Enabling MAC Protocol Implementations on Software-defined Radios

George Nychis, Thibaud Hottelier, Zhuochen Yang, Srinivasan Seshan, and Peter Steenkiste

Carnegie Mellon University

Wireless Media Access Control Protocols

- No single one-size-fits-all MAC
  - definition of performance, and how to achieve it, varies greatly
- Wireless MACs: *extremely diverse*
  - long-haul, mesh, lossy, dense, mobile …
- Novel fundamental wireless optimizations:
  - MIXIT, PPR, Successive IC, ZigZag, …

How can we easily implement diverse MAC protocols and optimizations?

Current MAC Protocol Development

Wireless NICs

- + High Performance (DSP)
- + Low cost ($30)

- – Closed source
  - most of the MAC
- – Fixed functionality:
  - Physical layer, 2.4GHz

Software Radios

- + Various open source platforms
- + *Fully* reprogrammable
  - and various frequencies!
- – Higher cost ($700-$10K)
- – Lower performance (GPP)
  - large delays

Implementing MACs on SDRs

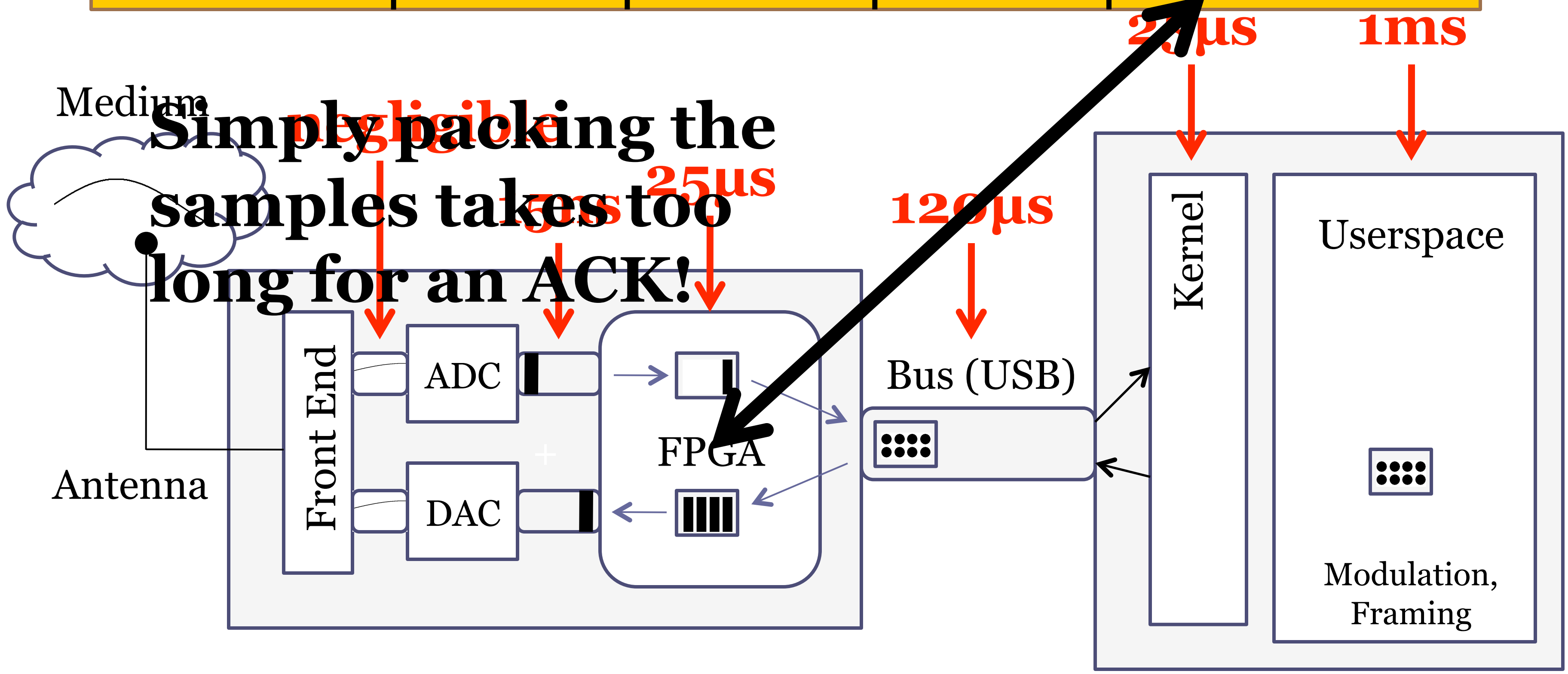
- Various projects using SDRs for evaluation:
  - MIXIT, PPR, Successive IC, ZigZag …
- The above all use GNU Radio + USRP:
  - “extreme” SDR→ *all processing in userspace*
  - great as a research platform (PHY+MAC)
- No high-performance MAC protocol implemented on GNU Radio & USRP

