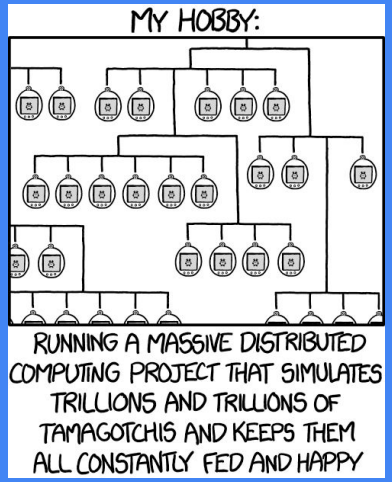
Section 2: Lab 2 Intro (ViewService)

CSE 452 Spring 2022



Lab 1 Questions

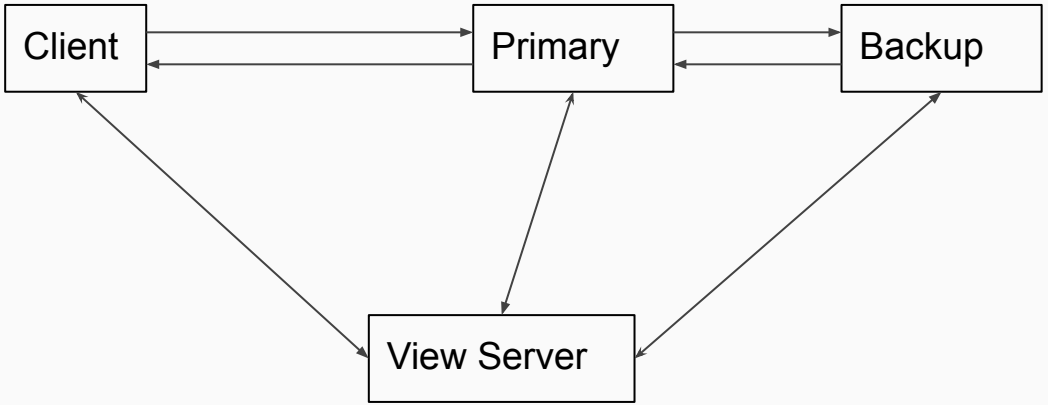
Use `make submit.tar.gz` and turn in via gradescope

If you haven’t already...

Add your partner to your GitLab repo specified by following these steps:

- Going to the designated GitLab repo that y’all decided to work on
- Going to Members
- Looking up your partner’s NETID
- Setting their role to Maintainer/Developer
- Click “Invite”

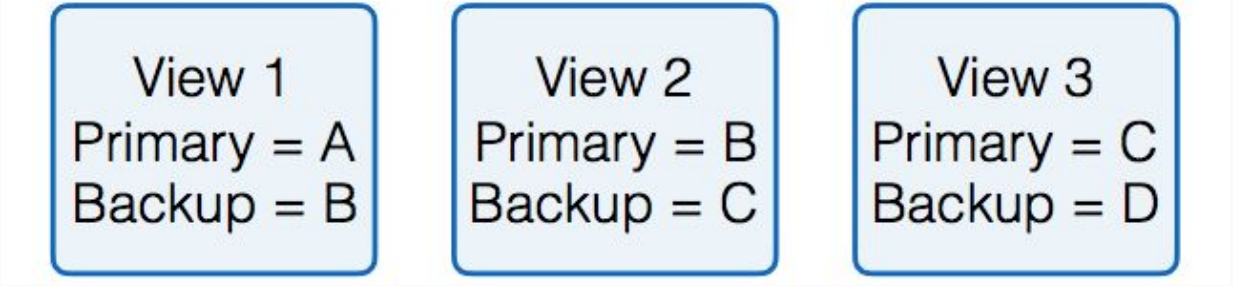
General Flow



Who are the players?

