Integration between Rucio and SENSE

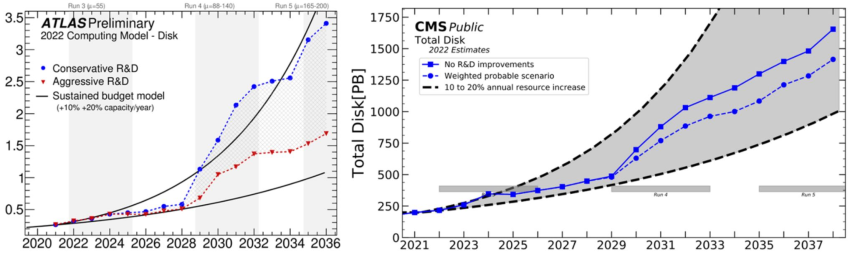
Frank Würthwein, Jonathan Guiang, Aashay Arora, **Diego Davila**, John Graham, Dima Mishin, Thomas Hutton, Igor Sfiligoi, Harvey Newman, Justas Balcas, Preeti Bhat, Tom Lehman, Xi Yang, Chin Guok, Oliver Gutsche, Phil Demar, Marcos Schwarz

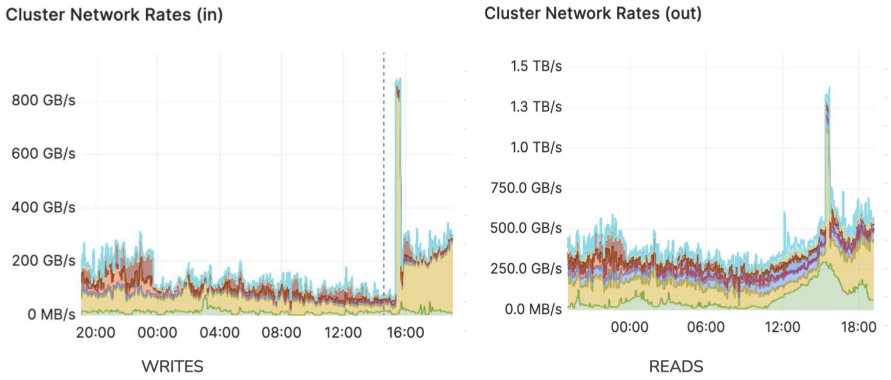
7th Rucio Workshop - October 2nd, 2024

Motivation

1. **High-Luminosity LHC**
2. Network is a finite resource
3. We hear a lot about **Storage requirements**, but not too much about **Network requirements,** even though we make a very intense use of it
4. It's time for us to be better Network users

ATLAS and CMS storage predictions





EOS data rates last 24h

Borrowed from Hugo's talk on Monday: https://indico.cern.ch/event/1343110/contributions/6105510/attachments/2937935/5160765/Copy%20of%20Tape%20and%20Disk%20evolution%20for%20the%20exabyte%20era.pdf

Previously in “SENSE” …

SENSE can:

- Orchestrate network services
- Negotiate bandwidth allocations
- Create **guaranteed bandwidth-allocated paths**

Not all transfers are equally important

What is a ***guaranteed bandwidth-allocated*** path?

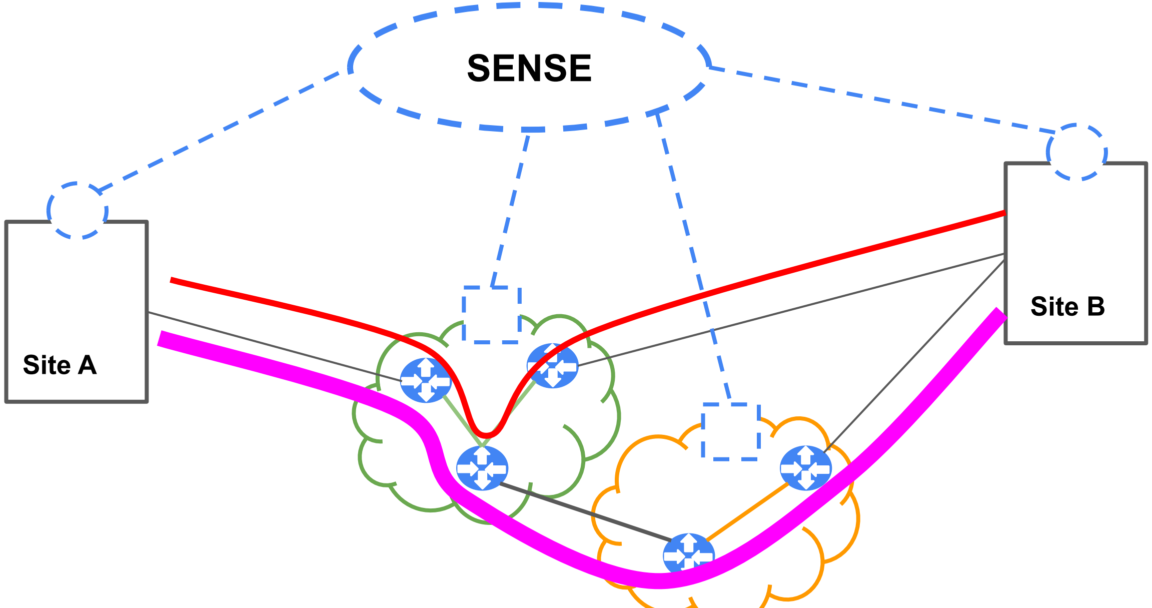
For network people

A mix of:

