

Ivy Slackers Multihop Routing



Barbara Hohlt, Lance Doherty,
Kris Pister

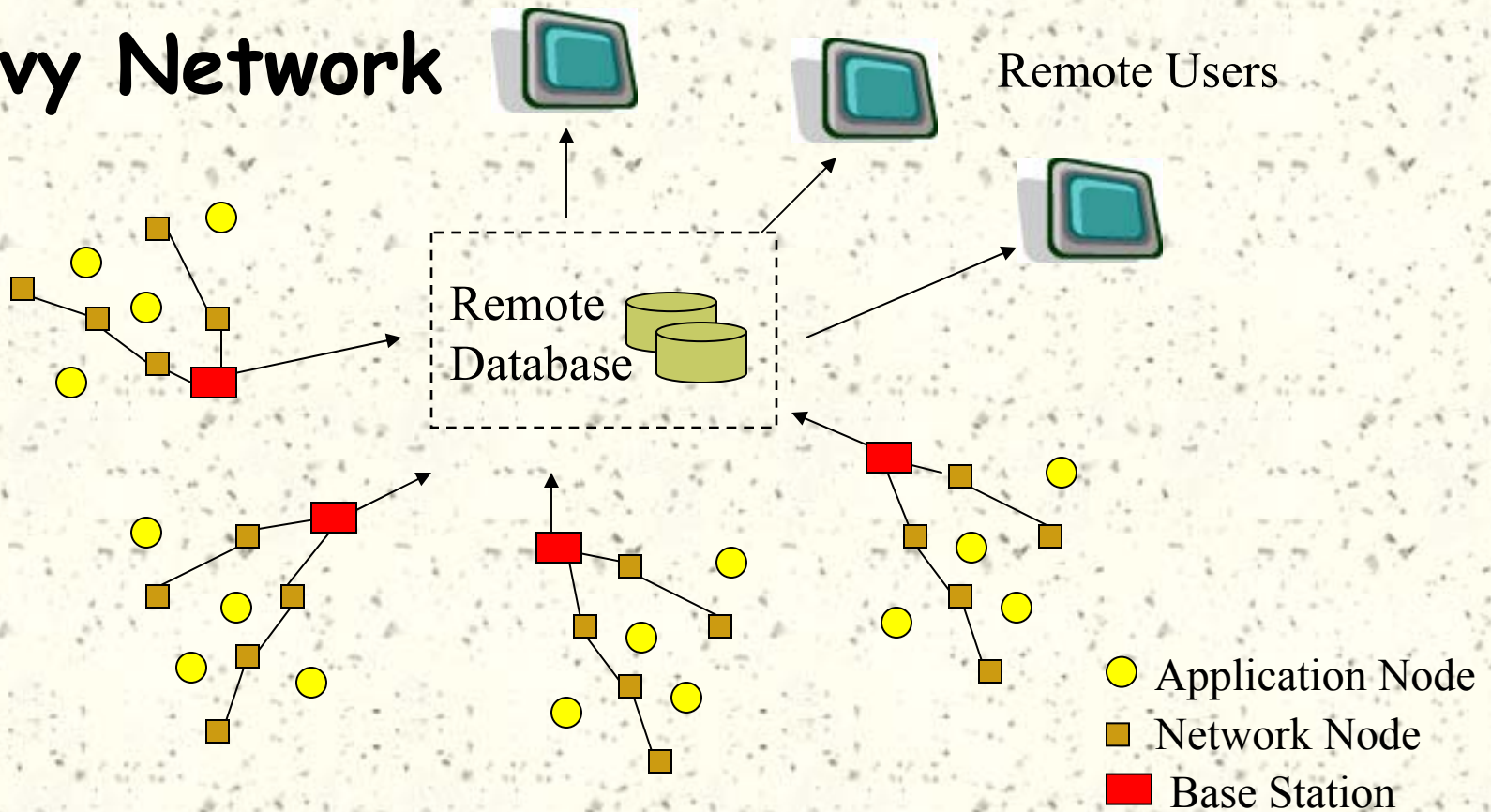
Nest Retreat
January 2003

What is Slackers ?

