CSE 291: Wireless and Communication in the Internet of Things

Zigbee; Mesh Routing; Flooding

Pat Pannuto, UC San Diego

ppannuto@ucsd.edu

Today's Goals (are ambitious)

- Quick coverage of Zigbee
- Routing in mesh networks
- Flooding as a routing replacement

Outline

- Zigbee
- The Interoperability Problem
- Routing
- Flooding

ZigBee goals

- Enable automatic communication between devices
  - Low complexity
  - Low power
  - Focus on home automation and industrial control/monitoring
- From our perspective
  - 802.15.4 PHY and MAC
  - Plus well-defined Server/Client interactions
    - Similar to BLE (actually, BLE is similar to ZigBee)
  - Designed for higher-power devices than Thread or BLE
    - Although still relatively low power

ZigBee history

- Intertwined with the creation of 802.15.4
  - Both are founded around the same time
  - ZigBee Alliance involved in the original 802.15.4 specification
  - Original plan: 802.11/WiFi <-> 802.15.4/ZigBee

- Original specification 2004 (following 802.15.4 in 2003)
  - Updated 2006, 2007, 2015, (2017?)
  - 2015 version is also known as ZigBee Pro
  - We'll focus on 2015, but look at previous stuff too
    - Application layer stuff hasn't changed considerably

ZigBee resources

- ZigBee Specification (2015)
- ZigBee Cluster Library Specification (2016)

- Useful resources
  - ZigBee overview: https://www.cse.wustl.edu/~jain/cse574-14/ftp/j_13zgb.pdf
  - NXP library guides (include overview on ZigBee)
    - ZigBee Protocol: https://www.nxp.com/docs/en/user-guide/JN-UG-3113.pdf
    - ZigBee Cluster Library: https://www.nxp.com/docs/en/user-guide/JN-UG-3115.pdf
    - ZigBee Home Automation: https://www.nxp.com/docs/en/user-guide/JN-UG-3076.pdf

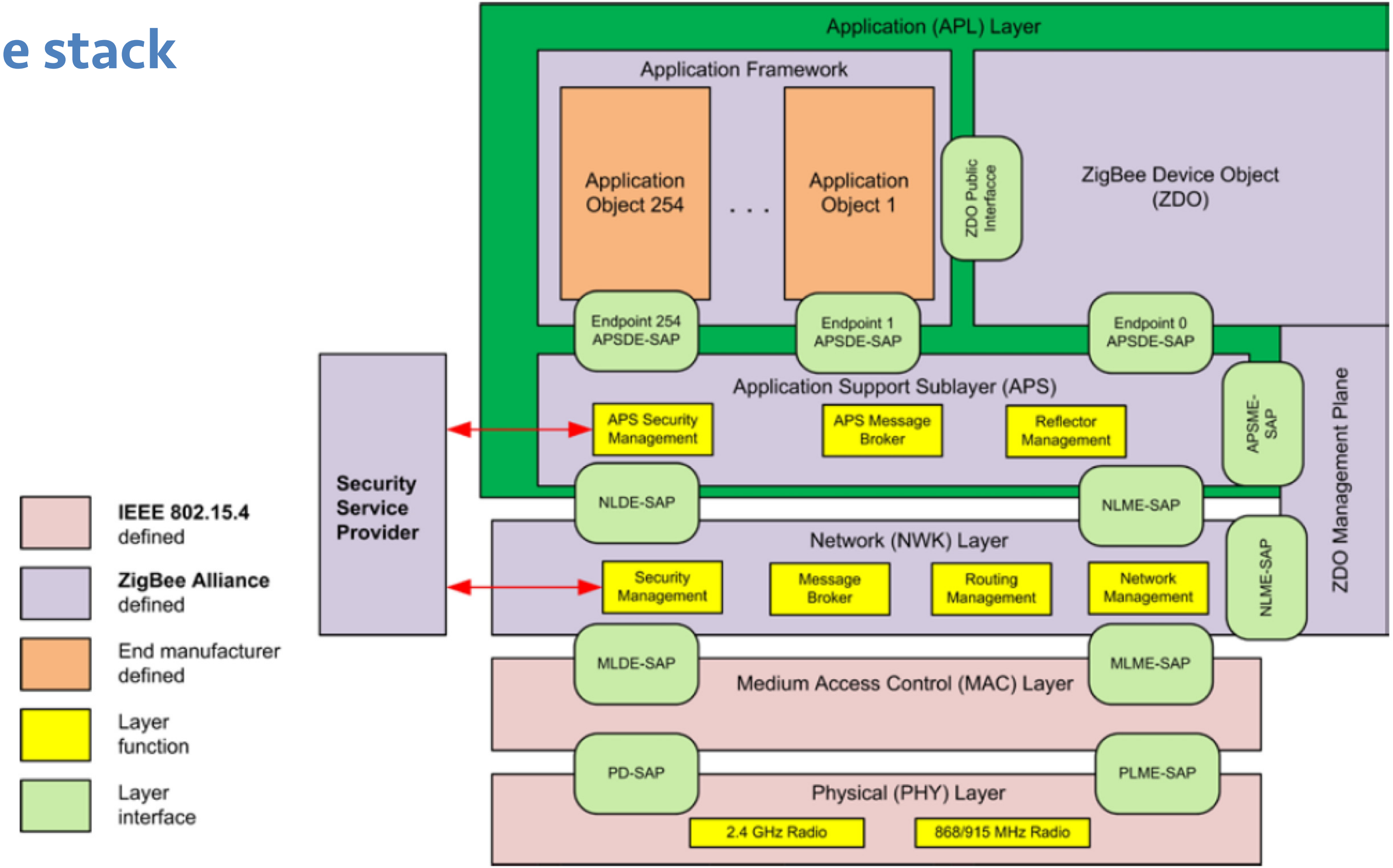
~~Zigbee~~ Connectivity Standards Alliance today

- 2021 rebrand (why?)
  - Zigbee fading in relevance, utility
  - Zigbee group creat[ed/ing] new standard: Matter
    - Announced Dec 2019; "first products expected early 2022"
- Setting up as a Thread competitor, focused on nailing Smart Home

Zigbee tl;dr: It's approximately an application stack on top of the IEEE 802.15.4 standard we talked about before

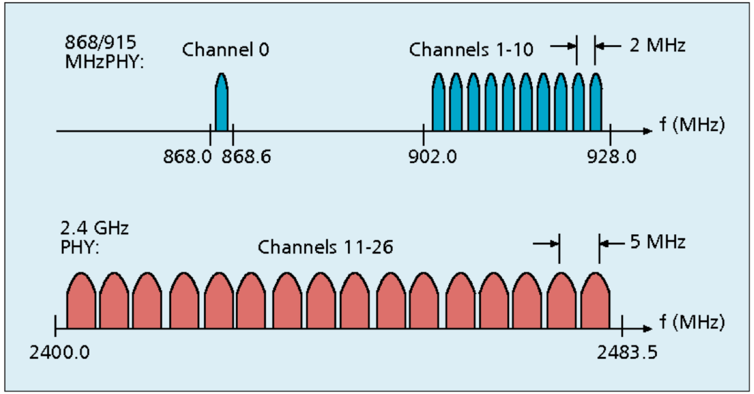
[In the interest of time, skipping presentation of most slides in this section]

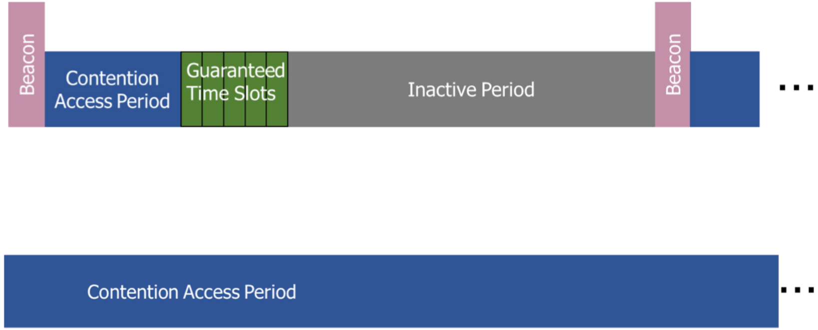
ZigBee stack



Use of 802.15.4

- Basic answer: everything
  - Reuse all of PHY (including non-2.4 GHz channels)
  - Reuse all of MAC (including beacon-enabled network and GTS)
    - Same CSMA/CA mechanism





ZigBee devices (same roles as 802.15.4 defines)

- ZigBee Coordinator (ZC)
  - Starts the network and decides on key parameters
  - Is also a Router
- ZigBee Router (ZR)
  - Higher-power, more-capable devices
  - Radios always on (except during inactive superframe)
  - Connect to one or more children
  - Connect to one or more routers
- ZigBee End Device (ZED)
  - Lower-power, less-capable devices
  - Always a child of one router

Older ZigBee - tree networks

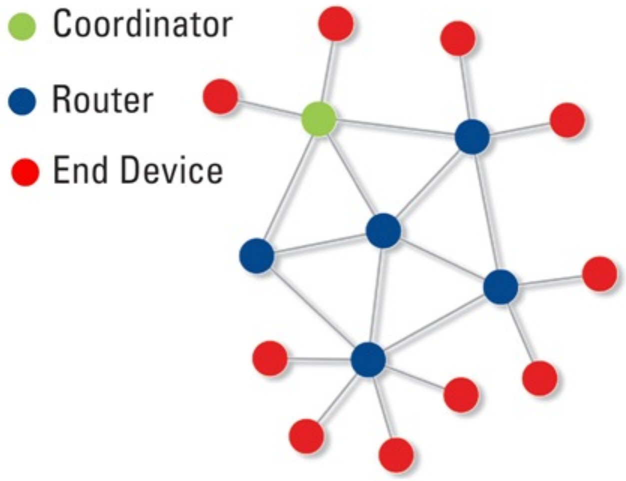
- Original preferred topology
- Uses beacon-enabled network
  - Synchronization via beacon superframes
  - Can reduce power requirements for routers
- Some things get simpler
  - Address assignment is simple
    - If you restrict network size
  - Routing is straightforward
    - But likely more hops for router-to-router communication

ZigBee tree network complications

- Distributed routing scheme limits topologies
  - There is a limit on number of routers
  - Each router has a maximum number of children
  - There is a maximum limit for router depth
  - Note: Thread has device count limits too!