- BGB rules
- Layer 2 paths
- Quality of Service (QoS) rules

Between 2 subnets

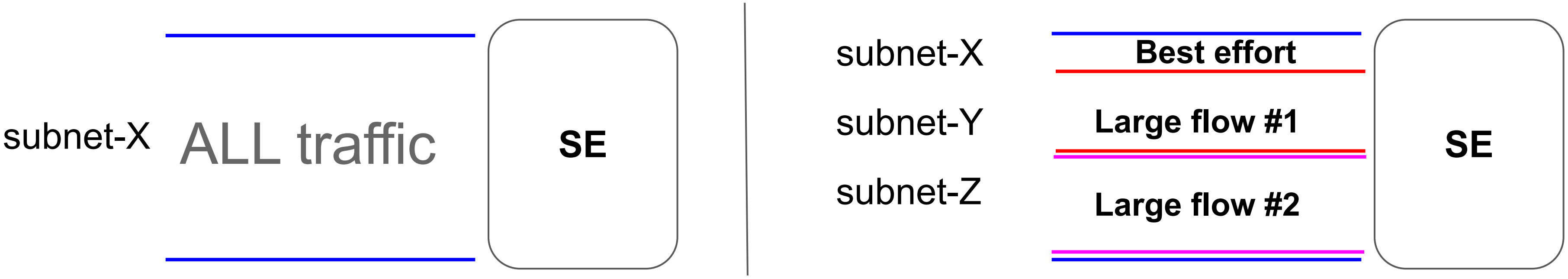
For simple mortals



Guaranteed Bandwidth-allocated (pink) vs Regular Best Effort path (red)

Subnets? Multiple endpoints, what?

SENSE builds these special paths based on subnets. Having **multiple “special paths”** on a given site, requires **multiple subnets**



subnet-X

ALL traffic

SE

subnet-X

subnet-Y

subnet-Z

Best effort

Large flow #1

Large flow #2

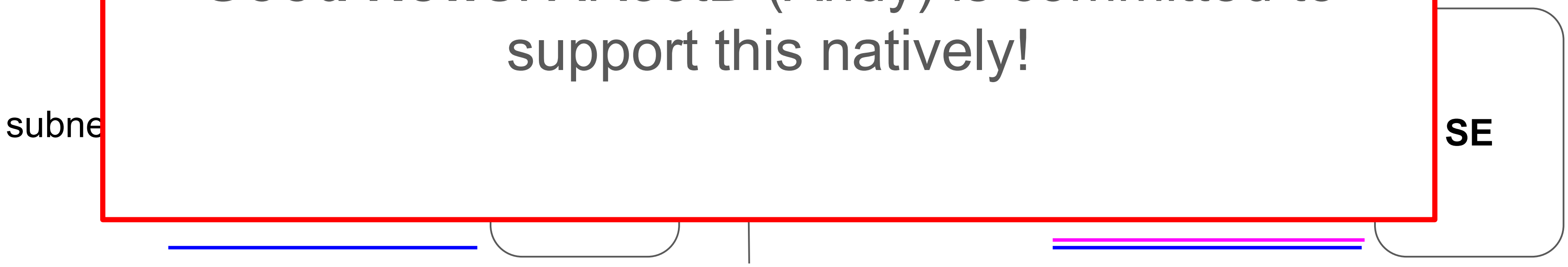
SE

We made the above work using a bunch of configurations and “Network Namespaces” magic. No need for extra hardware. Full presentation on this topic here:
https://indico.cern.ch/event/1386888/contributions/6104043/attachments/2927262/5139082/Network%20Isolation%20for%20multi-IP%20exposure%20in%20XRootD.pdf

Subnets? Multiple endpoints, what?

SE ecial

pa



Good News: XRootD (Andy) is committed to support this natively!

We made the above work using a bunch of configurations and “Network Namespaces” magic. No need for extra hardware. Full presentation on this topic here:
https://indico.cern.ch/event/1386888/contributions/6104043/attachments/2927262/5139082/Network%20Isolation%20for%20multi-IP%20exposure%20in%20XRootD.pdf

How can we use SENSE?

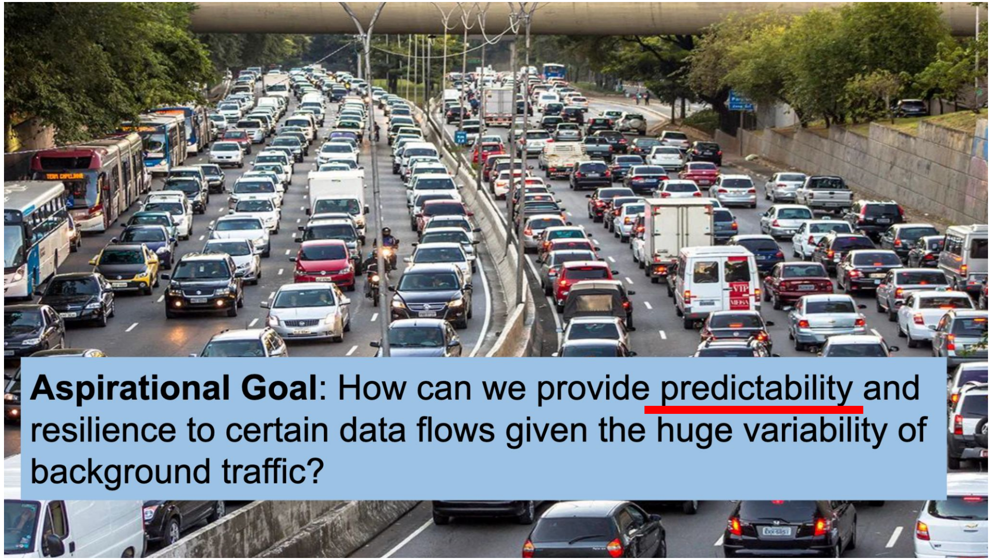
That’s where Rucio comes into the picture

Rucio orchestrates data movement (via TPCs) for our experiments. It knows how much data we need to move, where it has to go and its priority

...for Rucio to be the application that talks to SENSE

Objective

Enable Rucio to use SENSE to create **guaranteed bandwidth-allocated** paths for its more **important and large data flows**



Aspirational Goal: How can we provide predictability and resilience to certain data flows given the huge variability of background traffic?

