09-12-2020

Data Transfer Node (DTN) Tests on the GÉANT Testbeds Service (GTS)

Grant Agreement No.:	856726
Work Package	WP6
Task Item:	Task 2
Document ID:	GN4-3-20-265f683
Authors:	Iacovos Ioannou (CYNET), Damir Regvart (CARNET), Joseph Hill (UvA/SURF), Maria Isabel Gandia (CSUC/RedIRIS), Ivana Golub (PSNC), Tim Chown (Jisc), Susanne Naegele Jackson (FAU/DFN)

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No. 856726 (GN4-3).

Abstract
Many research projects need to transfer large amounts of data and use specialised Data Transfer Nodes (DTNs) to optimise the efficiency of these transfers. In order to help NRENs to better support user communities who seek to move large volumes of data across their network, a DTN Focus Group was set up in the Network Technologies and Services Development work package (WP6) of the GN4-3 project, which has performed several activities around DTN deployment, the results of which are summarised in this document.

Table of Contents

Table of Figures

Table of Tables

Executive Summary

As the capacity of the GÉANT European and international network links is being increased, it is of key importance to European research communities who need to move large volumes of data between facilities for processing, archiving, visualisation or other purposes that this capacity can be optimally exploited for sharing of large-scale research and science data.

European National Research and Education Networks (NRENs) typically offer high-capacity backbones, provisioned to minimise contention and packet loss, but the backbone network is not the only factor to consider towards achieving the best performance. The architectures of the communicating end-site networks, their storage systems, their tuning, and the software used for transfers are also of key importance.

Some research communities' research projects have already developed the necessary knowledge and experience on the transfer of large-scale science data over wide area networks, and these typically follow the best practice principles of "Science DMZ" [SCIENCEDMZ1] (whether they knowingly followed the Science DMZ design pattern, or had already evolved it themselves over time). Such principles include the use of Data Transfer Nodes (DTN), i.e. dedicated file servers with specific high-end hardware components with fast read and write capability and specialised transfer tools, which are configured and tuned purposely for wide-area data transfer. But there is also a "long tail" of communities that are yet to gain this knowledge and experience.

To understand the current situation in the NRENs and to help them better support such communities, the Network Technologies and Services Development Work Package (WP6) of the GN4-3 project ran a survey among several European NRENs in late 2019, the results of which highlighted both technical and non-technical considerations for achieving effective end-to-end large scale data transfers. To help address the technical issues, the WP6 team has studied, compared and tested a wide range of tools, through tests run on the GÉANT Testbeds Service [GTS], using virtualised container environments to more easily set up DTN platforms.

Regarding the non-technical issues, dissemination of best practices, and end-user engagement were found to be significantly relevant. The team has created a public wiki [DTN-Wiki] containing general DTN information, related use cases, applied methodologies, hardware and software systems and tools and the results of the various tests run with the transfer tools (including examples of containerised transfer tools), as well as the survey results.

1 Introduction

Large research projects, such as those related to high energy physics, genomics or astronomy, increasingly need to transfer great amounts of data to complete calculations and obtain results in a relatively short period.

To do this, in the past the physical shipping of hard disks full of data was the fastest option. The high bandwidth pipes offered by research and education networks however now make it possible to achieve large-scale, high-performance transfers via R&E networks. Examples of this include the Worldwide Large Hadron Collider Computing Grid [WLCG], and the AENEAS project [AENEAS-D41], which worked towards the data transfer architecture for the new Square Kilometre Array [SKA] project, for which transfers are expected at rates of at least 100Gbps.

Such transfers over wide area networks, even at currently more common rates around 10Gbps, require optimal end-to-end throughput through a combination of transfer tools for high-speed big file/multiple file data movement, specialised servers with the appropriate tuning hosted at sites with optimised local network architectures and high capacity network pipes. The complexity of data sources from multiple and distributed teams and complex science workflows, as well as the scaling and spanning of resources between multiple sites to store or process data, are a true challenge for those designing the supporting hardware and software architectures where data transfers are required.

To optimise large-scale data transfers between different sites, the best practice "Science DMZ" principles [SCIENCEDMZ1] [SCIENCEDMZ2], documented by ESnet in 2013, recommend the deployment of bespoke local network architectures and the use of dedicated, well-tuned, data transfer nodes (DTNs) to enhance performance.

The core principles of Science DMZ consist of three elements:

1. A "friction-free" network path, using highly capable network devices running wire speed with deep queues, and security policy enforcement specific to the science workflow.
2. Dedicated high-performance DTNs, with hardware, operating system and libraries all optimised for transfers, and including optimised data transfer tools such as Globus and GridFTP.
3. Network performance measurement, typically using perfSONAR.

The Science DMZ model therefore aims to minimise or ideally eliminate packet loss, while still applying security policies efficiently for that traffic, typically by use of stateless Access Control Lists (ACLs). This approach means that science data is handled differently from the day-to-day "business" traffic on the site (web, email, and other traffic), which would for example traverse the main campus firewall, where deep packet inspection techniques are used.

An example of a Science DMZ deployment is given below in Figure 1.1.

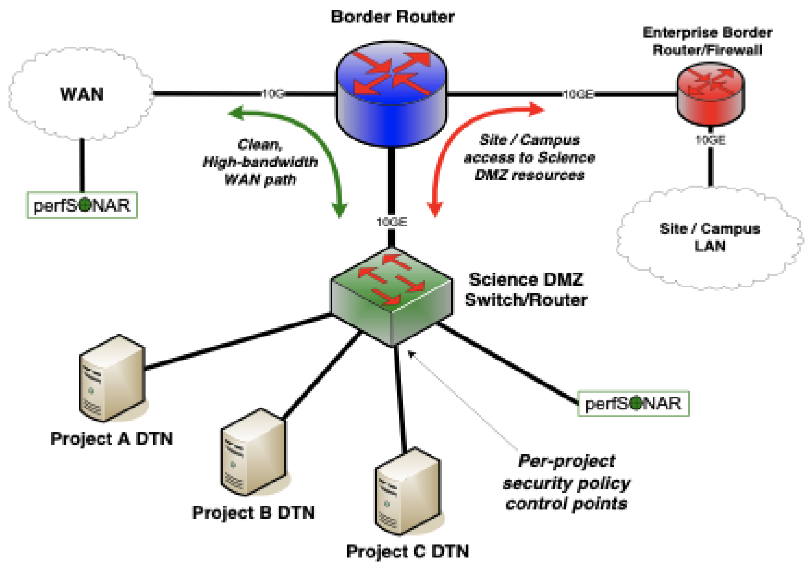
Border Router
Enterprise Border Router/Firewall
WAN
10G
10GE
Clean, High-bandwidth WAN path
Site / Campus access to Science DMZ resources
10GE
perfSONAR
10GE
Site / Campus LAN
Science DMZ Switch/Router
perfSONAR
Project A DTN
Project B DTN
Project C DTN
Per-project security policy control points

Image courtesy of ESnet[1]

Figure 1.1: Example Science DMZ deployment showing DTN placement

As the Figure shows, DTNs are typically located at a host organisation's (logical) network edge, using specific high-end hardware components with fast read and write capability and specialised transfer tools, and are configured and tuned purposely for wide-area data transfer. A perfSONAR [perfSONAR] measurement point is deployed in the DTN network to support measurements of network characteristics to other organisation DTNs who also have perfSONAR deployed.

It is worth noting that WP6 is also jointly a developer of the perfSONAR software with partners in the US [perfSONAR_info]. Deployment of perfSONAR alongside DTNs is highly recommended to assist in diagnosing any network performance issues.

Communities that have extensive long-standing experience of large-scale data transfers, such as the Worldwide Large Hadron Collider Computing Grid [WLCG] or the AENEAS project [AENEAS-D41] have already evolved such good practices, but a growing number of emerging disciplines have yet to adopt them. While there is a technical best current practice (BCP) available, non-technical issues, including lack of user engagement, mean that data transfers for many users are much less efficient, and slower, than the theoretical bandwidth available end to end – because such users are unaware of best practices they may for example use poor tools, or their file transfer systems may not be optimised or sit behind a heavyweight (for their application) campus firewall.

In order to help NRENs to better support user communities who seek to move large volumes of data across their network, a DTN Focus Group was set up in the Network Technologies and Services

[1] https://fasterdata.es.net/science-dmz/science-dmz-community-presentation/

Development work package (WP6) of the GN4-3 project, which has performed several activities around DTN deployment, the results of which are summarised in this document.

The WP6 Network Technology Evolution task conducted a survey among NRENs to gather their inputs about their current DTN status and activities, which is presented in more detail in Section 2 and is included in the DTN Wiki [DTN-Wiki] which is additionally populated with general information about DTNs, related use-cases, applied methodologies, hardware and software systems and tools.

Section 3 discusses the general theory behind optimising the setup and configuration of DTNs. The performance measurement tests run by the DTN Focus Group on the GÉANT Testbeds Service [GTS] are then described, which includes details of the GTS setup in Section 4, the use of Docker containers for data transfers in Section 5, and finally the results of the DTN tests in Section 6. Conclusions are then presented in Section 7.

