Timing System Evolution - Progress Towards Synchronous Data Distribution

Jukka Pietarinen

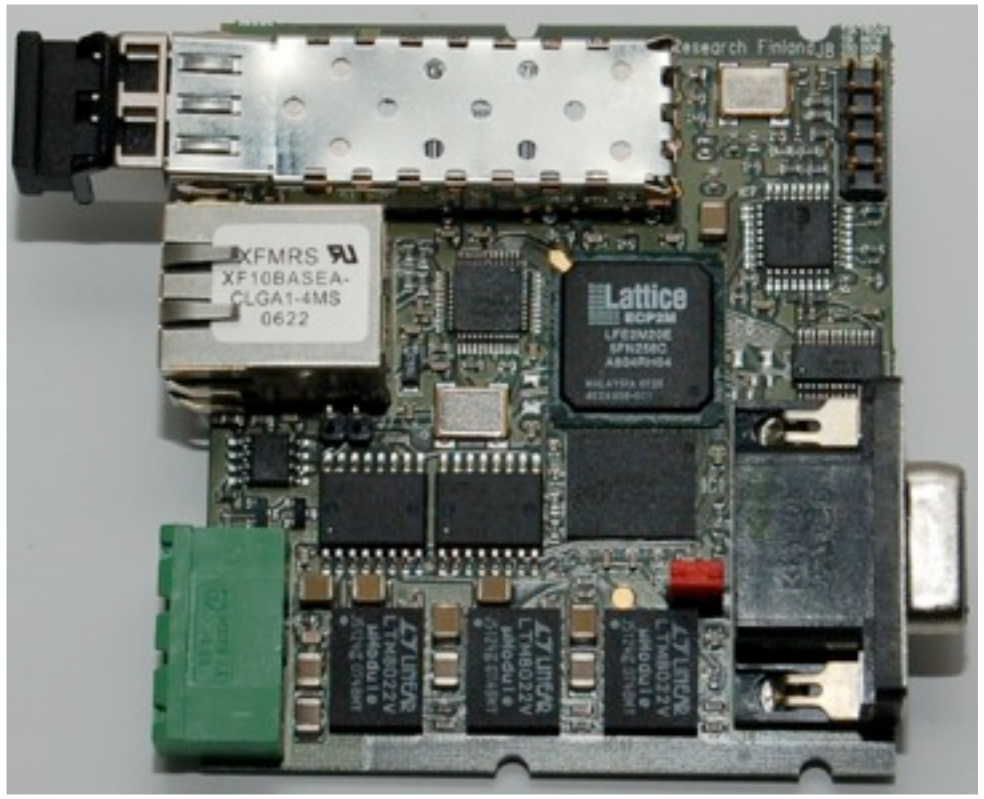
EPICS Meeting Vancouver
May 2009

New cPCI-EVRTG-300 under development

- 6U form factor CompactPCI
- two optical SFP outputs to drive GUN-TX
- two CML outputs
- two Universal I/O slot for four outputs of choice
- All outputs will provide
  - low jitter
  - fine tuning down to ~10 ps steps
- Will replace:
  - EVR-RF + GUN-TX Electron Gun Transmitter
  - EVR-RF + 4CHTIM Four Channel Timer

CompactRIO EVR prototype

- SFP transceiver for event link
- Lattice ECP2M FPGA
- 10/100 ethernet
- 64 Mbytes DDR2 memory
- 2 x 16 Mbits serial flash
- EEPROM
- 9 to 35 VDC power supply input
- DSUB15 with max. 11 I/O pins



Lattice Mico32

- Open source 32-bit soft core primarily targeted for Lattice FPGAs
- Lattice is the first FPGA vendor to submit its GNU derived tool sources to the OSF
- There is a RTEMS port to the Lattice Mico32 (lm32) target in current CVS
- Precompiled lm32 tools available for lm32 target including simulator in gdb (installation with apt/yum)
- Issues remaining:
  - poor JTAG debugger performance
  - boot loader