Outline of the Talk

- *Why* MAC implementation on SDRs *is challenging*
- *How* to overcome SDR limitations, enabling *high-performance* and *flexible* MAC implementations
  - A novel approach: ***Split-functionality* API**
- Present evaluation of the first high-performance MACs on an extreme architecture
- Implications and Conclusions

“Extreme” SDR Architecture

	CS	SIFS	DIFS	ACK-TO
802.11	<10µs	10µs	28µs	22µs

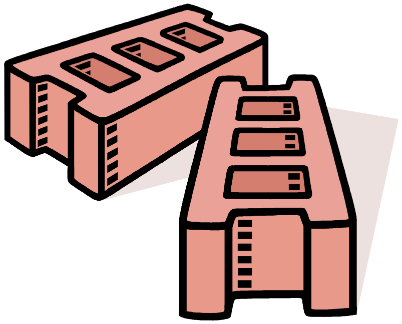


Solutions to Bypass Delay

- Common: *move the layers* closer to the frontend
  - WARP: PHY+MAC on the radio hardware
  - SORA: PHY+MAC in kernel, core ded., SIMD, LUT
- Completely viable solutions, *but*:
  - costly (hardware is more complex, WARP: $10K+)
  - can require special toolkits (e.g., XPS)
  - requires embedded architecture knowledge
  - portability and interface (SIMD, PCI-E)

An Alternate Solution

- *Split-functionality* approach, break all core MAC functions (e.g., carrier sense) in to 2 pieces:
    - 1 *small piece* on the radio hardware (performance)
    - 1 *piece* on the host (flexibility)
- Then, develop an API for the core functions
    - logical control channel and per-block metadata
    - per-packet control of the functions & hardware
    - applicable to other SDR architectures



