VIRTUAL

Connext®
2021

October 26th-27th

Erin McManus
Principal Engineer

✉ erin@rti.com

Kyoungho An
Senior Research Engineer

✉ kyoungho@rti.com

Product Advancements for Scaling Autonomous Systems

VIRTUAL Connext® 2021

Technical Deep Dive

Agenda

- WAN Connectivity
- Performance and Scalability
- Containerization

Agenda

- ☁ **WAN Connectivity**
- Performance and Scalability
- Containerization

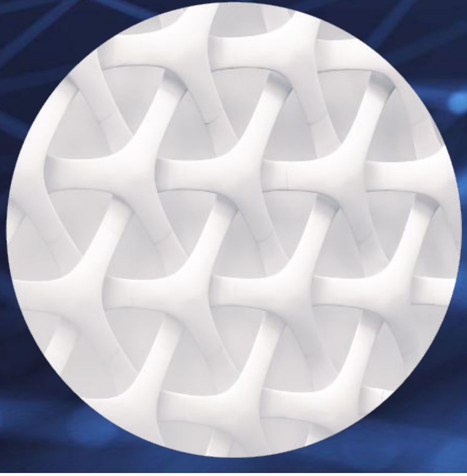
Globally Distributed Systems Face Unique Challenges

Limited Edge Resources

Increased Security Risks

Dynamically Discovering Systems

Connectivity Across WAN and Diverse Networks



Unreliable Networks

RTI Connext Anywhere

new

RTI Real-Time WAN Transport

UDP-based transport supporting network roaming

new

RTI Cloud Discovery Service

Facilitates endpoint discovery and NAT traversal

RTI Connext Secure

Secures data end-to-end scalably over the WAN networks

rti

Leverages All Capabilities of the Connext Framework

Communications Patterns: Publish/Subscribe, Request/Reply, and RPC, Queuing

Data Caching for late-joiner and disconnection periods

Fault Tolerance

Seamlessly integrate with RTI modules that provide additional services:
Recording and Replay
Database Integration
Web Integration

High performance transports:
Zero copy over SHMEM, UDP multicast

Rich development, visualization and test tools: Admin Console, Monitor, Connector

Content Filters at the Source send only what is relevant

Balance reliability and speed in delivery

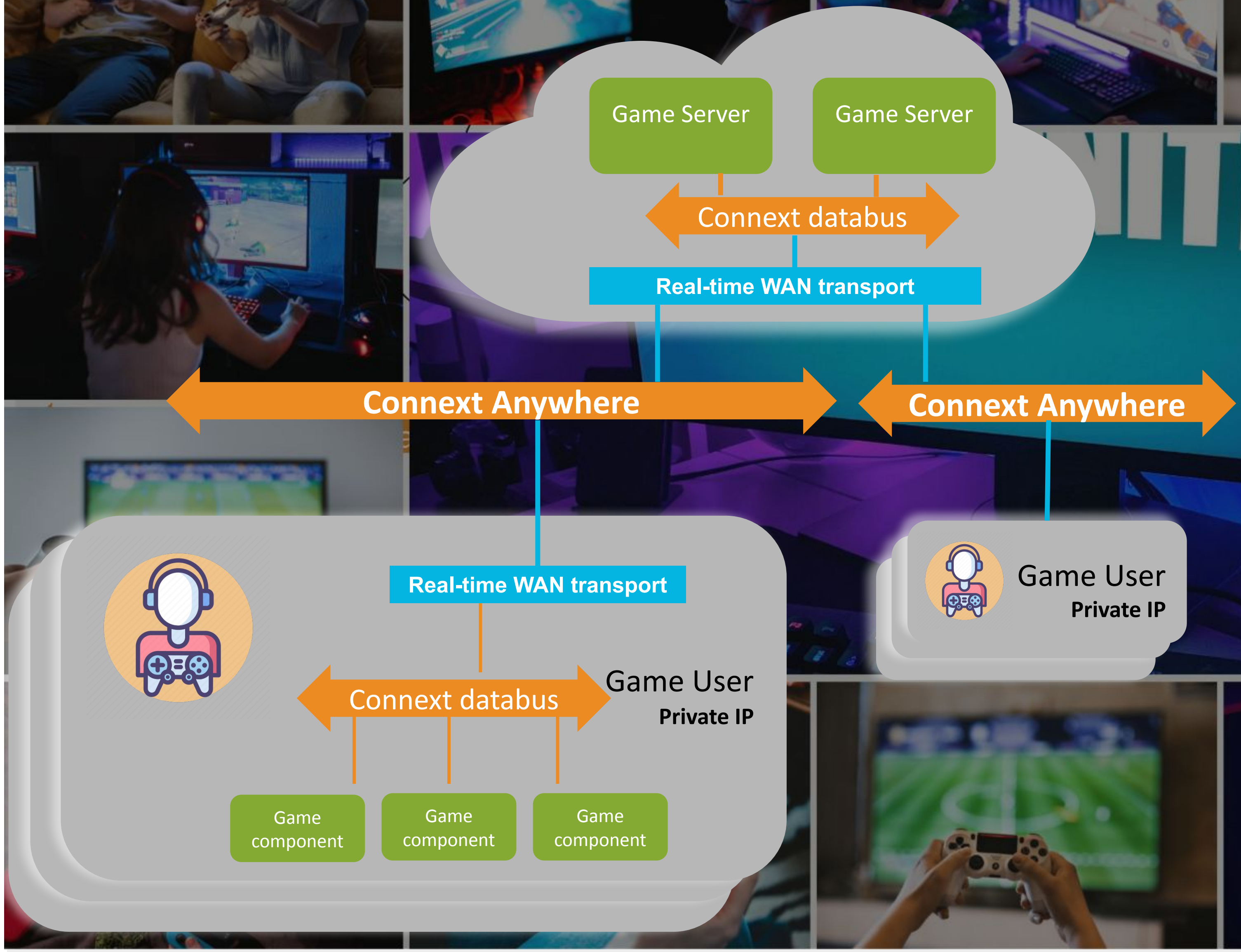
Monitoring: Ability to monitor the infrastructure and communication

Integrated and Evolvable Type System for incremental updates to applications

Performant & Flexible WAN Communications

	TCP Transport	UDP Transport	Real-Time WAN Transport
Private to Public Communication	✓	✗	✓
ivate to Private Communication across NATs	✗	✗	✓
IP Mobility	✓	✓	✓
Best Effort Delivery	✗	✓	✓
Reliable Delivery	✓	✓	✓
Transport Level Security	✓	✓	✗
Flow Level Security	✓	✓	✓
L4 Load Balancer Support	✗	✗	✓
L7 Load Balancer Support	✓	✓	✓

Deployment Scenarios



Game Server

Game Server

Connext databus

Real-time WAN transport

Connext Anywhere

Connext Anywhere

Real-time WAN transport

Connext databus

Game User
Private IP

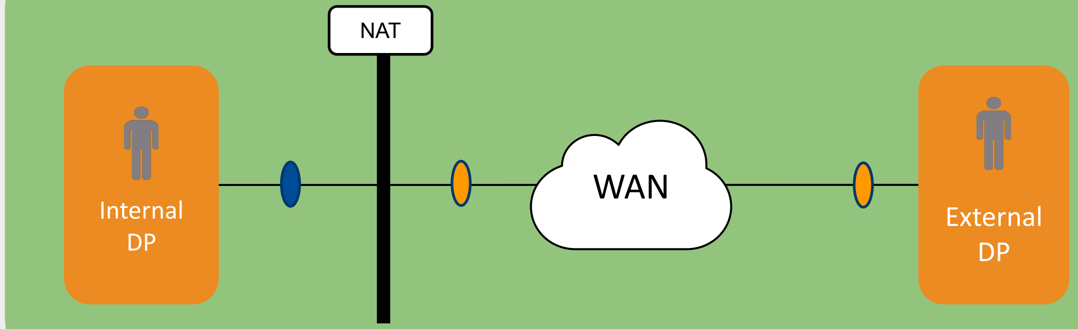
Game component

Game component

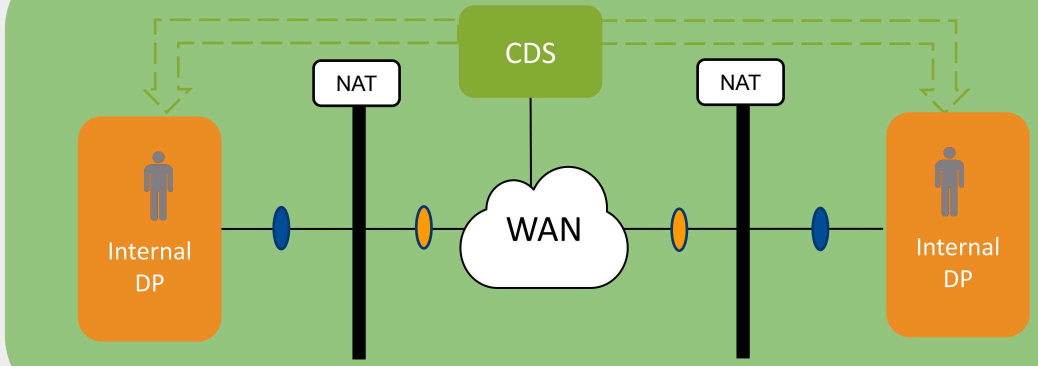
Game component

Game User
Private IP

P2P Communication Scenarios

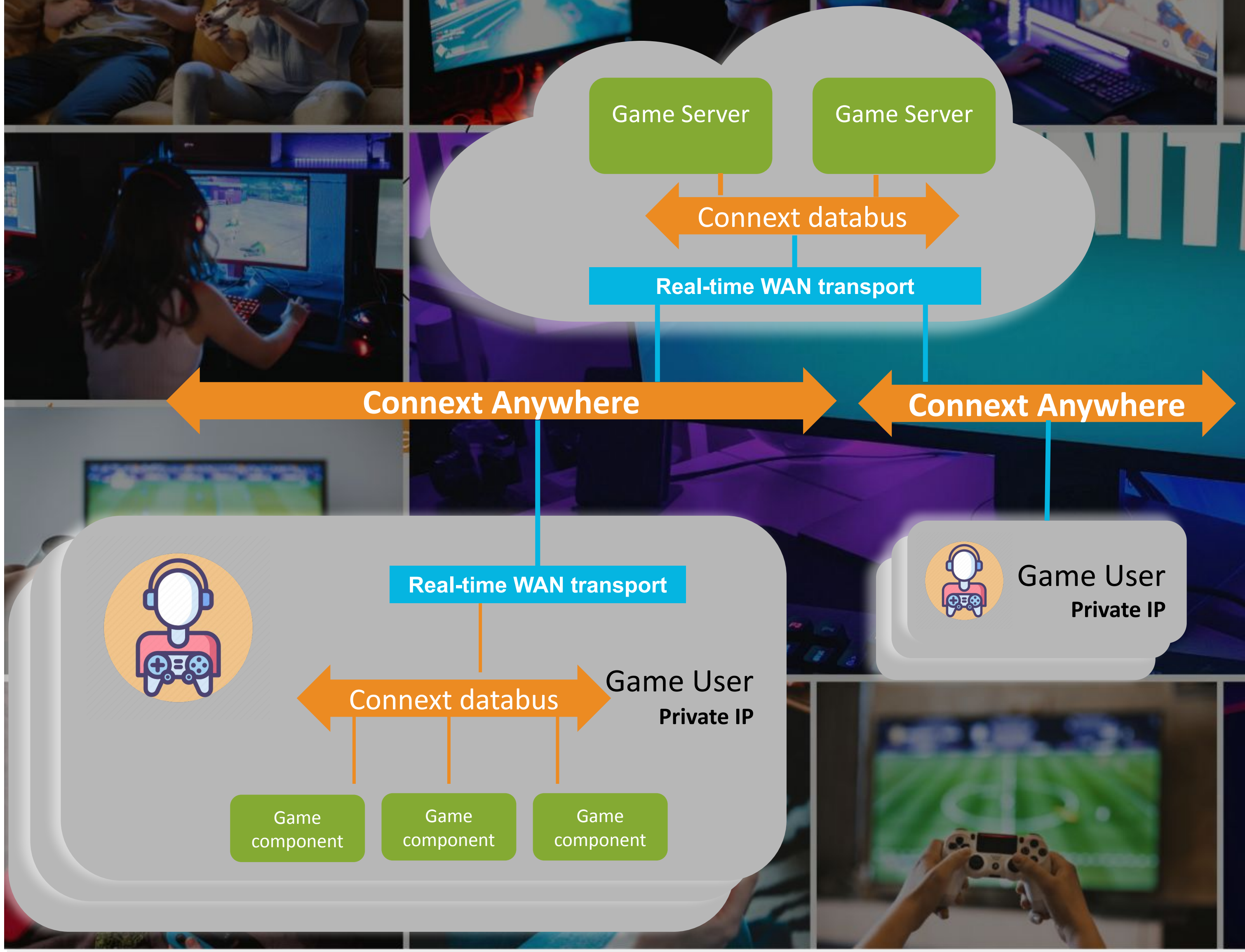


Publicly Reachable
Participant <-> Participant
behind **ANY NAT**



Both Participants Behind
Cone-NATs

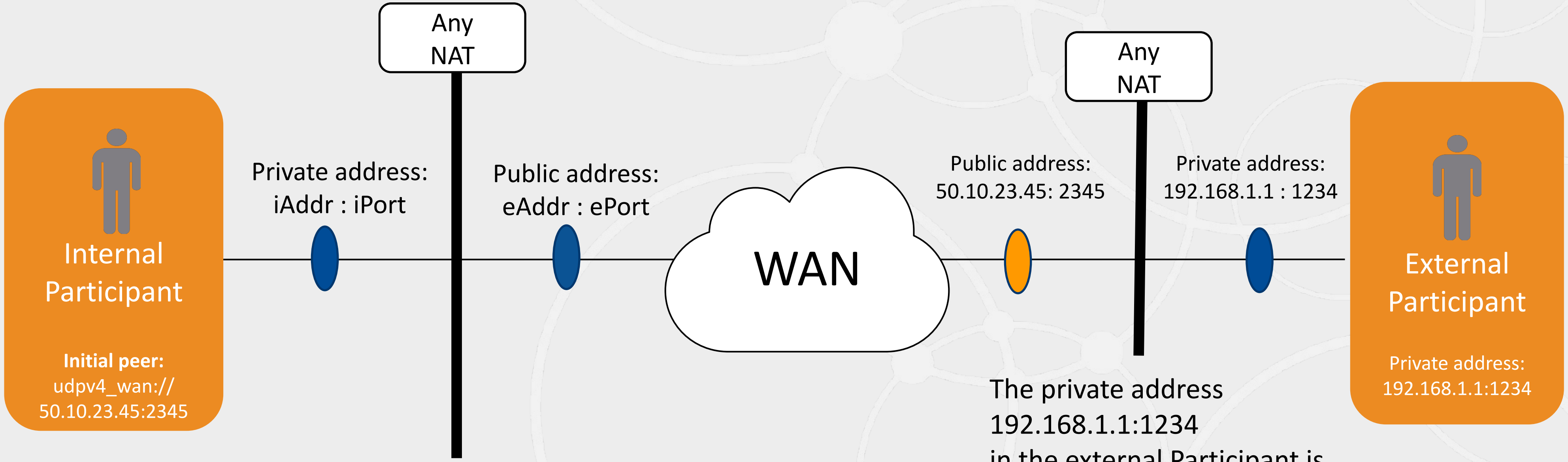
Edge to Cloud (private to public)



The Real-Time WAN Transport and RTI Routing Service provide a scalable solution to connect the edge to the cloud or on-prem data centers

- Support for multiple gateway component acting as entry points in data centers for edge applications
- Ability to send and receive information from the data centers
- High-speed and performant connection within data centers using UDPv4, multicast, shared memory and other transports

Peer-to-Peer with External Participant 1/3



The private address 192.168.1.1:1234 in the external Participant is statically mapped by changing the NAT-enabled router configuration to 50.10.23.45:2345

More than one Internal Participant communicating with the External Participant

Peer-to-Peer with External Participant 2/3

Internal Participant

Initial peer:
udpv4_wan://
50.10.23.45:2345

```
<dds>
    <qos_profile name="InternalParticipant">
        <participant_qos>
            <transport_builtin>
                <mask>UDPv4_WAN</mask>
            </transport_builtin>
            <discovery>
                <initial_peers>
                    <element>
                        0@udpv4_wan://50.10.23.45:2345
                    </element>
                </initial_peers>
            </discovery>
        </participant_qos>
    </qos_profile>
</dds>
```

50.10.23.45:2345 is the public IP address and port of the External Participant

0@ is not mandatory but it is recommended to reduce number of RTPS messages

Peer-to-Peer with External Participant 3/3

```
<dds>
    <qos_profile name="ExternalParticipant">
        <participant_qos>
            <transport_builtin>
                <mask>UDPv4_WAN</mask>
                <udpv4_wan>
                    <public_address>
                        50.10.23.45
                    </public_address>
                    <comm_ports>
                        <default>
                            <host>1234</host>
                            <public>2345</public>
                        </default>
                    </comm_ports>
                </udpv4_wan>
            </transport_builtin>
        </participant_qos>
    </qos_profile>
</dds>
```

