PMNet

In-Network Data Persistence

Presenter: **Korakit Seemakhupt**
Sihang Liu, Yasas Senevirathne, Muhammad Shahbaz, Samira Khan

University of Virginia

Purdue University

Presented at 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)

Artifact available at pmnet.persistentmemory.org

Summary

Motivation

- Datacenter applications usually store data in separate servers and manage through network
- Long latency of accessing data on the server slows down clients
- In-network compute reduces read requests' latency but not update requests

Key Insight

- Network devices lack native **data persistence** support and cannot maintain data upon failure

PMNet

- Enhance existing **P4** switch by adding persistent storage support
- Logs update requests in network devices and moves server's latency off the critical path
- Recovers server using logged requests in case of a failure
- Integrates in-network data persistence with data **replication** and **caching**

Evaluation

- Programs PMNet packet processing with **P4** and implements on FPGA
- Evaluates end-to-end system
- Improves throughput by 4.27x and tail latency by 3.23x over client-server baseline

Outline

Background and Motivation

In-network Data Persistence

PMNet Design

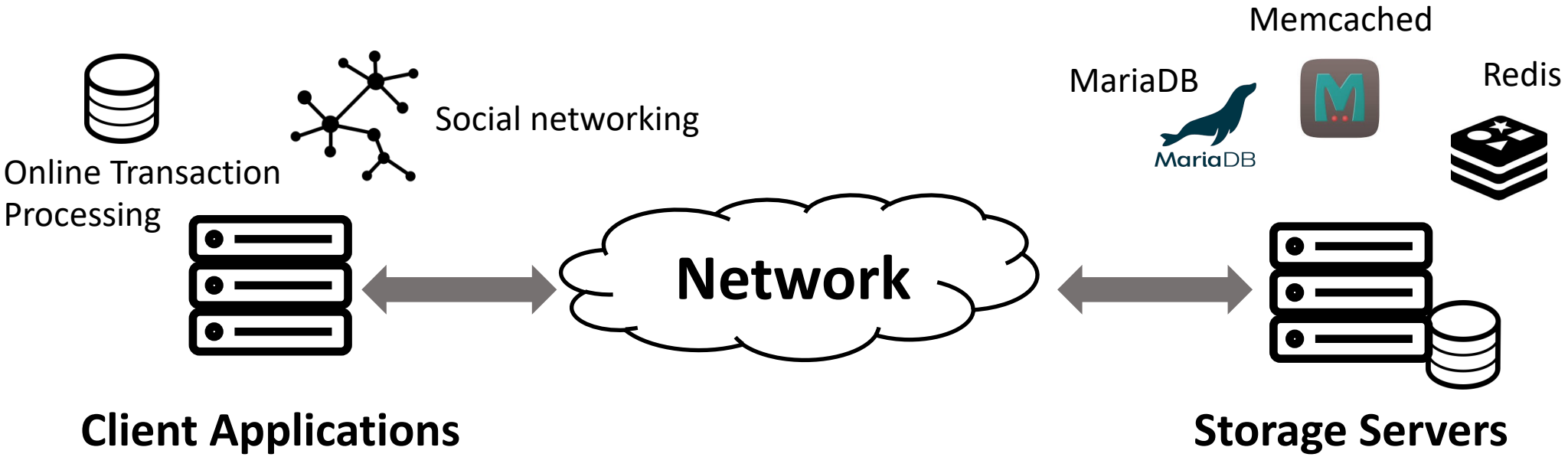
Caching and Replication

Evaluation

Conclusion

Storage Applications in Datacenter

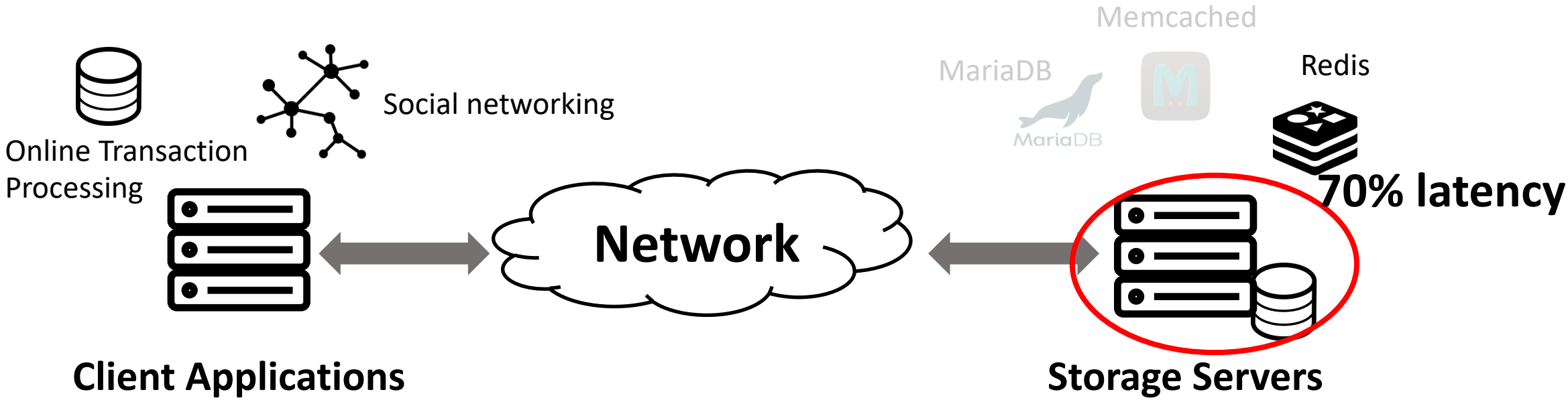
Common datacenter applications store data in separate storage servers



Latency of accessing data on the storage server is critical to the performance of these applications

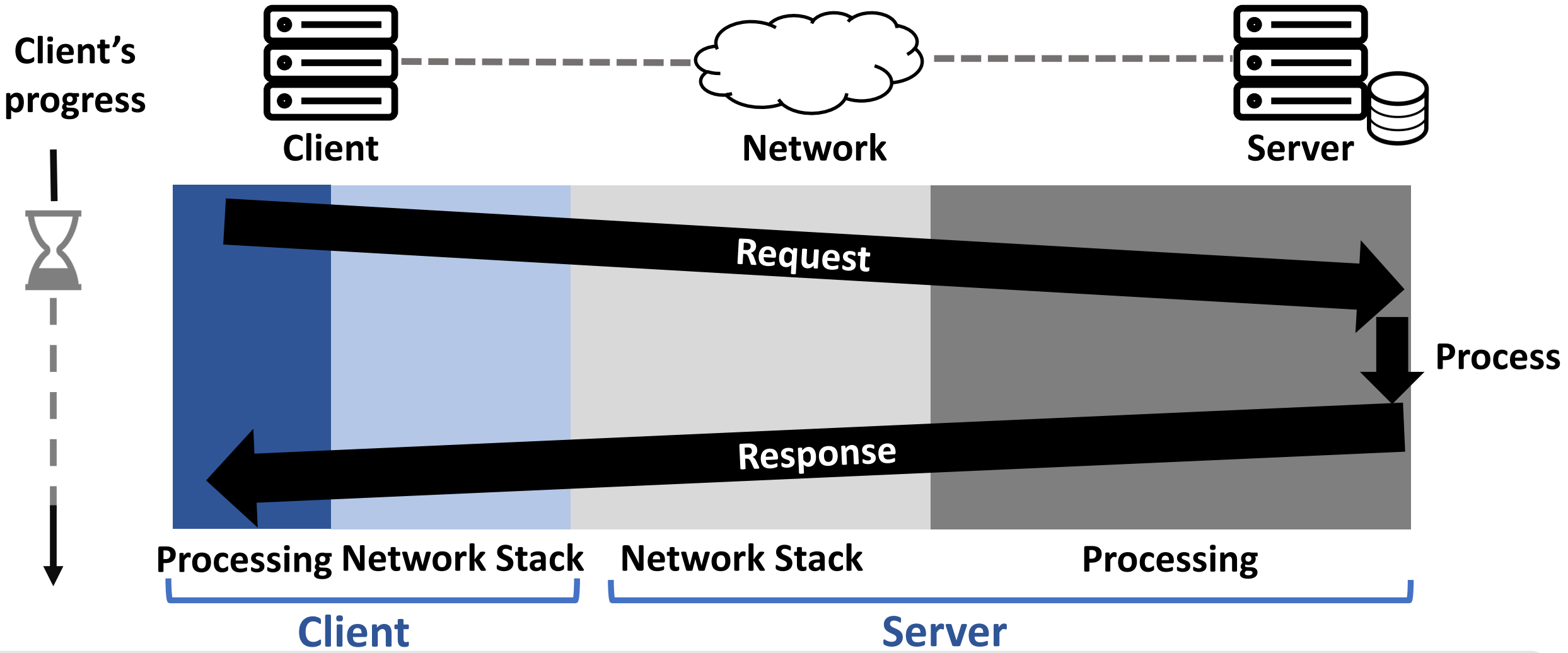
Storage Applications in Datacenter

Common datacenter applications store data in separate storage servers



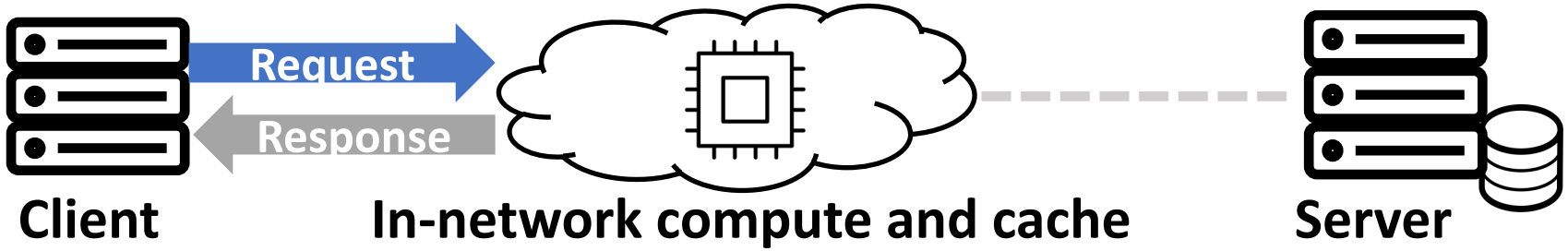
Our breakdown shows that 70% of latency is from **server side**

Latency of Accessing a Data Store



Observation: The client stalls waiting for the response from the server

Mitigating Server-side latency



In-network compute [Brainwave NPU ISCA'18, iSwitch ISCA'19, E3 ATC'19, iPipe SIGCOMM'19]

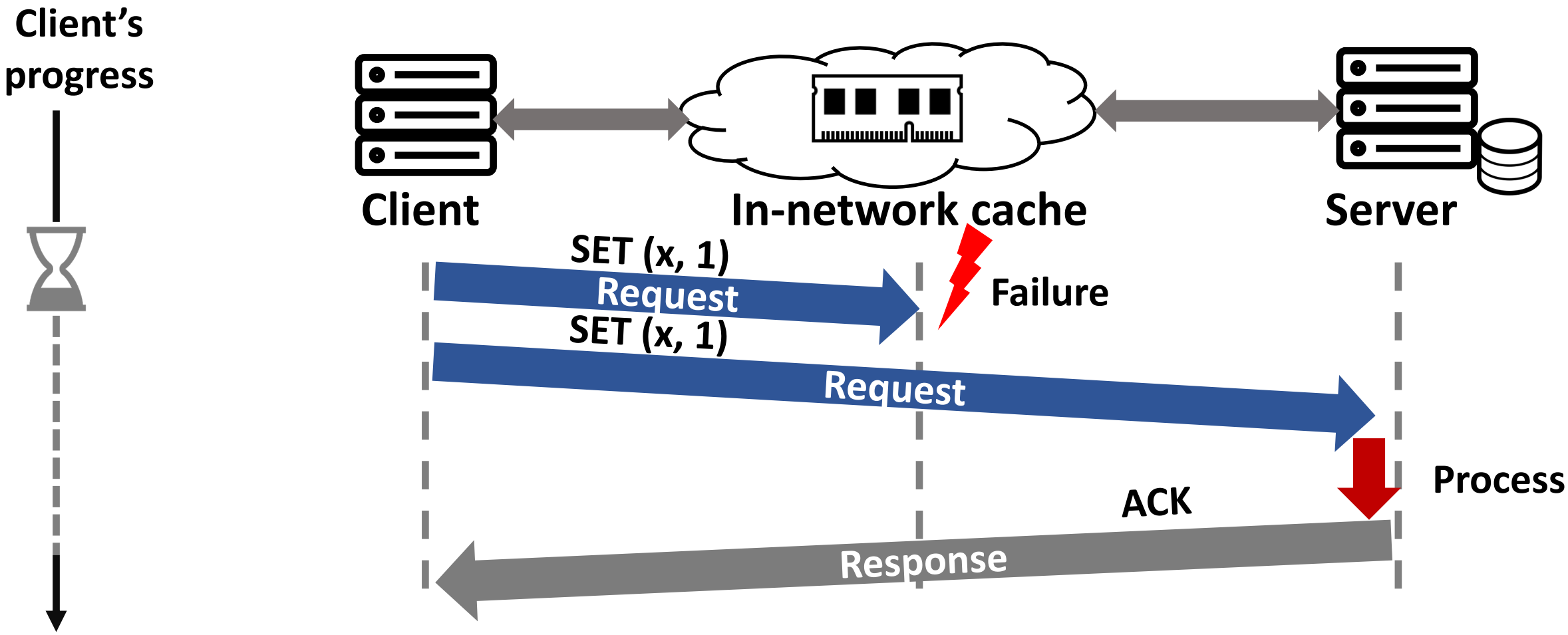
- Add compute logic to network devices, such as switches and NICs
- Reduce RTT of compute tasks

In-network data caching [NetCache SOSP'17, Incbricks SOSP'17, DistCache FAST'19]

- Add volatile cache in network devices
- Reduce RTT of GET requests

These works exploits **programmable network device** to serve read requests
Can we do the same for **update** requests?

