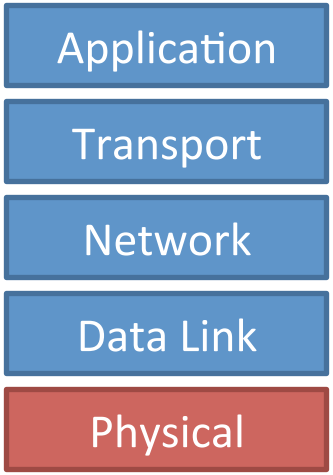
Cornell University

SoNIC: Precise Realtime Software Access and Control of Wired Networks

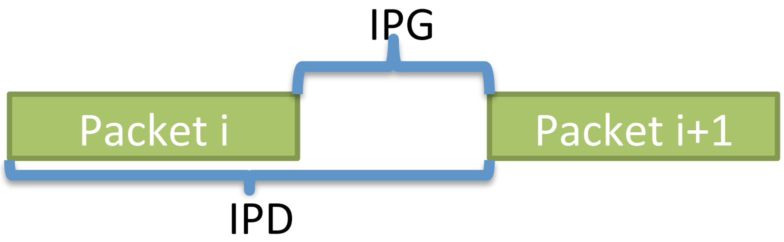
Ki Suh Lee, Han Wang, Hakim Weatherspoon

Cornell University

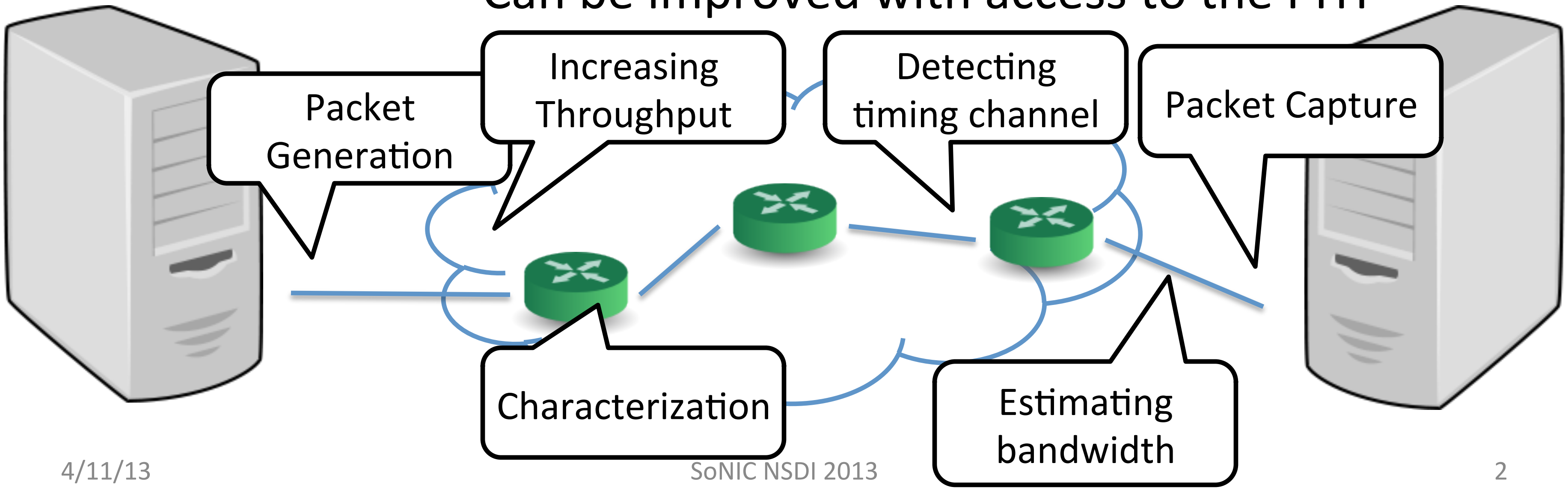
Interpacket Delay and Network Research



- Interpacket gap, spacing, arrival time, …



- Important metric for network research
  - Can be improved with access to the PHY



Network Research enlightened via the PHY

Application

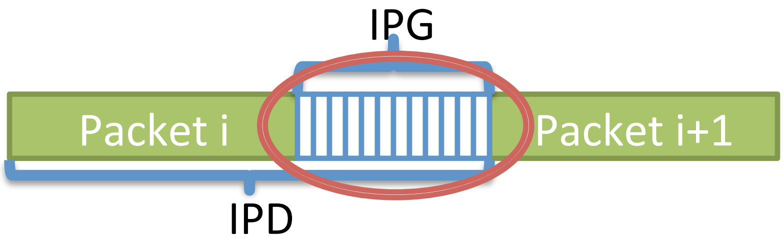
Transport

Network

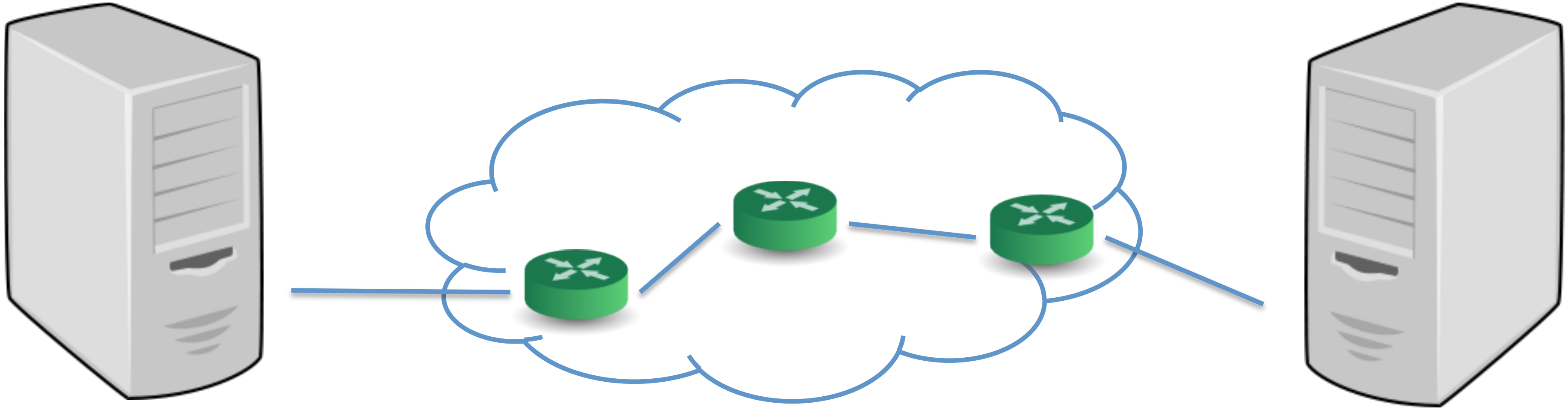
Data Link

Physical

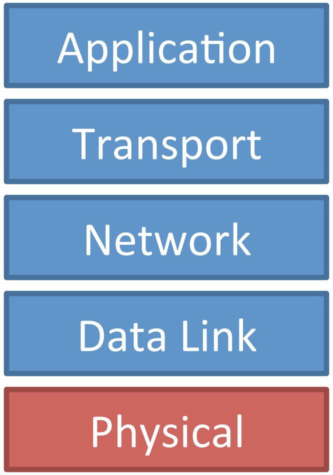
- Valuable information: Idle characters



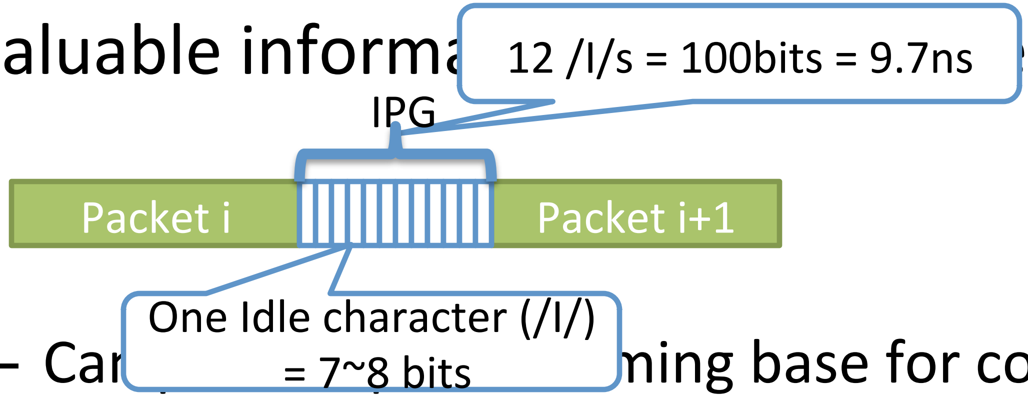
  – Can provide precise timing base for control
    - Each bit is ~97 *ps* wide