Timing System v. Synchronous Data Transfers

Timing System

- Low jitter
- High resolution
- Low latency

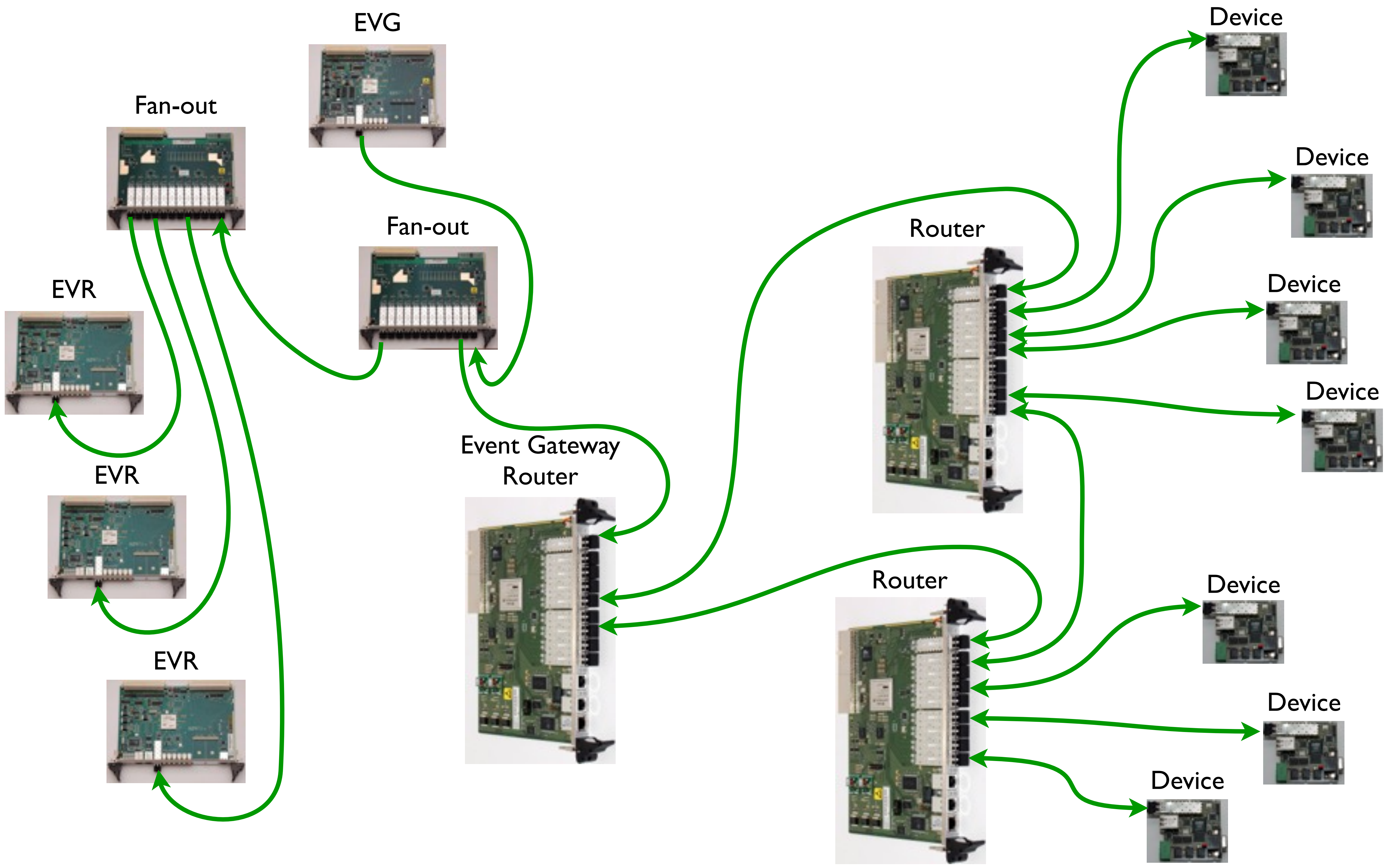
- Priority of Functions is:
  1. Events
  2. Pulses/Triggers
  3. Data
- Topology
  - Star with one master

Synchronous Data Transfers

- High throughput
- Low latency
- Redundancy
- Low jitter?

