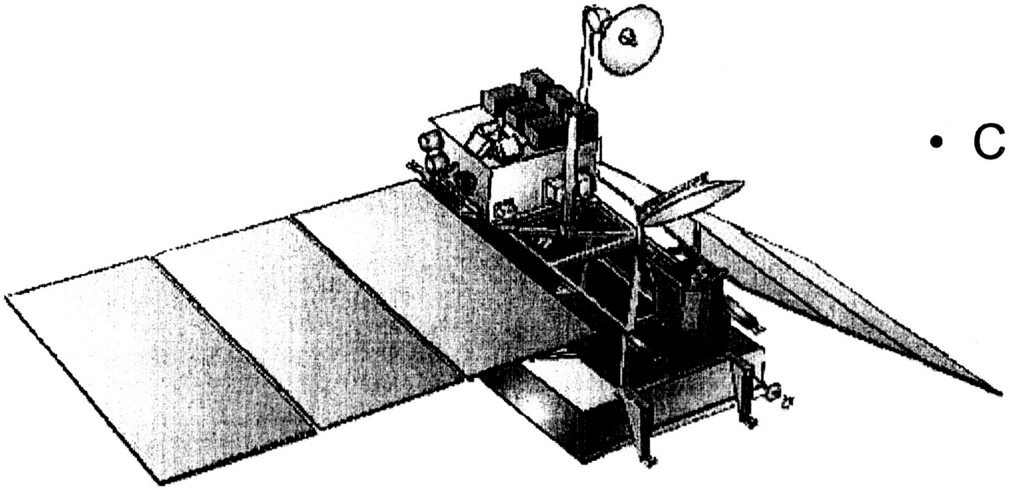
Building a Reliable Onboard Network with Ethernet: A GSFC Prototype

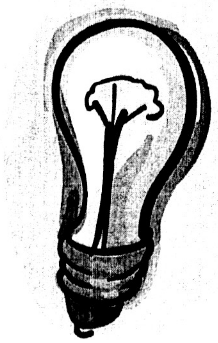
Jane Marquart
NASA/GSFC

Requirements



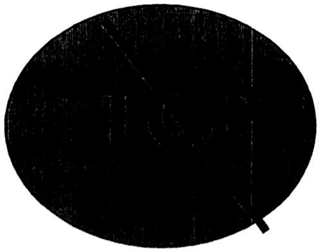
- Critical real-time data must be delivered **reliably** onboard the spacecraft

+

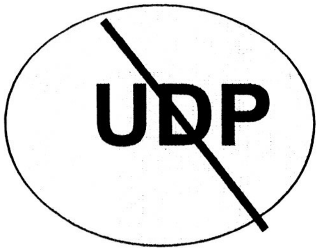


- Ethernet/IP Onboard Network Bus

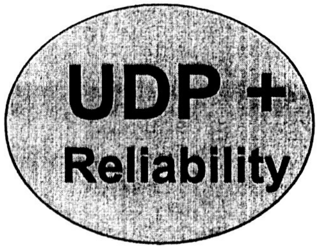
Protocol Options



TCP works but doesn't support reliability features required by typical flight software



UDP is a better fit, but requires reliability be implemented by the user



Where to implement reliability? Application layer or data link layer?

Reliability Trade

- Application layer:
  - Rx & tx packet latency through the stack is highly variable and non-deterministic (regardless of direction)
  - Latency up to several milliseconds on a MCP750 @ 233mhz
- Datalink layer:
  - avoids IP stack latency, but;
  - the issue is standardization and portability of a datalink layer solution.

IEEE 802.3 LLC

- **LLC** – “Logical Link Control” offers a IEEE standardized datalink layer reliability protocol, adding a 3 byte header to the ethernet frame

