Real-time 802.11 on WARP

Patrick Murphy
Mango Communications

Nov 2013

http://warpproject.org

http://mangocomm.com

A Bit of History

- Rice WARP funded by NSF in 2006
  - Led by Prof. Ashu Sabharwal
  - Rice team designed WARP v1 and v2 hardware
  - Community support, reference designs, 11 workshops
  - Distributed hardware to 25+ research groups
- Mango Communications founded in 2008
  - Took over hardware manufacturing & distribution
  - Took over all WARP development and support in early 2012
  - Released all-new WARP v3 hardware in mid-2012
    - Already most widely-adopted version of WARP hardware

WARP Reference Designs

- **WARPLab** Reference Design
- **OFDM** Reference Design

WARP Reference Designs

WARPLab

- Rapid PHY prototyping with MATLAB and WARP hardware
- Raw Tx/Rx waveforms via Ethernet
- Multi-antenna and multi-node from one script
- WARPLab 7
  - Re-designed from scratch in early 2013
  - Much cleaner code for multi-antenna / multi-node experiments
  - Much faster than previous versions
    - Custom mex function for network I/O
    - 2.1 msec to read 819 µsec of 40MHz “air” time

WARP Reference Designs

OFDM Ref Design

- MIMO OFDM PHY in FPGA
  - SISO, 2x2 multiplexing, 2x1 STBC, selection diversity
  - AF and DF cooperation
  - Custom frame format with 10 MHz bandwidth
- CSMA MAC in C
  - One software app for PHY control and full MAC
  - No higher layer MAC roles (AP vs STA, etc)
- Interoperates across all generations of WARP hardware

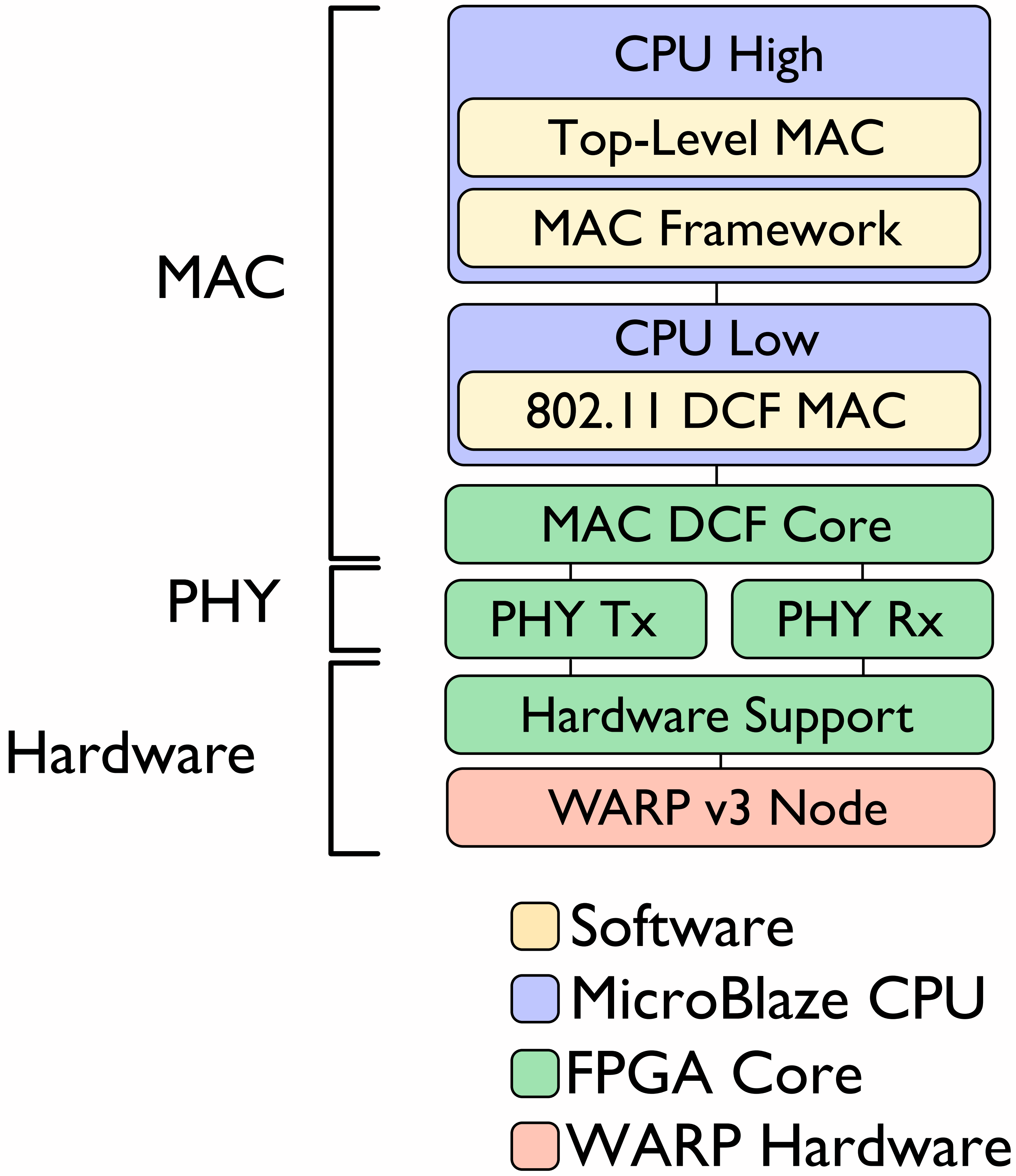
WARP Reference Designs

- **WARPLab** Reference Design
- **OFDM** Reference Design
- **802.11** Reference Design

802.11 Reference Design

- **Interoperability**
  - Real-time MAC & PHY in FPGA
  - No compromises on MAC timing or PHY features
  - AP & station implementations
- **Experimental Visibility**
  - Hooks throughout MAC & PHY
  - Framework for running experiments and understanding results
- **Extensibility**
  - All source code open
  - Behavior specified in software whenever possible
  - Interfaces for real world traffic

802.11 Reference Design



MAC

PHY

Hardware

CPU High

Top-Level MAC

MAC Framework

CPU Low

802.11 DCF MAC

MAC DCF Core

PHY Tx

PHY Rx

Hardware Support

WARP v3 Node

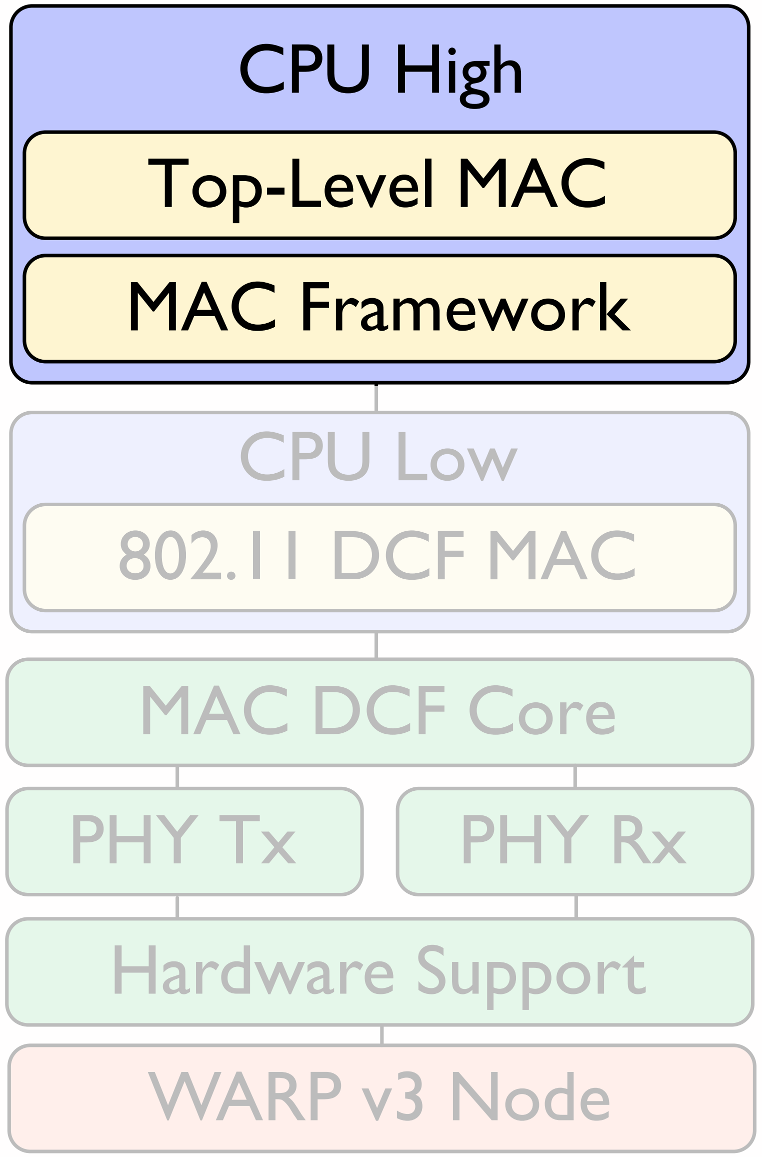
🟨 Software

🟦 MicroBlaze CPU

🟩 FPGA Core

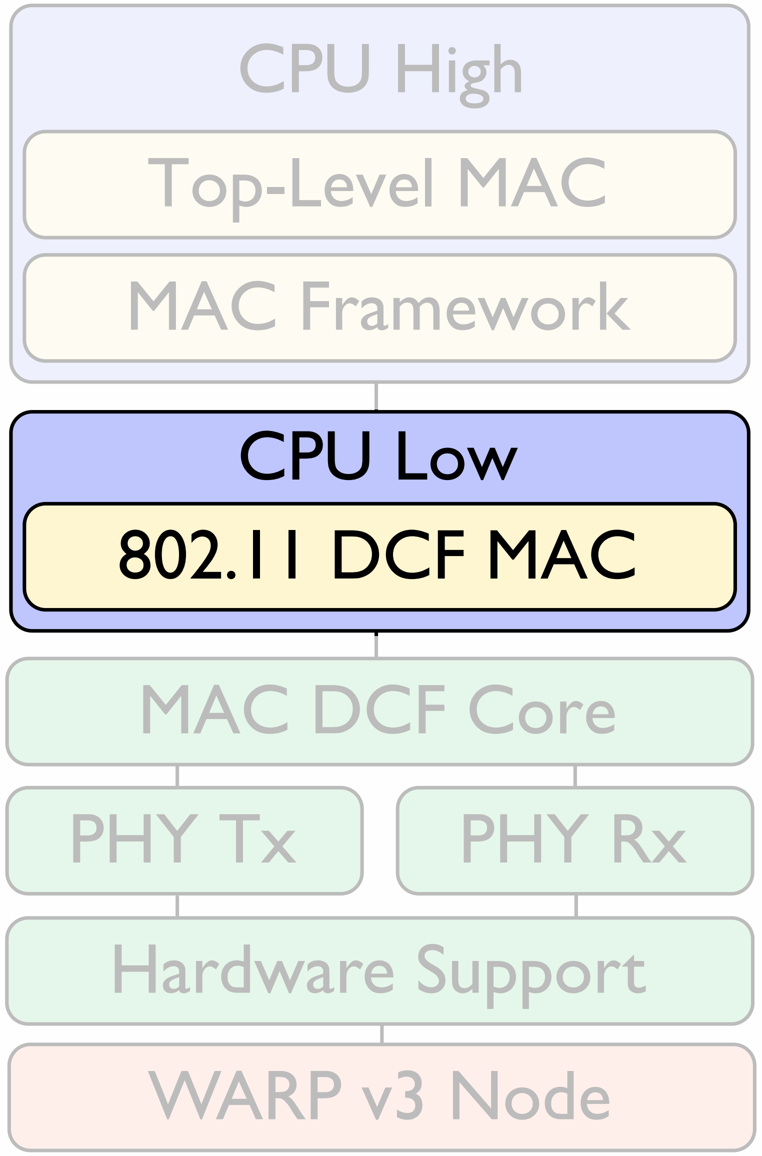
🟥 WARP Hardware

802.11 Reference Design



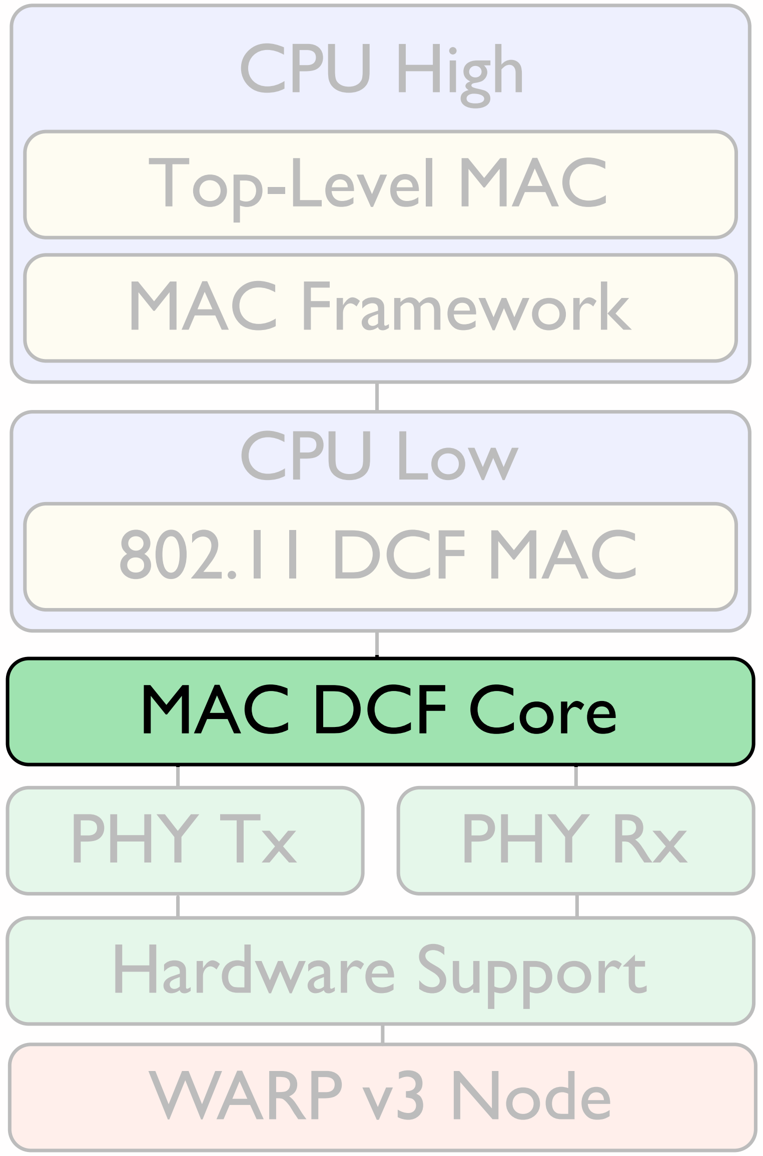
- CPU High
  - All inter-MPDU processing
  - Wired-wireless portal
  - Queueing
  - Role-specific behaviors
    - Beacons
    - Associations
  - Channel selection
  - Framework for common code

802.11 Reference Design



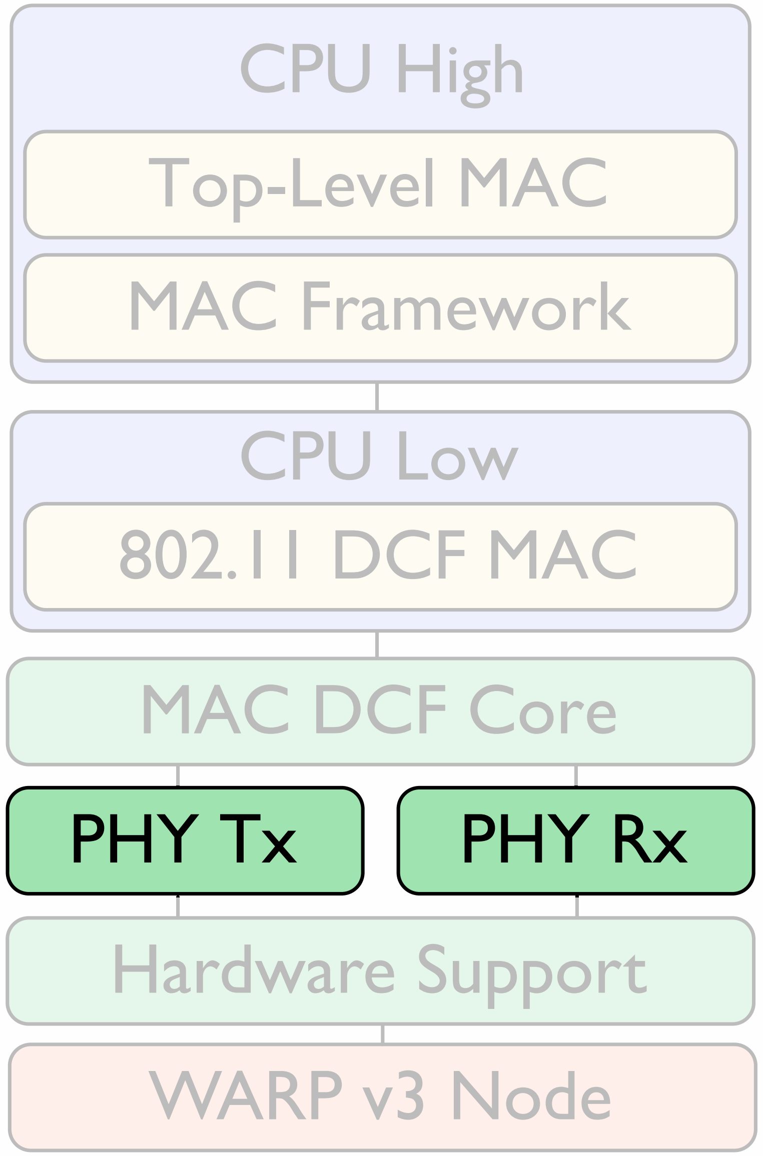
- CPU Low
  - All intra-MPDU processing
  - Rate selection
  - Re-transmissions
  - Backoff selections
  - PHY configuration
  - Same code for any top-level MAC

802.11 Reference Design



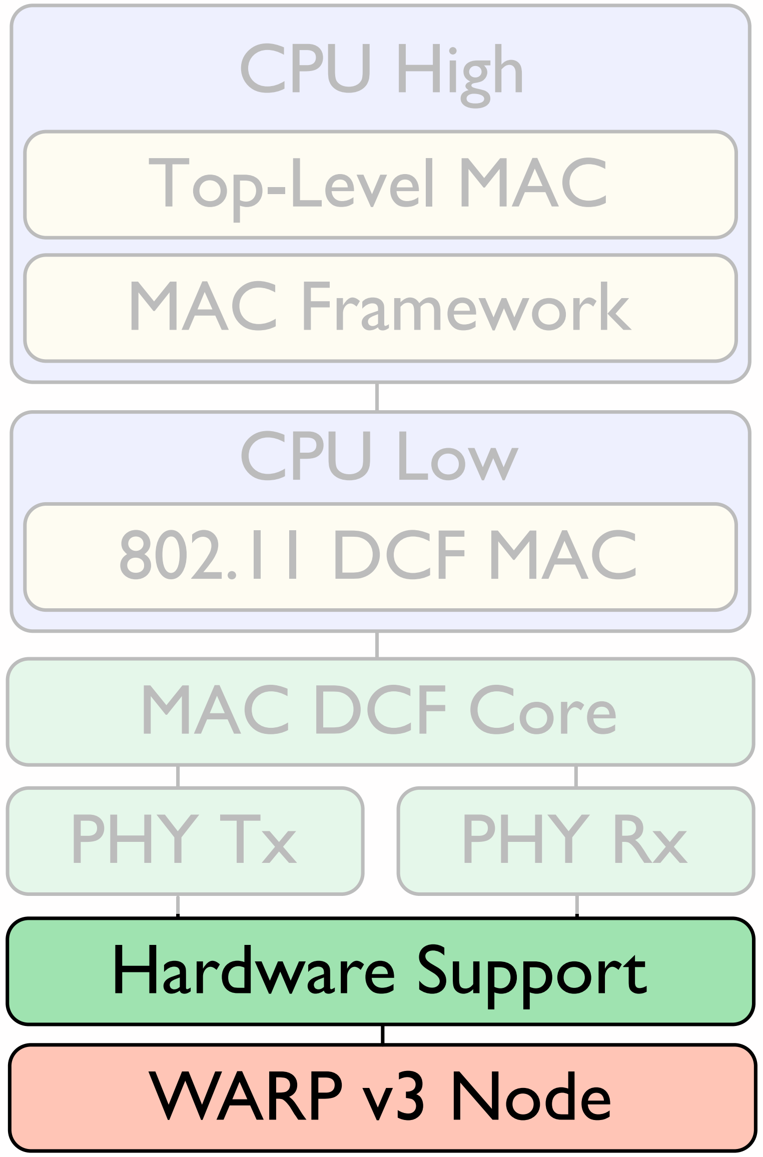
- DCF Core
  - DCF MAC state
    - Carrier sensing (CCA)
    - Slot timer
    - Backoff counter
    - NAV
  - Real-time PHY control
  - All parameters set by CPU Low

802.11 Reference Design



- PHY Cores
  - Designed in System Generator
  - OFDM Tx
  - OFDM & DSSS Rx
  - All SISO PHY rates
  - All synchronization real-time per pkt
  - 160MHz core clock
  - Flexible bandwidth (20MHz max)

802.11 Reference Design



- Hardware
  - Usual hardware support cores
    - radio_controller, ad_controller, Ethernet, etc.
  - Standard Mango WARP v3 node
  - FPGA resource utilization:

