Autobahn: Seamless high speed BFT

Paper presentation for COEN 6731

Presented by: Xiaohan Wang (40233750)

Chenyang Ma (40238872)

Introduction

“This work presents **Autobahn**, a novel high-throughput BFT protocol that offers both low latency and seamless recovery from blips. By combining a highly parallel asynchronous data dissemination layer with a low-latency, partially synchronous consensus mechanism, ……”

Autobahn —— Seamless partial synchronization of BFT consensus protocol

- Data dissemination layer: responsible for disseminating transaction data
- Consensus layer: responsible for reaching consensus.

Background

Partial Synchrony Model

A consensus protocol model in a distributed system. The partial synchronization model assumes that after GST(Global Stabilization Time), the network will enter a synchronization state, that is, the delay of message delivery will have a known upper limit.

- Before GST, the system may be in an **asynchronous state**, with no upper limit on message latency. Partial synchronization protocols do not guarantee progress at this stage, but they do guarantee security.
- After GST, the system enters a **synchronous state**. The protocol at this stage guarantees progress and optimizes latency.

Background

Problem:

In theory, after reaching GST, there will be endless stable periods, but in practice, **some synchronization is segmented.** Because various factors can cause the system to stagnate, GST needs to be reached again to restore stability.

“Unfortunately, we found that existing partial synchronization protocols are not actually efficient under true partial synchronization: they must choose between achieving low latency when meeting timeouts or maintaining robustness to failures.”

Background

To properly assess the performance of a partially synchronous system, we introduce the dual notions of **hangovers** and **seamlessness**.

- **good intervals**: The system is synchronous, and the consensus process is led by a correct replica.
- **Gracious Interval**: A special case of a good interval is when all copies are correct.
- **Hangovers**: A hangover is any performance degradation caused by a blip that persists beyond the return of a good interval.
- **Seamlessness**: A partially synchronous system is seamless if it meets the following two conditions: (1) experiences no protocol-induced hangovers; (2) No new blips are introduced.

Autobahn-Design Overview

standard assumptions: $N = 3f + 1$ replicas, at most f of which are faulty

Consists of two logically independent layers:

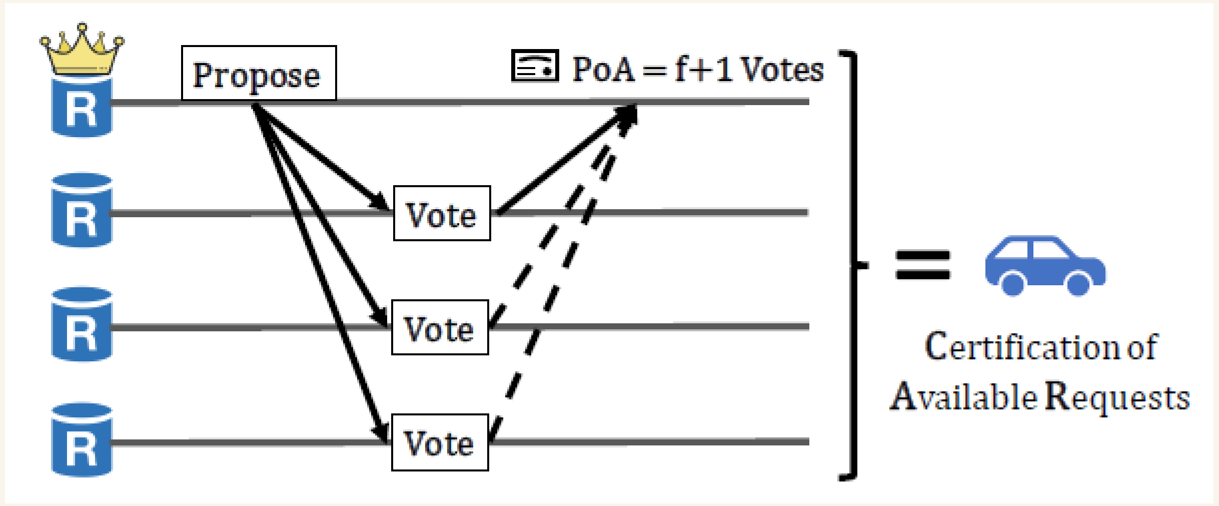
Data dissemination: Each replica independently generates data proposals and ensures Proof of Availability(PoA) of data through broadcasting and voting.

