Implementation and Evaluation of CYPHONIC client focusing on Sequencing mechanisms and Concurrency for packet processing

Ren Goto[1)], Kazushige Matama[1)], Ryouta Aihata[2)],
Shota Horisaki[3)], Hidekazu Suzuki[3)], Katsuhiro Naito[2)]

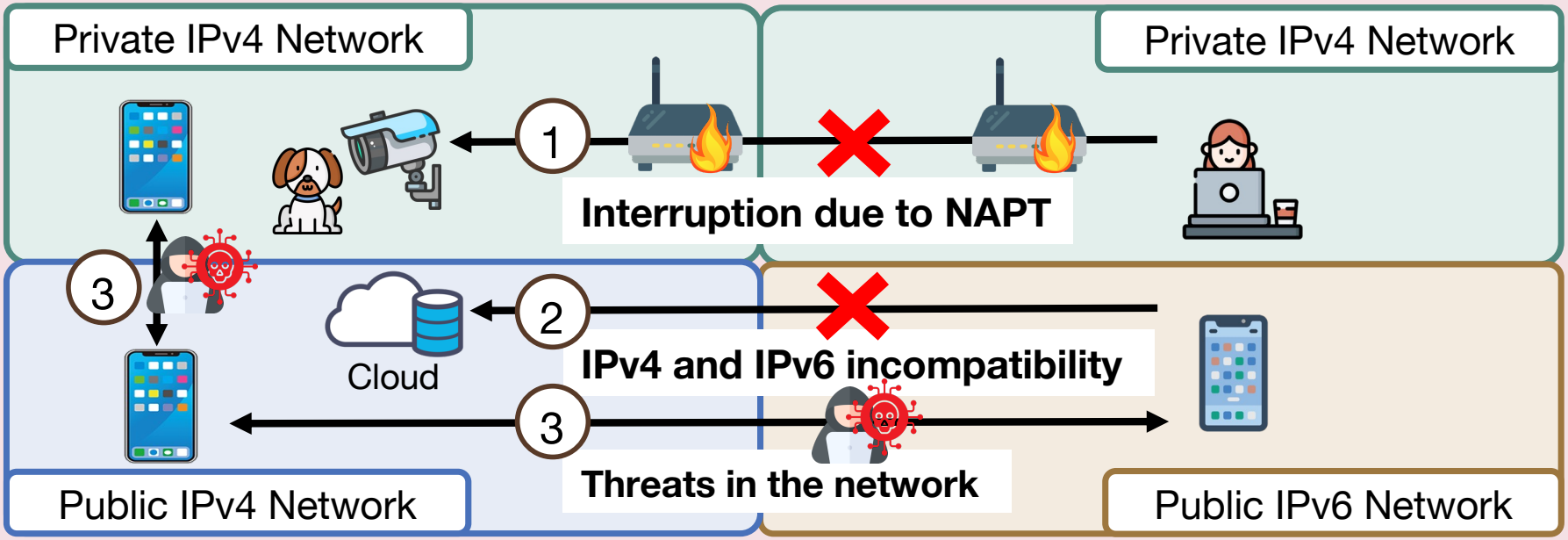
[1)] Graduate School of Business Administration and Computer Science, Aichi Institute of Technology
[2)] Faculty of Information Science, Aichi Institute of Technology
[3)] Faculty of Information Engineering, Meijo University

The 12th Global Conference on Consumer Electronics: GCCE 2023
2023, October, 13

Presentation outline

- Solutions for realizing P2P communication
- Overview of CYPHONIC
- Challenges with conventional systems and client programs
- Objectives
- Proposed schemes
- Performance evaluation
- Conclusions

CYPHONIC: Solutions for realizing P2P model



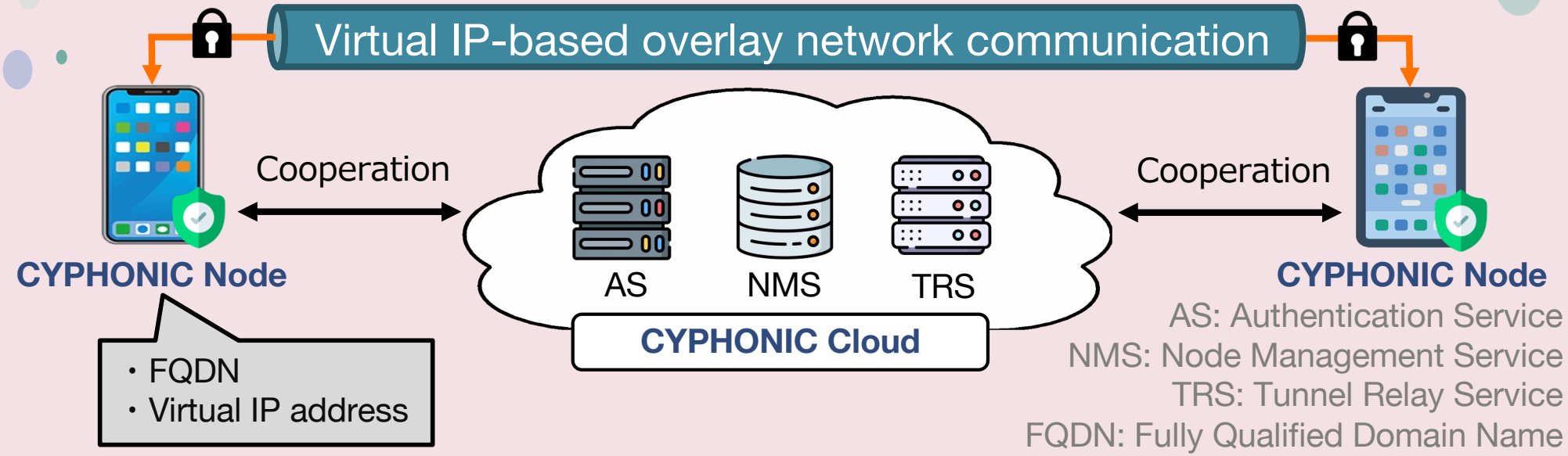
NAPT: Network Address Port Translation

- Requires 1 : Communication across NAPT router
- Requires 2 : Inter-connectivity with IPv4 to IPv6
- Requires 3 : Secure authentication and communication

CYber PHysical Overlay Network over Internet Communication

The communication framework comprehensively the issues of the P2P and provides secure communication services.

Overview of CYPHONIC



CYPHONIC Cloud

- These services provide device authentication and management functions.

