DSSM - Dead Simple Stream Multiplexing

Protocol

DSSM Protocol Specification

Basics

DSSM is a protocol for multiplexing 256 bidirectional byte streams over a single one.

DSSM does not provide any error detection, it depends on a reliable, ordered transport.

DSSM does not provide encryption. If this is required, the underlying transport must provide it.

DSSM uses a minimal set of control characters from the C0 ASCII character set for stream multiplexing, keep alive signaling and stream termination indication.

For *DSSM,* streams are stateless. Error signalling is out of scope of *DSSM.* Senders and receivers have to agree on stream and stream state semantics as well as on error signalling.

Signaling

The control characters `ENQ`, `ACK`, `NAK`, `SOH` and `EM` are used for signaling. The `ESC` control character is used as a prefix for transmitting any of these control characters or for transmitting itself inside a stream.

Initial State

Upon connection establishment of the underlying transport any data transmitted is considered to be sent by stream 0.

Stream switching

The character `SOH` signals a switch to another stream. The byte following the `SOH` character is interpreted as the stream number (0..255). All following characters up to the next signal are partof the respective stream.

Stream termination indication

A stream sender can signal the "end" of the current stream by sending the `EM` character followed by a status byte.

To *DSSM* there is no meaning of "end" of a stream or the stream "status". Especially, if a sender "ends" a stream and then continues sending data to the same stream, *DSSM* does not consider this an error.

Keep alive signaling

Whenever the `ENQ` character is received, the receiver should send back an `ACK` or `NAK` character before sending the next byte of the stream.

Whether `ACK`/`NAK` is considered a good/bad status indication, or whether the failure to respond an `ENQ` with `ACK`/`NAK` has any meaning is completely up to the implementation of sender and receiver.

Unsolicited `ACK`/`NAK` signals are not considered an error and must be discarded by the receiver.

Notes

Periodic `ENQ`/`ACK`/`NAK` signals can be used to keep an idle connection alive or to sense operational status of the opposite side.

`ENQ`/`ACK` can be used by a sender to meter round trip delay, transmission queue sizes or to flush remote buffers.

`ENQ`/`NAK` can be used to indicate fatal conditions or request restart of the underlying connection.

The receiver can choose to interpret a sequence of multiple `ENQ` `ACK` or `NAK` characters or unsolicited `ACK`/`NAK` characters as a denial of service attack. The same applies to sequences of stream switch or stream end signalling without interspersed stream data.

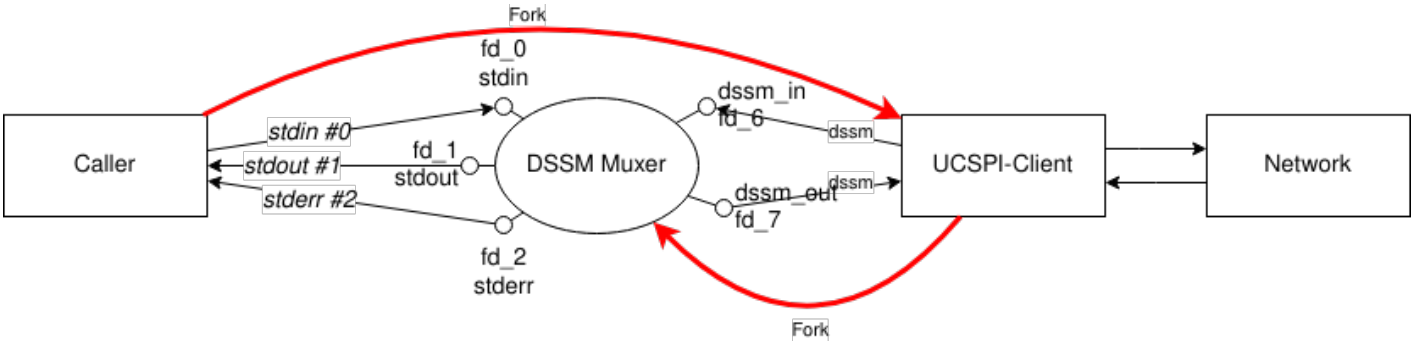
If the underlying transport is packet oriented, senders might decide to split the streams into pieces fitting into one packet.

Implementers must take care not to intermingle `ACK`/`NAK` signals into an `ESC`, `SOH` or `EM` sequence.

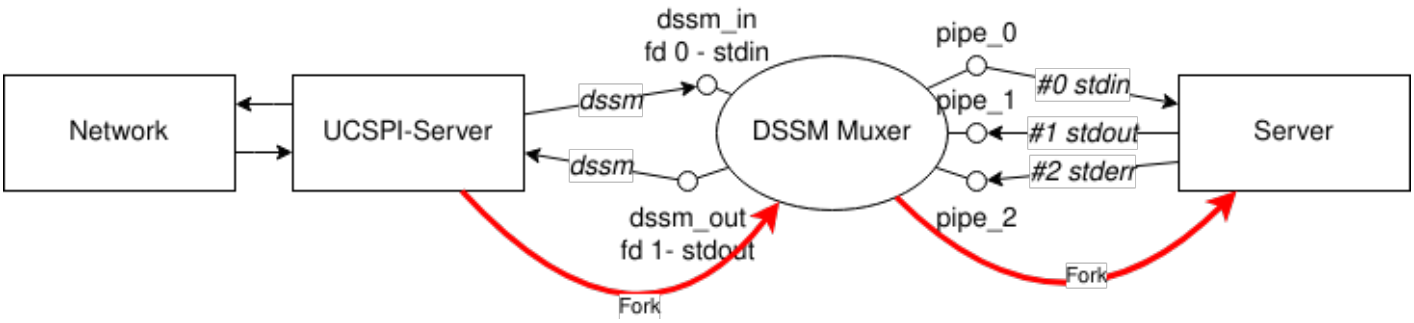
The rationale for defining *DSSM* is to provide the `stderr` backchannel for *rush*, the 'remote unsecure shell', inside a
single TCP connection in a leaner way than done by SSH.

Muxer

Diagram



stream #	in_fd	out_fd
0	stdin	-
1	-	stdout
2	-	stderr



stream #	in_fd	out_fd
0	-	pipe_0_out_wr
1	pipe_1_in_rd	-
2	pipe_2_in_rd	-