

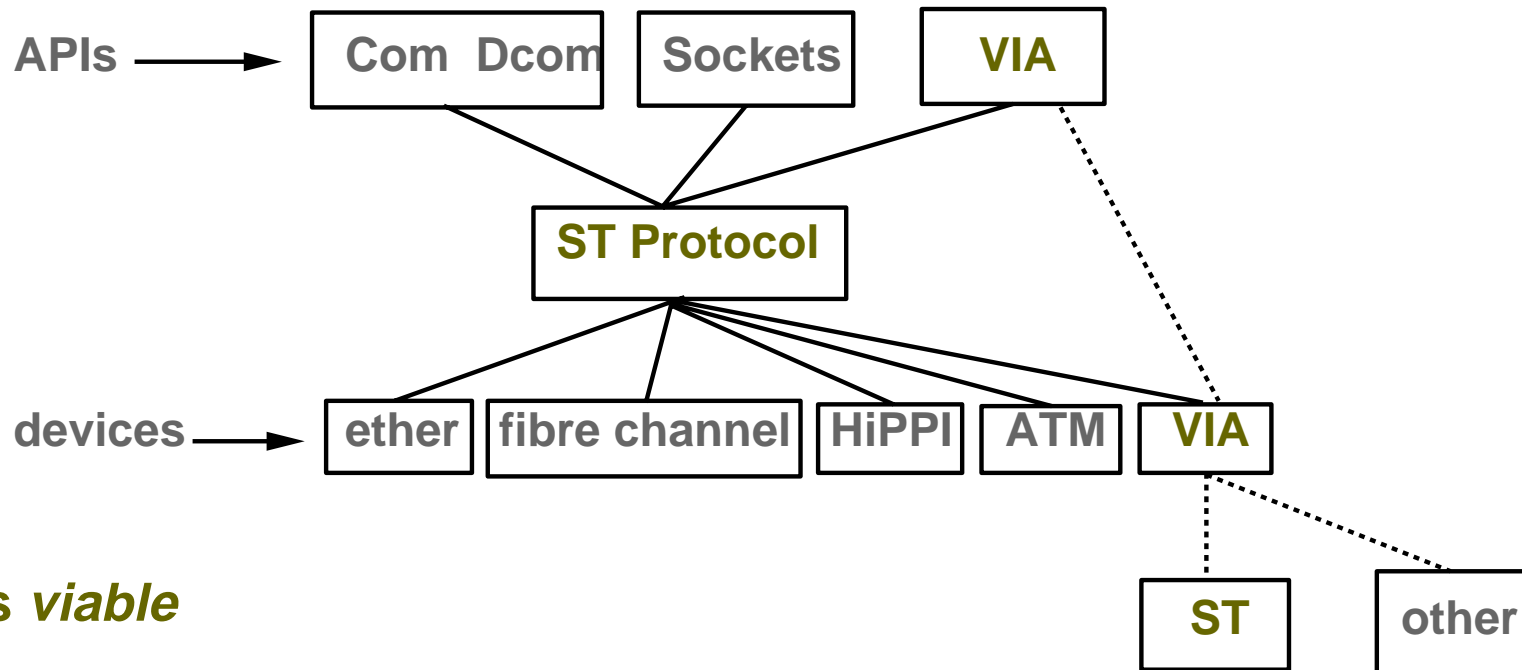
⁵**ST**

(Scheduled Transfer Protocol)

Overview



Regarding VIA



- ❑ **ST is viable**
- ❑ **useful above or below a VIA layer**
- ❑ **enable media bridging**
 - FC-to-ether, or ether-to-HiPPI, etc
 - solve mtu size and flow problems

