SCinet DTN-as-a-Service Developments
Se-young Yu, Jim Chen, Ezra Kissel, Eric Pouyoul, Xi Yang
November 13th, 2020 • XNet
The project is intended to deliver Data Transfer Node software and hardware platform as a prototype service to support the SC SCinet community.

Multi-year XNet experiment, beginning in 2017
- In recent years deployed with Kubernetes, NVMe-oF, and 400G LAN/WAN
- 2019 INDIS paper
- Efforts to continue SCinet DTNaaS for SC20 and beyond
- Part 2 of this talk will describe details on history and broader work

After SC19, a tentative plan to provide DTNaaS as a SCinet capability was formed
- SC20 architecture began the year considering support of DTNs/compute in strawman
- COVID pandemic changed the equation

---

Gathering resources for a remote DTNaaS testing environment

- DTNs on ESnet testbed, Starlight, IU, MAX, CERN/SURFnet
- Make use of dynamic path provisioning: SENSE and OESS (also some static vlans)
- Some logistical challenges impeded progress early on
• Develop additional tooling to manage local and distributed containers
  • New prototype provisioning framework with minimal setup overhead
  • API, CLI for service allocation and management
• Understand the performance of different available container networks on hosts
  • Single networking namespace impractical if hosting multiple services
• Investigate use of RDMA/RoCE support for container-based DTNs
• Continue NVMe-oF evaluation
  • Explore NVMe over RDMA (RoCEv2) in addition to TCP
• Thin layer for Docker and/or Kubernetes management
  • Idea is to simplify deployment and explicitly control container interface specifics
  • Many K8s CNIs are beginning to fully support these capabilities as well
• WebUI and RESTful API for endpoint control, exposes Docker Engine API

Portainer\(^1\) as an endpoint manager

1 [http://portainer.io](http://portainer.io)
DTNaaS controller and agent

- Stateful service that interacts with Portainer API to allocate and control containers
- Uses profiles to manage control and data port ranges
- Support RDMA/Infiniband support via SR-IOV virtual function profiles
- Volumes, devices, limits, and capabilities

- DTNaaS Agent runs on each endpoint and exposes system details
  - Interfaces, CPUs, memory, block devices, NUMA mappings, etc.

```
lbnnl-rdma
volumes: { /nvme/data, mode: rw }
devices: [ rdma_cm, uverbs ]
limits: [ memlock:-1 ]
data_nets: [ star222 ]
ctrl_port_range: [ 12000,12019 ]
data_port_range: [50100, 50199]
```

Service request using profile
A number of container network attachments are supported and available

- Host namespace, bridge, macvlan, ipvlan, sr-iov, etc.
- Which works best for high-performance networking in each deployment?
- Relatively easy to achieve matching performance with well-understood tuning params
  - E.g., container execution mapped to appropriate NUMA nodes for cores and memory

**GridFTP forking server, single core affinity, -p1 client parallelism**

**Utilization In**

100G links, GridFTP, forking server, 5 clients with -p4 parallelism
DTNaaS Client and CLI

- Admin interfaces for DevOps management
- Client library and CLI could be easily adapted for user access

```
dtncli> cd active
```

```
4: STARTED
8: STARTED
11: STOPPED
```

```
dtncli> ls
```

```
uid: 136939c0-880f-41d7-9146-b872b64d83c7
user: admin
state: STARTED
```

```
dtncli> cd services
```

```
dtncli> ls
```

```
surf-dtn-ppc
```

```
dtncli> ls
```

```
mgmt_net
```

```
data_net
```

```
data_net_name: net3989
```

```
data_ipv4: 192.168.4.179
```

```
data_vfmac
```

```
container_user: dtnaas
```

```
ctrl_port: 30001
```

```
ctrl_host: 192.91.245.27
```

```
docker_kwargs
```

```
image: dtnaas/gct:latest
```

```
profile: surf
```

```
classifiers:
```

```
container_id: 38530bb62066e4704249454e80f865618ea2d596120fccc6ae4bc71e98c083d04
```