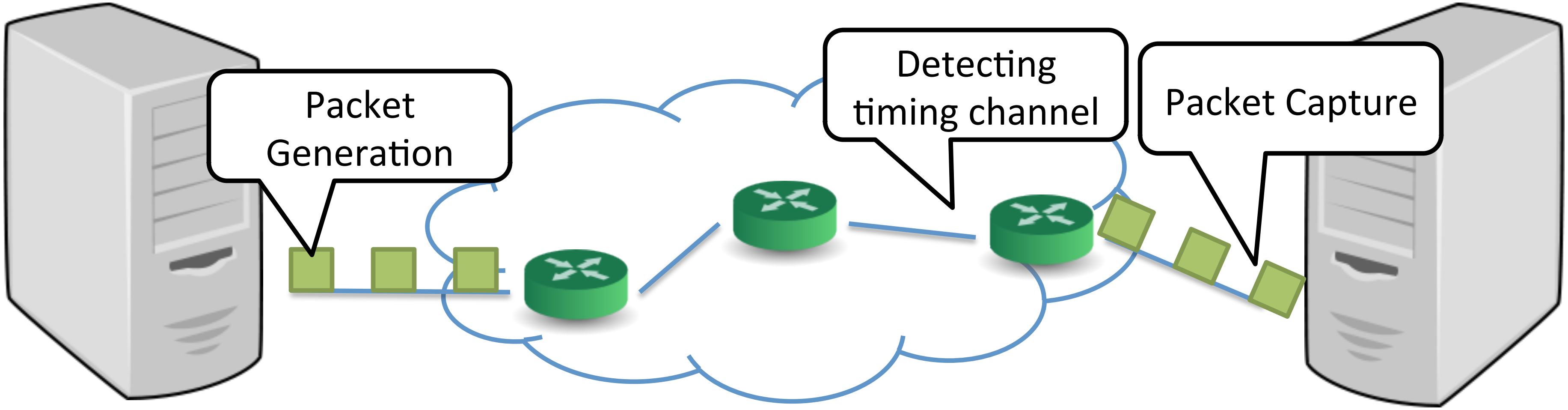


Network Research enlightened via the PHY



- Valuable informa rs
  - Car ming base for control
    - Each bit is ~97 *ps* wide

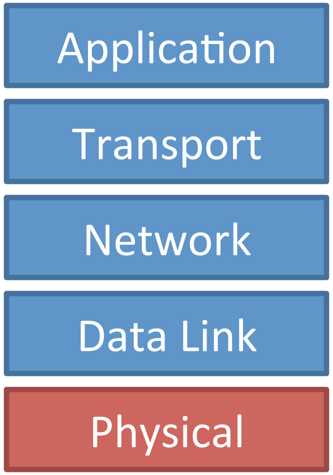




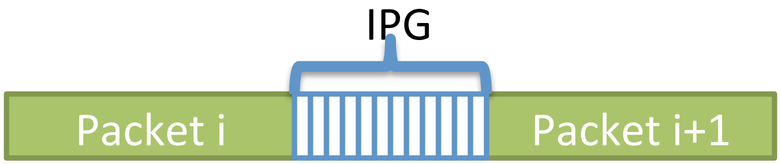
Principle #1: Precision

Precise network measurements is enabled via access to the physical layer (and the idle characters and bits within interpacket gap)

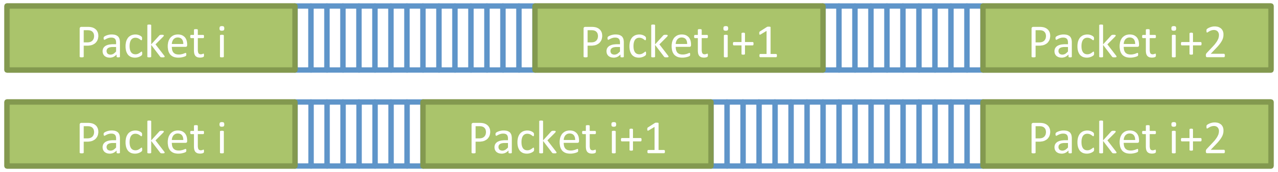
How to control the idle characters (bits)?



- Access to the entire stream is required



- Issue1: The PHY is simply a black box
  - No interface from NIC or OS
  - Valuable information is invisible (discarded)

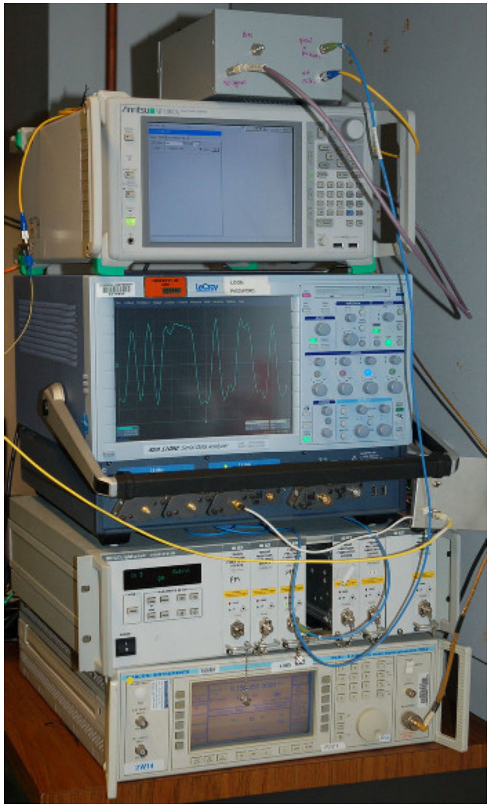


- Issue2: Limited access to hardware
  - We are network systems researchers
    a.k.a. we like software

Principle #2: Software

Network Systems researchers need software access to the physical layer

Precision + Software = Physics equipment???

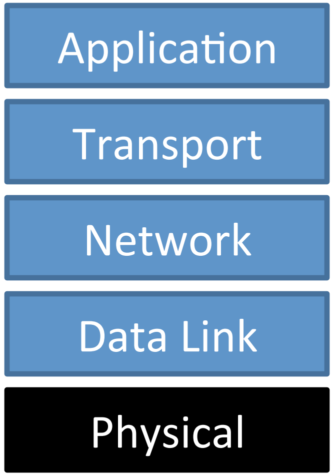


- BiFocals [IMC'10 Freedman, Marian, Lee, Birman, Weatherspoon, Xu]
  - Enabled novel network research
  - Precision + Software =
    Laser + Oscilloscope + Offline analysis
  - Allowed precise control in software
- Limitations
  - Offline (not *realtime*)
  - Limited Buffering
  - Expensive

Principle #3: Realtime

Network systems researchers need access and control of the physical layer (interpacket gap) continuously in realtime

Challenge



Application

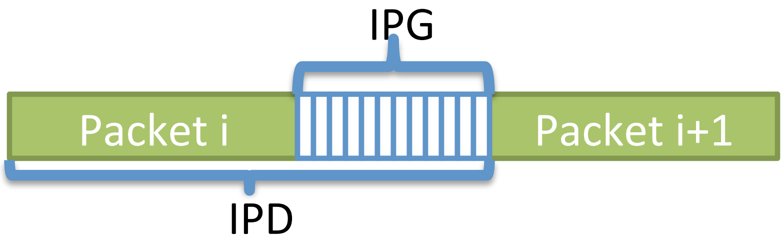
Transport

Network

Data Link

Physical

- Goal: Control *every* bit in *software* in *realtime*



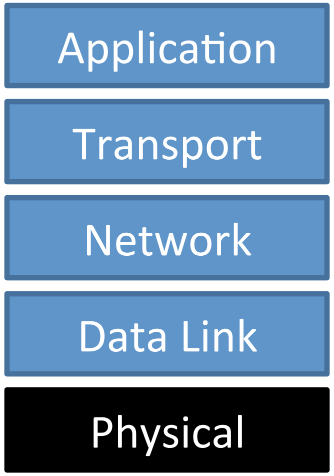
  - Enable novel network research

- Challenge
  - Requires unprecedented software access to the PHY

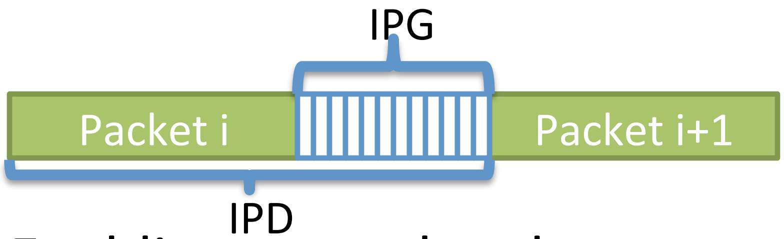
Outline

- Introduction
- SoNIC: Software-defined Network Interface Card
  - Background: 10GbE Network Stack
  - Design
- Network Research Applications
- Conclusion

