Prerequisite:

* Used version 2 of the dds-app image.
* Allow $APP\_ARGS to provide domainId
* Don’t need to expose ports(service) anymore

Steps:

1. Pub-sub on Openshift using multicast

|  |
| --- |
| # new project $ oc new-project pub-sub  # new publisher with domainId = 7 $ oc new-app --name publisher --docker-image=docker.io/pnminh/publisher --env APP\_ARGS=7 # new subscriber with domainId = 7 $ oc new-app --name subscriber --docker-image=docker.io/pnminh/subscriber --env APP\_ARGS=7 # no communication due to multicast is disabled $ oc logs dc/subscriber --since 10s HelloWorld subscriber sleeping for 4 sec... HelloWorld subscriber sleeping for 4 sec... HelloWorld subscriber sleeping for 4 sec…  # enable multicast $ oc annotate netnamespace pub-sub netnamespace.network.openshift.io/multicast-enabled=true netnamespace.network.openshift.io/pub-sub annotated # communication begins as multicast is enabled $ oc logs dc/subscriber --since 10s Received data   msg: "Hello World! count:141, time:Tue Apr 14 18:40:44 2020 " HelloWorld subscriber sleeping for 4 sec... Received data   msg: "Hello World! count:142, time:Tue Apr 14 18:40:48 2020 " HelloWorld subscriber sleeping for 4 sec…  # use a different domainId for publisher $ oc set env dc/publisher --env APP\_ARGS=6 deploymentconfig.apps.openshift.io/publisher updated # no communication due to publisher and subscriber using different domainIds $ oc logs dc/subscriber --since 10s HelloWorld subscriber sleeping for 4 sec... HelloWorld subscriber sleeping for 4 sec... HelloWorld subscriber sleeping for 4 sec... # update subscriber to use the same domainId with the publisher $ oc set env dc/subscriber --env APP\_ARGS=6 deploymentconfig.apps.openshift.io/subscriber updated # communication started successfully as expected as pub-sub use same domainId $ oc logs dc/subscriber --since 10s Received data   msg: "Hello World! count:46, time:Tue Apr 14 18:46:18 2020 " HelloWorld subscriber sleeping for 4 sec... Received data   msg: "Hello World! count:47, time:Tue Apr 14 18:46:22 2020 " |

1. Routing services
2. Routing service on OCP

Copy files to build routing services:

|  |
| --- |
| $ mkdir routing-service && cd routing-service $ cp -rf $NDDSHOME/resource/app/lib/x64Linux2.6gcc4.4.5 ./lib $ cp -rf $NDDSHOME/resource/app/bin/x64Linux2.6gcc4.4.5/rtiroutingservice . |

Create new Dockerfile for routing service:

|  |
| --- |
| FROM ubuntu:18.04 WORKDIR /app COPY . /app # Add the apps to the PATH ENV PATH=$PATH:/app \  LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:/app/lib \  CFG\_NAME=default \  ARG= CMD ./rtiroutingservice -cfgFile ./config.xml -cfgName $CFG\_NAME $ARGS |

Create docker image and push to docker registry.

Create a new config.xml file for the routing service. There are 4 environment variables that need to be used at runtime to be used with the files:

* Public IP address and port: PUBLIC\_ADDR
* Server binding port: SERVER\_PORT
* Domain ID to be used with external routing service: WAN\_DOMAIN
* Domain ID to be used with the real services: LAN\_DOMAIN

<?xml version="1.0" ?>

<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:noNamespaceSchemaLocation="http://community.rti.com/schema/6.0.0/rti\_routing\_service.xsd">

<qos\_library name="TCP\_WAN\_Demo\_lib">

<qos\_profile name="TCP\_WAN\_Demo\_profile">

<participant\_qos>

<transport\_builtin>

*<!--<mask>UDPv4</mask>-->*

<mask>MASK\_NONE</mask>

</transport\_builtin>

<property>

<value>

<element>

<name>dds.transport.load\_plugins</name>

<value>dds.transport.TCPv4.tcp1</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.library</name>

<value>nddstransporttcp</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.create\_function</name>

<value>NDDS\_Transport\_TCPv4\_create</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.parent.classid</name>

<value>NDDS\_TRANSPORT\_CLASSID\_TCPV4\_WAN</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.public\_address</name>

<value>$(PUBLIC\_ADDR)</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.server\_bind\_port</name>

<value>$(SERVER\_PORT)</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.disable\_nagle</name>

