Contention-Free MAC Protocols for Wireless Sensor Networks

Presented by ***Fikret Sivrikaya***

Joint work with ***Costas Busch, Malik Magdon-Ismail, Bulent Yener***

Computer Science Department, Rensselaer Polytechnic Institute
New York, USA

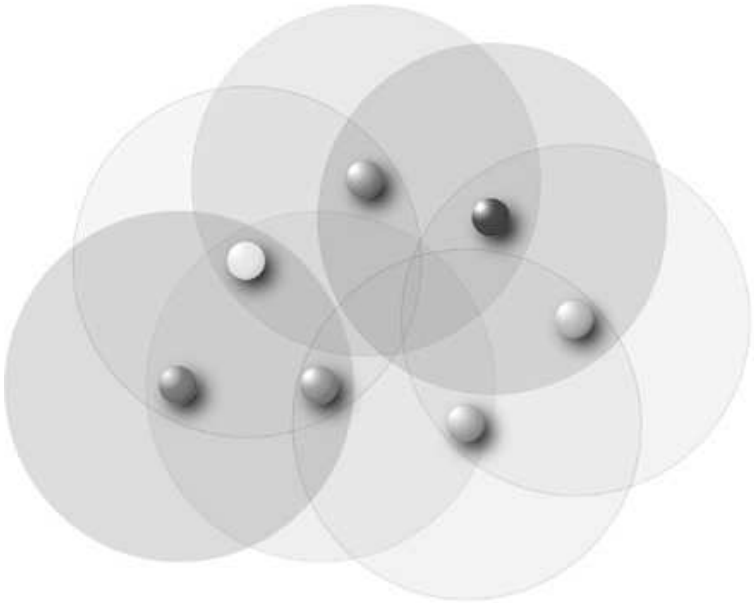
Outline

- Introduction
  - Sensor networks
  - MAC protocols
  - Previous work
- Model & Motivation
- Our Approach
  - LooseMAC Algorithm
  - TightMAC Algorithm
- Practical Considerations
- Summary & Future Work

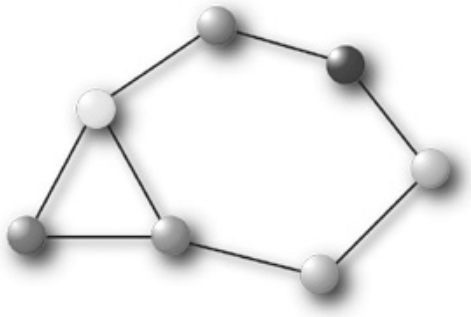
Wireless Sensor Networks

- A large number of limited power sensor nodes
- Distributed, multi-hop, ad-hoc operation; no infra-sctructure, no central control point
- Collect and process data from a target domain and transmit information back to specific sites
- Usage scenarios…
  - disaster recovery
  - military surveillance
  - health administration
  - environmental monitoring.

Wireless Sensor Networks



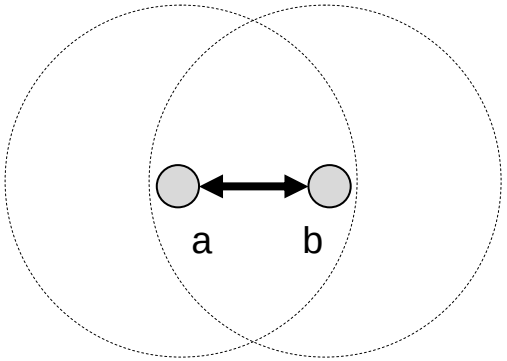
Each node has a **transmission range**, which determines its **neighbors**



Representation of the network as a graph

same transmission ranges $\Rightarrow$ symmetric links $\Rightarrow$ undirected graph

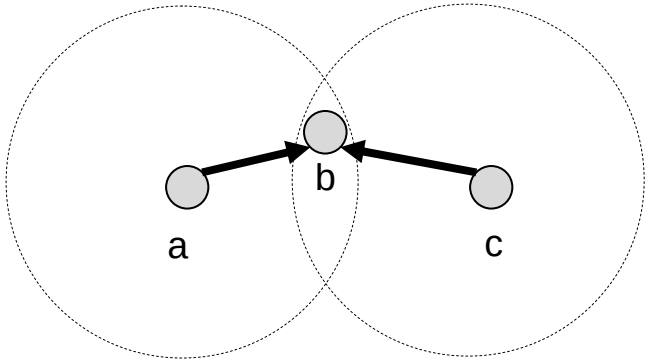
Interference / Collisions



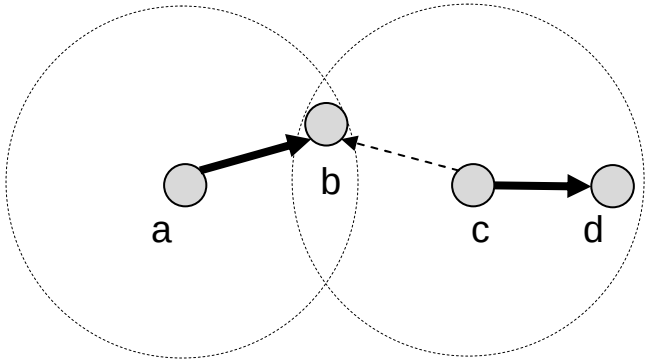
a and ***b*** interfere and hear noise only

Packets which suffered collisions should be re-sent.

Ideally, we would want all packets to be sent collision-free, only once…



Interference on node ***b***
("Hidden terminal problem")



Interference on node ***b***

MAC (Medium Access Control) Protocols

- Specify how nodes in a network access the shared communication channel.

- Two basic types
  - contention-based
  - contention-free

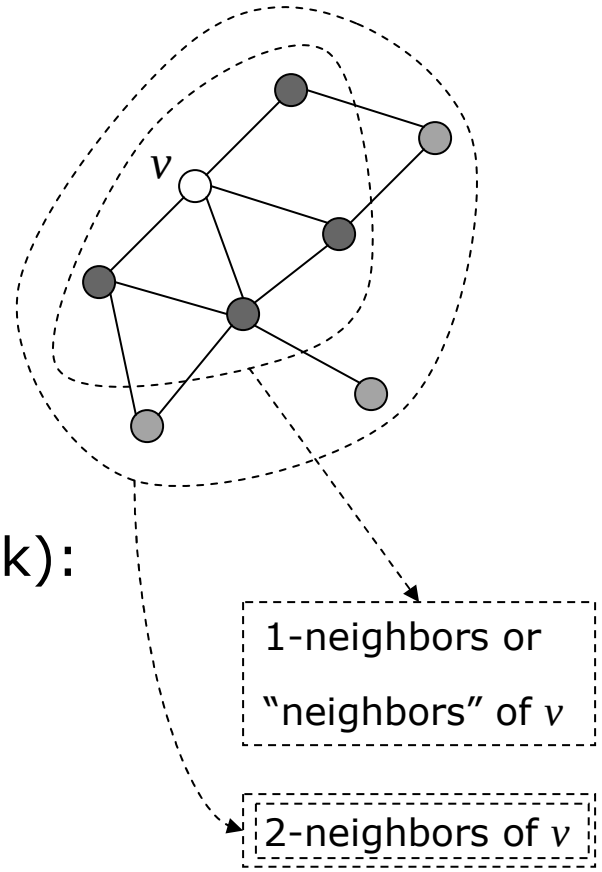
- Desired Properties of a Sensor Net. MAC Protocol
  - distributed
  - contention-free (collision free)
  - self-stabilizing
  - not require common global time reference