CYPHONIC Node

- The end device identifies the peer node with a unique FQDN and communicates directly with the peer node using a virtual IP address.
- The client program (CYPHONIC Daemon) provides communication processing functions.

CYPHONIC Nodes cooperation with CYPHONIC Cloud to autonomously construct tunnel between devices to establish direct communication.

Functions and Challenges of conventional client programs

1 State information associated with the exchange of signaling messages for tunnel construction exists inside the processing function.

Issues:

- Since the management of signaling information depends on the processing module, it is difficult to support multi-threading.
- Communication requests that occur at the same time cannot be processed in parallel.

2 In prototyping, single-threaded packet processing was implemented for simplicity of sequential packet processing.

Issues:

- When processing intercepted application data, processing load at one point affects other processing.
- When tunnel communication is established with multiple end devices, performance degradation is noticeable.

Objectives

Proposal of multi-thread based asynchronous processing scheme focusing on concurrency and packet ordering mechanisms

Supporting internal processing independent of state information

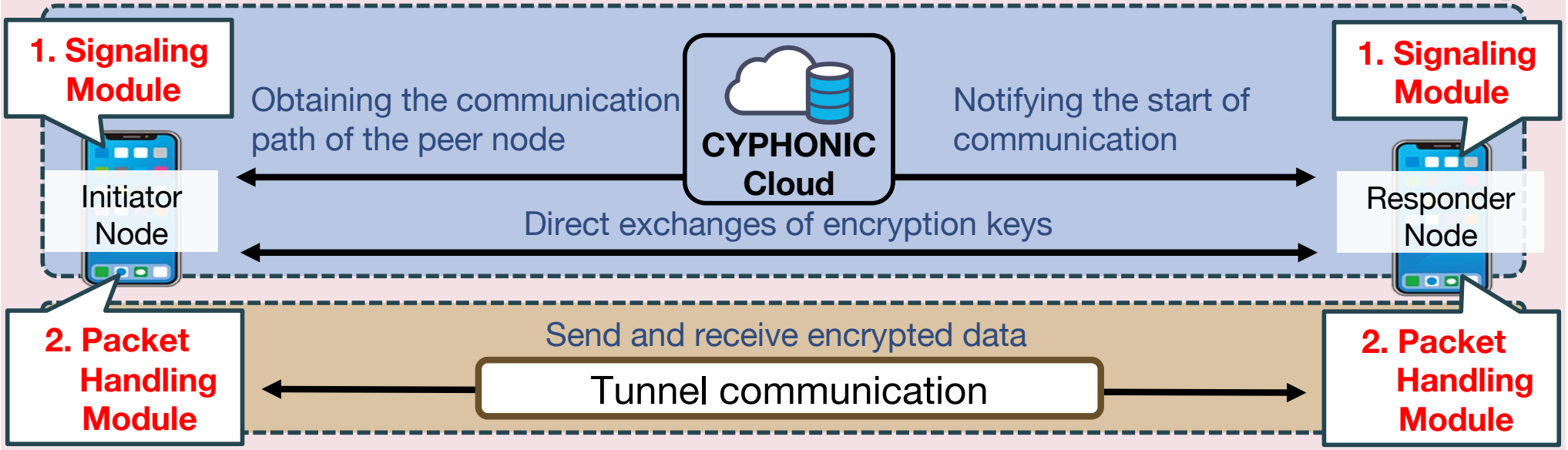
- Add state information inside signaling messages.
- Add an in-memory cache to temporarily store information.

Supporting multi-threaded based packet processing

- Multi-threaded based processing with dedicated worker threads.
- Add packet order maintenance mechanism for asynchronous processing.

Faster and more efficient multi-threaded based processing provides enhanced client programs.

Conventional system of client programs : Overview



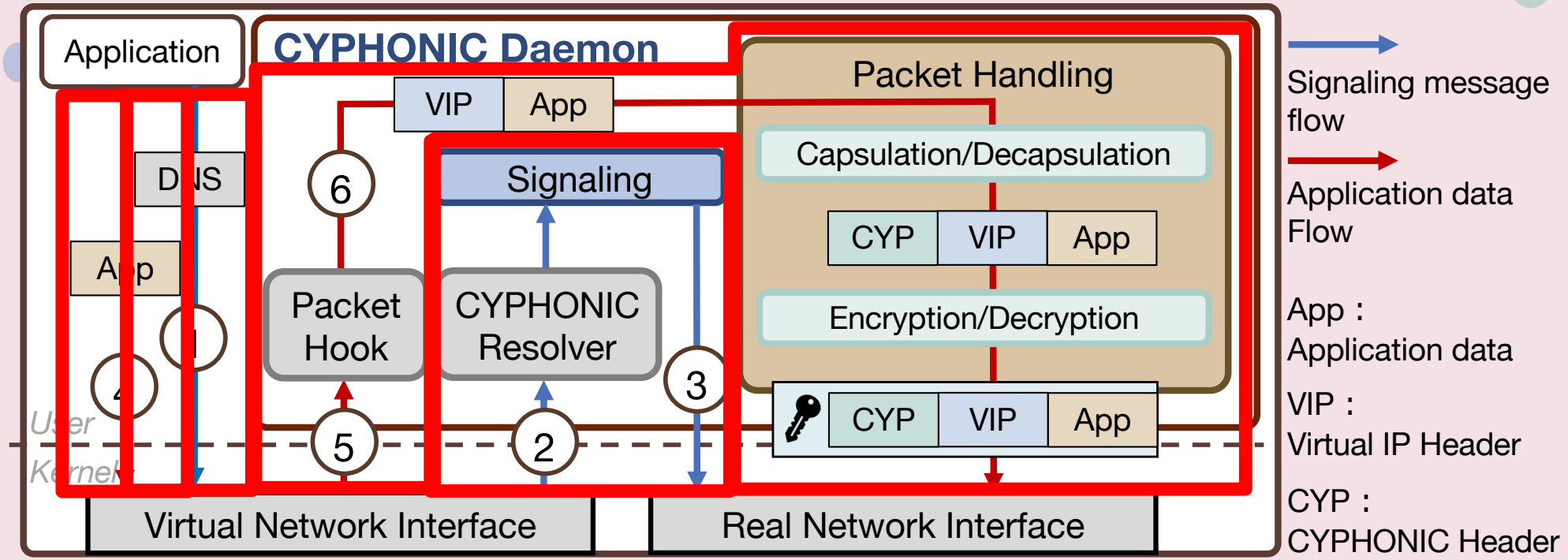
1. Signaling Module : Establishes an overlay network.

