CSE 122/222C ; WES 269

Low-Power MACs, Meshing, and Flooding

Pat Pannuto, UC San Diego

ppannuto@ucsd.edu

Low-Power MACs, Meshing, and Flooding Goals

- Investigate a decade of low power MAC research
  – Understand how low power MACs complicate routing

- Understand why IoT might want to use mesh networks

- Look at one ”traditional” mesh network algorithm in-depth

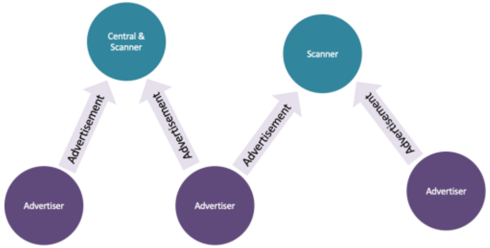
- Look at emerging “non-traditional” ideas, see counter-intuitive tradeoffs

Outline

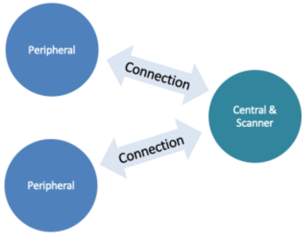
- **Low-Power MAC Design**
- Simple Routing
- Mesh Routing
- Flooding

THE question for low-power devices: When to talk, to **listen**?

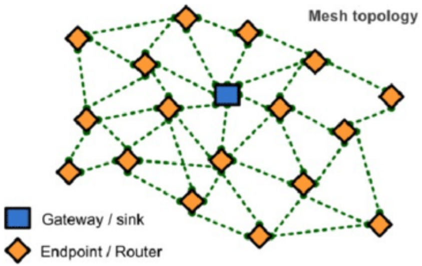
- Recall BLE, where advertisements are broadcast:
  - **Who** do I talk to is simple, anyone within one-hop
  - **When** do *advertisers* talk, listen?
  - **When** do *scanners* listen, talk?



- Recall BLE, where connections use a star topology:
  - **When** do *centrals*, *peripherals* talk and listen?



- **What happens when networks get bigger?**
  - **No device has a "global" view**



Always-on* Radios Simplify Protocols

*Or at least “usually-on” / “often-on”

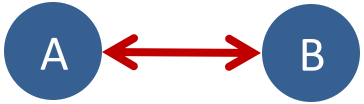
- Many protocols assume a more-powerful device with lots of energy
  - BLE: Central
  - Thread: Router/Leader
  - Zigbee: Router/Coordinator
  - WiFi: Router
  - LoRa: Gateway

- This assumption simplifies the “when to listen” problem
  - Powerful device: always listen
  - Low-power device: listen-after-talk or synchronized schedule

What do we do when everyone is equal?

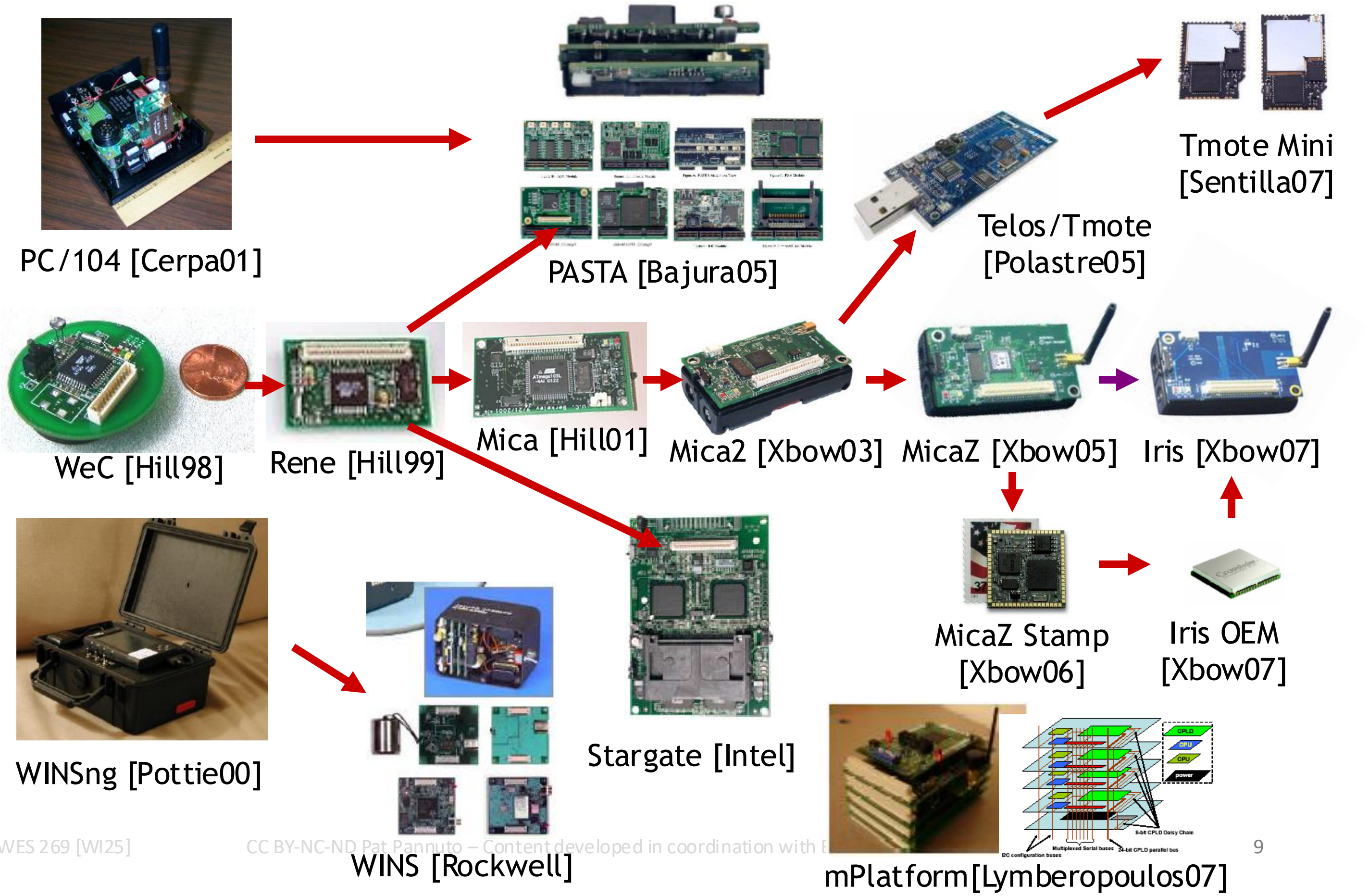
Asynchronous MAC principles

- Communication is possible if one device is receiving while other is transmitting
- *Both* devices want to minimize the time they are receiving or transmitting (i.e. their radio is on) to reduce power draw
- Devices can only coordinate using the data communication channel (i.e. no out-of-band communication)
  - No global synchronization mechanism
- Goal: scheme to schedule TX and RX to permit communication while minimizing energy
- Energy is paramount, but additional metrics:
  - Latency
  - Throughput
  - Reliability
  - Network scale




Early
2000s
POKéMON
Von Dutch
U.S.A.
Videos
Photos
Podcasts
Extras
Settings
Shuffle Songs
Now Playing

Early days of low power sensor nodes…



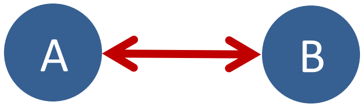
Low power: multiple year operation using batteries

- $P_{node} = P_{CPU} + P_{TX} + P_{RX} + P_{SLEEP}$
- $E_{BATT} = P_{node}$ * lifetime

- Lifetime ⬆, P_{node} ⬇
  - $P_{CPU} \gg P_{SLEEP}$
  - $P_{TX} \gg P_{SLEEP}$
  - $P_{RX} \gg P_{SLEEP}$
  - → Minimize compute, TX, RX
  - → Maximize sleep

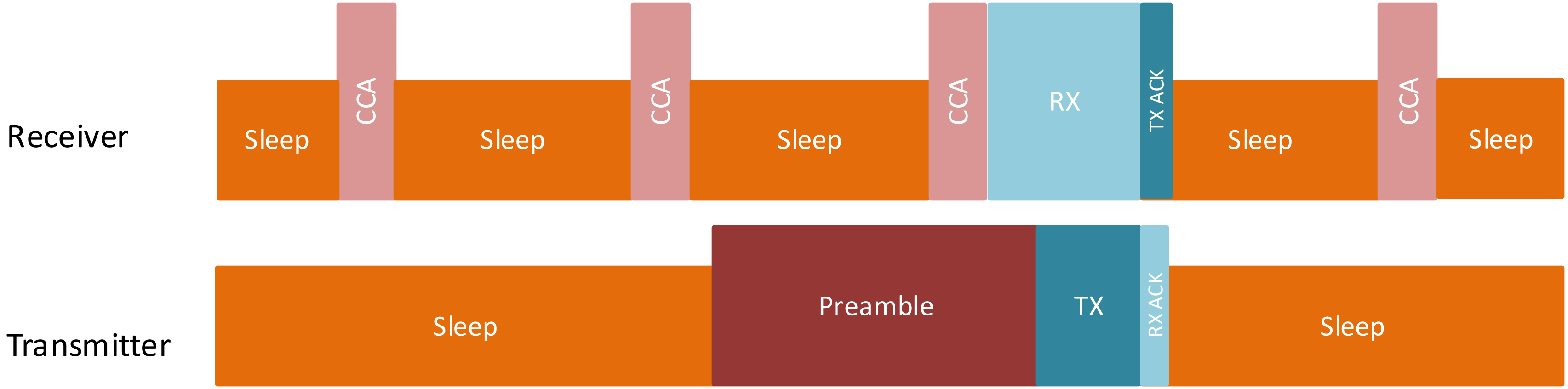
Low power MAC principles

- Communication is possible if one device is receiving while other is transmitting
- Devices can only coordinate using the data communication channel (i.e. no out-of-band communication)
  - No global synchronization mechanism
- Goal: scheme to schedule TX and RX to permit communication while minimizing energy
- Energy is paramount, but additional metrics:
  - Latency
  - Throughput
  - Reliability
  - Network scale



Both devices want to minimize the time they are receiving or transmitting (i.e. their radio is on) to reduce power draw

Low Power Listening (LPL) - B-MAC (2004)



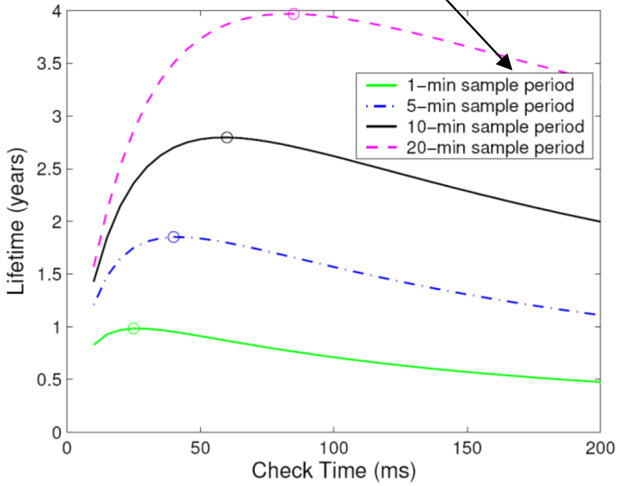
- Method:
  - Receiver periodically samples the channel
  - Transmitter sends a preamble long enough to ensure receiver will detect it
  - Upon detection, receiver stays awake to receive transmitted packet
  - Receiver ACKs if packet received correctly

Versatile Low Power Media Access for Wireless Sensor Networks. Joseph Polastre, Jason Hill and David Culler. SenSys 2004

LPL performance

- CCA check interval
  - Too small: energy wasted on idle listening
  - Too large: energy wasted on transmissions (long preambles)
- In general, it's better to have larger preambles than to check more often!

