



**Jose Lopez-Vega**  
**Principal Engineer**

 **jose@rti.com**

October 19th-20th



# Product Advancements for Scaling Autonomous Systems



Technical Deep Dive

# Agenda



WAN Connectivity



Performance and Scalability



Security - sneak preview on 6.1.1

# Agenda



## WAN Connectivity



## Performance and Scalability



## Security - sneak preview on 6.1.1



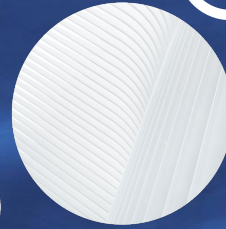
# Globally Distributed Systems Face Unique Challenges



**Increased Security Risks**



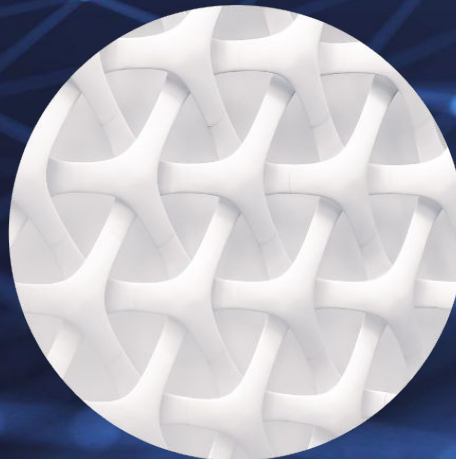
**Limited Edge Resources**



**Dynamically Discovering Systems**



**Connectivity Across WAN and Diverse Networks**



**Unreliable Networks**





# RTI Connex Anywhere

## WAN Connectivity Framework

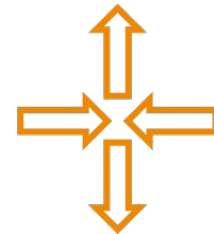
Seamlessly and reliably share data across WANs, including cellular networks, without compromising security. RTI's single-vendor WAN connectivity solution is friendly to container based deployment and network load balancers.



UDP-based



IP Mobility



NAT Traversal



Security



# Leverages All Capabilities of the Connex 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**



# RTI Connexx 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 Connexx Secure**  
Secures data end-to-end scalably over the WAN networks





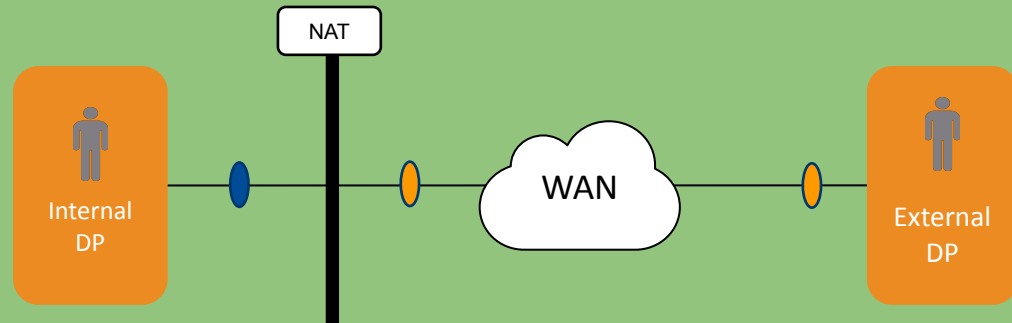
# Performant & Flexible WAN Communications



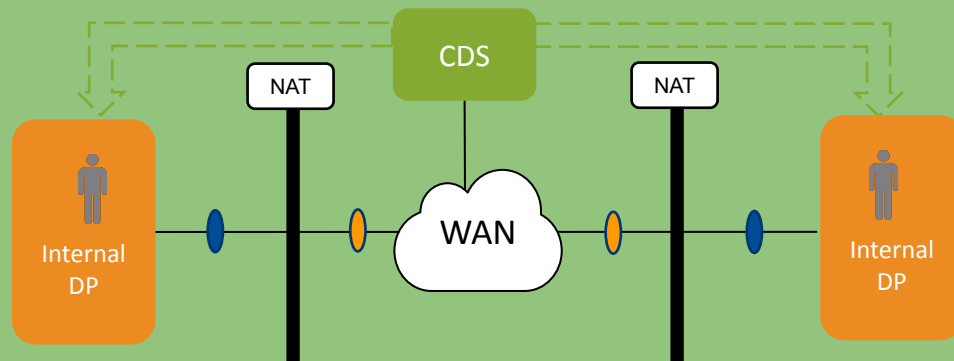
- Private to Public Communication
- Private to Private Communication
- IP Mobility
- Best Effort Delivery
- Reliable Delivery
- Transport Level Security
- Flow Level Security
- L4 Load Balancer Support
- L7 Load Balancer Support

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

# Network Setups for P2P Communication

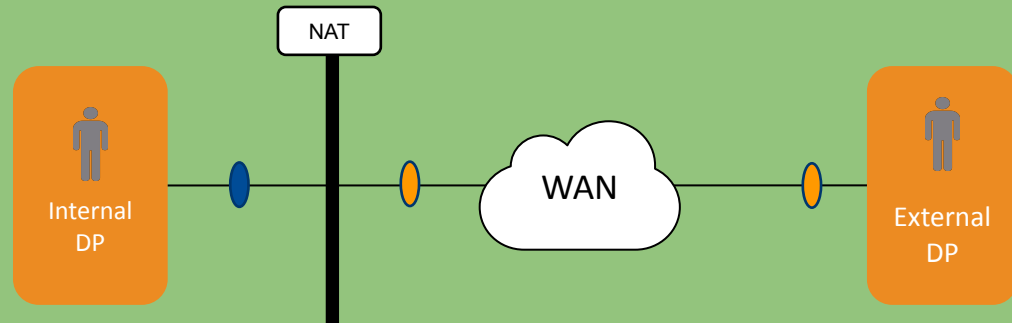


Public Address Participant -  
Participant behind **ANY NAT**

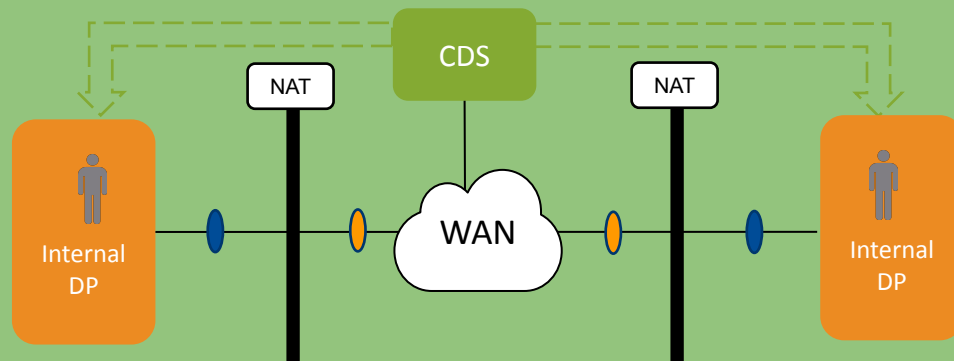


Both Participants Behind  
**Cone-NATs**

# Network Setups for P2P Communication

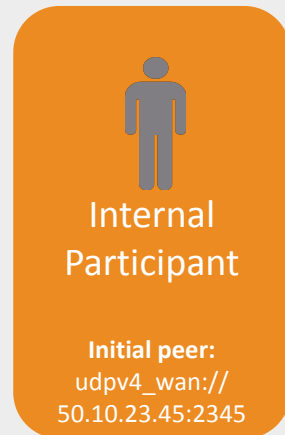


Public Address Participant - Participant behind **ANY NAT**



Both Participants Behind **Cone-NATs**

# Peer-to-Peer with External Participant: conf 1/2



```
<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: conf 2/2



```
<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

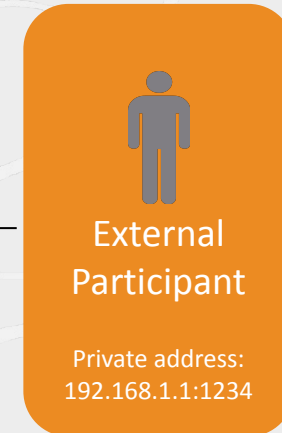
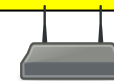
**192.168.1.1:1234** is the private IP address and port for the External Participant



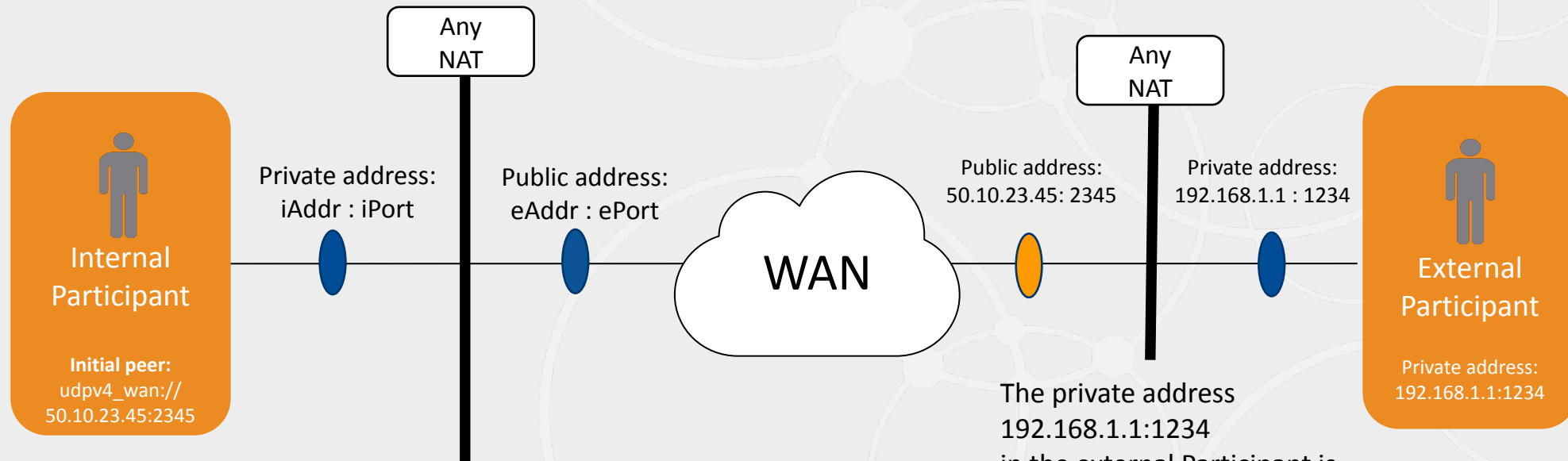
No need to set initial peers



Do not forget to configure your router!



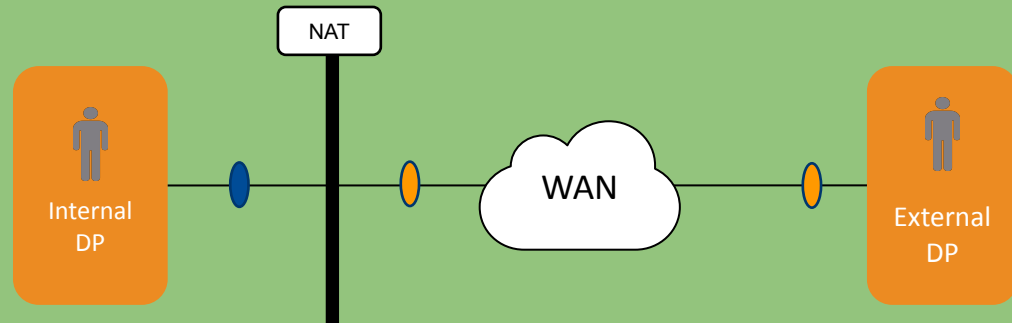
# Peer-to-Peer with External Participant



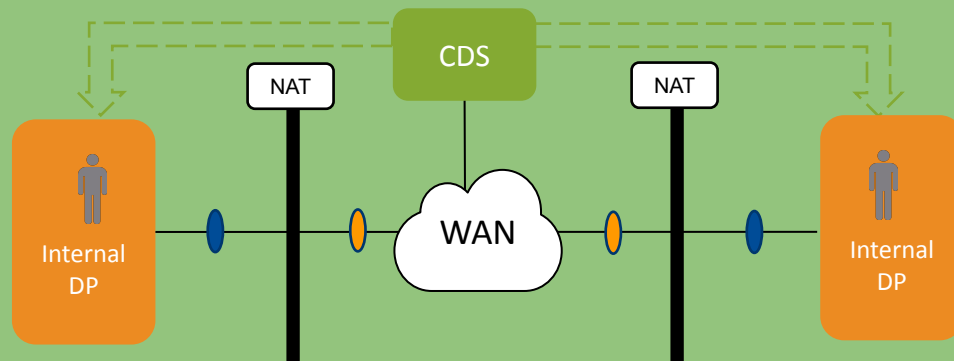
More than one Internal Participant communicating with the External Participant

