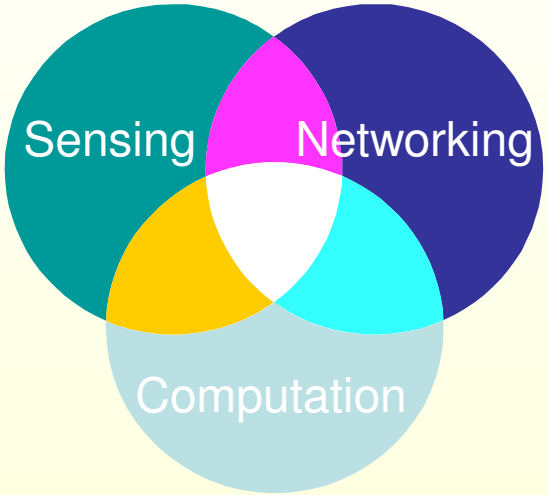
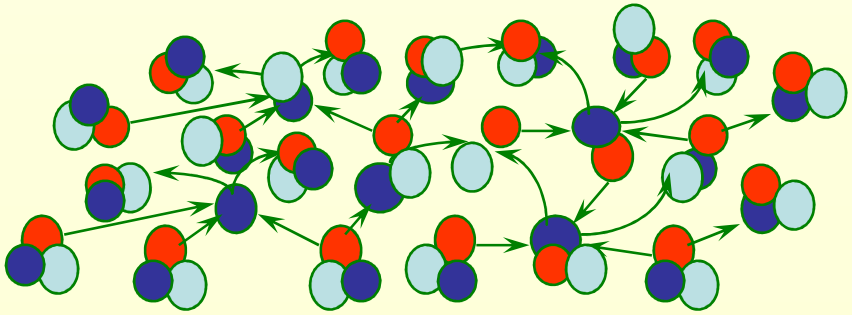
CS321: Information Discovery: Directed Diffusion and DHTs



Leonidas Guibas
Computer Science Dept.
Stanford University



Information Brokerage Services in Dynamic Environments

Information Brokerage

- Information providers (sources, producers) and information seekers (sinks, consumers) need ways to find out about and rendez-vous with each other
- Example: Surveillance from a remote node *r*:
  - *Send to r reports about animal detections in region A every t seconds*
    - Interrogation is propagated to sensor nodes in region *A*
    - Sensor nodes in region *A* are tasked to collect data
    - Data is sent back to the requestor *r* every *t* seconds

Scenario I: Tourists and Animals

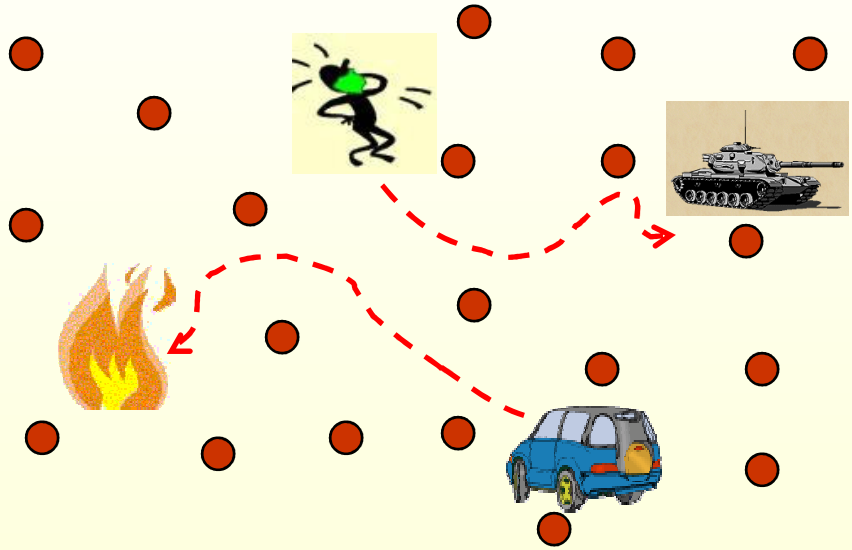
- A sensor network in a safari park.
- A tourist asks: where is the elephant (or giraffe, or zebra)?
- So which sensor has the data about the elephant (or giraffe, or zebra)?

Scenario II: Location Service

- A missing part in routing with geographical or virtual coordinates: how does the source know the location (or virtual coordinates) of the destination?
- Location service: a brokerage service that answers queries such as: where is the node with ID 23?
- Geographical routing:
- The source asks for the location of destination;
- The source routes by using geographical routing.
- Note: chicken and egg problem.

Information Brokerage

- Information providers and information seekers need ways to find out about and rendez-vous with each other
- Challenges in information discovery:
  - Neither knows where the other is
  - Stringent latency requirements
  - Highly dynamic environment
  - Limited computation and communication resources
- Essential for scaling to inter-networks of sensor networks
  - to enable multiple vendors, middleware services, search engines



Providers = sources

Seekers = sinks, consumers

The Challenge is a Dynamic Environment [From D. Estrin]

- The physical world is highly dynamic
  - Dynamic operating conditions (affect sensing, networking)
  - Dynamic availability of resources
    - … *particularly energy*!
  - Dynamic tasks
- Devices must adapt automatically to the current environment and the task requirements
  - Too many devices for manual configuration
  - Environmental conditions are unpredictable
- Unattended and untethered operation is key for many applications

Approach

- Energy is the bottleneck resource
  - And communication is a major consumer – need to avoid unnecessary communication over long distances
- Pre-configuration based on detailed global knowledge is rarely applicable
  - Achieve desired global behavior through localized interactions
  - Must empirically adapt to observed environment
- Leverage points
  - Small-form-factor nodes, densely distributed to achieve physical proximity to sensed phenomena
  - Application-specific, data-centric networks
  - Data processing/aggregation inside the network

Data-Centric Networks

- Traditional networks: routing is based on network ID (e.g., IP addresses).
- Can we design ommunication abstractions are based on data held by nodes, rather than node network addresses?
- Data-centric routing
  - Route to the node with the data the user wants.
- Data-centric storage
  - Store all the data with the general name (elephant) at the same node.

Data-Centric Paradigm

- Data-Centric View
  - The sensor network is queried for *specific data*
  - No *sensor-specific* query
  - Identity of source of data less important than the data
- Application Specific
  - In-sensor *processing*
  - In-sensor *caching* or *storage*

- Localized Algorithms
  - Achieve global objective through local coordination

Abstraction of Data-Centric Routing

- Information producer/consumer game
- Information producer
  - Can be anywhere in the network
  - Dynamic, mobile
  - Multiple producers generating data about the same data type
- Users = information consumers
  - Can be anywhere in the network
  - Can also be mobile
  - Concurrent multiple consumers

Challenges

- Information producers/consumers have do not know about each other.
- Yet we want them to find each other quickly.

- Main approaches:
- Push-based: producers do most of the work.
- Pull-based: consumers actively search.
- Push-pull: both producers/consumers search to find each other.

Brokerage Types

- Discover nodes having certain types of current data or event detections. Assume these node will keep (for a while) producing data of interest – continuous queries.
- Discover events detected in the past through in-network storage – snapshot, or discrete queries.

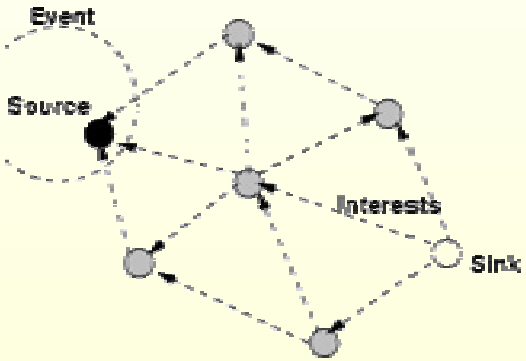
This Class

- Directed diffusion
  - Push-based / continuous queries
- Geographical hash table
  - Push-pull / discrete queries
  - In-network storage

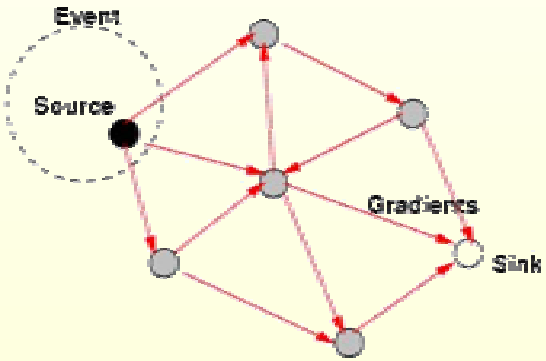
Continuous Queries: Directed Diffusion

Directed Diffusion

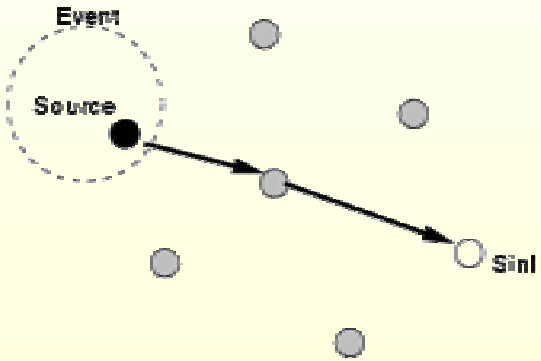
[Intanagonwiwat, Govindan, Estrin '00]



(a) Interest propagation



(b) Initial gradients set up



(c) Data delivery along reinforced path

Directed Diffusion Concepts

- Application-aware communication primitives
  - expressed in terms of named data *(not in terms of the nodes generating or requesting data)*
- A consumer of data, a sink node, initiates an interest in data with certain attributes
- Nodes diffuse the interest towards data producers (sources), via a sequence of local interactions
- This process sets up gradients in the network which channel the delivery of data
- Reinforcement (positive *and* negative) is used to converge to efficient routes
- Intermediate nodes opportunistically fuse interests, aggregate, correlate, or cache data ...

Data Naming

- Content-based naming
  - Data is named by attribute–value pairs
  - This makes matching of interests with data simple
  - Selecting a naming scheme is important and more complex ones can be considered
  - The nodes where information resides are not part of the naming scheme

Request: Interest

```
type = four-legged animal
   interval = 20 ms
   duration = 10 seconds
   rect = [-100,100,200,200]
```

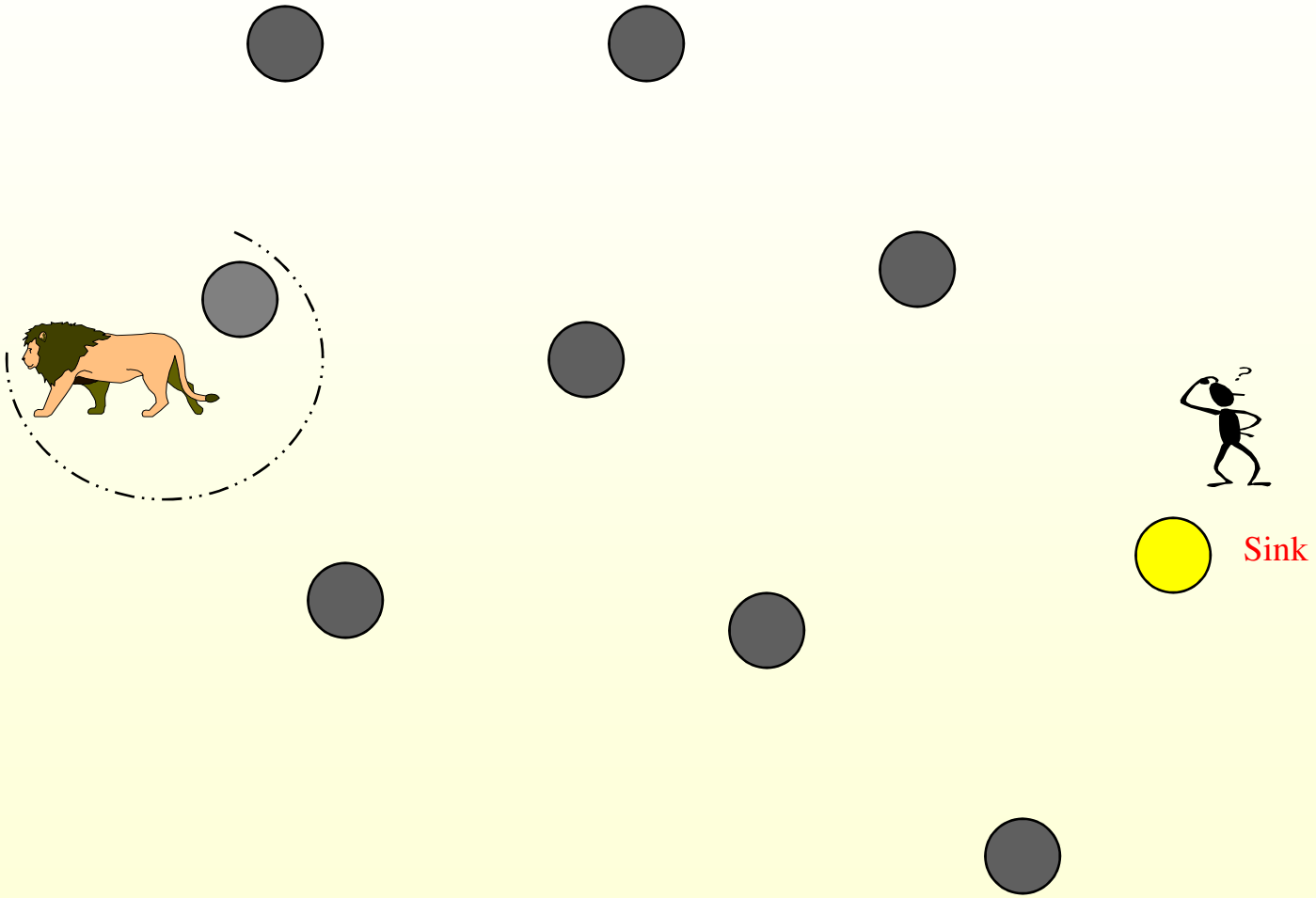
Reply: Data

```
type = four-legged animal
   instance = elephant
   location = [125, 220]
       Intensity = 0.6
       confidence = 0.85
       timestamp = 01:20:40
```

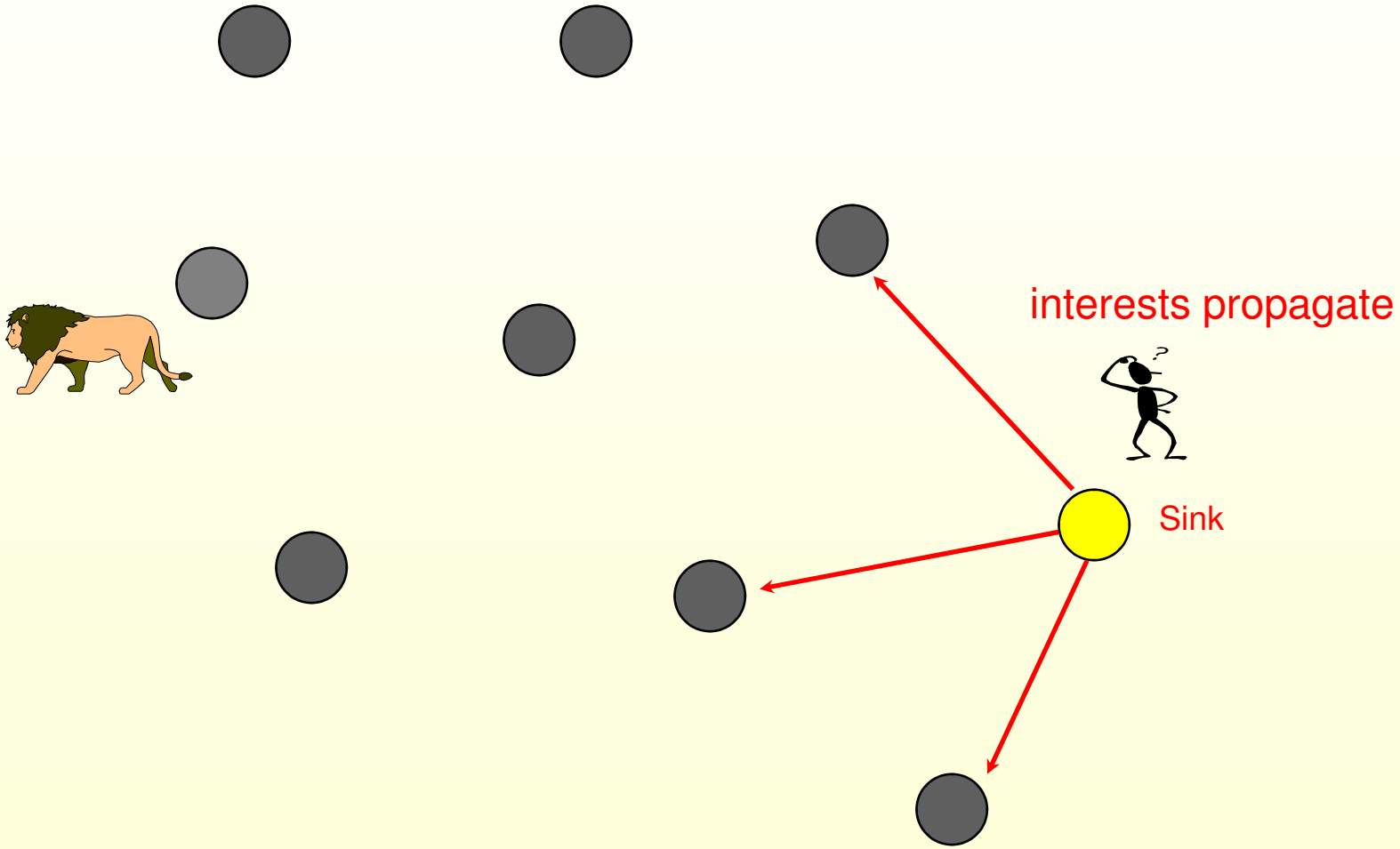
Interests and Gradients

- Interests describe data needed by a node in the sensor network
  - Interests are injected into the network at sinks.
  - Sinks broadcast the interest.
  - An interval specifies the event data rate desired.
  - *Initially, requested rates are much lower than needed.*
  - Each node maintains an interest cache.
- Each cache interest entry also contains gradients.
  - Specifies a data rate and a direction of data flow for each requesting neighbor
  - Data flows from the source to the sink along the gradient links