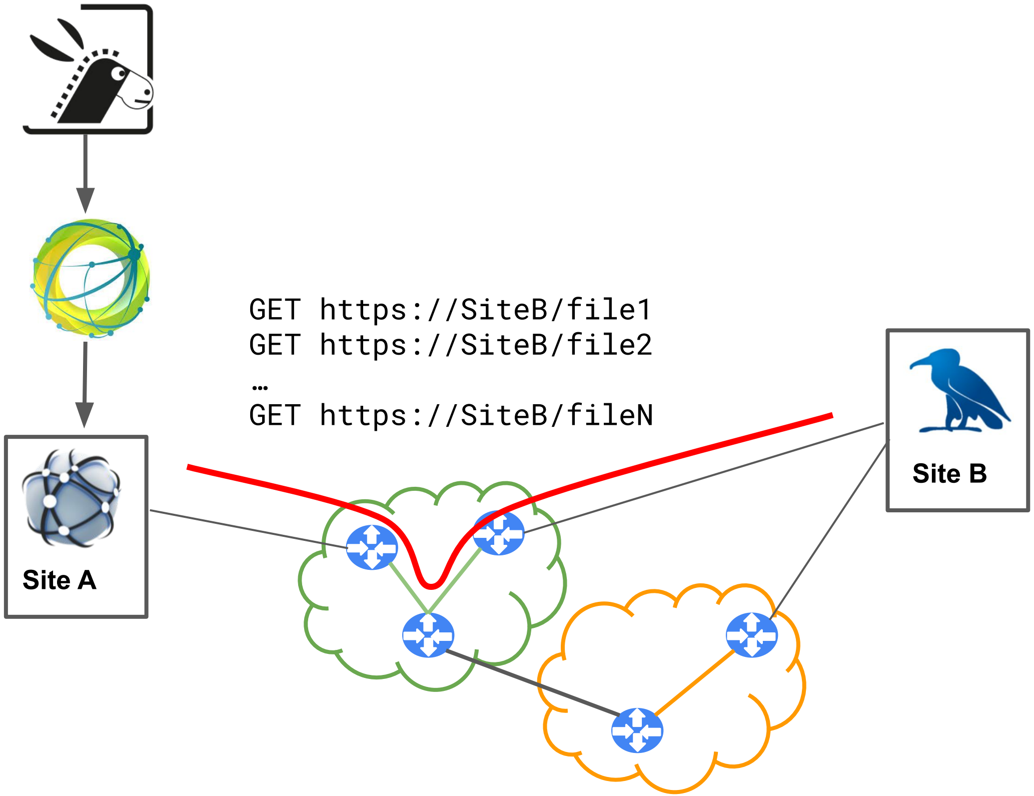
Borrowed from Inder’s talk:
https://indico.cern.ch/event/1343110/sessions/557886/attachments/2938714/5162279/SENSE%20Keynote%20Rucio%202024%20Monga.pdf

boring

A “~~normal~~” TPC transfer looks like

SOURCE	DEST
https://SiteB/file1	https://SiteA/file1
https://SiteB/file2	https://SiteA/file2
…	
https://SiteB/fileN	https://SiteA/fileN

```
COPY https://SiteB/file1
COPY https://SiteB/file2
…
COPY https://SiteB/fileN
```



```
GET https://SiteB/file1
GET https://SiteB/file2
…
GET https://SiteB/fileN
```

