WEBINAR

Using OpenDDS's RtpsRelay to Connect IoT/IIoT Applications

Webinar Host: Jen Wiese
Presenter: Justin Wilson

August 25, 2022

OpenDDS Foundation™
12140 Woodcrest Exec. Dr., Ste. 300
Saint Louis, MO 63141 USA

Limits of Liability and Disclaimer of Warranty

While every precaution has been taken in preparing this material, including research, development and testing, OpenDDS Foundation assumes no responsibility for errors or omissions. No liability is assumed by OpenDDS Foundation for any damages resulting from the use of this information.

Upcoming Events

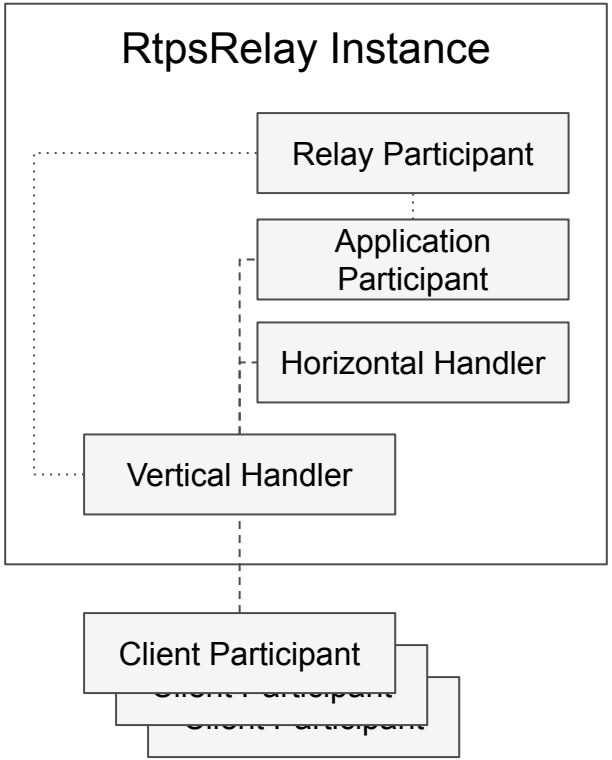
Online Training Classes in October 2022:

- Introduction to OpenDDS Programming (C++, Java)
- Building OpenDDS Applications with DDS Security (C++, Java)
- OpenDDS Essentials I (C++, Java)
  - QoS, Keys and Instances, & Built-In Topics
- OpenDDS Essentials II (C++, Java)
  - Configuration, Listeners, Conditions, & Content-Subscription

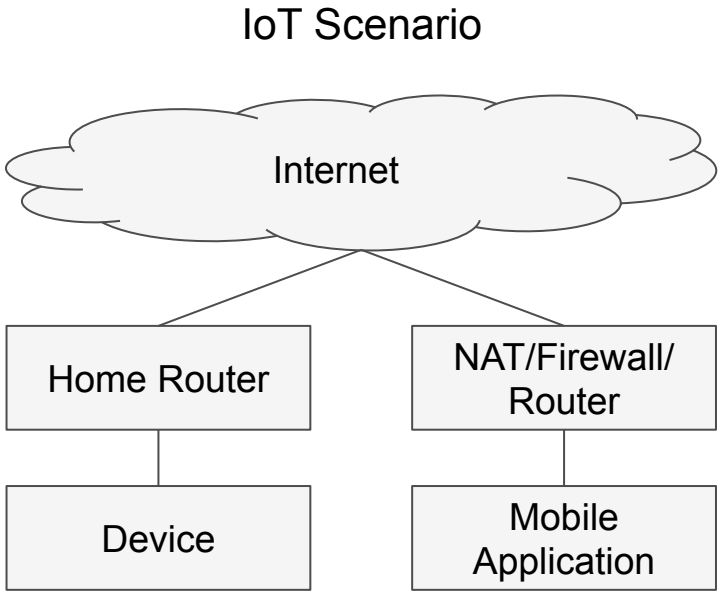
https://objectcomputing.com/services/training/catalog/middleware