- Receiving communication route instructions for the desired peer node.
- The encryption key is exchanged directly with the peer node according to the obtained route.

2. Packet Handling Module : Handles overlay network communications.

- In outgoing side, encapsulate application data, process encryption with a common key, and send it to the overlay network.
- In incoming side, decrypt and decapsulate data received through the overlay network and pass data to the application.

Conventional system of client programs : System model



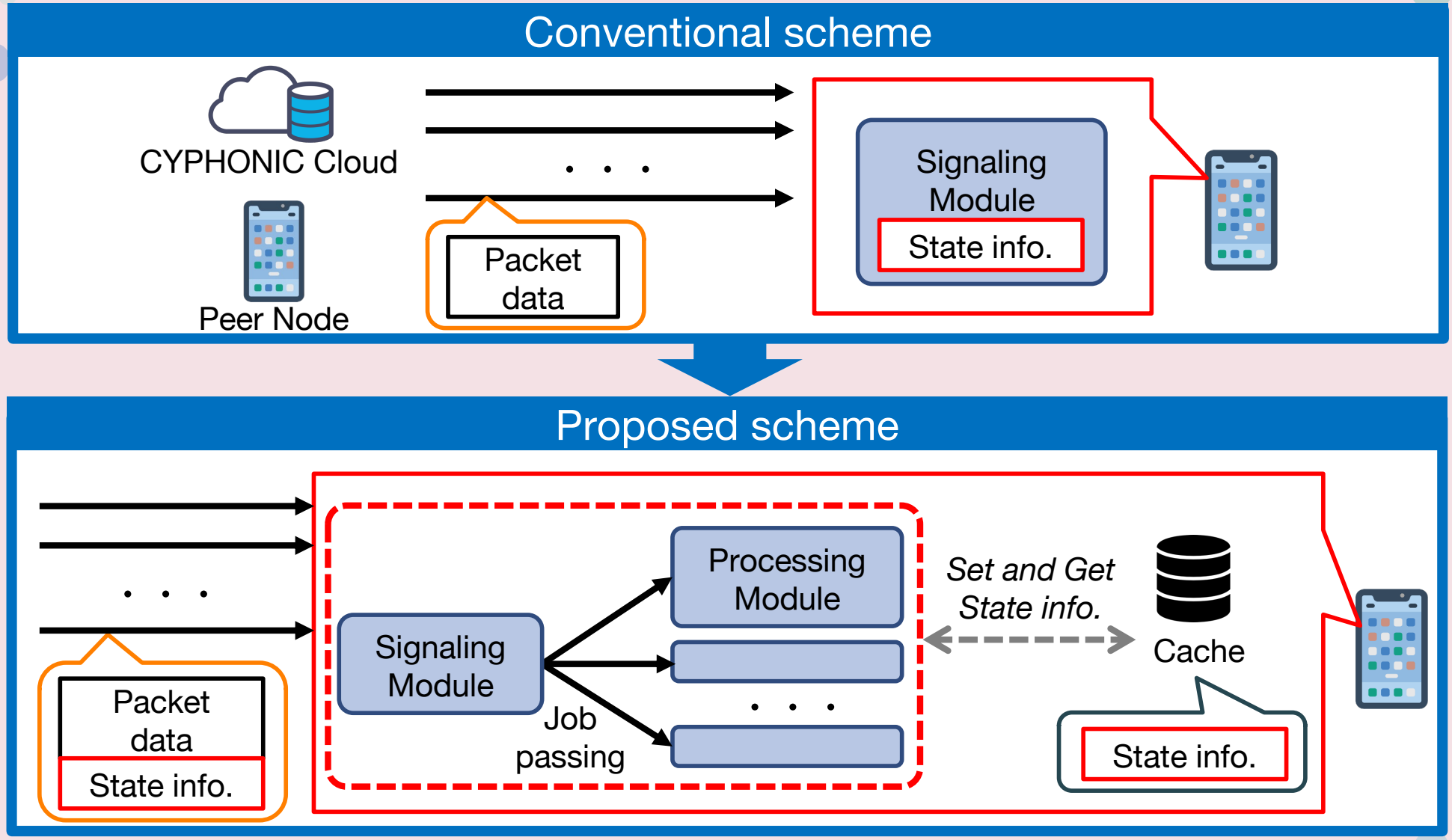
1. Establishes an overlay network

- Signaling Module initiates communication by triggering a DNS request containing the FQDN of the peer node.
- Signaling Module processes signaling messages associated with tunnel construction.

2. Handles overlay network communications

- Packet Handling Module encapsulates and encrypts the application data intercepted through the virtual interface.