- Priority of Functions is:
  1. Data Transfers
  2. Events
- Topology
  - Star / Ring

Timing/Data Network Topology



8B10B Encoding

- Common line code used e.g. in Gigabit Ethernet, PCI Express etc.
- Encode 8-bit data byte and a number of K-characters into 10 bit word
  - 256 data codes D00.0 to D31.7
  - twelve K-characters K28.0 to K28.7 and K23.7, K27.7, K29.7 and K30.7
- 16B20B encoding maps two bytes to a 20 bit word
- Balanced line code suitable for optical transmission
- Clock embedded in data
- There is a guaranteed transition every fourth bit except for the Comma character K28.5
- K28.5 is used for alignment

Timing System Protocol

- Uses 8B10B encoding
- Two byte "frame", 20 line bits/event clock cycle
  - every other byte is reserved for an event code
  - the other byte carries distributed bus data or can be multiplexed to carry also data buffers
- Data transfers are inefficient:
  - payload ¼ of full line bandwidth
- Increasing the amount of data transfered adds "noise" which reduces jitter performance

Synchronous Data Protocol

- We turn the scenario upside down:
  - We want efficient data transfers
    - We reserve **all** bandwidth for data
  - But we want to have deterministic, low latency events, too!
    - Allow "drop-in" events

Synchronous Data Line Code

Frame word		Function
D00.0	D00.0	Idle
D00.0	D00.0	Idle
K28.5	D00.0	Sync
D00.0	D00.0	Idle
K28.0	header_0	Start of data packet
header_1	header_2	
K28.5	D16.0	Event code 0x10
header_3	data_0	
...	...	End of data packet
D00.0	D00.0	Idle
D00.0	D00.0	Idle

K28.5 is needed every now and then for synchronization

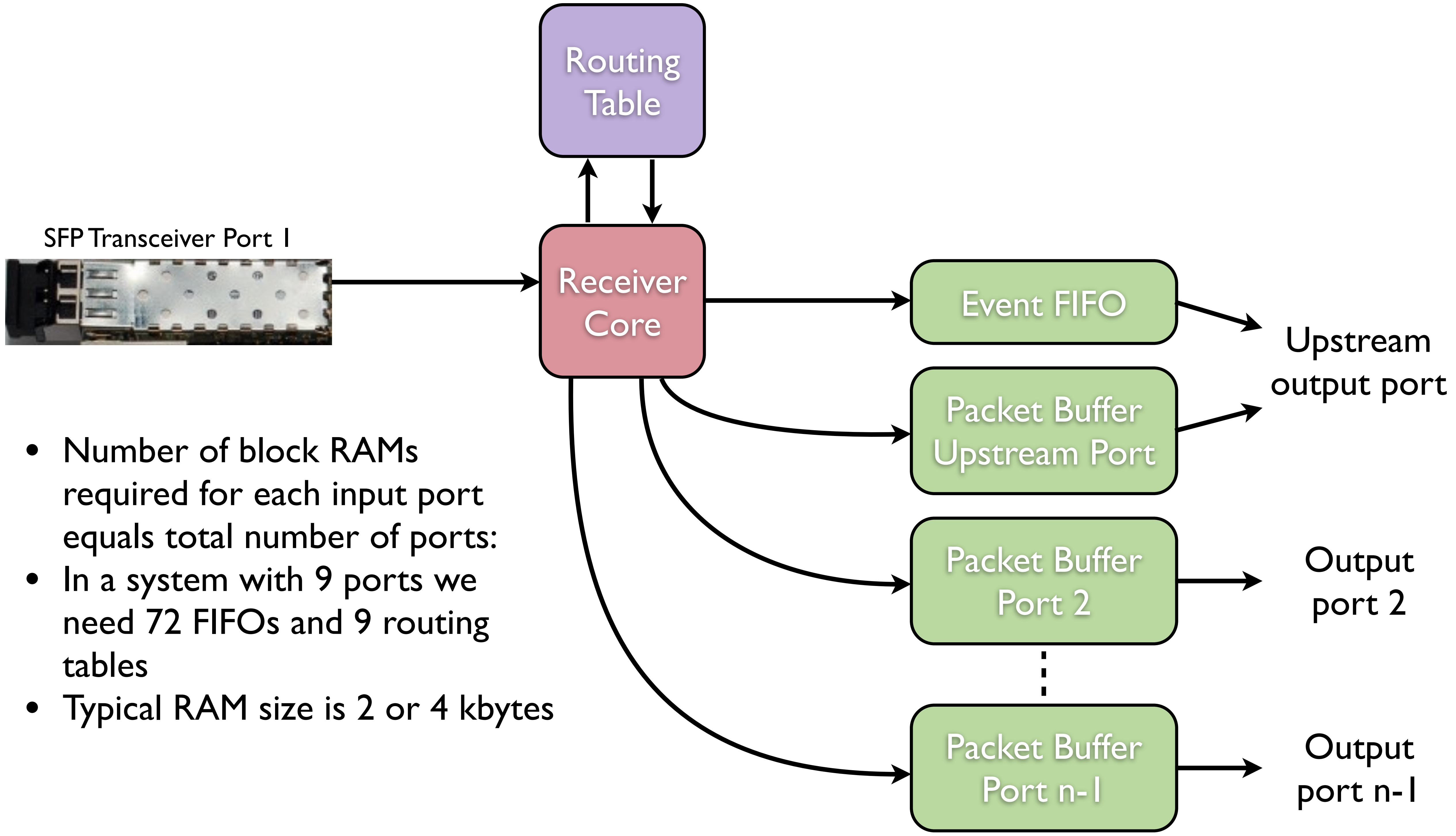
K28.0 starts a data transfer with header, payload and checksums

