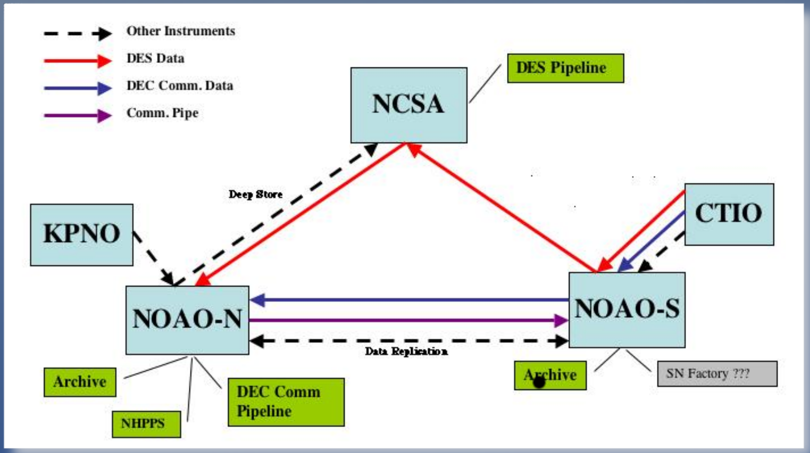
DTS Status and Review

DTS Background

DTS is designed for DECam, but will be used throughout NOAO observing systems:

- Must make better use of network
- Must allow data to be routed based on DES/Community use
- Must interface with both NOAO and DECam systems
- Must be configurable for data rates and sizes other than DES
- Must operate in variety of network environments



DTS Architecture

- XML-RPC based
  - Provides basic services without requiring persistent connections
  - Can be configured as unix service for automatic recovery in case of crash
  - Allows for remote monitoring and management
  - Client applications are language independent
  - Central DTS process providing services at each site
  - Highly multi-threaded and configurable

DTS Components

- DTSD – DTS Daemon
  - Provides all DTS services on the machine (assumed one per 'site')
  - Responsible for managing transport queues (one Manager thread for each queue)
  - Requires command port be open to firewall
  - Sandboxed filesystem view for security
  - Can be run as *xinetd* service or run entirely in user-space
  - Set/Get methods permit remote management

DTS Components

- DTSQ – DTS Queueing Agent
  - Queues data for ingestion into the DTS system
  - Provides quick response so it won't block caller
  - Acts as a *dtsd* to provide its own transport methods to the DTS, i.e. SISP requires no other components be installed
  - Logs all requests
  - Verifies DTS status before allowing transfer
  - Permits recovery of failed requests for later re-queueing
  - Leaves a “token” file with details of transfer on success

DTS Components

- DTSD – DTS Daemon
- DTSQ – DTS Queueing Agent
- DTSH – DTS shell and command-line tool
- DTSMON – DTS monitoring application

DTS Components

- DTSH – DTS shell and command-line tool
  - Allows direct interaction with DTS site
  - Provides scripting capability for DTS commands
- DTSMON – DTS monitoring application
  - Simple monitoring application, echos messages generated by remote DTS components
  - Runs on a reserved port
  - Meant for operations use, not required for

SISPI-DTS Interfaces

At the telescope: IB system invokes *dtsq*

```
dtsq -q des -f /path/image.fits
```

- The '-q' names the transport queue to be used
- The '-f' forks the command after verifying DTS will accept the file, transfer happens in child
- IB gets immediate OK/ERR response
- On success, details of transfer (time of request, time of completion, comments, etc) left on calling system
- On error of transfer, request is logged for later recovery

