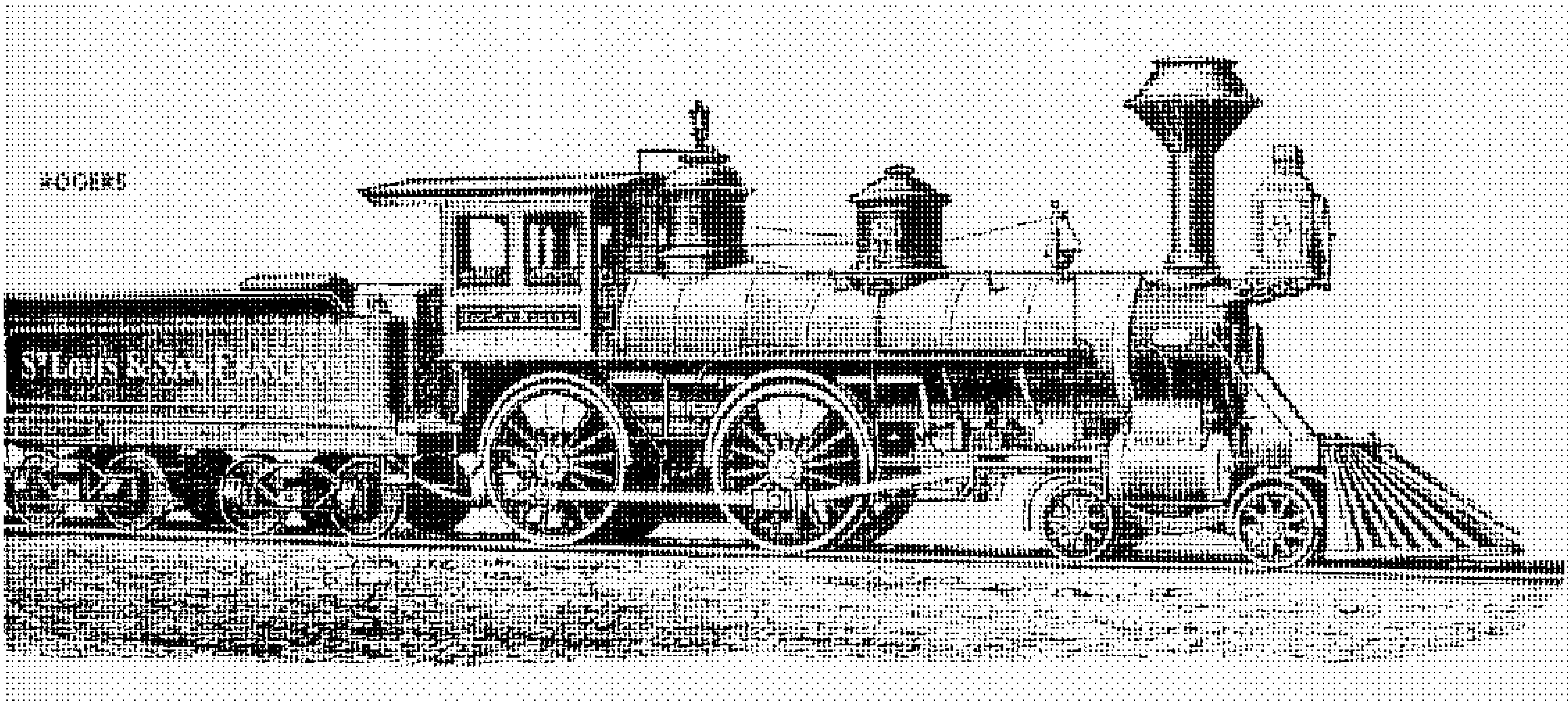


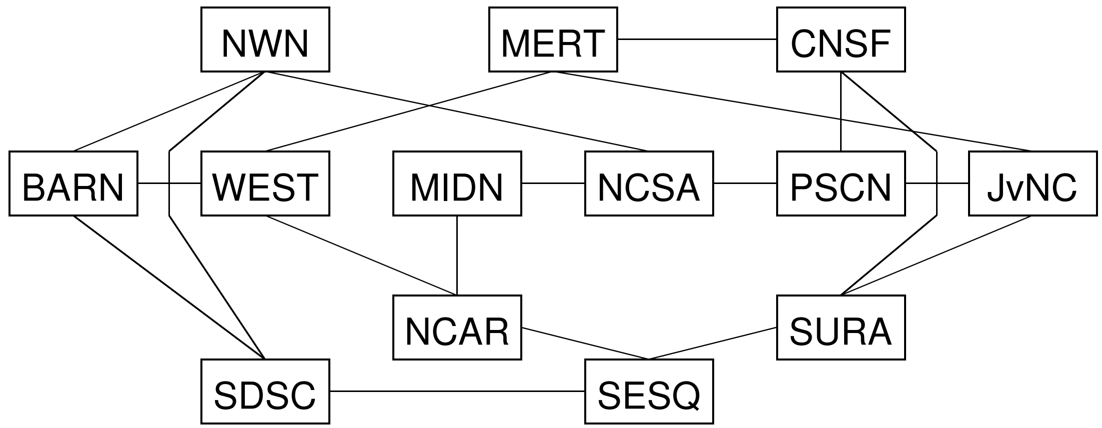
The Highball Project

Profs. David L. Mills, Charles G. Boncelet, and John G. Elias
Electrical Engineering Department
University of Delaware
http://www.eecis.udel.edu/~mills   mills@udel.edu

The Highball Project

- The players, past and present
  - Protocol design and synchronization algorithms: David Mills
  - Scheduling algorithms: Charles Boncelet, Ajit Thyagarajan
  - Hardware design and fabrication, John Elias
  - Grad students: Paul Schragger (scheduling algorithms), Alden Jackson (protocol simulation), Timothy Hall (control program and hardware), timekeeping analysis (Kenneth Monington)
  - The renewable Undergraduate Army also serves
- The resources
  - DARPA/CSTO $849K over 4 years