SoNIC: Software-defined Network Interface Card

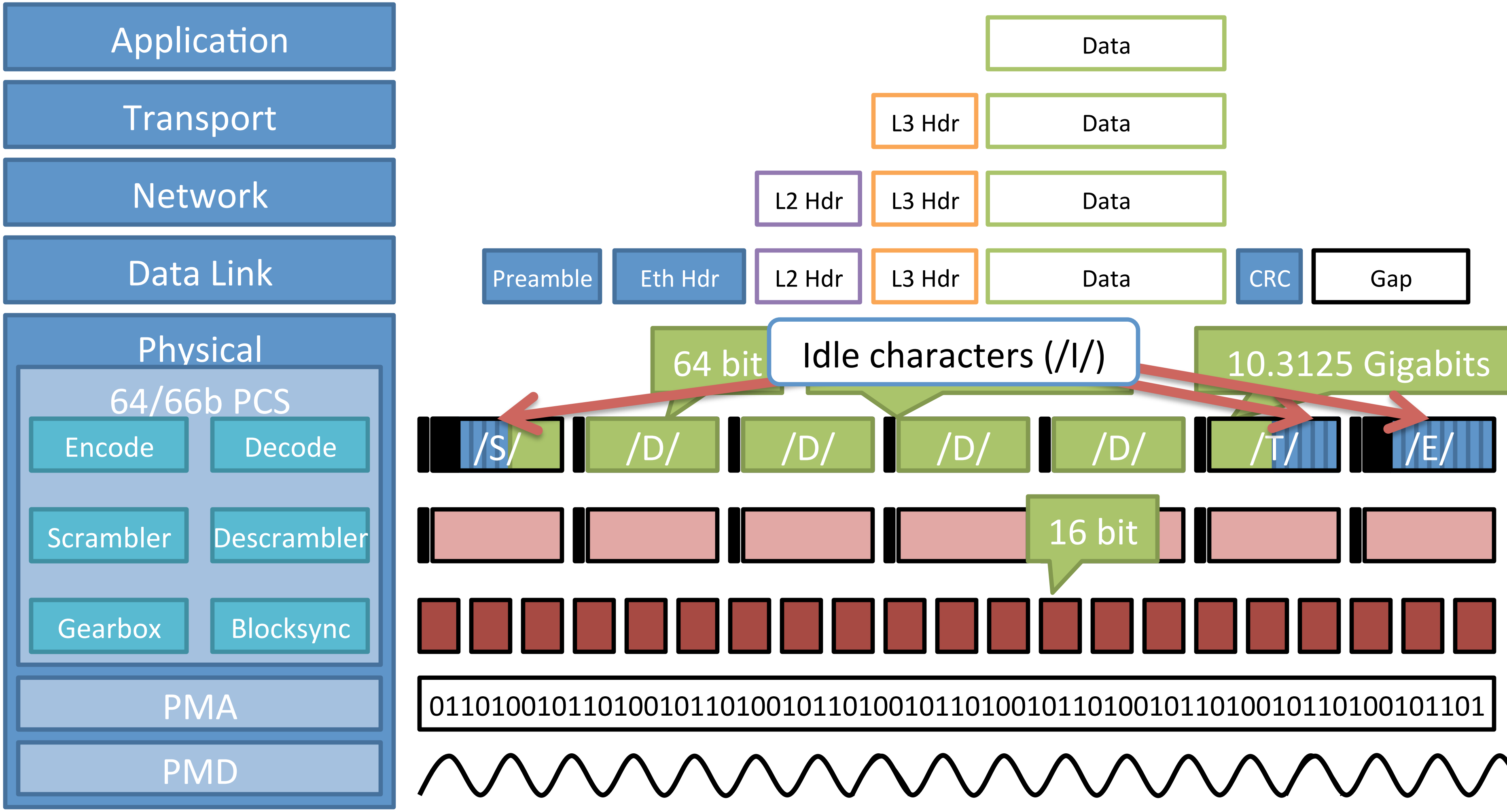


- Implements the PHY in software

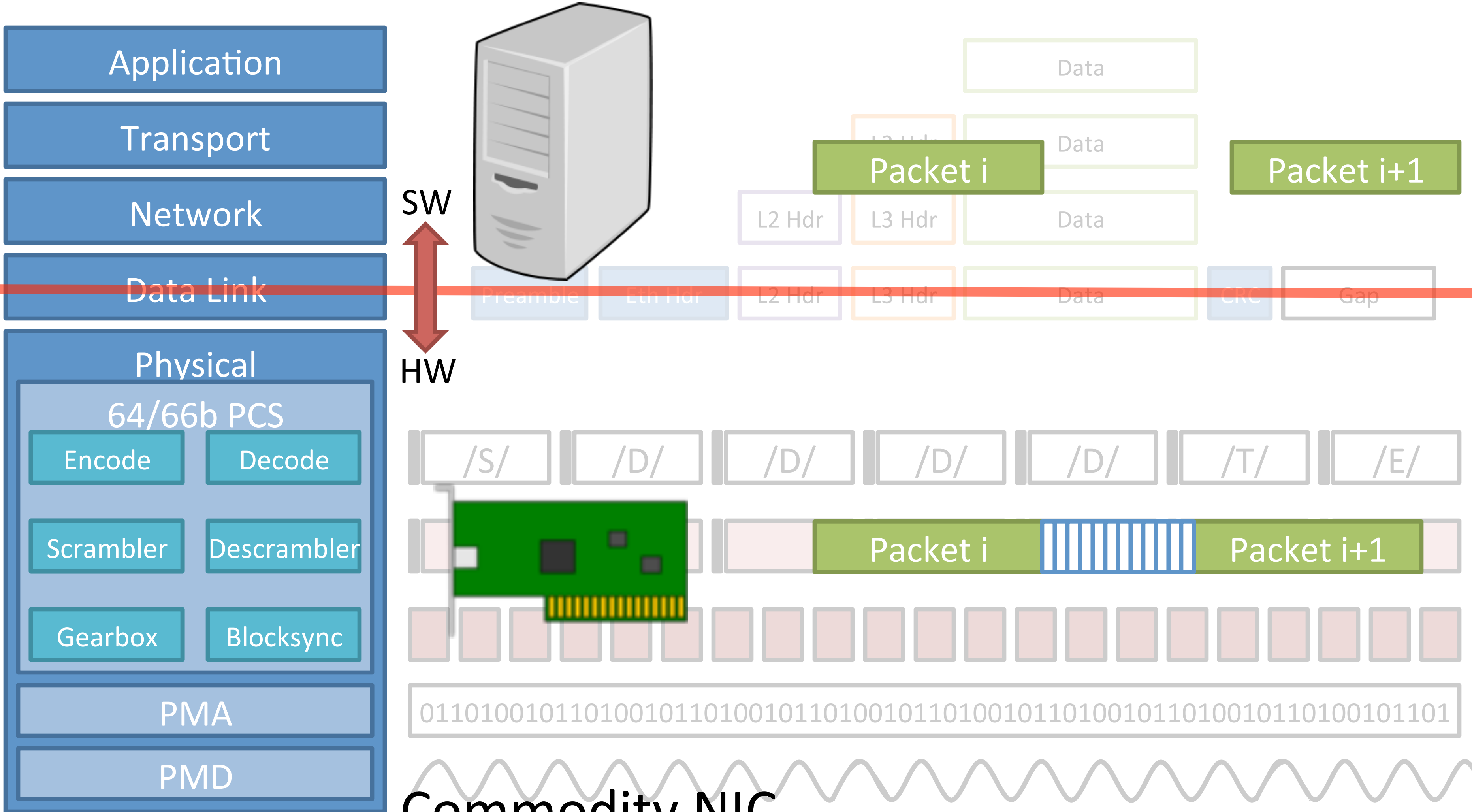


  - Enabling control and access to every bit in realtime
  - With commodity components
  - Thus, enabling novel network research
- How?
  - Backgrounds: 10 GbE Network stack
  - Design and implementation
    - Hardware & Software
    - Optimizations