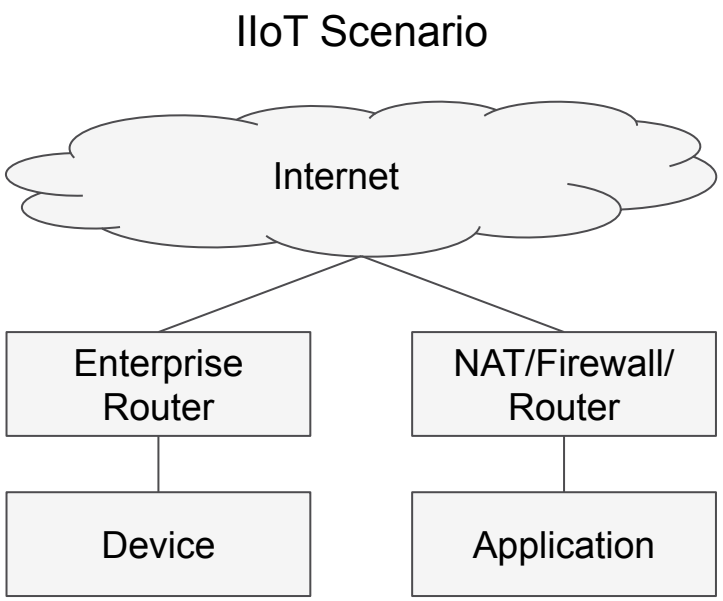
Agenda

- Welcome and Introductions
- Motivation for the RtpsRelay
- Design of the RtpsRelay
- Operation of the RtpsRelay
- Configuring the RtpsRelay
- Monitoring and Logging
- Using the RtpsRelay
- Gotchas
- Q&A



Motivation for the RtpsRelay: Scenarios





Motivation for the RtpsRelay: RTPS

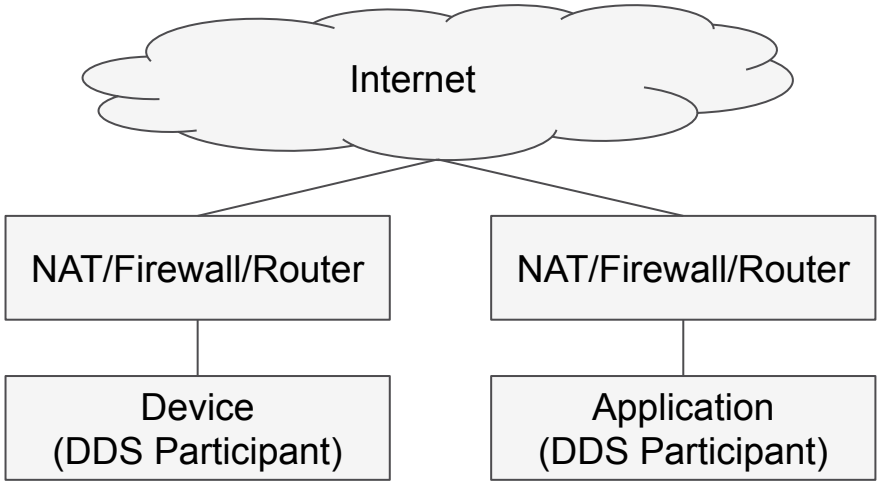
We want our devices and applications to be DDS Participants

RTPS (Real-Time Publish Subscribe) is a UDP protocol that gives us

- Interoperability
- Security
- XTypes

RTPS Discovery works by using multicast to advertise an IP address that a peer can use to contact a participant

Motivation for the RtpsRelay: Connectivity Obstacles

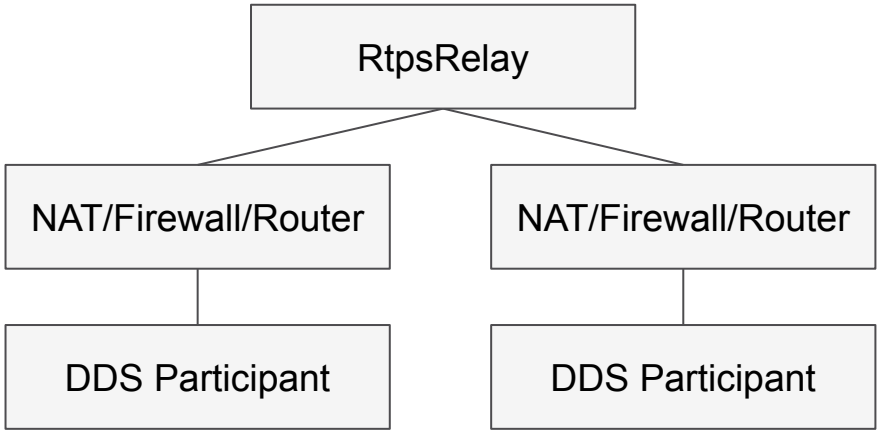


Multicast doesn't work over the public internet

No stable public IP address to advertise

No direct connectivity

Motivation for the RtpsRelay: Problem Statement



Create a horizontally scalable service called the RtpsRelay that can forward RTPS messages between connected participants