  - NSF/DNCRI $100K over 3 years, renewed $100K over 3 years
  - U.S. Navy and U.S. Coast Guard $loose change
  - DARTnet nationwide testbed for synchronization experiments
  - Dedicated Ethernet/FDDI LANs: 18 workstations, 5 routers, 2 special servers with total of 500MB RAM, 10GB disk and 3xT1 to WANs

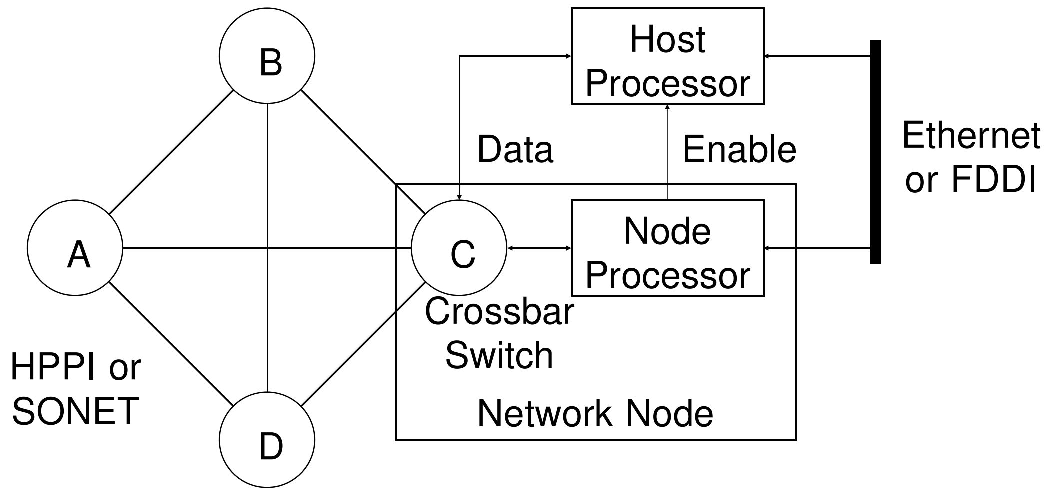
Target Applications



NSFNET
Backbone

- WAN interconnect of high-speed LAN clusters (NREN overlay?)
- Additional targets include space tracking, telemetry, control and remote sensing for NASA Space Station and similar applications
- Technology applicable to DARPA Multiple Satellite System at 10 Mbps, NASA Advanced Communications Technology Satellite at 1 Gbps
- Possible alternative for city stoplights (scheduling algorithm)
- Provide alternative to ATM for high speed bursty users