10GbE Network Stack

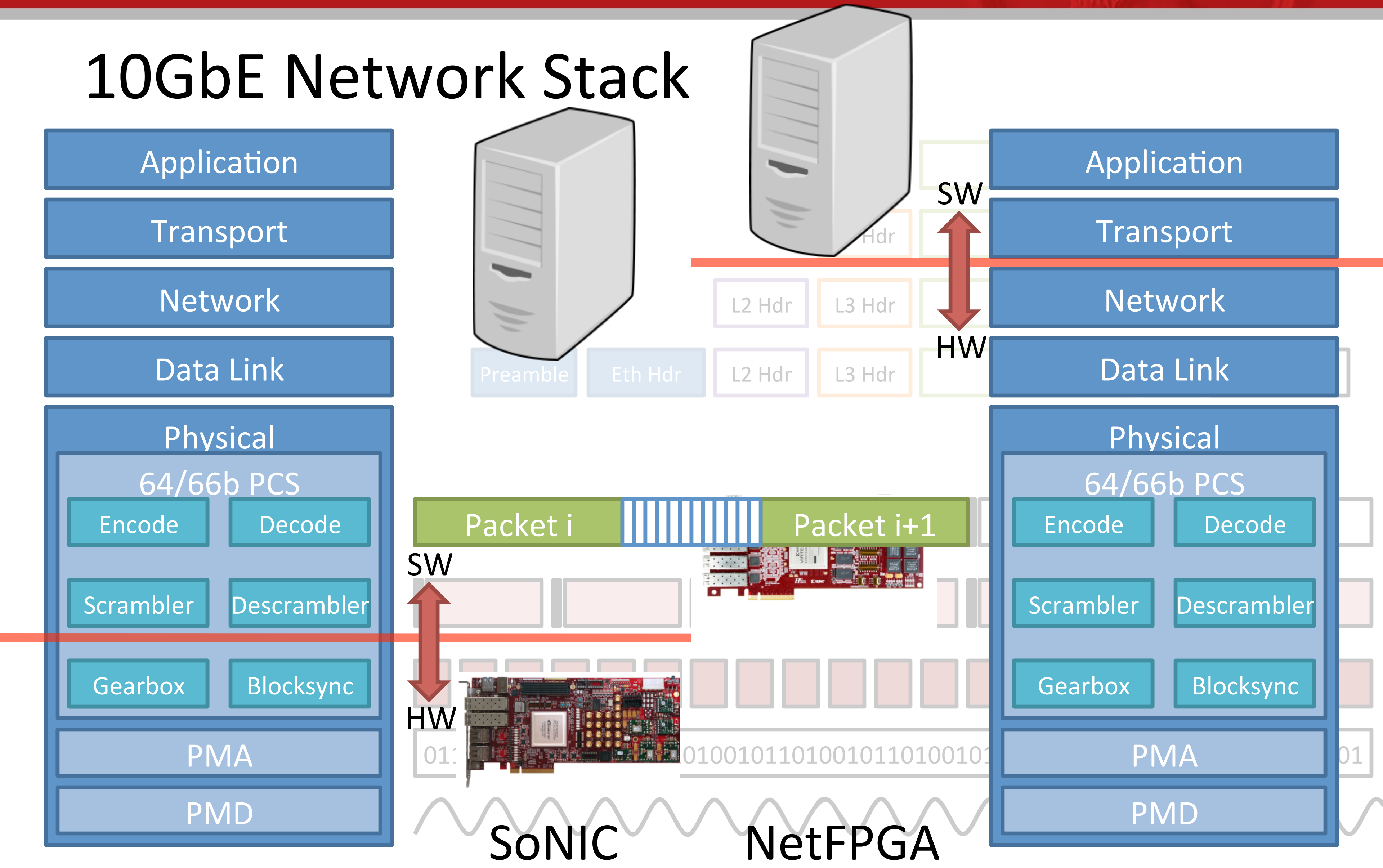


10GbE Network Stack



Commodity NIC

10GbE Network Stack



Application

Transport

Network

Data Link

Physical

64/66b PCS

Encode | Decode

Scrambler | Descrambler

Gearbox | Blocksync

PMA

PMD

Application

Transport

Network

Data Link

Physical

64/66b PCS

Encode | Decode

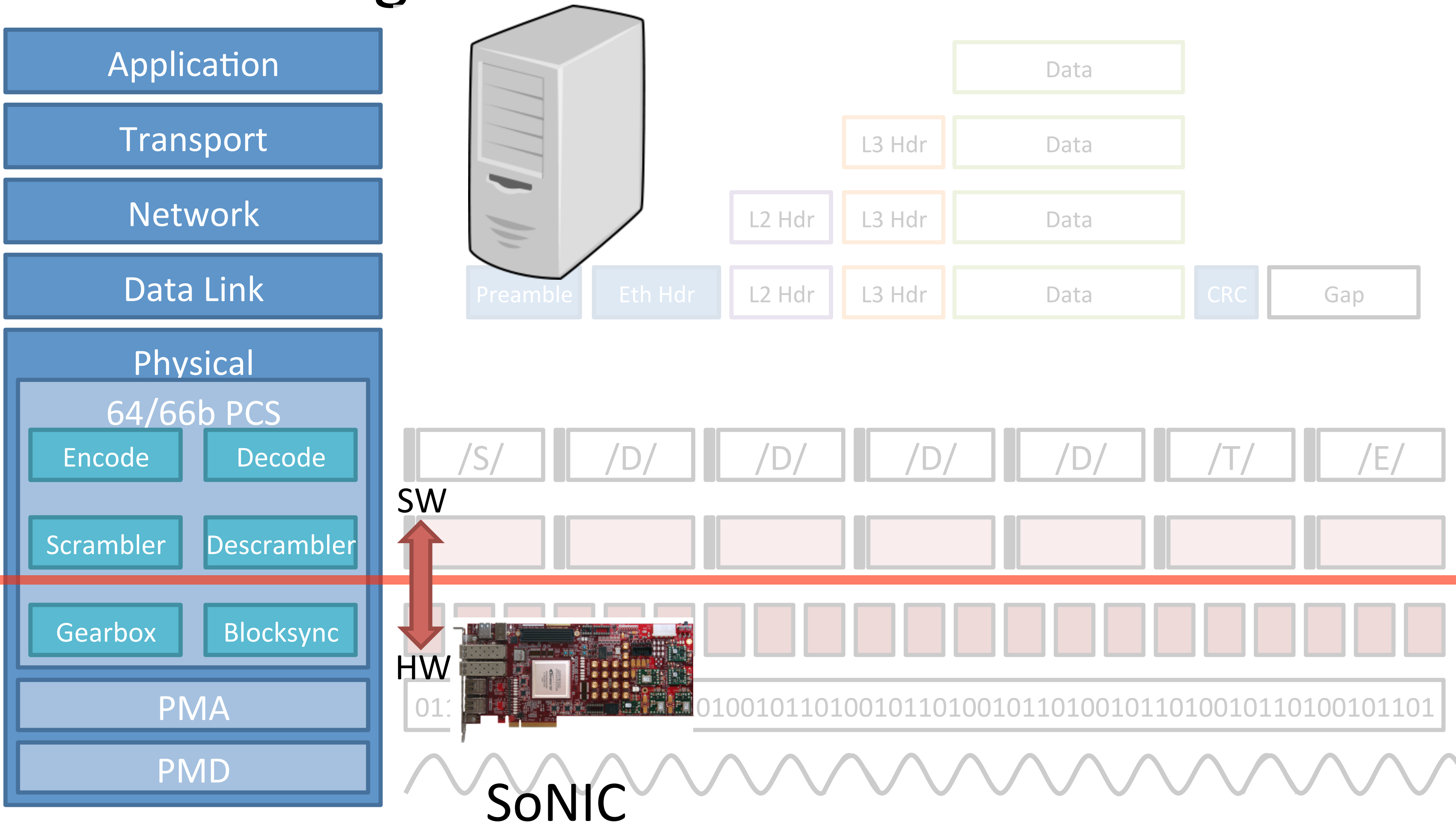
Scrambler | Descrambler

Gearbox | Blocksync

PMA

PMD

SoNIC Design



Application

Transport

Network

Data Link

Physical

64/66b PCS

Encode

Decode

Scrambler

Descrambler

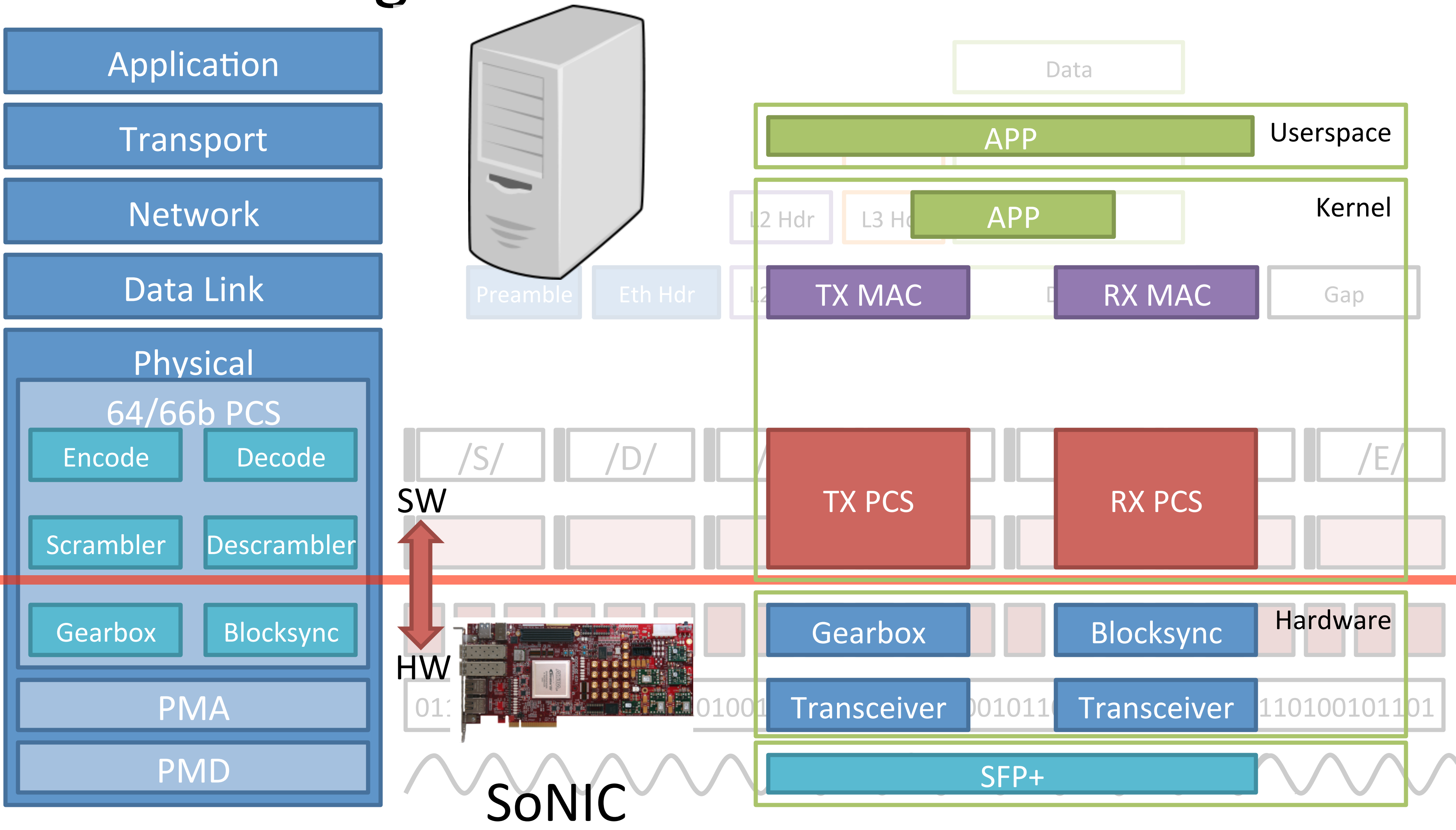
Gearbox

Blocksync

PMA