Previous Work

- Contention-based (random access)
  - ALOHA
  - CSMA (Carrier Sense Multiple Access)
  - IEEE 802.11
- Contention-free
  - FDMA
  - TDMA
  - CDMA
- Multi-layered approach
  - ASCENT (nodes decide themselves to be on or off)
  - S-MAC (virtual clusters based on common sleep schedules)

Notations Used…

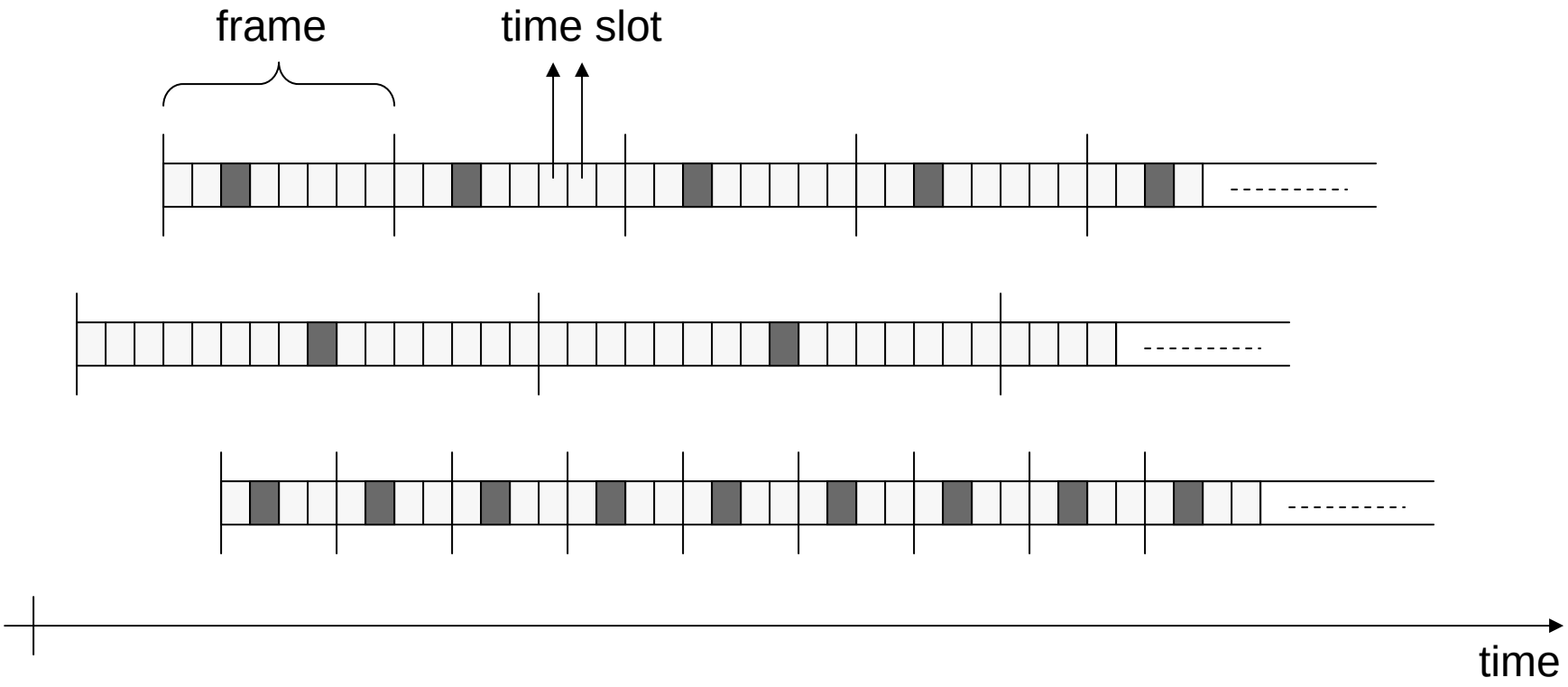
- *k*-neighborhood of a node *v*:
  $\Delta_k(v)$
- *k*-neighborhood **size** of a node *v*:
  $\delta_k(v) = |\Delta_k(v)|$
- **max** *k*-neighborhood size (in the network):
  $\delta_k = max_v\, \delta_k(v)$



- Let n be the number of nodes in the network

Our Approach

- TDMA-like framed approach



Our Approach

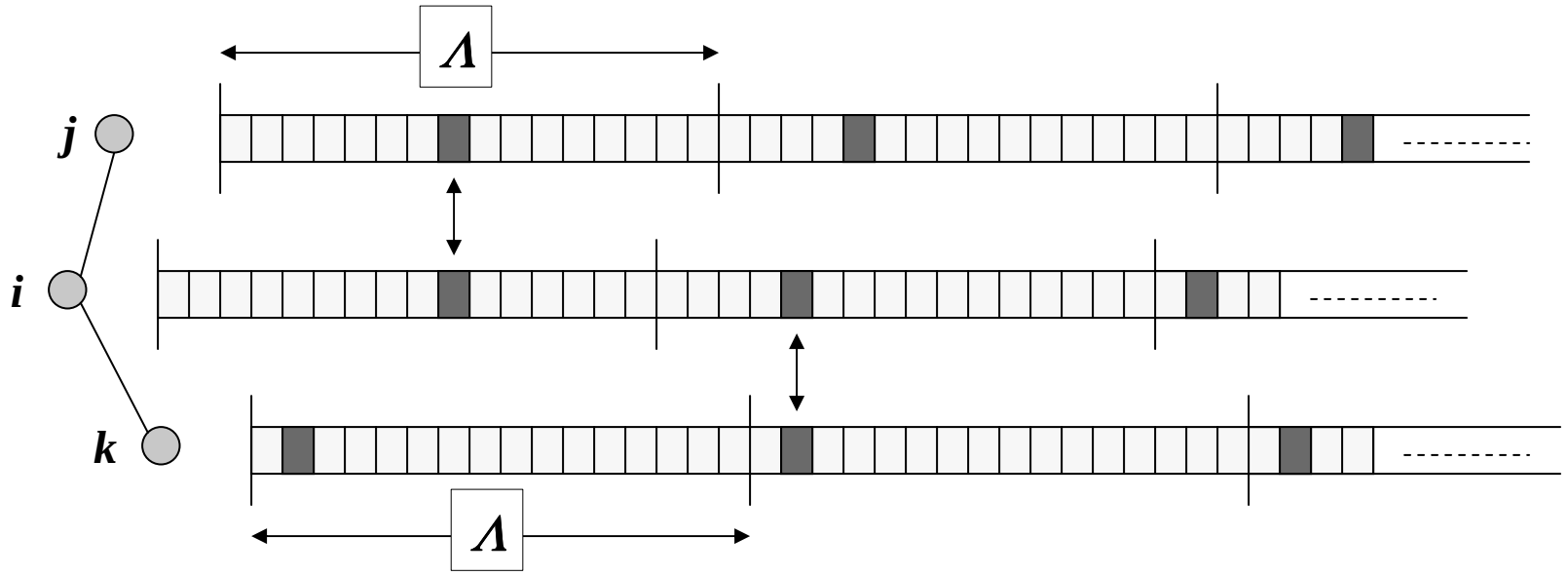
- **LooseMAC**
  - Same frame size at all nodes
  - Simple
  - Lower throughput (due to large frames)

