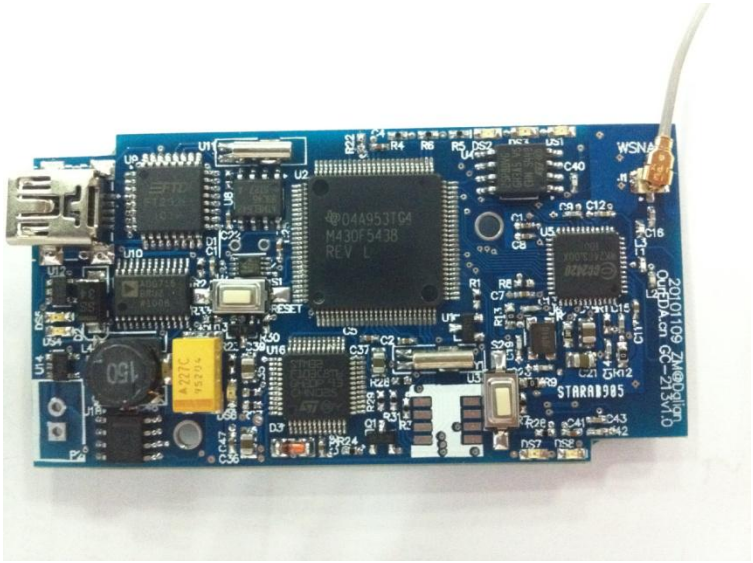
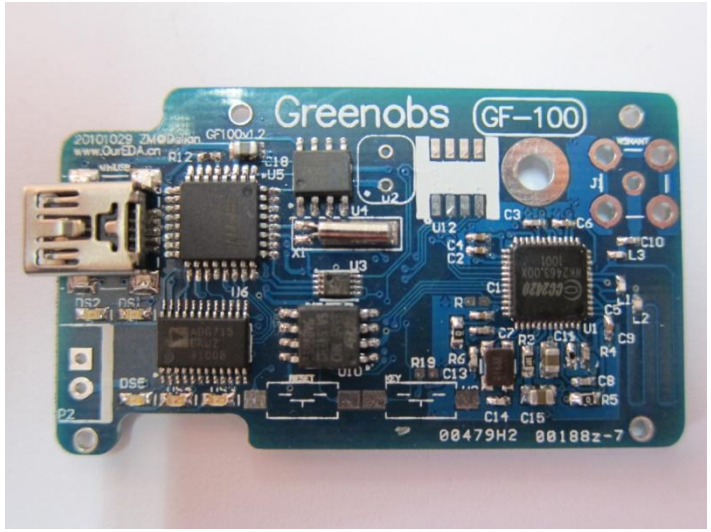
CS557: Cyber-Physical Systems

Wireless Sensor Networks

Technologies, Systems, and Applications

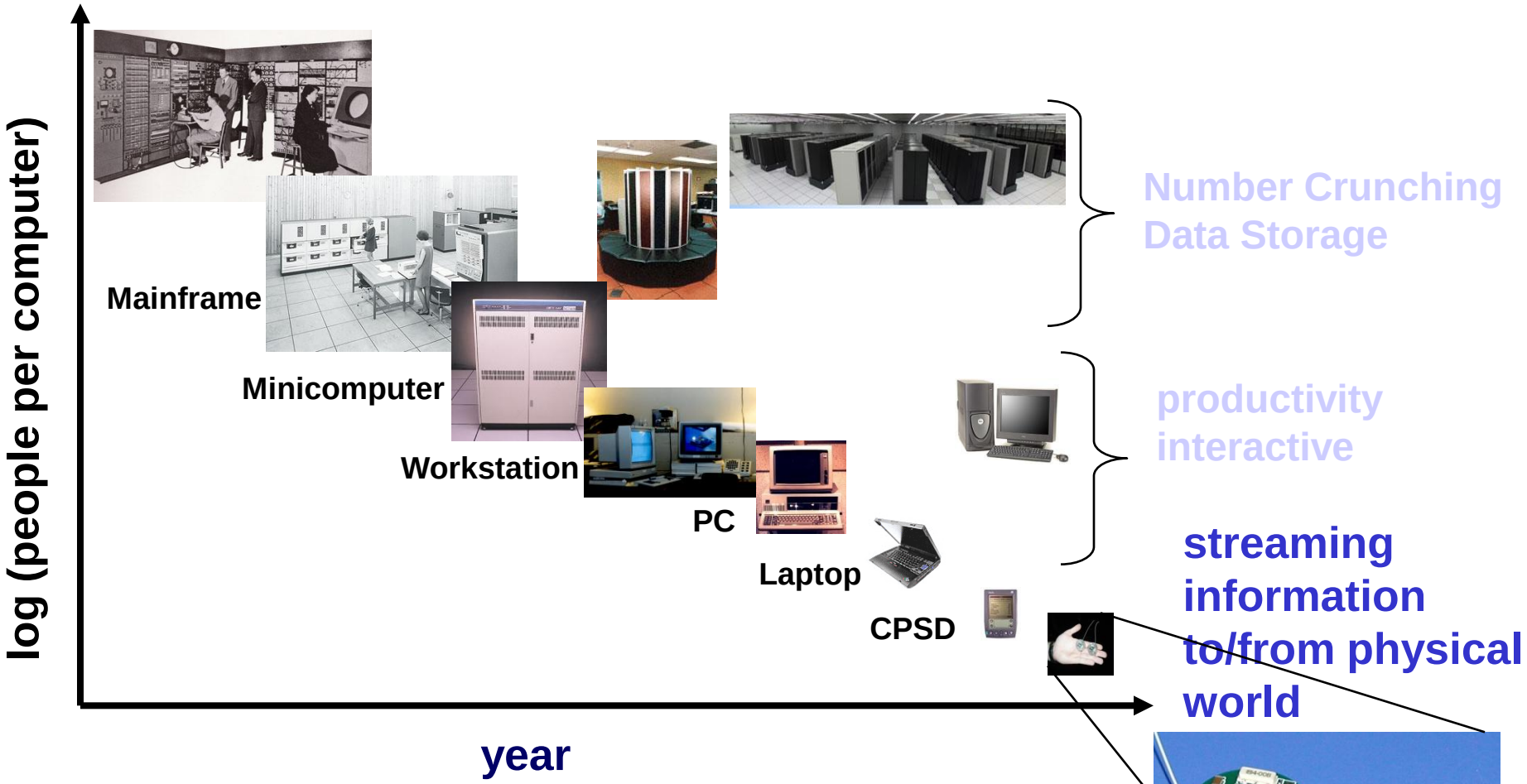
Based on lectures by
Prabal Dutta, WenZhan Song, and many others

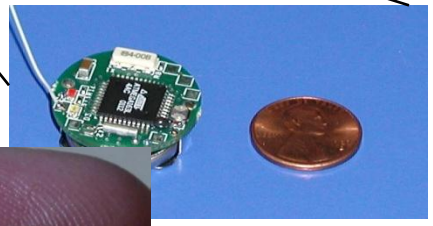


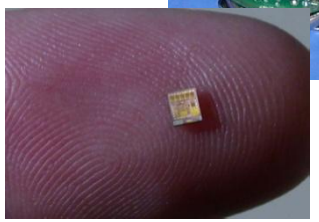


BACKGROUND

Bell's Law of Computer Classes:
A new computing class roughly every decade







"Roughly every decade a new, lower priced computer class forms based on a new programming platform, network, and interface resulting in new usage and the establishment of a new industry."

Adapted from
D. Culler

Moore's Law:
IC transistor count doubles every two years

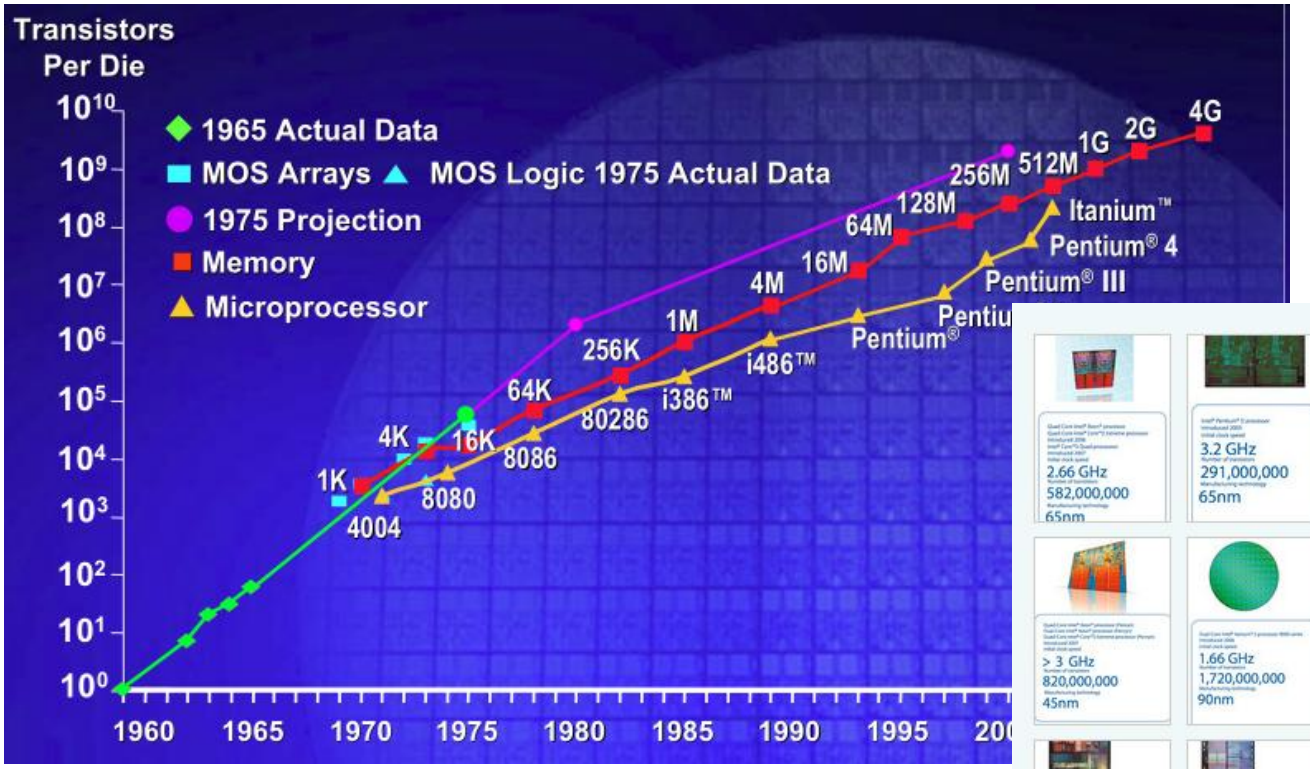
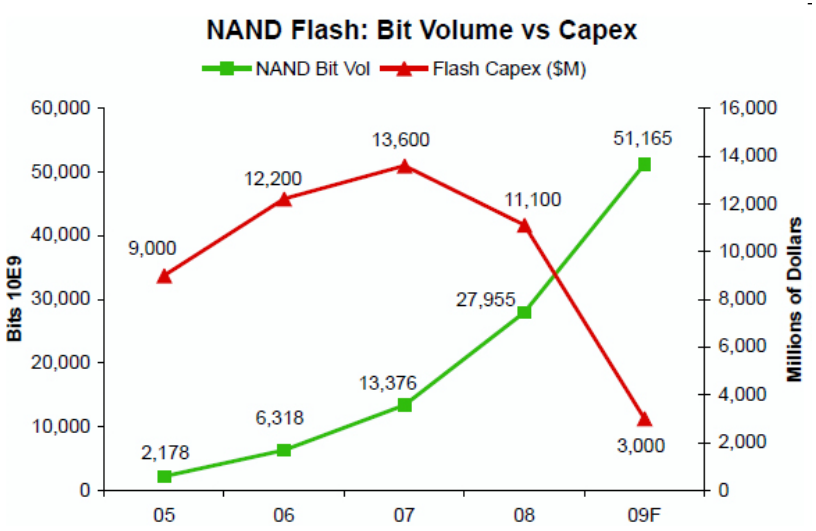


Photo Credit: Intel

Flash memory scaling:
Rise of density & volumes; Fall (and rise) of prices



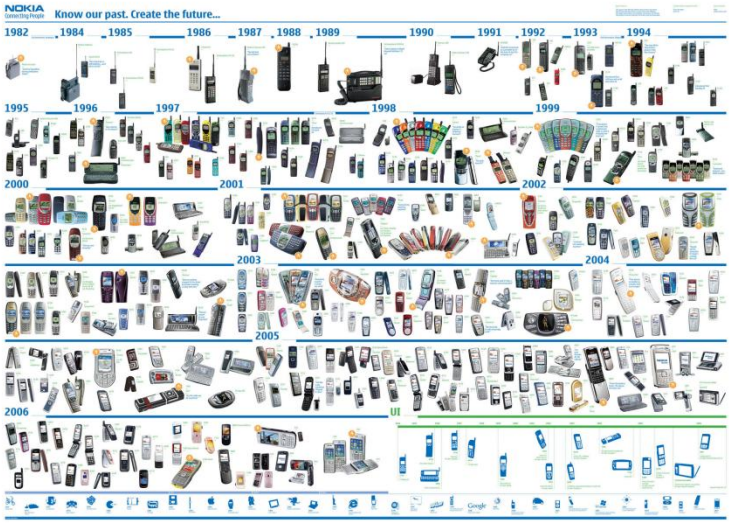
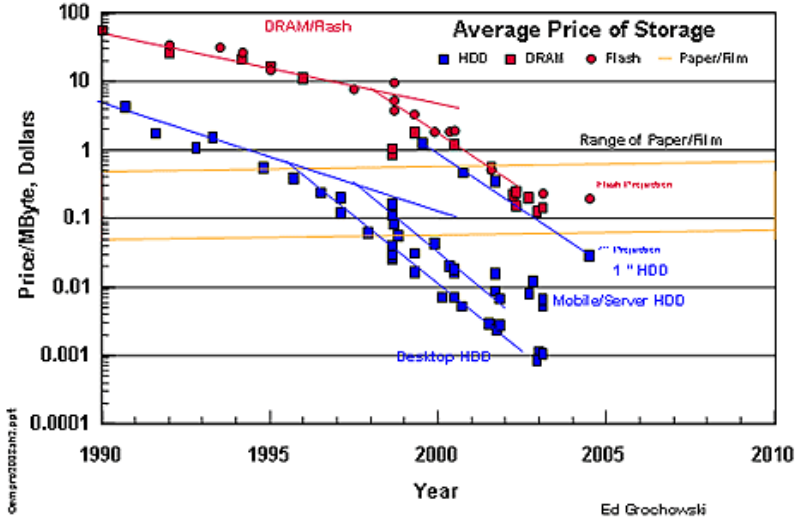
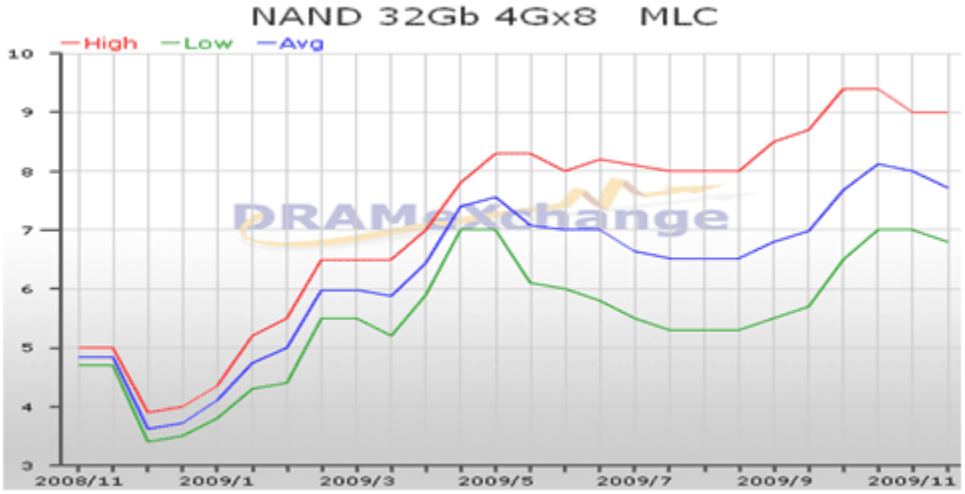


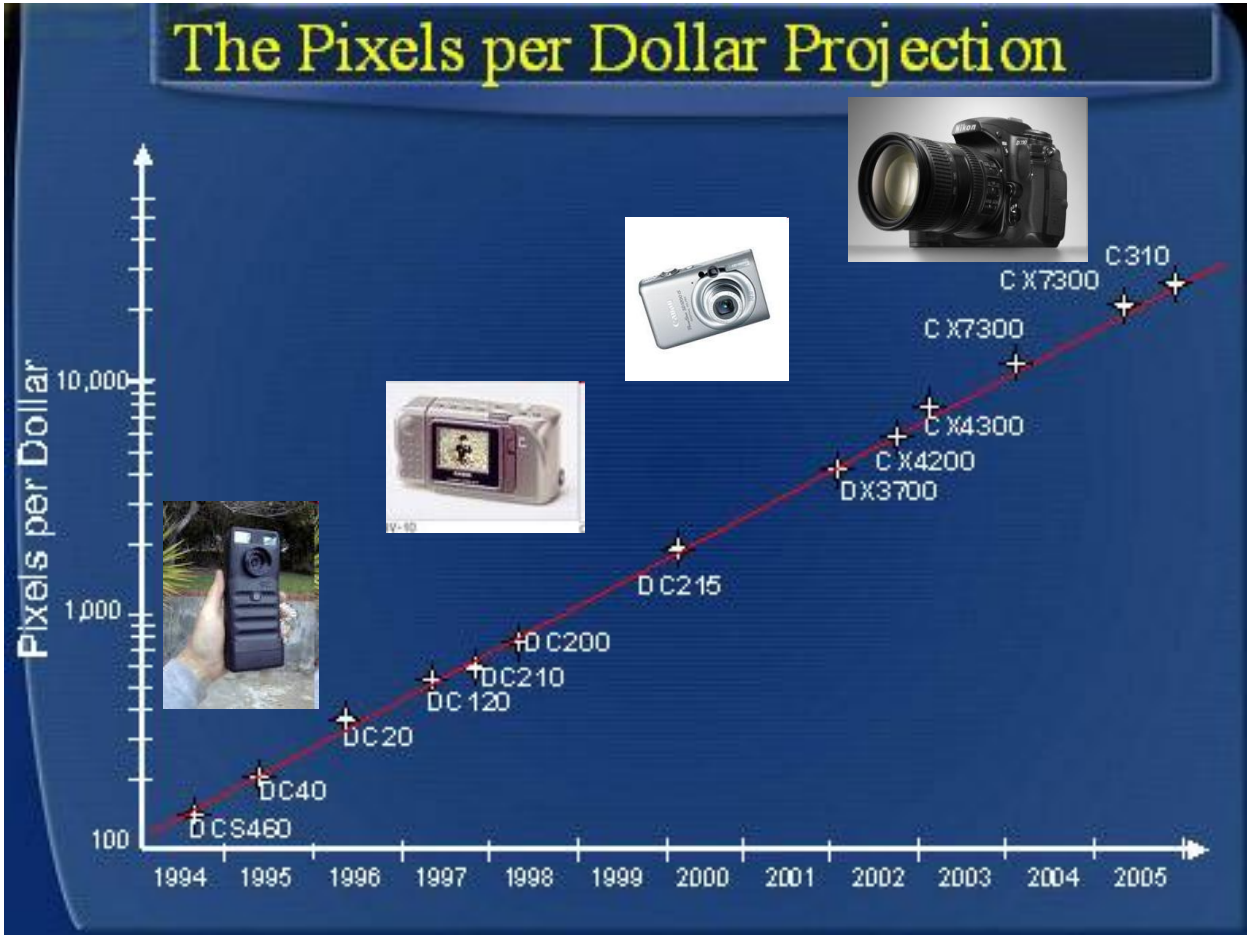
Figure-1 32Gb MLC NAND Flash contract price trend





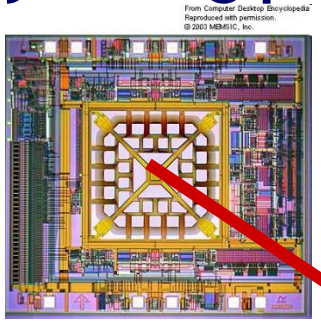
Source: DRAMeXchange, Nov. 2009.

Hendy's "Law": Pixels per dollar doubles annually

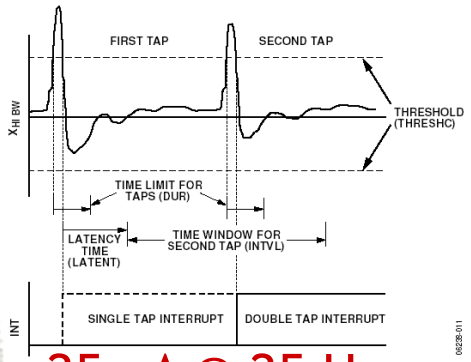


Credit: Barry Hendy/Wikipedia

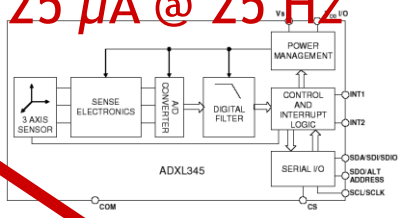
MEMS Accelerometers: Rapidly falling price and power



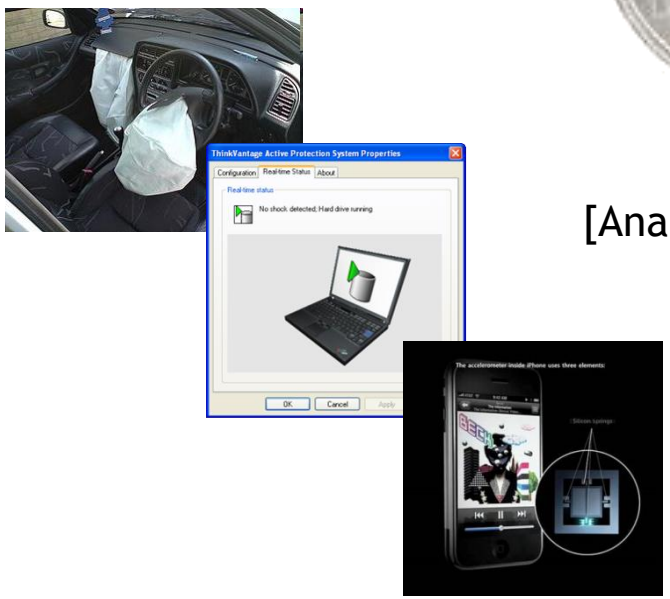
O(mA)



25 μA @ 25 Hz

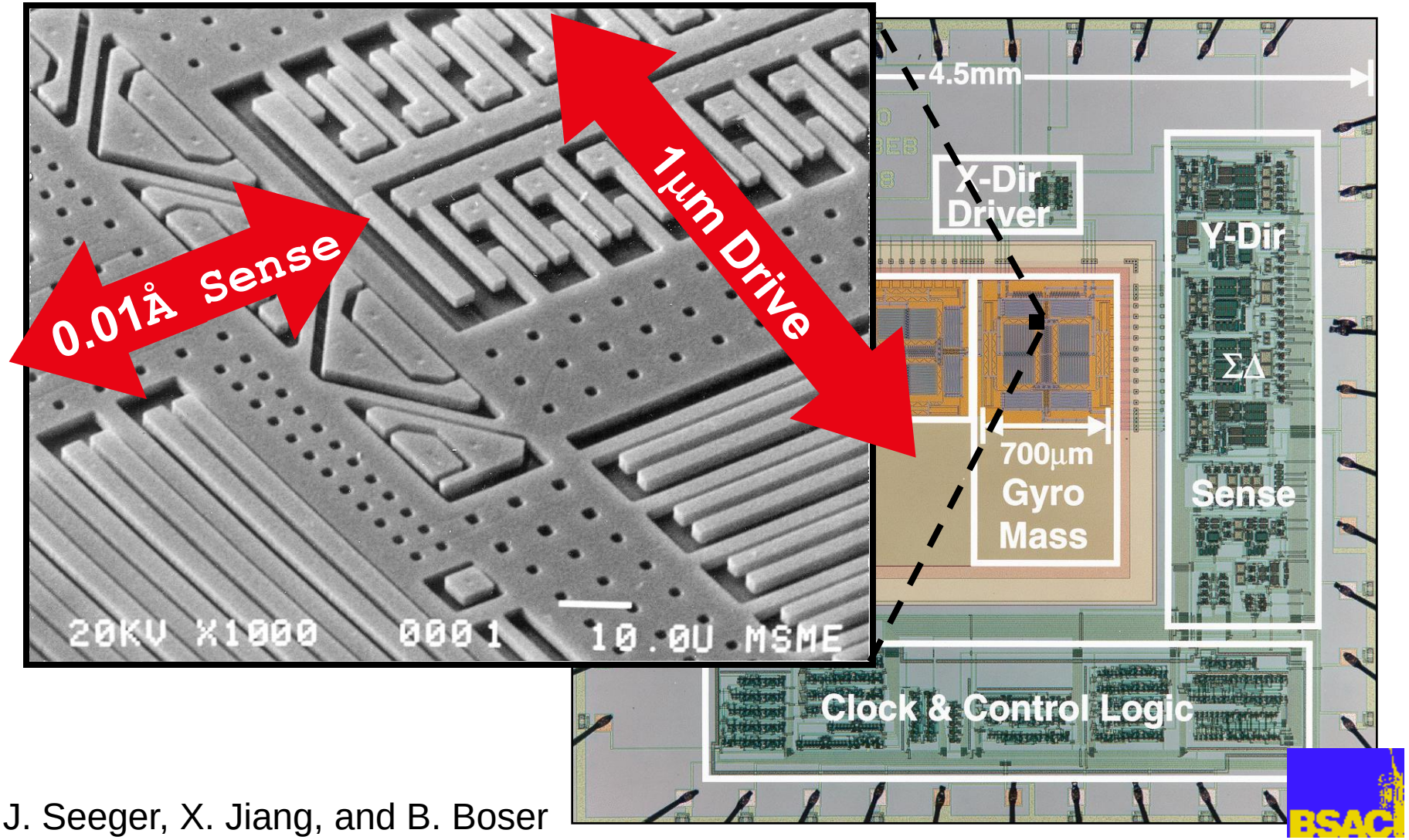


ADXL345
[Analog Devices, 2009]



10 μA @ 10 Hz @ 6 bits
[ST Microelectronics, annc. 2009]

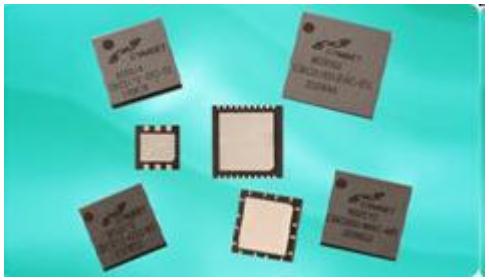
MEMS Gyroscope Chip



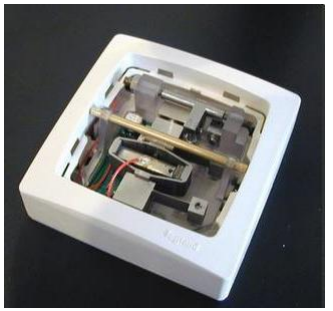
J. Seeger, X. Jiang, and B. Boser

Energy harvesting and storage: Small doesn't mean powerless…

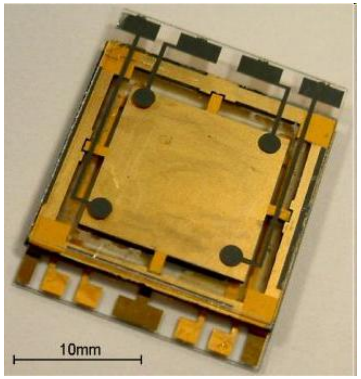
RF [Intel]



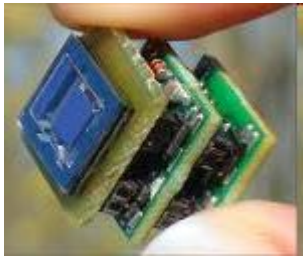
Thin-film batteries



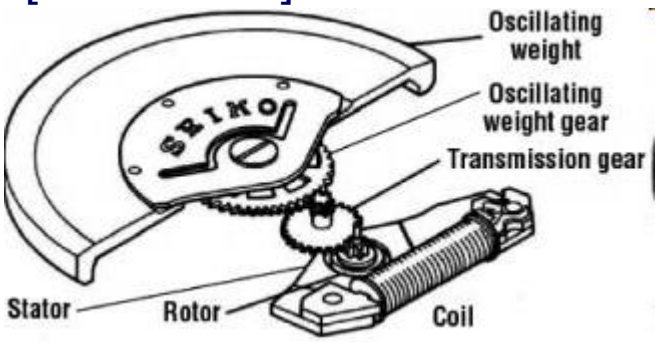
Shock Energy Harvesting
CEDRAT Technologies

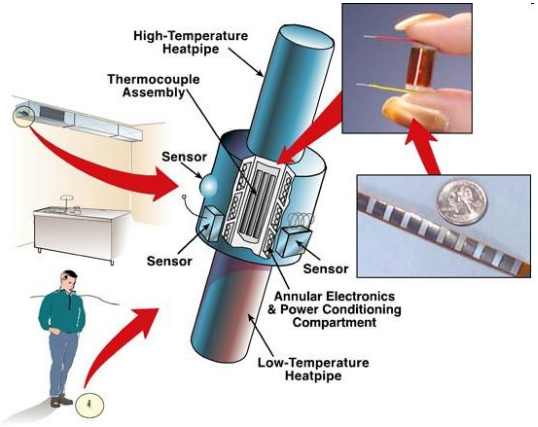


Electrostatic Energy Harvester [ICL]



Piezoelectric [Holst/IMEC]





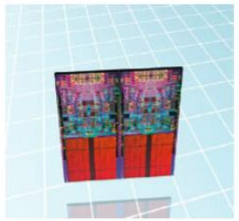
Thermoelectric Ambient Energy Harvester [PNNL]

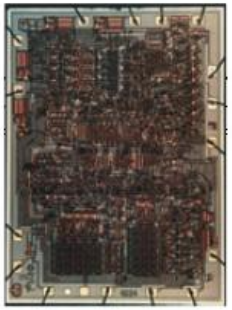
Perpetual operation: Living off the land (or air)

[Sample09]
Harvest 60 μW
From 4.1 km away
Using 5 dBi antenna
From a 960 kW TV station
On channel 48

[Powercast09]
Harvest milliwatts
From centimeters away
Using a ~ 0 dBi antenna
From your iPhone

Bell's Law, Take 2: Corollary to the Laws of Scale

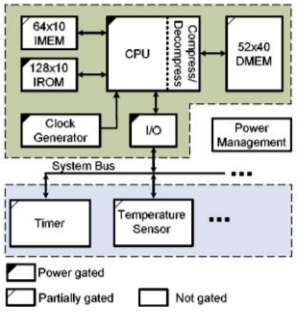


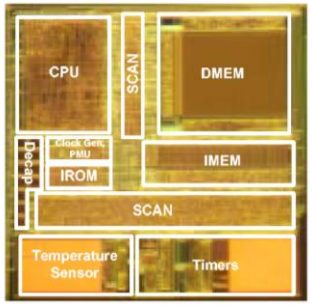


Intel® 4004 processor
Introduced 1971
Initial clock speed
108 KHz
Number of transistors
2,300
Manufacturing technology
10µ

15x size decrease
40x transistors
55x smaller λ

Quad-Core Intel® Xeon® processor
Quad-Core Intel® Core™2 Extreme processor
Introduced 2006
Intel® Core™2 Quad processors
Introduced 2007
Initial clock speed
2.66 GHz
Number of transistors
582,000,000
Manufacturing technology
65nm





UMich Phoenix Processor
Introduced 2008
Initial clock speed
106 kHz @ 0.5V Vdd
Number of transistors
92,499
Manufacturing technology
0.18 µ

Photo credits: Intel, U. Michigan

Outline

Fundamental Technology Trends

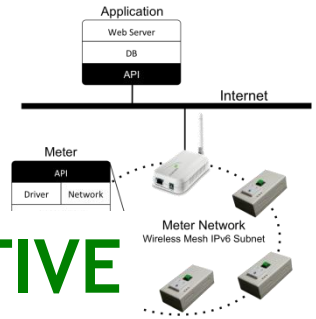
Emerging Application Drivers

Hardware/Software Systems

A decade of sensor network applications

Low Mobility High

Power: High / Low



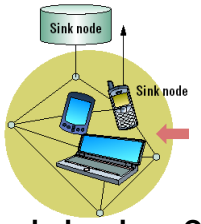
[Werner-A

ACTIVE

ang09]

ifton07]

"Macroscopes"



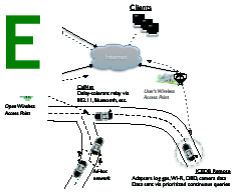
[Abdel

04]

ACTIVE

"Mob

[Aoki

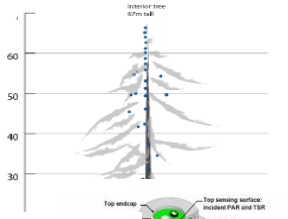


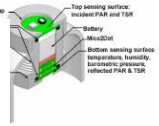
[Hull06]

[Wark07]

ncz08]

"Power

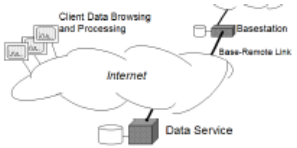




SOLVED

EMERGING

[Tolle05]



[Szewczyk04]

"Mic

Mobisc

[Malinowski

[Thiele08]

ZigBee Alliance

WirelessHART

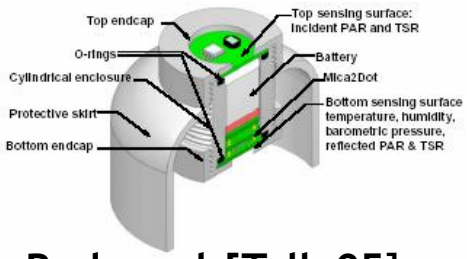
ISA

ipso Alliance

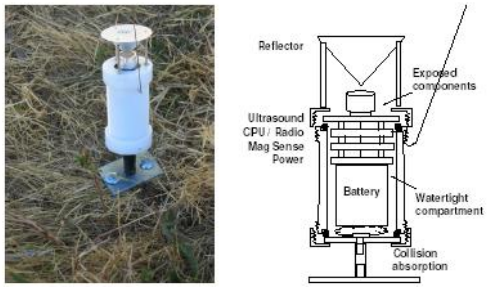
IETF

IEEE

Monitoring trees, cars, people, watersheds, and tires



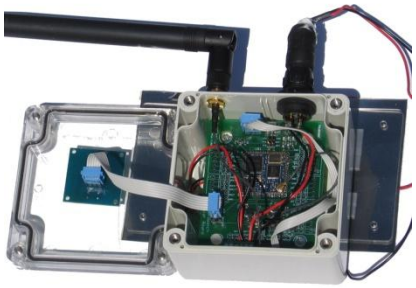
Redwoods[Tolle05]



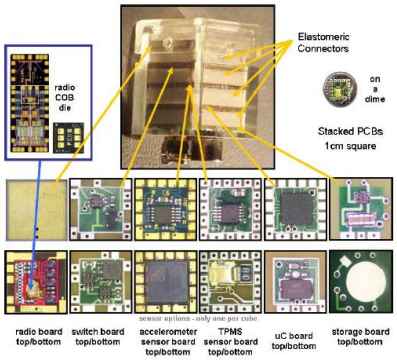
PEG [Sharp05]

Radar [Dutta06]

Shimmer[Intel06]



HydroWatch [Taneja07]



PicoCube[Chee08]

Mobiscopes

High Mobility

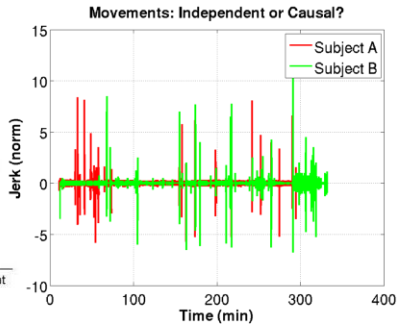
Common Sense

Higher

Sleep Tracker

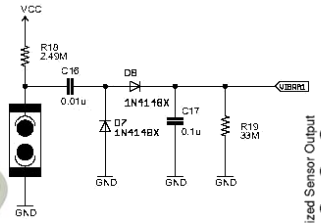
FIGURE 1-1

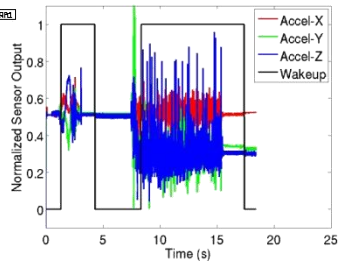
My wife tells me that my legs were restless again last night. She said that she didn't sleep a wink, but I slept fine.

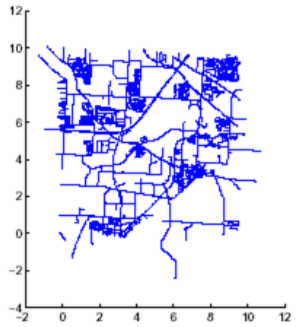


Power

Auto Witness

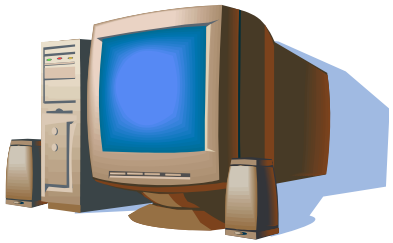


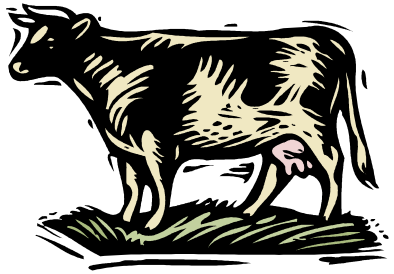




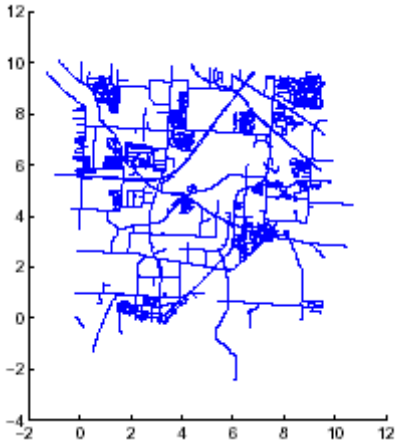
Lower

AutoWitness





- "LoJack" for everyday things
  - You: Tag your things
  - City
    - Deploy a network of detectors
    - Track stolen objects
- Cost and convenience dictates
  - $10/tag
  - 10 years of lifetime
  - Small size
- Tags must run on 2 uA average current
  - Sleep for years normally
  - Active for days after theft



[Mitra09]

Periodic Limb Movements

- Affects 30% of those over 65
- PLM = limb muscle jerks mostly while asleep
  - Not Restless Legs Syndrome
  - Formerly called *noctural myoclonus*
  - Not hypnic jerks (they're lower freq)
- Diagnosing the symptoms
  - Most are unaware of their kicking
  - Must link movements to poor sleep
  - Requires nights in sleep study lab
- Treatment depends on whether motion
  - Disturbs only the bed partner
  - Disturbs the patient

FIGURE 1-1

My wife tells me that my legs were restless again last night. She said that she didn't sleep a wink, but I slept fine.

[Buchfurher06]

The bed partner's complaint of being kicked is often the main reason for seeking medical help for PLMS, because maintaining harmony in a relationship that is disrupted by nighttime kicking can be an important issue.

Outline

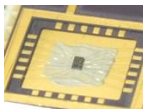
Fundamental Technology Trends

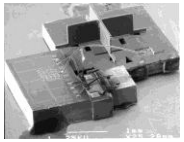
Emerging Application Drivers

Hardware/Software Systems

The System Challenge

Monitoring & Managing Spaces and Things





applications

data mgmt

service

network

system

architecture

technology

MEMS sensing

Comm.

Proc

Store

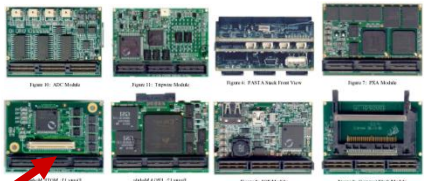
uRobots actuate

Power

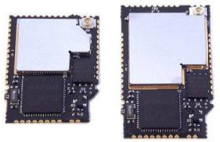
Miniature, low-power connections to the physical world

Despite the tight application-platform coupling, modular platform decompositions remained in focus

PC/104 [Cerpa01]



PASTA [Bajura05]



WeC [Hill98]

Rene [Hill99]

Mica [Hill01]

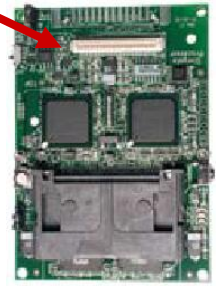
Mica2 [Xbow03]

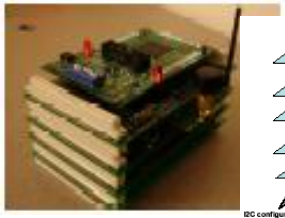
MicaZ [Xbow05]

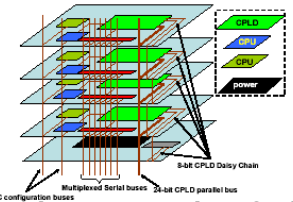
Iris [Xbow07]

WINSng [Pottie00]

WINS [Rockwell]

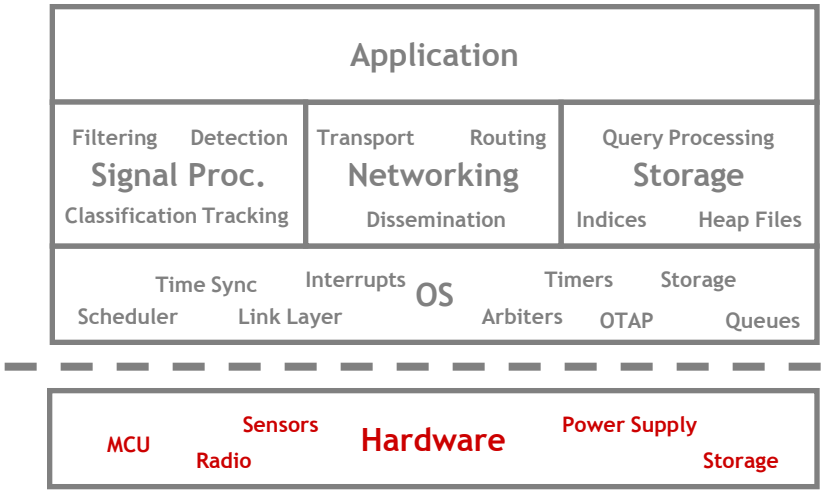


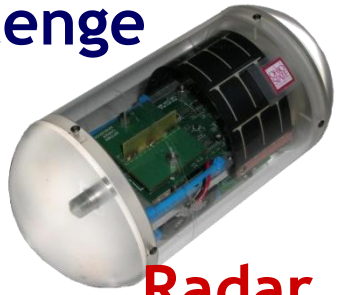




mPlatform[Lymberopoulos07]

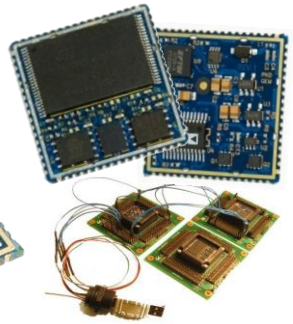
Hardware Platforms: Addressing the application-specific challenge



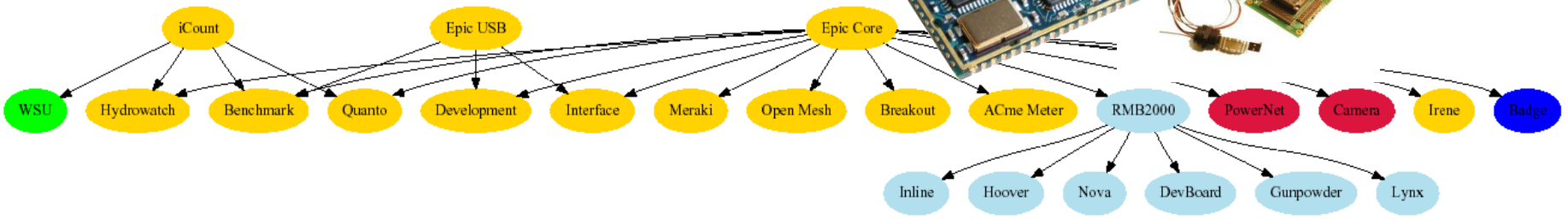


Radar

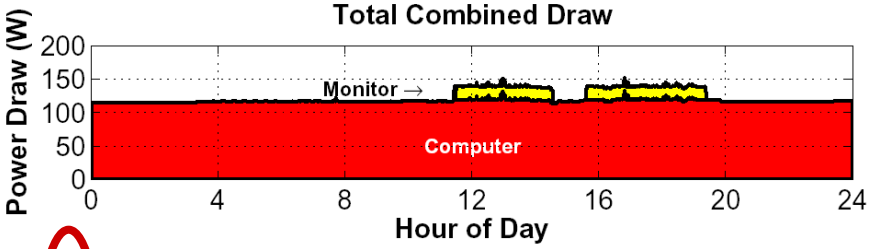
Trio



ExScal

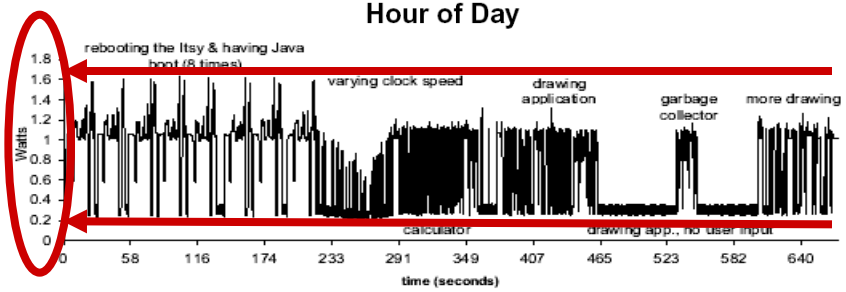


Low-Power, Duty-Cycle Operation is Critical

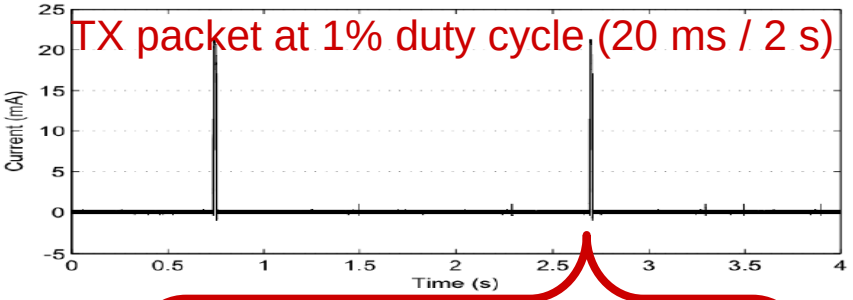


86,400,000 ms

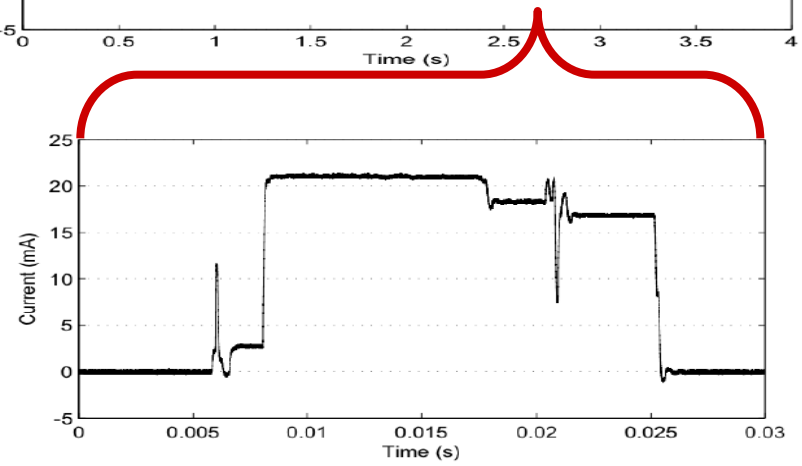
[Farkas00]



640,000 ms



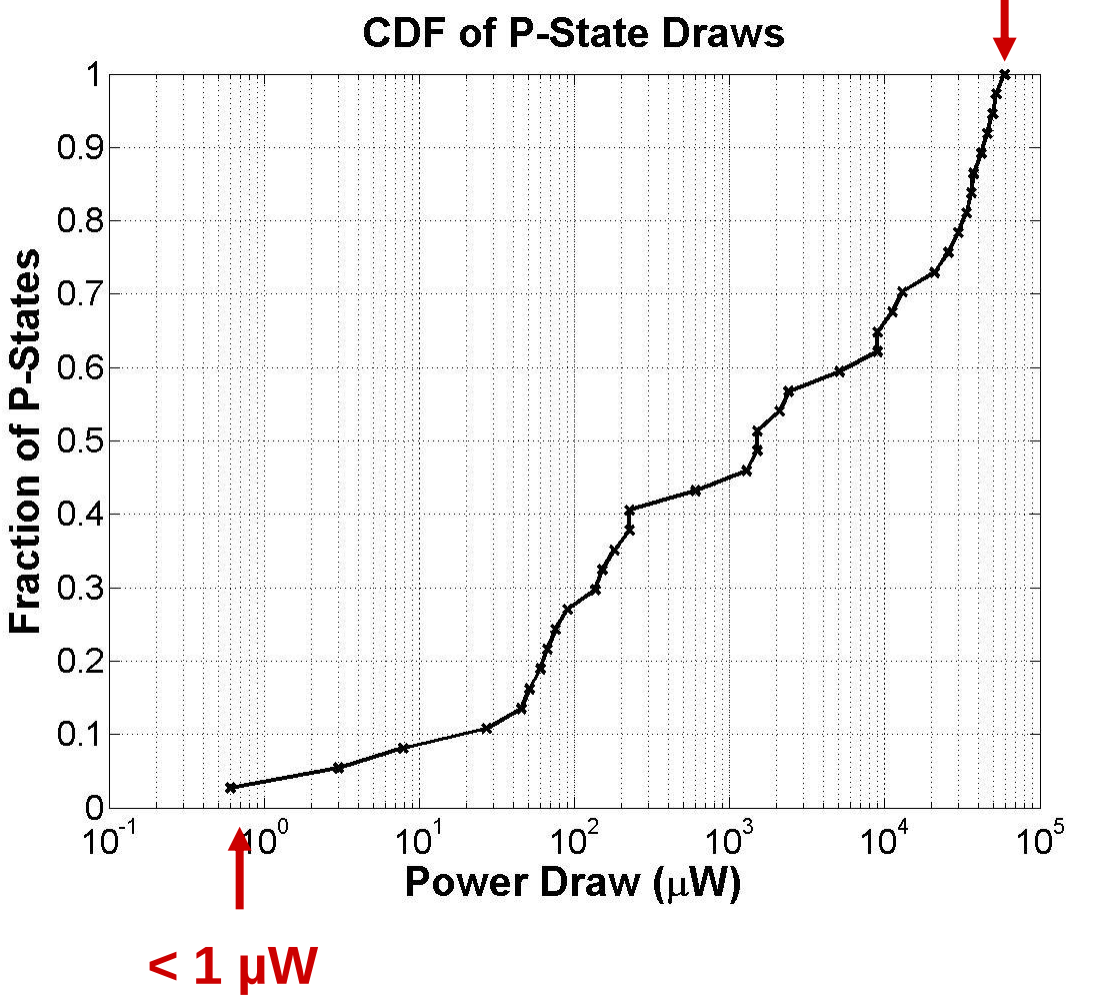
4,000 ms



30 ms

Dynamic range in power draw exceeds 10,000:1

> 50 mW



< 1 μW

Energy Sink	Power State	Current
Microcontroller		
CPU	ACTIVE	500 μA
	LPM0	75 μA
	LPM1†	75 μA
	LPM2	17 μA
	LPM3	2.6 μA
	LPM4	0.2 μA
Voltage Reference	ON	500 μA
ADC	CONVERTING	800 μA
DAC	CONVERTING-2	50 μA
	CONVERTING-5	200 μA
	CONVERTING-7	700 μA
Internal Flash	PROGRAM	3 mA
	ERASE	3 mA
Temperature Sensor	SAMPLE	60 μA
Analog Comparator	COMPARE	45 μA
Supply Supervisor	ON	15 μA
Radio		
Regulator	OFF	1 μA
	ON	22 μA
	POWER_DOWN	20 μA
Batter Monitor	ENABLED	30 μA
Control Path	IDLE	426 μA
Rx Data Path	RX (LISTEN)	19.7 mA
Tx Data Path	TX (+0 dBm)	17.4 mA
	TX (-1 dBm)	16.5 mA
	TX (-3 dBm)	15.2 mA
	TX (-5 dBm)	13.9 mA
	TX (-7 dBm)	12.5 mA
	TX (-10 dBm)	11.2 mA
	TX (-15 dBm)	9.9 mA
	TX (-25 dBm)	8.5 mA
Flash	POWER_DOWN	9 μA
	STANDBY	25 μA
	READ	7 mA
	WRITE	12 mA
	ERASE	12 mA
LED0 (Red)	ON	4.3 mA
LED1 (Green)	ON	3.7 mA
LED2 (Blue)	ON	1.7 mA

Energy-efficiency affects all layers of system design

eSENSE: Energy Efficient Stochastic Sensing Framework for Wireless Sensor Platforms

ata Gathering in Sensor Networks

Haiyang Liu, Abhishek Chandra and Jaideep Srivastava
Dept. of Computer Science and E
Minneapoli
{hliu, chandra, sriv

al von Rickenbach Roger Wattenhofer

Energy-Efficient Data Representation and Routing for Wireless Sensor Networks Based on a Distributed Wavelet Compression Algorithm *

Lucid Dreaming: Reliable Analog Event Detection for Energy-Constrained Applications

Sasha Jevtic† Mathew Kotowsky‡ Robert P. Dick† Peter A. Dinda† Charles Dowding⋆

sjevtic@eecs.northwestern.edu, {kotowsky, dickrp, pdinda, c-dowding}@northwestern.edu

†EECS Dept. Northwestern University
‡Infrastructure Technology Inst. Northwestern University
⋆Civil & Environmental Engg. Northwestern University

nio Ortega and Bhaskar Krishnamachari
tems, University of Southern California
lifornia, USA
ega@sipi.usc.edu, bkrishna@usc.edu

otimized Image Communication on
al Detection under the Platforms
eless Sensor Networks*

nmad Rahimi [2]
or [1]
a, Los Angeles
ornia, Los Angeles
, Los Angeles
mhr@cens.ucla.edu

Energy-efficient Coverage for Target Detection in Wireless Sensor Networks

um
sity
PA 18015
ehigh.edu

Wei Wang, Vikram Srinivasan, Kee-Chaing Ch
Department of Electrical and Computer Er
National University of Singapore
{wang.wei,elevs,eleckc}@nus.edu.sg,wangban

Power Scheduling for Wireless

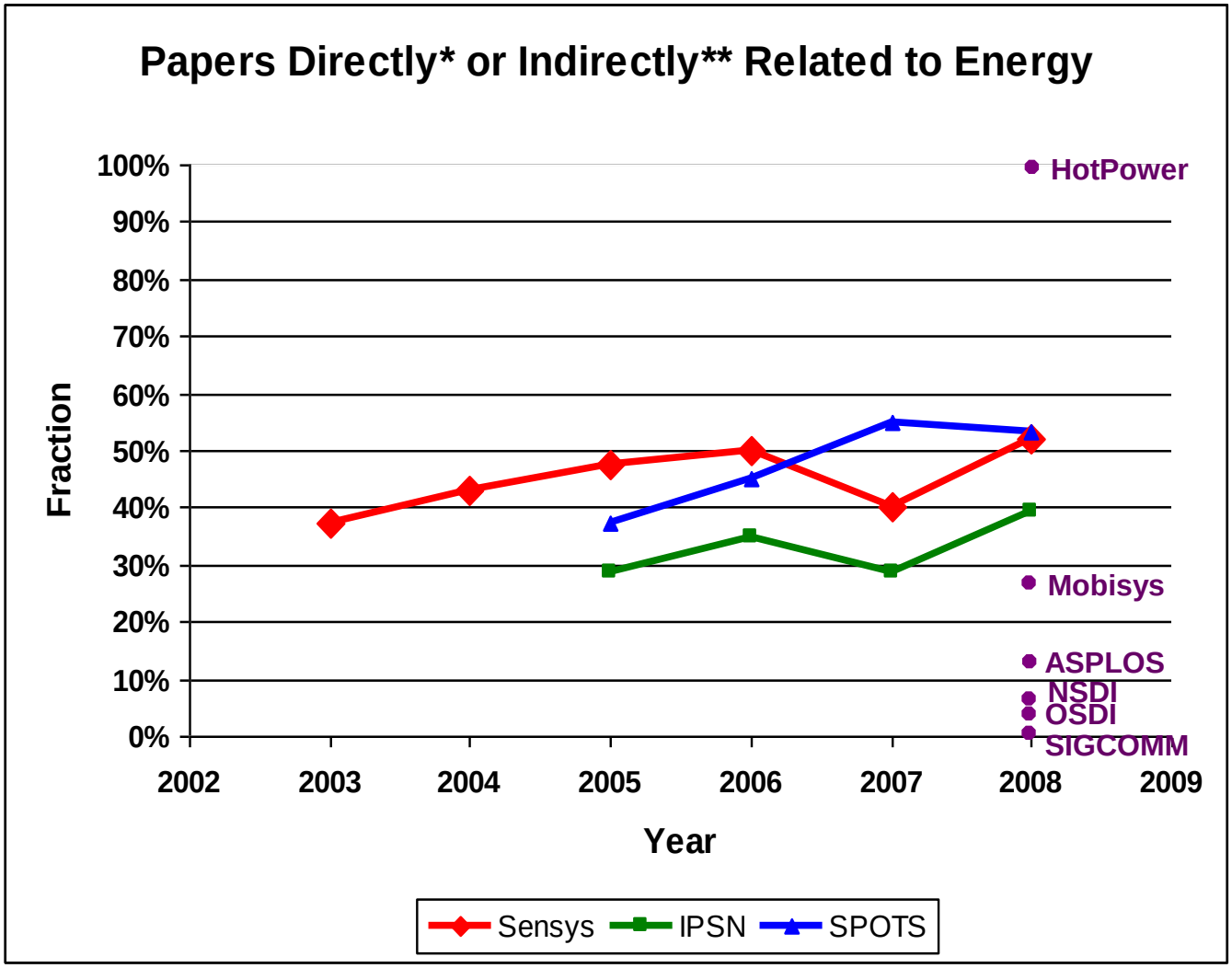
Energy-efficient routing in wireless sensor networks for delay sensitive applications

D. Ranganathan, P. K. Pothuri, V. Sarangan, and S. Radhakrishnan

Rice University
Houston, Texas 77251-1892
crozell@rice.edu

Rice University
Houston, Texas 77251-1892
dhj@rice.edu

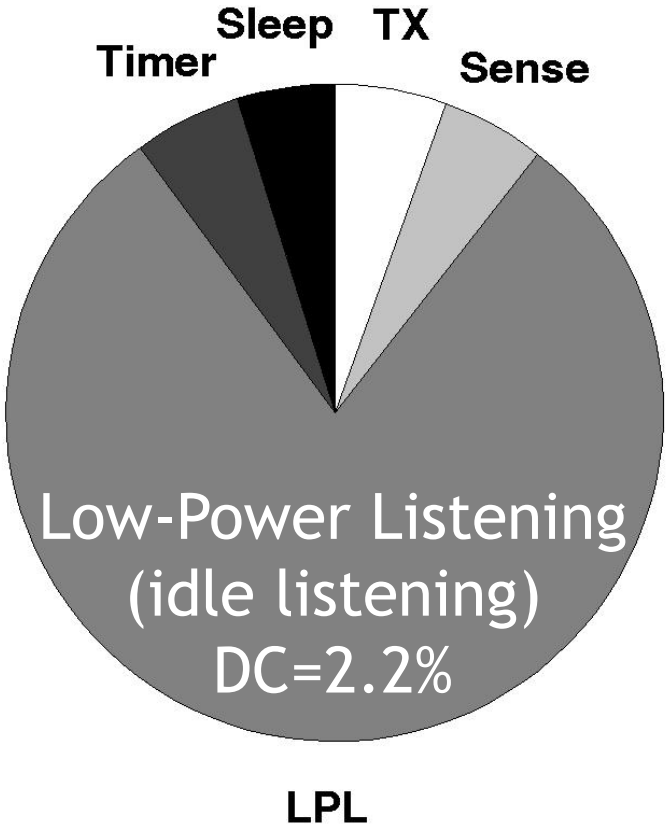
Energy-efficiency pervades the sensornet research agenda



* Directly: Energy is part of the title, abstract, or central argument

** Indirectly: Energy is part of the motivation, evaluation, or key tradeoff

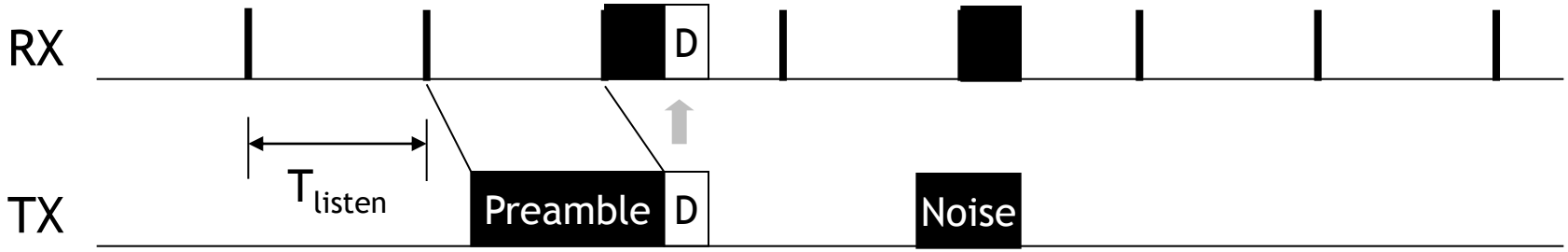
Radio idle listening dominates the power budget, even at radio duty cycles of just 1 to 2%

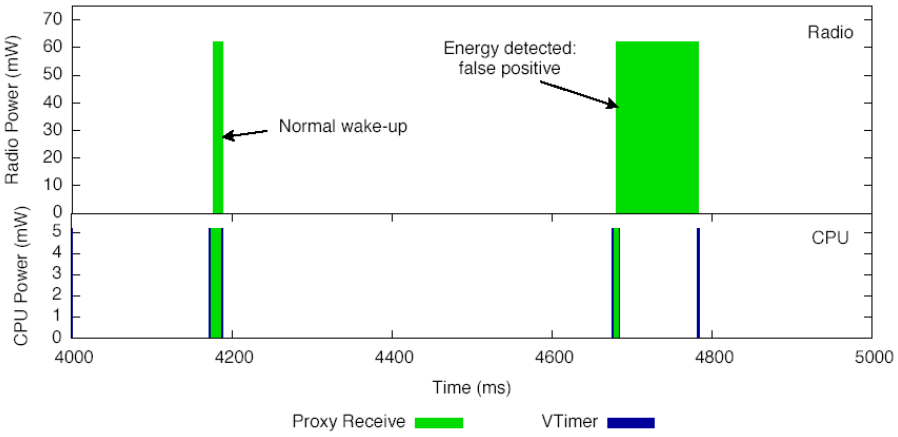


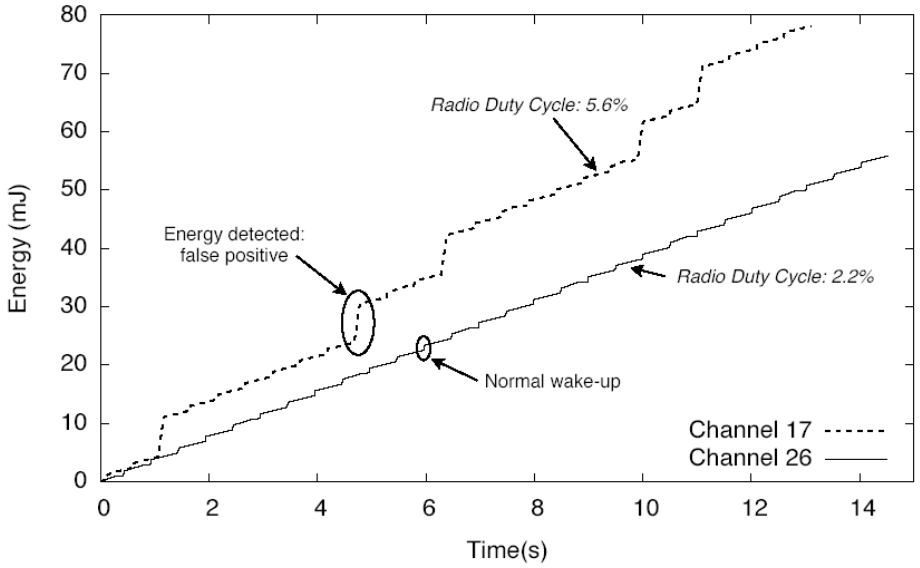
R. Szewczyk, A. Mainwaring, J. Polastre, D. Culler,
"An Analysis of a Large Scale Habitat Monitoring Application",
ACM SenSys' 04, November, 2004, Baltimore, MD

Therefore, the radio is kept *mostly off*

Low-Power Listening





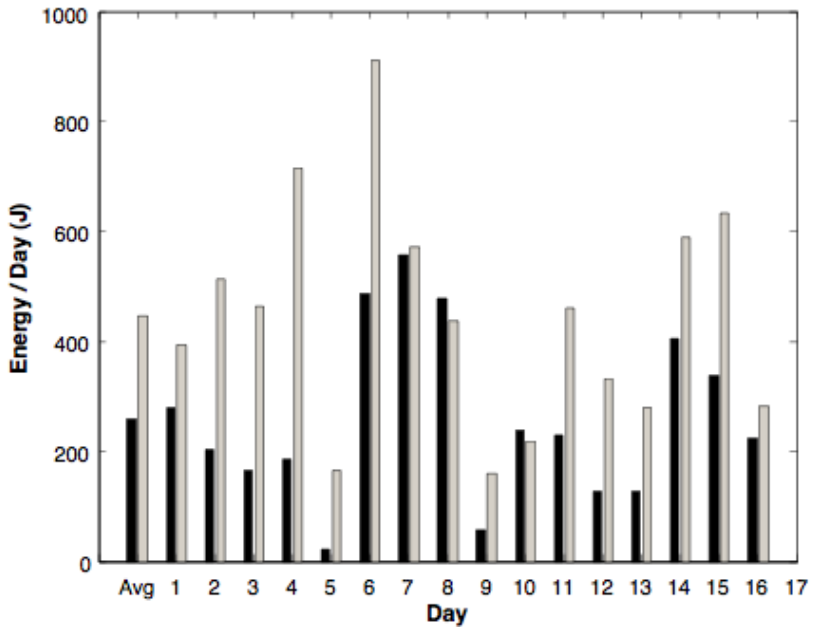


Overhearing adds significant unpredictability to node lifetime

Mobility makes energy and communication challenges fundamentally harder in low-power systems

- Energy
  - Must carry it along
  - Or harvest it from the ambient environment
  - And deal with inherent uncertainty of harvesting
  - Implies dynamic duty cycles
  - How to measure usage?
- Link
  - Topological turmoil
  - Link. What link?
  - Never before seen link
  - What radio channel?
  - When, where to look?
  - Can't just probe during deployment
  - History is a poor/no guide
- Network
  - Stability of routing peers?
  - Routers, hosts, both?
- Transport
  - Disruption-tolerant
  - Store-and-forward

"Weather + mobility = uncertain energy budget"
- Jacob Sorber, Sensys 2007



J. Sorber et al., "Eon: A Language and Runtime for Perpetual Systems", *Sensys'07*, Sydney, Australia

Mobility constrains the platform design

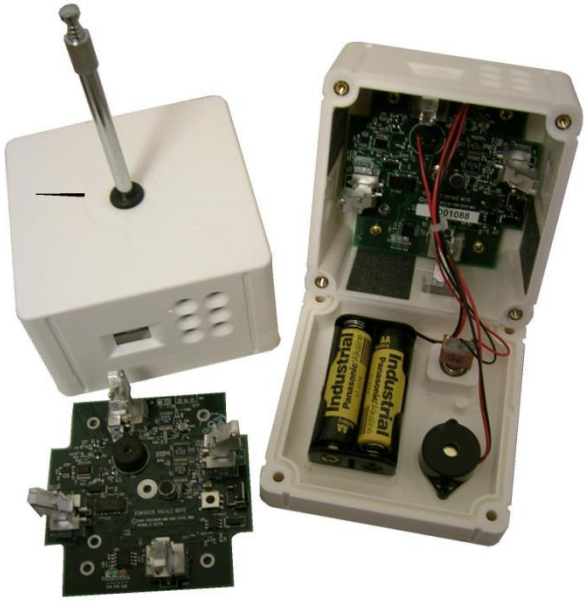
- Limited energy source
  - Fraction of 2 “AA” batteries
- Wider lifetime (average power)
  - Sleep Tracker: Weeks (50 uA)
  - AutoWitness: Decade (2 uA)
- Wider dynamic range
  - O(10 mA) active current
  - O(1 uA) sleep current
  - O(0.001 - 1%) duty cycle
- CPU O(10 MIPS)
- RAM O(10 KB)
- ROM O(100 KB)
- Radio O(100 kbps)
- Flash O(1 MB – 1 GB)

200 mA-Hr

Irene [Dutta09]

200 mA-Hr

RatPack [Thiele08]



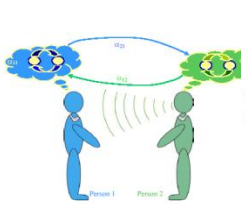
2000 mA-Hr

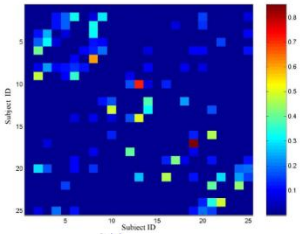
[Dutta05]

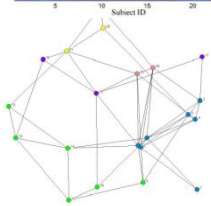
Mobility drives new communication patterns

Talking

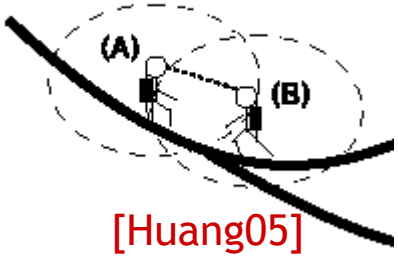
[Liu04]







[Choudury04,07]



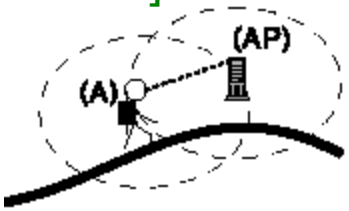
[Huang05]

Docking

[Wark07]

[Malinowski07]

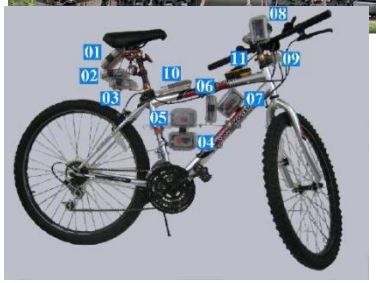
[Borriello04]



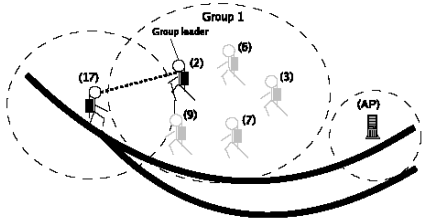
[Huang05]

Flocking

[UP08]

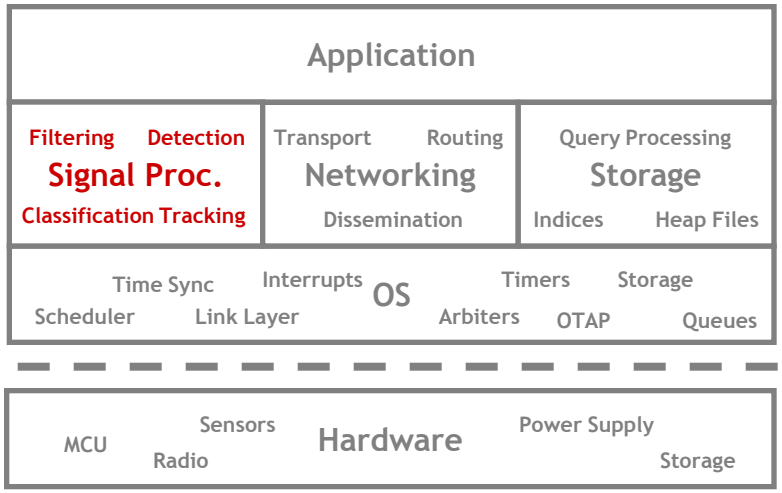


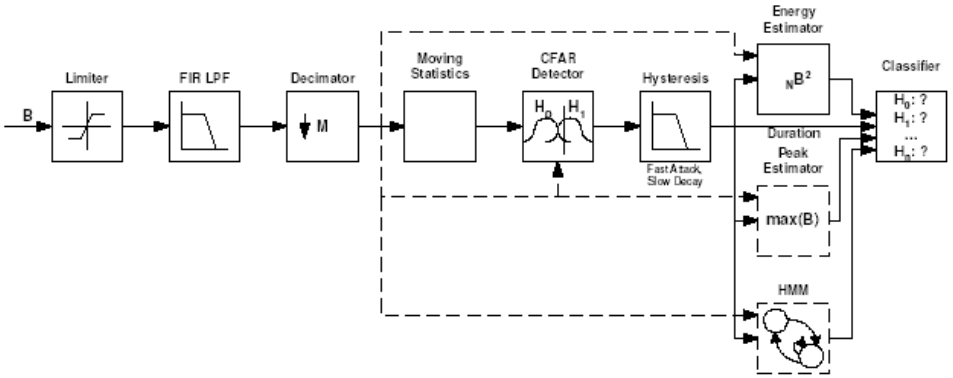
[Eisenman08]

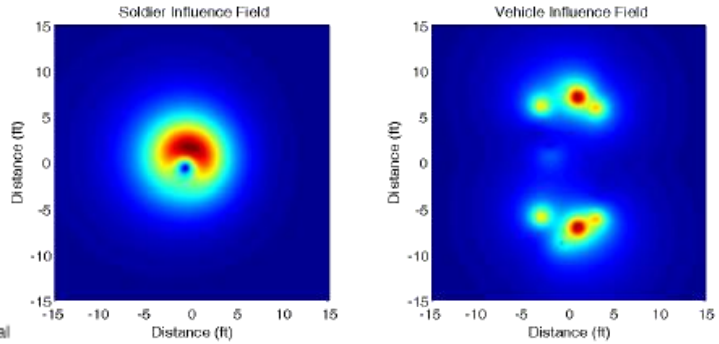


[Huang05]

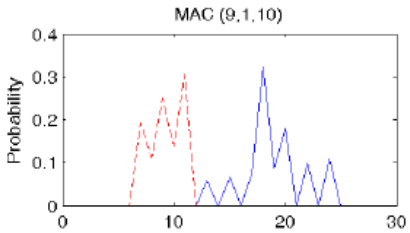
Signal Processing without Floating Point Operations?

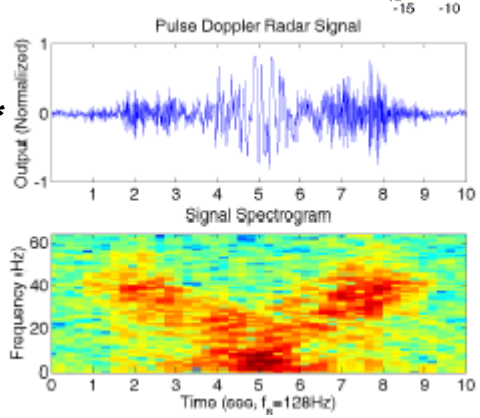


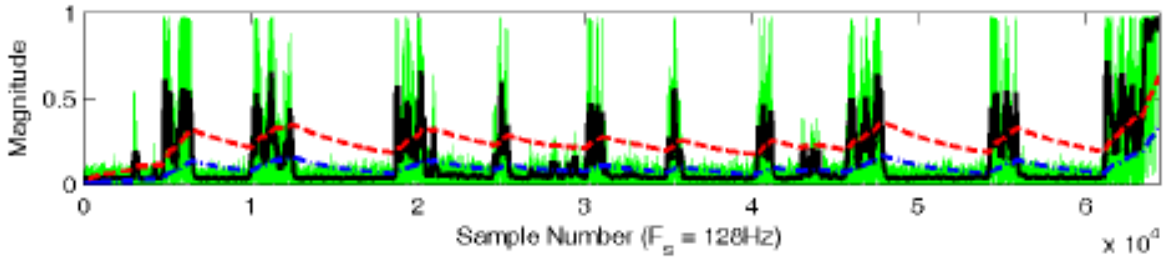


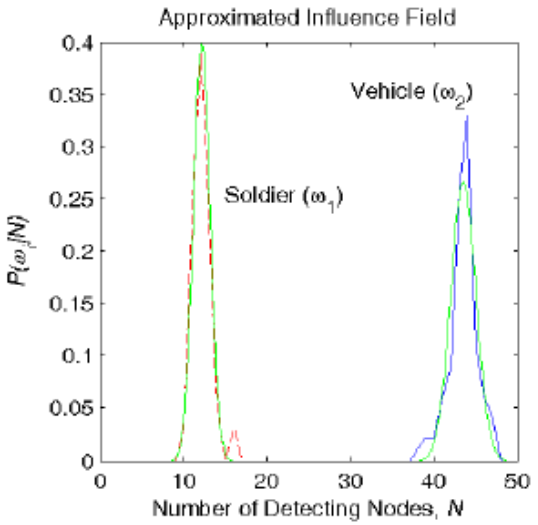


Dutta, et al., "Radar" [IPSN'06]
Dutta, et al., "ExScal" [IPSN'05]*
Arora, *Dutta*, et al. [ComNet'04]*

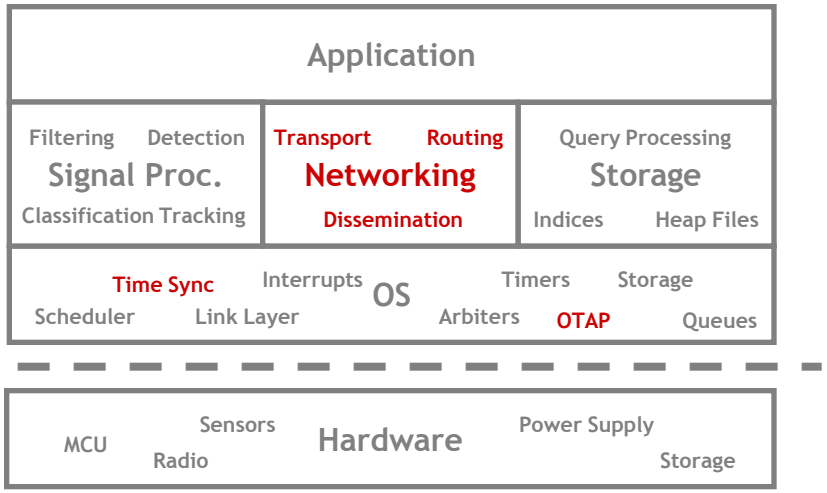


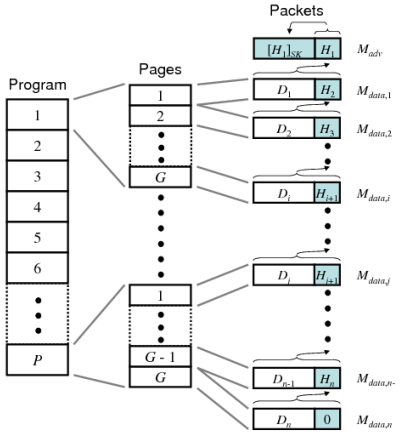






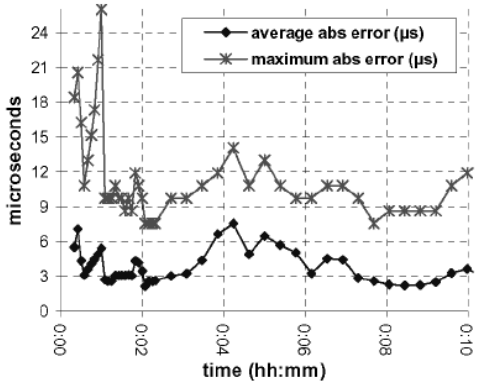
Networking and Middleware



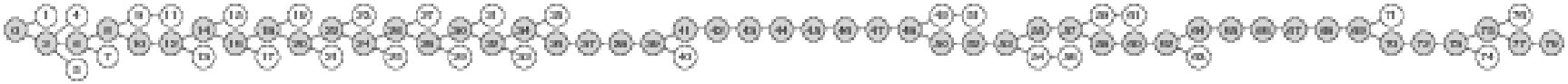


Secure Deluge

IPv6/6lowpan router



Time Sync



Flush Bulk Transport Protocol

Programming Sensor Nodes

- Tutorials to familiarize you with tools, software
  - *csl.stanford.edu/~pal/pubs/tos-**programming**-web.pdf*
  - *csl.stanford.edu/~pal/pubs/**tinyos-programming**.pdf*
- Should be fun
  - Monitor your own sleeping patterns
  - Measure you personal energy consumption
  - Visualize real-world social networks
- Should be instructive
  - Learn to use the TinyOS toolchain
  - Program sensor nodes in C and nesC
  - Deploy and IPv6 low-power wireless network

Research Project

- Goal of this course is to do graduate research
  - Work in group of 3, or alone
  - Pick a research problem of your own
  - or
  - Meet with instructor to discuss other ideas

- Ideally, the project dovetails with thesis topic

- Should be related to one of the themes
  - Technologies
  - Systems
  - Applications

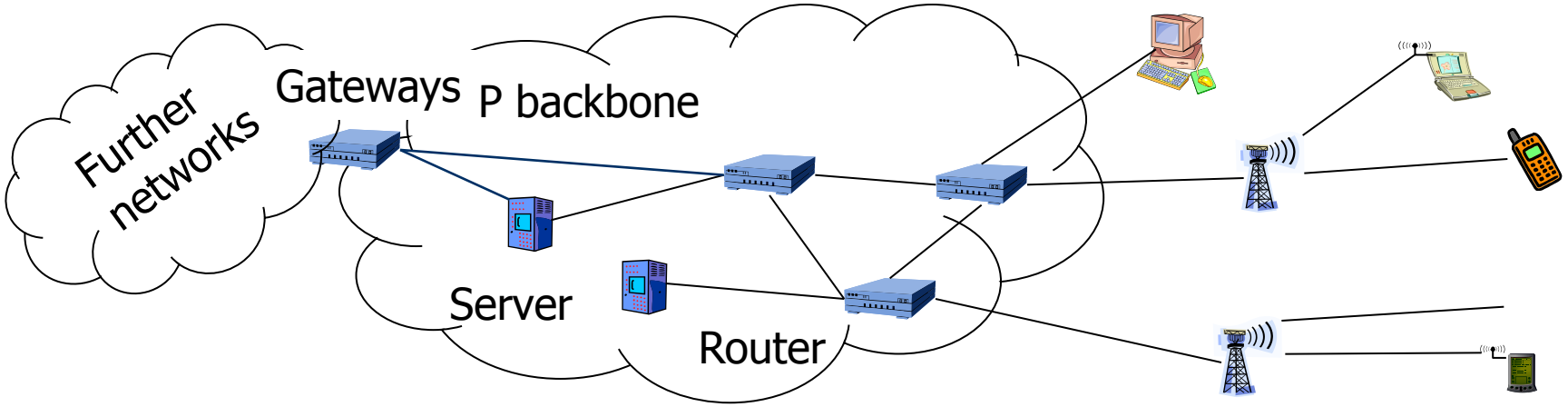
WSN INTRO

Outline

- ***Infrastructure for wireless?***
- (Mobile) ad hoc networks
- Wireless sensor networks
- Comparison

Infrastructure-based wireless networks

- Typical wireless network: Based on infrastructure
  - E.g., GSM, UMTS, …
  - Base stations connected to a wired backbone network
  - Mobile entities communicate wirelessly to these base stations
  - Traffic between different mobile entities is relayed by base stations and wired backbone
  - Mobility is supported by switching from one base station to another
  - Backbone infrastructure required for administrative tasks

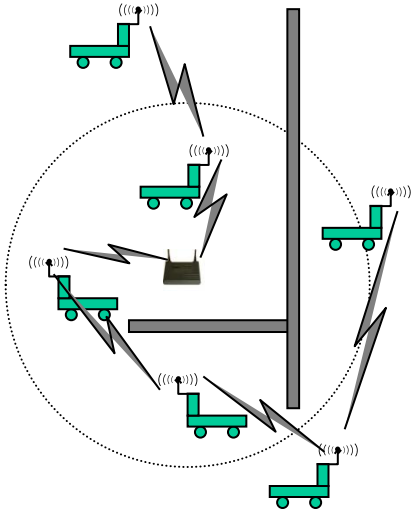


Infrastructure-based wireless networks - Limits?

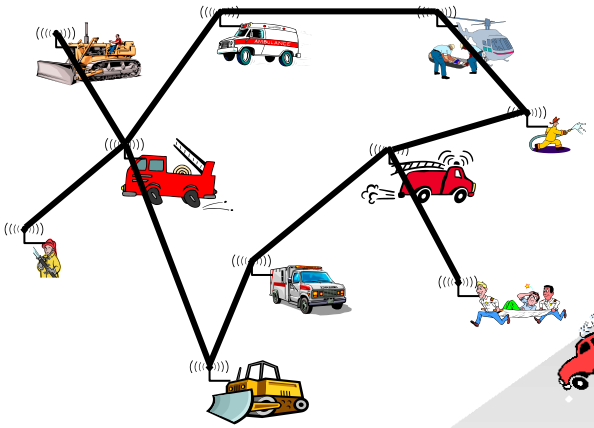
- What if …
  - No infrastructure is available? – E.g., in disaster areas
  - It is too expensive/inconvenient to set up? – E.g., in remote, large construction sites
  - There is no time to set it up? – E.g., in military operations

Possible applications for infrastructure-free networks

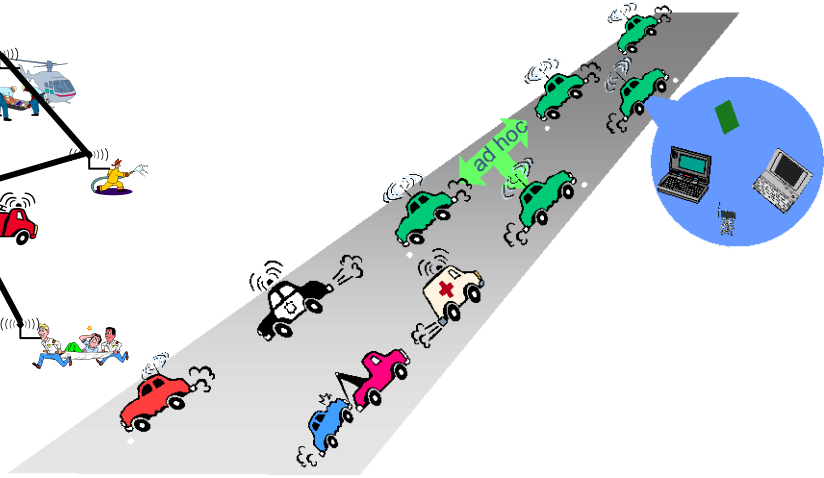
- Factory floor automation



- Disaster recovery



- Car-to-car communication



- Military networking: Tanks, soldiers, …
- Finding out empty parking lots in a city, without asking a server
- Search-and-rescue in an avalanche
- Personal area networking (watch, glasses, PDA, medical appliance, …)
- …

Outline

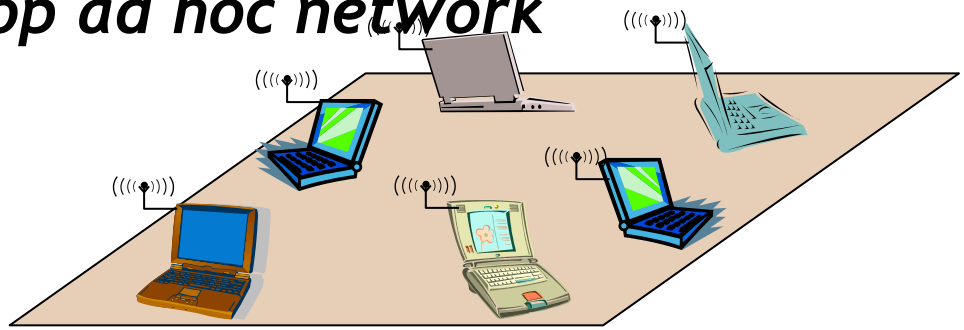
- Infrastructure for wireless?
- ***(Mobile) ad hoc networks***
- Wireless sensor networks
- Comparison

Solution: (Wireless) ad hoc networks

- Try to construct a network without infrastructure, using networking abilities of the participants
  - This is an ***ad hoc network*** – a network constructed "for a special purpose"

- Simplest example: Laptops in a conference room –

  a ***single-hop ad hoc network***



Problems/challenges for ad hoc networks

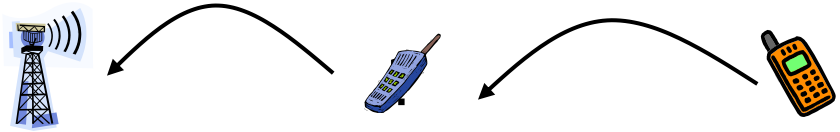
- Without a central infrastructure, things become much more difficult
- Problems are due to
  - Lack of central entity for organization available
  - Limited range of wireless communication
  - Mobility of participants
  - Battery-operated entities

No central entity -> self-organization

- Without a central entity (like a base station), participants must organize themselves into a network (***self-organization***)
- Pertains to (among others):
  - Medium access control – no base station can assign transmission resources, must be decided in a distributed fashion
  - Finding a route from one participant to another

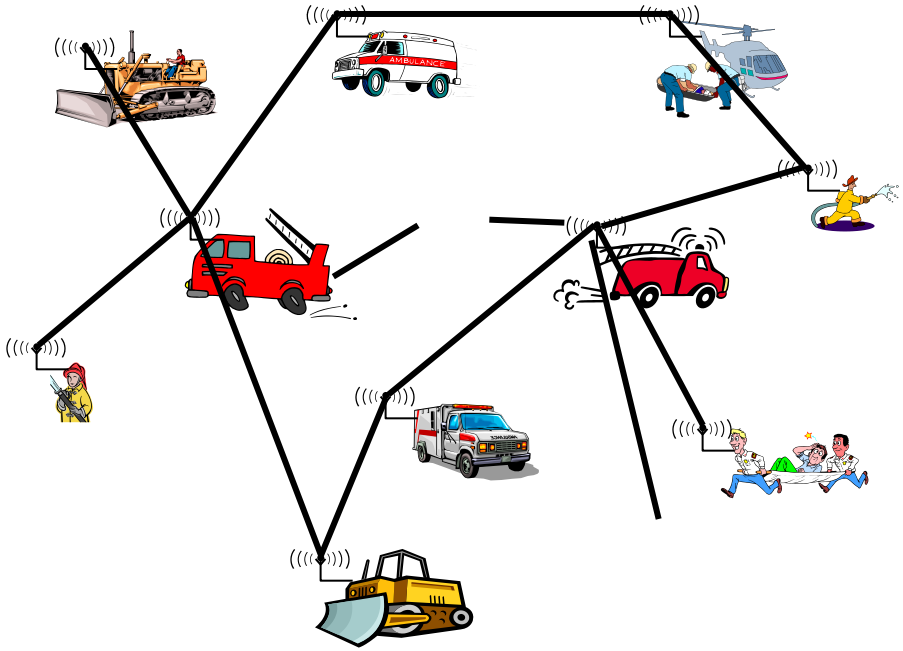
Limited range -> multi-hopping

- For many scenarios, communication with peers outside immediate communication range is required
  - Direct communication limited because of distance, obstacles, …
  - Solution: ***multi-hop network***



Mobility -> Suitable, adaptive protocols

- In many (not all!) ad hoc network applications, participants move around
  - In cellular network: simply hand over to another base station
- In ***mobile ad hoc networks (MANET)***:
  - Mobility changes neighborhood relationship
  - Must be compensated for
  - E.g., routes in the network have to be changed
- Complicated by scale
  - Large number of such nodes difficult to support



Battery-operated devices -> energy-efficient operation

- Often (not always!), participants in an ad hoc network draw energy from batteries
- Desirable: long run time for
  - Individual devices
  - Network as a whole

! Energy-efficient networking protocols

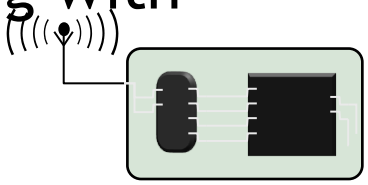
- E.g., use multi-hop routes with low energy consumption (energy/bit)
- E.g., take available battery capacity of devices into account
- How to resolve conflicts between different optimizations?