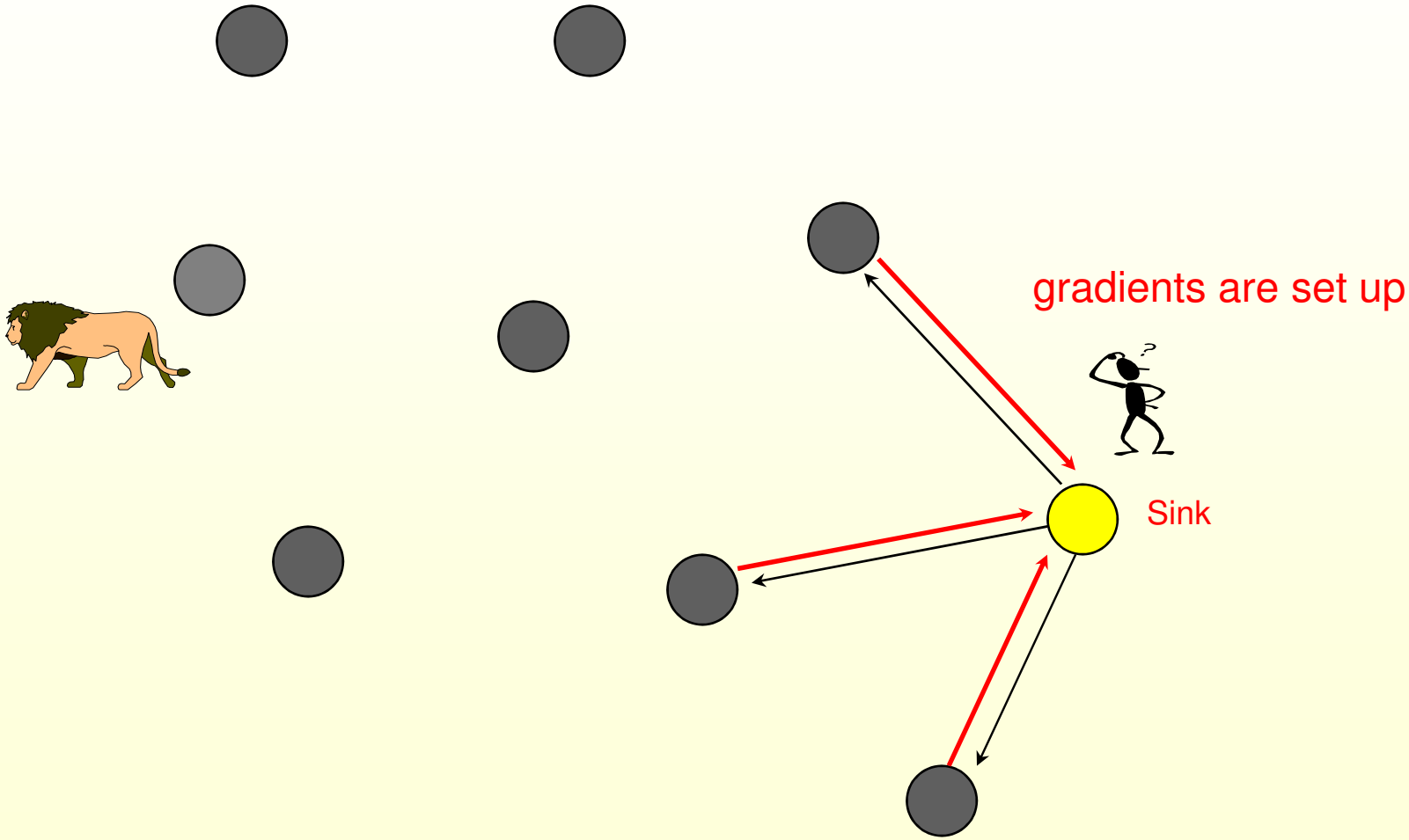
Illustrating Directed Diffusion



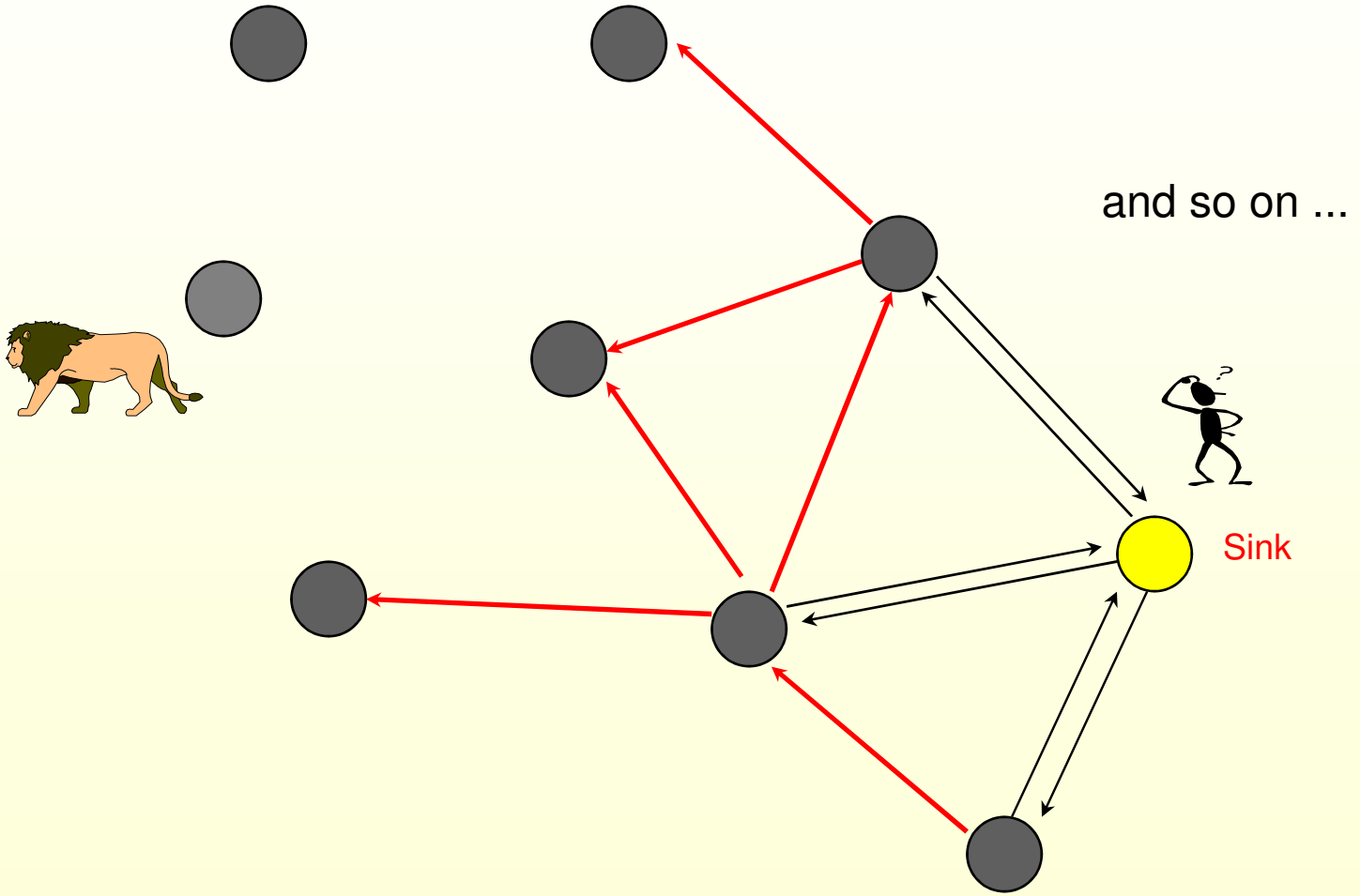
Illustrating Directed Diffusion



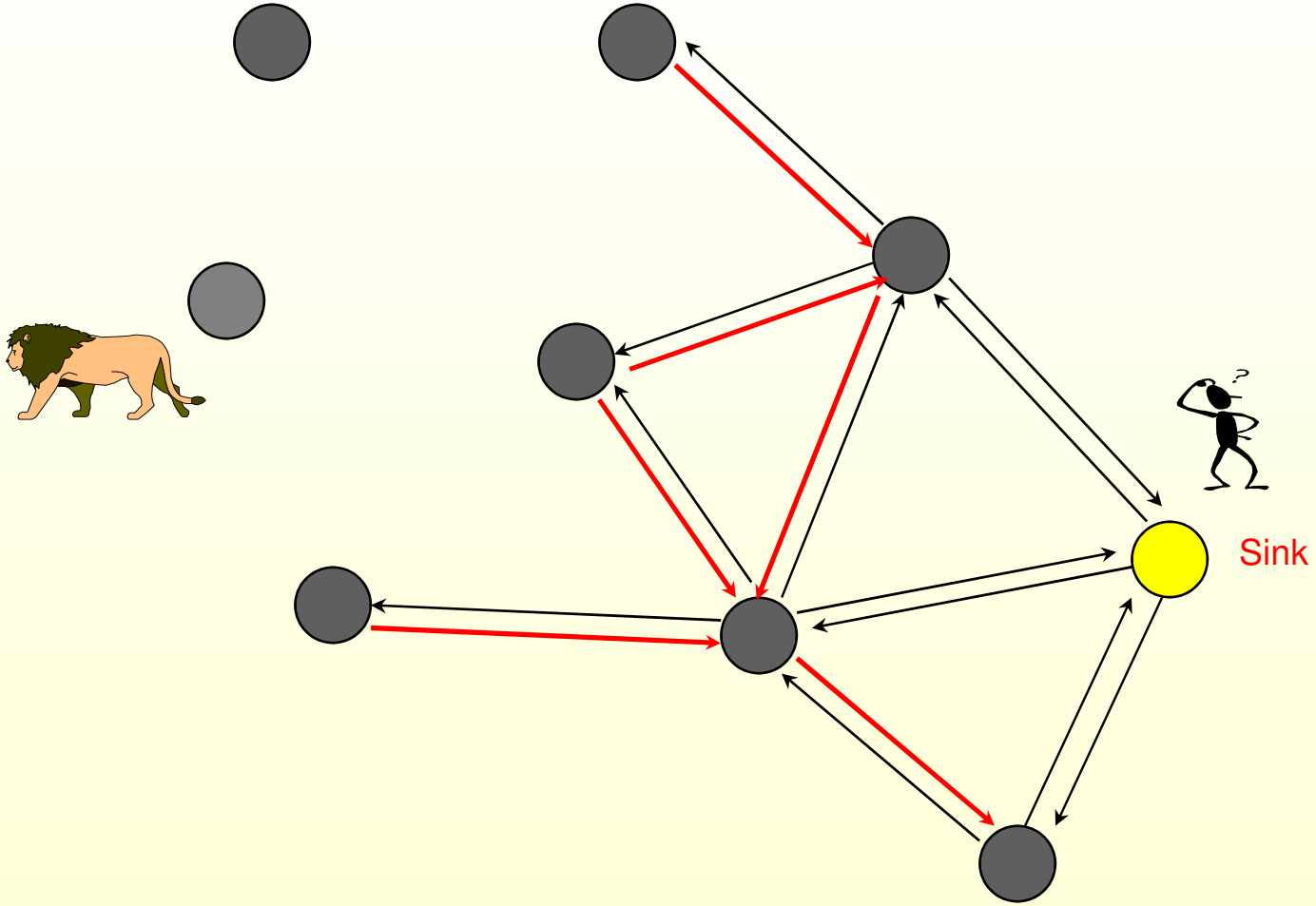
Illustrating Directed Diffusion



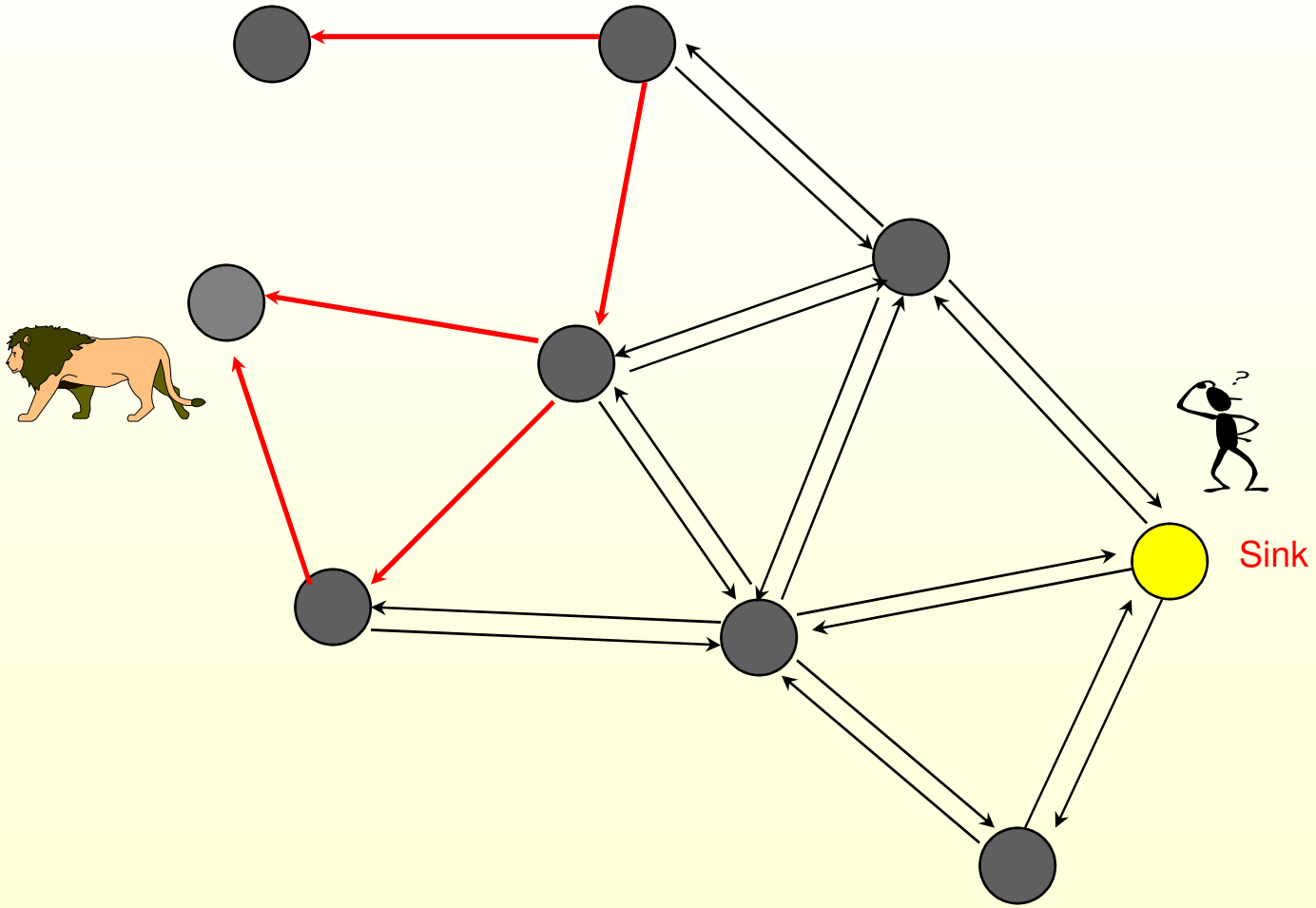
Illustrating Directed Diffusion



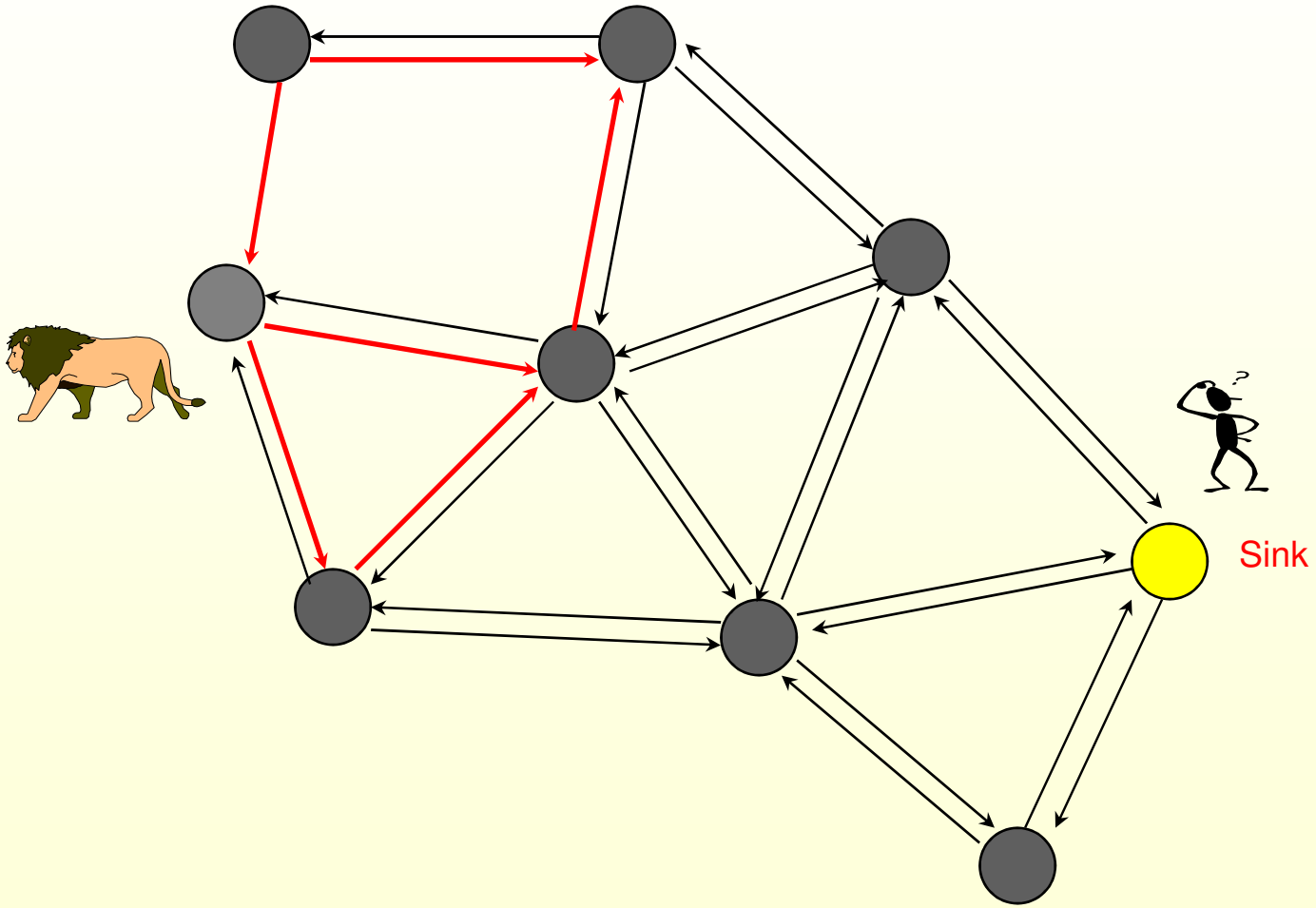
Illustrating Directed Diffusion



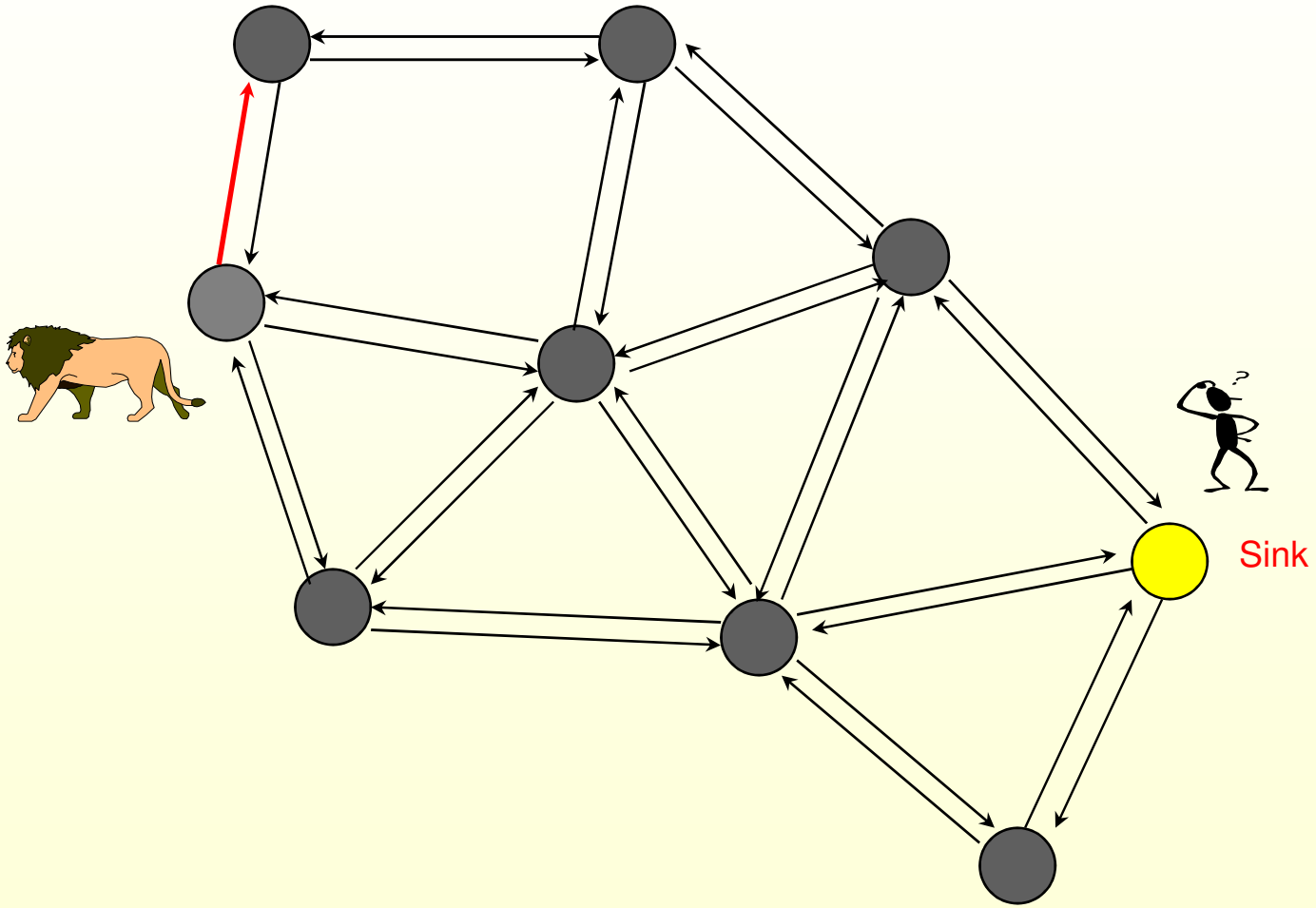
Illustrating Directed Diffusion



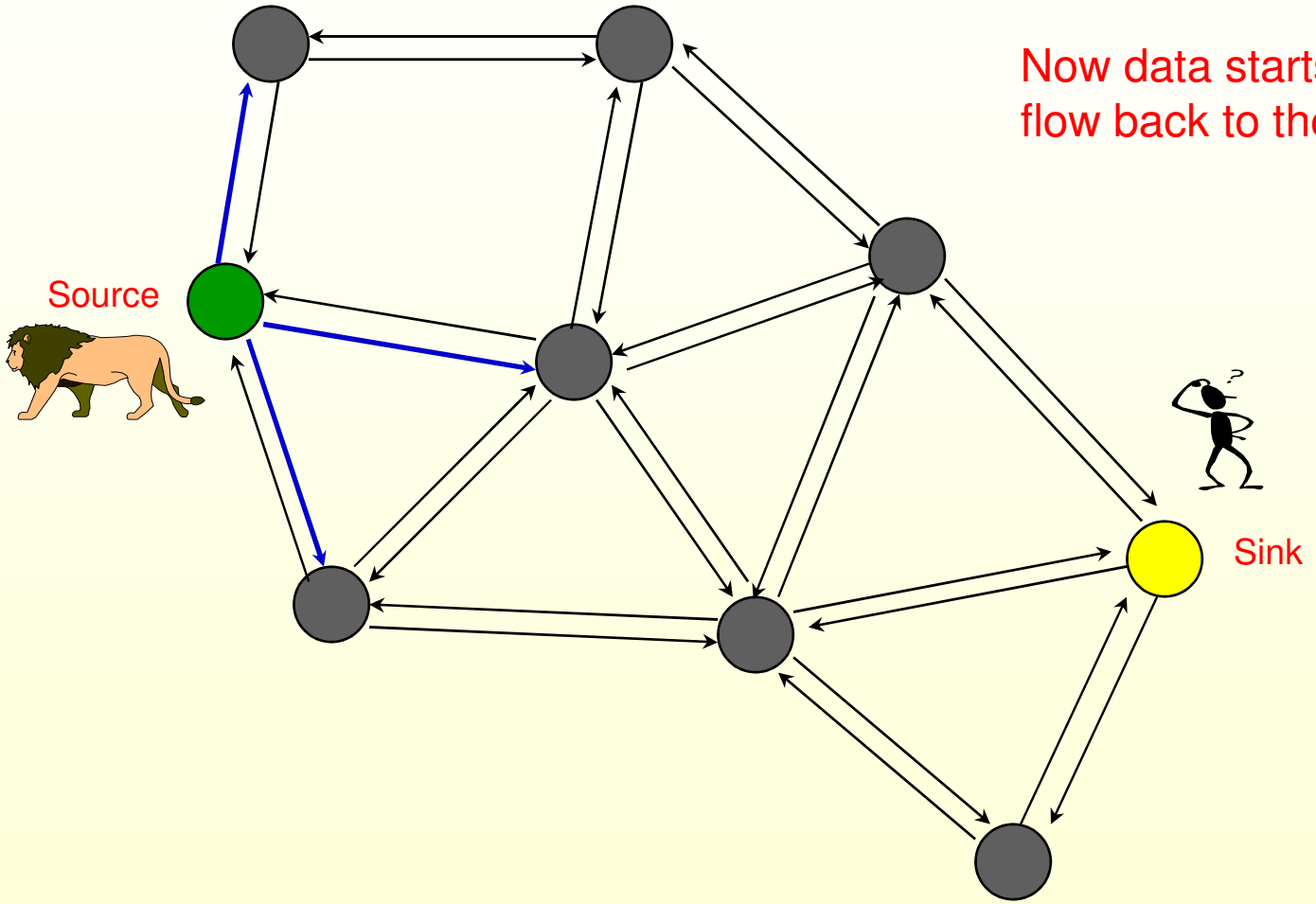
Illustrating Directed Diffusion



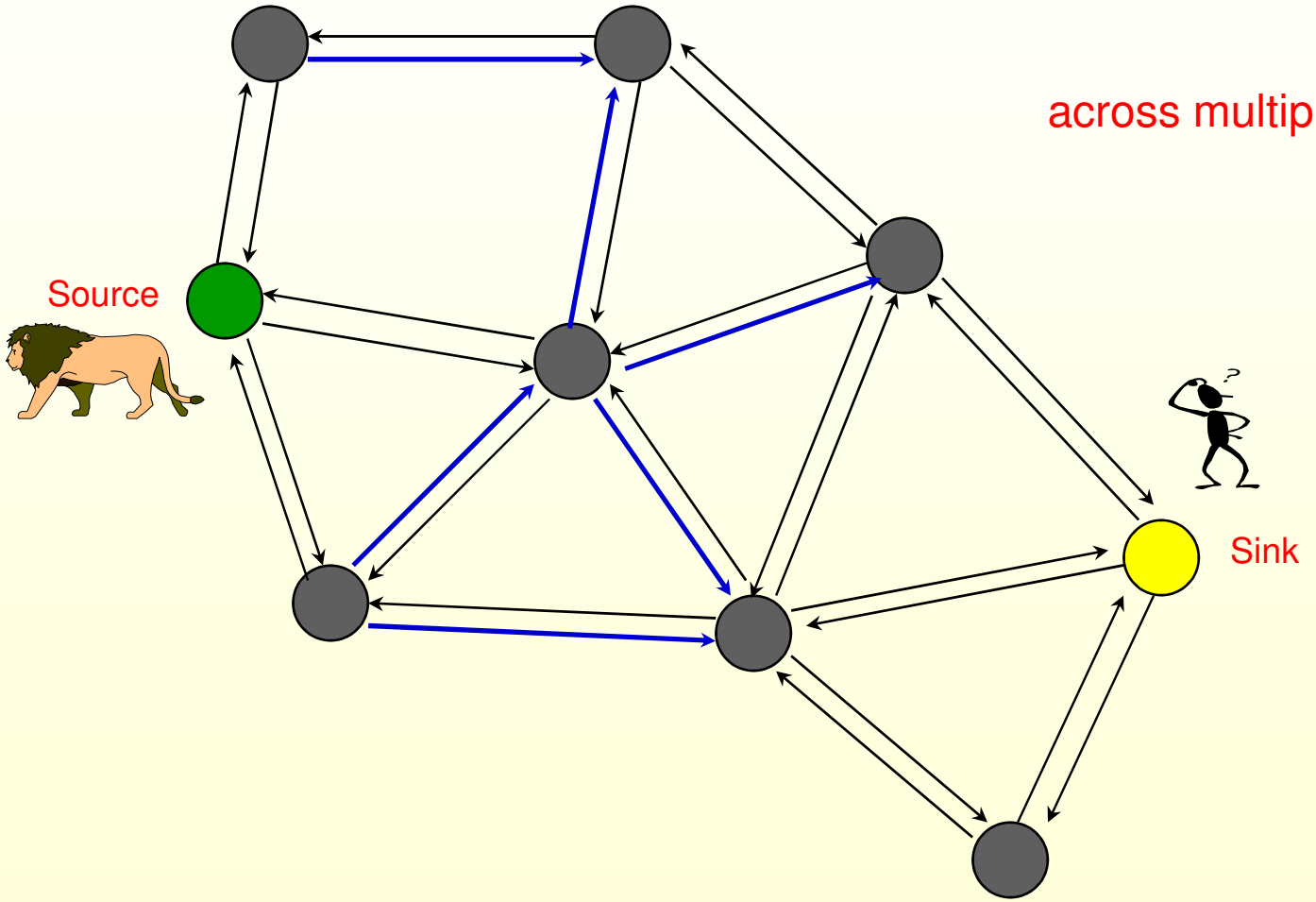
Illustrating Directed Diffusion



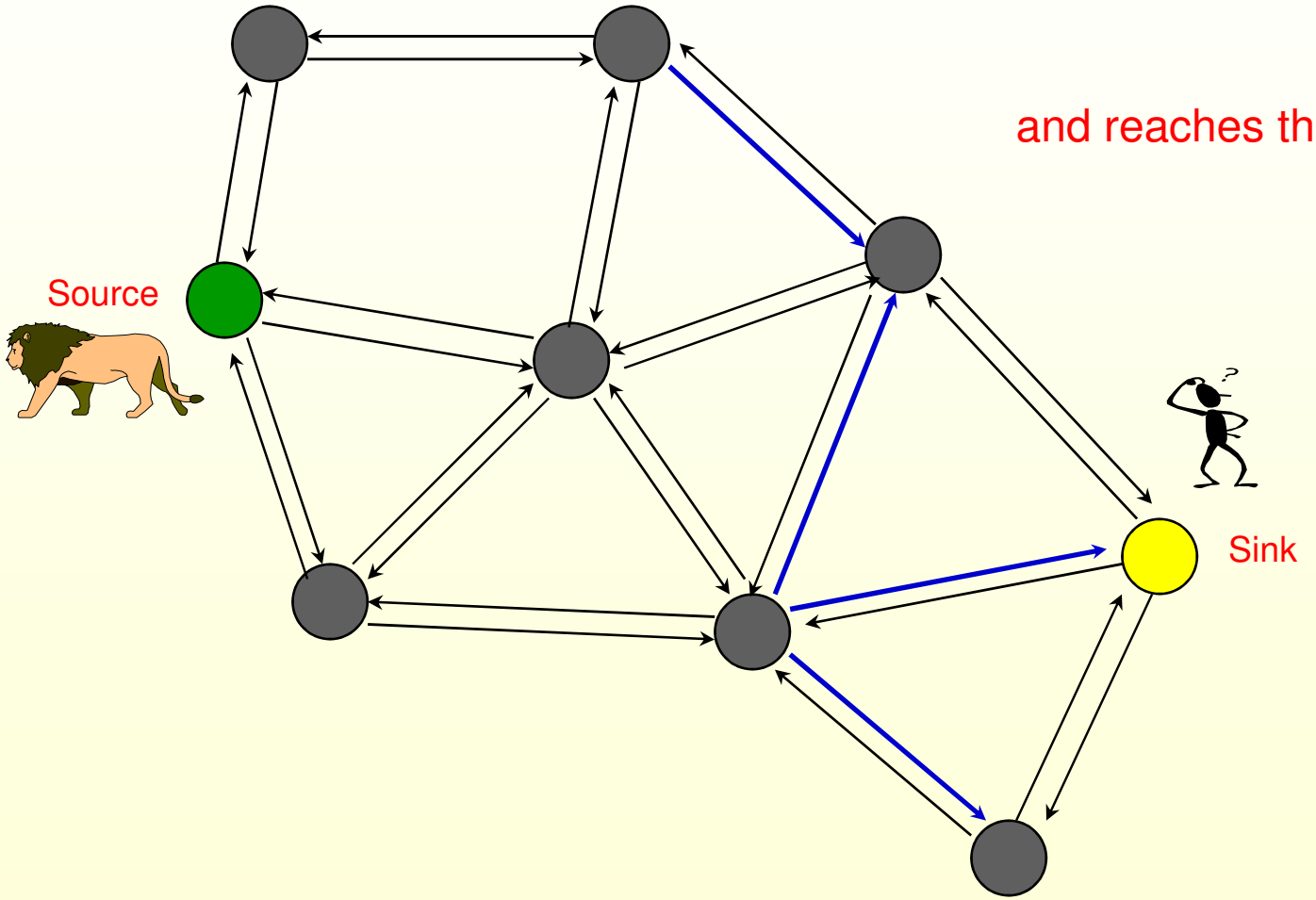
Illustrating Directed Diffusion