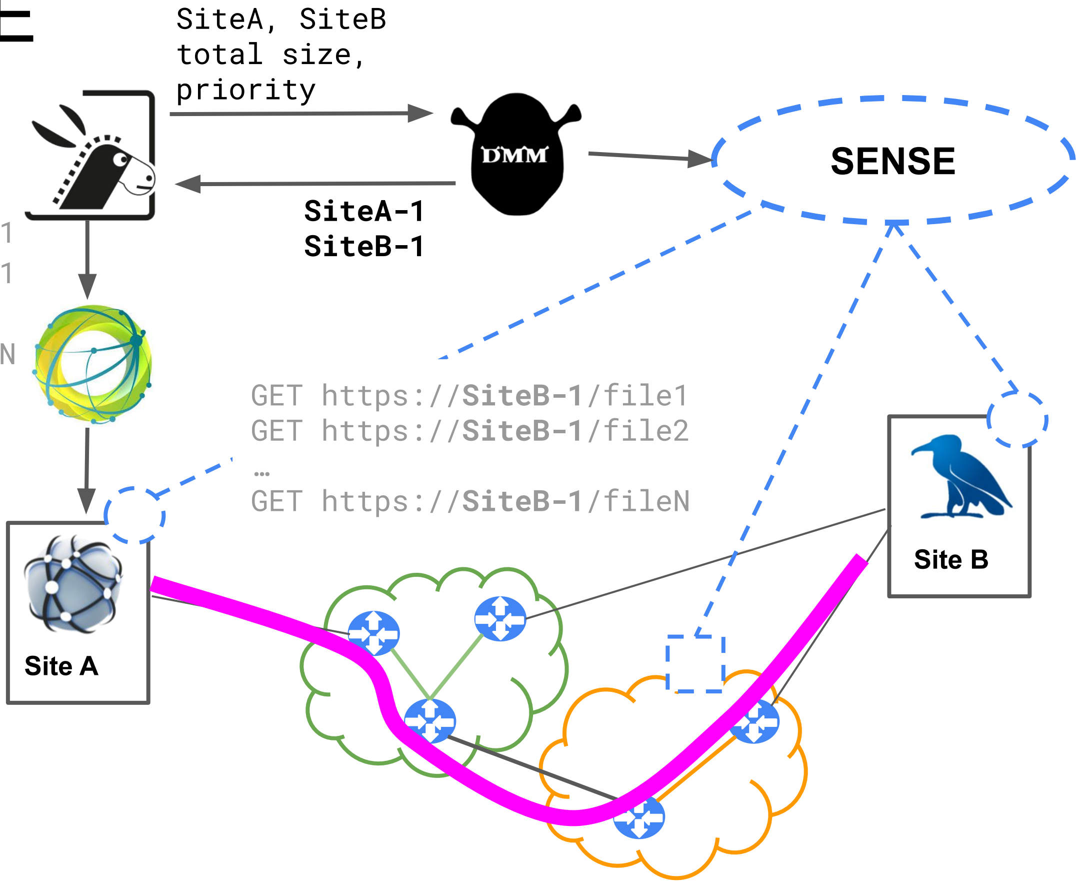
Changes to Rucio are minimal

…actually a ~40 lines patch in:
`/lib/rucio/transfertool/fts3.py`

And now with SENSE

SOURCE	DEST
https://**SiteB-1**/file1	https://**SiteA-1**/file1
https://**SiteB-1**/file1	https://**SiteA-1**/file1
…	
https://**SiteB-1**/fileN	https://**SiteA-1**/fileN

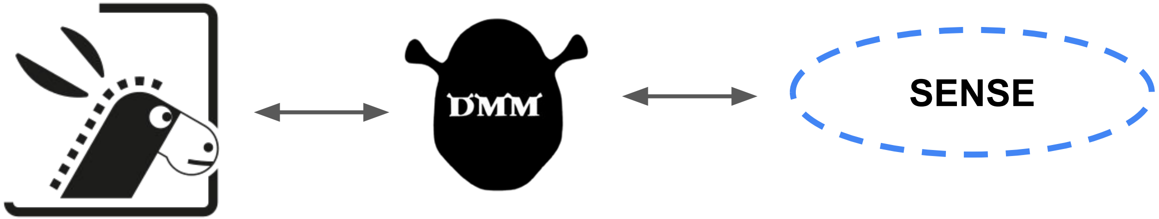
```
COPY https://SiteB-1/file1
COPY https://SiteB-1/file2
…
COPY https://SiteB-1/fileN
```



```
GET https://SiteB-1/file1
GET https://SiteB-1/file2
…
GET https://SiteB-1/fileN
```

Data Movement Manager (DMM)

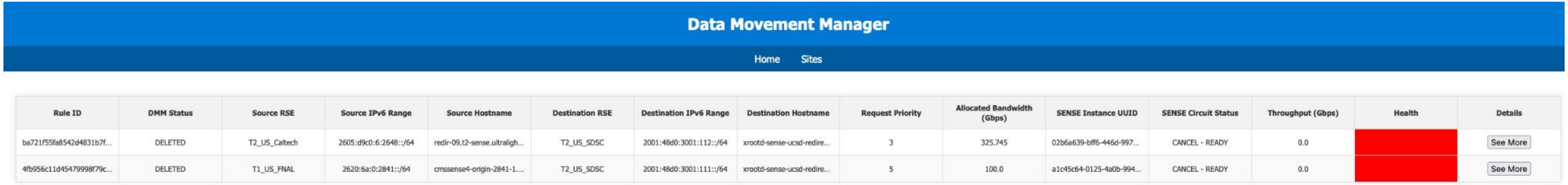
- Homemade SW
- Interface between Rucio and SENSE
- Knows about the different endpoints available on each site
- Calculates bandwidth request based on relative priorities
- Request network services to SENSE based on Rucio’s priorities
- Monitors the usage of the requested paths



SENSE

Aashay Arora, main developer of DMM

DMM Dashboard



Rule ID	DMM Status	Source RSE	Source IPv6 Range	Source Hostname	Destination RSE	Destination IPv6 Range	Destination Hostname	Request Priority	Allocated Bandwidth (Gbps)	SENSE Instance UUID	SENSE Circuit Status	Throughput (Gbps)	Health	Details
ba721f55fa8542d4831b7f...	DELETED	T2_US_Caltech	2605:d9c0:6:2648::/64	redir-09.t2-sense.ultraligh...	T2_US_SDSC	2001:48d0:3001:112::/64	xrootd-sense-ucsd-redire...	3	325.745	02b6a639-bff6-446d-997...	CANCEL - READY	0.0	🟥	See More
4fb956c11d45479998f79c...	DELETED	T1_US_FNAL	2620:6a:0:2841::/64	cmssense4-origin-2841-1....	T2_US_SDSC	2001:48d0:3001:111::/64	xrootd-sense-ucsd-redire...	5	100.0	a1c45c64-0125-4a0b-994...	CANCEL - READY	0.0	🟥	See More

