subject to certain limits:
  - In a **synchronous network**, adversary must deliver any message sent by an honest node to its recipient(s) within Δ seconds. Here, Δ is a known bound.
  - In an **asynchronous network**, adversary can delay any message for an arbitrary, yet finite amount of time. However, it must eventually deliver every message sent by the honest nodes.

Communication

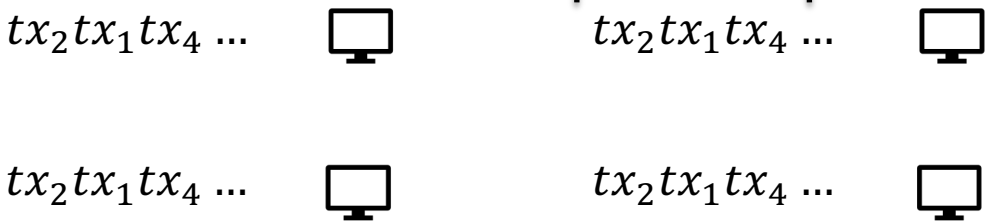
- **Partial synchrony**:
  - There exists a known $\Delta < \infty$ and an event called the **Global Stabilization Time (GST)** such that
    - **GST** eventually happens after some finite time that can be chosen arbitrarily by the adversary.
    - A message sent by an honest node at time t is delivered to its recipient(s) by time $\Delta + \max(\mathbf{GST}, t)$.
  - Network is '*asynchronous*' until **GST**, after which it behaves like a '*synchronous*' network.

State Machine Replication (SMR)

A Centralized Bank

Blockchain (State Machine Replication)

Log (Ledger): an ever-growing, linearly-ordered *sequence* of transactions.

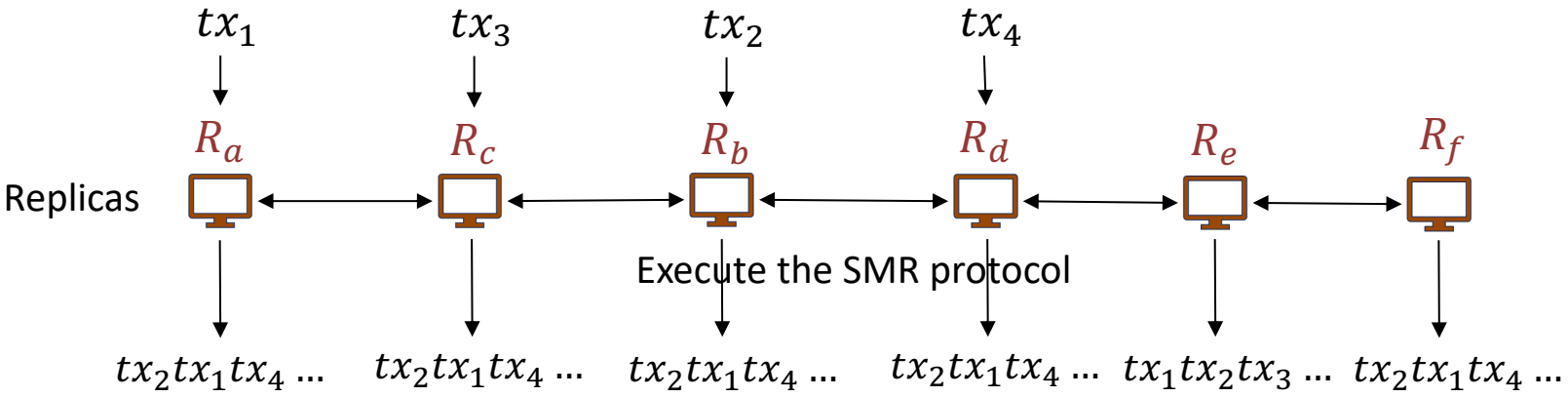


State Machine Replication (SMR)

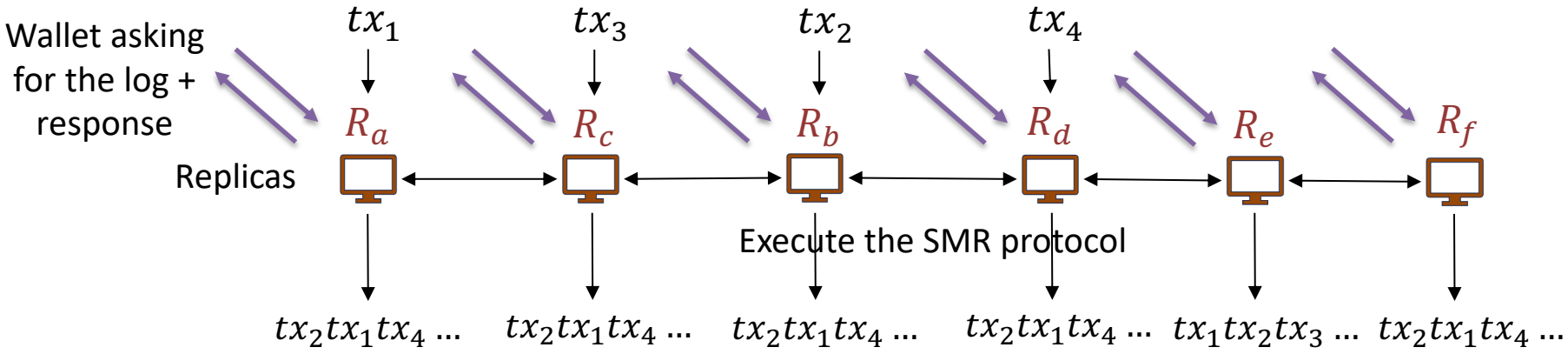
Two parties of SMR:

- *Replicas* receive transactions, execute the SMR protocol and determine the log.
- *Clients* are the learners: They communicate with the replicas to learn the log.

