Chapter 6
Routing
and
Topology control
Overview

1. Routing basics
2. Proactive vs. reactive protocols
3. Addressing: geographic, id-centric, data-centric
4. Topology control
Network in computer science

- Network is a graph $G = (V,E)$
  - $V$ – set of all nodes
  - $E$ – set of all edges: $(v_1,v_2) \in E \subseteq V^2$

  $V = \{ A, B, C, \ldots \}$
  $E = \{ (A,B), (B,C), (C,F), \ldots \}$

- Routing
  - Find a (the best) path from source to destination

- WSN graph – idealized model
  - connectivity is function of time
  - nodes mobility
  - asymmetric links
Routing in wireless networks

Motivation

- Broadcast wireless medium, routing necessary?
- Radio range limited, direct communication source-sink impossible...
- ...multi hop communication required
- Routing necessary
  - Find (optimal) route to destination on source...
  - ..and on intermediates nodes (forwarding)
Routing in wireless networks

Motivation

• Direct communication sometimes possible

• Multi hop communication (often) more efficient than a direct one
  – TX energy \( \sim \) distance\(^\alpha\) (\(\alpha \geq 2\))
  – (too) many receivers (overhearing)
    -> MAC solution?
  – Collisions, reduced throughput
Sensor Network Challenges

- nodes do not operate
  - hardware/software failures (repairs inconvenient)
  - hostile environment

- nodes move around (?)

- wireless communication unreliable

- scalability: number of nodes from several to many thousands

- limited energy
  - energy-efficient routes (e.g. routes that consume little energy, load balancing)
  - small protocol overhead (route maintenance consumes energy)
Energy-efficient Routing

Goals

- Minimize energy/bit (the path that consumes least energy)
  Example: A-B-E-H

- Maximize network lifetime
  Time until first node failure, loss of coverage, partitioning

- Maximum total battery capacity (without detours)
  Sum battery levels of paths, and choose the one with the greatest level (A-C-F-H)
Routing in OSI Reference Model

**Diagram:**

- **Application (PHY + MAC)**
- **Presentation**
- **Session**
- **Transport**
- **Network**
- **Data link (MAC)**
- **Physical**

```
<table>
<thead>
<tr>
<th>dest</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>E</td>
</tr>
<tr>
<td>G</td>
<td>F</td>
</tr>
</tbody>
</table>
```

Routing between nodes A to G:

- A -> B -> C -> E -> F -> G

Data flow through the layers:
Routing maintenance

- Routing: additional information required (e.g. neighbours, network topology,...)

- Routing information exchanged with other nodes
  - What information?
  - With which nodes exchanged?

- How to determine route? Usually routing tables
Distance Vector Routing

- Called Bellman-Ford (1957), Routing Information Protocol (RIP) in Internet
- each node maintains routing table (no topology graph)
- nodes known distances (hops, delay...) to their neighbours
- periodically distance vectors sent to all neighbours
- Routing table determined from received distance vectors

<table>
<thead>
<tr>
<th>dest</th>
<th>next</th>
<th>distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>G</td>
<td>B</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>L</td>
<td>J</td>
<td>3</td>
</tr>
</tbody>
</table>
Distance Vector Routing

+ Fast good news propagation
- Slow bad news propagation (count-to-infinity problem)

(a) Initially
1 2 3 4
1 2 3
1 2
1

(b) After 1 exchange
1
3 2 3
3 4 3
5 4 5
5 6 5
7 6 7
7 8 7
...
Count-to-infinity solution

Destination Sequenced Distance-Vector Routing (DSDV)

• Count-to-infinity problem solution: sequence numbers!

sequence numbers origin from destination, contained in update packets and routing tables
Link State Routing

- Link state = neighbor connectivity
- Maintain (store) network topology graph
- Calculate the best path locally (e.g. Dijkstra shortest path algorithm)

Network graph maintenance:
1. Discover neighbours (ID, link cost)
2. Broadcast neighbour list (link state packets) - send to all nodes!
3. On RX update network graph
### Distance Vector vs Link State Routing