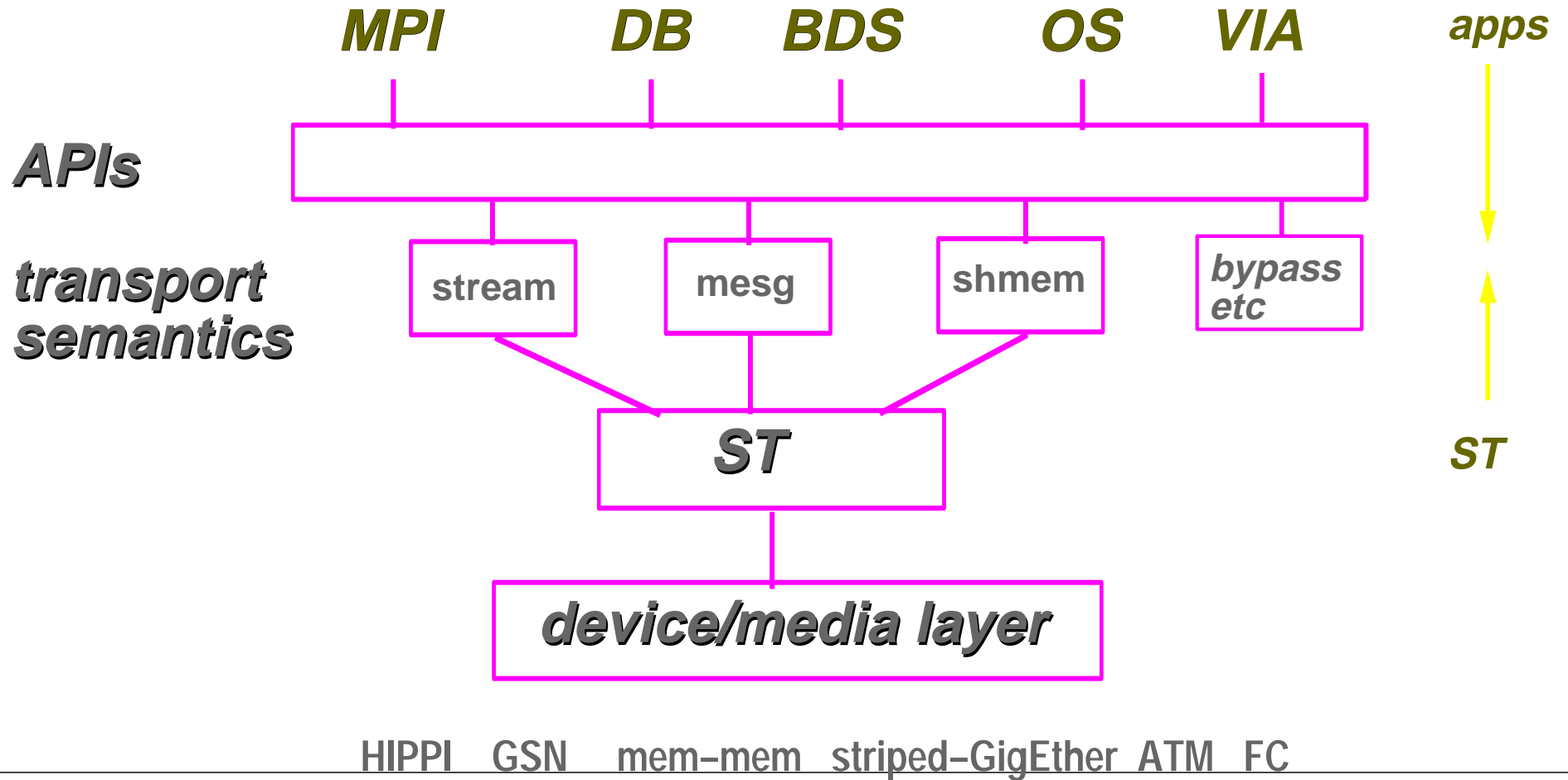


Regarding Applications

- ❑ **MPI**
*uses OS Bypass (ST and VIA equivalent)
FCS in 1996, migrating to ST*
- ❑ **BDS**
*Bulk Data Service
mature product, Rev 2.2 now
93 MB/s over HIPPI-800, 720 MB/s over GSN (HiPPI-6400)*
- ❑ **Database Applications**
Informix (1996), Oracle, others
- ❑ **Cellular Irix**
*fault-tolerant distributed OS
using ST over memory, ST over network media*
- ❑ **HPC Applications**
applications using get/put primitives migrate to ST infrastructure



applications meet services



Network Evolution: bandwidth, latency, technique

- 1988 Protocol Engine Project
transport in vlsi (IP,TCP,XTP) is complex, limited ROI
- FDDI
checksum support, page-flipping, interrupt reduction
- 1993 HIPPI
100 MB/s, unlimited mtu
- 1994 HPC clusters over HIPPI
message latency is paramount.
OS Bypass protocol: ~100 us latency, 90+MB/s channel bw
similar to VaxCluster, Hamlyn, HiPPI-MI, VIA
- 1998 Gigabyte System Network (GSN aka HIPPI-6400 aka OC128)
OS Bypass becomes Scheduled Transfer Protocol
799.92 MByte/s sustained, <1 us latency,
hardware flow and error control, 5-8M IP packets/s
evolve to H25600 (OC512) and H51200 (OC1024)



Have Big Data Pipe, now what?

- ❑ establish interoperability conventions***
- ❑ define addressing, translation, mappings***
- ❑ establish connection protocol***
- ❑ define transfer semantics***
- ❑ establish reliability models***
- ❑ define APIs and other software interfaces***
- ❑ build on existing work where possible***



Most Important

High Bandwidth + Multiple Clients == Instant Congestion
in switches, bridges, routers
in adaptor cards,
in host software and buffers.

example: 10 systems at 100 MB/s saturates a 1 GB/s adaptor.
must turn over 65536 16K buffers/sec.

penalties: descriptor queue and buffer overload, lost data,
aggravated retransmission – "ATM disease".

solution: memory is "cheap" – transform to space domain.

*Transfer only to allocated memory resources.
Provide strict controls.
Many variations are possible.*



Opinions

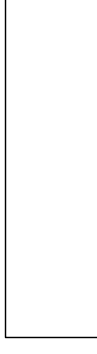
- ❑ users prefer:
 - multi-platform
 - multi-vendor
 - multi-LAN
- ❑ problem space is large
- ❑ VIA and FC do not address all the issues
 - VIA (Rev 1.0) does not define a protocol
- ❑ ST is a viable candidate
- ❑ Hardware ages quickly

Protocols are forever



ST "transactional" design

- based on 2-way, 3-way, 4-way handshakes
- "transactional" protocol
synthesize streams and allocation windows on top of simple transactions
- concurrent transactions for pipelining, striping, error control
- state machines defined for complete protocol
(not just connection management)
see VIA Rev 1.0 (states: Idle, Pending, Connected, Error)
see Annex E of ST for transfer FSM specifications
- hardware implements dma and message validation
software implements protocol state machines



