Nano-RK:

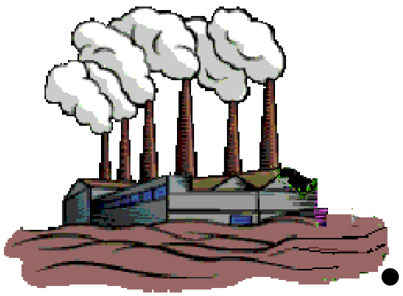
an Energy-aware Resource-centric RTOS for Sensor Networks

Anand Eswaran, Anthony Rowe, Raj Rajkumar
{aeswaran,agr,raj}@ece.cmu.edu
Real-Time and Multimedia Systems Lab
Carnegie Mellon University

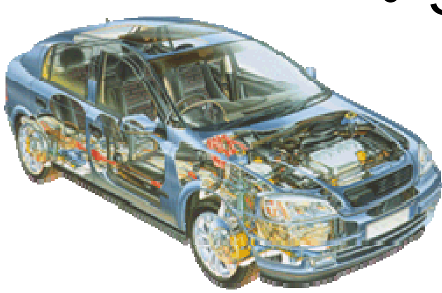
Presented by Anthony Rowe
RTSS 2005

FireFly Sensor Networks

Secure Sensor Networks for Physical Infrastructures



- Develop a secure software platform for infrastructure sensor networks to be deployed in physical structures such as factories, buildings, homes, bridges, campuses, vehicles, ships and planes.
- Provide continual monitoring of operational health and safety,
- Report malfunctions (nearly) instantly, and
- Support mobile nodes and track people.



What is nanoRK?

- **A Real-time Operating System for sensor nodes for use in wireless sensor networks**
  - **Priority-Based Preemptive Multitasking**
  - **Resource Reservations**
  - **Built-in Multi-hop Networking Support**

Related Work

- Tiny OS (Berkeley)
  - Large public following and support
  - Compact
  - Cooperative Multitasking
  - No timing guarantees for tasks
- Mantis OS (University of Colorado)
  - No real-time scheduling
  - No reservation support
- uCos, Emerald, OSEK
  - Still too large
  - Not network centric

NanoRK Motivation

- **Quicker Development Cycles**
  - Traditional OS abstractions allow for quicker learning curves and help abstract away low level details
- **Scaling Technology**
  - Sensor Nodes will become increasingly complex as they begin to manage multiple tasks
  - Local Processing is much cheaper than using the network
- **Resource Reservations (Energy Budget)**
  - Reservations can enforce energy and communication budgets to minimize negative impact on network lifetimes from unintentional errors or malicious behavior on some nodes

NanoRK Motivation

- **Why Priority-based scheduling?**

	Period	Execution Time
Network Radio	Sporadic	10ms
Audio Sensor	200 hz	10us
Light Sensor	166 hz	10us
Smart Camera	1 hz	300ms
Global Positioning	5 hz	10ms

Time-triggered task interleaving can become daunting…

Furthermore, what if a new sensor is added or a period changes?

Goals of a Sensor RTOS (1 of 2)

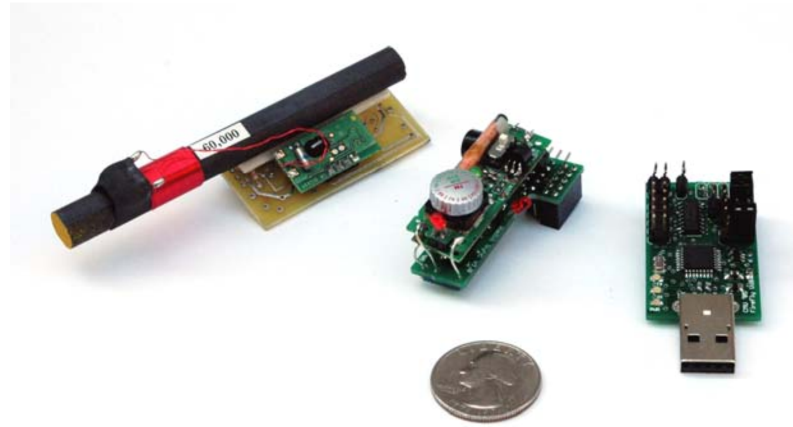
- **Multitasking with Priority-Based Preemption**
  - Respond on a timely basis to critical events
  - Support predictability and schedulability
- **Built-in Network Support**
  - Decrease coding effort to implement custom communication protocols
- **Enforce Energy Usage Limits**
  - Meet Battery Lifetime Requirements