- Needs a beacon scheduling mechanism
  - Each parent must both participate in a superframe
  - And also send their own superframe beacons
  - Need to keep inactive period large if there is significant router depth
  - Each beacon includes a TX offset field specifying parent beacon time
    - Helps prevent hidden terminal problem

Modern ZigBee – mesh networks



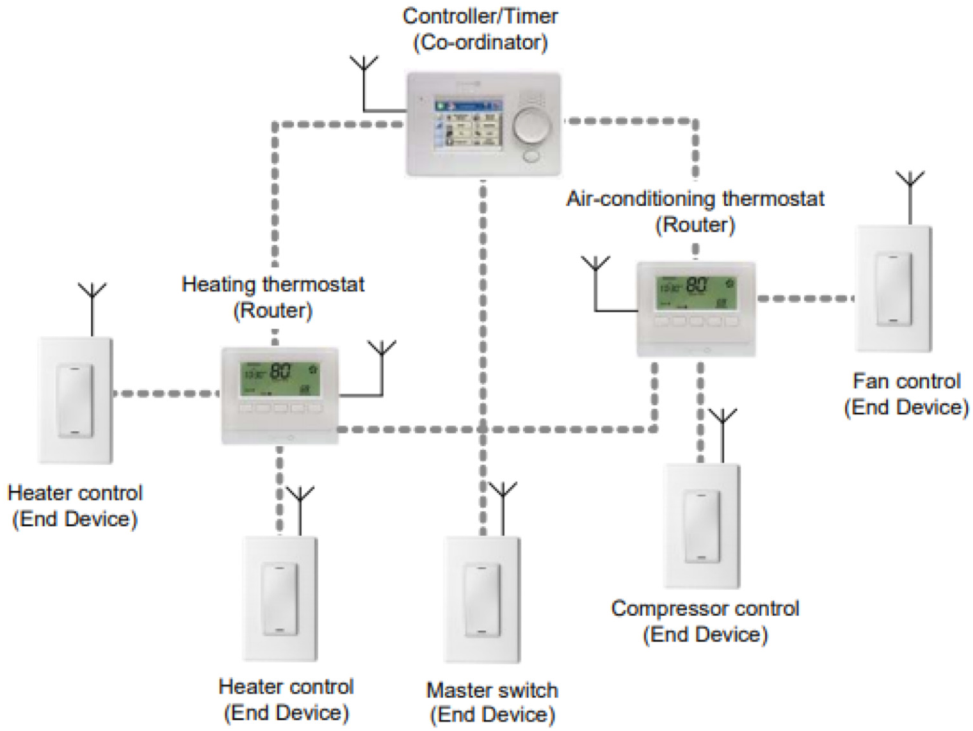
- Presently preferred topology
- Uses non-beacon-enabled network
  - All routers are always-on devices
  - Allows arbitrary communication between routers
- Some tradeoffs
  - Likely higher power routers
  - Routing more complicated (potentially better algorithms though)
  - Addressing more complicated
    - Assign random addresses to each node
    - Include a method for address conflict resolution

ZigBee End Device polling

- Packets are held in ZigBee Routers for up to 7.68 seconds
  - Compare to undefined duration for Thread (at least minutes)
  - Reduction in "low energy" capability for end devices
  - Limiting timeouts makes Router design simpler

- ZigBee codifies polling behavior for End Devices
  - Long Polling - steady state polling period, example: 7.5 seconds
  - Short Polling – polling period while waiting on data, example: 1 second

Example ZigBee network



ZigBee application-layer terms

- Devices act as servers and clients

- Profiles – details application-level features
  - Includes network configurations
    - For example: security or reliability

  - Includes definitions of various Device Types
    - Specify a collection mandatory and optional Clusters
    - Clusters – collection of Attributes and Commands
      - Attributes – information, readable and/or writable
      - Commands – control, writable, may elicit a response

Analogies between BLE and ZigBee

- No analogy
- BLE Profile
- BLE Service
- BLE Characteristic

- ZigBee Profile
- ZigBee Profile + Device
- ZigBee Cluster
- ZigBee Attribute
- Also ~ZigBee Commands

ZigBee profiles

- Broad classes of device purposes
  - Contains multiple Device Type definitions

Profile ID	Profile Name
0101	Industrial Plant Monitoring (IPM)
0104	Home Automation (HA)
0105	Commercial Building Automation (CBA)
0107	Telecom Applications (TA)
0108	Personal Home & Hospital Care (PHHC)
0109	Advanced Metering Initiative (AMI)

- Define more features of device than the profiles from BLE
  - Pick various optional network/MAC features, like security or commissioning

ZigBee Device Types

- A collection of Clusters
  - Some mandatory and some optional

- Lists Clusters as Server side or Client Side
  - Server side Cluster is an *input*
  - Client side Cluster is an *output*

- Example: light bulbs implement server, switches implement client

ZigBee Clusters

- A collection of Attributes and Commands
  - Analogous to BLE Services
  - Can be optional or mandatory

- ZigBee Cluster Library defines standard Clusters
  - Lists Attributes and Commands for each
  - Attributes
    - Type – uint8, enum, bitmap, string, etc.
    - Permissions - Read/Write/Report (receive automatic updates)
    - How to interpret meaning of value
  - Commands
    - Field(s), Type of each, Interpretation of each

Example ZigBee profile: Home Automation Device Types

Generic Devices

- On/Off Switch
- On/Off Output
- Remote Control
- Door Lock
- Door Lock Controller
- Simple Sensor
- Smart Plug

Intruder Alarm System Devices

- IAS Control and Indicating
- IAS Ancillary Control
- IAS Zone
- IAS Warning Device

- Lighting
- On/Off Light
- Dimmable Light
- Colour Dimmable Light
- On/Off Light Switch
- Dimmer Switch
- Colour Dimmer Switch
- Light Sensor
- Occupancy Sensor

HVAC Devices

- Thermostat

Each bullet point is a **Device Type**

Which is a list of mandatory and optional Clusters

Example Device Types: door lock and door lock controller

Server (Input) Side	Client (Output) Side
Mandatory	
Basic	
Identify	
Door Lock	
Scenes	
Groups	
Optional	
See Table 1 on page 26	See Table 1 on page 26
Alarms	Time
Power Configuration	OTA Bootload
Poll Control	

Table 6: Clusters for Door Lock

Server (Input) Side	Client (Output) Side
Mandatory	
Basic	Door Lock
Identify	Scenes
	Group
	Identify
Optional	
See Table 1 on page 26	See Table 1 on page 26

Table 7: Clusters for Door Lock Controller

Example Cluster: door lock attributes

Identifier	Name	Type	Access	Def	M/O
0x0000	*LockState*	enum8	Read Only Reportable	-	M
0x0001	*LockType*	enum8	Read Only	-	M
0x0002	*ActuatorEnabled*	bool	Read Only	-	M
0x0003	*DoorState*	enum8	Read Only Reportable	-	O
0x0004	*DoorOpenEvents*	uint32	Read/Write	-	O
0x0005	*DoorClosedEvents*	uint32	Read/Write	-	O
0x006	*OpenPeriod*	uint16	Read/Write	-	O
0x0010	*NumberOfLogRecordsSupported*	uint16	Read Only	0	O
0x0011	*NumberOfTotalUsersSupported*	uint16	Read Only	0	O
0x0012	*NumberOfPINUsersSupported*	uint16	Read Only	0	O
0x0013	*NumberOfRFIDUsersSupported*	uint16	Read Only	0	O
0x0014	*NumberOfWeekDaySchedulesSupportedPerUser*	uint8	Read Only	0	O
0x0020	*EnableLogging*	bool	Read*Write Reportable	0	O
0x0021	*Language*	string (3bytes)	Read*Write Reportable	0	O
0x0022	*LEDSettings*	uint8	Read*Write Reportable	0	O

Table 7-10. *LockType* Attribute Values

Value	Definition
0x00	Dead bolt
0x01	Magnetic
0x02	Other
0x03	Mortise
0x04	Rim
0x05	Latch Bolt
0x06	Cylindrical Lock
0x07	Tubular Lock
0x08	Interconnected Lock
0x09	Dead Latch
0x0A	Door Furniture

Example Cluster: door lock commands (client side)

Command ID	Description	M/O
0x00	Lock Door	M
0x01	Unlock Door	M
0x02	Toggle	O
0x03	Unlock with Timeout	O
0x04	Get Log Record	O
0x05	Set PIN Code	O
0x06	Get PIN Code	O
0x07	Clear PIN Code	O
0x08	Clear All PIN Codes	O
0x09	Set User Status	O
0x0A	Get User Status	O
0x0B	Set Weekday Schedule	O
0x0C	Get Weekday Schedule	O
0x0D	Clear Weekday Schedule	O
0x0E	Set Year Day Schedule	O
0x0F	Get Year Day Schedule	O

Octets	Variable
Data Type	octstr
Field Name	PIN/RFID Code

- Server-side
  - Performs actions when it receives these commands
- Client-side
  - Capable of sending these commands

Example ZigBee profile: Smart Energy

- Interactions with energy providers for efficiency and cost savings

- Devices
  - Energy service interface
  - Metering device
  - Load control device

- Clusters
  - Demand response
  - Metering
  - Price
  - Key establishment (e.g. security)

Server Side	Client Side
Mandatory	
	Demand Response and Load Control
	Time
Optional	
	Price
	Calendar
	Device Management
	MDU Pairing
Energy Management	
Alarms	
Tunneling	Tunneling

Example: demand response cluster

- No attributes, only commands

Command Identifier	Description	M/O
0x00	Load Control Event	M
0x01	Cancel Load Control Event	M
0x02	Cancel All Load Control Events	M

Load Control Command Payload

Octets	4	2	1	4	2	1	1
Data Type	uint32	map16	uint8	UTC	uint16	uint8	uint8
Field Name	Issuer Event ID (M)	Device Class (M)	Utility Enrollment Group (M)	Start Time (M)	Duration in Minutes (M)	Criticality Level (M)	Cooling Temperature Offset (O)

