



# ***A Power-Aware Routing Algorithm***

Erol Gelenbe, Ricardo Lent

School of Electrical Engineering and Computer Science  
University of Central Florida

July 22, 2003

- Routing and forwarding
- Many routing proposals require periodic exchange of routing information
- Cognitive Packet Networks (CPN). A new routing approach:
  - Emphasis on learning
  - Employ focalized information collection
  - Users and network needs define a target QoS for each connection

# *Mobile Ad Hoc Networks*

---

- Initially intended for military and disaster recovery applications. Today, interesting for civilian applications
- Problem: Unreliable system
  - Dynamic topology (mobility and wireless connections)
  - Limited resources (e.g. battery lifetime)
  - Shared access (hidden node problem, exposed terminal problem)
- Previous proposals flood the network to find destinations (DSR, AODV, ZRP, etc.)

# Ad Hoc Cognitive Packet Networks

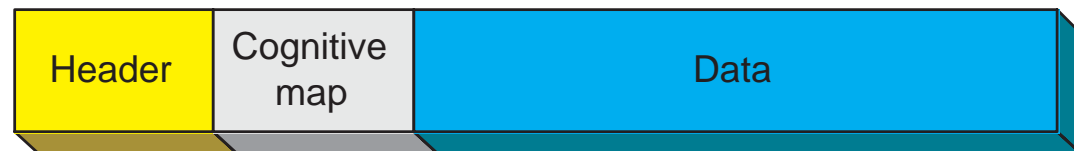
AHCPN are packet switching networks that implement adaptive, QoS-driven routing

*Data Plane:*

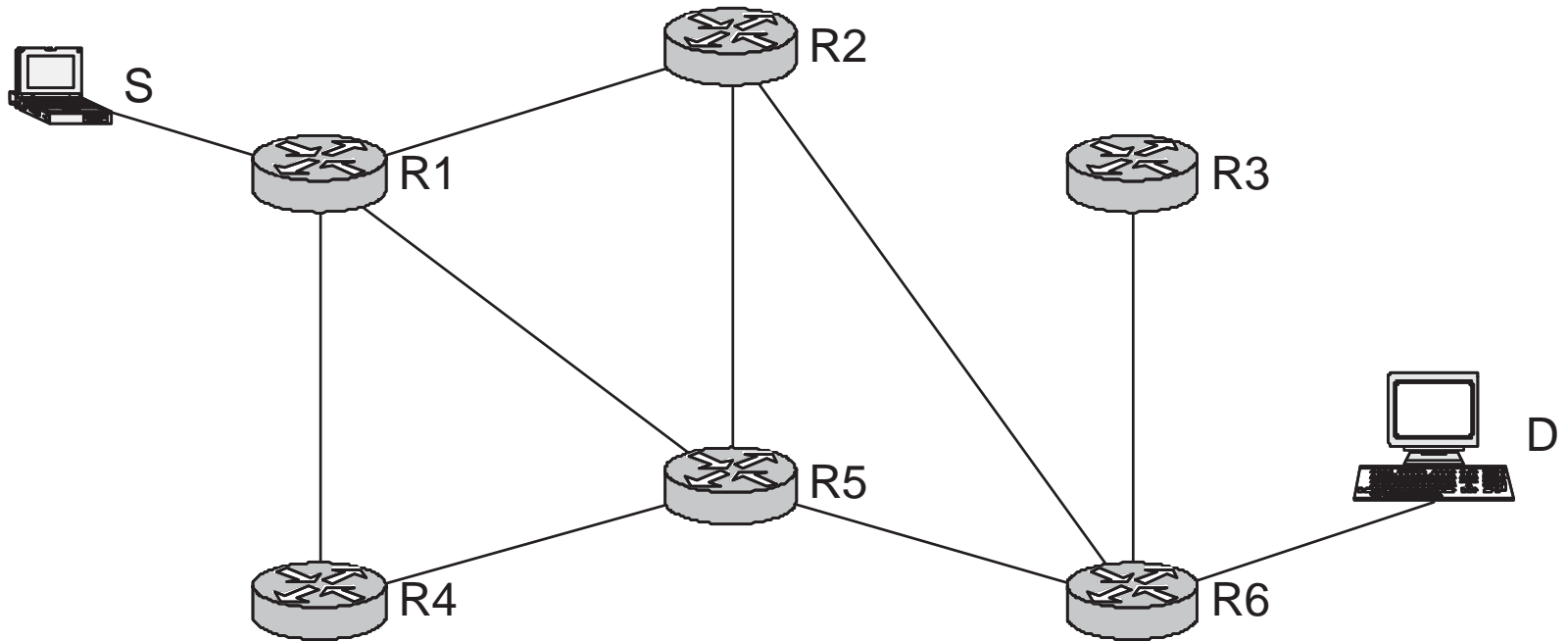
- **DUMB Packets** carry payload (source routing)