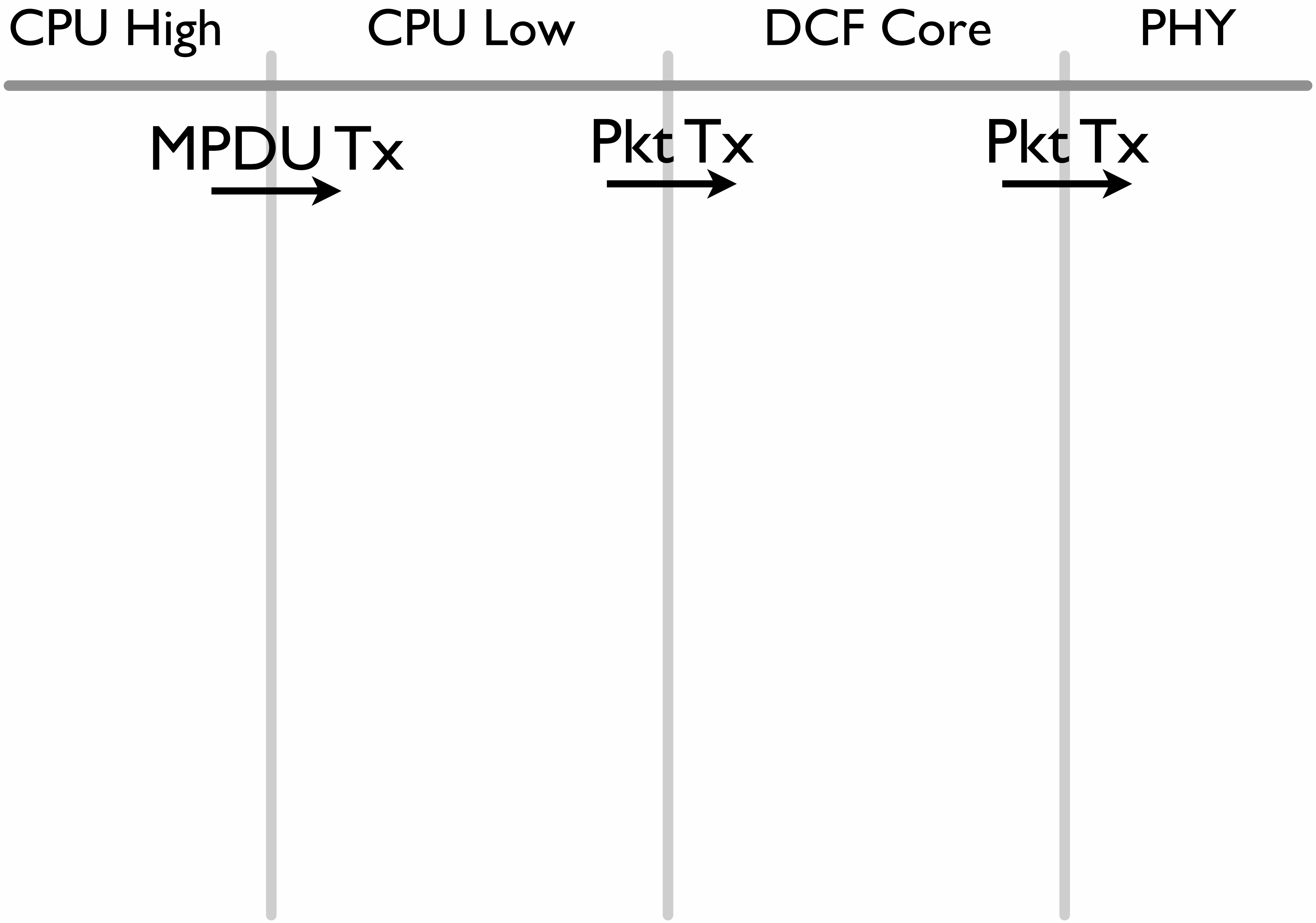
	LUT	FF	Mult	BRAM
802.11 Usage	62983	65073	135	245*
V6 Chip Total	150720	301440	768	416
% Used	**41%**	**21%**	**17%**	**55%**