Octets	1	2	2	1	1	1
Data Type	uint8	int16	int16	int8	uint8	map8
Field Name	Heating Temperature Offset (O)	Cooling Temperature Set Point (O)	Heating Temperature Set Point (O)	Average Load Adjustment Percentage (O)	Duty Cycle (O)	Event Control (M)

Endpoints

- Each ZigBee device has a number of Endpoints (up to 240)
  - Number by which remote applications can contact it
  - Analogous to a Port in TCP/UDP

- Each Endpoint has one Device Type attached to it
  - Communication refers to the Endpoint number,
    - Then the Cluster ID within it,
    - Then the Attribute/Command ID within that
  - Endpoints can be queried to determine what they provide

- Special case: Endpoint 0 – ZigBee Device Object
  - All devices must implement the ZigBee Device Object
  - Attributes and Commands for controlling a network device
  - Network parameters are configured just like a light or door lock

Example Endpoints for a device

An example endpoint implementation:

Endpoint # - Profile Name: Device Type

0 - ZigBee Device Profile (ZDP): ZDO

1 - HA: Thermostat

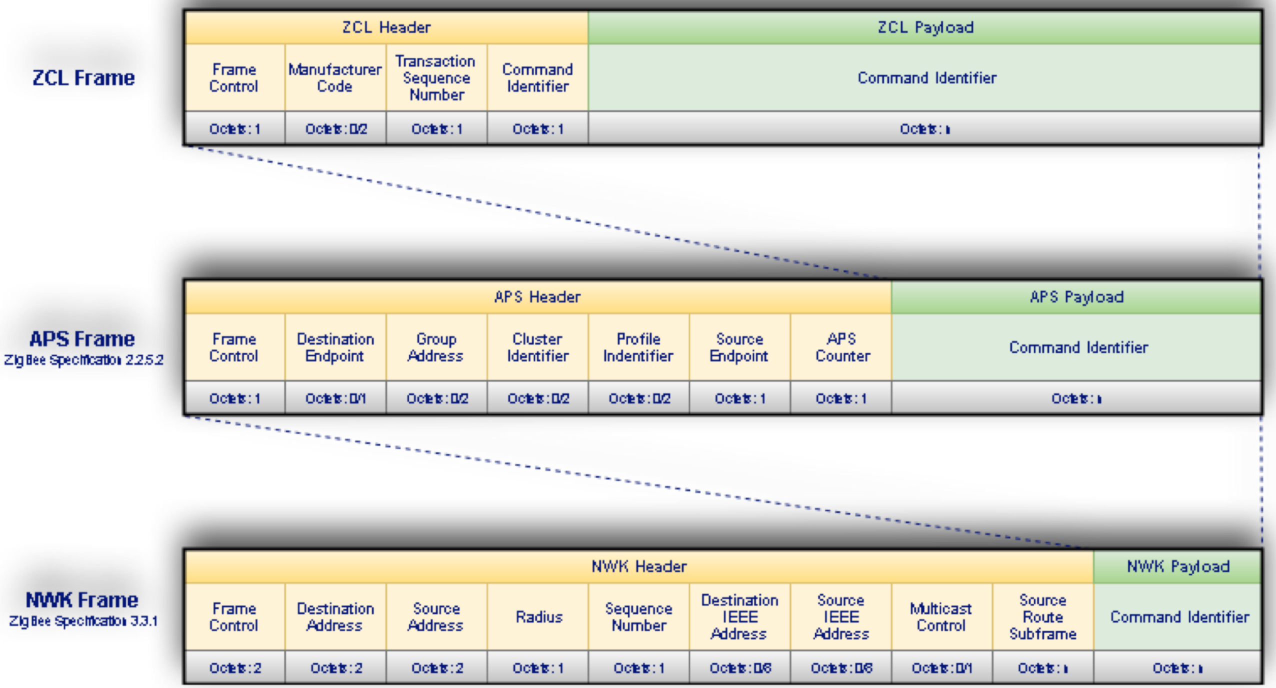
2 - HA: On/Off Output

3 - SE: In-Home Display

4 - MSP: Proprietary vendor extensions

- Even simple devices hopefully have three endpoints:
  1. ZigBee Device Object
  2. <Their functionality>
  3. Over The Air Bootloader (code updates)

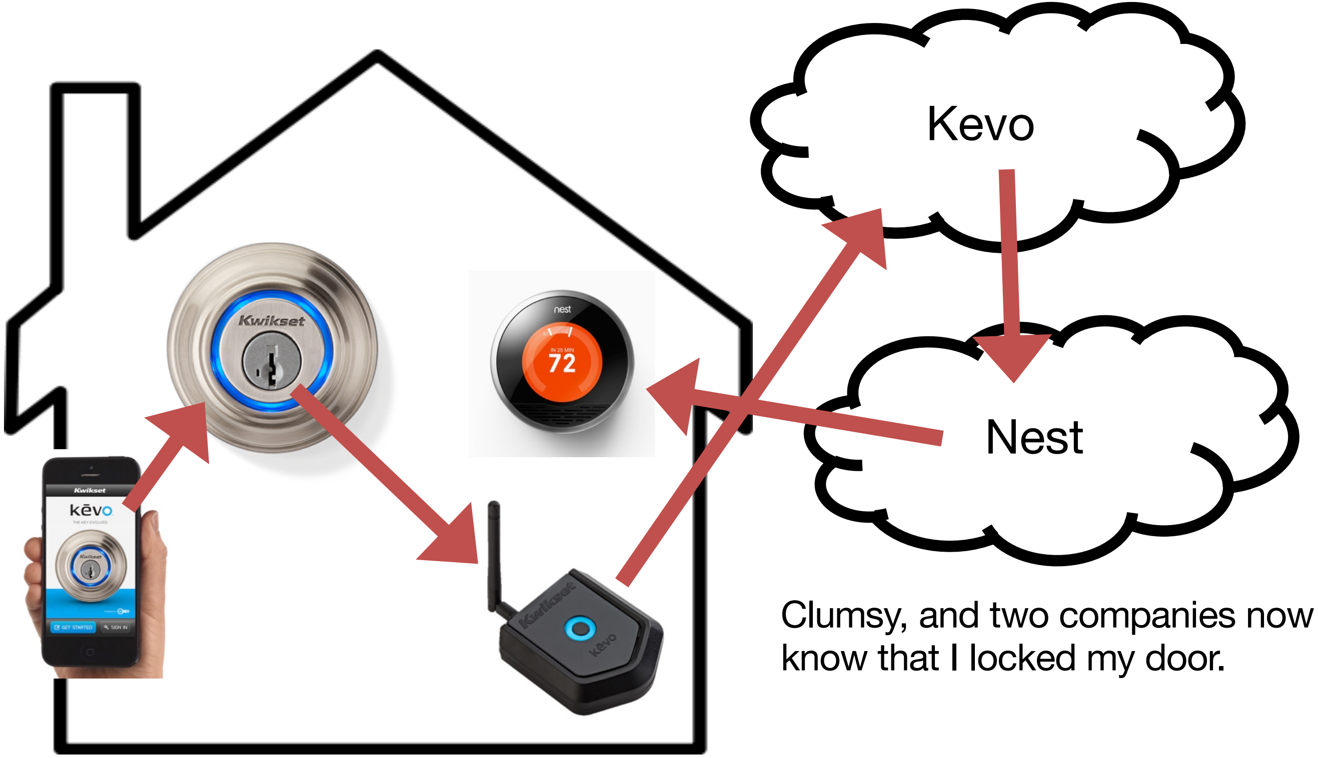
ZigBee application layer packets



Outline

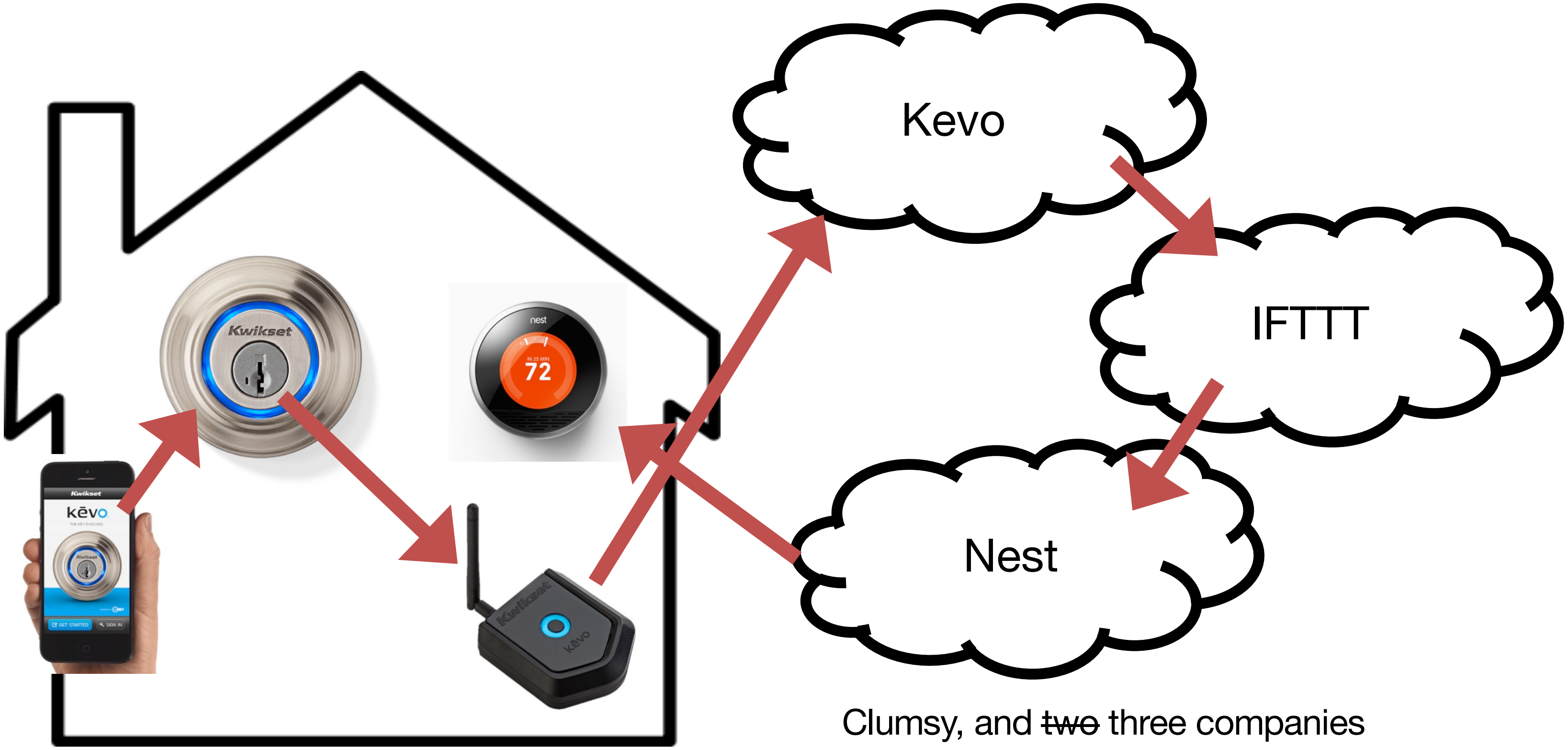
- Zigbee
- **The Interoperability Problem**
- Routing
- Flooding

“When I leave, turn down the AC”



Clumsy, and two companies now know that I locked my door.