Goals of a Sensor RTOS (2 of 2)

- **Unified Sensor / Network API**
  - Sensors and actuators support in a device driver manner that is abstracted away from the user
- **Small Footprint**
  - Current trends in microcontrollers show larger ROM sizes (64Kb to 128KB) and smaller RAM sizes (2KB to 8KB)

FireFly Hardware

- Atmega128L
- CC2420 (802.15.4)
- Light, Temp, Audio, PIR, Acceleration, Ultrasound
- Global AM band Synchronization



FireFly Synchronization Hardware

NanoRK Resources

Component	Resource
Context Swap Time	45 μS
Mutex Structure Overhead	5 Bytes per Resource
Stack Size Per Task	32→128 bytes (64 bytes by default)
OS Struct Overhead	50 bytes Per Task
Network Overhead	164 bytes
Total Typical Configuration: (8 tasks, 8 mutexes, 4 16 byte network buffers)	**2KB RAM, 10KB ROM**

Current Hardware Platform:

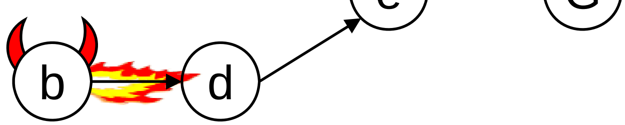
4Mhz Atmega128L with Chipcon CC2420 802.15.4 transceiver

NanoRK Reservations

- CPU
  - Each Task can be given a budget of how long it is allowed to execute per a given period
- Network
  - Per Task Budget on Network Usage
  - Transmit and Receive Packets and/or Bytes
- Sensors / Actuators
  - Number of System Calls to a particular peripheral per a given period

Virtual Energy Reservations

- {CPU, Network, Sensors}
  - Together comprise the total energy usage of the node
- Static Offline Budget Enforcement
  - It is possible to calculate a node lifetime given a certain energy budget