- ViewServer
- Primary
- Backup
- Other servers waiting in reserve…
- Clients

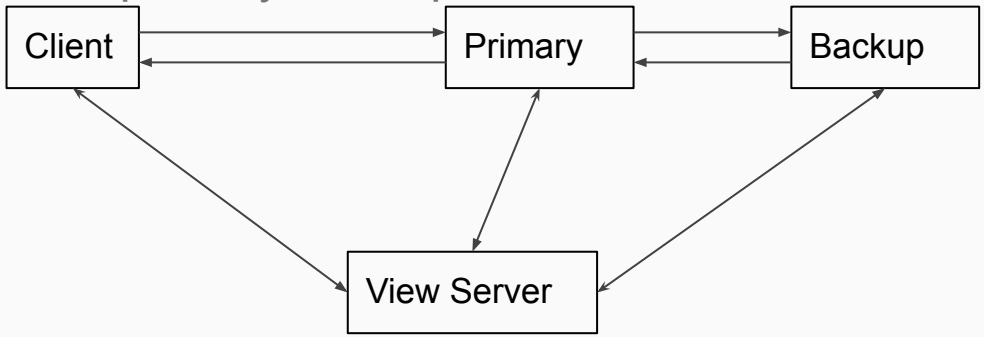
What's a view?




```
@Data
class View implements Serializable {
    private final int viewNum;
    private final Address primary, backup;
}
```

View Server

- Controls who is primary, who is backup, called a View
- System goes through sequence of views (increasing view number)
- Servers and clients contact the view server to learn identity of primary and backup
- Transitions between views ONLY when the current primary has acknowledged the current view (avoid split-brain)
- View server needs to “know” when primary/backup fail
- Single point of failure :(



Pings and Server failure

Heartbeat messages : Ping RPCs. Informs the View Server:

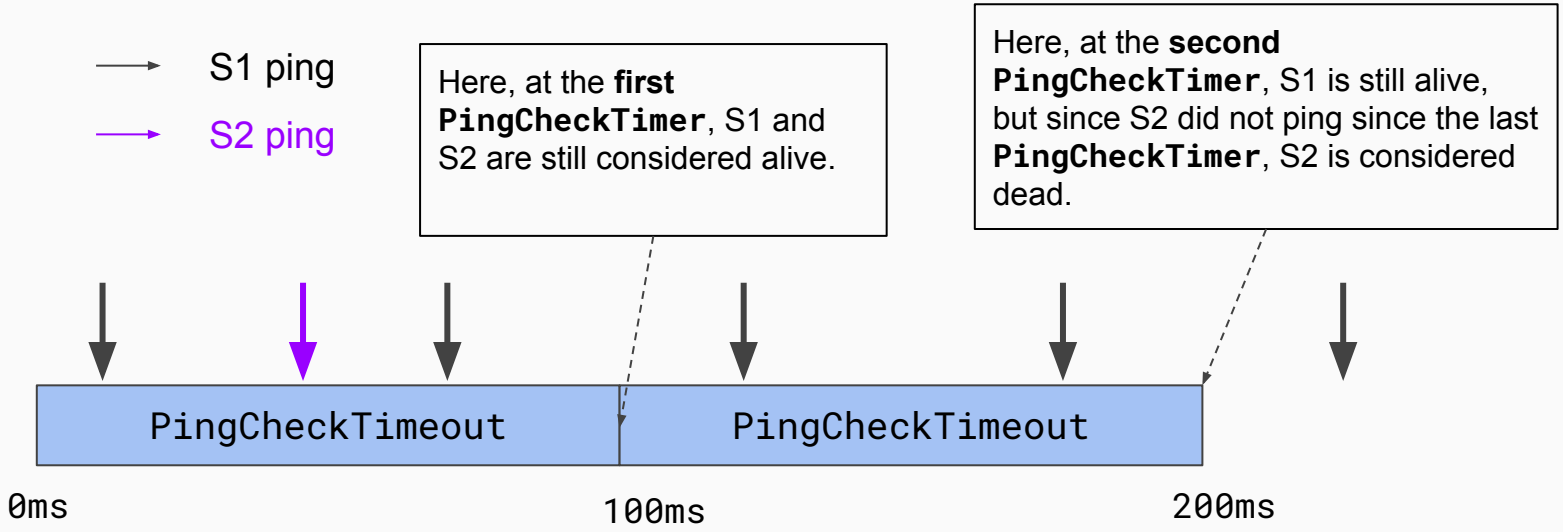
1) confirmation that the server is alive
2) the most recent view the server knows (why?)

Also lets the view server inform the server of the current view

Detect failure by absence of pings:

- Not received ping from server for some amount of time
  - For lab 2: Do **NOT** store timestamps on the view server! Needs to be deterministic for search tests.

Explaining PingCheckTimeout



"If the ViewServer doesn't receive a Ping from a server in-between two consecutive PingCheckTimers, it should consider the server to be dead"

The first timeout in which it's alive counts as one of the two!

