Geographic and Diversity Routing in Mesh Networks

COS 463: Wireless Networks
Lecture 7
Kyle Jamieson

[Parts adapted from B. Karp, S. Biswas, S. Katti]

Course Contents

1. **Wireless From the Transport Layer Downwards**
   – Transport over wireless, link layer, medium access, routing

2. **Overcoming Bit Errors**
   – Error Detection/correction, convolutional & “Rateless” codes

3. **An Introduction to the Wireless Channel**
   – Noise, Multipath Propagation, radio spectrum

4. **Practical/Advanced Wireless Physical Layer concepts**
   – OFDM, channel estimation, MIMO etc.

5. **Boutique topics**
   – Visible light communication, low power, Wi-Fi localization

Today

1. **Geographic (Location-Based) Mesh Routing**
   - **Greedy Perimeter Stateless Routing: GPSR**
   - Cross-link Detection Protocol: CLDP

2. Diversity Mesh Routing
   - ExOR (Roofnet)
   - Network Coding

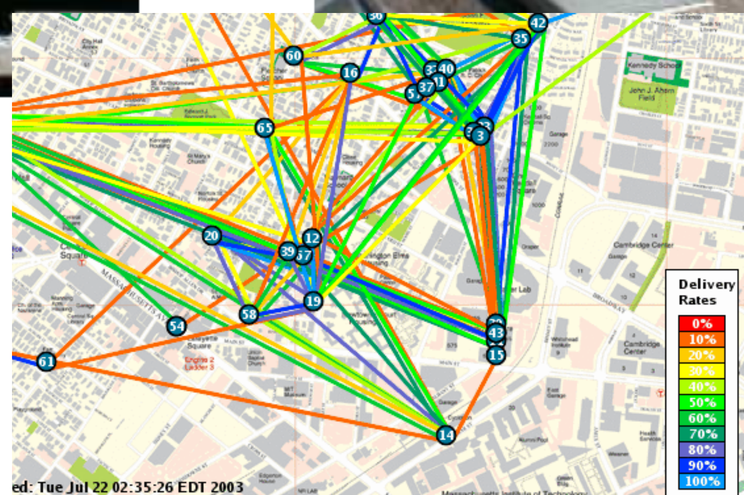
Context: Ad hoc Routing

- 1990s: availability of **off-the-shelf Wi-Fi cards, laptops**
- 1994: **First papers on DSDV and DSR routing** spark interest in routing on mobile wireless (ad hoc) networks
- 2000: **GPSR**
- 2000: Estrin *et al.*, and the Berkeley ***Smart Dust*** project sparks interest in **wireless sensor networks (*sensornets*)**

Deborah Estrin

Original Motivation (2000): Rooftop Networks

- **Share the broadband access network** among many geographically-close households, using a **wireless mesh**



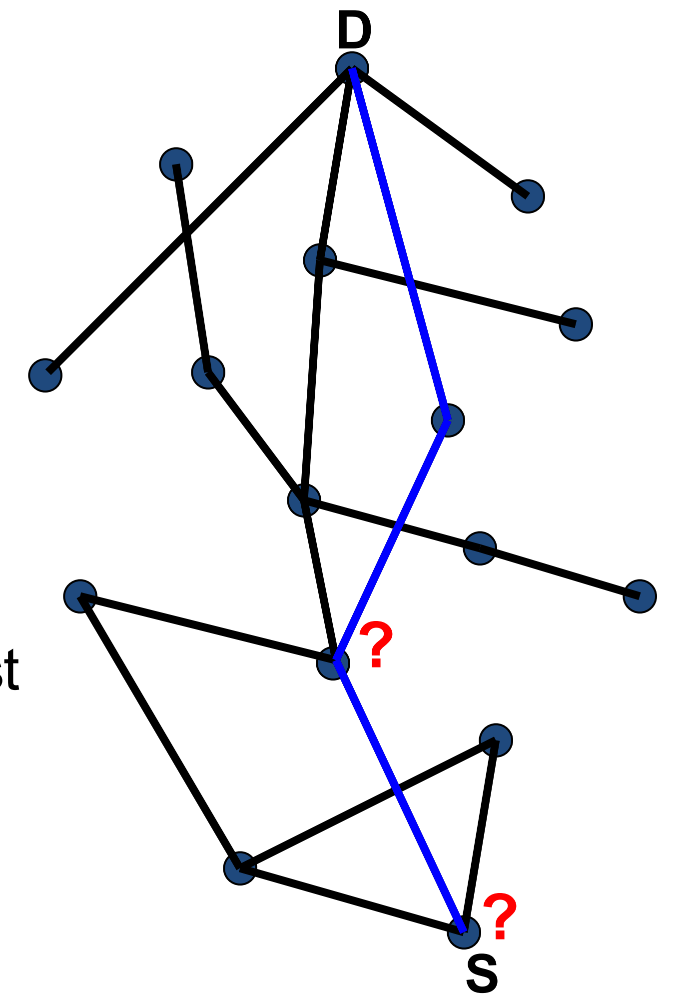
Motivation (2010+): Sensornets++

- Many sensors, widely dispersed
- **Sensor:** radio, transducer(s), small CPU, storage, battery
- Multiple wireless hops, **forwarding** sensor-to-sensor to a base station
- Sensornets redux *ca.* 2015+: ***Internet of Things***
  - Related concept, smaller numbers: ***Edge Computing***

What communication primitives will ***thousand-*** *or* ***million-node*** *sensornets need?*

The Routing Problem

- Each router has ***unique ID***
- Packets stamped with **destination ID**
  - Router must **choose next hop** for received packet
  - Routers communicate to **accumulate state** for use in forwarding decisions
- Evaluation metrics:
  - **Minimize:** Routing protocol message cost
  - **Maximize:** E2E throughput
  - **Minimize:** Per-router state

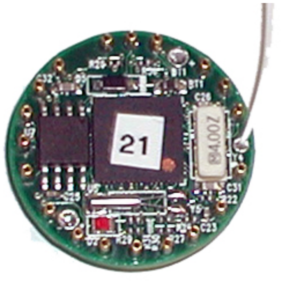


Scalability in Sensor Networks

- **Resource constraints** drive **(slightly modified) goals:**

- **State per node: minimize**
- **Energy consumed: minimize**
- **Bandwidth consumed: minimize**

- **System scale** in nodes: **maximize**
- Message **delivery success rate: maximize**



Scaling Routing

- **Link State:** Push full topology map to all routers, **O(# links) state**
- **Distance Vector:** Push distances across network, **O(# nodes) state**
- **Dynamic Source Routing (DSR):**
  - Flood queries on-demand to learn source routes
  - Cache replies, **O(# nodes) state**
- **Internet routing** scales because of **hierarchy & IP prefix aggregation; but not easily applicable** in sensornets

Can we achieve per-node routing state independent of # nodes?

Greedy Perimeter Stateless Routing (GPSR)

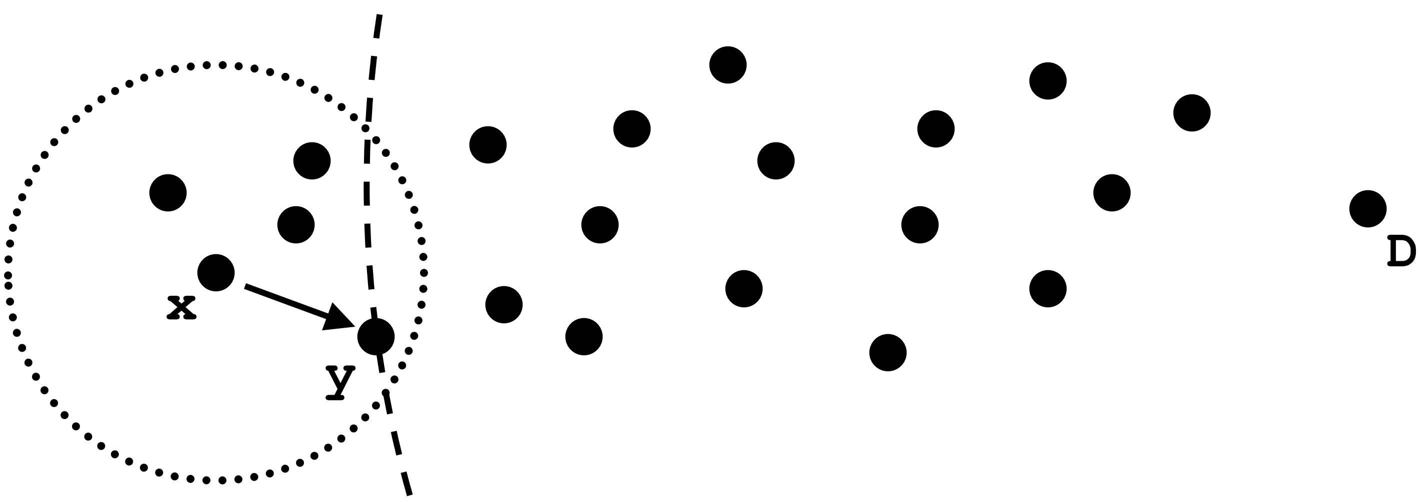
- **Central idea:** Machines know their geographic locations, route using **location**
- Packet destination field = **location of destination**
- Assume some node **location registration/lookup system** to support **host-centric addressing**
- Node's **state** concerns **only one-hop neighbors:**
  - **Low per-node state: O(density)**
  - **Low routing protocol overhead: state pushed only one hop**

Assumptions

- Nodes **all** know **their own** locations
- **Bi-directional radio links** (unidirectional links may be excluded)
- Network nodes placed roughly **in a plane**
- Fixed, *uniform* radio **transmitter power**
- ***Unit Graph* Connectivity*:*** Node **connected to all others** in a fixed ***radio range,*** and **none outside this range**

Greedy Forwarding

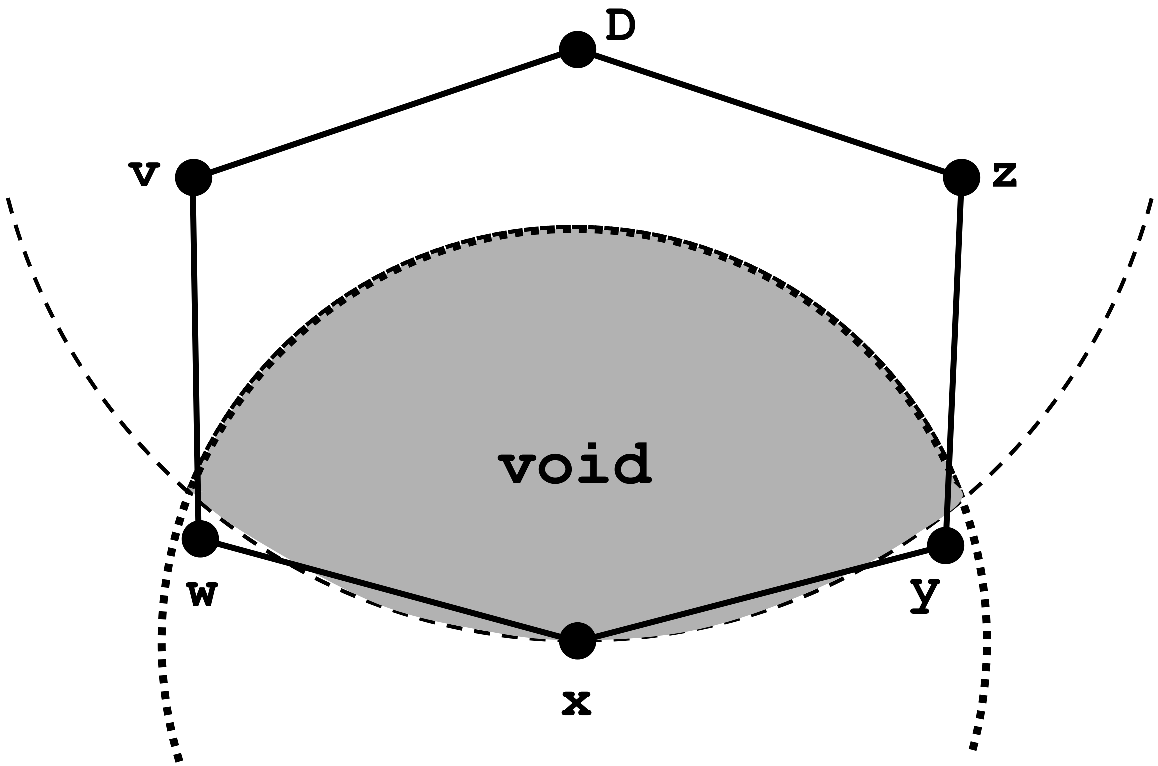
- Nodes **learn immediate neighbors' positions** from beaconing/piggybacking on data packets
- Locally optimal, **greedy next hop choice:**
  - Neighbor **geographically nearest to** destination



Neighbor must be strictly closer to avoid loops

Greedy Forwarding Failure

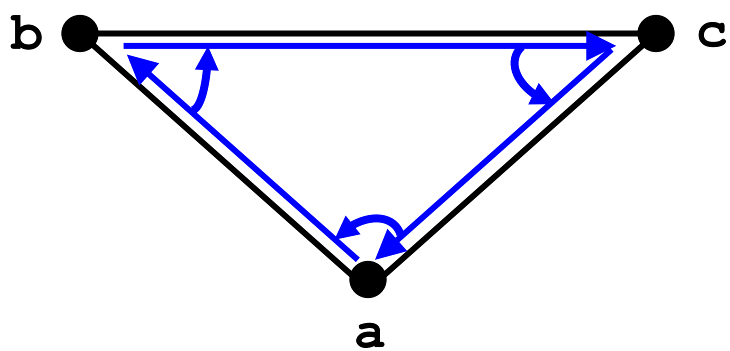
- **Greedy forwarding not always possible!** Consider:



How can we circumnavigate voids, relying only on one-hop neighborhood information?