Consensus: Leader use tip cut to efficiently summarize data channel status, coordinate viewpoints, and sort all certified data proposals in a total order.

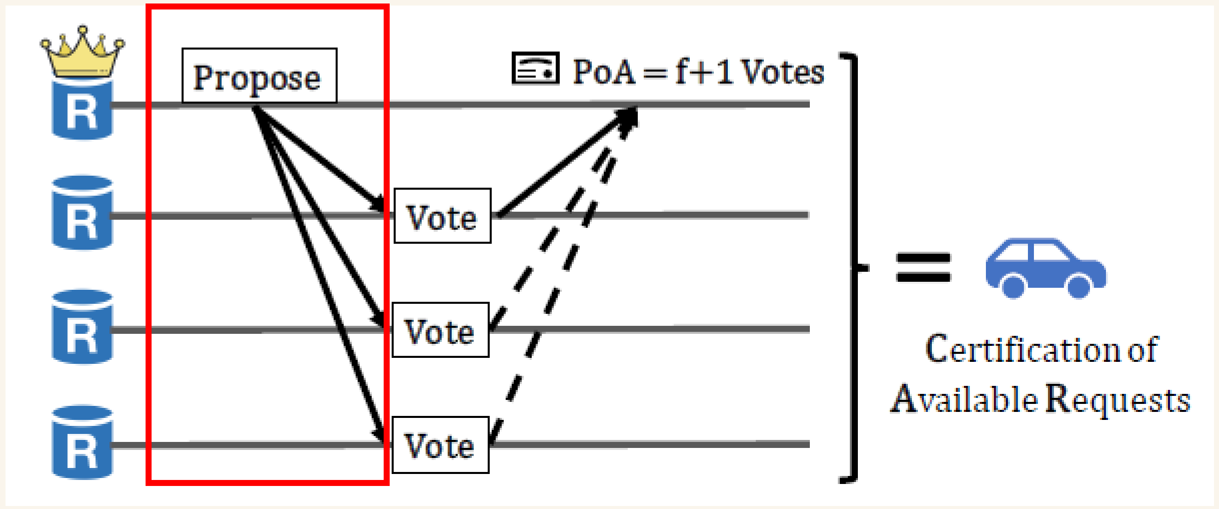
Autobahn-Data dissemination layer

Three properties:

1. Instant referencing
2. Non-blocking sync
3. Timely sync



Autobahn-Data dissemination layer



Step 1

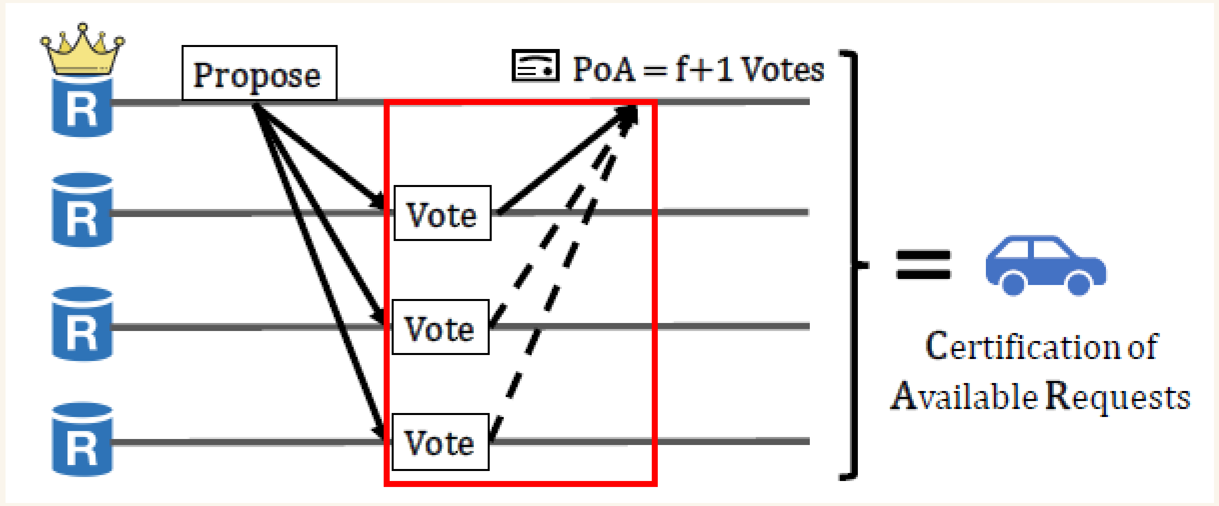
The proposer packages a batch of things to create a new data propose, then broadcast the message to all the other replicas.