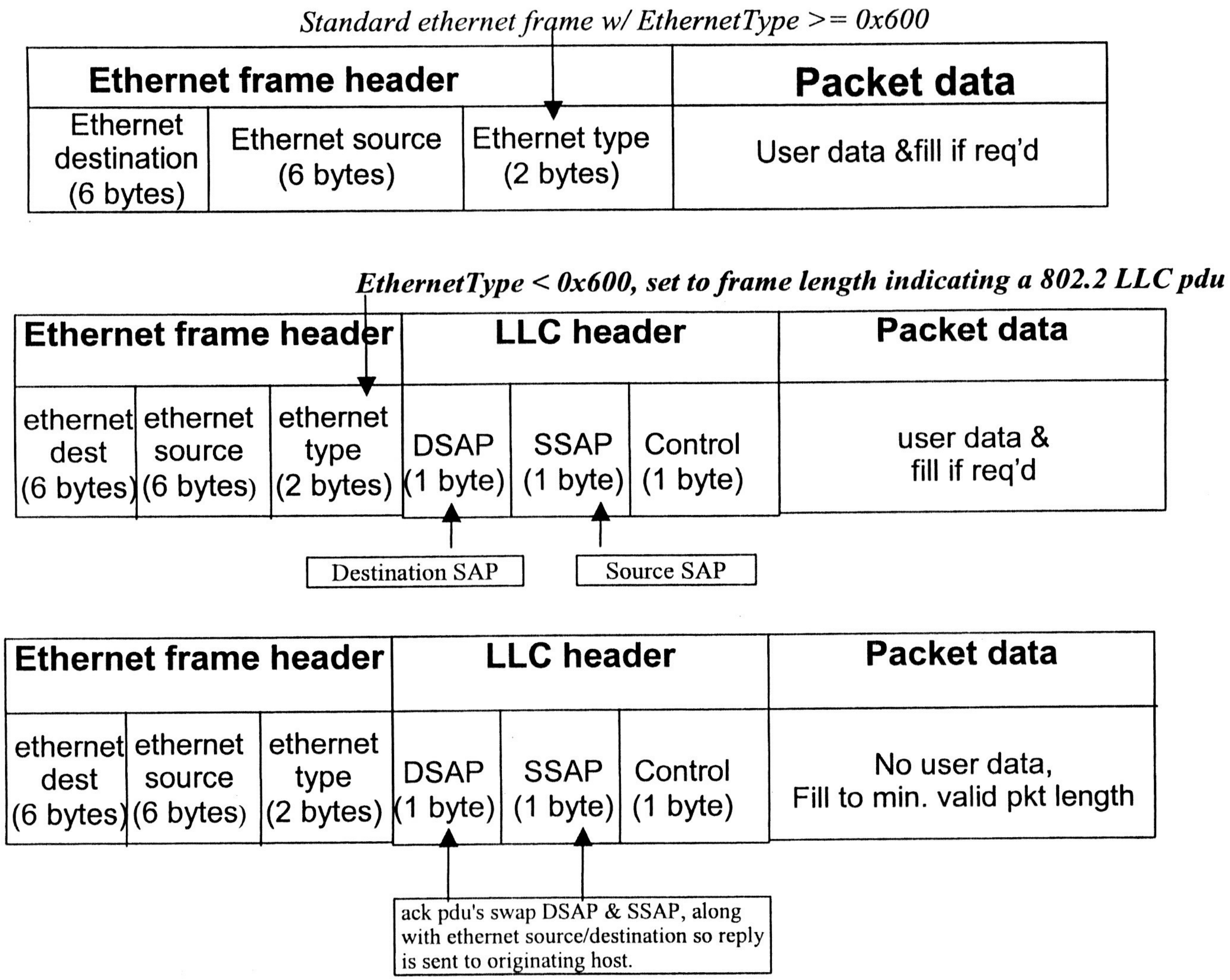
- **Supplies 3 major types of service;**
  - **Type 1 (unreliable packet exchange, same as plain ethernet)**
  - **Type 2 (reliable, statefull, connection oriented; used by FDDI, Wireless, Token Ring)**
  - **Type 3 (reliable, stateless, connectionless)**

- **Type 3 selected**
  - **Stateless but reliable**
  - **Well-defined (and simple) state machine**
  - **Suitable for a variety of physical layers**
  - **Packet overhead is 3 bytes**

Data-link Layer Reliability: LLC

- Ack/retry algorithm implemented within the NIC driver.
- Latency & delay well defined since control is right at the hardware.
- Widely supported, if not tolerated, by common OS IP stacks, routers, etc.
- Well defined & supported but ad-hoc mapping onto standard ethernet framing.
- Reliability protocol only useable between conformant drivers.

