# Pushing the Envelope

How do I tranfer a1 MByte Payload?

# Usually sufficient:

- UINT32 setWorkAreaSize(UINT32 size);
  - 64 Kbyte default
  - Used as an interim repository in double-buffering situations
  - Set **prior** to SystemInit();
  - Release 4.0:
    - checks the largest registered property size and if smaller than current work area does a re-alloc (*except VxWorks*).
    - Adjusts the size of the incoming request for Stock Properties if greater than current work area.

# Transport Issues

- TINE uses UDP by default
  - Limited flow control
    - Set at the server side (all clients are ‘equal’)
    - int **SetBurstLimit**(int npackets); // default = 20
    - Number of packets sent without a ‘pause’
    - If a 10 Mbit net is attached (you’re at home with your DSL link) then a large data transfer will fail unless the burst limit is set to a small number (<10)
    - If you want to multicast large images as fast as you can, set to a large number (1000) and forget about those DSL guys.
- Use the CM_CONNECT flag -> TCP
  - Has flow control (but all timeouts are still respected)
  - You can’t multicast on TCP

# Transport Issues

- Your server won't have trouble sending a large data payload.
- Your client might need some help:
  - SystemAssignBufferSpace(UINT32 rcvBufferSpace,UINT32 sndBufferSpace);
    - Assigns recv and send buffer space for a socket in the network stack.
    - This buffer space acts as a FIFO for a socket
    - Default is 64 Kbytes.
    - This call is not always successful!
      - Windows seems to get it right almost always
      - Linux seems to need some tweaking (e.g.)

        /sbin/sysctl –w sys.net.core.netdev_max_backlog=2000