Rule ID

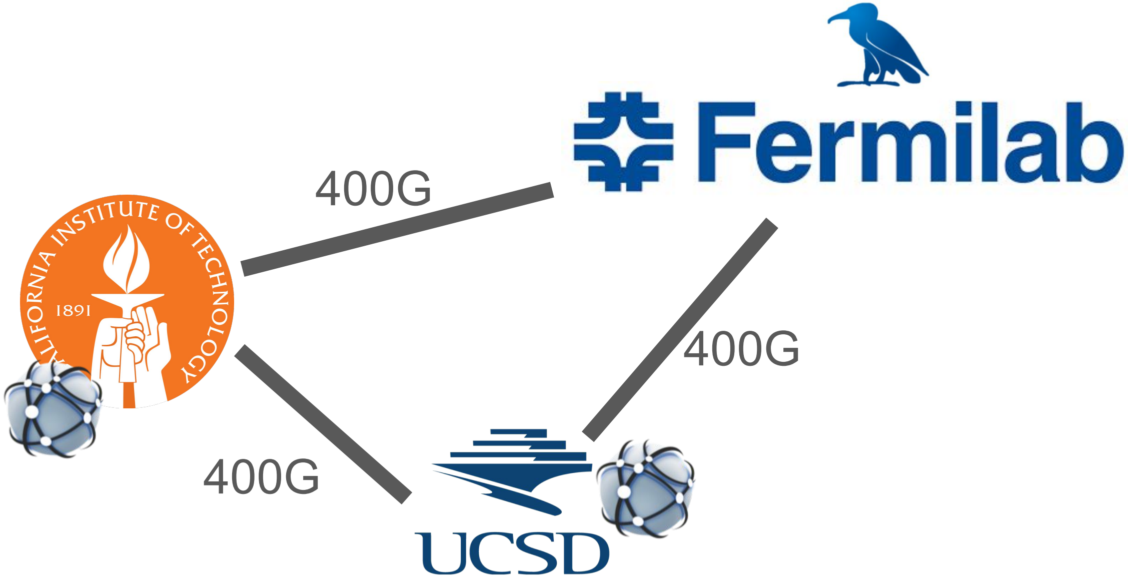
Source/Dest IP ranges

Priority

BW allocated

BW used

Current Status



Our own patched Rucio

fts3-pilot@ CERN

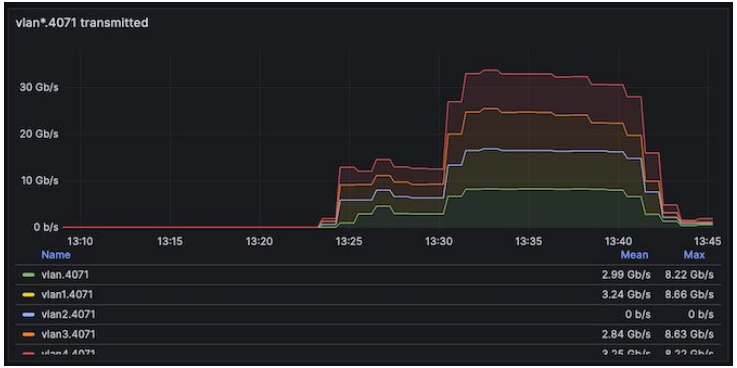
Highly interconnected testbed: FNAL, Caltech and UCSD

Q. Why “highly interconnected”? because this is R&D for HL-LHC

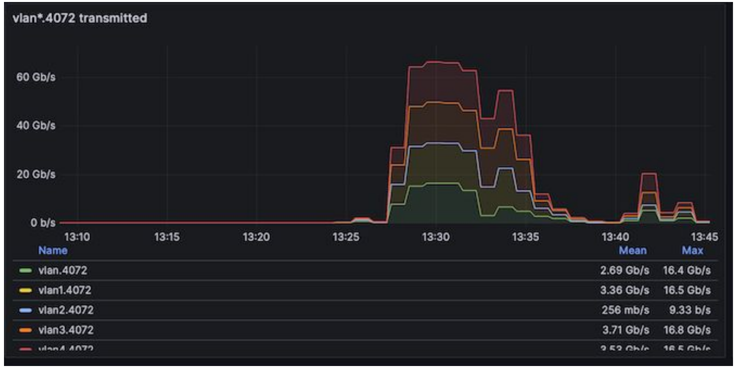
Achieved milestones

M1. Multi SENSE-managed data transfers between 2 sites

Two independent data flows travel, between two test-SEs, with different bandwidth allocations based on their priority



Name	Mean	Max
vlan.4071	2.99 Gb/s	8.22 Gb/s
vlan1.4071	3.24 Gb/s	8.66 Gb/s
vlan2.4071	0 b/s	0 b/s
vlan3.4071	2.84 Gb/s	8.63 Gb/s
vlan4.4071	3.25 Gb/s	8.22 Gb/s



Name	Mean	Max
vlan.4072	2.69 Gb/s	16.4 Gb/s
vlan1.4072	3.36 Gb/s	16.5 Gb/s
vlan2.4072	256 mb/s	9.33 b/s
vlan3.4072	3.71 Gb/s	16.8 Gb/s
vlan4.4072	3.53 Gb/s	16.5 Gb/s

Two data flows traveling isolated with their own allocated bandwidth (33/66 Gbps) between a pair of sites

Achieved milestones (cont’d)

M2. Multi SENSE-managed data transfers between 2 different pairs of sites.