View active sessions

```
dtncli> transfer starlight-dtn:/data/IT surf-dtn-ppc:/data/IT gridftp
```

```
Found suitable existing session 8
Starting transfer for starlight-dtn -> surf-dtn-ppc using transfer type gridftp
```

```
dtncli> show transfers
```

```
1: (starlight-dtn -> surf-dtn-ppc [gridftp])
```

```
dtncli> show transfers log 1
```

```
Dest: sshftp://dtnaas@192.91.245.27:30001//data/IT
```

```
Connecting to sshftp://dtnaas@165.124.33.182:30000//data/It ...
```

```
3967287296 bytes 756.70 MB/sec avg 756.70 MB/sec inst
```

```
dtncli>
```

```
dtncli>
```

```
```

```
dtncli>
```

```
dtncli>
```

```
dtncli>
```

```
```

```
dtncli>
```

```
```

```
dtncli>
```

```
dtncli>
```

```
```

```
dtncli>
```

```
```

```
dtncli>
```

```
```

```
dtncli>
```

```
```

```
dtncli>
```

```
```

```
dtncli>
```

```
```

```
dtncli>
```

```
```

Manage transfers
### DTNaaS Client and CLI

Create new sessions across DTN endpoints using profiles

```bash
dtncli> session create nersc-tbn-7,starlight-dtn image dtmaas/ofed:latest profile star-rdma
Initialized new session with id "12"
dtncli> ls
4: STARTED [{'nersc-tbn-6', 'nersc-tbn-7'}, dtmaas/ofed)
8: STARTED [{'starlight-dtn'}, 'surf-dtn-ppc', dtmaas/gct:latest]
12: STARTED [{'nersc-tbn-7', 'starlight-dtn'}, dtmaas/ofed:latest]
dtncli> cd 12
dtncli> ls
uidiv: 6e6f7984-8cfd-44b2-ac06-0774ed8af7cd
user: admin
state: INITIALZED
allocations
services
request
```

### Supports multiple transfer/test types

<table>
<thead>
<tr>
<th>CMD</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dtncli&gt; ls</td>
<td>List current sessions and their status</td>
</tr>
<tr>
<td>4: STARTED</td>
<td>Session with id 4: nersc-tbn-6 and nersc-tbn-7</td>
</tr>
<tr>
<td>8: STARTED</td>
<td>Session with id 8: starlight-dtn</td>
</tr>
<tr>
<td>12: STARTED</td>
<td>Session with id 12: nersc-tbn-7 and starlight-dtn</td>
</tr>
</tbody>
</table>

```bash
dtncli> ls
4: STARTED  [{'nersc-tbn-6', 'nersc-tbn-7'}, dtmaas/ofed]
8: STARTED  [{'starlight-dtn'}, 'surf-dtn-ppc', dtmaas/gct:latest]
12: STARTED [{'nersc-tbn-7', 'starlight-dtn'}, dtmaas/ofed:latest]
dtncli> session start 12
Starting session "12"
dtncli>
dtncli> ls
4: STARTED  [{'nersc-tbn-6', 'nersc-tbn-7'}, dtmaas/ofed]
8: STARTED  [{'starlight-dtn'}, 'surf-dtn-ppc', dtmaas/gct:latest]
12: STARTED [{'nersc-tbn-7', 'starlight-dtn'}, dtmaas/ofed:latest]
dtncli> transfer nersc-tbn-7:/data/1f starlight-dtn:/data/1f_2 rdma
Found suitable existing session 12
Starting transfer for nersc-tbn-7 -> starlight-dtn using transfer type rdma
```

```bash
dtncli> show transfers
1: (starlight-dtn -> surf-dtn-ppc [gridftp])
2: (nersc-tbn-7 -> starlight-dtn [rdma])
```

```bash
dtncli> show transfers log 2
15 | port=18515 | ib_port=1 | tx_depth=16 | sl=0 | duplex=0 | cna=1 |
```

Created SLAB buffer with SIZE: 2147483648 PARTITIONS: 4

Metadata exchange complete
```bash
(0.0-2.0 sec) 16.64 GB  66.57 GB/s
(2.0-4.0 sec) 18.25 GB  73.01 GB/s
```

```bash
dtncli> show transfers log 2
(0.0-2.0 sec) 16.64 GB  66.57 GB/s
(2.0-4.0 sec) 18.25 GB  73.01 GB/s
(4.0-6.0 sec) 18.25 GB  73.01 GB/s
(6.0-8.0 sec) 18.25 GB  73.01 GB/s
(8.0-10.0 sec) 18.25 GB  73.01 GB/s
```

```bash
dtncli> show transfers log 2 src
(2.0-4.0 sec) 18.25 GB  73.01 GB/s
(4.0-6.0 sec) 18.25 GB  73.01 GB/s
(6.0-8.0 sec) 18.25 GB  73.01 GB/s
(8.0-10.0 sec) 18.25 GB  73.01 GB/s
(10.0-12.0 sec) 18.25 GB  73.01 GB/s
```

```
```