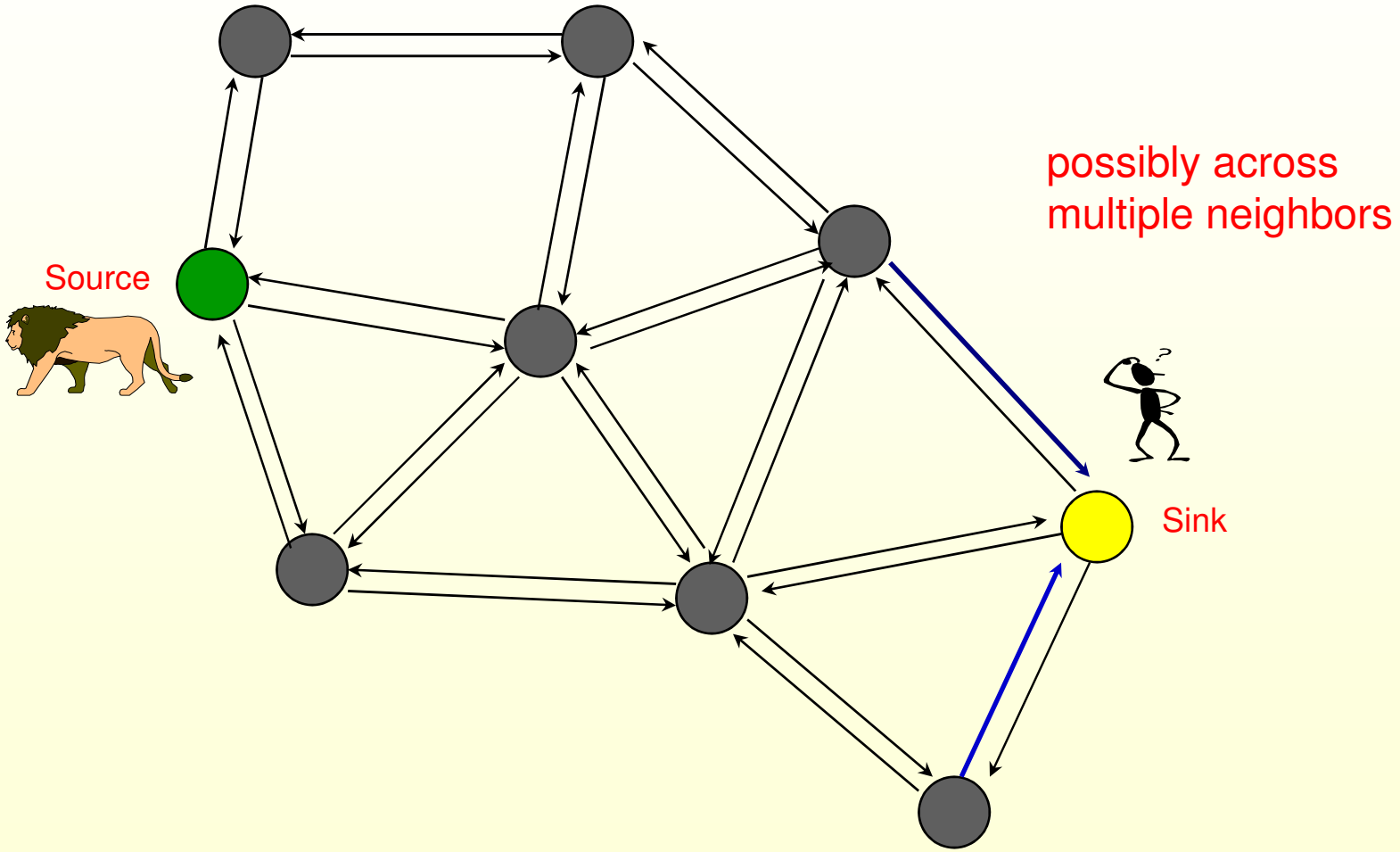
Illustrating Directed Diffusion



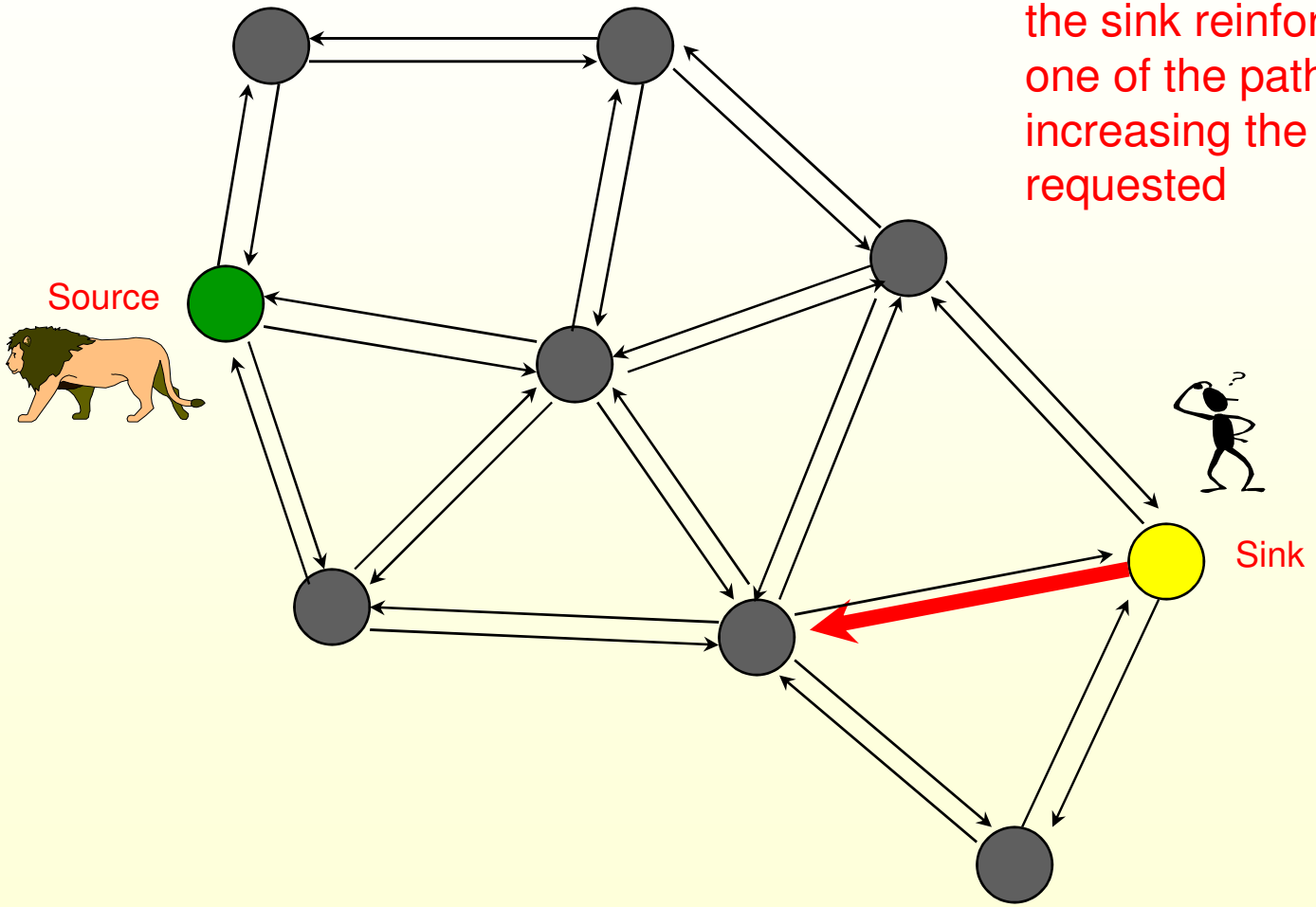
Illustrating Directed Diffusion



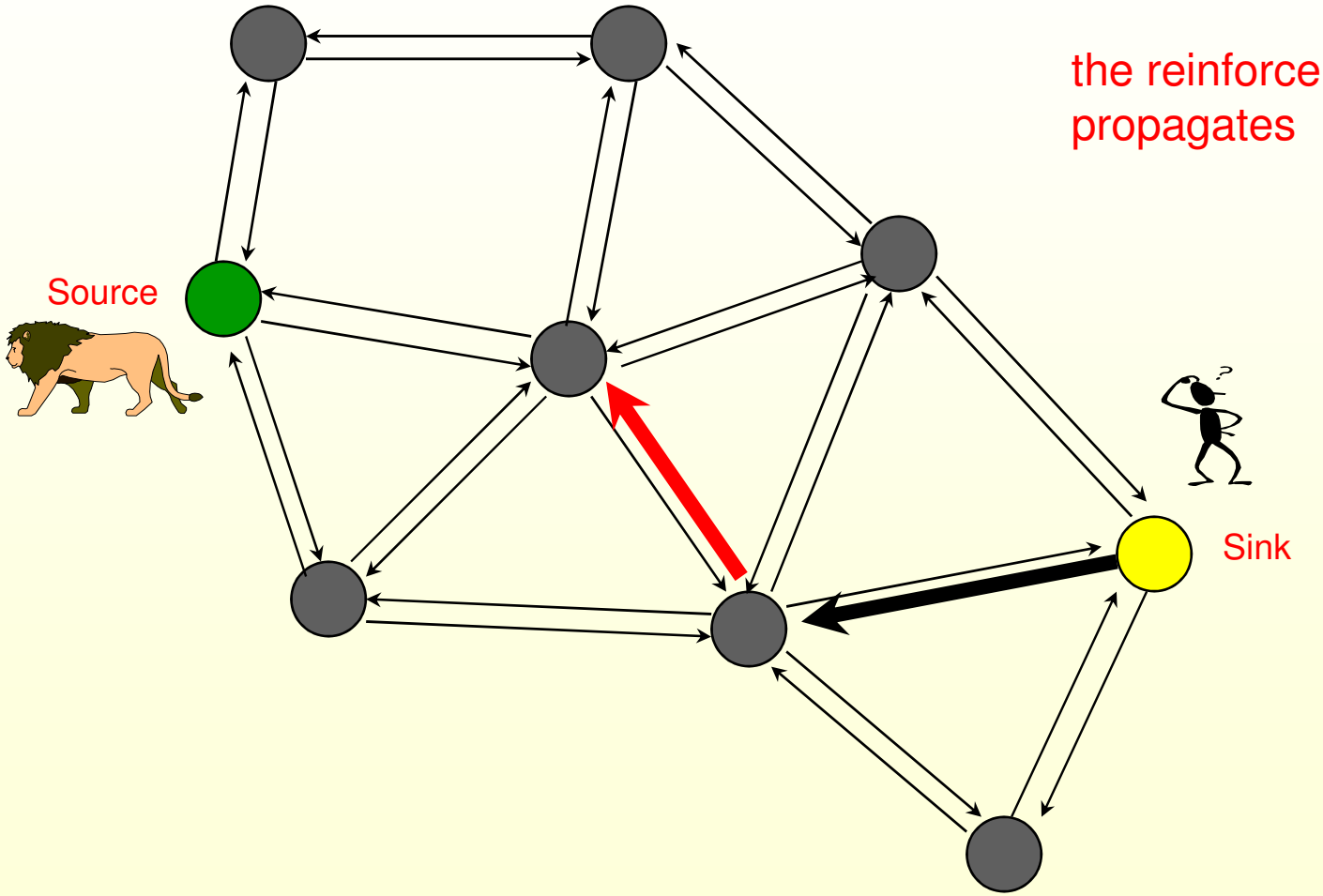
Illustrating Directed Diffusion



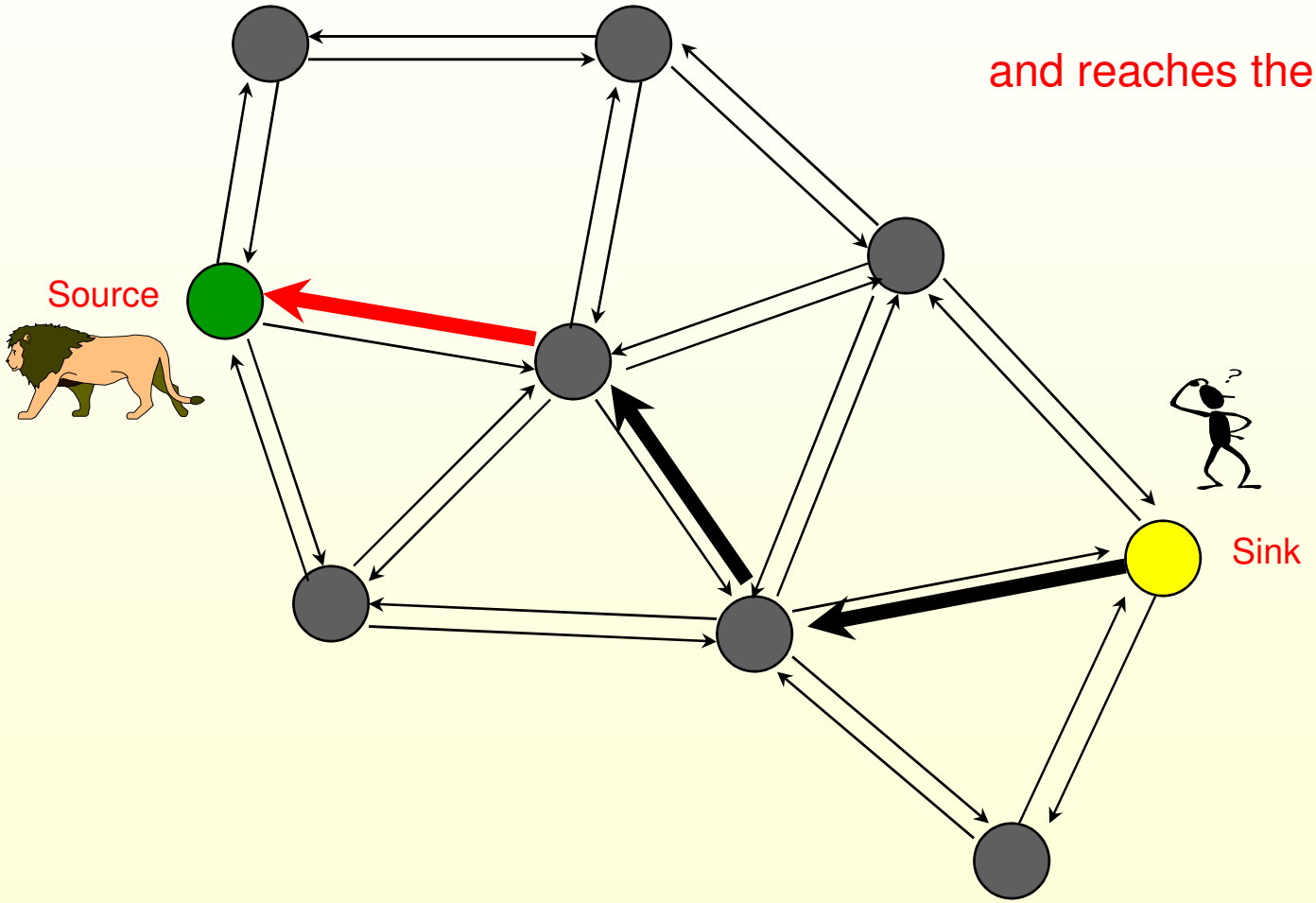
Illustrating Directed Diffusion



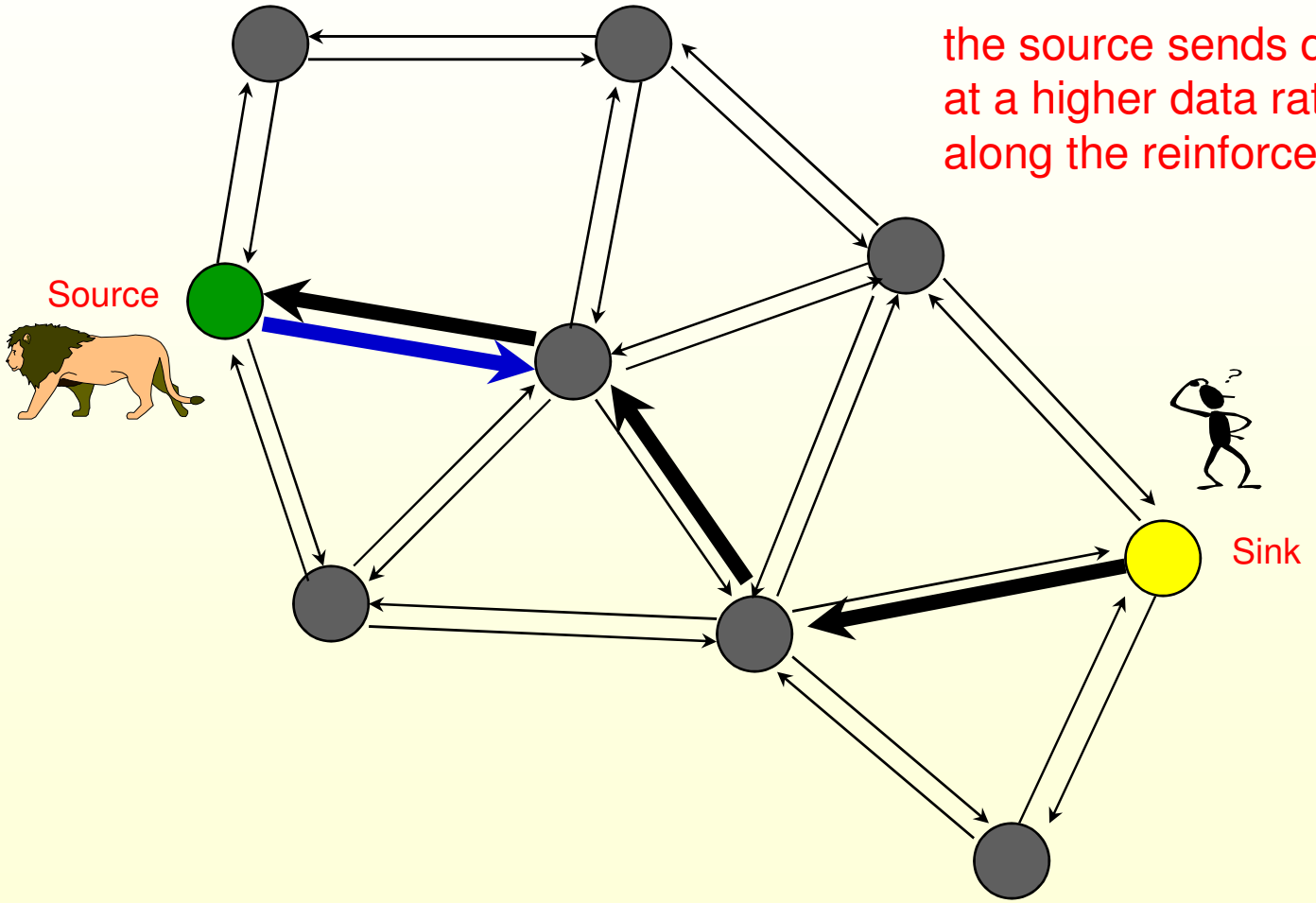
Illustrating Directed Diffusion



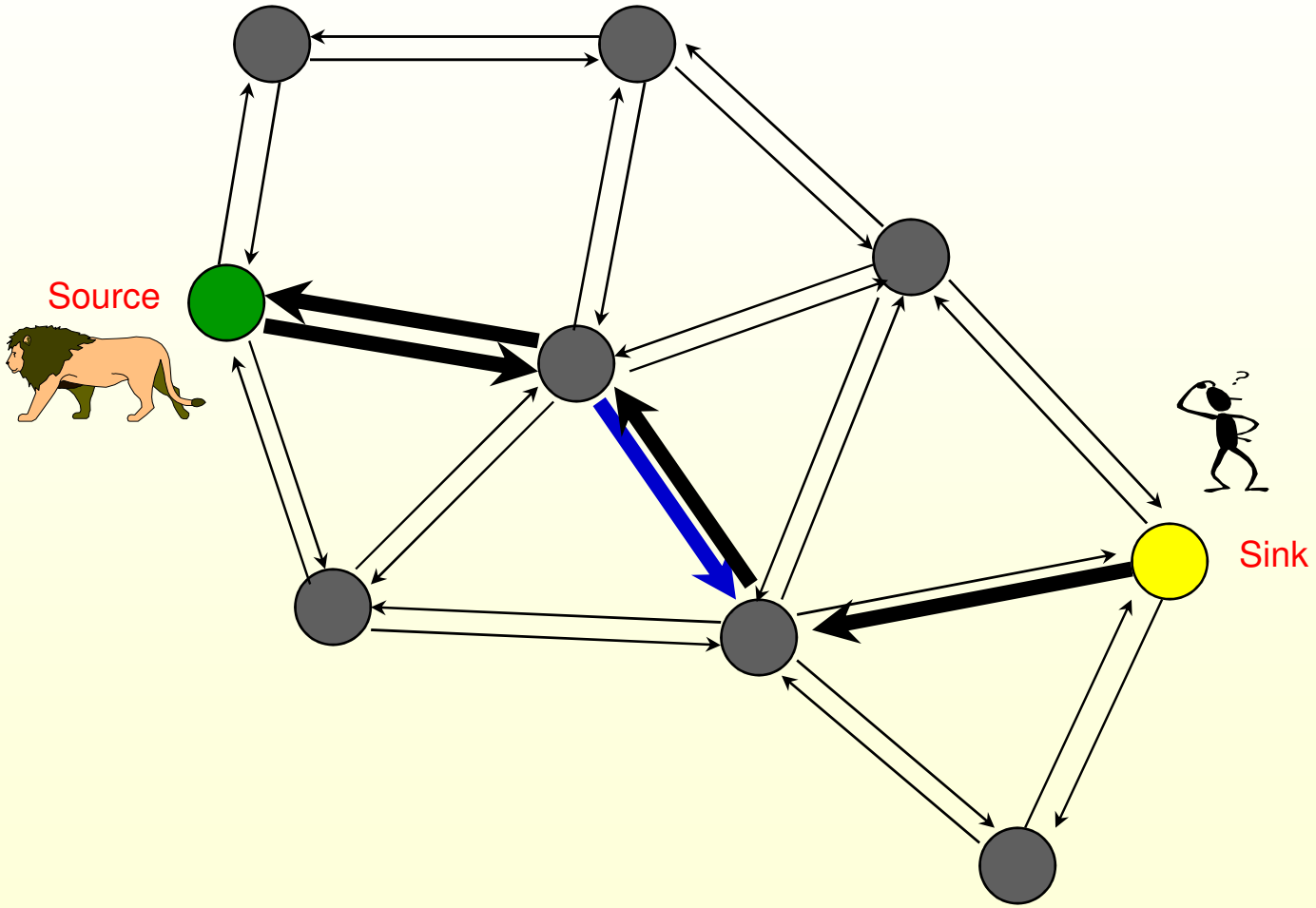
Illustrating Directed Diffusion



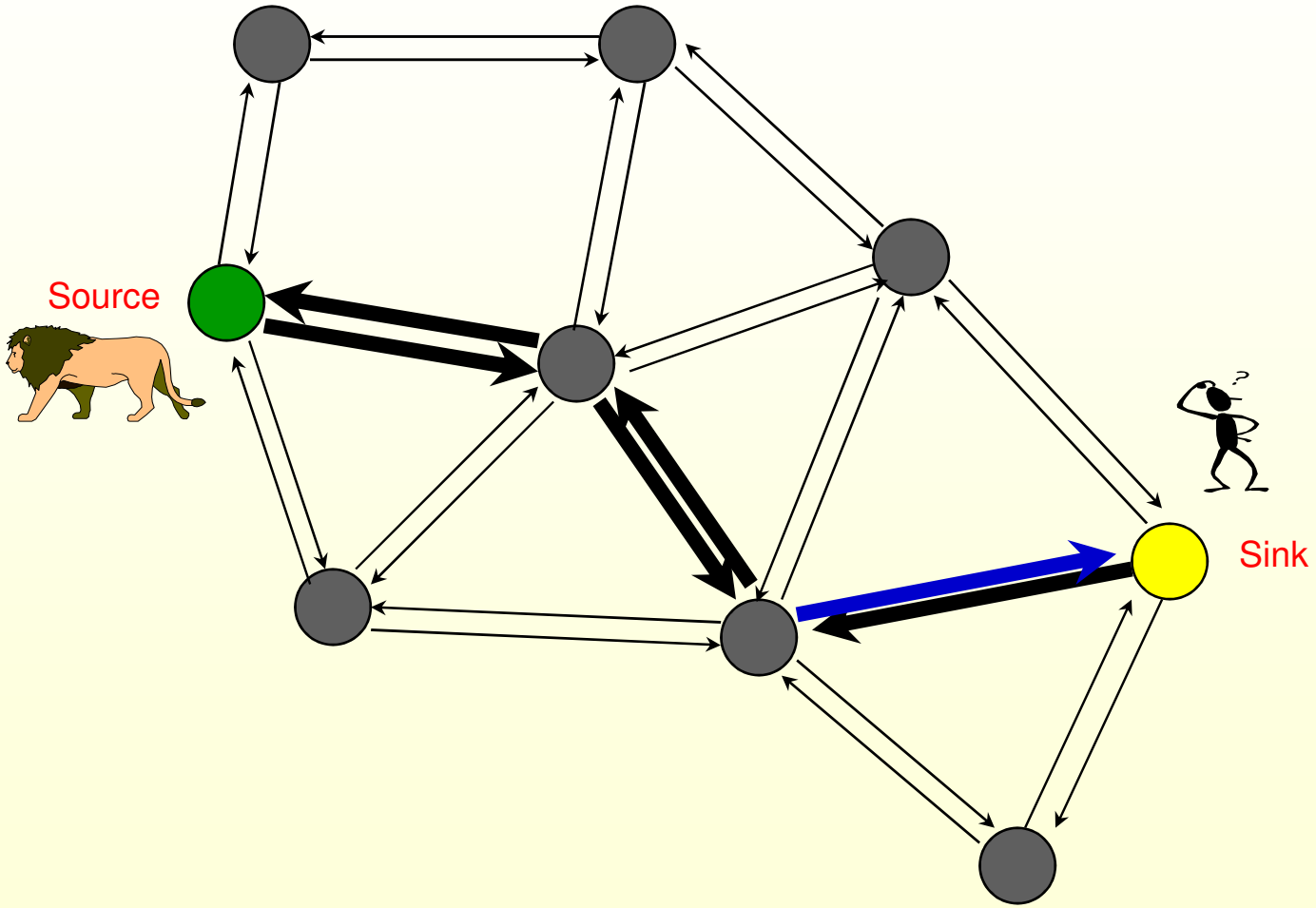
Illustrating Directed Diffusion



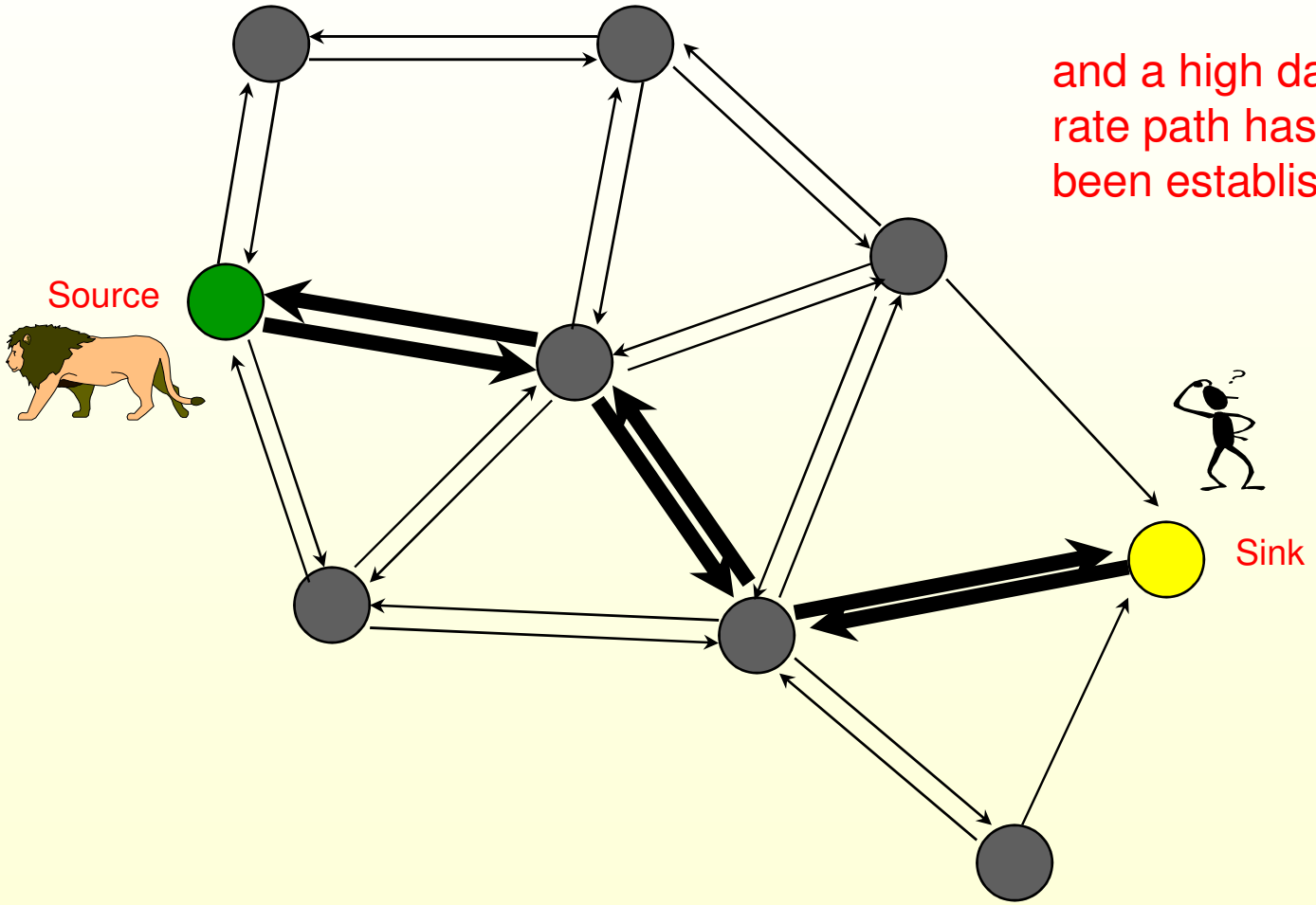
Illustrating Directed Diffusion



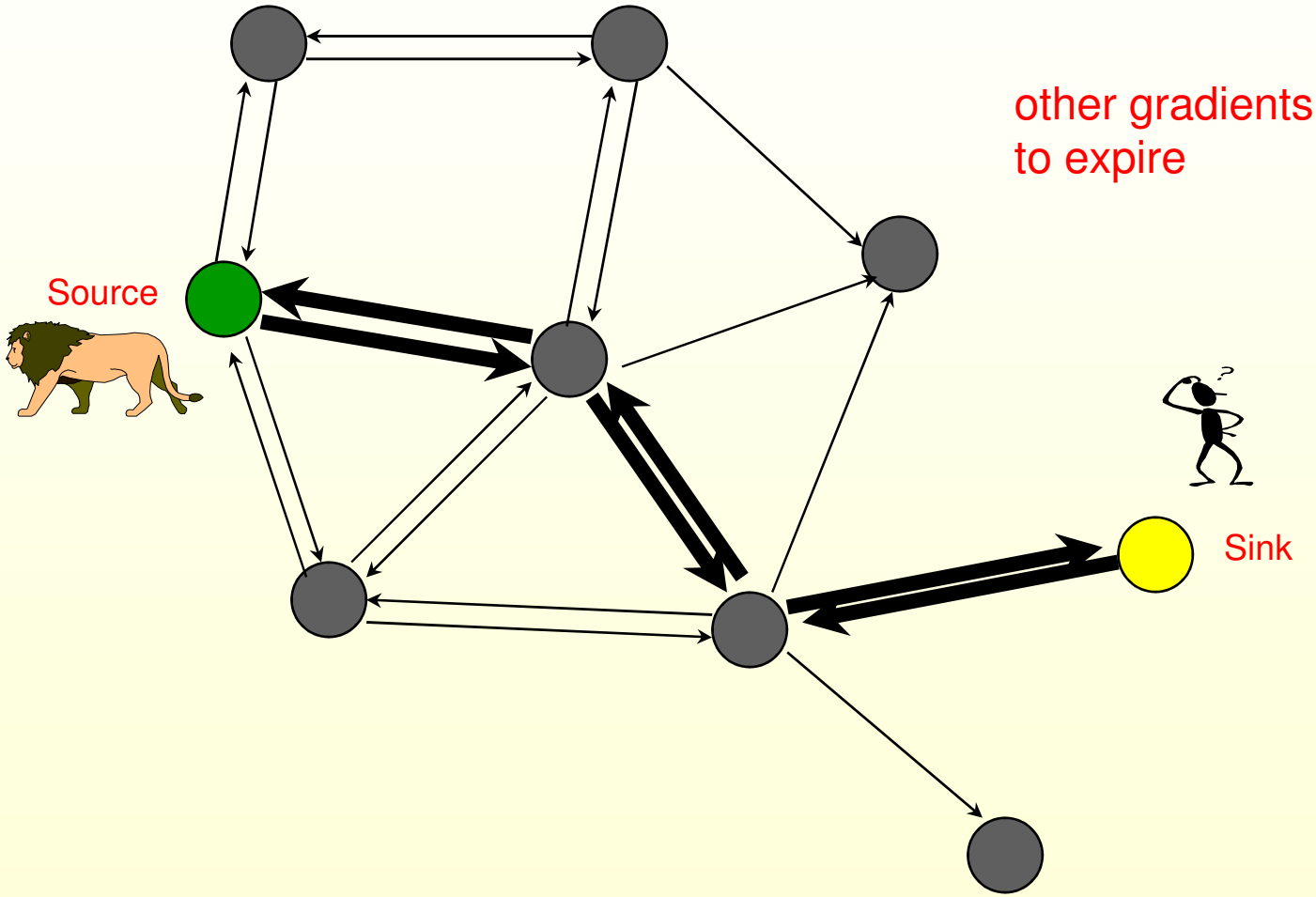
Illustrating Directed Diffusion



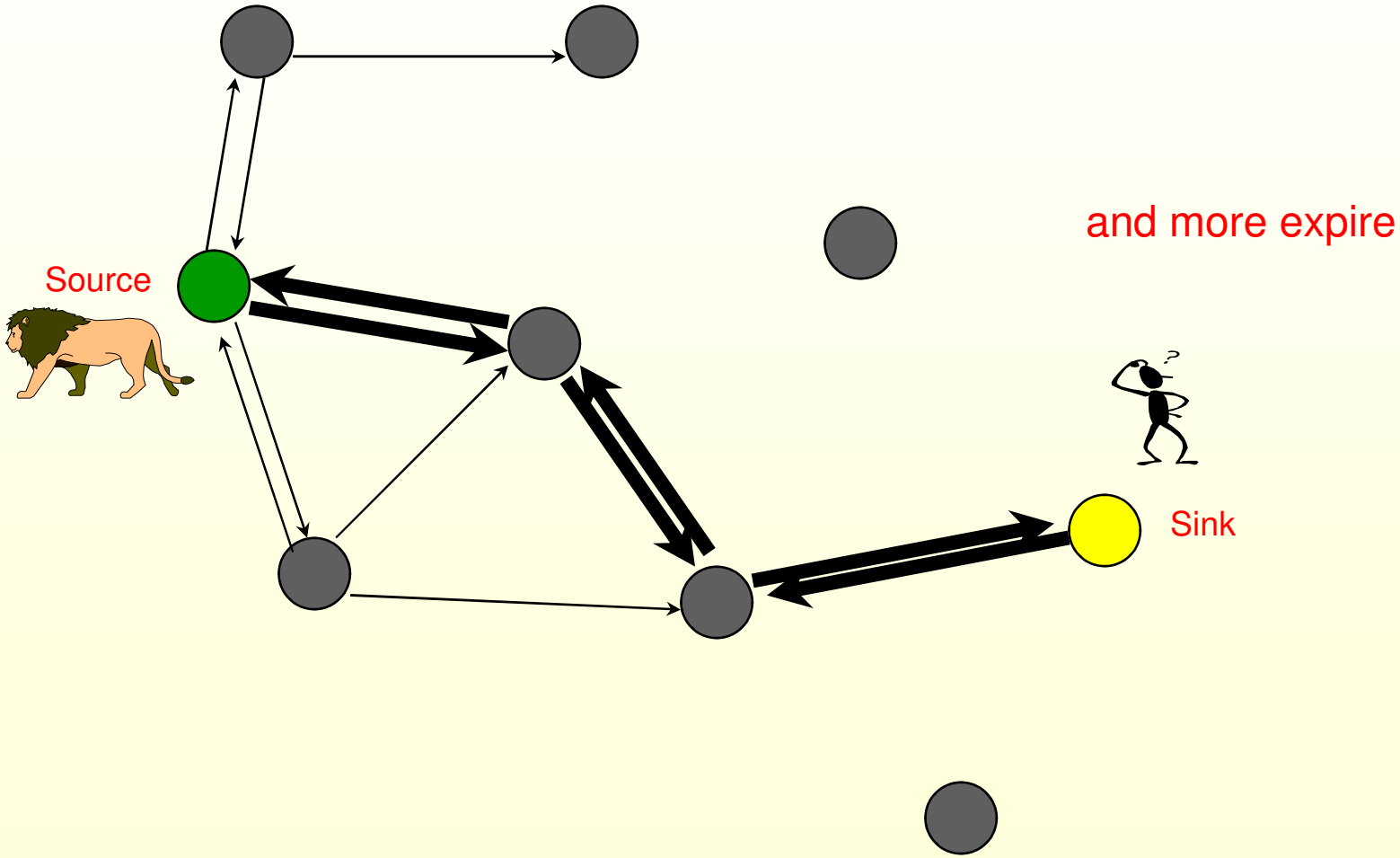