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`

# Network Setups for P2P Communication




Public Address Participant - Participant behind **ANY NAT**



Both Participants Behind **Cone-NATs**

# Peer-to-Peer With Participants Behind Cone NAT: conf 1/2



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: conf 2/2



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>
```



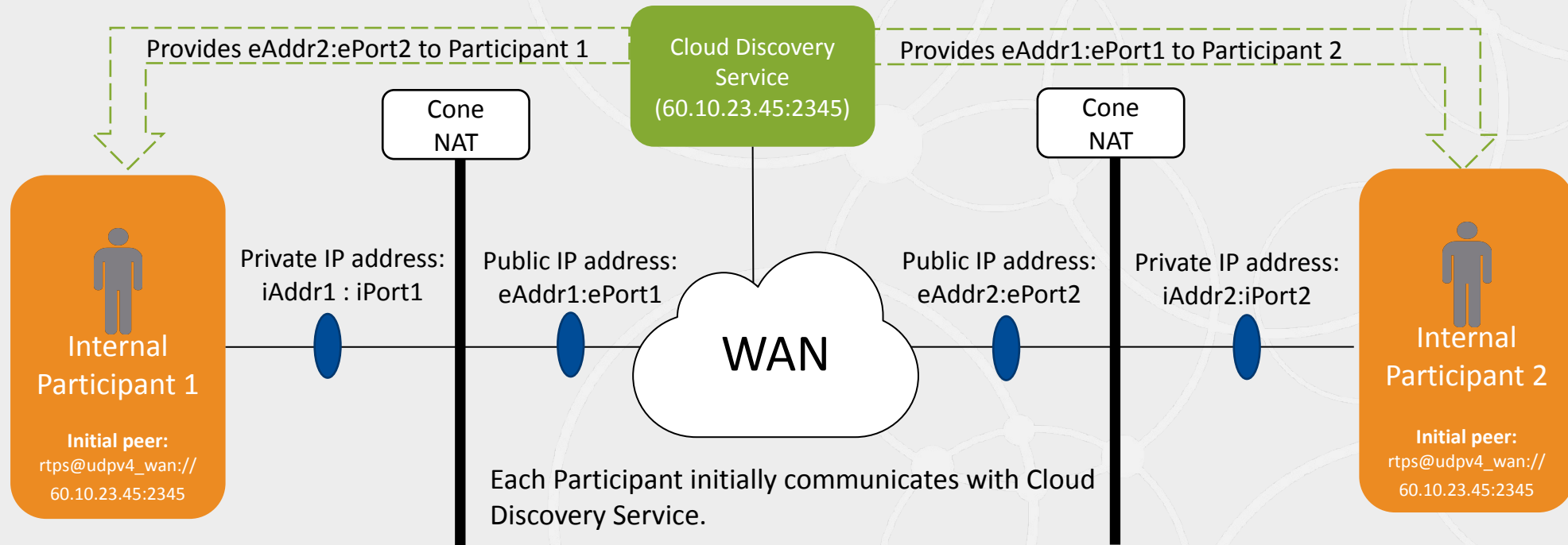
60.10.23.45:2345 is the public IP address and port of CDS



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



# Peer-to-Peer With Participants Behind Cone NAT



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.



# Avoid IP Fragmentation over Cellular

```
<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>
```

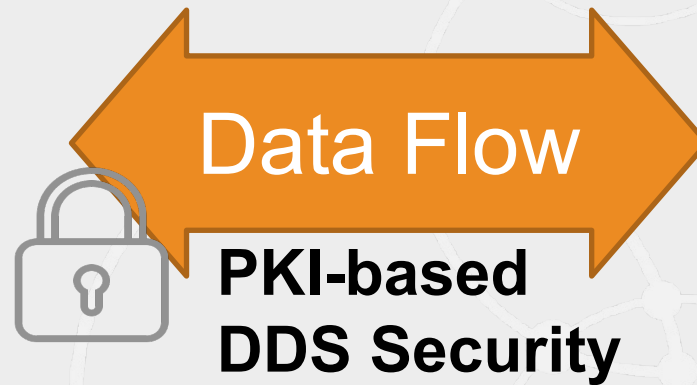
IP fragmentation disabled by setting transport MTU to 1400 bytes

256 bytes for metadata. User payload up to 1144 bytes



# WAN Security: the challenge

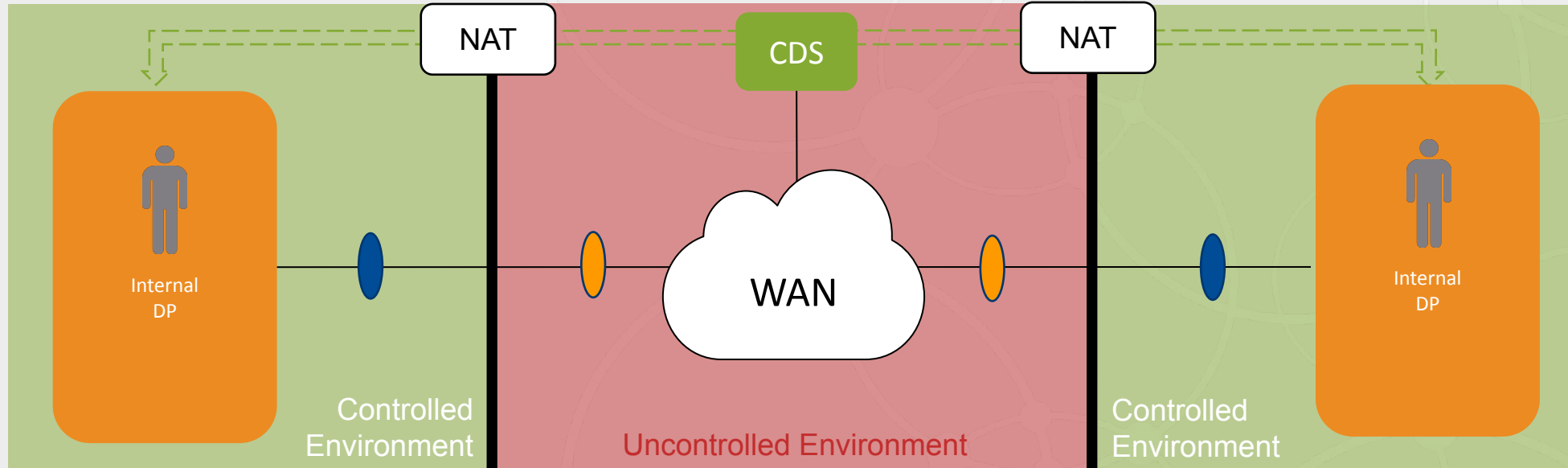
- DDS Security provides **granular security for the databus**



- DDS Security is enforced starting with a PKI-based authentication that triggers **after bootstrapping/participant discovery**



# WAN Security: the challenge



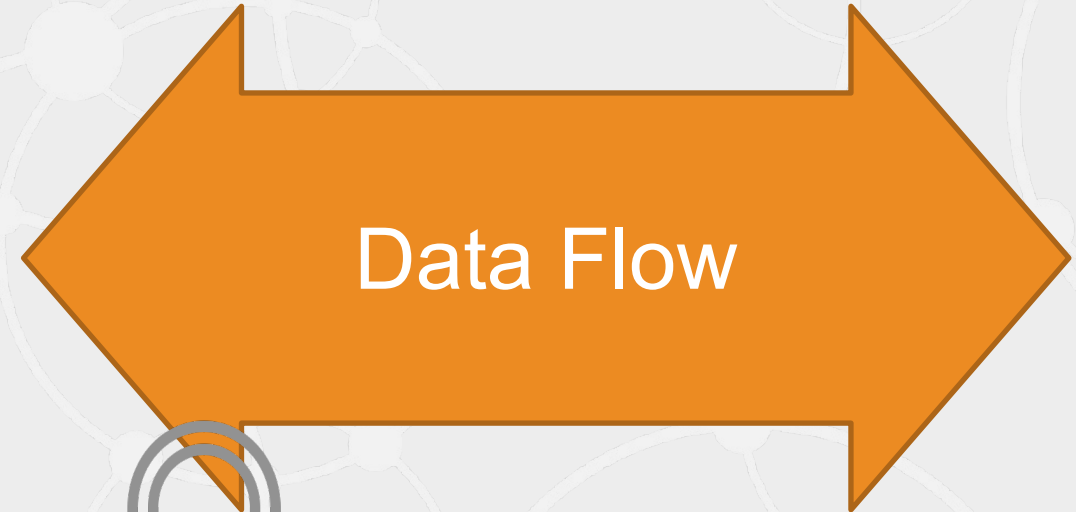
- We have no control at all of what happens in the WAN
- Mitigating **DoS attackers** or hiding the bootstrapping info becomes **much more harder**

# WAN Security: Connex approach



- Two level protection:

**new**  
RWT Management +  
Discovery  
Bootstrapping



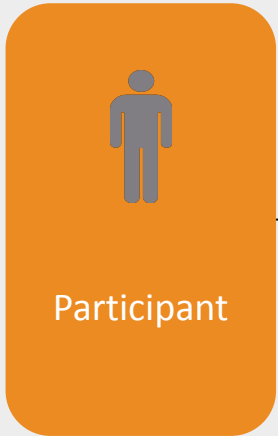
**PSK – based protection**



**PKI-based DDS Security**

6.1  
behavior

# WAN Security: Connex approach



Participant

