NAME

dtscp − direct copy of files between DTS nodes

SYNOPSIS

dtscp [<*opts*>] <src> <dest>

OPTIONS

The *dtscp* application accepts the following options:

−h Print a help summary to the terminal and exit. No processing is done following this flag.

-v verbose output flag changes the level of output that will be displayed to the user. Useful for troubleshooting. The default is off.

-q quiet flag is the opposite of verbose very little output will be shown as the program is run. The default is off.

-version
Print build version date

-a <alias>
set contact alias //not completed

-c [fname]
specify a DTS configuration file that DTSCP will read and use certain options from. The default is to read from $HOME/.dts_config.

-f <file>
transfer files listed in <file> which can be in various formats. Each line of paths need to be seperated by an enter character. If this flag isn't specified DTSCP wont read any files.

Examples:

Local absolute path: /foo/bar

Local relative path: ../file.fits

Remote sandbox path: srchost:/directory/

-p use PULL mode for transfer. The default transfer method if this PULL mode isn't specified is PUSH for transfers. Try using if having a problem with file transfers.

-q <qname>
Deliver the files or directory to the specific queue on the dest (destination machine). By default DTSCP will deliver to a non-queue specific directory.

-u use UDT transfer mode which is talked about specifically later. The default if UDT isn't used is parallel TCP sockets.

-H <port>
set hi transfer port. This applies locally and remotely for both the source and destination.

!!WARNING!! If flag N threads is specificed (H port - L port) must be equal or larger than N.

-L <port>
set low transfer port. This applies locally and remotely for both the source and destination.

!!WARNING!! If flag N threads is specificed (H port - L port) must be equal or larger than N.

-N <N>
set number of threads (and ports) that are needed to transfer the file. Note: DTS will wait until all threads are connected before transfering any files. The default is 1 thread.

!!WARNING!! If flag N threads is specificed (H port - L port) must be equal or larger than N.

-P <port>
set (RPC) command port this is the port DTSCP will attempt to connect to a DTSD daemon on. If remote -> remote transfer is used both remote nodes will use this port. The default value is 3000.

-R <rate in Mbps>
set UDT congestion control rate. A rough estimate for this would be 90% of your tested iperf udp connection rate. May take a few tries get optimal speeds. Just because 90 is used which is 90 Mbps doesn't mean you'll necessarily get 90Mbps in your transfer speed. The default value is off and will use built in congestion control.

-T print transfer time information

+d debug flag

DESCRIPTION

The *dtscp* task is used to transfer files and/or directories between hosts in a DTS network.

The <dest> arguments are of the form:

[<node> ':'] <file_template>

where a <node> name, if present, indicates the fully qualified domain name (or IP address) of a machine running a DTSD (DTS Daemon) that will serve as the source or destination machine. The <file_template> may contain metacharacters (e.g. a '*') that will be expanded on the source machine to create a list of files to transfer. Arguments may be either the names of files or directories; Directories will be transferred recursively automatically.

When copying a source file to a target file which already exists, DTSCP will replace the contents of the target file. If DTSCP fails to complete the transfer the user will be notified on why the transfer could not be completed (to the best of the client's ability).

DTSCP is designed to mimic the file operation and path handling of SCP. Some of the same ways of indicating a file transfer or a directory transfer that are used in SCP should be simular in DTSCP.

CONFIGURATION FILES

DTSCP is defaulted to use the $HOME/.dts_config file on the local machine which can be overwridden by the using the -c flag specified above. It is possible to set configuration file, but any flags that are set in the command line supersede the settings for the configuration file in this instance. DTSCP Config precedence is laid out as follows:

specified flags > specified configuration file > default configuration file

If no default configuration file exists and no configuration file is specified then ALL critical specified flags must be set otherwise an error will be prompted. Keep in mind that nodes in this case must be specified as ip addresses. or hostnames unless aliased with the -a flag.

The following are the **CRITICAL** flags that need to be set if no configuration file is present: High port, low port and RPC command port.