**71 BRAM used by ChipScope ILA in Rx PHY*

Design v0.6-beta

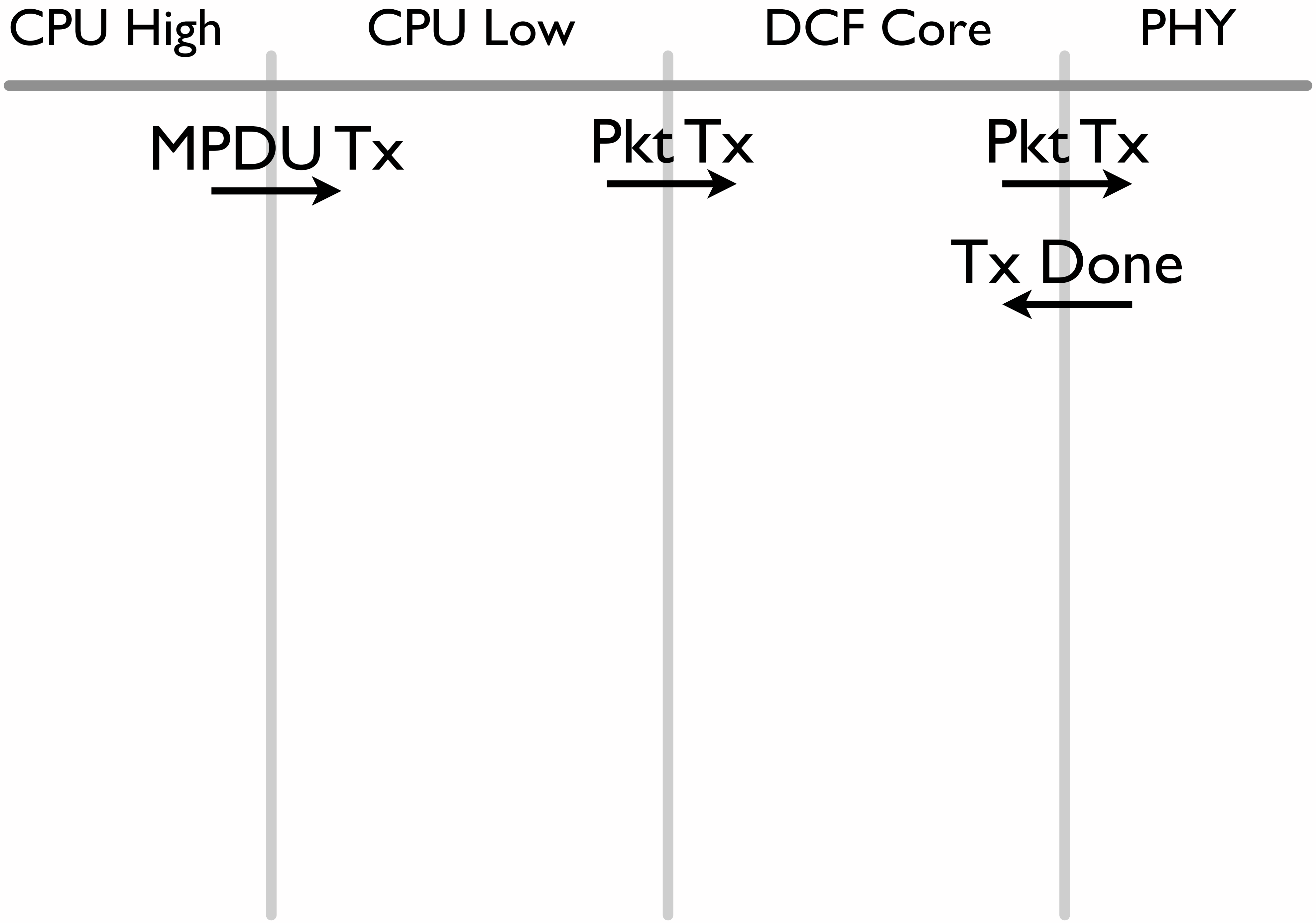
Packet Flow Example

Data Tx



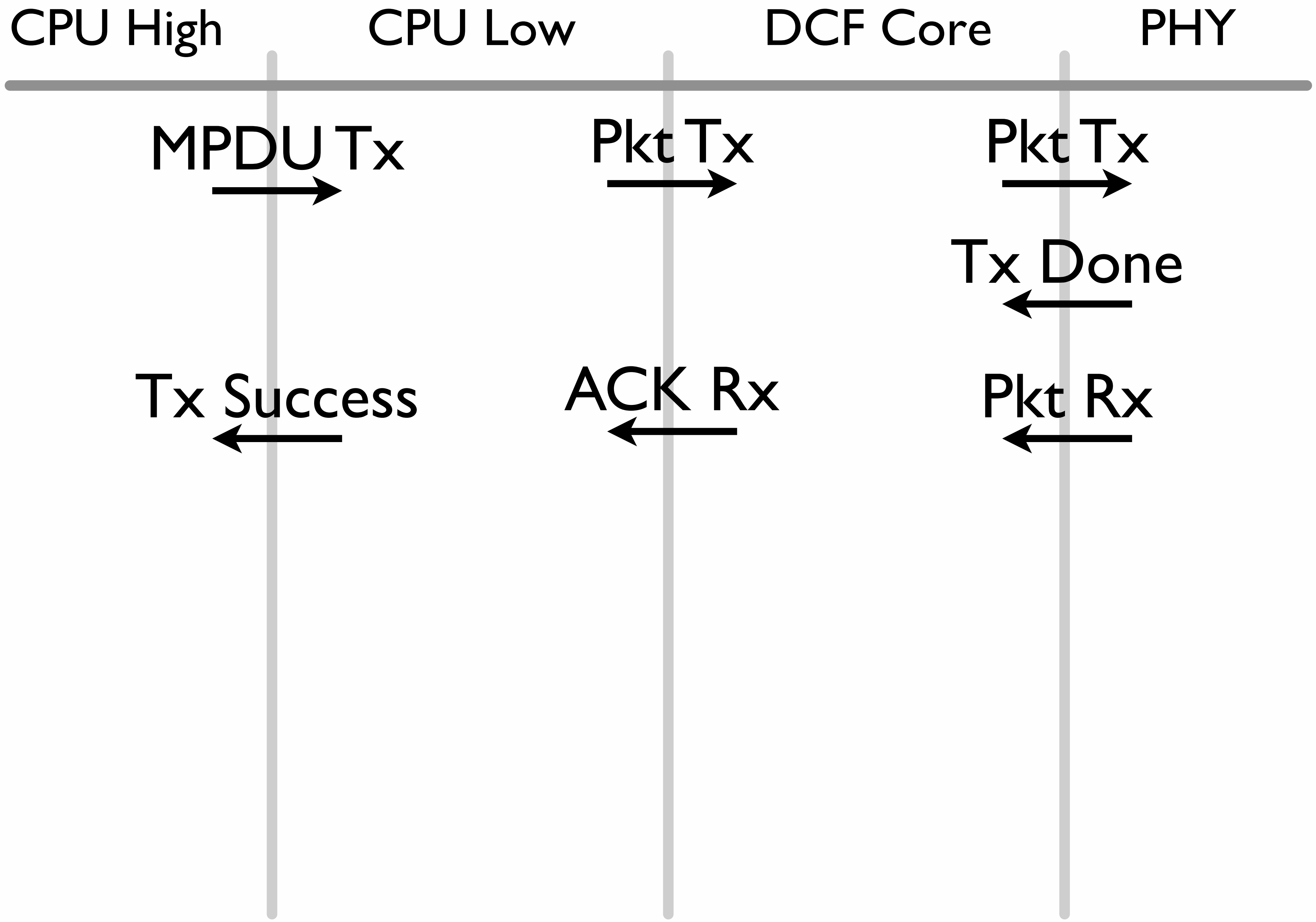
Packet Flow Example

Data Tx



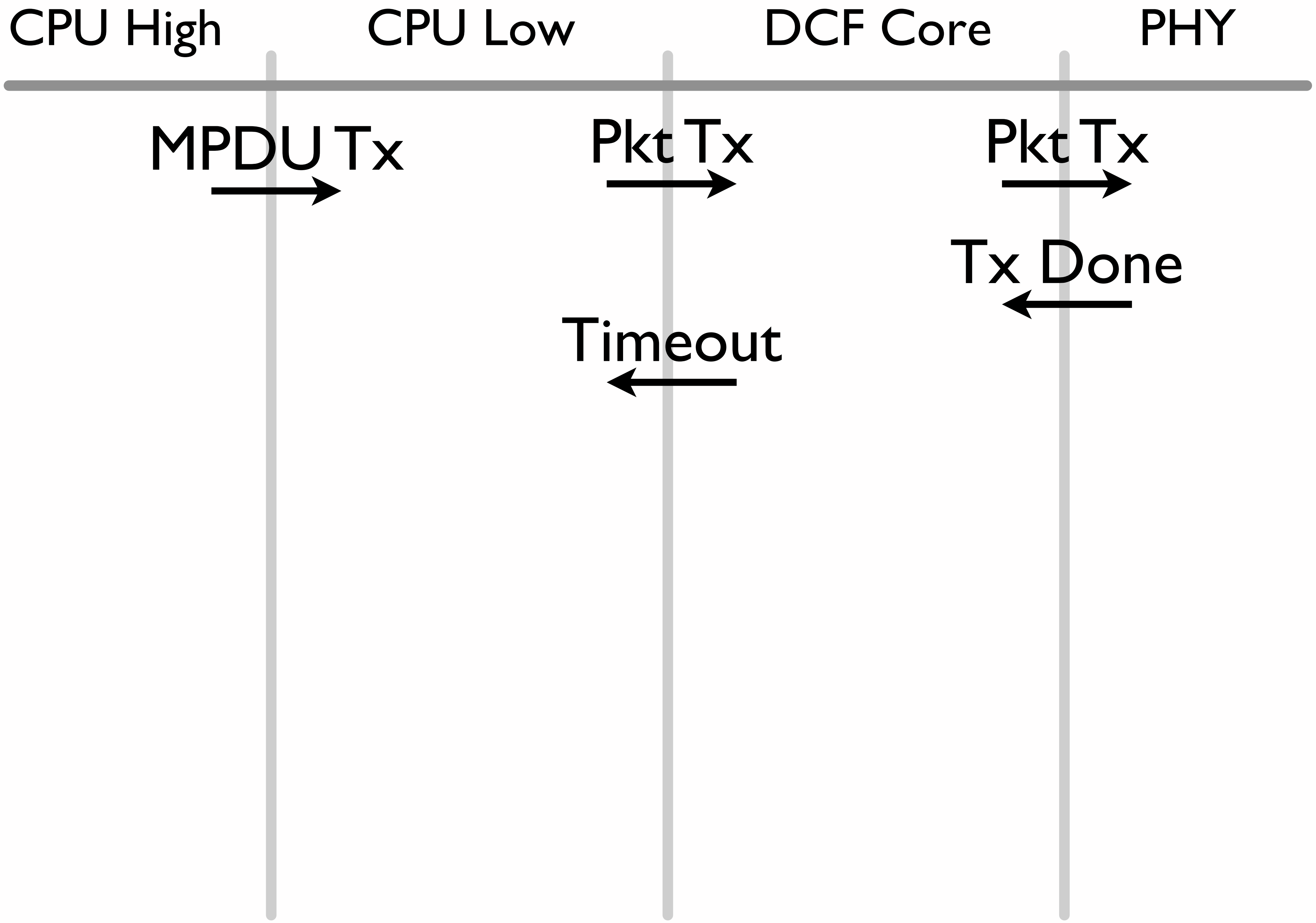
Packet Flow Example

Data Tx



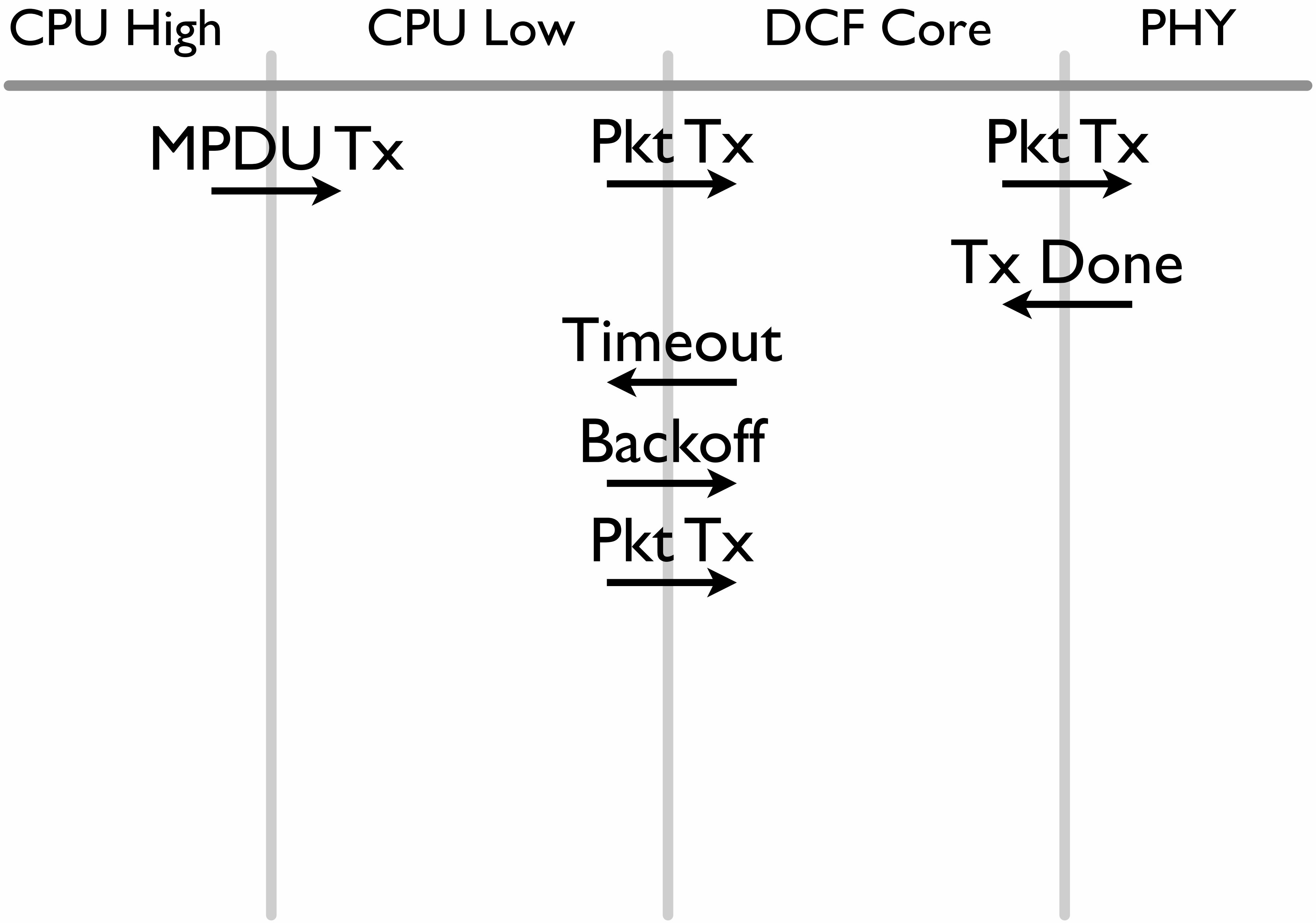
Packet Flow Example

Data Tx with Re-transmission



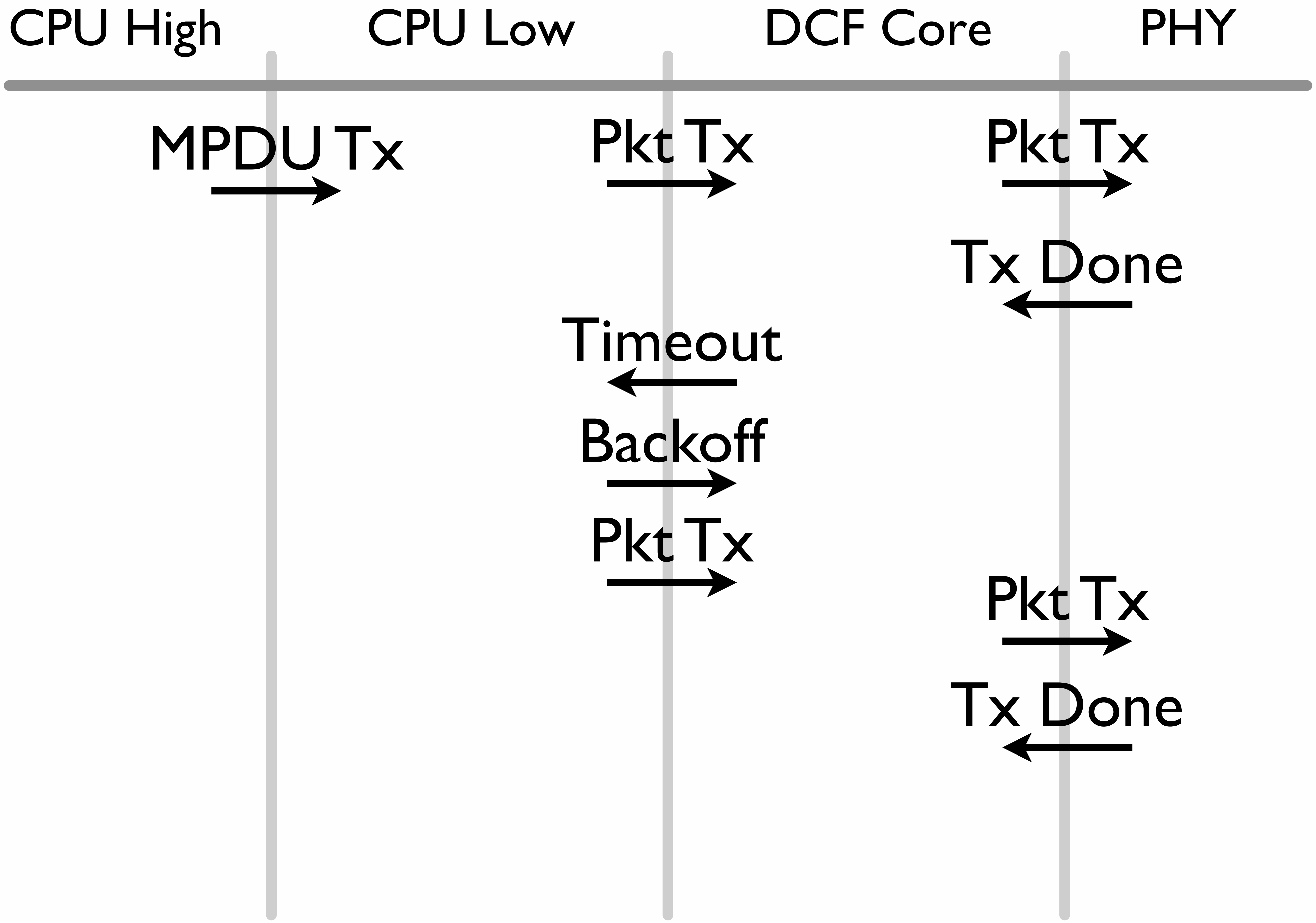
Packet Flow Example

Data Tx with Re-transmission



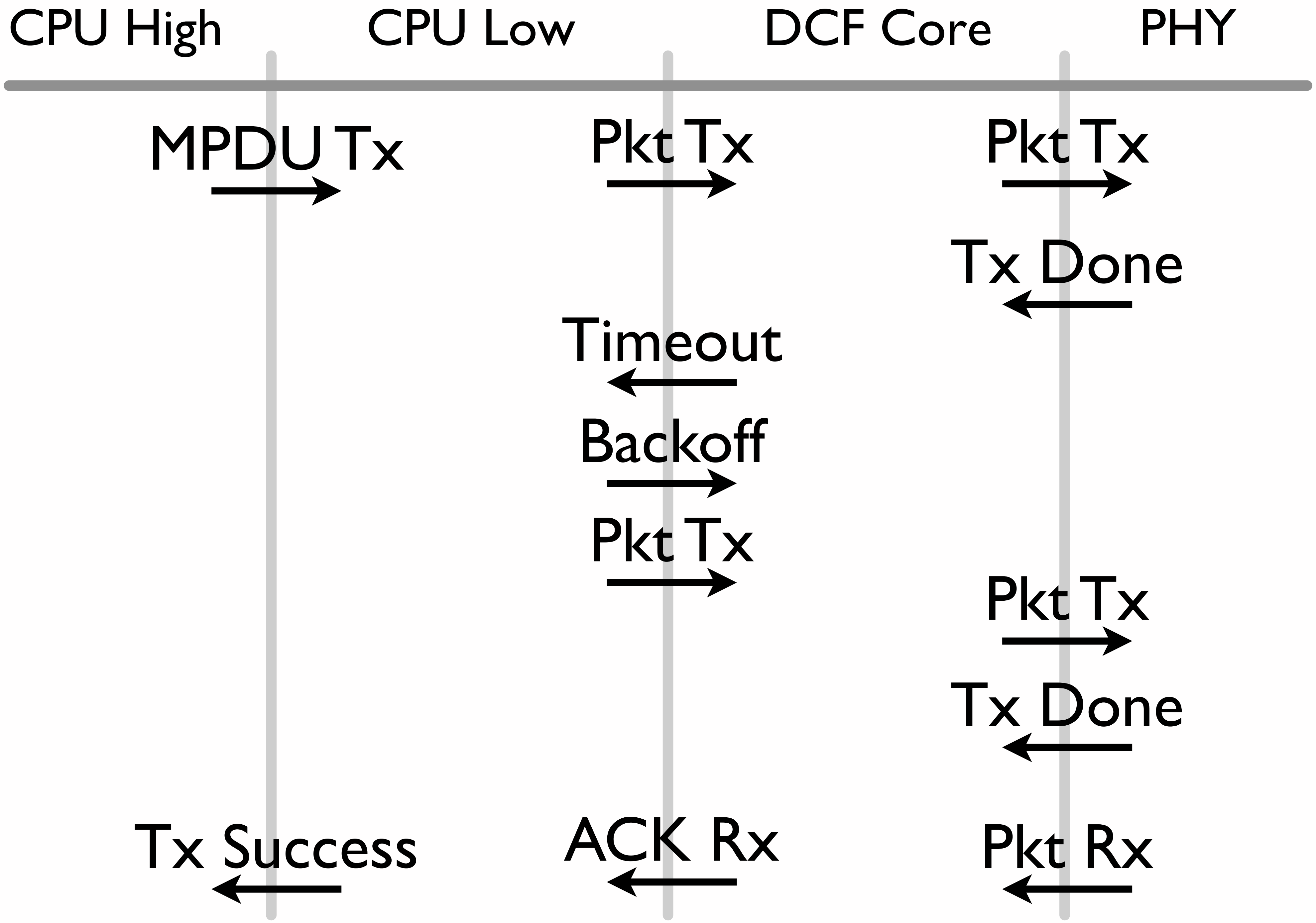
Packet Flow Example

Data Tx with Re-transmission



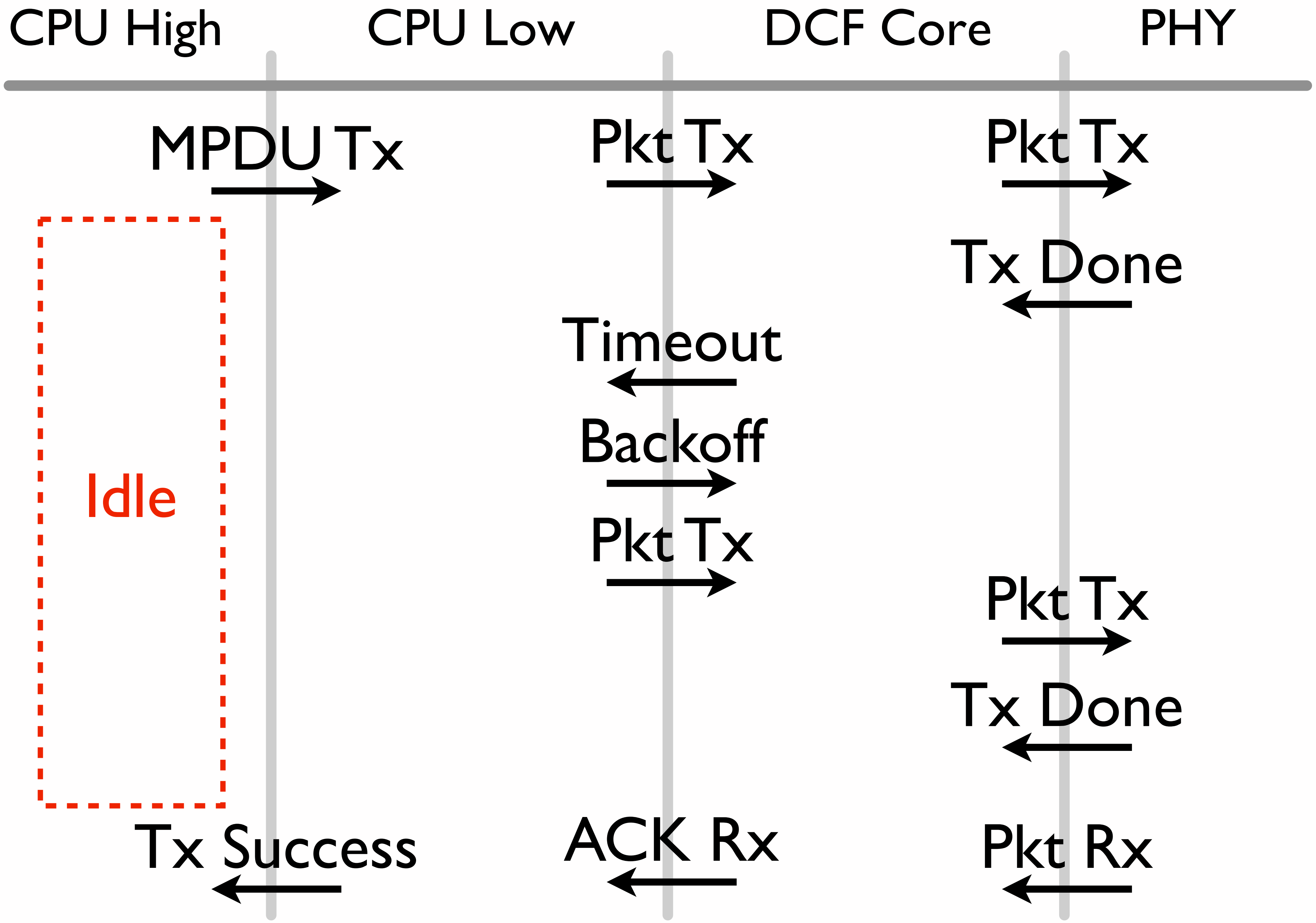
Packet Flow Example

Data Tx with Re-transmission



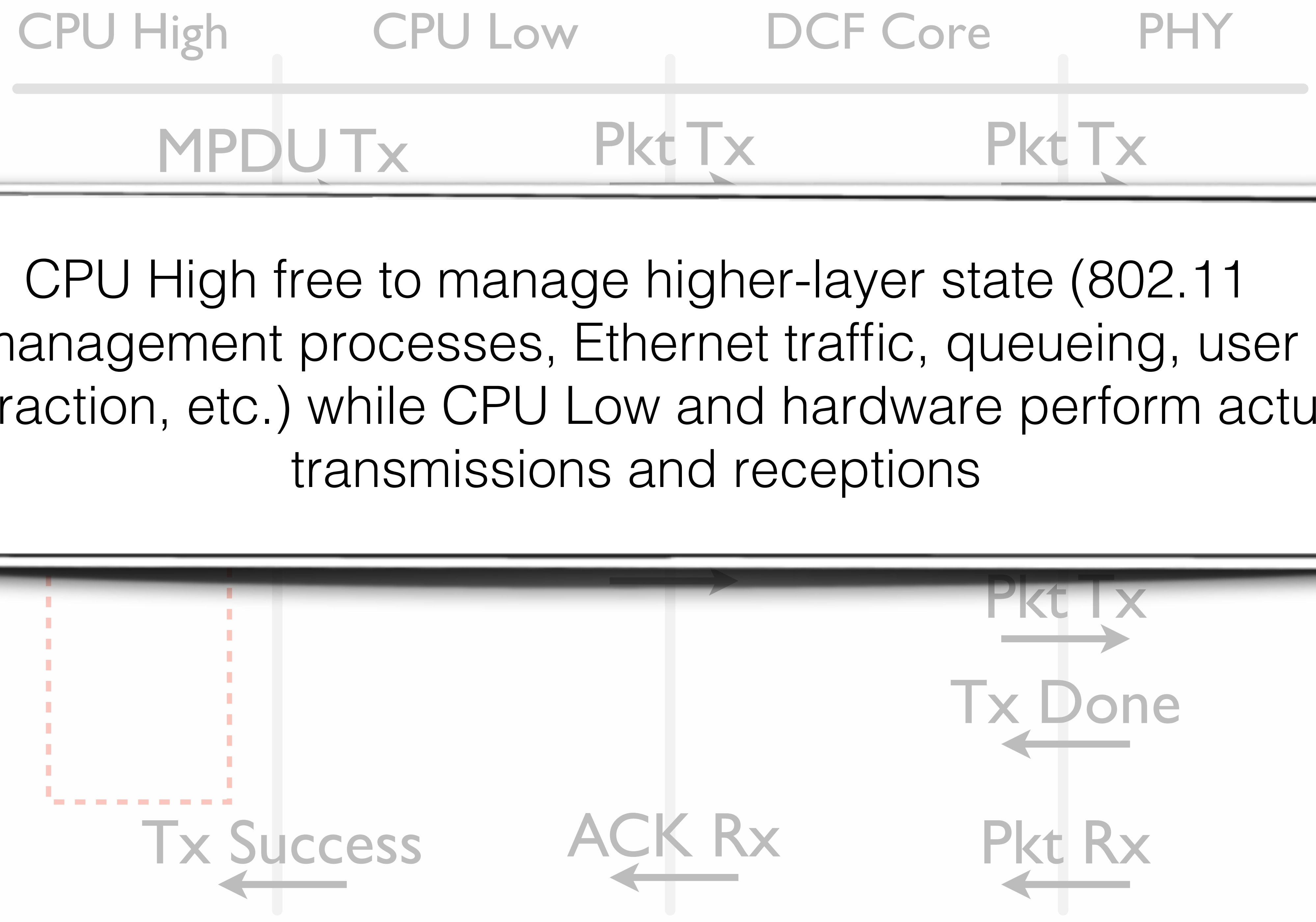
Packet Flow Example

Data Tx with Re-transmission



Packet Flow Example

Data Tx with Re-transmission



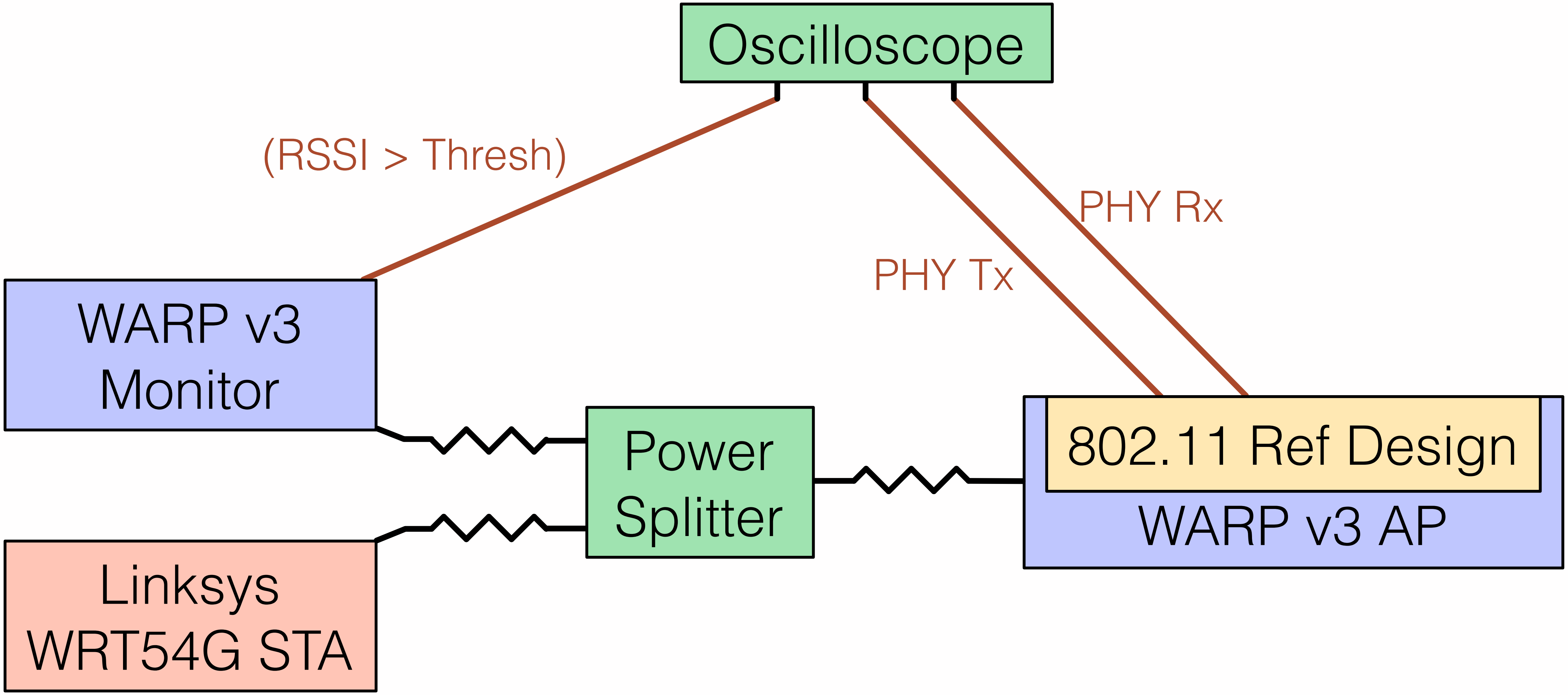
CPU High free to manage higher-layer state (802.11 management processes, Ethernet traffic, queueing, user interaction, etc.) while CPU Low and hardware perform actual transmissions and receptions

MAC Timing Calibration

- Standard gives precise IFS durations
- Tight tolerance required
    - Example: SIFS = (16 ± 0.9)µs on medium
- Must account for hardware & implementation latencies
    - Tricky calibration problem

