EPJ Web of Conferences **245**, 07022 (2020) https://doi.org/10.1051/epjconf/202024507022
CHEP 2019

The NOTED software tool-set improves efficient network utilization for Rucio data transfers via FTS

Coralie Busse-Grawitz[1,*], *Edoardo* Martelli[2,*], *Mario* Lassnig[3], *Andrea* Manzi[2], *Oliver* Keeble[2], and *Tony* Cass[2]

[1]ETH (Zurich, Switzerland) – email: bcoralie@ethz.ch
[2]CERN (Geneva, Switzerland), IT department – email: firstname.lastname@cern.ch
[3]CERN (Geneva, Switzerland), EP-ADP group – email: Mario.Lassnig@cern.ch

Abstract. We present NOTED, a project to better exploit WAN bandwidth for Rucio and FTS data transfers. We describe the developed software tool-box and a successful demonstration of automated load-balancing in the production network. The first component is the Transfer Broker: It identifies large data transfers for which network reconfiguration is both possible and beneficial, and translates the transfer information into parameters that can be used by network controllers. It will make this information available via a public interface. The second component is a demonstration of a Network Controller. It periodically reads the parameters provided by the Transfer Broker, and so decides which actions to apply to improve the network utilisation for a given transfer.

1 Introduction

The original idea behind the NOTED (Network-Optimized Transfer of Experimental Data) project was developed in the context of the LHCOPN/LHCONE (LHC Optical Private Network/LHC Open Network Environment) community [1]: The goal is to better exploit the bandwidth in WAN networks by load-balancing traffic instead of congesting the best path.

To achieve this, we tackle three key challenges: (i) exposing high-level data-transfer intentions at the network level, (ii) leveraging the resulting information for effective network actions, and (iii) tackling challenges (i) and (ii) in an inter-domain setting.

Exposing high-level data-transfer intentions at the network level: In order to effectively speed up large file transfers, identifying congestion points and then dynamically reconfiguring the network falls short of the full optimisation potential. This is because the congestion itself does not contain information about how long it will last: In particular, a network controller that relies on congestion measurements alone would risk to apply changes when the origin of the issue (e.g., a big file transfer) is reaching its completion, making the optimisation useless or even counterproductive.

To address this problem, the NOTED project has developed a Transfer Broker (§4) that interfaces with file transfer services to collect and publish high-level information about data transfers—information such as when a large data transfer is going to happen, its volume or duration, and its origin and destination in network terms. The Transfer Broker compiles this information into both human- and machine-friendly formats.

*main authors

Translating enriched transfer information to network actions: The second challenge is to take concrete network optimisation actions, i.e. to make dynamic router configuration changes based on transfer information.

NOTED tackles this challenge with a Network Controller (§5) that, in our prototype implementation, adjusts BGP weights based on information provided by the Transfer Broker. Other network optimisation actions are foreseen, as described in §7.

Coordination in an inter-domain network: The LHCONE [2] and the LHCOPN [3] networks consist of several independent domains. Coordinated actions across multiple domains are thus required to configure devices along the entire path.

NOTED plans to address this problem by publishing the information compiled by the Transfer Broker to a public website. Multiple Network Controllers can fetch transfer information from this website and take appropriate actions in the network domain(s) they manage. We demonstrate the potential of this approach in §6.

2 Data management and file transfer services

The ability of the NOTED project to optimise data transfers relies critically on the fact that these are managed, not random. In principle, the NOTED Transfer Broker could interface with any data management or file transfer service, but, within the HEP (High Energy Physics) domain, Rucio and FTS (the File Transfer Service) are responsible for managing the bulk of all data transfers.

Rucio [4] is a framework to manage and distribute data produced by LHC experiments in order to make it available for processing and analysis at collaborating WLCG [5] (Worldwide LHC Computing Grid) sites. Rucio schedules bulk file transfers and uses FTS [6] to actually copy the files to the requested destinations.

FTS [6] manages the transfer of individual files between different endpoints. For this project, the most important aspect is its state machine. In particular, it identifies the state of a transfer from submission to completion/failure with the state `SUBMITTED`, i.e. files about to pass through the network. Notably, when a big transfer is requested, the number of files in the `SUBMITTED` state increases, since many files have to wait for actual transfer [7].

3 NOTED overview

As discussed in §1, two components are required in order to perform effective network reconfigurations to improve data transfer efficiency: a Transfer Broker and a Network Controller.

The Transfer Broker obtains, aggregates and re-formats transfer and network information and makes it available in a human- and machine-friendly format.