In-network Caching: Update Requests



In-network write caching with no **persistent storage** can cause data loss upon failure

Outline

Background and Motivation

In-network Data Persistence

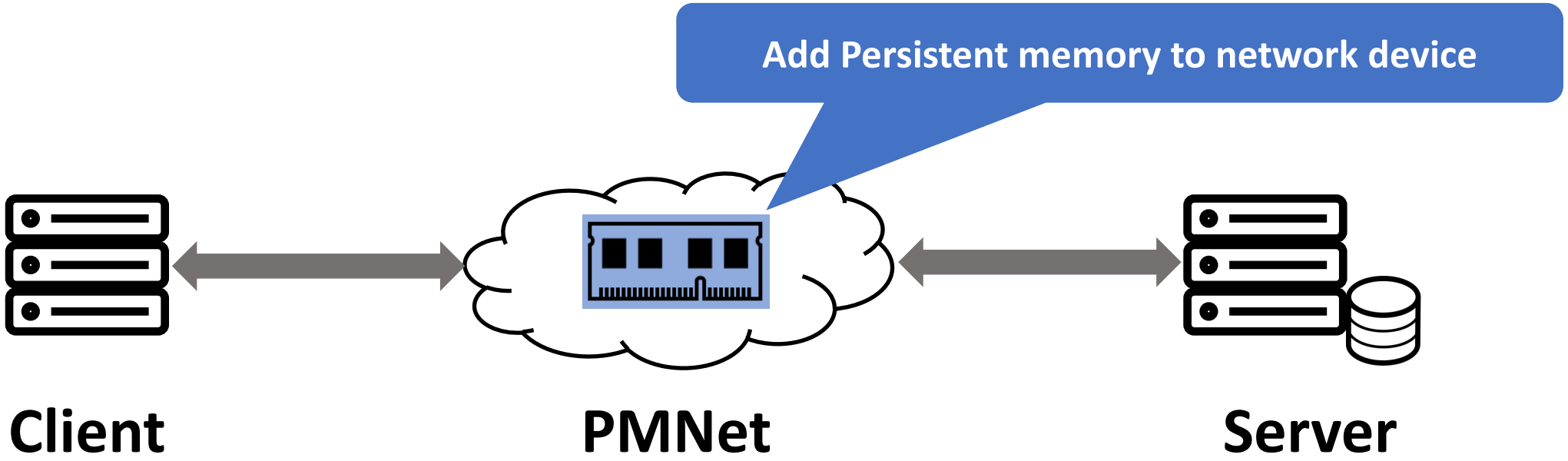
PMNet Design

Caching and Replication

Evaluation

Conclusion

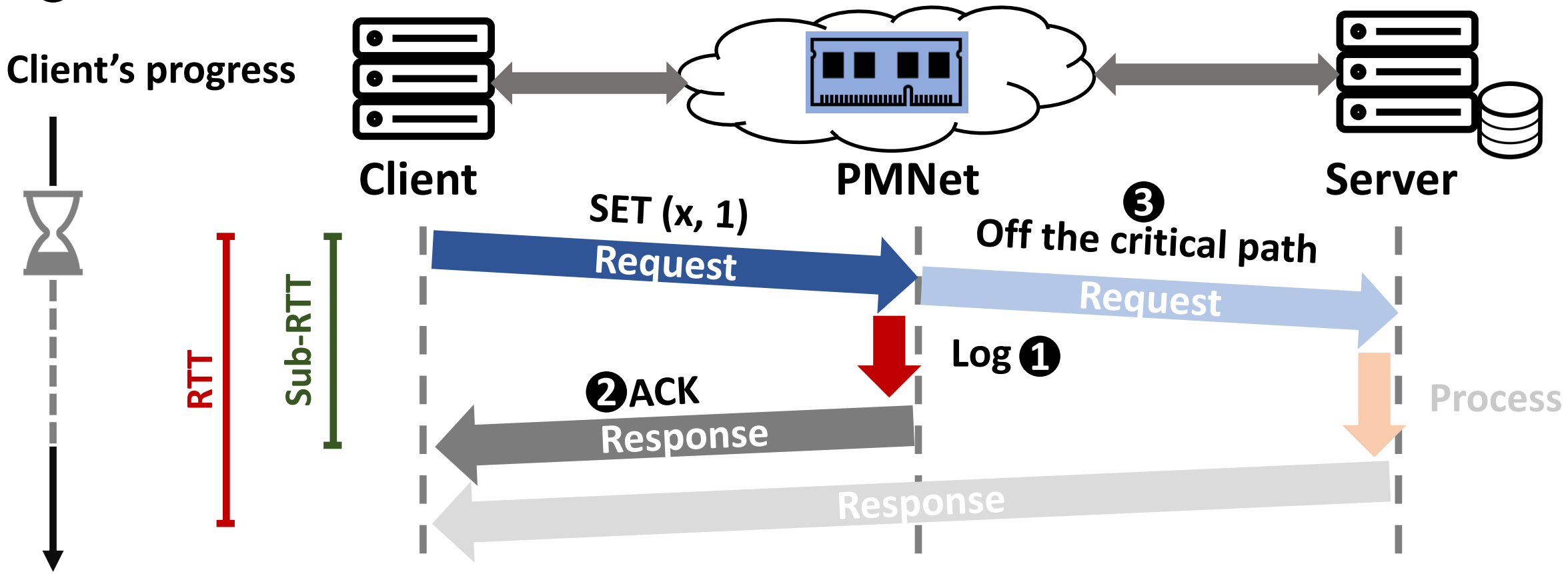
Our Proposal: **In-network Data Persistence**



Add persistent memory to **log** update requests

Key Idea: **Persistent Logging**

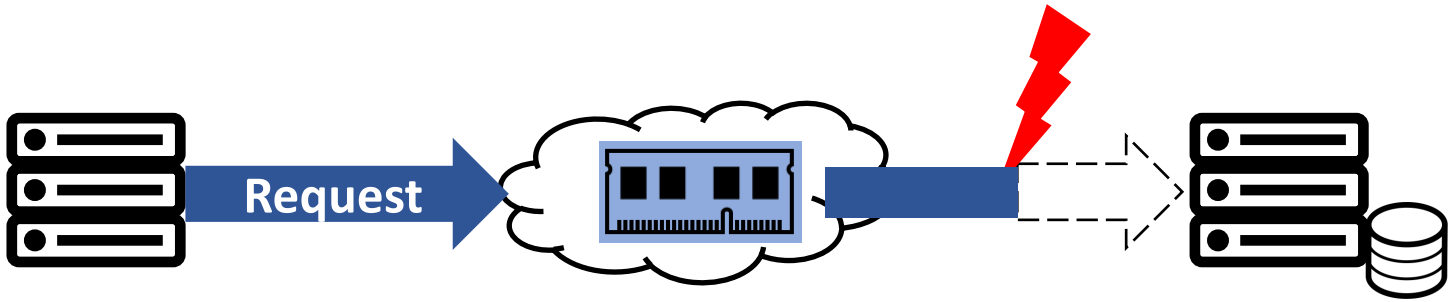
❶ Log request
❷ Send ACK to unblock the client as soon as the update request persists
❸ Forward the request to the server



PMNet enables **sub-RTT** data persistence in the network

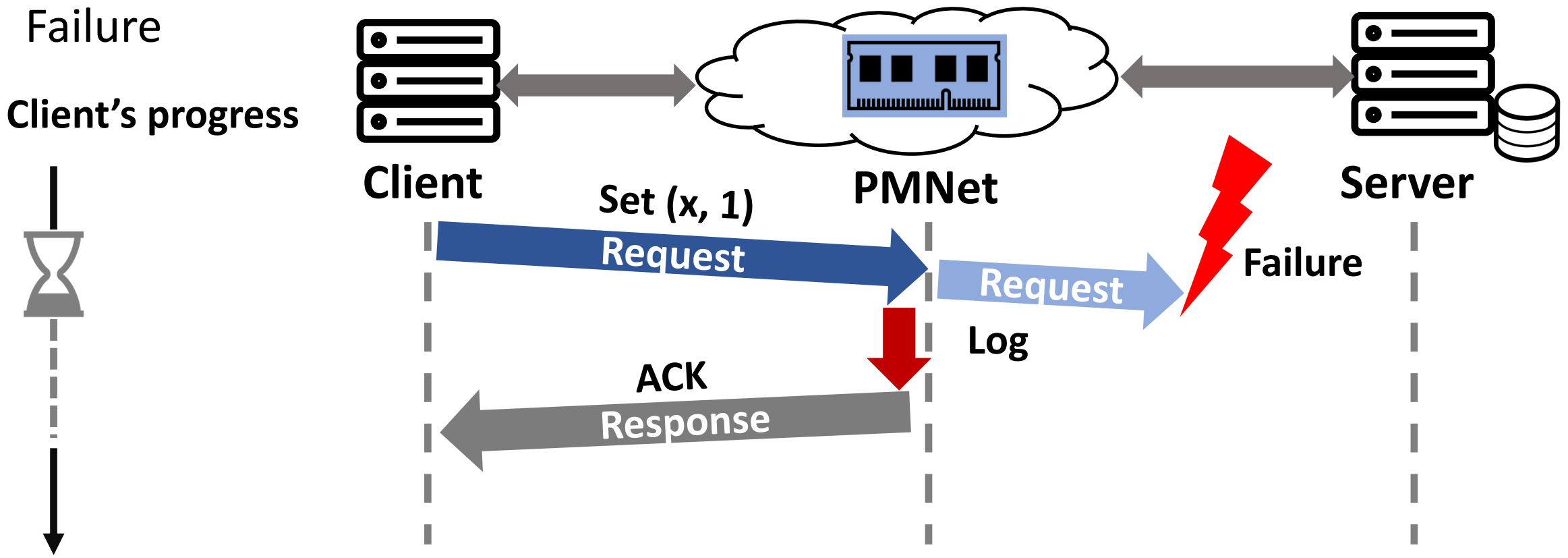
Persistent Logging Challenge

How to recover lost packets?