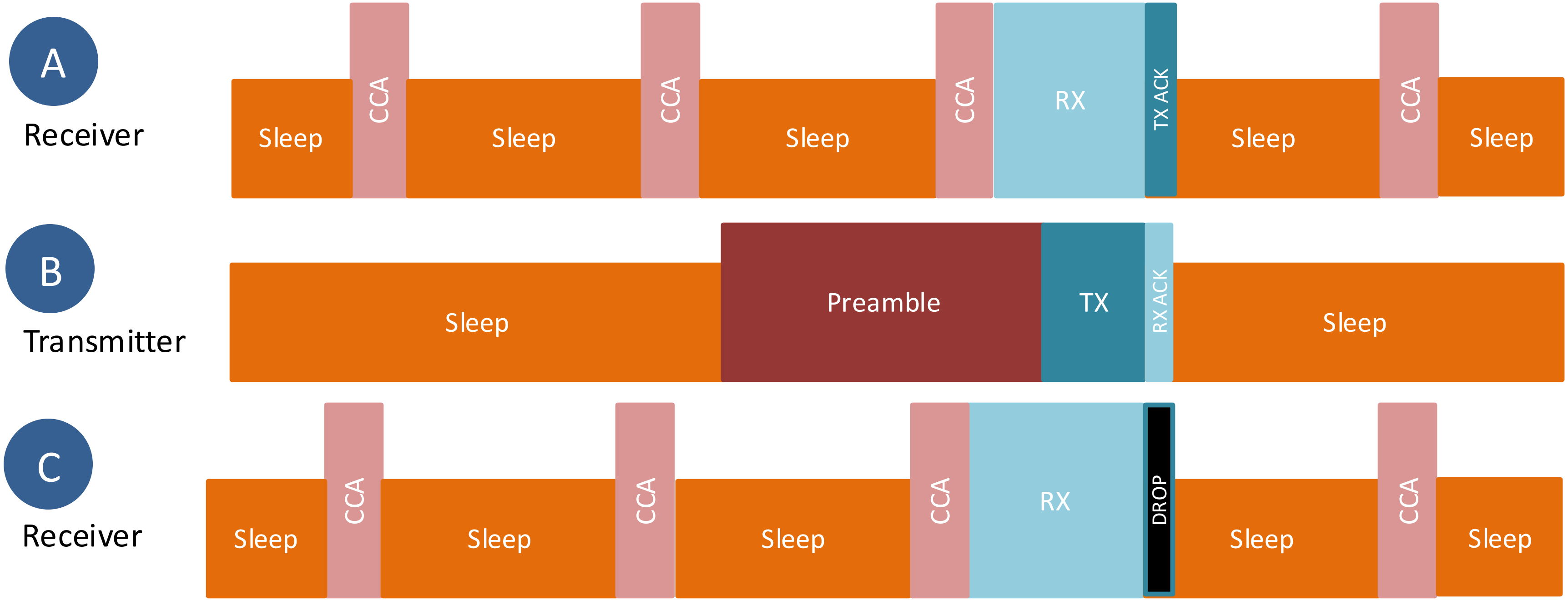
Sensor sample period – controls the packet rate



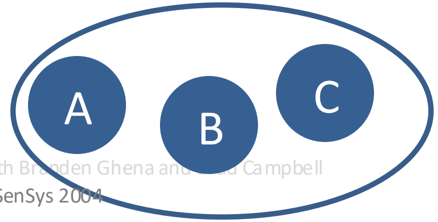
Time between listening
(checking if the channel is idle)

Versatile Low Power Media Access for Wireless Sensor Networks. Joseph Polastre, Jason Hill and David Culler. SenSys 2004

LPL Drawback

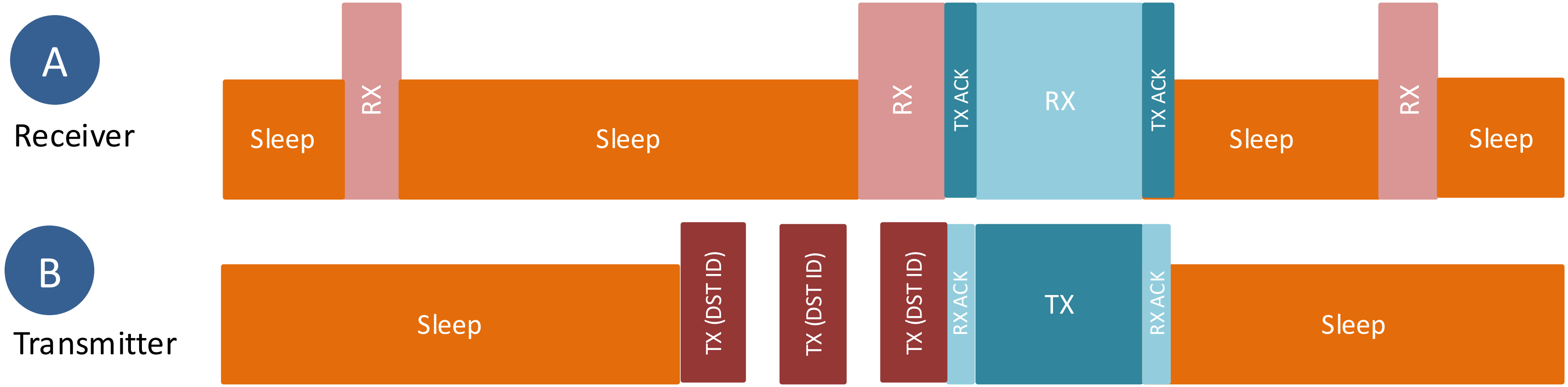


- Spend time listening to packets for someone else!



Versatile Low Power Media Access for Wireless Sensor Networks. Joseph Polastre, Jason Hill and David Culler. SenSys 2004

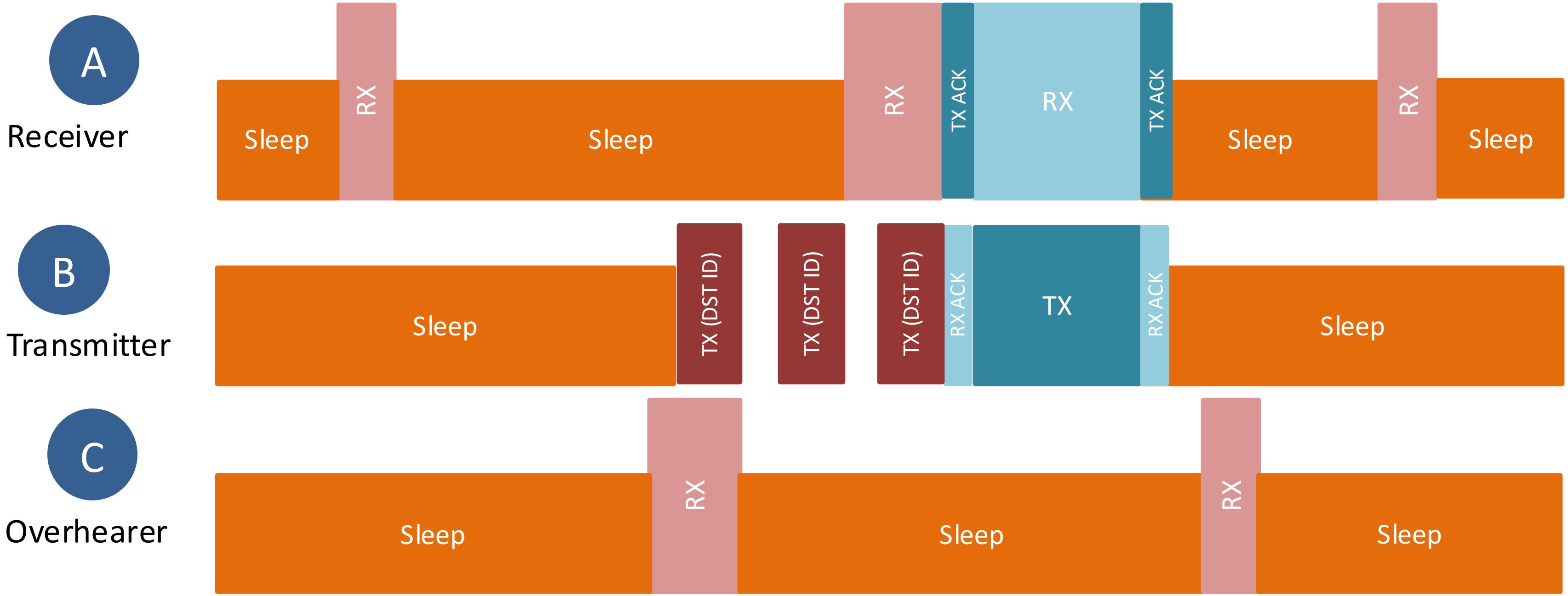
X-MAC: Shorter preambles and destination information



- Method
  - Transmitter sends a short “I have a packet for this address” packet
  - Nodes periodically listen, and ACK if the packet is for them
  - Upon receiving the ACK, the transmitter sends full packet
  - Receiver successfully receives the full packet.

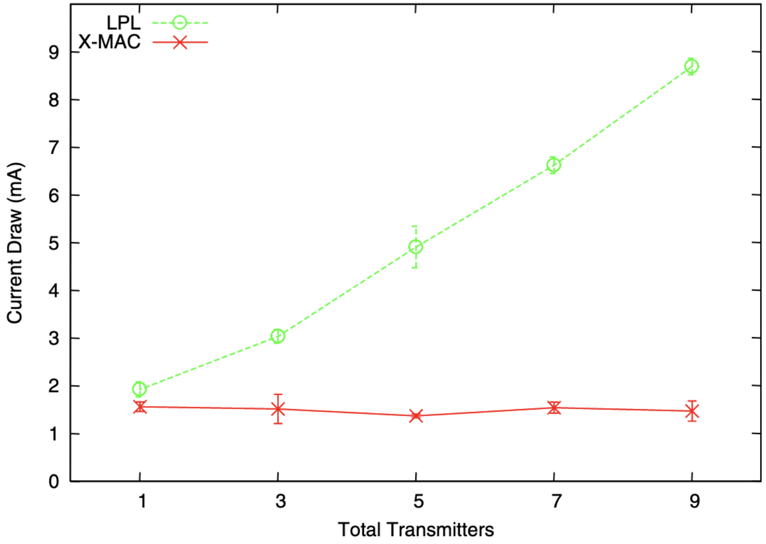
X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. Michael Buettner, Gary V. Yee, Eric Anderson, Richard Han. SenSys 2006.

X-MAC: Overhearing node drops out early



X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. Michael Buettner, Gary V. Yee, Eric Anderson, Richard Han. SenSys 2006.

X-MAC: lower power draw than LPL with multiple nearby transmitters

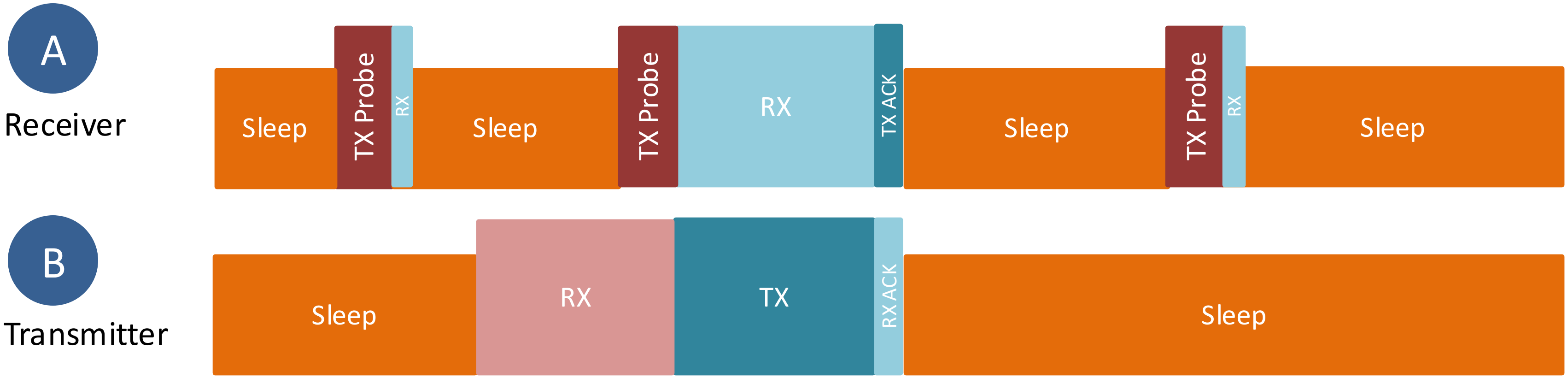


X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. Michael Buettner, Gary V. Yee, Eric Anderson, Richard Han. SenSys 2006.

LPL (B-MAC) and X-MAC are **transmitter initiated MAC protocols**

- Receiving nodes continuously periodically sample the wireless channel
  - When they detect a transmission they listen to receive packets
- Transmitting nodes are only active when they want to transmit
- Receiving nodes prone to unnecessary wakeups
  - Have to receive *all* nearby transmissions to see if the packet happens to be for them
  - Unnecessary wakeups lead to increasing receive energy → shorter lifetime
  - In general, difficult for receiver to know if it should stay awake or go back to sleep

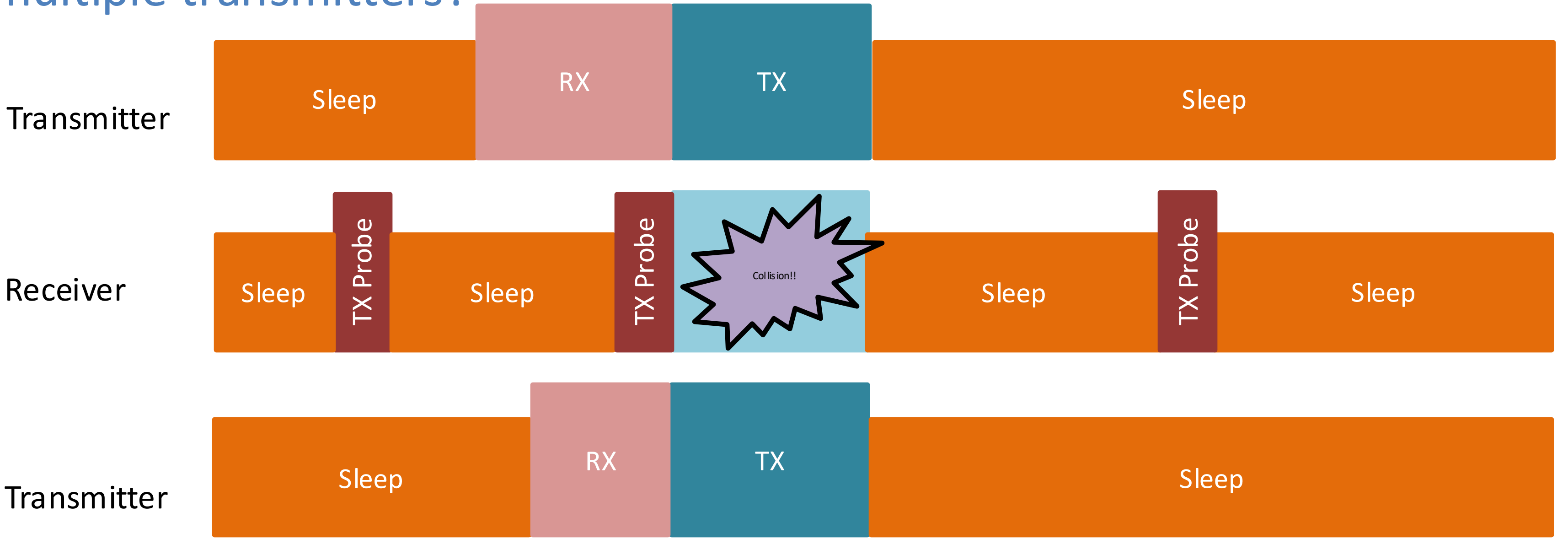
Low Power Probing (LPP) is *receiver* initiated



- Method
  - Receiver periodically transmits a probe, listens afterward
  - Transmitter listens for probe packet from intended recipient
  - Transmitter sends packet after receiving the correct probe packet

RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. Yanjun Sun, Omer Gurewitz, David B. Johnson. SenSys 2008.
Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. Razvan Musaloiu-E., Chieh-Jan Mike Liang, Andreas Terzis. IPSN 2008.

