Lecture 09
IoT Network Routing

CS397/497 – Wireless Protocols for IoT

Branden Ghena – Spring 2024

With slides from Federico Ferrari (ETH Zurich)
and assistance from Andreas Biri (ETH Zurich)

Materials in collaboration with
Pat Pannuto (UCSD) and Brad Campbell (UVA)

Northwestern

Today’s Goals

- Overview of routing for mesh networks
  - Walkthrough of one protocol (AODV: what ZigBee uses)

- Describe research in improving data dissemination

Outline

- **Simple Routing**
- Mesh Routing
- Better Flooding
- Low-power Access Control

Routing goals

- Have a packet, have a destination, how do we connect them?

- We'll think about a couple of approaches here
  - Simple techniques
    - Broadcast, tree structures
  - Mesh techniques
    - Understand the available routes and select a "good" one

Simple routing solutions

- Broadcast
  - The link-layer solution for everything

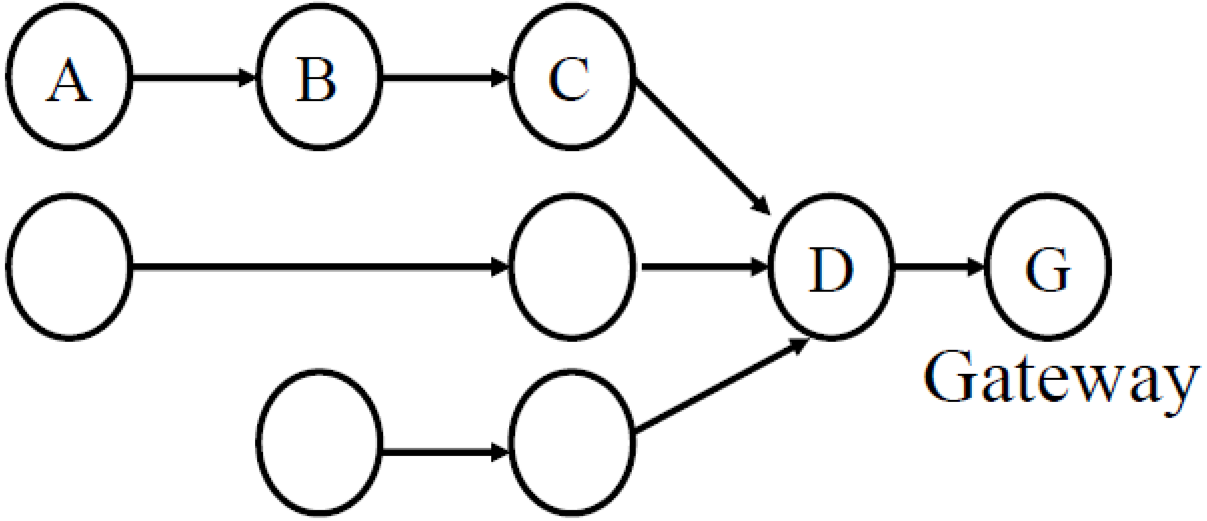
- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.

- 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)

Many-to-one routing (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

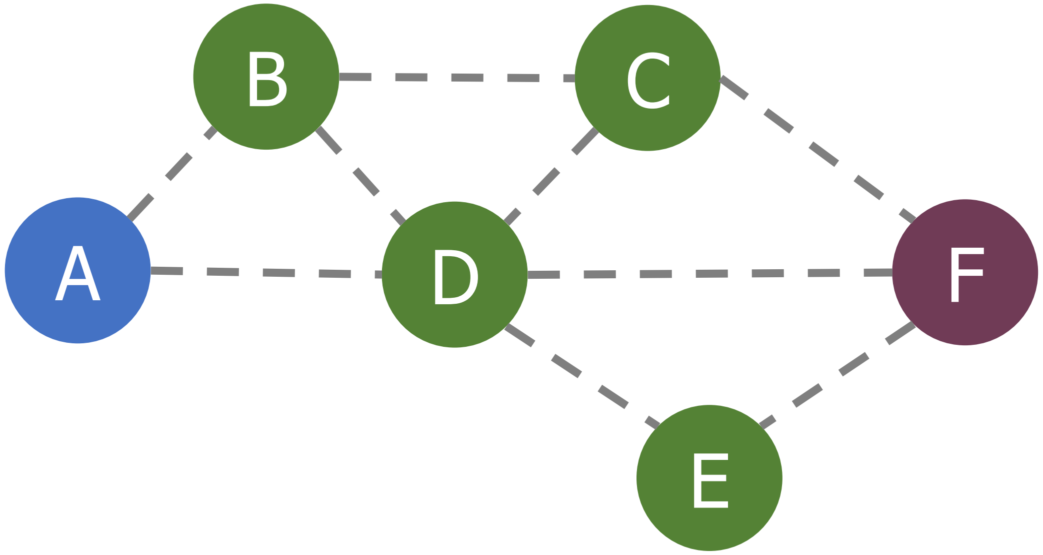


Outline

- Simple Routing
- **Mesh Routing**
- Better Flooding
- Low-power Access Control

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



Flooding

- Mesh equivalent of broadcast
  - Each node sends to each other node
  - Eventually packets will reach the desired destination
  - Not really routing at all…

- **Question: how do we make it stop?**

Flooding

- Mesh equivalent of broadcast
  - Each node sends to each other node
  - Eventually packets will reach the desired destination
  - Not really routing at all…

- **Question: how do we make it stop?**
  - Maximum retransmissions counter on each packet
    - Decrement at each hop. Drop packet when it hits zero
    - Need some guess for how many hops to destination

  - Or keep some history of recently flooded packets
    - Don’t retransmit a recently sent packet

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

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 documentation: https://www.ietf.org/rfc/rfc3561.txt

AODV Route Requests (RREQs)

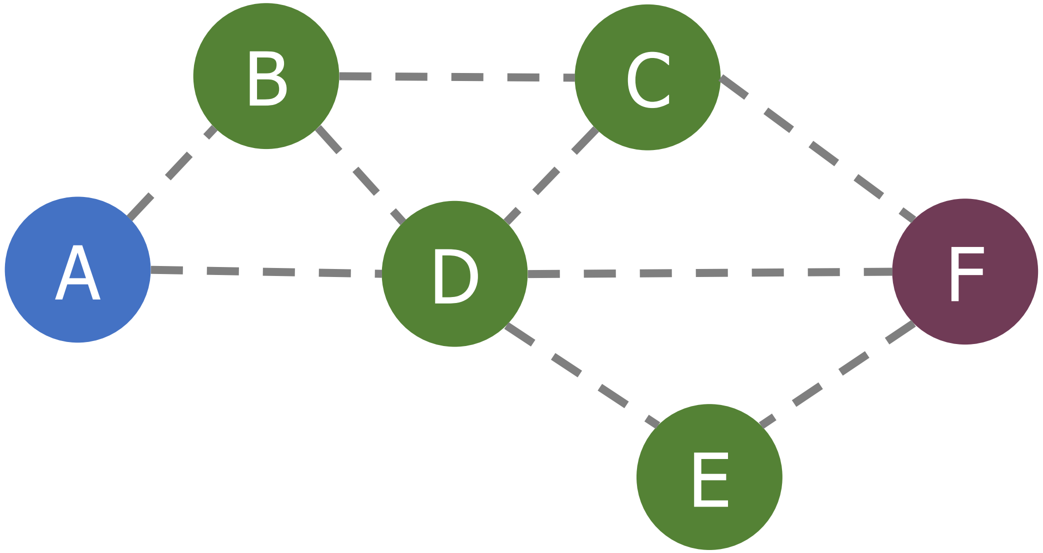
Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count

- Request ID identifies this RREQ
  - Used to discard duplicates during flooding
  - Unless less hops and equally recent, OR more recent
- 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

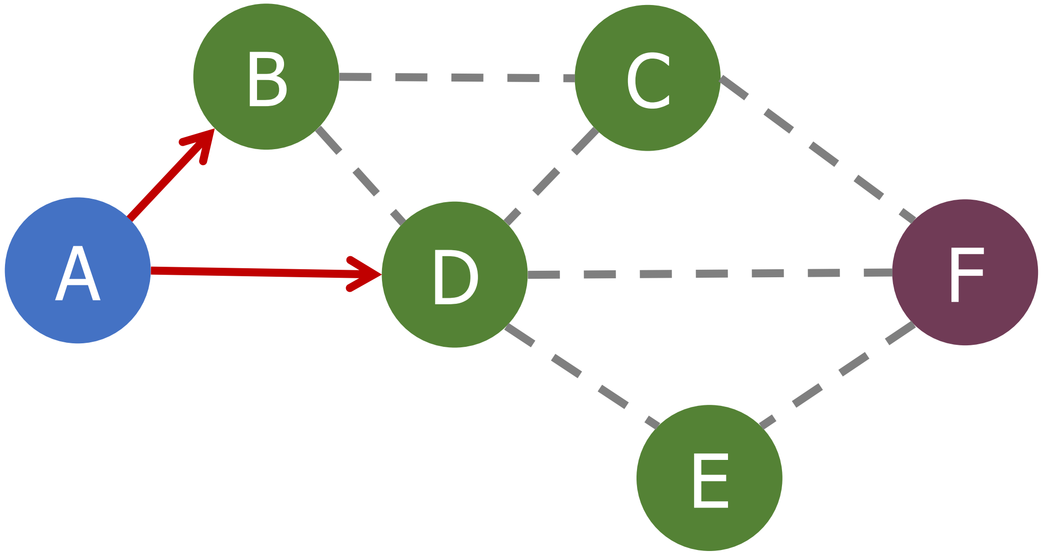
A wants to find a route to F, so it sends out an RREQ



Example AODV RREQ (A to F)

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

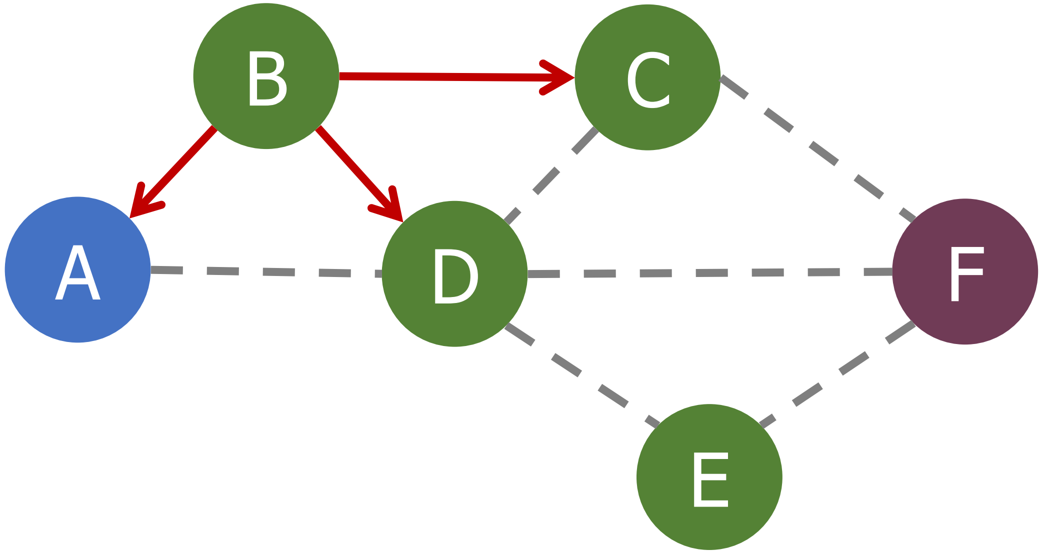
B and D also opportunistically add a routing table entry for A



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



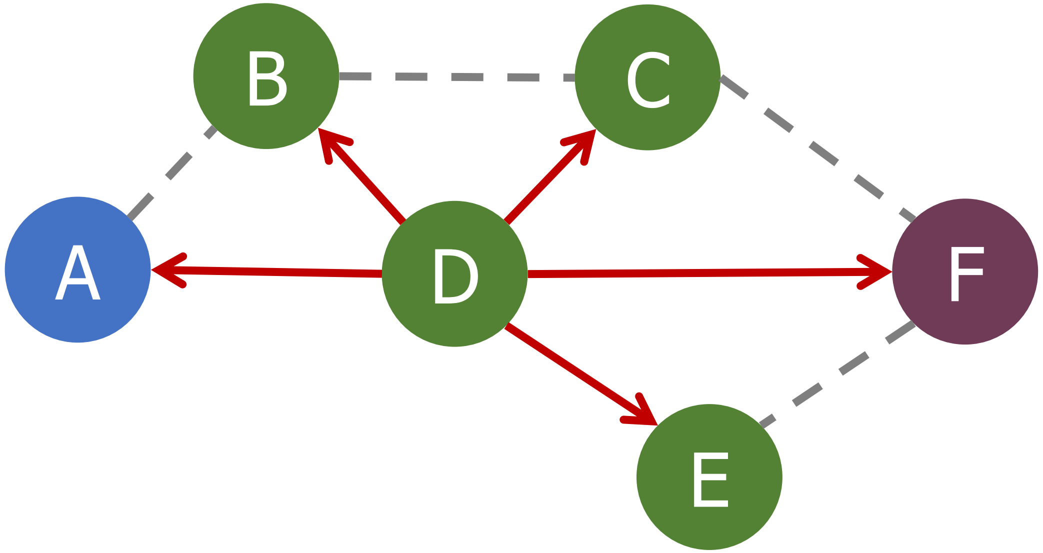
Example AODV RREQ (A to F)

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 add routing table entries to A