A Prop should include:

- **pos**: the position of the proposal in the lane
- **parents**: the hash of previous lane tip
- **cert**: the availability proof of the previous proposal

A proposer, when proposing a new data proposal, must include a reference to its previous car's data proposal.

Autobahn-Data dissemination layer



Step 2

checks whether the received Prop is valid:

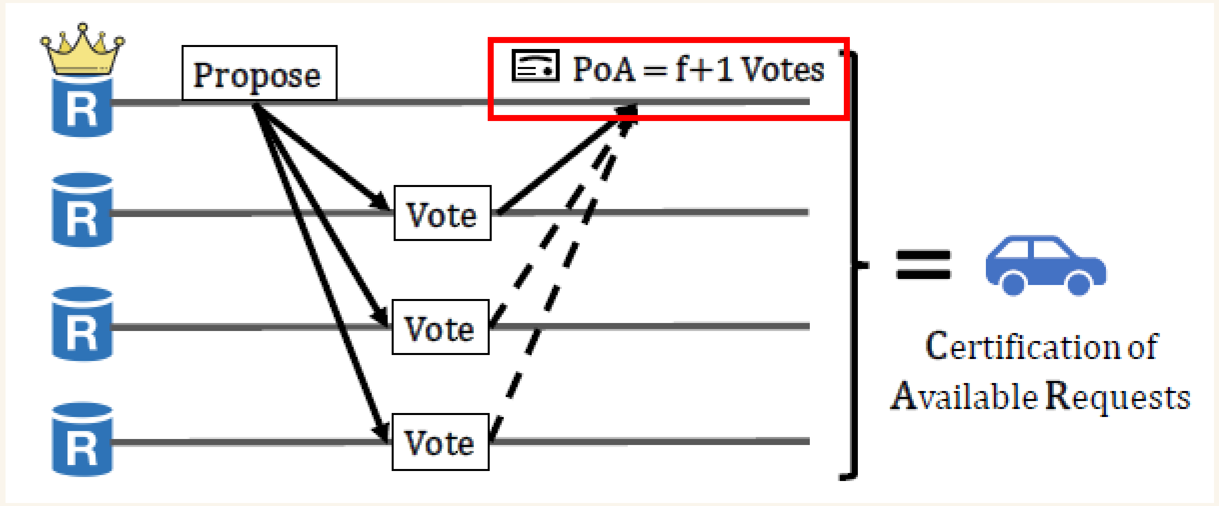
1. it has already voted for the parent of this proposal;
2. it has not voted for this lane position before.

then, vote.

Recursiveness: Validating the head of the lane is thus sufficient to reference arbitrary lane state (enabling instant referencing) and confirm that it has been disseminated (necessary for non-blocking sync).

$\longrightarrow$ allowing the consensus layer to experience timely-sync.

Autobahn-Data dissemination layer



Step 3

The proposer aggregates f+1 distinct matching Vote messages into a PoA, which it will include in the next car.