It should be mentioned that GÉANT has been proactive in taking part in projects such as AENEAS to work with science communities to help them establish infrastructures for large-scale data transfer [DTN-PMW] [DTN-JISC] [DTN-LHC] [DTN-CERN]. GÉANT has operated two 100Gbps DTNs in support of this work. There are also some very good examples of work being carried out on high-performance data transfers internationally, e.g. Big Data Express [BDEXPRESS] and the Pacific Research Platform [PRP]. This document however is focused on the "long tail" of science and providing a resource for NRENs who wish to better understand how they might support the research communities in that long tail.

2 The DTN Survey and DTN Wiki

GN4-3 WP6 has been investigating how European NRENs support their user communities in making optimal use of their networks for large scale data transfers, including the use of DTNs. This was done by exploring DTN work in the GÉANT NREN community and abroad, looking at use cases, related projects, tools, architectures and methodologies.

The team conducted a survey among NRENs, 29 of which submitted responses. A presentation of the survey results, including a summary of the findings and followed by a discussion, was given at the 18th STF meeting [18STF-DTN].

One of the conclusions drawn from the survey results was that there is no clear requirement to develop a specific DTN hardware or software solution, given that very few NRENs (only two among those who responded) operate a DTN for their users on their network backbone. Many of the responses indicated that NRENs see throughput issues as almost always being in the "last mile" of the organisations running the data transfers, typically related to local network limitations, poorly tuned systems or non-performant firewalls. A range of good data transfer software tools also exists, as evidenced and used by the CERN research community and the WLCG.

On the other hand, a challenge in disseminating best practices to the "long tail" of research disciplines was mentioned. This is evidenced by the fact that despite good practice having been established, the uptake of that practice is not as widespread as would be desirable. While NRENs are already active in such work, as reported in the survey (with site visits, case studies, and promotion of Science DMZ in campuses), further support appears to be required.

Following up on the survey, a DTN Focus Group was created, formed of participants in the sub-task with experience and interest in data transfer architectures. The work of the DTN Focus Group, which aims to provide guidance to NRENs looking to support scientists from all disciplines that need to transfer large amounts of data, is further explained in the following sections of this document.

From the survey it also emerged that there was a perceived need in the community for knowledge sharing, with the aim of increasing the level of collaboration and information exchange between the NRENs. For this purpose, the team has created and maintains a DTN wiki [DTN-Wiki] listing tools, use cases, related projects, links to events and useful information related to DTN deployment. The wiki and survey results represent a resource to support scientists from all disciplines and NRENs who wish to better serve their communities in finding the easiest yet most effective approaches for transferring large amounts of data.

Please note that the DTN Wiki page is updated and maintained by the DTN team. Therefore, the sections and contents may change according to the work that is completed by the team and/or other projects or initiatives for high-volume data transfers. Any feedback from the community or suggestions for the wiki and DTN work are welcome. Contacts for the team members and links to enable discussion of the topic via email and a Slack channel can also be found on the wiki [DTN-Wiki].

3 Optimising DTN Configurations

Several factors can affect the performance of data transfers. To achieve a high level of performance, ensuring the DTN elements of a Science DMZ deployment are properly configured and tuned is of key importance.

Each use case should be examined in terms of the specific requirements involved. It is important to keep in mind that changes that improve performance in one data transfer scenario might negatively affect it in another, as well as to understand what effect these changes have on end-to-end transfer performance as a whole. For example, TCP tuning parameters for long-distance (high round trip time (RTT)) transfers are generally different from those for short-distance transfers. Specific values for such parameters are available from various sources. ESnet provides general performance advice [FASTERDATA] which includes settings suitable for certain DTN setups, e.g. Linux tuning information [ESnet]. In this section, some of the key issues are presented and discussed, and references to additional information included where available.

DTNs commonly mount a connected file system, whether a Storage Area Network (SAN) or High Performance Computing (HPC) network, with a network interface to either transmit or receive data files. Dedicated software transfer tools such as [GridFTP], [XRootD], [XDD], [FDT], [BBCP], [FTS-CERN], etc. are best installed on a DTN instance to achieve better input/output performance concerning data transfer.

Since DTNs are placed in a high-performance network “on-ramp” at sites, as shown in Figure 1.1 above, for security reasons, only software for dedicated data transfers is installed on the servers with "allow" access only to the endpoint sites (not open to the normal Internet traffic), and any in-network filtering is typically performed by efficient ACLs, rather than full, heavyweight stateful firewalls (of a type that would protect the rest of the site network for its day-to-day business traffic).

Please note that while following the next practices for tuning on the DTN endpoints for a data transfer, it is also important for the local network architecture to be appropriately designed.

In the following sections, examples of DTN tuning for Linux DTNs are described. More information and details, for example on pacing and MTU settings, can be found at [FASTERDATA2].

3.1 Networking

Various kernel parameters affect network settings. These kernel parameters can be inspected and modified using the *sysctl* tool or the files under */proc/sys/*. Below, they are referred to using the *sysctl* name.

3.1.1 Socket Parameters

The following are some of the settings that affect socket networking parameters for all protocols.

```
net.core.rmem_default

net.core.wmem_default

net.core.rmem_max

net.core.wmem_max
```

The *rmem* parameters refer to the socket receive buffer size and the *wmem* refer to the send buffer size. Under Transmission Control Protocol (TCP) socket parameters, a sending host will typically need to buffer data to support the bandwidth delay product (BDP) of the link, i.e. enough memory for all bits in flight; the higher the RTT of the link, the more memory that is required (see the TCP Throughput Calculator [TTC] resource for an example of a BDP and buffer size calculator). As the names imply, those ending in *default* set the default value used, and those ending in *max* set the maximum value that can be requested by a program. A privileged program can set a buffer size beyond the maximum value [socket].

3.1.2 TCP Socket Parameters

The following parameters affect socket sizes for the Transmission Control Protocol (TCP) protocol in particular.

```
net.ipv4.tcp_mem

net.ipv4.tcp_rmem

net.ipv4.tcp_wmem
```

The tcp_rmem and tcp_wmem parameters are similar to the socket parameters in that they influence the socket buffer size but they are set differently. Each takes three values: a minimum, a default and a maximum. The default value overrides the value set by rmem_default and wmem_default. The maximum values do not override the settings of rmem_max and wmem_max. The minimum and maximum values set the range in which TCP can dynamically adjust the buffer size. This does not affect the buffer size that a program can request. The setting tcp_mem affects how TCP manages its memory usage. It takes three values: low, pressure and high. Below the value of low, TCP will stop regulating memory usage (if it was previously). Above the value of pressure, TCP will start regulating its memory usage. The high value is the maximum amount of memory that TCP will use. These are not set in bytes but in number of pages of memory. They refer to global memory usage, not individual sockets [tcp].

Typically, larger buffer sizes are better for higher throughput or data transfers with a high round-trip-time. However, unnecessarily high values will waste RAM.

Another TCP parameter which can have a significant effect on performance is the congestion control algorithm used. The following parameters influence which algorithm is used:

```
net.ipv4.tcp_available_congestion_control

net.ipv4.tcp_allowed_congestion_control

net.ipv4.tcp_congestion_control
```

The tcp_available_congestion_control parameter is read-only and shows what algorithms are available on the system. The tcp_congestion_control parameter sets the default algorithm used; the Fasterdata site recommends htcp, but the newer TCP Bottleneck Bandwidth and Round-trip time (BBR) developed by Google and being standardised by the Internet Engineering Task Force (IETF) is certainly worth considering due to its greater resilience to packet loss). A program may instead use any of the algorithms set in tcp_allowed_congestion_control and a privileged program can use any of the available algorithms [tcp].

In order for a particular algorithm to be available, it may require that a module is loaded into the kernel. While choosing the right congestion control algorithm can improve performance it also affects fairness. On shared links, some algorithms may negatively impact the network performance of other systems. For example, TCP BBR is more “aggressive” and thus may dominate some traditional TCP algorithms.

The ss command can be used to get detailed information about current TCP connections including what congestion control algorithm they are using:

```
ss --tcp --info
```

3.1.3 NIC Driver Settings

In addition to kernel parameters, network performance can be tuned by adjusting settings that affect the actual network hardware. NIC manufacturers may supply proprietary tools to adjust settings. However, some network driver settings can be modified using the ethtool command. Two important settings that should be looked at are the size of the send and receive rings and the offload features:

```
ethtool --show-ring INTERFACE

ethtool --show-features INTERFACE
```

Larger rings typically improve throughput but can have a negative impact on latency or jitter. Offloading features allow some work that would need to be done by the CPU to be done on the NIC instead. This can lead to lower CPU utilisation and higher throughput.

3.1.4 MTU Settings

The Maximum Transmission Unit (MTU) size can have a significant effect on throughput. For most typical Ethernet networks, the default MTU is 1500 bytes. This means each Ethernet frame sent will have a maximum size of 1500 bytes. However, some networks may support larger MTUs, often up to 9000 bytes. Where such MTUs are available end to end, greater throughput can be achieved (see the TCP Throughput Calculator [TTC] to explore the effect). A higher MTU also puts less pressure on devices on the path in terms of their requirements to support high packet per second processing.

Many NREN backbone networks support 9000 MTU, while some support slightly higher to allow for some framing overhead. Where a Science DMZ network is connected directly onto an NREN backbone through its site router, it may be possible to configure some or all of the Science DMZ to a higher MTU.