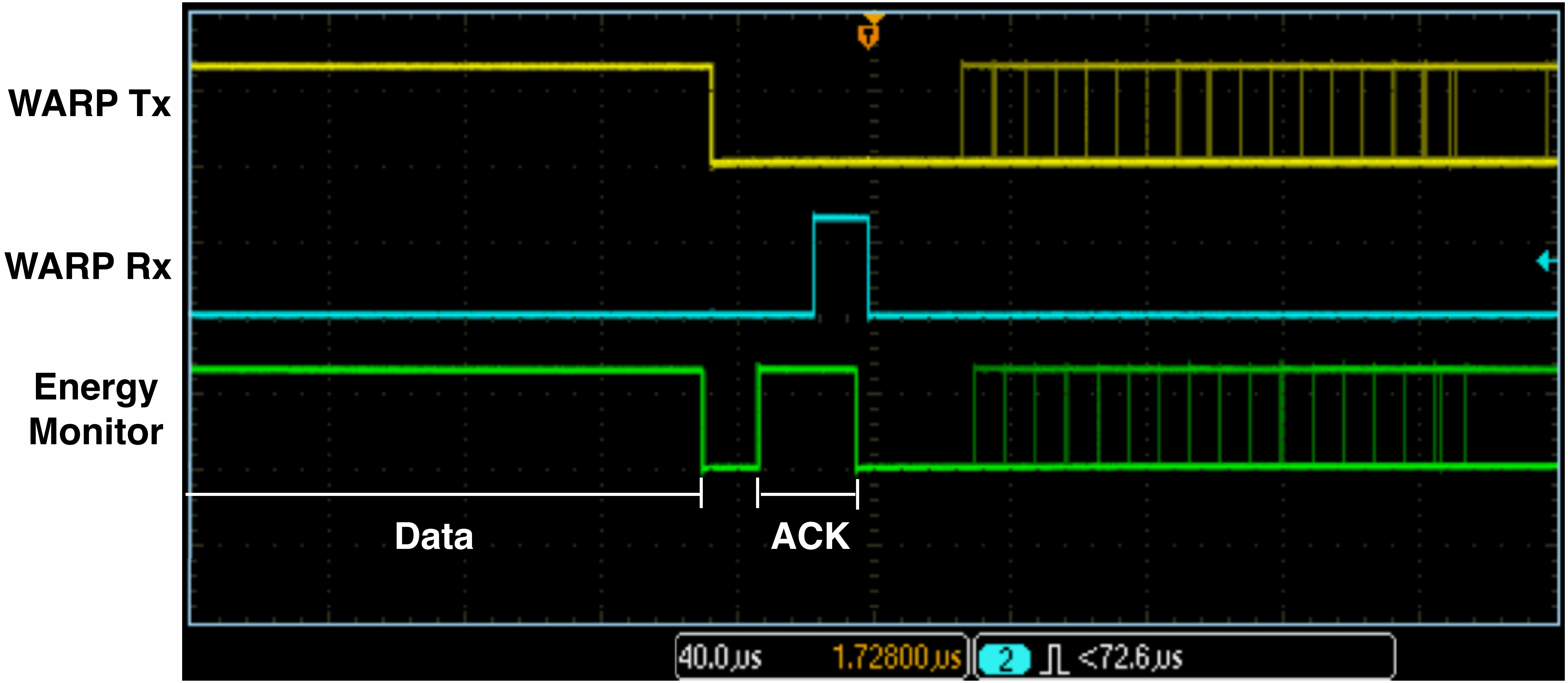
MAC Timing Calibration

Hardware Setup



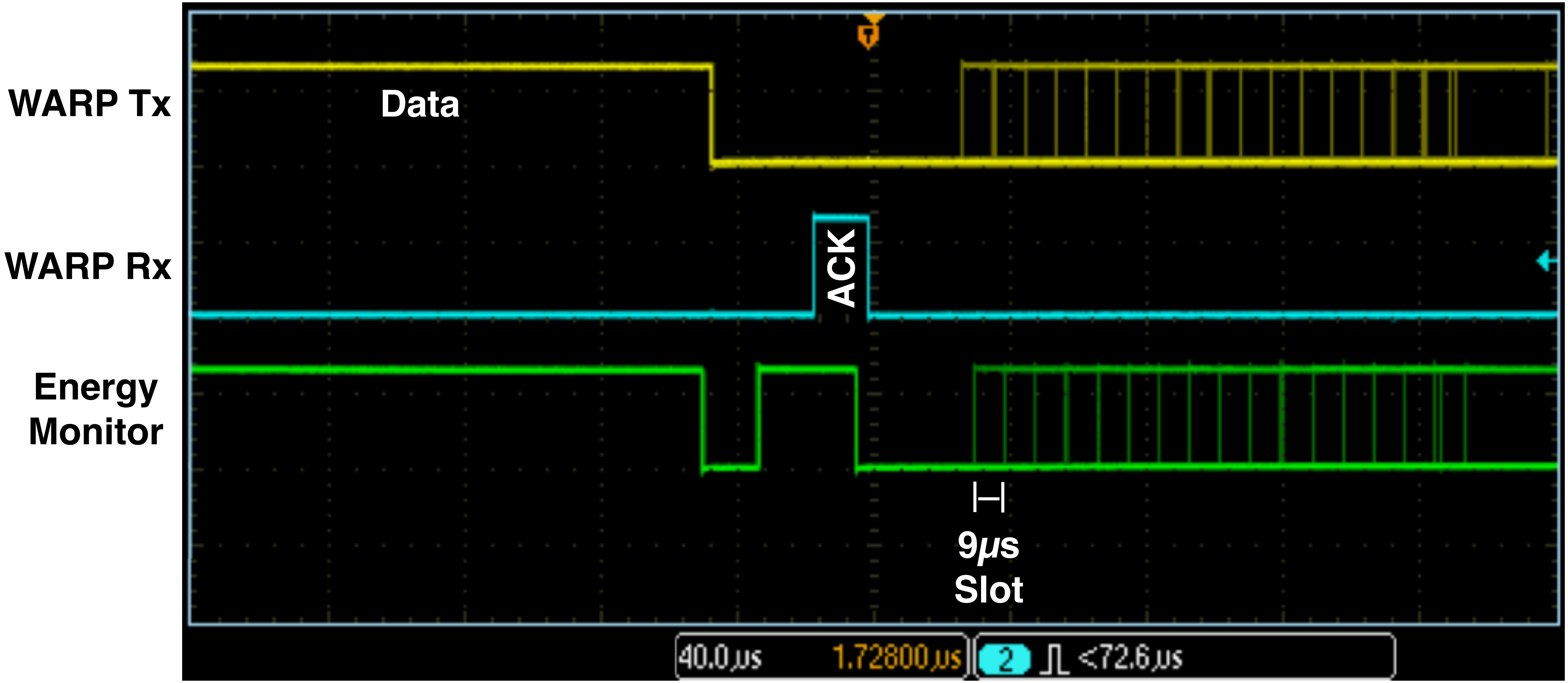
MAC Timing Calibration

WARP AP ⇄ Wi-Fi Station



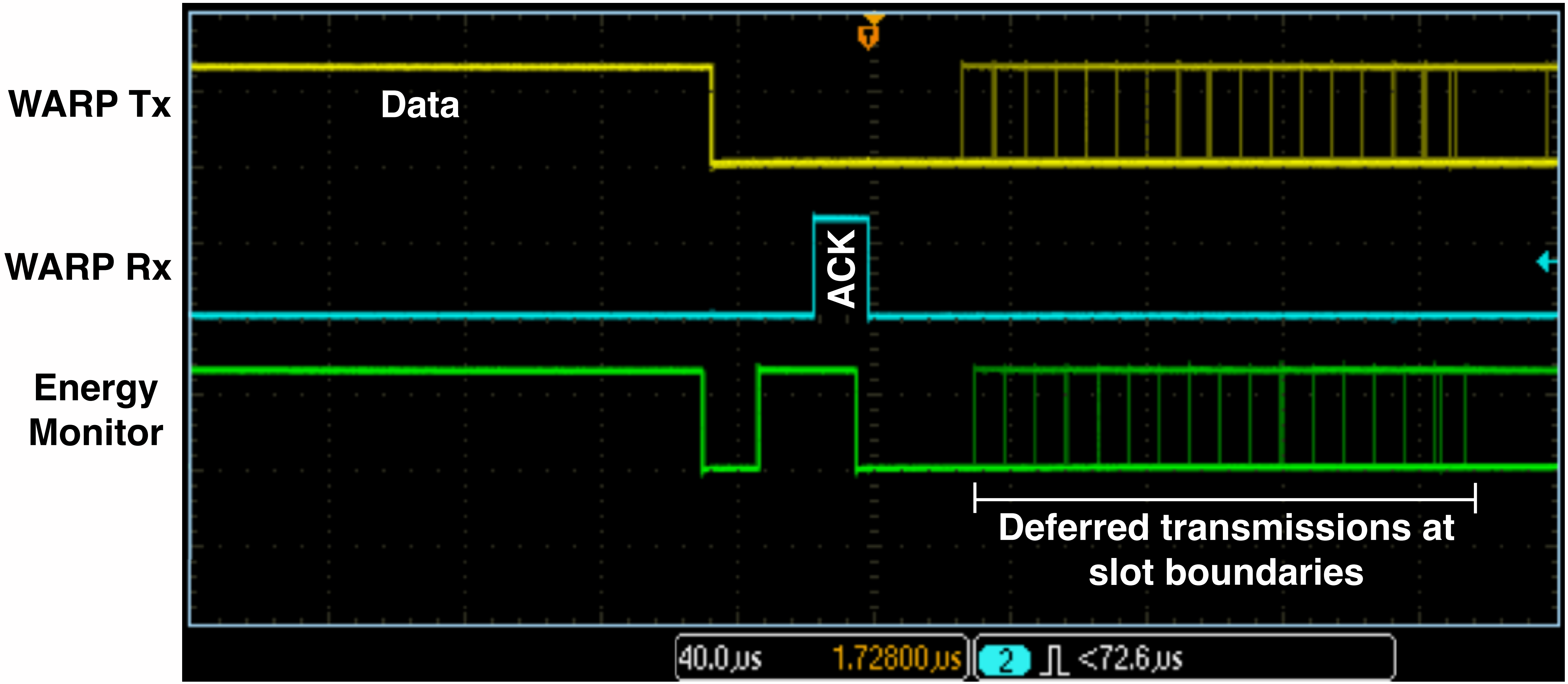
MAC Timing Calibration

WARP AP ⇄ Wi-Fi Station



MAC Timing Calibration

WARP AP ⇄ Wi-Fi Station



WARP Tx

Data

WARP Rx

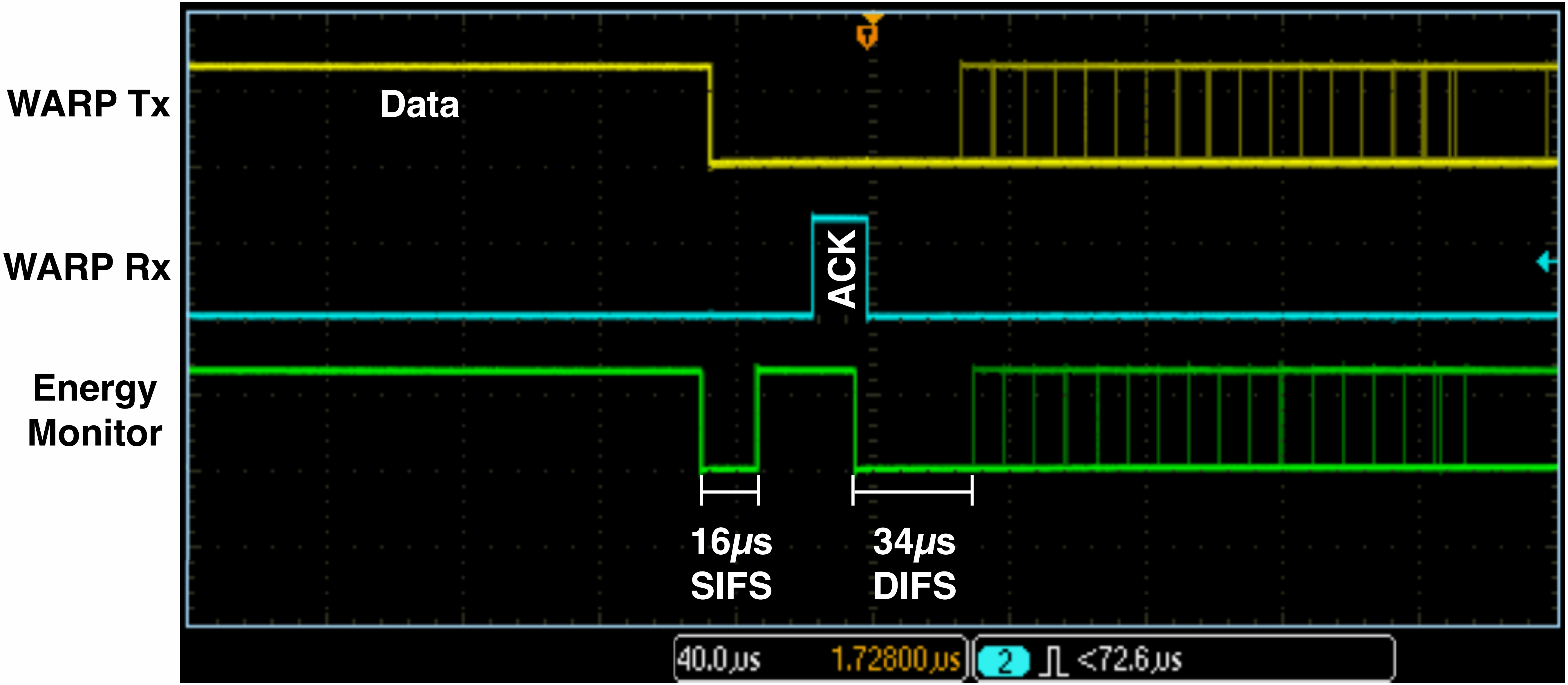
ACK

Energy Monitor

Deferred transmissions at slot boundaries

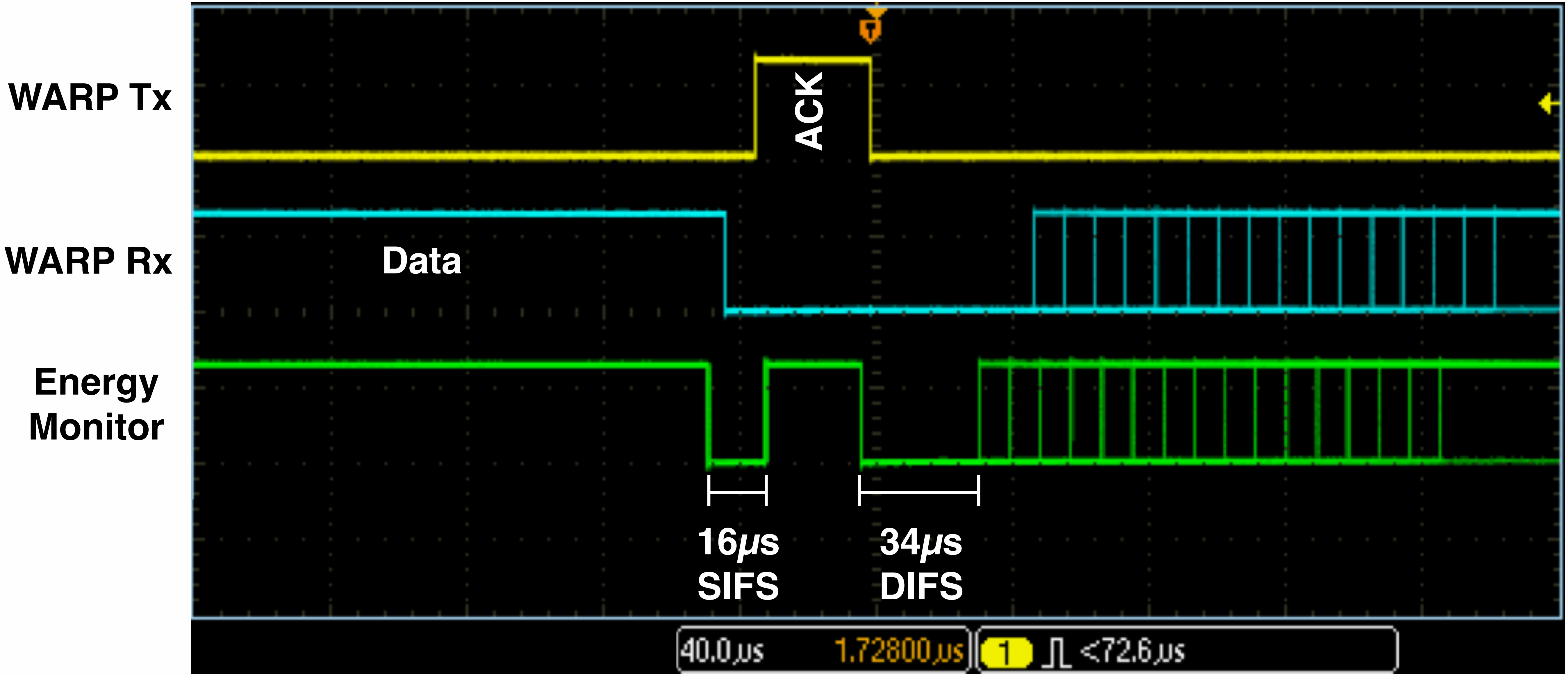
MAC Timing Calibration

WARP AP ⇄ Wi-Fi Station



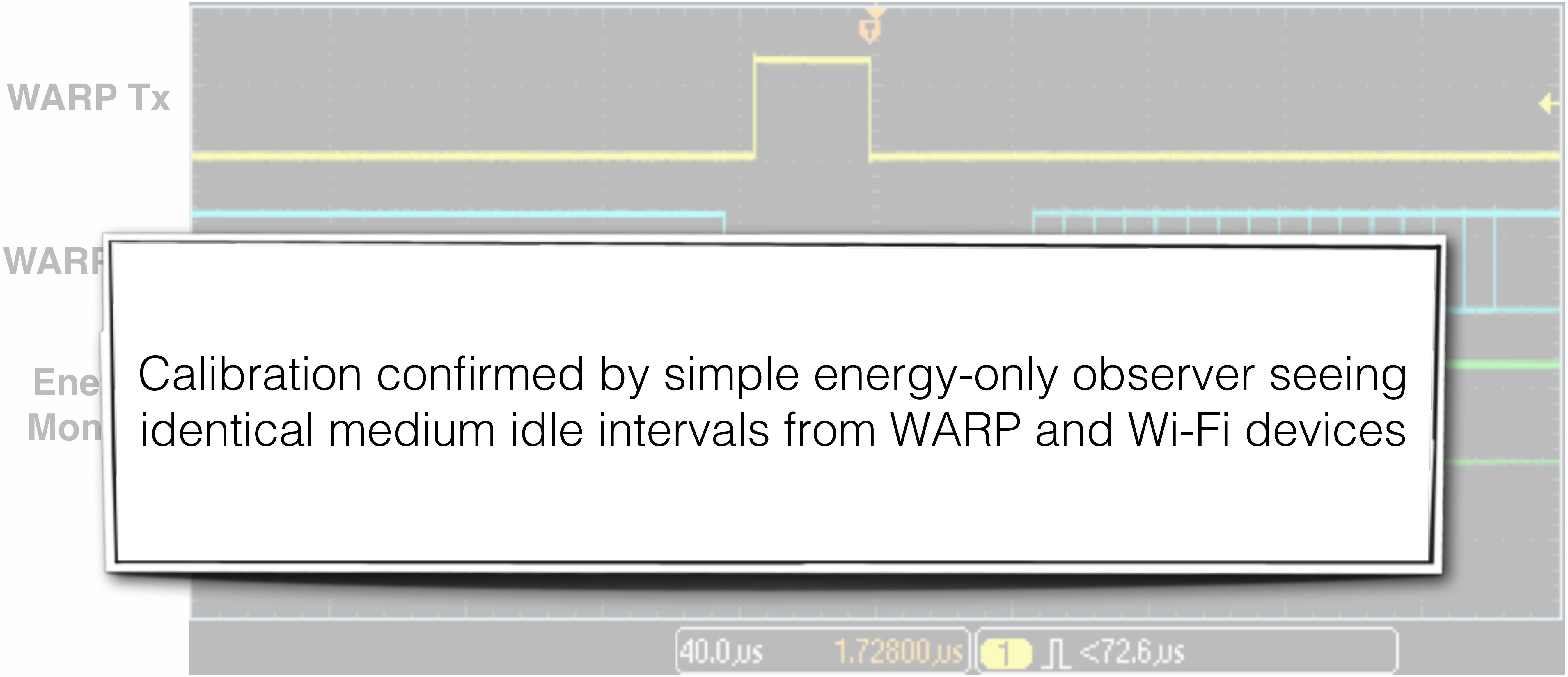
MAC Timing Calibration

WARP AP ⇄ Wi-Fi Station



MAC Timing Calibration

WARP AP ⇄ Wi-Fi Station



Calibration confirmed by simple energy-only observer seeing identical medium idle intervals from WARP and Wi-Fi devices

802.11 Reference Design

- **Interoperability**
  - Real-time MAC & PHY in FPGA
  - No compromises on MAC timing or PHY features
  - AP & station implementations
- **Experimental Visibility**
  - Hooks throughout MAC & PHY
  - Framework for running experiments and understanding results
- **Extensibility**
  - All source code open
  - Behavior specified in software whenever possible
  - Interfaces for real world traffic

Experiment Framework

WARPnet

- New framework for real-time control and measurement
- Directly observe PHY/MAC events at all nodes in real-time
- Hooks throughout 802.11 Reference Design MAC and PHY

Experiment Framework

Baseline Implementation

Transmit Events

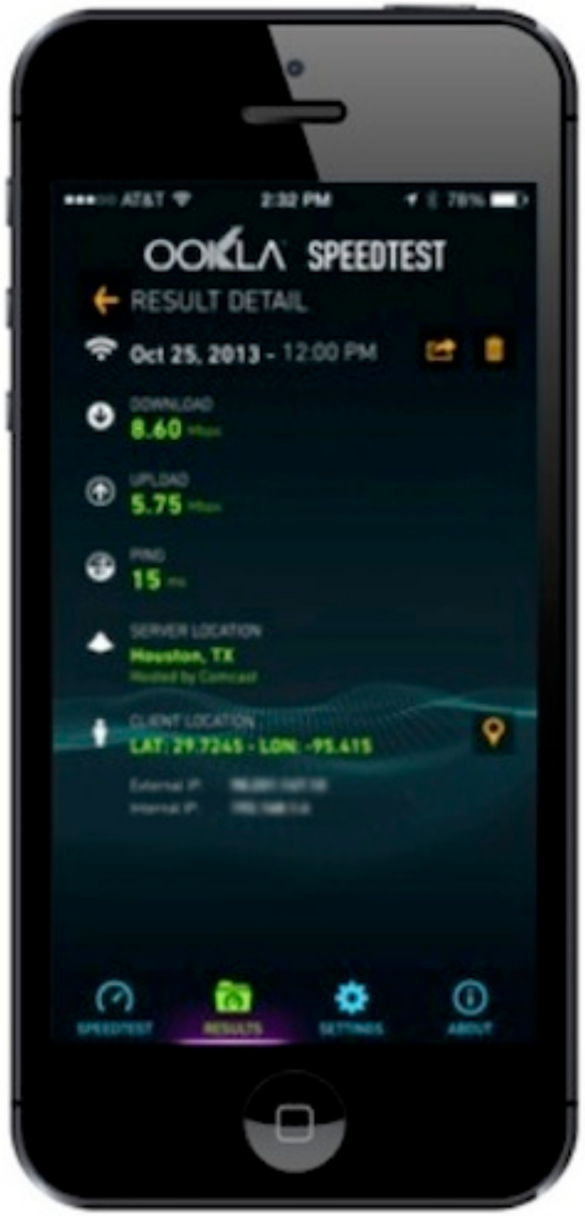
- Timestamp
- Tx Power, Rate, Length
- MAC headers
- Sequence number
- Tx result (ACK/timeout, number of re-transmissions, etc.)

Receive Events

- Timestamp
- Rx Power, Rate, Length
- MAC headers
- Sequence number
- Per-subcarrier channel estimates
- Rx result (FCS good/bad)

Experiment Framework

Demonstration



OOKLA SPEEDTEST

RESULT DETAIL

Oct 25, 2013 - 12:00 PM

DOWNLOAD 8.60

UPLOAD 5.75

PING 15

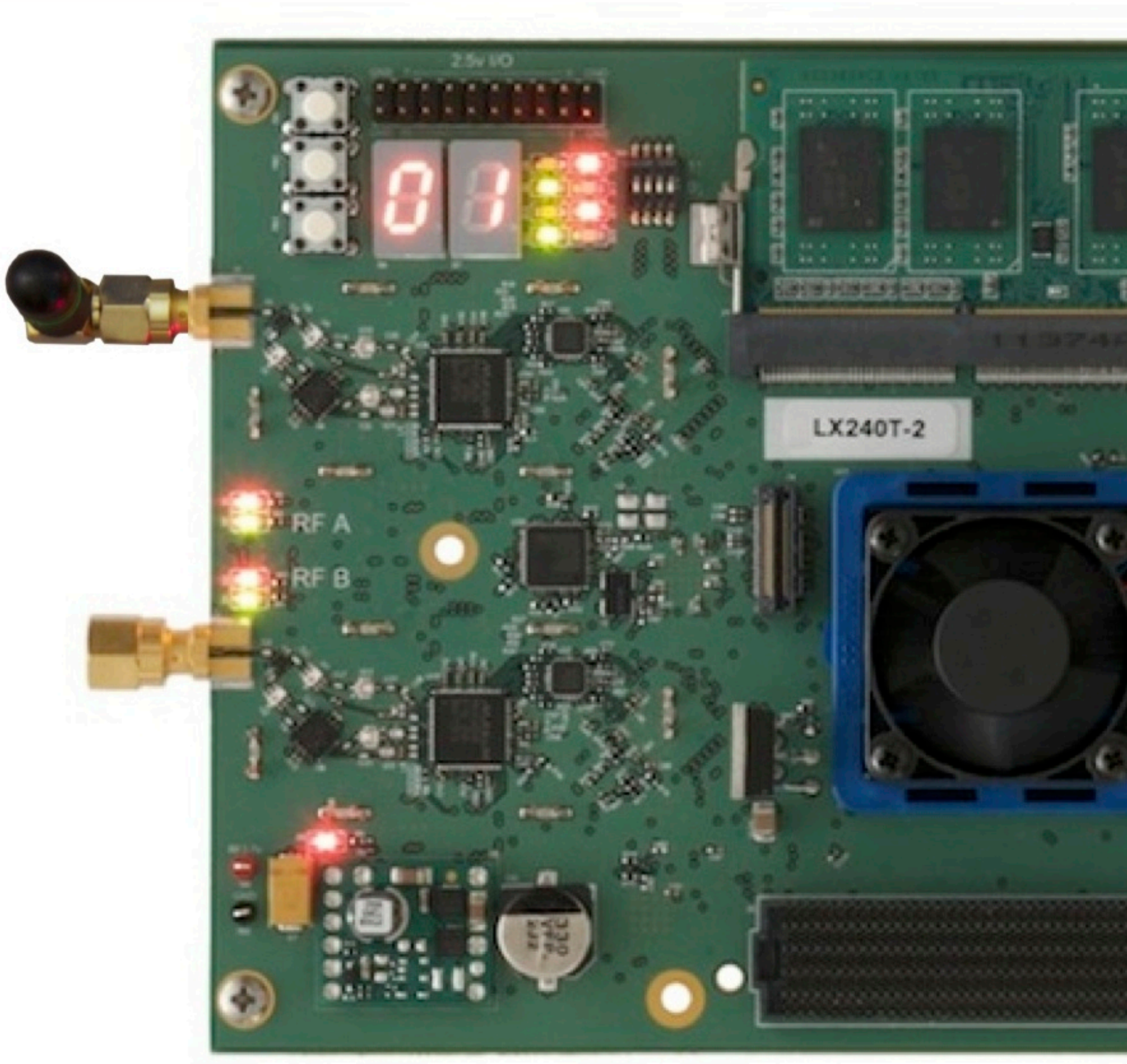
SERVER LOCATION Houston, TX

CLIENT LOCATION LAT: 29.7245 - LON: -95.415

Wi-Fi Client

802.11 Wireless Link

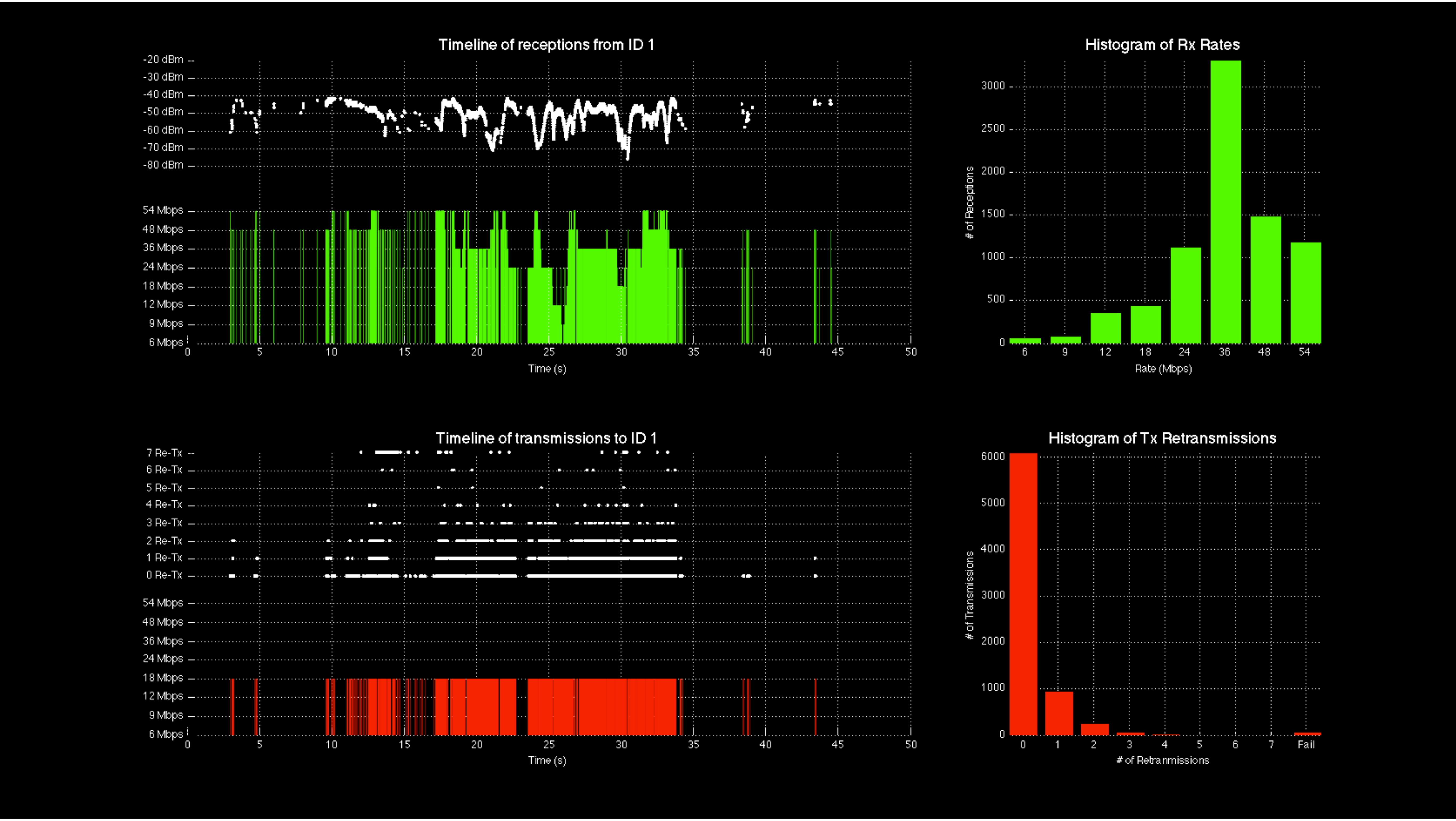
TCP Speed Test
(Uplink then Downlink)