EXAMPLE lines read from .dts_config by DTSCP (In a real dts configuration file there will be more lines that aren't used by DTSCP.)

```
dts      #START OF DTS SCRIPT FILE
   name      denali  # This is the alias or node name for a local or remote machine
   host      denali.tuc.noao.edu                          # hostname or address for the machine
   port      8832  # RPC COMMAND PORT which is for communication port for all DTS
   loPort    7535  # LOPORT is the lowest port available for node file transfer
   hiPort    7600  # HIPORT is the highest port available for node file transfer

    queue   #START OF DTS QUEUE THIS IS THE QUEUE ON THE STATED NODE ABOVE (LOCAL TO THE NODE A
        name           test # specific queue name
        method          dts  # default transfer method dts which i or UDT read TRANSPORT METHODS below.
        nthreads        1    # No. of outbound threads for file transfer
        udt_rate        30   # UDT Mbps congestion attempted transfer rate
        mode            push # suggested mode transfer for this queue. //Not yet used.
        port           3023 # BASE transfer port for the queue (has to be in range of hiPort and loPort)
        keepalive       yes  # keep connection open?  [NYI]
```

TRANSPORT METHODS

DTSCP has support for multiple transport methods. The two transport methods that are currently implemented are TCP parallel sockets and UDT.

TCP Parallel Sockets

TCP parallel sockets stripe the file into small pieces dictated by the number of threads that are specified. Currently 1 socket per thread is being used. Hence if 50 threads are specified 50 sockets will attempt to connect to do a transfer. The transfer will not commence until all of the sockets are connected. Since each socket needs it's own port, it is crucial to meet the requirements of (HIPORT - LOWPORT >= NTHREADS). Otherwise the transfer may fail because not all sockets/threads had connected. This transport method can be useful for networks with a low RTT and low packet loss rate. (Congestion Window based method). Keep in mind that the CPU utilization for TCP is less than that for UDT.

UDT

The other transport method is called UDT which given the correct conditions can provide greater performance over TCP parallel sockets especially when the RTT is over 100MS (see http://udt.sourceforge.net/doc/gridnets-v3.ppt).

DTSCP's UDT has a rate congestion control option included based on the UDPBlast protocol (see udt.sourceforge.net/doc/sc05-v3.ppt slide 11). By specifying the rate for UDT it will calculate a packet send period which will in turn affect the transfer performance. The congestion window is also set to a large size on this mode which means itcan reach link capacity VERY QUICKLY. (Rate based method).

If the rate is not set UDT will use the default built in UDT congestion control. It is also a rate based method but estimates the bandwidth using packet pairs. It takes 7.5 seconds using this algorithm to reach 90% of the link capacity. Which means sending files that take longer than 7.5 seconds will be more feasable using this algorithm.

Push & Pull Models

The push and pull methods are implemented to provide a simple way to workaround strict firewall configurations. For example a node named node1

Push and pull are opposites of each other in the sense of which node opens the client sockets and which node opens the server sockets. Push (open server sockets on the source node). Pull (open server sockets on the destination node).

// Refer to the diagram in SPIE2010_DTS.pdf page 7 of 13.

RETURN STATUS

On exit the **dtscp** dtscp will return a zero indicating success, or a one indicating an error.

EXAMPLES

1) Different ways of transfering files

```
Transfer from a file list containing local relative paths, local
absolute paths or remote sandbox paths.

% dtscp -f xfer.list  dest.edu:

xfer.list contains:
hello.fits
/home/dir/hello.fits
src.edu:/hello.fits

Using local wildcards and remote wildcards
% dtscp *.fits dest.edu:
% dtscp src.edu:*.fits dest.edu:

Multiple source files/hosts.
% dtscp foo.fits foo2.fits src.edu: dest.edu:
```