LPP introduces a new challenge for multiple nodes. What happens with multiple transmitters?

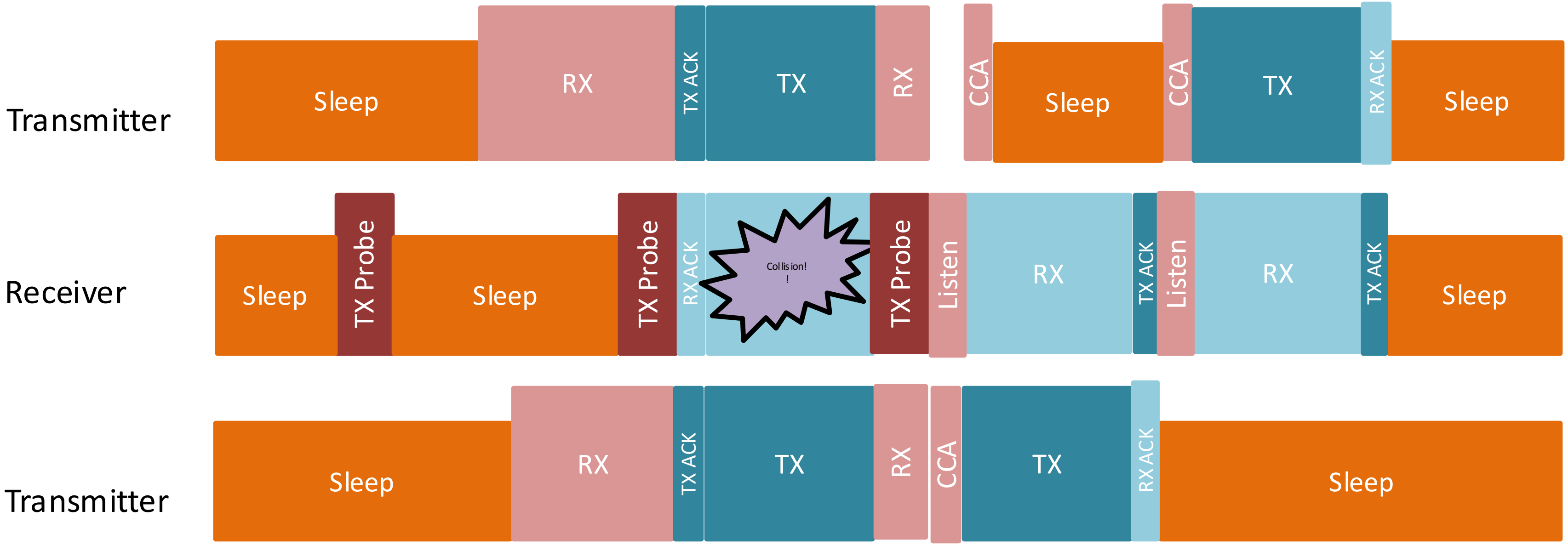


- If multiple transmitters receive a probe, and they both want to transmit, it leads to a collision and a lost packet
- Receiver is unsure if there *actually* was a packet, or just noise/interference on the channel

RI-MAC: A Receiver-Initiated Asynchronous Duty Cycle MAC Protocol for Dynamic Traffic Loads in Wireless Sensor Networks. Yanjun Sun, Omer Gurewitz, David B. Johnson. SenSys 2008.
Koala: Ultra-Low Power Data Retrieval in Wireless Sensor Networks. Razvan Musaloiu-E., Chieh-Jan Mike Liang, Andreas Terzis. IPSN 2008.

A-MAC resolves this with **backcast**

("~~constructive interference~~", i.e., coherent combining)



- Method
  - Transmitters send an ACK in response to a probe packet
  - With multiple transmitters, the ACKs collide, but interfere *non-destructively*, so the receiver still receives an ACK
  - The receiver stays awake to receive a packet, but the packet collides
  - The receiver sends a new probe, informing transmitters there was a collision
  - Transmitters use CSMA/CA to send packets [Recall, random initial backoff 0–7 slots]

Design and Evaluation of a Versatile and Efficient Receiver-Initiated Link Layer for Low-Power Wireless. Prabal Dutta, Stephen Dawson-Haggerty, Yin Chen, Chieh-Jan Mike Liang, and Andreas Terzis. SenSys 2010.

Summary: asynchronous low power MAC protocols

- Transmitter initiated protocols
  - Receivers periodically sample the channel
  - Transmitters transmit sufficiently to ensure recipients heard
  - Examples: LPL, X-MAC
  - Pros:
    - Simple, intuitive
    - Can balance TX and RX energy with sample interval
  - Cons:
    - Receivers prone to false wake-ups
    - High idle listening energy costs

- Receiver initiated protocols
  - Receivers periodically transmit probes
  - Transmitters listen for a probe before sending a packet
  - Examples: RI-MAC, Koala, A-MAC
  - Pros:
    - Fixed, small listening energy cost for receivers
  - Cons:
    - Multiple transmitters leads to collisions

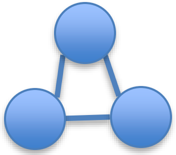
Outline

- Low-Power MAC Design
- **Simple Routing**
- Mesh Routing
- Flooding

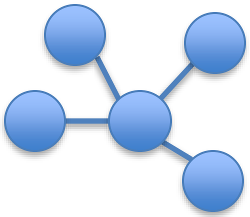
Routing goals

- Have a packet, have a destination, how do we connect them?
- Simple techniques
  - Broadcast, tree structures
- Mesh techniques
  - Understand the available routes and select a “good” one

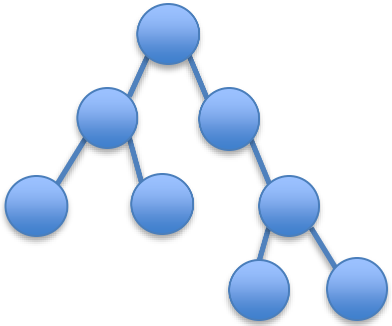
Refresher: Simple routing solutions



- Broadcast
  - The link-layer solution for everything

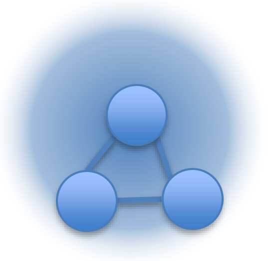


- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.

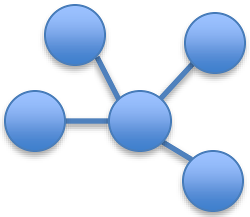


- Tree topology
  - "Star of stars"
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)

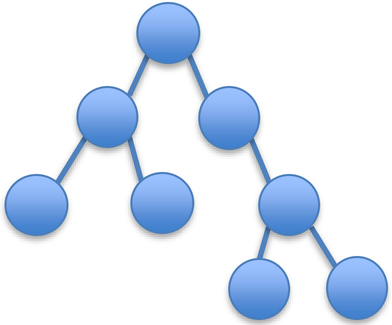
Refresher: Simple routing solutions



- Broadcast
  - The link-layer solution for everything
  - ***Range:*** Limited to one "broadcast domain" (all nodes must be physically close)

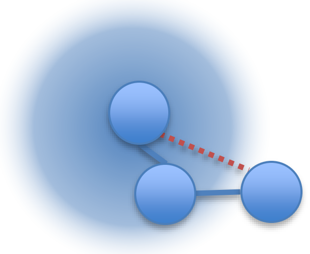


- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.

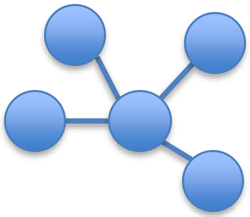


- Tree topology
  - "Star of stars"
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)

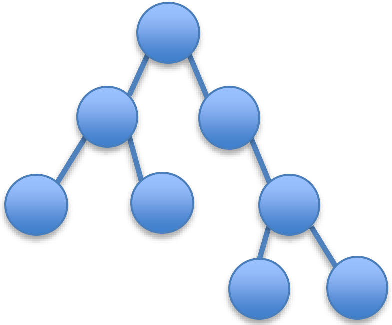
Refresher: Simple routing solutions



- Broadcast
  - The link-layer solution for everything
  - ***Range:*** Limited to one “broadcast domain” (all nodes must be physically close)

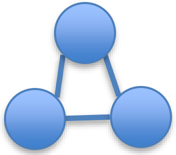


- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.

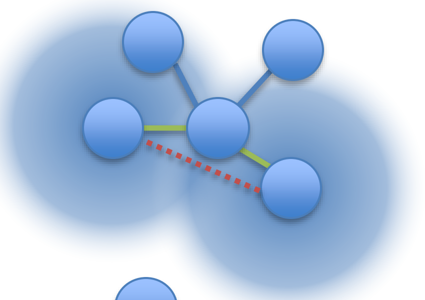


- Tree topology
  - “Star of stars”
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)

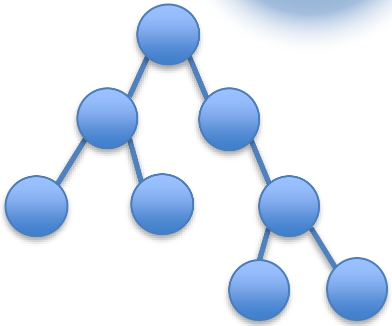
Refresher: Simple routing solutions



- Broadcast
  - The link-layer solution for everything
  - ***Range:*** Limited to one "broadcast domain" (all nodes must be physically close)

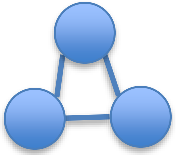


- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.
  - ***Range:*** All nodes must be in range of the parent (but not necessarily each other)

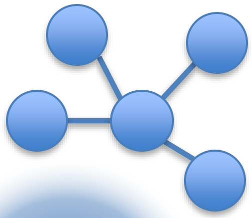


- Tree topology
  - "Star of stars"
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)

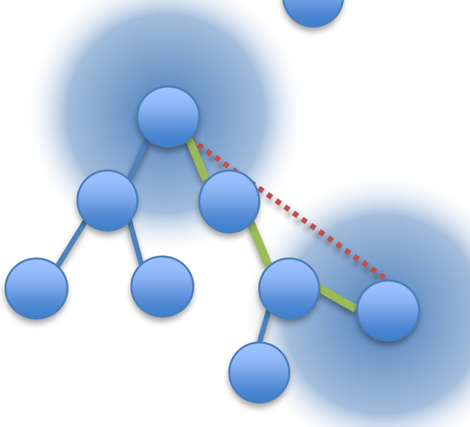
Refresher: Simple routing solutions



- Broadcast
  - The link-layer solution for everything
  - ***Range:*** Limited to one “broadcast domain” (all nodes must be physically close)

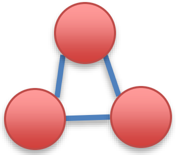


- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.
  - ***Range:*** All nodes must be in range of the parent (but not necessarily each other)

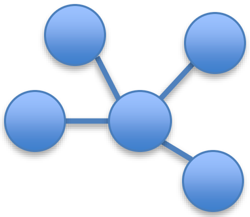


- Tree topology
  - “Star of stars”
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)
  - ***Range:*** Works as long as a node is within one hop of any other node

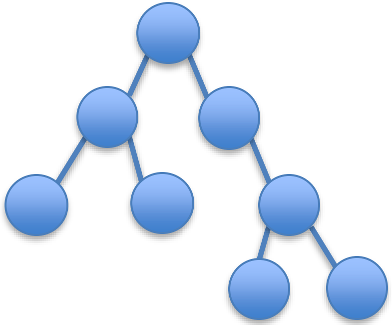
Refresher: Simple routing solutions



- Broadcast
  - The link-layer solution for everything
  - ***Range:*** Limited to one “broadcast domain” (all nodes must be physically close)
  - ***Burden:*** Naïve version assumes everyone is always listening