Mango 802.11 Reference Design AP

Experiment Framework

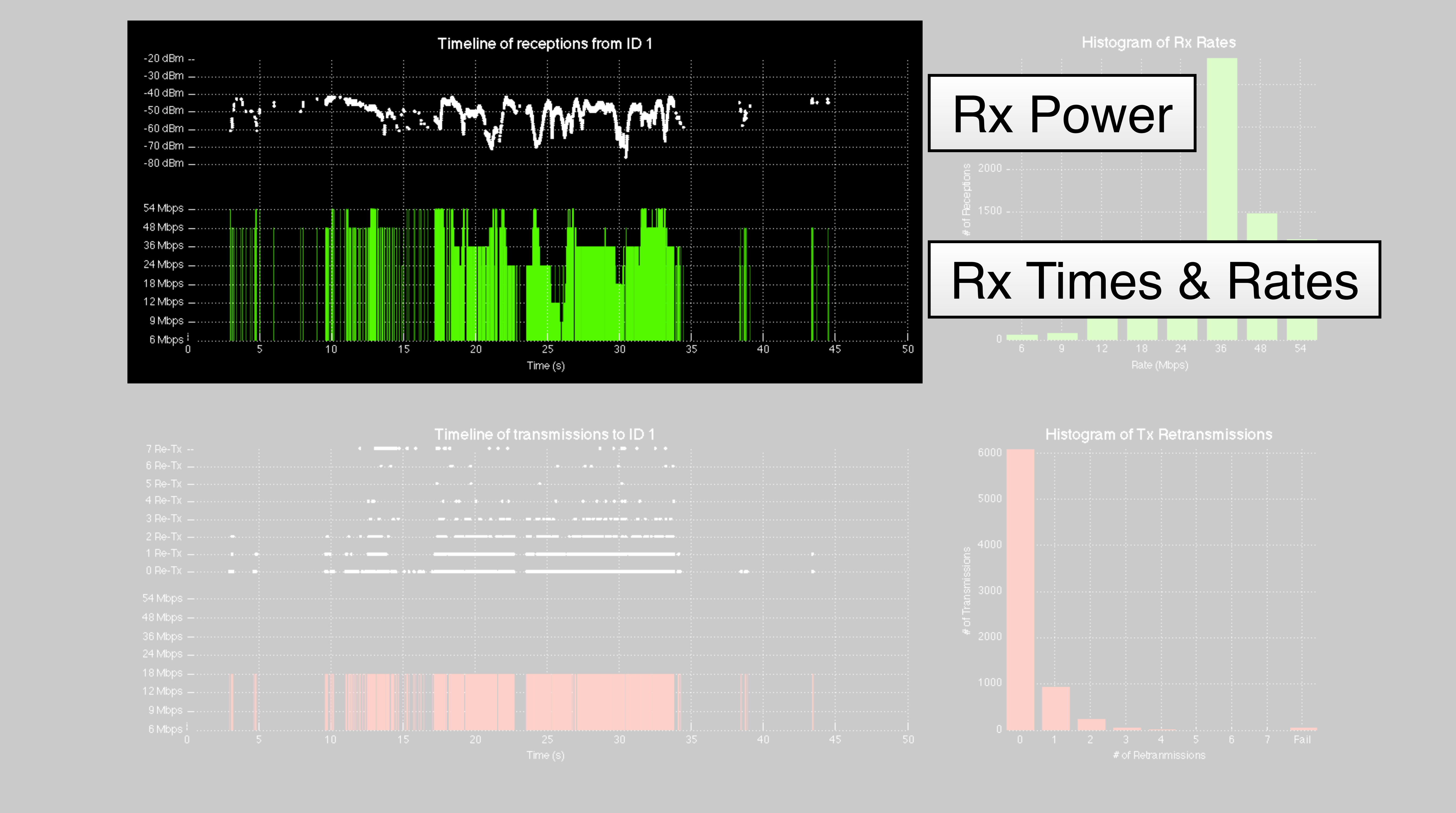
Demonstration



Visualization of WARPnet log for 50 second experiment

Experiment Framework

Demonstration



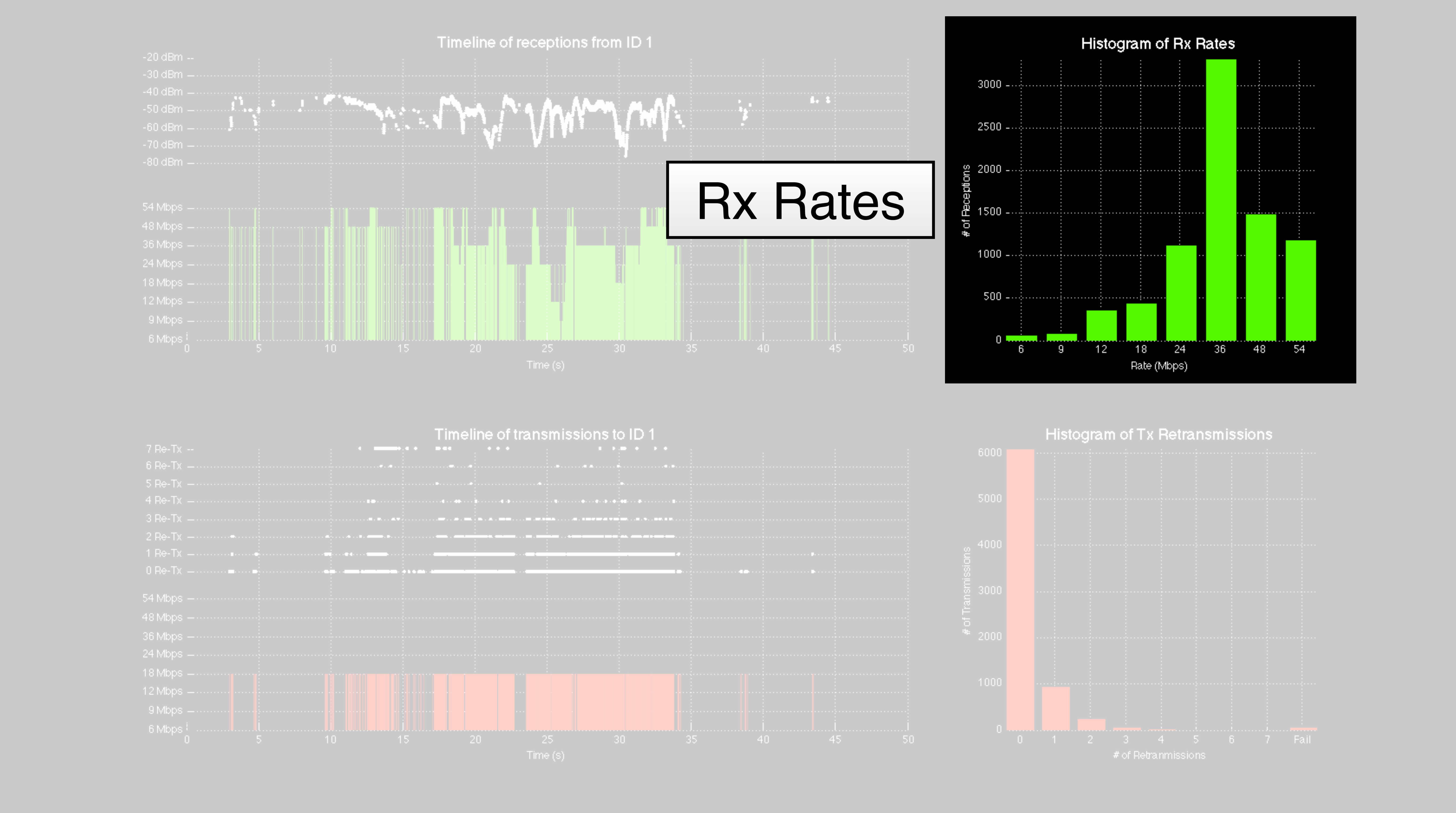
Rx Power

Rx Times & Rates

Visualization of WARPnet log for 50 second experiment

Experiment Framework

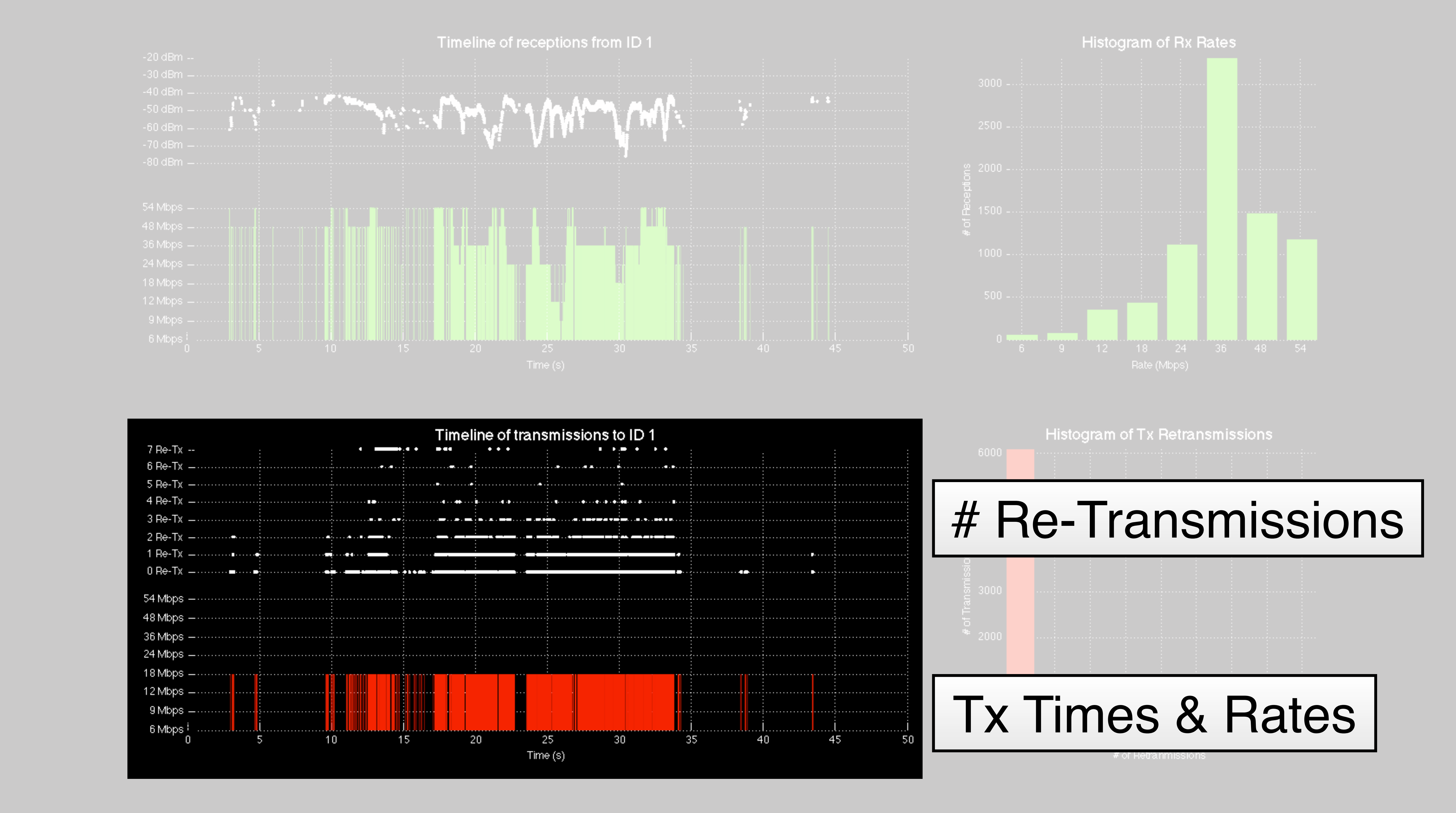
Demonstration



Rx Rates

Visualization of WARPnet log for 50 second experiment

Experiment Framework
Demonstration



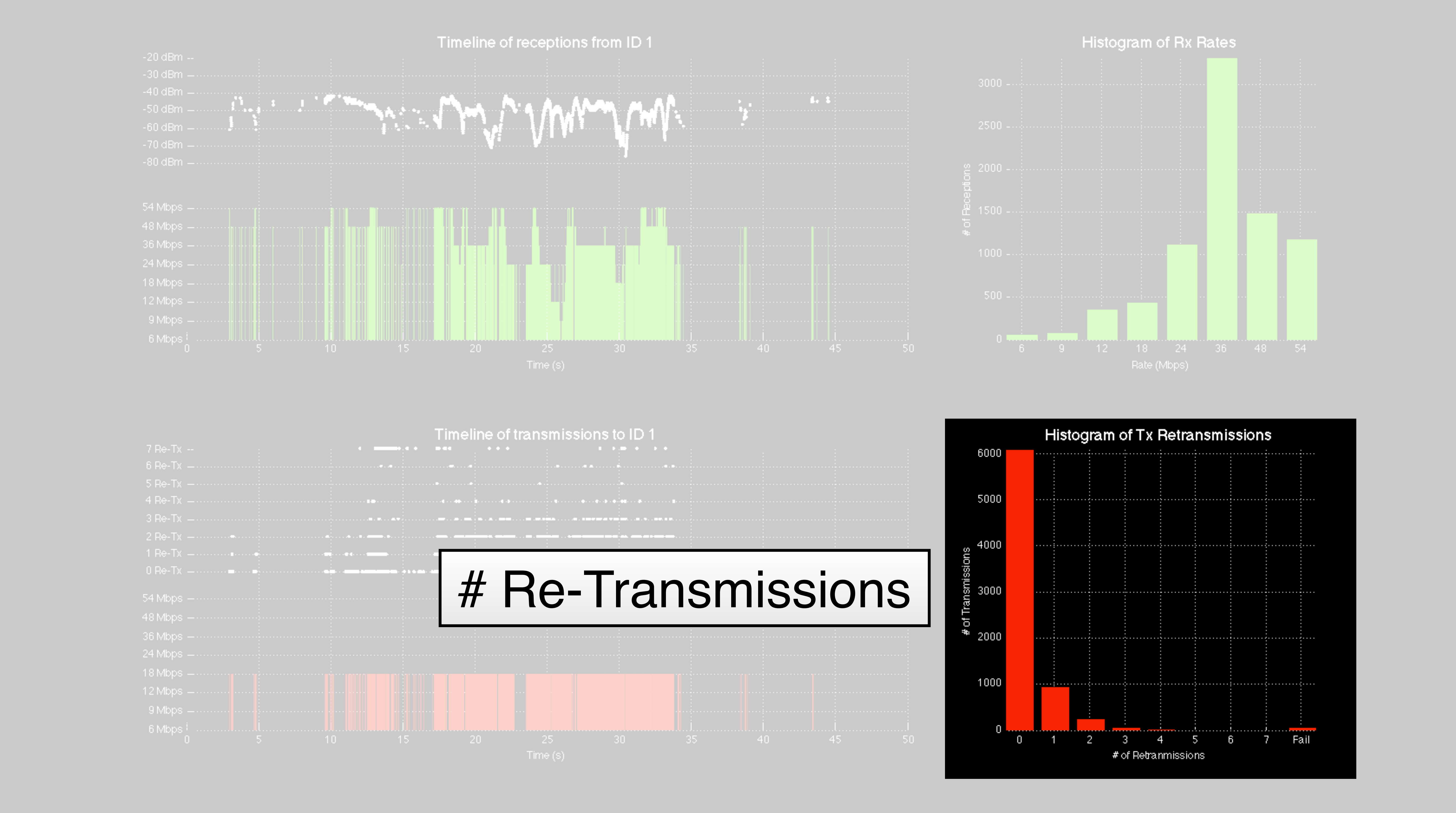
# Re-Transmissions

Tx Times & Rates

Visualization of WARPnet log for 50 second experiment

Experiment Framework

Demonstration

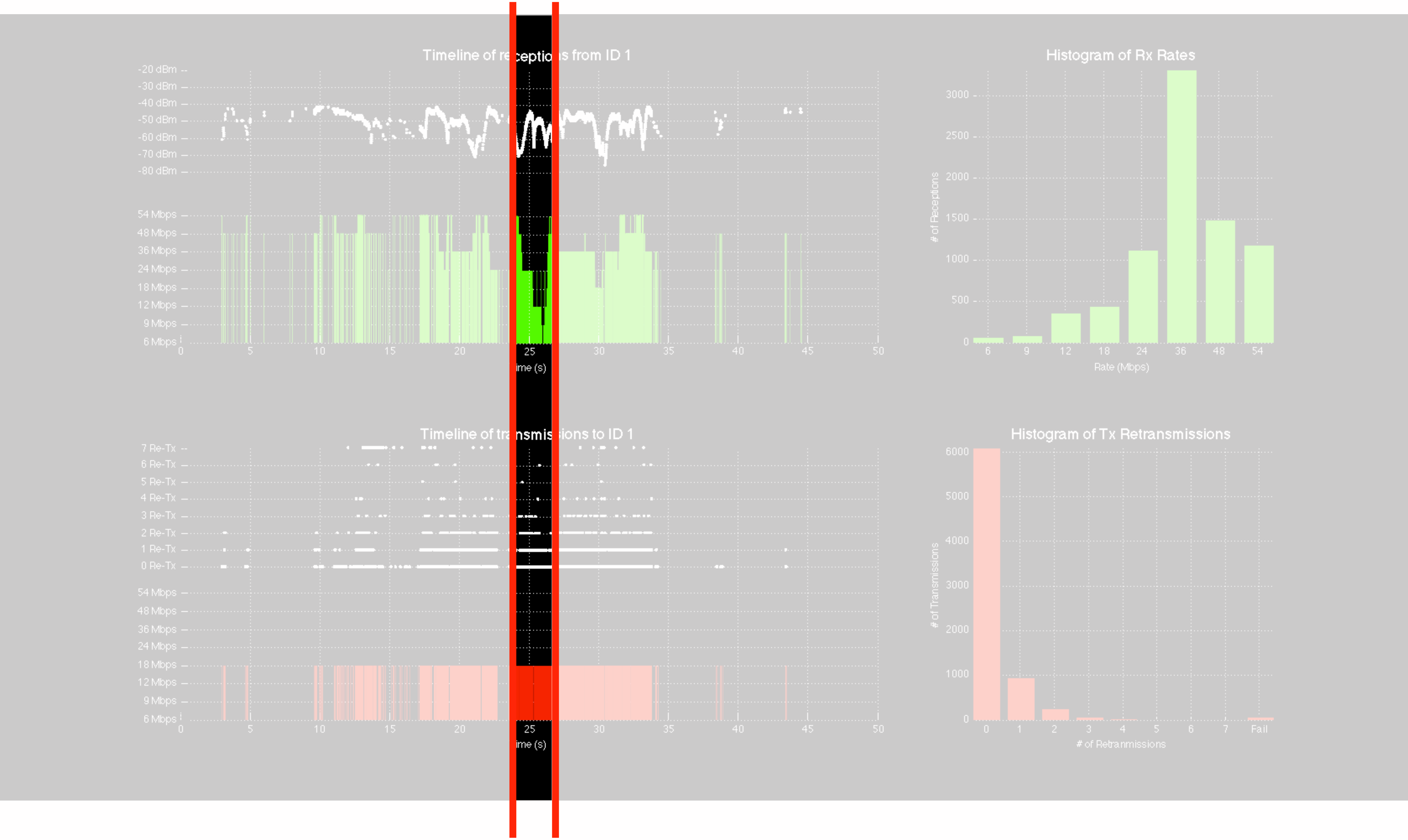


# Re-Transmissions

Visualization of WARPnet log for 50 second experiment

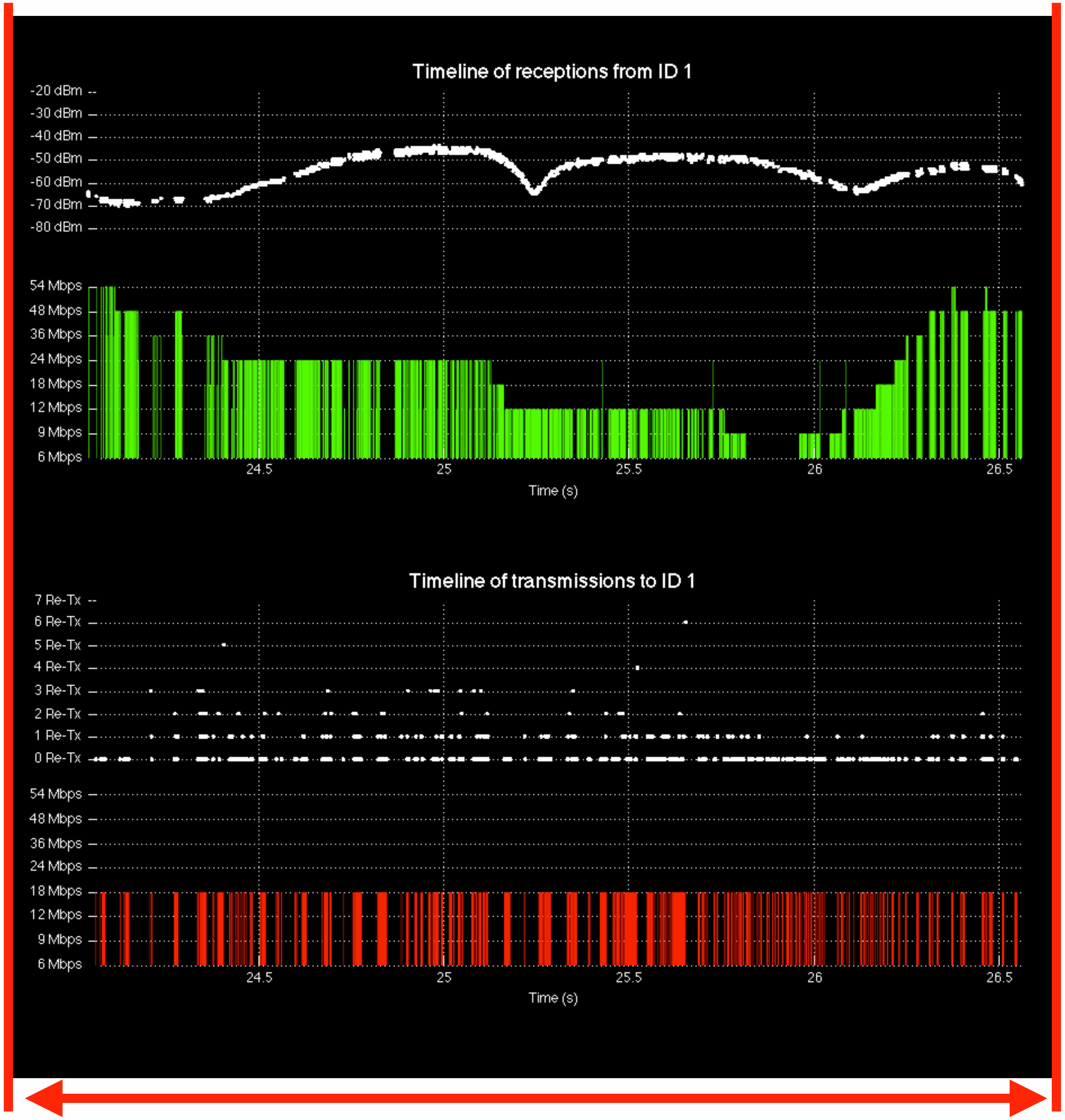
Experiment Framework

Demonstration



Experiment Framework

Demonstration



≈2.5 seconds

Rx Power

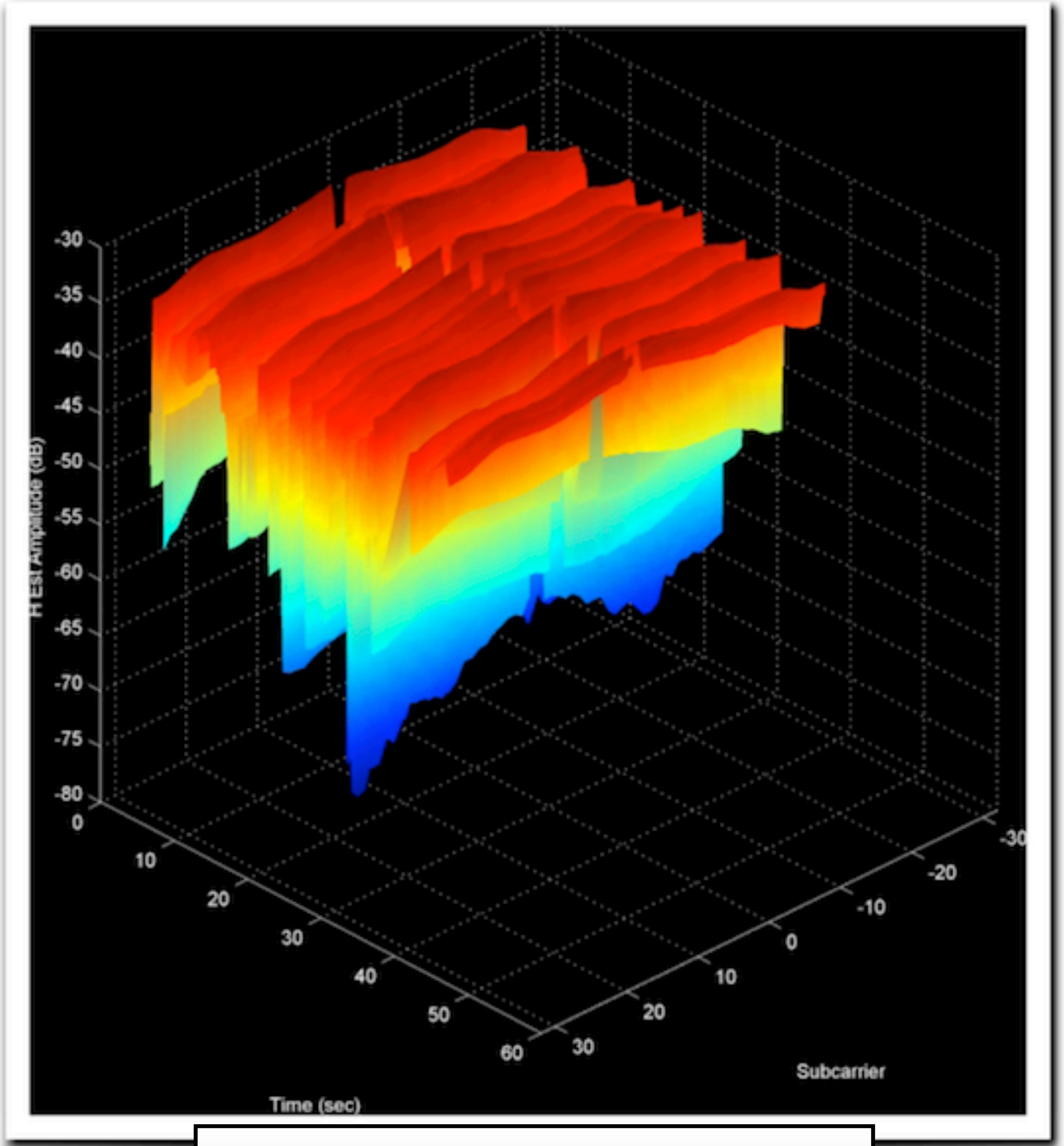
Rx Times & Rates

# Re-Transmissions

Tx Times & Rates

Experiment Framework

Demonstration



Channel Estimates

Experiment Framework

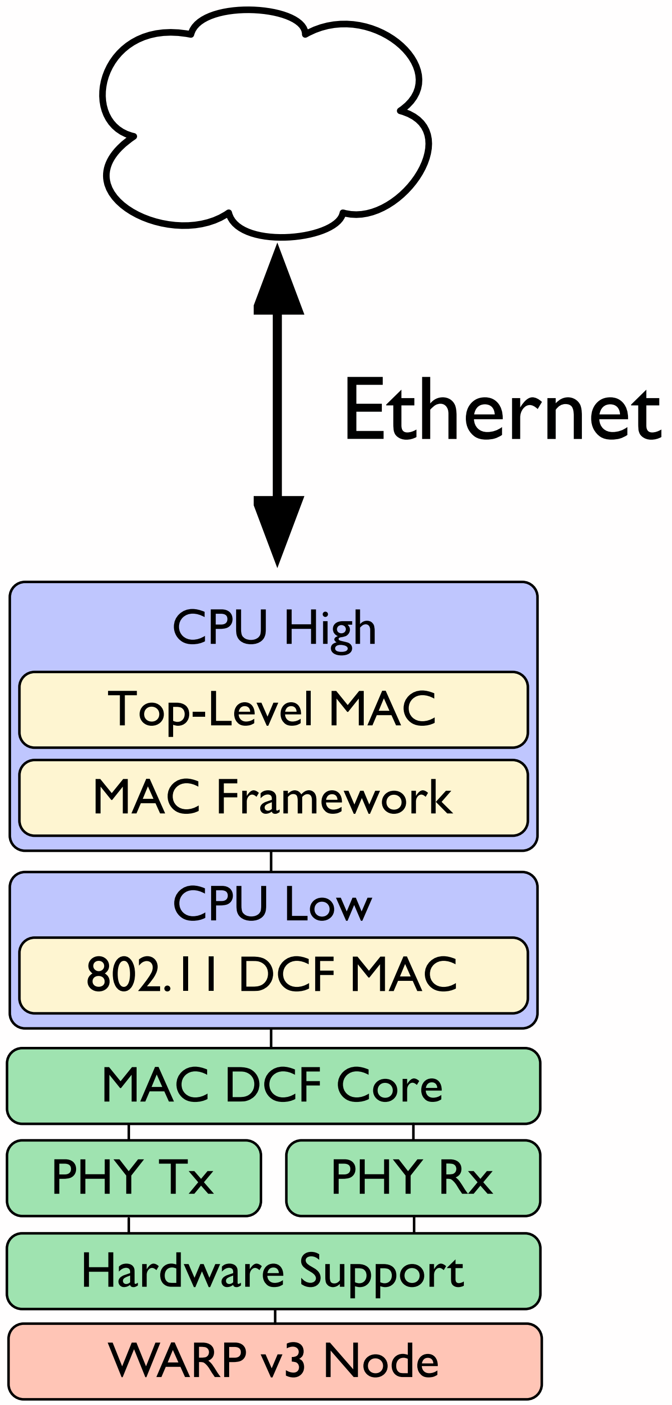
