

Lecture 10

IoT Network Routing

CS397/497 – Wireless Protocols for IoT
Branden Ghen a – Winter 2021

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

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

Routing goals

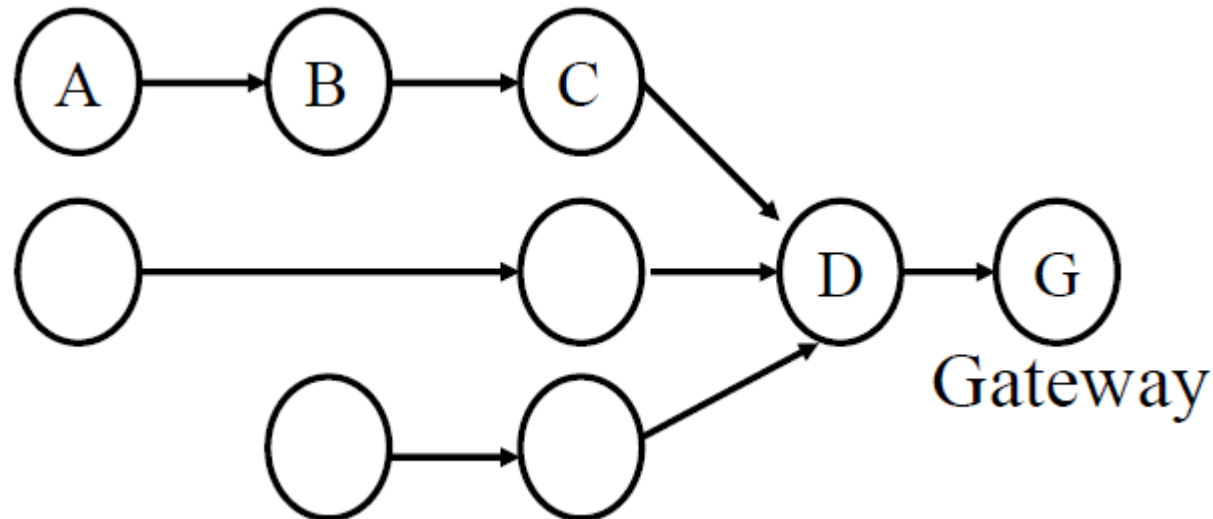
- Have a packet, have a destination, how do we connect them?
- Simple techniques
 - Broadcast, tree structures
- Mesh techniques
 - Understand the available routes and select a “good” one

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

- 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

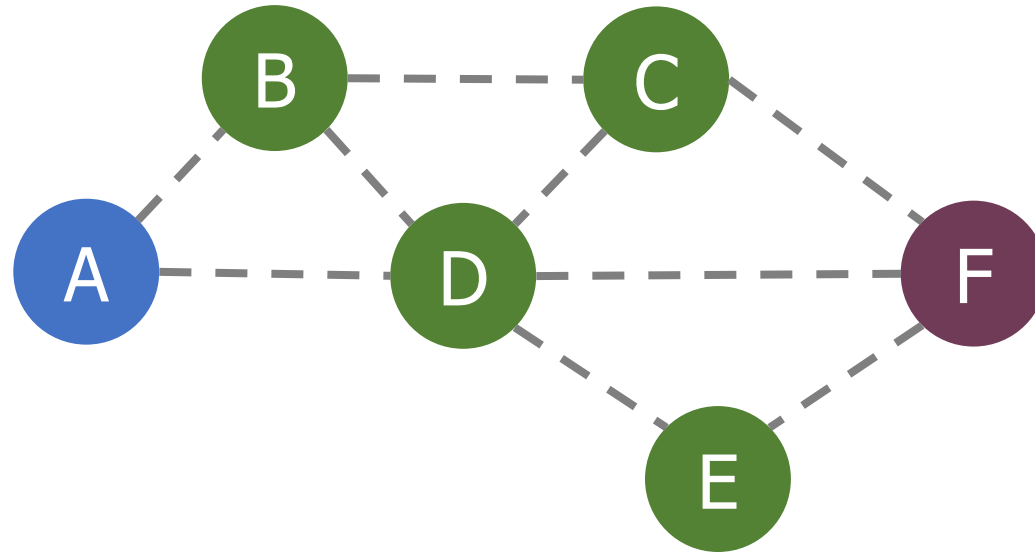


Outline

- Simple Routing
- **Mesh Routing**
- Better Flooding

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
 - 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 Route Requests (RREQs)

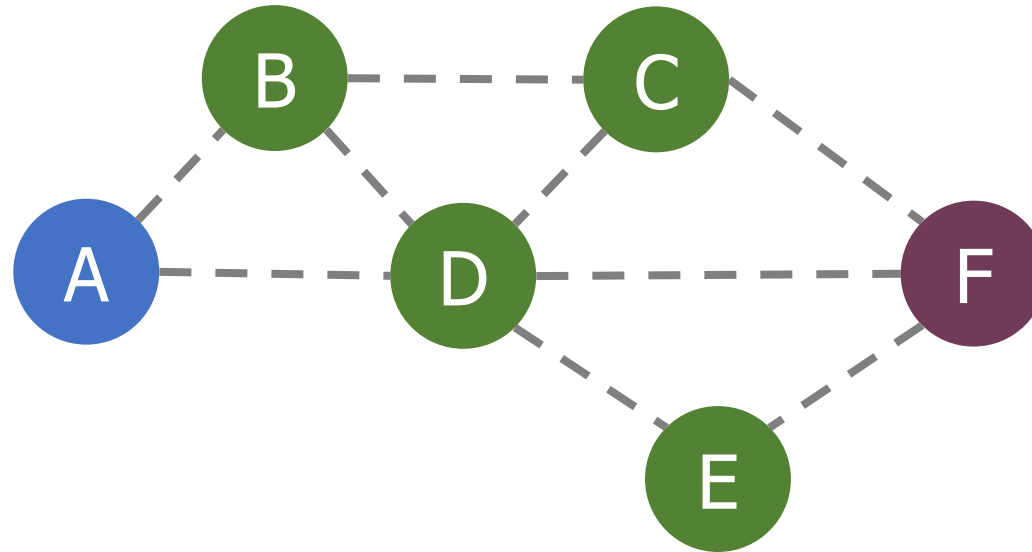
Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count
------------	----------------	---------------------	--------------	-------------------	-----------

- 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

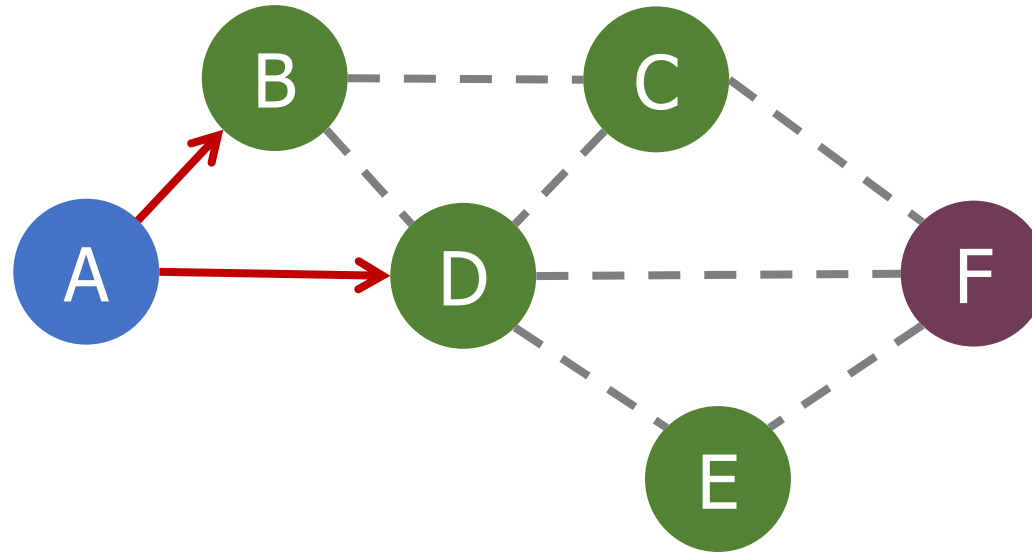
A wants to find a route to F, so it sends out an RREQ