An event code can be slipped between the data transfer

Router - Data Packet Routing

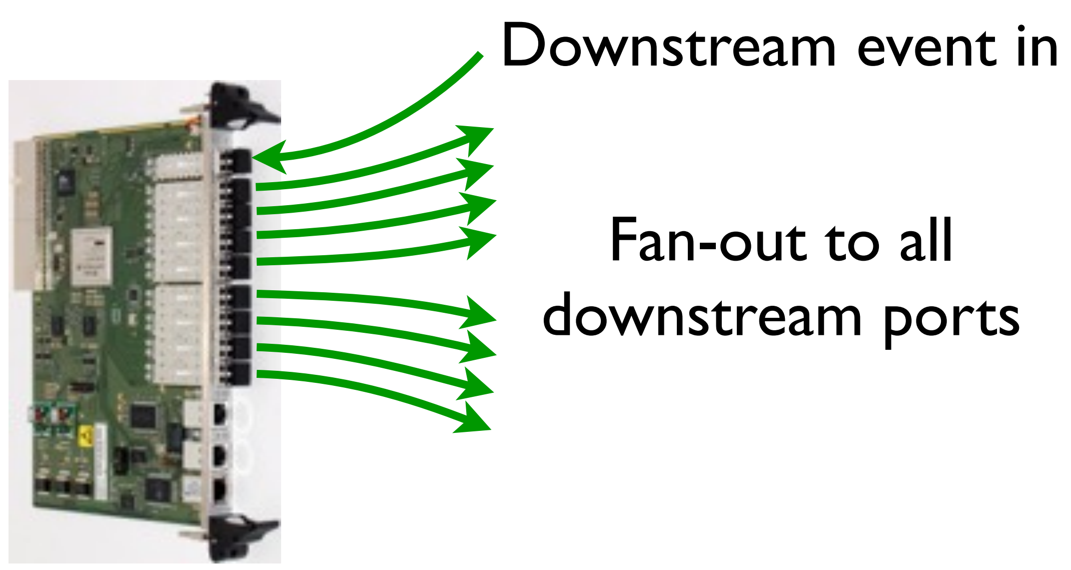
- Routers pass data packets between nodes
- The data packet header has an ID field
- Each router input has a routing table that maps each ID code to a bit vector representing each output port
- we can define on a port-by-port and ID basis where to forward the data packets received on a port
- This allows for different configurations:
  - High-speed point-to-point connections
  - Point-to-multipoint transmission
  - Broadcasts
  - Redundant rings

