

Distributed Systems

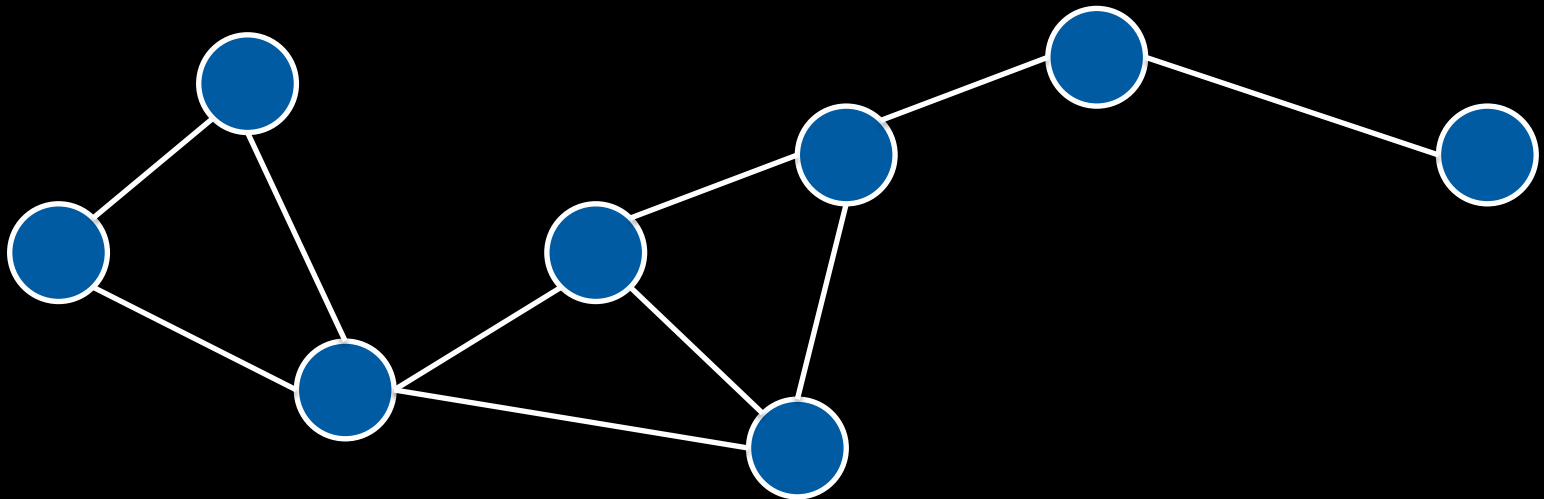
26. Mobile Ad Hoc Mesh Networks

Paul Krzyzanowski

pxk@cs.rutgers.edu

Mesh Networks

- Mobile Ad-hoc networks, sensor networks, ...
- Decentralized networking
- No need for routers or access points
- Each node participates in routing

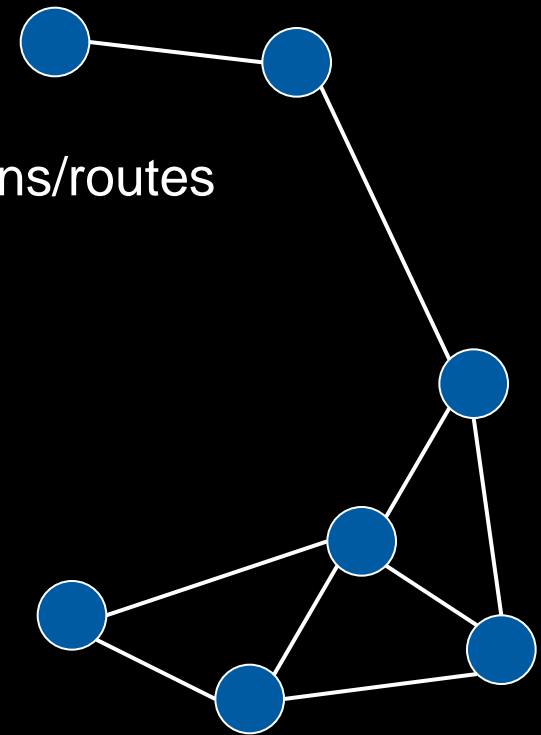


Mesh Networks

- Topology of the network has to be discovered
 - Nodes come and go or move around
 - Topology may change frequently → on-demand routing
 - New nodes announce themselves
 - Other nodes listen for announcements
- Often low-range, battery-powered devices
 - Multiple hops are common