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



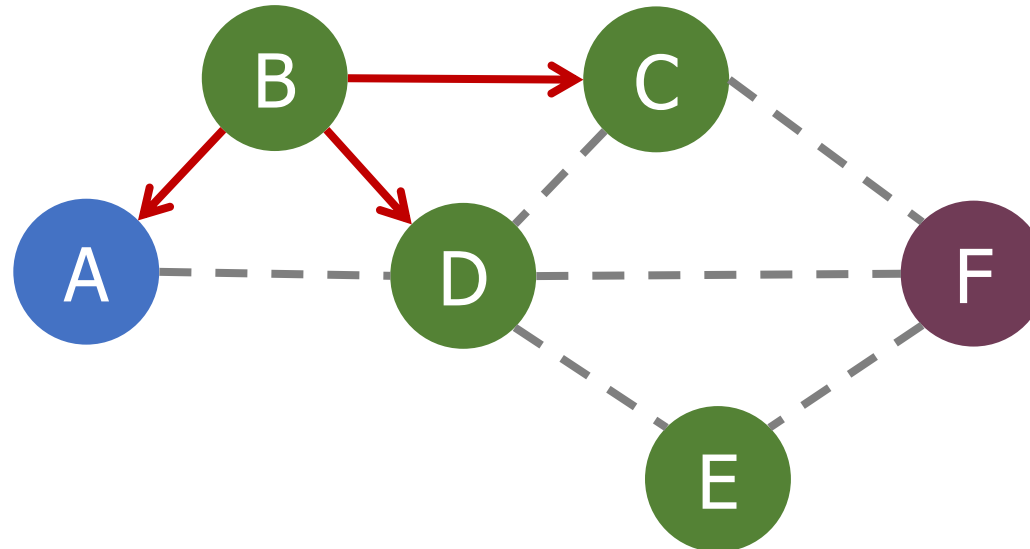
Example AODV RREQ (A to F)

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



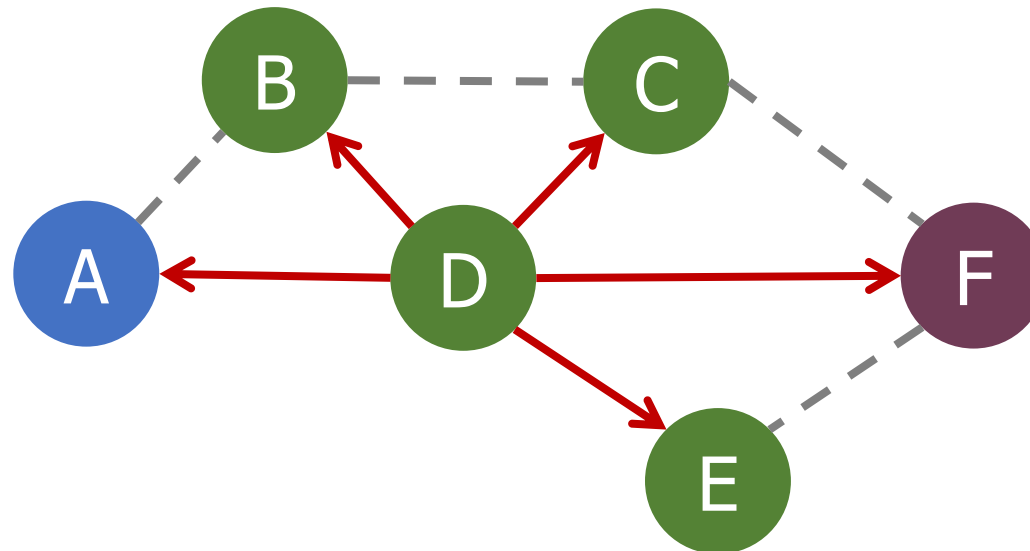
Example AODV RREQ (A to F)

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

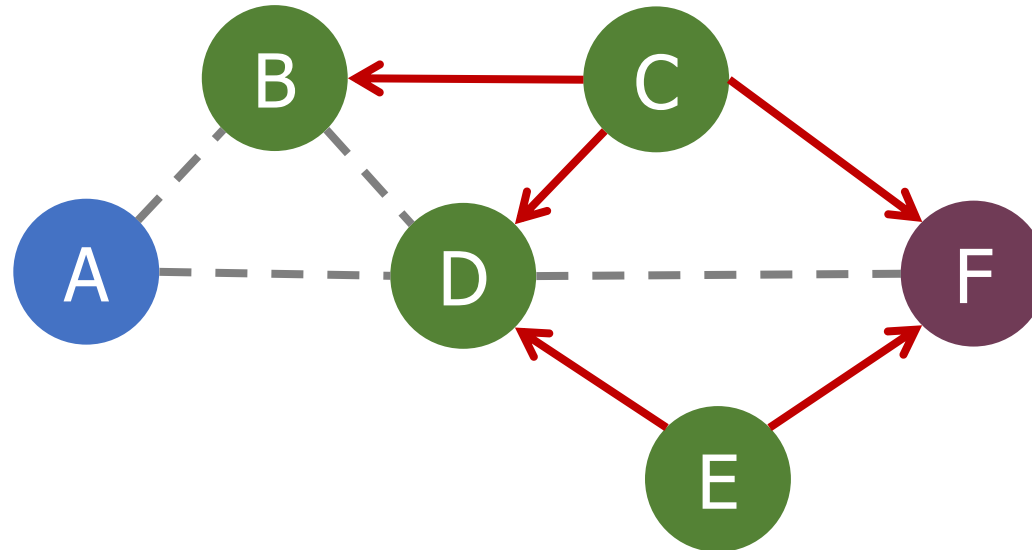


Example AODV RREQ (A to F)

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
-------------------	------------------------	----------------------	--------------