Traversing a face

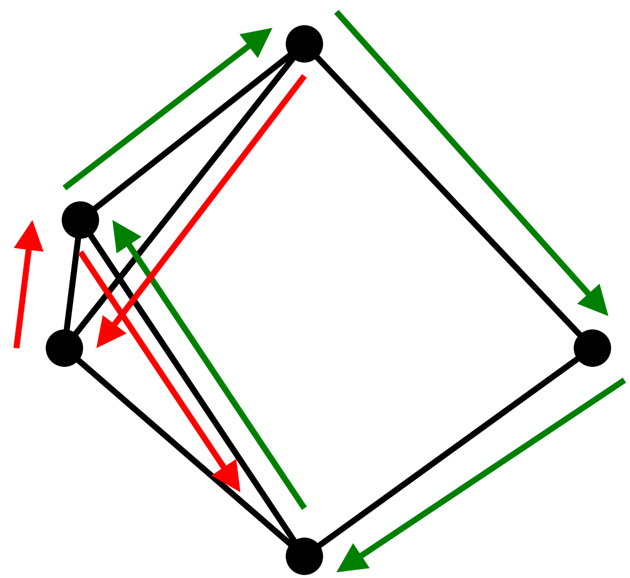
- Arriving at node **x** from node **y,** along edge **(x, y):**
  - ***Right-hand rule:*** depart **x** from the edge next in the **counterclockwise order** about **x,** after edge **(x, y)**



- Traverses the **interior** of a closed polygon in **clockwise edge order**

Planar vs. Non-planar Graphs

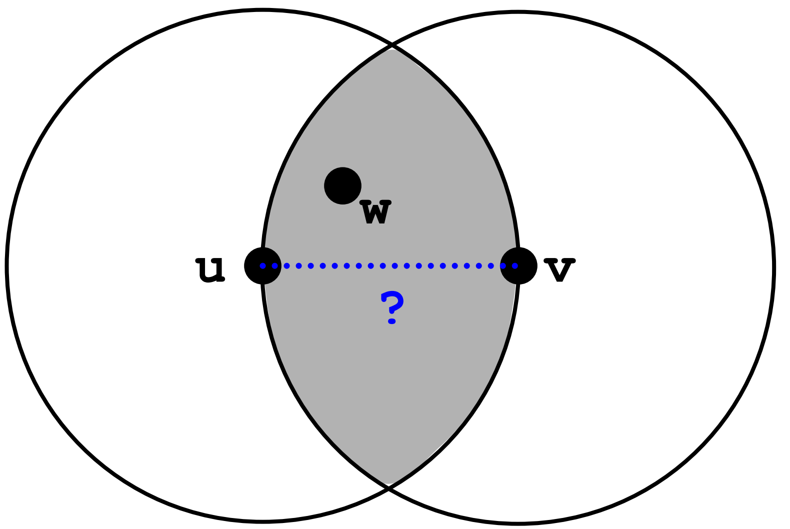
- On graphs with edges that cross (***non-planar*** graphs), **right-hand rule** **may not tour enclosed face boundary**



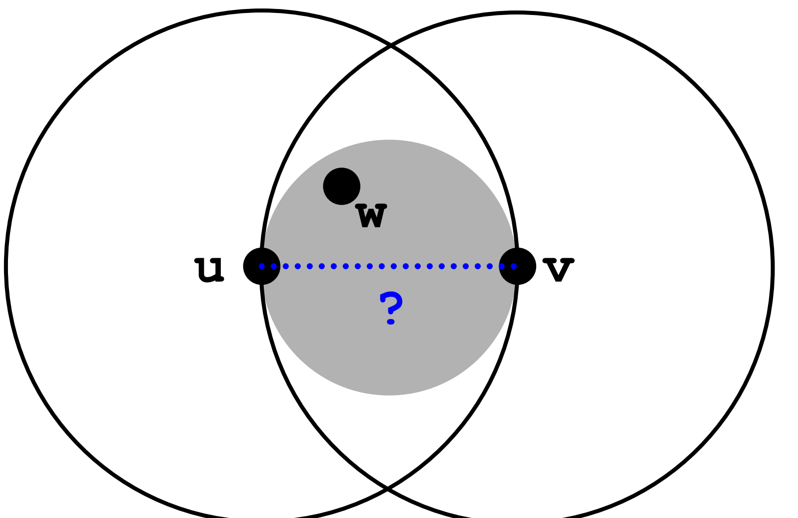
- How to **remove crossing edges** **without partitioning graph?**
  - And using **only single-hop neighbors' positions?**

Planarized Graphs

- Relative Neighborhood Graph (RNG) and Gabriel Graph (GG)
  - **Unit graph connectivity assumption**



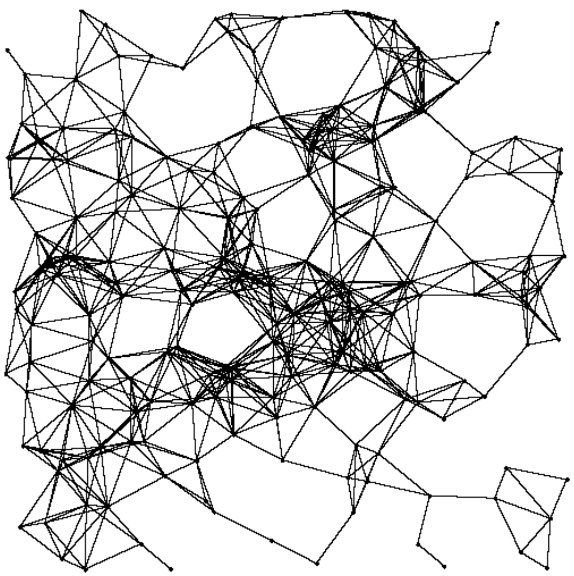
RNG



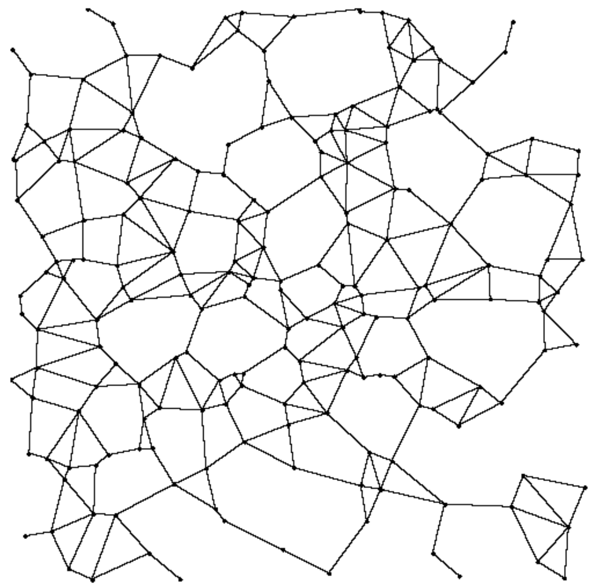
GG

- RNG ⊆ GG

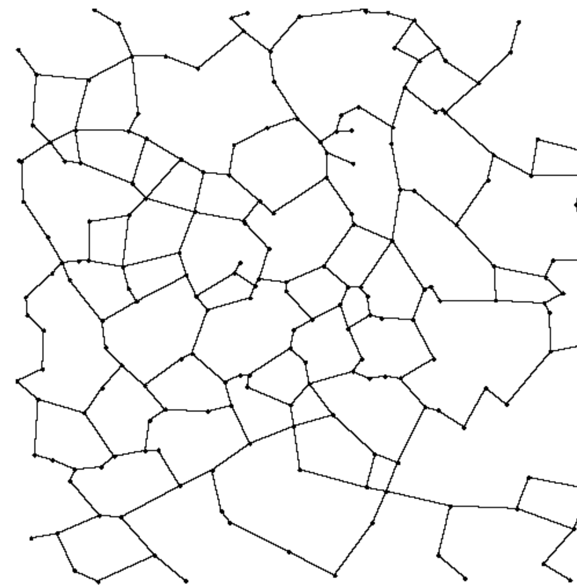
Planarized Graphs



Full Graph

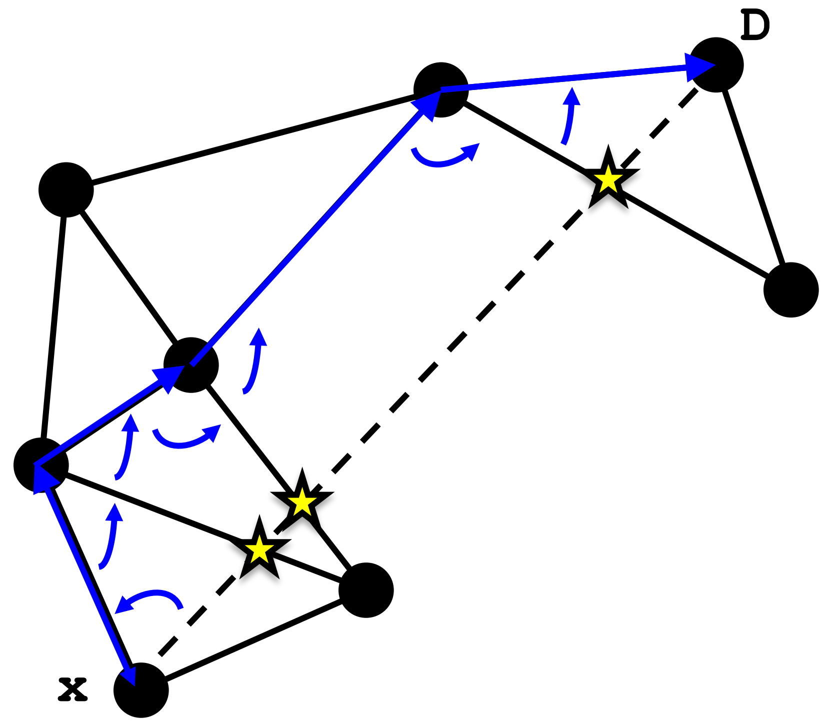


Gabriel Graph Subgraph (GG)



Relative Neighborhood Subgraph (RNG)

Perimeter Mode Forwarding



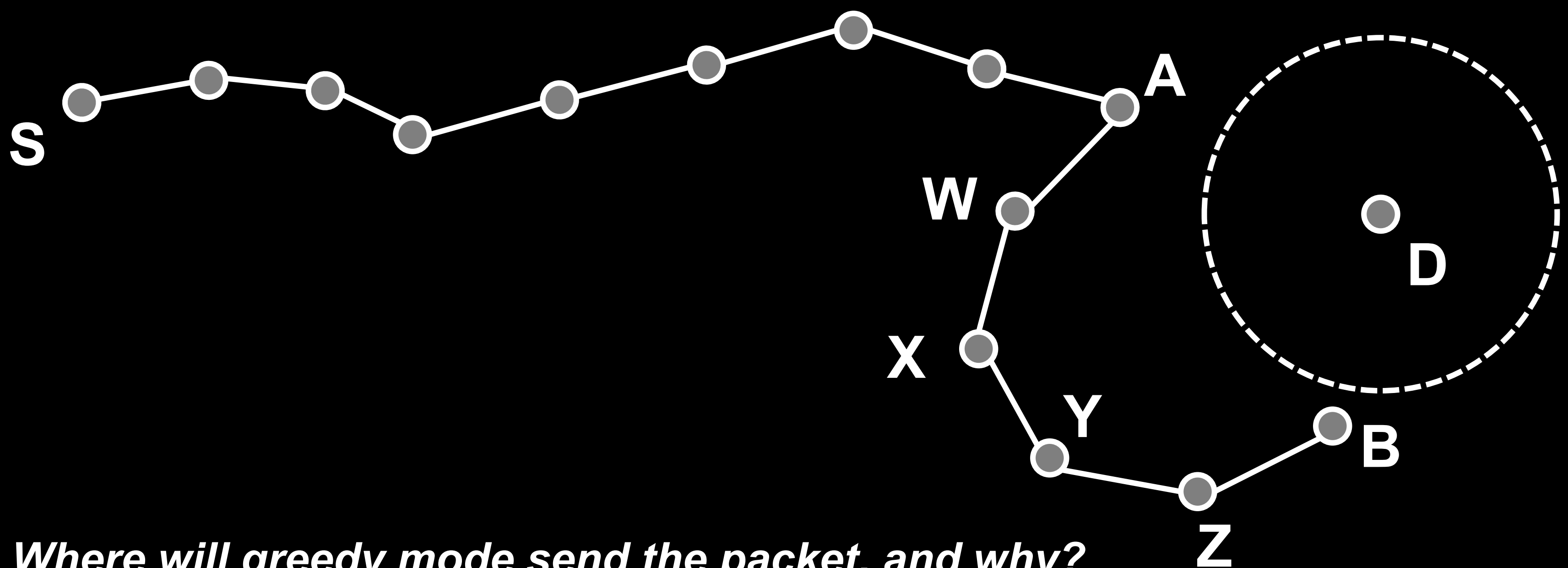
- *x* forwards packet to first edge counterclockwise about *x* from line *xD*
- Traverse face by **right-hand rule,** until crossing *xD* **at a point that is closer** than *x* to *D*
- **Face change:** Repeat with next-closer face, and so on

Full Greedy Perimeter Stateless Routing

- All packets **begin in greedy mode**
  - **Greedy mode** uses **full graph**
  - Upon greedy-forwarding **failure:** **(1)** node **marks** **its location** in packet, **(2)** **marks** packet in **perimeter mode**
- **Perimeter mode** packets follow **planar** graph traversal:
  - Forward along **successively closer faces** by right-hand rule, until reaching destination
  - Packets **return to greedy mode** upon reaching node **closer to destination** than perimeter mode **entry point**