Illustrating Directed Diffusion



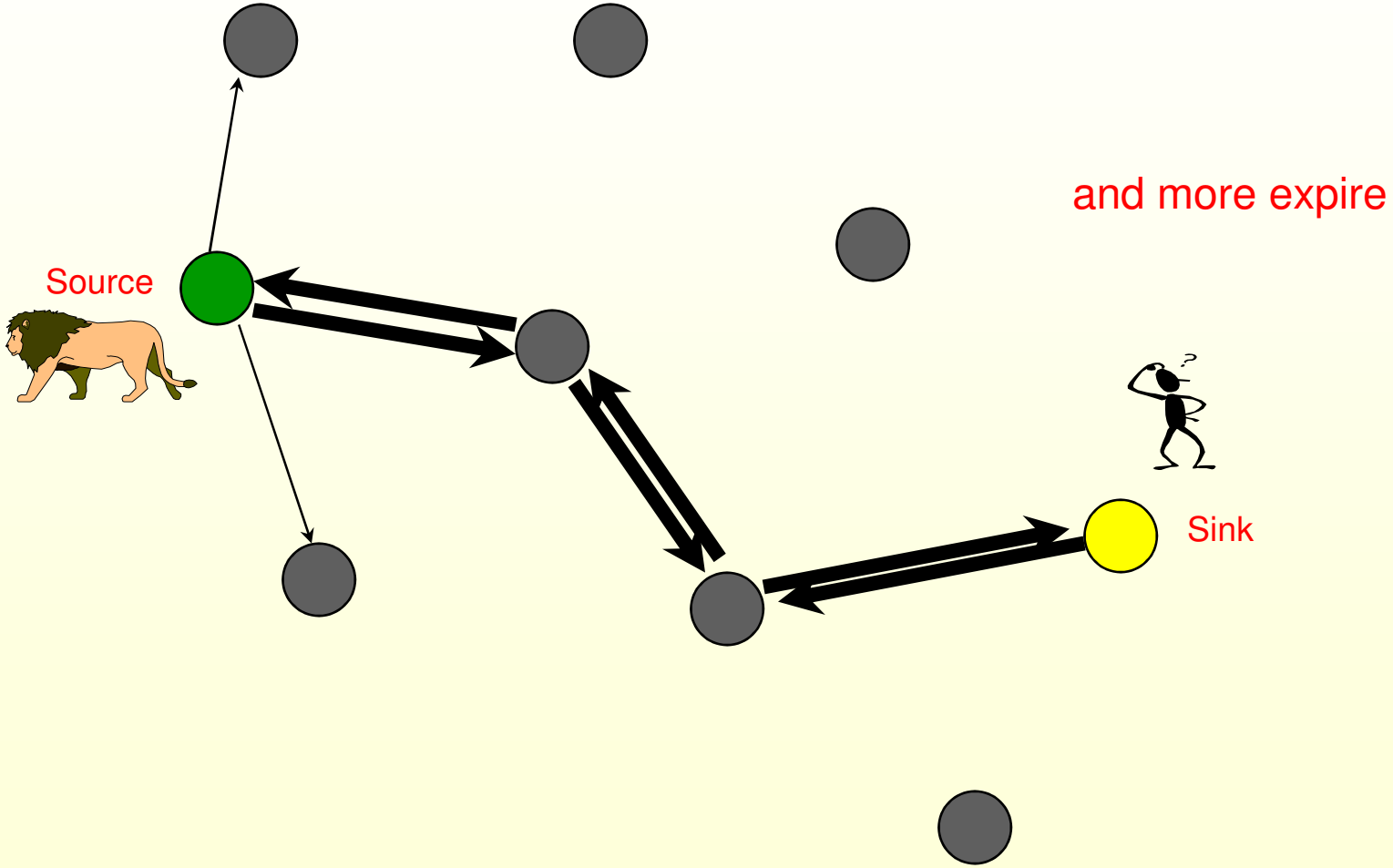
Illustrating Directed Diffusion



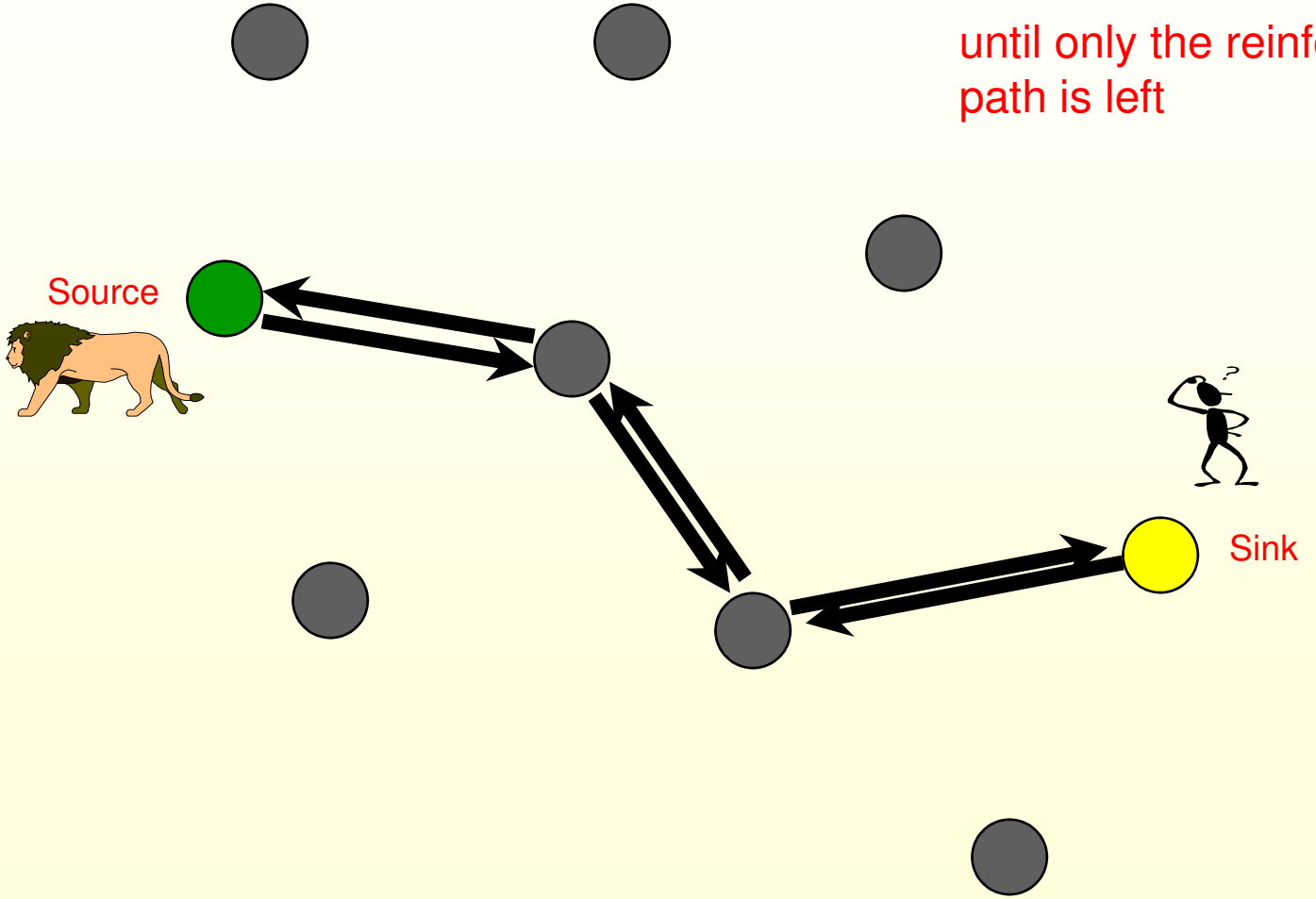
Illustrating Directed Diffusion



Illustrating Directed Diffusion

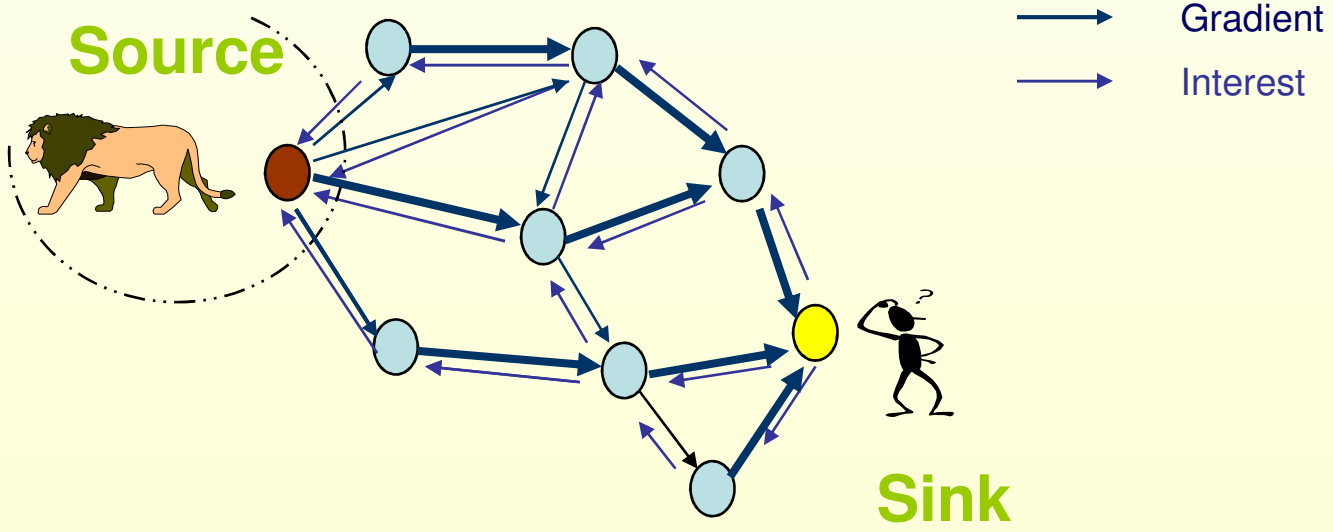


Illustrating Directed Diffusion



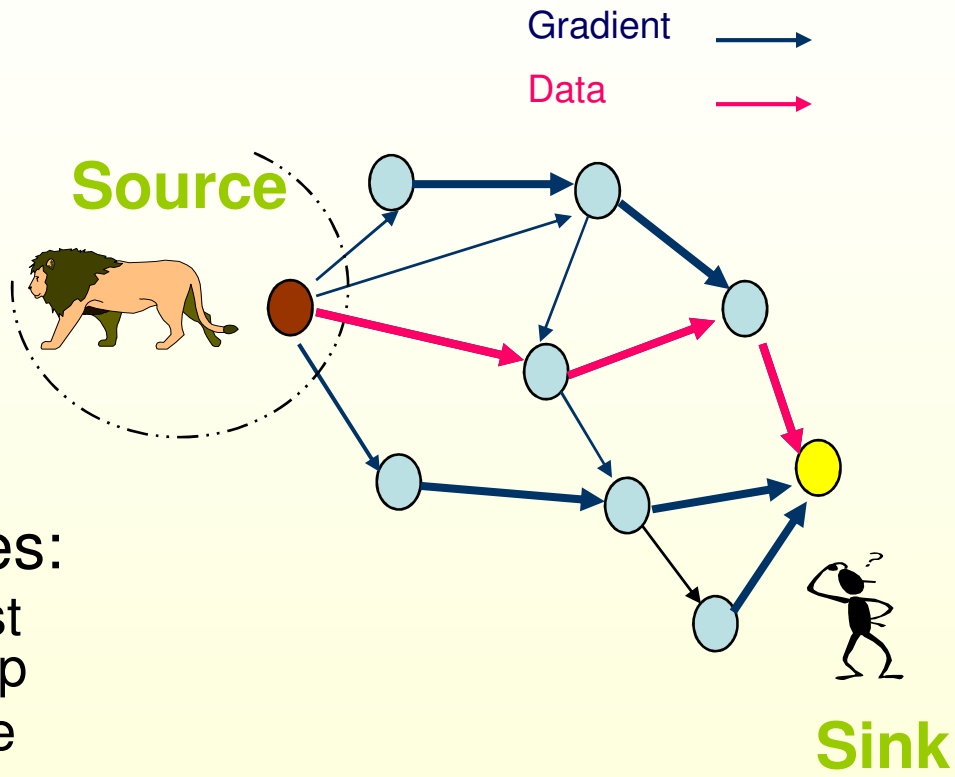
Interest Propagation

- Involves flooding the network
- Could use constrained or bidirectional flooding based on source location.
- Directional propagation can also be based on previously cached data.



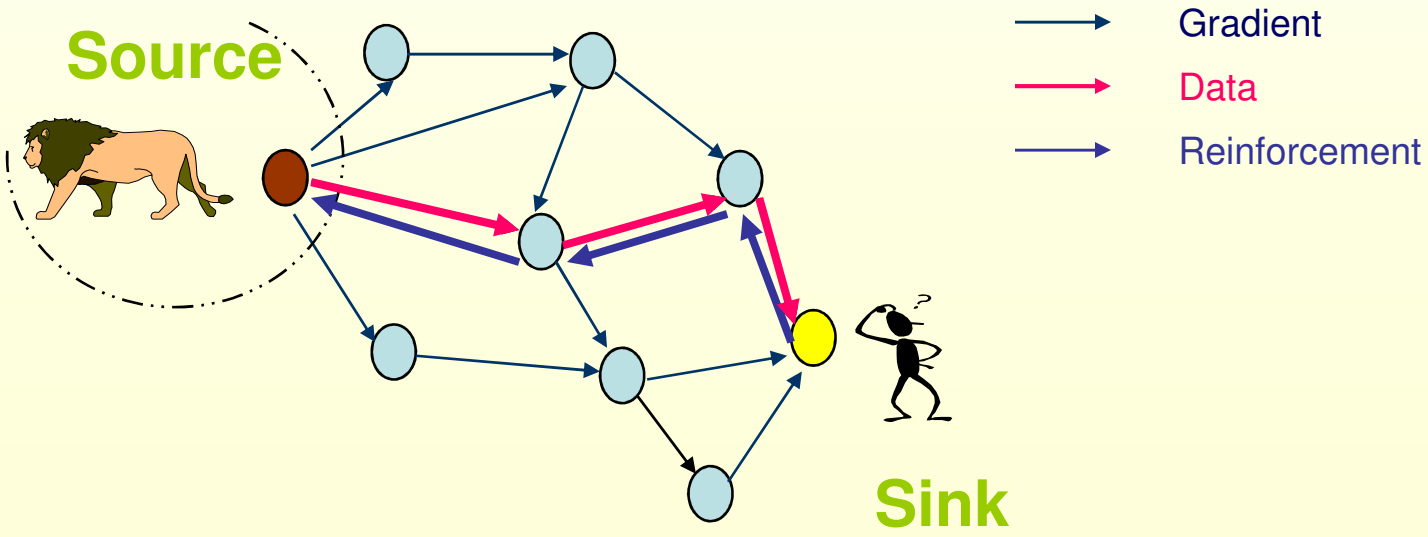
Data Propagation

- Source nodes match signature waveforms from codebook against observations