SISPI-DTS Interfaces

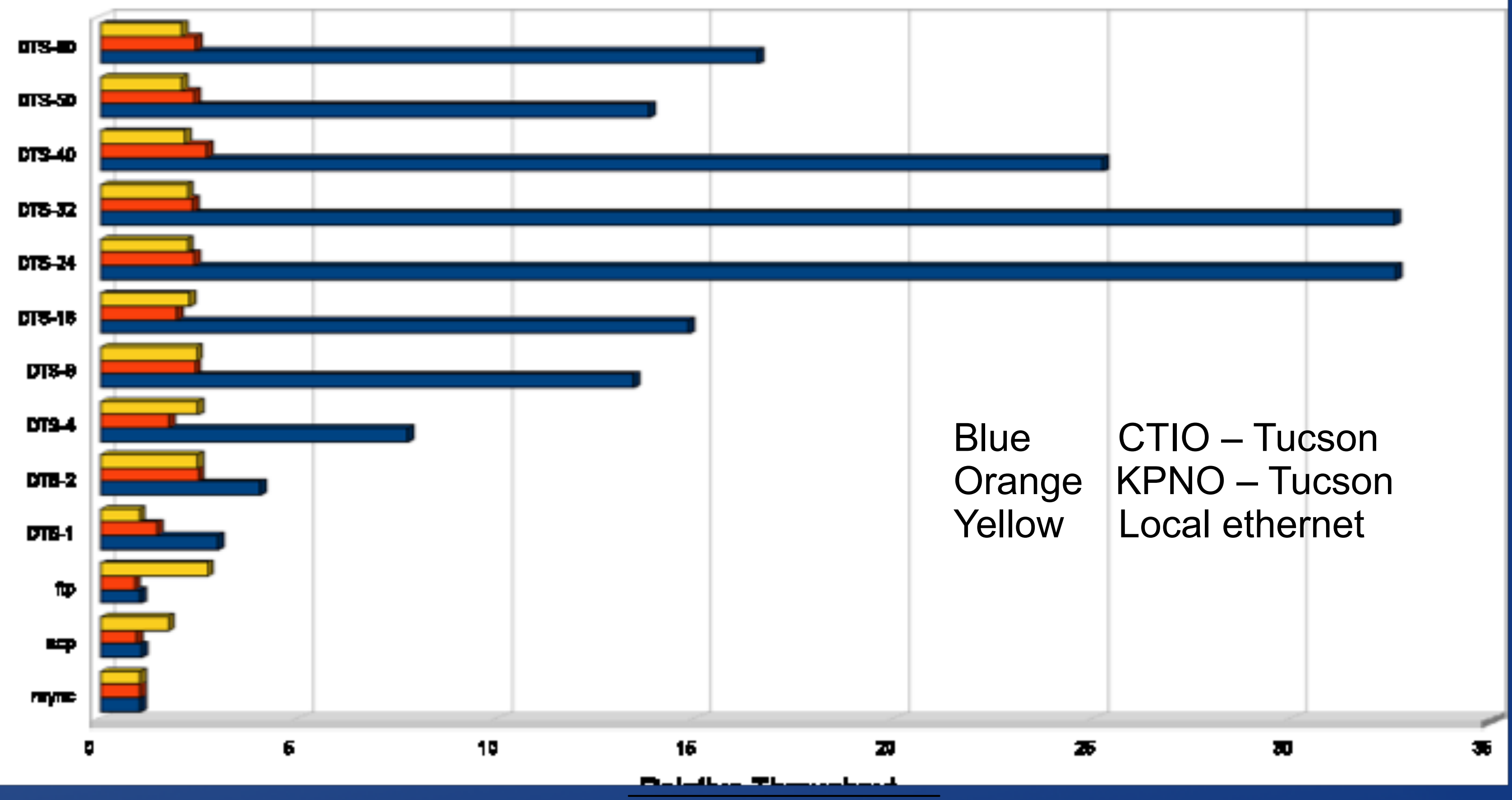
At the DES Pipeline: DTS invokes an external 'delivery' application. This application

- takes a single argument: the local path to the file
- Is expected to make a private copy of the file, moving it out of the DTS sandbox where it can be manipulated
- Must be provided by DES since only they know what they want to do with the image once it arrives

DTS Status

- Still in hi-priority development
- All components implemented (some partially)
- Provides point-to-point data transport, queue management and data routing still in development
- Snapshot release available for IB interface testing (in progress)
- Scripting capability can be used to support Nov T&E run without configuring full system

DTS Relative Throughput



Blue	CTIO – Tucson
Orange	KPNO – Tucson
Yellow	Local ethernet

Transfer throughput normalized to rsync over that connection

November Support

- Using dtsh we can do simple P2P transfer to one or more sites, require only dtsh at the telescope and dtsd running remotely.
- For example, could create script file 'dts_xfer' such as

```
#!/usr/local/bin/dtsh -f

push -t <IP at Fermi> $1 &
give -t <IP in Brazil> $1 &
```

This would be invoked by the observing environment simply as

```
dts_xfer /path/image.fits
```

And background transfers would begin of the file to a remote “drop directory. More requirements are needed if anything more sophisticated in needed.