50.10.23.45:2345 is the public IP address and port of the External Participant

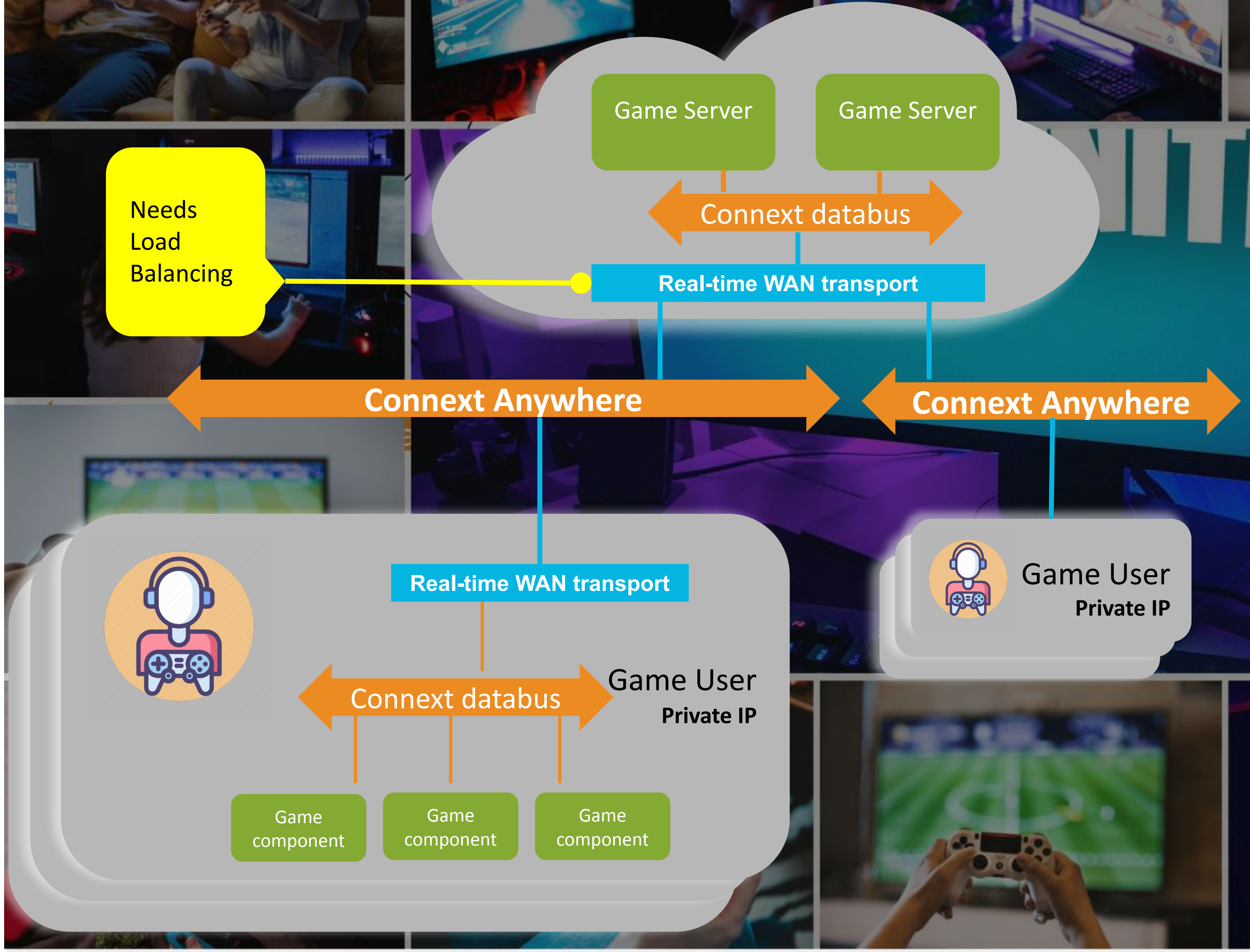
192.168.1.1:1234 is the private IP address and port for the External Participant

No need to set initial peers

Do not forget to configure your router!

External Participant
Private address:
192.168.1.1:1234

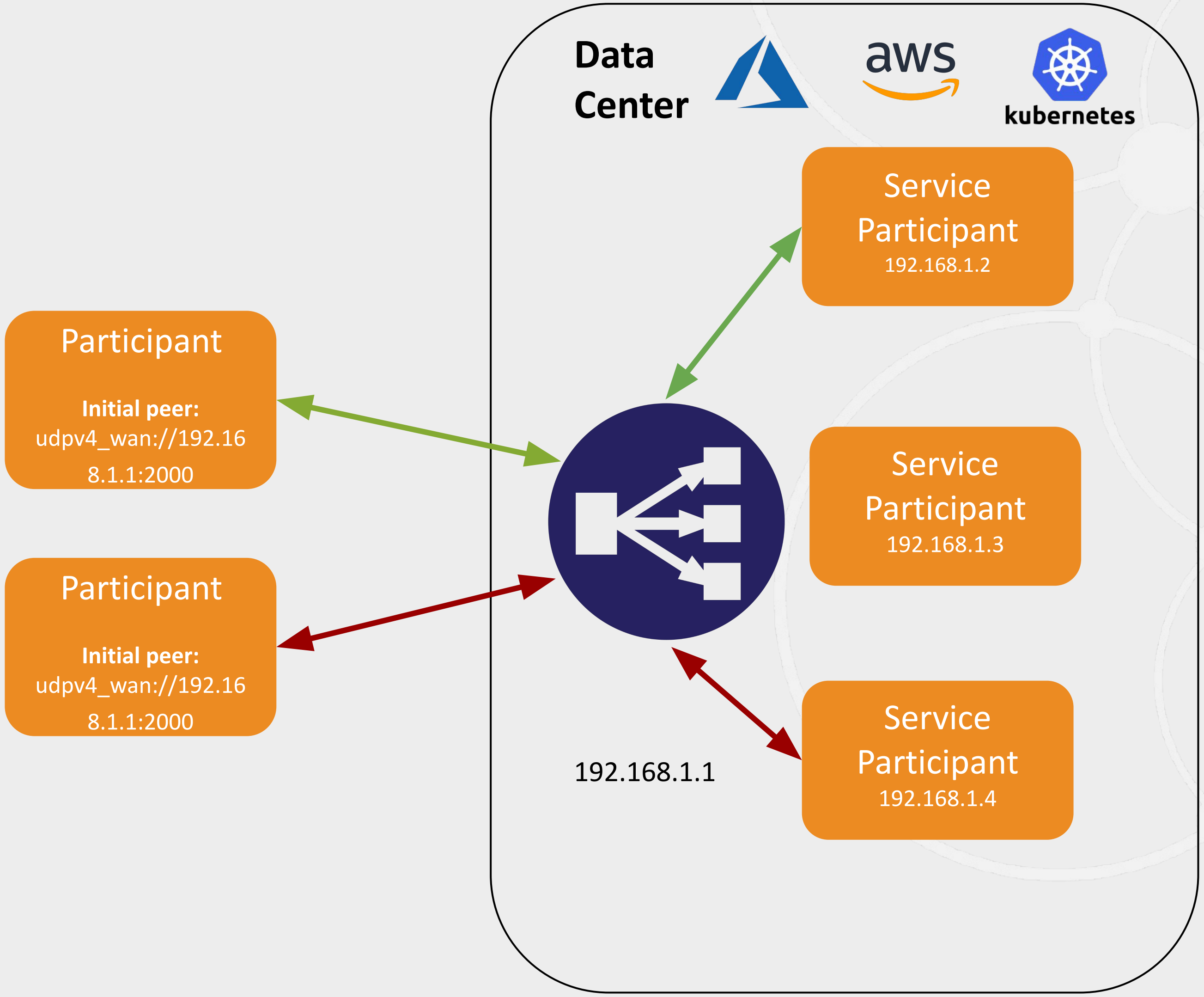
Edge to Cloud (private to public)



The Real-Time WAN Transport and RTI Routing Service provide a scalable solution to connect the edge to the cloud or on-prem data centers

- **Support for multiple gateway component acting as entry points in data centers for edge applications**
- **Ability to send and receive information from the data centers**
- **High-speed and performant connection within data centers using UDPv4, multicast, shared memory and other transports**

Participant Load Balancing Using a Layer 4 Load Balancer



Data Center

aws

kubernetes

Participant

Initial peer: udpv4_wan://192.168.1.1:2000

Participant

Initial peer: udpv4_wan://192.168.1.1:2000

192.168.1.1

Service Participant 192.168.1.2

Service Participant 192.168.1.3

Service Participant 192.168.1.4

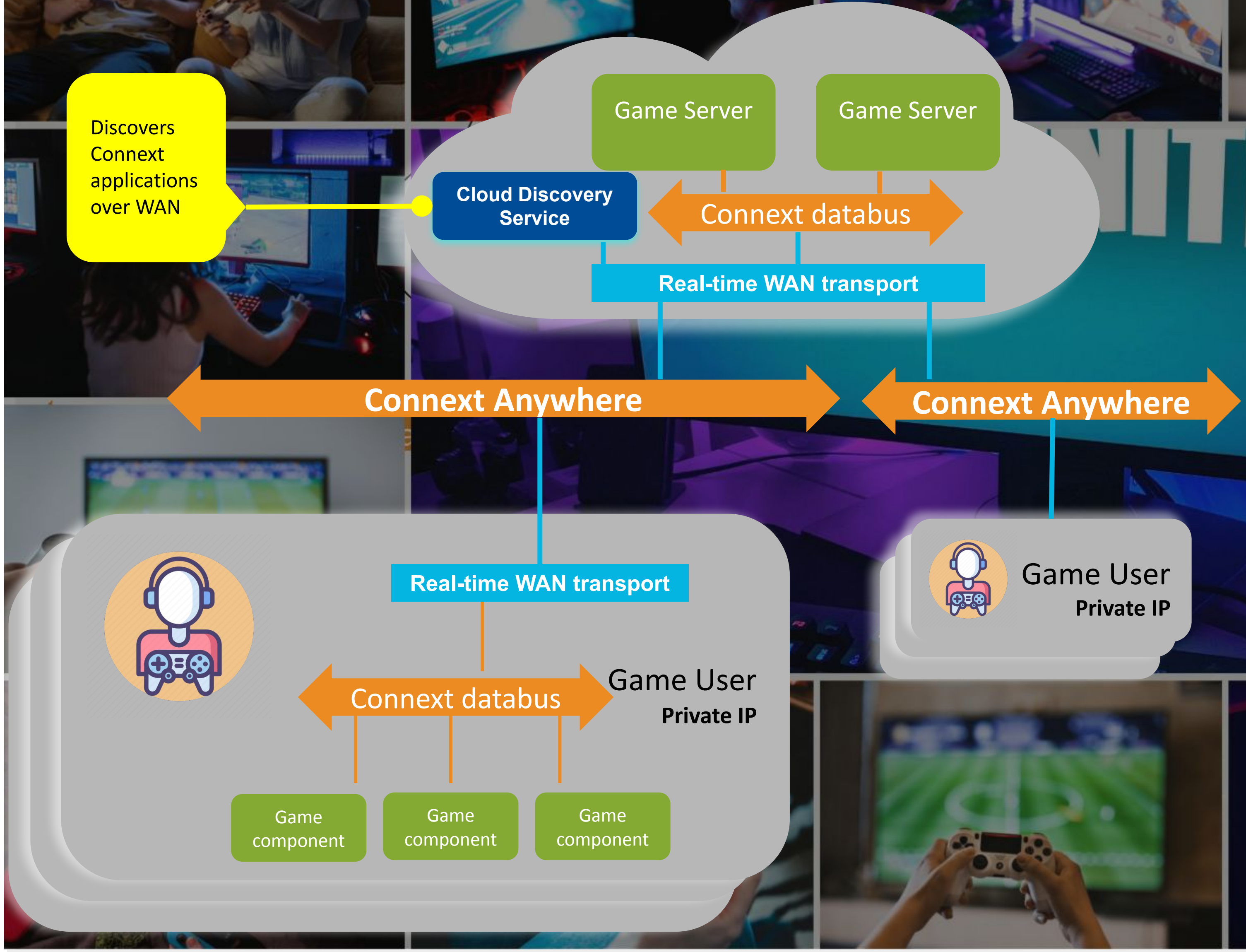
Layer 4 Load Balancing Requires Configuring RWT to use a Single Port

All RTPS traffic send and receive in UDP port 1234

```
<udpv4_wan>
    <comm_ports>
        <default>
            <host>1234</host>
            <public>2345</public>
        </default>
    </comm_ports>
</udpv4_wan>
```

Only for External Participants

Peer-to-Peer Edge-to-Edge (Private to Private)

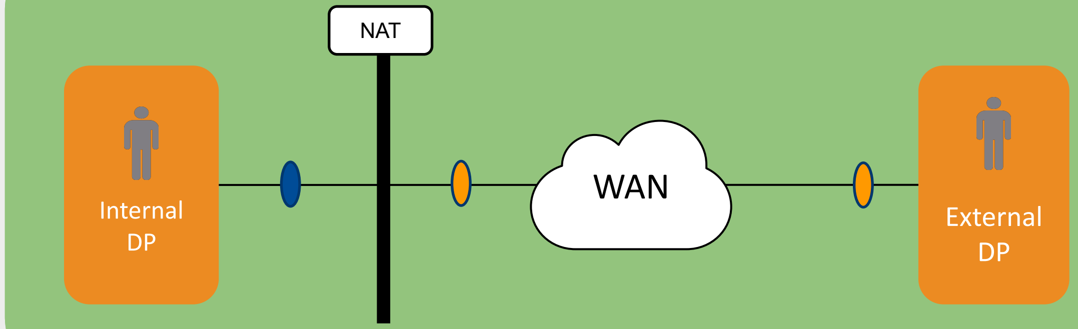


Real-time WAN Transport in combination with Cloud Discovery Service allows peer-to-peer communication* between Connext applications running in the WAN

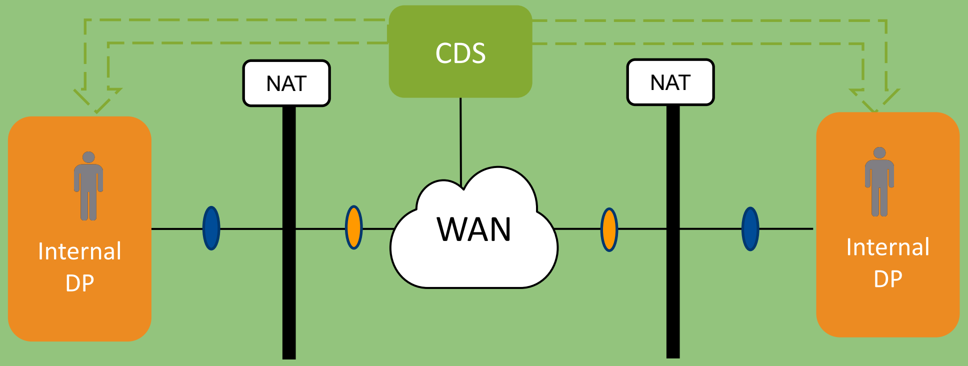
In this configuration, RTI Cloud Discovery Service facilitates discovery between Connext enabled applications across the WAN

** Peer-to-peer communications is possible only in Cone NAT environments*

P2P Communication Scenarios

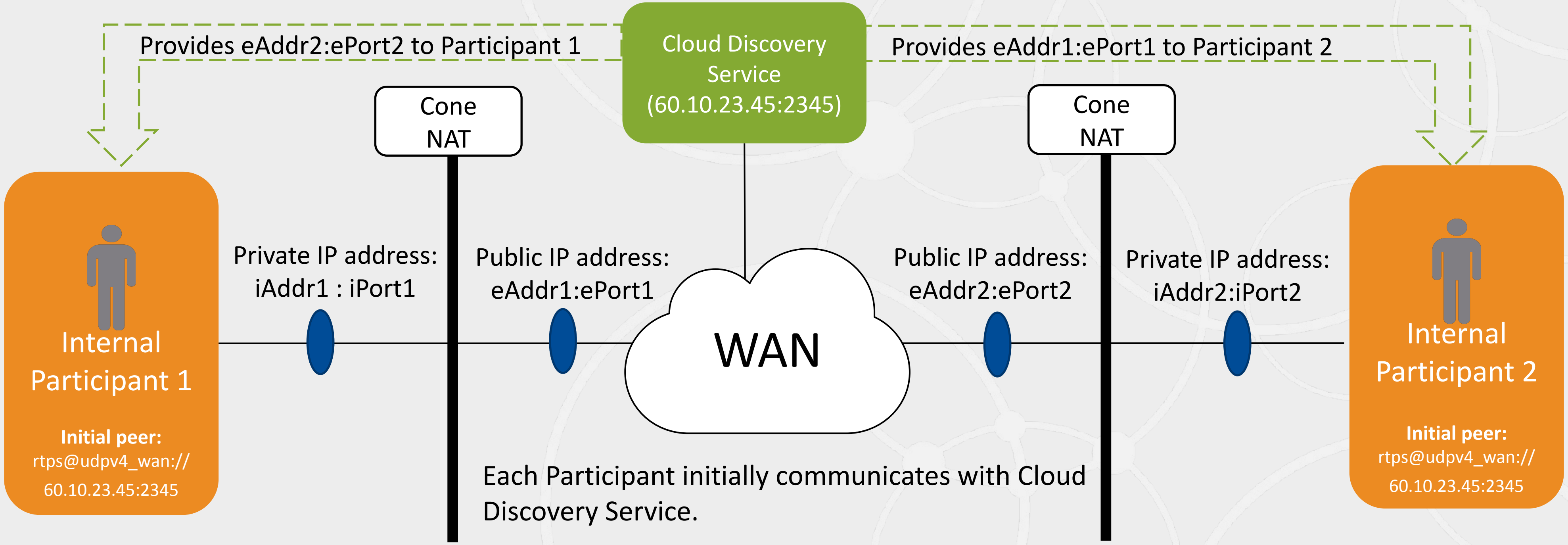


Publicly Reachable
Participant <-> Participant
behind **ANY NAT**



Both Participants Behind
Cone-NATs

Peer-to-Peer With Participants Behind Cone NAT 1/3

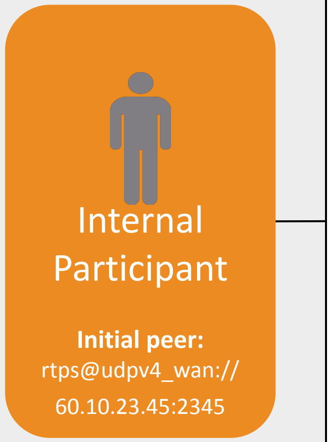


Each Participant initially communicates with Cloud Discovery Service.

