Routing in Variable Topology Networks

Dr. Alhussein Abouzeid

Electrical, Computer and Systems Engineering (ECSE)

Rensselaer Polytechnic Institute

Variable Topology Networks

- P2P networks for distributed resource sharing: File sharing: Gnutella, Freenet, ..etc. or CPU sharing: Globus, BlueGrid.
  - Topology changes: resources (e.g. peers, files, etc.) appear/ disappear.
  - Find (discover) where the resource is.
  - Allocate "jobs" to "resources" so as to maximize/minimize some performance measure.

- Sensor/actuation networks
  - Topology changes: unreliable nodes (turn ON/OFF randomly) and random dynamic environment (e.g. winds).
  - Need to disseminate information to/from the sensors/sinks.

- Mobile ad-hoc networks
  - Topology changes: continuous node movement
  - Need to discover a peer/destination.

Single-Path Routing in Ad-hoc Networks

- Assumptions
  - Stationary but unreliable nodes representing
    - an unreliable sensor network, nodes turn ON/OFF randomly
    - an unreliable computing grid, random service interruptions
  - A state-full reactive routing protocol is used
- Fundamental Question:
    - What is the interplay between application-level traffic requirements, network topology (e.g. node density and size) and routing overhead?
    - Can a state-full routing protocol be infinitely scalable?
    - We will consider
      - Regular or Random Node placement
      - Flat Routing

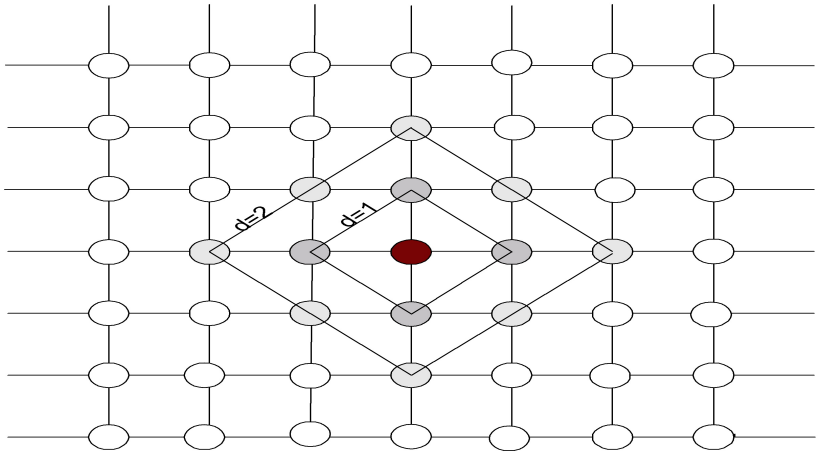
Traffic and Routing Scalability

- Argument: If all nodes communicate with all other nodes with equal probability, routing overhead grows without bound as N grows.

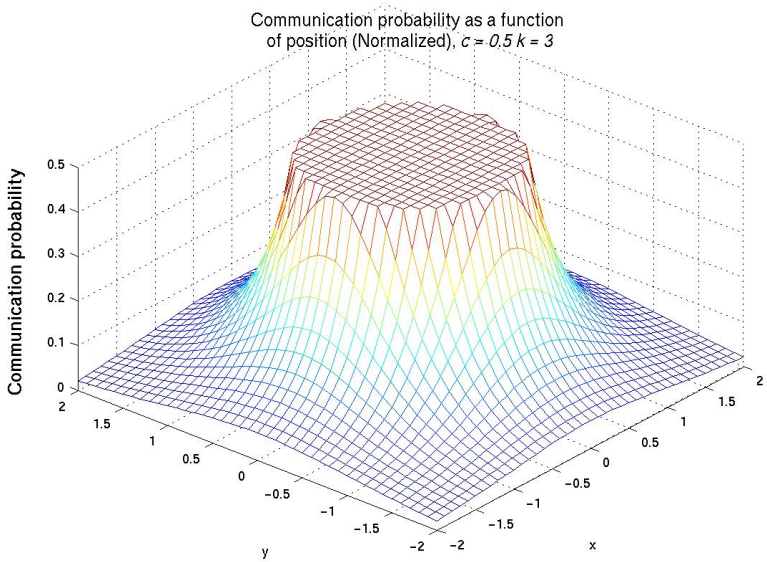
- If nodes tend to communicate with "nearby" nodes, will the routing overhead be bounded (i.e. O(1)) as "N→∞"?

- But we still want to keep the model that any node can communicate with any other node.

Network Model



A Manhattan grid (degree 4 graph).
Nodes turn ON/OFF randomly.
Network is always connected.



Communication probability as a function of the distance between nodes

Two nodes i and j at a distance $r_{i,j}$ hops have an active path (i.e. $I=1$) with probability

$$P[I=1]=P(r_{i,j})=\frac{c}{r_{i,j}^{k}}$$

- c is a scaling contant
- k is a design-controlled constant
- What is the minimum k to guarantee scalable operation?

Summary of Results

- For a given node, compute
  - $E[S_i]$:= average number of sessions initiated by the node
  - $E[S_t]$:= average number of sessions destined to the node
  - $E[S_r]$:= average number of sessions passing through the node
  - T(a) := Min Time To Live "T" such that the probability of not finding the destination node is less than a small value "a"
  - $E[N_{find}]$:= average number of RREQ packets
  - $E[N_{off}]$:= average number of RERR packets

Flooding of Control Packets

- Many protocols perform (potentially *limited*) flooding of control packets (instead of data packets) to discover routes (paths)
- Discovered routes are subsequently used to send data
- Overhead of control packet flooding is amortized over data packets transmitted between consecutive control packet floods
- E.g. DSR and AODV

Dynamic Source Routing (DSR)

[Johnson96]

- Initiated when node S wants to send data to node D, but does not know a valid route node D, node S initiates a route discovery by flooding the network with a Route Request (RREQ) packet

- Each node appends own identifier when forwarding RREQ
- Destination D on receiving the first RREQ, sends a Route Reply (RREP)

- RREP is sent on a route obtained by reversing the route appended to received RREQ

- RREP includes the route from S to D on which RREQ was received by node D

Ad Hoc On-Demand Distance Vector Routing (AODV) [Perkins99Wmcsa]

- DSR includes source routes in packet headers
  - Resulting large headers can sometimes degrade performance, particularly when packet payload is small

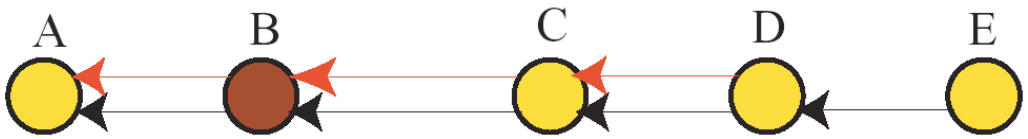
- AODV attempts to improve on DSR by maintaining routing tables at the nodes, so that data packets do not have to contain routes

- RREQ/RREP packets are forwarded similar to DSR

Route Caching

- Each node caches a new route it learns by *any means**
- Node X on receiving a Route Request for some node D can send a Route Reply if node X knows a route to node D
- Use of route cache
  - can speed up route discovery
  - can reduce propagation of route requests

Note on Route Error Notification



- Two paths as shown, and node B fails.
- Two modes:
  - "Source-route" error notification (e.g. DSR): node D receives two RERR packets from C, one for each route
  - "Distance-vector" error notification (e.g. AODV): node D receives only one RERR from C.