Goal of SMR is to ensure that the clients learn the *same* log.



State Machine Replication (SMR)



C_1 $LOG_t^1 = tx_2tx_1tx_4$ …

C_2 $LOG_t^2 = tx_2tx_1tx_4$ …

Wallets are an example of a client.

Wallets ask the replicas what the correct log is.

Clients (Wallets)

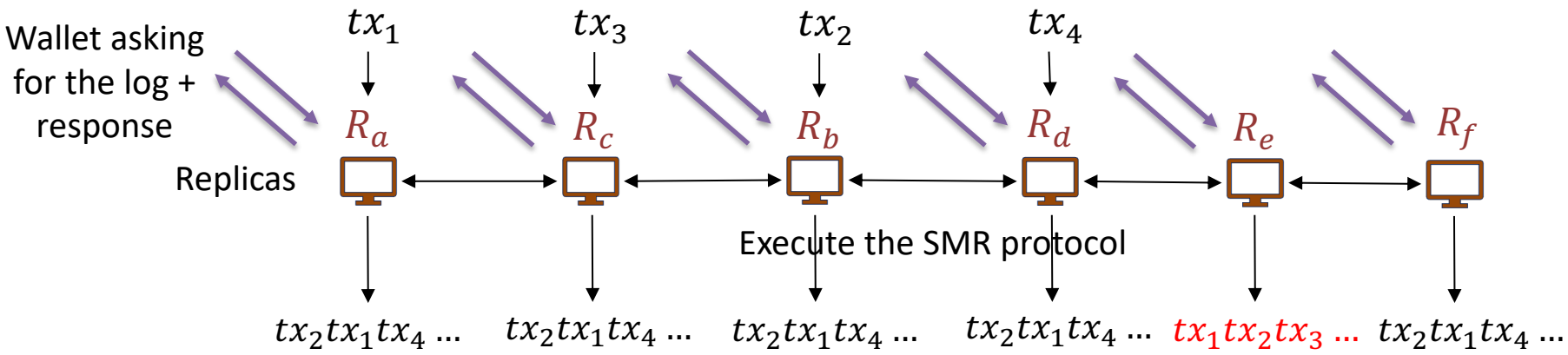
Clients (Wallets)

C_3 $LOG_t^3 = tx_2tx_1tx_4$ …

C_4 $LOG_t^4 = tx_2tx_1tx_4$ …

Wallets do not execute the SMR protocol and do not talk to each other.

State Machine Replication (SMR)



C_1 $LOG_t^1 = tx_2tx_1tx_4$ …

C_2 $LOG_t^2 = tx_2tx_1tx_4$ …

Clients (Wallets)

C_3 $LOG_t^3 = tx_2tx_1tx_4$ …

Clients (Wallets)

C_4 $LOG_t^4 = tx_2tx_1tx_4$ …

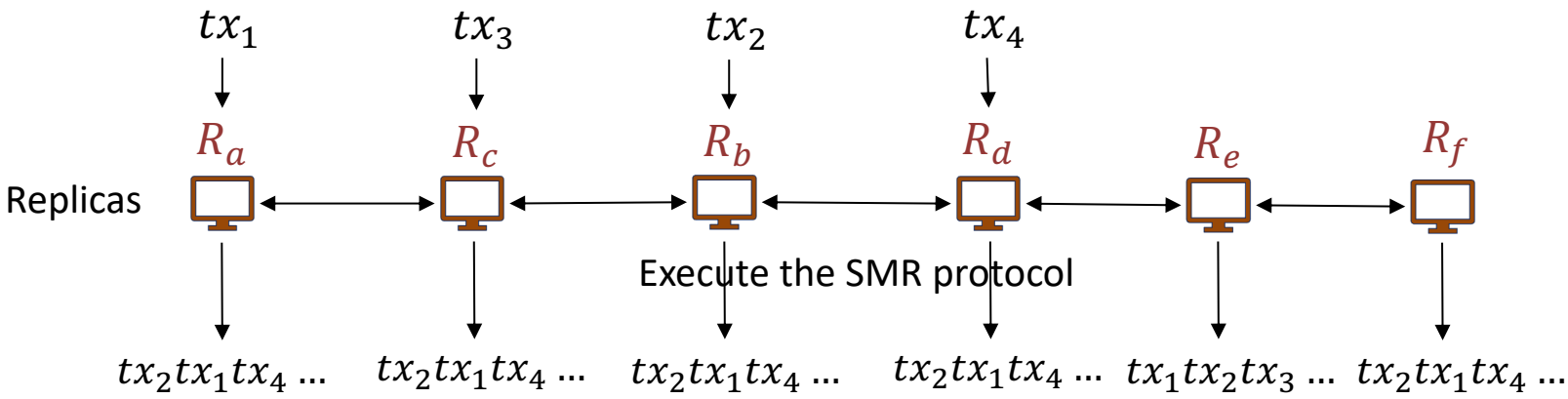
How does a wallet learn the correct log from the replicas?

- It asks the replicas what the correct log is.
- Wallet then accepts the answer given by majority of the replicas as its log.

Wallet learns the correct log if over half of the replicas are honest!