Cloud Discovery Service resolves the public addresses of an Internal Participants and it communicates this addresses to other Internal Participants.

Cloud Discovery Service also serves as a directory, so that a participant only needs to know about the CDS public address to connect to multiple peers automatically.

Peer-to-Peer With Participants Behind Cone NAT 2/3



Internal Participant

Initial peer:
rtps@udpv4_wan://
60.10.23.45:2345

```
<dds>
    <qos_profile name="InternalParticipant">
        <participant_qos>
            <transport_builtin>
                <mask>UDPv4_WAN</mask>
            </transport_builtin>
            <discovery>
                <initial_peers>
                    <element>
                        rtps@udpv4_wan://60.10.23.45:2345
                    </element>
                </initial_peers>
            </discovery>
        </participant_qos>
    </qos_profile>
</dds>
```

60.10.23.45:2345 is the public IP address and port of CDS

Peer-to-Peer With Participants Behind Cone NAT 3/3

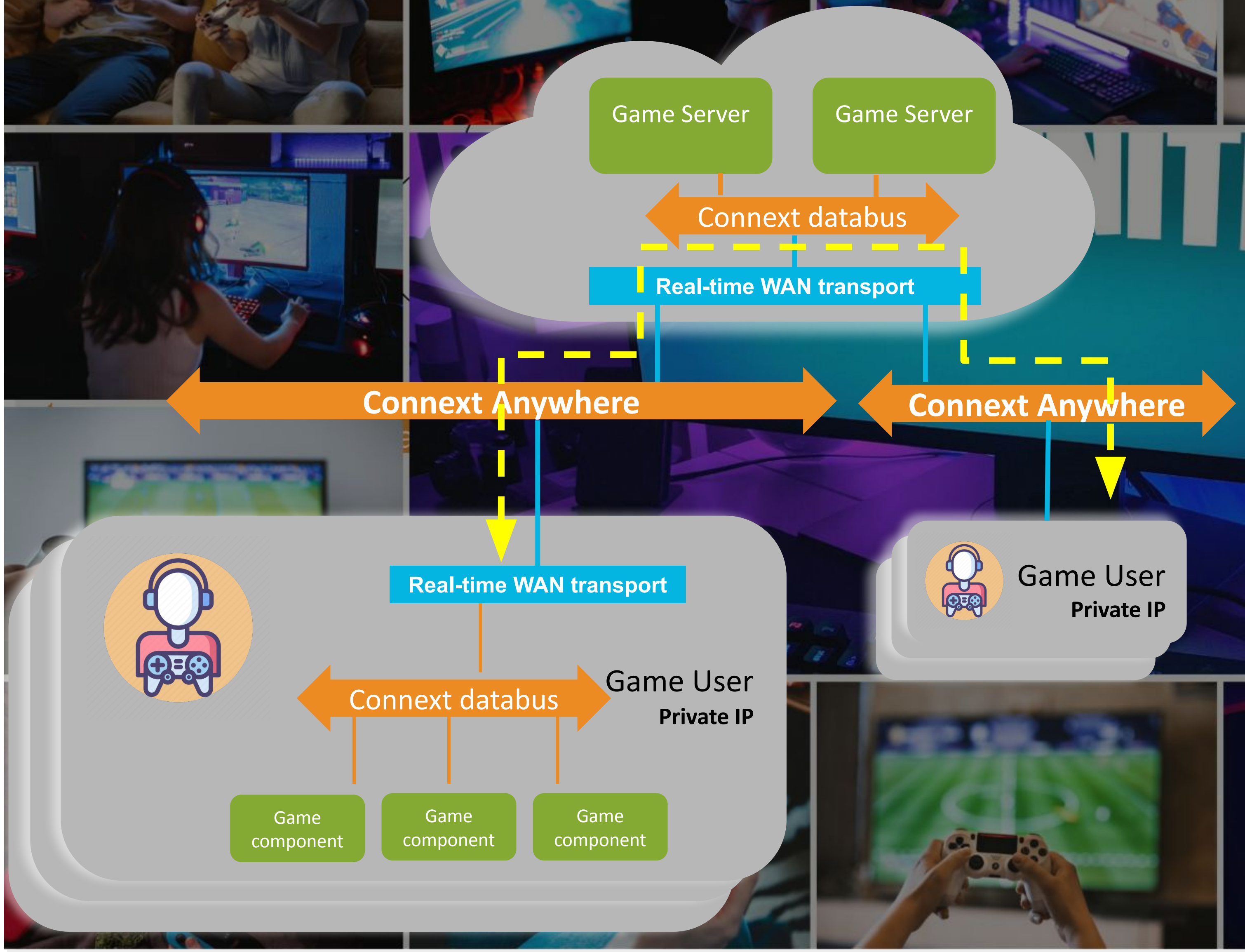
Cloud Discovery Service (60.10.23.45:2345)

```
<dds>
    <cloud_discovery_service name="CDS">
        <transport>
            <element>
                <alias>builtin.udpv4_wan</alias>
                <receive_port>2345</receive_port>
                <property>
                    <element>
                        <name>
                            dds.transport.UDPv4_WAN.builtin.public_address
                        </name>
                        <value>60.10.23.45</value>
                    </element>
                </property>
            </element>
        </transport>
    </cloud_discovery_service>
</dds>
```

60.10.23.45:2345 is the public IP address and port of CDS

Do not forget to configure your router if CDS is behind a NAT!

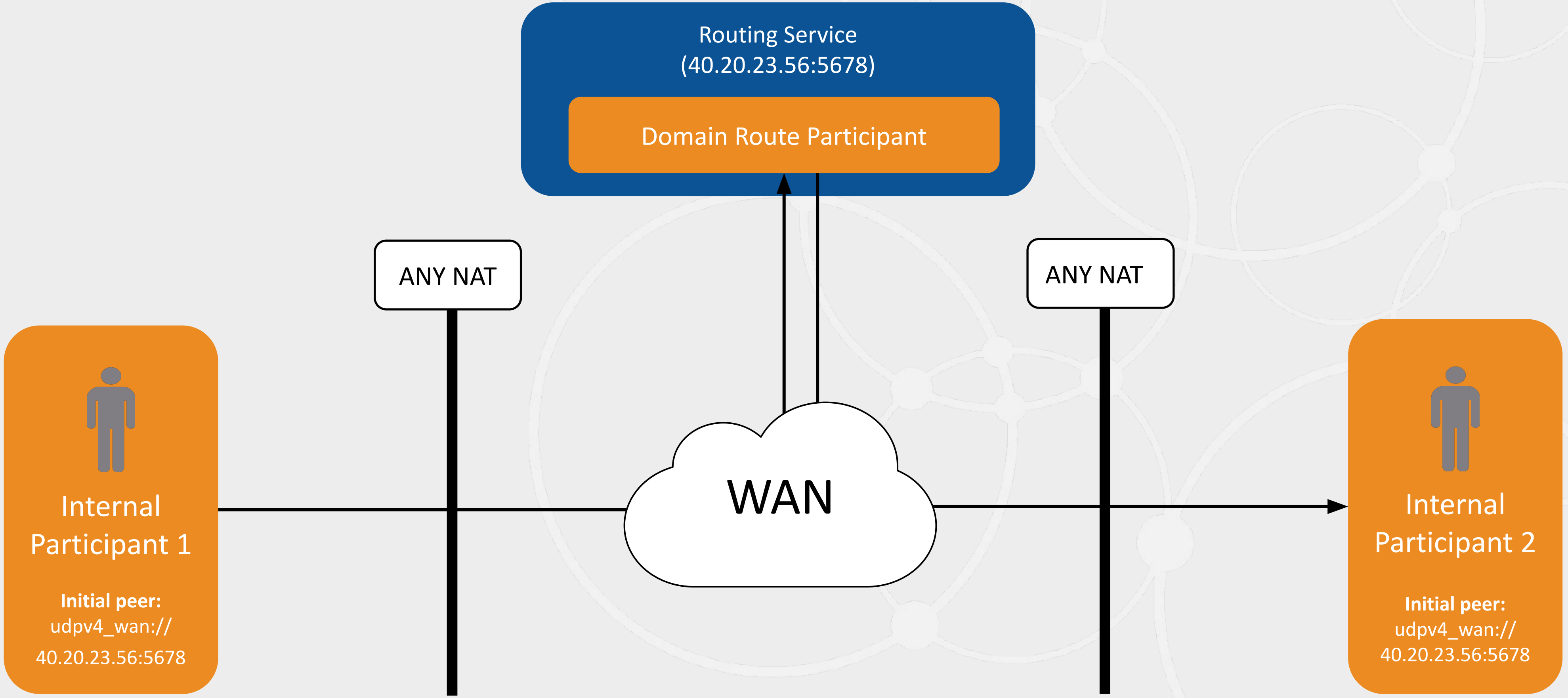
Relayed Edge-to-Edge (Relayed Private to Private)



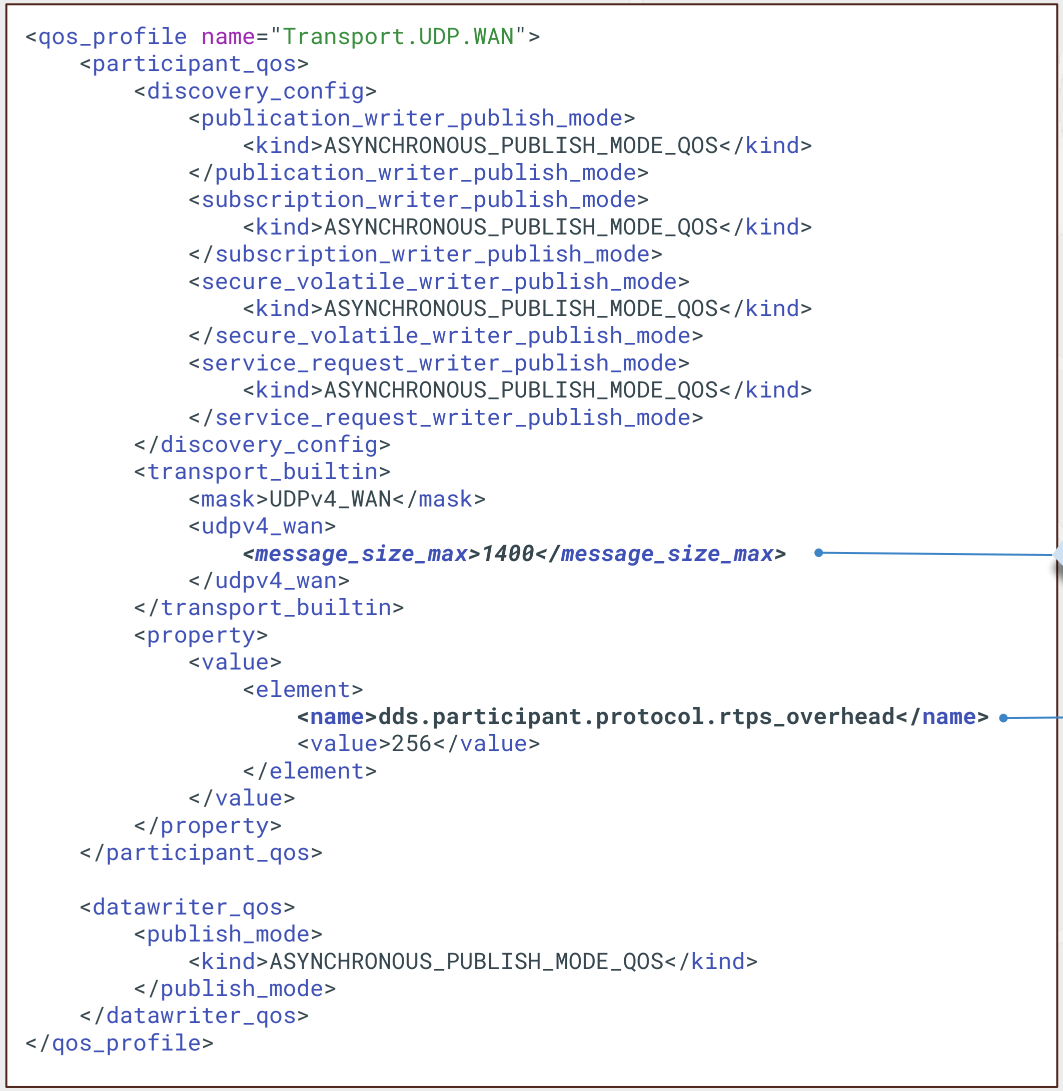
Real-time WAN Transport also supports applications communicating over the WAN under any circumstances, such as NATed environments

Connext-enabled applications can communicate logically peer-to-peer using Routing Service with Real-time WAN Transport as a relay component

Relay Using Routing Service



Avoid IP Fragmentation over Cellular



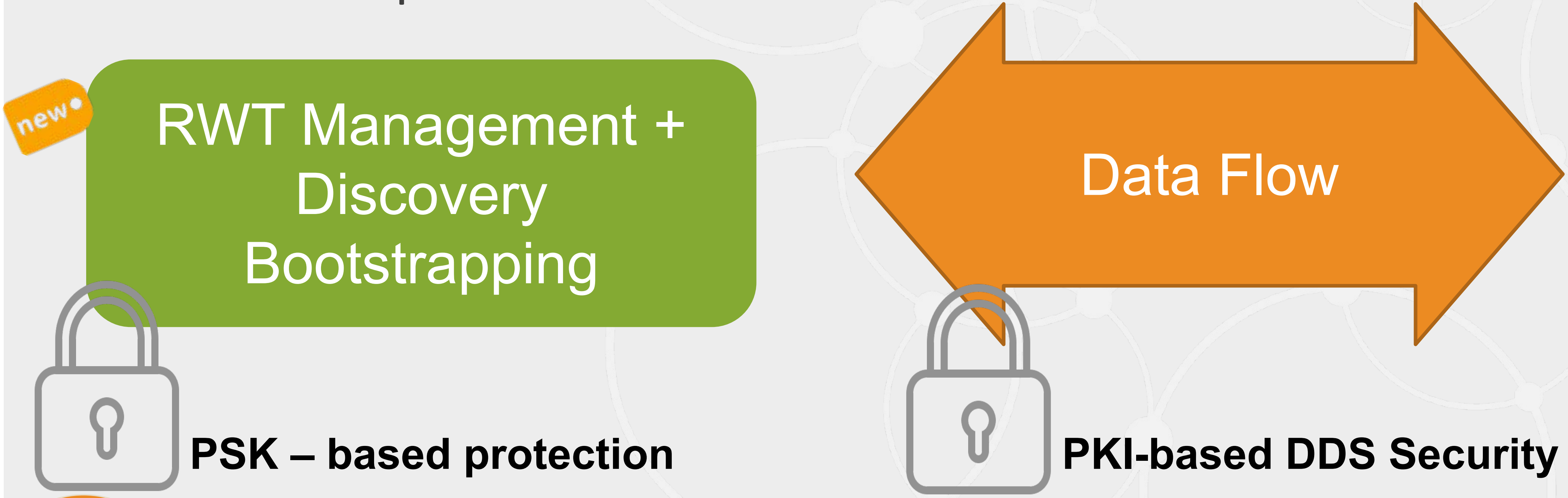
```xml
<qos_profile name="Transport.UDP.WAN">
    <participant_qos>
        <discovery_config>
            <publication_writer_publish_mode>
                <kind>ASYNCHRONOUS_PUBLISH_MODE_QOS</kind>
            </publication_writer_publish_mode>
            <subscription_writer_publish_mode>
                <kind>ASYNCHRONOUS_PUBLISH_MODE_QOS</kind>
            </subscription_writer_publish_mode>
            <secure_volatile_writer_publish_mode>
                <kind>ASYNCHRONOUS_PUBLISH_MODE_QOS</kind>
            </secure_volatile_writer_publish_mode>
            <service_request_writer_publish_mode>
                <kind>ASYNCHRONOUS_PUBLISH_MODE_QOS</kind>
            </service_request_writer_publish_mode>
        </discovery_config>
        <transport_builtin>
            <mask>UDPv4_WAN</mask>
            <udpv4_wan>
                <message_size_max>1400</message_size_max>
            </udpv4_wan>
        </transport_builtin>
        <property>
            <value>
                <element>
                    <name>dds.participant.protocol.rtps_overhead</name>
                    <value>256</value>
                </element>
            </value>
        </property>
    </participant_qos>

    <datawriter_qos>
        <publish_mode>
            <kind>ASYNCHRONOUS_PUBLISH_MODE_QOS</kind>
        </publish_mode>
    </datawriter_qos>
</qos_profile>
```