PDU Formats



Standard ethernet frame w/ EthernetType >= 0x600

Default datalink pdu

Ethernet frame header			Packet data
Ethernet destination (6 bytes)	Ethernet source (6 bytes)	Ethernet type (2 bytes)	User data &fill if req'd

EthernetType < 0x600, set to frame length indicating a 802.2 LLC pdu

Reliable pdu (data packet)

Ethernet frame header			LLC header			Packet data
ethernet dest (6 bytes)	ethernet source (6 bytes)	ethernet type (2 bytes)	DSAP (1 byte)	SSAP (1 byte)	Control (1 byte)	user data & fill if req'd

Destination SAP

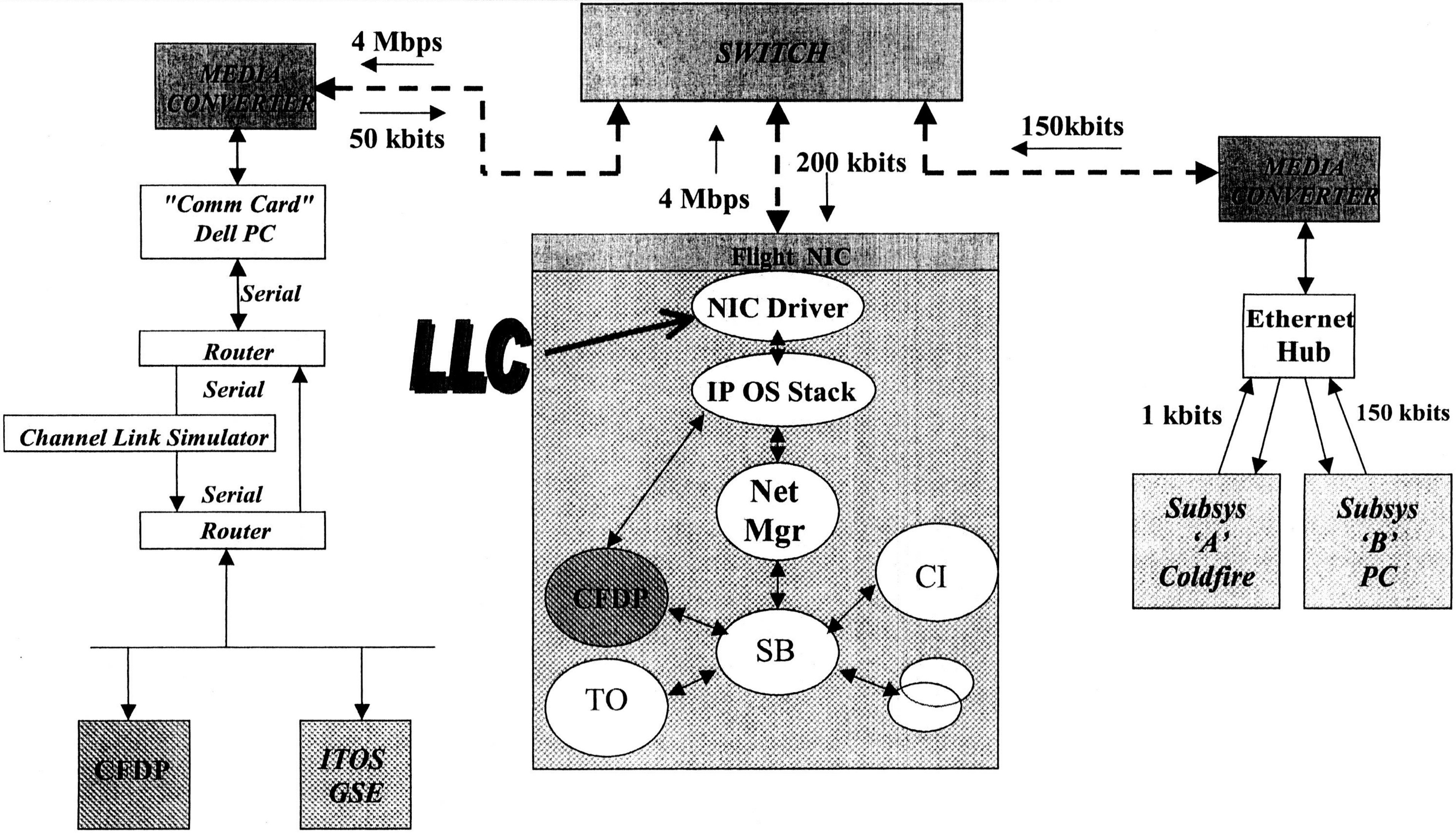
Source SAP

Reliable pdu (ack packet)