TTL Optimizations: Expanding Ring Search

- TTL field is initialized to some value which is the maximum number of hops that the packet should travel
- Every intermediate node decrements this field by one.
- If the value of the field is zero, the packet is dropped.
- Route Requests are initially sent with small Time-to-Live (TTL) field, to limit their propagation
  - DSR also includes a similar optimization
- If no Route Reply is received, then larger TTL tried

Summary of Results

- For a given node, compute
  - $E[S_i]$:= average number of sessions initiated by the node
  - $E[S_t]$:= average number of sessions destined to the node
  - $E[S_r]$:= average number of sessions passing through the node
  - T(a) := Min Time To Live "T" such that the probability of not finding the destination node is less than a small value "a"
  - $E[N_{find}]$:= average number of RREQ packets
  - $E[N_{off}]$:= average number of RERR packets

Summary of Results

- For an (infinite network) N→ ∞:

$$T(\alpha) = \min\left\{ i \,\middle|\, \frac{\sum_{d=i+1}^{\infty} \frac{1}{d^{k-1}}}{\sum_{d=1}^{\infty} \frac{1}{d^{k-1}}} \le \alpha \right\}; k > 2$$

$$E[S_t] = 4c * f(k-1); k > 2$$

$$E[S_r] = 4c * (f(k-2) - f(k-1)); k > 3$$

$$E[S] = 4c * (f(k-2) + f(k-1)); k > 3$$

$$p^r = 1 - \frac{2f(k-1)}{f(k-2) + f(k-1)}$$

$$N_{find} = 1 + 2T(T-1)$$

$$E[N_{off}] = 4c * (f(k-2) - f(k-1)); k > 3$$

(local repair with src route notif.)

$$E[N_{off}] = 4c * (f(k-3) - f(k-2)); k > 4$$

(src route notif.)

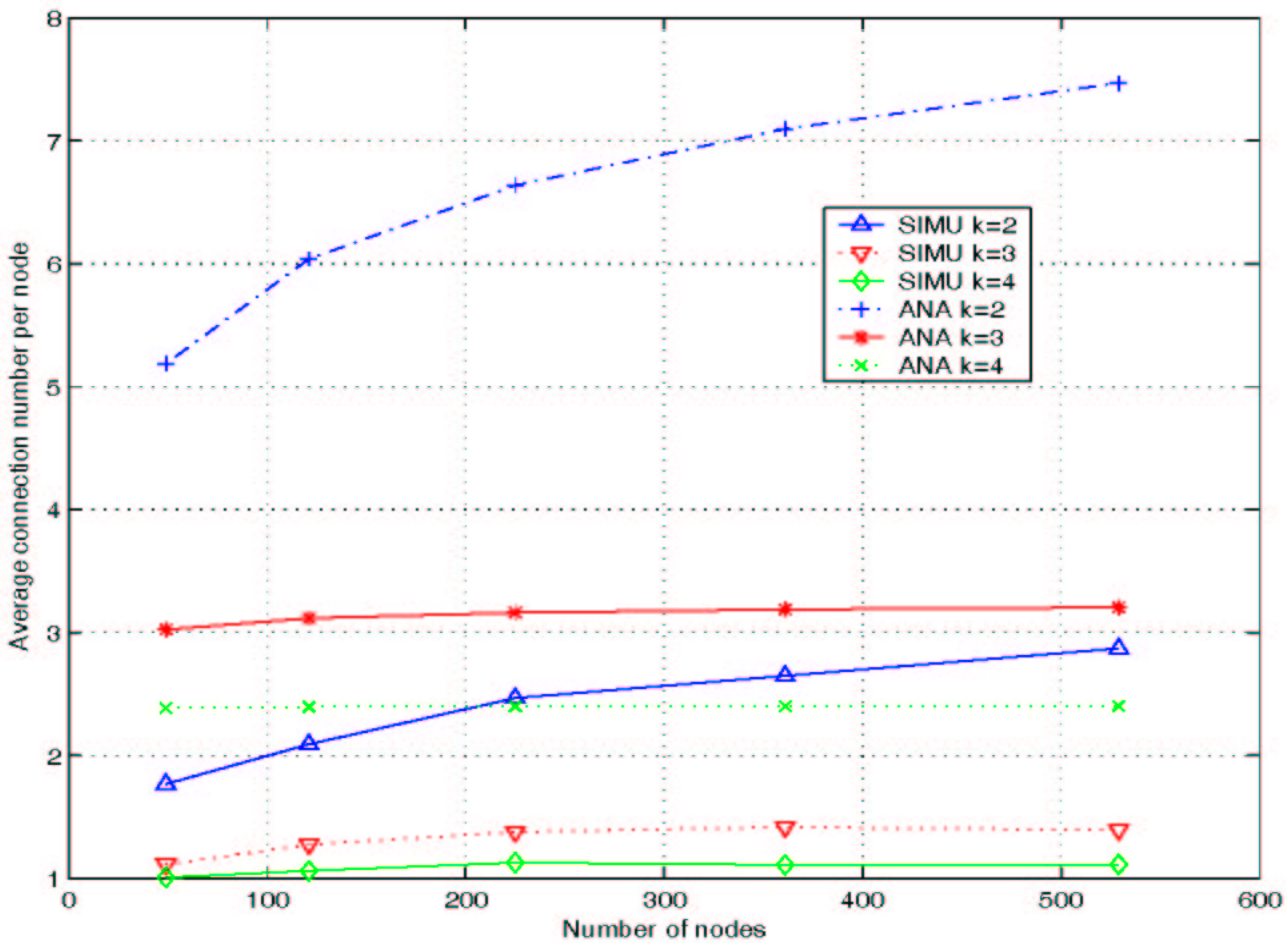
where $f(k) @ \sum_{d=1}^{\infty} \frac{1}{d^k}$

- Main Result: **k>4** necessary for bounded overhead.