ST Objectives

bulk bandwidth

"long message algorithm", large block support, direct dma

low latency

"short message" algorithm, OS Bypass

multi-peer

multiple clients/server

shared memory

"persistent buffers", get/put primitives

striped transfers

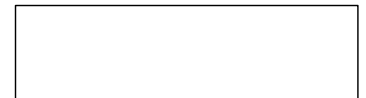
bandwidth aggregation

media independence

MTU/pagesize indifference,
mappings for **ethernet, ATM, Fibre Channel**
provides fix for weenygrams, flow/congestion management

leverage standards where possible

ARP, IANA port administration, IEEE addressing



ST Description

- ❑ ***Define Terms***
- ❑ ***Message Structure***
- ❑ ***Message Processing***
- ❑ ***Protocol Exchanges***
- ❑ ***APIs and Implementations***



Transfer

Transfer (destination buffer, i.e. reserved memory)

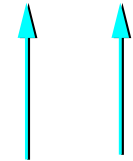



gigabytes



Block (flow/ack/resend/stripe unit)

megabytes



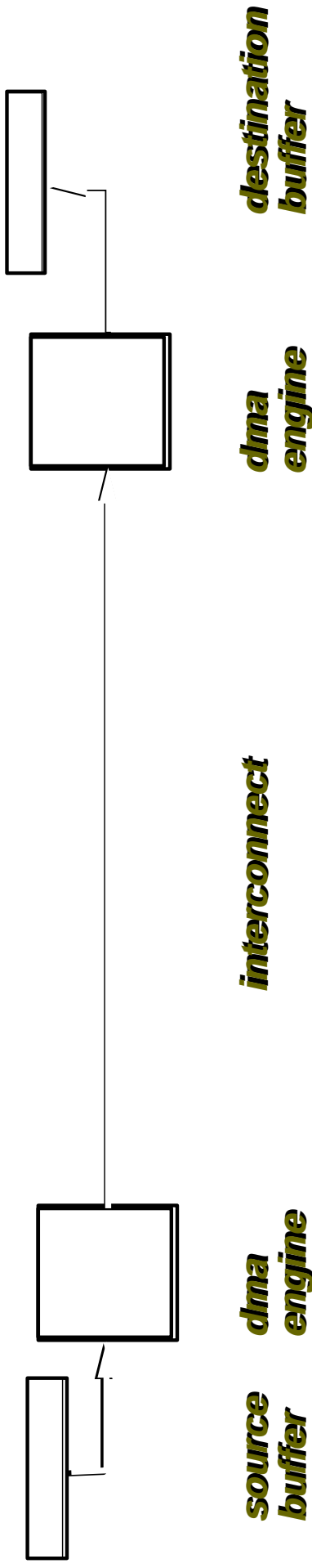
 ***Network Message (transmission unit: STU)***

OS page

 ***μpackets, cells, etc***



Scheduled Transfer



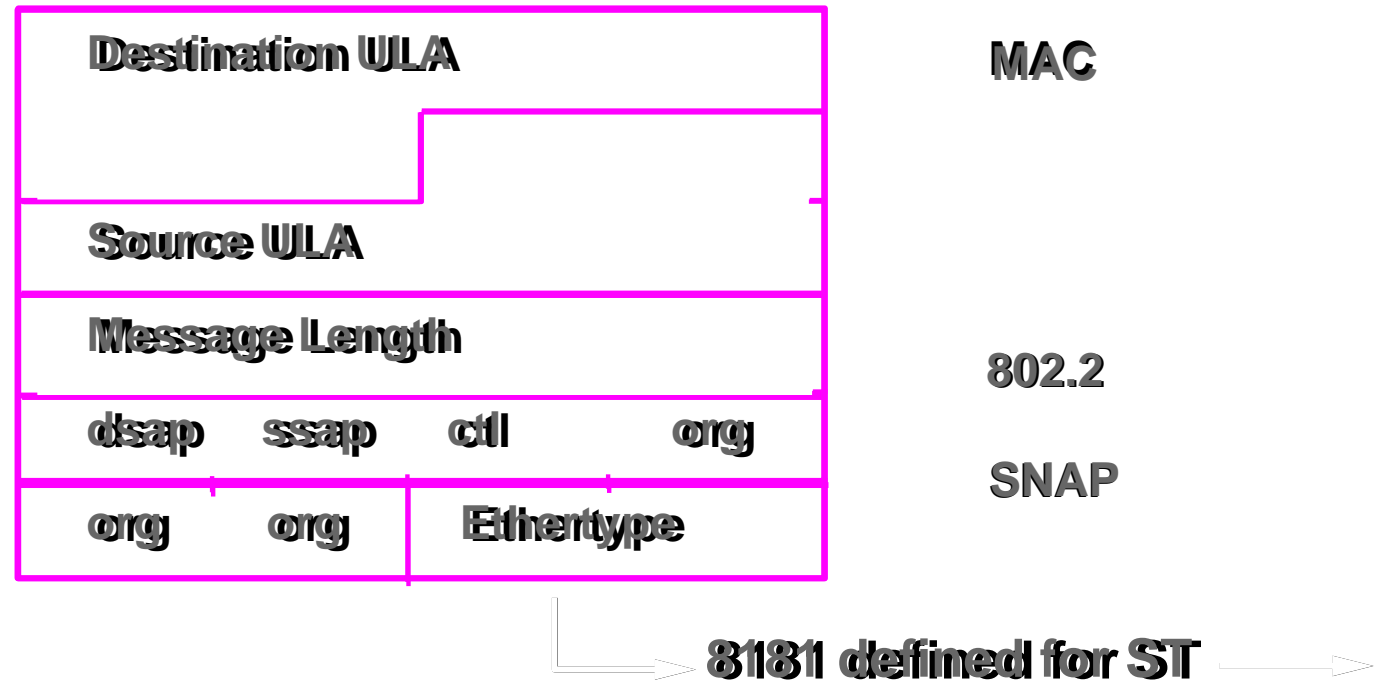
"Schedule" the source and destination buffers and enable dma engines for concurrent execution.

