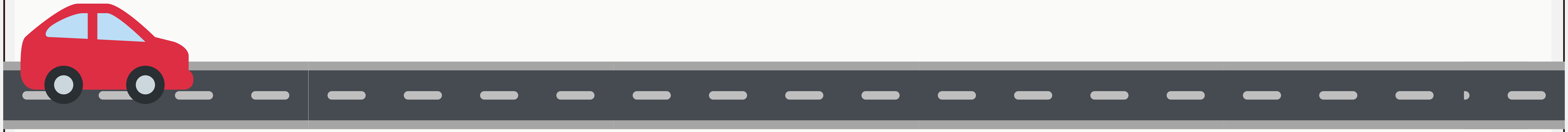
AUTOBAHN: SEAMLESS HIGH SPEED BFT

Neil Giridharan, Florian Suri-Payer, Ittai Abraham, Lorenzo Alvisi, Natacha Crooks

Presented By:

Neha Pradeep, Ansha Prashanth, Sakshi Singh, Sarika Dinesh

What are Autobahns?



Partial Synchrony

(As discussed in the paper)

- **In theory**:
  - Partial synchrony assumes that once the Global Stabilization Time (GST) is reached, the system behaves as if operating within a synchronous network.

- **In real deployments**:
  - Synchronous periods after GST are often interrupted by **blips**.
  - Timeouts during these phases create backlogs, leading to high latency even after GST.

Blips: Short periods during which the system's progress stalls due to network disruptions (timeout violation, replica failures, DDoS attacks, etc.)

Good and Gracious Intervals

- **Good Interval**:
  - Period when the system is synchronous and the consensus process is led by a correct replica.
  - Good intervals capture the periods during which progress is guaranteed

- **Gracious Period**:
  - A specific type of good interval where all replicas are correct.

Hangovers

Any performance degradation caused by a blip that persists beyond the return of a good interval.

- **Unavoidable Hangovers**:
  - Caused by physical network limitations (e.g., insufficient bandwidth, message delays).
  - No protocol can provide progress beyond pace of the network.

- **Protocol-induced Hangovers**:
  - Caused by suboptimal system design, where protocol logic (timeouts, commit rule, etc.) introduces unnecessary delays.
  - Avoidable through careful protocol design.

Seamlessness

A partially synchronous system is seamless if :

- it experiences no protocol-induced hangovers, and
- it does not introduce any mechanisms that make the protocol newly susceptible to blips.

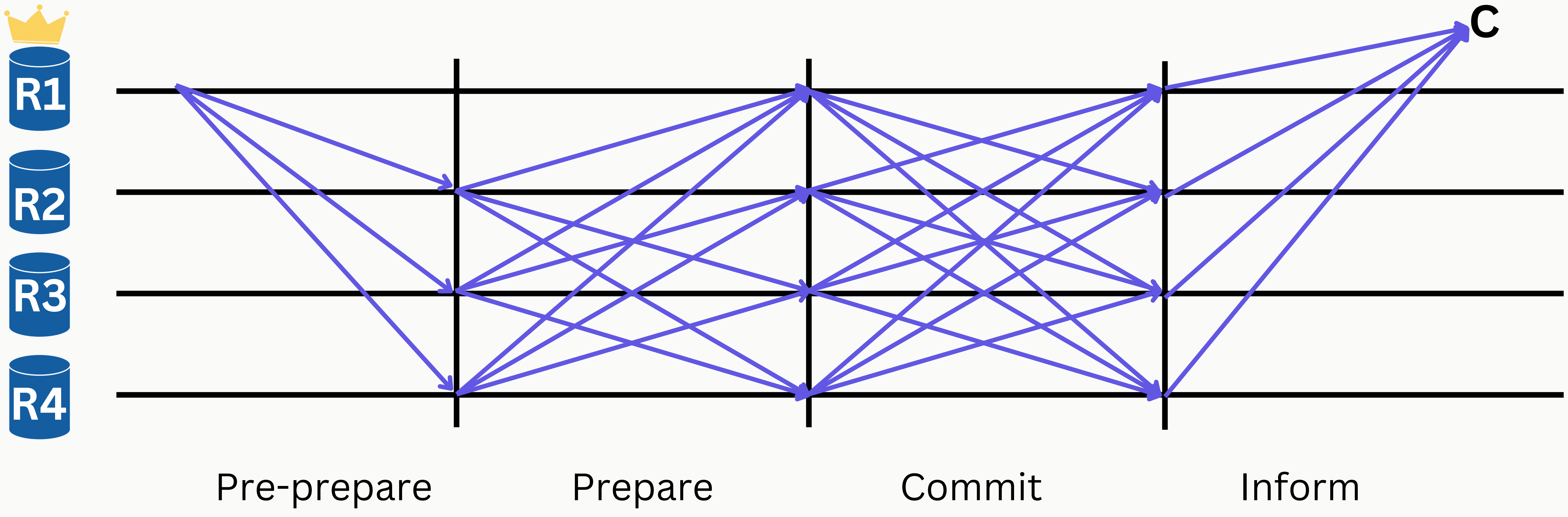
Why do we need Autobahn?

Traditional BFT protocols

Data Dissemination
+
Ordering Logic

Tight Coupling
Lead to protocol-induced hangover

PBFT Overview:



- If consensus ordering logic stalls, data dissemination also stalls.
- When network recovers, backlog commitment time - in order of backlog size.

Challenges of Traditional BFT Protocols

- Tightly couple data dissemination and ordering.
- Optimize for performance after GST (minimizing message exchanges in synchronous cases).
- Resilience issues: Struggle to resume operations after a brief blip.
- Protocol-Induced Hangovers: Blips cause loss of throughput, generating large request backlogs. Performance issues persist even after synchrony is restored.

DAG based Protocols

- Decouple data dissemination and ordering.
- High throughput: Use a DAG of temporally related data proposals for efficient data dissemination.
- Reduced impact of blips: Better resilience compared to traditional BFT.
- Asynchronous model: Optimizes for worst-case message arrivals and uses randomness for progress, without relying on timeouts.