Mesh Networking

- Hop node-to-node until the destination is reached
 - Nodes can act as repeaters to nearby peers
 - Robust connectivity: find alternate routes
- Dynamic routing
 - Table-based: maintain fresh lists of destinations/routes
 - Dynamic: find route on demand
 - Hierarchical
 - Geographical
 - Power-aware
 - Multicast



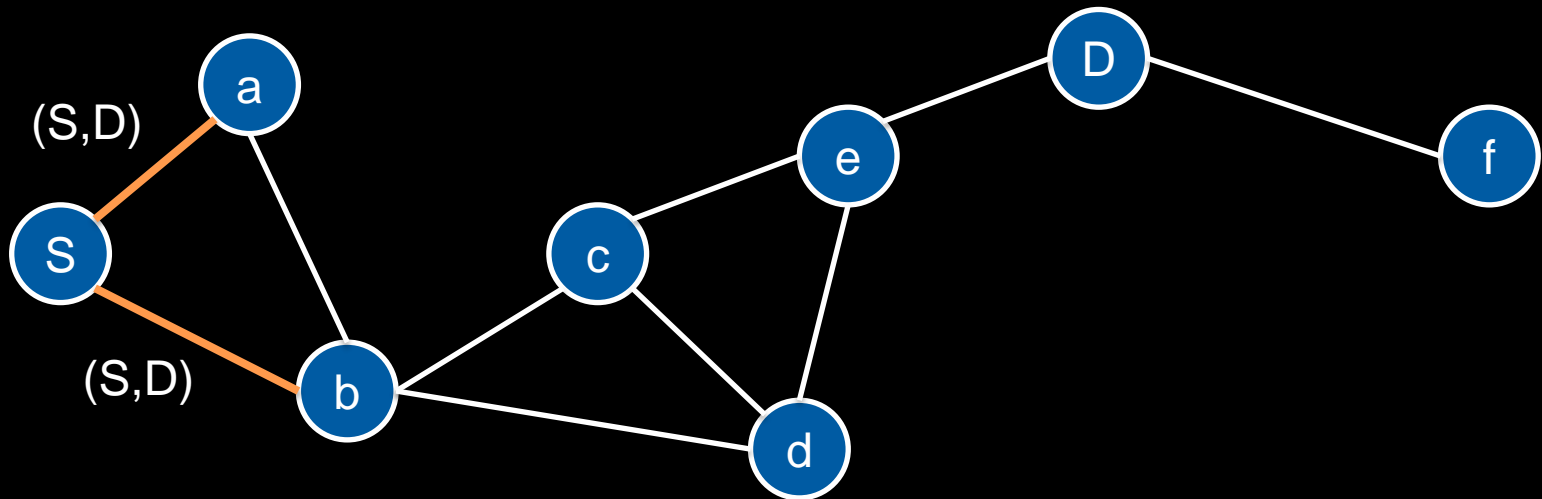
Dynamic Source Routing (DSR)

- On-demand routing
- **Source routing**: sender learns the complete set of hops needed to reach the destination
 - Every packet carries the path of hops in its header
- Intermediate nodes do not need to store routing information to route packets