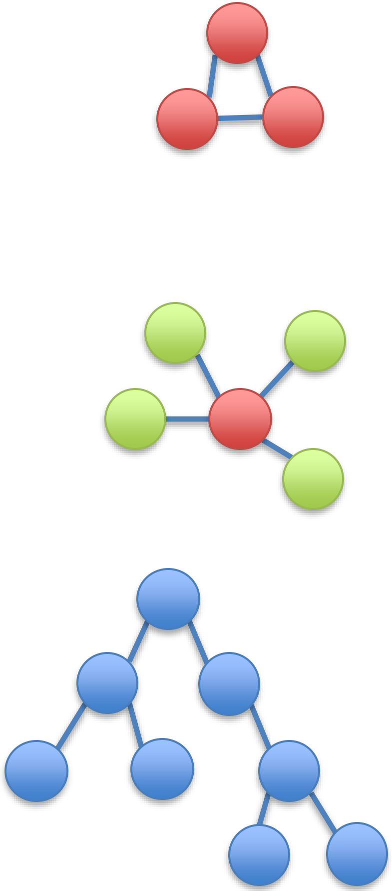


- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.
  - ***Range:*** All nodes must be in range of the parent (but not necessarily each other)



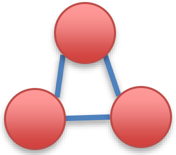
- Tree topology
  - “Star of stars”
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)
  - ***Range:*** Works as long as a node is within one hop of any other node

Refresher: Simple routing solutions

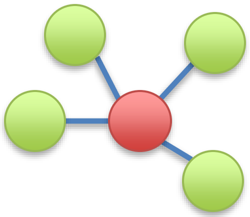


- Broadcast
  - The link-layer solution for everything
  - ***Range:*** Limited to one “broadcast domain” (all nodes must be physically close)
  - ***Burden:*** Naïve version assumes everyone is always listening
- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.
  - ***Range:*** All nodes must be in range of the parent (but not necessarily each other)
  - ***Burden:*** The parent must participate in every packet; listen often for new arrivals
- Tree topology
  - “Star of stars”
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)
  - ***Range:*** Works as long as a node is within one hop of any other node

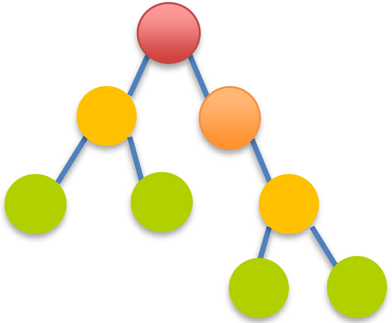
Refresher: Simple routing solutions



- Broadcast
  - The link-layer solution for everything
  - ***Range:*** Limited to one “broadcast domain” (all nodes must be physically close)
  - ***Burden:*** Naïve version assumes everyone is always listening



- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.
  - ***Range:*** All nodes must be in range of the parent (but not necessarily each other)
  - ***Burden:*** The parent must participate in every packet; listen often for new arrivals



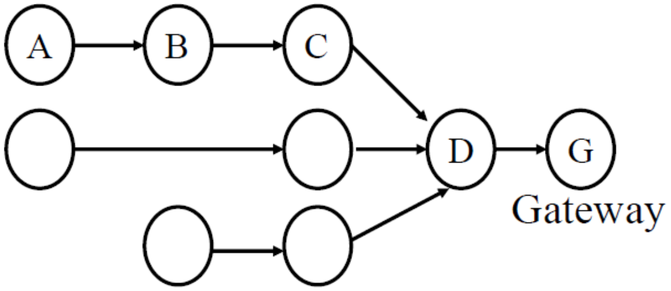
- Tree topology
  - “Star of stars”
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
    - Original ZigBee approach (knowledge built into addressing scheme)
  - ***Range:*** Works as long as a node is within one hop of any other node
  - ***Burden:*** “Higher” parents see more traffic; everyone must listen for new arrivals

Many variations, e.g., “Many-to-one routing”

e.g. Collection Tree Protocol (CTP)

- Tree optimization for sensor networks
  - Keep all devices except the “gateway” as simple as possible

- Each device only needs to remember hop to gateway
  - If gateway wants to send message back, it must include a full hop path

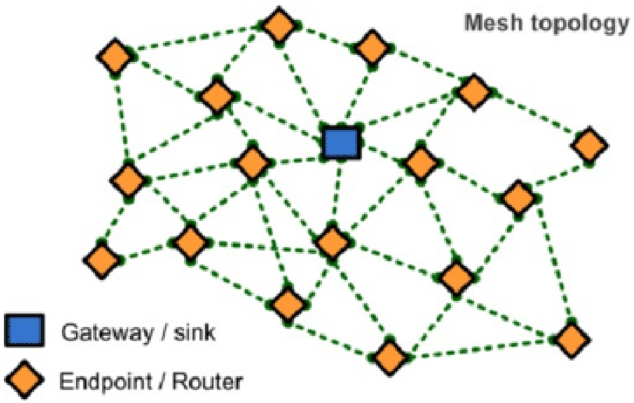


Outline

- Simple Routing
- **Mesh Routing**
- Efficient Flooding (Synchronous Transmissions)

Refresher: Mesh networks

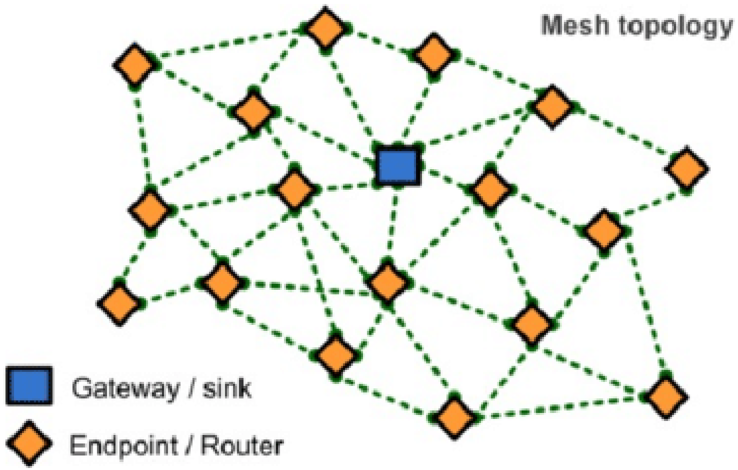
- Most devices can communicate with multiple neighbors
- What are advantages of mesh?
  - Devices can communicate over longer distances
  - Device failures less likely to collapse the entire network
- What are disadvantages of mesh?
  - Some nodes must spend more energy communicating
  - Network protocol becomes more complicated to manage routing



Enabling full mesh networks

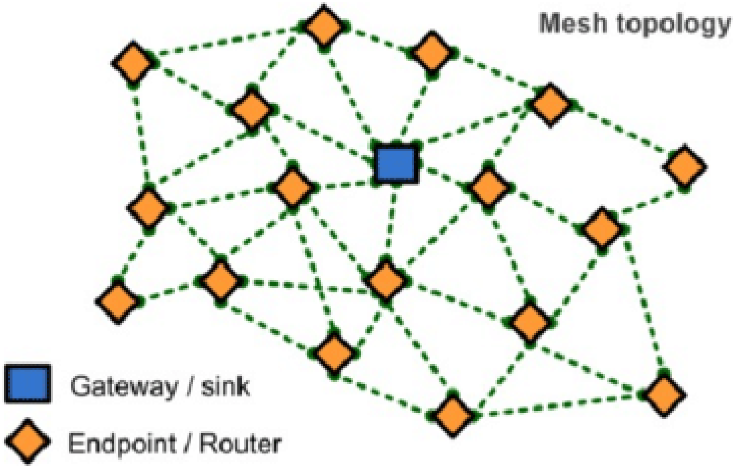
- All nodes:
  - Can send and receive data
  - Route and forward data in the network
- Nodes connect with their neighbors
- Gateway provides access to other networks/internet
- We're interested in meshes that
  - Use low power nodes (battery powered)
  - Are low data rate with small packets
- 802.15.4 provides a foundation to build mesh on top of:

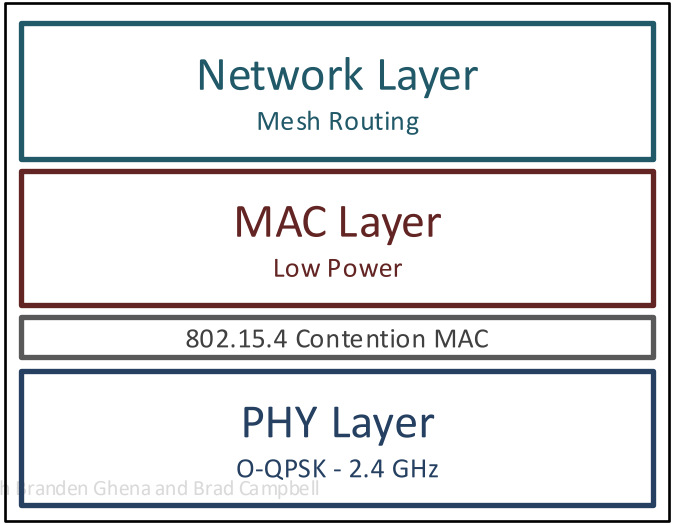
Contention Access Period



Mesh network layers

- PHY Layer
  - 802.15.4 physical layer
- MAC Layer: 802.15.4
  - 802.15.4 spec gives us the option of implementing our own
  - Technically this layer is here, but we generally ignore it
- MAC Layer
  - Need to allow to TWO low power devices to communicate
  - Devices are off most (~99%) of the time
  - No assumption of energy asymmetry (as with PAN coordinator or BLE)
- Network Layer: Mesh
  - Handles organizing the network
  - How should packets travel from nodes to the gateway?
  - How should packets travel from the gateway to nodes?
  - How should packets travel from node to node?





Quantitative intuition of 'why bother meshing'

- Free Space Path Loss (FSPL) and the Friis transmission equation
  - Measure how RF signals travel through space
  - $FSPL = 20 * \log_{10}(\frac{4\pi R}{\lambda})$, measured in dB
  - $P_{RX} = P_{TX} * \frac{G_{TX} * G_{RX} * \lambda^2}{(4\pi R)^2 * L}$, measured in W [Friis]
  - $P_r = P_t + G_t + G_r + 20\log_{10}(\frac{\lambda}{4\pi d})$, Friis, re-written in dB

Traditionally written as transmit **d**istance

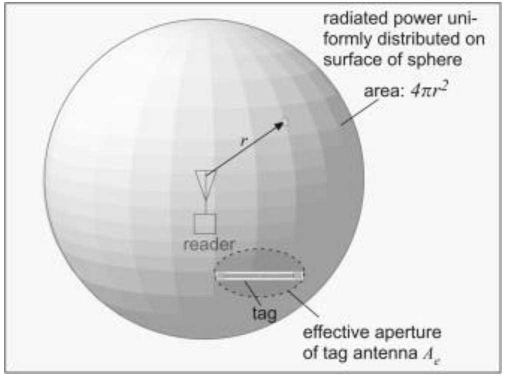


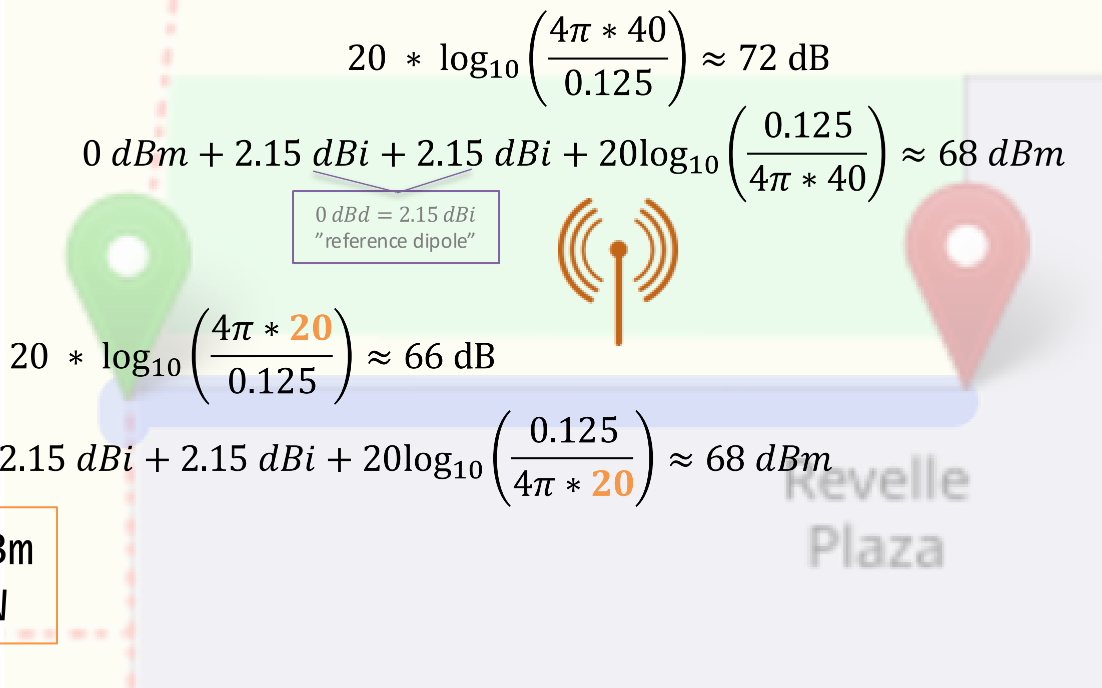
Image: https://www.sciencedirect.com/topics/computer-science/transmitting-antenna

A hand-wavy quantitative analysis of ‘why bother meshing’

Revelle Plaza is ~40m wide; how to get from one side to the other?

$$\frac{3e8\ m/s}{2.4e9\ Hz} = 0.125\ m = \lambda$$

```
0 dBm
1 mW
```



$$20 * \log_{10}\left(\frac{4\pi * 40}{0.125}\right) \approx 72\ \text{dB}$$

$$0\ dBm + 2.15\ dBi + 2.15\ dBi + 20\log_{10}\left(\frac{0.125}{4\pi * 40}\right) \approx 68\ dBm$$