“When I leave, turn down the AC”



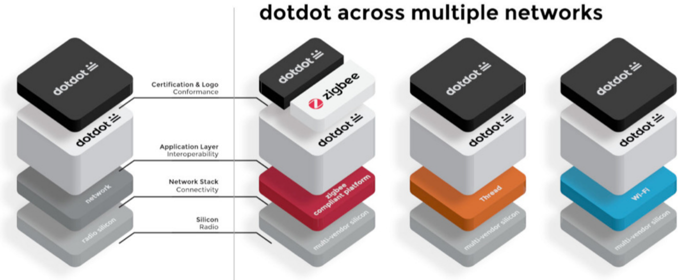
Clumsy, and ~~two~~ three companies now know that I locked my door.

What does it looks like when it doesn't go through three different clouds?

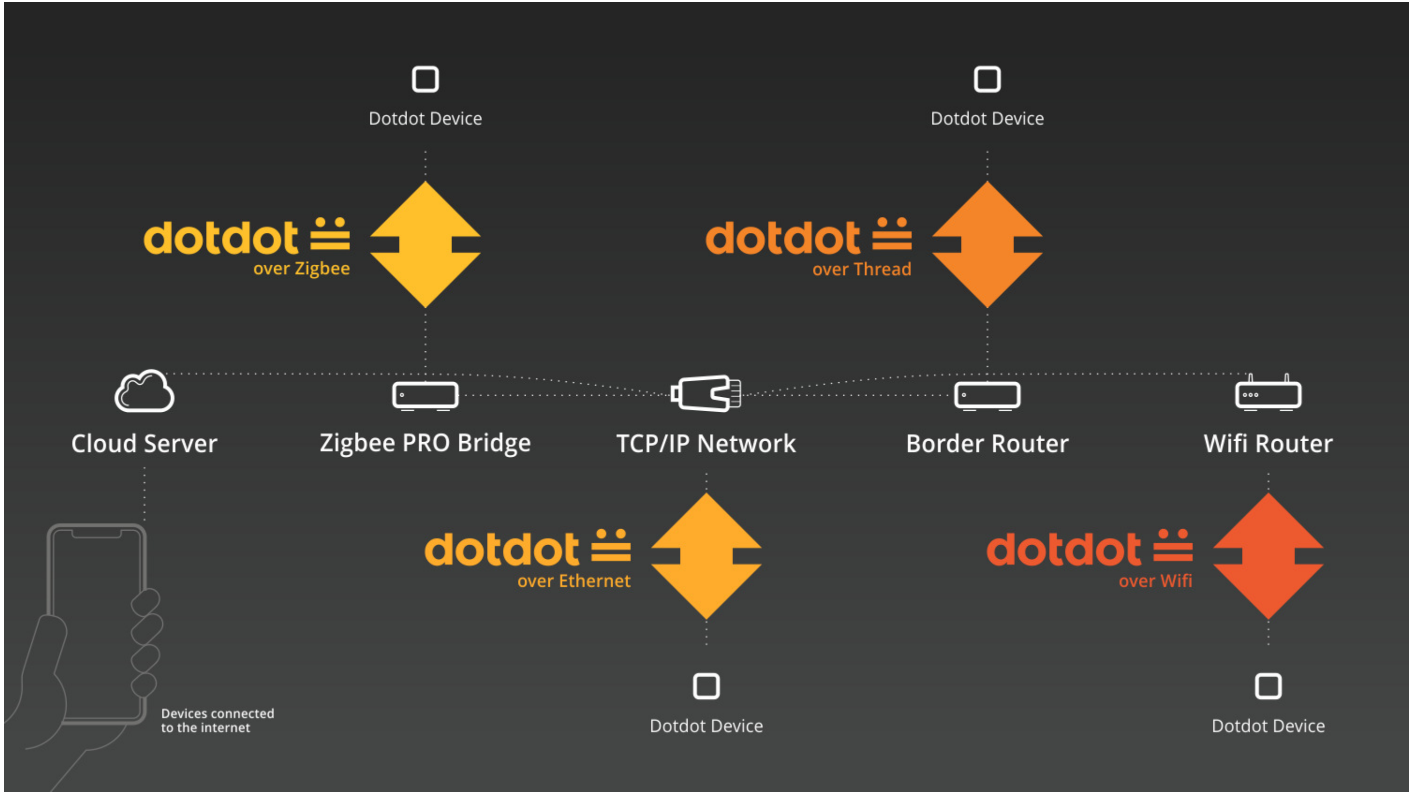
- "Standardization" is the answer? Custom adaptations?

Interoperability, the lack thereof, and ~~Zigbee's~~ CSA's proposed solution

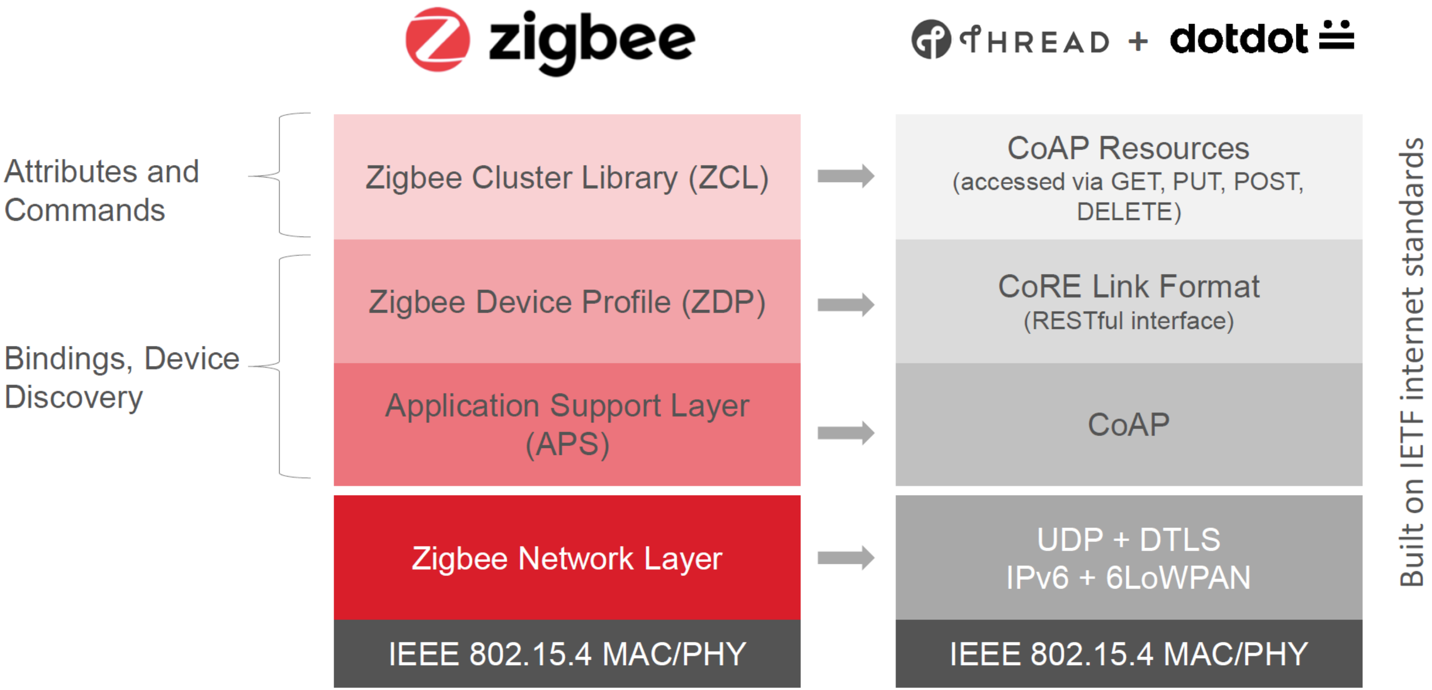
- Zigbee's device provisioning set themselves up for this problem, but also meant they were early to ideas of how to fix it
- The specification for how to interact with devices is far above anything network-specific
- dotdot is a recent effort to spread ZigBee Clusters more widely
  - Runs same application-layer on top of various lower layers
  - ZigBee, BLE, Thread, WiFi, Ethernet