State Machine Replication (SMR)

Going forward, we will focus primarily on the replicas and the execution of the SMR by the replicas.



SMR vs. Byzantine Generals

- **Single shot vs. Multi-shot**
  - **Byzantine Generals Problem** is single shot consensus. Each node outputs a single value.
  - **State Machine Replication** is multi-shot. Each client *continuously* outputs a log, which is a sequence of transactions (values).
- **Who are the learners?**
  - In **Byzantine Generals Problem**, the nodes executing the protocol are the same as the nodes that output decision values.
  - In **State Machine Replication**, protocol is executed by the replicas, whereas the goal is for the clients to learn the log. Replicas must ensure that the clients learn the same log.

Security for SMR: Definitions

Concatenation ($A||B$):

- Suppose we have sequences $A = tx_1 tx_2$ and $\mathrm{B} = tx_3 tx_4$. What is $A||B$?

$$A||B = tx_1 tx_2 tx_3 tx_4$$

Prefix relation ($A \preccurlyeq B$): Sequence A is said to be a prefix of sequence B, if there exists a sequence C (that is potentially empty) such that $B = A||C$.

Suppose we have $A = tx_1 tx_2 tx_3 tx_4$, $B = tx_1 tx_2 tx_3$ and $D = tx_1 tx_2 tx_4$.

- Is B a prefix of A?
  - Yes
- Is D a prefix of A?
  - No

Security for SMR: Definitions

Two sequences A and B are consistent if either $A \preccurlyeq B$ is true or $B \preccurlyeq A$ is true or both statements are true.

Are these two logs consistent: $LOG^{Alice} = tx_1 tx_2 tx_3 tx_4$, $LOG^{Bob} = tx_1 tx_2 tx_3$?

- Yes!

What about $LOG^{Alice} = tx_1 tx_2 tx_3$, $LOG^{Bob} = tx_1 tx_2 tx_3 tx_4$?

- Yes!

What about $LOG^{Alice} = tx_1 tx_2$, $LOG^{Bob} = tx_1 tx_3$?

- No!

Security for SMR

Let LOG_t^i denote the log outputted by a client i at time t.

Then, a **secure** SMR protocol satisfies the following guarantees:

Safety (Consistency):

- For any two clients i and j, and times t and s: either $LOG_t^i \preccurlyeq LOG_s^j$ is true or $LOG_s^j \preccurlyeq LOG_t^i$ is true or both (Logs are consistent).

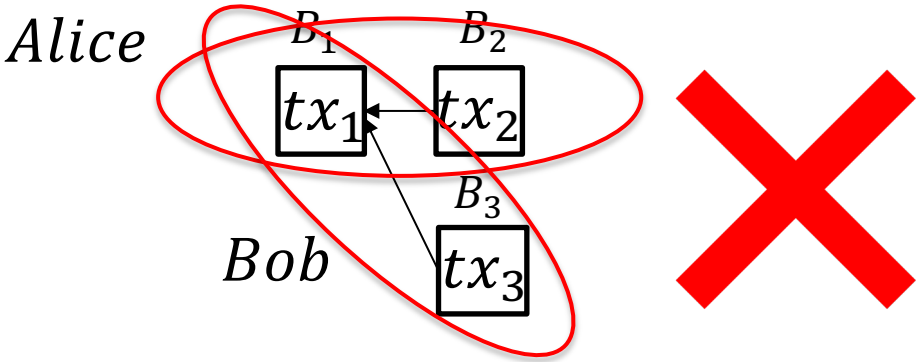
Liveness:

- If a transaction tx is input to an honest replica at some time t, then for all clients i, and times $s \geq t + T_{conf}$: $tx \in LOG_s^i$.

No double spend

No censorship

Security for SMR



Safety violation!!

Blockchain Protocols

Transactions are often batched into blocks to enhance throughput.

Let $\boldsymbol{ch_t^i}$ denote the chain *accepted* by a client i at time t.

Safety (Consistency):

- For any two clients i and j, and times t and s: either $\boldsymbol{ch_t^i} \preccurlyeq \boldsymbol{ch_s^j}$ is true or $\boldsymbol{ch_s^j} \preccurlyeq \boldsymbol{ch_t^i}$ is true or both (Chains are consistent).
- **Liveness:** If a transaction tx is input to an honest replica at some time t, then for all clients i, and times $s \geq t + T_{conf}$: $tx \in \boldsymbol{ch_s^i}$.

Desiderata

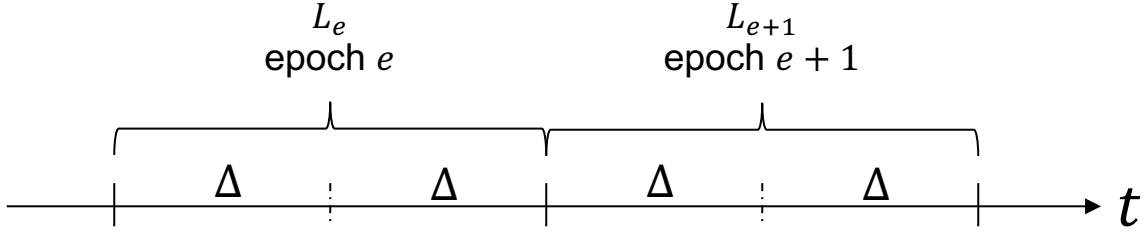
We want a SMR protocol that is secure under partial synchrony with the best possible resilience (for replicas) against a Byzantine adversary (that controls the corrupted replicas).