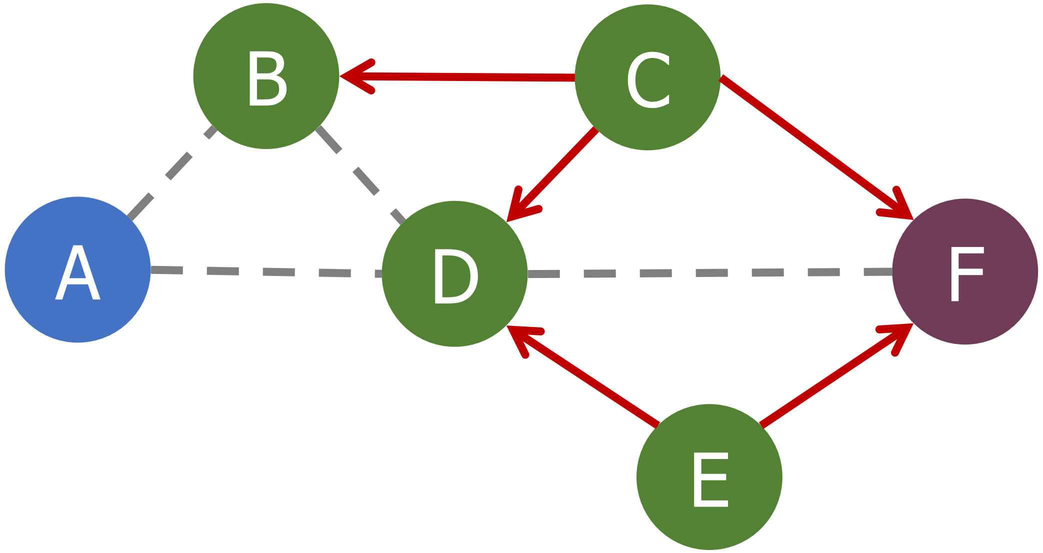


Example AODV RREQ (A to F)

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

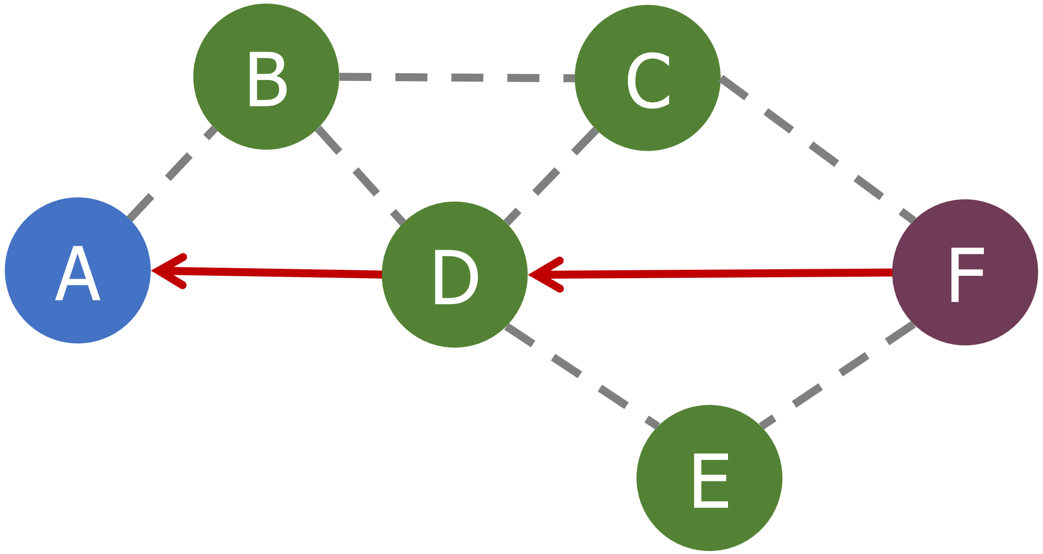
- Reply is sent unicast back to the source via newly constructed route
  - Each node along the way already knows the route back

- Includes most recent own sequence number as a sense of recency
  - “Destination” from the perspective of the original RREQ
  - No need for source sequence number anymore

Example AODV RREP (F to A)

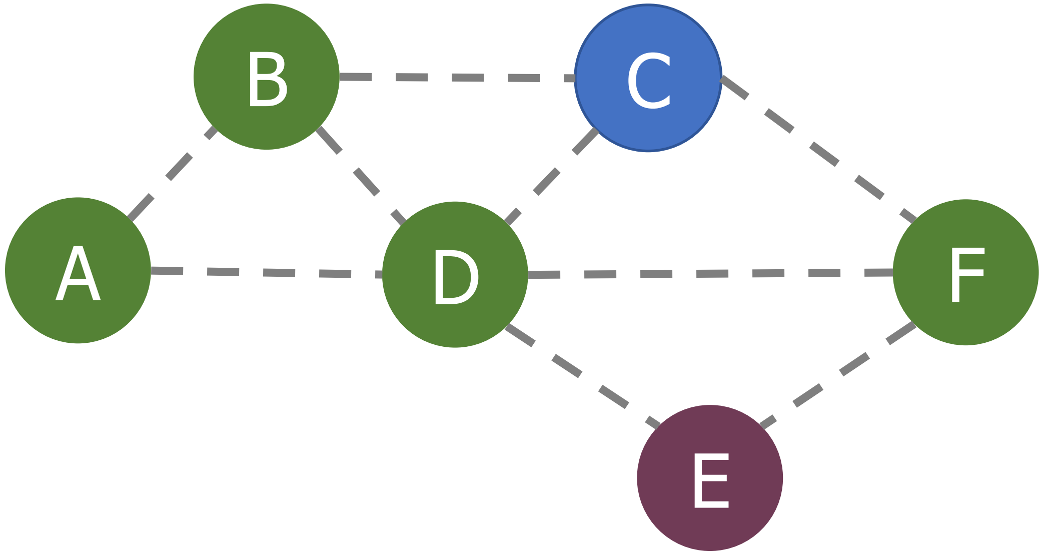
Source Address	Destination Address	Destination SeqNo	Hop Count
F	A	7	2

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



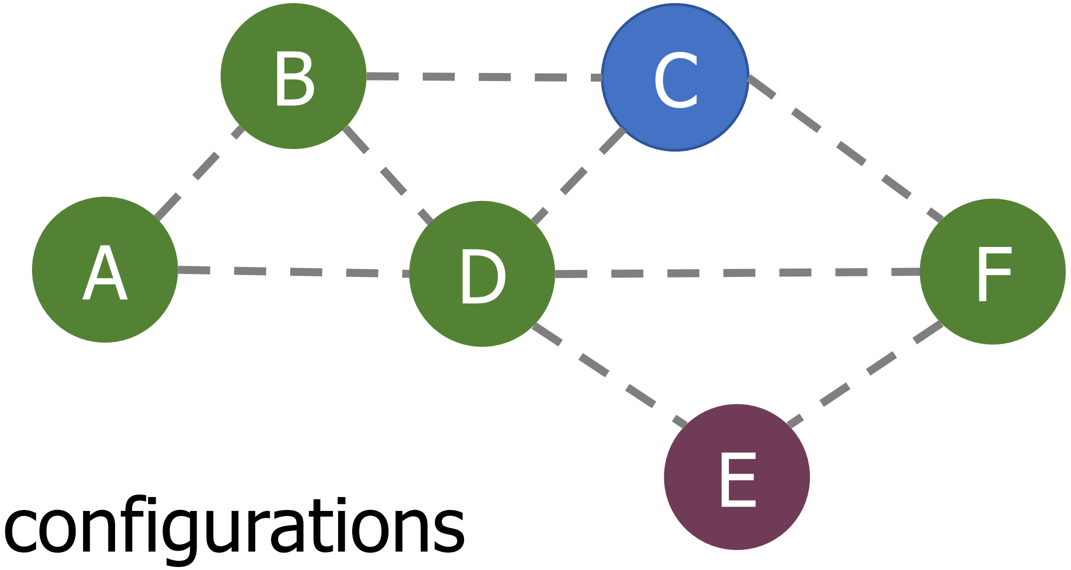
Break + Practice

- 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



RREP optimization

- An intermediate node responds with RREP if it already has a path to destination with a more recent Destination sequence number

- Source may get multiple RREP responses with different recency and hop counts
  - So, some intermediate node could respond “here’s the route I knew of when sequence number was 5”
  - Then, destination node could respond “here’s the route right now, I’m actually on sequence number 12”
  - Likely want the most recent
    - If equally recent, use the least hops

When to update your route

- Routing table entries are updated on RREP if:
  - Sequence number in routing table is marked as invalid
  - Destination sequence number in the RREP is greater the listed one
  - Sequence numbers are the same, but the route was marked as inactive
  - Sequence numbers are the same, but the hop count is smaller

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

Alternative: 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
        - Either old next-hop it was already aware of
        - Or new next-hop through neighbor (with cost of their hops + 1)

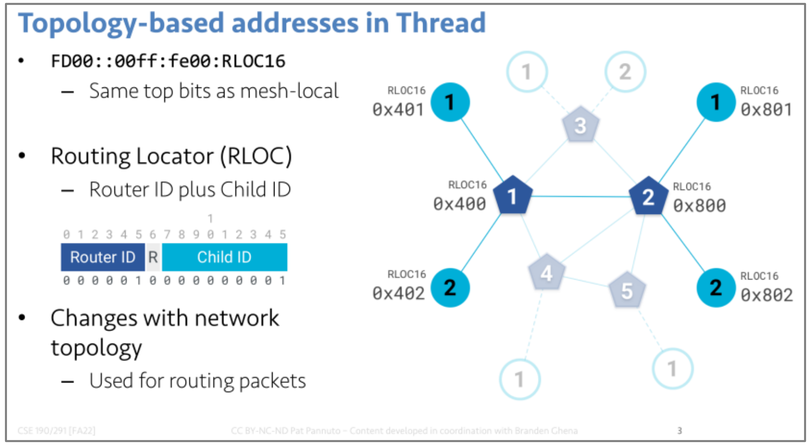
    - Need to be careful to avoid loops!

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



Topology-based addresses in Thread

- `FD00::00ff:fe00:RLOC16`
  - Same top bits as mesh-local
- Routing Locator (RLOC)
  - Router ID plus Child ID
- Changes with network topology
  - Used for routing packets

CSE 190/291 [FA22] CC BY-NC-ND Pat Pannuto – Content developed in coordination with Branden Ghena 3

- Multicast uses a data dissemination protocol (Trickle)
  - Or falls back to flooding

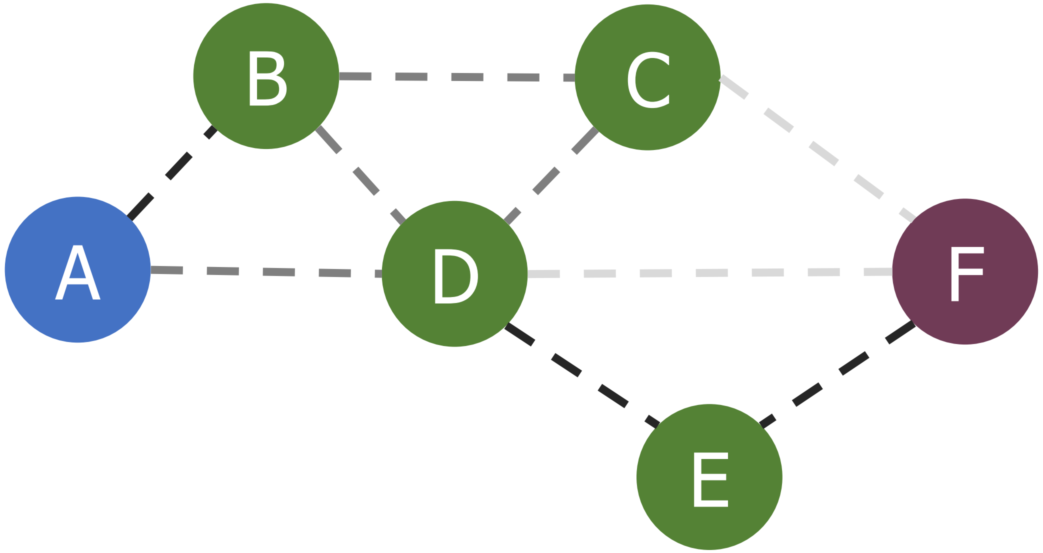
Break + Discussion

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

Reliability as a cost metric

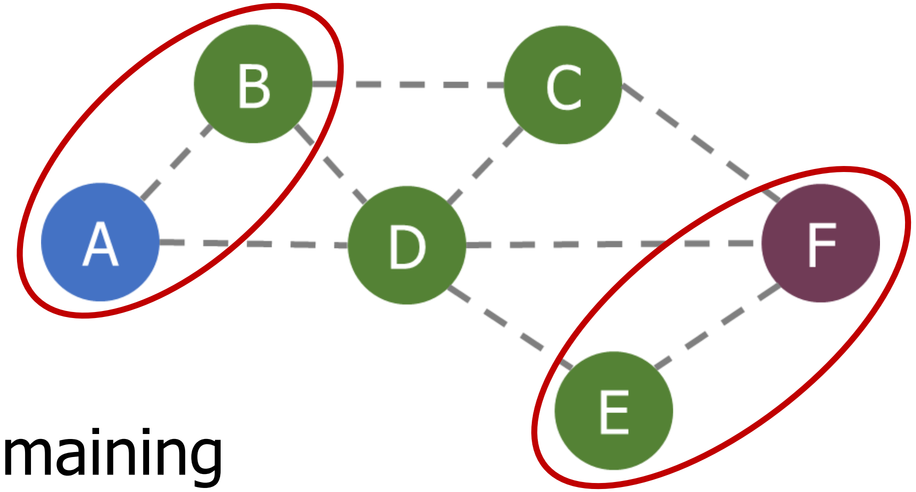
- Link quality can very from node to node
  - Fewest hops might not be the “fastest” or “most reliable” path

