TinyNET: A Tiny Network Framework for TinyOS

Angelo P. Castellani,
Paolo Casari and
Michele Zorzi

SIGNET Group

February 10, 2009

Outline

1 Outline
2 Motivation
3 TinyNET Overview
- TinyNET Architecture
  - Application layer
  - Network layer
- Inter-component interfaces

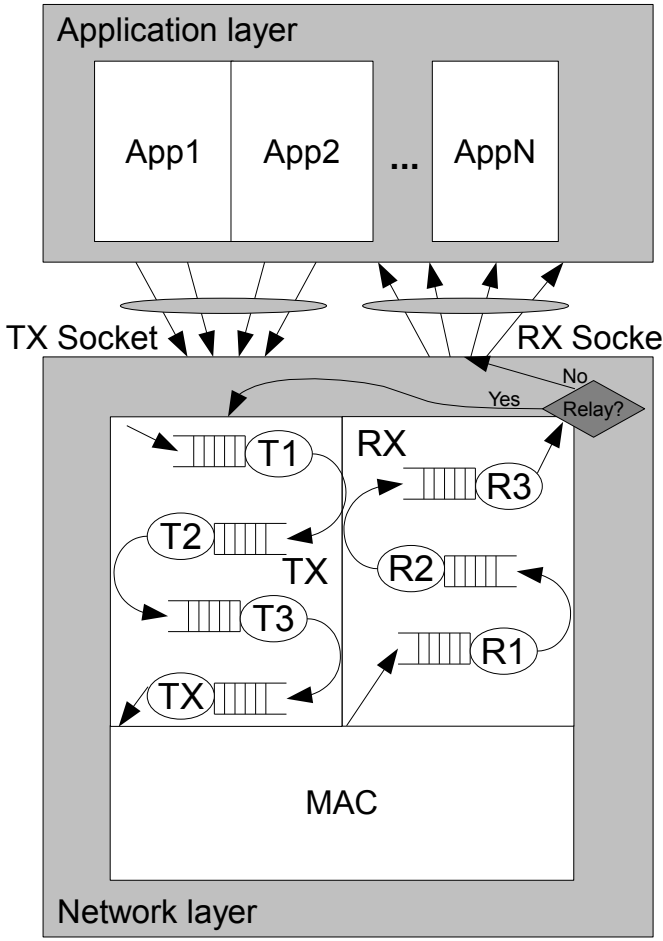
4 Experience with TinyNET
5 Conclusions
6 Future work
7 Questions

Motivation

- TinyOS is a mature operating system designed for Wireless Sensor Networks (WSNs)
- The communication abstraction is the Active Message
  - Active Message datagrams are directly passed on to the RF subsystem
  - Active Message supports only basic service abstraction through AM type (UDP-like)
- There is no network stack implemented
  - Good for simple applications
  - A modularized network stack may represent an advantage when designing complex systems

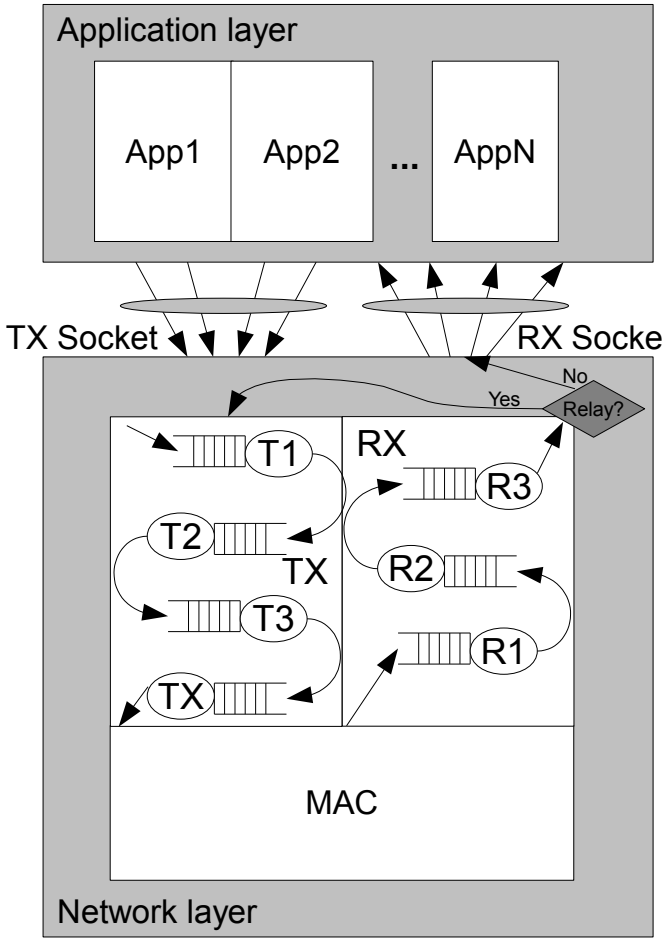
TinyNET Architecture

- TinyNET sits on top of TinyOS (different folder)
  - Independent development
  - Easy porting