Shortcomings of DAG based Protocols

- Data synchronization is on the timeout-critical path (before voting) which causes high latency during consensus.
- Data proposals must go through multiple rounds of Reliable Broadcast, i.e. process n-f votes to attain non-equivocation.
- Not possible to infer full causal history from single DAG node in constant time. It can only be inferred by recursively tracing back the path of the edges.

Autobahn approach

Data Dissemination Layer

- Continuous, independent, parallel data broadcast.
- Moves at the pace of the network.
- Propose and Vote cycles to confirm availability.
- Cars and Lanes abstraction

Consensus Layer

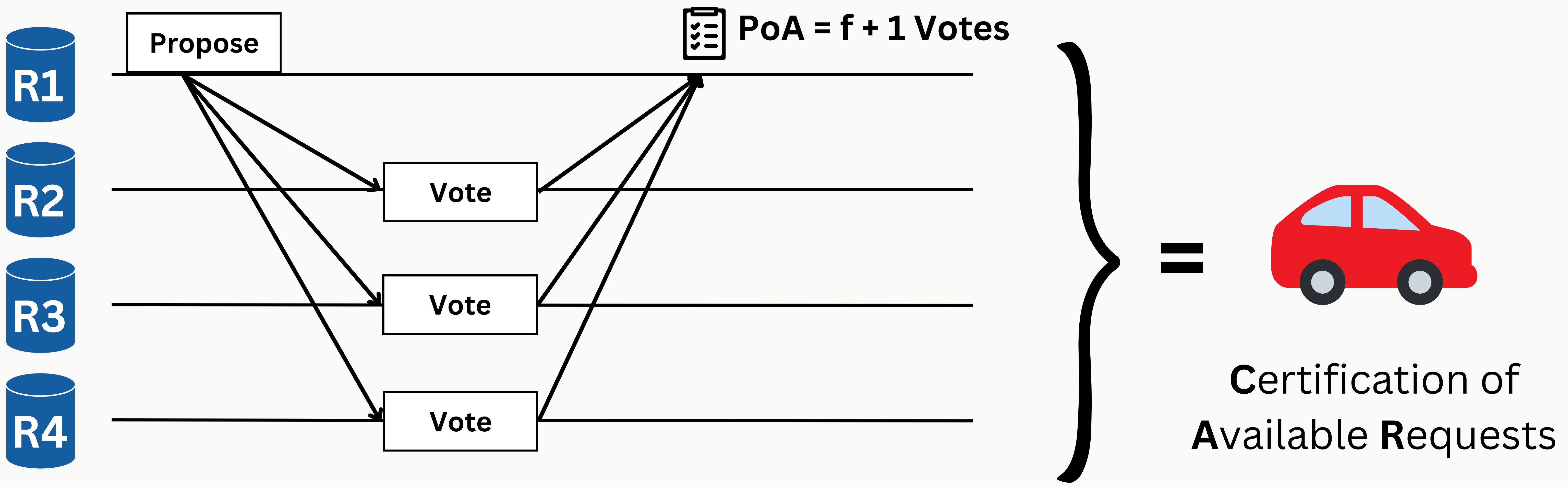
- Agreement achieved on "cuts" or snapshots of the latest data from each lane.
- Can commit an arbitrarily large backlog independent of size.

Data Dissemination Layer

- All replicas act as proposers.
- Data Lane = local chain that implicitly assigns an ordering to its data proposals.
- **Reliable Inclusion** - successfully disseminated data proposals will commit during good intervals.
- Data synchronization completes in parallel with agreement due to voting rules.
- Supports
    - **instant referencing** (quickly verifies history using the latest entry),
    - **non-blocking sync** (no data synchronization occurs on the critical voting path), and
    - **timely sync** (data syncing completes before consensus commits).

