A Critique of
"SMUX Protocol Specification"

for UCSB CS 290I Winter 1999

Mihai Christodorescu

A working draft of the W3C, the SMUX protocol is designed to allow for the multiplexing of various application–level protocols over one TCP connection. The starting point is the perceived inability of the HTTP protocol to take advantage of the streaming capabilities of the underlying transport protocol TCP. HTTP allows for one request per connection, which demands several TCP connections to be opened, or for several requests per connection, but serialized.

The keyword in this paper is "experimental." In the good tradition of the Internet, the creation of a new protocol is subjected to public discussion. Thus, this revision of the document is highly tentative, and expects feedback, rather than a standard.

First of all, it is not clear of the benefits/advantages of building SMUX on top of TCP, rather than making it a transport protocol of equal weight to TCP. This comes back to the interface provide by SMUX. At a minimum, a TCP–like interface needs to be provided, to allow for insignificant modifications of the application level protocols. Rewriting HTTP is not a solution, since this is exactly what we are avoiding (otherwise, one could design a much better HTTP that does not require multiplexing). Also, functionality to do selective mux–ing would be needed, so certain application level protocols would use a single "mux–ed" stream (standard TCP), while others (such as HTTP) would require multiple streams to be mux–ed.

Error handling is not defined. If a receiver drops a packet, how are ACKs or NACKs piggybacked? What if a certain packet that is

mux–ed in the SMUX packet cannot be added to the application queue (maybe because the application is busy and has a full queue)? In this case, defining a way to drop SMUX packet components, with retransmission afterwards would be the way to go. If there is no way to address component packets from an SMUX packet, then mux–ing several real–time streams cannot be reliably done (due to the different effects a varying jitter can have on application–level protocols).

Another issue not covered is the way multiplexing is done. They say the policy is to be defined by the implementation. This would require a way to decide on both ends about the policy. Negotiation protocols are available, but some standard issues to be negotiated are to be determined. A simple policy might be to give equal size packets to each protocol from the list to be mux–ed together (the democratic way). But this might kill all the efficiency and usability of bursty applications. For example, mixing MPEG stream and HTML streams must be done with more bandwidth allocated to MPEG and less to HTML. This would be a size–based policy.

The paper does not provide any implementation details, or statistics related to the RTT gain/loss in various environments. Overall the paper is very sketchy.