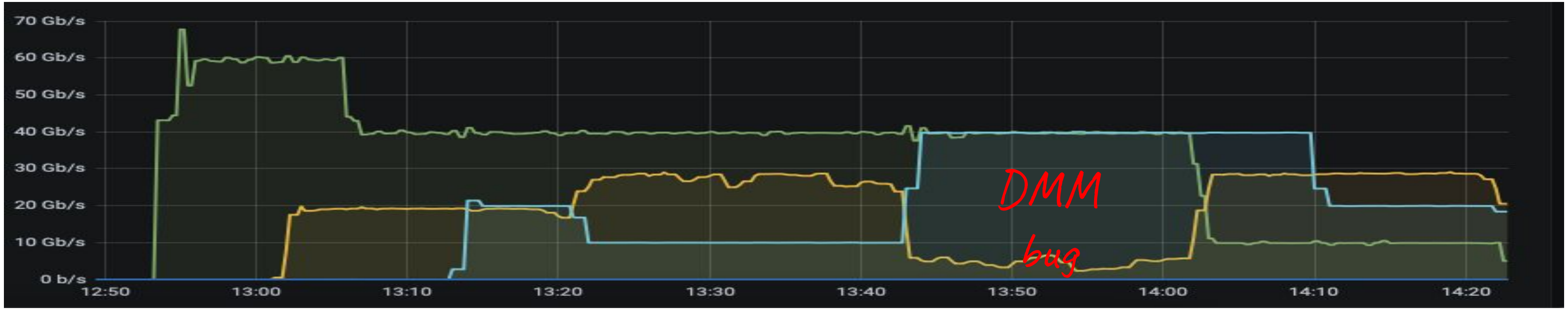
Using three sites A,B and C, two different data flows are created: A =>B & C =>B

Basically adding a third site in the mix

Achieved milestones (cont'd)

M3. Bandwidth allocation adjustment over ongoing data transfers

Two different data flows are created between 2 pairs of sites and their bandwidth allocation is changed on-the-fly by updating the Rucio's rule priority of one of these data flows.



Two Rucio data flows (blue and yellow) + background traffic (green) sharing 80 Gbps of Network capacity at Caltech. The bandwidth shares are modified on-the-fly.

What next?

1. Adding more Sites into our testbed
    a. Working with UNL, Purdue and Vanderbilt
    b. Ongoing deployments of 400 Gbps capable nodes into CERN and MGHPCC
2. Testing with prod-infrastructure
    a. Caltech (partially done)
    b. UCSD (in progress)
3. Exploring options for places without network control
    a. Playing with FRR(*) in FABRIC (See details in Justas's talk)
4. Add ATLAS sites into our testbed

(*) FRR free and open source Internet routing protocol suite for Linux and Unix platforms
https://frrouting.org/

Thanks!
Questions?

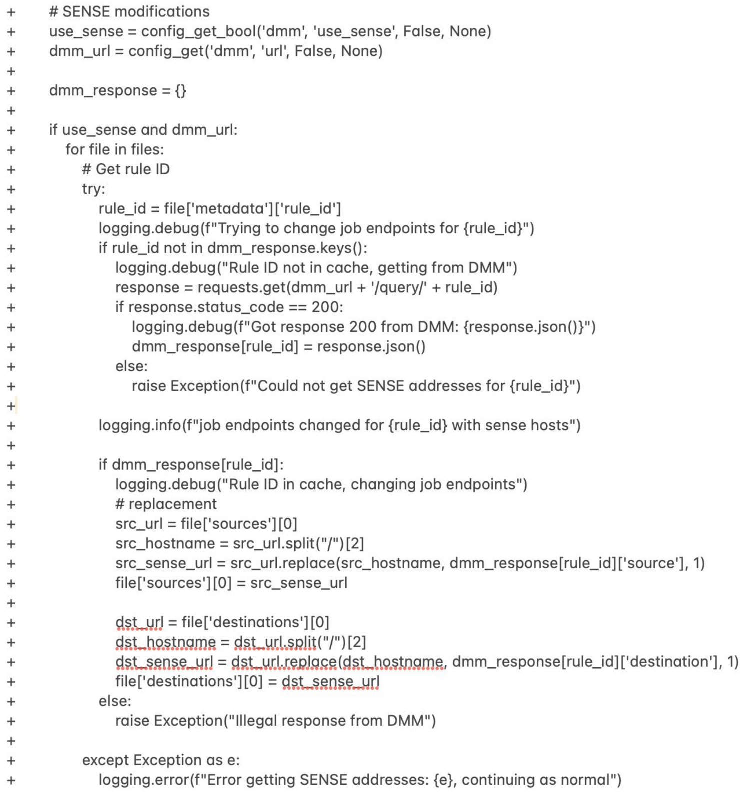
ESnet iris hep Fermilab CALIFORNIA INSTITUTE OF TECHNOLOGY 1891 UCSD CMS Compact Muon Solenoid

ACKNOWLEDGMENTS

This ongoing work is partially supported by the US National Science Foundation (NSF) Grants OAC-1836650, MPS-1148698, and PHY-1624356. In addition, the development of SENSE is supported by the US Department of Energy (DOE) Grants DE-SC0015527, DE-SC0015528, DE-SC0016585, and FP-00002494. Finally, this work would not be possible without the significant contributions of collaborators at ESNet, Caltech, FNAL and SDSC.