Exposing this service on the Internet allows participants without direct connectivity to communicate

No modifications to the RTPS protocol

DDS and RTPS Refresher

A list of *partitions* can be associated with a Publisher or Subscriber

SPDP - Simple *Participant* Discovery Protocol

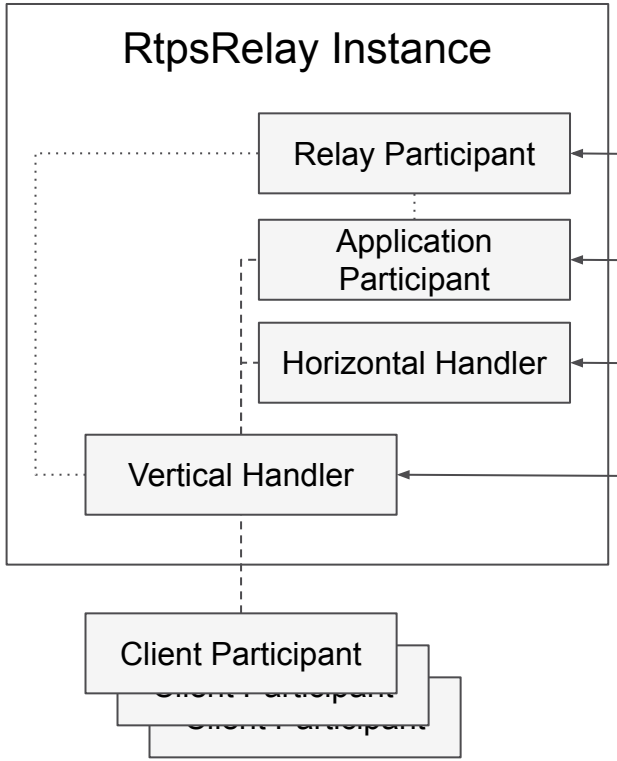
Participants are identified by a unique 12 byte string called the *GuidPrefix*

SEDP - Simple *Endpoint* Discovery Protocol

- An endpoint corresponds to a writer/publication or reader/subscription
- An endpoint is identified by the GuidPrefix + 4 bytes

RTPS messages can contain zero or more INFO_DST sub-messages that contain the GuidPrefix of the intended recipient