Early examples: Modcomp links, DEC DR11b, DEC CI, VaxCluster also: VIA, HP Hamlyn, SGI OS Bypass, HiPPI MI

Appropriate for both standard networks or specialized media.



MAC Header



Example: GSN header is equivalent to Ethernet+SNAP plus length field.



STU (ST Transmission Unit)



*payload defined by ethertype
during virtual connection setup
e.g. Fibre Channel, IP, MPI, etc*



ST Header Summary

MAC header	
Op flags	Param
D port	S port
KEY	
Cksum	B_id
Bufx	
Offset	
Sync	
B_num	
D_id	
S_id	

OPCODE, SEQUENCE NUMBER

LOOKUP PARAMETERS

Buffer ID

MEMORY LOCATION

table(Bufx) + Offset

SYNCHRONIZATION

BLOCK NUMBER

TRANSACTIONS

ST Header Processing (on NIC)

MAC header	
Op	flags
D port	Param
KEY	S port
Cksum	B_id
Bufx	
Offset	
Sync	
B_num	
D_id	
S_id	

Minimal Hardware

validate Dport and Key

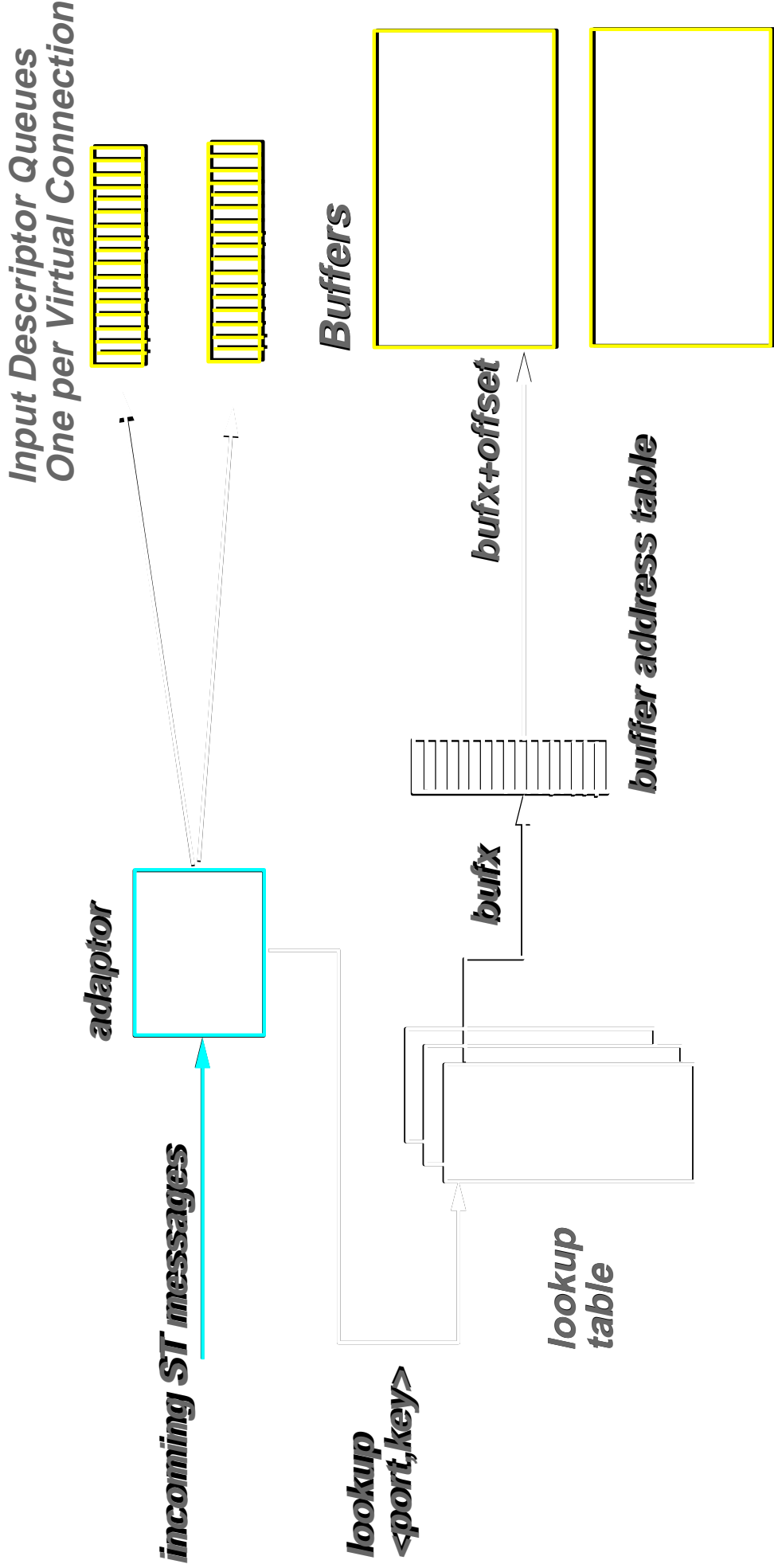
if (Op == DATA)
deliver payload to
(bufx + offset)

check flags for Interrupt

optional: cksum, D_id, B_id

Header delivered to software
for other processing

DMA Address (*bufx+offset*) Mapping



**Incoming *bufx+offset* references host memory.
Operation completion writes descriptor queue.**

ST Destination Processing

Control Path

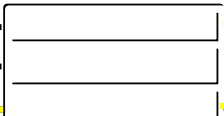
Data Path

software

ST Engine and API Interfaces

input queue

DATA
RTS
DATA

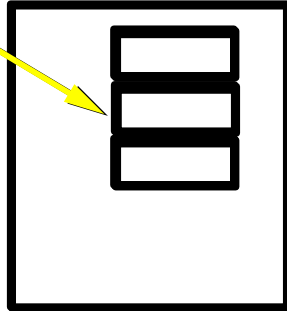


adaptor processing

```
if (ethertype==ST) {  
    decode + validate  
    dma to bufx+offset  
    push descriptor to queue  
}
```