*Control Plane:*

- **SMART or Cognitive Packets** use a form of flooding or a RNN/RL algorithm to take routing decisions
- **ACK Packets** distribute routing information (source routing)

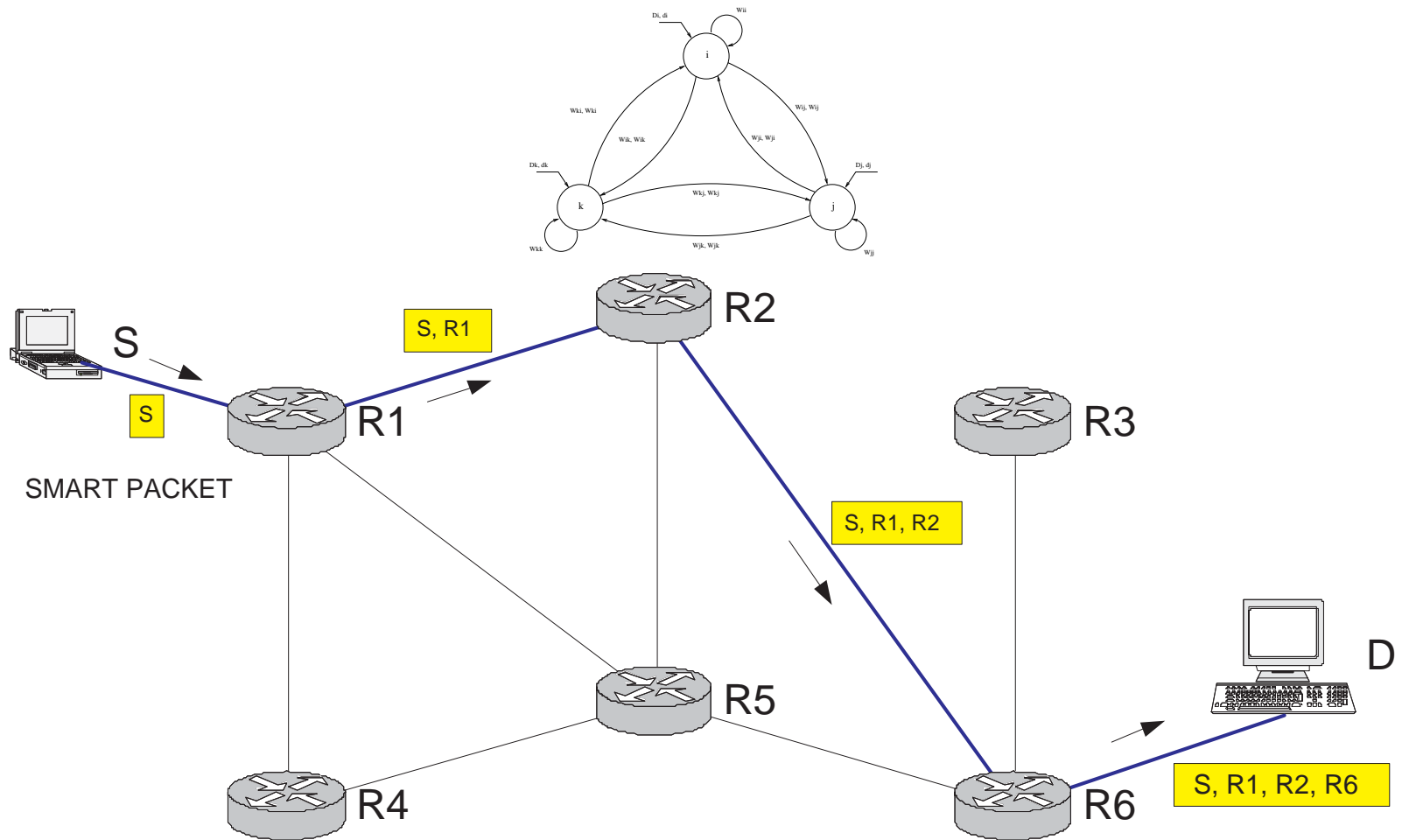


# CPN Routing



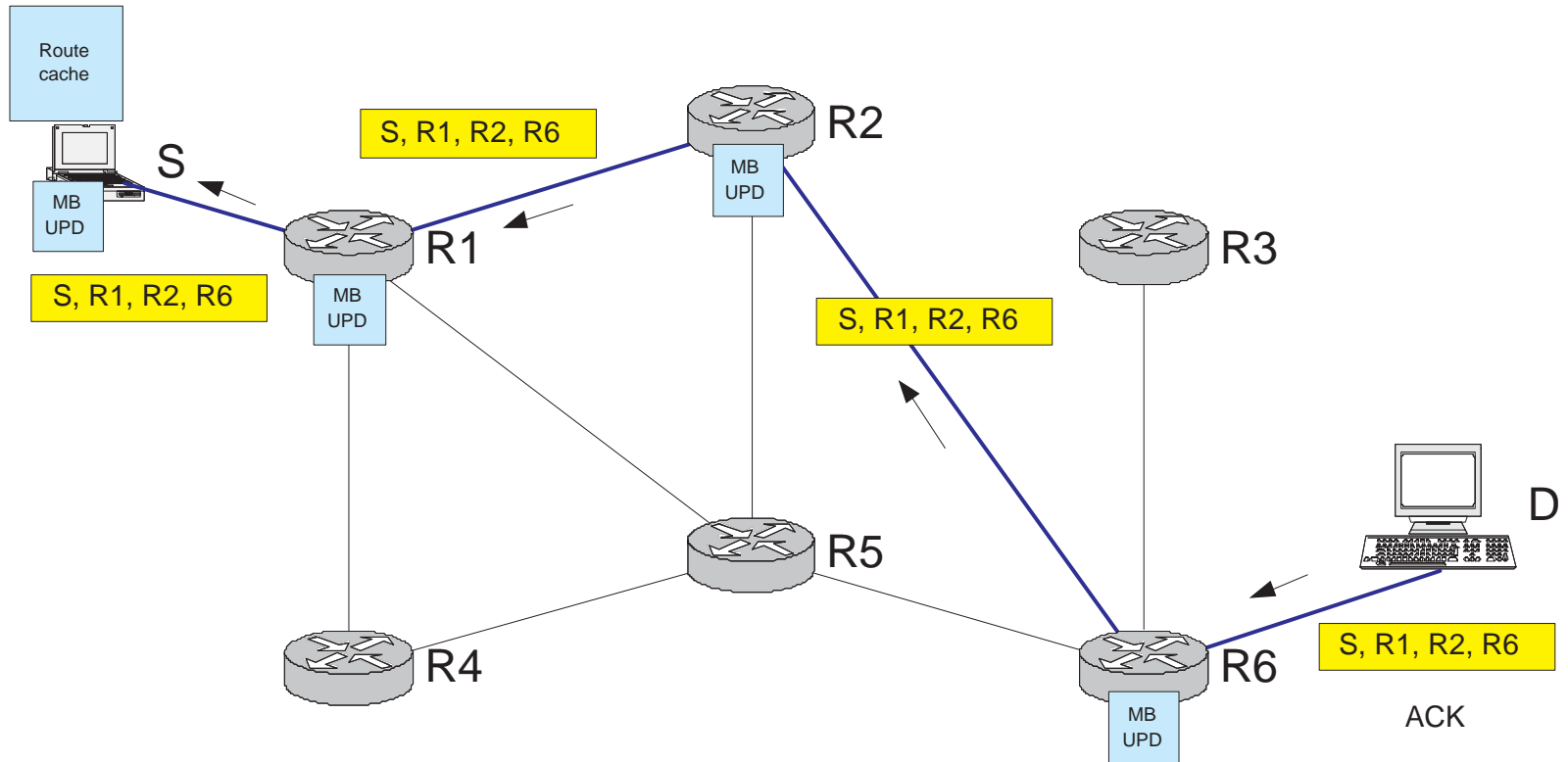
Routing goal = minimization of round-trip delay