Design of the RtpsRelay: Single Instance



Gossips “routing table” and other information

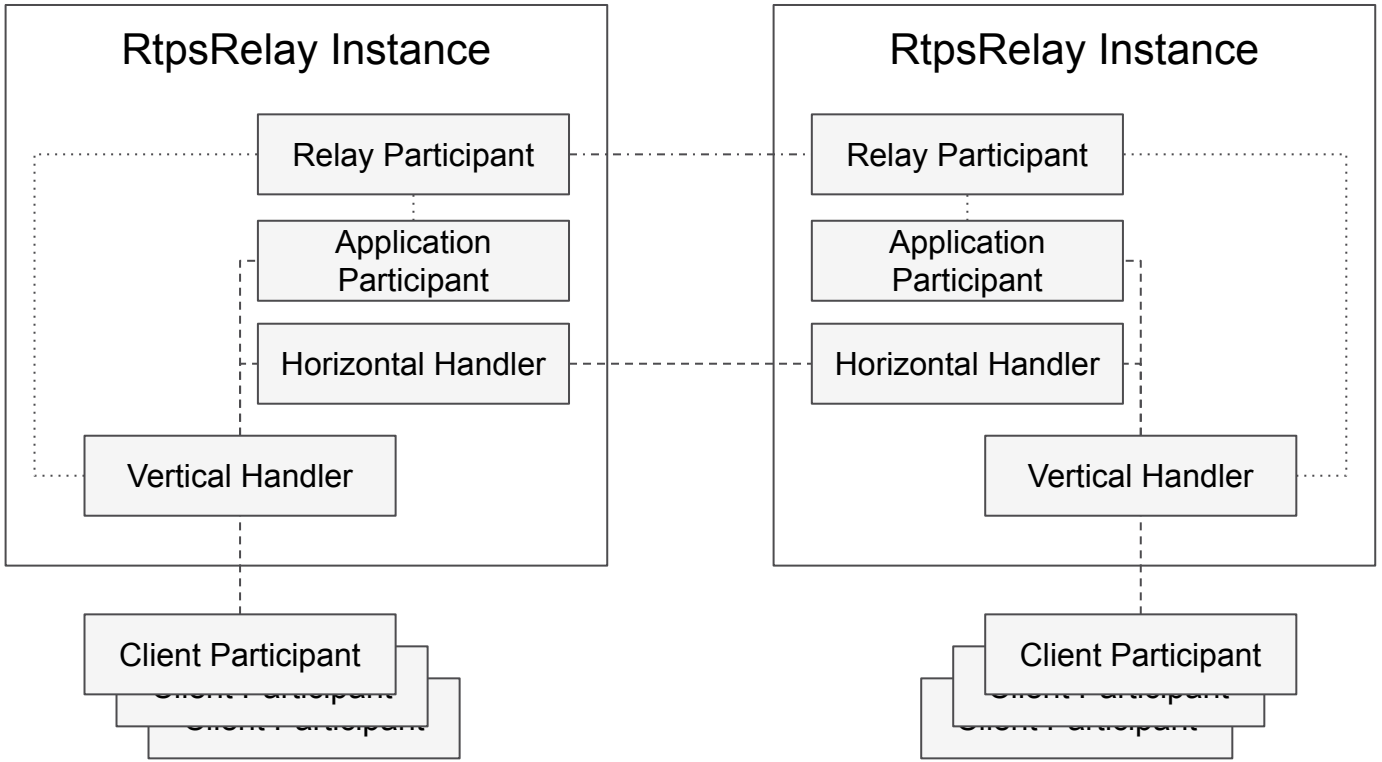
Discovers Client Participants

Exchanges wrapped RTPS Messages with other instances

Exchanges RTPS Messages with Participants

Vertical and Horizontal Handler replicated three times:
SPDP, SEDP, DATA

Design of the RtpsRelay: Horizontal Scaling



Design of the RtpsRelay: RtpsRelay is an OpenDDS App

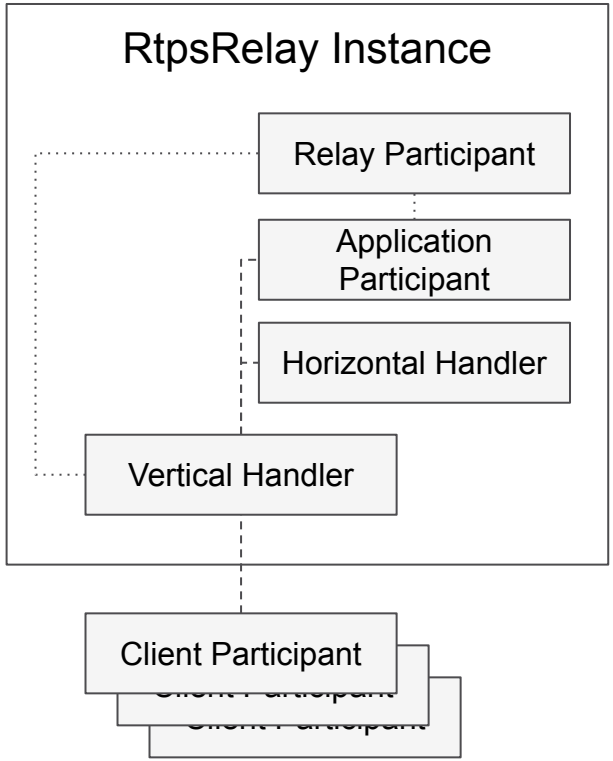
Application Participant

- DDS Participant in the same domain as the Client Participants
- (Securely) Discovers readers/writers and partitions
- Builds local routing table (Client Participant, Partitions)
- Aggregates partitions and sends to Relay Participant

Relay Participant

- DDS Participant in a different domain than the Application Participant
- Discovers other RtpsRelay Instances
- Shares partitions that are interesting to this instance
- Shares status and statistics