Highball Architecture



- Data are buffered only in the hosts or on network links, not in the nodes
- Hosts send reservation requests to local node, which encapsulates them in periodic control bursts sent via multicast to all other nodes
- Nodes run a distributed scheduling algorithm which constructs synchronous switch schedules for each node
- Switch schedule reconfigures crossbar just-in-time as bursts fly by
- Hosts gate data burst to local crossbar when enabled by local node

Assumptions

- Very low latency between host and local node
- Very large delay-bandwidth product: 30 ms and 1 GHz typical
- Data bursts are very large and occur relatively infrequently at each host
- Network is transparent to data rate, transmission format and modulation (with analog data interfaces and crossbar switches)
- Transmission can be unicast or multicast and on-demand or scheduled
- Service is intended for:
  - Transparent data rates and encoding formats; e.g., MSK codecs
  - Point-to-point and multicast isochronous applications; e.g., interactive visualization and motion video
  - Burst file transfers for national data forests and supercomputers
  - Real-time remote-sensing and data acquisition; e.g., ocean radar, digital map retrieval

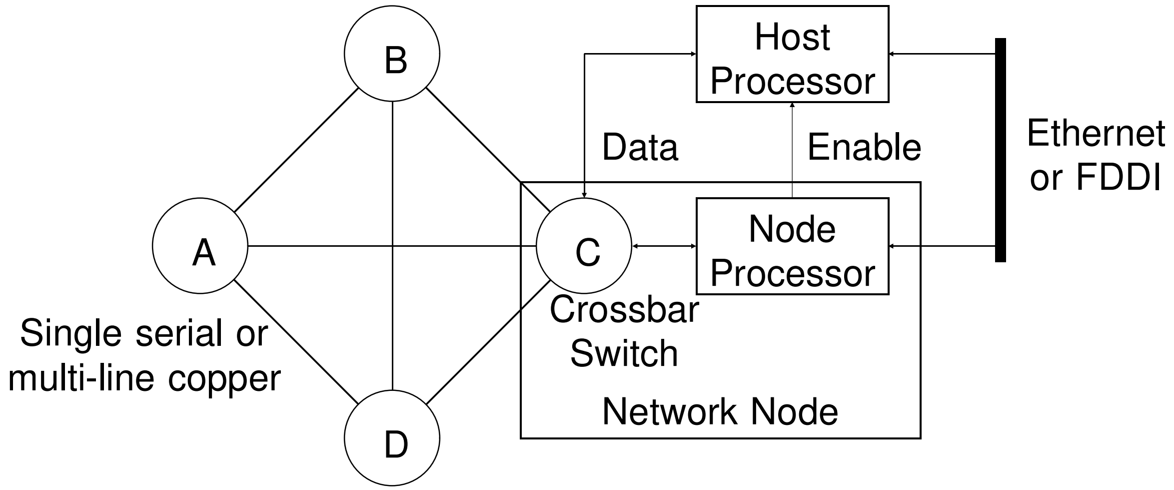
Scheduling Paradigms and Simulations

- Exhaustive enumeration of all paths (Trailblazer)
  - Efficient probabilistic enumeration; baseline for complexity evaluation
- Selective augmentation of existing paths (Pathfinder)
  - Computational efficient for most schedules; exceptions expected to be rare
- Breadth-first search (labeling algorithm)
  - Adaptive techniques designed to reduce the impact of scaling
- Schedule simulator implemented and tested
  - Explore heuristics techniques
  - Assist evaluation and comparison
- Protocol simulator implemented and tested
  - Explore initial-synchronization issues
  - Determine impact of errors