# CPN Routing (cont'd)



- *Cognitive map* stores addresses and timestamps
- Potential loops are removed

# CPN Routing (cont'd)



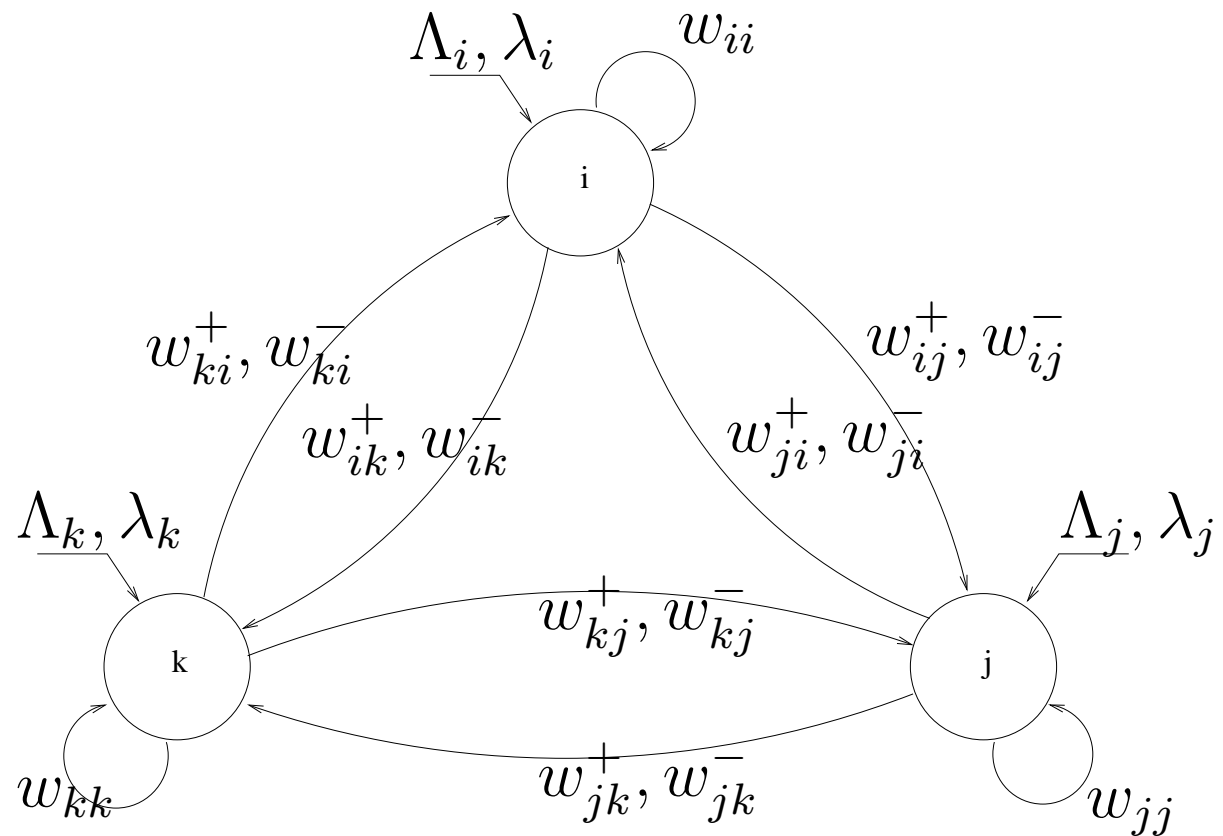
- ACK updates *Mailboxes* (average RTT to destination)
- ACK creates or updates route in *route cache*
- DP uses source routing from information in *route cache*
- An SP ratio ensures route maintenance

# Random Neural Networks\*

- Mathematical structure similar to a queuing network
- Fully connected network of  $N$  neurons
- Neuron  $i$  is characterized by a potential level  $q_i$  (*excitation level*)
- Neurons fire unit-amplitude spikes
- Spikes create excitatory or inhibitory signals to other neurons
- Always converge to steady state in finite time

\* *Spiked Random Neural Networks*, E. Gelenbe, Neural Computation 93

# RNN example



$w_{ij}^+ = r(i)p^+(i, j)$  and  $w_{ij}^- = r(i)p^-(i, j)$

are the rates at which neuron  $i$  sends excitation and inhibition spikes to neuron  $j$

(when  $i$  is excited) respectively

# Steady-state Equations

Probability that the neuron is excited:

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(i)}$$

$$\lambda^+(i) = \sum_j q_j w_{ji}^+ + \Lambda_i, \quad \lambda^-(i) = \sum_j q_j w_{ji}^- + \lambda_i$$

In CPN, the largest  $q_i$  selects the next-hop for the packet.

