Virtual Circuits in PhEDEx, an update from the ANSE project

Lăpădătescu V[1] and Wildish T[2], on behalf of the ANSE project
[1] Caltech, USA
[2] Princeton University, USA

E-mail: awildish@princeton.edu

Abstract.
The ANSE project has been working with the CMS and ATLAS experiments to bring network awareness into their middleware stacks. For CMS, this means enabling control of virtual network circuits in PhEDEx, the CMS data-transfer management system. PhEDEx orchestrates the transfer of data around the CMS experiment to the tune of 1 PB per week spread over about 70 sites.

The goal of ANSE is to improve the overall working efficiency of the experiments, by enabling more deterministic time to completion for a designated set of data transfers, through the use of end-to-end dynamic virtual circuits with guaranteed bandwidth.

ANSE has enhanced PhEDEx, allowing it to create, use and destroy circuits according to it's own needs. PhEDEx can now decide if a circuit is worth creating based on its current workload and past transfer history, which allows circuits to be created only when they will be useful.

This paper reports on the progress made by ANSE in PhEDEx. We show how PhEDEx is now capable of using virtual circuits as a production-quality service, and describe how the mechanism it uses can be refactored for use in other software domains. We present first results of transfers between CMS sites using this mechanism, and report on the stability and performance of PhEDEx when using virtual circuits.

The ability to use dynamic virtual circuits for prioritised large-scale data transfers over shared global network infrastructures represents an important new capability and opens many possibilities. The experience we have gained with ANSE is being incorporated in an evolving picture of future LHC Computing Models, in which the network is considered as an explicit component.

Finally, we describe the remaining work to be done by ANSE in PhEDEx, and discuss future directions for continued development.

1. Introduction

PhEDEx[1] is the data-transfer management tool for CMS. It gathers requests for data-transfer from users and from automated components of the CMS computing system, schedules the transfers at each destination site, then transfers the data in a reliable and robust manner, and reports the results back to a central database. The core architecture has changed little since PhEDEx was first put into production in 2004, before the Worldwide LHC Computing Grid (WLCG[2]) was operating, and the underlying assumptions that were built into its behaviour then are still present today.

Chief among those assumptions was the expectation that the network would be over-subscribed and unreliable, so PhEDEx was designed to back off fast when errors happened

and to retry slowly. This strategy allows time for debugging problems without congesting the network with too many transfers that are likely to fail.

Since then, the network has in fact proven to be more reliable and performant than expected, and the hierarchical nature of transfers envisioned in the original CMS computing model[3] has been abandoned in favour of allowing transfers between any two sites that wish it[4].

As a result, CMS has begun to investigate new ways to interact with the network, with a view to making it an active component of the computing model in the future[5].The ANSE project[6] has been working closely with PhEDEx since 2013 to this end, integrating network-awareness into PhEDEx by enabling it to create, use, and destroy virtual network circuits to improve transfer performance. First results have already been reported, showing that PhEDEx can now transparently switch to using a network circuit when one is available[7]

2. Circuit-awareness and PhEDEx

PhEDEx consists of a set of software agents interacting with a central database. Circuit-awareness can be integrated into PhEDEx in a number of places, the two most important being in the *FileRouter* agent or the *FileDownload* agent.

2.1. Circuits in the FileRouter agent

The FileRouter agent is a 'central' agent, meaning that there is only one instance of it in the entire PhEDEx system. There are many central agents, each specialised for a unique function. The FileRouter agent decides which source to select for file-transfer to a given destination site, and builds a transfer-queue for each destination site. As such, it has a global overview of the entire transfer system, and could be augmented to adjust its choice of sources to maximise the potential use of virtual circuits. E.g. where two destination sites would compete for a given network link it could choose to not build a transfer queue for one destination until the other has completed its work, to reduce competition.

One disadvantage of modifying the FileRouter agent is that it is not synchronised with the actual transfer to the destination, i.e. the FileRouter builds a queue for the destination but does not determine when that queue is processed. This may invalidate any assumptions it makes about how much traffic will be on any given network link at any point in time, which means it may be wrong about the best way to optimise the traffic.

Another disadvantage is that modifications to the FileRouter are more complex than alternative solutions. Any bugs or problems could affect the entire system, precisely because it's a central agent, and the design and development of a prototype would take longer than was deemed practical.

For these reasons, the ANSE project decided not to implement circuit-control in the FileRouter agent.

2.2. Circuits in the FileDownload agent

The FileDownload agent is a 'site' agent, meaning that every site that wishes to download data via PhEDEx runs one or more copies of this agent. It processes the transfer queue provided by the FileRouter agent for that site, reporting the results back to the central database.