- 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
- **Better Flooding**
- Low-power Access Control

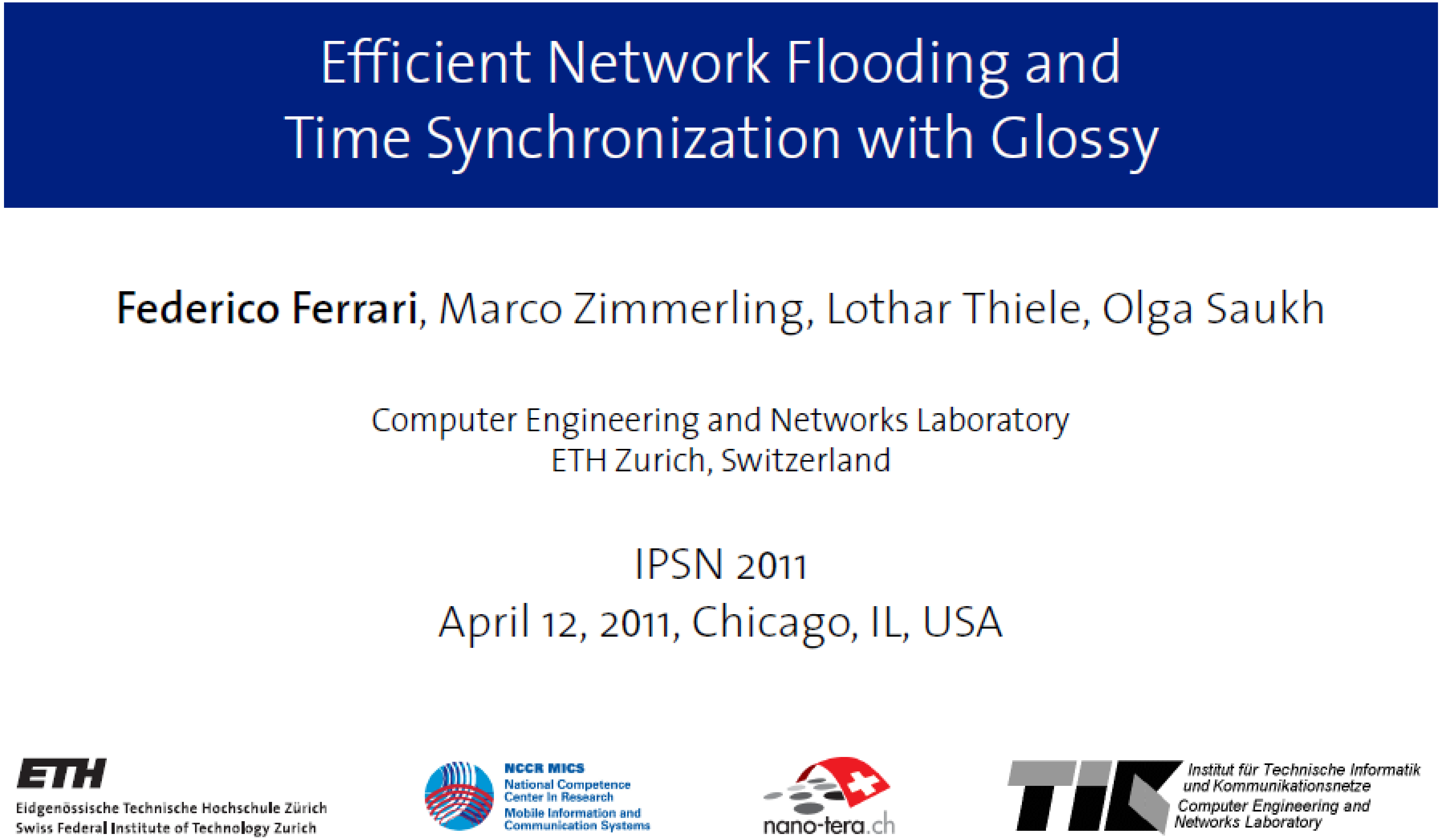
Flooding is a recreation of broadcasts

- 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

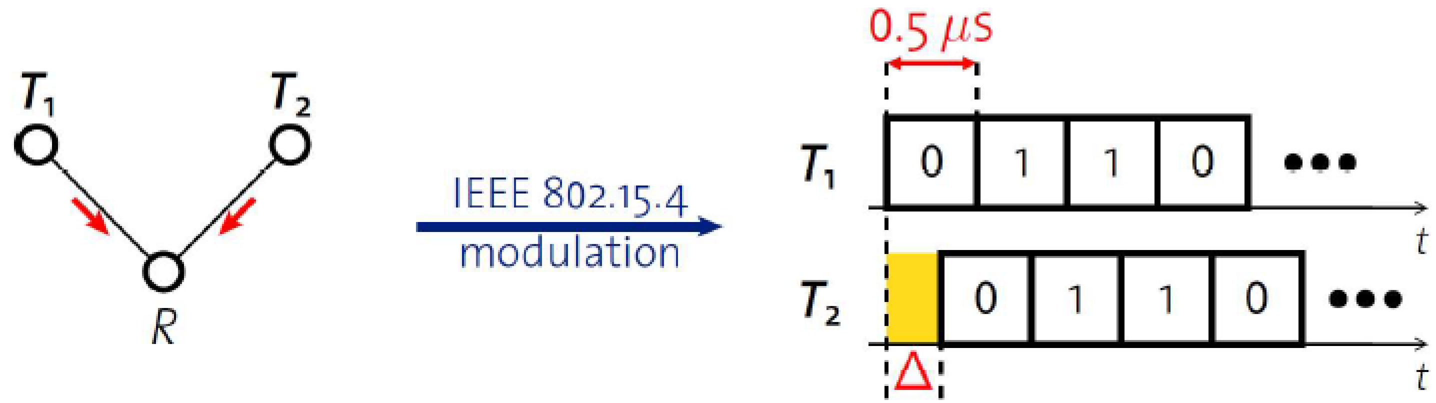
NCCR MICS
National Competence Center In Research
Mobile Information and Communication Systems

nano-tera.ch

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

Synchronous transmissions

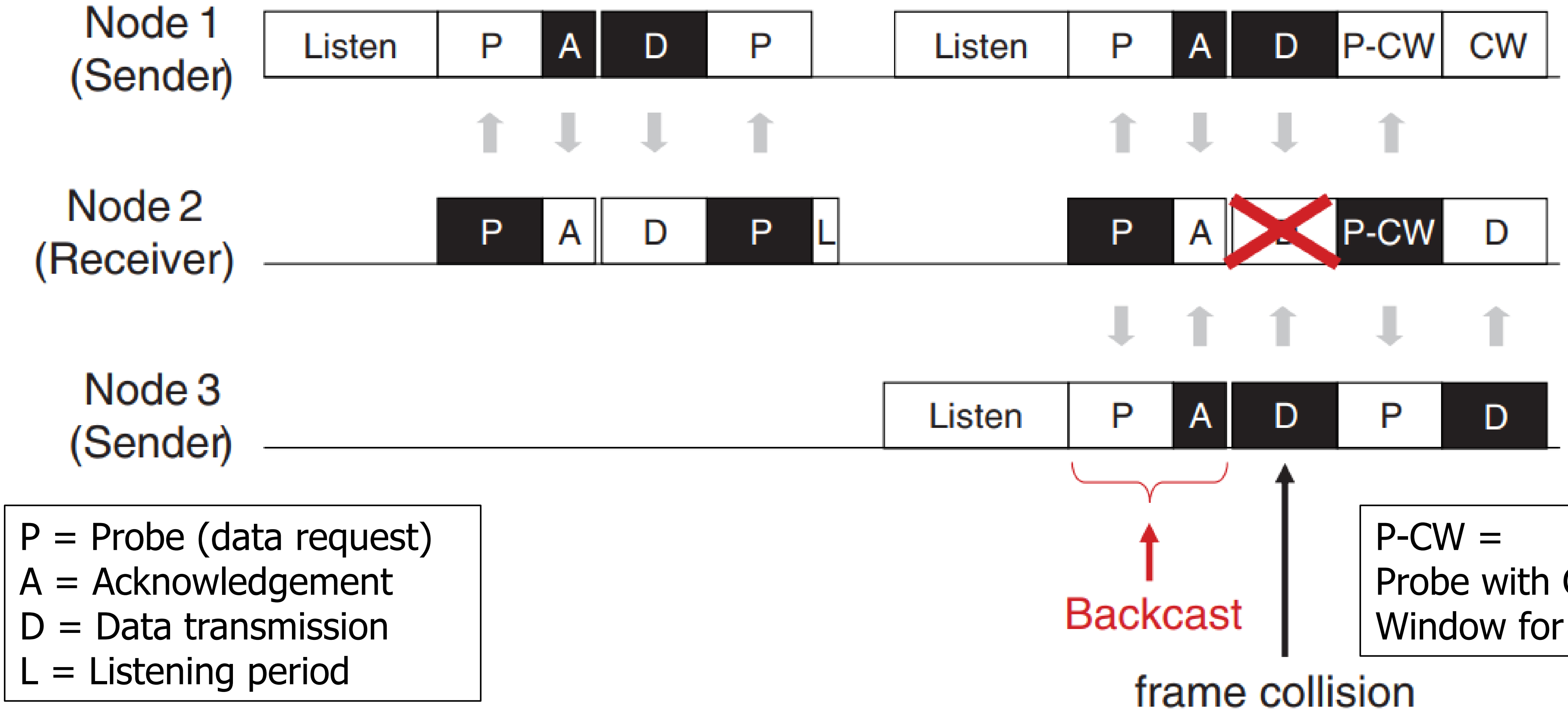
- Multiple nodes transmit **same packet** at **same time**



- 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)

Sidebar: broadcast transmission acknowledgements



P = Probe (data request)
A = Acknowledgement
D = Data transmission
L = Listening period

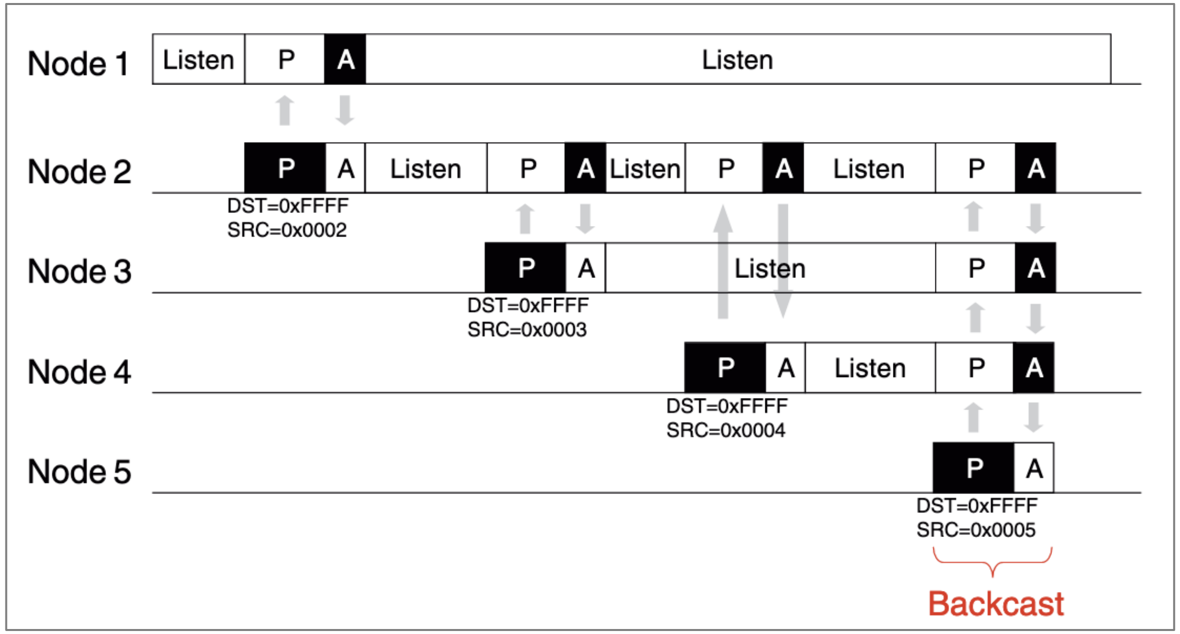
P-CW =
Probe with Contention
Window for response

A-MAC: https://web.eecs.umich.edu/~prabal/pubs/papers/dutta12amac.pdf

How do we avoid interference?