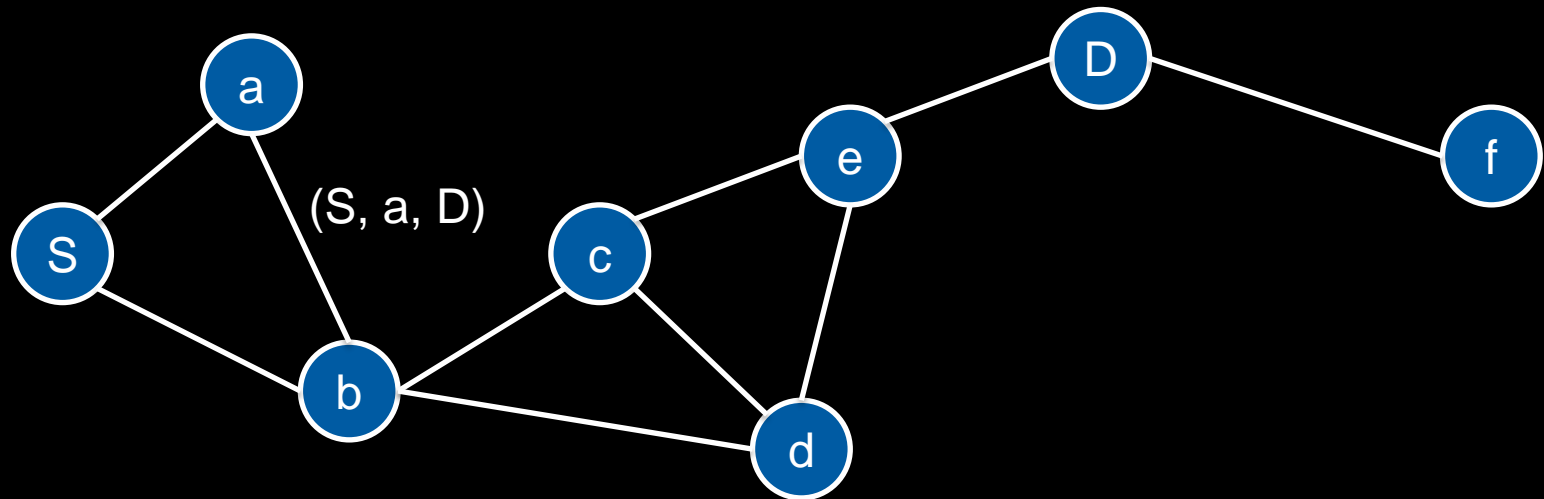
Route discovery via flooding

- Broadcast *Route Request* message for some destination node, D , to all connected (reachable) nodes
- Receiving node:
 - If it saw a packet with this ID, ignore request
 - If node \equiv node D : send back a *Route Reply* message to the path in the header
 - Else: add node ID to the path and forward *Route Request* to connected nodes
- Note:
 - Path is maintained throughout the *Route Request*
 - Loops are prevented (don't route to a node in the path)