The FileRouter therefore has an accurate view of the site-local inbound network traffic. Implementing circuits here is relatively simple, and allows deployment and testing site-by-site, with consequently minimised risks in the event of problems.

The only significant disadvantage to implementing circuit-control in the FileDownload agent is the lack of global oversight, with the consequence that two sites may end up competing for virtual circuits on a given link at the same time. Despite this, the ease with which circuits could be implemented in this agent was the winning factor, and the first prototype used a modified FileDownload agent to achieve its results (see figure 1).

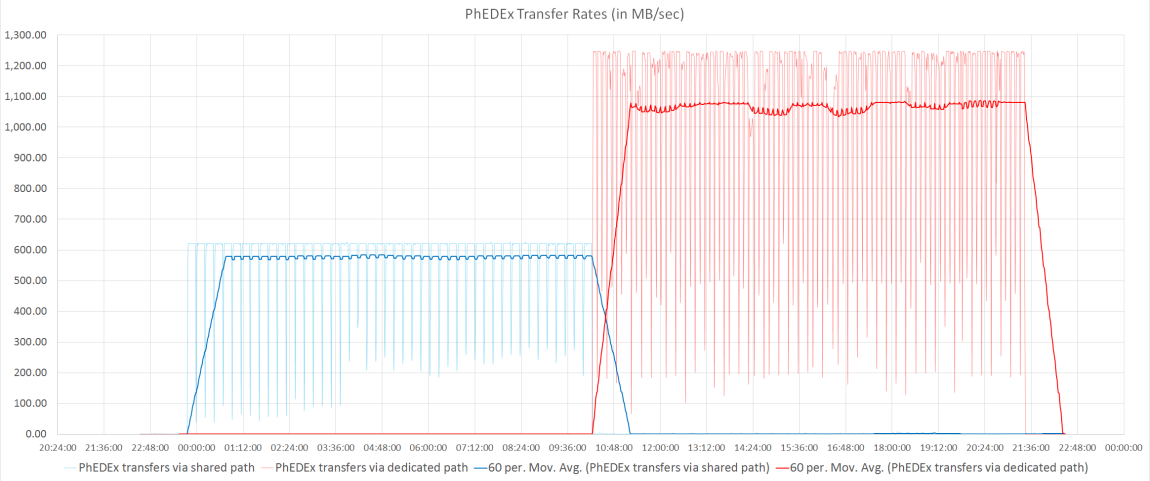

Figure 1. View of PhEDEx transfers on a 10 Gbps link between Geneva and Amsterdam. The transfer performance on a shared path (shown in blue) is impeded by competition on the link. When a circuit becomes available on a dedicated path, PhEDEx transparently switches to using that circuit (shown in red), thereby doubling the throughput.

2.3. Towards a Production-ready System

The original prototype used DYNES[8], the DYnamic NEtwork Services tool for circuit-creation. DYNES is, unfortunately, no longer supported, so in refactoring the code to build a production-ready system we needed another interface. The Network Service Interface (NSI[9]) is the natural successor. After some refactoring and redesign, we now have a production-ready system which can interface to either DYNES or NSI as a circuit-creation mechanism using a plugin. The addition of other mechanisms should be straightforward, depending on the details of their API.

3. Difficulties using NSI circuits

3.1. Circuit extent and capability

NSI circuits do not extend all the way to the storage nodes of a site, they stop at the border router. This leaves the classic 'last mile' problem of maintaining performance end-to-end in any given transfer.

NSI does not necessarily provide bandwidth guarantees. This is not a show-stopper in itself, but it does greatly limit the value of the circuit. Even a soft guarantee (e.g. time-averaged performance) would be acceptable[10], but the total lack of guarantee in some situations is a serious limitation which we hope the network providers will address in the near future.

NSI is also not widely adopted yet, and availability as a production service is limited. However, it is supported by a lot of large network providers, so availability is expected to grow over time.

3.2. Layer 2 vs. Layer 3 circuits

There is one major difference between the circuits provided by DYNES and the circuits provided by NSI that poses a serious problem for our use of circuits in PhEDEx. DYNES provides layer-3 circuits, i.e. circuits at the IP level. NSI only provides layer-2 circuits, at the ethernet level.

This is a problem because PhEDEx is high-level middleware, and knows only about source and destination hostnames and port numbers. It knows nothing about low-level details of the network such as ethernet addresses or anything to do with the topology of the network, it merely calls a transfer-layer tool (such as FTS[11] or FDT[12]) to handle the actual transfer. These

tools take their input in the form of a SURL (a 'Storage URL') which specifies the host, port, protocol and local pathname on that host to access the file. PhEDEx is able to switch between different layer-3, IP-based circuits by changing the hostname (or IP address) that it uses in the SURL, the rest happens transparently.