The example tuning for the GTS DTN setup shown in Appendix A uses an MTU of 8986 bytes, using a simple ifconfig to set the MTU value.

3.2 Storage

When performing disk-to-disk transfers, tuning the storage devices can be just as important as network tuning. Storage also needs sufficient read and write performance so that it does not become a bottleneck for data transfers. The type of storage device used and how it connects to the system will affect what parameters can be tuned. The following subsections include some of the areas that should be examined when deciding on what hardware to use and how to maximise performance.

3.2.1 Bus vs Disk Speed

It is important to look at a storage device's transfer rate and not just at that of the interface. A drive may internally have a transfer rate that is much lower than the speed of the interface it connects to. Two common connection types for storage are Serial Advanced Technology Attachment (SATA) and Serial Attached SCSI (SAS). There are multiple versions of both, but SATA-3 and SAS-3 are common. SATA-3 has a data throughput of 6Gbps, and SAS has a throughput of 12Gbps. Most storage devices also transfer data sequentially faster than random access so expect a performance impact when data is accessed non-sequentially.

3.2.2 Spinning versus Solid State Disks

Traditional spinning disks tend to be slower than solid-state disks, but cheaper and available in larger capacities. Both types of disks can connect to the system via SATA or SAS. Spinning disks typically have slower internal throughput than either the SATA or SAS bus, meaning that they cannot sustain transfers at the bus speed. Some SSDs, but not all, are capable of matching the bus speed.

3.2.3 NVMe

Solid-state disks can also be attached to a system using NVMe. This means that they are directly attached to the PCIe bus, which is substantially faster than either the SATA or SAS bus. The disadvantage of using the PCIe bus is that, compared to SATA or SAS, fewer storage devices will be able to connect. Only so many PCIe lanes are available per CPU and they are also needed for other hardware devices. PCIe versions 3.1 and 4 are common. Standard NVMe drives use four PCIe lanes each giving them a potential throughput of 31.5Gbps with PCIe 3.1 and 63.0Gbps with PCIe 4.0.

3.2.4 RAID

When a single storage device cannot achieve the desired level of performance, multiple devices aggregate into one logical device. RAID allows multiple devices to be combined in various configurations. While RAID is intended mostly to increase reliability, depending on the configuration, performance and/or reliability of the storage system can be improved. High storage throughput can be achieved by utilising multiple disks in a RAID-0 configuration at the cost of reduced reliability.

A RAID controller is a hardware device that is between the host and storage devices. It performs the necessary calculations to present multiple storage devices to the operating system as one logical device. Instead of using a physical controller, RAID can also be implemented in software. Zettabyte File System [ZFS] is a file system that supports software RAID arrays. This allows for the aggregation of storage devices to increase read and write performance without the use of dedicated hardware. However, this may not achieve the same performance as a hardware implementation and will create additional work for the CPU.

3.2.5 Remote Storage

Multiple types of storage adapters can be used to connect to remote storage such as InfiniBand, Fibre Channel or even Ethernet. The storage adapter selected has similar requirements to the network adapter. The throughput to the storage system should be at least as fast as the desired data transfer rate and offloading features are desirable to reduce the load on the CPU.

3.3 Architecture

Besides the storage and networking subsystems, there are a few other elements that should be considered. The underlying hardware architecture everything else is running on is of course important. The selected CPU and memory must fit the use case and be configured properly. AMD provides a variety of guides for tuning their EPYC processors [EPYC].

3.3.1 Fast Cores vs Many Cores

Data transfer applications can be CPU intensive. However, whether it is better to have a few very fast cores or many slower cores depends on whether the application is single or multi-threaded. To complicate matters, this may also depend on the data set. Applications such as GridFTP cannot use multiple threads to transfer a single file but can use a separate thread for each file in a data set [Globus-FAQ].

3.3.2 NUMA

Non-Uniform Memory Access (NUMA) is when access times differ depending on which CPU is accessing which portion of memory. How devices are connected also determines which NUMA node they are in. For the best performance, a process should run in the same NUMA node as the data it needs to access. Netflix has shown that tuning NUMA parameters could substantially improve data transfer performance [NUMA]. Tools such as numactl and numad can be used to manage NUMA policies. To determine which NUMA node a device is in, lspci can be used. For best performance, it may be necessary to physically move a NIC or storage controller, so that they are in the same NUMA node. For NUMA layout to be visible to an operating system, it must be enabled in the BIOS.

The following command can be used to display NUMA nodes on a system:

```
numactl --hardware
```

3.3.3 IRQ Handling

For best performance, interrupts should be handled as close to the hardware as possible, while not overloading a single CPU. The daemon **irqbalance** controls how hardware interrupts are distributed over the CPU cores.

3.3.4 Power Saving

Features designed to save power when the system is not fully utilised may be desirable in production. When measuring system performance, they should be disabled to provide a more accurate measure of what a system is capable of. The utility **cpupower** controls power management features of a CPU. The available options depend on the processor.

Here is an example of how to change the CPU frequency governor on all CPU cores:

```
cpupower --cpu all frequency-set --governor performance
```

4 Setting up DTN Tests on GTS

In this section, the GÉANT Testbeds Service (GTS), its configuration for the DTN tests described later in the document, and the software transfer tools chosen for the tests are described.

4.1 The GTS Testbed

GTS provides users in the GÉANT research community with an automated platform on which to build virtual networks that can include resources such as VMs, Bare Metal Servers (BMS) or switch elements. The purpose of GTS is to allow rapid prototyping and tests of novel networking and telecommunications concepts, at scale, and across a geographically practical European footprint. In terms of its users, GTS is intended to aid research teams exploring novel SDN-based solutions and requiring the use of a high-performance distributed infrastructure.

GTS can furthermore be utilised by application and software development teams needing an isolated testbed to demonstrate their designs without affecting live internet traffic. GTS is rationally isolated from the production GÉANT network to guarantee the integrity of live applications and can support multiple isolated networks concurrently permitting teams to work without affecting each other [GTS]. Figure 4.1 shows the GTS nodes setup map in Europe. Sites are currently connected with 10Gbps links.