Outline

- Infrastructure for wireless?
- (Mobile) ad hoc networks
- ***Wireless sensor networks***
  - ***Applications***
  - Requirements & mechanisms
- Comparison

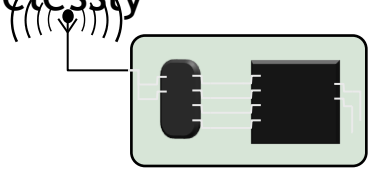
Wireless sensor networks

- Participants in the previous examples were devices close to a human user, interacting with humans



- Alternative concept:

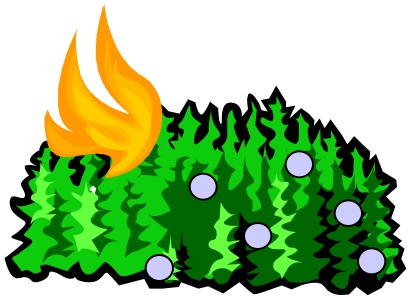
  Instead of focusing interaction on humans, focus on interacting with ***environment***
  - Network is ***embedded*** in environment
  - Nodes in the network are equipped with ***sensing*** and ***actuation*** to measure/influence environment
  - Nodes process information and communicate it wirelessly

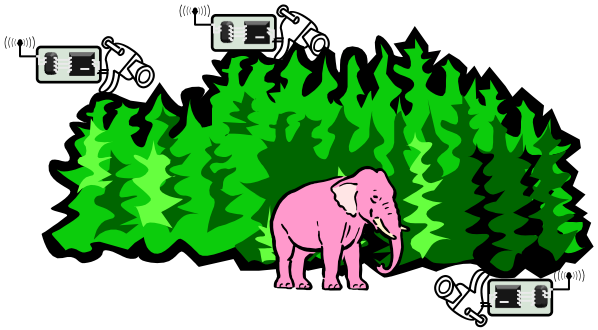


! ***Wireless sensor networks*** (WSN)
  - Or: ***Wireless sensor & actuator networks*** (WSAN)

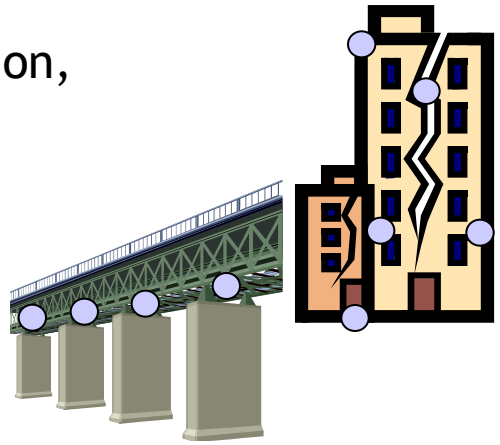
WSN application examples

- Disaster relief operations
  - Drop sensor nodes from an aircraft over a wildfire
  - Each node measures temperature
  - Derive a “temperature map”
- Biodiversity mapping
  - Use sensor nodes to observe wildlife



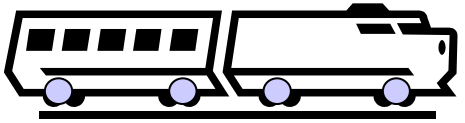


- Intelligent buildings (or bridges)
  - Reduce energy wastage by proper humidity, ventilation, air conditioning (HVAC) control
  - Needs measurements about room occupancy, temperature, air flow, …
  - Monitor mechanical stress after earthquakes



WSN application scenarios

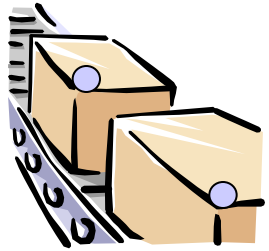
- Facility management
  - Intrusion detection into industrial sites
  - Control of leakages in chemical plants, …
- Machine surveillance and preventive maintenance
  - Embed sensing/control functions into places no cable has gone before
  - E.g., tire pressure monitoring



- Precision agriculture
  - Bring out fertilizer/pesticides/irrigation only where needed
- Medicine and health care
  - Post-operative or intensive care
  - Long-term surveillance of chronically ill patients or the elderly
- Environmental monitoring
  - Volcano monitoring

WSN application scenarios

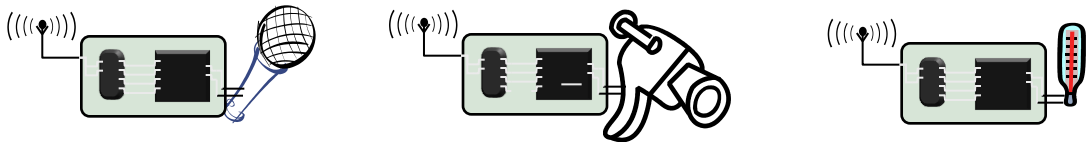
- Logistics
  - Equip goods (parcels, containers) with a sensor node
  - Track their whereabouts – ***total asset management***
  - Note: passive readout might suffice – compare RF IDs



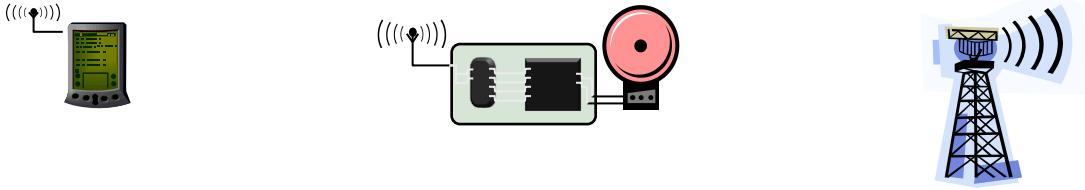
- Telematics
  - Provide better traffic control by obtaining finer-grained information about traffic conditions
  - ***Intelligent roadside***
  - Cars as the sensor nodes

Roles of participants in WSN

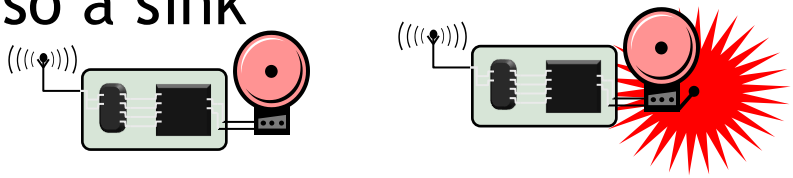
- ***Sources*** of data: Measure data, report them "somewhere"
  - Typically equip with different kinds of actual sensors



- ***Sinks*** of data: Interested in receiving data from WSN
  - May be part of the WSN or external entity, PDA, gateway, …



- ***Actuators***: Control some device based on data, usually also a sink



Structuring WSN application types

- ***Interaction patterns*** between sources and sinks classify application types
  - ***Event detection***: Nodes locally detect events (maybe jointly with nearby neighbors), report these events to interested sinks
    - ***Event classification*** additional option
  - ***Periodic measurement***
  - ***Function approximation:*** Use sensor network to approximate a function of space and/or time (e.g., temperature map)
  - ***Edge detection:*** Find edges (or other structures) in such a function (e.g., where is the zero degree border line?)
  - ***Tracking*****:** Report (or at least, know) position of an observed intruder (“pink elephant”)

Deployment options for WSN

- How are sensor nodes deployed in their environment?
  - Dropped from aircraft ! ***Random deployment***
    - Usually uniform random distribution for nodes over finite area is assumed
    - Is that a likely proposition?
  - Well planned, fixed ! ***Regular deployment***
    - E.g., in preventive maintenance or similar
    - Not necessarily geometric structure, but that is often a convenient assumption
  - ***Mobile*** sensor nodes
    - Can move to compensate for deployment shortcomings
    - Can be passively moved around by some external force (wind, water)
    - Can actively seek out “interesting” areas

Maintenance options

- Feasible and/or practical to maintain sensor nodes?
  - E.g., to replace batteries?
  - Or: unattended operation?
  - Impossible but not relevant? Mission lifetime might be very small
- Energy supply?
  - Limited from point of deployment?
  - Some form of recharging, energy scavenging from environment?
    - E.g., solar cells

Outline

- Infrastructure for wireless?
- (Mobile) ad hoc networks
- ***Wireless sensor networks***
  - Applications
  - ***Requirements & mechanisms***
- Comparison

Characteristic requirements for WSNs

- Type of service of WSN
  - Not simply moving bits like another network
  - Rather: provide ***answers*** (not just numbers)
  - Issues like geographic scoping are natural requirements, absent from other networks
- Quality of service
  - Traditional QoS metrics do not apply
  - Still, service of WSN must be “good”: Right answers at the right time
- Fault tolerance
  - Be robust against node failure (running out of energy, physical destruction, …)
- Lifetime
  - The ***network*** should fulfill its task as long as possible – definition depends on application
  - Lifetime of individual nodes relatively unimportant
  - But often treated equivalently

Characteristic requirements for WSNs

- Scalability
  - Support large number of nodes
- Wide range of densities
  - Vast or small number of nodes per unit area, very application-dependent
- Programmability
  - Re-programming of nodes in the field might be necessary, improve flexibility
- Maintainability
  - WSN has to adapt to changes, self-monitoring, adapt operation
  - Incorporate possible additional resources, e.g., newly deployed nodes

Required mechanisms to meet requirements

- Multi-hop wireless communication
- Energy-efficient operation
  - Both for communication and computation, sensing, actuating
- Auto-configuration
  - Manual configuration just not an option
- Collaboration & in-network processing
  - Nodes in the network collaborate towards a joint goal
  - Pre-processing data in network (as opposed to at the edge) can greatly improve efficiency

Required mechanisms to meet requirements

- Data centric networking
  - Focusing network design on ***data***, not on ***node identifies*** (id-centric networking)
  - To improve efficiency
- Locality
  - Do things locally (on node or among nearby neighbors) as far as possible
- Exploit tradeoffs
  - E.g., between invested energy and accuracy

Outline

- Infrastructure for wireless?
- (Mobile) ad hoc networks
- Wireless sensor networks
- ***Comparison***

MANET vs. WSN

- Many commonalities: Self-organization, energy efficiency, (often) wireless multi-hop
- Many differences
  - ***Applications, equipment***: MANETs more powerful (read: expensive) equipment assumed, often "human in the loop"-type applications, higher data rates, more resources
  - ***Application-specific***: WSNs depend much stronger on application specifics; MANETs comparably uniform
  - ***Environment interaction***: core of WSN, absent in MANET
  - ***Scale***: WSN might be much larger (although contestable)
  - ***Energy***: WSN tighter requirements, maintenance issues
  - ***Dependability/QoS***: in WSN, individual node may be dispensable (network matters), QoS different because of different applications
  - ***Data centric*** vs. id-centric networking
  - ***Mobility***: different mobility patterns like (in WSN, sinks might be mobile, usual nodes static)

Wireless fieldbuses and WSNs

- Fieldbus:
  - Network type invented for real-time communication, e.g., for factory-floor automation
  - Inherent notion of sensing/measuring and controlling
  - Wireless fieldbus: Real-time communication over wireless

! Big similarities

- Differences
  - Scale – WSN often intended for larger scale
  - Real-time – WSN usually not intended to provide (hard) real-time guarantees as attempted by fieldbuses

Enabling technologies for WSN

- Cost reduction
  - For wireless communication, simple microcontroller, sensing, batteries
- Miniaturization
  - Some applications demand small size
  - “Smart dust” as the most extreme vision
- Energy scavenging
  - Recharge batteries from ambient energy (light, vibration, …)

Conclusion

- MANETs and WSNs are challenging and promising system concepts
- Many similarities, many differences
- Both require new types of architectures & protocols compared to “traditional” wired/wireless networks
- In particular, application-specificness is a new issue

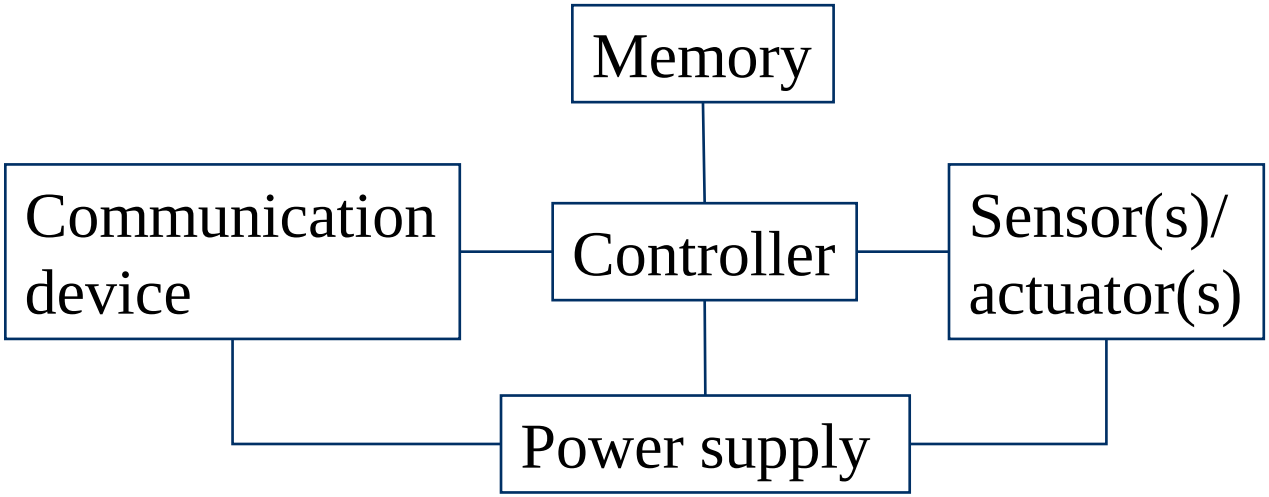
WSN NODES

Outline

- ***Sensor node architecture***
- Energy supply and consumption
- Runtime environments for sensor nodes
- Case study: TinyOS

Sensor node architecture

- Main components of a WSN node
  - Controller
  - Communication device(s)
  - Sensors/actuators
  - Memory
  - Power supply



Ad hoc node architecture

- Core: essentially the same
- But: Much more additional equipment
  - Hard disk, display, keyboard, voice interface, camera, …
- Essentially: a laptop-class device

Controller

- Main options:
  - Microcontroller – general purpose processor, optimized for embedded applications, low power consumption
  - DSPs – optimized for signal processing tasks, not suitable here
  - FPGAs – may be good for testing
  - ASICs – only when peak performance is needed, no flexibility

- Example microcontrollers
  - Texas Instruments MSP430
    - 16-bit RISC core, up to 8MHz, versions with 2-10 kbytes RAM,
      several DACs, RT clock, prices start at 0.49 US$
  - Atmel ATMega
    - 8-bit controller, larger memory than MSP430, slower

Communication device

- Which transmission medium?
  - Electromagnetic at radio frequencies? ✓
  - Electromagnetic, light?
  - Ultrasound?

- Radio transceivers transmit a bit- or byte stream as radio wave
  - Receive it, convert it back into bit-/byte stream

Transceiver characteristics

- Capabilities
  - Interface: bit, byte, packet level?
  - Supported frequency range?
    - Typically, somewhere in 433 MHz – 2.4 GHz, ISM band
  - Multiple channels?
  - Data rates?
  - Range?
- Energy characteristics
  - Power consumption to send/receive data?
  - Time and energy consumption to change between different states?
  - Transmission power control?
  - Power efficiency (which percentage of consumed power is radiated?)
- Radio performance
  - Modulation? (ASK, FSK, …?)
  - Noise figure? NF = SNR_I/SNR_O
  - Gain? (signal amplification)
  - Receiver sensitivity? (minimum S to achieve a given E_b/N_0)
  - Blocking performance (achieved BER in presence of frequency-offset interferer)
  - Out of band emissions
  - Carrier sensing & RSSI characteristics
  - Frequency stability (e.g., towards temperature changes)
  - Voltage range

Transceiver states

- Transceivers can be put into different operational ***states***, typically:
  - ***Transmit***
  - ***Receive***
  - ***Idle*** – ready to receive, but not doing so
    - Some functions in hardware can be switched off, reducing energy consumption a little
  - ***Sleep*** – significant parts of the transceiver are switched off
    - Not able to immediately receive something
    - ***Recovery time*** and ***startup energy*** to leave sleep state can be significant

- Research issue: Wakeup receivers – can be woken via radio when in sleep state (seeming contradiction!)

Example radio transceivers

- Almost boundless variety available
- Some examples
  - RFM TR1000 family
    - 916 or 868 MHz
    - 400 kHz bandwidth
    - Up to 115.2 kbps
    - On/off keying or ASK
    - Dynamically tuneable output power
    - Maximum power about 1.4 mW
    - Low power consumption
  - Chipcon CC1000
    - Range 300 to 1000 MHz, programmable in 250 Hz steps
    - FSK modulation
    - Provides RSSI
  - Chipcon CC 2400
    - Implements 802.15.4
    - 2.4 GHz, DSSS modem
    - 250 kbps
    - Higher power consumption than above transceivers
  - Infineon TDA 525x family
    - E.g., 5250: 868 MHz
    - ASK or FSK modulation
    - RSSI, highly efficient power amplifier
    - Intelligent power down, “self-polling” mechanism
    - Excellent blocking performance

Example radio transceivers for ad hoc networks

- Ad hoc networks: Usually, higher data rates are required
- Typical: IEEE 802.11 b/g/a is considered
  - Up to 54 MBit/s
  - Relatively long distance (100s of meters possible, typical 10s of meters at higher data rates)
  - Works reasonably well (but certainly not perfect) in mobile environments
  - Problem: expensive equipment, quite power hungry

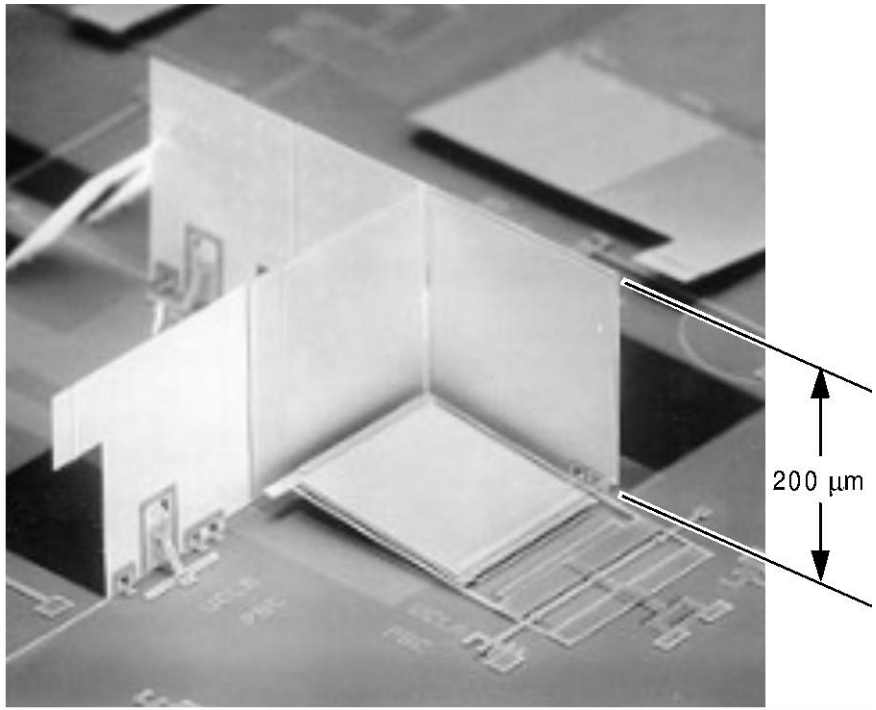
Wakeup receivers

- Major energy problem: ***RECEIVING***
  - Idling and being ready to receive consumes considerable amounts of power
- When to switch on a receiver is not clear
  - Contention-based MAC protocols: Receiver is always on
  - TDMA-based MAC protocols: Synchronization overhead, inflexible
- Desirable: Receiver that can (only) check for incoming messages
  - When signal detected, wake up main receiver for actual reception
  - Ideally: ***Wakeup receiver*** can already process simple addresses
  - Not clear whether they can be actually built, however

Optical communication

- Optical communication can consume less energy
- Example: passive readout via corner cube reflector
  - Laser is reflected back directly to source if mirrors are at right angles
  - Mirrors can be “titled” to stop reflecting

! Allows data to be sent back to laser source



Ultra-wideband communication

- Standard radio transceivers: Modulate a signal onto a carrier wave
  - Requires relatively small amount of bandwidth
- Alternative approach: Use a large bandwidth, do not modulate, simply emit a “burst” of power
  - Forms almost rectangular pulses
  - Pulses are very short
  - Information is encoded in the presence/absence of pulses
  - Requires tight time synchronization of receiver
  - Relatively short range (typically)
- Advantages
  - Pretty resilient to multi-path propagation
  - Very good ranging capabilities
  - Good wall penetration

Sensors as such

- Main categories
  - Any energy radiated? Passive vs. active sensors
  - Sense of direction? Omidirectional?

  - Passive, omnidirectional
    - Examples: light, thermometer, microphones, hygrometer, …
  - Passive, narrow-beam
    - Example: Camera
  - Active sensors
    - Example: Radar

- Important parameter: Area of coverage
  - Which region is adequately covered by a given sensor?

Outline

- Sensor node architecture
- ***Energy supply and consumption***
- Runtime environments for sensor nodes
- Case study: TinyOS

Energy supply of mobile/sensor nodes

- Goal: provide as much energy as possible at smallest cost/volume/weight/recharge time/longevity
  - In WSN, recharging may or may not be an option
- Options
  - Primary batteries – not rechargeable
  - Secondary batteries – rechargeable, only makes sense in combination with some form of energy harvesting
- Requirements include
  - Low self-discharge
  - Long shelf live
  - Capacity under load
  - Efficient recharging at low current
  - Good relaxation properties (seeming self-recharging)
  - Voltage stability (to avoid DC-DC conversion)

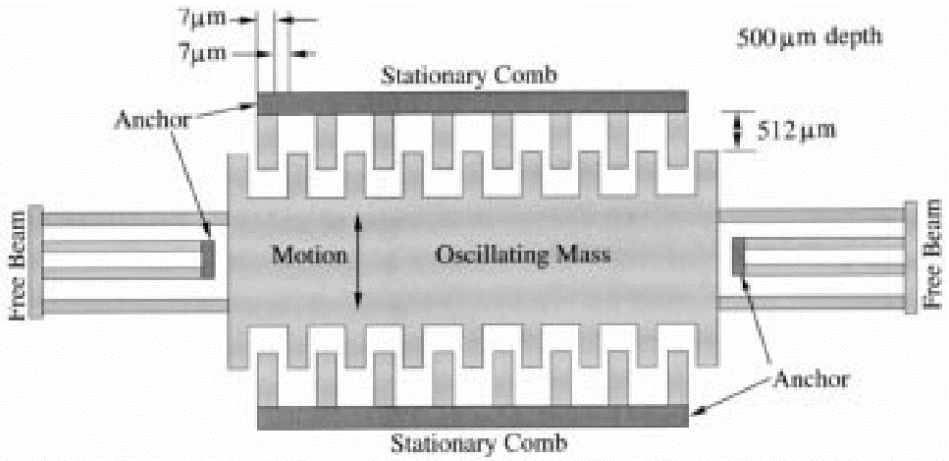
Battery examples

- Energy per volume (Joule per cubic centimeter):

Primary batteries			
Chemistry	Zinc-air	Lithium	Alkaline
Energy (J/cm^3)	3780	2880	1200
Secondary batteries			
Chemistry	Lithium	NiMHd	NiCd
Energy (J/cm^3)	1080	860	650

Energy scavenging

- How to recharge a battery?
  - A laptop: easy, plug into wall socket in the evening
  - A sensor node? – Try to ***scavenge*** energy from environment
- Ambient energy sources
  - Light: solar cells – between 10 $\mu W/cm^2$ and 15 mW/cm^2
  - Temperature gradients – 80 μ W/cm^2 @ 1 V from 5K difference
  - Vibrations – between 0.1 and 10000 μ W/cm^3
  - Pressure variation (piezo-electric) – 330 μ W/cm^2 from the heel of a shoe
  - Air/liquid flow
    (MEMS gas turbines)



Energy scavenging - overview

Energy source	Energy density
Batteries (zinc-air)	$1050 - 1560\,\mathrm{mWh/cm^3}$
Batteries (rechargable lithium)	$300\,\mathrm{mWh/cm^3}$ (at $3 - 4\,\mathrm{V}$)
Energy source	**Power density**
Solar (outdoors)	$15\,\mathrm{mW/cm^2}$ (direct sun)
	$0.15\,\mathrm{mW/cm^2}$ (cloudy day)
Solar (indoors)	$0.006\,\mathrm{mW/cm^2}$ (standard office desk)
	$0.57\,\mathrm{mW/cm^2}$ (< 60 W desk lamp)
Vibrations	$0.01 - 0.1\,\mathrm{mW/cm^3}$
Acoustic noise	$3 \cdot 10^{-6}\mathrm{mW/cm^2}$ at 75 Db
	$9,6 \cdot 10^{-4}\mathrm{mW/cm^2}$ at 100 Db
Passive human-powered systems	1.8 mW (shoe inserts)
Nuclear reaction	$80\,\mathrm{mW/cm^3}$, $10^6\,\mathrm{mWh/cm^3}$

Energy consumption

- A “back of the envelope” estimation
- Number of instructions
  - Energy per instruction: 1 nJ
  - Small battery (“smart dust”): 1 J = 1 Ws
  - Corresponds: 10^9 instructions!
- Lifetime
  - Or: Require a single day operational lifetime = 24*60*60 =86400 s
  - 1 Ws / 86400s = ***11.5 μW*** as max. sustained power consumption!
- Not feasible!

Multiple power consumption modes

- Way out: Do not run sensor node at full operation all the time
  - If nothing to do, switch to ***power safe mode***
  - Question: When to throttle down? How to wake up again?

- Typical modes
  - Controller: Active, idle, sleep
  - Radio mode: Turn on/off transmitter/receiver, both

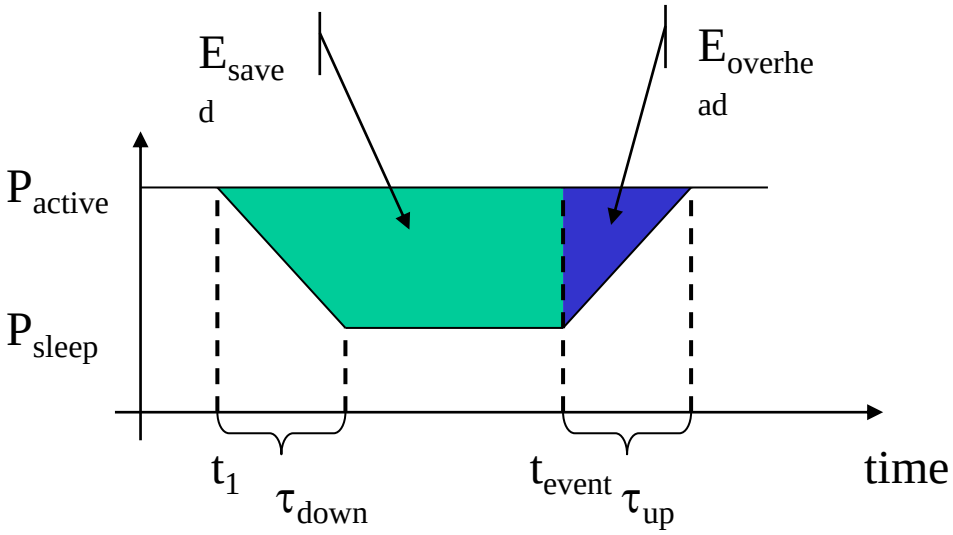
- Multiple modes possible, “deeper” sleep modes
  - Strongly depends on hardware
  - TI MSP 430, e.g.: four different sleep modes
  - Atmel ATMega: six different modes

Some energy consumption figures

- Microcontroller
  - TI MSP 430 (@ 1 MHz, 3V):
    - Fully operation 1.2 mW
    - Deepest sleep mode 0.3 μW – only woken up by external interrupts (not even timer is running any more)
  - Atmel ATMega
    - Operational mode: 15 mW active, 6 mW idle
    - Sleep mode: 75 μW

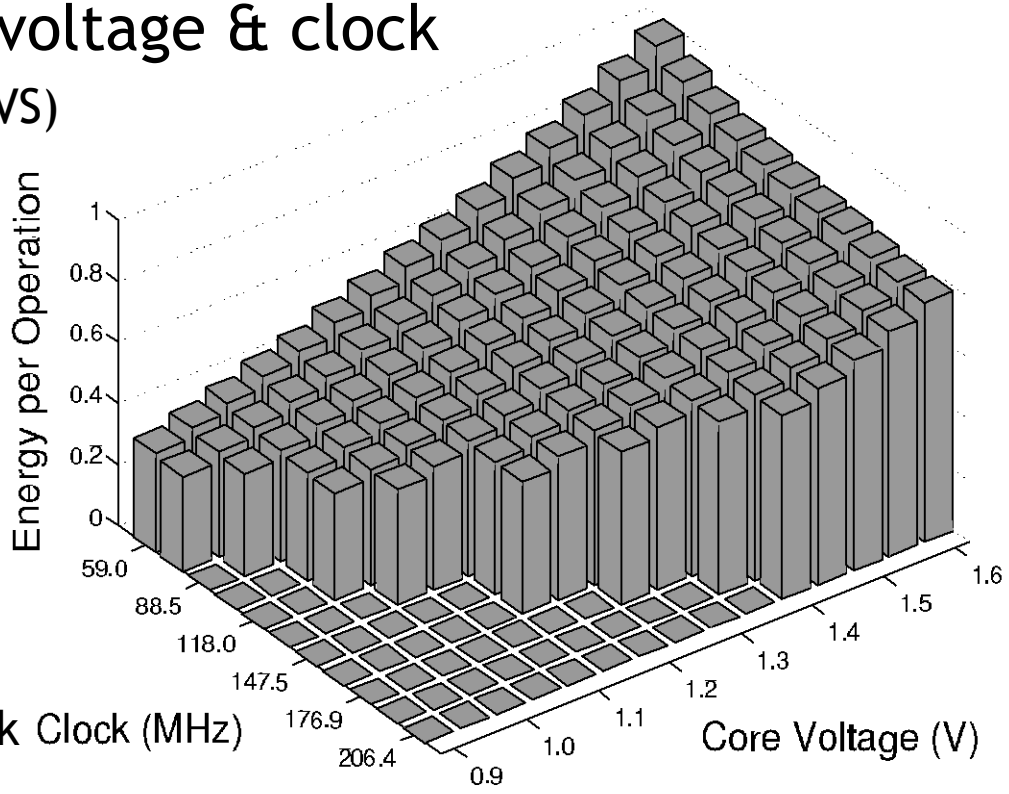
Switching between modes

- Simplest idea: Greedily switch to lower mode whenever possible
- Problem: Time and power consumption required to reach higher modes not negligible
  - Introduces overhead
  - Switching only pays off if $E_{saved} > E_{overhead}$
- Example: Event-triggered wake up from sleep mode
- Scheduling problem with uncertainty (exercise)



Alternative: Dynamic voltage scaling

- Switching modes complicated by uncertainty how long a sleep time is available
- Alternative: Low supply voltage & clock
  - ***Dynamic voltage scaling*** (DVS)
- Rationale:
  - Power consumption P depends on
    - Clock frequency
    - Square of supply voltage
    - $P / f V^2$
  - Lower clock allows lower supply voltage
  - Easy to switch to higher clock
  - But: execution takes longer



Memory power consumption

- Crucial part: FLASH memory
  - Power for RAM almost negligible
- FLASH writing/erasing is expensive
  - Example: FLASH on Mica motes
  - Reading: 1.1 nAh per byte
  - Writing: 83.3 nAh per byte

Transmitter power/energy consumption for n bits

- Amplifier power: $P_{amp} = \alpha_{amp} + \beta_{amp} P_{tx}$
  - P_{tx} ***radiated power***
  - α_{amp}, β_{amp} constants depending on model
  - Highest efficiency ($\eta = P_{tx} / P_{amp}$) at maximum output power
- In addition: transmitter electronics needs power P_{txElec}
- Time to transmit n bits: $n / (R \cdot R_{code})$
  - R nomial data rate, R_{code} coding rate
- To leave sleep mode
  - Time T_{start}, average power P_{start}

$$E_{tx} = T_{start} P_{start} + n / (R \cdot R_{code}) (P_{txElec} + \alpha_{amp} + \beta_{amp} P_{tx})$$

- Simplification: Modulation not considered

Receiver power/energy consumption for n bits

- Receiver also has startup costs
  - Time T_{start}, average power P_{start}
- Time for n bits is the same $n / (R ¢ R_{code})$
- Receiver electronics needs P_{rxElec}
- Plus: energy to decode n bits $E_{decBits}$

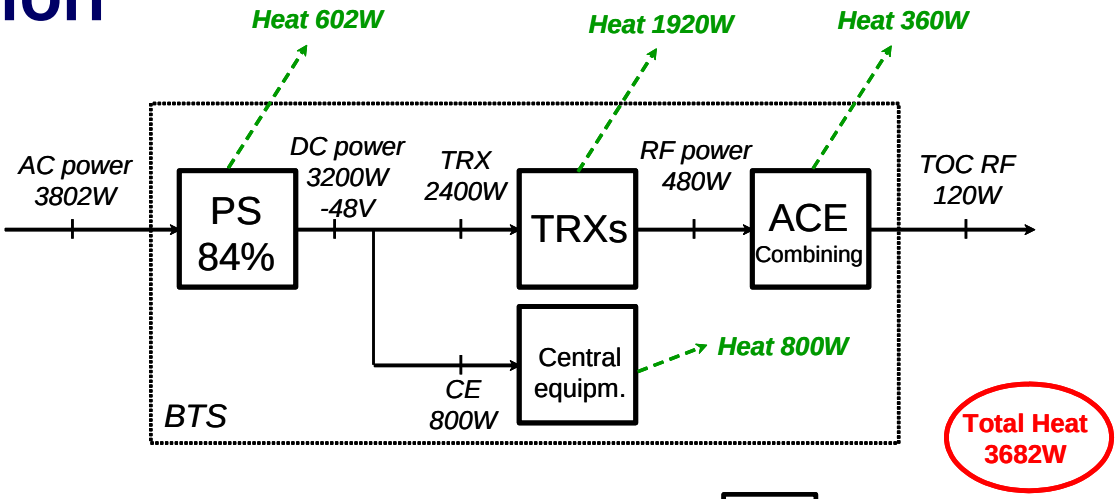
$$! E_{rx} = T_{start} P_{start} + n / (R ¢ R_{code}) P_{rxElec} + E_{decBits} (R)$$

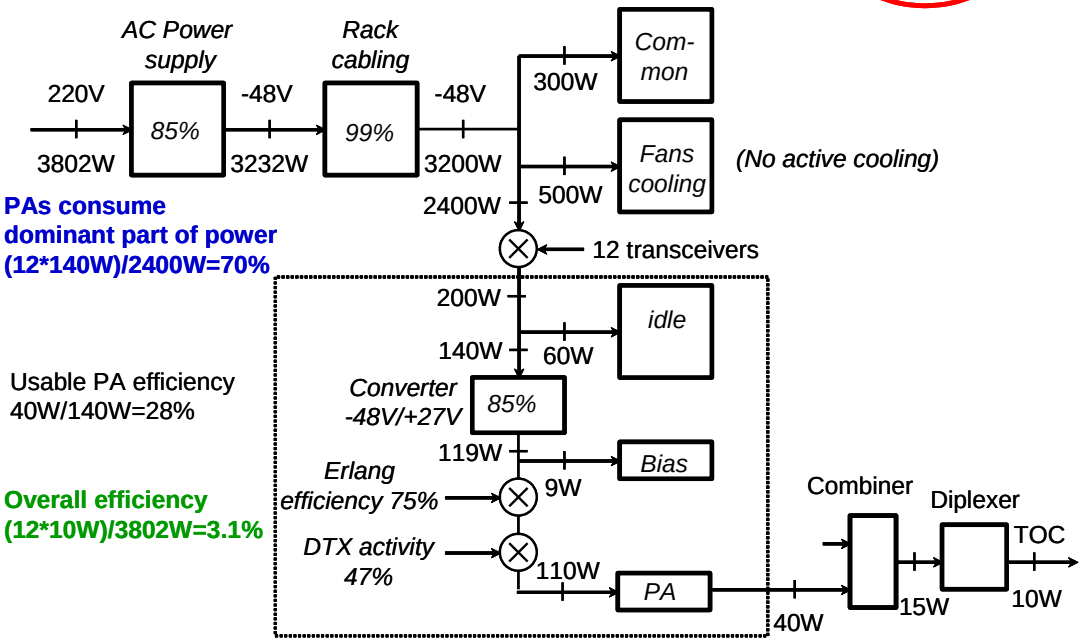
Some transceiver numbers

Symbol	Description	Example transceiver		
		μAMPS-1 [559]	WINS [670]	MEDUSA-II [670]
α_{amp}	Eq. (2.4)	174 mW	N/A	N/A
β_{amp}	Eq. (2.4)	5.0	8.9	7.43
P_{amp}	Amplifier pwr.	179 – 674 mW	N/A	N/A
P_{rxElec}	Reception pwr.	279 mW	368.3 mW	12.48 mW
P_{rxIdle}	Receive idle	N/A	344.2 mW	12.34 mW
P_{start}	Startup pwr.	58.7 mW	N/A	N/A
P_{txElec}	Transmit pwr.	151 mW	$\approx$ 386 mW	11.61 mW
R	Transmission rate	1 Mbps	100 kbps	OOK 30 kbps ASK 115.2 kbps
T_{start}	Startup time	466 μs	N/A	N/A

Comparison: GSM base station power consumption

- Overview
- Details
- (just to put things into perspective)





Controlling transceivers

- Similar to controller, low duty cycle is necessary
  - Easy to do for transmitter – similar problem to controller: when is it worthwhile to switch off
  - Difficult for receiver: Not only time when to wake up not known, it also depends on ***remote*** partners
  - Dependence between MAC protocols and power consumption is strong!
- Only limited applicability of techniques analogue to DVS
  - Dynamic Modulation Scaling (DSM): Switch to modulation best suited to communication – depends on channel gain
  - Dynamic Coding Scaling – vary coding rate according to channel gain
  - Combinations

Computation vs. communication energy cost

- Tradeoff?
  - Directly comparing computation/communication energy cost not possible
  - But: put them into perspective!
  - Energy ratio of “sending one bit” vs. “computing one instruction”: Anything between 220 and 2900 in the literature
  - To communicate (send & receive) one kilobyte = computing three million instructions!
- Hence: try to compute instead of communicate whenever possible
- Key technique in WSN – ***in-network processing!***
  - Exploit compression schemes, intelligent coding schemes, …

Outline

- Sensor node architecture
- Energy supply and consumption
- ***Runtime environments for sensor nodes***
- Case study: TinyOS

Operating system challenges in WSN

- Usual operating system goals
  - Make access to device resources abstract (virtualization)
  - Protect resources from concurrent access
- Usual means
  - Protected operation modes of the CPU
  - ----hardware access only in these modes
  - Process with separate address spaces
  - Support by a memory management unit
- Problem: These are not available in microcontrollers
  - No separate protection modes, no memory management unit
  - Would make devices more expensive, more power-hungry

Operating system challenges in WSN

- Possible options
  - Try to implement “as close to an operating system” on WSN nodes
    - In particular, try to provide a known programming interface
    - Namely: support for processes!
    - Sacrifice protection of different processes from each other

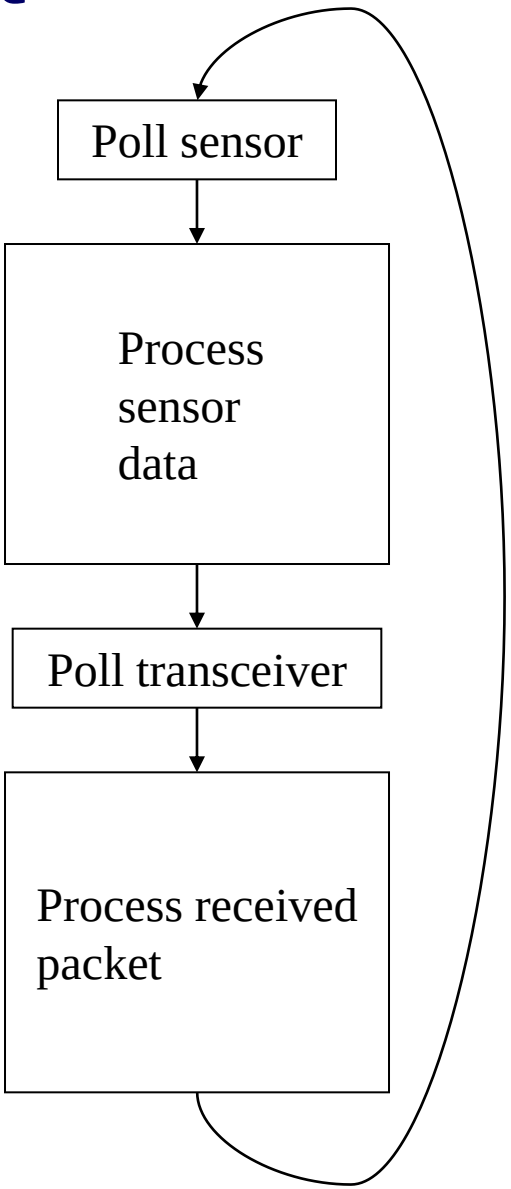
    ! Possible, but relatively high overhead
  - Do (more or less) away with operating system
    - After all, there is only a single “application” running on a WSN node
    - No need to protect malicious software parts from each other
    - Direct hardware control by application might improve efficiency
- Currently popular verdict: no OS, just a simple run-time environment
  - Enough to abstract away hardware access details
  - Biggest impact: Unusual programming model

Main issue: How to support concurrency

- Simplest option: No concurrency, sequential processing of tasks
  - Not satisfactory: Risk of missing data (e.g., from transceiver) when processing data, etc.

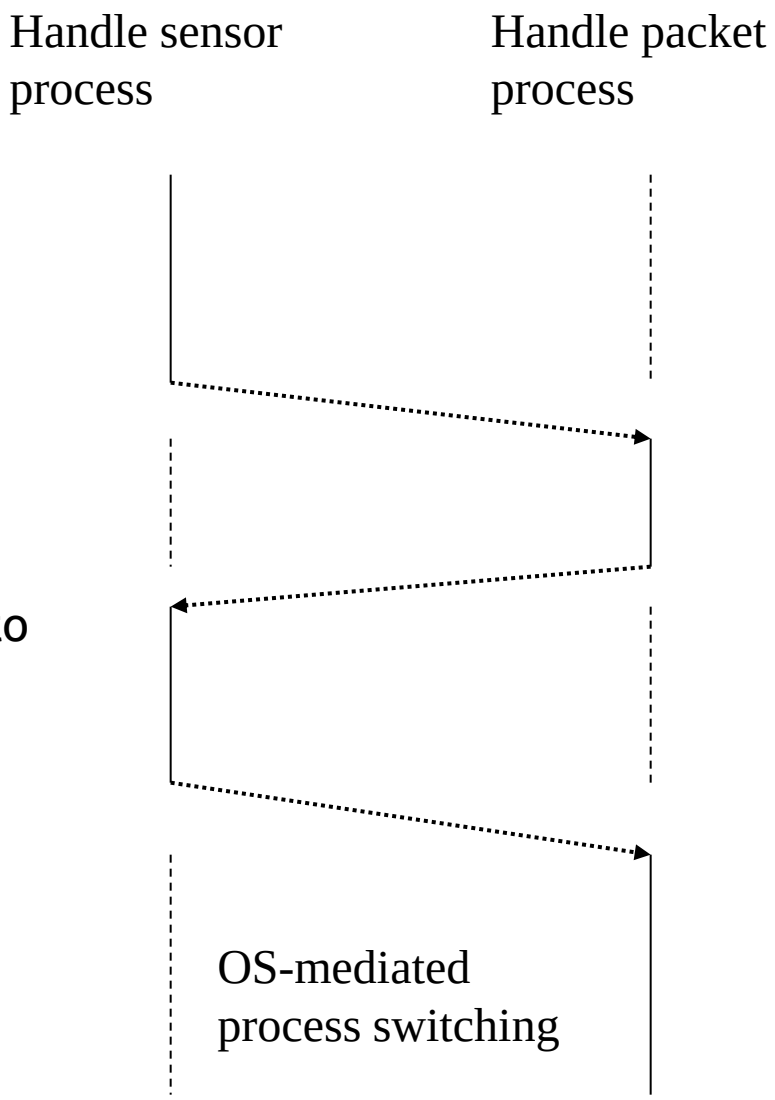
  ----Interrupts/asynchronous operation has to be supported

- Why concurrency is needed
  - Sensor node's CPU has to service the radio modem, the actual sensors, perform computation for application, execute communication protocol software, etc.



Traditional concurrency: Processes

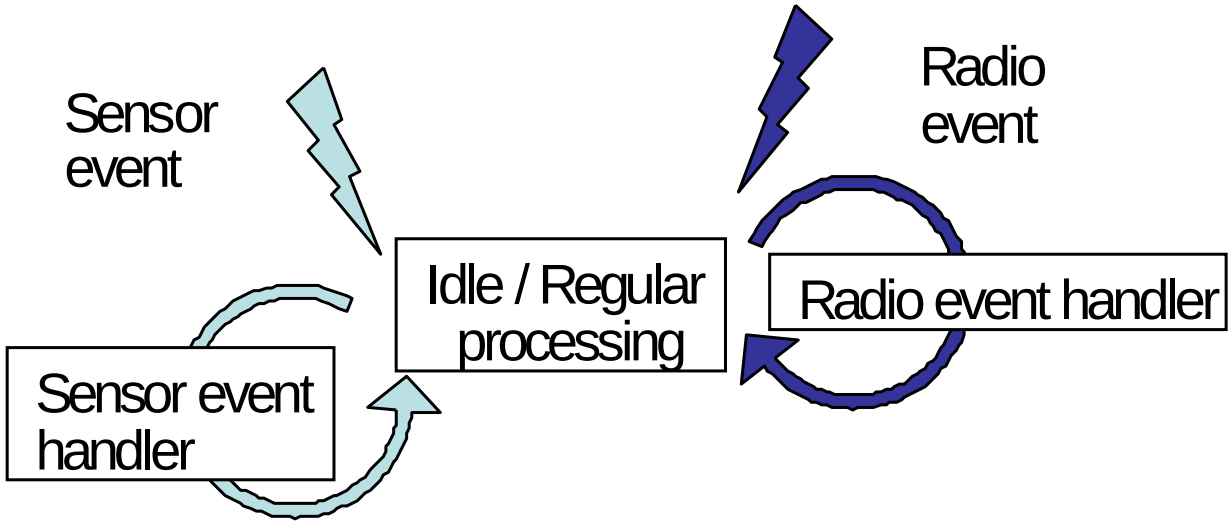
- Traditional OS: processes/threads
  - Based on interrupts, context switching
  - But: not available – memory overhead, execution overhead
- But: concurrency mismatch
  - One process per protocol entails too many context switches
  - Many tasks in WSN small with respect to context switching overhead
- And: protection between processes not needed in WSN
  - Only one application anyway



Event-based concurrency

- Alternative: Switch to ***event-based programming model***
  - Perform regular processing or be idle
  - React to events when they happen immediately
  - Basically: interrupt handler
- Problem: must not remain in interrupt handler too long
  - Danger of loosing events
  - Only save data, post information that event has happened, then return

  ! ***Run-to-completion*** principle
  - Two contexts: one for handlers, one for regular execution



Components instead of processes

- Need an abstraction to group functionality
  - Replacing “processes” for this purpose
  - E.g.: individual functions of a networking protocol
- One option: ***Components***
  - Here: In the sense of TinyOS
  - Typically fulfill only a single, well-defined function
  - Main difference to processes:
    - Component does not have an execution
    - Components access same address space, no protection against each other
  - NOT to be confused with component-based programming!

API to an event-based protocol stack

- Usual networking API: sockets
  - Issue: blocking calls to receive data
  - Ill-matched to event-based OS
  - Also: networking semantics in WSNs not necessarily well matched to/by socket semantics
- API is therefore also event-based
  - E.g.: Tell some component that some other component wants to be informed if and when data has arrived
  - Component will be posted an event once this condition is met
  - Details: see TinyOS example discussion below

Dynamic power management

- Exploiting multiple operation modes is promising
- Question: When to switch in power-safe mode?
  - Problem: Time & energy overhead associated with wakeup; greedy sleeping is not beneficial (see exercise)
  - Scheduling approach
- Question: How to control dynamic voltage scaling?
  - More aggressive; stepping up voltage/frequency is easier
  - Deadlines usually bound the required speed form below
- Or: Trading off fidelity vs. energy consumption!
  - If more energy is available, compute more accurate results
  - Example: Polynomial approximation
    - Start from high or low exponents depending where the polynomial is to be evaluated

Outline

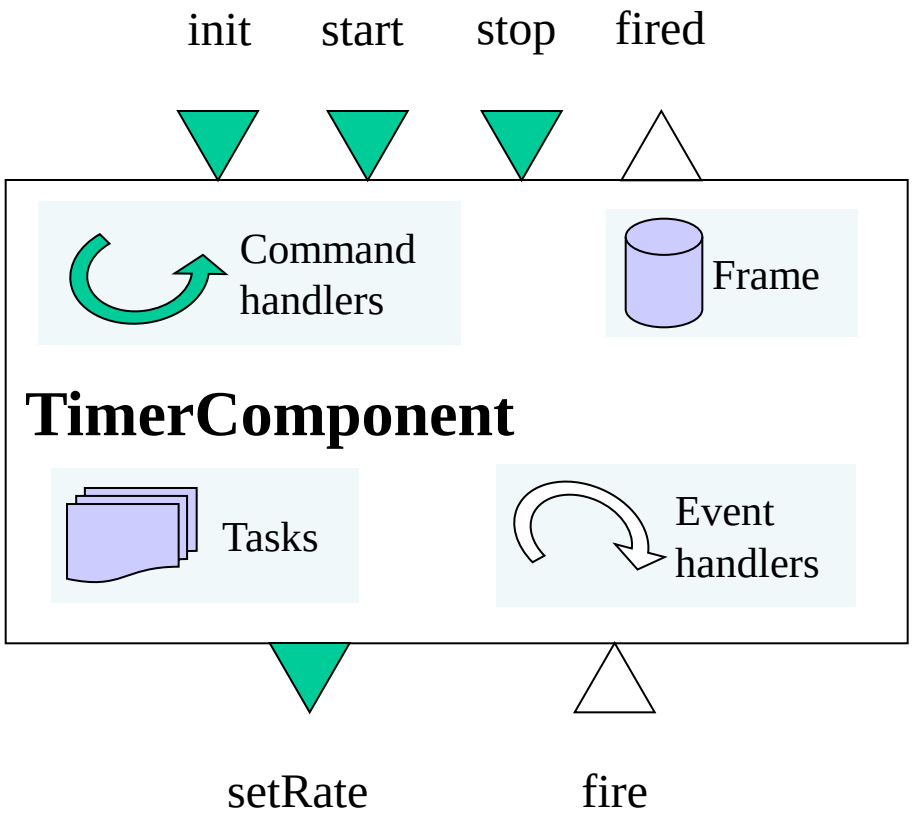
- Sensor node architecture
- Energy supply and consumption
- Runtime environments for sensor nodes
- ***Case study: TinyOS***

Case study embedded OS: TinyOS & nesC

- TinyOS developed by UC Berkely as runtime environment for their “motes”
- nesC as adjunct “programming language”
- Goal: Small memory footprint
  - Sacrifices made e.g. in ease of use, portability
  - Portability somewhat improved in newer version
- Most important design aspects
  - Component-based system
  - Components interact by exchanging asynchronous events
  - Components form a program by ***wiring*** them together (akin to VHDL – hardware description language)

TinyOS components

- Components
  - Frame – state information
  - Tasks – normal execution program
  - Command handlers
  - Event handlers
- Handlers
  - Must run to completion
  - Form a component's interface
  - Understand and emits commands & events
- Hierarchically arranged
  - Events pass upward from hardware to higher-level components
  - Commands are passed downward



Handlers versus tasks

- Command handlers and events must run to completion
  - Must not wait an indeterminate amount of time
  - Only a ***request*** to perform some action
- Tasks, on the other hand, can perform arbitrary, long computation
  - Also have to be run to completion since no non-cooperative multi-tasking is implemented
  - But can be interrupted by handlers

  ! No need for stack management, tasks are atomic with respect to each other

Split-phase programming

- Handler/task characteristics and separation has consequences on programming model
  - How to implement a blocking call to another component?
  - Example: Order another component to send a packet
  - Blocking function calls are not an option

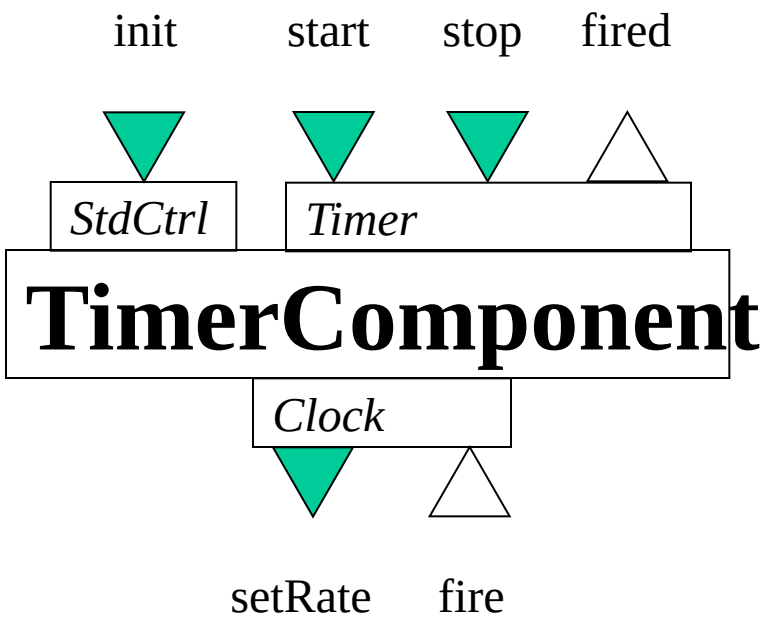
! Split-phase programming

  - First phase: Issue the command to another component
    - Receiving command handler will only receive the command, post it to a task for actual execution and returns immediately
    - Returning from a command invocation does not mean that the command has been executed!
  - Second phase: Invoked component notifies invoker by event that command has been executed
  - Consequences e.g. for buffer handling
    - Buffers can only be freed when completion event is received

Structuring commands/events into interfaces

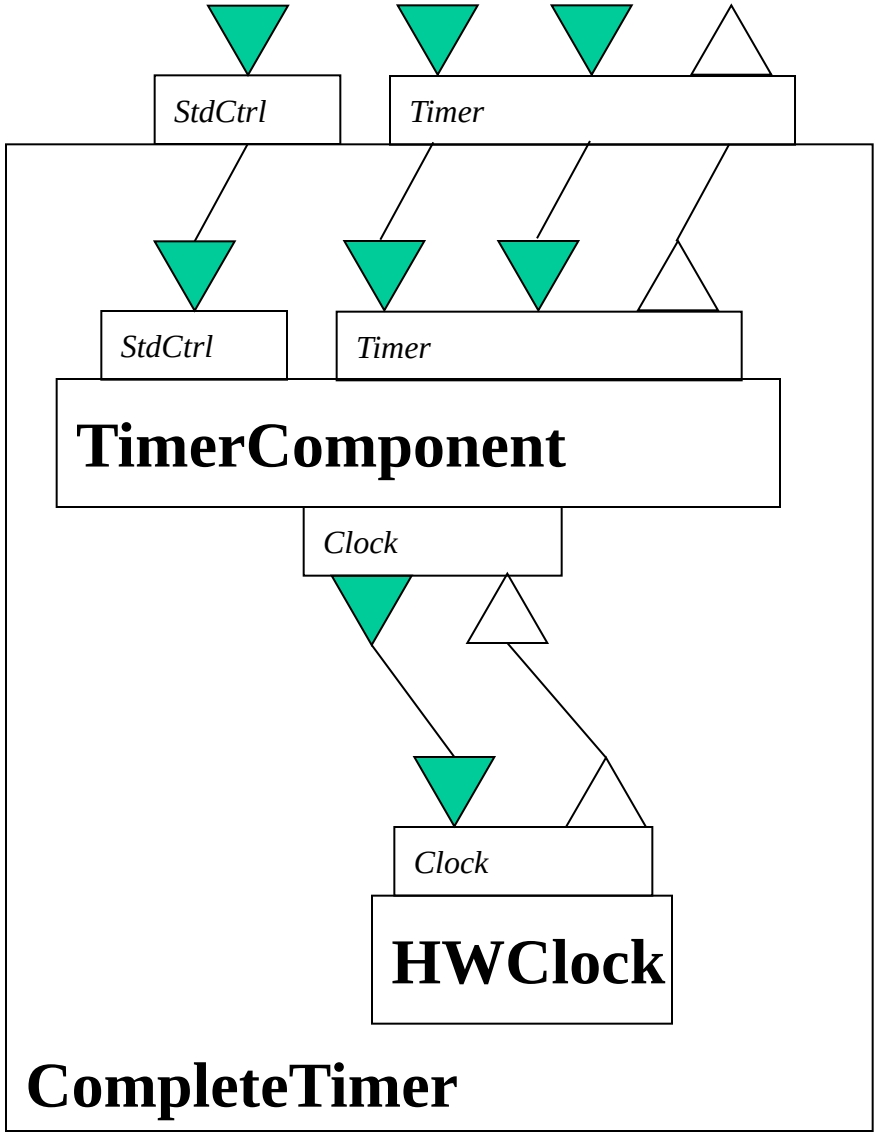
- Many commands/events can add up
- nesC solution: Structure corresponding commands/events into ***interface types***
- Example: Structure timer into three interfaces
  - StdCtrl
  - Timer
  - Clock

- Build configurations by wiring together corresponding interfaces



Building components out of simpler ones

- Wire together components to form more complex components out of simpler ones
- New interfaces for the complex component



Defining modules and components in nesC

```
interface StdCtrl {
  command result_t init();
}

interface Timer {
  command result_t start (char type, uint32_t interval);
  command result_t stop ();
  event result_t fired();
}

interface Clock {
  command result_t setRate (char interval, char scale);
  event result_t  fire ();
}

module TimerComponent {
  provides {
    interface StdCtrl;
    interface Timer;
  }
  uses interface Clock as Clk;
}
```

Wiring components to form a configuration

```
configuration CompleteTimer {
  provides {
    interface StdCtrl;
    interface Timer;
  }
  implementation {
    components TimerComponent, HWClock;
    StdCtrl = TimerComponent.StdCtrl ;
    Timer = TimerComponent.Timer;
    TimerComponent.Clk = HWClock.Clock;
  }
}
```

Summary

- For WSN, the need to build cheap, low-energy, (small) devices has various consequences for system design
  - Radio frontends and controllers are much simpler than in conventional mobile networks
  - Energy supply and scavenging are still (and for the foreseeable future) a premium resource
  - Power management (switching off or throttling down devices) crucial
- Unique programming challenges of embedded systems
  - Concurrency without support, protection
  - De facto standard: TinyOS

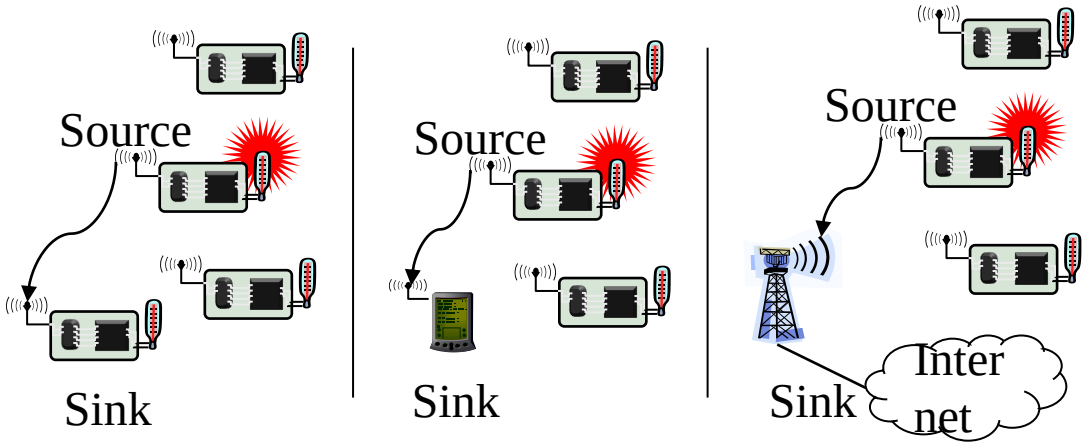
WSN ARCHITECTURE

Outline

- ***Network scenarios***
- Optimization goals
- Design principles
- Service interface
- Gateway concepts

Basic scenarios: sensor networks

- Sensor network scenarios
  - ***Sources***: Any entity that provides data/measurements
  - ***Sinks***: Nodes where information is required
    - Belongs to the sensor network as such
    - Is an external entity, e.g., a PDA, but directly connected to the WSN
      - Main difference: comes and goes, often moves around, …
    - Is part of an external network (e.g., internet), somehow connected to the WSN

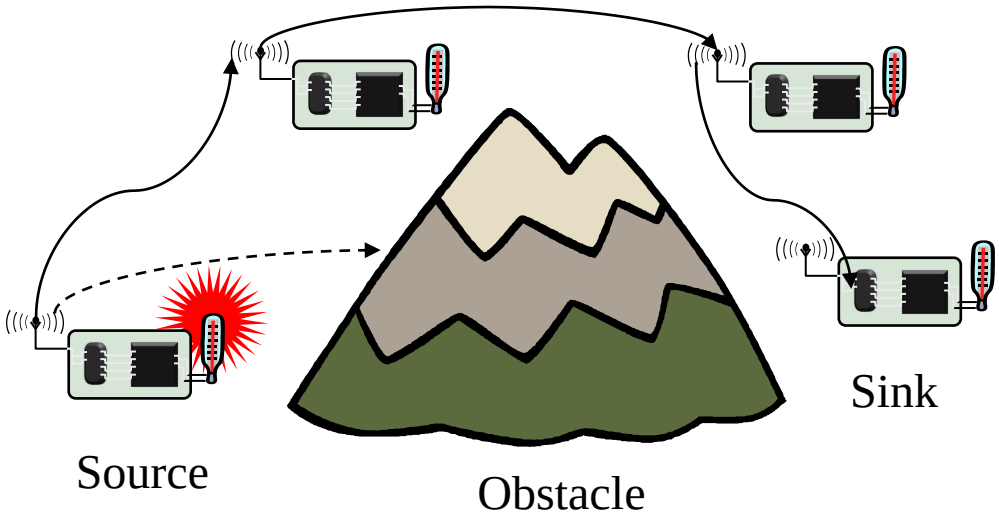


  - Applications: Usually, machine to machine, often limited amounts of data, different notions of importance

Single-hop vs. multi-hop networks

- One common problem: limited range of wireless communication
  - Essentially due to limited transmission power, path loss, obstacles
- Option: multi-hop networks
  - Send packets to an intermediate node
  - Intermediate node forwards packet to its destination
  - ***Store-and-forward*** multi-hop network

- Basic technique applies to both WSN and MANET
- Note: Store&forward multi-hopping NOT the only possible solution
  - E.g., collaborative networking, network coding
  - Do not operate on a per-packet basis



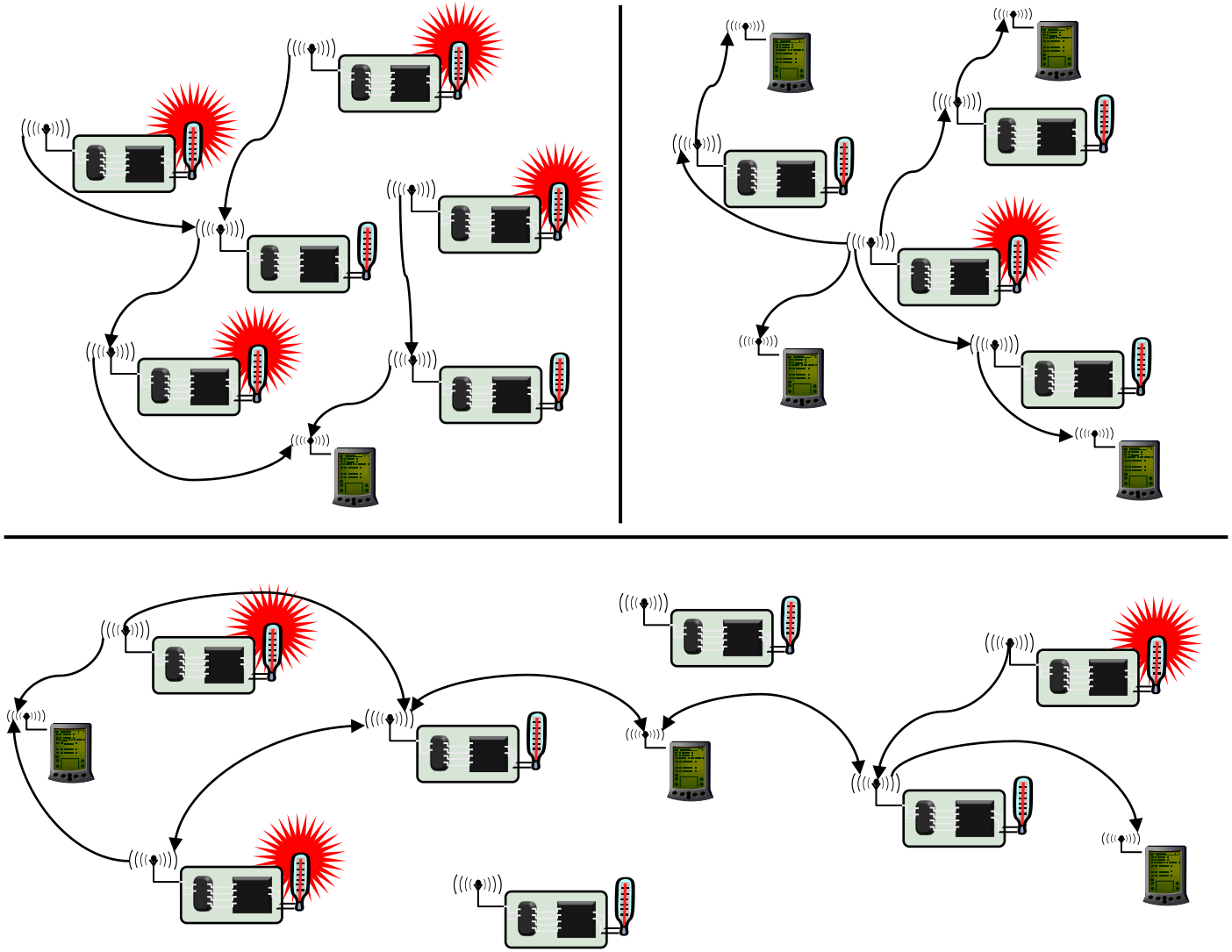
Energy efficiency of multi-hopping?

- Obvious idea: Multi-hopping is more energy-efficient than direct communication
  - Because of path loss $\alpha > 2$, energy for distance d is reduced from cd^{α} to $2c(d/2)^{\alpha}$
    - c some constant

- However: sometime this is wrong, or at least very over-simplified
  - Need to take constant offsets for powering transmitter, receiver into account

! Multi-hopping for energy savings needs careful choice

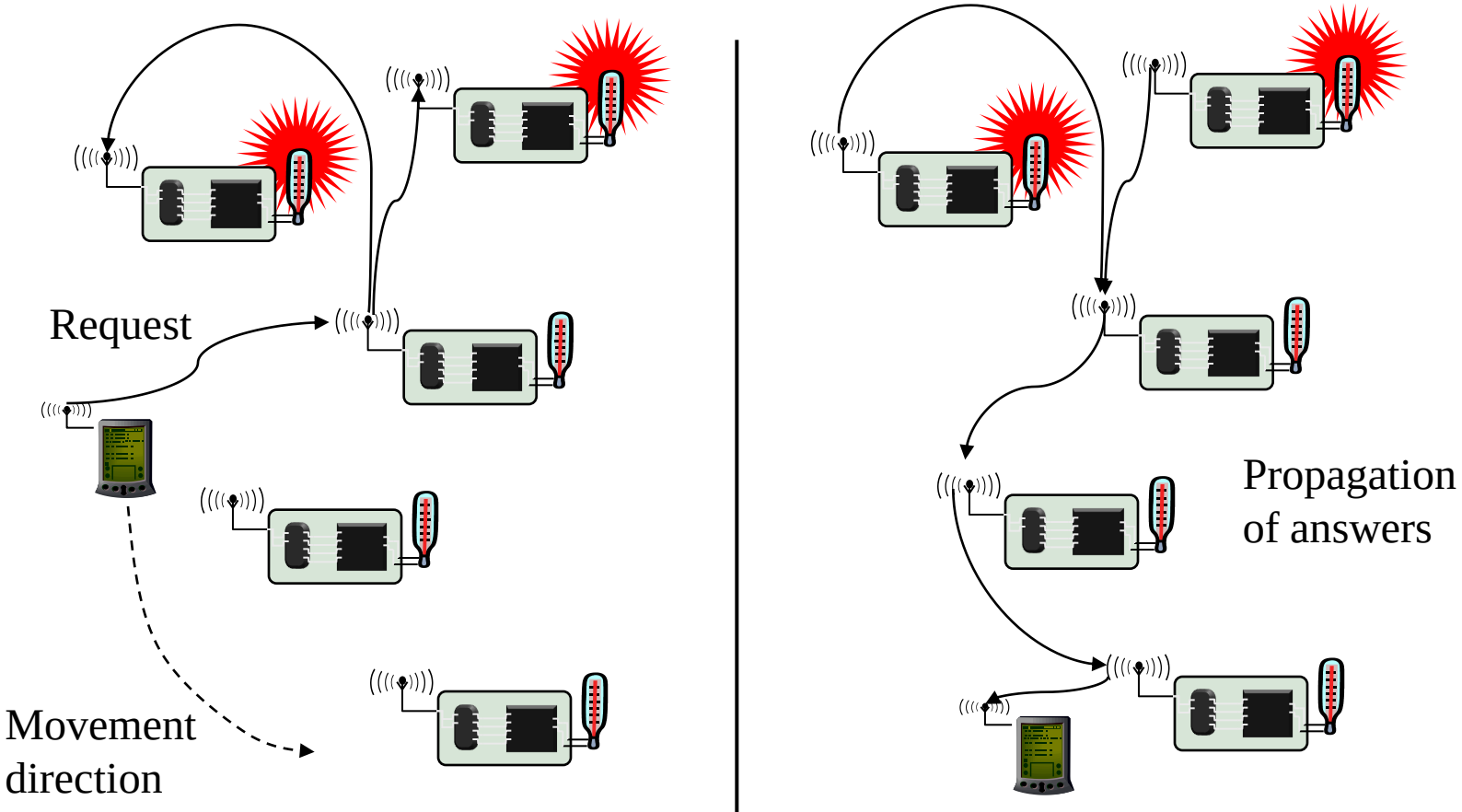
WSN: Multiple sinks, multiple sources



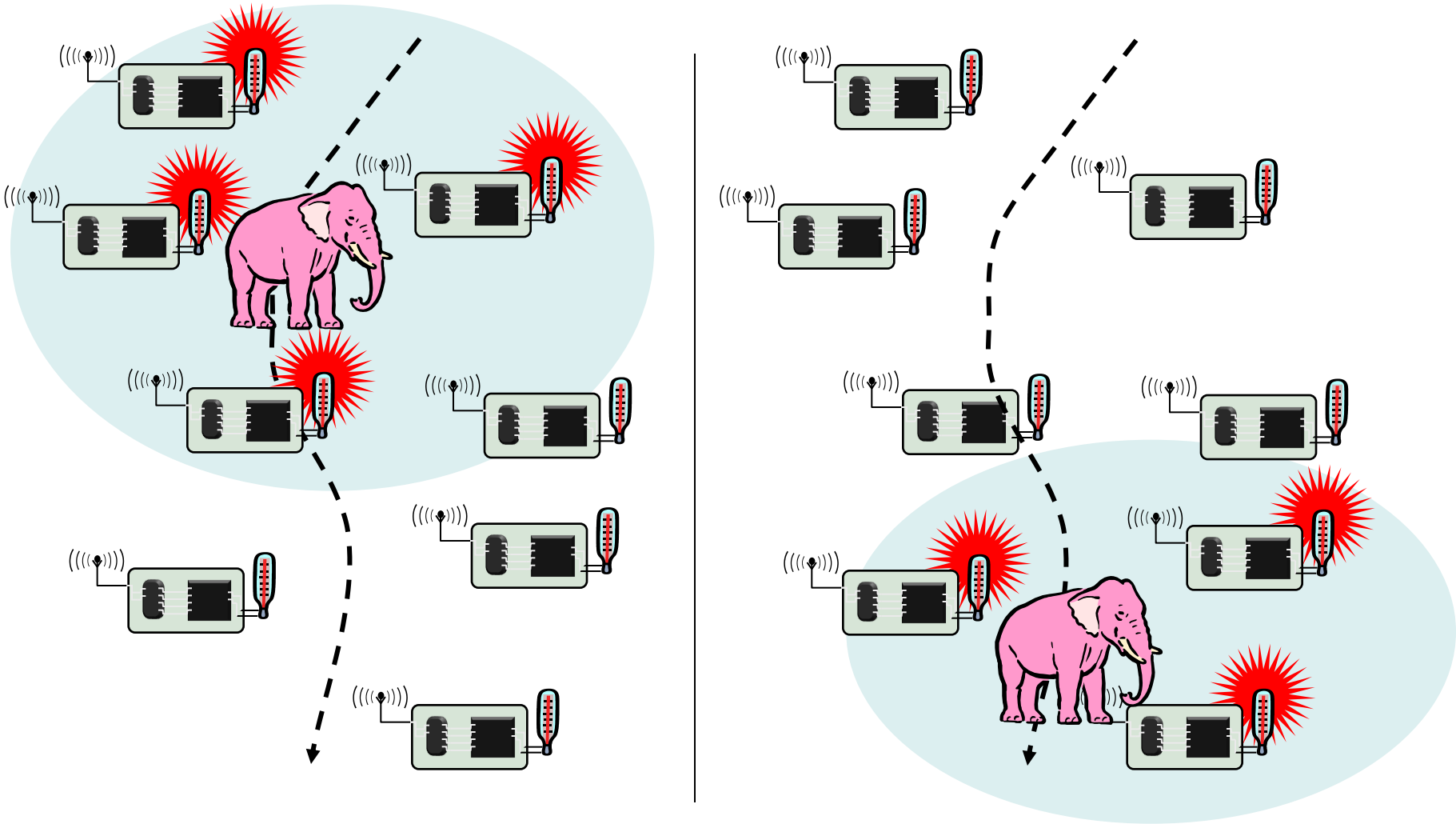
Different sources of mobility

- Node mobility
  - A node participating as source/sink (or destination) or a relay node might move around
  - Deliberately, self-propelled or by external force; targeted or at random
  - Happens in both WSN and MANET
- Sink mobility
  - In WSN, a sink that is not part of the WSN might move
  - Mobile requester
- Event mobility
  - In WSN, event that is to be observed moves around (or extends, shrinks)
  - Different WSN nodes become “responsible” for surveillance of such an event

WSN sink mobility



WSN event mobility: Track the pink elephant



Here: Frisbee model as example

Outline

- Network scenarios
- ***Optimization goals***
- Design principles
- Service interface
- Gateway concepts

Optimization goal: Quality of Service

- In MANET: Usual QoS interpretation
  - Throughput/delay/jitter
  - High perceived QoS for multimedia applications
- In WSN, more complicated
  - Event detection/reporting probability
  - Event classification error, detection delay
  - Probability of missing a periodic report
  - Approximation accuracy (e.g, when WSN constructs a temperature map)
  - Tracking accuracy (e.g., difference between true and conjectured position of the pink elephant)
- Related goal: robustness
  - Network should withstand failure of some nodes

Optimization goal: Energy efficiency

- Umbrella term!
- Energy per correctly received bit
  - Counting all the overheads, in intermediate nodes, etc.
- Energy per reported (unique) event
  - After all, information is important, not payload bits!
  - Typical for WSN
- Delay/energy tradeoffs
- Network lifetime
  - Time to first node failure
  - Network half-life (how long until 50% of the nodes died?)
  - Time to partition
  - Time to loss of coverage
  - Time to failure of first event notification

Optimization goal: Scalability

- Network should be operational regardless of number of nodes
  - At high efficiency
- Typical node numbers difficult to guess
  - MANETs: 10s to 100s
  - WSNs: 10s to 1000s, maybe more (although few people have seen such a network before…)
- Requiring to scale to large node numbers has ***serious*** consequences for network architecture
  - Might not result in the most efficient solutions for small networks!
  - Carefully consider actual application needs before looking for n ! 1 solutions!

Outline

- Network scenarios
- Optimization goals
- ***Design principles***
- Service interface
- Gateway concepts

Distributed organization

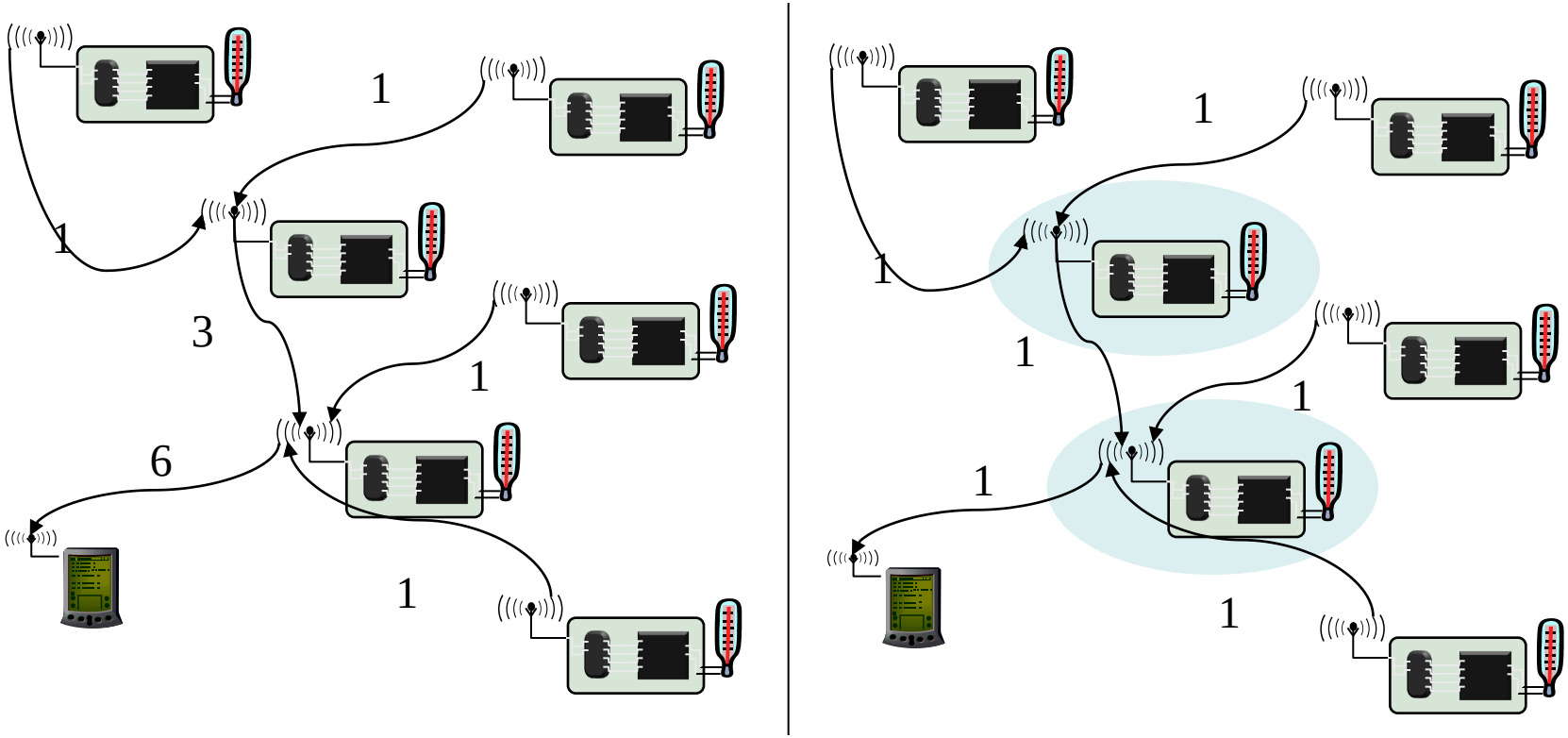
- Participants in a MANET/WSN should cooperate in organizing the network
  - E.g., with respect to medium access, routing, …
  - Centralistic approach as alternative usually not feasible – hinders scalability, robustness
- Potential shortcomings
  - Not clear whether distributed or centralistic organization achieves better energy efficiency (when taking all overheads into account)
- Option: “limited centralized” solution
  - Elect nodes for local coordination/control
  - Perhaps rotate this function over time

In-network processing

- MANETs are supposed to deliver bits from one end to the other
- WSNs, on the other end, are expected to provide information, not necessarily original bits
  - Gives addition options
  - E.g., ***manipulate*** or ***process*** the data in the network
- Main example: aggregation
  - Apply composable aggregation functions to a convergecast tree in a network
  - Typical functions: minimum, maximum, average, sum, …
  - Not amenable functions: median

In-network processing: Aggregation example

- Reduce number of transmitted bits/packets by applying an aggregation function in the network



In-network processing: signal processing

- Depending on application, more sophisticated processing of data can take place within the network
  - Example edge detection: locally exchange raw data with neighboring nodes, compute edges, only communicate edge description to far away data sinks
  - Example tracking/angle detection of signal source: Conceive of sensor nodes as a distributed microphone array, use it to compute the angle of a single source, only communicate this angle, not all the raw data
- Exploit ***temporal*** and ***spatial correlation***
  - Observed signals might vary only slowly in time ! no need to transmit all data at full rate all the time
  - Signals of neighboring nodes are often quite similar ! only try to transmit differences (details a bit complicated, see later)

Adaptive fidelity

- Adapt the effort with which data is exchanged to the currently required accuracy/fidelity
- Example event detection
  - When there is no event, only very rarely send short “all is well” messages
  - When event occurs, increase rate of message exchanges
- Example temperature
  - When temperature is in acceptable range, only send temperature values at low resolution
  - When temperature becomes high, increase resolution and thus message length

Data centric networking

- In typical networks (including ad hoc networks), network transactions are addressed to the ***identities*** of specific nodes
  - A “node-centric” or “address-centric” networking paradigm
- In a redundantly deployed sensor networks, specific source of an event, alarm, etc. might not be important
  - Redundancy: e.g., several nodes can observe the same area
- Thus: focus networking transactions on the data directly instead of their senders and transmitters ! ***data-centric networking***
  - Principal design change

Implementation options for data-centric networking

- Overlay networks & distributed hash tables (DHT)
  - Hash table: content-addressable memory
  - Retrieve data from an unknown source, like in peer-to-peer networking – with efficient implementation
  - Some disparities remain
    - Static key in DHT, dynamic changes in WSN
    - DHTs typically ignore issues like hop count or distance between nodes when performing a lookup operation
- Publish/subscribe
  - Different interaction paradigm
  - Nodes can ***publish*** data, can ***subscribe*** to any particular kind of data
  - Once data of a certain type has been published, it is delivered to all subscribes
  - Subscription and publication are decoupled in time; subscriber and publisher are agnostic of each other (decoupled in identity)
- Databases

Further design principles

- Exploit location information
  - Required anyways for many applications; can considerably increase performance
- Exploit activity patterns
- Exploit heterogeneity
  - By construction: nodes of different types in the network
  - By evolution: some nodes had to perform more tasks and have less energy left; some nodes received more solar energy than others; …
- Cross-layer optimization of protocol stacks for WSN
  - Goes against grain of standard networking; but promises big performance gains
  - Also applicable to other networks like ad hoc; usually at least worthwhile to consider for most wireless networks

Outline

- Network scenarios
- Optimization goals
- Design principles
- ***Service interface***
- Gateway concepts

Interfaces to protocol stacks

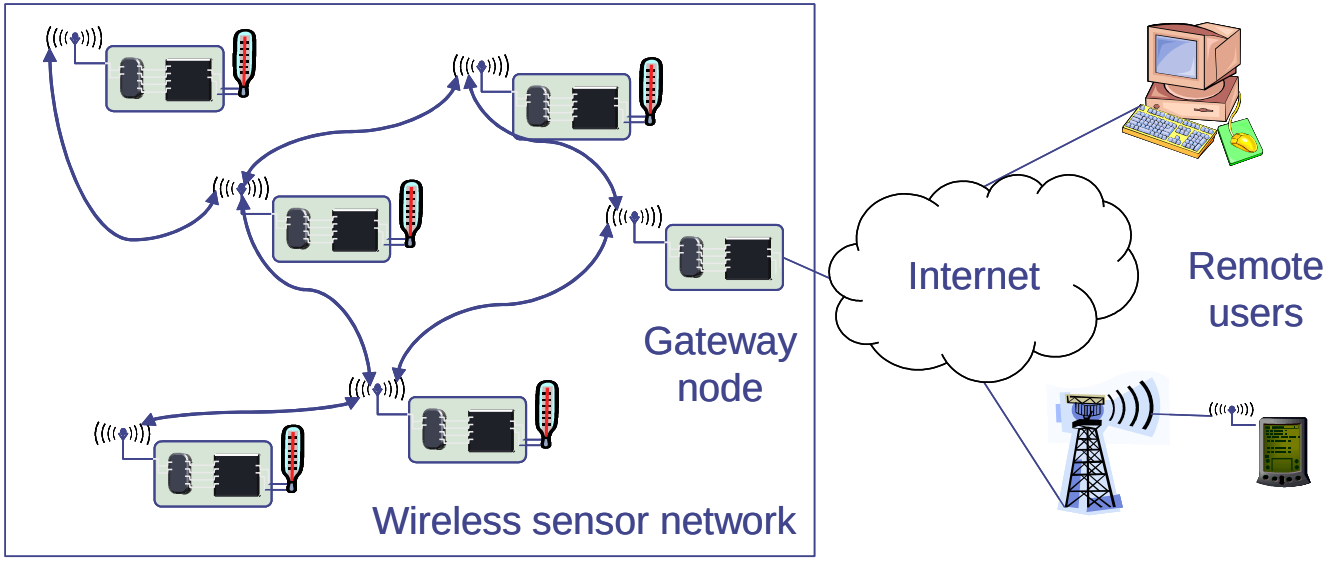
- The world's all-purpose network interface: sockets
  - Good for transmitting data from one sender to one receiver
  - Not well matched to WSN needs (ok for ad hoc networks)
- Expressibility requirements
  - Support for simple request/response interactions
  - Support for asynchronous event notification
  - Different ways for identifying addressee of data
    - By location, by observed values, implicitly by some other form of group membership
    - By some semantically meaningful form – "room 123"
  - Easy accessibility of in-network processing functions
    - Formulate complex events – events defined only by several nodes
  - Allow to specify accuracy & timeliness requirements
  - Access node/network status information (e.g., battery level)
  - Security, management functionality, …
- No clear standard has emerged yet – many competing, unclear proposals

Outline

- Network scenarios
- Optimization goals
- Design principles
- Service interface
- ***Gateway concepts***

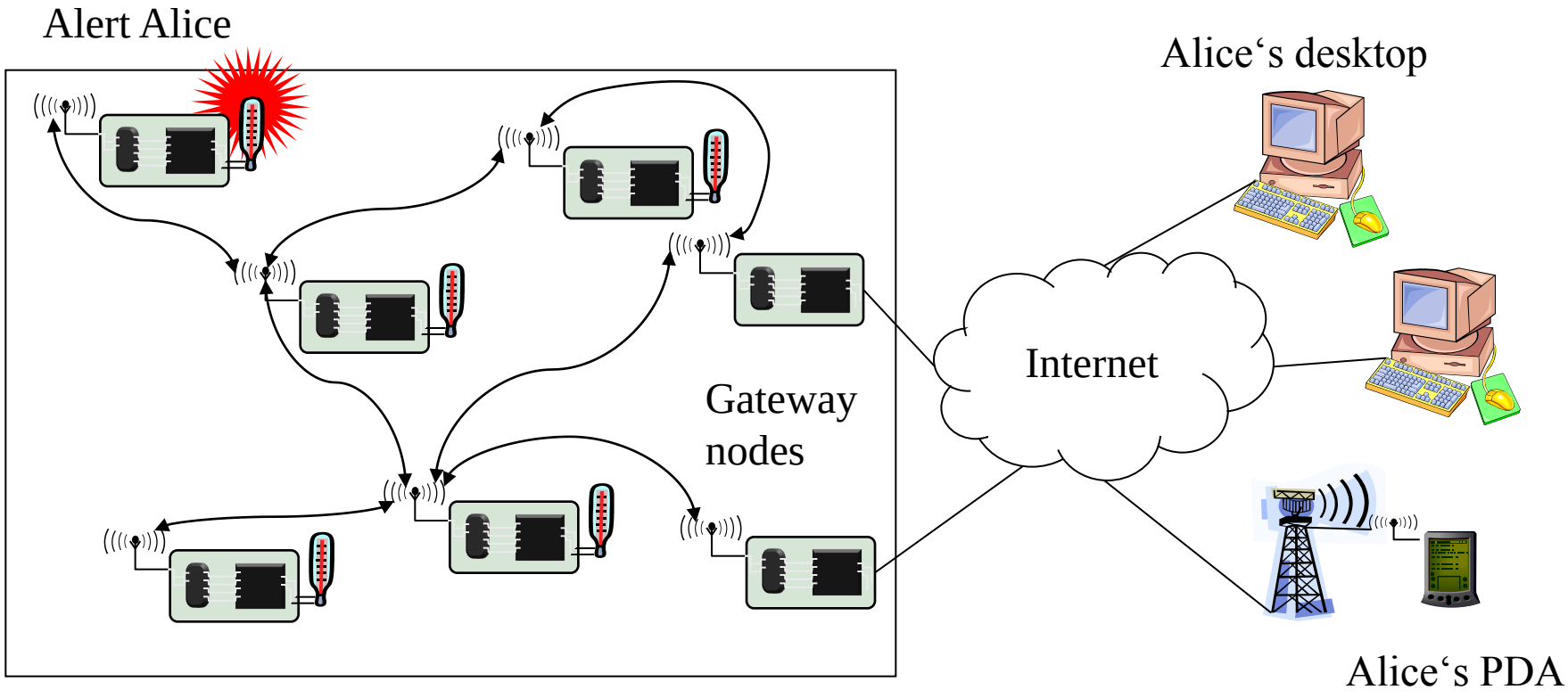
Gateway concepts for WSN/MANET

- Gateways are necessary to the Internet for remote access to/from the WSN
  - Same is true for ad hoc networks; additional complications due to mobility (change route to the gateway; use different gateways)
  - WSN: Additionally bridge the gap between different interaction semantics (data vs. address-centric networking) in the gateway
- Gateway needs support for different radios/protocols, …



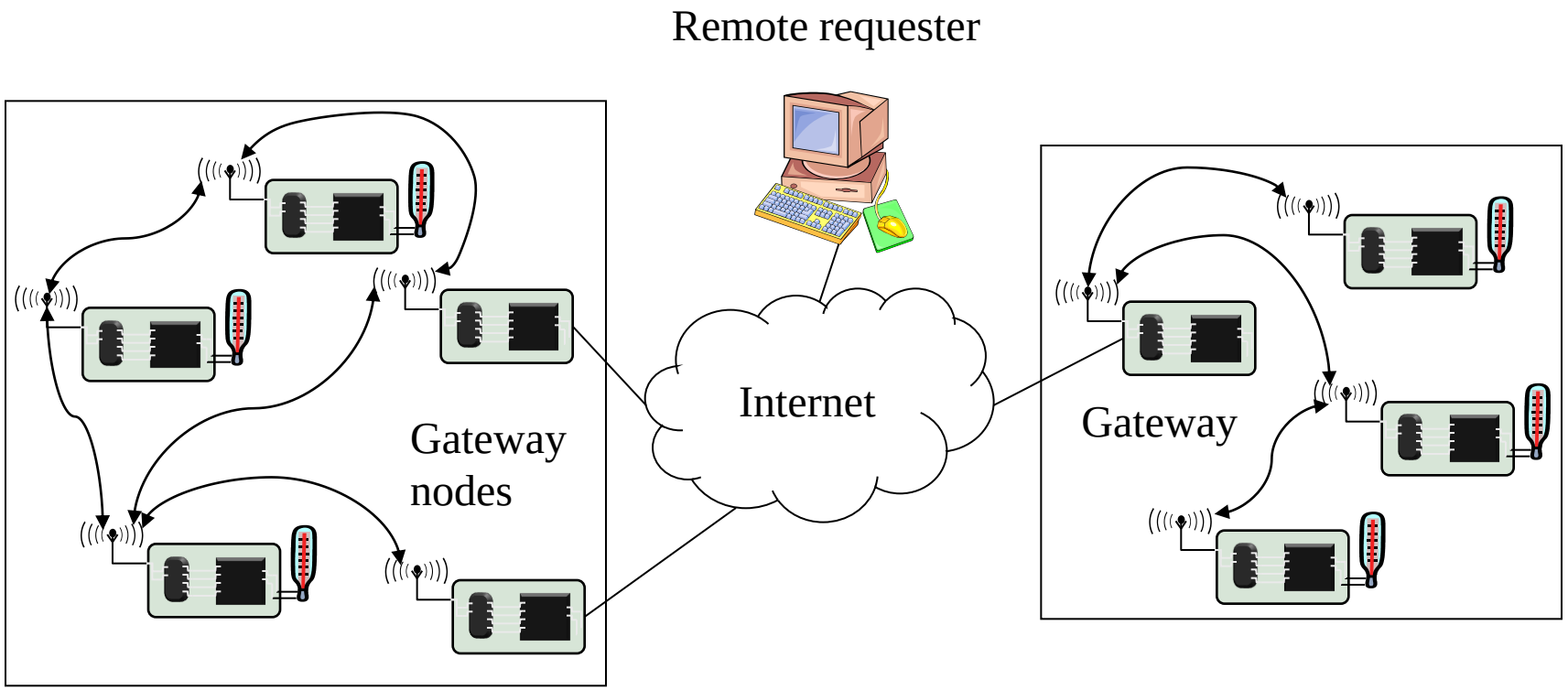
WSN to Internet communication

- Example: Deliver an alarm message to an Internet host
- Issues
  - Need to find a gateway (integrates routing & service discovery)
  - Choose “best” gateway if several are available
  - How to find Alice or Alice’s IP?