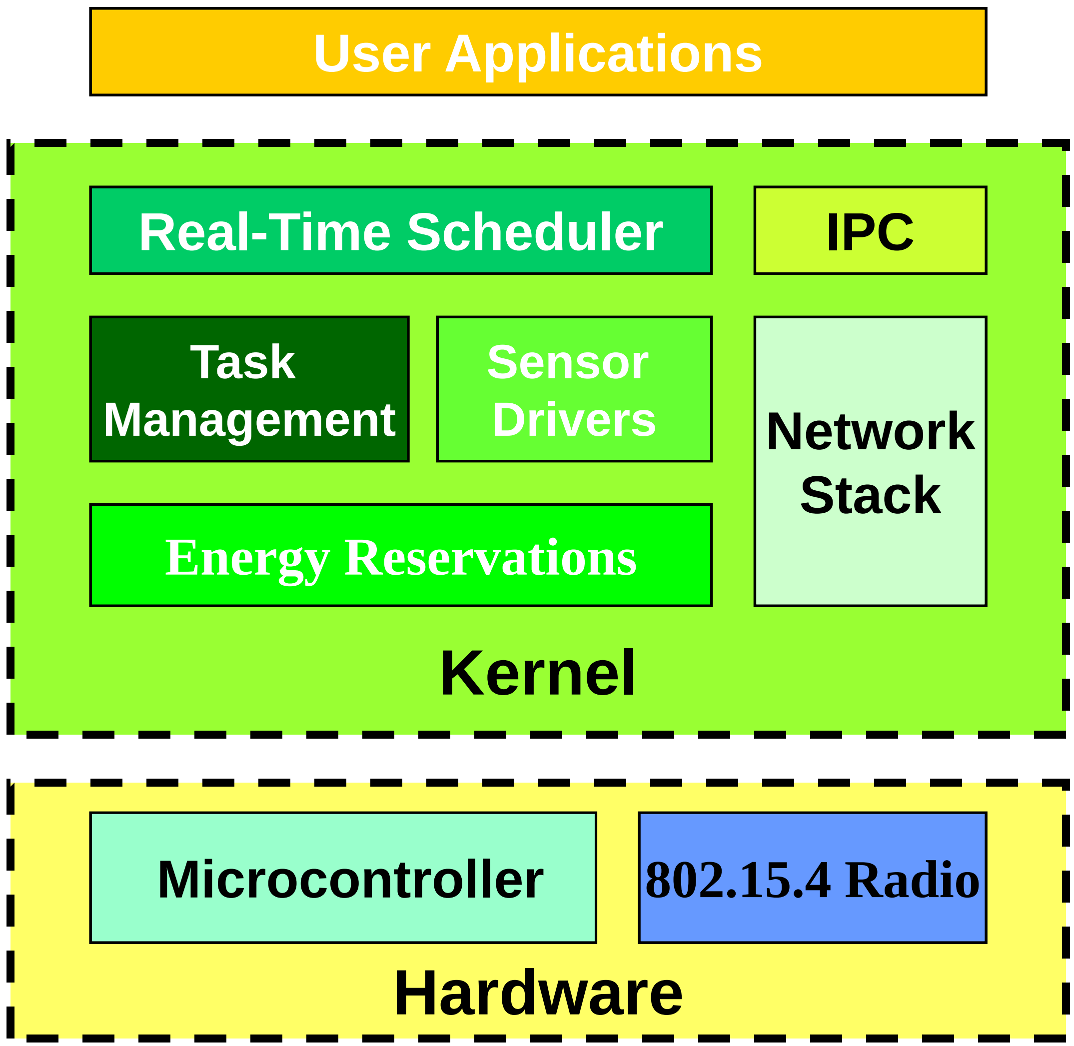


Node	Reserve [TX, RX]	TX Rate	Lifetime w/ out Reserve	Lifetime w/ Reserve
a	[1,2]	1	8 years	8 years
b	X	300	3.5 days	3.5 days
c	[1,2]	1	5 years	5 years
d	[1,2]	1	3.9 days	5 years
e	[1,2]	1	4 days	2.9 years

NanoRK Reservations

- What if a reservation is violated?
- Hard Reservation
  - For CPU reservations, the task is preempted
  - For Sensor/Network reservations, an error is returned to the task making the system call
- Firm Reservations
  - The task is allowed to consume slack resources until they have been depleted