<value>1</value>

</element>

</value>

</property>

</participant\_qos>

</qos\_profile>

</qos\_library>

<routing\_service name="RS-TCP">

<domain\_route name="TCPWAN\_UDPLAN">

<participant name="1">

<domain\_id>$(WAN\_DOMAIN)</domain\_id>

<participant\_qos base\_name="TCP\_WAN\_Demo\_lib::TCP\_WAN\_Demo\_profile" />

</participant>

<participant name="2">

<domain\_id>$(LAN\_DOMAIN)</domain\_id>

</participant>

<session name="Session">

<auto\_topic\_route name="Forward">

<publish\_with\_original\_info>true</publish\_with\_original\_info>

<input participant="1">

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

<creation\_mode>IMMEDIATE</creation\_mode>

</input>

<output participant="2">

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

<creation\_mode>IMMEDIATE</creation\_mode>

</output>

</auto\_topic\_route>

<auto\_topic\_route name="Backward">

<publish\_with\_original\_info>true</publish\_with\_original\_info>

<input participant="2">

<creation\_mode>IMMEDIATE</creation\_mode>

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

</input>

<output>

<creation\_mode>IMMEDIATE</creation\_mode>

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

</output>

</auto\_topic\_route>

</session>

</domain\_route>

</routing\_service>

</dds>

Create configmap from the config.xml file (to be used by the routing service)

|  |
| --- |
| $ oc create configmap routingservice-inbound --from-file=config.xml configmap/routingservice-inbound created |

Create routing service statefuleset resource (routingservice\_statefulset.yaml). This statefuleset does:

* Pull routing service image from registry
* Use the configMap for the config.xml file created above as a volume in the current directory (the directory where the app is running)
* Set all environment variables used by the config.xml. Note that the LAN\_DOMAIN variable needs to be set to the same value with publisher and subscriber domainId above, e.g. (6)

---

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: rs-tcp

spec:

selector:

matchLabels:

app: routing-service *# has to match .spec.template.metadata.labels*

serviceName: "routing-service"

replicas: 1

template:

metadata:

labels:

app: routing-service *# has to match .spec.selector.matchLabels*

spec:

containers:

- name: routing-service

image: docker.io/pnminh/dds-routingservice

ports:

- containerPort: 7400

volumeMounts:

- name: config-volume

mountPath: /app/config.xml

subPath: config.xml

env:

- name: CFG\_NAME

value: "RS-TCP"

- name: ARGS

value: "-verbosity 5"

- name: PUBLIC\_ADDR *# value for dds.transport.TCPv4.tcp1.public\_address*

value: "10.15.74.53:7400"

- name: SERVER\_PORT *# has to match containerPort. value for dds.transport.TCPv4.tcp1.server\_bind\_port*

value: "7400"

- name: WAN\_DOMAIN *# value for WAN domain ID*

value: "100"

- name: LAN\_DOMAIN *# value for LAN domain ID*

value: "6"

volumes:

- name: config-volume

configMap:

name: routingservice-inbound

Apply the resource:

|  |
| --- |
| $ oc apply -f routingservice\_statefulset.yaml  statefulset.apps/rs-tcp created  $ oc get statefulsets  NAME READY AGE  rs-tcp 1/1 3m16s  $ oc get pod/rs-tcp-0  NAME READY STATUS RESTARTS AGE  rs-tcp-0 1/1 Running 0 4m27s |

Make sure the routing service can see all the participants. Here HelloWorld example is recognized

|  |
| --- |
| $ oc logs pod/rs-tcp-0 --tail=5 [/routing\_services/RS-TCP/domain\_routes/TCPWAN\_UDPLAN|STREAM\_DISCOVERED|/sessions/Session|/routes/Forward@Example HelloWorld|ENABLE]  [/routing\_services/RS-TCP/domain\_routes/TCPWAN\_UDPLAN/sessions/Session/routes/Backward@Example HelloWorld/inputs/Input1|ENABLE] stream=Example HelloWorld [/routing\_services/RS-TCP/domain\_routes/TCPWAN\_UDPLAN/sessions/Session/routes/Backward@Example HelloWorld/outputs/Output1|ENABLE] stream=Example HelloWorld [/routing\_services/RS-TCP/domain\_routes/TCPWAN\_UDPLAN/sessions/Session/routes/Backward@Example HelloWorld/inputs/Input1] subscription matched: current\_count\_change=1, current\_count=1 [/routing\_services/RS-TCP/domain\_routes/TCPWAN\_UDPLAN/sessions/Session/routes/Backward@Exa |

Create a service for the routing service with external IP so it can talk to the external routing service .In this scenario we create a ClusterIP service (routingservice\_clusterIP\_service.yaml).---

apiVersion: v1

kind: Service

metadata:

name: rs-tcp

spec:

type: ClusterIP

selector:

statefulset.kubernetes.io/pod-name: rs-tcp-0

ports:

- protocol: TCP

port: 7400

targetPort: 7400

externalIPs:

- 10.15.74.53

NodePort service can also be used (routingservice\_nodePort\_service.yaml)

---

apiVersion: v1

kind: Service

metadata:

name: rs-tcp-nodeport

spec:

type: NodePort

selector:

statefulset.kubernetes.io/pod-name: rs-tcp-0

ports:

- protocol: TCP

port: 7400

nodePort: 30008

Apply the resource (clusterIP service in this scenario):

|  |
| --- |
| $ oc apply -f routingservice\_clusterIP\_service.yaml  service/rs-tcp configured $ oc get svc rs-tcp  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE rs-tcp ClusterIP 172.30.136.10 10.15.74.53 7400/TCP 5m5s |

Note: Testing NodPort type does not require any extra configurations, but setting cluster externalIP requires the setup for egress for the namespace where the routing service lives:

|  |
| --- |
| $ oc patch hostsubnet ocp4-worker0 --type=merge -p '{"egressIPs": ["10.15.74.52","10.15.74.53"]}' hostsubnet.network.openshift.io/ocp4-worker0 patched  # make sure the IP is not used by other namespace $ oc patch netnamespace pub-sub --type merge -p '{"egressIPs": ["10.15.74.53"]}' netnamespace.network.openshift.io/pub-sub patched |

1. Routing service on dev node

First create a config file, rs-extern.xml. 3 environment variables need to be provided:

* INITIAL\_PEERS: tcp protocol with the OCP’s routing service public IP and port
* WAN\_DOMAIN: The same wan domainId with the OCP’s routing service
* LAN\_DOMAIN: The local domainId, will need to match the one used by local publisher/subscriber

<?xml version="1.0" ?>

<dds xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://community.rti.com/schema/6.0.1/6.0.0/rti\_routin

g\_service.xsd">

<routing\_service name="RS-TCP-EXTERN">

<domain\_route name="TCPWAN\_UDPLAN">

<participant name="1">

<domain\_id>$(WAN\_DOMAIN)</domain\_id>

<participant\_qos>

<discovery>

<initial\_peers>

<element>$(INITIAL\_PEERS)</element>

</initial\_peers>

</discovery>

<transport\_builtin>

<mask>MASK\_NONE</mask>

</transport\_builtin>

<property>

<value>

<element>

<name>dds.transport.load\_plugins</name>

<value>dds.transport.TCPv4.tcp1</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.library</name>

<value>nddstransporttcp</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.create\_function</name>

<value>NDDS\_Transport\_TCPv4\_create</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.parent.classid</name>

<value>NDDS\_TRANSPORT\_CLASSID\_TCPV4\_WAN</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.server\_bind\_port</name>

<value>0</value>

</element>

<element>

<name>dds.transport.TCPv4.tcp1.disable\_nagle</name>

<value>1</value>

</element>

</value>

</property>

</participant\_qos>

</participant>

<participant name="2">

<domain\_id>$(LAN\_DOMAIN)</domain\_id>

</participant>

<session name="Session">

<auto\_topic\_route name="AllForward">

<publish\_with\_original\_info>true</publish\_with\_original\_info>

<input participant="1">

<creation\_mode>IMMEDIATE</creation\_mode>

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

</input>

<output>

<creation\_mode>IMMEDIATE</creation\_mode>

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

</output>

</auto\_topic\_route>

<auto\_topic\_route name="AllBackward">

<publish\_with\_original\_info>true</publish\_with\_original\_info>

<input participant="2">

<creation\_mode>IMMEDIATE</creation\_mode>

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

</input>

<output>

<creation\_mode>IMMEDIATE</creation\_mode>

<deny\_topic\_name\_filter>rti/\*</deny\_topic\_name\_filter>

</output>

</auto\_topic\_route>

</session>

</domain\_route>

</routing\_service>

</dds>

Create a script file to add environment variables, rs\_extern\_envs.sh, note that the local domainID is set to 11.

|  |
| --- |
| #! /bin/bash export INITIAL\_PEERS=tcpv4\_wan://10.15.74.53:7400 export WAN\_DOMAIN=100 export LAN\_DOMAIN=11 |

Run the script to install the variables

|  |
| --- |
| $ chmod +x rs\_extern\_envs.sh $ source rs\_extern\_envs.sh |

Run the external routing service and the publisher/subscriber:

|  |
| --- |
| $ rtiroutingservice -cfgFile ./rs-extern.xml -cfgName RS-TCP-EXTERN -verbosity 5 $ HelloWorld\_publisher 11 |

3. Troubleshooting:

In case we cannot see the traffic go through, one way to debug is to use tcpdump:

|  |
| --- |
| $ tcpdump -i any host 10.15.74.53 |