DLS (1988): If $f \geq n/3$, no SMR protocol can guarantee security under a partially synchronous network against a Byzantine adversary.

Given $f < \frac{n}{3}$, can we obtain SMR protocol that satisfies safety and liveness under partial synchrony against a Byzantine adversary?

First Attempt: Baby Streamlet

- Time proceeds in epochs of 2Δ seconds (see the figure below).
- There are n replicas, *fixed* before the protocol execution starts, and every replica knows the public keys of all other replicas (authenticated channels between replicas).
- Each epoch e is assigned a leader L_e by a public hash function $\mathrm{H}: \mathbb{Z}^+ \rightarrow [n]$.
- **Blocks:** Each block is associated with an epoch.



First Attempt: Baby Streamlet

At each epoch $e = 1, 2, \ldots$

- **Propose:** At the beginning of epoch e, L_e identifies the longest chain that it has seen so far and proposes a new block with epoch e extending the longest chain.

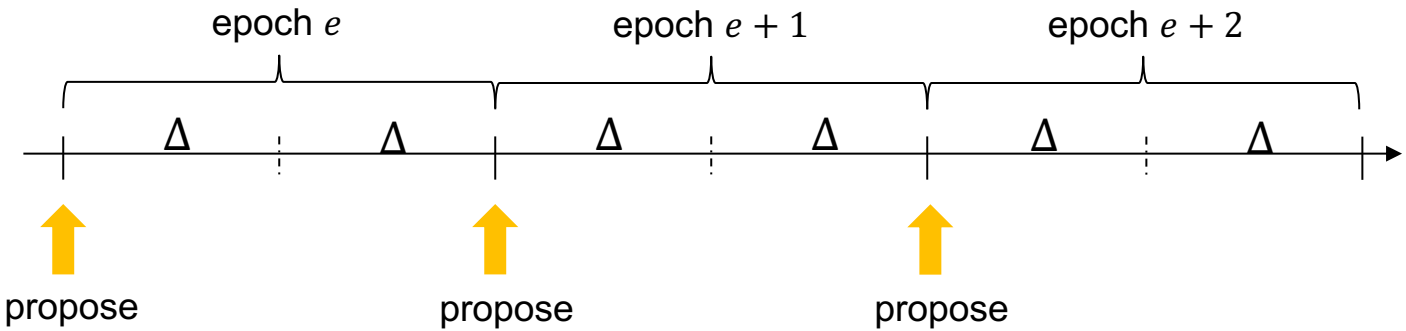
Finalization rule: A replica *finalizes* the block (and its prefix) at the tip of the longest chain. If there are multiple such blocks, it finalizes the one with the smaller epoch.

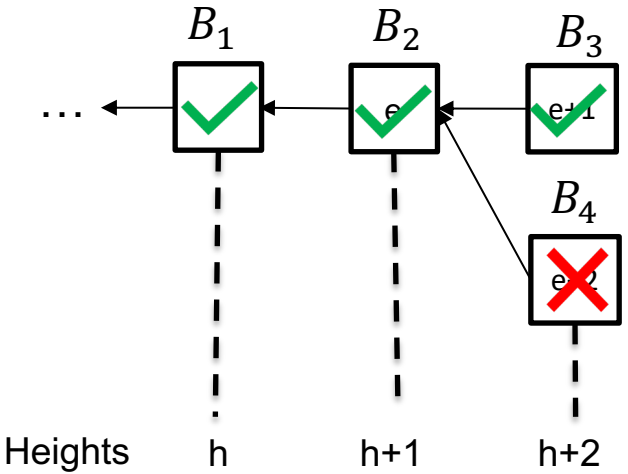
What does finalize mean?

It means that the replica accepts the finalized chain…

and decides the transactions in that chain as its log!

First Attempt: Baby Streamlet

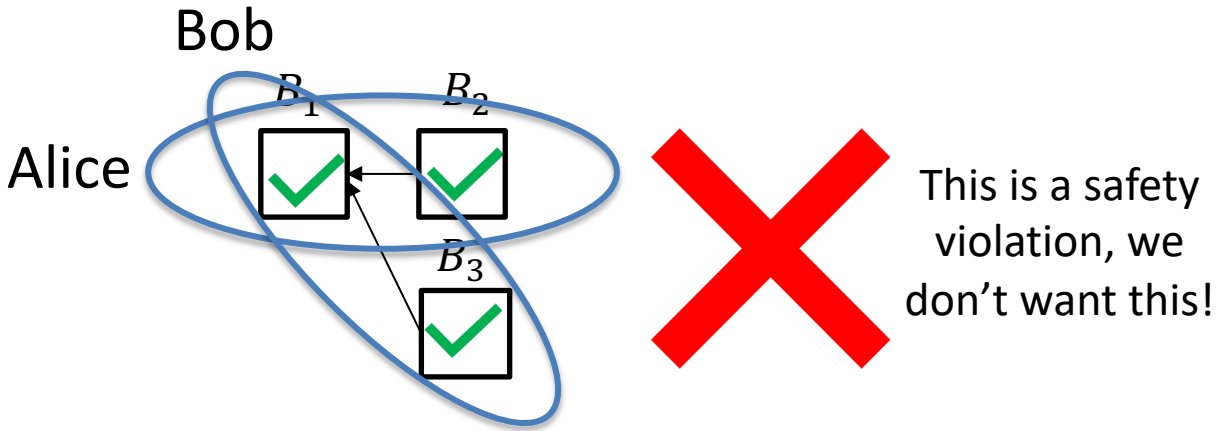




Security for Baby Streamlet

Given $f < \frac{n}{3}$, is Baby Streamlet secure under partial synchrony against a Byzantine adversary?