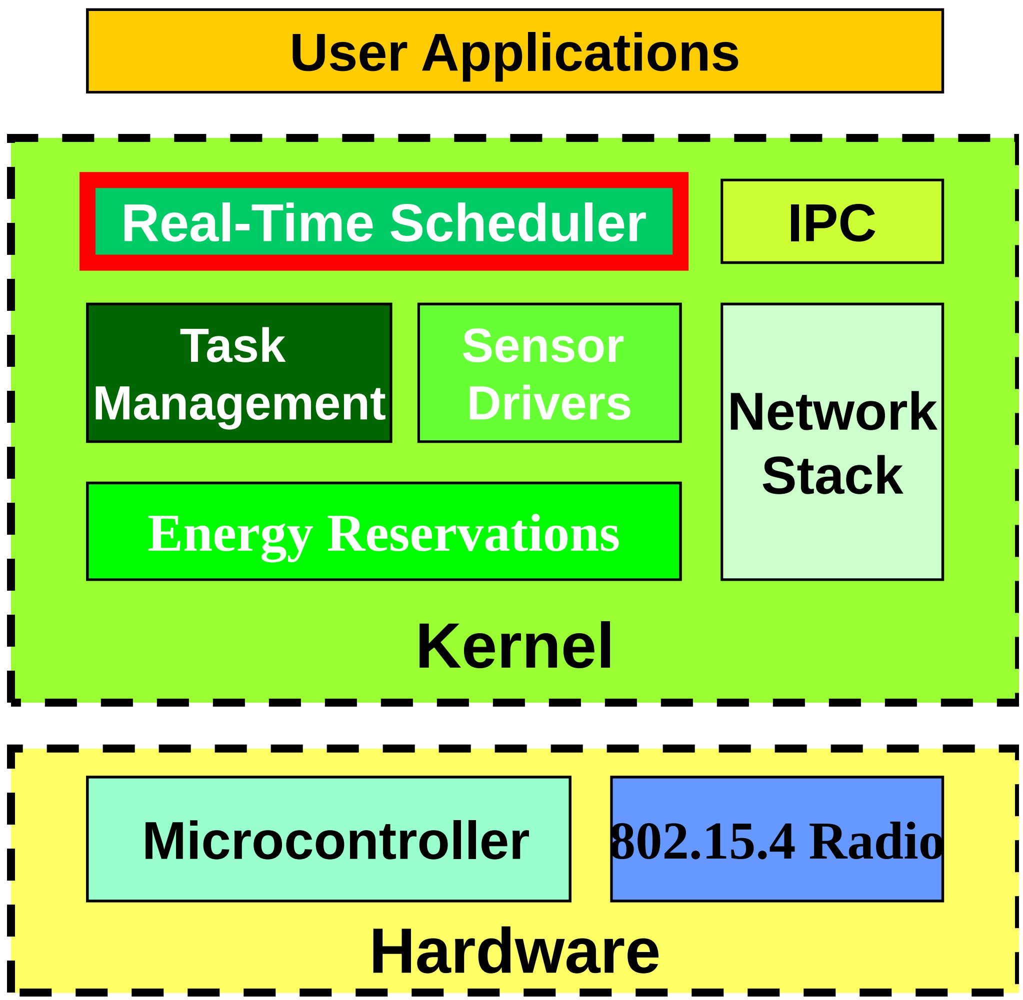
NanoRK Architecture



NanoRK Architecture

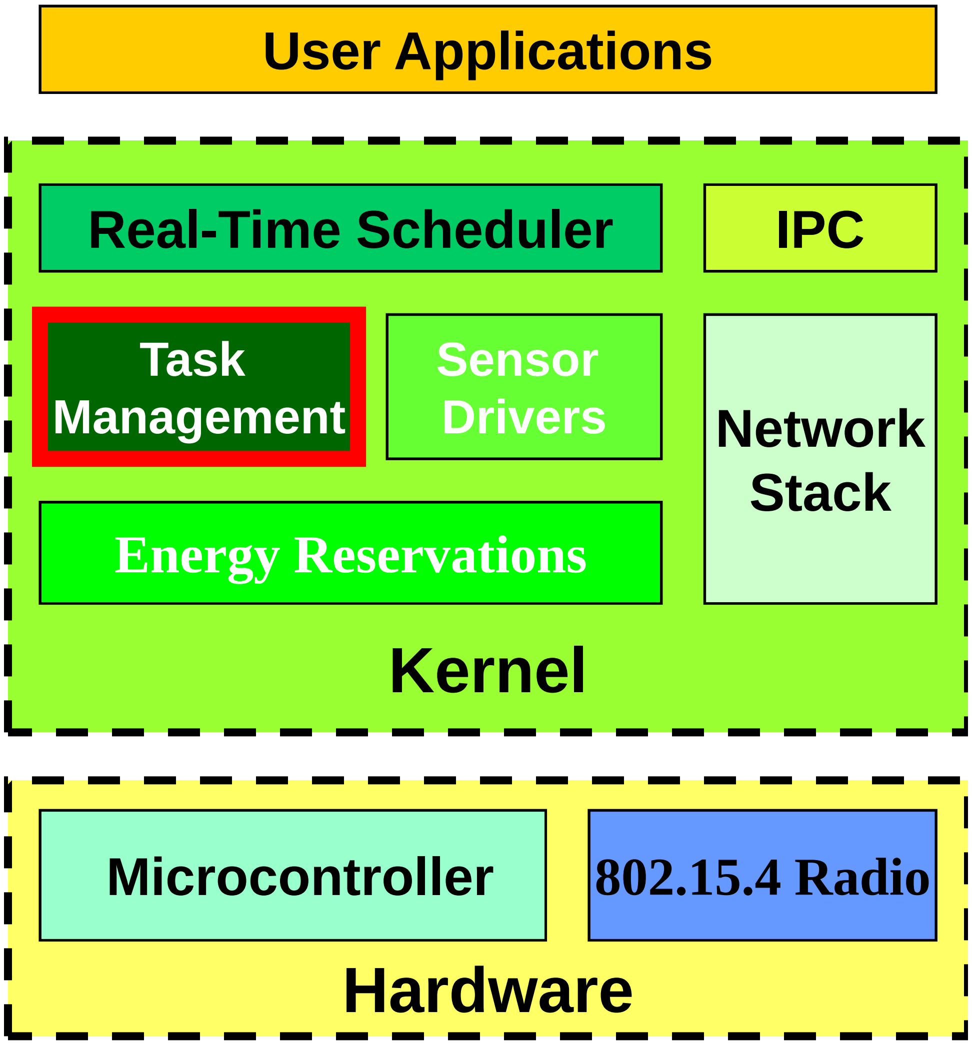
Real-Time Scheduler

- Priority-based Scheduling
- Static Design-time Framework
- Priority Ceiling Protocol