dotdot provides ZigBee-style control over various networks



Example dotdot over Thread

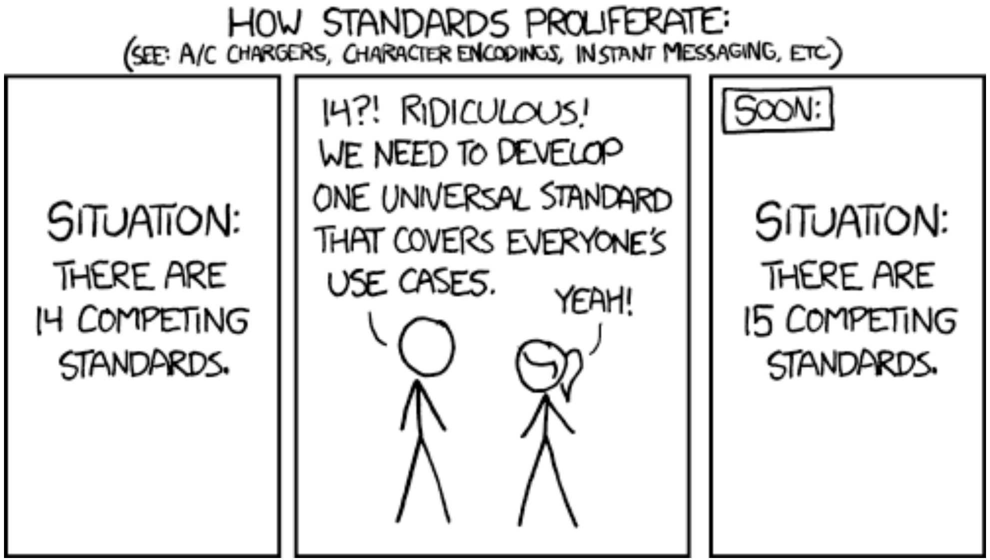


ZCL to CoAP mappings

Resource	Methods	URI
Resource discovery	GET	/zcl
Endpoints	GET	/e
Attributes	GET, PUT, POST	/a
Commands	GET, POST	/c
Bindings	GET, PUT, POST, DELETE	/b
Report Configuration	GET, PUT, POST, DELETE	/r
Report Notification	POST	/n
Group Notification	POST	/g
EZ-Mode Commissioning	GET, POST	/m

Is ZCL the right standard for device interactions?

- Seems better than making something new from scratch



https://xkcd.com/927/

Outline

- Zigbee
- The Interoperability Problem
- **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

Simple routing solutions

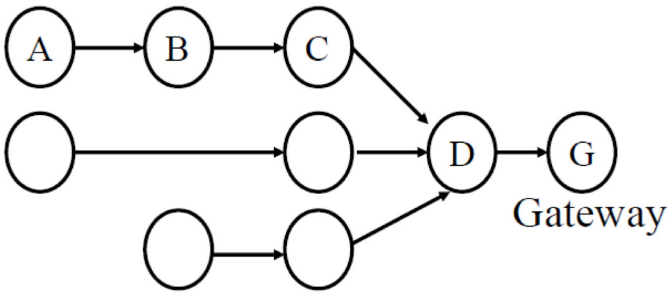
- Broadcast
  - The link-layer solution for everything
- Star topology
  - Only one location to send to: parent
  - Single parent needs to store information about all children
    - Addresses, schedules, etc.
- Tree topology
  - “Star of stars”
  - Two choices: send to descendent or send to parent
  - Each parent needs to store information about all children beneath it
  - Original ZigBee approach (knowledge built into addressing scheme)

Many-to-one routing

e.g. Collection Tree Protocol (CTP)

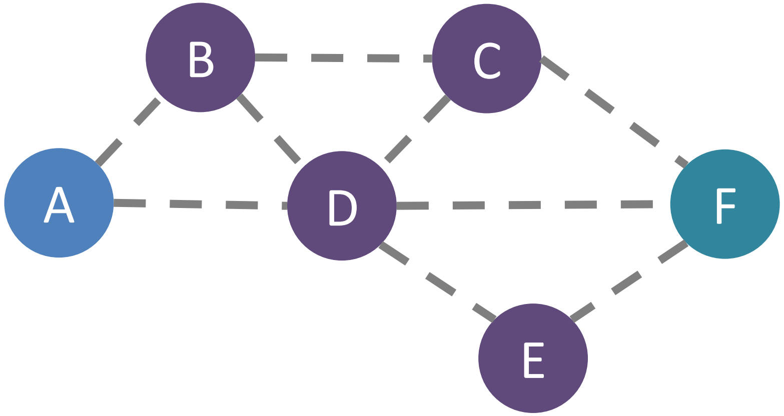
- Tree optimization for sensor networks
  - Keep all devices except the "gateway" as simple as possible

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



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)

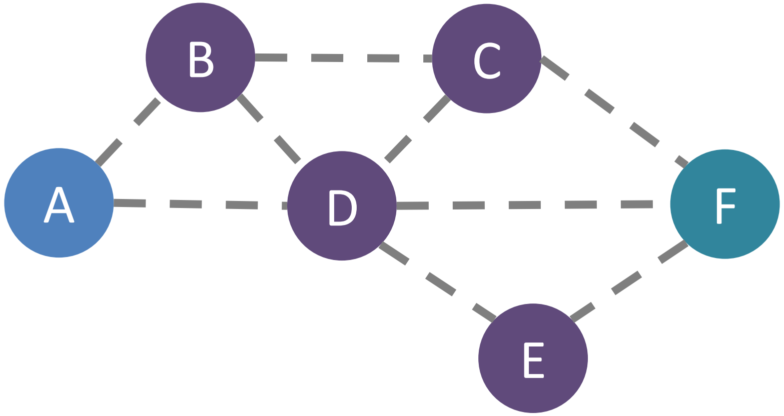
Request ID	Source Address	Destination Address	Source SeqNo	Destination SeqNo	Hop Count

- 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

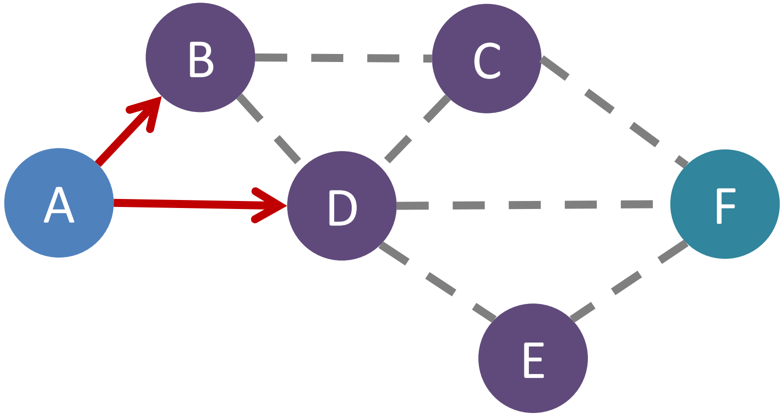
A wants to find a route to F, so it sends out an RREQ



Example AODV RREQ (A to F)

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

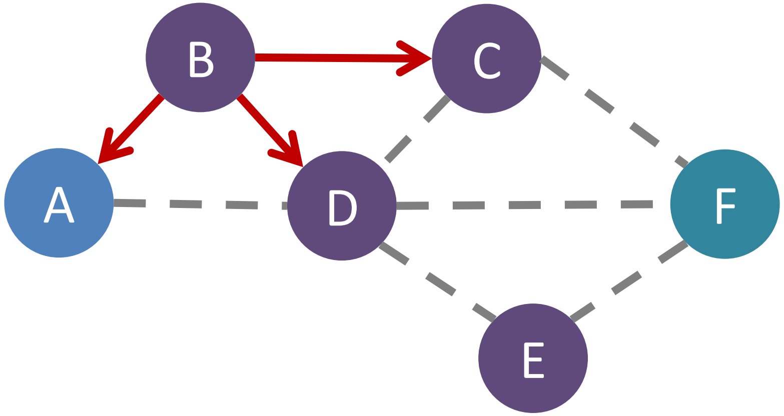
B and D also opportunistically add a routing table entry for A



Example AODV RREQ (A to F)

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

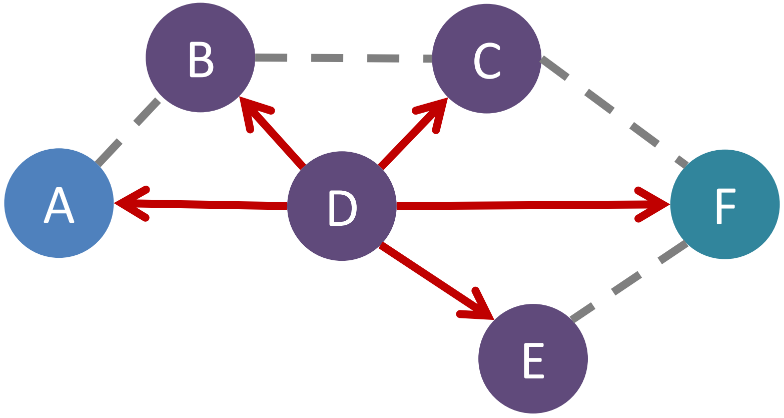
B goes first via some access control protocol (D also in contention)
A and D ignore duplicate Request ID
C opportunistically adds a routing table entry to A



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

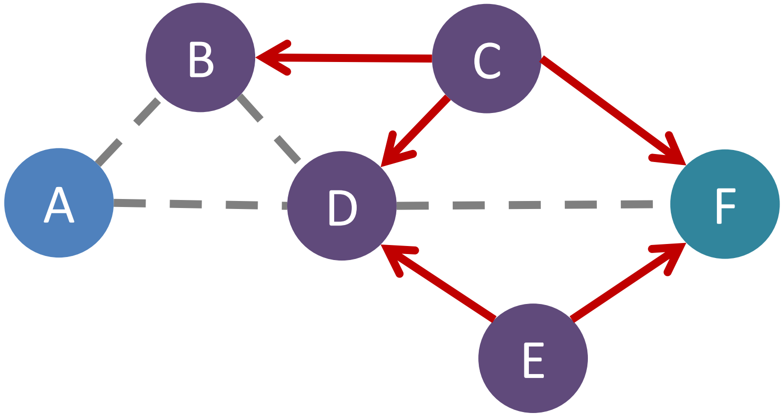


Example AODV RREQ (A to F)

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)