Let's try to prove safety first!



Can two blocks with the **same** epoch number be finalized at the same height?

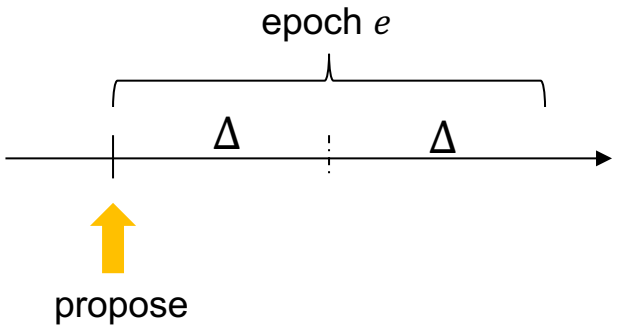
Security for Baby Streamlet

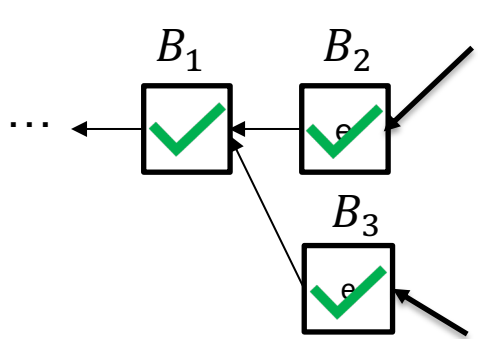
Can two blocks with the **same** epoch number be finalized at the same height?

Yes ☹

Why?

Leader can be adversarial, and network can be asynchronous (before GST)!





Adversary sends B_2 to Alice.
Bob **does not** see B_2 until GST.

Not safe!

Adversary sends B_3 to Bob.
Alice **does not** see B_3 until GST.

Second Attempt: Teen Streamlet

- Time proceeds in epochs of 2Δ seconds.
- There are n replicas, *fixed* before the protocol execution starts, and every replica knows the public keys of all other replicas (authenticated channels between replicas).
- Each epoch e is assigned a leader L_e by a public hash function $\mathrm{H}: \mathbb{Z}^+ \rightarrow [n]$.
- **Blocks:** Each block is associated with an epoch.
- **Votes:** A vote on a block by a replica is its signature on the block.
- **Notarization:** A block is said to be *notarized* in the view of a replica if the replica observes over $2n/3$ signatures from distinct replicas on the block.

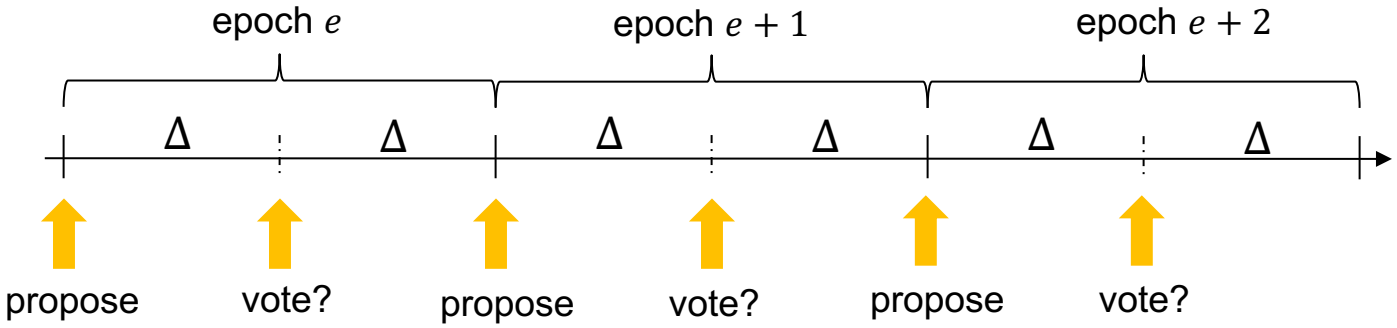
Second Attempt: Teen Streamlet

At each epoch $e = 1, 2, \ldots$

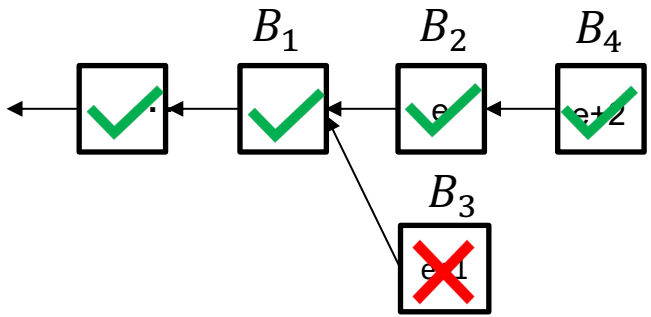
- **Propose:** At the beginning of epoch e, L_e identifies the longest notarized chain that it has seen so far and proposes a new block with epoch e extending the longest notarized chain.
- **Vote:** Δ seconds into epoch e, each honest replica votes for the *first* valid epoch e proposal from L_e that extends the longest notarized chain in its view. (In the absence of such a block, the replica does not vote.)

Finalization rule: A replica finalizes a block and its entire prefix chain once it observes the block to be notarized.

