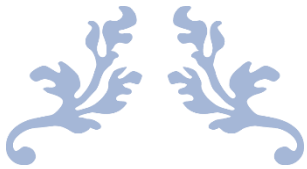


WIRELESS NETWORKS FOR MOBILE APPLICATIONS SIMPLE (FOR REAL)

Gabriel Rovesti

1 SUMMARY

Disclaimer

This file does not pretend to be correct but tries to provide a helpful comprehensive material to complete summarize the course slides (which the professor never had the enthusiasm to properly teach, unlistenable in my honest opinion, many times reading the slides and remembering some things to say at the moment). Much work was done here in order to maintain and give some proper context to all the topics the course had to offer, which is quite a lot – here, if possible, you will find everything explained thoroughly and in detail).

The classes were definitely not useful, at least I would say so: the professor has a lor of knowledge, he improvises given he knows a lot of things and it shows, but from a learner point of view, this is definitely not effective. These notes are shared in order to give complete notes and only listen to him. Otherwise, he runs so fast you can only write or listen. I decided to write instead, given he adds no value to any slide as said here (at least when you are in class; recordings are useful, sometimes more context is added, and I suggest that particularly for images. See for yourself and you will see what you need.

As other files of mine both in Italian and English, this summarized pretty much the entire course, considering projects discussions, seminars, case studies and main material itself. I wouldn't say this is a good reference, this is a thing you will find for yourself, if you want to of course: anyway, it's a lot of work. Consider also for each chapter I also added in name the set of slides it refers to, avoiding if possible usage of slides (here, everything useful is covered, just check for yourself if you don't trust) or any other material, other than the one you feel most confident in using of course.

Some notes here, sentences and images, are taken from present files on MEGA (such as "wnma-notes.1.0.pdf" and "WirelessNetworkNotes – Riassunto Completo.pdf".) This was made to create, here and there in little places, some simpler explanations considering the work already present and possibly integrate the present one in this.

I tried to integrate what they did very well, particularly from the first file: summarize precisely and concisely each concept, giving the idea and many times giving more and more details if pertinent, just to make you immediately understand everything even if you have no background to this. I hope to have made something out of this. Division in sections and subsections is personal and I think it's fairly clear when you use it for your exam.

I truly hope this can be useful in a subject really complex on the number of topics but also really vast and open to future developments.

Consider the order of the chapters follows exactly this year (2023-2024) topics order of presentation and lessons alike. In any other case, feel free to reach me to give some feedback over its content or to ask me things about it (even to thank me, does not kill me that much).

2 PRESENTATION (WNMA00)

A link find material of reference (will not be updated overtime though):
https://www.math.unipd.it/~cpalazzi/WNMA.html

Everything else is on Moodle. The program is as follows:

- Introduction, wireless systems, protocols architecture, issues and measures
  - Physical Layer (fundamentals and mobility effects)
  - Data Link Layer (fundamentals on duplexing, TDMA, FDMA, CDMA)
  - Network Layer (addressing/routing with device mobility)
  - Transport Layer (Reliable communication and mobility impact on TCP)
  - Application Layer (Geolocalized services, DTN, smart applications, distributed sensing, crowd computing, intelligent transportation system,…)

- Wireless Network Architectures: management and challenges
  - WLAN, Infrastructure and Hot-Spot Networks
  - Wireless Mesh Networks (WMN)
  - Sensor Networks (Sensor Networks)
  - Mobile Ad Hoc Networks (MANETs)
  - Vehicular Ad-Hoc Networks (VANETs)
  - Flying (Drone) Ad-Hoc Networks (FANETs)
  - Satellite systems, challenged networks

- Consumer market technology; main standards; advanced issues:
  - IEEE 802.11b/g/a/e/n/s/p
  - IEEE 802.15.1 (Bluetooth)
  - IEEE 802.15.4 (ZigBee)
  - RFID

- Services:
  - Location-based services
  - Client/Server and alternative service paradigms
  - Wireless Internet
  - Pervasive wireless communication systems
  - Other fields where Wireless Networks apply: existing and visionary services

The project can be on whatever scenario for the exam, will be any kind of project.

- Practical implementation or study of course-related scenarios
  - Performance evaluation of protocols in wireless scenarios
  - Development of applications for mobile environments (e.g., videogames or other applications for smartphones)

The specs for the project are as follows:

- Project development
  - To be decided with the Professor
  - Project (simulation, implementation testing)
    - Paper presentation + oral discussion
  - Survey (topics to be determined)
    - Slide presentation + Paper + oral discussion
- Oral discussion on class material (slides, etc.)
  - Material may very depending on the project
- TOTAL CREDITS: 6 (4 + 2)
  - Availability of more challenging projects as first step of a Thesis

4 pages are enough, but students find them too short for the report for the project. We will discuss with the professor, and we both must agree on that (via a meeting or I don't know). It may be much related to Networking but also something completely different. One can create a big project to satisfy both Mobile Programming and Multimedia and this course; this is also a chance.

The project has this kind of evaluation criteria:

Project evaluation
- Difficulty
- Results
- Autonomy
- Possible topics:
  - Critical analysis of the state of the art, new solutions for wireless issues (protocols, algorithms), verification through simulations or real experiments, wireless systems implementation (e.g., games on mobile phones), mobile applications, sensing, drones…
  - Possible shared projects with other classes (e.g., Mobile Programming and Multimedia)

This class covers:

- Design, analysis, and implementation of protocols and algorithms in (mobile) wireless network systems and their implication in the design of popular/innovative mobile applications

But does not cover:

- Modulation schemes, transmitter/Receiver design, signal processing and antenna design, source coding / channel coding, privacy / Security

Other things:

- The project can be done in pairs (strongly suggested, 2/3 people per team)
- The exam is oral, not written. The project can be delayed and decided to a term even after the examination (you can do one, the other or together, according to your needs). The teacher is flexible on this.

We are supposed to read papers to explore and further dive on a particular topic, to absorb concepts, for ourselves and to rationalize different ideas and different solutions. Via the intranet of UniPD inside the Department, we can freely access our specific papers.

Written by Gabriel R.

General criteria on how to write the papers:

- Projects/papers consist of 4 parts:
  - Problem identification
  - State of the art discussion
  - Solution design
  - Performance evaluation

- Each paper you read is someone's project
  - Many papers have actually emerged from class projects
  - Read them critically
  - Ask yourself
    - Is the problem really important? Should you care?
    - Is the solution sound? Under what assumptions?
    - Do you have other (better) ideas?
    - Is evaluation biased? Are results shown only in good light?

It's important to discuss ideas and thoughts with the professor, mainly on an area and direction and find new solutions for more problems. Also:

Protocol evaluation typically requires coding
- Think what you would like to do
- Options are:
  - Coding on real devices (sensors, smartphones, routers)
  - Coding in existing network simulators (ns2, ns3, Qualnet, etc.)
  - Coding your own simulator
  - Theoretical projects involve MATLAB, CPLEX, etc.

Project ideas take time … think now and then
- Spending 3 hours for 10 days better than 10 hours for 3 days

- Choose a topic of interest
- Find related scientific bibiliography
  - scholar.google.com
  - http://ieeexplore.ieee.org/
  - http://dl.acm.org/

- Prepare a 20min presentation and discussion on the topic
  - Read about (depending on their size and of the number of team members) 5 related papers critically
  - For each paper ask yourself
    - What is the problem? Is it really important? Why?
    - Is the solution sound? Under what assumptions? What kind of experiments/analysis were performed?
    - Are there common classes of solutions?
    - Do you have other (better) ideas?
    - Is evaluation biased? Are results shown only in good light?

- These projects *should* (if possible) be presented during class hours

3 WIRELESS DEVELOPMENT AND WIRELESS SYSTEMS (WNMA01)

Present wireless networks are exploding in popularity, given the huge number of devices constantly connected and the growing demand of data rate, which constantly open new scenarios and demands for more. This implied an important growth of Wi-Fi and different generations (G – 3G, 4G, 5G and even 6G in the future) technologies also thanks to the emerging of apps with both low and high data demand.

Wireless technology is now integrated into interdisciplinary applications:

- constantly modifying and uploading new data from users (via social for instance)
  - *Web 2.0* (users can write the website contents by themselfs, e.g. Facebook pages)
- opening *multiple realities scenarios*
  - augmented-AR/virtual-VR/mixed-MR/tele-presence
- ultra responsive at our touch in a highly reliable/available way with ultra-low latency
  - *Tactile Internet*
- intraconnected in realtime with wide range of data demands and devices
  - between Web 2.0 and sensing technologies, collecting data on users intelligently
  - *Web Squared* (collecting data to create the service, e.g. Google Maps that collects data via sensors to create the route)

Future wireless networks will be more and more based upon *ubiquitous communication among people and devices*, allowing seamless wireless access and cellular data everywhere and even smart data/networks. Here, we should consider different *constraints* such as *bandwidth, delay, energy and connectivity*.

In designing networks, we face many challenges:

- *Wireless channels* are a *difficult* and *capacity-limited*
  - with respect to the wired counterpart, which will always be better
  - interference, errors, delays, nominal speed and all factors to consider
- Wireless networks *are very hard to plan*
  - Traffic patterns, user locations, and *network conditions are constantly changing*
  - Nodes position can change
    - often do not have patterns which can be exploited heterogeneously
- Applications are *heterogeneous* with *hard constraints* that must be met by the network
- Energy and delay constraints *change design principles* across all layers of the protocol stack
  - In a shared channels with rules, usually hard to implement, regulating traffic leads to delays
  - Consider the battery requirements in the mobile case especially, limited

There are applications that don't need to read data immediately and other applications that are more interactive => we need to deal with that

Below the general multimedia requirements (where BER stands for Bit Error Rate):

	Voice	Data	Video	Game
Delay	low	irrelevant	low	low
Packet Loss	low	no	low	low
Bit Error Rate	10^{-3}	10^{-6}	10^{-6}	10^{-3}
Data Rate	8-32 Kbps	1-100 Mbps	1-20 Mbps	32-100 Kbps
Traffic	Continuous	Bursty	Continuous	Continuous

From this, consider:

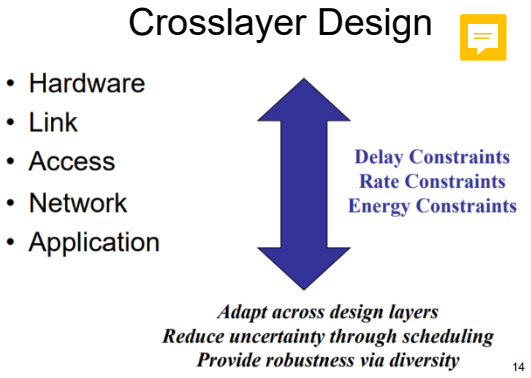
- *Delay*: amount of time required to push all packet's bits
  - it has to be low for real time applications (video/voice/games)
    - in case of data the main thing is just to deliver it
- *Packet Loss*: loss of some data in a network, caused by various problems
  - here, we don't tolerate loss in data, in other means it can be definitely more bearable
- *Bit Error Rate* (BER): number of bit errors per unit time
  - calculated dividing the quantity of bits received in error by total bits transmitted
  - we assume that, when we lose a bit, we lose the whole packet
  - Packet Error Rate is different: considers the integrity of entire packet rather than bits
- *Data Rate*: how much bandwidth is consumed and how fast they are transmitted
- *Type of Traffic*: a device transmitting an uninterrupted stream of data
  - in case of data, we are interested only in the total time of stream (bursty)
  - we would like to have as less packet loss as possible (to have it continuous)

In general, we consider that:

- *One-size-fits-all protocols and design do not work well*
  - because <u>each media has specific requirements</u>
- Wired networks use this approach, with poor results

In computer networks design, we have layers, and dealing with them may be difficult if we want to keep them separated. We base ourselves on the <u>crosslayer design</u>**,** where different layers blur between each other, leading to better performances overall via abstraction and better communication. This way, information goes from one layer to others too.

It's advisable to focus "on your layer" when creating a network application, without having something universal but functional (without sacrificing exploitation of more possibilities).



What we essentially mean is creating something carefully, having each layer naturally interact with others, without "crossing borders" unsafely, but *propagated*. This design is therefore commonly used in networks.

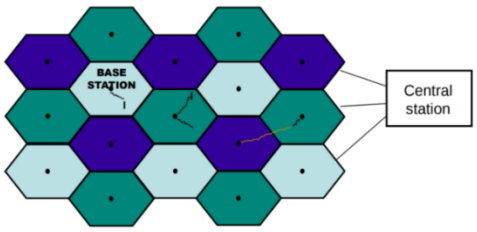
3.1 Current Wireless Systems

Let's introduce the *current wireless systems*, ranking the ones most used.

3.1.1 Cellular Systems

They have geographic regions which are divided into *cells*:

- Each one has their own signal (different colors for different frequencies to avoid interferences)
- Carefully crafted areas allow the spatial reuse of frequencies/timeslots/codes in between:
  - at the center of each one, there is an *antenna* (also called *base station*)
    - coordinating hand-offs and control functions via base stations
      - *horizontal* handoff = *same* technology
      - *vertical* handoff = *different* technology
    - more transmission distance = more power
  - there can be co-channel interference between same color cells
- Shrinking increases capacity but also relaxes networking burden
  - larger cells = more people and more bandwidth used
    - if too many, you won't be able to use your device
- Data is bursty, whereas voice is continuous
  - Typically require different access and routing strategies
- 3G "widens the data pipe"
  - $384\ Kbps$ (now even $1.6\ Mbps$ – $3.2\ Mbps$ – $7.2\ Mbps$)
  - Packet-based switching for both voice and data
- 4G/5G are more and more focused on data
  - High bandwidth/High reliability/Low latency



3.1.2 Wireless Local Area Networks (WLANs)

They connect "local" computers (between a $100\ m$ range), breaking data into packets (circa $1500\ B$) in even smaller size ($500\ B$). In them, channel access is shared (random access) and has to be coordinated properly (MAC layer).

- There is an Access Point, connected to the Internet through a wire and using 802.11
  - this backbone provides *best-effort service*
    - it prioritizes delivering data with no guarantees of quality or timeliness
    - bandwidth cannot be determined
    - if backbone is faster, it can become bottleneck
- In between there are APs (Access Points)
  - devices are connected wirelessly between them
  - they act as bridges and break data further in between
- This can lead to *overhead*
  - due to the need to retransmit smaller packets (given overhead is fixed)
  - splitting can be more forgiving
    - when facing a big error rate, we don't want to lose all packets
- Packets are then recomposed inside of the sender

Written by Gabriel R.