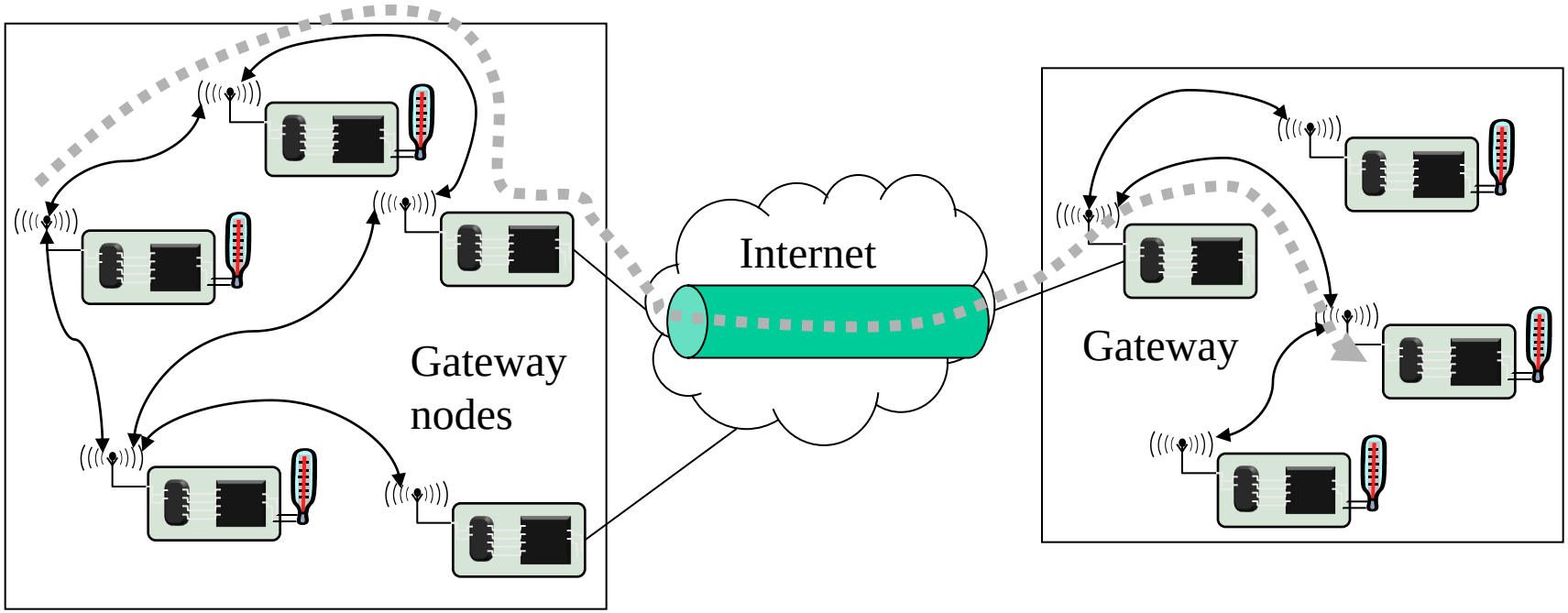
Internet to WSN communication

- How to find the right WSN to answer a need?
- How to translate from IP protocols to WSN protocols, semantics?



WSN tunneling

- Use the Internet to “tunnel” WSN packets between two remote WSNs



Summary

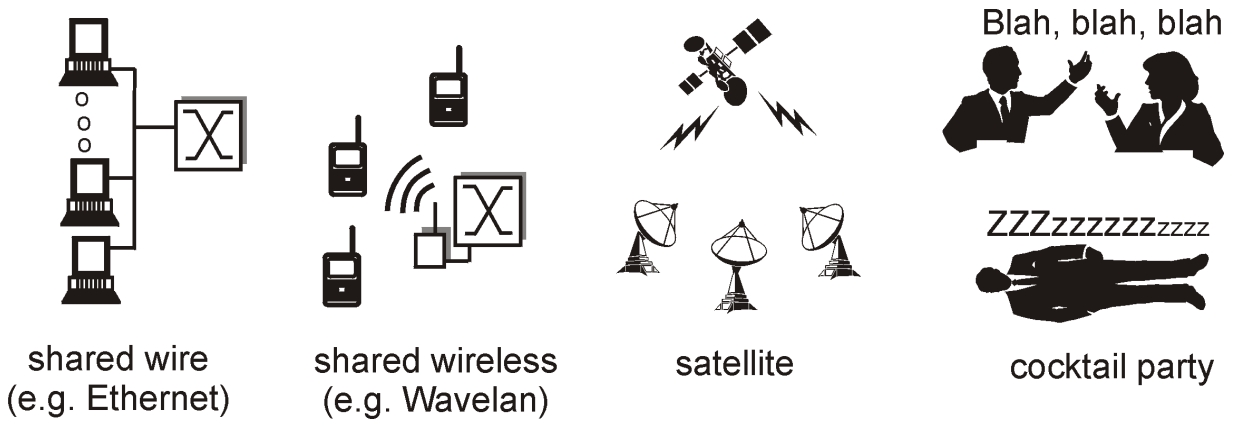
- Network architectures for ad hoc networks are – in principle – relatively straightforward and similar to standard networks
  - Mobility is compensated for by appropriate protocols, but interaction paradigms don't change too much
- WSNs, on the other hand, look quite different on many levels
  - Data-centric paradigm, the need and the possibility to manipulate data as it travels through the network opens new possibilities for protocol design
- The following chapters will look at how these ideas are realized by actual protocols

MAC OF WSN

Multiple Access Links and Protocols

Two types of “links”:

- point-to-point
  - PPP for dial-up access
  - point-to-point link between switch and host
- broadcast (shared wire or medium)
  - traditional Ethernet
  - upstream HFC
  - 802.11 wireless LAN



Multiple Access protocols

- single shared broadcast channel
- two or more simultaneous transmissions by nodes: interference
  - collision if node receives two or more signals at the same time

multiple access protocol

- distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit
- communication about channel sharing must use channel itself!
  - no out-of-band channel for coordination

Ideal Multiple Access Protocol

Broadcast channel of rate R bps

1. When one node wants to transmit, it can send at rate R.
2. When M nodes want to transmit, each can send at average rate R/M
3. Fully decentralized:
   - no special node to coordinate transmissions
   - no synchronization of clocks, slots
4. Simple

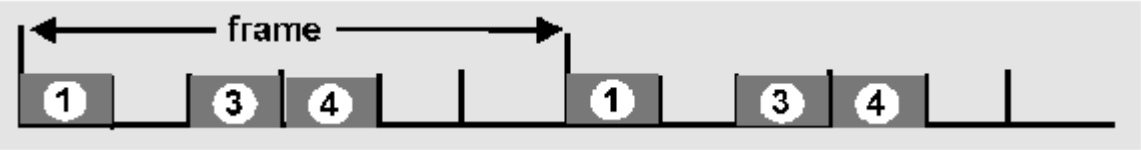
MAC Protocols: a taxonomy

- Static Channel Allocation
  - divide channel into smaller “pieces” (time slots, frequency, code)
  - allocate piece to node for exclusive use
- Dynamic Channel Allocation
  - channel not divided, allow collisions
  - “recover” from collisions

Static Channel Allocation: TDMA

TDMA: time division multiple access

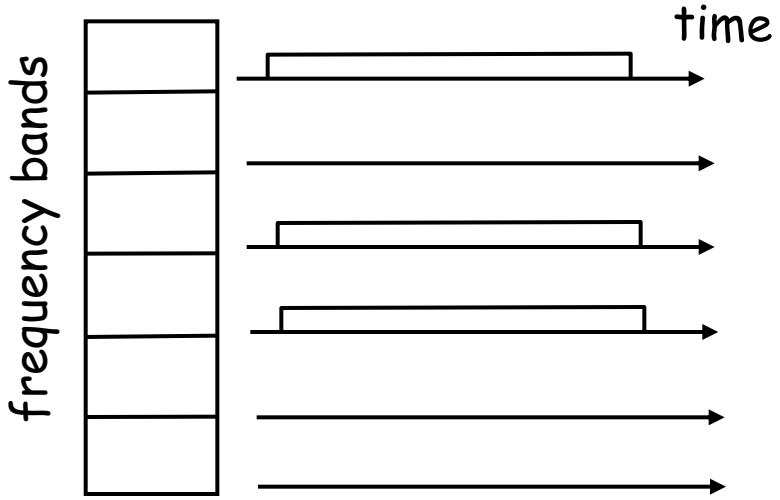
- access to channel in "rounds"
- each station gets fixed length slot (length = pkt trans time) in each round
- unused slots go idle
- example: 6-station LAN, 1,3,4 have pkt, slots 2,5,6 idle



Static Channel Allocation: FDMA

FDMA: frequency division multiple access

- channel spectrum divided into frequency bands
- each station assigned fixed frequency band
- unused transmission time in frequency bands go idle
- example: 6-station LAN, 1,3,4 have pkt, frequency bands 2,5,6 idle



Dynamic Channel Allocation

- ALOHA, slotted ALOHA
- CSMA, CSMA/CD, CSMA/CA – in future slides
- Collision free protocols
  - Bit-map protocol
  - Binary countdown

Not currently used in major system yet

- Limited contention protocols
  - Adaptive tree walk
- Wavelength division multiple access protocol

Ethernet uses CSMA/CD

- No slots
- adapter doesn't transmit if it senses that some other adapter is transmitting, that is, carrier sense
- transmitting adapter aborts when it senses that another adapter is transmitting, that is, collision detection
- Before attempting a retransmission, adapter waits a random time, that is, random access

MAC FOR WSN

Goals of this chapter

- Controlling when to send a packet and when to listen for a packet are perhaps the two most important operations in a wireless network
  - Especially, idly waiting wastes huge amounts of energy
- This chapter discusses schemes for this medium access control that are
  - Suitable to mobile and wireless networks
  - Emphasize energy-efficient operation

Overview

- ***Principal options and difficulties***
- Contention-based protocols
- Schedule-based protocols
- IEEE 802.15.4

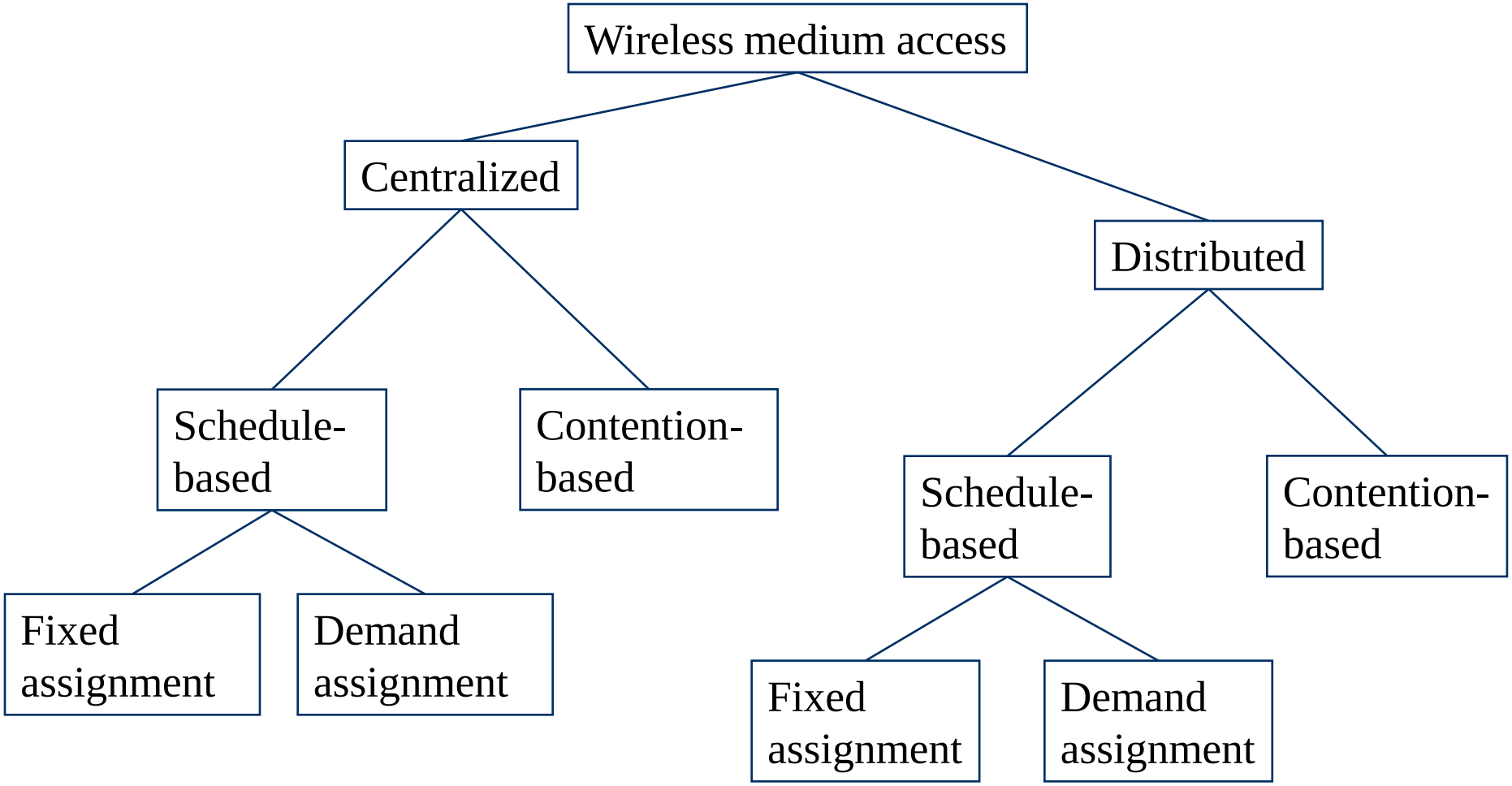
Principal options and difficulties

- Medium access in wireless networks is difficult mainly because of
  - Impossible (or very difficult) to send and receive at the same time
  - Interference situation at receiver is what counts for transmission success, but can be very different from what sender can observe
  - High error rates (for signaling packets) compound the issues
- Requirement
  - As usual: high throughput, low overhead, low error rates, …
  - Additionally: energy-efficient, handle switched off devices!

Requirements for energy-efficient MAC protocols

- Recall
  - Transmissions are costly
  - Receiving about as expensive as transmitting
  - Idling can be cheaper but is still expensive
- Energy problems
  - ***Collisions*** – wasted effort when two packets collide
  - ***Overhearing*** – waste effort in receiving a packet destined for another node
  - ***Idle listening*** – sitting idly and trying to receive when nobody is sending
  - ***Protocol overhead***
- Always nice: Low complexity solution

Main options



Centralized medium access

- Idea: Have a central station control when a node may access the medium
  - Example: Polling, centralized computation of TDMA schedules
  - Advantage: Simple, quite efficient (e.g., no collisions), burdens the central station
- Not directly feasible for non-trivial wireless network sizes
- But: Can be quite useful when network is somehow divided into smaller groups
  - Clusters, in each cluster medium access can be controlled centrally – compare Bluetooth piconets, for example

! Usually, distributed medium access is considered

Schedule- vs. contention-based MACs

- ***Schedule-based*** MAC
  - A ***schedule*** exists, regulating which participant may use which resource at which time (TDMA component)
  - Typical resource: frequency band in a given physical space (with a given code, CDMA)
  - Schedule can be ***fixed*** or computed ***on demand***
    - Usually: mixed – difference fixed/on demand is one of time scales
  - Usually, collisions, overhearing, idle listening no issues
  - Needed: time synchronization!
- ***Contention-based*** protocols
  - Risk of colliding packets is deliberately taken
  - Hope: coordination overhead can be saved, resulting in overall improved efficiency
  - Mechanisms to handle/reduce probability/impact of collisions required
  - Usually, ***randomization*** used somehow

Overview

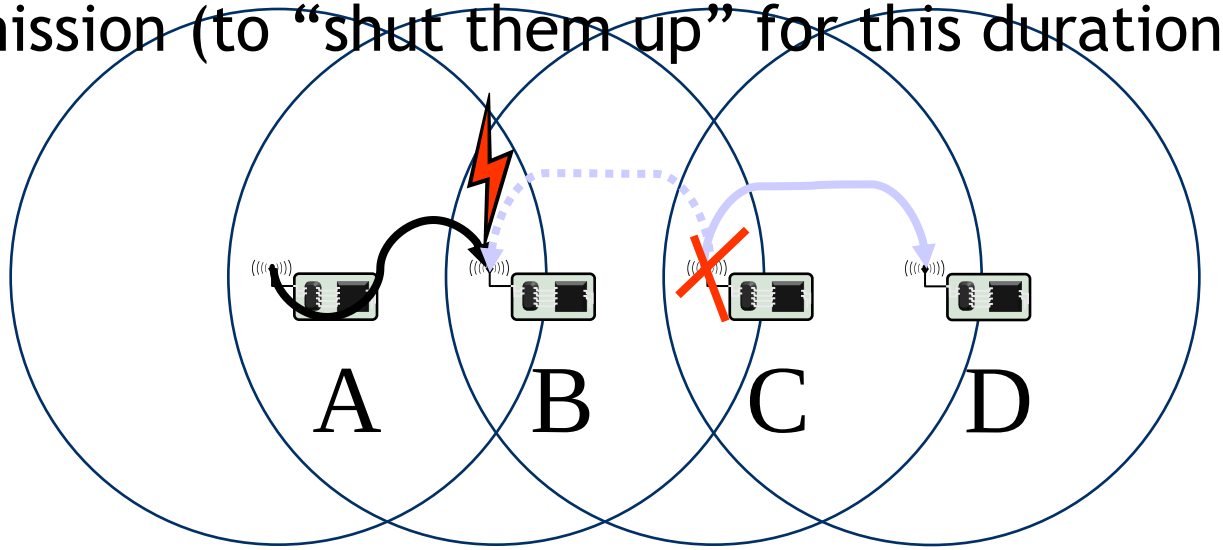
- Principal options and difficulties
- ***Contention-based protocols***
  - MACA
  - S-MAC, T-MAC
  - Preamble sampling, B-MAC
  - PAMAS
- Schedule-based protocols
- IEEE 802.15.4

Distributed, contention-based MAC

- Basic ideas for a distributed MAC
  - ALOHA – no good in most cases
  - Listen before talk ***(Carrier Sense Multiple Access, CSMA)*** - better, but suffers from ***sender*** not knowing what is going on at ***receiver***, might destroy packets despite first listening

! Receiver additionally needs some possibility to inform possible senders in its vicinity about impending transmission (to “shut them up” for this duration)

Hidden terminal scenario:



Also: recall exposed terminal scenario

Main options to shut up senders

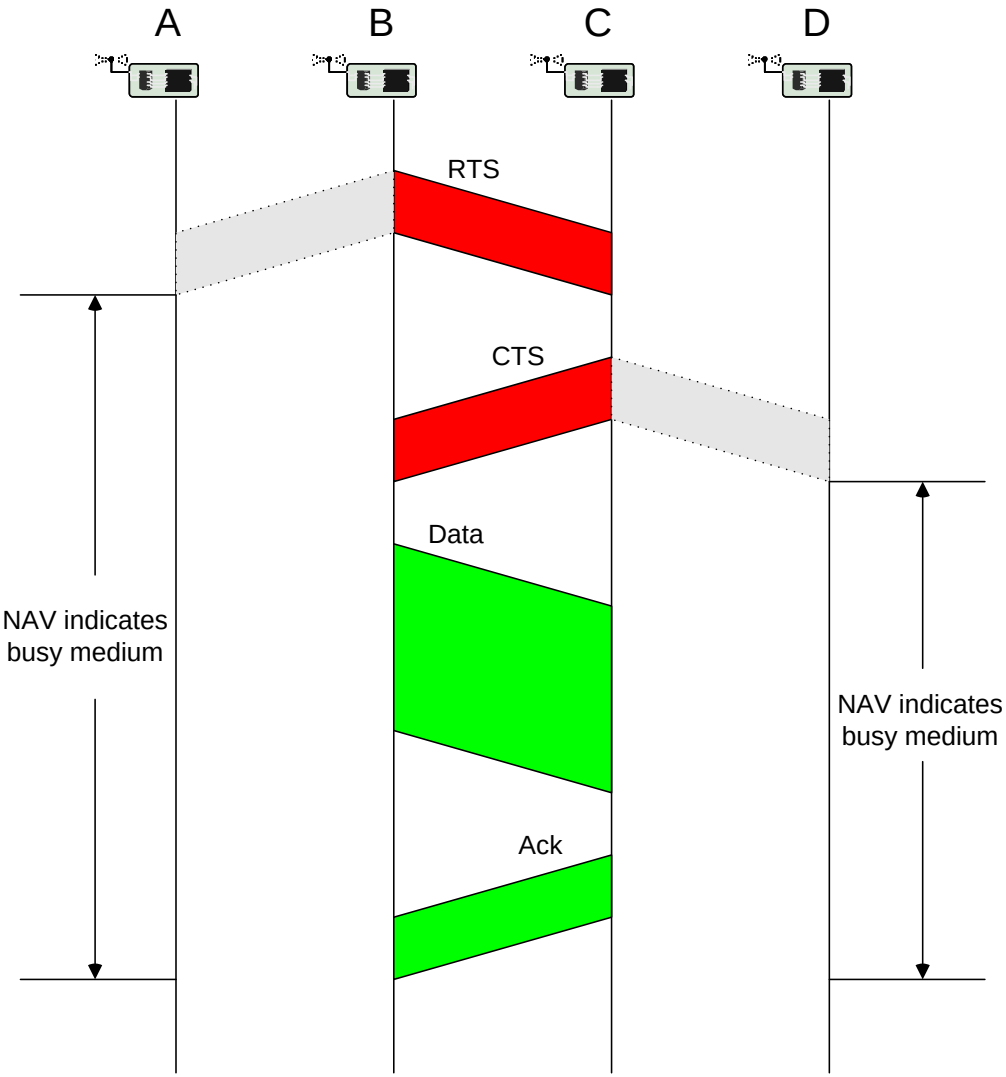
- Receiver informs potential interferers ***while*** a reception is on-going
  - By sending out a signal indicating just that
  - Problem: Cannot use same channel on which actual reception takes place

  ! Use separate channel for signaling
  - ***Busy tone*** protocol
- Receiver informs potential interferers ***before*** a reception is on-going
  - Can use same channel
  - Receiver itself needs to be informed, by sender, about impending transmission
  - Potential interferers need to be aware of such information, need to store it

Receiver informs interferers before transmission – MACA

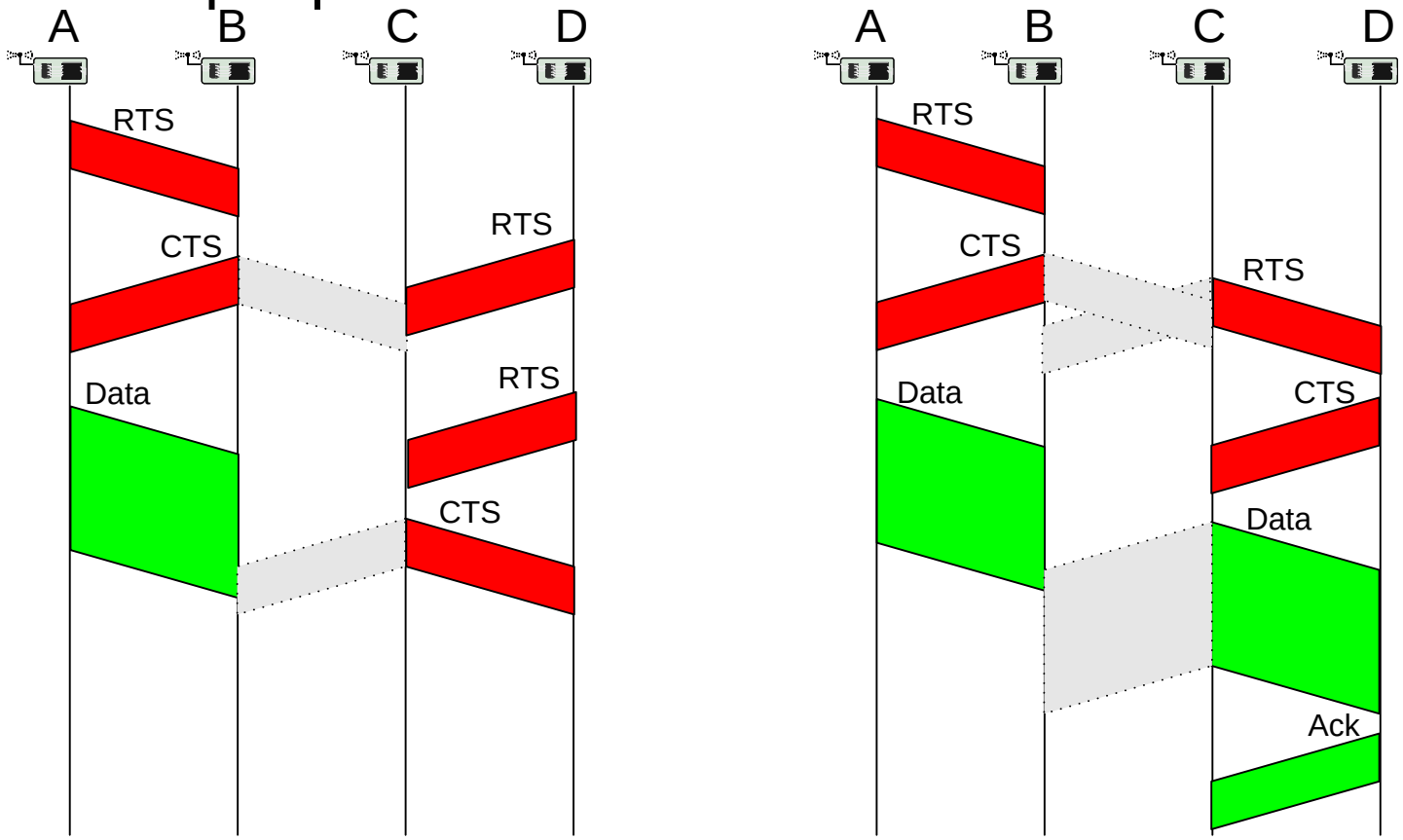
- Sender B asks receiver C whether C is able to receive a transmission
  Request to Send (RTS)
- Receiver C agrees, sends out a ***Clear to Send*** (***CTS***)
- Potential interferers overhear either RTS or CTS and know about impending transmission and for how long it will last
  - Store this information in a ***Network Allocation Vector***
- B sends, C acks

! ***MACA protocol*** (used e.g. in ***IEEE 802.11***)



RTS/CTS

- RTS/CTS ameliorate, but do not solve hidden/exposed terminal problems
- Example problem cases:



MACA Problem: Idle listening

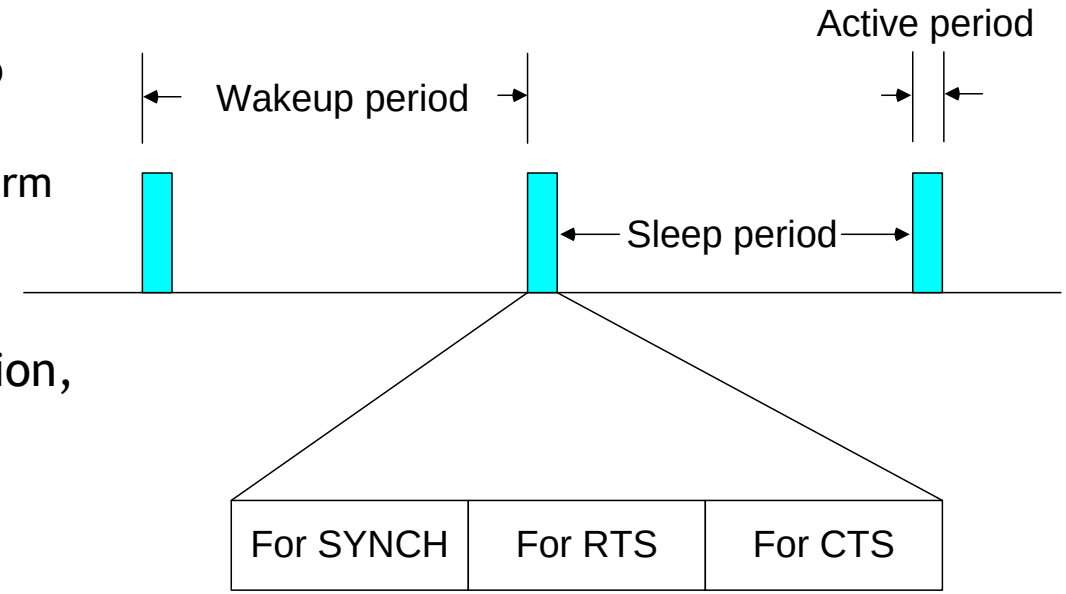
- Need to sense carrier for RTS or CTS packets
  - In some form shared by many CSMA variants; but e.g. not by busy tones
  - Simple sleeping will break the protocol
- IEEE 802.11 solution: ATIM windows & sleeping
  - Basic idea: Nodes that have data buffered for receivers send ***traffic indicators*** at pre-arranged points in time
  - Receivers need to wake up at these points, but can sleep otherwise
- Parameters to adjust in MACA
  - Random delays – how long to wait between listen/transmission attempts?
  - Number of RTS/CTS/ACK re-trials?
  - …

Medium Access Control in Sensor Nets

- Important attributes of MAC protocols (in the past)
  1. **Collision avoidance**
  2. **Energy efficiency**
  3. **Scalability in node density**
  4. Latency
  5. Fairness
  6. Throughput
  7. Bandwidth utilization

Sensor-MAC (S-MAC)

- MACA's idle listening is particularly unsuitable if average data rate is low
  - Most of the time, nothing happens
- Idea: Switch nodes off, ensure that neighboring nodes turn on simultaneously to allow packet exchange (rendez-vous)
  - Only in these ***active periods***, packet exchanges happen
  - Need to also exchange wakeup schedule between neighbors
  - When awake, essentially perform RTS/CTS
- Use SYNCH, RTS, CTS phases
- After they start data transmission, they do not follow the sleep schedule – transmit until all segments of a packet done



S-MAC: Periodic Listen and Sleep

- Motivation: Idle listening consumes significant energy
  - Nodes do not sleep in IEEE 802.11 ad hoc mode

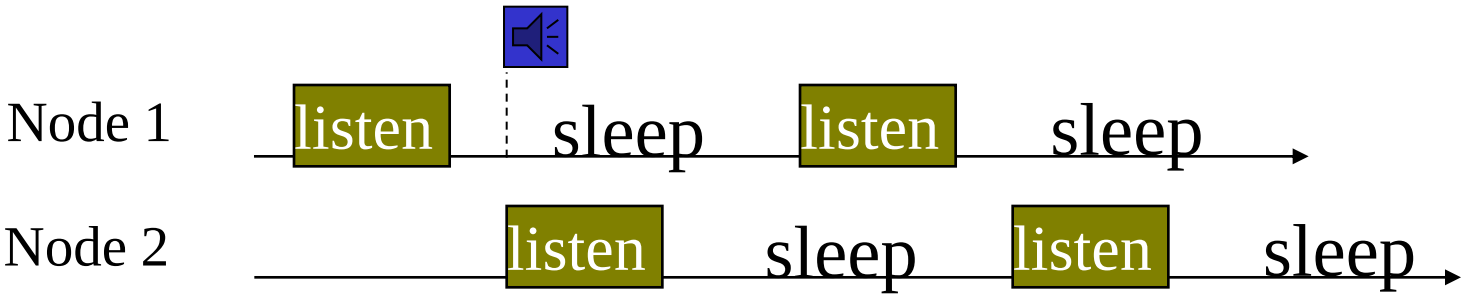
- **Solution:** Periodic listen and sleep
  - Turn off radio when sleeping
  - Reduce duty cycle to ~10% (200 ms on/2s off)
  - Increased latency for reduced energy

S-MAC: Periodic Listen and Sleep

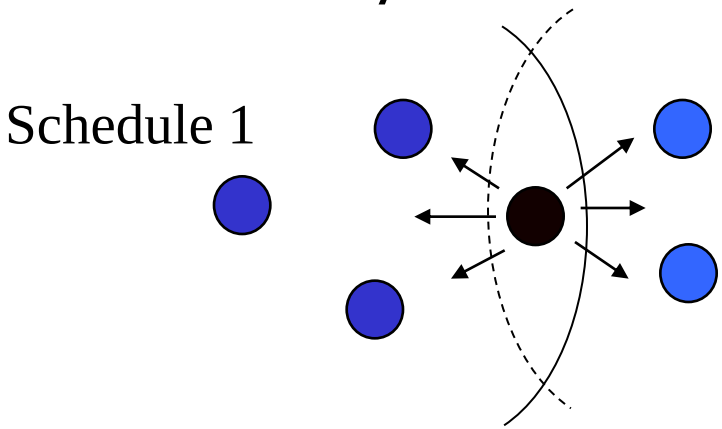
- Schedule maintenance
  - Remember neighbors' schedules
    - to know when to send to them
  - Each node broadcasts its schedule every few periods
  - Refresh on neighbor's schedule when receiving an update
  - Schedule packets also serve as beacons for new nodes to join a neighborhood

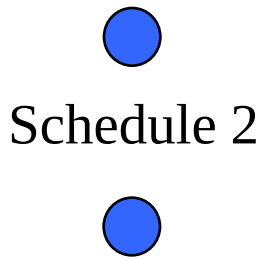
S-MAC: Periodic Listen and Sleep

- Schedules can differ



- *Preferable if* neighboring nodes have same schedule
  — easy broadcast & low control overhead

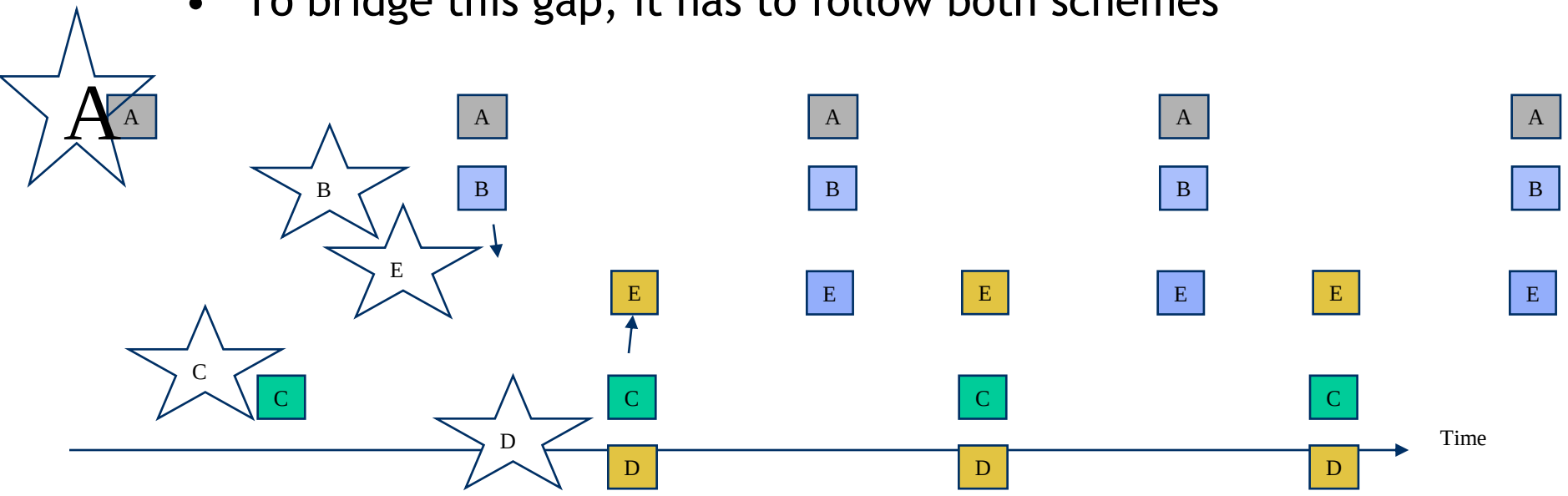




Border nodes:
two schedules
broadcast twice

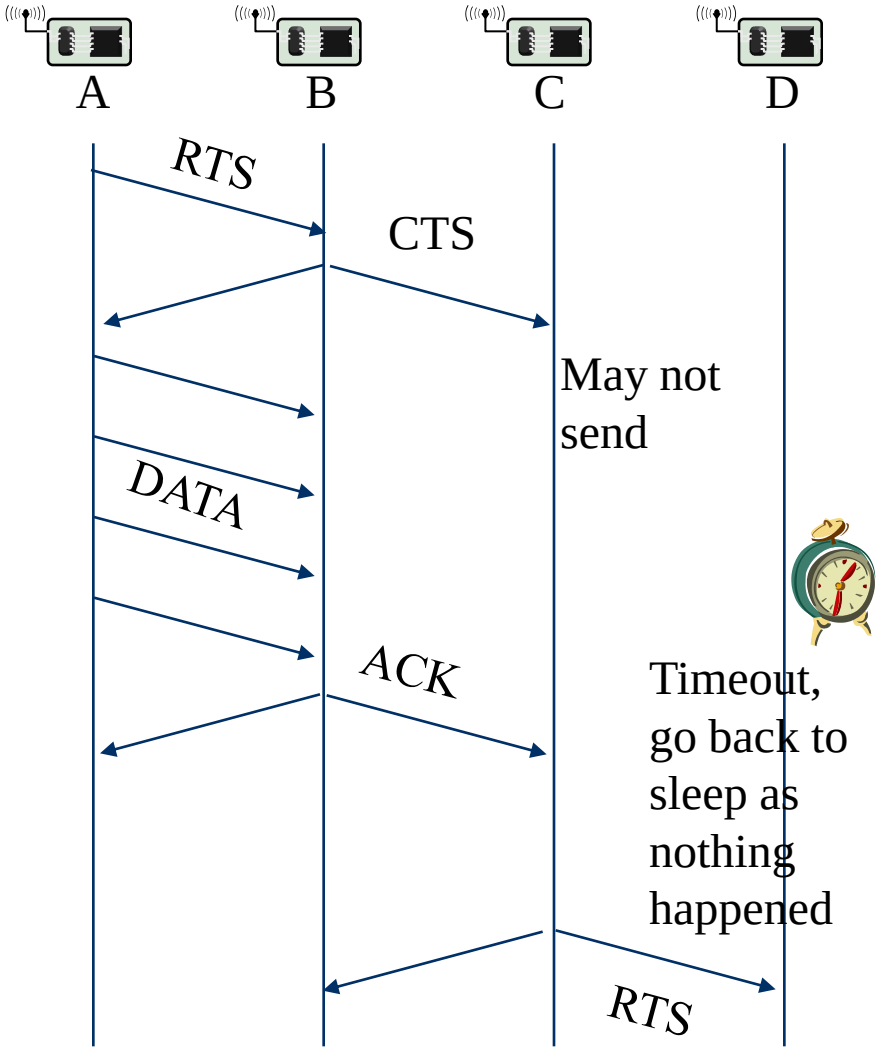
S-MAC synchronized islands

- Nodes try to pick up schedule synchronization from neighboring nodes
- If no neighbor found, nodes pick some schedule to start with
- If additional nodes join, some node might learn about two different schedules from different nodes
  - “Synchronized islands”
- To bridge this gap, it has to follow both schemes



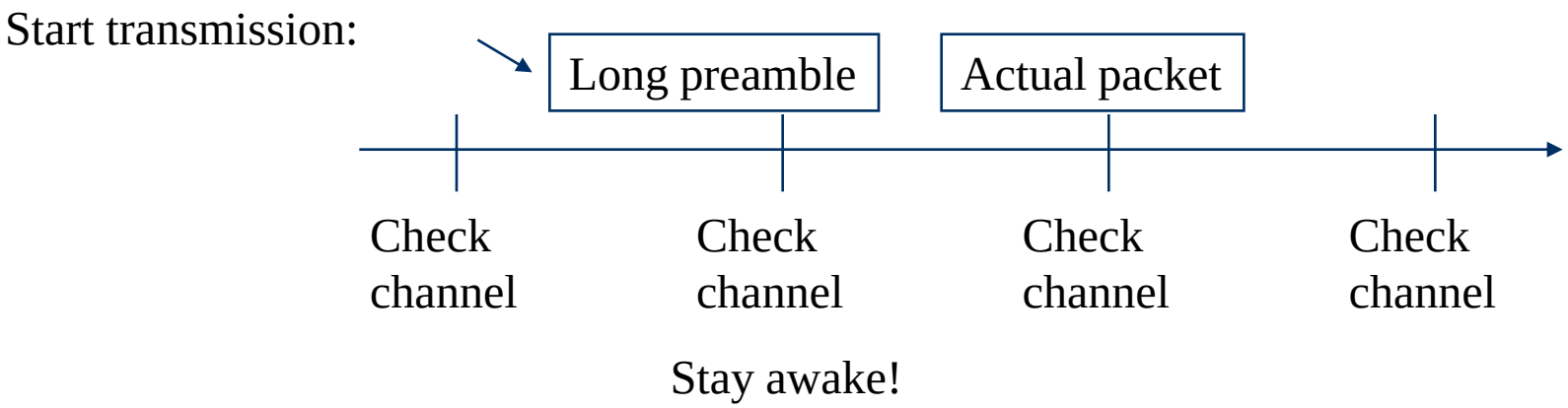
Timeout-MAC (T-MAC)

- In S-MAC, active period is of constant length
- What if no traffic actually happens?
  - Nodes stay awake needlessly long
- Idea: Prematurely go back to sleep mode when no traffic has happened for a certain time (=timeout): T-MAC
  - After the SYNC section of the active period, there is a short window to send or receive RTS and CTS packets. If no activity happens in that period, the node goes to sleep.
  - Adaptive duty cycle!
- One ensuing problem: Early sleeping
  - C wants to send to D, but is hindered by transmission A->B



Preamble Sampling

- So far: Periodic sleeping supported by some means to synchronize wake up of nodes to ensure rendez-vous between sender and receiver
- Alternative option: Don't try to explicitly synchronize nodes
  - Have receiver sleep and only periodically sample the channel
- Use ***long preambles*** to ensure that receiver stays awake to catch actual packet
  - Example: WiseMAC



B-MAC

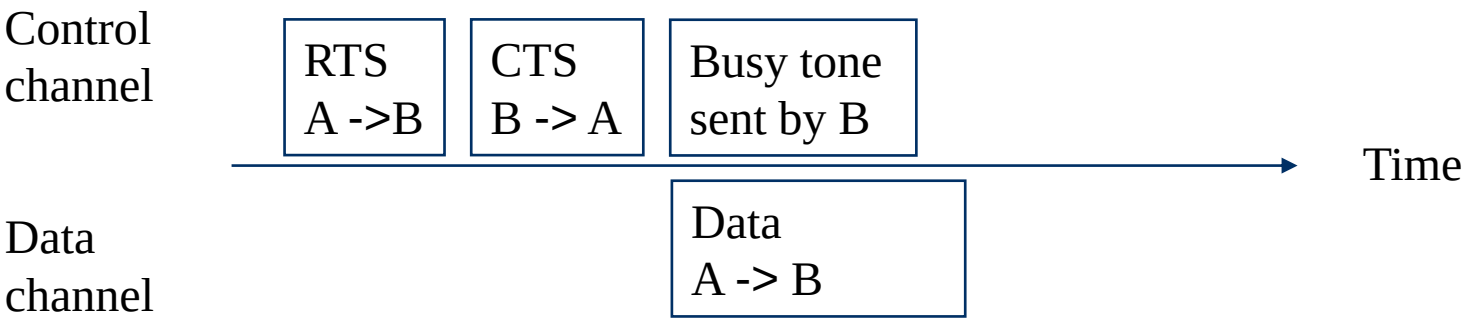
- Combines several of the above discussed ideas
  - Takes care to provide practically relevant solutions
- Clear Channel Assessment
  - Adapts to noise floor by sampling channel when it is assumed to be free
  - Samples are exponentially averaged, result used in gain control
  - For actual assessment when sending a packet, look at five channel samples – channel is free if even a single one of them is significantly below noise
  - Optional: random backoff if channel is found busy
- Optional: Immediate link layer acknowledgements for received packets

B-MAC II

- Low Power Listening (= preamble sampling)
  - Uses the clear channel assessment techniques to decide whether there is a packet arriving when node wakes up
  - Timeout puts node back to sleep if no packet arrived
- B-MAC does *not* have
  - Synchronization
  - RTS/CTS
  - Results in simpler, leaner implementation
  - Clean and simple interface
- Currently: Often considered as the ***default WSN MAC*** protocol

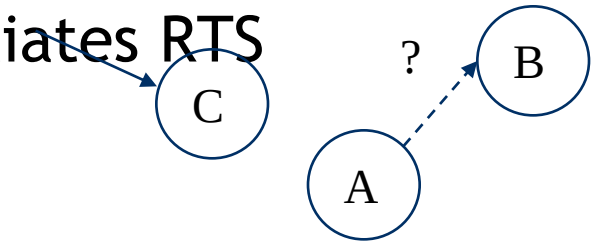
Power Aware Multiaccess with Signaling - PAMAS

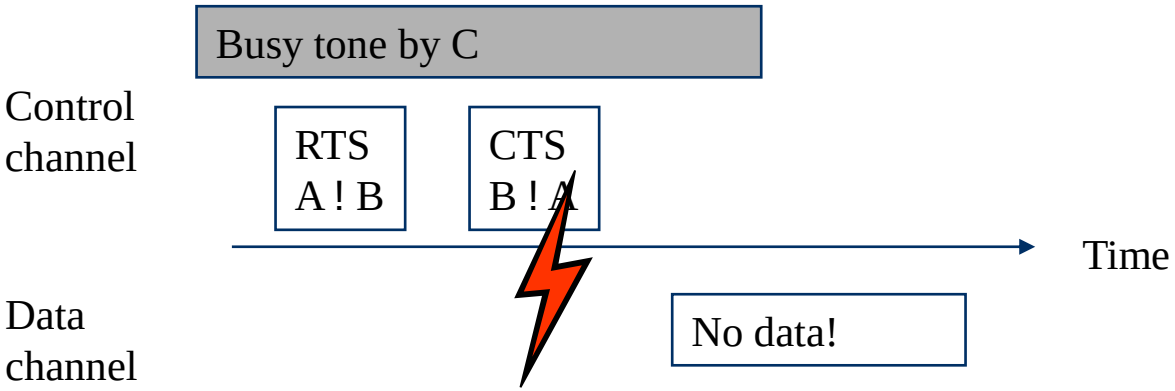
- Idea: combine busy tone with RTS/CTS
  - Results in detailed overhearing avoidance, does not address idle listening
  - Uses separate ***data*** and ***control channels***
- Procedure
  - Node A transmits RTS on control channel, does not sense channel
  - Node B receives RTS, sends CTS on control channel if it can receive and does not know about ongoing transmissions
  - B sends busy tone as it starts to receive data



PAMAS – Already ongoing transmission

- Suppose a node C in vicinity of A is already receiving a packet when A initiates RTS
- Procedure
  - A sends RTS to B
  - C is sending busy tone (as it receives data)
  - CTS and busy tone collide, A receives no CTS, does not send data





Similarly: Ongoing transmission near B destroys RTS by busy tone

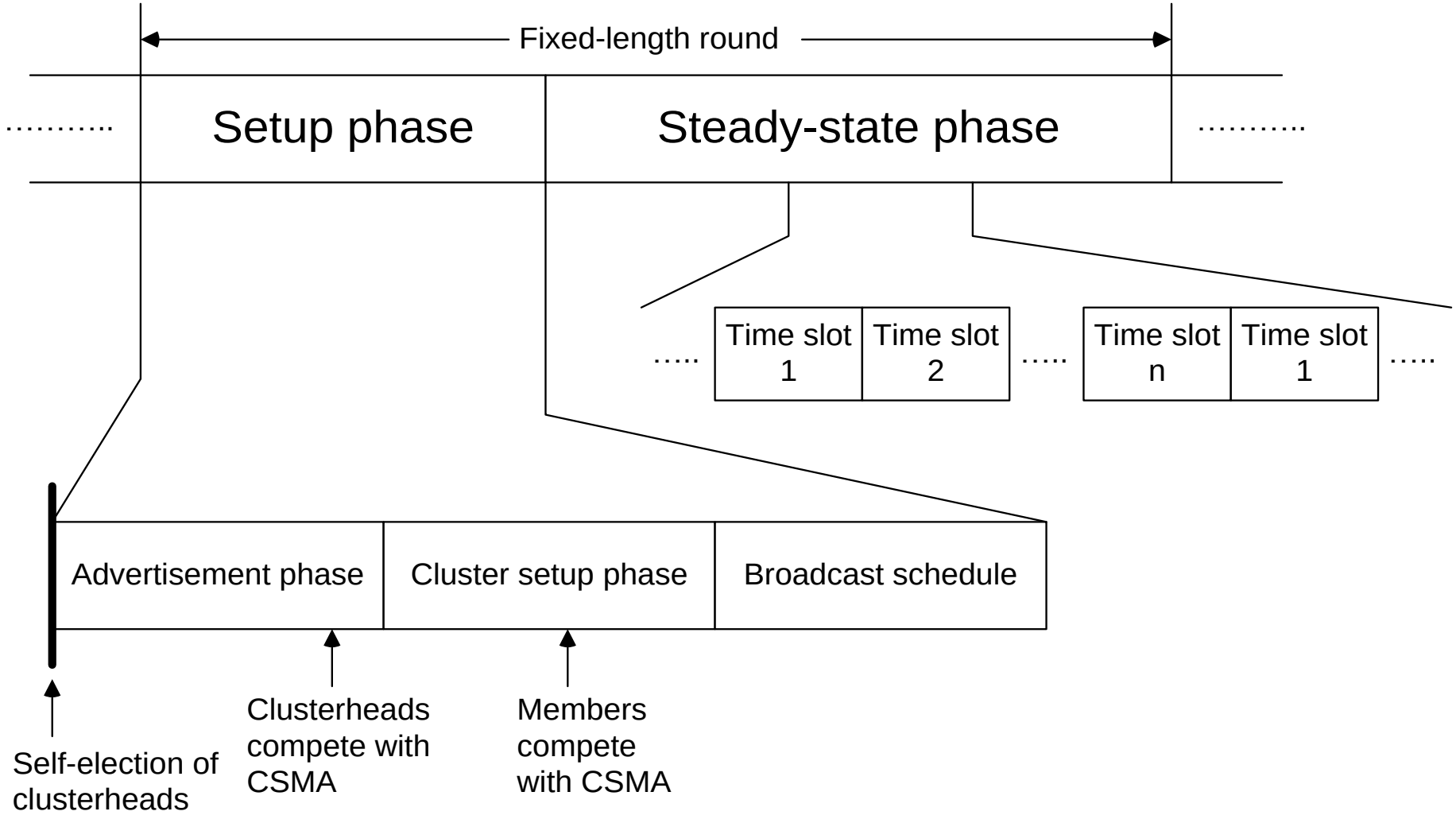
Overview

- Principal options and difficulties
- Contention-based protocols
- ***Schedule-based protocols***
  - LEACH
  - SMACS
  - TRAMA
- IEEE 802.15.4

Low-Energy Adaptive Clustering Hierarchy (LEACH)

- Given: dense network of nodes, reporting to a central sink, each node can reach sink directly
- Idea: Group nodes into "***clusters***", controlled by ***clusterhead***
  - Setup phase; details: later
  - About 5% of nodes become clusterhead (depends on scenario)
  - Role of clusterhead is rotated to share the burden
  - Clusterheads advertise themselves, ordinary nodes join CH with strongest signal
  - Clusterheads organize
    - CDMA code for all member transmissions
    - TDMA schedule to be used within a cluster
- In steady state operation
  - CHs collect & aggregate data from all cluster members
  - Report aggregated data to sink using CDMA

LEACH rounds

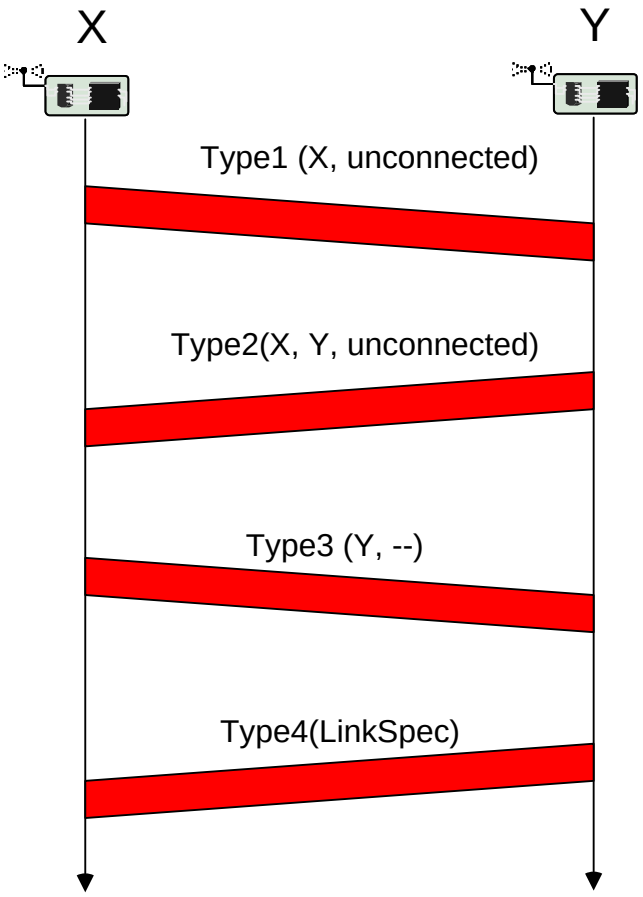


SMACS

- Given: many radio channels, superframes of known length (not necessarily in phase, but still time synchronization required!)
- Goal: set up directional ***links*** between neighboring nodes
  - Link: radio channel + time slot at both sender and receiver
  - Free of collisions at receiver
  - Channel picked randomly, slot is searched greedily until a collision-free slot is found
- Receivers sleep and only wake up in their assigned time slots, once per superframe
- In effect: a local construction of a schedule

SMACS link setup

- Case 1: Node X, Y both so far unconnected
  - Node X sends invitation message
  - Node Y answers, telling X that is unconnected to any other node
  - Node X tells Y to pick slot/frequency for the link
  - Node Y sends back the link specification
- Case 2: X has some neighbors, Y not
  - Node X will construct link specification and instruct Y to use it (since Y is unattached)
- Case 3: X no neighbors, Y has some
  - Y picks link specification
- Case 4: both nodes already have links
  - Nodes exchange their schedules and pick free slots/frequencies in mutual agreement



Message exchanges protected by randomized backoff

TRAMA

- Nodes are synchronized
- Time divided into cycles, divided into
  - Random access periods
  - Scheduled access periods
- Nodes exchange neighborhood information
  - Learning about their two-hop neighborhood
  - Using ***neighborhood exchange protocol***: In random access period, send small, incremental neighborhood update information in randomly selected time slots
- Nodes exchange schedules
  - Using ***schedule exchange protocol***
  - Similar to neighborhood exchange

TRAMA - adaptive election

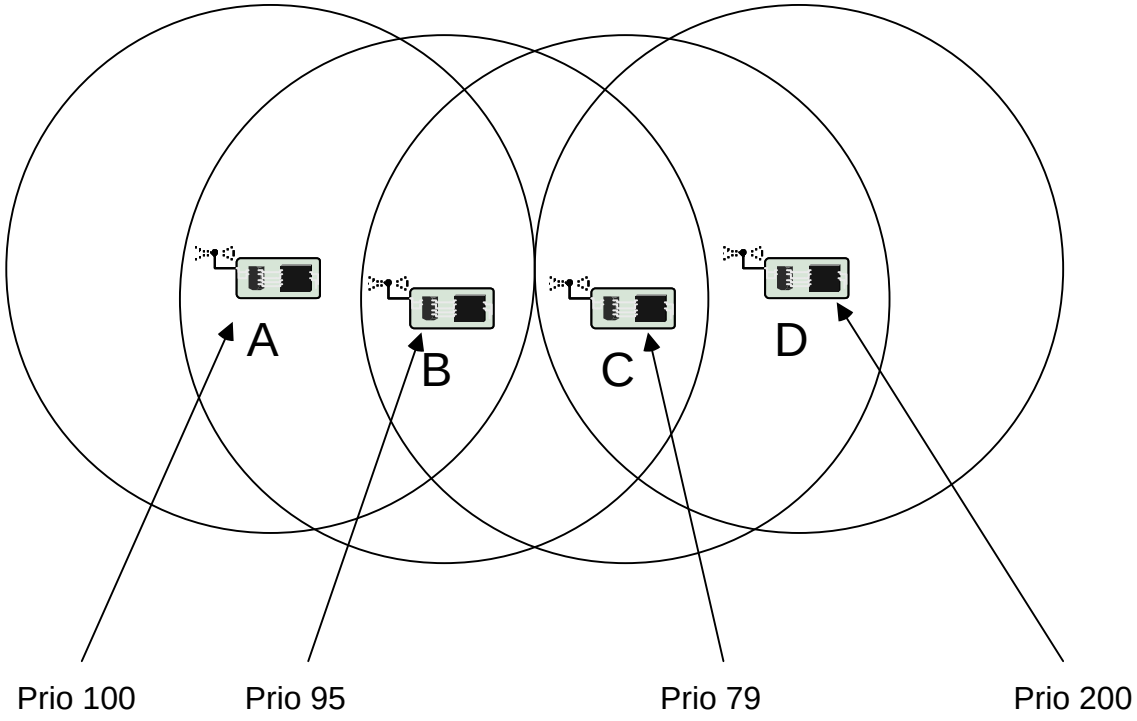
- Given: Each node knows its two-hop neighborhood and their current schedules
- How to decide which slot (in scheduled access period) a node can use?
  - Use ***node identifier*** x and globally known ***hash function*** h
  - For time slot t, compute ***priority*** p = h (x © t)
  - Compute this priority for next k time slots for node itself and all two-hop neighbors
  - Node uses those time slots for which it has the highest priority

Priorities of node A and its two neighbors B & C

	t = 0	t = 1	t = 2	t=3	t = 4	t = 5
A	14	23	9	56	3	26
B	33	64	8	12	44	6
C	53	18	6	33	57	2

TRAMA - possible conflicts

- When does a node have to receive?
  - Easy case: one-hop neighbor has won a time slot and announced a packet for it
  - But complications exist – compare example
- What does B believe?
  - A thinks it can send
  - B knows that D has higher priority in its 2-hop neighborhood!
- Rules for resolving such conflicts are part of TRAMA



Comparison: TRAMA, S-MAC

- Comparison between TRAMA & S-MAC
  - Energy savings in TRAMA depend on load situation
  - Energy savings in S-MAC depend on duty cycle
  - TRAMA (as typical for a TDMA scheme) has higher delay but higher maximum throughput than contention-based S-MAC

- TRAMA disadvantage: substantial memory/CPU requirements for schedule computation

Overview

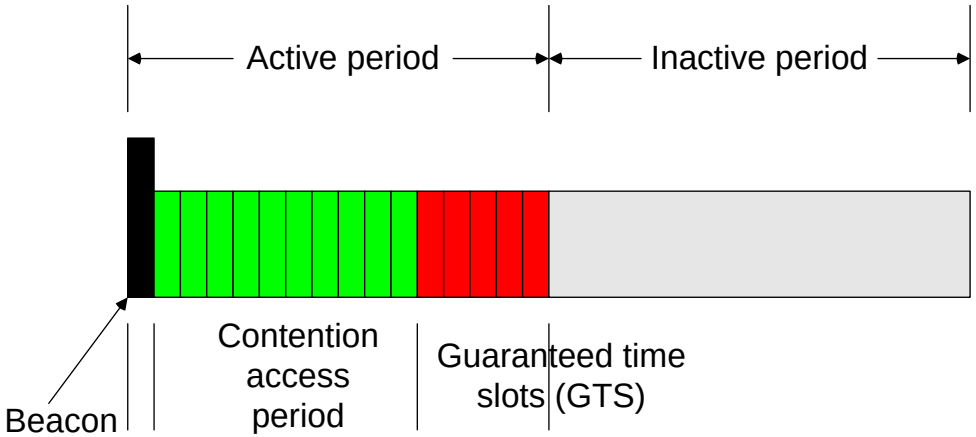
- Principal options and difficulties
- Contention-based protocols
- Schedule-based protocols
- ***IEEE 802.15.4***

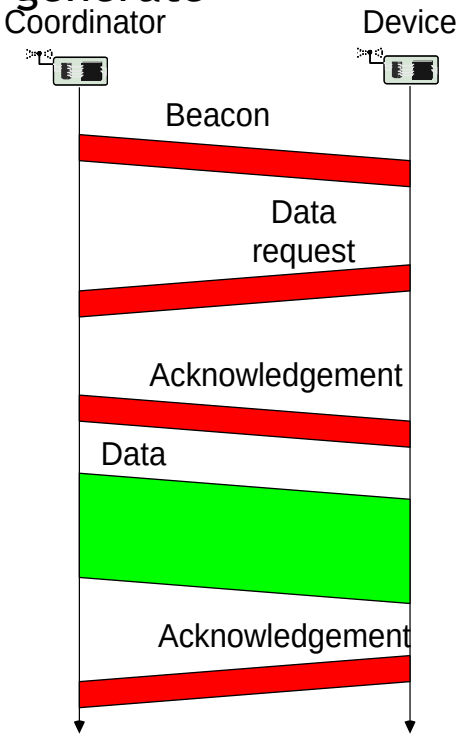
IEEE 802.15.4

- IEEE standard for low-rate WPAN applications
- Goals: low-to-medium bit rates, moderate delays without too stringent guarantee requirements, low energy consumption
- Physical layer
  - 20 kbps over 1 channel @ 868-868.6 MHz
  - 40 kbps over 10 channels @ 905 – 928 MHz
  - 250 kbps over 16 channels @ 2.4 GHz
- MAC protocol
  - Single channel at any one time
  - Combines contention-based and schedule-based schemes
  - Asymmetric: nodes can assume different roles

IEEE 802.15.4 MAC overview

- Star networks: ***devices*** are associated with ***coordinators***
  - Forming a PAN, identified by a PAN identifier
- Coordinator
  - Bookkeeping of devices, address assignment, generate beacons
  - Talks to devices and peer coordinators
- Beacon-mode superframe structure
  - GTS assigned to devices upon request





Wakeup radio MAC protocols

- Simplest scheme: Send a wakeup “burst”, waking up all neighbors ! Significant overhearing
  - Possible option: First send a short ***filter packet*** that includes the actual destination address to allow nodes to power off quickly
- Not quite so simple scheme: Send a wakeup burst including the receiver address
  - Wakeup radio needs to support this option
- Additionally: Send information about a (randomly chosen) data channel, CDMA code, … in the wakeup burst
- Various variations on these schemes in the literature, various further problems
  - One problem: 2-hop neighborhood on wakeup channel might be different from 2-hop neighborhood on data channel
  - Not trivial to guarantee unique addresses on both channels

Further protocols

- MAC protocols for ad hoc/sensor networks is one the most active research fields
  - Tons of additional protocols in the literature
  - Examples: STEM, mediation device protocol, many CSMA variants with different timing optimizations, protocols for multi-hop reservations (QoS for MANET), protocols for multiple radio channels, …
  - Additional problems, e.g., reliable multicast

- This chapter has barely scratched the surface…

Summary

- Many different ideas exist for medium access control in MANET/WSN
- Comparing their performance and suitability is difficult
- Especially: clearly identifying interdependencies between MAC protocol and other layers/applications is difficult
  - Which is the best MAC for which application?

- Nonetheless, certain “common use cases” exist
  - IEEE 802.11 DCF for MANET
  - IEEE 802.15.4 for some early “commerical” WSN variants
  - B-MAC for WSN research not focusing on MAC

LINK LAYER OF WSN

Overview

- ***Error control***
- Framing
- Link management

Error control

- Error control has to ensure that data transport is
  - Error-free – deliver exactly the sent bits/packets
  - In-sequence – deliver them in the original order
  - Duplicate-free – and at most once
  - Loss-free – and at least once
- Causes: fading, interference, loss of bit synchronization, …
  - Results in bit errors, bursty, sometimes heavy-tailed runs (see physical layer chapter)
  - In wireless, sometimes quite high average bit error rates – 10^{-2} … 10^{-4} possible!
- Approaches
  - Backward error control – ARQ
  - Forward error control – FEC

Backward error control - ARQ

- Basic procedure (a quick recap)
  - Put header information around the payload
  - Compute a checksum and add it to the packet
    - Typically: Cyclic redundancy check (CRC), quick, low overhead, low residual error rate
  - Provide feedback from receiver to sender
    - Send ***positive*** or ***negative acknowledgement***
  - Sender uses timer to detect that acknowledgements have not arrived
    - Assumes packet has not arrived
    - Optimal timer setting?
  - If sender infers that a packet has not been received correctly, sender can retransmit it
    - What is maximum number of retransmission attempts? If bounded, at best a semi-reliable protocols results

Standard ARQ protocols

- Alternating bit – at most one packet outstanding, single bit sequence number
- Go-back N – send up to N packets, if a packet has not been acknowledged when timer goes off, retransmit all unacknowledged packets
- Selective Repeat – when timer goes off, only send that particular packet

How to use acknowledgements

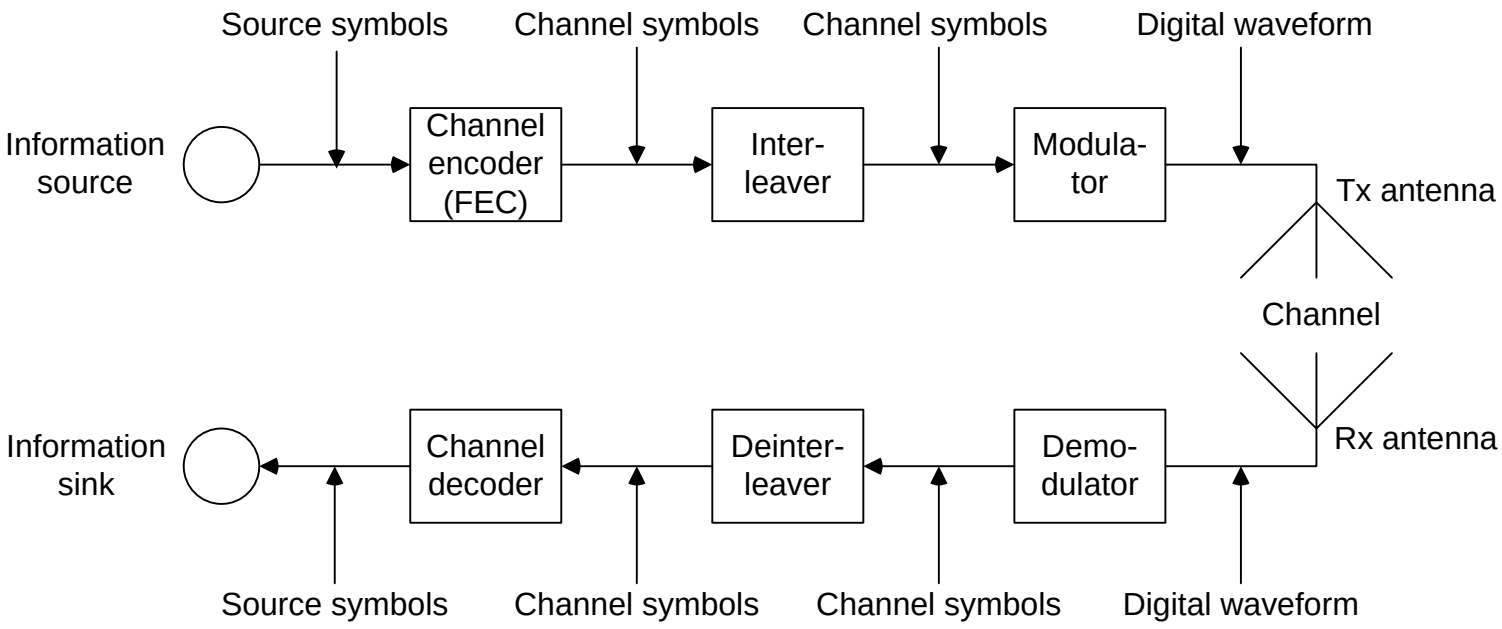
- Be careful about ACKs from different layers
  - A MAC ACK (e.g., S-MAC) does not necessarily imply buffer space in the link layer
  - On the other hand, having both MAC and link layer ACKs is a waste
- Do not (necessarily) acknowledge every packet – use cumulative ACKs
  - Tradeoff against buffer space
  - Tradeoff against number of negative ACKs to send

When to retransmit

- Assuming sender has decided to retransmit a packet – when to do so?
  - In a BSC channel, any time is as good as any
  - In fading channels, try to avoid bad channel states – postpone transmissions
  - Instead (e.g.): send a packet to another node if in queue (exploit multi-user diversity)
- How long to wait?
  - Example solution: Probing protocol
  - Idea: reflect channel state by two protocol modes, “normal” and “probing”
  - When error occurs, go from normal to probing mode
  - In probing mode, periodically send short packets (acknowledged by receiver) – when successful, go to normal mode

Forward error control

- Idea: Endow symbols in a packet with additional redundancy to withstand a limited amount of random permutations
  - Additionally: interleaving – change order of symbols to withstand burst errors



Error-Correction Code

Hamming code

- Theoretical lower limit
  - A code with m message bits and r check bits which will allow all single errors to be corrected
    - 2^m legal messages – legal codewords
    - For each legal message, there is n illegal codewords
    - Total number of bit pattern is 2^n
    - Therefore, $(n+1)2^m \leq 2^n$
    - $n=m+r$, $(m+r+1) \leq 2^r$
- Hamming Code
  - It can correct any single bit error
  - bits in power of 2 positions (1,2,4,8,) are check bits, the rest are m data bits
  - Check bit forces the parity of some collection of bit “1” to be even
  - To see which check bits the data bit in position k contributes to, rewrite k as a sum of powers of 2.
    - 11=1+2+8, 29=1+4+8+16

Hamming Code Example

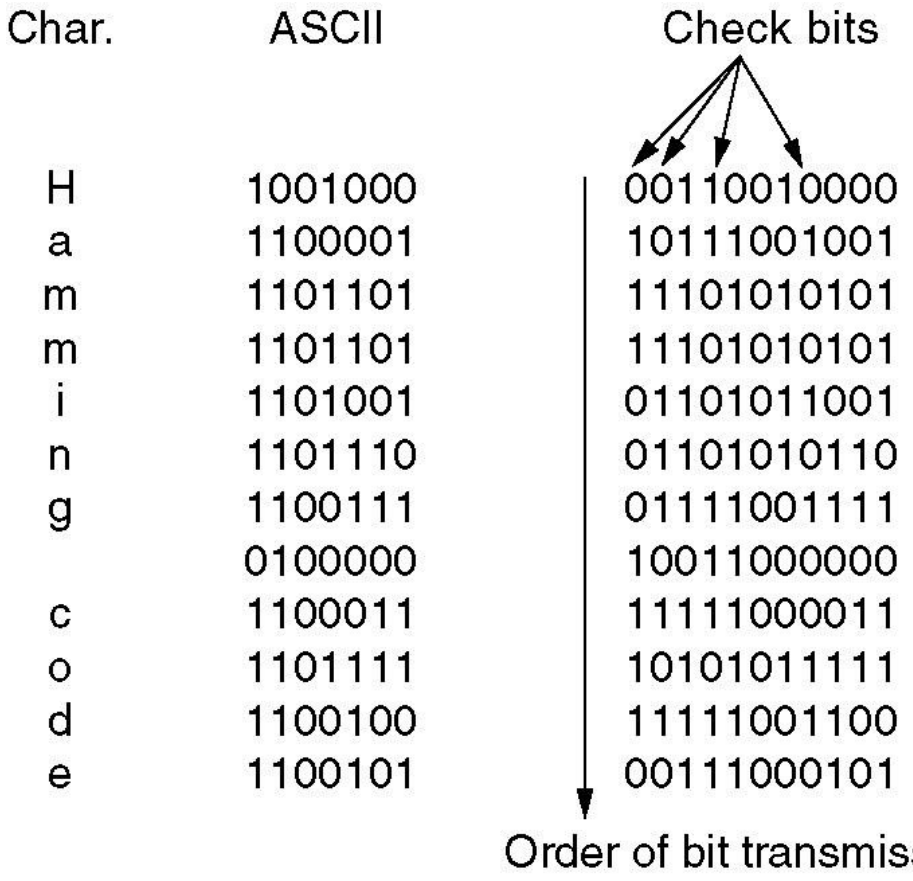
If a 12-bit hamming code 0x9B8 = 100110111000 arrives at receiver, is there any bit error? If so, which bit is wrong?

1	2	3=2+1	4	5=4+1	6=4+2	7=4+2+1	8	9=8+1	10=8+2	11=8+2+1	12=8+4
1	0	0	1	1	0	1	1	1	0	0	0

Check bit	Checked Data bits	Bits Sequence
1	3 5 7 9 11	1 0 1 1 1 0 (Yes)
2	3 6 7 10 11	0 0 0 1 0 0 (No)
4	5 6 7 12	1 1 0 1 0 (No)
8	9 10 11 12	1 1 0 0 0 (Yes)

Hence bit 6 = 2+4 is wrong, so correct hamming code should be 0x9F8 = 100111111000

Correct burst errors

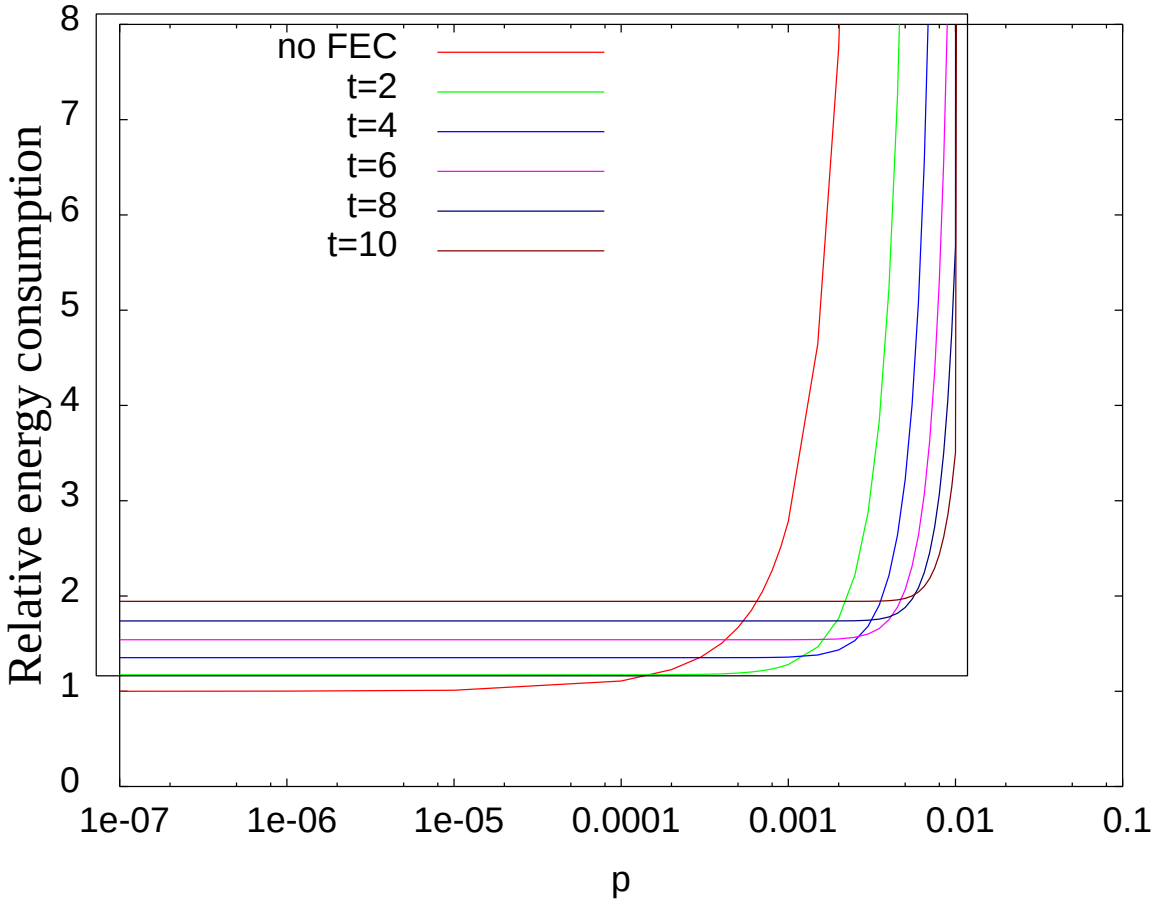


Use of Hamming code to correct k burst errors by send k ASCII character together, column by column. (Read Tanenbaum textbook page 195)

Comparison: FEC vs. ARQ

- FEC
  - Constant overhead for each packet
  - Not (easily) possible to adapt to changing channel characteristics
- ARQ
  - Overhead only when errors occurred (expect for ACK, always needed)
- Both schemes have their uses ! ***hybrid schemes***

t: error correction capacity



BCH + unlimited number of retransmissions

Power control on a link level

- Further controllable parameter: transmission power
  - Higher power, lower error rates – less FEC/ARQ necessary
  - Lower power, higher error rates – higher FEC necessary
- Tradeoff!

