Practical 3D Geographic Routing for Wireless Sensor Networks

Jiangwei Zhou[*], Yu Chen[+], Ben Leong[∇],
Pratibha Sundar Sundaramoorthy[∇]
[*]Xi’an Jiaotong University
[+]Duke University
[∇]National University of Singapore

Geographic Routing Algorithms

Exploit geometric information (coordinates) of network topology to improve scalability of point-to-point routing

Geographic Routing Algorithms

Greedy forwarding +

Recovery mode when local minimum is encountered

1. Efficient
2. Storage proportional to density, not size

Motivation

Previously proposed geographic routing algorithms assume “planar” network topology

⇒ Many modern sensor networks are three-dimensional

Two Questions

1. How do we get geographic routing to work for 3D networks?

Two Questions

2. How do existing point-to-point algorithms compare?
   (should we care?)

Outline

- Problem & Motivation
- Overview of related work & geographic routing
- Our Solution: GDSTR-3D
- Performance Evaluation
- Conclusion
- Future Work

Related work

- 2D geographic routing
  - GPSR (Karp & Kung, Mobicom 2000)
  - GOAFR+ family (Kuhn et al., Mobihoc 2003)
  - CLDP (Kim et al., NSDI 2005)
  - GDSTR (Leong et al., NSDI 2006)
- 3D geographic routing
  - GRG (Flury & Wattenhofer, Infocom 2008)
  - GHG (Liu & Wu, Infocom 2009)

Related work

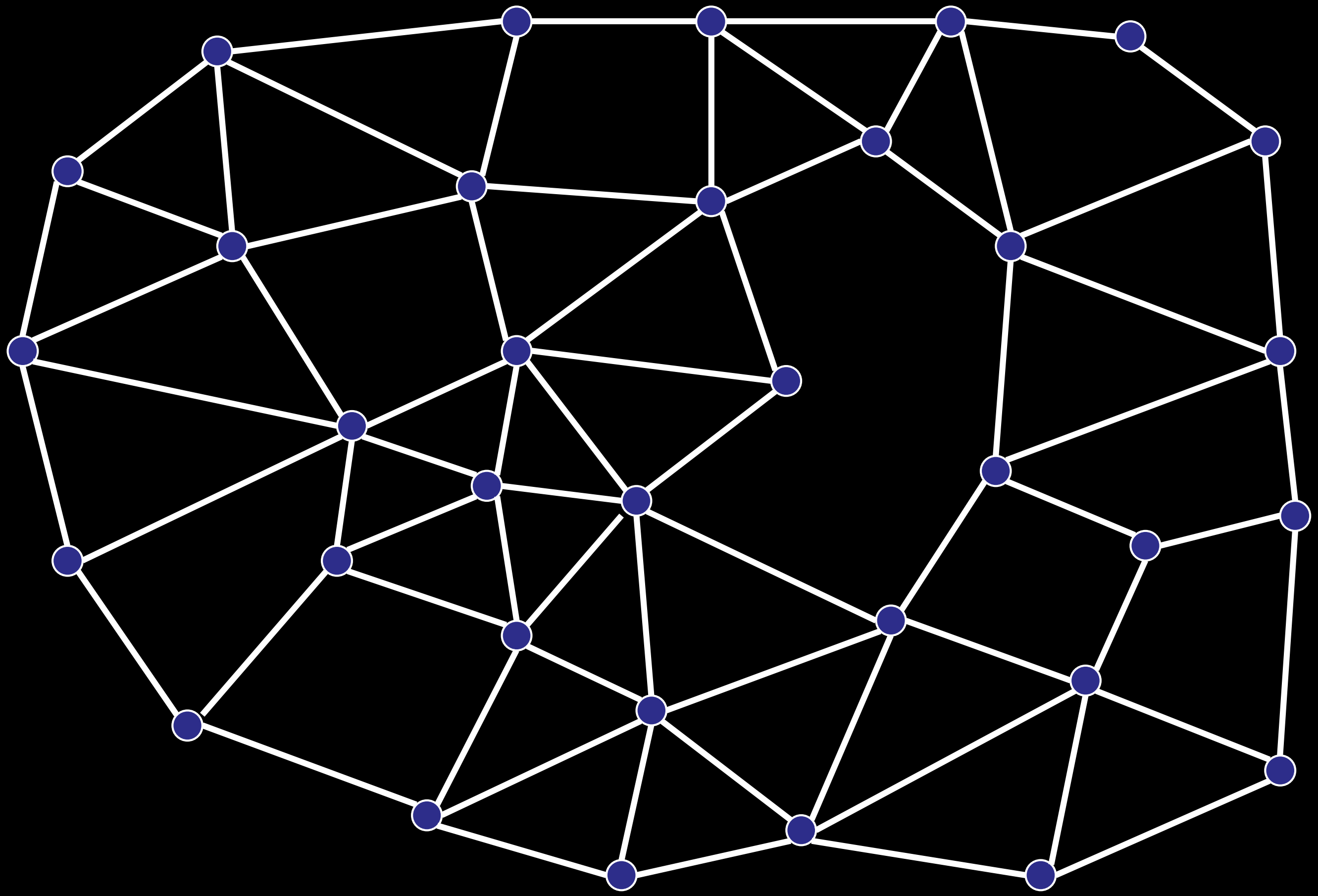
- Point-to-point
  - — AODV (Perkins, Milcom 1997)
  - — VPCR (Newsome & Song, SenSys 2003)
  - — BVR (Fonseca et al., NSDI 2005)
  - — VRR (Caesar et al., SIGCOMM 2006)
  - — S4 (Mao et al., NSDI 2007)
- Virtual Coordinates
  - — NoGeo (Rao et al., Mobicom 2003)
  - — PSVC (Zhou et al., ICNP 2010)

Our Approach

Extend GDSTR
to 3D

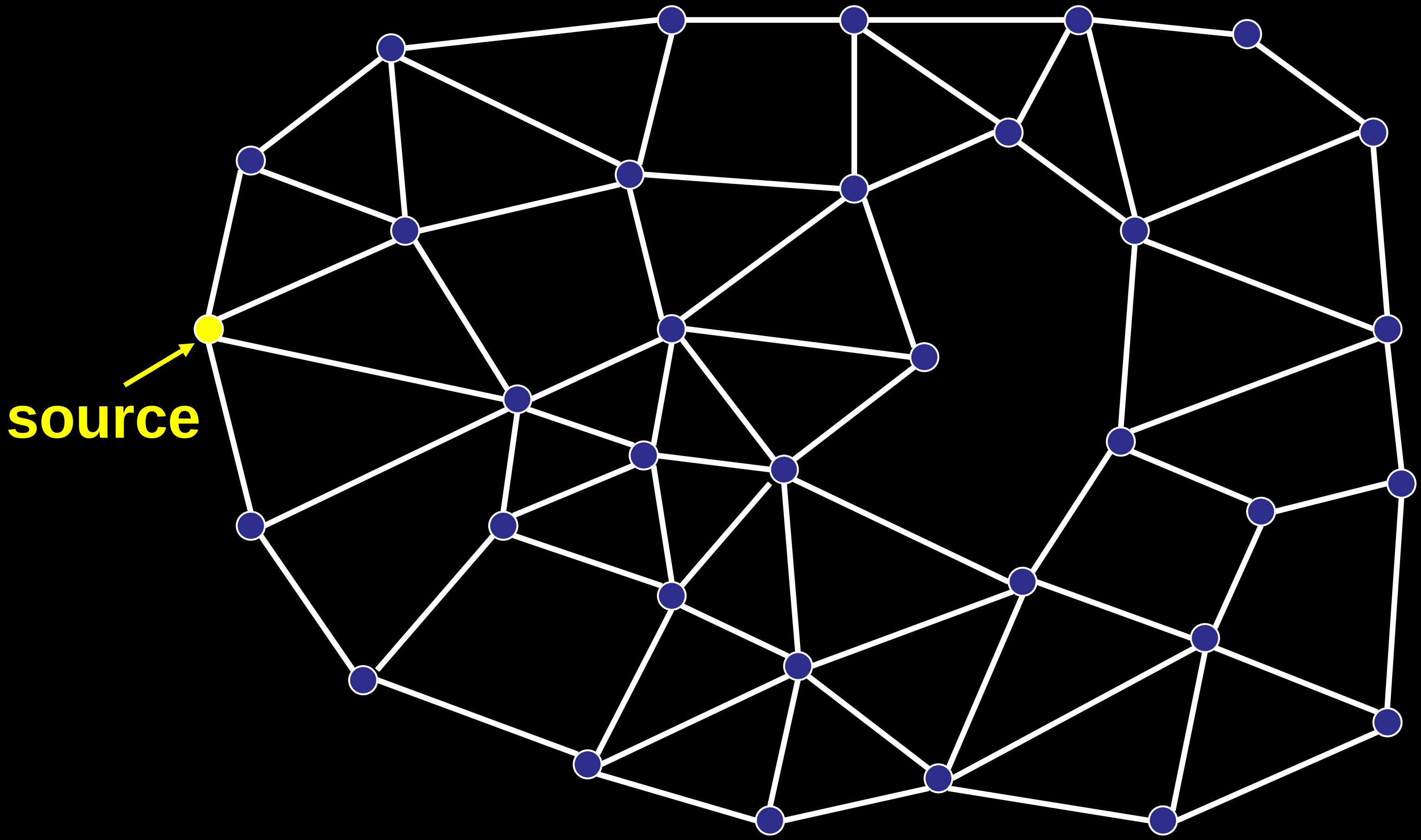
Complications!

Overview: Geographic Routing



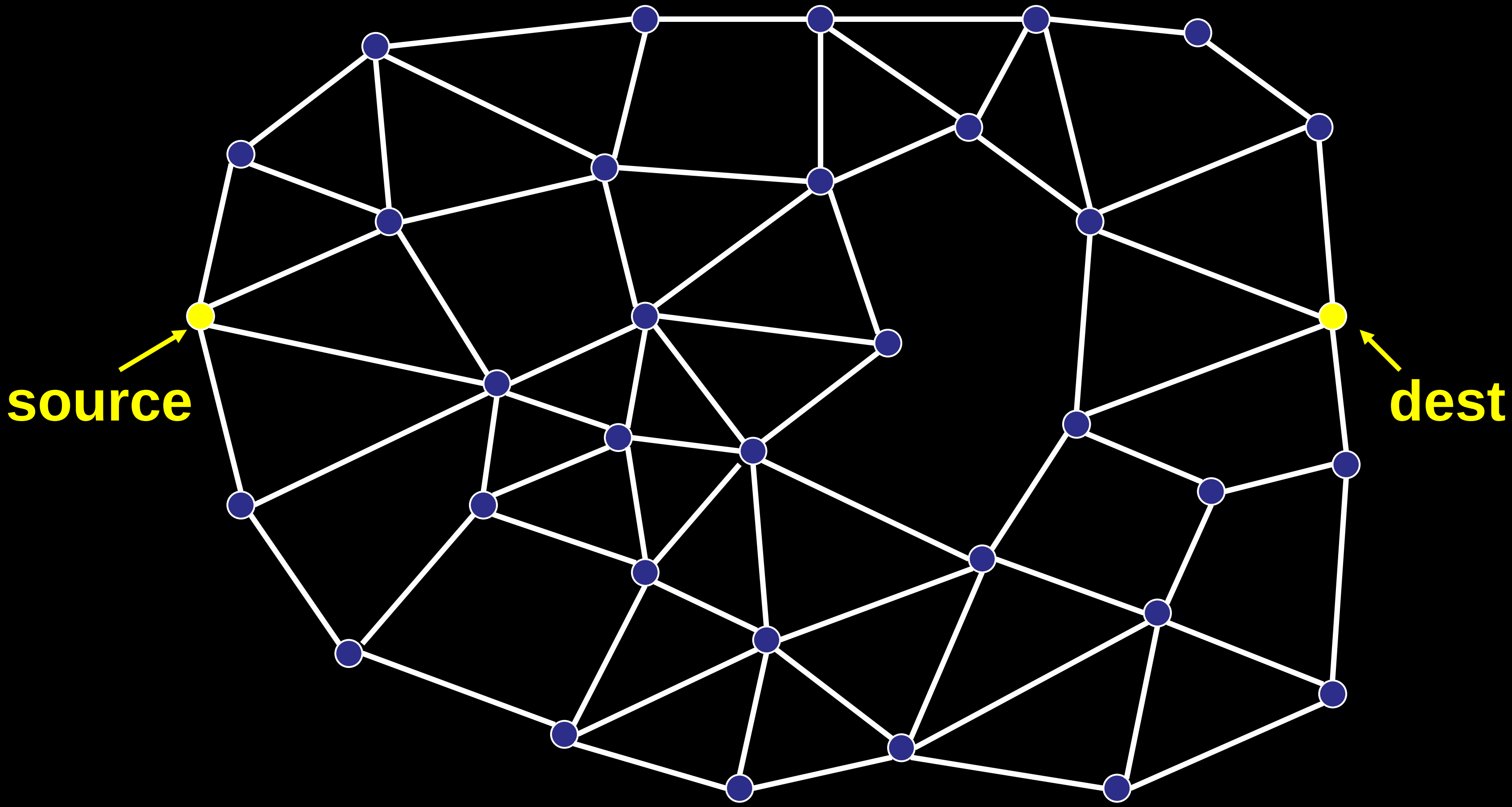
Nodes have coordinates

Overview: Geographic Routing



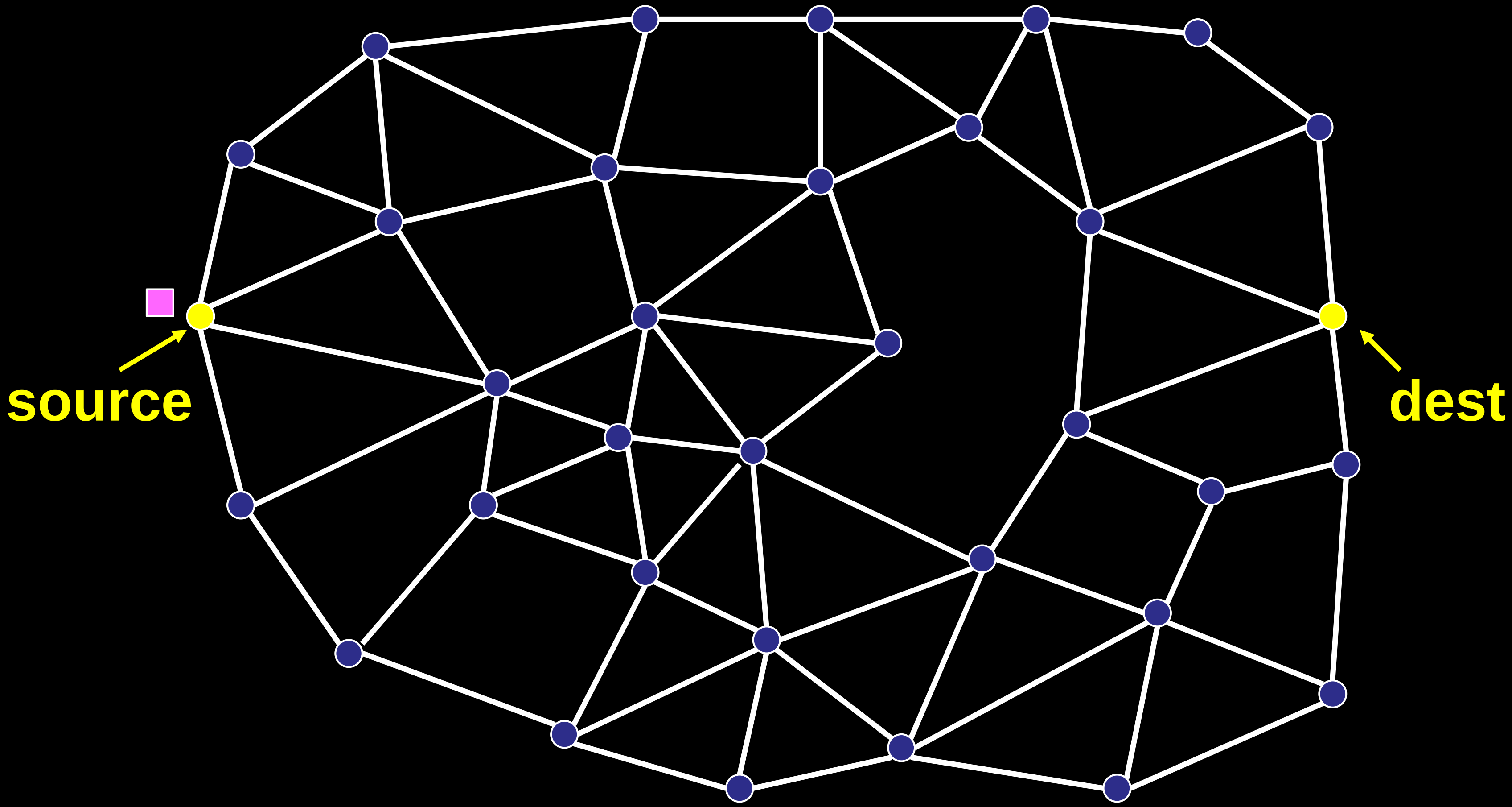
Nodes have coordinates

Overview: Geographic Routing



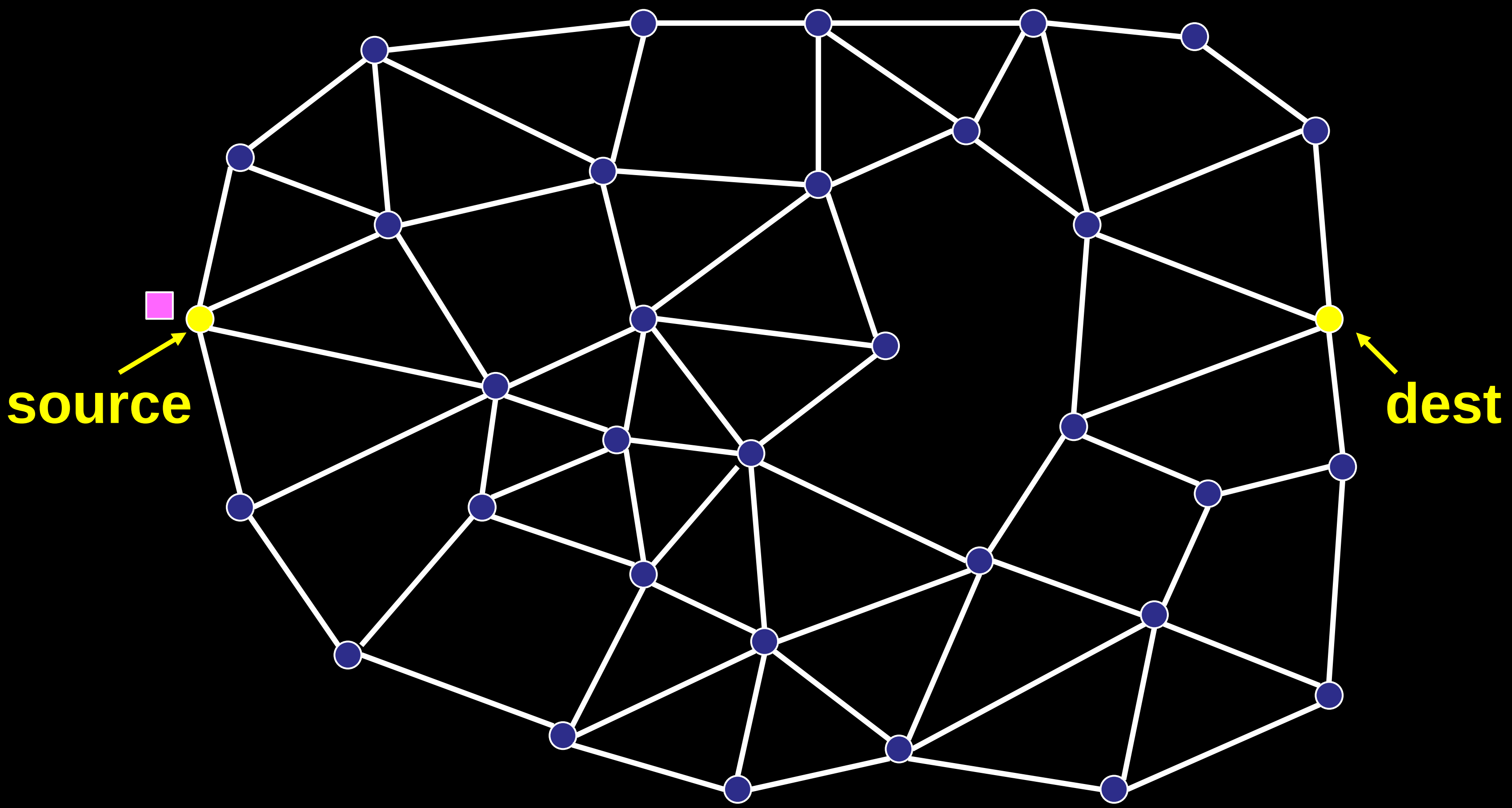
Nodes have coordinates

Overview: Geographic Routing



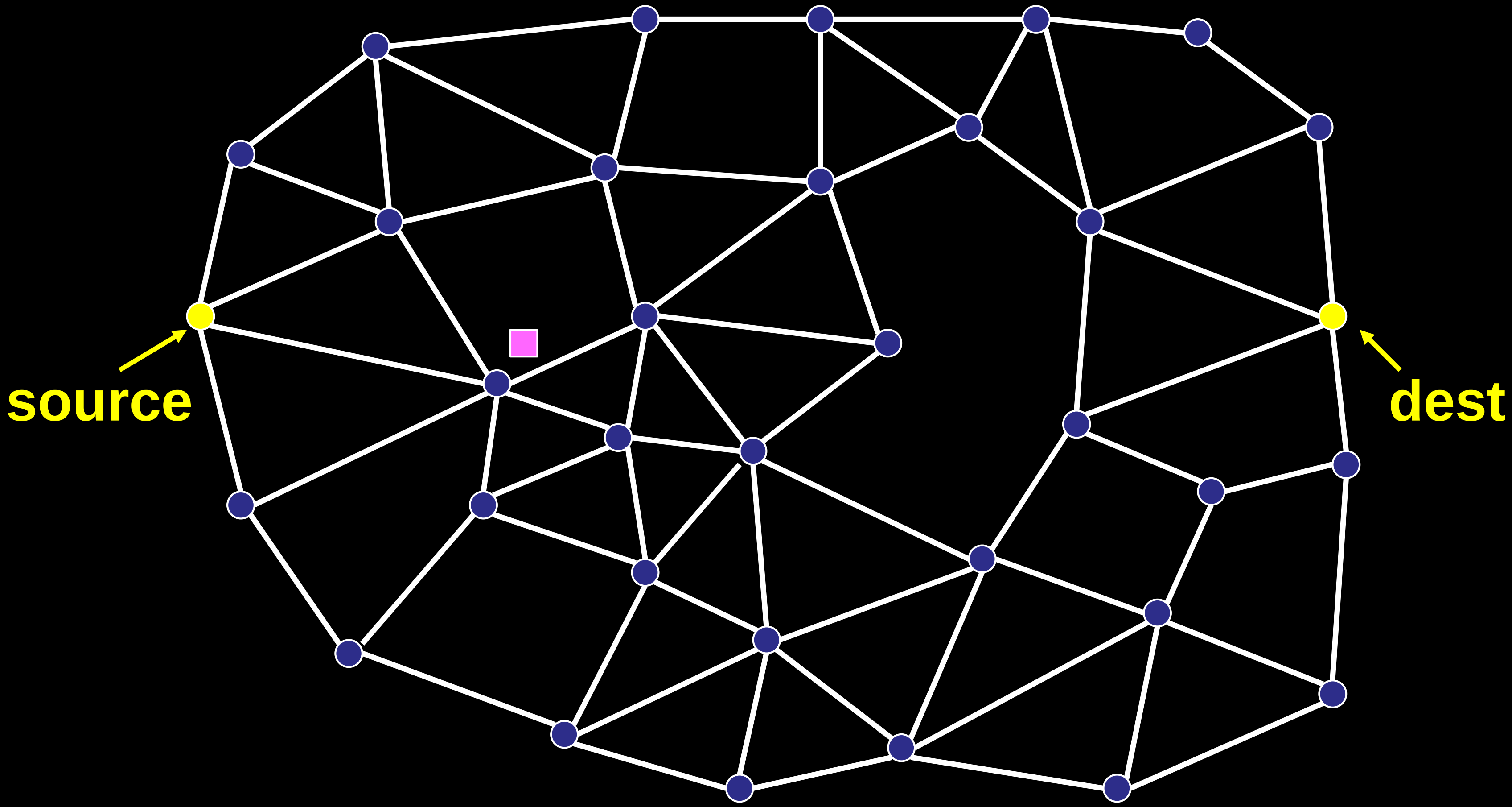
Packet contains coordinates of destination

Overview: Geographic Routing



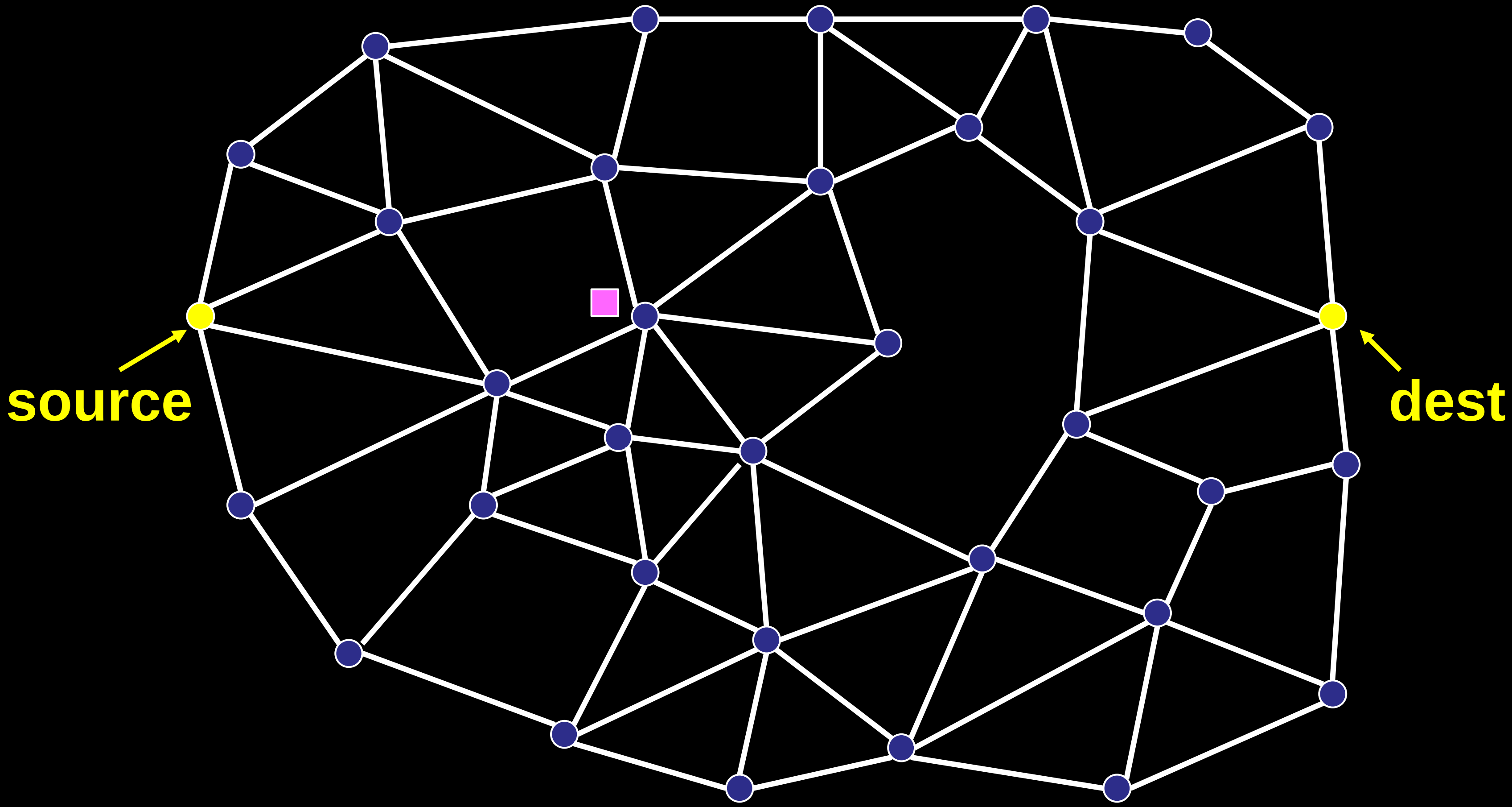
Greedy forwarding!