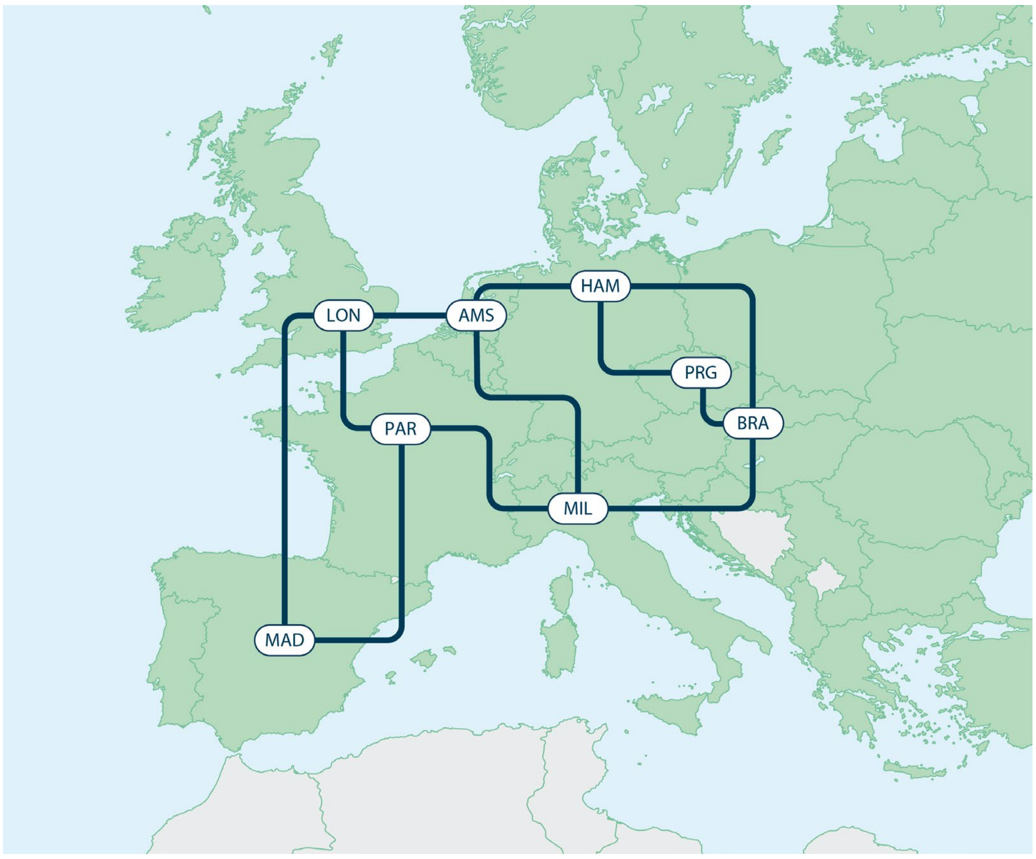

Figure 4.1: GTS physical infrastructure topology

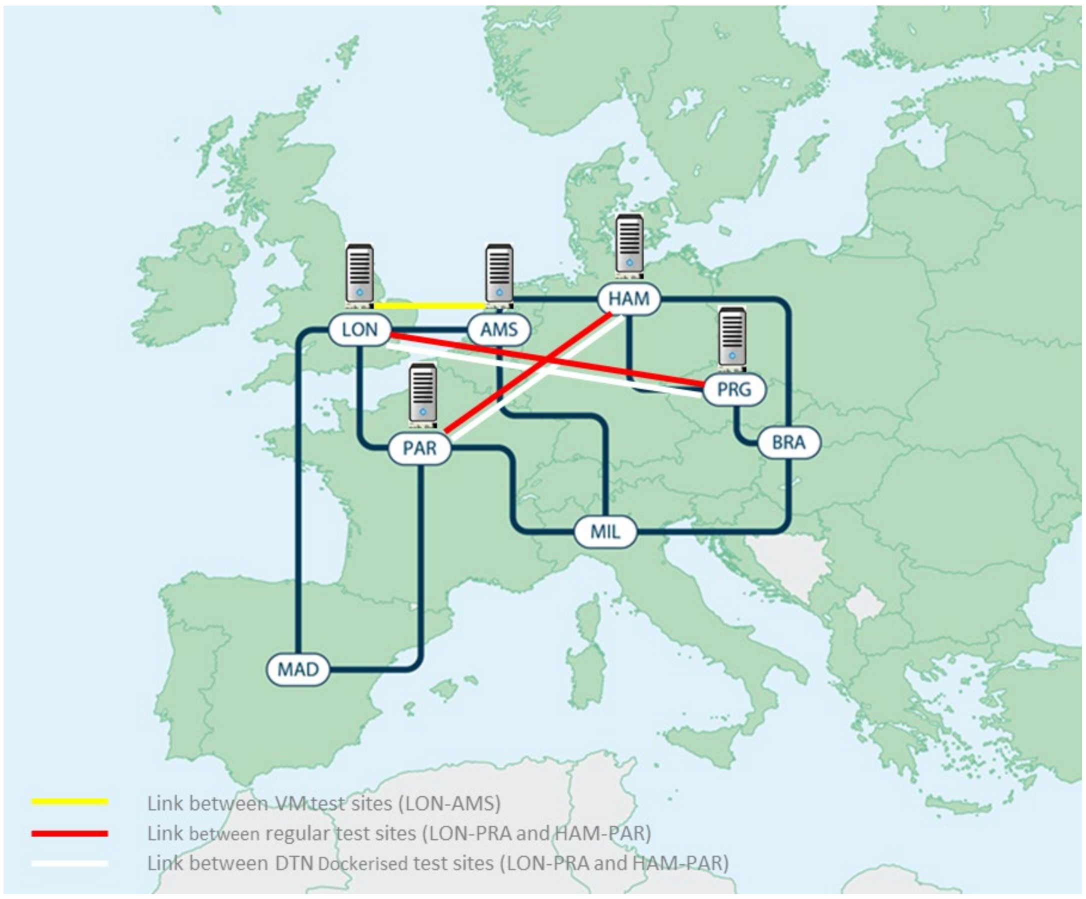
LON
AMS
HAM
PRG
BRA
PAR
MIL
MAD
Link between VM test sites (LON-AMS)
Link between regular test sites (LON-PRA and HAM-PAR)
Link between DTN Dockerised test sites (LON-PRA and HAM-PAR)

Figure 4.2: Bare metal servers used for DTN tests

The DTN Focus Group used GTS to set up data transfer tests using bare metal servers (both regular and dockerised) and VMs located in several GÉANT nodes in Europe (as shown in Figure 4.2), i.e. at Amsterdam, London, Hamburg, Paris and Prague.

The MTU used for all the tests was 8986 bytes.

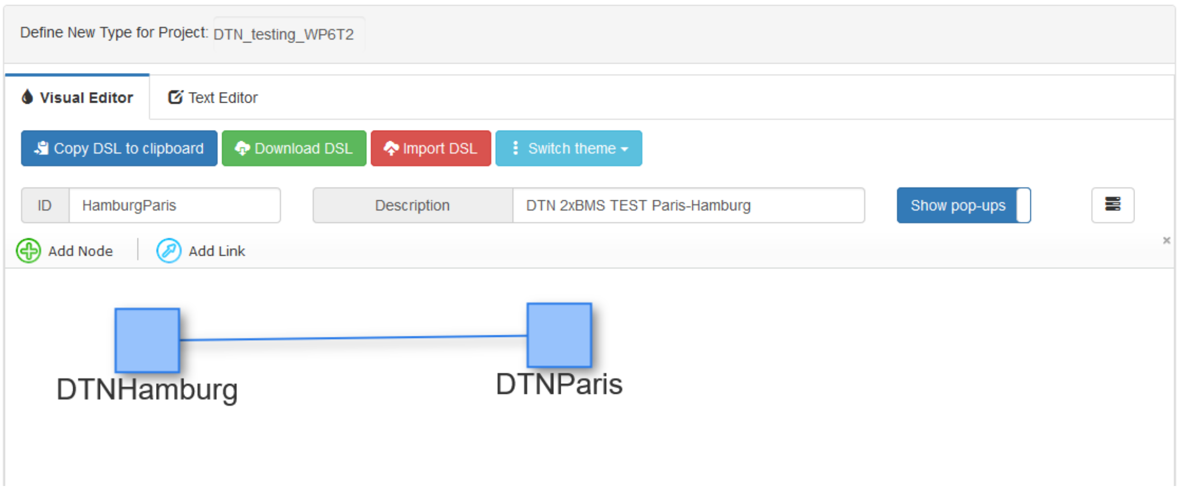

Figure 4.3: GTS logical infrastructure topology between Hamburg/Paris BMS [GTS-SVC]

Figure 4.3 shows the logical topology setup of two BMS servers in Hamburg and Paris that are connected through a common link (10Gbps link) in GTS, as seen in the GTS service Graphical User Interface. In this example, one node acts as the DTN Server and the other node acts as a DTN Client. The logical topology[2] setup taken from the GTS as shown at Figure 4.3, is the same for all the pairs of servers-clients used in the executed tests (i.e. Amsterdam - London, London - Prague, Hamburg - Paris). For the tests with VMs where the client and the server were in the same City (Amsterdam - Amsterdam), each VM was on a different hypervisor.

4.2 Software Transfer Tools Selected

For the tests, the group used three well-known big data transfer tools – GridFTP, FDT and XRootD – that are described in the following subsections.

4.2.1 GridFTP

GridFTP is an extension of the classic File Transfer Protocol (FTP). The protocol was defined within the GridFTP working group of the Open Grid Forum [OGF] and provided by the Globus Toolkit [GLOBUS]. The purpose of GridFTP is to provide a more reliable and high-performance file transfer to enable the transmission of very large files; it typically uses multiple parallel TCP streams, thus parallelising the transfer and making it less susceptible to packet loss on any one stream.

Until recently, GridFTP was used heavily by the WLCG, but they are now migrating towards use of an alternative third-party copy tool, particularly XrootD and/or HTTP/WebDAV[3].

Globus Connect uses GridFTP, and while the full server package requires a licence, the basic transfer tool element remains free at the time of writing for R&E use. [Globus Connect] is used by a number of university sites, and is capable of achieving very good data rates.

4.2.2 FDT

Fast Data Transfer [FDT] [FDT2] is a data transfer application which is capable of reading and writing at disk speed over TCP. FDT is based on an asynchronous, flexible multithreaded system and utilises the capabilities of the Java NIO libraries. Its primary features are:

- Streams a dataset (list of files) persistently, utilising a managed pool of buffers through one or more TCP sockets.
- Employs autonomous threads to read and write on each physical device.
- Transfers data in parallel on numerous TCP streams, while using appropriately sized buffers for disk I/O and the network.
- Re-establishes the files from buffers asynchronously.
- Resumes a file transfer session without loss, when needed.

[2] Logical topology in terms of link setup and servers setup is shown in the appendix A
[3] https://twiki.cern.ch/twiki/bin/view/LCG/ThirdPartyCopy

4.2.3 XRootD

[XRootD] is a software framework, distributed under the terms of the GNU LPGL License, for fast, low-latency and scalable data access that aims at giving high-performance, scalable, fault-tolerant access to any data repositories. It is organised as a hierarchical file system-like namespace, based on the concept of directory, and is based on a scalable architecture. It is currently being used by the WLCG for many of its large-scale data transfers.

5 Docker Containers for Data Transfers

5.1 Docker Environment

Docker [DOCKER] is a set of platforms as a service (PaaS) products that use OS-level virtualisation to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating system kernel and therefore use fewer resources than virtual machines. The service has both free and premium tiers. The software that hosts the containers is called Docker Engine. It was first started in 2013 and is developed by Docker, Inc.

5.1.1 Why DTNs on Docker?

The DTN Focus Group has worked on setting up DTNs on Docker as a way of easily implementing data transfer solutions as:

- It supports the setup of containers with specific services (DTN services) that can be accessed through the host IP address.
- It allows for sharing folders among the host and the containers.
- It permits unrestricted usage of CPU, memory and bandwidth (interface), only restricted by the hardware limitations of the host.
- Containers are light in terms of resource consumption.
- There can be multiple services (each service being a container) on a Docker host.
- It provides an easy way to make the tools available within a negligible time.