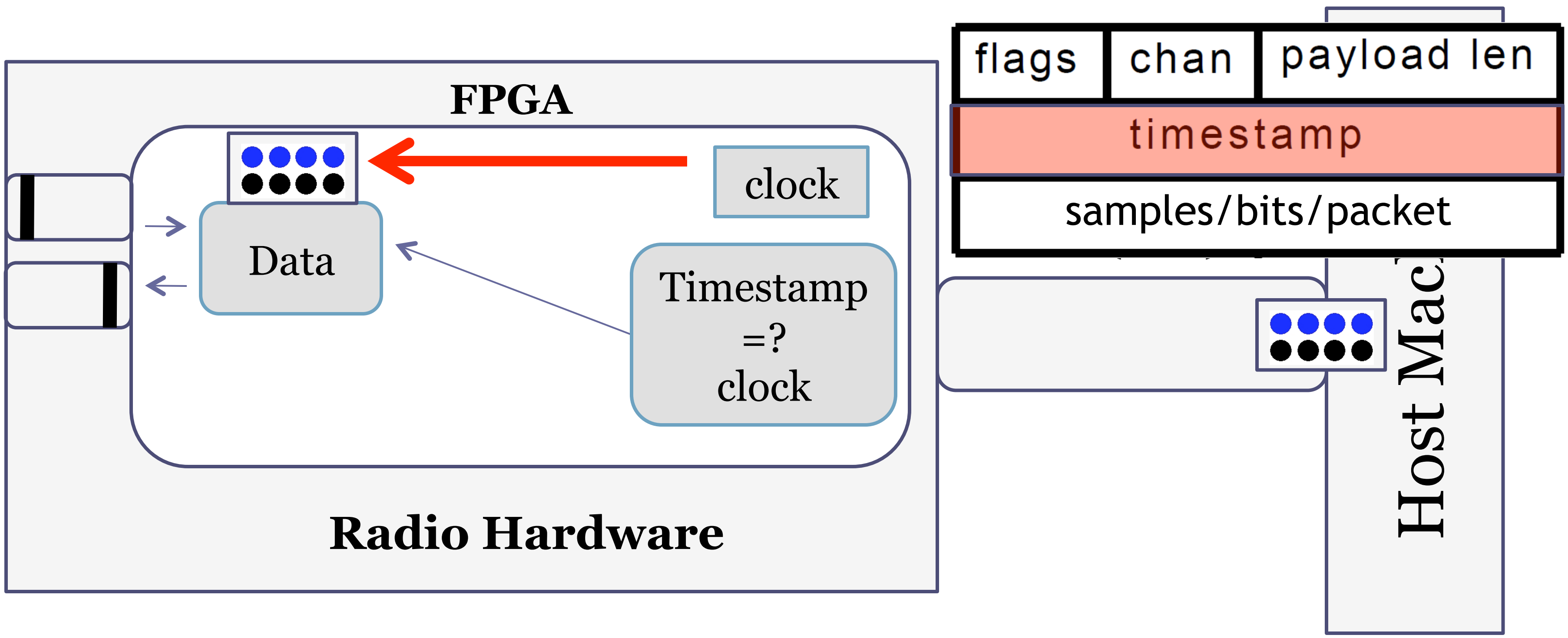
Indentifying the Core MAC Functions

- Building blocks of MAC protocols:
  - carrier sense
  - precision scheduling
  - backoff
  - fast-packet detection
  - dependent packet generation
  - fine-grained radio control
- Difficult to claim that any list is corre
  - reasonable first "toolbox"

Random Backoff
Power Control
SIFS/DIFS
ACK
Synchronization
MIMO
Frequency Hop
Guard Periods
Slot Times
Rate Adaptation
Beacons
Carrier Sense

Precision Scheduling

- *Split-functionality API approach:*
  - Scheduling on the host (flexibility)
  - Triggering on the hardware (performance)
  - requires a lead time that varies based on architecture



Precision Scheduling

- *Split-functionality API approach:*
  - Scheduling on the host (flexibility)
  - Triggering on the hardware (performance)
  - requires a lead time that varies based on architecture

- Average measured error in TX scheduling using GNU Radio and USRP:

	Split-func.	Kernel	Host
Precision	**125ns**	35µs	1ms

Revisiting the Core MAC Functions

- Building blocks of MAC protocols:
  - carrier sense
  - precision scheduling
  - backoff
  - fast-packet detection
  - dependent packet generation
  - fine-grained radio control

- Difficult to claim that any list is correct and complete
  - reasonable first “toolbox”

Fast-Packet Detection

- **Goal:** *accurately detect packets in the hardware*
- The longer it takes to detect a packet, the longer a response packet takes (*dependent packet)*
  - Can be used to trigger pre-modulated DPs (ACKs)
- Demodulate only when necessary (CPU intensive)
  - provides host confidence of a packet in the stream
  - not only detect a packet, but that it is for this radio
- Can be used in other architectures:
  - SORA: used to trigger core dedication
  - Kansas SDR: battery powered, reduces consumption

