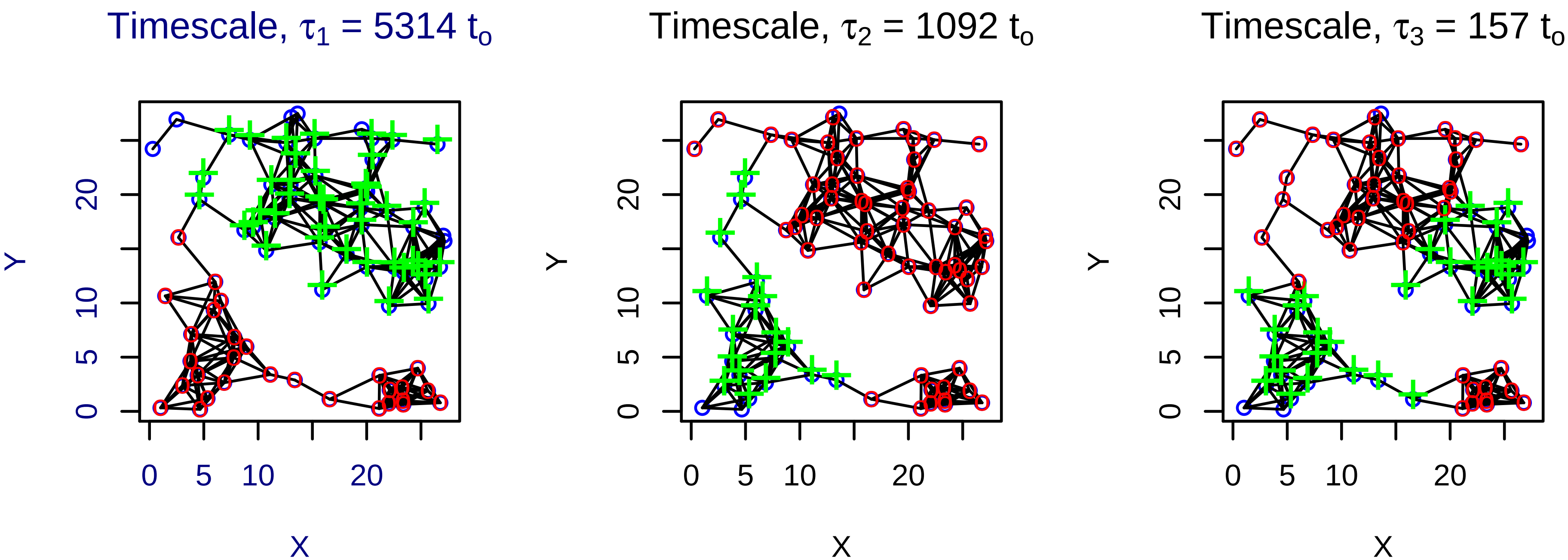
MAE 298, Lecture 8

April 27, 2006



“Spectral Methods, Sensor Nets and Self-organization”

Last time: spectral methods, eigen-spectrum

- If two distinct graphs have the same eigen-spectrum, they are likely isomorphic (esp for large graphs).

- Eigenvalues: degeneracy of $\lambda = 1$ tells us how many disconnected components in the graph.

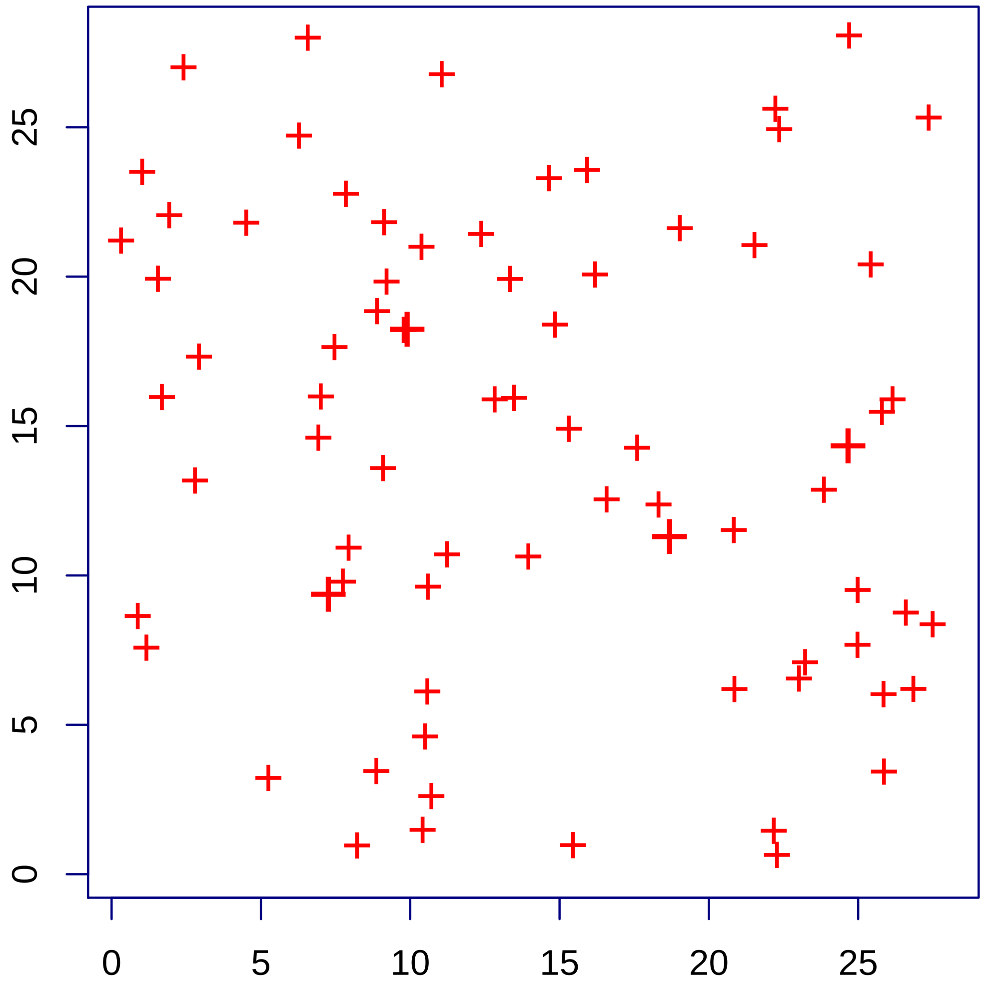
Summary: spectral methods, measures

- Mixing time (time to forget where the walk started)
- Relaxation time (related to mixing time, gives bounds)
- Cover time (time to occupy each node)
- Spectral gap: the largest mixing time, $t_{\max} = -1/\ln(\lambda_2)$

  – the larger $t_{\max}$ the longer it takes for a random walk to cover the graph.

  – the larger $t_{\max}$ the more accurately a graph can be partitioned into two pieces.

Applications: Wireless sensor networks



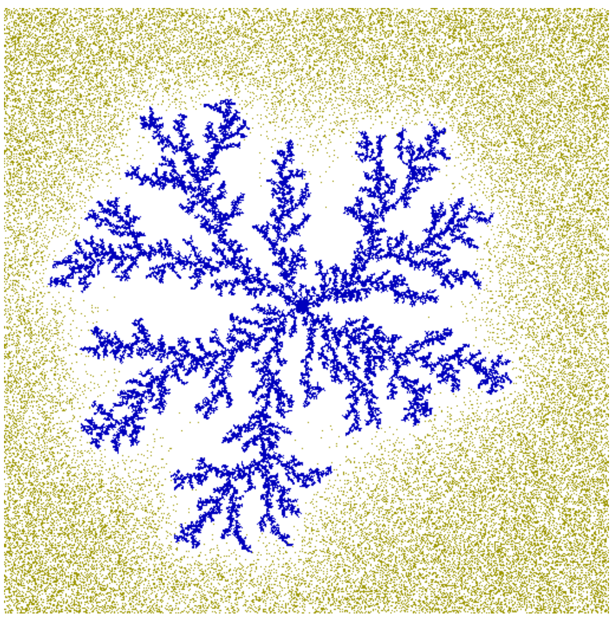
- Start with isolated sensor distributed at random.
- Is there a *local* way to build up global connectivity?
- Locality — why?