Stretch Break and GPSR question

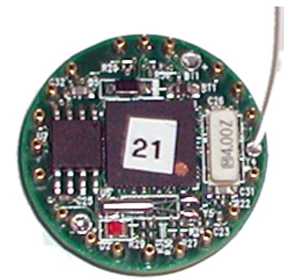
- GPSR/RNG, static network that obeys unit-graph connectivity assumption. **Destination D** is **disconnected** from other nodes



1. ***Where will greedy mode send the packet, and why?***
2. ***Where will GPSR send the packet, and why?***

GPSR: Making it Real

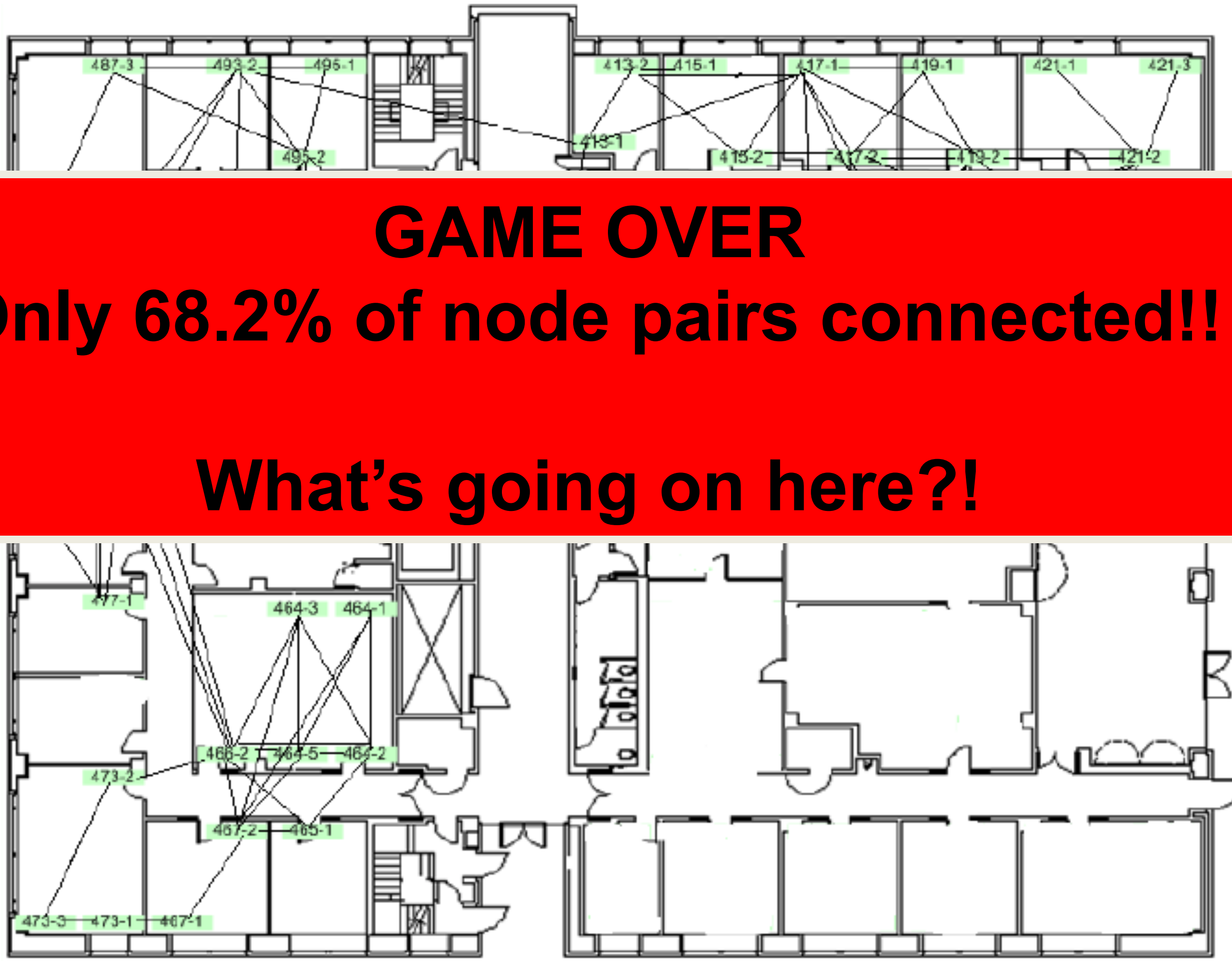
- **GPSR for Berkeley mote sensors**
  - 3,750 lines of nesC code



- Deployed on Mica 2 “dot” mote testbeds
  - 23-node, 50-node subsets of 100-node network in office building (office walls; 433 MHz)

- **Delivery success workload:** 50 packets between **all node pairs,** serially

50-Node Indoor Office Testbed

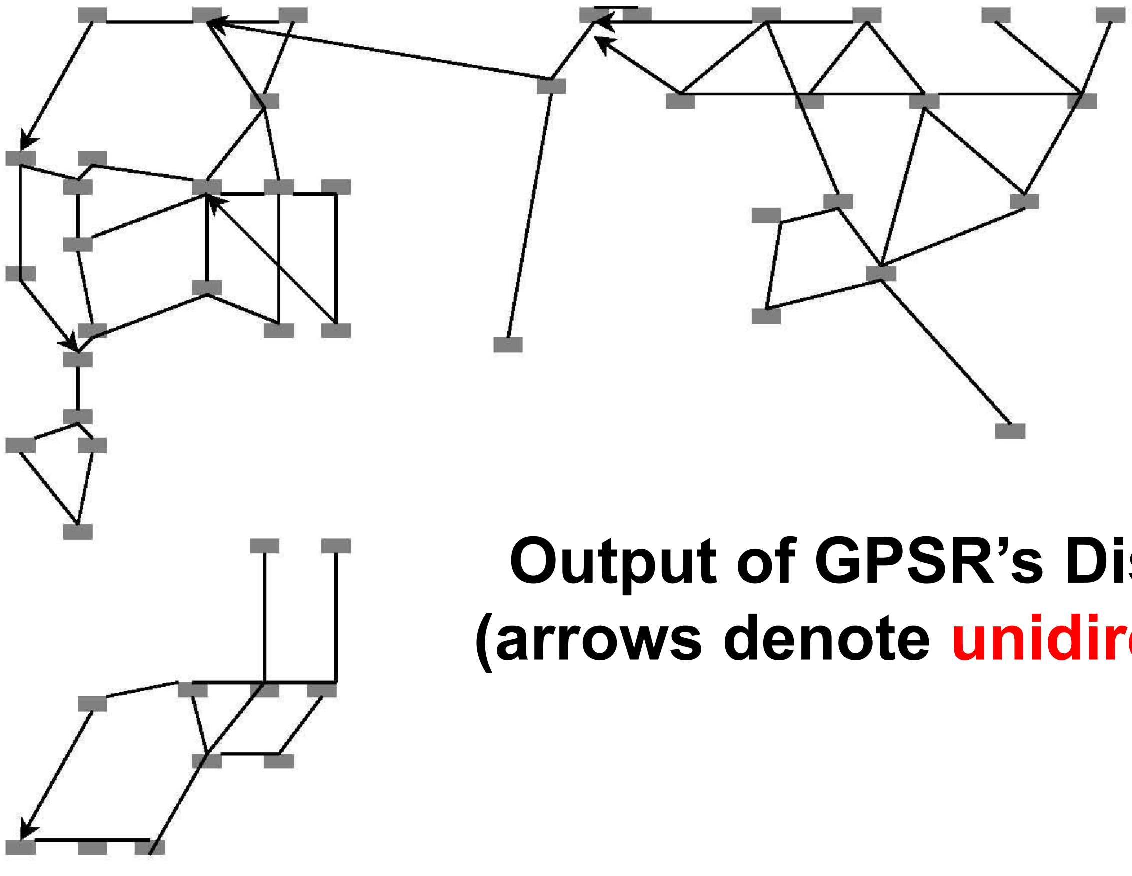


GAME OVER

Only 68.2% of node pairs connected!!

What's going on here?!

Planar, but Partitioned



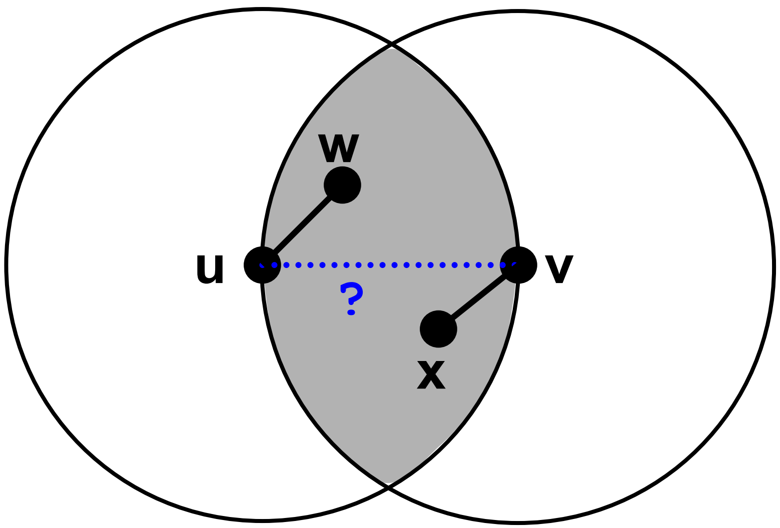
Output of GPSR's Distributed GG (arrows denote unidirectional links)

Assumptions Redux

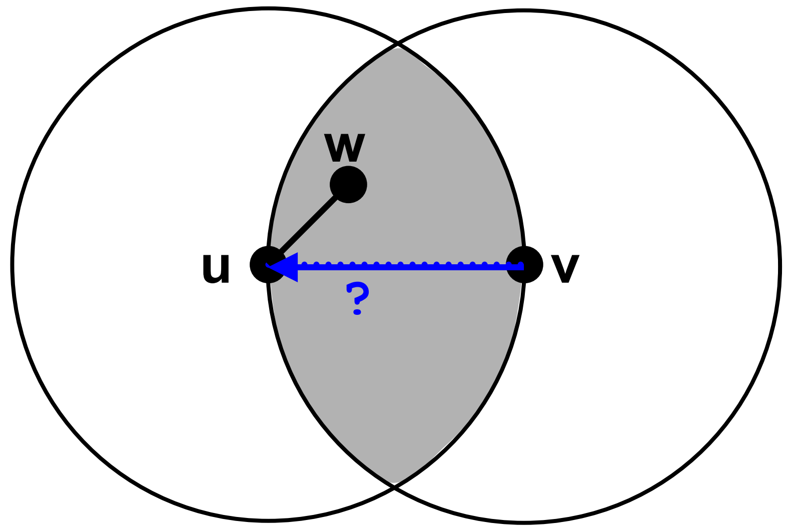
- **Bi-directional radio links** (unidirectional links may be excluded)
- Network nodes placed roughly **in a plane**
- ~~***Unit Graph:*** **Node always connected to all nodes within a fixed radio range, and none outside this range**~~
- Fixed, **uniform** radio **transmitter power**

Absorption, reflections, interference, antenna orientation differences, lead to *non-unit* graphs

Planarization Pathologies

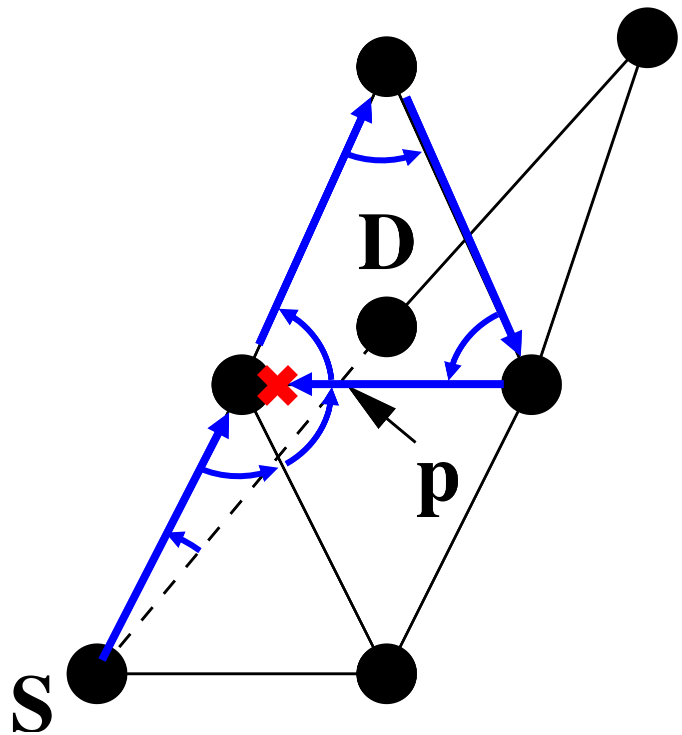


Partitioned RNG



RNG w/Unidirectional Link

Face Routing Failure (Non-Planar)



- Crossing links may cause **face routing to fail**

Cross-Link Detection Protocol (CLDP): Assumptions and Goals

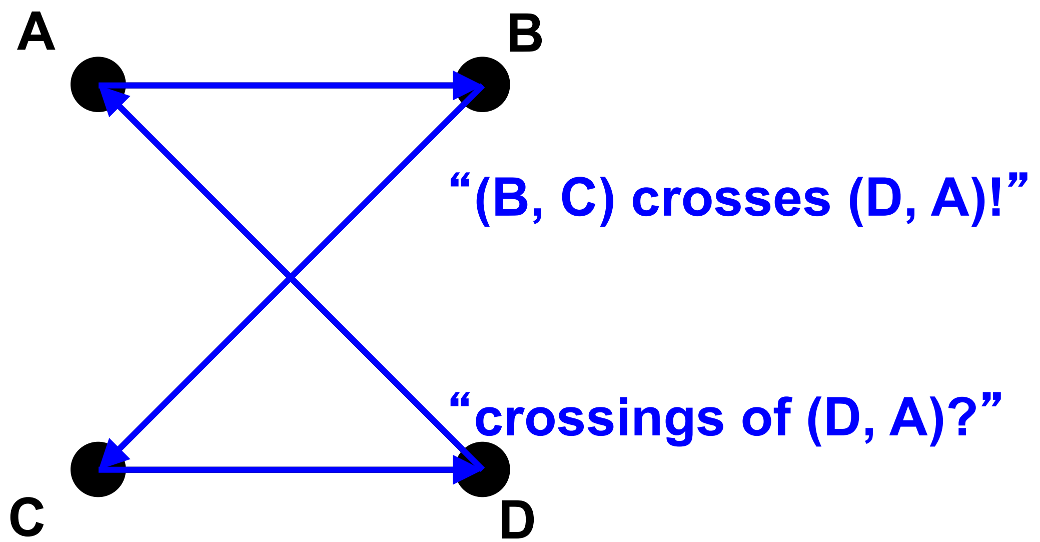
- Assumptions, revised:
  - Nodes know their own positions in 2D coordinate system
  - Connected graph
  - Bidirectional links
  - **No assumption whatsoever** about structure of graph

