

# New Transaction Type For 1394 Bridge Control

Dick Scheel  
Sony Electronics  
Richard.Scheel@am.sony.com  
+1-408-982-5834

# Proposal:

- Add a new transaction type for bridge control
  - Async
  - Addressed to a node on a remote bus
  - Not actually delivered to the addressed node
  - Bridge in async routing path to the addressed node intercepts & processes the packet

# Advantages

- Transaction initiator does not need to know which bridge on bus is in the route to the addressed node
- Portal uses existing address decode logic to accept the packet
  - Minor new logic to decode the new type, and put into portal's node receive queue rather than routing queue to other portal

# Uses:

- Isochronous path setup/teardown
  - Contents would be bridge commands
    - Example: Isochronous path setup commands
- Other bridge control uses?

# Choices to be made

- How many new types
- What transactions
- Encoding

# How many new types

- At least one:
  - Picked up & processed by bridge portal that would normally forward an async packet to the addressed destination node
- Possibly a second:
  - Picked up & processed by bridge portal that would normally put the packet onto the addressed destination bus (just forwarded normally by bridges earlier in the path)

# What transactions

- Block write only?
  - \*\* Our preference
- All block transactions (read, write, lock)?

# Encoding choices

- New tcode
- New extended\_tcode with existing tcode
  - \*\* Our preference
- Offset in private address space, with existing tcodes

# New tcode

- Pros:
  - Easy to specify
  - Leaves plenty of header space for parameters
  - Should not have problems with existing silicon
- Cons:
  - Takes one of the few remaining unused tcodes
    - Hard to convince the rest of the 1394 community
  - Another tcode for portals to decode (easy)

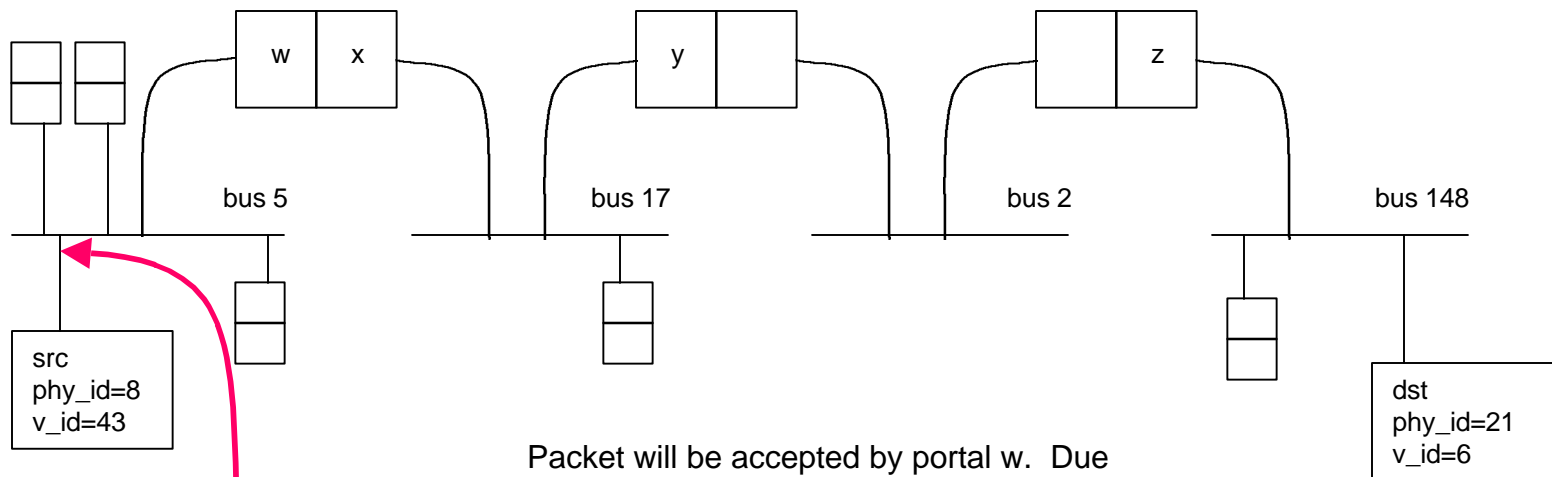
## New extended\_tcode with existing tcode

- Pros:
  - Uses header fields that are either:
    - Currently unused (read & write)
    - Expandable (lock only uses a few bits of extended\_tcode field)
- Cons:
  - Cannot be used with quadlet read/write transactions
    - Our proposed use would probably be all block transactions
  - Any chance that any existing silicon might have a problem?
    - Shouldn't, because these packets are intercepted by bridges & never delivered to the addressed destination node
    - Beware of errant device that puts packet on destination bus

# Offset in private address space, with existing tcodes

- Pros:
  - Shouldn't cause problems with existing silicon
- Cons:
  - Harder for portals to decode
    - No other portal functions require looking at the address offset in routed packets

# Basic tool operation



Packet: src=5/8, dst=148/6  
tcode=block\_write  
extended\_tcode = <new>

Packet will be accepted by portal w. Due to extended\_tcode, transaction will be processed in portal w's node rather than forwarded out portal x. The transaction processing may result in bridge x-w generating a new packet onto bus 17 with content very similar to the original packet.

# Example of use - isoch setup

- Controller sends command to listener
  - Uses the new transaction
  - Packet addressed to listener
  - Bridge in async route from controller to listener accepts & processes the packet
    - If sees that it isn't final bridge, then just puts it on the next bus
    - If sees that it is final, then (approximately)
      - Reserves isoch resources on listener bus
      - Puts "in stream" version of command on listener bus
        - » Addressed to talker
        - » Portal must eavesdrop, as usual
  - (2nd type of transaction could go directly to final bridge)

# Example - continued

- “In stream” command follows async route from listener to talker
  - At each bus, portal in route accepts & processes the packet
    - If sees that it isn’t final bridge, then (approximately)
      - Reserves isoch resources on bus
      - Sets up stream through bridge
      - Puts command on next bus
    - If sees that it is final, then (approximately)
      - Reserves isoch resources on bus
      - Sets up stream through bridge
      - Sends status to controller (normal async write)



# Isochronous setup example

## Step 2