Just to give a quick overview of many 802.11 WLAN Standards:

- *b* (old gen): only $2.4\ GHz$, speed $1-11\ Mbps$, range $100\ m$ / frequency hop
  - it's free frequency, so it's crowded
- *g* (legacy std): $2.4-5\ GHz$, speed up to $54\ Mbps$ / frequency multiplexing
  - quite popular, using a higher bandwidth, bringing changes in the physical layer
- *n* (current gen): $2.4-5\ GHz$, speed up to $300\ Mbps$, multiple I/O (MIMO: more here)
  - a device is easily recognized because it uses multiple antennas/multiple channels (MIMO)
- *ac* (current/emerging gen): $2.4-5\ GHz$, speed up to $500\ Mbps$, multiple I/O
- *s:* used for mesh networks
- *p*: used for vehicular networks

Other ones to briefly quote (written for completeness, to give context to a further chapter dedicated):

- *a*: original standard, 2.4 GHz, 1/2 Mbps, both radio and infrared level
- *d*: extensions for international roaming
- *e*: extensions for Quality of Service (you will see more here)
- *f*: standard for Inter Access Point Protocol (IAPP)
  - communications among multivendor systems
- *h*: dynamic channels selection, control of transmissive power
- *i*: integrations and extensions for safety
- *j*: extensions for Japan directives
- *k*: extensions for measurements in radio parameters
- *r*: extensions for fast roaming
- *t*: methods and metrics to measure and predict performance
- *u*: extensions for non-802.11 networks (cellular)
- *v*: management and administration of wireless networks

3.1.3 Satellite Systems

- They are used to cover very large areas
- They are usually a two-way system, optimized for one-way transmission
- We differentiate them according to the different orbit heights:
  - GEO satellites (Geostationary Earth Orbit) stay at about 39000 km
    - They are so high that remain stationary and can cover the whole surface
  - LEO satellites (Low Earth Orbit) stay lower, at 2000 km
    - They revolve continuously in order to not fall
  - If you did the splendid Marchiori (WIM/Networks) courses, you know about MEO
- In particular, satellites:
  - Can cover large areas depending on their height in the space
    - $> height \Rightarrow > covered\ area, > latency, < bandwidth$
    - $< height \Rightarrow < covered\ area, < latency, > bandwidth$
  - They are optimized for one-way transmission, especially for radio and movie broadcasting
    - usually satellite → earth is faster than the other way
    - a delay of between $25/50\ ms$ for the latter ones/LEOs to $120\ ms$ for GEOs
  - Most two-way systems like Iridium failed because of expenses and delays
    - this is an expensive alternative, mostly experimented as alternative to terrestrial communications

Written by Gabriel R.

3.1.4 Bluetooth

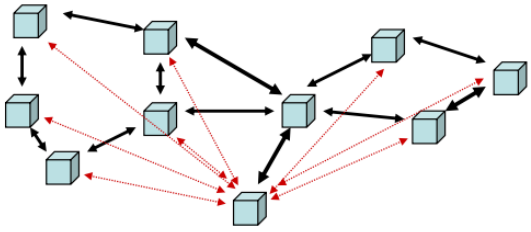
- Created as a low-cost cable replacement RF (radio frequency) technology
  - so this is why it's this short-range
  - at about $10\ m$, extendable in case to $100\ m$ through multihop
- This is a cheap and battery-light alternative to avoid consuming too much battery
- The signal strength decreases exponentially and not linearly with the distance
  - It requires exponential energy as distance grows
  - The band used is crowded ($2.4\ GHz$)
    - 4 channels
      - 1 data channel up to $700\ Kbps$
      - 3 voice channels
  - The short range is mainly used to avoid consuming too much battery
  - It's supported almost everywhere across devices nowadays
    - with very few applications beyond cable replacement
    - mainly used for device pairing but a standard de-facto

3.2 Emerging Wireless Systems

In this subsection, we will explain many kinds of new, emerging systems, giving an overview of them.

3.2.1 Ad-hoc Networks (ANETs)

They were initially developed for military purposes and are networks with no fixed infrastructure (no backbone/APs) and:



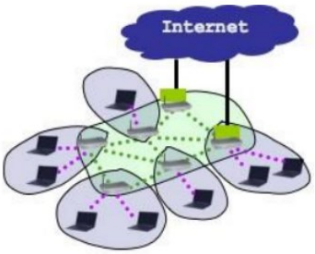
- They are very much flexible
  - nodes are *equal*
  - *based on peer-to-peer* (P2P) communications
- Routing is hard because of:
  - dynamic topology (unpredictable)
    - we never know the size and routing all at once
    - nodes keep moving/going away, so they reorganize automatically
  - capacity is unknown
    - if a node transmits, others are blocked to avoid interference
  - *multihopping*
    - extend coverage area to reach far away nodes from the sender
    - to reduce interferences
- Like the name says in Latin, "ad-hoc" literally "made for this"
  - made impromptu or improvised for a specific situation
- Here, each nodes participates in forwarding data for other nodes
  - both host and router, hence the bidirectional drawing above
  - there is no central node, useful when infrastructure deployment is hard
  - communications almost instantaneous between nodes

Ad-hoc networks provide a flexible network infrastructure for many emerging applications, but they can have many *design issues*:

- Transmission, access, and routing strategies for these networks are generally ad-hoc:
    - Per say, “made on-the-fly”
    - Routing can’t be univocally determined all at once, given its “movable” nature
        - we never know the size and routing of our network, both can change frequently
- Crosslayer design is critical for efficiency and very challenging
    - that may result in very low performance
    - be flexible and try to have secure routing, via encryption and authentication mechanisms
- Energy constraints impose interesting design tradeoffs for communication and networking
    - Choosing to increase transmission power to avoid multihopping leads to problems both in energy consumption and interference
        - if the distance doubles, the energy required is four time greater
        - it is possible to occupy the whole network with just one transmission
    - There needs to be a careful planning in order to “keep the service alive” (right QoS)
- More generally, the problems are: hops, bandwidth, collisions handling, energy consumption, topology dependency on device

3.2.2 Mesh Networks

It’s a type of networks between ad-hoc networks and wireless ones and:

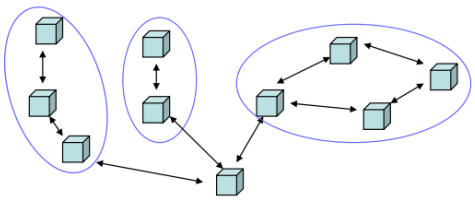


- They are ad-hoc *opportunistic* extensions of a fixed urban infrastructure
    - can connect to both regular (fully wired) and completely wireless access points
    - creating a low-cost, easily deployable and high-performance coverage
- They are easier than ANETs
    - because of their almost static topology
    - mobility is less frequent: otherwise, there is the need of constant updates
        - unless we have completely wireless access points
- Nodes not only communicate with their neighboring nodes but also serve as relays
    - this redundancy make mesh networks highly robust and resilient

There are many challenges however:

- Optimum routing protocols need to be chosen in order to achieve fairness and load balancing
- It is difficult to maintain a stable Quality of Service (QoS)
    - stating a quality standard and being able to guarantee its parameters
- There need to be efficient autonomous operations when the infrastructure fails
    - giving an automatic setup in case of infrastructure’s failures
- Multimedia applications demand specialized attention in ad-hoc networks
    - given their real-time and bandwidth-intensive nature
    - MAC protocols must be designed to support efficient flows of multimedia streams
        - handling different network conditions

3.2.3 Sensor Networks



It is a specific type of ad-hoc networks composed by *sensors*, where each one is used to monitor specific things and:

- there is at least one sensor as device in the network
- the main constraint in this network is *energy*
    - given nodes have a limited amount and are powered by non-rechargeable batteries
- data flows to centralized locations (to analyze/process/send data further away)
    - nodes may help other sensors to send messages
        - a good synchronization is needed
        - considering these networks are very large (up to 100000 nodes)
    - their data, usually, it's very simple: temperature values, pollution/water levels, etc.
        - there is a low-per-node rate
- nodes can cooperate in transmission, reception compression, and signal processing
    - more transmission means more bandwidth consumed
    - data must be highly correlated in time and space
        - aggregating should be mitigated properly to avoid losing portions of data

3.2.4 Distributed Control Over Wireless Links

There are networks dealing with distributed control over wireless links: you need to have a *good enough* capability to transmit control signals and data between. This usually automatic vehicles: cars, Unmanned Airborne Vehicles (UAVs), insect fliers and so on. A few key points:

- *Packet loss and variable transmission delays* are *common*
    - due to factors like interference and network congestion
    - this *can have a significant impact* on the controller performance
        - component who orchestrates the network state
- To address the challenges, *controller design must be robust to network faults*
    - avoiding queues, packet losses and delays
    - also assuring a good bandwidth and speed when delivering messages
- Distributed control systems should be designed to tolerate network faults gracefully
- Joint application and communication network design is crucial
    - to optimize both control performance and network efficiency

3.2.5 Mobile Ad-Hoc Networks (MANETs)

MANETs are self-configuring networks of mobile devices where nodes communicate with each other directly with an instantly deployable, re-configurable infrastructure created to satisfy a "temporary" need.

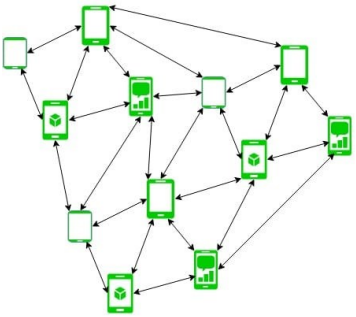
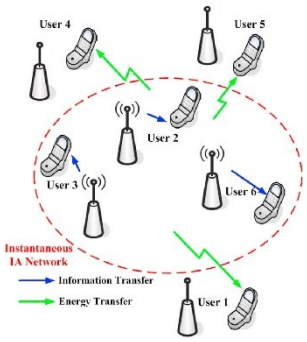


Figure - Mobile Ad Hoc Network

- The topology of the network continuously changes
  - the protocol used must adapt consequently
  - given each information can be obsolete at any time
- They are portable (e.g. sensors) or mobile (e.g. cars),
  - without a fixed infrastructure
  - crafting each according to needs
- They were used initially for military purposes
  - then extended to civilian tasks
    - disaster recovery, law enforcement, homeland defense, search/rescue in remote areas
    - environment monitoring (sensors), space/planet exploration
- They are no different from normal ANET, nodes may move a little more

3.2.6 Opportunistic Ad-Hoc Networks

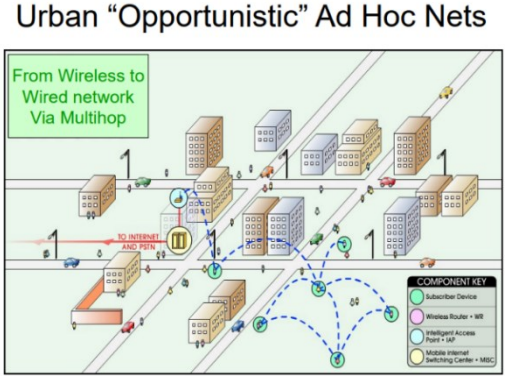
This is a type of wireless network which will "opportunistically" replace with an ad-hoc network the specific application needs when the available Internet connection access is too costly, inadequate, or temporarily unavailable (so, created when needed).



- These networks are often driven by commercial or practical needs
  - the thrive in scenarios where traditional infrastructure-based networks may be limited or expensive
- They are driven by application needs:
  - such as extending coverage indoor, sharing among friends
  - via Bluetooth, P2P networking in a vehicle grid
  - so, main feature is flexibility
- don't use homogeneous nodes (like in traditional ad-hoc networks)
  - instead, always try to use the best available

A use case of the previous are *Urban Opportunistic Ad-Hoc Networks*, where wireless connections may be intermittent due to obstacles like buildings and signal interference.

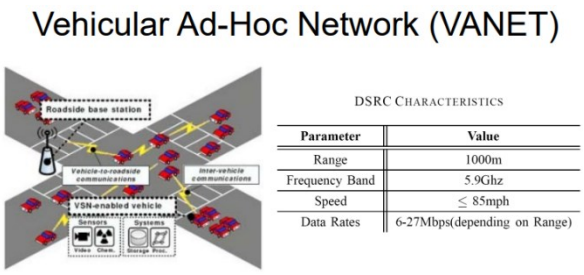
Opportunistic communication allows devices to make the most of brief connectivity windows, given they are deployed in densely populated urban areas, such as cities and metropolitan regions, adapting in routing and in communication.



Written by Gabriel R.

3.2.7 Vehicular Ad-Hoc Networks (VANETs)

Vehicular Ad-Hoc Networks (VANETs) are a specialized type of ad-hoc network designed to enable communication *among vehicles via on-board units* and between vehicles and roadside infrastructure in the context of transportation systems.



- This can be used for sharing information about road conditions, traffic incidents, vehicle safety
  - *in a predictable way, useful for crosslayers*
- In a controlled environment, monitoring and predictability are present
  - they also have a frequency reserved for vehicles communication (IEEE 802.11p)
- Since vehicles have engines, *reducing energy consumption is not a priority*
  - a group of vehicles on a road going in same direction move together with respect to each
- There are *other concerns* which arise, though:
  - *legal* problems: in case of accidents, whose fault is it?
    - usually, accidents happen when mixed with regular cars (SW malfunctioning/hijack)
  - *moral* decisions: if a car has to steer to avoid an accident but in doing so it hits a passer-by, what is the best decision?
- There are a lot of applications for these ones: public/safety and private applications

There will be two cases dedicated to these ones, so they are probably the most important for the exam:

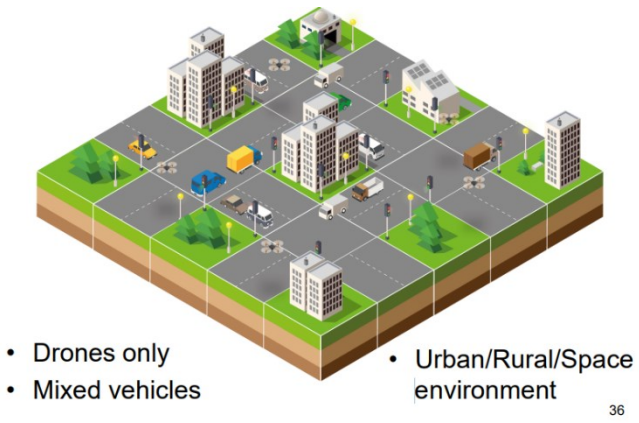
- a case study here
- an entire chapter here

3.2.8 Flying Ad-Hoc Networks (FANETs)

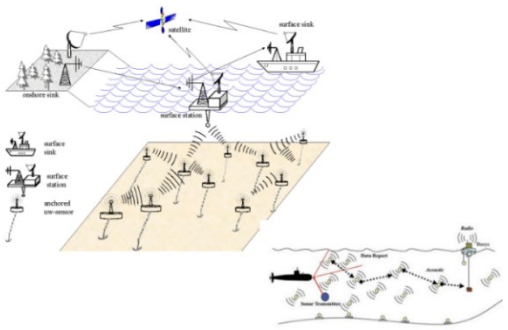
Another good example are the Flying Ad-Hoc Networks (FANETs), composed by drones only or mixed vehicles, usually inside a urban/rural/space environment.