- Seek a "planarization" algorithm that:
  - never partitions graph
  - always produces a **routable** graph; one on which GPSR routing never fails **(may contain crossings!)**

CLDP Sketch

- Nodes explicitly probe each of their own incident ***candidate*** links to **detect crossings by other links**
  - Probe packet **follows right-hand rule;** carries **locations of candidate link endpoints**
  - Probe packet records **first crossing link** it sees *en route*
- One of two crossing links "eliminated" when probe returns to originator
  - Originator may **mark candidate link *unroutable*** OR
  - **Request** remote crossing link be marked **unroutable**
- Probe and data packets only traverse **routable** links

CLDP: A Simple Example

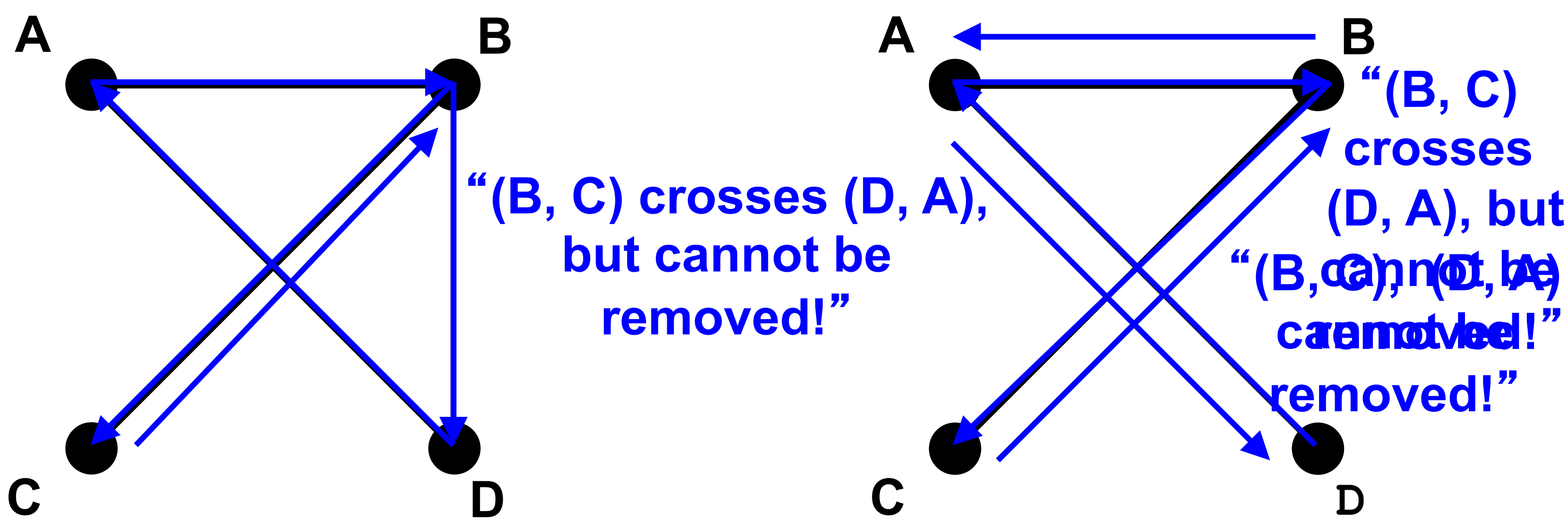


- All links initially marked "routable"
- Detected crossings result in link transition to "unroutable" (by D, or by B or C)

In a dense wireless network, most perimeters short (3 hops); most probes traverse short paths

CLDP and Cul-de-sacs

- Cul-de-sacs give rise to links that cannot be eliminated without partitioning graph
- Not all {edges, crossings} can be eliminated!



Routable graphs produced by CLDP may contain crossings, but these crossings never cause GPSR to fail

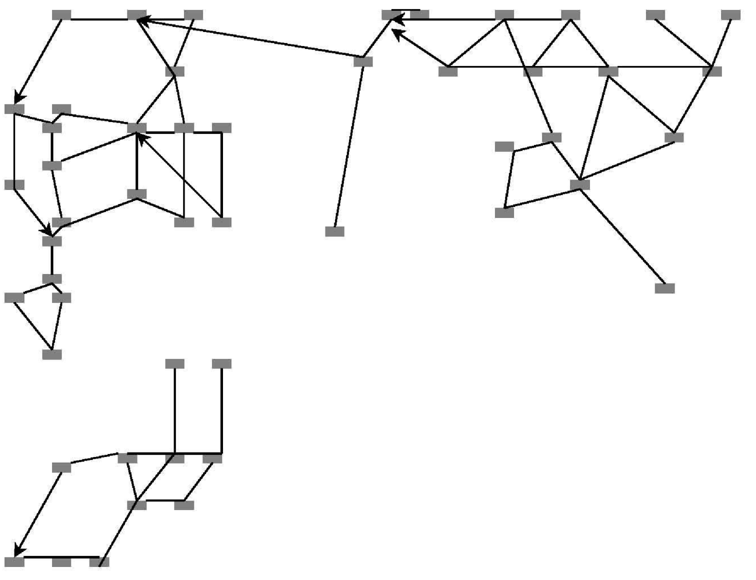
Summary: CLDP Protocol

- Link ***removable*** when a probe traverses either the link being probed (or its cross-link) in **only one direction**

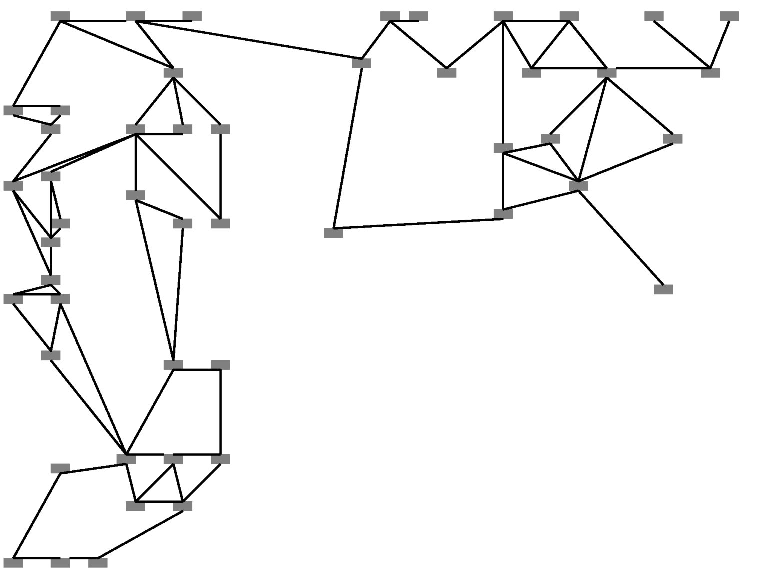
- If link *L* probed, crossing link *L'* found:
  - both *L* and *L'* removable: **remove *L***
  - *L* removable, *L'* not removable: **remove *L***
  - *L* not removable, *L'* removable: **remove *L'***
  - neither *L* nor *L'* removable: **remove no link**

Given a static, connected graph, CLDP produces a graph on which GPSR succeeds for all node pairs

Meanwhile, back in the testbed…

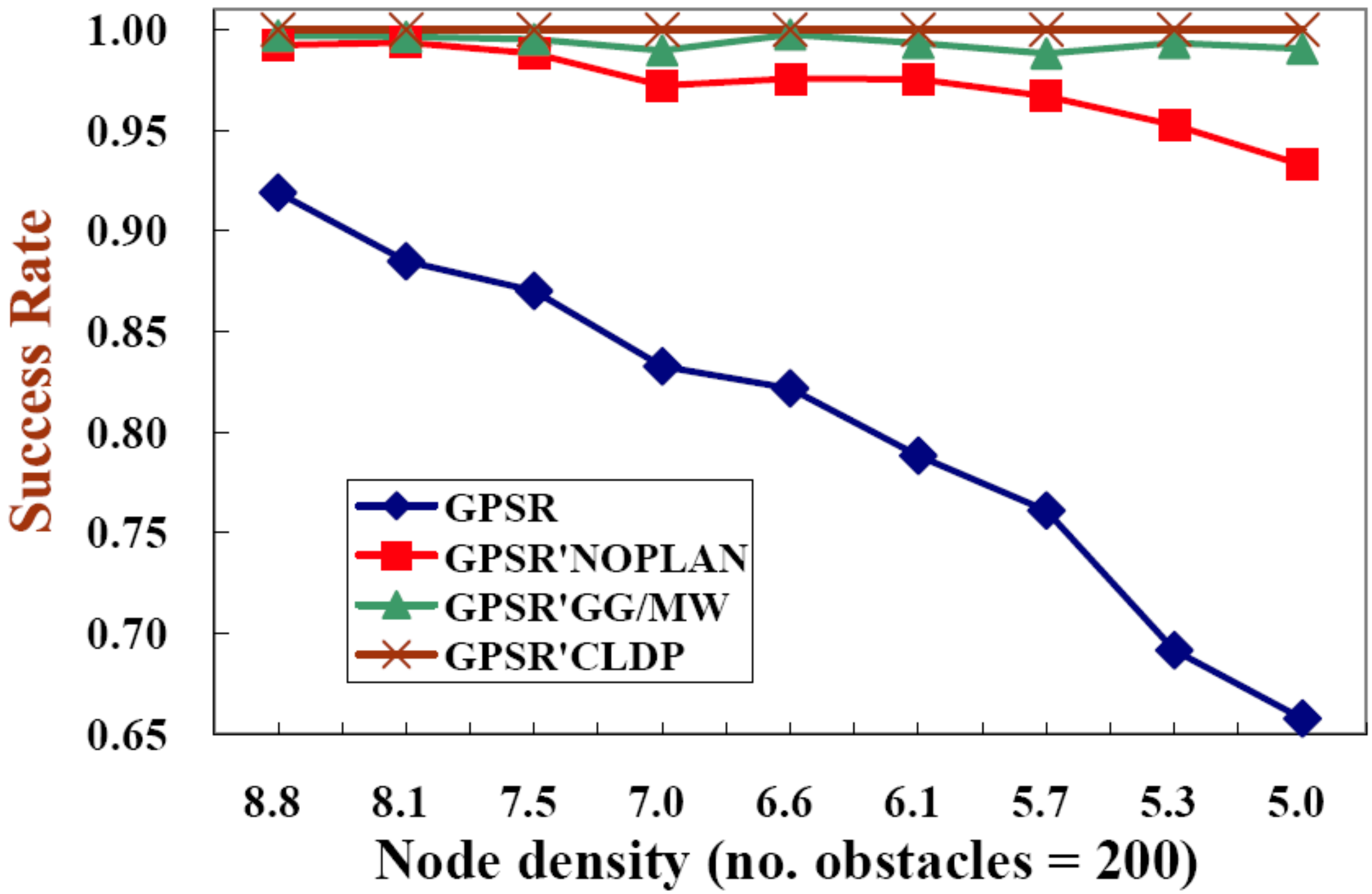


GG



CLDP

CLDP: Packet Delivery Success Rate (200 Nodes; 200 Obstacles)



Geographic Routing: Conclusions

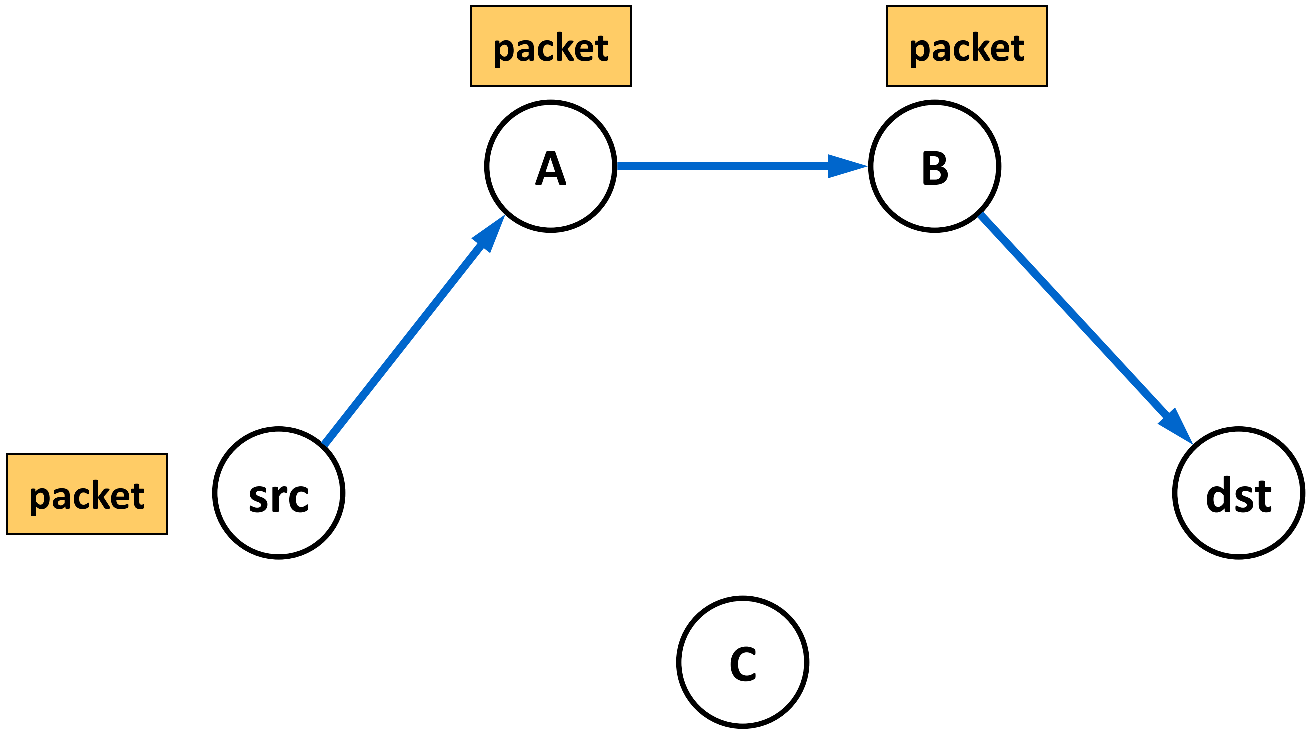
- **Resource constraints, failures, scale of deployment** make design of sensornet systems hard
- Geography a useful primitive for building sensor network applications (e.g., spatial queries)
- Any-to-any routing, with **GPSR and CLDP**
  - **O(density) state per node, correct on all networks**
- Geographic routing an example of the difference between paper designs and building real systems!