In-flight requests can be lost due to a crash

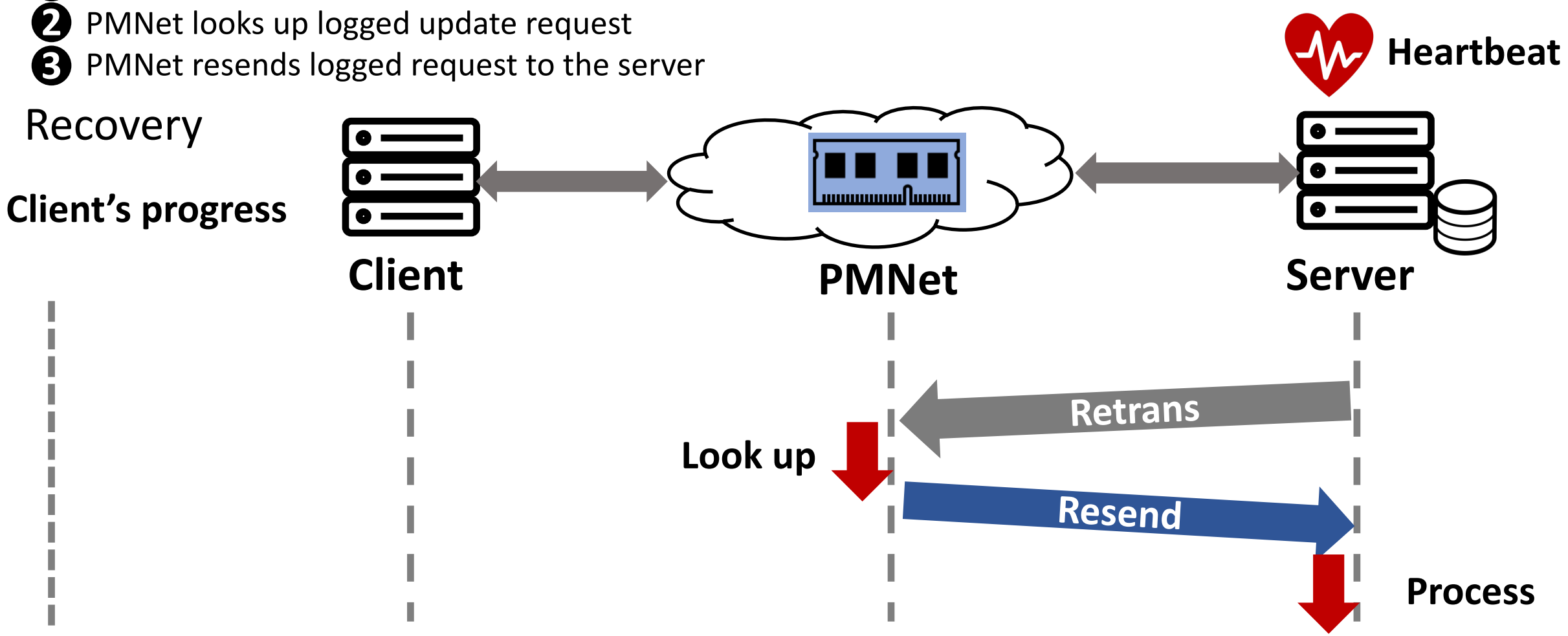
Challenge: **System Recovery**



The client receives ACK and **cannot resend** the request

Solution: **Recover from Persistent Logs**

1. Server sends Retrans
2. PMNet looks up logged update request
3. PMNet resends logged request to the server



PMNet recovers lost requests from **persistent logs**

Outline

Background and Motivation

In-network Data Persistence

PMNet Design

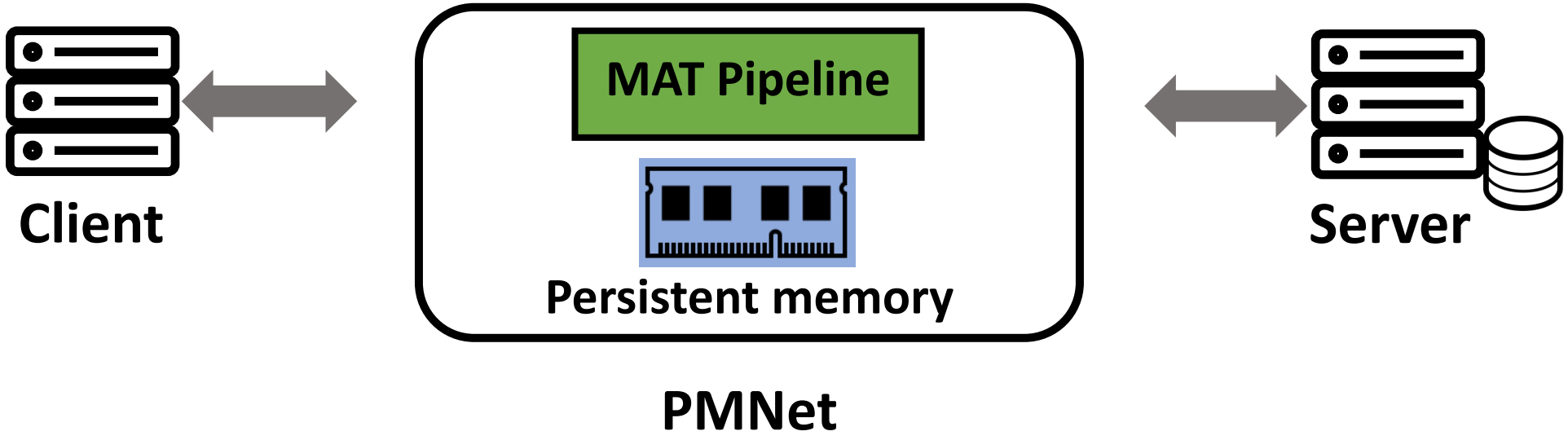
Caching and Replication

Evaluation

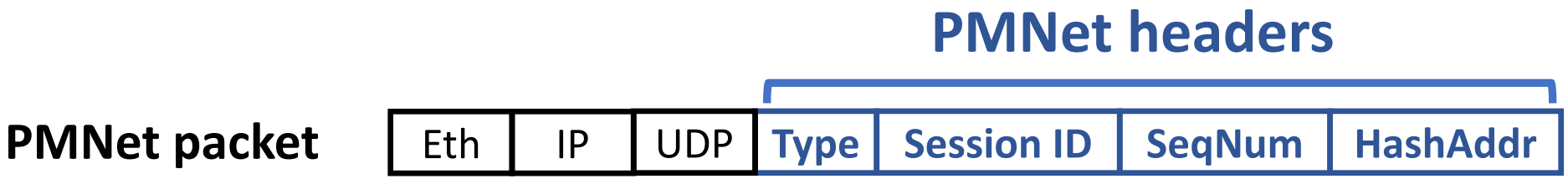
Conclusion

PMNet Design Overview

PMNet hardware: Use Match-Action Table (MAT) to control persistent memory

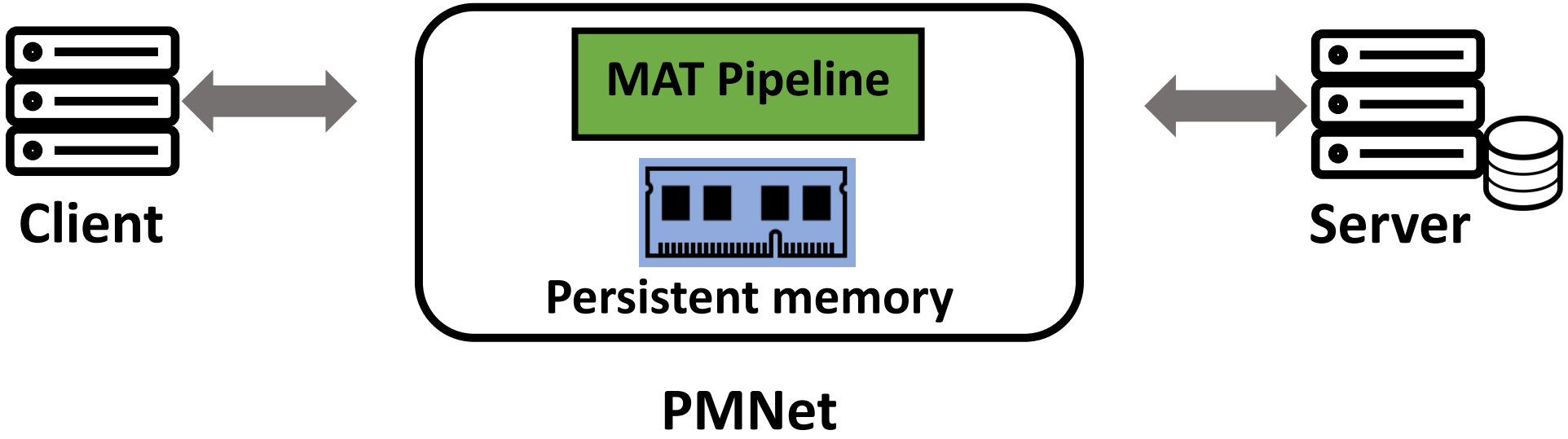


PMNet protocol: Identify PMNet packets and trigger processing



PMNet Design Overview

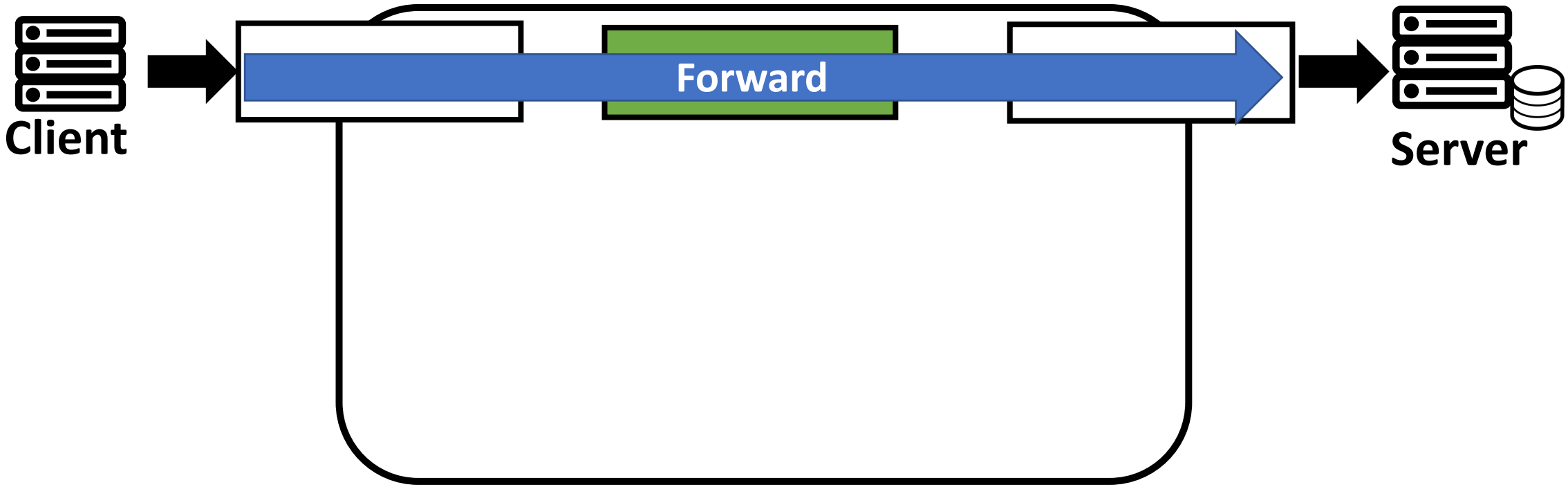
PMNet hardware: Use Match-Action Table (MAT) to control persistent memory



PMNet protocol: Identify PMNet packets and trigger processing

Baseline NIC/Switch Architecture

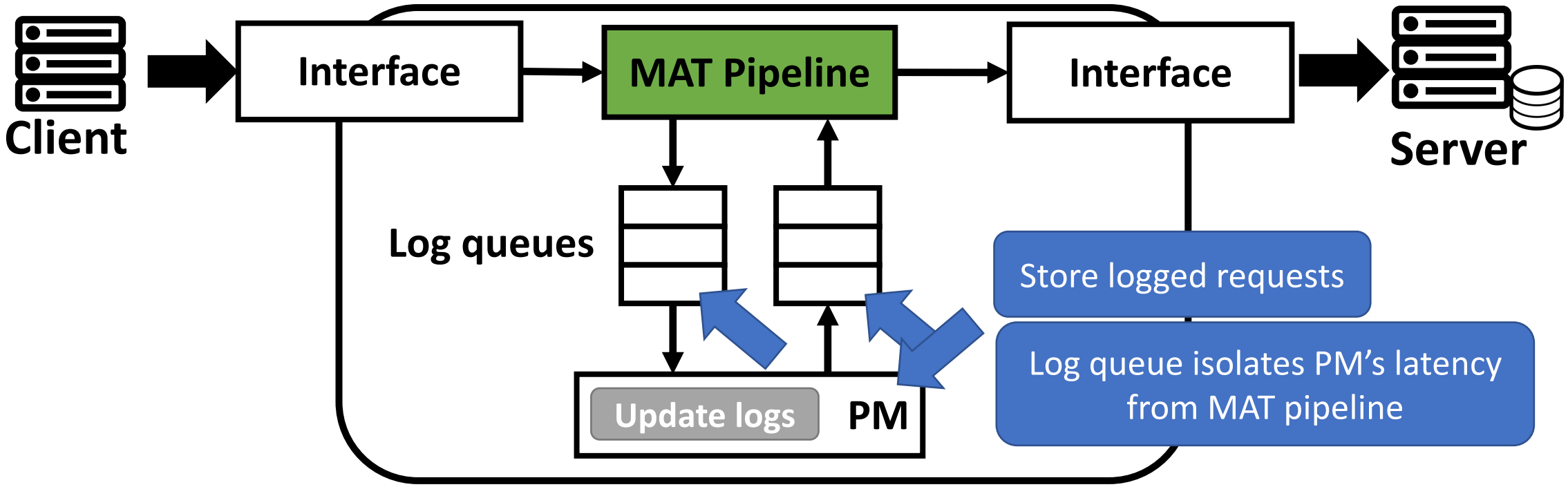
Baseline NIC and switch forward packet with rules in Match-action table (MAT) pipeline



Baseline NIC and switch **forwards** packet to the destination

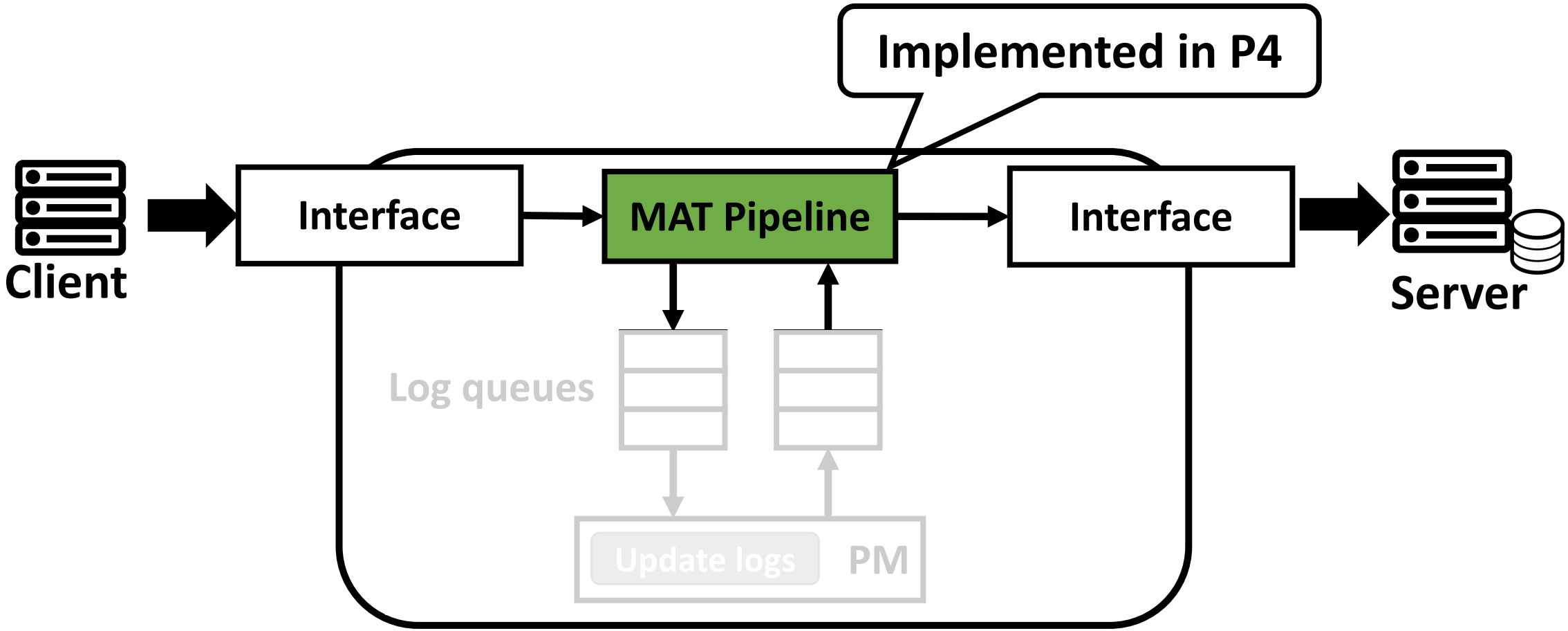
PMNet NIC/Switch Architecture

PMNet NIC and switch's MAT pipeline process PMNet packet in addition to other packets



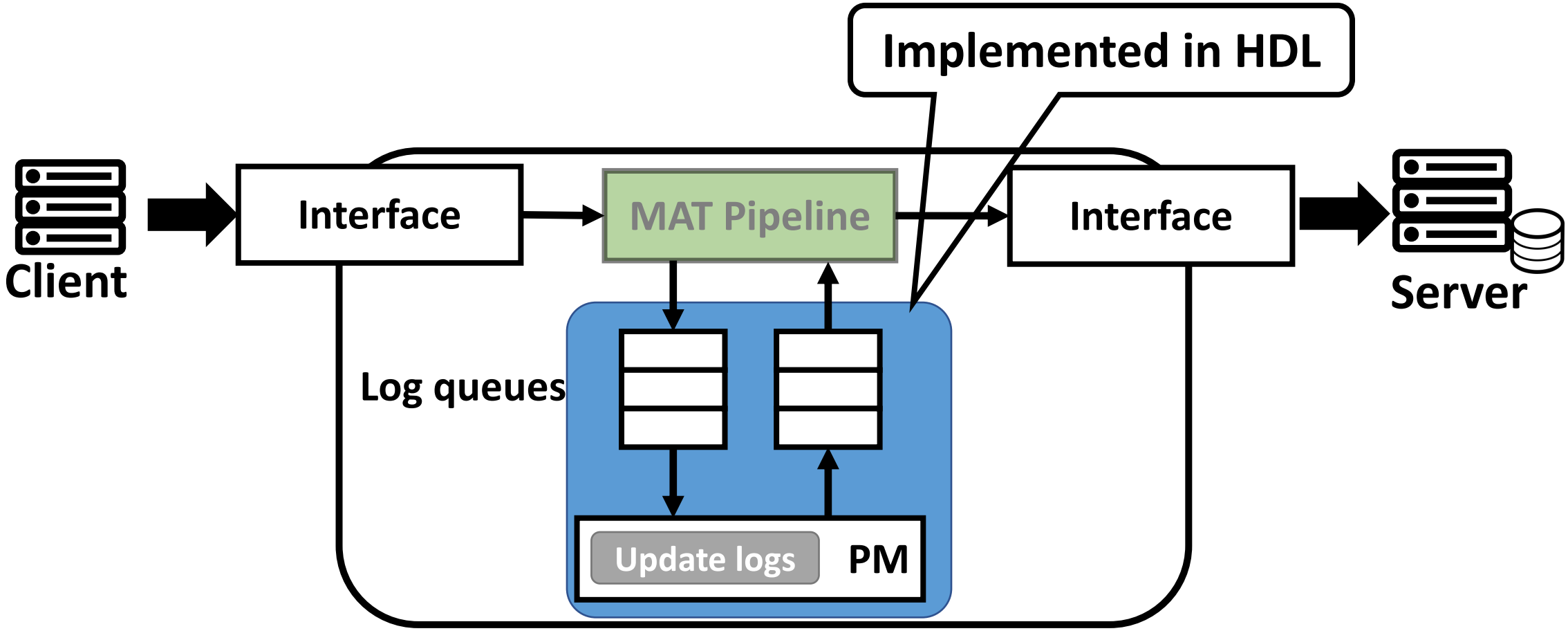
PMNet MAT Pipeline controls access to the **persistent memory**

PMNet NIC/Switch Hardware Design



PMNet MAT Pipeline is **compatible with P4-supported network devices**

PMNet NIC/Switch Hardware Design

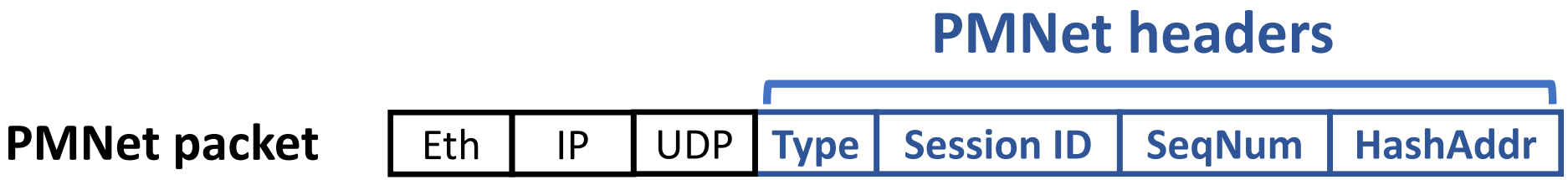


Request queue and memory controller are implemented in HDL
due to **lack of persistent storage support**

PMNet Design Overview

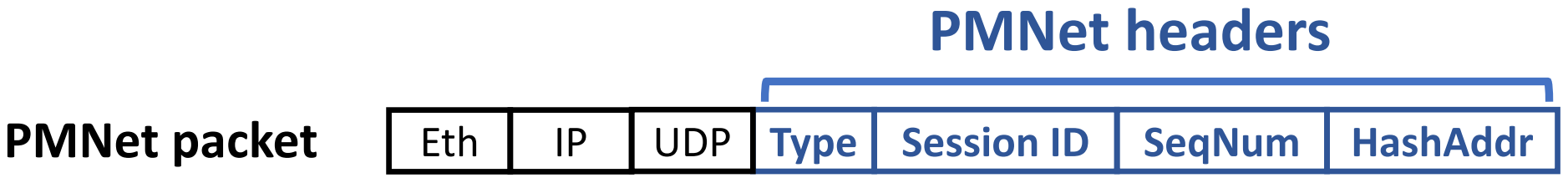
PMNet hardware: Use Match-Action Table (MAT) to control persistent memory