CAR Pattern for Data Proposal

PoA - Proof of Availability

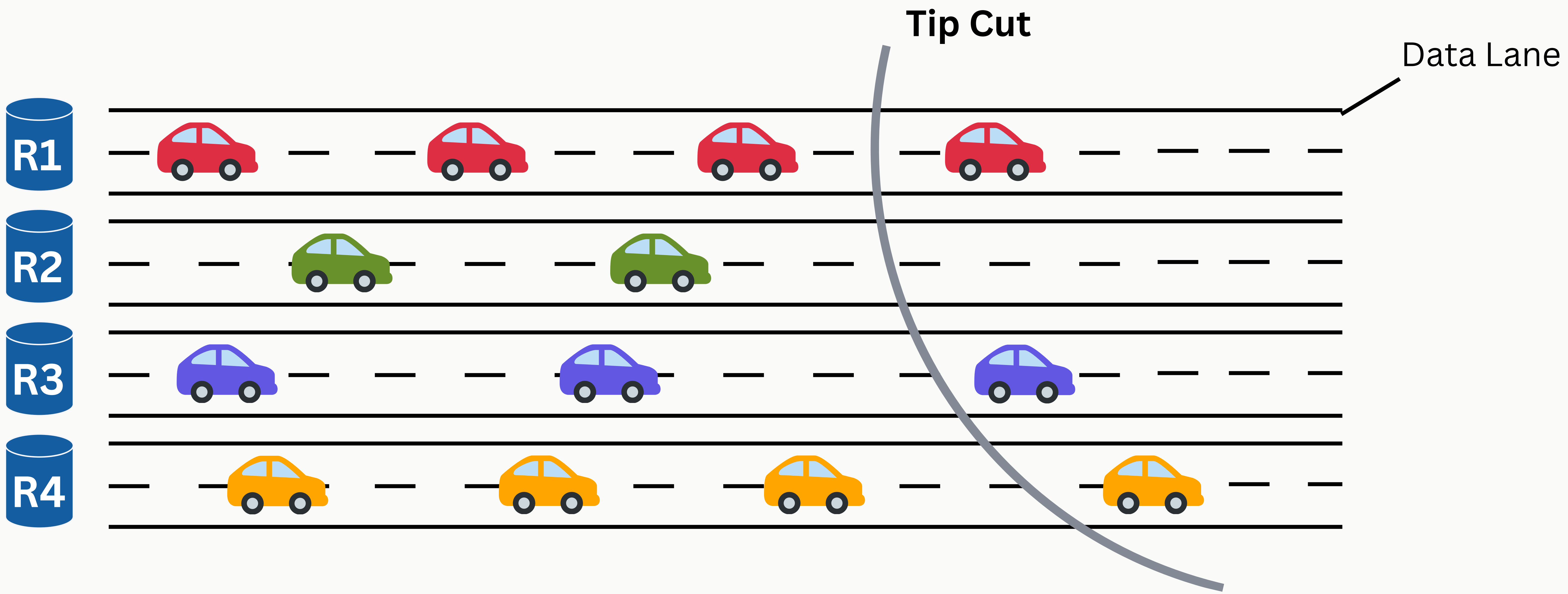


- A CAR consists of a batch of transactions
- PoA gurantees **atleast one** correct replica is in possession of data proposal
- Replicas vote for a car at position i only if its proposal references an approved proposal for car at i -1

Lane structure

- New data proposal must include a **reference** to its previous car’s data proposal.
- Data Lane = local chain that implicitly assigns an ordering to its data proposals.
- A successful car for block i **transitively** proves the **availability** of car for all blocks 0 to i-1 in the proposer’s lane.
- Each replica maintains a local view of all lanes.
- Once a replica has committed tip, it deterministically interleaves all n data lanes to form single total order.

Data Lanes and Tip Cut



- A lane is made up of series of cars that are chained together.
- Each replica operates its own lane at its own rate.
- Tip Cut: Summary of data lane state containing each replica's latest proposal.

Reaching consensus and commit

Consensus Layer

- The consensus layer helps different nodes agree on the same order of proposals, even if they receive them at slightly different times.

- It takes **snapshot** cut of all data lanes periodically.

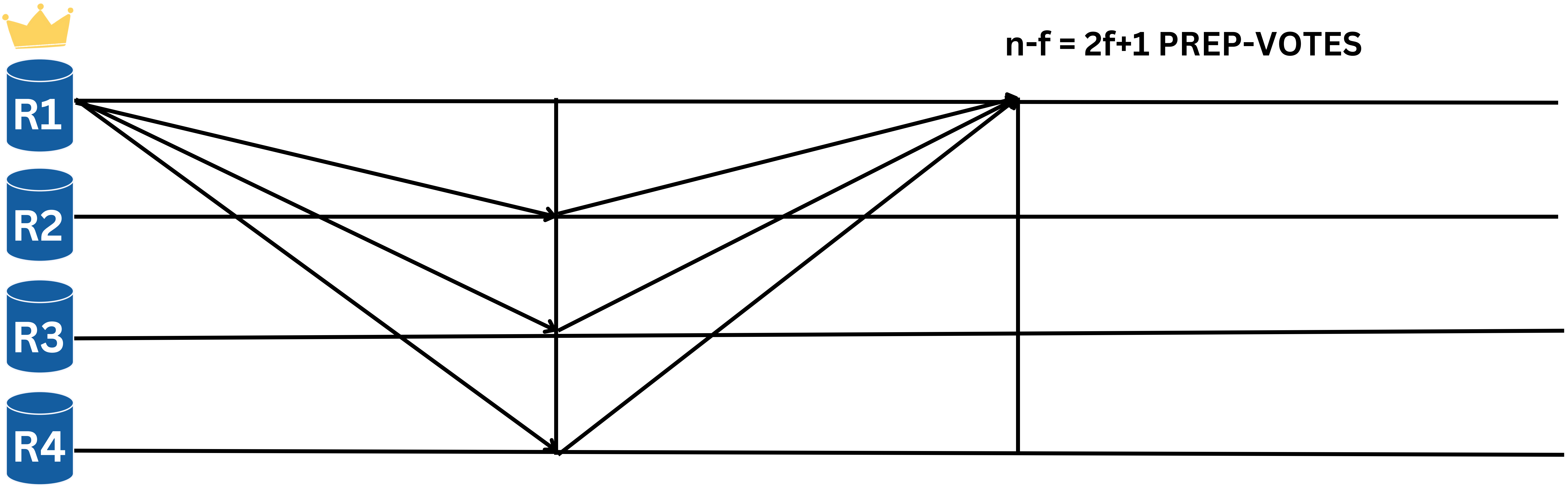
- Consensus proposal contains a **vector of** n **certified tip** references.

- Progresses as a series of **slots,** where every slot s has a **leader**.

- Within each slot, we follow view-based structure. Each view has 2 phases: **Prepare** and **Confirm**.
- Leader proposes new lane cut once slot s-1 commits.