<table>
<thead>
<tr>
<th>Distance Vector Routing</th>
<th>Link State Routing</th>
</tr>
</thead>
</table>
| • Send routing table to neighbours | • Send link state (neighbour list) to all nodes  
  → large overhead |
| • Suffers from count-to-infinity (long convergence) | • Better (faster) convergence than DV |

**In ARPANET:** Distance Vector Routing replaced by Link State Routing

Routing updates necessary, but when?
• Periodically
• On event, what event?
Overview

1. Routing basics

2. Proactive vs. reactive protocols

3. Addressing: geographic, id-centric, data-centric

4. Topology control
Proactive Protocols

- Routing structures (routing tables, network graph) maintained continually -> route updates sent regardless of data traffic

- Route updates sent even when no data traffic
  \[\rightarrow\] overhead

- Maintained continually:
  - periodically
  - on topology change (e.g. Node fails)

- Referred to as table-driven
Destination Sequenced Distance-Vector (DSDV)

• Modified Distance-Vector protocol

Routing table at node 15

<table>
<thead>
<tr>
<th>Dest</th>
<th>Next</th>
<th>Dist</th>
<th>Seq</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>2</td>
<td>26</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>3</td>
<td>26</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>
Destination Sequenced Distance-Vector (DSDV)

- Aging mechanism against count-to-infinity problem / loops each route entry with sequence number originated by destination

- Optimization
  - Reduction of full update periods
  - Incremental route updates (smaller) sent on topological changes
Fisheye State Routing (FSR)

- Goal: reduce Link State protocol overhead
- Detailed knowledge about distant nodes necessary?
- Imprecise knowledge = routing overhead reduction
- Data moves closer to dest -> enters regions with detailed knowledge
Fisheye State Routing (FSR)

- Link State with limited flooding
  - LS packets (node + its neighbors) exchanged periodically only with neighbors (LS: forward link states of all nodes)
  - Frequent TX update for near stations
  - Rare TX update for stations afar
Reactive Protocols

• Idea: No data traffic = no route maintenance

• Find route just before data transmission

• No route information (next hop, distance,...) initially;
Steps to be taken:

  1. Discover path to dest (routing packets flooded)
  2. Destinations unicast response
  3. Data transmission

• Discovered path used for future data transmissions

• Referred to as on-demand
Ad Hoc On-Demand Distance Vector Routing (AODV)

- Source sends route request (RREQ)
- Intermediate nodes record (memory) where RREQ came from required for reverse path
- Destination sends back directed route reply (RREP) reverse path used
- Intermediate nodes activate route routing table
Ad Hoc On-Demand Distance Vector Routing (AODV)

Link failure maintenance

• How to detect a failure?

• Source moves – route discovery reinitiated

• Failure along route
  • Upstream neighbor detects failure
  • Link failure notification to upstream nodes
  • Source gets link failure notification
  • Route discovery reinitiated

• Local repair
Dynamic Source Routing (DSR)

- Source sends route request (RREQ)
- Intermediate nodes add ID in RREQ and forward...
- ...or send back RREPLY when route to destination is known
- Destination sends back RREPLY
- Asymmetric links supported: Destination initiates route discovery to source
- Routing: packets contain the whole route
Dynamic Source Routing (DSR)

Optimizations:

- Route caching (in both directions: to source and to destination)
  - from forwarded packets
  - overhearing
- RREPLY created from local caches
- RREQ with hop limit (e.g. expanding ring starting from 0 hops)
Dynamic Source Routing (DSR)

Error recovery

basic DSR recovery

- package salvaging
  use local cache
Zone Routing Protocol (ZRP)

• Hybrid Protocol (Proactive and Reactive)

• Nodes maintain two zones
  • Intra-zone: proactive
  • Inter-zone: reactive

• Zone radius in hops

• Bordercast instead of broadcast (e.g. in RREQ)

• Behaviour adjustable
  • Larger radius: more proactive
  • Smaller radius: more reactive
# Proactive and Reactive Protocols