V.S. DAG-BFT must process n-f votes in order toachieve availability

$\downarrow$

Low latency, high throughput

Autobahn-Consensus layer

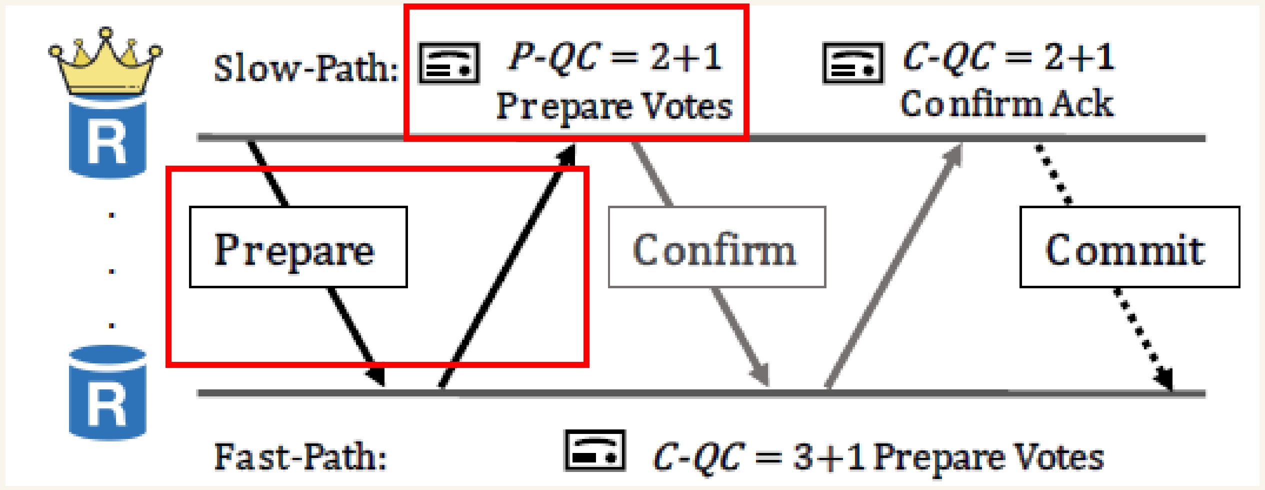
The role of the consensus layer:

to reconcile the views and to totally order all certified data proposals.

input: tip cut

1. Autobahn achieves consensus throughput that scales horizontally with the number of lanes;
2. Autobahn can commit an arbitrarily large backlog with complexity independent of its size.

Autobahn-Consensus layer



Prepare Phase (p1)

Leader

Presents a new section and broadcasts a Prepare message

Replica

Process the Prepare message and Vote, sending a Prepare Vote in response

Leader

Wait for at least 2f+1 matching prepare votes to be aggregated into P-QC(Prepare Quorum Certificate)

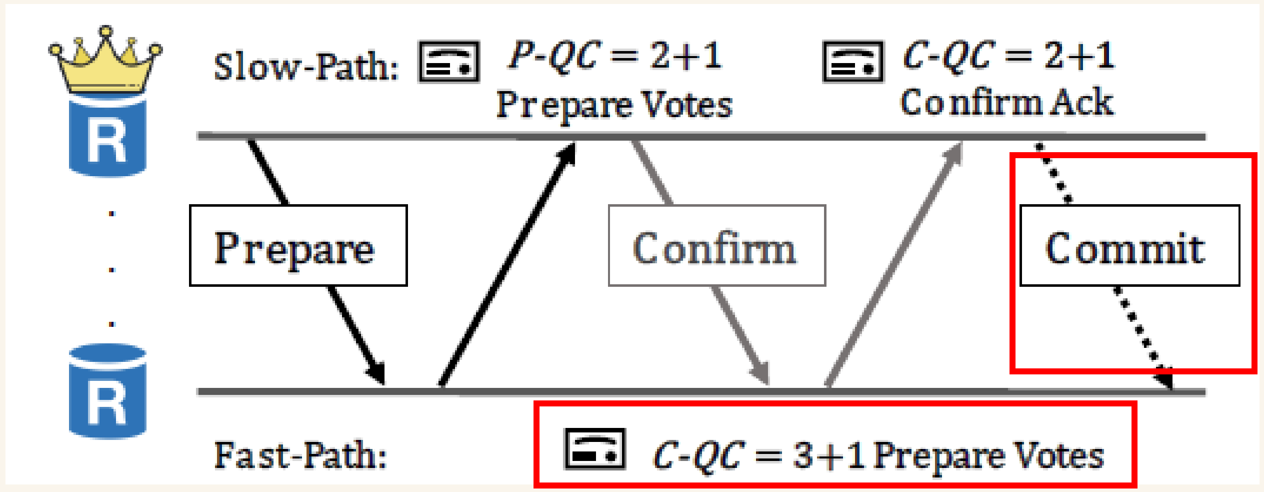
at most f of which are faulty,

$f < f + 1$

Ensure that the correct copy dominates the consensus process

Autobahn-Consensus layer

Fast Path (In gracious intervals)