- Nodes match data against interest cache, compute highest event-rate request from all gradients, and (re)-sample events at this rate
- Intermediate receiving nodes:
  - Find matching entry in interest cache; no match → silent drop
  - Check and update data cache (loop prevention, aggregation, duplicate suppression, etc.)
  - Retrieve all gradients, and resend message, performing frequency conversion if necessary



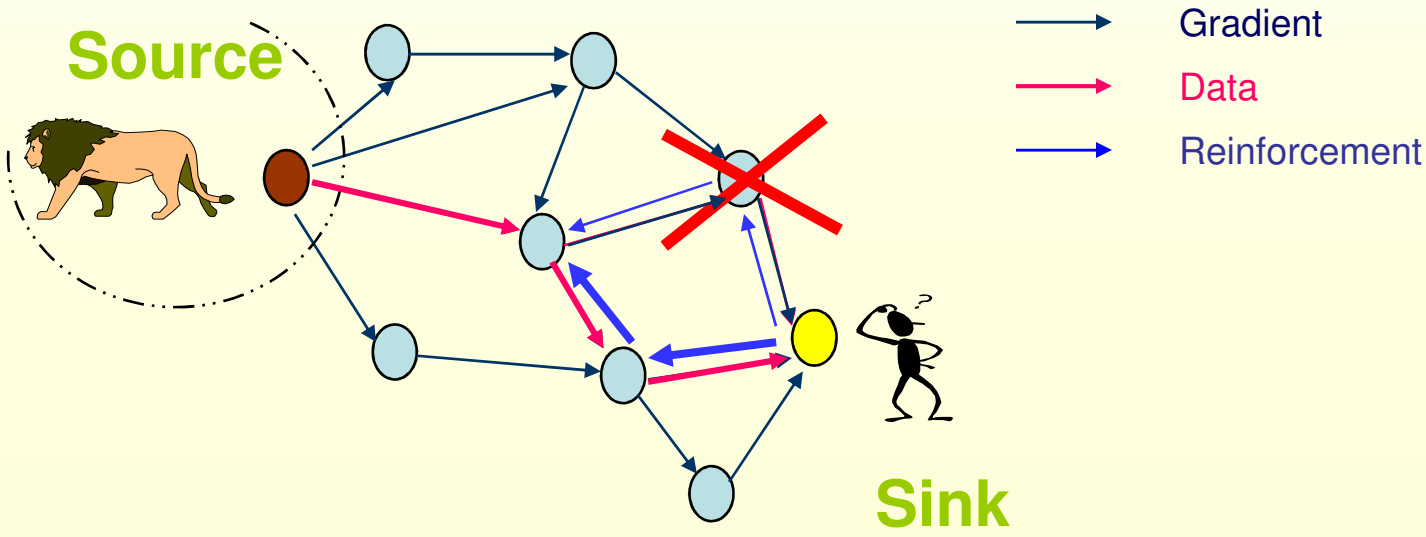
Reinforcement

- Reinforce one of the neighbor after receiving initial data.
  - Neighbor(s) from whom new events are received.
  - Neighbor who is consistently performing better than others.
  - Neighbor from whom most events are received.