2) Transfer all FITS files from the local machine to the DTS running on machine 'dest.edu' using the default configuration file

```
% dtscp *.fits dest.edu:             # to DTS default dir
% dtscp *.fits dest.edu:/data        # to system /data dir
% dtscp -p *.fits dest.edu:                   # to DTS default dir using pull method
        or
% ls *.fits > xfer.list
% dtscp -f xfer.list dest.edu:        # works with full path files as well.
        or
% dtscp -u -R 100 *.fits dest.edu:  # UDT congestion control rate of 100 Mbps (tries for 100Mbps)
        or
% dtscp -N 20 *.fits dest.edu:       # TCP over 20 sockets and 20 threads.
```

3) Transfer a single file to the remote machine

```
% dtscp lol.fits dest.edu:                      # local relative lol.fits to dest.edu sandbox root
% dtscp /home/foo/lol.fits dest.edu:            # local absolute path to dest.edu root
% dtscp /home/foo/lol.fits dest.edu:/mypath/            # local absolute path to dest.edu sandbox path
```

3) Transfer a local file or directory to a remote queue's delivery directory:

```
% dtscp -q DES foo.fits dest.edu: #local relative file to DES queue on dest.edu
% dtscp -q travis /hello/ dest.edu:  local absolute directory hello to queue travis on dest.edu
% dtscp -q travis ./hello/ dest.edu: #copy relative directory hello to queue travis on dest.edu
% dtscp -p -q travis ./hello/ dest.edu: #copy relative directory hello to dest.edu using pull
% dtscp -q travis -f myfiles.list dest.edu:  #copy files from local machine to travis queue on dest.edu
```

REMEMBER *myfiles.list* can be structured with absolute paths or relative paths that DTSCP can access.

4) Retrieve a file from a remote machine and store locally:

```
% dtscp src.edu:/lol.fits .            # copy from remote sandboxed path to the local relative path
% dtscp src.edu:/lol.fits /mydata/       # copy from remote sandboxed path to the local absolute path
```

5) Retrieve FITS files on machine 'src.edu' to the current local directory:

```
% dtscp src.edu:data/*.fits .          # copy *.fits from remote sandbox path to the local relative path
% dtscp src.edu:/data/*.fits /myfits/   # copy *.fits from remote sandbox path to the local absolute path
```

6) Transfer all files in the /data directory to a remote node:

```
% dtscp  /data dest.edu:               # transfer from local relative path to remote sandbox path
```

7) Retrieve all files from the remote node and place in the /data directory:

```
% dtscp dest.edu: /data:   #transfer from remote sandbox root to local absolute path
```

8) Transfer all files in the /data directory from remote node to different
remote node:

```
% dtscp src.edu:/data dest.edu:
```

9) Transfer all files from a remote node to a different remote queue.

```
% dtscp -q DES src.edu:  dest.edu:                          # transfers to bob's queue DES
% dtscp -v -q DES src.edu:  dest.edu:          # verbose

% dtscp -H 3001 -L 3000 -P 3005 src.edu: dest.edu:
            sSet HI transfer port for source and destination
            #set low transfer port for source and destination
            #set RPC command port for source and destination
```

10) Transfer files from a remote node to a different remote node:

```
% dtscp -p src.edu:/lol.fits dest.edu: #using pull always uses remote sandbox paths
% dtscp src.edu:lol.fits dest.edu: #using push always uses remote sandbox path
% dtscp src.edu:/home/foo/tmp/dts.foo/lol.fits dest.edu: #full absolute path always uses remote sandbox path



% dtscp src.edu:/home3/ dest.edu:/home2    #will copy home3 into home2 on remote node.
```

10) Multiple host submission to queue or file transfer.

```
% dtscp -q DES src.edu:/funny.txt src2.edu:/hello.txt dest.edu: #from two remote sources to a remote queue
# transfer file from tukana and dest.edu to tukana
% dtscp src.edu:/funny.txt src2.edu:/hello.txt dest.edu:
#NOTE if a file and directory are in the source then the destination path will be considered a directory.
```

BUGS

No known bugs with this release.

Revision History

Feb 2013 - First public release

Author

Travis Semple and Mike Fitzpatrick (NOAO), Feb 2013

SEE ALSO

dtsq, dtsh, dtsd