PMNet protocol: Identify PMNet packets and trigger processing



PMNet Design: Protocol

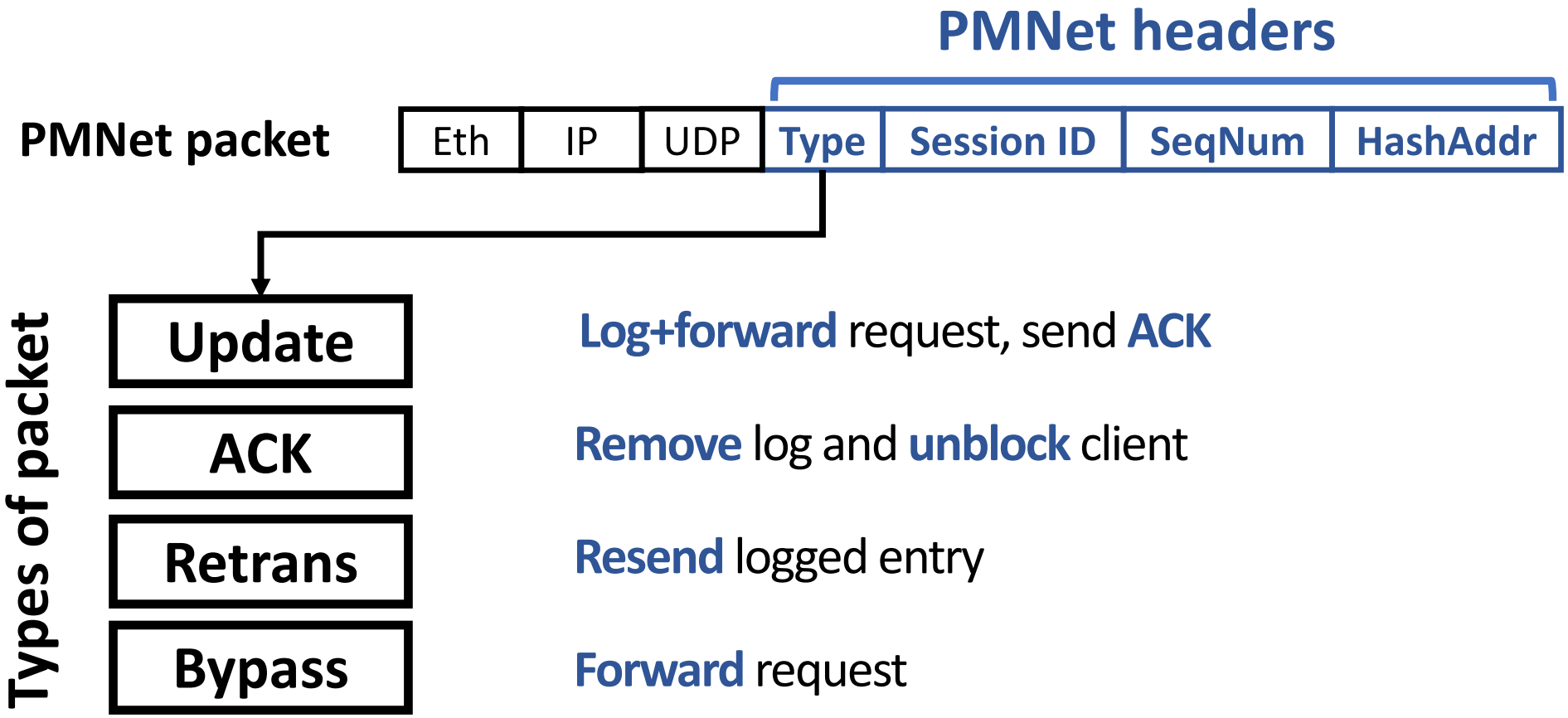
Defines four packet types on top of UDP




```
header pmnethds_h {
    bit<8>  type;
    bit<16> session_id;
    bit<32> seq_no;
    bit<32> hash_addr;
}
```

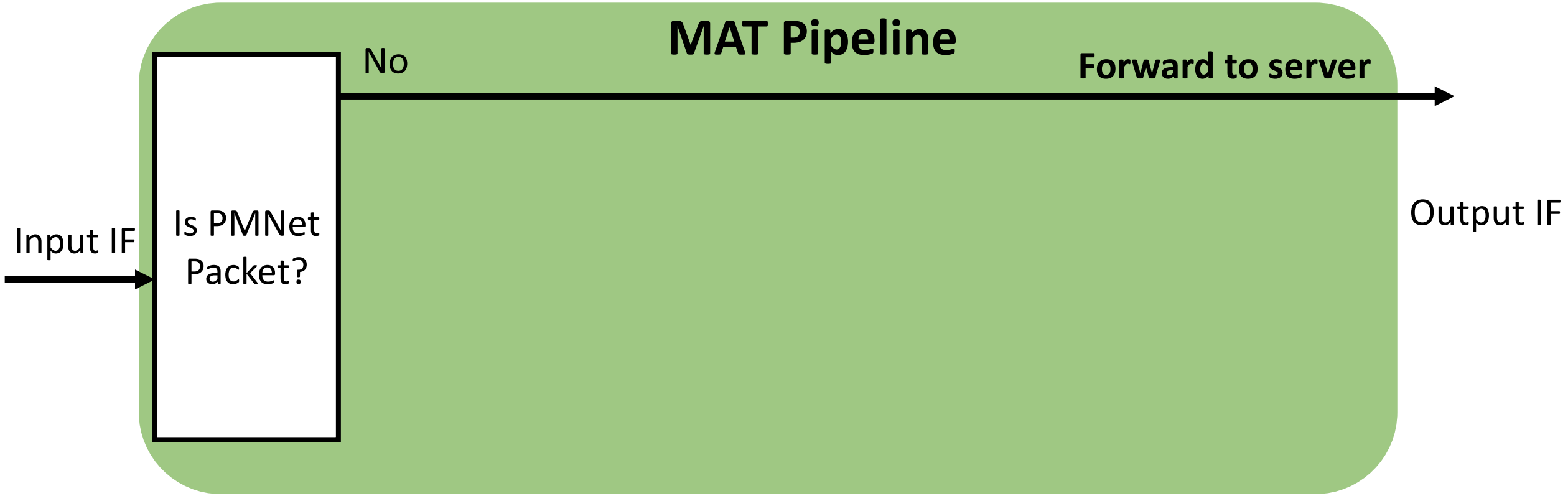
PMNet Design: Protocol

Defines four packet types on top of UDP



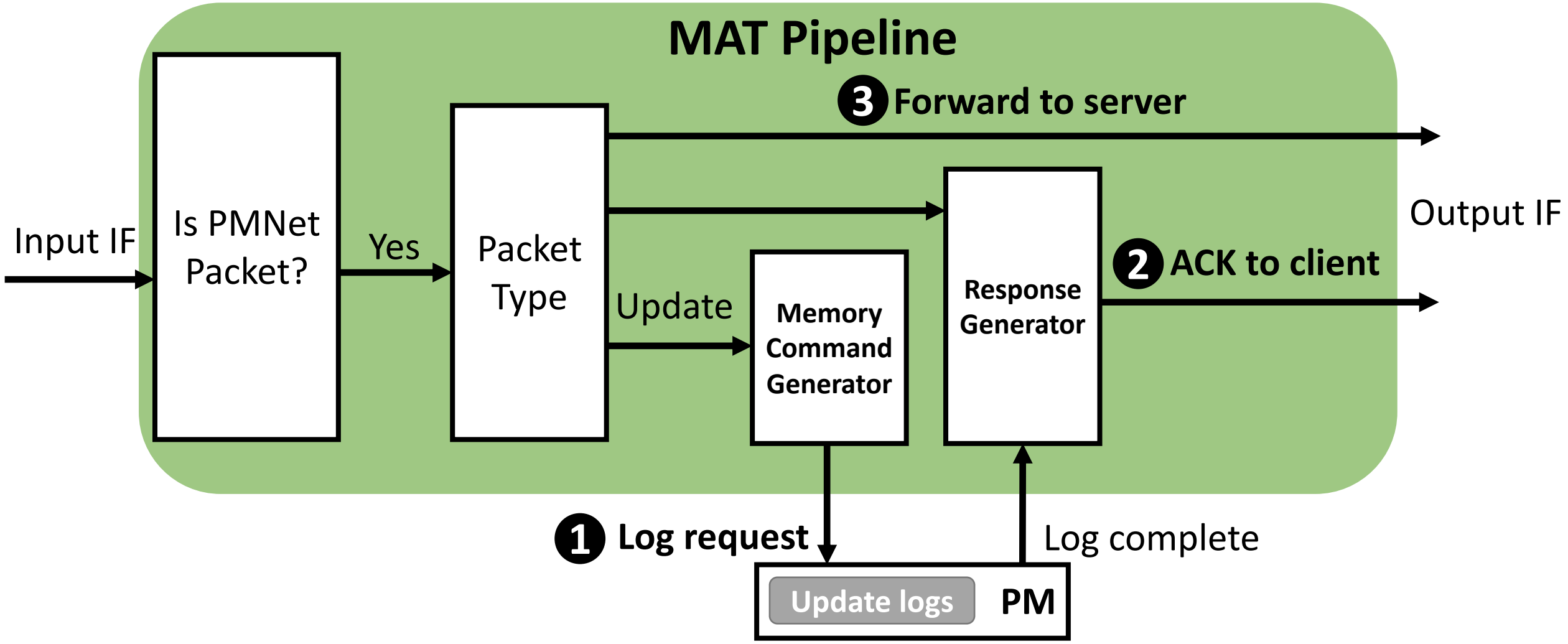
PMNet performs different operation based on **packet type**

PMNet packet processing: Non-PMNet packets



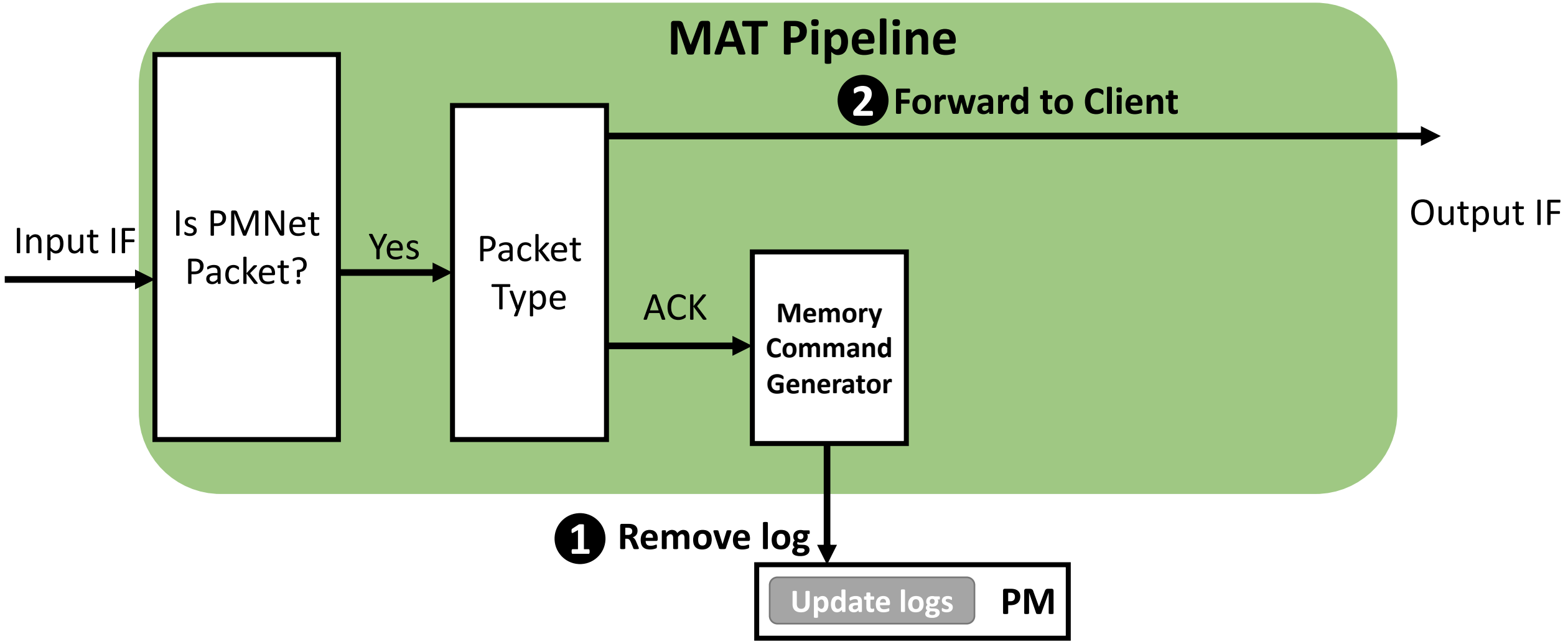
PMNet **forwards** Non-PMNet packets

PMNet packet processing: Update requests



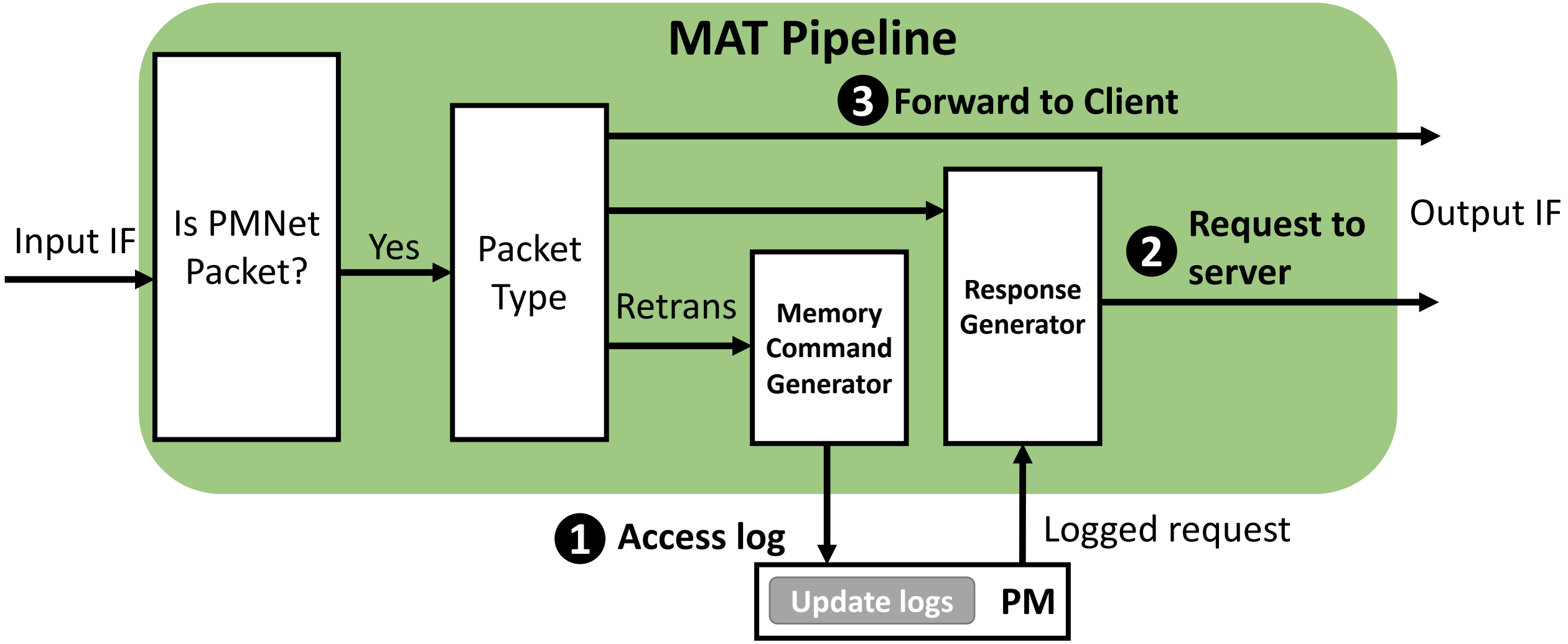
PMNet **logs and forwards** update requests and sends **ACK** to unblock client