Route discovery via flooding

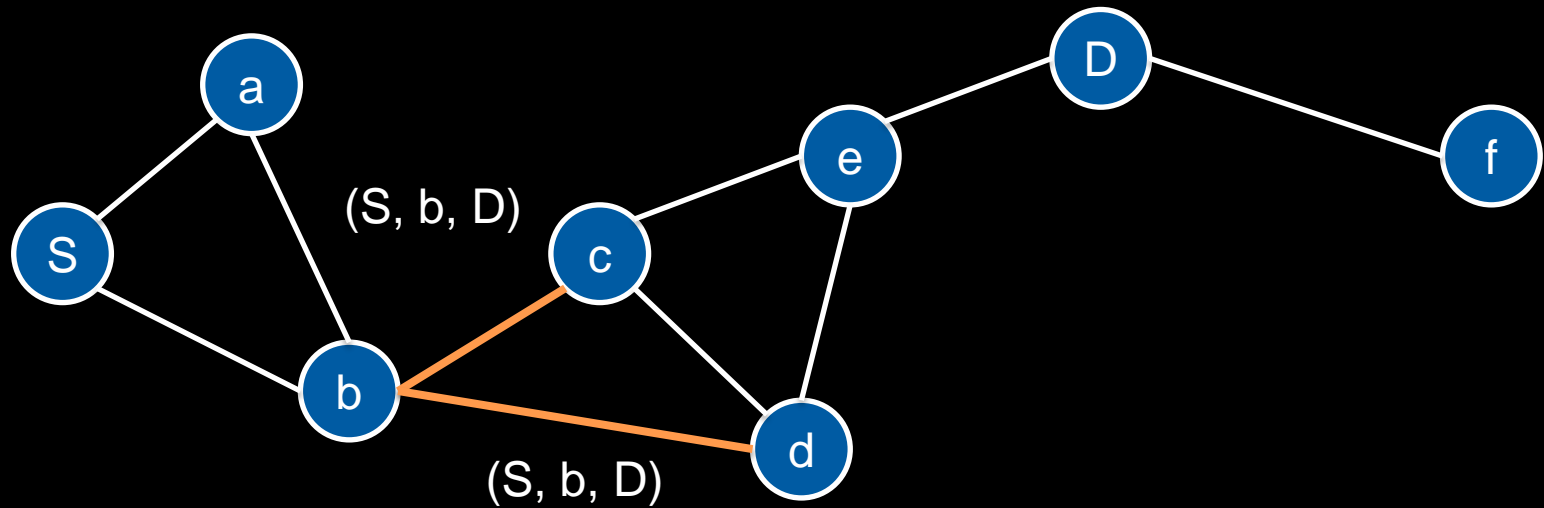


Route discovery via flooding

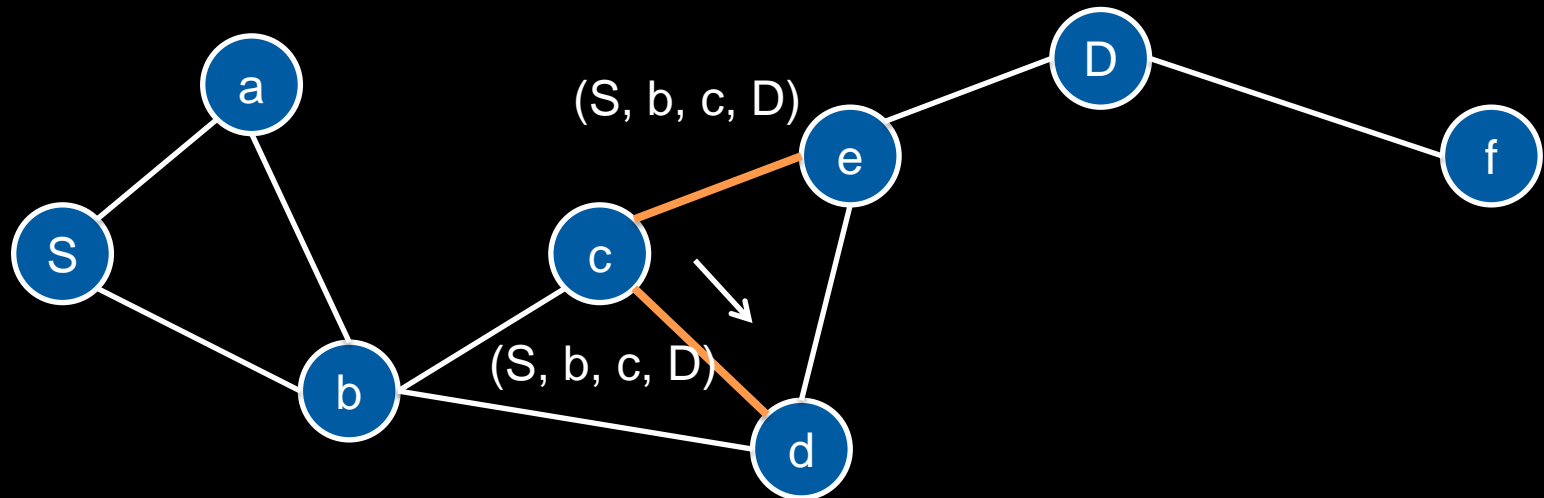


b will drop the request to route S to D if it saw the same Route Request ID