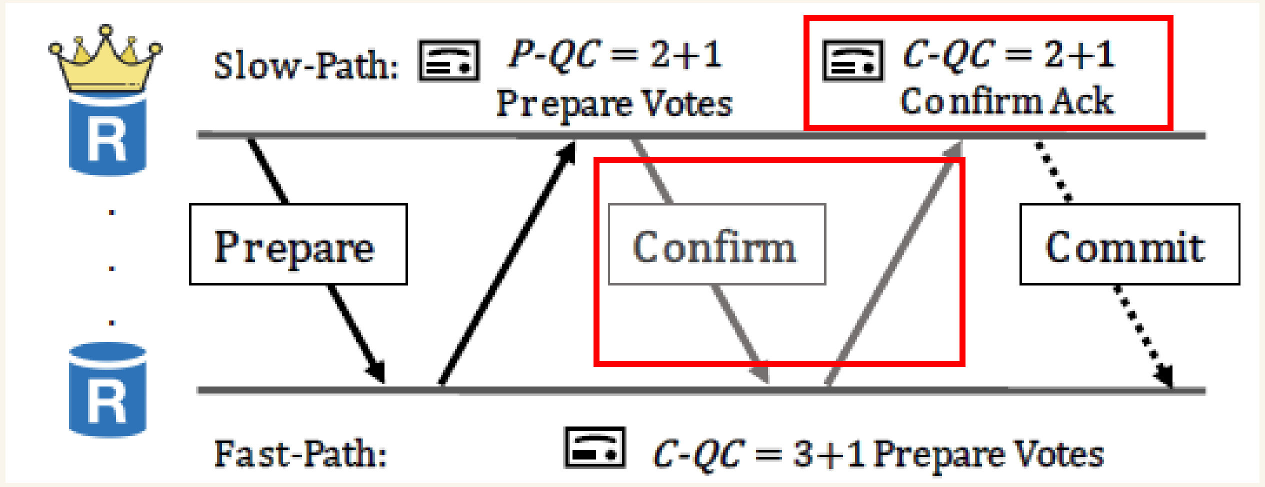


Slow-Path:
P-QC = 2+1 Prepare Votes
C-QC = 2+1 Confirm Ack
Prepare
Confirm
Commit
Fast-Path:
C-QC = 3+1 Prepare Votes

Leader

Receive Prepare Votes for all n copies, aggregate them into a Fast Commit QC (upgrade P-QC to C-QC), skip the confirm phase and broadcast directly.

Autobahn-Consensus layer



Confirm Phase (p2)

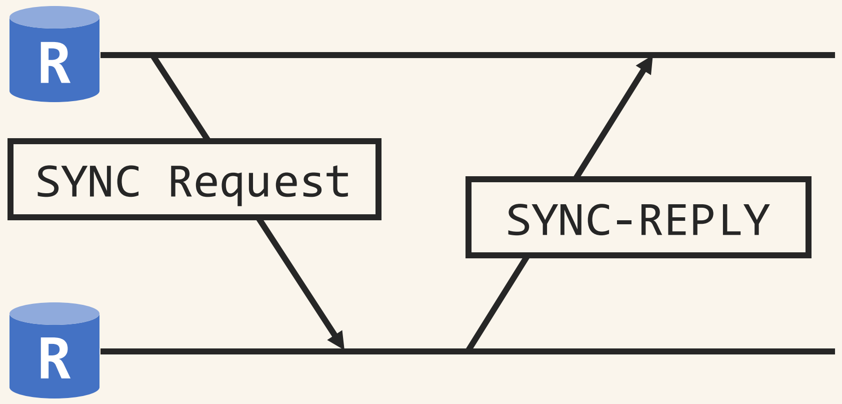
Leader	Broadcast a Confirm message to all replicas
Replica	returns an acknowledgment and buffers the P-QC locally
Leader	aggregates 2f+1 matching Confirm-Ack messages into a C-QC(Commit Quorum Certificate), and broadcasts it to all replicas.

Autobahn-Consensus layer

Processing committed cuts

Step 1: Seamless data synchronization

In order to handle the tip cut, replica needs to ensure that it has all the data proposals for each lane.