Second Attempt: Teen Streamlet



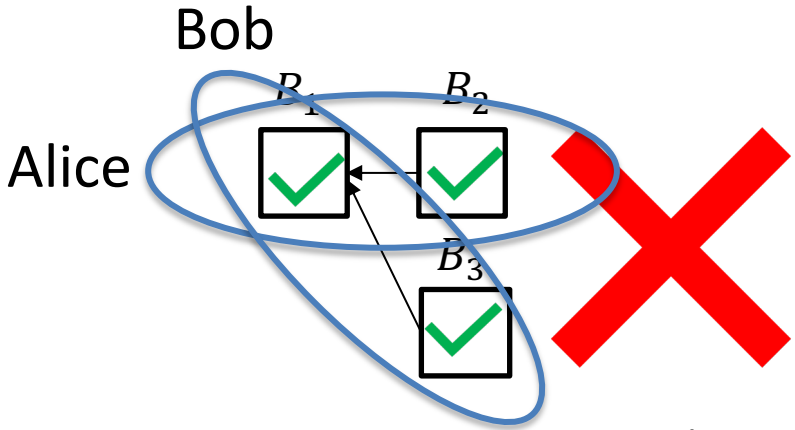
- Honest replicas ($> 2n/3$) vote for B_2.
- B_2 becomes notarized and **finalized**.
- No honest replica votes for B_3.
- With only adversarial votes ($< n/3$), B_3 cannot be notarized or finalized.
- Honest replicas ($> 2n/3$) vote for B_4.
- B_4 becomes notarized and **finalized**.



Security for Teen Streamlet

Given $f < \frac{n}{3}$, is Teen Streamlet secure under partial synchrony against a Byzantine adversary?

Let's again try to prove safety first!



Can two blocks with the **same** epoch number be notarized/finalized at the same height?

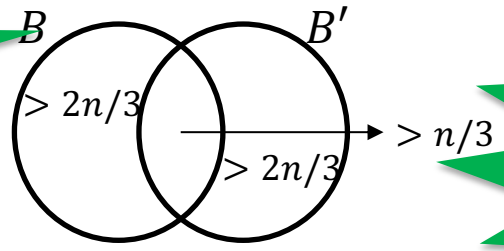
Security for Teen Streamlet

Lemma: Suppose $f < n/3$. Then, for each epoch e, there can be at most one notarized block with epoch number e in the view of any honest replica.

Proof: Towards contradiction, suppose there are two notarized blocks with epoch number e: B and B'.

Two blocks with the **same** epoch number cannot be notarized at the same height!

Number of votes on B and B':



Votes enable us to preserve safety even when the network is asynchronous.

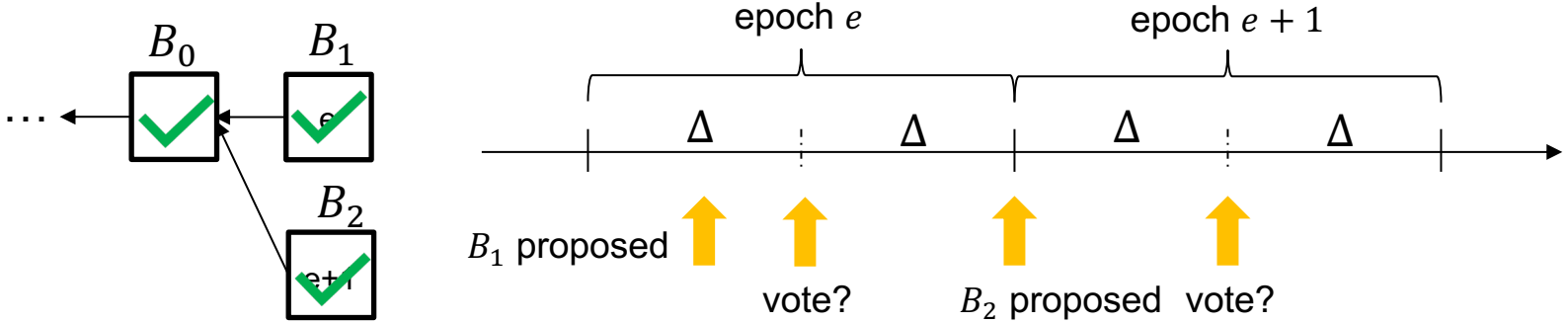
$\Rightarrow$ Over $n/3$ replicas voted for both B and B'.

Since $f < n/3$, one honest replica voted for both B and B'.

At each epoch, honest replicas vote for a *single* block with that epoch number. ✗

Security for Teen Streamlet

What about blocks with **different** epoch numbers? Can two blocks with different epoch numbers be notarized at the same height?



1. Suppose there are 100 replicas. Votes by 67 replicas are needed for a block to be notarized.
2. Block B_1 is proposed by an adversarial replica and shown to only 66 honest replicas.
3. 66 honest replicas vote for B_1, not enough to notarize it.
4. Block B_2 is proposed by an honest replica on the tip of the longest notarized chain.
5. Every honest replica votes for B_2, thus it is notarized.
6. Finally, an adversarial replica votes for B_1, making it notarized (**late vote**!!).

Not safe!

Security for Teen Streamlet

- We have not investigated whether Baby and Teen Streamlet satisfy liveness.
  - They might indeed satisfy liveness.
- However, we have shown that they do not satisfy safety,…

  …which already implies that they are not secure!
- A protocol that satisfies liveness but not safety is not a useful protocol.
- It will be vulnerable to double spend attacks!
- To prove security of a protocol, one should prove that the protocol satisfies *both* safety and liveness!