Prepare phase

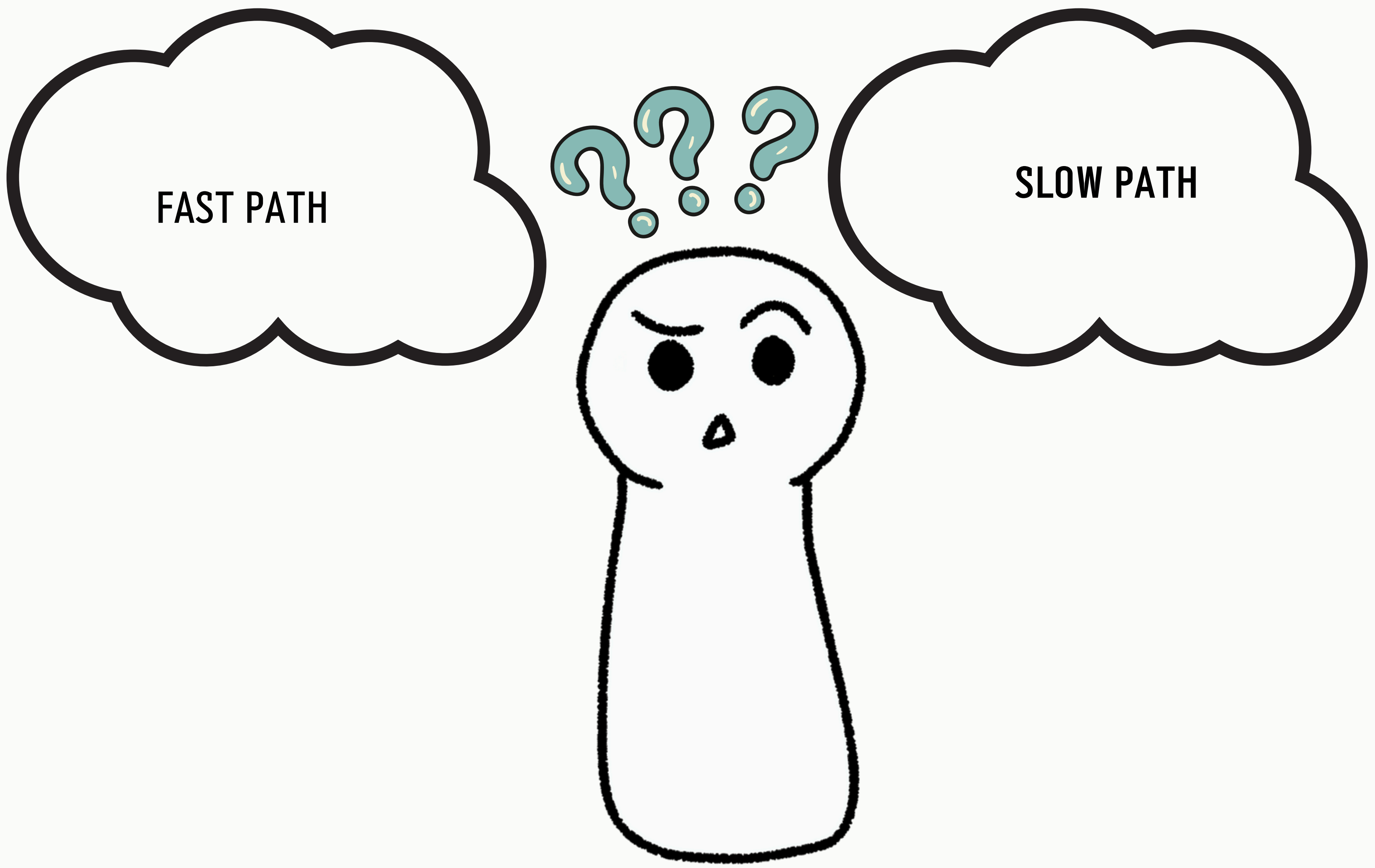
- Prepare phase ensures non-equivocation



Prepare:= ($\langle P \rangle L$,T)
$\langle P \rangle L$:=s,v,cut of latest certified lane tips
Consensus proposal P
Ticket T :=CommitQC$_{s-1}$ or TC$_{s,v-1}$

Prep-Vote:= $\langle$dig=h(P)$\rangle$R
R2,R3,R4 stores a copy of the proposal

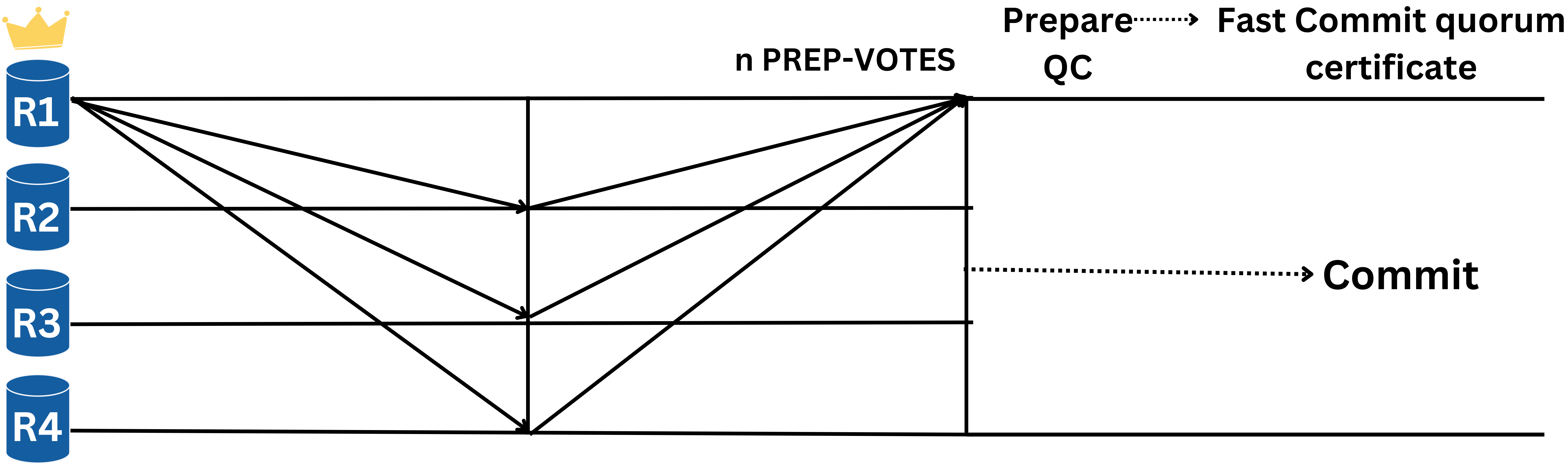
PrepareQC:= (s,v,dig, {PREP-VOTE})