- # Power scheduling multihop routing algorithm for Ivy sensor networks
- # Primary goal is significant reduction of the duty cycle per node
- # Ivy network nodes are sparsely positioned
- # Slowly varying periodic data demands

<http://www-bsac.eecs.berkeley.edu/projects/ivy>

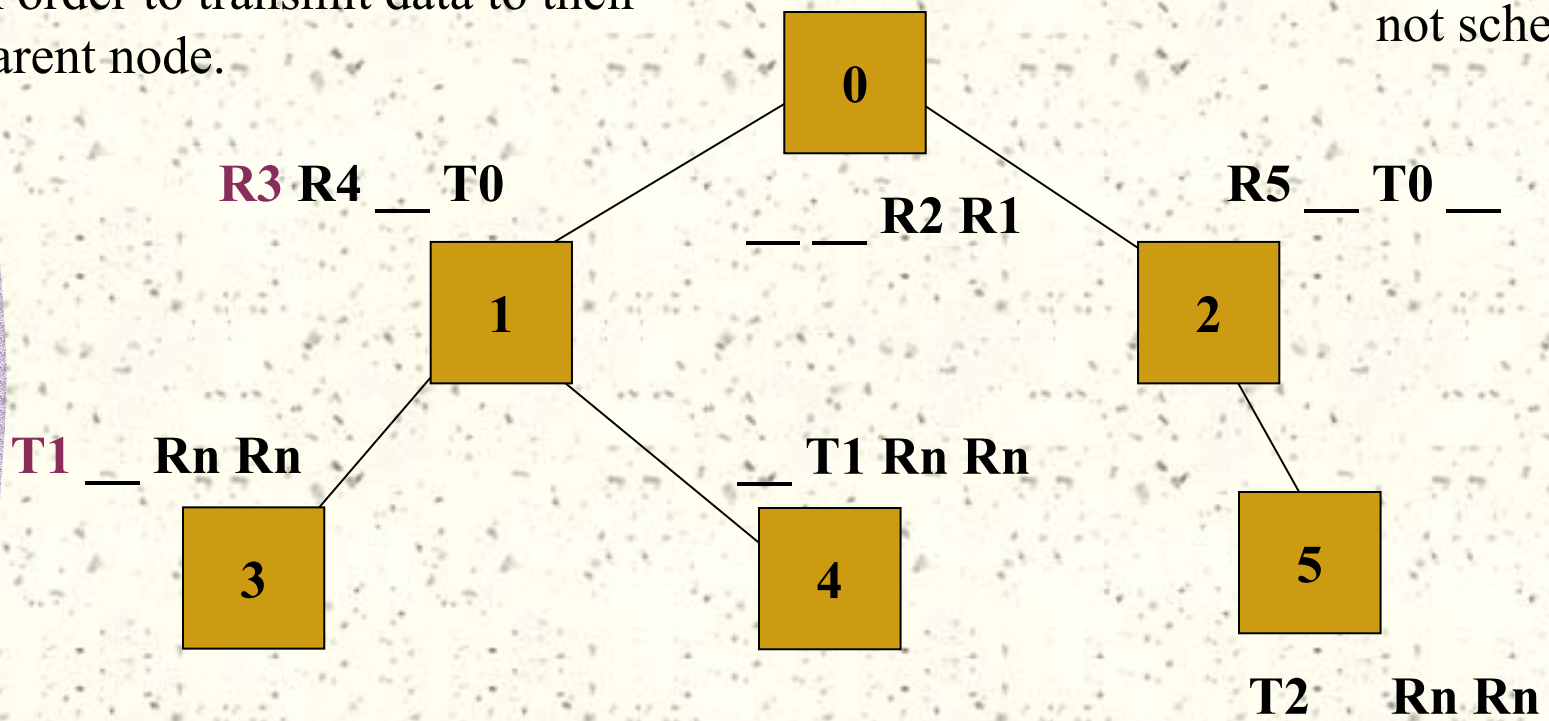
Ivy Network



Power Scheduling

Network nodes reserve time slots in order to transmit data to their parent node.

Sensor nodes can power down during the time slots that are not scheduled.



Slot States

- # Transmit
- # Receive
- # Advertise
- # Transmit Pending
- # Receive Pending
- # Idle

The Slacker Algorithm

- # Initialization
- # General Algorithm
- # Supply and Demand

Initialization & advertisements

Network nodes send advertisements once per cycle. These messages say:

- I am node #
- I am # hops from the base station
- The current time slot is #
- Transmit to me during time slot #

Initialization sequence

- # Listen to advertisements for at least one cycle
- # After one cycle select a parent
- # Synchronize to parent
- # pAS is the probability of accepting an advertisement for a slot
- # Send a reservation request for the advertised slot during the advertised time slot

Initialization sequence cont'...