Final Attempt: Streamlet*

At each epoch $e = 1, 2, \ldots$

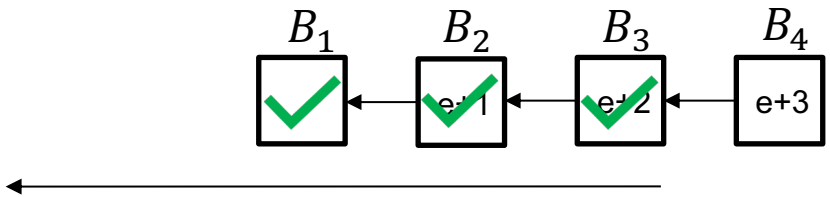
- **Propose:** At the beginning of epoch e, L_e identifies the longest notarized chain that it has seen so far and proposes a new block extending the longest notarized chain. (If there are multiple longest notarized chains, ties are broken *adversarially*.)
- **Vote:** Δ seconds into epoch e, each honest *replica* votes for the first valid epoch e proposal from L_e that extends one of the longest notarized chains in its view. (In the absence of such a block, the *replica* does not vote.)

Finalization rule: …

Streamlet: Textbook Streamlined Blockchains (2020)

Final Attempt: Streamlet*

Finalization rule: Upon seeing three adjacent blocks in a notarized chain with consecutive epoch numbers, a *client* finalizes the second of the three blocks, and its entire prefix chain.



Streamlet: Textbook Streamlined Blockchains (2020)

Security Theorem for Streamlet*

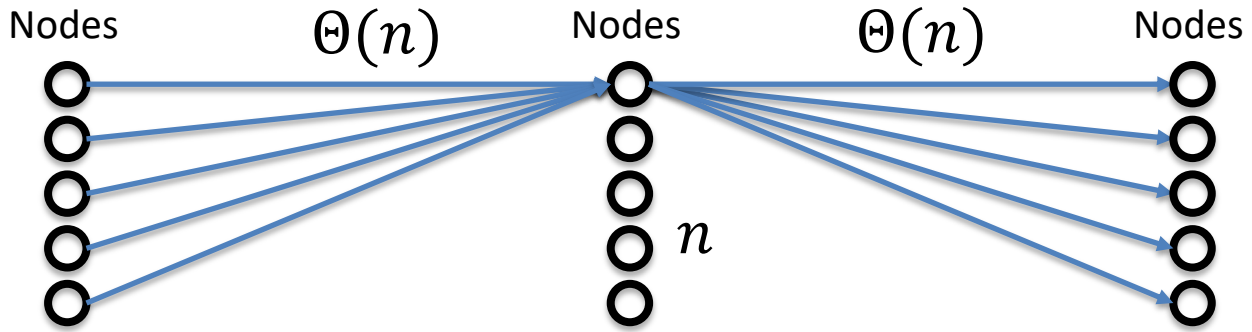
Theorem: Given $f < \frac{n}{3}$, Streamlet is secure under partial synchrony against a Byzantine adversary.

Is Streamlet the Endgame?

- Streamlet's requirement that epoch times are synchronized across replicas is too strong.
- Communication complexity of Streamlet is high: $\Theta(n^3)$ messages per block.

Why?

In Streamlet, every honest replica ($\Theta(n)$ in total) relays the votes it receives from every other replica (($\Theta(n)$) votes in total) to every other replica ($\Theta(n)$ in total)!



Is Streamlet the Endgame?

Do more practical protocols exist?

Yes!

Example: HotStuff

HotStuff achieves $\Theta(n)$ message complexity per block.

In HotStuff, there is no requirement for epochs to occur in perfect synchrony.

HotStuff: BFT Consensus in the Lens of Blockchain (2019)

END OF LECTURE

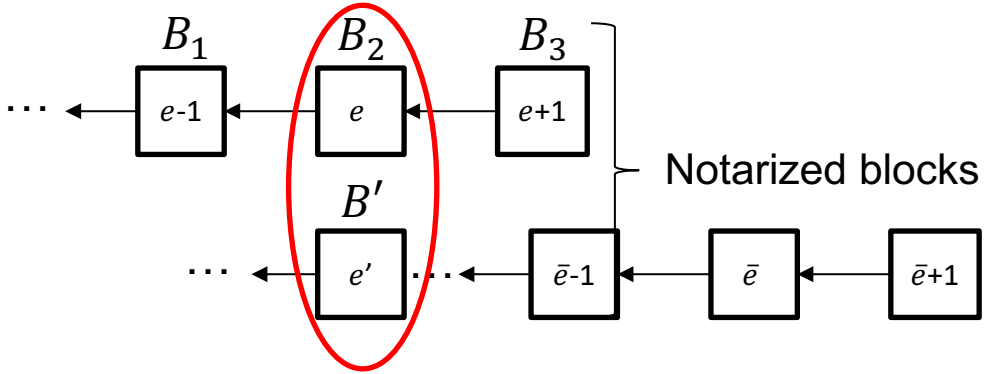
Next lecture: Consensus in the Internet Setting

Optional Slides

Slides going forward is optional material and present a simplified security proof for Streamlet*.

Security Proof: Safety (Optional)

Towards contradiction, suppose two conflicting blocks are finalized at epochs e and e'.