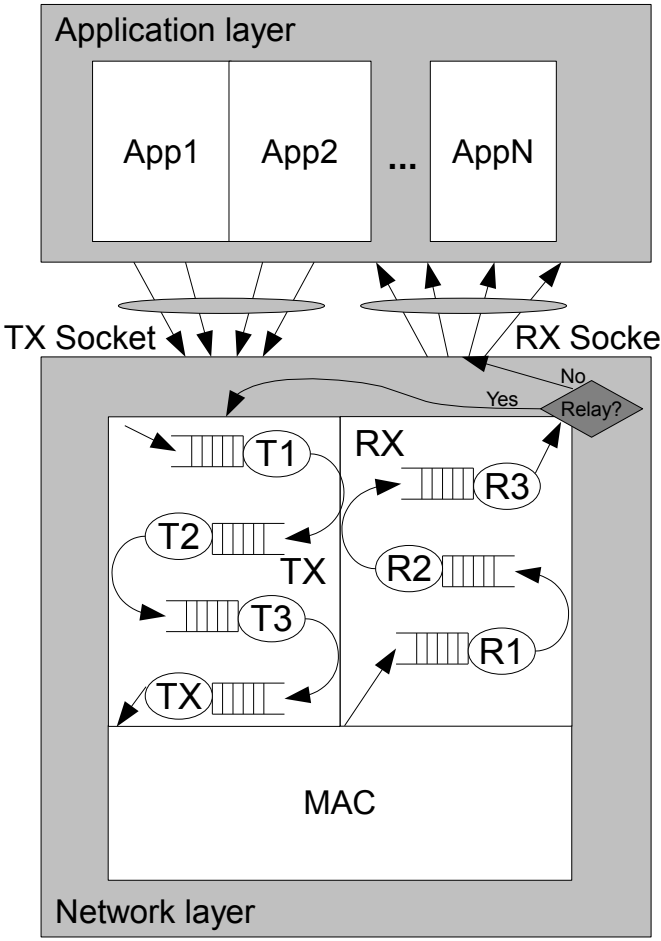
TinyNET Architecture

- TinyNET sits on top of TinyOS (different folder)
  - Independent development
  - Easy porting
- 2-layer design
  - Application layer
  - Network layer



TinyNET Architecture

- TinyNET sits on top of TinyOS (different folder)
  - Independent development
  - Easy porting
- 2-layer design
  - Application layer
  - Network layer
- Packets are scheduled across network modules
  - $AppX \rightarrow T\{1,2,3\} \rightarrow MAC \rightarrow RF$
  - $RF \rightarrow MAC \rightarrow R\{1,2,3\} \rightarrow AppX$



Application layer

- Very similar to the standard TinyOS development entry point
- Every network application is known and identified by an unique `net_app_id`
- Utility interfaces have been built for RF control operations
  - Radio subsystem on/off
  - Channel selection
- Control is centralized allowing
  - Broadcasting of notifications to all network applications
  - Concurrency resolution through locking (planned)

Network layer

- Full access to *every* RX/TX event
  - Useful for routing, localization, security, etc.
- Standard interface (shared by every kind of module)
  - Allows flexible operations (processing order, insertion/deletion of modules)
- Every network module has full access to the packet
  - Every module can rebuild the entire packet from scratch
- The network stack is assembled inside a configuration file

Inter-component interfaces

- **Application layer**
  - Provides custom `TX`/`RX` interface supporting new parameters (priority, TX power, memory swapping)
- **Network layer**
  - `ProcessTXPacket`/`ProcessRXPacket`: Provides full access to internal TinyNET data structures for every packet
  - `Route`: Used for relaying decisions
  - `TXSchedule`: Used by the MAC module to schedule transmissions
- **Hardware abstraction**
  - Provides uniform access to chip-specific radio parameters (channel, TX power, RSSI, LQI, etc.)
- **Legacy application interfaces**
  - Standard interfaces provided to access the framework: `AMSend`, `Receive`, `Packet` and `AMPacket`

Our experiences

- TinyNET has a cost in terms of ROM/RAM memory footprint
  - Fixed $\sim 3.5kB$ ROM overhead
  - $\sim 60B$ RAM + scheduling queues buffers
- We used it to build a complex system
  - multi-hop routing
    - node-to-sink (hop count-based)
    - sink-to-node (reverses node-to-sink routes)
  - IPv6/6LowPAN
  - environmental monitoring application
  - for further informations look at IWCMC '09
- $\longrightarrow$ Very fast development process
- $\longrightarrow$ Debugging and testing focused on single modules

Conclusions

- TinyNET offers a change of perspective
  - Divide-and-conquer system design
  - Quick start
  - Easy system integration
  - Encourages community code contribution
- Helps build a proper modular system
  - Simpler subproblems handled separately
  - Reduced effort in code inspection
  - Single modules can be swapped/added or deleted

Future work

- Flexible wiring of any number network modules
- RAM and ROM requirements optimization
- Radio resource locking subsystem
- Uniform access interfaces to IPv6/6LowPAN

Your turn... Questions?