FAST PATH
SLOW PATH

If prepareQC(Prep-vote)==no of replicas → FAST PATH

If prepareQC(Prep-vote) < no of Replicas → SLOW PATH

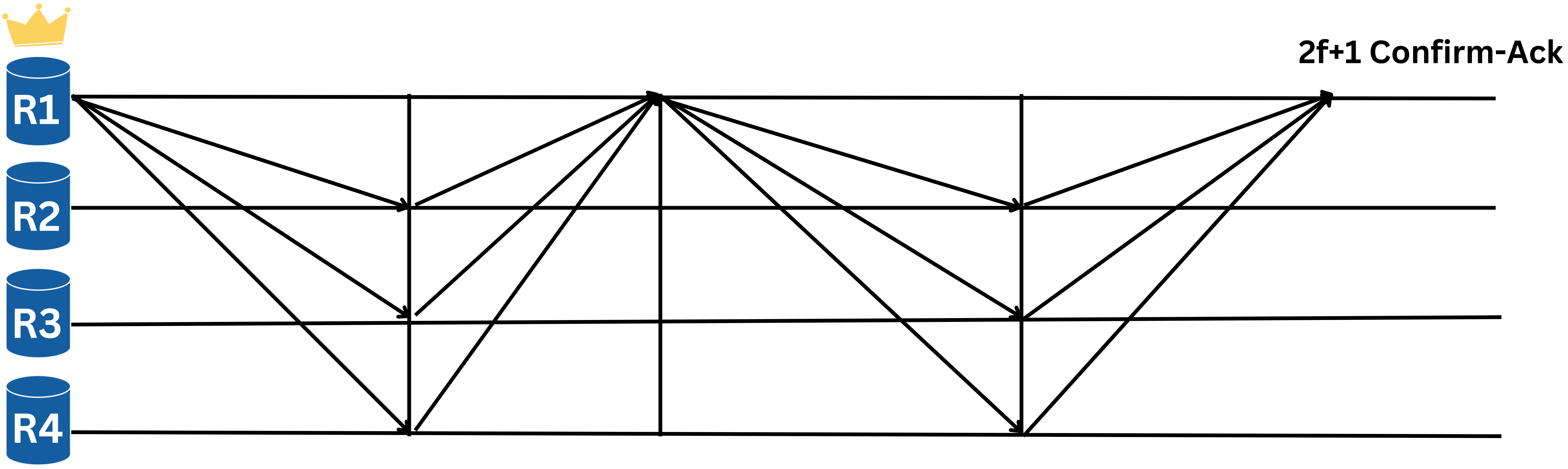
Fast Path



- During **gracious intervals**, leader can receive **n** such prep-votes.
- Quorum of n votes directly ensures durability across views.
- QC upgraded to fast commit quorum certificate, proceeds to commit step skipping the confim phase.

Confirm phase

- Confirm phases ensures durability across views



Confirm:= ⟨PrepareQC s,v⟩ is broadcasted

Confirm-Ack
R also buffers PrepareQC locally as conf[s]=PrepareQC

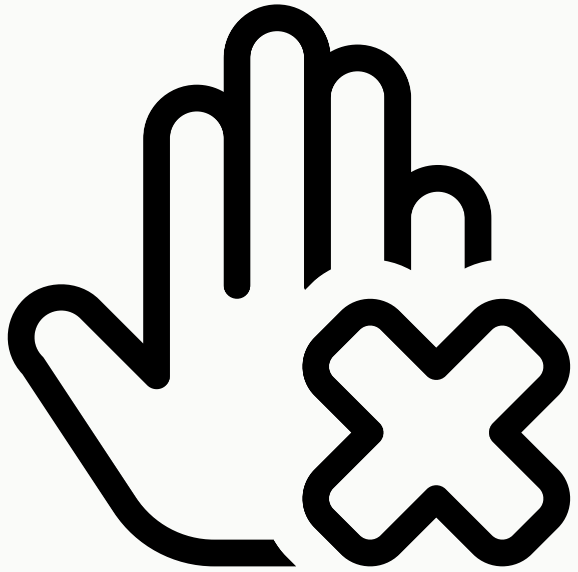
- Leader collects 2f+1 Confirm-Ack into **Commit QC** and broadcasts it to all replicas.

View Change

Why do we need View Change?

Malicious leader

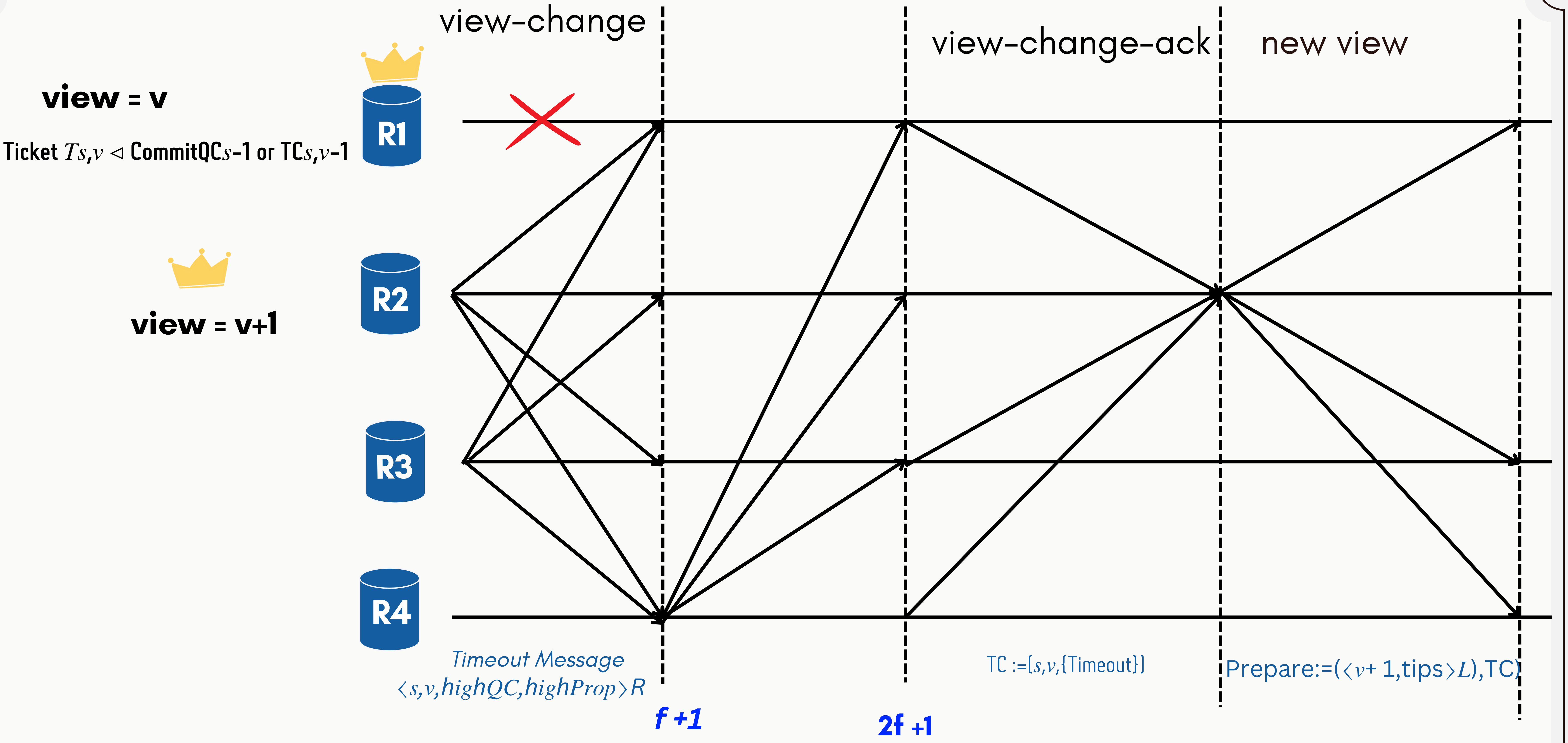
Timeout!!