They are designed for communication among autonomous flying vehicles, such as drones, Unmanned (without men/equipe) Aerial Vehicles (UAVs), or Remotely Piloted Aircraft Systems (RPAS).

Topology is highly dynamic and operate autonomously, so considering the 3D topology, protocols need to be carefully redesigned.



3.2.9 Underwater Sensor Networks



A usage to note about already talked about sensor networks are the Underwater Sensor Networks.

- In these, there is a good enough challenge: electromagnetic waves propagation in water
- It is possible to use *sound waves* (also ultrasound):
  - messages propagate in circles
  - but connection will be much slower
    - because sound is not fast like light
  - the big advantage in using them is the *chance for transmission at the same time*:
    - when far enough from each other, these will not collide
    - a collision happens if the receiver gets at the same time two different signals
      - not when two messages travel together
- Usually, those rely on battery
  - duty cycling, and energy harvesting are often employed to maximize battery life
    - prolonging battery span as long as possible

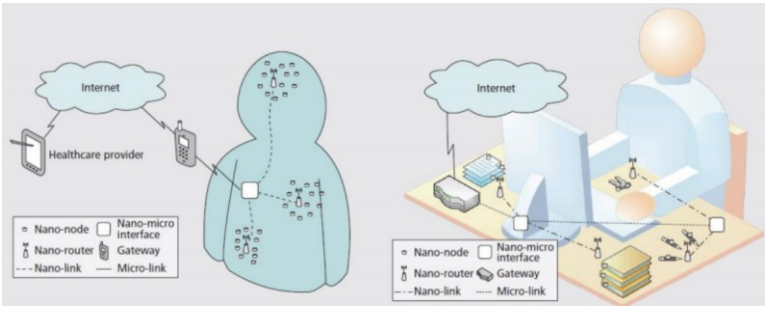
3.2.10 Radio Frequency Identification (RFID)

We also have Radio Frequency Identification (RFID), which is based on magnetic fields on *tags*, using *readers* (with high cost) and a *server* (like shown in figure).

- They store more info than simple barcodes and are usually *used inside supply chains*:
  - they allow to see preparation date, where it was created, who took care of it, etc.
  - identifying specific instances of products
- Usually, such tags can have:
  - no battery – the emitter charges the tag with energy
  - battery – the tag periodically emits its ID
    - check of product history, control with sensors, etc.
  - tags do not need to be within direct Line-of-Sight (LOS) of sender
- Systems can be built with:
  - lots of tags + one emitter – they are cheap
  - lots of emitters + one tag – they are expensive
- The charged tag sends a message containing all the information to the server that can then check it
  - they do not need a direct optical reading
  - but require an emitter of electro-magnetic waves that charge the tag
- They contain more information than simple barcodes, but they are more expensive
  - standards for this are currently under development

3.2.11 Nano-networks

We finally categorize the nano-networks, a type of communication network where extremely small devices or components, often at the *nanometer* scale (one billionth of a meter = $10^{-9}\ m$), communicate with each other to perform specific tasks.



- They rely on communication mechanisms that operate at the nanoscale
    - such as calcium signaling or molecular communication
    - which are explained in detail here
- They have potential applications in various fields
    - recent Nobel prizes were exactly studies about these kinds of networks (e.g. 2010 & 2023)

The main points of what was discussed up until now coming from slides:

- The wireless vision encompasses many exciting systems and applications
- Technical challenges transcend across all layers of the system design
- Wireless systems have limited performance and interoperability
- Standards and spectral allocation heavily impact the evolution of wireless technology
- Huge potential for future applications and systems

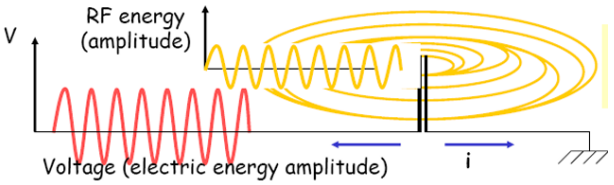
4 PHYSICAL LAYER (WNMA02 – WNMA03)

The physical layer provides an electrical, mechanical, and procedural interface to the transmission medium. In particular, converts into signals the bit to transmit to the channel in a suitable way according to the medium specific in usage. Here, an overview of signals and theory on those and antennas.

4.1 RADIO SIGNALS AND PROPERTIES

Energy inside RF (Radio Frequency Transmission) is based on electromagnetic energy generated by high frequency AC in *antennas*, which convert the wired current to RF and viceversa.

They can have different frequencies and lengths and propagate differently depending on the medium they need to cross. Most of the wireless technology is based on these signals.

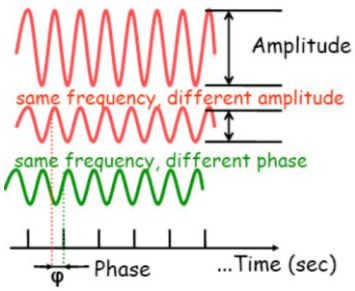


Usually, signals have three main properties to note (we care about these to create the right antennas lengths in terms of transmissions):

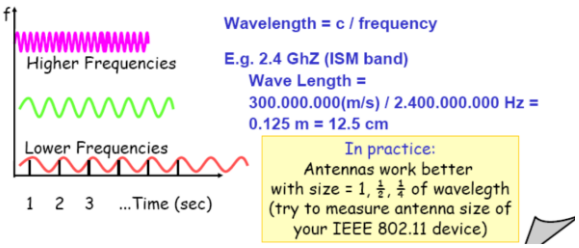
- Amplitude (m), which determines *how far a signal can travel*: the higher the amplitude, the further the RF signal goes. This is proportional to transmission energy (*loudness*). It's measured in:

$$Transmission\ Power\ (Watts) = \frac{Energy}{Time} = \frac{Joule}{Sec}$$

  - More energy (voltage) moves more electrons (current)
  - It's the *difference between the highest and lowest wave* peak
    - reaching $2x$ distance requires $4x$ power
  - $Power = Voltage * Current$

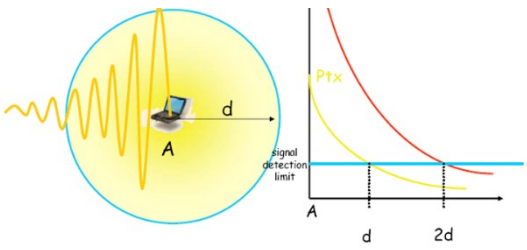


- Frequency (f), which is the *number of oscillations/tone* in one second (measured in hertz – cycle/sec)
  - Only specific frequencies can be used inside the wireless spectrum
  - some are free, others are regulated



- Wavelength, the *distance between two high (or low) peaks* (between spikes)
  - It is calculated as $\frac{c}{frequency}$ and gives antenna recommended length
  - This means that antennas work better with size $1, \frac{1}{2}, \frac{1}{4}$ of wavelength (size $\frac{1}{2^n}$ of wavelength)

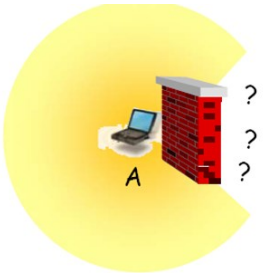
- Propagation, which determines the RF coverage of the signal (*how far it can possibly reach*) – from its origin, after a certain point the signal is no longer detectable
  - It depends on the strength of the signal itself
  - Signals become weaker in an exponential decline
    - a weak signal can be detected
      - but can't really use it



Written by Gabriel R.

- The signal reduces with a power transmission factor
  - specifically a cubic one
  - considering obstacles and there can be problems caused by the environment
- Propagation range depends on power, obstacles, receiver's sensitivity and many factors

In this case, remember that obstacles can reflect or absorb waves and it depends on material and frequency used. In general, remember the following rule of thumb:



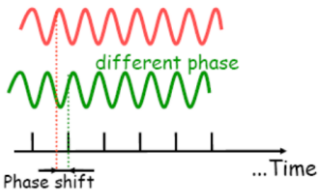
- high frequencies = good for short distances/more affected by obstacles
  - gets weaker faster
  - at some point the signal cannot be read anymore
- low frequencies = good for long distances/less affected by obstacles
  - these signals remain more readable

RF have other properties that describe their behavior, for example:

- Phase (ϕ), determining *the wave shift* (in degrees or radians) with respect to a reference signal
  - describing *the position in a specific moment* and *has a specific shape* known as *waveform*
    - usually, they propagate in a toroid form (a donut-shaped one)
  - It keeps the coherence of the signal for proper reception
    - deletes interference and differences in signals may mitigate effects in degradation
  - We might come across interference in synchronization and signal cancellation

In phase context, we distinguish between:

- Positive phase (left-shift, early wavefront)
- Negative phase (right-shift, late wavefront)



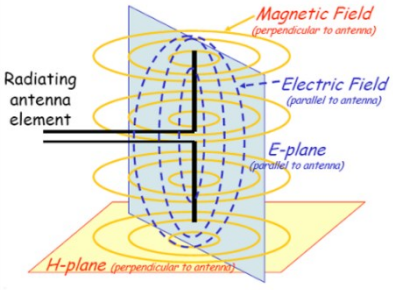
A *wavefront* is an imaginary surface representing points in space that are reached by a wave at the same instant. This is useful to determine the surface and its dimension according to the shape, aiding in understanding propagation according to different influencing factors.

RF echoes arriving at receivers with different phase may have positive or negative effects:

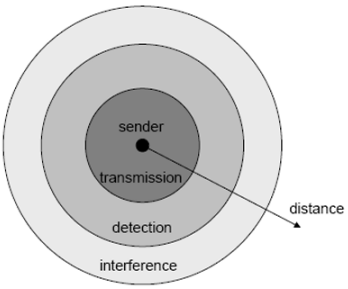
- If RF echoes arriving at receivers have different phases but are in-phase
  - they can experience *constructive interference* (amplification)
- If RF echoes arrive at receivers with different phases and are out-of-phase
  - they can experience *destructive interference* (cancellation/nullification)

Continuing with other RF factors:

- Polarization, which is the *physical orientation of the antenna*
  - It tells us how the electric field in the wave oscillates
    - this orientation affects how the wave interacts with antennas and objects
  - It can be either:
    - Horizontal (electric field parallel to ground)
    - Vertical (electric field perpendicular to ground)
      - This is typically used in WLANs
      - If 2 antennas are perpendicular to ground → better transmission/radiation



<u>Propagation ranges</u> depend on many things: power, obstacles, the receiver's sensitivity and many other factors. In particular, we can determine the following:

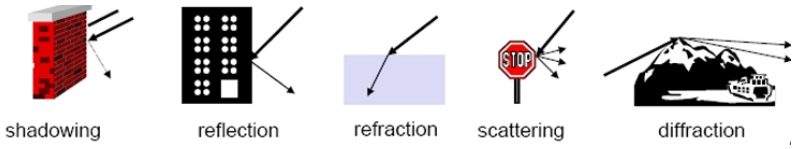


- *Transmission range* = how far the communication reaches
  - Communication possible, low error rate
  - This is what we care about: how far we can transmit
- *Detection range* = how far can the signal be found
  - Detection of the signal possible
  - no communication possible via exchanging messages
- *Interference range* = the distance at which the signal is too far away from the sender to be detected
  - Signal may not be detected, it may add to background noise
  - There may be many other factors, like distance/environment, etc.

There can also be <u>propagation effects</u>:

- how electromagnetic waves travel from a transmitter to a receiver according to the speed of light
  - usually in a straight line
- considering the receiving power is $rp = \frac{1}{d^2}$
  - where d is the distance between sender and receiver

The receiving power can be influenced by different factors (listed here visually to help you memorize):

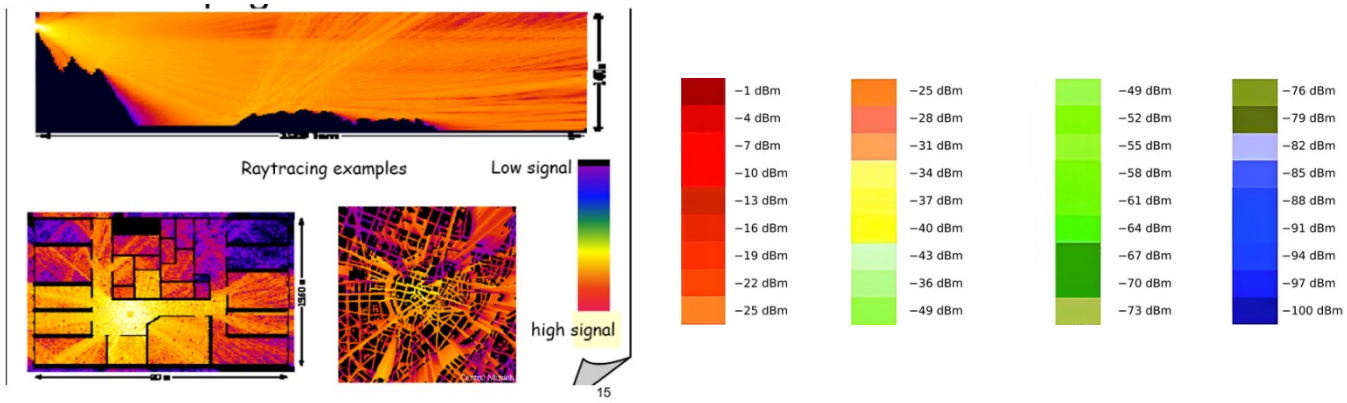


For the sake of simplicity, each factor will be briefly explained as follows:

- *Fading*
  - Signal strength variation due to multipath propagation, *frequency-dependent*
- *Shadowing*
  - Signal attenuation caused by *obstacles*, leading to varying signal strength
- *Reflection*
  - Waves bouncing off *large obstacles*, affecting signal paths
- *Refraction*
  - Bending of waves due to *obstacle density*, influencing signal propagation
    - happens when passing to a different medium
- *Scattering*
  - Waves changing direction when interacting with small obstacles
    - power is sent in different directions
- *Diffraction*
  - Bending of waves *around edges*, enabling signal penetration beyond obstacles
    - splitting the signal in two halves

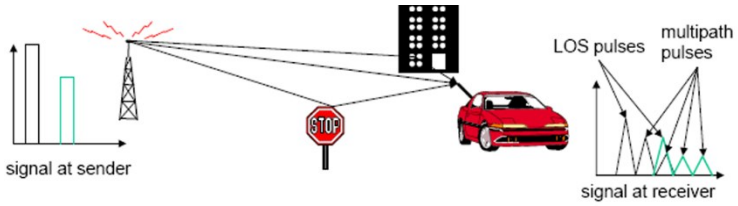
The left figure shows how propagation effects occur in presence of different signal power via raytracing (calculating light path when it interacts with areas). Note that in the picture, yellow is wrong and it's the reverse (red is the low, violet is the strong, Palazzi told us that in the lesson).