Docker is constructed with the vision of providing flexibility, dynamicity and easy integration of updates on containers. A DevOps engineer can easily update the DTN software to add another container, push the container to the live system and delete the old container. This functionality is valuable for DTN testing and integration in terms of reducing risk and complexity in updates and when new major updates have to be enabled in the system. The aforementioned functionality will reduce time, cost and risk when updating the DTN services and software. In addition, Docker fully utilises the capabilities of the Linux Kernel via LXC[4], which allows sandboxing processes from one another, and controlling their resource allocations, i.e. this feature helps to fully optimise the utilisation of DTN processes in containers [DOCKER_LCX], [DOCKER_LCX1].

[4] LXC is a userspace interface for the Linux kernel containment features. Through a powerful API and simple tools, it lets Linux users easily create and manage system or application containers.

5.1.2 Docker Architecture

The Docker architecture is a very straightforward client-server architecture [DOCKER-OVW]. The client communicates with the Docker daemon (that can be installed on an identical or separate remote machine), which does the building, running, and distributing of the Docker containers. The Docker client and daemon communicate using a Rest API over UNIX sockets or a network interface.

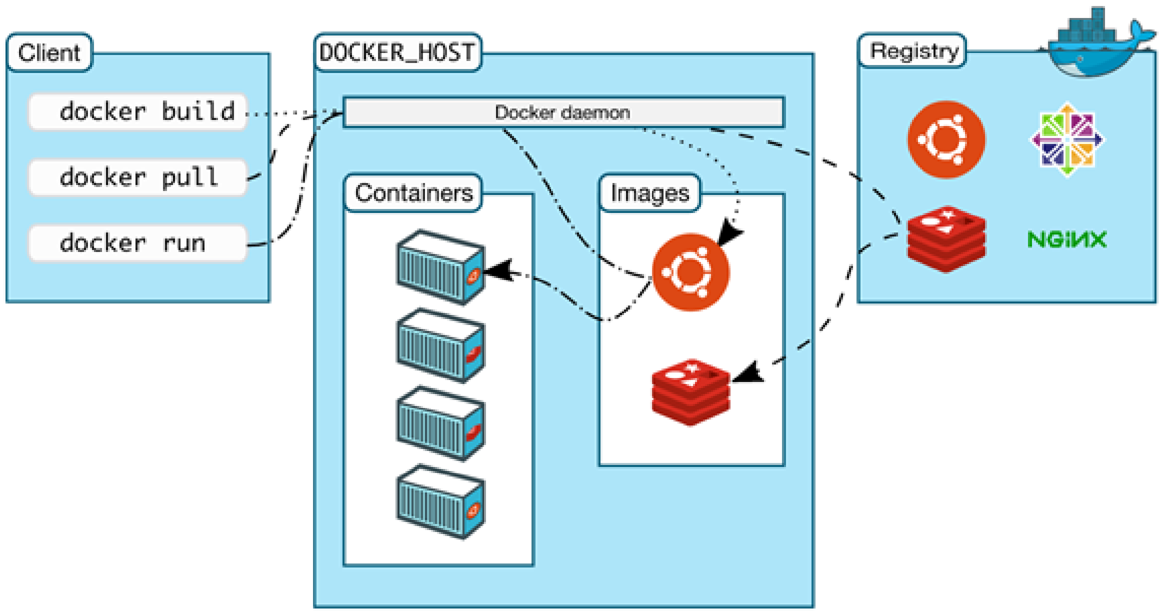

Figure 5.1: Docker architecture [DOCKER-OVW]

The architecture components of Docker are the following:

- Docker daemon (system service called "dockerd"): listens for requests and manages Docker objects such as images, containers, networks, and volumes alone or in coordination with other daemons that are installed on other machines.
- Docker client (docker): is a command line tool for connecting the Docker daemon. The docker command uses the Docker API.
- Docker registry: stores Docker images (anyone can set up their own registry on premises). An example is the "Docker Hub" which is a public registry that anyone can use (Docker is configured to look for images on Docker Hub by default). There are some pre-specified commands in order to access the registry i.e. the docker pull/push and docker run commands. The aforesaid commands make the required images to be pulled from/pushed to the configured registry.
- Docker objects: Docker is creating and using images, containers, networks, volumes, plugins, and other objects. The aforesaid components are handled as objects by Docker.
- Image: this is a read-only template with instructions for creating a Docker container. An image may be based on another image. Users could also create their own images with the use of Dockerfile.
- Container: is a runnable instance of an image, allowing users to create, start, stop, move, or delete a container using the Docker API or CLI with the use of Docker client. A container can

be connected to one or more networks, have storage attached to it, or a new image even be created based on its current state.

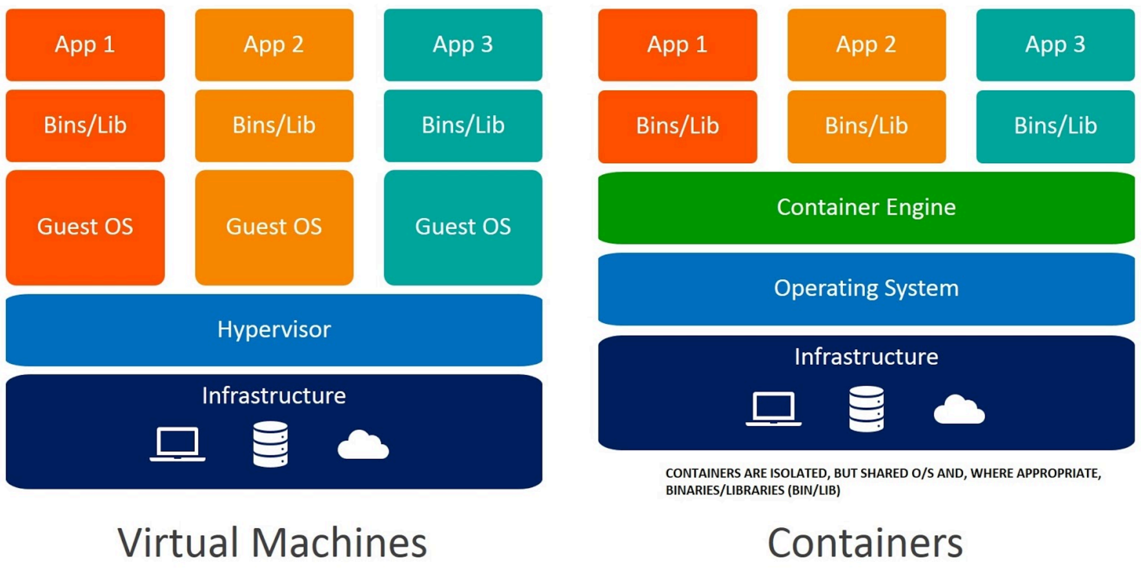

Figure 5.2: Containers vs VMs [CONT-VM]

Containers and VMs differ in terms of their scalability and portability:

- Containers are small in disk size. They do not package anything bigger than an app and all the libraries and files fundamental to run it. The lightweight nature of containers with the use of a shared operating system makes the move across multiple environments effortless.

- VMs are relatively huge in disk size. They include their own OS, permitting them to perform numerous resource-intensive functions at once. The expanded resources offered to VMs permit them to abstract, split, duplicate, and emulate entire servers, OSs, desktops, databases, and networks.

Virtual machines and containers also diverge in numerous behaviours, but the principal difference between them is that containers provide a technique to virtualise an OS so that different workloads can run on a single OS instance. Conversely, when utilising VM technology, the hardware is being virtualised to run multiple OS instances. Therefore, containers' speed, agility, and portability make them an excellent pillar for technology deployments.

5.2 Docker Installation with Docker Containers on GTS

The DTN Focus Group team has run several tests in a virtual environment using containers with Docker. This section focuses on the implementation of a Dockerised environment to support specific file transfer services (i.e. XRootD, GridFTP and FDT).

In parallel, any research and education institution interested in the development of a data transfer service can readily deploy their own virtual DTN servers following some general specifications. More specifications and guidelines can be found at [DTN-Wiki] and other good references are available at [FASTERDATA], [DOCKER_ESNET_GITHUB]. Once this code is ready for testing, after the setup of the Dockerised environment and the installation of the containers, the participating entities will be immediately able to run the initial testing and evaluation of DTN software, which would typically include criteria such as:

- Performance of data transfers for different data workflow characteristics: large bulk transfers, lots-of-small-file transfers, and streaming data transfers.
- Ease of use for end-users.
- Trust negotiation between the end hosts and security of the transfer.

Memory consumption and CPU consumption can be managed within containers in Docker, although in this case there is no restriction for the memory and CPU of the containers [DOCKER_RES_CONS]. Thus, the containers have a pre-set CPU and memory value, but if they need more, they can directly request it from the host server. In this example, the services configured in the containers are gridFTP, XRootD and FDT, using ports other than the default ports.

In the example, “ctop” is used to get statistics from the containers (bandwidth used, CPU used, memory used). In addition, a client script is created for transferring huge files. This script is monitored with the use of “bmon” and “iftop” tools for measuring bandwidth usage of the links. During the process, iPerf was used for link testing. The results of the tests can be found in Section 6.

Scripts are provided for reviewing the usability of DTNs and with the target of achieving the best possible results. The scripts are tested on GTS and they are fully automatic for any operating system (Centos/Ubuntu). Guidelines are provided on the wiki site [DTN-Wiki] along with the scripts for a step-by-step installation of Docker and DTN services on the host, and the execution scripts of the test script (script that is doing transfers from the client to the Dockerised test server). For a more in-depth understanding of Docker installation, additional content with guidelines can be found on the web in a plethora of articles for different operating systems [DOCKER_INST_CENTOS1], [DOCKER_INST_CENTOS2], [DOCKER_INST_UBUNTU].