```
<dds>
  <qos_profile name="Participant">
    <participant_qos>
      <transport_builtin>
        <mask>UDPv4_WAN</mask>
      </transport_builtin>
      ...
      <property>
        <element>
          <name>com.rti.serv.secure.cryptography.rtps_protection_key</name>
          <value>str:key0</value>
        </element>
        <element>
          <name>com.rti.serv.secure.authentication.participant_discovery_protection_key</name>
          <value>str:key1</value>
        </element>
      </property>
    </participant_qos>
  </qos_profile>
</dds>
```



rtps\_protection\_key is the PSK protecting RWT messages

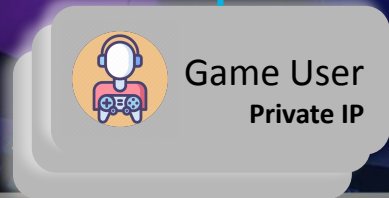
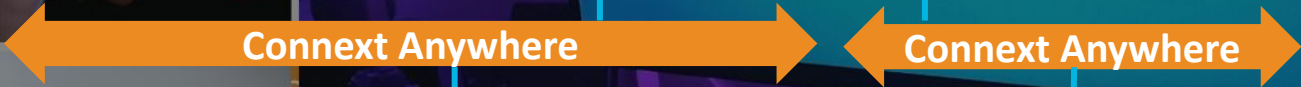
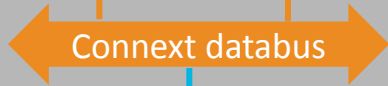
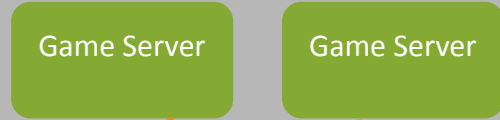


participant\_discovery\_protection\_key is the PSK protecting participant discovery

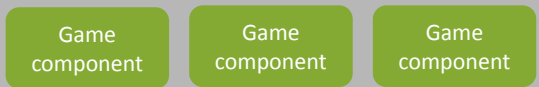
6.1  
behavior



# Deployment Scenarios



Game User  
Private IP



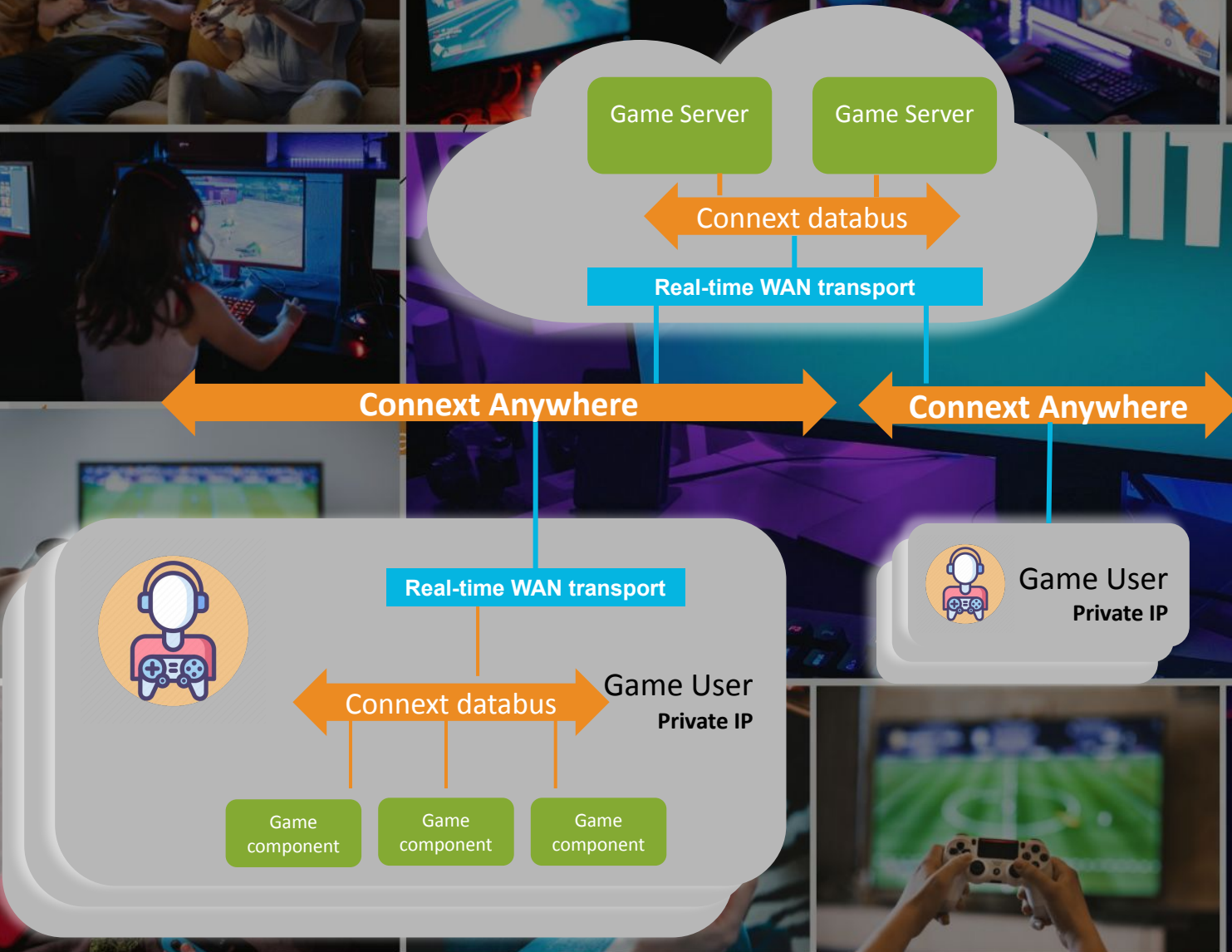


# Edge to Cloud (private to public)



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



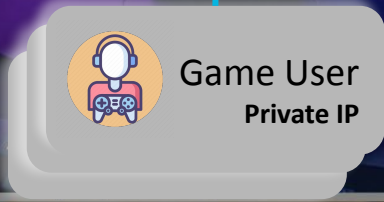
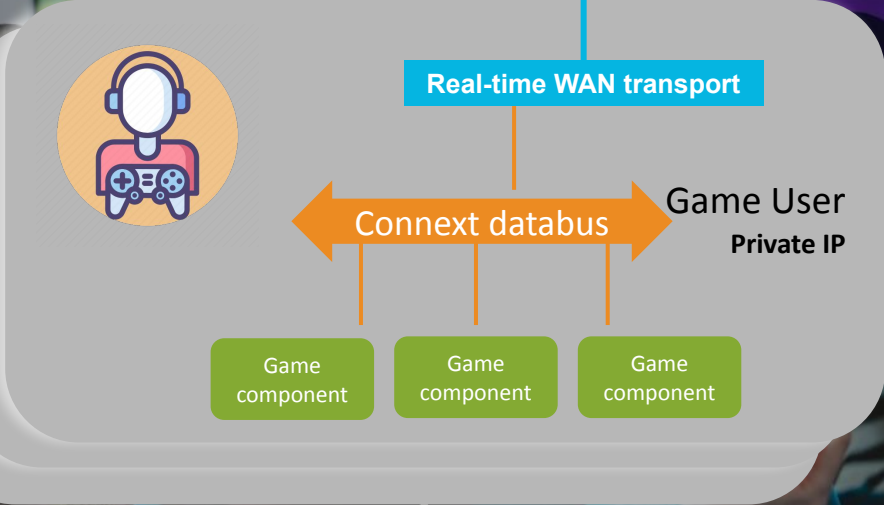
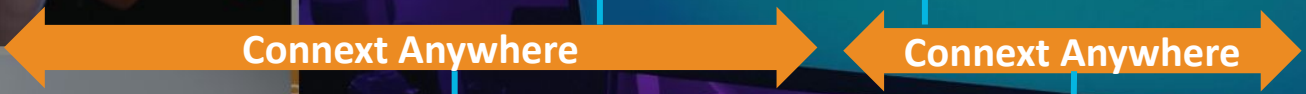
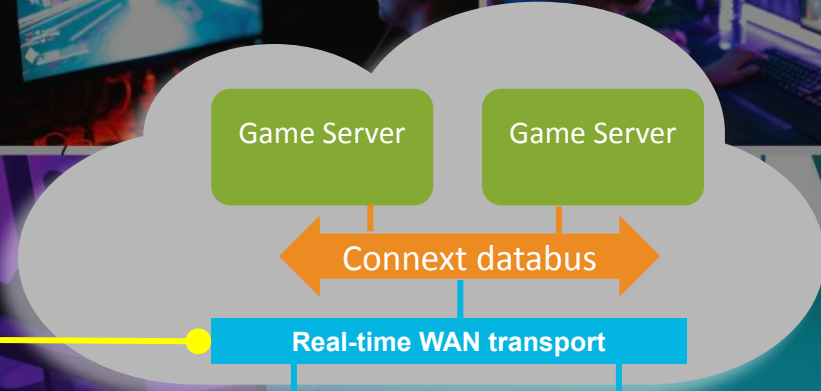
# Edge to Cloud (private to public)



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

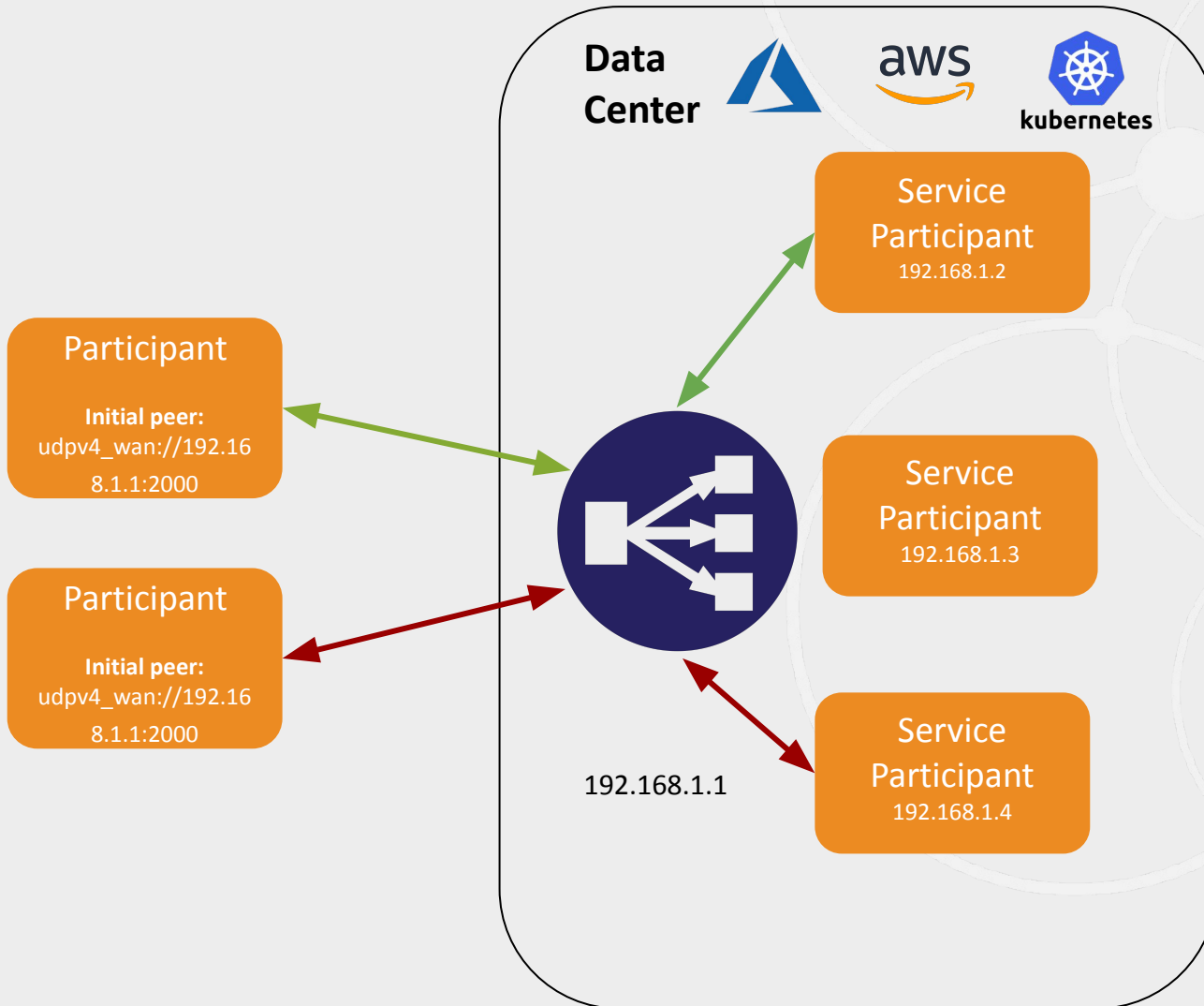
- Support for multiple gateway component acting as entry points in data centers for edge applications
- Ability to send and receive information from the data centers
- High-speed and performant connection within data centers using UDPv4, multicast, shared memory and other transports

Needs Load Balancing





# Participant Load Balancing Using a Layer 4 Load Balancer



**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>
```

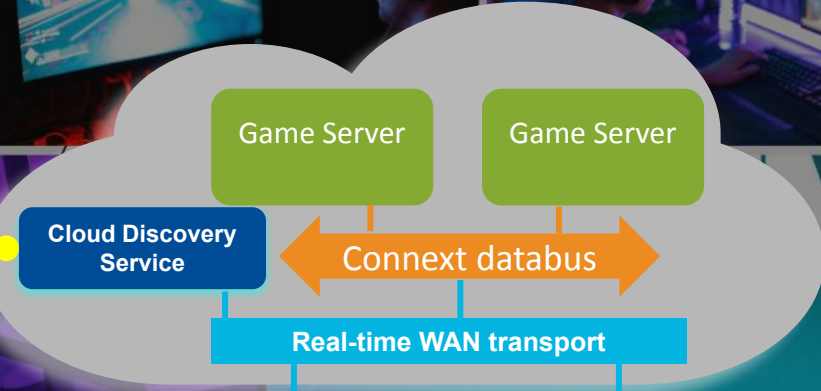
Only for External Participants



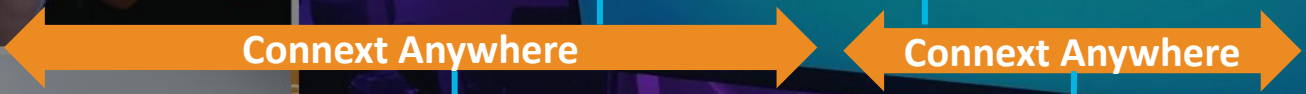
# Peer-to-Peer Edge-to-Edge (Private to Private)



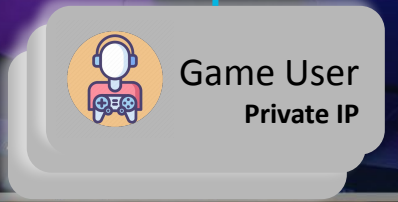
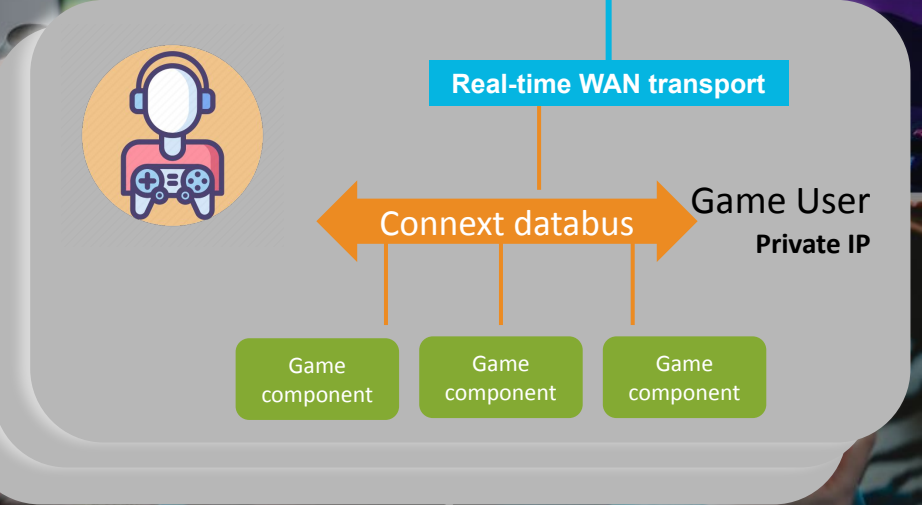
Discovers Connex applications over WAN