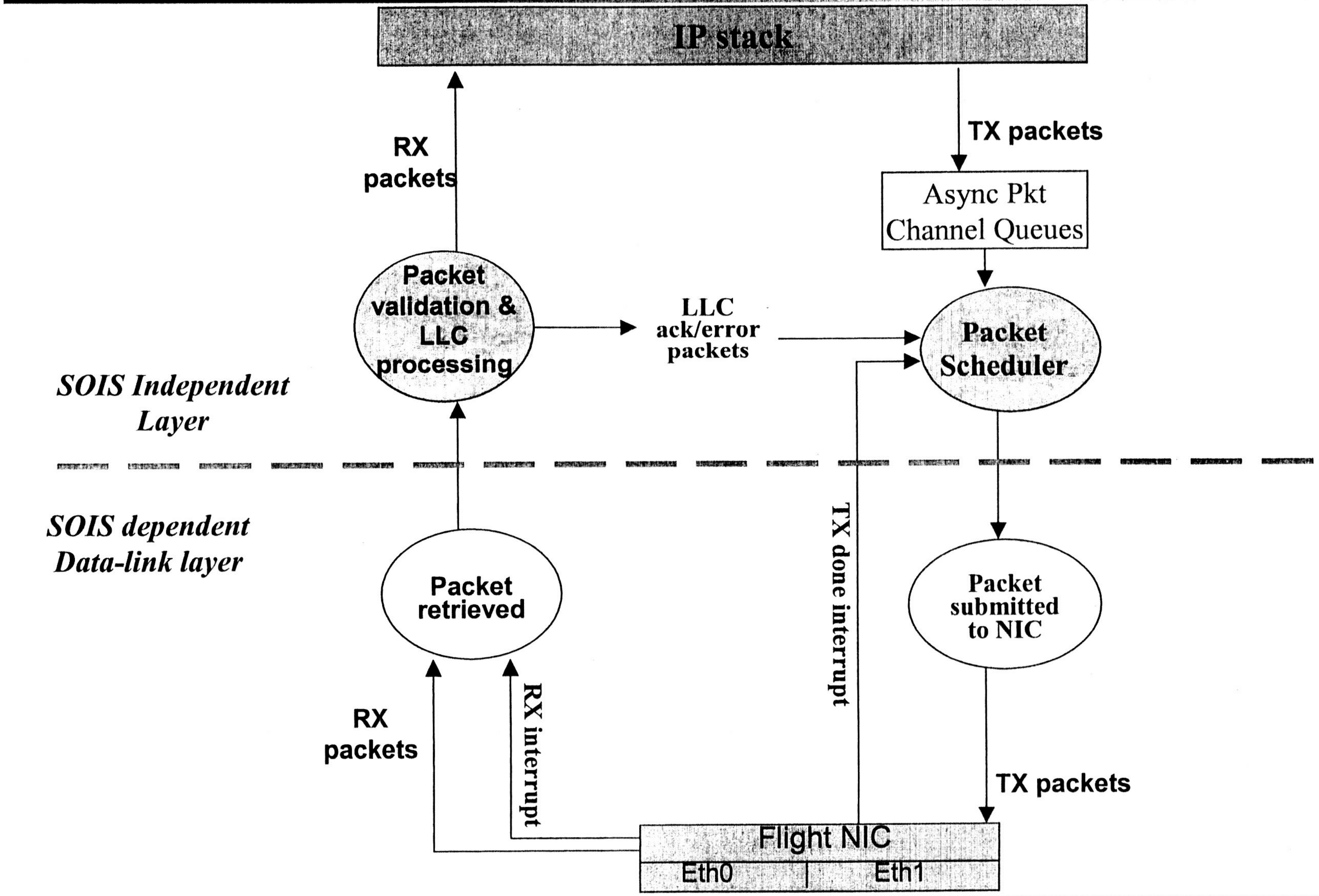
Ethernet frame header			LLC header			Packet data
ethernet dest (6 bytes)	ethernet source (6 bytes)	ethernet type (2 bytes)	DSAP (1 byte)	SSAP (1 byte)	Control (1 byte)	No user data, Fill to min. valid pkt length

ack pdu's swap DSAP & SSAP, along with ethernet source/destination so reply is sent to originating host.