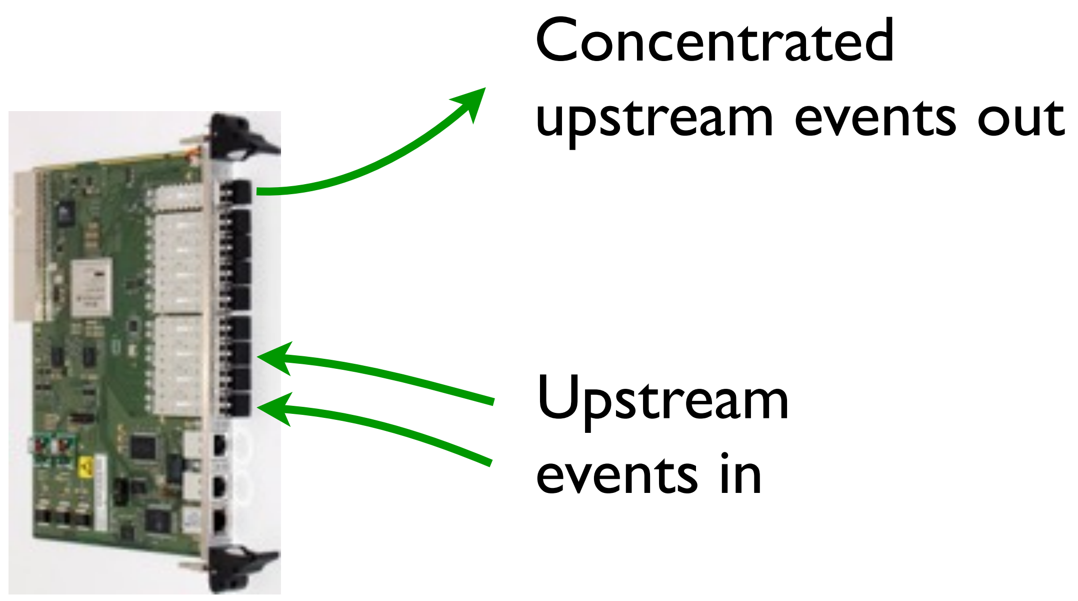
Router Upstream Input Port Logic Blocks



- Number of block RAMs required for each input port equals total number of ports:
- In a system with 9 ports we need 72 FIFOs and 9 routing tables
- Typical RAM size is 2 or 4 kbytes

Router Event Handling Principles





- Events travel only in star topology
- In a two-way system each node could send an event to the EVG
- The fan-out concentrator handles events in a similar way