PMNet packet processing: ACK



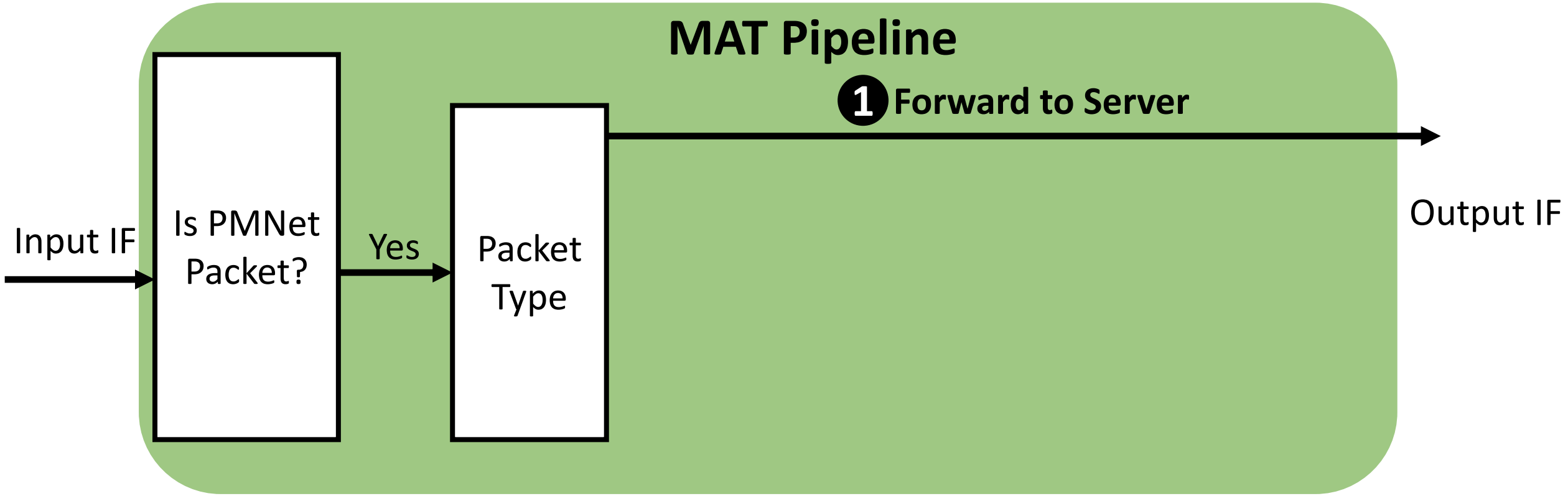
PMNet ACK **removes** the logged entry

PMNet packet processing: Retrans



PMNet Retrans **resends** the logged request

PMNet packet processing: Bypass



PMNet **logs and forwards** update requests and sends **ACK** to unblock client

Outline

Background and Motivation

In-network Data Persistence

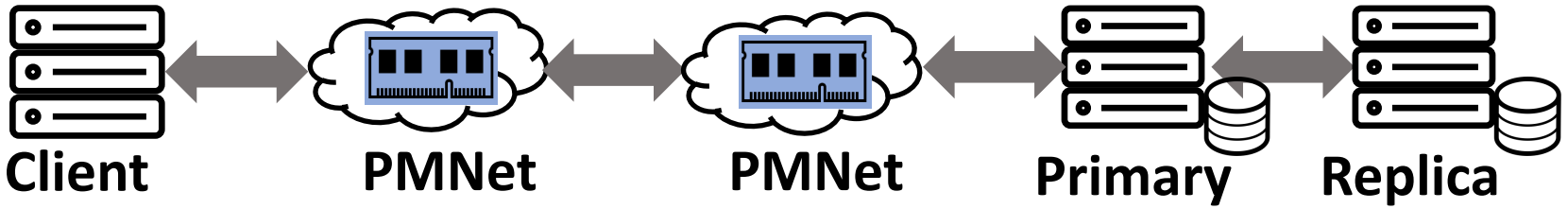
PMNet Design

Caching and Replication

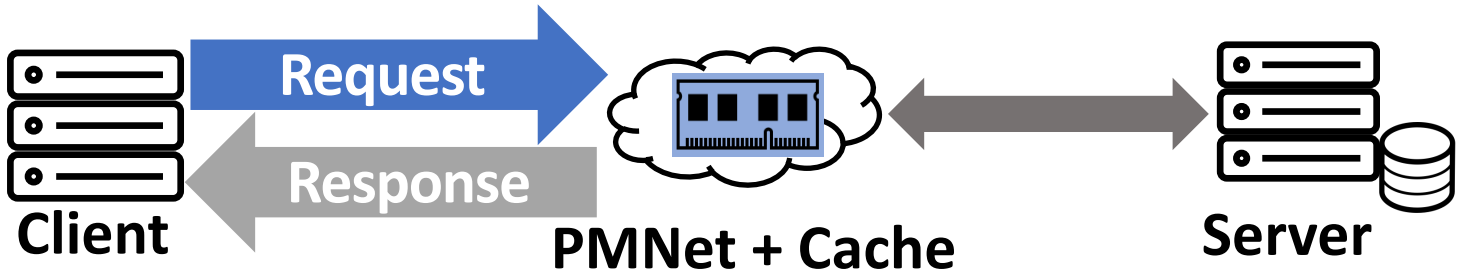
Evaluation

Conclusion

PMNet Use-cases

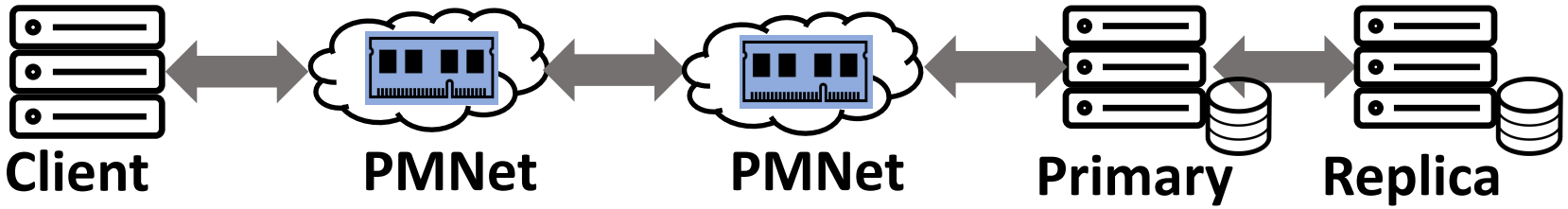


Replication



Caching

PMNet Use-cases



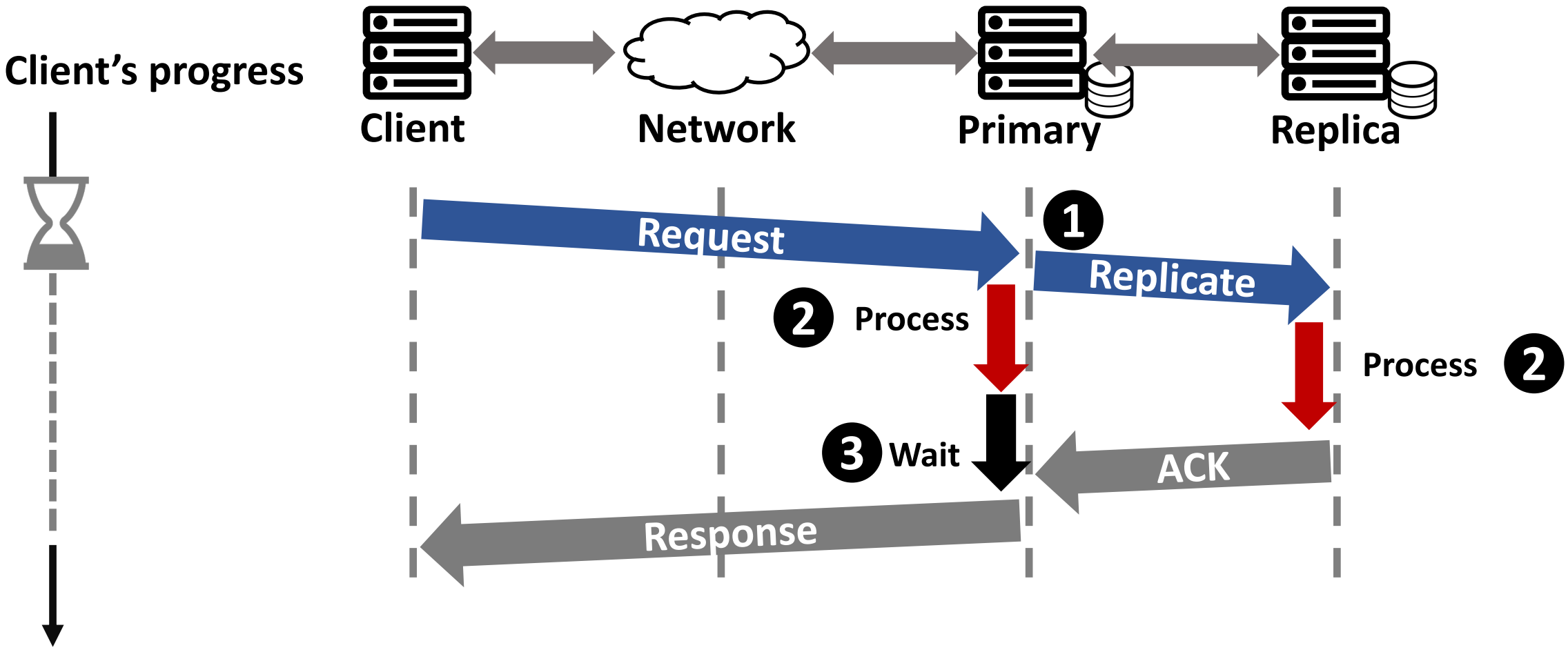
Replication

Caching

32

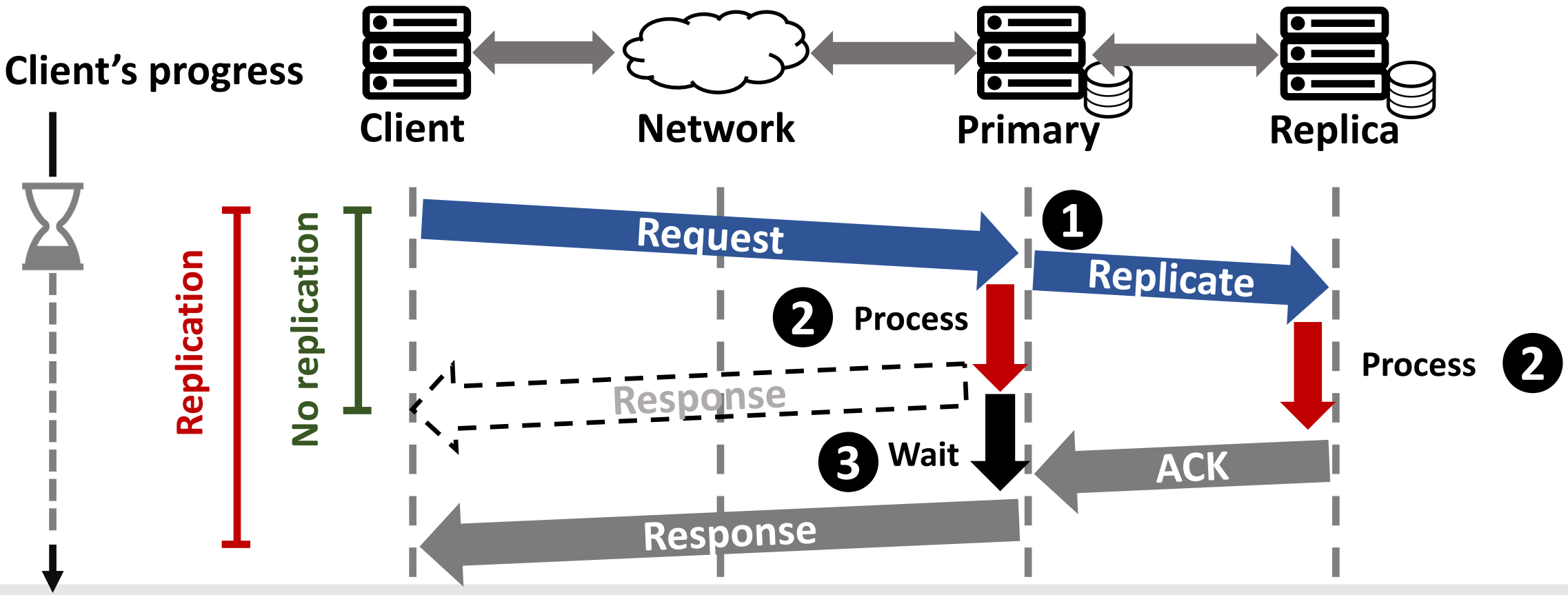
PMNet Replication: Baseline Replication

1. Replicate request to all servers
2. Process the request
3. Wait until all servers respond



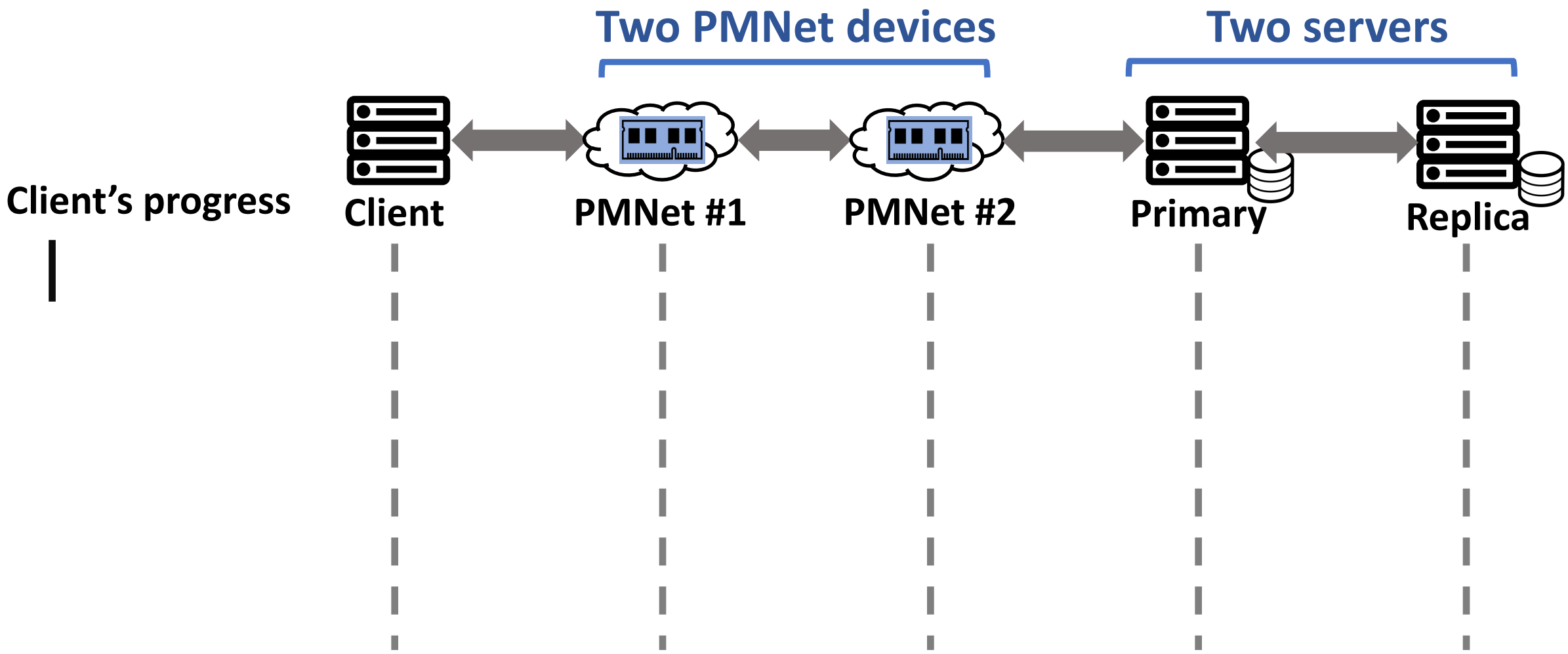
PMNet Replication: Baseline Replication

1. Replicate request to all servers
2. Process the request
3. Wait until all servers respond



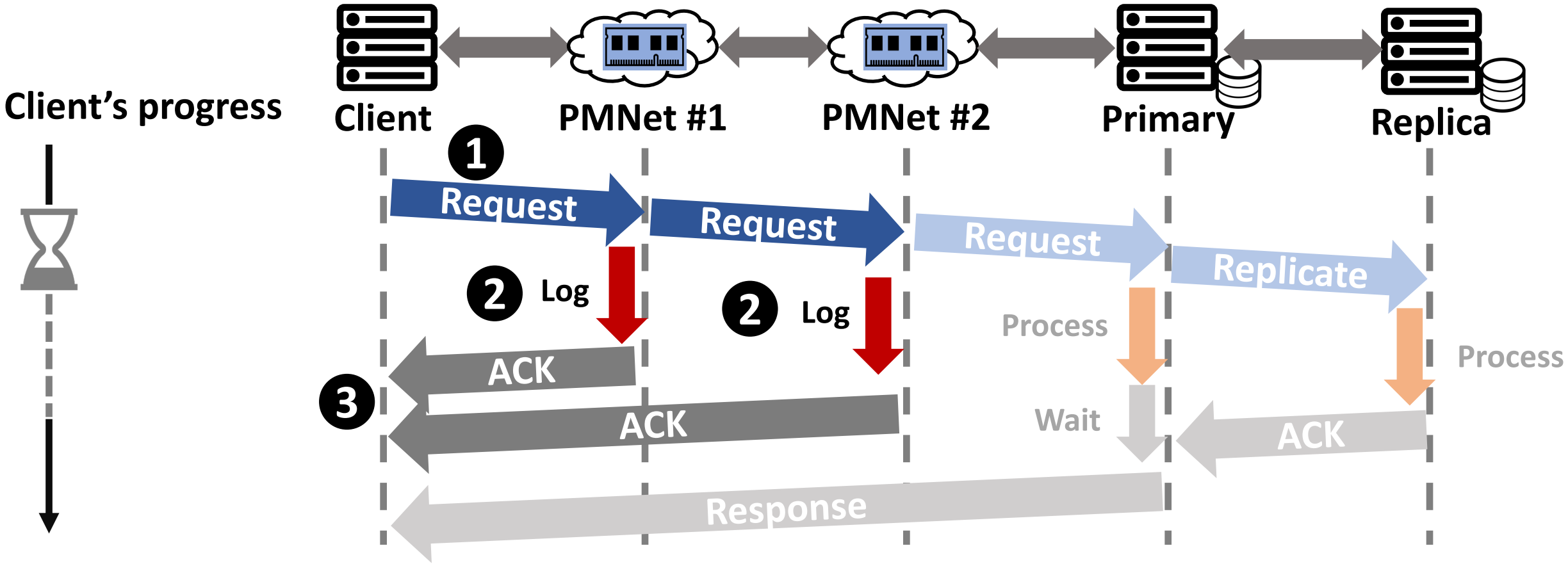
Replication increases **blocking latency**