The Network Controller uses the Transfer Broker's information to reconfigure network devices under its control, such that it optimises traffic routing in its domain to improve data transfer efficiency.
Figure 1 illustrates the overall NOTED setup .

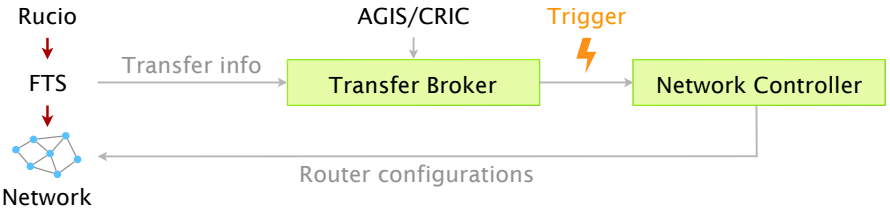

Figure 1. NOTED Overview. (Red arrows: *x uses y* relations, grey arrows: information flow)

4 The Transfer Broker

The Transfer Broker publishes information about high-volume data transfers with the necessary details—start and stop times, source and destination—for a Network Controller to take appropriate actions.

The following paragraphs present the Transfer Broker's inputs, its outputs and how it generates the outputs based on the inputs. Finally, we shortly describe its implementation.

Inputs

The Transfer Broker takes the number of files in `SUBMITTED` state for each source-destination pair as an input from FTS. It also queries the AGIS (ATLAS Grid Information System) [8] database to obtain the IP prefixes of a given site[1].

Outputs

The Transfer Broker outputs start and stop triggers, which indicate that a big file transfer has started or stopped for a certain source-destination pair.

The triggers are in JSON format; they include the source and the destination, their IP prefixes, and the action, i.e., whether a big transfer just started or stopped. Figure 2 shows the format with an example.

```
{
    "source": {
        "name": "<source_name>",
        "ip_prefixes": [
            "<ip_prefix_1>",
            ...
        ]
    }
    "destination": {
        "name": "<dest_name>",
        "ip_prefixes": [
            "<ip_prefix_1>",
            ...
        ]
    }
    "action": "start/stop"
}
```

```
{
    "source": {
        "name": "CERN-PROD",
        "ip_prefixes": [
            "1.1.1.1/24",
            ...
        ]
    }
    "destination": {
        "name": "SARA-MATRIX",
        "ip_prefixes": [
            "2.2.2.2/24",
            ...
        ]
    }
    "action": "start"
}
```

Figure 2. Format of the Transfer Broker output with an example on the right.

[1]Previously, CERN Twiki pages contained the mapping of site name to IP prefixes. This information was copied to AGIS, and will be integrated into CRIC [9]. CRIC is intended to be used by all experiments in the future.

Generating triggers
To obtain the triggers as above, the Transfer Broker needs the start and stop information for a given source-destination pair, as well as the source and destination including their IP prefixes.

Start and stop: As mentioned in §2, the number of files in the `SUBMITTED` state in FTS is an indicator for the size of the file transfer and the extent to which the transfer is limited by the network. The Transfer Broker thus approximates the start and stop times of a large transfer by comparing the FTS queue size with configurable start and stop thresholds.
Apart from filtering with a threshold, the Transfer Broker also maintains a transfer watchlist to avoid unnecessary triggers. For a given source-destination pair, a second start trigger is only sent if it has previously sent a stop trigger.

Source, destination and their IP prefixes: The Transfer Broker reads the `SUBMITTED` queues grouped by source and destination from FTS. However, FTS identifies sites by their storage elements' (SE) names. To obtain the sites' IP prefixes from AGIS, the Broker hence first translates the SE to the AGIS name.

AGIS contains the full IP prefix range associated with each site. We use these full IP prefixes for the following reason: Rucio and FTS identify sites by the name of the head-node of the storage systems, but these are not directly involved in the file transfer; they simply dispatch the file transfer operation to the storage elements hosting the data to be transferred. Hence we take all IP addresses into account a site's storage elements could have.

Implementation
The Transfer Broker consists of 1200 lines of code in Go [10], version `go1.12.7`. It queries a Grafana [11] monitoring instance of FTS to obtain the transfer information.

5 The Network Controller

The role of the Network Controller is to change the network configuration based on information from the Transfer Broker. The prototype is CERN-specific since the actions to be taken depend on the local network infrastructure. However, it can easily be used as a model for site-specific implementations elsewhere.

Below, we describe the load-balancing mechanism implemented by the Controller, how it automates the corresponding configuration changes, and its implementation details.