Network Synchronization

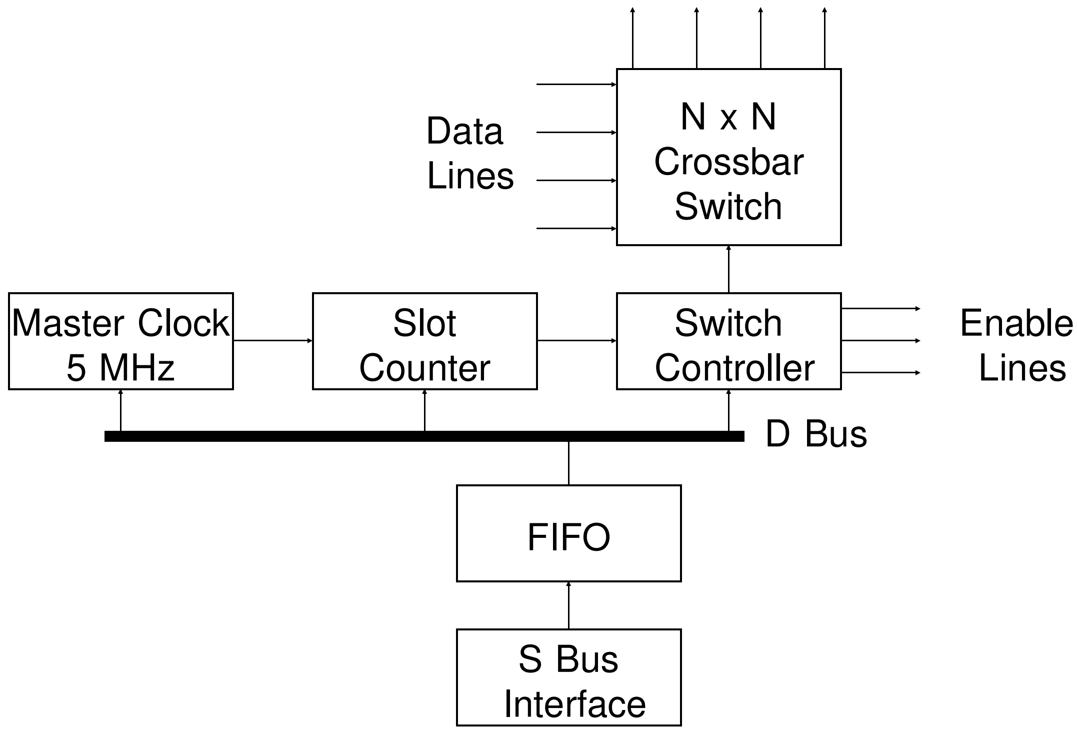
- New Network Time Protocol (NTP) Version 3
  - Assured error bounds provided to users in new interfaces
  - Computations refined for accuracies to microsecond regime
  - Kernel modifications for precision timekeeping in Sun and DEC
  - NTP specified in Estelle (Darren New)
  - NTP security analysis (Matt Bishop)
  - NTP Version 3 specification RFC-1305 now draft standard
- DARTnet experiments
  - Ten sites with SPARCstation-1 routers and modified Sun kernels
  - Experiments showed many deficencies in kernel and daemon timekeeping software, later corrected
- Timekeeping enhancements
  - Cesium clock and LORAN-C receiver for precision timekeeping
  - GPS receivers with accuracy 100 ns relative to UTC
  - Inexpensive LORAN-C receiver for possible replication