A more correct one might be the right figure (found it myself online):



Multipath propagation is a phenomenon in which RF signals take multiple paths to reach the receiver.

- Happens when original signal reflects/scatters off surfaces
    - or experiences different propagation conditions
    - resulting in several delayed copies of the signal arriving at the receiver



This can bring:

- Time dispersion of the signal over time
    - It happens because of interference with neighbor symbols – Inter Symbol Interference (ISI)
- Phase shift of the signal because of signal distortion in different parts
    - The final signal received could be better or worse than the one originally sent

We normally use *decibel (dB)* ($\frac{1}{10}$ of a Bel, which by itself measures the ratio of two values of a power or root-power quantity on a logarithmic scale), a measurement unit designed to *express power loss*:

- it is more practical to use given the logarithmic decay of wireless signals
    - if positive = power gain / if negative = power loss
- it allows to make easy calculations on “resulting power”
- it measures the *logarithmic relative* strength between 2 signals
    - giving relative power measurements
- it is complementary to other linear absolute measures for energy and is used as reference
- power difference between transmitted signal and received signal is calculated in dB

4.2 ANTENNAS

We talk about antennas, which convert electrical energy in RF waves (transmission) and viceversa (reception). Consider different factors about them:

- Their *size* is related to *RF frequency of transmission and reception*
- Their *shape* is related to *RF radiation pattern*
- There are radiation patterns of different antenna types (dipole, yagi, isotropic, parabolic, etc.)
- The *positioning* of antennas is crucial trying to *get maximum coverage* of workspace

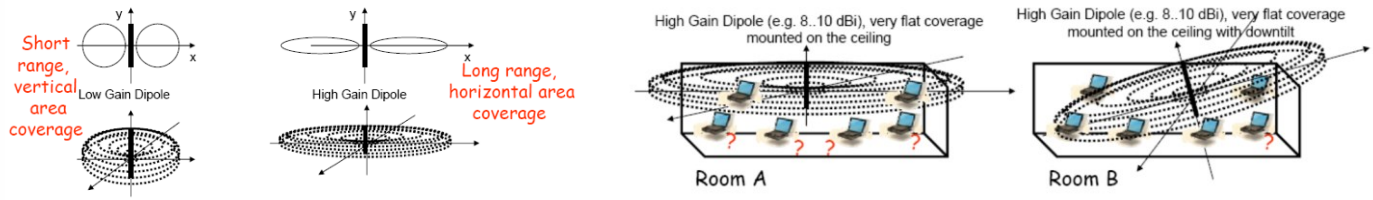
Written by Gabriel R.

There are three main types of real antennas to consider.

4.2.1 Omni-directional antennas

They radiate RF power equally in all directions around the vertical axis (Y-axis), radiating equally in all horizontal directions (usually in a doughnut-shaped pattern). Common example is dipole antenna (see APs).

- They are used when in need of:
  - uniform radio coverage around a central point
  - point-to-multipoint connections (e.g. star topology)
- They can cover both short and long range
- We measure the *tilt*
  - the degree of inclination of antenna with respect to Y-axis
  - this can be exploited to reach devices on different floors – right figure
- To prevent radiation sprawl, it's important to direct an antenna's vertical pattern
  - in a process called *downtilt*
  - which some antennas allow in a variable degree
  - half signal dispersed "in the sky", 2nd half better exploited
- The dipole has passive gain due to concentration (shape) of radiation
  - whereas active gain when we obtain it with power amplifiers
  - (external sources of energy) – left figure
- Near and below the dipole the signal is weak
  - a better radiation is obtained in sub-areas around the dipole



There is the problem of *how and when to mount* this type of antennas:

- *Ceiling mount*
  - Centralized and unobstructed coverage, suitable for open spaces
  - May face some attenuation because of dense materials like concrete or metal
- *Wall mount*
  - Strategical placing for optimal coverage in specific directions
  - May limit the overall range compared to the previous one
- Indoor = potential signal absorption or reflection / Outdoor = potential interference sources

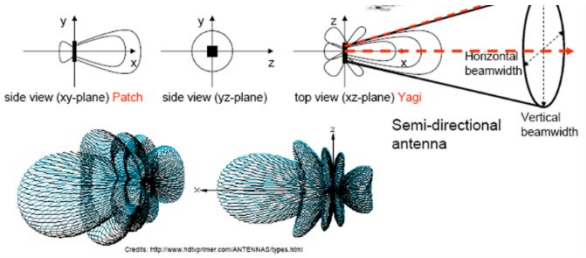
4.2.2 Semi-directional antennas

They don't transmit in all directions and can be positioned against walls without wasting too much signal:

- They usually transmit in (mostly) one direction to have the stronger signals
  - concentrating between a limited space (figure below details signals propagation)
- Radio frequency power is equally distributed only on $\frac{1}{2}$ direction
  - also few go behind that direction
- Used commonly to provide a network bridge

Written by Gabriel R.

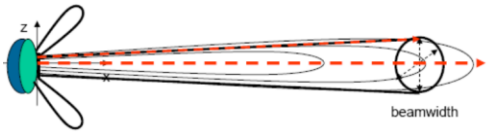
- More commonly used as a central device to provide unidirectional coverage from APs to client or for outdoor point-to-point communication
- Common types are:
    - *patch*: flat antennas mounted on walls and short coverage
        - used in Wi-Fi or RFID
    - *panel*: flat antennas mounted on walls, but they are larger ones
        - used in outdoor deployments
    - *yagi*: highly directional and with high gain, made by a rod with tines sticking out
        - commonly used in reception of TV signals



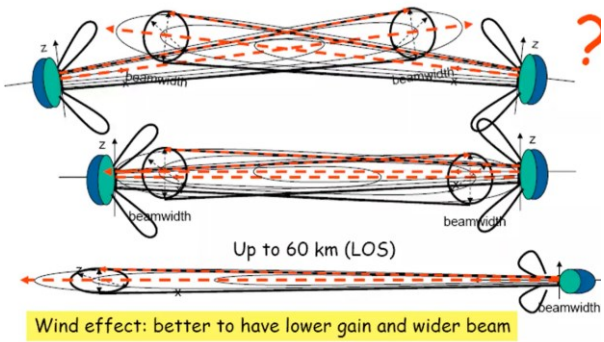
4.2.3 High-directional antennas

They have a radio frequency power distributed on one specific direction, allowing for a specific targeting signal towards an obstacle (see figure below)

- They are used by parabolic dishes and grids – common usage point-to-point link
- They reach very far, covering a distance up to $60\ km$ assuming no obstacles in between
    - this is the so-called Line Of Sight – LOS
        - straight line between sender and receiver with no obstructions
- Ensure proper alignment of communicating antennas by placing the sender and receiver at an appropriate distance (problems like out of beam alignment can happen)
    - This prevents signal misalignment against the curve of the earth
    - It also ensures resistance to wind or adverse weather conditions that could cause tilting – this happens when sender and receiver are not perfectly aligned



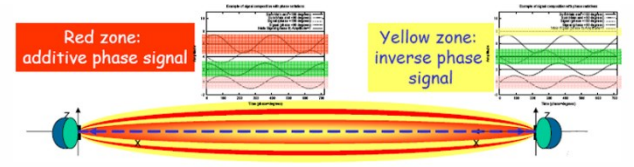
Below an example of the need of exact alignment:



4.2.4 Fresnel Zone

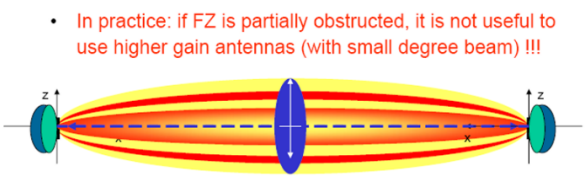
Another thing to consider is the Fresnel Zone (FZ), which is an ellipsoidal region of space surrounding the direct Line-of-Sight (LOS) between a transmitter and a receiver when transmitting (see figure below).

- The most additive RF signal is concentrated in this zone
- Here, the power is not concentrated all in the center and doesn't proceed in a straight line
  - it serves as a measure to determine how good the signal is without problems
- There is the need of no obstacles (useless increasing power if Fresnel Zone is not free)
  - because signal disperses, we need to consider that



Also, remember that:

- The FZ radius depends only on the distance d between antennas and frequency f of the RF signal
- The type of antenna, the beam width (degree) and the gain (dBi – "i" for isotropic) have no effects
  - Yagi vs Dish have same FZ even with very different degree beams (30 vs 5)

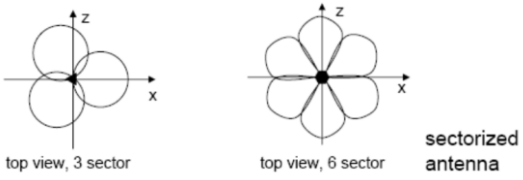


- So, it is independent from the energy used – only depends on distance and frequency

4.2.5 Sectorised-directional antennas

They are multiple antennas focusing RF signals within a specific angular sector (limited coverage area, typically in a pie-shaped or sector-shaped patterns).

- They usually work in arrays or with space multiplexing (allowing channel reuse) – see figure
- They assign the same frequency for such antennas to avoid collisions between each other



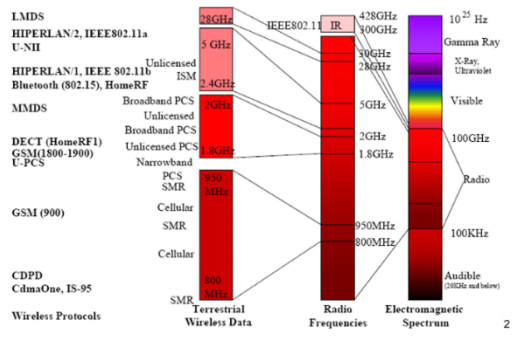
An interesting example from slides comes from the Radio Network used between University of Bologna and campuses in Emilia-Romagna region. More specifically:

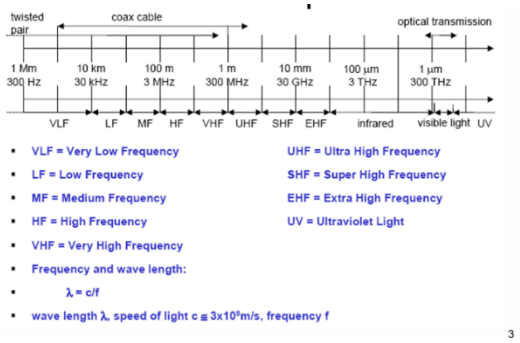
- The transmission goes asynchronously (ATM – Asynchronous Time Mode)
  - using specific heights and types of antennas
  - taking carefully into account the effect of the Fresnel Zone
- The signal attenuation was both considered in free space or rain intensity/transmission frequency
- This was shown at the end of WNMA02; from the following paragraph onward is WNMA03

4.3 WIRELESS SPECTRUM

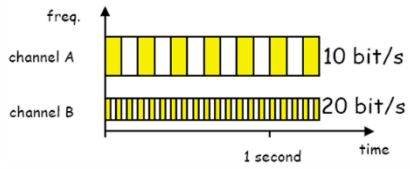
The wireless spectrum, or simply spectrum, is the *range of all radio frequencies* used for wireless communication and can be regulated or free.

- It encompasses a broad band of electromagnetic frequencies, including radio waves, microwaves, and even light, where each one has a dedicated area
- Here you can have a look at a breakdown of them (left) and all of the frequencies (right)

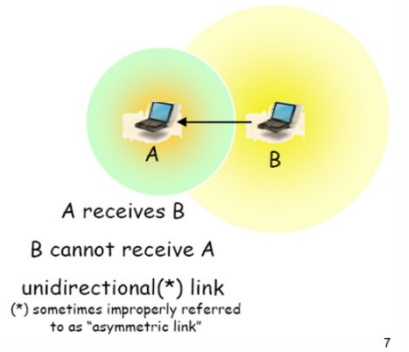




- How can wireless channels have different bandwidth (BW)?
  - Bits *already run at the speed of light* (we *can't make them faster* than that)
    - there is a gain in encoding/decoding given bits go at the same speed
  - The *channel pipe* being bigger (*wider spectrum*) influences different bandwidths
    - A broader spectrum accommodates more frequencies and more data
    - Instead, we can think of sending them faster according to channel pipe
      - because of variable spectrum, sending data more frequently
    - The variable spectrum has more space, but also more risks to be accounted for
      - e.g. errors, interferences, etc.
  - The channel requires less time to accommodate one bit on the channel
    - sending bits more frequently means making them more packed)



The transmission range between two nodes needs to intersect at least a bit, otherwise they can be both isolated when far (because of shadowing, signals can reach the other), like happens here:



We have a unidirectional link if, given A and B, A receives B but B cannot receive A. Sometimes it's improperly referred as "asymmetric link" (asymmetric = if they transmit at different speed rates).

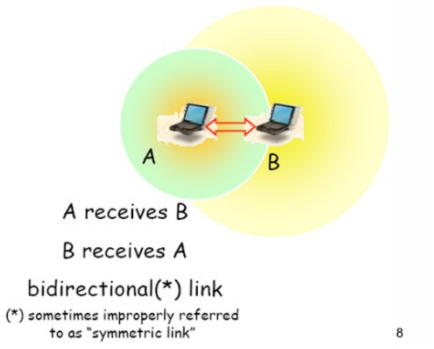
This case happens when one of the hosts is just broadcasting or flooding the channel – with a unidirectional link, if such happens, the other one waits for an ACK that never happens. One host doesn't know the other exists so bandwidth is wasted.