WARPnet

- Prototype framework built in MATLAB
    - Too slow for multi-node experiments with long logs
- Re-implementation in Python is underway
    - Will be part of 802.11 Reference Design v1.0 release

802.11 Reference Design

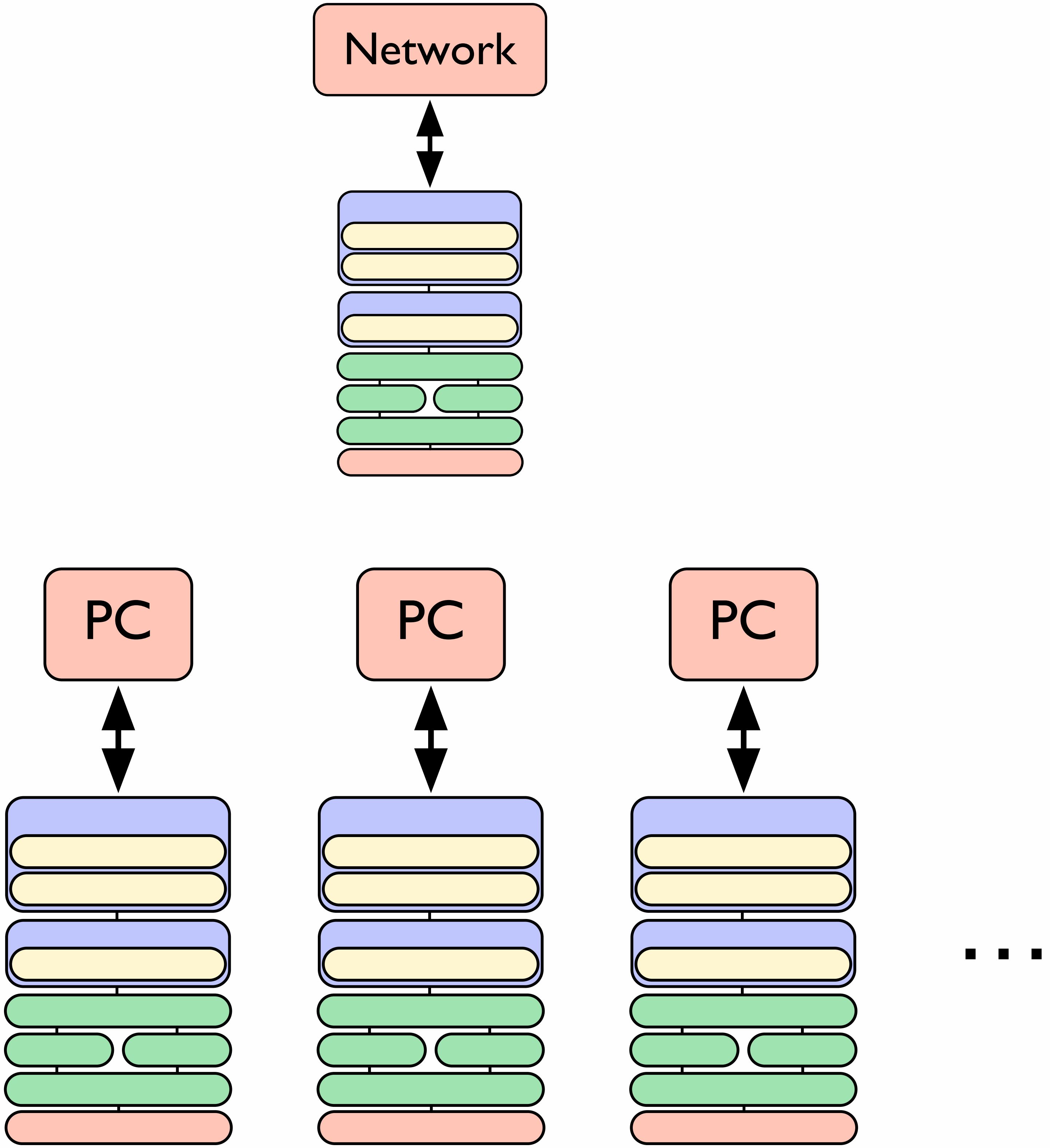
- **Interoperability**
  - Real-time MAC & PHY in FPGA
  - No compromises on MAC timing or PHY features
  - AP & station implementations
- **Experimental Visibility**
  - Hooks throughout MAC & PHY
  - Framework for running experiments and understanding results
- **Extensibility**
  - All source code open
  - Behavior specified in software whenever possible
  - Interfaces for real world traffic

Networks & Real Traffic



- Ethernet encapsulation is straightforward
- Wired-wireless bridging already built
  - AP wired to PC or network
  - PC wired to each STA

Networks & Real Traffic



AP

Network

Stations

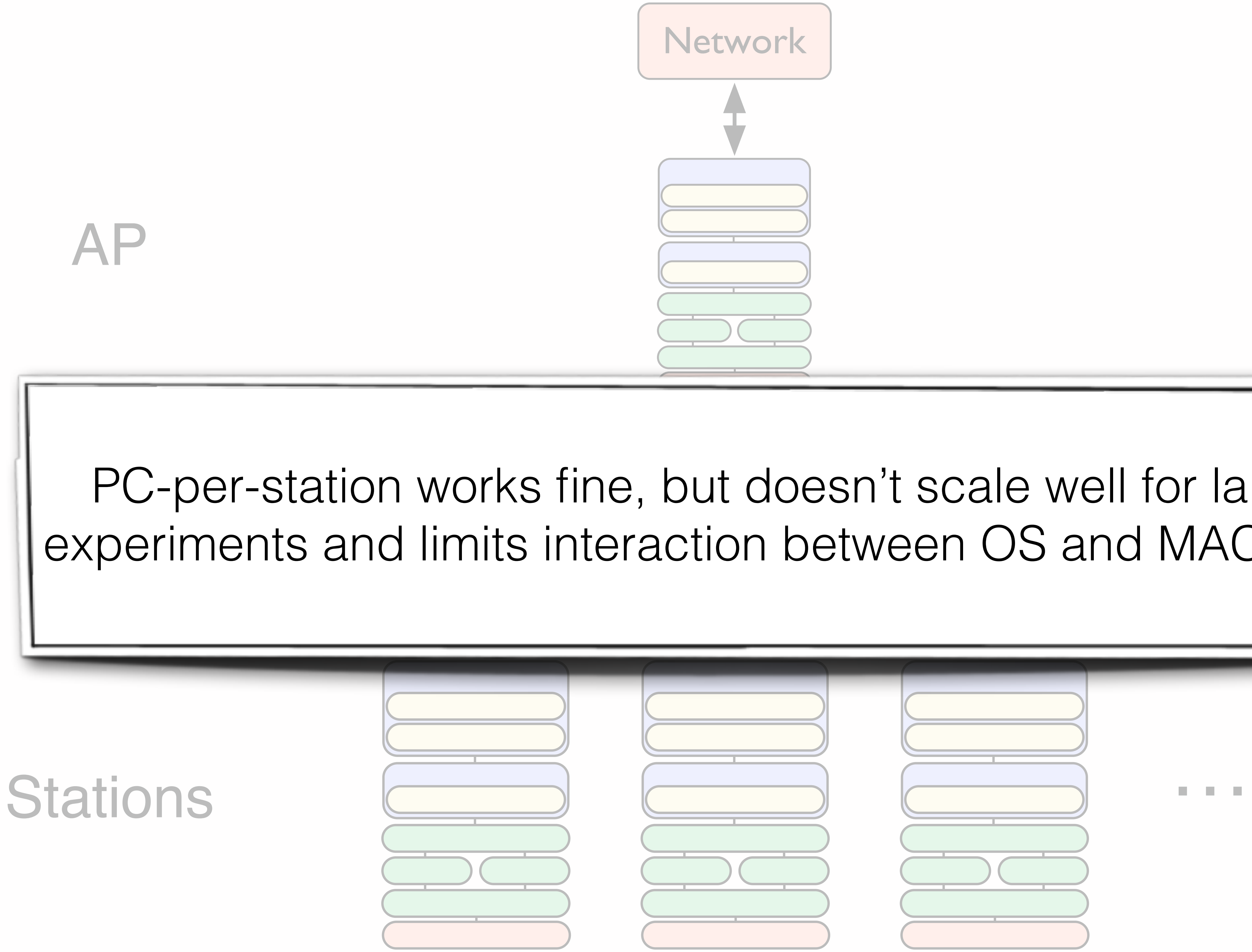
PC

PC

PC

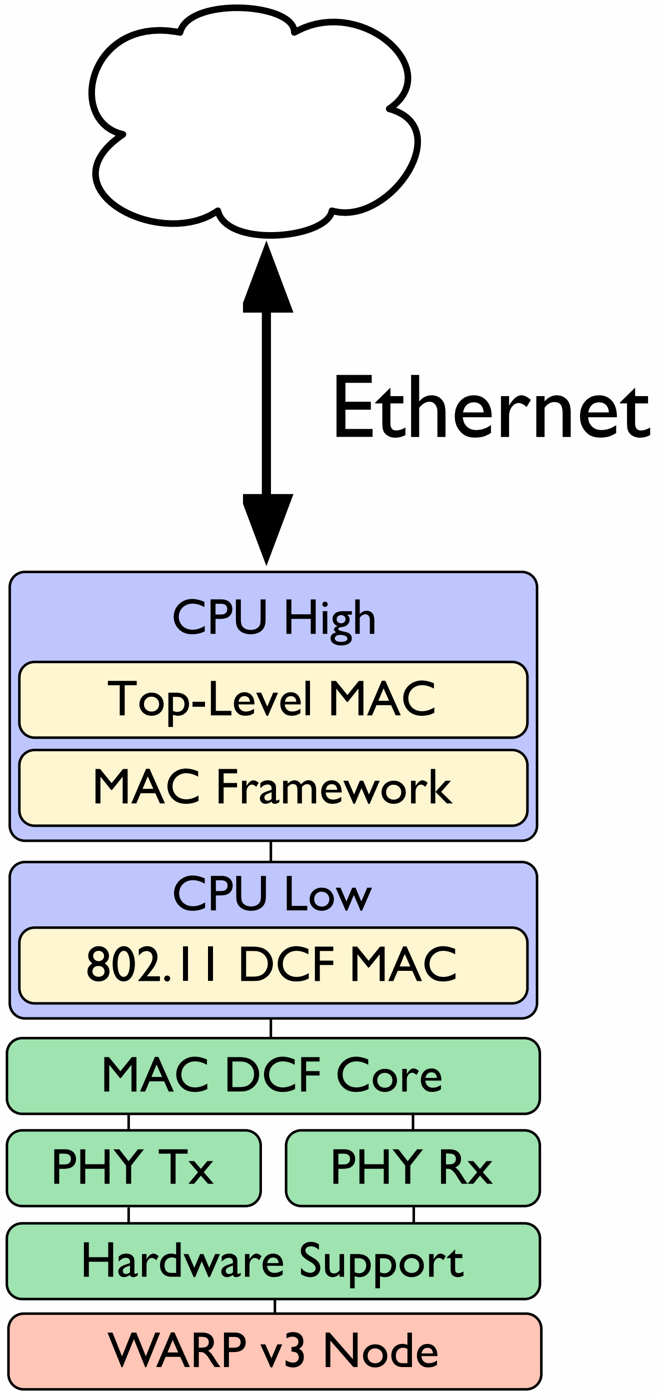
…

Networks & Real Traffic



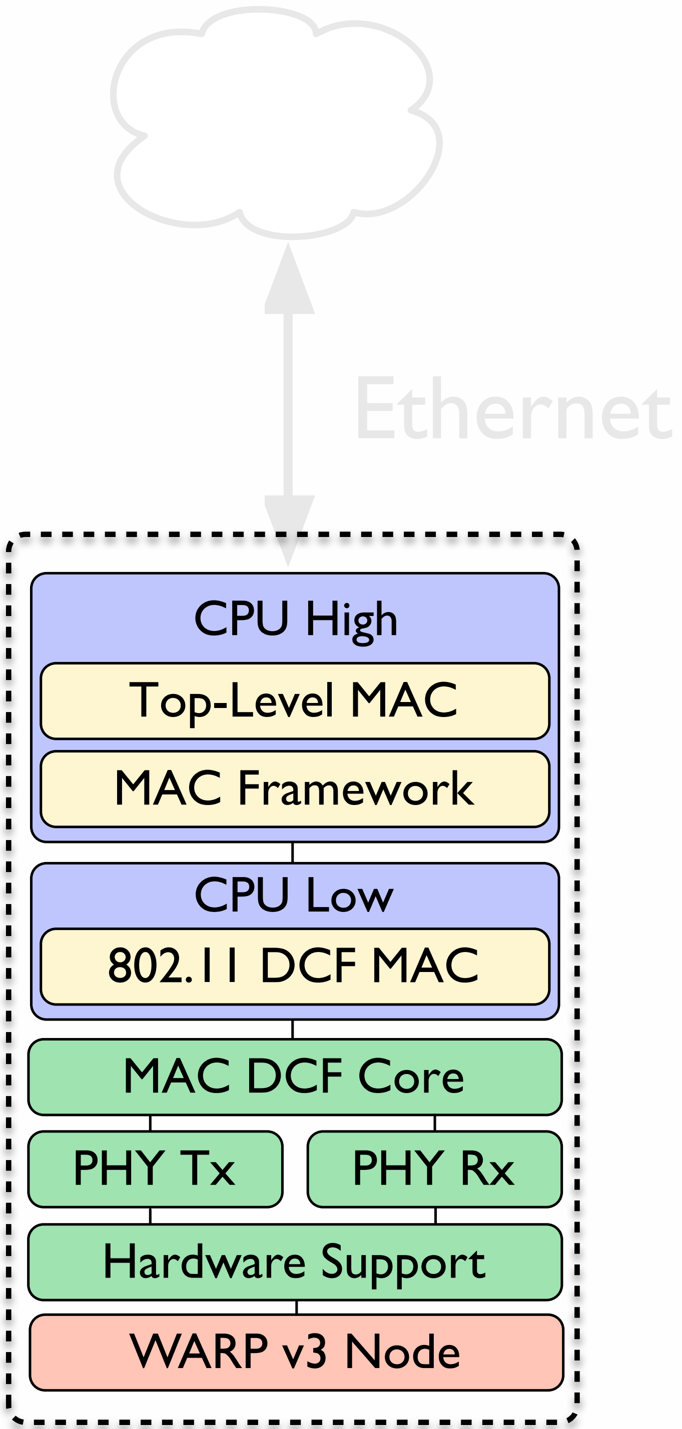
PC-per-station works fine, but doesn't scale well for larger experiments and limits interaction between OS and MAC/PHY.