NanoRK Architecture

- **Task Management**
  - Two 32 bit counters
    - { Seconds, Nanoseconds }
  - Periodic Task Switching
    - Triggered by a One Shot Timer or an Event
  - Enables Fine Grained Timing Control



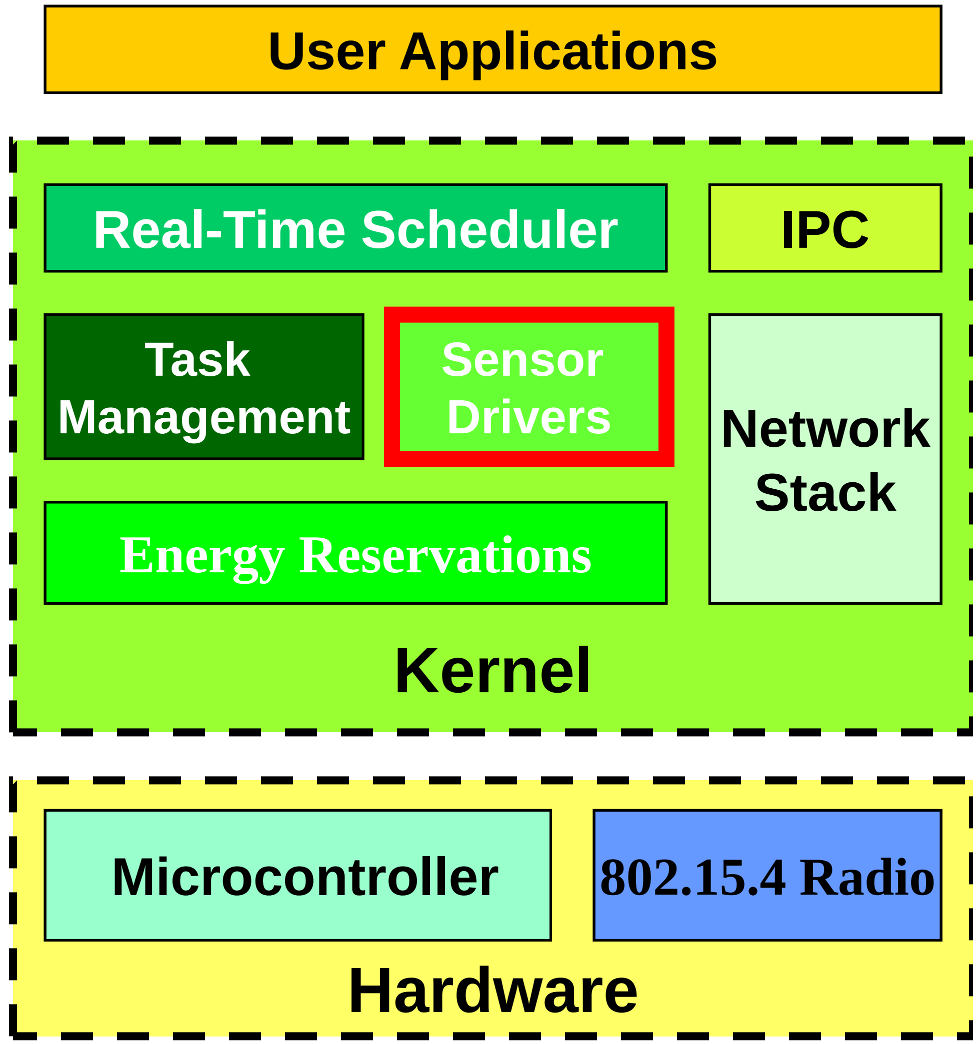
Nano-RK Task Specs

```
nrk_task_type Task1;

Task1.task = Sound_Task;
Task1.Ptos = (void *) &Stack1[STACKSIZE - 1];
Task1.TaskID = 1;
Task1.priority = 3;
Task1.Period = 1000;   // ms
Task1.set_reserve[CPU] = 50;  //ms
Task1.set_reserve[NETWORK_PACKETS] = 3;
Task1.set_reserve[SENSE_ACTUATE] = 12;
nrk_activate_task(Task1);
```