Validations Through Simulations

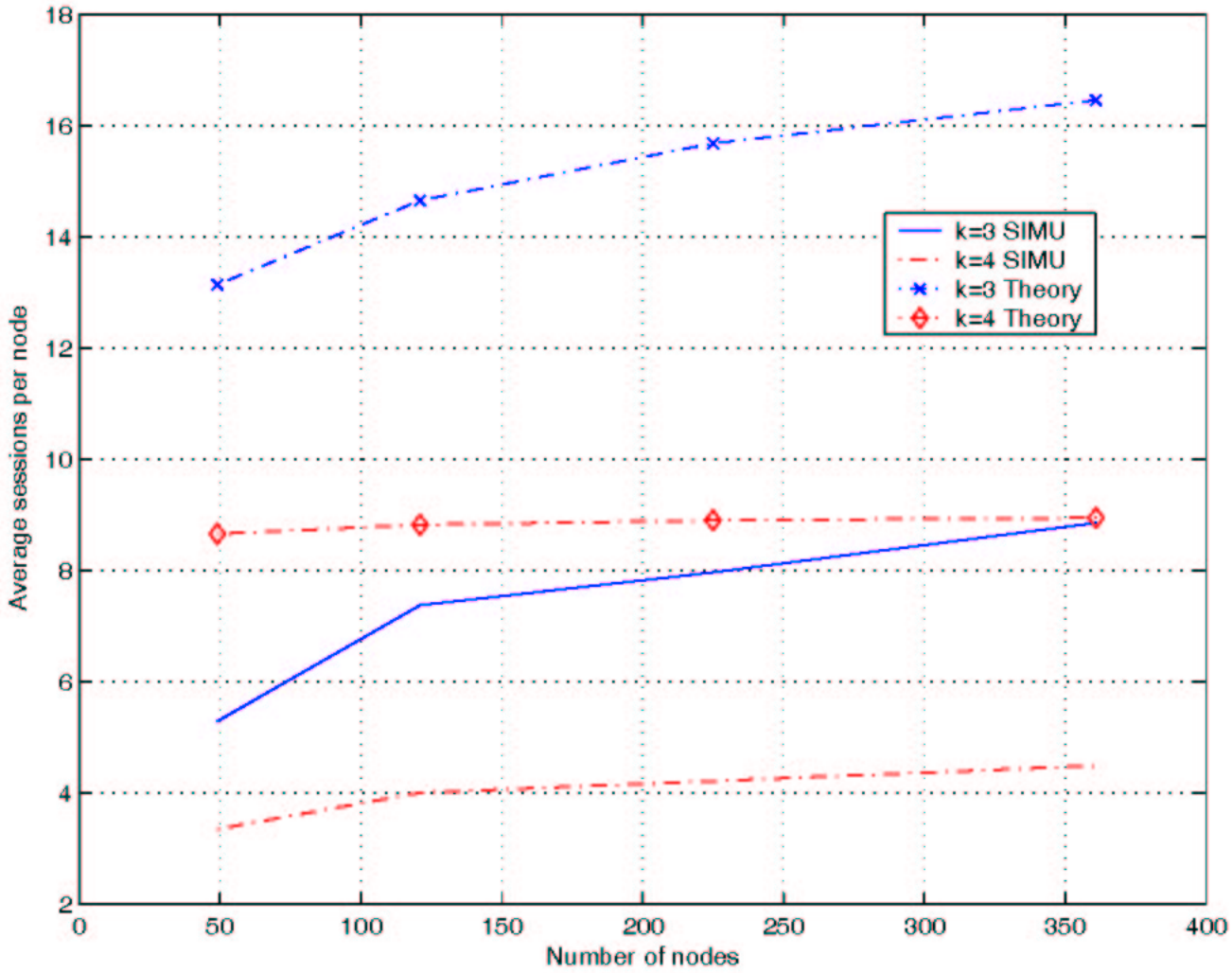
- We use *ns-2* simulator with CMU's ad-hoc networks extensions (MAC 802.11, Omni antenna, etc.)
- Purpose: Validate the dependencies (trends) rather than absolute values
- Difficulty: Cannot simulate too large neworks
- Networks of size N=49, 121, 225, 361 and 529
- Use CBR $\sqrt{N}$
- Each point represents average of 5 runs with same topology but different random traffic seed
- Sum from 1 till the network diameter

$E[S_t]$


Average connection number per node
Number of nodes
SIMU k=2
SIMU k=3
SIMU k=4
ANA k=2
ANA k=3
ANA k=4
0
100
200
300
400
500
600
1
2
3
4
5
6
7
8