The steps for setting up a Dockerised environment and executing tests in this environment are the following:

1. The Docker install script, which is implemented for Ubuntu (installDockerUbuntu18_04.sh) and Centos (installDockerCentos8.sh).
2. The Docker containers install script, which is implemented for all Operating Systems (runAllDockers.sh); this creates three services (i.e. FDT, XRootD, GRIDFTP) where each service is run in one container on the hosting machine using a specific image and pre-specified software defined by the service. Afterwards, it starts the containers and runs the services.
3. The execution of a client script, which is implemented for all Operating Systems (test_DifferentPorts250GBNew.sh); this runs on the client host and transfers 250 GB of dummy data to the server.
4. The usage of tools to monitor the transfer from the services (via the Dockerised environment) as well as the client bandwidth utilisation (via the monitoring of the interface bandwidth utilisation).

The whole sequence can be found in a video published in the DTN wiki [VIDEO].

6 DTN Tests Results

This section describes the results gathered from the execution of the DTN test using different setups. For the tests, the aforementioned services and software shown in Section *4 Setting up DTN Tests on GTS* are used. Additionally, the system, software and hardware parameters in the tests are tuned accordingly (as shown in Section *3 Optimising DTN Configurations*). Note that some optimisations needed tuning at the kernel level and hardware level.

The results of the various tests reported in this section were reported at the Performance Management Workshop [PMW] in March 2020 and an online Infoshare [DTN-Info] is being held on 9 December 2020. The materials from the infoshare will be made available on the [DTN Wiki].

The following topologies were used:

1. Virtual Machines, shorter distance (AMS-AMS):
    a. 1 CPU
    b. 2 CPU
    c. 4 CPU
2. Virtual machines, longer distance (AMS-LON):
    a. 1 CPU
    b. 2 CPU
    c. 4 CPU
3. Bare metal servers, shorter distance (HAM-PAR).
4. Bare metal servers, longer distance (LON-PRA):
    a. R430
    b. R520
5. Dockerised environment on bare metal servers, test 1 (HAM-PAR).
6. Dockerised environment on bare metal servers, test 2 (LON-PRA).

For the bare metal tests, large files were used for transferring data between the client and the server. A single file of 256 GB was used for each test. This file size was chosen to ensure that the system could not buffer the entire file in memory for reading or writing. In addition, before each test, the write buffer was flushed and the disk cache was dropped. The optimisation parameters used are shown in the results section.

An important note about the results is that they may have been affected not only by the hardware (i.e. CPUs, memory, link) and the distance but also by the fact that the links are not dedicated and other tests could have been running simultaneously on GTS.

In the rest of this document, "regular tests" refers to those run on the GTS environment, using both virtual machines and bare metal servers, but not considering the Dockerised environment tests, which are specified separately.

The following matrixes show a summary of the setups for each test, the software installed, and the performance achieved, as well as the links to the information on how to install each software in GTS and to the report for each test. Finally, some comments related to each setup and test are included.

Virtual machine	1 CPU		2 CPU		4 CPU	
Nodes/Tools	AMS-AMS	AMS-LON	AMS-AMS	AMS-LON	AMS-AMS	AMS-LON
iPerf	9.90Gbps	9.90Gbps	9.90Gbps	9.90Gbps	9.90Gbps	9.90Gbps
gridFTP	8.30Gbps	8.36Gbps	8.86Gbps	8.47Gbps	8.50Gbps	7.51Gbps
FDT	9.32Gbps	7.90Gbps	9.19Gbps	8.49Gbps	8.98Gbps	7.77Gbps
XRootD	2.60Gbps	2.60Gbps	2.60Gbps	2.60Gbps	2.60Gbps	2.60Gbps

	Docker		Bare Metal Servers		
	R430	**R430**	**R430**	**R520**	**R430**
Nodes/Tools	**HAM-PAR**	**LON-PRA**	**HAM-PAR**	**LON-PRA**	**LON-PRA**
iPerf	9.2Gbps	9.0Gbps	9.41Gbps	9.32Gbps	9.43Gbps
gridFTP	8.53Gbps	8.50Gbps	8.58Gbps	3.30Gbps	8.52Gbps
FDT	8.87Gbps	8.70Gbps	9.39Gbps	4.12Gbps	9.39Gbps
Xrootd	8.00Gbps	8.00Gbps	8.00Gbps	3.13Gbps	7.99Gbps

Table 6.1: Test Setups

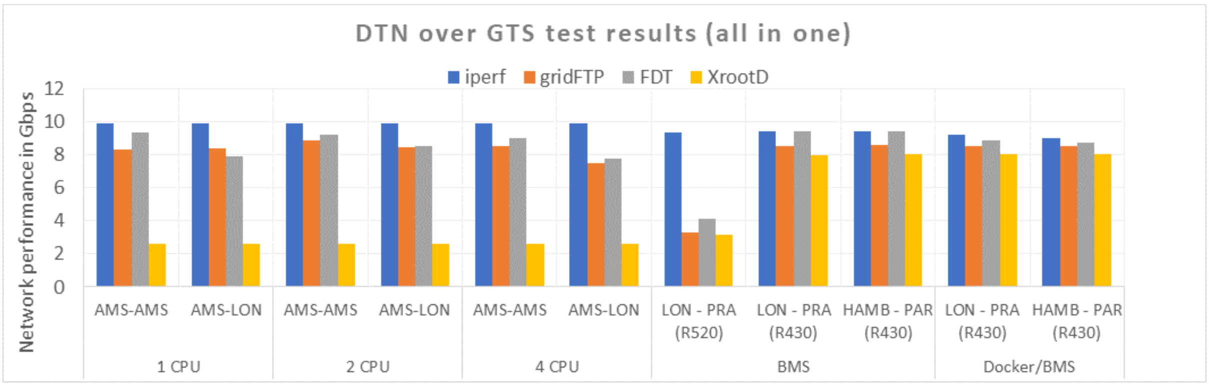

Figure 6.1: DTN over GTS Test Results with BMS and VMs

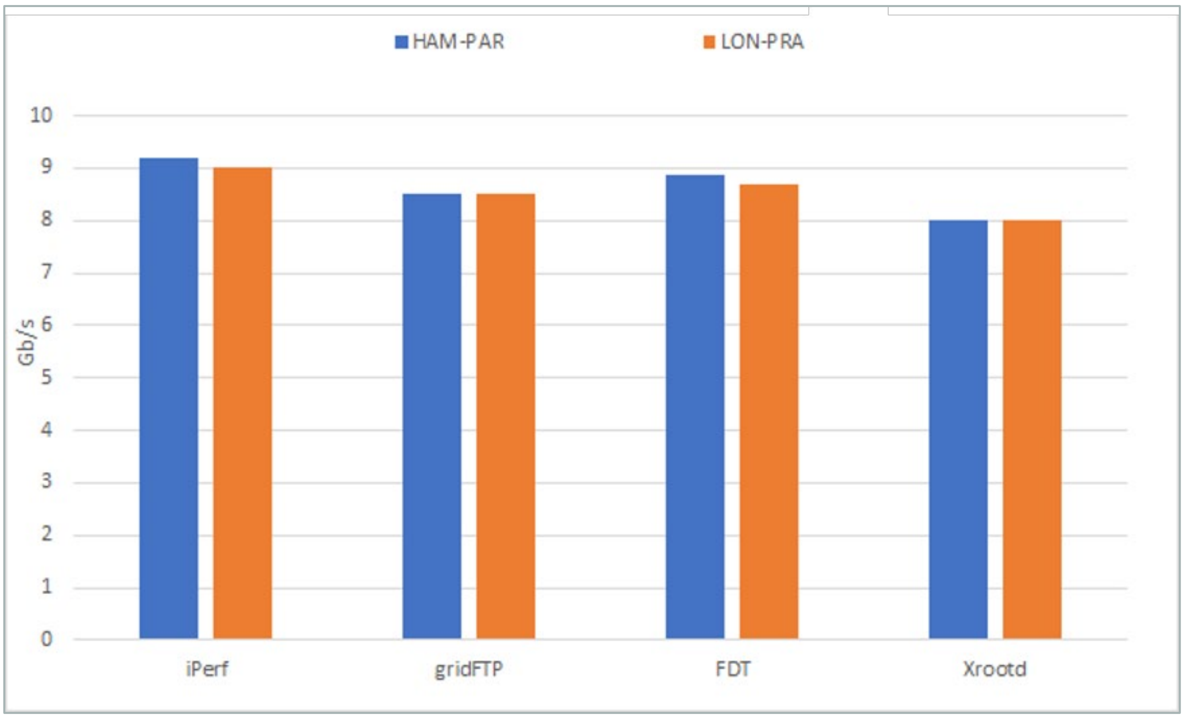

Figure 6.2: DTN over GTS Test Results with Docker

6.1 Regular Tests

The tests started with DTNs using several software tools installed in virtual machines with 1, 2 or 4 CPUs and followed with the same software tools in BMSs.