- Fundamental insight:
  - “say exactly the same thing at exactly (enough) the same time” leads to coherence instead of 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:



Node 1 | Listen | P | A | Listen

Node 2 | P | A | Listen | P | A | Listen | P | A | Listen | P | A

DST=0xFFFF
SRC=0x0002

Node 3 | P | A | Listen | P | A

DST=0xFFFF
SRC=0x0003

Node 4 | P | A | Listen | P | A

DST=0xFFFF
SRC=0x0004

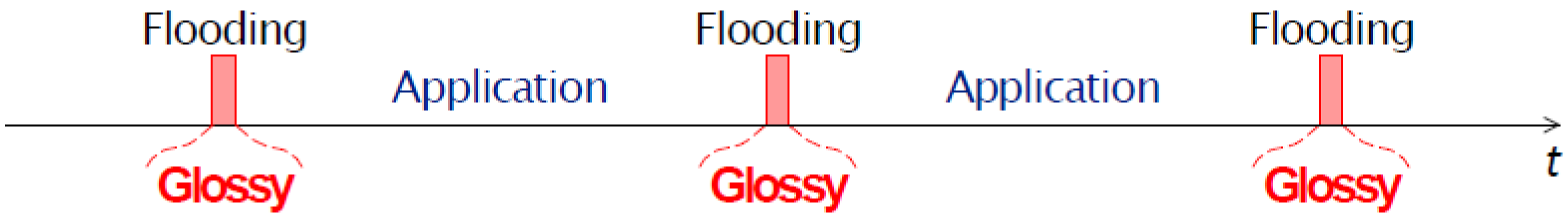
Node 5 | P | A

DST=0xFFFF
SRC=0x0005

Backcast

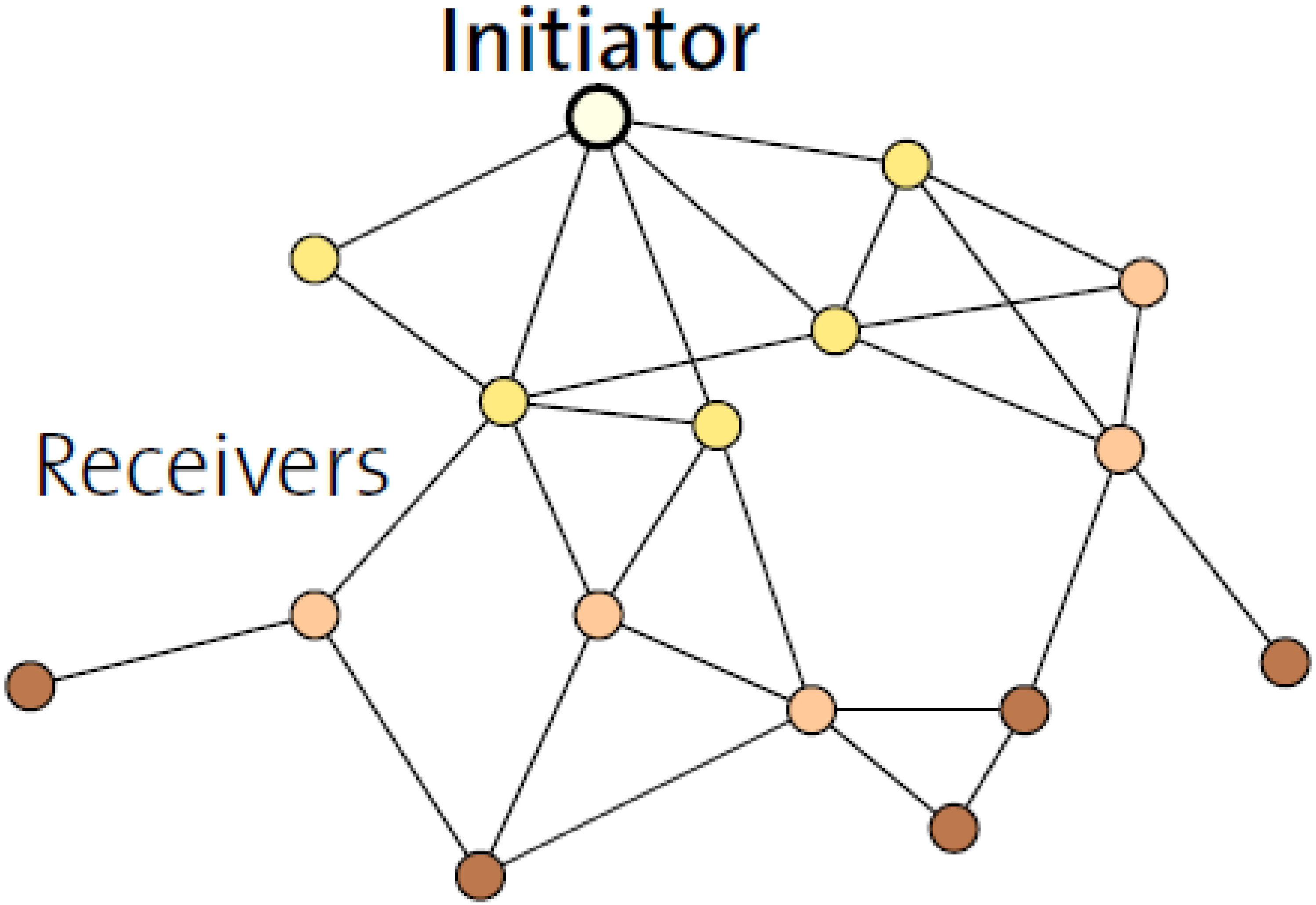
Glossy key techniques

- Temporally decouple network flooding from application tasks

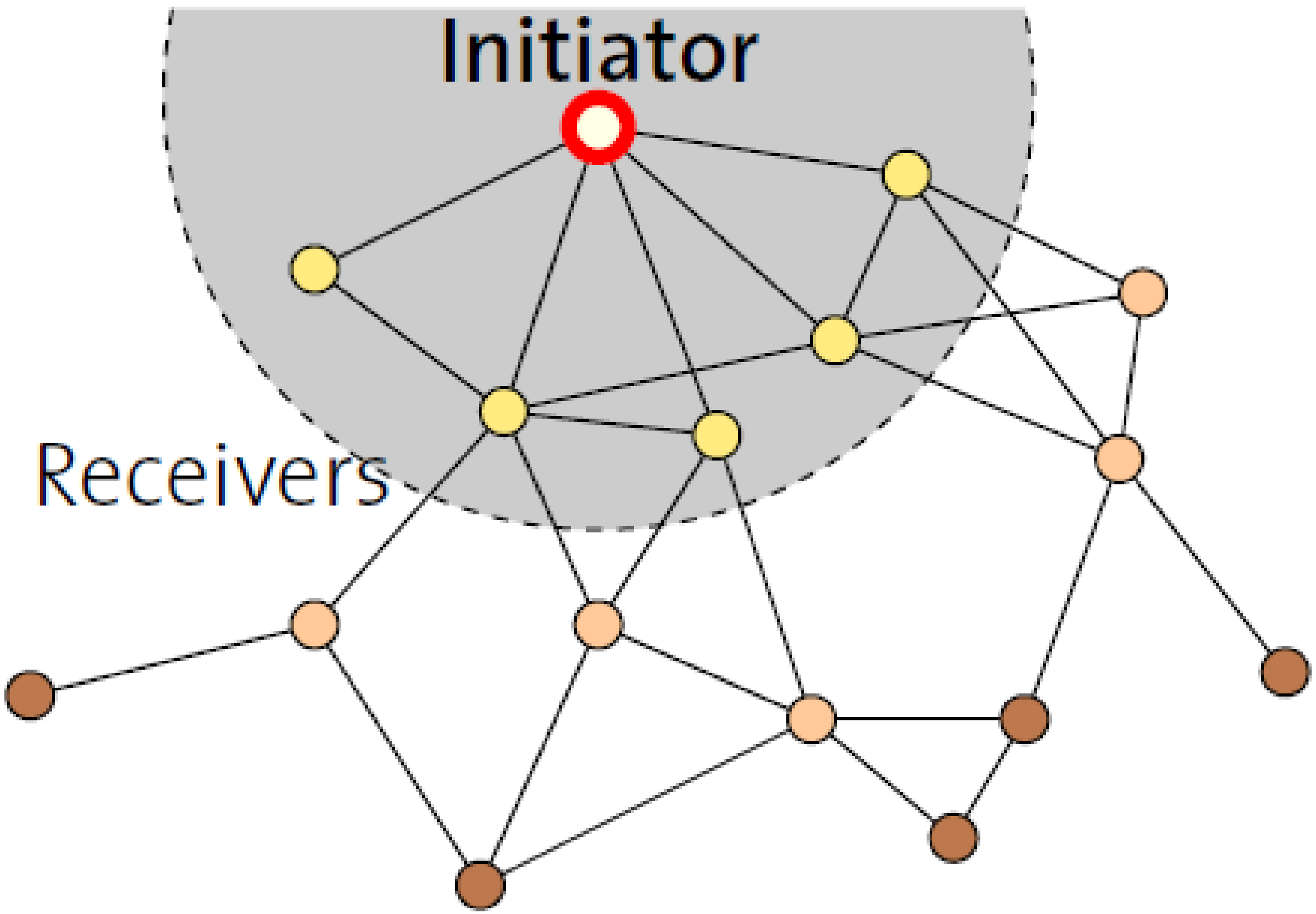


- Exploit synchronous transmissions for fast network flooding

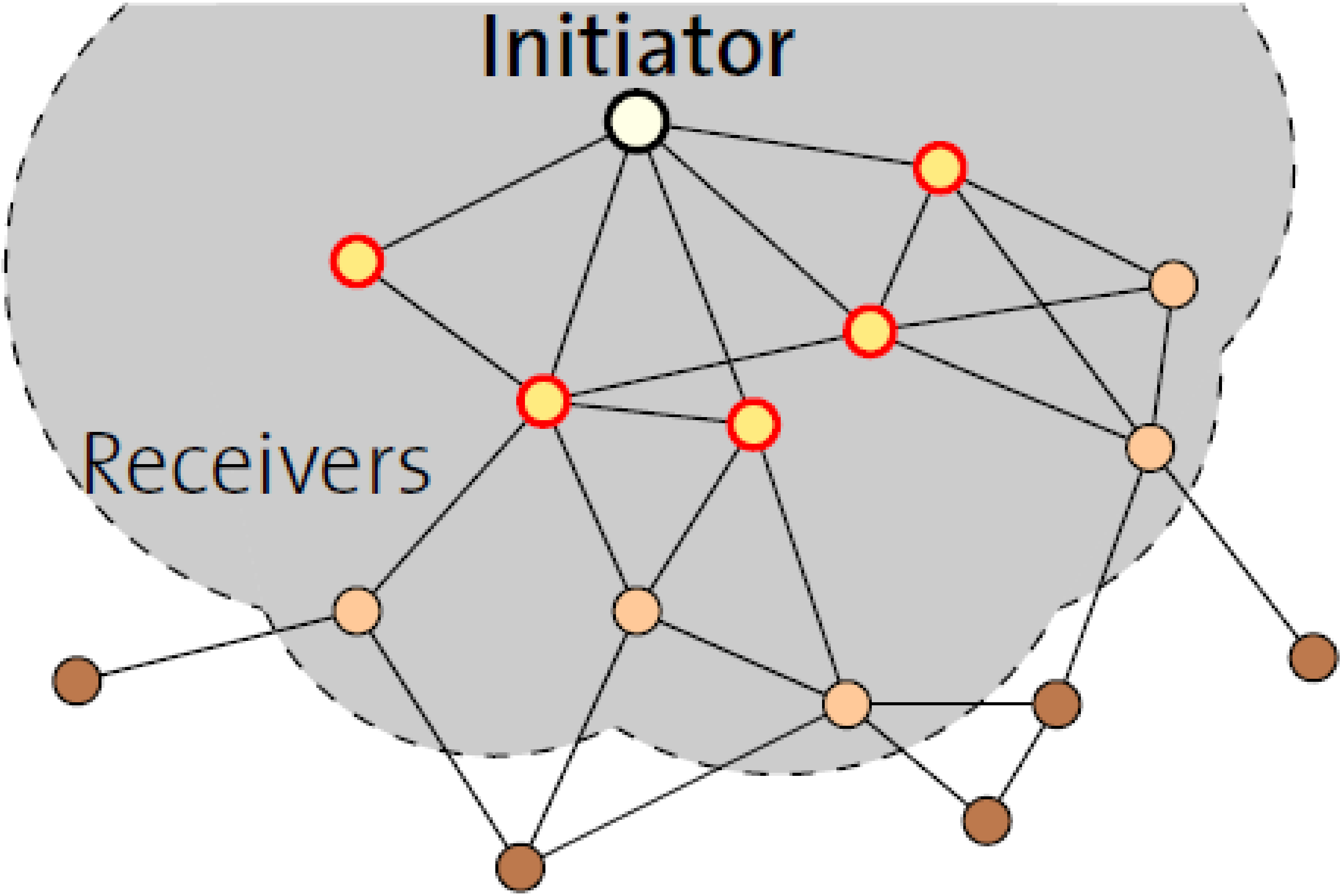
Fast packet propagation in Glossy