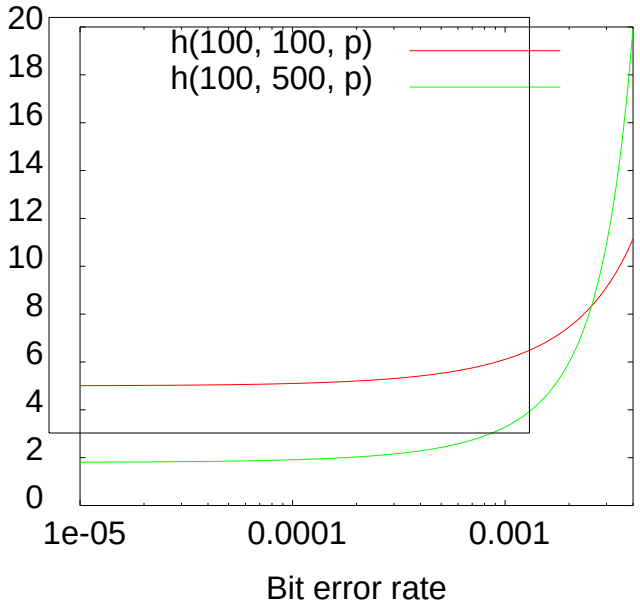
Overview

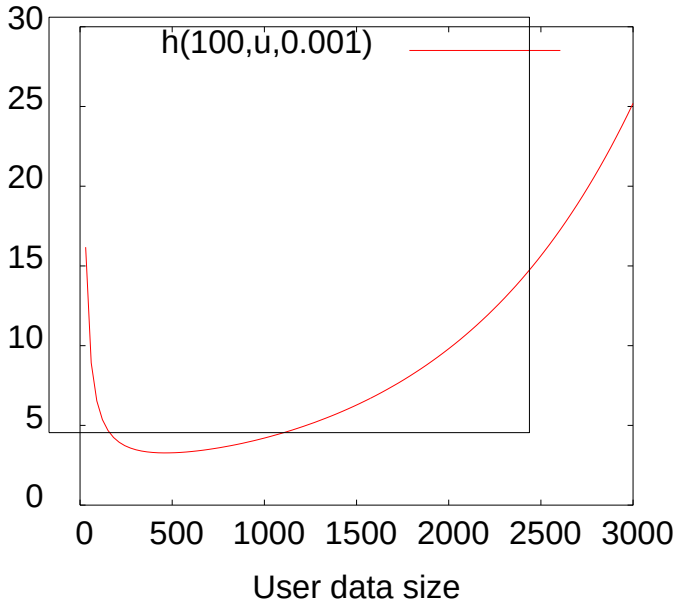
- Error control
- ***Framing***
- Link management

Frame, packet size

- Small packets: low packet error rate, high packetization overhead
- Large packets: high packet error rate, low overhead
- Depends on bit error rate, energy consumption per transmitted bit

- Notation: h(overhead, payload size, BER)



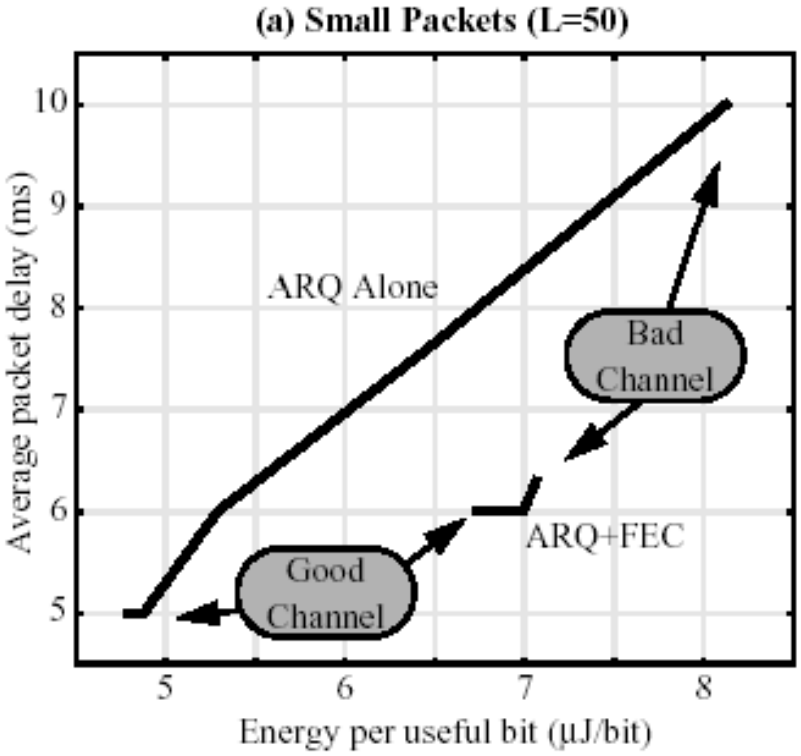


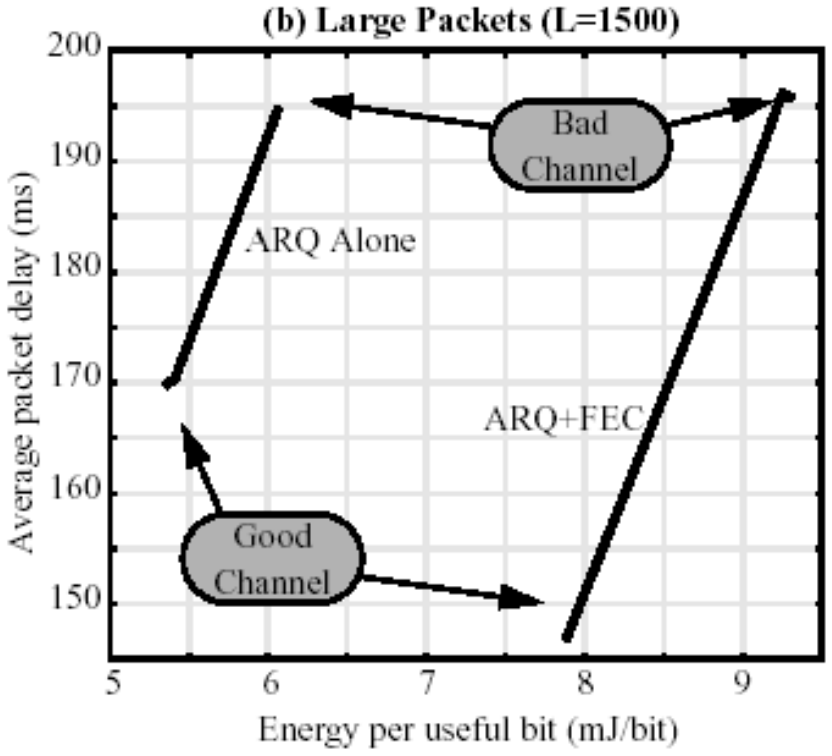
Dynamically adapt frame length

- For known bit error rate (BER), optimal frame length is easy to determine
- Problem: how to estimate BER?
  - Collect channel state information at the receiver (RSSI, FEC decoder information, …)
  - Example: Use number of attempts T required to transmit the last M packets as an estimator of the packet error rate (assuming a BSC)
- Second problem: how long are observations valid/how should they be aged?
  - Only recent past is – if anything at all – somewhat credible

Putting it together: ARQ, FEC, frame length optimization

- Applying ARQ, FEC (both block and convolutional codes), frame length optimization to a Rayleigh fading channel
  - Channel modeled as Gilbert-Elliot





Overview

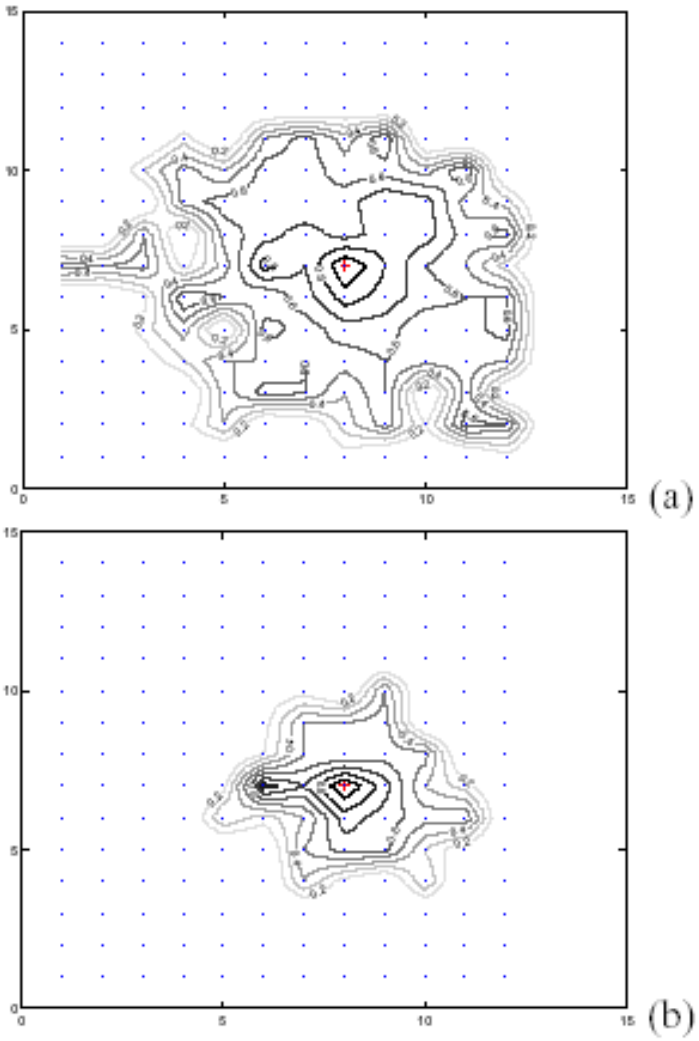
- Error control
- Framing
- ***Link management***

Link management

- Goal: decide to which neighbors that are *more or less* reachable a link should be established
  - Problem: communication quality fluctuates, far away neighbors can be costly to talk to, error-prone, quality can only be estimated
- Establish a ***neighborhood table*** for each node
  - Partially automatically constructed by MAC protocols

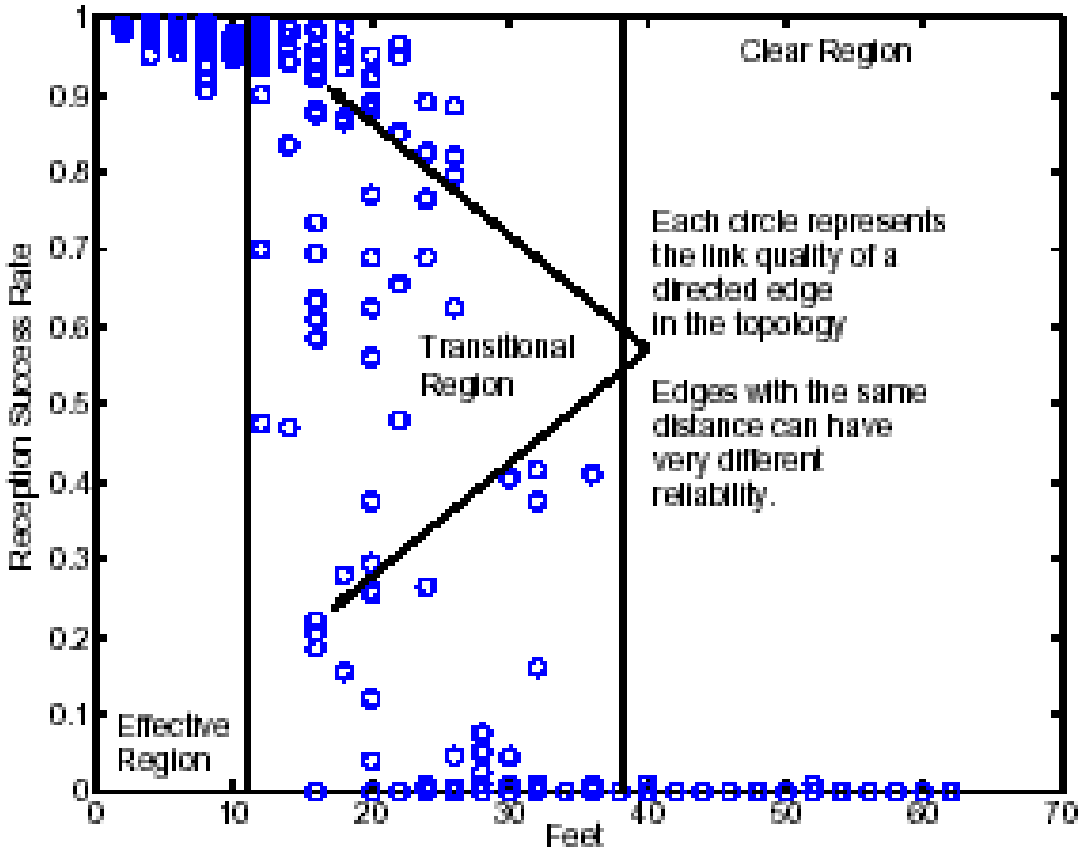
Link quality characteristics

- Expected: simple, circular shape of “region of communication” – not realistic
- Instead:
  - Correlation between distance and loss rate is weak; iso-loss-lines are not circular but irregular
  - Asymmetric links are relatively frequent (up to 15%)
  - Significant short-term PER variations even for stationary nodes



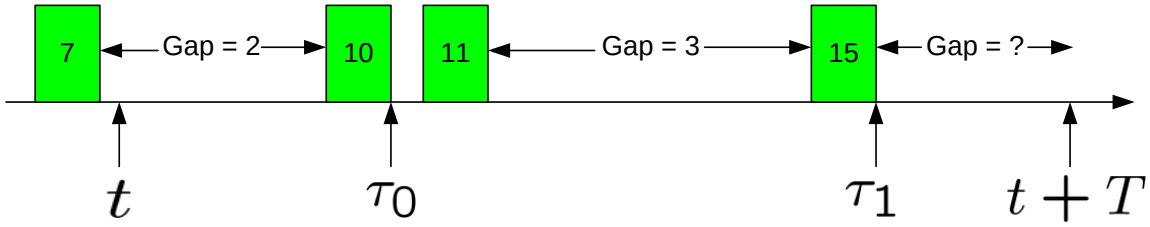
Three regions of communication

- ***Effective region***: PER consistently < 10%
- ***Transitional region:*** anything in between, with large variation for nodes at same distance
- ***Poor region***: PER well beyond 90%



Link quality estimation

- How to estimate, on-line, in the field, the actual link quality?
- Requirements
  - Precision – estimator should give the statistically correct result
  - Agility – estimator should react quickly to changes
  - Stability – estimator should not be influenced by short aberrations
  - Efficiency – Active or passive estimator



- Example: WMEWMA only estimates at fixed intervals

$$P_n = \alpha P_{n-1} + (1-\alpha)\frac{r_n}{r_n + f_n}$$

r_n: received packets in interval

f_n: packets identified as lost

Conclusion

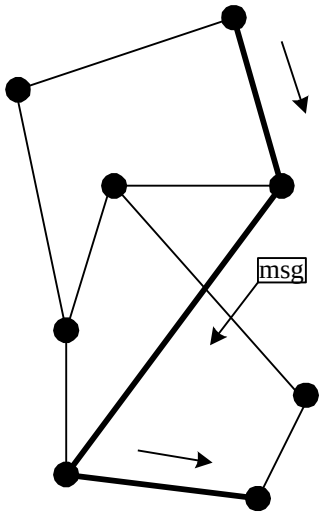
- Link layer combines traditional mechanisms
  - Framing, packet synchronization, flow control

  with relatively specific issues
  - Careful choice of error control mechanisms – tradeoffs between FEC & ARQ & transmission power & packet size …
  - Link estimation and characterization

WSN ROUTING

Routing Overview

- Network with nodes, edges
- In any network of diameter > 1, the routing & forwarding problem appears
- Goal: Devise scheme for transferring message from one node to another
  - Which path?
  - Who decides – source or intermediate nodes?

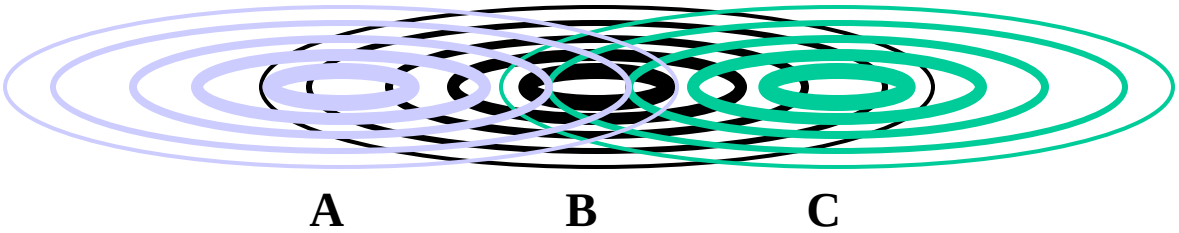


Characteristics of Ad Hoc Communications

- Characteristics are dominated by heterogeneity and variability
  - Mobility characteristics (speed, predictability, uniformity, synthetic vs. empirical models , …)
  - Wireless characteristics (broadcast nature of the net, packet losses due to transmission errors, limited range, hidden and exposed terminals, partitioning)
  - Application / traffic characteristics and patterns (P2P, real time, unicast, multicast, geocast, CBR, VBR, self-similar, …)
  - System characteristics (distribution, absence of infrastructure, (unpredictable) high dynamics, (a)symmetry …)

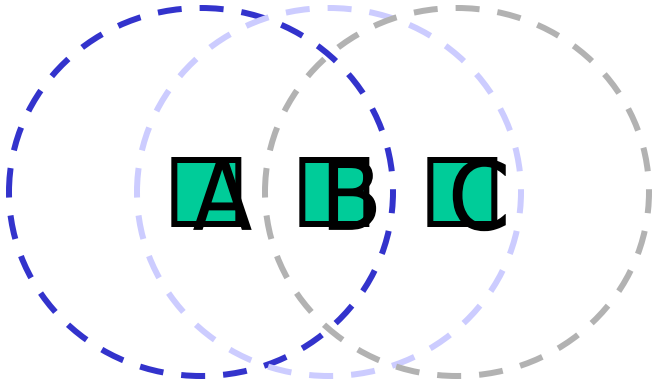
Hidden and Exposed Terminals

- Hidden terminals
  - A sends to B, C cannot receive A
  - C wants to send to B, C senses a “free” medium
  - collision at B, A cannot receive the collision
  - A is “hidden” for C



- Exposed terminals
  - B sends to A, C wants to send to another terminal (not A or B)
  - C senses carrier, finds medium in use and has to wait
  - A is outside the radio range of C, therefore waiting is not necessary
  - C is “exposed” to B

Why Specialized Ad Hoc Routing



In WANETs

- Some nodes may be out of range of others
- Must use other peer nodes as routers to forward packets
- Need to find new routes as nodes move or conditions change (highly dynamic and unpredictable)
- Routing protocol captures and distributes state of network
- Routing strategy (algorithm) computes shortest paths

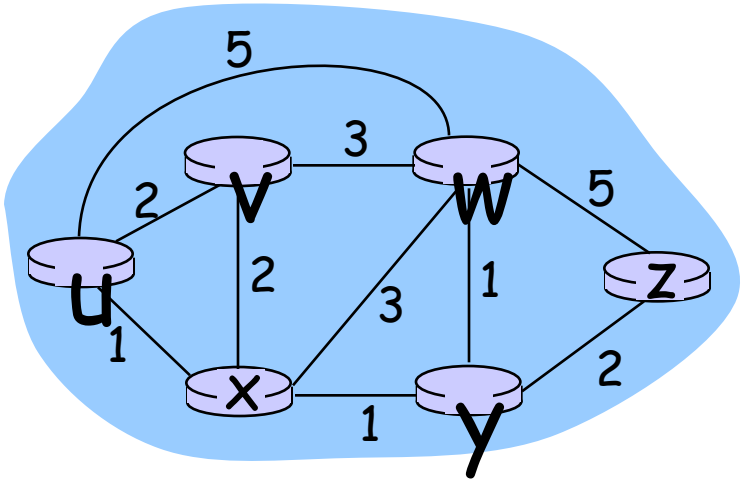
Requirements for Ad Hoc Routing

- The routing protocol needs to
  - Converge fast
  - Minimize signaling overhead
- The routing strategy (algorithm) may include
  - Shortest distance
  - Minimum delay
  - Minimum loss
  - Minimum congestion (load-balancing)
  - Minimal interference
  - Maximum stability of routes or maximal signal strength
  - Minimum energy (power aware routing)
- Standard Internet routing cannot fulfill these requirements
  - Assumes infrastructure, assumes symmetrical conditions, assumes plenty of resources, misses metrics, …

Review of Internet Routing

- Intra-AS (autonomous systems) Routing
  - OSPF: Open Shortest Path First
    - Link State algorithm (Global)
      - LS packet dissemination (entire network)
      - Complete topology map at each node
      - Route computation using Dijkstra's algorithm
  - RIP: Routing Information Protocol
    - Distance Vector algorithm (Decentralized)
      - Router knows physically-connected neighbors, link costs to neighbors
      - Distance vectors: exchanged among neighbors
      - Iterative process of computation, exchange of info with neighbors

Graph abstraction



Graph: G = (N,E)

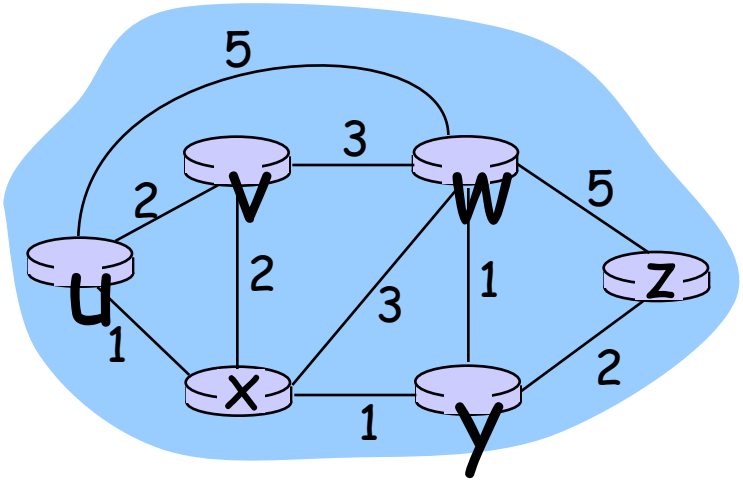
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



- c(x,x') = cost of link (x,x')

  - e.g., c(w,z) = 5

- cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3, \ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

Routing algorithm: algorithm that finds least-cost path

A Link-State Routing Algorithm

Dijkstra's algorithm

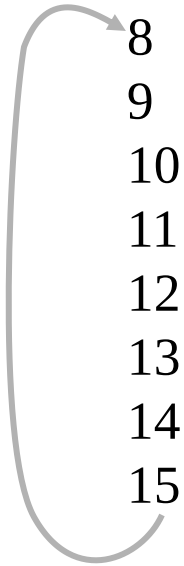
- net topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k dest.'s

Notation:

- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

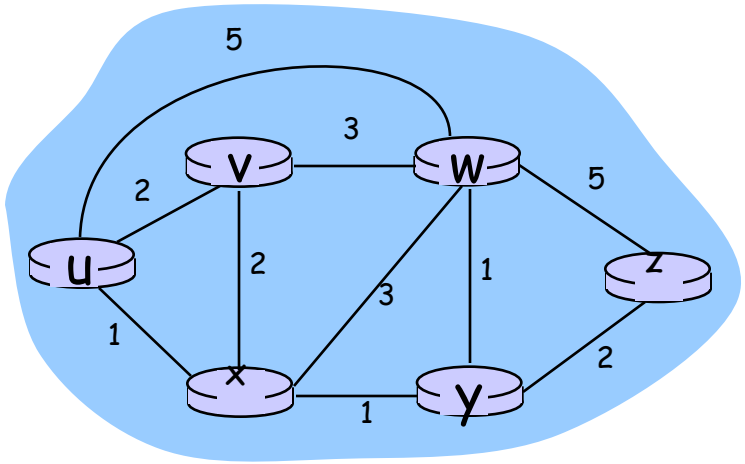
Dijsktra's Algorithm

```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
7
8  Loop
9     find w not in N' such that D(w) is a minimum
10    add w to N'
11    update D(v) for all v adjacent to w and not in N' :
12       D(v) = min( D(v), D(w) + c(w,v) )
13    /* new cost to v is either old cost to v or known
14     shortest path cost to w plus cost from w to v */
15  until all nodes in N'
```



Dijkstra's algorithm: example

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	4,y
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					
5	uxyvwz					



Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

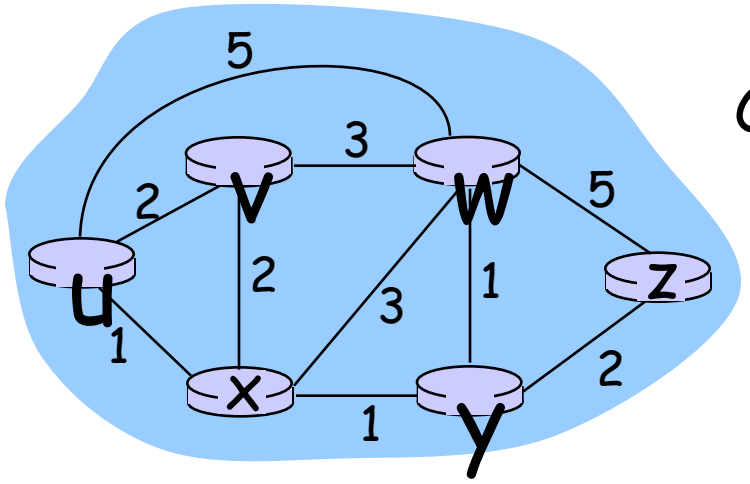
$d_x(y)$:= cost of least-cost path from x to y

Then

$$d_x(y) = \min \{c(x,v) + d_v(y) \}$$

where min is taken over all neighbors of x

Bellman-Ford example



Clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$
$$= \min \{2 + 5, \; 1 + 3, \; 5 + 3\} = 4$$

Node that achieves minimum is next hop in shortest path ➜ forwarding table

Distance Vector Algorithm

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x knows cost to each neighbor v: $c(x,v)$
- Node x maintains $\mathbf{D}_x = [D_x(y): y \in N]$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm

Basic idea:

- Each node periodically sends its own distance vector estimate to neighbors
- When node a node x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow min_v\{c(x,v) + D_v(y)\} \quad for\ each\ node\ y \in N$$

- Under minor, natural conditions, the estimate $D_x(y)$ *converge the actual least cost* $d_x(y)$

Distance Vector Algorithm

Iterative, asynchronous:

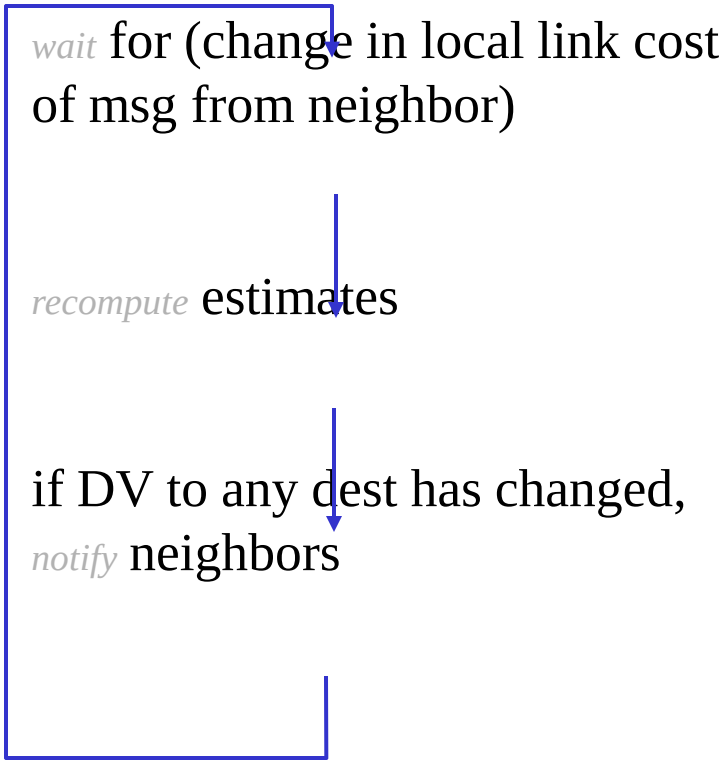
each local iteration caused by:

- local link cost change
- DV update message from neighbor

Distributed:

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

Each node:



$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$
$= \min\{2+0\ ,\ 7+1\} = 2$

$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$
$= \min\{2+1\ ,\ 7+0\} = 3$

node x table

from	cost to x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from	cost to x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

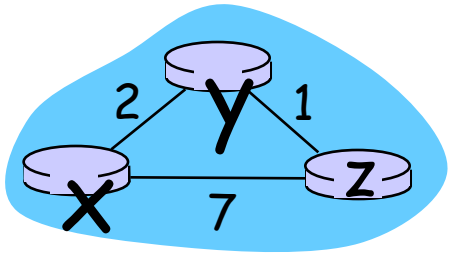
from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from	cost to x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from	cost to x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from	cost to x	y	z
x	0	2	3
y	2	0	1
z	3	1	0



time

Ad Hoc Routing

- Simple solution: Flooding of data packets
  - Does not need any information (routing tables) – simple
  - Packets are usually delivered to destination
  - But: extremely high overhead, usually not acceptable
  - Many protocols perform (limited) flooding of control packets
    - To discover routes
    - Overhead of control packet flooding is amortized over data packets transmitted between consecutive control packet floods
  - Modified flooding based ad hoc broadcast routing

Ad Hoc Routing - Classification

- Uniform Protocols vs Non-Uniform Protocols
  - All nodes are equal
  - Some nodes have special roles (e.g. Clusterhead, Gateway-node)
- Flat Routing vs. Hierarchical/Clustered Routing
  - Network has no hierarchy
  - Trying to structure/cluster the network
- Proactive (table-driven) vs Reactive (on-demand)
  - Trade-off latency vs overhead
  - Hybrid (combination)
- Source Routing vs Destination Routing
- Other:
  - Geographical/Position-based routing
  - Power efficient routing
  - Multipath routing
  - ……

Who determines route?

- Source (“path”) routing
  - Source specifies entire route: places complete path to destination in message header: A – D – F – G
  - Intermediate nodes just forward to specified next hop: D would look at path in header, forward to F
  - Like airline travel – get complete set of tickets to final destination before departing…

Who determines route?

- Destination ("hop-by-hop") routing
  - Source specifies only destination in message header: G
  - Intermediate nodes look at destination in header, consult internal tables to determine appropriate next hop
  - Like postal service – specify only the final destination on an envelope, and intermediate post offices select where to forward next…

Comparison

- Source routing
  - Moderate source storage (entire route for each desired dest.)
  - No intermediate node storage
  - Higher routing overhead (entire path in message header, route discovery messages)
- Destination routing
  - No source storage
  - High intermediate node storage (table w/ routing instructions for all possible dests.)
  - Lower routing overhead (just dest in header, only routers need deal w/ route discovery)

When does routing operate ?

- Main question to ask: ***When*** does the routing protocol operate?

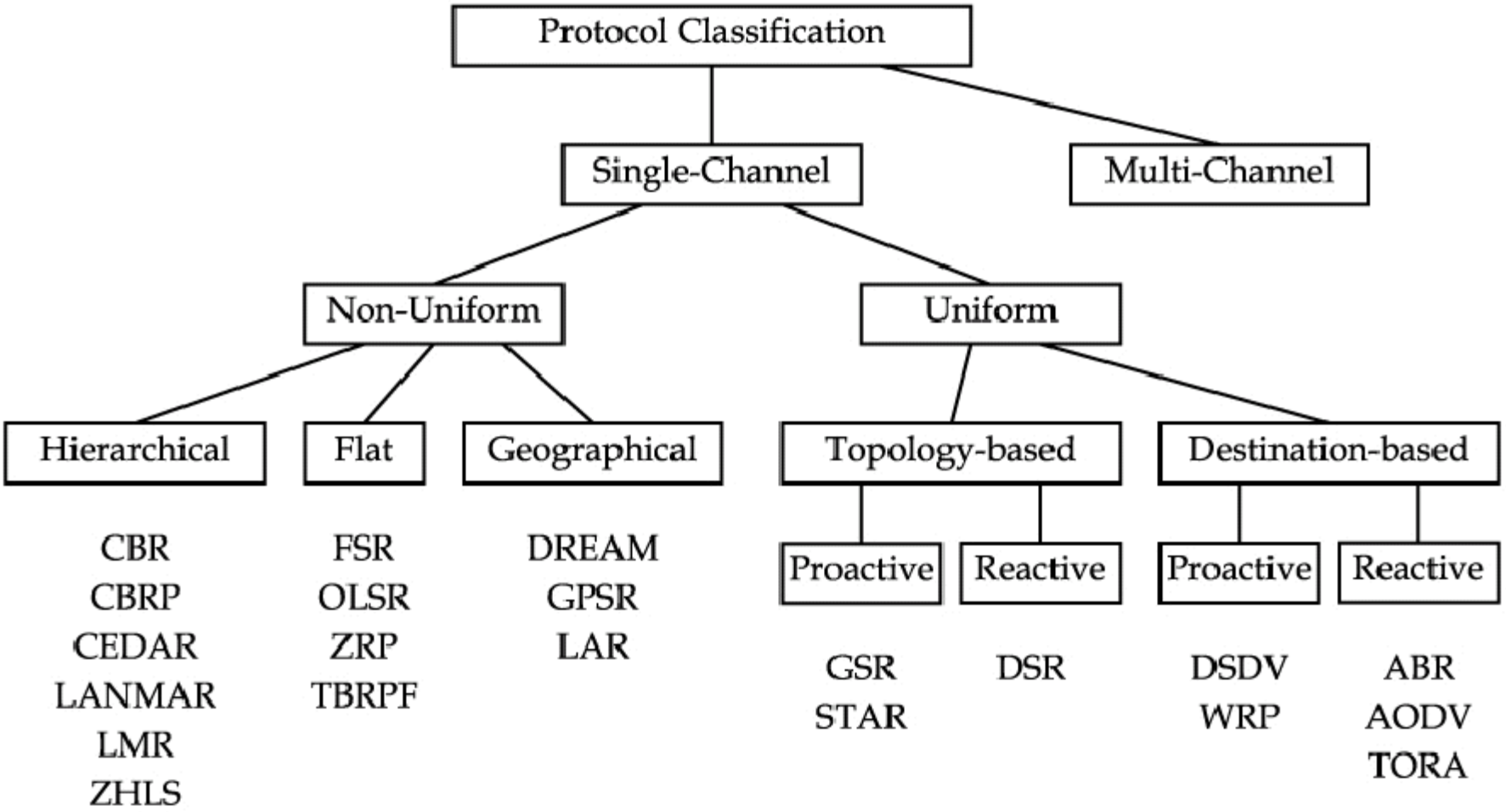
- Option 1: Routing protocol ***always*** tries to keep its routing data up-to-date
  - Protocol is ***proactive*** (active before tables are actually needed) or ***table-driven***
- Option 2: Route is only determined when actually needed
  - Protocol operates ***on demand or reactive***
- Option 3: Combine these behaviors
  - ***Hybrid*** protocols

Proactive vs. Reactive

- Proactive routing maintains routes to every other node in the network
- Table driven
- Regular routing updates impose large overhead
- No latency in route discovery, i.e., data can be sent immediately
- Most routing information might never be used
- Suitable for high traffic networks
- Bellman-Ford type algorithms

- Reactive routing maintains routes to only those nodes which are needed
- Cost of finding routes is expensive since flooding is involved
- Might be delay before transmitting data
- Cache information from other nodes' transmissions
- May not be appropriate for real-time applications
- Good for low/medium traffic networks

Taxonomy of Routing Protocols



Routing in Ad Hoc

- Lots and lots of protocols suggested
- Sample Protocols
  - **Table Driven / Proactive**: DSDV
  - **On-Demand-Driven Reactive**: AODV, DSR
  - **Hybrid**: ZRP
  - **Geographic Routing**: GPSR, LAR
  - **Hierarchical Routing**: CBRP

Destination Sequenced Distance Vector

DSDV

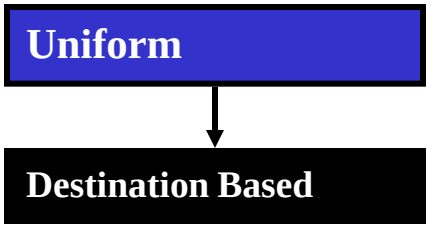
C. Perkins and P. Bhagwat
IBM and U of Maryland

"Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers“, ACM SIGCOMM, 1994.

Introduction

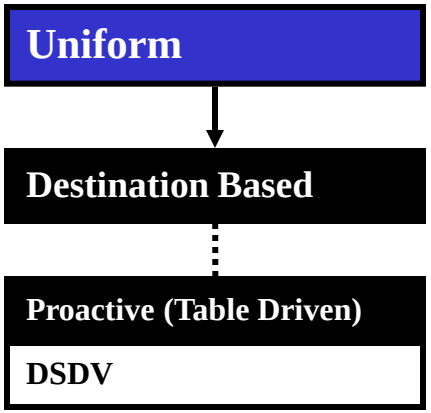
- DSDV is Uniform
  - Each node sends and responds to routing control message the same way
  - No hierarchical structure
  - Avoids the resource costs involved in maintaining high-level structure
  - Scalability may become an issue in larger networks

Introduction



- DSDV is Destination Based
  - Nodes maintain only local topology information
    (e.g. 1 or 2-hop neighborhood)
  - No global view of topology
  - Possible inconsistencies

Introduction

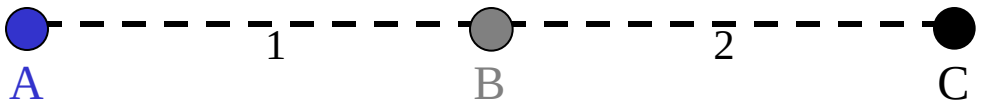


- DSDV is Proactive (Table Driven)
  - Each node maintains routing information for all known destinations
  - Routing information must be updated periodically (no sleeping nodes)
  - Traffic overhead even if there is no change in network topology
  - Maintains routes which are never used

Distance Vector

- Basic Routing Protocol
  - known also as Distributed Bellman-Ford or RIP
- Every node maintains a routing table
  - all available destinations
  - the next node to reach to destination
  - the number of hops to reach the destination
- Periodically send table to all neighbors to maintain topology
- Bi-directional links are required!

Distance Vector (Tables)

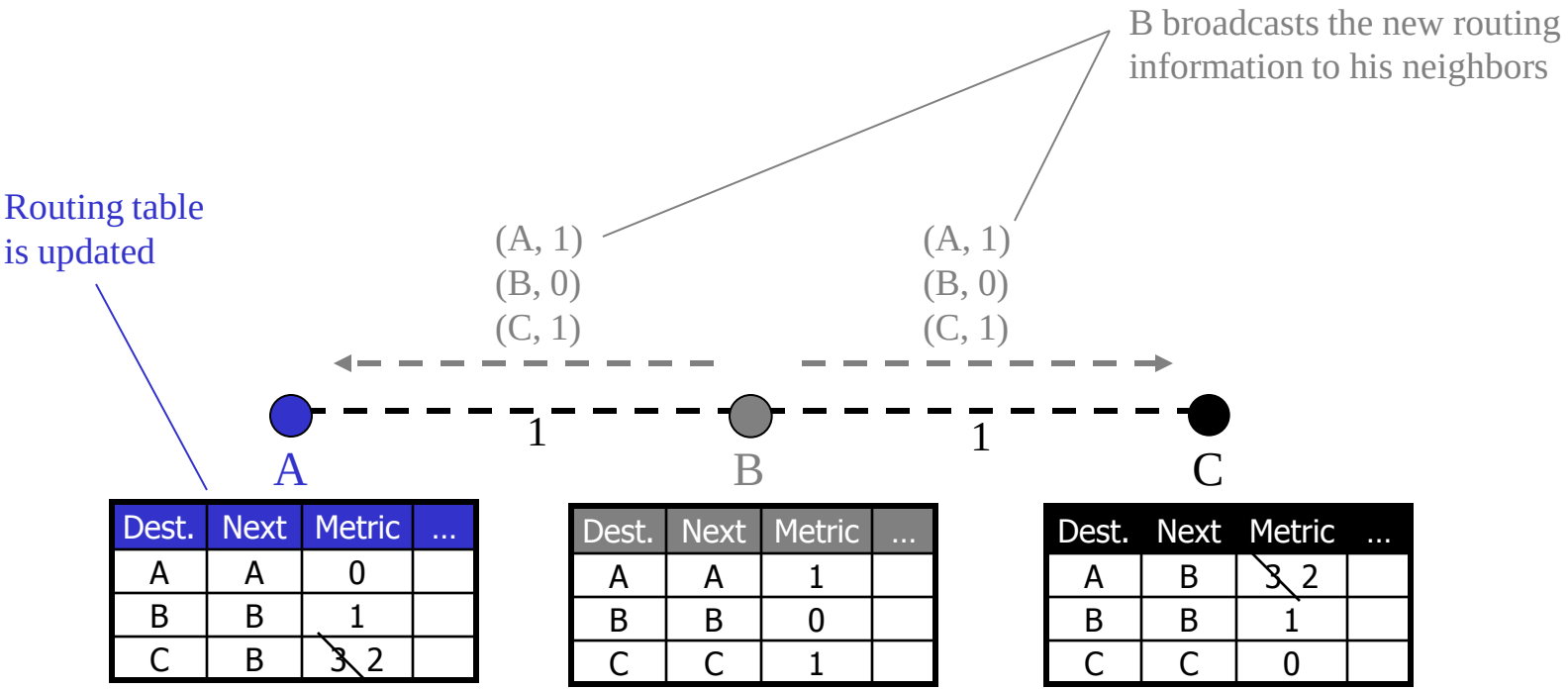


Dest.	Next	Metric	...
A	A	0	
B	B	1	
C	B	3	

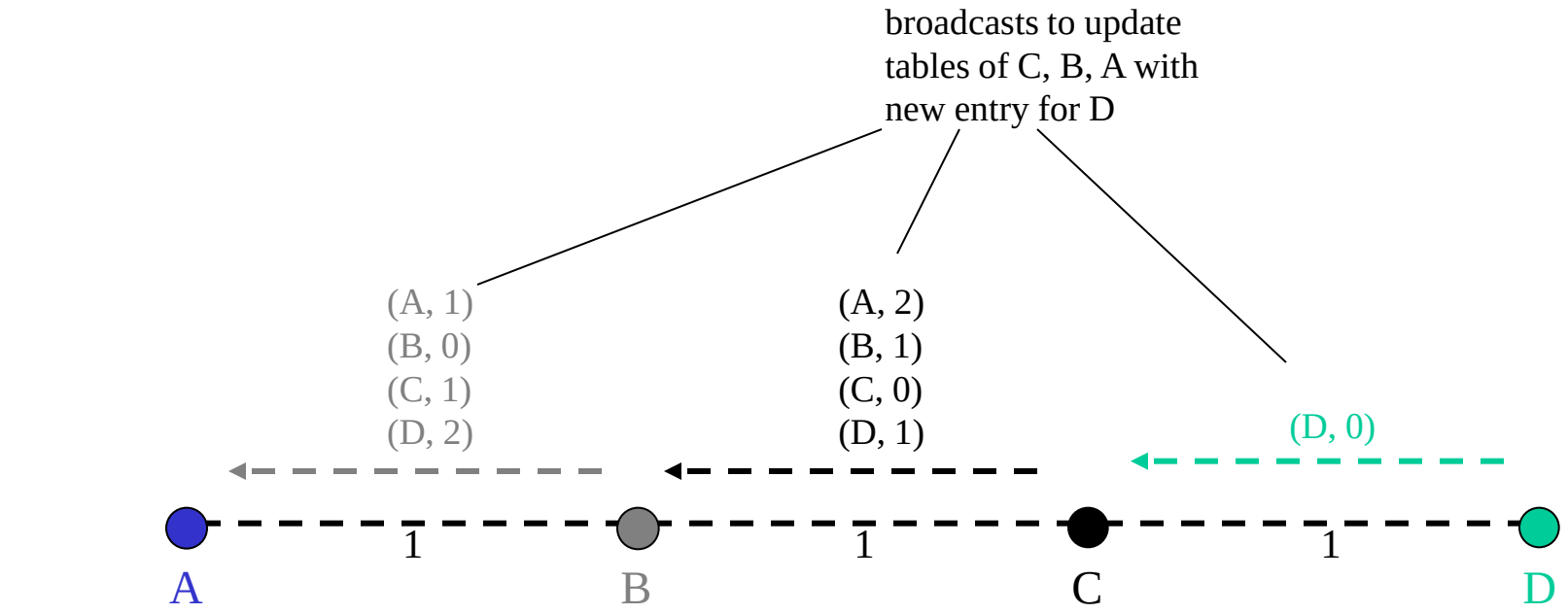
Dest.	Next	Metric	...
A	A	1	
B	B	0	
C	C	2	

Dest.	Next	Metric	...
A	B	3	
B	B	2	
C	C	0	

Distance Vector (Update)



Distance Vector (New Node)

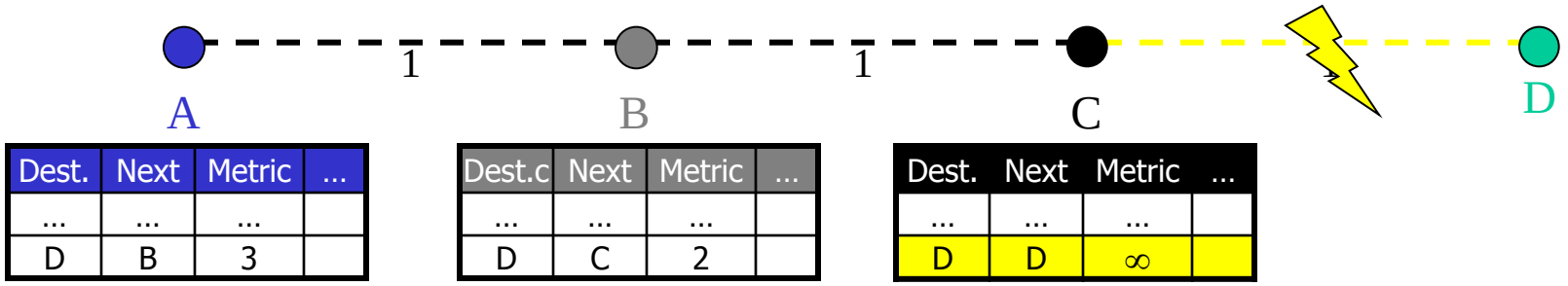


Dest.	Next	Metric	...
A	A	0	
B	B	1	
C	B	2	
D	B	3	

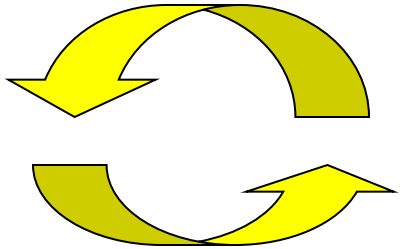
Dest.	Next	Metric	...
A	A	1	
B	B	0	
C	C	1	
D	C	2	

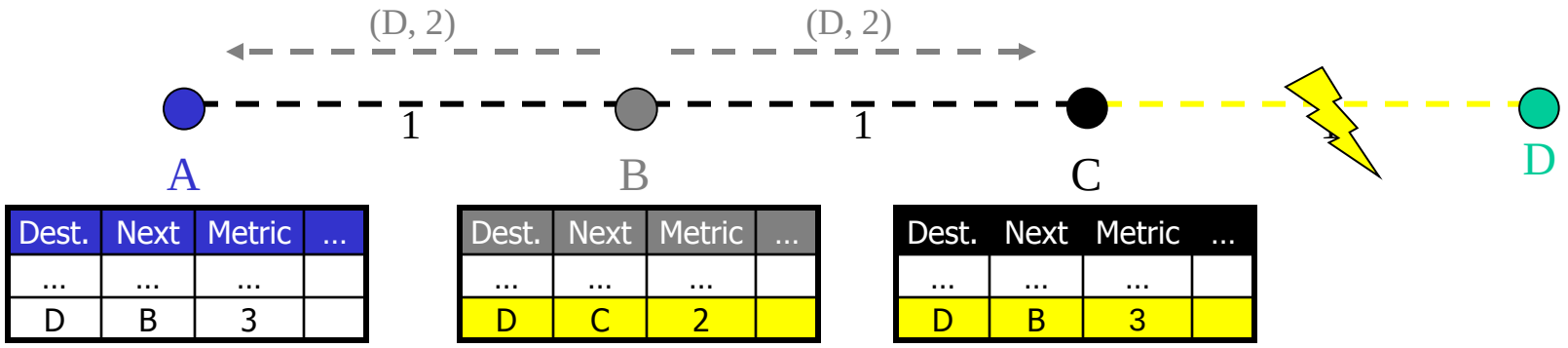
Dest.	Next	Metric	...
A	B	2	
B	B	1	
C	C	0	
D	D	1	

Distance Vector (Broken Link)

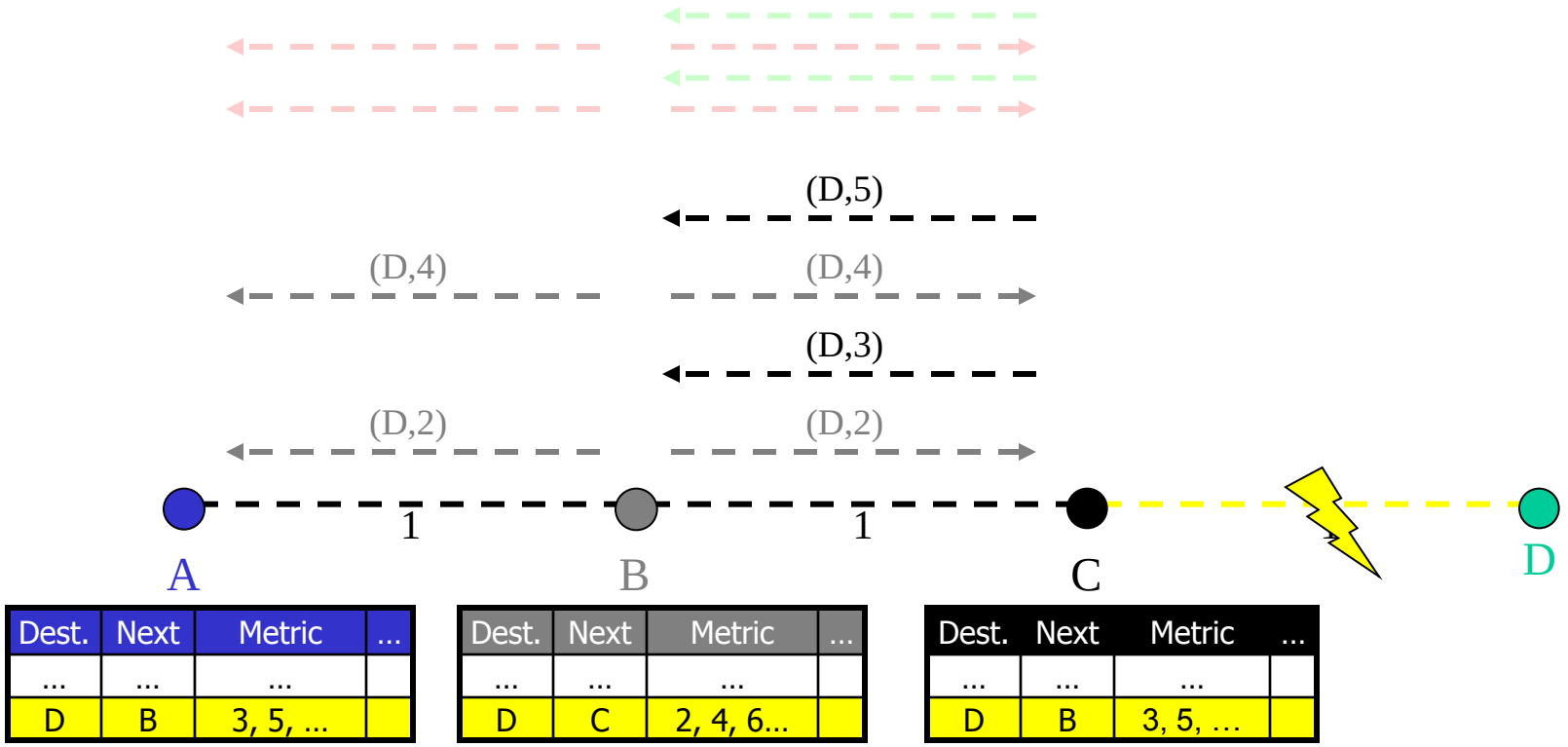


Distance Vector (Loops)





Distance Vector (Count to Infinity)



Dest.	Next	Metric	...
...	...	...	
D	B	3, 5, ...	

Dest.	Next	Metric	...
...	...	...	
D	C	2, 4, 6...	

Dest.	Next	Metric	...
...	...	...	
D	B	3, 5, …	

Distance Vector

- DV not suited for ad-hoc networks!
  - Loops
    - Bandwidth reduction in network
    - Unnecessary work for loop nodes
  - Count to Infinity
    - Very slow adaptation to topology changes.
- Solution -> DSDV

DSDV Protocol

- Keep the simplicity of Distance Vector
- Guarantee loop freeness
  - New Table Entry for Destination Sequence Number
- Allow fast reaction to topology changes
  - Make immediate route advertisement on significant changes in routing table
  - but wait with advertising of unstable routes

DSDV (Table Entries)

Destination	Next	Metric	Seq. Nr	Install Time
A	A	0	A-550	001000
B	B	1	B-102	001200
C	B	3	C-588	001200
D	B	4	D-312	001200

- **Sequence number** originated from destination. Ensures loop freeness.
- **Install time** when entry was made (used to delete stale entries from table)

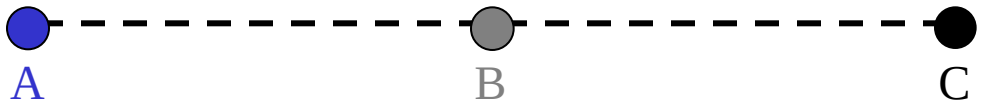
DSDV (Route Advertisements)

- Advertise to each neighbor own routing information
  - Destination Address
  - Metric = Number of Hops to Destination
  - Destination Sequence Number
  - Other info (e.g. hardware addresses)
- Rules to set sequence number information
  - On each advertisement increase own destination sequence number (use only even numbers)
  - If a node is no more reachable (timeout) increase sequence number of this node by 1 (odd sequence number) and set metric = ∞.

DSDV (Route Selection)

- Update information is compared to own routing table
  - 1. Select route with higher destination sequence number (This ensure to use always newest information from destination)
  - 2. Select the route with better metric when sequence numbers are equal.

DSDV (Tables)

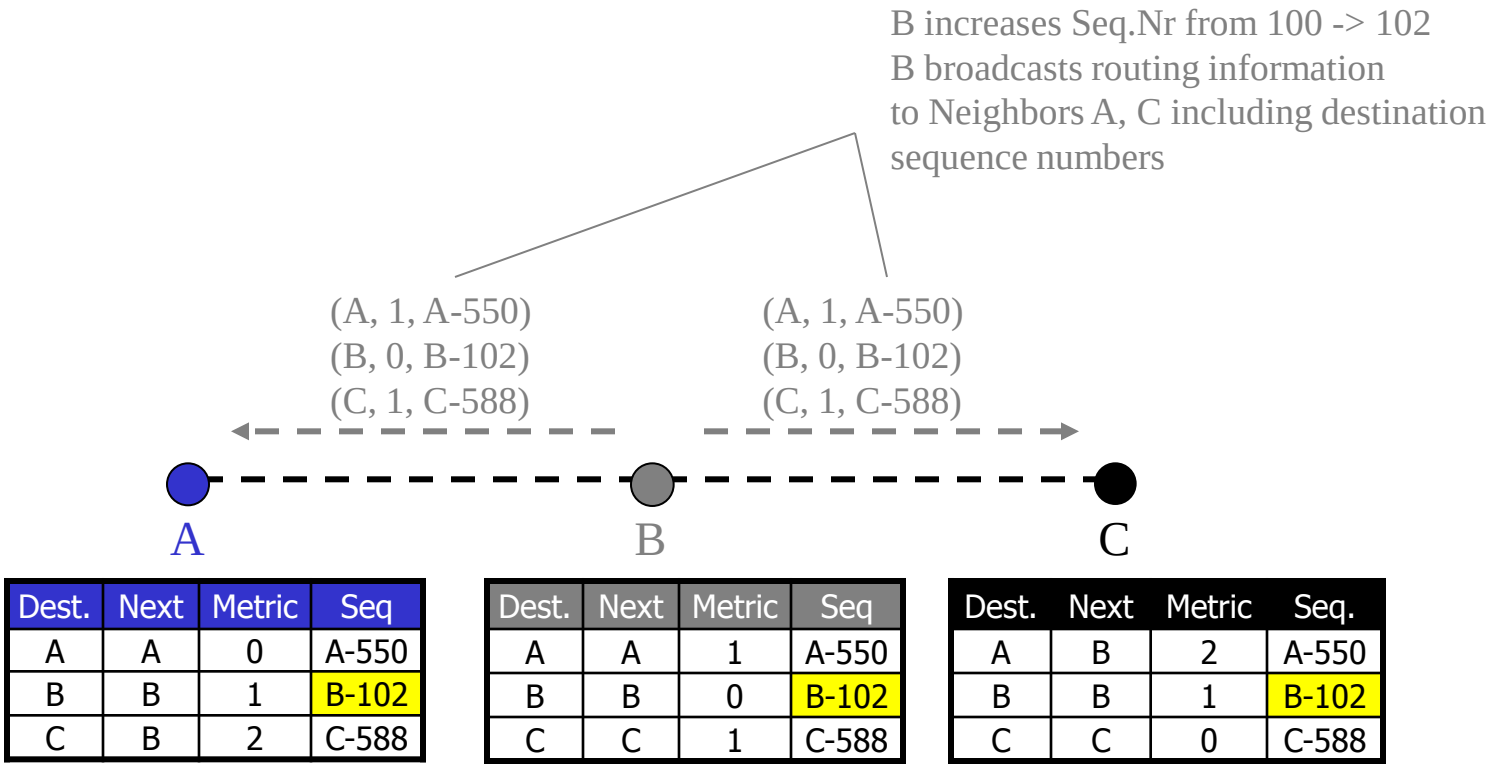


Dest.	Next	Metric	Seq
A	A	0	A-550
B	B	1	B-100
C	B	2	C-586

Dest.	Next	Metric	Seq
A	A	1	A-550
B	B	0	B-100
C	C	2	C-588

Dest.	Next	Metric	Seq.
A	B	1	A-550
B	B	2	B-100
C	C	0	C-588

DSDV (Route Advertisement)



Dest.	Next	Metric	Seq
A	A	0	A-550
B	B	1	B-102
C	B	2	C-588

Dest.	Next	Metric	Seq
A	A	1	A-550
B	B	0	B-102
C	C	1	C-588

Dest.	Next	Metric	Seq.
A	B	2	A-550
B	B	1	B-102
C	C	0	C-588

DSDV (Respond to Topology Changes)

- Immediate advertisements
  - Information on new Routes, broken Links, metric change is immediately propagated to neighbors.

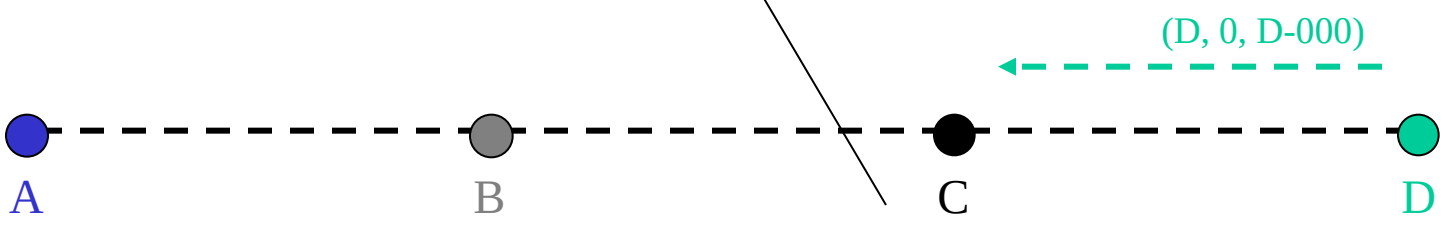
- Full/Incremental update:
  - Full Update: Send all routing information from own table.

  - Incremental Update: Send only entries that has changed. (Make it fit into one single packet)

DSDV (New Node)

2. Insert entry for D with sequence number D-000
Then immediately broadcast own table

1. D broadcast for first time
Send Sequence number D-000

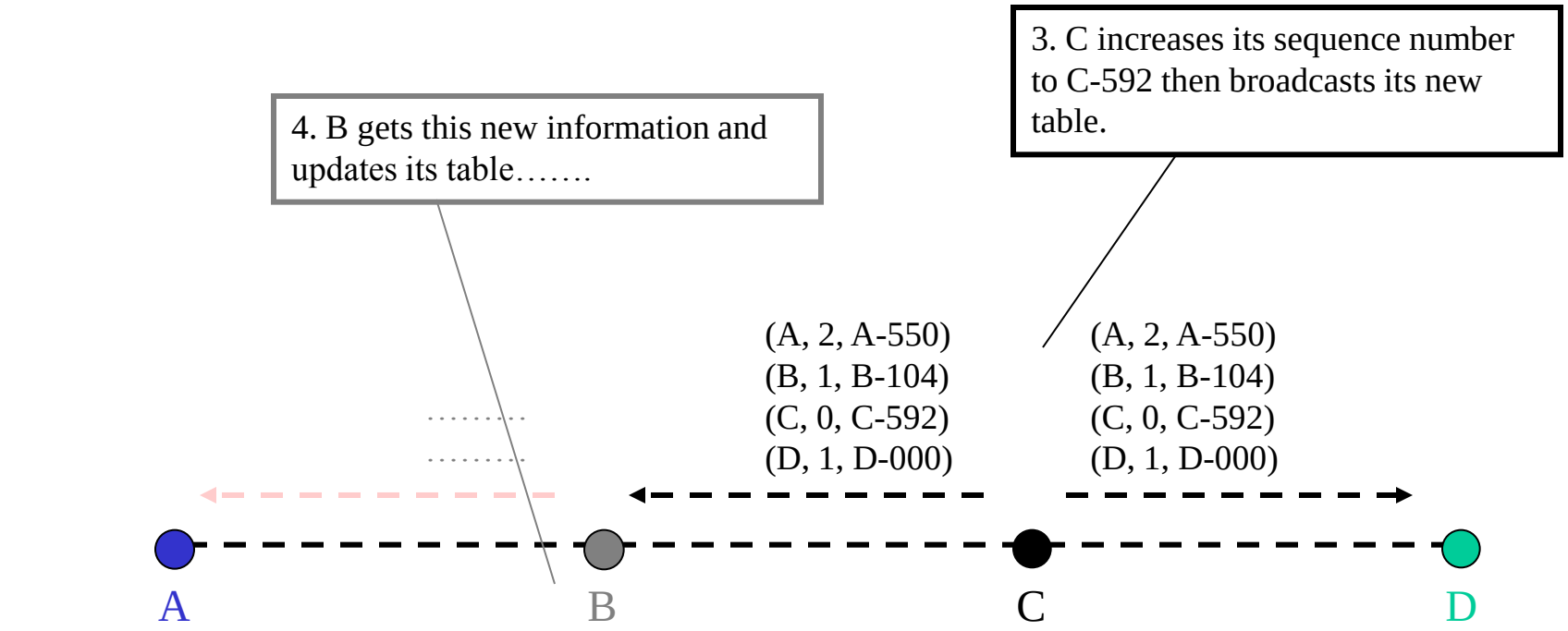


Dest.	Next	Metric	Seq.
A	A	0	A-550
B	B	1	B-104
C	B	2	C-590

Dest.	Next	Metric	Seq.
A	A	1	A-550
B	B	0	B-104
C	C	1	C-590

Dest.	Next	Metric	Seq.
A	B	2	A-550
B	B	1	B-104
C	C	0	C-590
D	D	1	D-000

DSDV (New Node cont.)



Dest.	Next	Metric	Seq.
A	A	0	A-550
B	B	1	B-104
C	B	2	C-590

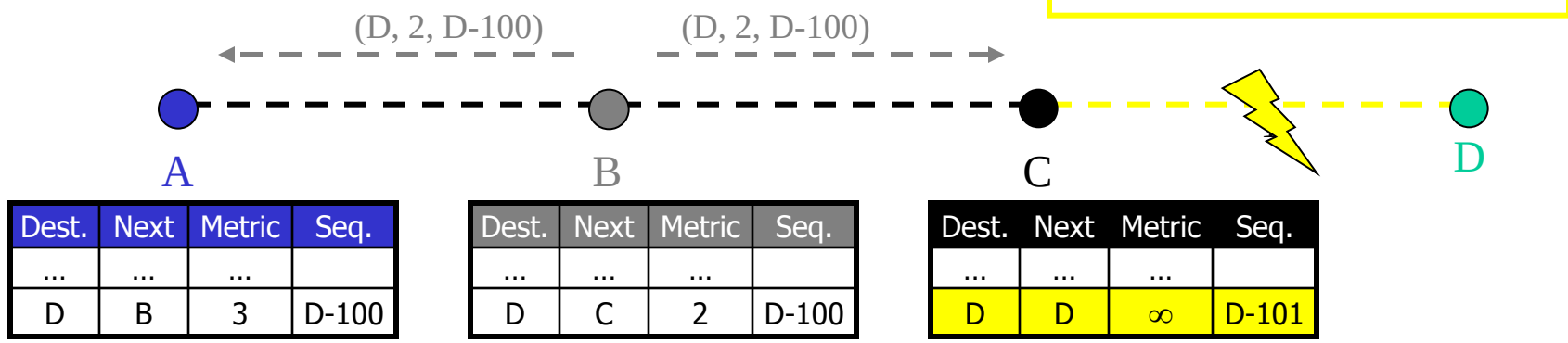
Dest.	Next	Metric	Seq.
A	A	1	A-550
B	B	0	B-104
C	C	1	C-592
D	C	2	D-000

Dest.	Next	Metric	Seq.
A	B	2	A-550
B	B	1	B-104
C	C	0	C-592
D	D	1	D-000

DSDV (no loops, no count to infinity)

2. B does its broadcast
-> no affect on C (C knows that B has stale information because C has higher seq. number for destination D)
-> no loop -> no count to infinity

1. Node C detects broken Link:
-> Increase Seq. Nr. by 1
(only case where not the destination sets the sequence number -> odd number)

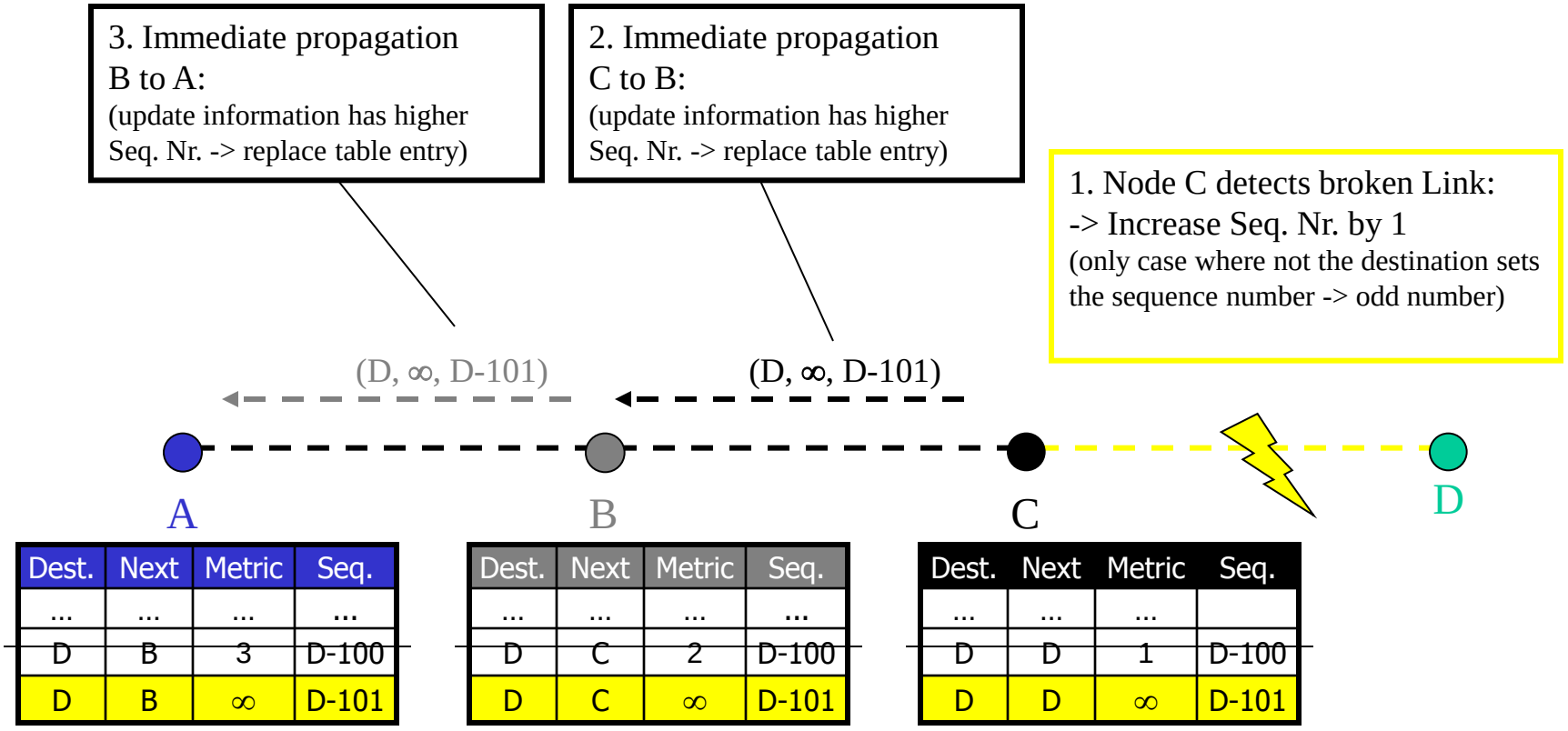


Dest.	Next	Metric	Seq.
...	...	...	
D	B	3	D-100

Dest.	Next	Metric	Seq.
...	...	...	
D	C	2	D-100

Dest.	Next	Metric	Seq.
...	...	...	
D	D	∞	D-101

DSDV (Immediate Advertisement)



DSDV properties

- Advantages
  - Simple (almost like Distance Vector)
  - Loop free through destination seq. numbers
  - No latency caused by route discovery
- Disadvantages
  - No sleeping nodes
  - Bi-directional links required
  - Overhead: most routing information never used
  - Scalability is a major problem

Reactive Protocols

Dynamic Source Routing (DSR)

- Every packet carries the routing sequence
- Intermediate nodes may learn routes on “heard” traffic (RREQ, RREP, DATA)
- No periodic sending of routing packets
- May piggyback route requests on route replies

Ad-Hoc On Demand Distance Vector (AODV)

- Uses “traditional” routing tables
- Hello messages sent periodically to identify neighbors
- Sequence numbers guarantees freshness
- Route requests are sent in reverse direction, i.e. only uses bi-directional links

Dynamic Source Routing

DSR

D. B. Johnson and D. A. Maltz
CMU

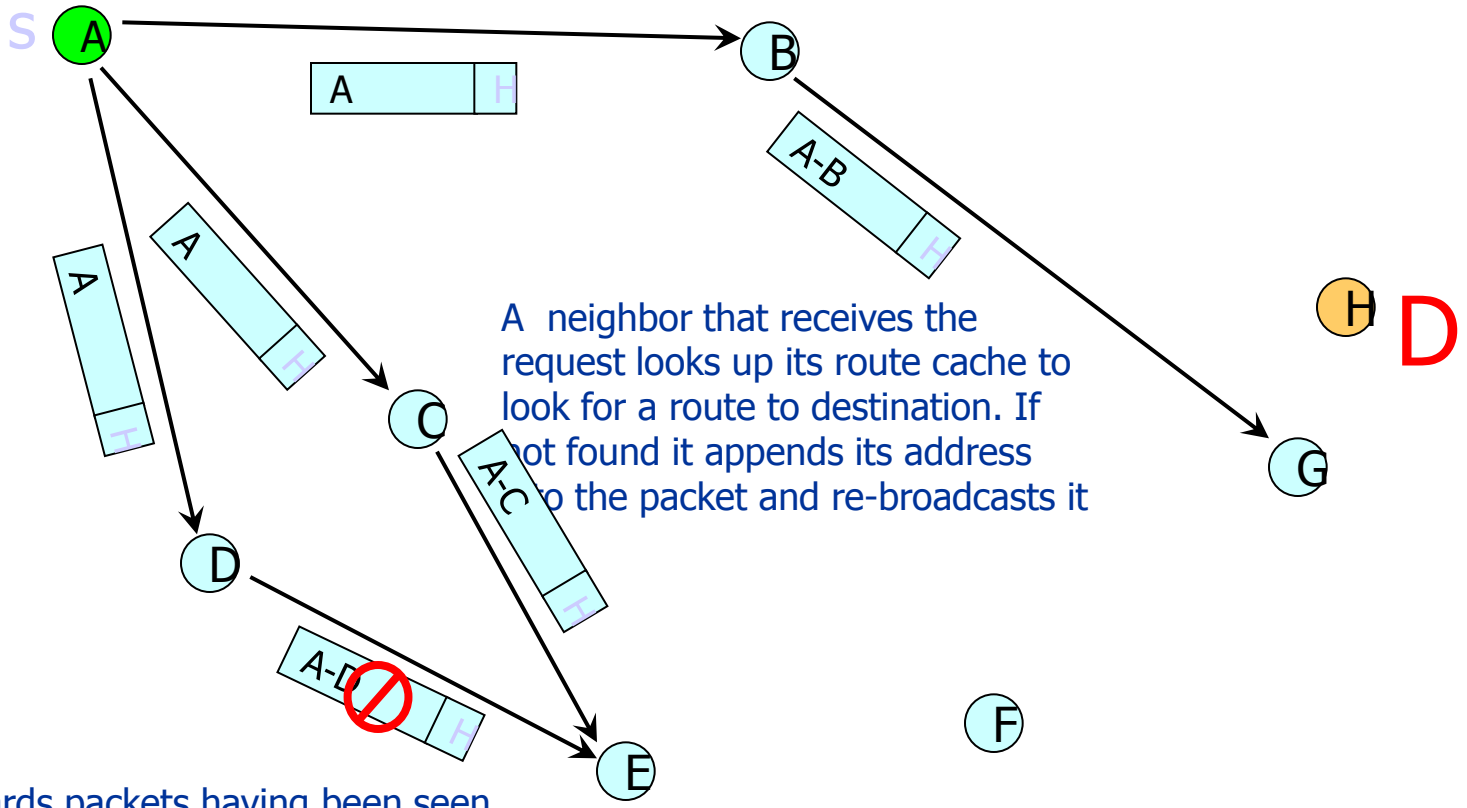
"Dynamic Source Routing in Ad-Hoc Wireless Networks”,
Mobile Computing, edited by T. Imielinski and H. Korth, Chapter 5, Kluwer Academic Publishers, 1996.

Dynamic Source Routing (DSR)

- To send a packet the sender constructs a *source route* in the packet's header (**Source Routing**)
- The source route has the address of every host through which the packet should be forwarded to reach its destination
- No Periodic Announcements
- Each host in the ad hoc network maintains a *route cache* in which it stores source routes it has learned
- Each entry in the route cache has an expiration period, after which it will be deleted
- If the sender doesn't have a route to a destination it then attempts to find out by using a *routing discovery* process
- While waiting for the routing discovery to complete the sender continues sending and receiving packets with other hosts
- Each host uses a *route maintenance* procedure to monitor the correct operation of a route

DSR - Route Discovery

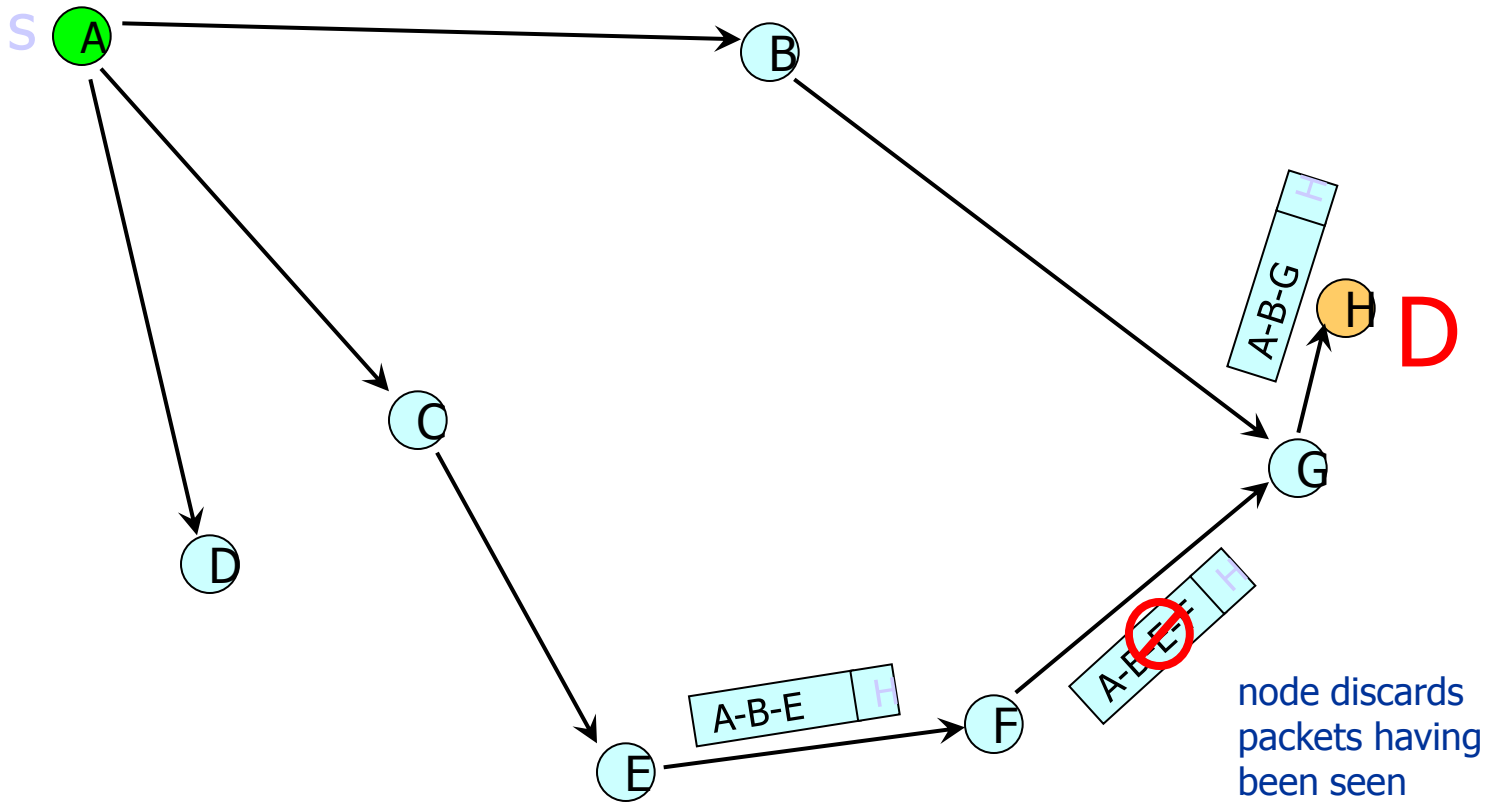
Source broadcasts a route packet with the address of the source and destination



A neighbor that receives the request looks up its route cache to look for a route to destination. If not found it appends its address to the packet and re-broadcasts it

node discards packets having been seen

DSR - Route Discovery

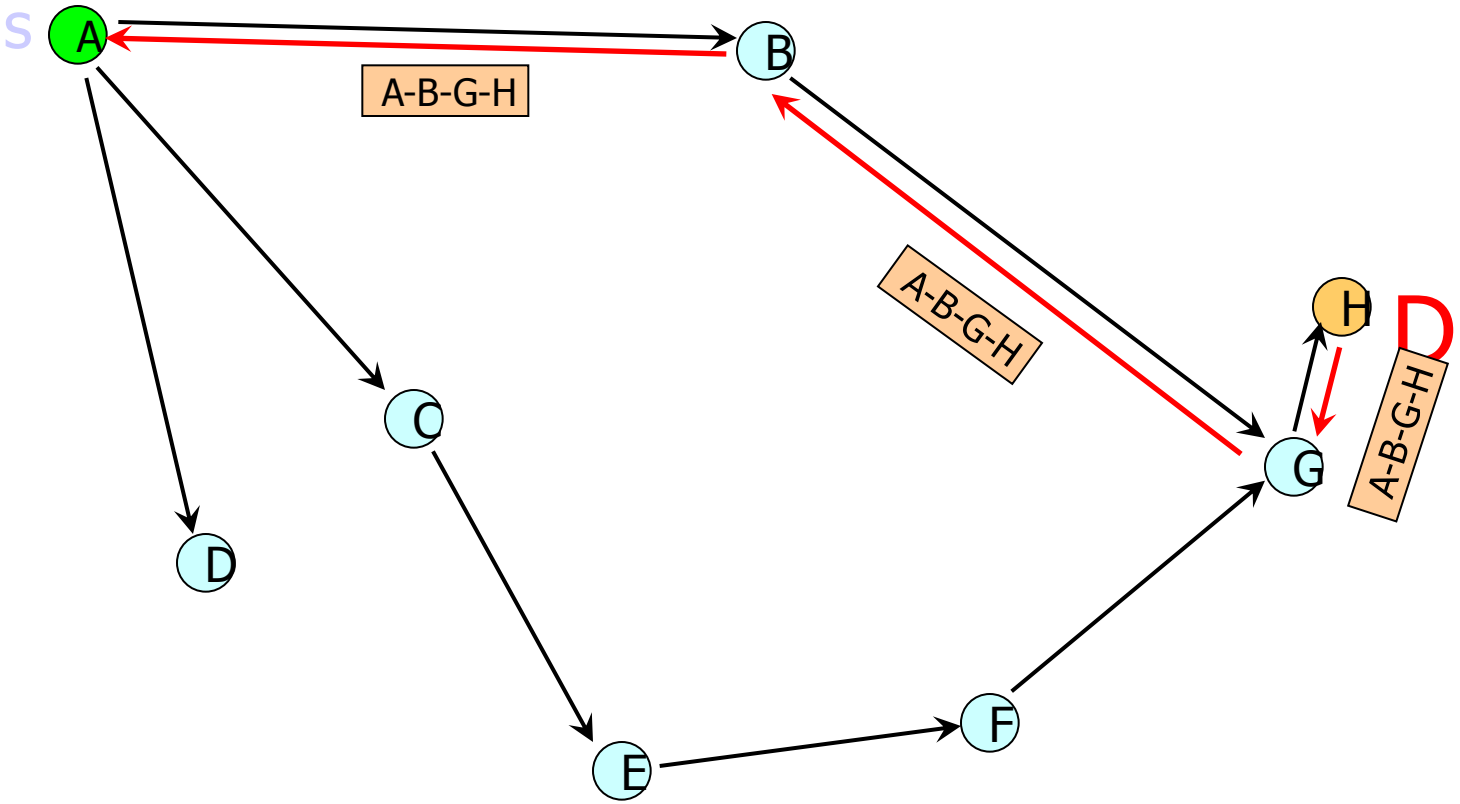


How to send a reply packet

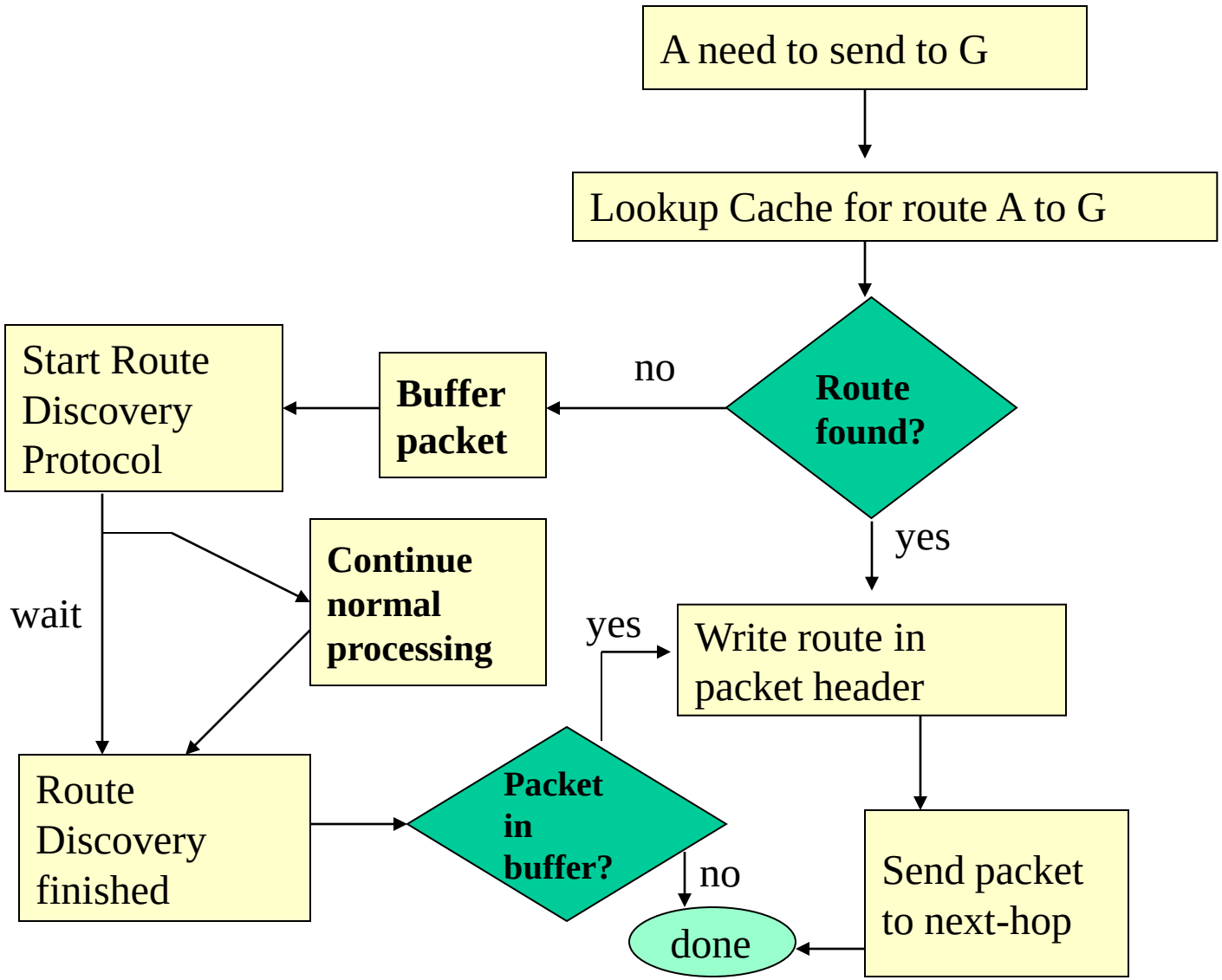
- If the destination has a route to the source in its route cache, use it
- Else if symmetric links are supported, use the reverse of route record
- Else if symmetric links are not supported, the destination initiate route discovery to source

DSR - Route Discovery

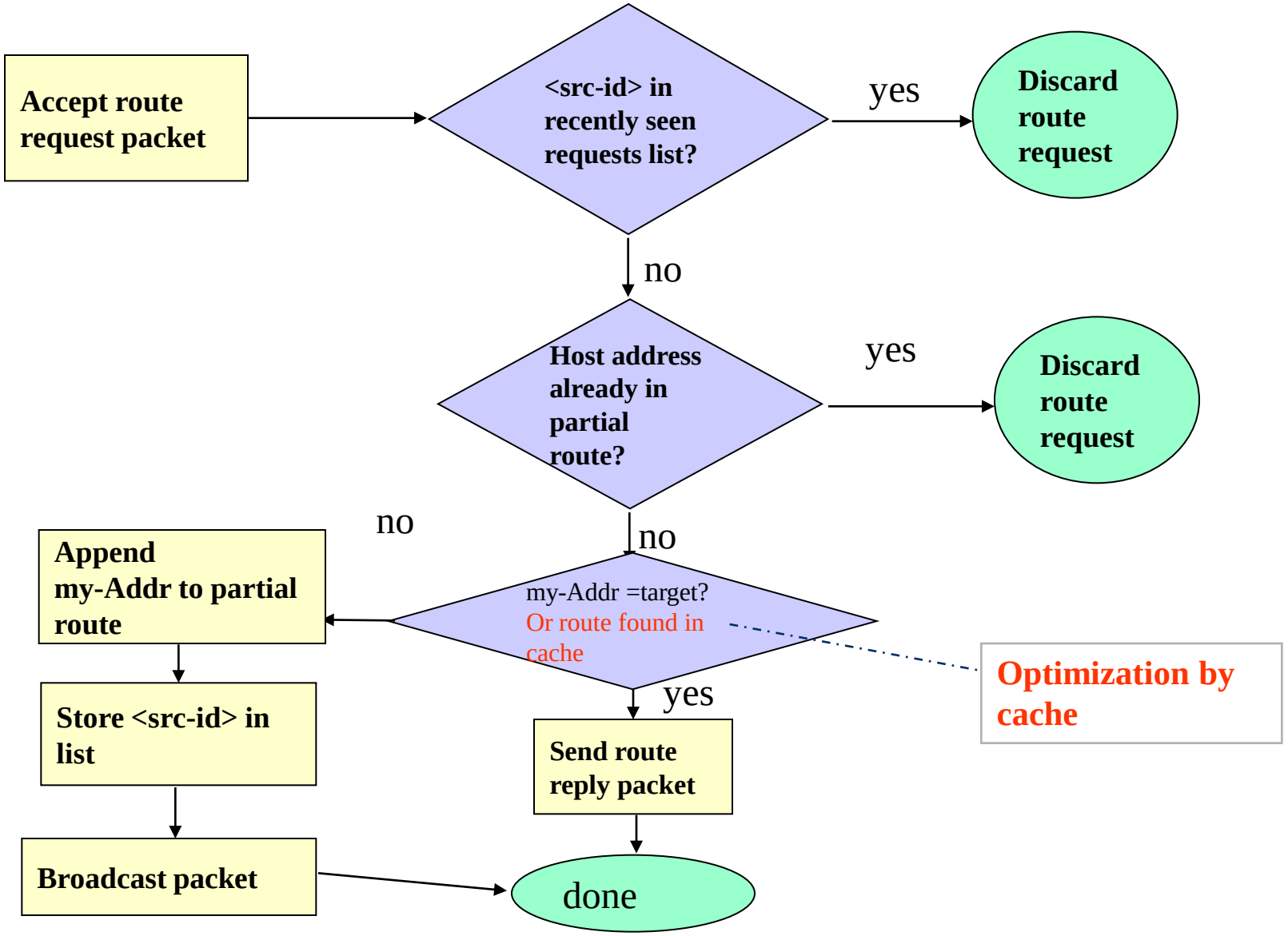
reply packet follows the reverse path of the route request packet recorded in broadcast packet



Route Discovery: at source A



Route Discovery: At an intermediate node



DSR – Route Maintenance

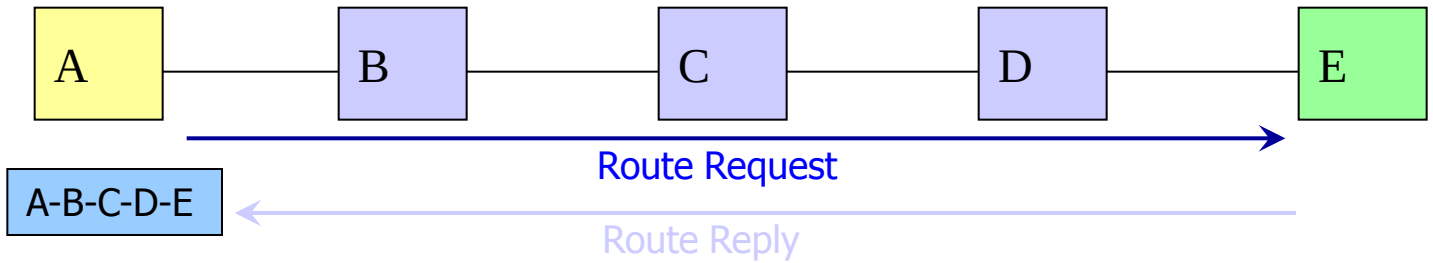
- Usually the data link layer has a mechanism to detect a link failure
- When a link failure is detected the host sends a *route error packet* to the original sender of the packet
- The route error packet has the address of the host who detected the failure and the host to which it was attempting to transmit the packet
- When a route error packet is received by a host, the hop in error is removed from the host's route cache, and all routes which contain this hop are truncated at that point
- To return the route error packet the host uses a route in its route cache to the sender of the original packet. It the host does not have a route it can reverse the route information carried in the packet that could not be forwarded because of the link error. The later assumes that only bidirectional links are being used for routing
- Another option to return route error packets is to perform a route discovery process to find a route to the original sender and use that route

DSR - Optimizations

Several optimizations are possible to reduce the amount of overhead traffic

Route Cache

During the process of route discovery and maintenance a host receives, directly or indirectly, information about routes to other hosts thus minimizing the need to search for that information in the future. For example in the following ad hoc network let's assume that node A performs a route discovery to E

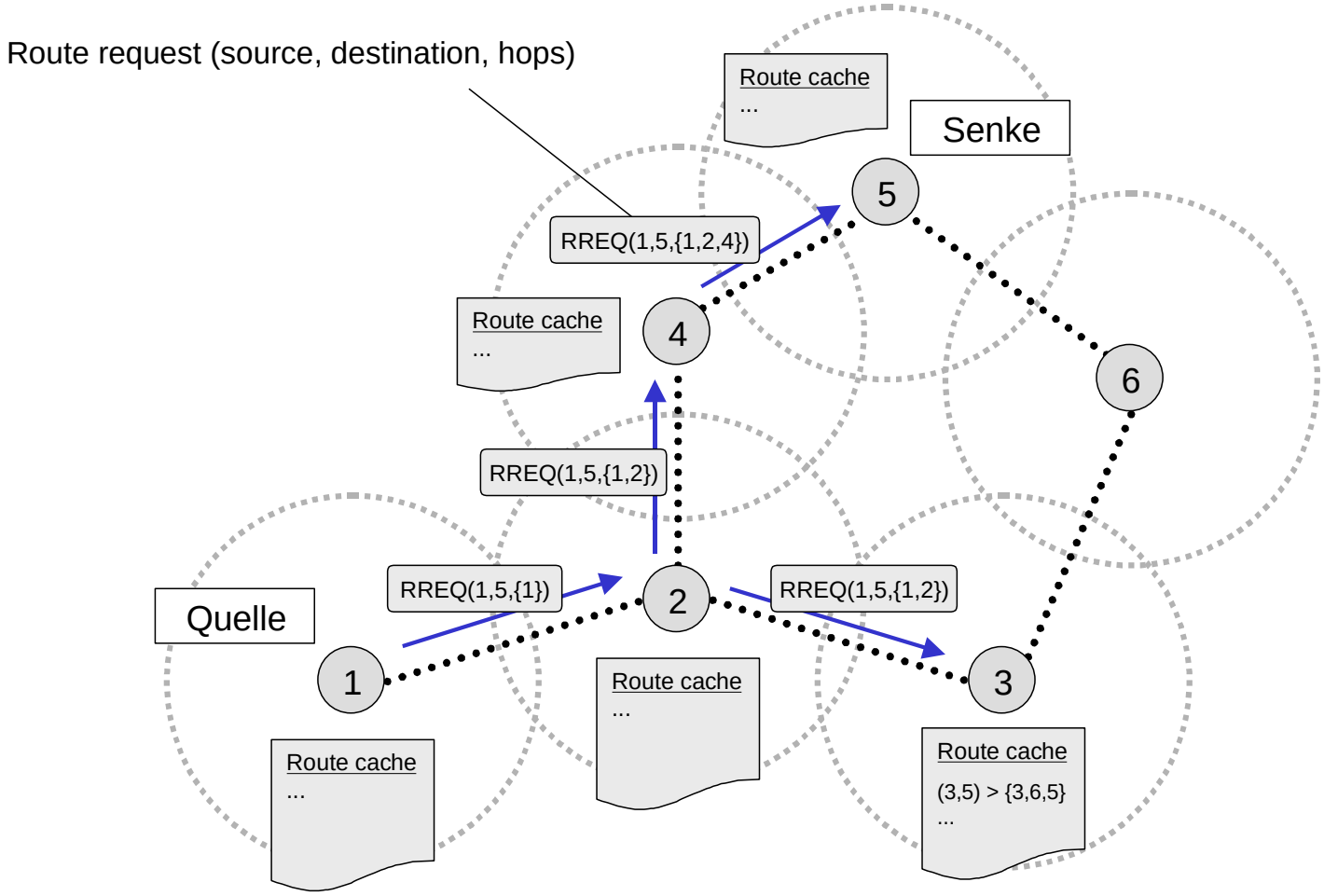


Since hosts B, C, and D are on the route to E, host A also learns the routes to B, C, and D. Likewise these "intermediate hosts" learn about routes to each other by looking into the content of the route reply packet

Route Discovery – with Cache

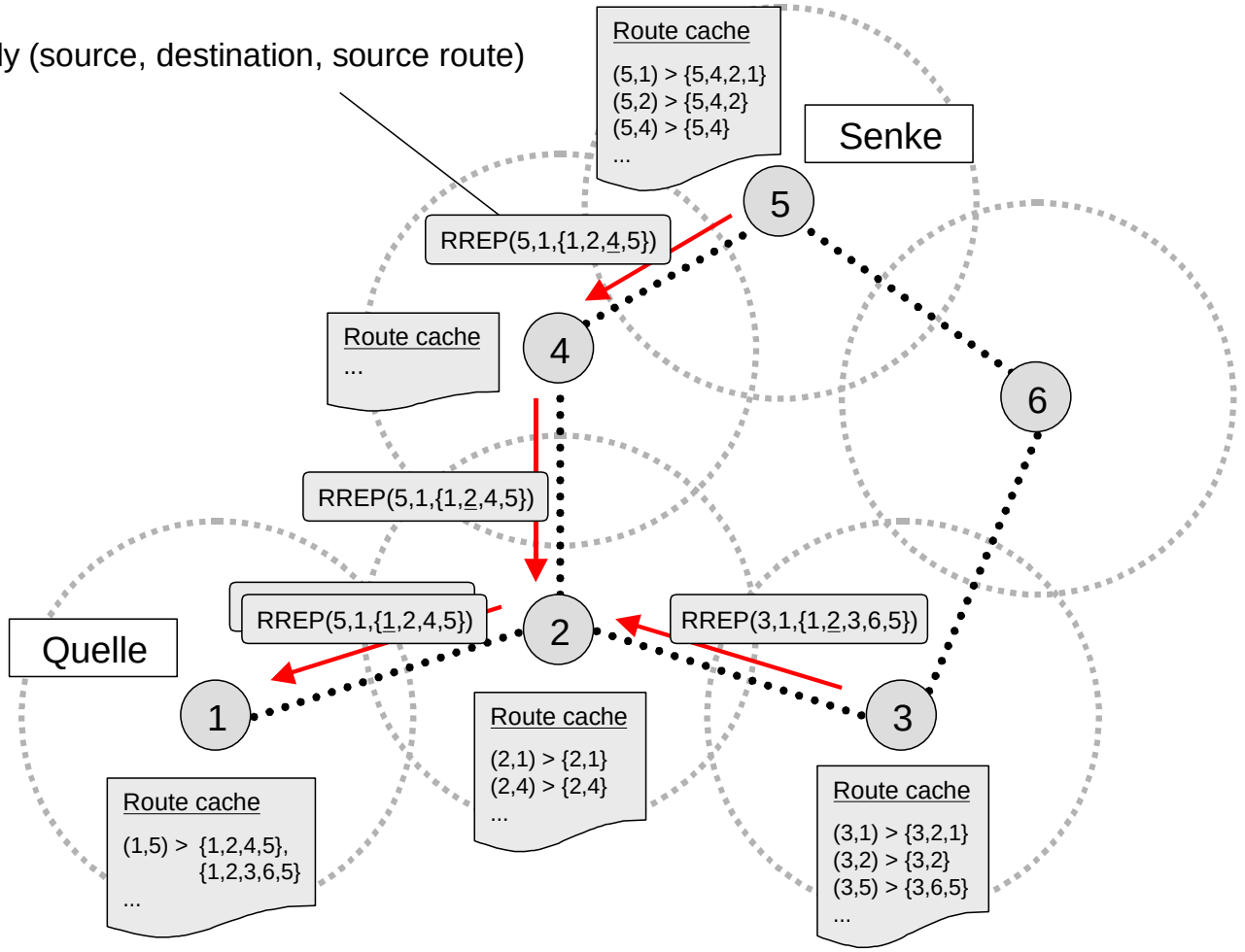
- The source sends a broadcast packet which contains source address, destination address, request id and path.
- If a host saw the packet before, discards it.
- Otherwise, the route looks up its route caches to look for a route to destination, If not find, appends its address into the packet, rebroadcast,
- If finds a route in its route cache, sends a route reply packet, which is sent to the source by route cache or the route discovery.