PMD

SoNIC Design and Architecture



Application

Transport

Network

Data Link

Physical

64/66b PCS

Encode

Decode

Scrambler

Descrambler

Gearbox

Blocksync

PMA

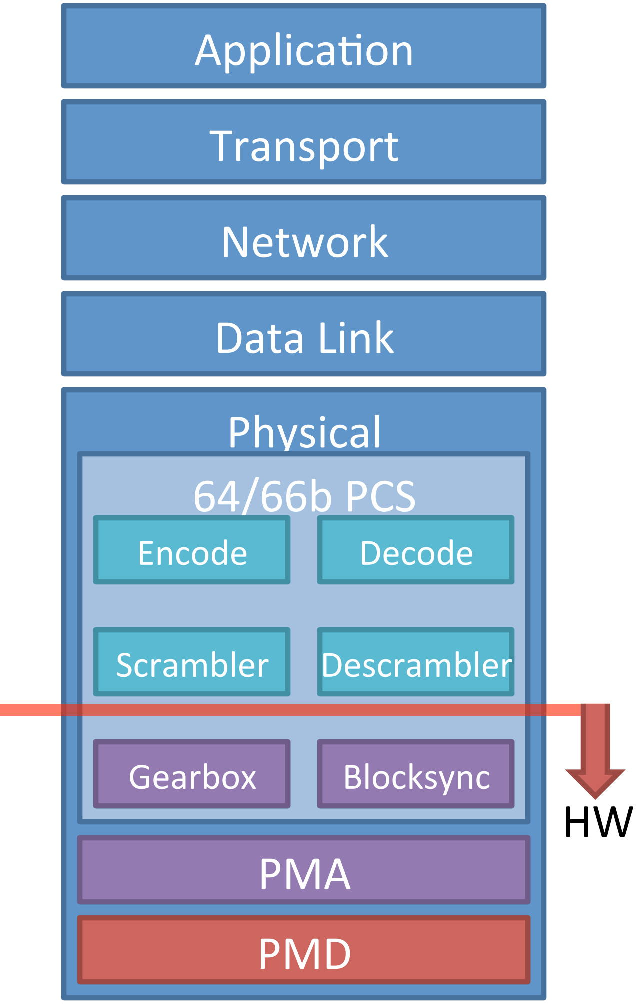
PMD

SW

HW

SoNIC

SoNIC Design: Hardware



Application

Transport

Network

Data Link

Physical

64/66b PCS

Encode

Decode

Scrambler

Descrambler

Gearbox

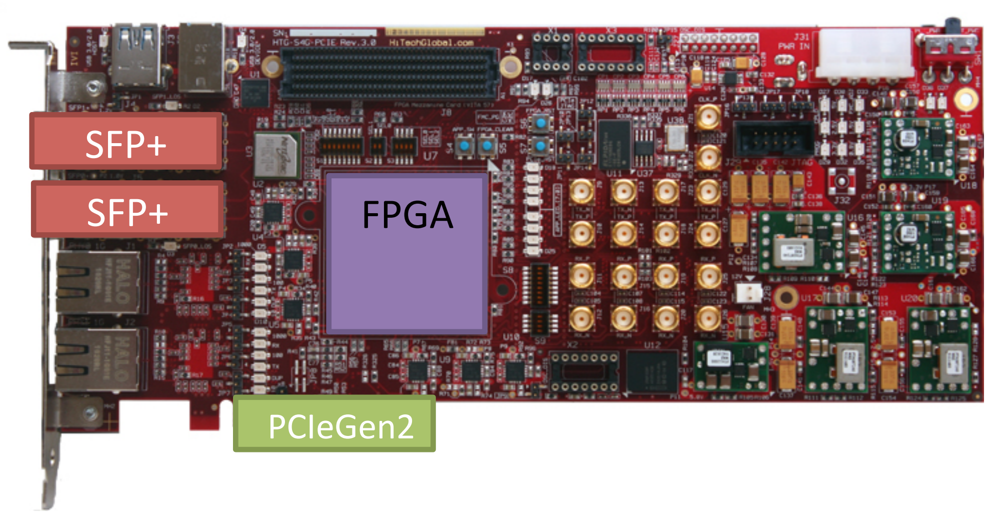
Blocksync

PMA

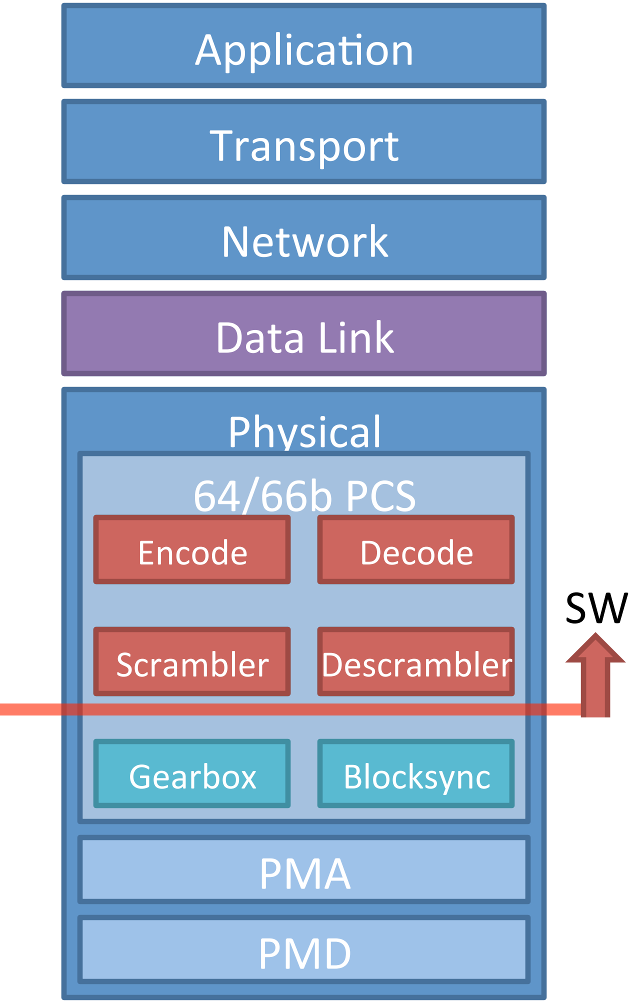
PMD

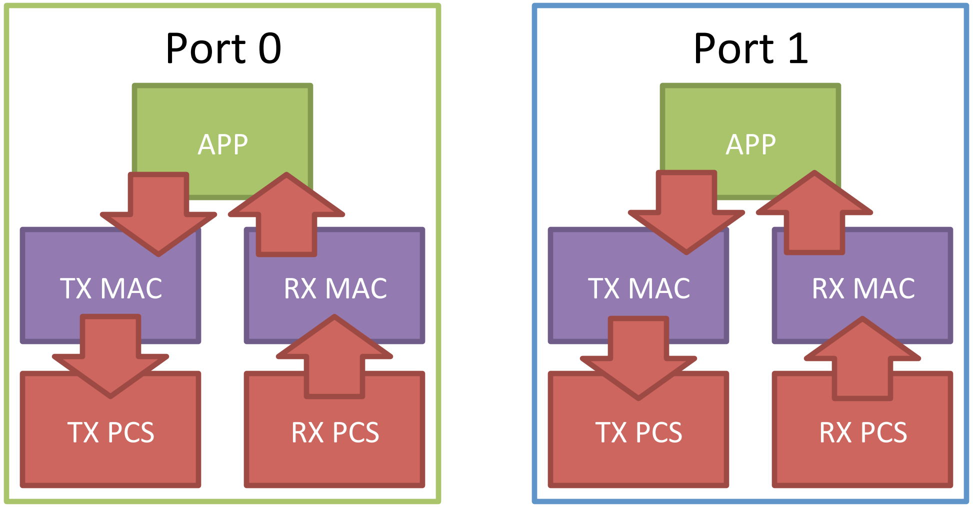
HW

- To deliver every bit from/to software
  - High-speed transceivers
  - PCIe Gen2 (=32Gbps)
- Optimized DMA engine