IP fragmentation disabled by setting transport MTU to 1400 bytes

256 bytes for metadata. User payload up to 1144 bytes

WAN Security: Connext approach

- Two level protection:



new

RWT Management + Discovery Bootstrapping

Data Flow

PSK – based protection

PKI-based DDS Security

6.1 behavior

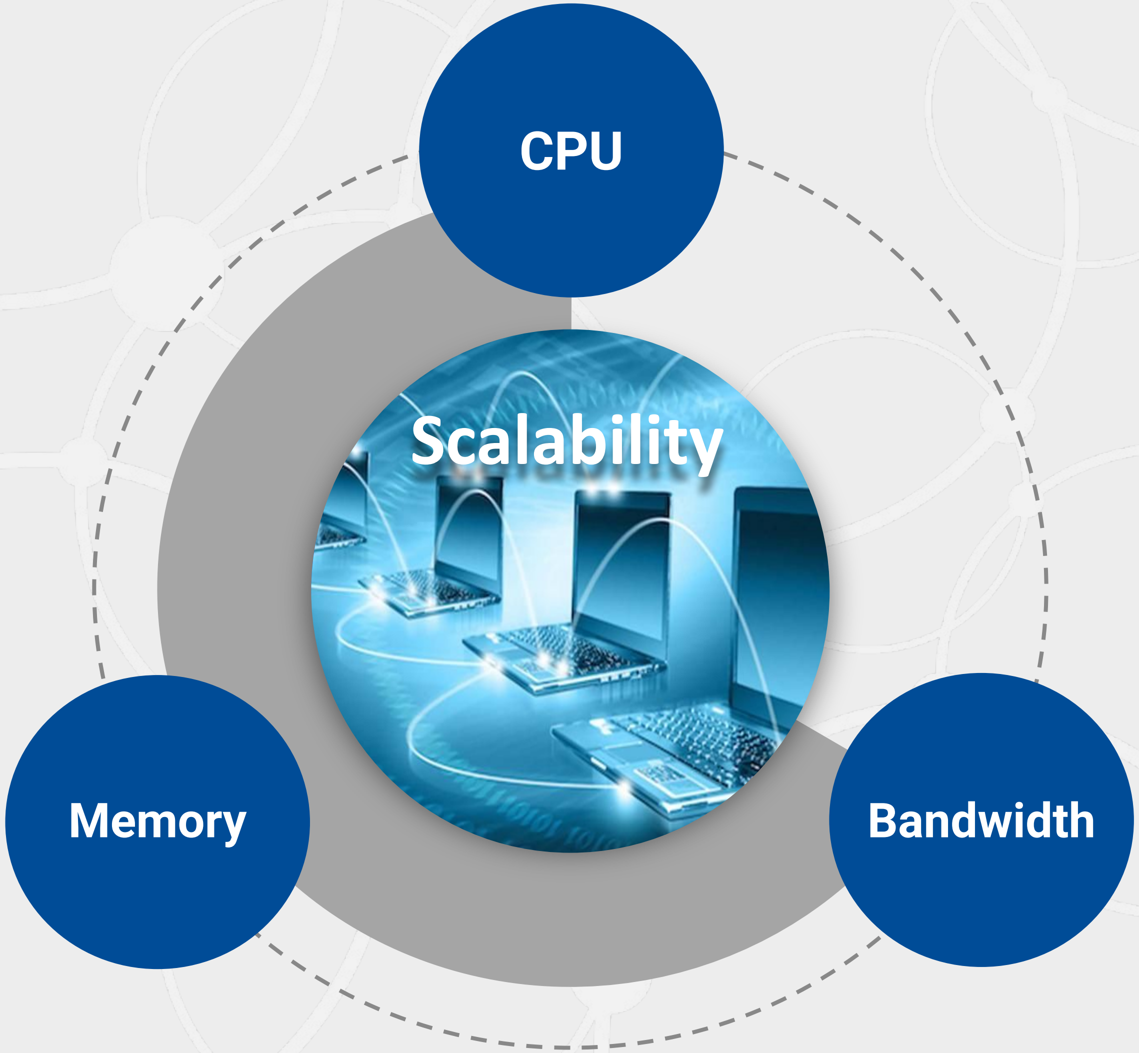
Agenda

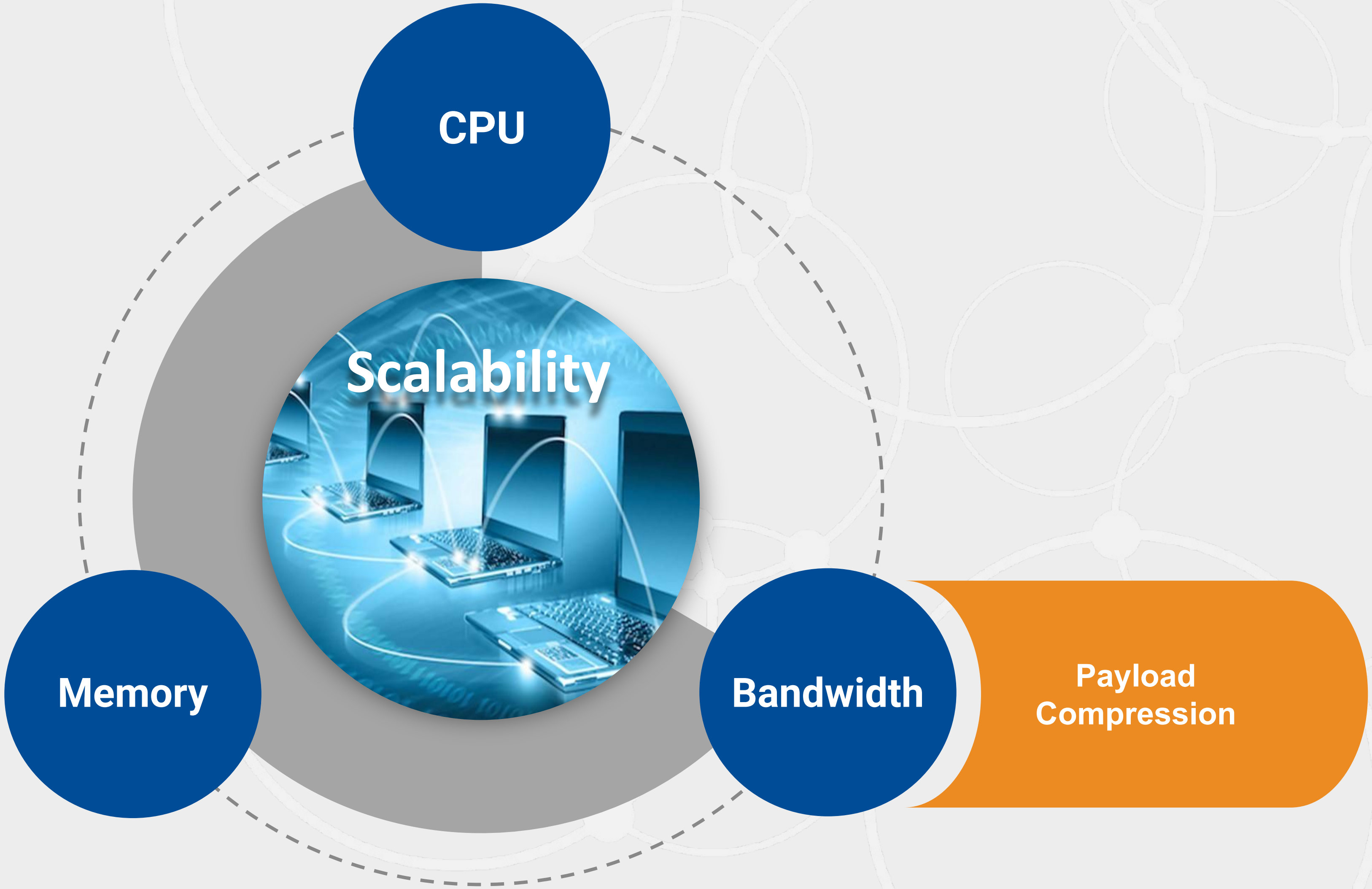
- WAN Connectivity
- **Performance and Scalability**
- Containerization

Trends & Needs

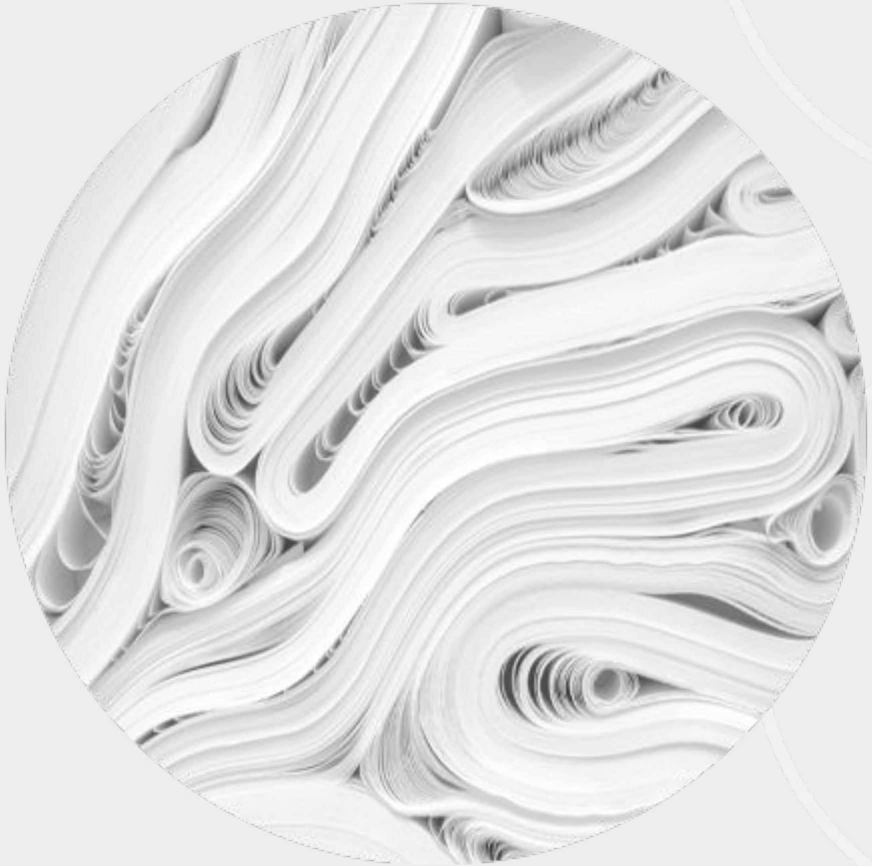
- Data being exchanged are larger
- Scale of systems are increasing
- Systems are more dynamic with many unknowns (numbers of endpoints, amount of data)

Connext allows *You* to build systems that continue to perform as your resource needs evolve and grow




CPU
Scalability
Memory
Bandwidth
Payload Compression

Out-of-Box Payload Compression



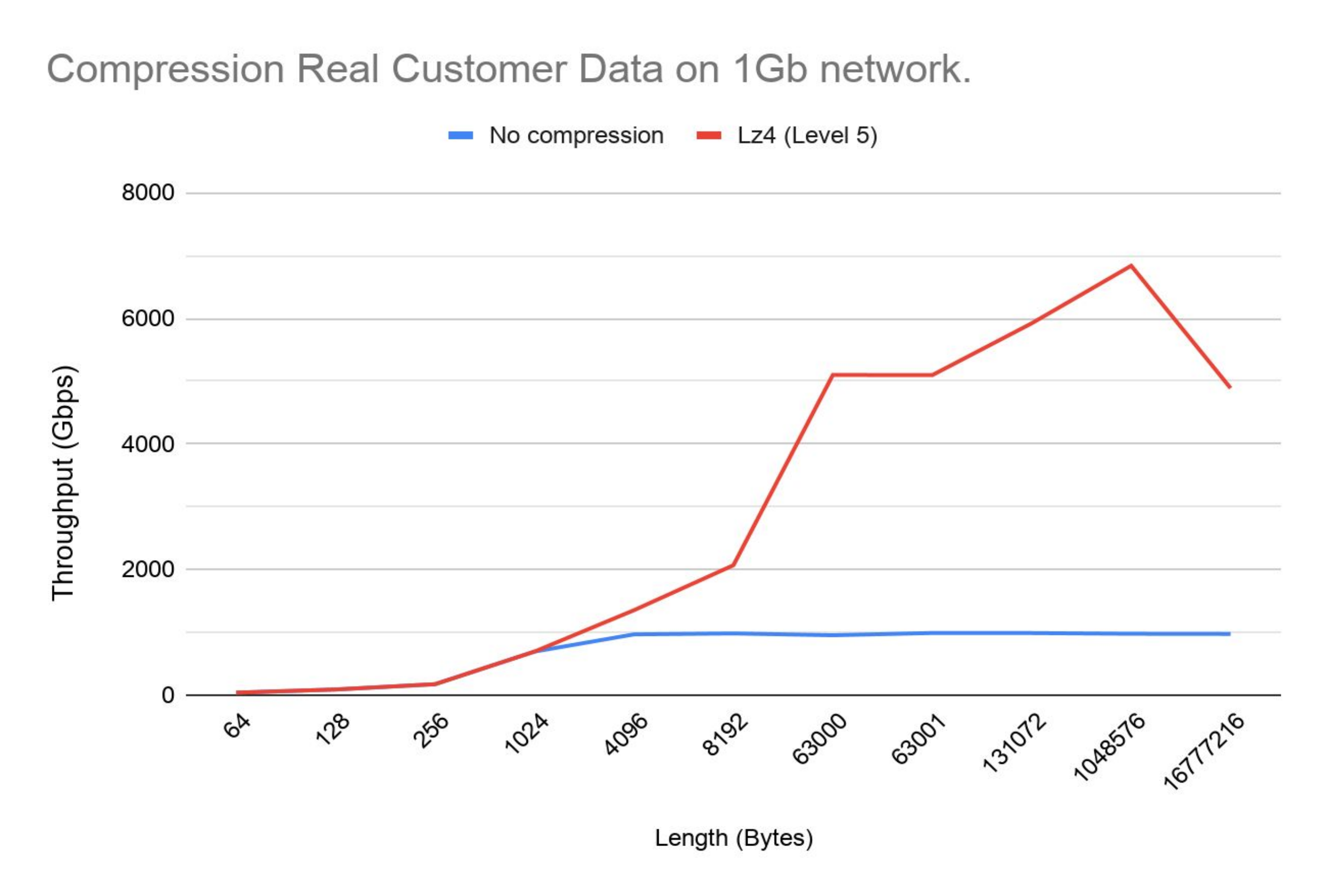
Core Libraries now offer

- 3 lossless compression algorithms
  - zlib, bzip2, LZ4
- Enabling compression through XML
- Payload size based threshold to enable compression
- Configurable compression level

Compressing data can speed up data transfer, and decrease costs of network bandwidth

* Expected for next release

Compression Increases Throughput



Throughput increase on low transmission networks rate (1Gb).