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



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

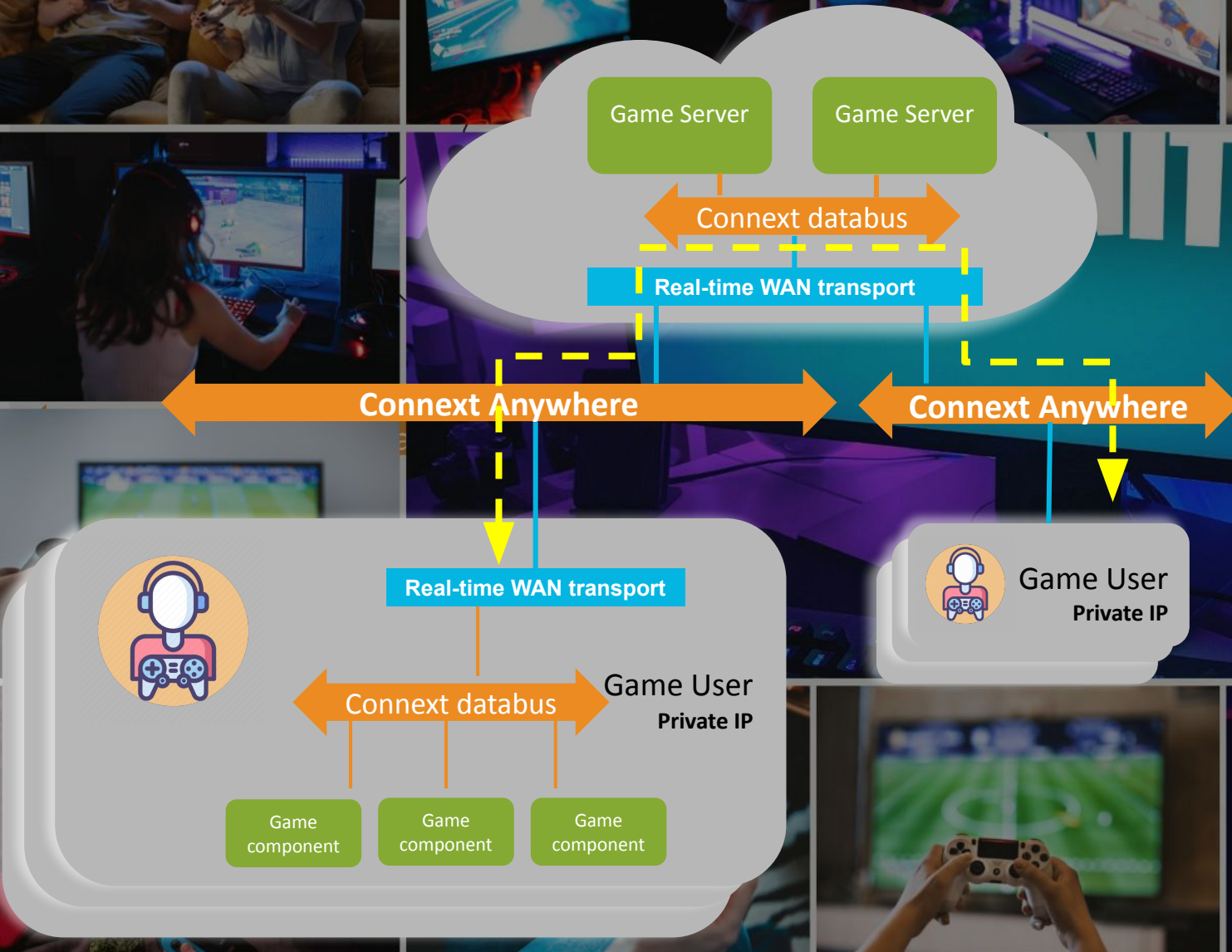


\* Peer-to-peer communications is possible only in Cone NAT environments





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

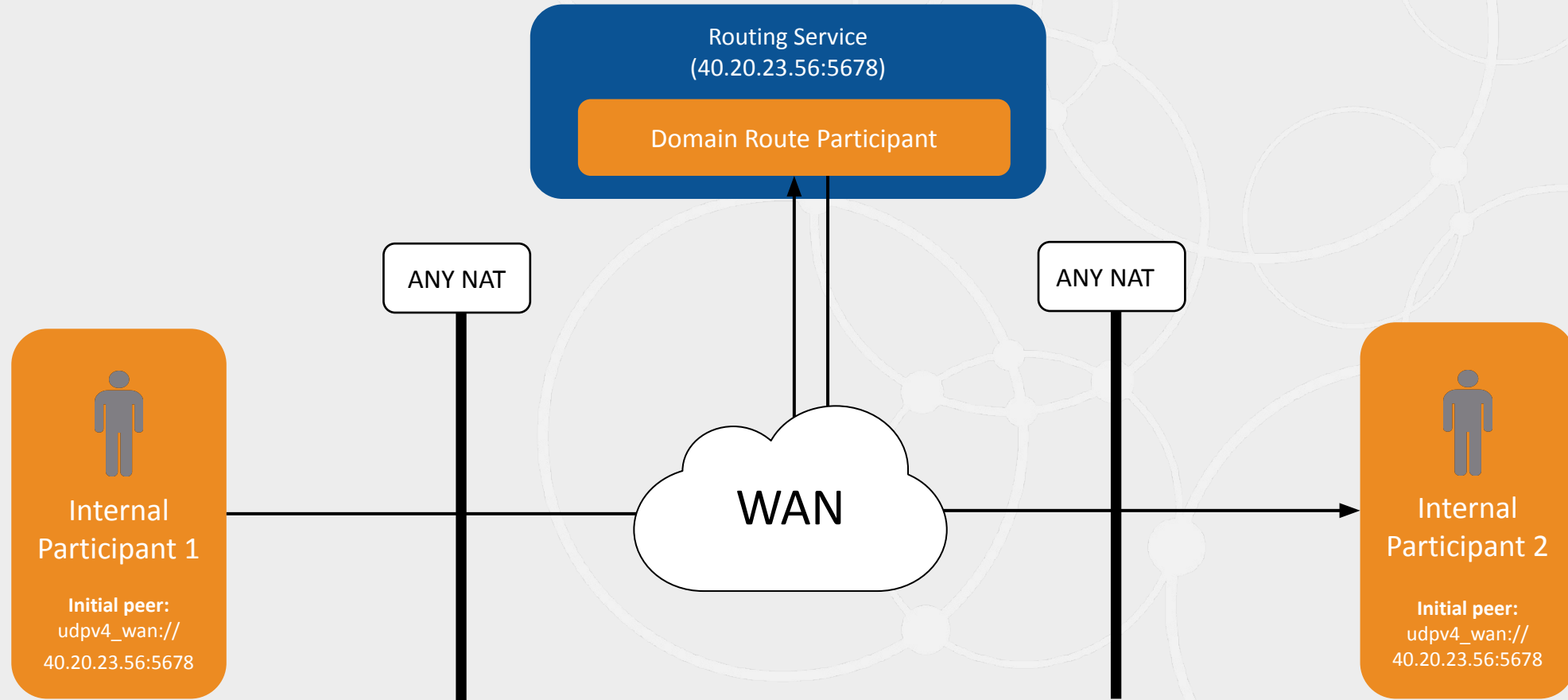


Real-time WAN Transport also supports applications communicating over the WAN under any circumstances, such as NATed environments

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



# Relay Using Routing Service



# Relay Using Routing Service



Routing Service  
(40.20.23.56:5678)

Domain Route Participant

```
<dds>
  <routing_service name="RS">
    <domain_route name="TwoWayDomainRoute">
      <participant name="RSParticipant">
        <participant_qos>
          <transport_builtin>
            <mask>UDpv4_WAN</mask>
            <udp4_wan>
              <public_address>
                40.20.23.56
              </public_address>
              <comm_ports>
                <default>
                  <host>5678</host>
                  <public>5678</public>
                </default>
              </comm_ports>
            </udp4_wan>
          </transport_builtin>
        </participant_qos>
      </participant>
      <!-- Sessions and Topic Routes here -->
    </domain_route>
  </routing_service>
</dds>
```





RTWAN - Saved to my Mac

Transitions Animations Slide Show Review View Tell me

Color: 18 A A A

Convert to SmartArt Picture Shapes Text Box Arrange Quick Styles Show Outline Design Ideas

Wi-Fi: en0

No.	Time	Source	Destination	Protocol	Length	Differentiated Services Field	Source Port	Destination Port	Info
4667	2021-05-21 13:04:59.32950	172.58.140.144	192.168.1.73	RTPS	206	0x00	49540	16000	INFO_TS, DATA_FRAG, HEARTBEAT
4668	2021-05-21 13:04:59.329668	172.58.140.144	192.168.1.73	RTPS	106	0x00	16000	62503	INFO_DST, ACKNACK
4669	2021-05-21 13:04:59.329792	192.168.1.73	172.58.140.144	RTPS	106	0x00	16000	62503	INFO_DST, ACKNACK
4670	2021-05-21 13:04:59.329812	192.168.1.73	172.58.140.144	RTPS	106	0x00	16000	62503	INFO_DST, ACKNACK
4671	2021-05-21 13:04:59.329872	192.168.1.73	172.58.140.144	RTPS	106	0x00	16000	62503	INFO_DST, ACKNACK
4672	2021-05-21 13:04:59.332691	172.58.140.144	192.168.1.73	RTPS	1382	0x00	49540	16000	INFO_TS, DATA_FRAG
4673	2021-05-21 13:04:59.332697	172.58.140.144	192.168.1.73	RTPS	206	0x00	49540	16000	INFO_TS, DATA_FRAG, HEARTBEAT
4674	2021-05-21 13:04:59.332698	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4675	2021-05-21 13:04:59.332698	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4676	2021-05-21 13:04:59.332699	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4677	2021-05-21 13:04:59.332702	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4678	2021-05-21 13:04:59.332807	192.168.1.73	172.58.140.144	RTPS	106	0x00	16000	62503	INFO_DST, ACKNACK
4679	2021-05-21 13:04:59.335542	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4680	2021-05-21 13:04:59.335544	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4681	2021-05-21 13:04:59.335545	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4682	2021-05-21 13:04:59.335545	172.58.140.144	192.168.1.73	RTPS	106	0x00	49540	16000	INFO_DST, ACKNACK
4689	2021-05-21 13:04:59.363267	192.168.1.73	172.58.140.144	RTPS	1382	0x00	16000	62503	INFO_TS, DATA_FRAG
4690	2021-05-21 13:04:59.363295	192.168.1.73	172.58.140.144	RTPS	206	0x00	16000	62503	INFO_TS, DATA_FRAG, HEARTBEAT
4696	2021-05-21 13:04:59.396843	192.168.1.73	172.58.140.144	RTPS	1382	0x00	16000	62503	INFO_TS, DATA_FRAG
4697	2021-05-21 13:04:59.396857	192.168.1.73	172.58.140.144	RTPS	218	0x00	16000	62503	INFO_TS, DATA_FRAG, HEARTBEAT, INFO_TS

wireshark\_Wi-FIGNKP30.pcapng

Packets: 5157 · Displayed: 3375 (65.4%) · Dropped: 0 (0.0%)

Profile: Default

> Frame 4672: 1382 bytes on wire (11056 bits), 1382 bytes captured (11056 bits) on interface en0, id 0  
> Ethernet II, Src: e0:22:03:7c:fa:a9 (e0:22:03:7c:fa:a9), Dst: Apple\_02:d5:0d (f8:ff:c2:02:d5:0d)  
> Internet Protocol Version 4, Src: 172.58.140.144, Dst: 192.168.1.73  
> User Datagram Protocol, Src Port: 49540, Dst Port: 16000  
> Real-Time Publish-Subscribe Wire Protocol

```
0:04:08.442544000 56688 0x7fd0e0c240 ERROR libav : error while decoding MB 33 14, bytestream -6
0:04:11.044686000 56688 0x7fd0e0b29c0 ERROR libav : error while decoding MB 17 21, bytestream -5
0:04:11.069498000 56688 0x7fd0e080f0 ERROR libav : error while decoding MB 32 6, bytestream -6
```

Taskbar with various application icons including Parallels Desktop, RTI Launcher, and other system utilities.