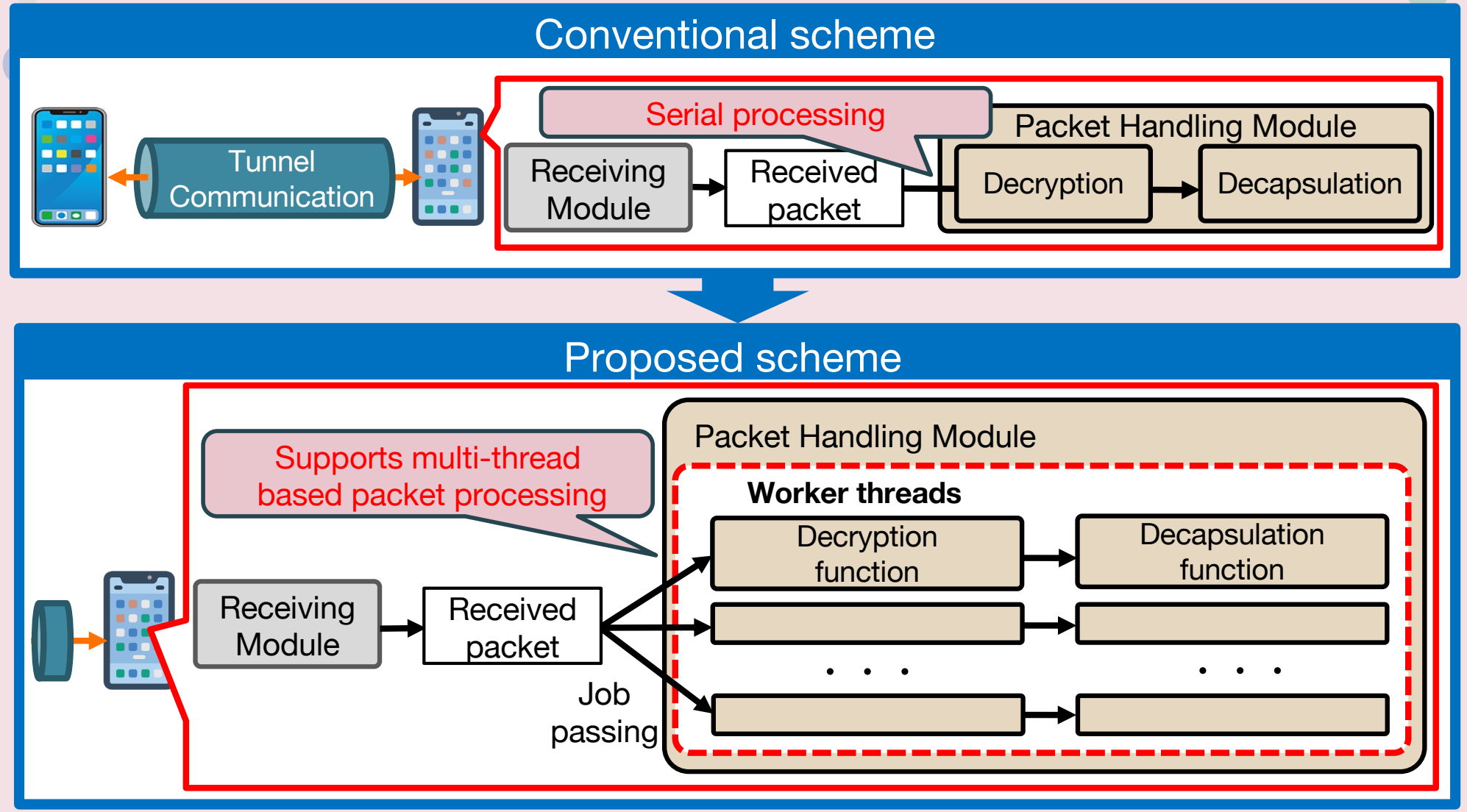
Internal processing independent of state information



State information is added to the packet and stored in a cache store.

➡ Processing modules can easily access the state information and process multiple operations concurrently.

Multi-threaded based packet processing



Prepare a dedicated worker thread for the encryption/encapsulation process.

➡ Receiving thread passes the packets to each worker thread for concurrency processing.

Implementation issues of the proposed scheme

1. Thread creation and allocation

- Creation of worker threads may cause processing delays.

➡ Dedicated processing threads are pre-generated and receive jobs from parent threads.

2. Transaction in multi-thread processing

> Transaction:
> Sequence of signaling from sending a request to receiving and processing a response.

- Transactions must be identified between all asynchronously executed modules.

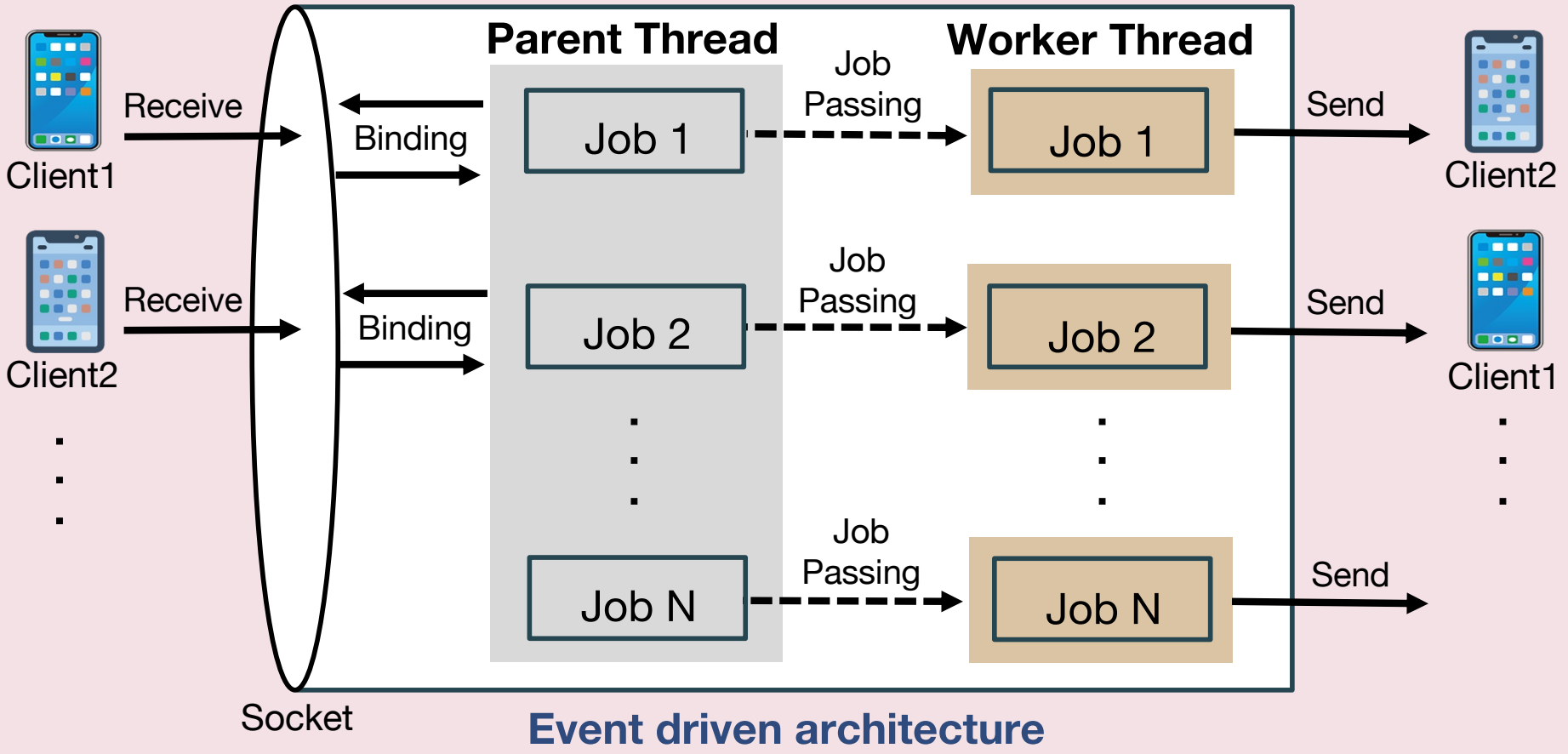
➡ Include key information uniquely identifying the cache in all incoming and outgoing messages.