Because NSI doesn't provide layer-3 circuits, this mechanism is insufficient in itself, we need to create a layer-3 circuit on top of the layer-2 circuit. Unfortunately, this requires priviledged access to the sites' networking infrastructure, and sites are naturally reluctant to allow intrusive access of this nature.

The problem with creating a level-3 circuit on top of a level-2 circuit has essentially two components. First, creating a domain-wide level-3 circuit is not difficult, either technically or politically, indeed this is what DYNES provided in the past. In the absence of DYNES, or some agreed replacement for it, we need to create circuits triggered by the demands of the experiments middleware. This requires routing and topological information, something which PhEDEx has no knowledge of, and indeed PhEDEx should not know such things.

The second component of the problem is that domain-wide circuits may not be appropriate, in that they may accidentally allow traffic which should not be on the circuit to use it, thereby competing with the transfer we want to prioritise. During Run-2 of the LHC this is not likely to be such a problem, email and web traffic is hardly likely to compete with multi-terabyte data transfers. Even inter-experiment competition is not likely to be large because most places where CMS and ATLAS share sites are already well provisioned anyway (Tier-1s or larger Tier-2s). Circuits are more likely to be useful at the edges of the experiments' networks, and there they share fewer sites, so competition between experiments is likely to be lower. Within an experiment such as CMS, if we have multiple flows of data between two sites and we wish to prioritise one over the other, we can simply suspend the lower priority flow at the level of PhEDEx, so we don't need a circuit-level interface to manage that yet. There is still the possibility of competition between different data-management systems within CMS, e.g. between PhEDEx and AAA[13] (the CMS implementation of an xrootd[14] federation) or ASO[15] (the system for re-patriating user-files output from analysis jobs around the grid). To first order this is also expected to be small during Run-2, and we ignore it for now.

However, one of the goals of ANSE is not just to provide tools for CMS and ATLAS, but to provide tools for the broader science community. There are many experiments being built that will have significant networking needs and which could benefit from this work. Therefore this problem cannot be simply ignored, and needs to be addressed.

4. Circuits at sub-domain granularity

As mentioned, not all the traffic between two sites should necessarily go via a circuit, should one exist. Typically, sites have a cluster of data-servers which host data, and any or all of them may participate in a given bulk transfer between two sites.

To build circuits at granularity finer than the IP domain of the participating sites, it is important to know the exact set of data-servers that will serve or receive the data. PhEDEx itself may not have access to that information. PhEDEx constructs the SURL for the source and destination from its own knowledge, but the SURL typically points to a gateway-host that redirects the transfer to the actual hosts that will take part, exactly analogous to a web-site redirecting a URL. PhEDEx does not see this 'Transfer URL', or TURL, because it invokes the transfer tool (FTS, FDT etc) with the SURL, and it is this transfer tool that is redirected to the appropriate TURL.

Our proposed solution is illustrated in figures 2 and 3. In figure 2, the experiment-specific middleware is either PhEDEx or the 'External app', which communicates with the ResourceManager to request circuits. The ResourceManager is independent of the experiments, and can become part of the WLCG middleware. This component already exists, having been