When can a view transition occur? (1)

- First view is (STARTUP_VIEWNUM): {null, null}
- The first ping of some server (server A) should result in transition of startup view to INITIAL_VIEWNUM (should be {primary=A, null})
- View INITIAL_VIEWNUM+1 should be {primary=A, backup=B} if there is a backup (server B) available
- Primary acknowledges first non-null view (INITIAL_VIEWNUM) with its own ping
  - What if a server has pinged since?
    - Should be added as backup when the primary acknowledges (In other words, transition to a new view)

When can a view transition occur? (2)

From start-up view (special case):

- Can transition to initial view

Any other view (general case):

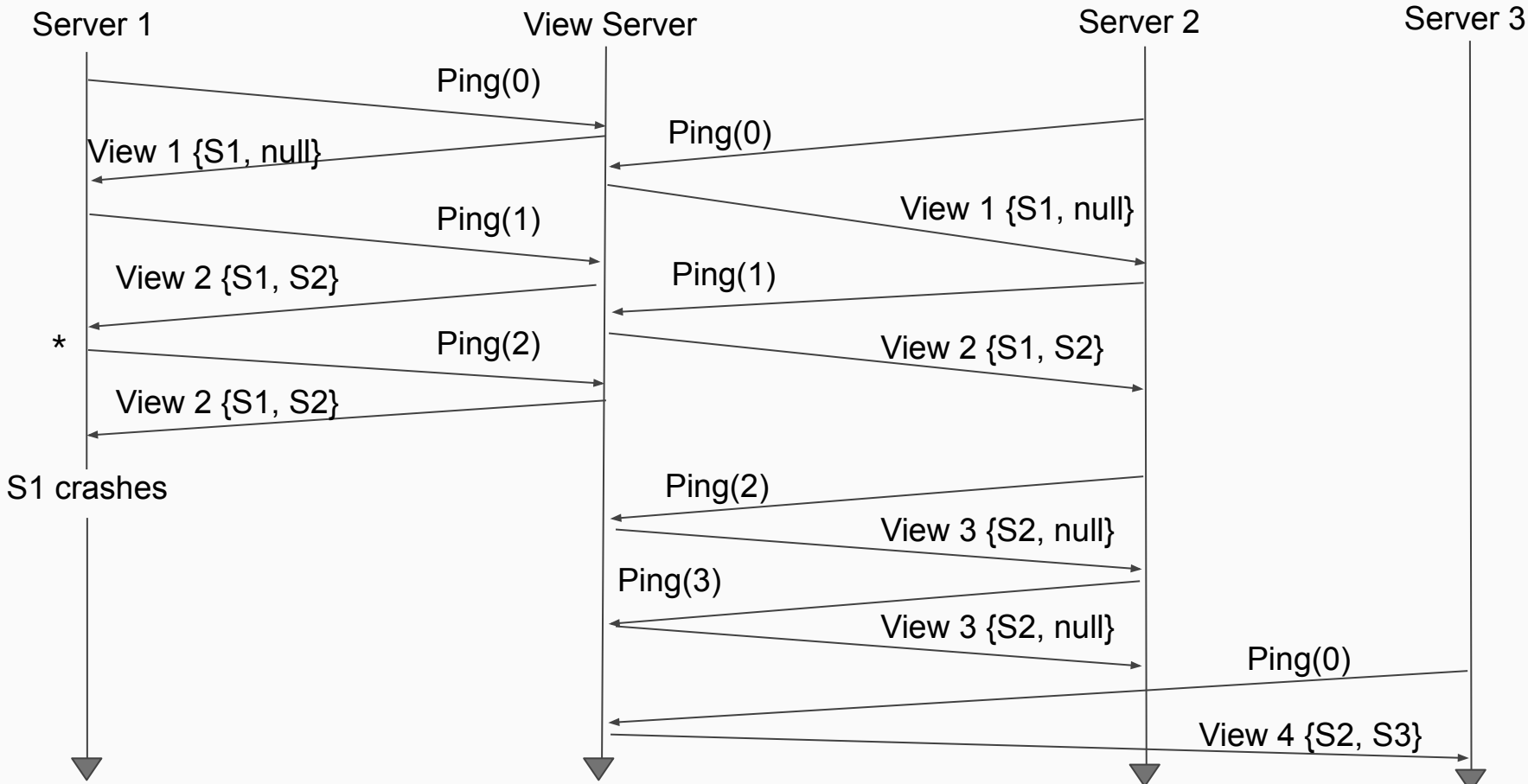
- The current view must have been acknowledged by the primary before the ViewServer can transition to the next view
  - If the primary has not acknowledged the current view yet, the ViewServer may not move onto another View