- # During the same time slot the parent node sends a confirmation acknowledgment
- # If an acknowledgment is received, initialization is over
- # Else listen for more advertisements

Advertisement Protocol

Node #1

Node #2

Slot State

Slot State

Receive Pending

Idle (Listen 1 cycle)

Receive

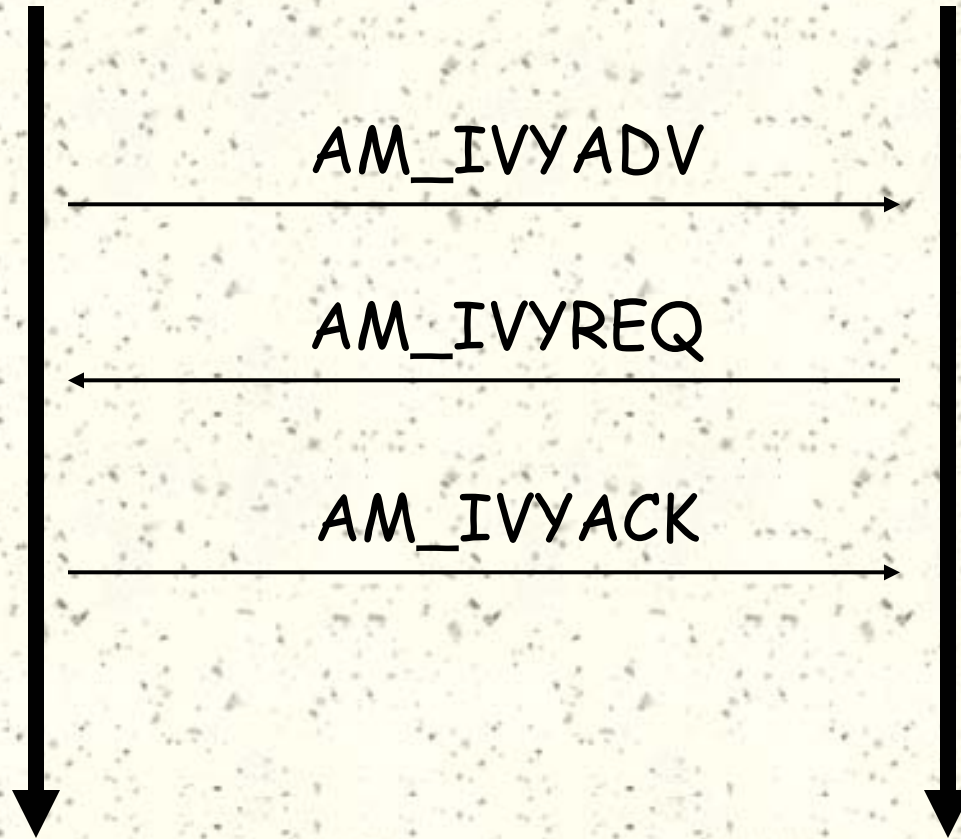
Transmit Pending

Transmit

AM_IVYADV

AM_IVYREQ

AM_IVYACK



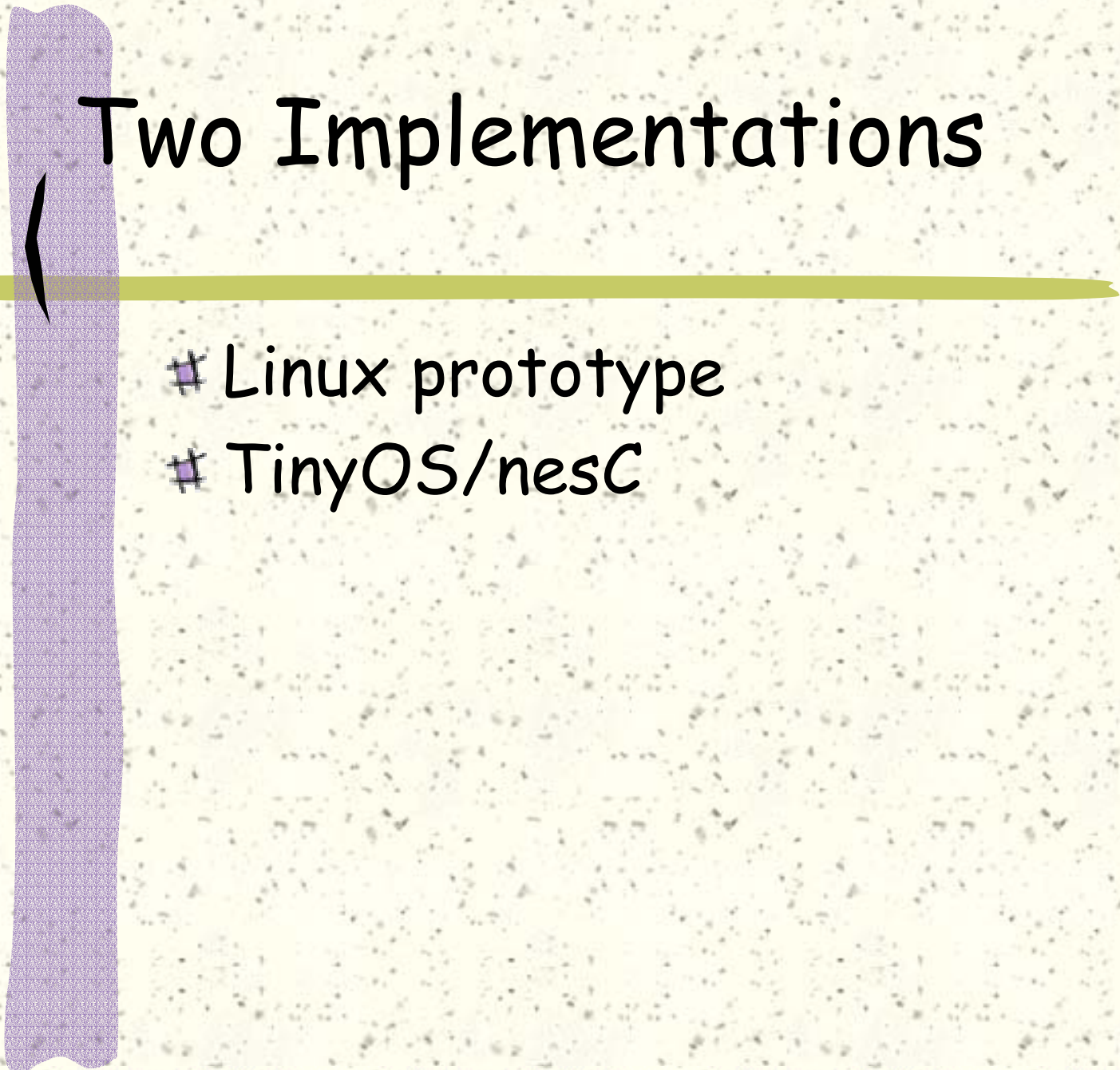
The Algorithm

1. Init - listen 1 cycle for parent
2. For current cycle, pick an advertisement slot and a reservation slot randomly from idle slots
3. Loop through slots in cycle
 1. Receive message from downstream
 2. Transmit message upstream
 3. Broadcast an Advertisement
 4. Listen for a reservation request
 5. Send reservation request
 6. Idle
4. Repeat 2 - 4

Supply and Demand

1. Start with demand = 1, supply = 0
2. If supply < demand
 1. Listen for advertisements
 2. Increment supply
3. If supply \geq demand
 1. Send advertisements
 2. Increment demand
4. Repeat 2-4