- **TightMAC**
  - Nodes may have different frame sizes
  - More complex
  - Higher throughput (due to smaller frames)

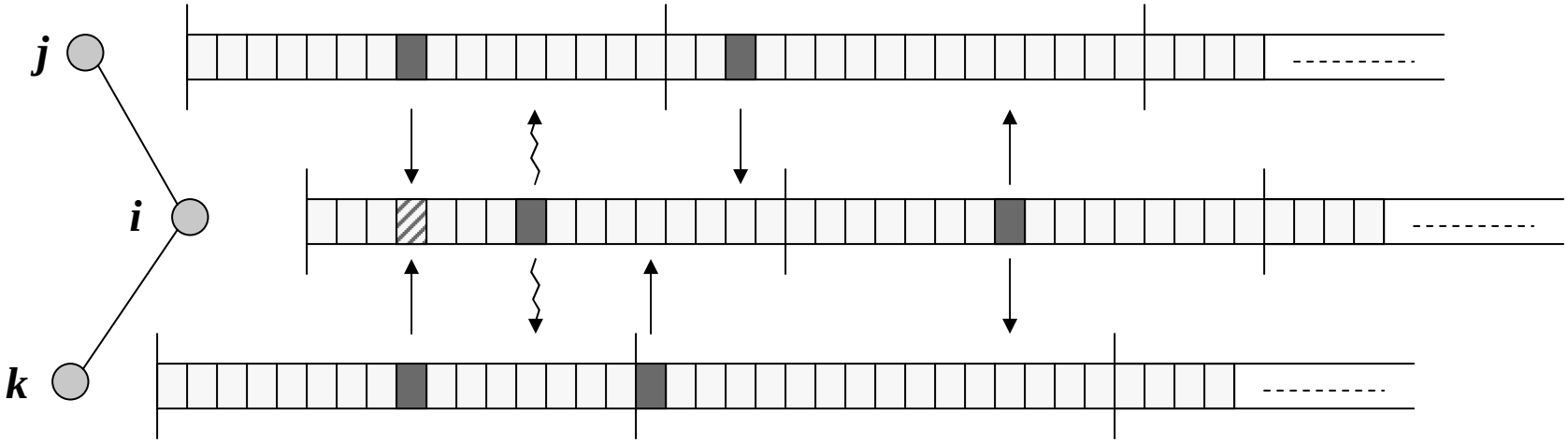
LooseMAC – Basic Idea

Schedule nodes' transmission times so that neighbor nodes do not transmit at the same time.

→ Repeatly select a **random time slot** until it is collision-free in the 2-neighborhood.



LooseMAC - Hidden Terminal Problem



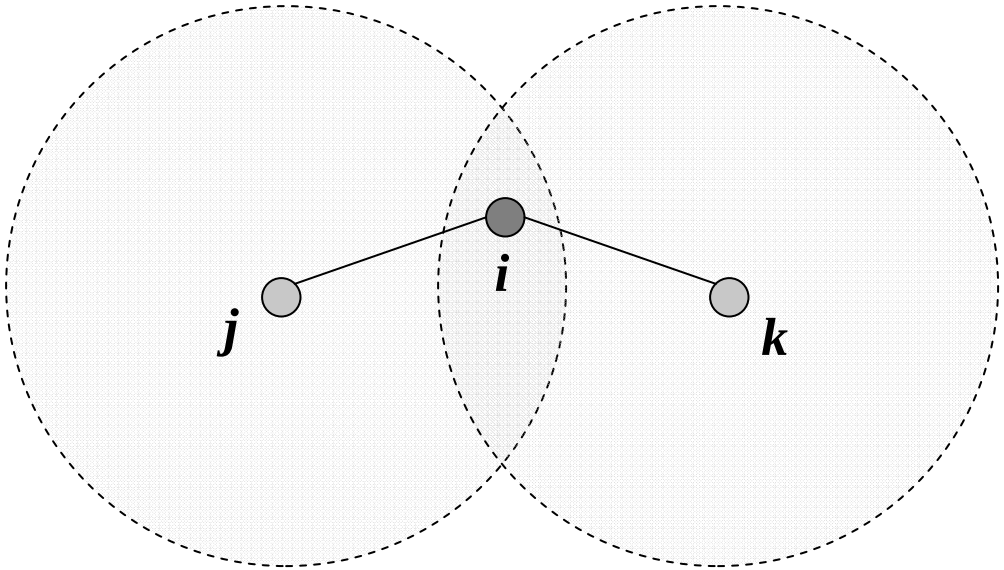
i reports the collision between j and k, so that they select new random slots.

Algorithm LooseMAC

Algorithm LooseMAC(node i)

```
1:   Divide time into frames of size Λ;
2:   ready ← FALSE;
3:   while not ready do
4:      Select a slot σ_i randomly in the frame;
5:      Send a "beacon" message in slot σ_i;
6:      Listen for a period of Λ time slots;
7:      if no collision is detected by i and no
              neighbor of i reports a conflict then
8:         ready ← TRUE;
```

