

Localized Techniques for Broadcasting in Wireless Sensor Networks



Presenter: Chiara Petrioli

Paper accepted at
ACM DIALM-POMC
Philadelphia, October 1st 2004

■ Ad Hoc Networks

- Infrastructureless wireless networks in which nodes acts as relay for packets generated and addressed to different nodes

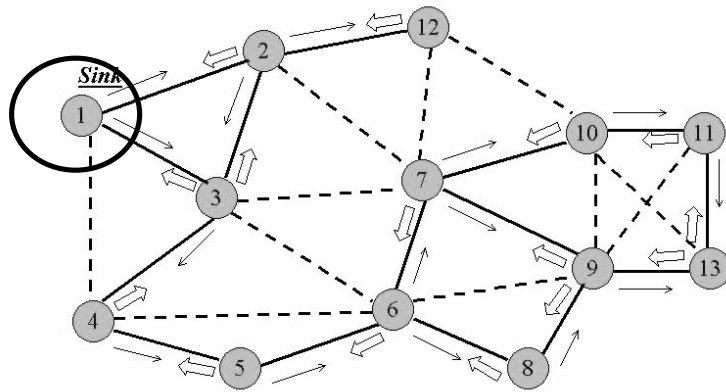
■ Wireless Sensor Networks

- Ad Hoc networks whose nodes are equipped with sensors, and a wireless transceiver, designed to monitor events and deliver information on events to 'sinks' –gateways to other networks or capable of processing such info
 - High Volumes of Nodes
 - Simple, inexpensive devices
 - Energy, memory and computing capabilities of the nodes very limited
 - Low data rate of the communication channel, low traffic
 - Communication from and to the sinks

Why is Broadcasting important for Ad Hoc and Sensor Networks

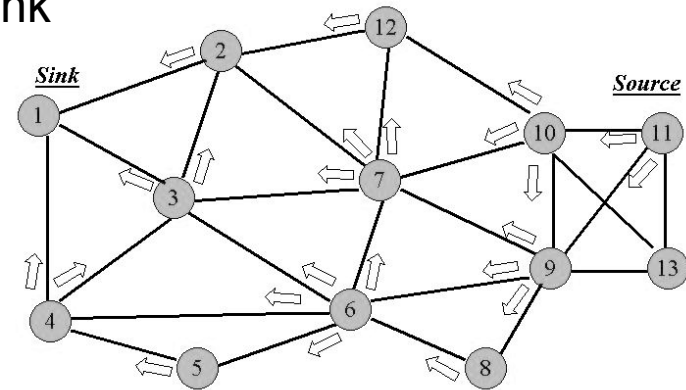
- Many routing protocols rely on *flooding* of information
 - Ad Hoc Networks: AODV, DSR
 - Sensor Networks: Directed Diffusion
- **Communication in Wireless Sensor Networks (WSN) is inherently one-to-all or many-to-one (independently of the particular routing protocol adopted)**

The sink propagates interests 'send me information on pink elephants every 10min'



a)

Upon detecting events matching a given Interests sources send packets back to the sink



d)

The trivial approach: flooding

- Flooding: upon receiving a packet each node rebroadcasts it to all its neighbors *APART* from the one from which it has received it
 - Pros: each node transmits the message once, all nodes are reached by the broadcasting process (ideal channel)
 - Cons: very resource consuming
 - High network load → this increases the probability of collisions especially at low data rate (WSN)
 - High energy consumption (for transmitting and receiving broadcast packets)

Broadcasting in Ad Hoc and WSN

Desirable Features

■ Low overhead

- Need of a small amount of information (one-hop neighborhood knowledge) for the broadcasting process to operate, keep to the minimum the amount of extra information transmitted.

■ Percentage of nodes re-broadcasting the message

■ Percentage of nodes involved in the reception of the message (reception of the message might happen several times). The two items above affect

- Network load, collision probability
- Energy consumption

■ Reliability (high percentage of reached nodes)

■ Scalable, simple, resource saving

Localized techniques for

broadcasting

- Simple, distributed solutions only relying on a minimum amount of information (e.g. one-hop info achievable via hello messages).
 - Is this really the minimum that can be achieved? Cannot we use protocols which do not require any info on the neighborhood? *Yes, BUT this kind of information tends to be needed when awake/asleep schedules are taken into account (first we will consider the case when neighbors and awake/asleep schedule are known then we will extend to the case when they are not)*
- Contributions of the paper:
 - Design of energy-efficient scalable simple localized techniques for broadcasting
 - Idea 1: local rules for generating a sparse virtual topology over which flooding is performed
 - Idea 2: on line schemes
 - Impact of different nodes deployment (not necessarily uniform on broadcasting performance)
 - Comparative evaluation of these solutions and probabilistic flooding

The Irrigator Protocol

- Given are n points distributed uniformly at random in a unit square
 - Nodes have a fixed transmission range equal to r , there is an edge between two nodes iff they are in visibility (i.e. their euclidean distance is $\leq r$)
 - c is a parameter named local connectivity
 - We select a subgraph G_c of the topology graph G_r as follows:
 - Each node randomly selects c among its neighbors
 - A link (u,v) is included in G_c if at least one among u,v selected the other
 - What is the likelihood that G_c is connected and how are G_c connectivity properties related to those of G_r ?
 - Extensive simulations show that for $c \geq 4,5$ G_c has the same global connectivity properties of G_r .
 - Number of connected components
 - Relative size of the giant componentAre the same for varying nodes densities, and simulation scenarios
- BUT** G_c has a much more reduced number of links over G_r
- reduced load, reduced energy consumption

Irrigator: features and implementation

Se i nodi conoscono i loro vicini e quando saranno attivi...

■ How to implement it:

- *Gather info on the one hop neighborhood via basic hello message exchange*
- *Communicate in the next hello message the selected neighbors (since c is small it is a few extra bytes)*

■ Desirable features

- Low overhead OK
- Low percentage of nodes rebroadcasting Not focusing on it
- Low percentage of nodes having to receive the message
- Lower network load and energy consumption
- Simple, distributed, scalable
- Reliable (not deterministically but whp)

Analytical results

■ Irrigator

- When $c \geq 2$, as n goes to infinity, the probability that the giant component of G_c includes all nodes goes to 1.

Gossip and Fireworks

■ Gossip – probabilistic flooding

- Each node receiving a broadcast message rebroadcast with probability p

■ Fireworks

- Each node receiving a broadcast message
 - with probability p rebroadcast it to all its neighbors (APART the one from which it has received it)
 - With probability $(1-p)$ it selects c among its neighbors to which to rebroadcast

■ Fireworks rationale:

- On-line approach, meeting all the low overhead, low resource consuming, simple, scalable, reliable desirable features
- Does not limit the number of nodes rebroadcasting as Gossip effectively does

Analytical results

■ Irrigator

- When $c \geq 2$, as n goes to infinity, the probability that the giant component of G_c includes all nodes goes to 1.

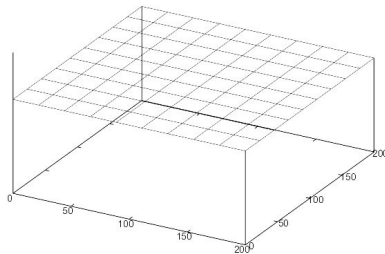
■ Fireworks

- If $p = \log^*(n)/n$, as n goes to infinity, the probability of reaching all nodes goes to 1.