$0\ dBd = 2.15\ dBi$
"reference dipole"

$$20 * \log_{10}\left(\frac{4\pi * 20}{0.125}\right) \approx 66\ \text{dB}$$

$$\_\ dBm + 2.15\ dBi + 2.15\ dBi + 20\log_{10}\left(\frac{0.125}{4\pi * 20}\right) \approx 68\ dBm$$

```
-5.2 dBm
 0.3 mW
```

```
-72 dBm (6.3e-8 mW)
-68 dBm (1.6e-7 mW)
```

Multi-hop mesh…

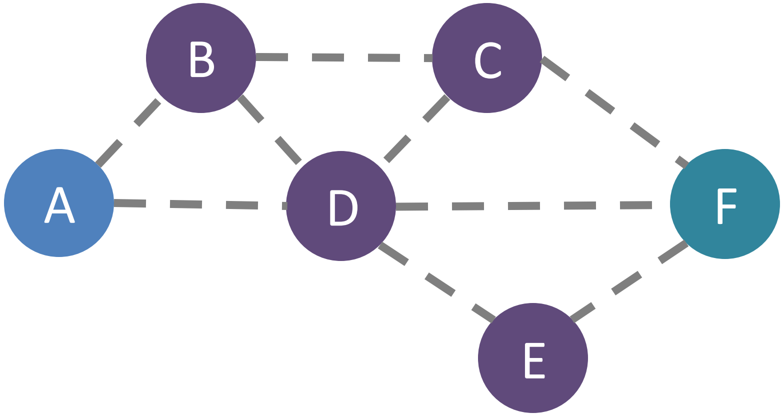
- Reduces TX power
  - But adds RX, sync cost!
- Improves aggregate network coverage
- Can improve robustness
- Reduces collision domain

Outline

- Simple Routing
- **Mesh Routing**
  - Motivation
  - **Example Protocols**
- Efficient Flooding (Synchronous Transmissions)

Mesh Routing

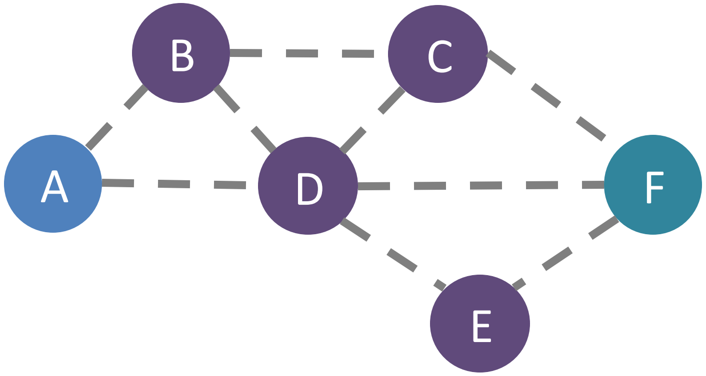
- Mesh topology makes routing question more complicated
  - Multiple hops in a route
  - Multiple routes between source and destination
  - Becomes a graph theory question based on cost metric



Reactive routing

- Build up a map of the routes through a network
  - Hopefully the “optimal” routes

- Map routes in reaction to a packet arrival
  - Sensor devices are slow and limited
  - Most likely to resend to same prior address
  - Discover a route when it is needed, then cache for next time



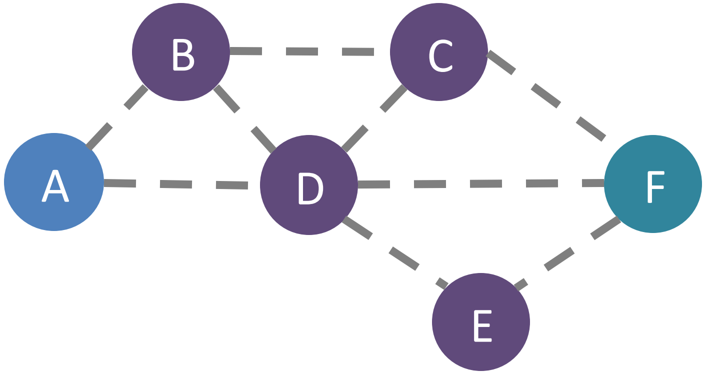
B Neighbor Table	
A	c7:e3:45:00:1f:e4
C	c7:e3:45:ab:44:c6
D	54:99:10:aa:b1:c3

B Forwarding Table	
DST	**Next Hop**
F	C
A	A

Neighbor discovery

- Which nodes are nearby to me?
  - i.e. within one packet transmission range
- General Techniques
  - Periodically listen for nearby transmissions
    - Overhear packets
    - Source addresses indicate neighbors
  - Periodically send neighbor discovery probes
    - Announce your existence
    - Ask for neighbors to respond
  - Centrally collect neighbor information
    - Gateway disseminates to nodes as needed

B Neighbor Table	
A	c7:e3:45:00:1f:e4
C	c7:e3:45:ab:44:c6
D	54:99:10:aa:b1:c3



Ad-hoc On-demand Distance Vector Routing (AODV)

- On-demand: Construct routes only when needed
- Modern ZigBee routing approach (for Mesh topology)

- Routing table
  - Destination node -> Next hop (for all cached destinations)
  - Store only next hop instead of full route
    - All routers along the path must also have Destination->Next mappings
  - Also keep hops-to-destination and last-seen-destination-sequence-number

- Route discovery
  - Upon demand: check table
  - If not cached send Route Request (RREQ) via Flooding
    - Route is unknown, so flooding is needed

AODV Route Requests (RREQs)

Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count

- Request ID identifies this RREQ
  - Used to discard duplicates during flooding

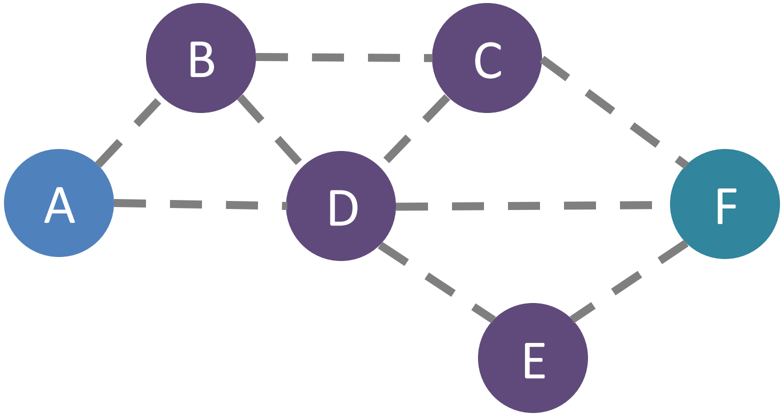
- Sequence Numbers are per-device, monotonically increasing
  - Used as a notion of "how recently" device has been seen
  - Source SeqNo is the source's most recent sequence number
  - Destination SeqNo is the most recently seen from the destination by the source. (Defaults to zero)

- Hop Count is the number of hops this request has taken
  - Starts at 1 and incremented by each transmitter along the path

Example AODV RREQ (A to F)

Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count

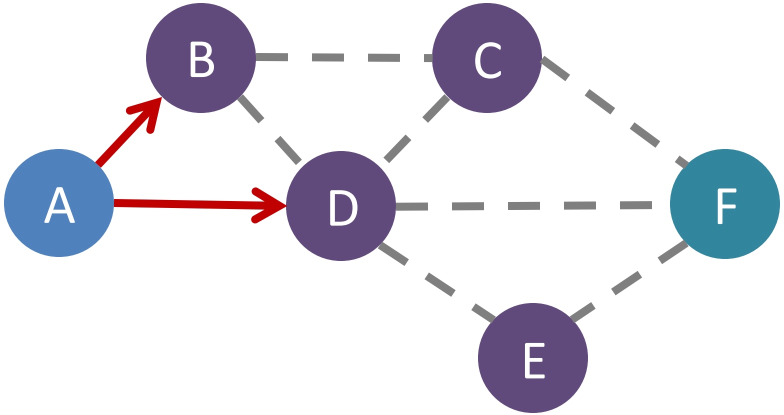
A wants to find a route to F, so it sends out an RREQ



Example AODV RREQ (A to F)

Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count
1	A	F	1	0	1

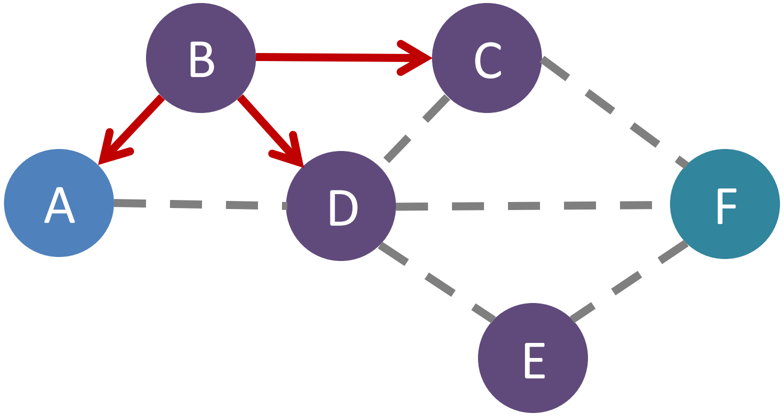
B and D also opportunistically add a routing table entry for A



Example AODV RREQ (A to F)

Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count
1	A	F	1	0	2

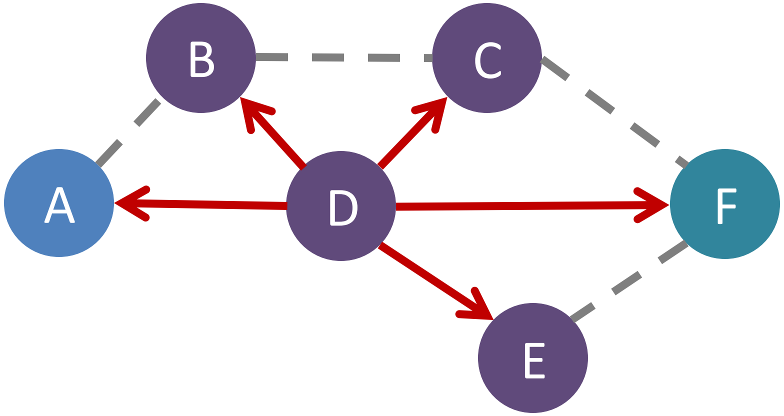
B goes first via some access control protocol (D also in contention)
A and D ignore duplicate Request ID
C opportunistically adds a routing table entry to A



Example AODV RREQ (A to F)

Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count
1	A	F	1	0	2

D goes next by some access control protocol (C also in contention)
A, B, and C ignore duplicate Request ID
E and F opportunistically adds a routing table entry to A

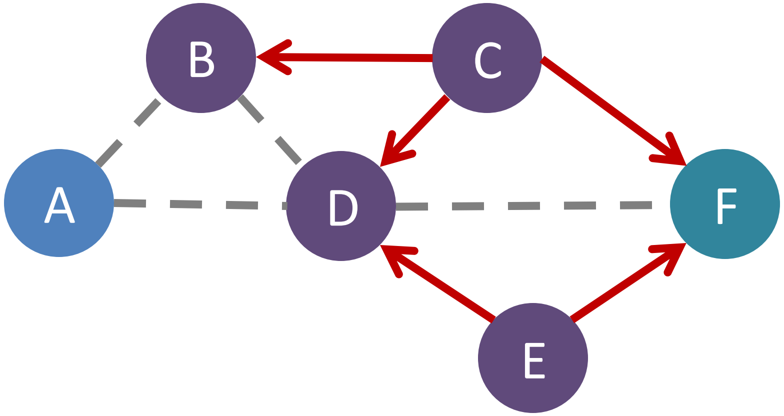


Example AODV RREQ (A to F)

Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count
1	A	F	1	0	3

C and E repeat this process with Hop Count 3 (but everyone ignores them)

- They go one-at-a-time, but I'm getting tired of drawing these
- Actually, they're in contention with the response from F



AODV Route Response (RREP)

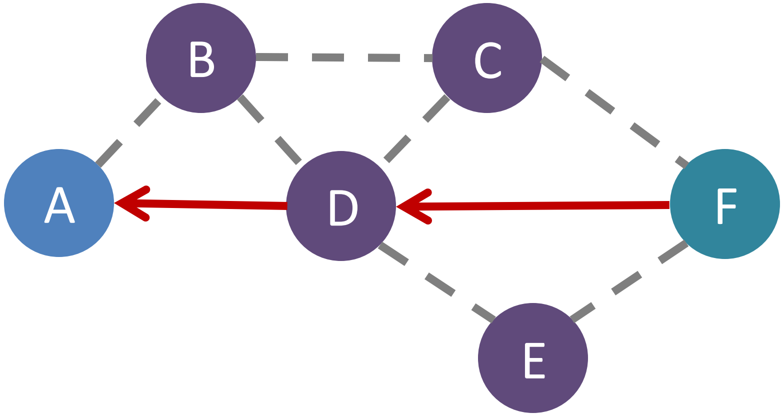
Source Address	Destination Address	Destination SeqNo	Hop Count

- Reply is sent unicast back to the source via newly constructed route
  - Each node along the way already knows the route back
- Includes most recent destination sequence number as a sense of recency
  - No need for source sequence number anymore

Example AODV RREP (F to A)