Networks & Real Traffic



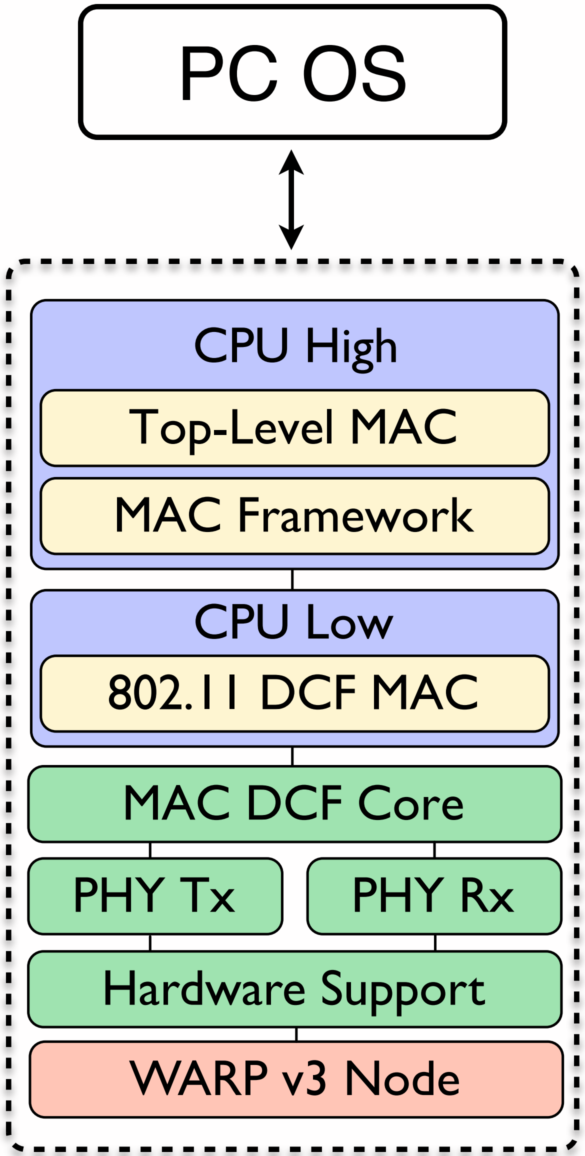
Networks & Real Traffic

802.11 NIC?

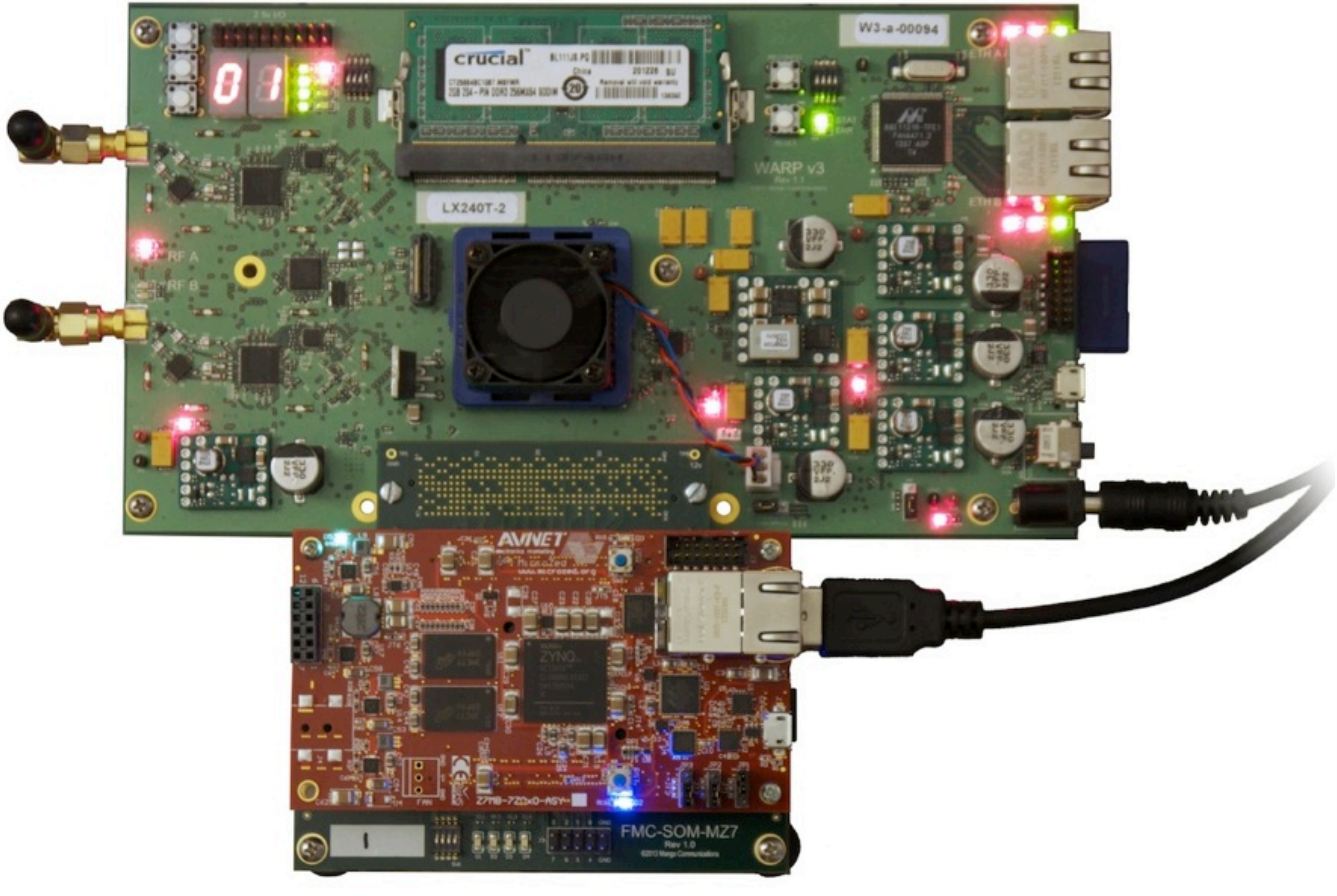


Networks & Real Traffic

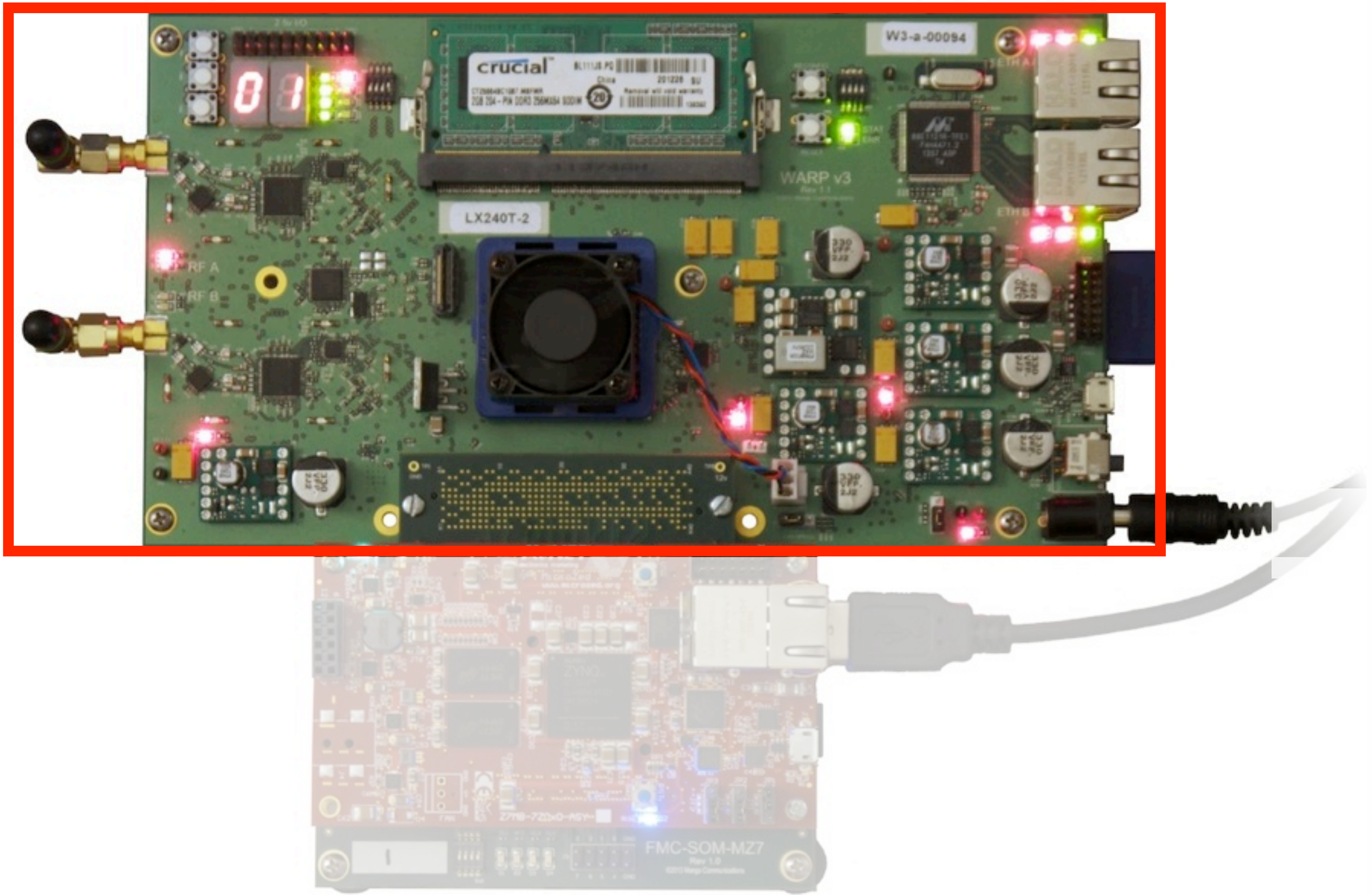
802.11 NIC?



Proof of Concept: Wireless NIC

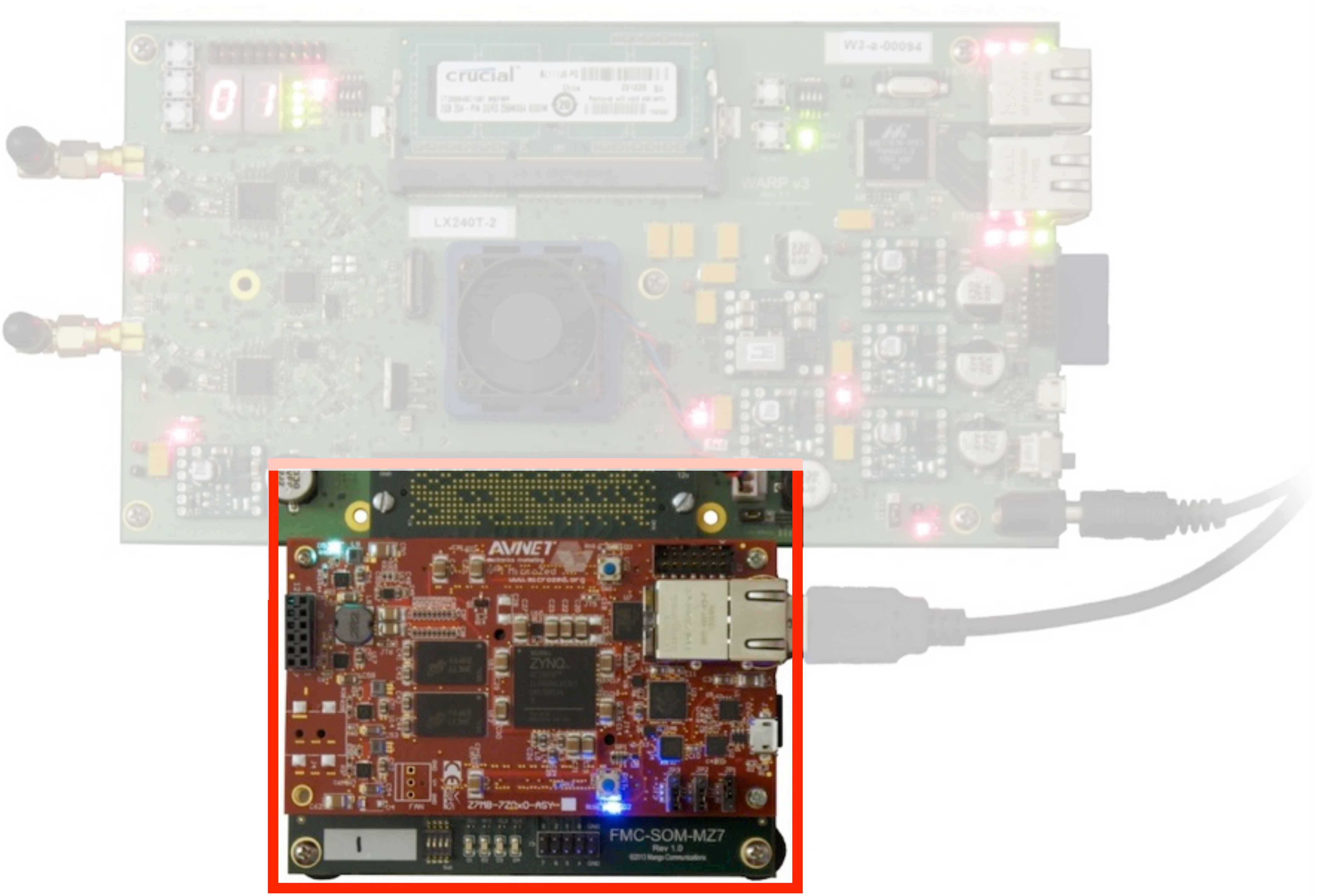


Proof of Concept: Wireless NIC



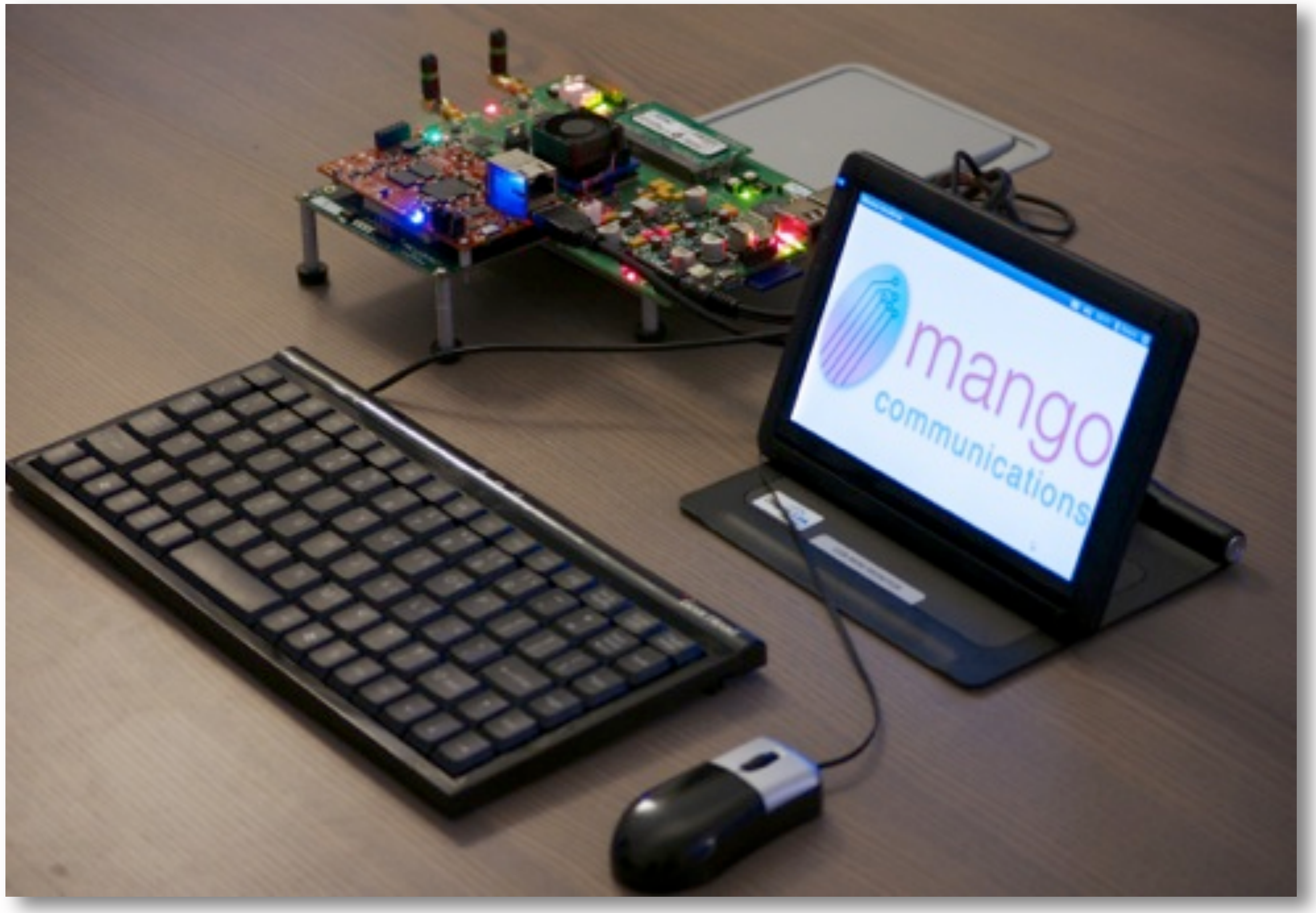
WARP v3 Node

Proof of Concept: Wireless NIC



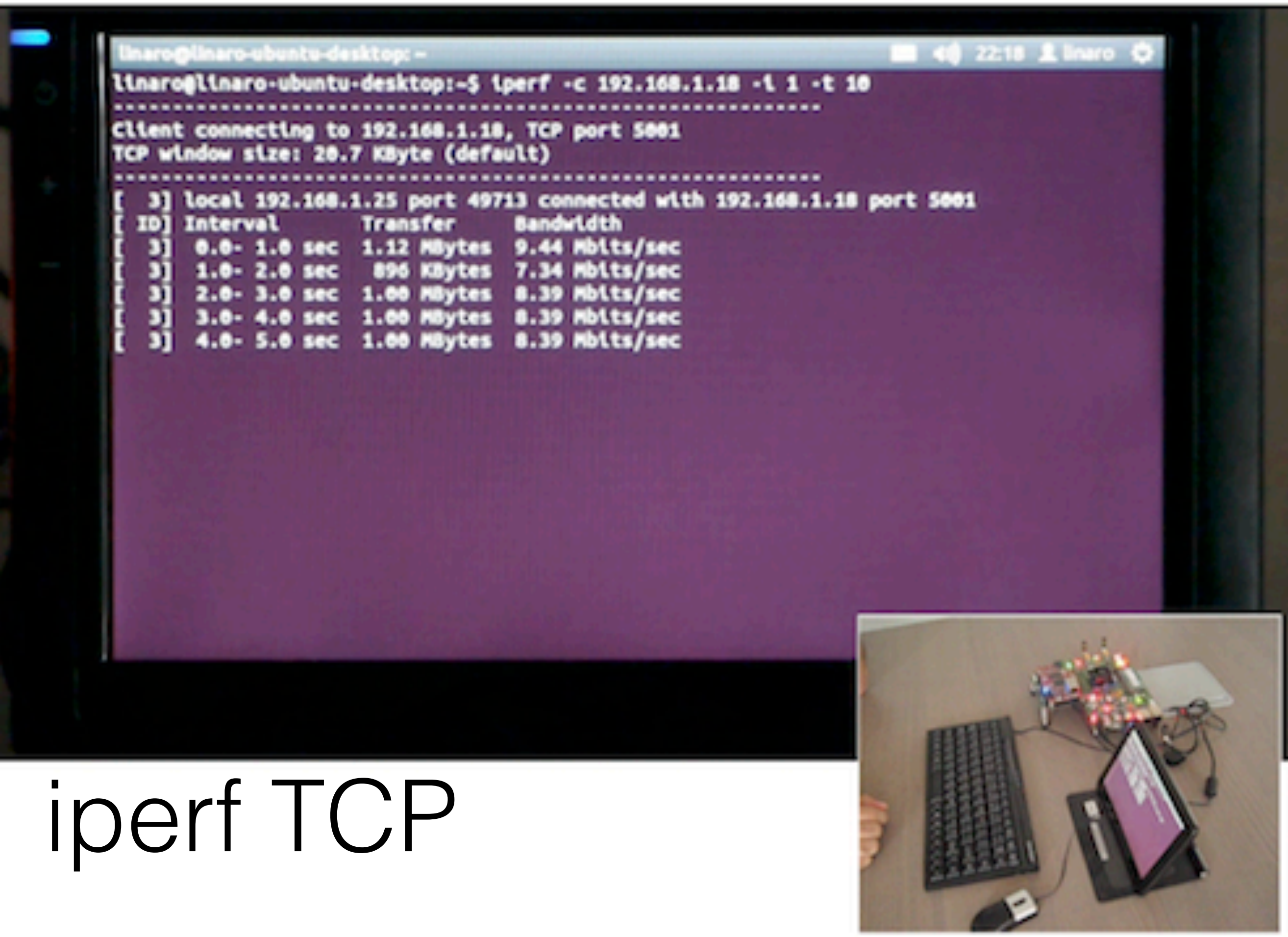
Zynq FMC Module using Avnet MicroZed SOM