Overview: Geographic Routing



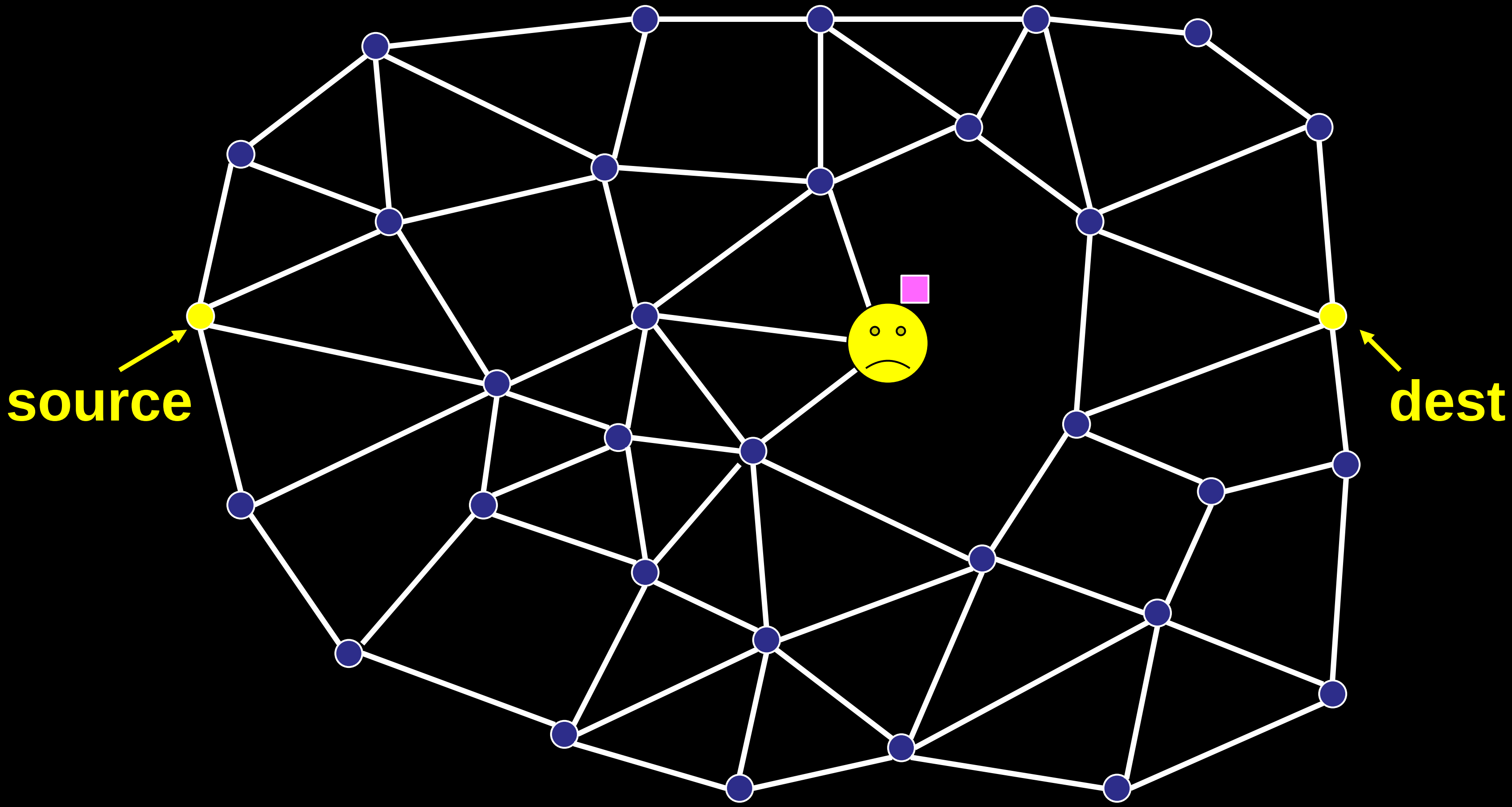
Greedy forwarding!

Overview: Geographic Routing



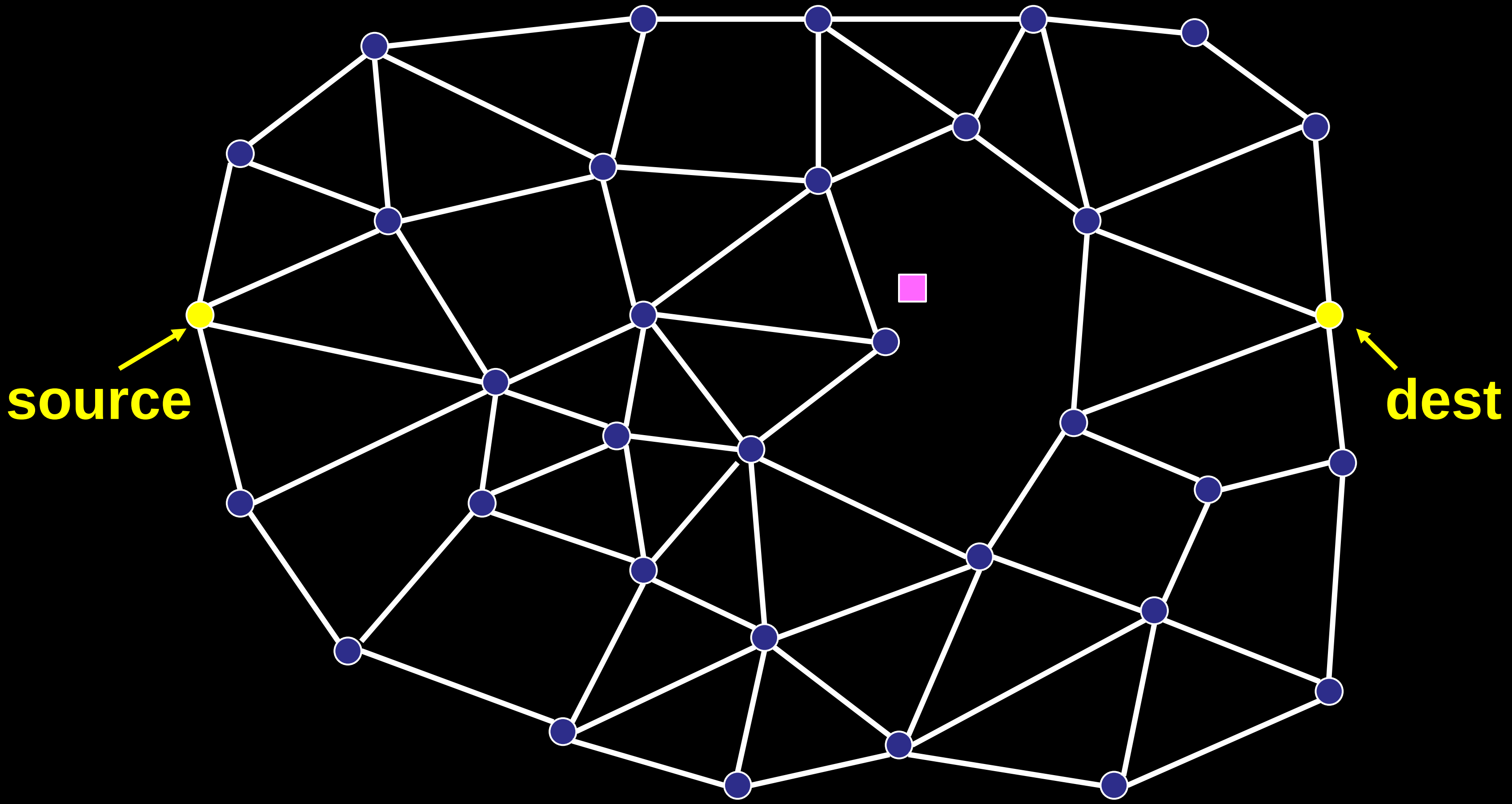
Greedy forwarding!

Overview: Geographic Routing



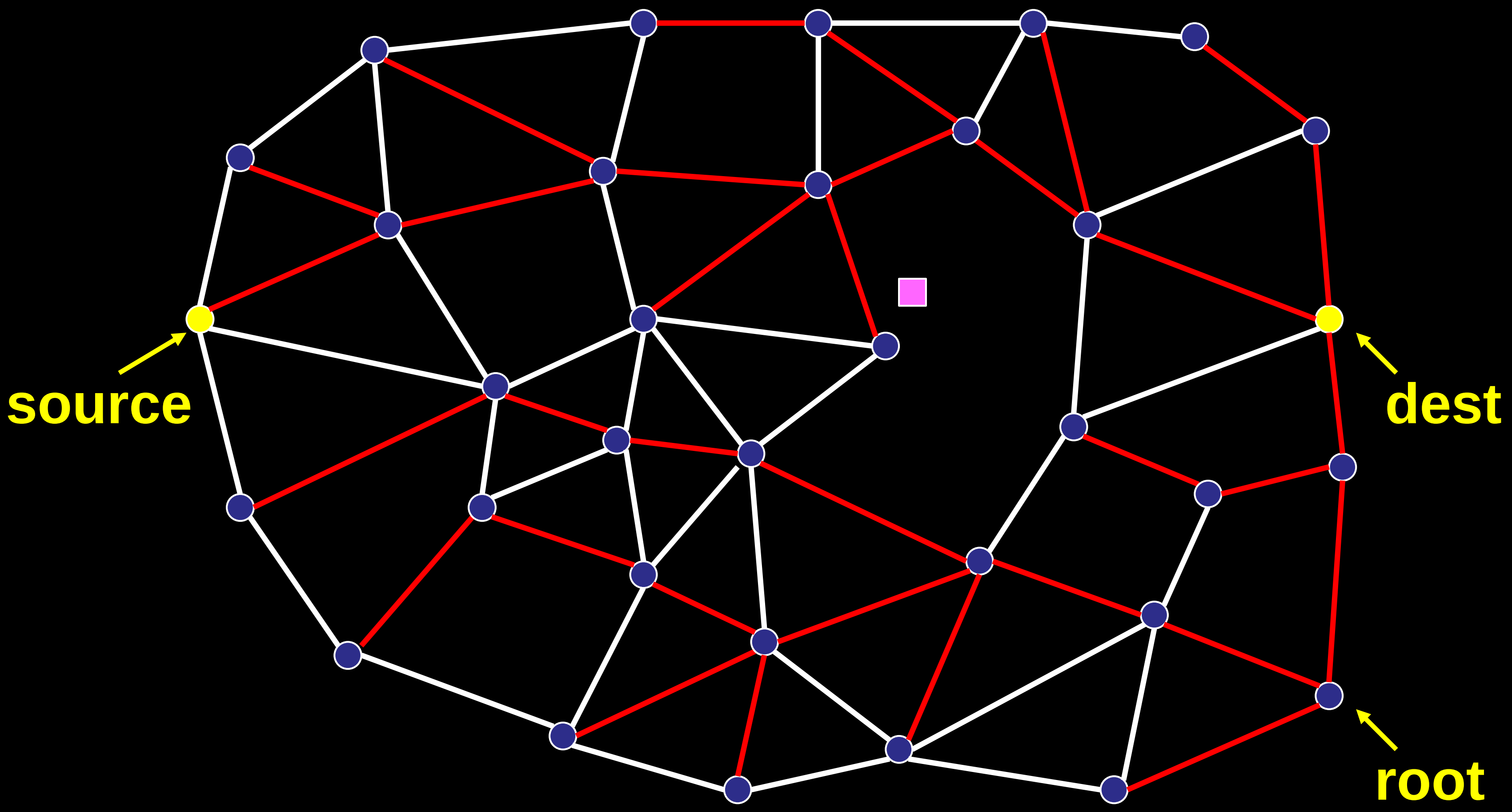
Dead end! (local minima)