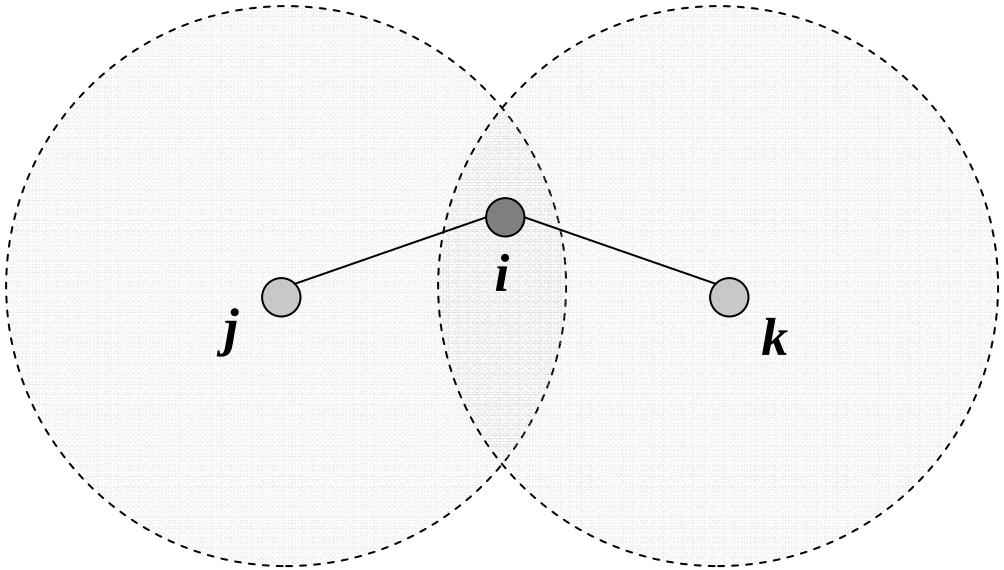
A node leaves the network…



$\sigma_j \neq \sigma_k$

No problem!…

A node joins the network…



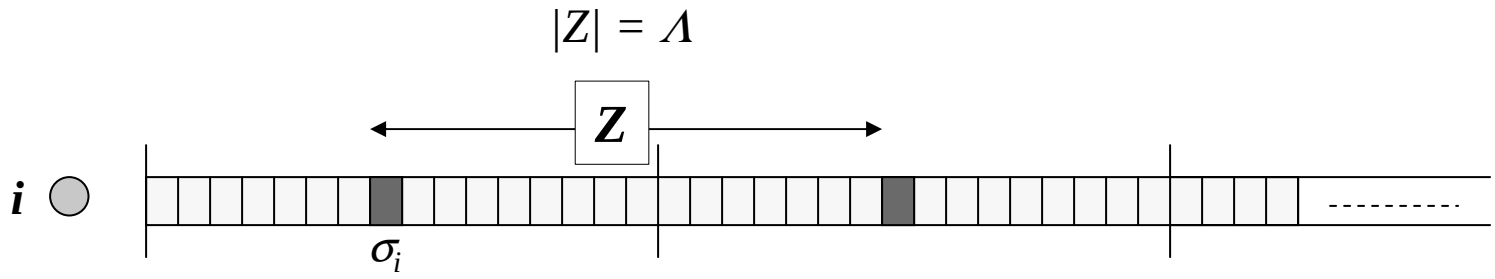
$\sigma_j \equiv \sigma_k$

Problem!... j and k are now 2-neighbors and have conflicting time slots...

…remedy: “fresh” nodes

- When a node joins the network, it is in a special status called “fresh”
- A fresh node i informs its neighbors about its status by control messages
- When a neighbor node j of i receives this message, it becomes non-ready
- We guarantee that every neighbor of i receives the “fresh” control message from node i

Probability Analysis for Convergence



i **fails** to become ready if **one or more** of the following occurs:

1. a neighbor of *i* coflicts with σ_i — p_1
2. *i* hears a collision during *Z* — p_2
3. *i* receives a conflict report during *Z* — p_3

Probability of failure = $p_1 + p_2 + p_3 \leq \dfrac{2\delta_1 + 8\delta_1^2 + 16\delta_1^3}{\Lambda} \leq c\delta_1^3 / \Lambda$ for some c

Set $\Lambda \geq 4\delta_1^3 \Rightarrow$ probability of failure ≤ 1/4

Convergence of LooseMAC

- For some constant c, if $\Lambda \geq c \min\{\delta_1^3, \delta_2^2\}$, with probability at least $1\text{-}1/n$:
  - all non-ready nodes become ready within $\Lambda \log n$ time slots
  - each node sends at most $O(\log n)$ control messages.
- Each message has size $O(\log n)$ bits:
  - sender's id ($\log n$ bits)+ fresh status (1 bit) + coflict report (1 bit)
- After convergence all transmissions are collision-free, and we define **throughput** of each node to be the inverse of its frame size; $1/\Lambda$

Algorithm TightMAC

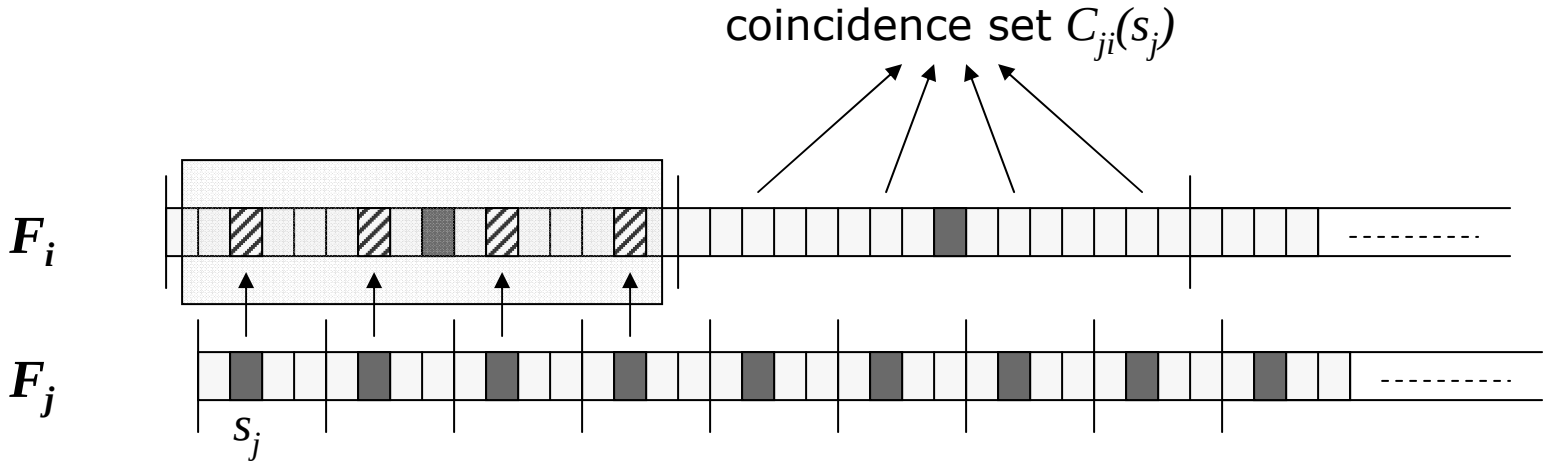
- Nodes may have different frame sizes.
- Runs on top of LooseMAC.
- Motivation: “tighten” the frames to increase throughput.

TightMAC Frame Size

- Node i selects a frame size proportional to ϕ_i, where

$$\phi_i = \max_{j \in \Delta_2(i)} \delta_2(j) \qquad \left[\text{max 2-neighborhood size among } i\text{'s 2-neighbors}\right]$$