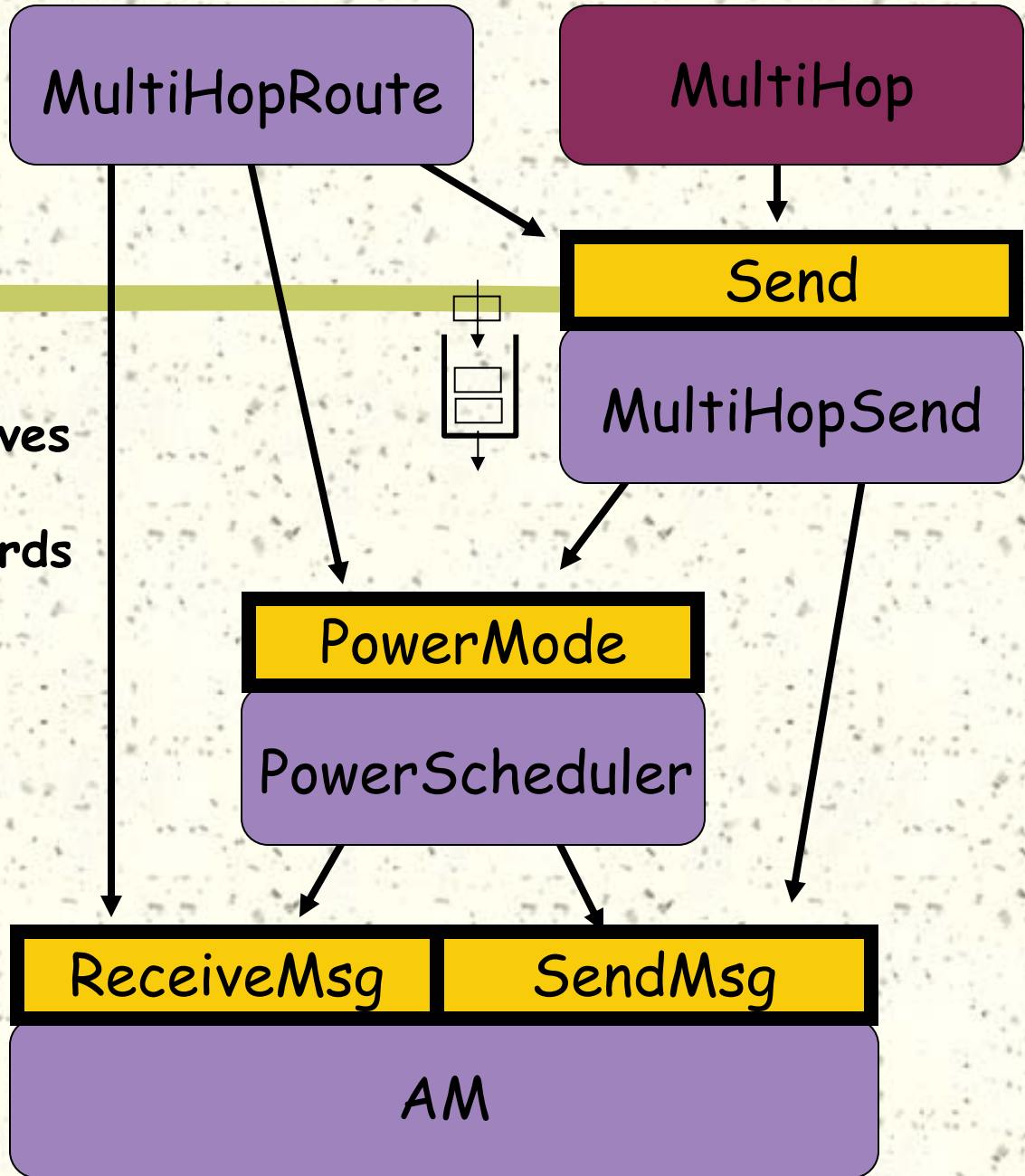
Two Implementations



- # Linux prototype
- # TinyOS/nesc

nesC Program Design

- MultiHopRoute receives messages
- MultiHopSend forwards messages
- PowerScheduler sends/receives advertisements



0	R	R	T	RP	R	RP	T	R	T	T	T	R	A	T	I
1	T	I	R	I	T	I	I	I	RP	RP	A	T	R	TP	I
2	A	T	RP	I	I	I	RP	T	R	I	I	I	I	I	I
3	I	I	RP	I	A	I	I	I	T	I	I	RP	I	I	I
4	I	I	T	RP	I	I	A	I	I	I	I	RP	I	I	I
5	I	RP	I	I	I	A	I	I	I	I	I	RP	T	I	I

Experiment: Sample schedules from a **6 sensor node** network with a demand of 1. The number of **slots per cycle** is 15.

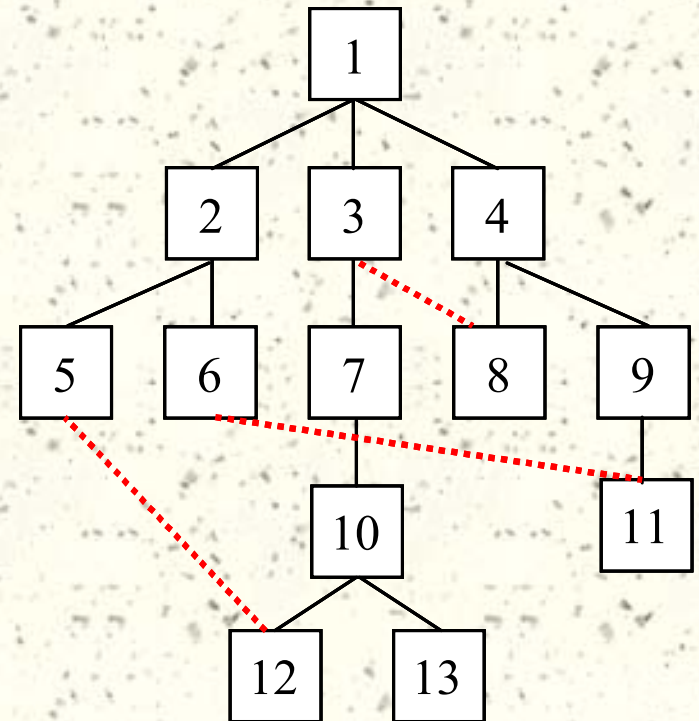
Transmit
 Receive
 Advertise
 Transmit Pending
 Receive Pending
 Idle