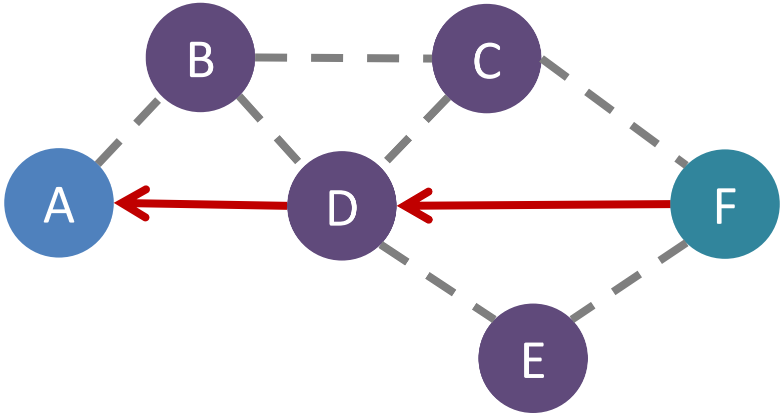
Source Address	Destination Address	Destination SeqNo	Hop Count

- 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)

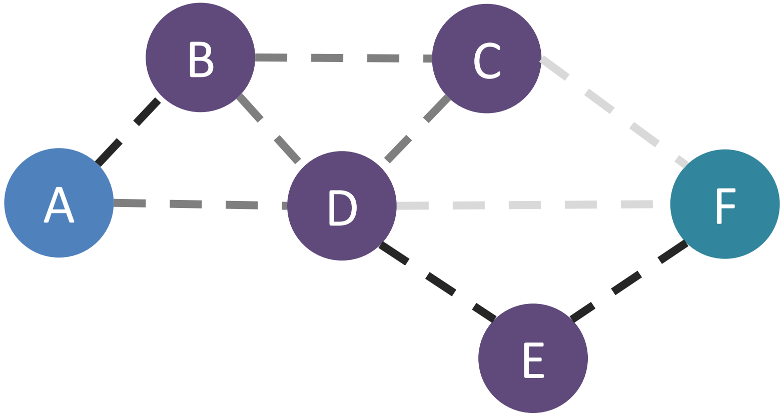
  - Need to be careful to avoid loops!

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

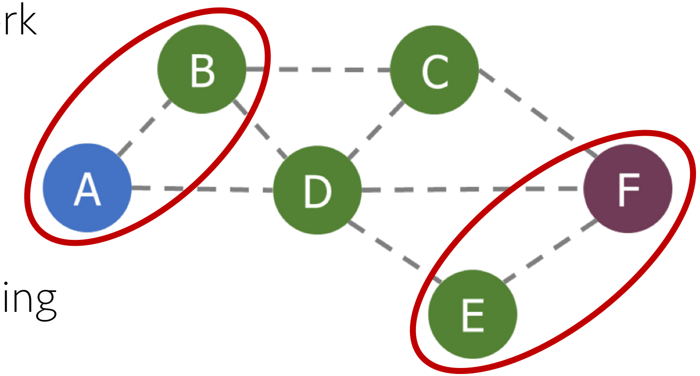
Reliability as a cost metric

- Link quality can 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

- Zigbee
- The Interoperability Problem
- Routing
- **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

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 few lectures back I mentioned ‘broadcast ACKs can work’, here’s how:

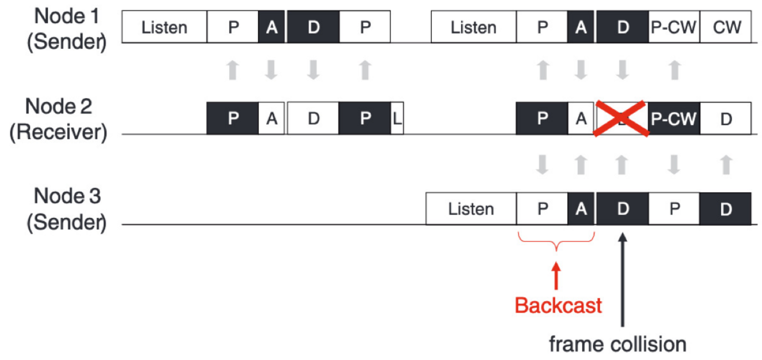


Fig. 2. A contention-free transfer (left) and a collision (right). Although the auto-ack frames collide, they do so nondestructively, so the receiver correctly decodes their superposition as a valid frame. Hence, the receiver concludes that traffic is pending, so it retransmits a probe with an explicit contention window, which Node 3 wins.

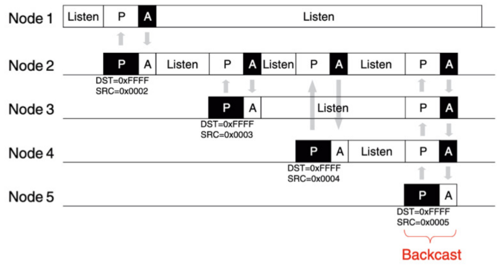


Fig. 4. Asynchronous network wakeup with A-MAC. Although Nodes 2, 3, and 4 all ACK Node 5’s query probe, the ACK collision is nondestructive, and Node 5 remains awake to communicate.

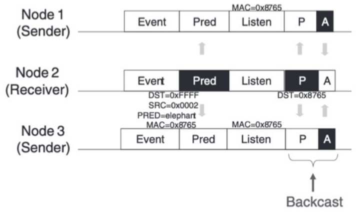
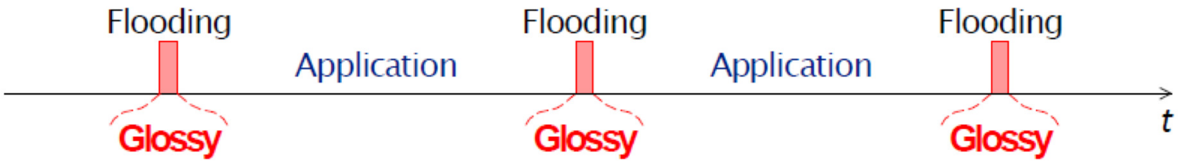


Fig. 5. Pollcast implemented using the A-MAC architecture. All nodes observe an “elephant sighting” event. Node 2 wishes to corroborate this observation with its neighbors. It uses backcast to efficiently determine if any neighbor also observed this event.

Glossy key techniques

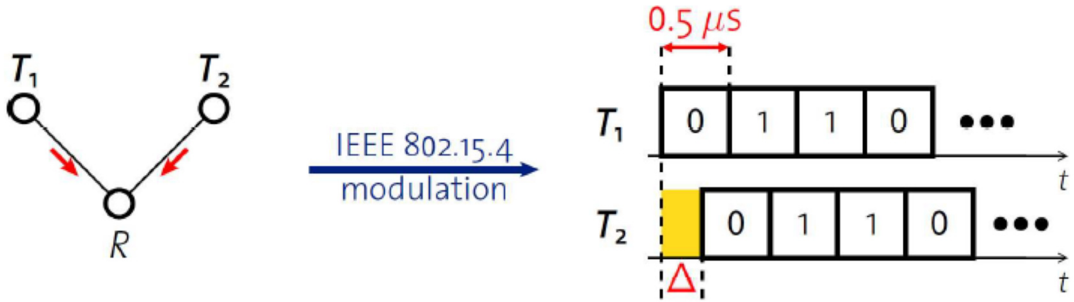
- Temporally decouple network flooding from application tasks



- Exploit synchronous transmissions for fast network flooding

Synchronous transmissions

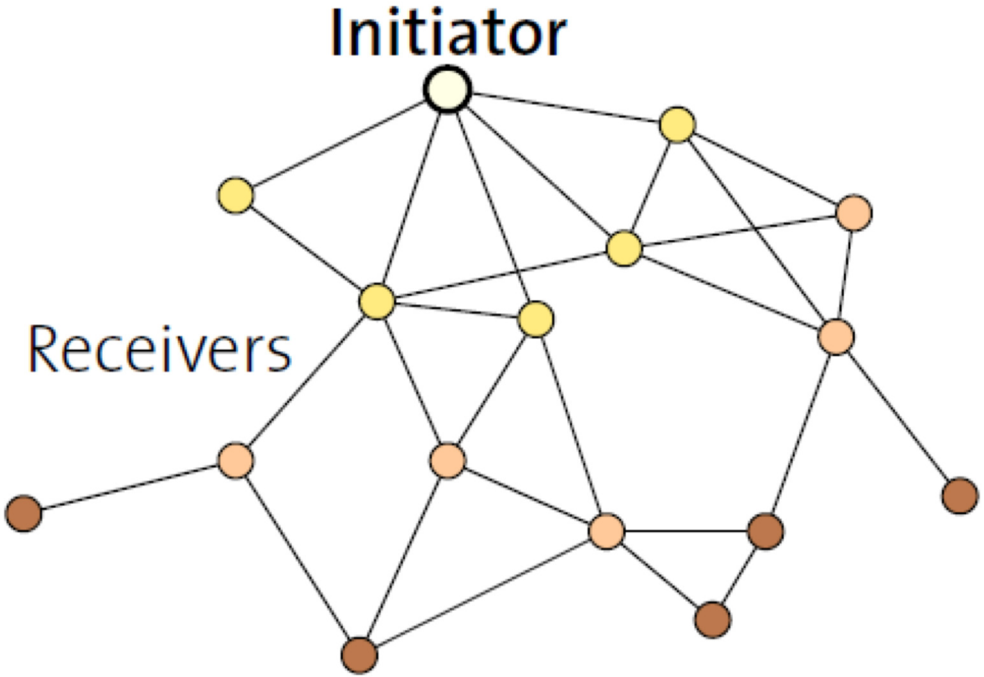
- Multiple nodes transmit **same packet** at **same time**