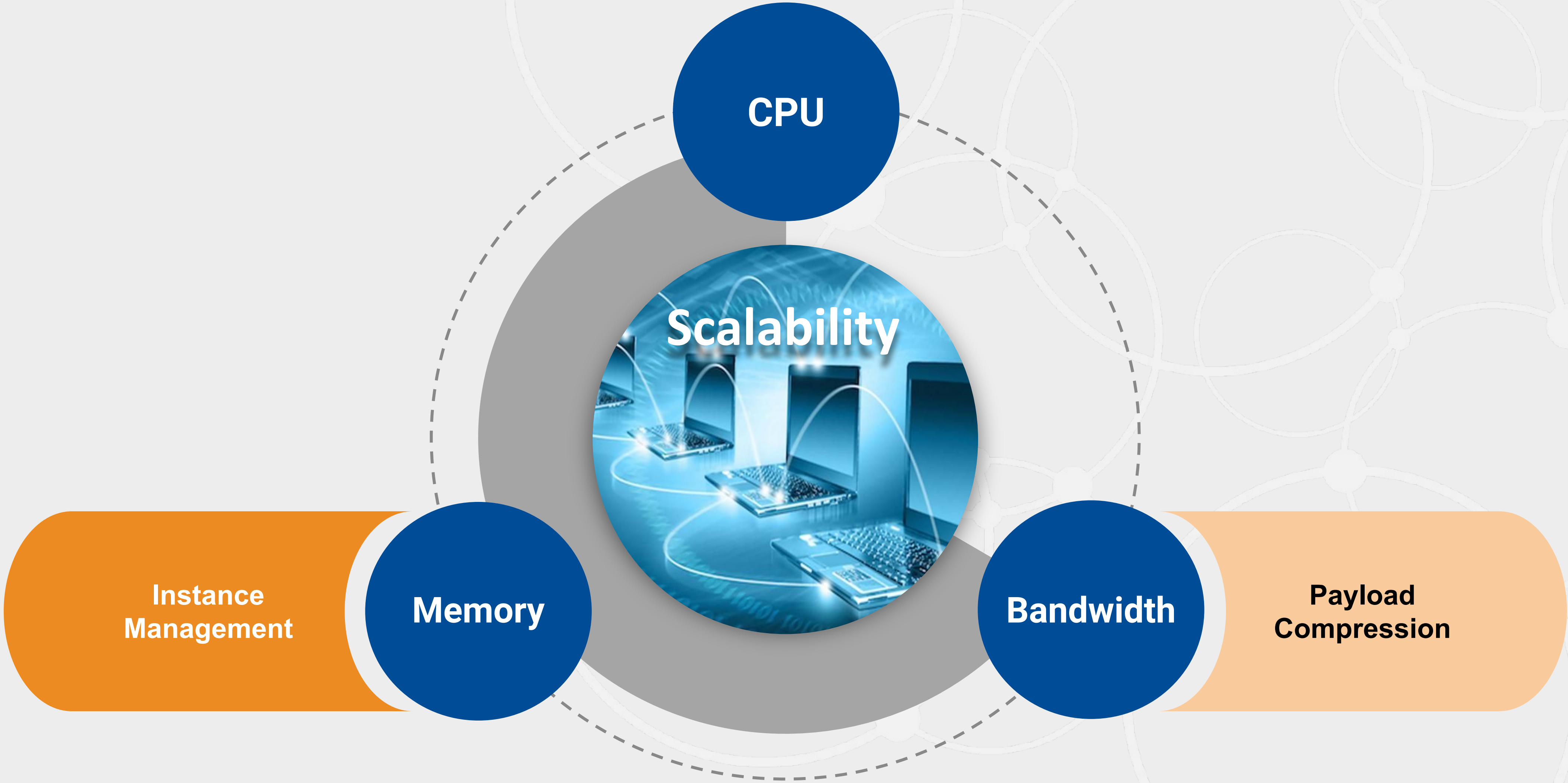
Compression Considerations

Enable when

- → You want to send data on low bandwidth networks (less than 1Gbps)
- → You want to increase throughput (use LZ4)
- → You want to reduce bandwidth network usage (use ZLIB)
- → You want to send large data samples, or at least over 4Kb

Do not enable when

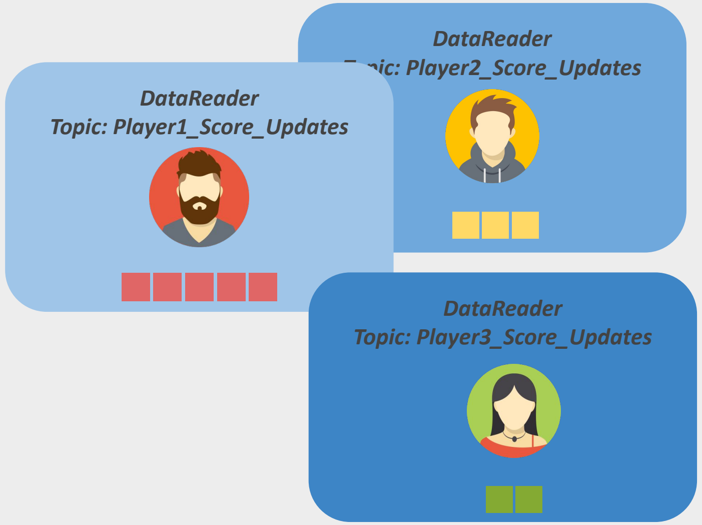
- → You have low latency requirements
- → You are transmitting data that is not compressible (e.g. already compressed images, video)
- → You have limited CPU
- → You are transmitting small data samples


CPU
Scalability
Instance Management
Memory
Bandwidth
Payload Compression

Instances Offer Scalability in Representing Real-World Objects

Alternatives to represent real-world objects

Separate Topics Per Object

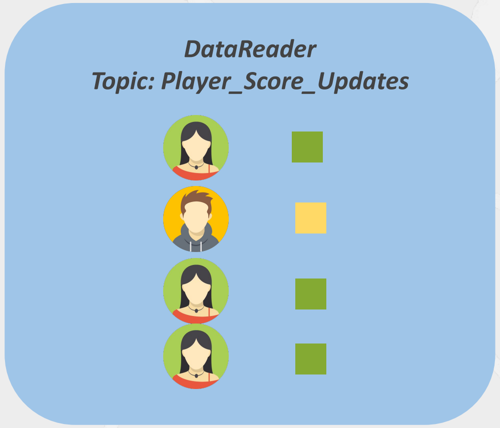


DataReader
Topic: Player2_Score_Updates

DataReader
Topic: Player1_Score_Updates

DataReader
Topic: Player3_Score_Updates

Attribute in a Topic to track Object ID

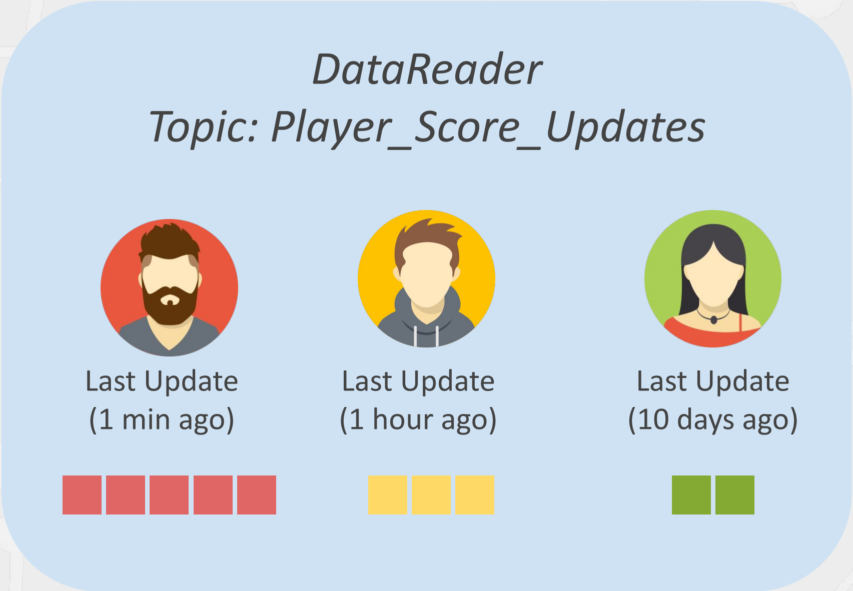


DataReader
Topic: Player_Score_Updates

More Memory and Discovery Time

Burden on application to manage lifecycle and code to maintain state of each object

Instances



DataReader
Topic: Player_Score_Updates

Last Update (1 min ago)

Last Update (1 hour ago)

Last Update (10 days ago)

Less Memory and Discovery Time

Lifecycle Management

QoS Configurable Per Instance

Instances Configurable With Individual QoS

Some QoS policies are applied per instance, and other QoS policies configure instance management

QoS Applied per Instance

DEADLINE
DESTINATION_ORDER
HISTORY
new DURABILITY
PRESENTATION
TIME_BASED_FILTER

QoS to Configure Instance Management

DataReader

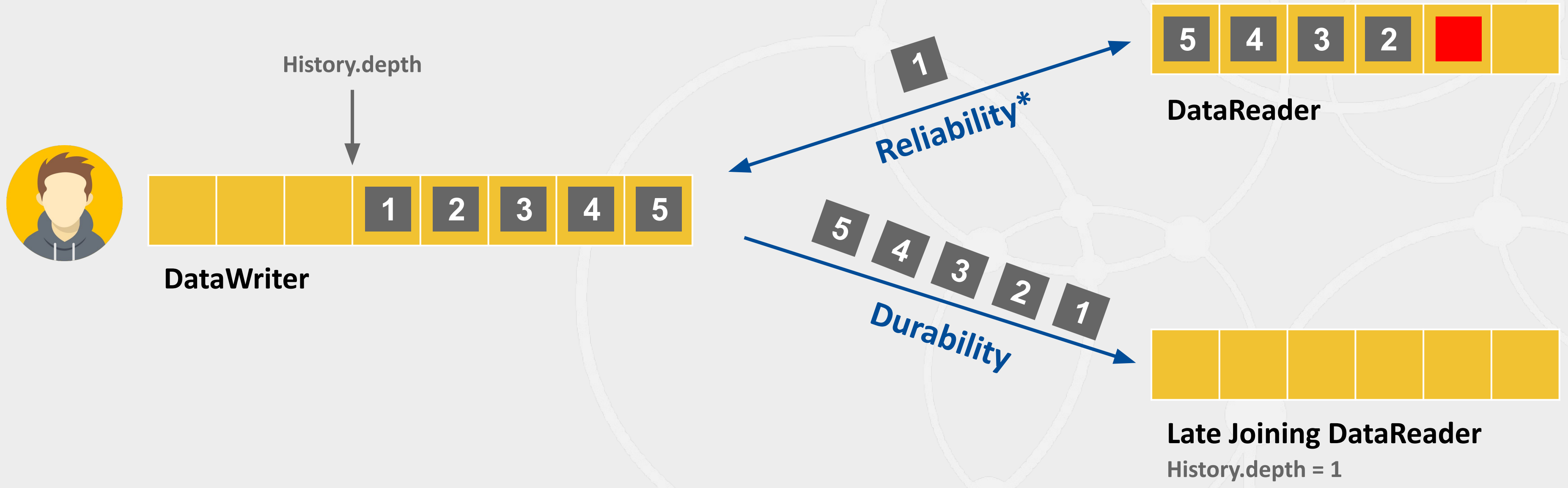
OWNERSHIP
RESOURCE_LIMITS
DATA_READER_PROTOCOL
new DATA_READER_RESOURCE_LIMITS
READER_DATA_LIFECYCLE

DataWriter

OWNERSHIP
RESOURCE_LIMITS
OWNERSHIP_STRENGTH
DATA_WRITER_RESOURCE_LIMITS
WRITER_DATA_LIFECYCLE
DATA_WRITER_PROTOCOL

Challenge: Tight Coupling of Durability & Reliability

Durability: ability to update newly joined applications with historical data

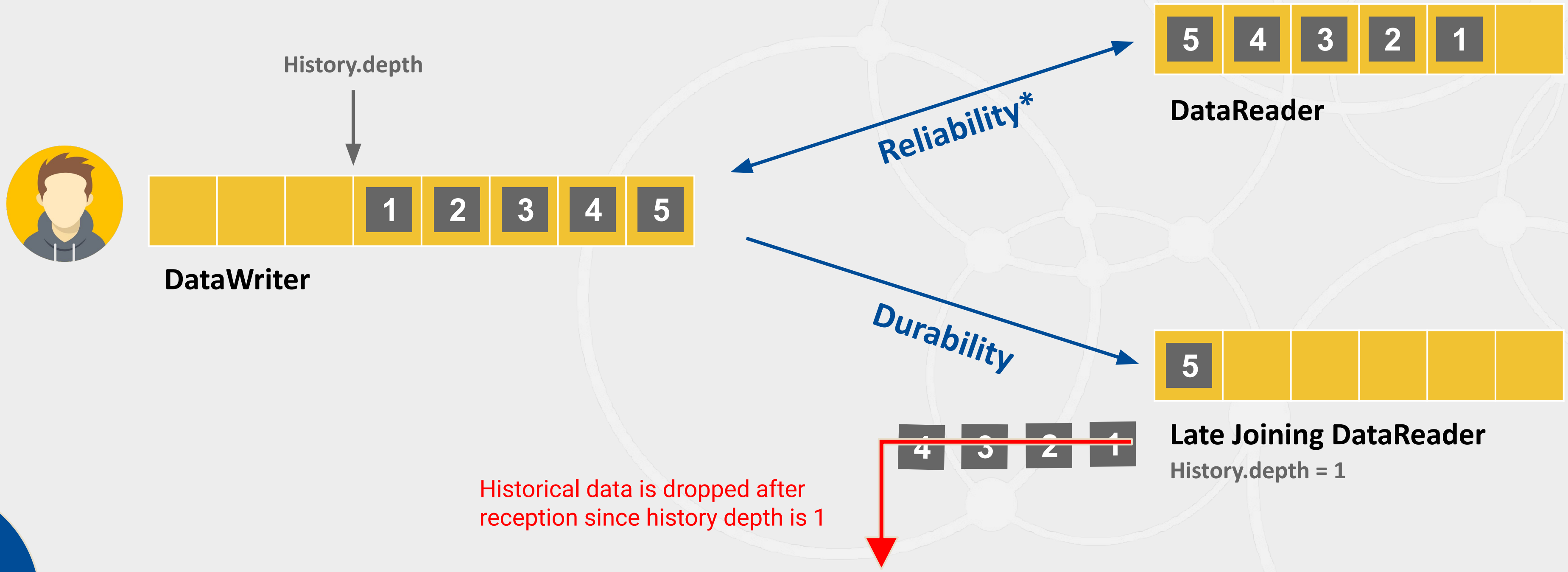


< 6.1 behavior

* Reliability supports data loss on the wire using historical data stored at the DataWriter

Challenge: Tight Coupling of Durability & Reliability

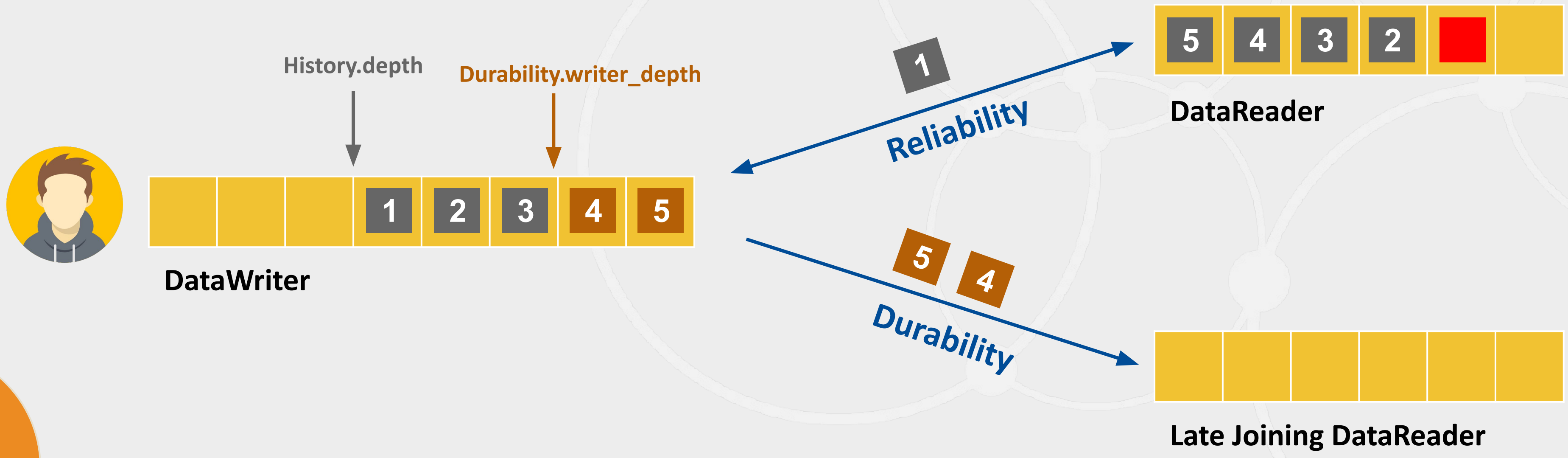
Durability: ability to update newly joined applications with historical data



< 6.1
behavior

Decoupled Durability from Reliability

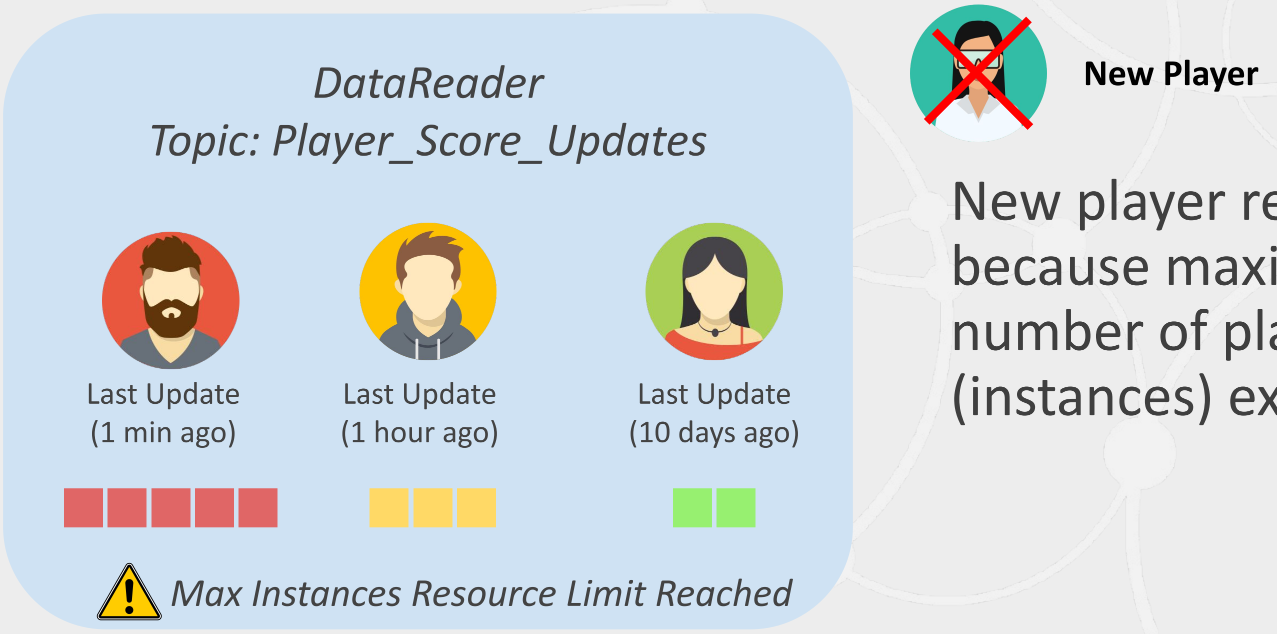
QoS parameter	Purpose
new durability.writer_depth	How many samples per instance to send to late-joining DataReaders
history.depth	How many samples per instance to store to repair data loss