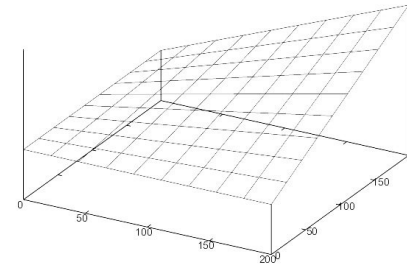
Simulation Scenarios

- $N \leq 300$ nodes with transmission radius equal to 30m are deployed in a squared area of side $L=200\text{m}$;
- Two nodes are neighbors in the visibility graph iff their Euclidean distance is less or equal to 30m (unit disc graph assumption).
- Nodes deployments:

Uniform

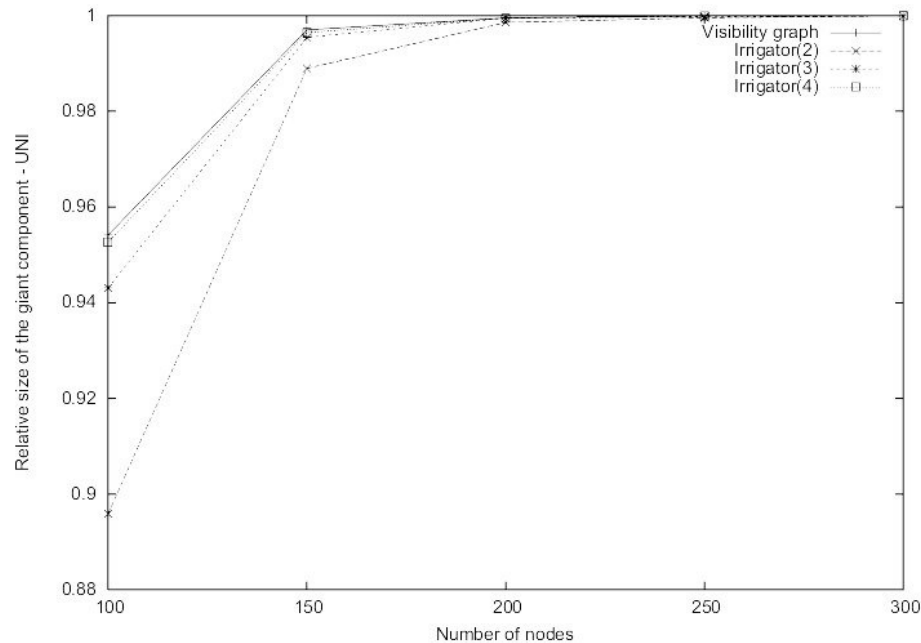


Hill



- A source is randomly selected among the nodes in the visibility graph Giant Component and the broadcasting process is simulated.
- Results (averaged over 100 runs on different topologies) refer to:
1) number of nodes involved in message transmission, 2) number of links over which the message is transmitted, 3) percentage of successfully reached nodes.

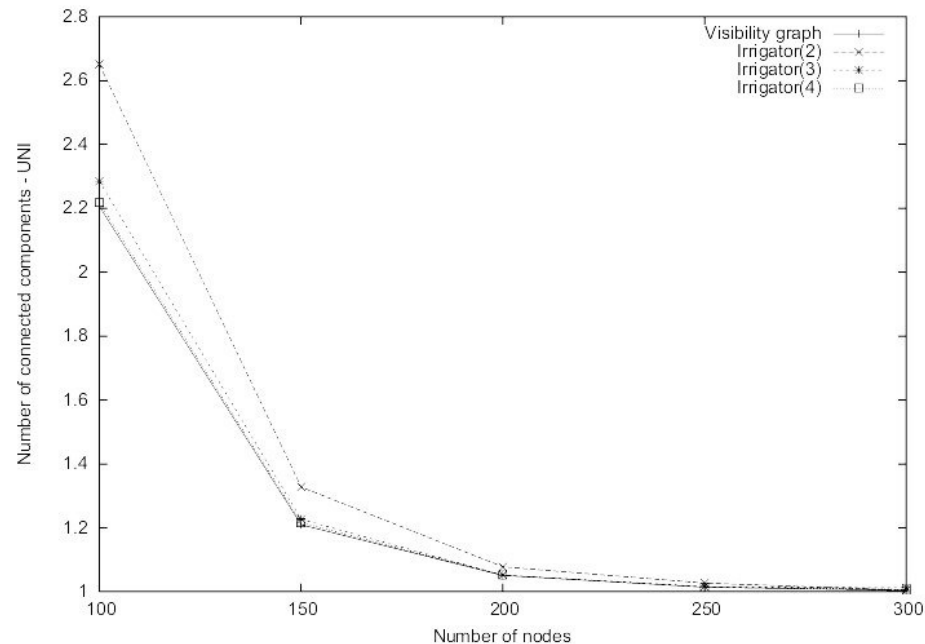
Relative size of the Giant Component (Unif. deployment, Irrigator)



Irrigator (c=2,3,4)

- For the visibility graph G_r and for G_c , $c \geq 4$ in the Irrigator protocol, the plots are basically identical, independently of the density.

Number of connected components (Unif. deployment, Irrigator)



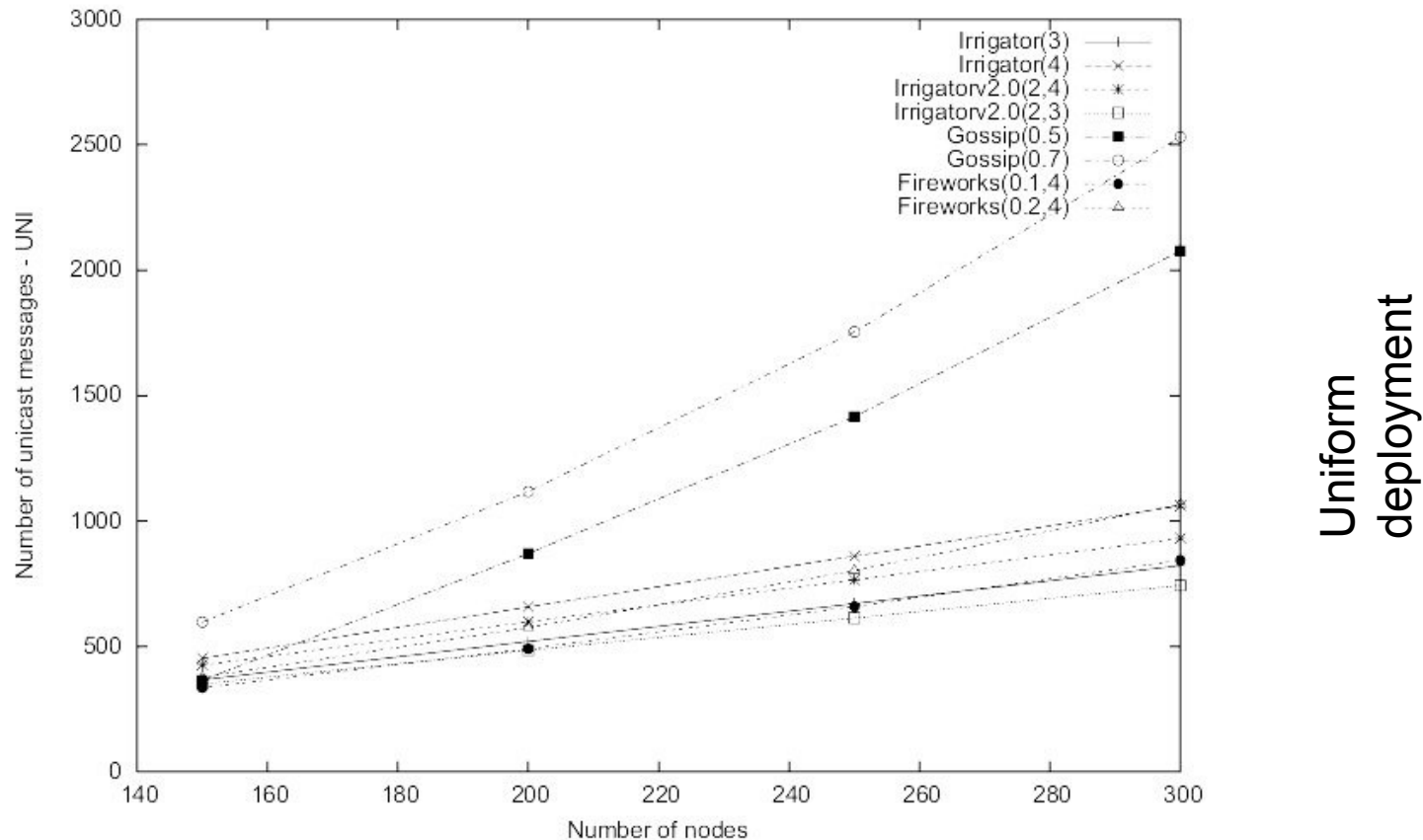
- Gr and Gc, $c \geq 4$ Irrigator has the same behavior \rightarrow as far as global connectivity is concerned it does not pay off to set up all possible links. The virtual sparse topology generated by Irrigator is enough to maintain global connectivity.