# Agenda



WAN Connectivity



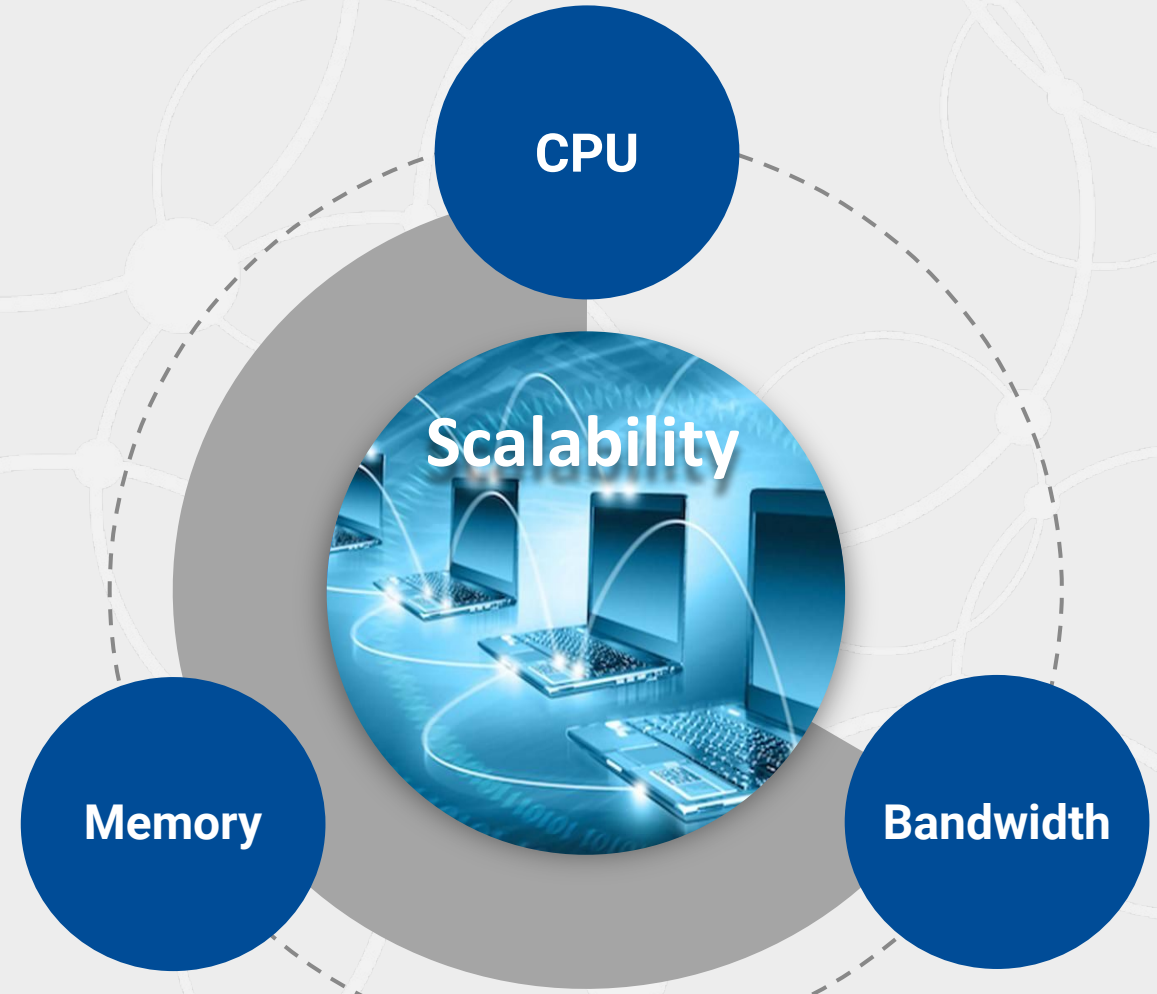
**Performance and Scalability**

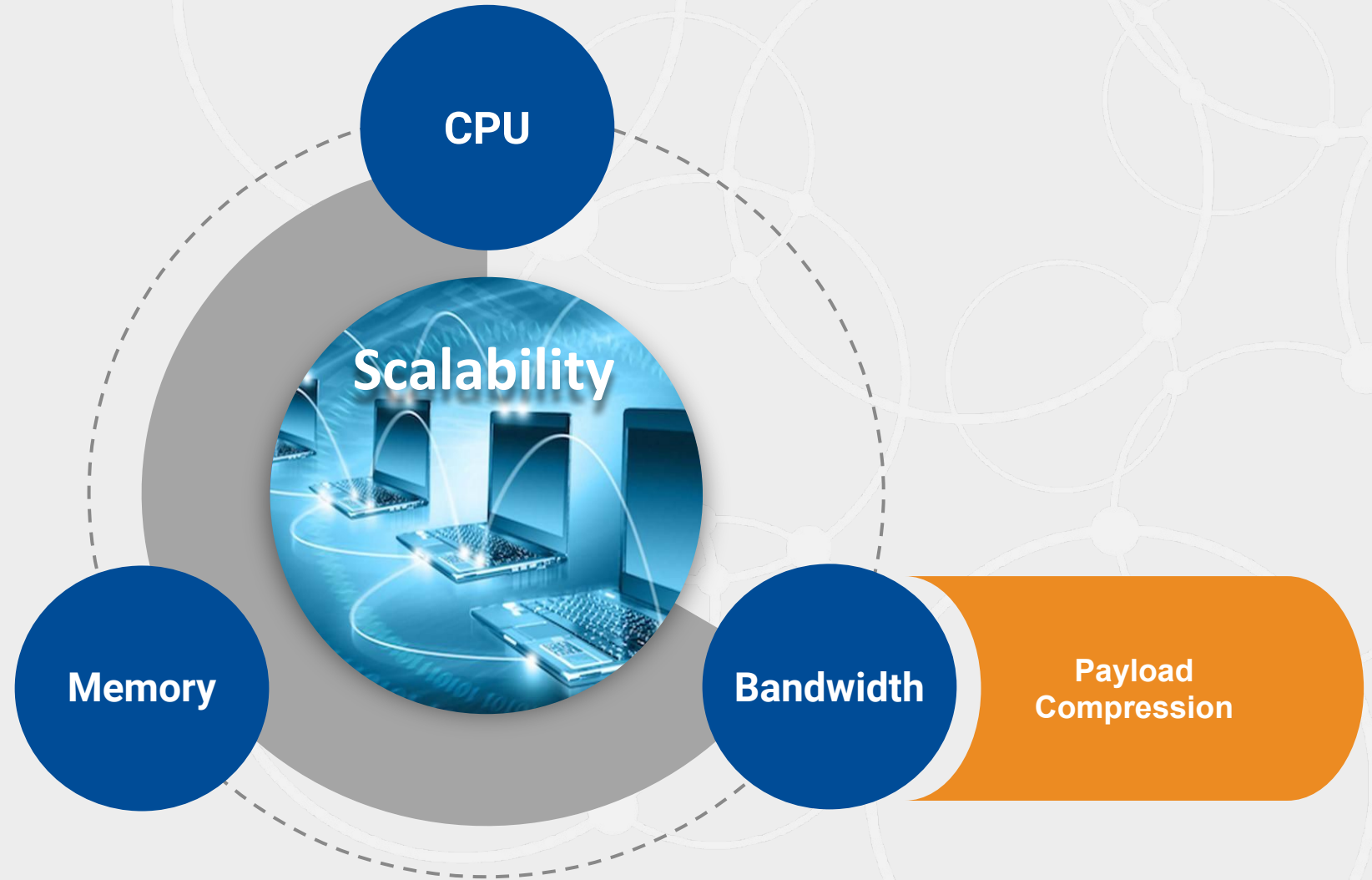


Security - sneak preview on 6.1.1



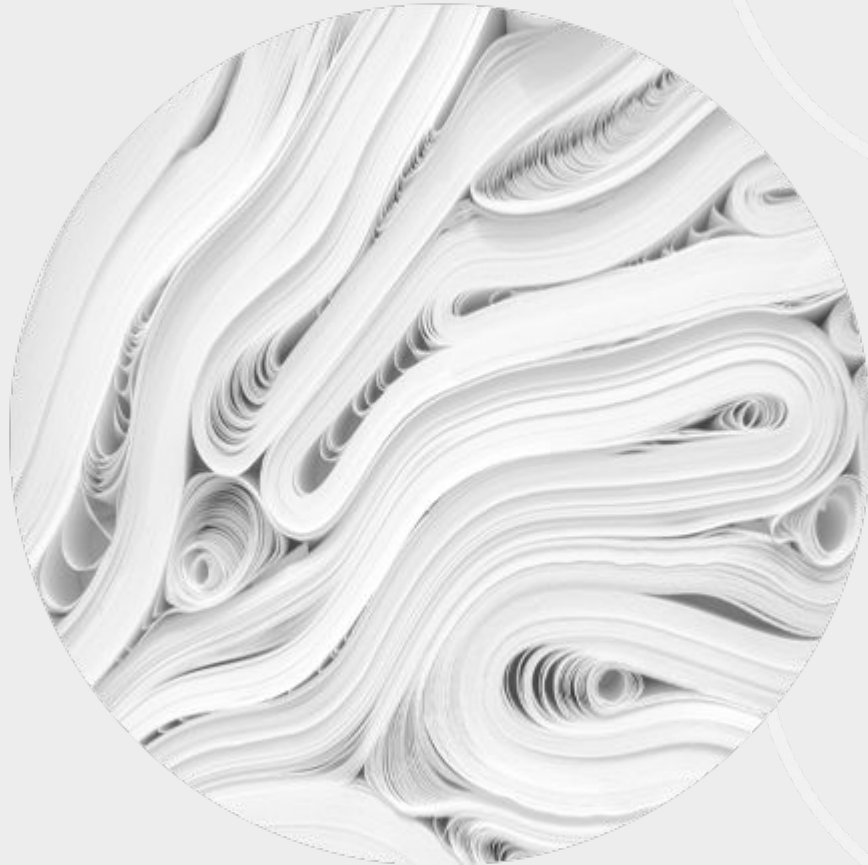
Connex allows *You* to build systems that continue to perform as your resource needs evolve and grow







# 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

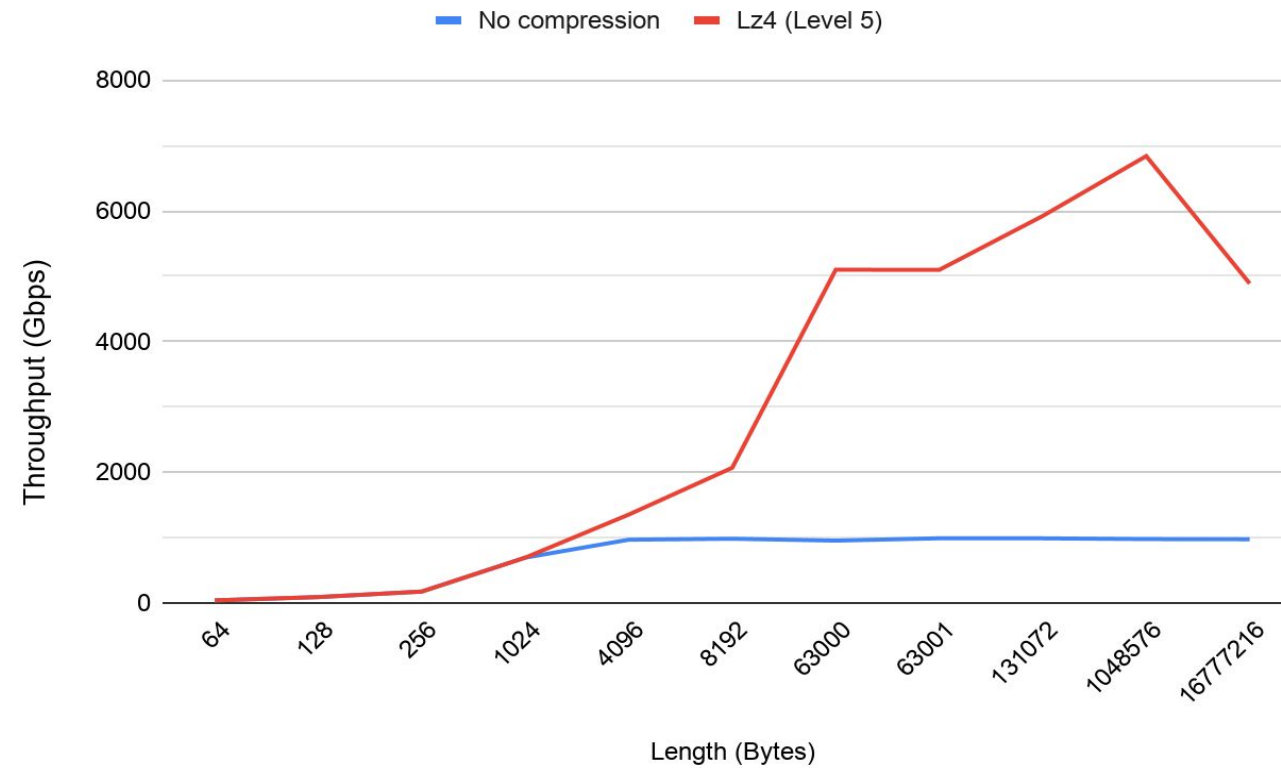




# Compression Increases Throughput



Compression Real Customer Data on 1Gb network.



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

# Connex Compression Options



When is the sample compressed

What is compressed

How is compressed

How is enabled

When should I use it?