Proof of Concept: Wireless NIC



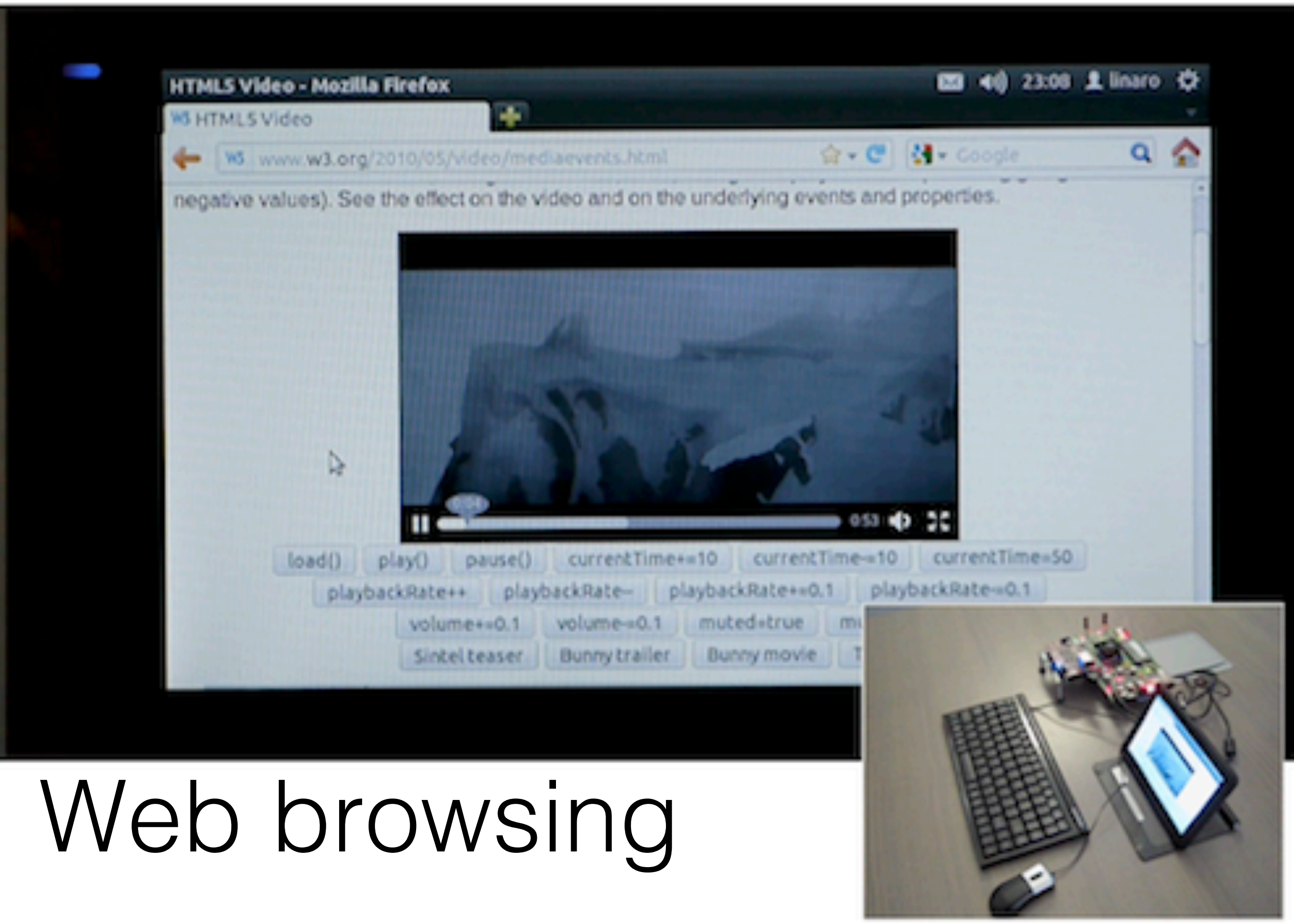
- Proof of concept
  - Ubuntu Desktop in Zynq
  - WARP v3 + 802.11 as NIC
  - User-mode driver via TUN/TAP
  - USB display, keyboard & mouse
  - Observe- no Ethernet connections

Proof of Concept: Wireless NIC



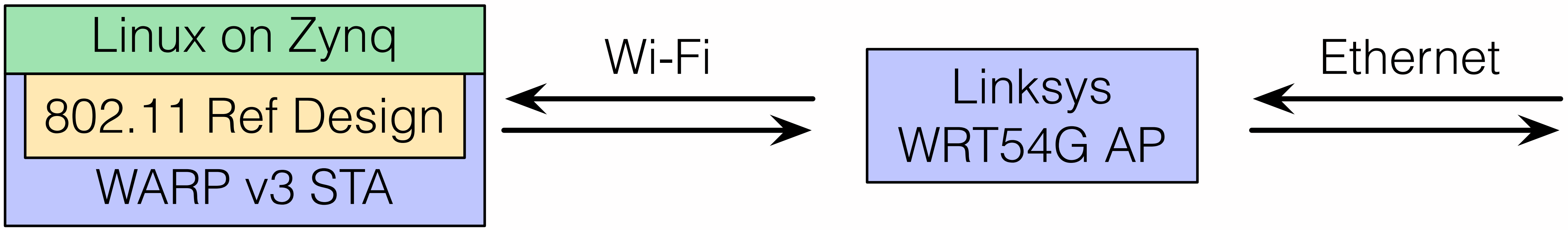
```
linaro@linaro-ubuntu-desktop:~$ iperf -c 192.168.1.18 -i 1 -t 10
------------------------------------------------------------
Client connecting to 192.168.1.18, TCP port 5001
TCP window size: 20.7 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.25 port 49713 connected with 192.168.1.18 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  1.0- 2.0 sec   896 KBytes  7.34 Mbits/sec
[  3]  2.0- 3.0 sec  1.00 MBytes  8.39 Mbits/sec
[  3]  3.0- 4.0 sec  1.00 MBytes  8.39 Mbits/sec
[  3]  4.0- 5.0 sec  1.00 MBytes  8.39 Mbits/sec
```

iperf TCP



negative values). See the effect on the video and on the underlying events and properties.

Web browsing


Linux on Zynq

802.11 Ref Design

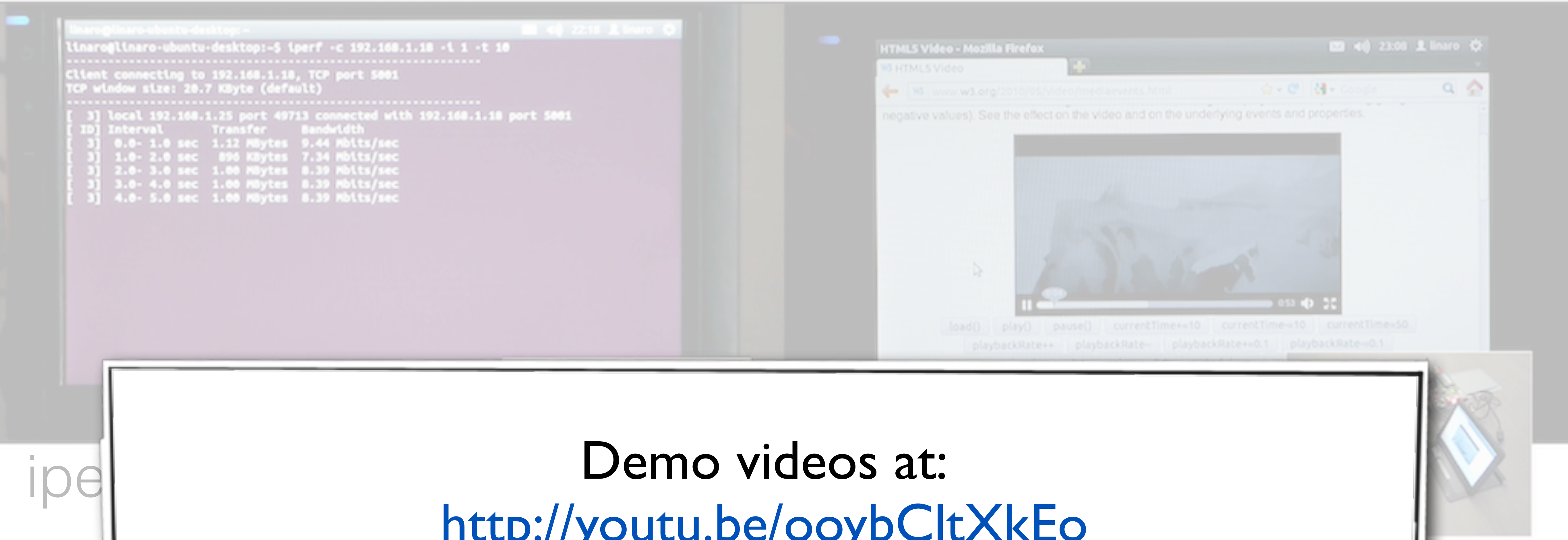
WARP v3 STA

Wi-Fi

Linksys WRT54G AP

Ethernet

Proof of Concept: Wireless NIC

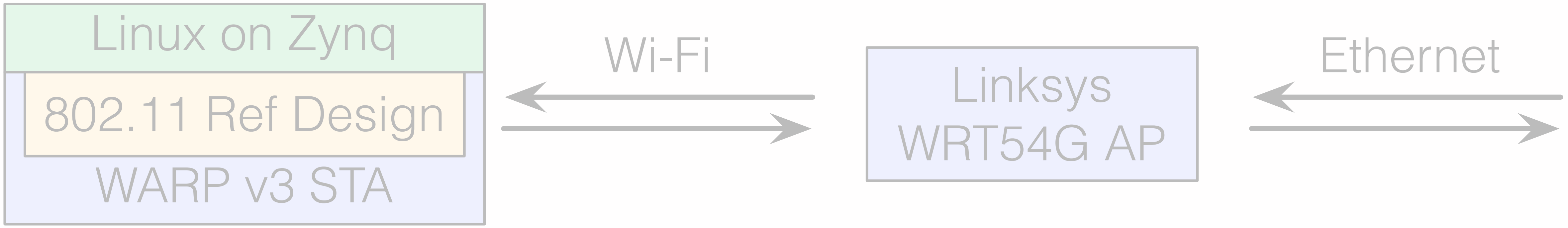


```
linaro@linaro-ubuntu-desktop:~$ iperf -c 192.168.1.18 -i 1 -t 10
------------------------------------------------------------
Client connecting to 192.168.1.18, TCP port 5001
TCP window size: 20.7 KByte (default)
------------------------------------------------------------
[  3] local 192.168.1.25 port 49713 connected with 192.168.1.18 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 1.0 sec  1.12 MBytes  9.44 Mbits/sec
[  3]  1.0- 2.0 sec   896 KBytes  7.34 Mbits/sec
[  3]  2.0- 3.0 sec  1.00 MBytes  8.39 Mbits/sec
[  3]  3.0- 4.0 sec  1.00 MBytes  8.39 Mbits/sec
[  3]  4.0- 5.0 sec  1.00 MBytes  8.39 Mbits/sec
```

negative values). See the effect on the video and on the underlying events and properties.

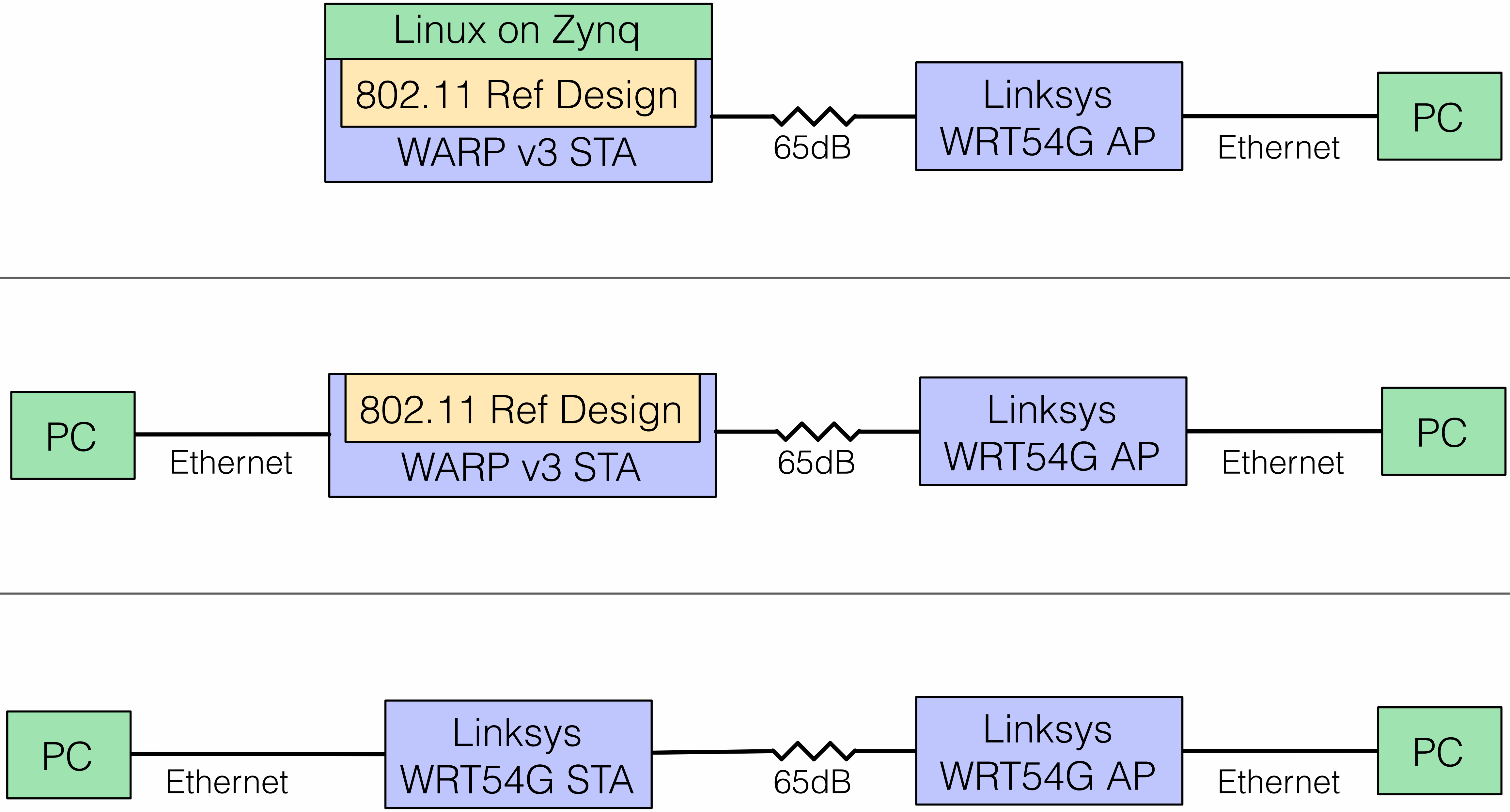
ipe

Demo videos at:
http://youtu.be/ooybClt XkEo



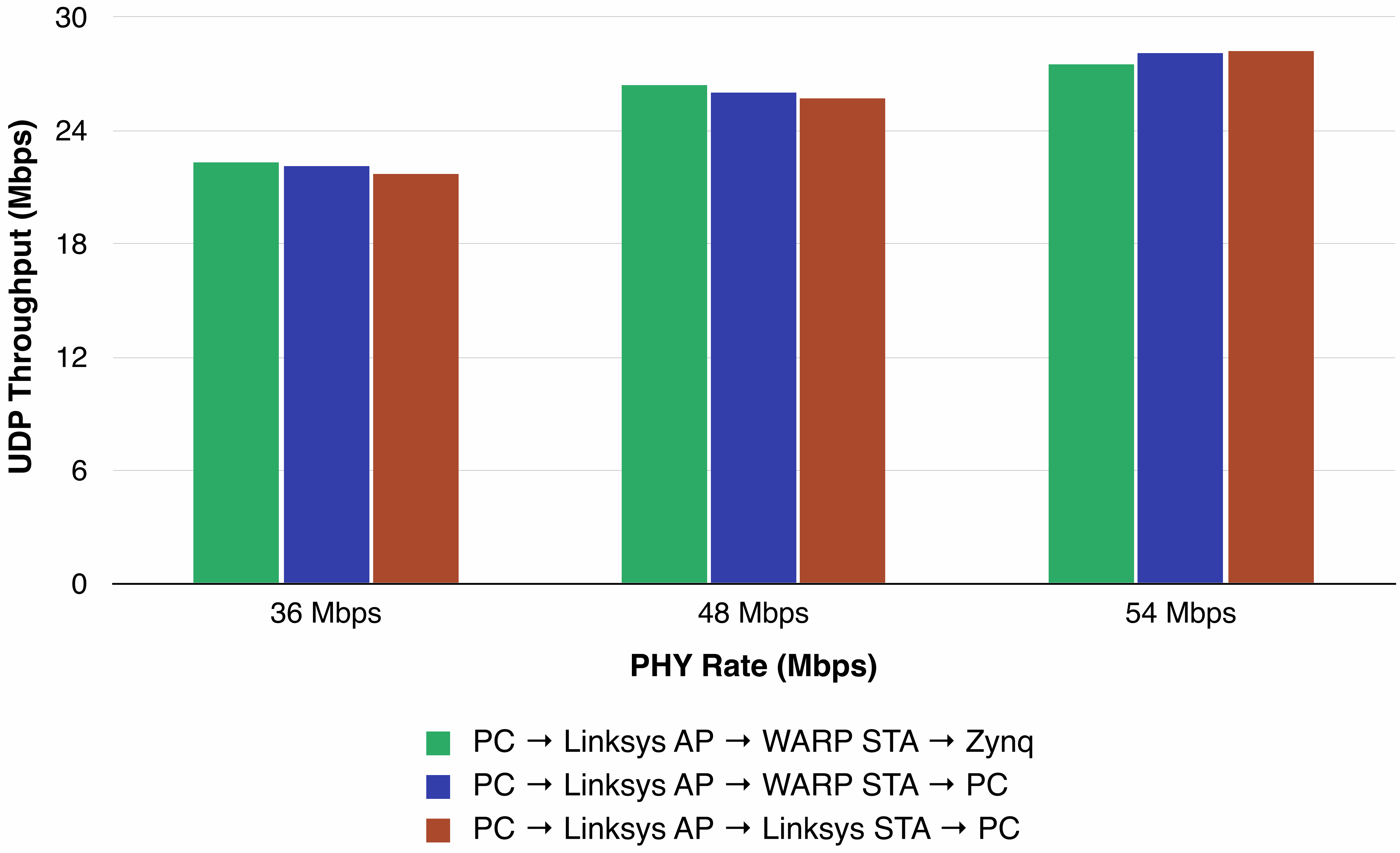
Proof of Concept: Wireless NIC

Preliminary Characterization



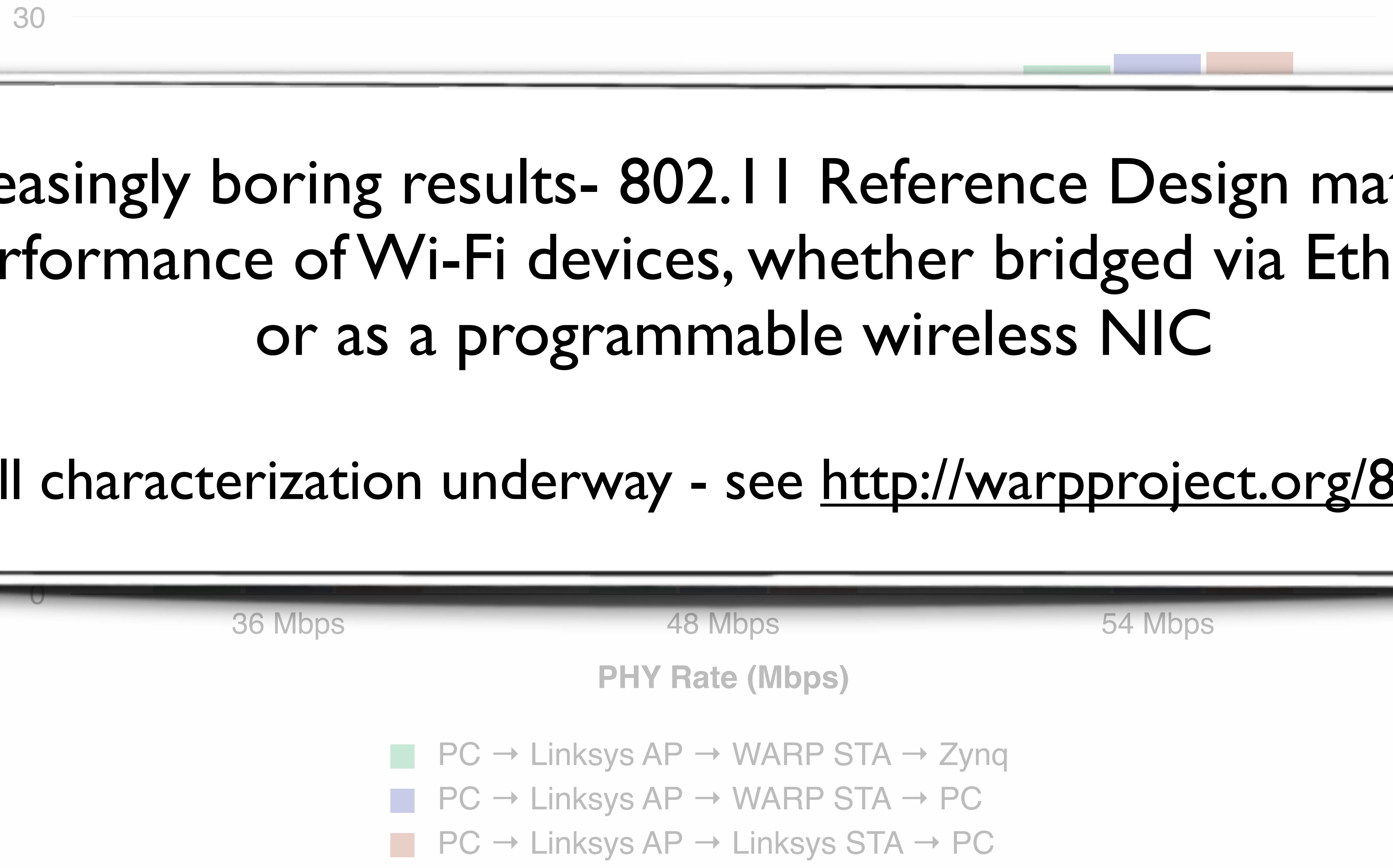
Proof of Concept: Wireless NIC

Preliminary Characterization



Proof of Concept: Wireless NIC

Preliminary Characterization



Pleasingly boring results- 802.11 Reference Design matches performance of Wi-Fi devices, whether bridged via Ethernet or as a programmable wireless NIC

(Full characterization underway - see http://warpproject.org/802.11)

Mango 802.11 Reference Design

- All source, documentation & characterization online:

  http://warpproject.org/802.11

- Current version is 0.6-beta
- Aiming for 1.0 release in January
  - WARPnet framework in Python
  - Cleaned and commented C code
  - Migration to Xilinx ISE 14.7
- Feedback is always welcome

mango
communications