We also have bidirectional link when A receives B and B receives A.

Similar to the previous one, this is improperly referred to "symmetric link" (so, transmission of both at same speed).

It can be both:

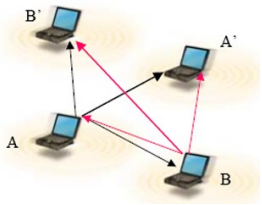
- *symmetric*: all devices send at the *same* speed
- *asymmetric*: *different* speed of transmission between devices in the network



4.4 Wireless Technology, Coverage and Multiplexing

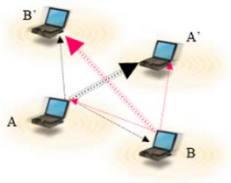
Let's give some examples of wireless network technology:

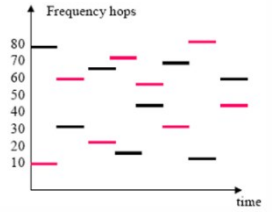
- Narrowband radio system, a type of wireless communication system that transmits/receives using a single, licensed and narrow as possible radio frequency (or having a small fractional bandwidth)
  - This is normally used for long distance, so Line-of-Sight (LOS) is needed
  - This kind of communication has very little bandwidth and low data rates
  - To avoid undesired cross-talk between channels, coordination and license for each site are required (so, again, low rate)
  - Its usage is quite limited, pretty much for satellites and not for WLANs



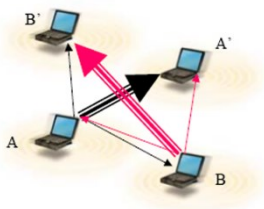
There is spread spectrum technology, using specific frequencies (bandwidth efficiency vs reliability and security):

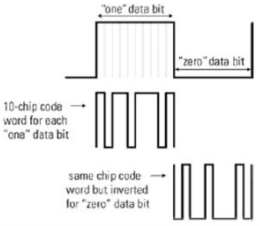
- Frequency Hopping Spread Spectrum (FSSS), which is a technique used to rapidly switch the carrier frequency in a pattern of frequencies known by both transmitter and receiver ("secret" to them)
  - To better use the whole band
    - we can divide it in smaller frequencies
  - It uses *carrier hopping* in a pattern sequence, avoiding jamming and interference
    - to unintended receivers
      - this appears as impulse noise
  - Because of the continuous hopping, it has a low throughput overall
  - In any case, it has also a low power and cost compared to the following one





- Direct Sequence Spread Spectrum (DSSS), which is a modulation technique used which works by sending a redundant bit pattern over a large spectrum
  - It increases probability of recovering lost original bits and it may avoid retransmission
  - To unintended devices, this appears as low power wideband noise
  - Original data signal is combined with higher rate bit sequence, known as chipping code
  - It's quite more expensive overall than the previous one





Written by Gabriel R.

- infrared technology, where the transmission requires short range and LoS, using frequencies just below the visible light achieving overall good bandwidth
    - It cannot penetrate opaque objects (it's easily obstructed) and has low diffusion
    - It is also constrained to the LoS which limitates mobility
    - It works well in short-range technology like indoor, PAN, LAN nets
        - given its high bandwidth and data-rate potential while being pretty inexpensive

For your visual reference, a summarizing comparison table:

	PROS	CONS
Frequency Hopping Spread Spectrum (FHSS)	• Use less power than DSSS • Lower cost • Increased security due to frequency switching	• Lower throughput than DSSS
Direct Sequence Spread Spectrum (DSSS)	• High performance • Low interference • Increased security due to chip coding	• Expensive
Narrowband Microwave	• Long distance	• Line-of-sight with satellite dish • Requires FCC license • Not designed for WLAN use
Infrared	• High bandwidth	• Easily obstructed • Inexpensive

We define the coverage using different names for the various networks:

- *Wireless Wide/Metropolitan Area Networks (WWANs/WMANs)*, which cover *large geographical areas*, like countries or regions/metropolitan areas and they employ different technologies:
    - *Satellites* (LEO – Low Earth Orbit/GEO – Geo-stationary Earth Orbit)
        - GEO can cover the whole globe (with just 3 of them)
            - but have limited RTT ($500\ ms$)
        - LEO suffer from handovers/handoffs (switching connections from one to another)
            - due to satellite mobility (so, low coverage and nodes switching)
    - *Cellular of multi-infrastructure WLANs*
        - There are grid of Access Points (APs) connected to local Mobile Terminals (MTs)
        - The MTs are hence connected to backbones

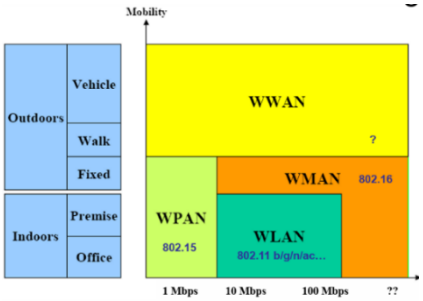
- *Wireless Local Area Networks (WLANs)*, which are used for *local area coverage*, like campuses, and buildings (even home-based ones). They usually are of two types of them:
    - Ad-hoc
        - They are meant to be P2P "on-the-fly" communication
        - Here, the network "is" the set of computers, with no administration/setup/costs
    - Infrastructure
        - They have a centralized control unit (AP + Local Server)
        - There is roaming between cells (they take the available network at any point)
        - There is resource sharing and backbone connection

- *Wireless Person Area Networks (WPAN)*, which is used for reduced area coverage, for example Bluetooth, e.g., audio headsets, pointing devices
    - They are not suitable for much more, even house coverage is too much
    - This is basically a cable connection alternative for in-home/office/workspace connection
    - Common technology and protocols are required (e.g. HomeRF, Bluetooth)

- *Wireless Indoor Area Networks (indoor)*, which have a really short coverage
    - They basically cover in room/workspace device connection

Written by Gabriel R.

Their positioning may be summarized as follows:



The following tables describe wireless vs wired comparison:

Attribute	Wireless PAN/LAN	Wired LAN/PAN
Throughput	10-100 Mbps	10-100 Mbps (and more)
Integrity & Reliability	Subject to interference	Highly reliable
Simplicity/ Ease of Use	• No need to pull cable • Set up time is significantly lower • Moves, additions & changes much simpler	• Cable required • Set up time is significantly higher
Security	• Susceptible to interception • encryption	• Not as susceptible to interception

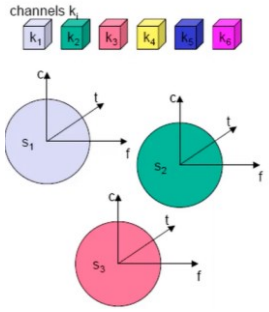
Attribute	Wireless LAN/PAN	Wired LAN/PAN
Cost	• Initial investment in hardware costs more • Installation expenses and maintenance costs can be significantly lower	• Investment cost in hardware lower • Installation and maintenance costs can be significantly higher
Scalability	simple to complex networks	simple to complex networks
Safety	Very little exposure to radio frequency energy	No exposure to radio frequency energy
Mobility	Provides access to real-time information anywhere	Does not support mobility

We need to consider the following environment challenges:

- maintain specific needs for services or applications
- deal with limited resources (bandwidth, energy, device limits for I/O, UIs)
- deal with mobility of users (in their number) and devices
- deal with QoS (Quality of Service) problems in reliability and negotiation
- consider APs need to have both the wired and wireless protocols
    - because they don't just relay the information, they have to translate it

We use multiplexing to have different channels sharing the same medium while using different dimensions:

- space (s_i)
    - devices are distant from each other, and they have as guard the physical space
    - they all share the same frequency with no interference
- time (t)
    - one carrier uses the whole bandwidth at a time
    - as guard, time between transmissions has to be synchronized
    - this ensures high throughput for many users but has to be precisely controlled



- frequency (f)
    - the frequency is divided into smaller bands
    - one host uses a single piece the whole time
    - as guard, we have safety frequency between the bands
    - while requires no dynamic coordination and works also for analog systems
    - it's also inflexible, traffic is unbalanced and overall it wastes bandwidth
- code (c)
    - each channel has a unique code (it uses spread spectrum technology)
    - messages overlapping/signal combination is common (each medium transmit at same time)
    - the receiver will decode only what is of his interest
    - at a cost of lower data rates and need to regenerate the signal at receiver
    - it needs no synchronization, achieves more bandwidth, better protection against tapping

Written by Gabriel R.

5 MAC LAYER (WNMA04 – WNMA05)

There needs to be a proper way to control access to a shared channel (*access control*), guaranteeing coordination and scheduling of transmissions among competing neighbors (providing *flow control* and *multiplexing*). This is handled by the Multiple Access Control (MAC) Layer, having as *features*:

- coordination and scheduling of transmissions
- broadcasting of wireless transmission (at the speed of light), deciding when/where to transmit
- define the correct access control to shared channels
    - avoid collisions (no collision detection = resilience), both with detection and not
        - if both receive at the same time, they will not transmit
    - deciding how to transmit (when/where), given the hosts competition
    - guarantee a good resources utilization (both capacity/power to hosts)

Generally, as *goals*, we can identify the following ones:

- having low latency to avoid over usage of battery (*efficiency*)
- guaranteeing a good channel utilization, with no collisions (using it as much as possible)
- achieving best effort + real time support for good channel optimization
    - no guarantee of delivery, variable factors in network load

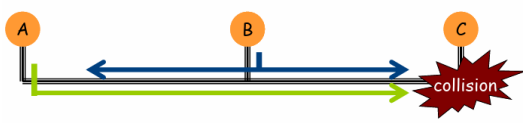
This layer *coordinates* activities like in a human conversation: equal time to talk, don't interrupt the others and don't monopolize the conversation, raise your hand if you want to ask something and do not fall asleep when somebody is talking. These are general rules, so not valid for every possible situation.

Considering B = available bandwidth on the channel, the desirable MAC Layer's *aims* are:

- *Efficiency* in bandwidth use (sum of rates = B): the maximum possible
    - If just one node, it should transmit at B rate
- *Resilience*: avoid collisions
- *Fairness*: if N nodes want to transmit each should have $B_N = \frac{B_{tot}}{N}$ bandwidth available (in average)
- *Robustness*: the protocol should be decentralized (no single point of failure)
    - this serves to find new paths easily for transmission
- *Simplicity*: the protocol should be easily implementable

Given multiple nodes share a channel, the *Channel Access Problem* arises, considering simultaneous communication is not possible (pairwise communication is desired).

- MAC protocols suggest schemes to schedule communication, maximizing number of communications, guaranteeing fairness among all transmitters).
- Trivial solution: simply "transmit and pray", meaning "try to communicate and hope for the best"
    - There are plenty of collisions and there is poor throughput at high load
    - This can be efficient when the number of transmissions is very low



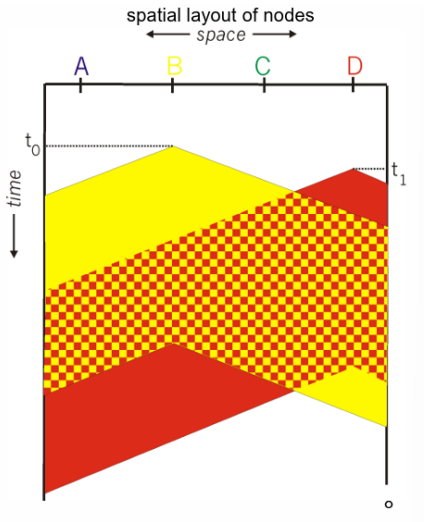
To fix the transmit and pray, we should transmit only from one part, but this requires knowledge of the channel, in particular having a sense of what is going on the network currently.

Written by Gabriel R.

A good idea might be deferring transmissions when signals are on the channel, so we might need a protocol able to "listen" somehow and see if there's anyone – this fix is guaranteed by the following protocol.

That's exactly the principle of Carrier Sense Multiple Access (CSMA), which is a contention-based protocol where multiple devices on a network listen to the communication channel (the "carrier") before attempting to transmit data ("carrier sensing").

The key idea is to avoid data collisions, which occur when multiple devices try to send data simultaneously, causing signal interference and making the transmitted data incomprehensible.



- Transmitters listen to the channel before sending
  - they will be waiting when signal on channel
- Collisions can still occur
  - propagation delay non-zero between transmitters
- When collision
  - entire packet transmission time wasted
- Distance and propagation delay in determining collision probability should be considered

When the channel is a cable it's easier because you just have to listen to it before transmitting any data.

- In a wireless environment, it is not possible to check what has been sent and if a collision happened, because when you're transmitting you can't receive
- Otherwise you'll end up having a collision; instead, you can check if a package has arrived at the destination with the ACKs

5.1 MAC Layer Protocols

A MAC Protocol should coordinate transmissions from different stations, minimizing or avoiding collisions. There are many MAC protocols to consider, and we can classify (also, hybrid solutions are possible).

- Random Access
  - *Without carrier sensing* (ALOHA, Slotted ALOHA)
  - *With carrier sensing* (CSMA, CSMA/CD, MACAW)
- Controlled Access
  - *Centralized*: there is an entity that is responsible to regulate the access to the channel
    - *Channel Partitioning* (e.g. TDMA, FDMA, CDMA = Time/Frequency Code/Division Multiple Access)
  - *Distributed*: channel access is controlled by a distributed application, with peer nodes
    - For example, the token ring, deciding via a token who's gonna transmit
- Polling ("Taking Turns") = used to see which nodes want to access the network (e.g., PCF)
  - There is a central control entity continuously asking the state of each device

Specifically, in Random Access Protocols:

- A node transmits *at random* (i.e. no pre-coordination among nodes) at *full* channel data rate R
- If the transmissions of two or more nodes *collide*, they retransmit at random times
  - The protocol specifies how to detect/recover in a different way
    - e.g., delayed retransmissions

Written by Gabriel R.

Consider for each the following parameters:

- N: *number* of stations
- P: *probability* that each station transmits in the slot
- S: probability of *successful* transmission
- C: probability of *collision*
- E: *empty* slots

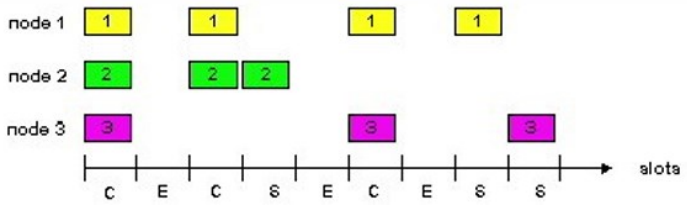
5.1.1 ALOHA Protocols

Examples of this kind of protocols are ALOHA, developed in the 70's by University of Hawaii to have islands able to communicate among themselves and with the mainland.