ZRTPS

Upon send

The entire RTPS message

zlib, bzip2, custom  
compression libraries

Separate dynamically linked  
library

Small payloads over low  
bandwidth networks

new

Payload

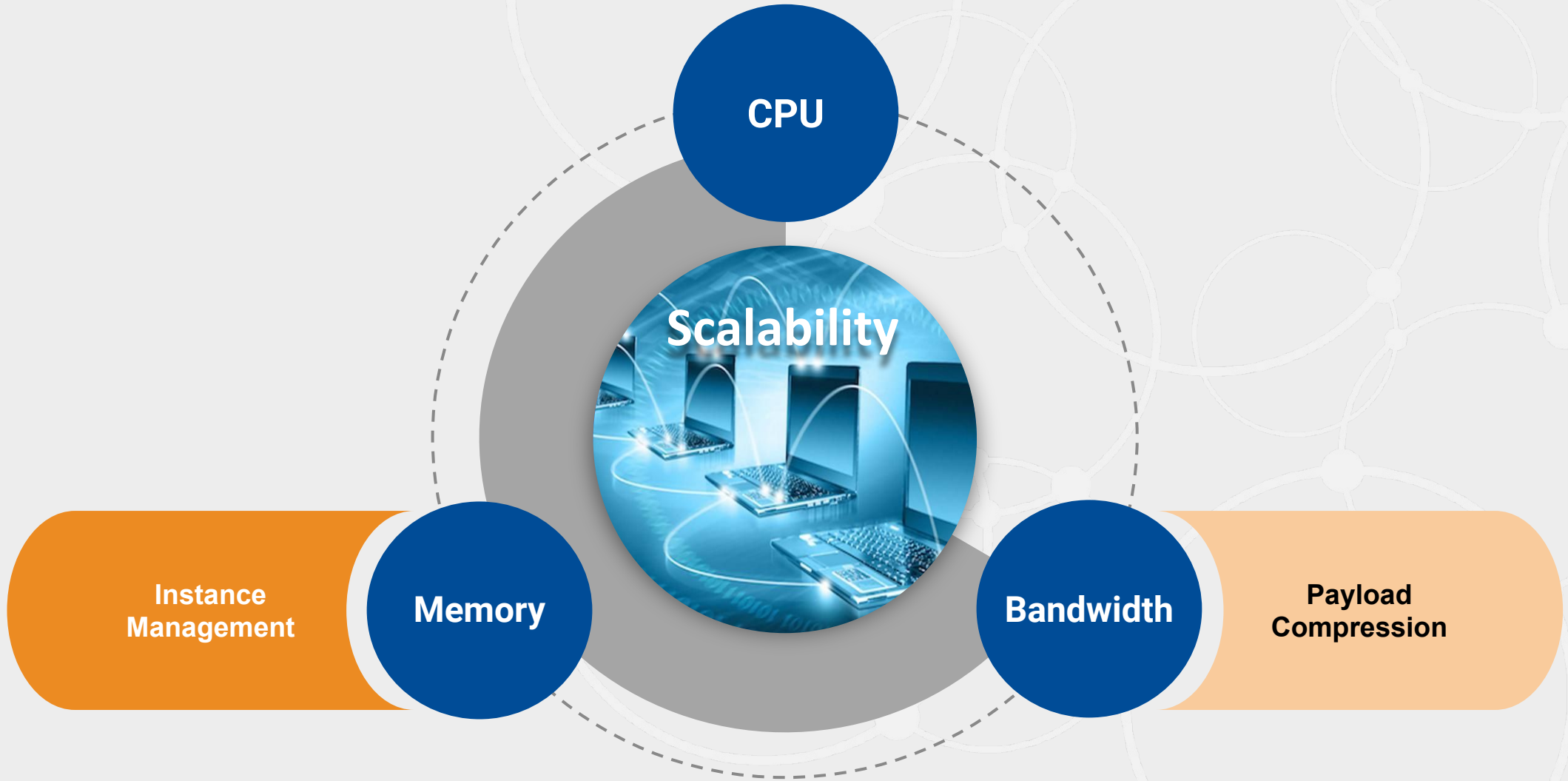
Upon serialization /  
refiltering

User payload

bzip2, zlib, lz4

Built-in

Large payloads



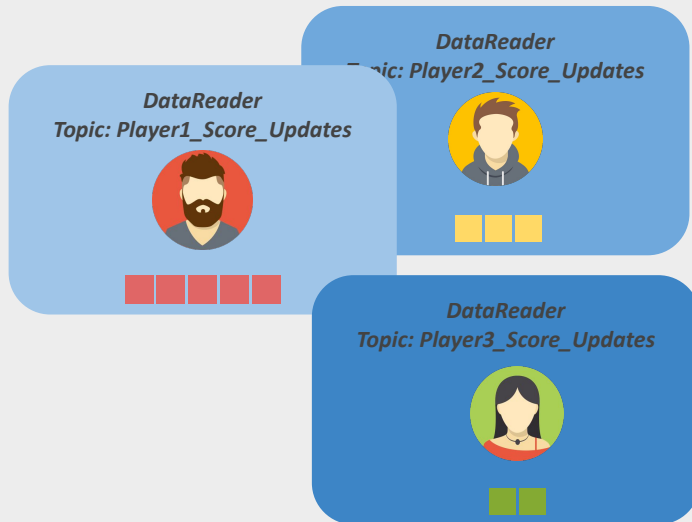


# Instances Offer Scalability in Representing Real World Objects

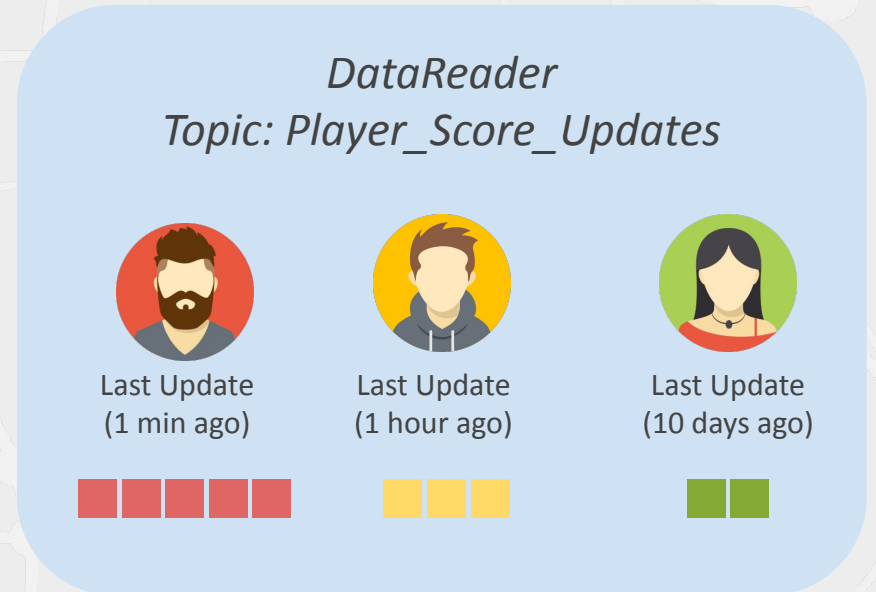
Alternatives to represent real-world objects

Instances

Separate Topics Per Object



Attribute in a Topic to track Object ID



More Memory and Discovery Time



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



Less Memory and Discovery Time



Lifecycle Management



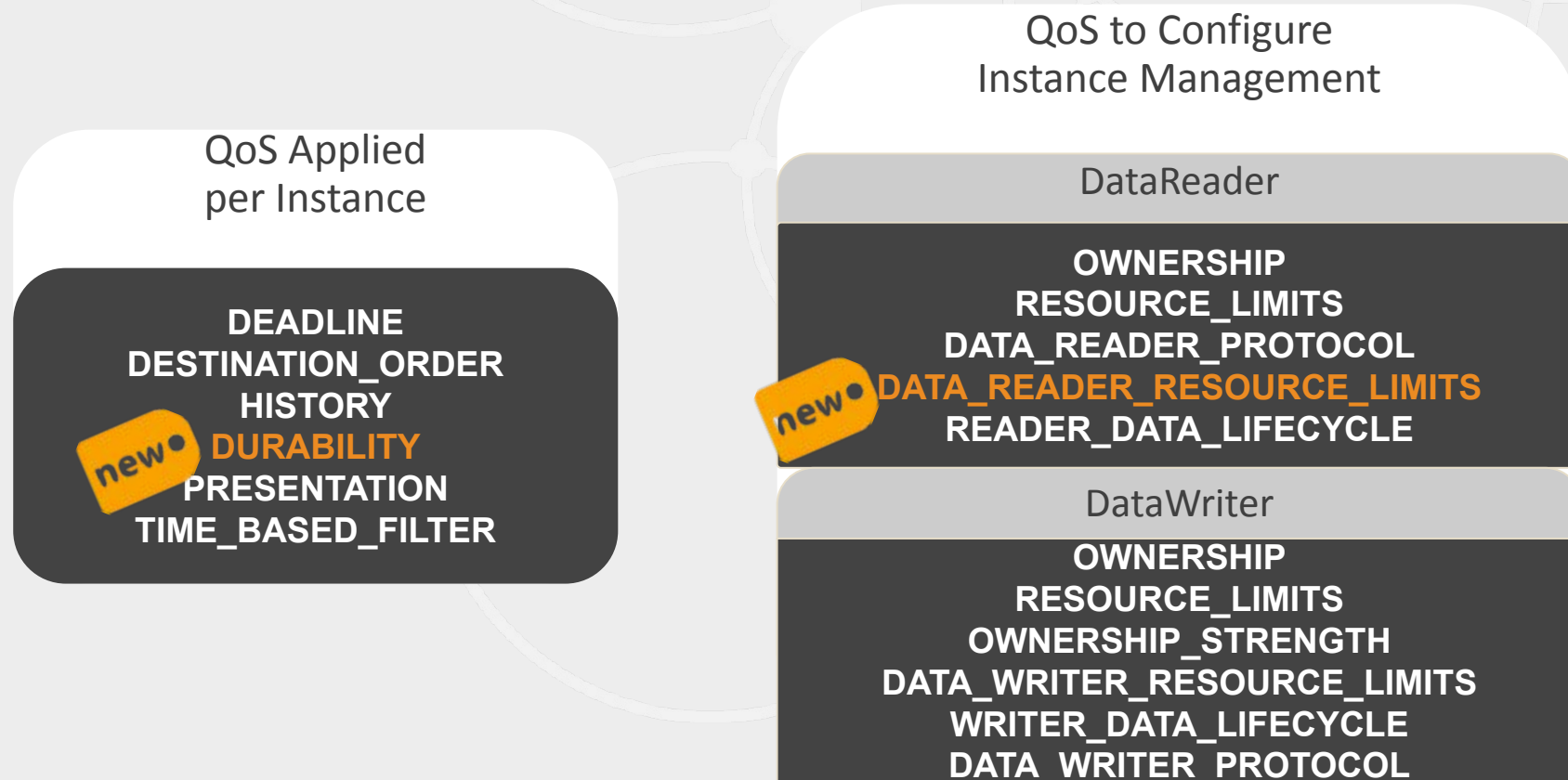
QoS Configurable Per Instance





# Instances Configurable With Individual QoS

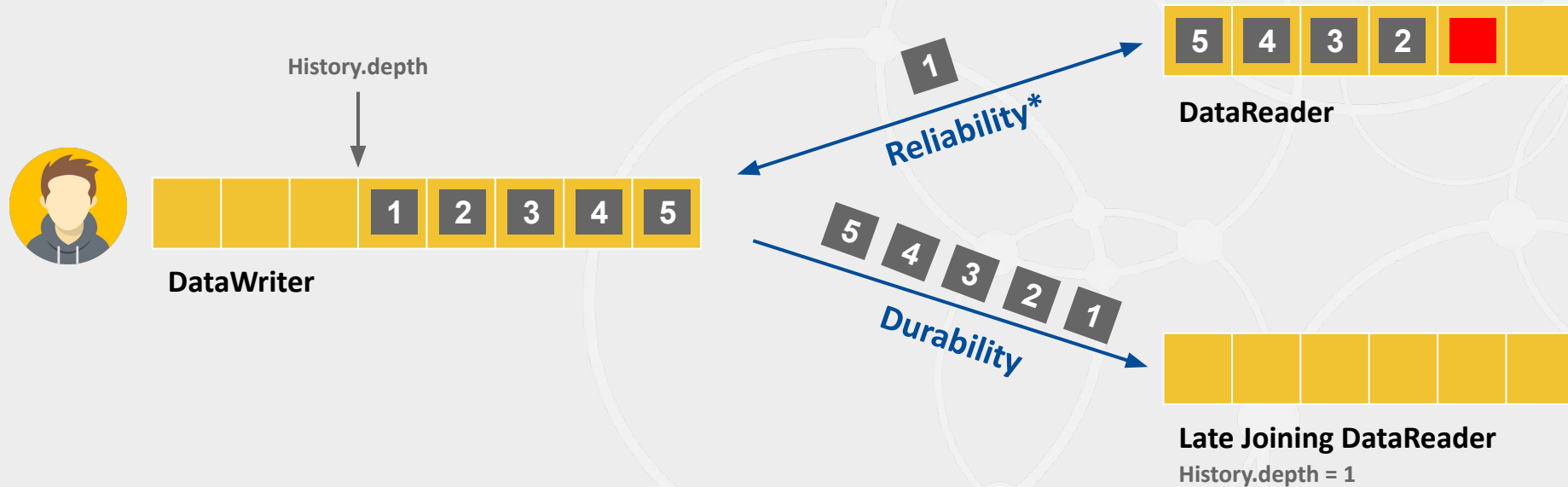
Some QoS policies are applied per instance, and other QoS policies configure instance management