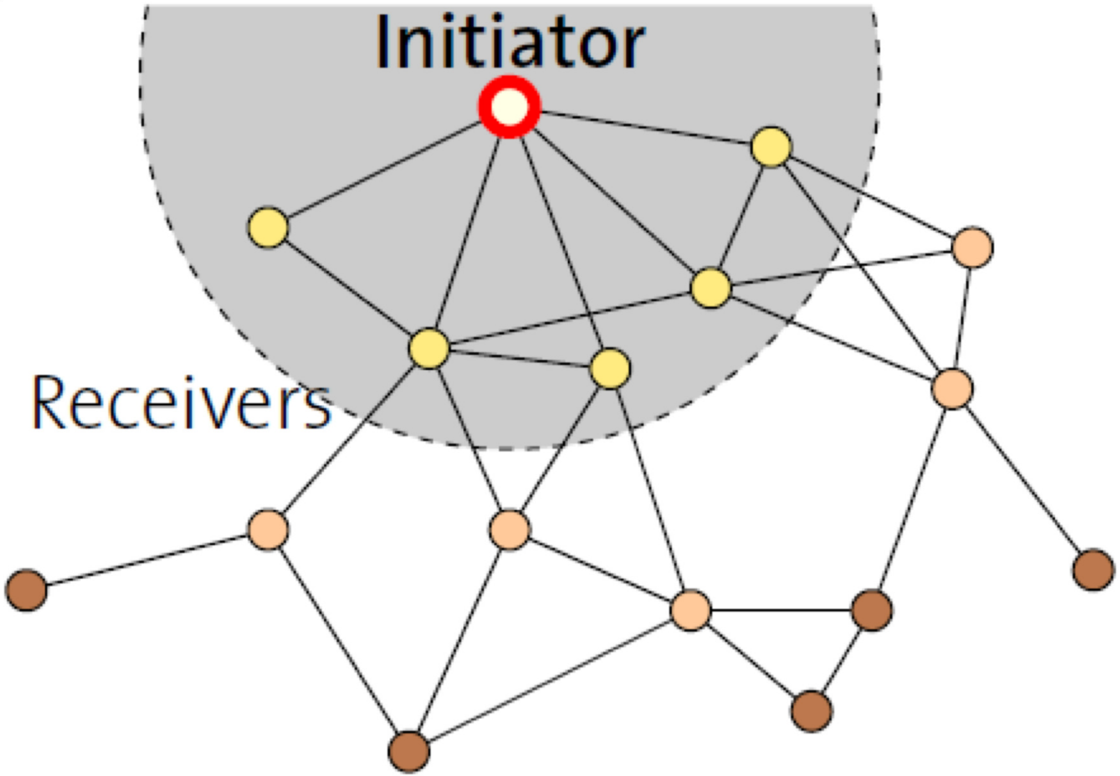
- R can receive packet with high probability if Δ is small
  - May even improve probability of reception (more energy at receiver)