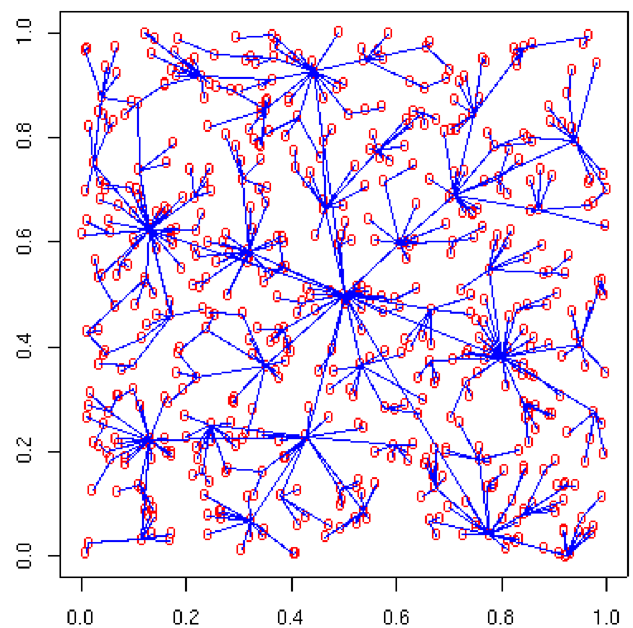
Locality

1. Locality $\sim$ distributed
2. Adapt quickly to changing environment
3. Minimal growth in overhead with increasing system size
4. "Self-organizing"

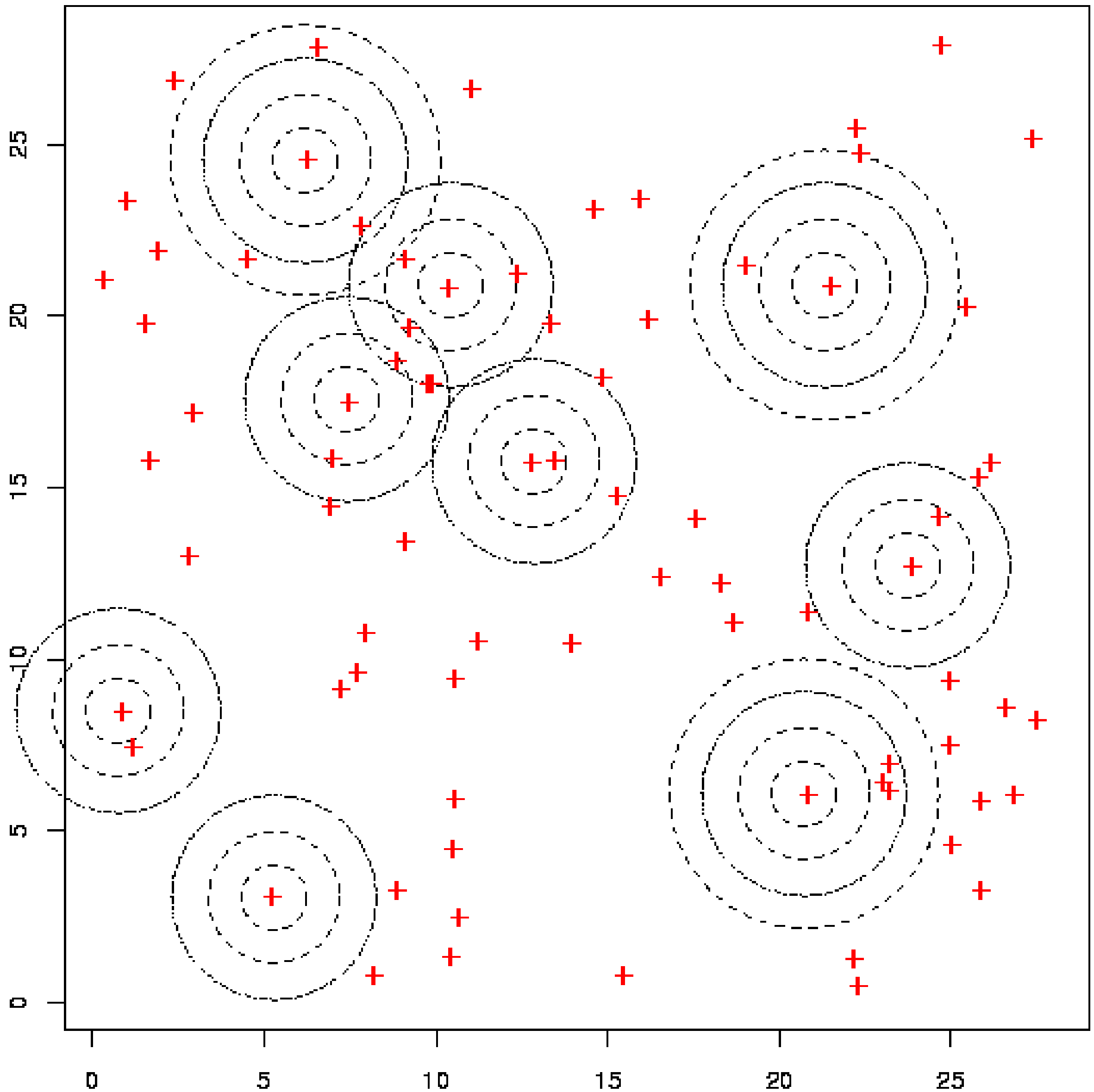
"self-organization"

- Not quantitatively defined.
- (Wikipedia:) Self-organization is a process in which the internal organization of a system , normally an open system , increases in complexity without being guided or managed by an outside source. Self-organizing systems typically (though not always) display emergent properties.