- Each node selects a frame size which is an exact power of 2

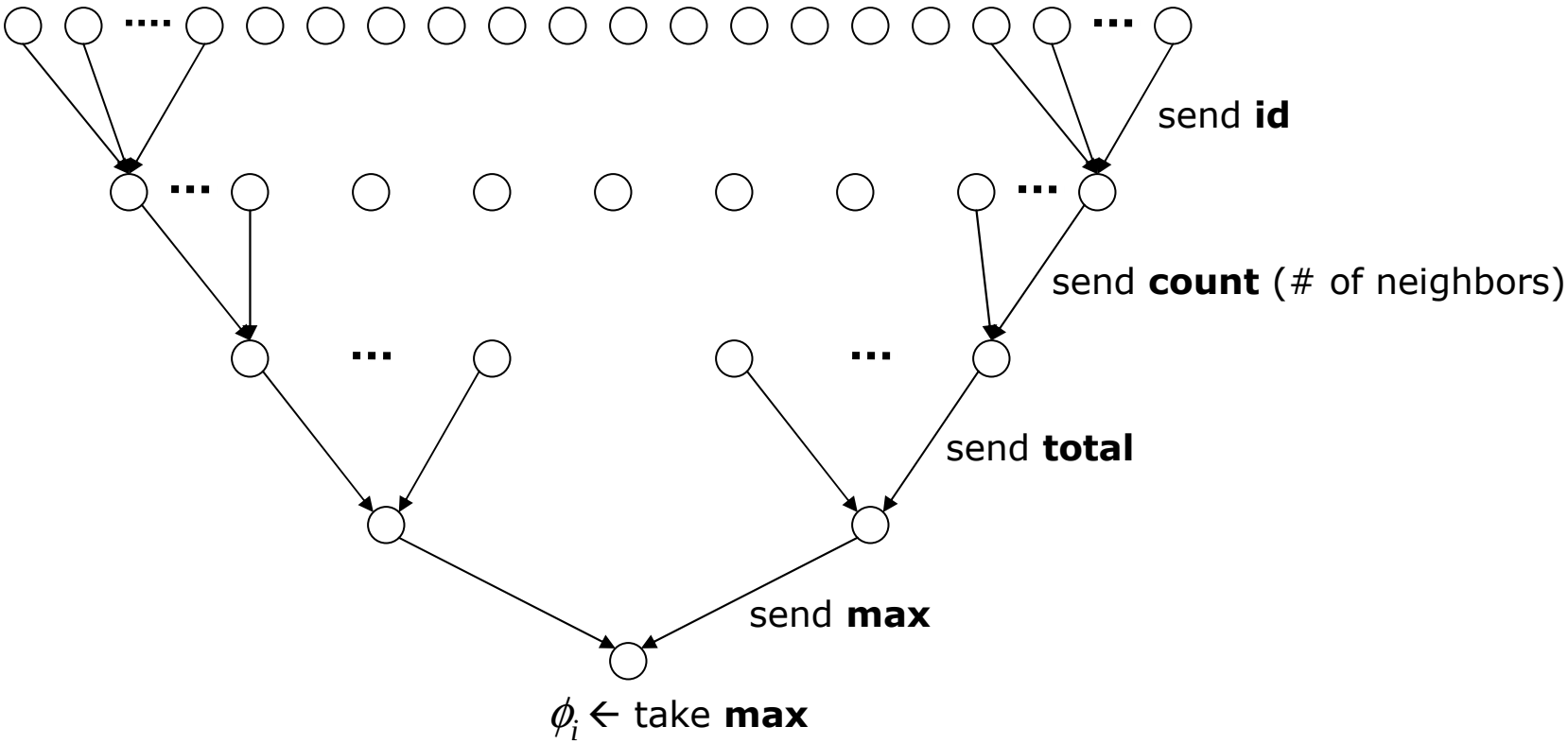


2-neighborhood size

- 2-neighborhood size calculation
  - receive ids from all neighbors, and broadcast them all
  - then a node receives ids of all 2-neighbors
  - take **union**; exact 2-neighborhood size, but high msg complexity

- Alternatively
  - receive ids from neighbors, and broadcast the count
  - then a node receives 1-neighbor counts from all neihbors
  - take the **sum**; an upper bound on the 2-neighborhood size, less msg complexity

How is ϕ_i calculated?



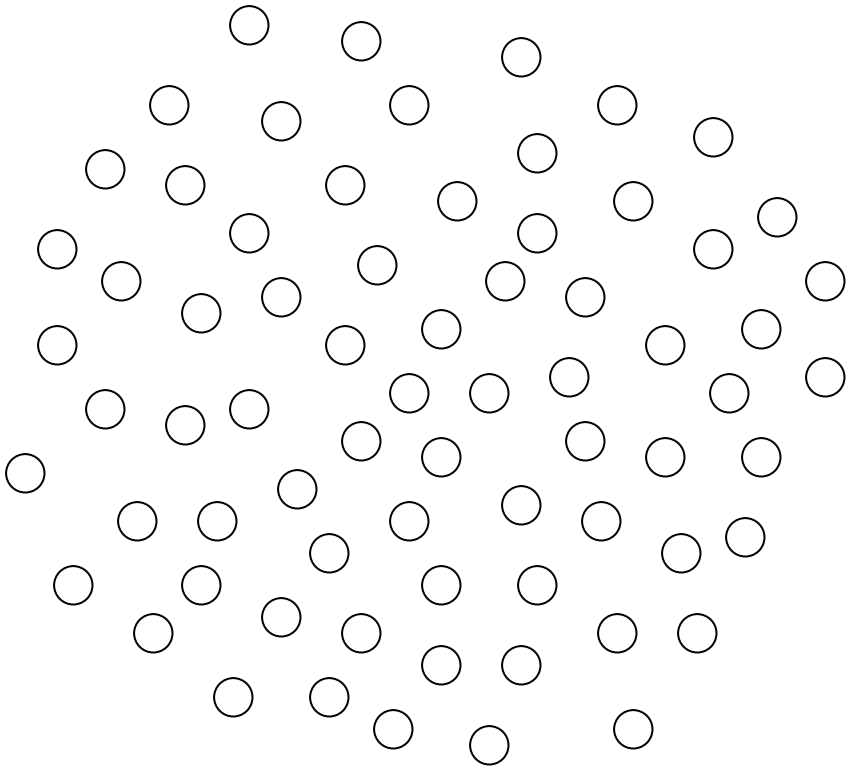
Ready levels

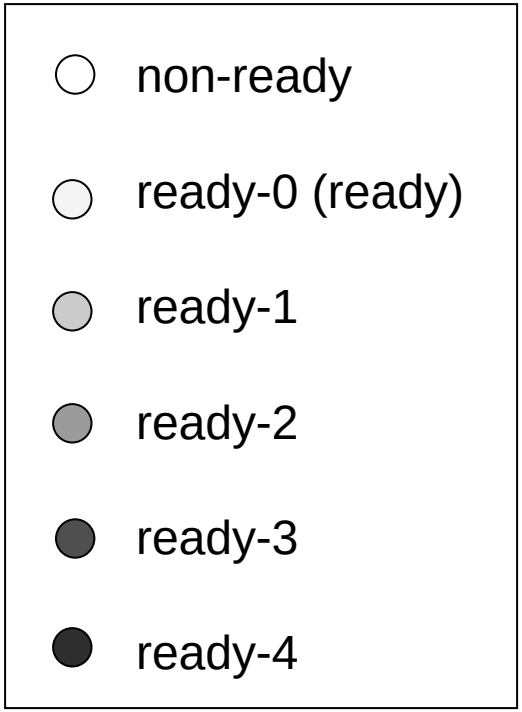
- All 2-neighbors of a node *i* should be *ready* so that *i* can proceed to TightMAC phase.

- Introduce five "ready levels";
  - ready-0 (a.k.a. ready)
  - ready-1
  - ready-2
  - ready-3
  - ready-4

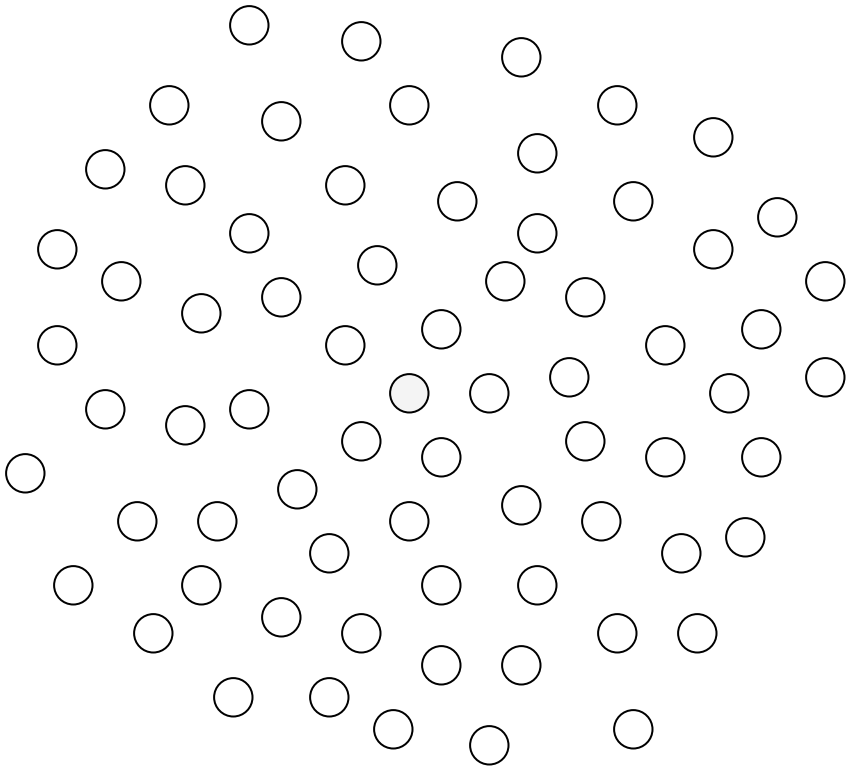
- When all neighbors of *i* are *ready-k*, *i* becomes *ready-(k+1)*.