3. Multi-threaded Packet Handling Module

- Packet order may differ between receiving and sending.

➡ Packet ordering schemes and sequential processing are essential.

Thread creation and allocation

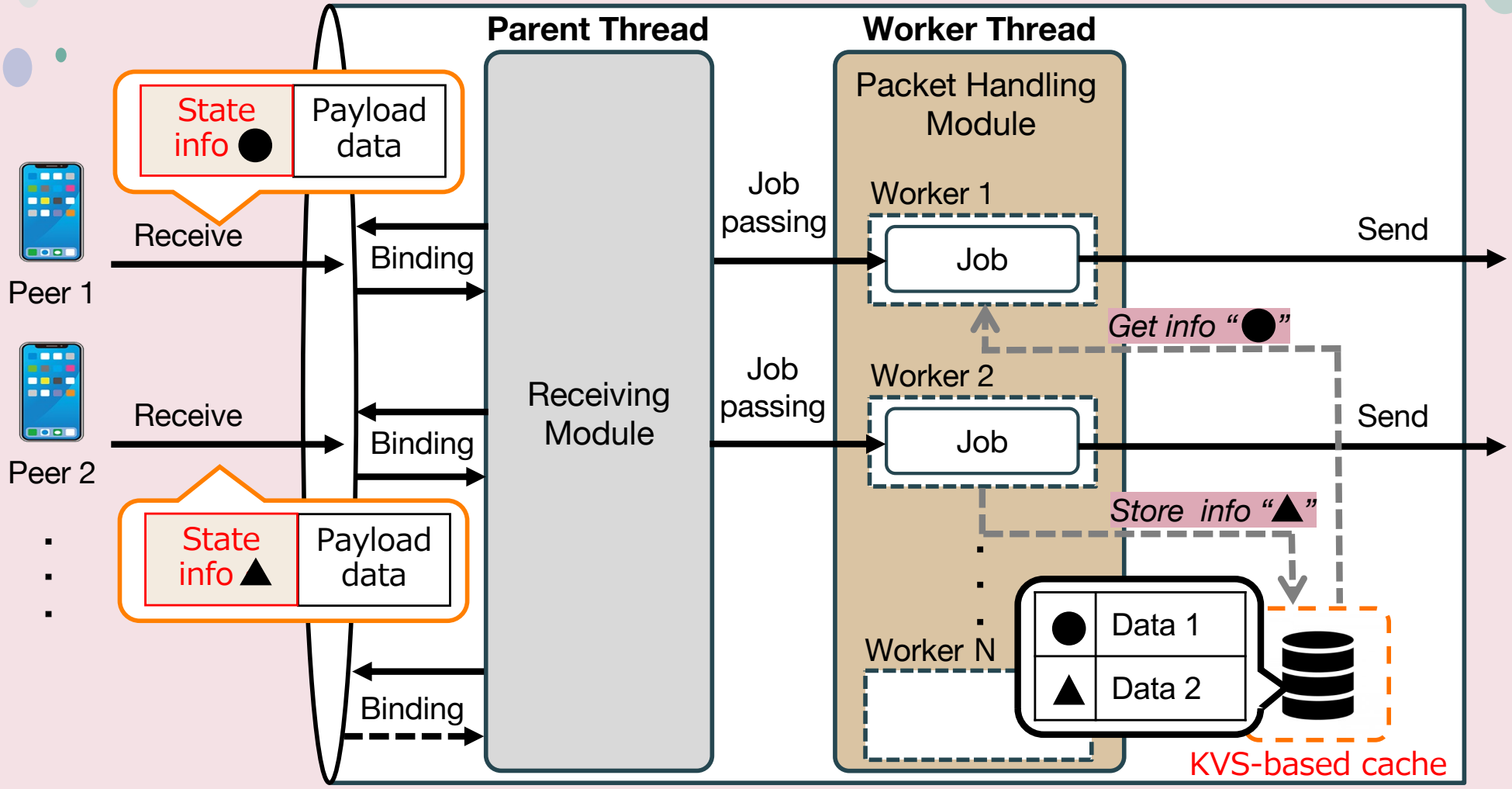


Event driven architecture

Multi-threading based on event-driven architecture

- Pre-generated worker threads are used for processing, reducing resource request overhead.
- Next processing can be performed without waiting for the request to be fully processed