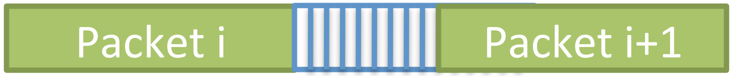


SoNIC Design: Software

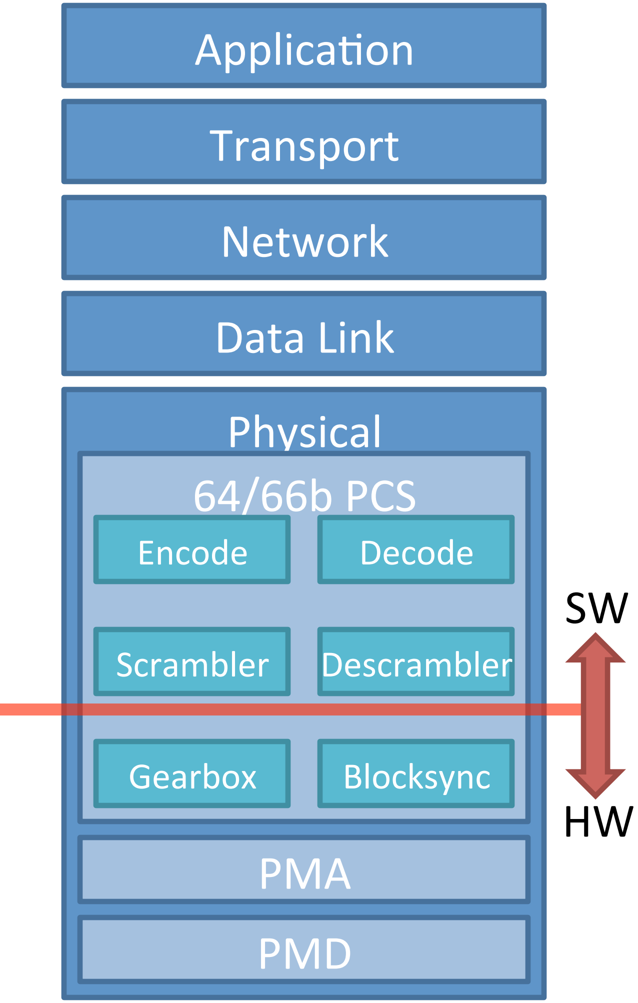


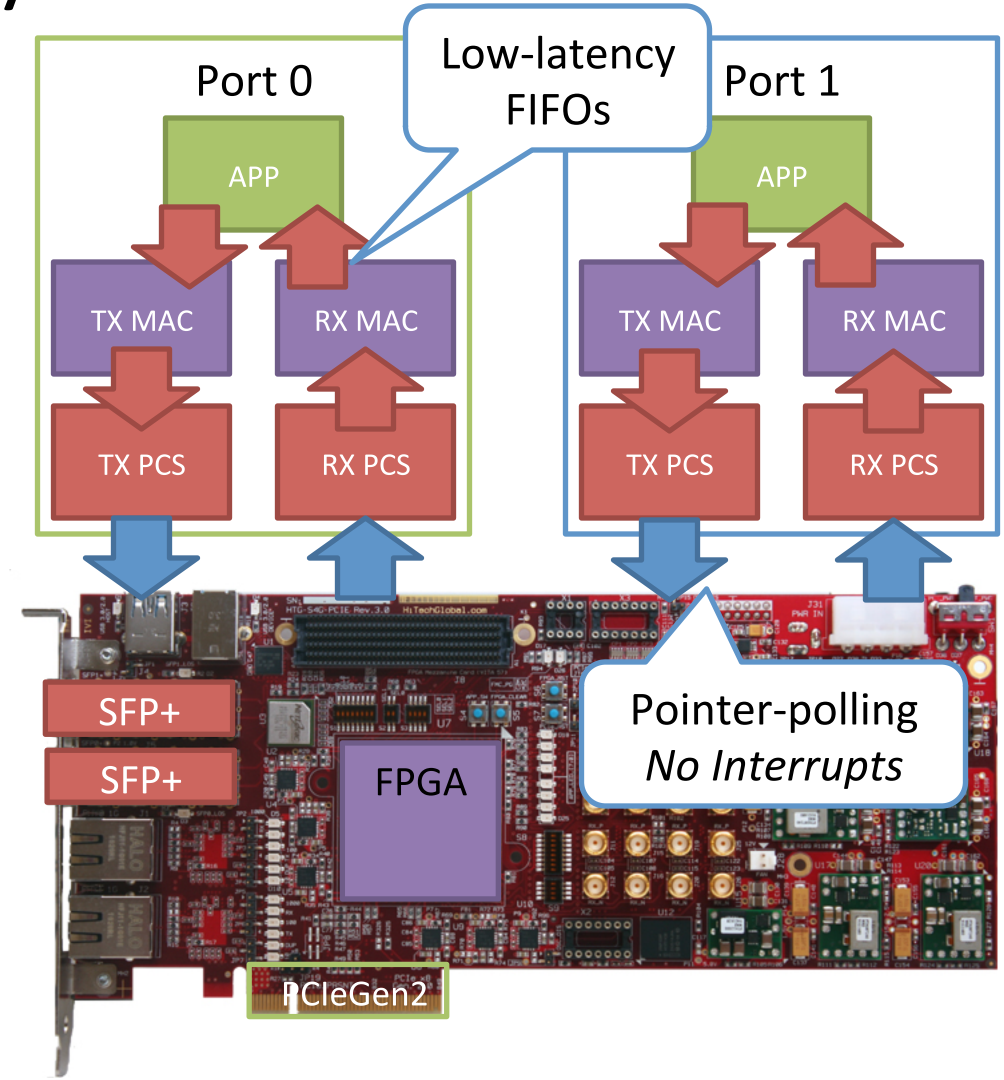


- Dedicated Kernel Threads
  - TX / RX PCS, TX / RX MAC threads
  - APP thread: Interface to userspace

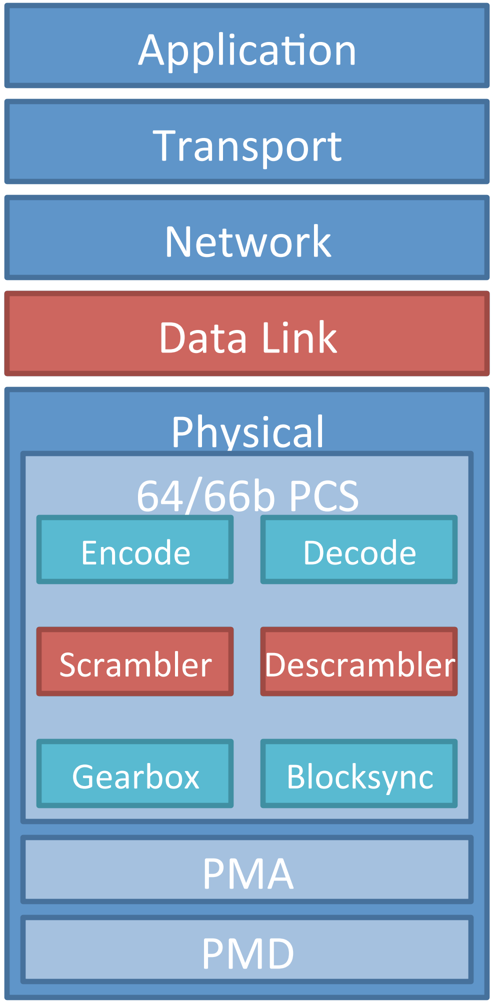


SoNIC Design: Synchronization





SoNIC Design: Optimizations



Application

Transport

Network

Data Link

Physical

64/66b PCS

Encode

Decode

Scrambler

Descrambler

Gearbox

Blocksync

PMA

PMD

- Scrambler $G(x) = x^{58} + x^{39} + 1$

Naïve Implementation	Optimized Implementation
$s \leftarrow$ state $d \leftarrow$ data **for** $i = 0 \rightarrow 63$ **do** $in \leftarrow (d \gg i) \,\&\, 1$ $out \leftarrow (in \oplus (s \gg 38) \oplus (s \gg 57)) \,\&\, 1$ $s \leftarrow (s \ll 1) \mid out$ $r \leftarrow r \mid (out \ll i)$ state $\leftarrow s$ **end for**	$s \leftarrow$ state $d \leftarrow$ data $r \leftarrow (s \gg 6) \oplus (s \gg 25) \oplus d$ $r \leftarrow r \oplus (r \ll 39) \oplus (r \ll 58)$ state $\leftarrow r$
0.436 Gbps	21 Gbps

- CRC computation
- DMA engine

SoNIC Design: Interface and Control

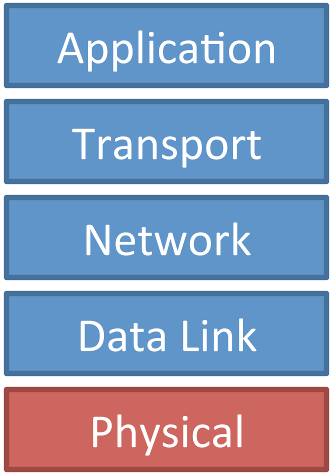
- Hardware control: *ioctl* syscall
- I/O : character device interface
- Sample C code for packet generation and capture

```
#include "sonic.h"

struct sonic_pkt_gen_info info = {
.mode = 0,
.pkt_num = 1000000000UL,
.pkt_len = 1518,
.mac_src = "00:11:22:33:44:55",
.mac_dst = "aa:bb:cc:dd:ee:ff",
.ip_src = "192.168.0.1",
.ip_dst = "192.168.0.2",
.port_src = 5000,
.port_dst = 5000,
.idle = 12,
};

/* OPEN DEVICE*/
fd1 = open(SONIC_CONTROL_PATH, O_RDWR);
fd2 = open(SONIC_PORT1_PATH, O_RDONLY);
/* CONFIG SONIC CARD FOR PACKET GEN*/
ioctl(fd1, SONIC_IOC_RESET)
ioctl(fd1, SONIC_IOC_SET_MODE, PKT_GEN_CAP)
ioctl(fd1, SONIC_IOC_PORT0_INFO_SET, &info)

/* START EXPERIMENT*/
ioctl(fd1, SONIC_IOC_START)
// wait till experiment finishes
ioctl(fd1, SONIC_IOC_STOP)

/* CAPTURE PACKET */
while ((ret = read(fd2, buf, 65536)) > 0) {
// process data
}

close(fd1);
close(fd2);
```

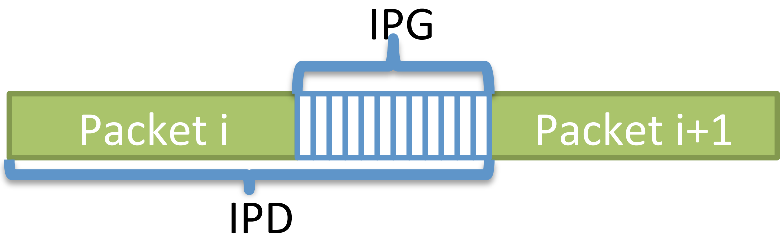
Outline

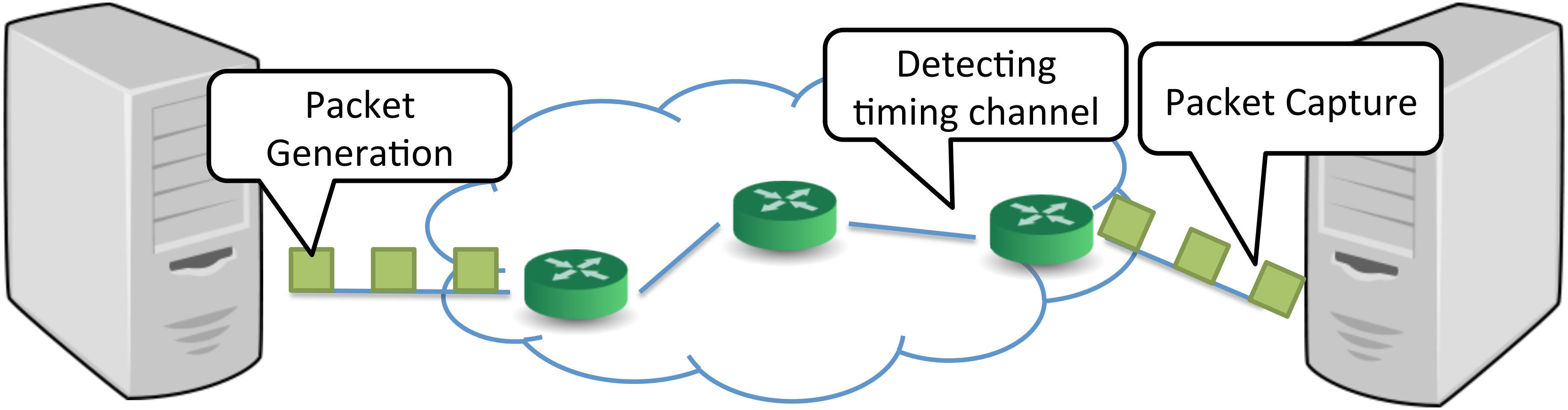
- Introduction
- SoNIC: Software-defined Network Interface Card
- Network Research Applications
  - Packet Generation
  - Packet Capture
  - Covert timing channel
- Conclusion