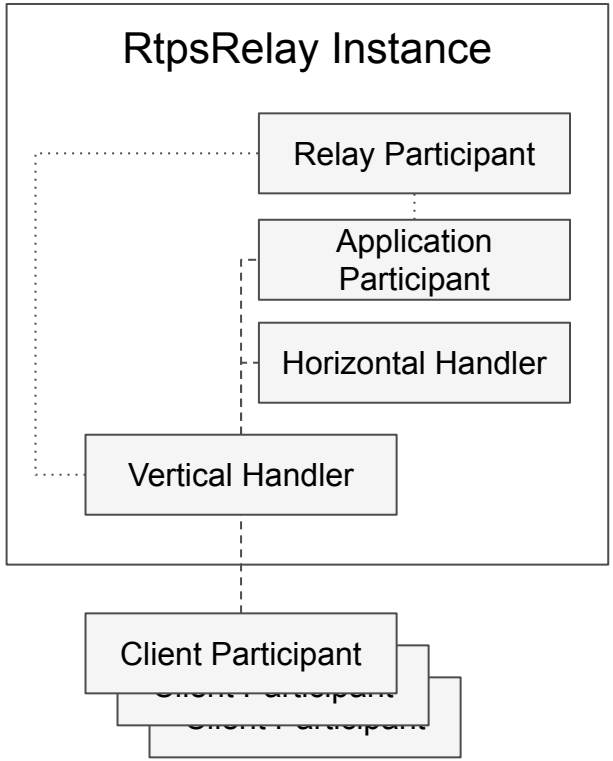
Operation of the RtpsRelay: SPDP



1. Client Participant sends SPDP to Vertical Handler
2. Vertical Handler forwards message to Application Participant
3. Application Participant sends (own) SPDP to Vertical Handler
4. Vertical Handler forwards to Client Participant

Client Participant discovery is complete.

Operation of the RtpsRelay: SEDP



1. Client Participant sends SEDP to Vertical Handler
2. Vertical Handler forwards message to Application Participant
3. Application Participant sends (own) SEDP to Vertical Handler
4. Vertical Handler forwards to Client Participant

Endpoint discovery complete.

Operation of the RtpsRelay: Routing Tables

Application Participant uses built-in topics to record the partitions associated with each Publication/Subscription in lookup tables:

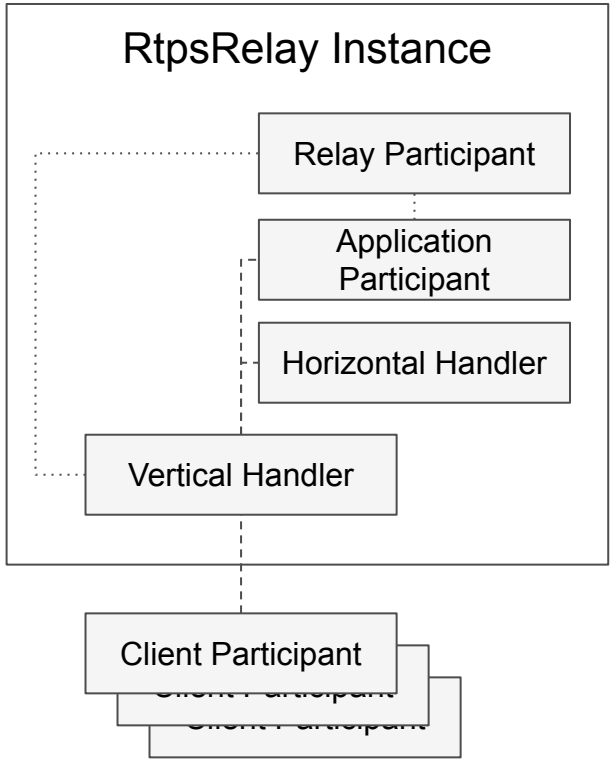
- GuidPrefix -> Partitions
- Partition -> GuidPrefix

The complete list of Partitions is published to other instances via the Relay Participant

Each instance builds a lookup table:

- Partition -> Instances

Operation of the RtpsRelay: SPDP, SEDP, DATA



1. Client Participant sends message to Vertical Handler
2. Look up partitions for sending Client Participant
3. Extract destination GUIDs
4. Look up interested instances using partitions
5. Prefix RTPS message with partitions and GUIDs and send to Horizontal Handler of interested instances
6. Horizontal Handler processes wrapped message
   a. If destination GUIDs, use Vertical Handler to send to any connected Client Participants
   b. Else look up GUIDs using partitions and use Vertical Handler to send

Operation of the RtpsRelay: Other Features

Undirected SPDP/SEDP is forwarded to Application Participant

Optimization to avoid sending to own horizontal handler

Most recent SPDP message for a Client Participant is cached and “replayed” when a new Client Participant is discovered in a shared partition

- This was added to accelerate discovery among Client Participants
- Will be removed in a future release (new Client Participant will resend SPDP)