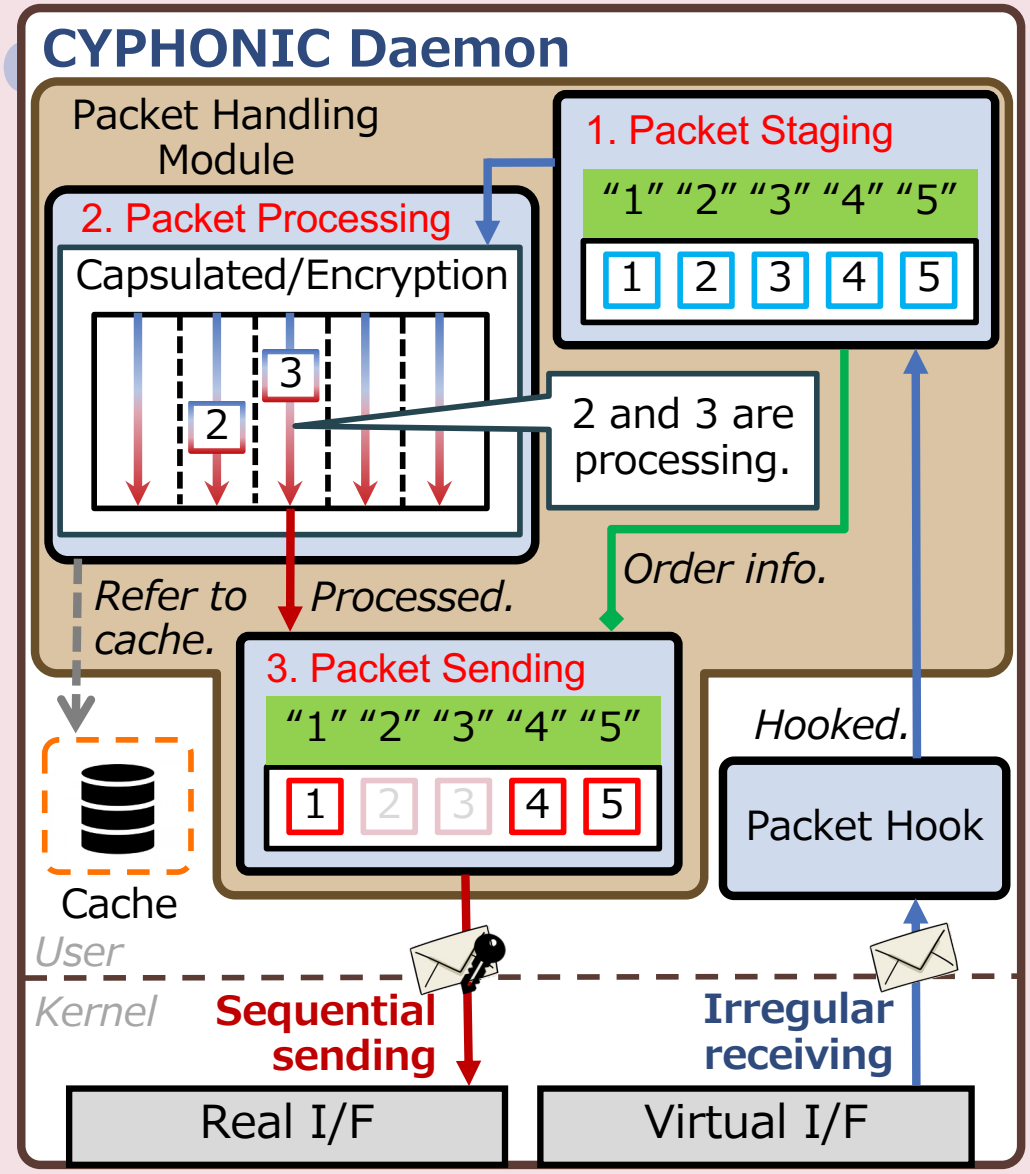
Transaction in multi-thread processing



Cooperative processing by multiple worker threads

- A key is added to the packet to reference cache information.
- By introducing a cache store, a new worker thread can access the state information generated by the previous worker thread.

Multi-threaded Packet Handling Module



1. Packet Staging Module

Buffers incoming packets and stores the order of reception.

2. Packet Processing Module

Pass processing to worker threads for asynchronous capsulation/encryption.

3. Packet Sending Module

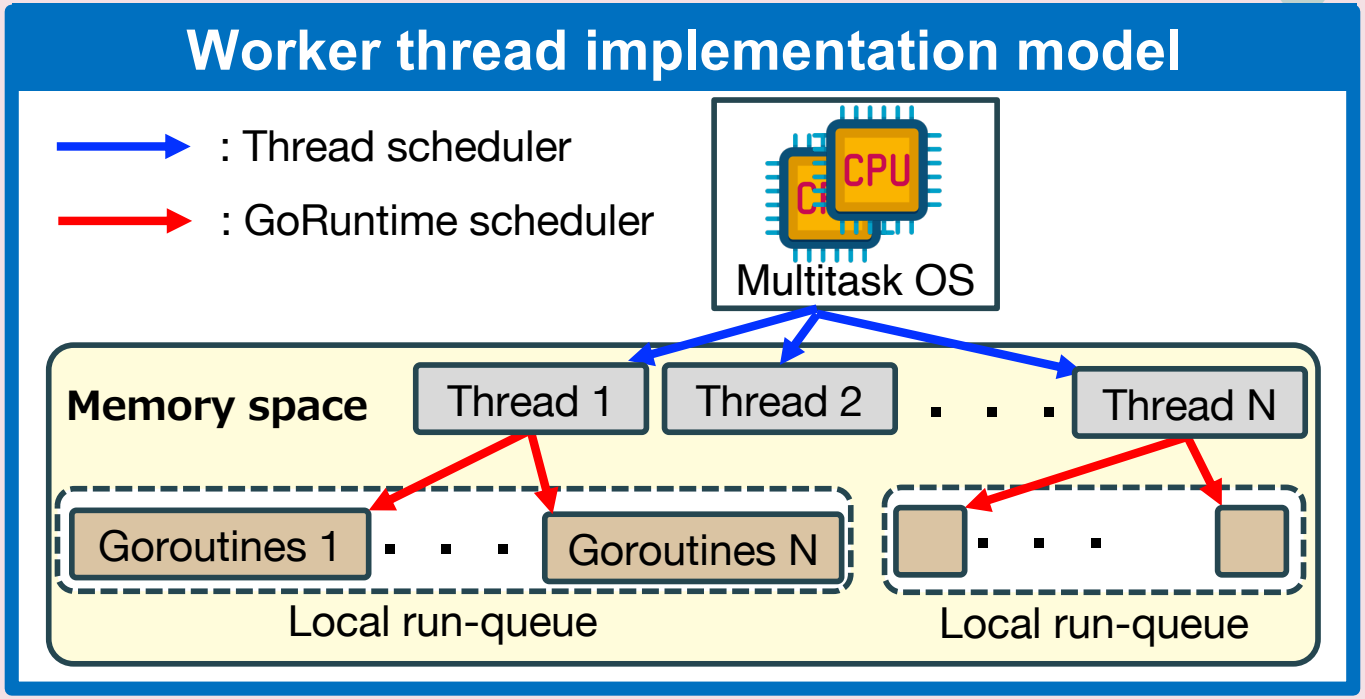
Processed packets are added to the queue and sent in the order received.

Ordering mechanisms and Sequential processing model

- Packet order can be maintained regardless of worker thread processing status.

Implementation of the proposed system

CYPHONIC Daemon	
Runtime	Go ver 1.20
Worker thread	Goroutines
Mutex	sync package



Event-driven model

- Goroutine creates $M{:}N$ scheduler capable of processing N concurrently for M logical cores.

➡ Context switches are hidden from the OS.

Sequential processing scheme

- A single packet gets a lock before being passed to a worker thread by mutex and is unlocked when processing is complete.

➡ Prohibit unauthorized access to packets being processed.

Verification subjects and evaluation environment

Performance evaluation of CYPHONIC node when multiple tunnels are established.

Network performance

- TCP and UDP throughput measurement.

➡ We used by iperf3.