Route discovery via flooding

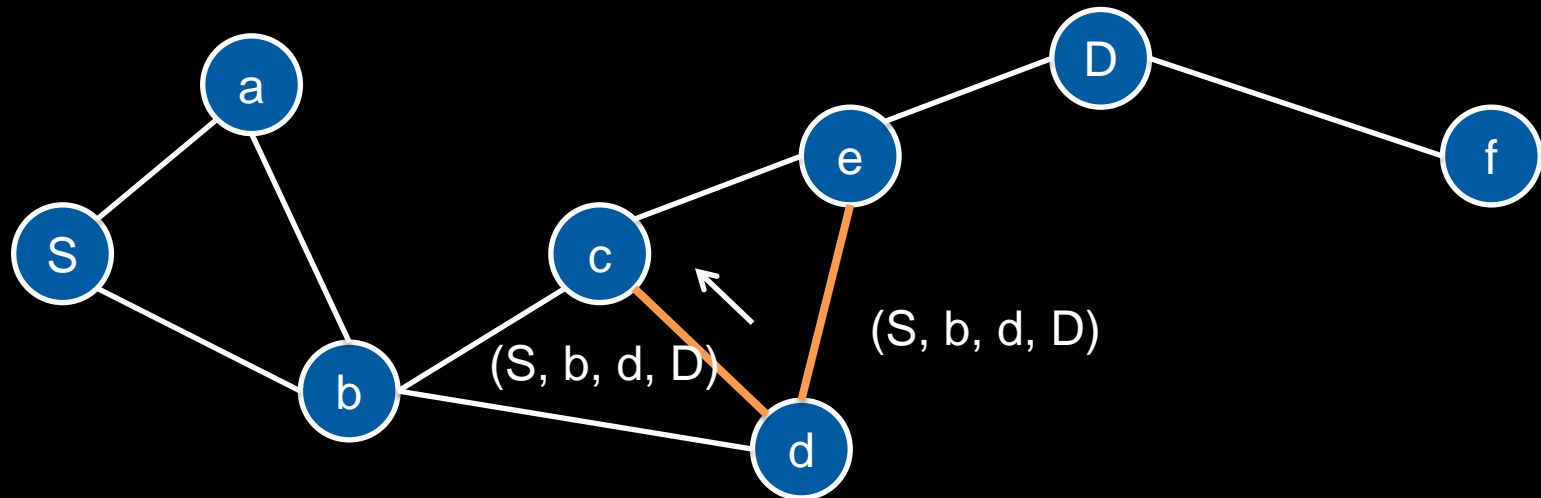


Route discovery via flooding



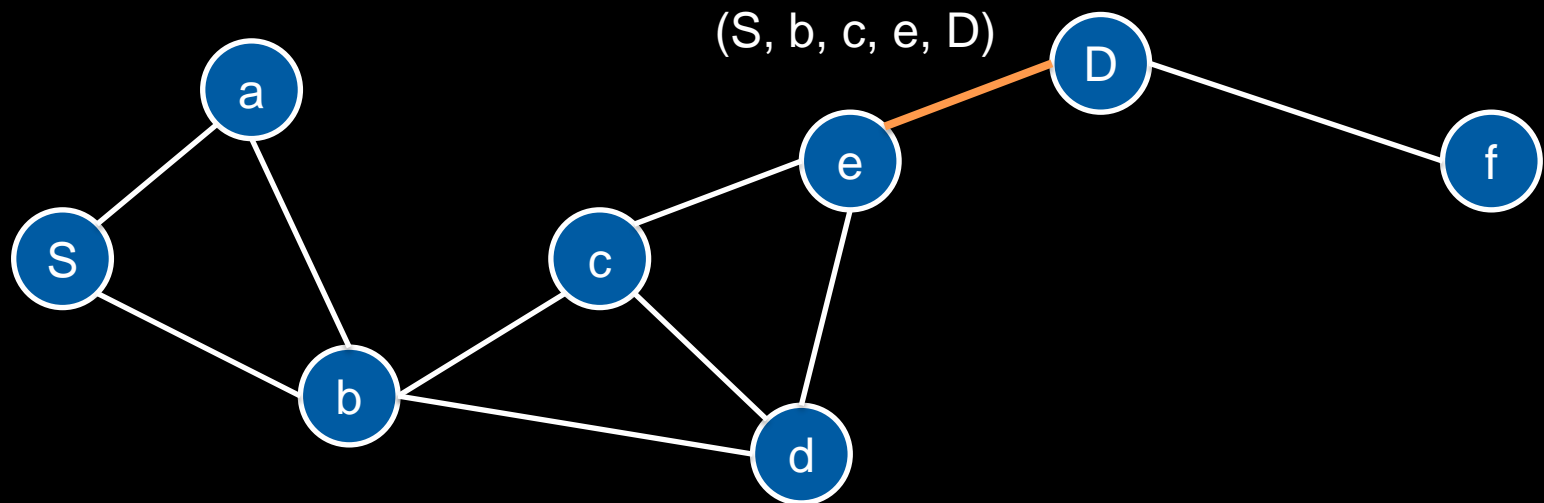
d will drop the request to route S to D since it saw the same Route Request ID from b earlier