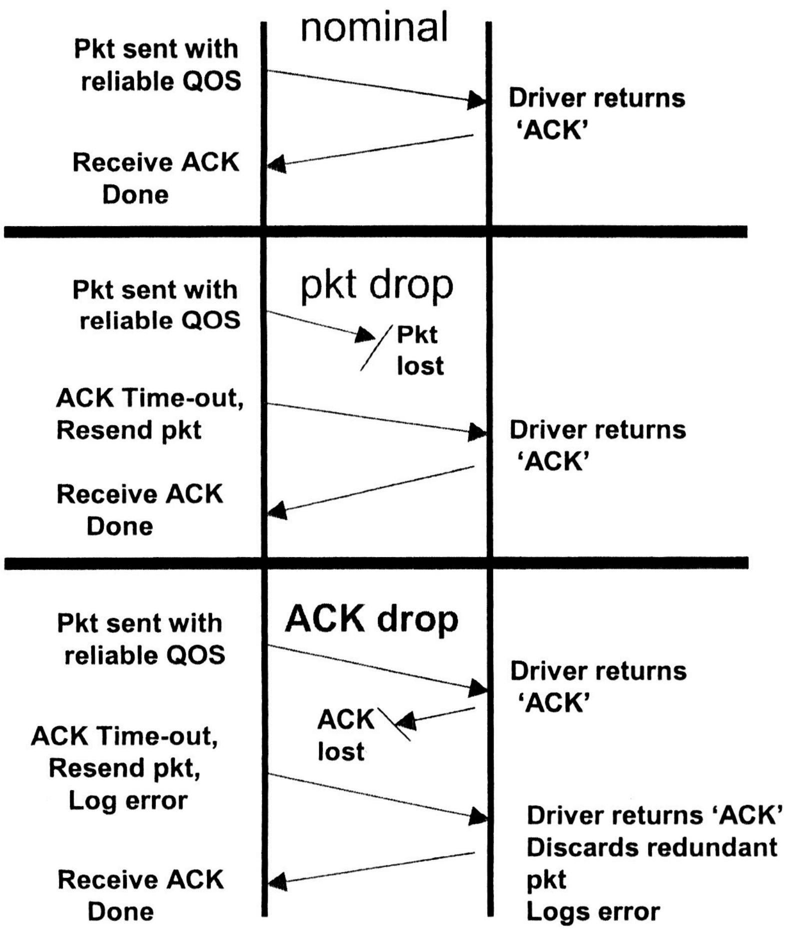
Ethernet/IP Multi-node Testbed



NIC Driver Architecture



Use Case Examples



nominal

Pkt sent with reliable QOS

Driver returns 'ACK'

Receive ACK Done

pkt drop

Pkt sent with reliable QOS

Pkt lost

ACK Time-out, Resend pkt

Driver returns 'ACK'