- Based on Non-blocking Sync: Replicas can participate in consensus voting even if their local state is inconsistent.

- Based on Timely Sync: Replicas can synchronize the missing data with a single message exchange.

Autobahn-Consensus layer

- Processing committed cuts

Step 2: Creating a total order

After synchronization, the replica needs to create a total order based on the information in the section.

1. Determine the starting point of each channel;
2. Update the latest submission location;
3. Append the new proposal to the log.

Example:

Lane A: Propose 1----Propose 2----Propose 3

Lane B: Propose 1----Propose 2----Propose 3

1. start_A.pos=2; start_B.pos=3
2. last_commit[A]=3; last_commit[B]=3
3. Total order: A_2 A_3 B_3

Autobahn-Consensus layer

- Lane Coverage

  A sufficient number of new data proposals (at least n-f) are required in the channel, otherwise there is no need to initiate consensus.

  - Flexibility: Used to control the speed at which consensus is advanced.

- Reliable Inclusion

  Ensures that all certified proposals (including those with Byzantine copies) are reliably submitted.

  - No discrimination against slow lanes;
  - Limit the waste of Byzantine copies

View change

If the leader fails, Autobahn relies on a timeout mechanism to elect a new leader. The specific steps are as follows:

- Step 1: Start a timer. If it times out, broadcast a "timeout message" and collect timeout messages.
- Step 2: If at least F+1 replicas receive the timeout message, they will enter a state of rebellion. When a replica receives 2F+1 timeout messages, it assembles a Timeout Certificate (TC).
- Step 3: All replicas switch to the new view and start a new timer. During the election of a new leader, the TC must be carried as a valid credential.
- Step 4: Select the highest-certified proposal or the proposal with majority support from the TC as the new leader, broadcast the new proposal, and continue progress after verification by the replicas.

Parallel Multi-Slot Agreement

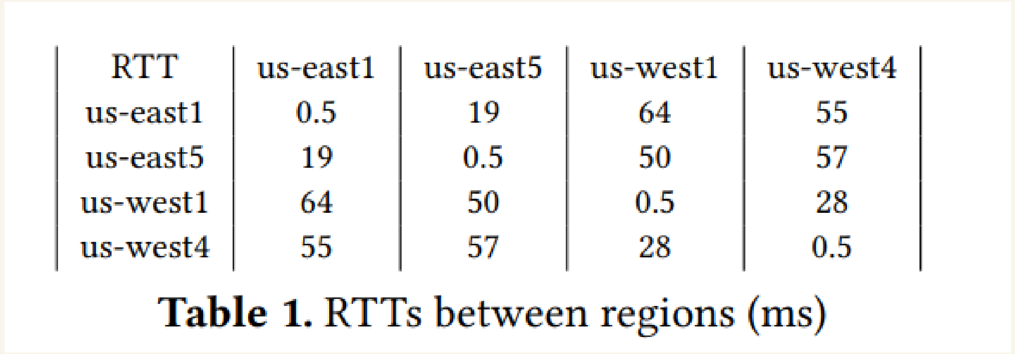
- Traditional BFT protocols must wait for the previous slot to be committed before processing the new slot. Unfinished parallel consensus instances may accumulate, leading to sequential delays and occupying system resources.
- Autobahn allows the next slot instance (S) to start in advance without waiting for the previous slot (S-1) to be committed. Each slot instance independently maintains the proposal, voting, and certificate states without interfering with each other.

Experimental Setup

The experiment is based on the Google Cloud Platform (GCP) to build a distributed system testing environment, with nodes deployed across four regions within the United States to simulate real-world cross-regional network scenarios.

Each node's hardware configurations: a 16-core CPU, 128GB RAM, 1TB SSD storage, and a 10Gbps network bandwidth.

Network latency between regions is quantified using round-trip time (RTT), with specific values provided in the table.