Number of traversed links (Unif. deployment, Irrigator)

- Irrigator $c=4$ reduction in the number of links over basic flooding

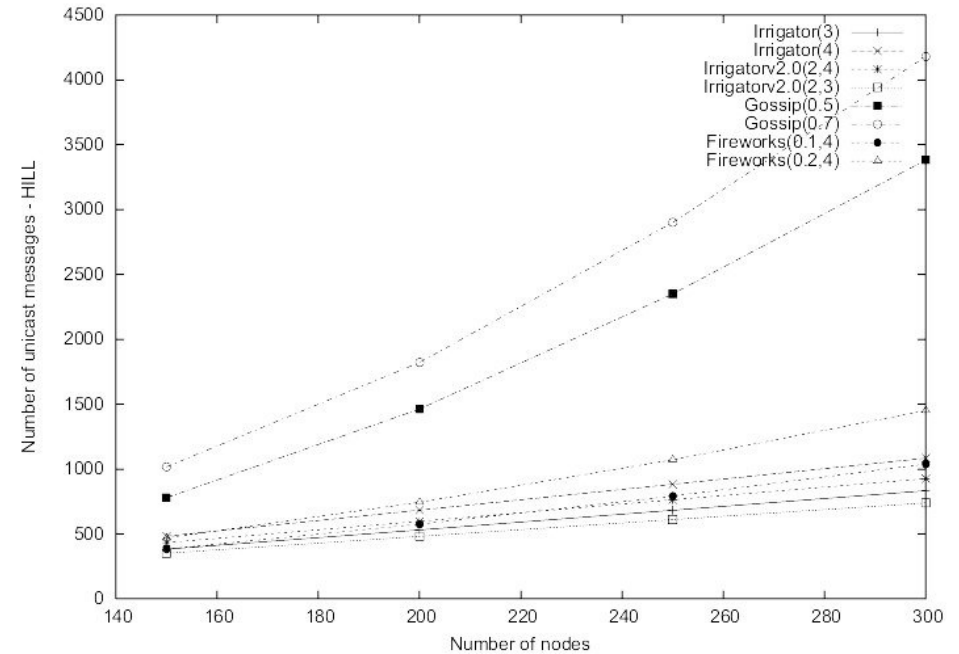
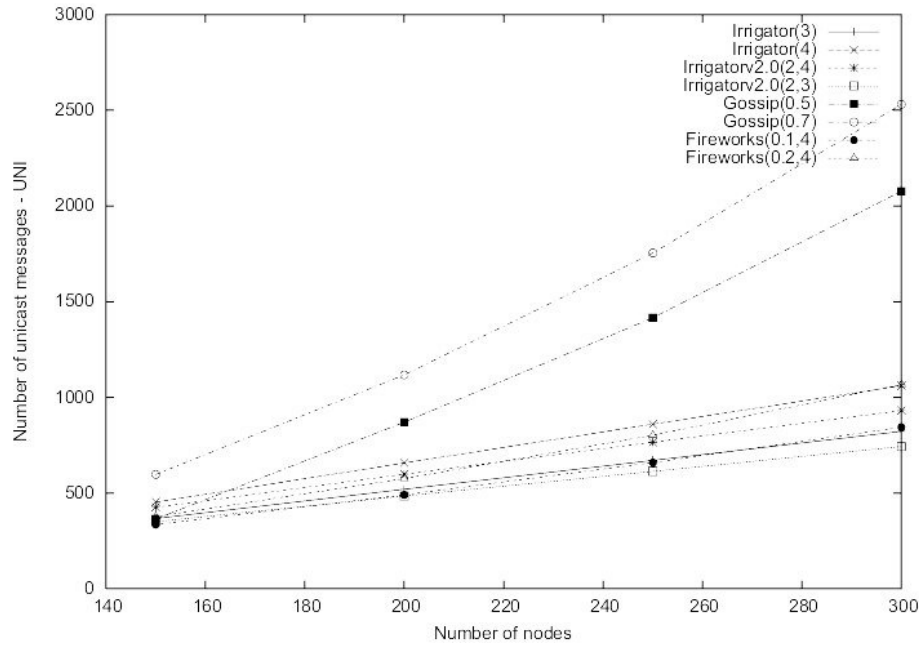
n	Uniform	Hill
150	33%	>50%
300	60%	75%

Number of broadcast packets received



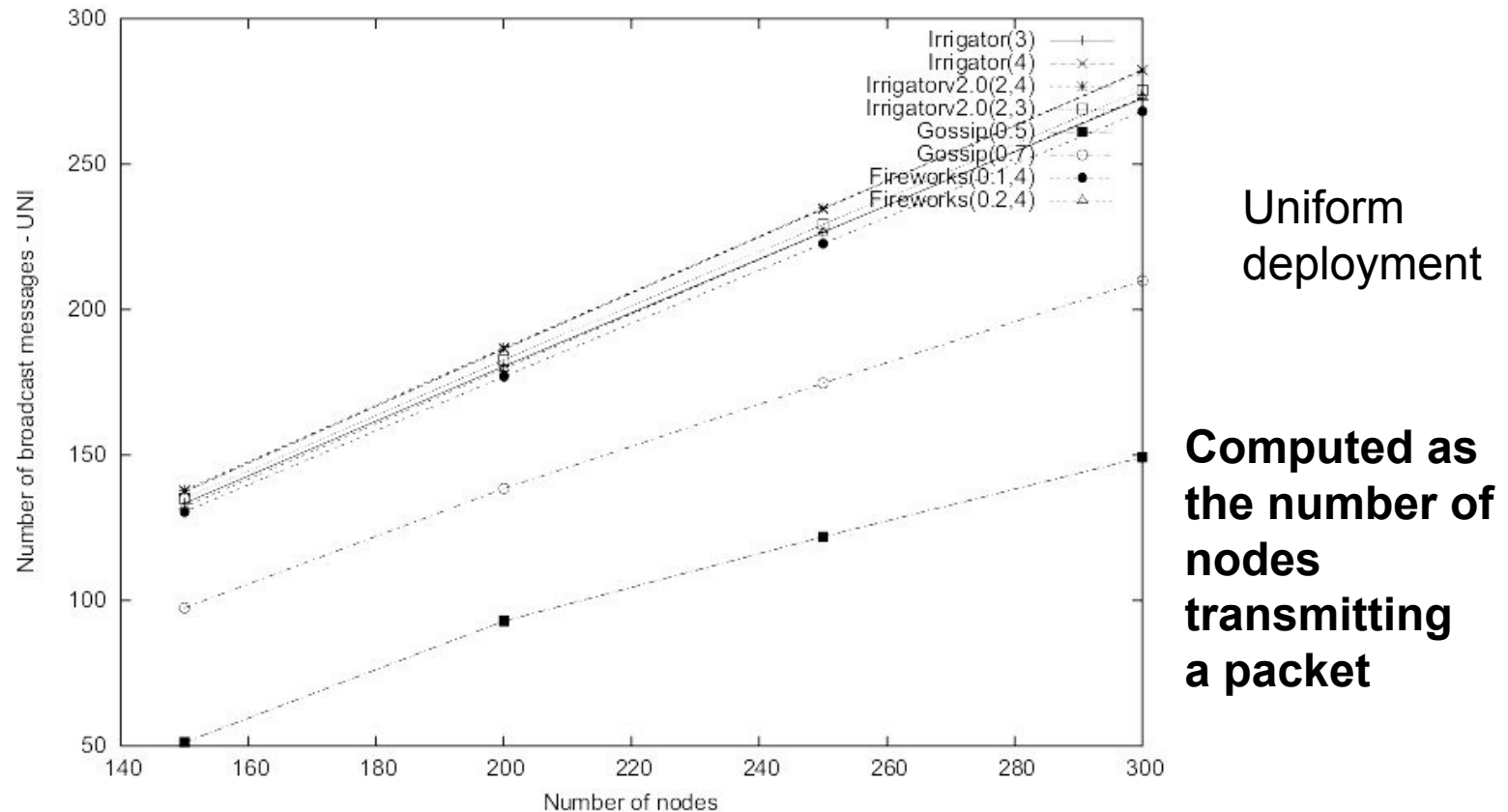
- Improvements increase with n (and the density) and with the Hill distribution
- Fireworks has similar performance for small n and then increases the number of traversed links, which is 30% higher than Irrigator at $n=300$, Hill