Route Request - with Cache



Route Reply - with Cache

Route reply (source, destination, source route)



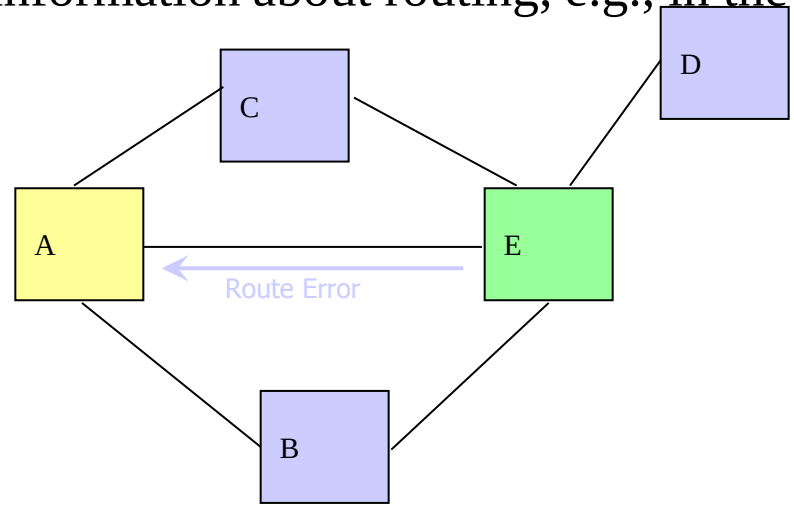
DSR - Optimizations

Piggybacking on Route Discoveries

To minimize the delay in delivering a data packet when there is no route to the destination and a route discovery process is needed one can piggyback the data on the route request packets

Learning by "listening"

If the host operate in promiscuous receiving mode, i.e. they receive and process every transmission in their range, then they can obtain substantial information about routing, e.g., in the following network,



Nodes B, C, and D, listen a process the route error packet from E to A. Since the route error packet identifies precisely the hop where the failure was detected hosts B, C, and D can update their route cache with this information

DSR - summary

- On-demand, potentially minimum control message overhead if topology does not change often
- Do not exchange the routing update periodically, so overhead transmission is greatly reduced.
- Packet delays/jitters(variations of delays) associated with on-demand routing
- Route caching used to minimize route discovery overhead
- Scalability problem: Not easily scalable to large networks since high route discovery latency for large network.
- The need to place the entire route in the route replies and data packets translates in large on the air packet overhead (packet header size grows with route length due to source routing)
- Allows for the possibility to keep multiple routes to a destination in the route cache
- CPU and memory use demands on each host are high since the routes have to be continuously maintained and updated

Ad-Hoc on Demand Distance Vector

AODV

C. E. Perkins, E. M. Royer
SUN and UCSB

"Ad-Hoc on Demand Distance Vector Routing", Proc. IEEE WMCSA '99, 1999.

AODV Algorithm

- The algorithm's primary objectives are
  - To broadcast discovery packets only when necessary
  - To distinguish between local connectivity management and general topology maintenance
  - To disseminate information about changes in local connectivity to those neighboring nodes

- Algorithm
  - Path Discovery
  - Reverse Path Setup
  - Forward Path Setup
  - Route Table Management
  - Path Maintenance
  - Local Connectivity Management

Ad-Hoc On Demand Distance Vector (AODV)

- Based on the Destination-Sequenced Distance-Vector (DSDV) algorithm
- On-demand route acquisition
- Nodes maintain route cache and uses destination sequence number for each route entry
- Does nothing when connection between end points is still valid
- Route Discovery Mechanism is initiated, by broadcasting a Route Request Packet (RREQ), when a route to new destination is needed
- The neighbors forward the request to their neighbors until either the destination or an intermediate node with a “fresh enough” route to the destination is located
- Route Reply Packets (RREP) are transmitted upstream the path taken by the Route Request packet to inform the original sender (an intermediate nodes) of the route finding
- Route Error Packets (RERR) are used to erase broken links

AODV-Path Discovery

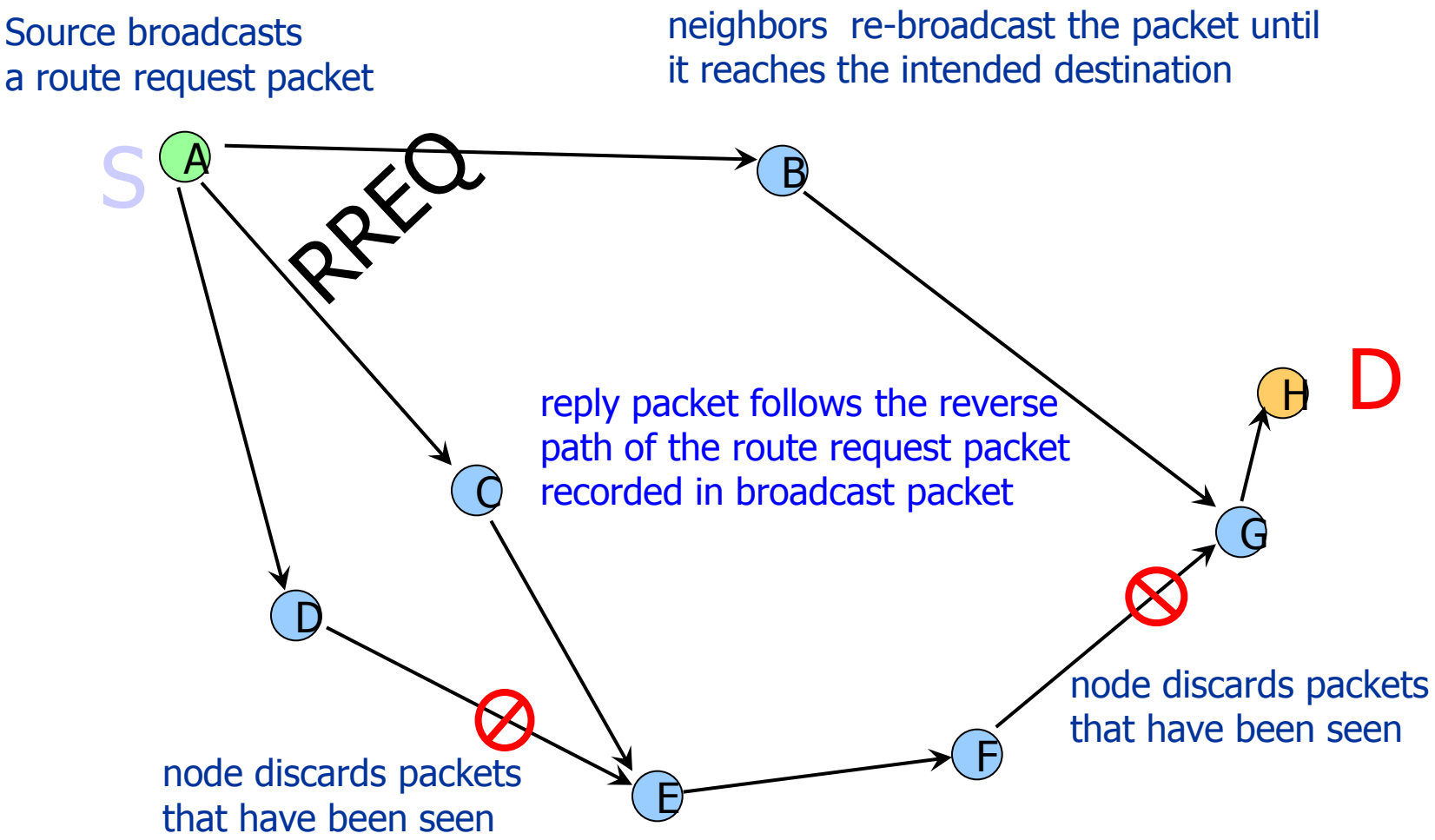
- If there is existing routes to the destination,
  - Sends the packet to this route

- initiated whenever a node needs to communicate with another node.
- initiates path discovery by broadcasting a route request (RREQ) packet to its neighbors.
- RREQ contains :
  - Source_addr, source_sequence_#, broadcast_id, dest_addr, dest_sequence_#, hop_count

- The pair <Source_addr, broadcast_id> uniquely identifies a RREQ.
  - Broadcast_id is incremented whenever the source issues a new RREQ

- At intermediate node, if it receive redundant RREQ (same broadcast_id, source address), drop the RREQ.
- If node is the destination then it sends Route Reply (RREP) else it forwards the RREQ to its neighbors (hop count ++)
- It keeps information about the RREQ to setup the *forward* and *reverse path*

AODV - Path Finding



AODV-Path Discovery (cont.)

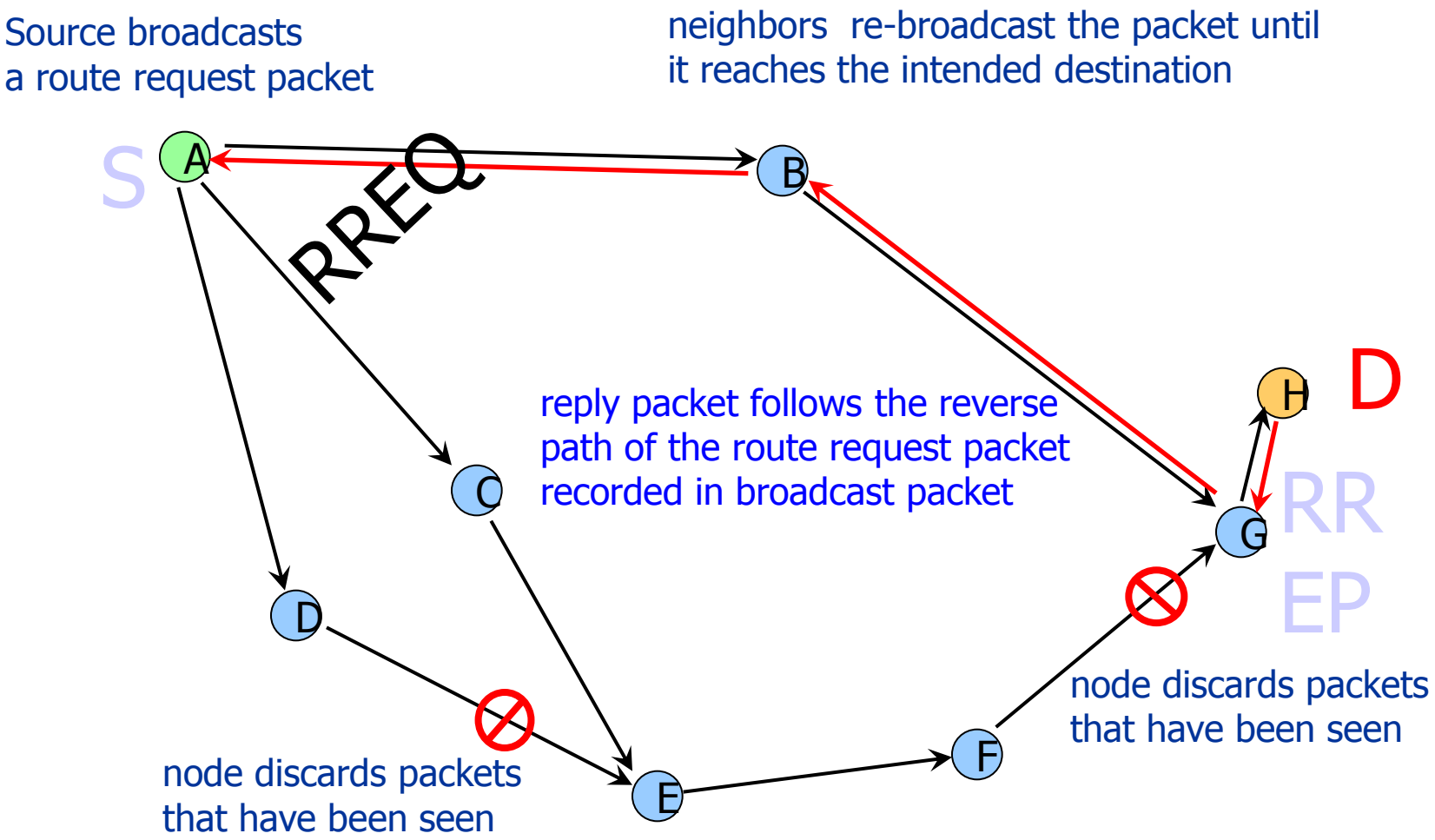
- If the RREQ is lost
  - Retry the broadcast route discovery
  - After rreq_retries attempts,
    - Notify the application that the destination is unreachable

- If a node cannot satisfy the RREQ
  - To implement the reverse path setup, intermediate node keeps track of the following information.
    - Destination IP
    - Source IP
    - Broadcast_id
    - Expiration time for reverse Path route entry
    - Source node's sequence number.

AODV -Reverse Path Setup

- Node set up a reverse route entry for the source node
- A Node records the address of the neighbor from which it received the first copy of the RREQ
- Reverse route entry is contain :
  - Source node's IP address and sequence number
  - Number of hops to the source node
  - IP address of the neighbor from which the RREQ was received
- The node knows how to forward a RREP (from reverse route entry)

AODV-Reverse Path Setup (cont.)



AODV-Forward Path Setup

- RREP Sent in response to the RREQ
- After a node receives the RREQ
  - If an intermediate node has a route for the desired destination
    - Compare the destination sequence number
      - Rebroadcast the RREQ or reply with RREP
  - If a node does have a current route to the destination
    - Send RREP to its neighbor
  - Anything else
    - Forward RREQ with incremented hop count to adjacent

- A RREP contains the following information:
  - Source address, dest_addr
  - dest_sequence number, hop_count
  - lifetime

AODV-Forward Path Formation (cont.)

- When an intermediate node receives the RREP
  - Set up a Forward entry to the destination in its route table
- Forward path entry contains :
  - IP address of the destination
  - IP address of the neighbor from which the RREP arrived
  - Hop count to the destination
    - To obtain its hop count, the node increments the value by 1
  - Lifetime
    - Not used within lifetime, it is deleted
- In the case that received more than one neighbor, node forwards the first RREP

AODV-Forward Path Formation (cont.)

- As soon as source node receives first RREP, it begins data transmission

- If a node receives further RREPs, it updates its routing information
  - In the case that RREP contains greater destination sequence number than the previous RREP or same with a smaller hop count, propagates the RREP

AODV-Route Table Management

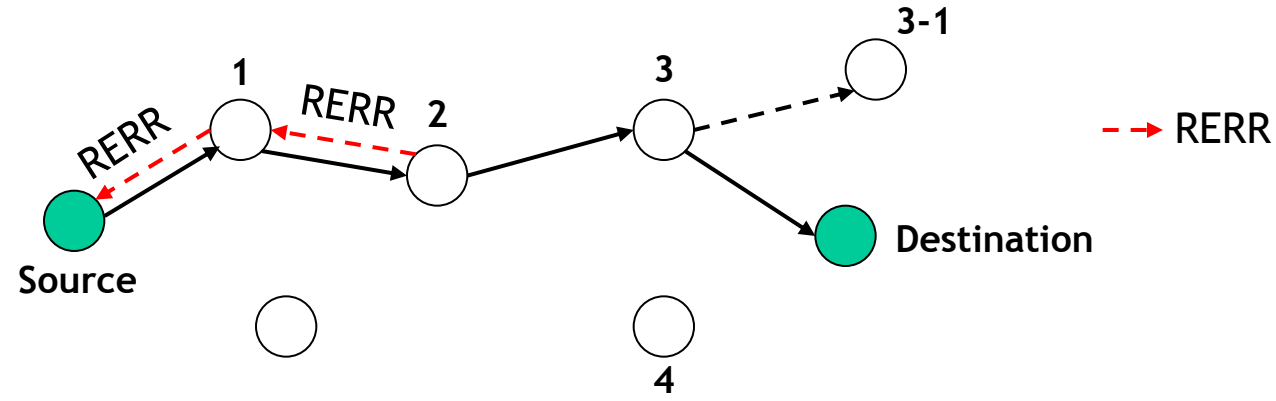
- Each route table entry contains :
  - Destination
  - Next Hop
  - Number of hops (metric)
  - Sequence number for the destination
  - Active neighbors for this route
  - Expiration time for the route table entry

- Route request expiration timer
  - To Purge reverse path routing entries (not lie on the path from the source to the destination)

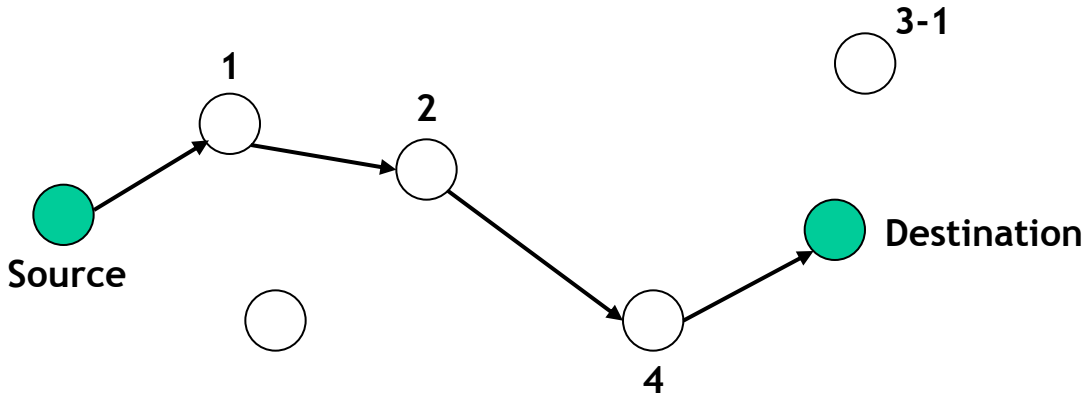
AODV-Path Maintenance

- When source node move
  - Reinitiate the route discovery
- When node (destination, intermediated) move
  - RERR is sent to the affected source nodes
  - Broadcasts the RERR to these neighbors (precursor nodes)
  - Neighbors mark hop count (infinity) and propagate the RERR to their precursor nodes.
  - When source node receives the RERR
    - Reinitiate route discovery

AODV-Path Maintenance (cont.)



(a) Node move



(b) New route find

AODV-Local Connectivity Management

- If the Node has not sent any packets to all of its active downstream neighbors within **hello_interval**
  - Broadcasts to its neighbors a hello message
  - hello message contains :
    - Destination address, Destination sequence number
    - the node's address, the latest sequence number
  - Hello message prevented from being rebroadcast outside neighbor
    - Because of Time to live(TTL) value of 1

- An indication that the local connectivity has changed
  - Receiving broadcast or a hello from a new neighbor
  - Failing to receive allowed_hello_loss consecutive hello messages form a node in the neighborhood

Ad Hoc On-Demand Distance Vector

- Combination of DSR and DSDV
  - On-demand mechanism of route discovery and route-maintenance from DSR
  - Plus the hop-by-hop routing, sequence numbers and periodic beacons from DSDV

Zone Routing Protocol

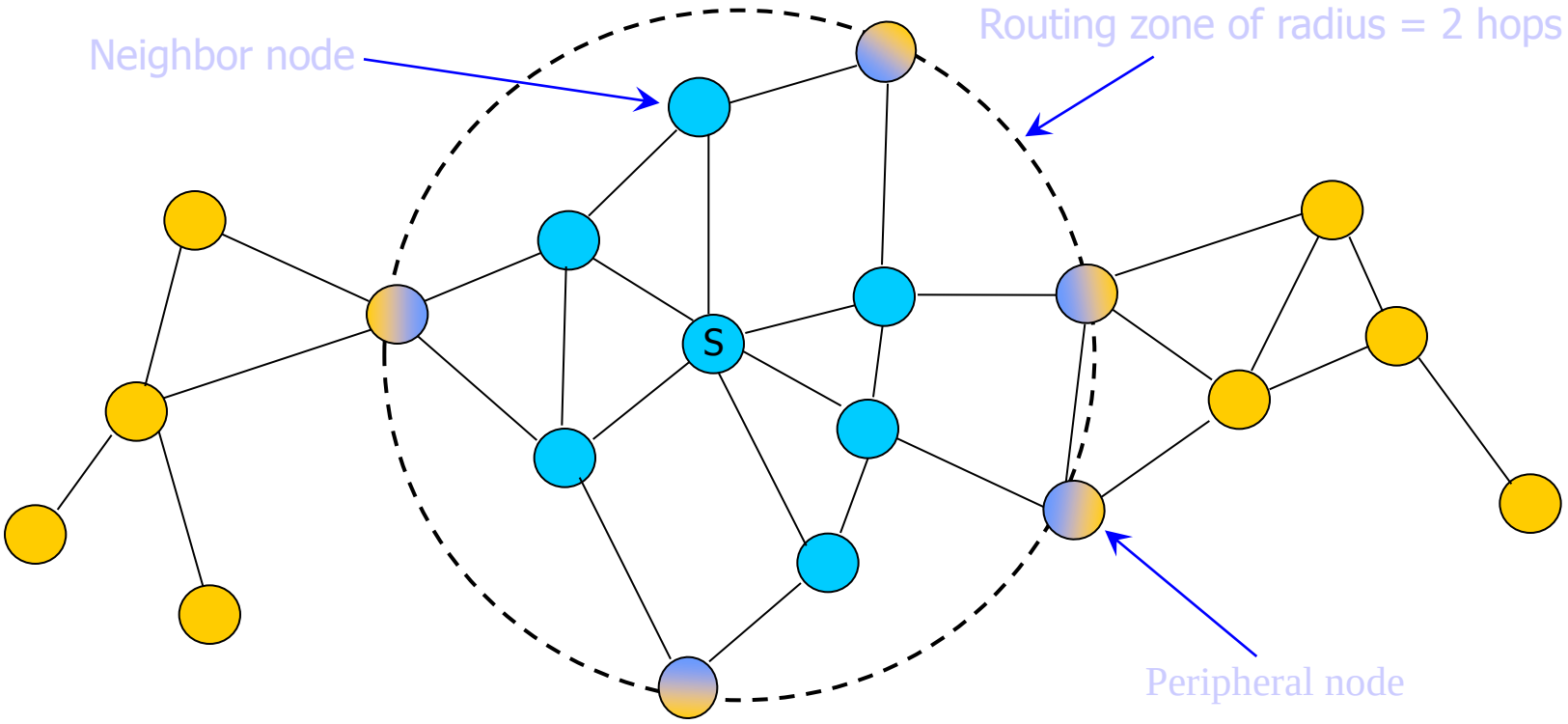
ZRP

M.R. Pearlman, and Z.J. Haas
Cornell

"Determining the Optimal Configuration for the Zone Routing Protocol", IEEE JSAC, 1999, vol. 17, no. 8

Zone Routing Protocol (ZRP)

- Hybrid reactive/proactive protocol
- Proactive procedure only to the nodes within a routine zone of radius ρ
- Reactive procedure to nodes beyond the routing zone by querying only a subset of the network nodes



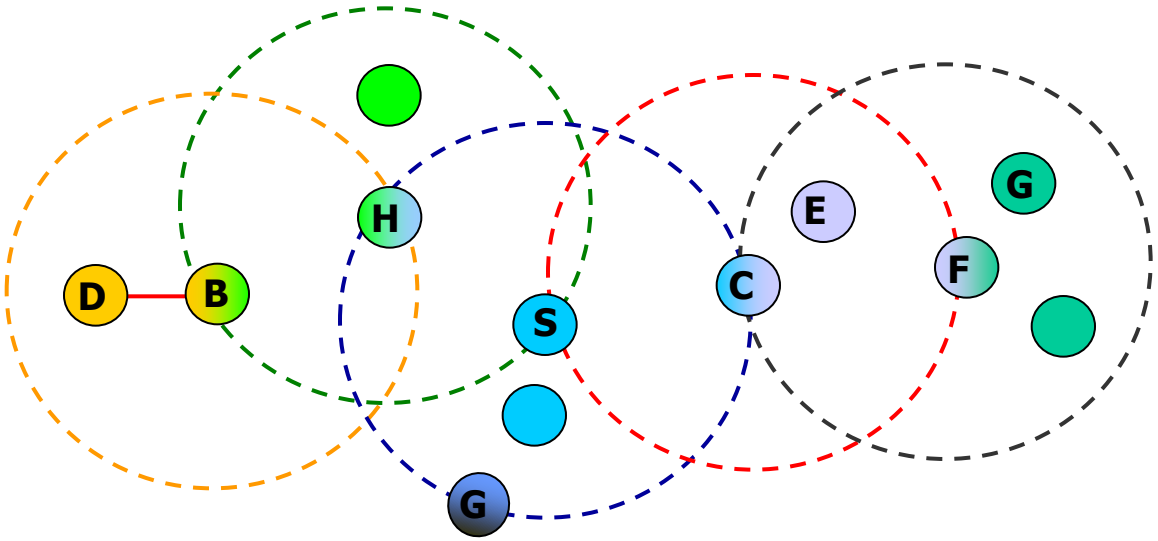
ZRP - Routing Zones

- A collection of nodes which are within the zone radius of another node
- Zone radius of a node is defined in terms of number of hops from that node
- The transmission power as well as the propagation conditions and receiver sensitivity determines the set of *neighbors*
- Each node has its own routing zone
- Routing zones of different nodes may overlap
- Each node maintains routing information to all nodes within its own routing zone
- The nodes uses a proactive mechanism to learn about the topology of its routing zone, this mechanism is called *Intrazone Routing Protocol* (IARP)

ZRP - Interzone Routine

- The *Interzone Routing Protocol* (IERP) is responsible for reactively discovering routes to destinations located beyond a node's routing zone
- The *Bordercast Resolution Protocol* (BRP) allows the node to send messages only to its peripheral nodes
- Efficient querying of specific nodes rather than flooding the whole network
- Bordercasting can be implemented using efficient multicast techniques
- A single route query returns multiple route replies, which can be used to determine the best route based on relative quality
- Because the routing zones overlap, a node can be a member of many routing zones
- It is important to have a mechanism to detect duplicate route queries and reduce excessive control traffic

ZRP - IERP (example)



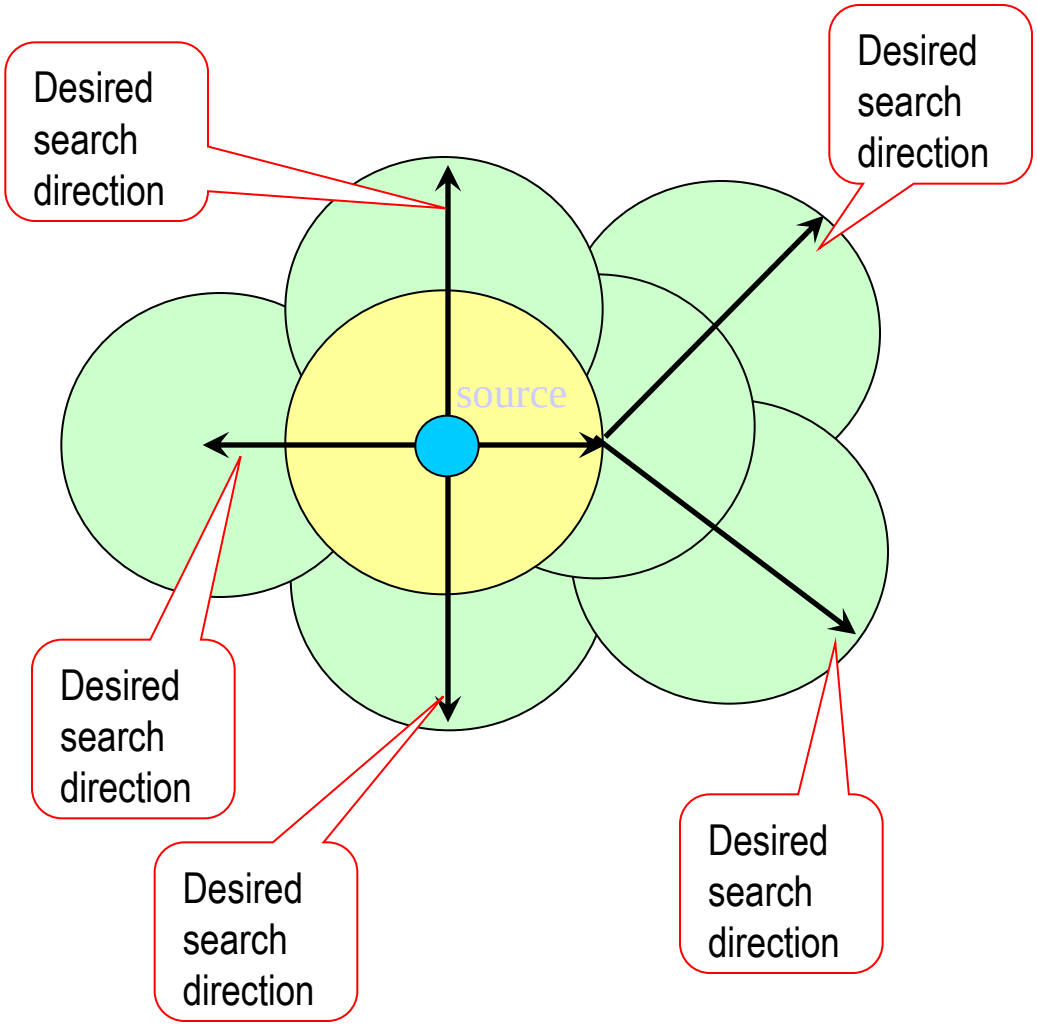
- Source S needs to send a packet to destination D
- S checks whether D is within its routing zone. If yes, S knows a path to D
- If not S bordercasts a query to its peripheral nodes (C, G, and H)
- These nodes, after verifying that D is not within their routing zones, bordercast the query to their peripheral nodes
- B, a peripheral node of H, recognizes D as being in its routing zone and responds to the query indicating the path S→H →B →D

ZRP - Guiding the Search in Desirable Directions

To minimize route query traffic a procedure is needed to steer the query packets outwards from the source's routing zone and away from each other.

This problem is addressed through the following mechanisms:

- *Loop-back Termination* (LT)
- *Query Detection* (QD1/QD2)
- *Early Termination* (ET)
- *Selective Bordercasting* (SBC)

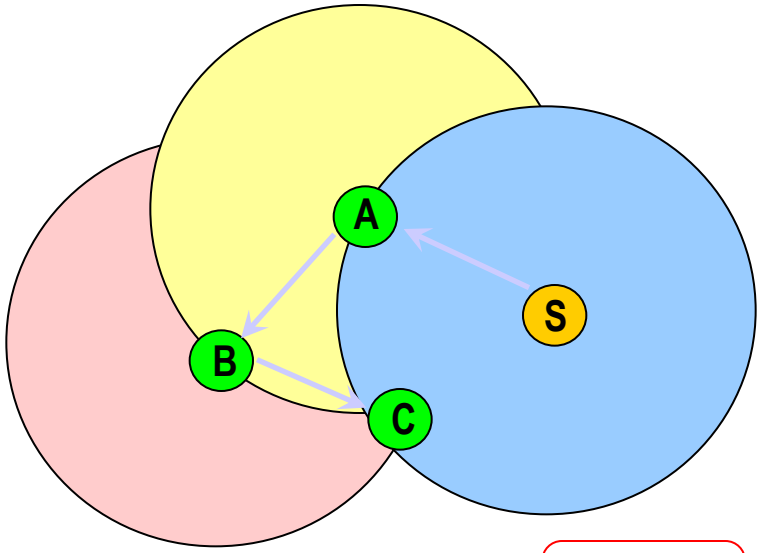


ZRP - Query Control Mechanisms

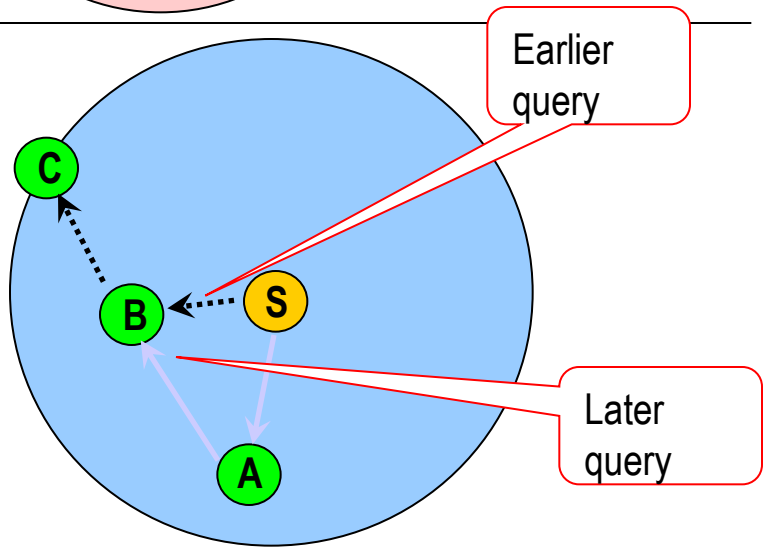
Loop-back Termination
The query is terminated when the accumulated route (excluding the previous node) contains the host which lies in routing zone, e.g.,

route = {S→A→B→C}

C terminates the query, because S is in the C's routing zone.



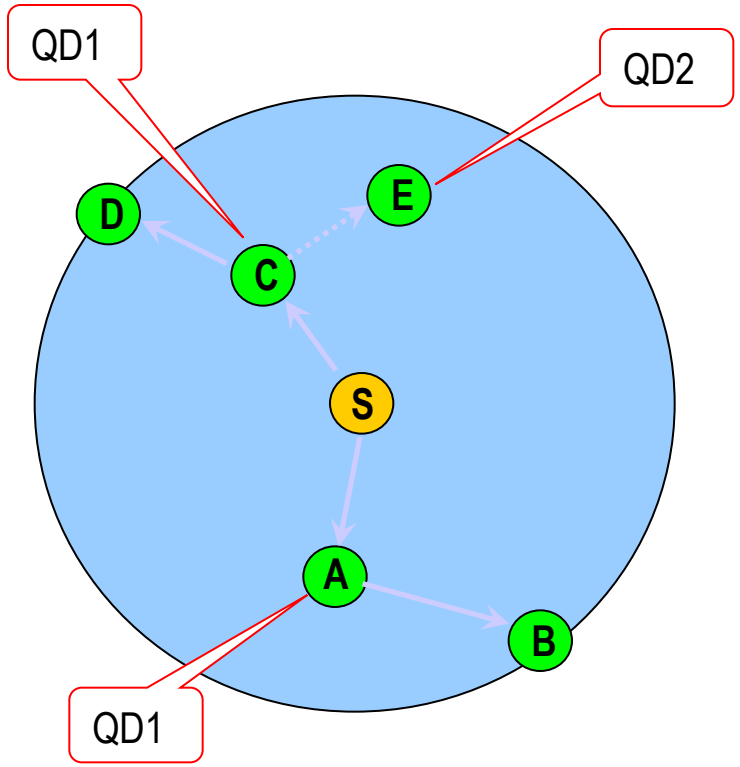
Early Termination
When a thread penetrate into previously covered areas, the excess traffic can be terminated by extending the ability of the intermediate nodes, e.g. intermediate node A passes along a query to B. B terminates the thread because a different thread of the same query has been detected earlier.



ZRP - Query Control Mechanisms

Query Detection

- Only the node that bordercasts a route request is aware that its zone is covered by the query
- When the peripheral nodes continue to bordercast to their peripheral nodes, the query may be relayed through the same nodes again
- ZRP provides two query detection methods, QD1 and QD2, to notify the remaining nodes through some form of eavesdropping without incurring additional control traffic



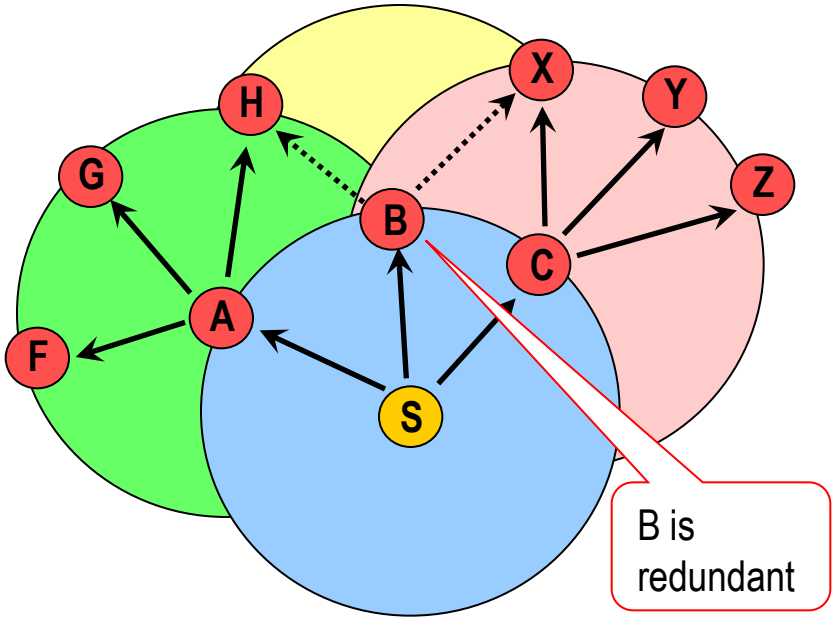
QD1 - Allows intermediate nodes (which relay queries to the edge of the routing zone) to detect queries, e.g., A and C can detect passing route request packet and record that S's routing zone has been queried

QD2 - In single channel networks, it is possible for queries to be detected by any node within range of a query transmitting node, e.g., E may be able to receive C's transmission and record the query information

ZRP – Query Control Mechanisms

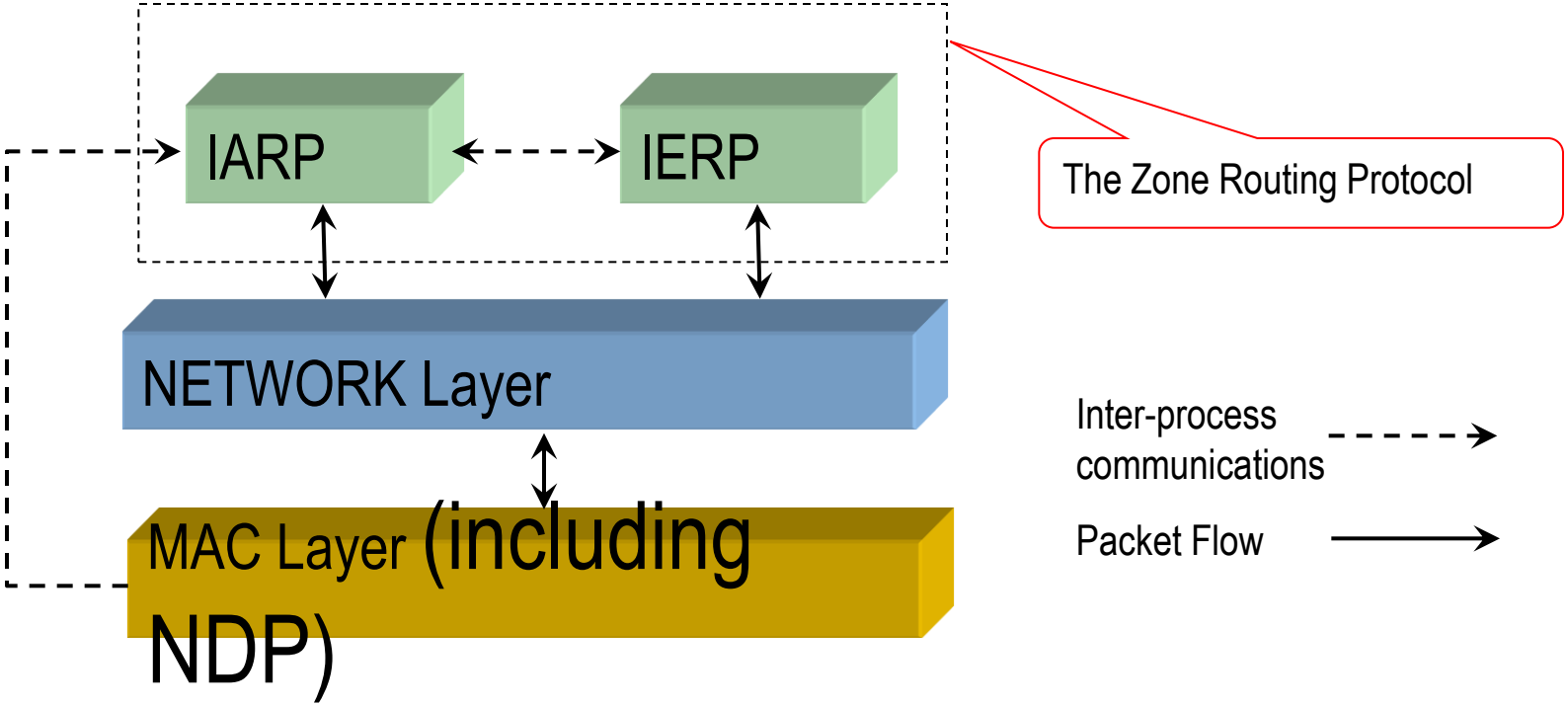
Selective Bordercasting (SBC)

- Rather than bordercast queries to all peripheral nodes, the same coverage can be provided by bordercasting to a chosen subset of peripheral nodes
- Requires IARP to provide network topology information for an extended zone that is twice the radius of the routing zone
- A node will first determine the subset of other peripheral nodes covered by its assigned inner peripheral nodes
- The node will then bordercast to this subset of assigned inner peripheral nodes which forms the minimal partitioning set of the outer peripheral nodes



- S's inner peripheral nodes are A, B and C
- Its outer peripheral nodes are F, G, H, X, Y and Z
- Two inner peripheral nodes of B (H and X) are also inner peripheral nodes of A and C
- S can then choose to eliminate B from its list of bordercast recipients since A can provide coverage to H and C can cover X

ZRP - Architecture



- Route updates are triggered by the MAC-level Neighbor Discovery Protocol (NDP)
- IARP is notified when a link to a neighbor is established or broken
- IERP reactively acquires routes to nodes beyond the routing zone
- IERP forwards queries to its peripheral nodes (BRP) keeping track of the peripheral nodes through the routing topology information provided by IARP

ZRP - Summary

- ZRP combines two completely different protocols, one proactive and the other reactive, into a single protocol based on clustering of nodes into *routing zones*
- Proactive IARP maintains routing tables within a routing zone
- Reactive IERP performs route discovery outside the zone
- ZRP can perform worse than flooding without proper query control mechanisms.
- *Query Detection*, *Early Termination*, and *Loop-back Termination* provide significant improvements compared with purely reactive and purely proactive schemes
- Use of *Selective Bordercasting* reduces significantly the amount of inter-zone control traffic

Ad Hoc Routing

- Sample Protocols
  - **Table Driven / Proactive**: DSDV
  - **On-Demand-Driven Reactive**: AODV, DSR
  - **Hybrid**: ZRP
  - **Geographic Routing**:
    Greedy, Face, GFG/GPSR, LAR
  - **Hierarchical Routing**:
    CBRP
  - **Other Routing**

Geographic routing

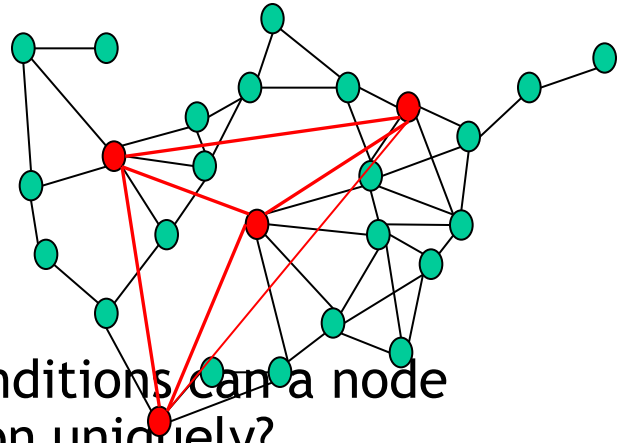
- Routing tables contain information to which next hop a packet should be forwarded
  - Explicitly constructed
- Alternative: Implicitly *infer* this information from physical placement of nodes
  - Position of current node, current neighbors, destination known – send to a neighbor in the right direction as next hop
  - ***Geographic routing***, ***geometric routing*** and ***position-based routing*** (use position information to aid in routing)
- Options
  - Send to any node in a given area – ***Geocasting***

Location Information

- Consider a node S that needs to find a route to node D.
- Assumption:
  - each host in the ad hoc network knows its current location precisely
  - node S knows that node D's location
- Might need a ***location service*** to map node ID to node position
- Location services in ad hoc networks, refer to
  - A survey on position-based routing in mobile ad hoc networks, M. Mauve, J. Widmer, and H. Hartenstein, IEEE Network, Vol. 15 No. 6, 2001.

Localization

- Problem: Given the positions of beacons, and some relative distances between nodes, determine the positions of the nodes

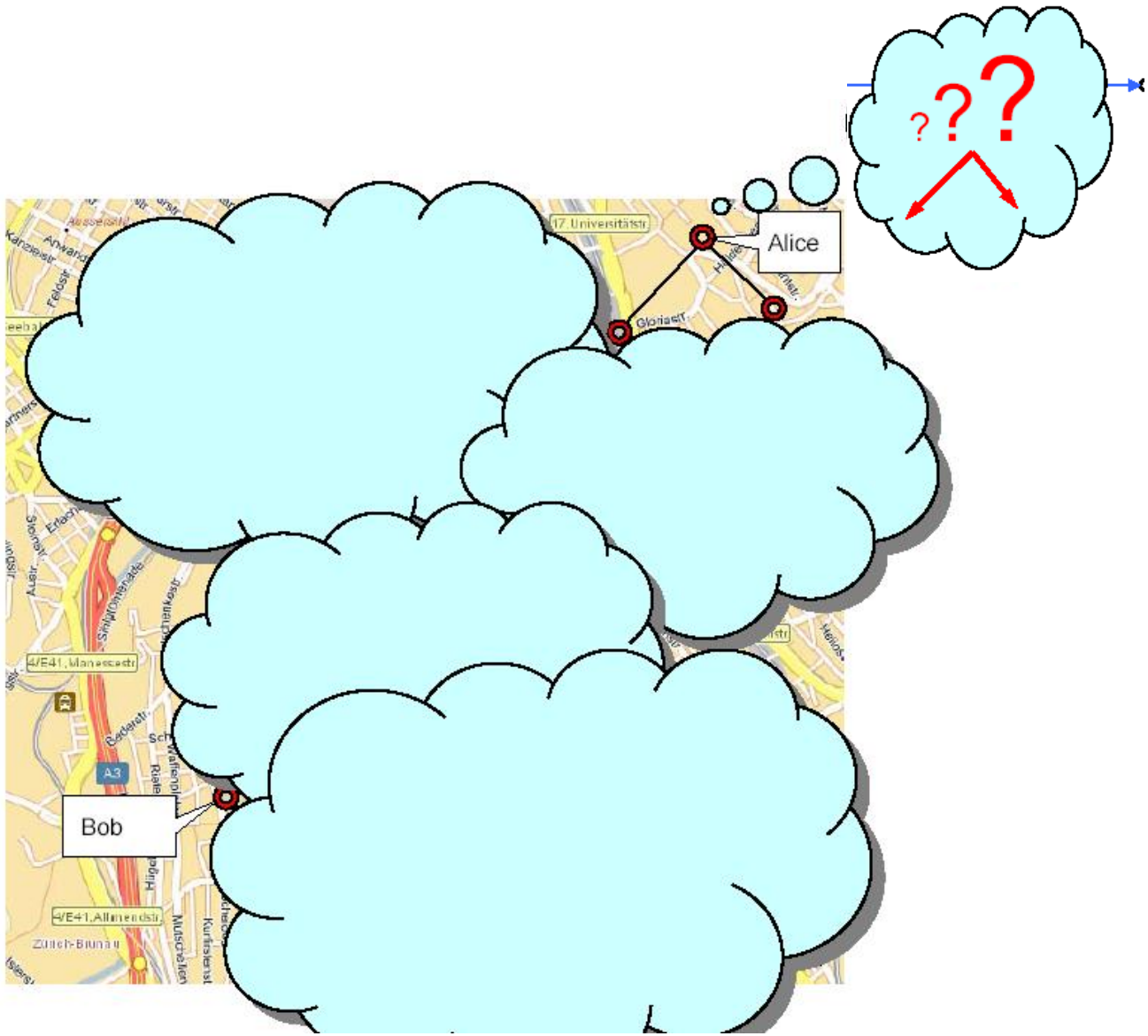


- Two questions
  - Localizability: under what conditions can a node uniquely determine its location uniquely?
  - Computation: how to determine the location?

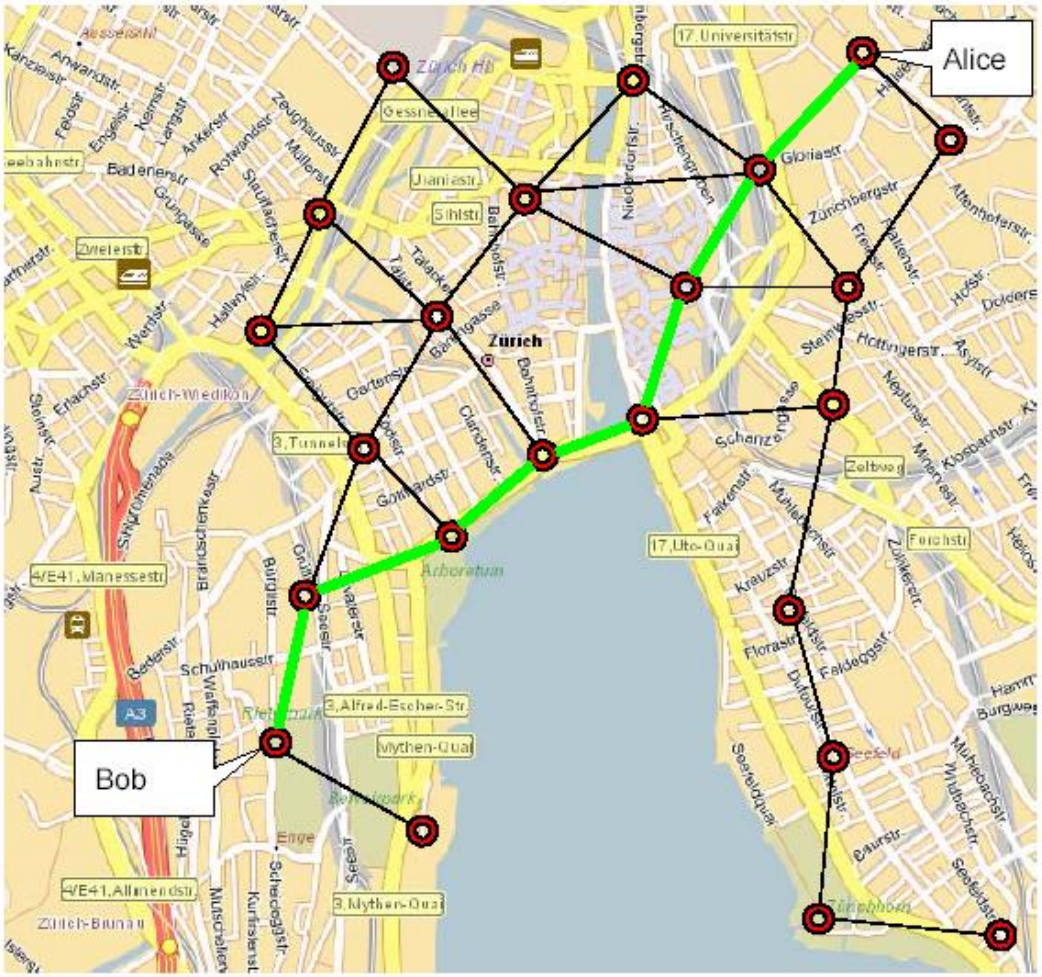
Localized Routing

- Also called online routing
- Every node can make decision based on local info, do not need to maintain routing table

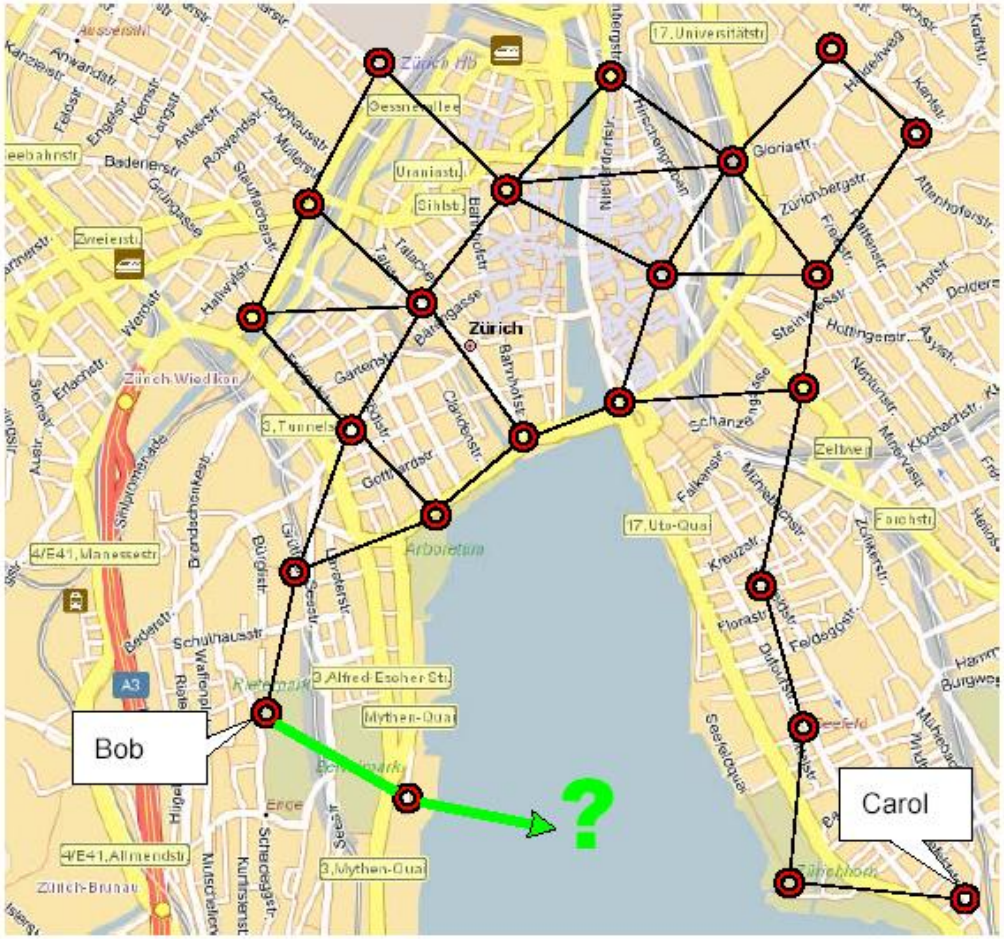
Strictly Local



Greedy Routing

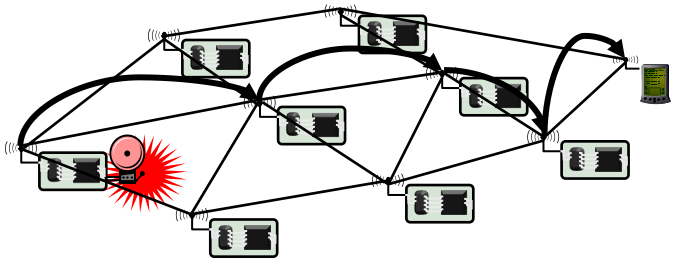


Greedy Routing ?

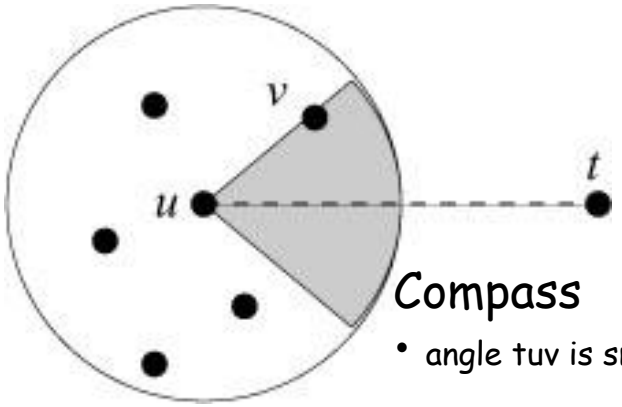


Basics of position-based routing

- "Most forward within range r" strategy
  - Send to that neighbor that realizes the most forward progress towards destination
  - NOT: farthest away from sender!

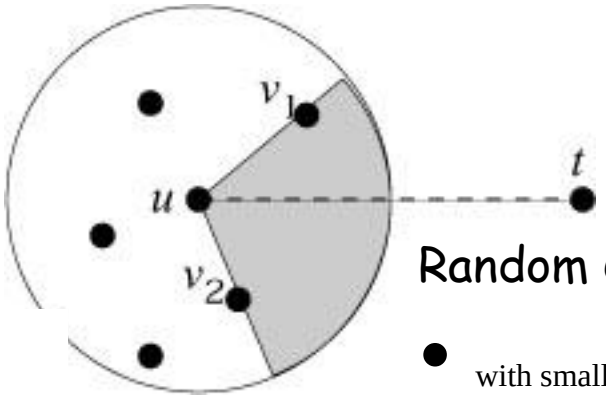


- Nearest node with (any) forward progress
  - Idea: Minimize transmission power
- Directional routing /Compass routing
  - Choose next hop that is angularly closest to destination
  - Choose next hop that is closest to the connecting line to destination
  - Problem: Might result in loops!



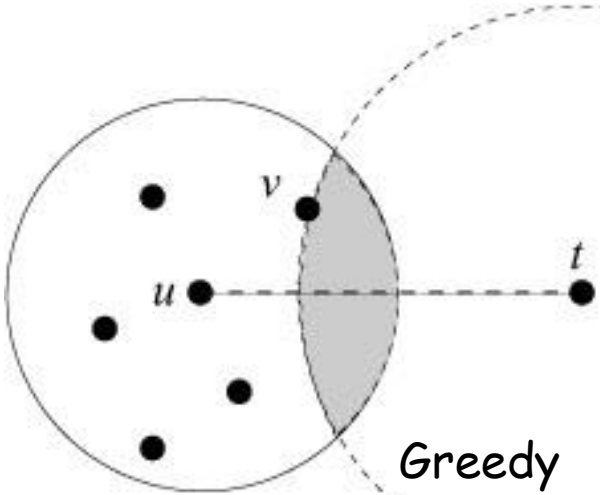
Compass

- angle tuv is smallest



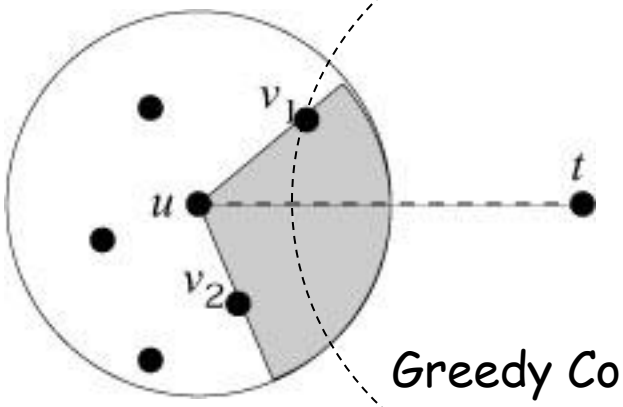
Random Compass

- with smallest angle either clockwise or couterclockwise

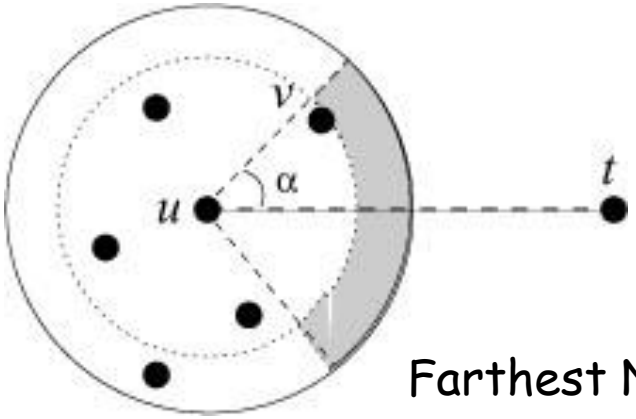


Greedy

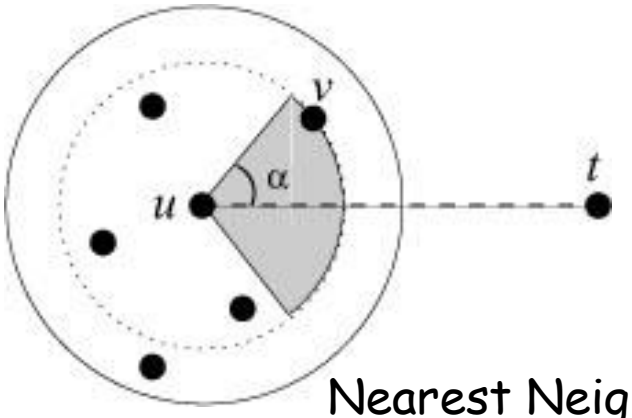
•Find v with min |vt|
And |uv|<R



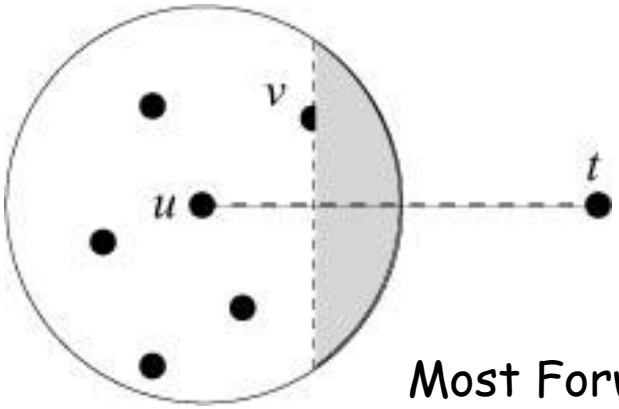
Greedy Compass



Farthest Neighbor



Nearest Neighbor

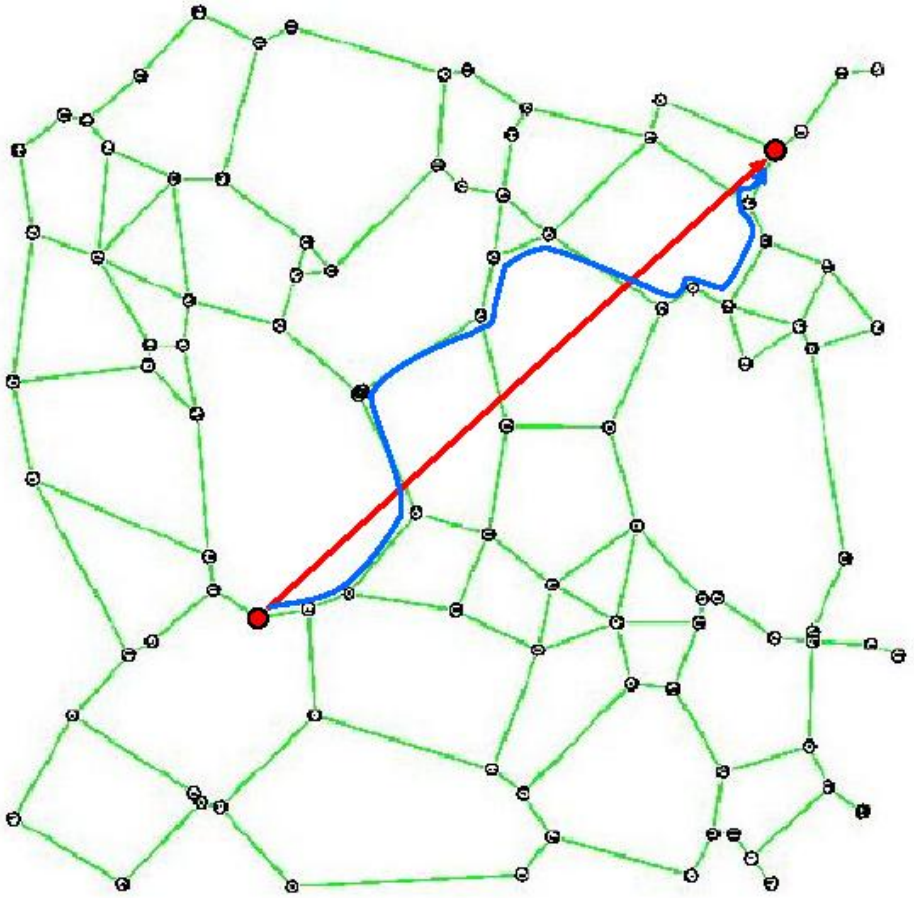


Most Forwarding

Greedy Routing

- Greedy routing looks promising

- Maybe there is a way to choose the next neighbor and a particular graph where we always reach the destination?



Greedy Fails

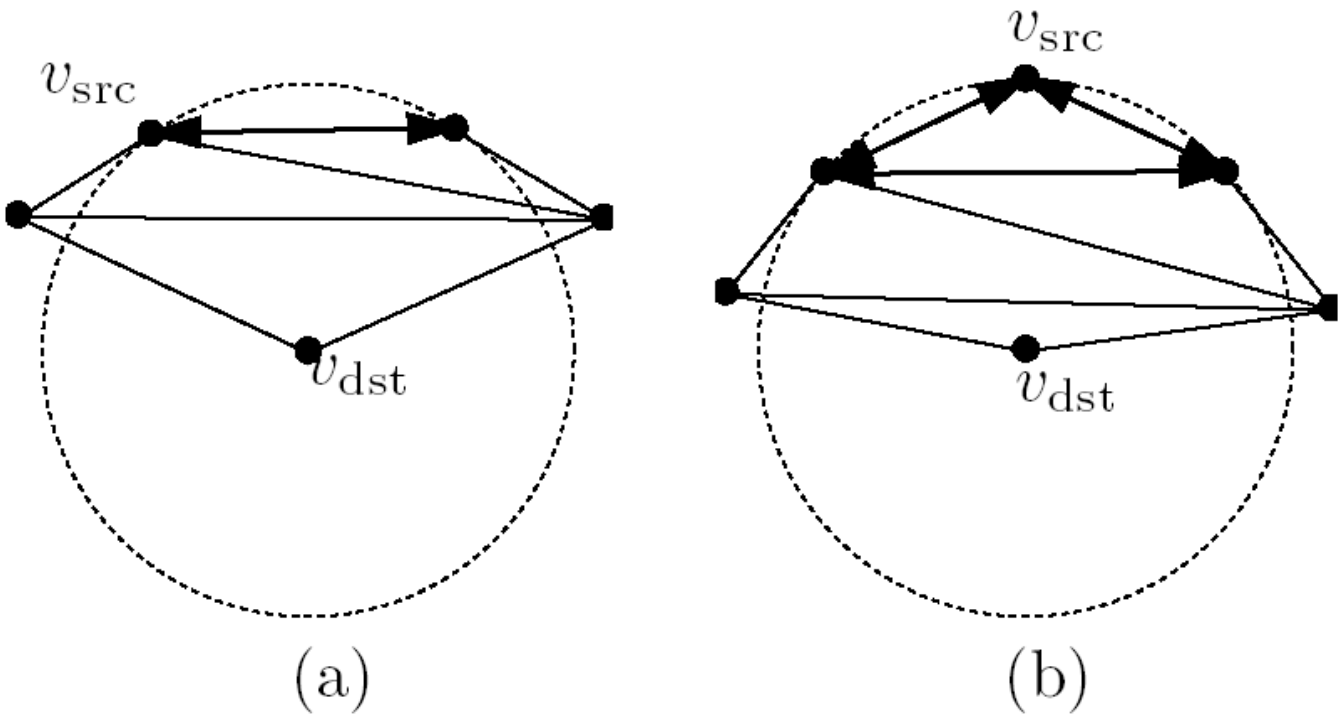
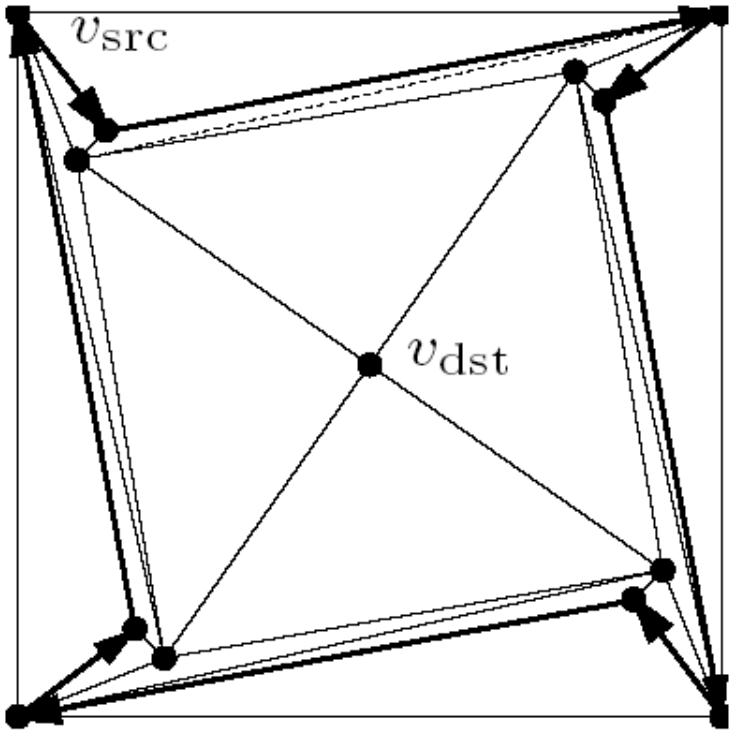
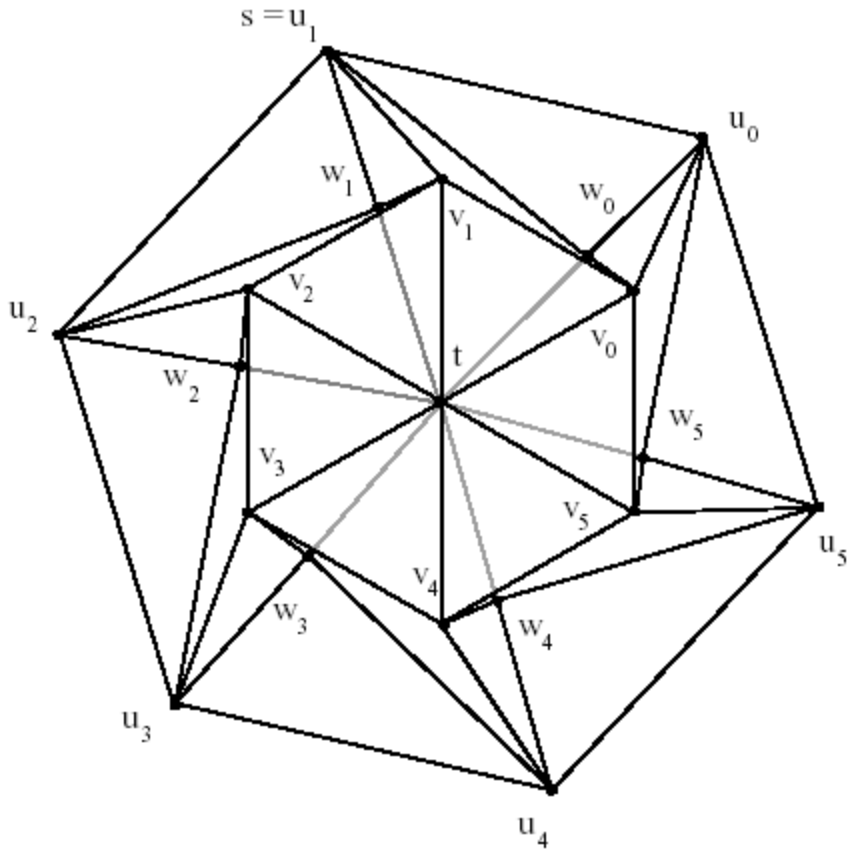


FIG. 2.2. *Triangulations that defeat the greedy routing algorithm.*

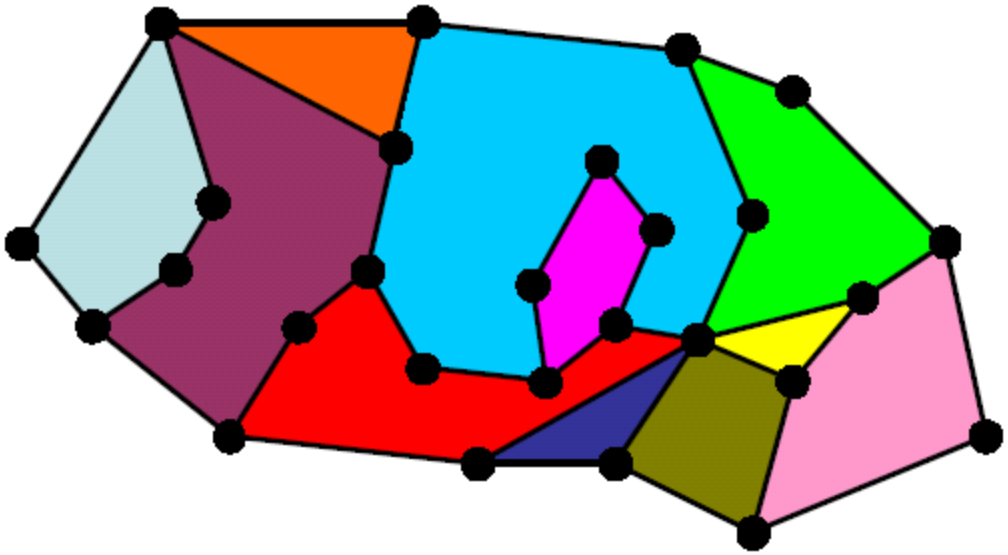
Compass Fails





Face Routing

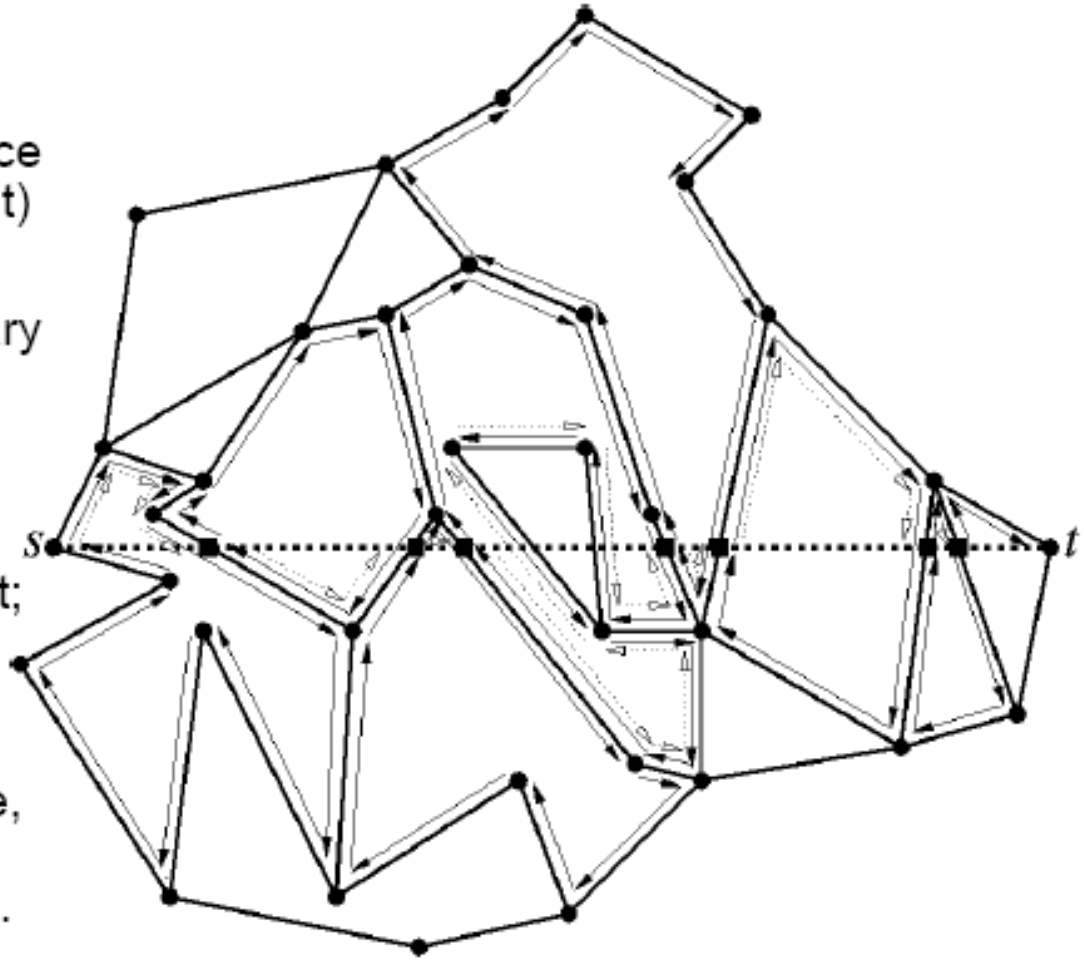
- To avoid void, use face routing (using right hand rule) on planar graphs



Face Routing

0. Let f be the face incident to the source s, intersected by (s,t)

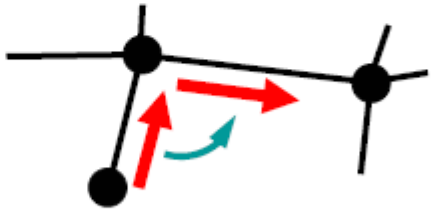
1. Explore the boundary of f; remember the point p where the boundary intersects with (s,t) which is nearest to t; after traversing the whole boundary, go back to p, switch the face, and repeat 1 until you hit destination t.