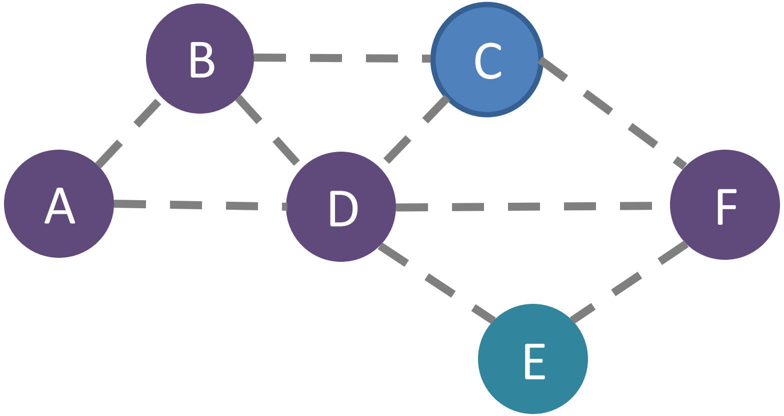
Source Address	Destination Address	Destination SeqNo	Hop Count
F	A	7	2

F sends response back to A via D
D opportunistically adds a routing table entry for F



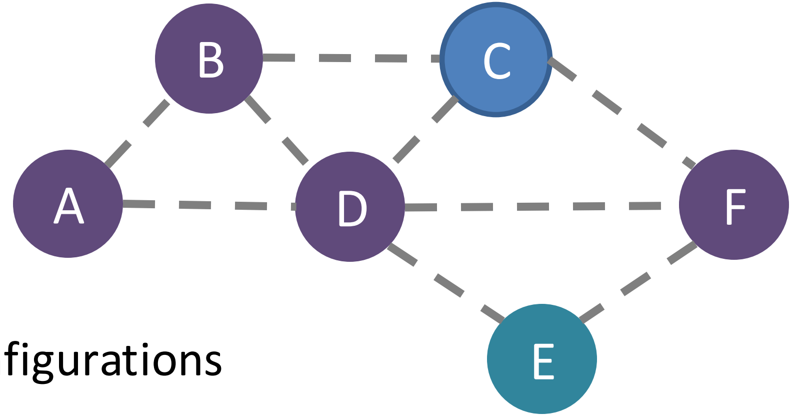
Break + Practice

- C wants to send a packet to E
  - What RREQ(s) are sent and what RREP(s) are sent?



Break + Practice

- C wants to send a packet to E
  - What RREQ(s) are sent and what RREP(s) are sent?
    - RREQs:
      - C -> (B,D,F)
      - (B or D) -> A
      - (D or F) -> E
    - RREPs:
      - E -> (D or F) -> C
  - Network could have multiple configurations



Route maintenance in AODV

- If a link in the routing table breaks, all active neighbors are informed with Route Error (RERR) messages
  - After some number of retransmissions and timeouts
  - RERR contains destination address that broke
- Nodes receiving RERR can start RERQ for destination address
  - Which lets them find a new path through the network
  - And updates everyone's cached next-hops

Dynamic Source Routing (DSR)

- Another reactive routing technique
  - Similar design as AODV
- In DSR, routing tables have full route to destination
  - Each packet transmission includes a list of hops to destination
  - So the route to an important destination only has to be stored on the source device that cares about it
  - Intermediate nodes do not need any route storage for that destination
    - Cost is extra bytes used in each packet for route
- During discovery, all paths are returned by destination
  - So source gets a full list of possible route choices

Tradeoffs for reactive routing

- Upside: no transmissions unless there is demand
  - Routes might appear, disappear, reappear, but no need to update if no one actually wants to transmit anything

- Downside: large, variable delay when actually sending a packet
  - Full RREQ/RREP protocol before data can actually be sent
  - Route might have broken at some point
    - So data will be sent based on cached information
    - RERR will occur
    - RREQ/RREP will occur
    - Then data will be sent again

Proactive routing

- Alternative to reactive is to know the routes ahead of time
- Periodically query for the possible routes in the network
  - Save all routes that are important (maybe just all routes?)
  - Also redetermine routes whenever topology changes (nodes join/leave)
- Upside: when a packet arrives, route to destination is already known
- Downside: requires more memory and effort on part of routers
  - Wastes some network bandwidth on checking for route changes

Distance-Vector

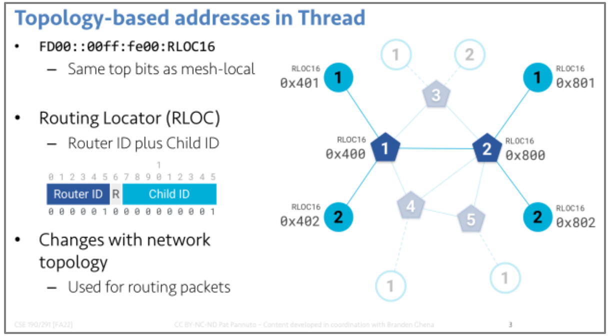
- Keep routes as “next hops” rather than full routes
  - AODV uses this method (DV for Distance Vector)

- Can be combined with proactive techniques too
  - Each router periodically informs neighbors of its shortest paths to each destination (in terms of hop count)
    - Essentially just broadcast your routing table
  - Routers choose the best route available
    - Old next-hop it was already aware of
    - New next-hop through neighbor (with cost of their hops + 1)

  - Need to be careful to avoid loops!

Preview: Thread routing

- Uses a proactive, distance-vector protocol for unicast routing
- If node is a child, send packet to parent router
- If node is a router,
  - Consult table for address within mesh
    - (RLOC helps here!)
  - Send to border router for address outside of mesh
- Multicast uses a data dissemination protocol ([Trickle](#))
  - Or falls back to flooding

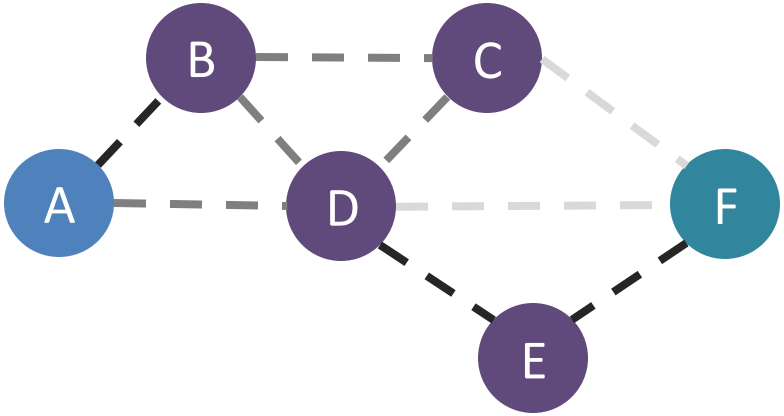


Break + Discussion

- Hop count is one possible metric for determining routes
- What else might be considered?

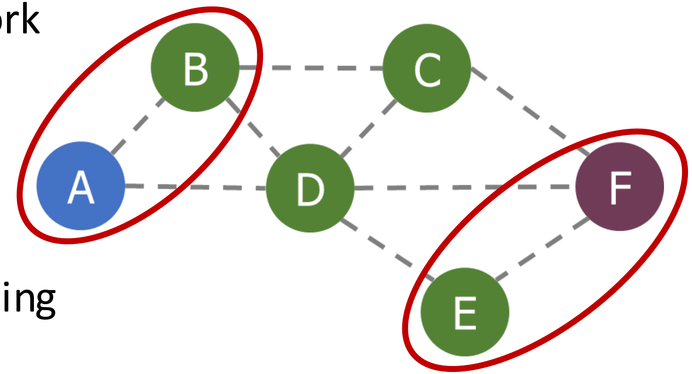
Reliability as a cost metric

- Link quality can very from node to node
  - Fewest hops might not be the “fastest” or “most reliable” path
- ETX: minimize “expected transmissions”
  - Measure link quality over time to determine each link’s reliability



Alternative cost metrics

- Spatial reuse
  - Prefer transmission on links that do not interfere with each other
  - Improves ability to pipeline data through network
  - Example: A<->B and E<->F



- Energy availability
  - Prefer routing through nodes with more remaining available energy
  - Prefer wall-powered nodes over battery-powered
- Arbitrarily complex combinations possible

Outline

- Simple Routing
- Mesh Routing
- **Efficient Flooding (Synchronous Transmissions)**

Traditional flooding is a ’mesh-local broadcast’

- Goal: get information to all nodes
  - This is the problem of “data dissemination”
- Problem: difficult in Mesh topologies
  - Packet loss, retransmission delays
- Really, the desire for data dissemination is just to broadcast to all nodes
  - But broadcast transmissions don’t reach far enough to cover entire mesh

Glossy: What if we expand broadcast range by having multiple nodes participate?

Efficient Network Flooding and
Time Synchronization with Glossy

Federico Ferrari, Marco Zimmerling, Lothar Thiele, Olga Saukh

Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland

IPSN 2011
April 12, 2011, Chicago, IL, USA

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

NCCR MICS
National Competence Center in Research
Mobile Information and Communication Systems

nano-tera.ch

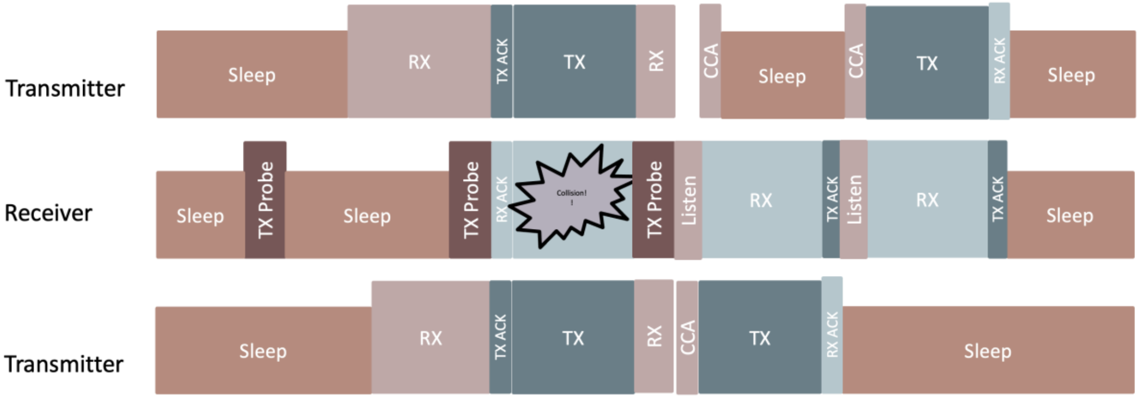
TIK
Institut für Technische Informatik und Kommunikationsnetze
Computer Engineering and Networks Laboratory

Glossy foundations: A-MAC and Backcast

- A little while back, we mentioned that A-MAC allowed (relied on!) “interfering packets”
- I.e., concurrent transmissions of ACKs

A-MAC resolves this with **backcast**

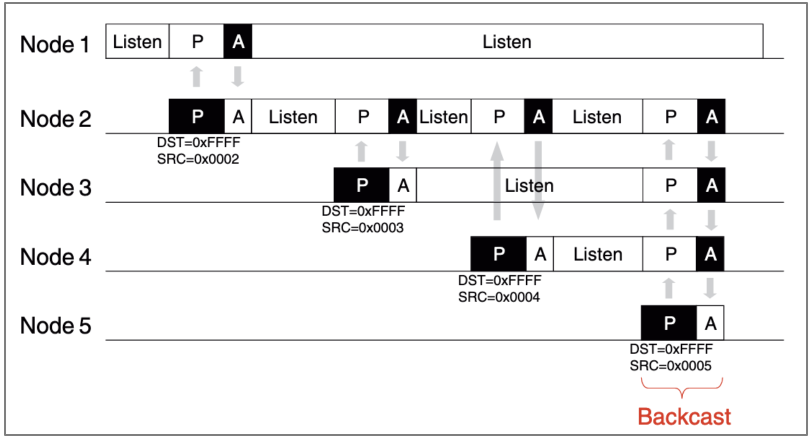
(“~~constructive interference~~”, i.e., coherent combining)



- Method
  - Transmitters send an ACK in response to a probe packet
  - With multiple transmitters, the ACKs collide, but interfere *non-destructively*, so the receiver still receives an ACK
  - The receiver stays awake to receive a packet, but the packet collides
  - The receiver sends a new probe, informing transmitters there was a collision
  - Transmitters use CSMA/CA to send packets [Recall, random initial backoff 0–7 slots]

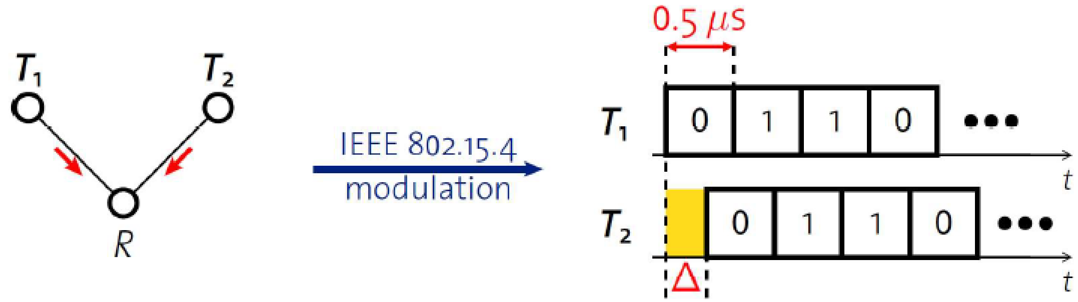
Fundamental Insight: Say exactly the same thing at exactly (enough) the same time → coherence not interference

- In 802.15.4, an ACK is sent *exactly* 192 μs after receiving
  - The ACK packet contains *only* the sequence number being ACK'd
  - *Does not* contain source address of the ACK-er
  - Which is why this works:



Synchronous transmissions extend this concept beyond ACKs to arbitrary packets (with data)

- Multiple nodes transmit **same packet** at **same time**

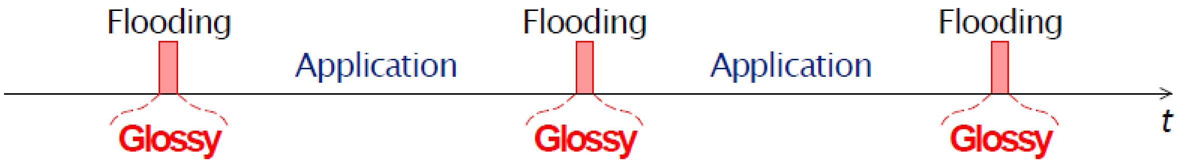


- R can receive packet with high probability if Δ is small
  - May even improve probability of reception (more energy at receiver)

- 500 ns is 1/32 of a symbol for 802.15.4 (chip duration)

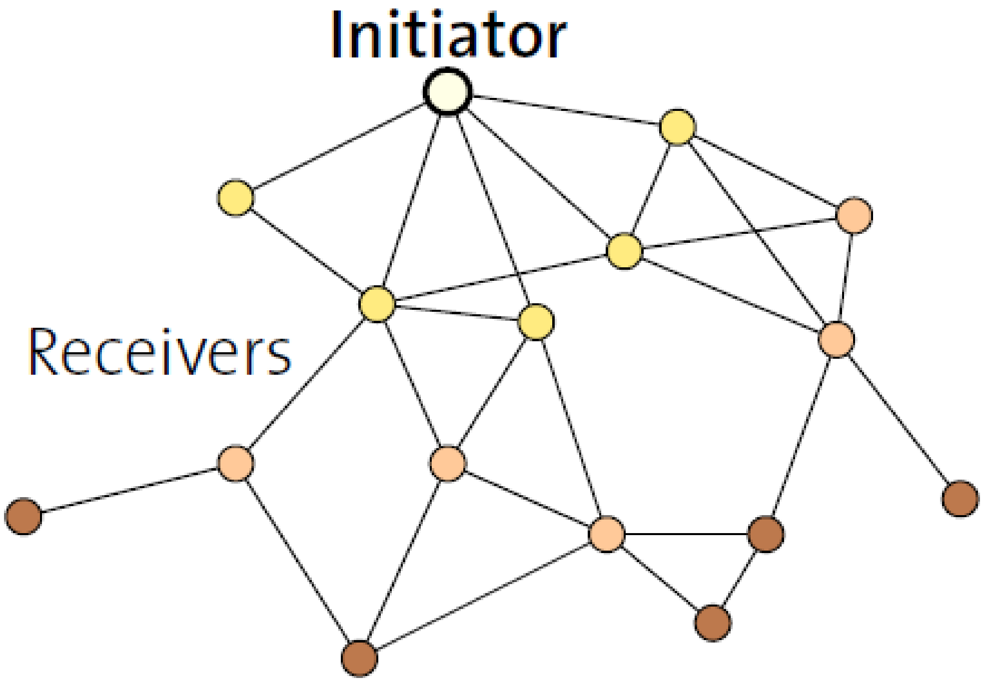
Glossy key techniques

- Temporally decouple network flooding from application tasks



- Exploit synchronous transmissions for fast network flooding

Fast packet propagation in Glossy



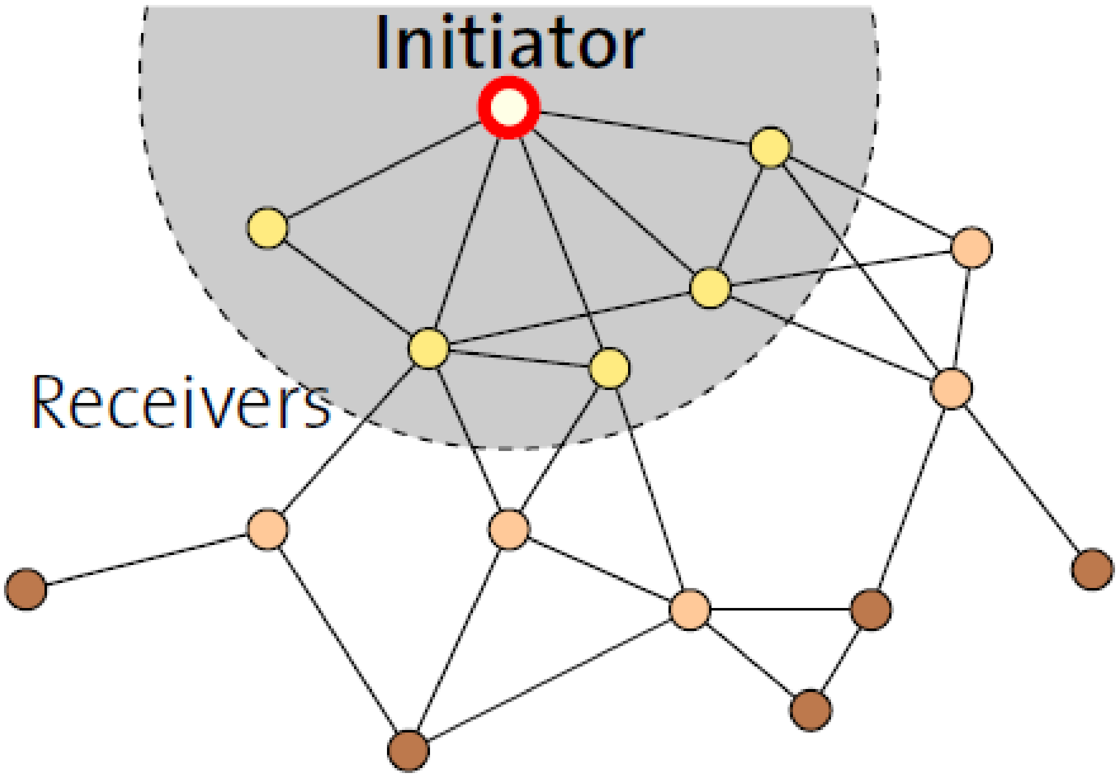
Intuition behind flooding…

GO!

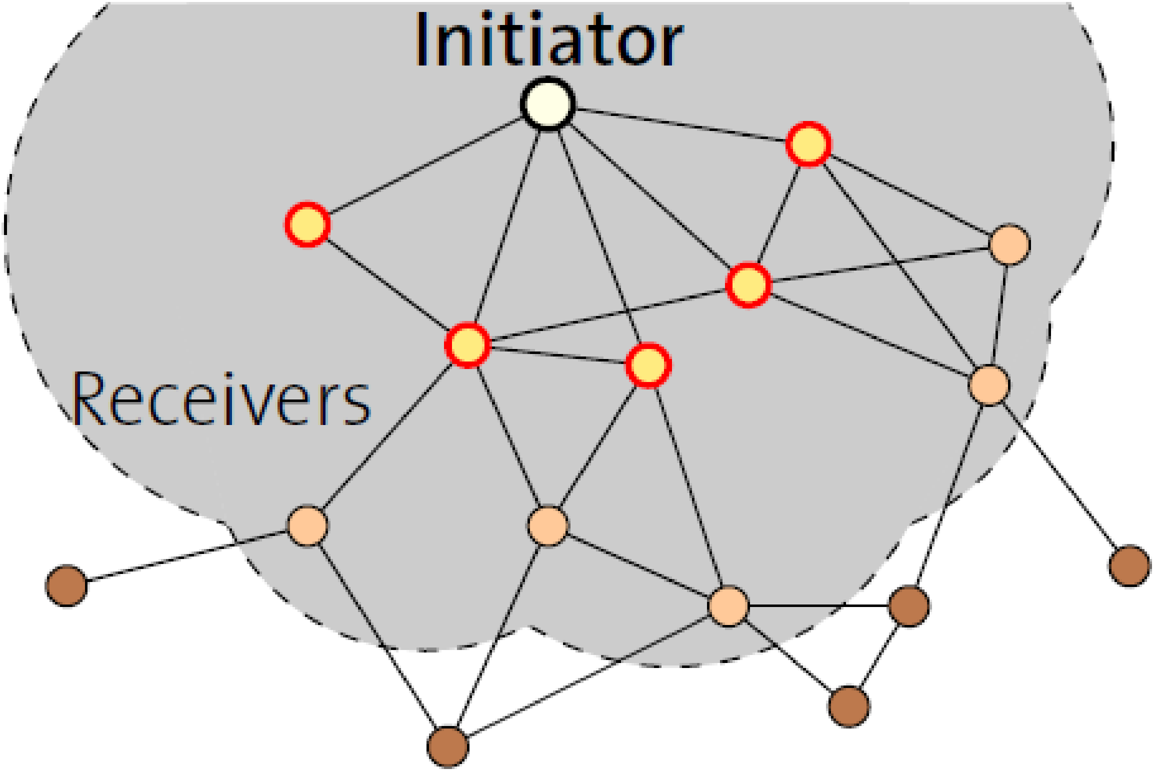
TRITONS!
TRITONS!
UC San Diego
TRITONS
THE WEST
UC San Diego