Strawman Architecture Overview



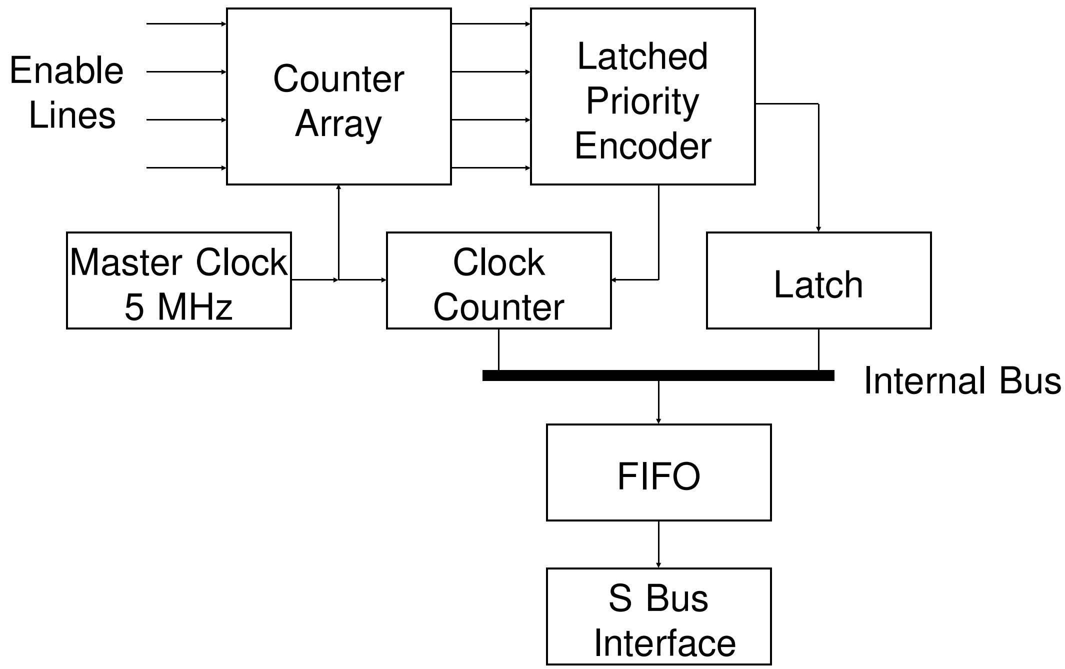
- 3/4-node demonstration network
  - Intended primarily as testbed for synchronization, reservation and scheduling algorithms; links won't carry traffic until later
  - Data transmission may be serial using commercial crossbar switches or parallel using fabricated ones at 600Mbps to 2Gbps
  - FDDI for reservation and backup data
  - Node processors use SPARCstation IPC
  - Host processors use Sun SPARCstation and DEC Alpha

Crossbar Switch and Controller



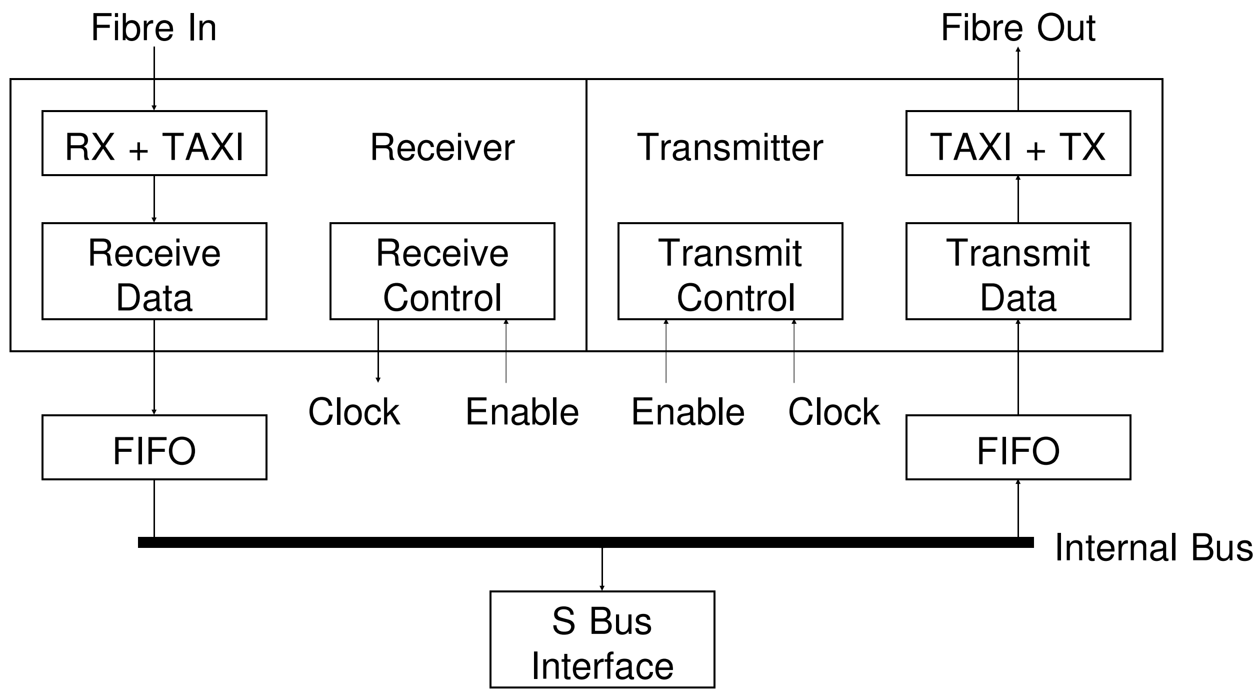
- Master clock VCXO controls all node functions; synchronized with NTP
- Slot counter establishes duration of configuration
- Latched switch controller establishes configuration and enable lines