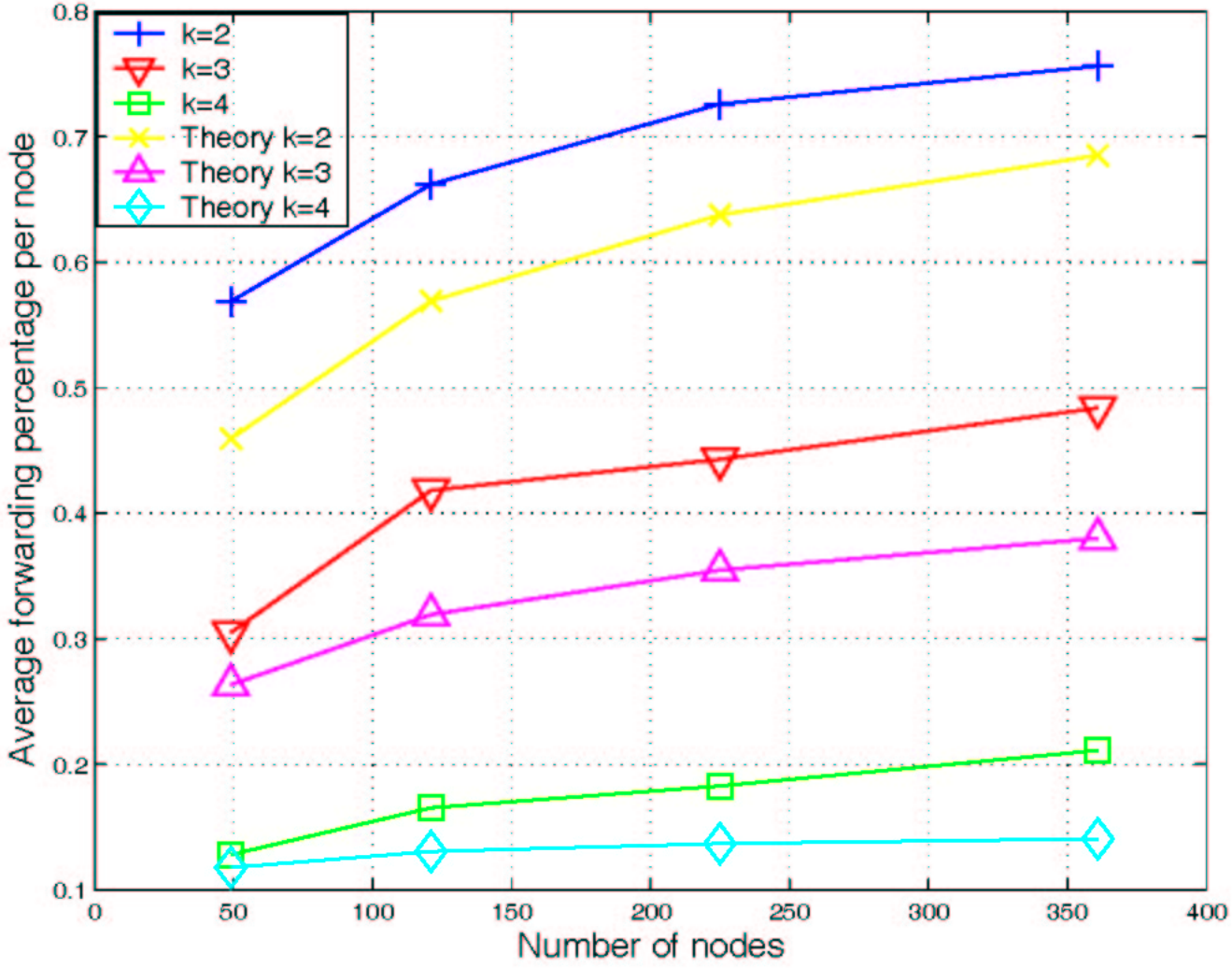
- k>2

E[S]

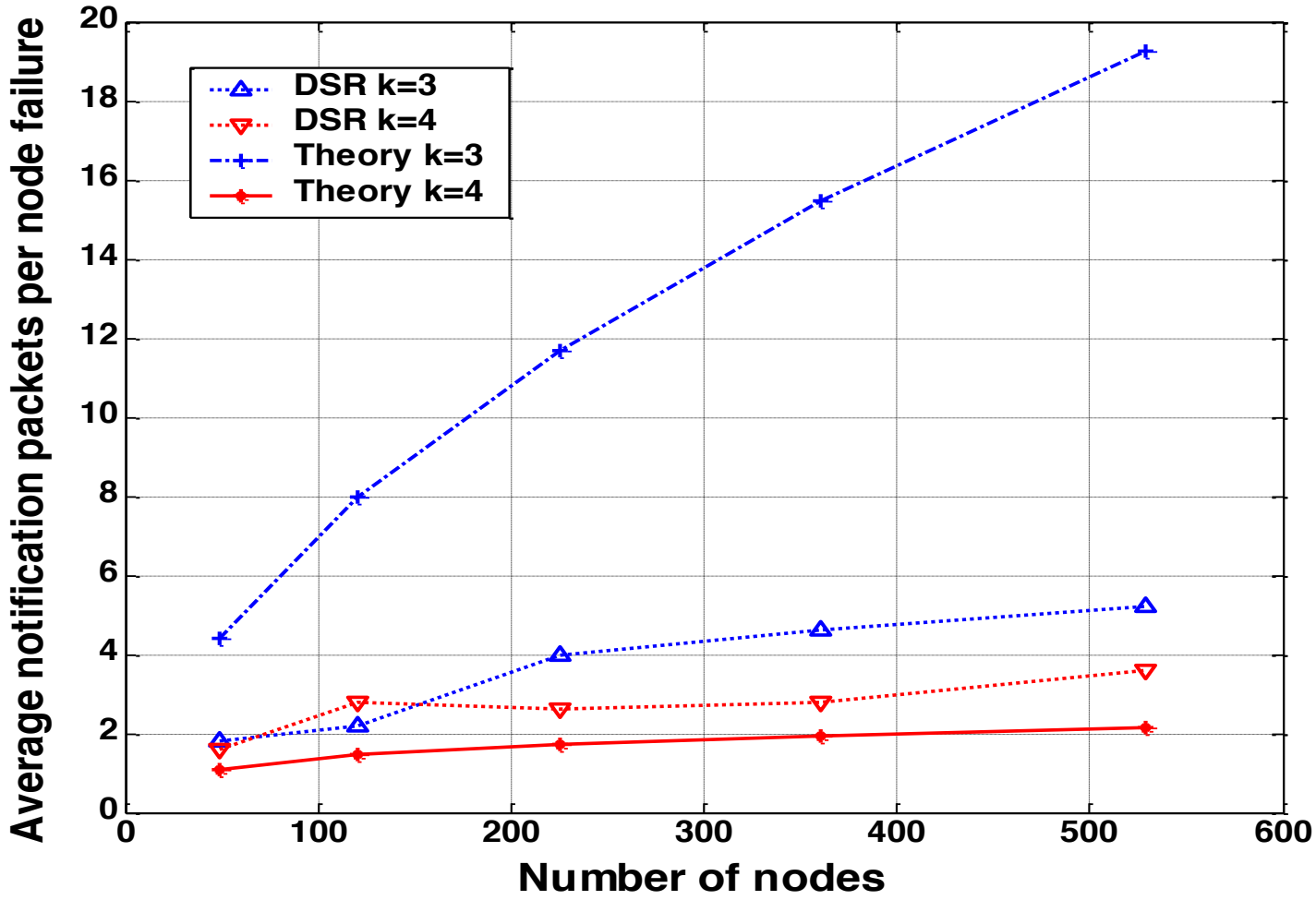

18
16
14
12
10
8
6
4
2
0
50
100
150
200
250
300
350
400
Average sessions per node
Number of nodes
k=3 SIMU
k=4 SIMU
k=3 Theory
k=4 Theory