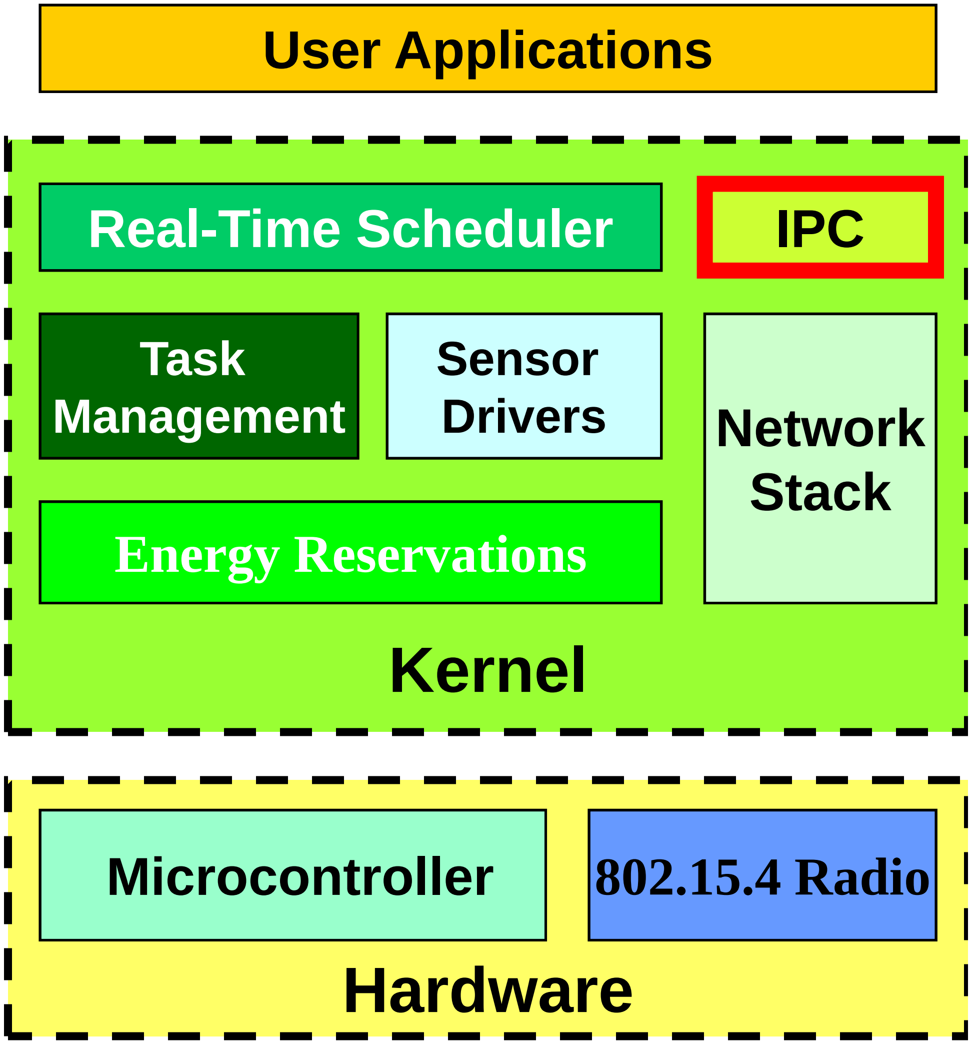
NanoRK Architecture

- Sensor Drivers
  - System Calls
    - Allows for arbitration if two tasks read from a non-atomic sensor
  - Raw ADC Values
    - Temp = 67
  - Real World Values
    - Temp = 24 (°C)



NanoRK Architecture

- **Inter Process Communication**
  - Use Semaphores to arbitrate shared memory communication
  - "Message Boxes"



NanoRK Network Stack

- **Zerocopy Buffering Mechanism**
  - **TX and RX buffers are in application space**
  - **Kernel Manages network data through pointer manipulation into application space**
- *Port* **Abstraction**
  - **A port allows applications to direct their data to individual tasks on each node**
- **Network Task**
  - **Handles Forwarding Packets, Routing, etc.**