Number of broadcast packets received Uniform vs. Hill



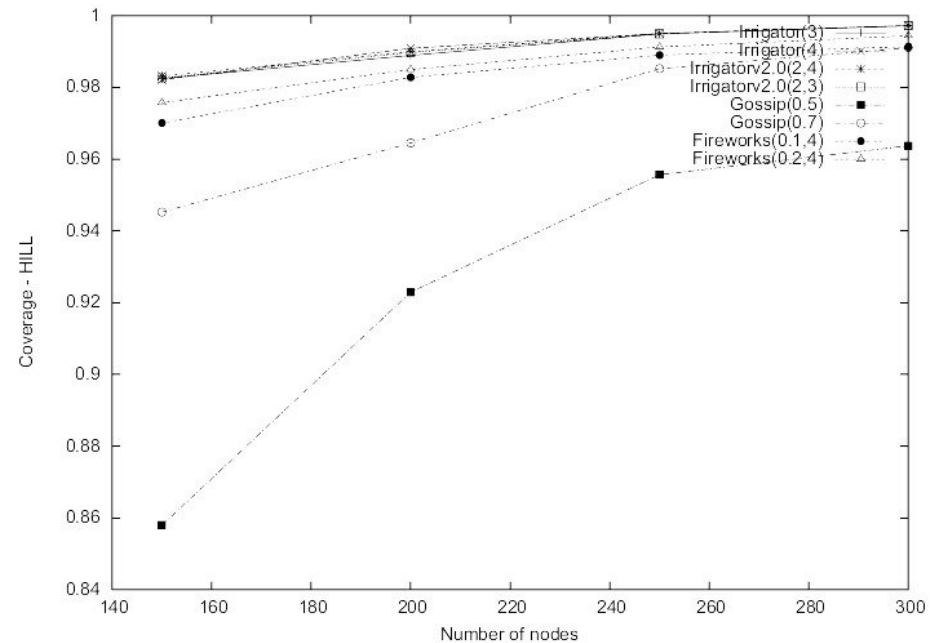
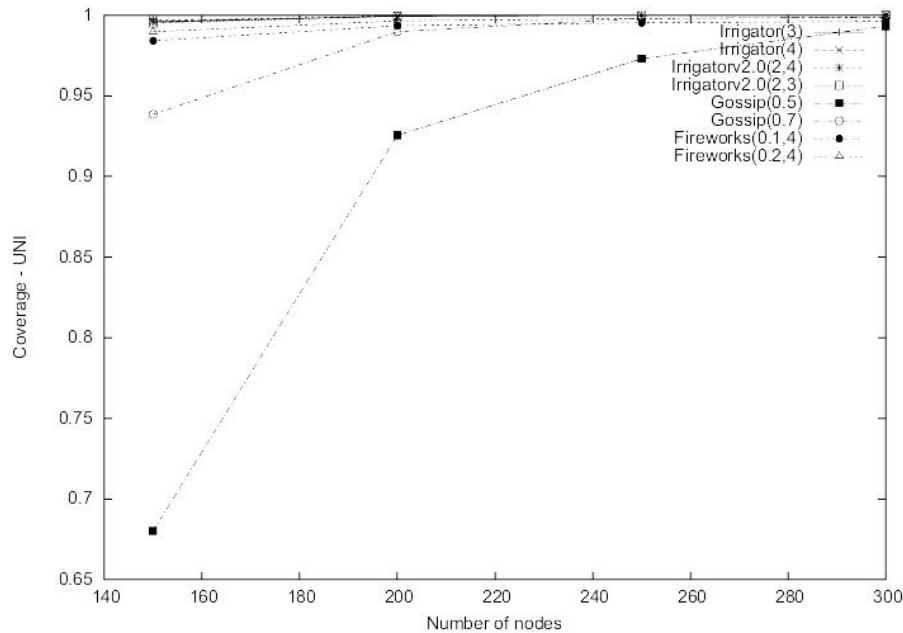
- Improvements increase with n (and the density) and with the Hill distribution
- Fireworks has similar performance for small n and then increases the number of traversed links, which is 30% higher than Irrigator at $n=300$, Hill

Number of broadcast packets transmitted



- With respect to this metric Gossip shows better performance as expected. The price to pay is increased energy consumption (2-3 times as much as the other protocols) and lower reliability.

Percentage of successfully reached nodes (Coverage)



- Irrigator reliability comparable to basic Flooding, Fireworks no more than 2% degradation
- In Gossip a high number of links have to be traversed for the protocol to be reliable. Uniform distribution, $p=0.5$, only 65% of the nodes are successfully reached.

Open problem

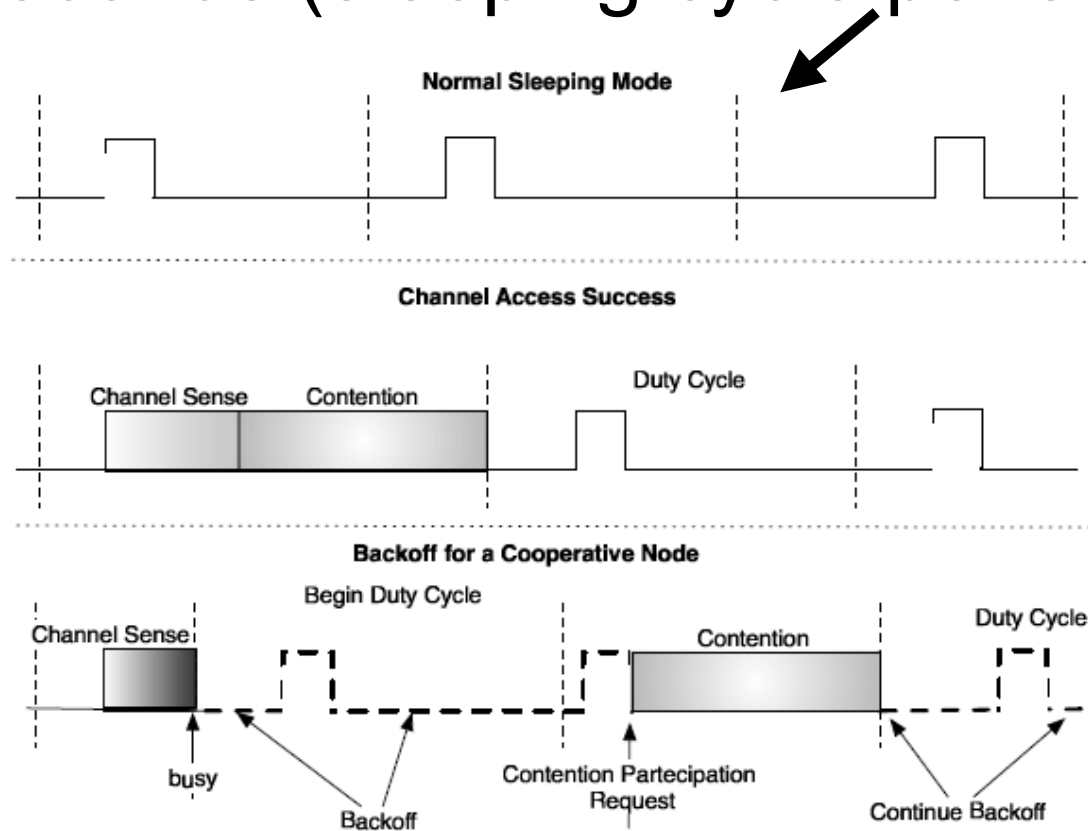
- Easy to apply Fireworks if I know my neighbors and their wake-up schedule
 - This is not the case in GeRaF
 - It would be desirable to have schemes which do not require this assumption
 - If I don't know my neighbors and their awake/asleep schedule it is difficult even to do a flooding!
- How can this (open) problem be solved?
 - It is possible to estimate the number of neighbors based on the knowledge of the duty cycle and of the number of ack I receive when I transmit to my neighbors
 - Use this information to know when a node should stop broadcasting the interest to its neighbors (having reached all the ones it wanted to reach)....