Overview: GDSTR



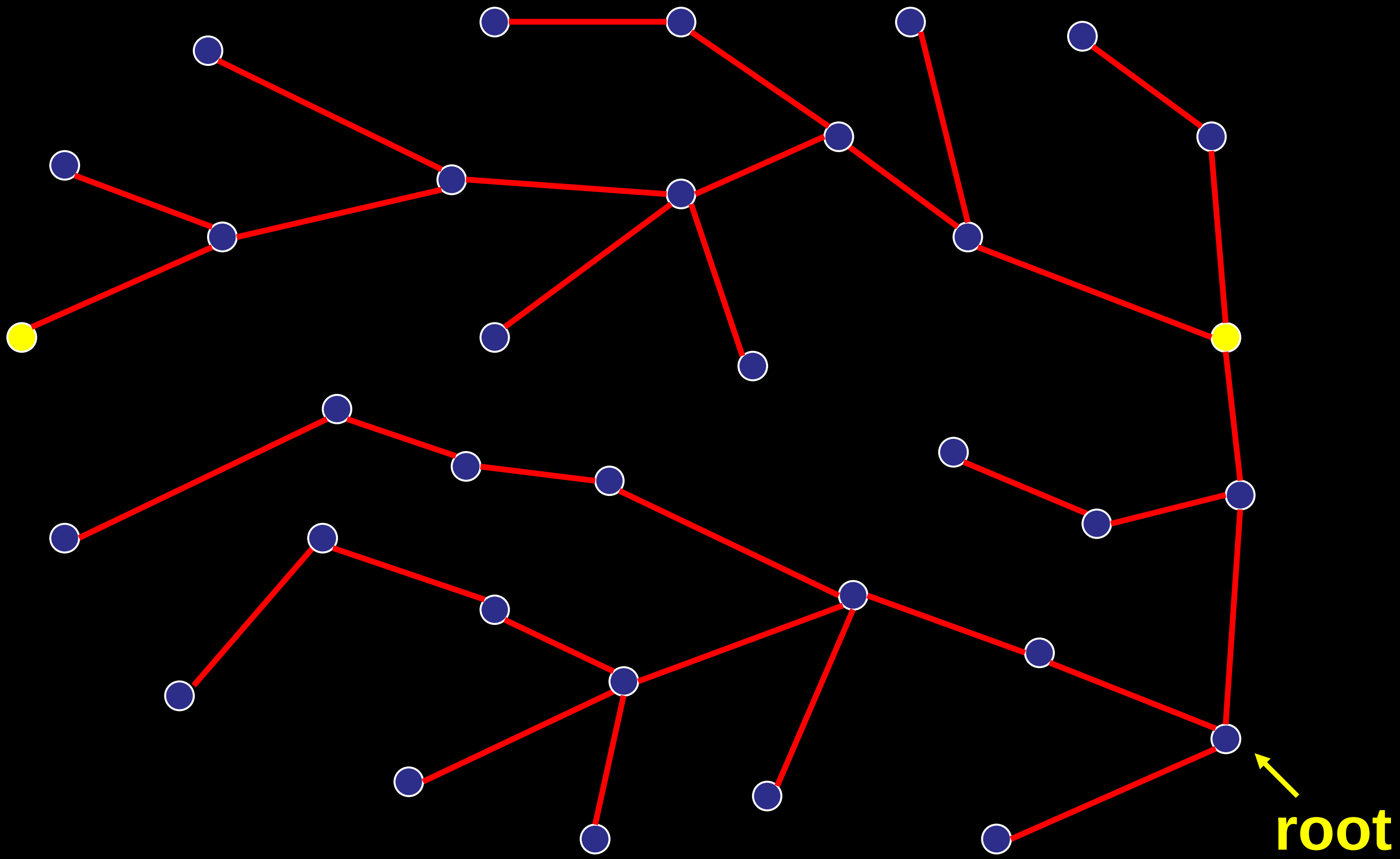
Distributed Spanning Tree

Overview: GDSTR



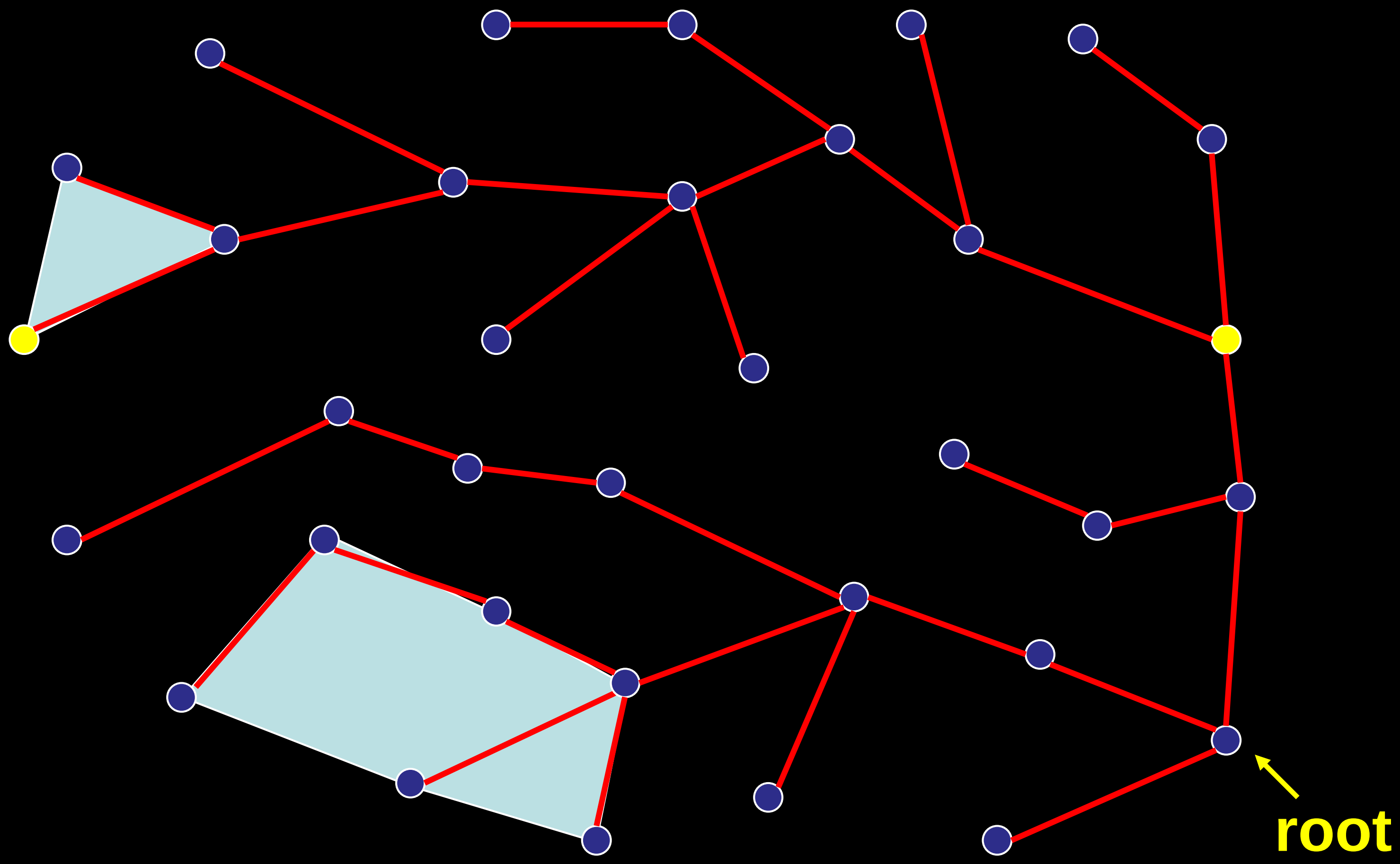
Distributed Spanning Tree

Overview: GDSTR



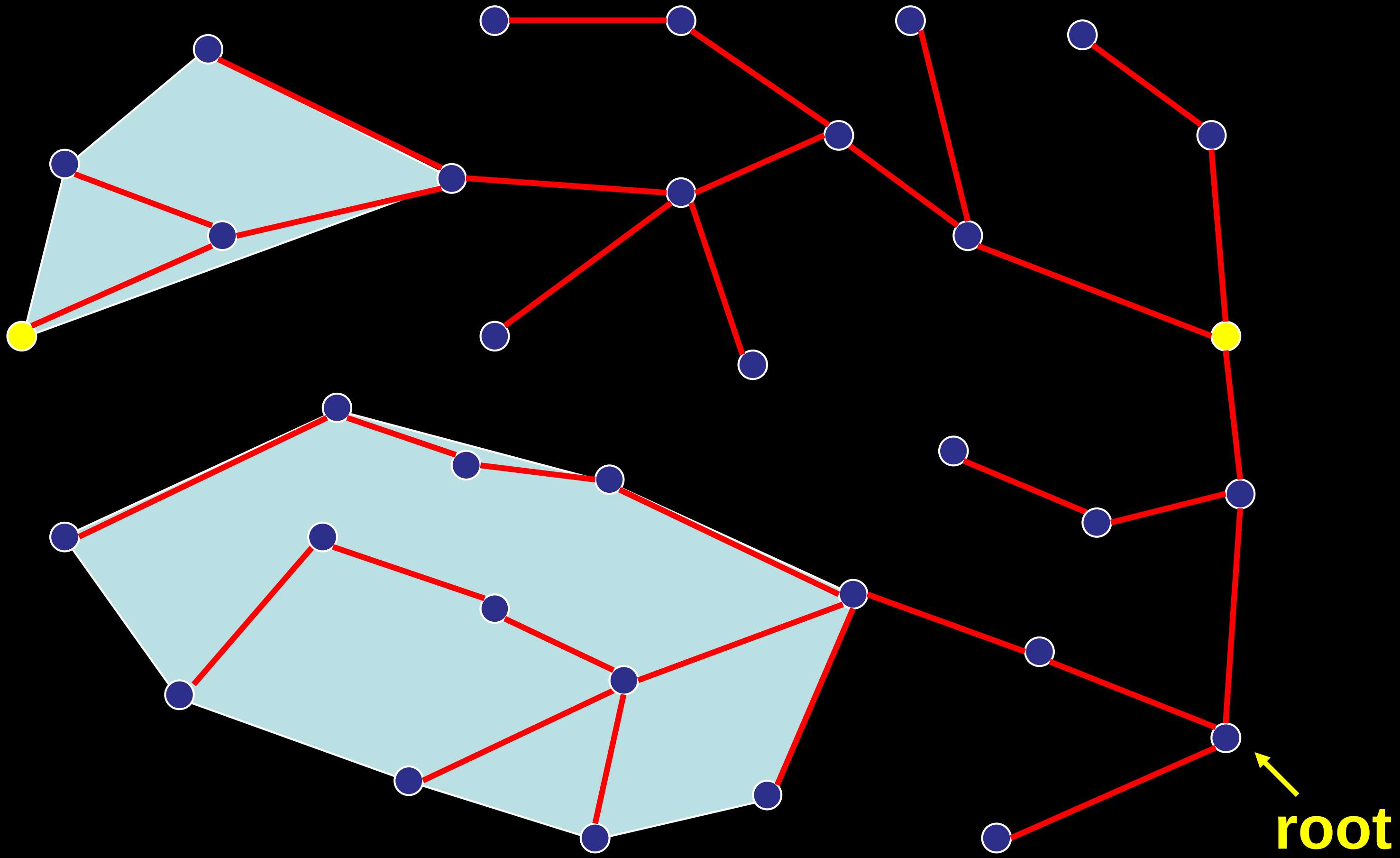
Aggregate coordinates with convex hulls

Overview: GDSTR



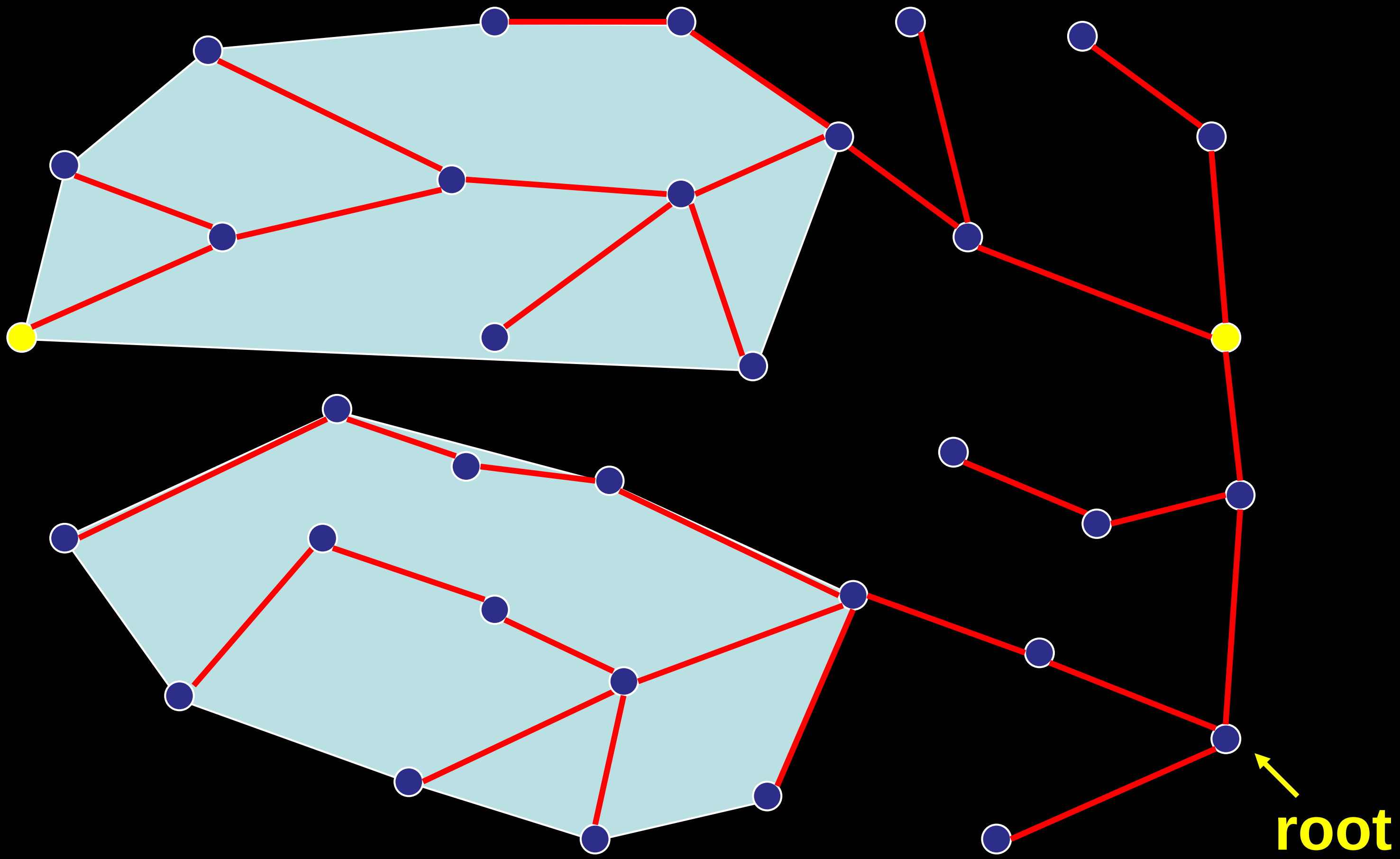
Aggregate coordinates with convex hulls

Overview: GDSTR



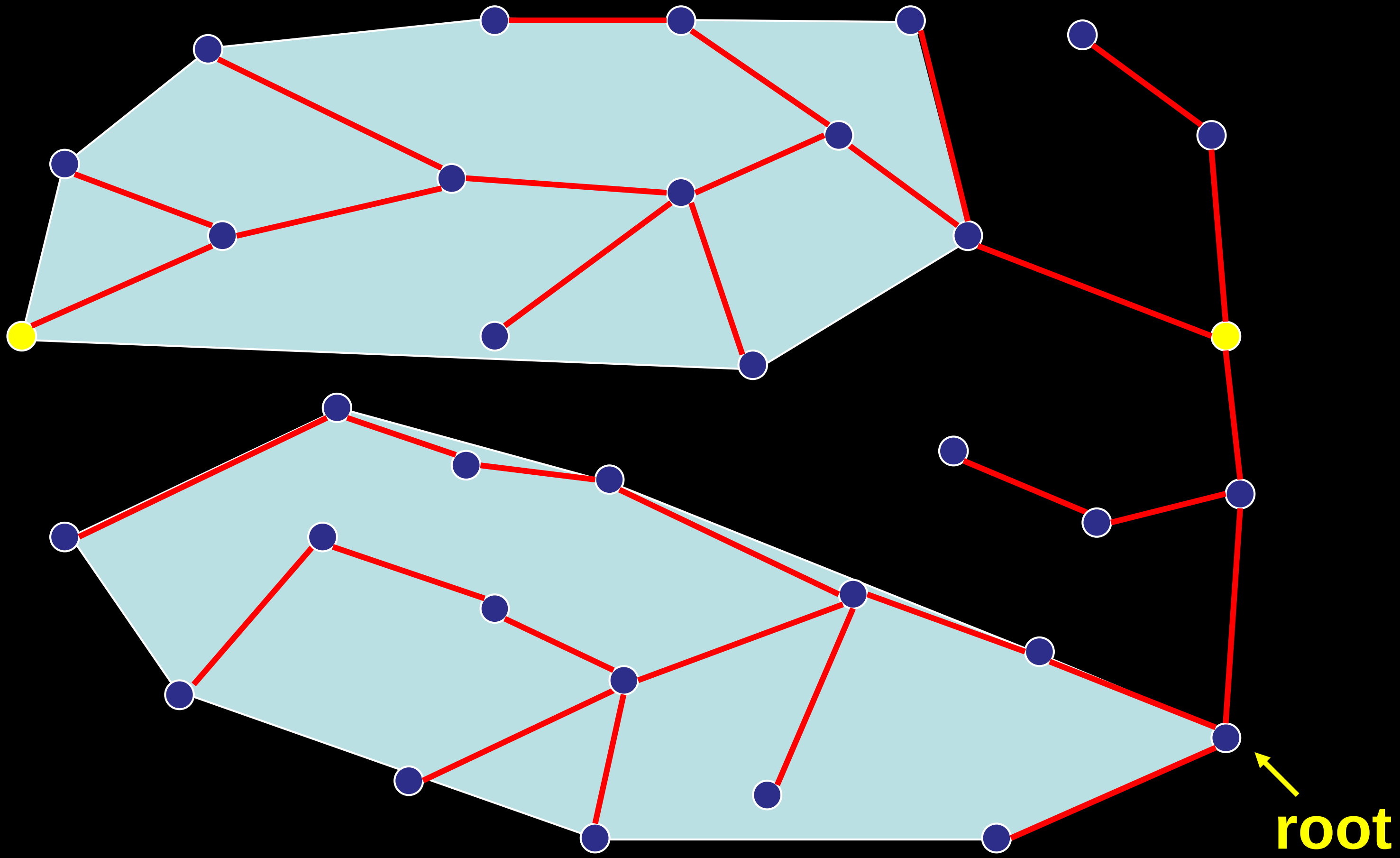
Aggregate coordinates with convex hulls

Overview: GDSTR



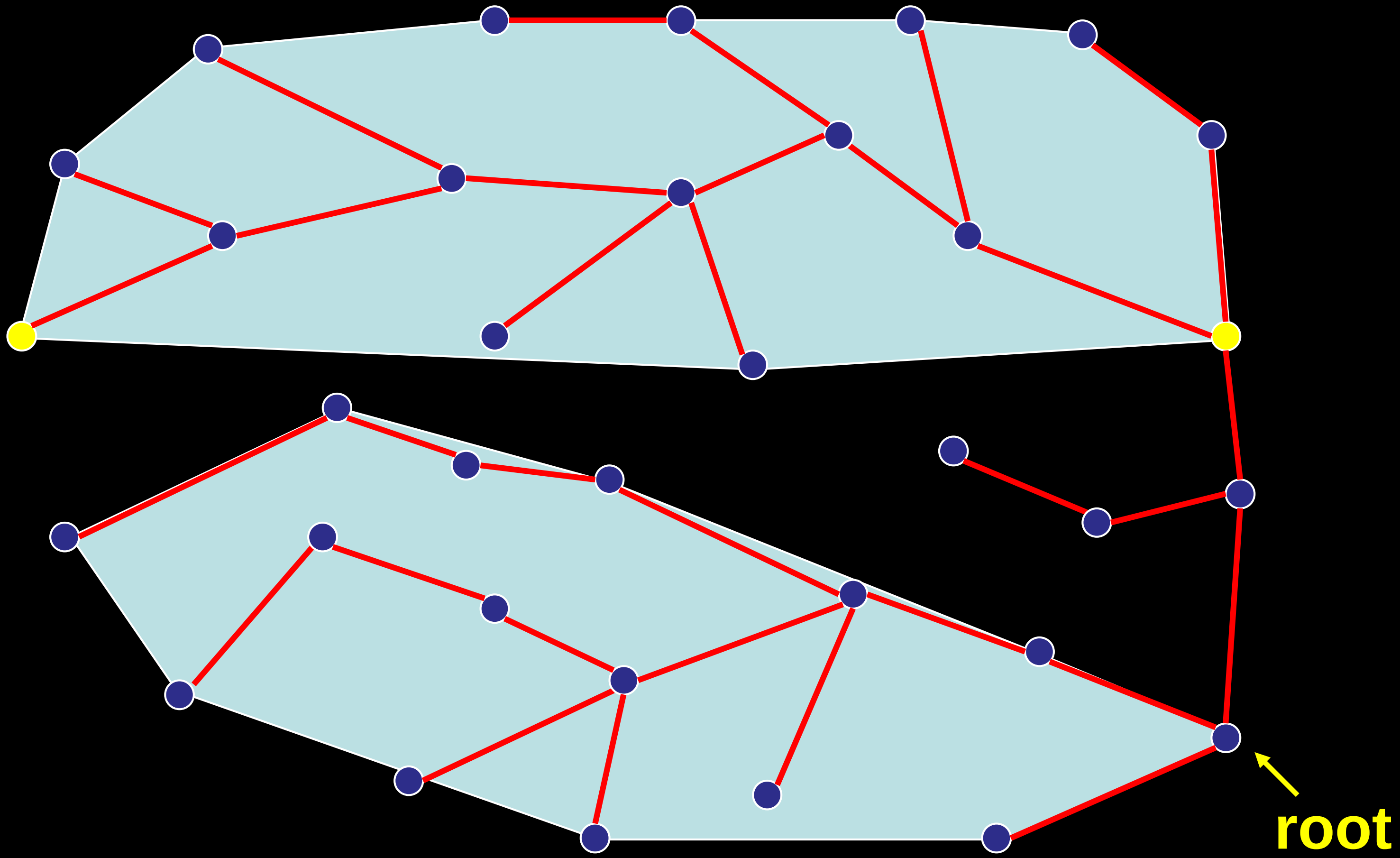
Aggregate coordinates with convex hulls

Overview: GDSTR



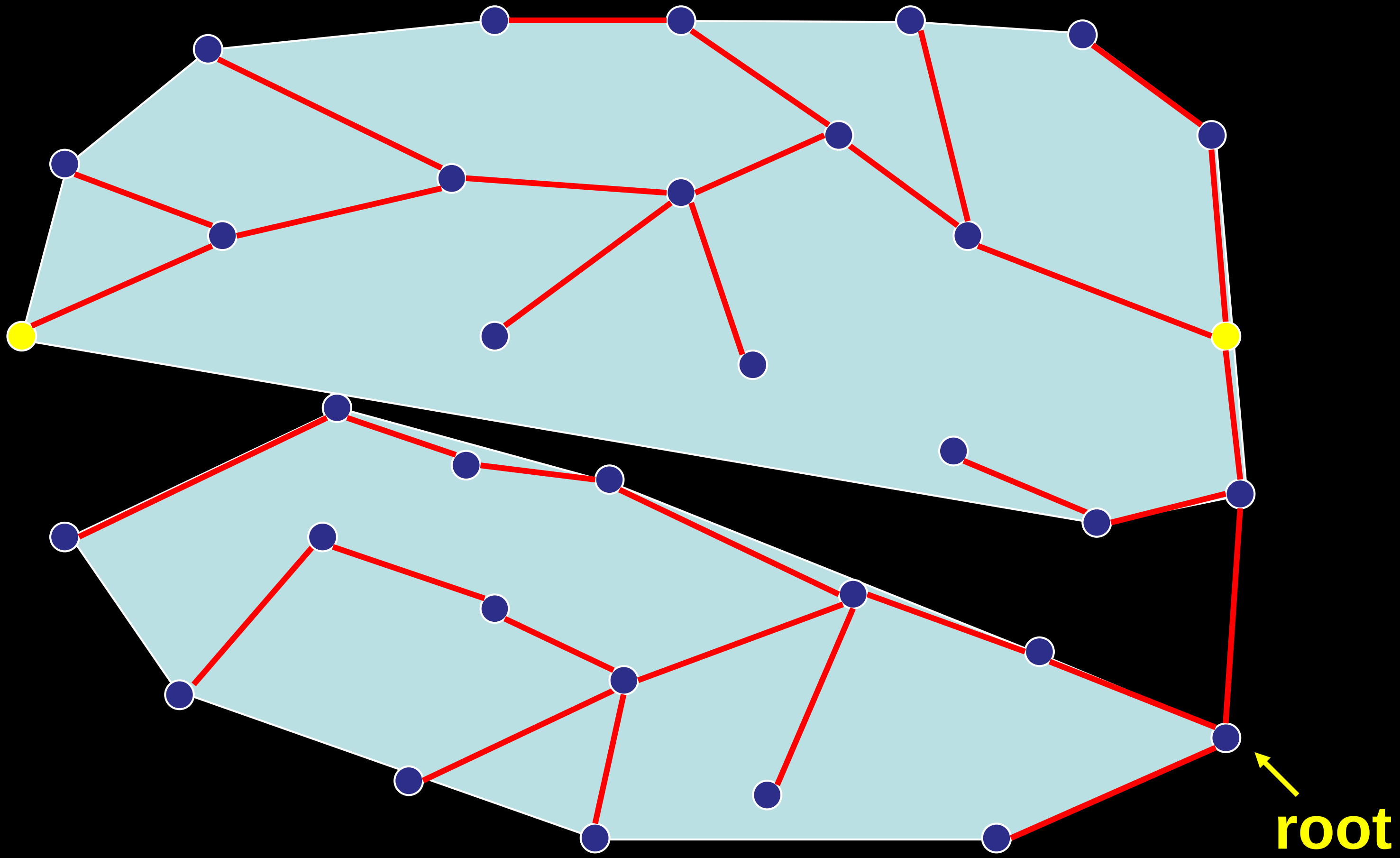
Aggregate coordinates with convex hulls

Overview: GDSTR



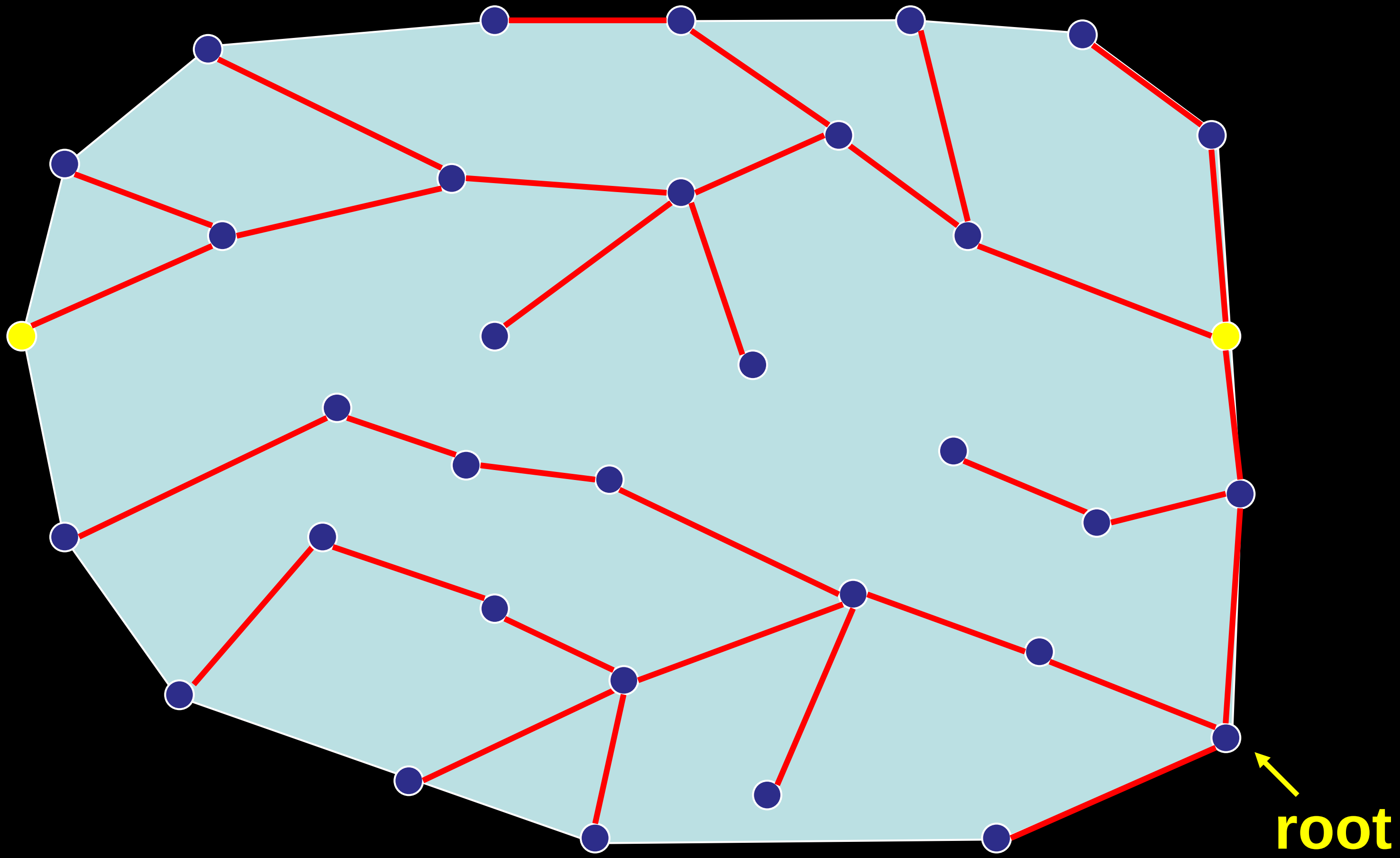
Aggregate coordinates with convex hulls

Overview: GDSTR



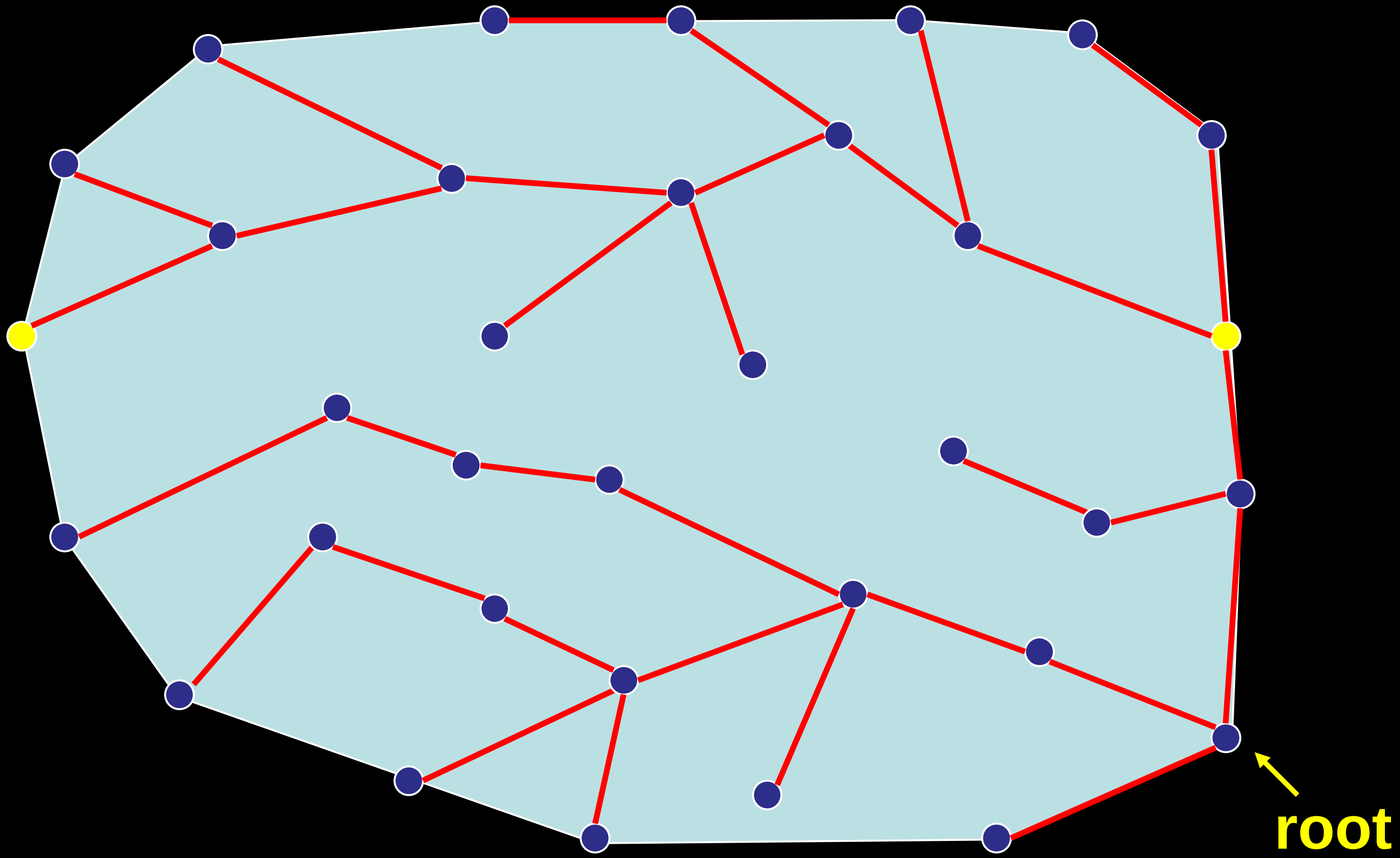
Aggregate coordinates with convex hulls

Overview: GDSTR



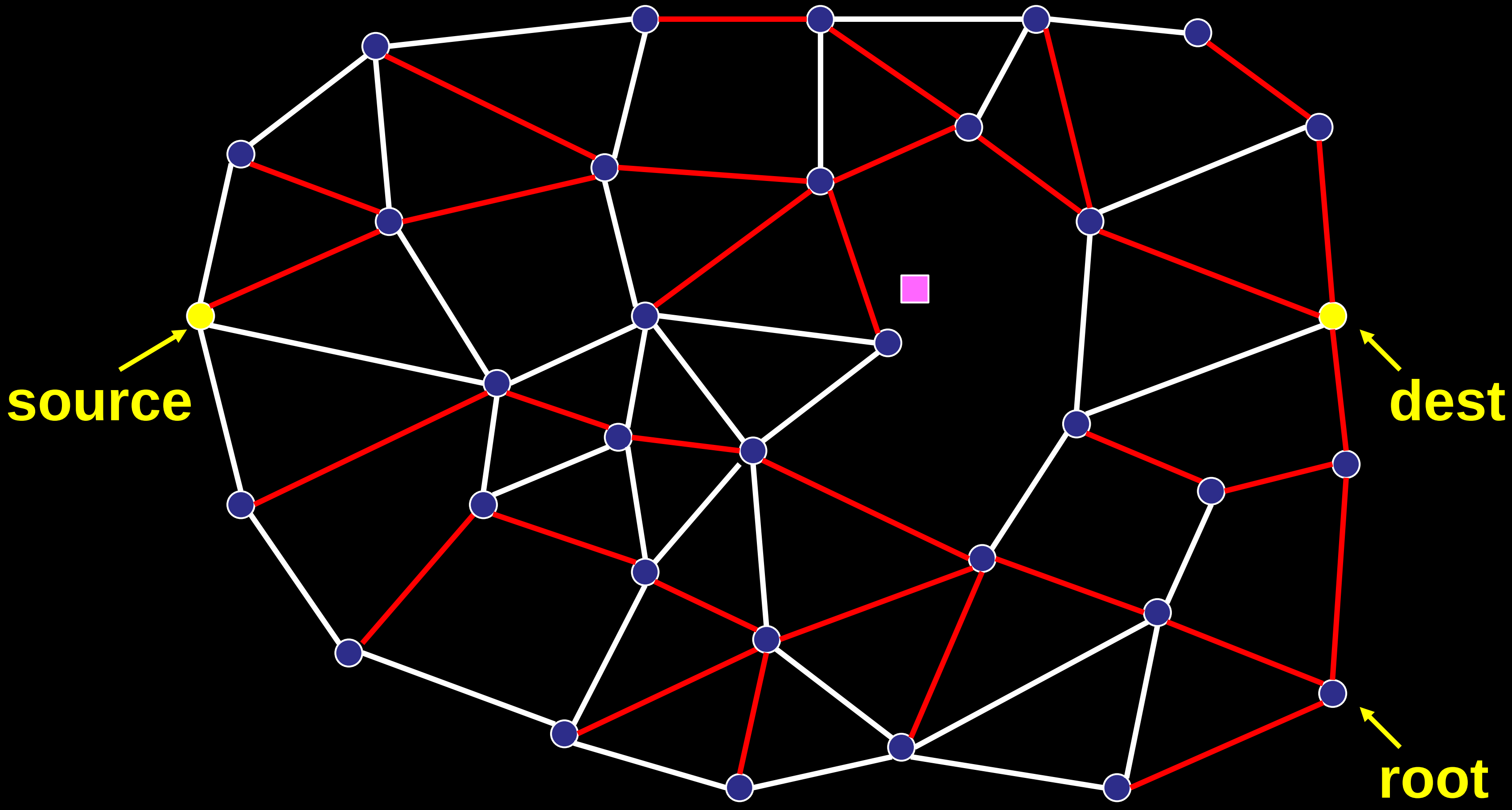
Aggregate coordinates with convex hulls

Overview: GDSTR



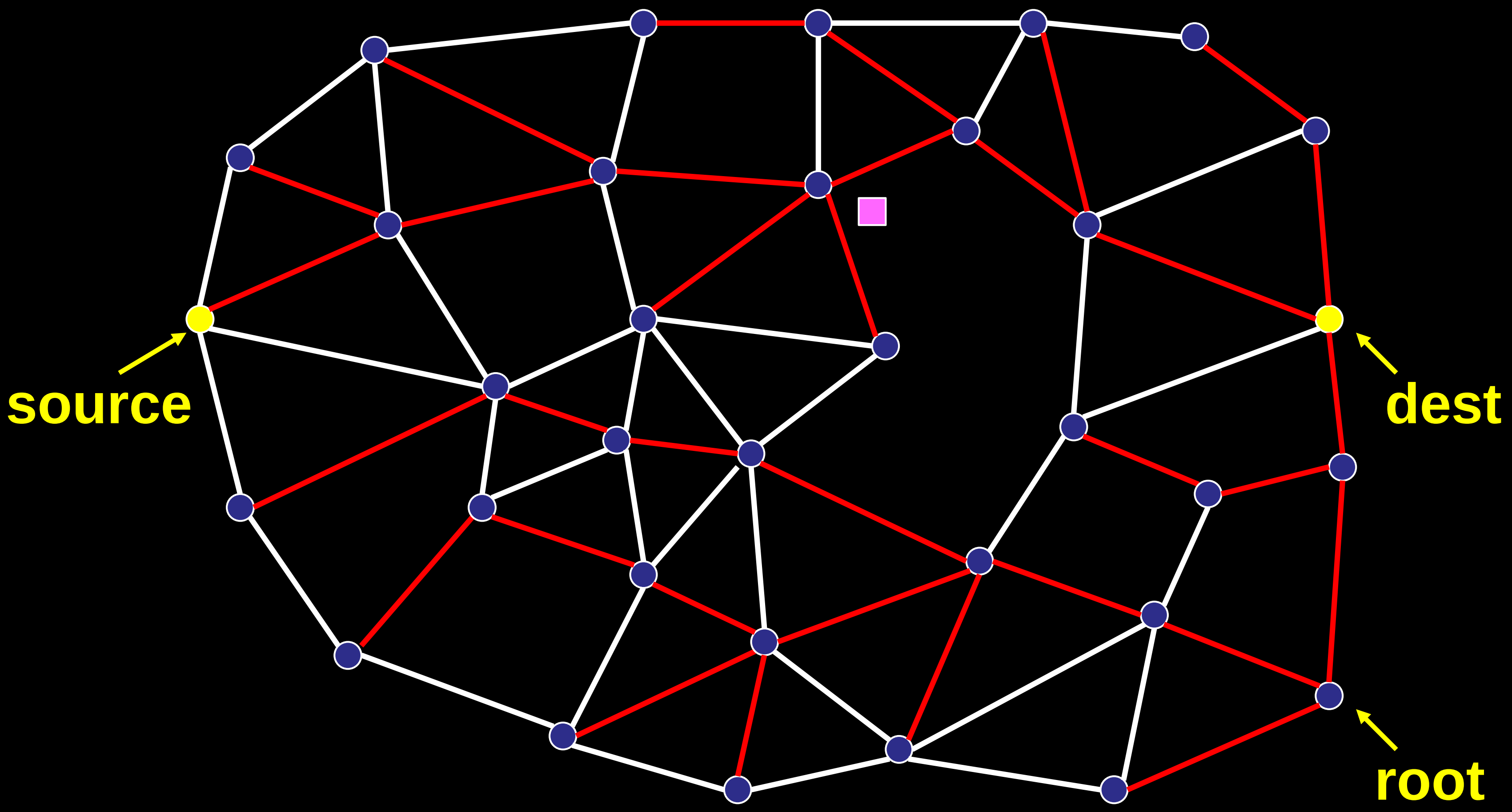
Hull Tree

Overview: GDSTR



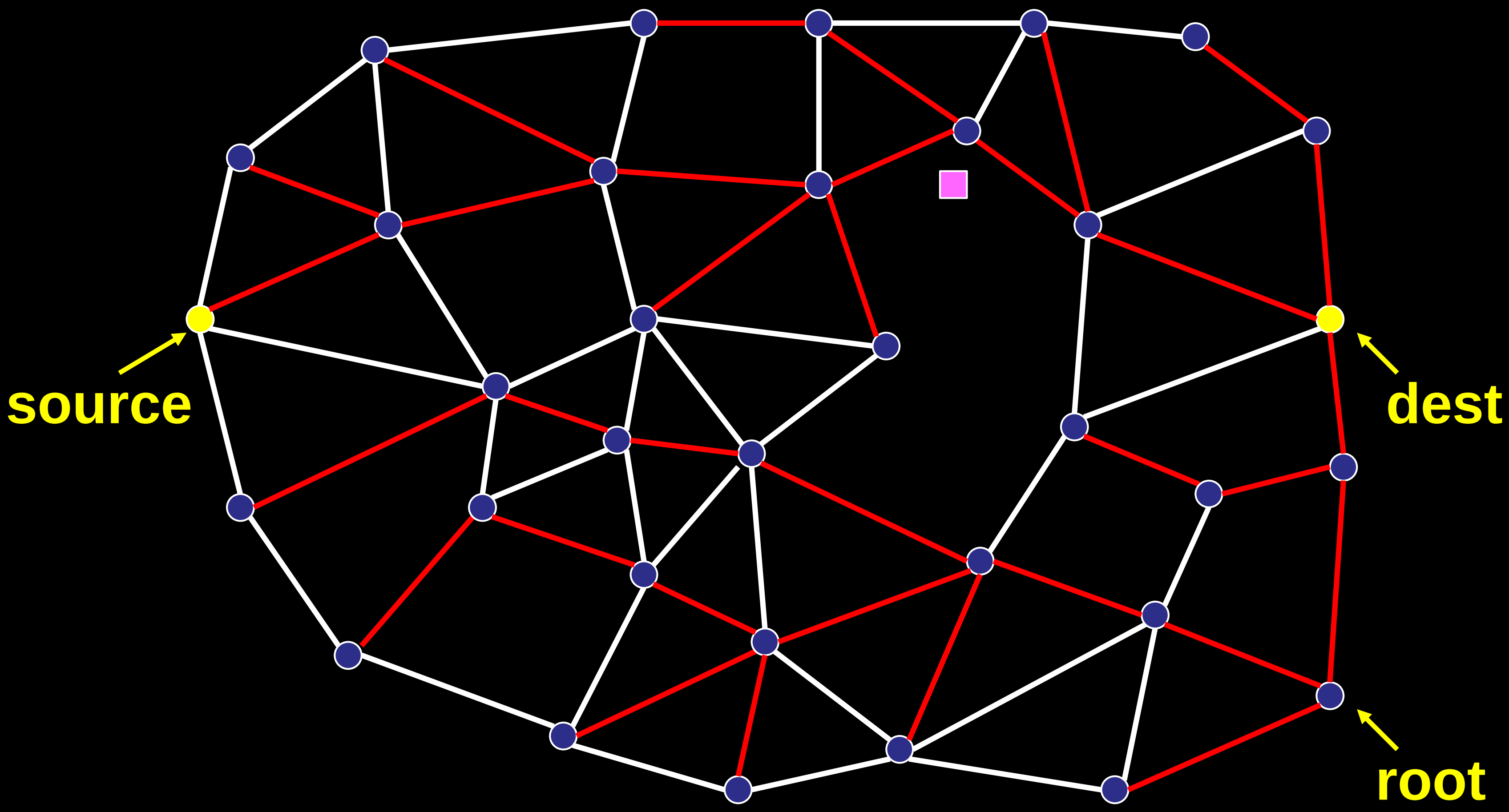
Remember minimum ⇒ tree traversal

Overview: GDSTR



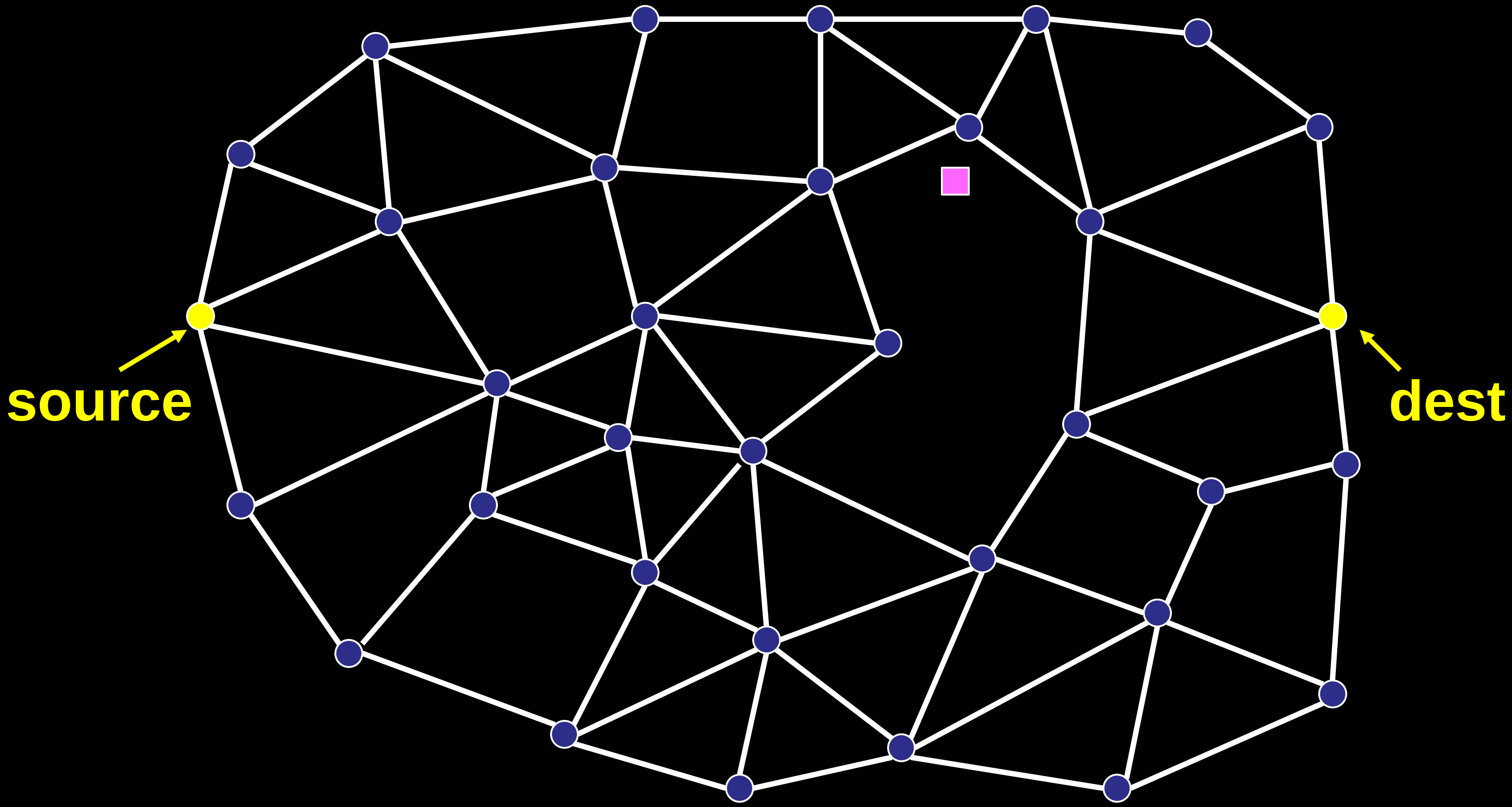