Operation of the RtpsRelay: Connectivity Checks

RtpsRelay maintains a lookup table:

- Client Participant -> IP:port

NATs close ports after a period of inactivity

Client Participants must send traffic to Vertical Handlers to keep ports open and mapping up-to-date

This is done using a STUN Binding Requests and STUN Binding Indications

IETF STUN specification: https://www.rfc-editor.org/info/rfc8489

Configuring the RtpsRelay: General

-Id	ID of the RtpsRelay
-HorizontalAddress	Listening address:port for Horizontal Handlers
-VerticalAddress	Listening address:port for Vertical Handlers
-RelayDomain	The DDS Domain for the Relay Participant
-ApplicationDomain	The DDS Domain for the Application Participant
-UserData	User data string for Application Participant
-BufferSize	Size of send and receive buffers
-Lifespan	Duration of Client Participant IP:port records
-InactivePeriod	Mark Client Participant as inactive after this duration
-AllowEmptyPartitions	Allow Client Participants with no partitions

Configuring the RtpsRelay: Security

-IdentityCA	Path to Identity CA Certificate
-PermissionsCA	Path to Permissions CA Certificate
-IdentityCertificate	Path to certificate for Application Participant
-IdentityKey	Path to key for Application Participant
-Governance	Path to governance file
-Permissions	Path to permissions file
-RestartDetection	Use hash of subject name in GUID to detect restarts

Configuring the RtpsRelay: Logging

-LogWarnings	Log warnings
-LogDiscovery	Log messages received on the Application Participant's built-in-topics
-LogActivity	Log messages concerning the IP:port mapping for a Client Participant
-LogRelayStatistics -PublishRelayStatistics	Log/Publish statistics for the relay on this interval
-LogHandlerStatistics -PublishHandlerStatistics	Log/Publish statistics for handlers on this interval
-LogParticipantStatistics -PublishParticipantStatistics	Log/Publish statistics for individual Client Participants when they are removed

Logging: Examples

```
2022-08-01 14:26:02.898@LM_INFO@(93971|123145367801856) INFO:
RelayParticipantStatusReporter::add_participant add local participant
0103784f.439cae5c.6f1d525a.000001c1
{"key":{"value":[1,3,120,79,67,156,174,92,111,29,82,90,0,0,1,193]},"user_data":{"value":[84,10
4,101,32,80,117,98,108,105,115,104,101,114]}} 0.005 s into session
```

```
2022-08-01 14:25:53.903@LM_INFO@(93971|4376229376) INFO: GuidAddrSet::record_activity
0103784f.439cae5c.6f137135.000001c1 added 0.000 s into session
```

```
2022-08-01 14:25:53.903@LM_INFO@(93971|4376229376) INFO: GuidAddrSet::record_activity VSEDP
0103784f.439cae5c.6f137135.000001c1 is at 127.0.0.1:50126 0.000 s into session total=1
```

```
2022-08-01 14:25:53.903@LM_INFO@(93971|4376229376) INFO: GuidAddrSet::record_activity
0103784f.439cae5c.6f137135.000001c1 was admitted 0.000 s into session
```

```
2022-08-01 14:25:53.904@LM_INFO@(93971|4376229376) INFO: GuidAddrSet::record_activity VSPDP
0103784f.439cae5c.6f137135.000001c1 is at 127.0.0.1:17910 0.001 s into session total=1
```