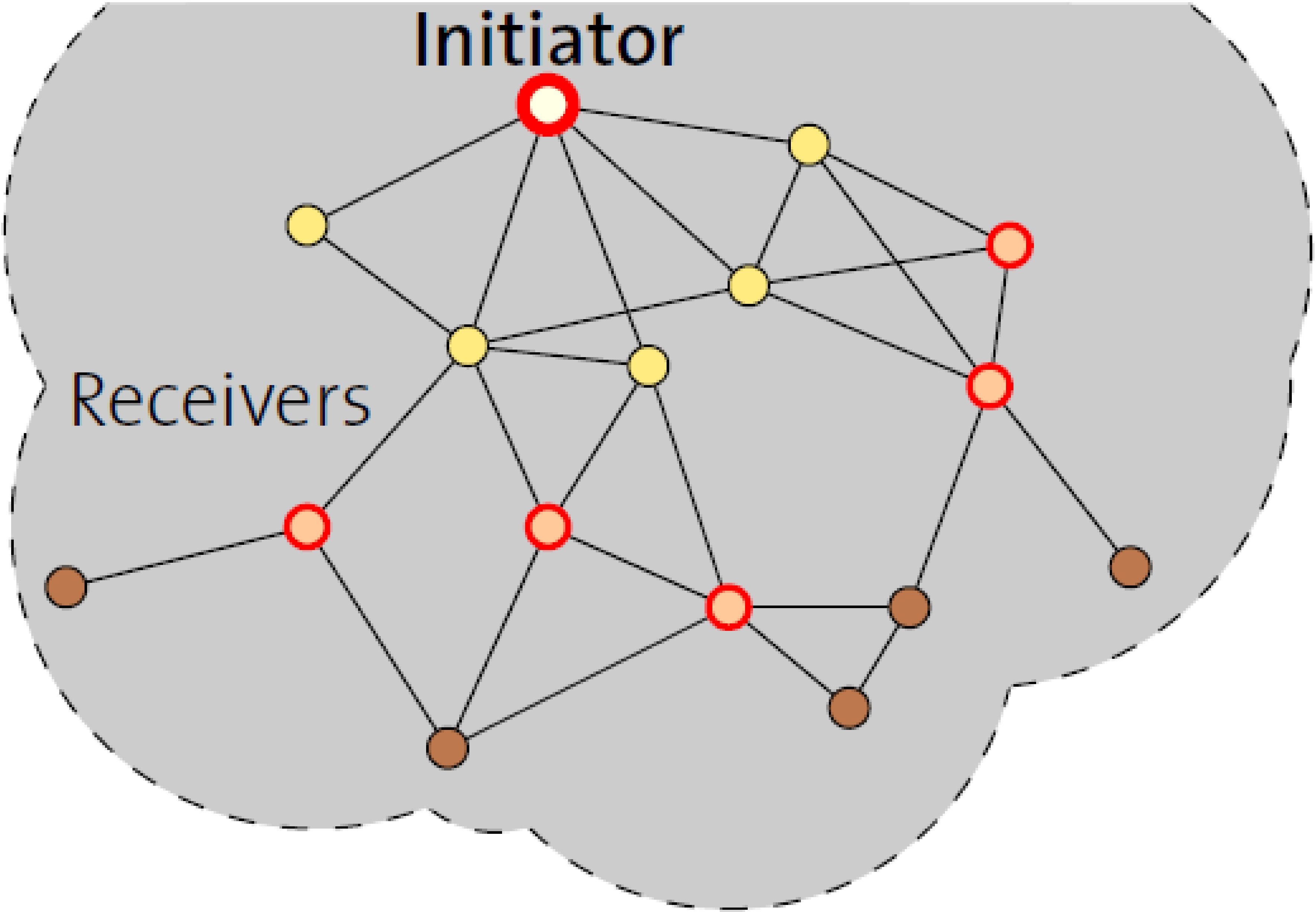
Fast packet propagation in Glossy



Fast packet propagation in Glossy

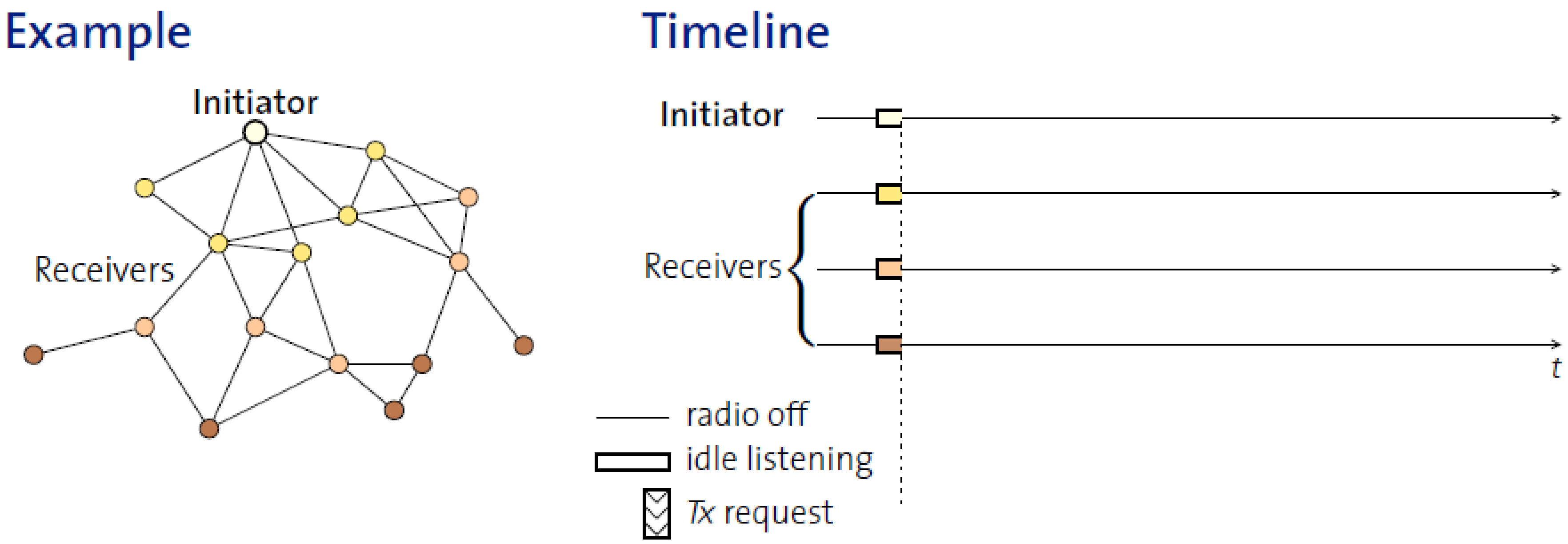


Fast packet propagation in Glossy



Glossy details

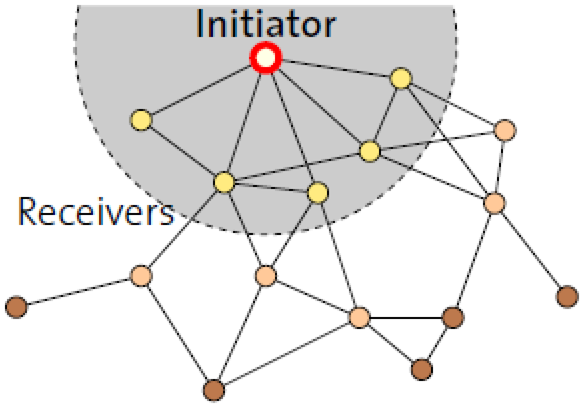
- When Glossy starts
  - All nodes turn on radios to receive



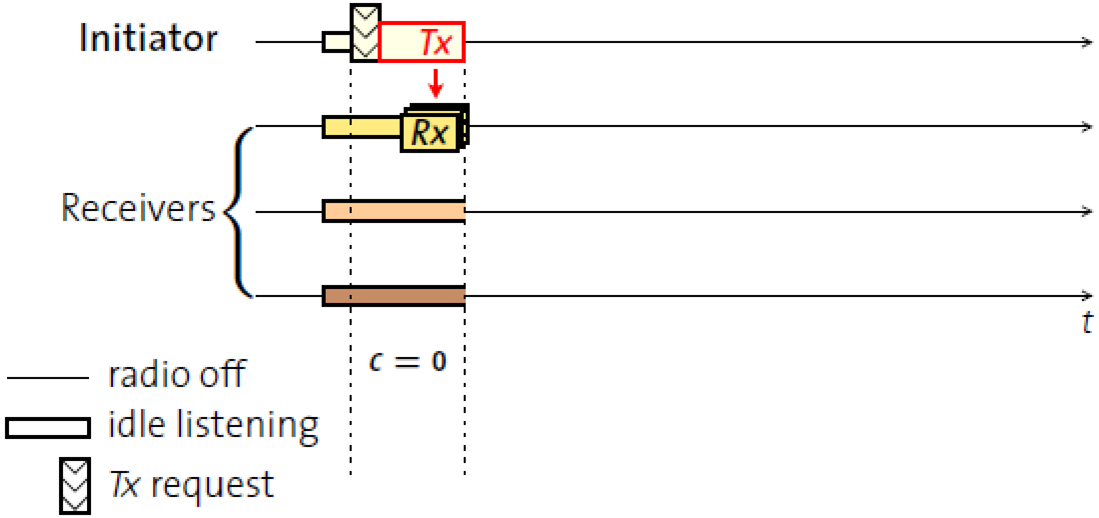
Glossy details

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

Example



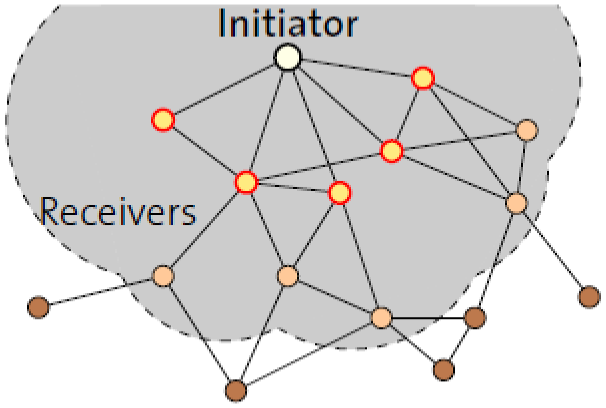
Timeline



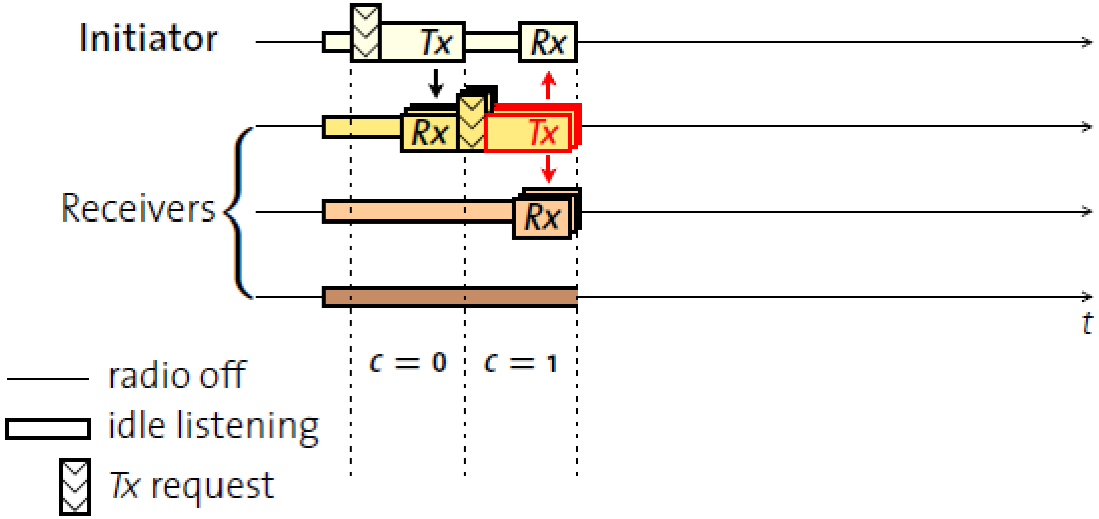
Glossy details

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

Example

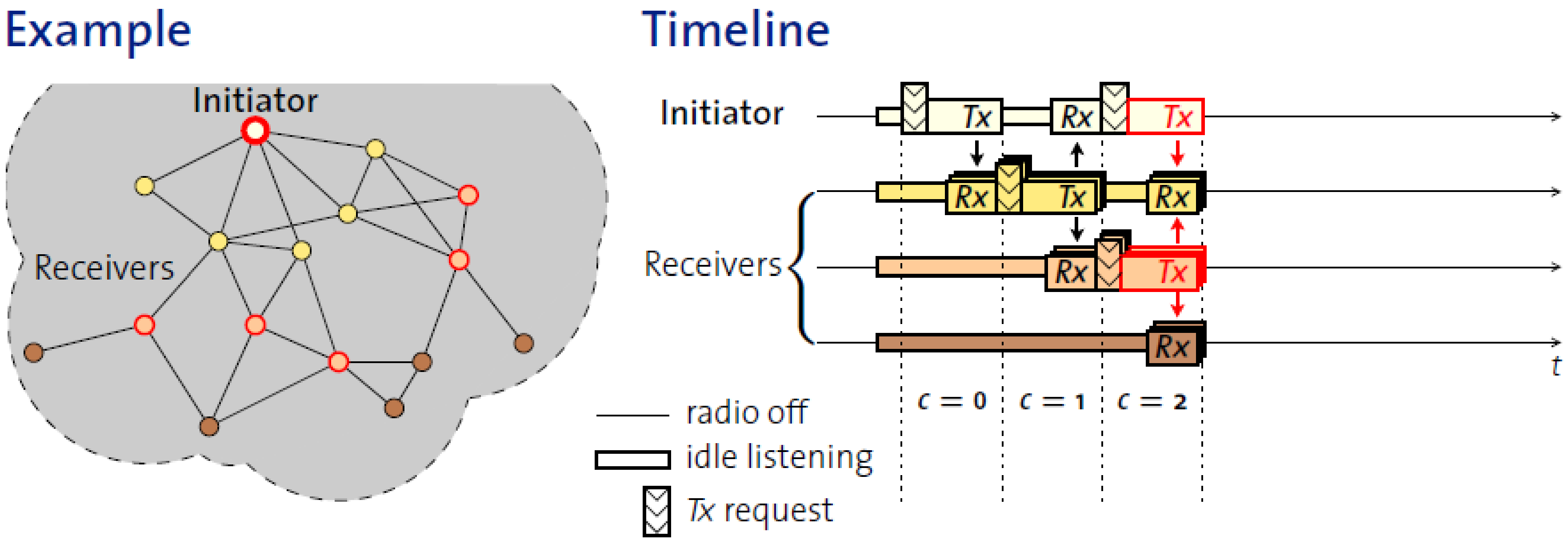


Timeline



Glossy details

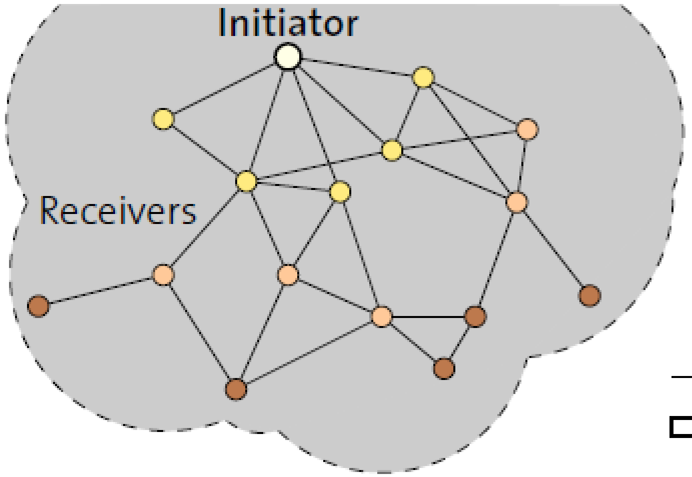
- At packet reception:
  - Increment relay counter **C**
  - Transmit synchronously (at a fixed period after the reception)