Tree Traversal

Overview: GDSTR



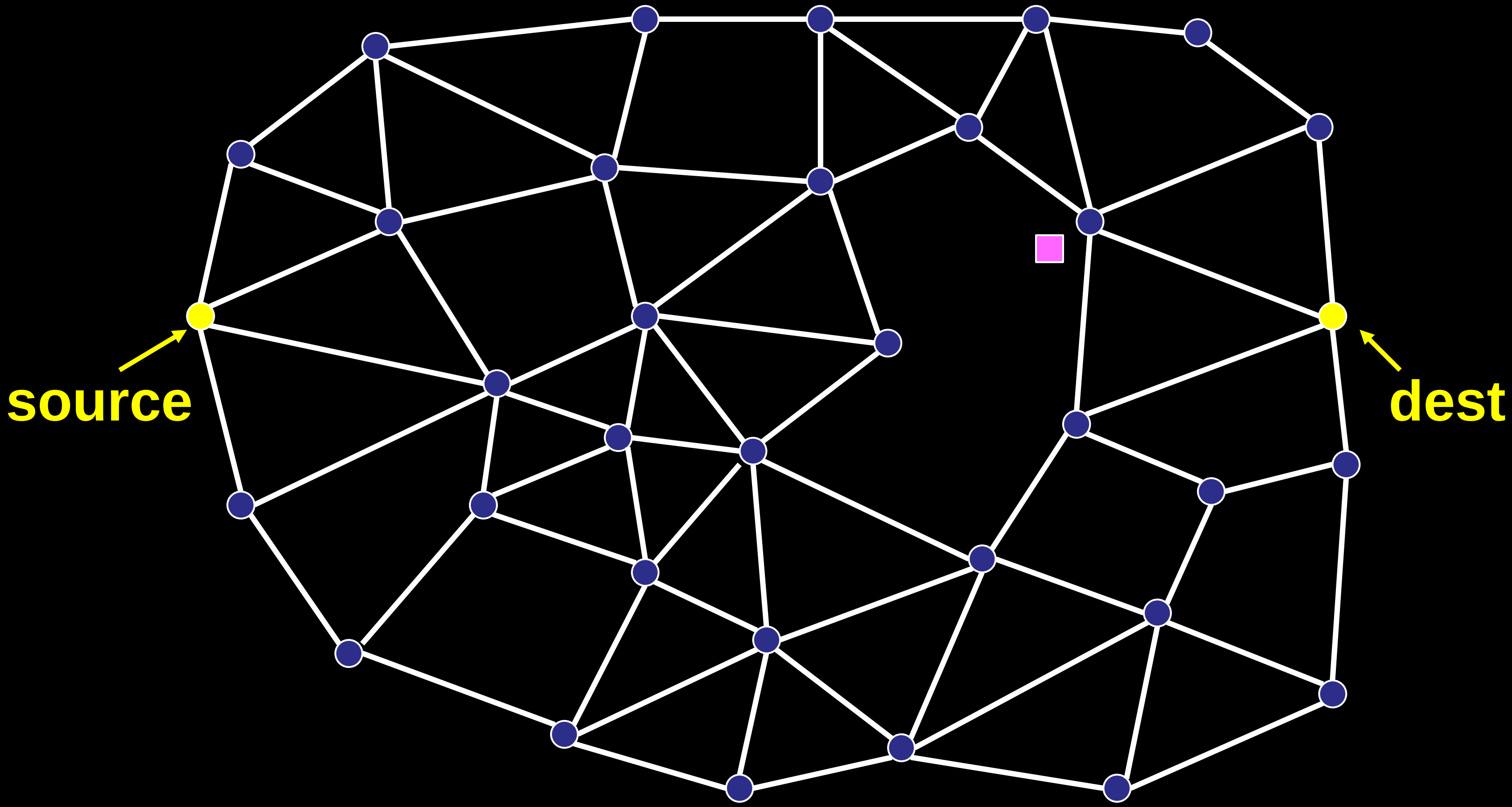
Tree Traversal

Overview: GDSTR



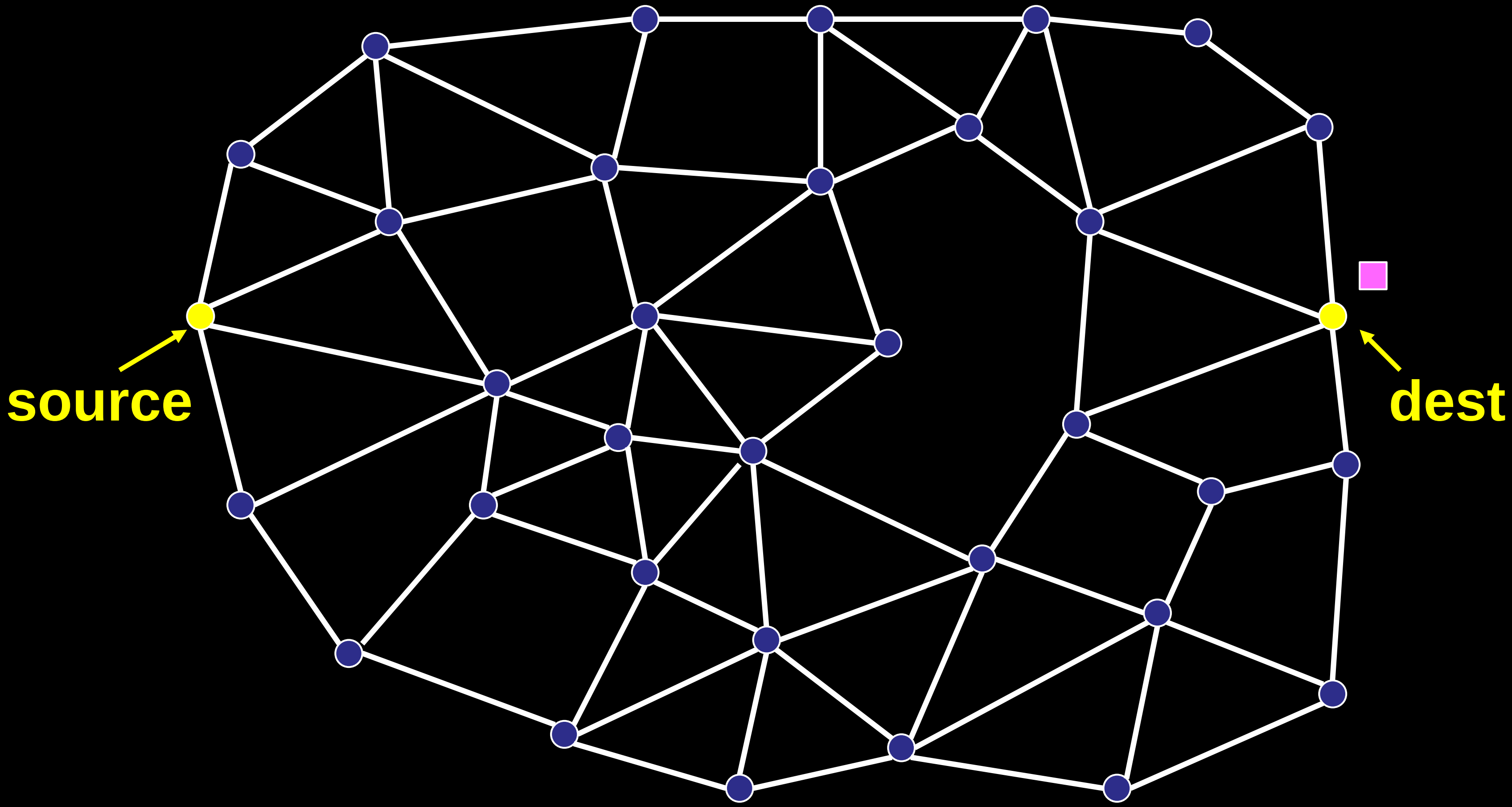
Back to Greedy Forwarding!

Overview: GDSTR



Back to Greedy Forwarding!

Overview: GDSTR

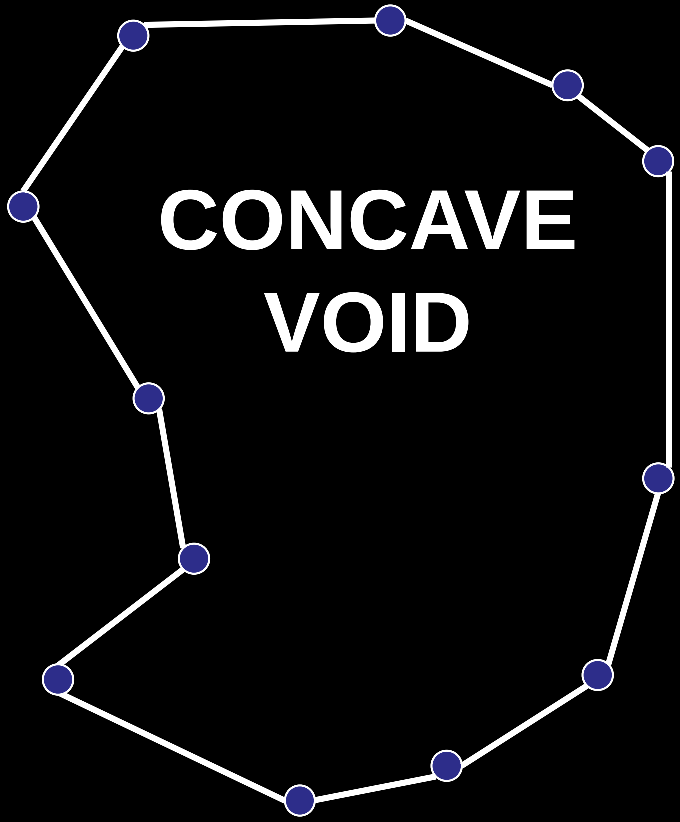


Done!!

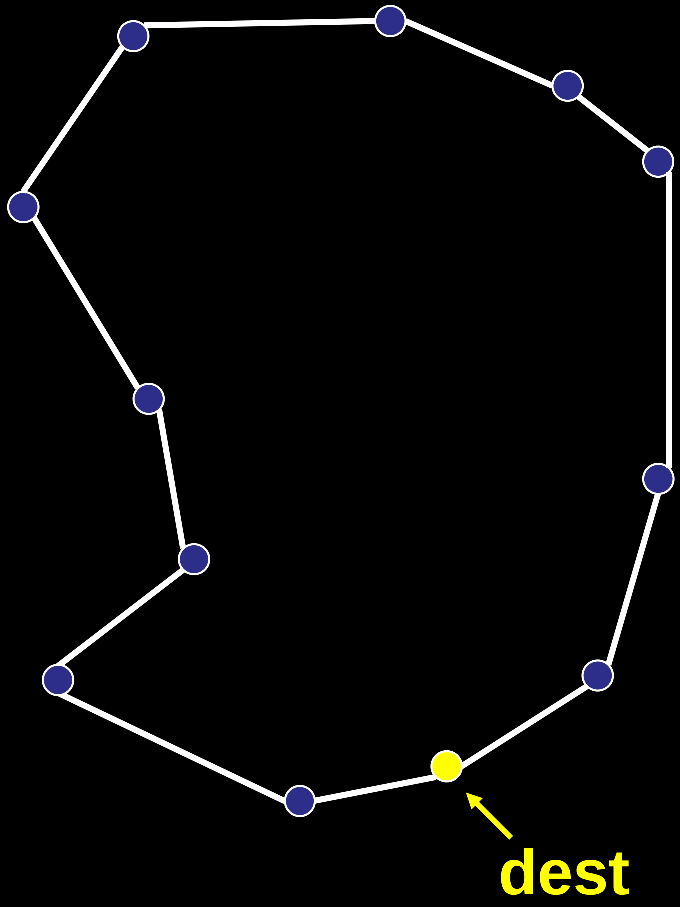
Why do hull trees work?

- Used only to escape from local minimum
- Cheap to build – *O(log n)*

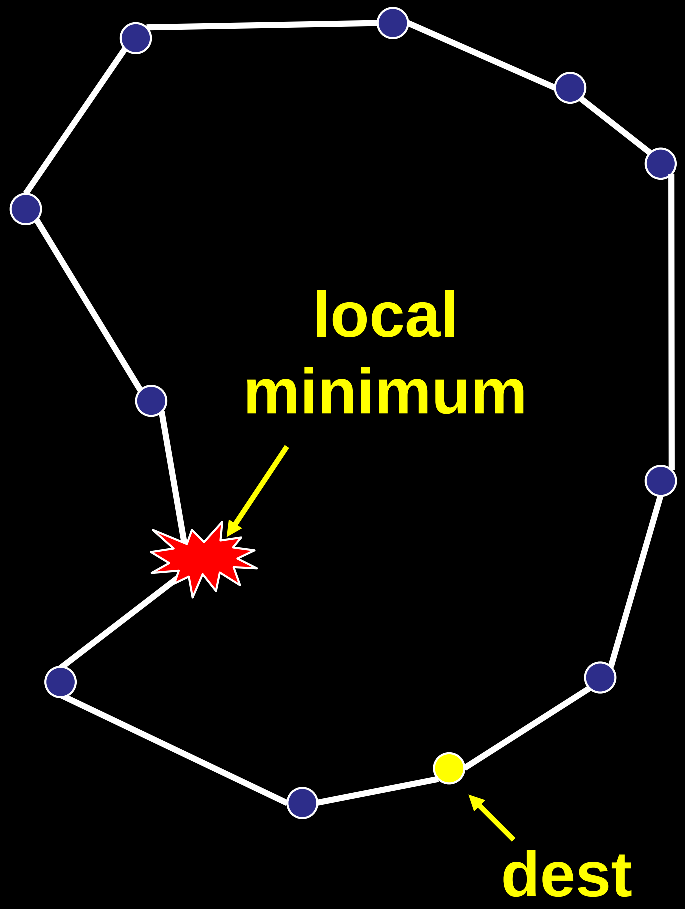
Caveat



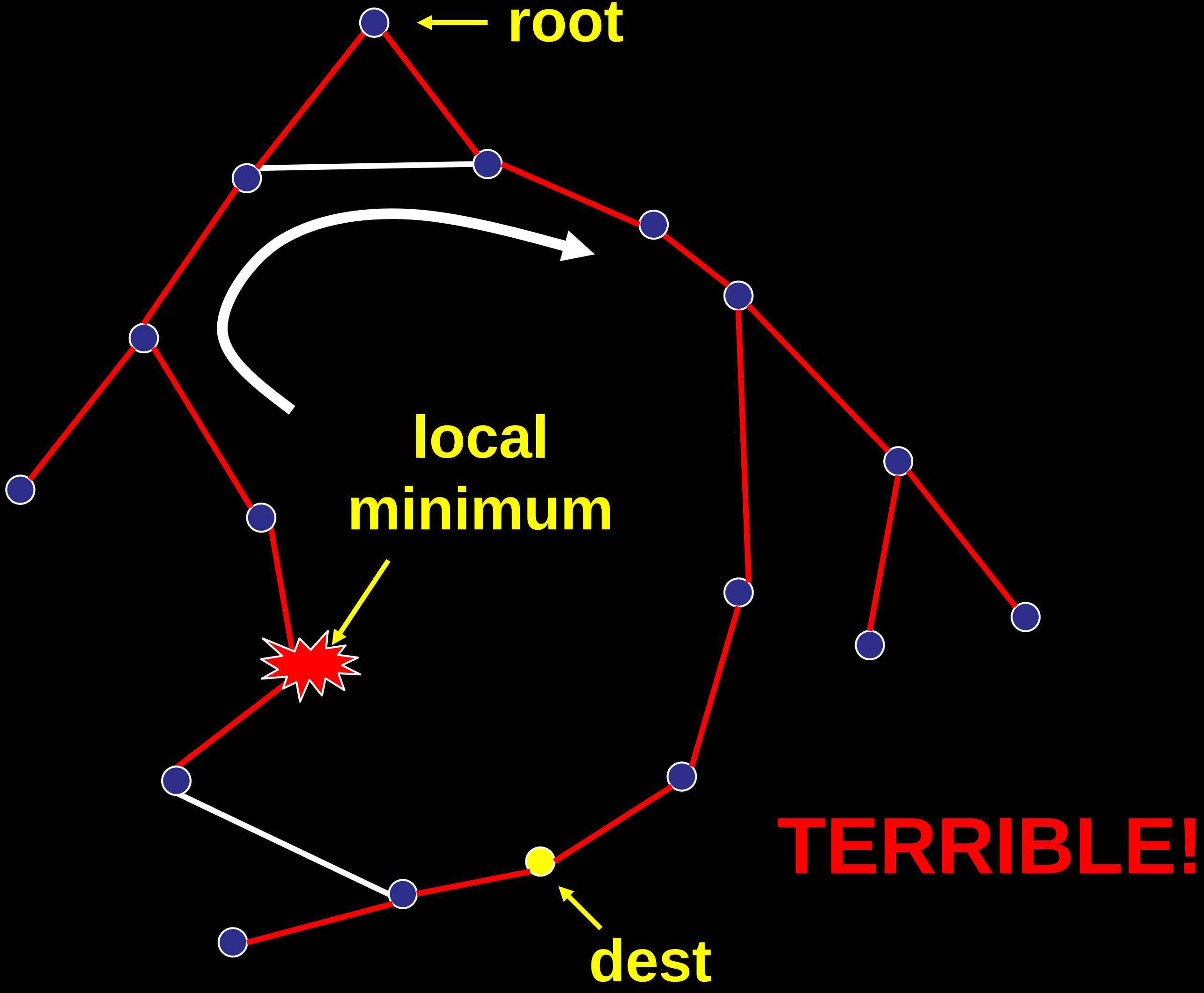
Caveat



Caveat

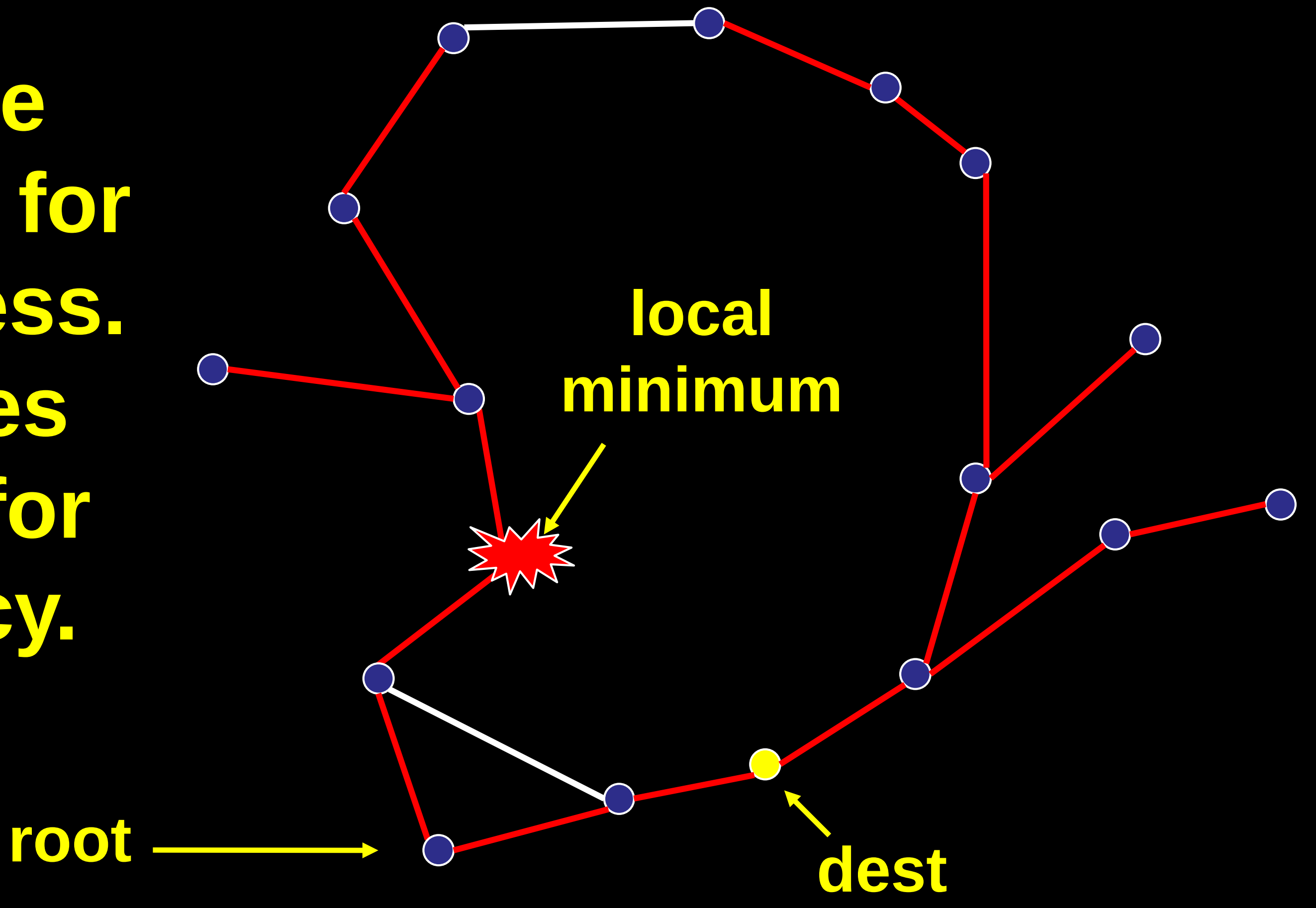


Caveat



Need TWO hull trees rooted at opposite ends

One tree sufficient for correctness. Two trees needed for efficiency.



Our Approach

Extend GDSTR
to 3D

Complications!

Challenges

(Why is it hard in TinyOS?)

- TinyOS does not support dynamic memory allocation
- CC2420 radio supports up to 128 bytes in size and has a limited data rate
- Limited DRAM and flash memory
- Precision of floating point operations is limited

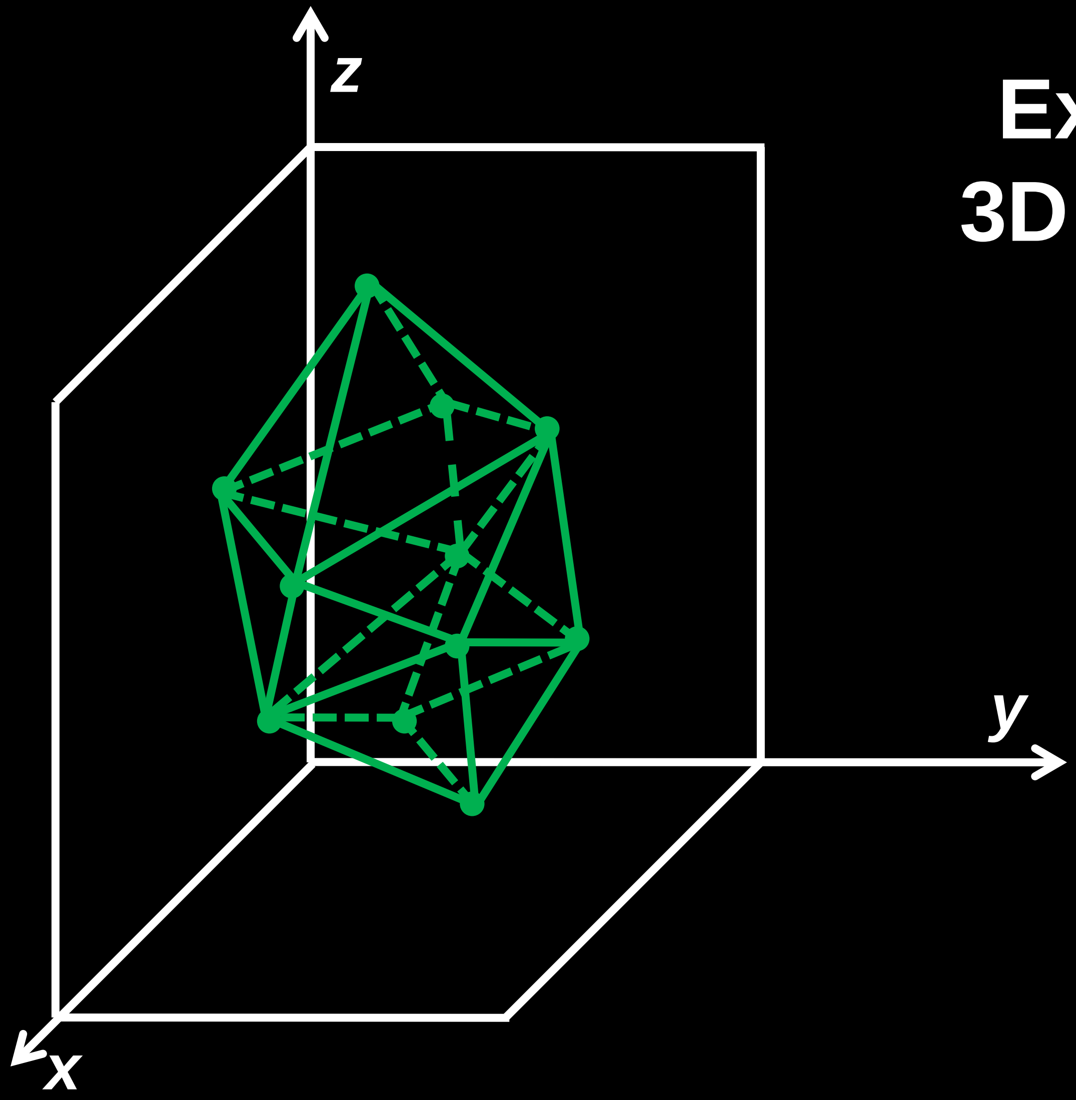
Naïve Implementation of 3D Convex Hull

- Computations are costly
- Need to store auxiliary data structures for efficiency ⇒ storage costly
- Messages too big