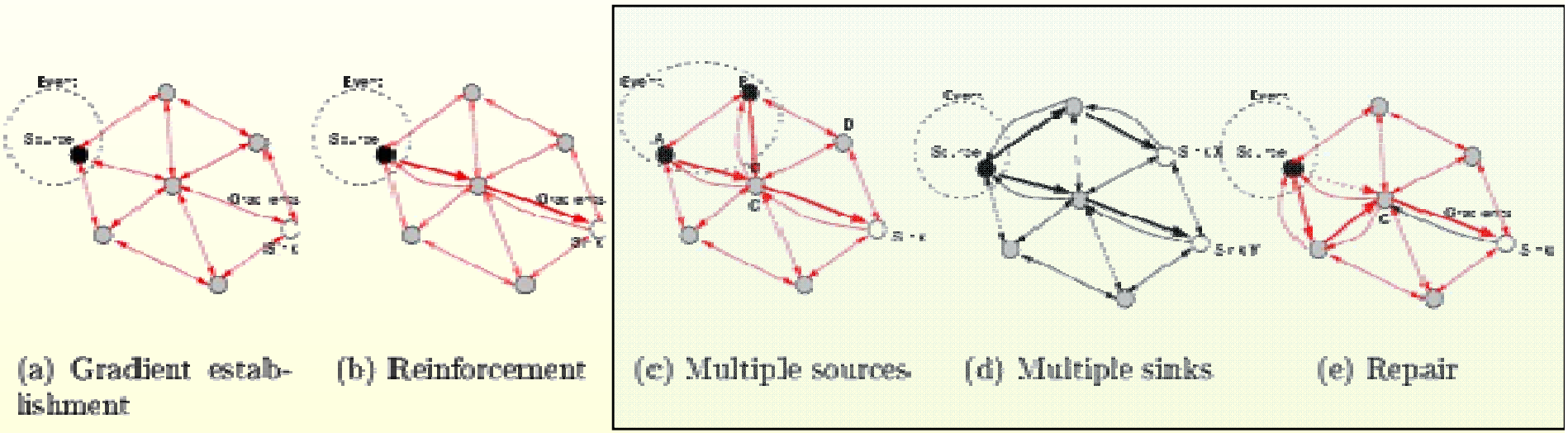


Negative Reinforcement

- Explicitly degrade some paths by re-sending *interests* with lower data request rates.
- Cache entries time out if not reinforced



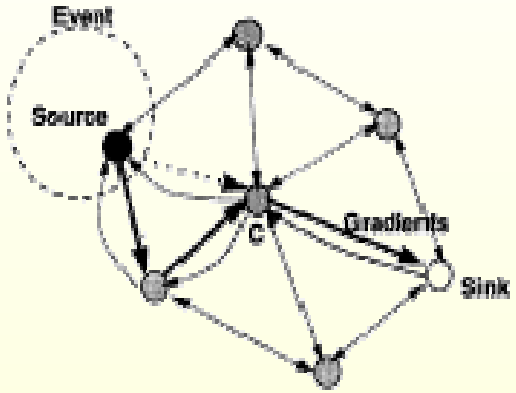
Other Aspects of Directed Diffusion



(a) Gradient establishment (b) Reinforcement (c) Multiple sources (d) Multiple sinks (e) Repair

Figure 2: Illustrating different aspects of diffusion.

Local Repair for Failed Paths



- Intermediate nodes on a previously reinforced path can apply reinforcement rules (useful for failed or degraded paths)
- C detects degradation
  - By noticing that the event reporting rate from its upstream neighbor (source) is now lower
  - By realizing that other neighbors have been transmitting previously unseen location estimates.
- C applies reinforcement rules
- Problem: wasted resources (e.g., if other downstream nodes also do the same)
- Avoid this by interpolating location estimates from the events

DD Adaptation Summary

- Reinforcement (optimization):
  - Data-driven rules; ex., new msg. from neighbor → resend original with smaller sampling interval
  - This neighbor, in turn, reinforces upstream nodes
  - Local rule : minimize delay; other rules are possible
  - Passive negative reinforcement (timeouts) or active (negative weights)
- Multiple sources + reinforcement
  - Works in some cases, open for further exploration
- Multiple sinks: Exploit prior setup (i.e., use cache)
- Intermediate nodes use reinforcement for local repair
  - Cascading reinforcement discoveries from upstream can be a problem; one soln.: interpolate requests to preserve status-quo

Local Behavior Choices

1. For propagating interests
   - In our example, flood
   - More sophisticated behaviors possible: e.g. based on cached information, GPS
2. For setting up gradients
   - Highest gradient towards neighbor from whom we first heard interest
   - Others possible: towards neighbor with highest energy
3. For data transmission
   - Different local rules can result in single path delivery, striped multi-path delivery, single source to multiple sinks and so on.
4. For reinforcement
   - reinforce one path, or part thereof, based on observed losses, delay variances etc.
   - other variants: inhibit certain paths because resource levels are low

DD Design Space

Diffusion element	Design Choices
Interest Propagation	• Flooding • Constrained or directional flooding based on location • Directional propagation based on previously cached data
Data Propagation	• Reinforcement to single path delivery • Multipath delivery with selective quality along different paths • Multipath delivery with probabilistic forwarding
Data caching and aggregation	• For robust data delivery in the face of node failure • For coordinated sensing and data reduction • For directing interests
Reinforcement	• Rules for deciding when to reinforce • Rules for how many neighbors to reinforce • Negative reinforcement mechanisms and rules

Figure 3: Design Space for Diffusion

Initial Simulation Study of Diffusion

- Key metric
  - Average Dissipated Energy per event delivered
    - captures energy efficiency and network lifetime
- Compare directed diffusion to
  - flooding
  - centrally computed dissemination tree (omniscient multicast)

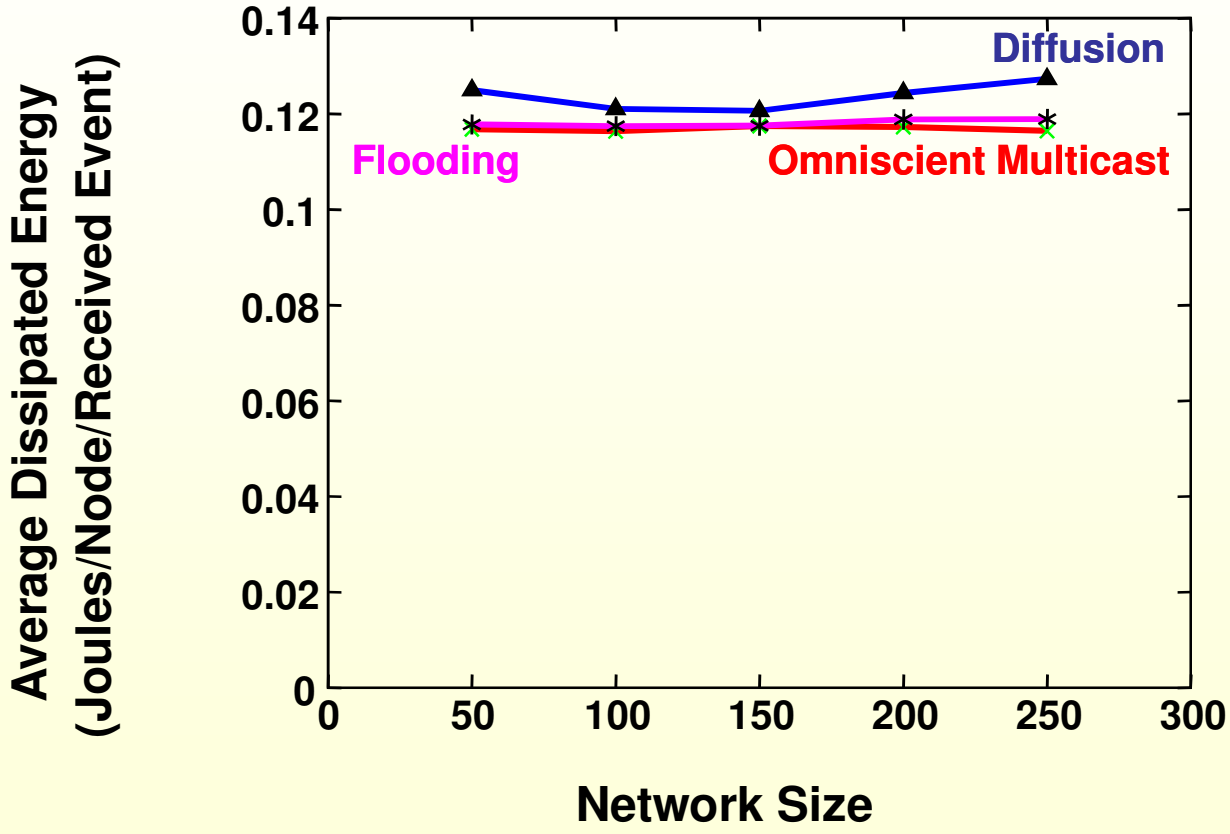
Diffusion Simulation Details

- Simulator: *ns-2*
- Network Size: 50-250 Nodes
- Node Transmission Range: 40m
- Constant Density: 1.95×10^{-3} nodes/m^2 (9.8 nodes in radius)
- MAC: Modified contention-based MAC
- Energy Model: Mimic a realistic sensor radio [Pottie 2000]
  - 660 mW in transmission, 395 mW in reception, and 35 mw in idle mode

Diffusion Simulation

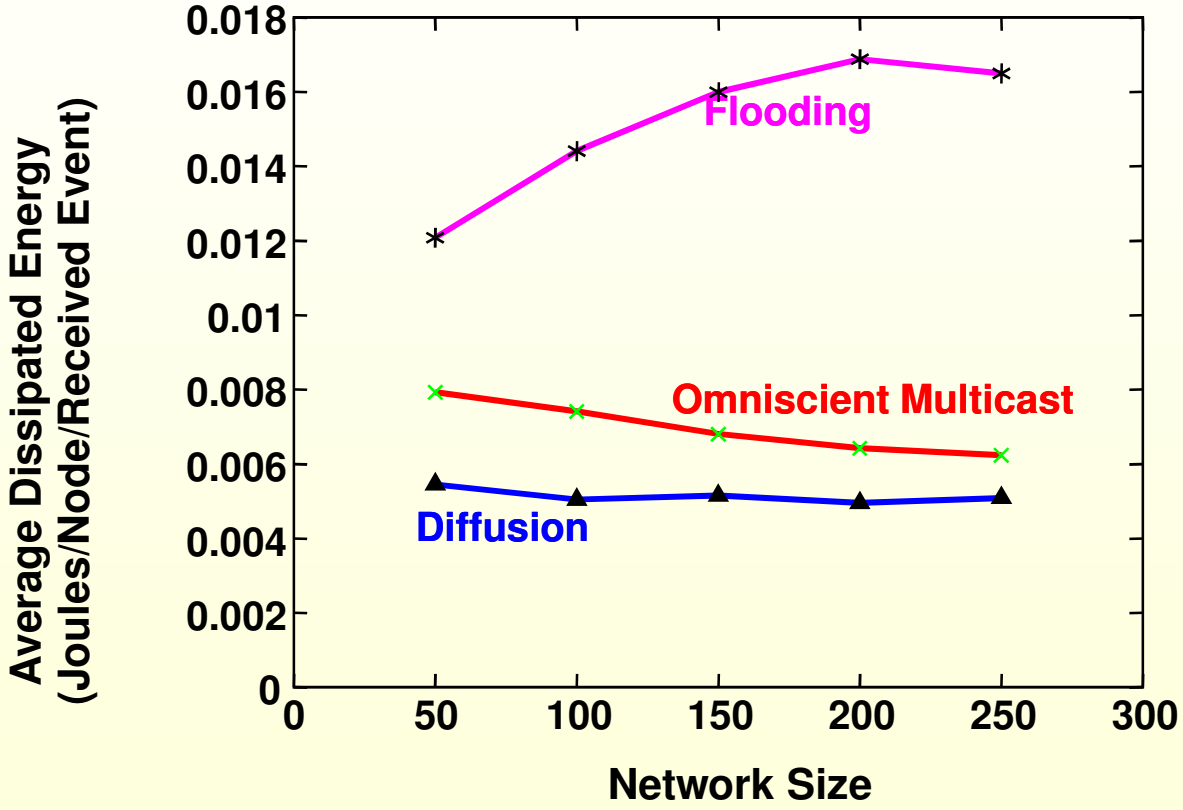
- Surveillance application
  - 5 sources are randomly selected within a 70m x 70m square field
  - 5 sinks are randomly selected across the field
  - High data rate is 2 events/sec
  - Low data rate is 0.02 events/sec
  - Event size: 64 bytes
  - Interest size: 36 bytes
  - All sources send the same location estimate for base experiments

Average Dissipated Energy (**Standard 802.11** energy model)



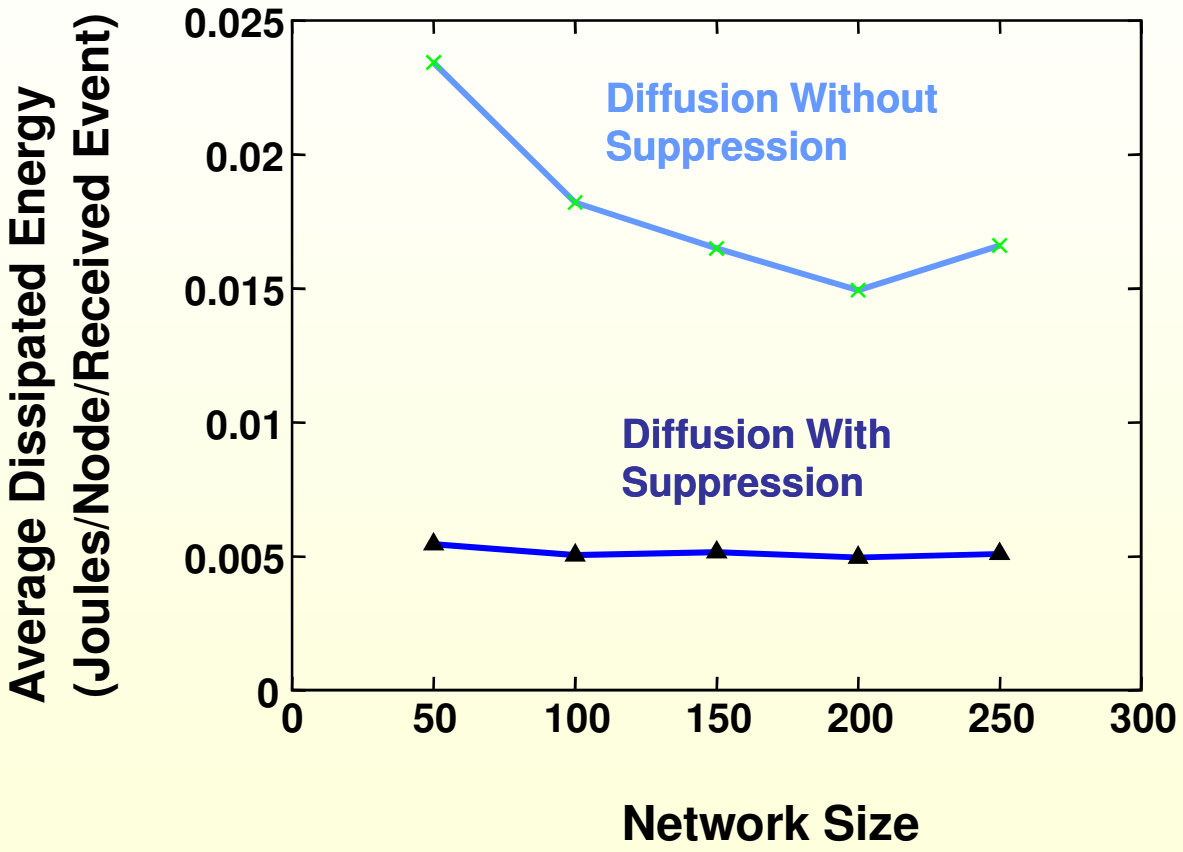
Standard 802.11 is dominated by idle energy

Average Dissipated Energy (**Sensor radio** energy model)



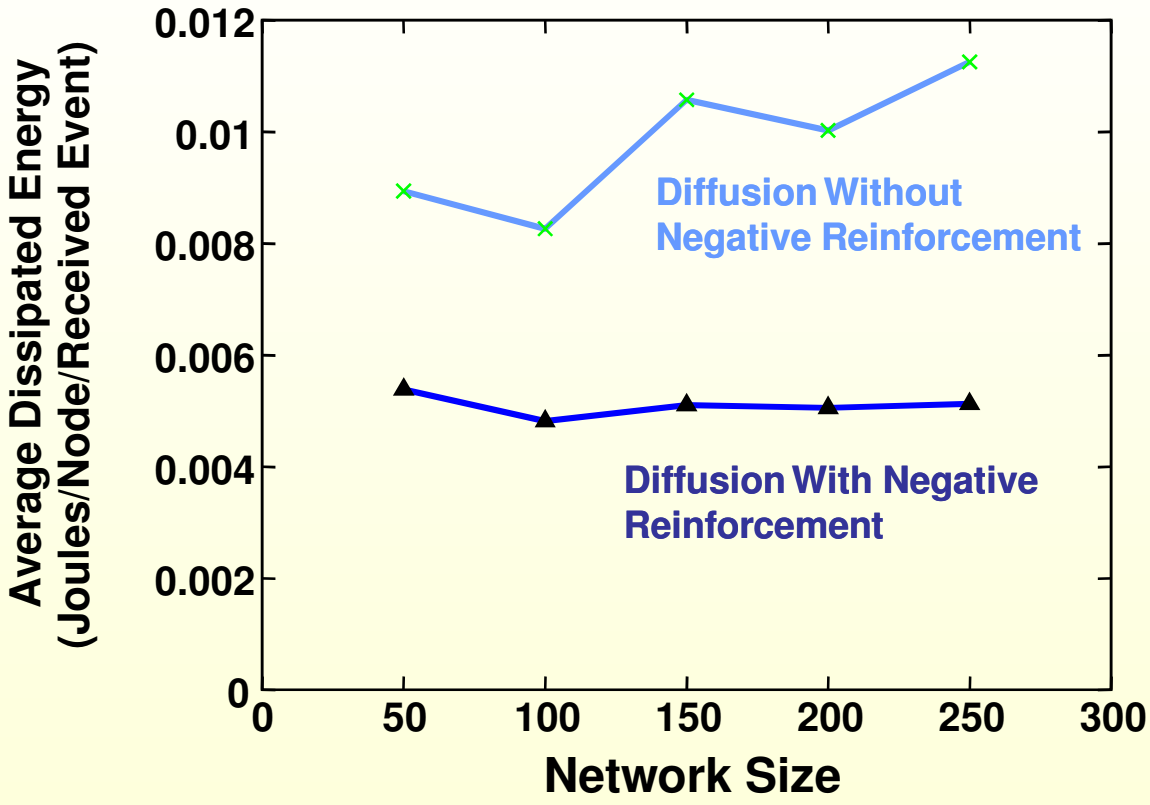
Diffusion can outperform flooding and even omniscient multicast. Why ?

Impact of In-Network Processing



Application-level duplicate suppression allows diffusion to reduce traffic and to surpass omniscient multicast.

Impact of Negative Reinforcement



Reducing high-rate paths in steady state is critical

Summary of Diffusion Results

- Under the investigated scenarios, diffusion outperformed omniscient multicast and flooding
- Application-level data dissemination has the potential to improve energy efficiency significantly
  - Duplicate suppression is only one simple example out of many possible ways.
  - Data aggregation
- All layers have to be carefully designed
  - Not only network layer but also MAC and application level
- More experimentation is needed

Tiny Diffusion

- Implementation of Diffusion on resource constrained UCB motes
  - 8 bit CPU, 8k program memory, 512 bytes data memory
  - Subset of full system
  - Retains only gradients and condenses attributes to a single tag
  - Entire system runs in less than 5.5 KB memory

Contd…

- Tiny OS adds ~3.5 KB and 144 bytes of data (inclusive support for radio and photo sensor
- Diffusion adds ~2k code and 110 bytes of data to tiny OS

Tiny Diffusion Functionality

- Resource constrained
- Limited cache size -- currently 10 entries of 2 bytes each
- Limited ability to support multiple traffic streams. Currently supports five concurrently active gradients

Pull vs. Push Variations

- One could also diffuse data from source, in search of relevant sinks – a completely dual approach
- Or one could try a combination push/pull strategy:
  - pull: sink 2-D, source 0-d
  - push: sink 0-d, source 2-d
  - what about: sink 1-d, source 1-d (double rulings)

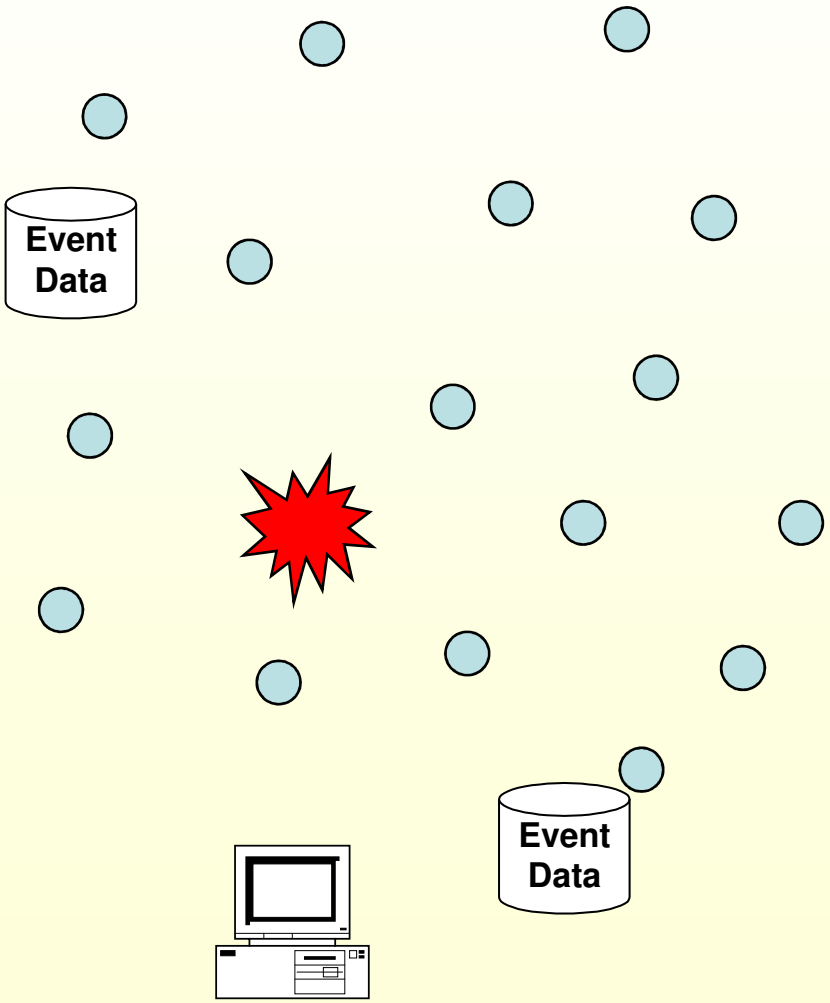
Alternative Methods

- Query flooding
  - Expensive for high query/event ratio
  - Allows for optimal reverse path setup
  - Gossiping schemes can be use to reduce overhead
- Event Flooding
  - Expensive for low query/event ratio
  - there are effective methods for gradient setup
- Note :
  - Both of them provide shortest delay paths

