

## ST-API Proposal

Jim Pinkerton, SGI, June 10, 1998

### APPROACH

- Optimize / specify common case
- Provide direct mapping for only what we do well
  - Layer the rest

### GOALS

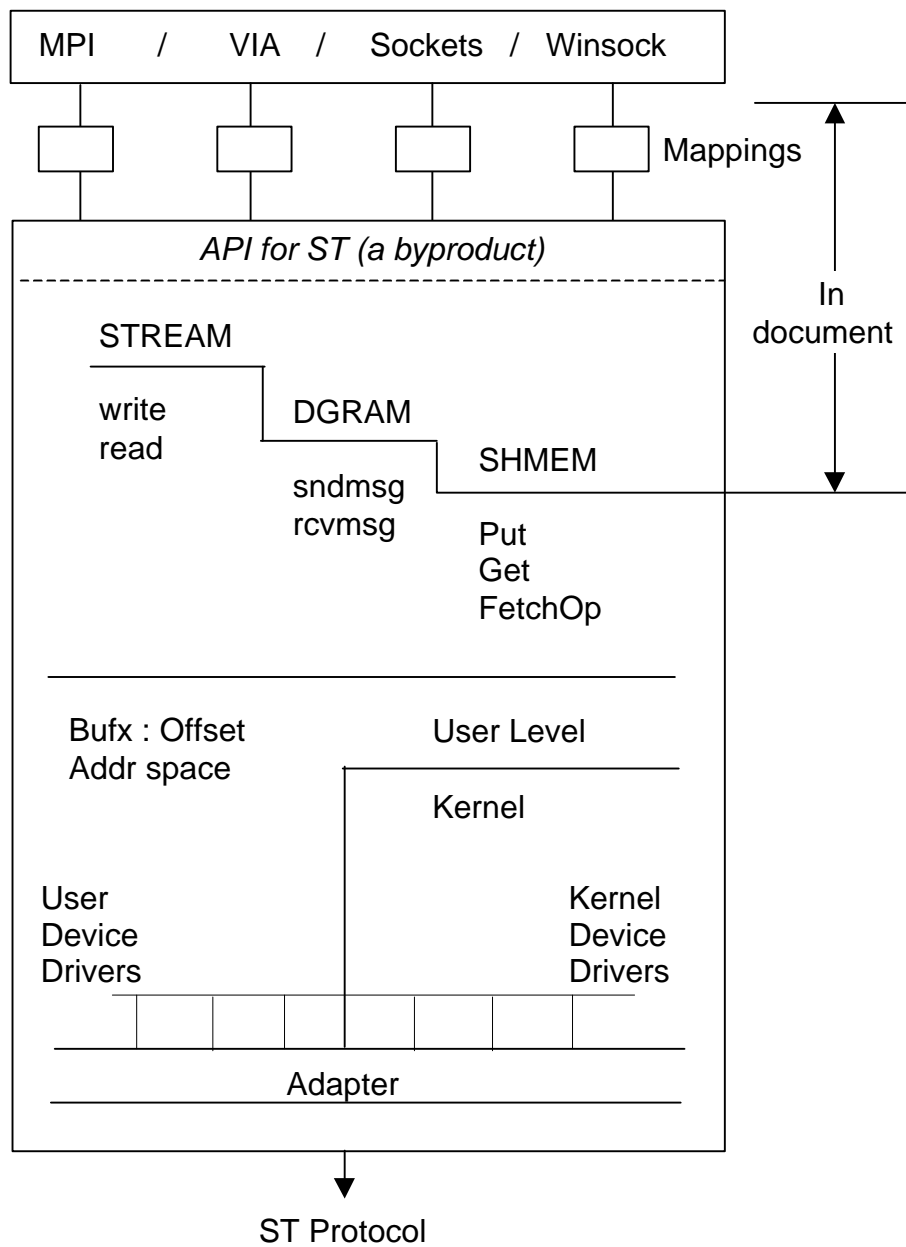
- Support an OS bypass
  - Kernel supports
    - Connection management
    - Buffer management
      - Bufx Range
      - Pinning / Unpinning buffers
      - Mapping buffers to Bufx in adapter
- Socket API with extensions
- Lowest layer supports all ST fuctions
- Highest layer is simplified abstraction
  - One of 4 models
    - Stream
    - Reliable datagram
    - Unreliable datagram
    - Memory
  - Linear address space
    - Do not expose Bufx:Offset
- Device independent
- Addressing of end devices leverages IP