System halts the process

New Leader

View Change in Autobahn



- current-view$R$ ≤ Timeout.v
- not received a CommitQC for that slot.

Optimization

- Pipelining Cars

  Impact: Increases efficiency but having many proposals that might never be accepted, causing wasted work and resources.

- Uncertified Tips
  i) Leader Tips

  Impact: Reduces overall wait time if leader is honest else causes small delay.

  ii) Optimistic Tips

  Impact: Increases the process speed but causes temporary synchronization issues.

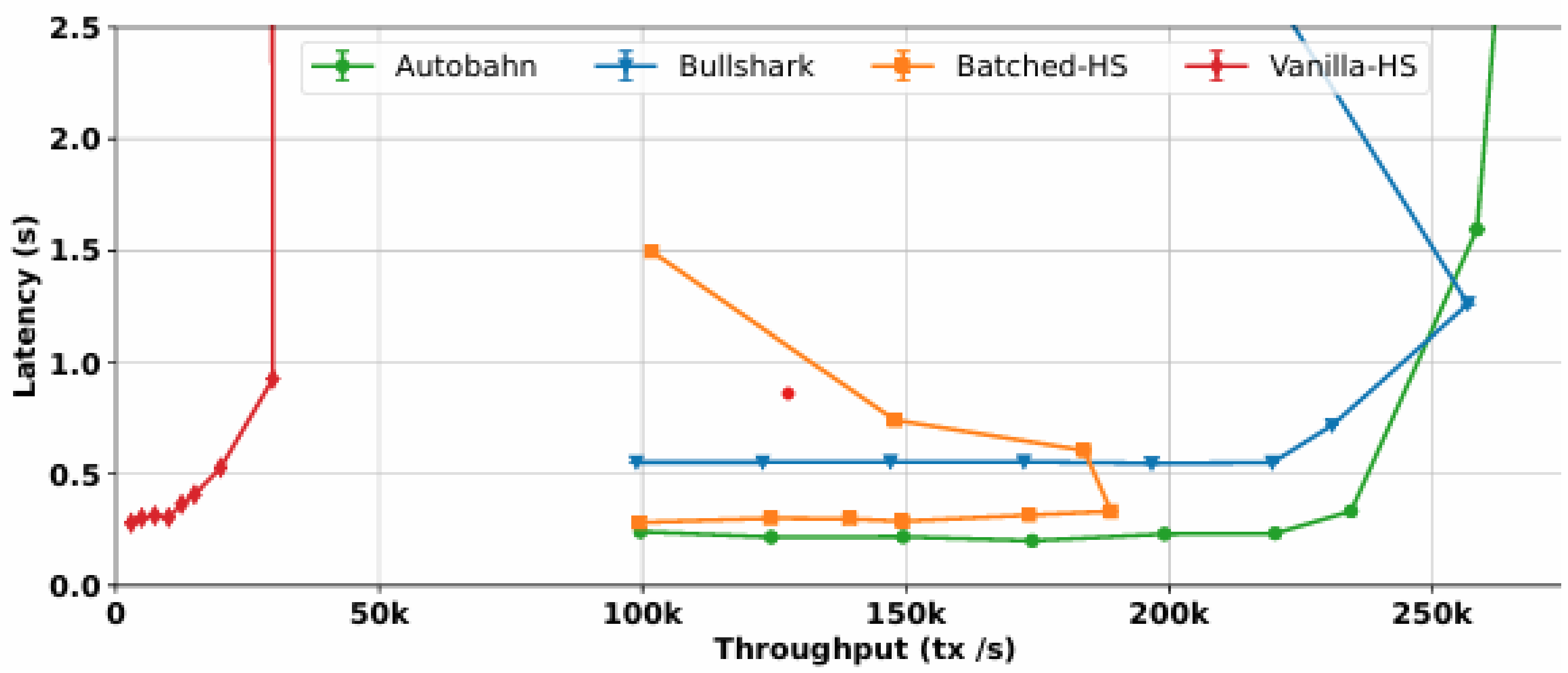
- Signature Aggregation

  Impact: Simplifies communication thereby reducing the complexity.