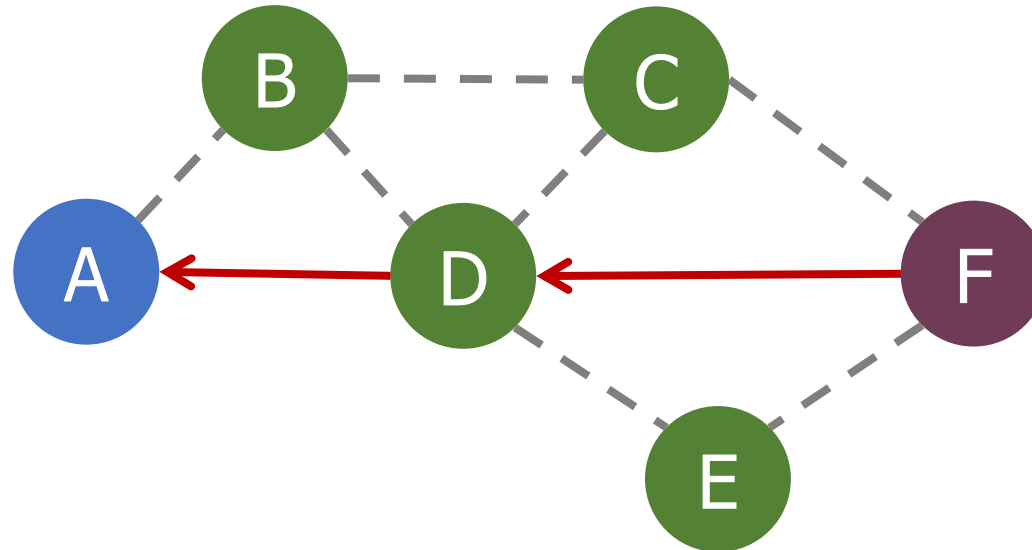
- Reply is sent unicast back to the source via newly constructed route
 - Each node along the way already knows the route back
- Includes most recent destination sequence number as a sense of recency
 - No need for source sequence number anymore

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



RREP optimization

- An intermediate node responds with RREP if it has a path to destination with a more recent Destination sequence number
 - Source may get multiple RREP responses with different recency and hop counts
- Note: we're optimizing for some combination of most recent and lowest hop count

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

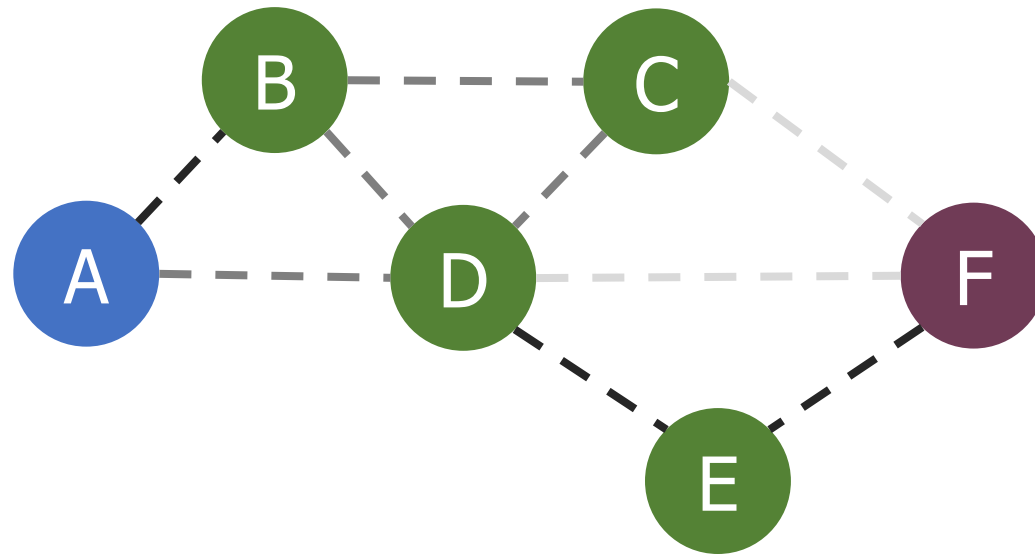
- Keep routes as “next hops” rather than full routes
 - AODV uses this method (DV for Distance Vector)
- Can be combined with proactive techniques too
 - Each router periodically informs neighbors of its shortest paths to each destination (in terms of hop count)
 - Essentially just broadcast your routing 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!

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

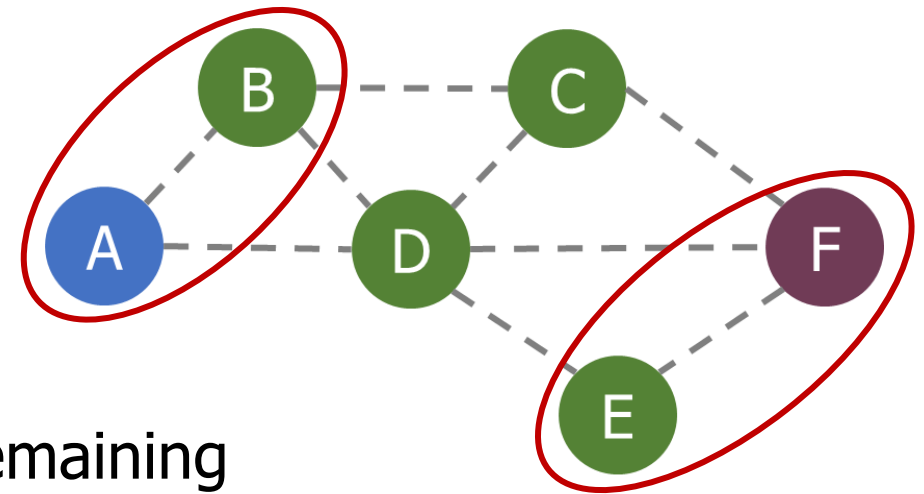
Reliability as a cost metric

- Link quality can vary 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: $A \leftrightarrow B$ and $E \leftrightarrow F$



- 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**

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

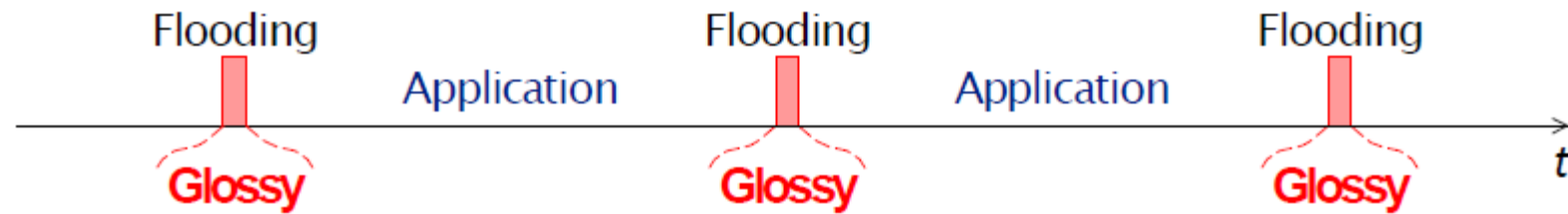
Federico Ferrari, Marco Zimmerling, Lothar Thiele, Olga Saukh

Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland

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

Glossy key techniques

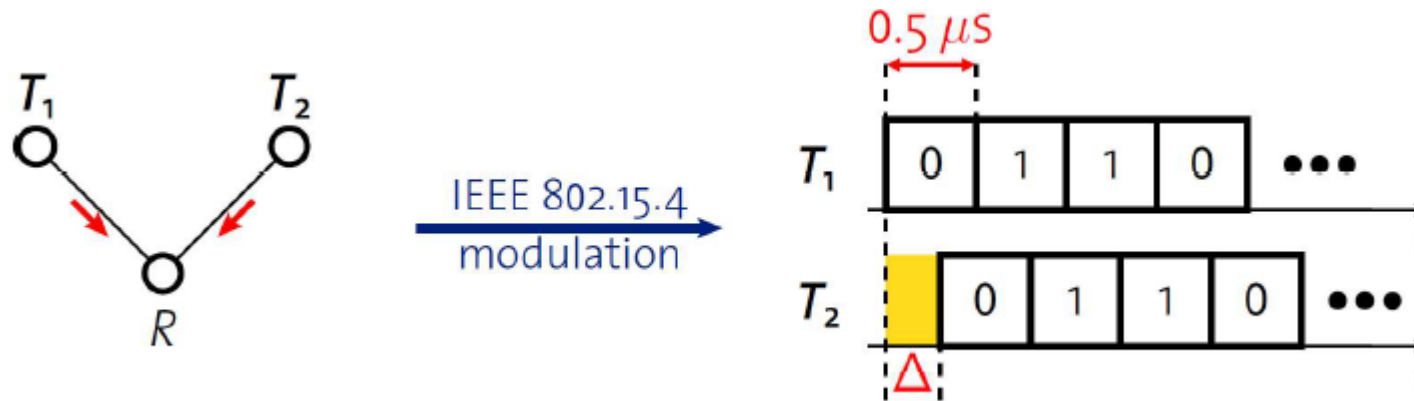
- Temporally decouple network flooding from application tasks



- Exploit synchronous transmissions for fast network flooding

Synchronous transmissions

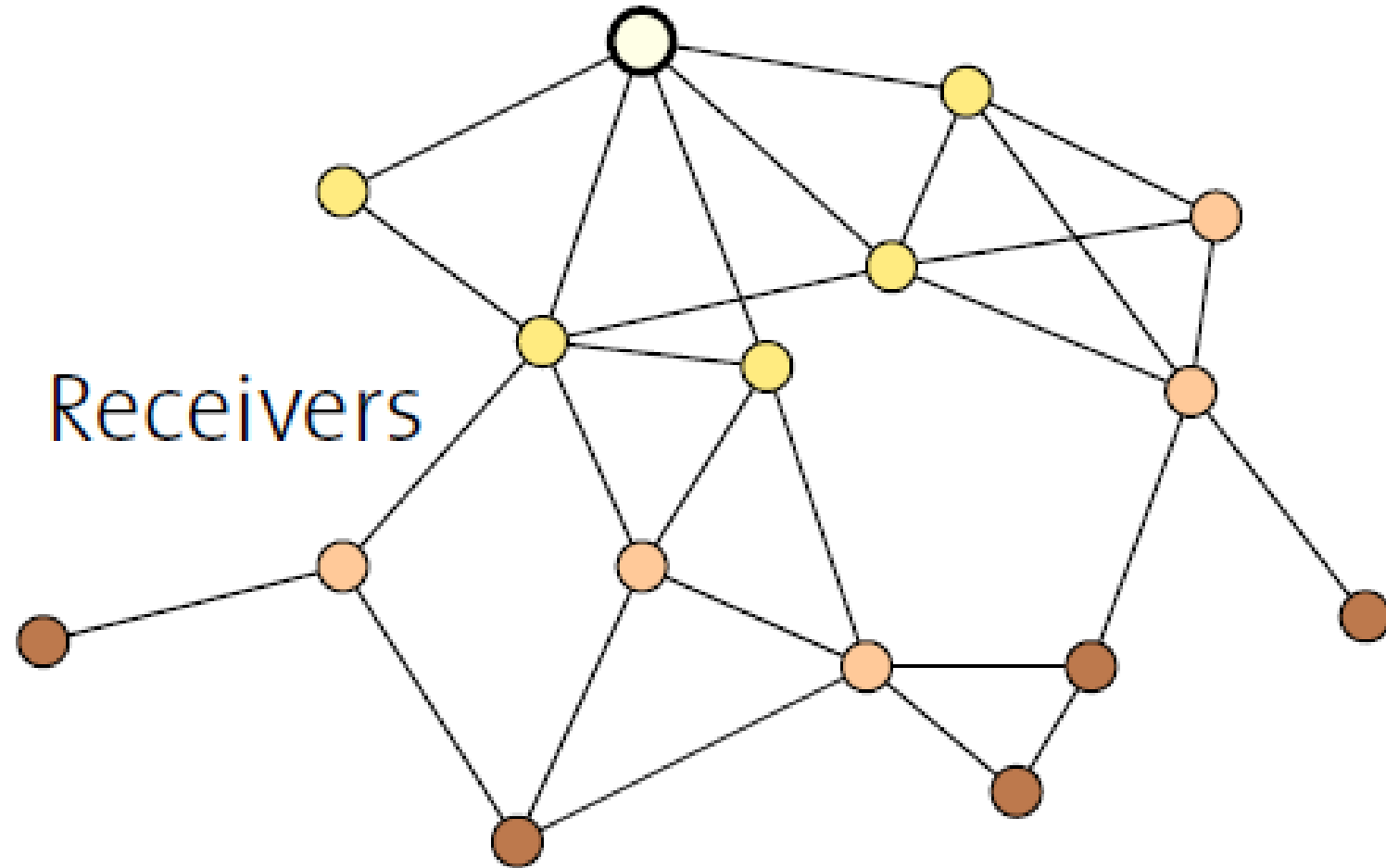
- Multiple nodes transmit **same packet** at **same time**



- 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)