# Durability QoS Parameter for Handling Instance History

Durability: ability to update newly joined applications with historical data



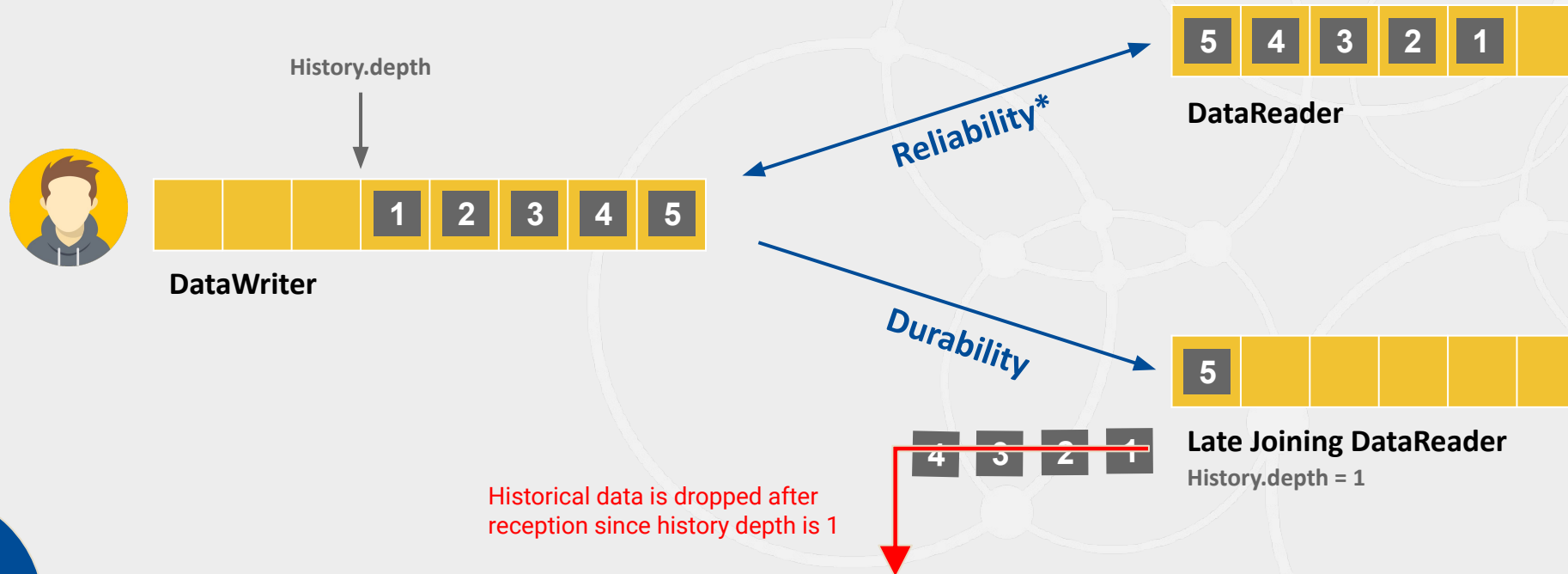
< 6.1  
behavior

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



# Durability QoS Parameter for Handling Instance History

Durability: ability to update newly joined applications with historical data



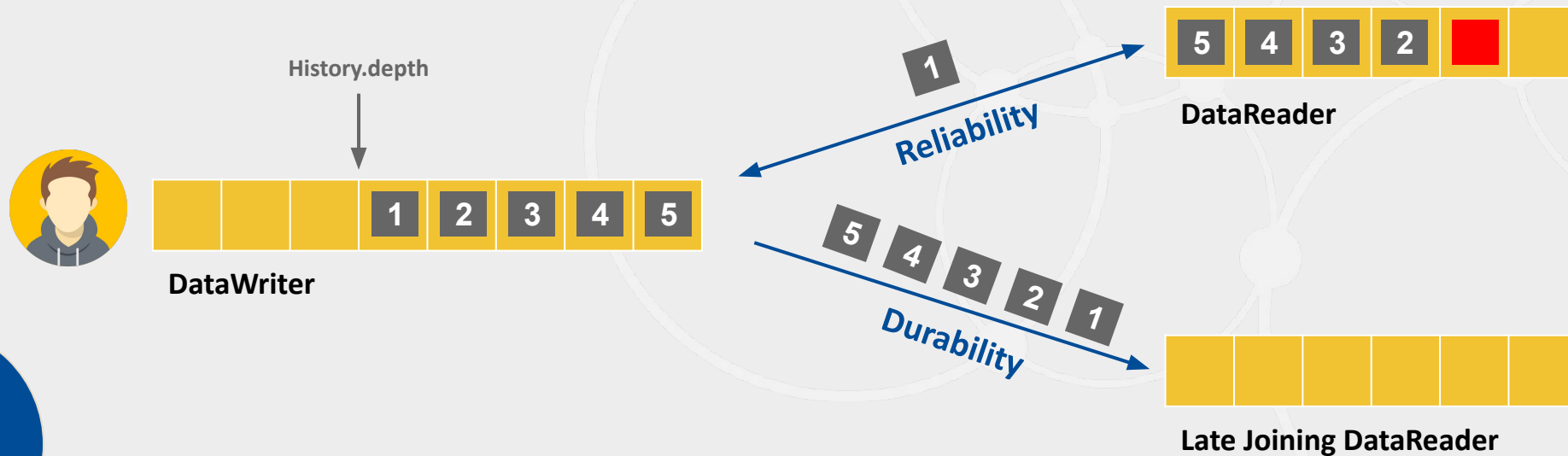
< 6.1  
behavior



# Challenge: Tight Coupling of Durability & Reliability

Durability and Reliability shared resources

- Previously no way to configure Durability separately from Reliability
- Large Reliability window to reduce sample loss resulted in high bandwidth for late joiners



< 6.1  
behavior

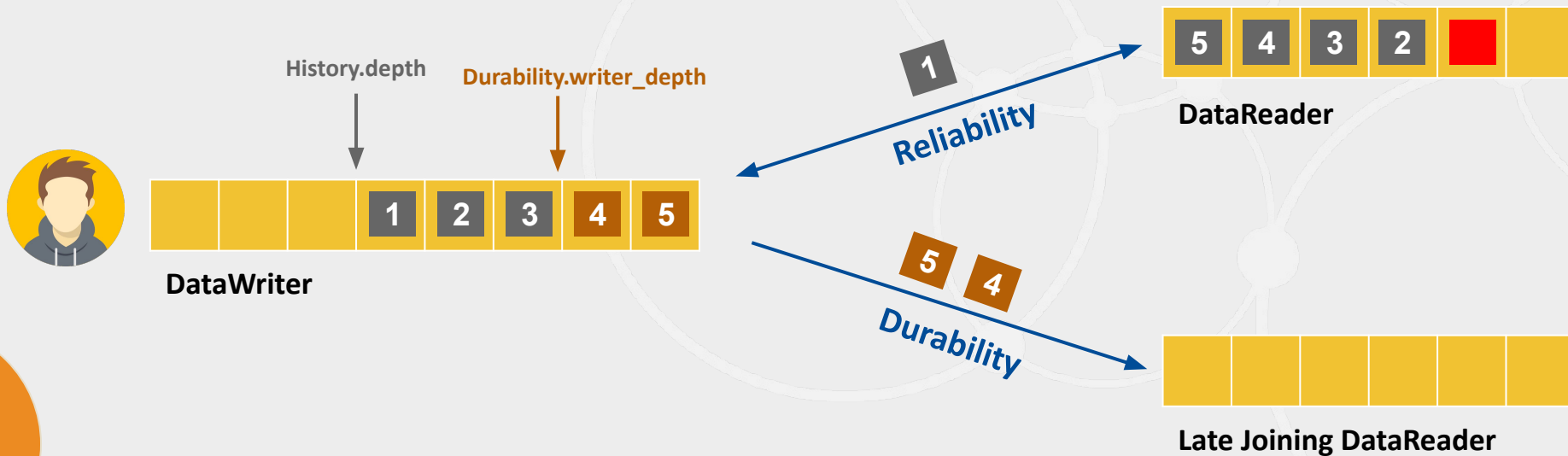






# Decoupled Durability from Reliability

QoS parameter	Purpose
<b>new</b> 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*



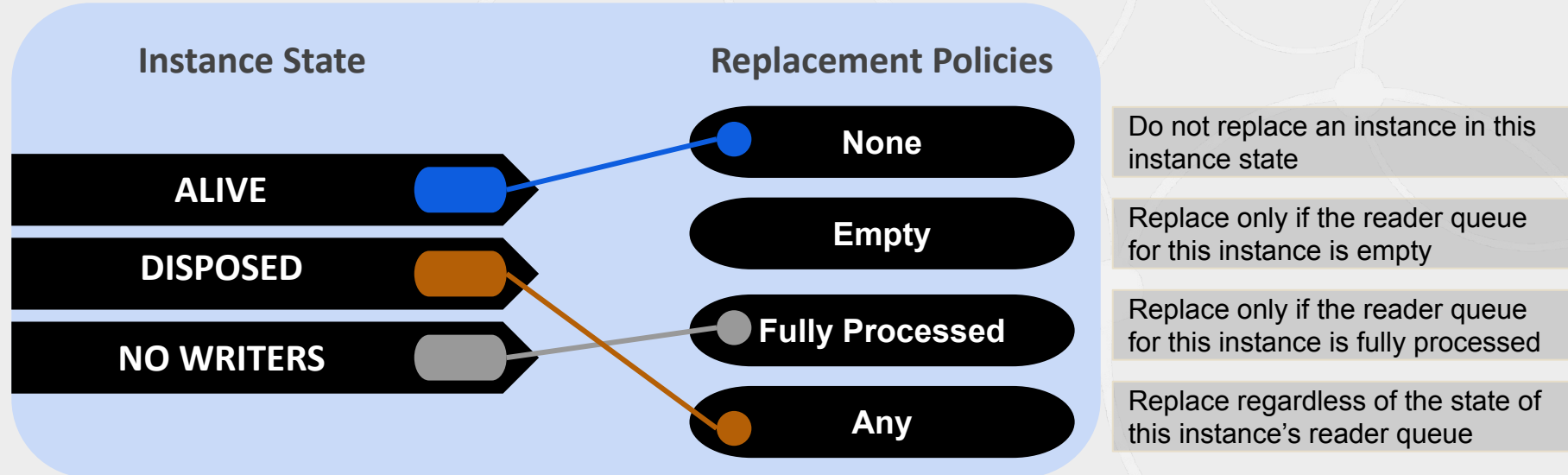
**New Player**

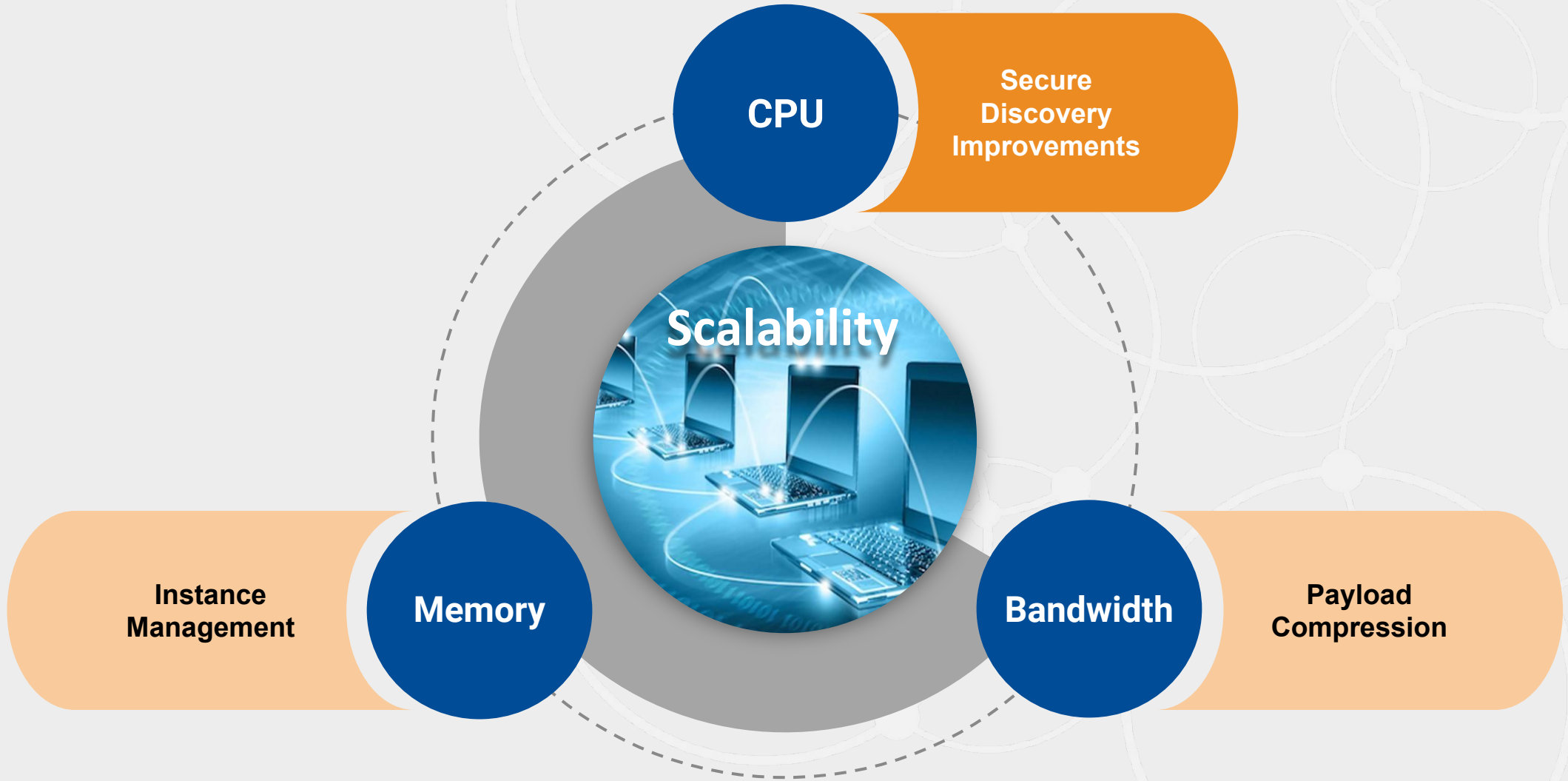
New player rejected because maximum number of players (instances) exceeded



# 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







# More Deterministic Detection of Communication Availability



- Key Problem:
  - Race condition between discovery completion and data flow starting
- Approach:

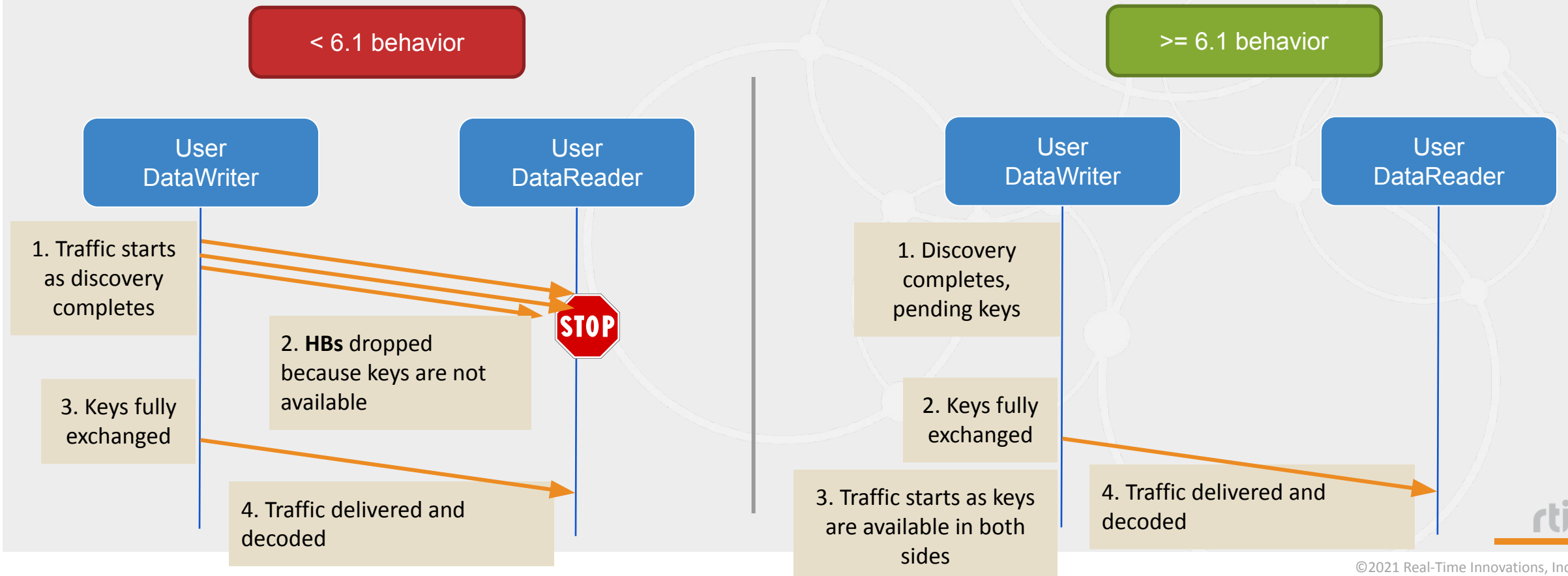
*Treat **cryptographic key exchange status** (key reception AND delivery) as **QoS compatibility**, which means:*