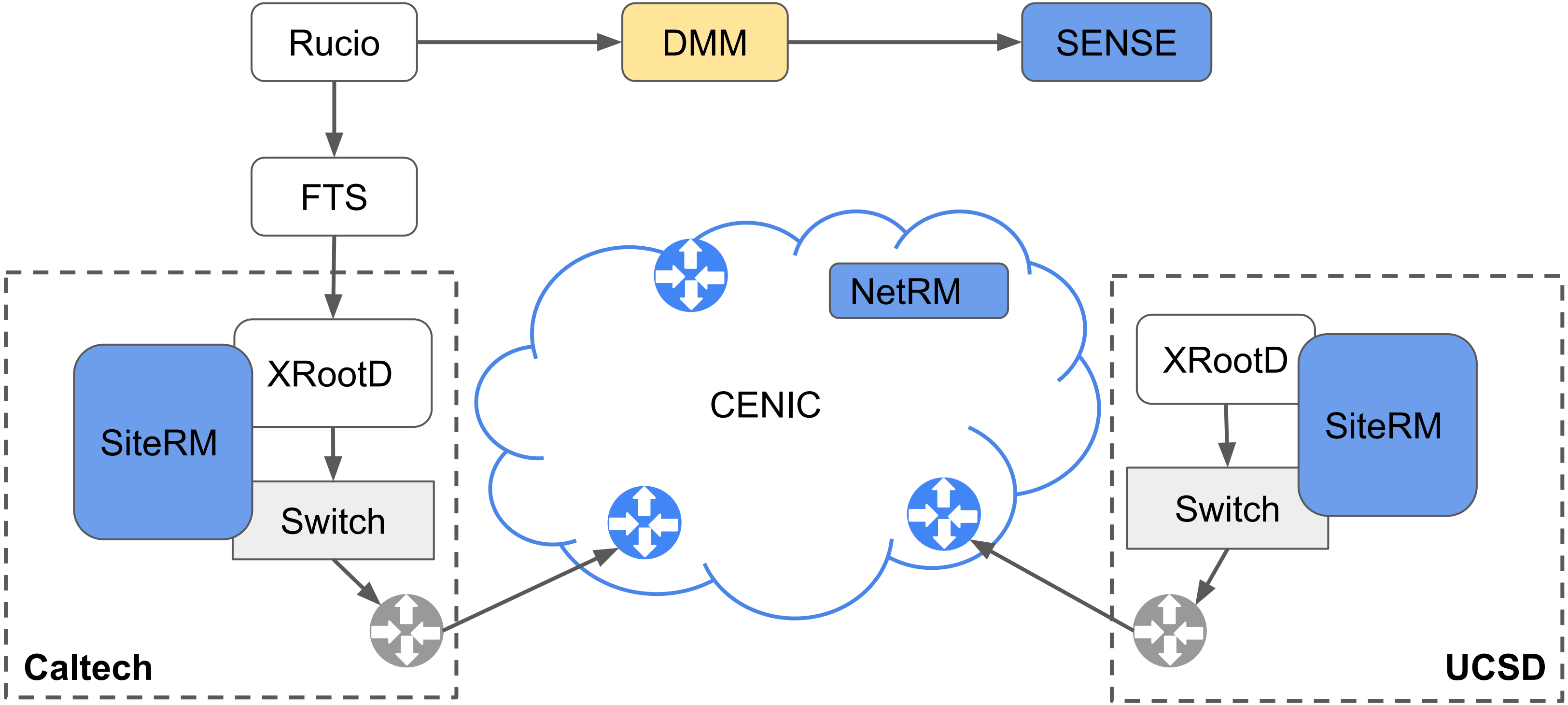
Background slides

Rucio patch: `/lib/rucio/transfertool/fts3.py`



```
+    # SENSE modifications
+    use_sense = config_get_bool('dmm', 'use_sense', False, None)
+    dmm_url = config_get('dmm', 'url', False, None)
+
+    dmm_response = {}
+
+    if use_sense and dmm_url:
+        for file in files:
+            # Get rule ID
+            try:
+                rule_id = file['metadata']['rule_id']
+                logging.debug(f"Trying to change job endpoints for {rule_id}")
+                if rule_id not in dmm_response.keys():
+                    logging.debug("Rule ID not in cache, getting from DMM")
+                    response = requests.get(dmm_url + '/query/' + rule_id)
+                    if response.status_code == 200:
+                        logging.debug(f"Got response 200 from DMM: {response.json()}")
+                        dmm_response[rule_id] = response.json()
+                    else:
+                        raise Exception(f"Could not get SENSE addresses for {rule_id}")
+
+                logging.info(f"job endpoints changed for {rule_id} with sense hosts")
+
+                if dmm_response[rule_id]:
+                    logging.debug("Rule ID in cache, changing job endpoints")
+                    # replacement
+                    src_url = file['sources'][0]
+                    src_hostname = src_url.split("/")[2]
+                    src_sense_url = src_url.replace(src_hostname, dmm_response[rule_id]['source'], 1)
+                    file['sources'][0] = src_sense_url
+
+                    dst_url = file['destinations'][0]
+                    dst_hostname = dst_url.split("/")[2]
+                    dst_sense_url = dst_url.replace(dst_hostname, dmm_response[rule_id]['destination'], 1)
+                    file['destinations'][0] = dst_sense_url
+                else:
+                    raise Exception("Illegal response from DMM")
+
+            except Exception as e:
+                logging.error(f"Error getting SENSE addresses: {e}, continuing as normal")
```

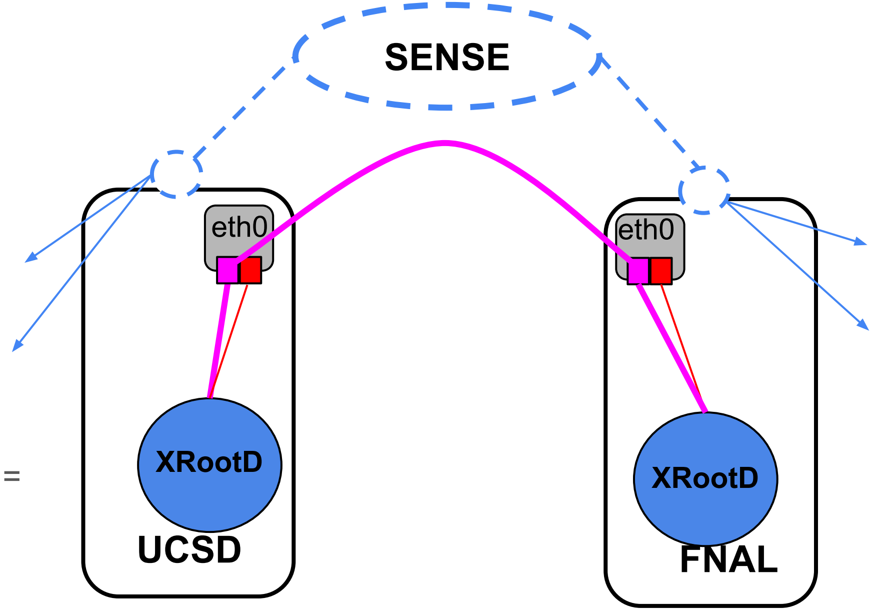
This is how Rucio + DMM + SENSE looks like