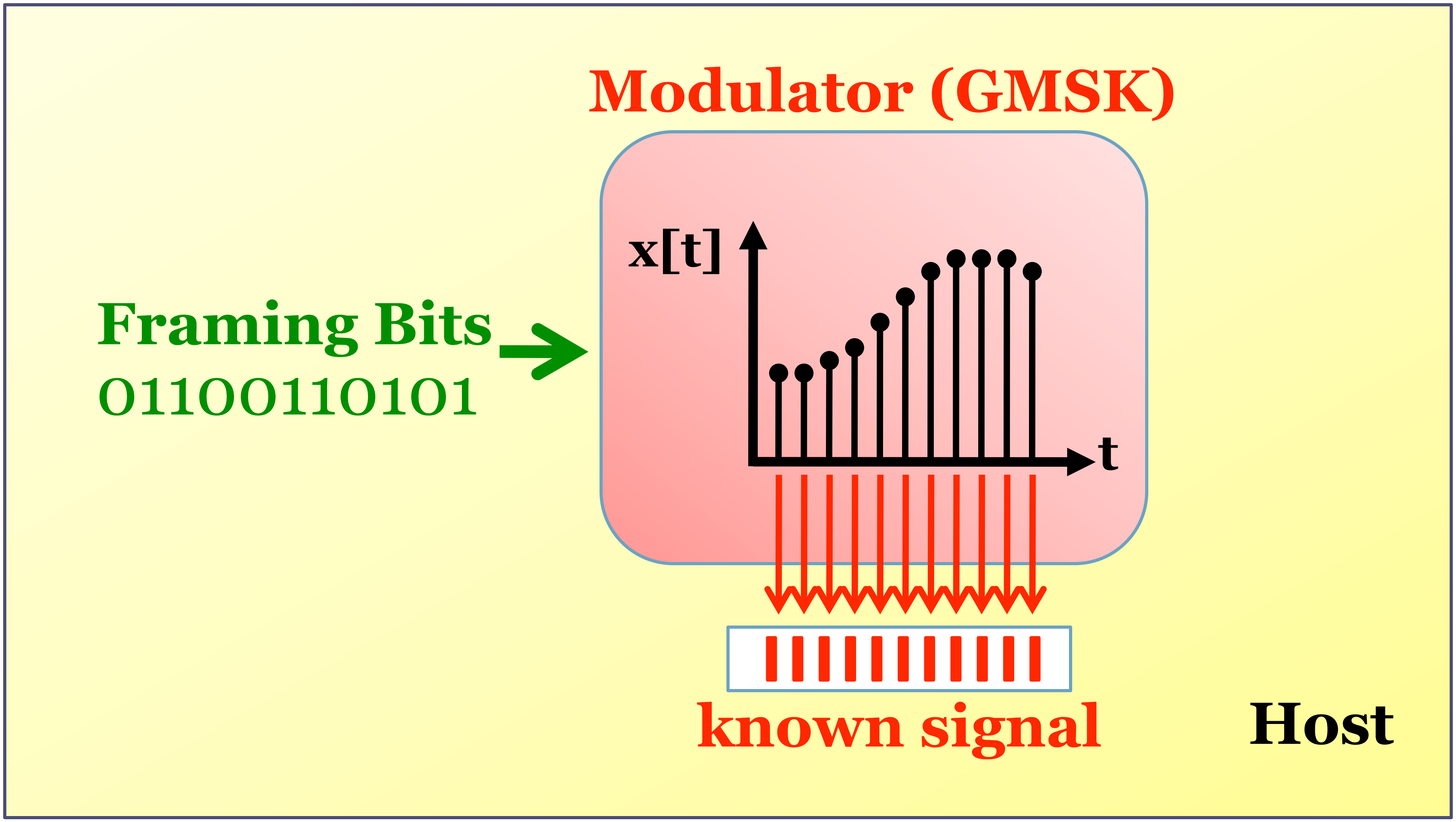
Fast-Packet Detection in Hardware

- Perform signal detection using a ***matched filter***
  - optimal linear filter for maximizing SNR
  - widely used technique in communications
  - flexible to all modulation schemes
  - cross-correlation of unknown & known signals