# Reinforcement Learning (RL)

## Definitions:

- Goal to achieve  $G$  (example  $G = D$ )
- Reward function  $R = G^{-1}$
- A threshold function  $T_l$ :

$$T_l = \beta T_{l-1} + (1 - \beta) R_l \quad ; 0 < \beta < 1$$

Learning comes from past experience:

If  $T_{l+1} \leq R_{l+1}$

- $w_{ji}^+ \leftarrow w_{ji}^+ + R_l$ , *(reward)*

- $w_{ji}^- \leftarrow w_{ji}^- + \frac{R_l}{n-2}$ ,  $k \neq j$ ,

else

- $w_{ji}^+ \leftarrow w_{ji}^+ + \frac{R_l}{n-2}$ ,  $k \neq j$ , *(punishment)*

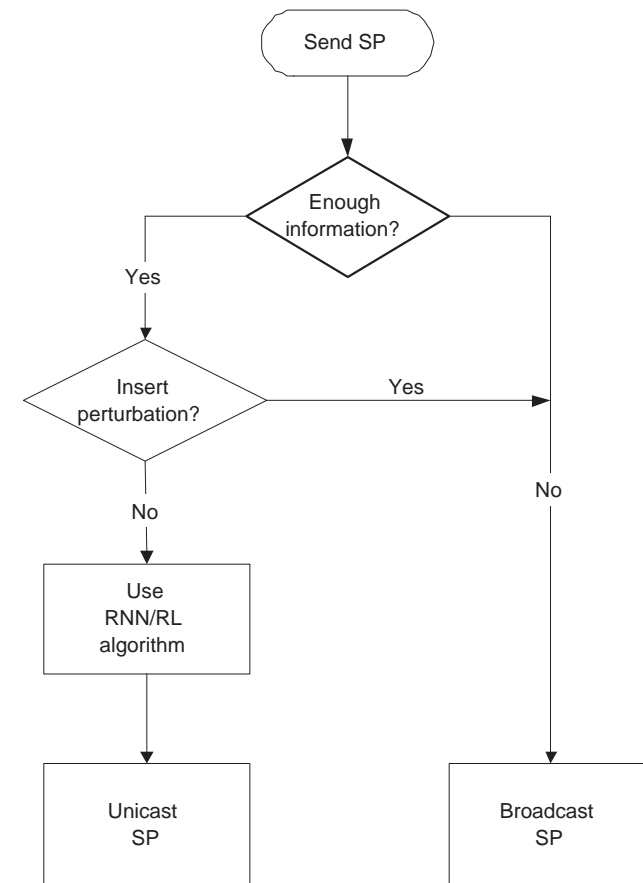
- $w_{ji}^- \leftarrow w_{ji}^- + R_l$ .

$R_l$  is calculated from information stored in Mailbox.

Values are re-normalized after this.

# Ad Hoc CPN (recap)

- Layer-2/3 address resolution
  - Listen to transmissions
  - Assign neighbors a time-to-live limit
- SP may use broadcasts
  - Packets are tagged with a packet id
  - Whenever regular RNN/RL is not possible
  - For route maintenance (*broadcast selection ratio*)



Assume that an SP takes a path  $(n_1, n_2, \dots, n_d)$ , then at any node  $n_i$ :

$$G_i = P_p(n_i, n_d)D(n_i, n_d) + [1 - P_p(n_i, n_d)](T_o + G_i)$$

where

- $G_i$  = routing goal ( $R = G^{-1}$ )
- $P_p(n_i, n_d)$  = path availability from  $n_i$  to  $n_d$   
(probability to find a path available for routing)
- $D(n_i, n_d)$  = round-trip delay from  $n_i$  to  $n_d$
- $T_o$  = timeout constant (penalty)