Solution #1

Use SiteRM to Insert routing rules on both sides of the “special path”

```
==== Routing Table ====
TO   FNAL-pink USE pink

FROM FNAL-pink USE pink

DEFAULT use red
========================
```

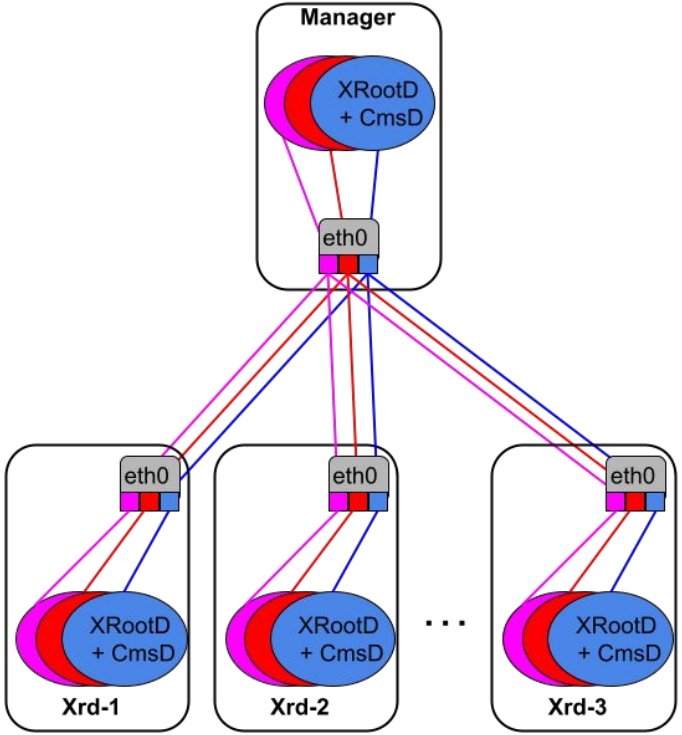


```
==== Routing Table ====
TO   UCSD-pink USE pink

FROM UCSD-pink USE pink

DEFAULT use red
========================
```

Solution #2

Use Network Namespaces to isolate multiple XRootD/CmsD instances, each of them attached to a different subnet

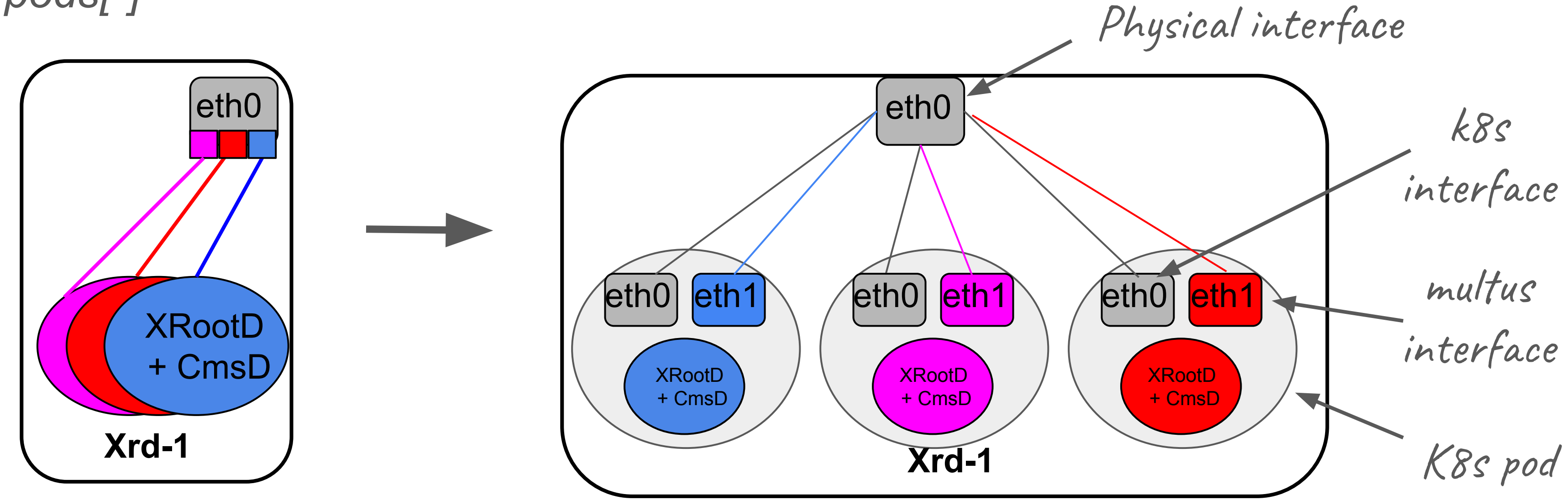
Each instance only sees 1 IP and its own (very simple) Routing Table



Each color globe represents an XRootD/CmsD instance in a separated network namespace

Solution #3

Similar to #2 but using Kubernetes and **Multus**: *a container network interface (CNI) plugin for Kubernetes that enables attaching multiple network interfaces to pods[*]*



[*] Multus: https://github.com/k8snetworkplumbingwg/multus-cni