### OTHER GOALS

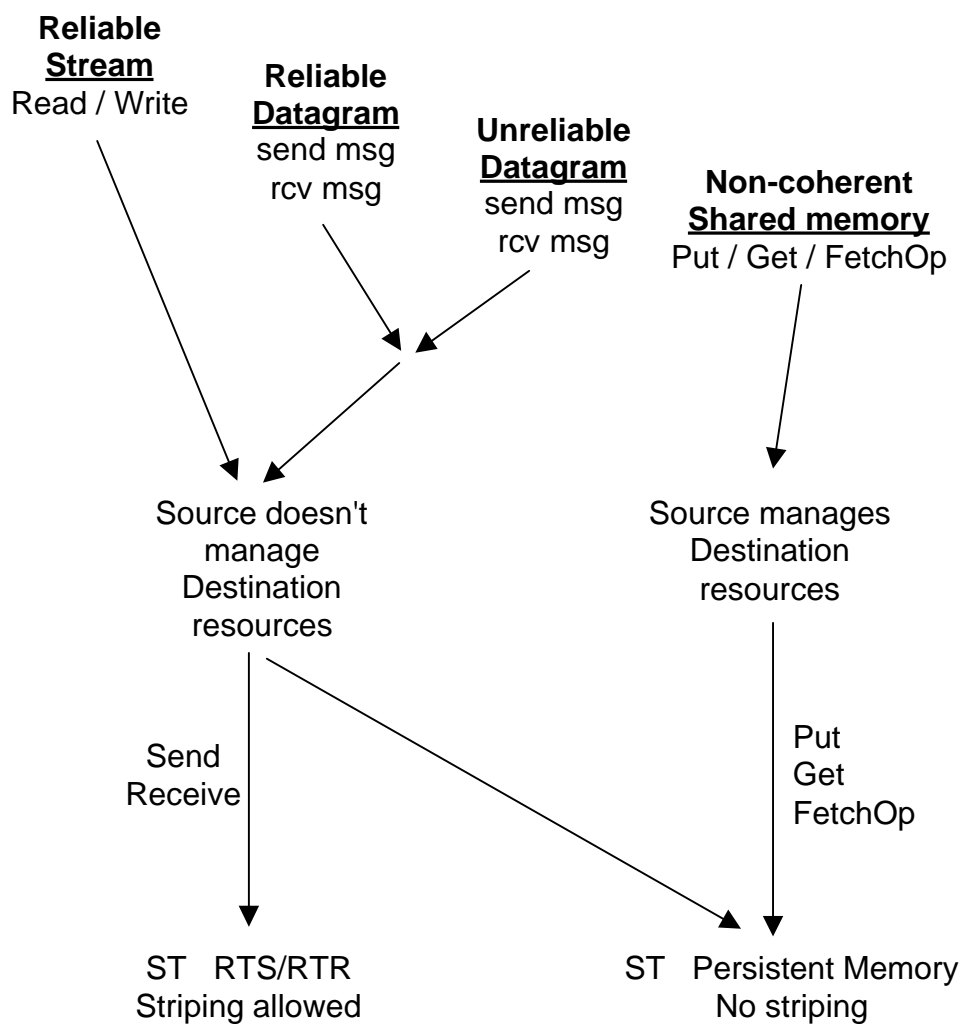
- Expose the destination descriptor to the application
- Destination interrupt
- Number of data channels
- Non-blocking transfers
- Asynchronous transfers (if time)

### EXPLICIT NON-GOALS

- QOS



# API Taxonomy



## Connection setup

```
sd = socket (int domain, int type, int protocol)
    sd = returned socket
    domain = AF_INET
    type = { SOCK_STREAM, SOCK_SEQPACKET, SOCK_DGRAM,
            SOCK_MEM }
    Protocol = IPPROTO_STP
    bind ( )           // for bypass must bind at both Client and Server

set/getsockopt ( )   // initialize various parameters

listen ( ) / accept ( ) / connect ( )

getsockopt (rinfo...) // get information about remote end device

// send data using mode specified in socket call

close (sd)
```

---

Notes for the items in bold type (*red color on a screen*) -

(these items are a "big deal")

**AF\_INET** means we are using the IP address family

**SOCK\_MEM** is new - used for the Put/Get/FetchOp API

**Client** - means the client must call "bind" as well as the server

**IPPROTO\_STP** is new - need to get an official IP protocol number for ST

**set/getsockopt ( )** - extensions to existing socket API to support ST