Key ideas

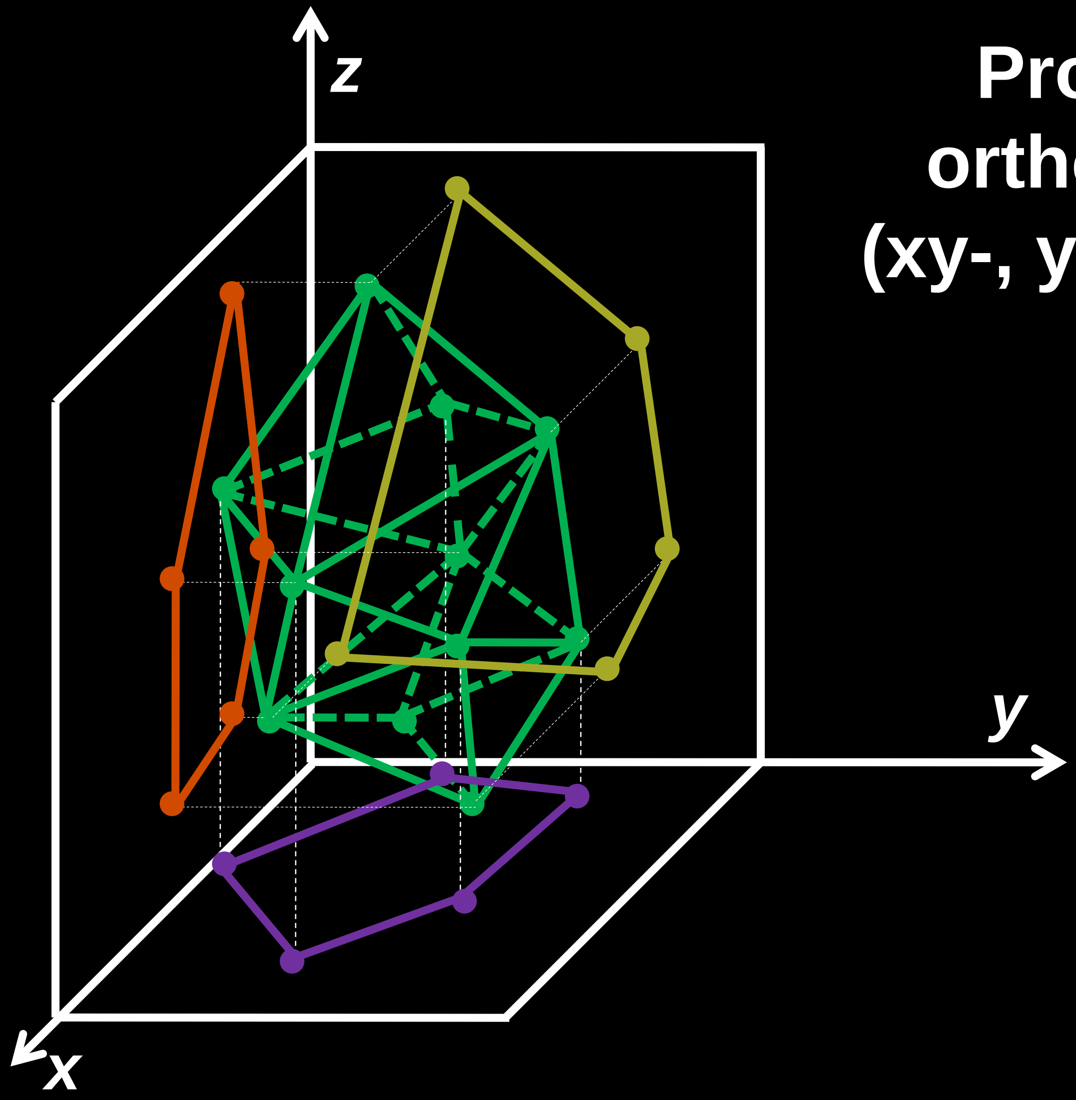
1. Approximate 3D Convex Hull with 2 x 2D Convex Hull
2. Use two-hop greedy forwarding
3. Simplify (details in paper)

GDSTR-3D



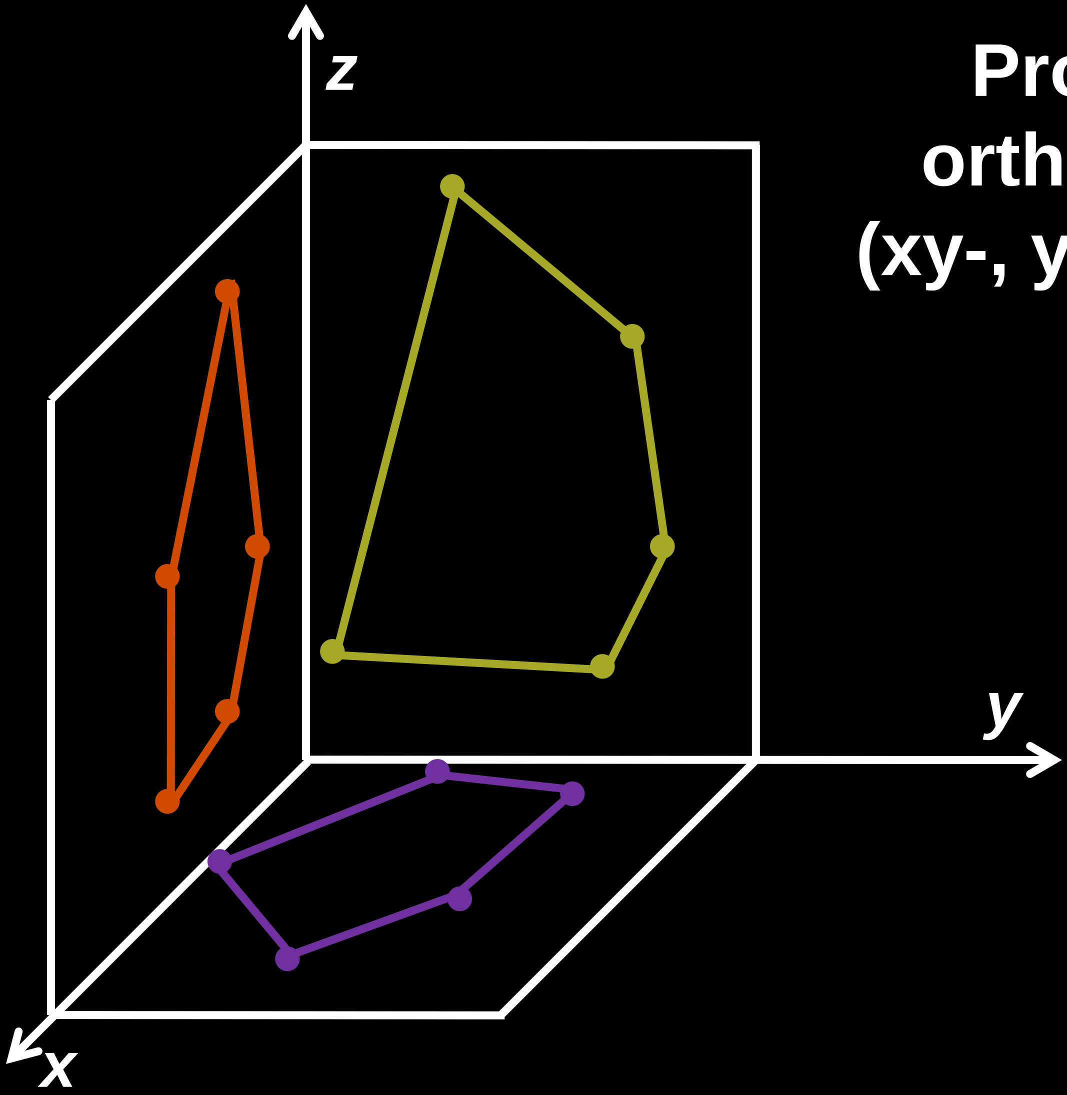
Example of a
3D convex hull

GDSTR-3D



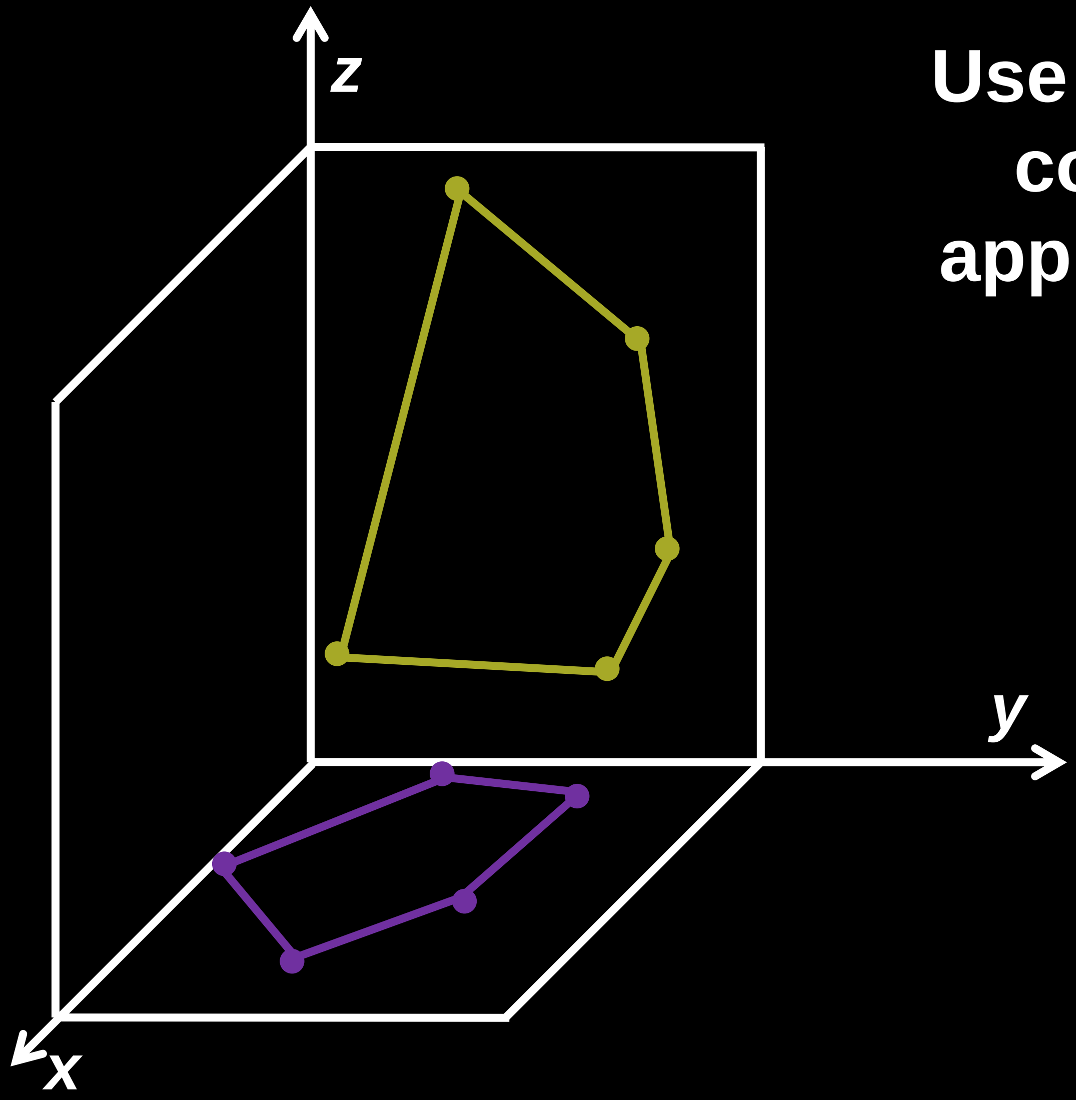
Projection onto orthogonal planes (xy-, yz-, and zx-plane)

GDSTR-3D



Projection onto orthogonal planes (xy-, yz-, and zx-plane)

GDSTR-3D



Use two of these 2D convex hulls to approximate the 3D convex hull

Performance Evaluation

Metrics

1. Success rate
2. Hop stretch
3. Maximum Storage
4. Message Overhead

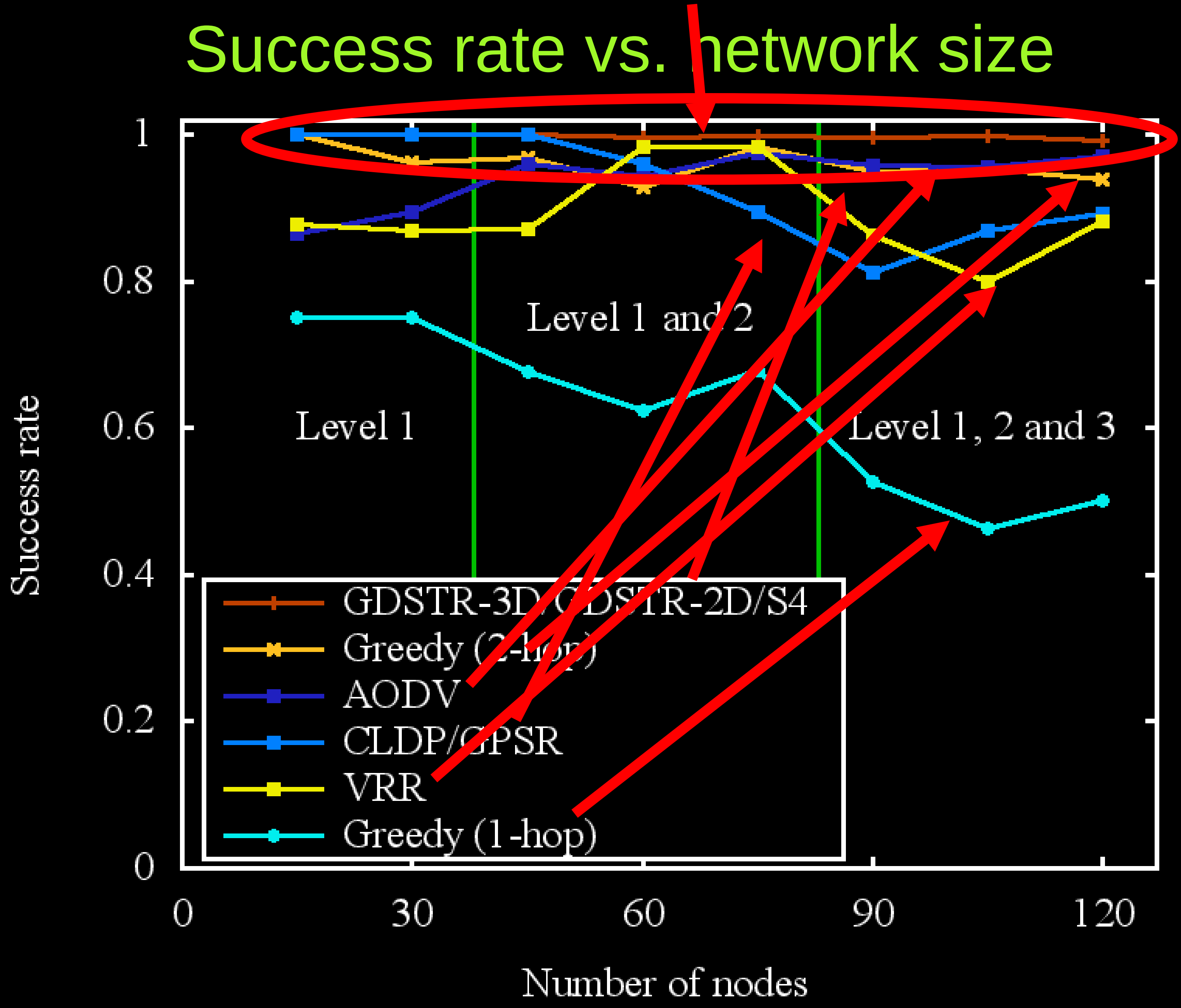
Indriya Testbed (NUS)

- 127 TelosB motes distributed over 3 floors
- Picked random subsets of nodes on 1, 2 and 3 floors

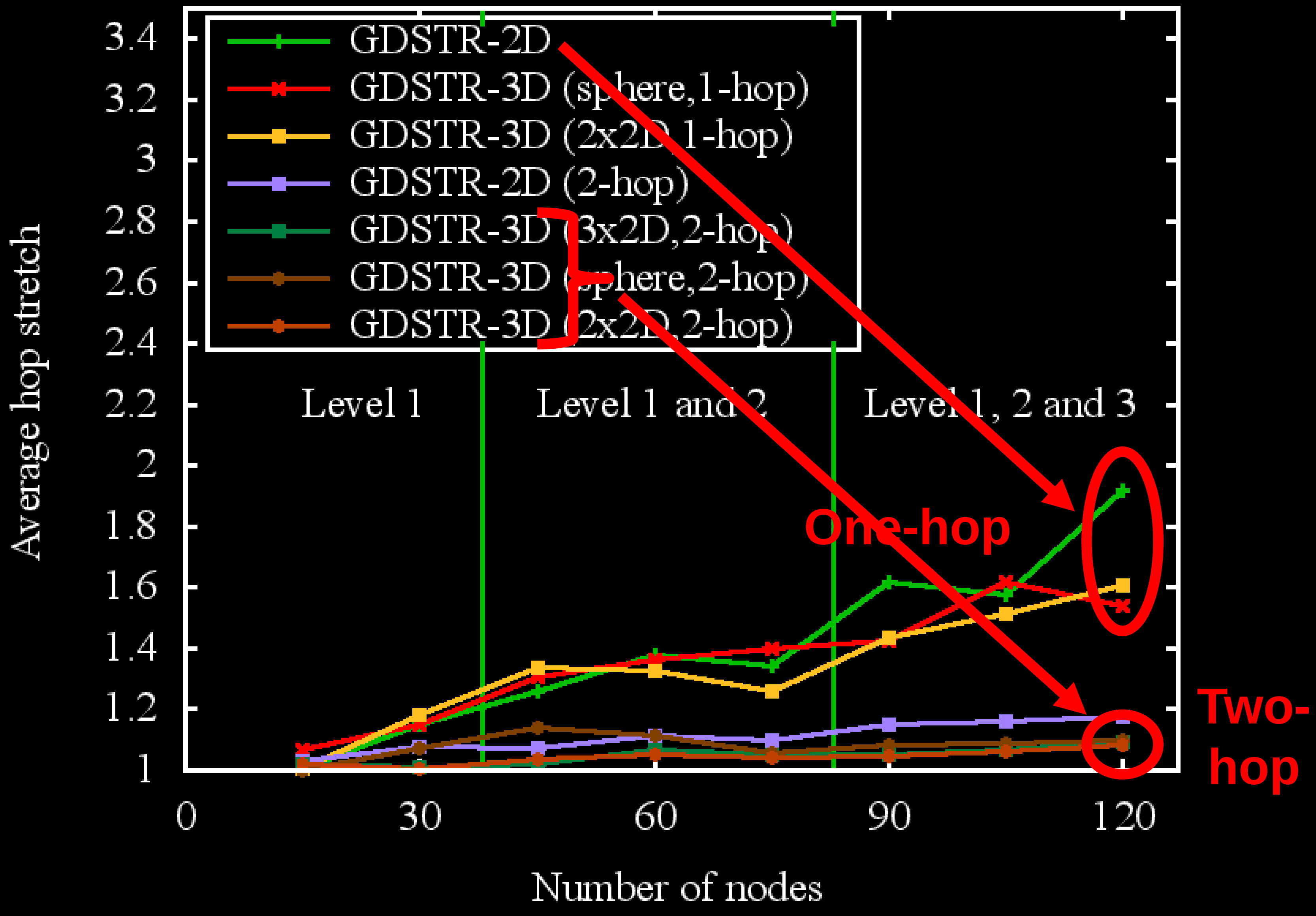
Algorithms

1. GDSTR-3D
2. GDSTR **(-2D)**
3. CLDP/GPSR **(2D Face Routing)**
4. AODV
5. VRR
6. S4

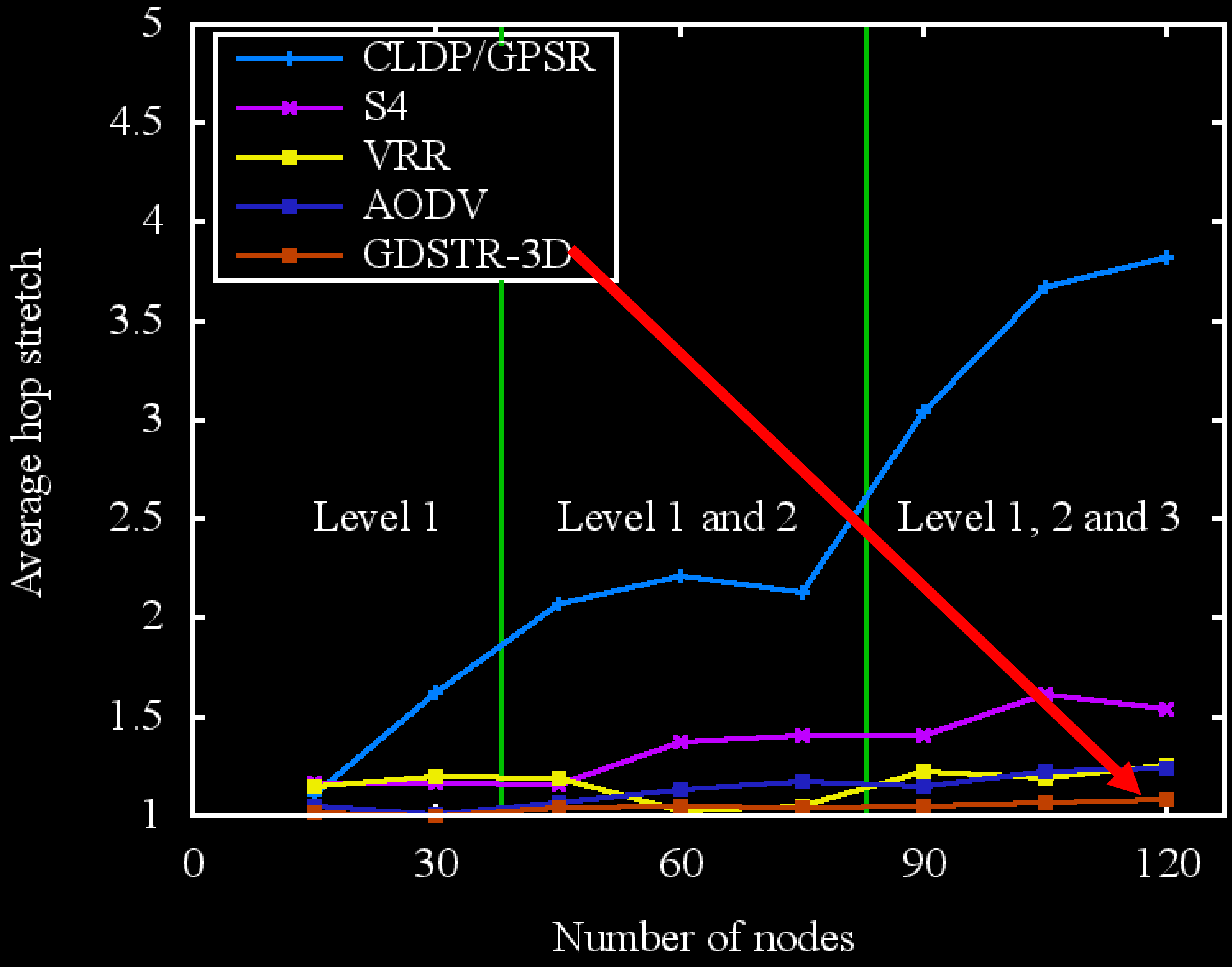
Success rate vs. network size



Hop stretch (GDSTR+) vs. network size



Hop stretch vs. network size

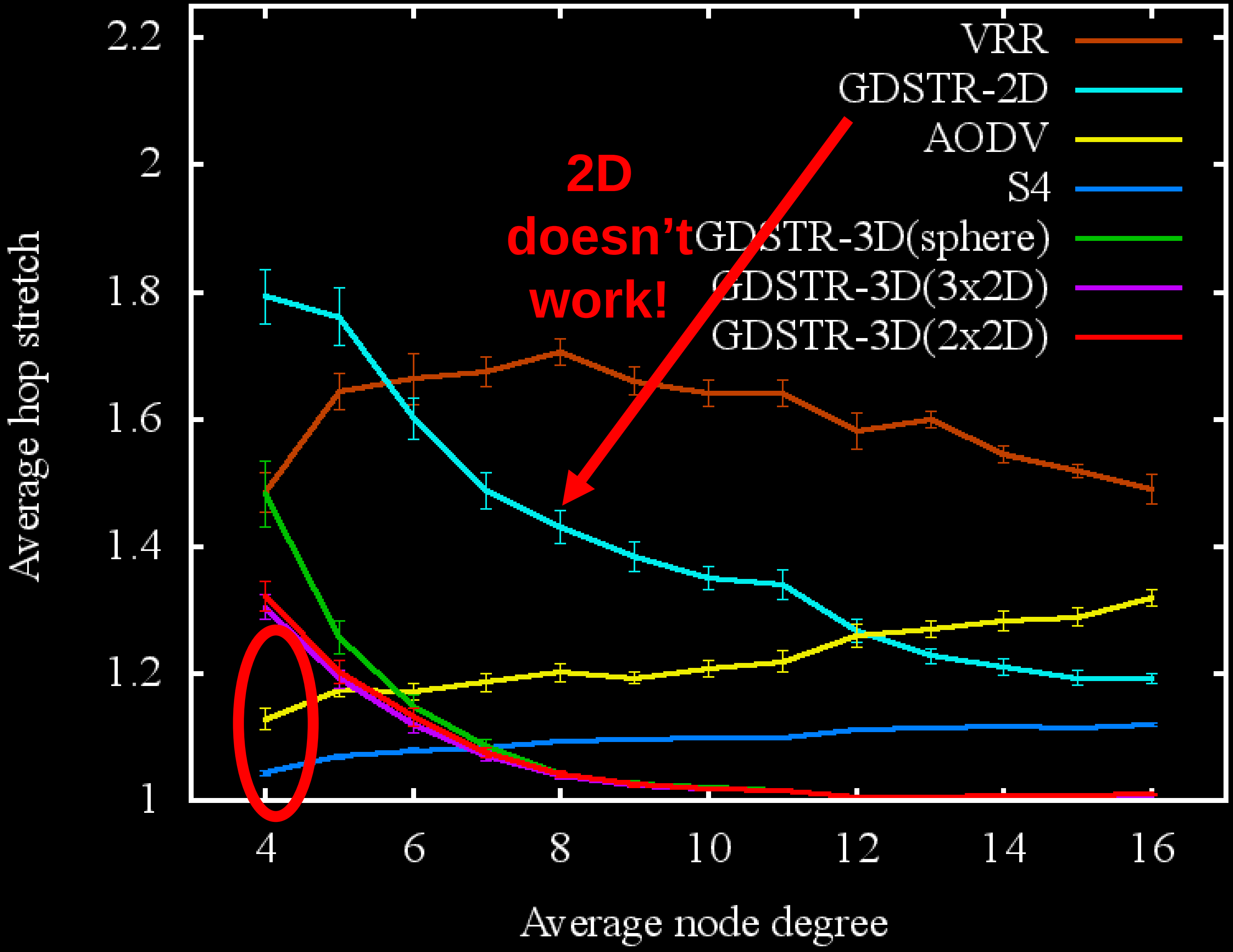


Size of compiled binaries & source code

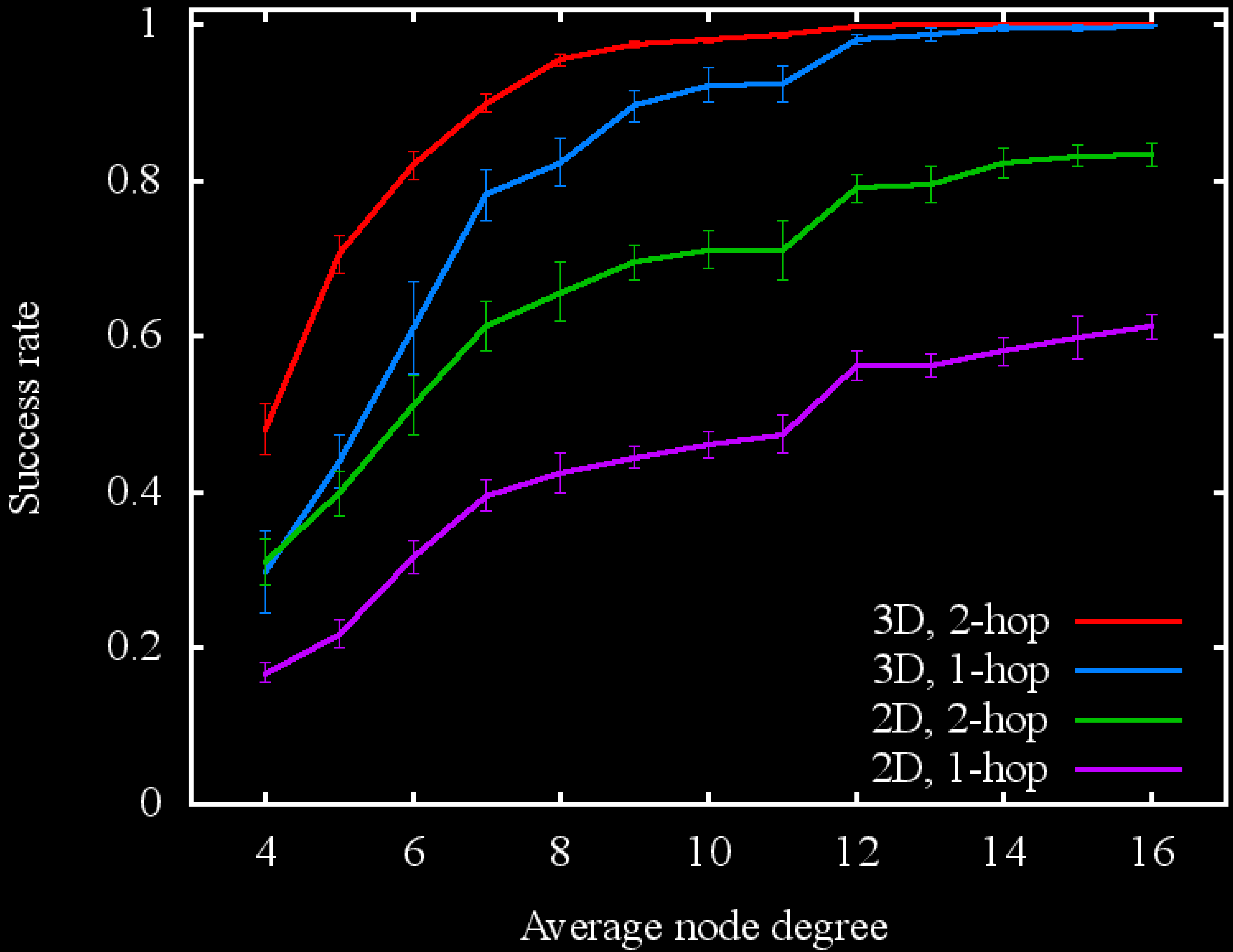
Algorithm	Compiled binary Size (KB)	Lines of code
GDSTR-3D	39.5	2,757
GDSTR	33.8	2,641
CLDP/GPSR	47.5	2,500
S4	43.2	3,997
VRR	45.1	4,135
AODV	21.1	1,294