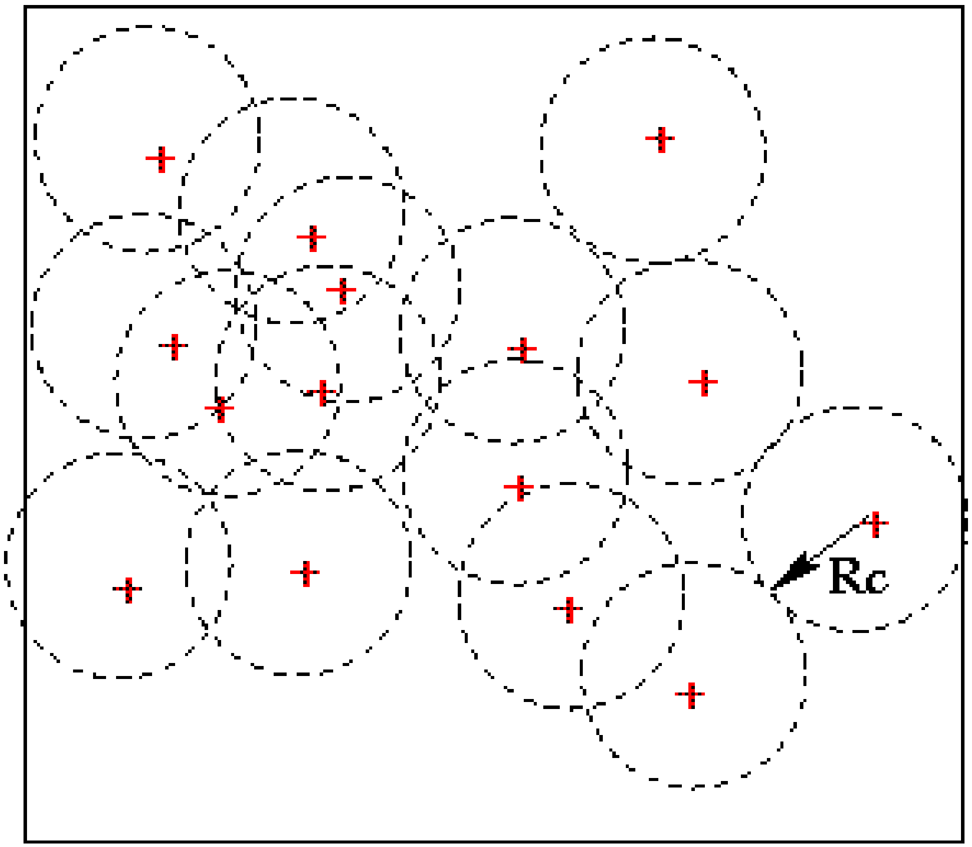


Beaconing



A geometric graph problem

One idea — percolation

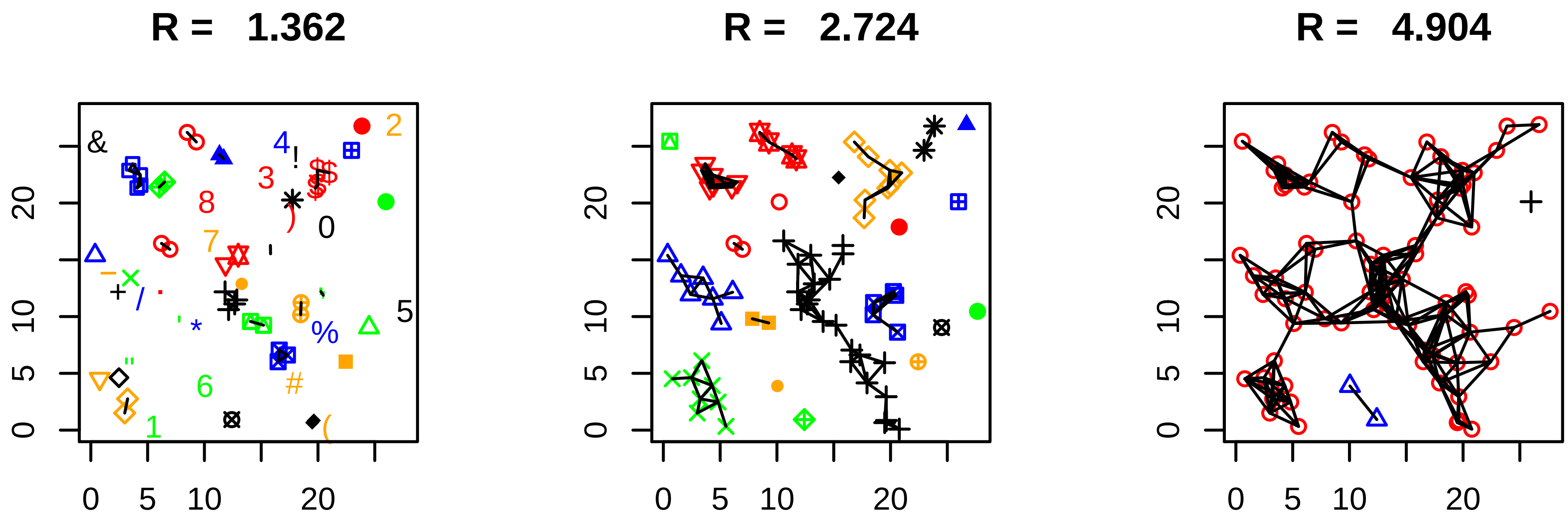


Call the graph describing connectivity of nodes: G_R

Is this a local algorithm?

(How to determine R_c?)

How to determine R_c?



Keep increasing until only one eigenvalue $\lambda = 1$

Percolation

Why is it bad?

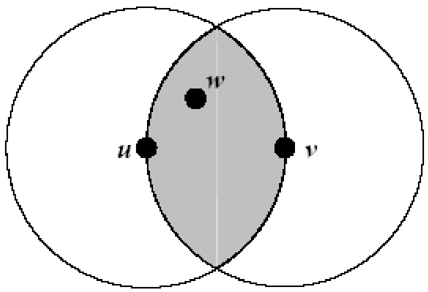
- Farthest away node sets operating power for all
- Need to *communicate* this value R_c (critical operating range)
- Assumes wireless footprint a uniform disk

Why is in good?

- Guarantees full global connectivity. In the asymptotic limit ($N \to \infty$) know how R_c scales with N. So for large N can use theoretical estimate rather than $\lambda = 1$ construction.
- Want small range R to conserve power and also reduce interference. Percolation is a "sweet spot" (full connectivity with out too much interference).

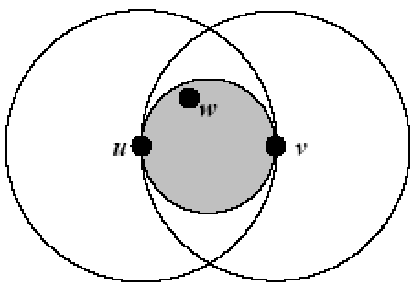
Refining percolation graph G_R (also called the "unit disk graph")

RNG & GG



Relative Neighbor Graph (RNG)

- An edge (u, v) exist if $\forall$w, d(u,v) $\leq$ max(d(u, w), d(w, v))

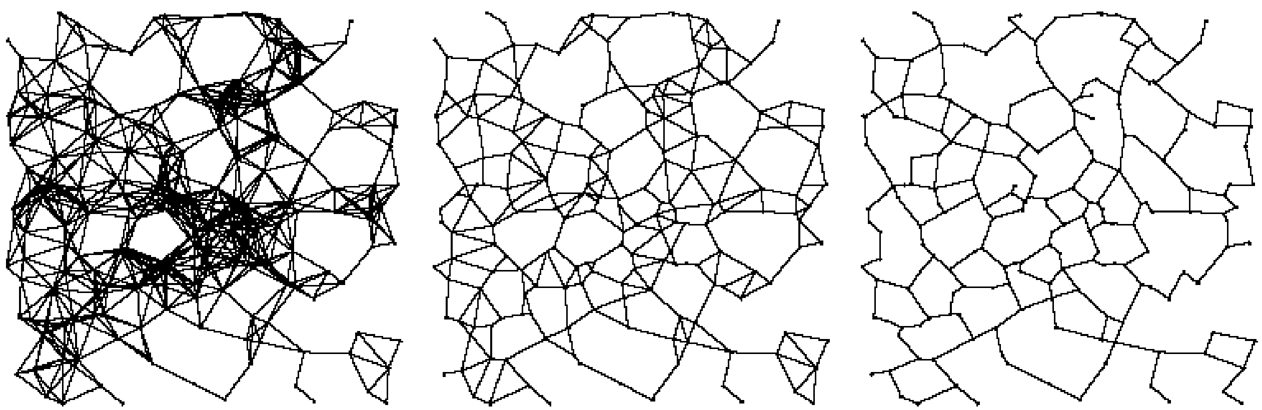


Gabriel graph (GG)

- An edge (u, v) exist, if no other vertex w is present within the circle

$$\forall w \neq u, v : d^2(u,v) \leq d^2(u,w) + d^2(w,v)$$

Planarized Graph (Cont'd)