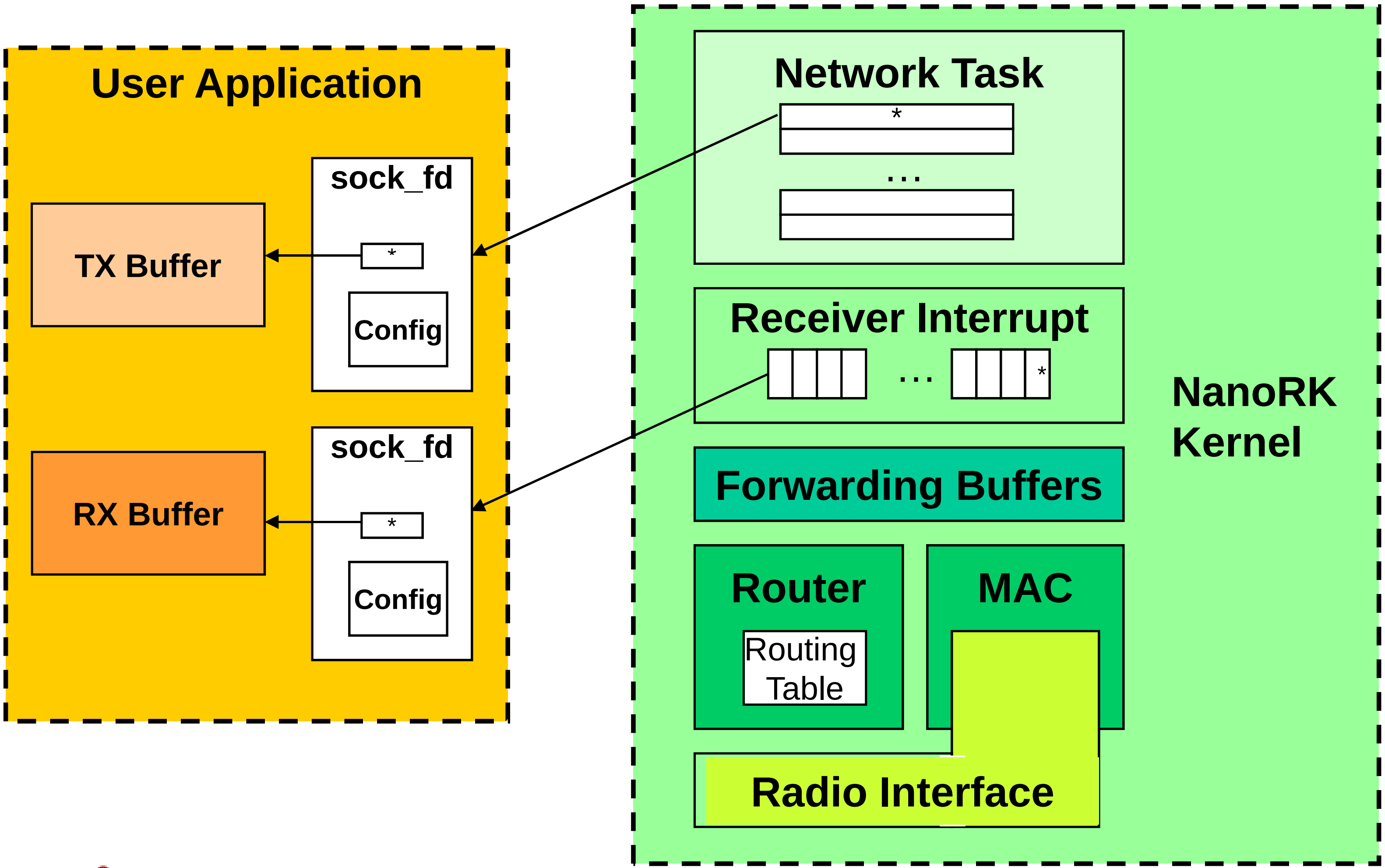
Network Stack

NanoRK Network Stack

- **CSMA MAC Support**
  - **Low Power Listen (LPL) MAC support**
- **TDMA MAC Support**
  - ***RT-Link:* Globally Time Synchronized MAC protocol**
- **Ad-Hoc Routing Support**
  - **Tasks can access low level routing table**
  - **Ex: An AODV or DSR *TASK* can be responsible for establishing and managing routes**