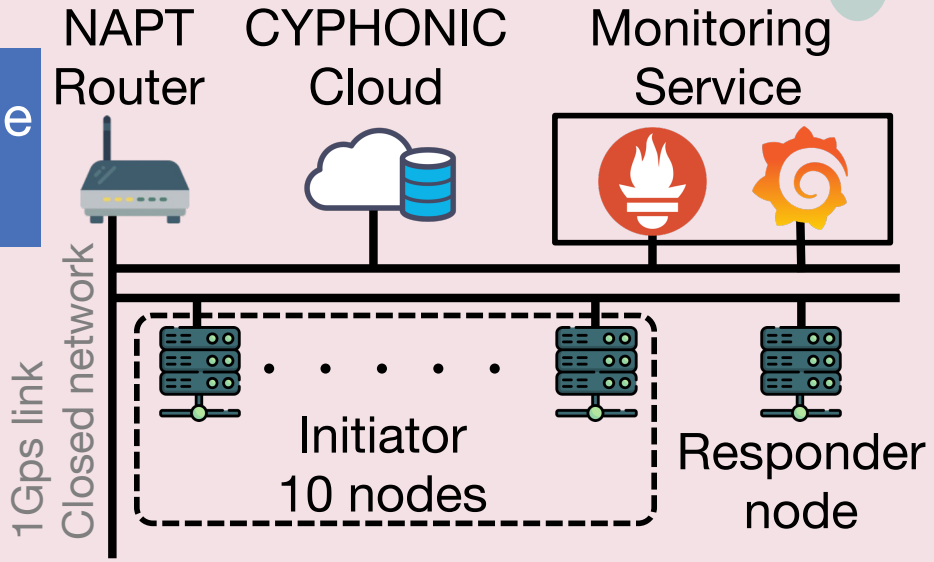
- Measurement of RTT by ICMP.

➡ We used ping.

Internal processing trends

- Trends in OS threads and Goroutines count and memory usage.

➡ We used by NodeExporter and GoMetrics.



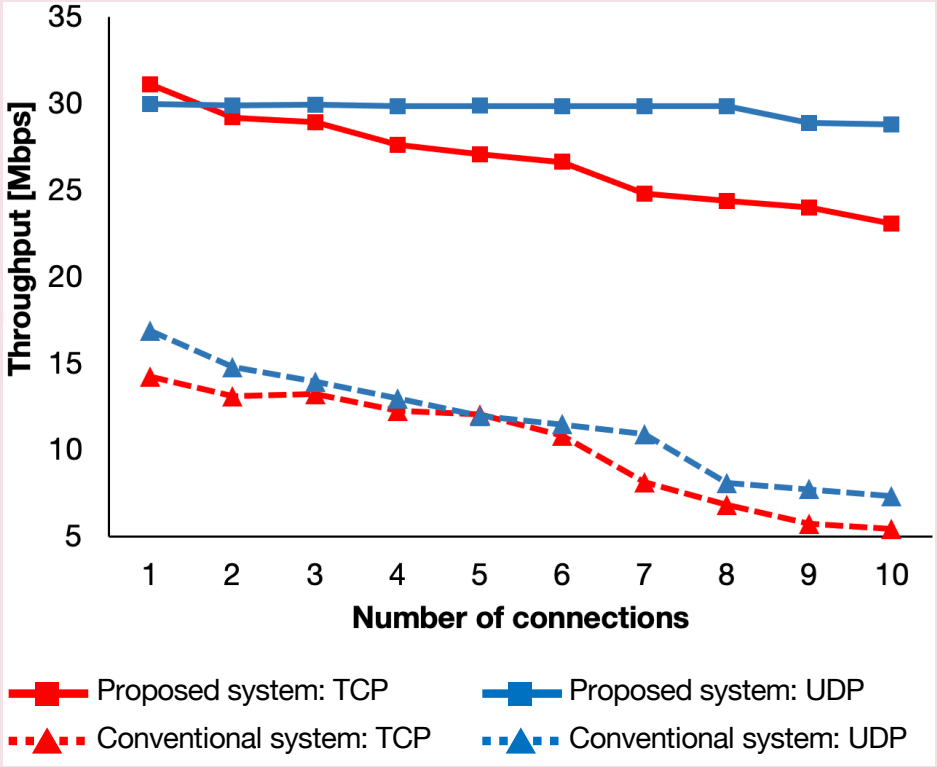
Virtual Machine (CYPHONIC Node)	
OS	Ubuntu 22.04 Jammy Jellyfish
CPU	Intel(R) Core(TM) i9-13900 CPU@5.60GHz, 2-cores / 2-threads
Memory	1 GiB

1 Responder node and 10 Initiator nodes are provisioned in the closed network network.

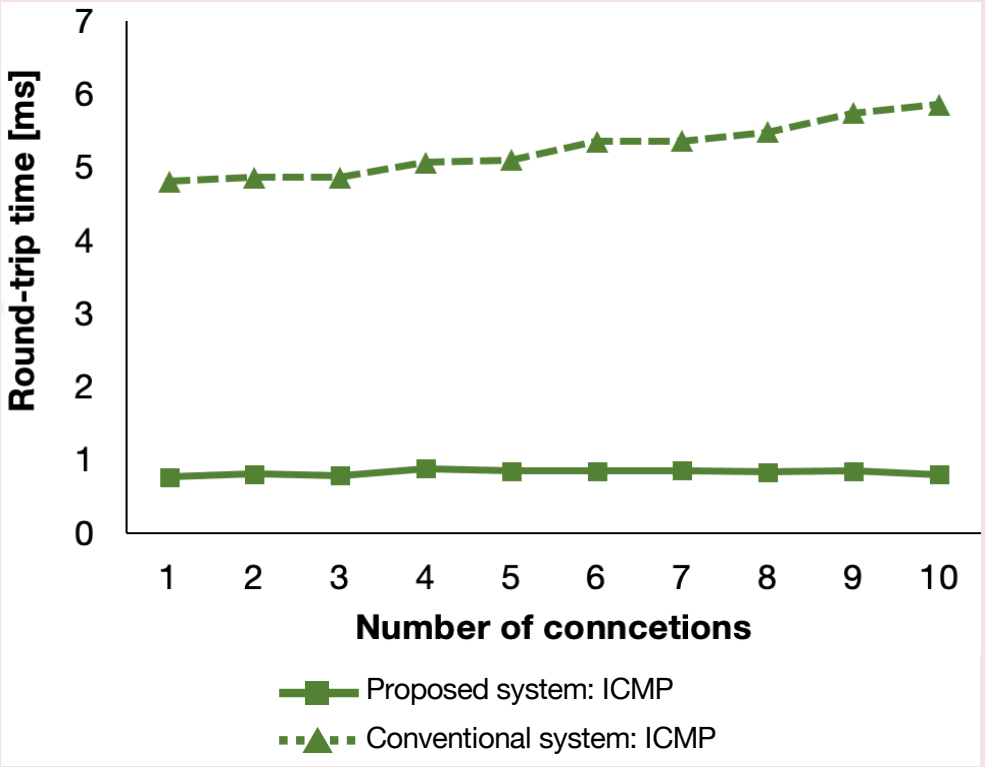
➡ Establish tunnel connections with up to 10 peer nodes.