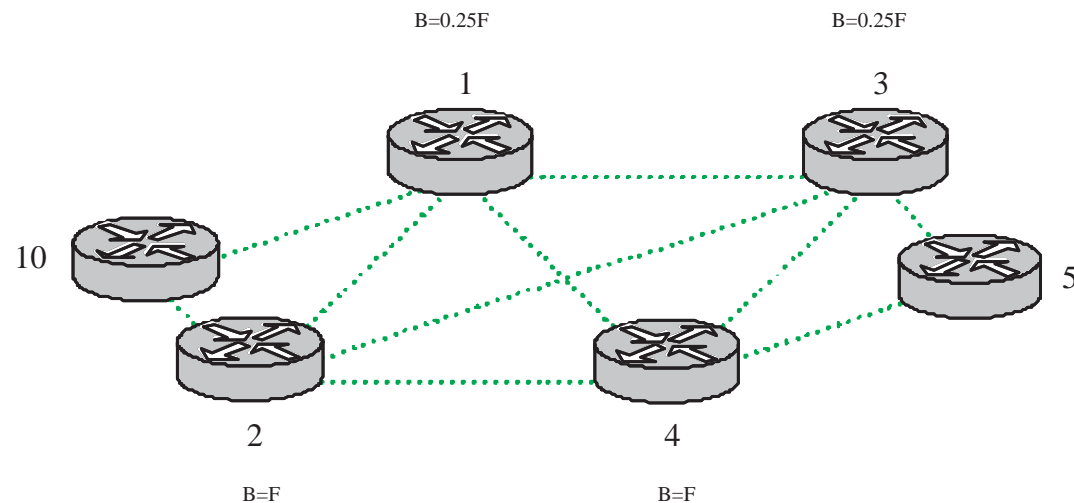
$$P_p(n_i, n_d) = \prod_{j=i}^{d-1} P_n(n_{j+1}) P_l(n_j, n_{j+1})$$

$$P_n(n_i) = \frac{B_i}{B_m}$$

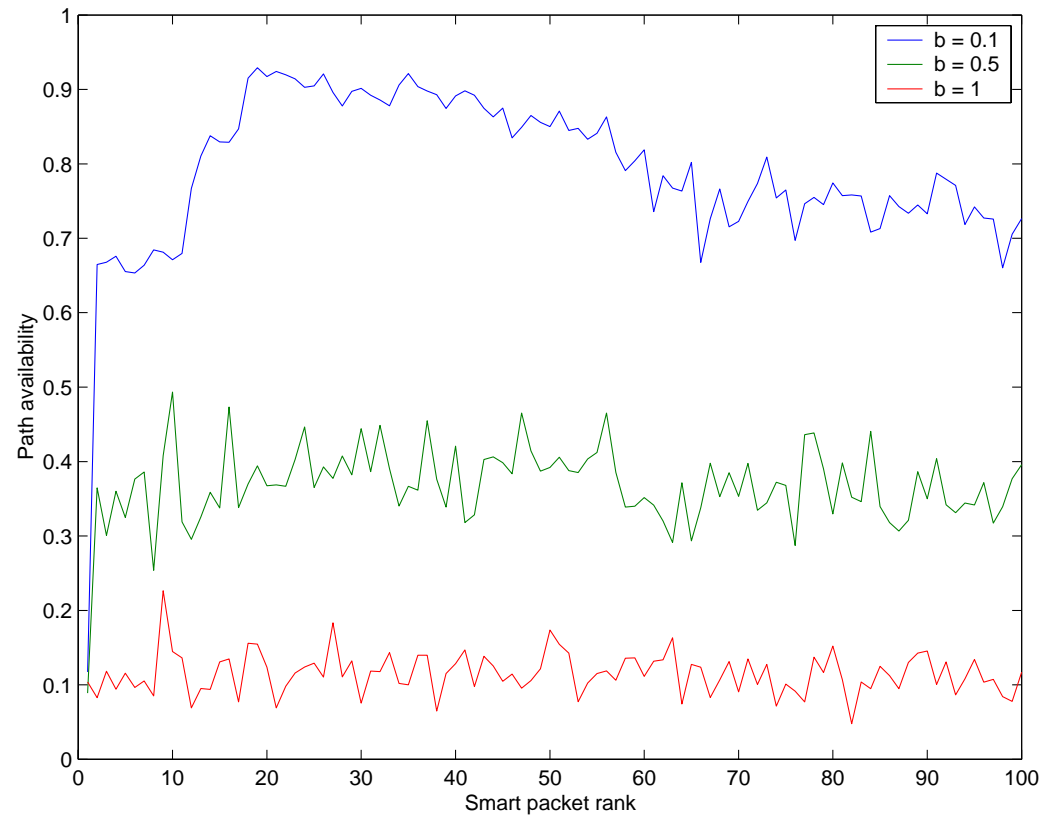
- $B_i$  = Battery lifetime at  $n_i$
- $B_m$  = Maximum battery lifetime