<table>
<thead>
<tr>
<th>Proactive</th>
<th>Reactive</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Maintain routing structures (routing tables, network graph) continually</td>
<td>• Do nothing when no data traffic</td>
</tr>
<tr>
<td>• periodically</td>
<td>• Find path shortly before data transmission</td>
</tr>
<tr>
<td>• on event (topology change)</td>
<td>1. Discover path (routing packets)</td>
</tr>
<tr>
<td>• Route updates sent even when no data traffic</td>
<td>2. Transmit data</td>
</tr>
<tr>
<td>→ overhead</td>
<td>• Referred to as on-demand</td>
</tr>
<tr>
<td>• Referred to as <strong>table-driven</strong></td>
<td>• Referred to as on-demand</td>
</tr>
</tbody>
</table>
## Summary

<table>
<thead>
<tr>
<th>Proactive</th>
<th>Reactive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Advantages:</strong></td>
<td><strong>Advantages:</strong></td>
</tr>
<tr>
<td>• No delays before data transmission</td>
<td>• Overhead proportional to data transmission mostly much smaller than proactive (what if nodes move frequently?)</td>
</tr>
<tr>
<td>• Alternative paths without delay</td>
<td></td>
</tr>
<tr>
<td>• QoS support</td>
<td></td>
</tr>
<tr>
<td><strong>Drawbacks:</strong></td>
<td><strong>Drawbacks:</strong></td>
</tr>
<tr>
<td>• Large protocol overhead</td>
<td>• Delay before data transmission</td>
</tr>
</tbody>
</table>
ZigBee Routing

ZigBee Alliance

IEEE 802.15.4

- Security Service Provider (SSP)
- ZigBee Device Objects (ZDO)
- App. Support Sub-Layer (APS)
- Network Layer (NWK)
- 802.15.4 Medium Access
- 802.15.4 PHY - Layer

- ZigBee Stack
  - 802.2 LLC
  - SSCS

- Application
  - UDP
  - IP

Application Layer (APL)

Silicon

IP

UDP

IEEE 802.15.4
802.15.4/ZigBee topology

Topology determines the routing algorithm:

- Mesh: AODV
- Tree routing
ZigBee TreeRouting

- 1 PAN coordinator

- Each node must join the network (PAN or router)

- End-device gets address

- Router gets address + range of addresses for children

- Routing based on addresses:
  - Forward to children if there’s a child with the address
  - Otherwise, forward to parent
Overview

1. Routing basics
2. Proactive vs. reactive protocols
3. Addressing: geographic, id-centric, data-centric
4. Topology control
Identity-centric

- Each node with an unique (?) address, but what address length?
  - Ethernet MAC: 48-bit
  - IP: v4 32-bit, v6 64-bit
  - IEEE 802.15.4: 16-bit short address and 64-bit extended address

- Address length influences:
  - Size of transmitted frames
  - Size of routing tables

- Trade-off:
  - Short-addresses: less memory needed, less data transmitted but less nodes addresses
  - Long-address: opposite
Identity-centric

• Address allocation:

  • Assign an entity an address from a given pool of possible addresses
    • Offline / online
    • ZigBee: nodes gets their address on network join

  • Distributed address assignment (centralized like DHCP does not scale)

  • Let every node randomly pick an address; for given size of address space, risk of duplicate addresses
Identity-centric

- Directed towards a well-specified particular destination (sink)
- Support for unicast, multicast, and broadcast messages
Data-centric

- Data is focus of attention
  - Redundant nodes; doesn’t matter which node exactly (which ID) sends data information is focus of attention
  - Reporting nodes do not care about the sink’s identity
  - Routing decisions according to the “data” (i.e. Routing table holds interests to neighbors mapping)
  - May reduce network traffic if data is redundant
In-Network Processing

No data-centric paradigm

- May lead to unnecessary traffic due to redundant data

Data-centric

- May reduce network traffic (if data is redundant)
- Early error discovery possible
Data aggregation

Chain-based aggregation

Tree-based aggregation

Grid-based aggregation
Data-aggregation

Tradeoff: latency vs efficiency

\[
\begin{align*}
\text{0} & \rightarrow \text{1} \quad \Delta t \\
\text{1} & \rightarrow \text{2} \quad \Delta t \\
\text{2} & \rightarrow \text{n-1} \quad \Delta t \\
\text{n-1} & \rightarrow \text{n} \quad \Delta t \\
\end{align*}
\]

\[
\begin{align*}
\text{0} & \rightarrow \text{1} \quad \Delta t \\
\text{1} & \rightarrow \text{2} \quad 2 \Delta t \\
\text{2} & \rightarrow \text{n-1} \quad (n-1) \Delta t \\
\text{n-1} & \rightarrow \text{n} \quad n \Delta t \\
\end{align*}
\]
Publish/subscribe paradigm