Logging: Examples


```
2022-08-01 14:26:02.892@LM_INFO@(93971|4376229376) STAT: Relay Statistics
{"relay_id":"relay1","interval":{"sec":5,"nanosec":914555000},"rtps":{"messages_in":0,"bytes_in":0,"
messages_ignored":0,"bytes_ignored":0,"input_processing_time":{"sec":0,"nanosec":0},"messages_out":0
,"bytes_out":0,"messages_dropped":0,"bytes_dropped":0,"output_processing_time":{"sec":0,"nanosec":0}
},"stun":{"messages_in":0,"bytes_in":0,"messages_ignored":0,"bytes_ignored":0,"input_processing_time
":{"sec":0,"nanosec":0},"messages_out":0,"bytes_out":0,"messages_dropped":0,"bytes_dropped":0,"outpu
t_processing_time":{"sec":0,"nanosec":0}},"max_gain":0,"error_count":0,"local_active_participants":1
,"new_address_count":0,"expired_address_count":0,"max_queue_size":1,"max_queue_latency":{"sec":0,"na
nosec":0},"local_participants":0,"local_writers":0,"local_readers":0,"relay_partitions_pub_count":2,
"relay_address_pub_count":2,"spdp_replay_pub_count":2}

2022-08-01 14:26:04.758@LM_INFO@(93971|123145367801856) STAT: Participant Statistics
{"guid":{"guidPrefix":[1,3,120,79,67,156,174,92,111,29,82,90],"entityId":{"entityKey":[0,0,1],"entit
yKind":193}},"name":"SPDP","session_time":{"sec":1,"nanosec":865322000},"rtps":{"messages_in":9,"byt
es_in":2228,"messages_ignored":0,"bytes_ignored":0,"input_processing_time":{"sec":0,"nanosec":0},"me
ssages_out":6,"bytes_out":1632,"messages_dropped":0,"bytes_dropped":0,"output_processing_time":{"sec
":0,"nanosec":0}},"stun":{"messages_in":2,"bytes_in":88,"messages_ignored":0,"bytes_ignored":0,"inpu
t_processing_time":{"sec":0,"nanosec":0},"messages_out":1,"bytes_out":52,"messages_dropped":0,"bytes
_dropped":0,"output_processing_time":{"sec":0,"nanosec":0}}}
```

Configuring the RtpsRelay: Monitoring

-LogThreadStatus	Log the status of the threads in the RtpsRelay
-ThreadStatusSafetyFactor	Restart if thread monitoring is enabled and the thread has not checked in for this many reporting intervals
-UtilizationLimit	Don't accept new Client Participants if the CPU utilization of any thread is above this limit
-PublishRelayStatusLiveliness	Liveliness uses for publishing Relay Status
-PublishRelayStatus	Publish Relay Status on this interval

Logging: Examples


```
2022-08-01 14:26:01.723@LM_INFO@(93971|123145371557888) INFO: Thread Status
{"thread_id":"4376229376 (RtpsRelay Main)","utilization":0.00011174211970922023}
{"sample_state":1,"view_state":2,"instance_state":1,"source_timestamp":{"sec":1659381959,"nano
sec":717466000},"instance_handle":31,"publication_handle":0,"disposed_generation_count":0,"no_
writers_generation_count":0,"sample_rank":0,"generation_rank":0,"absolute_generation_rank":0,"
valid_data":true,"opendds_reserved_publication_seq":1}
```

Monitoring: Relay Participant Topics

The Relay Participant publishes a variety of topics:

- Relay Partitions - partitions that a relay instance is interested in
- Relay Addresses - horizontal addresses used by instances
- Relay Status - indicates if an instance is admitting new Client Participants
- Relay Participant Status - alive, active, and user data for Client Participants
- SPDP Replay - replay requests for partitions
- (Client) Participant Statistics
- Handler Statistics
- Relay Statistics

Configuring the RtpsRelay: Example config file

```
[common]
DCPSGlobalTransportConfig=myconfig
DCPSThreadStatusInterval=5                  # Enable thread status monitoring for admission control

[config/myconfig]
passive_connect_duration=3600
transports=rtps

[domain/1]                                  # This is for the Relay Participant
DiscoveryConfig=relay_rtps_discovery

[rtps_discovery/relay_rtps_discovery]
CheckSourceIp=0
SedpMulticast=0
SedpMaxMessageSize=1400

[transport/rtps]
transport_type=rtps_udp
use_multicast=0
max_message_size=1400
```

Configuring the RtpsRelay: Example config file cont.

```
[domain/0]                                  # This is for the Application Participant
DiscoveryConfig=application_rtps_discovery

[rtps_discovery/application_rtps_discovery]
CheckSourceIp=0
SedpMulticast=0                             # Don't use multicast
SpdpRtpsRelayAddress=127.0.0.1:4445         # Application Participant talks to local vertical ports
SedpRtpsRelayAddress=127.0.0.1:4446
RtpsRelayOnly=1                             # Application Participant only talks to local vertical ports
MinResendDelay=0
SedpMaxMessageSize=1400                     # Set based on Internet MTU
ResendPeriod=3600                           # Send SPDP announcement once every hour
LeaseDuration=14400                         # Very long lease duration
UndirectedSpdp=0                            # Do not send undirected SPDP messages
PeriodicDirectedSpdp=1                      # Periodically send directed SPDP messages
AuthResendPeriod=.1                         # Timing parameters
SedpHeartbeatPeriod=200
SedpNakResponseDelay=100
SedpResponsiveMode=1
SendBufferSize=2097152                      # Use bigger send and receive buffers
RecvBufferSize=2097152
SedpPassiveConnectDuration=14400000         # Wait a long time for association
```

Use the RtpsRelay: Client Participant Configuration

For `rtps_discovery`, in config file

- `SpdpRtpsRelayAddress`
- `SedpRtpsRelayAddress`

For `transport` (must be `transport_type=rtps_udp`), in config file

- `DataRtpsRelayAddress`

These settings are also available via the RtpsDiscovery API and RtpsUdpTransport API.