- It's a simple random-access protocol, in which devices simply transmit without a schedule
- It's based on positive/negative acknowledgements
  - via monitoring and control of a central hub
  - the total throughput efficiency (e) of this protocol is $\frac{1}{2e}$

There are two types of ALOHA protocols:

- Slotted ALOHA (S-ALOHA), where time is divided into equal size slots (= full packet size)
  - A newly arriving station transmits at the beginning of the next slot
    - so, every transmission must stay inside a defined time slot
  - If collision occurs:
    - assuming channel feedback, e.g. the receiver informs the source of a collision
    - the source retransmits the packet at each slot with probability P
      - given the N stations, until successful
  - This protocol is fully decentralized
    - no central authority, each node has relevance
  - Throughput (how much we transmit) efficiency is $\frac{1}{e}$

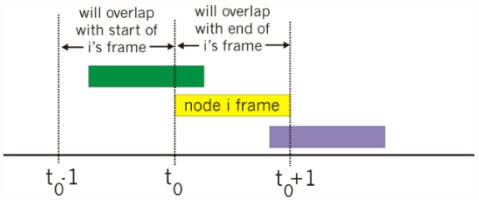


the value of S is:
- $S = p(1-p)^{(N-1)}$ by a single node
- $S = Np(1-p)^{(N-1)}$ by any of N nodes

- throughput efficiency is about $\frac{1}{e}$ → and:
  1. obtaining $p = \frac{1}{N}$ (p should be tailored based on N)
  2. substituting p to $S = Np(1-p)^{(n-1)} \Rightarrow S = N\frac{1}{N}(1-\frac{1}{N})^{(N-1)}$
  3. solving S at the limit obtaining $S = \frac{1}{e}$

- Pure (Unslotted) ALOHA; a simpler version which *does not require slots* and does not check if medium is busy or not, it just transmits. Here:
  - A node transmits without awaiting for the beginning of a slot
  - Collision probability increases (packet can collide with packets transmitted in a "vulnerable" window twice as large as in Slotted ALOHA)
  - Throughput is reduced by one half, i.e. $S = \frac{1}{2e}$

the value of S is:
- $S = p(1-p)^{2(N-1)}$ by a single node
- $S = Np(1-p)^{2(N-1)}$ by any of N nodes



- throughput efficiency is about $\frac{1}{2e}$ → every transmission can occupy 2 slots → and:
  1. obtaining $p = \frac{1}{2(N-1)}$ (p should be tailored based on N)
  2. substituting p to $S = Np(1-p)^{2(N-1)} \Rightarrow S = N\frac{1}{2(N-1)}(1-\frac{1}{2(N-1)})^{2(N-1)}$
  3. solving S at the limit obtaining $S = \frac{1}{2e}$

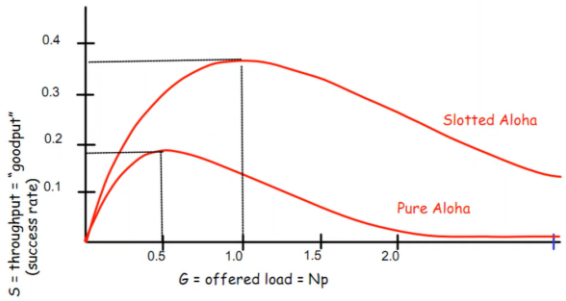
Written by Gabriel R.

Let's make some considerations about the ALOHA protocols:

- They are usually *not efficient*, consider they trigger a lot of retransmissions:
  - Consider 1 is the full channel availability
    - Pure Aloha throughput – $18.4\ \%\ (0.184)$
    - Slotted Aloha throughput – $36.8\ \%\ (0.368)$
- They are *unfair*
  - aggressive senders can capture the channel
- They are *robust*
  - there is no entity to control the system
  - decentralized, would require clock/slot synchronization
- They are *simple*, considering:
  - No coordination for Pure ALOHA
  - Just synchronization for Slotted ALOHA

Low performances of Pure Aloha and Slotted Aloha are due to the lack of coordination among nodes and efficiency can be improved if each node behaves coherently with what other nodes do.

Right here you can see precisely.



5.1.2 CSMA Protocols

With Carrier Sense Protocols, each node continuously listens to the channel to be aware of what other nodes are doing and then the protocol algorithm tries to avoid transmission at the same time. These come in different versions based on the way they handle transmission and collisions:

- 1-persistent CSMA
  - Before transmitting, each node listens to the channel:
    - If the channel is *free* → it transmits the packet *immediately*
    - If the channel is *busy* → it waits for the channel to be free
  - If two nodes are waiting the channel to become free → they will probably cause a collision
    - because they will start transmitting at the same time
    - if there is a collision
      - the transmitting node waits for a random time
      - then tries again to transmit (desynchronization)
  - Propagation time has a significant impact on performances
    - More propagation time means more collisions (and hence less efficiency)
    - E.g., an host transmitting and can't hear another one
  - There could be collisions even with no propagation time
    - If two nodes want to transmit
      - while there is a third one that is currently occupying the channel
    - As soon as the channel becomes free
      - both waiting nodes will transmit immediately, at the same time
      - collision happens

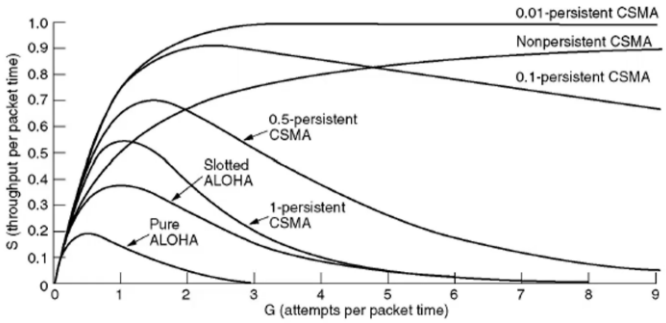
Written by Gabriel R.

- non-persistent CSMA
    - Before transmitting, each node listens to the channel
        - if the channel is *free*, then it transmits the packet *immediately*
        - if the channel is *busy*, then it waits for a *random time* and then tries again
    - This solution is less aggressive than 1-persistent CSMA

- p-persistent CSMA
    - Slot based system
    - Before transmitting, each node listens to the channel:
        - if the channel is *free*
            - it *transmits with probability* p at the beginning of the next slots
        - *otherwise waits for a random time* with probability $1 - p$
            - then tries again the procedure
    - The aggressiveness of this protocol depends on the parameter p (number of nodes)
        - many nodes – it may be conservative
            - bandwidth waste depending on number of collisions
        - few nodes – it may be aggressive
            - bandwidth waste depending on time of channel not used

- CSMA with Collision Detection (CSMA/CD): it has carrier sensing and deferral like in CSMA, but collisions are detected within a few bit times, feedbacking all stations. Specifically:
    - Persistent transmission is implemented
    - When collision is detected:
        - transmission is aborted, usually using *jam signals*
            - then entering backoff periods, trying to retransmit data
        - reducing the channel wastage considerably
    - CSMA/CD can approach channel utilization $= 1$ in LANs
        - low ratio of propagation over packets transmission time
    - CD *is easy in wired LANs* (e.g., Ethernet, which was were this one was used)
        - it can measure signal strength on the line, code violations
        - compare tx and receive signals
    - CD *cannot be done* in WLANs (*or wireless networks in general*)
        - the receiver is shut off while transmitting, to avoid damaging it with excess power

Here is a comparison between all different versions of CSMA and ALOHA versions:

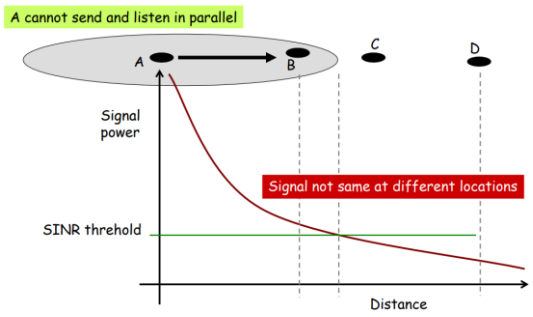


5.1.3 Wireless Access Control

Consider:

- transmission strength drops exponentially as the distance grows
- there are metrics like *Signal-to-Interference-plus-Noise Ratio (SINR)*
  - measure of the strength of the desired signal (energy of signal fading away with distance)
  - relative to the strength of both interference and background noise in a channel
- this threshold serves as a *measure of path loss* in wireless networks
  - in wired ones, it represents the correct reception of data
    - given there exists already a path from sender to receiver

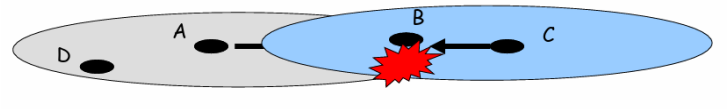
The SINR threshold represents the bound in which the signal can't be detected anymore (wireless media disperse energy in figure):



- C can hear but not correctly receive the message
- A can't send/listen at the same time
- There can be collisions between hosts
  - E.g., Collision with $A \rightarrow D$ or $B \rightarrow C$
- Problem is, A cannot send and listen in parallel
  - while signal is not the same at different locations

In a scenario like the following figure, where signal reception is based on SINR:

- A is out of C range (and viceversa), but both have B on range
  - They will send together, having a collision at receiver B
- Both hosts:
  - don't know their position
  - can only hear themselves – no listening while transmitting
  - can't determine the signal quality at receiver



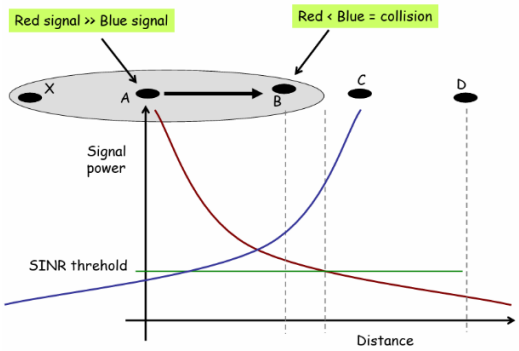
The specific formula to calculate the SINR (even with inverse formulas) is here on left figure, while the right one calculates SINR over signals:

$$SINR = \frac{SignalOfInterest(SoI)}{Interference(I) + Noise(N)}$$

$$SoI_B^A = \frac{P^A_{transmit}}{d^\alpha_{AB}}$$

$$I_B^C = \frac{P^C_{transmit}}{d^\alpha_{CB}}$$

$$\rightarrow \quad SINR_B^A = \frac{\frac{P^A_{transmit}}{d^\alpha_{AB}}}{N + \frac{P^C_{transmit}}{d^\alpha_{CB}}}$$

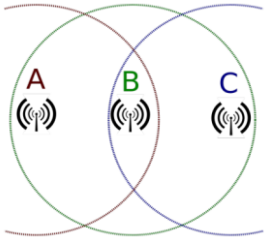


Consider in the formulas also interference is considered and also N (noise) is a somehow a fixed value.

5.1.4 Hidden/Exposed Terminal Problems

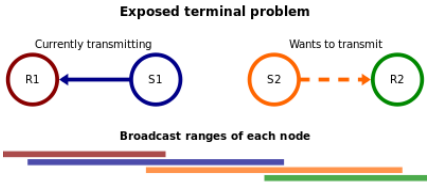
Two problems, which will be explained in detail in the next part, arise:

- Hidden Terminal problem (HT)
  - Scenario
    - C is transmitting to B
    - A wants to send to B
      - but cannot hear C is already sending
    - A transmits to B
      - B has two incoming transmissions
        - has interferences



  - Practical summary
    - nodes don't know the topology
      - a node might be visible from an AP
      - but not from other nodes communicating with that AP
        - due to difficulties in the media access control sub-layer
    - they transmit to same node without knowing other's transmission
    - collision happen (sad story) = degrades throughput due to collisions

- Exposed Terminal problem (ET)
  - Scenario
    - Two receivers ($R1, R2$) out of range of each other
    - Two transmitters ($T1, T2$) the middle of range of each other
    - If a transmission between $S1$ and $R1$ occurs
    - Node $S2$ is prevented from transmitting to $R2$
    - After carrier sense it conclude it will interfere with its neighbor $S1$
    - Note that $R2$ could still receive $S2$ transmission given it's out of range of $S1$



  - Practical summary
    - transmission to different receivers, even if channel is free
    - one don't transmit thinking channel is busy (sad story again)
      - poor performance by wasting valuable transmission opportunities

In summary:

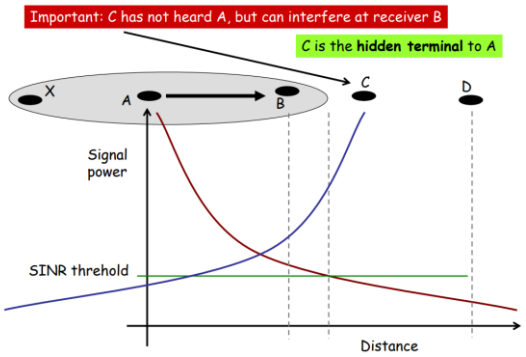
- *Hidden* – two nodes outside of each's range perform transmission to a node that is in each range
  - transmitters hidden from each other
- *Exposed* – a node is within range of a transmitting node, and it cannot transmit to any other

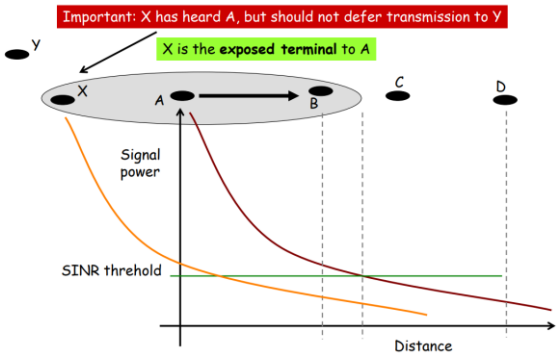
What to do to solve them?

- RTS/CTS *try to solve* (do not totally) these problems (see later for more)
- Signal detection in threshold could be a possible solution
  - Node intelligently choosing to sense the channel
  - This mitigates the collisions, but worsens the exposed terminal problem

Written by Gabriel R.

Below, as described, the figures related to the chapter (SINR and the problems just discussed). WNMA04 set of slides ends with this part.