Route discovery via flooding



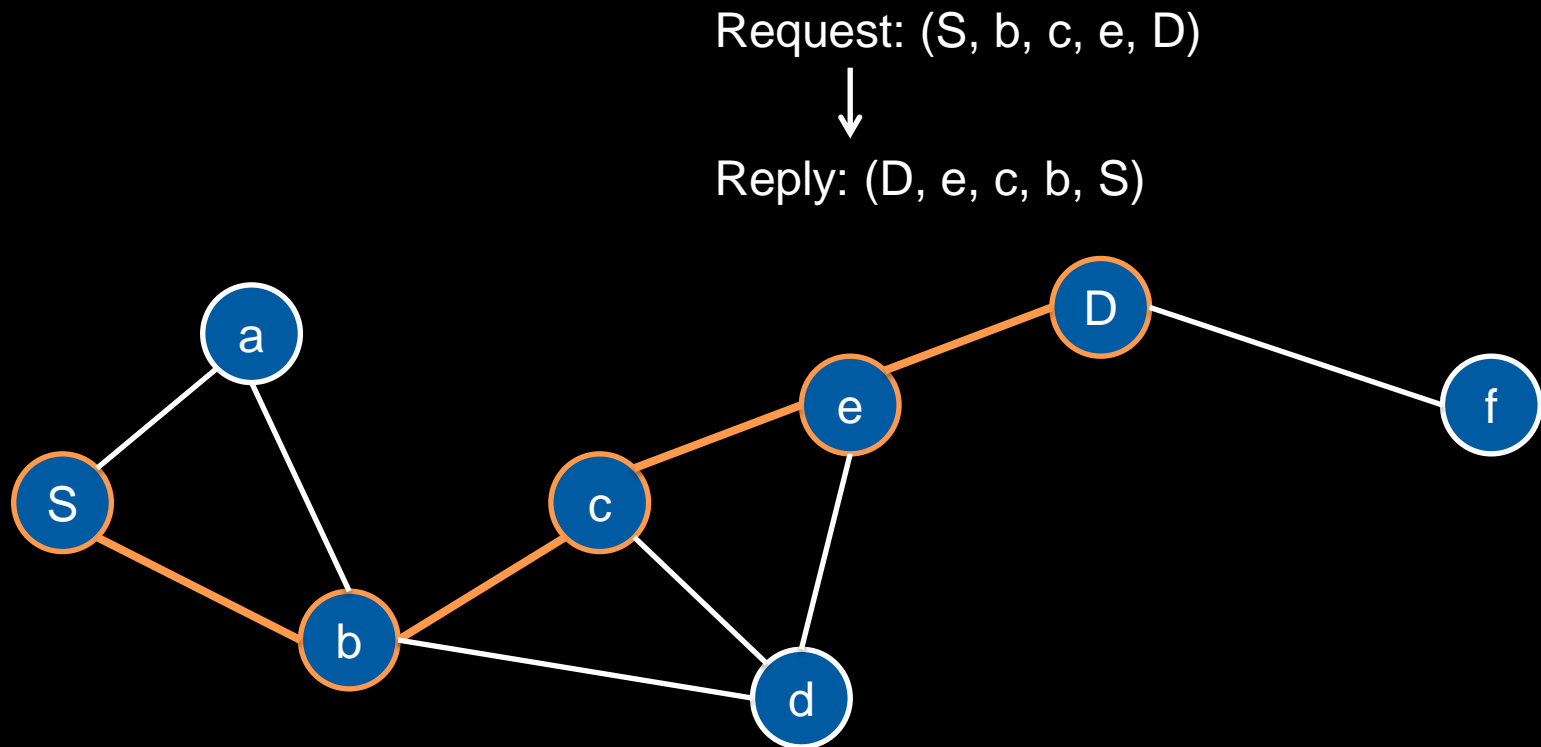
At around the same time, c will drop the request to route S to D since it saw the same Route Request ID from b earlier

Route discovery via flooding



e routes to d. D is the destination!

Route discovery via flooding



D sends a Route Reply back to S via the path.

Source Routing

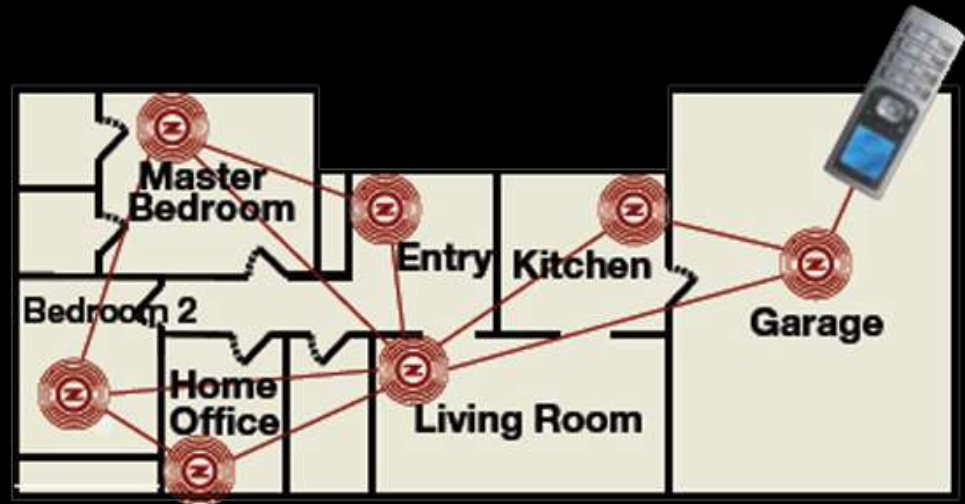
- Once a route is known
 - Path is specified with the packet header
 - If delivery fails, path will need to be rediscovered
- Optimizations
 - Every node that sees packets or *Route Reply* messages learns how to reach all the nodes listed in the route.
 - Nodes can maintain tables to minimize learning routes
 - More-frequently used routes may be considered to be more accurate
 - Maintain and optimize for Distance Vector
 - Distance Vector = number of hops
 - Replace known routes with new routes with smaller distance vector
 - Or test new routes with a potentially smaller distance vector

