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

-q quiet flag

-version
print build version

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

-c [fname]
set config file

-f <file>
transfer files listed in <file>

-n no-op flag

-p use PULL mode for transfer

-q <qname>
deliver to specified queue

-r recursive transfer (of directories)

-t [name]
command target host name

-u use UDT transfer mode

-H <port>
set hi transfer port

-L <port>
set low transfer port

-N <N>
set number of threads

-P <port>
set (RPC) command port

-R <rate>
set UDT transfer rate (Mbps)

-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 <src> or <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 may be transferred recursively if the '-r' flag is present, otherwise only the toplevel contents of a directory will be transferred. //scp doesn't foll ow this

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 overwritten 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 configuration of .dts_config read by DTSCP:

```
dts
   name     denali                  # the ALIAS
   host     denali.tuc.noao.edu             # host address for the ALIAS
   port     8832                  # RPC COMMAND PORT
   loPort   7535                   # LOPORT
   hiPort   7600                   # HIPORT

queue
   name          test          # queue name
   method         dts           # default transfer method dts or UDT
   nthreads       1            # No. of outbound threads for dts
   udt_rate                   # UDT transfer rate
   keepalive      yes           # keep connection open?  [NYI]
```

TRANSPORT METHODS

TCP Parallel Sockets

DTSCP has support for multiple transport methods. The two transport methods that are currently implemented are TCP parallel sockets and UDT. TCP parallel sockets stripe the file into small pieces depending on the number of threads that are declared. 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 off of the UDPBlast protocol. 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 it can reach link capacity VERY QUICKLY. (Rate based method).

If the rate is not set UDT will use the default built in UDT congestion control which is again stated on this page (http://udt.sourceforge.net/doc/gridnets-v3.ppt) 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.

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 with a strict firewall may be able to open outgoing socket connections but not able to open any incoming socket connections. Our second node, node2 doesn't have a strict firewall.

If one was to transfer files from source node2 -> destination node1 due to the strict firewall on node1 we'd open the sockets on node2 which means the transport mode is a push.

If one was to transfer files from source node1 -> destination node2 due to the strict firewall on node1 we'd open the sockets on node2 still which means the transport mode is a pull.

Push and pull determine which node opens the client sockets and which opens the server sockets. Push (open server sockets on the source node). Pull (open server sockets on the destination node).

// Referring to your diagram in SPIE2010_DTS.pdf page 7 of 13.

RENDEZVOUS MODE (UDT ONLY):

This is used to bypass firewalls, you provide two ports when UDT is ran two client connections are made. Useful for strict firewalls or punching through NAT. **//Not implemented yet**

RETURN STATUS

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

EXAMPLES

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

```
% dtscp *.fits fred.edu:            # to DTS default dir
% dtscp *.fits fred.edu:/data       # to system /data dir
% dtscp -p *.fits fred.edu:                # to DTS default dir using pull method
      or
```

```
% ls *.fits > xfer.list
% dtscp -f xfer.list fred.edu:          # works with full path files as well.
        or
% dtscp -c dts.cfg *.fits fred.edu:  # custom config file
        or
% dtscp -u -R 100 *.fits fred.edu:  # UDT rate of 100 Mbps
        or
% dtscp -N 20 *.fits fred.edu:        # TCP over 20 sockets
 or

# custom config file with UDT and rate of 100 Mbps
% dtscp -c dts.cfg -u -R 100 *.fits fred.edu:

 or

# reads alias from config transfers to the DTS default dir will
# only work if config file has entry for the 'fred' dts node

% dtscp *.fits fred:

# Transfer a single file to the remote machine

% dtscp lol.fits tukana:                    # lol.fits to tukana root
% dtscp /home/foo/lol.fits tukana:          # abs path to tukana root
% dtscp /home/foo/lol.fits tukana:/mypath/          # abs path to tukana sandbox
% dtscp ../lol.fits tukana:                 # rel path to tukana root
```

2) Copy a local file or directory to a remote queue's delivery directory:

```
% dtscp -q DES foo.fits fred.edu: #copy to DES queue on fred.edu
% dtscp -r -q travis /hello/ tukana: #copy absolute directory hello to tukana
% dtscp -r -q travis ./hello/ tukana: #copy relative directory hello to tukana
% dtscp -p -r -q travis ./hello/ tukana: #copy relative directory hello to tukana using pull
% dtscp -q travis -f myfiles.list tucana:  #copy file list into travis queue on tukana
```

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

//TODO make it work with remote paths with multiple hosts.

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

```
% dtscp tukana:/lol.fits .          # copy to the relative path
% dtscp tukana:/lol.fits /mydata/       # copy to the absolute path
```

4) Retrieve FITS files on machine 'fred.edu' to the current local directory:

```
% dtscp fred.edu:data/*.fits .       # DTS sandbox path
% dtscp fred.edu:/data/*.fits .       # system absolute path
```

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

```
% dtscp -r /data fred.edu:                  # recursive transfer
```

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

```
% dtscp -r fred.edu /data:                    # recursive transfer

// currently won't work with root. but will with path e.g.  fred.edu:/dir2
// 0 byte file send needs to be fixed.
```

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

```
% dtscp -r fred.edu:/data bob.edu: #recursive transfer
```

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

```
% dtscp -q DES fred.edu:  bob.edu:                    # transfers to bob's queue DES
% dtscp -v -q DES fred.edu:  bob.edu:        # verbose
% dtscp -t bob.edu -q DES fred.edu: bob.edu:  # using cmd target host name

% dtscp -H 3001 -L 3000 -P 3005 fred.edu: bob.edu:
        #set theHI port for destination
        #low port for destination
        #RPC command port for source
        #push


% dtscp -p -H 3001 -L 3000 -P 3005 fred.edu: bob.edu:
        #set theHI port for source
        #low port for source
        #RPC command port for source
        #pull
```

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

```
% dtscp -p tukana2:/lol.fits tukana: #using pull
% dtscp tukana2:lol.fits tucana: #using push
% dtscp tukana:/home/foo/tmp/dts.foo/lol.fits tukana2: #full absolute path

# to q DES on bob.edu
% dtscp -q DES fred.edu:/funny.txt fred.edu:/hello.txt bob.edu:

# copy directory home3 into home2 recursively.
% dtscp -r tukana2:/home3/ tukana:/home2
```

10) Multiple host submission to queue or file transfer.

```
# transfer file from tukana and fred.edu to tukana2 queue travis
% dtscp -q travis tukana:/funny.txt fred.edu:/hello.txt tukana2:

# transfer file from tukana and fred.edu to tukana
% dtscp tukana:/funny.txt fred.edu:/hello.txt tukana2:
```

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