5.2 802.11 PROTOCOL

IEEE 802.11 specifies the set of MAC and PHY protocols for implementing WLAN connections using the same protocol. Wireless MAC proved to be non-trivial:

- A research work in 1992 brought MACA (Multiple Access with Collision Avoidance)
  - it aimed to address the challenges of contention and collision
    - using a two-way handshake mechanism for collision avoidance
- A research work in 1994 brought MACAW (Multiple Access with Collision Avoidance for Wireless)
  - same operating principles of 802.11, widely used in ad-hoc networks
    - it enhances the previous one adding RTS/CTS frames (see below for meaning)
  - it helps solving the hidden terminal problem, but not the exposed terminal problem
- These led to IEEE 802.11 committee and the standard was ratified in 1999
- Its general aim is to have a common way to allow communication between nodes

The 802.11 standards regard in particular Wireless LANs (WLANs):

- *mostly indoor*
- *base station* (like cellular) or *ad-hoc networking* (mostly point-to-point)
- other standards included are HiperLAN (High Performance Radio LAN – ETSI) and Bluetooth

Consider the following remarks for 802.11 *features*:

- there are different applications:
  - e.g., nomadic Internet access, portable computing, ad-hoc networking (multi-hopping)
- unlicensed frequency spectrum bands are used → $900\ MHz, 2.4\ GHz, 5\ GHz$
- it defines both MAC and physical layer
  - e.g., radio frequency, header, size, ack, etc.
- WLAN configurations can work:
  - with or without control station (Access Point)
  - with ad-hoc/peer-to-peer networks

In all following sections, these terms will be *always* used from now on (a very brief overview here):

- DIFS (Distributed Inter Frame Space)
  - Time allocated for waiting for the channel to be free
  - If channel is occupied
    - timer restarts from the beginning
  - Low-priority timeframe for asynchronous data service

- SIFS (Short Inter Frame Space)
  - Brief interval used for processing high-priority procedures
    - like CTS (Clear To Send – see below), Data, and ACK (Acknowledgement)
  - High-priority timeframe for critical operations in the network

- RTS (Request To Send)
  - A frame indicating a node's intention to transmit and requesting permission

- CTS (Clear To Send)
  - A frame granting permission for the sender to transmit

- CW (Contention Window) – Note: it's different from $cwnd$ = Congestion Window (see later here)
  - A range of slots waited after a successful DIFS, with a slot equivalent to 1 SIFS
    - if the channel is occupied → timer stops and restarts from there
    - if there is a collision → CW is increased at max 1024 slots
    - when succeeding → CW resets to the value
  - Function
    - timer management: increased after collisions and reset after success
  - Many times indicated also with cw (almost always in this file from now on)
    - it varies in a range, so many times you will see cw_{min} and cw_{max}
  - It's basically a backoff value, later represented as BOV (Backoff Value), indeed

- NAV (Network Allocation Vector)
  - Time declared by sender to hold the medium (how long to defer from accessing medium)
    - it allows other nodes to conserve energy by entering a sleep state

The steps for *establishing a connection* through the MAC protocol in 802.11 are:

1. DIFS
   a. time where the channel waiting for the channel to be free
   b. if a node gains access to the channel, go to step 2, otherwise go to step 3
2. Data transmission
3. NAV
   a. time where the others have to wait before starting another transmission
4. SIFS
   a. time where the channel has to be free
5. MAC ACK

The contention time starts before the next frame and after SIFS/PIFS/DIFS occurs. The packages transmission order is the following:

1. DIFS
2. RTS
3. SIFS
4. CTS
5. SIFS
6. Data
7. SIFS
8. ACK

5.2.1 CSMA/CSMA-CA Versions

IEEE 802.11 MAC defines how devices access and share the wireless medium.

Let's delve into the CSMA version of the protocol:

- Sense channel idle for DIFS sec (so, wait for channel to be free), then:
  - if free → transmit frame (no Collision Detection) and receiver returns ACK after SIFS
  - if busy → there is a *randomized binary backoff* mechanism employed
    - DIFS – restarts the timer
    - cw – it stops and restarts at same timer position when newly free
- Here, also NAV is used, which imposes a minimum deferral time, even if no traffic is sensed
  - The sender declares a specific time (max $32{,}767\ \mu s$) to hold the medium using NAV
  - Allows other nodes to go to sleep, conserving energy during the declared time

More concisely, CSMA version works like this:

- transmitter waits a DIFS time ($\approx 16\ \mu s$)
- if the channel is:
  - busy – it restarts the DIFS counter
  - free – it goes to cw
- transmitter waits in the cw
- if the channel is
  - busy – it stops the timer and restarts at the same time of cw
  - free – it begins to transmit
- transmitter sends data → for a max of $33\ \mu s$
- receiver sends back an ACK after SIFS time ($9\ \mu s$) → automatic retransmission in case of errors

This CSMA version *doesn't entirely solve* the *hidden terminal effect* to deal with:

- Two or more terminals are within the communication range of a common AP or base station
  - but they cannot directly sense each other due to physical obstacles or signal attenuation
- Devices might falsely believe the wireless channel is idle and initiate transmissions simultaneously
  - thus leading to packet collisions

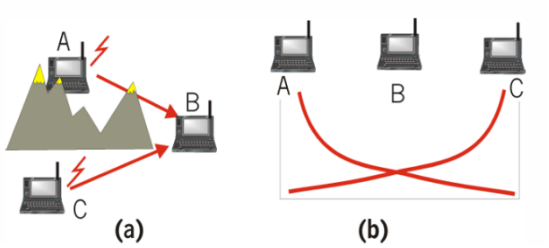
Consider for instance three hosts labeled A, B, C all within range and the following scenario:

- A cannot directly hear C
- C cannot directly hear A

Written by Gabriel R.

If both A and C attempt to transmit data simultaneously, they might not be aware of each other's presence, and their data packets could collide at B, causing data loss and retransmissions.

This situation results in inefficiency and reduced network throughput.



The solution is using CSMA/CA (on by default in our home networks), *which improves (but still does not solve)* the hidden terminal problem.

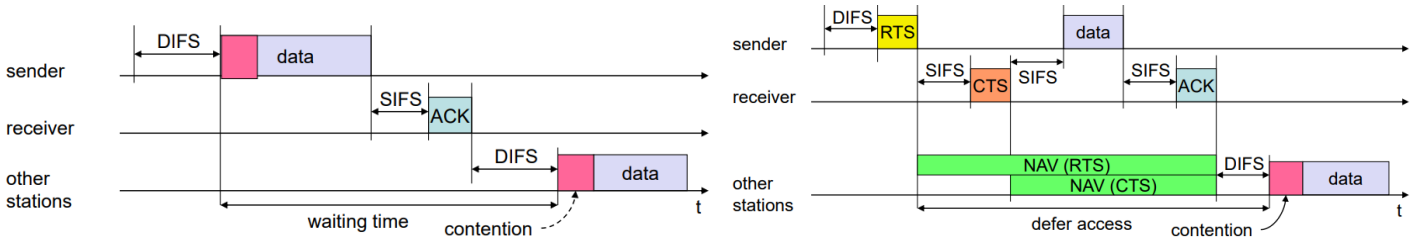
- It uses the Collision Avoidance (CA) adding:
  - RTS (Request To Send) – it freezes stations near the transmitter ("booking" the channel)
  - CTS (Clear To Send) – it freezes stations within range of receiver (but possibly hidden from transmitter), sent to AP which will respond in broadcast
    - this prevents collisions by hidden stations during data transfer
    - station could be possibly hidden from transmitter
  - RTS and CTS are very short: collisions during data phase are thus very unlikely
- Given it's carrier sensing → nodes listen before transmitting
- If the channel is
  - free
    - nodes begin to send the RTS/CTS and data
  - busy
    - NAV defer access to medium and if it was in:
      - DIFS – it restarts the timer
      - *cw* – it stops and restarts at same timer position when newly free
  - receiver returns to emitter ACK after SIFS amount of time

The mechanism works this way:

- transmitter waits a DIFS time (waiting for the channel to be free) and CW before sending data
  - Carrier Sense based on CCA, Clear Channel Assessment: if medium is busy/idle or not
- if the channel is:
  - busy – it restarts the DIFS counter (back to previous one)
  - free – it goes to *cw* (go to next point)
- transmitter waits in the *cw* (random backoff value – BOV – that tells the node to wait to send)
- if the channel is:
  - busy
    - it stops the timer and restarts at the same time of *cw* and goes into NAV
  - free
    - it begins to transmit (go to next point)
  - transmitter sends a RTS to receiver with reservation parameter
    - reservation declares amount of time the data packet needs the medium
  - receiver sends back a CTS to transmitter after SIFS (if ready to receive)
  - transmitter sends data at once after SIFS
  - receiver sends back an ACK after SIFS
    - this happens if packet was received correctly (Cyclic Redundancy Check – CRC)
    - automatic retransmission of packets in case of transmission errors
- other stations store medium reservations distributed via RTS and CTS

Written by Gabriel R.

Below, a visual comparison when sending unicast packets without CA – CSMA (left) and with – CSMA/CA (right), using RTS/CTS:



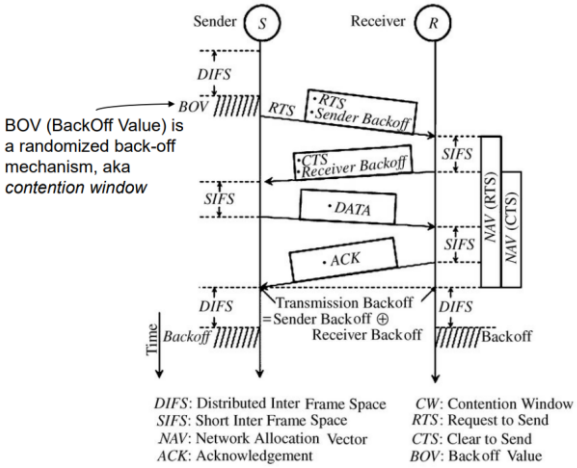
The figure inside slide of CSMA/CA poses a question: *What if we invert DIFS time with CW?*

- Stations would start contending for the medium immediately after sensing it
  - This could lead to increased contention and a higher likelihood of collisions
- After sensing the medium busy
  - stations would still need to wait for a free DIFS and then the random backoff value
- In a scenario where different contention window sizes are employed, this could impact fairness
  - See here for more about fairness

The CA phase starts after the DIFS phase, and it is larger when there are many nodes, smaller if fewer.

When a node fails to receive CTS in response to its RTS, it can act in many ways, further explained in other subsections.

The figure here details the actual data sending procedure.



So, to summarize a successful data transfer (A to B) consists of the following sequence of frames:

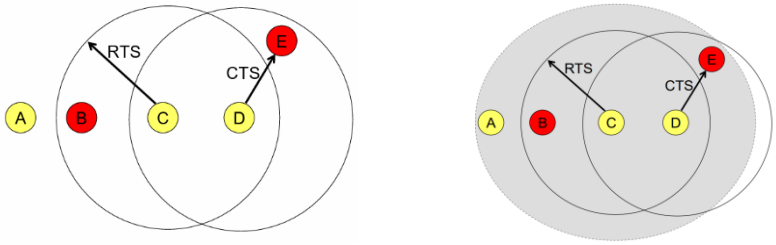
- 1. “Request To Send” frame (RTS) from A to B
- 2. “Clear To Send” frame (CTS) from B to A
- 3. “Clear To Send” frame (CTS) from B to A
- 4. “Data Sending” frame (DS) from A to B
- 5. “DATA” fragment frame from A to B
- 6. “Acknowledgment” frame (ACK) from B to A

The exposed terminal problem here occurs again even in the RTS/CTS situation (it’s solved only if stations are synchronized, and frame sizes/data speed are the same):

- B should be able to transmit to A (left figure)
  - RTS prevents this
    - reserves the channel even in case of not active transmission
- B should be able to transmit to A (right figure)
  - Carrier sensing makes the situation worse
    - felt like someone is transmitting
    - leads to unnecessary channel contention and delays

Written by Gabriel R.

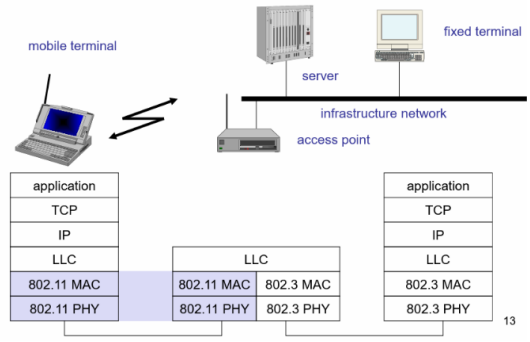
- While RTS/CTS could be helpful, it creates situations of no transmission
    - not solving the exposed terminal problem



In conclusion, some thoughts:

- *802.11 does not solve HT/ET problems completely (it mitigates them, does not solve them)*
    - It alleviates the problem through RTS/CTS and recommends a larger Carrier Sense (CS) zone
        - enlarging the detection, potentially meaning more nodes
- *Large CS (Carrier Sense) zone aggravates exposed terminals*
    - Spatial reuse is reduced, creating a tradeoff
    - RTS/CTS packets consume additional bandwidth
    - The backoff mechanism can also be wasteful of available channel time
- The search for the best MAC protocol is still on and 802.11 is being optimized too
- Thus, wireless MAC research still alive

In IEEE 802.11 we define as such the *device positioning*, making different layers communicate:



There are different versions of 802.11 protocols and for accessing to MAC layer, further explained in the next sections (considering DCF/PCF = Distributed/Point Control Function). Access methods are:

- MAC-DCF CSMA/CA (mandatory)
    - collision avoidance via randomized back-off mechanism
    - minimum distance between consecutive packets (interframe spacing)
    - ACK packet for acknowledgements (not for broadcasts)
- MAC-DCF CSMA/CA with RTS/CTS (optional)
    - thanks to RTS/CTS, makes a distributed handshake
    - reduces chances of hidden terminal problem
- MAC-PCF (optional)
    - access point polls terminals according to a list

Inside the MAC Layer we have *priorities*:

- they are defined through different inter frame spaces
- there are no guaranteed or hard priorities

Written by Gabriel R.