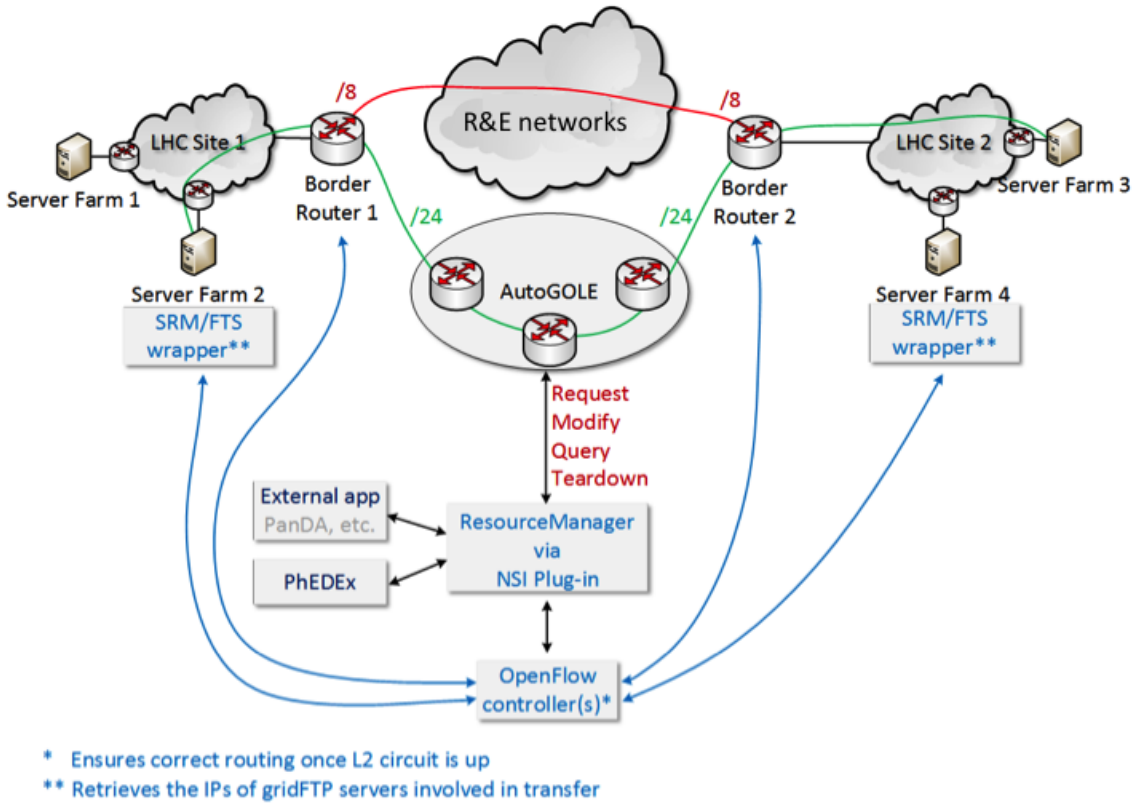

Figure 2. The middleware view of the proposed solution. The ResourceManager requests NSI circuits and informs OpenFlow controllers of their existence. The SRM/FTS wrapper at each site informs the OpenFlow controller of the transfers that are actually taking place. The OpenFlow controller dynamically updates the sites' routing to steer the appropriate traffic into the circuit.

refactored from the original prototype reported in [7]. The ResourceManager communicates with OpenFlow network controllers, telling them of the creation or destruction of NSI circuits and the list of files that are to be transfered over the circuit.

A wrapper-script runs on the data-servers at the sites, wrapping the calls to gridFTP so it can intercept them without having to modify gridFTP itself. The wrapper notifies the same OpenFlow network controllers that a transfer is about to start. The OpenFlow controllers then modify the routing table at the sites to divert the traffic onto a circuit, if one exists.

While non-trivial, this solution does not require privileged software to run on the sites data-servers, and cleanly separates the experiments from the network management layer. The use of a wrapper script to talk to the OpenFlow controllers means that the underlying transfer technology (gridFTP or whatever) does not have to be modified. It does require that OpenFlow controllers be installed at participating sites, but once there, they offer a single point of control for the circuit behaviour, making the system easier to understand and debug.

Figure 3 shows more detail of the workflow involved. PhEDEx (or some other experiment middleware) requests an NSI circuit, which is established between the sites border routers. Meanwhile, the wrapper script is constantly informing the OpenFlow controller of the set of files being transfered at any point in time. This is all the OpenFlow controller needs to know when to establish a level-3 circuit from the data-server to the border router and over the level-2

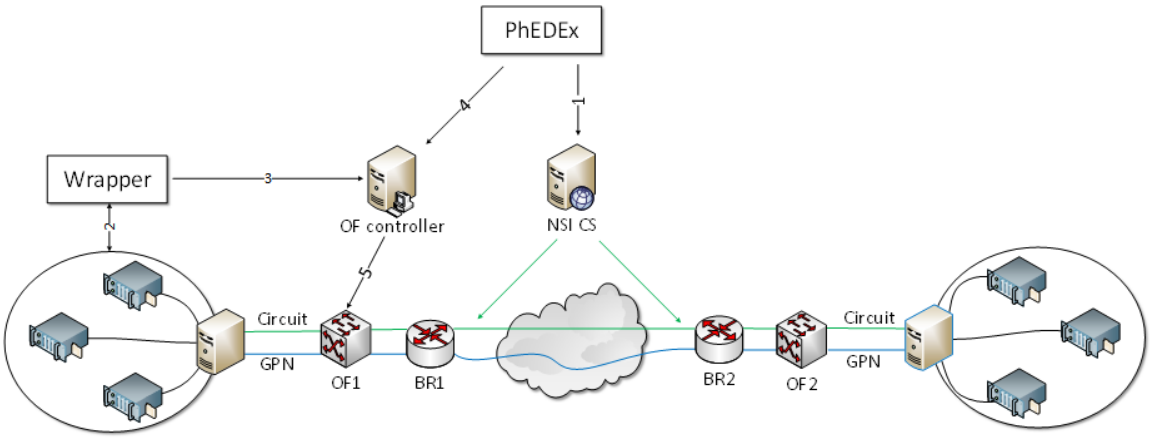

Figure 3. Workflow of the proposed solution. PhEDEx requests a circuit, meanwhile the wrapper script is constantly informing the OpenFlow controller of the files that are in transfer at any moment. Once PhEDEx has a circuit, it informs the OpenFlow controller of its existence, and the controller diverts the relevant traffic to use the circuit.

circuit to the destination.

