Directed Diffusion: Intanagonwiwat, Govindan, Estrin [Intanagonwiwat00a]

CSci551: Computer Networks
SP2006 Thursday Section
John Heidemann

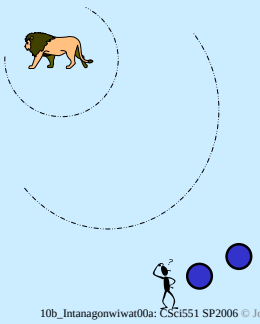
Key ideas

- data-centric routing
  - no host addresses
  - instead have query and attribute-based routing
- query propagation
  - localized: all communication is between neighbors
  - terms: gradients, path reinforment
- nodes interpret and process information
  - nodes have application-specific information/code
  - rather than just "routing packets"
  - nodes can actually process the data (ex: aggregation)
- multicast protocol

Sensor Network Alternatives

- remote
- centralized
- distributed

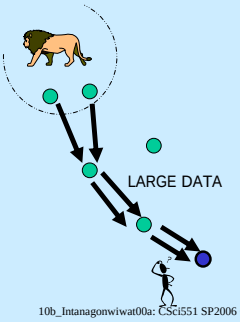
Today's Remote Sensors



- **remote** approach
  - few, large, expensive sensors are far from phenomena
  - they use complex algorithms to factor out noise
- problem:
  - SNR decreases rapidly with distance

  ⇒noise limits performance

Today's Centralized Sensors

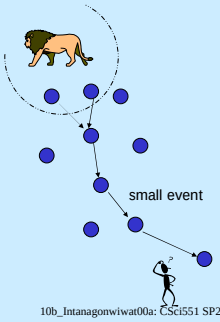


- **centralized** approach
  - some, cheap?, dumb sensors are close to phenomena
  - collected data is sent to process at smart, expensive **central node** (or nodes)
- problem:
  - raw data is large for wireless

  ⇒ lots of energy to send

  ⇒ expensive high bw radio

Future Sensor Networks

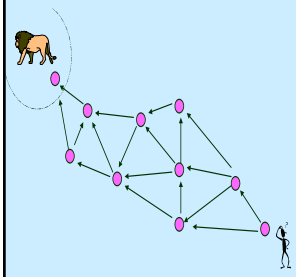


- **distributed** approach
  - many small, smart, cheap sensors close to phenomena
  - *all* sensor nodes interpret and process collected data
  - sensors may *aggregate* responses
- problem:
  - routing data with many nodes
  - dealing with dynamics (node failures, radio propagation)

Directed Diffusion

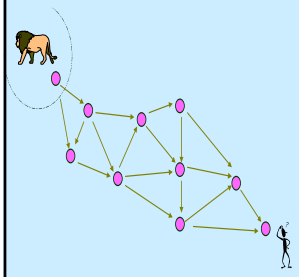
- users express interest in data (becoming *sink)*
  - specified by *attributes*, not IP address
- sink sends out *interests*
  - by default: flooded through network
  - could use attributes for help (geography)
  - could use cached old routes
- *sources* reply to interests with data
  - first, send *exploratory ("low rate")* data
  - flooded on return paths
- *sink* reinforces a path
  - sets up *reinforced path*
  - *non-exporatory ("high rate")* data only follows reinforced paths

Interest Propagation



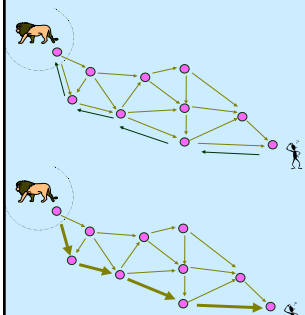
- Initial interest specifies low data rate as exploratory
  - The desired data rate will be achieved by reinforcement
- After receiving an interest, the node creates states and re-sends to a subset of its neighbors
  - Flood the interest
  - Direct interest or limit scope using GPS info
  - Direct interest using route history

Exploratory Data Propagation and Gradient Establishment



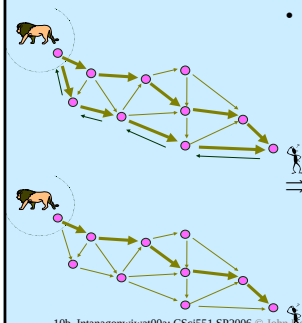
- sensors's first data is *exploratory (low-rate data)*
- sent throughout network, establishing *gradients*
  - map attributes to next hop at each node in network
  - nodes have multiple gradients

Reinforcement



- sink *reinforces* some path to get *high rate* or *non-exploratory* data
- each hop propagates reinforcement back to sources
- which link to reinforce?
  - default: lowest latency
  - alternatives: maximum remaining energy, or greedy tree

Negative Reinforcement



- should detect and prune unnecessary paths
  - (paths that send the same info)
  - implicit negative reinforcement (just let gradient time out)
  - explicit negative reinforcement