6.1 behavior

Tracking and managing resources for multiple objects is non-trivial

- All resource limits are unlimited by default. This can lead to:
  - Users setting a fixed number of instances to bound growth
  - Not being able to reclaim resources from unused instances to accept new instances



DataReader
Topic: Player_Score_Updates

Last Update
(1 min ago)

Last Update
(1 hour ago)

Last Update
(10 days ago)

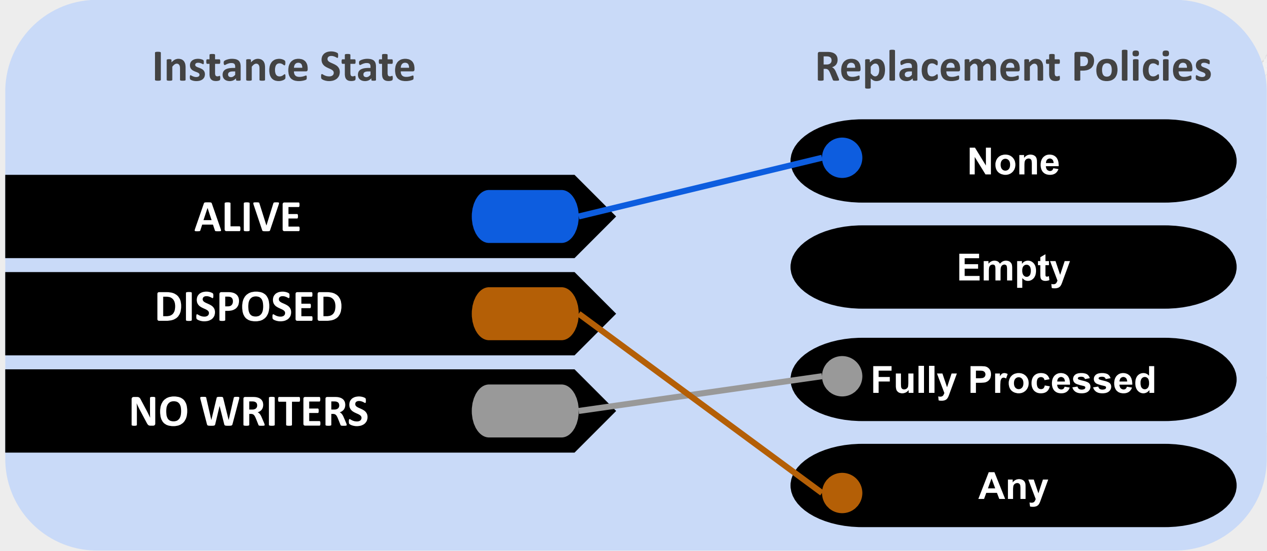
⚠ *Max Instances Resource Limit Reached*

New Player

New player rejected because maximum number of players (instances) exceeded

New Instance Replacement Policy

- New instance_replacement field in the DataReaderResourceLimitsQosPolicy:
  - User can configure which instances can be replaced when max_instances resource limit is hit
  - Replacement policy can be configured per-instance-state
  - instance replacement starts with the least-recently-updated (LRU) instance that matches the allowed criteria

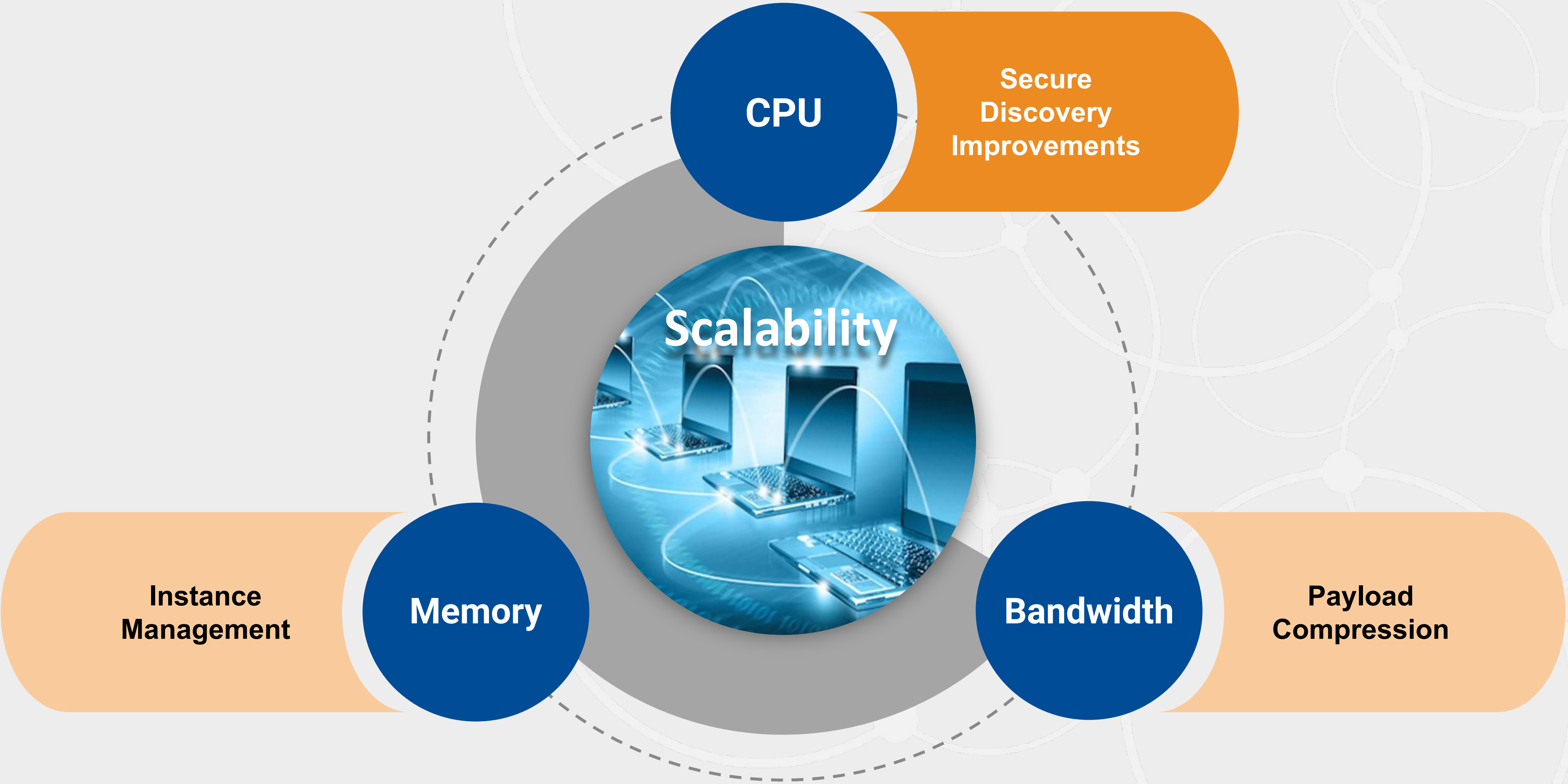


Do not replace an instance in this instance state

Replace only if the reader queue for this instance is empty

Replace only if the reader queue for this instance is fully processed

Replace regardless of the state of this instance's reader queue



Scalability

CPU

Secure Discovery Improvements

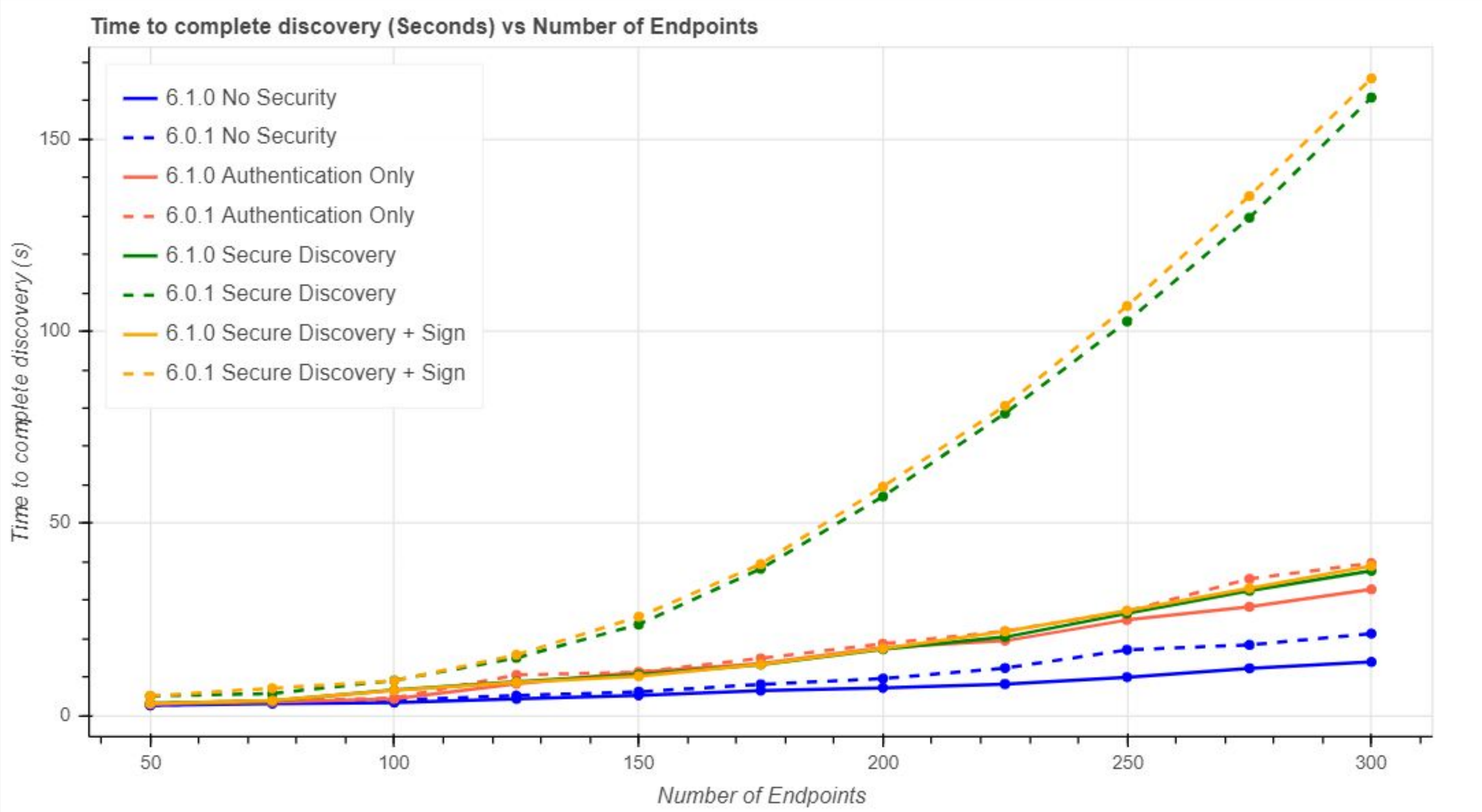
Memory

Instance Management

Bandwidth

Payload Compression

Faster Secure Discovery



Agenda

- WAN Connectivity
- Performance and Scalability
- <u>Containerization</u>

Connext on Kubernetes

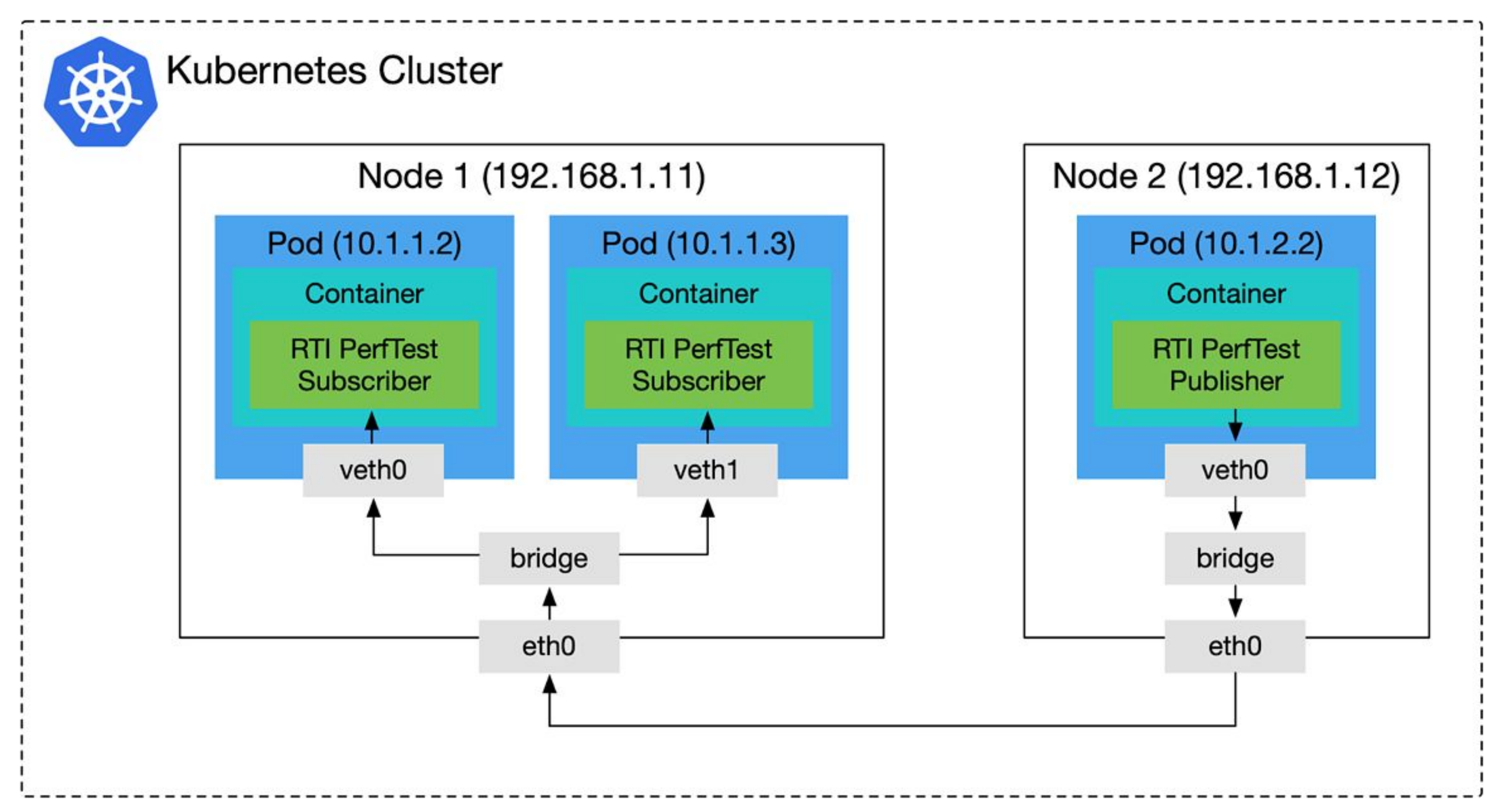
What is Kubernetes?

- Kubernetes is an **orchestrator** for containerized apps.
- So, what is an orchestrator?
- An orchestrator is a system that **deploys** and **manages** distributed applications.
- What an orchestrator does.
  - scale it up and down
  - perform updates and rollbacks
  - self-heal and more

How is Kubernetes relevant to Connext?