TOSSIM Experiments

Hop stretch vs. network density

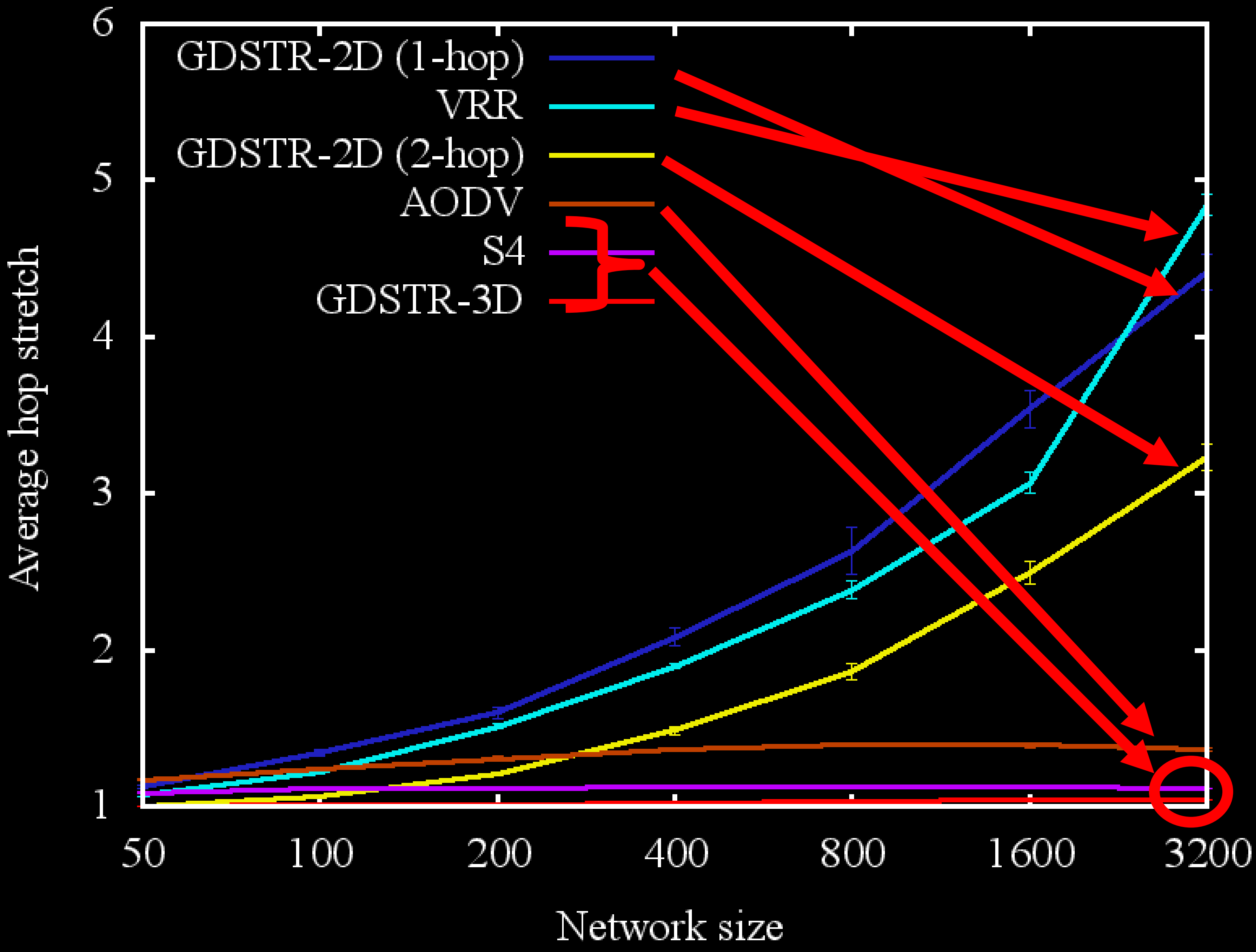


Greedy forwarding success rate vs. network density

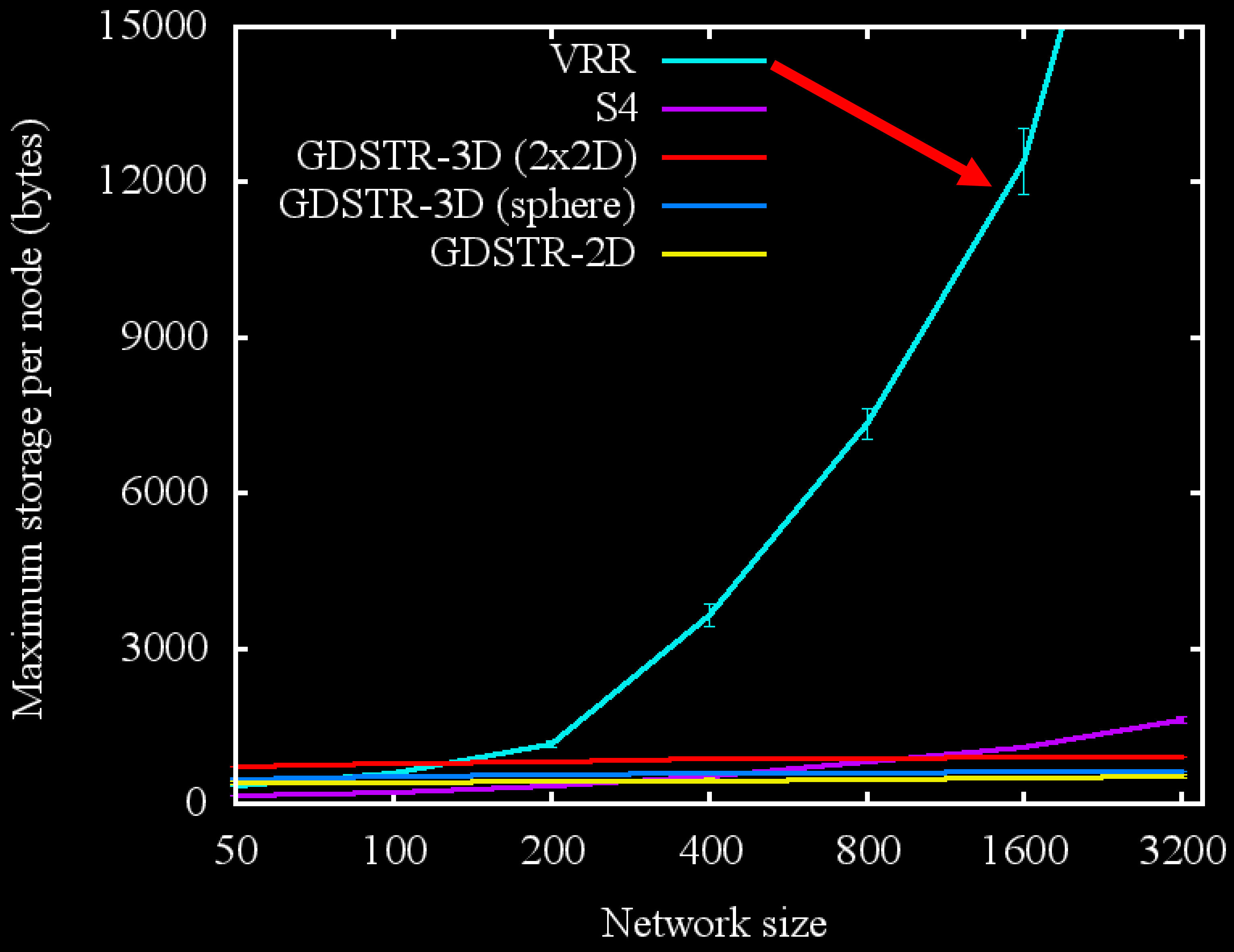


Scaling Up

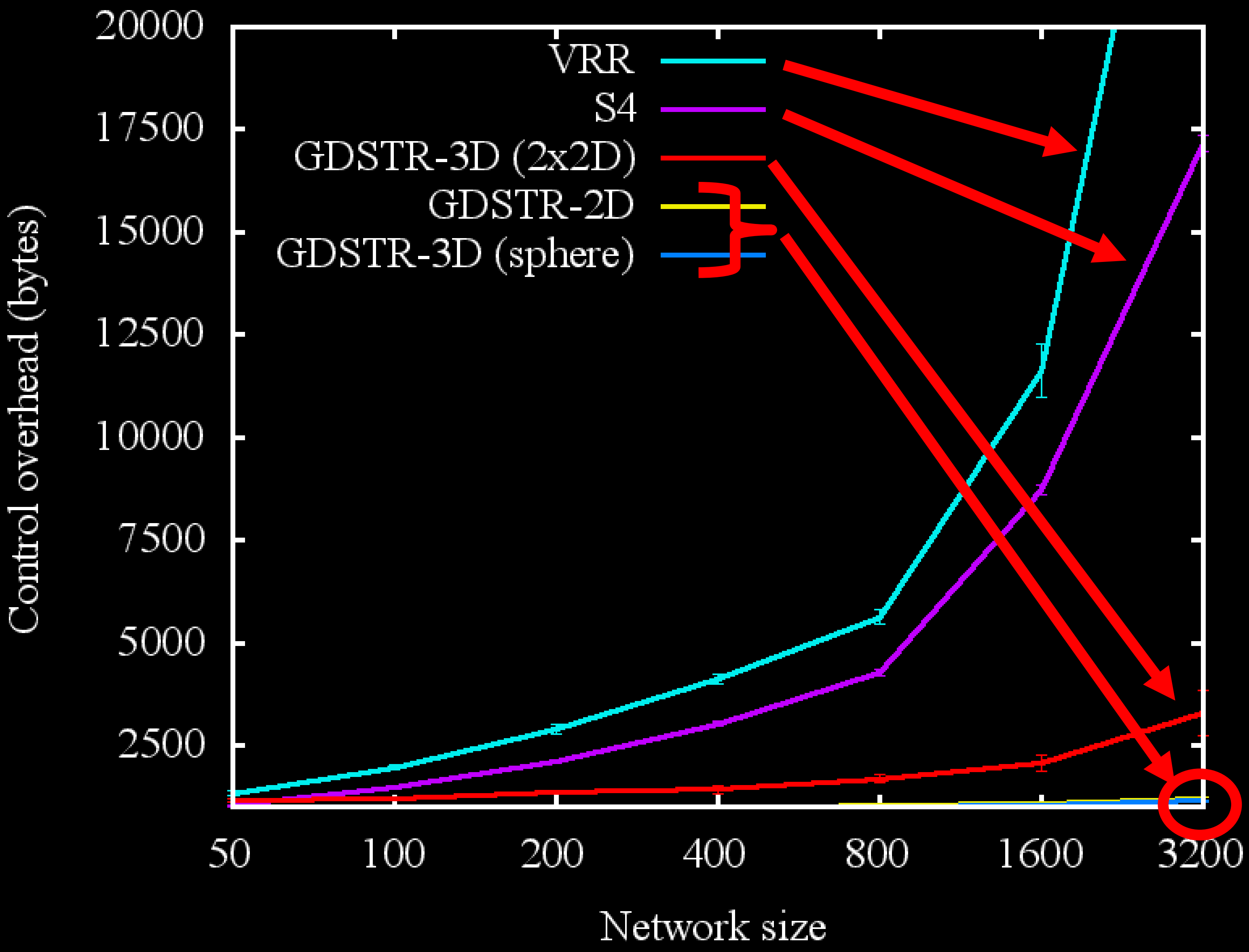
Hop stretch vs. network size



Maximum storage vs. network size



Message overhead (bytes) vs. network size



Summary: Scaling Up (3,200 nodes)

Algorithm	Stretch	Storage	Overhead
GDSTR-3D	✓	✓	-
GDSTR-2D	✗	✓	✓
S4	✓	✓	✗
VRR	✗	✗	✗
AODV	-	✓	?

Comprehensive comparison of GDSTR-3D to

1. AODV
2. VRR
3. S4

Details in the paper.

Key Contributions

1. Practical 3D geographic routing
   - 2x2D hulls for aggregation
   - Two-hop greedy
2. Comprehensive comparison of state-of-art point-to-point algorithms for TinyOS

Summary

For small sensor networks (<200 nodes): pick your favorite algorithm. ☺

For large sensor networks (~3,200 nodes), geographic routing algorithms are most scalable:

- relatively low overheads
- storage matters, but is not overriding consideration

Life’s complicated

Tradeoffs at a glance

Algorithm	Needs coordinates?	Needs location service	Reactive?
GDSTR-3D	✓	✓	✗
S4	✗	✓	✗
VRR	✗	✗	✗
AODV	✗	✗	✓

Future Work

- More Thorough Comparison
  - link losses
  - quantify cost of location service/ coordinate assignment
  - resilience
  - incremental costs
  - traffic pattern/load
- Sleep-wake duty cycle
- Reduce memory footprint

TinyOS Source Code

Available here:

https://sites.google.com/site/geographicrouting

Or email me: benleong@gmail.com

Questions?

Thank You

For large sensor networks , geographic routing algorithms are most scalable:

- guarantee packet delivery
- storage cost is proportional to network density but size
- motes have small RAM

Choice:

- Extend existing 2D geographic routing algorithms to implement a 3D routing algorithm

- GDSTR is a natural candidate for extension

Routing in 3D:

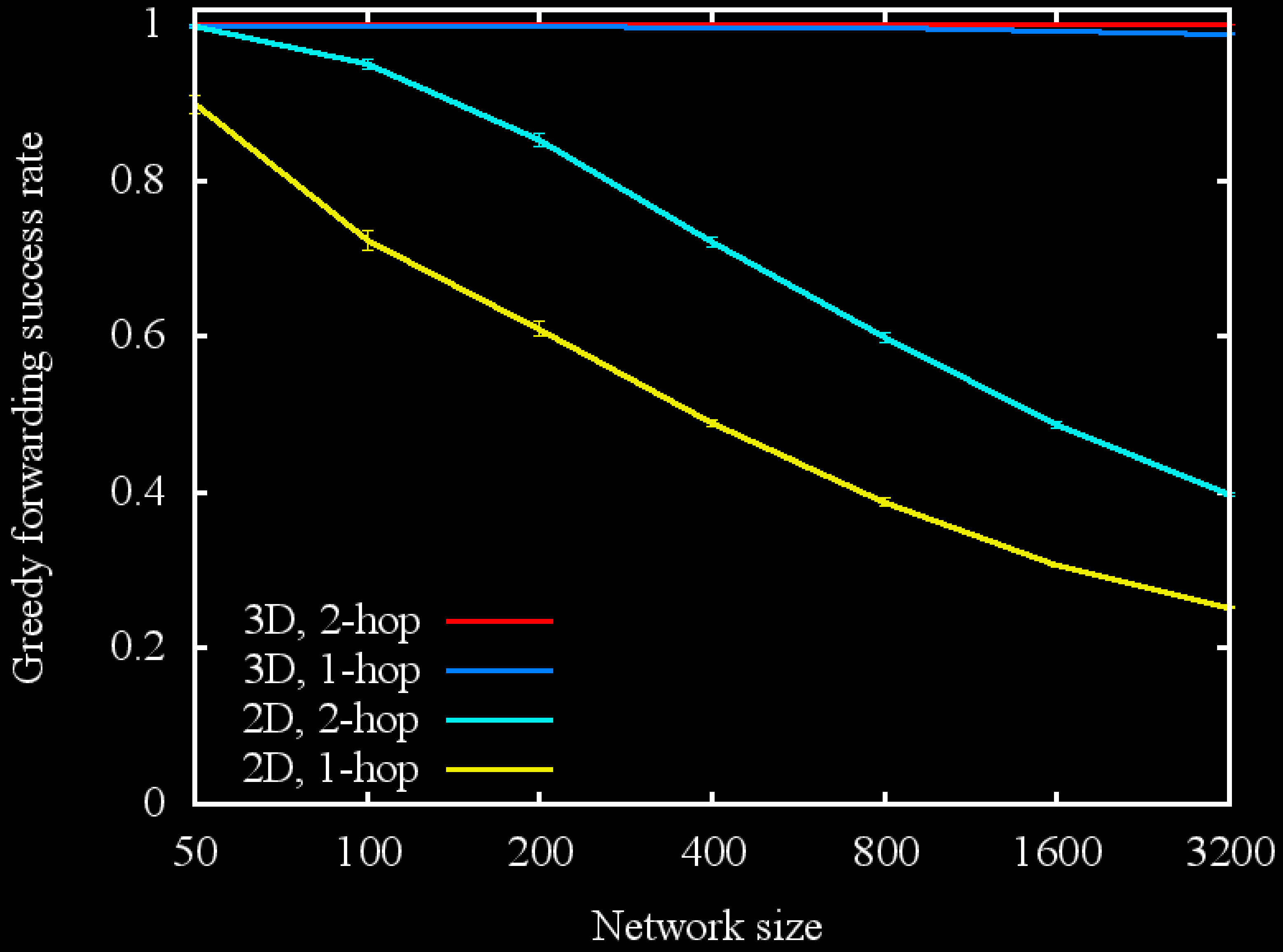
- Geographic routing in 3D topologies is intrinsically harder than routing in 2D topologies since greedy forwarding tends to encounter more local minima in general 3D topologies
- It is not entirely straightforward to extend GDSTR to 3D because that 3D convex hulls require significantly more storage and are much more computationally costly

Solution:

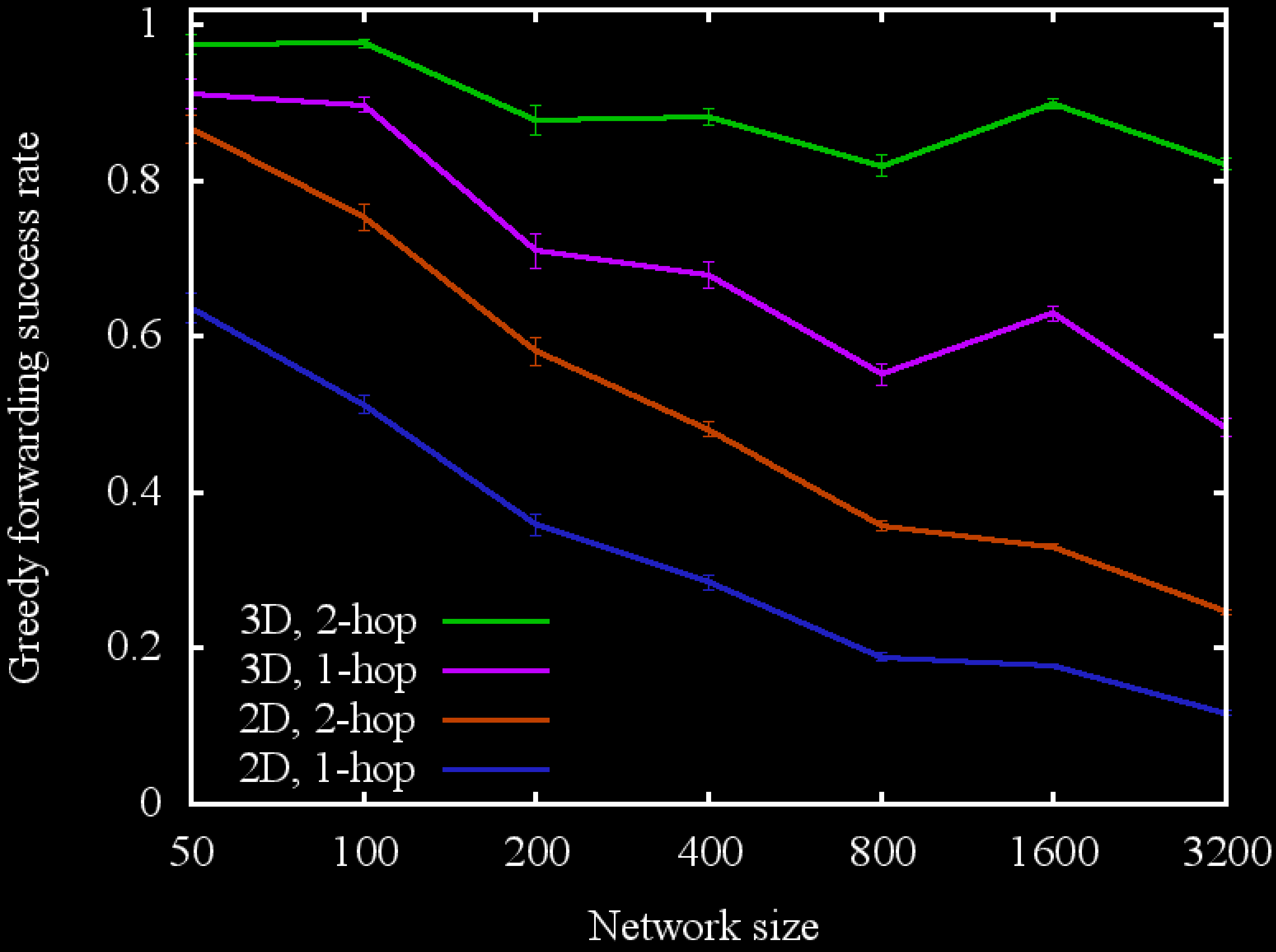
- Extend greedy forwarding by using 2-hop neighbor information to improve the greedy forwarding success rate in 3D networks
- Approximate 3D convex hulls with two 2D convex hulls

All graphs

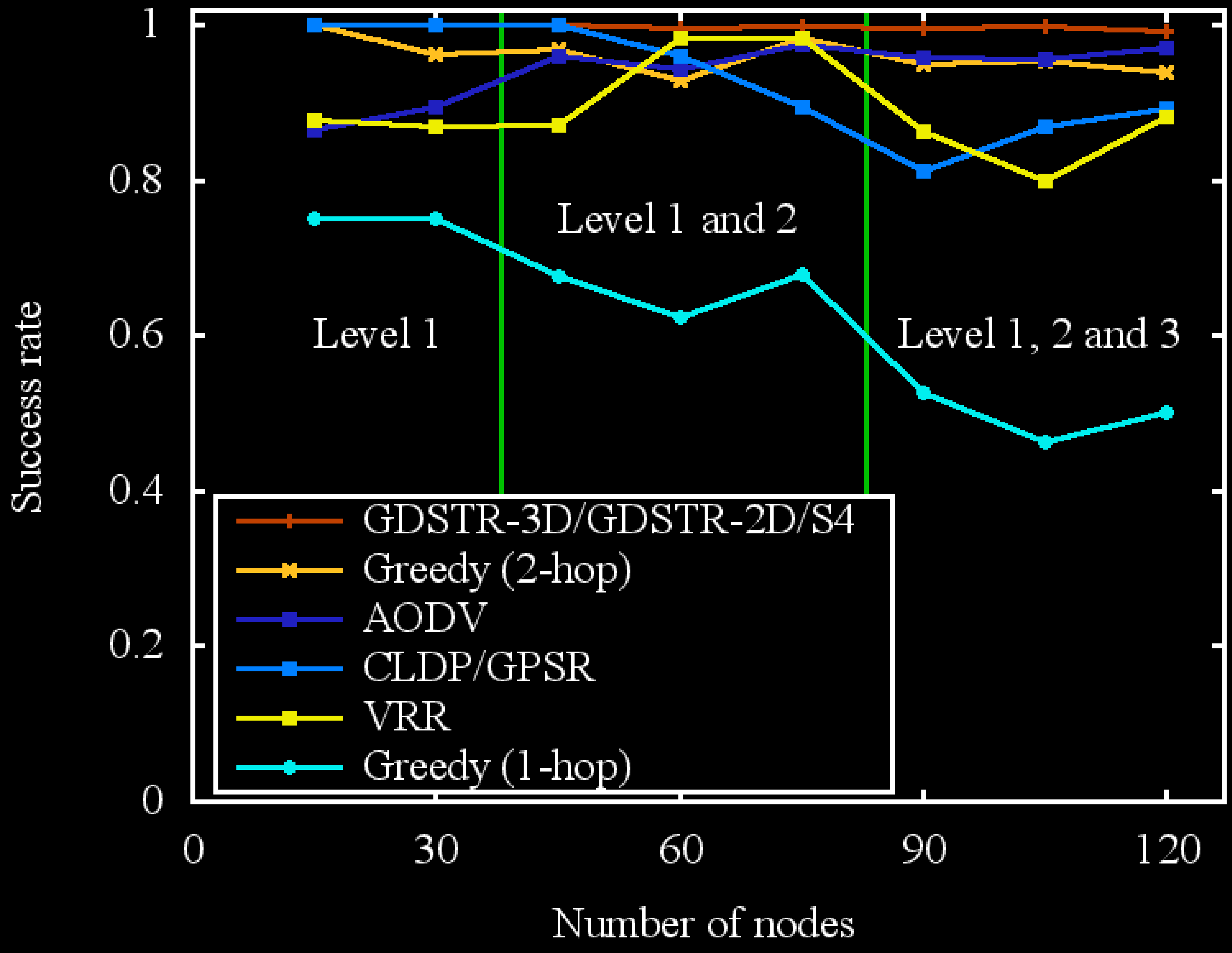
Greedy forwarding success rate vs. network size(high density)



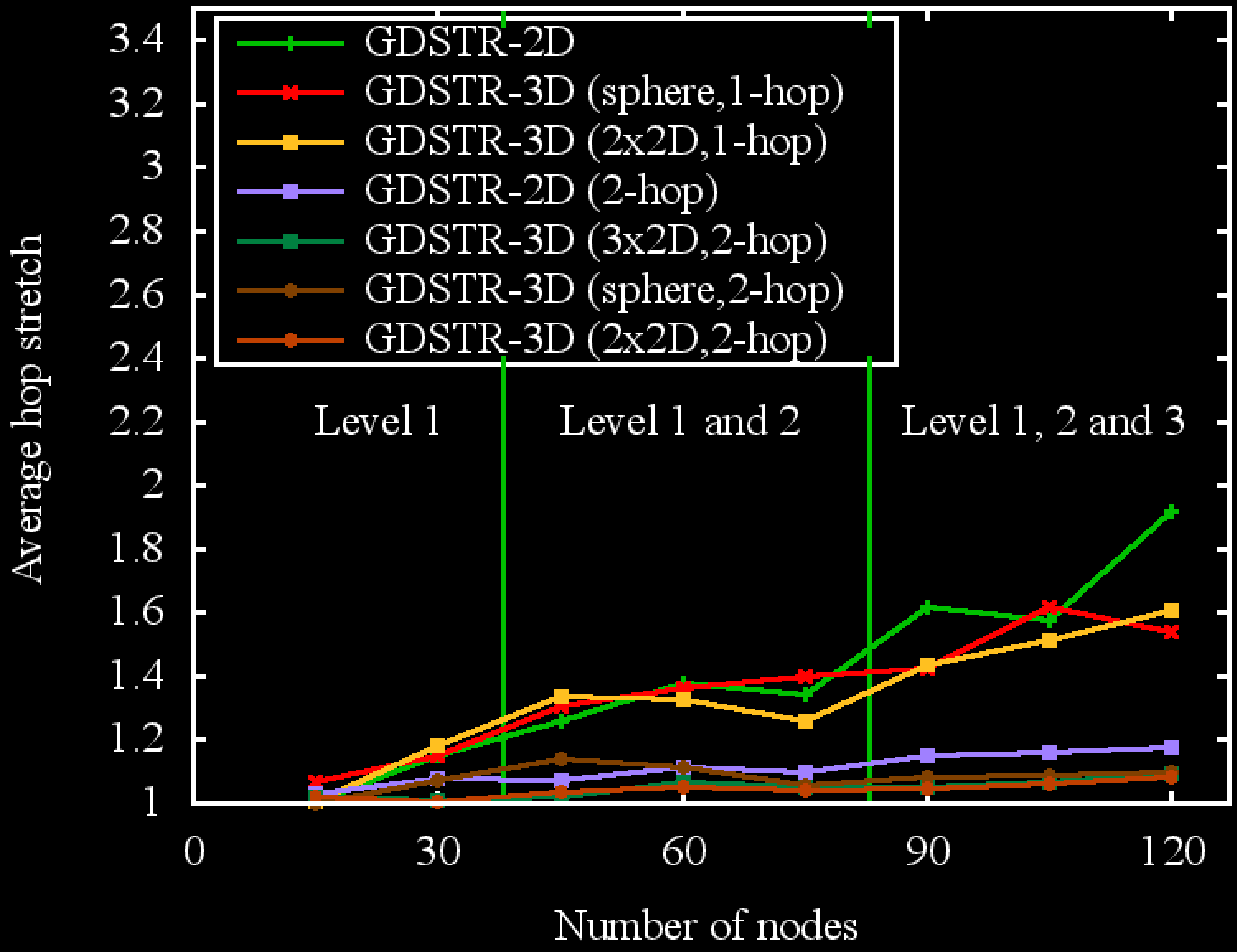
Greedy forwarding success rate vs. network size(low density)