Discrete Queries

Approaches to Accessing Data in Sensor Networks

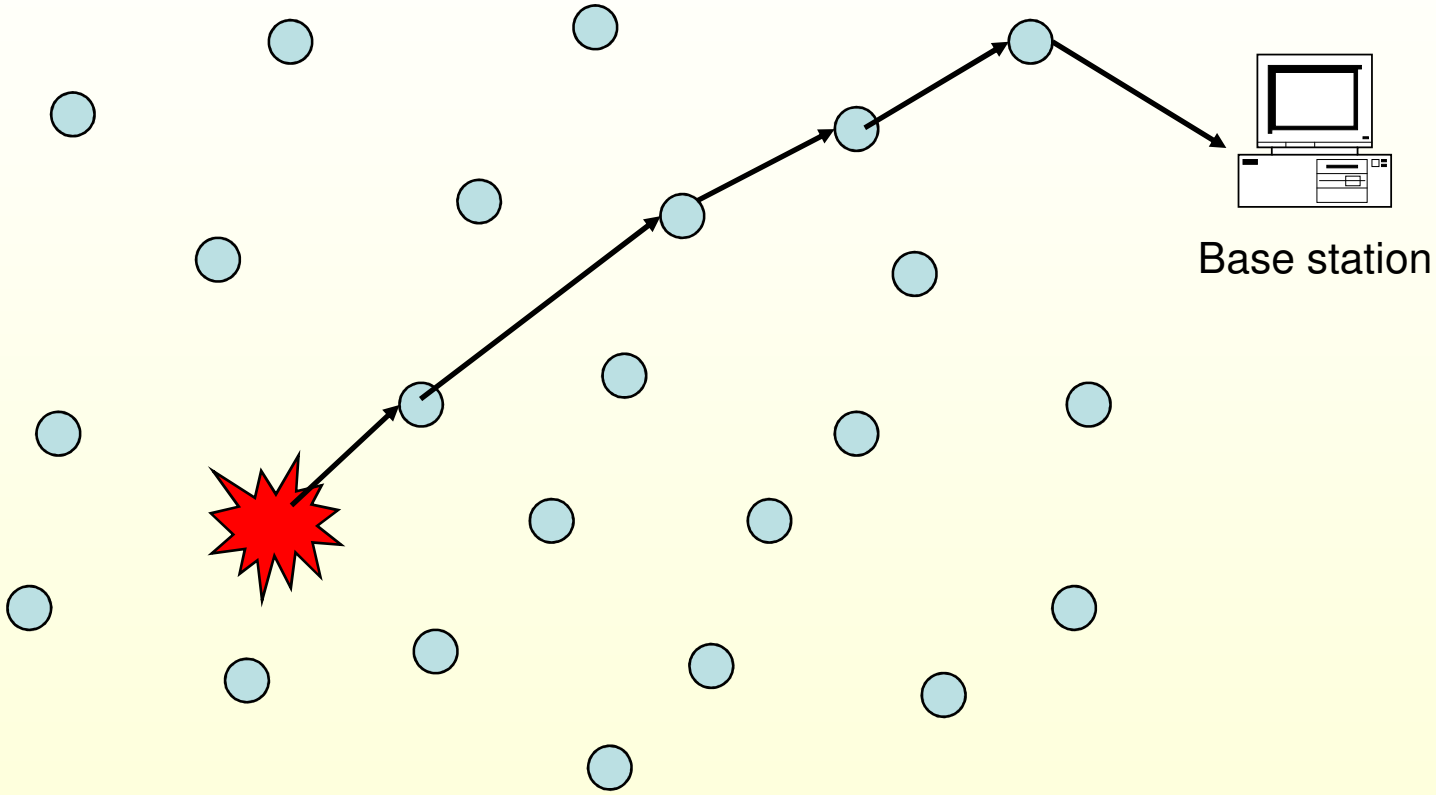
- Warehousing approach
  - Device data is extracted in a predefined way
  - Data is stored in a centralized DB server
  - Queries are evaluated on the centralized DB server
- Distributed approach
  - Data is stored in different locations in the network
  - Queries are evaluated by contacting individual devices
  - Portions of queries are executed on the devices



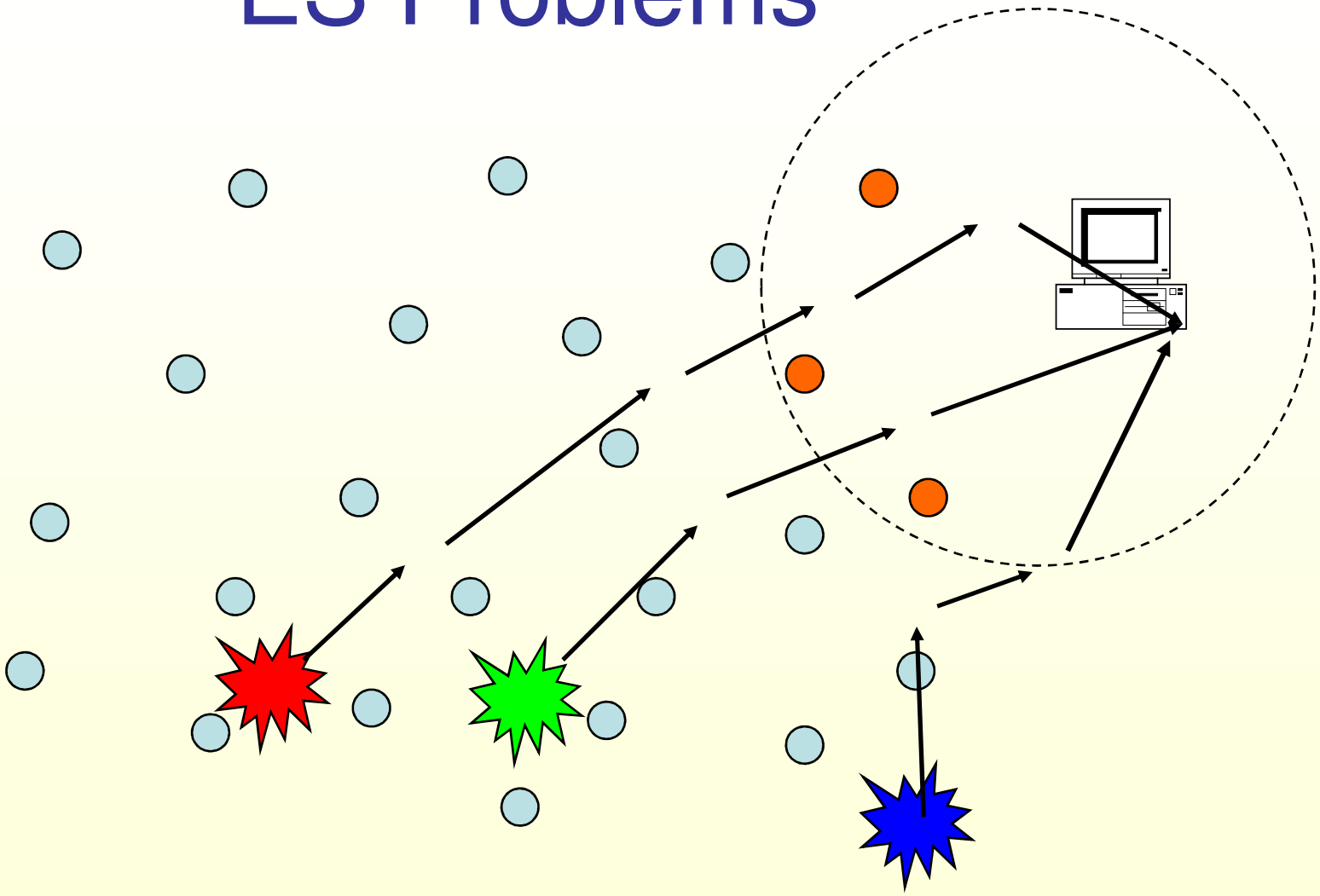
Possible Approaches on Where to Store Data

- External Storage (ES) [Centralized]
- Local Storage (LS) [Distributed]
- Data-Centric Storage (DCS) [Distributed]

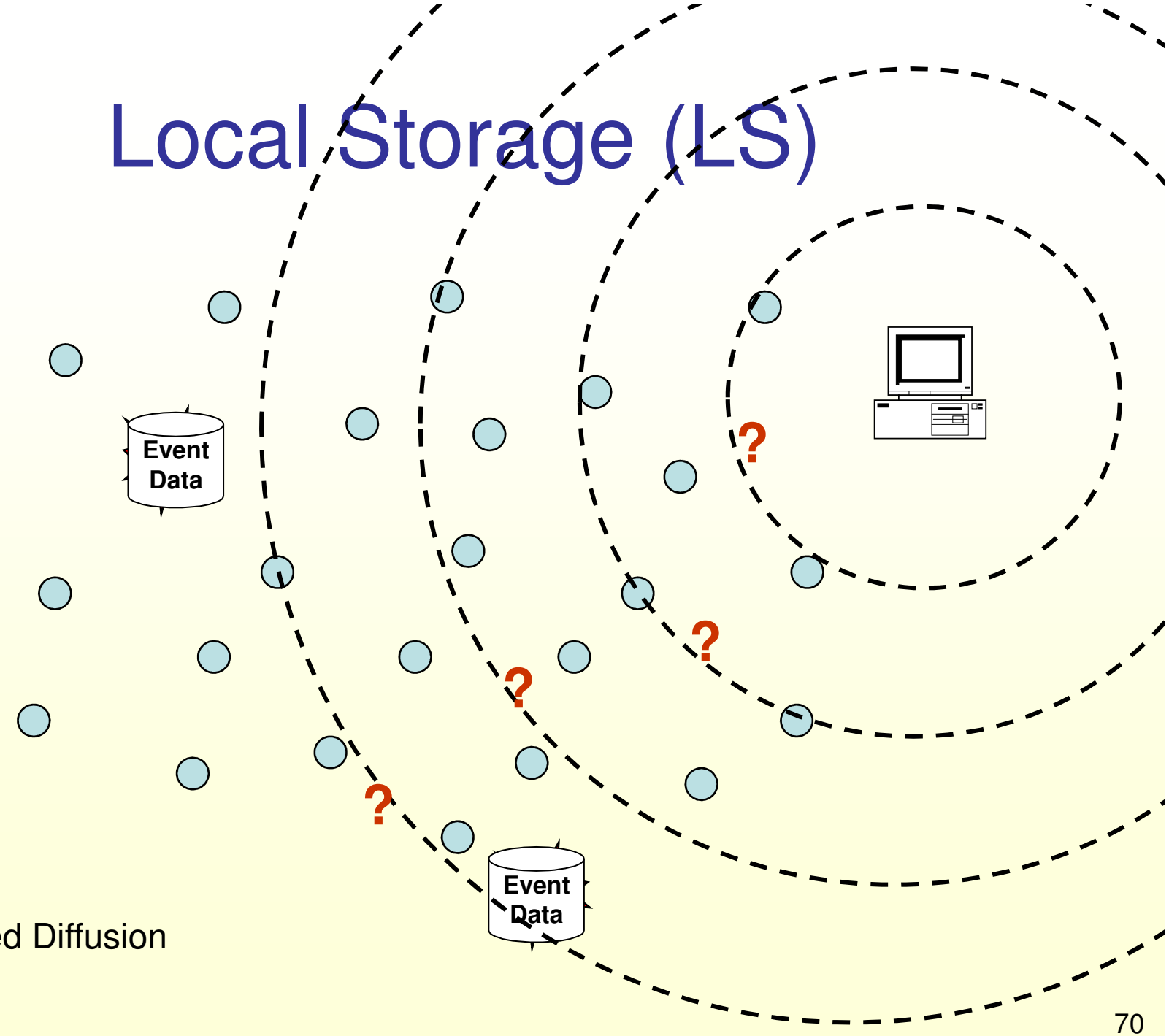
External Storage (ES)



ES Problems

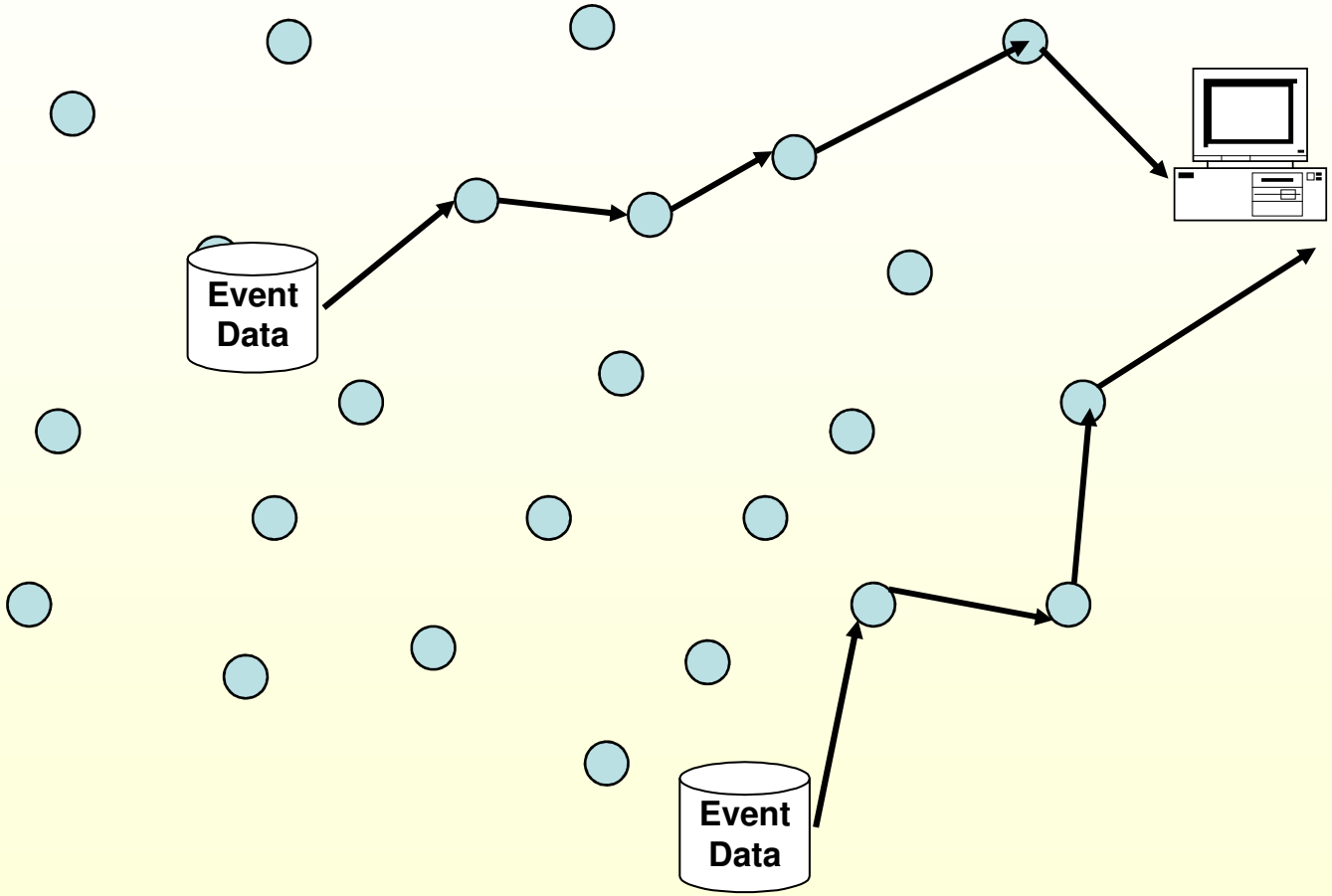


Local Storage (LS)



Directed Diffusion

Local Storage (LS)



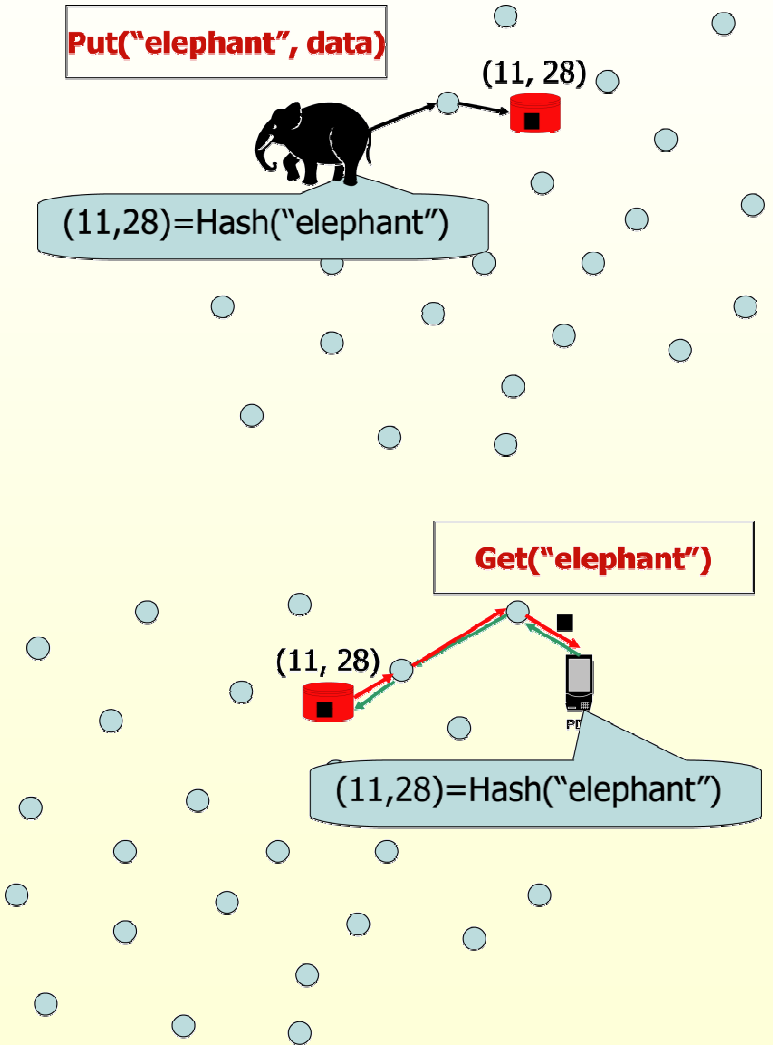
Data-Centric Storage (DCS)

- Data-Centric: data is named by attributes
- Event data is stored, by name, at home nodes; home nodes are selected by the named attributes
- Queries also go to the home nodes to retrieve the data (instead of to the nodes that detected the events)
- Home nodes are determined by a hash function, routes by GPSR: Geographic Hash Tables

Geographic Hash Tables (GHT)

[Ratnasamy, Karp, Shenker, Estrin, Govindan, Yin, Yu '03]

- Event data is stored, by name, at home nodes; home nodes are selected by the named attributes, via a hash function
- Queries also go to the home nodes to retrieve the data (instead of to the nodes that detected the events)
- Routing usually done using a geographic routing protocol (GPSR)



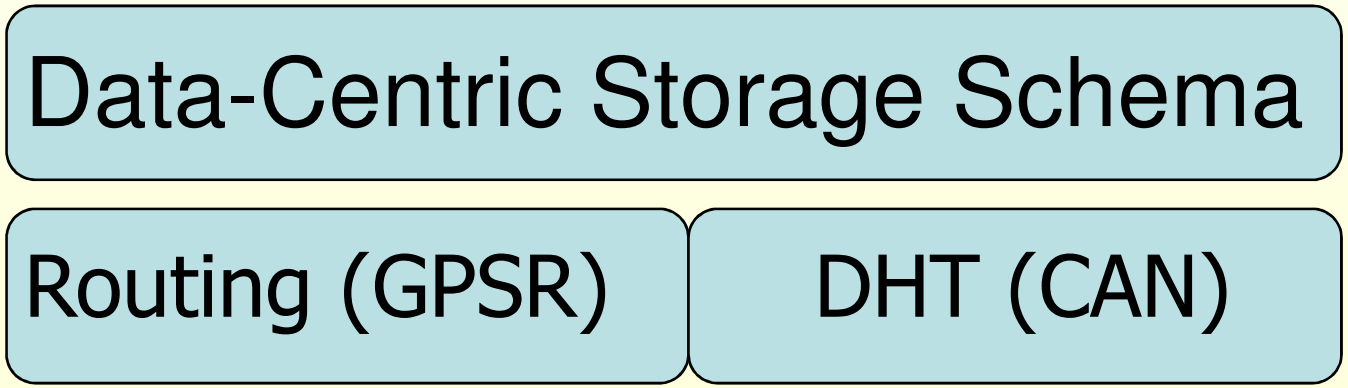
Algorithms Used by GHT

- PEER-TO-PEER attribute look up system

  (data object is associated with key and each node in the system is responsible for storing a certain range of keys)
- Geographic hash tables use GPSR for routing (Greedy perimeter stateless routing)