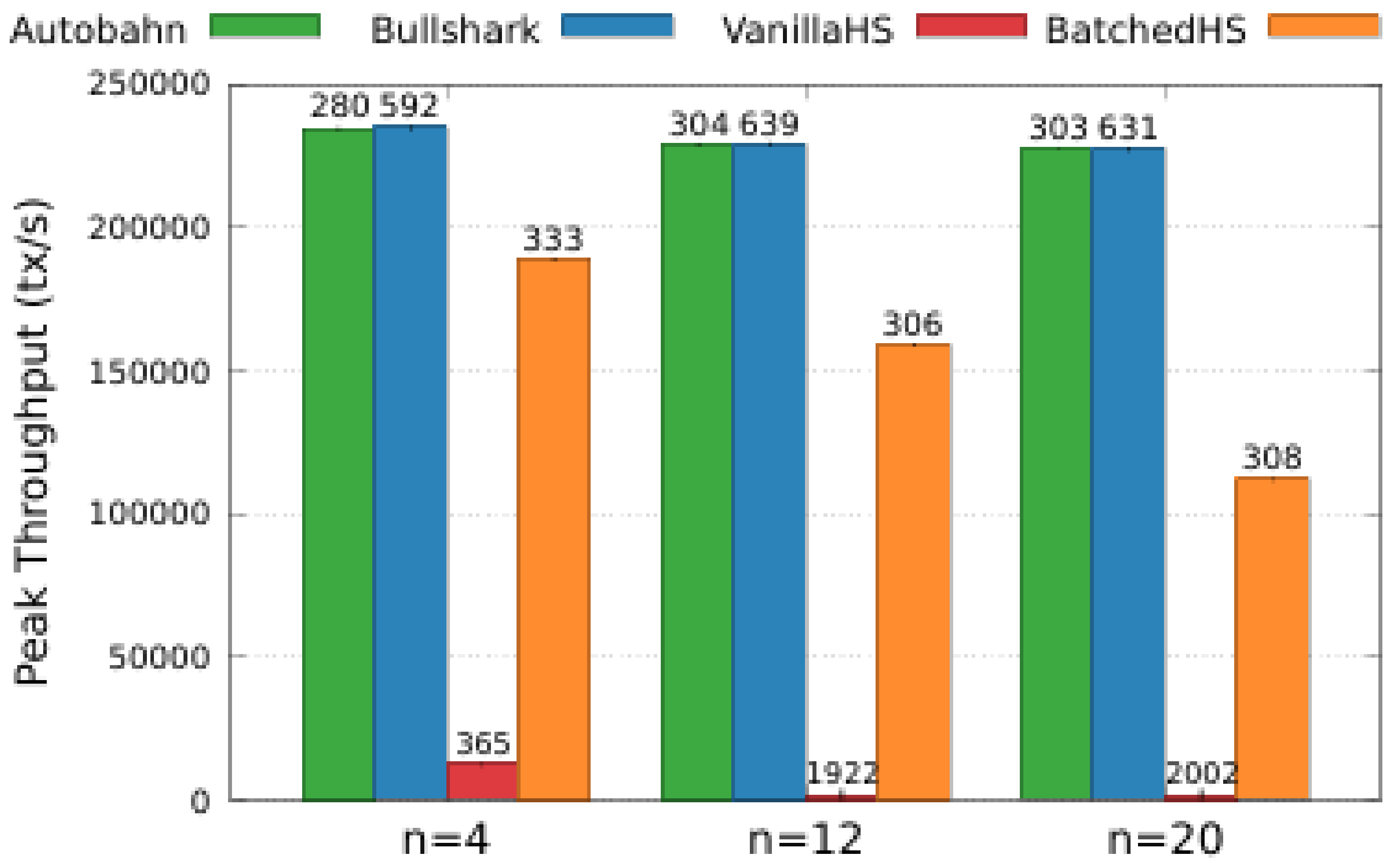
- All to All communication

  Impact: Reduces Latency but loses linear efficiency.

Evaluation: Autobahn vs Others (Performance and Scalability)