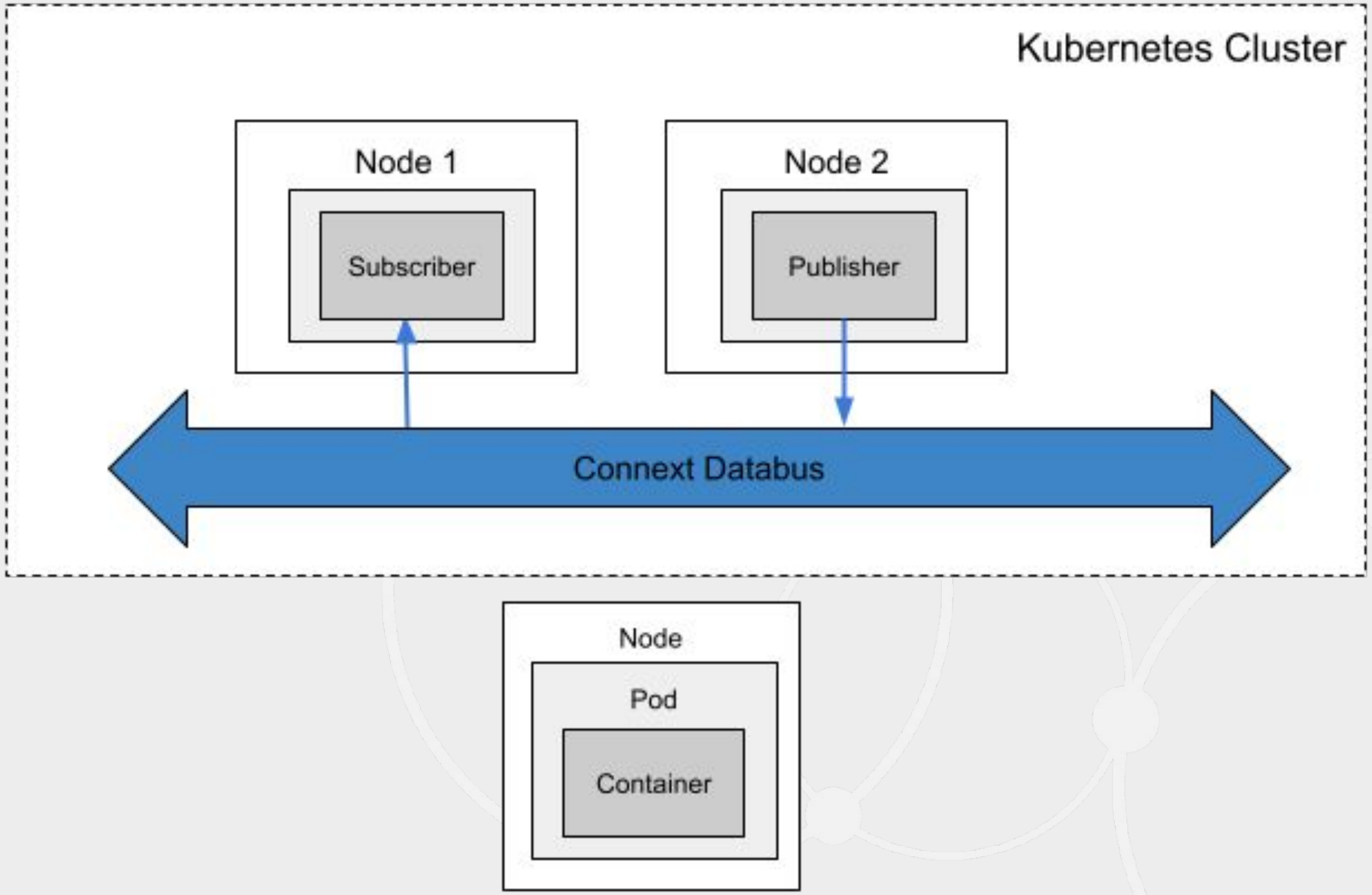
- Deploying and managing large-scale distributed systems is **complex**.
- Deploying such a system with **Connext** is **no exception**.
- **Potential issues** with Kubernetes
  - Discovery
  - External communications
  - Load balancing

Kubernetes Networking



Use Cases

Use Case 1 - Communications Inside a K8s Cluster



https://github.com/rticommunity/kubernetes-examples/tree/master/ddsping

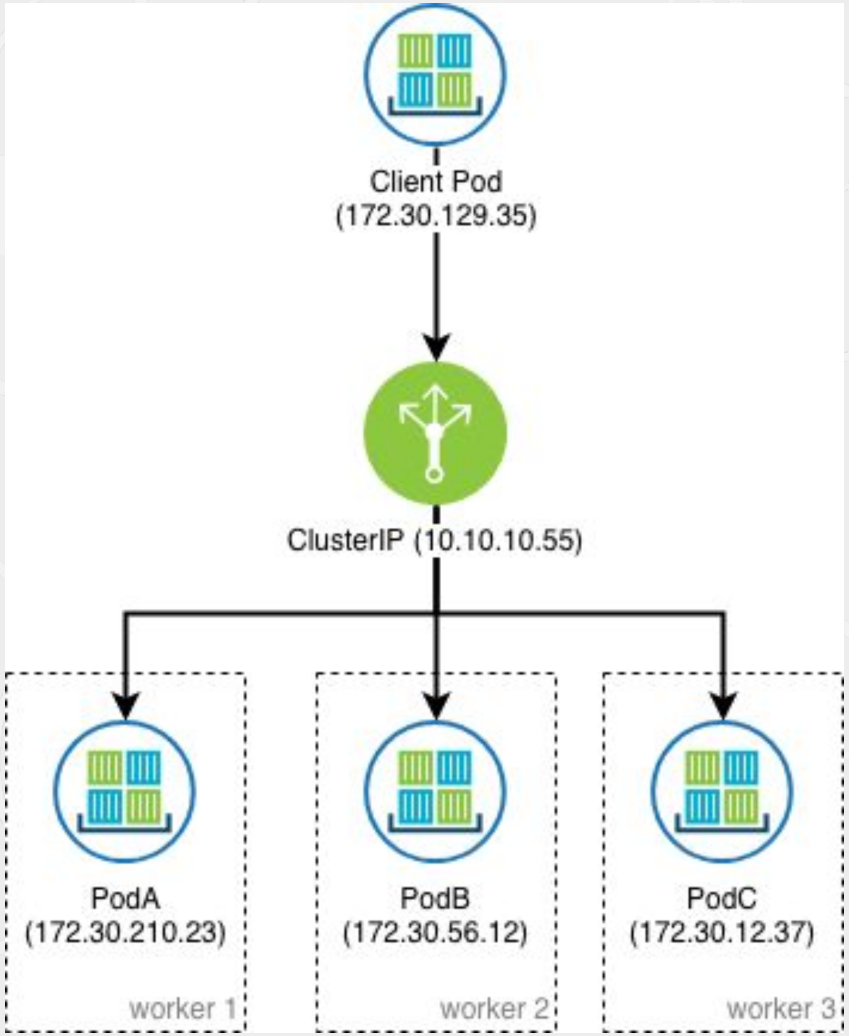
Kubernetes Service

- Pod IPs are **ephemeral**
  - if a pod crashes or is deleted and a new pod is created in its place, it most likely receives a new IP address.
- If a pod needs to communicate with another pod, it needs **a way to discover the IP address of the pod**.
- **Kubernetes services** provide a mechanism for **locating other pods (stable domain name/IP address)**.

https://www.ibm.com/docs/en/cloud-private/3.2.0?topic=networking-kubernetes-service-types

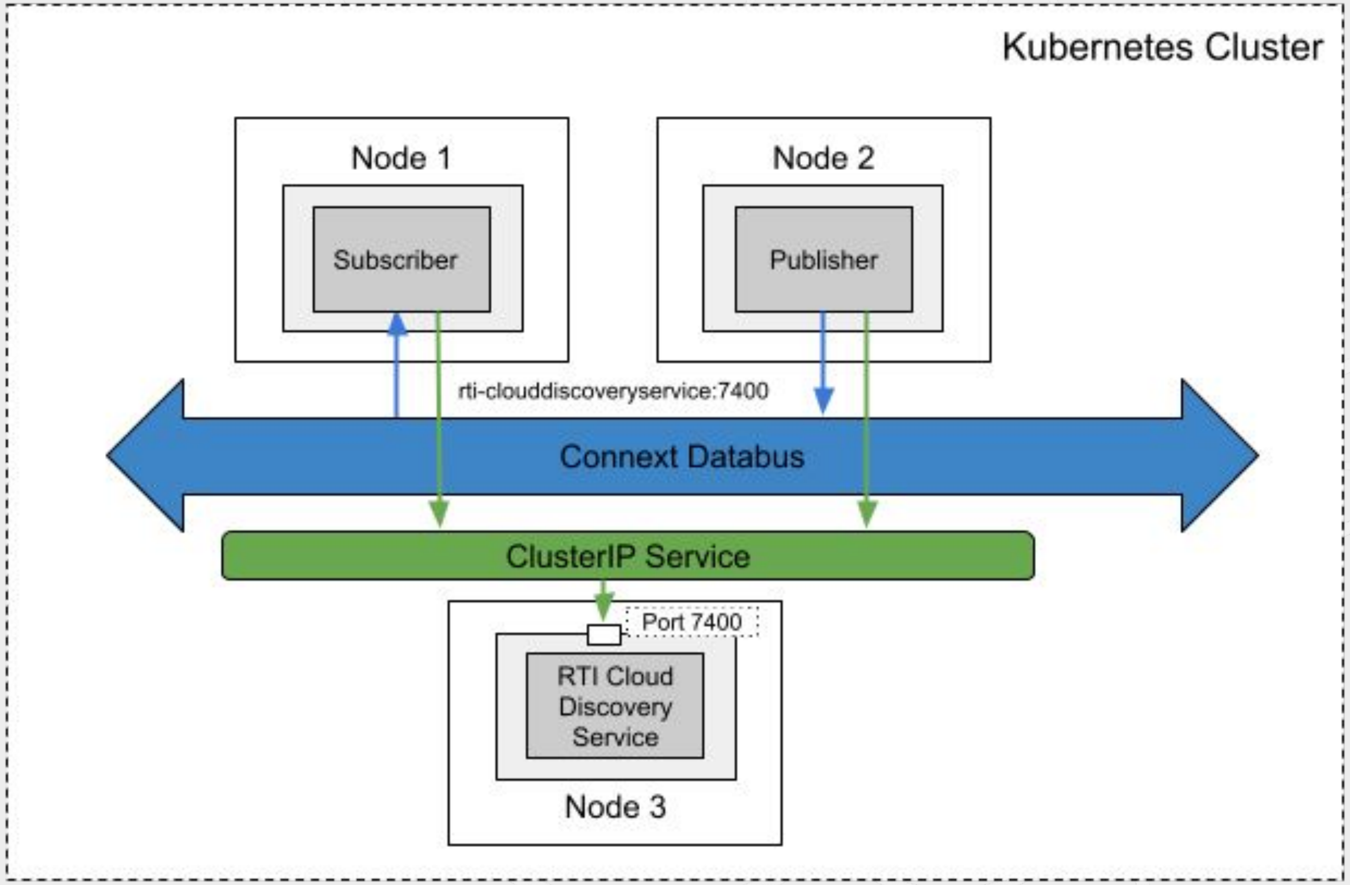
Cluster IP Service

- A ClusterIP service is the **default** Kubernetes service.
- It provides **a domain name/ IP address** that other apps inside the cluster can access.
- ClusterIP provides a **load-balanced IP address**.



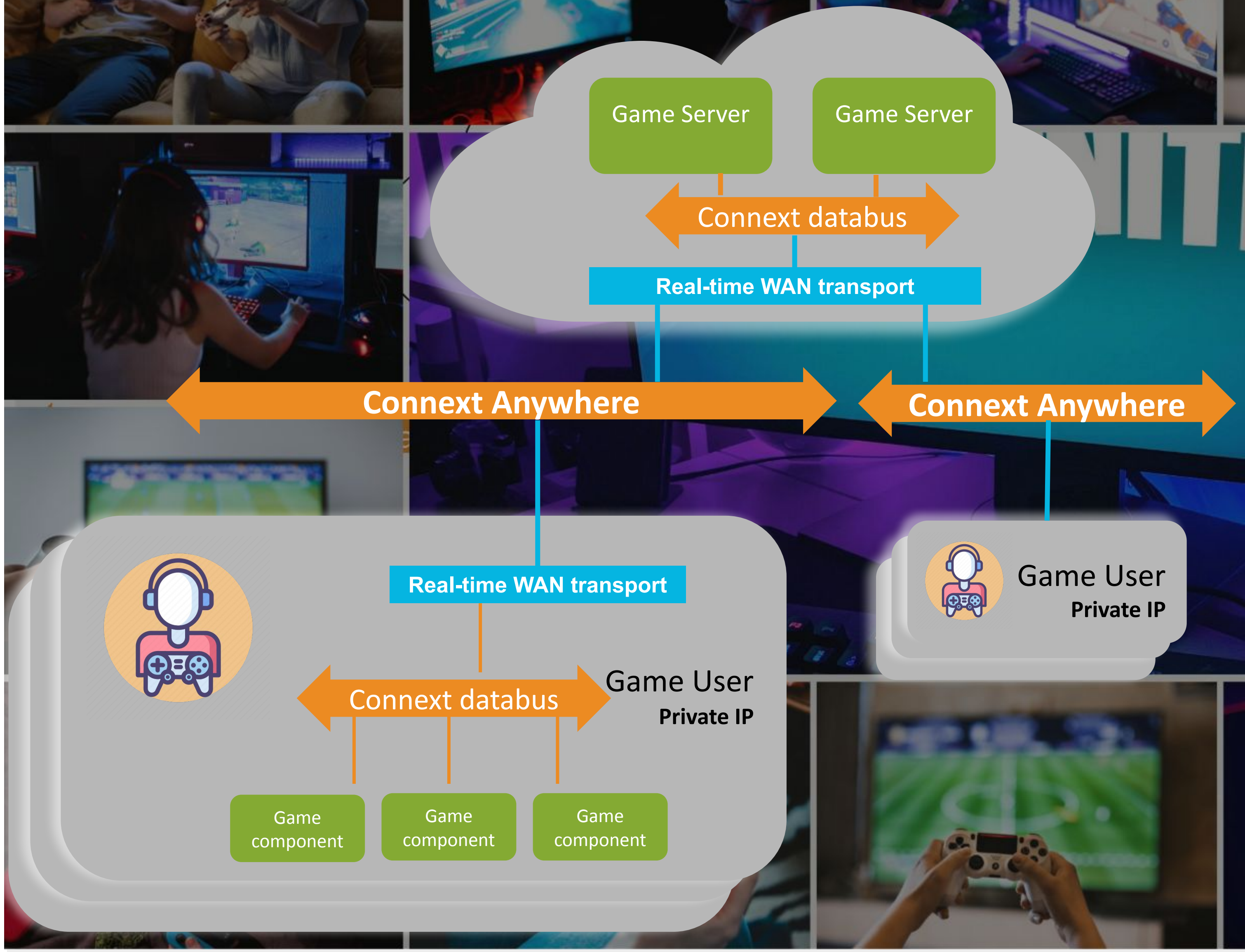
https://www.ibm.com/docs/en/cloud-private/3.2.0?topic=networking-kubernetes-service-types

Use Case 2 - Using CDS for Discovery



https://github.com/rticommunity/kubernetes-examples/tree/master/ddsping_cds

Edge to Cloud (private to public)

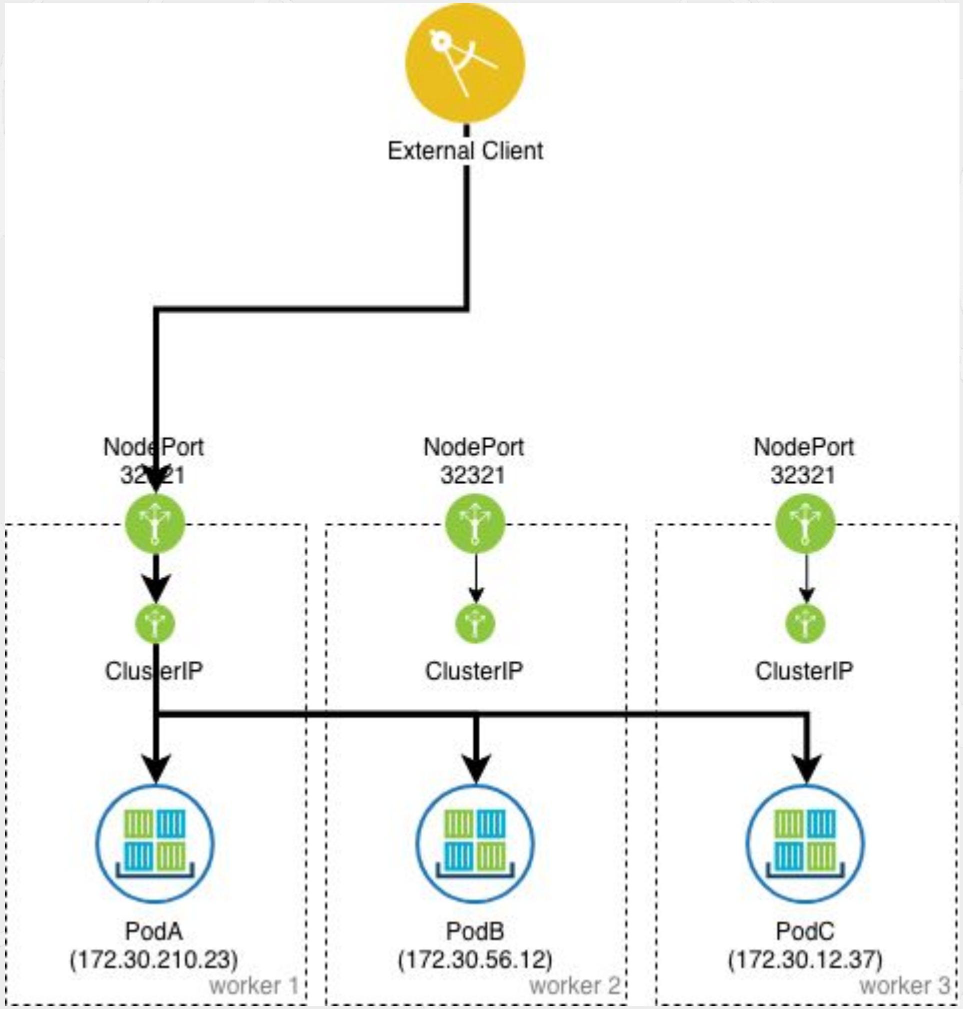


The Real-Time WAN Transport and RTI Routing Service provide a scalable solution to connect the edge to the cloud or on-prem data centers