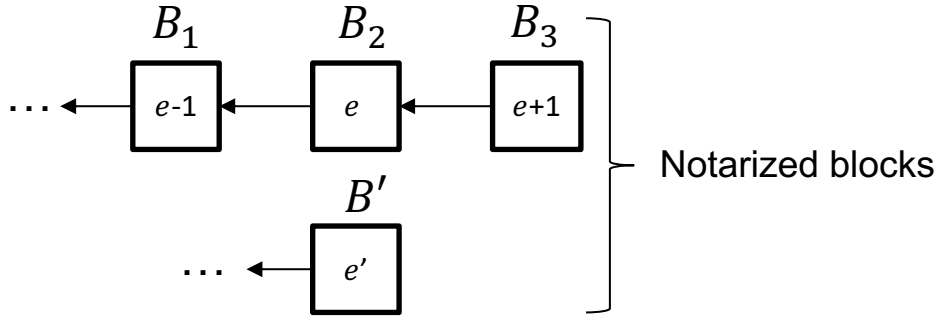
$e = e'$?

Because of the Lemma

Security Proof: Safety (Optional)

Towards contradiction, suppose two conflicting blocks are finalized at epochs e and e'.

Voting rule: vote for the proposal if it extends the longest notarized chain in view



$e < e'$? ✗

By the Lemma, $e + 1 < e'$.

Since B_3 is notarized, $> n/3$ honest replicas (called this set S) voted for it at epoch e+1.

Replicas in the set S saw B_2 as notarized at epoch e+1.

Replicas in the set S do not vote for B'. (Voting rule!!)

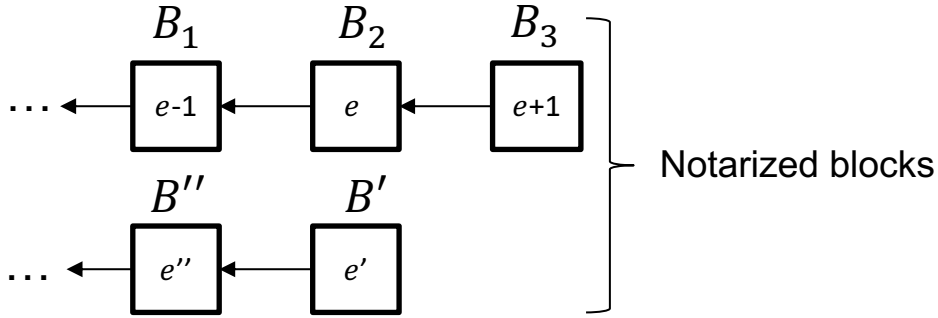
Since $|S| > n/3$, B' cannot gather $> 2n/3$ votes.

B' cannot get notarized.

Security Proof: Safety (Optional)

Voting rule: vote for the proposal if it extends the longest notarized chain in view

Towards contradiction, suppose two conflicting blocks are finalized at epochs e and e'.



$e' < e$?

By the Lemma, $e' < e - 1$.
Since B' is notarized, $> n/3$ honest replicas (called this set S) voted for it at epoch e'.
Replicas in the set S saw B'' as notarized at epoch e'.
Replicas in the set S do not vote for B_1. (Voting rule!!)
Since $|S| > n/3$, B_1 cannot gather $> 2n/3$ votes.
B_1 cannot get notarized.

Security Proof: Liveness (Optional)

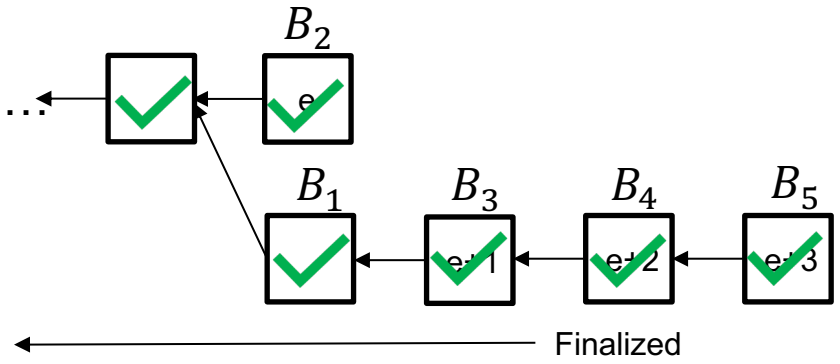
If there are 4 consecutive slots, e.g., $e, e+1, e+2, e+3$ with honest leaders after GST, at the end of these slots, every client finalizes a new block proposed by an honest replica.

- 1 slot to undo adversary's actions.
- 3 slots to finalize a new block.

An epoch leader is honest with probability at least 2/3.

What is the expected latency for Streamlet?

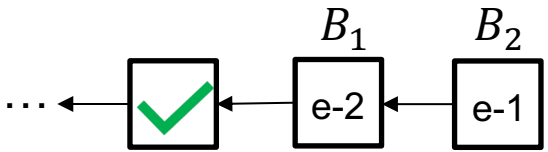
Security Proof: Liveness (Optional)



1. Block B_2 is proposed by an honest replica; there are 66 votes on B_1 in honest view (recall the Baby Streamlet attack).
2. Adversary releases 1 vote on B_1 to 1 an honest replica (call this R), making the total number of votes on B_1 equal to 67 in R's view.
3. As a result, B_1 is now notarized in R's view, and B_2 is no longer on the longest notarized chain in R's view. R does not vote for it, and B_2 is not notarized as it gathers only 66 votes.
4. Next, B_3 is proposed.
5. Adversary releases 1 vote on B_2 to all honest replicas, making it notarized.

Security Proof: Liveness (Optional)

There cannot be two ‘dangling’ consecutive blocks that are later notarized.



Notarized at or after epoch e?

Not possible!