- k>3

p_r

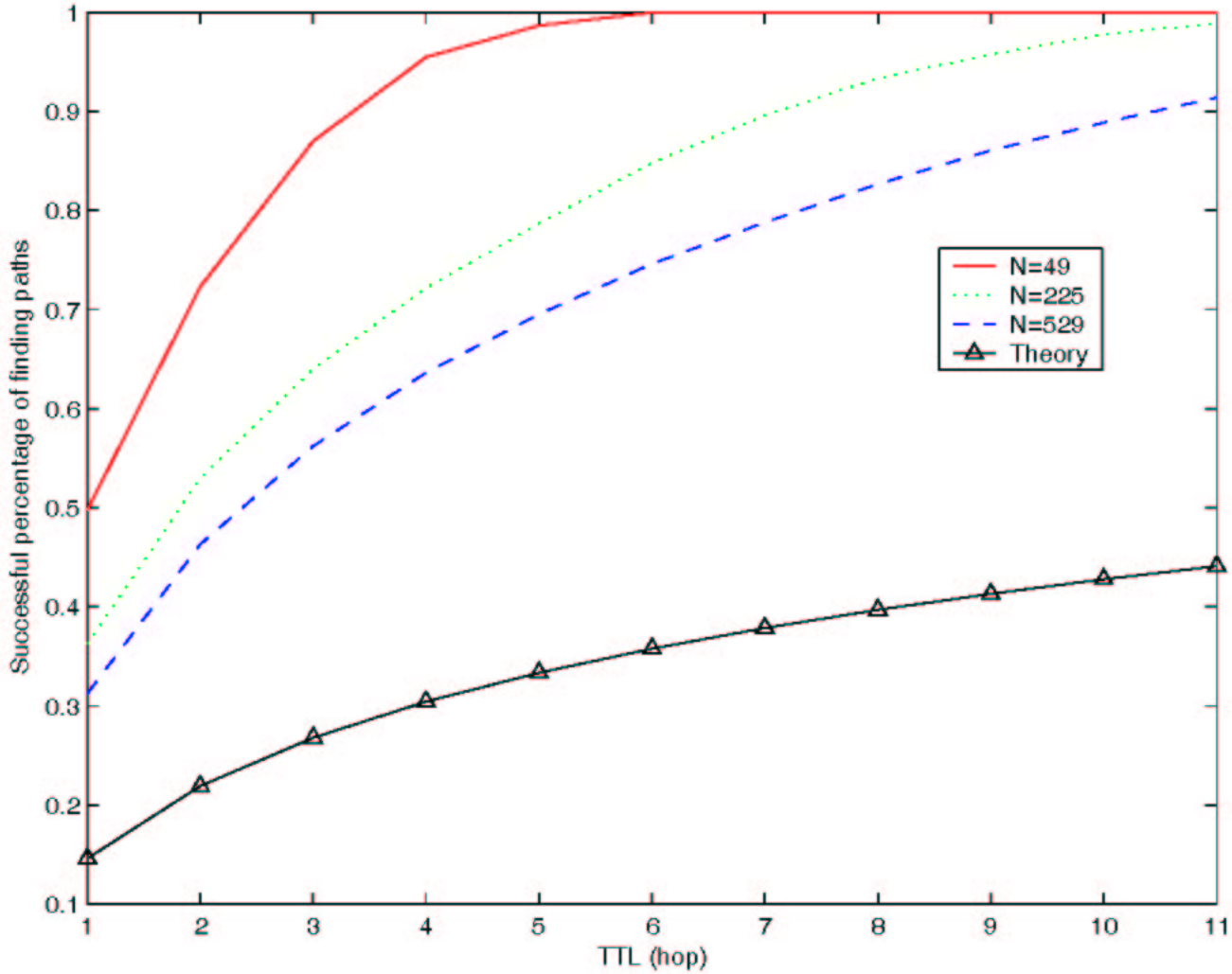

k=2
k=3
k=4
Theory k=2
Theory k=3
Theory k=4
Average forwarding percentage per node
Number of nodes
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
50
100
150
200
250
300
350
400

$E[N_{off}]$


DSR k=3
DSR k=4
Theory k=3
Theory k=4
Average notification packets per node failure
Number of nodes
0
2
4
6
8
10
12
14
16
18
20
0
100
200
300
400
500
600

- k>4

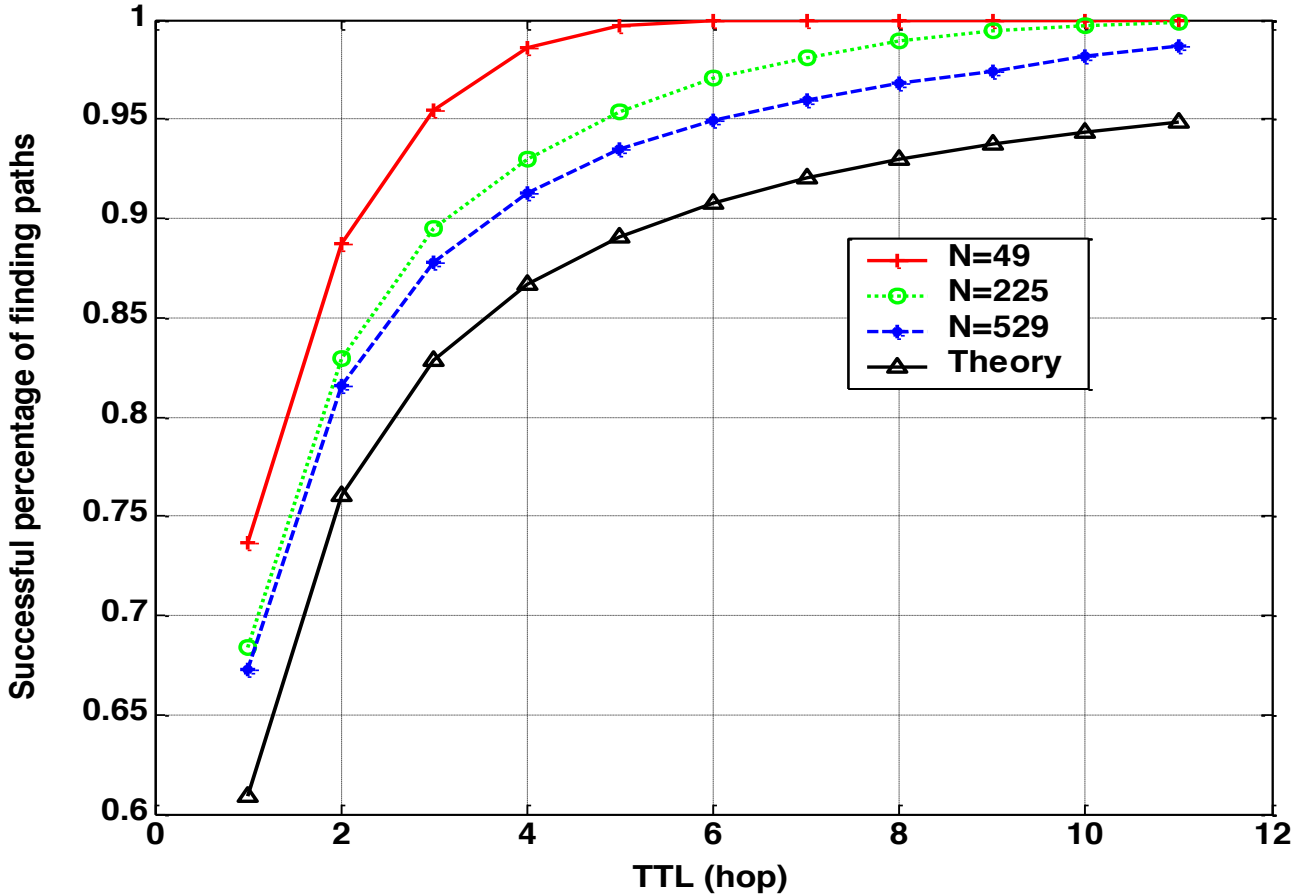
T(¥)