RTT	us-east1	us-east5	us-west1	us-west4
us-east1	0.5	19	64	55
us-east5	19	0.5	50	57
us-west1	64	50	0.5	28
us-west4	55	57	28	0.5

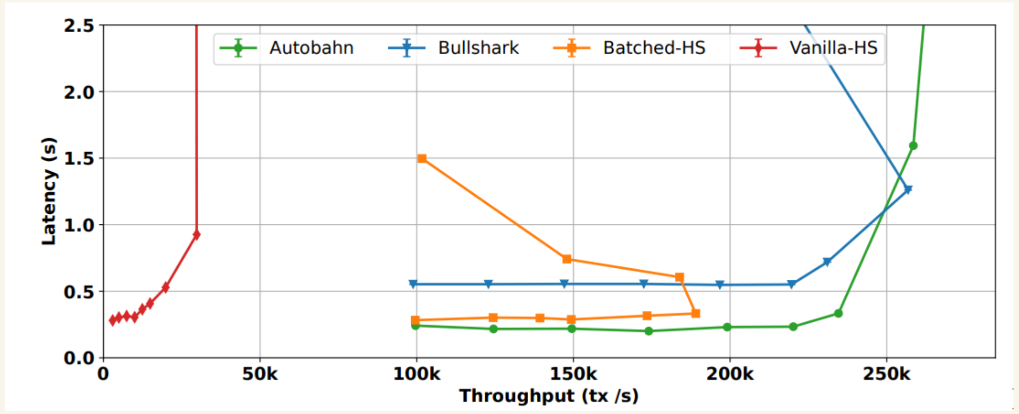
Table 1. RTTs between regions (ms)

Experimental Setup

The experiment runtime is fixed at the 60s, using a rotating leader mechanism to avoid single points of failure and setting a 1s timeout mechanism to trigger fault tolerance and recovery. All tests exclude the fixed client-to-node network latency (about 15ms).

- Input load: The total number of transactions sent by all clients per second, where each transaction consists of 512 bytes of random data.
- Comparison of four protocols: Autobahn, Bullshark, BatchedHS, and VanillaHS.
- Performance evaluation: Throughput (the number of transactions processed per second, tx/s), latency (the time from transaction arrival to commitment, ms).

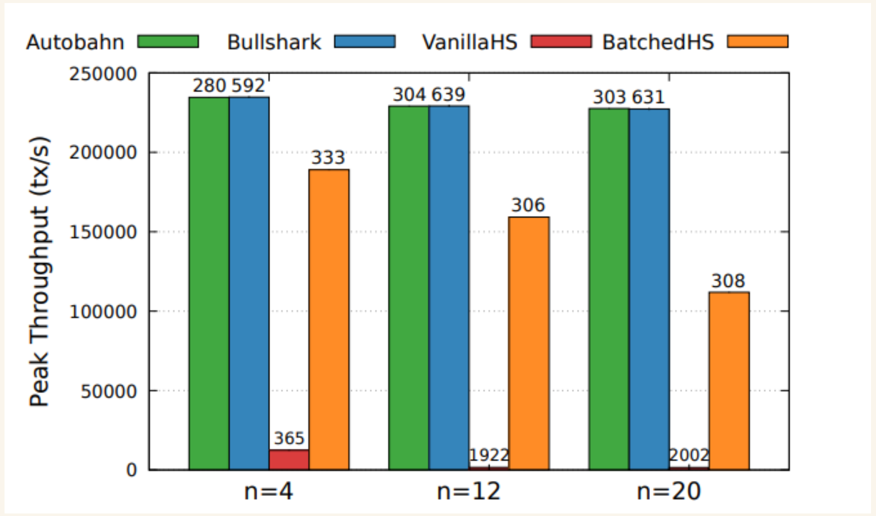
Performance of Autobahn in ideal conditions



- Lowest Latency: 280ms
- High Throughput: 234K tx/s
- Fast Coordination: Fast path requires only 4ms (traditional 10ms)

- **Lower latency:** Faster transaction confirmation, improving system responsiveness
- **Sustained high throughput:** Suitable for high-concurrency applications, enhancing scalability
- **Fast-path optimization:** Reduces consensus overhead, improving protocol efficiency