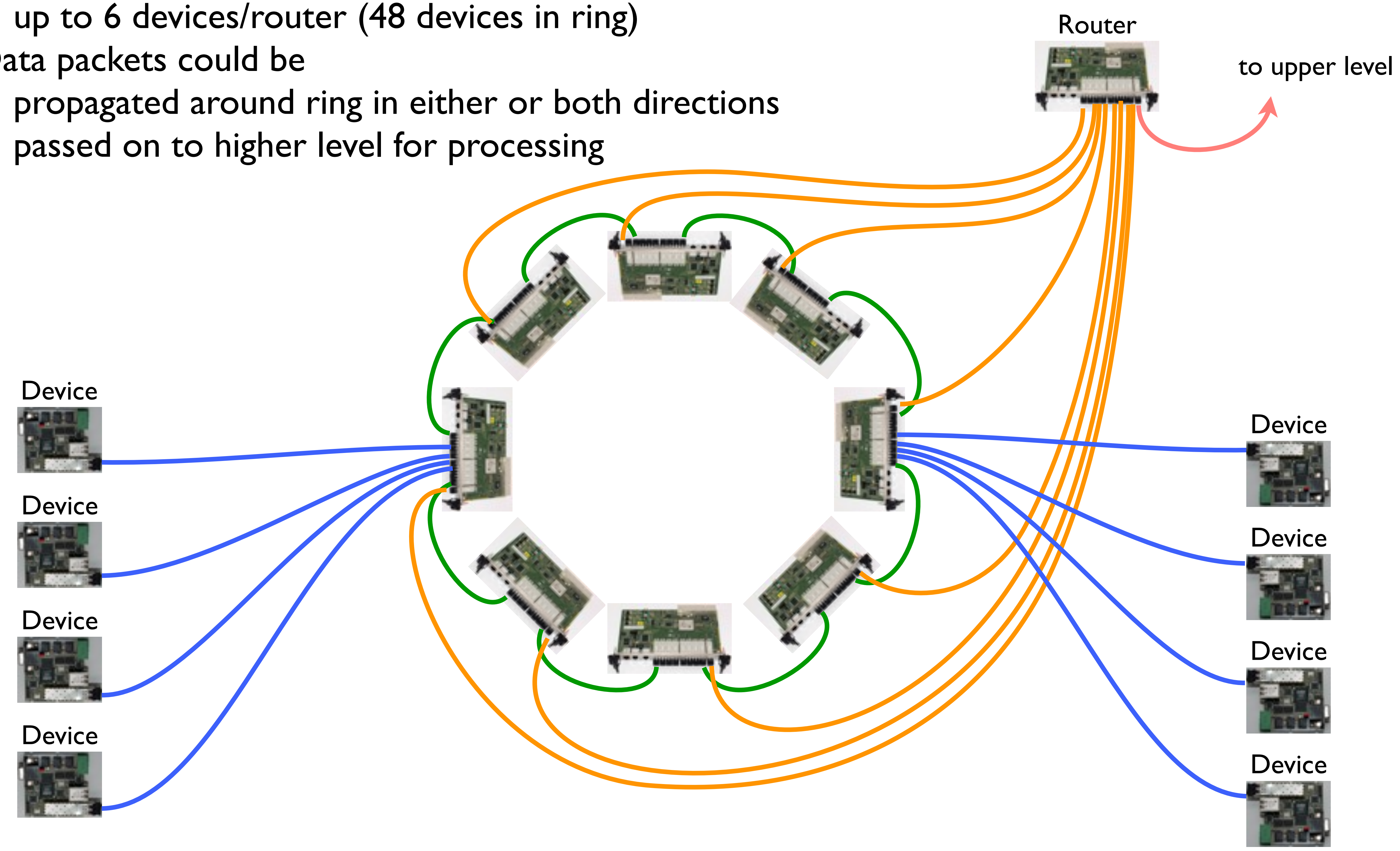
Routing Examples

Ring of eight routers

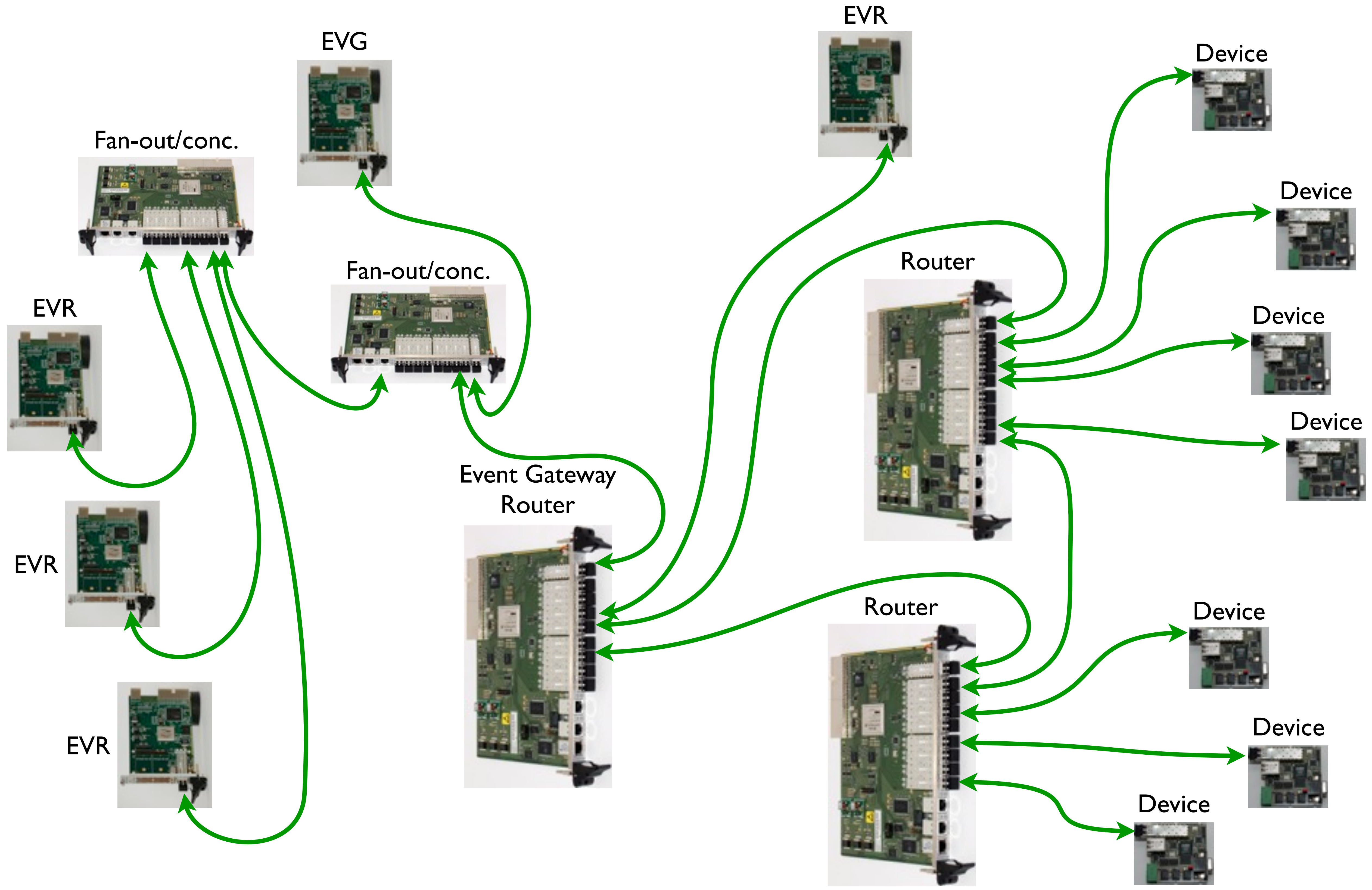
- up to 6 devices/router (48 devices in ring)

Data packets could be

- propagated around ring in either or both directions
- passed on to higher level for processing



Timing/Data Network Topology (hybrid)



Some thoughts...

- Routing has to be designed carefully to avoid overflowing buffers
- This is low level: no flow control, no guarantee that packets reach destination
- Events could be used to synchronize transfers
- “Local” events could be generated to synchronize packet transfers
- Redundancy (two-way ring transfers) could be implemented
- Overall latency consists of fiber delay and ~100 to 200 ns link receive/transmit latency

Open Issues

- How much jitter is acceptable in the data network?
  - Is 8 ns added jitter per Router level acceptable or out of the question?
  - To reduce jitter a recovered clock could be utilized up to some level - no practical experience yet
- How many nodes/IDs are enough?
  - 4096 different IDs would consume one block RAM (32 kbit).
- The distributed bus is not supported. Is this acceptable?

Where to go from here

- Existing hardware “re-use”
  - cPCI-FCT-8 Fan-Out Concentrator HW could be used as an Event Gateway or Router with new firmware
  - All EVR HW (VME, cPCI, PMC, cRIO) could be used as end-points in the new network
- New hardware:
  - redesign of Diamond fast orbit feedback PMC module with four SFP transceivers (mini router)