Today

1. Geographic (Location-Based) Mesh Routing

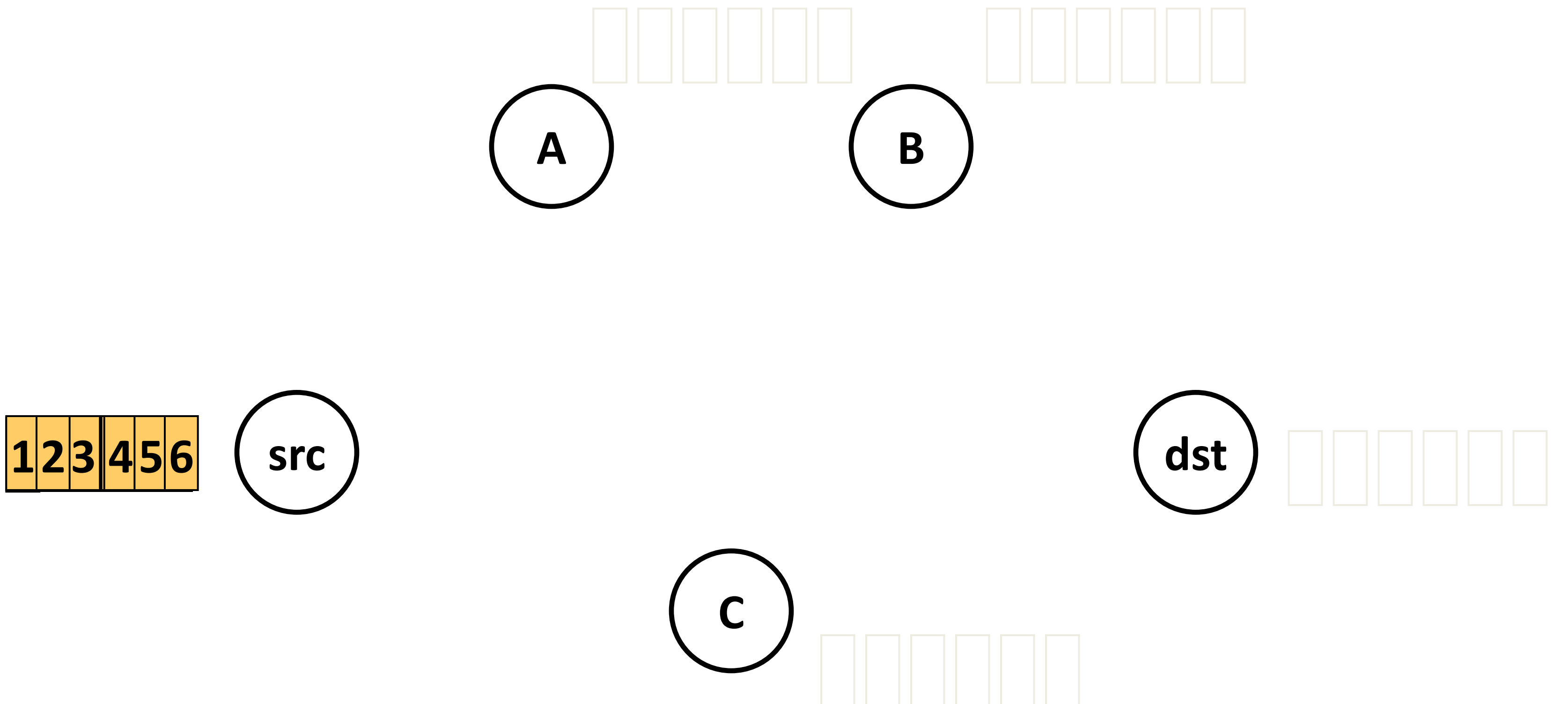
2. **Diversity Mesh Routing**
   - **ExOR (Roofnet)**
   - Network Coding

Initial approach: Traditional routing



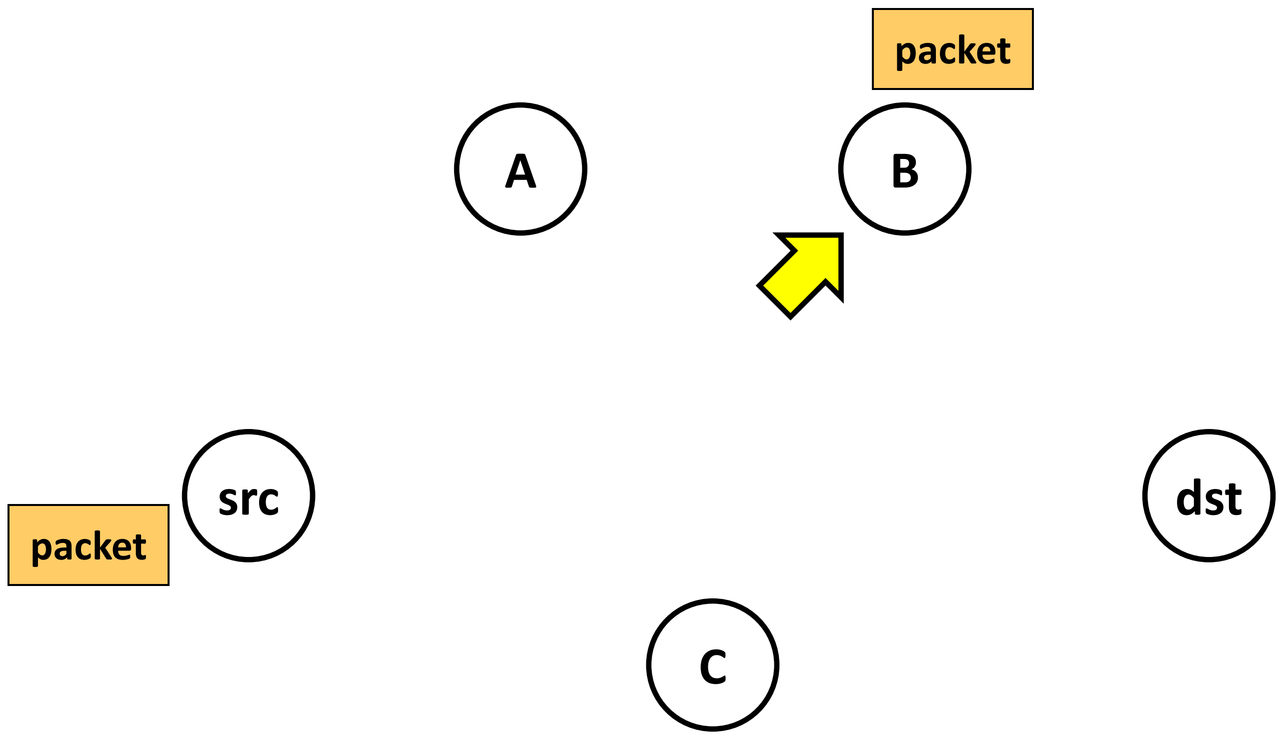
- Identifies a **route,** forward over those links
- Abstracts radio to look like a wired link

But radios aren't wires



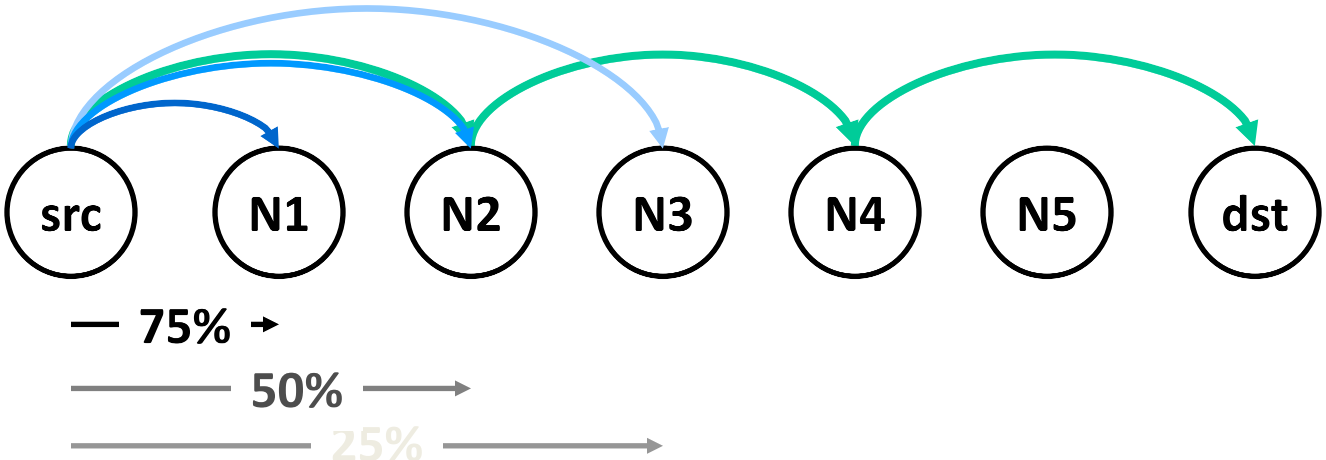
- Every packet is **broadcast**
- Reception is **probabilistic**

ExOR: exploiting probabilistic broadcast



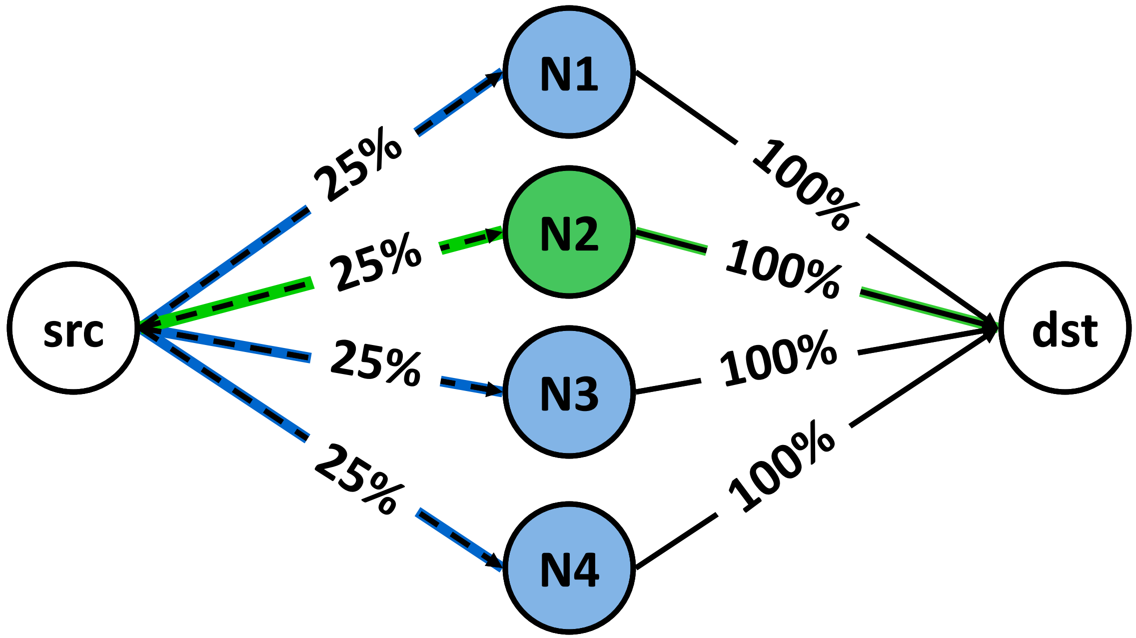
- Decide who forwards **after** reception
- Goal: only ***closest*** receiver should forward
- **Challenge:** agree efficiently, avoiding duplicate xmits

Why ExOR might increase throughput? (1)



- Throughput $\cong {}^{1}/_{\text{\# transmissions}}$
- Best traditional route is over the 50% hops: $3({}^{1}/_{0.5}) = 6$ tx
- ExOR exploits **lucky long** receptions
- ExOR recovers **unlucky short** receptions