Load-balancing mechanism: In a previous proof-of-concept [12], traffic between CERN and NL-T1, the Dutch WLCG Tier-1, was load balanced across the LHCOPN and LHCONE connections by manually setting the CERN BGP metrics of these two paths to be equal. Since this was very effective, our prototype Network Controller automates this approach.

Automatic configuration changes: The Network Controller periodically reads the Transfer Broker's triggers. It then adapts the corresponding BGP metrics by using routing policies. The controller has two built-in safety mechanisms. First, if another party has locked the configuration, it retries obtaining write access several times with a timeout. Second, to avoid unintended side-effects, the controller only changes the routing policies if the pre-existing ones are as expected. If it finds unexpected policies, it reverts its changes on all routers.

Implementation
The Network Controller consists of 800 lines of code in Python 3.6.8 [13]. It makes use of the `PyEZ` library [14] to remotely configure Juniper routers.

6 Full chain demonstration test

We ran a test transfer on the production network (LHCONE/OPN) to demonstrate the full automated toolchain, from (i) the Transfer Broker to (ii) the Network Controller to (iii) the network, on 11 December 2019 [15]. The Rucio team scheduled two transfers of 20TB each, the first from CERN to NL-T1 and the second to DE-KIT, a German WLCG Tier-1. These transfers were not concurrent since the current implementation is only capable of initiating one load-balancing action at a time.

Transfer Broker
The Transfer Broker correctly published the start and stop triggers for the two transfers.

Network Controller
Unfortunately, due to an unexpected dependency of timeout policies on the router access port, the Network Controller failed to properly issue the load-balancing commands to the CERN LHCOPN/ONE routers during the transfer to NL-T1. However, after quickly changing the router access port, the controller successfully load-balanced the second transfer to DE-KIT.

Effect on the network
Figure 3 shows the traffic measurements at DE-KIT. The top and bottom graph show the traffic in LHCOPN and LHCONE, respectively. The highlighted green part shows the statistics when the load balancing was active, with the load spread over the two networks. The final part of the transfer is solely via LHCOPN. This is because the load balancing was removed at the point the FTS queue went below the stop threshold, not after the transfer was fully completed. This was a deliberate choice to highlight the ability of the controller to both end and initiate network load balancing.

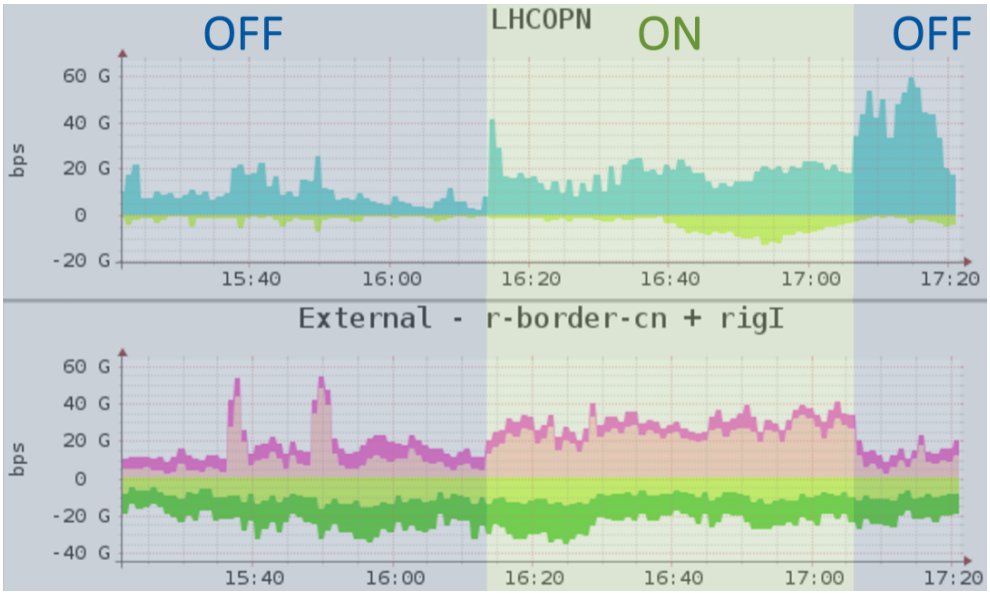

Figure 3. The load-balancing successfully spreads the 20TB transfer to DE-KIT over both LHCOPN and LHCONE. (blue/green: load-balancing off/on, respectively)

7 Future Work

Future plans for NOTED include extending the Transfer Broker and further developing the Network Controller, notably by using Segment Routing [16] and network status information.