View Transition Timeouts

- **Only move to a new view (i + 1) if the primary of view (i) has acknowledged view (i)!**
- What happens if the primary fails? (assume current view has been ack'd)
  -
- What happens if backup fails?
  -
- What happens if the primary fails with no backup?
  -
- What happens if both primary and backup fail?
  -

Example Call Flow Diagram

* S1 sends application state to S2 and gets an ack back before Ping(2), acknowledging the view



Primary and Backup

- Client asks the View server for the latest view.
- Only the Primary responds to the client.
- Can a non-primary server get a client request? What should it do?
- Primary should pass requests to the backup and receive an ack before executing and responding to the client. What are some issues that can happen?
- What needs to be done when primary has a new backup?
  - Transfer state to backup
    - TIP: Send entire Application in a new message type
  - Ignore requests until state transfer complete

Remember the rules!

1. Primary in view i+1 must have been backup or primary in view i
2. Primary must wait for backup to accept/execute each op before doing op and replying to client
3. Backup must accept forwarded requests only if view is correct
4. Non-primary must reject client requests
5. Every operation must be before or after state transfer

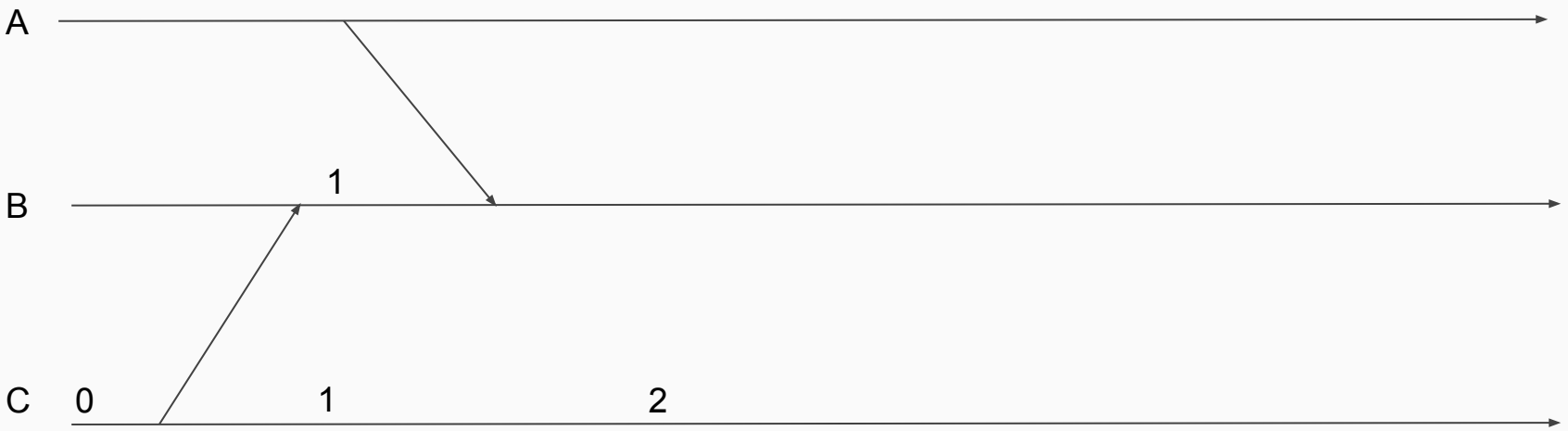
Exercise

Suppose we have three nodes, A, B, C that send messages to each other and each one keeps local log of its events.

Log consistency: if something happened on A or B that could affect the state at some point in C's log, we **must** include it if we include the event in C's log.