- **Support for multiple gateway component acting as entry points in data centers for edge applications**
- **Ability to send and receive information from the data centers**
- **High-speed and performant connection within data centers using UDPv4, multicast, shared memory and other transports**

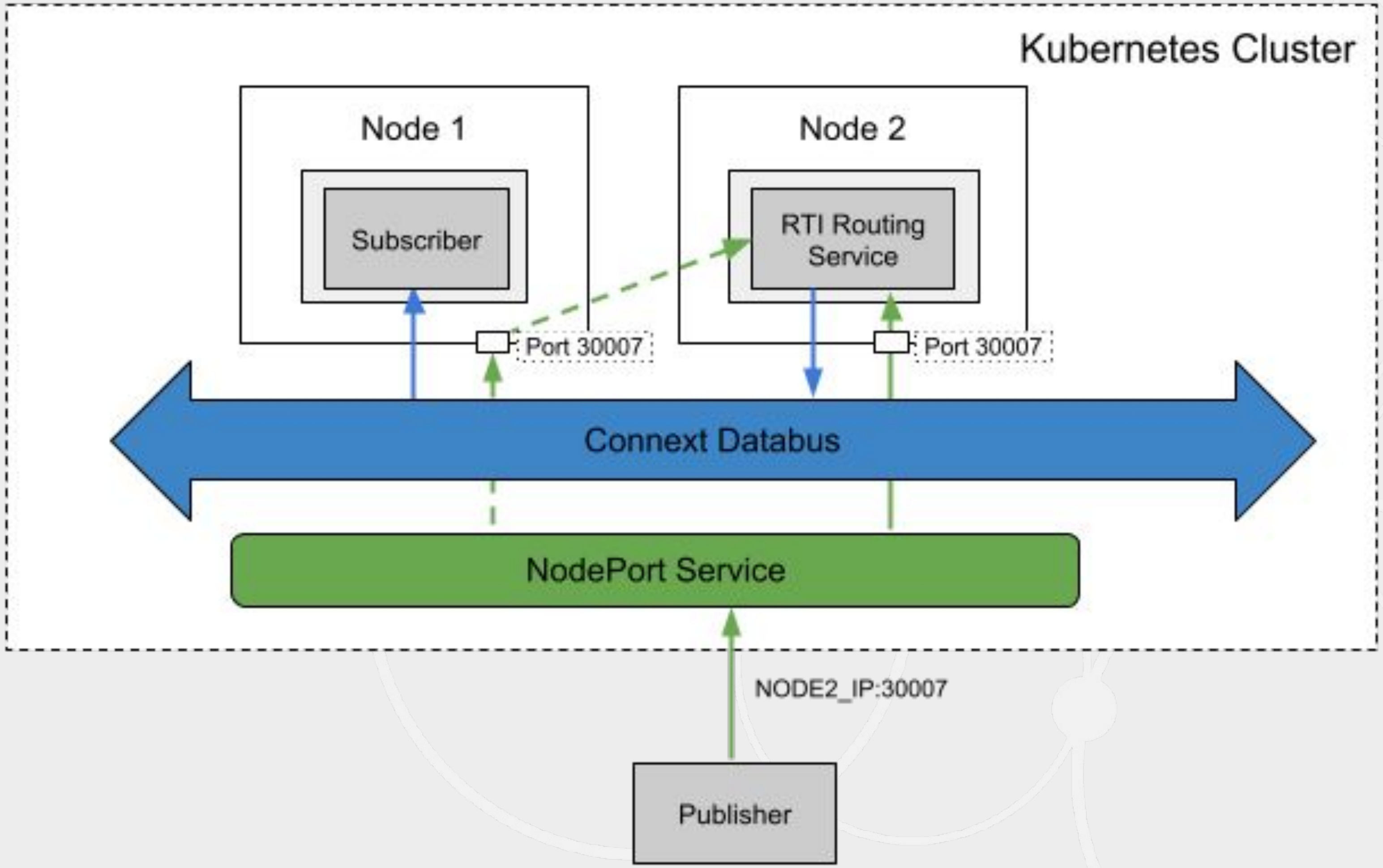
NodePort Service

- NodePort service is **the simplest way to expose a service** outside of the cluster.
- NodePort exposes **the ClusterIP service outside of the cluster** on node ports (default 30000-32767).
- Each node listens to the exposed port and **forwarding traffic on the NodePort to the ClusterIP**.



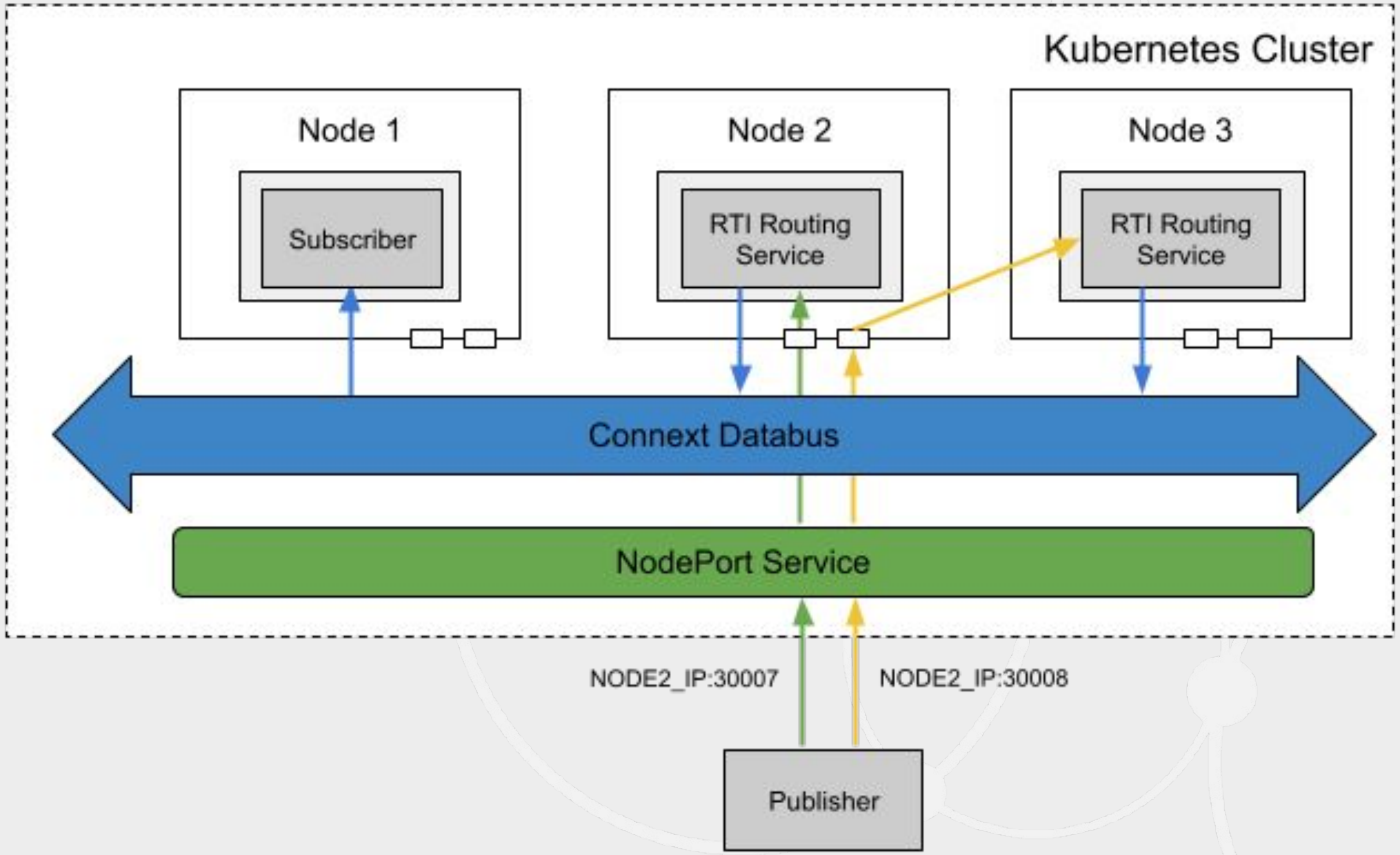
https://www.ibm.com/docs/en/cloud-private/3.2.0?topic=networking-kubernetes-service-types

Use Case 3 - Using RS for External Communications



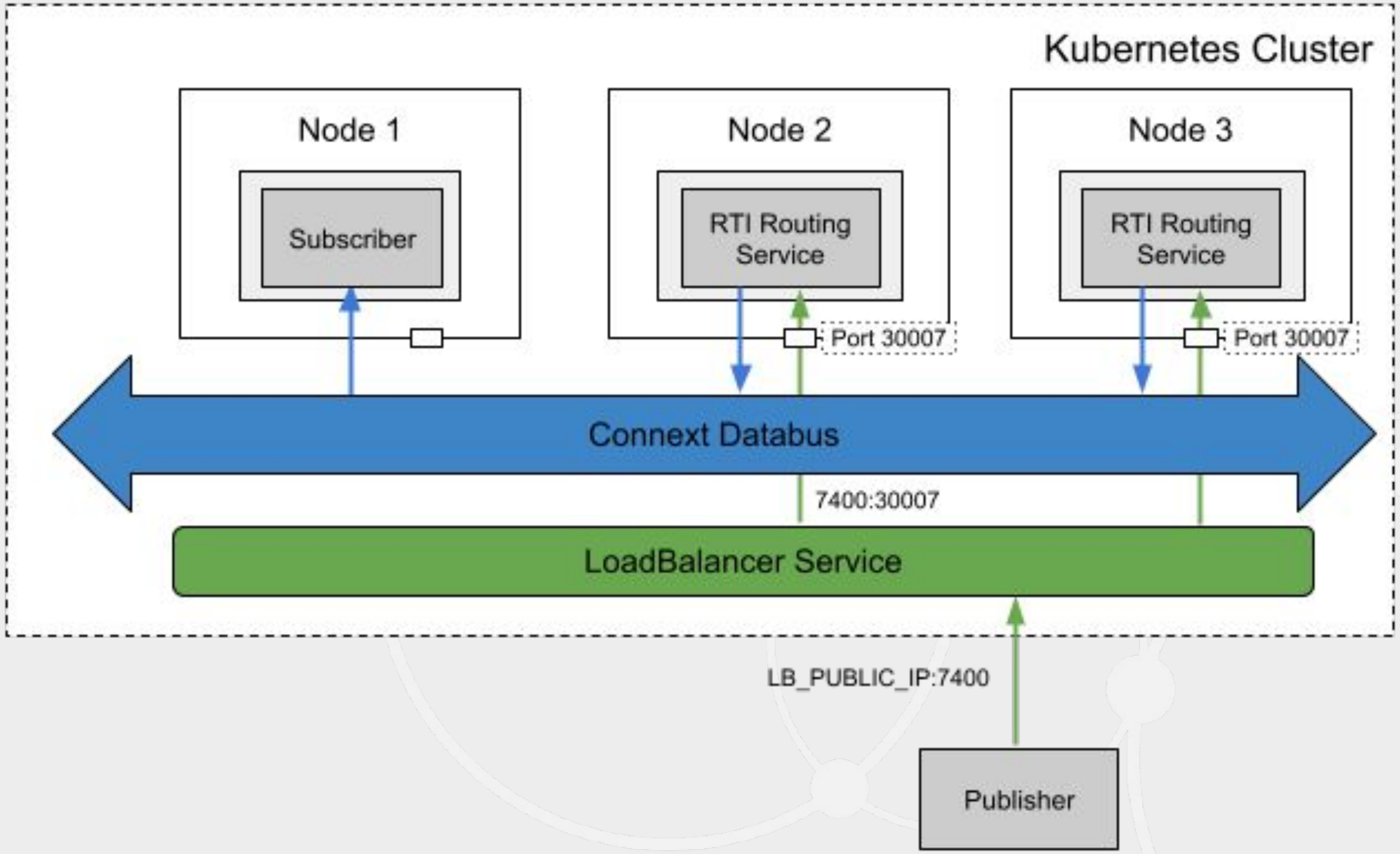
https://github.com/rticommunity/kubernetes-examples/tree/master/routingservice_rwt

Use Case 3.1 - Replicating Routing Services



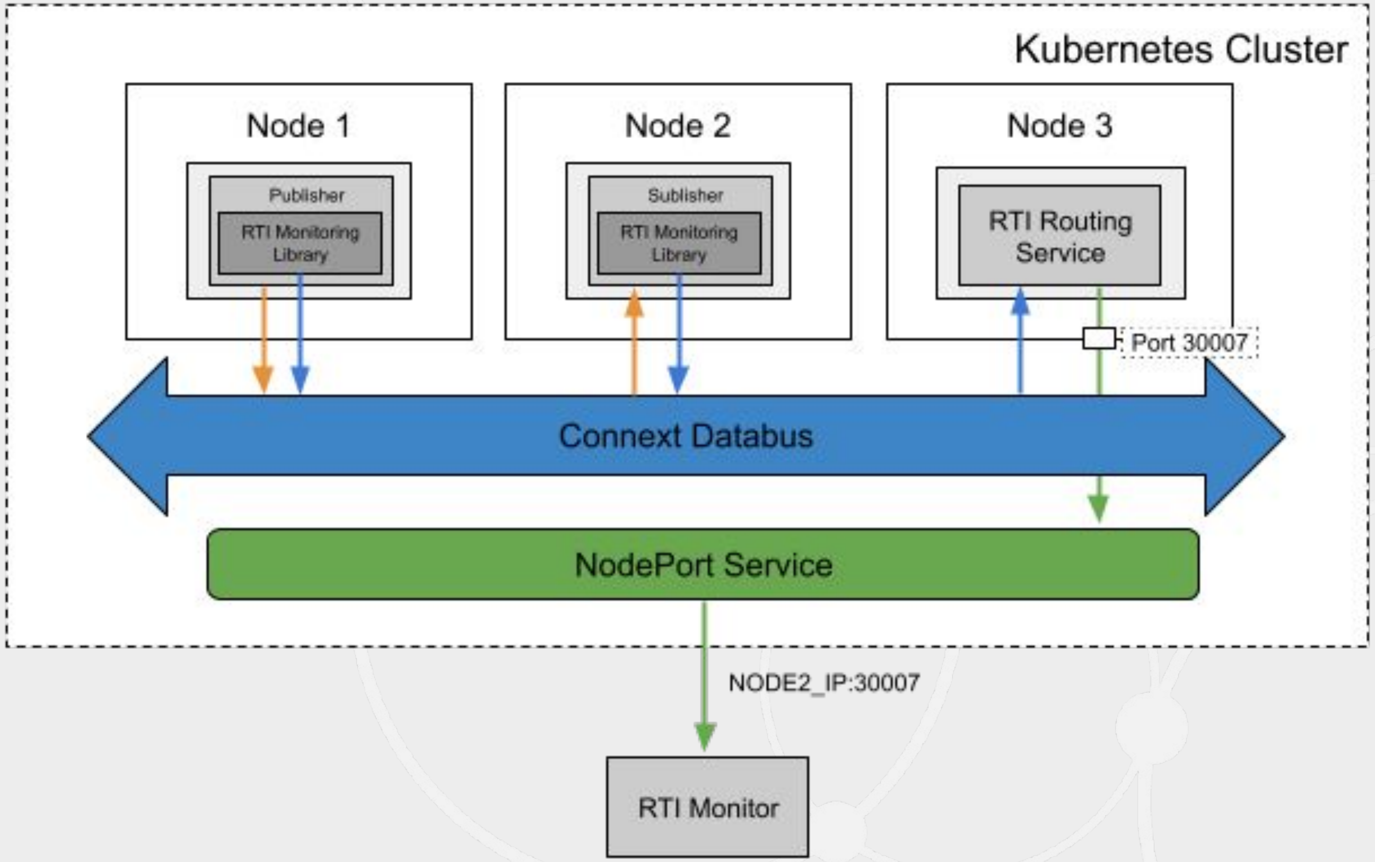
https://github.com/rticommunity/kubernetes-examples/tree/master/routingservice_rwt_replicated

Use Case 3.2 - Load Balancing Routing Services



https://github.com/rticommunity/kubernetes-examples/tree/master/routingservice_rwt_lb

Demo - Monitoring Connext Applications



https://github.com/rticommunity/kubernetes-examples/tree/master/routingservice_rwt_monitoring

Demo - Monitoring DDS Applications in a Cluster

https://github.com/rticommunity/kubernetes-examples/tree/master/routingservice_rwt_monitoring

Resources

- RTI Community GitHub Repository
  - https://github.com/rticommunity/kubernetes-examples
- Paper
  - "A Comprehensive Performance Evaluation of Different Kubernetes CNI Plugins for Edge-based and Containerized Publish/Subscribe Applications", 2021 IEEE International Conference on Cloud Engineering (IC2E), October 4-8, 2021
- Blogs
  - https://www.rti.com/blog/kubernetes-explained-how-it-can-improve-software-delivery-in-large-scale-dds-systems
  - https://www.rti.com/blog/a-step-by-step-guide-how-to-deploy-dds-applications-on-kubernetes