Hardware: two types of servers were used:

BMS	**Dell R430**	**Dell R520**
Processor	2x20C/40T Intel Xeon E5-2660v3 @ 2,6 GHz	1x8C/16T Intel Xeon E5-2450 v2 @ 2,5 GHz
Chipset	Intel C610 series	Intel C600 series
Memory	ECC DDR4 2133 MHz	ECC DDR3 1600 MHz RAM
RAM	128 Gb	32Gb
Storage	6xSSD, 372 GB, 6,0Gbps	2xSSD, 372 GB, 6,0Gbps
RAID controllers	PERC H730	PERC H710
Network Controller	1GB and 10Gb ethernet NIC	1GB and 10Gb ethernet NIC

Table 6.2: Types of Bare Metal Servers used

OS used for client and server: Ubuntu-18.04-server

Performance metric: Performance measured as the effective transfer rate, calculated as the ratio of the Data File Size and the Elapsed Real Time, and this also represented the bandwidth utilisation. Two file sizes were used – 250 GB and 420 GB.

Parameters tuned:

```
sudo sysctl -w net.ipv4.tcp_rmem="4096 16777216 2147483647"

sudo sysctl -w net.ipv4.tcp_wmem="4096 16777216 2147483647"
```

In the first investigation, DTNs on virtualised environments using VMs were tested. For the direct measurement of the links between the client site and the server, iPerf was used. The results in virtualised environments showed that all evaluated programs achieved high bandwidth utilisation except XRootD. For the latter, the reason for not achieving high bandwidth utilisation was the virtualisation itself. XRootD is highly dependant on hardware resources and the virtualised environment with the virtual hypervisor on the physical system with the lack of direct access to disks and buffer memory resulted in low bandwidth utilisation. On the other hand, the rest of the DTN services and client programs investigated (i.e. FDT, gridFTP) achieved higher bandwidth utilisation. A note about the aforementioned investigation is that all tests were executed using memory-to-memory transfers. This first experiment was executed in virtual servers with different distances between the client and the server. In this case, both VMs were in Amsterdam for the shortest distance tests while for the longest distance test one VM was in Amsterdam and the other in London.

Additionally, the tests with VMs showed that better results were achieved with 2xCPU than with 4xCPU. This may be due to different reasons: the different server architecture, Hyper-V VM Load Balancing, or the fact that DTN tools do not know how to better use the CPU resources. It is important to note that during this investigation, changing the VM parameters was not possible. As mentioned before, the results may also be affected by the fact that the links are not dedicated as well as by some tools being multi-stream transfers while others are not, so it may depend on streams and affinity to CPUs. The tests were repeated at different time slots, to see if a pattern could be found regarding performance, but no significant changes were detected.

At the next investigation, the examination of DTN services (same as above) focused on the usage of physical servers. In the beginning, the performance of the DTN services and applications using BMSs was examined at a short distance (Hamburg-Prague), with the latest technology servers available in GTS (Dell R430). The results for all the DTN services and applications were good and all the services performed well in terms of transferring large amounts of data between them (as usual, more than 250GB).

Longer distance data transfers were then also investigated over the GTS testbed between London and Prague with the use of BMSs. The London and Prague nodes, with longest optical distance and BMS availability, were used with a combination of Dell R430 and Dell R520 hardware. This means the available system was not the same as for the previous tests. The Dell R520 is weaker in terms of CPU, memory and storage throughput than the R430 that was acting as a client in the previous tests. The results of the tests (as listed in the results table) showed that the bandwidth utilisation dropped for all the DTN services and applications, highlighting the importance of the hardware to get the best results.

At the final investigation, longer-distance transfers between London and Prague with the use of BMS were investigated. However, in this review the available system was the same as the one used in the previous short-range examinations (Dell R430). The results for bandwidth utilisation showed that nearly identical measurements to those recorded between Hamburg and Prague can be achieved. Distances in Europe are not relevant enough to show any differences like those achieved with South-Africa or Australia in the AENEAS project, for instance [AENEAS], as the impact of packet loss rises significantly for higher RTT (see the TCP Throughput Calculator [TTC] to calculate theoretical network limits).

Overall, the best data rates achieved with 10Gbps links were obtained with FDT, reaching 9.4 Gbps, gridFTP followed with 8.9Gbps and finally XRootD with 8.0Gbps. To get the best results with 10Gbps network links, the servers must have a storage throughput that is at least also 10Gbps. In the tests, this was possible using a RAID array on the R430 but not on the R520. Also, there is a large demand in memory for the packet transfers to be saved in the memory buffer and then at the disks. It is worth mentioning that BMS and "Dockerised BMS" have similar results in the tests applied in a BMS environment.

6.2 Dockerised Tests

Hardware: Dell R430 server (2x20C/40T Intel Xeon E5-2660v3 @ 2,6 GHz, 128 Gb ECC DDR4 2133 MHz RAM, 6xSSD, 372 GB, 6,0Gbps HDD)

O/S used for client and Dockerised server: Ubuntu-18.04-server

DevOps Software used: Docker.

Performance metric: Performance measured as the effective transfer rate (Data File Size / Elapsed Real Time)

Parameters tuned:

```
sudo sysctl -w net.ipv4.tcp_rmem="4096 16777216 2147483647"
sudo sysctl -w net.ipv4.tcp_wmem="4096 16777216 2147483647"
```

In the first investigation, the Dockerised environment with the services was in Paris and Hamburg was the client to run the client script. This was the setup with the shortest distance between client and server Dockerised instances. Please note that Amsterdam-Amsterdam tests are not executed in a Dockerise environment.

In the second investigation, the Dockerised environment with the services was in London, and Prague was the client to run the client script. This was the setup with the longest distance between the client and the server. Comparing both tables, it is evident that there is no major change in bandwidth utilisation from the shorter distance BMS test. However, there was a slightly reduced bandwidth.

The scripts in containers are using a sharing folder in Linux (other volume) called /data for interchanging big files. In the evaluations carried out, there was a 10Gbps direct connection between Paris and Hamburg utilised for the investigation of DTN performance. The services on the Dockerised environment used different TCP port numbers from the default ones, to make sure the test would not be impacted by any QoS or security configuration that might have been configured on the machines.

From the ctop results, it can be seen that each container required more memory and CPU than the Docker already assigned to it. This is due to the requirements of the DTN services [DOCKER_CTOP].

For Dockerised environments, the best data rates were also achieved with FDT, up to 9.2Gbps, followed by gridFTP with 8.6Gbps and finally XRootD with 8.1Gbps. As with the regular tests, all services need the servers to have storage throughput that is at least 10Gbps in order to achieve the required speed. Also, there is a large demand in memory for the packet transfers to be saved in the memory buffer and then at the disks.

6.3 Test Conclusions

Comparing the results achieved with and without Docker, with a different number of CPUs in a VM and with longer and shorter distances, as shown in the tables above, it can be concluded that:

1. There are minor differences in results when using 1, 2 or 4 CPUs in VMs.
2. There are minor differences in results when using VMs and BMSs in terms of memory-to-memory transfers except for XRootD.
3. There are minor differences in results with shorter and longer distances between the server and the client.
4. There are minor differences in results with Dockerised and non-Dockerised environments, proving Docker is a good solution to set up DTNs, at least when using 10Gbps links.
5. Disk-to-disk transfers were slower than disk-to-buffer and buffer-to-disk.
6. The Dockerised tests were repeated with Centos 7 and the results were similar to the results at Ubuntu 18.04.
7. Virtualisation affects the XRootD DTN service and application as shown from the results because XRootD appears to be highly dependent on hardware resources.
8. The chosen hardware can impact the achieved performance (Dell R520 vs Dell R430).
9. BMS and "Dockerised BMS" have similar results.

It should be noted that tuning the tests in a shared network environment (VMs case) may affect the performance.

7 Conclusions

A growing number of projects, disciplines and teams in research and education communities around the world need to transfer large amounts of data, which not only requires high capacity networks, but also specialised transfer tools and dedicated servers with appropriate tuning. While some more established disciplines such as the WLCG have evolved or deployed Science DMZ principles, others in the "long tail" of science have yet to do so.

The GN4-3 project's Network Technologies and Services Development work package DTN Focus Group team has worked on several aspects of DTN deployment. A survey was run at the end of 2019 among several European NRENs to find out which was their current status and the best possible approach for them in this area. As the survey concluded that both technical and non-technical issues needed to be considered, the project has worked in both these directions.

From the technical point of view, the DTN Focus Group team studied and tested several high-performance software tools over the GÉANT Testbeds Service (GTS), using VMs with several setups, bare metal servers without virtualisation and virtualised environments created using Docker, with 10Gbps links over different distances across Europe.

The first perhaps unsurprising conclusion of these tests is that more powerful resources always lead to better results. But the team also observed that some software tools seem to behave better in bare metal servers than in a virtualised environment, whereas the results for other tools in bare metal servers and virtual machines are quite similar. Another finding is that, although the results are usually better when bare metal servers are used, there is only a minor difference for most of the tools compared to the tests results achieved using virtualised environments, at least for tests run over 10Gbps links. This implies that NRENs or organisations for whom 10Gbps transfers are sufficient (moving 100TB in one day is roughly a 10Gbps throughput) can deploy a virtual solution, but they do then need to be aware of which tools run better in such an environment. The team has created the necessary scripts to make containers with predefined data transfer tools available to download from the DTN wiki, to promote the use of predefined virtual containers that can easily be installed by institutions and NRENs offering DTN services.