Receive ACK Done

ACK drop

Pkt sent with reliable QOS

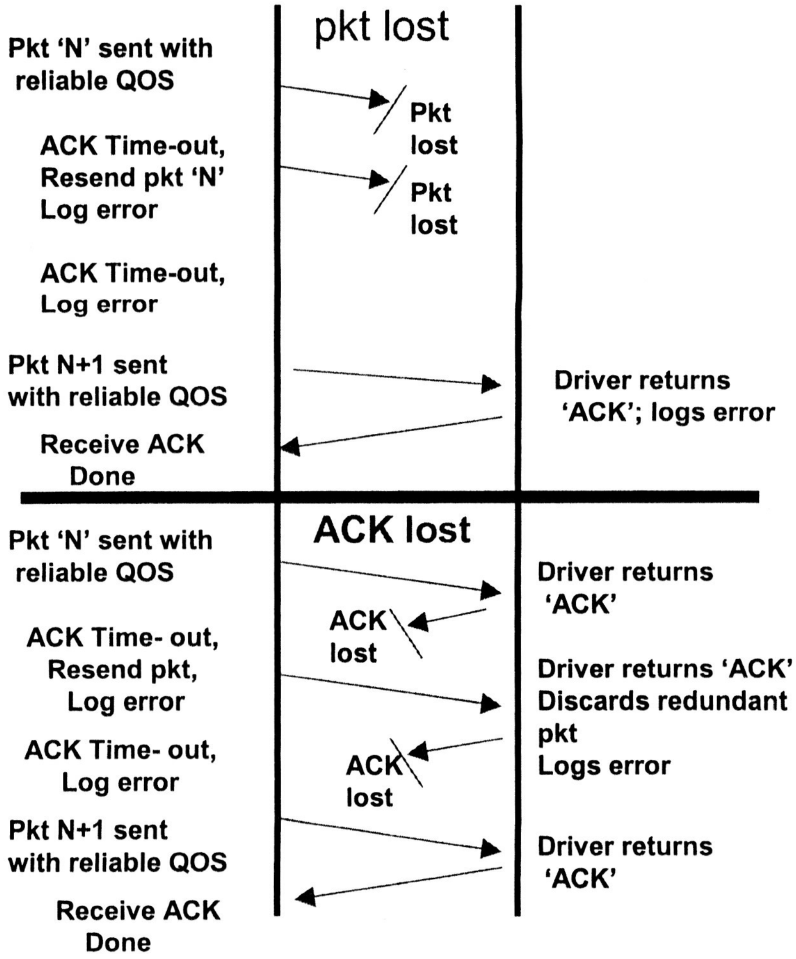
Driver returns 'ACK'

ACK lost

ACK Time-out, Resend pkt, Log error

Driver returns 'ACK' Discards redundant pkt Logs error

Receive ACK Done



pkt lost

Pkt 'N' sent with reliable QOS

Pkt lost

ACK Time-out, Resend pkt 'N' Log error

Pkt lost

ACK Time-out, Log error

Pkt N+1 sent with reliable QOS

Driver returns 'ACK'; logs error

Receive ACK Done

ACK lost

Pkt 'N' sent with reliable QOS

Driver returns 'ACK'

ACK lost

ACK Time- out, Resend pkt, Log error

Driver returns 'ACK' Discards redundant pkt Logs error

ACK Time- out, Log error

ACK lost

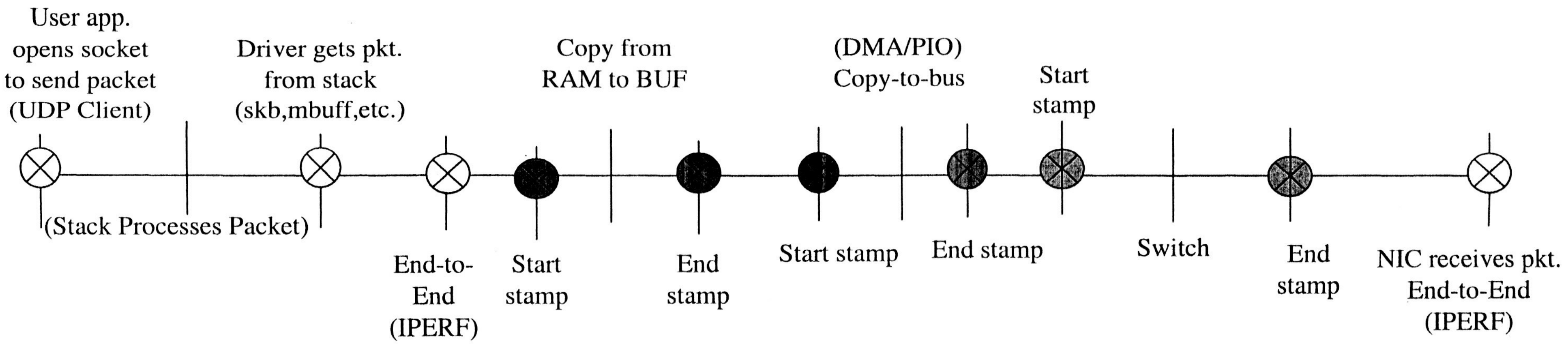
Pkt N+1 sent with reliable QOS

Driver returns 'ACK'