Original Unit Disk Graph

GG

RNG

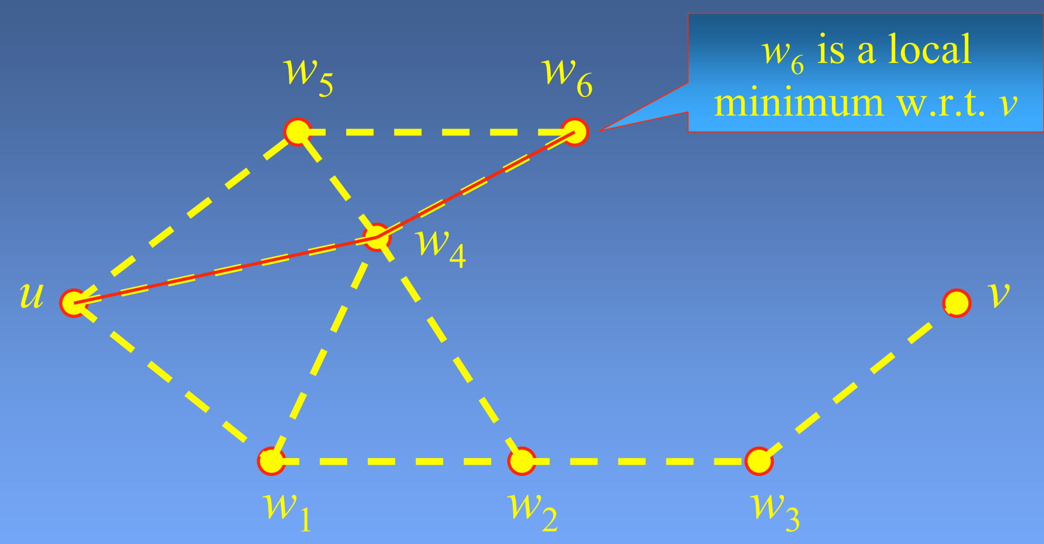
- Preserve connectivity of G_R, but sparser (and also planar).

- There is an R-package that computes the disk, Gabriel and relative neighbor graph given an input set of coordinates (http://rss.acs.unt.edu/Rdoc/library/spdep/html/graphneigh.html)

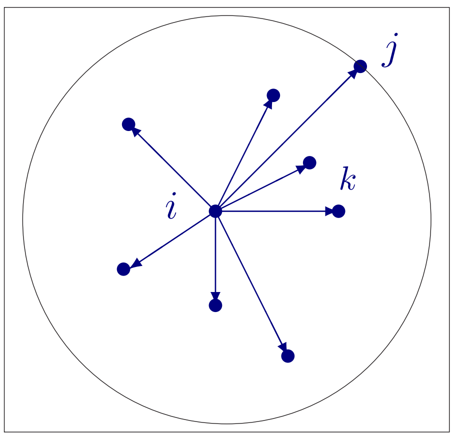
Planar disk graphs $\Longrightarrow$ greedy routing

- What is greedy forward routing?
- Packets are discarded if there is no neighbor which is nearer to the destination node than the current node; otherwise, packets are forwarded to the neighbor which is nearest to the destination node.
- Each node needs to know the locations of itself, its 1-hop neighbors and destination node.
- Pros: easy implement
- Cons: deliverability (stuck in local voids)

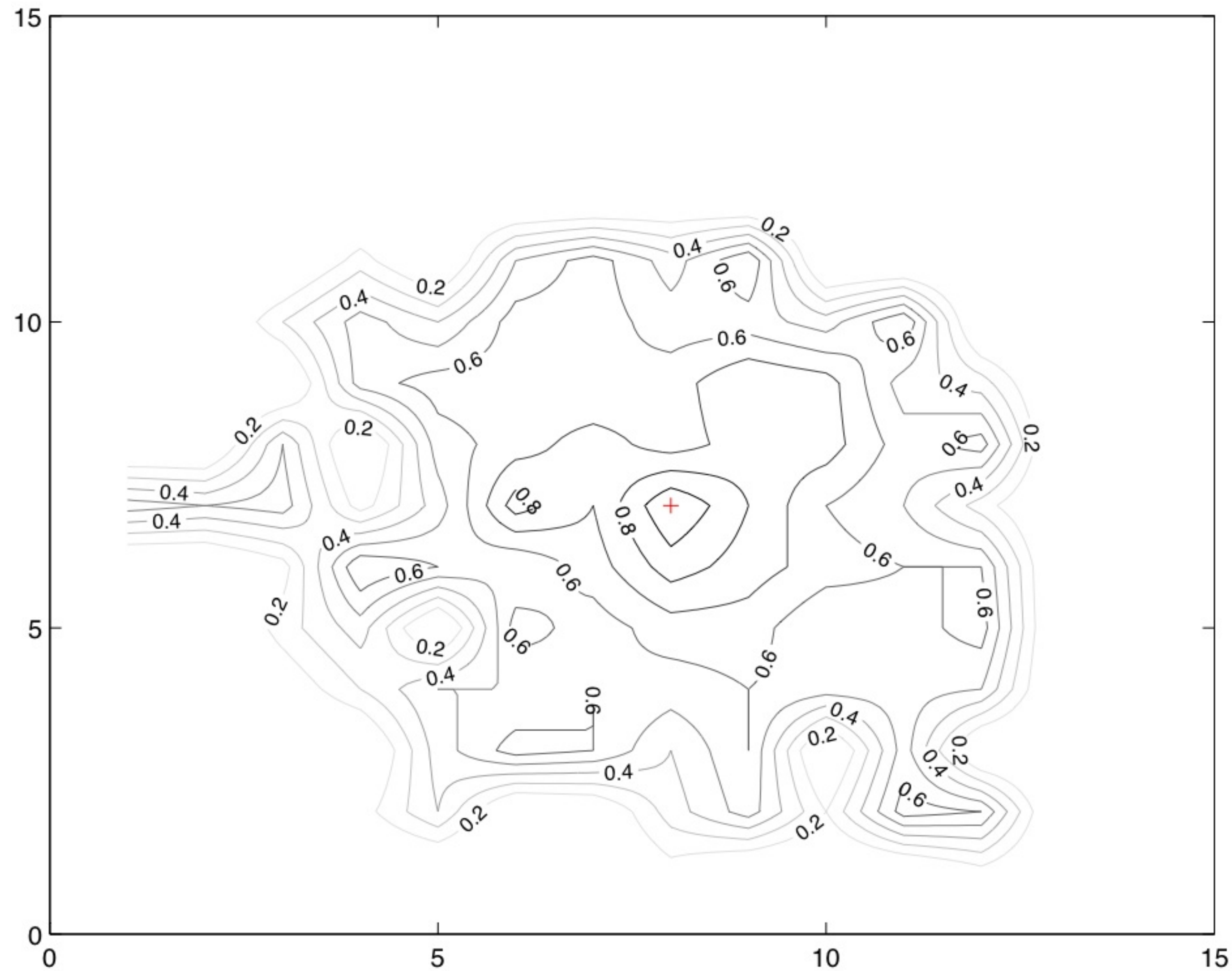
Examples



Theory versus reality
Wireless footprints



Monotonicity

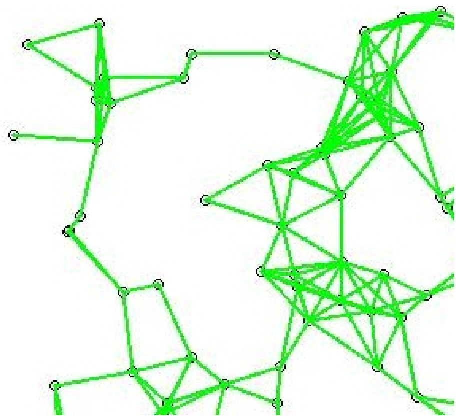


Uniform-disk model

Experimental
[Ganesan, et. al., 2002]

Adaptive Power Topology control
(Local algorithms to build connectivity)

- Percolation (Common power) neglects natural clustering.
  - Too much power consumption and unnecessary interference.
  - Misses certain paths which could optimize traffic.