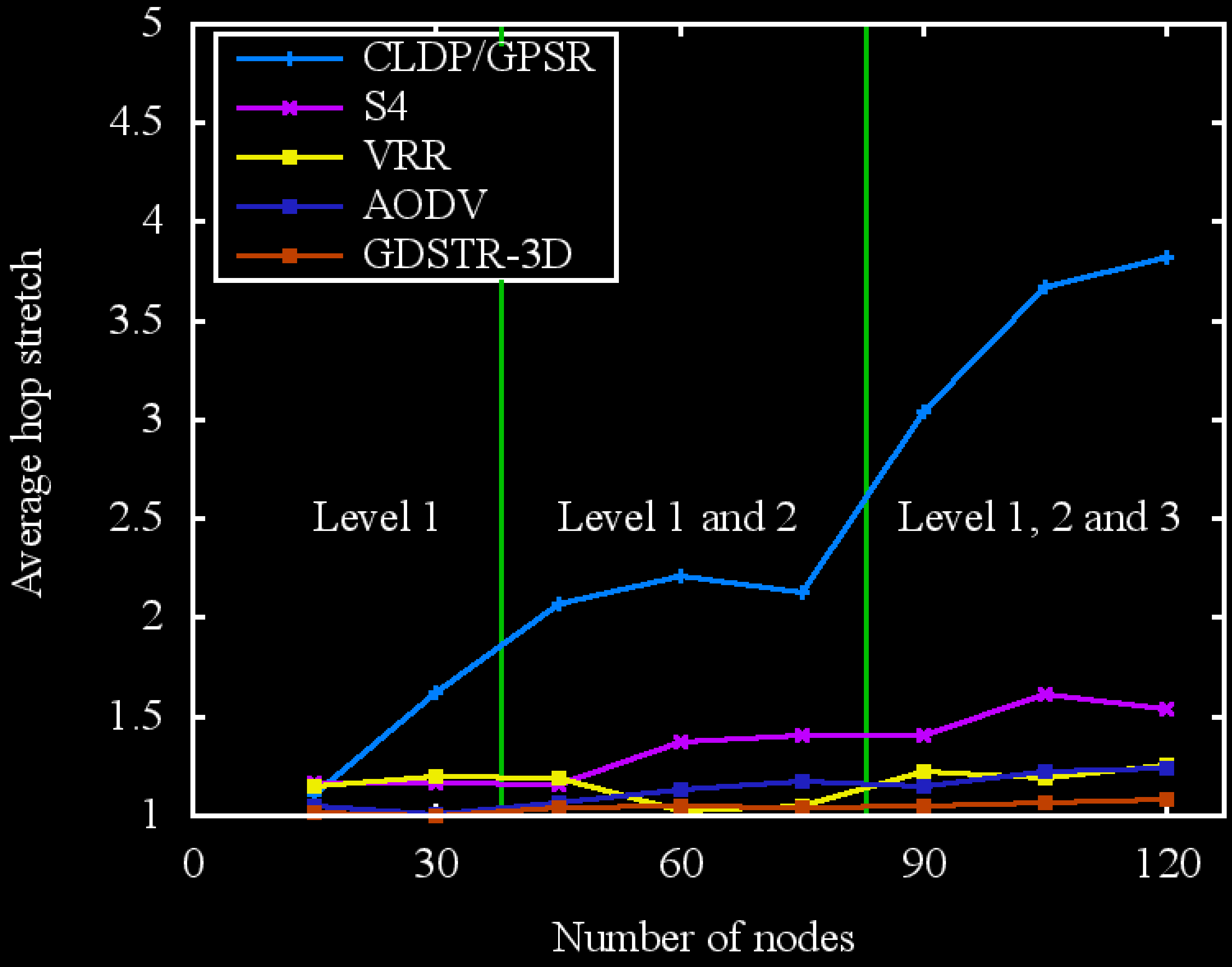
Success rate vs. network size



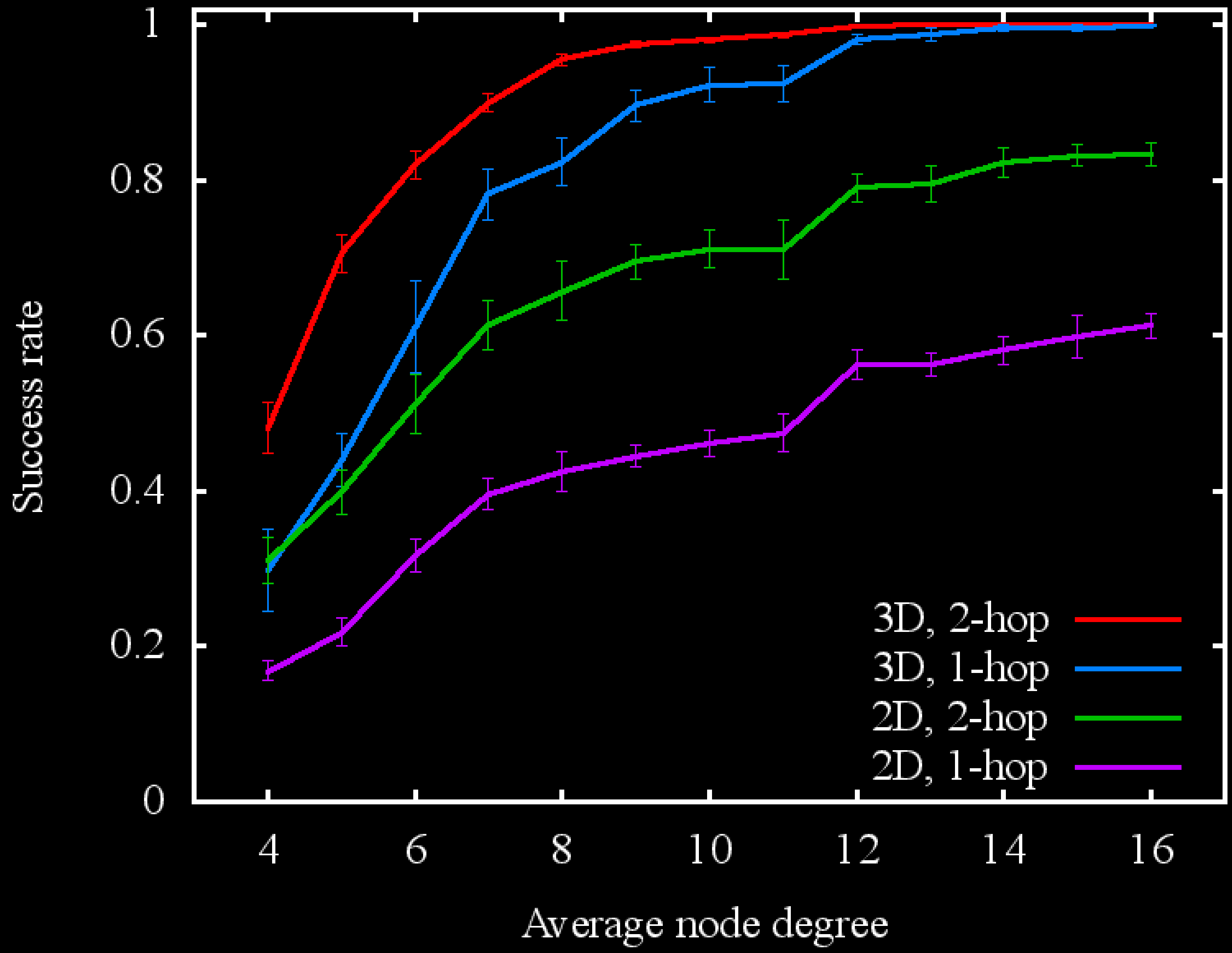
Hop stretch(GDSTR+) vs. network size



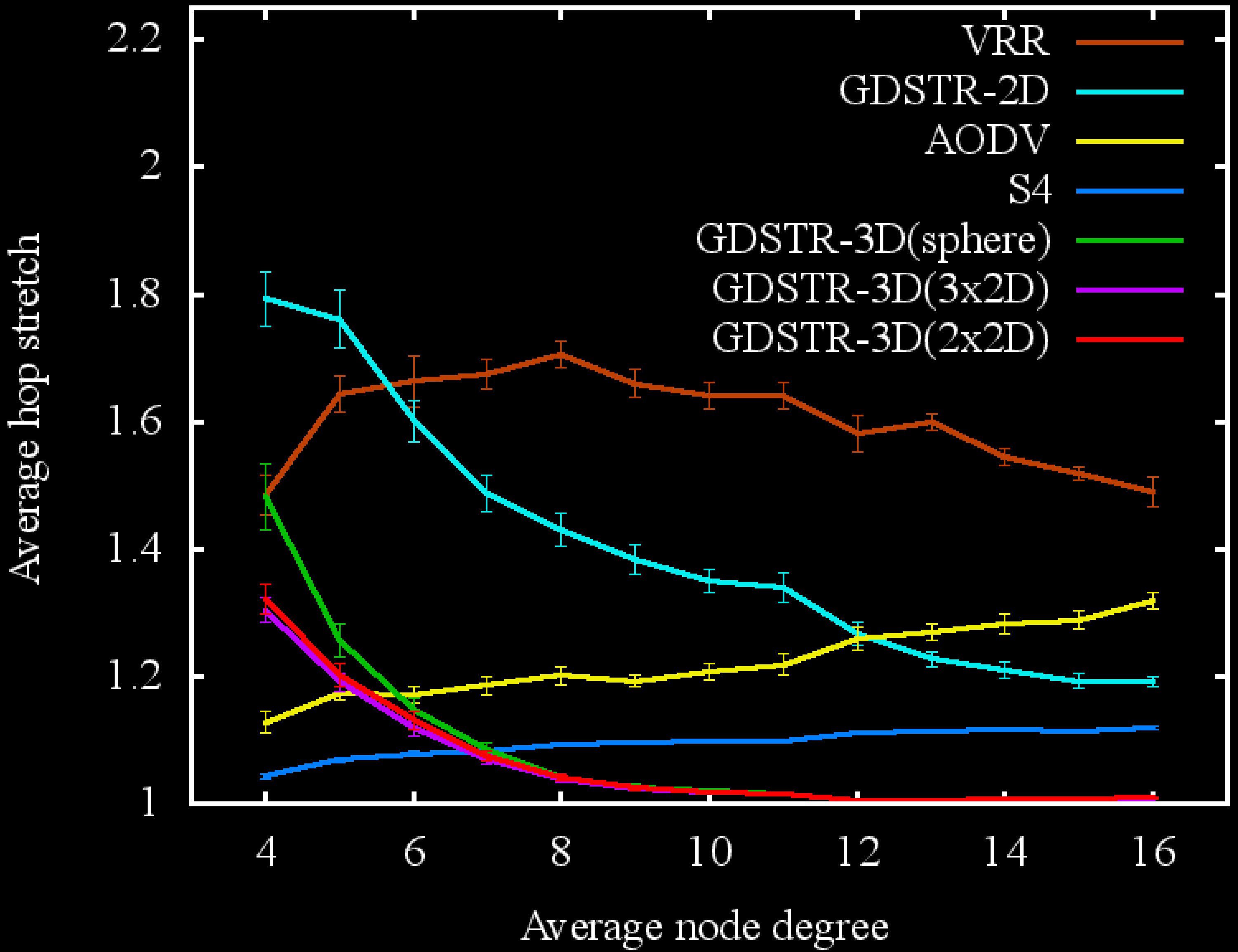
Hop stretch vs. network size



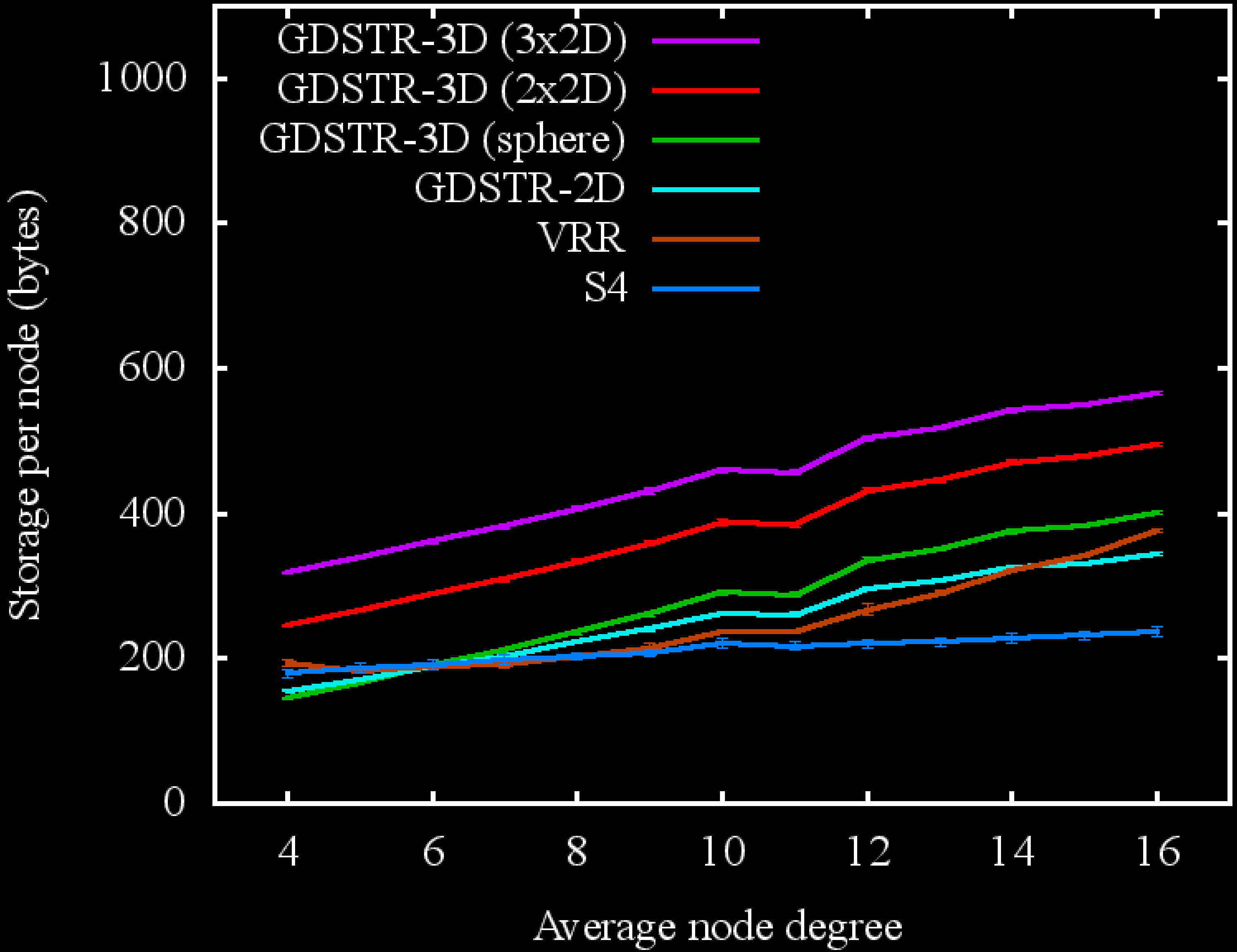
Greedy forwarding success rate vs. network density