Network Research Applications



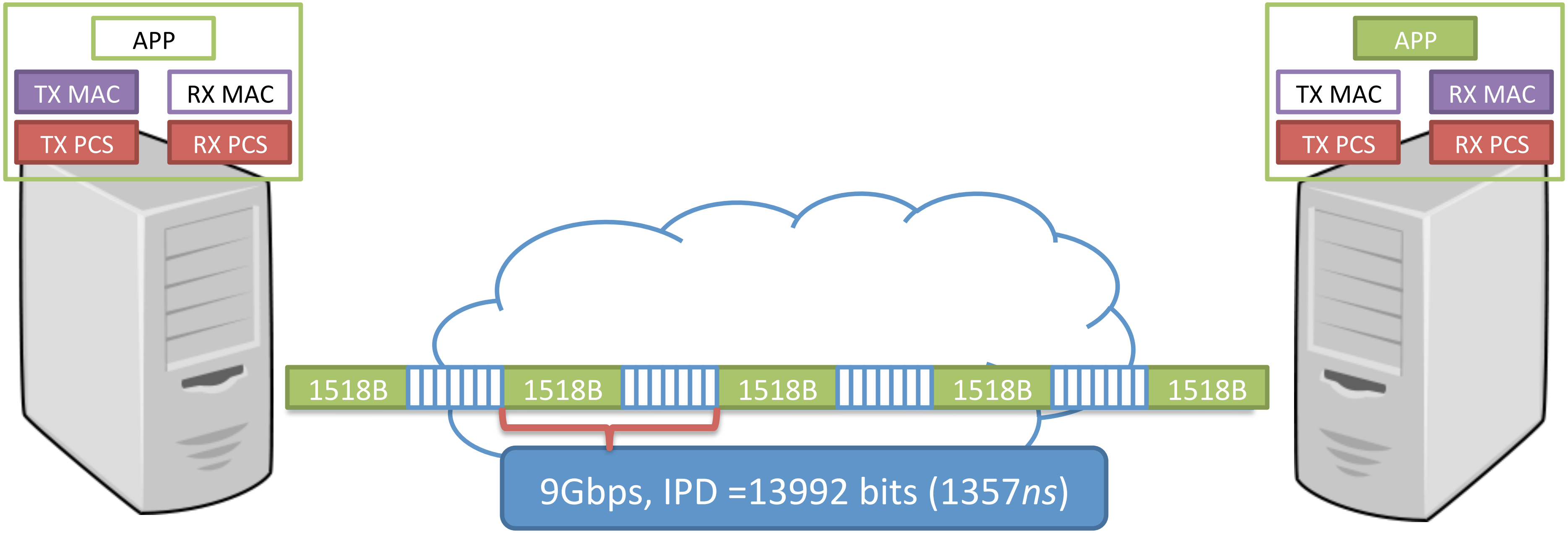
- Interpacket delays and gaps





Packet Generation and Capture

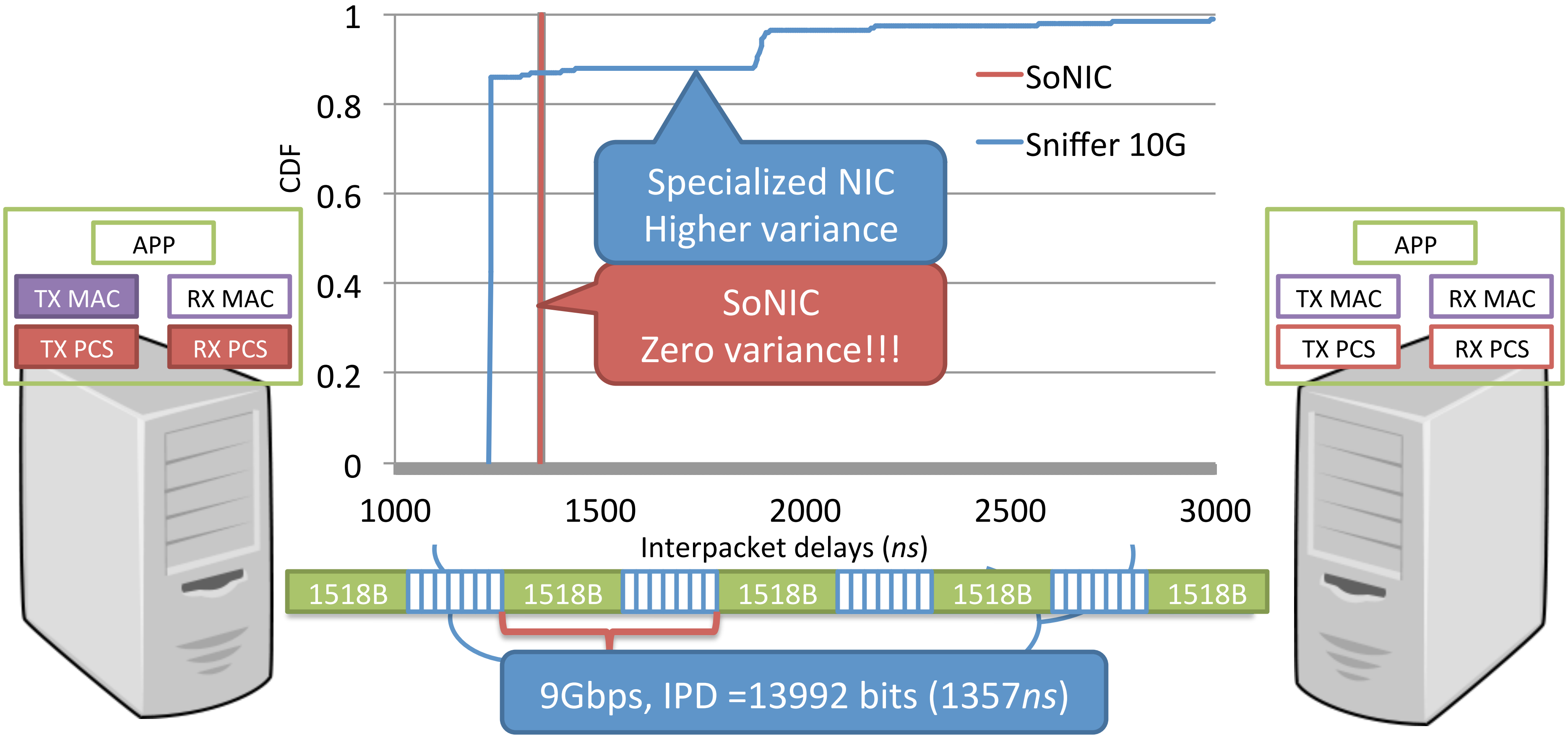
- Basic functions for network research
  - Generation: SoNIC allows control of IPGs in # of /I/s
  - Capture: SoNIC captures what was sent with IPGs in bits



9Gbps, IPD =13992 bits (1357*ns*)

Packet Generation

- *SoNIC allows precise control of IPGs*



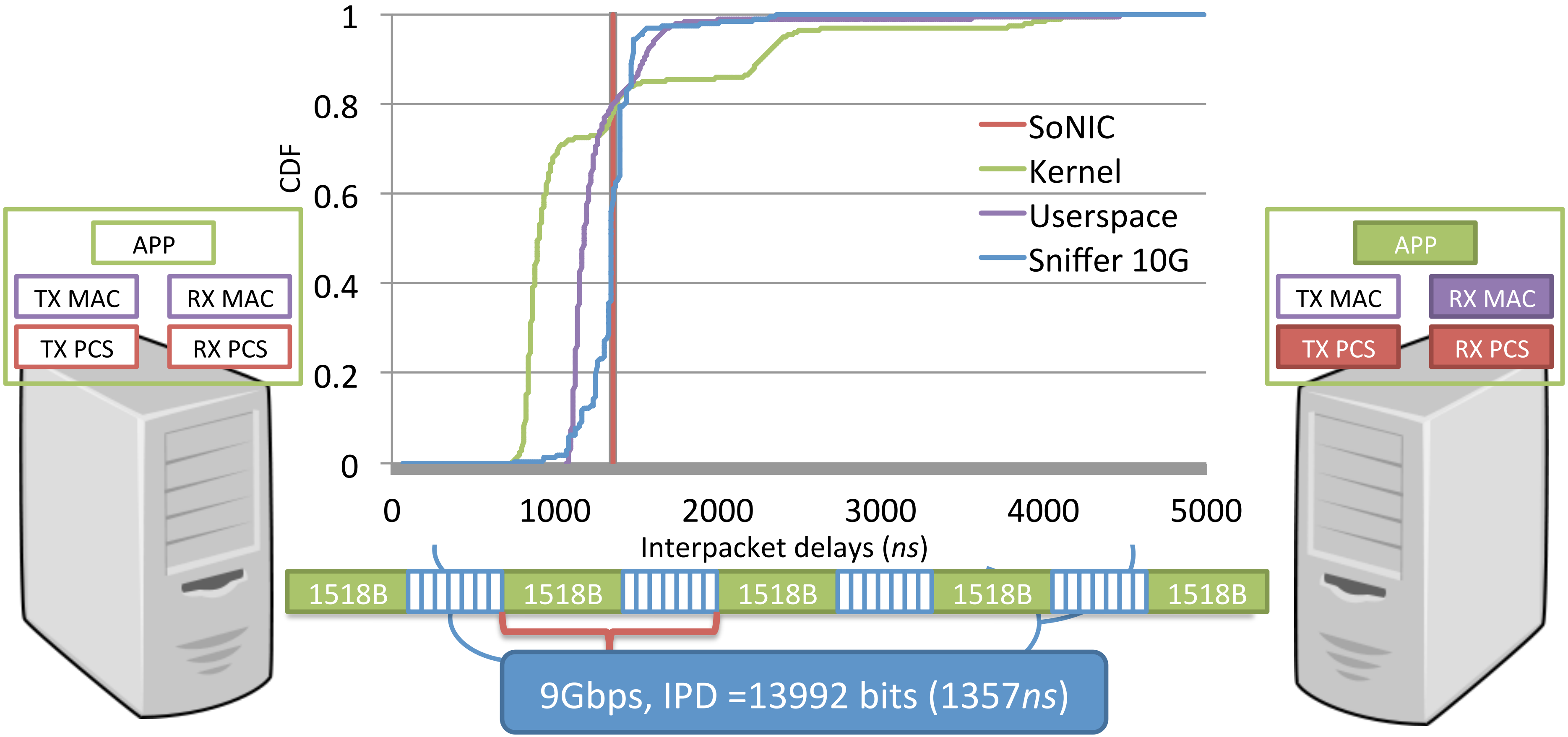
Specialized NIC
Higher variance

SoNIC
Zero variance!!!