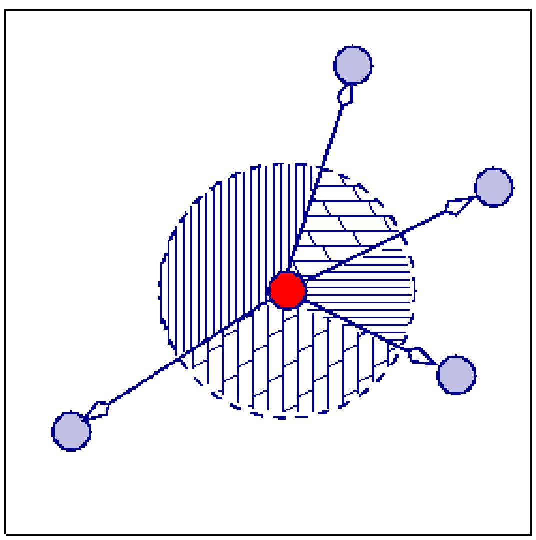


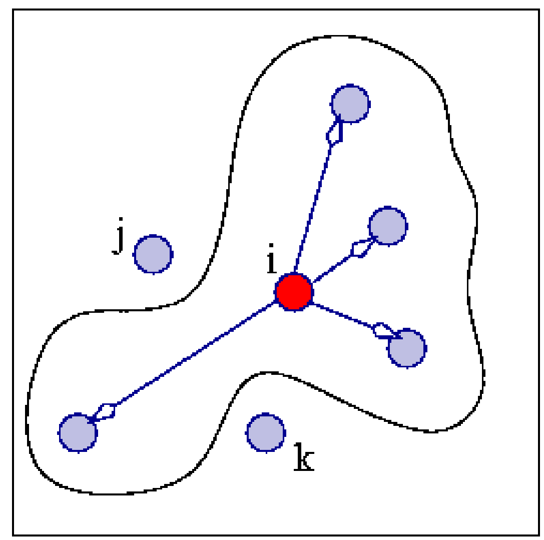
- How to build up a connected network using only local information?
  - Moreover, want to avoid uniform disk requirement

Adaptive Power Topology control

- Adaptive power topology control *(APTC)*
  [Wattenhofer, Li, Bahl, and Wang. Infocom 2001]
  [D'Souza, Ramanathan, and Temple Lang. Infocom 2003]

Each node *individually* increases power until it has a neighbor in every θ sector around it:

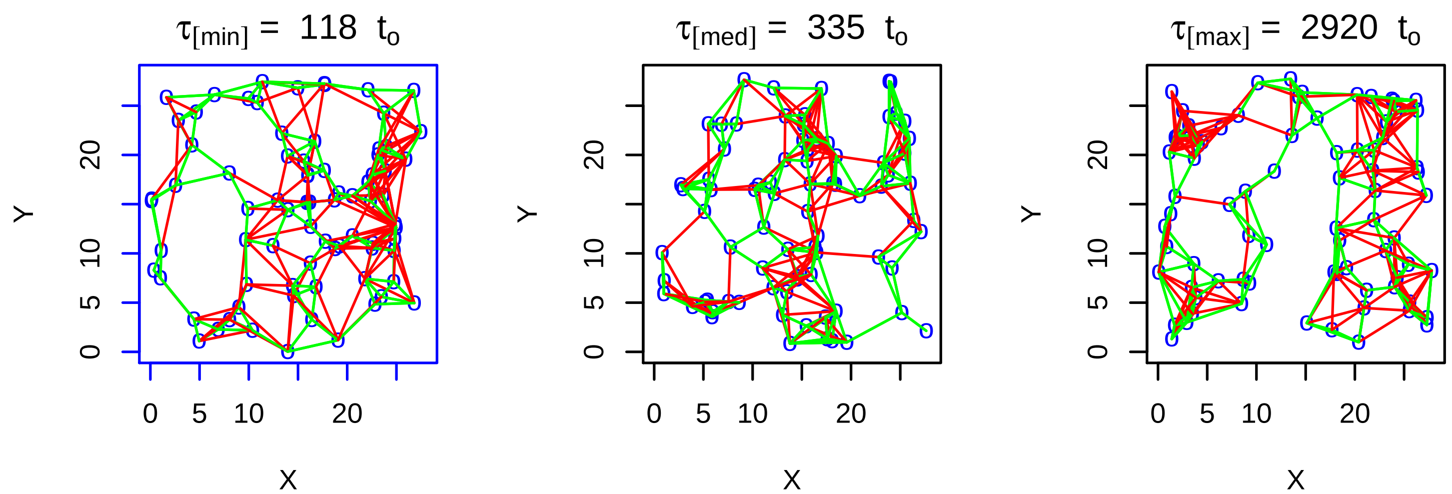




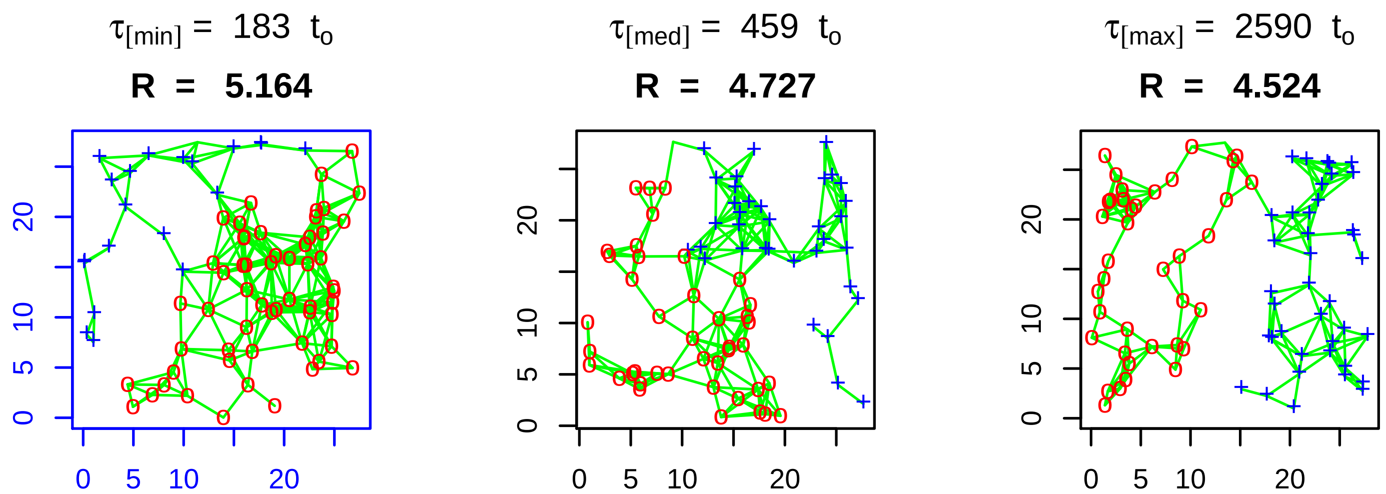
Call the graph describing connectivity of nodes: G_θ

Sample topology

Topology control:



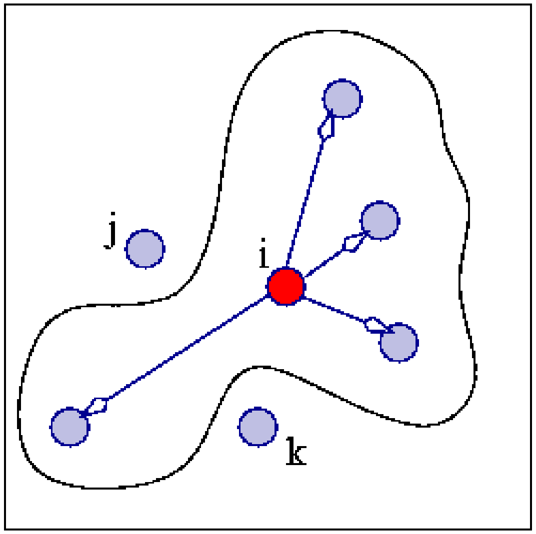
Percolation:



Beyond the uniform disk model

[D'Souza, Galvin, Moore, Randall, IPSN 2006]

- Can use a local geometric θ-constraint to ensure full network connectivity, *independent of wireless footprint!*.

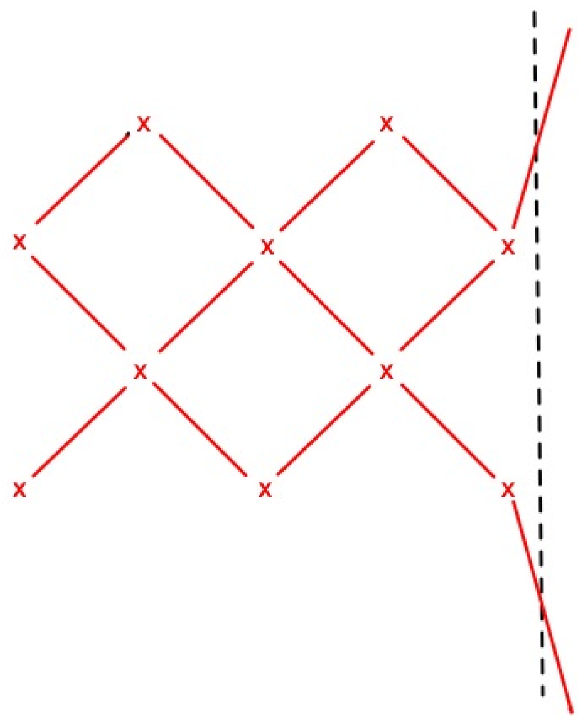


- Requires constraints on boundary nodes. (Carefully deploy boundary nodes so can communicate, or else have hard-wired boundary channel; then interior nodes can be scattered haphazardly).

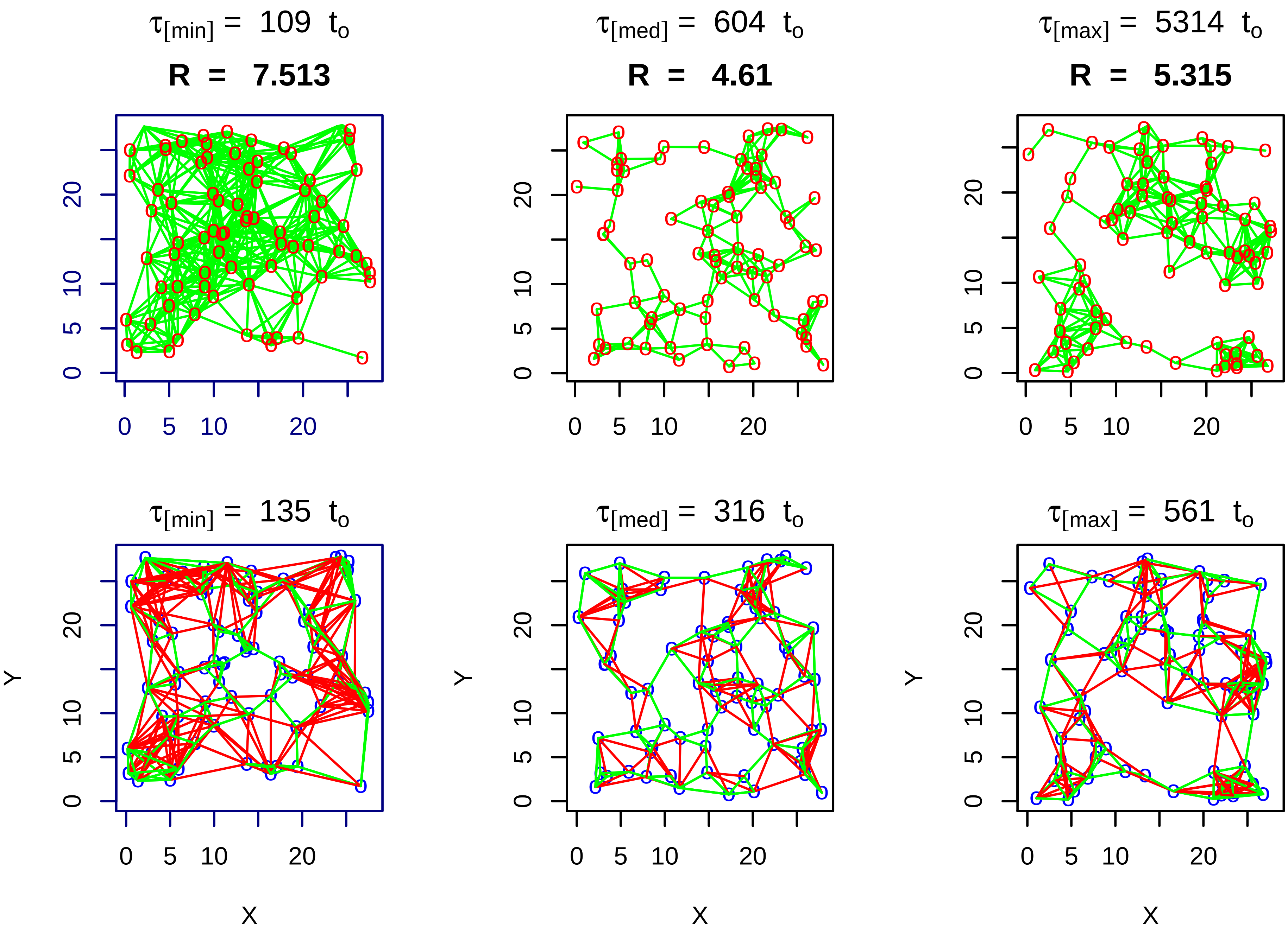
Proof overview

Theorem 1. *If G_θ satisfies the θ-constraint at every internal node with $\theta < \pi$ and all of the boundary nodes are known to be connected, then G_θ is fully connected.*

Proof: We need only show every internal node v has a path in G_θ to some node on the boundary.