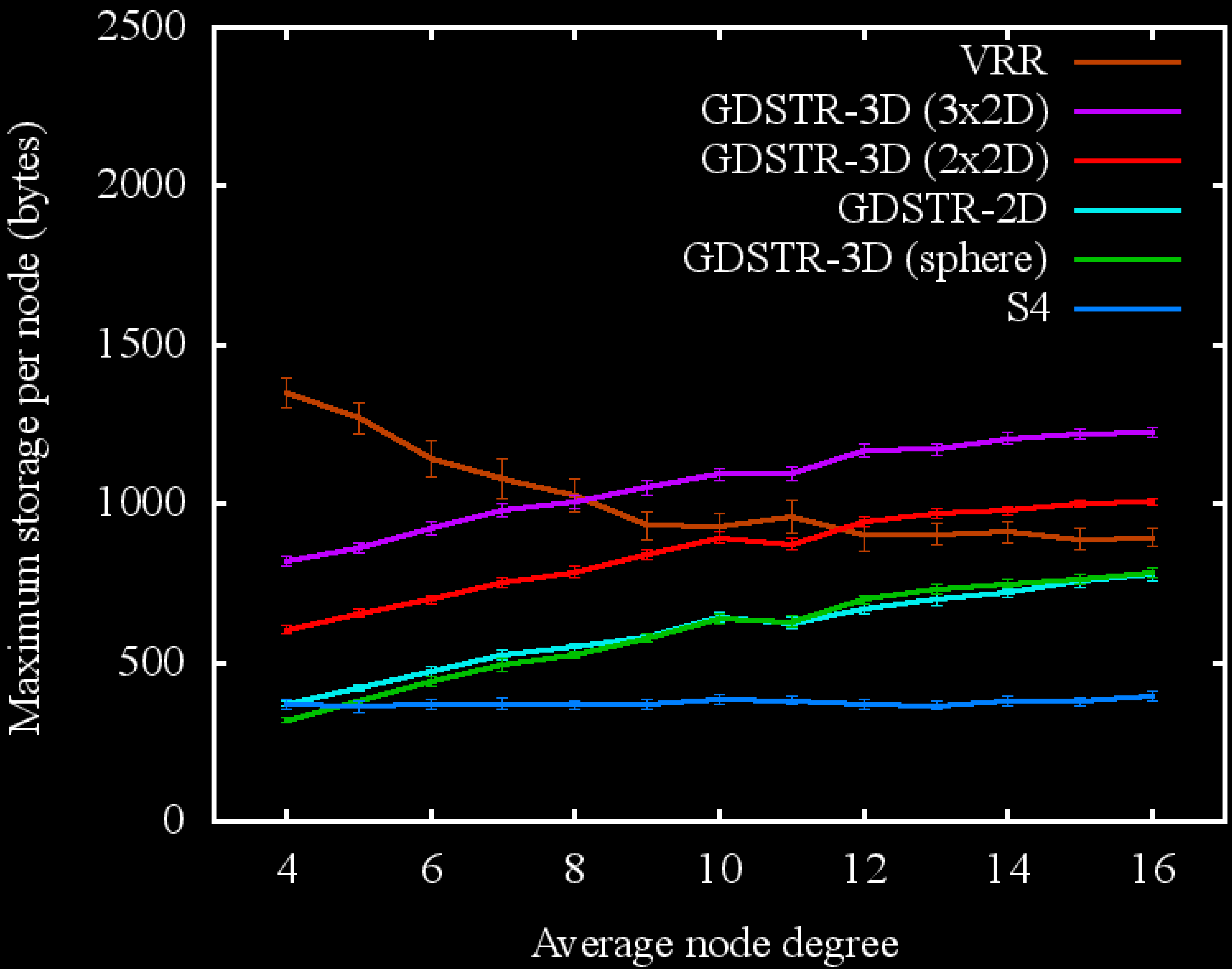
Hop stretch vs. network density



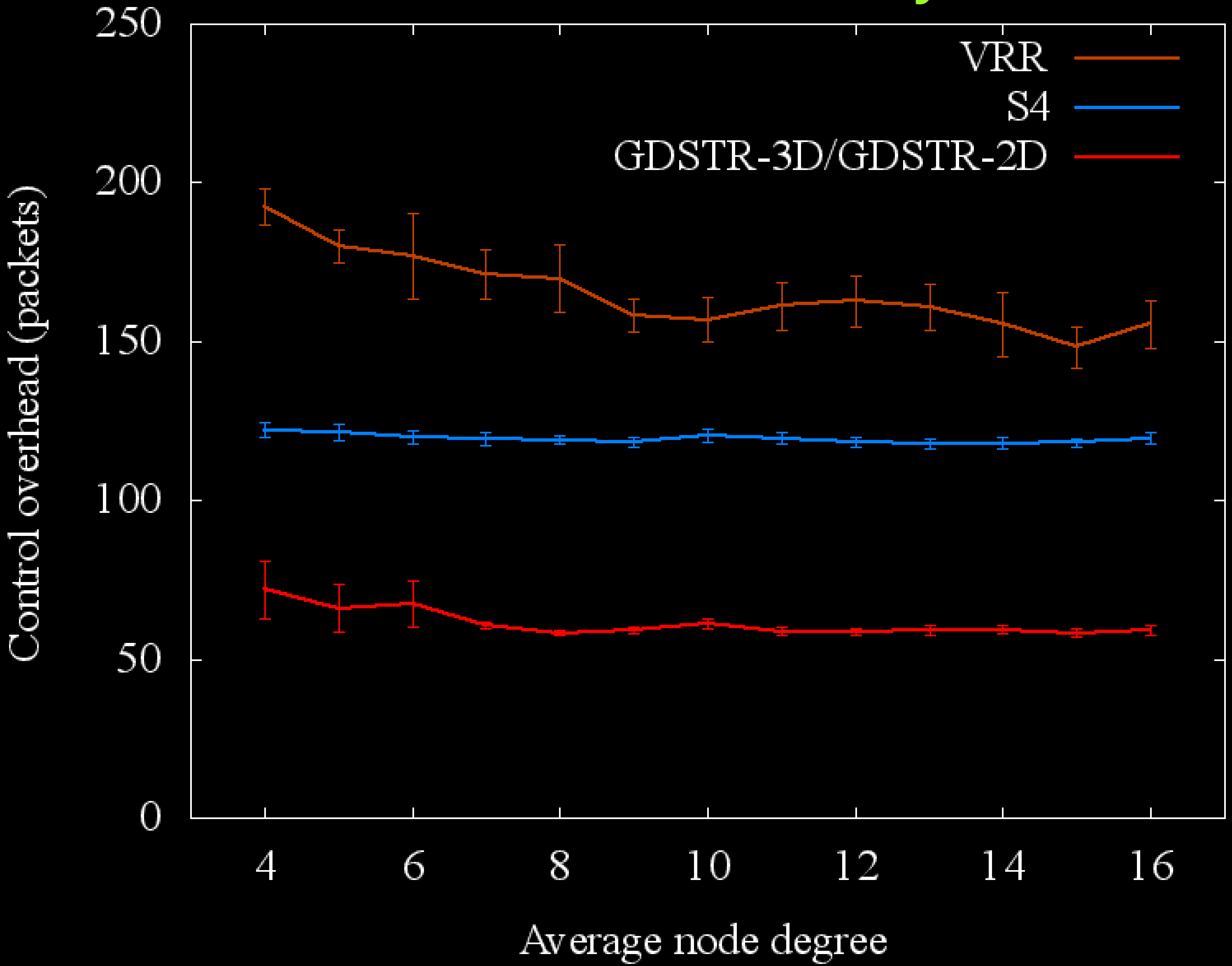
Average storage vs. network density



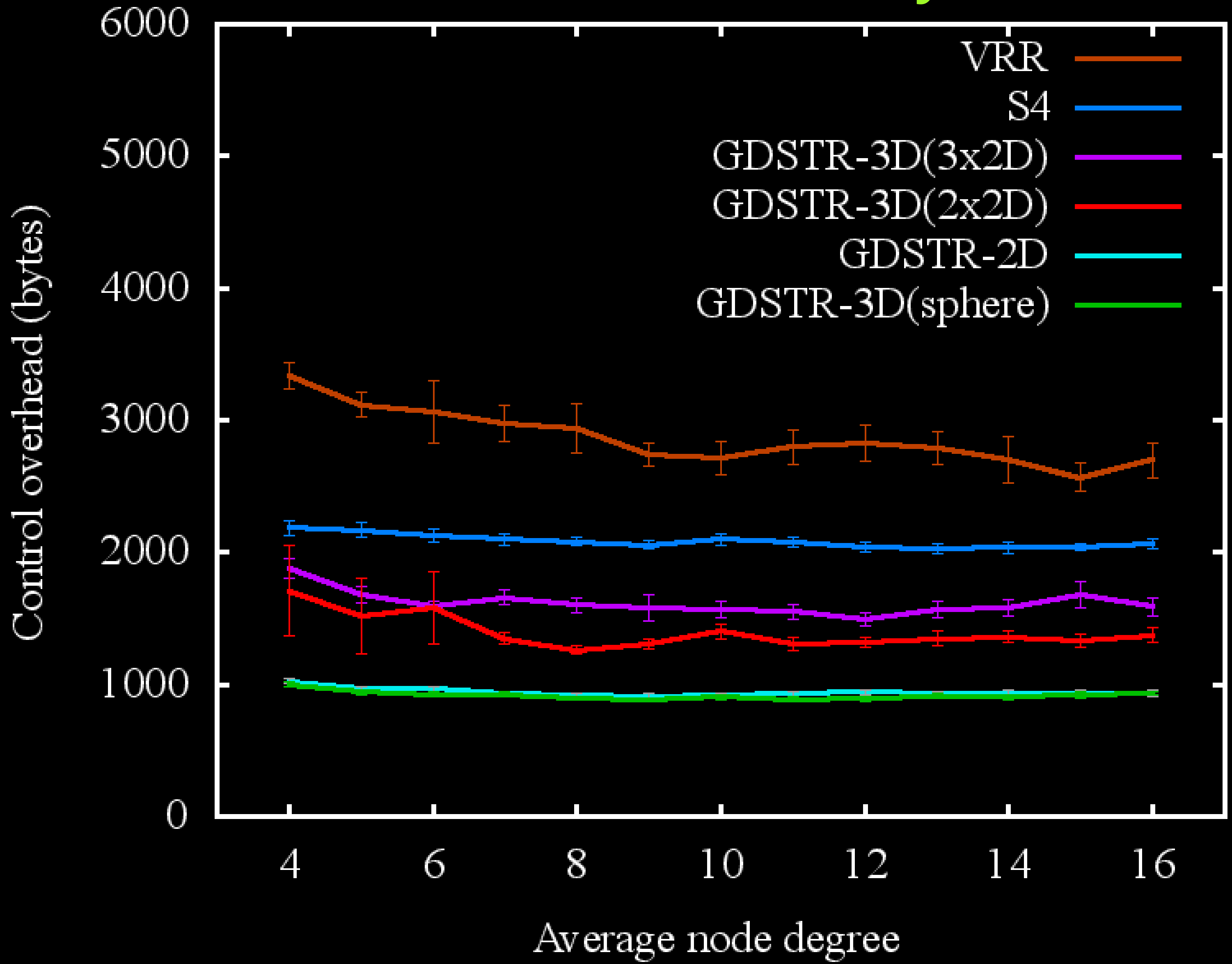
Maximum storage vs. network density



Message overhead(packets) vs. network density

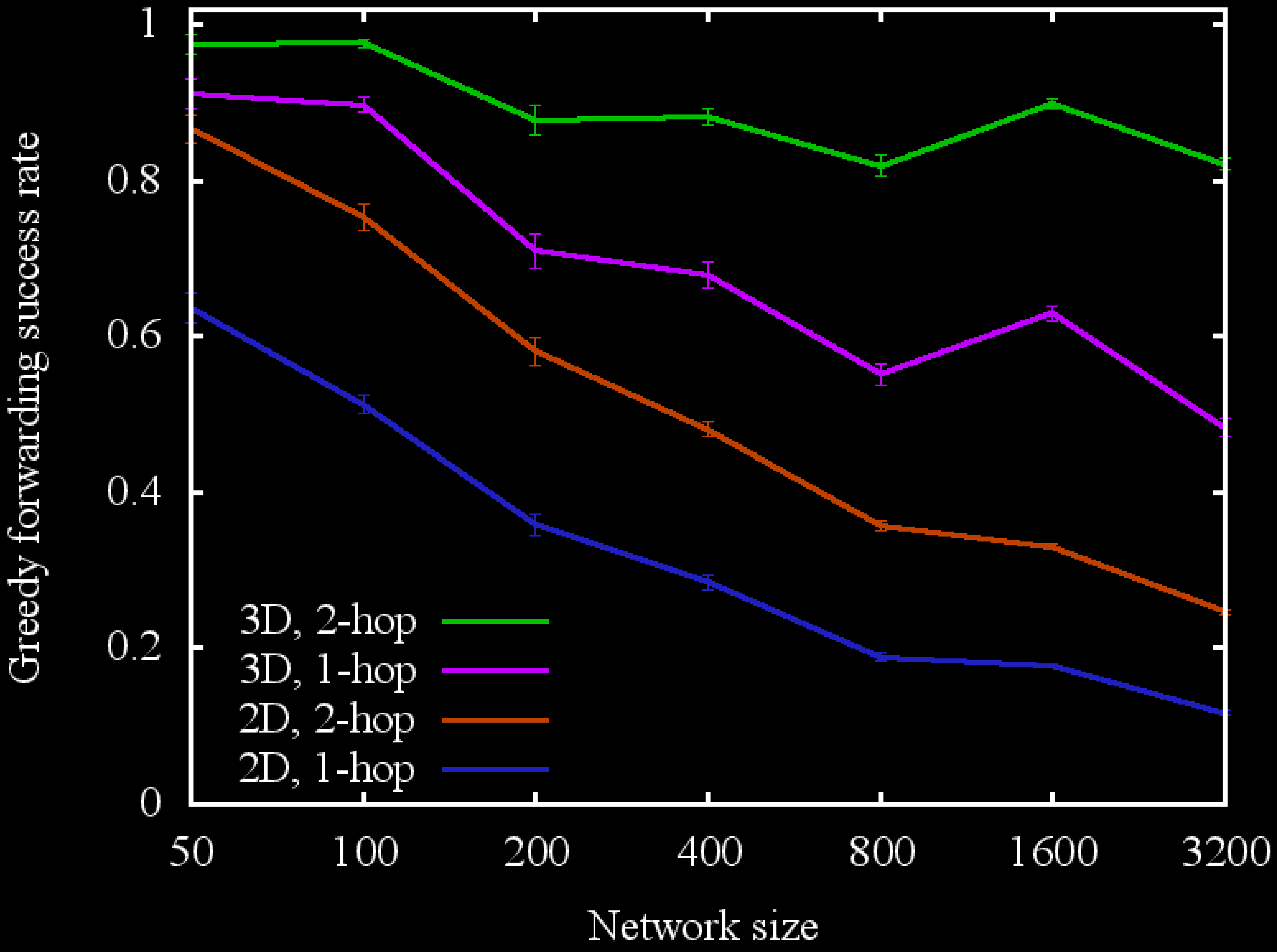


Message overhead(bytes) vs. network density

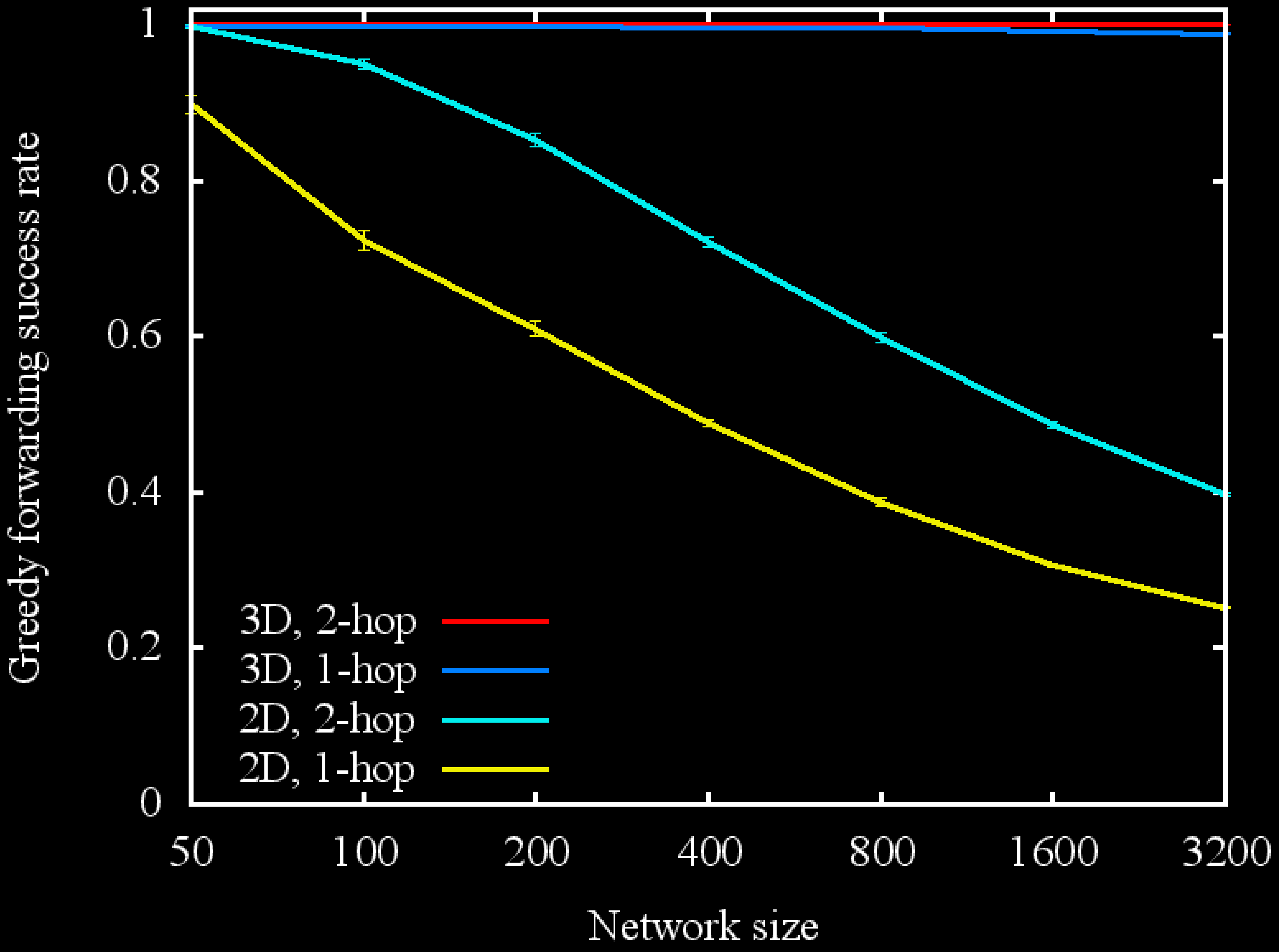


Scaling up

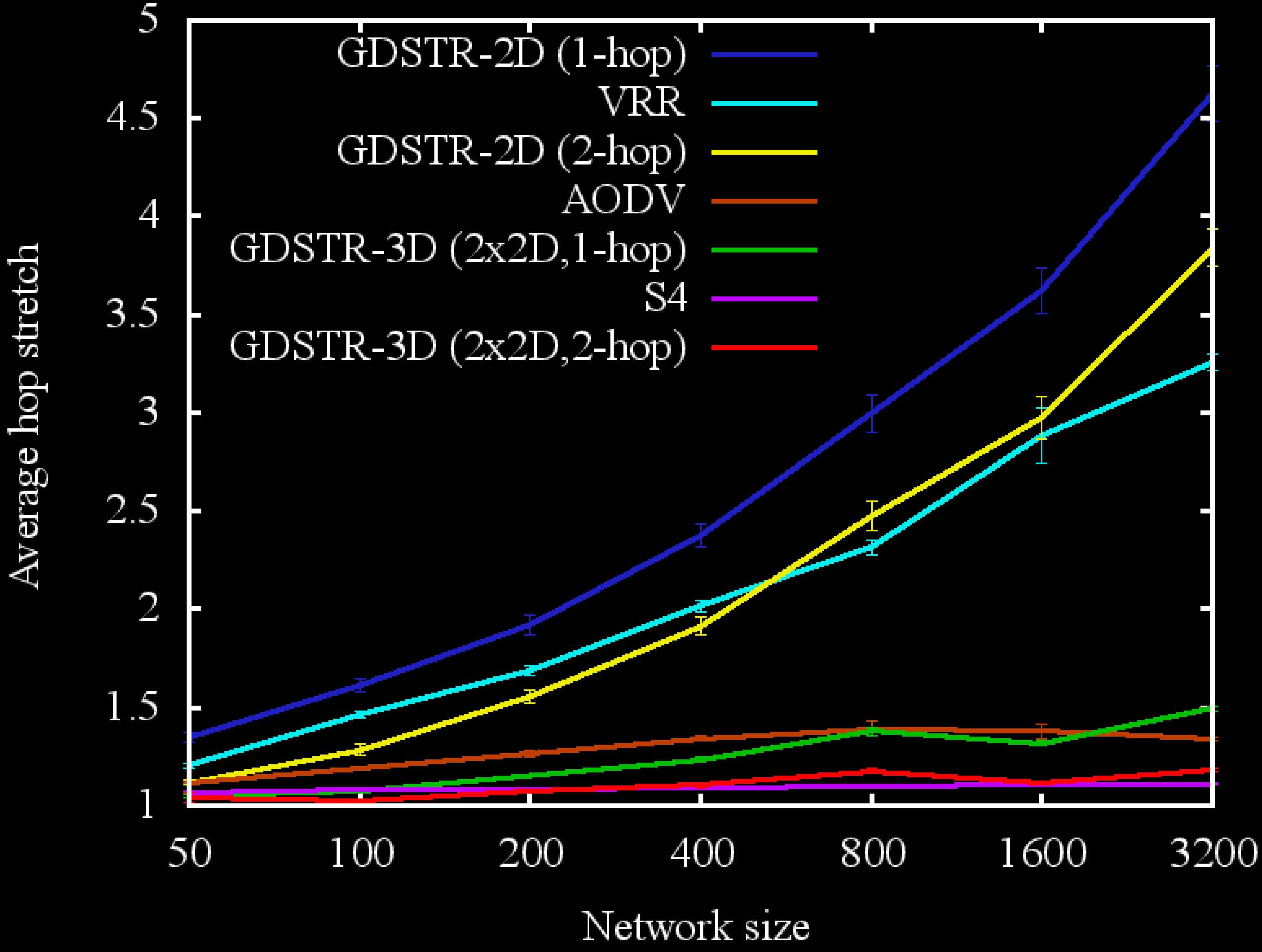
Greedy forwarding success rate vs. network size(low density)



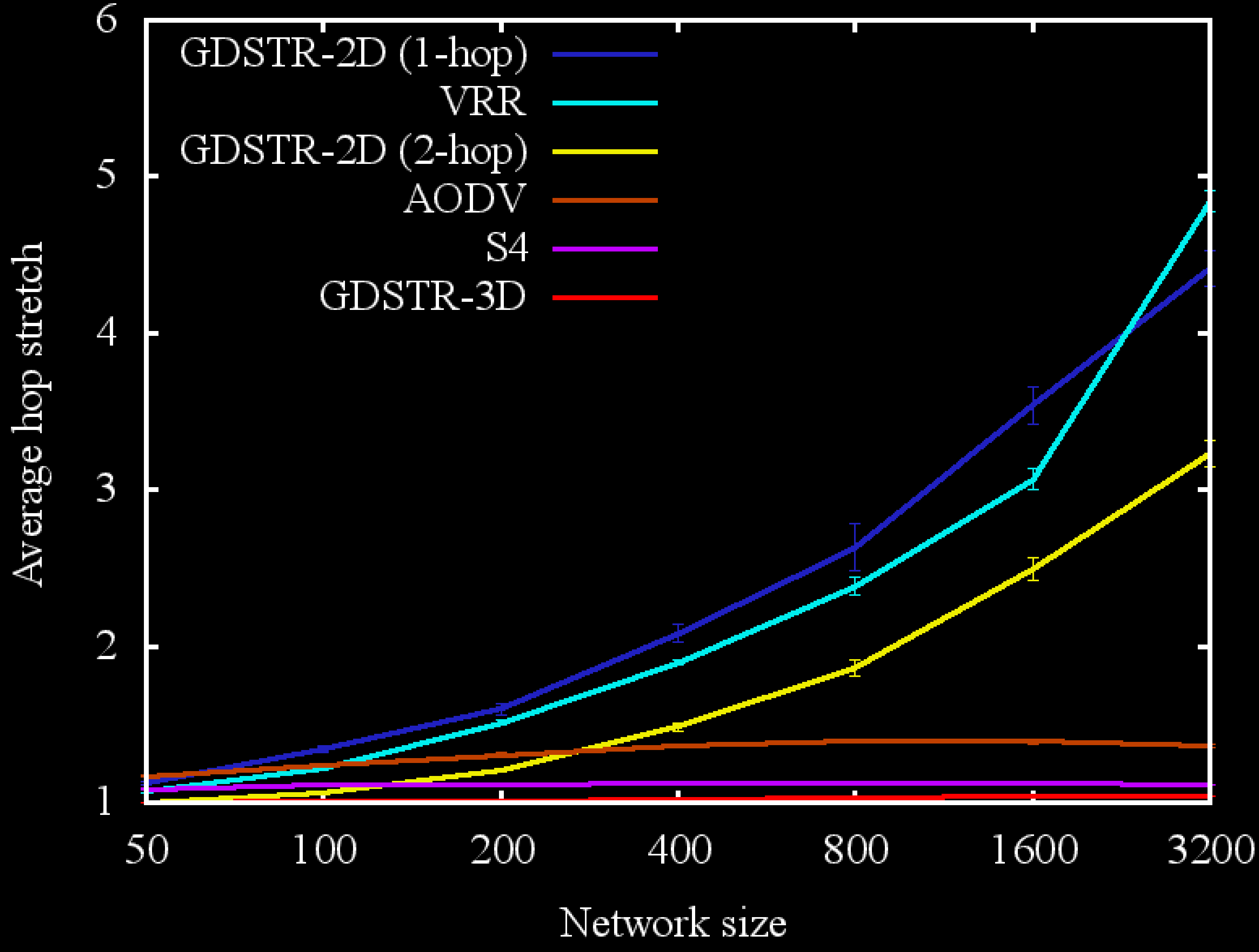
Greedy forwarding success rate vs. network size(high density)



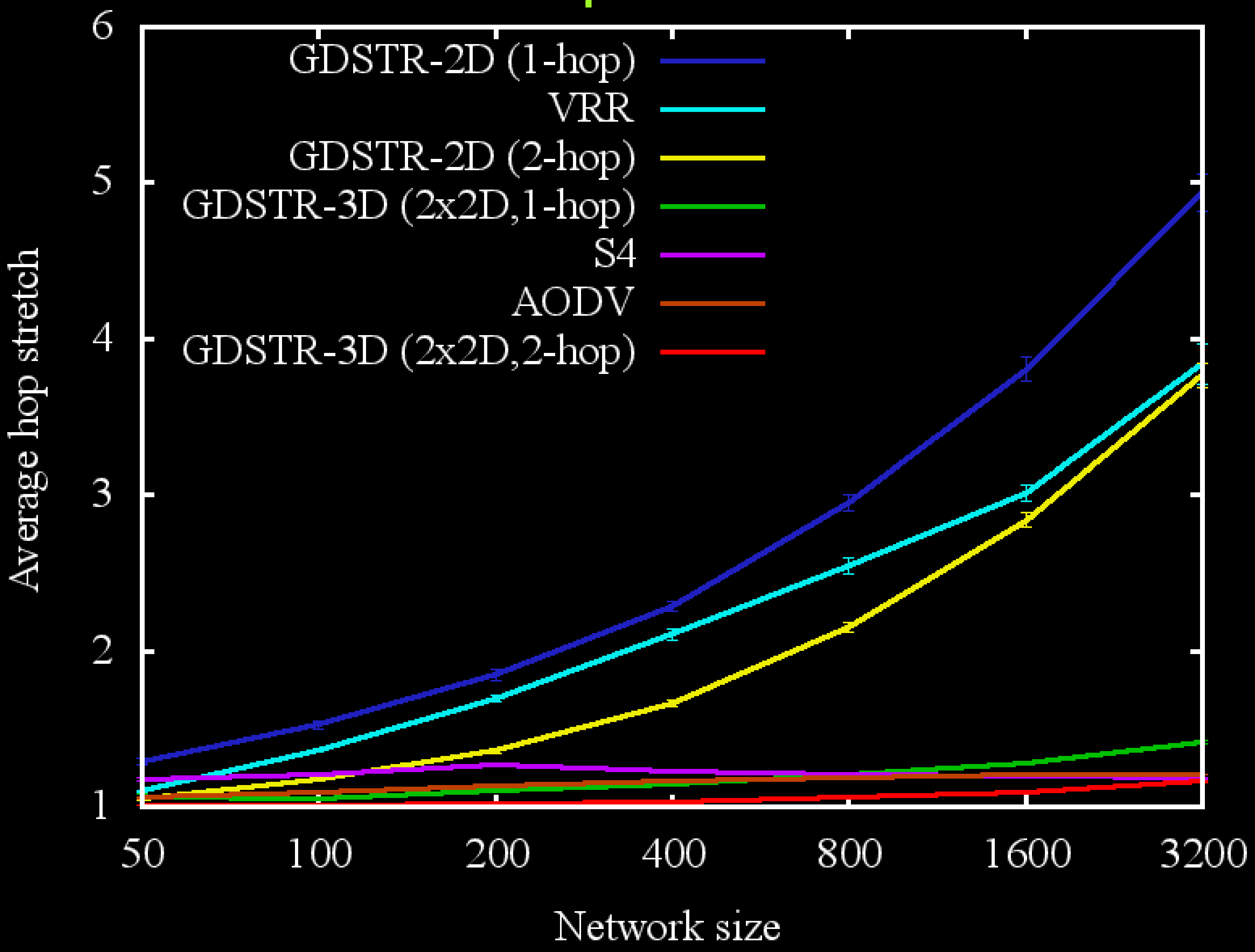
Hop stretch vs. network size(low density)



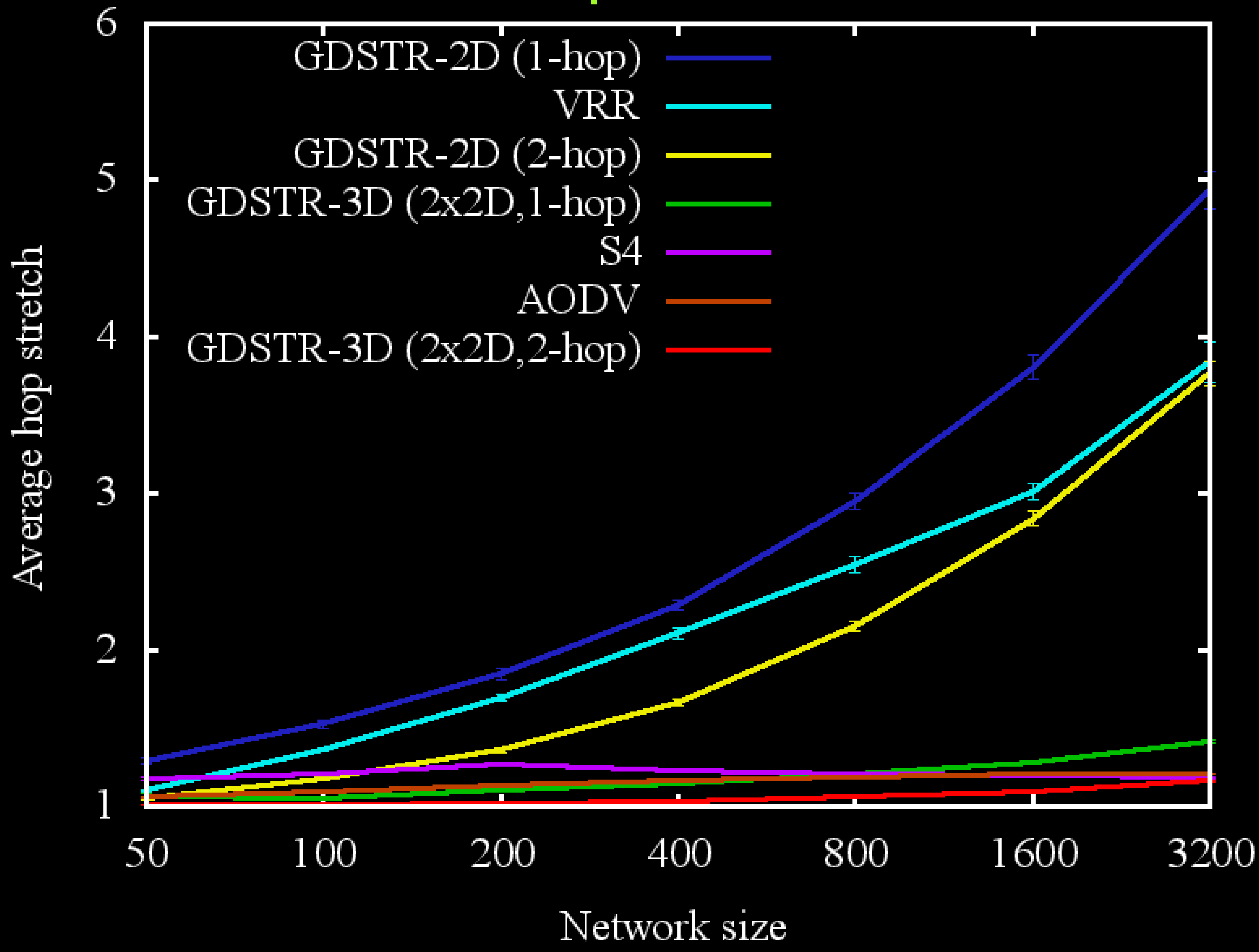
Hop stretch vs. network size(high density)



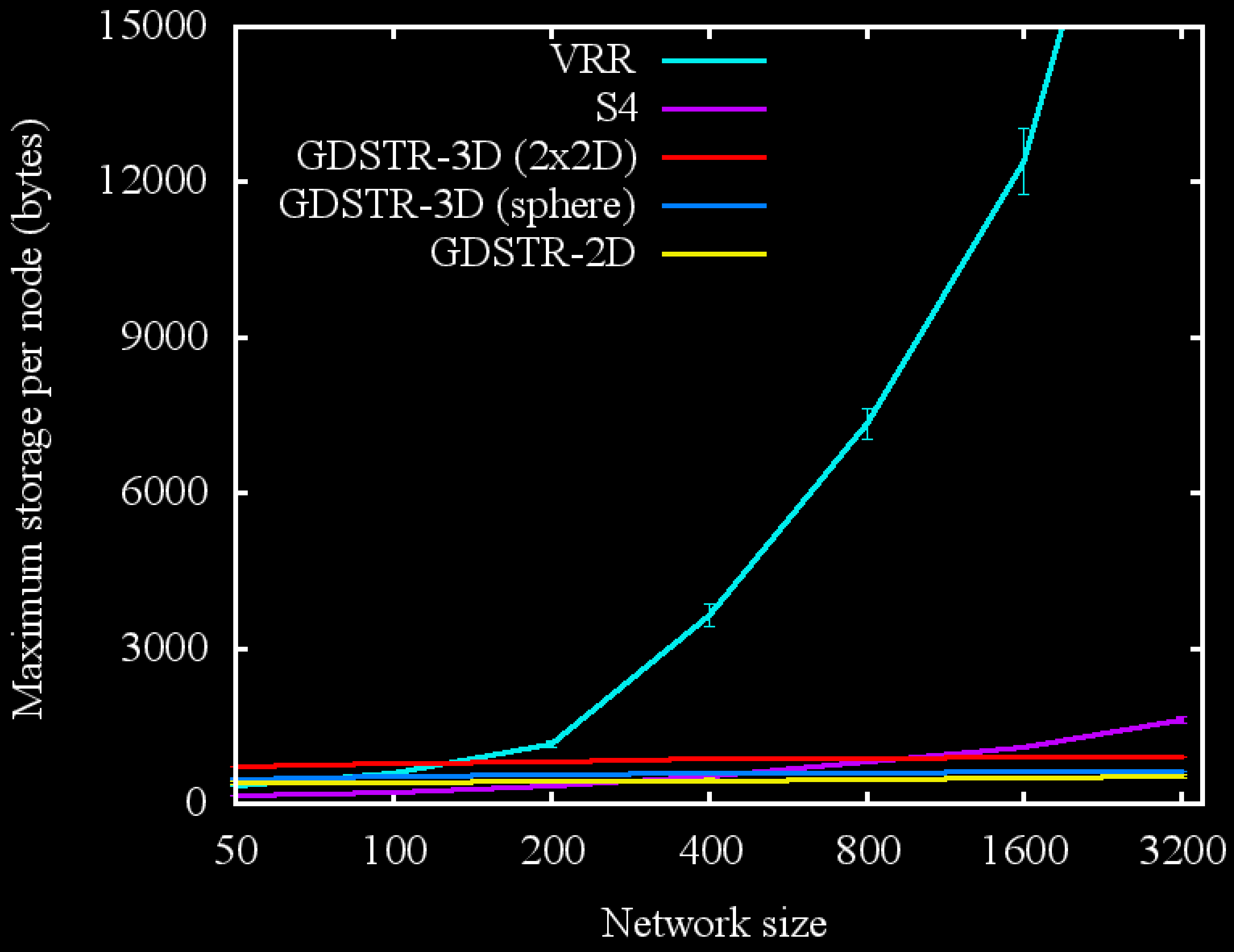
Hop stretch vs. network size with multiple obstacles



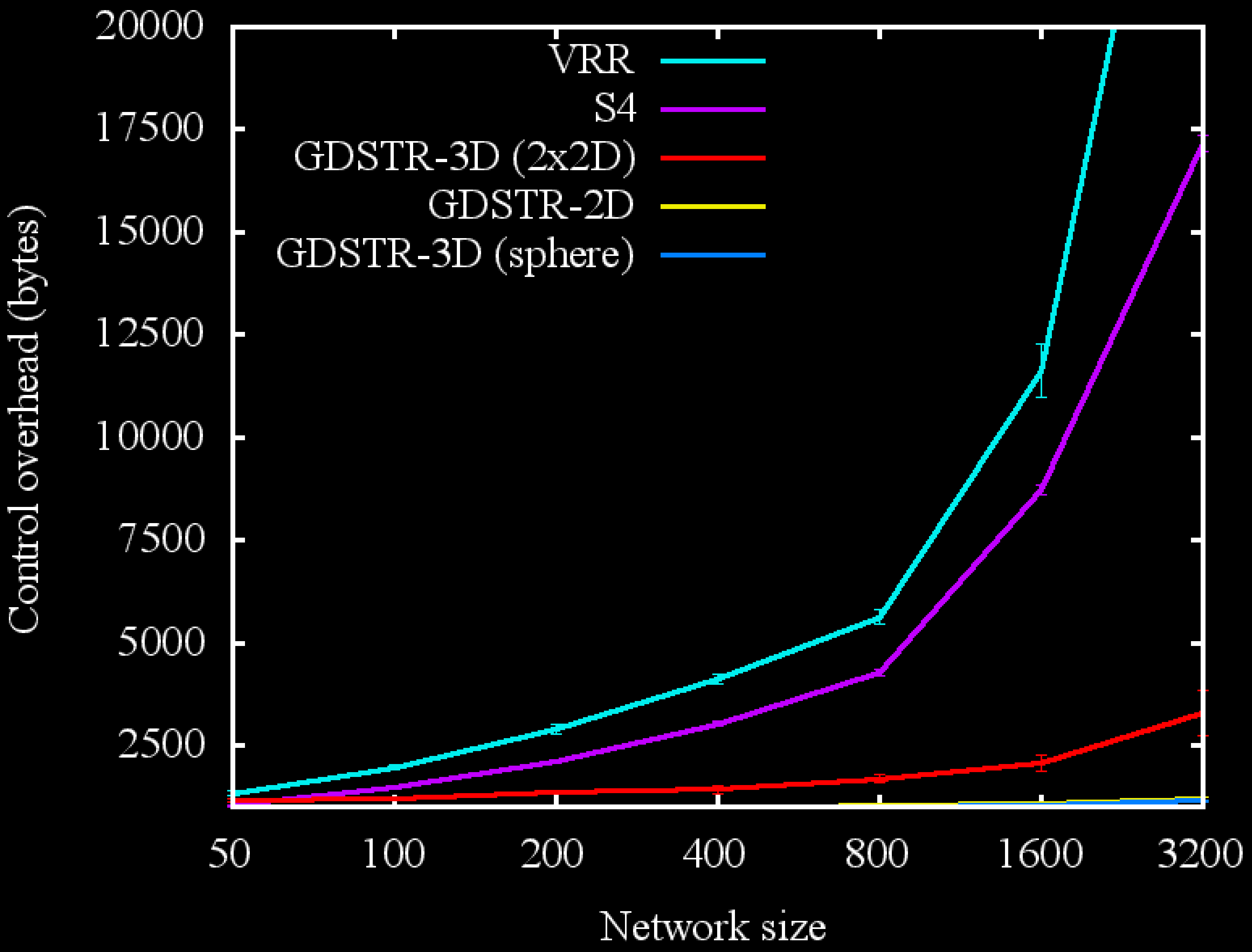
Hop stretch vs. network size with multiple obstacles



Maximum storage vs. network size



Message overhead(bytes) vs. network size


VRR
S4
GDSTR-3D (2x2D)
GDSTR-2D
GDSTR-3D (sphere)
Control overhead (bytes)
Network size