PMNet Replication: Replication with PMNet



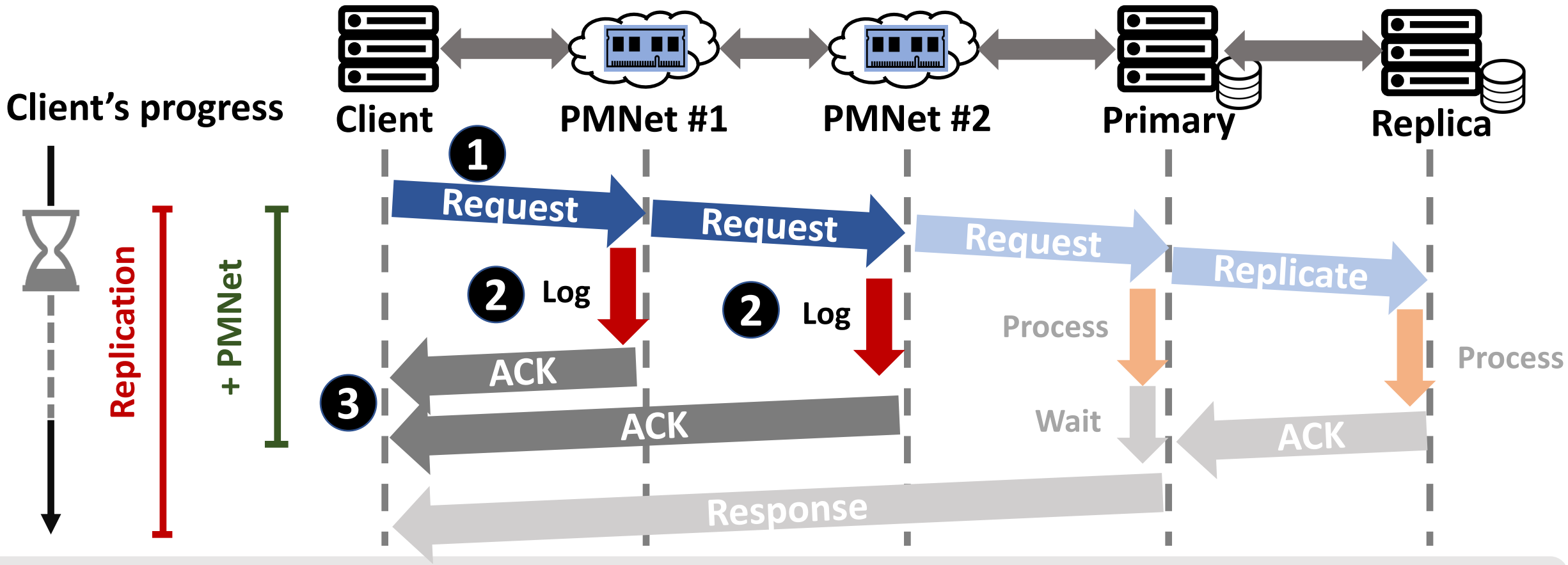
PMNet Replication: Replication with PMNet

1. The client sends the request and waits for 2 ACKs
2. PMNet #1 and #2 log the request and send ACK to the client
3. Client waits until it receives both ACKs



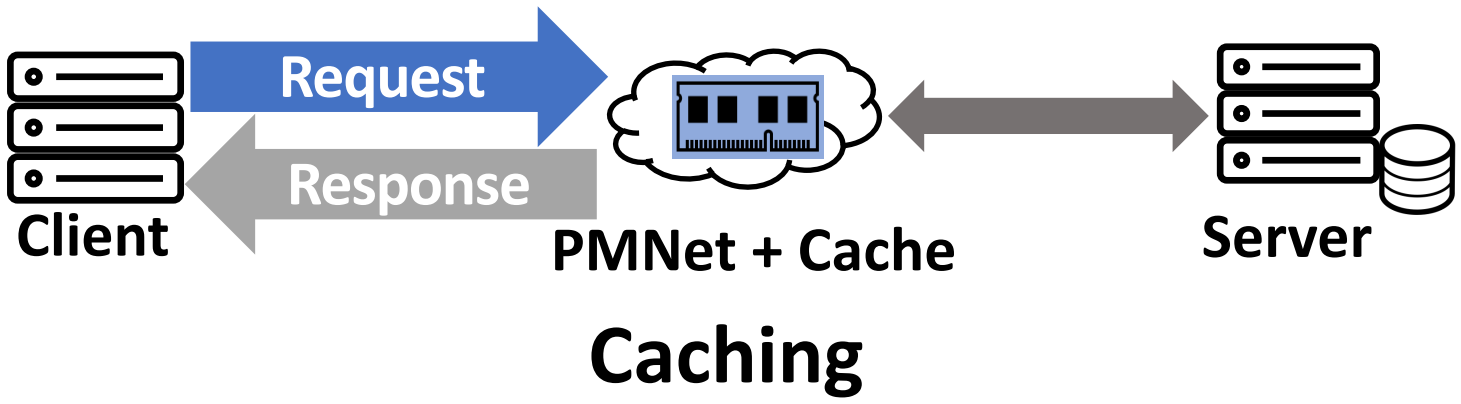
PMNet Replication: Replication with PMNet

1. The client sends the request and waits for 2 ACKs
2. PMNet #1 and #2 log the request and send ACK to the client
3. Client waits until it receives both ACKs



PMNet exploits **persistent logging** to move replication off the critical path

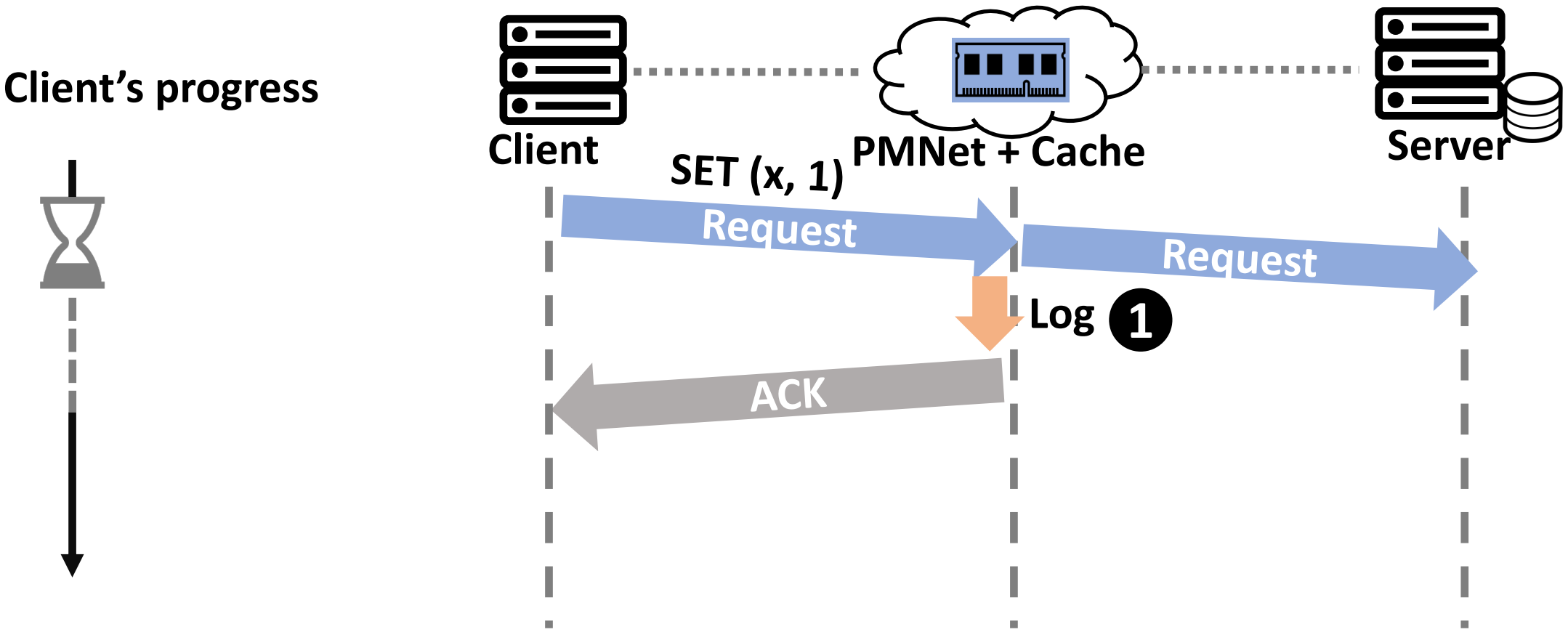
PMNet Use-cases



38

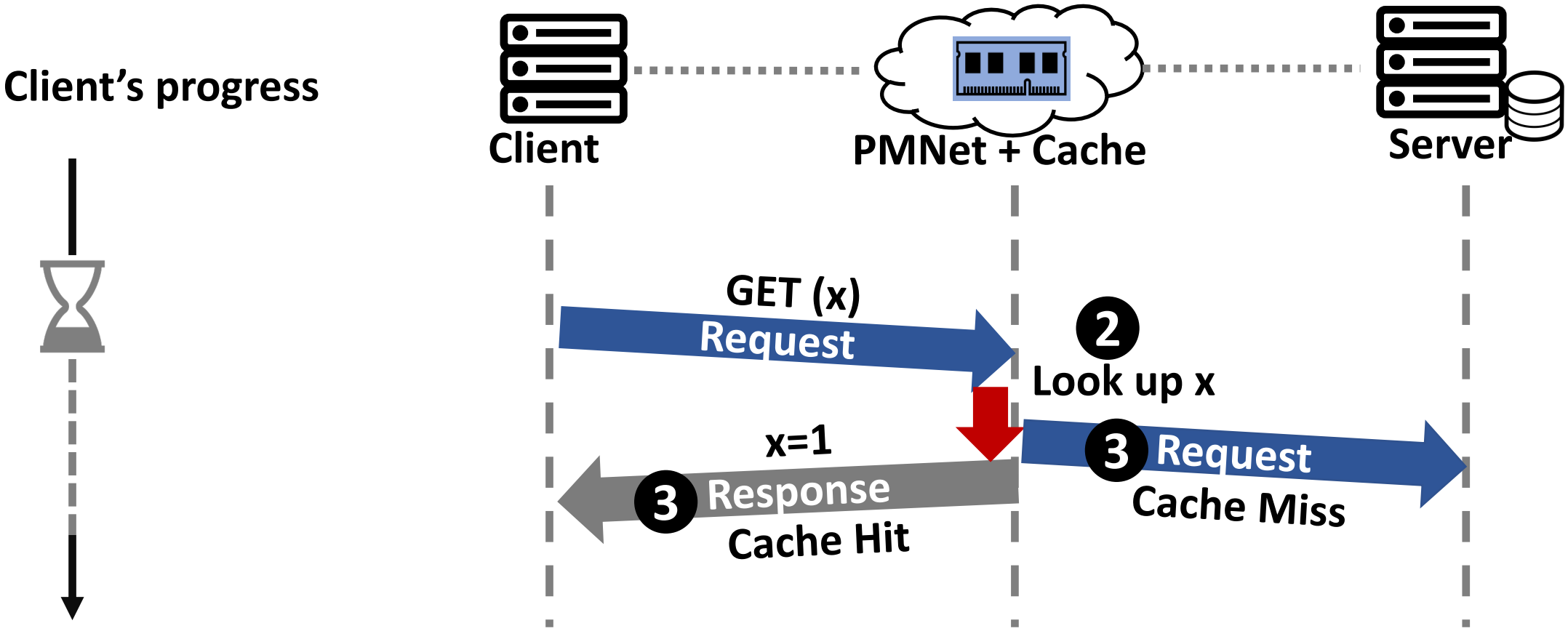
In-network Caching: Update Requests

❶ PMNet logs update requests



In-network Caching: Read Requests

1. PMNet logs update requests
2. PMNet receives read request and looks up an associated logged request in the PM
3. PMNet responds the read request (Hit) or forward the request (Miss)



PMNet exploits **logged entry** and **recovery** mechanism to respond **read** requests.