IRIS

- Integrates
 - a GeRaF-like awake asleep schedule
 - a MAC and hop count routing
 - Fireworks-like interest dissemination (without requiring the knowledge of neighbors and of their wake-up schedule)
- On going work we are doing
 - Simulated → good performance
 - Implemented
- We will show you a small demo Friday!!

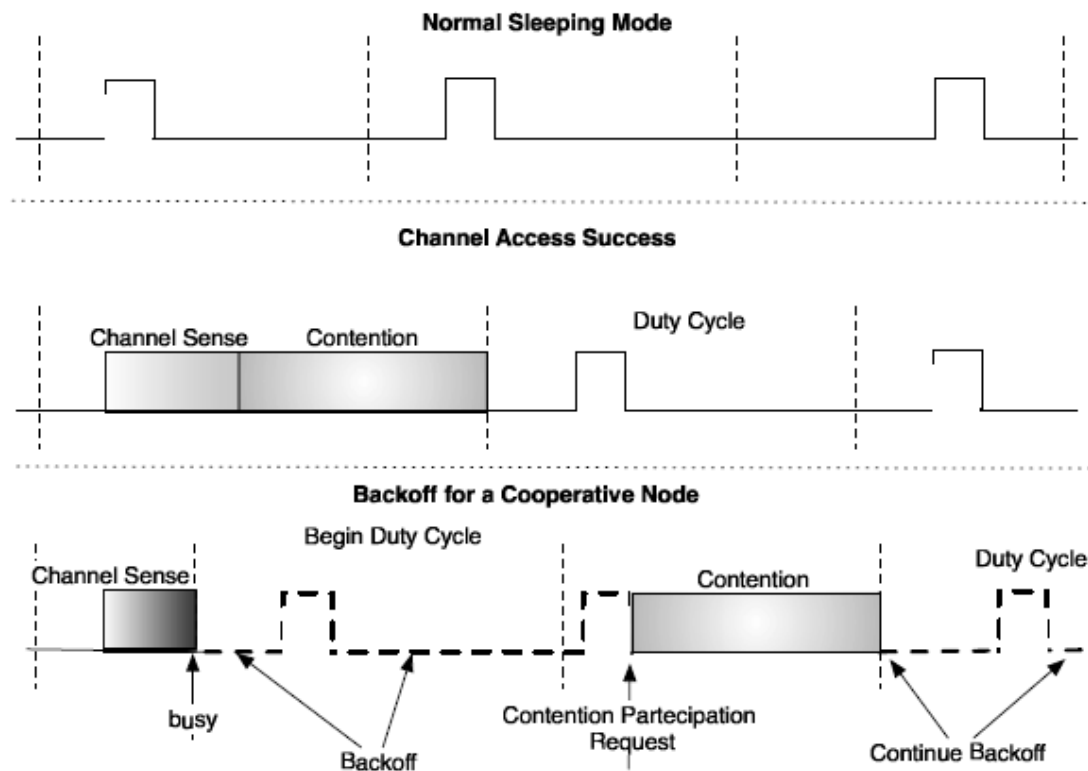
IRIS-Nodes sleeping behavior

- A given node divides time into periods of T seconds (sleeping cycle periods).



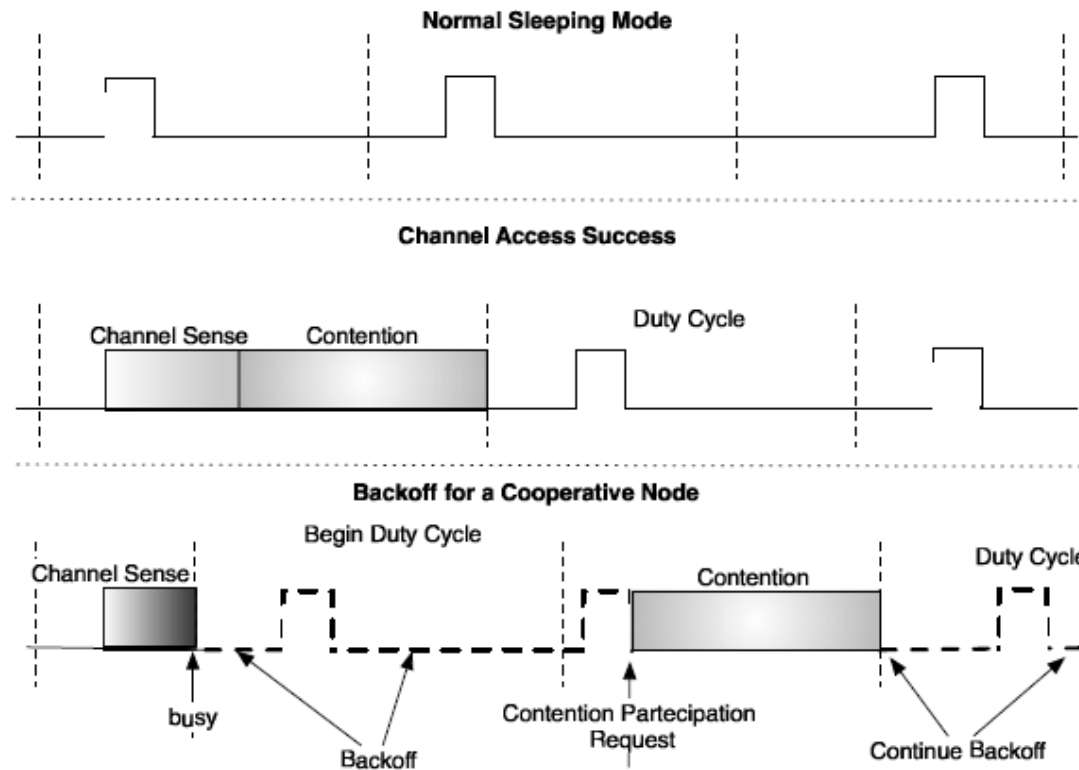
IRIS-Nodes sleeping behavior

- At the end of every sleeping cycle it randomly picks a real number t_a in $[0, T(1-d)]$ (d is the duty cycle).
- In the following sleeping cycle the node will sleep for the first t_a seconds, will then wake up for $T*d$ seconds, and go to sleep again till