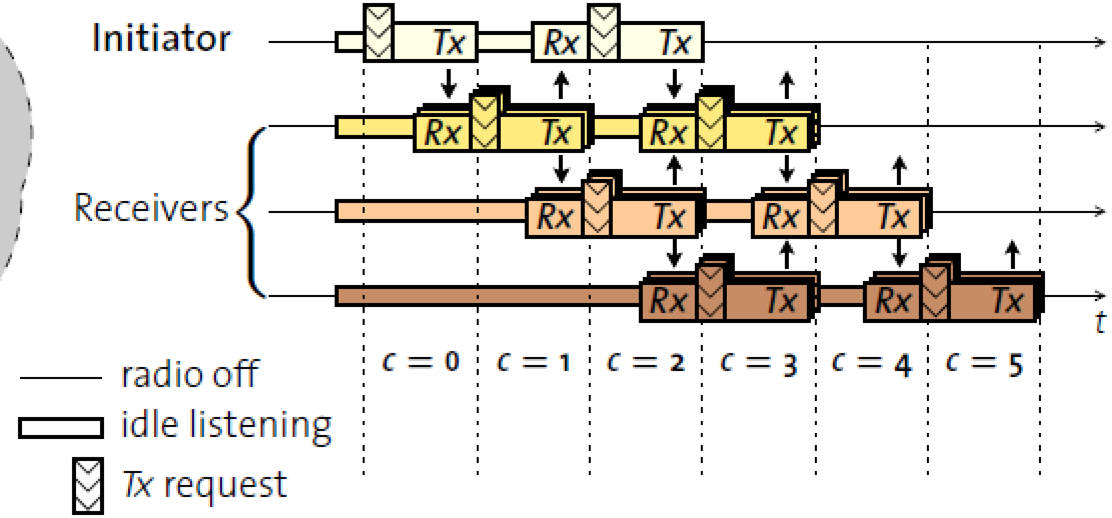
Glossy details

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

Example



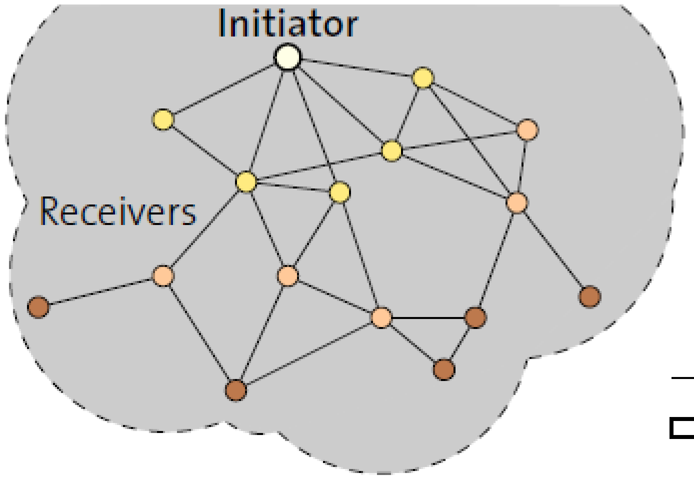
Timeline $(N = 2)$



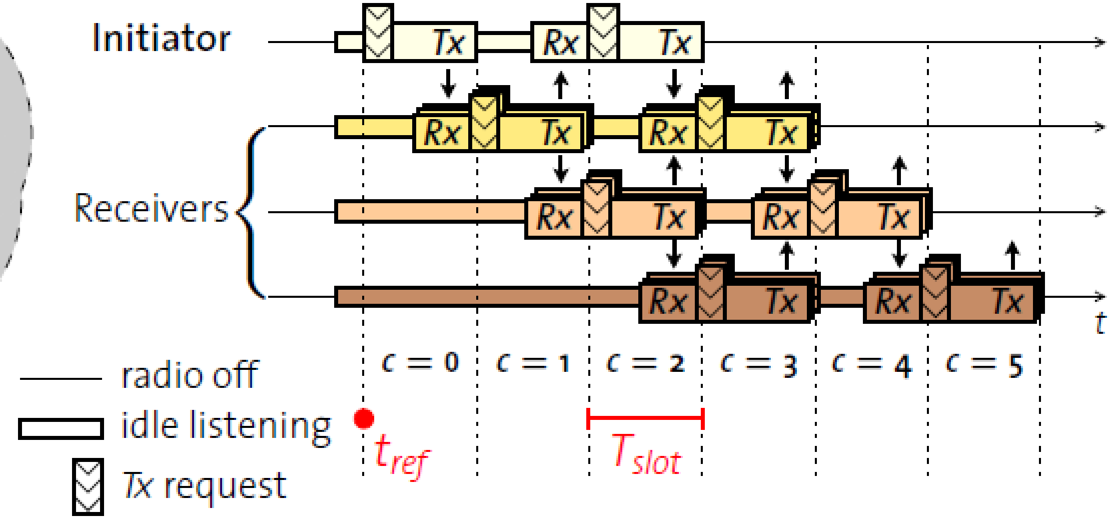
Glossy details

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

Example



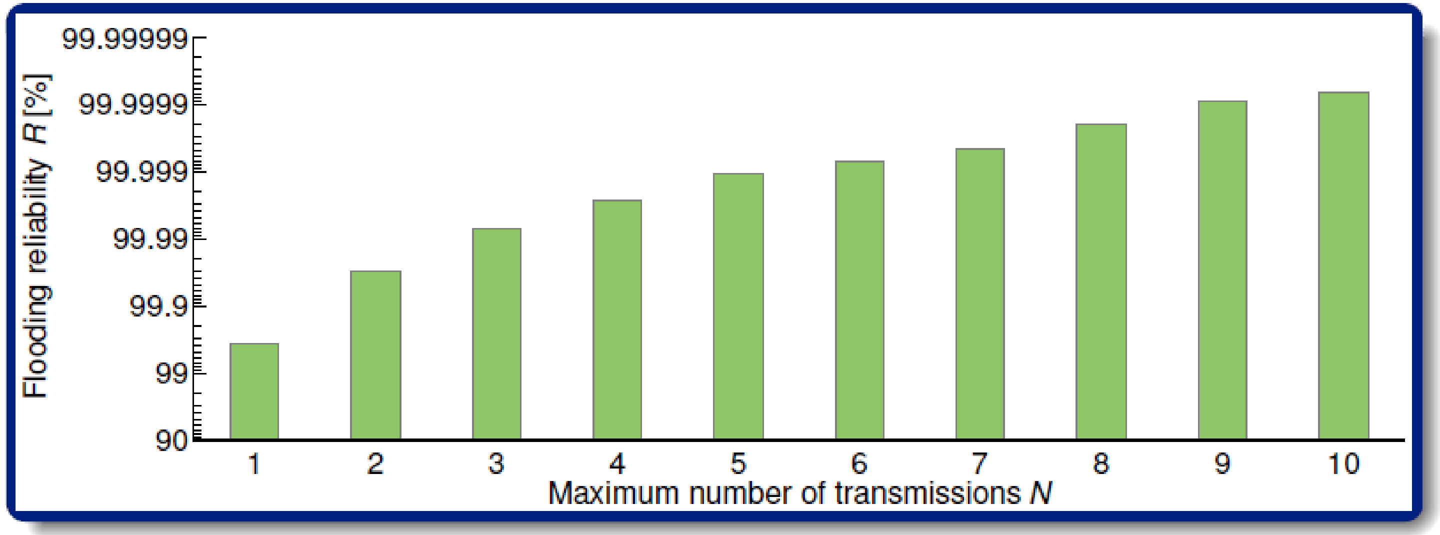
Timeline ($N = 2$)



Glossy implementation

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

- Need to pick an N for reliability



Application of Glossy: avoid routing altogether

- Low-Power Wireless Bus (LWB)
  - Federico Ferrari, Zimmerling, Mottola, Thiele. SenSys'12

- Use Glossy for all device 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

Outline

- Simple Routing
- Mesh Routing
- Better Flooding
- **Low-power Access Control**

Always-on Radios Simplify Protocols

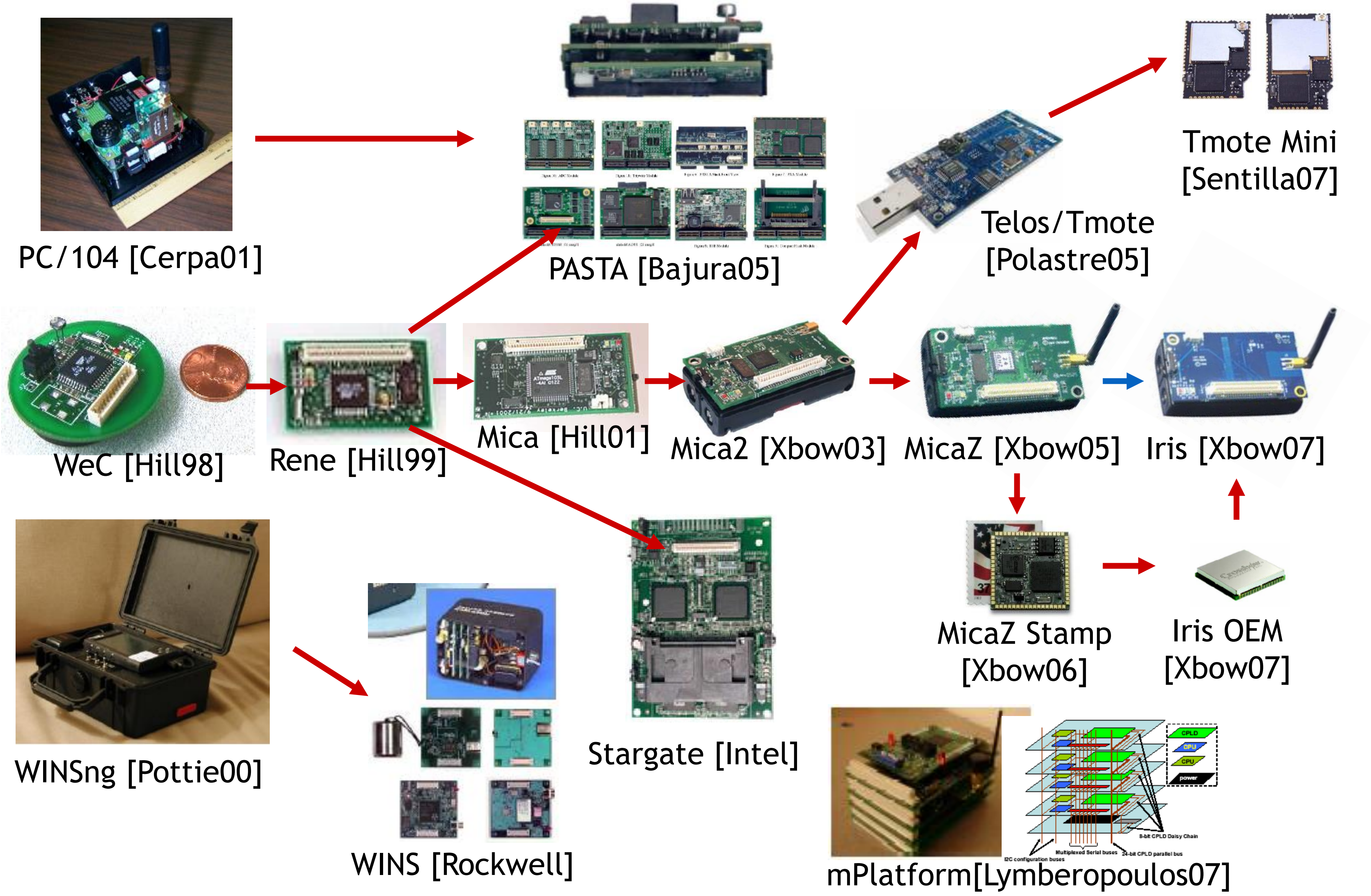
- Many protocols assume a more-powerful device with lots of energy
  - BLE: Central
  - Thread: Router/Leader
  - Zigbee: Router/Coordinator
  - WiFi: Router
  - LoRa: Gateway

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

Pokémon
bop it!
Von Dutch
U.S.A.
Soulja Boy
A LITTLE BIT DRAMATIC

Early 2000s

Early days of low power sensor nodes



PC/104 [Cerpa01]

PASTA [Bajura05]