9Gbps, IPD =13992 bits (1357*ns*)

Packet Capture

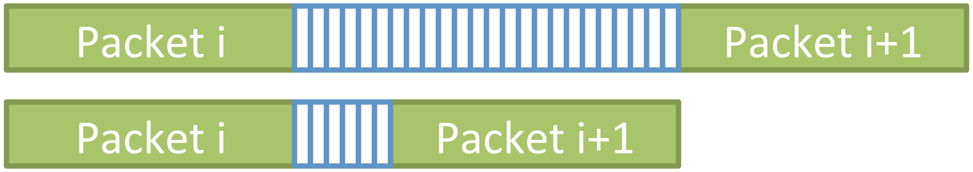
- *SoNIC captures what is sent*



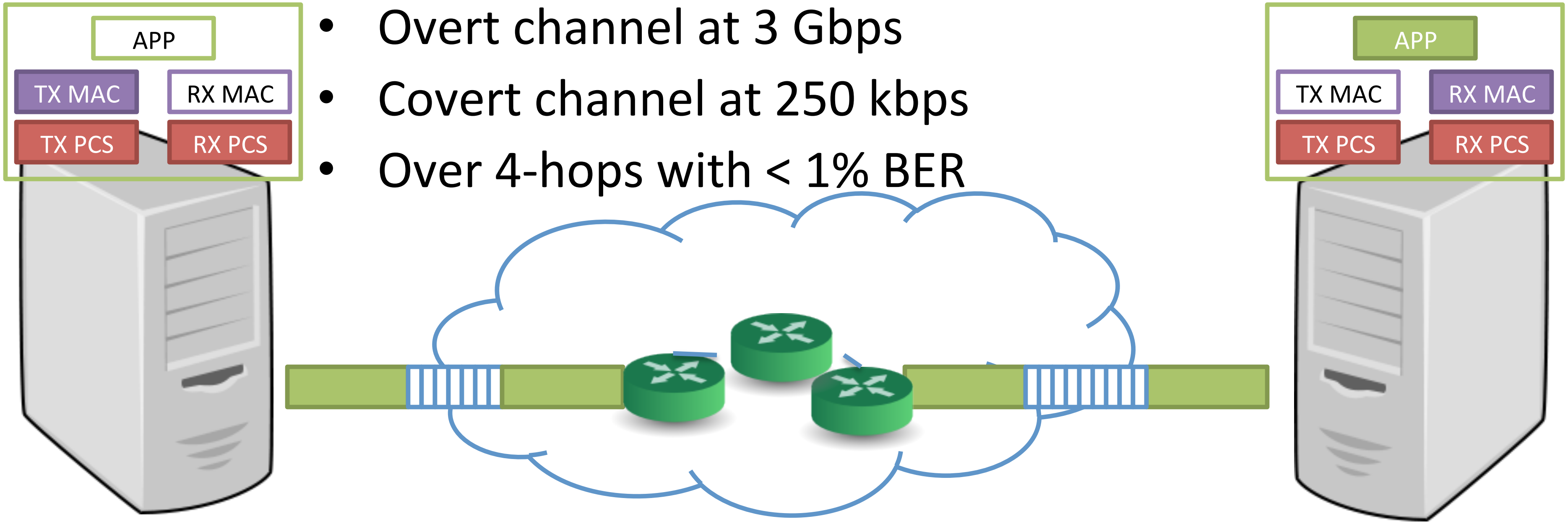
9Gbps, IPD =13992 bits (1357*ns*)

Covert Timing Channel

- Embedding signals into interpacket gaps.
  - Large gap: ‘1’
  - Small gap: ‘0’

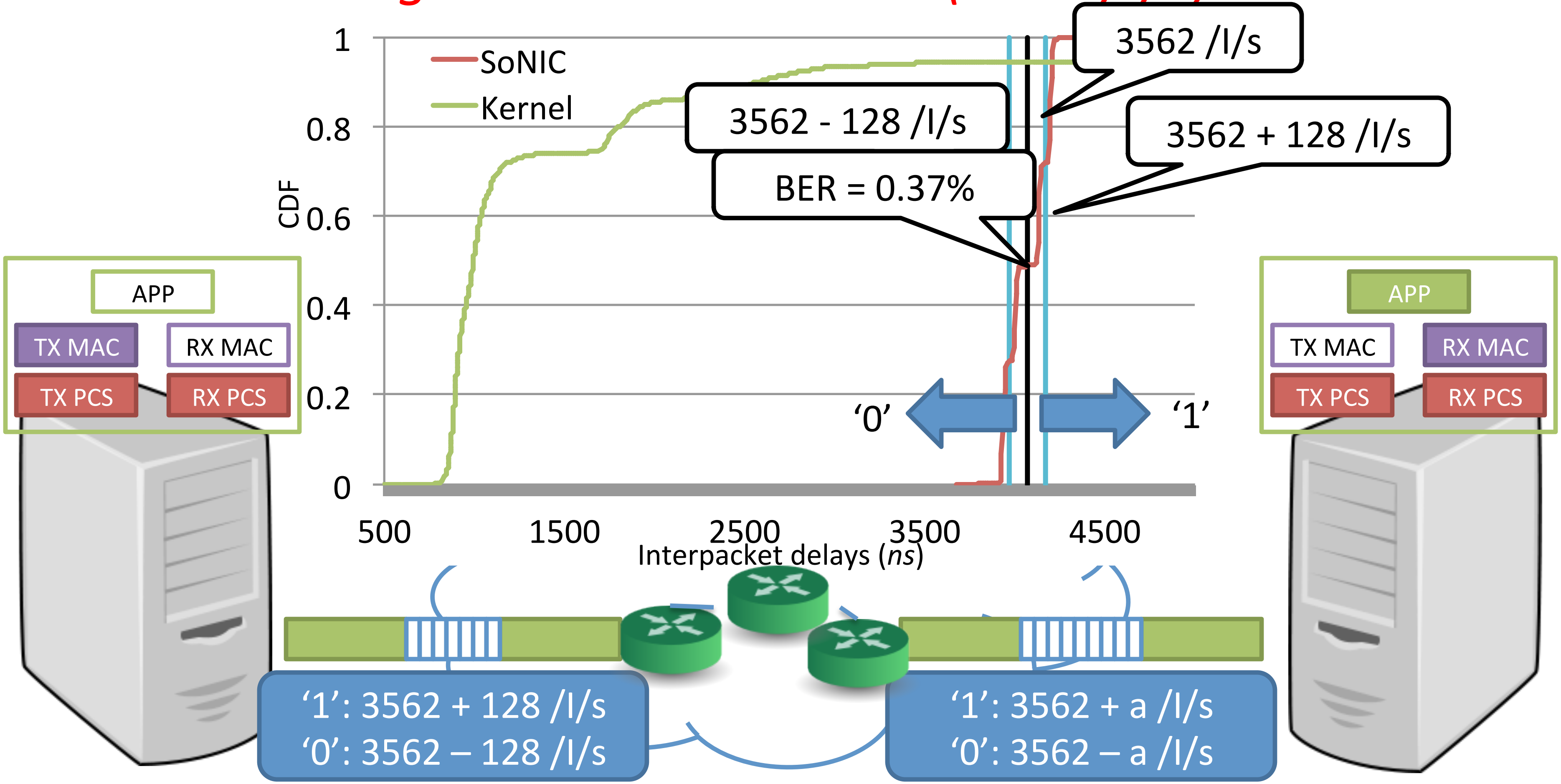


- Covert timing channel by modulating IPGs at 100ns
  - Overt channel at 3 Gbps
  - Covert channel at 250 kbps
  - Over 4-hops with < 1% BER



Covert Timing Channel

- *Modulating IPGS at 100ns scale (=128 /I/s)*



'1': 3562 + 128 /I/s
'0': 3562 – 128 /I/s

'1': 3562 + a /I/s
'0': 3562 – a /I/s

Contributions

- Network Research
  - Unprecedented access to the PHY with commodity hardware
  - A platform for cross-network-layer research
  - Can improve network research applications
- Engineering
  - Precise control of interpacket gaps (delays)
  - Design and implementation of the PHY in software
  - Novel scalable hardware design
  - Optimizations / Parallelism
- Status
  - Measurements in large scale: DCN, GENI, 40 GbE

Conclusion

- Precise Realtime Software Access to the PHY
- Commodity components
  - An FPGA development board, Intel architecture
- Network applications
  - Network measurements
  - Network characterization
  - Network steganography
- Webpage: http://sonic.cs.cornell.edu
  - SoNIC is available Open Source.

Thank you

Demo tonight!

http://sonic.cs.cornell.edu