⇒ *negative* reinforcement

  - implicit negative reinforcement (just let gradient time out)
  - explicit negative reinforcement

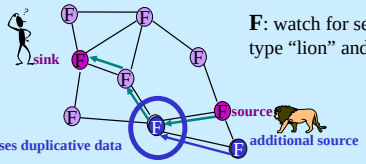
Comments

- how does this compare to other routing protocols?
  - different
    - application specific, in that addressing is done in app-specific attributes, not IP addresses
    - routing information is spread throughout network (unlike DSR where it's in the header)
    - multicast (multiple sources and sinks)
    - has in-network processing
  - similar
    - flooding of interest like DSR RREQ
    - reinforcement sort of like DSR path selection
- what parts of this protocol are inefficient?
  - flooding
    - in both directions!
    - there are ways to get rid of one flood
  - risk of sending data on parallel links

Naming and attributes

- IP
  - IP address
    - but sensor nets have many nodes, and they're ad hoc
    - why IP? unique, globally addressable, aggregatable
  - and DNS hostnames and URLs
- diffusion
  - not necessarily global unique ids, just need to identify your neighbors
  - replaces higher-level naming (DNS, etc.) with attributes
  - combines name resolution with routing
  - not perfect
    - fair amount of flooding (resource discovery)
    - not obvious how to aggregate attributes
  - interesting *in the data*, not the nodes

Filters for In-Network Processing



F: watch for sensor data type "lion" and aggregate it

- Support *app-specific*, *in-network* processing
  - duplicate suppression
  - aggregation
  - collaborative signal processing
  - caching, etc.
- Mechanism:
  - assume filters are pre-deployed in net
  - match on attributes
  - filter can take any action (send new msgs, suppress messages, etc.)

Why In-Net Processing?

- why do in-net processing in sensor nets and not in the internet?
  - could do in-net processing in the internet
    - but it's usually considered wrong
    - and easy to get wrong
  - want to be more efficient
    - processing is cheaper than communication
      - when this paper was written, ~1000 instructions == tx 1 bit of data
      - the internet is the opposite way (with backbone routers)
  - also if you're doing *control* in your sensornet
    - need to think about latency

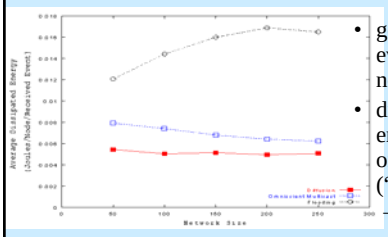
Differences from Traditional Networking (IP)

- data centric operation
  - care about data, not end hosts
  - route according to attributes, not address
- in-network processingv

Evaluation

- compare:
  - directed diffusion
  - flooding
  - omniscient mcast (no setup cost)
- metrics:
  - energy
  - delay
  - event delivery ratio
- varying:
  - network size (but constant node density)
  - effects of in-network processing (duplicate suppression)
  - effects of negative reinforcement
  - radio energy model

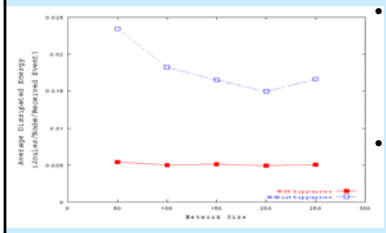
Evaluation: Energy vs. Size



[Intanagonwiwat00a, figure 4a]

- good performance even as number of nodes grows
- diffusion uses *less* energy than omniscient multicast ("optimal")
  - how? in-network processing… diffusion is doing aggregation and so sends less data

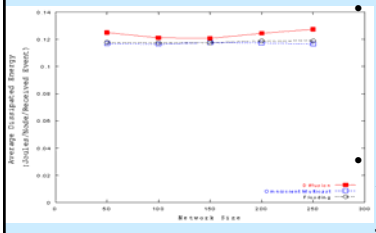
Evaluation: Effect of Duplicate Suppression



[Intanagonwiwat00a, figure 6b]

- duplicate suppression is critical to diffusion
- shows the importance of app-specific in-net processing

Result Sensitivity: Radio Energy Model



[Intanagonwiwat00a, figure 6c]

- used "TDMA-MAC" style energy model
  - has low idle energy
  - idle:rx:tx 1:10:16
- standard 802.11 wastes most energy just listening
  - idle:rx:tx 1:1:1.6

Additional Directions

- SOSP '01 paper:
  - how attributes work in detail
  - experimental (not simulation) results
  - nested-processing (another example of in-net processing)
- SenSys 2003 paper:
  - other versions of diffusion: "push" and "one-phase pull"
  - should have a *family* of protocols (different performance in diff. cases, same APIs)
- other kinds of in-net processing?
  - collaborative signal processing

Other questions/observations?

- mobility?