$$P_l(n_i, n_{i+1}) = 1$$

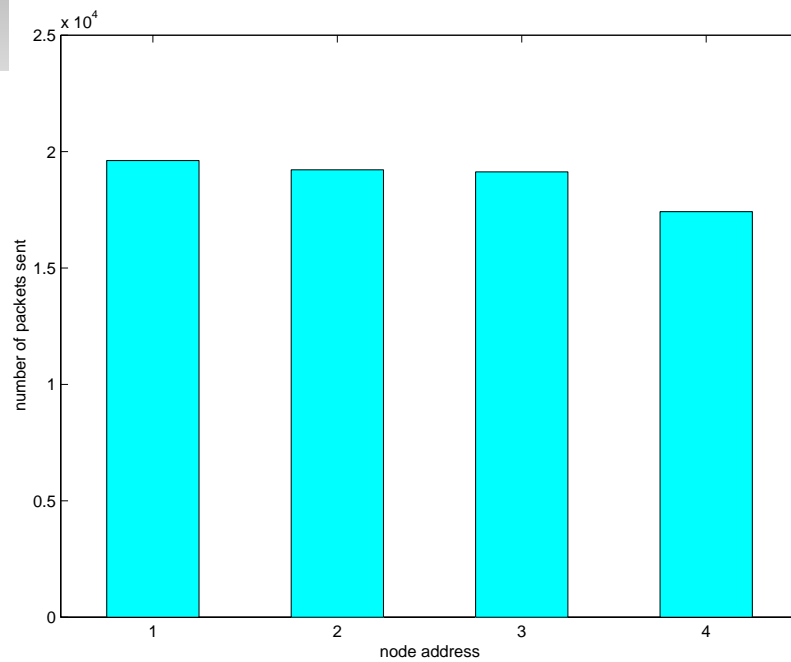
- Experiments on real wireless testbed are hard to conduct
- Simulation model integrated into NS-2
  - The VINT Project (UCB, LBL, USC/ISI, Xerox PARC)
  - C++/oTCL
  - CMU Monarch's wireless extensions