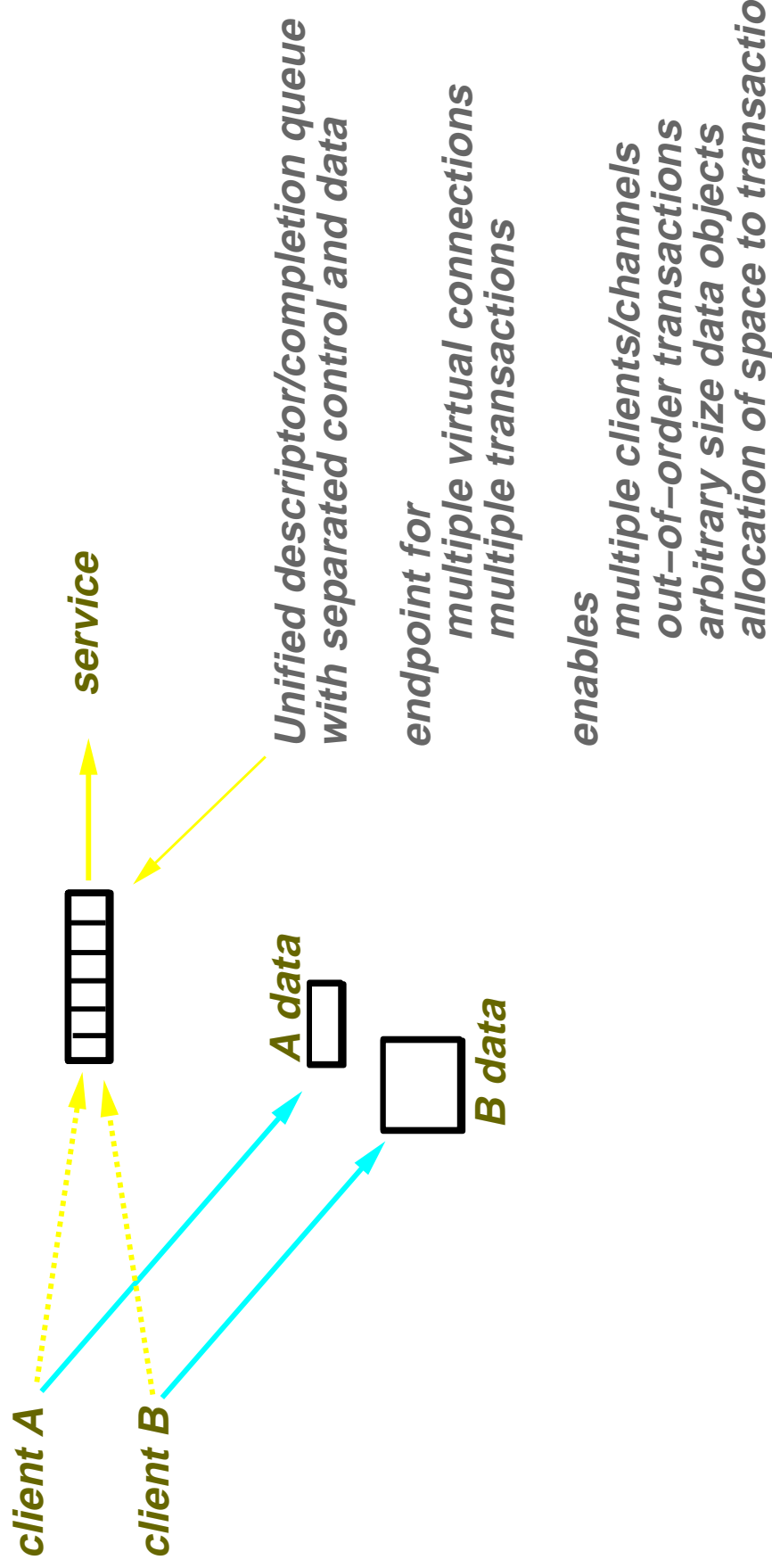
**"long message"
RTS/CTS/DATA**



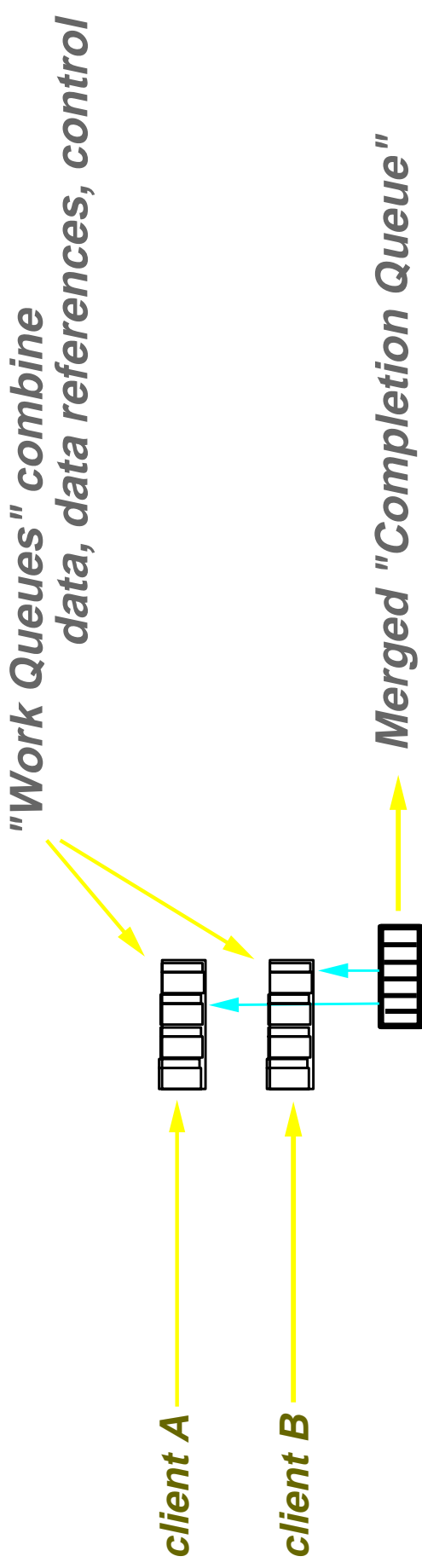
**"short messages"
sent to a page with
multiple writes**



Importance of DST Processing



VIA Destination Processing



*Multiple Queue Model
doubles descriptor-processing cache miss rate
limits scalability
RDMA protocol not defined*



ST Exchanges

- ❑ ***Virtual Connection Management***
- ❑ ***Data Movement***
- ❑ ***Striping and Retransmission***
- ❑ ***Memory Operations***

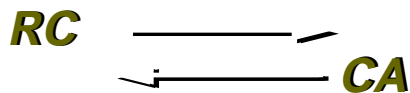


Connection Management

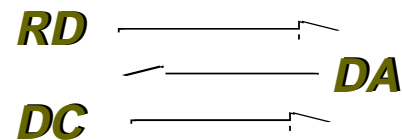
Setup 2-way handshake;
connect to IANA-defined *port* at MAC address