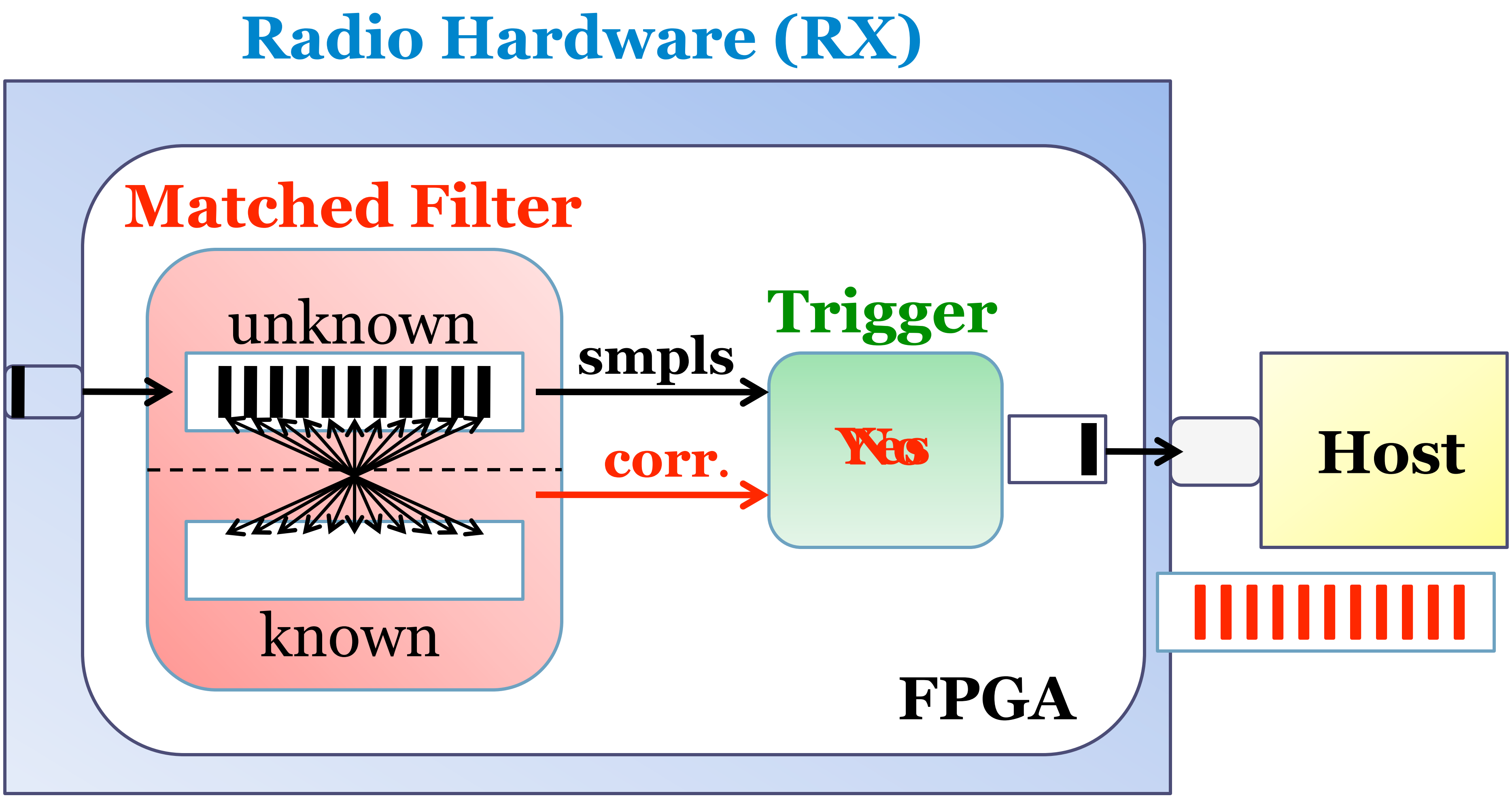
Incoming sample stream

Modulated framing bits

Packet Detection Host Setup



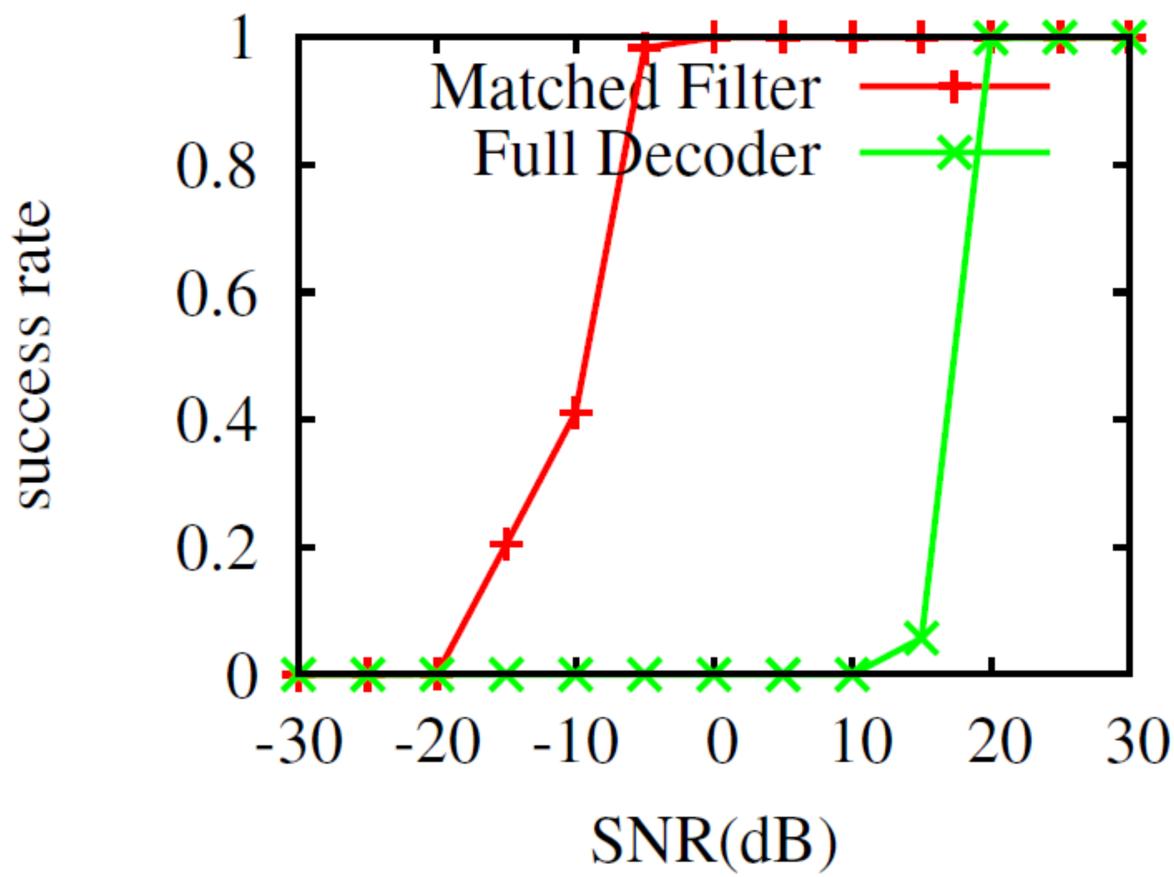
Packet Detection in Hardware



Fast Packet Detection Accuracy

- **Simulation**: detect 1000 data packets *destined to the host* in varying noise using GMSK and the mfilter

- Confirmed in real world (in paper)
- 100% accuracy detecting frames
- <.5% false detections (i.e., falsely claiming an incoming packet)



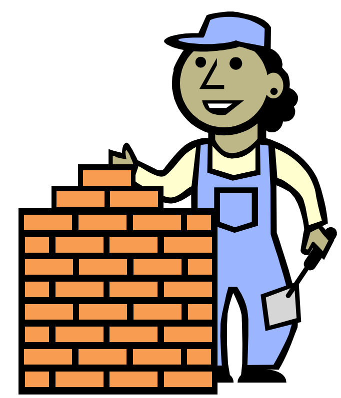
Revisiting the Core MAC Functions

- Building blocks of MAC protocols:
  - carrier sense
  - precision scheduling
  - backoff
  - fast-packet detection
  - dependent packet generation
  - fine-grained radio control

… details in the paper!