The Transfer Broker will be extended to (i) make it a public service; (ii) reduce dependencies; and (iii) add more information to its output. After a satisfactory implementation, support for alternative file transfer services beside FTS will be added. Work will be needed to ensure the compatibility of the information provided by the Transfer Broker when there are multiple sources.

Public service: Since the Transfer Broker currently is a prototype, it is not yet attached to a public website. We aim to provide a public service that any Network Controller can use in the future.

Reducing dependencies: Both FTS' ActiveMQ [17] and the Grafana APIs can be used to gather information concerning ongoing and future data transfers. Grafana shows pre-processed, easy to interpret data, but its API introduces a further dependency. Hence we intend to let the Transfer Broker directly interface with ActiveMQ in the future.

Extending the transfer information: To further expose high-level data transfer intentions, we envisage a closer integration with Rucio, s.t. the Transfer Broker can publish more detailed information. In particular, this could include the precise size of transfers and predictions of the start and end time.

Fairness of the Network controller: actions taken locally by network controllers may impact other networks and their users. All the possible actions that will nbe implemented should be agreed with the affected network providers and users.] **Segment Routing** [16] is an emerging technology that allows traffic engineering operations by tagging the transmission packets. We will explore this as an alternative to BGP as a mechanism to implement network optimisations.

Network status information could be leveraged by the Network Controller. Currently, the Controller load-balances traffic without checking beforehand whether the alternative path is already congested. In the future, the Network Controller could take both triggers and information from Network Monitoring services into account, to improve the action-selection mechanism. Also, this feedback creates a full information loop, opening avenues for more advanced control system designs.

Data Transfer Nodes: use of tools specialized in data transfers is an alternative approach that could be considered as an alternative to the proposed ideas.

8 Conclusions

The NOTED project has demonstrated that it is possible to interpret and complement the information available in file transfer services like FTS, and to effectively use this understanding of the traffic characteristics to improve how the files are transferred over the network.

It has also demonstrated that it is possible to use such information to implement network optimizations in an automatic way to improve the bandwidth utilization.

Notably, we have delivered a ***Transfer Broker***, implemented in Go, that exposes high-level data-transfer intentions in network aware terms (§4); a proof-of-concept ***Network Controller***, implemented in Python, that uses this information to dynamically load-balance traffic over two paths by automatically adapting router configurations, (§5); and ***successfully demonstrated*** use of both on production wide-area networks (§6).

References

[1] E. Martelli, *Smart WAN for data intensive science* (LHCONE meeting at BNL (US), April 2017)
https://indico.cern.ch/event/581520

[2] *LHCONE: LHC Open Network Environment*
https://twiki.cern.ch/twiki/bin/view/LHCONE/WebHome

[3] *LHCOPN*
https://twiki.cern.ch/twiki/bin/view/LHCOPN/WebHomev

[4] M. Barisits, T. Beermann, F. Berghaus et al. *Rucio: Scientific Data Management.* Computing and Software for Big Science 3, article 11 (2019).
https://doi.org/10.1007/s41781-019-0026-3

[5] *WLCG: Worldwide LHC Computing Grid*
https://wlcg-public.web.cern.ch/

[6] FTS Development Team at CERN, "FTS" File Transfer Service.
https://fts.web.cern.ch/
http://fts3-docs.web.cern.ch/fts3-docs/

[7] *FTS3 Documentation: Job and file state machine*
http://fts3-docs.web.cern.ch/fts3-docs/docs/state_machine.html

[8] *ATLAS Grid Information System*
http://atlas-agis.cern.ch

[9] *CRIC: Computing Resource Information Catalog*
http://cms-cric.cern.ch/

[10] *Go*
https://golang.org/

[11] *Grafana Labs: The open observability platform*
https://grafana.com/

[12] E. Martelli, *CERN-NLT1 load balancing over LHCOPN and LHCONE* (LHCONE meeting at Fermilab (US), November 2018)
https://indico.cern.ch/event/725706

[13] *Python*
https://www.python.org/

[14] *Junos pyEZ Developer Guide*
https://www.juniper.net/documentation/en_US/junospyez/information-products/pathway-pages/junos-pyez-developer-guide.html

[15] C. Busse-Grawitz, *The NOTED project* (LHCONE workshop at CERN (CH), January 2020)
https://indico.cern.ch/event/828520

[16] *Segment Routing*
https://www.segment-routing.net/

[17] *ActiveMQ*
https://activemq.apache.org/