***DataWriter/DataReader not compatible with remote DataReader/DataWriter until keys are fully exchanged***

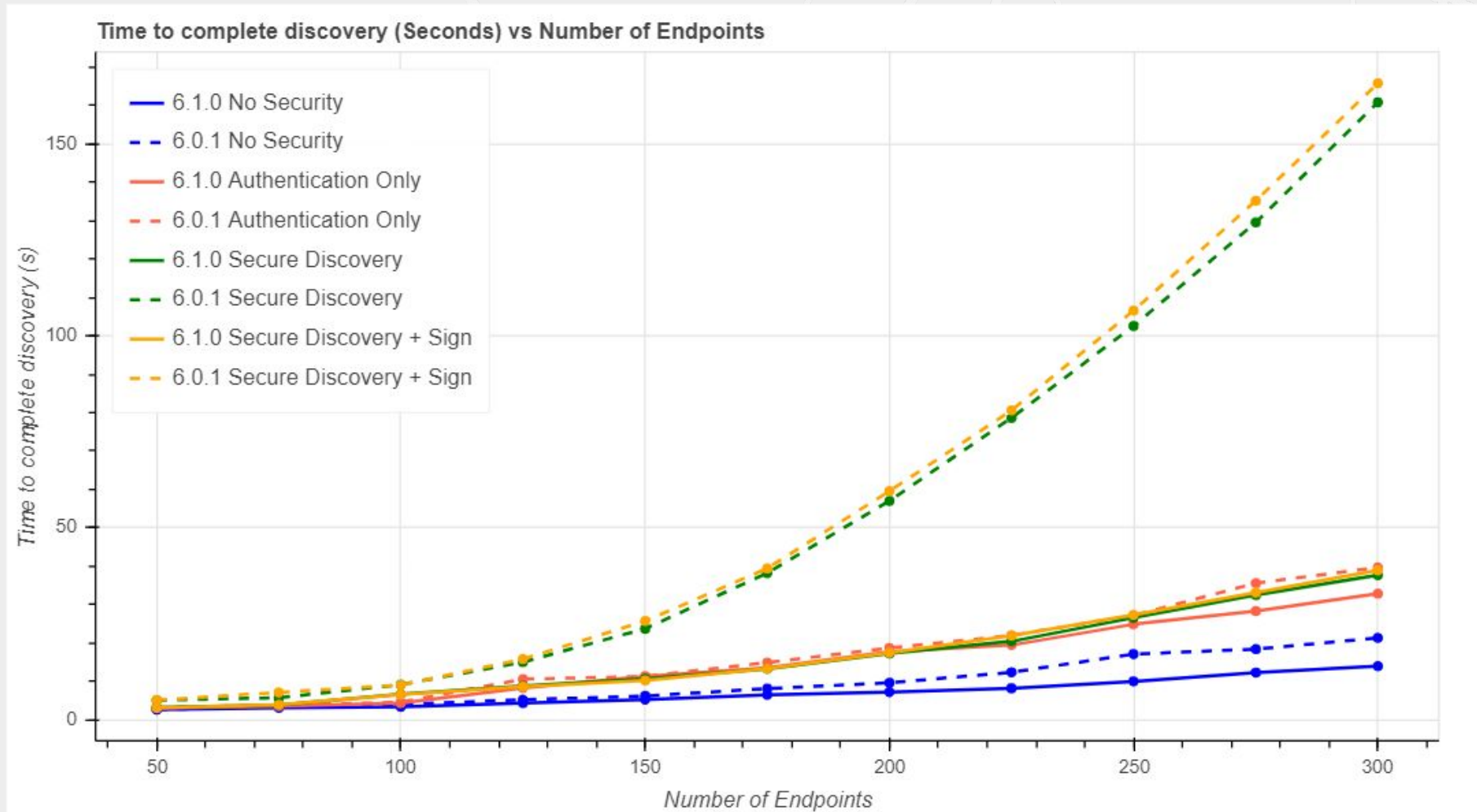
# More Deterministic Detection of Communication Availability



- Using crypto key exchange status to evaluate compatibility enables for a much more efficient system startup



# Significant Impact in Real-World Discovery Perf.



# Agenda

 WAN Connectivity

 Performance and Scalability

 **Security - sneak preview on 6.1.1**





# Security 6.1.1 update

---

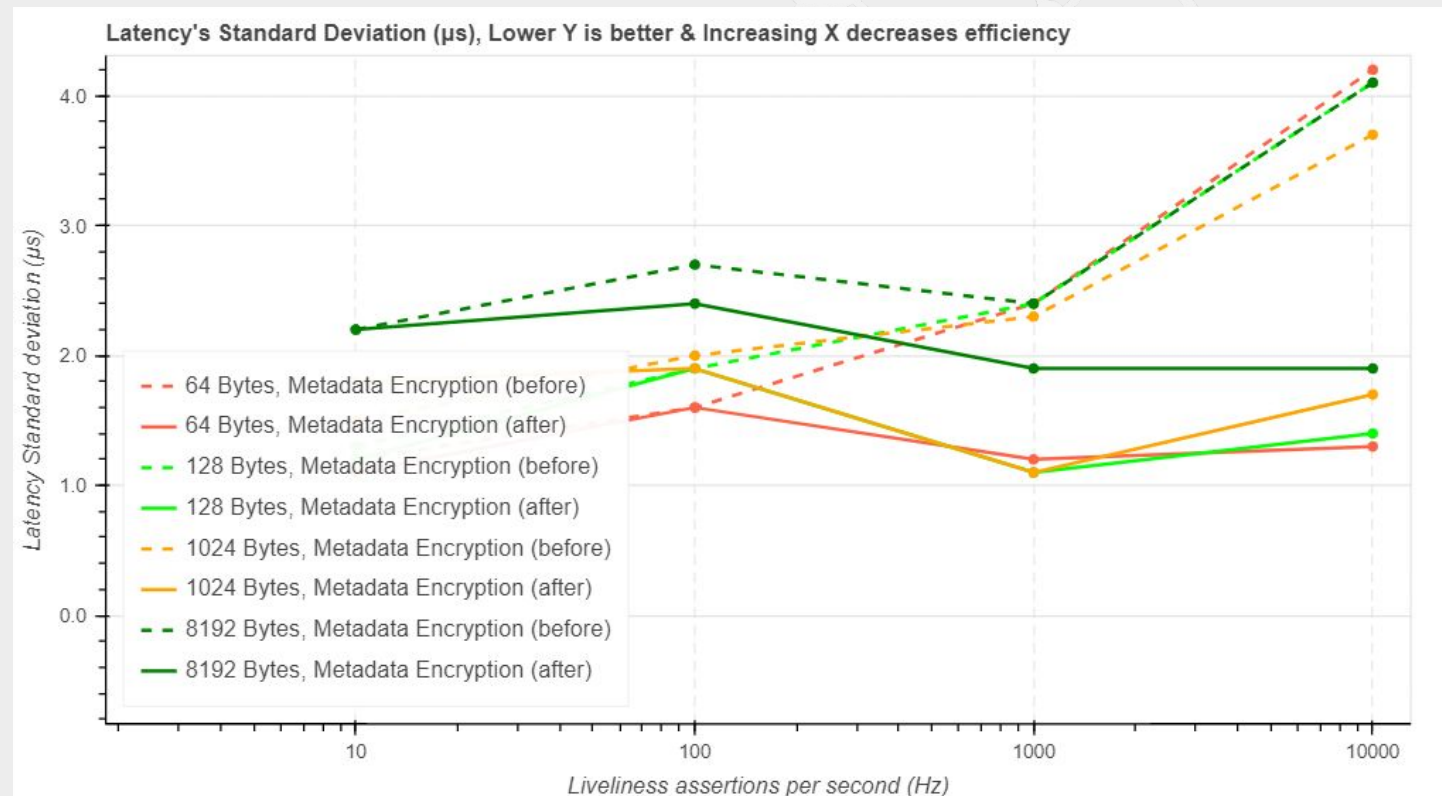
A sneak peak on the upcoming Connex Secure 6.1.1 release!



# Optimized submessage protection



- Latency jitter has been greatly reduced



# Decoupling Upgrade of OpenSSL version from Connex



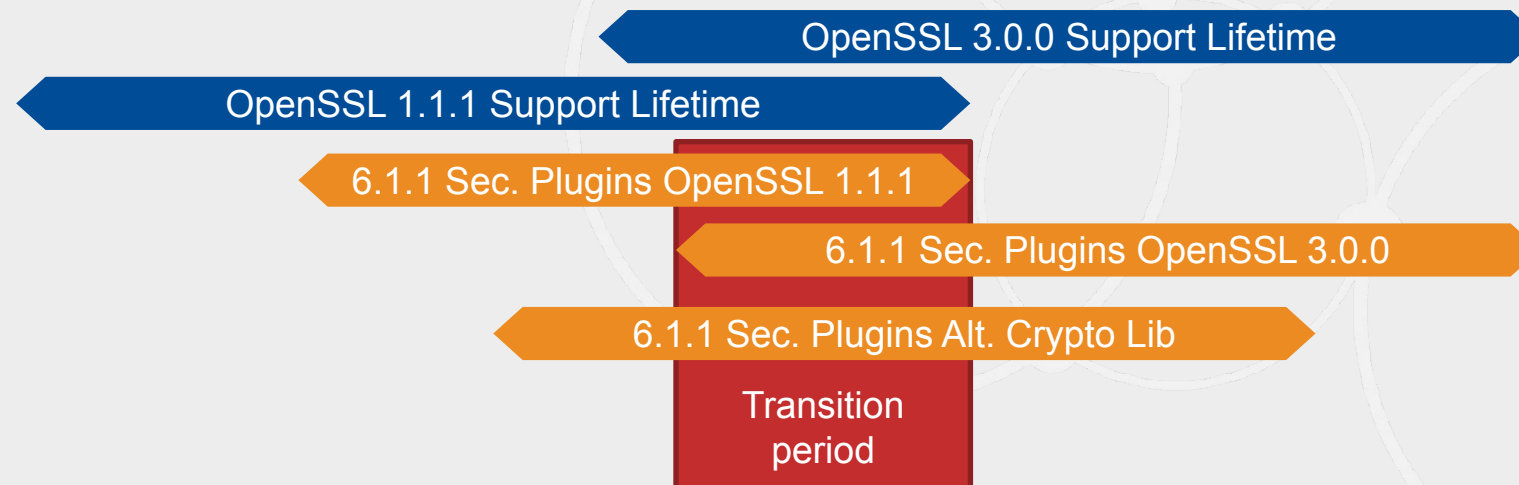
- **Common problem:** Connex libraries support model does not match OpenSSL support model
  - OpenSSL goes EOS before Connex goes
  - Updating OpenSSL version for a Connex version often requires ABI/API breaking changes that are not acceptable as a bug fix patch



# Decoupling Upgrade of OpenSSL version from Connex



- **Solution:** Connex Secure 6.1.1 supports adding new crypto-library flavors to the Security Plugins in the future
  - Alternative **OpenSSL** versions
  - Alternative **commercial crypto** libraries





# Security Plugins + WolfSSL

- Supported starting with Connex Secure 6.1.1
- Based on **WolfSSL 4.7**
- Available upon request
- Shipped as *alternative* version of the security plugins installers + libraries
- Also supported by the Security Plugins SDK

# Security Plugins + WolfSSL: Benefits

- **Commercial** implementation
- **Modular** implementation
- Offers a **potential path for FIPS**
  - OpenSSL FIPS dropped to historical list
  - OpenSSL 3.0 FIPS not ready yet



# Security Plugins + WolfSSL: Support

- **Wire-interoperable** with Security Plugins - OpenSSL
- **Supports most of existing Security Plugins features.**
  - Exceptions:
    - DSA (only ECDSA and RSA)
    - DH (only ECDH)
    - x509 v3 key usage extensions
    - OpenSSL specific features:
      - Engines
      - Store

# New Security Plugins SDK Test Suite



- **Common problem:** Users were missing a way to validate **Security Plugins** customizations.
  - Adding a complete test suite is very **time-consuming**.
  - Makes **modifying** the Security Plugins SDK **hard for Users**.
  - No way to test **alternative crypto libraries** (e.g., wolfSSL).
  
- **Solution:** New **Security Plugins SDK Test Suite**

# New Security Plugins SDK Test Suite



- **Included** as part of the **6.1.1 Security Plugins SDK** bundle
- Includes **300+** unit and feature **tests**
  - Set will grow as new features are added to the product
- Covers full Security Plugins functionality
  - Authentication
  - Access Control
  - Cryptography
  - Logging



# New Security Plugins SDK Test Suite



- Cmake-generated build system
- Depends on Cmocka 3<sup>rd</sup> party library
- Supported out of the box in the same platforms as the SDK:
  - Linux x64, Windows x64 \*, and MacOS x64.



**cmocka**

\* Visual Studio 2012 not supported





# Try a full version of Connex DDS for 30 days

TRY CONNEXT AT  
[RTI.COM/DOWNLOADS](https://rti.com/downloads)

Includes resources to get  
you up and running fast

# Stay Connected



[rti.com](https://rti.com)  
*Free trial of Connex DDS*



[rtisoftware](https://www.facebook.com/rtisoftware)



[@rti\\_software](https://twitter.com/rti_software)



[connexpodcast](#)



[@rti\\_software](https://www.instagram.com/rti_software)



[rti.com/blog](https://rti.com/blog)