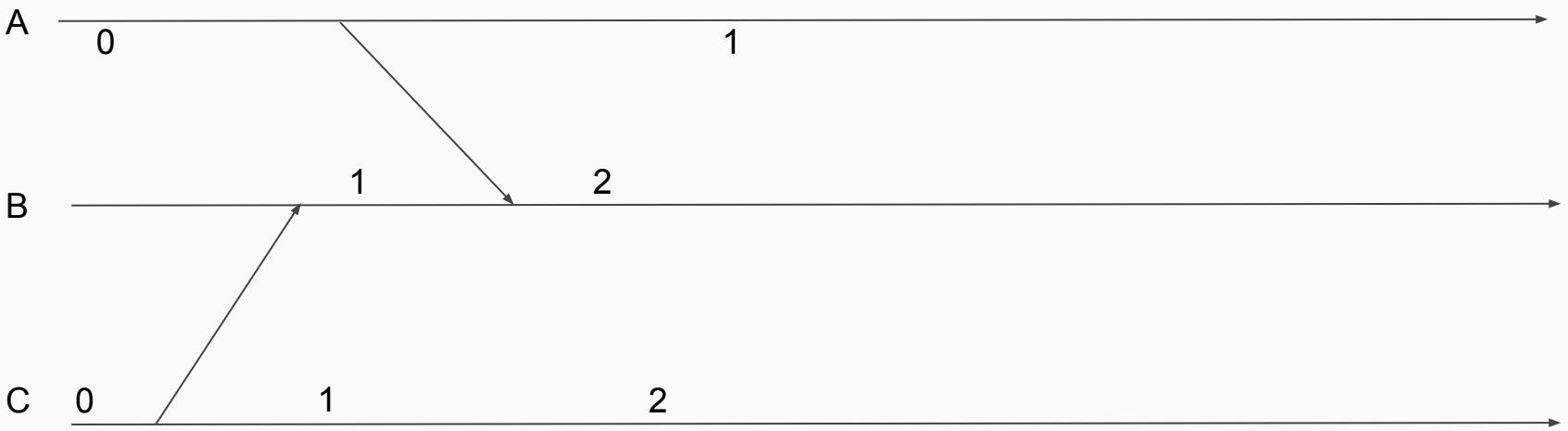
We do not have access to a sufficiently accurate real-time clock (otherwise, trivial - include all events in A and B that occur before C's event).

Example



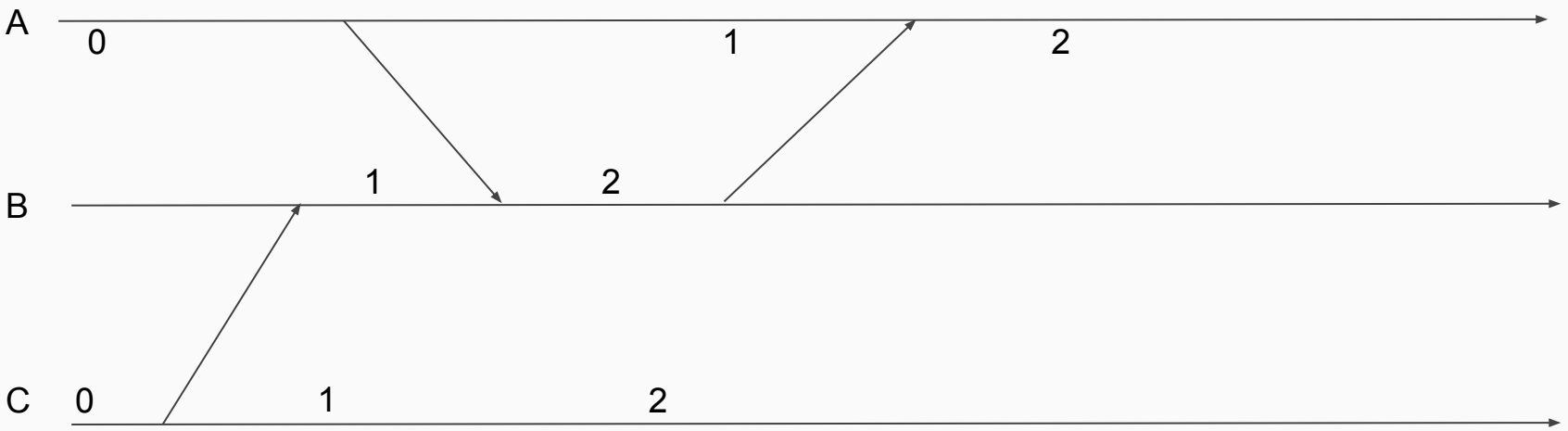
At B:1, what parts of C's log **must** we include to be consistent? What **can** we include?

Example



At B:2, what parts of C's log **must** we include to be consistent? What parts of A's log?

Example



If we include B:1 and not B:2, what parts of A's log must we **exclude** to be consistent?