Network Stack

Detailed NanoRK Network Stack

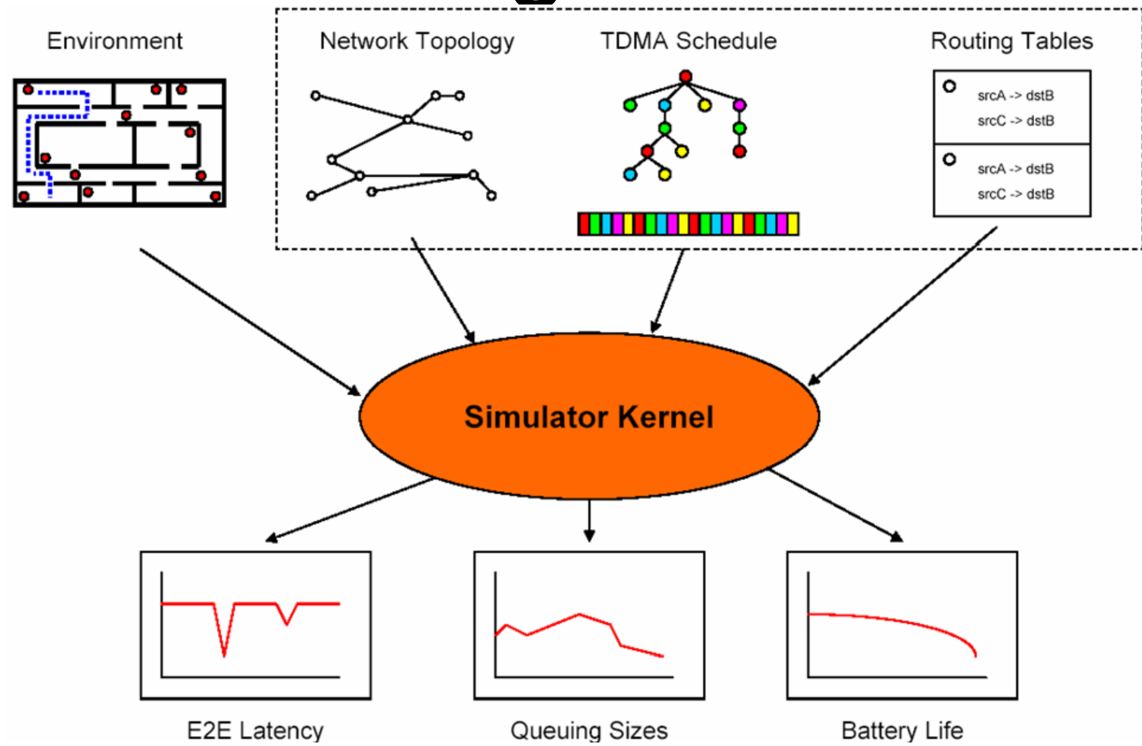


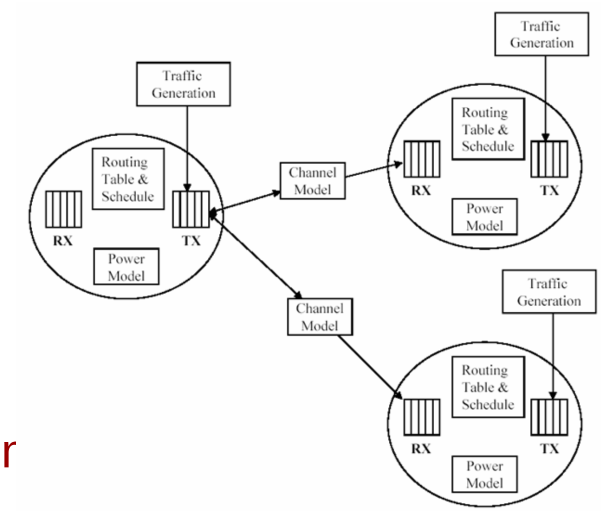
NanoRK Limitations

- No User / Kernel Space Boundary
  - Single Memory Space with no MMU
- Somewhat higher memory overhead
  - Compared to non-multitasking operating systems

Simulation / Modeling Tool

- **Represent Network as a Graph**
  - Easily evaluate different scheduling algorithms
- **Cycle-Accurate Power and Channel Modeling**
  - Analyze how different schedules effect power, throughput, and latency
- ***Hybrid* simulator**
  - Allow virtual expansion of our real network
  - Works with real nodes





Future Work

- A public domain release
- Dynamic upgrades
- Support for multiple microcontrollers
- Family of applications
- Complete set of tools
- Field deployment

Conclusions

- NanoRK: a sensor network Operating System
  - priority-based preemptive multitasking
  - task synchronization
  - multi-hop communications support
- Reservations can enforce system-wide energy and communication usage
- Easy-to-use high level Networking and Sensor Abstractions

Questions?