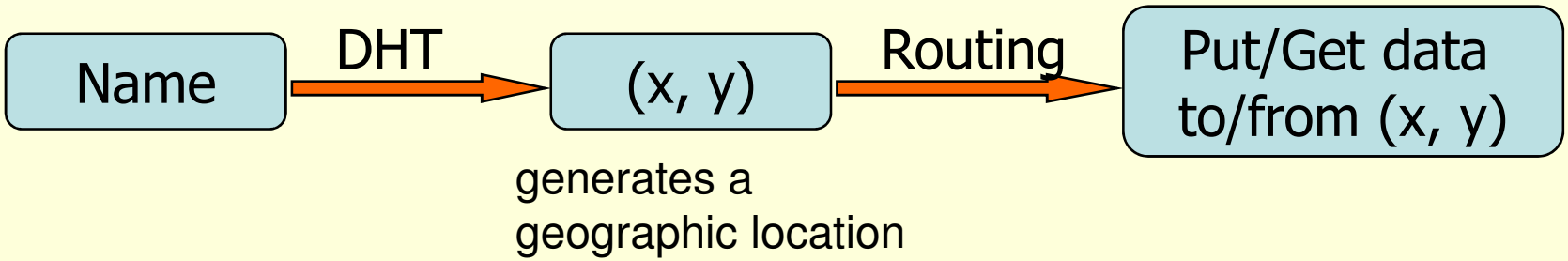
The Big Picture for GHTs

- Based on P2P lookup algorithm (Ratnasamy) and geographic routing (Karp)



Distributed Hash Table (DHT)

- **`void Put(key,value)`**
  - Stores **`value`** in home node of the sensor network, according to attribute **`key`**
- **`Value Get(key)`**
  - Retrieve **`value`** from home node of the sensor networks according to **`key`**

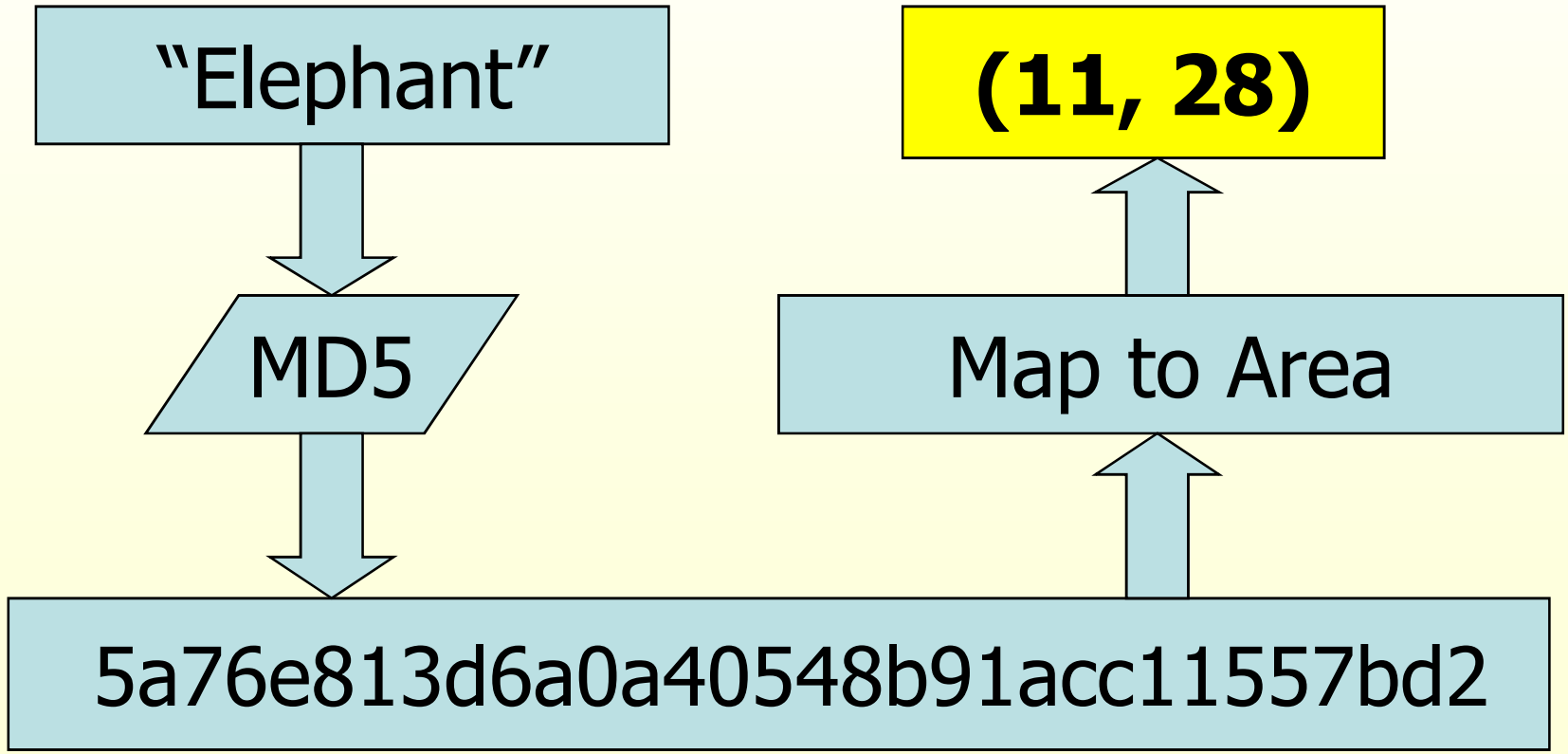


Properties of DHT

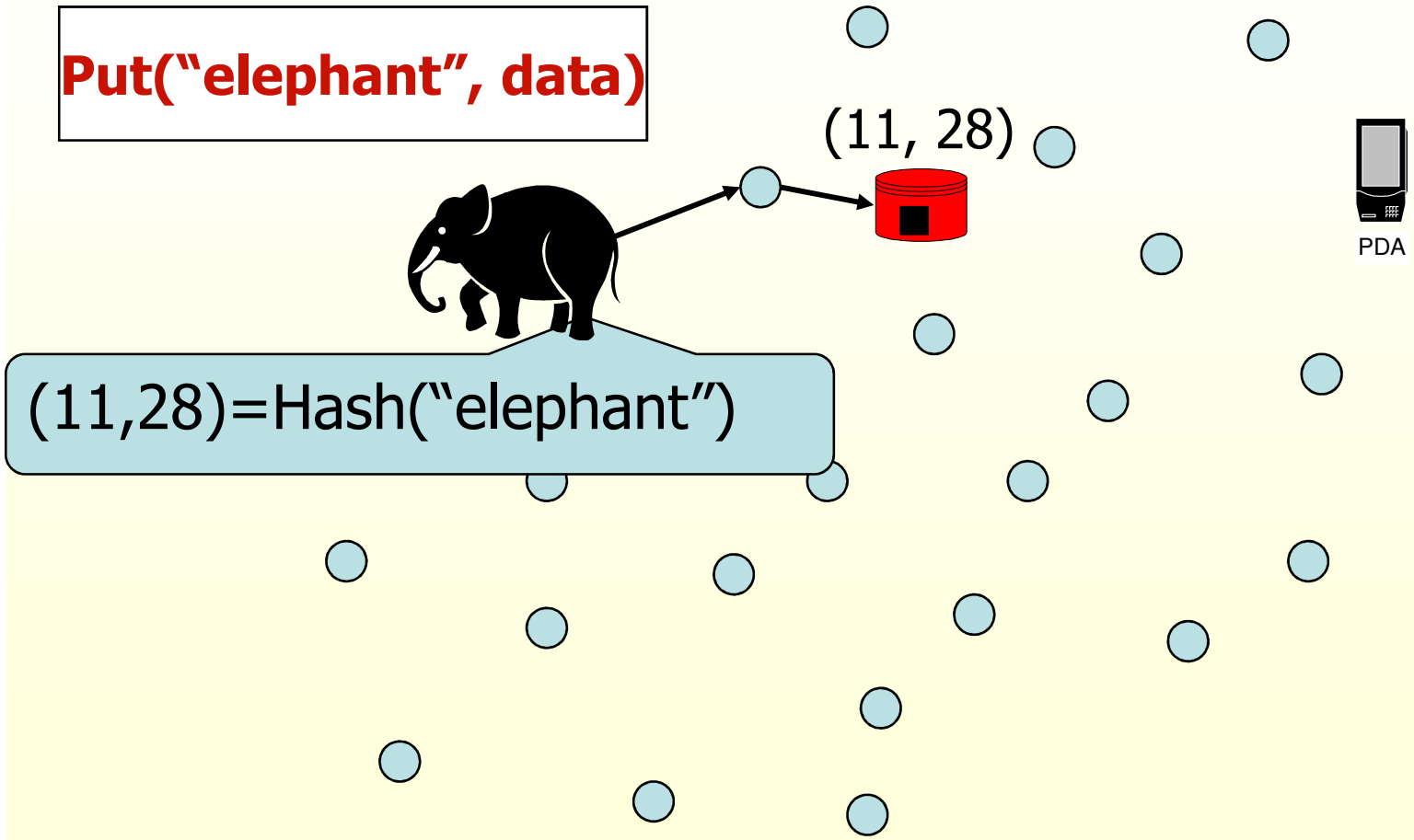
- Uses a distributed hash function
  - Hash function is known to all nodes
  - Every home node takes care of roughly the same amount of event types (load balancing)
  - Evenly distributed geographically (geographic balancing)
- Hash candidate: Message Digest Algorithms
  - Such as SHA-1, MD5

DHT - Example

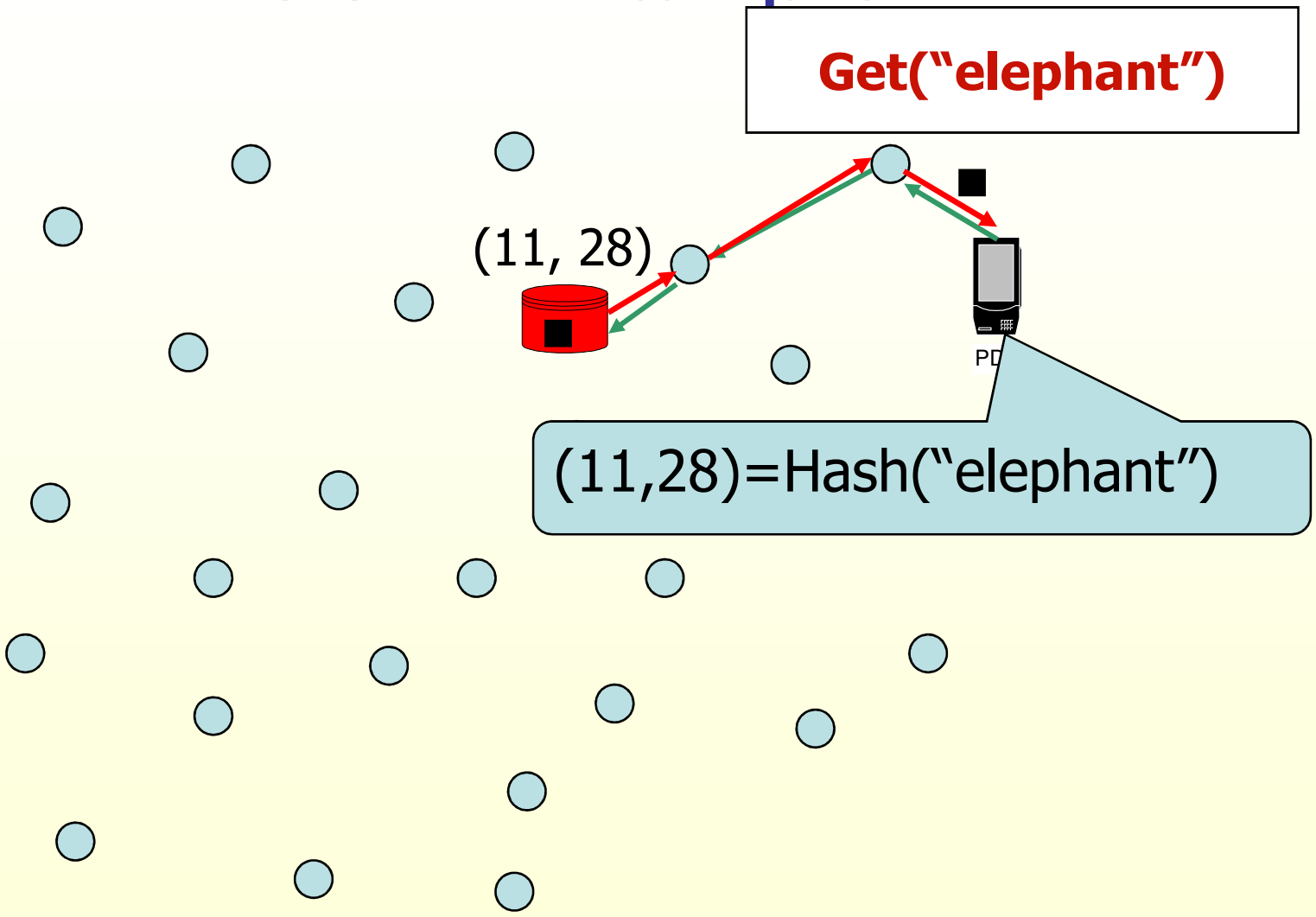
- Example



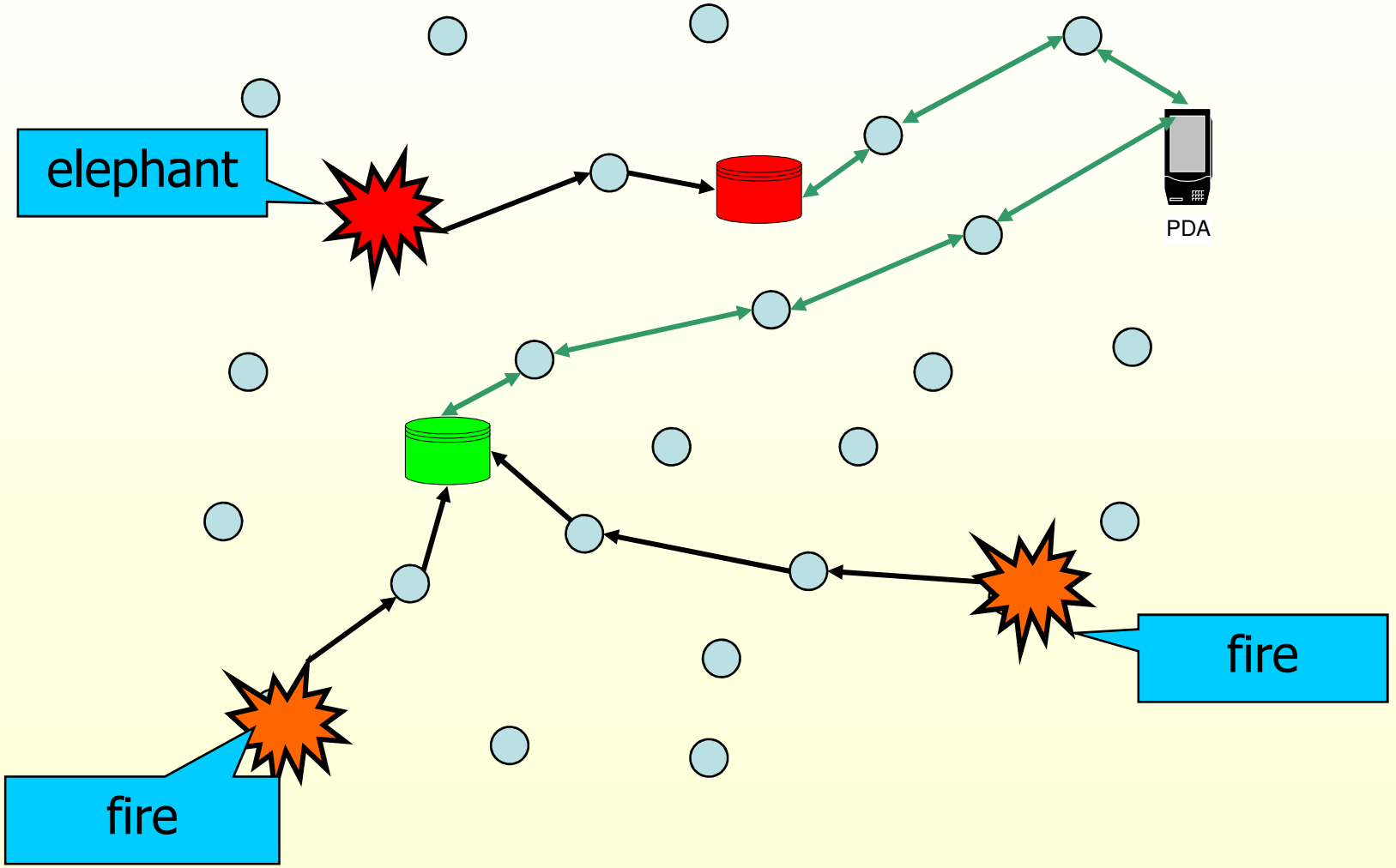
DCS – Example Revisit



DCS – Example

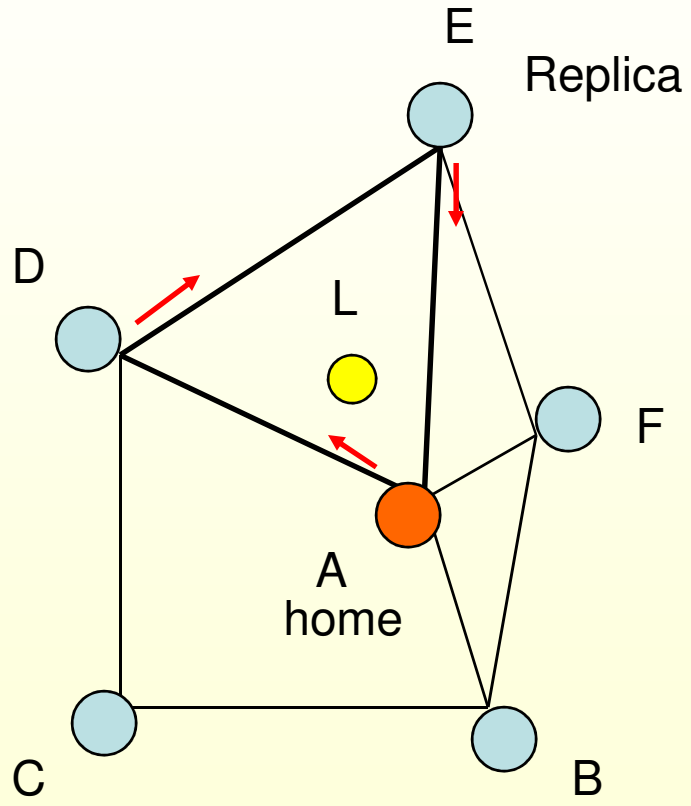


DCS – Example



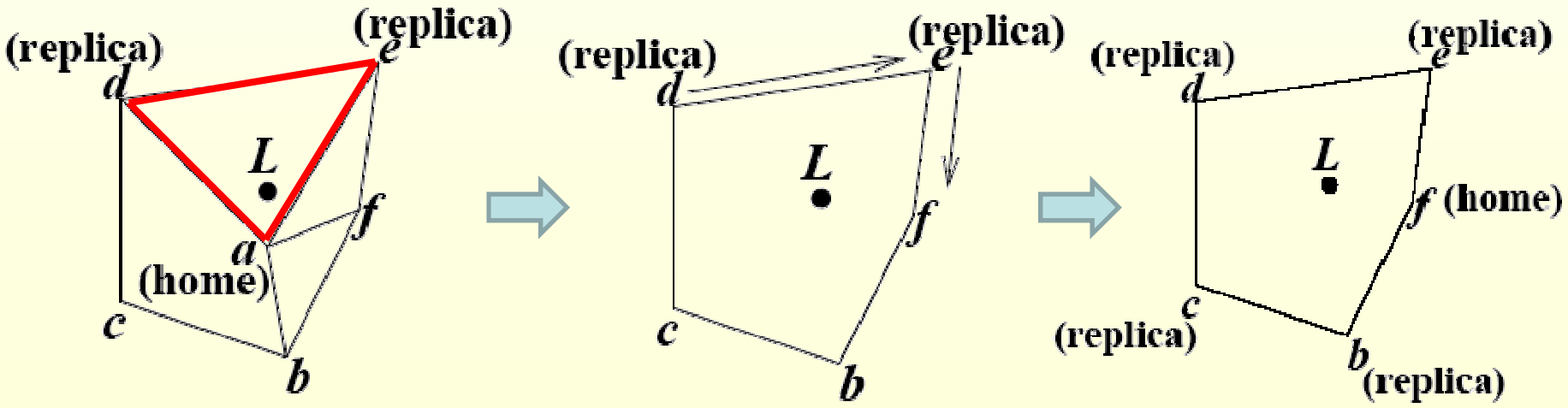
Home Node and Home Perimeter

- In GHT packet is not addressed to specific node *but only to a specific location in the field*
- The packet will circle around the face of the GPSR face containing the destination location
- The packet will traverse the entire perimeter that encloses the destination and eventually be consumed at the home node (the node closest to destination) – and that perimeter is known as the home perimeter



GHT: maintenance

- Home node periodically refresh replication by sending a packet to the hashed location L.
- If the timer of the replica times out, then a replica node initiates a refresh.



Geographic Hash Table (GHT) Summary

- Advantages:
  - simple.
  - load balancing in storage.
- Disadvantages:
  - Not locality-sensitive. Consumer may travel far to fetch data even if the producer is close.
  - Fault tolerance: home nodes could fail or move
  - Overloads nodes on hole boundaries.
  - Nodes with popular data become bottlenecks.

Information Brokerage Issues

- Find a good balance between cost of information replication (storage size) and cost of information discovery (query time)
- Load balance
- Robustness

- Information brokerage is intimately coupled with
  - how network nodes are named
    - do we have coordinates?
  - how routing is done in the network

Information Locality and
Distance-Sensitive Information Brokerage:
if producer and consumer are at a distance d, the query cost should be O(d)

Conclusions

- Brokerage between information providers and seekers is a fundamental problem in wireless sensor networks
- Reactive protocols are best, to accommodate dynamics both in the phenomena being monitored, as well as in the network itself
- Both push and pull paradigms apply, and various combinations
- In-network storage can provide rendez-vous points between data producers and consumers

The End