Timestamp Capture



- Used to capture timestamps upon arrival and departure of hello bursts
- On counter overflow, enable ID and clock counter latched in FIFO
- Counters provide programmable delay for simulation and/or delay compensation

Data Transceiver and Interface



- Transceiver uses TAXI chipset and optical interfaces at 200 Mbps
- Can handle full-duplex transfers at DMA speeds

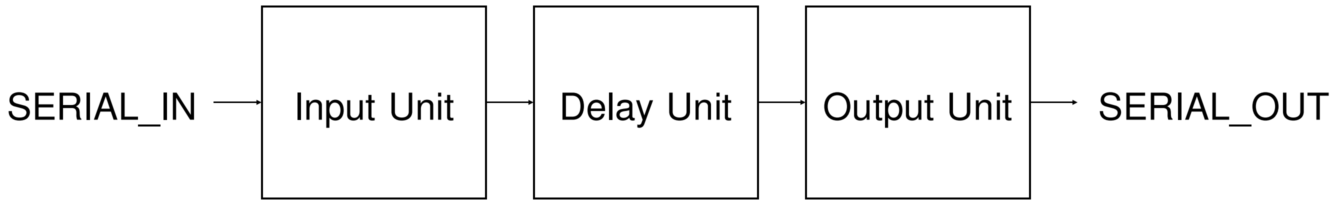
Control Transceiver

SBus Interface

	Information	Checksum	Postamble

Time

Link Delay Unit



- Operates at link speeds of up to possibly several hundred Mbps
- Input and output units consist of 32-bit shift registers
- Delay unit operates as 32x1M FIFO
- PLO lockup time is basic switching-time limitation
- May use out-of-band network synchronization for preliminary evaluation

Reducing Average Reservation Delays

- Techniques borrowed from parallel, discrete-event simulation (PDES)
  - Provides for control of network logical clock (not real-time)
  - Preserves relative ordering of events (reservations)
  - Requires analysis of state space and calculation of future dependencies as the result of received reservation requests
- Conservative (deterministic) techniques
  - Search state space to develop dependency graph (lookahead)
  - Advance logical clock when logical paths merge (Chandy-Misra)
  - Detects when bursts can be sent early while avoiding collisions
- Optimistic (nondeterministic) techniques
  - Remember hello bursts and state transitions for some interval
  - Detect schedule inconsistencies using message digest included in hello burst.
  - Notify hosts when inconsistency detected and rollback to prior stable state (timewarp)

Early Commit

- Assume network topology and individual link delays are known in advance
- When a reservation is heard, save it and determine when it will arrive at every other node until the destination
- At each arrival run the scheduling algorithm to see if the burst can be scheduled without conflict
  - If so and a schedule has not been computed, compute the schedule and delete the reservation
  - If so and a schedule has already been computed, delete the reservation
  - If not, save the reservation and continue
- When the reservation arrives at the destination, compute the schedule if not already computed, and delete the reservation,
- While this is compute-intensive, there are many heuristic optimizations