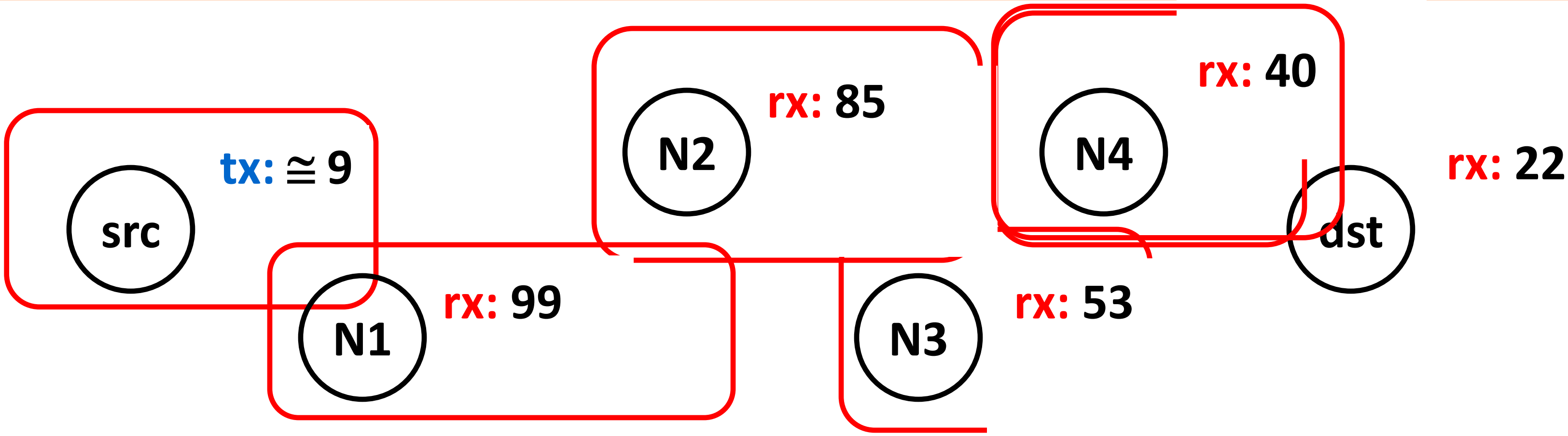
Why ExOR might increase throughput (2)



- Traditional routing: $^1/_{0.25} + 1 = 5$ tx
- ExOR: $^1/_{(1-(1-0.25)^4)} + 1 \approx 2.5$ transmissions

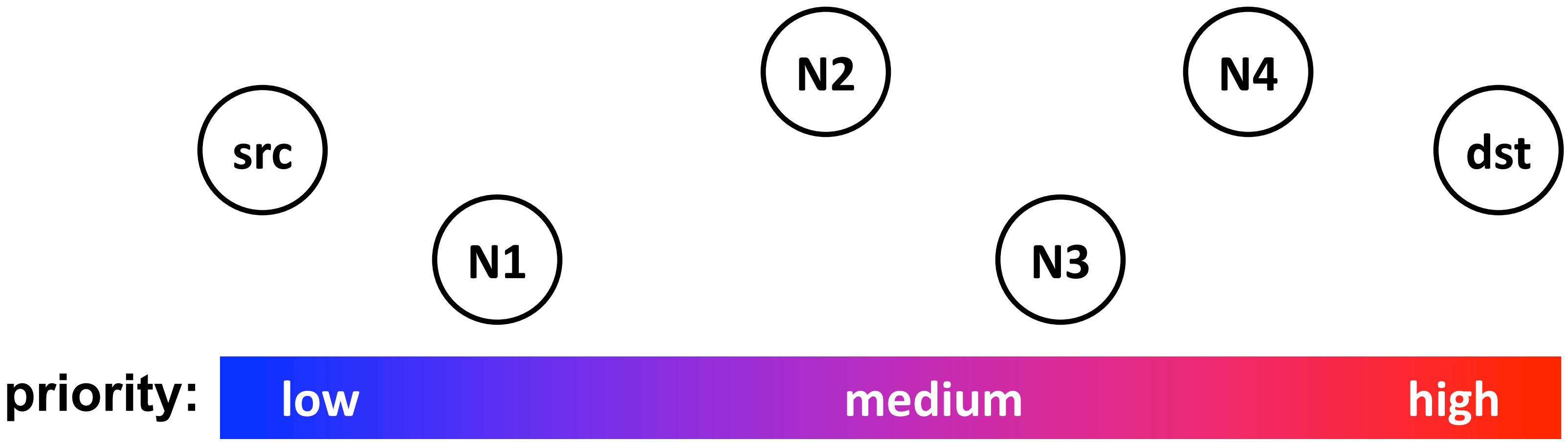
- **Diversity of links, paths** in mesh networks

ExOR packet batching



- Finding the closest receiver involves coordination **overhead**
  - Want to **avoid** paying this overhead once per packet
- **Idea:** Send **batches of packets** to amortize overhead
- Node closest to the destination sends first
  - Other nodes **listen,** send **just the remaining** packets in turn
- Repeat schedule until destination has whole batch

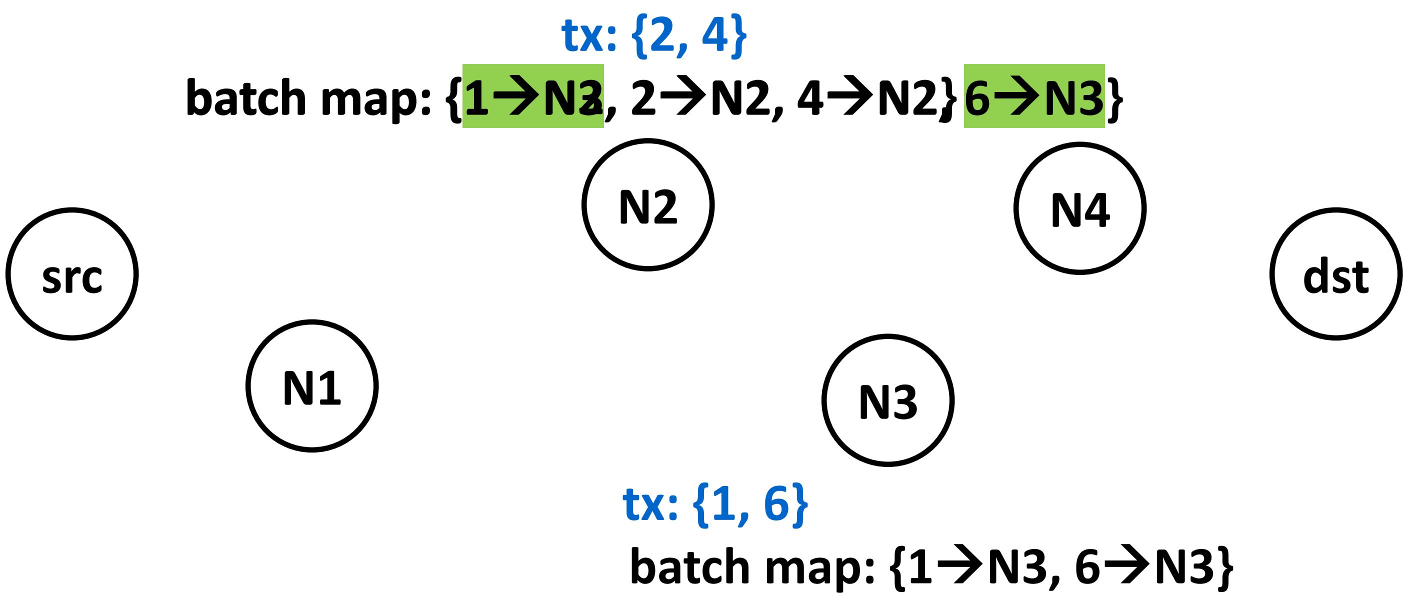
The *forwarder list* establishes transmit order



- One node sends at a time, highest priority first

- Source includes a ***forwarder list*** in ExOR header
  – The forwarder list is sorted by ***path ETX*** metric to dst
    - Link ETX: Expected number of transmissions required
    - Nodes periodically flood link ETX measurements
    - Path ETX is weighted shortest path (Dijkstra's algorithm)

Batch maps **track who received what**

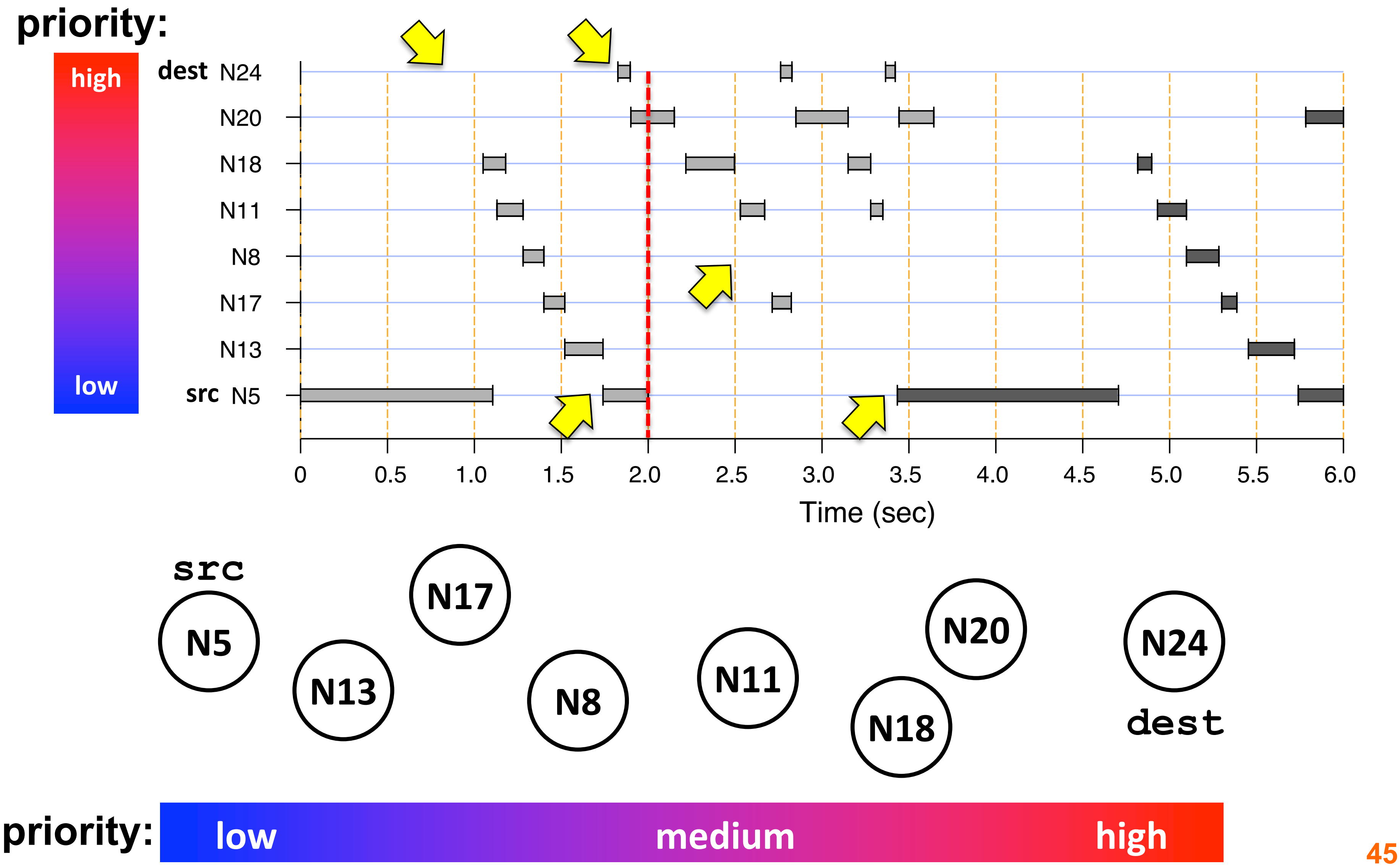


- Nodes include a ***batch map*** in every data packet header
  - For each packet, batch map gives **highest priority node** known to have **received** a copy of that packet
  - Nodes **suppress** packets **higher priority node** received
  - Allows source to **receive acknowledgement**

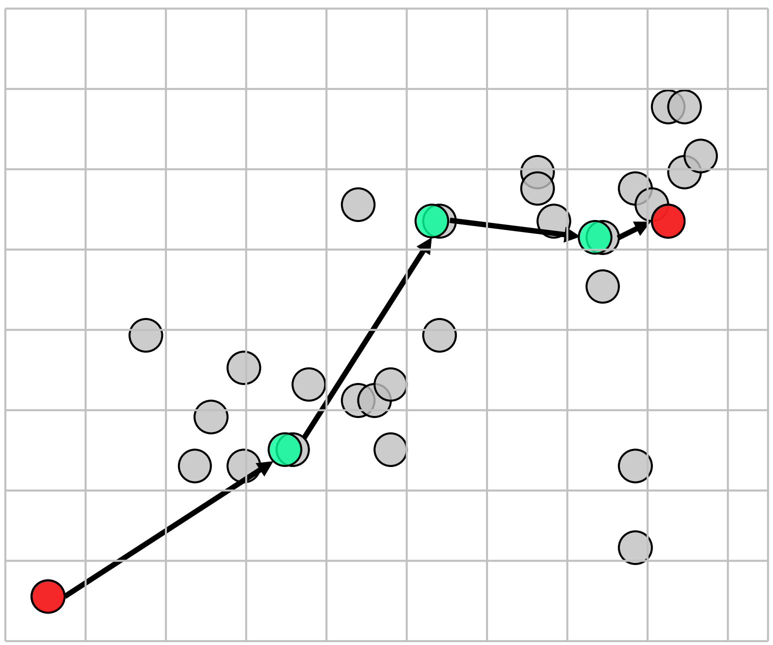
Completion

- If node's batch map indicates higher priority node has received > 90% of the batch, it remains quiet
- **Removes excessive overhead** due to "straggler" packets that get unlucky due to wireless conditions
- ExOR routing itself only guarantees > 90% delivery
- Destination requests remaining < 10% packets via **traditional routing**