GO!
GO!
GO!
UCSanDiego
TRITONS
THE WEST
UCSanDiego

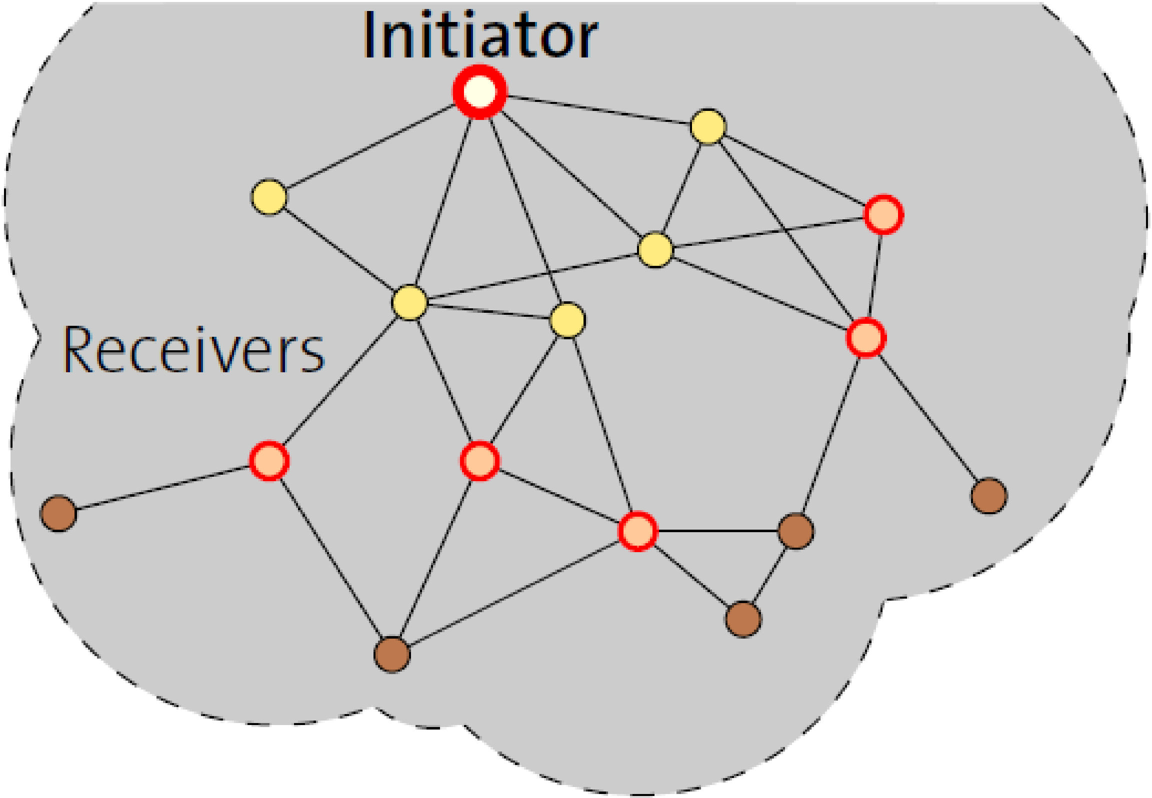
Fast packet propagation in Glossy



Fast packet propagation in Glossy

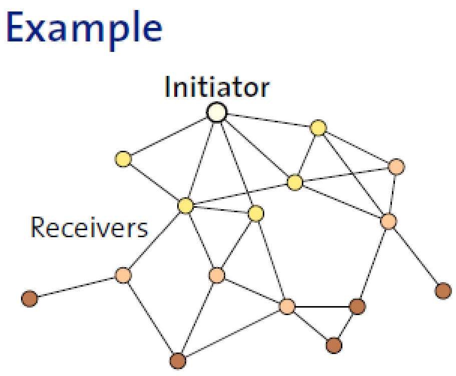


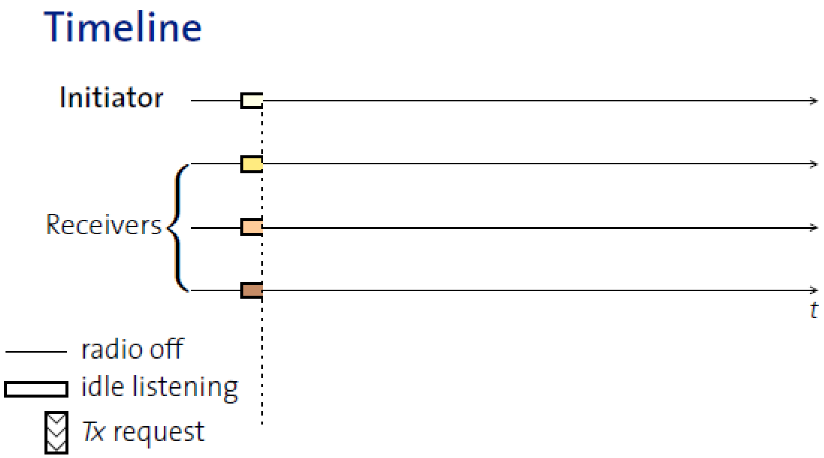
Fast packet propagation in Glossy



Glossy details

- When Glossy starts
  - All nodes turn on radios to receive

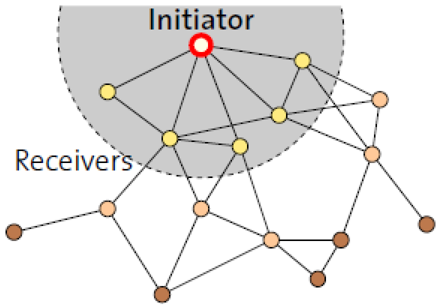




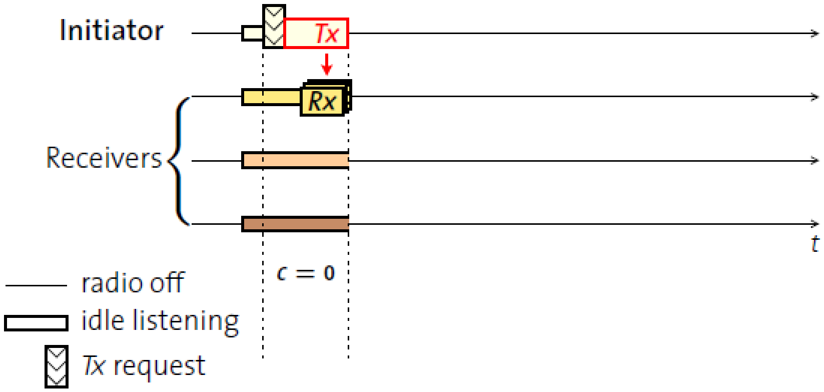
Glossy details

- Initiator
  - Set relay counter in packet, **C** = 0
  - Broadcast packet

Example

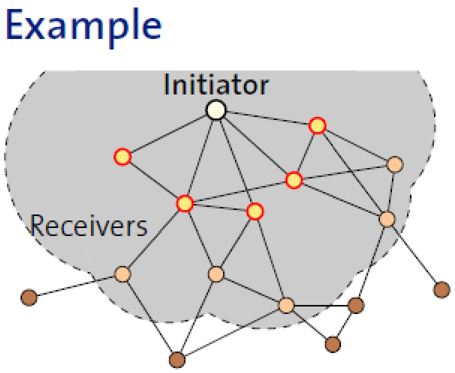


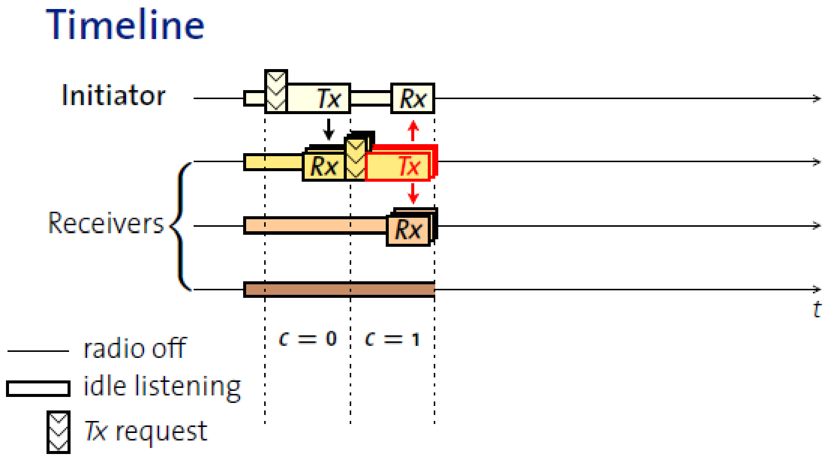
Timeline



Glossy details

- At packet reception:
  - Increment relay counter **C**
  - Transmit synchronously (at a fixed period after the reception)

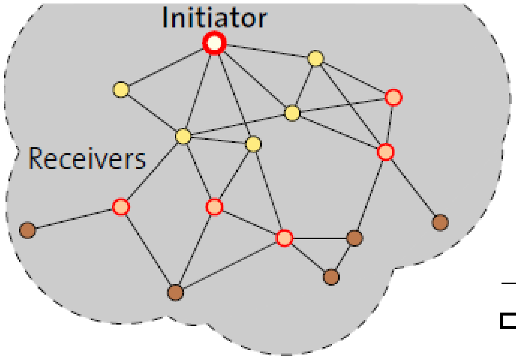




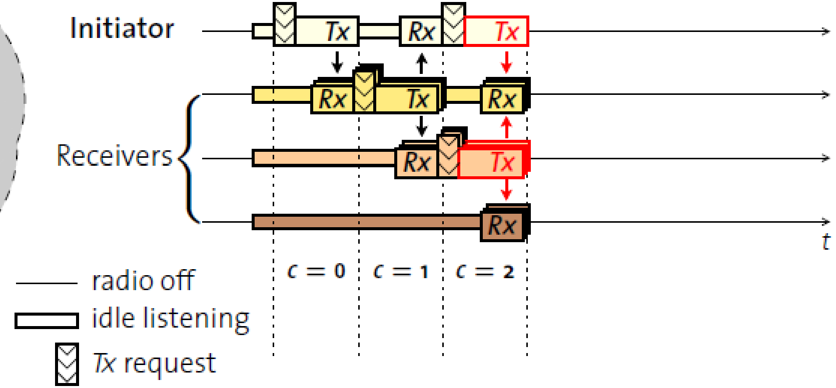
Glossy details

- At packet reception:
  - Increment relay counter **C**
  - Transmit synchronously (at a fixed period after the reception)

Example



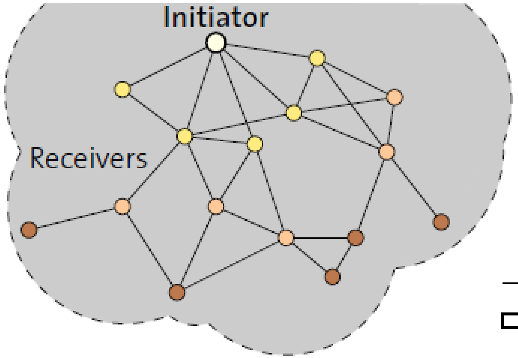
Timeline



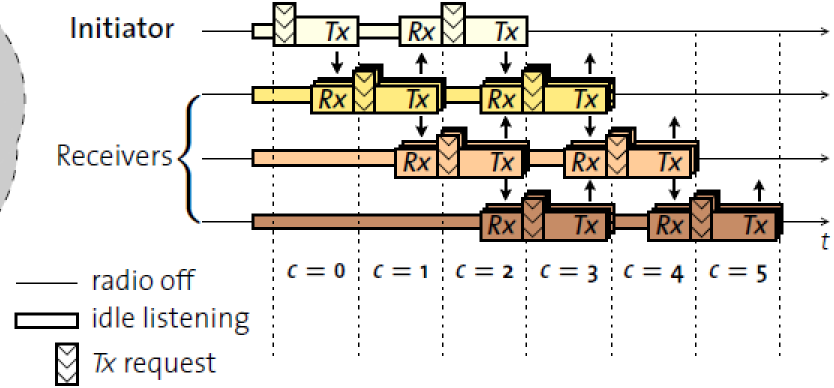
Glossy details

- Stop rebroadcasting and turn off radio when
  - Already transmitted N times
  - Networks pick N for reliability/energy tradeoff

Example



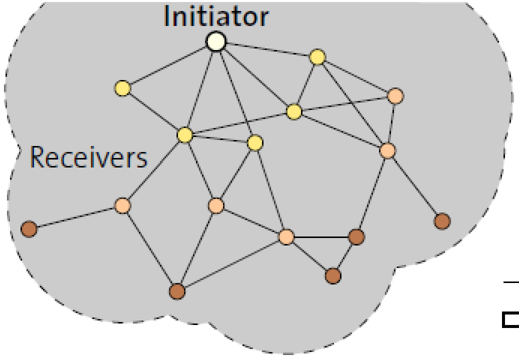
Timeline ($N = 2$)



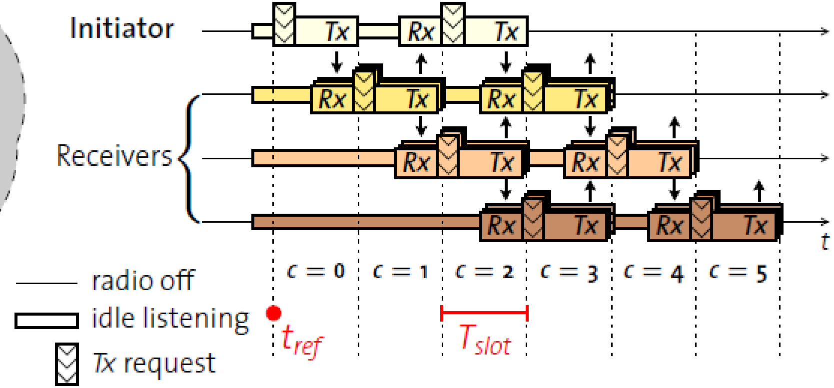
Glossy details

- T_{slot} is constant by design
  - Needs to be to make constructive interference work
- Beginning of slot (t_{ref}) provides synchronization point
  - As a bonus, all nodes are synchronized after flooding event

Example



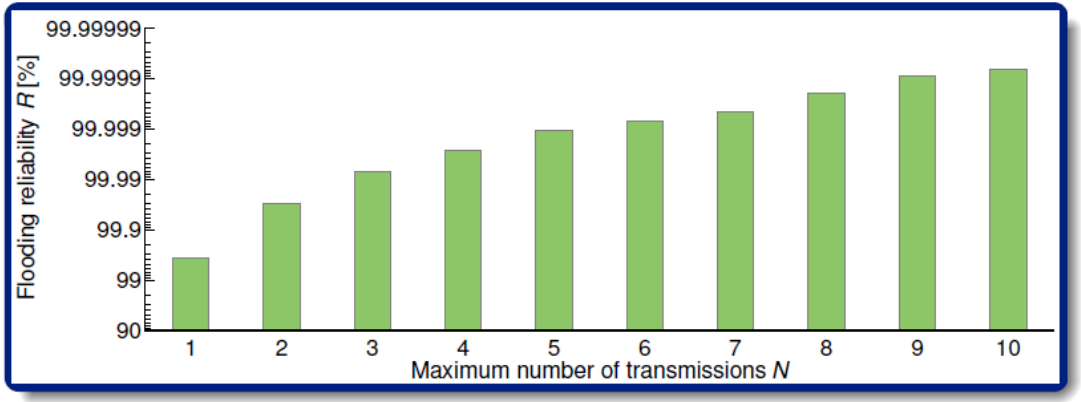
Timeline $(N = 2)$



Glossy implementation

- Device must be able to have tight time bounds on rx/tx
  - 500 ns wiggle room maximum
  - Includes receive, processing, transmission

- Need to pick an N for reliability

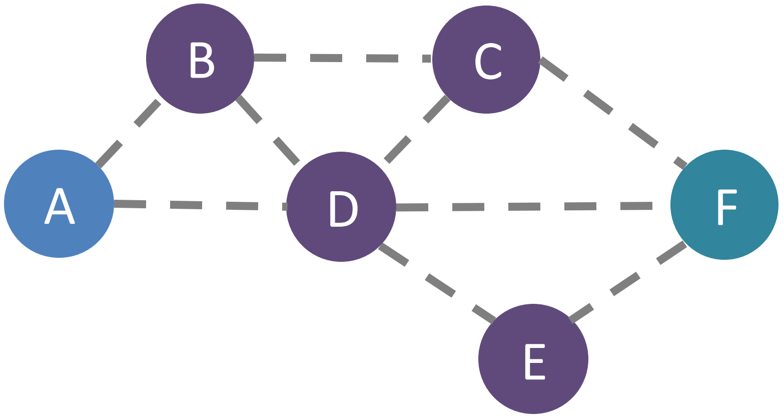


Extreme application of Glossy: Avoid routing altogether

- Low-Power Wireless Bus (LWB)
  - Federico Ferrari, Zimmerling, Mottola, Thiele. SenSys'12
- Use Glossy floods for all communication
  - Make one broadcast domain (a bus) where all nodes communicate
  - Avoids all issues of routing, everything is a broadcast
    - Works for unicast, multicast, anycast, and broadcast transmissions
- General idea: TDMA Glossy floods
  - Synchronization is already given to nodes by Glossy
  - One coordinator makes the TDMA schedule

Implications of Flooding

- Synchronous Transmission Protocols were a major disruption
  - Showed that flooding data from A → F used *less net energy* than routing A → F
    - (Net here is energy consumed by all nodes in the network)
  - Why??
  - What kinds of applications, traffic patterns would synchronous transmissions be worse than routing for?



Up Next: Thread

- Lab 3A:
  - Understanding basics of 15.4
  - (Simplest) Low Power MAC implementation
- Lab 3B:
  - Investigating Thread networks, routing
  - Application-layer services