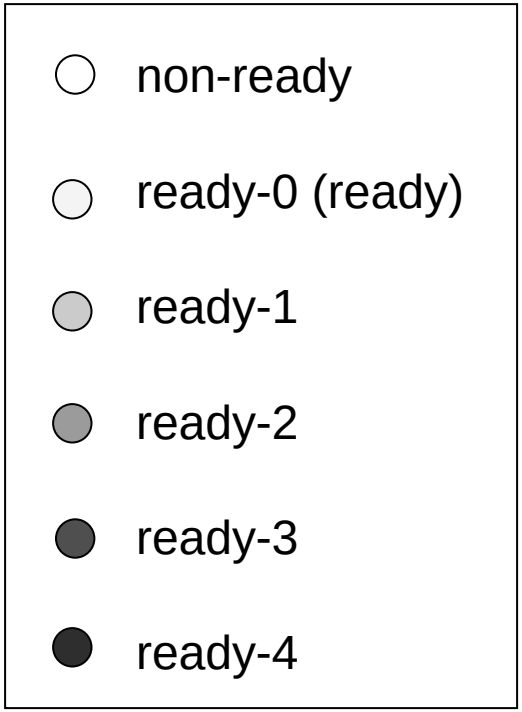
Ready levels



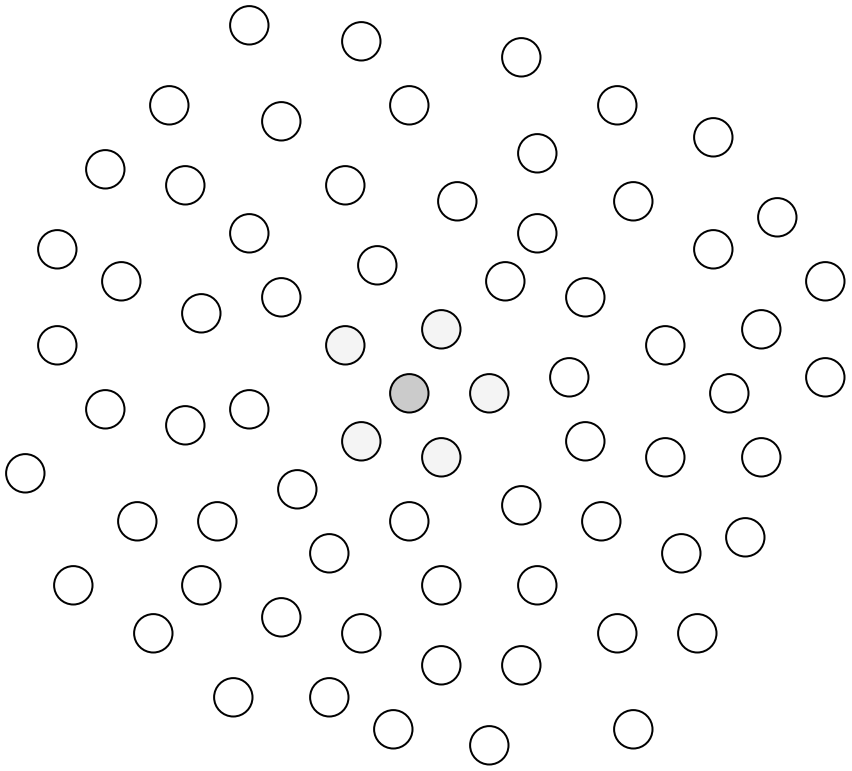


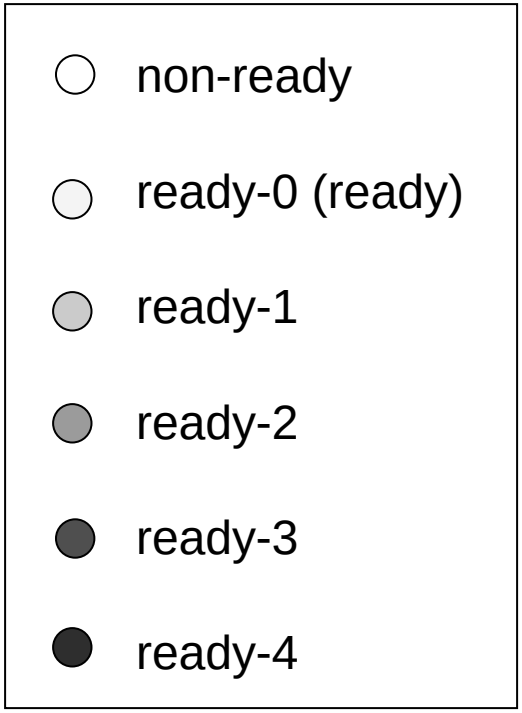
Ready levels



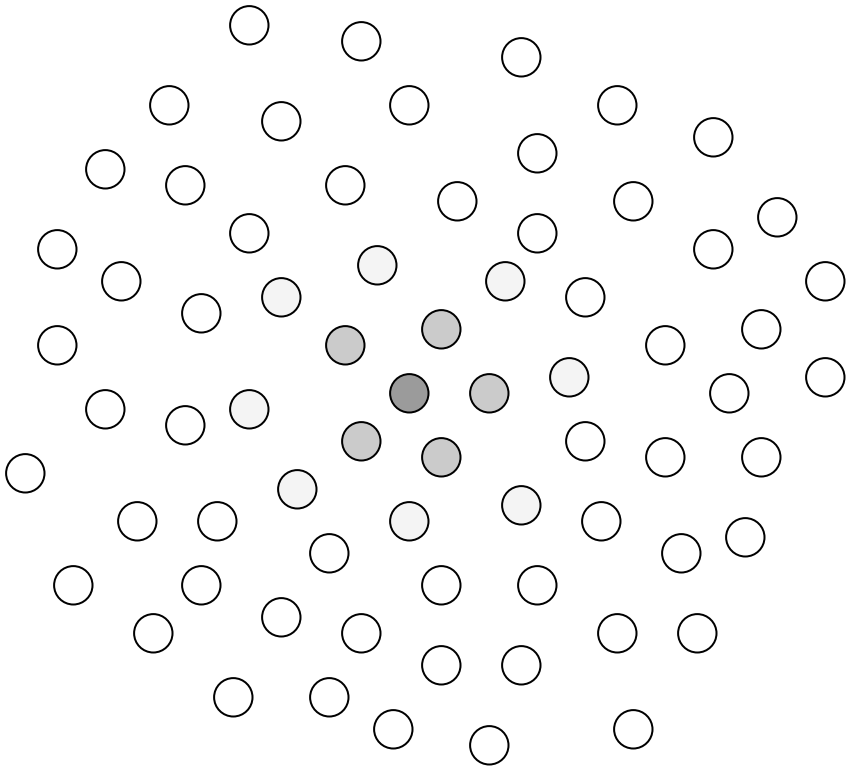


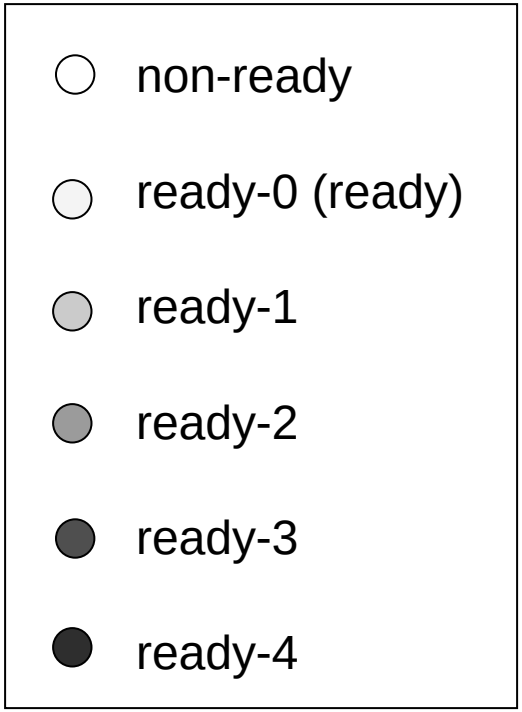
Ready levels



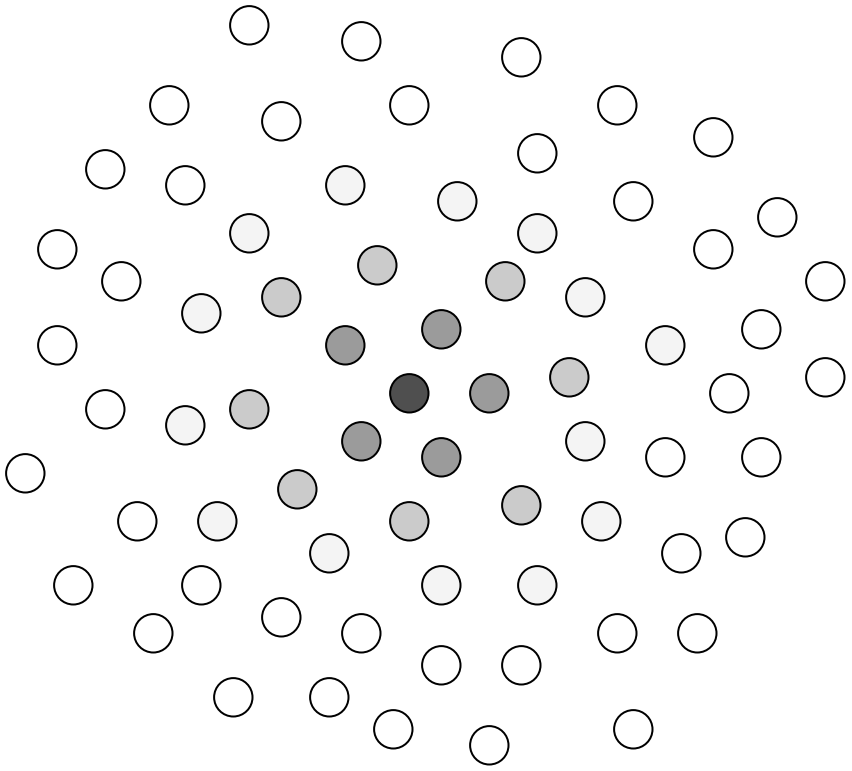


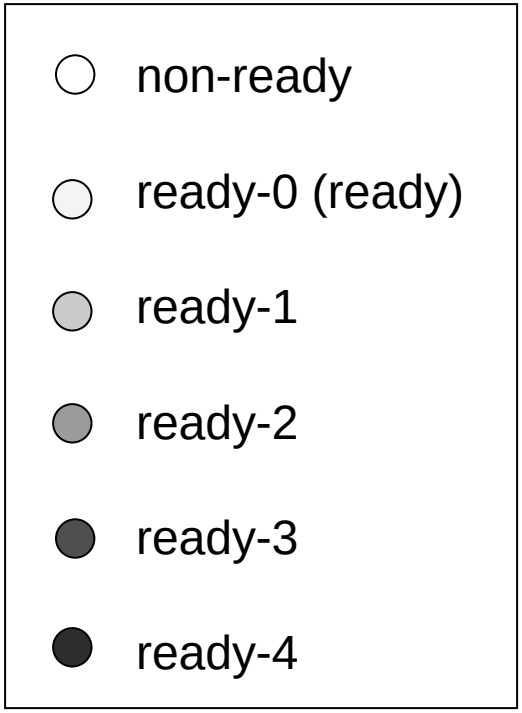
Ready levels



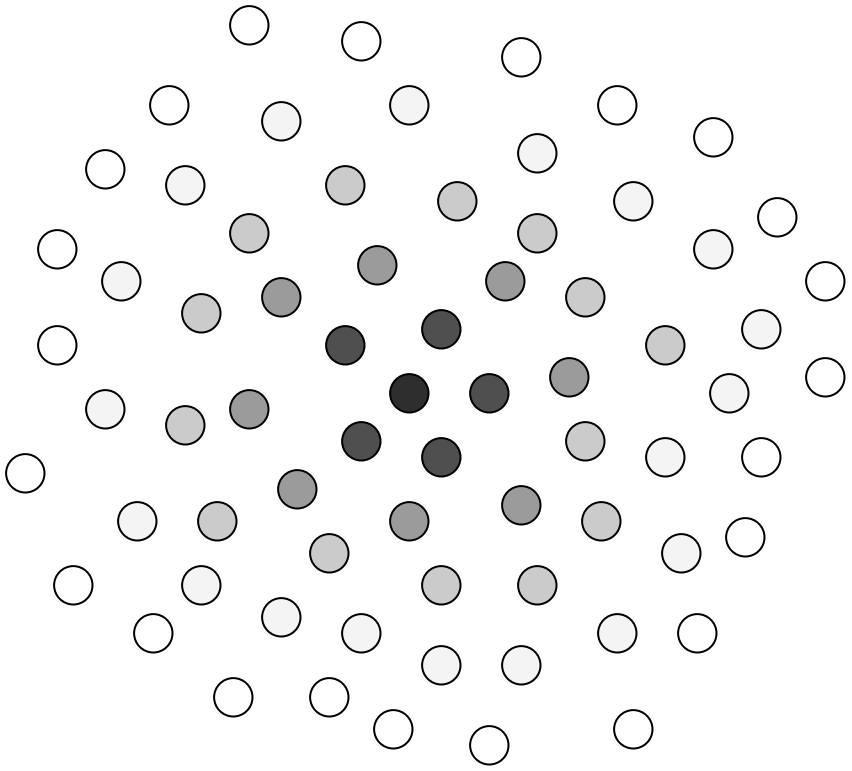


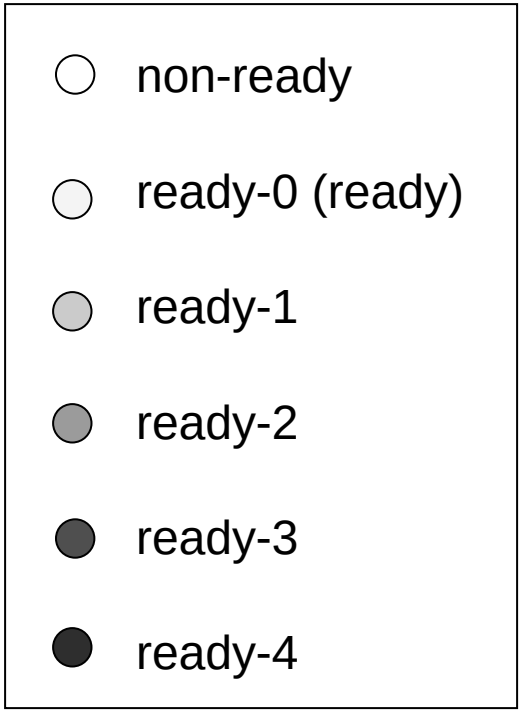
Ready levels



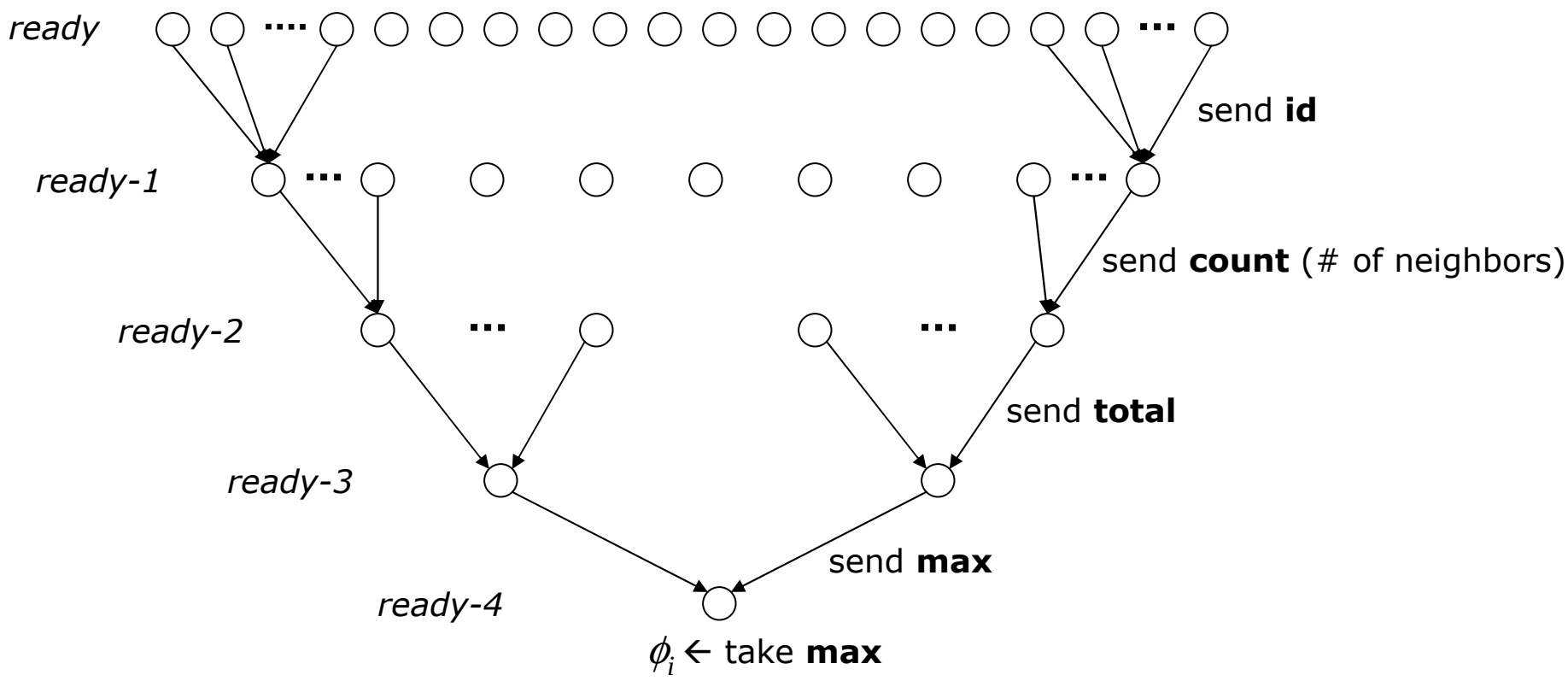


Ready levels





Ready Levels - another view



Algorithm TightMAC

Algorithm TightMAC(node i)

```
1:   repeat
2:       Execute LooseMAC(i)
3:   until i becomes ready-4
4:   Transmit neighborhood information and compute φ_i;
5:   Create the tight frame F_i with |F_i| = 2^⌈log 6φ_i⌉;
6:   Inform neighbors for the relative position of F_i, with
     respect to i's loose slot;
7:   Execute FindTightSlot(i);
8:   Start using the tight frame;
```

$|F_i| = 2^{\lceil \log 6\phi i \rceil}$ ➔ find smallest power k of 2 such that $2^k \geq 6\phi_i$

Algorithm FindTightSlot