Evaluation results of the Communication performance

- **TCP and UDP throughput**



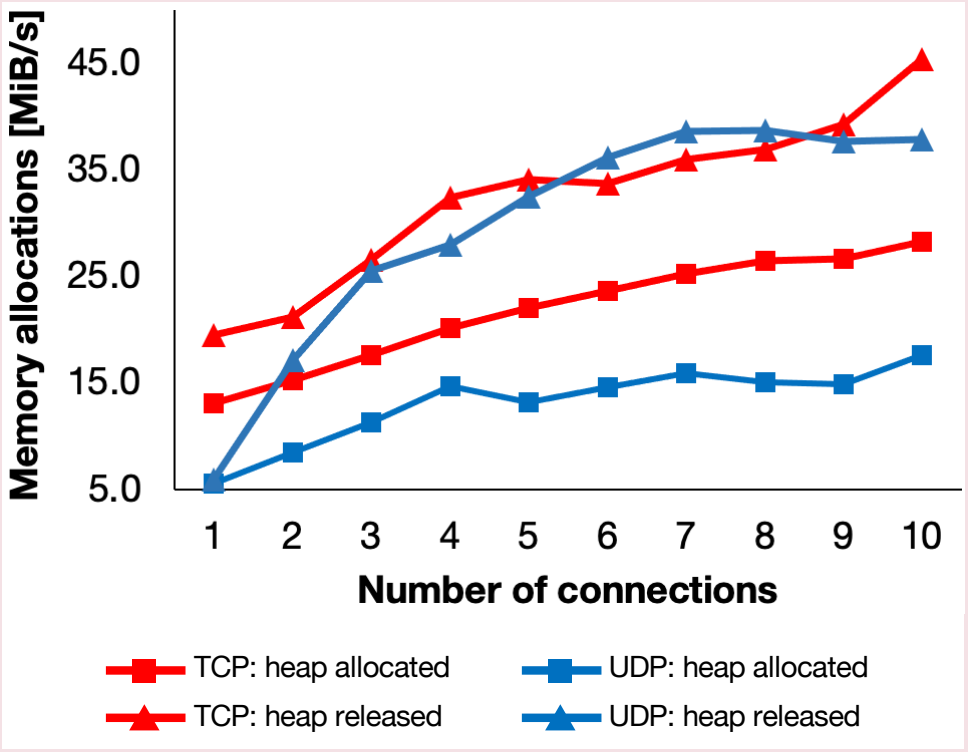
- **Communication delay**



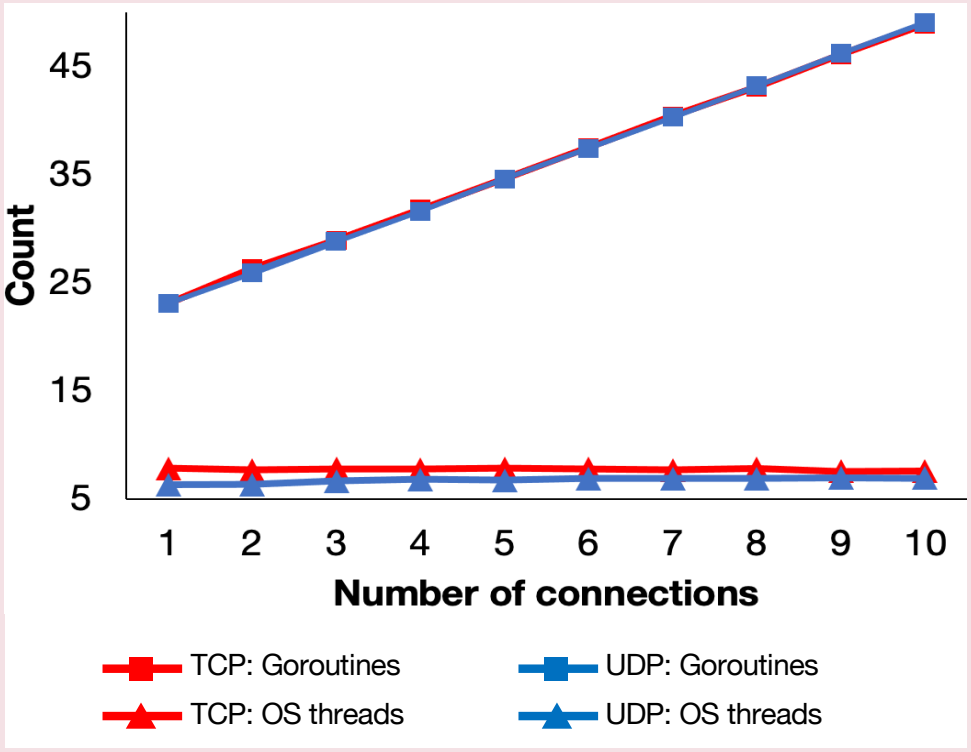
- Focusing on a single connection, TCP and UDP improved throughput by 16.9 Mbit/sec and 13.1 Mbit/sec, respectively, and communication delay was improved by 4.0 ms.
- We confirmed that the proposed scheme has a small increase in communication delay even when the number of connections increase.

Evaluation results of the Application performance

- **Trends in APM metrics**



- **Trends in Goroutine and Thread**



APM: Application Performance Management

- The heap area is properly released at the end of the connection, so that more memory is released than allocated.

- While worker threads increase, OS threads remain constant.
- Hides the overhead associated with thread generation and context switches from the OS.

Conclusions

We Proposed multi-thread based asynchronous processing scheme focusing on concurrency and packet ordering mechanisms

Supporting internal processing independent of state information

- Add state information inside signaling messages.
- Add an in-memory cache to temporarily store information.

Supporting multi-threaded based packet processing

- Multi-threaded based processing with dedicated worker threads.
- Add packet order maintenance mechanism for asynchronous processing.

Significantly improves throughput and maintains constant communication delay due to increased connections.

The proposed processing model, CYPHONIC client processing performance can be significantly improved.