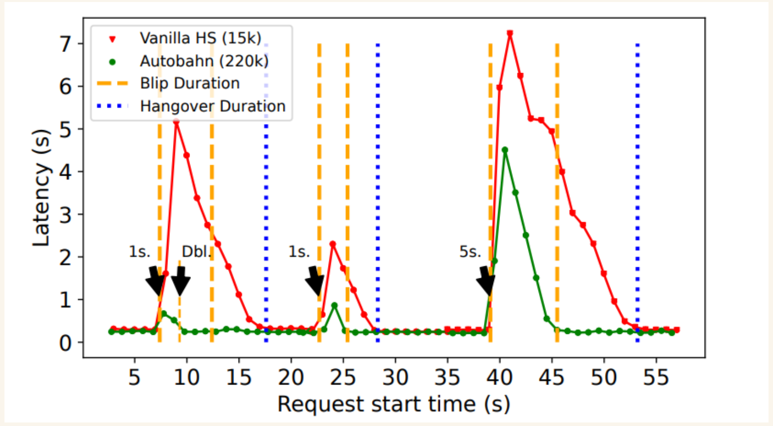
Increasing the Number of Nodes (N)



- Autobahn maintains stable throughput at 250K tx/s as the number of nodes increases from 4 to 20, with no significant latency fluctuation.

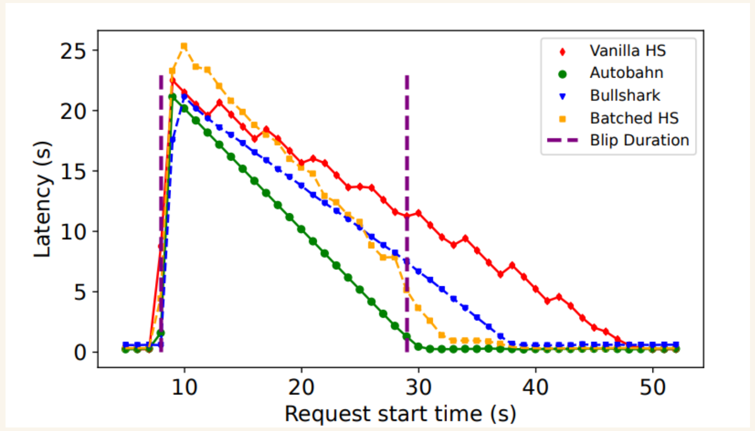
- BatchedHS performs well at small scales (N=4) with 189K tx/s, but as N increases to 20, throughput drops sharply by 41% (down to 108K tx/s).
- VanillaHS experiences a drastic performance collapse at N=20, with throughput plummeting to 1.5K tx/s (only 10% of its initial performance) and latency surging by 6.6×.

Performance of leader failure



- When the leader fails, Autobahn triggers the timeout mechanism (1s), causing only a brief latency fluctuation. It fully recovers to normal levels (280ms) within 1 second, while maintaining stable throughput (234K tx/s).
- VanillaHS experiences a latency surge to 6.2s during the failure, and even after recovery, it remains delayed by 1.3 seconds (3.5× higher than normal). Throughput drops drastically by 90% (from 15K tx/s to 1.5K tx/s).

Recovery performance after network partition



- Autobahn maintains a throughput of 220K tx/s during the partition (by propagating to F+1 replicas). After the partition is resolved, it synchronizes missing data within 1 second, and latency returns to normal levels.
- BatchedHS/Bullshark requires 9s to recover after the partition is resolved, with a decline in throughput.
- VanillaHS completely fails (almost zero) and requires manual intervention to recover.

Strength

- **Autobahn achieves low latency and efficient coordination** by leveraging the fast path and optimistic hints (referencing unauthenticated but reliable proposals), significantly reducing waiting time in the consensus process.
- **Autobahn demonstrates strong fault tolerance**, remaining resilient to temporary failures such as leader crashes or network partitions. It can quickly recover, ensuring seamless consensus execution.
- **Autobahn's architecture is not constrained by a specific consensus mechanism**, making it easily adaptable to future algorithmic improvements and upgrades, including advancements in asynchronous protocols.

Shortcomings

- In some cases, Autobahn may enable unauthenticated proposal optimization, which can further reduce latency but may lead to data synchronization blockage and potential Byzantine attacks.
- Protocol design introduces many mechanisms, including parallel multi-slot protocol, fast channel, lane structure, etc, which increase the complexity of system implementation.
- Autobahn's performance advantage relies on favorable network conditions (e.g., low latency and high bandwidth), making it difficult to apply in high-latency or unstable network environments.