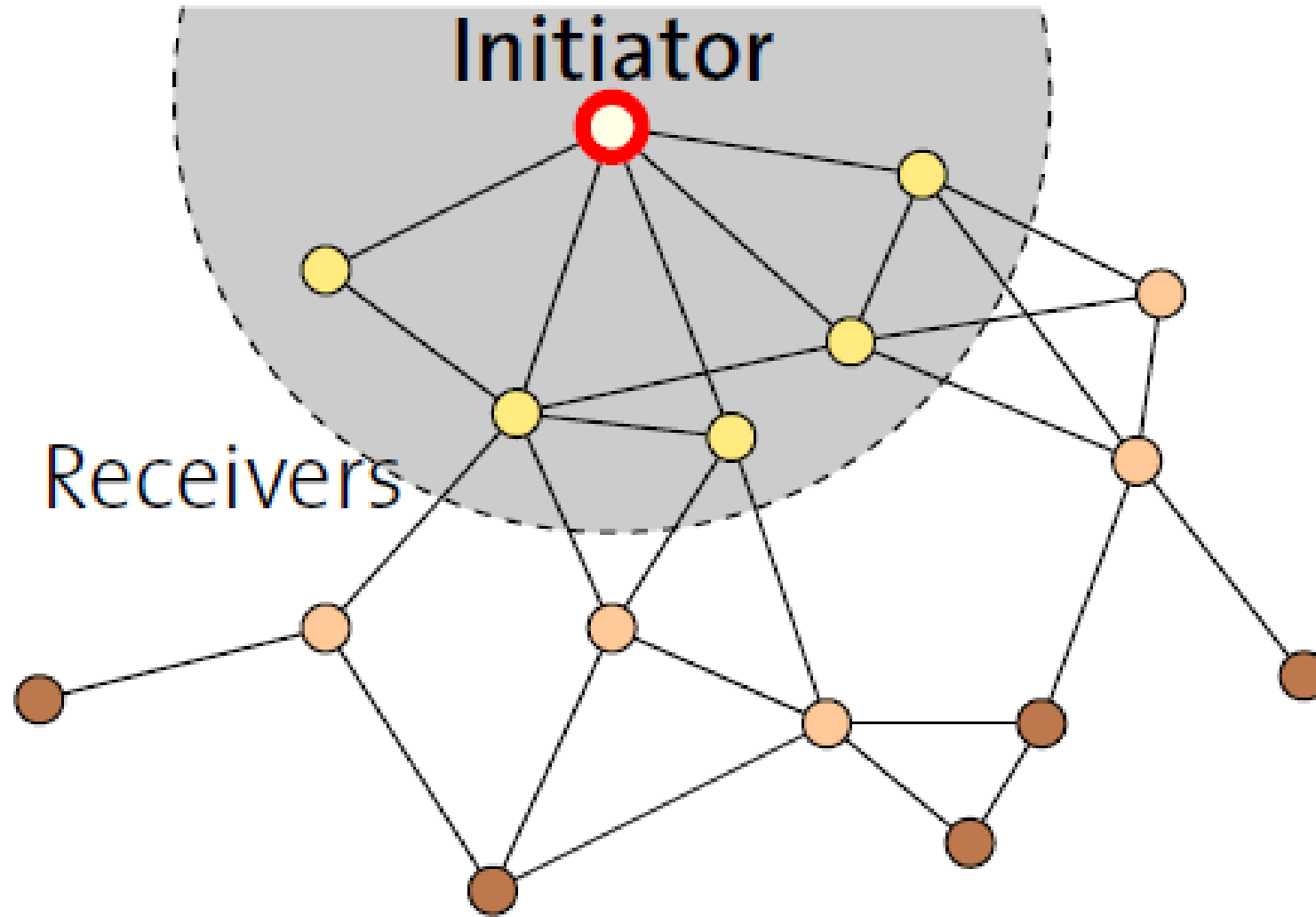
Fast packet propagation in Glossy

Initiator

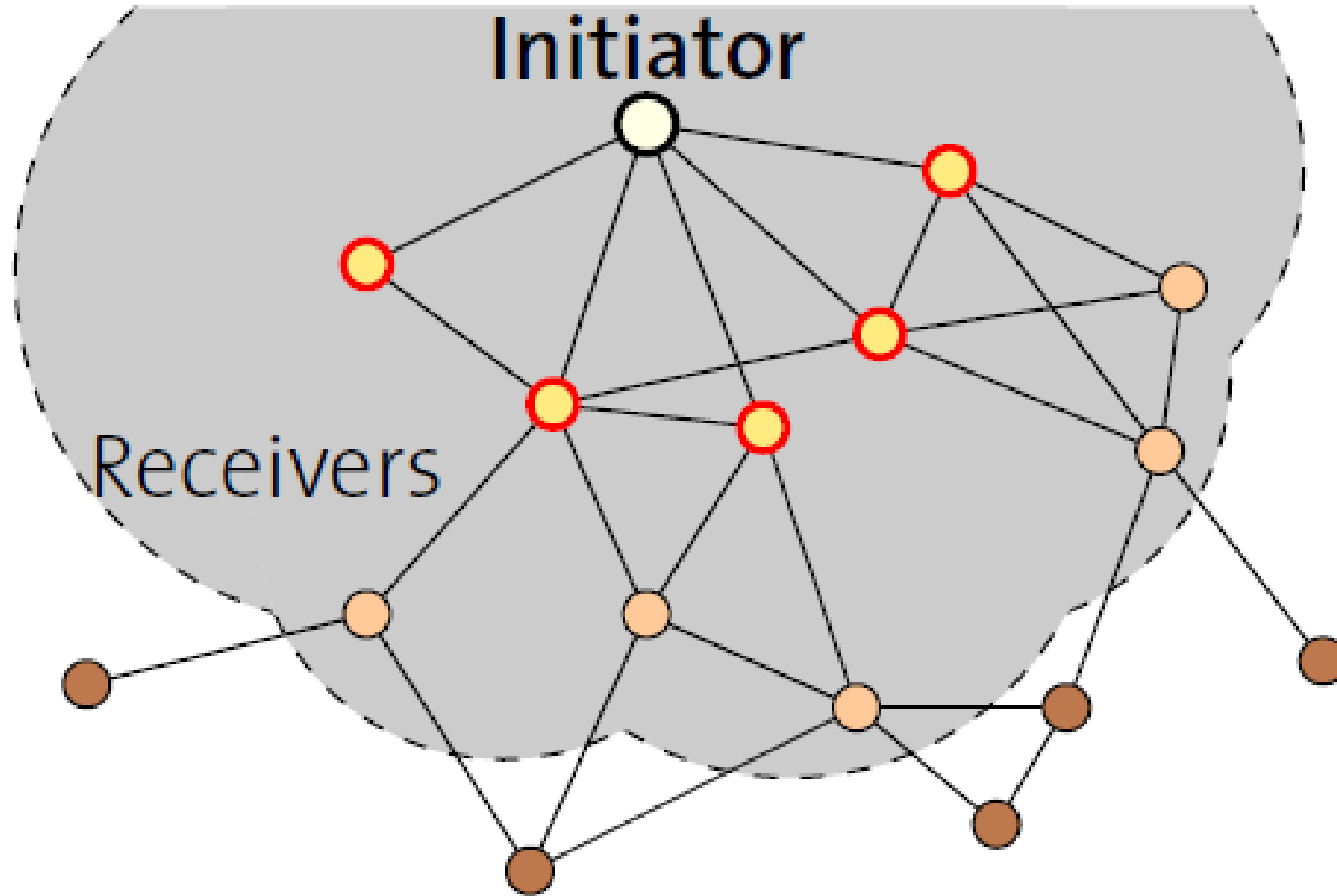


Receivers

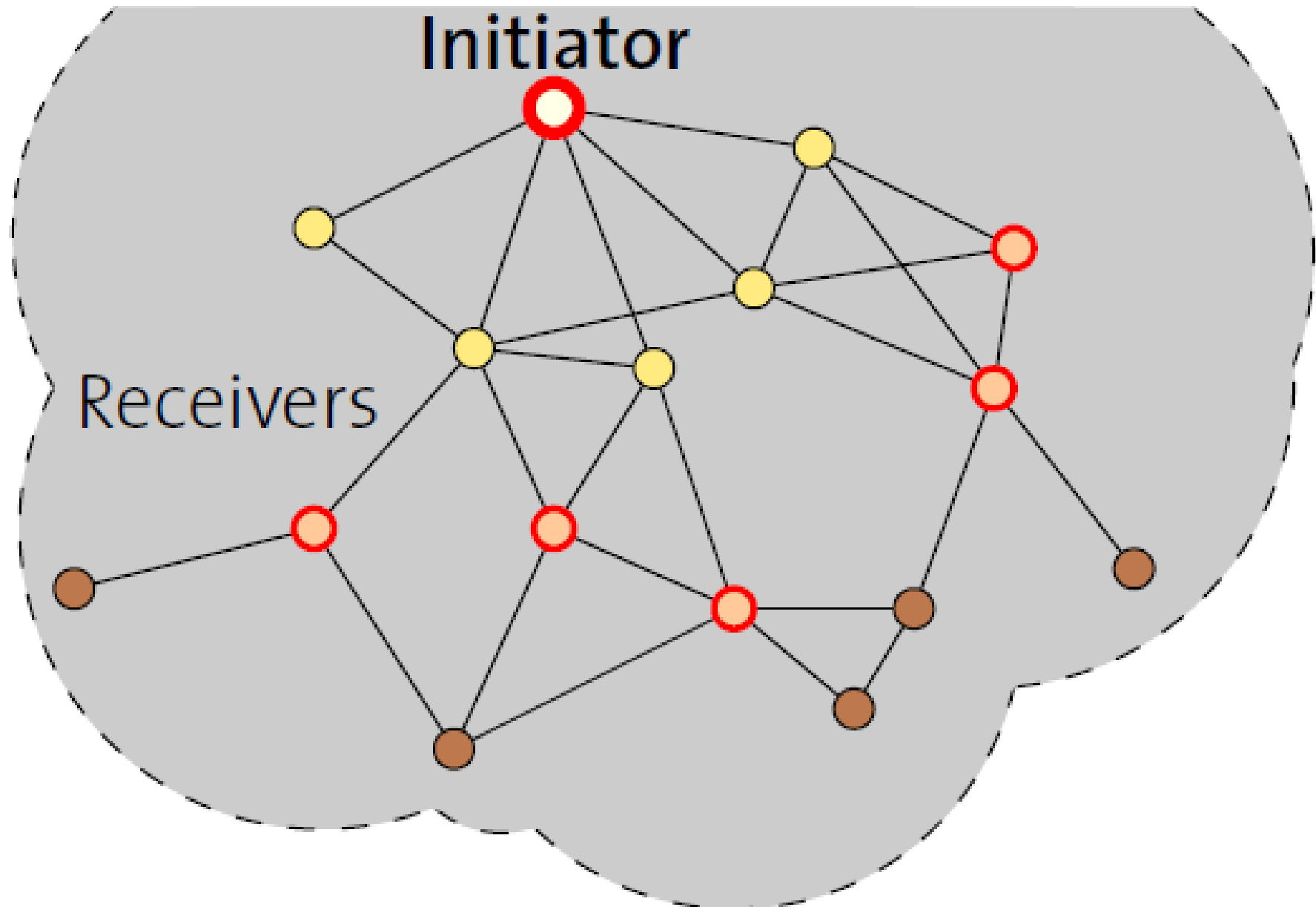
Fast packet propagation in Glossy



Fast packet propagation in Glossy



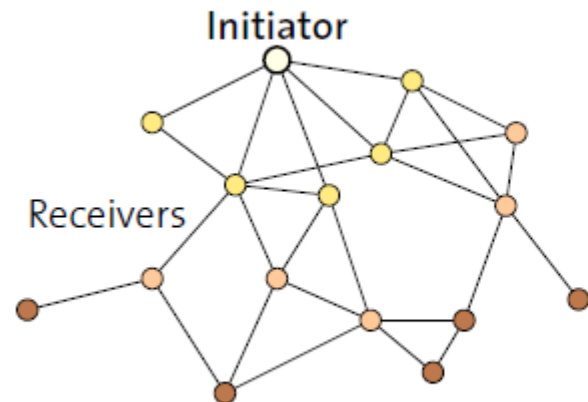