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

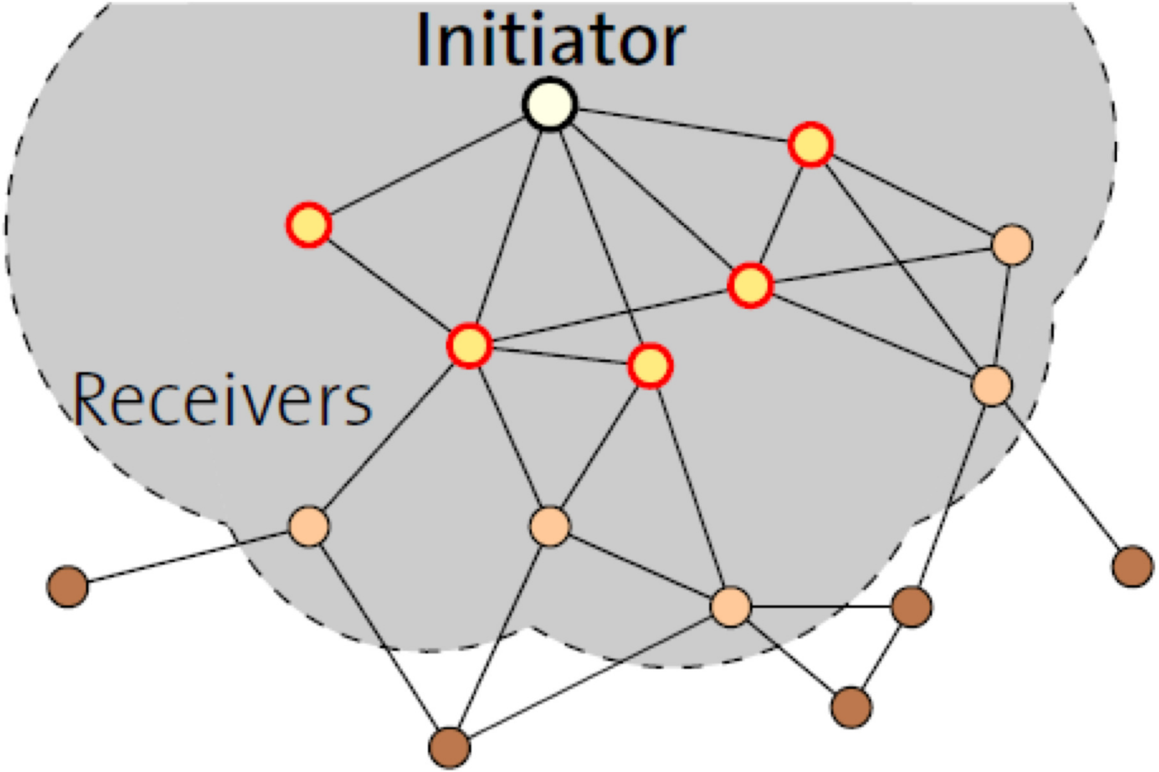
Fast packet propagation in Glossy



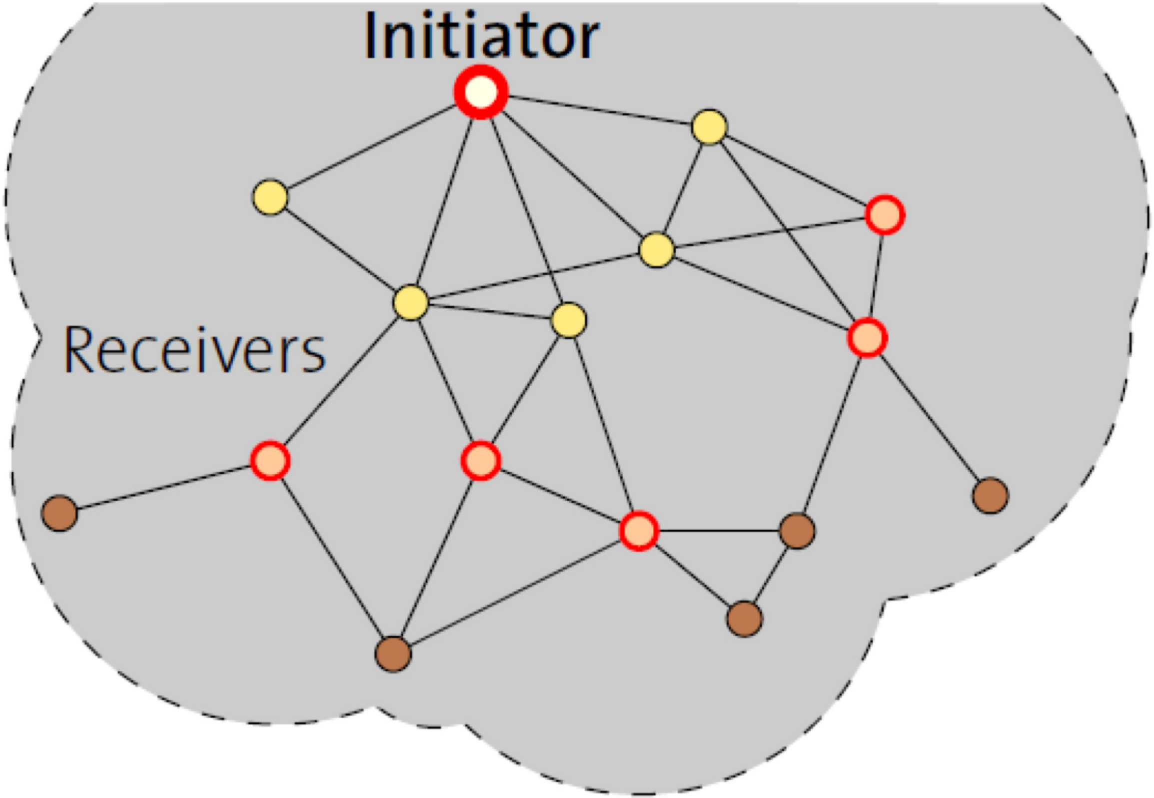
Fast packet propagation in Glossy



Fast packet propagation in Glossy



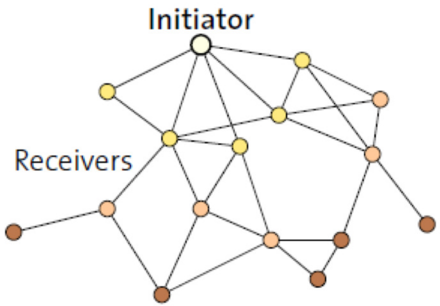
Fast packet propagation in Glossy



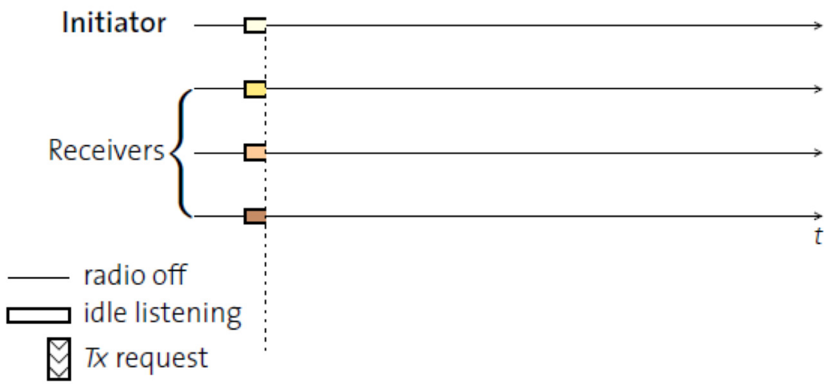
Glossy details

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

Example

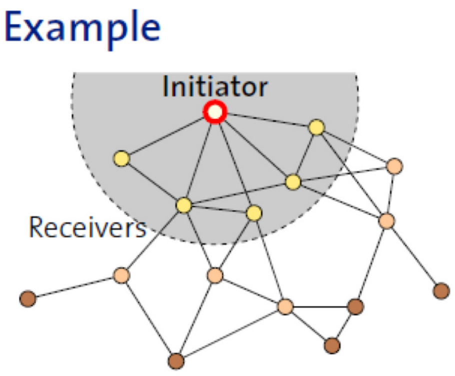


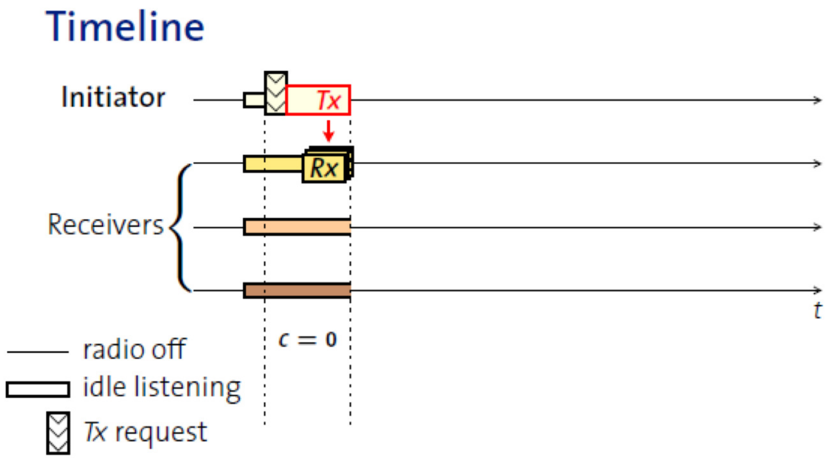
Timeline



Glossy details

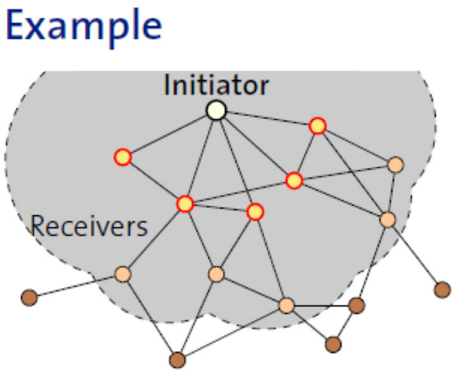
- Initiator
  - Set relay counter in packet, C = 0
  - Broadcast packet

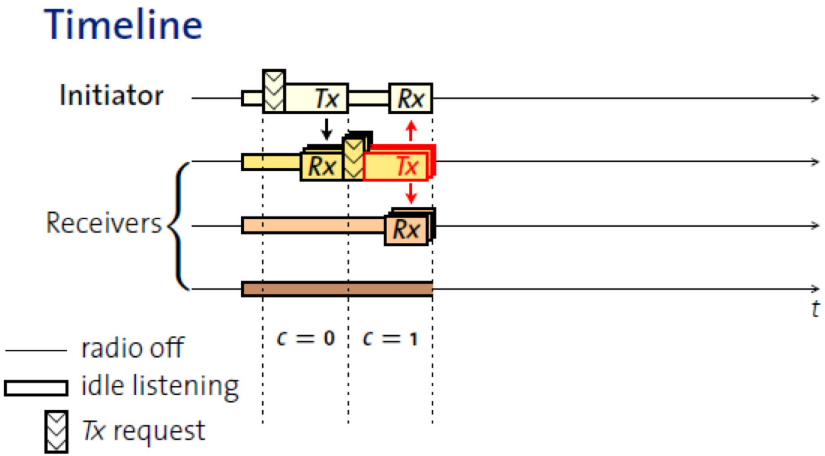




Glossy details

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

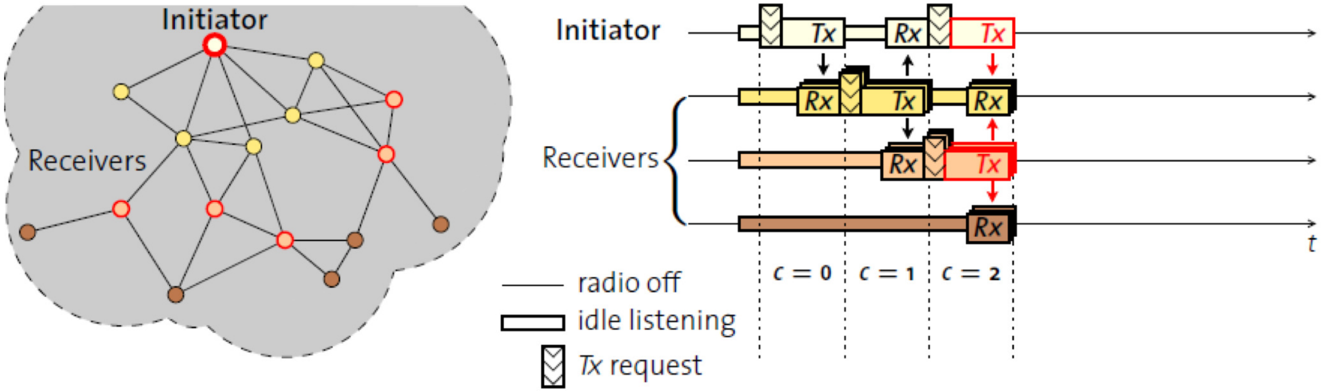




Glossy details

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

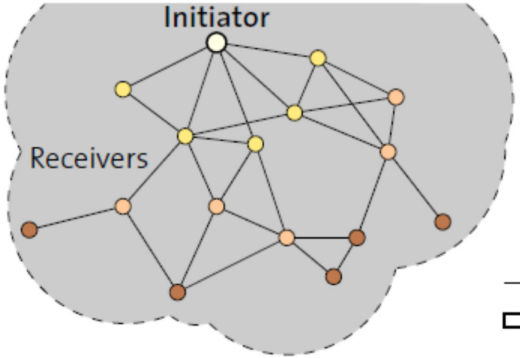
Example **Timeline**



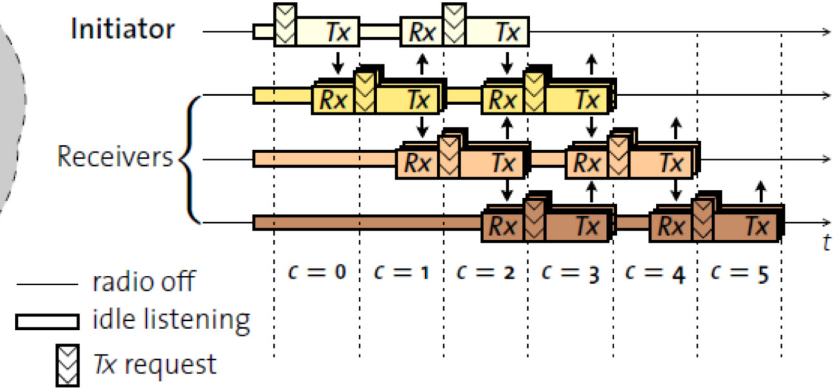
Glossy details

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

Example



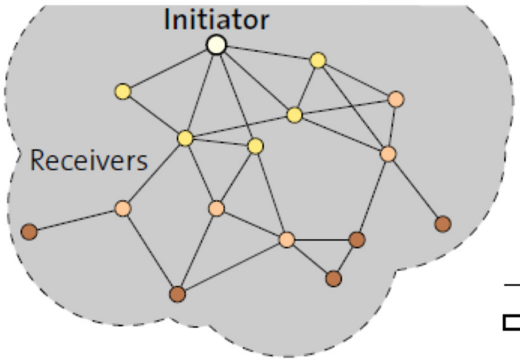
Timeline ($N = 2$)



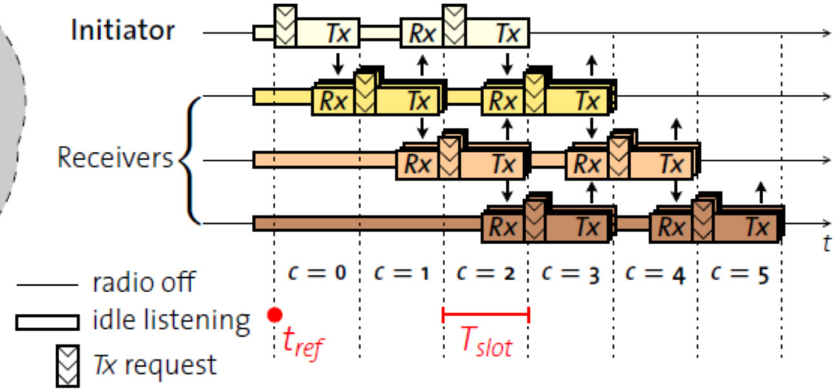
Glossy details

- 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



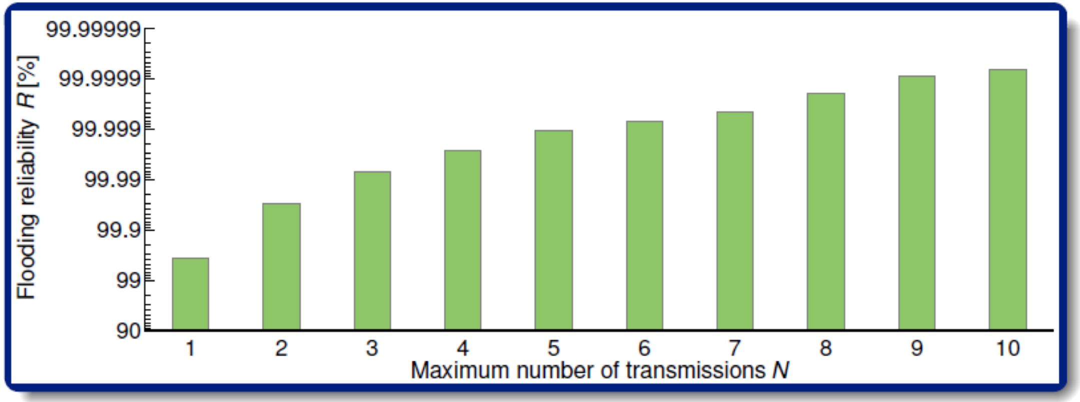
Timeline $(N = 2)$



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



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

Next time: WiFi