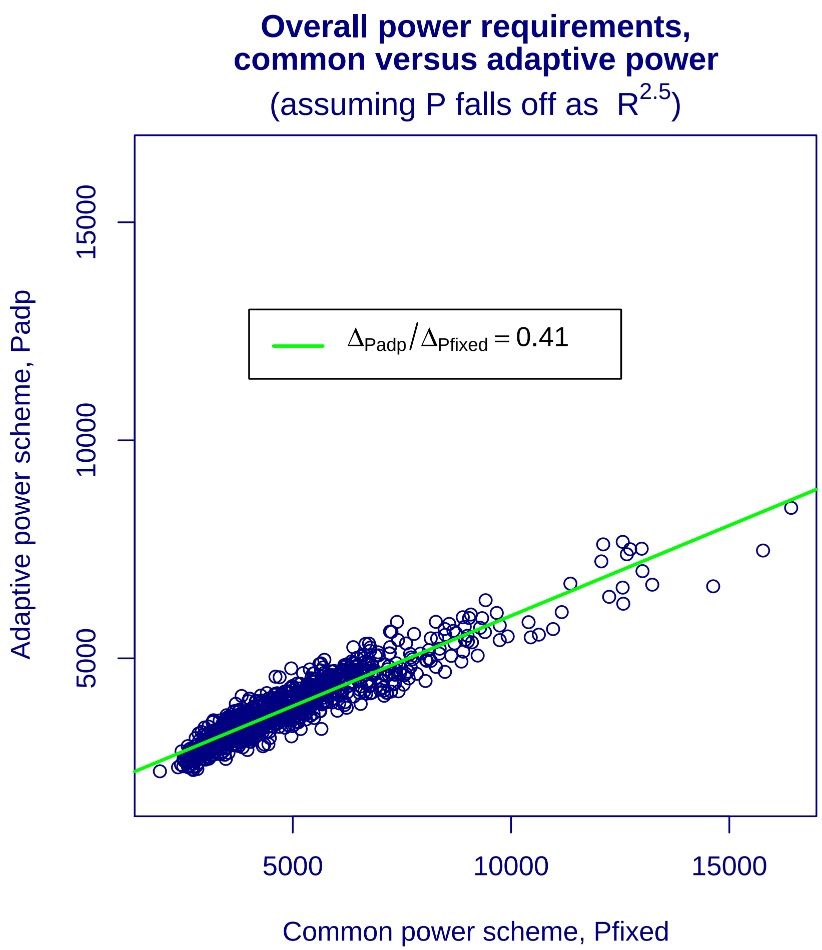


Comparison to percolation scheme



How to quantify "better"?

- Need performance metrics
- Direct measures: energy consumption



Power Control: Cross-Layer Design Issues

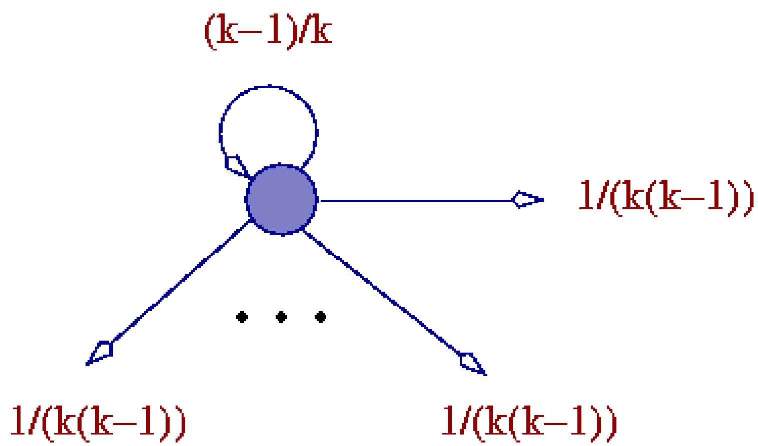
- Physical Layer
  - Power control affects quality of signal
- Link Layer
  - Power control affects number of clients sharing channel
- Network Layer
  - Power control affects topology/routing
- Transport Layer
  - Power control changes interference, which causes congestion
- Application/OS Layer
  - Power control affects energy consumption

The “protocol stack”

In general, networks layered

- Social networks
- $\longrightarrow$ Email networks
- $\longrightarrow$ Data networks
- $\longrightarrow$ Protocol networks
- $\longrightarrow$ Physical networks

Approximating interference



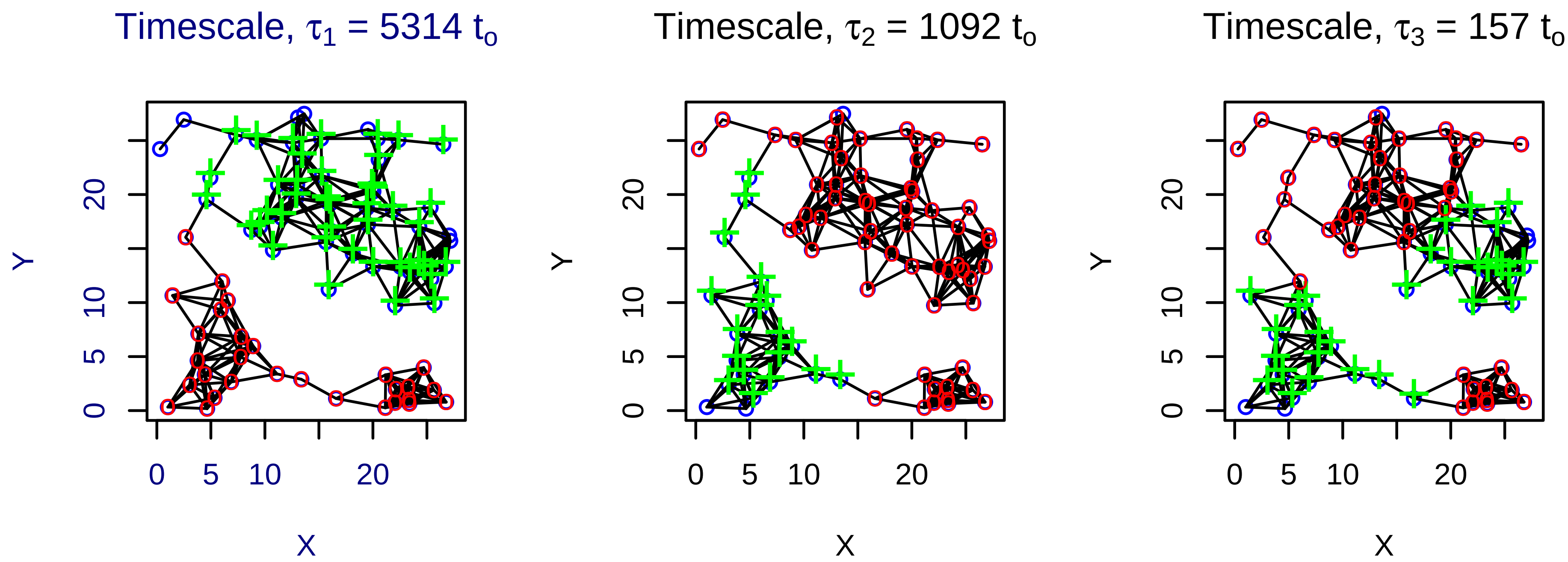
State transition matrix:

$M_{ii} = (k_i - 1)/k_i$, for diagonal elements.