Fast packet propagation in Glossy



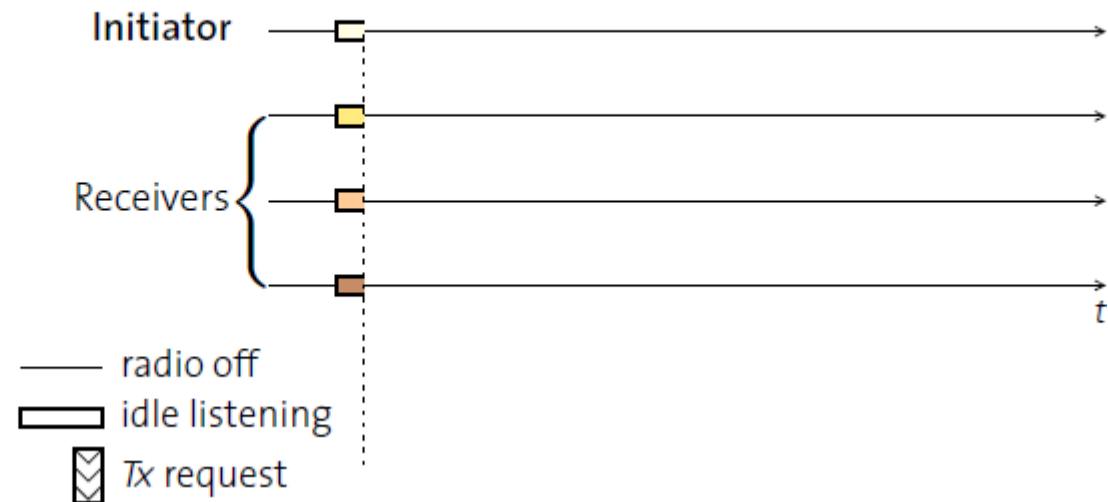
Glossy details

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

Example



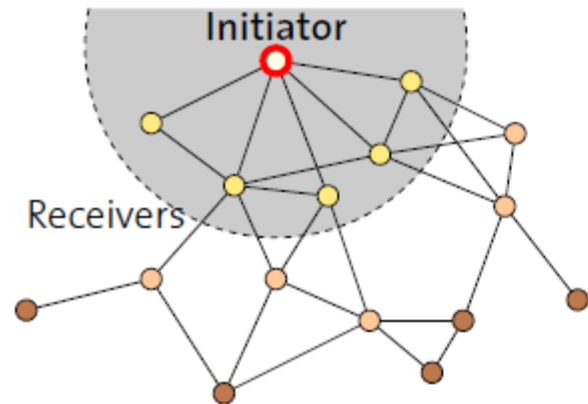
Timeline



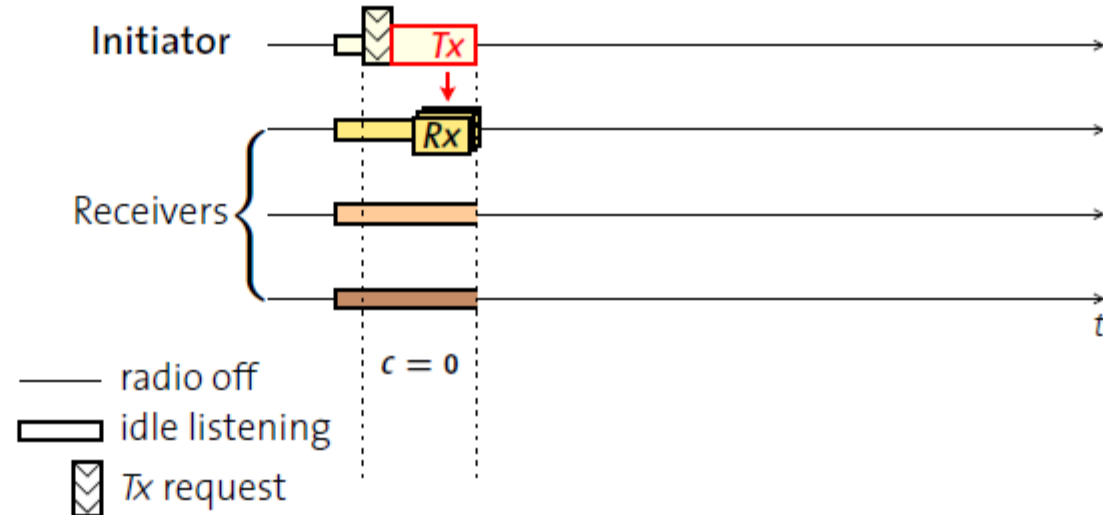
Glossy details

- Initiator