Table Driven Routing

- When a node learns of a newly accessible node, it adds it to its routing table
 - Propagates updated routing table to other nodes (those nodes forward it, etc.)
 - Everyone's routing table is kept updated
- No need to perform dynamic routing
- No need to send source route in each packet
- Downside: extra network traffic for table propagation
 - May be worthwhile if the environment is not dynamic

Example: ZigBee

- ZigBee (IEEE 802.15.4)
 - 192 kbps
 - 100-1000 ft. range
- ZenSys Z-Wave



Screw-in lamp
module
\$38

Dimmer switch
\$38

Outdoor plug
module
\$40

ZigBee

- Wireless star or mesh network
 - Star: single coordinator; no routing
- Self-configuring
- Redundant paths
- Self-healing

ZigBee device types

- **Coordinator**
 - Starts & controls network
 - Stores info about the network
 - Repository for security keys
- **Router**
 - Extends network coverage, provides backup routes
 - Connects to coordinator, other routers, & end devices
- **End devices**
 - Transmit or receive a message
 - Does not perform routing – must connect to a coordinator or router
- **Roles may be combined:**
 - Light socket: router or coordinator & end device
 - Battery-powered light switch: end device (mostly off)

Joining & Routing

- **Joining**

- Device sends a request to [re] join a network to the coordinator or router
- In the simplest case, it has the network key

- **Routing**

- Based on **Distance Vector** routing
 - Distance Vector = number of hops to the destination
- Each router maintains a routing table with entries for each destination
- **Routing table** entry stores a distance vector & next router

- **Learning routes**

- Originating device broadcasts a *route request*
- Destination device sends a *route reply*
- Intermediate nodes build up routing tables

ZigBee Security

- **Encryption**
 - All network traffic is encrypted with 128-bit AES
- **Trust Center**
 - Maintains network key; periodically sends key updates (new key encrypted with old key)
 - Runs on a designated trusted device
 - Decides whether to allow new devices onto its network
- **Keys**
 - **Master keys:** initial shared secret between two devices. Used to generate Link Keys
 - **Network keys:** all devices on a network share the same key
 - **Link keys:** optional for application-level encryption between two devices

ZigBee setup

- Out-of-the-box device can join any network
- Installer:
 - Uses dedicated device with ZigBee coordinator
 - Device joins a “**commissioning network**”
 - Installer commissions device to identify correct network & security settings
 - Master key & Trust Center address configured
- If master key is not configured
 - May sent on an insecure channel during configuration
 - Some implementations (Certicom’s) support public-key based key exchange using ECC (Elliptic Curve Cryptography)
 - Less compute (power) overhead than RSA or Diffie-Hellman

Network-Based Plug & Play

Ad-hoc networking and auto-discovery

- Device/service discovery and control
 - Microsoft, Intel: UPnP
 - Apple: Bonjour
- Universal Plug and Play architecture
 - <http://www.upnp.org>
- Networked devices
 - Unreliable: nodes added/removed unpredictably
 - Programs need to talk to programs (services)

Bonjour

- Apple (primarily)
- Goal: advertise & discover services on a LAN
- allocate addresses without a DHCP server
 - Use 169.254/16 **zeroconf** range
- translate between names and IP addresses **without a DNS server**
 - **Multicast DNS**: Use IP multicast for DNS queries (**mDNSResponder**)
- **locate or advertise services** without using a directory server
 - Use DNS services (DNS or multicast DNS)
 - Structured Instance Names
 - SRV record: **<Instance>.<Service>.<Domain>** gives **target IP, port**
 - TXT record with same name: extra info as key/value pairs
 - PTR record: service type to see all instances of the service

SRV example

- Example DNS SRV record

```
myprinter._printer._tcp.local. 120 IN SRV 0 0 5432 myserver.local.
```

The diagram shows three callout boxes pointing to specific parts of the SRV record: 'TTL' points to the value '120', 'port' points to the port number '5432', and 'host' points to the target host 'myserver.local.'.