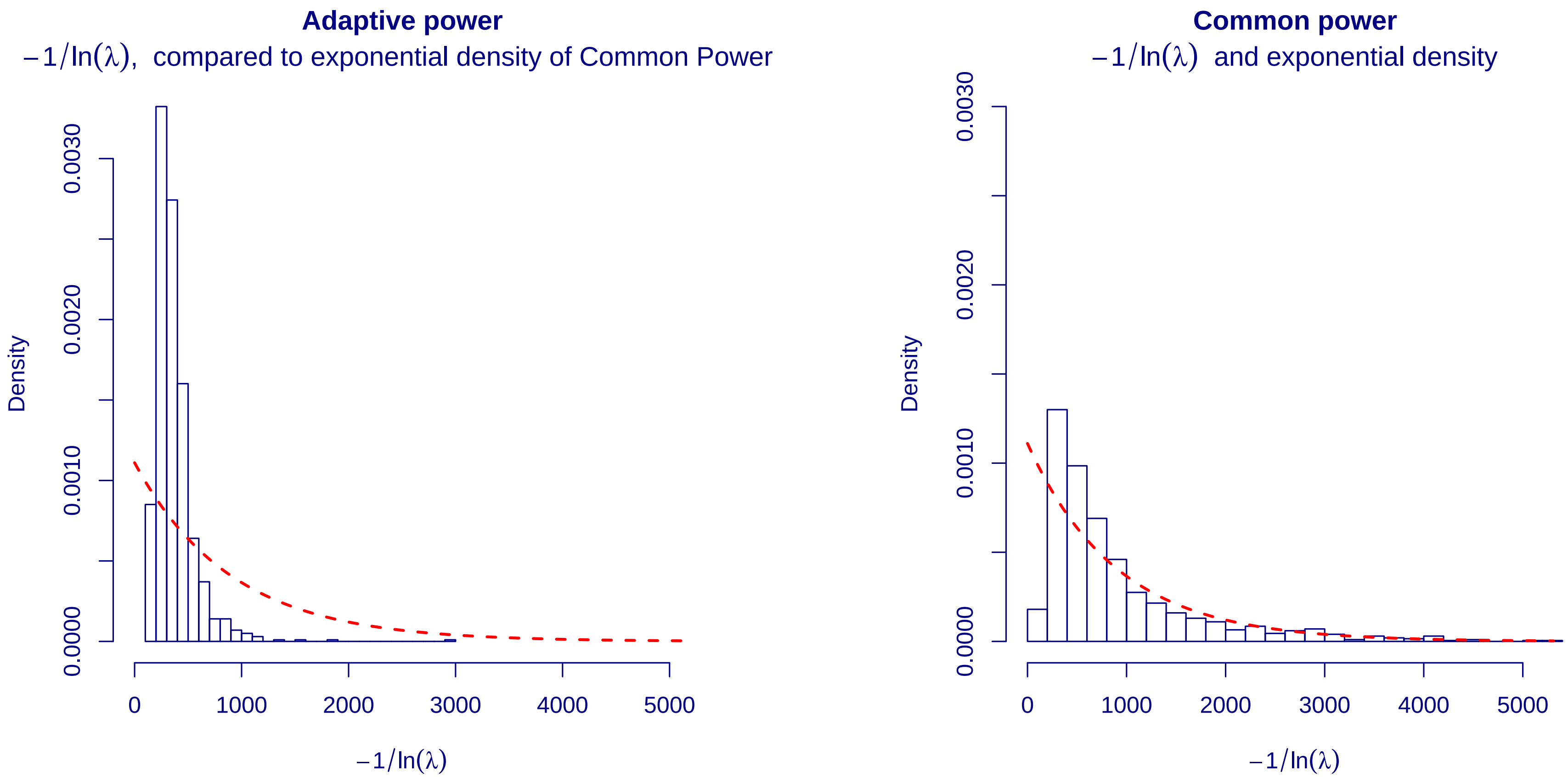
$M_{ij} = 1/(k_i - 1)k_i$, if an edge exists between i and j.

Network self-discovery time
(Using mixing time as a proxy)

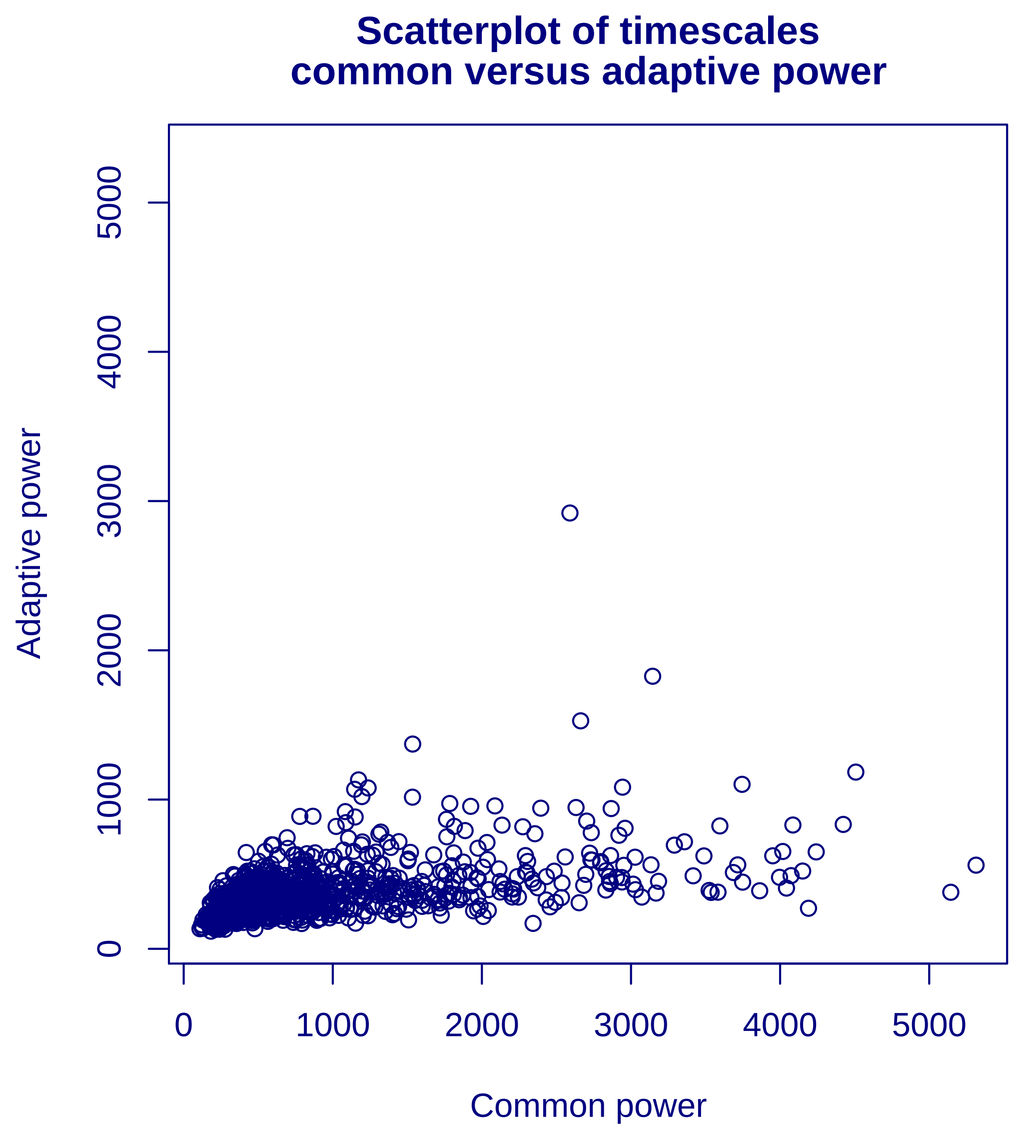
$$\boxed{\tau = -1/\ln(\lambda_2)}$$


Timescale, τ1 = 5314 to
Timescale, τ2 = 1092 to
Timescale, τ3 = 157 to
X
Y
0
5
10
20

Comparison of timescales



Comparison of timescales



Regimes for routing

Two timescales:

1. $t_{info} = -1/\ln(\lambda_2)$

2. $t_{network}$: time for network *topology* to change.

Routing:

- If $t_{network} \ll t_{info}$, network essentially static during packet routing, so build up routing information.

- If $t_{network} \gg t_{info}$, any info on the network topology will be immediately obsolete, so no routing strategy.

Is there a sharp threshold? Or even any way to bound these behaviors? (What routing protocols work best in which regimes?

Other pressing issues: Sensor networks

- Deployed networks: optimal sensor placement
- Gossip algorithms: spreading shared information quickly through local exchanges.