Face Routing Properties

- All necessary information is stored in the message
  - Source and destination positions
  - Point of transition to next face

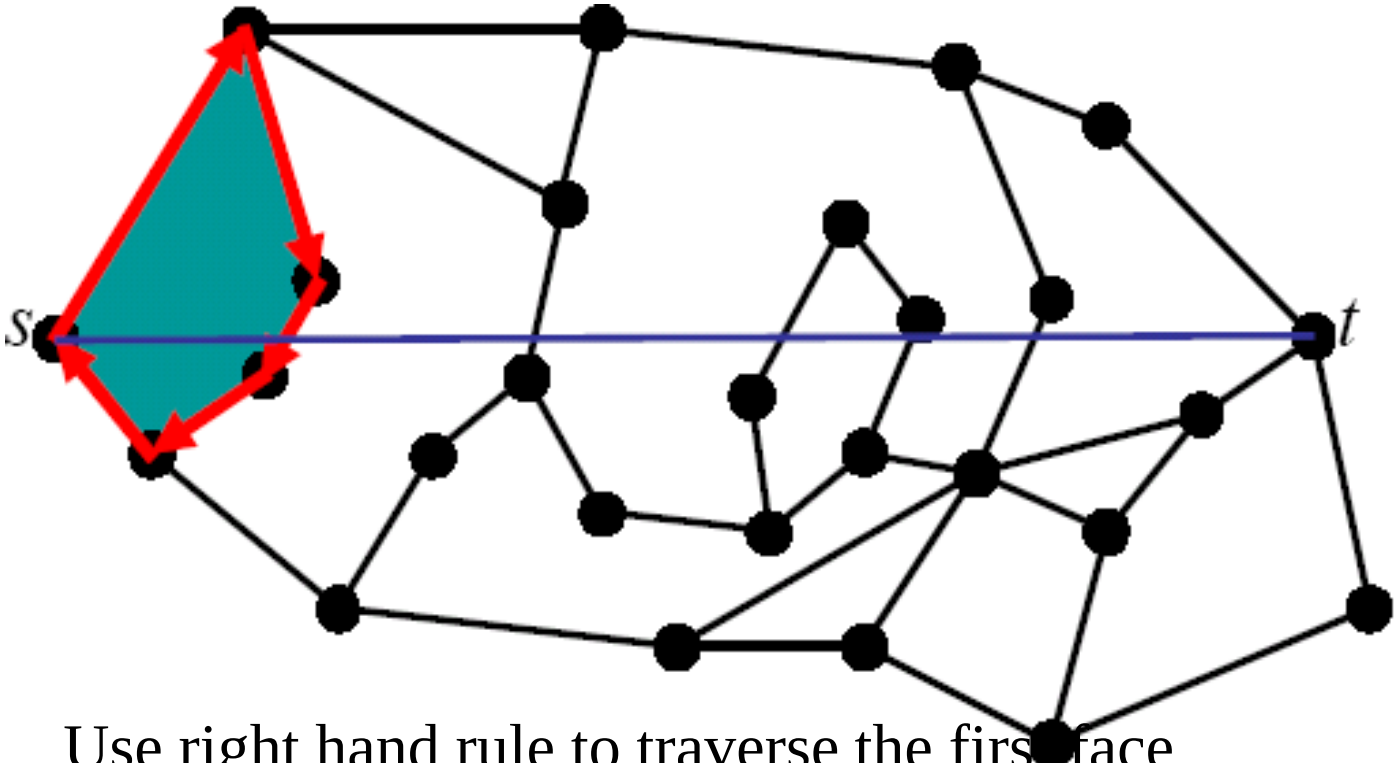
- Completely local:
  - Knowledge about direct neighbors‘ positions sufficient
  - Faces are implicit



“Right Hand Rule”

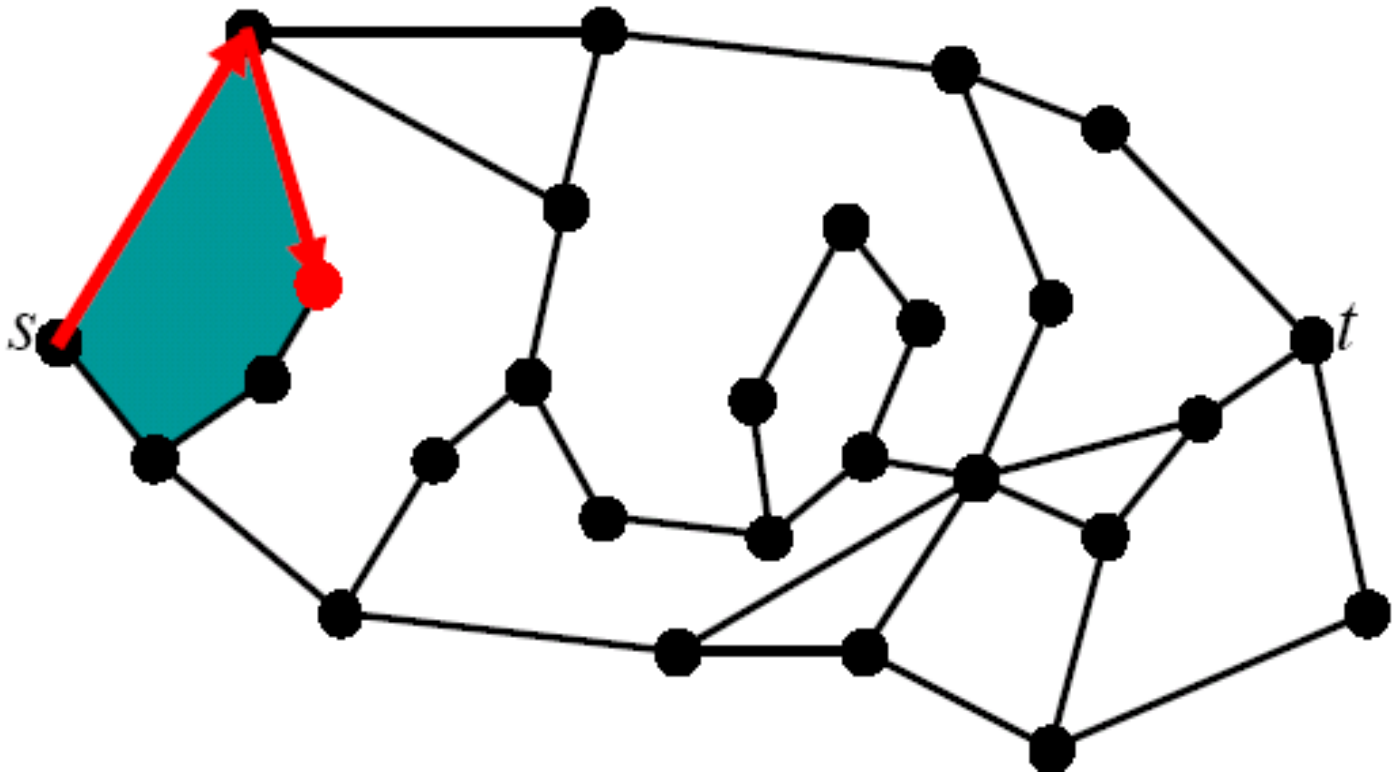
- Planarity of graph is computed locally (not an assumption)
  - Computation for instance with Gabriel Graph

Face Routing



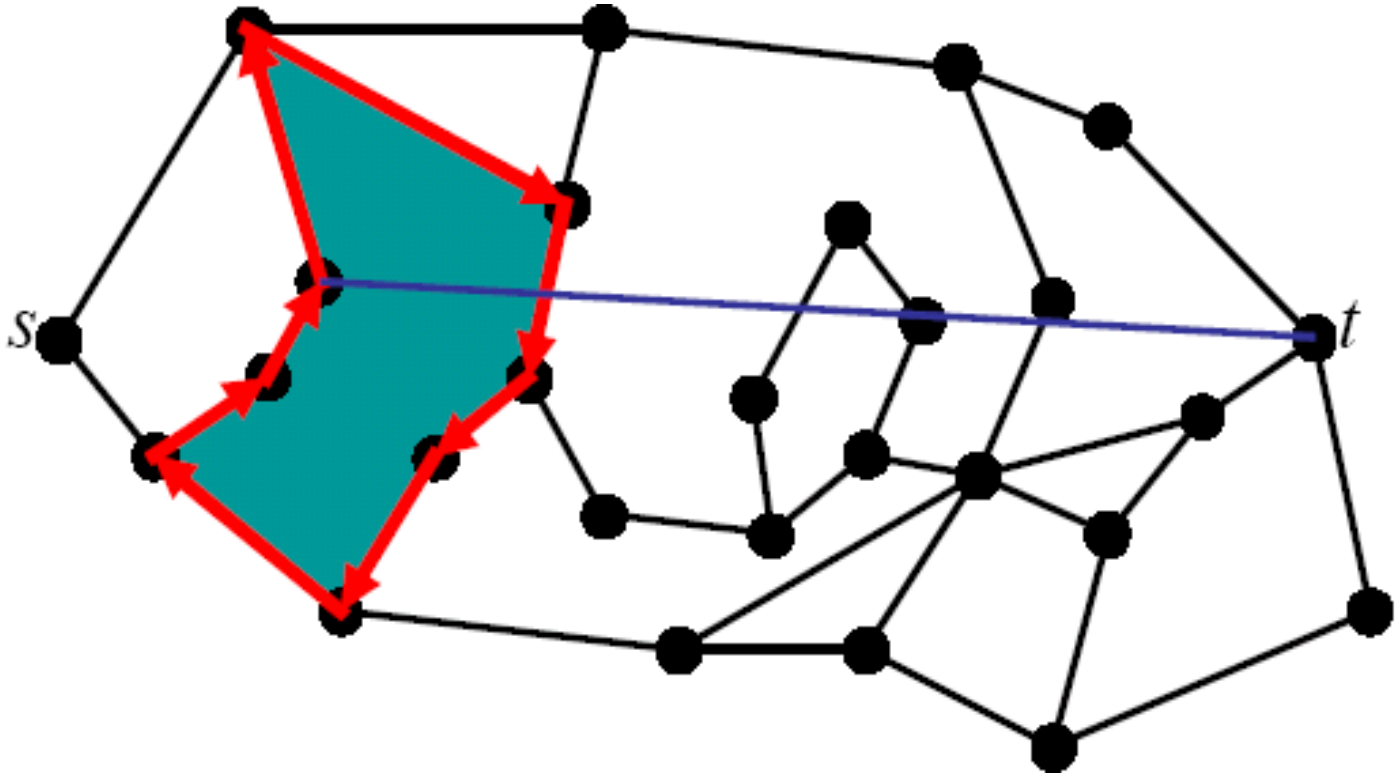
Use right hand rule to traverse the first face

Face Routing



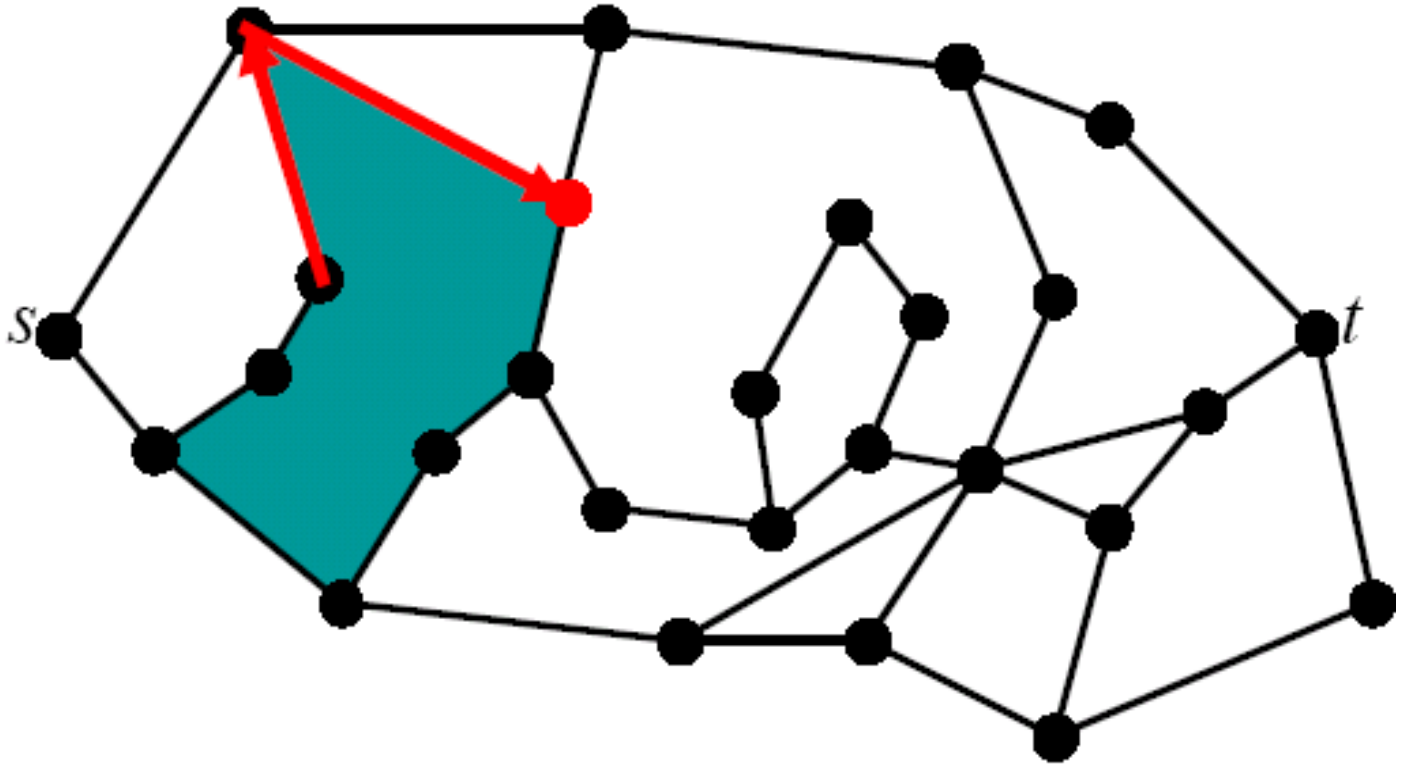
Go to the point that is furthest away from s

Face Routing

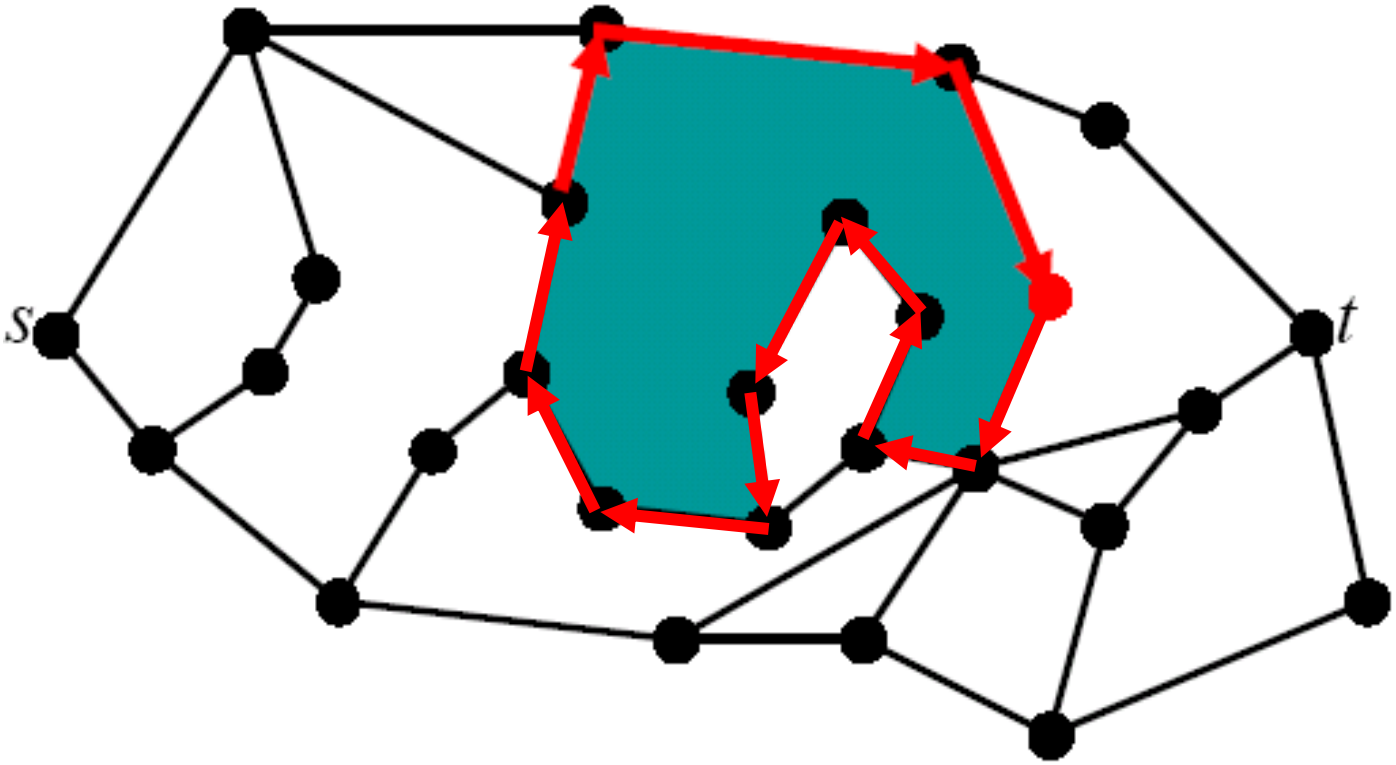


Use right hand rule to traverse the second face

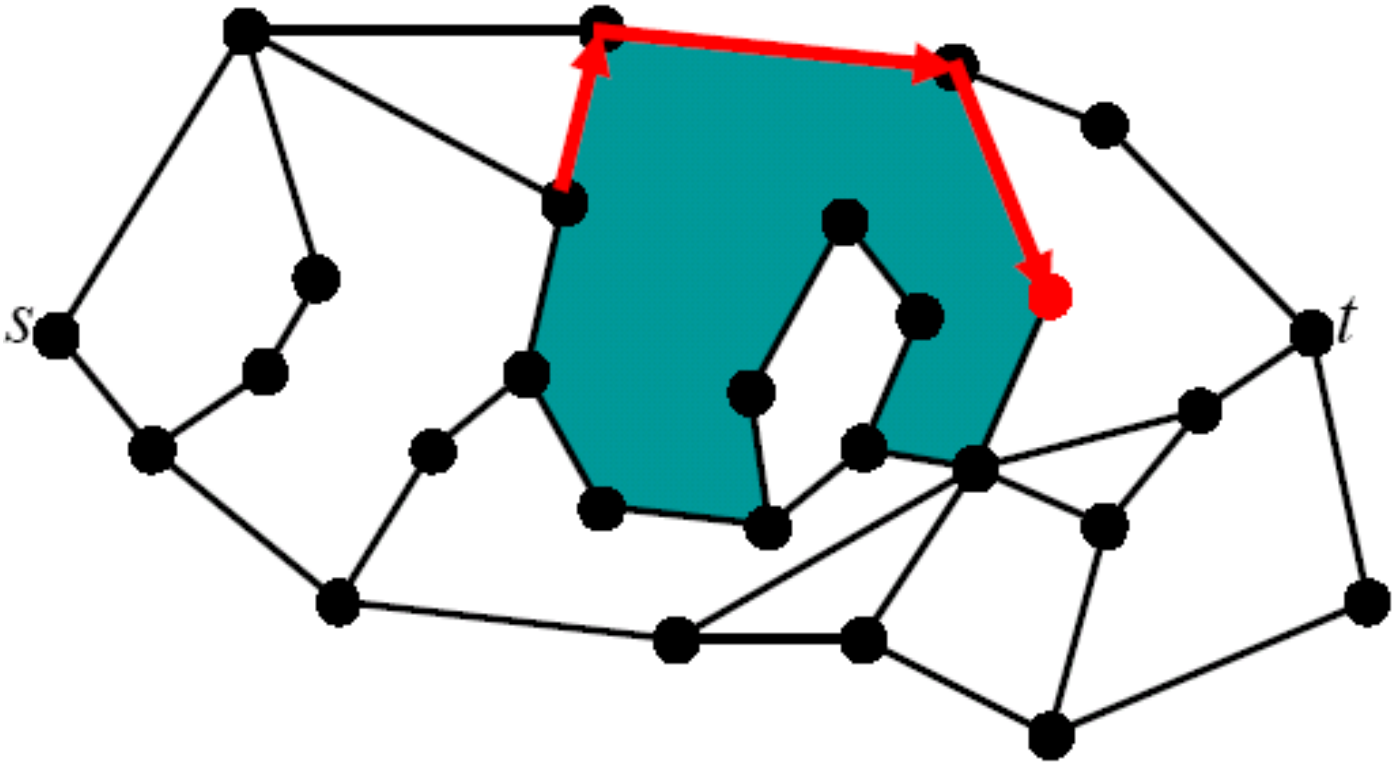
Face Routing



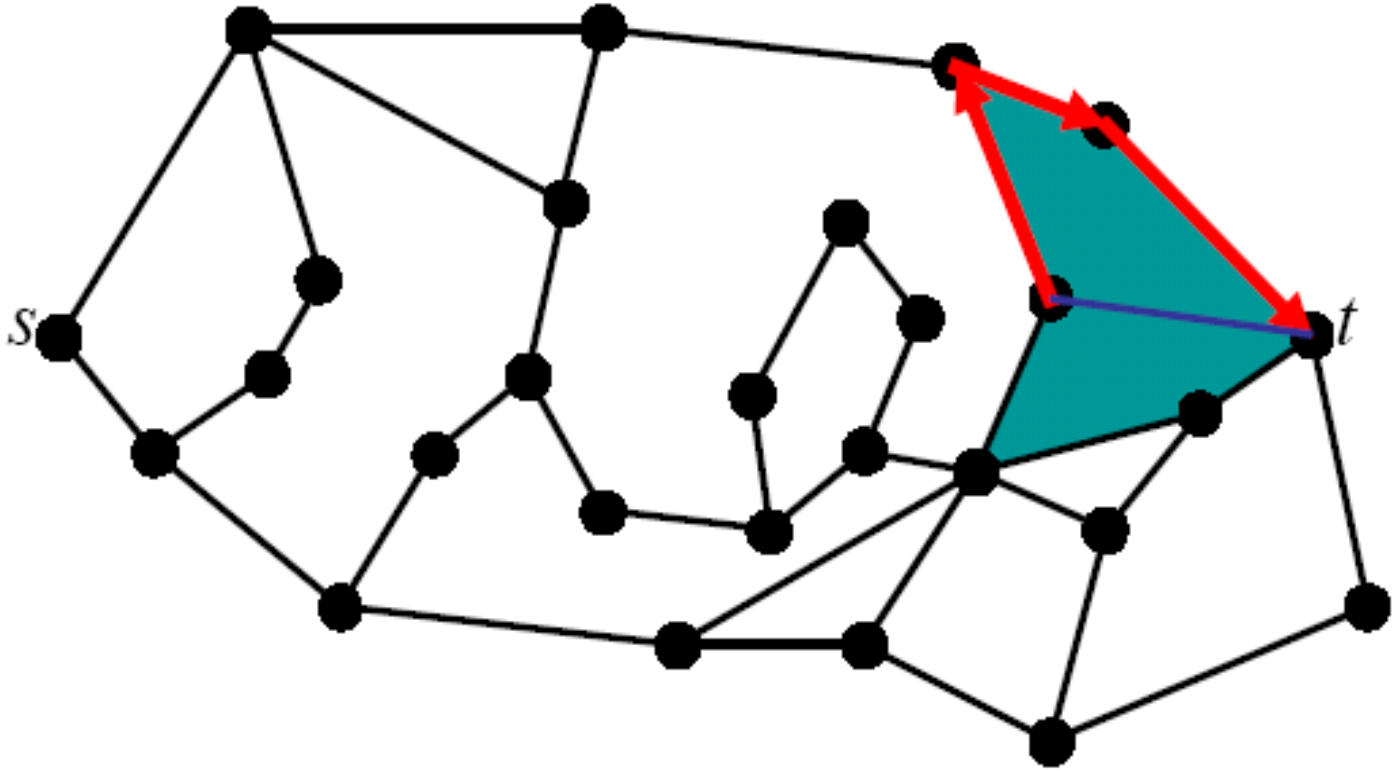
Face Routing



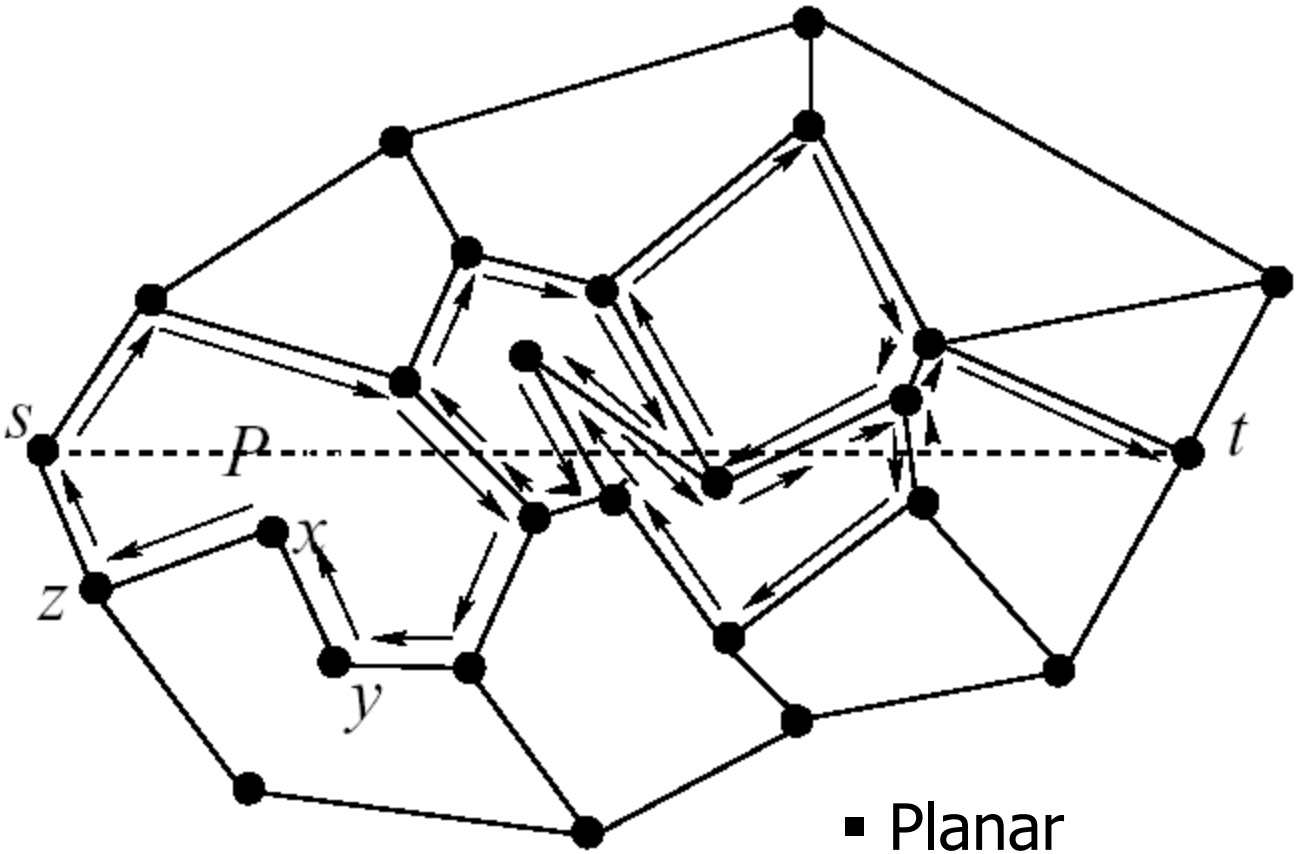
Face Routing



Face Routing



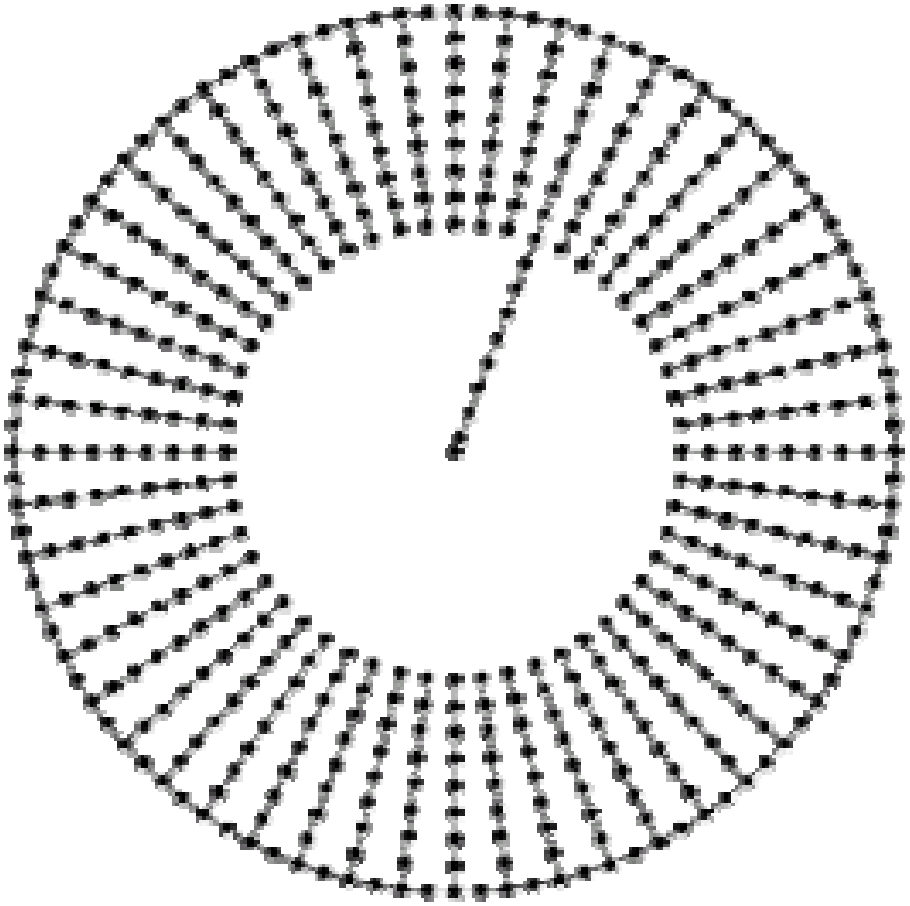
Face Routing



- Planar
- Guarantee the delivery
- Not localized

Performance of Face Routing in Terms of Hop Count

- A worst case scenario
  - destination is central node
  - source is any node on ring
  - any spine can go to middle
  - O(c) nodes along ring and O(c) nodes along each spine
- Best path length: O(c)+O(c)
- Geographic routing:
  - Test O(c) spines of length O(c)
  - Cost $O(c^2)$ instead of O(c)



Greedy Perimeter Stateless Routing

GPSR

Brad Karp and H.T.Kung
Harvard University

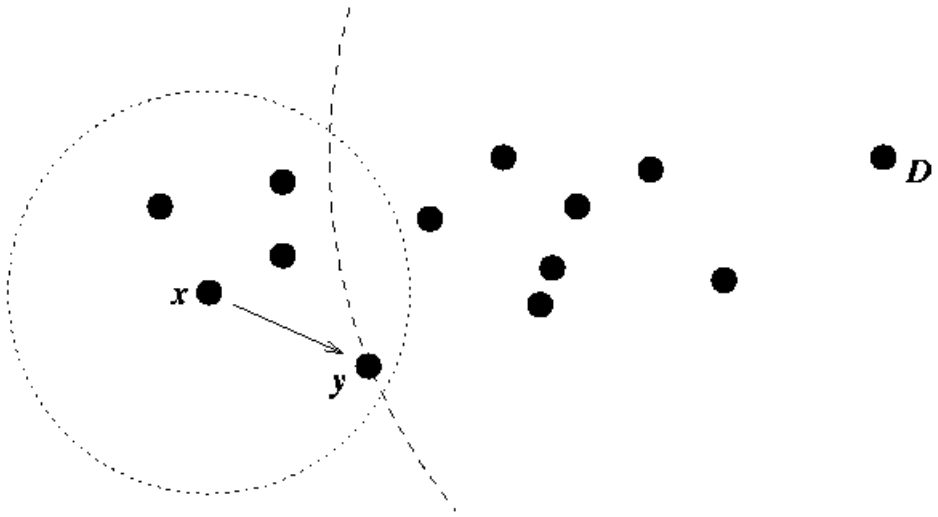
"GPSR: Greedy Perimeter Stateless Routing
 for Wireless Networks",
ACM Mobicom 2000.

GPSR

- Greedy Perimeter Stateless Routing for Wireless Networks.
- Use ***geography*** to achieve scalability.
- Reduced state requirement:
  - Traditional shortest-path (Distance Vector) requires state proportional to the total number of destinations.
  - On-demand ad-hoc routing require state proportional to the number of active destinations.
  - GPSR requires ***only*** single hop information. Depends only on the network density and not on the total number of destinations in the network.

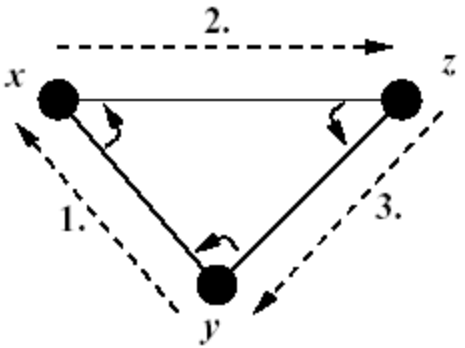
Greedy Mode

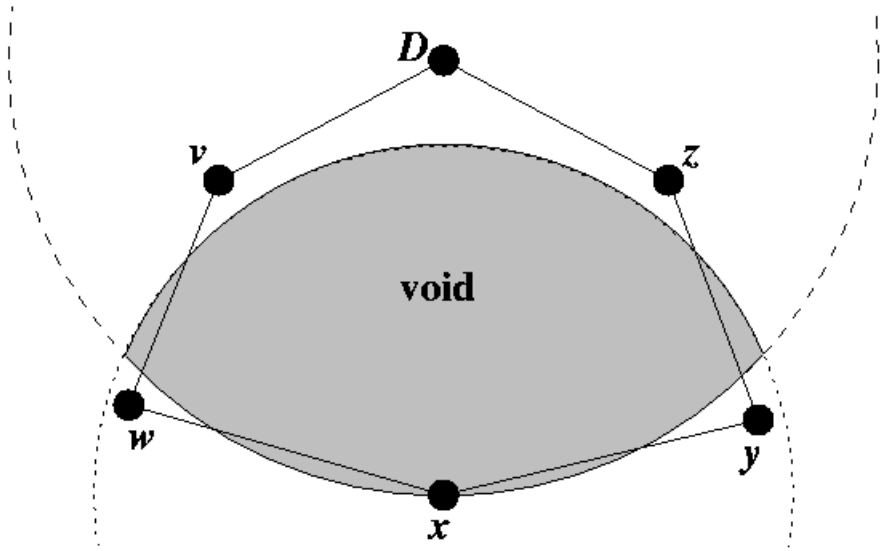
- Choose the next hop node, which is closest to destination.
  - Switch to Perimeter mode if local maxima occurs



Perimeter Mode

- allows GPSR to deal with holes (local maxima).
- planarization of the graph (get rid of cross-edges, GG and RNG).
- traverse progressively closer polygon to get out of the holes (right hand rule).
- after it progresses, i.e. closer than the point which greedy algorithm fails, returns back to greedy mode.



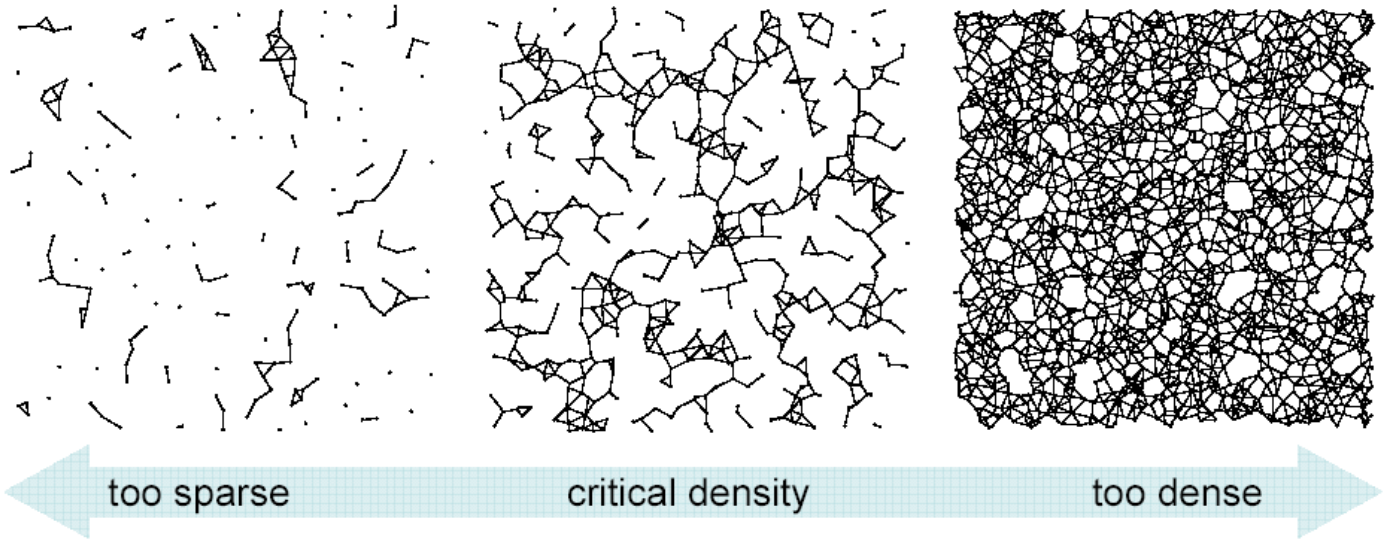


Geographic Routing

Kleinrock et al.	Various 1975ff	MFR et al.	Geometric Routing proposed
Kranakis, Singh, Urrutia	CCCG 1999	Face Routing	First correct algorithm
Bose, Morin, Stojmenovic, Urrutia	DialM 1999	GFG	First average-case efficient algorithm (simulation but no proof)
Karp, Kung	MobiCom 2000	GPSR	A new name for GFG
Kuhn, Wattenhofer, Zollinger	DialM 2002	AFR	First worst-case analysis. Tight $\Theta(c^2)$ bound.
Kuhn, Wattenhofer, Zollinger	MobiHoc 2003	GOAFR	Worst-case optimal and average-case efficient, percolation theory
Kuhn, Wattenhofer, Zhang, Zollinger	PODC 2003	GOAFR+	Currently best algorithm, other cost metrics, etc.

Average Case

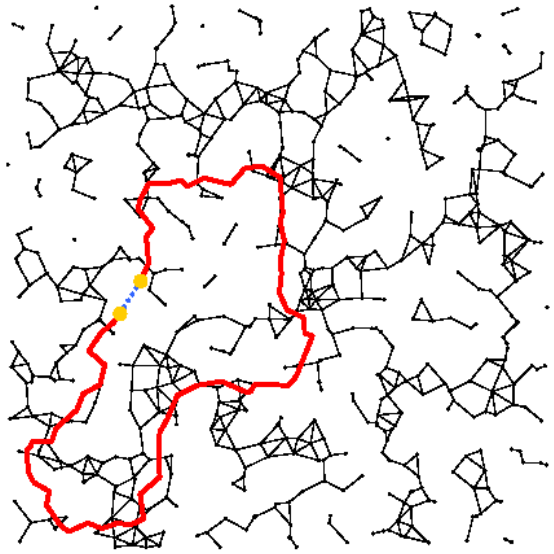
- Not interesting when graph not dense enough
- Not interesting when graph is too dense
- Critical density range ("percolation")
  - Shortest path is significantly longer than Euclidean distance



Shortest Path vs. Euclidean Distance

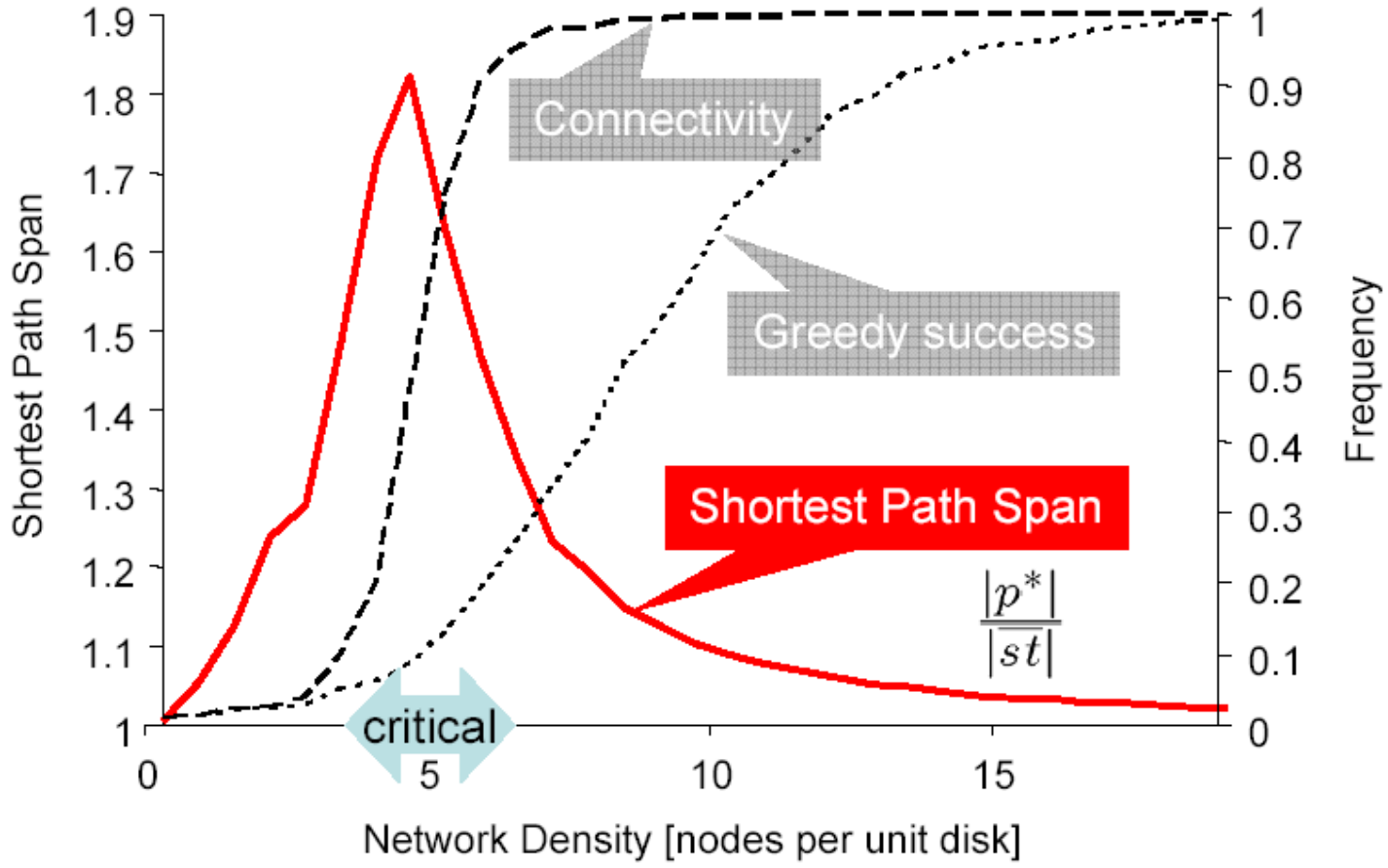
- Shortest path is significantly longer than Euclidean distance

$$\frac{|p^*|}{|\overline{st}|}$$

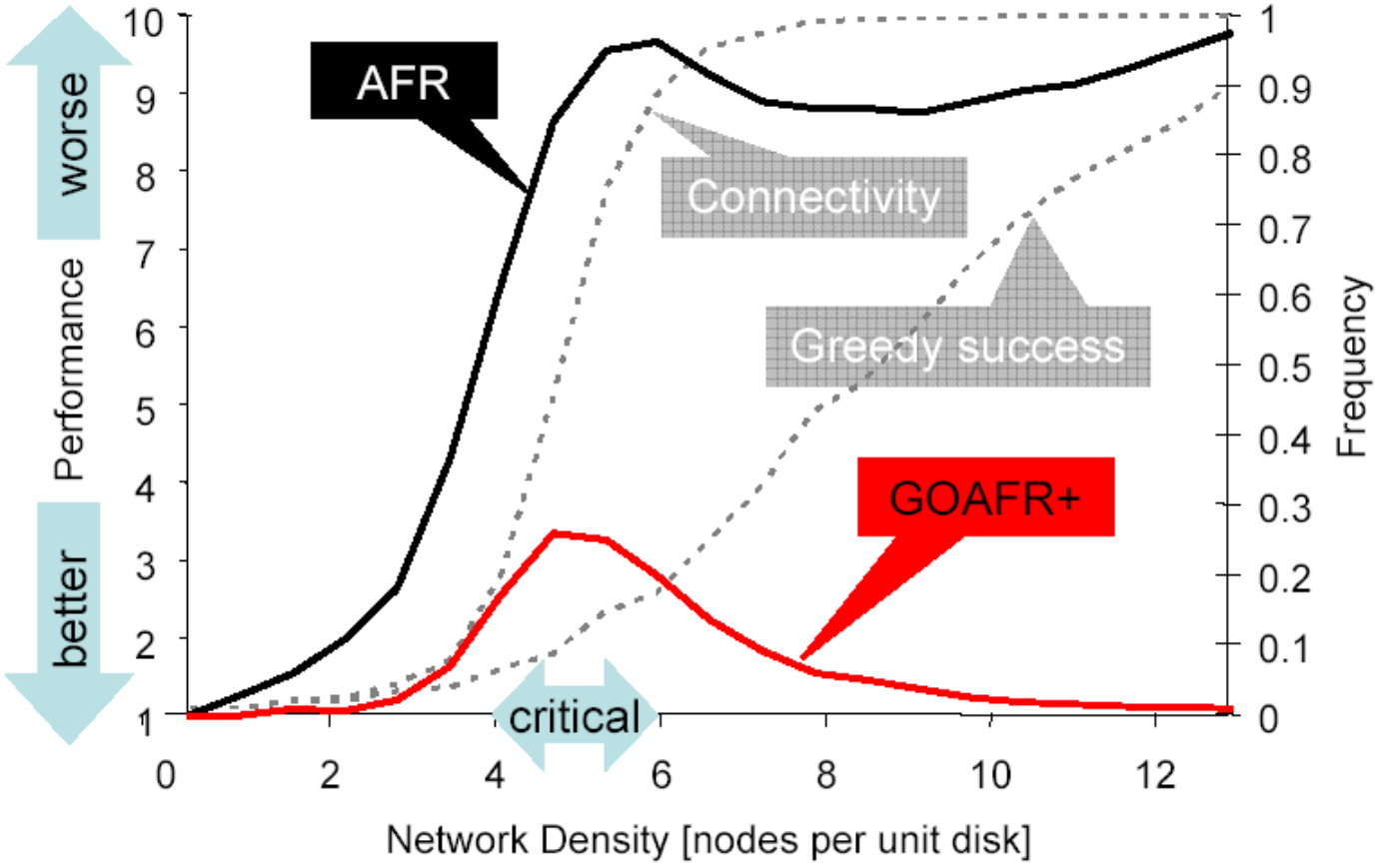


- Critical density range mandatory for the simulation of any routing algorithm (not only geographic)

Random Graphs: Critical Density Range



Simulation on Random Graphs

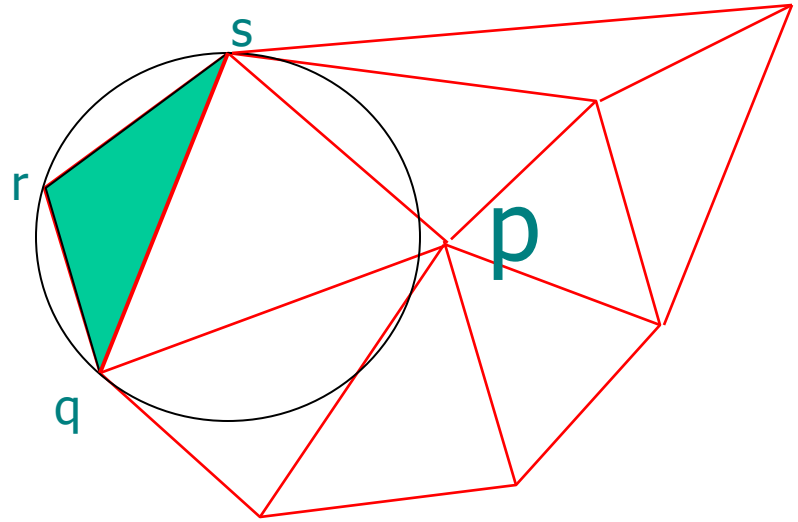


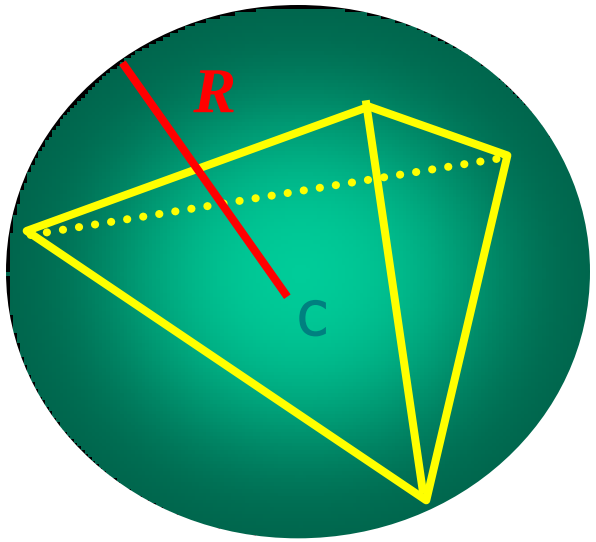
Routing on Delaunay

P. Bose and P. Morin
Carleton University

"Online routing in triangulations",
Annual Int. Symp. on Algorithms and Computation
ISAAC 99, 1999.

Delaunay Triangulation





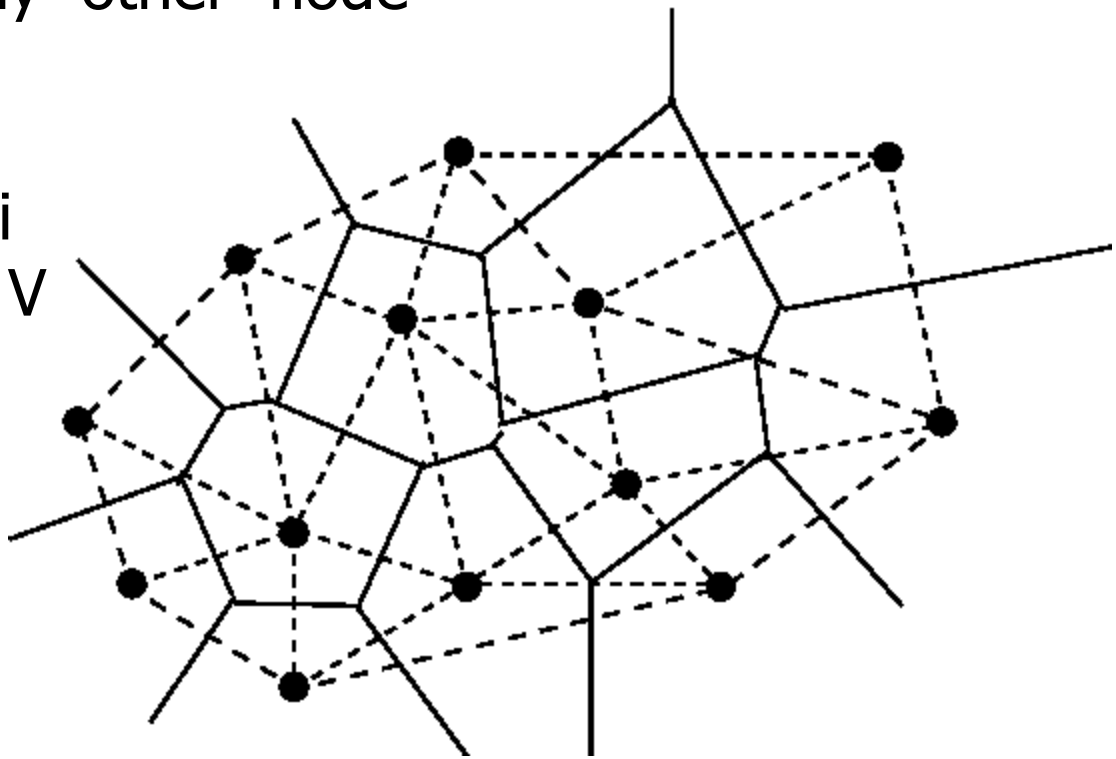
For every simplex (triangle in 2D, tetrahedron in 3D), circumsphere (c, R) is empty.

Delaunay vs Voronoi

Voronoi Region, Vor(p)
A collection of two dimensional points s.t. every point is closer to p than to any other node

Voronoi Diagram
The union of all Voronoi region Vor(p) where $p \in V$

Delaunay Triangulation
is the dual of
Voronoi Diagram



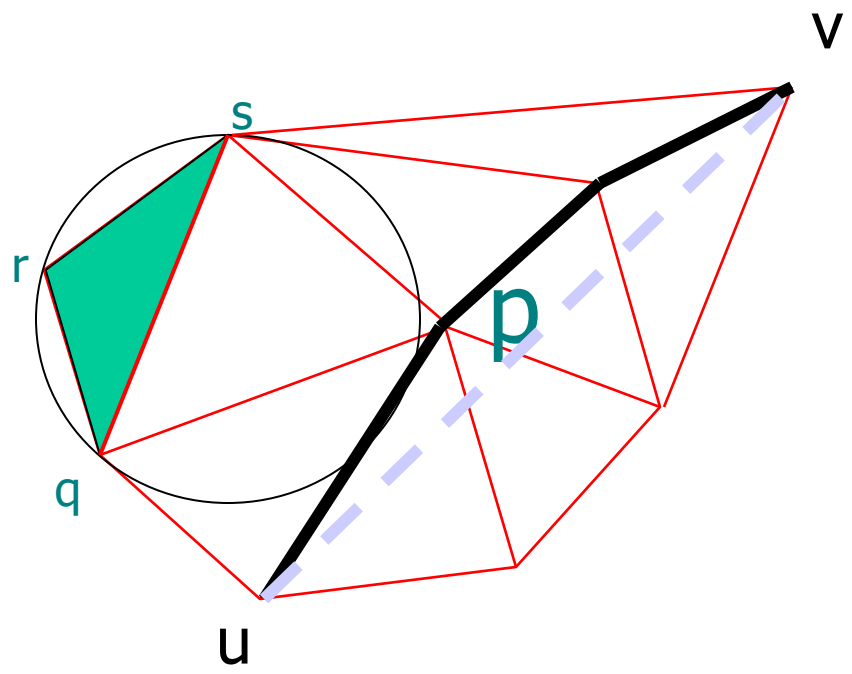
Ask for Delaunay

- Morin [2001] proved the following localized routing methods guarantee the delivery if Delaunay triangulation used as the underlying structure
  - Greedy routing
  - Compass routing
  - Greedy compass routing

Delaunay Triangulation

- Planar graph
  - No intersection
- Spanner
  - Constant Stretch Factor

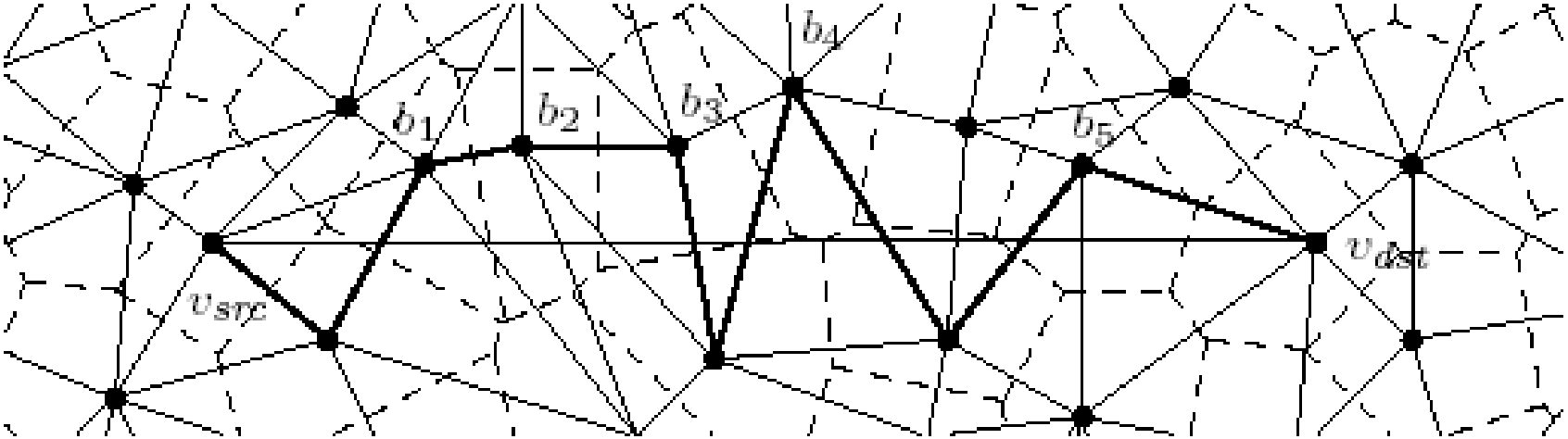
$$\frac{2\pi}{3\cos\frac{\pi}{6}} \approx 2.42$$



Routing on Delaunay

- Proposed by Bose and Morin [1999]
- Basic idea is to find a path in Del with length no more than $(1+\sqrt{5})\frac{\pi}{2}\|uv\|$ which consists of two parts:
  - Direct DT path
  - Shortcut path

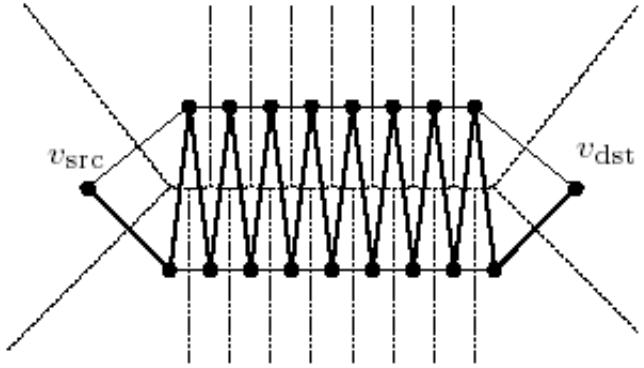
Routing on Delaunay



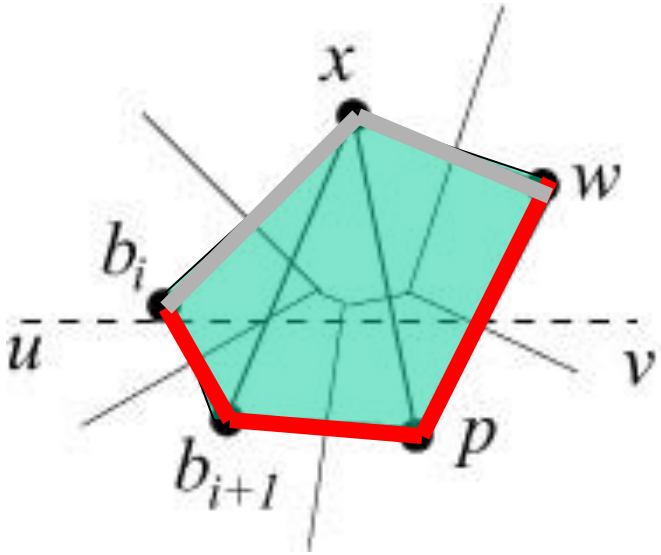
Direct DT path

Given two nodes *u* and *v*, let *b*0 = *u*, *b*1, *b*2, $\cdots$, *bm*−1, *bm* = *v* be the nodes corresponding to the sequence of Voronoi regions traversed by walking from *u* to *v* along the segment *uv*. If a Voronoi edge or a Voronoi vertex happens to lie on the segment *uv*, then choose the Voronoi region lying above *uv*.

Routing on Delaunay



Shortcut path



Shortcut is the upper boundary of the tunnel $T(u, v)$ that connects *bi* and *bj*

Routing on Delaunay

- Use direct DT path as long as it is above uv
- When some nodes $b_i>0$ and $b_{i+1}<0$
  - Use either direct DT path or shortcut path
  - Exploring both in parallel manner until one reaches next $b_j>0$
- Many detailed
  - How to find next node in direct DT path locally
  - How to find next node in shortcut path locally
  - How to determine whether node b_j is reached

Routing on Delaunay

- The distance traveled by the above routing method is $9c_{dfs}$-competive,
  here

$$c_{dfs} = (1+\sqrt{5})\frac{\pi}{2}$$

- However, Delaunay triangulation CONNOT be constructed locally
  - May need globe info
  - Cannot communicate through long edges

Routing on Local Delaunay

- Build Delaunay Locally w.h.p.
- When $\pi r_n^2 \geq \frac{8 \ln n}{n}$,
  - The longest edge in Del is at most r_n with probability $1 - \frac{1}{n}$,

    in other words, LDel=Del w.h.p.

- "Efficient Localized Routing for Wireless Ad Hoc Networks", X.-Y. Li, Y. Wang, and O. Frieder,
  IEEE ICC 2003.

Location-Aided Routing

LAR

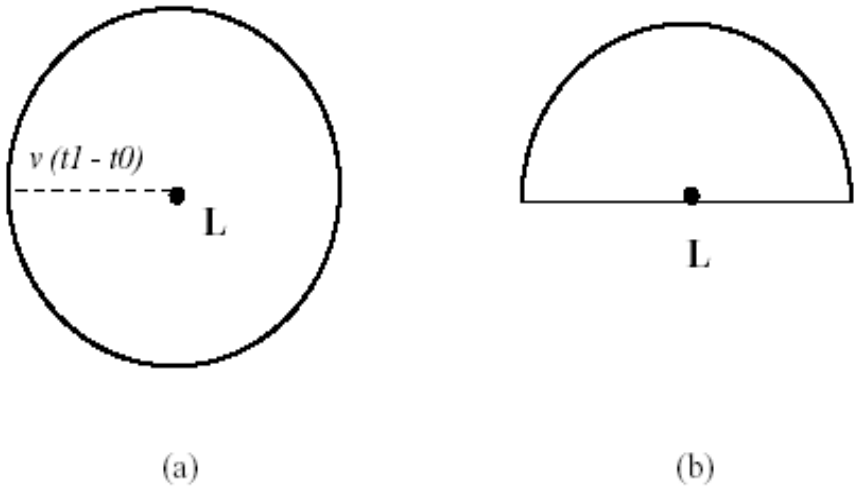
Young-Bae Ko and Nitin H. Vaidya
Texas A&M University

"Location-Aided Routing(LAR) in Mobile Ad Hoc Networks", ACM MOBICOM'98, 1998.

Location-Aided Routing

- Main Idea
  - Using location information to reduce the number of nodes to whom route request is propagated.
  - Location-aided route discovery based on “limited” flooding

Expected Zone



expected zone of D ---- the region that node S expects to contain node D at time t_1, only an estimate made by node S

Request Zone

- LAR's limited flooding
  - A node forwards a route request only if it belongs to the *request zone*
  - The request zone should include
    - expected zone
    - other regions around the expected zone
  - No guarantee that a path can be found consisting only of the hosts in a chosen request zone.
    - timeout
    - expanded request zone

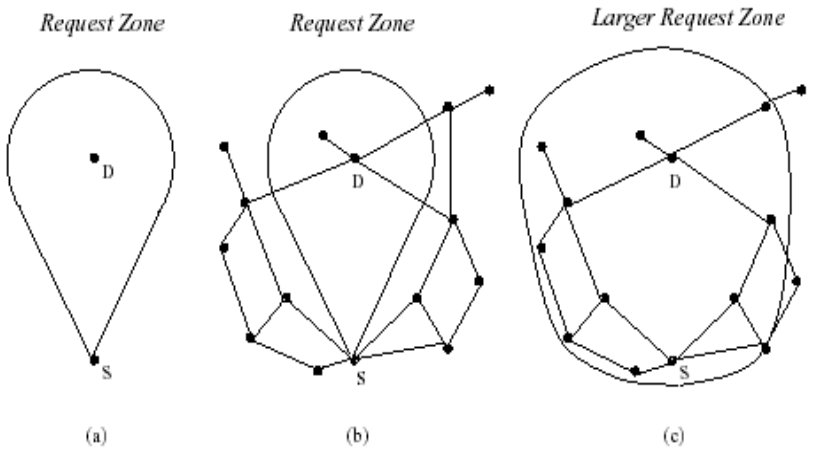


Figure 3: Request zone: An edge between two nodes means that they are neighbors

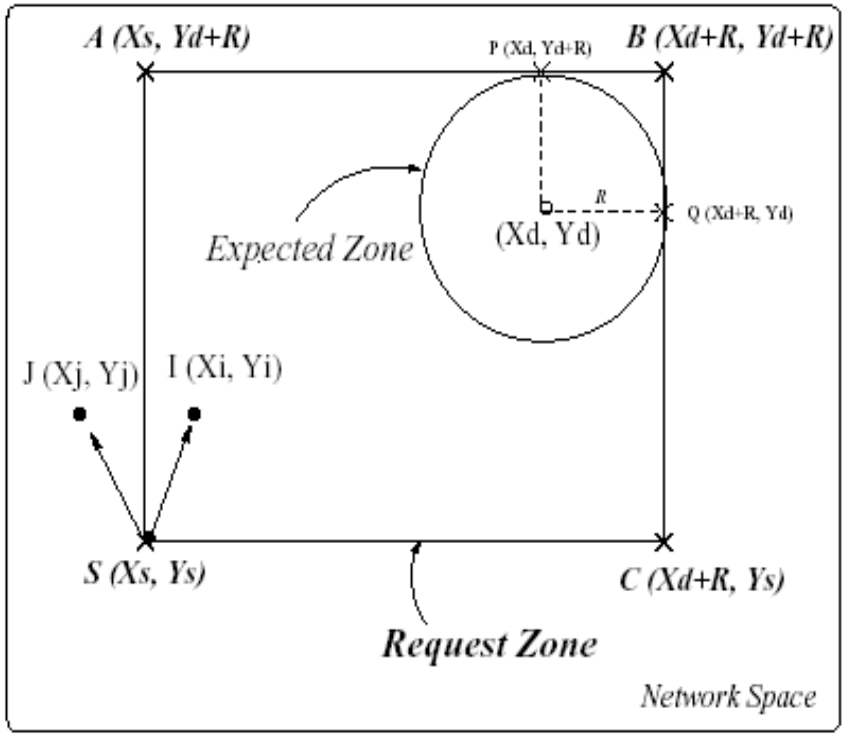
- Trade-off between
  - latency of route determination
  - the message overhead

Membership of Request Zone

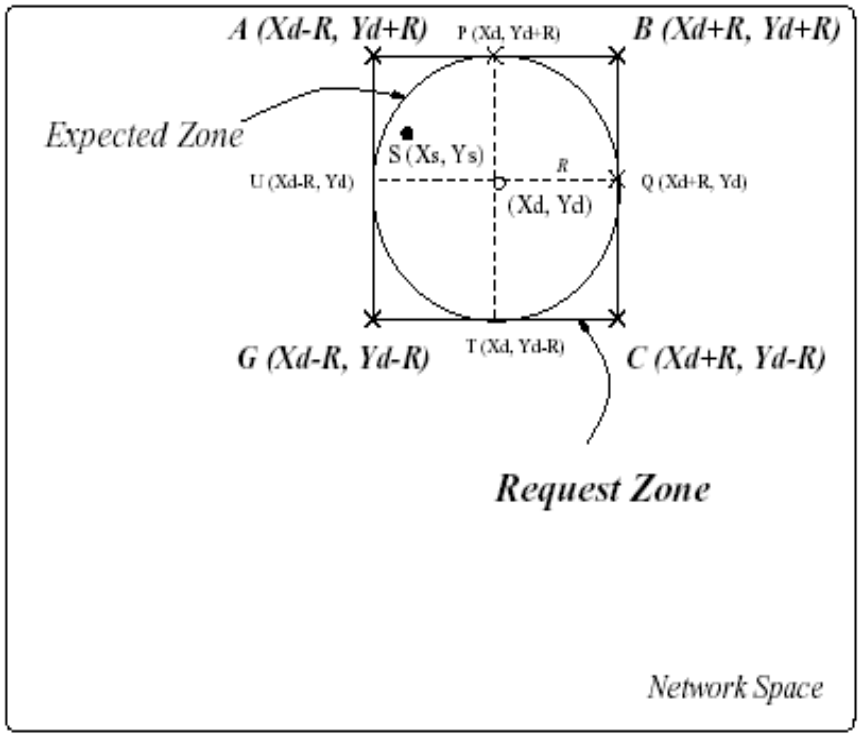
- How a node determine if it is in the request zone for a particular route request

- LAR scheme 1
- LAR scheme 2

LAR Scheme 1

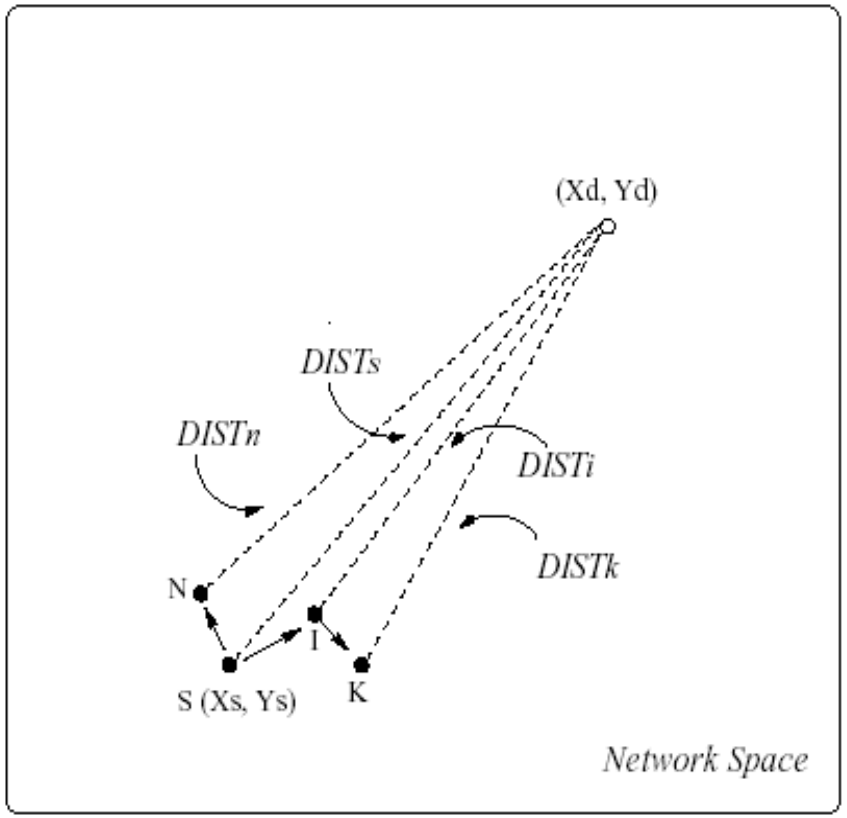


(a) Source node outside the Expected Zone



(b) Source node within the Expected Zone

LAR Scheme 2



S knows the location (X_d, Y_d) of node D at time t_0

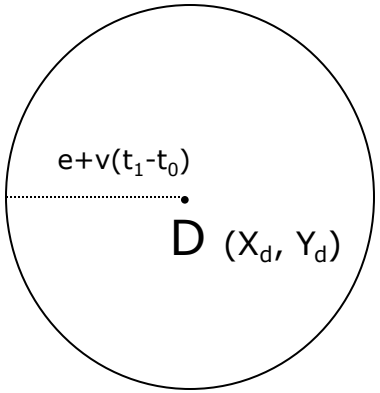
Node S calculates its distance from location (X_d, Y_d): $DIST_s$

Node I receives the route request, calculates its distance from location (X_d, Y_d): $DIST_i$

For some parameter δ,
If $DIST_s + \delta \geq DIST_i$, node I replaces $DIST_s$ by $DIST_i$ and forwards the request to its neighbors; otherwise discards the route request

Error in Location Estimate

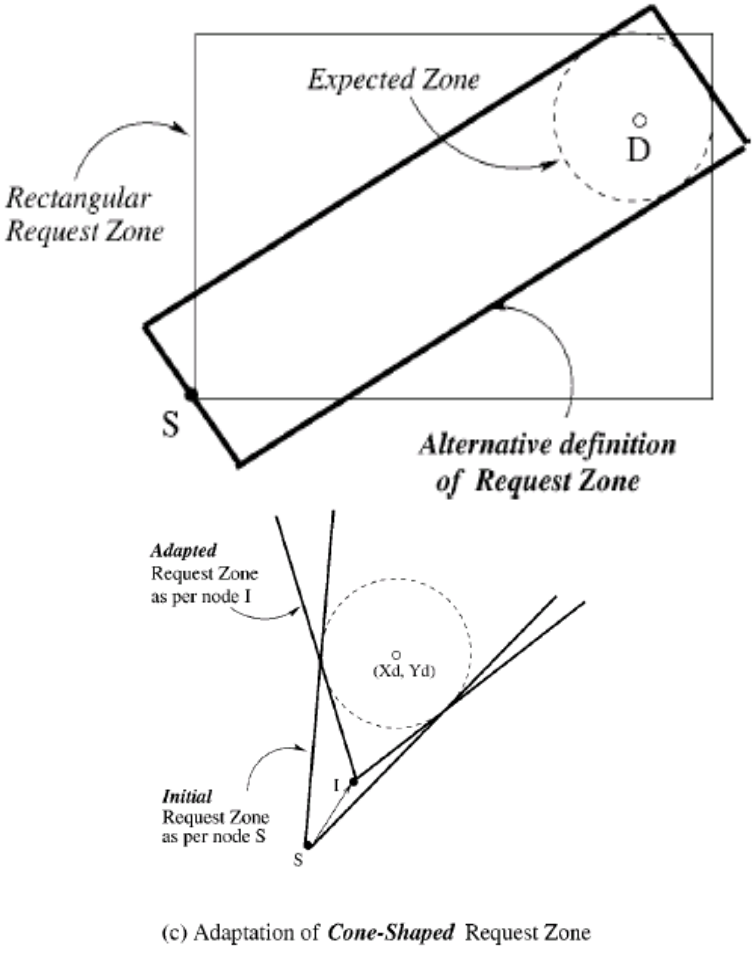
- Let e denote the maximum error in the coordinates estimated by a node.
- Modified LAR scheme 1



Expected Zone

Variations and Optimizations

- Alternative Definitions of Request Zone
  - increasing the request zone gradually?
- Adaptation of Request Zone
- Propagation of Location and Speed Information
- Local Search



(c) Adaptation of ***Cone-Shaped*** Request Zone

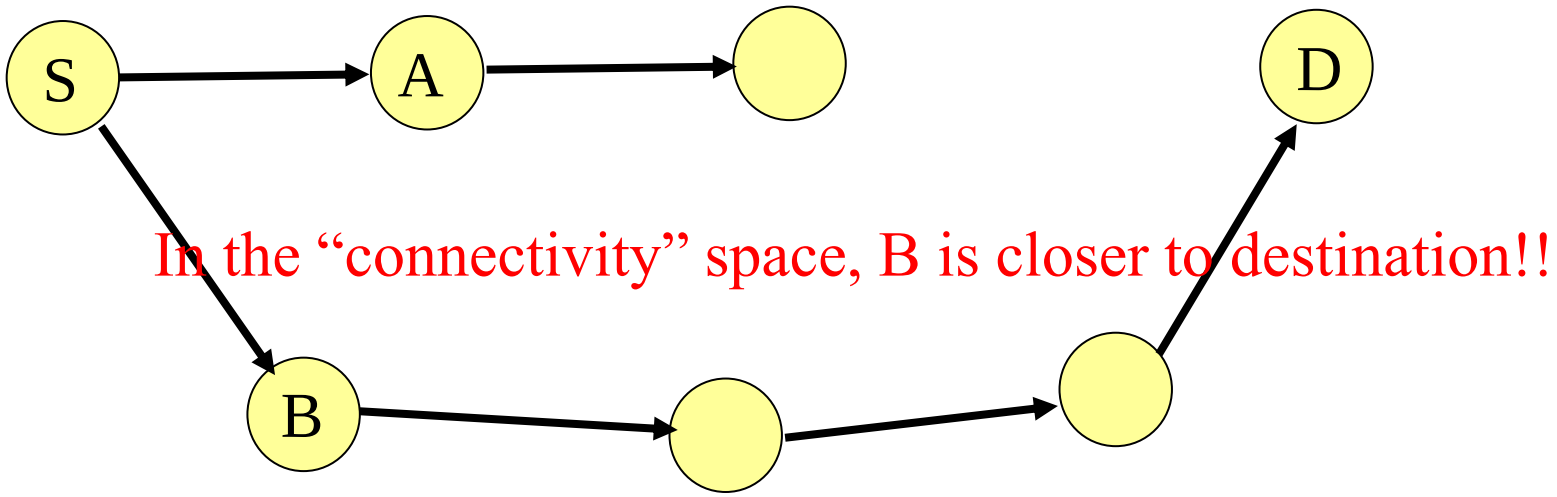
Geographic Routing Without Location Information

AP, Sylvia, Ion, Scott and Christos
UC Berkeley

ACM International Conference on Mobile Computing
and Networking (Mobicom'03), 2003

Why Geographic Routing Without Location?

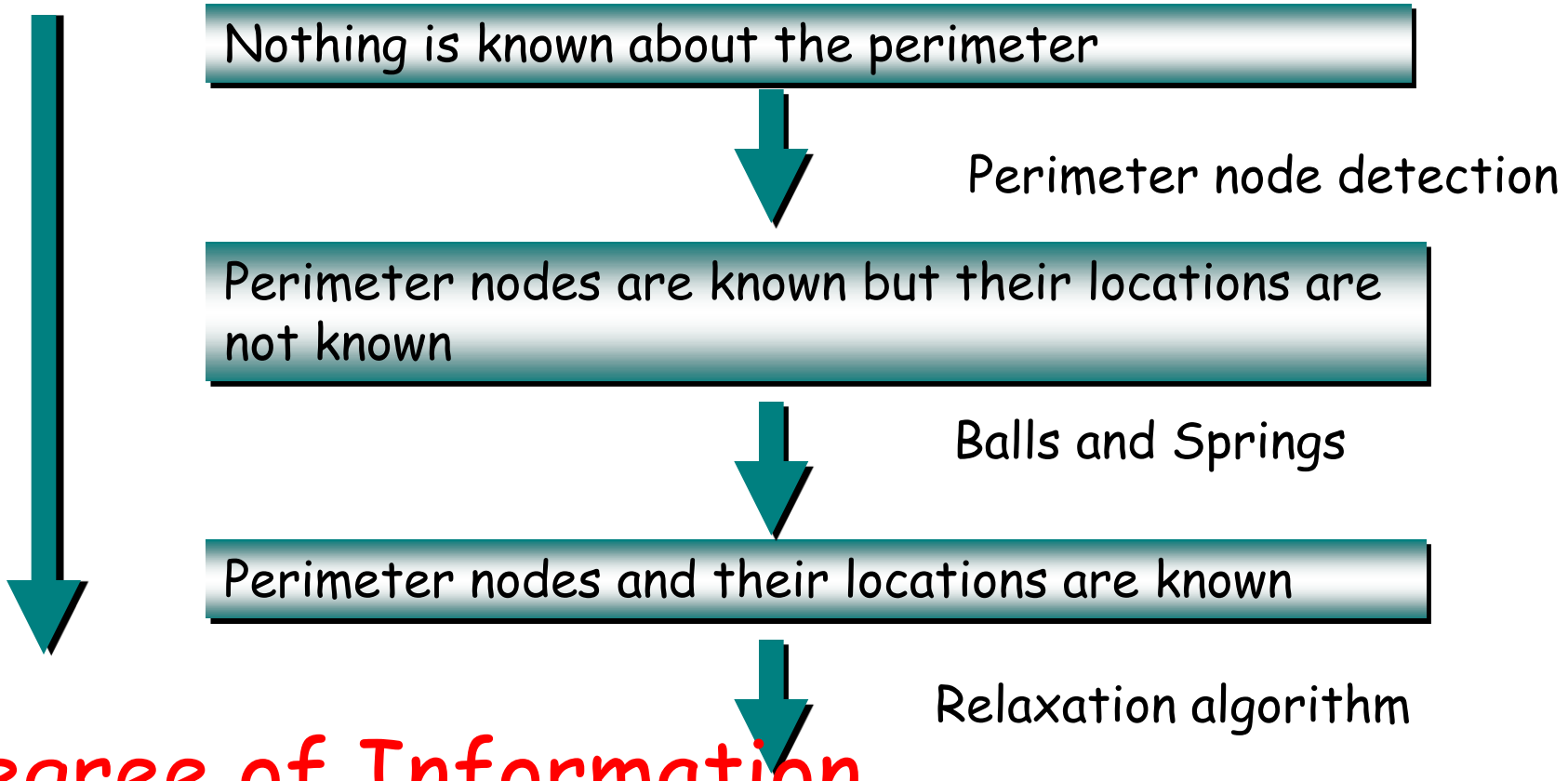
- Location is hard to get
  - GPS takes power, doesn't work indoors, difficult to incorporate in small sensors
  - The network localization problem is hard
- True location may not be useful if there are obstacles



Overview

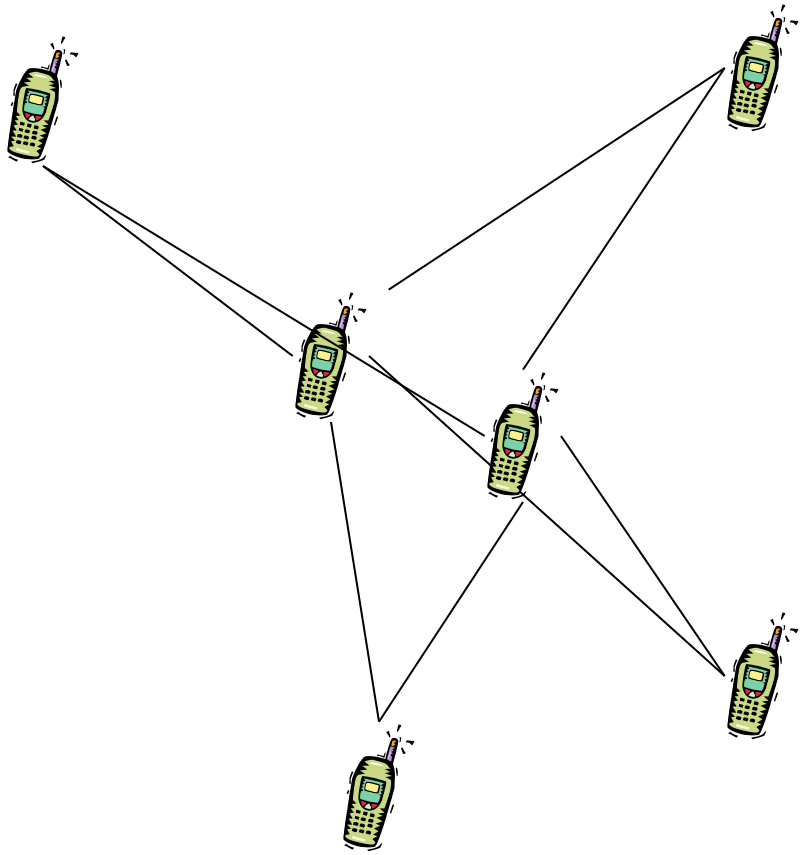
- Objective: assign coordinates to nodes so that
  - the coordinates are computed efficiently,
  - routing works well using the computed coordinates
- Coordinates reflect true connectivity and not the geographic locations of the nodes
  - Need not be accurate representations of the underlying geography
  - Reflect the underlying connectivity
- Progress in three steps
  - The perimeter nodes and their locations are known
  - The perimeter nodes are known but not their coordinates are not known
  - Nothing is known

Step by Step Approach



The Perimeter Nodes and Their Locations Are Known

- Image a rubber band from each node to each (connected) neighbor
- The force of a rubber band is proportional to its length, directed to the neighbor



The Perimeter Nodes and Their Locations Are Known

- Iterative process for picking coordinates for a node
- Some nodes along the periphery of the network know their correct (relative) locations and are fixed
- Other nodes compute coordinates by relaxation
  - Assume that nodes are connected by rubber bands and slowly converge to the equilibrium

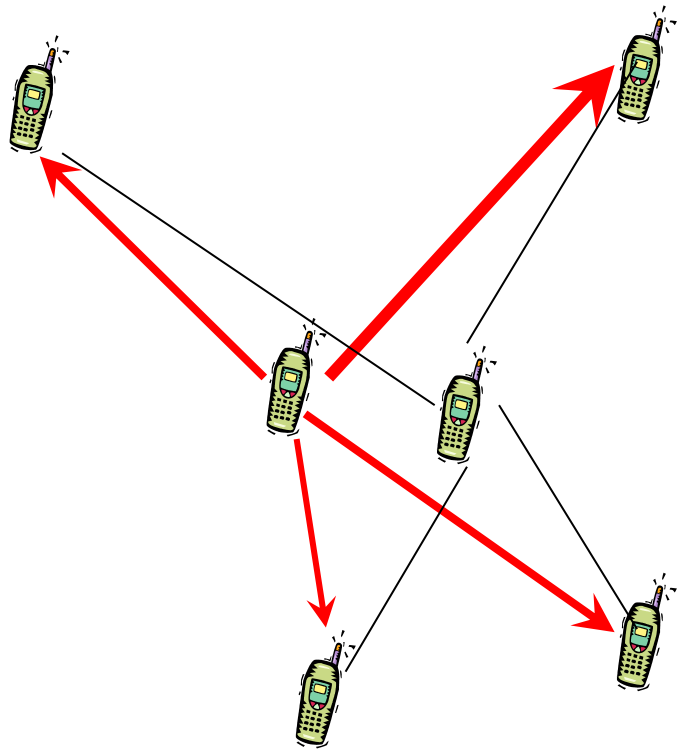
The Perimeter Nodes and Their Locations Are Known

- The equilibrium is achieved when the position p of a node is equal to the average of its neighbors

$$\sum (p_i - p) = 0 \Rightarrow p = \frac{\sum p_i}{n}$$

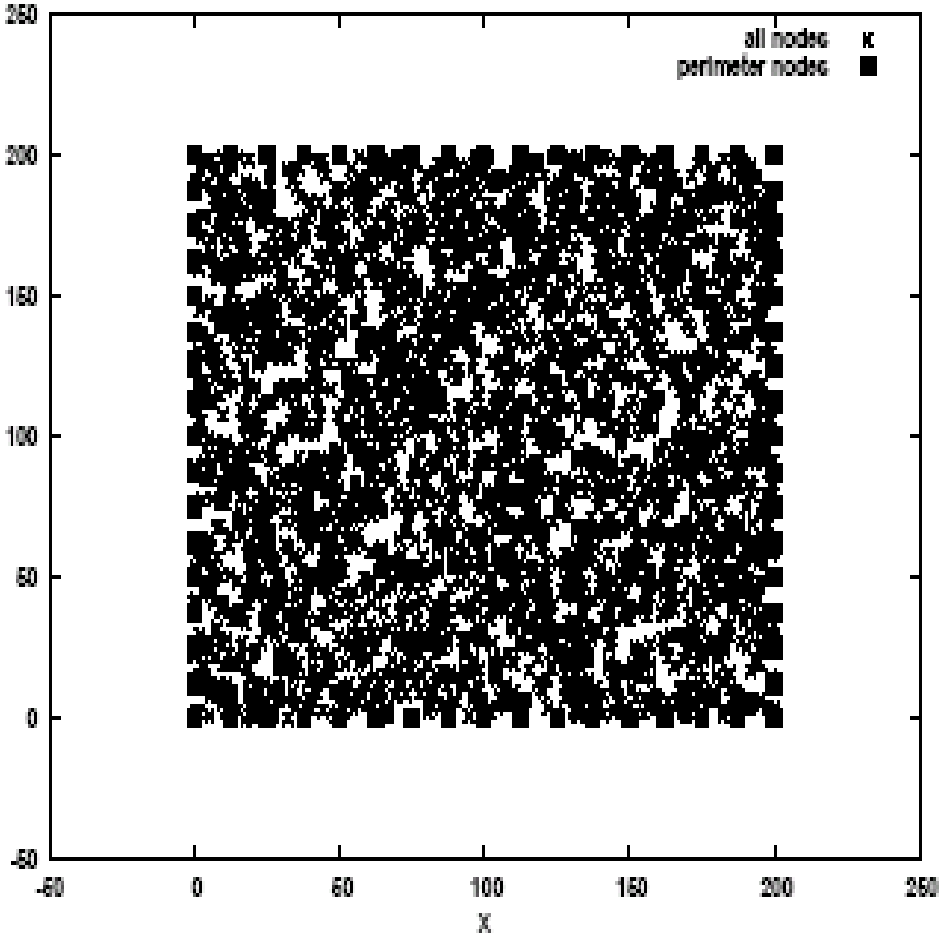
where n is number of neighbors

- Algorithm
  - each node sends its position to its neighbors
  - A node updates its new position to be the average of those of its neighbors



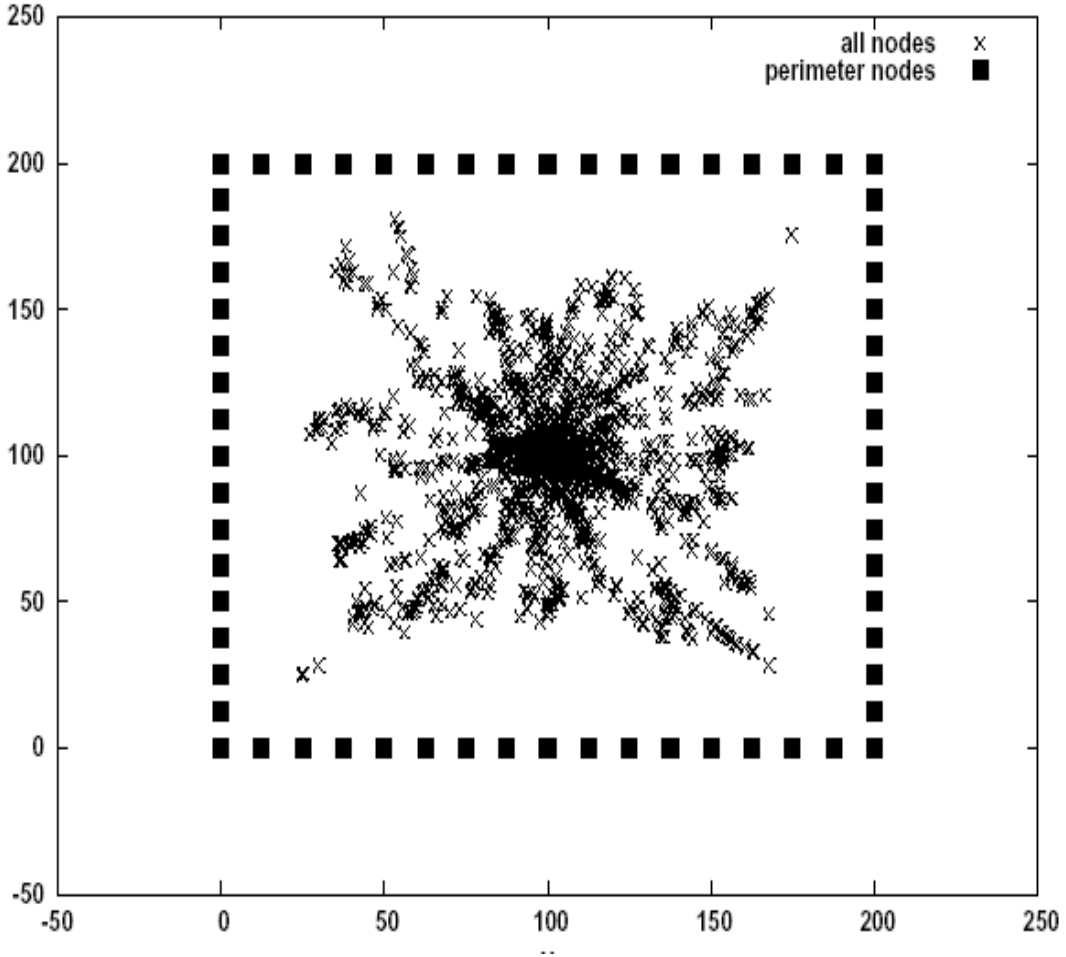
$$p(t+1) = \frac{\sum p_i(t)}{n}$$

Perimeter Nodes Are Known (True Positions)



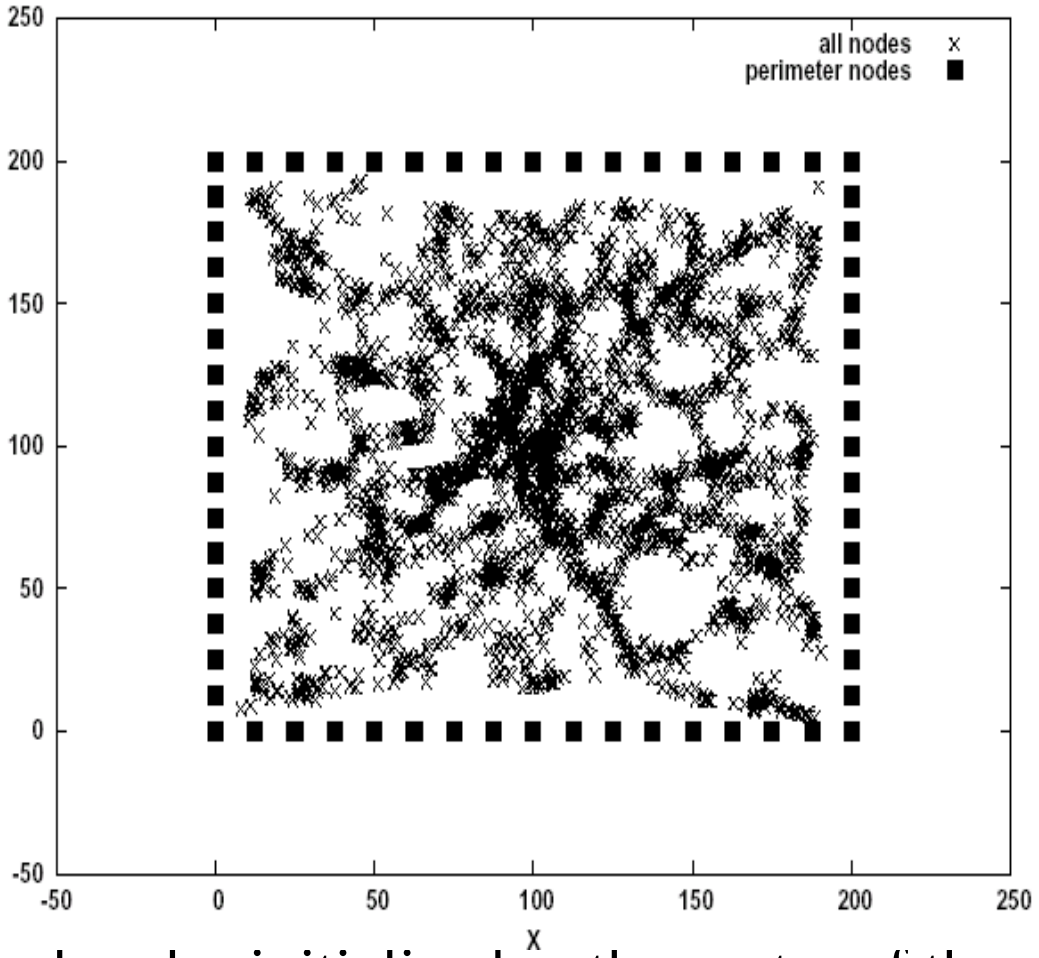
3200 nodes; 64 perimeter nodes on the boundary

Perimeter Nodes Are Known (10 iterations)



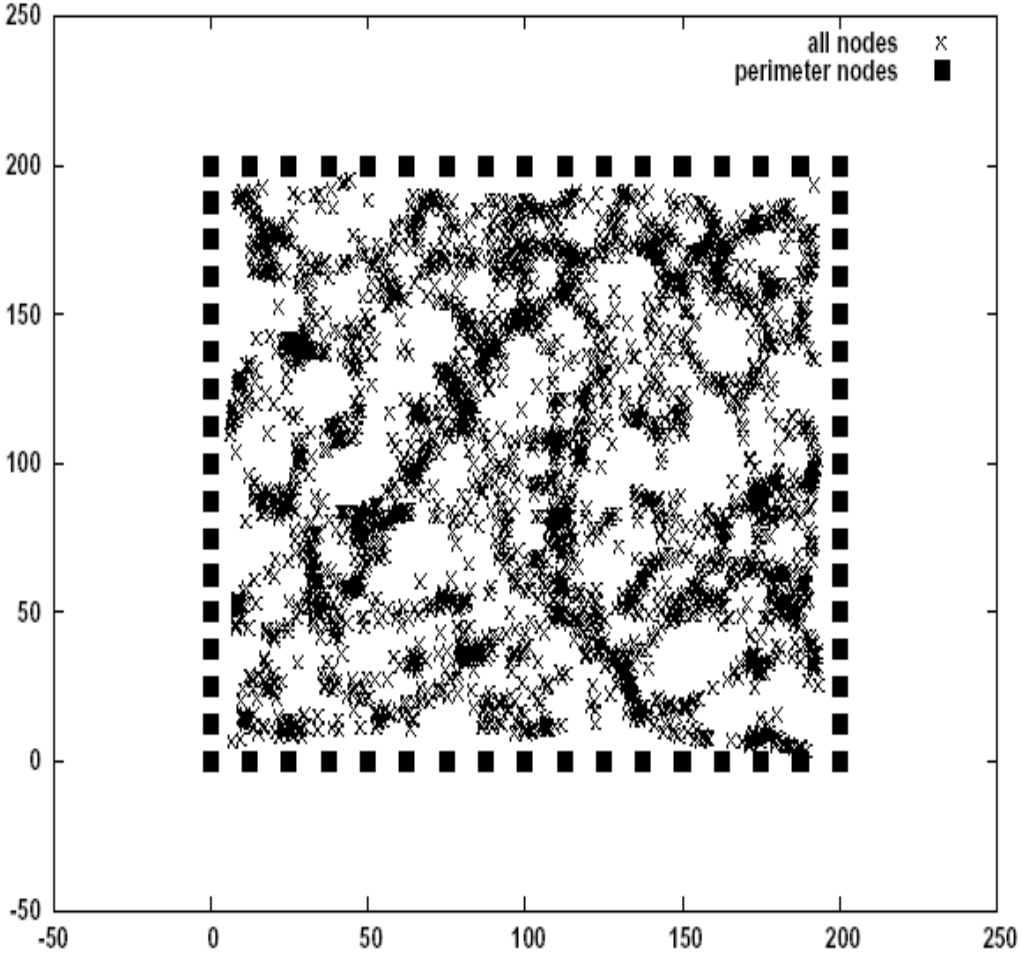
Internal nodes initialized as the center of the square

Perimeter Nodes Are Known (100 iterations)



Internal nodes initialized as the center of the square

Perimeter Nodes Are Known (1000 iterations)