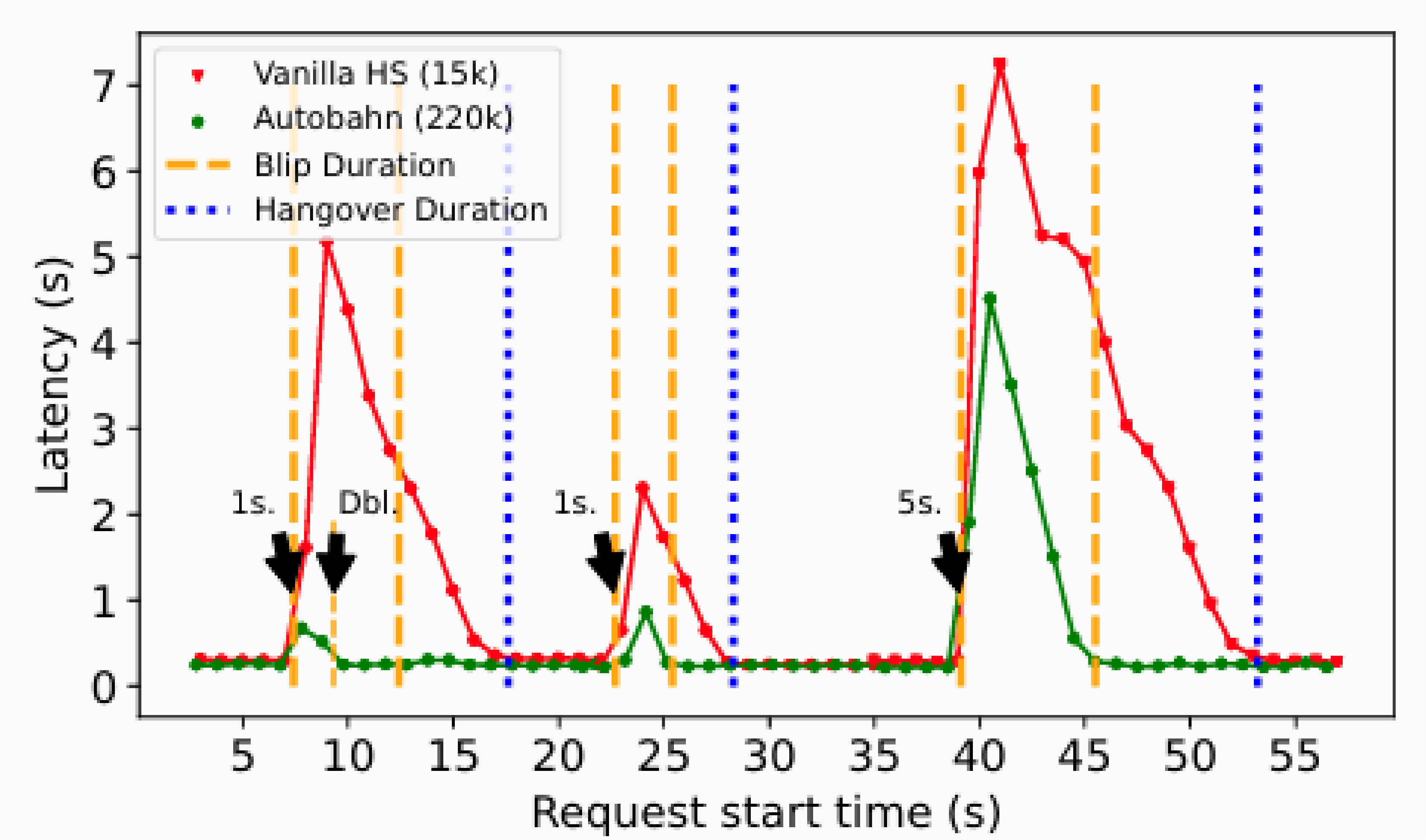


i. Throughput and Latency under increasing load

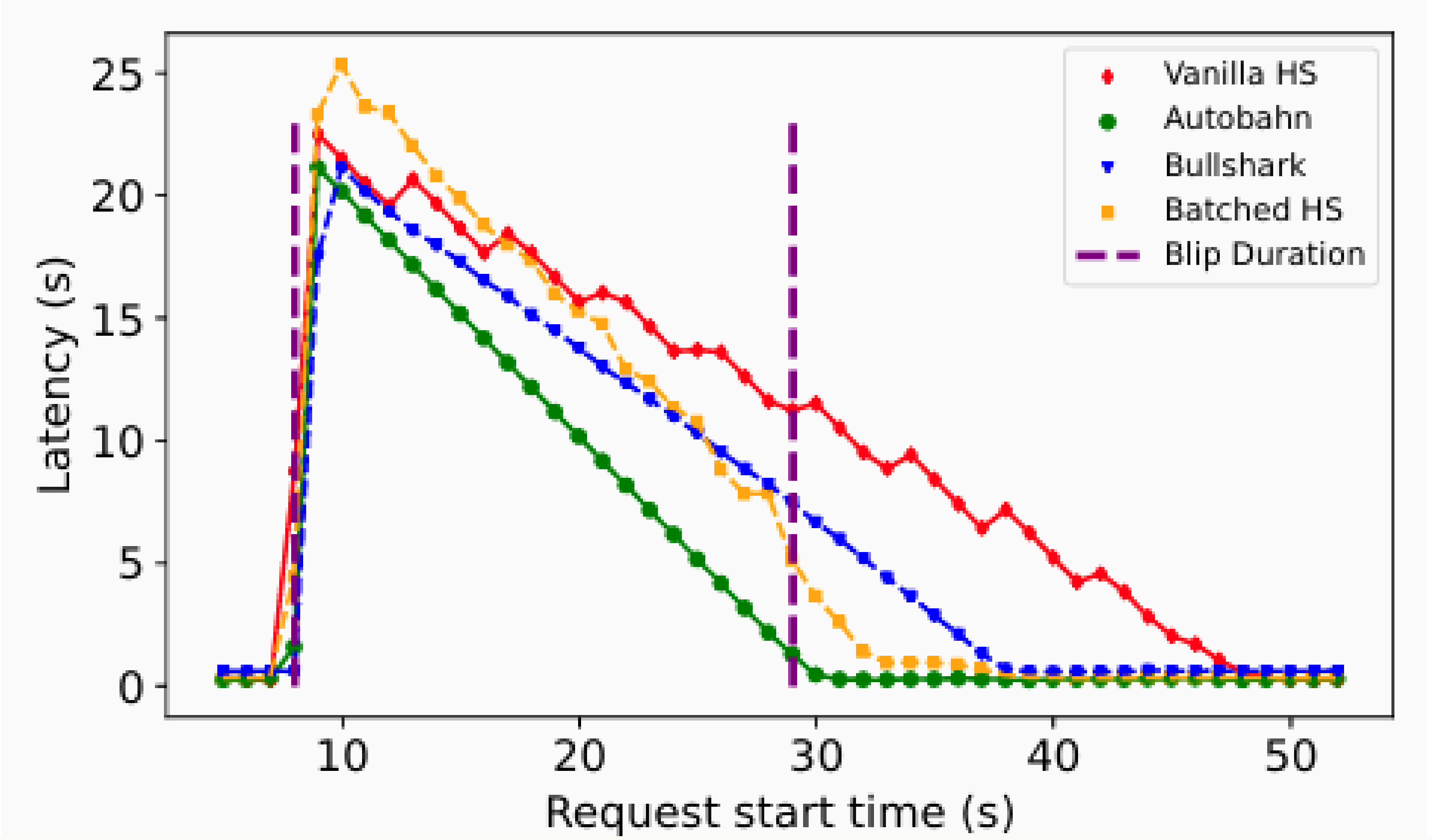


ii. Peak throughput for varying n. Numbers atop bars show measured latency (ms)

Evaluation: Autobahn vs Others (Latency)



iii. Leader failures



iv. Partial Partition

Thank you!