Outline

Background and Motivation

In-network Data Persistence

PMNet Design

Caching and Replication

Evaluation

Conclusion

Methodology

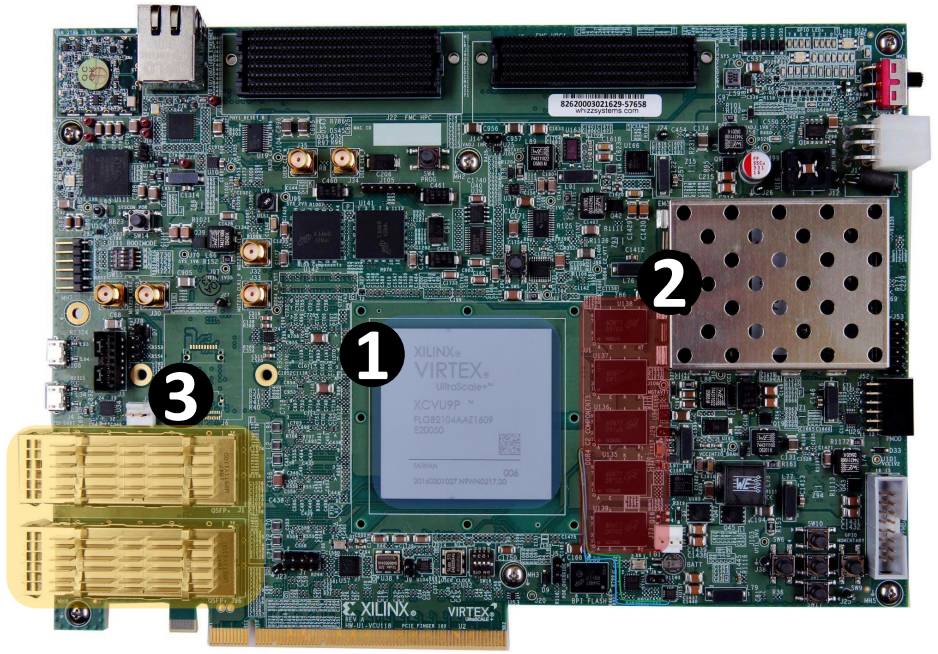
Hardware

PMNet Xilinx VCU118 Evaluation platform

Server Intel Cascade Lake, 20 Cores, 192GB DRAM, **256GB DCPMM**

Client Intel Haswell, 6 Cores, 64GB DRAM

Persistent memory on the server



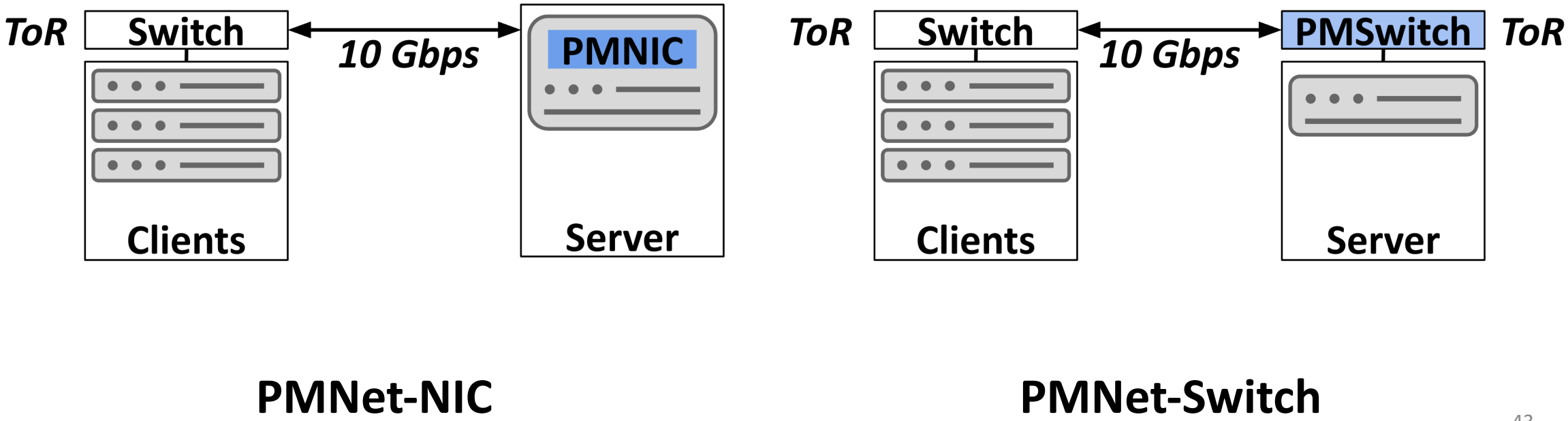
PMNet evaluation platform

❶ MAT pipeline (P4)
❷ Emulated persistent memory
❸ Network interfaces

Methodology

Design points

- **PMNet-Switch**: PMNet as a bump-in-the-wire in the TOR switch of server rack
- **PMNet-NIC**: PMNet as a bump-in-the-wire in the server's NIC
- **Client-Server**: A baseline design that only persists update requests on the server



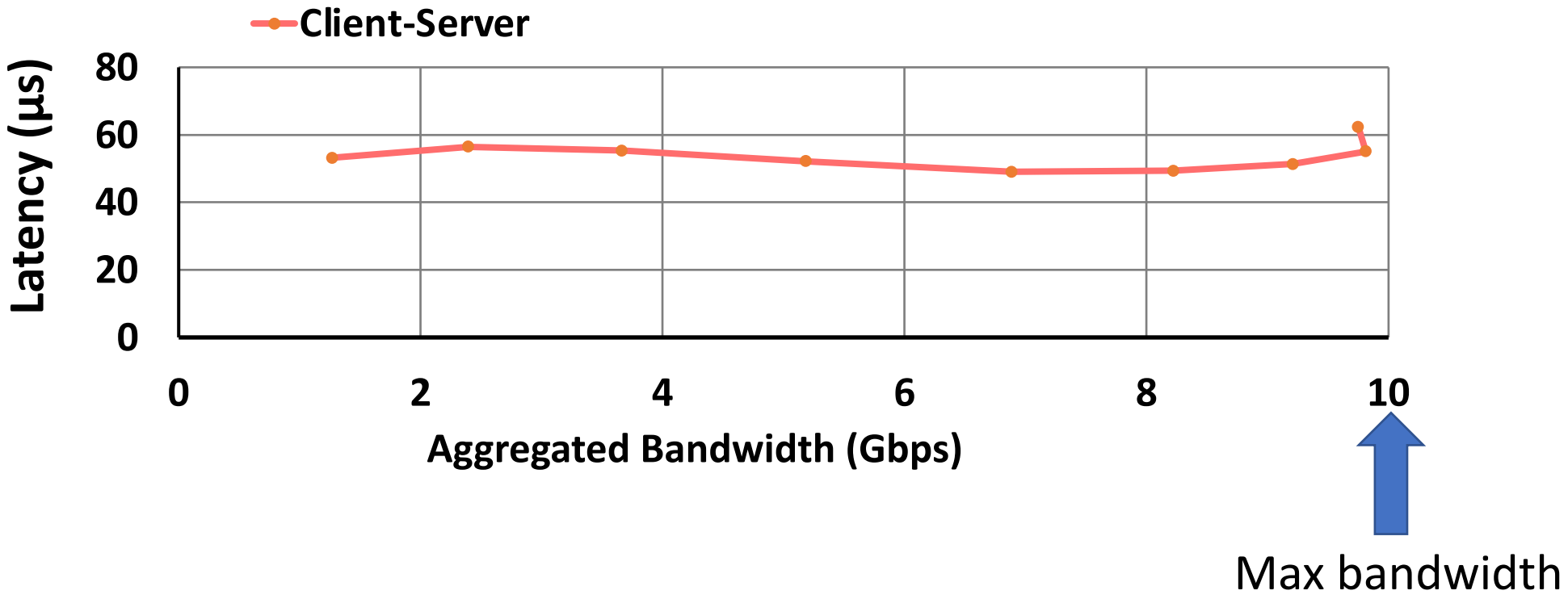
Methodology

Workloads

- **Microbenchmarks**: Empty request handler on the server
- **Persistent-memory-optimized datastore workloads**:
  - PMDK-based key-value stores:
    B-tree, C-tree, RB-tree, Hashmap, Skiplist
  - Redis: TPC-C, Twitter clone

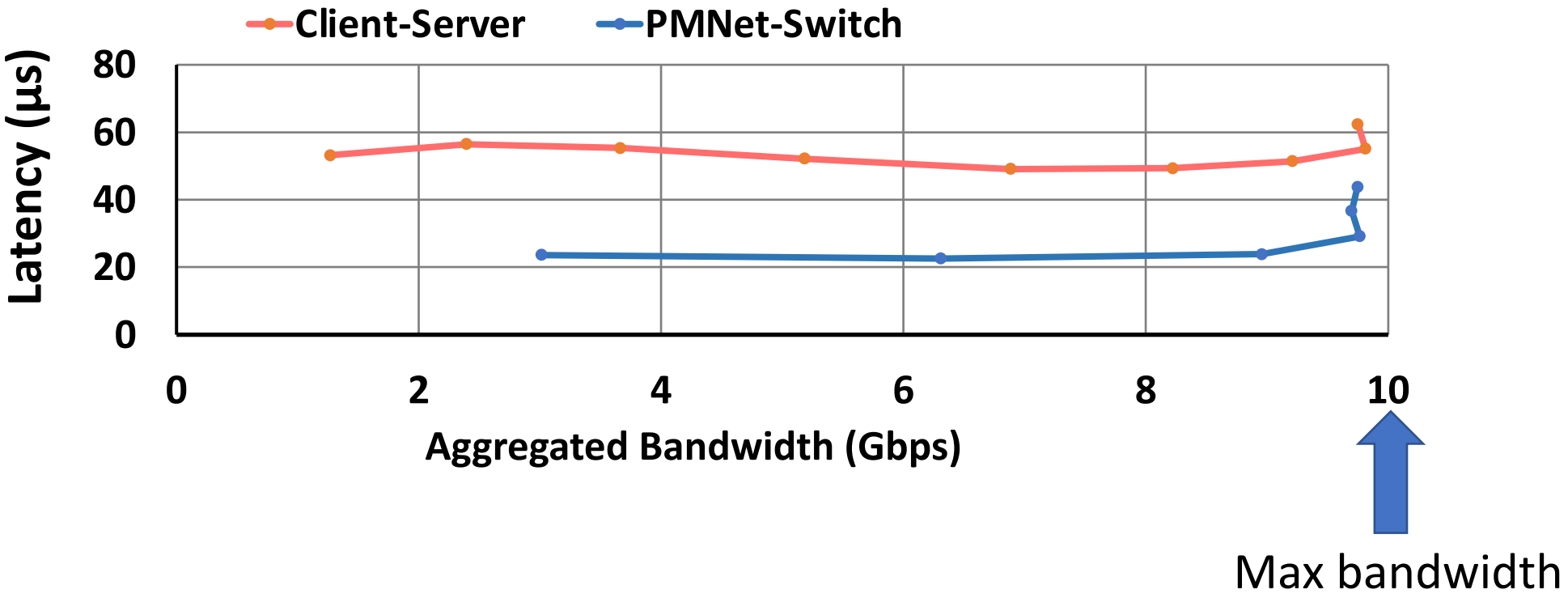
Results: Microbenchmarks

Update request Bandwidth vs. Latency



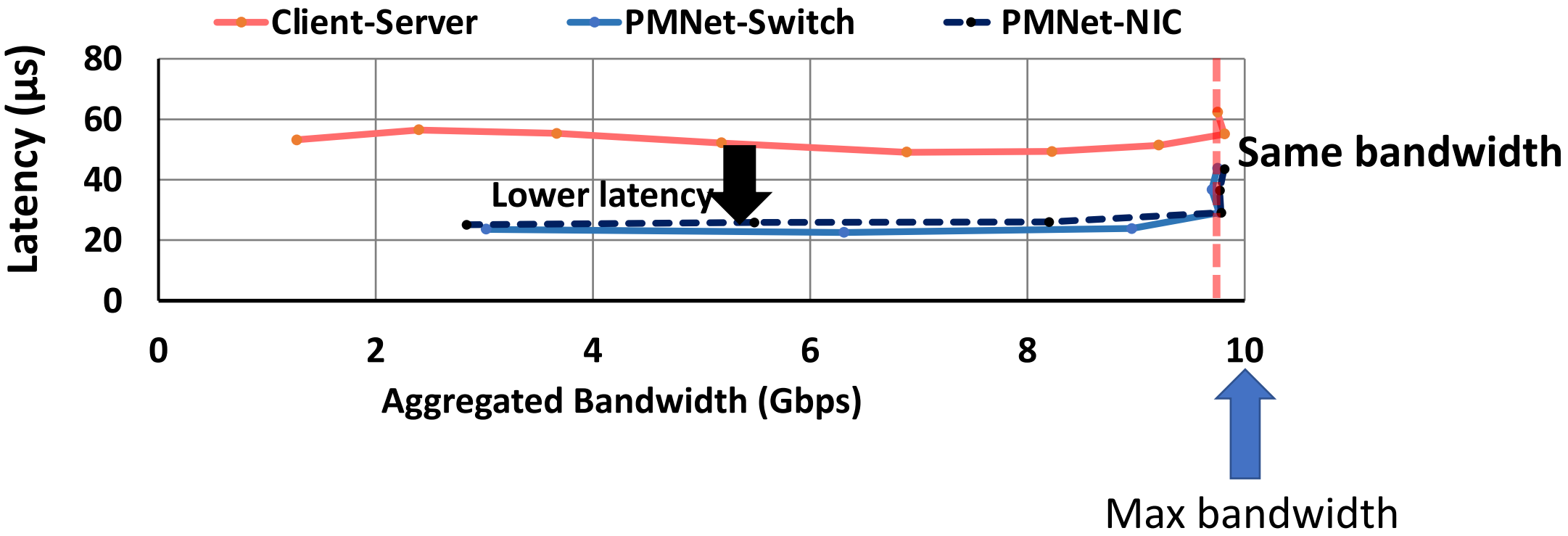
Results: Microbenchmarks

Update request Bandwidth vs. Latency



Results: Microbenchmarks

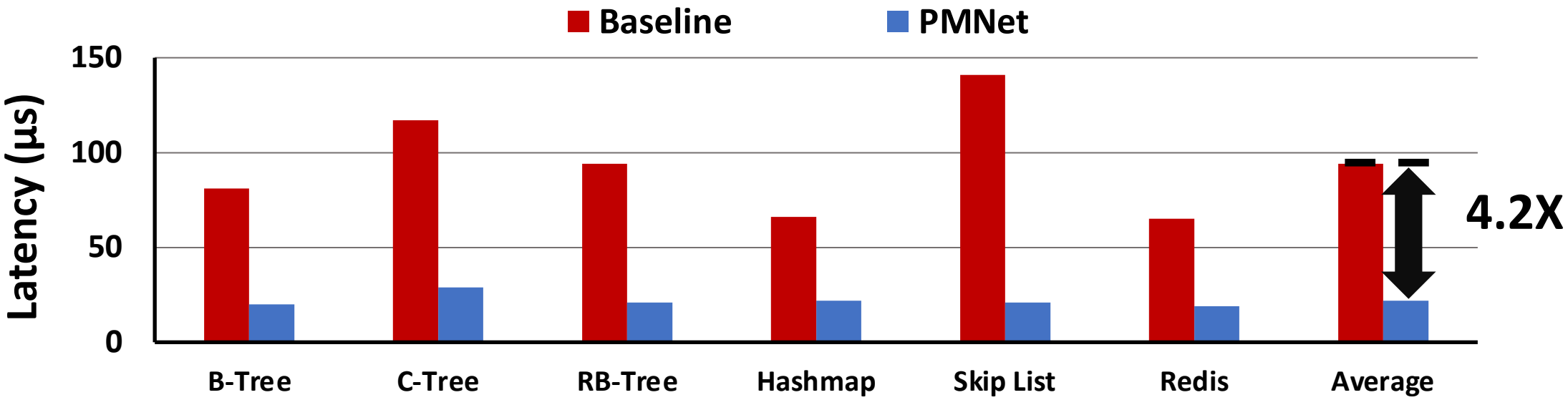
Update request Bandwidth vs. Latency



Both PMNet-Switch and PMNet-NIC provide **lower update latency** and **same bandwidth** as the baseline.