resolved by ARP; exchange *keys* and rendezvous ports,
capabilities, and local buffer preferences.

Teardown 3-way handshake;
needed to avoid aliasing.

Multi-peer multiple clients connect to same rendezvous port
at "server"; share resources, flow control.



Request Connection
Connection Answer



Request Disconnect
Disconnect Answer
Disconnect Complete



Transfer Setup (push)

RTS (Request to Send) <length>



.....
RA (Request Answer) <blocksize>

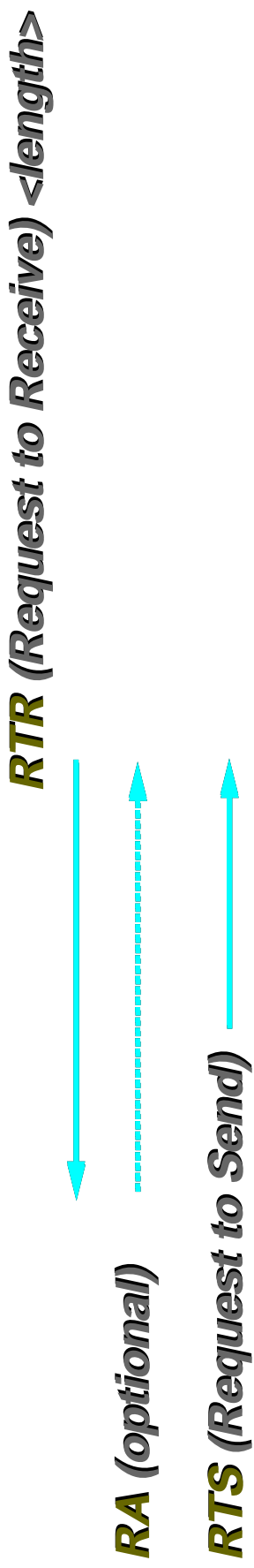


CTS (Clear to Send)
<blocksize, Block #, bufx, offset, Block ID

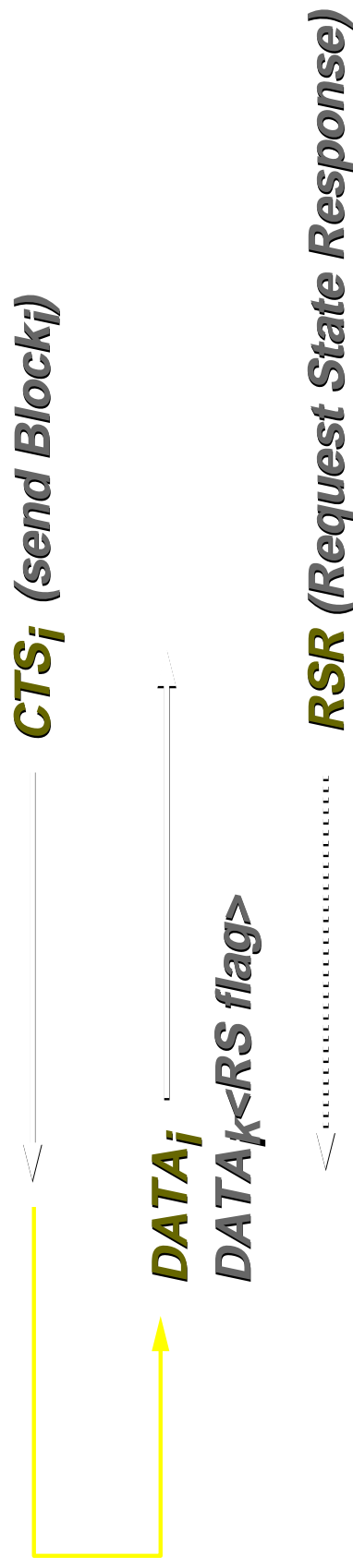


RA is optional
CTS "exposes" memory (bufx range) at destination.