Decoupling

- **in space**: publisher and subscribers not aware of each other (e.g. sink ID not known to source)
- **in time**: publishing and notification at different times (Event Service as intermediate storage)
- **in flows**: asynchronous (no blocking)

Typical operation: sinks subscribe, sources publish
Directed Diffusion

• Realization of publish/subscribe paradigm
• Data dissemination based on reverse path

Basic behavior

• Each sensor node names its data with one or more attributes
• Other nodes express their interest depending on these attributes
• The sink node has to periodically refresh its interest if it still requires data to be reported to it
• Data is propagated along the reverse path of the interest propagation
Directed Diffusion

Interest propagation

type = four-legged animal
interval = 1s
rect = [-100, 200, 200, 400]
timestamp = 01:20:40
expiresAt = 01:30:40

• Data transmission

type = four-legged animal // type of animal seen
instance = elephant // instance of this type
location = [125, 220] // node location
intensity = 0.6 // signal amplitude measure
confidence = 0.85 // confidence in the match
timestamp = 01:20:40 // event
generation time
Directed Diffusion

Steps:

1. **Interest propagation**
   send named data to neighbours

2. **Convergast tree creation: gradients set up**
   store who interest came from
   (no global unique id required)
   Routing table based on data!

3. **Data propagation**
   send data according to gradients
Rumor Routing

- Agent-based path creation algorithm

- Two extremes:
  - Query flooding if few sinks and many sources
  - Event flooding if few sources and many sinks
  - Rumor routing in between
  - Deliver data to temporary sinks
Rumor Routing

• Agents, or “ants” are long-lived entities created at random by nodes
• These are basically packets which are circulated in the network to establish shortest paths to events that they encounter
Rumor Routing

Optimizations:

- Propagation of several events
- Path shortening
Geographic Routing

• Physical locations (e.g. GPS, time of arrival, time difference of arrival,...)

• No route maintenance required
  • Routing overhead decreased
  • Forward decision locally (based on positions)
Greedy Routing

• Forward to node closer to destination

• Known positions
  • own
  • destination (in packet)
  • last sender (in packet)
  • neighbours (e.g. one-hop broadcasts)

• Strategies
  • most forward within radius (MFR)
  • nearest with forward progress (NFP)
  • neighbor closest to straight line between sender and destination (compass routing)
Greedy Routing

• ...not always possible, what then?

• Traversal of planar graph when greedy fails
  • F local optimum, no neighbour closer to dest -> switch to graph traversal
  • G closer to dest than F, resume greedy

• More sophisticated approaches:
  • Greedy Perimeter Stateless Routing
  • Face-2
Location-Aided Routing (LAR)

- DSR with location information
- limited flooding in route request:
  - expected zone estimation (circle) based on previous destination location; a rectangular region (4 edge points) included in RREQ
  - known destination location -> only nodes closer to the destination forward RREQ
Summary

• Forwarding based on: Locations, Addresses or/and Data

• Geographic routing (locations):
  – Routing overhead decreased (e.g. No route maintenance)

• Data-centric routing
  – Focus of attention: data
  – Often more suitable for WSN than pure id-centric:
    • Flexibility (source-sink decoupling)
    • In-network processing
Overview

1. Routing basics
2. Proactive vs. reactive protocols
3. Addressing: geographic, id-centric, data-centric
4. Topology control
Motivation
- Tasks carried out by dozens, hundreds, thousands... of nodes

- MAC Problems (e.g. collisions, overhearing)
- Enormous routing overhead
- High transmission power consumes a lot of energy

- Choose links explicitly
- Control transmit power
- Backbone, clusterheads
- Number of routing nodes reduced
- Idea: Make topology less complex
- Topology: Which node is able/allowed to communicate with which other nodes
- Topology control needs to maintain invariants, e.g., connectivity

**Taxonomy**

- **Flat network** – all nodes have essentially same role
- **Hierarchical network** – assign different roles to nodes; exploit that to control node/link activity

- Transmission power
- Backbones
- Clustering
- Sleep schedules
Metrics

- **Connectivity** – If two nodes connected in G, they have to be connected in G0 resulting from topology control

- **Stretch factor** – should be small
  - **Hop stretch factor**: how much longer are paths in G0 than in G?
  - **Energy stretch factor**: how much more energy does the most energy-efficient path need?