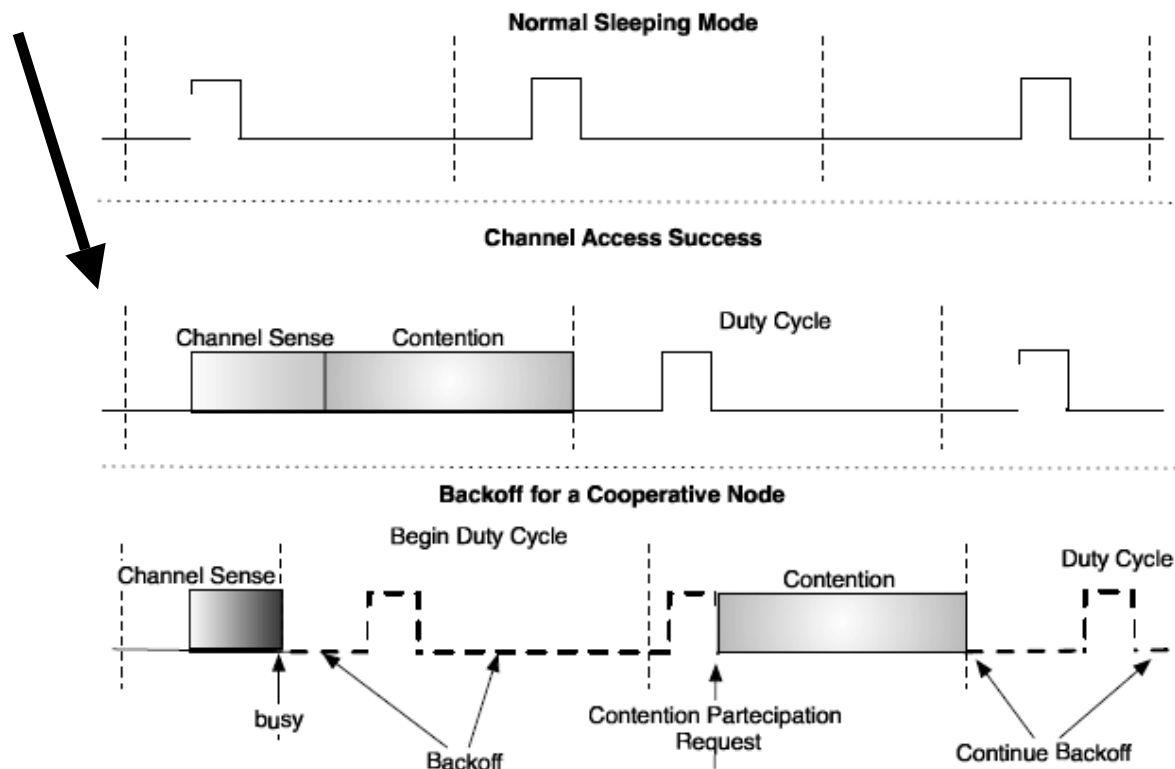
IRIS-Nodes sleeping behavior

- Nodes sleeping cycles are completely asynchronous.



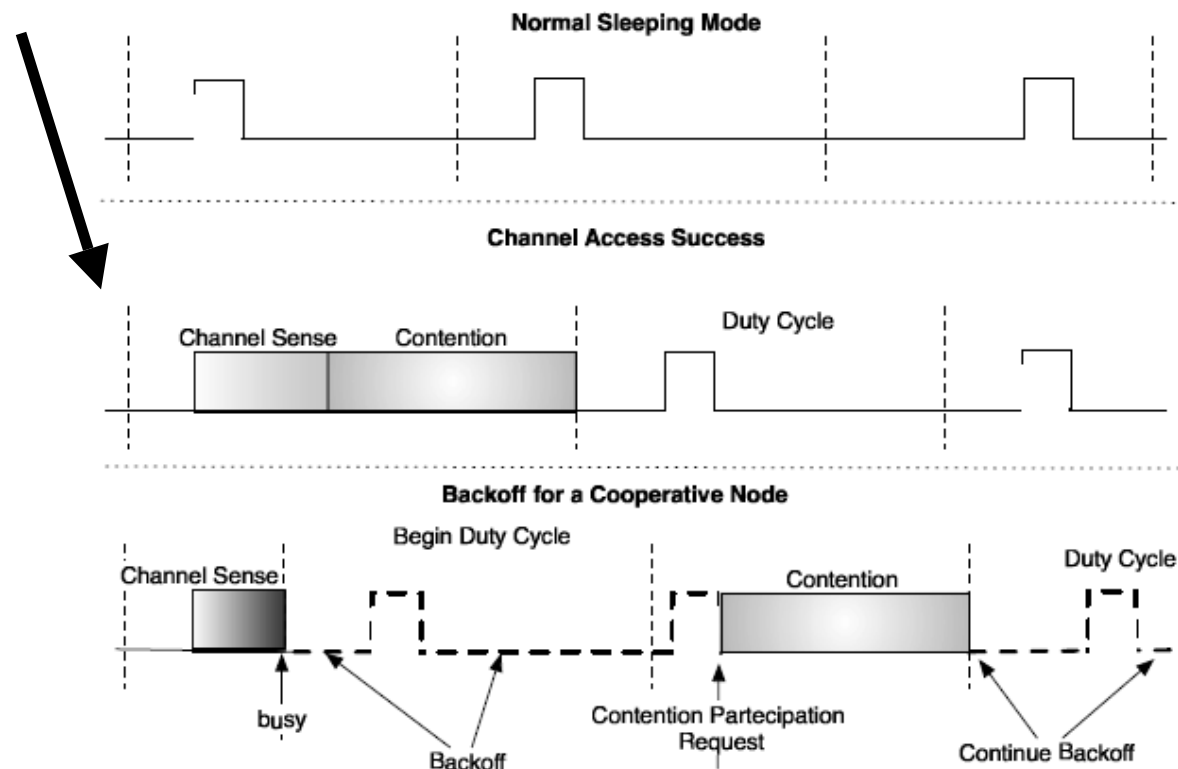
IRIS-Nodes sleeping behavior

- When a node has data to send it does so using a CSMA-based protocol (more details follow)
 - The channel is sensed
 - If it is idle the node starts a contention to select a relay among its neighbors