Transfer Setup (pull)

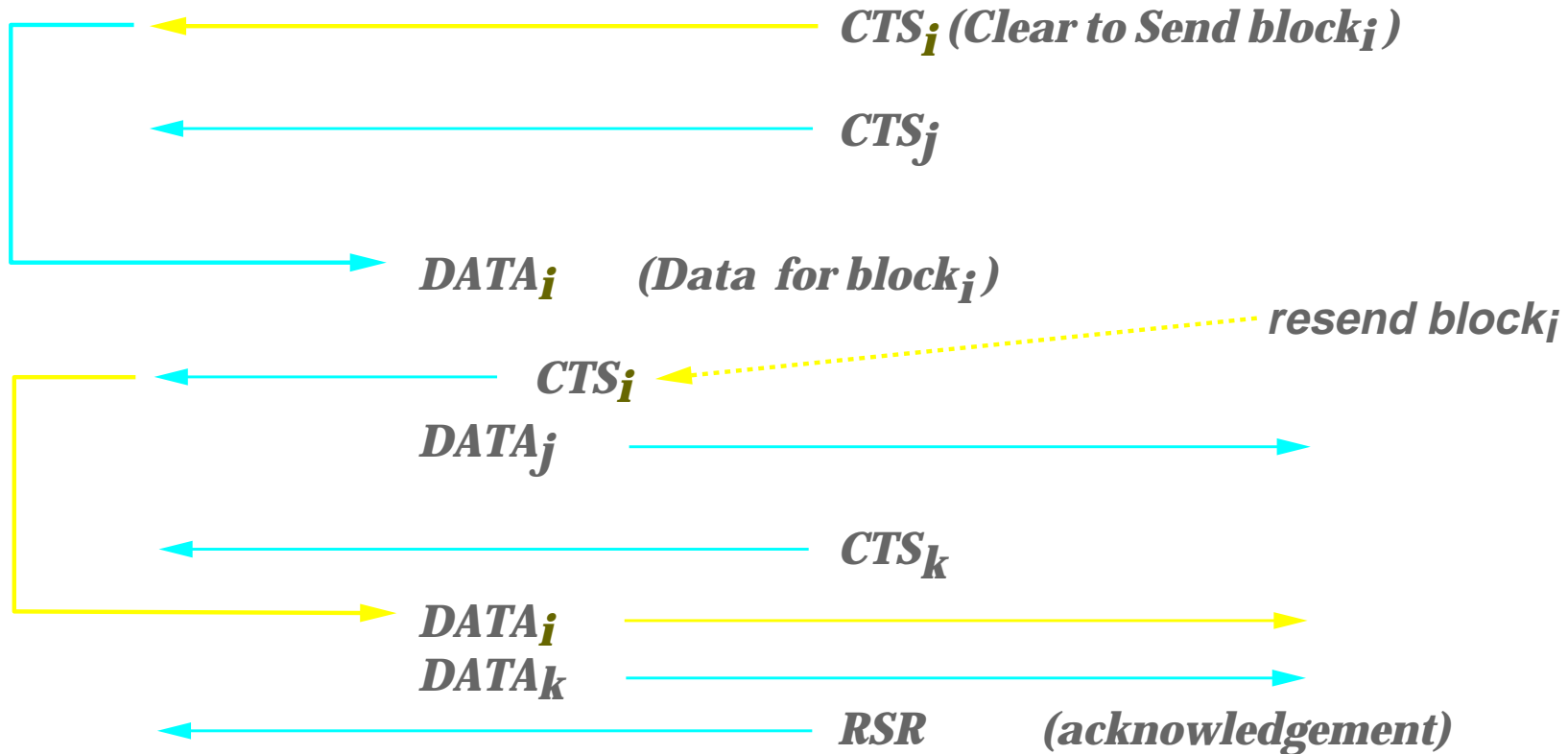


Data Movement (push or pull)



**RSR returns: high-water contiguous received block #,
status of a requested block #,
echo of transmitted SYNC field.**

Striping and Retransmission



Concurrent DATA blocks (CTS) enabling
out-of-order, retransmission, striping



Error Control



Handshakes and sequences covered by timeout.
Essential for operation over IP, unreliable media, media bridges.
Not needed over reliable media, e.g. HiPPI-6400, some FC classes.