Successful percentage of finding paths
TTL (hop)
N=49
N=225
N=529
Theory
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
1
2
3
4
5
6
7
8
9
10
11

k=2

- k>2

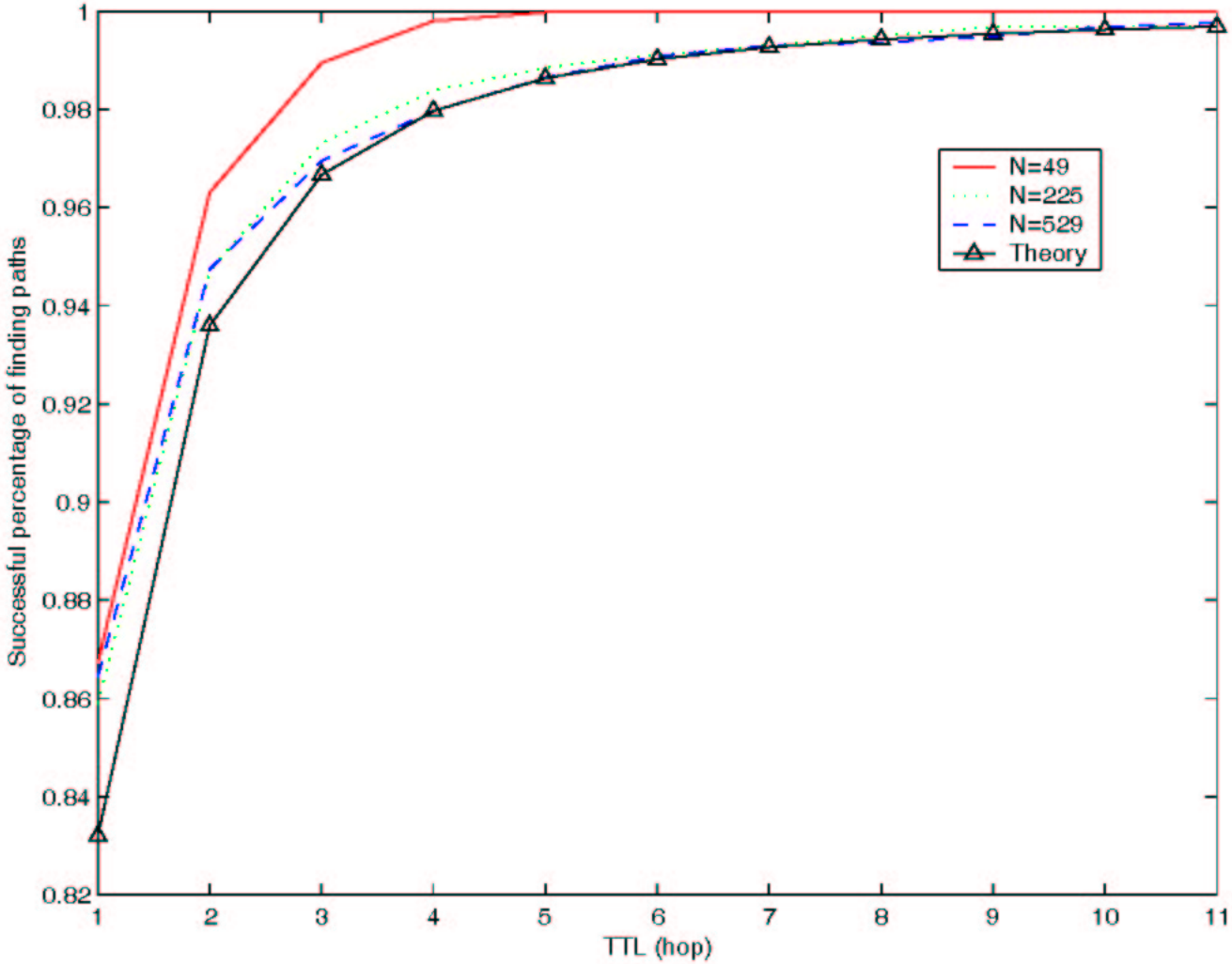
T(W)


Successful percentage of finding paths
TTL (hop)
N=49
N=225
N=529
Theory
1
0.95
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0
2
4
6
8
10
12

k=3

- k=3

T(¥)


Successful percentage of finding paths
TTL (hop)
N=49
N=225
N=529
Theory
1
0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
1
2
3
4
5
6
7
8
9
10
11

k=4

- k>2

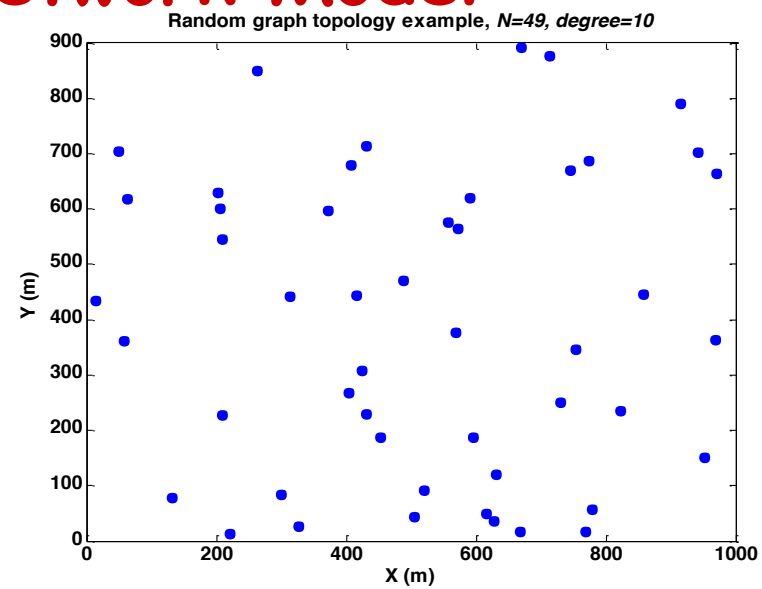
More Realistic Network Model

$$P_N(n) = \frac{(\lambda A)^n}{n!} e^{-\lambda A}$$

$$P(r_{i,j}) = \begin{cases} \dfrac{c}{r_1^k} & 0 \le r_{i,j} < r_1 \\ \dfrac{c}{r_{i,j}^k} & r_{i,j} \ge r_1 \end{cases}$$

where $0 \le c \le r_1^k$

$$h_{i,j} \approx \left\lceil \beta \frac{r_{i,j}}{r_0} \right\rceil$$



Random distribution (Poisson Field) with average degree 10 (above conectivity threshold)