They can be changed at any time.

Use the RtpsRelay: Custom Built-In Topics

The ParticipantLocation Topic has flags that indicate a participant is reachable via the RtpsRelay.

The ConnectionRecord Topic contains samples for RtpsRelay connections.


```
2022-08-01
14:26:02.897@LM_INFO@{"topic":{"name":"OpenDDSConnectionRecord","type_name":"CONNECTION_RECORD
_BUILT_IN_TOPIC_TYPE"},"sample":{"guid":[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],"address":"127.0.0.1
:4444","protocol":"RtpsRelay:STUN","latency":{"sec":0,"nanosec":1313000}},"sample_info":{"samp
le_state":2,"view_state":1,"instance_state":1,"source_timestamp":{"sec":1659381962,"nanosec":8
97025000},"instance_handle":6,"publication_handle":0,"disposed_generation_count":0,"no_writers
_generation_count":0,"sample_rank":0,"generation_rank":0,"absolute_generation_rank":0,"valid_d
ata":true,"opendds_reserved_publication_seq":1}}
```

Gotchas

Design with Partitions	For efficient message routing
Bootstrapping the Relay Participants	Multicast may not be supported
Use DDS Security with restart detection	Because clients can get into a crash loop
Use Thread Monitoring and Admission Control	Because of restart under load
Firewalls block UDP messages	Monitoring ConnectionRecords and RtpsRelay logging can diagnose problem
Firewalls close ports	Ensure that STUN connectivity checks are frequent enough
IP Fragmentation not globally supported	Configure fragmentation at the RTPS level so IP Fragmentation doesn’t occur

Thank you!

Any Questions?

OpenDDS FOUNDATION™

Articles and Talks

On-demand Webinar:

- Designing a Secure Cloud-Enabled Peer-to-Peer IoT Application
  - objectcomputing.com/resources/publications/mnb/2019/06/20/interoperable-internet-enabled-dds-applications

Articles:

- Interoperable Internet-Enabled DDS Applications
  - objectcomputing.com/resources/publications/mnb/2019/06/20/interoperable-internet-enabled-dds-applications
- Bringing Multicast to the Cloud for Interoperable DDS Applications
  - objectcomputing.com/resources/publications/mnb/2019/03/01/bringing-multicast-cloud-interoperable-dds-applications

Further Study: On-Demand Webinars

Data Distribution with an Open and Secure DDS (DDS Security)

objectcomputing.com/resources/events/webinars/opendds-security

Designing a Distributed Application using DDS QoS

brighttalk.com/webcast/12231/281491

XTypes in OpenDDS 3.16

objectcomputing.com/products/opendds/resources/introducing-xtypes

Getting Started as an OpenDDS Code Contributor

objectcomputing.com/products/opendds/resources/opendds-code-contribution-tutorial

OpenDDS Resources

OpenDDS project

opendds.org

Source repo

github.com/objectcomputing/OpenDDS

OpenDDS support, training, consulting, development

objectcomputing.com/products/opendds

OpenDDS 3.21 Release Notes

github.com/objectcomputing/OpenDDS/releases/tag/DDS-3.21

OpenDDS Foundation

OpenDDS Foundation is a not-for-profit organization that exists to support and collectively lead the open source OpenDDS® project. The Foundation is supported by a Technology Advisory Board that ensures the technology continues to reflect and serve its diverse and growing user community.

OpenDDS Foundation works to ensure technical innovation and advancement of the OpenDDS project, evangelize and promote the project as a leading technology in the data distribution space, and build and support an ecosystem of complementary documentation, functionality, and services.

As a not-for-profit organization, OpenDDS Foundation relies on the financial support of contributing members to support and grow the project. Businesses and community members are encouraged to actively participate in the project's success by becoming contributing members through one of our sponsorship programs.

LET'S CONNECT!


info@opendds.org


opendds.org/foundation


@OpenDDS


linkedin.com/showcase/opendds


github.com/objectcomputing/OpenDDS