Results: Key-value store workloads

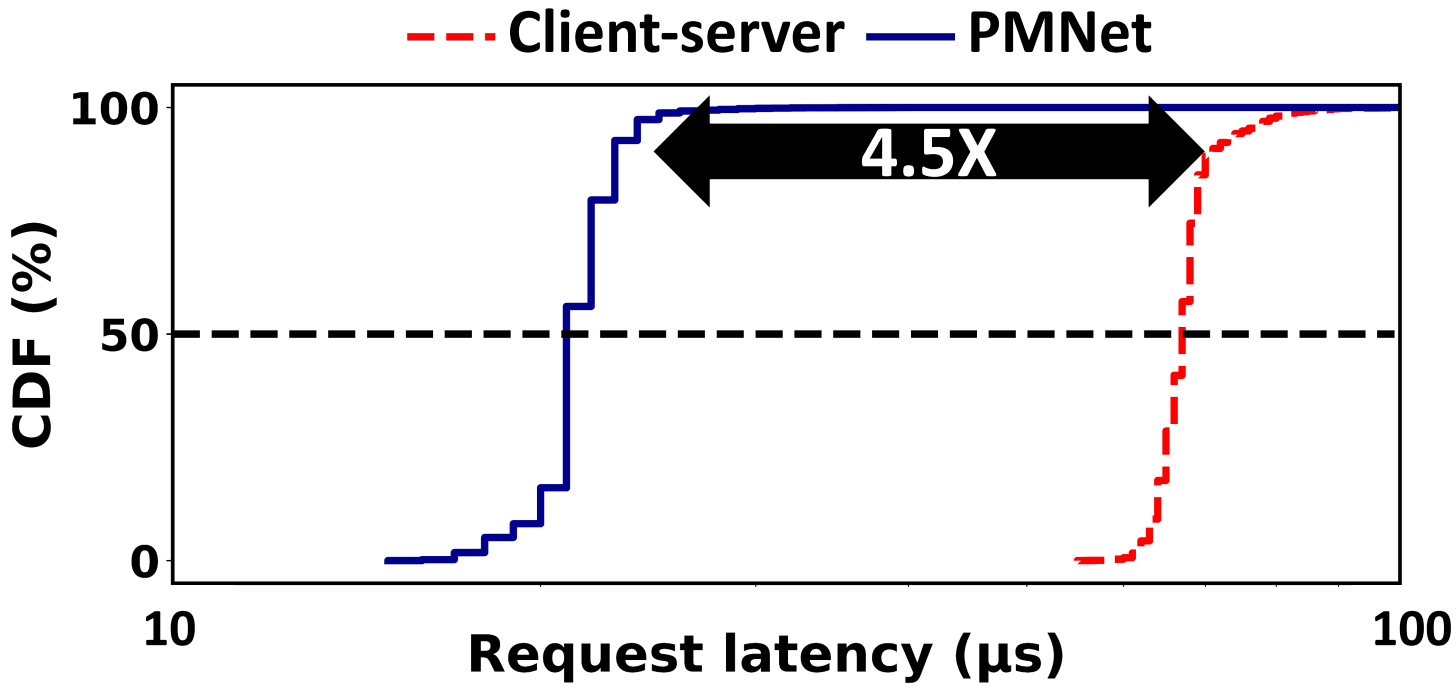
Average Update Latency



PMNet effectively reduces the **average latency** of update requests

Results: Key-value store workloads

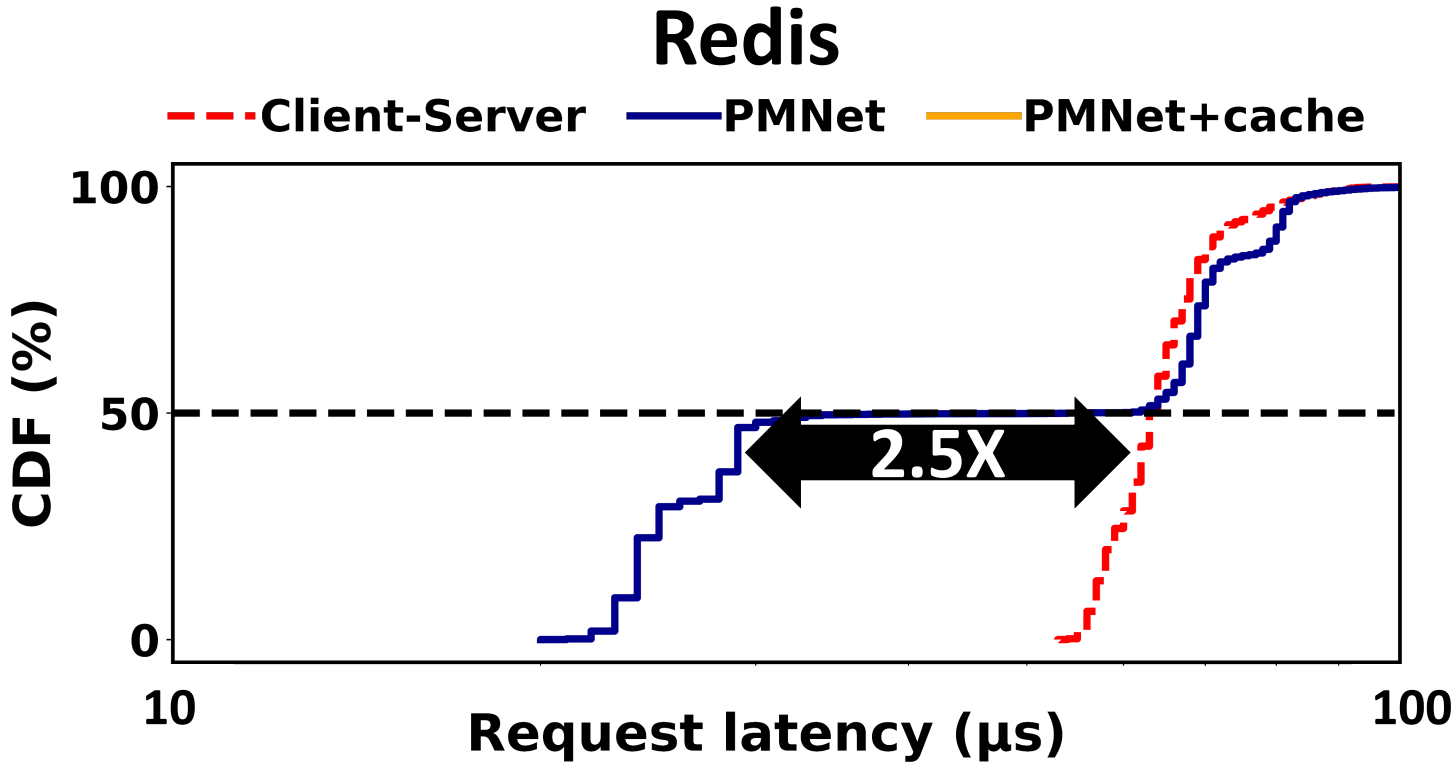
Tail latency (99th-P): 100% Update requests in Redis



PMNet significantly improves 99-percentile **tail-latency** of update requests

Results: Key-value store workloads

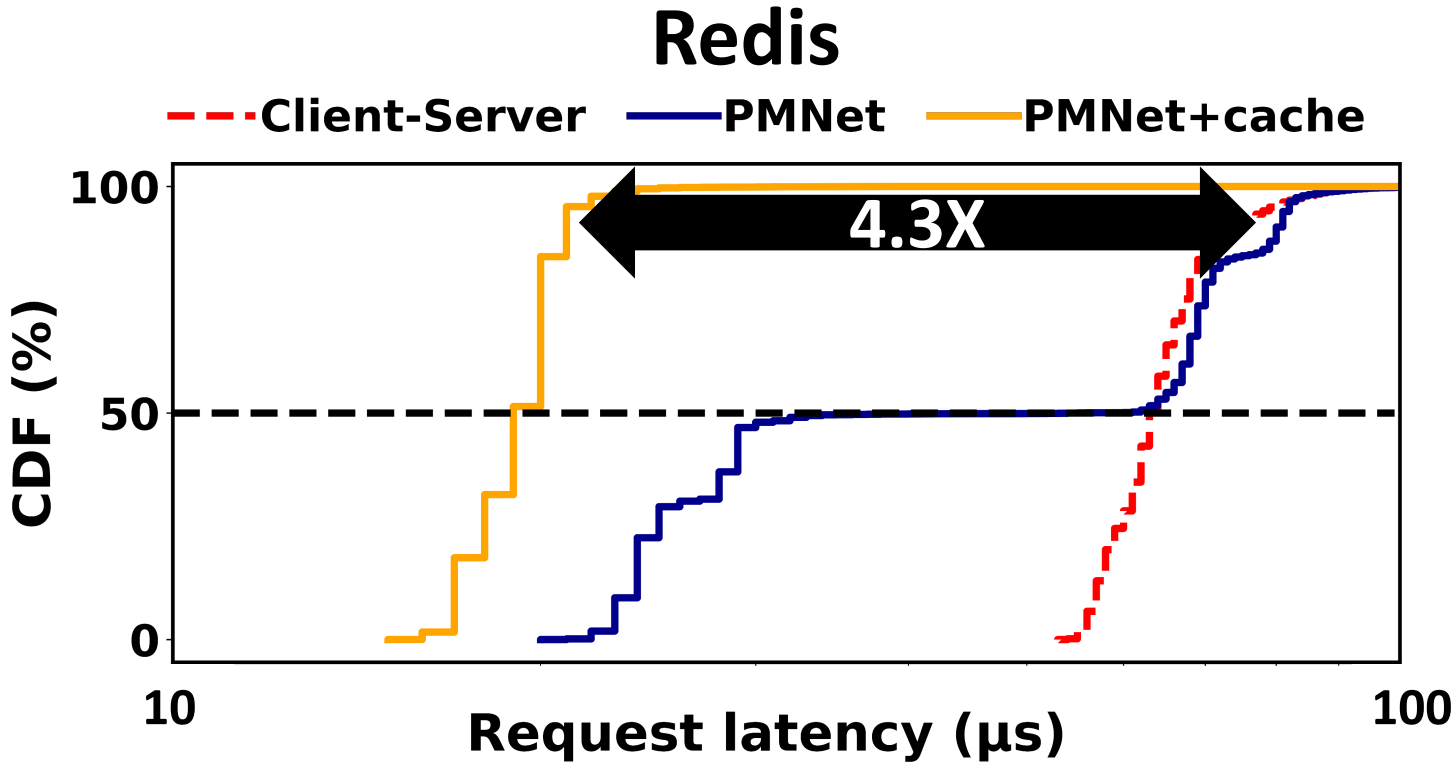
Tail latency: 50%-50% Update-read requests & Cache



Without read cache, PMNet only improves **update** requests' latency.

Results: Key-value store workloads

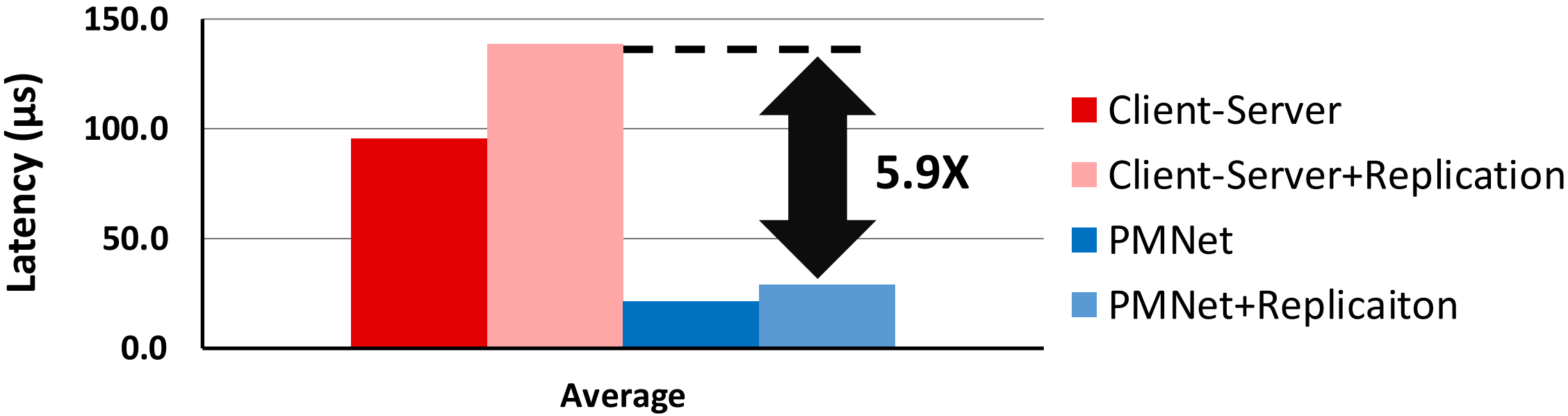
Tail latency: 50%-50% Update-read requests & Cache



With read cache, PMNet also improves **read** requests' latency.

Results: Key-value store workloads

3-way Server Replication (R=3), all workloads average



PMNet **reduces replication latency** while offering the same level of fault tolerance.

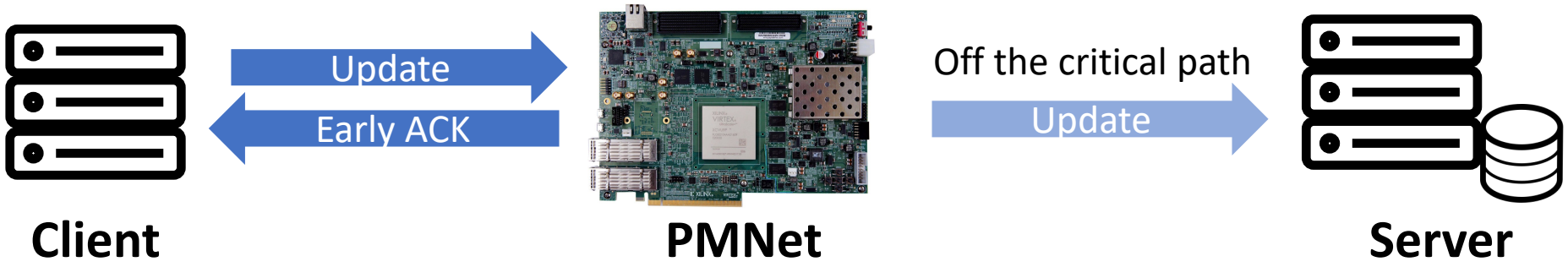
PMNet

Contributions

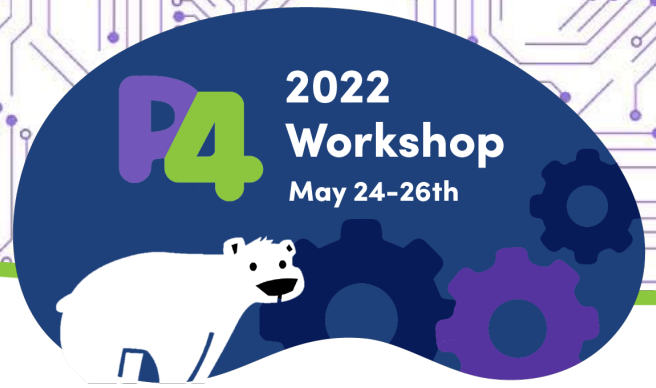
- PMNet introduces a new use-case of in-network computing, providing **data persistence** to the network to accelerate update requests
- PMNet integrates **replication** and **read caching** with **P4 programmable data planes**

Results

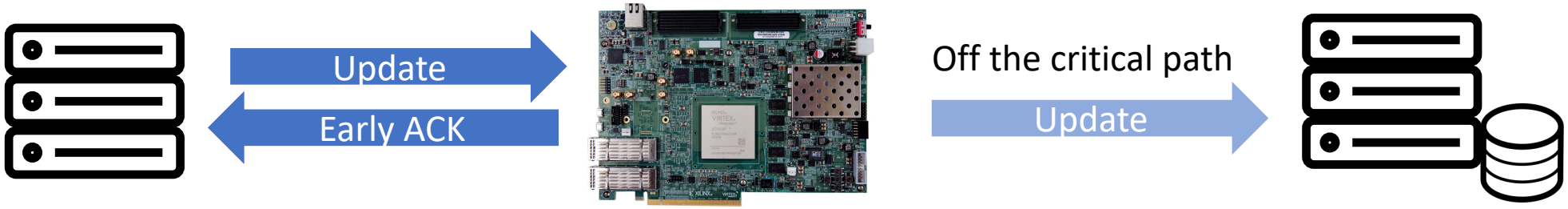
- PMNet improves update throughput by 4.2X and tail latency by 3.2X on average
- PMNet improves 3-way replication (R=3) latency by 5.9X on average
- With Read caching, PMNet improves 50-50% mixed Read-update performance by 3.3X



Presented at 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)
Artifact available at pmnet.persistentmemory.org



Thank You



PMNet A PM-enhanced P4 switch

Presented at 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)

Artifact available at pmnet.persistentmemory.org