Internal nodes initialized as the center of the square

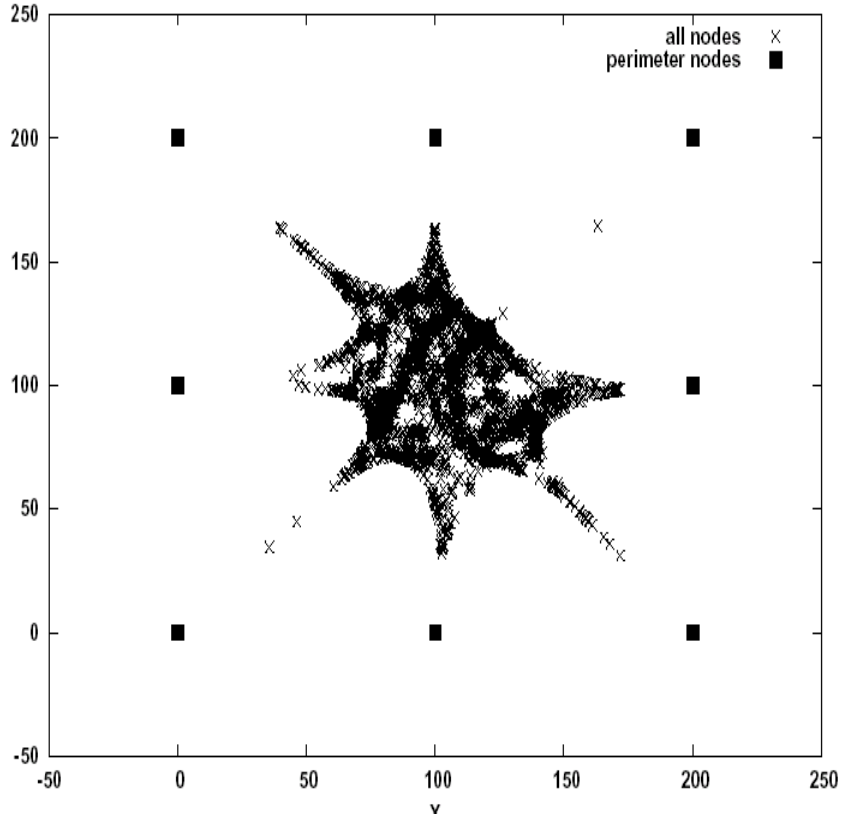
Routing Performance

- 32000 packets with random source-destination pairs

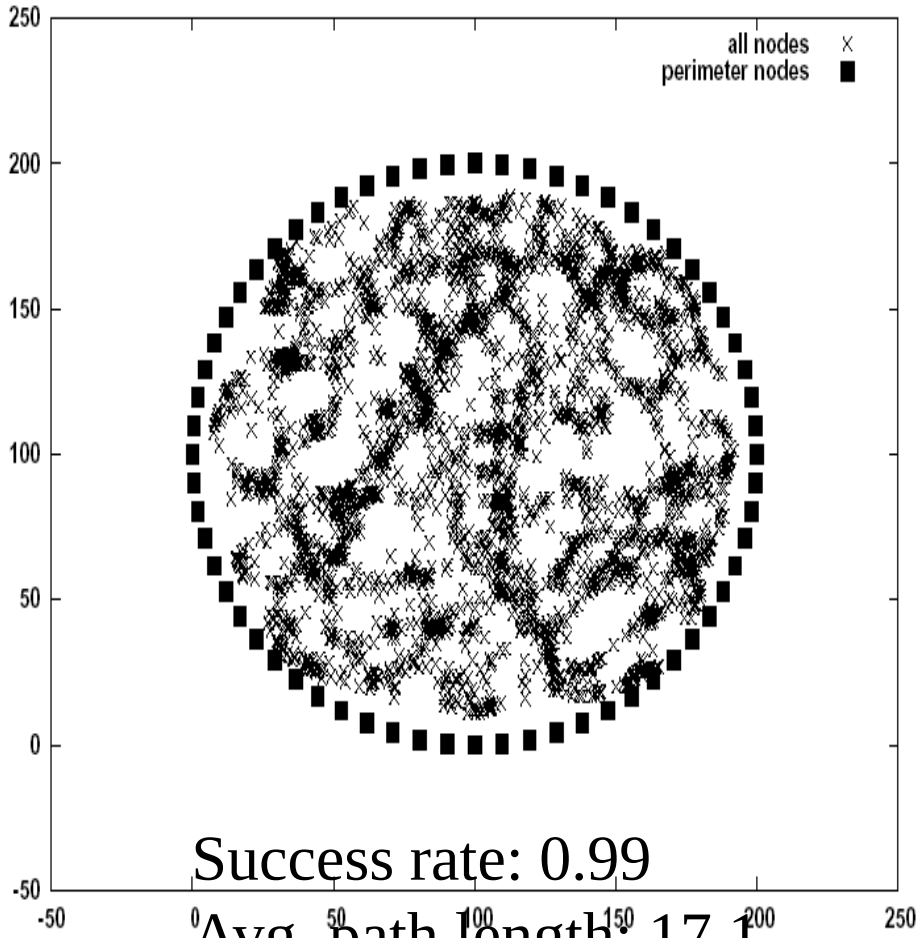
	Success rate	Average path length
True position	0.989	16.8
Virtual position	0.993	17.1

Success rate: using (distance) greedy routing

Two More Scenarios

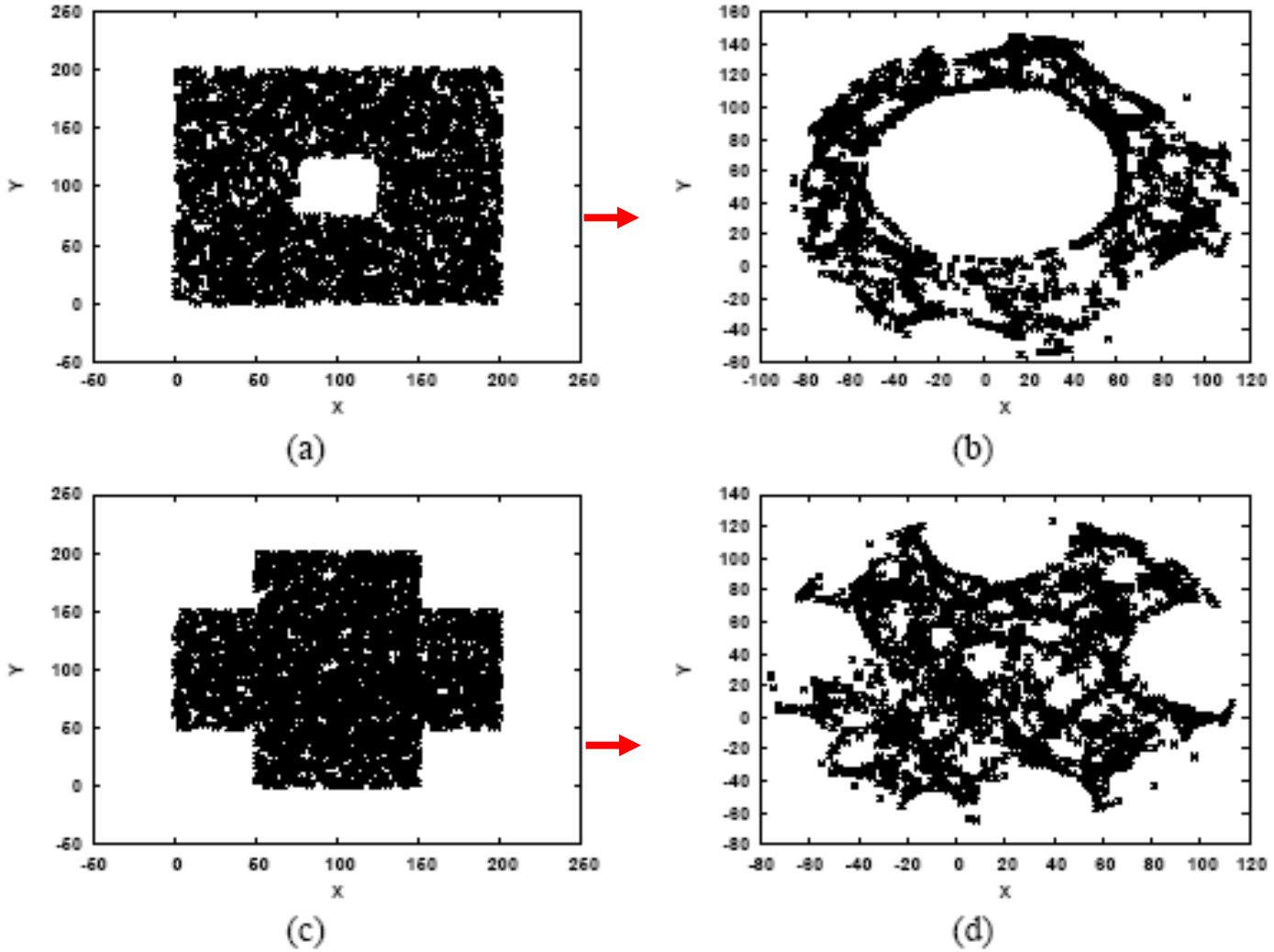


Success rate: 0.981
Avg. path length: 17.3



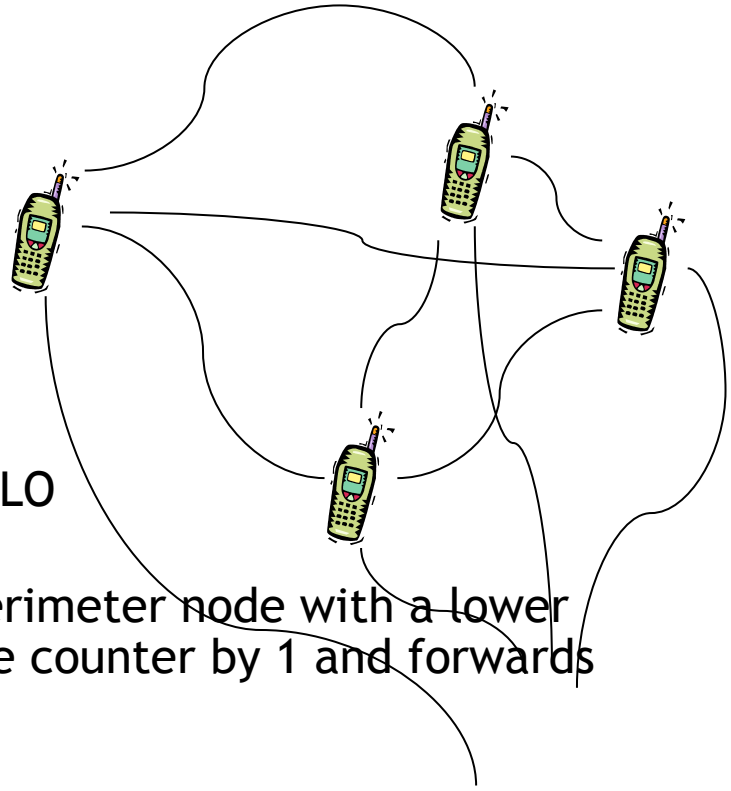
Success rate: 0.99
Avg. path length: 17.1

Weird Shapes



The Perimeter Nodes Are Known But Their Locations Are Not Known

- Assume the distance between two perimeter nodes is the (minimum) number of hops to go from one to the other



- Distances can be derived by flooding the network
  - Each perimeter node sends a HELLO message with a hop counter of 0
  - when seeing a message from a perimeter node with a lower hop counter, a node increases the counter by 1 and forwards it

- Stage 1 : Each perimeter node broadcasts a HELLO message to the entire network
- Stage 2 : Each perimeter node broadcasts its perimeter vector to the entire network
- Stage 3 : Every perimeter node uses a triangulation algorithm to compute the coordinates of all other perimeter nodes

- Balls and Springs
  - Ball : each perimeter node
  - Spring : each ball is attached by a spring
  - Spring's length : the hop count distance

$$\sum_{i,j \in perimeter\_set} (measured\_dist(i,j) - dist(i,j))^2$$

- Seen as minimizing the potential energy when a ball attached to every other ball by a spring

Virtual Coordinates by Triangulation

- Each perimeter node solves the minimization problem:

$$\min \sum_{i,j \text{ perimeter nodes}} (\mathrm{d}_{i,j} - \| p_i - p_j \|)^2$$

- Detail

$\text{Compute } D^2, \text{ where } D = [d_{ij}]$

$J = I - e^T e / n; e = [1,1,...1]$

$H = -\frac{1}{2} JD^2 J$

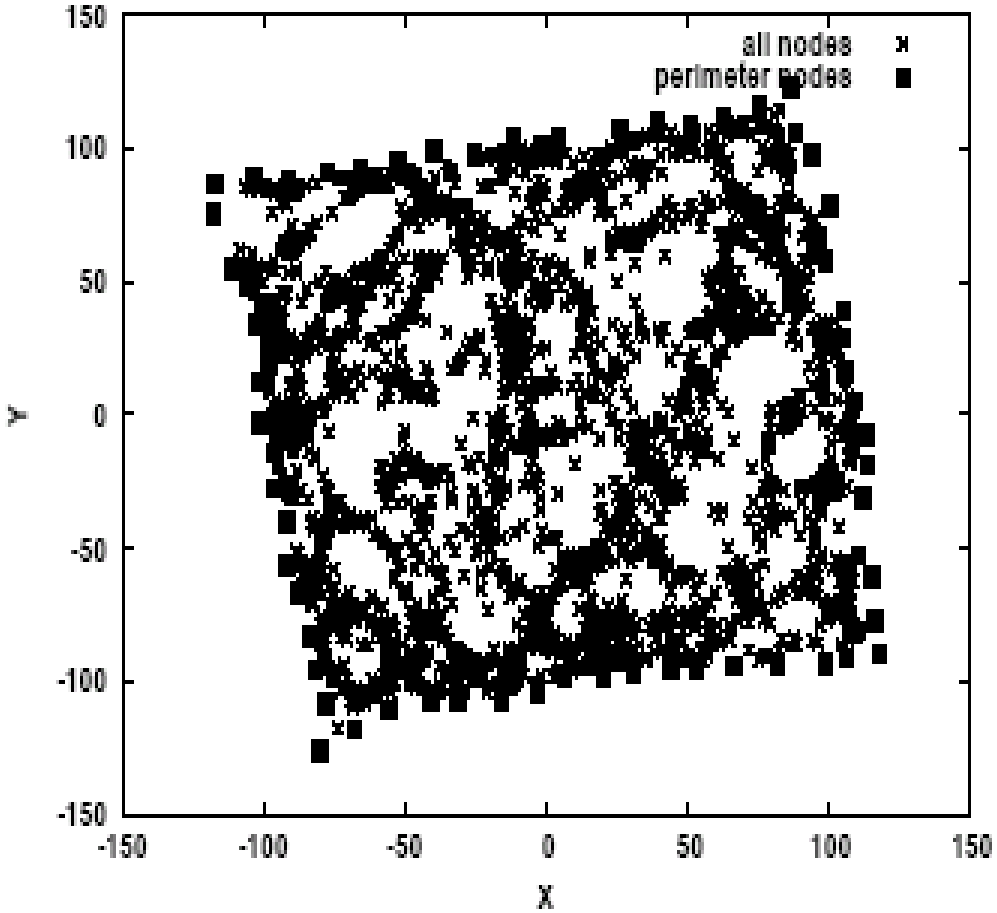
$H = UVU^T$

$V_2 : \text{the largest 2 eigen values}$

$U_2 : \text{the first 2 columns of U}$

$\mathrm{X} = \mathrm{U}_2 V_2^{\frac{1}{2}}$

Convergence and Performance



One iteration: success rate = 0.992; avg. path length = 17.2

Ten iterations: success rate = 0.994; avg. path length = 17.2

Nothing is Known about the Perimeter

- Bootstrap nodes
  - Special perimeter nodes or run leader election to select the two nodes
- Bootstrap Nodes flood the network and every node discovers its distance to these bootstrap nodes
- Nodes use the following criterion to decide whether they are perimeter nodes

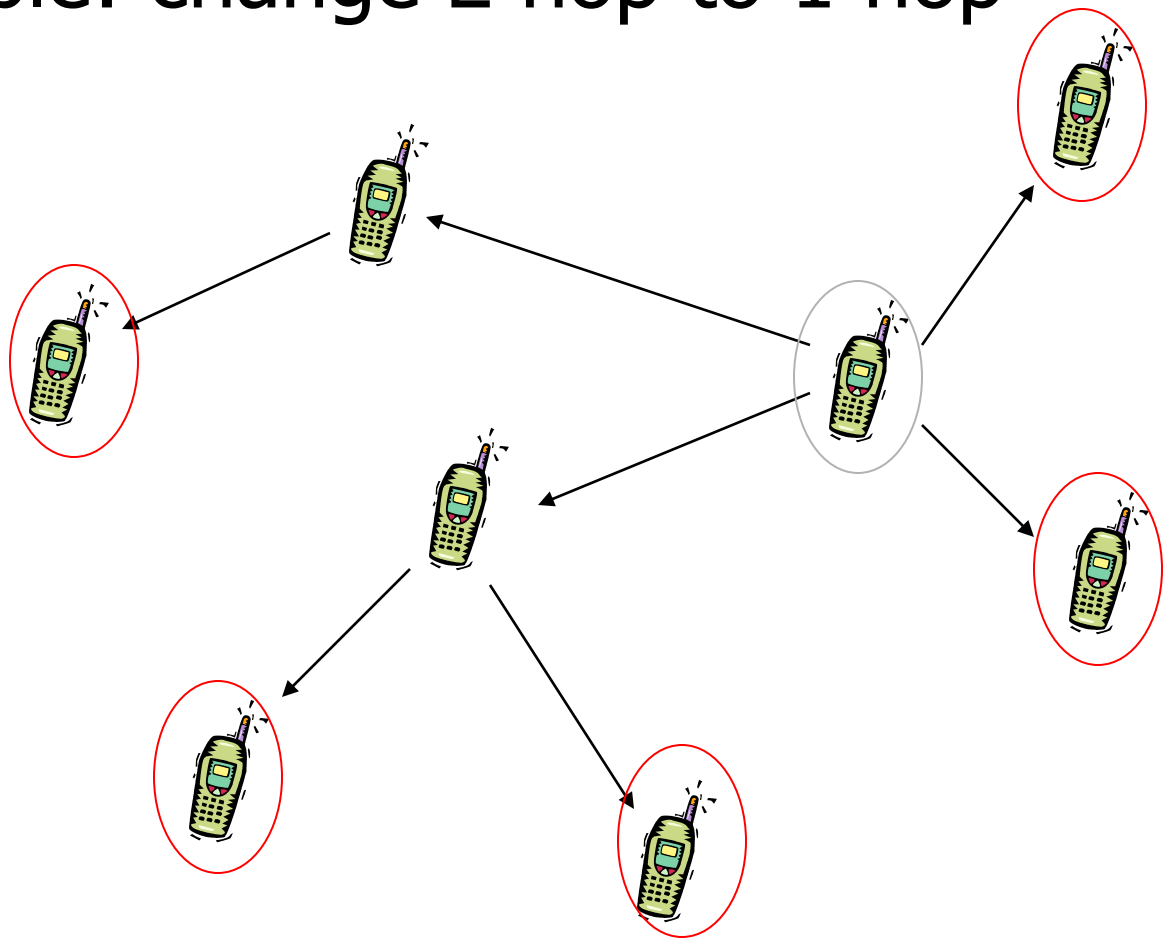
Selecting Perimeter Nodes

Rule: A node is a perimeter node if:

It is farthest away from the first bootstrap node among all its two-hop neighbors

Perimeter Node Detection

Example: change 2 hop to 1 hop



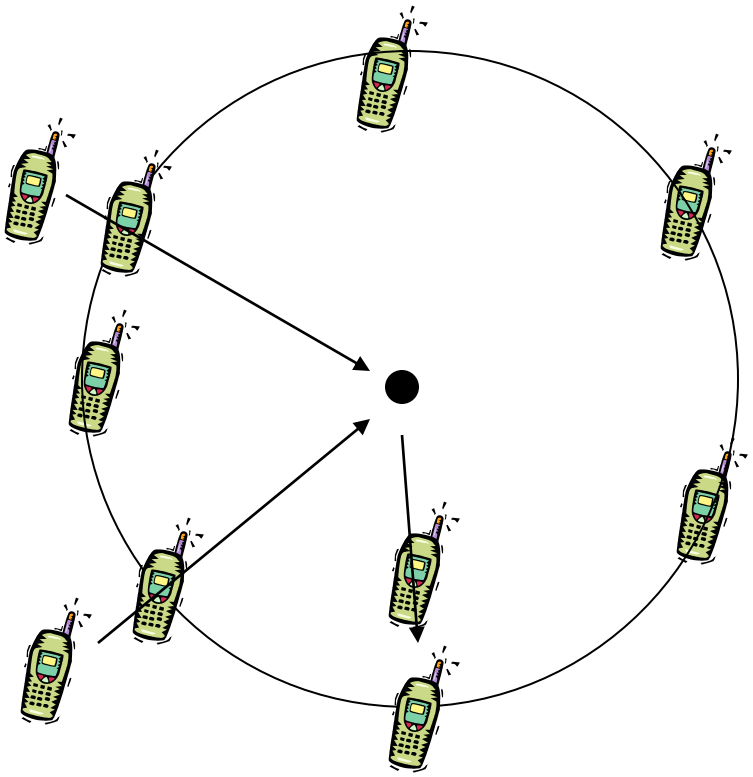
Overall algorithm - Bootstrap the coordinate assignment

1. Two designated bootstrap beacon nodes broadcast to the entire network
2. Node uses the distance to determine whether is a perimeter node
3. Every perimeter node sends a broadcast message to the entire network to enable every other node to compute its perimeter vector
4. Perimeter and bootstrap nodes broadcast their perimeter vectors to the entire network
5. Each node uses these inter-perimeter distances to compute normalized coordinates for both itself and the perimeter nodes

Overall Algorithms[Cont'd] - normal operation

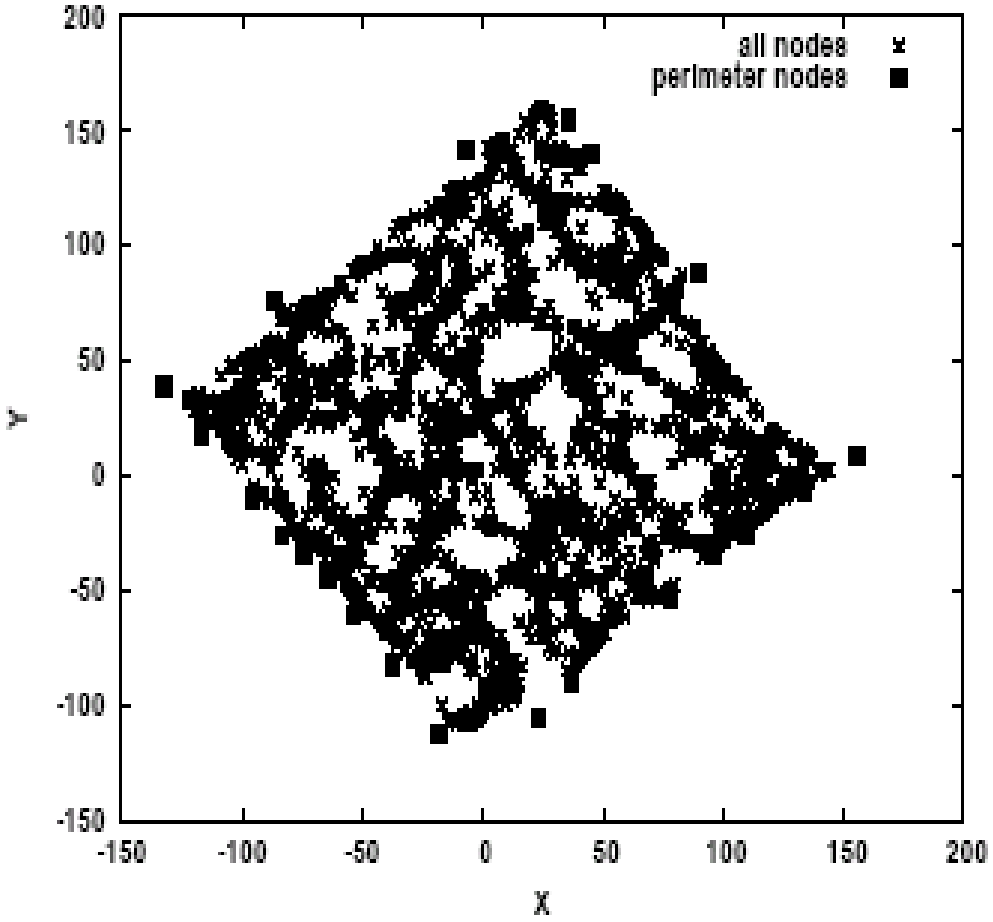
5. Perimeter nodes stay fixed while other nodes run a relaxation algorithm
6. A designated bootstrap node periodically broadcasts by which nodes periodically re-asses whether they lie on the perimeter or not

Projecting on Circle After First Computation



- Circle: center is center of gravity; radius is average of distance of the perimeter nodes to the CG
- Motivation: maintain a consistent coordinate space

Convergence and Performance



Ten iterations: success rate = 0.996; avg. path length = 17.3

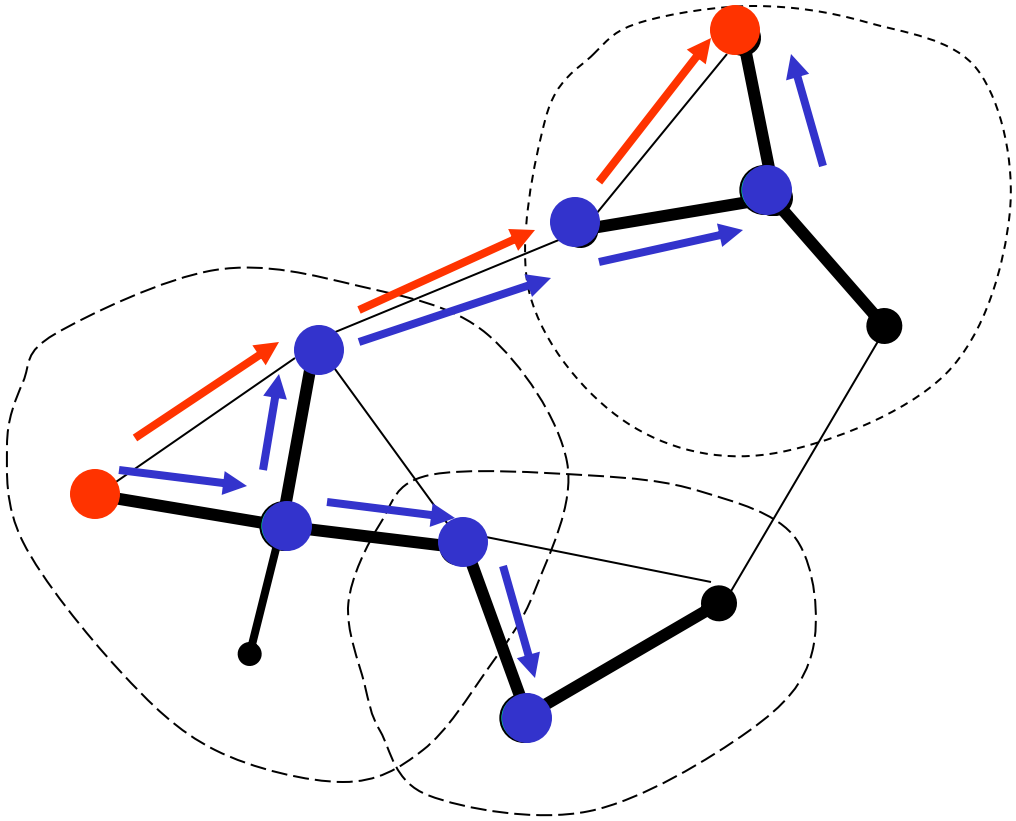
Cluster Based Routing Protocol

CBRP

Mingliang Jiang, Jinyang Li and Y.C. Tay.
National University of Singapore

“Cluster Based Routing Protocol (CBRP)”, Internet Draft draft-ietf-manet-cbrp-spec-01.txt,
August 1999.

- use clustering approach to minimize on-demand route discovery traffic
- use “local repair” to reduce route acquisition delay and new route discovery traffic
- suggest a solution to use uni-directional links



Some Terminologies

- A cluster head must have bi-directional links to all its member nodes.
- A node will be a member of all those clusters for which it has a bi-directional link to the cluster heads.
- These are called host clusters for the node.

Data Structures

- Neighbor Table
  - Id, Role , Status of the link
- Cluster Adjacency Table (CAT)
  - Keeps info. about adjacent clusters
  - Contains
    - Id of neighboring cluster
    - the gateway node (a member) to reach the neighboring cluster head
    - the status of the link

Data Structures (Contd.)

- Two-hop Topology Database
  - each node broadcasts its neighbor table information periodically in HELLO packets.
  - Therefore, by examining the neighbor table from its neighbors, a node is able to gather ‘complete’ information about the network topology that is at most two-hops away from itself.

HELLO Messages

- Every node periodically broadcasts HELLO messages to its neighbors.
- HELLO message from a node contains its neighbor table and its cluster adjacency table (CAT).
- Nodes update their neighbor tables and CAT when they receive HELLO messages from their neighbors.

HELLO Messages (Contd.)

- When a node A receives HELLO message from a node B
  - A adds B to its neighbor table if B is not present in its table.
  - If B is already in the table update the status of link from B to A if required.
  - Update the role of B if it has changed.

Cluster Formation

- A node can be in any of the three states
  - A cluster head
  - A cluster member
  - Undecided (Looking for a head)
- An undecided node starts a timer and broadcasts a HELLO message.
- Any cluster head that receives this message sends out HELLO message back.

Cluster Formation

- If the node has bi-directional link to that cluster head it chooses that node as its cluster head and regards itself as a member of that cluster head.
- If it does not find any head till the timer expires and it declares itself as a cluster head.

Cluster Formation

- If two cluster heads have bi-directional links to each other one of them gives his status as a head and becomes member of the other head. The node with a smaller id continues to be a cluster head.
- However the cluster heads wait for a certain period of time before this
- This ensures that if two cluster heads are just close for a short time when they are on a move, cluster re-formation does not happen.

Adjacent Cluster Discovery

- For a member node neighboring cluster head is the one that is two hops away. i.e. one that can be reached via an intermediate node. This node is called a Gateway node.
- A node can find out about its neighboring cluster heads by looking at the neighbor tables of its neighbors received in the HELLO messages.

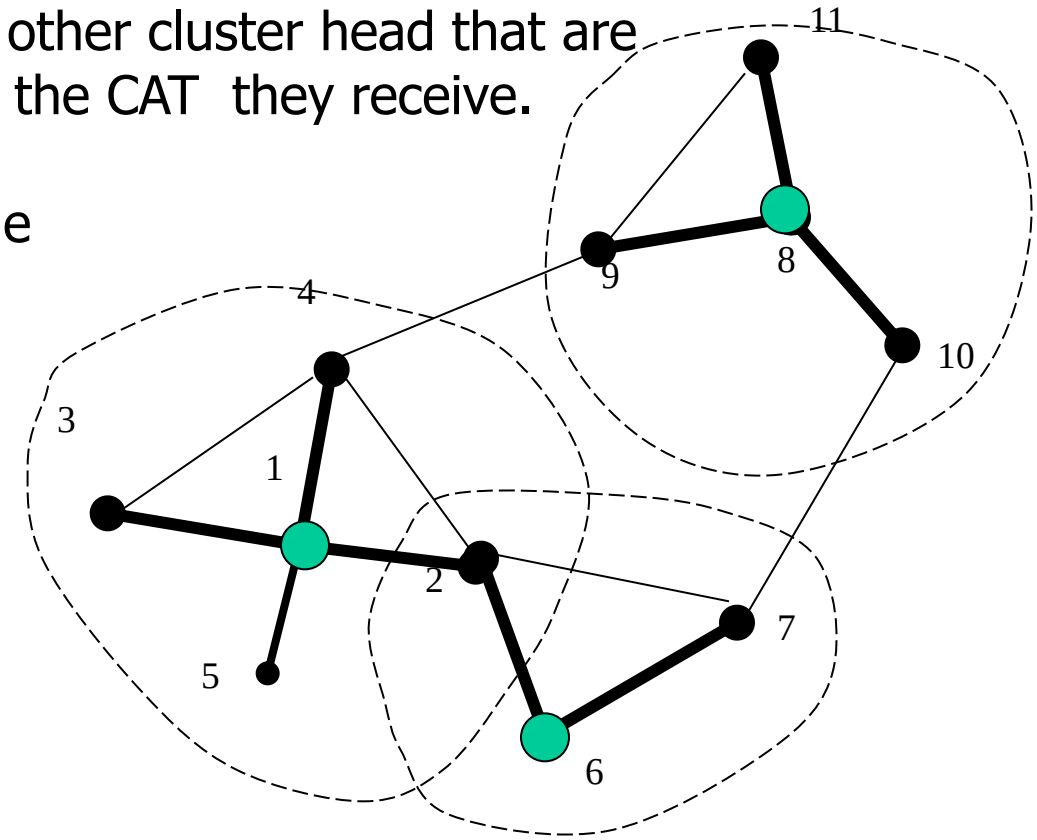
Adjacent Cluster Discovery

Nodes also broadcasts their CAT in the HELLO message.

Cluster heads can learn about other cluster head that are three hops away by looking at the CAT they receive.

e.g. 4's Cluster Adjacency Table

Adj cluster ID	Gateway
8	9
6	2



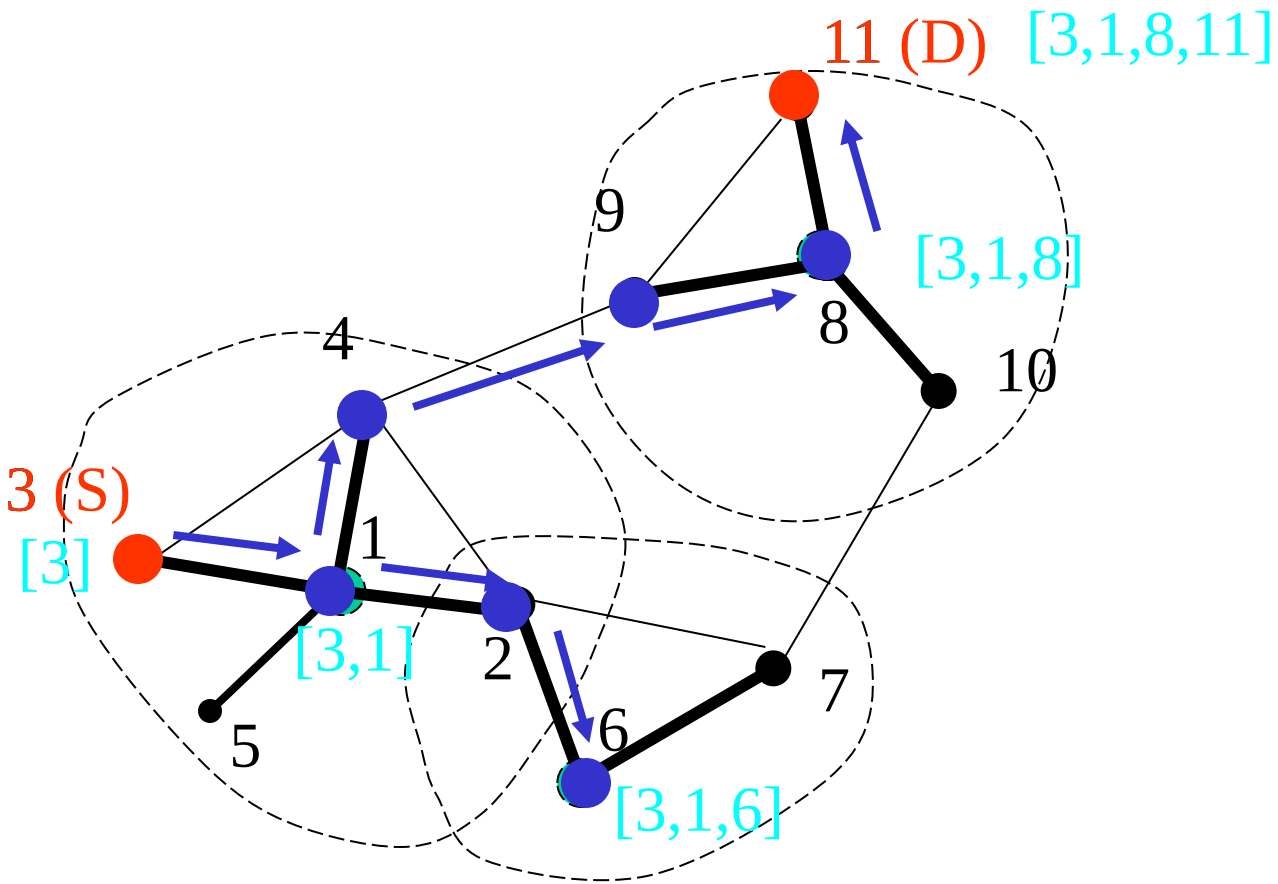
Route Discovery

- When a node say A wants to discover route to a node say D it broadcasts a RREQ packet.
- This packet contains a list of host and neighboring clusters heads. For neighboring cluster heads even the gateway nodes are mentioned.
- The idea is only cluster heads should forward the packet further.
- If a member node receives RREQ packet it simply drops it.
- However if a member node is listed as a Gateway node it unicasts the RREQ to the cluster head for which it is a Gateway node.

Route Discovery

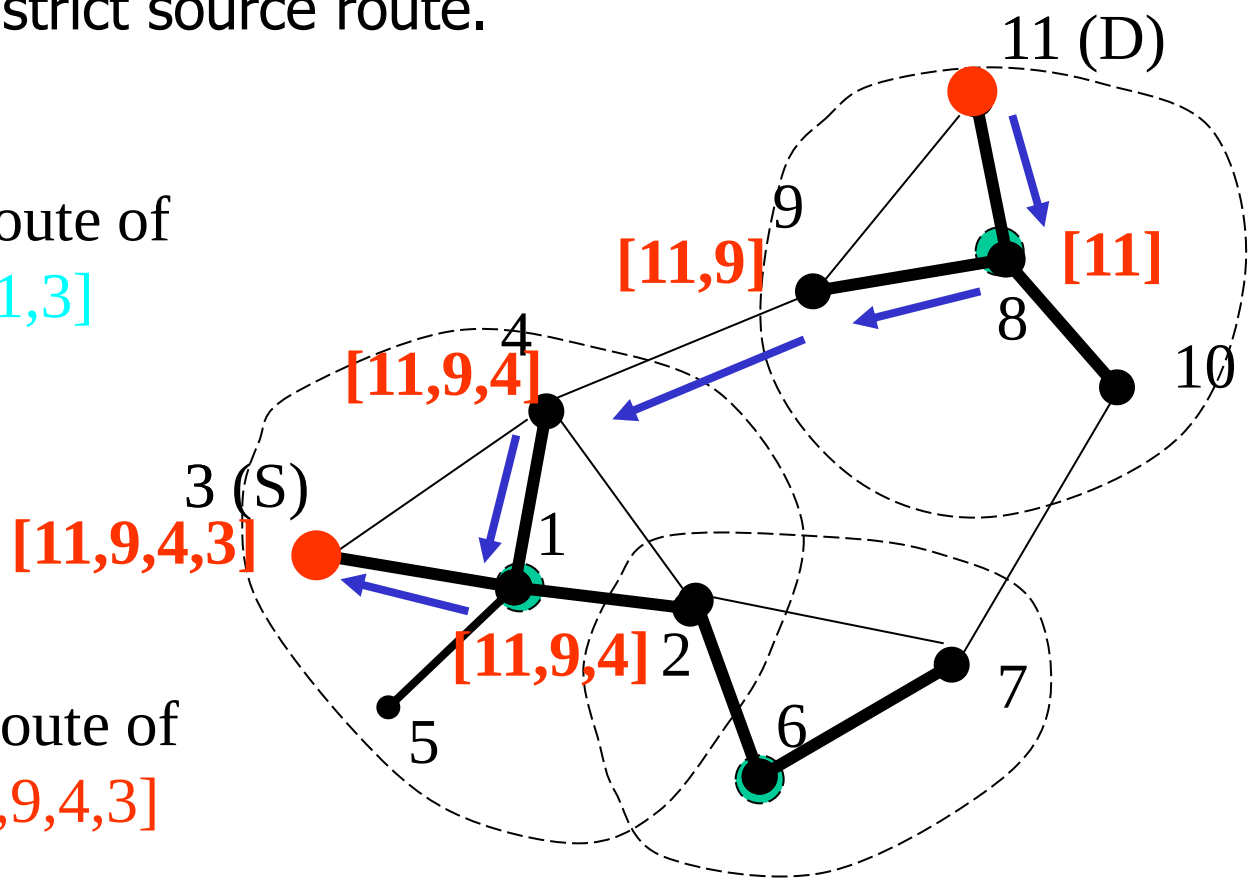
- When a cluster head receives RREQ, it adds itself on the partial route contained in the packet.
- It adds the neighboring cluster heads to which the packet is to be forwarded from its own CAT along with their gateway nodes and then re-broadcasts their packet.
- Thus the RREQ passes through a number of cluster heads and eventually reaches D.
- D upon receiving the RREQ sends and RREP back.
- The RREP travels the same set of cluster heads that the RREQ traveled.
- On the way entire hop-by-hop path is added to the RREP along with the Gateway nodes.

Source S “floods” all clusterheads with Route Request Packets (RREQ) to discover destination D



- Route reply packet (RREP) is sent back to source along reversed "loose source route" of clusterheads.
- Each clusterhead along the way incrementally compute a hop-by-hop strict source route.

the reversed loose source route of RREP: [11,8,1,3]

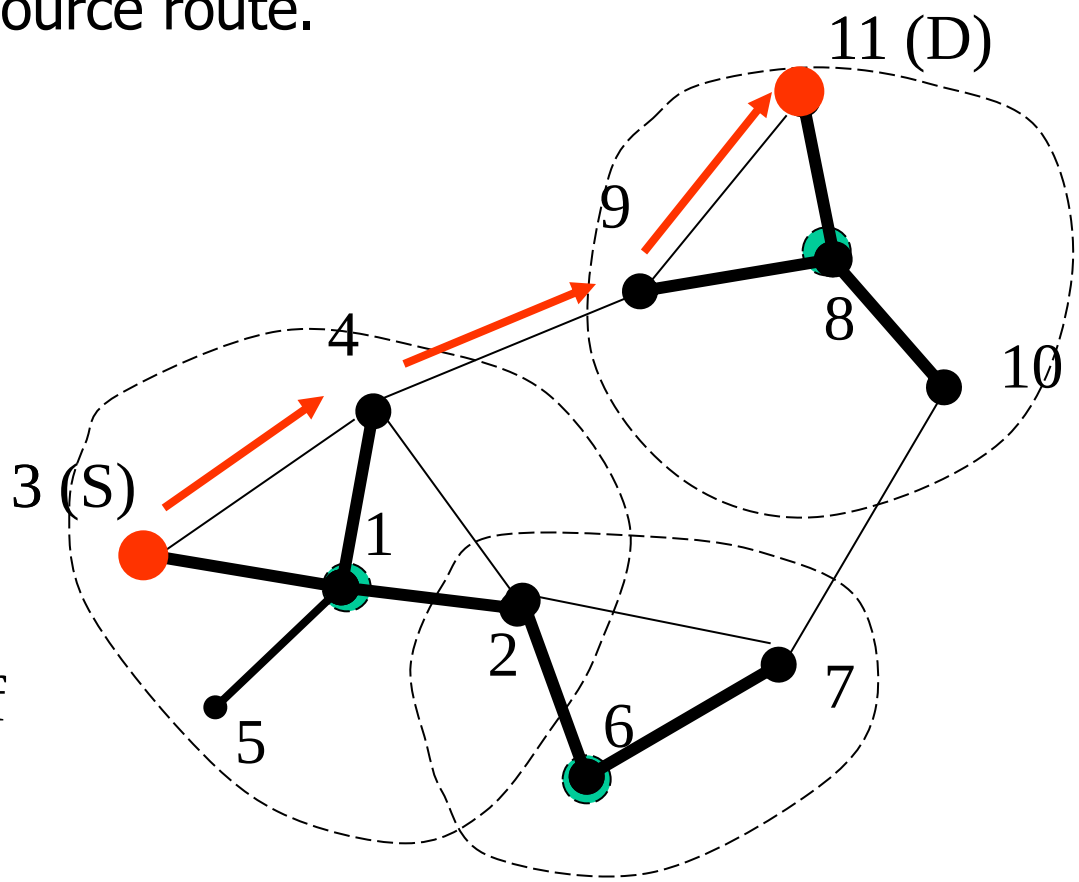


the computed strict source route of 3->11 is: [11,9,4,3]

- Route reply packet (RREP) is sent back to source along reversed "loose source route" of clusterheads.
- Each clusterhead along the way incrementally compute a hop-by-hop strict source route.

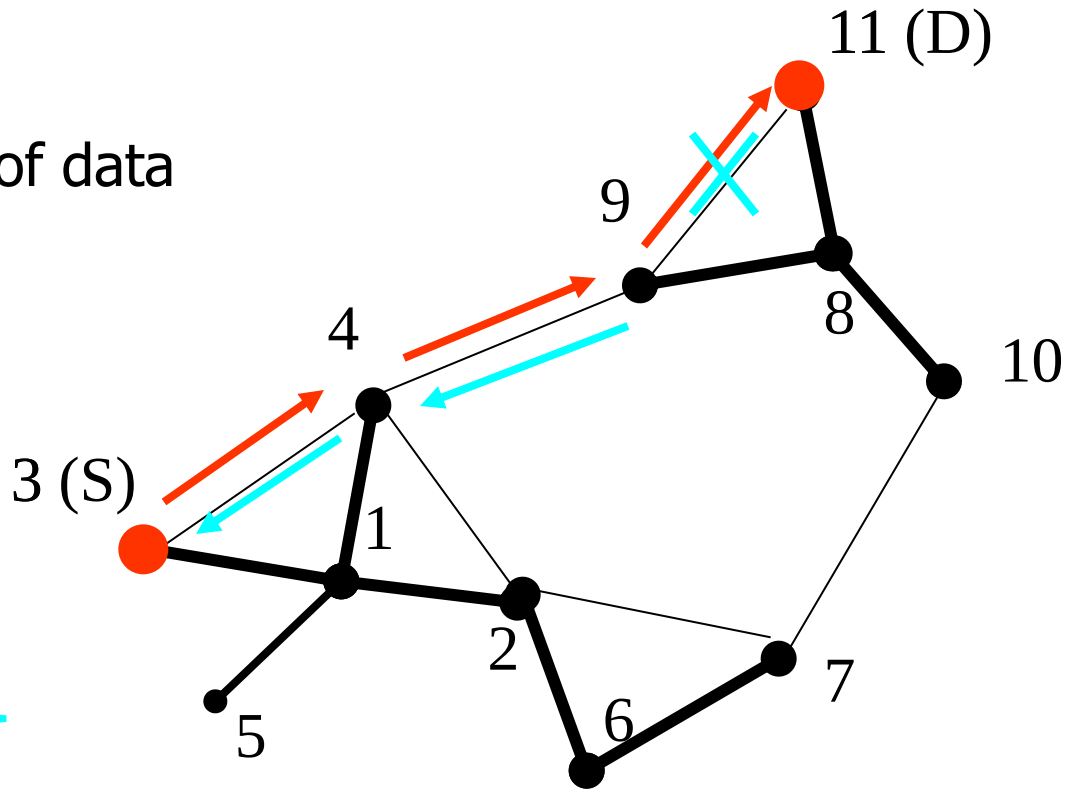
the reversed loose source route of RREP: [11,8,1,3]

the computed strict source route of 3->11 is: [11,9,4,3]



- Use source routing for actual packet forwarding
- A forwarding node sends a Route Error Message (ERR) to packet source if the next hop in source route is unreachable

Source route header of data packet: **[3,4,9,11]**



Route error (ERR) down link: **{9->11}**

Problems with CBRP

- Pitfalls with uni-directional links
  - Discovery of (dead) uni-directional links
- Source Routing, overhead bytes per packet.
- Clusters small, 2 levels of hierarchy, scalable to an extend.

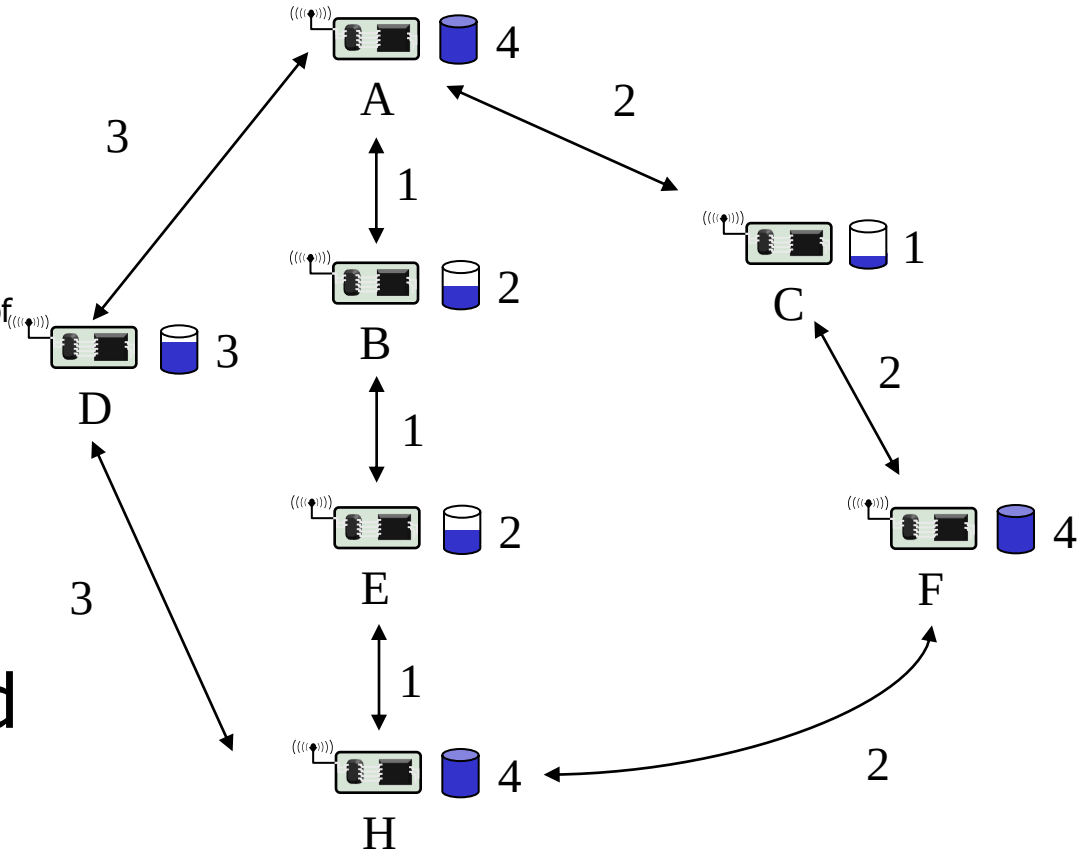
Other Routing

- Multipath Routing
- Energy-Conserving Routing
- Security-Aware Routing

Energy-efficient unicast: Goals

- Particularly interesting performance metric: Energy efficiency

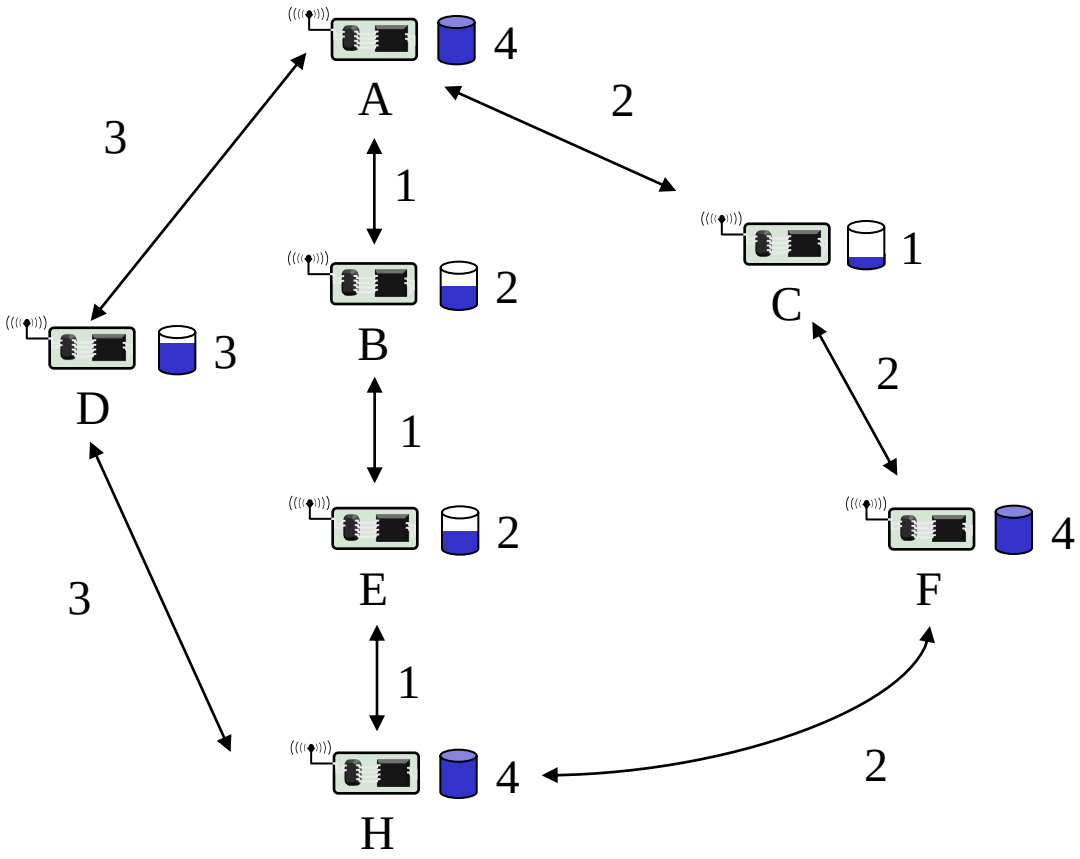
- Goals
  - Minimize energy/bit
    - Example: A-B-E-H
  - Maximize network lifetime
    - Time until first node failure, loss of coverage, partitioning
- Seems trivial – use proper link/path metrics (not hop count) and standard routing



Example: Send data from node A to node H

Basic options for path metrics

- Maximum total available battery capacity
  - Path metric: Sum of battery levels
  - Example: A-C-F-H
- Minimum battery cost routing
  - Path metric: Sum of reciprocal battery levels
  - Example: A-D-H
- Conditional max-min battery capacity routing
  - Only take battery level into account when below a given level
- Minimize variance in power levels
- Minimum total transmission power

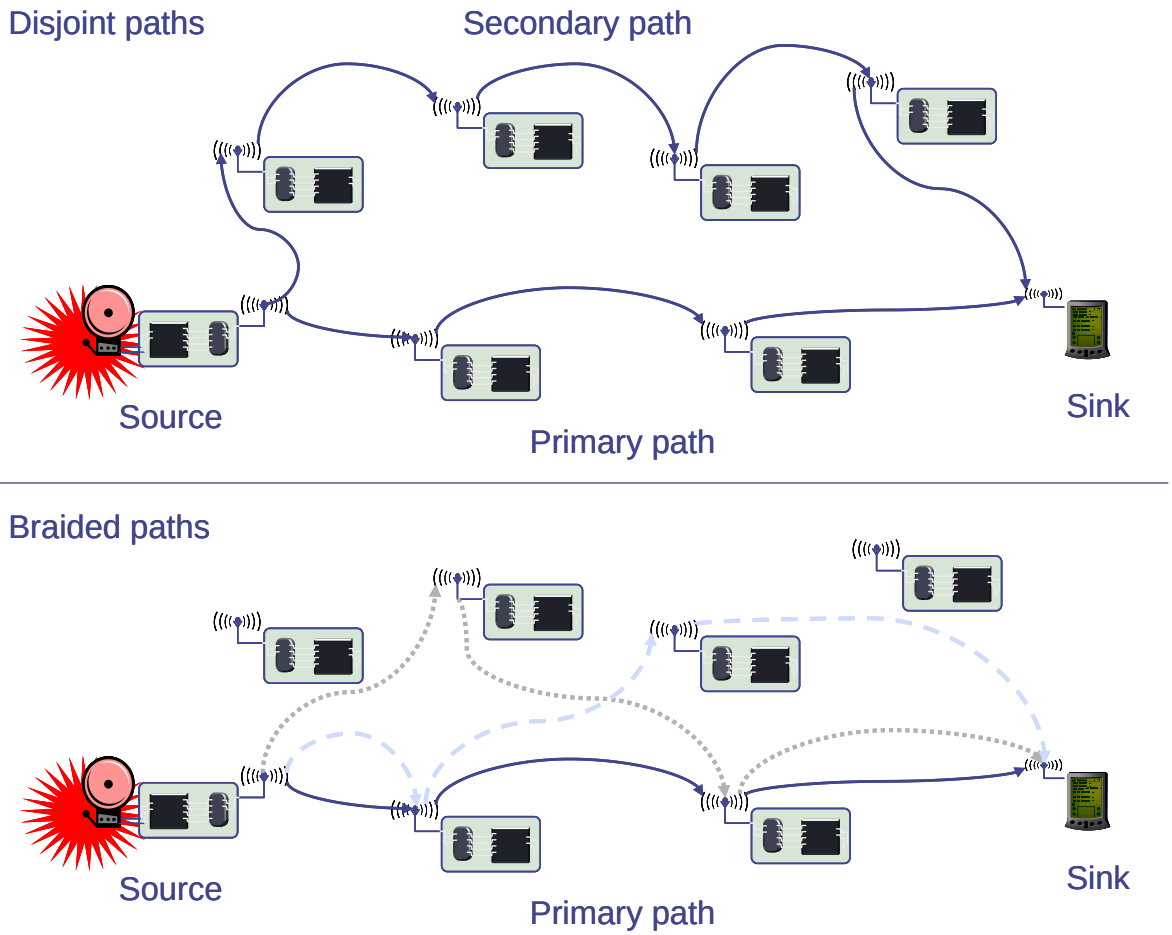


A non-trivial path metric

- Previous path metrics do not perform particularly well
- One non-trivial link weight: $w_{ij} = e_{ij}(\lambda^{\alpha_i} - 1)$
  - w_{ij} weight for link node i to node j
  - e_{ij} required energy, λ some constant, α_i fraction of battery of node i already used up
- Path metric: Sum of link weights
  - Use path with smallest metric
- Properties: Many messages can be send, high network lifetime
  - With admission control, even a competitive ratio logarithmic in network size can be shown

Multipath unicast routing

- Instead of only a single path, it can be useful to compute multiple paths between a given source/destination pair
  - Multiple paths can be ***disjoint*** or ***braided***
  - Used simultaneously, alternatively, randomly, ...



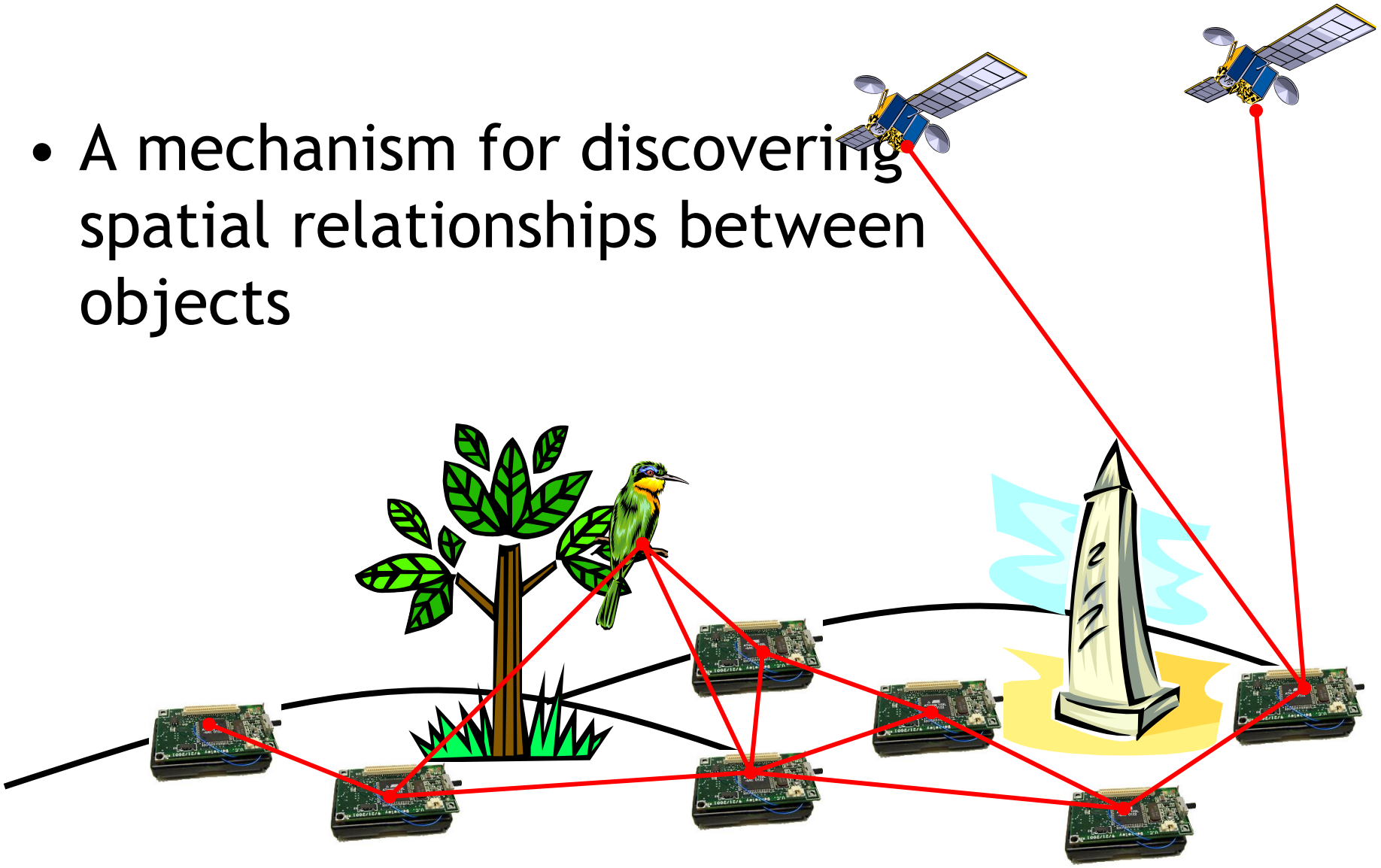
WSN LOCALIZATION

Goals of this chapter

- Means for a node to determine its physical position (with respect to some coordinate system) or symbolic location
- Using the help of
  - Anchor nodes that know their position
  - Directly adjacent
  - Over multiple hops
- Using different means to determine distances/angles locally

What is Localization

- A mechanism for discovering spatial relationships between objects

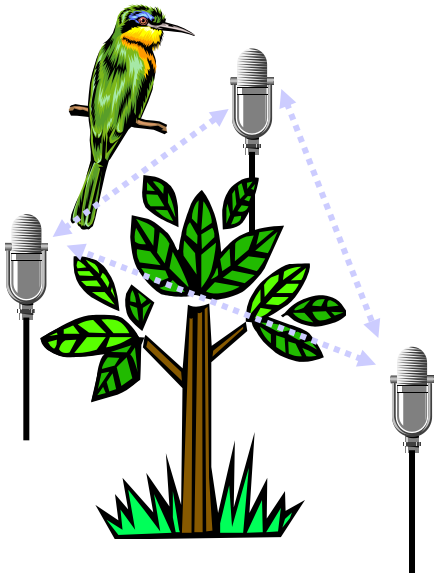


Why is Localization Important?

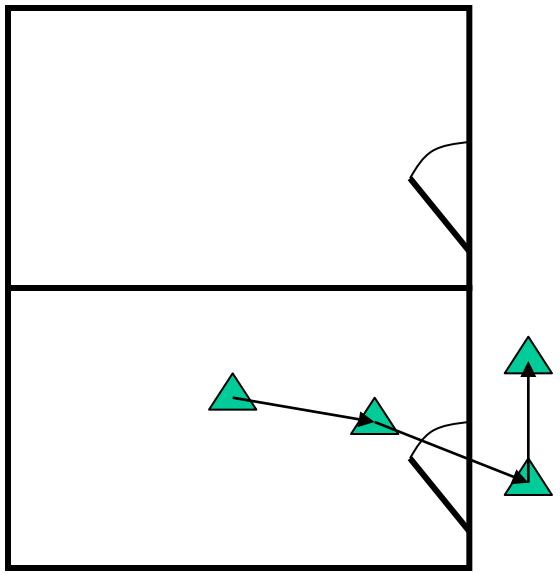
- Large scale embedded systems introduce many fascinating and difficult problems…
  - This makes them much more difficult to use…
  - BUT it couples them to the physical world
- Localization measures that coupling, giving raw sensor readings a physical context
  - Temperature readings ⇒ temperature map
  - Asset tagging ⇒ asset tracking
  - “Smart spaces” ⇒ context dependent behavior
  - Sensor time series ⇒ coherent beamforming

Variety of Applications

- Two applications:



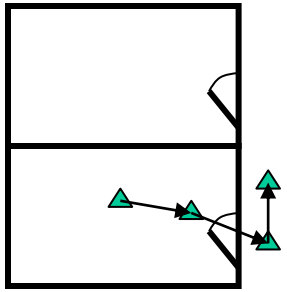
Passive habitat monitoring:
Where is the bird?
What kind of bird is it?



Asset tracking:
Where is the projector?
Why is it leaving the room?

Variety of Application Requirements

- Very different requirements!
  - Outdoor operation
    - Weather problems
  - Bird is not tagged
  - Birdcall is characteristic but not exactly known
  - Accurate enough to photograph bird
  - Infrastructure:
    - Several acoustic sensors, with known relative locations; coordination with imaging systems



  - Indoor operation
    - Multipath problems
  - Projector is tagged
  - Signals from projector tag can be engineered
  - Accurate enough to track through building
  - Infrastructure:
    - Room-granularity tag identification and localization; coordination with security infrastructure

Multidimensional Requirement Space

- Granularity & Scale
- Accuracy & Precision
- Relative vs. Absolute Positioning
- Dynamic vs. Static (Mobile vs. Fixed)
- Cost & Form Factor
- Infrastructure & Installation Cost
- Communications Requirements
- Environmental Sensitivity
- Cooperative or Passive Target

Axes of Application Requirements

- Granularity and scale of measurements:
  - What is the smallest and largest measurable distance?
  - e.g. cm/50m (acoustics) vs. m/25000km (GPS)
- Accuracy and precision:
  - How close is the answer to “ground truth” (accuracy)?
  - How consistent are the answers (precision)?
- Relation to established coordinate system:
  - GPS? Campus map? Building map?
- Dynamics:
  - Refresh rate? Motion estimation?

Axes of Application Requirements

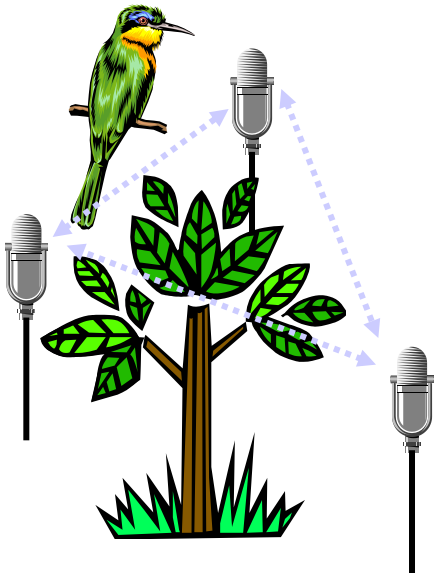
- Cost:
  - Node cost: Power? $? Time?
  - Infrastructure cost? Installation cost?
- Form factor:
  - Baseline of sensor array
- Communications Requirements:
  - Network topology: cluster head vs. local determination
  - What kind of coordination among nodes?
- Environment:
  - Indoor? Outdoor? On Mars?
- Is the target known? Is it cooperating?

Variety of Mechanisms

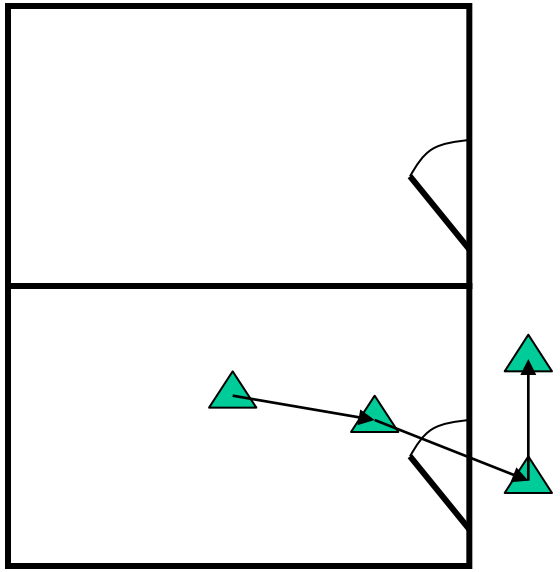
- We've seen a broad spectrum of application requirements
- There are also a broad spectrum of localization mechanisms appropriate for different applications

Returning to our two Applications…

- Choice of mechanisms differs:



Passive habitat monitoring:
Minimize environ. interference
No two birds are alike

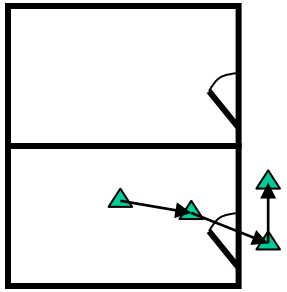


Asset tracking:
Controlled environment
We know exactly what tag is like

Variety of Localization Mechanisms

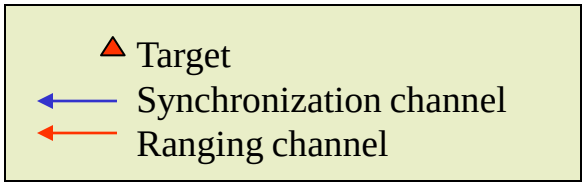
- Very different mechanisms indicated!

  - Bird is not tagged
    - Passive detection of bird presence
  - Birdcall is characteristic but not exactly known
  - Bird does not have radio; TDOA (Time Difference of Arrival) measurement
  - Passive target localization
    - Requires
      - Sophisticated detection
      - Coherent beamforming
      - Large data transfers

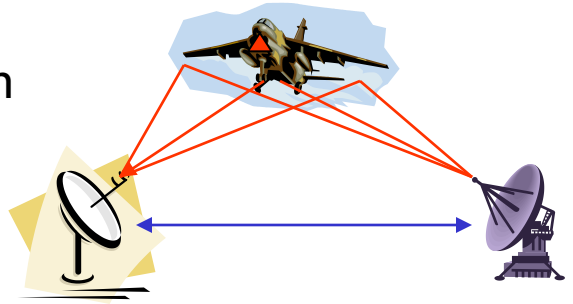


  - Projector is tagged
    - Projector might know it had moved
  - Signals from projector tag can be engineered
  - Tag can use radio signal to enable TOF (Time of Flight) measurement
  - Cooperative Localization
    - Requires
      - Basic correlator
      - Simple triangulation
      - Minimal data transfers

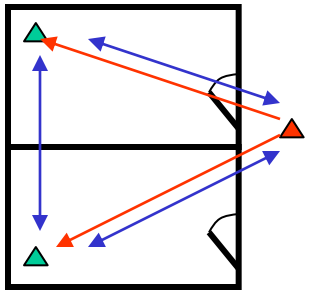
Active Mechanisms



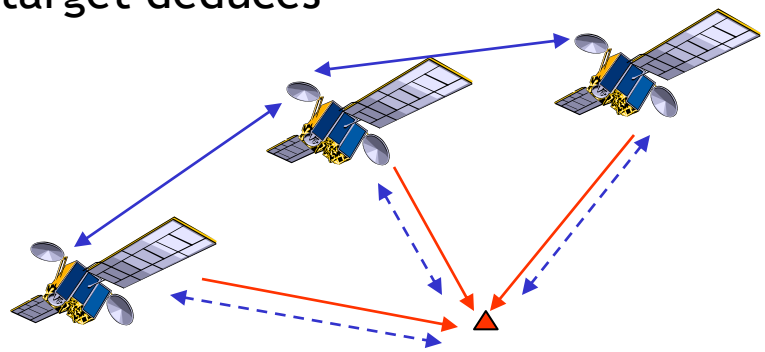
- Non-cooperative
  - System emits signal, deduces target location from distortions in signal returns
  - e.g. radar and reflective sonar systems



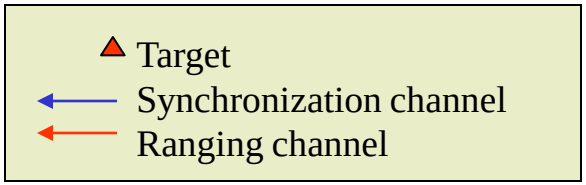
- Cooperative Target
  - Target emits a signal with known characteristics; system deduces location by detecting signal
  - e.g. ORL Active Bat, GALORE Panel, AHLoS



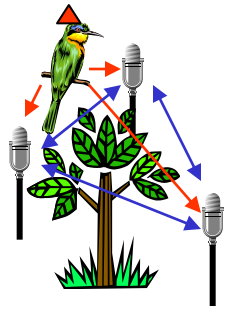
- Cooperative Infrastructure
  - Elements of infrastructure emit signals; target deduces location from detection of signals
  - e.g. GPS, MIT Cricket



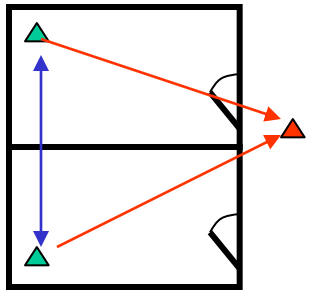
Passive Mechanisms



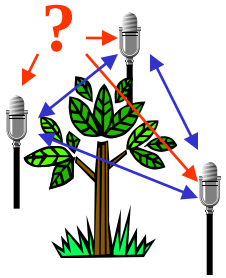
- Passive Target Localization
  - Signals normally emitted by the target are detected (e.g. birdcall)
  - Several nodes detect candidate events and cooperate to localize it by cross-correlation



- Passive Self-Localization
  - A single node estimates distance to a set of beacons (e.g. 802.11 bases in RADAR [Bahl et al.], Ricochet in Bulusu et al.)



- Blind Localization
  - Passive localization without a priori knowledge of target characteristics
  - Acoustic “blind beamforming” (Yao et al.)



Active vs. Passive

- Active techniques tend to work best
  - Signal is well characterized, can be engineered for noise and interference rejection
  - Cooperative systems can synchronize with the target to enable accurate time-of-flight estimation
- Passive techniques
  - Detection quality depends on characterization of signal
  - Time difference of arrivals only; must surround target with sensors or sensor clusters
    - **TDOA requires precise knowledge of sensor positions**
- Blind techniques
  - Cross-correlation only; may increase communication cost
  - Tends to detect “loudest” event.. May not be noise immune

Active and Cooperative Ranging

- Measurement of distance between two points
  - Acoustic
    - Point-to-point time-of-flight, using RF synchronization
    - Narrowband (typ. ultrasound) vs. Wideband (typ. audible)
  - RF
    - RSSI from multiple beacons
    - Transponder tags (rebroadcast on second frequency), measure round-trip time-of-flight.
    - UWB ranging (averages many round trips)
    - Psuedoranges from phase offsets (GPS)
    - TDOA to find bearing, triangulation from multiple stations
  - Visible light
    - Stereo vision algorithms
      - Need not be cooperative, but cooperation simplifies the problem

Passive and Non-cooperative Ranging

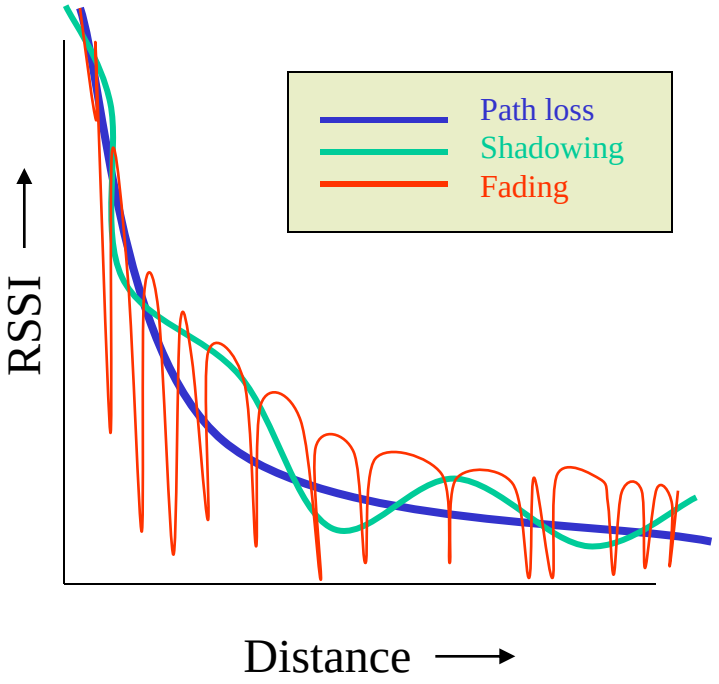
- Generally less accurate than active/cooperative
  - Acoustic
    - Reflective time-of-flight (SONAR)
    - Coherent beamforming (Yao et al.)
  - RF
    - Reflective time-of-flight (RADAR systems)
    - “Database” techniques
      - RADAR (Bahl et al.) looks up RSSI values in database
      - “RadioCamera” is a technique used in cellular infrastructure; measures multipath signature observed at a base station
  - Visible light
    - Laser ranging systems
      - Commonly used in robotics; very accurate
      - Disadvantages: directionality, expense, no positive ID of target

Using RF for Ranging

- RF TOF techniques
  - Accurate, deterministic transponders hard to build
    - Temperature-dependence problems in timing of path from receiver to transmitter
    - But, you can use “RBS” techniques… (compare receptions)
  - Measuring TOF requires fast, synchronized clocks to achieve high precision ($c \approx 1$ ft/ns)
    - Fast synchronized clocks generally at odds with low power
    - Trade-off: synchronized infrastructure vs. nodes (e.g. GPS)
- Ultra wide-band ranging for sensor nets?
  - Current research focus in RF community
  - Based on very short wideband pulses, measure RTT to fixed, surveyed base stations
  - FCC licensing?

Practical Difficulties with RSSI

- RSSI is extremely problematic for fine-grained, ad-hoc applications
  - Path loss characteristics depend on environment ($1/r^n$)
  - Shadowing depends on environment
  - Short-scale fading due to multipath adds random high frequency component with huge amplitude (30-60dB) – very bad indoors
    - Mobile nodes might average out fading.. But static nodes can be stuck in a deep fade forever
  - The relative orientation of antennas among nodes makes difference.
- Potential applications
  - Approximate localization of mobile nodes, proximity determination
  - "Database" techniques (RADAR)



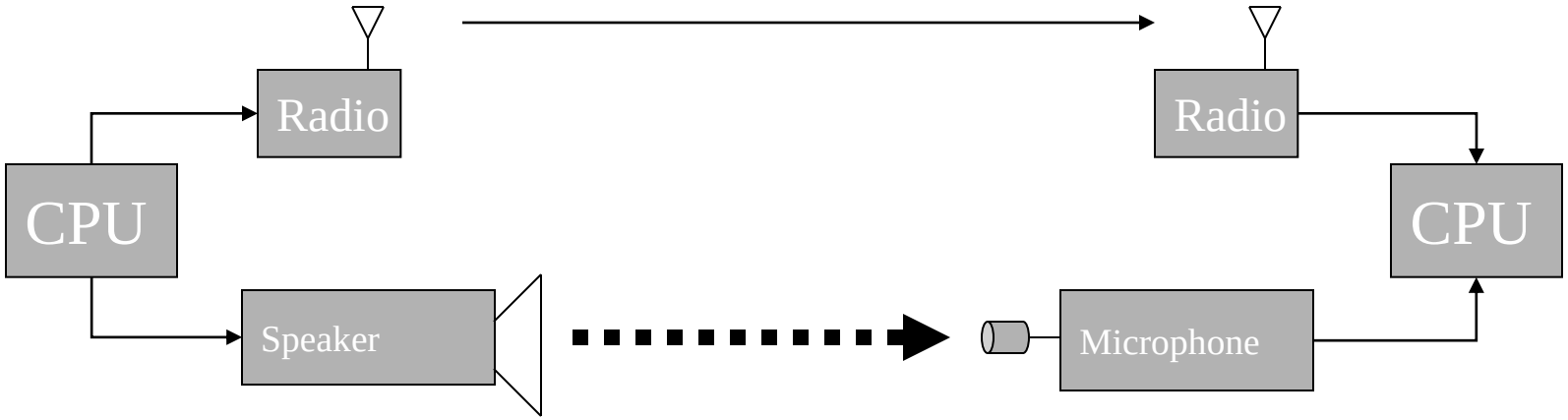
Ref. Rappaport, T, *Wireless Communications Principle and Practice*, Prentice Hall, 1996.

Using Acoustics for Ranging

- Key observation: Sound travels slowly!
  - Tight synchronization easily achieved using RF signaling
  - Slow clocks are sufficient (v = 1 ft/ms)
  - With LOS, high accuracy can be achieved cheaply
  - Coherent beamforming can be achieved with low sample rates
- Advantages
  - Acoustics have lower path loss than RF near the ground, because ground reflections in acoustics don't cancel
  - Audible acoustics have very wide range of wavelengths
- Disadvantages
  - Poor penetration $\Rightarrow$ detector picks up reflections in Non-LOS
  - Audible sound: good channel properties, but often inappropriate

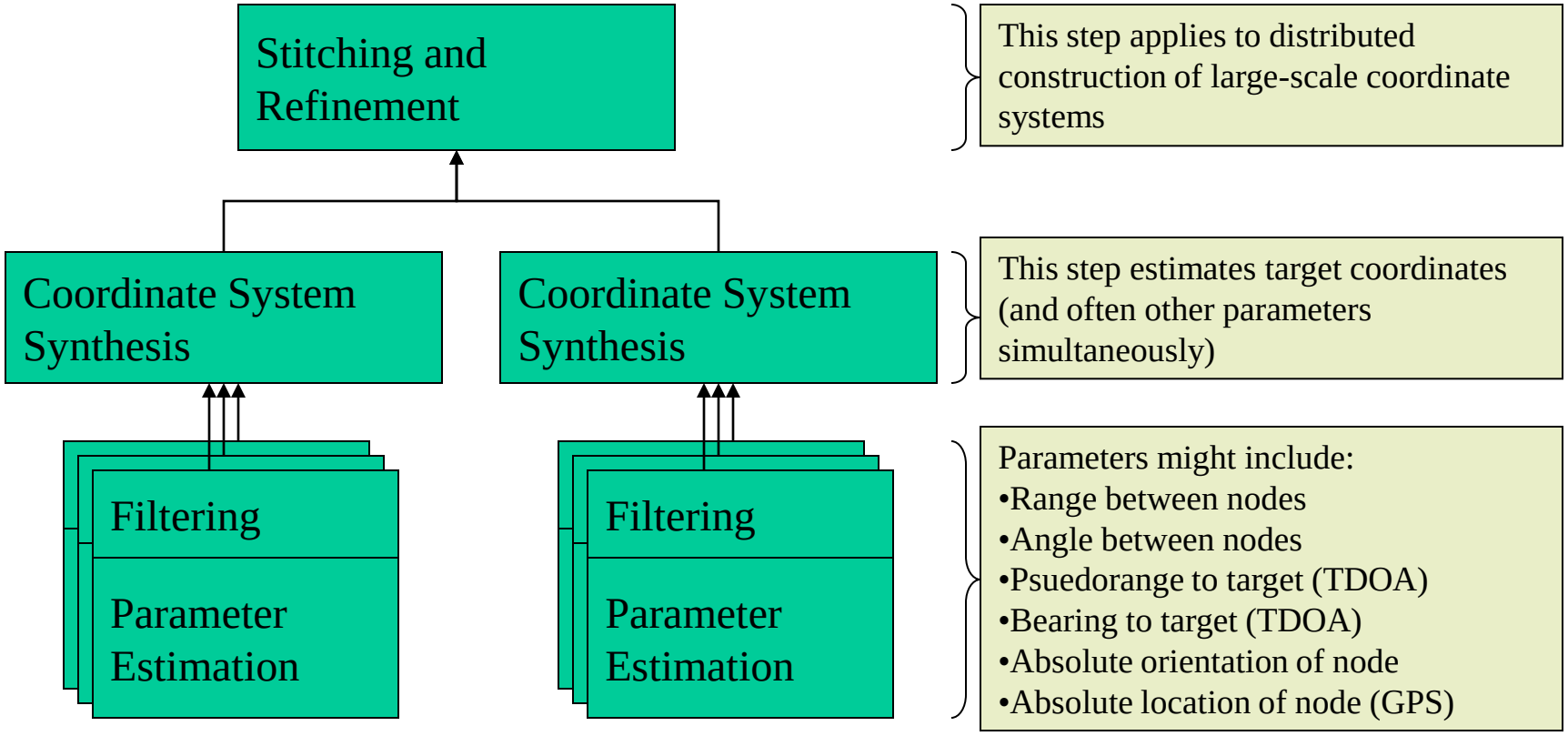
Typical Time-of-Flight AR System

- Radio channel is used to synchronize the sender and receiver (or use a service like RBS!)
- Coded acoustic signal is emitted at the sender and detected at the emitter. TOF determined by comparing arrival of RF and acoustic signals



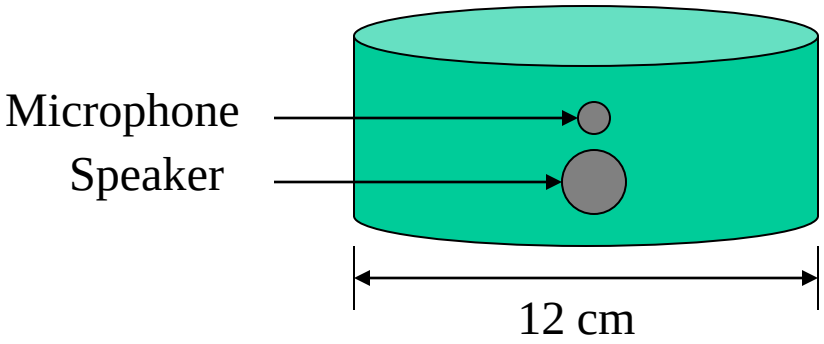
Localization System Components

- Generally speaking, what is involved with a "localization system"?



Example of a Localization System

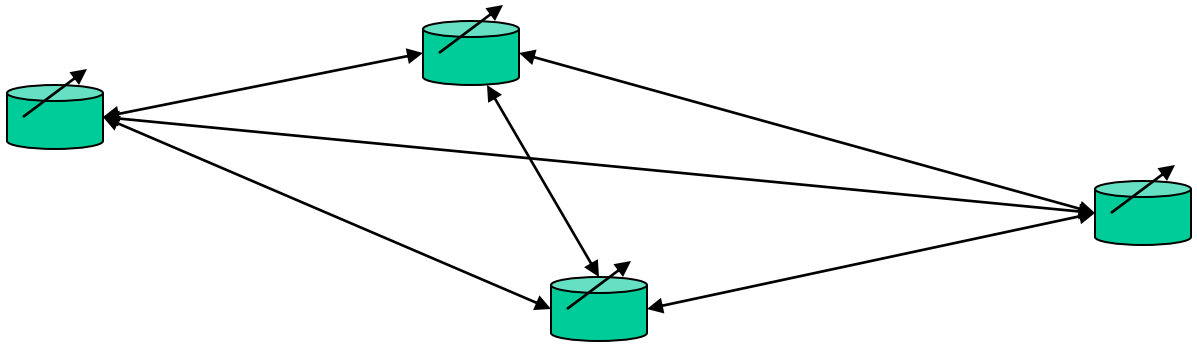
- SHM system, developed at Sensoria Corp.



Each node has 4 speaker/ microphone pairs, arranged along the circumference of the enclosure. The node also has a radio system and an absolute orientation sensor that senses magnetic north.

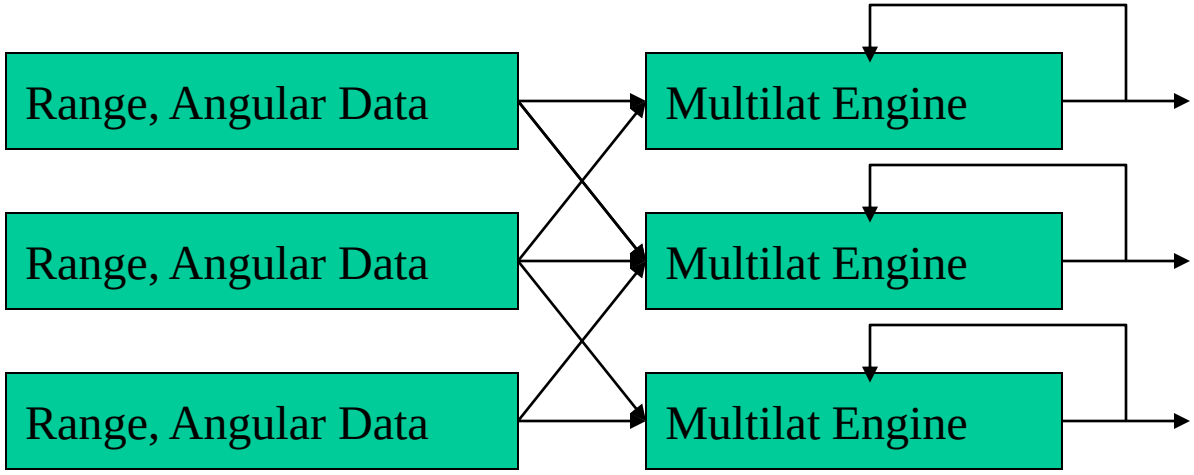
System Architecture

- Ranging between nodes based on detection of coded acoustic signals, with radio synchronization to measure time of flight
- Angle of arrival is determined through TDOA and is used to estimate bearing, referenced from the absolute orientation sensor
- An onboard temperature sensor is used to compensate for the effect of environmental conditions on the speed of sound



System Architecture

- Nodes periodically emit acoustic pulses. Other nodes detect these pulses and compute a range and angle of arrival.
- Range data, angle data, and absolute orientation are broadcast N hops away.
- Based on this table of ranges, angles, and orientations, each node applies a multilateration algorithm with iterative outlier rejection to compute a consistent coordinate system.