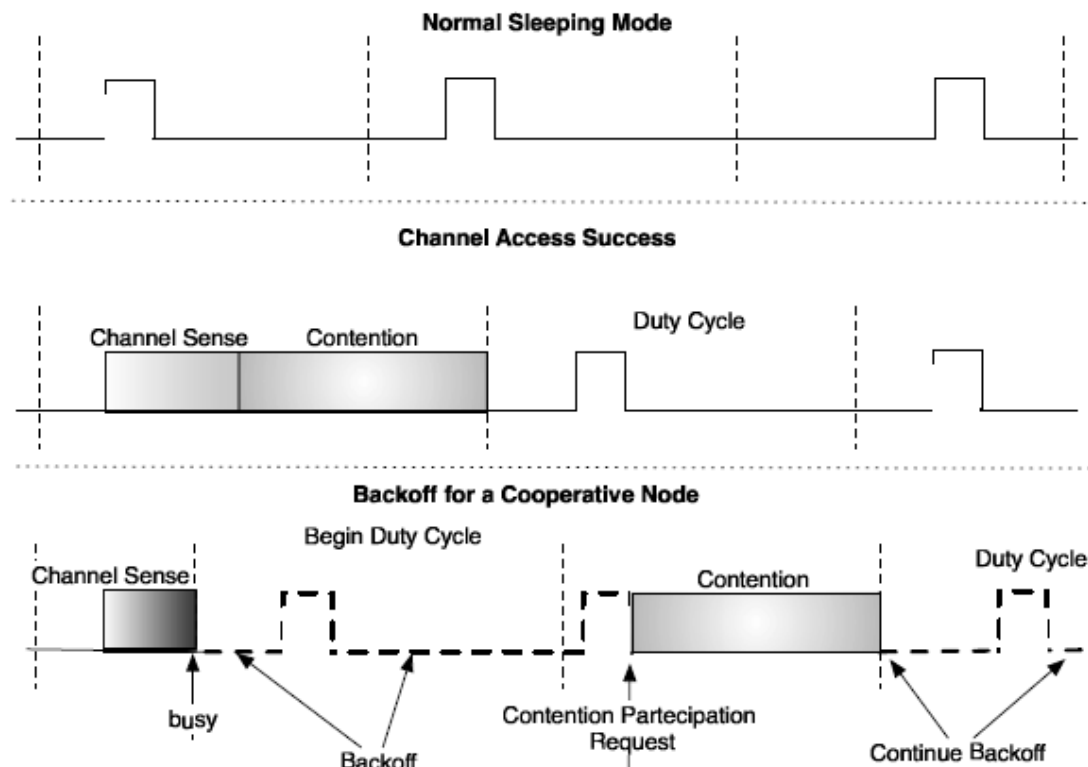
IRIS-Nodes sleeping behavior

- The node remains active until the contention is completed, a relay is selected and the packet is successfully transmitted to it (i.e. until an ACK is received)



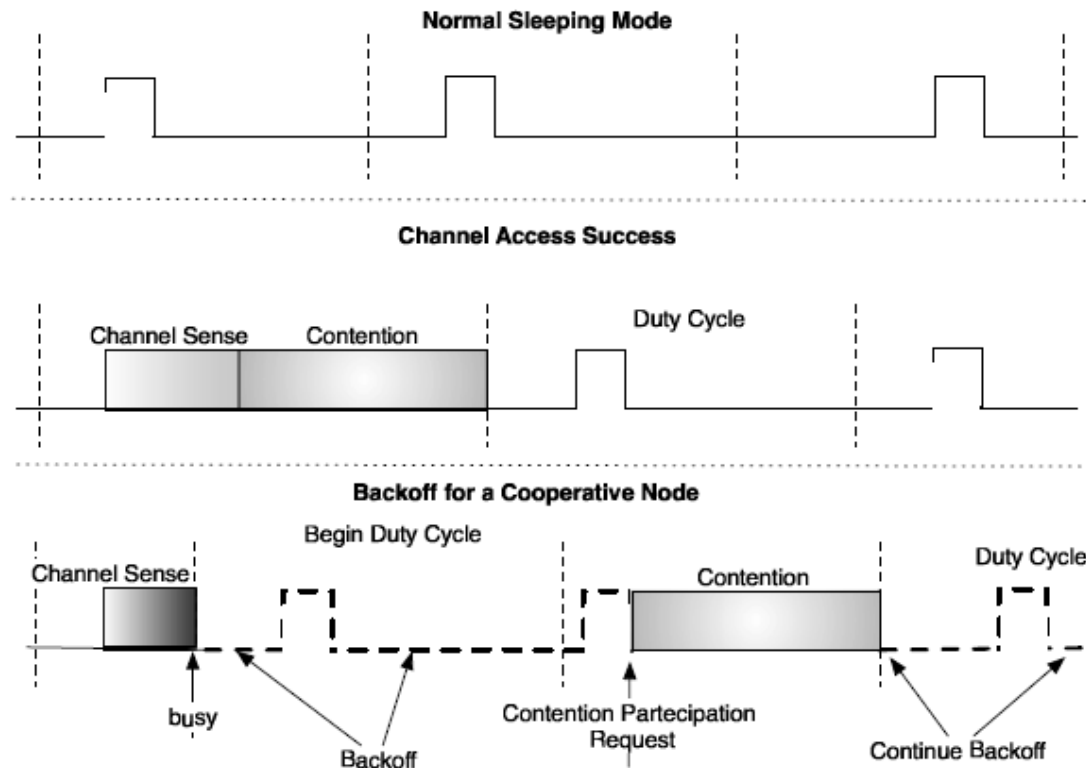
IRIS-Nodes sleeping behavior

- If the channel is sensed busy a backoff timer is started
- The backoff timer is decreased only when the node is in sleeping mode



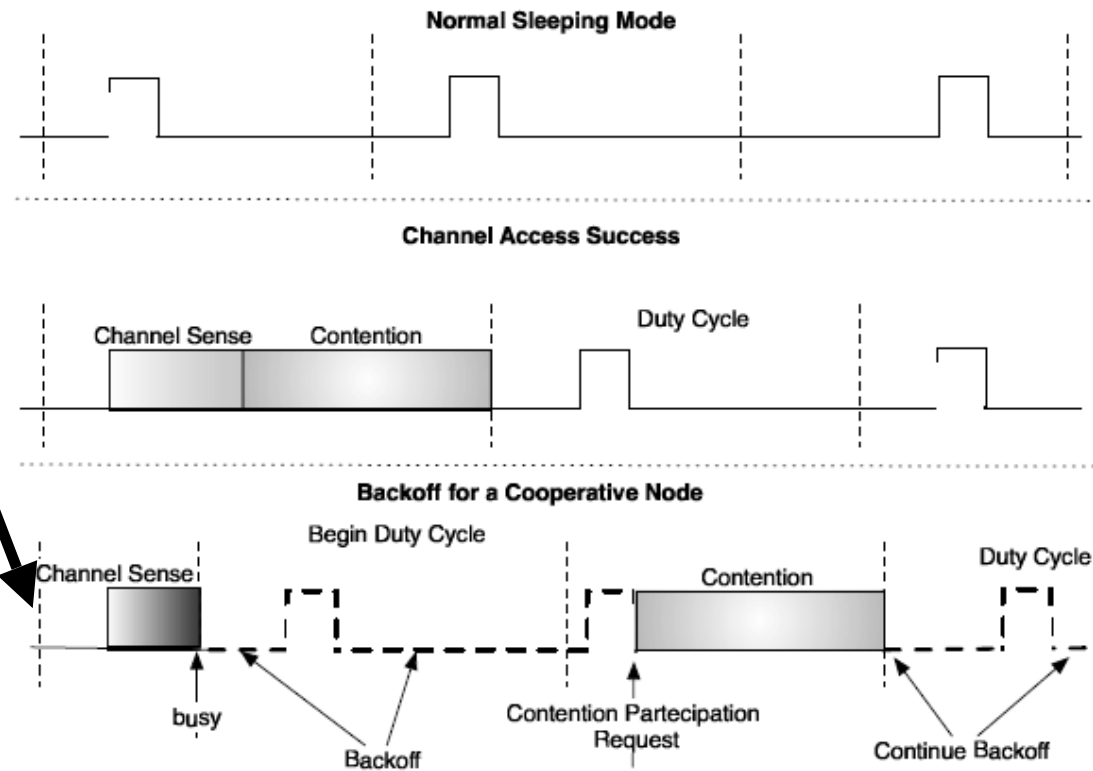
IRIS-Nodes sleeping behavior

- At the end of the backoff the node performs a new channel sense. If the channel is busy a new backoff is performed (the backoff interval is doubled)



IRIS-Nodes sleeping behavior

- A node in backoff can participate to contentions (we call him cooperative node).
 - It follows the basic sleeping cycle.
 - If a contention starts when it is ON it participates to it.



IRIS–Nodes density estimation

- Given a nodes duty cycle d how can each node estimate its number of neighbors?
 - We proceed sampling the active neighbors in ‘rounds’
 - If rounds are enough separated in time sets of active neighbors are statistically independent → each node is active in a round with probability d
 - At each round each node estimates the cardinality of the set of active neighbors
 - several different possibilities...
 - turns out that a simple approach: each active node answers with a jitter (i.e. randomly selecting a time in a WINDOW interval) is both simple and effective.