Reducing Vulnerability to Errors

- Note that nodes do not see the data, so don't know if collisions occur
- Assume that correct operation can be verified only by comparing the schedules computed by each node with that for all other nodes
- This can be done efficiently by including a crypto-checksum or message digest (e.g., MD5) of the calculated schedule in every hello burst
- Assume hello-burst synchronization is preserved, which amounts to an out-of-band signalling channel
- Assume bursts can only be lost (fail-stop) and uncorrupted (non-Byzantine) in replicated, communicating, finite-state machine model
- Recovery from errors uses the same technique as optimistic PDES by rolling back (possibly more than once) to a prior stable state and running through saved hello bursts until missing burst(s) are found
- This compute-intense operation is not expected to occur very often(!)

Synchronization Issues

- There are two ways to synchronize the network:
  - Using NTP and occasional timestamps exchanged in hello bursts
  - Using GPS or (cheap) LORAN-C timing receivers at each node
- Experiments with NTP and conventional workstations suggest hosts can be synchronized to the network to within a few tens of microseconds using software timestamps; this requires:
  - Minor kernel driver modifications which reduce timestamp latency
  - SunOS, Ultrix and OSF/1 kernel hacks to implement NTP phase-lock loop in the kernel,
- Where guard times in this order (i.e., a few kB at 1Gps), can be tolerated, this avoids the requirement for the messy enable signal, since the node can signal the host with the computed transmission time
- For better accuracy, SBus interface can be used as a disciplined (adjustable frequency) oscillator and set of software-readable counters

Lessons Learned

- Scaling up in size and speed can be easy or hard
  - The current rate of 1 ms/reservation is sufficient for an NSFnet topology with 13 nodes and 2-ms frames, but not much larger
  - One approach is to scale the frame size with the number of nodes; another is to use quasi-persistent scheduling in which the dwells are adjusted incrementally as a function of node utilization
- Synchronization turned out to be harder than it first appeared
  - All kinds of cruft appeared in the error budget due to sloppy hardware and kernel code
  - The NTP algorithms had to be intricately tuned to extend the range of operation to the microsecond regime
- The toughest nut to crack is probably the robustness issue
  - The techniques of PDES are particularly promising
  - So are the techniques of fault-tolerant distributed computing
  - However, fault propagation needs to be studied and bounds developed

Future Work

- Algorithms
  - Improve scheduler speeds
  - Reduce reservation delays (Early Commit)
  - Reduce vulnerability to errors and node crashes
  - Design multicast NTP for synchronization
- Hardware and software
  - Integrate strawman hardware and software components
  - Complete testing of Lowball board for Suns
- High^{2}ball
  - COMSAT Labs does all the work
  - We have all the fun

- DARPA/NASA Advanced Communication Technology Satellite (ACTS)
  - Satellite provides 622 Mbps in three satellite-switched beams to selected earth terminals
  - 3-meter antenna comes complete with 18-wheeler and concrete base
  - Satellite to be launched next year
  - Experiments run up to three years
- HPCCI Proposal ($1.1M over three years)
  - COMSAT Labs builds (3) 622-Mbps HPPI/SONET Multiplexors
  - The Multiplexors are installed along with matching SPARCstations at existing supercomputer sites
  - UDel ports the Highball technology to ACTS
  - UDel designs and conducts experiments using existing DARTNET technology

Summary

- Accomplishments
  - Designed, built and tested reservation and scheduling algorithms
  - Designed, built and tested precision synchronization technology
  - Designed, built and tested prototype SBus interface
  - Developed techniques to minimize scheduling delays and errors
- Present status
  - Second generation SBus FPGA interface complete and in test
  - Kernel precision clock driver complere and in test
  - NTP modifications for precision clock complete and in test
- Future plans
  - Integrate node controller, software driver and control program
  - Complete FPGA firmware for switch controller and transceiver
  - Complete design and implementation of protocol software
- Further information: http://www.eecis.udel.edu/~mills