Putting it all together…

- Core MAC functions and the *split-functionality API* implemented on GNU Radio & USRP
- “The proof is in the pudding” – we implement two popular MACs
  - 802.11-like and Bluetooth-like protocols
  - shows ability in keeping flexibility
  - used to evaluate total performance gain



CSMA 802.11-like Protocol

- Uses the following core functions:
  - Carrier sense, backoff, fast-packet recognition, and dependent packets
- Compare host based-implementation to *split-functionality* implementation
  - host implements everything in GNU Radio (GPP)
- Cannot interoperate with 802.11 due to limitations of the USRP, but possible with USRP2

802.11-like Protocol Evaluation

- USRP (SDR board) configuration:
  - Target bitrate of 500Kbps
  - Use 2.485GHz, avoid 802.11 interference
  - Ten transfers of 1MB files between pairs of nodes

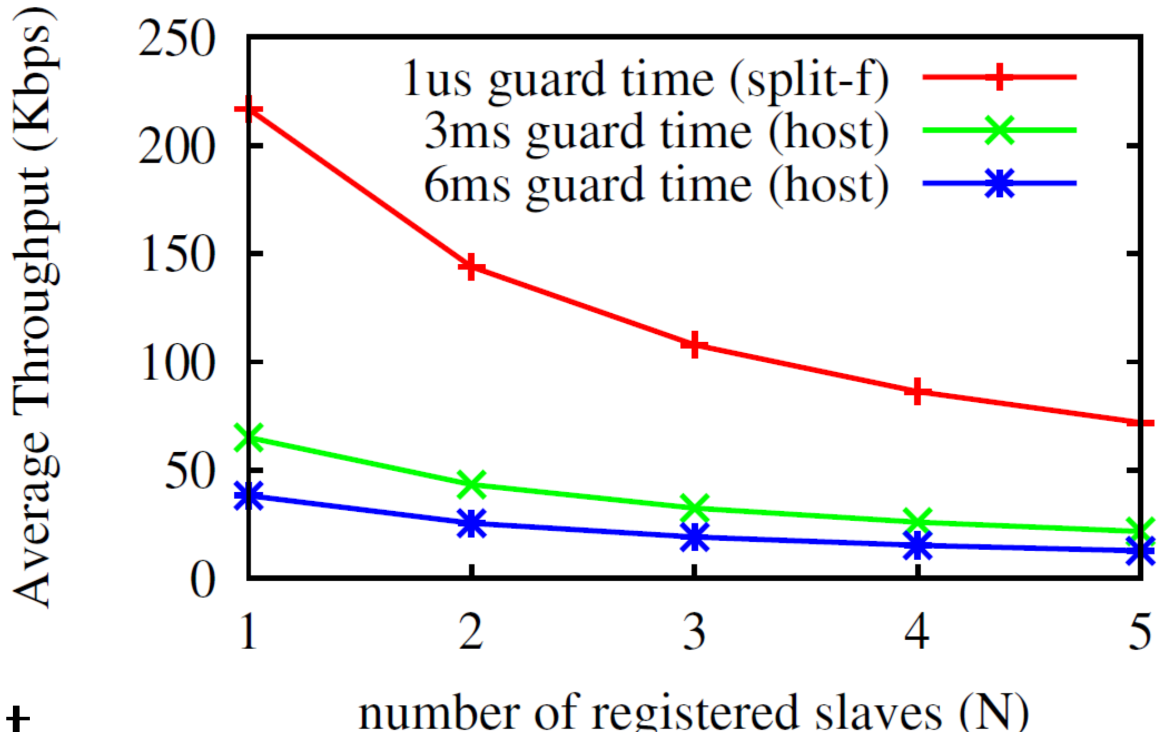
	pairs	Avg (Kbps)	min	max
split – func.	1	408	387	415
host	1	215	190	240
split – func.	2	205	201	210
host	2	112	101	130

TDMA Bluetooth-like Protocol Design

- TDMA-based protocol like Bluetooth:
  - Construct piconet consisting of a master & slaves
  - Slaves synchronize to a master's beacon frame
  - 650μs slot times
- Compare *split-functionality* to host-based again
- Bluetooth-like since the USRP cannot frequency hop at Bluetooth's rate

Bluetooth-like Protocol Evaluation

- USRP: target bitrate of 500Kbps
- Perform ten 100KB file xfers
- Vary number of slaves
- Vary guard time (needed to account for scheduling error)



Conclusions

- The API developed enables a *split-functionality* approach:
  - maintains flexibility & performance
  - aspects applicable to other architectures
- Identified core MAC functions suitable as a first "toolbox" that can be extended
- First to implement high-performance MACs on an extreme SDR such as GNU Radio & USRP