**Title:**
**SABUL (Simple Available Bandwidth Utilization Library)**
**UDT (UDP-based Data Transfer Protocol)**

**Contacts (Include institutions)**:
Yunhong Gu [ygu@cs.uic.edu]
Robert Grossman [grossman@uic.edu]

**URLs / RFCs / Papers**
**http://sourceforge.net/projects/dataspace/**

**Principle / Description of Operation**
The primary goal of SABUL/UDT is to utilize the abundant bandwidth over current long haul networks, such as computational grids. Fairness is important as well. Particularly, two of the major fairness objectives are to be independent of RTT and TCP friendly.
SABUL uses UDP to transfer data and TCP to transfer control information. UDT uses UDP only for both data and control information.
Below is a brief description of data transfer in one direction.
The sender sends out a data packet every inter-packet time, which is updated by the rate control. However, it cannot send out the packet if the number of unacknowledged packets exceeds a threshold, updated by the flow control. A retransmission packet has higher priority than first time packet.
The receiver receives and reorders data packet. If it detects packet loss, an NAK packet will be sent back reporting the loss. Selective acknowledgement is used in the protocol, which generates an ACK packet every constant time if there is any packet to acknowledge.
The receiver also measures the packet arrival speed and the link capacity, which will be sent back together with the ACK packet.
The sender sends back an ACK2 packet for each received ACK packet, which is used for the receiver to measure RTT, as well as to decide the next ACK value (i.e., it must be greater than the last received ACK2).
The receiver also checks the RTT variance to check if there is a delay increasing trend. If so, it sends back a delay increasing warning packet to the sender.

**Supported operation mode:**
memory to memory (general transport)

**Authentication**:
None.

**Implementations / API**:
C++ library on Linux/BSD/Solaris.
NS-2 simulation module.

**Congestion Control Algorithms:**

The SABUL/UDT congestion control algorithm combine rate based control and window based control.
The window based flow control limits the number of unacknowledged packets, and the window size is updated to the product of *receiver side packet arrival speed * (RTT + SYN)*, where SYN is the constant increase interval.
The rate control changes the inter-packet sending time according to packet delay and packet loss. It is using AIMD algorithm, where the decrease factor is 1/9, while the increase factor is related to the link capacity, which is probed using data packet pairs. Note that the increase is not based on any interval related to RTT, but is constant and fixed in SABUL/UDT.
Packet delay increasing trend is detected through the variance of RTT.
The slow start begins with 2-packet flow window and 0 inter-packet interval sending rate.

**Fairness**:
For single bottleneck scenario, the congestion control algorithm is basically AIMD, and it guarantees intra-protocol fairness.
In addition, the fairness is approximately independent to network delay, i.e., connections sharing the same bottleneck but with different RTTs can share the bottleneck bandwidth fairly (equally, in this case).

**TCP Friendly**:
Yes. Since SABUL/UDT uses delay as sign of congestion, when coexisting with TCP, it can only occupy the bandwidth that TCP fails to utilize (e.g., because TCP's inefficiency over high BDP network).

**Predictable Performance Model:**
Given the router queue size is large enough (at least link *capacity * (RTT + SYN)*), our protocol can utilize at least 94% of the bandwidth, independent of link capacity and delay. Multiple parallel connections obtain the same throughput as a single connection between the same end hosts.

**Results:**

**Target Usage Scenario:**
SABUL/UDT is general purpose transport protocol.