- DNS TXT record

- May contain additional information.

- Example:

- Different print queues for printer services on the same IP address

- Information is application-specific

- PTR record

```
_printer._tcp.local. 28800 PTR myprinter._printer._tcp.local.
```

- Allows one to query DNS for all services of type _printer.

Bonjour steps

- New device
 - Is there a DHCP server?
 - If yes, get IP address and routing info
 - If no, pick an address in the link-local (zeroconf) range: 169.254.0.0
 - Test the address and claim it if nobody responds
 - Start up Multicast DNS responder
 - Requests a chosen hostname
 - Multicasts query to see if it's taken
 - Claims it if not taken
 - Start up service (get port)
 - Publish service (friendly name, service name, address, port)
 - Create SRV record `friendly_name.service_name._tcp.local` that points to the hostname and port for the service
 - Create PTR record `service_name._tcp.local`

UPnP strategy

- Send data only over network
 - No executables
- Use standard protocols
- Leverage standards
 - HTTP, XML
- Basic IP network connectivity

Communication

- Between...
 - Control points
 - Controller usually client
 - Device controlled
 - Usually server
- Device may take on both functions



Step 0

- Control point and device get addresses
 - DHCP
 - Or AutoIP
 - IETF draft: automatically choose IP address in ad-hoc IPv4 network
 - Pick address in 169.256/16 range – see if it's used



Step 1

- Control point finds device
 - Devices advertise (broadcast) when added
 - Periodic refresh
 - Control points search as needed
 - Devices respond
 - Search for types of service
 - Guarantee minimal capabilities



Step 2

- Control point learns about device capabilities
 - SSDP: Simple Service Discovery Protocol
 - IETF draft
 - Administratively scoped multicast
 - Unicast responses
 - Get URL for description
 - Actions, state variables expressed in XML



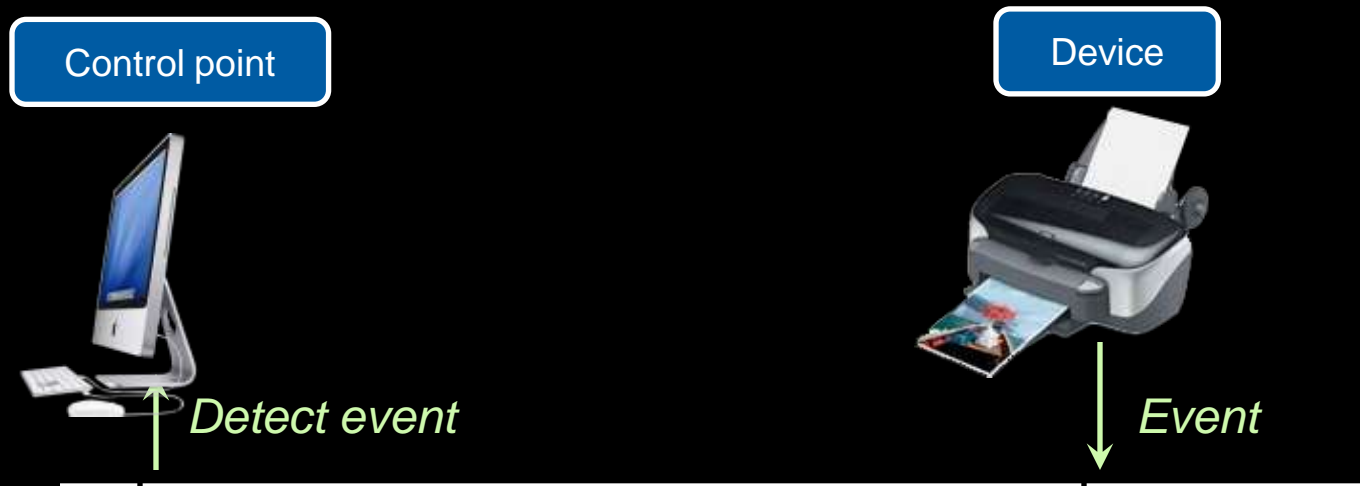
Step 3

- Control point invokes actions on device
 - Send request, get result
 - SOAP messages



Step 4

- Control point listens to state changes of device
 - Push model
 - GENA: General Event Notification Architecture



Step 5

- Control point controls device and/or views device status with HTML



The End