Algorithm FindTightSlot(node i)

```
1:   while true do
2:      with probability 1/φ_i:
3:         Select a random slot s_i in F_i;
4:         Send the position of s_i (relative to its
                                          loose slot);
5:         Listen for a period of Λ time slots;
6:         if no conflict is reported by any neighbor then
7:            return s_i;
```

Complexity of TightMAC

- The network stabilizes within $O(\delta_1^2 \Lambda \log n)$ timeslots, with probability at least $1\text{-}1/\Theta(n)$
- Each node sends $O(\log n)$ messages
- Each message is of size $O(\log n)$ bits.

Practical Considerations

- How does a node detect collisions?
  - distinguish collisions from bakground noise by a threshold
- What if time slots are not alligned?
  - corectness not affected, performance affected only by a constant factor
- What about clock skew?
  - either run a simple clock skew algorithm
  - or re-run and self-stabilize whenever the skew causes collisions.

Summary and Future Work

- Presented distributed, contention-free, self-stabilizing MAC protocols for sensor networks.
  - LooseMAC
  - TightMAC
- Future research directions
  - Simulation analysis (e.g. for analyzing the effects of topology change rate)
  - Compare the performance with existing protocols

References

- C. Busch, M. Magdon-Ismail, F. Sivrikaya, B. Yener, "*Contention-Free MAC protocols for Wireless Sensor Networks*." Technical Report, Rensselaer Polytechnic Institute, 2004. Available at http://www.cs.rpi.edu/research/tr.html.

- N. Abramson, "*The ALOHA System - Another Alternative for Computer Communications*." Proceedings of the AFIPS Conference, vol. 37, pp. 295-298, 1970.

- "*Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*." IEEE standards 802.11, January 1997.

- W. Ye, J. Heidemann, D.Estrin, "*Medium Access Control with Coordinated, Adaptive Sleeping for Wireless Sensor Networks*." [SMAC] IEEE/ACM Transactions on Net-working, vol. 12, no. 3, pp. 493-506, June 2004.

- A. Cerpa, D. Estrin, "*ASCENT: Adaptive Self-configuring Sensor Network Topologies*." Proceedings of INFOCOM'02, 2002.