Get/Put Operations

- ❑ *Request Memory Block (RMA) requests remote memory. Memory Block Available (MBA) returns a bufx range.*
- ❑ *same mechanism as RTS/CTS.*
- ❑ *the DATA message implements a memory Put operation.*
- ❑ *GetData implements a remote Get.*

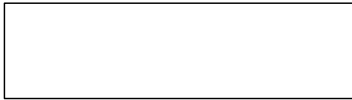
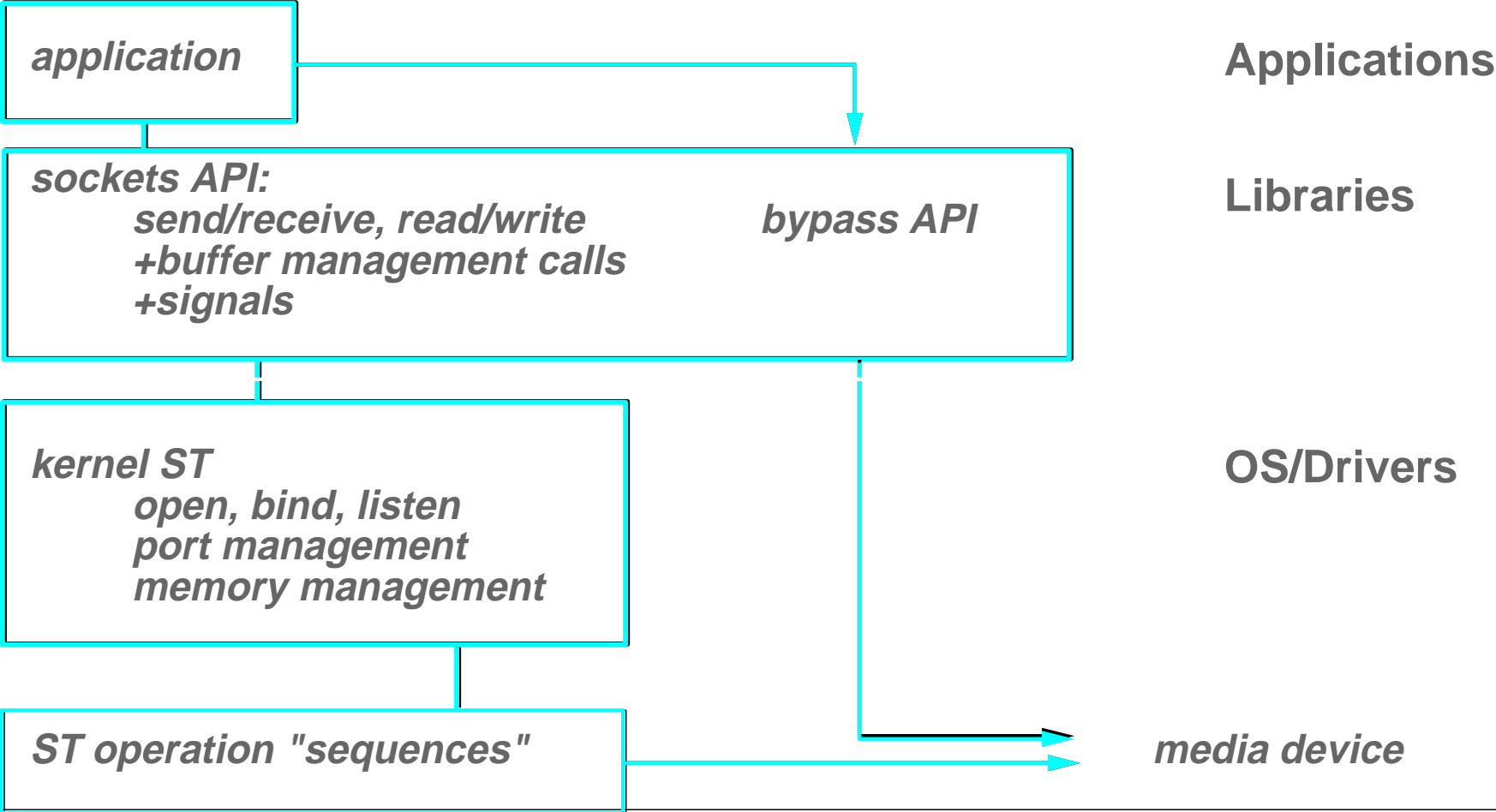


Other Operations

- ❑ ***Fetch & Op***
defined as opcode to GetData
provides atomic inc/dec for distributed barriers, locks
- ❑ ***third-party transfers***
possible by manipulation of CTS fields
similar to IPI-3 scheme
enabler for network-attached storage
- ❑ ***(optional) credit-based flow control defined***
for receive descriptor queues: simple, effective,
essential for high-load congestion control



ST – API Structure



Connection API

socket(domain, type, protocol);

INET	SEQPACKET	STP
	SOCKSTREAM	
	DGRAM	
	RDGRAM	

connect

listen

accept

bind

**plus, additions to namespace for i/f discovery
and fault-recovery model (reconnect to alternate)**

***Implementation: all primitives are in OS kernel even when
OS bypass is employed for data movement.***



Synchronous Data Transfer ***(blocking)***

read/write(fd, buf, cc);

sockread/sockwrite(fd, buf, cc);

transparent to existing applications.
may do RTS/CTS per i/o unless streaming is enabled.



Asynchronous Data Transfers

enabled by: ***setsockopt(fd, SET_NTRANS, &max_number);***

sets max number asynh requests,
user must declare a vector of transaction structures.

engaged by: ***index = st_send/st_rcv(fd, buf, cc, flags);***

where *index* is the *vector[index]* transaction struct
assigned by ST, and
type selects {DATA,RDATA,CNTL,MSG}

completion: ***st_done(fd, &list, flags);***

where *list* is an array of ints.
list[0] contains the number of list entries returned.
each *list* entry contains the *index* of a completed i/o.
ST transactions are cleared by *st_done()*.



Summary

- ❑ *a transactional protocol*

allows explicit buffer control

enables zero-copy OS Bypass

allows both streaming and record semantics

provides handle for asynchronous interfaces

- ❑ *"VIAble" protocol – either above or below*

- ❑ *www.noc.lanl.gov/~det/*