 - Set relay counter in packet, $\mathbf{C} = 0$
 - Broadcast packet

Example



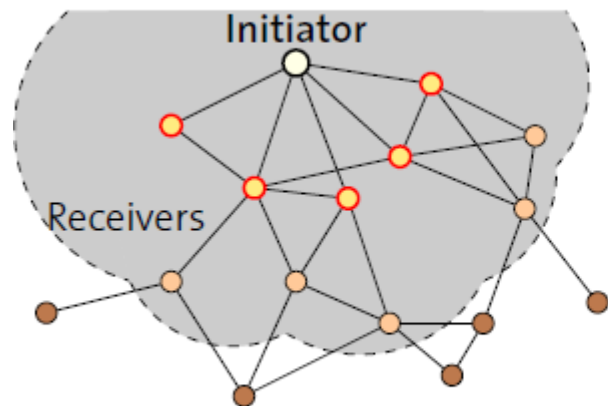
Timeline



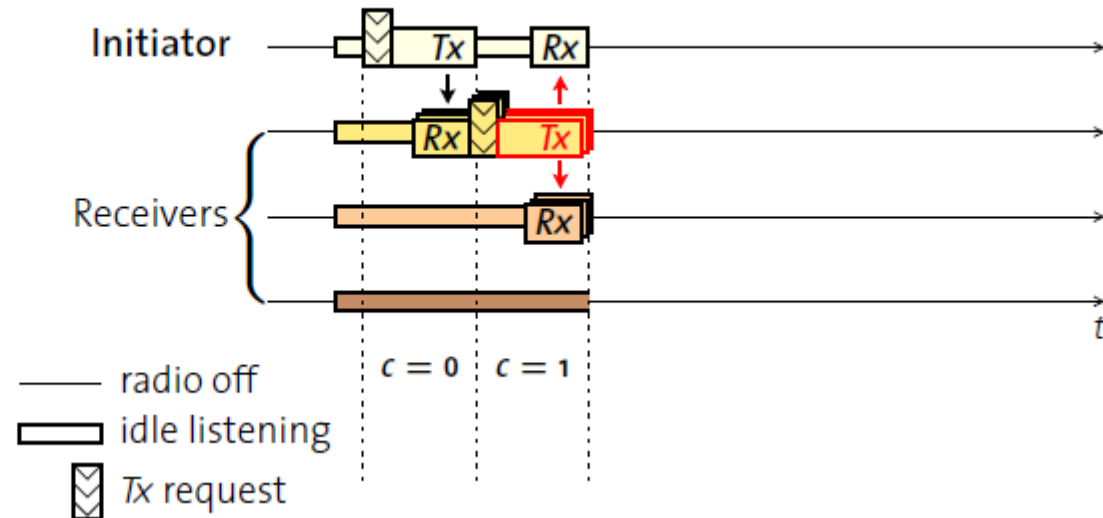
Glossy details

- At packet reception:
 - Increment relay counter \mathbf{C}
 - Transmit synchronously (at a fixed period after the reception)