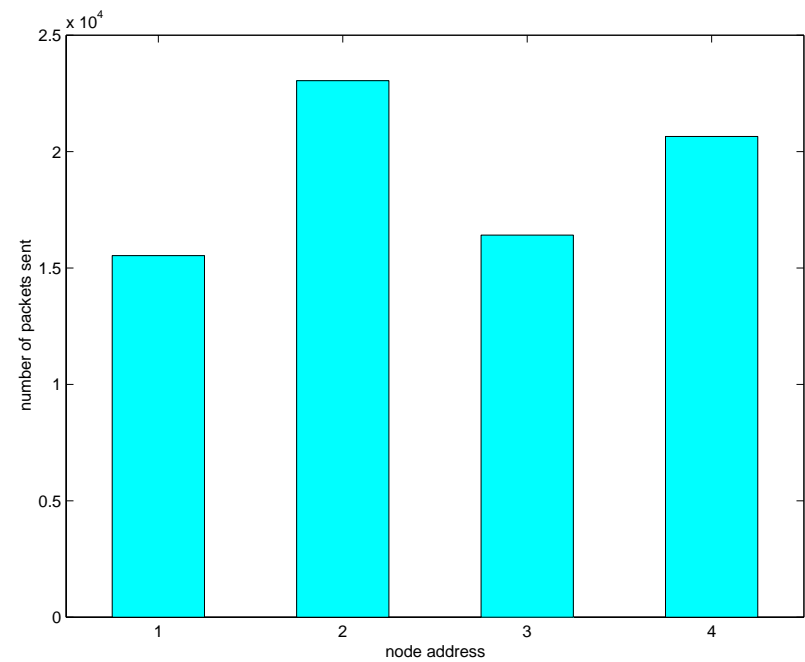
# Learning of Path Availability



# Packet Departures per Core Node

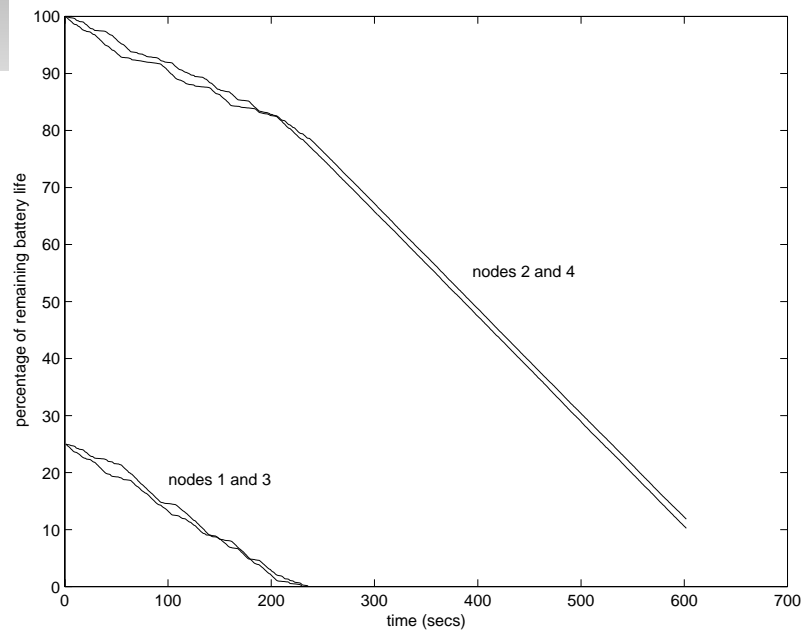


(a)  $G = f(D)$

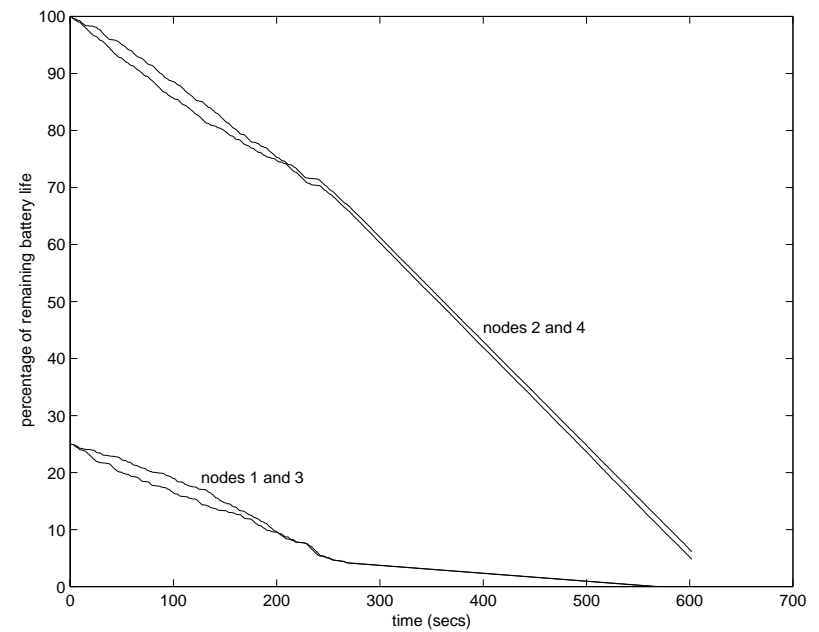


(b)  $G = f(D, E)$

# Percentage of Remaining Battery Lifetime Versus Simulated Time



(c)  $G = f(D)$



(d)  $G = f(D, E)$

# Conclusions

- We have introduced a routing algorithm capable of handling the dynamics of MANETs.
- Our solution presented a good balance between flooding and learning to acquired routes
- We formulated learning goals for smart packets in terms of round-trip delay and remaining battery lifetime at nodes
- Our simulations demonstrated the ability of our proposal to avoid low power routes, which eventually would lead to better quality communications.