IRIS–Nodes density estimation

- Say that an estimation procedure lasts r rounds
- Let k_i be the number of active neighbors at the i -th round *which have not been counted before*
- After r rounds the probability that the number of sampled active neighbors k_1, k_2, \dots, k_r if the number of neighbors is n and the duty cycle is d is given by:

$$L(n, k_1, \dots, k_r, d) = \prod_{i=1}^r \binom{n - S(i)}{k_i} d^{k_i} (1-d)^{n-S(i)-k_i} \quad (1)$$

- n is the number of neighbors and:

$$S(i) = \begin{cases} 0 & i = 1 \\ \sum_{j=1}^{i-1} k_j & i > 1 \end{cases}$$

IRIS–Nodes density estimation

- We use a maximum likelihood estimator: the *estimated* number of neighbors \tilde{n} will be that value which maximizes $L(n, k_1, \dots, k_r, d)$

$$\tilde{n} = \arg \max_n L(n, k_1, k_2, \dots, k_r, d)$$

How many samples are needed to have an accurate estimate?

**AN is obtained analytically
WINDOW are the simulated values**

IRIS-Interest Dissemination

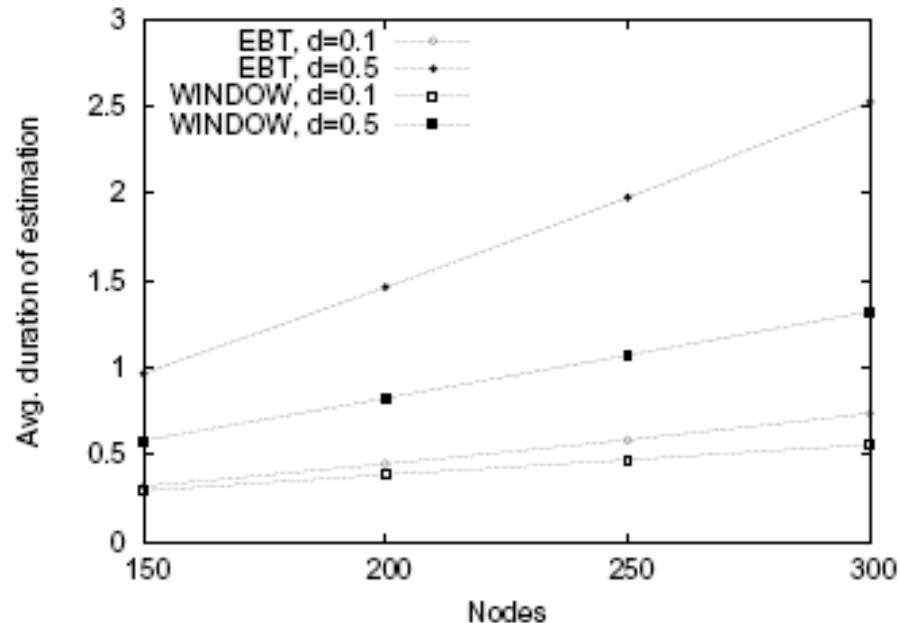
- Fireworks extension (Fireworks integrated with the process of neighbors estimation)
 - each node which receives a message tosses a coin
 - with probability p it transmits to all its neighbors → how does it know when the interest has reached all its neighbors?
 - With probability $(1-p)$ it transmits to c of its neighbors → can get ACK but if after some time it hasn't received c ACKs how does it know whether it has c neighbors?
 - Interest dissemination performed jointly with neighbor estimation (I transmit the interest, active nodes wait for a jitter and ACK. I use this as a sample of the number of active neighbors, needed to estimate the number of neighbors. The estimate of the number of neighbors is thus used to decide when to stop retransmitting the interest).
 - Together with the interest I transmit also an hop count → at the end of the dissemination each node knows its hop count from the sink.

Simulation results

- 150-300 nodes scattered randomly and uniformly in a square area with side 200m
- Node transmission radius= 30m
- Sink is placed at the center of the area
- Channel capacity: 38.4Kbps
- Energy model and other parameters: EYES prototypes
- Averages over 100 experiments each lasting 2000 s

Fireworks ($c=4, p=0.2$)

Average duration of a round



How many rounds: 7 are enough in these scenarios

The simpler the counting process, the lower the number of active Neighbors, the faster the round

Time needed by WINDOW at 300 Nodes: 11s (d=0.5), 26s (d=0.1)

- We are using less rounds than what derived analytically (what was shown to produce a low estimation error analytically) → in WINDOW (200 nodes, d=0.5) this can lead to a 55% estimation error!
 - The reason we kept estimation short is that in practice Fireworks is robust enough to reach all nodes even with such estimation error!
 - In realistic scenarios multiple interest dissemination would allow to converge fast to a very accurate estimation.

IRIS-Convergecasting

- Let us assume node i is the one having a packet to transmit and that its hop count is h
- the forwarding process is at stage t
 - t is the number of times a decision has been made on whether to stay at the same level or advancing of one hop toward the sink
 - we reset t when we advance of one hop (i.e. if we reach stage t it means that for $t-1$ times we have decided to select a relay at the same distance in hops from the sink)

IRIS-Convergecasting

- The set $N_i(h)$ and $N_i(h-1)$ denote the sets of neighbors of node i which are respectively h and $h-1$ hops from the sink
- For each (i,j) , j in $N_i(h) \cup N_i(h-1)$ we associate a cost c_j .
 - Let $j^*(t,h)$ e $j^*(t,h-1)$ be the two candidate relays h and $h-1$ hops from the sink which have the minimum cost
 - The cost captures metrics such as residual energy, queue occupancy, link stability etc.

IRIS-Convergecasting

- Let us define

$$C_{tot}(t) = C_{par}(t) + c^t$$

Minimum cost to advance of one hop after t steps

$$C_{par}(t) = \begin{cases} 0 & t < 1 \\ \sum_{k=0}^{t-1} c^k & t \geq 1 \end{cases}$$

Minimum cost to go to a node at the same level

h-1

h

- The minimum cost of all paths to a node with hop count h-1 (when the packet is transmitted by nodes at hop count h) is

$$C_{tot}^{min}(t) = \min_{0 \leq k \leq t} \left\{ C_{tot}(k) \right\}$$

IRIS-Convergecasting

- Let us define

$$C_{tot}(t) = C_{par}(t) + c^t$$

Minimum cost to advance of one hop after t steps

$$C_{par}(t) = \begin{cases} 0 & t < 1 \\ \sum_{k=0}^{t-1} c^k & t \geq 1 \end{cases}$$

Minimum cost to go to a node at the same level

h-1


h

- The minimum cost of all paths to a node with hop count h-1 (when the packet is transmitted by nodes at hop count h) is

$$C_{tot}^{min}(t) = \min_{0 \leq k \leq t} \left\{ C_{tot}(k) \right\}$$

IRIS-Convergecasting

Must be estimated
Captures the expected
cost to go from the next
relay at the same level
up of one level



- We advance of one hop if

$$\mathcal{B}_2 = \left\{ X_t : C_{tot}^{min}(t) - C_{par}(t+1) \leq \epsilon \right\}$$

- How to implement this? Nodes in $N_i(n) \cup N_i(n-1)$ are inquired, the 'best node' is selected in both the two sets, previous equation is computed to decide whether to forward to the best relay at the same level or to the best relay $h-1$ hops from the sink

Simulation results

- Parameters as before...
- Two types of nodes: rich nodes have 240J initial energy, poor ones only 48J initial energy

IRIS-Convergecasting

- All packets are successfully delivered to the sink
- Reasonable latency (under second or few seconds) till the capacity is reached. More scalable than previous approaches (MACRO, GeRaF)
- Allows to select as relays resource-rich nodes more frequently