Telos/Tmote [Polastre05]

Tmote Mini [Sentilla07]

WeC [Hill98]

Rene [Hill99]

Mica [Hill01]

Mica2 [Xbow03]

MicaZ [Xbow05]

Iris [Xbow07]

MicaZ Stamp [Xbow06]

Iris OEM [Xbow07]

WINSng [Pottie00]

WINS [Rockwell]

Stargate [Intel]

mPlatform[Lymberopoulos07]

Low power goal: multi-year operation using batteries

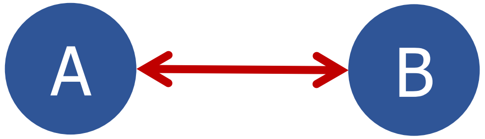
- $\text{Power}_{\text{node}} = P_{\text{CPU}} + P_{\text{TX}} + P_{\text{RX}} + P_{\text{SLEEP}}$
- $\text{Energy}_{\text{BATT}} = P_{\text{node}} * \text{lifetime}$

- For Lifetime to increase, P_{node} must decrease
  - $P_{\text{CPU}} \gg P_{\text{SLEEP}}$
  - $P_{\text{TX}} \gg P_{\text{SLEEP}}$
  - $P_{\text{RX}} \gg P_{\text{SLEEP}}$
  - Solution: Minimize Compute, TX, and RX
    - Maximize Sleep

Low power MAC principles

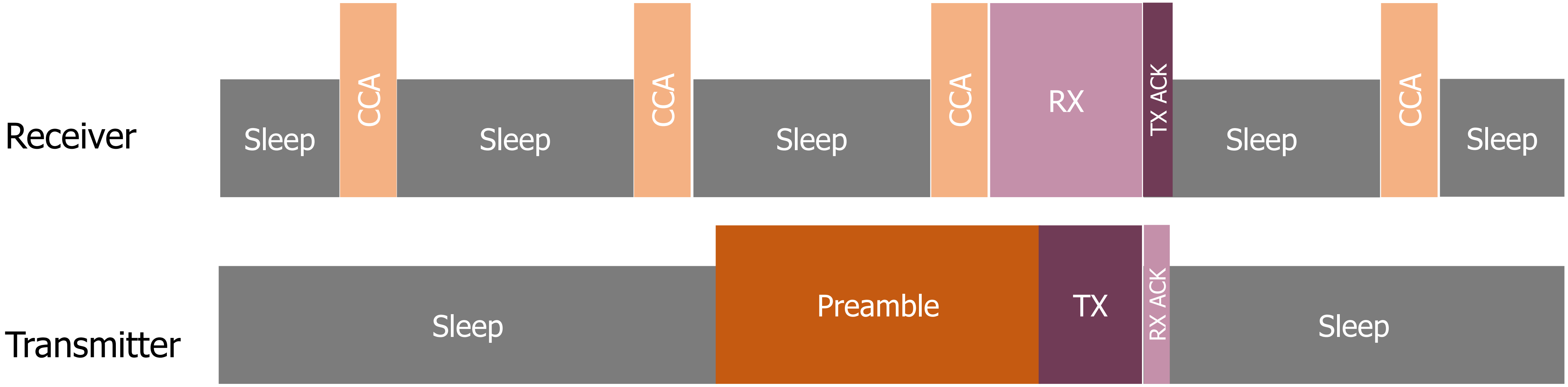
- Communication is possible if one device is receiving while other is transmitting
- Devices can only coordinate using the data communication channel (i.e. no out-of-band communication)
  - No global synchronization mechanism

- 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)



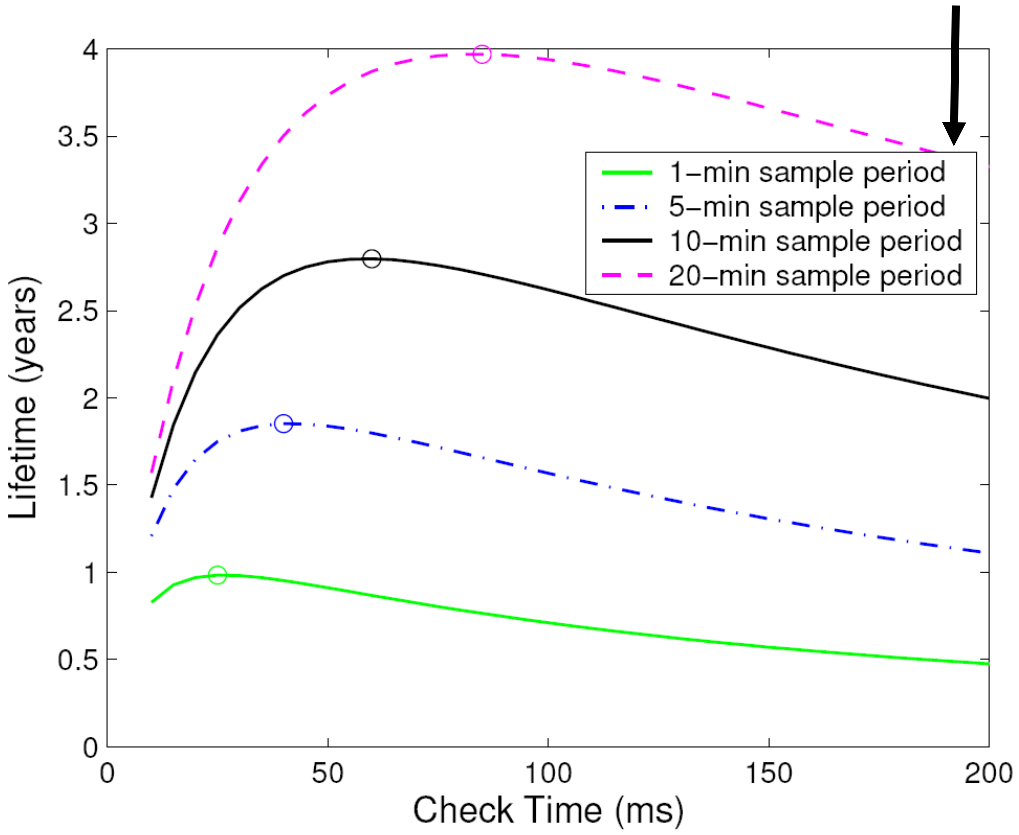
- Method:
  - Receiver periodically samples the channel
  - Transmitter sends a preamble long enough to ensure receiver will detect it
  - Upon detection, receiver stays awake to receive transmitted packet
  - Receiver ACKs if packet received correctly

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!

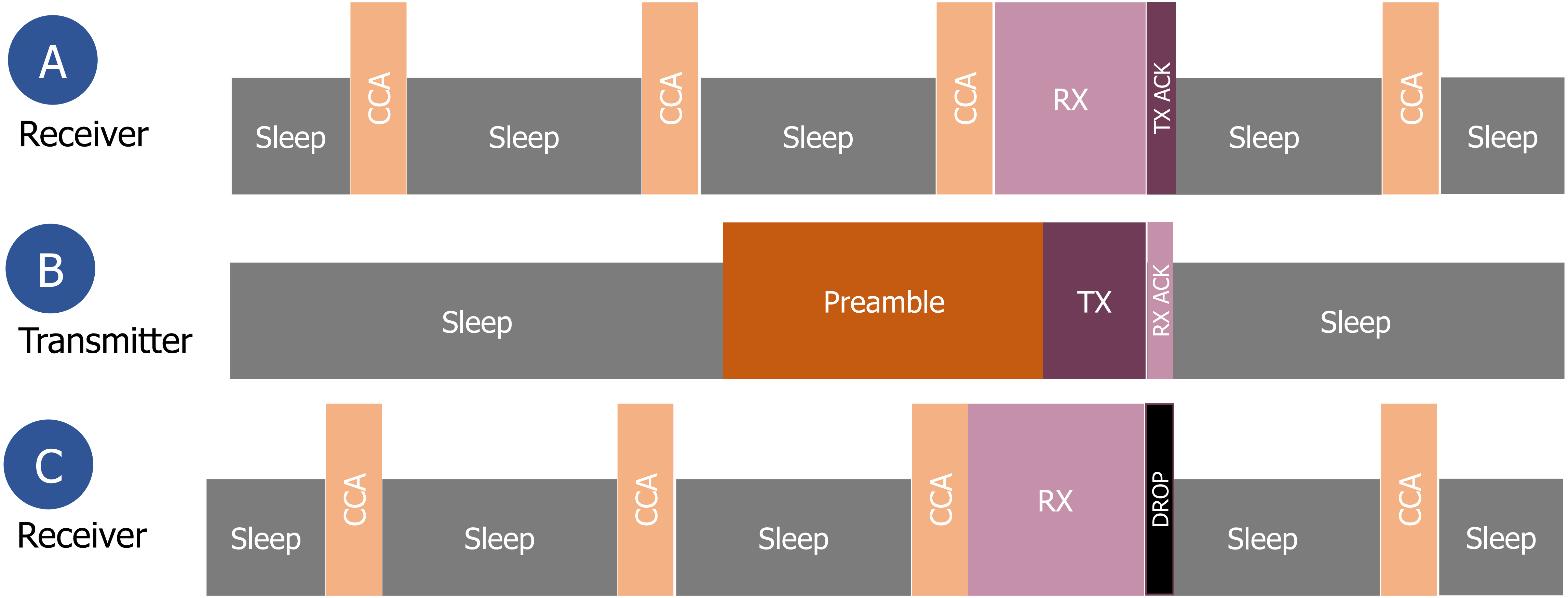


Sensor sample period controls the packet rate

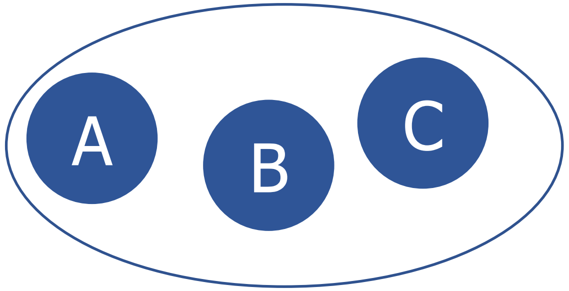
Time between listening
(checking if the channel is idle)

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

LPL Drawback

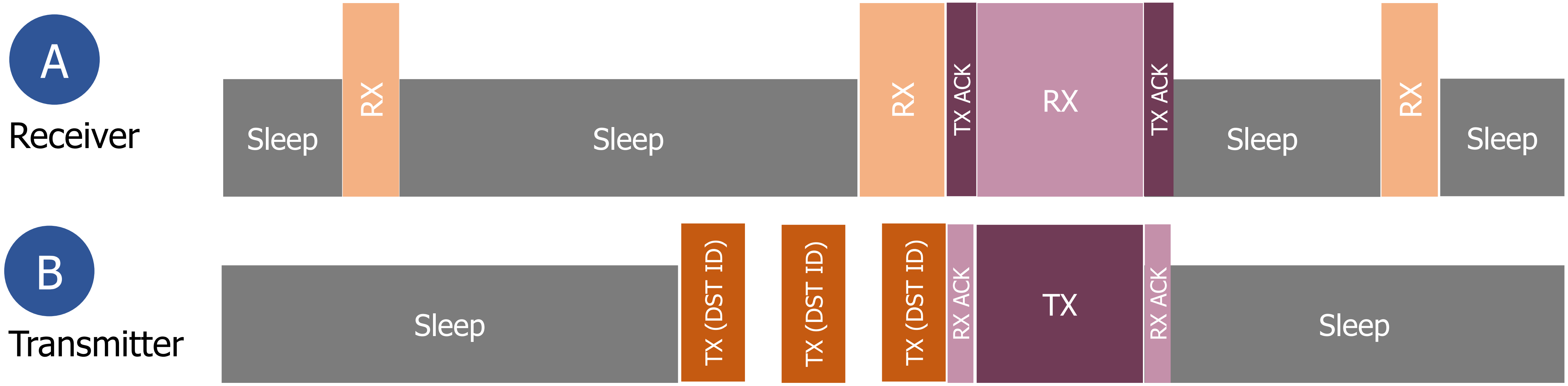


- Spend time listening to packets for someone else!