- **Throughput** – removing nodes/links can reduce throughput, by how much?

- Robustness to mobility

- Algorithm overhead
Flat Network

- Main option: Control transmission power
  - Do not always use maximum power
  - Selectively for some links or for a node as a whole
  - Topology looks “thinner”
  - Less interference
- Idea: Controlling transmission power corresponds to controlling the number of neighbors for a given node
• Cannot set the tx range to X meters, only dBm control

• Set power to X meters: too simplistic as no clear border between RECEIVE and NOT RECEIVE:
  – BER (and PER) changes
  – However, further nodes may have lower BER! (especially indoors)

• tmote sky example:
  – Full power (0 dBm): > 80m indoors, but some nodes <80 m worse PER
  – -25 dBm: a few meters only

<table>
<thead>
<tr>
<th>TXCTRL register</th>
<th>Output Power [dBm]</th>
<th>Current Consumption [mA]</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xA0FF</td>
<td>0</td>
<td>17.4</td>
</tr>
<tr>
<td>0xA0FB</td>
<td>-1</td>
<td>16.5</td>
</tr>
<tr>
<td>0xA0F7</td>
<td>-3</td>
<td>15.2</td>
</tr>
<tr>
<td>0xA0F3</td>
<td>-5</td>
<td>13.9</td>
</tr>
<tr>
<td>0xA0EF</td>
<td>-7</td>
<td>12.5</td>
</tr>
<tr>
<td>0xA0EB</td>
<td>-10</td>
<td>11.2</td>
</tr>
<tr>
<td>0xA0E7</td>
<td>-15</td>
<td>9.9</td>
</tr>
<tr>
<td>0xA0E3</td>
<td>-25</td>
<td>8.5</td>
</tr>
</tbody>
</table>
Backbone

Construct a backbone network
- Some nodes “control” their neighbors; they form (minimal) dominating set
- Each node should have controlling neighbor

• Controlling nodes have to be connected (backbone)

• Only links within backbone and from backbone to controlled neighbors are used

• Formally: Given graph $G = (V,E)$, construct $D \subseteq V$ such that

$$\forall v \in V : v \in D \lor \exists d \in D : (v, d) \in E$$
• Partition nodes into clusters

• Each node in exactly one group (Except from gateways - “bridging” between two or more clusters)

• Hierarchical routing:
  – Nodes send packets to cluster heads
  – Cluster heads deliver packets to appropriate cluster

  – Only cluster heads involved in routing
  – Routing (i.e. routing tables) limited to cluster heads
Clustering (LEACH)

- Serving as a clusterhead can put additional burdens on a node
  - For MAC coordination, data forwarding...

- Let this duty rotate among various members
  - Periodically reelect – useful when energy reserves are used as discriminating attribute
  - LEACH – determine an optimal percentage P of nodes to become clusterheads in a network
    - Use 1/P rounds to form a period
    - In each round, nP nodes are elected as clusterheads
    - At beginning of round r, node that has not served as clusterhead in this period becomes clusterhead with probability P/(1-p(r mod 1/P))
Summary

- Large networks:
  - Massive routing overhead
  - Collisions

- Topology control required
  - Flat networks (power control)
  - Hierarchy (backbone, clustering)
  - Adaptive node activity
Routing in Ad Hoc & WSN Summary

- Internet routing (Distance-Vector, Link State) not directly usable in Ad Hoc & WSN
- No routing „one size fits all”, very application specific
- Data-centric: efficient means to energy efficiency (?)
- Topology control essential in large networks
Thank you for your attention!