Regarding the non-technical issues, the dissemination of best practices and end-user engagement were found to be particularly relevant. To help address these the project has created and populated a wiki with general DTN information, related use cases, applied methodologies, hardware and software systems and tools, as well as the results of the various tests run with the transfer tools and the survey results.

Appendix A DTN DSL Structure Example

The DTN DSL structure used for the creation of Hamburg-Paris BMS servers in DTN (similar DSL structure[5] used for all the setups) is as follows:

```
HamburgParis {
 id = "HamburgParis"
 description = "DTN 2xBMS TEST Paris-Hamburg"


 baremetalserver {
  id = "DTNHamburg"
  x = -351
  y = -244
  image = "ubuntu-18.04-server.iso"
  port { id = "p1" }
  location = "ham"
 }


 baremetalserver {
  id = "DTNParis"
  x = 39
  y = -249
  image = "ubuntu-18.04-server.iso"
  port { id = "p1" }
  location = "par"
 }


 link {
  id = "DTNHamburgDTNParisnum1"
  port { id = "src" }
```

[5] The id, adjacency and description are changed accordingly

```
  port { id = "dst" }
 }

 adjacency DTNHamburg.p1, DTNHamburgDTNParisnum1.src
 adjacency DTNParis.p1, DTNHamburgDTNParisnum1.dst
}
```

The following are some tuning parameters in Linux that can be used in the GTS DTN setup of the Server/Client along with explanations:

```
# allow testing with buffers up to 64MB

net.core.rmem_max = 67108864

net.core.wmem_max = 67108864

#

# increase Linux autotuning TCP buffer limit t 32MB

net.ipv4.tcp_rmem = 4096 87380 33554432

net.ipv4.tcp_wmem = 4096 65536 33554432

#

# recommended default congestion control is htcp

net.ipv4.tcp_congestion_control=htcp

#

# recommended for hosts with jumbo frames enabled

net.ipv4.tcp_mtu_probing=1

#

# recommended for CentOS7+/Debian8+ hosts

net.core.default_qdisc = fq

ifconfig ethXXX txqueuelen 10000

ifconfig ethXXX mtu 8986
```

References

[18STF]	https://wiki.geant.org/display/APM/18th+STF+-+Copenhagen%2C+October+2019 (eduGAIN access needed)
[18STF-DTN]	https://wiki.geant.org/download/attachments/126977976/GN4-3_WP6_T1_DTN_STF_Oct_2019-v10.pdf?api=v2 (eduGAIN access needed)
[AENEAS]	https://www.aeneas2020.eu/about-aeneas/project-overview/
[AENEAS-D41]	https://www.aeneas2020.eu/documents/AENEAS-D4.1.pdf : "Copyright © Members of the AENEAS Collaboration, 2017. See www.aeneas2020.eu for details of the AENEAS project and the collaboration"
[BBCP]	https://www.slac.stanford.edu/~abh/bbcp/, https://github.com/slaclab/bbcp
[BDEXPRESS]	https://bigdataexpress.fnal.gov/
[CONT-VM]	https://www.weave.works/assets/images/bltb6200bc085503718/containers-vs-virtual-machines.jpg https://www.startusingdocker.tech/
[DOCKER]	https://www.docker.com/
[DOCKER_CTOP]	https://computingforgeeks.com/top-command-for-container-metrics/
[DOCKER_ESNET_GITHUB]	https://github.com/esnet/docker
[DOCKER_INST_CENTOS1]	https://computingforgeeks.com/install-docker-and-docker-compose-on-rhel-8-centos-8/
[DOCKER_INST_CENTOS2]	https://www.centlinux.com/2019/12/install-docker-ce-docker-compose-centos-8.html
[DOCKER_INST_UBUNTU]	https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04
[DOCKER_LCX]	https://docs.docker.com/engine/faq/#what-does-docker-technology-add-to-just-plain-lxc
[**DOCKER_LCX1**]	https://linuxcontainers.org/
[DOCKER-OVW]	https://docs.docker.com/get-started/overview/
[DOCKER_RES_CONS]	https://docs.docker.com/config/containers/resource_constraints/
[DTN-CERN]	https://indico.cern.ch/event/952623/contributions/4023107/attachments/2111105/3551134/WLCG-20200929-PRACE-HPC-network-challenges.pdf
[DTN-Info]	https://events.geant.org/e/DTN/
[DTN-JISC]	https://www.slideshare.net/JISC/data-transfer-experiments-over-100g-paths
[DTN-LHC]	https://indico.cern.ch/event/828520/contributions/3677232/attachments/1968653/3274246/CERN_LHCONE_SKA-AENEAS_v1.pdf
[DTN-PMV]	https://wiki.geant.org/download/attachments/84476097/AENEAS-DTN_SIG-PMV_Copenhagen_29Nov17_v2.pptx?version=1&modificationDate=1511950994932&api=v2

[DTN-Wiki]	https://wiki.geant.org/display/DTN/
[EPYC]	https://developer.amd.com/resources/epyc-resources/epyc-tuning-guides/
[ESnet]	ESnet, Linux Tuning - https://fasterdata.es.net/host-tuning/linux/
[FASTERDATA]	http://fasterdata.es.net/science-dmz/DTN/reference-implementation/
[FASTERDATA2]	https://fasterdata.es.net/science-dmz/DTN/tuning/
[FDT]	https://github.com/fast-data-transfer/fdt
[FDT2]	http://monalisa.cern.ch/FDT/
[FTS-CERN]	https://information-technology.web.cern.ch/services/file-transfer#:~:text=The%20GRID%20Data%20Transfer%20Service,will%20retry%20if%20this%20fails.
[GLOBUS]	https://www.globus.org/
[Globus Connect]	https://www.globus.org/globus-connect
[Globus-FAQ]	Globus FAQs:Transfer and Sharing - https://docs.globus.org/faq/transfer-sharing/
[GridFTP]	https://www.globus.org/ and https://github.com/globus/globus-toolkit
[GTS]	https://www.geant.org/Services/Connectivity_and_network/GTS/Pages/GTS-Overview.aspx
[GTS-IMAGE]	https://wiki.geant.org/display/gn43wp7/GTS+User+Guide+v7.0-internal%2C++rev.02
[GTS-SVC]	https://gts.geant.org/
[NUMA]	**N**UMA Siloing in the FreeBSD Network Stack - https://people.freebsd.org/~gallatin/talks/euro2019.pdf
[OGF]	https://www.ogf.org/ogf/doku.php
[perfSONAR]	https://www.perfsonar.net/
[perfSONAR_info]	https://www.geant.org/Services/Connectivity_and_network/Pages/perfSONAR.aspx
[PRP]	https://pacificresearchplatform.org/
[PMW]	https://wiki.geant.org/display/PUB/Performance+Management+Workshop
[SCIENCEDMZ1]	https://www.es.net/assets/pubs_presos/sc13sciDMZ-final.pdf
[SCIENCEDMZ2]	https://fasterdata.es.net/science-dmz/
[SKA]	https://www.skatelescope.org/
[socket]	Linux manual page, socket - https://man7.org/linux/man-pages/man2/socket.2.html
[tcp]	Linux manual page, tcp - https://man7.org/linux/man-pages/man7/tcp.7.html
[TTC]	https://www.switch.ch/network/tools/tcp_throughput/
[VIDEO]	https://www.youtube.com/watch?v=TH1ePbWOJKs&feature=youtu.be
[WLCG]	https://wlcg.web.cern.ch/
[XDD]	https://github.com/bws/xdd
[XRootD]	https://xrootd.slac.stanford.edu/
[ZFS]	https://www.freebsd.org/doc/handbook/zfs.html

Glossary

ACL	Access Control List
AENEAS	Advanced European Network of E-infrastructures for Astronomy with the SKA
BBCP	BaBar CoPy
BIOS	Basic Input/Output System
BMS	Bare Metal Server
BBR	Bottleneck Bandwidth and Round-trip time
CPU	Central Processing Unit
DD	Copy and Convert (called DD as CC was in use)
DMZ	DeMilitarized Zone
DTN	Data Transfer Node
DSL	Domain Specific Language
ESDC	European Science Data Centre
ESRC	European SKA Regional Centre
FDT	Fast Data Transfer
FTP	File Transfer Protocol
GTS	GÉANT Testbeds Service
HPC	High Performance Computing
IETF	Internet Engineering Task Force, the premier Internet standards body
I/O	Input/Output
LAN	Local Area Network
LCX	Linux Containers
NIC	Network Interface Card
NIO	Non-bloking I/O
NUMA	Non-Uniform Memory Access
NVMe	Non-Volatile Memory Express
OGF	Open Grid Forum
PaaS	Platform as a Service
PCIe	Peripheral Component Interconnect express
RAID	Redundant Array of Independent Disks
RTT	Round trip time
SAN	Storage Area Network
SAS	Serial Attached SCSI
SATA	Serial Advanced Technology Attachment
SCSI	Small Computer System Interface
SDN	Software Developed Networks
SKA	Square Kilometre Array
SSD	Solid State Drive
TCP	Transmission Control Protocol
VM	Virtual Machine
XDD	The eXtreme DD Toolset