Overview

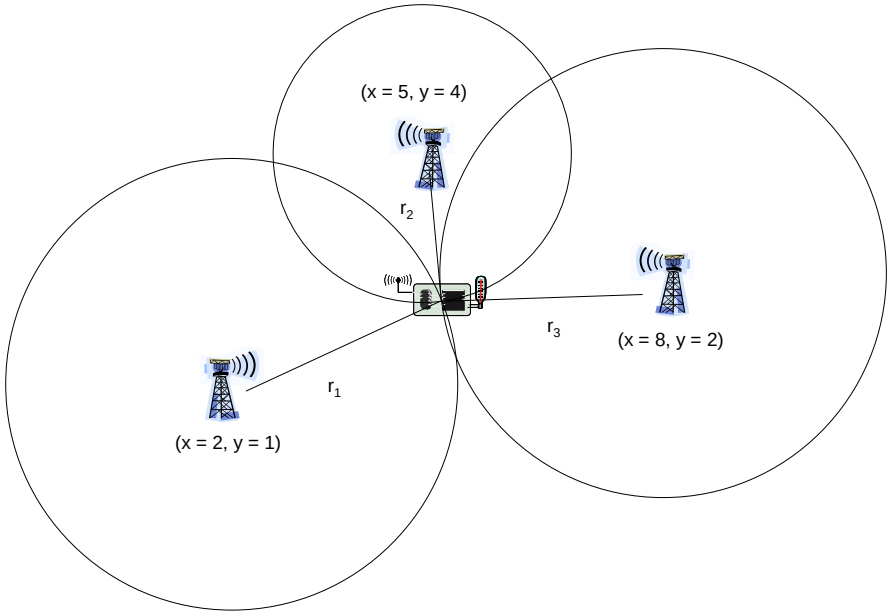
- ***Basic approaches***
- Trilateration
- Multihop schemes

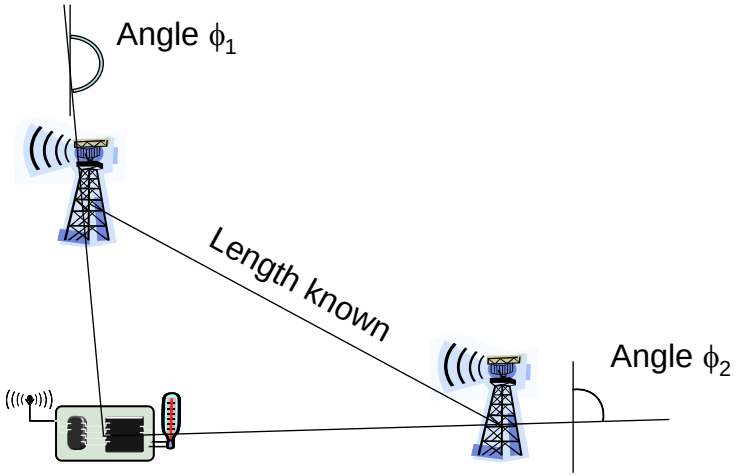
Localization & positioning

- Determine ***physical position*** or ***logical location***
  - Coordinate system or symbolic reference
  - Absolute or relative coordinates
- Options
  - Centralized or distributed computation
  - Scale (indoors, outdoors, global, …)
  - Sources of information
- Metrics
  - Accuracy (how close is an estimated position to the real position?)
  - Precision (for repeated position determinations, how often is a given accuracy achieved?)
  - Costs, energy consumption, …

Main approaches (information sources)

- Proximity
  - Exploit finite range of wireless communication
  - E.g.: easy to determine location in a room with infrared room number announcements
- (Tri-/Multi-)***lateration*** and ***angulation***
  - Use distance or angle estimates, simple geometry to compute position estimates
- Scene analysis
  - Radio environment has characteristic "signatures"
  - Can be measured beforehand, stored, compared with current situation



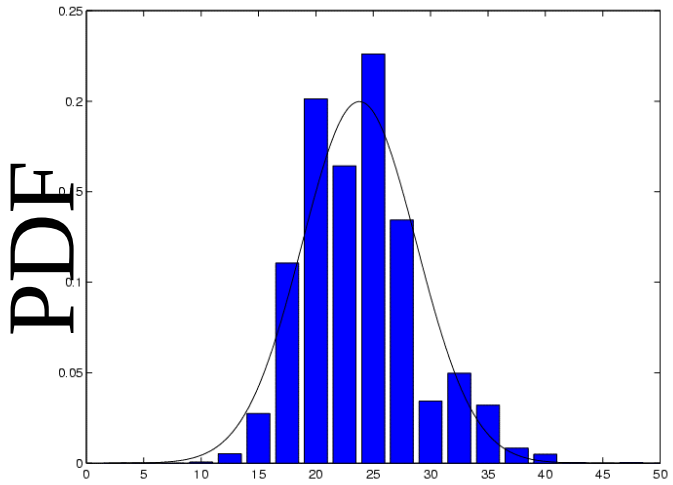


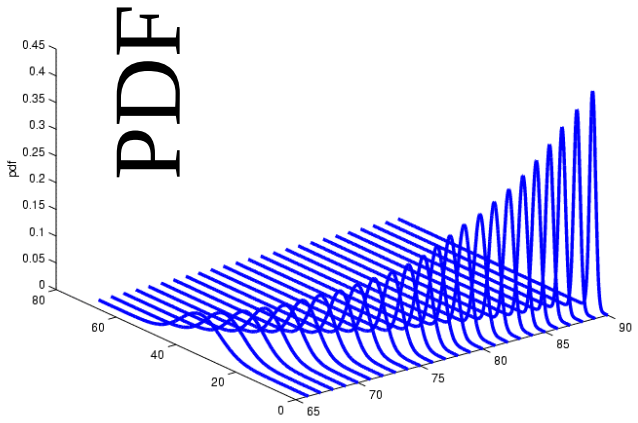
Estimating distances - RSSI

- Received Signal Strength Indicator
  - Send out signal of known strength, use received signal strength and path loss coefficient to estimate distance

$$P_{\text{recv}} = c\frac{P_{\text{tx}}}{d^{\alpha}} \Leftrightarrow d = \sqrt[\alpha]{\frac{cP_{\text{tx}}}{P_{\text{recv}}}}$$

  - Problem: Highly error-prone process – Shown: PDF for a fixed RSSI





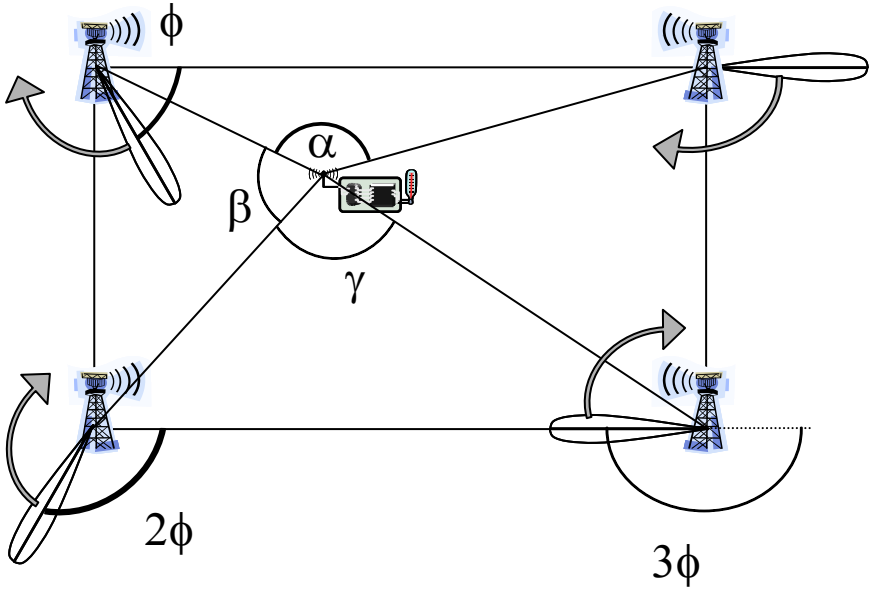
Estimating distances – other means

- Time of arrival (ToA)
  - Use time of transmission, propagation speed, time of arrival to compute distance
  - Problem: Exact time synchronization

- Time Difference of Arrival (TDoA)
  - Use two different signals with different propagation speeds
  - Example: ultrasound and radio signal
    - Propagation time of radio negligible compared to ultrasound
  - Compute difference between arrival times to compute distance
  - Problem: Calibration, expensive/energy-intensive hardware

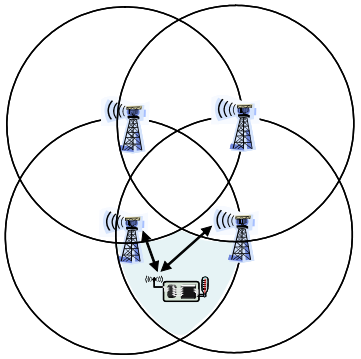
Determining angles

- Directional antennas
  - On the node
  - Mechanically rotating or electrically “steerable”
  - On several access points
    - Rotating at different offsets
    - Time between beacons allows to compute angles

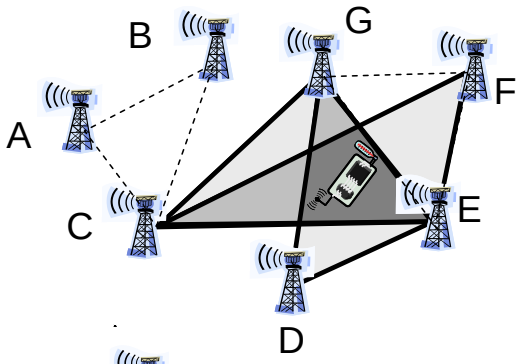


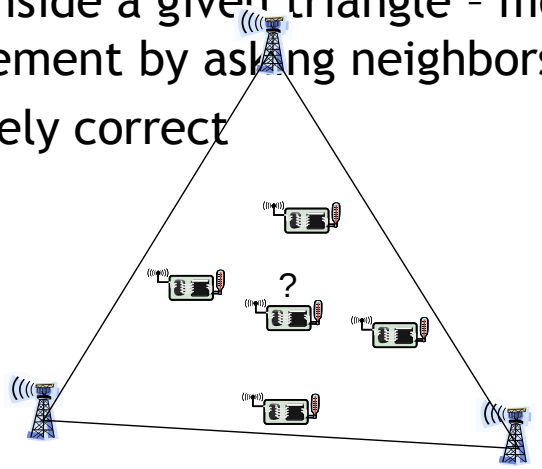
Some range-free, single-hop localization techniques

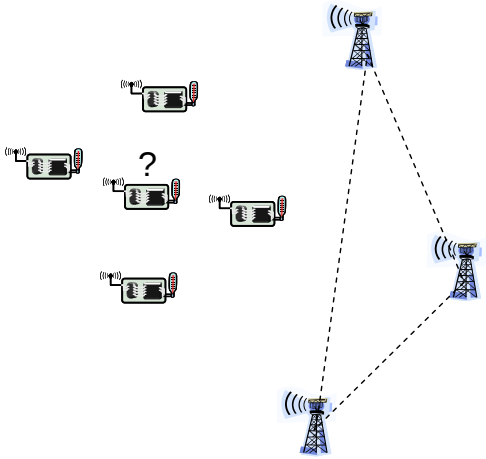
- ***Overlapping connectivity***: Position is estimated in the center of area where circles from which signal is heard/not heard overlap



- ***Approximate point in triangle***
  - Determine triangles of anchor nodes where node is inside, overlap them
  - Check whether inside a given triangle – move node or simulate movement by asking neighbors
  - Only approximately correct







Overview

- Basic approaches
- ***Trilateration***
- Multihop schemes

Trilateration

- Assuming distances to three points with known location are exactly given
- Solve system of equations (Pythagoras!)
  - (x_i, y_i) : coordinates of ***anchor point*** i, r_i distance to anchor i
  - (x_u, y_u) : unknown coordinates of node

$$(x_i - x_u)^2 + (y_i - y_u)^2 = r_i^2 \text{ for } i = 1, \ldots, 3$$

  - Subtracting eq. 3 from 1 & 2:

$$(x_1 - x_u)^2 - (x_3 - x_u)^2 + (y_1 - y_u)^2 - (y_3 - y_u)^2 = r_1^2 - r_3^2$$
$$(x_2 - x_u)^2 - (x_2 - x_u)^2 + (y_2 - y_u)^2 - (y_2 - y_u)^2 = r_2^2 - r_3^2.$$

  - Rearranging terms gives a linear equation in (x_u, y_u)!

$$2(x_3 - x_1)x_u + 2(y_3 - y_1)y_u = (r_1^2 - r_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2)$$
$$2(x_3 - x_2)x_u + 2(y_3 - y_2)y_u = (r_2^2 - r_2^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2)$$

Trilateration as matrix equation

- Rewriting as a matrix equation:

$$2\begin{bmatrix} x_3 - x_1 & y_3 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{bmatrix}\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} (r_1^2 - r_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) \\ (r_2^2 - r_2^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2) \end{bmatrix}$$

- Example: $(x_1, y_1) = (2,1)$, $(x_2, y_2) = (5,4)$, $(x_3, y_3) = (8,2)$,

$r_1 = 10^{0.5}$ $r_2 = 2$ $r_3 = 3$

$$2\begin{bmatrix} 6 & 1 \\ 3 & -2 \end{bmatrix}\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} 64 \\ 22 \end{bmatrix}$$

! $(x_u, y_u) = (5,2)$

Trilateration with distance errors

- What if only distance estimation $r_i^0 = r_i + \varepsilon_i$ available?

$$2\begin{bmatrix} x_n - x_1 & y_n - y_1 \\ \vdots & \vdots \\ x_n - x_{n-1} & y_n - y_{n-1} \end{bmatrix} \begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} (r_1^2 - r_n^2) - (x_1^2 - x_n^2) - (y_1^2 - y_n^2) \\ \vdots \\ (r_{n-1}^2 - r_n^2) - (x_{n-1}^2 - x_n^2) - (y_{n-1}^2 - y_n^2) \end{bmatrix}$$

$$\|\mathbf{Ax} - \mathbf{b}\|_2$$

- Use (x_u, y_u) that minimize mean square error, i.e,

Minimize mean square error

- Look at square of the of Euclidean norm expression (note that $\|\mathbf{v}\|_2^2 = \mathbf{v}^{\mathrm{T}}\mathbf{v}$ for all vectors v)

$$\|\mathbf{Ax}-\mathbf{b}\|_2^2 = (\mathbf{Ax}-\mathbf{b})^{\mathrm{T}}(\mathbf{Ax}-\mathbf{b}) = \mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{Ax} - 2\mathbf{x}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{b} + \mathbf{b}^{\mathrm{T}}\mathbf{b}$$

$$2\mathbf{A}^{\mathrm{T}}\mathbf{Ax} - 2\mathbf{A}^{\mathrm{T}}\mathbf{b} = 0 \Leftrightarrow \mathbf{A}^{\mathrm{T}}\mathbf{Ax} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

Look at derivative with respect to x, set it equal to 0:

  - ***Normal equation***
  - Has unique solution (if A has full rank), which gives desired minimal mean square error

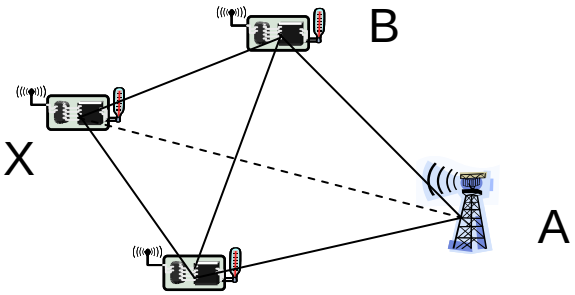
- Essentially similar for angulation as well

Overview

- Basic approaches
- Trilateration
- ***Multihop schemes***

Multihop range estimation

- How to estimate range to a node to which no direct radio communication exists?
  - No RSSI, TDoA, …
  - But: Multihop communication is possible

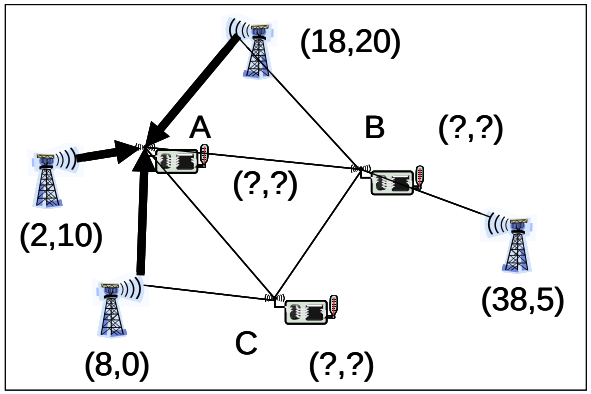


- Solutions:
  - Idea 1: Count number of hops, assume length of one hop is known (***DV-Hop***)
    - Start by counting hops between anchors, divide known distance
  - Idea 2: If range estimates between neighbors exist, use them to improve total length of route estimation in previous method (***DV-Distance***)
- Then, in presence of range estimates and a sufficient number of neighbors, a node can actually try to compute its true Euclidean distance to a faraway anchor.

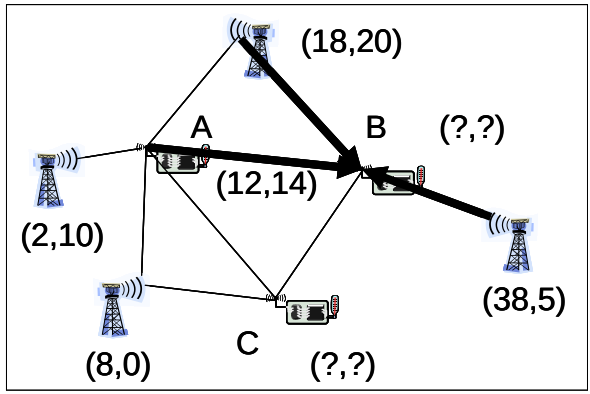
Iterative multilateration

- Assume some nodes can hear at least three anchors (to perform triangulation), but not all
- Idea: let more and more nodes compute position estimates, spread position knowledge in the network
  - Problem: Errors accumulate

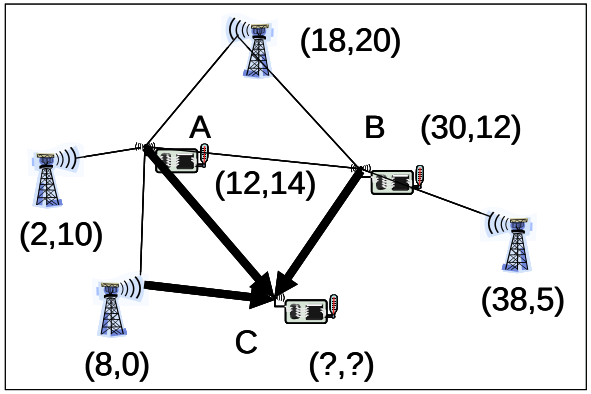
I:



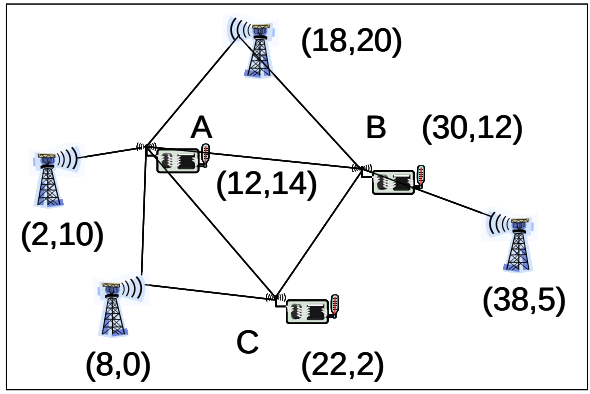
II:



III:

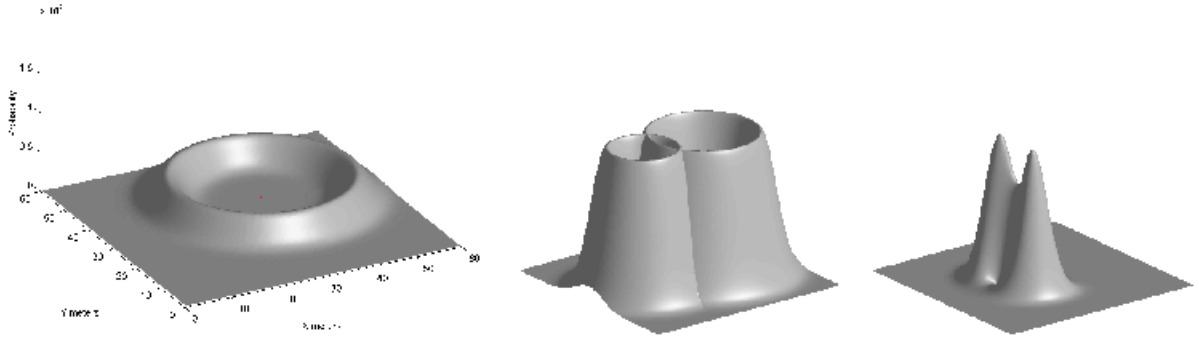


IV:



Probabilistic position description

- Similar idea to previous one, but accept problem that position of nodes is only probabilistically known
  - Represent this probability explicitly, use it to compute probabilities for further nodes



(a) Probability density function of a node positions after receiving a distance estimate from one anchor

(b) Probability density functions of two distance measurements from two independent anchors

(c) Probability density function of a node after intersecting two anchor's distance measurements

Connectivity in multihop networks

What if the distances between nodes are know?

Can we compute a coordinate system without beacons (or anchors)?

Yes, one possible way is to use MDS…

Obtaining a Coordinate System from Distance Measurements: Introduction to MDS

MDS maps objects from a high-dimensional space to a low-dimensional space,
while preserving distances between objects.

similarity between objects ⟶ coordinates of points

Classical metric MDS:

- The simplest MDS: the proximities are treated as distances in an Euclidean space
- Optimality: LSE sense. Exact reconstruction if the proximity data are from an Euclidean space
- Efficiency: singular value decomposition, $O(n^3)$

LOCALIZATION USING MDS-MAP (Shang, et al., Mobihoc'03)

The basic MDS-MAP algorithm:

1. Given connectivity or local distance measurement, compute shortest paths between all pairs of nodes.

2. Apply multidimentional scaling (MDS) to construct a relative map containing the positions of nodes in a local coordinate system.

3. Given sufficient anchors (nodes with known positions), e.g, 3 for 2-D or 4 for 3-D networks, transform the relative map and determine the absolute positions of the nodes.

It works for any n-dimensional networks, e.g., 2-D or 3-D.

Applying Classical MDS

1. Create a proximity matrix of distances D
2. Convert into a double-centered matrix B

$$\mathrm{B} = -\frac{1}{2}\left(I - \frac{1}{N}U\right)D^2\left(I - \frac{1}{N}U\right)$$

NxN identity matrix

NxN matrix of 1s

NxN matrix of 1s

$$B = VAV^T$$

3. Take the Singular Value Decomposition of B

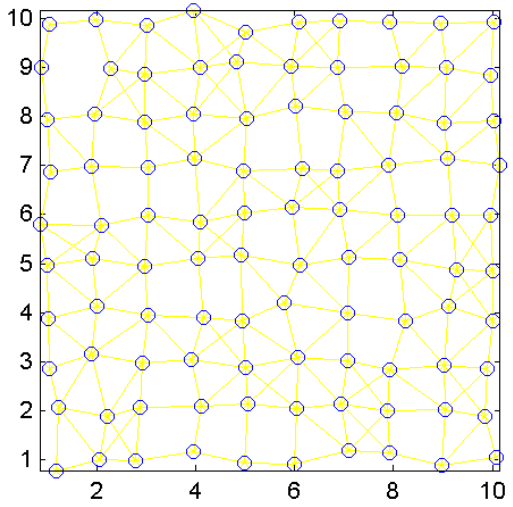
$$X = VA^{\frac{1}{2}}$$

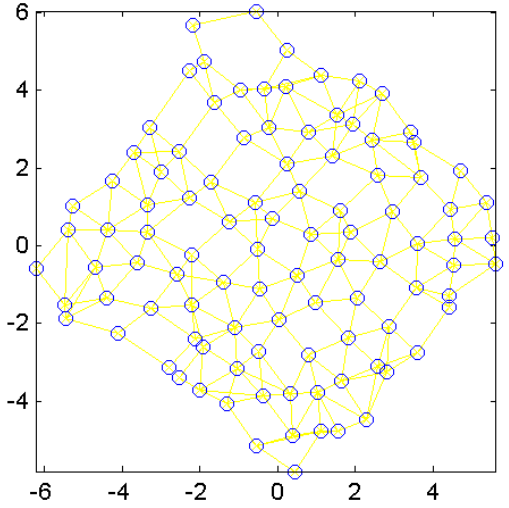
4. Compute the coordinate matrix X (2D coordinates will be in the first 2 columns)

Example: Localization Using Multidimensional Scaling (MDS) (Yi Shang et. al)

The basic MDS-MAP algorithm:

1. Compute shortest paths between all pairs of nodes.
2. Apply classical MDS and use its result to construct a relative map.





3. Given sufficient anchor nodes, transform the relative map to an absolute map.

MDS-MAP ALGORITHM

1. Compute all-pair shortest paths. $O(n^3)$
   Assigning values to the edges in the connectivity graph:
   Known connectivity only: all edges have value 1 (or R/2)
   Known neighbor distances: the edges have the distance values

2. Apply classical MDS and use its result to construct a 2-D (or 3-D) relative map. $O(n^3)$

3. Given sufficient anchor nodes, convert the relative map to an absolute map via a linear transformation. $O(n+m^3)$

- Compute the LSE transformation based on the positions of anchors. $O(m^3)$, m is the number of anchors
- Apply the transformation to the other unknown nodes. $O(n)$

MDS-MAP (P) – The Distributed Version

1. Set-up the range for local maps R_{lm} (# of hops to consider in a map)
2. Compute maps of individual nodes
    1. Compute shortest paths between all pairs of nodes
    2. Apply MDS
    3. Least-squares refinement
3. Patch the maps together
    - Randomly pick a node and build a local map, then merge the neighbors and continue until the whole network is completed
4. If sufficient anchor nodes are present, transform the relative map to an absolute map

MDS-MAP(P,R) – Same as MDS-MAP(P) followed by a refinement phase

MDS-MAP(P) (Shang and Ruml, Infocom'04)

The basic MDS-MAP works well on regularly shaped networks, but not on irregularly shaped networks.

MDS-MAP(P) (or *MDS-MAP based on patches of local maps*)

1. For each node, compute a local relative map using MDS
2. Merge/align local maps to form a big relative map
3. Refine the relative map based on the relative positions (optional). *(When used, referred to as MDS-MAP(P,R))*
4. Given sufficient anchors, compute absolute positions
5. Refine the positions of individual nodes based on the absolution positions (optional)

SOME IMPLEMENTATION DETAILS OF MDS-MAP(P)

1. For each node, compute a local relative map using MDS
   - Size of local maps: fixed or adaptive

2. Merge/align local maps to form a big relative map
   - Sequential or distributed; scaling or not

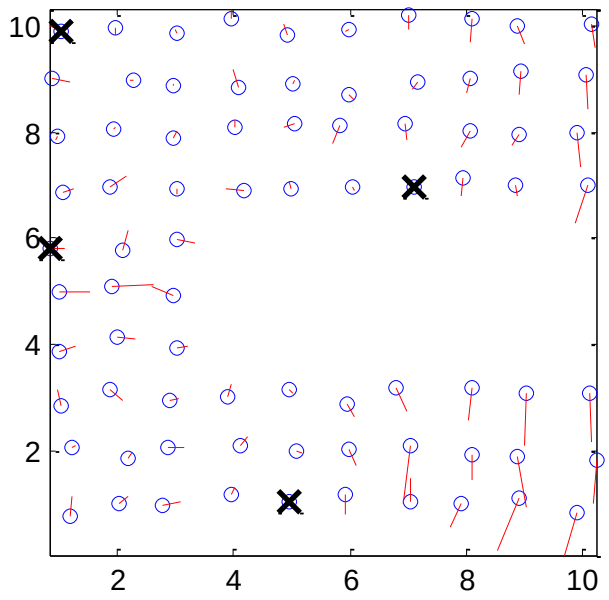
3. Refine the relative map based on the relative positions
   - Least squares minimization: what information to use

4. Given sufficient anchors, compute absolute positions
   - Anchor selection; centralized or distributed

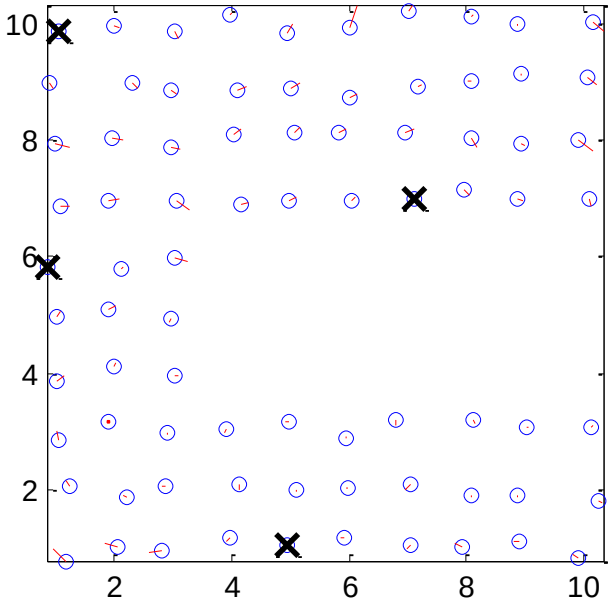
5. Refine the positions of individual nodes based on the absolution positions
   - Minimizing squared errors or absolute errors

AN EXAMPLE OF C-SHAPE GRID NETWORKS

Connectivity information only



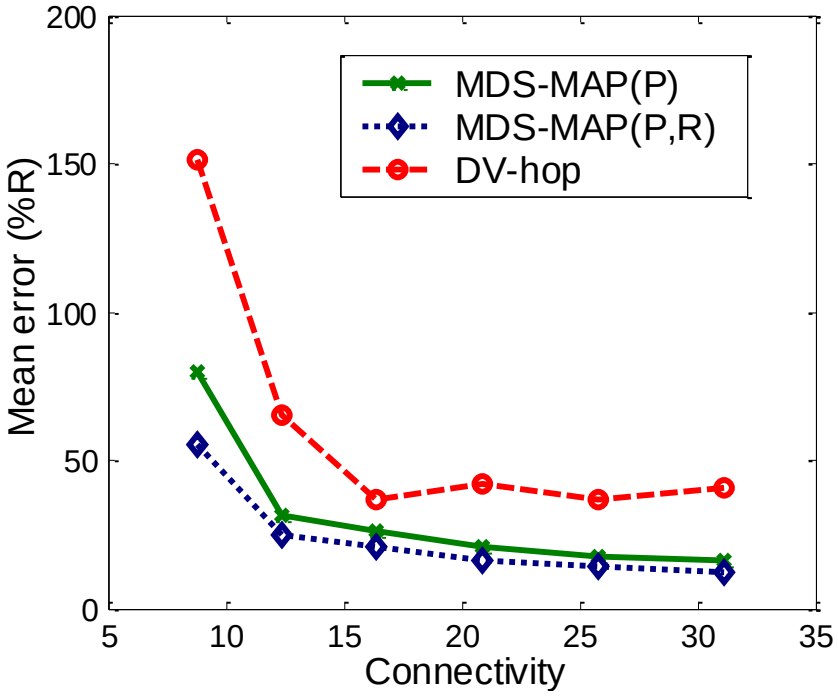
Known 1-hop distances with 5% range error



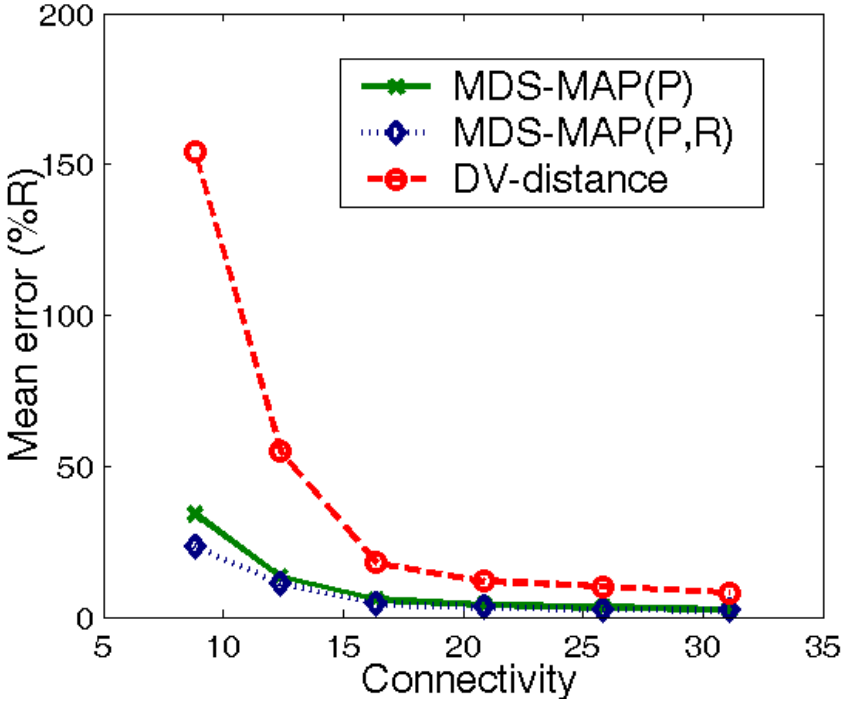
MDS-MAP(P) without both optional refinement steps.

RANDOM UNIFORM PLACEMENT

Connectivity information only



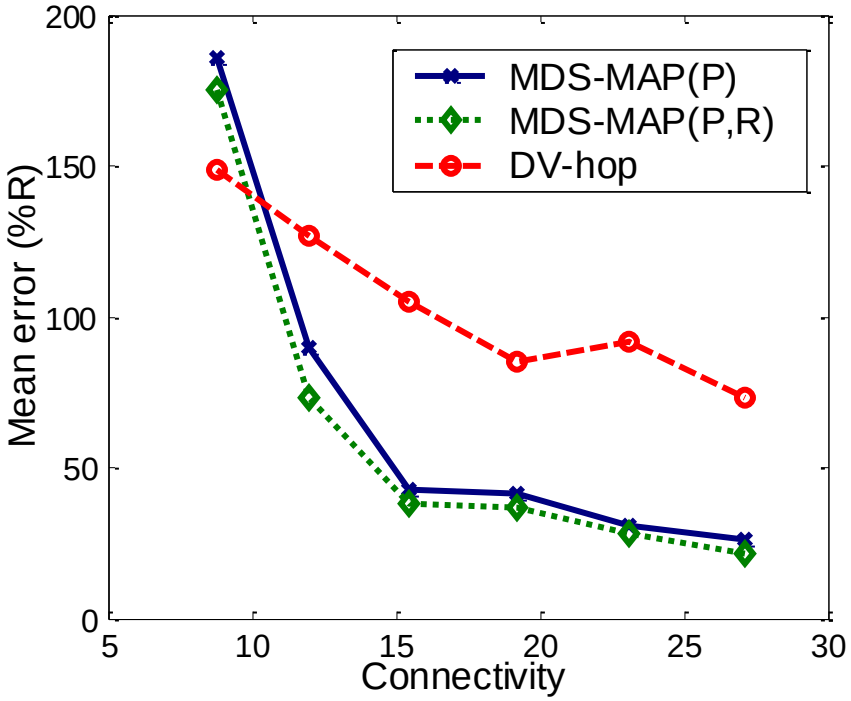
Known 1-hop distances with 5% range error



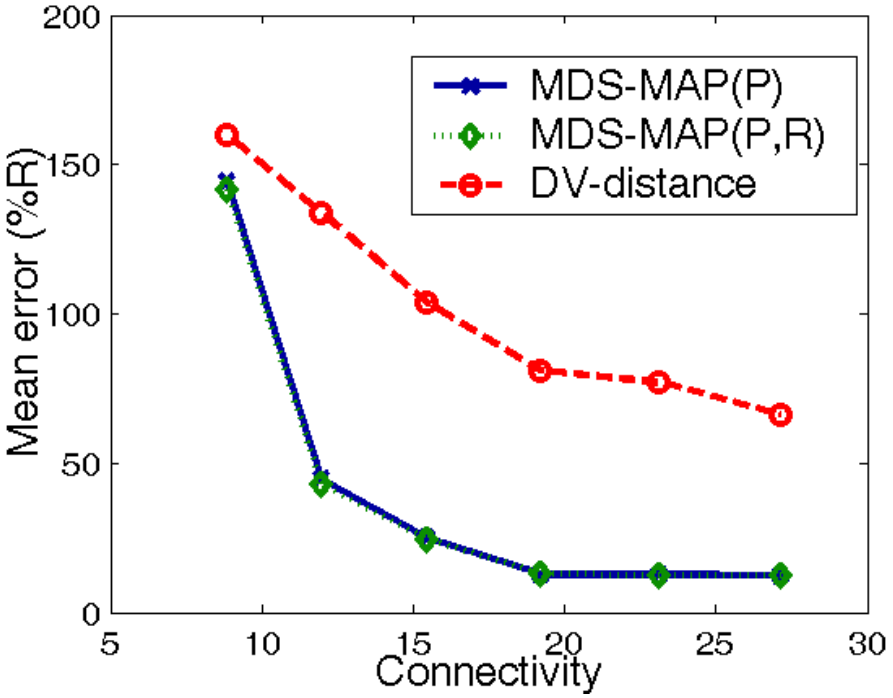
200 nodes; 4 random anchors

RANDOM C-SHAPE PLACEMENT

Connectivity information only



Known 1-hop distances with 5% range error



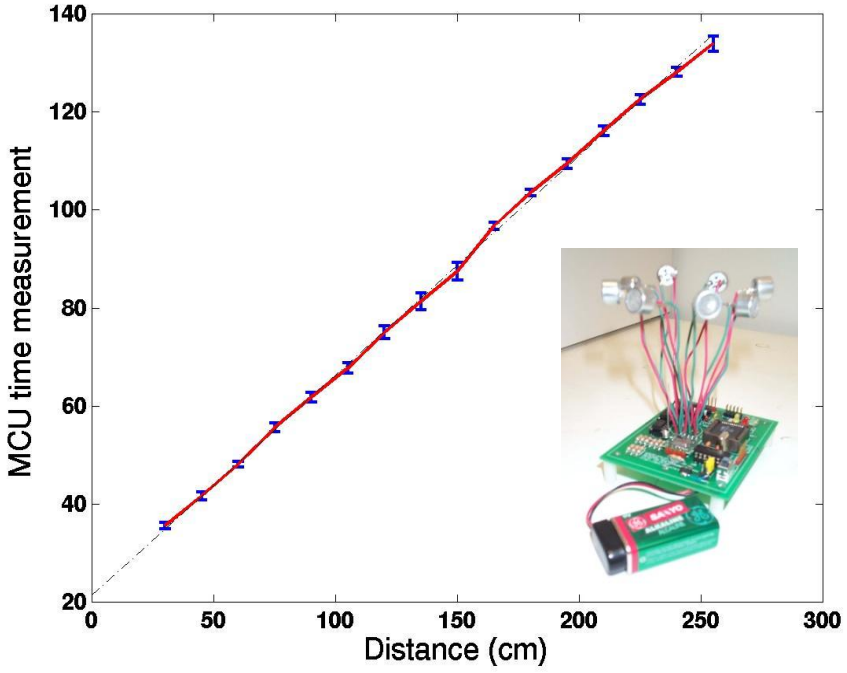
160 nodes; 4 random anchors

Where is the challenge?

- The results are numerically correct based on the used measurement model
- Two main assumptions
  - Hops are proportional to distance
  - Error represented as a function of distance
- Is that the case in practice?

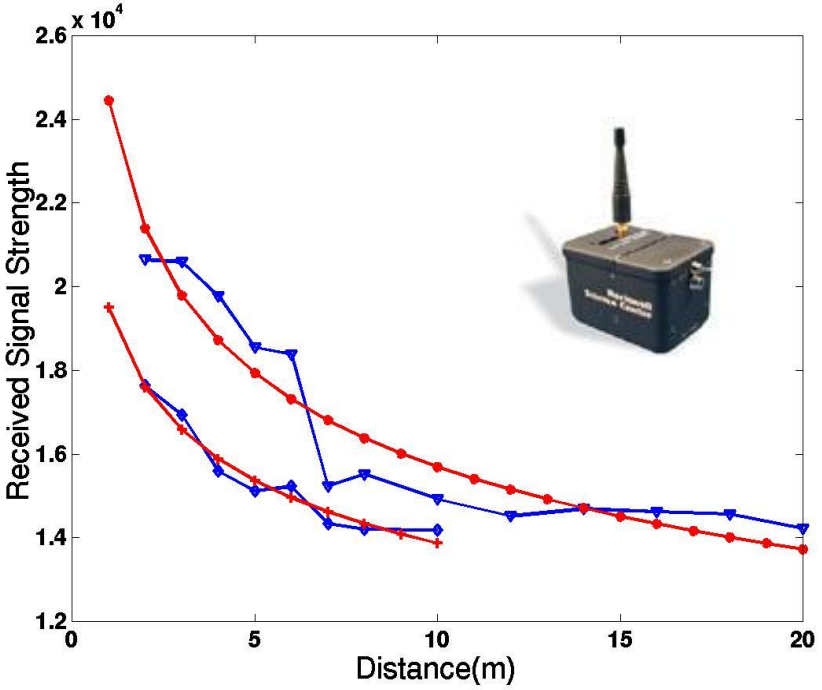
Back to the Physical Layer: RSSI In Node Localization

Ultrasound ToA



Max range 3m, accuracy 2cm

RSSI in football field



Max range 20m, accuracy 7m
Power levels: 7,10

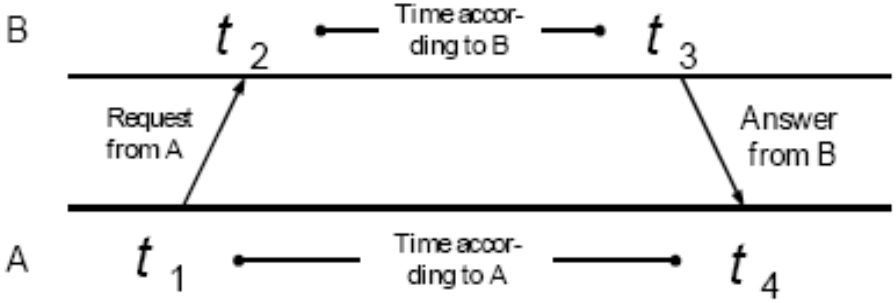
Conclusions

- Determining location or position is a vitally important function in WSN, but fraught with many errors and shortcomings
  - Range estimates often not sufficiently accurate
  - Many anchors are needed for acceptable results
  - Anchors might need external position sources (GPS)
  - Multilateration problematic (convergence, accuracy)

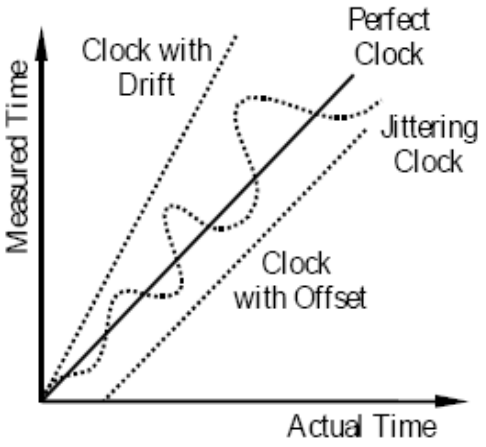
WSN SYNCHRONIZATION

Time Synchronization

- Time sync is critical at *many* layers in sensor nets
  - Coordination of wake-up and sleep times
  - TDMA schedules
  - Ordering of sensed events in habitat environments
  - Estimation of position information
  - …
- Scope of a Clock Synchronization Algorithm
  - Packet delay/latency
  - Offset between clocks
  - Drift between clocks



Up to 40 microsecond drift per second in Mica platform



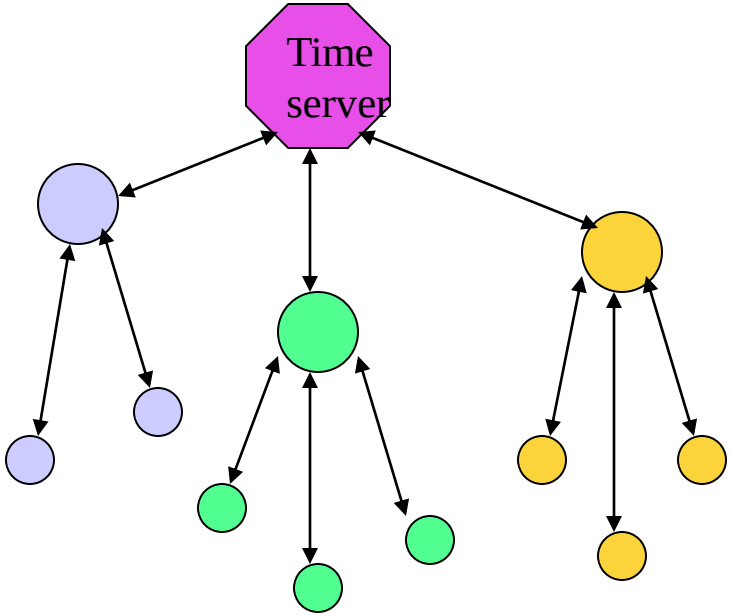
Ref: based on slides by J. Elson

Conventional Approaches

- GPS at every node
  - E.g. some GPSs provide 1 pps @ O(10ns) accuracy
  - But
    - doesn't work everywhere
    - cost, size, and energy issues
- NTP
  - some well known "primary time servers" are synchronized via GPS, atomic clock etc.
  - pre-defined server hierarchy (stratums)
  - nodes synchronize with one of a pre-specified list of time servers
  - Problems:
    - potentially long and varying paths to time-servers due to multi-hopping and short-lived links
    - delay and jitter due to MAC and store-and-forward relaying
    - discovery of time servers
  - Perfectly acceptable for most cases
    - E.g. Internet (coarse grain synchronization)
    - Inefficient when fine-grain sync is required
      - e.g. sensor net applications: localization, beamforming, TDMA etc

Network Time Protocol

Tiered architecture



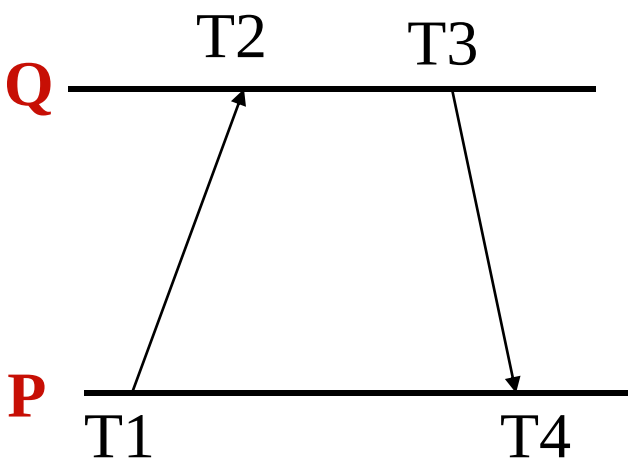
NTP [mills 1995] defines an architecture for a time service and a protocol to distribute time information over the Internet

Broadcast mode: **least accurate**

Procedure call: **medium accuracy**

Peer-to-peer mode: **higher level servers use this for max accuracy**

P2P mode of NTP



Let Q's time be ahead of P's time by δ. Then

$$T2 = T1 + T_{PQ} + \delta$$
$$T4 = T3 + T_{QP} - \delta$$

$$RTT\ y = T_{PQ} + T_{QP} = T2 + T4 - T1 - T3$$
$$\delta = (T2 - T4 - T1 + T3) / 2 - (T_{PQ} - T_{QP}) / 2$$

Ping several times, and obtain the smallest value of y. Use it to calculate δ

x

Between y/2 and -y/2

Limitations of What Exists

- Existing work is a critical building block

BUT…

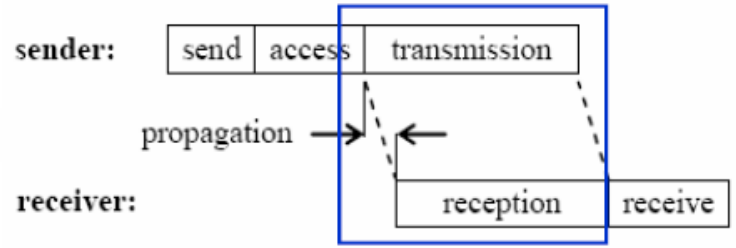
- Energy
  - e.g., we can't always be listening or using CPU!
- Wide range of requirements within a single app; no method optimal on all axes
- Cost and form factor: can disposable motes have GPS receivers, expensive oscillators? Completely changes the economics…
- Needs to be fully decentralized, infrastructure-free
- Need microsecond synchronization in certain WSN applications

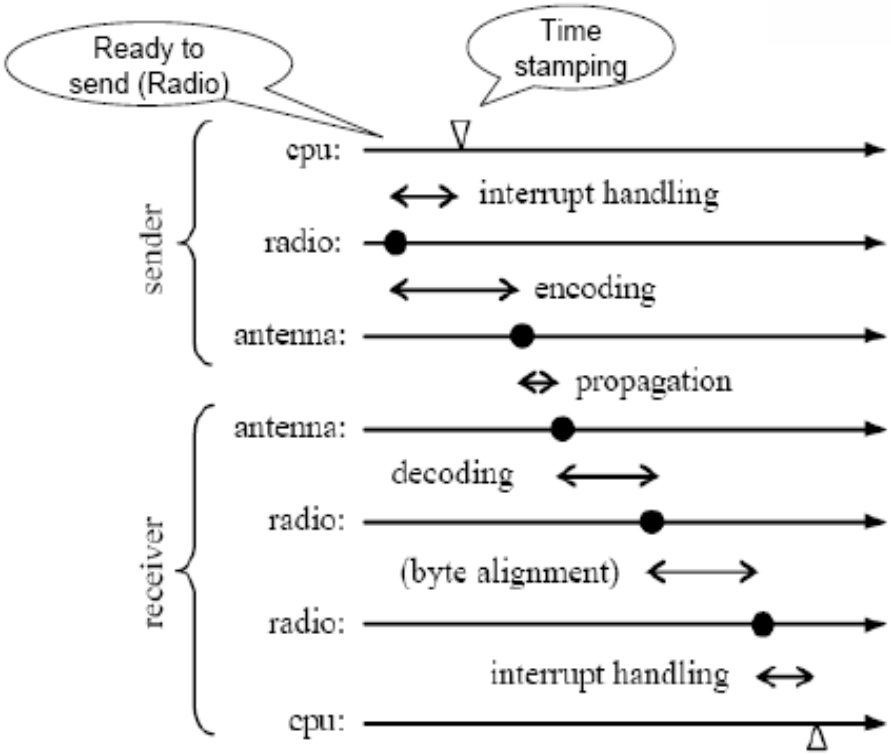
Ref: based on slides by J. Elson

Sources of time synchronization error
Common denominator: *non-determinism*

- Send time
  - Kernel processing
  - Context switches
  - Transfer from host to NIC
- Access time
  - Specific to MAC protocol
    - E.g. in Ethernet, sender must wait for clear channel

- Propagation time
  - Dominant factor in WANs
    - Router-induced delays
  - Very small in LANs
  - Asymmetric packet delays
- Receive time





Radio chip raising ~~ microcontroller responding to an interrupt

Radio chip encode and transform a part of the messages to waves

Radio chip transform and decode waves to binary data

If necessary, need to correct the byte alignment

Overview

- Protocols based on receiver/receiver synchronization
- Protocols based on sender/receiver synchronization
- Summary

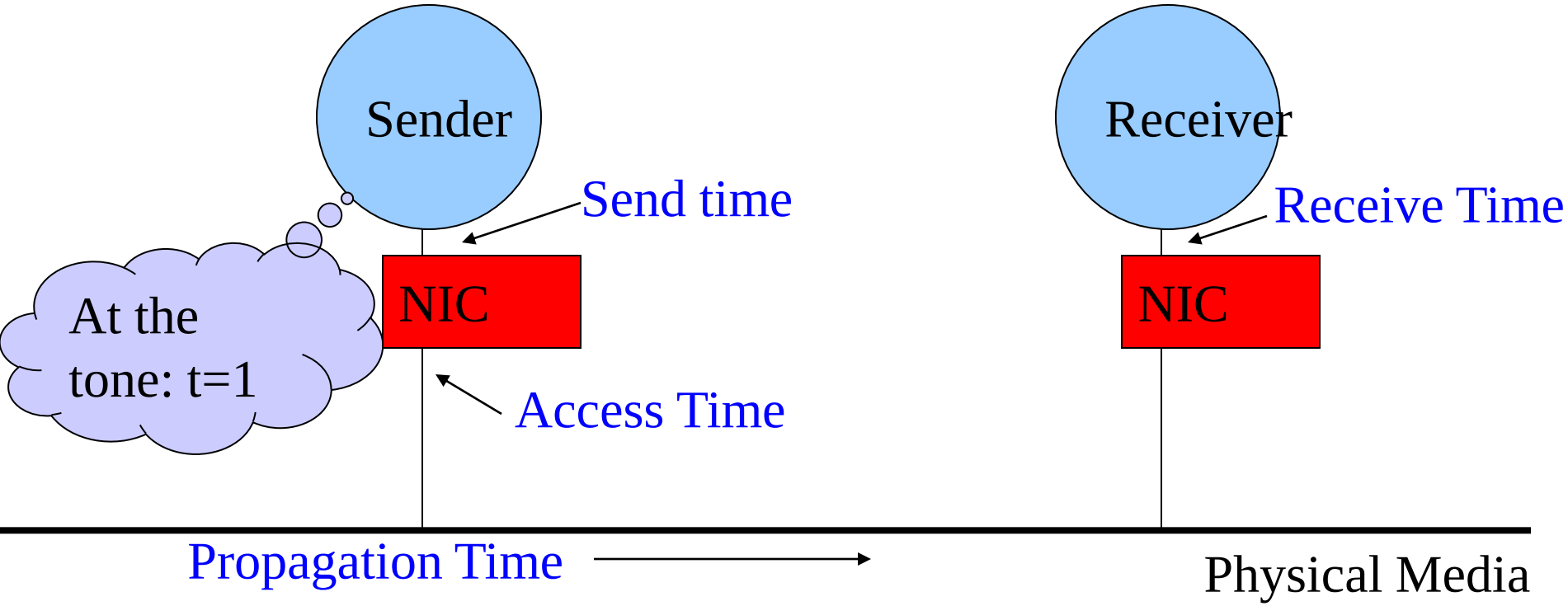
New Sync Method: Reference Broadcast

- *Reference-broadcast synchronization*: Very high precision sync with slow radios
  - Beacons are transmitted, using physical-layer broadcast, to a set of receivers
  - Time sync is based on the *difference* between reception times; don't sync sender w/ receiver!
- *Post-facto synchronization*: Don't waste energy on sync when it is not needed
  - Timestamp events using free-running clocks
  - After the fact, reconcile clocks
- *Peer-to-peer sync*: no master clock
- *Tiered Architectures*: Range of node capabilities

Ref: based on slides by J. Elson

Traditional Sync

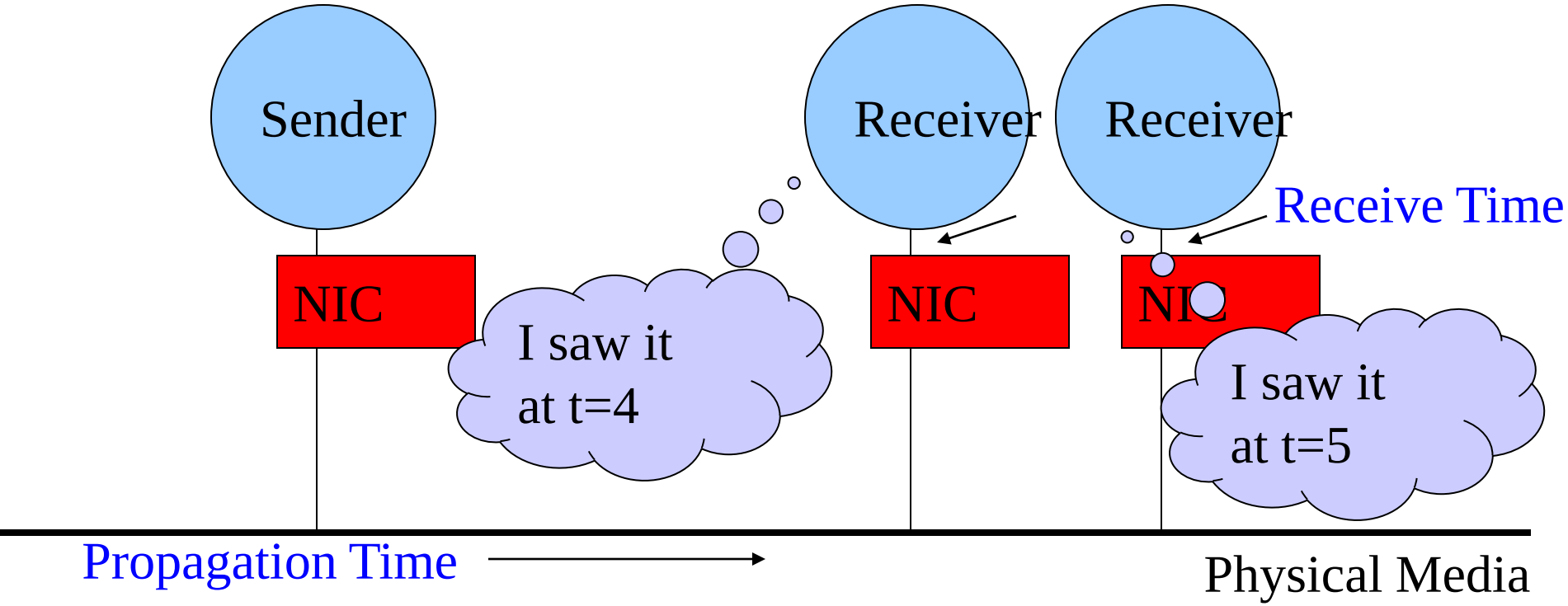
Problem: Many sources of unknown, nondeterministic latency between timestamp and its reception



Ref: based on slides by J. Elson

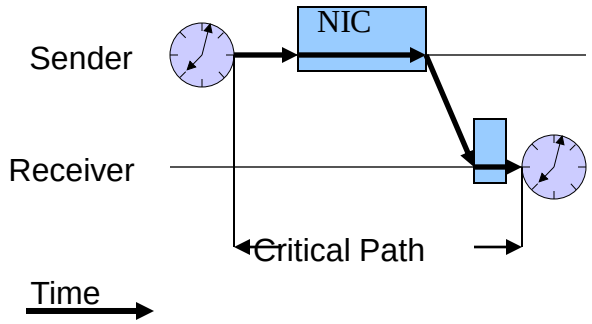
Reference Broadcast Sync

Sync 2 receivers *with each other*, NOT sender with receiver

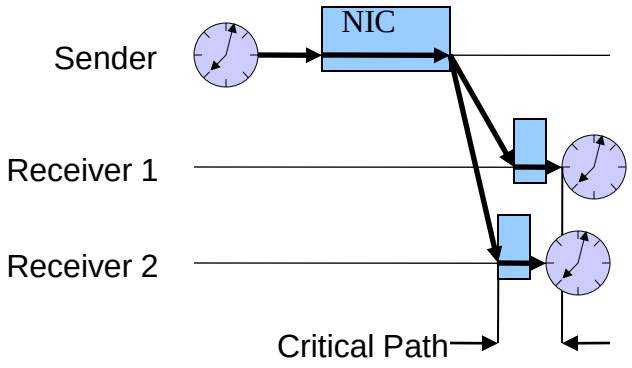


Ref: based on slides by J. Elson

RBS reduces error by removing much of it from the critical path



Traditional critical path: From the time the sender reads its clock, to when the receiver reads its clock



RBS: Only sensitive to the differences in receive time and propagation delay

Ref: based on slides by J. Elson

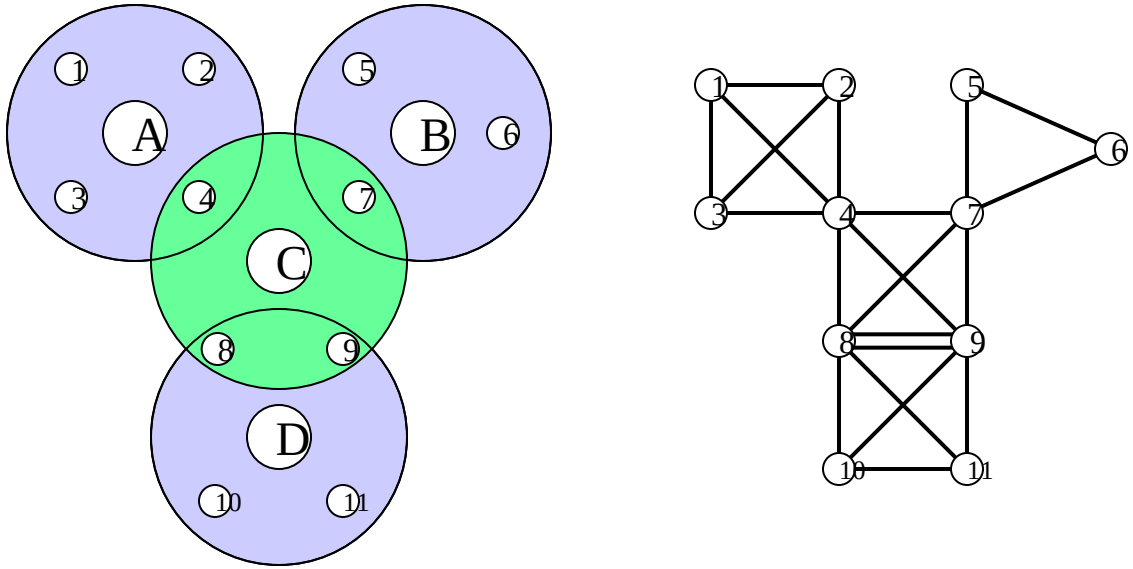
Observations about RBS

- RBS removes *send* and *access time* errors
- Broadcast is used as a *relative* time reference
- Each receiver synchronizing to a *reference packet*
  - Ref. packet was injected into the channel at the same instant *for all receivers*
- Message doesn't contain timestamp
  - Almost any broadcast packet can be used, e.g ARP, RTS/CTS, route discovery packets, etc

Ref: based on slides by J. Elson

Time Routing

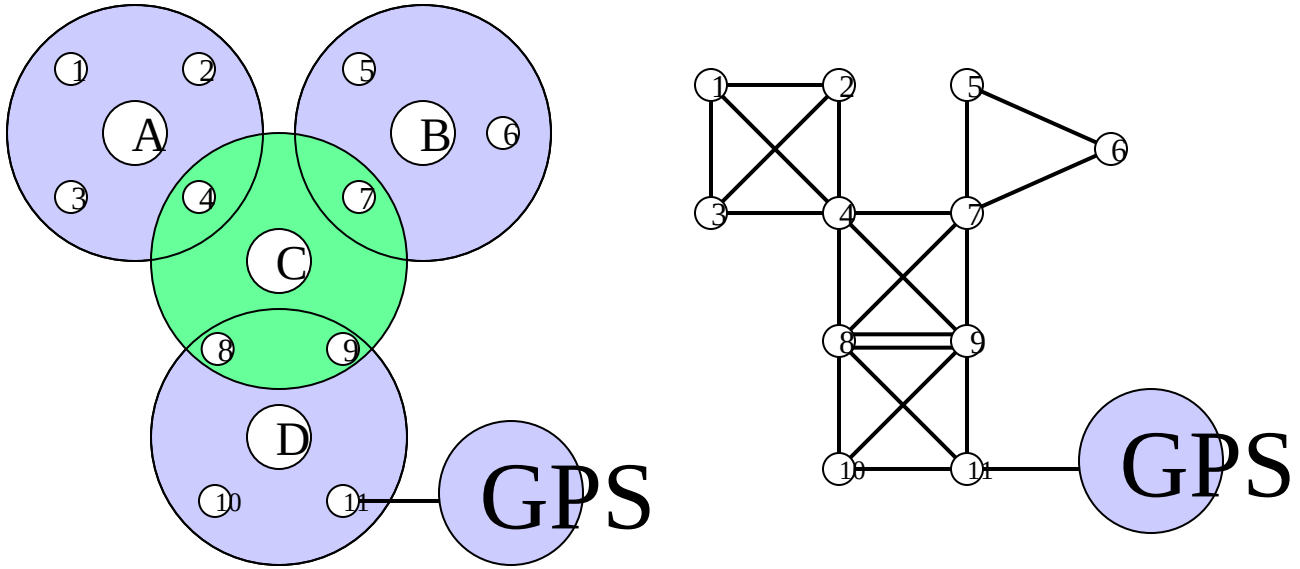
The physical topology can be easily converted to a logical topology; links represent possible clock conversions



Use shortest path search to find a “time route”; Edges can be weighted by error estimates

External Standards (UTC)

The multihop algorithm can also be easily used to sync an RBS domain to an external standard such as UTC



GPS's PPS generates a series of "fake broadcasts": "received" by node 11's local clock and UTC

Overview

- Protocols based on receiver/receiver synchronization
- Protocols based on sender/receiver synchronization
- Summary

Time-sync Protocol for Sensor Networks (TSPN)

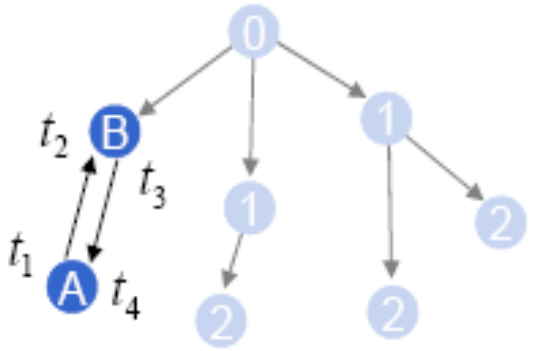
- Traditional sender-receiver synchronization (RTT-based)
- *Initialization phase: Breadth-first-search flooding*
  - Root node at level 0 sends out a *level discovery* packet
  - Receiving nodes which have not yet an assigned level set their level to +1 and start a random timer
  - After the timer is expired, a new level discovery packet will be sent
- *Synchronization phase*
  - Root node issues a *time sync* packet which triggers a random timer at all level 1 nodes
  - After the timer is expired, the node asks its parent for synchronization using a *synchronization pulse*
  - The parent node answers with an *acknowledgement*
  - Thus, the requesting node knows the round trip time and can calculate its clock offset
  - Child nodes receiving a synchronization pulse also start a random timer themselves to trigger their own synchronization

Time-sync Protocol for Sensor Networks (TSPN)

$$t_2 = t_1 + S_A + A_A + P_{A,B} + R_B$$

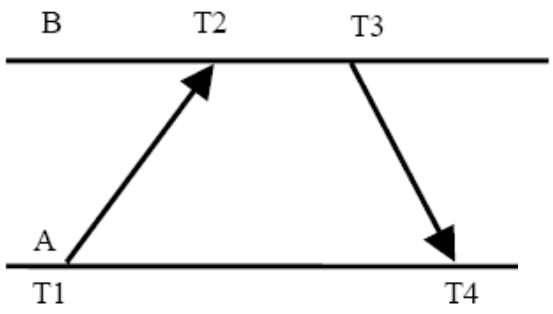
$$t_4 = t_3 + S_B + A_B + P_{B,A} + R_A$$

$$\theta = \frac{(S_A - S_B) + (A_A - A_B) + (P_{A,B} - P_{B,A}) + (R_B - R_A)}{2}$$



- Time stamping packets at the MAC layer
- In contrast to RBS, the signal propagation time might be negligible
- About “two times” better than RBS
- Again, clock drifts are taken into account using periodical synchronization messages

TPSN Error Analysis



$$t2 = t1 + S_A + P_{A->B} + R_B \ldots (2)$$

$$T2 = T1 + S_A + P_{A->B} + R_B + D_{t1}^{A->B} \ldots (3)$$

$$T4 = T3 + S_B + P_{B->A} + R_A - D_{t4}^{A->B} \ldots (4)$$

Note $D_{t3}^{B->A} \approx D_{t4}^{B->A} = -D_{t4}^{A->B}$. Further, $D_{t1}^{A->B}$ can be broken into two components as follows:

$$D_{t1}^{A->B} = D_{t4}^{A->B} + RD_{t1->t4}^{A->B} \ldots (5)$$

Here $RD_{t1->t4}^{A->B}$ refers to the relative drift between the nodes

$$(2 * \Delta) = S^{UC} + P^{UC} + R^{UC} + RD_{t1->t4}^{A->B} + (2 * D_{t4}^{A->B}) \ldots (6)$$

Here S^{UC}, R^{UC} and P^{UC} stand for the uncertainty at sender, at receiver and in propagation time respectively. They are given by the following equations:

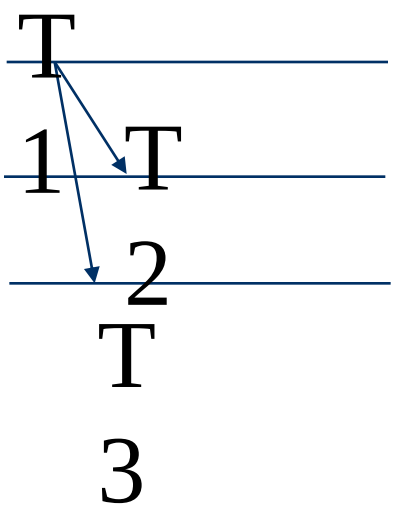
$$S^{UC} = S_A - S_B \ldots (7)$$

$$R^{UC} = R_B - R_A \ldots (8)$$

$$P^{UC} = P_{A->B} - P_{B->A} \ldots (9)$$

$$Error = \Delta - D_{t4}^{A->B} = \frac{S^{UC}}{2} + \frac{P^{UC}}{2} + \frac{R^{UC}}{2} + \frac{RD_{t1->t4}^{A->B}}{2} \ldots (10)$$

RBS Error Analysis



$$T2 = T1 + S_C + P_{C->A} + R_A + D_{t1}^{C->A} \ldots (11)$$

$$T3 = T1 + S_C + P_{C->B} + R_B + D_{t1}^{C->B} \ldots (12)$$

$$\Delta = (P_{C->B} - P_{C->A}) + (R_B - R_A) + (D_{t1}^{C->B} - D_{t1}^{C->A}) \ldots (13)$$

$$D_{t1}^{C->B} - D_{t1}^{C->A} = D_{t1}^{A->B} = D_{t4}^{A->B} + RD_{t1->t4}^{A->B} \ldots (14)$$

$$Error = \Delta - D_{t4}^{A->B} = P_D^{UC} + R^{UC} + RD_{t1->t4}^{A->B} \ldots (15)$$

Here P_D^{UC} represents the uncertainty in propagation time between two distinct node pairs and is given by:

$$P_D^{UC} = P_{C->B} - P_{C->A} \ldots (16)$$

Flooding Time Sync Protocol (FTSP)

- Implemented on Mica platform
- ~1 Microsec accuracy
- MAC-layer timestamp
- Skew compensation with linear regression (accounts for drift)
- Periodic flooding – robust to failures and topology changes
- Handles large scale networks

Mica2 experiment parameters

Table 1. The sources of delays in message transmissions

Time	Magnitude	Distribution
Send and Receive	0 – 100 ms	nondeterministic, depends on the processor load
Access	10 – 500 ms	nondeterministic, depends on the channel contention
Transmission / Reception	10 – 20 ms	deterministic, depends on message length
Propagation	< 1μs for distances up to 300 meters	deterministic, depends on the distance between sender and receiver
Interrupt Handling	< 5μs in most cases, but can be as high as 30μs	nondeterministic, depends on interrupts being disabled
Encoding plus Decoding	100 – 200μs, < 2μs variance	deterministic, depends on radio chipset and settings
Byte Alignment	0 – 400μs	deterministic, can be calculated

Remove Uncertainties

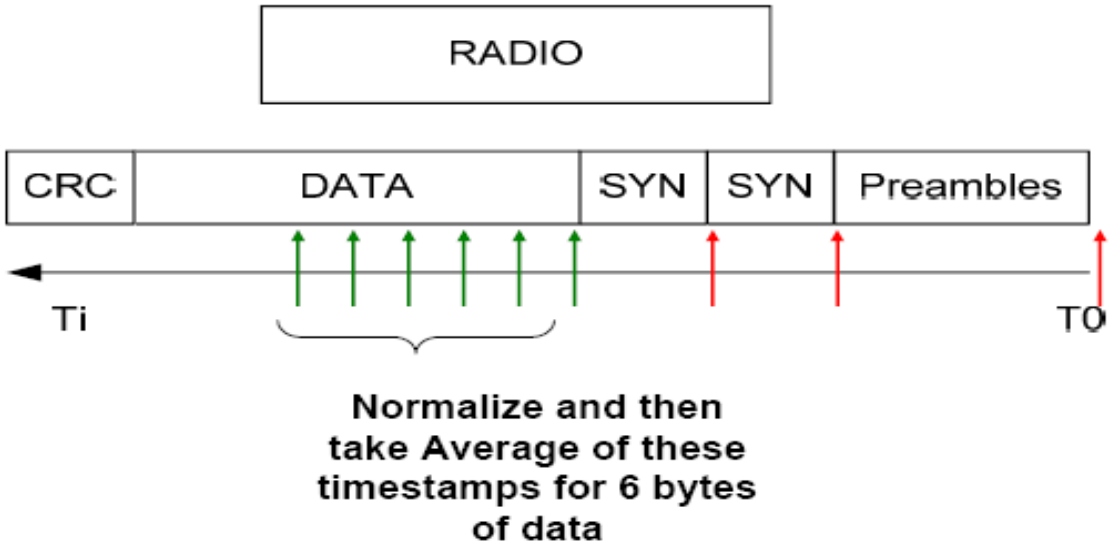
- Eliminate Send Uncertainty
  - Get time in the MAC layer
- Eliminate Access Time
  - Get time after the message has access to the channel
- Eliminate Receive Time
  - Record local time message received at the MAC layer

Time Stamping

Reduce the jitter of the interrupt handling and encoding/ decoding times by recording multiple time stamps both on the sender and receiver sides

When to time stamp the message

– Radio layer, after the second SYN sent out, 6 timestamps in row, take the average and send only 1 timestamp



Flooding Time Sync Protocol (FTSP)

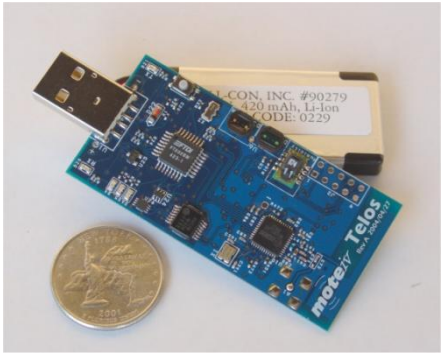
- Root maintains global time for system
- All others sync to the root
- Nodes form an ad hoc structure rather than a spanning tree
- Root broadcasts a timestamp for the transmission time of a certain byte
- Every receiver time stamps reception of that byte
- Account for deterministic times
- Differences are the clock offsets

Flooding Time Sync Protocol (FTSP)

- (Below) MAC-layer timestamp
- Correct sender timestamp to account delays
  - Compute final offset error
- Result: 1.48 µsec accuracy for 1 hop
- Available in TinyOS

Telos Platform

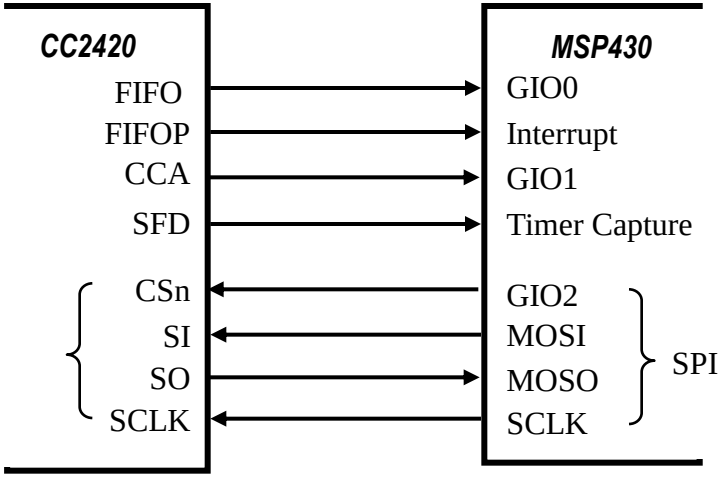
Telos wireless platform
(revision A)

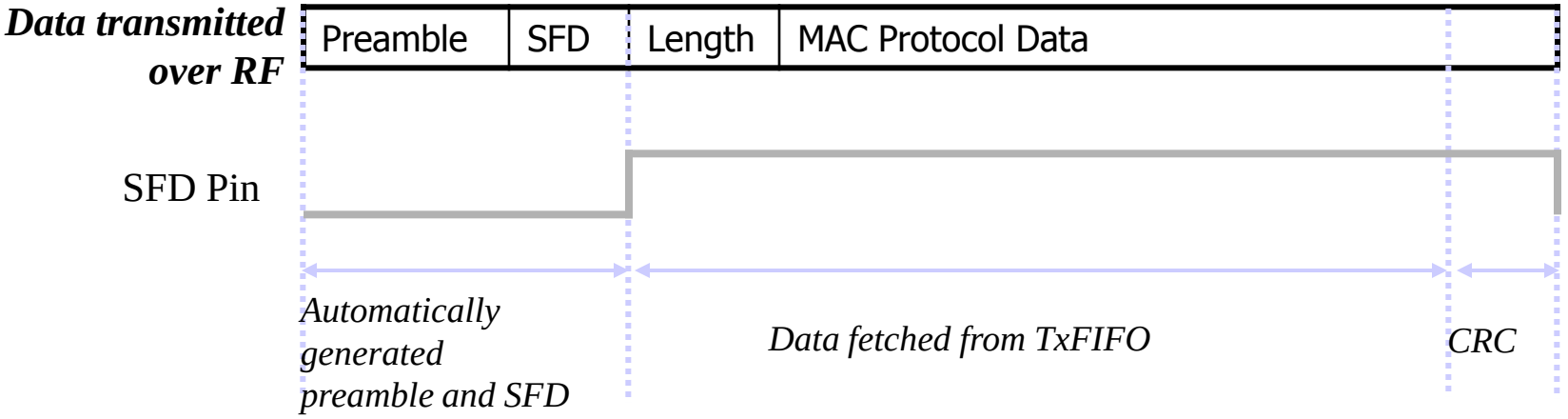


- Texas Instruments 16-bit MSP430F149 microcontroller (2KB RAM, 60KB ROM)
- Chipcon 2420, 250kbps, 2.4GHz, IEEE 802.15.4 compliant wireless transceiver with programmable output power
- Integrated onboard antenna with 50m range indoors and 125m range outdoors
- Integrated humidity, temperature, and light sensors

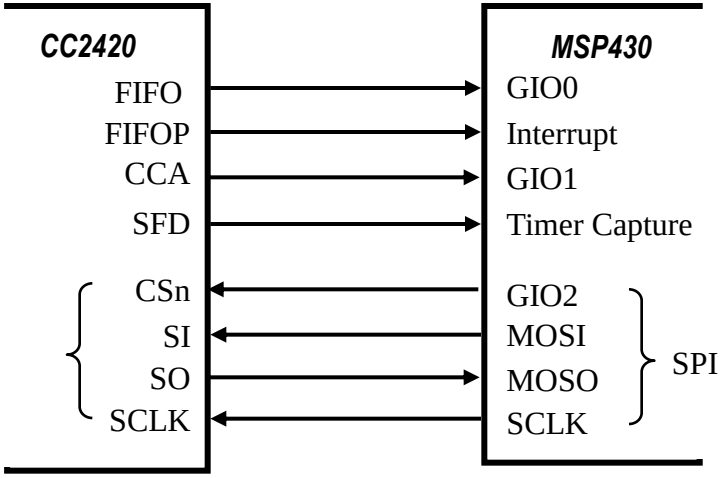
http://www.ece.uah.edu/~milenka/docs/am_ssst05_synch.ppt
http://www.isis.vanderbilt.edu/projects/nest/people/brano/pubs/poster_timesync_TTX2.ppt

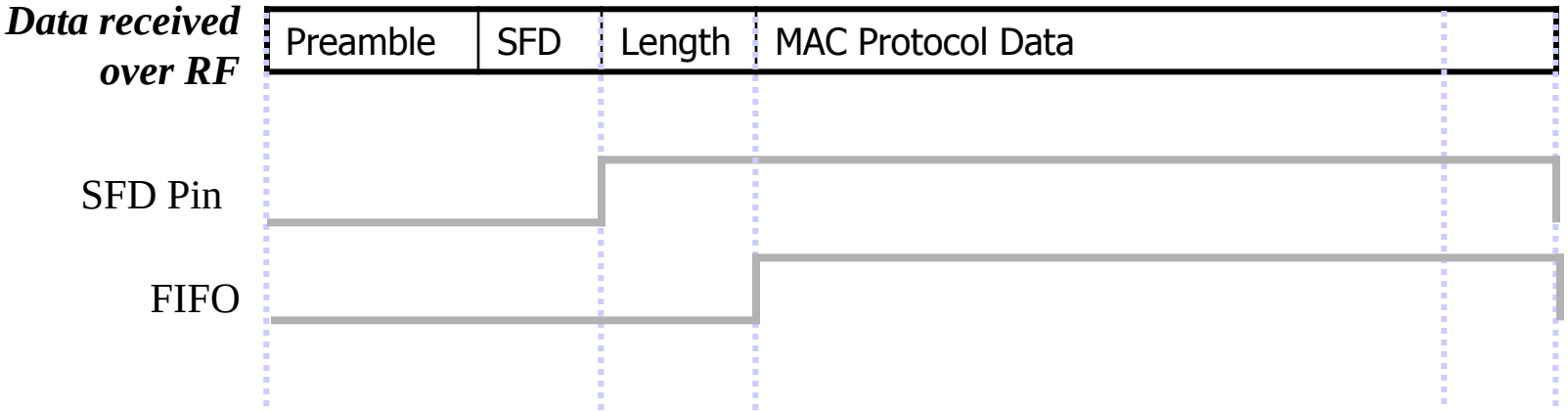
Transmit Mode



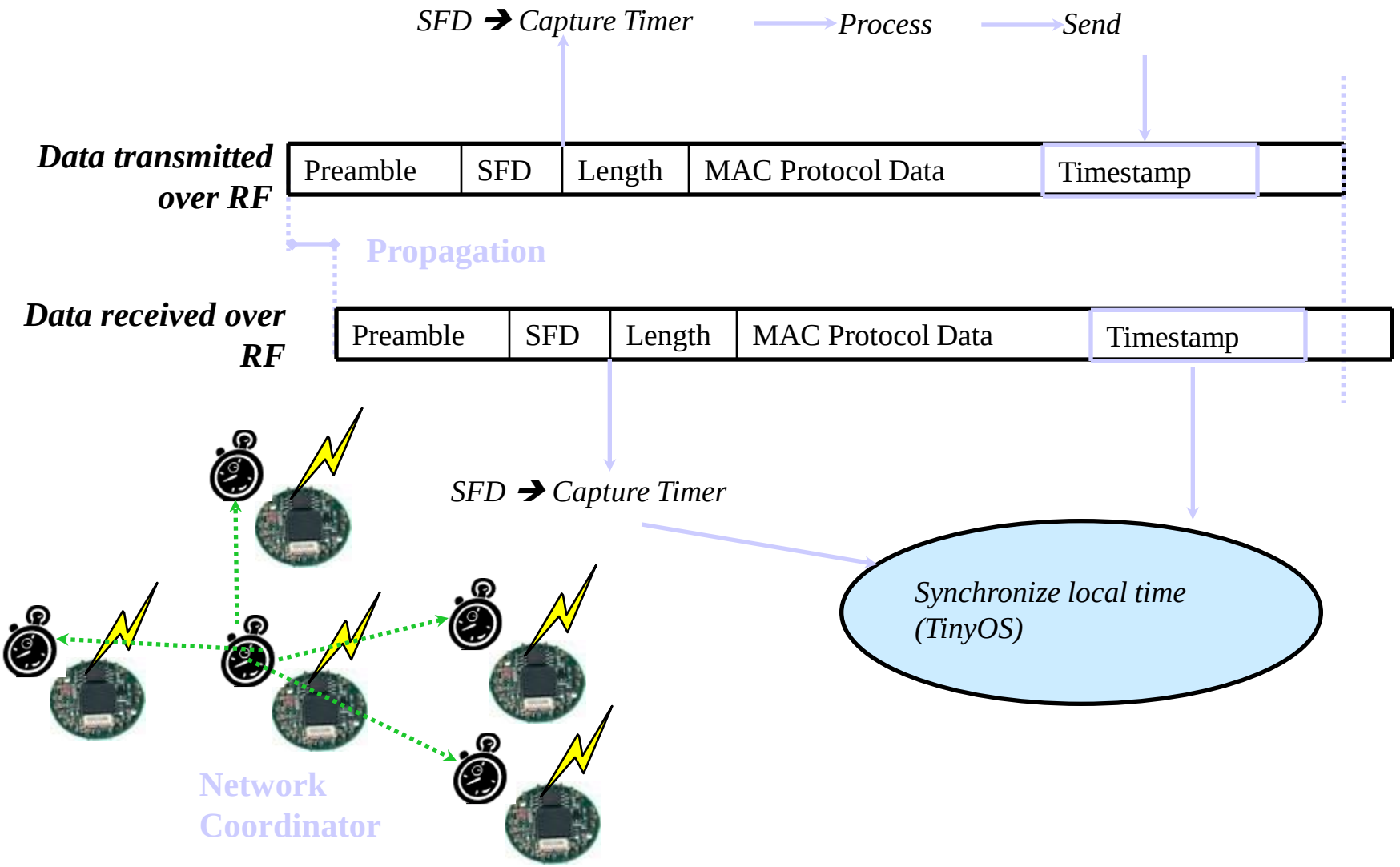


Receive Mode



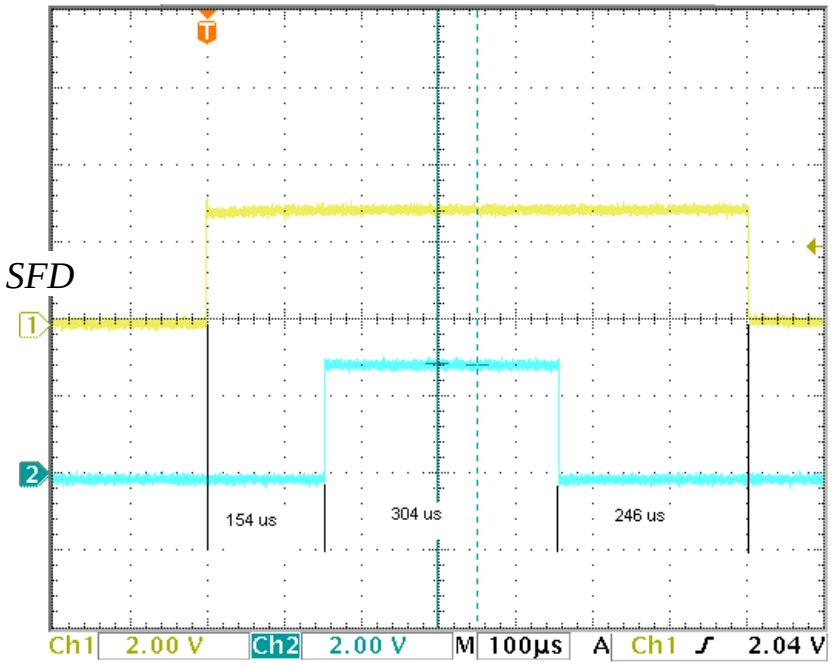


Mechanism for Time Synchronization



Inserting the Timestamp

- Network coordinator
  - Starts the transmission (time sync header)
  - Captures timer and converts to a global timestamp
  - Inserts it into the message (sends over SPI)
- Is this enough time not to underrun the TxFIFO in CC2420?
  - Time capture and calculate timestamp: ≈ 150 μs
  - Send timestamp: ≈ 300 μs
  - Sync message transmission: ≈ 700 μs

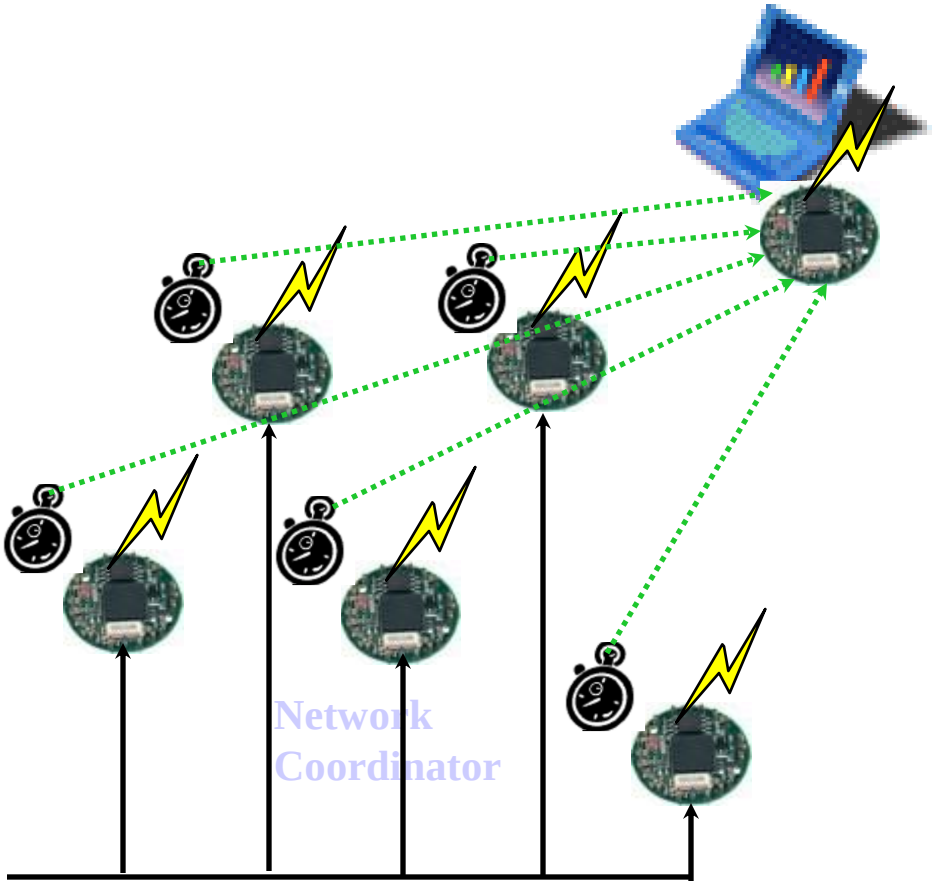


TinyOS Extensions

- nesC interface
  - Get current global time
  - Calculate how long until the next sync message
    - Useful to put to motes to sleep mode
  - Convert a local time to the global time
- Timestamps are based on 32768 Hz crystal
  - Stable, but slow (limit the resolution)
- MSP430 can run up to 8MHz
  - Internal DCO (Digitally Controlled Oscillator)
  - Poor stability

Testing Environment

- Master node + slave nodes connected to a common signal
- Synchronize the network
- Nodes report the global timestamp every time the common signal changes its state
- Compare the global time, reported from the master, versus global times reported from slaves



Results

Scenario	*A*	*B*	*C*	*D*
Sync message frequency (sec)	2	10	30	30
Total duration (min)	2	2	2	120
Average error (ticks)	0.49	0.61	0.81	0.67
Std. Deviations(ticks)	0.56	0.53	0.48	0.49

Overview

- Protocols based on receiver/receiver synchronization
- Protocols based on sender/receiver synchronization
- Summary

Summary

- Time synchronization is important for both WSN applications and protocols
- Using hardware like GPS receivers is typically not an option, so extra protocols are needed
- Post-facto synchronization allows for time-synchronization on demand, otherwise clock drifts would require frequent re-synchronization and thus a constant energy drain
- Some of the presented protocols take significant advantage of WSN peculiarities like:
  - small propagation delays
  - the ability to influence the node firmware to timestamp outgoing packets late, incoming packets early
- Of course, there are many, many more schemes ....