Validations Through Simulations

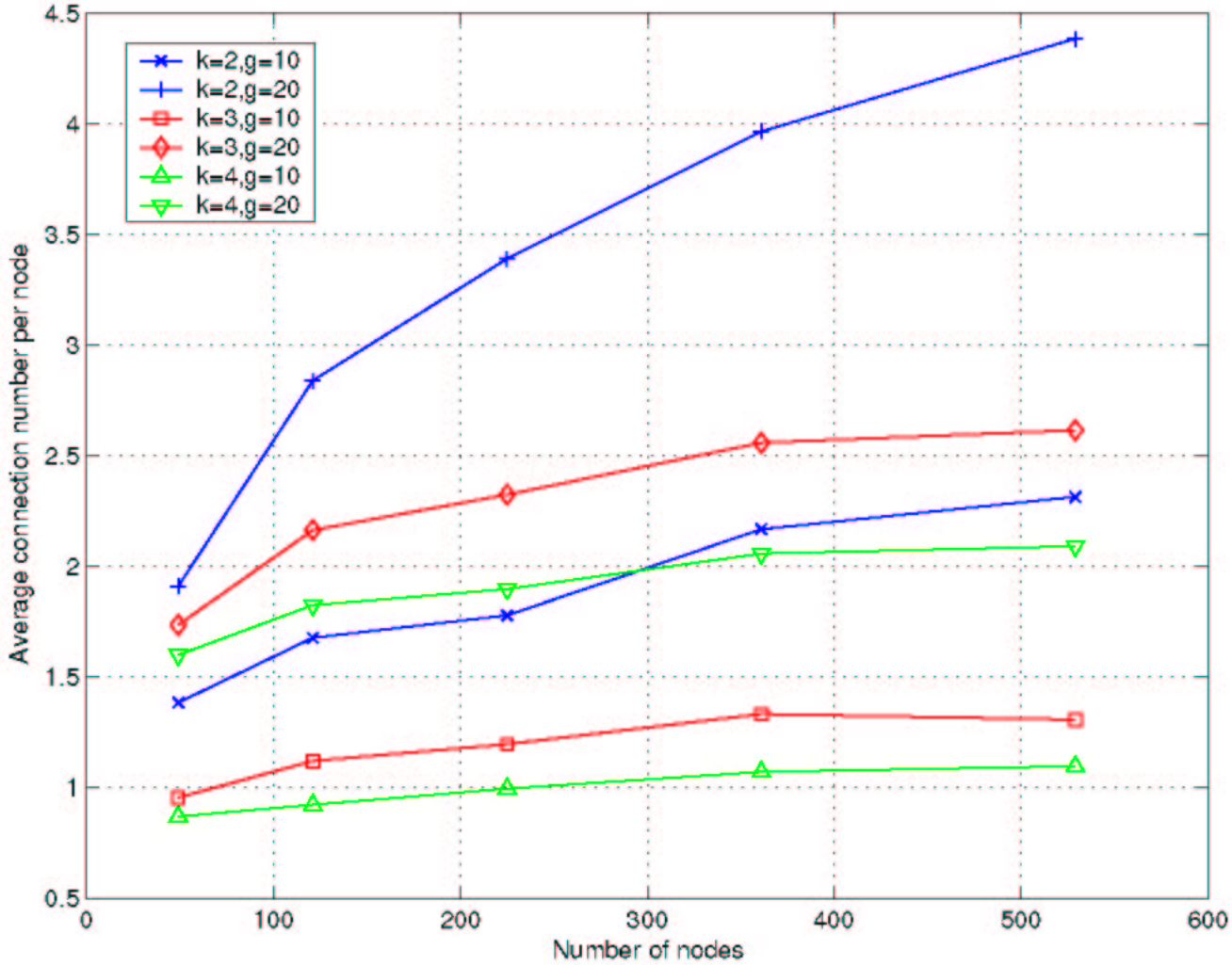
- We use *ns-2* simulator with CMU's ad-hoc networks extensions (MAC 802.11, Omni antenna, etc.)
- Purpose: Validate the dependencies (trends)
- Difficulty: Cannot simulate too large neworks
- Networks of size N=49, 121, 225, 361 and 529
- For random topologies, choose area A, average degree g