Receive ACK Done

Timeline for NIC/OS Measurements

Transmit Path



Key:

⊗ = pending for future revisions

● = timestamp marker (software)

⊗ = logic analyzer timestamp

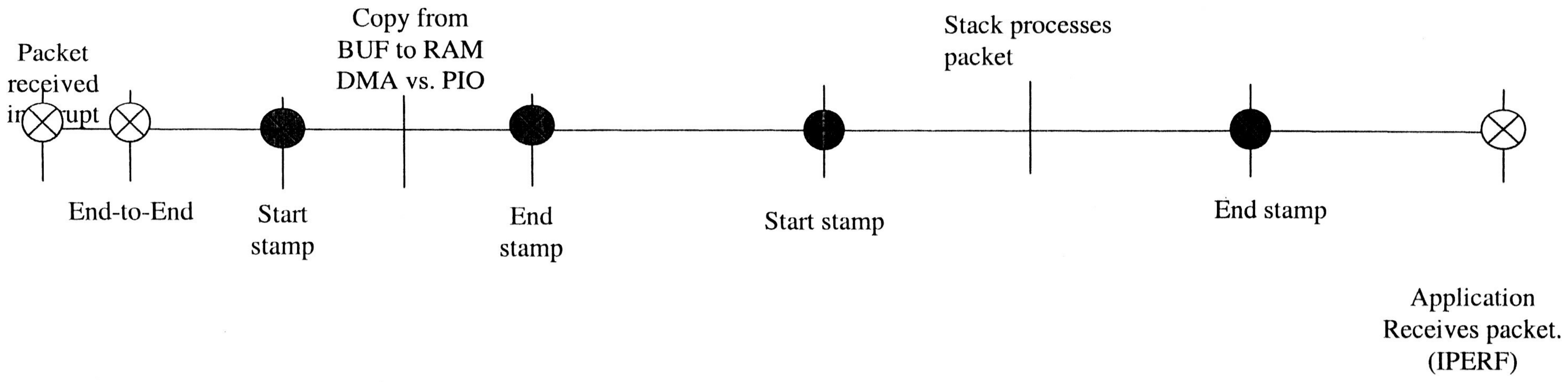
Proposed Metrics:

1.) Stack Process Time

2.) Driver SRAM copy function
3.) DMA vs. Programmed I/O time
4.) Interrupt Latency
5.) Network Bus Latency w/jitter
6.) Switch Latency

Timeline for NIC/OS Measurements

Receive Path


Packet received interrupt
End-to-End
Start stamp
Copy from BUF to RAM DMA vs. PIO
End stamp
Start stamp
Stack processes packet
End stamp
Application Receives packet. (IPERF)

Key:

⊗ = pending for future revisions

● = timestamp marker (software)

◍ = logic analyzer timestamp

Metrics:

1.) Stack Process Time

2.) Driver SRAM Copy Function

3.) DMA vs. Programmed I/O time

4.) Interrupt Latency

RTEMS Performance Metrics

Receive Path:

1.) Stack Process Time(packet hand-off to application) = TBD

2.) Driver SRAM Copy BUF-to-RAM = TBD
3.) DMA vs. Programmed I/O read-from-bus = TBD

Transmit Path:

1.) Stack Process Time(application hand-off to driver) = TBD

2.) Driver SRAM copy RAM-to-BUF = TBD
3.) DMA vs. Programmed I/O write-to-bus = TBD

End-to-End:

1.) Switch Latency = TBD
2.) End-to-End (IPERF/NUTTCP) = TBD

The Team

Software

Greg Menke

Hardware

Mike Lin, Code 561

Scott Edfors, Code 561

Prototype contributors

Art Ferrer

Freemon Johnson

Alan Cudmore

Jonathan Wilmot

Jane Marquart

For more information contact: Jane Marquart at
jane.marquart@nasa.gov