Example



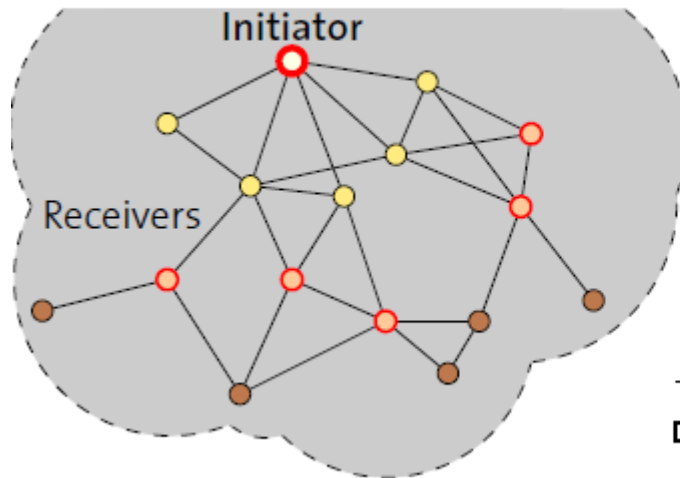
Timeline



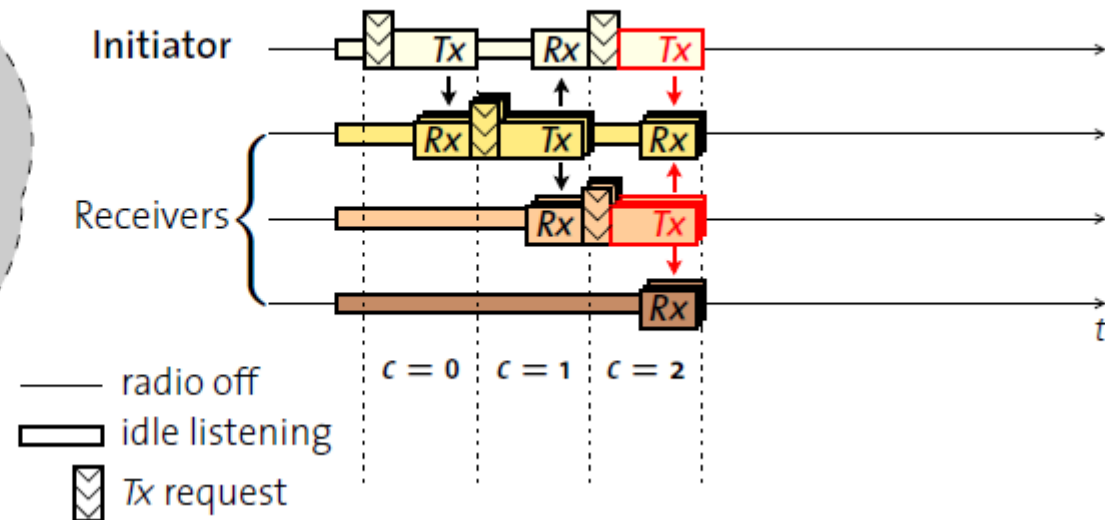
Glossy details

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

Example



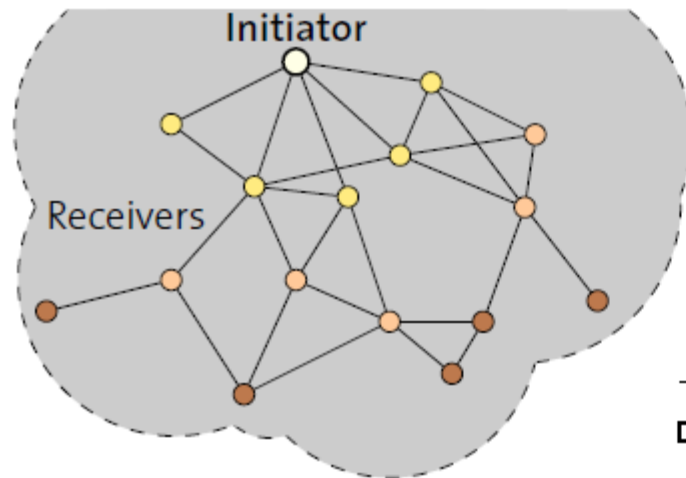
Timeline



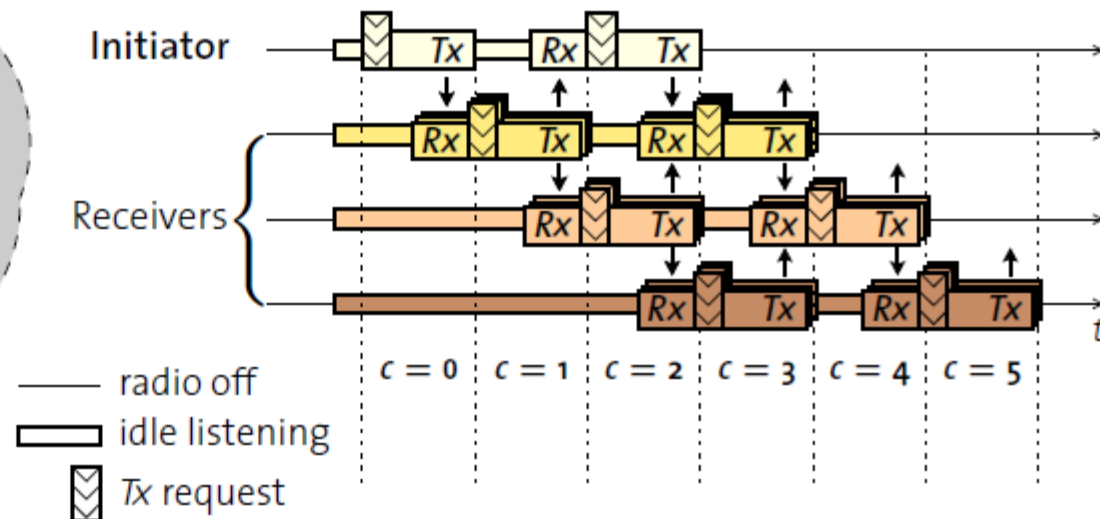
Glossy details

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

Example



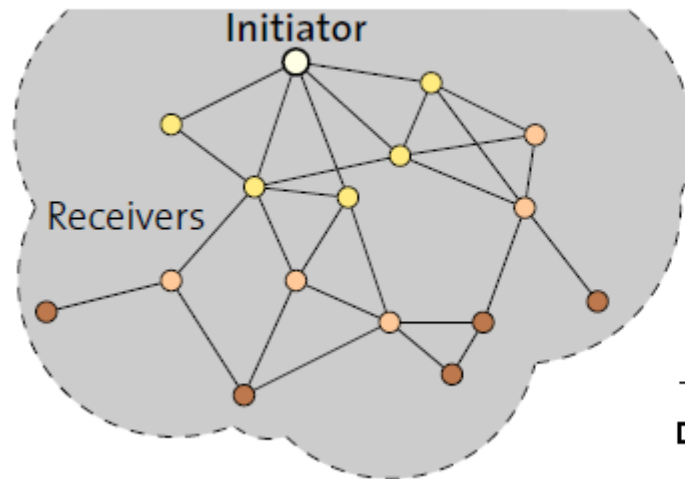
Timeline ($N = 2$)



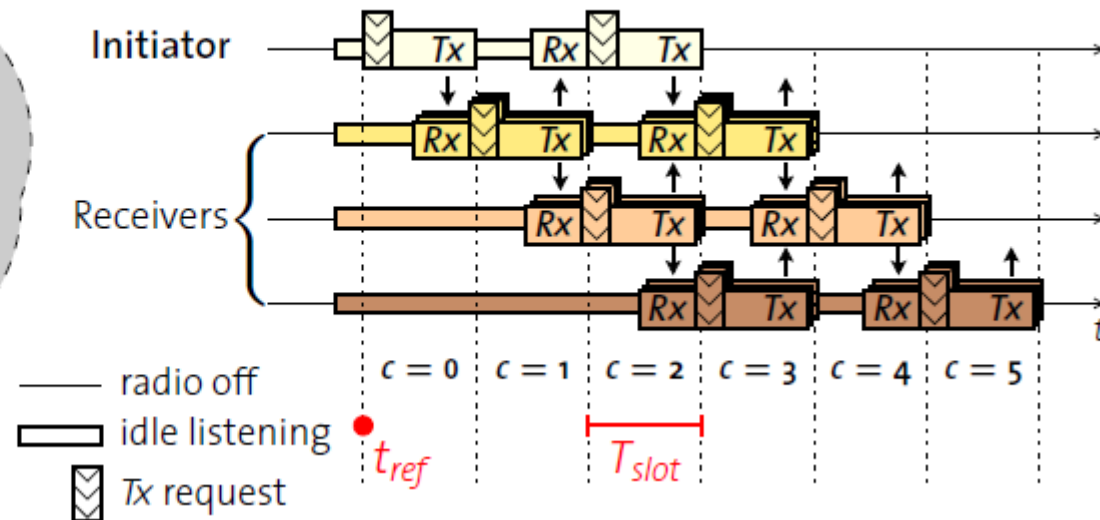
Glossy details

- 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

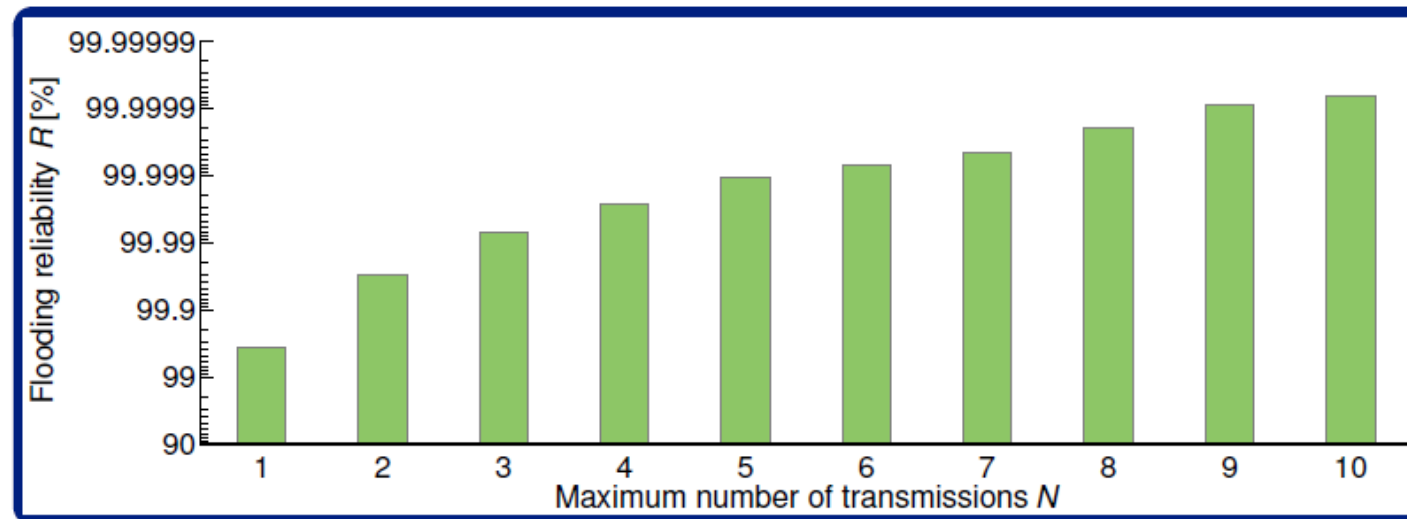


Timeline ($N = 2$)



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



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