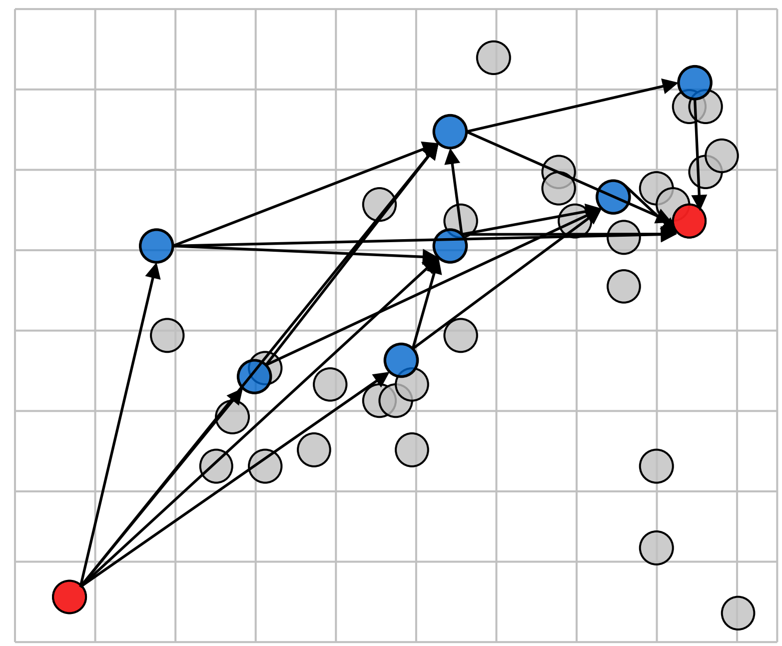
Transmission timeline



ExOR uses more links in parallel

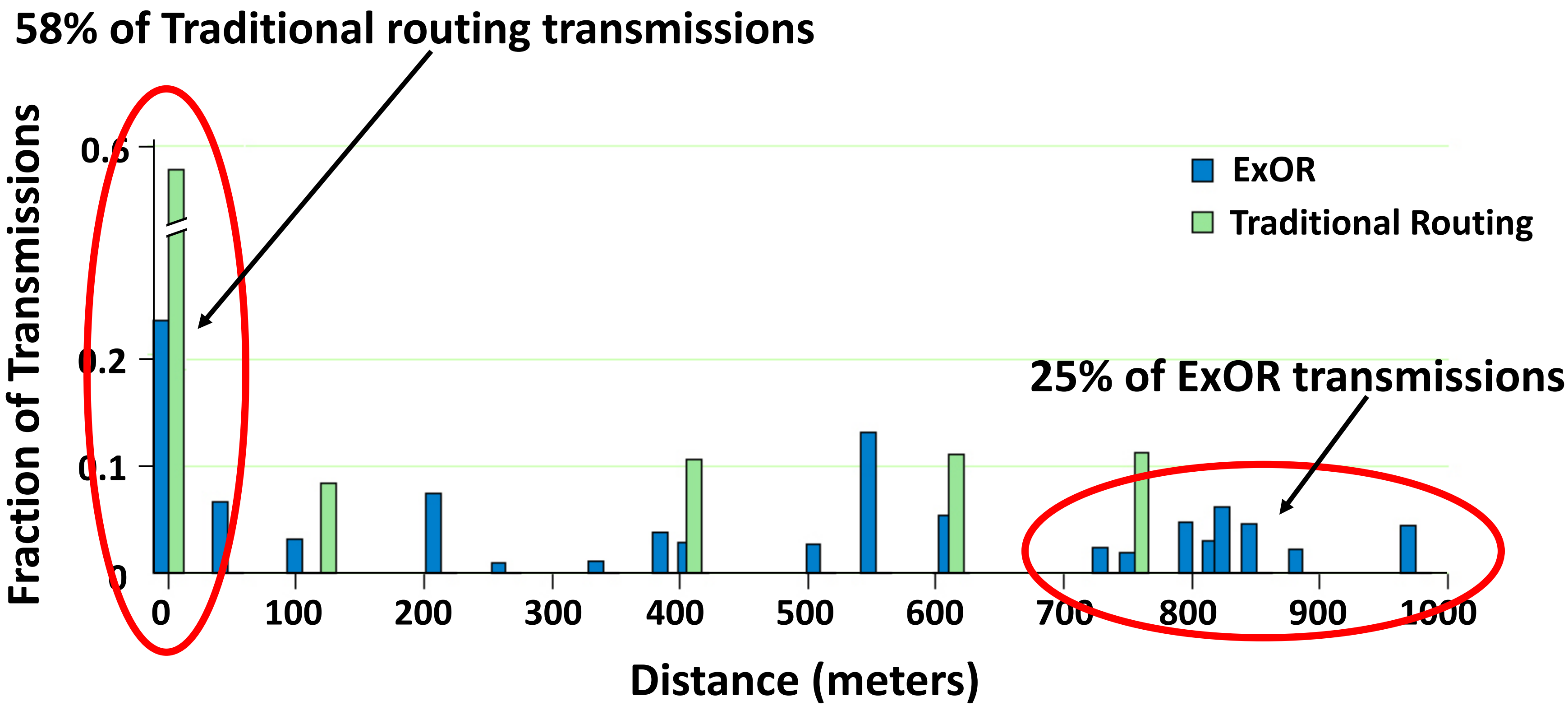


Traditional: 3 forwarders, 4 links



ExOR: 7 forwarders, 18 links

ExOR moves packets farther



ExOR average: 422 meters/tx **Traditional:** 205 meters/tx

Today

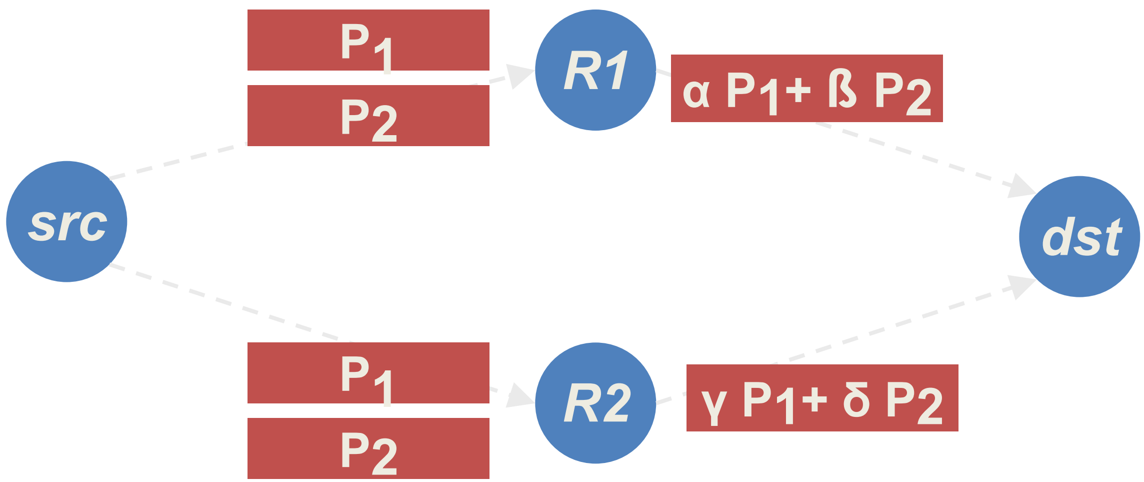
1. Geographic (Location-Based) Mesh Routing

2. **Diversity Mesh Routing**
   - ExOR (Roofnet)
   - **Network Coding**

Network Coding

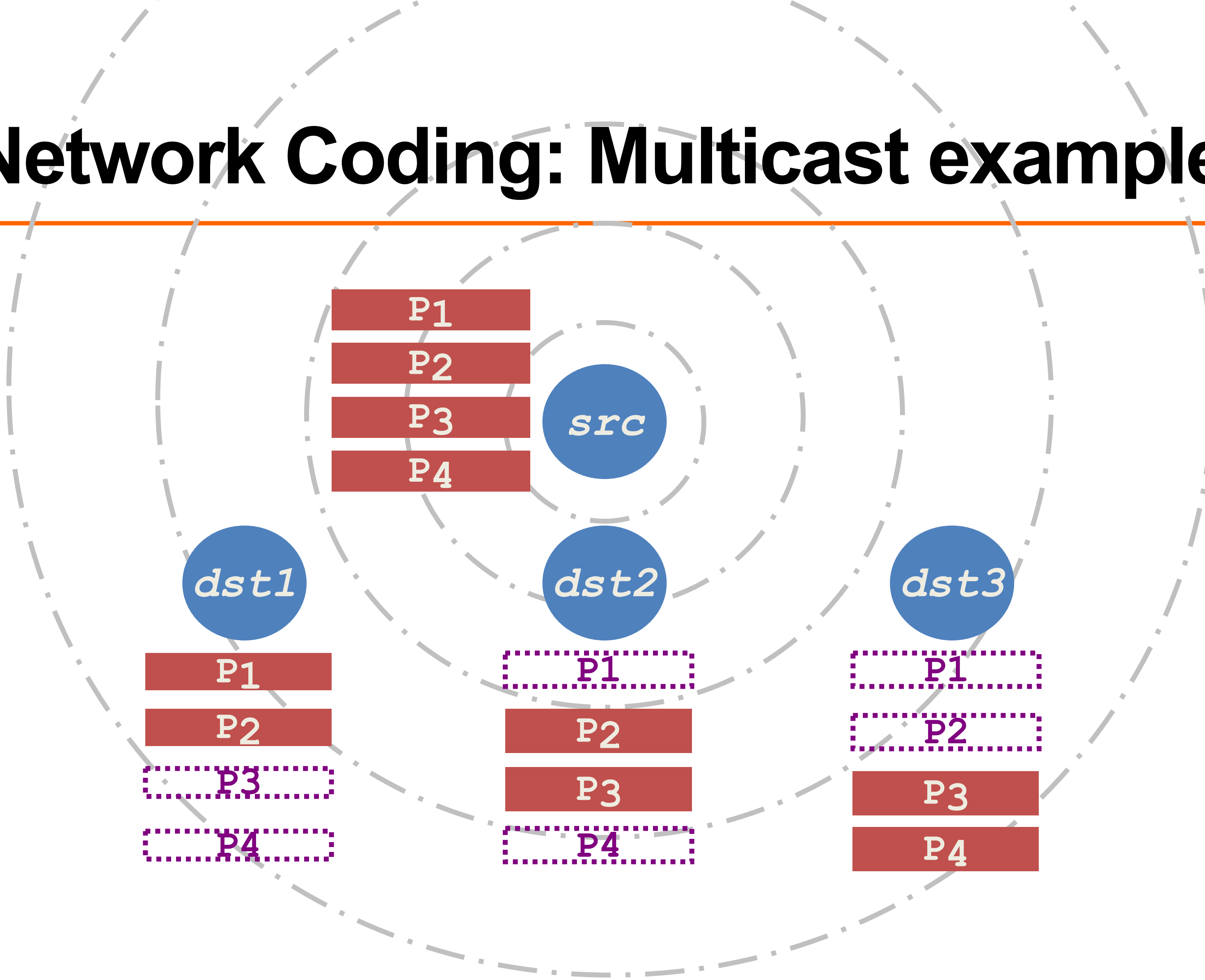
- **ExOR** uses a global scheduler
  - **Requires coordination:** Every node knows who received what
  - **Only one node transmits at a time,** others listen
- **Network Coding Idea:** Nodes do not relay received packets verbatim
  - Instead **combine several packets together** to send in one single transmission

Random Linear Codes



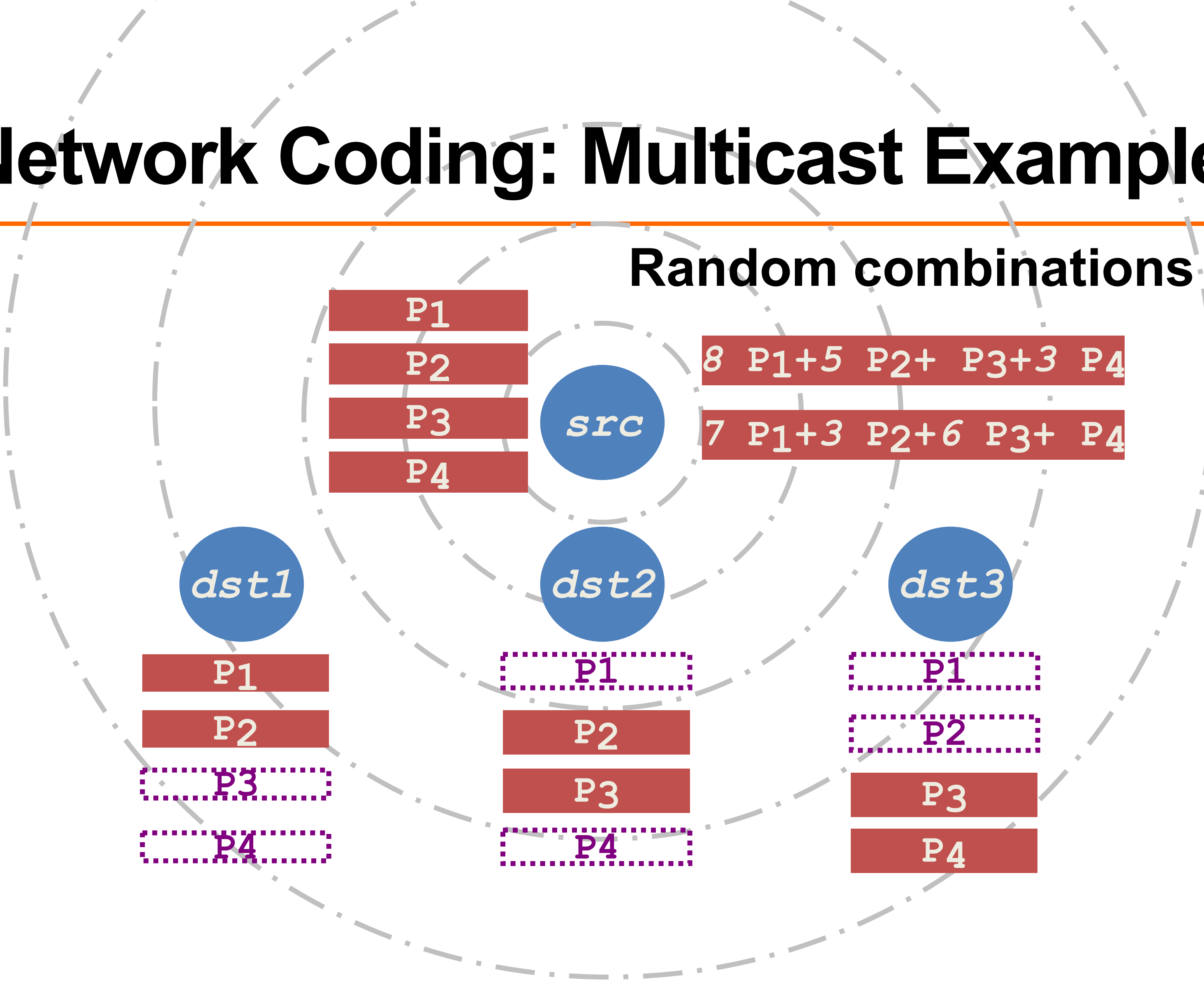
- Each router forwards **random linear combinations** of packets
  - Randomness **makes duplicates unlikely**
  - **No scheduler; No coordination**
    - **Simple, better exploits spatial reuse**