$$g: \ \Theta(\log N)$$

$$A = \frac{\pi r_0^2 N}{g}$$

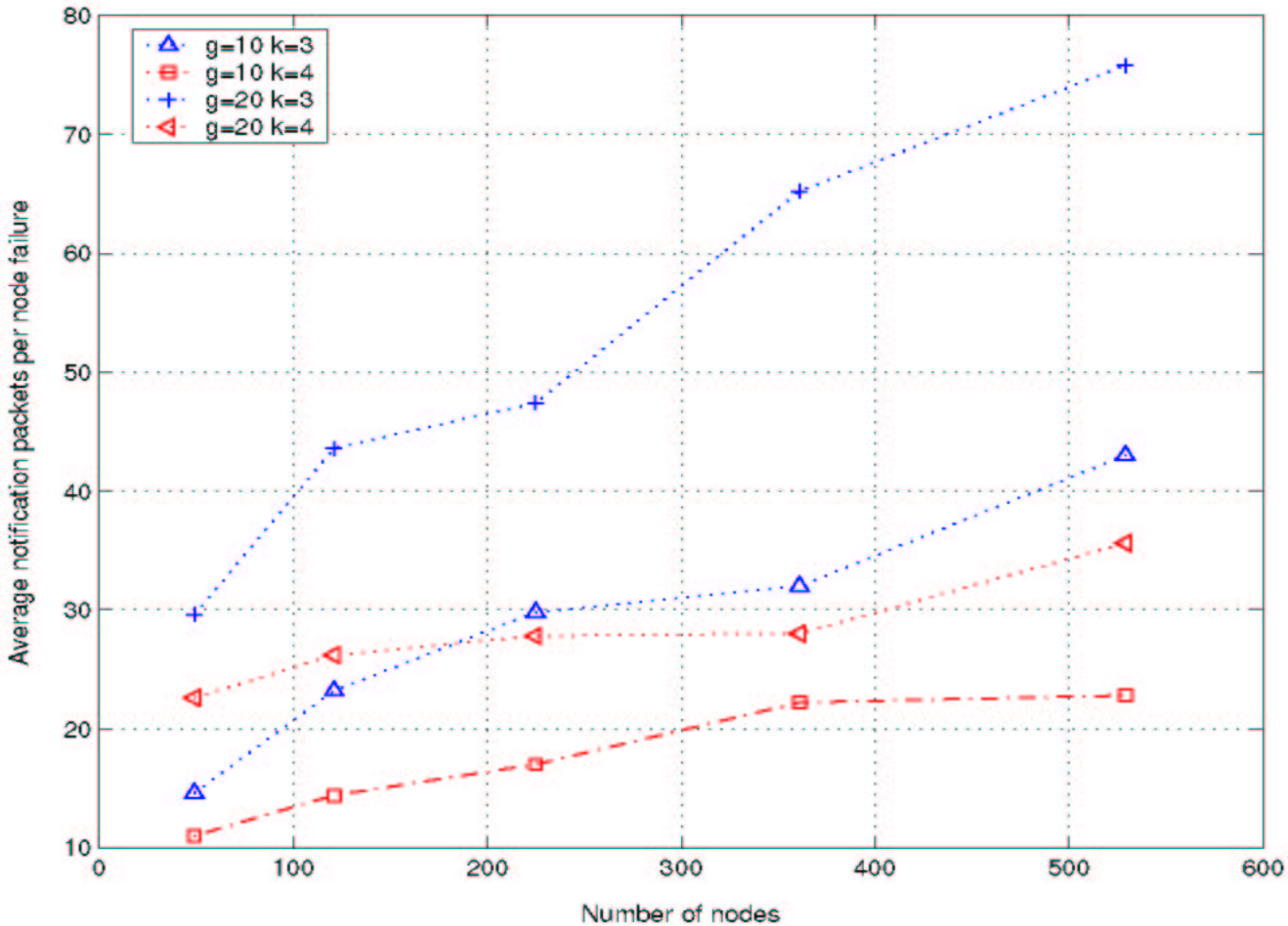
$$r_0 = 250\text{m.}$$

$E[S_t]$


k=2,g=10
k=2,g=20
k=3,g=10
k=3,g=20
k=4,g=10
k=4,g=20
Average connection number per node
Number of nodes
0.5
1
1.5
2
2.5
3
3.5
4
4.5
0
100
200
300
400
500
600

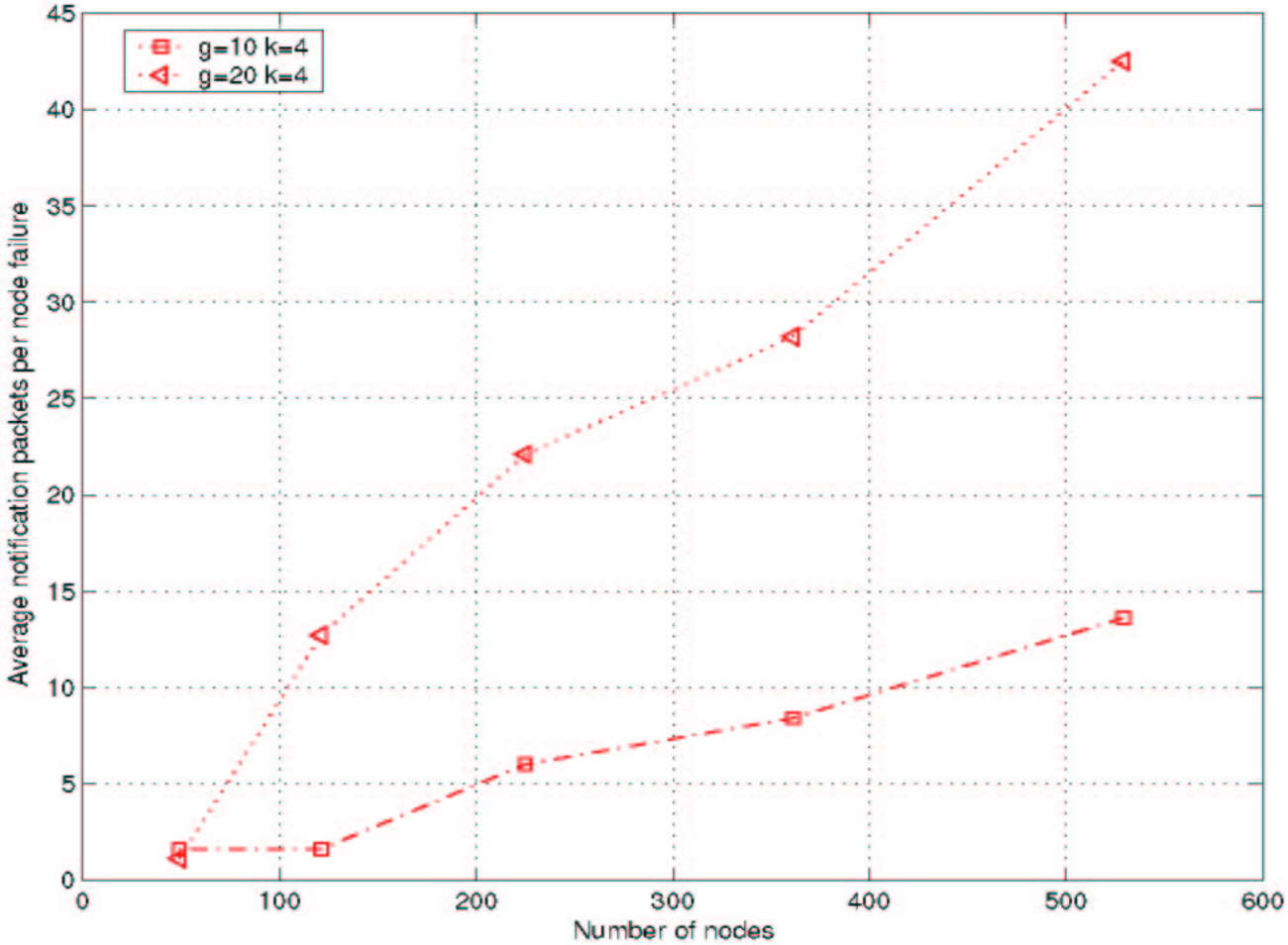
- k>2

$E[N_{off}]$


g=10 k=3
g=10 k=4
g=20 k=3
g=20 k=4
Average notification packets per node failure
Number of nodes
80
70
60
50
40
30
20
10
0
100
200
300
400
500
600

- k>3

$E[N_{off}]$


g=10 k=4
g=20 k=4
Average notification packets per node failure
Number of nodes
0
5
10
15
20
25
30
35
40
45
0
100
200
300
400
500
600

- k>4