Demo

- # **Red Toggle** - init phase
- # **Green Toggle** - synched with parent
- # **Yellow On** - send advertisement
- # **Red On** - send confirmation
acknowledgment

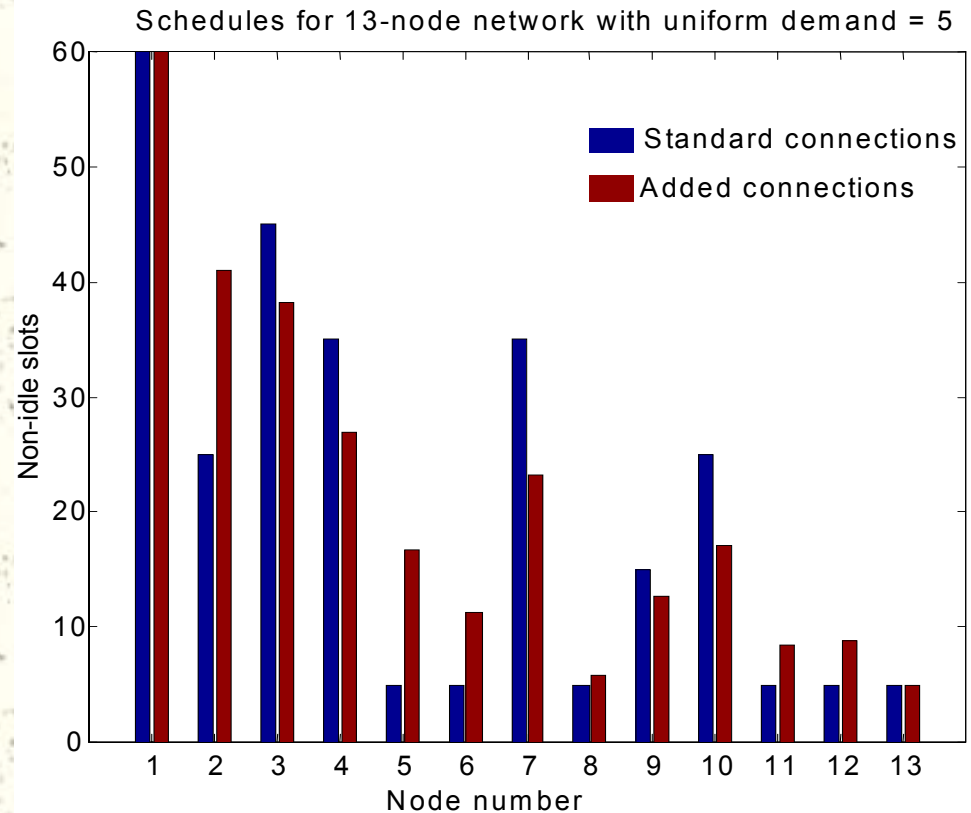
Simulation of more realistic networks

Connectivity of a **13-node** network with. Solid lines represent the standard connection set while the dotted lines are added for the augmented connection set. Each network node has several **application nodes** (not shown) communicating to each of the network nodes.



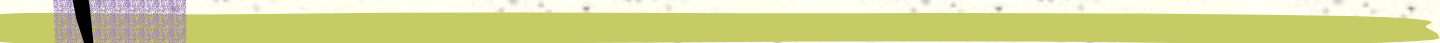

Duty Cycle

A bar chart showing the number of slots occupied by either transmission or reception following completion of the schedule of the 13-node network with a demand of 5 at each node. The number of slots per cycle is 125 and $pAS = 0.5$. The results are averaged over a set of 100 trials.



Future Work

- # Application node code (fluctuating supply/demand)
- # Probability
- # Time synchronization
- # Investigate the optimal local and global sleep cycles
- # Compare to simulation experiments



END