*Astronomical Data Analysis Software and Systems XIX*
*ASP Conference Series, Vol. 434, © 2010*
*Y. Mizumoto, K.-I. Morita, and M. Ohishi, eds.*

# DTS: The NOAO Data Transport System

M. Fitzpatrick

*National Optical Astronomy Observatory, Tucson, AZ, USA*

**Abstract.** The Data Transport System (DTS) provides automated, reliable, high-throughput data transfer between the telescopes, archives and pipeline processing systems used by the NOAO centers in the northern and southern hemispheres. DTS uses an XML-RPC architecture to eliminate the need for persistent connections between the sites, allowing each site to provide or consume services within the network only as needed. The RPC architecture also permits remote control and monitoring of each system, and for client applications to be language-independent (e.g. a web interface to display transfer status or a task to queue data which is more tightly coupled with the acquisition system being used).

## 1 Introduction

The DTS project was initiated specifically to meet the data transport needs of the Dark Energy Survey (DES) scheduled to begin operating at CTIO in 2010. The Dark Energy Camera (DECam) used by the survey is a 500 Mpixel mosaic of 62 CCDs, producing roughly 500 MB per exposure (compressed). The DES observing cadence is expected to generate some 200-300 GB/night that must be moved to the pipeline facility at NCSA within 18 hrs. When not being used for the DES, the camera will be available for general use on the Blanco 4 m and all raw data is to be archived at NOAO/Tucson archive center.

Meeting the requirements of DES is challenging because of the limited bandwidth currently available out of Chile and the need to route data differently depending on whether it was acquired as part of the DES by a generic observer. Although the bandwidth and connectivity to Chile are expected to improve by the time the survey begins, traditional transport methods (*rsync, SRB*, etc) don't have sufficient throughput to meet the timing requirements, or the infrastructure needed to manage the complexity of the operational environment. Additionally, we plan to migrate all NOAO data transport to DTS in the future, meaning the solution has to be configurable for a wide variety of data sizes and observing modes (i.e. the optimal transfer method for a single 500 MB image may not be ideal for smaller files such as spectra).

## 2 Architecture

The DTS uses an XML-RPC architecture to provide a suite of basic services at each site without requiring persistent connections within the system. The central component is a DTS Daemon (*dtsd*) process responsible for managing the data queues and responding to client service requests (which, in most cases, will be other *dtsd* tasks in the system). The *dtsd* can be configured to run as a standard unix service, providing continuous availability and automated recovery in the event of a system crash. The *dtsd* is a highly multi-threaded application capable of managing any number of transport queues and client connections. Because of its service architecture, the DTS can be easily monitored or managed by remote clients, simplifying operations of the distributed system from a central location.

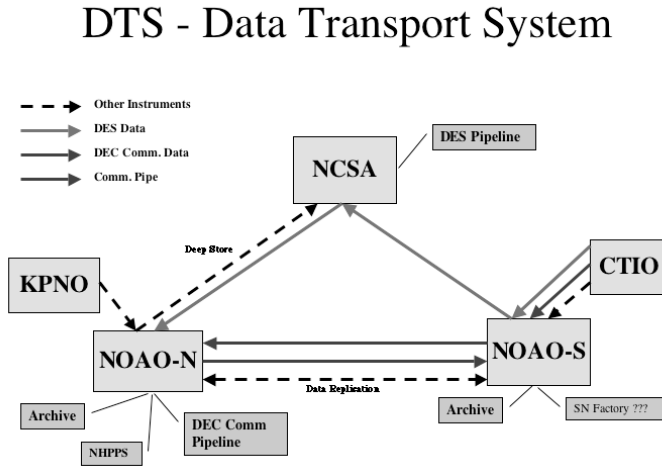


Figure 1. Illustration of the various data paths that must be supported by the DTS. As data make there way through the system, different actions are required at each stop (e.g. archive ingest, pipeline delivery etc) before moving on to the next queue. The DTS uses external processes to manage these "delivery" actions rather than implicit knowledge of what is required at each location.

Data are introduced to the DTS system via a specialized client called *dtsq* which is responsible for moving the data from the local machine to a remote machine hosting the DTS. At the telescope, a *dtsq* would be triggered at the end of each exposure, moving the data from each dome to a centralized 'mountain cache' where the DTS would then queue the data for transport to a downtown facility and beyond. This replaces the current 'lpd' based scheme, but still provides the same quick response and recoverable queueing functionality.

A simple *dtsmon* application is available on a reserved port, allowing all system messages to be centrally monitored. A command-line tool called the *dtsh* provides a means to test system services as well as a scripting capability to automate actions (e.g. generate transfer reports, manage queues, etc).

Transfer queues are configured individually and operate in a normal FIFO fashion, as a scheduled queue (triggered at fixed intervals), or as a priority queue for time-sensitive delivery. Priority queues block other transfers until empty to make all the network resources available for transport and won't normally be configured for regular use.

DTS can be configured as a system of federated sites and transport queues, but it can also be used for personal use by simply starting a *dtsd* on a remote machine and using the dtsh application to transfer a file. Because the system provides basic filesystem services, recursive transfers of directories and remote updates similar to rsync are also possible.