5. Conclusions

PhEDEx has successfully managed large-scale data-transfers for CMS for over 10 years now, and continues to be a workhorse of the experiments' operations model. Despite this, the design decisions of a decade ago are no longer true, and are a handicap to future optimisation and enhancement of the way CMS transfers data. PhEDEx must evolve if it is to continue to satisfy the needs of CMS during Run-3 and beyond. The ANSE project has been working with PhEDEx for some time now, and has made major progress in bringing network awareness to it. A proof-of-concept prototype has been built which shows that PhEDEx can cleanly exploit level-3 circuits, switching to take advantage of them when they exist, switching back to using the general purpose network when they go away, with no degradation in transfer quality. From this prototype, a refactored, production-ready version has been built. This is not PhEDEx-specfic, nor even CMS-specific or HEP-specific, and can be used to bridge the gap between experiment middleware and a network that supports level-2 or level-3 circuits. To fully exploit level-2 circuits, more work must be done. We have identified a candidate solution which, we believe, is practical and implementable with reasonable effort.

6. Acknowledgements

This work is supported in part by grants from the DOE Offices of HEP and Advanced Scientific Computing (DE-SC0007346), the NSF (OCI-1246133), and Cisco Research Grants (Microgrant-2014-128271) to Caltech. We thank ANSE team members Julian Bunn, Samir Cury, Dorian Kcira, Iosif Legrand, Azher Mughal, Shan McKee, Harvey Newman, and Ramiro Voicu for their feedback and help.

Dear Reviewer, this section is still missing a few grant numbers. Would you please ask me to re-submit with an update, as I will have those numbers by the time you read this. Thank you, and I apologise for the inconvenience.

References

[1] Egeland R, Wildish T and Metson S 2008 Data transfer infrastructure for CMS data taking, *XII Advanced Computing and Analysis Techniques in Physics Research (Erice, Italy: Proceedings of Science)*

[2] Eck C *et al.* 2005 LHC Computing Grid Technical Design Report *CERN-LHCC-2005-024*
[3] Grandi C, Stickland D and Taylor L 2005 The CMS Computing Model *CERN-LHCC-2004-35/G-083, CMS note 2004-031*
[4] Wildish T 2015 Understanding the Tier-2 Traffic in CMS during Run-1 *submitted to CHEP 2015*
[5] Bonacorsi D and Wildish T 2013 Challenging CMS Computing with Network-Aware Systems *J.Phys.Conf.Ser. 513 (2014)*
[6] LHCONE Point-to-Point Service Workshop, December 2012 *http://indico.cern.ch/event/215393/session/1/contribution/8/mate*
[7] Lăpădătescu V and Wildish T 2014 Integrating Network-Awareness and Network-Management into PhEDEx *POS (ISGC 2014) 021*
[8] Dynes: DYnamic NEtwork System *https://www.terena.org/activities/e2e/ws3/slides/101129-dynes-Artur.pdf*
[9] OGF Network Service Interface *https://www.terena.org/activities/e2e/ws2/slides2/11_NSI_Eduard.pdf*
[10] Wildish T 2015 Bandwidth Sharing in LHCONE: An analysis of the problem *submitted to CHEP 2015*
[11] Alvarez Ayllon A, Kamil Simon M, Keeble O and Salichos M 2013 FTS3 - Robust, simplified and high-performance data movement service for WLCG *J.Phys.Conf.Ser. 513 (2014)*
[12] Fast Data Transfer (FDT) *http://fdt.cern.ch/*
[13] Bloom K 2013 CMS Use of a Data Federation *J.Phys.Conf.Ser. 513 (2014)*
[14] XROOTD, http://xrootd.slac.stanford.edu/
[15] Andreeva J, Balcas J, Ciangottini D, Hernandez J M, Karavakis E, Mascheroni M, Riahi H, Tanasijczuk A J, Vaandering E W and Wildish T 2015 Asynchronous Stage-Out of user-data in CMS *submitted to CHEP 2015*