Network Coding: Multicast example



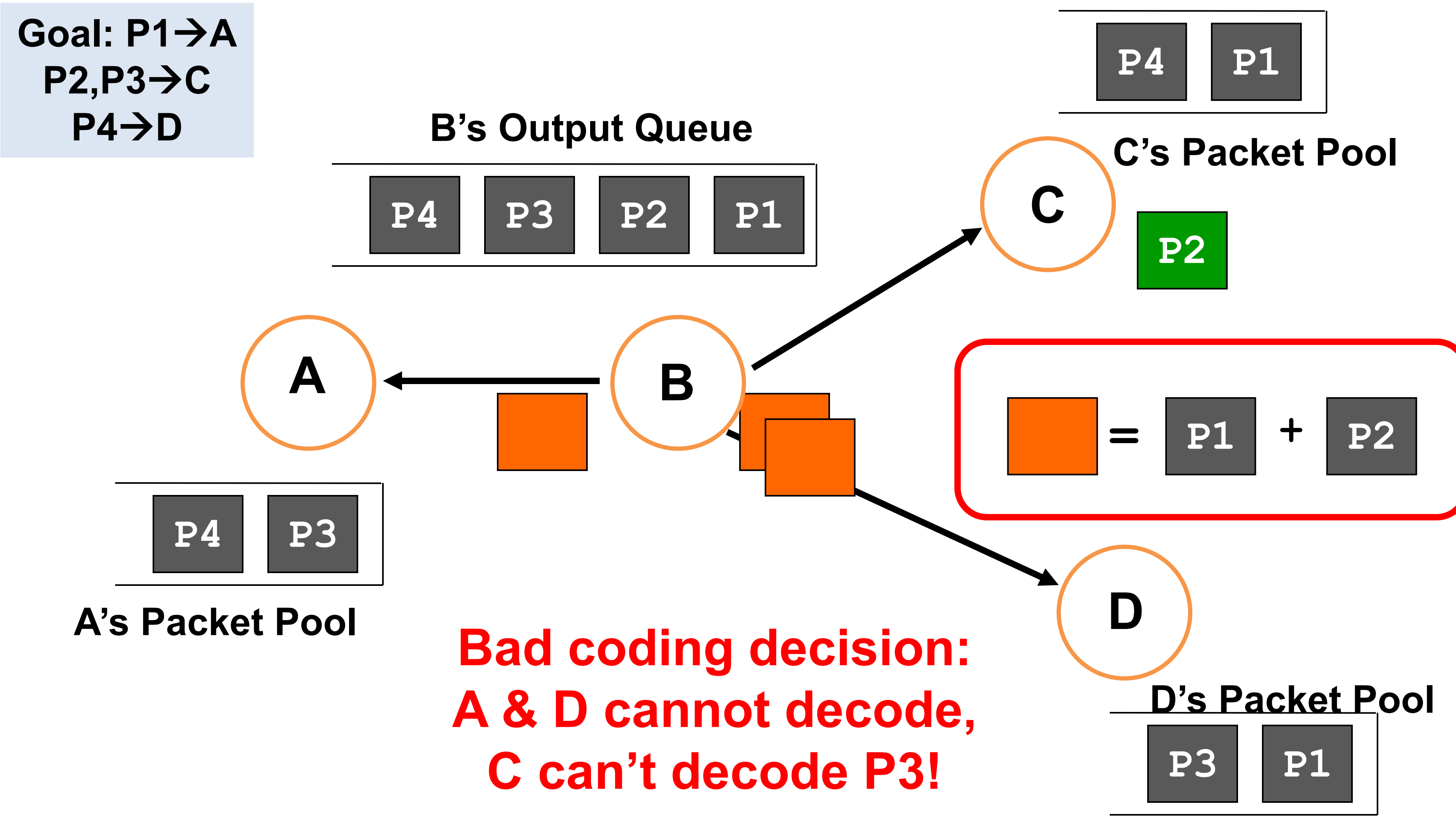
Without coding → source retransmits all 4 packets

Network Coding: Multicast Example



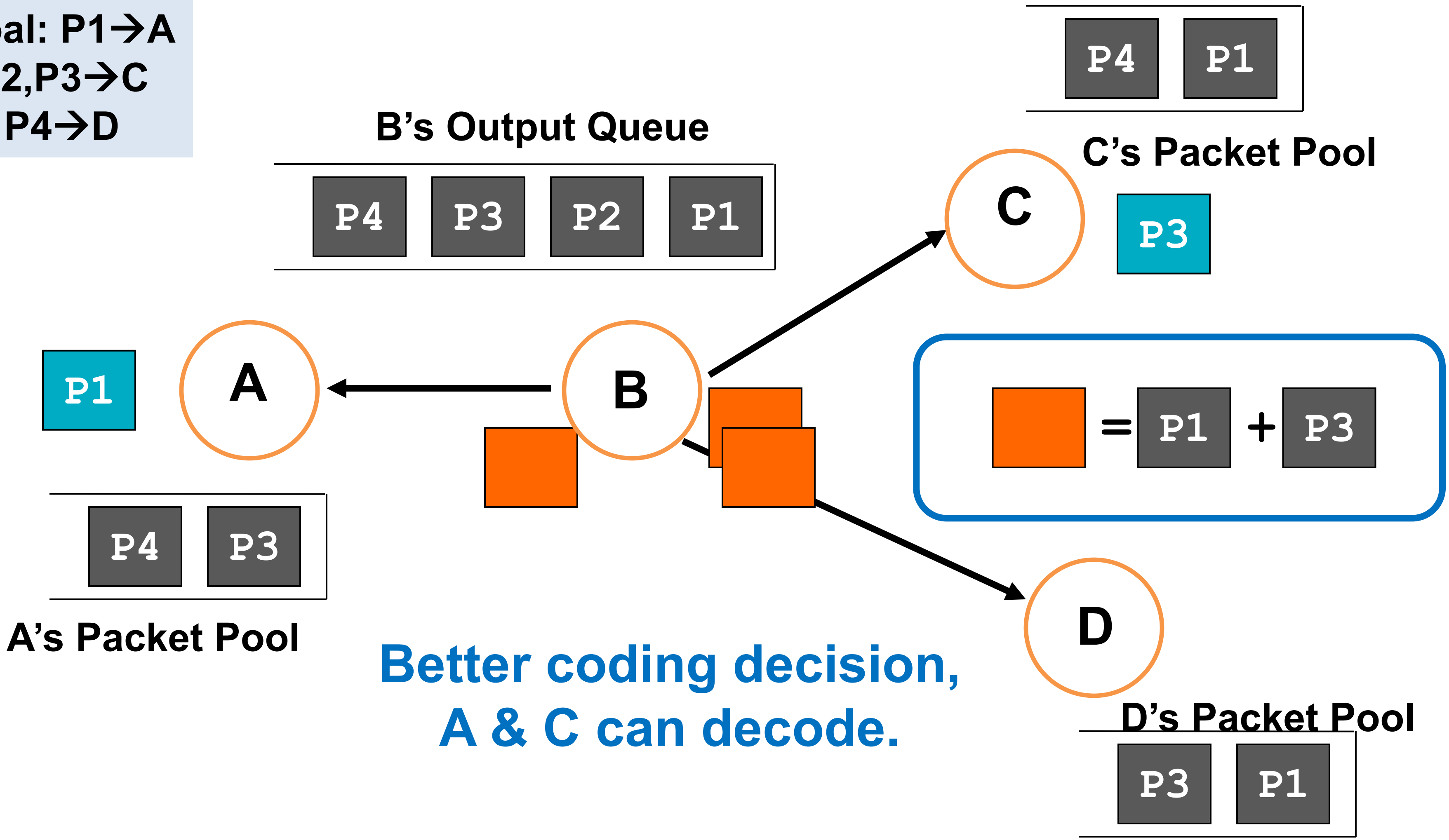
With random coding → 2 packets are sufficient

Choice of Coding Matters



Choice of Coding Matters

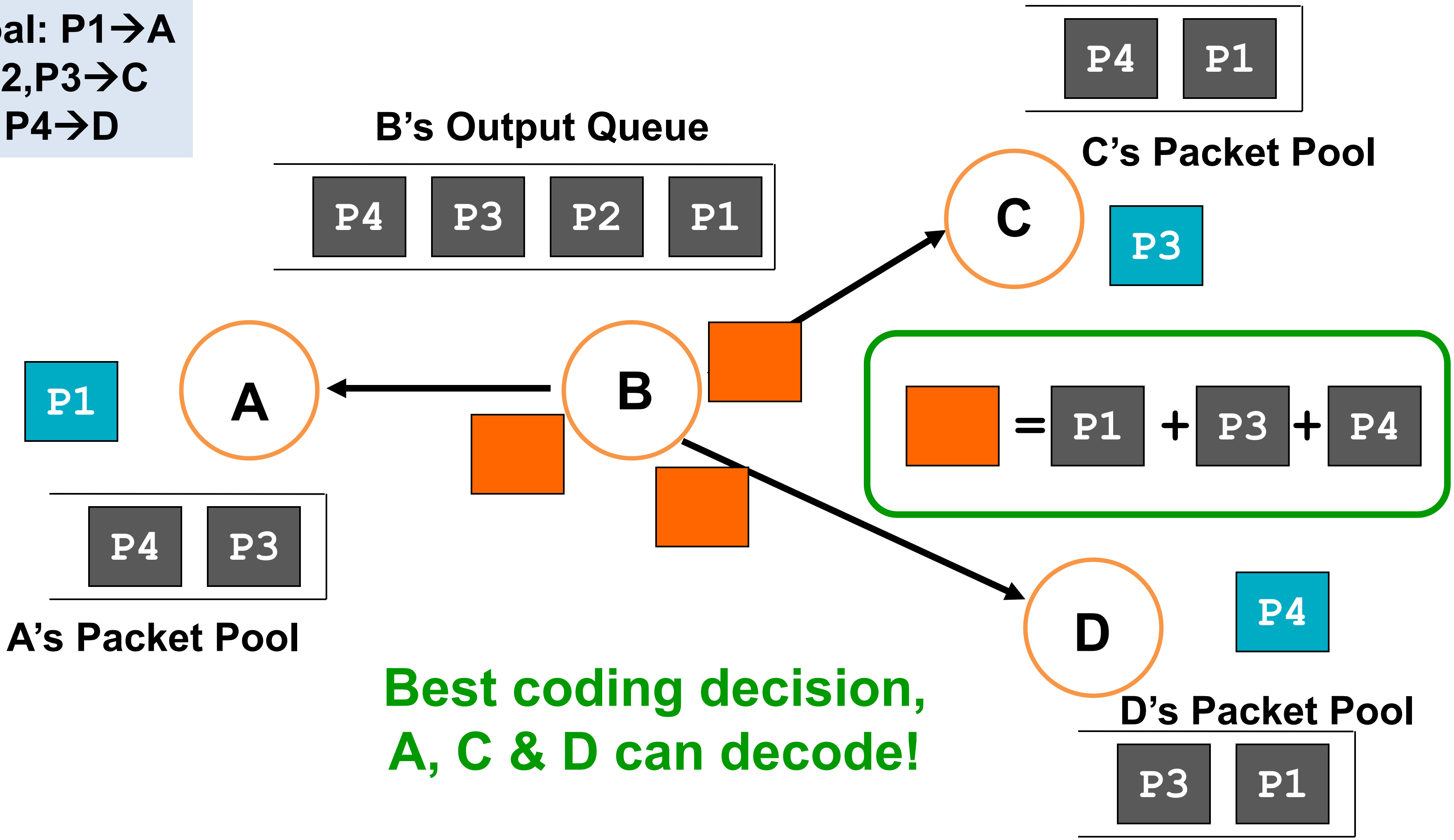
Goal: P1→A
P2,P3→C
P4→D



Better coding decision,
A & C can decode.

Choice of Coding Matters

Goal: P1→A
P2,P3→C
P4→D



Best coding decision, A, C & D can decode!

Network coding: Caveats

- Practical **throughput gains** over ExOR / traditional routing:
  - With **static nodes**
  - **Traffic quantities** need to be **large enough**
    - **Delay increases** **(batching)**
  - **Opposing flows** need to exist in some traffic topologies

Thursday Topic:
[463 Part II: Overcoming Bit Errors]
Detecting and Correcting Errors

Next Week's Precepts:
Lab 2 Introduction