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

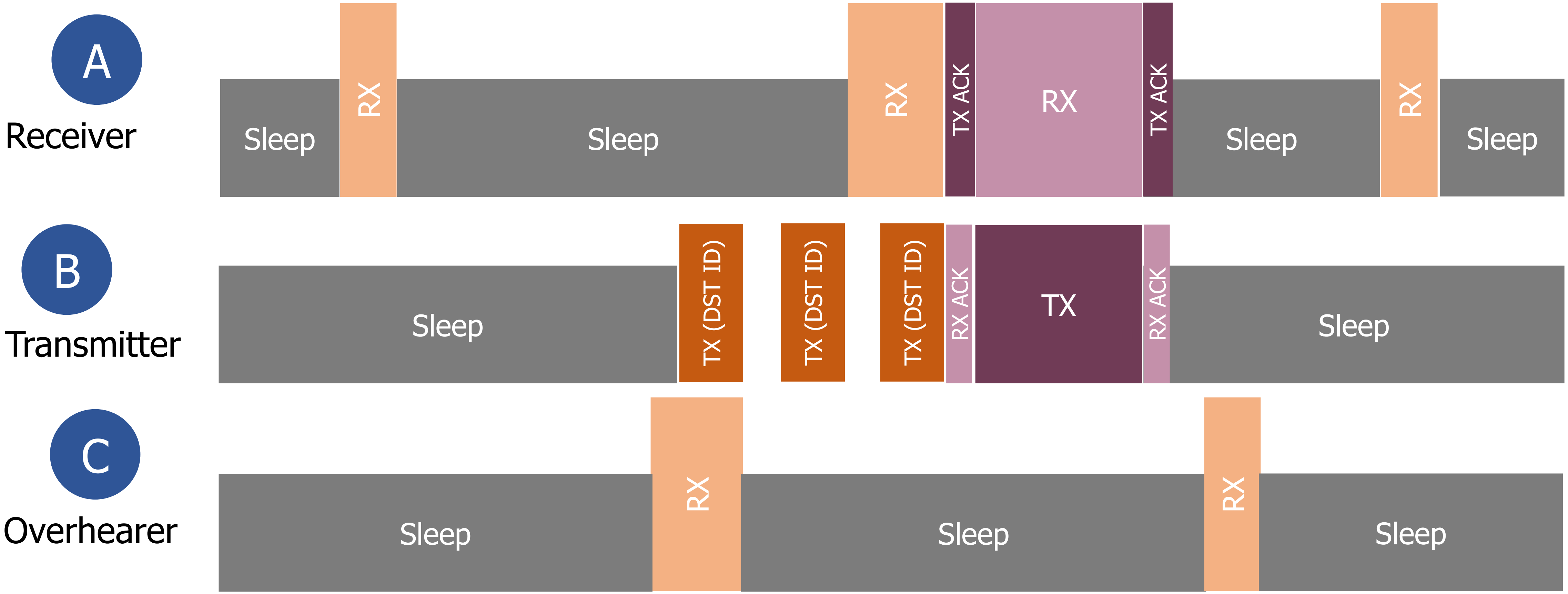
X-MAC: Shorter preambles and destination information



- 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.

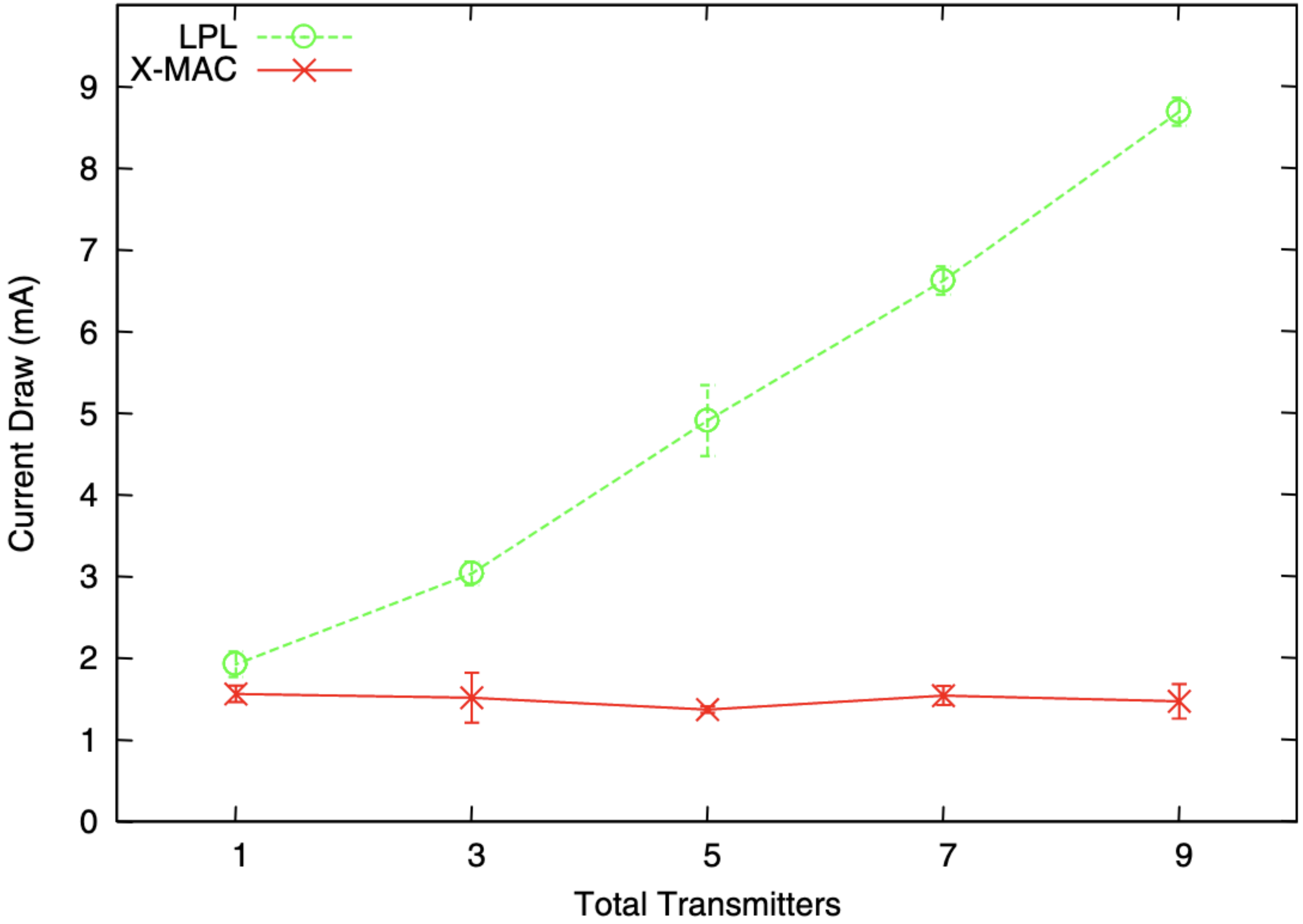
X-MAC: A Short Preamble MAC Protocol for Duty-Cycled Wireless Sensor Networks. Michael Buettner, Gary V. Yee, Eric Anderson, Richard Han. SenSys 2006.

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.

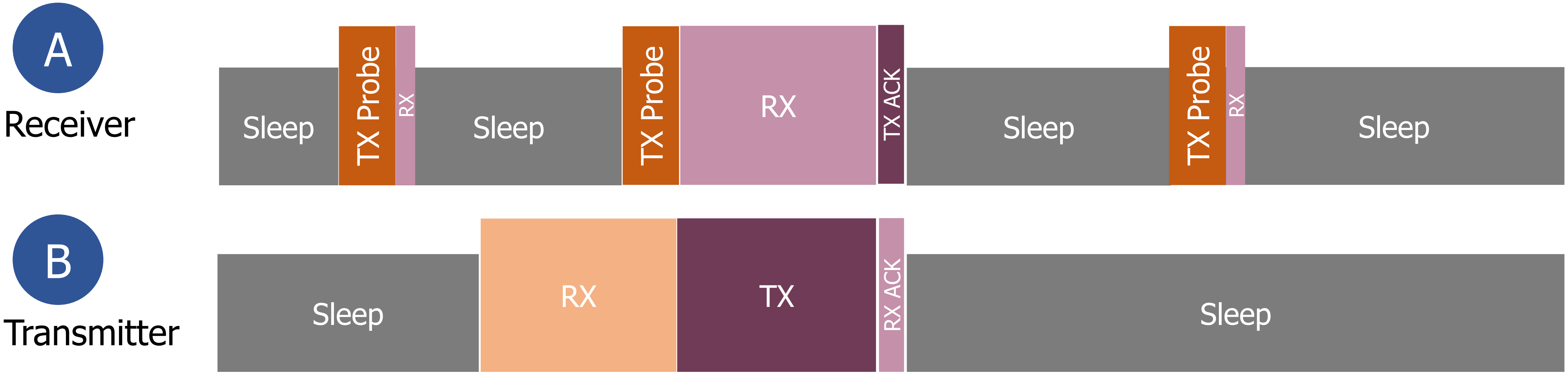
LPL (B-MAC) and X-MAC are transmitter-initiated MAC protocols

- 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

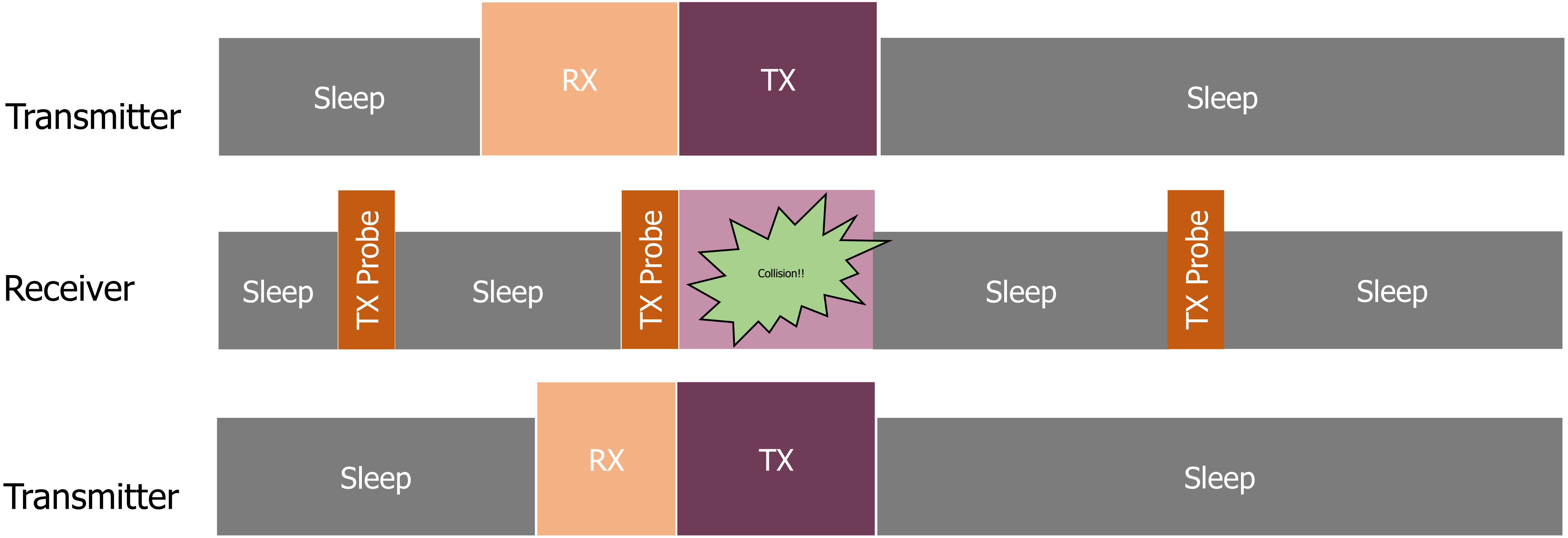
Low Power Probing (LPP) is *receiver* initiated



- 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
  - Works a lot like BLE advertisements!!

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.

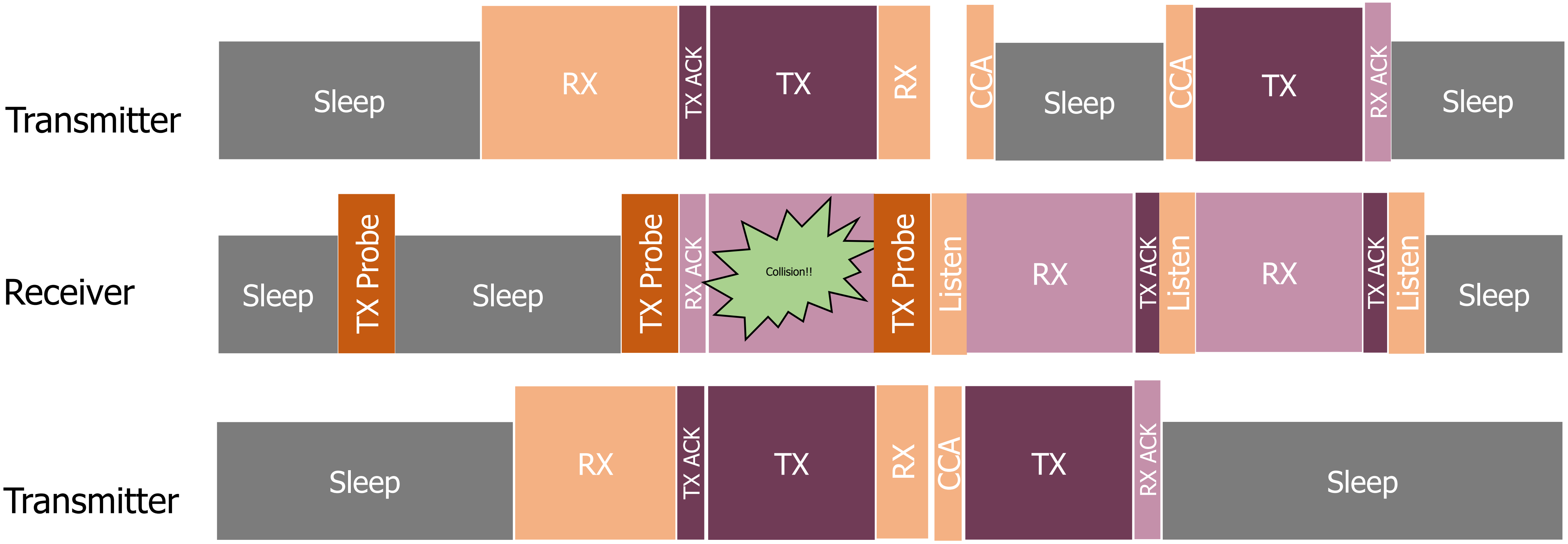
LPP introduces a new challenge for multiple nodes. What happens with multiple transmitters?



- If multiple transmitters receive a probe, they both transmit, leading to a collision and a lost packet
- Receiver is unsure if there even was a packet at all, 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



- 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

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

- Simple Routing
- Mesh Routing
- Better Flooding
- Low-power Access Control