## 3 Transport Methods

Bulk data transport is handled as an RPC call to the destination machine. Four different transport calls are provided in order to control the direction of transport and the client-server relationship between the parties. This allows for successful transport when one of the ends is behind a firewall and only the DTS command port is available, and in cases where multiple clients may need to connect to a single DTS such as in the observatory environment.

Transport modes include:

| | |
|---|---|
| `PUSH` | file is moved from A to B, where B supplies the server sockets |
| `PULL` | file is moved from B to A, where B supplies the server sockets |
| `GIVE` | file is moved from A to B, where A supplies the server sockets |
| `TAKE` | file is moved from B to A, where A supplies the server sockets |

A transfer command can be sent to either host A or B with the appropriate model to achieve the identical results. Because of the RPC nature of the system, third-party transfers are also possible, allowing an operator to manually move data between remote sites as needed.

Data transport itself is independent of these transfer models. The default transport is done using parallel TCP/IP sockets but can be easily configured to use another method if desired (e.g. *rsync* can be used to do the actual transport while still using the DTS infrastructure to control the queues, delivery actions, and so on). Using the parallel socket transport, a file is divided amongst each of the sockets and transferred on separate threads called 'stripes'.

Data integrity is guaranteed via checksum matching before and after each transfer, however we permit a number of checksum policies that do checking on individual buffer transfers, larger stripes of the data on each thread, or the entire file. In the event of a failed checksum we may only need to retransmit a portion of the file and not the entire 500 MB or more. The transport method, the number of sockets used, various buffer and TCP window sizes etc. can all be tuned to the individual needs of a particular data path.

## 4 DTS Performance

Measuring network performance is a tricky business, especially when the results depend on the time the test was conducted and what other network loads are present. For this reason, we conducted sample tests at various times during the day over an extended period to arrive at average relative values rather than absolute numbers. Figure-2 shows the results of DTS transfers compared to other methods and the relative gains that can be achieved. Because we were using deployed machines, it was not always possible to install the software needed to completely test other transport methods such as *GridFTP*, *FDT*, or a variety of UDP transport protocols in common use in the high-energy physics community.

The chart highlights the effect that network latency can have on the transport protocol chosen, and thus the need to be able to configure the system for the various network environments we'll encounter. The blue bars represent the movement of data from CTIO to Tucson over a 45 Mbps connection having a network RTT of 300 ms, the orange bars represent the KPNO to Tucson connection over a similar 45 Mbps bandwidth link but with a RTT of only 3 ms, and the yellow bars represent a connection between two machines connected by a local gigabit ethernet. Since rsync is a very common general-purpose means of moving data, we've normalized the results to the performance of rsync over those connections. DTS testing was done using different numbers of processing threads, however it should be noted that these results can be affected by the speed and number of CPUs/cores available and in theory a more capa-

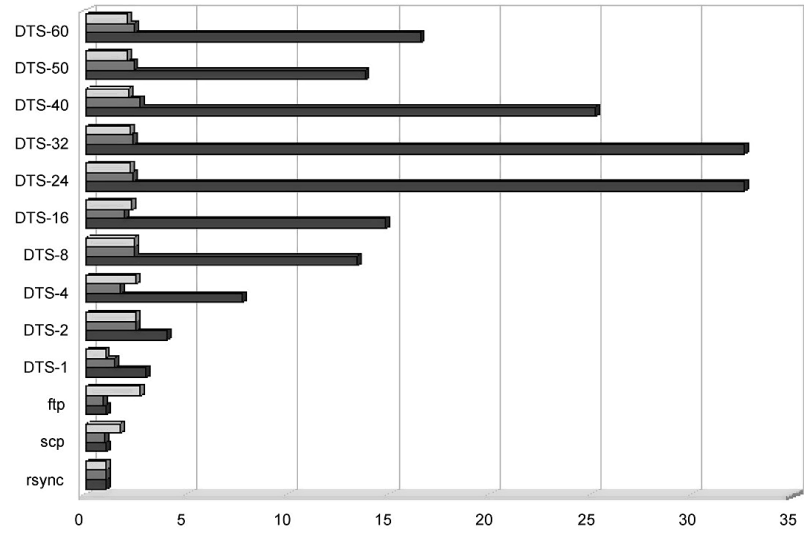


Figure 2. Performance of DTS relative to other methods. We normalize all results to the performance of rsync over that connection.

ble machine should be able to effectively handle more threads (either on a dedicated transport queue, or multiple queues being maintained simultaneously).

The results show that we can gain as much as a 32X increase in speed using DTS operating on 24-30 sockets, but note we only see such a gain in performance over hi-latency connections. The performance increase is much less dramatic over low-latency connections of similar bandwidth, however we don't require as many processing threads to achieve peak performance. The FDT streaming protocol in use by CERN for the LHC project was the only other method we were able to test directly and confirmed that DTS was able to achieve comparable throughput performance.

In the expected deployment (shown in Figure-1), we are able to configure the system appropriately for each connection path. We expect that as we gain experience with the system we will be able to better to tune the system for optimal performance.

## 5 Future Work

The DTS is still in development and undergoing preliminary testing for use in the Dark Energy Survey data acquisition system. We plan to make the software publicly available throughout its development. Near-term work will concentrate on functionality needed to replace the existing transport infrastructure within NOAO and tuning needed to optimize available network bandwidth.