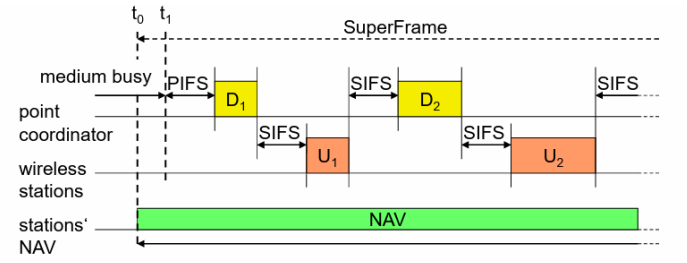
We use the same interframe spacings as before, but considering the different coordinations:

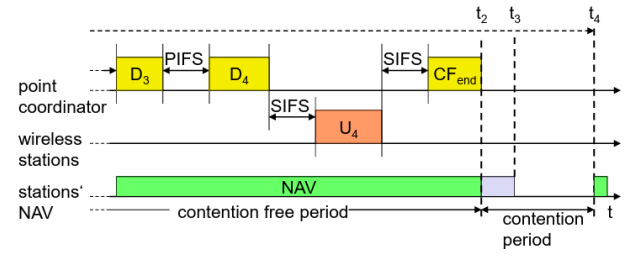
- SIFS (Short Inter Frame Spacing)
  - Highest priority, for ACK, CTS, data, polling response
- PIFS (Point Coordination Function Inter Frame Spacing)
  - Medium priority, for time-bounded service using PCF
    - hence AP has always more priority to access the medium
- DIFS (Distributed Coordination Function Inter Frame Spacing)
  - Lowest priority, for asynchronous data service
  - Time for waiting for the channel to be free

5.2.2 PCF - Point Coordination Function/Polling

MAC-PCF (Point Coordination Function) is a component of the IEEE 802.11 standard that provides a centralized method for controlling access to the wireless medium. Specifically (see also figures below):

- It allows a central coordinator (called *Point Coordinator*) to manage when devices can transmit
  - It is often the Access Point (AP), which has complete control over transmissions
- It asks to each node if there is something to transmit and it is managed by round-robin
  - There are no collisions and it's ok with many nodes
- The time is divided into *SuperFrames*
  - These are composed of multiple time slots, defined as PIFS
    - shorter than DIFS but longer than SIFS, which establish priorities hierarchically
  - Each SuperFrame begins at t_0 and ends at t_1, providing an organization framework
- It is particularly well-suited for networks that require prioritization of traffic
  - support for time-sensitive services, given its ability to minimize contention-related issues





It works concretely this way:

- AP announces whether it supports PCF in the beacon (small pieces of data)
- AP periodically broadcasts beacons and nodes use these to learn about APs
- The polled stations transmit and if they have nothing to send, then they must transmit null frames
- The node and the AP authenticate each other
  - The node associates with that AP
  - The node sends an Association Request Management Frame
    - Here the node announces to the AP if it is:
      - pollable
      - capable to transmit during the Contention Free Period (CFP)
  - The AP replies with an association response
- Note: PCF has more priority than DCF; so, a repetition interval has been designed to cover both

Some considerations about CSMA versions and the polling just discussed:

- *the exposed terminal problem is not improved*
    - with CSMA is even worse; you enlarge the detection, but potentially there are more nodes
    - it happens messages are sent simultaneously, so they may collide
- it is *always better to use RTS/CTS on CSMA/CA and polling* versions because:
    - even with more bandwidth consumed, the probability of collision is smaller
    - it's an improvement at the cost of limited transmission overhead
    - if data packets are very small (like VoIP or gaming)
        - CTS/RTS are not necessary and slow down the transmission creating overhead
- the positioning can deal with different MAC layers (802.11 – WLAN networks/802.3 – Ethernet)

5.2.3 Synchronization

Synchronization in 802.11 is useful for many things:

- APs send beacons in infrastructure networks
- *Timing Synchronization Function (TSF)* = keeps the timers of all BSS stations synchronized
    - in Power Management
        - beacons sent at well-known intervals (called beacon intervals)
            - transmission may be delayed by CSMA deferral
        - all station timers are synchronized
        - they send a timestamp, containing timer value at transmit time
    - in Coordination Timing
        - using a timer to predict the start of CF (Contention Free) burst
    - in Hop Timing at the physical layer
        - stations synchronized, the will hop at the same time

The power management approach allows idle stations to go sleep mode saving energy (saved into AP).

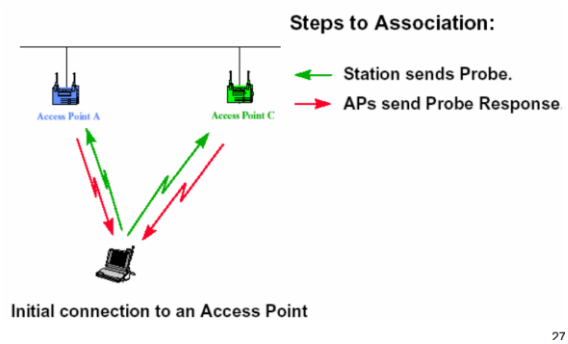
- it operates in infrastructure mode, where communication goes through an AP
- APs buffer packets for sleeping stations
    - AP announces which stations have frames buffered
    - Traffic Indication Map (TIM) sent with each Beacon
        - info on traffic load/channel state periodically sent with association info
- Messages are sent after TIM interval
- Power Saving Stations wake up periodically, listening for beacons
    - the energy a node consumes to wake up is more than the one used to stay asleep
        - sleep is intended for long times
- TSF assures AP and Power Save stations are synchronized (this is low-power)
    - stations will wake up to hear a Beacon
    - its timer will keep running when stations are sleeping
- Independent BSS also have Power Management
    - Basic Service Set = the most basic network building block
    - it typically includes one access point (AP) and one or more associated stations

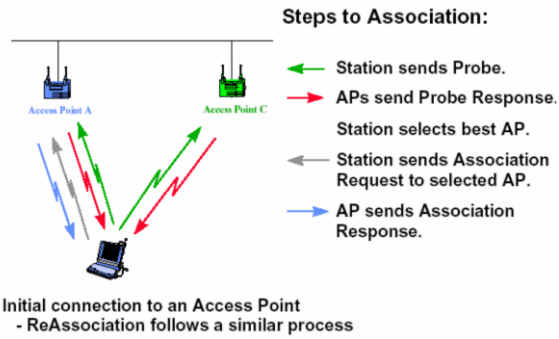
- Broadcast frames are also buffered in AP and sent only after DTIM (has more priority)
    - Delivery Traffic Indication Message (DTIM)
        - multiple of TIM
        - informs clients about presence of buffered multicast/broadcast data on AP
- Stations wake up prior to an expected DTIM
    - The higher the DTIM period, the longer a client device may sleep
    - therefore the more power a particular client device may save
- After a DTIM, the channel follows normal access rules (usually CSMA/CA)

The scanning phase involves:

- finding and joining networks, finding new APs while roaming or initialize independent BSSs
    - BSSs can either be with APs (infrastructure) or communicating directly (ad-hoc)
- MAC layer uses a common mechanism for all physical, given it can be:
    - *active* scanning
        - on each channel
            - send a probe → wait for a probe response
            - send an association request → wait for association request response
            - as shown below, reassociation follows the same principles
    - *passive* scanning
        - find networks simply by listening for beacons

The following is an active scanning example:





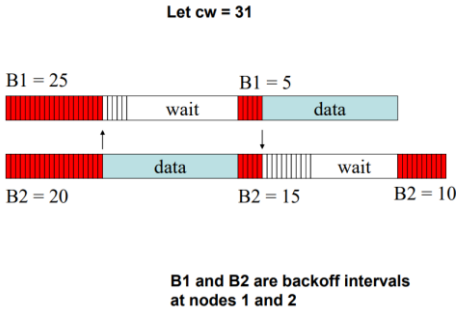
5.2.4 DCF - Distributed Coordination Function/Congestion Avoidance

Since the number of nodes attempting to transmit simultaneously may change with time, some mechanism to manage congestion is needed. This is obtained adjusting dynamically the contention window (cw).

In IEEE 802.11, Congestion Avoidance (DCF – Distributed Coordination Function) is achieved this way:

- Before transmitting a packet (after DIFS)
  - randomly choose a backoff interval in the range $[0, cw]$
- Count down the backoff interval when medium is idle (free)
  - count-down is suspended (goes standby) if medium becomes busy in range $[0, cw]$
  - when busy, it defers its transmission
- When backoff interval reaches 0
  - transmit packet (or RTS)
- When packet is received
  - send positive ACKs

You can briefly see for yourself the following example of DCF transmission to briefly apply the reasoning for the exam visually here on the right.



To realize *congestion control*:

- since the number of nodes attempting to transmit simultaneously may change with time
- it is achieved by *dynamically adjusting the contention window cw*

There is a *contention window tradeoff* (in choosing its *dimension*):

- the time spent counting down backoff intervals contributes to MAC overhead
  - choosing a *large cw* → large backoff intervals and can result in larger overhead
  - choosing a *small cw* → larger number of collisions
    - more likely that two nodes countdown to 0 simultaneously

So, in DCF as *binary exponential backoff* (uses feedback to adjust network conditions):

- when a node fails to receive CTS in response to its RTS
  - it increases the contention window
  - cw is doubled (up to an upper bound – maximum 5 times, max size is 1024)
- when a node successfully completes a data transfer
  - it restores cw to the original cw_{min}

There is a problem (and not a small one at that):

- the binary exponential back-off at times starves flows
- a probability of a node acquiring the channel may potentially decrease
  - when number of packets transmitted start increasing

5.2.5 MILD - Multiplicative Increase and Linear Decrease/Fairness Issue

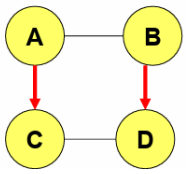
To prevent such large variations in the back-off values, a Multiplicative Increase and Linear Decrease (MILD) back-off mechanism is used inside MACAW (which, again, used cw to determine packet transmission rates).

- When a node fails to receive CTS in response to its RTS, it multiplies cw by 1.5
  - less aggressive than 802.11, which multiplies by 2
- When a node successfully completes a transfer, it reduces cw by 1
  - more conservative than 802.11, where cw is restored to cw_{min}
  - 802.11 reduces cw much faster than it increases it
  - MACAW: cw reduction slower than the increase
    - Exponential Increase Linear Decrease
  - MACAW can avoid wild oscillations of cw when congestion is high

Using this scheme (but also similar others like MIMD/AIAD, with same meaning in acronyms above), may not reach stability *or give anyone the proper time to talk* (a MAC goal, if you remember).

The fairness issue in networking refers to the challenge of distributing available bandwidth or network resources equitably among all participating nodes or devices.

- Many definitions possible
  - The simplest one: nodes should receive *equal bandwidth*
- Bandwidth should be tailored to *how much they want to transmit*
  - Otherwise, it would be a waste of time/bandwidth



Two flows

Consider this example about fairness issue (also plausibly explicable by above figure here):

- Assume that initially, A and B both choose a backoff interval in range $[0,31]$ but their RTSs collide
- Nodes A and B then choose from range $[0,63]$
  - Node A chooses 4 slots and B chooses 60 slots
  - After A transmits a packet, it next chooses from range $[0,31]$
  - It is possible that A may transmit several packets before B transmits its first packet

In this case, unfairness occurs *when one node has backed off much more than some other node*. That's where MACAW comes in, trying to solve the problem this way:

- When a node transmits a packet → it appends its current cw value to the packet
- All nodes hearing that cw value → will use it for their future transmission attempts
- Effect → reset all competing nodes to the same ground rule (level)

Specifically, there are two main ways to establish fairness:

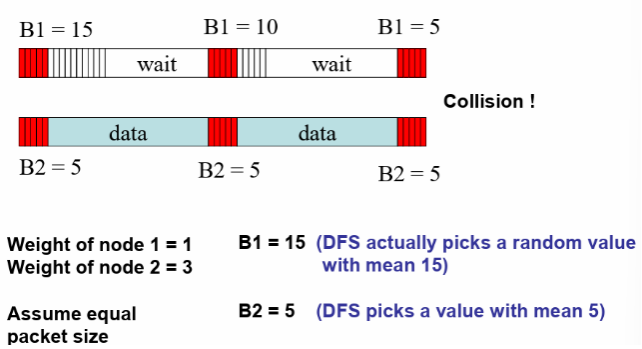
1) weighted fair queuing
   a. we *assign a weight to each node*
   b. *bandwidth* used by each one being *proportional to the weight* assigned to the node

2) Distributed Fair Scheduling (DFS)
   a. algorithm designed to address the concept of weighted fair queueing in networks
   b. fully distributed algorithm, meaning it operates in a decentralized manner

It works this way:

- It chooses backoff intervals proportional to *(packet size / weight)*
- It attempts to mimic the Self-Clocked Fair Queueing (SCFQ)
    - SCFQ assigns virtual time and credits to individual flows
    - It serves them in order of their virtual time
    - It adjusts them based on flow behavior, ensuring fair distribution

The following is an example of DFS for you to see:

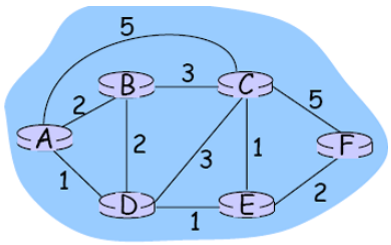


(This following part more for discussion/completeness because I like to give you context to things)

In the latest slides of this set (WNMA05), it's shown a tool called *inSSIDer*, which is a Wi-Fi network scanning and monitoring tool designed to help users analyze and optimize wireless networks.

- It is commonly used for monitoring Wi-Fi channels and improving network performance
- It scans the surrounding area for Wi-Fi networks
    - providing a list of detected networks
        - including SSIDs (Service Set Identifiers) = identifiers of Wi-Fi/WLANs
        - signal strengths, encryption types
    - this way we can gather detailed information
        - about signal strength, channel width, signal amplitudes and shapes
        - according to the frequency and their names
- It's present as an example to discuss about fairness
    - and how each concept presented here tries to solve problems their own way
    - each with pros and cons to consider properly

6 NETWORK LAYER (WNMA06)



Routing determines the "good" path (sequence of routers) through a network from source to destination. Usually, we cannot have a direct path to destination (in ad-hoc networks).

We use graph abstractions for routing algorithms (see left figure) where:

- nodes = routers
- edges = physical links
  - link cost: delay, cost or congestion level
  - the number in edges are infact costs
- goal = find a "good" path (*typical definition* = reaching minimum cost path)
  - it can be the shortest, the less expensive, the fastest – many definitions of "good"

Remember this one: *we can improve the definition of algorithms, not optimize them*.

In this case, we make classifications for the routing algorithms, starting from an information classification:

- Global
  - All routers have complete topology/link cost info
    - The algorithms need to have this info *before* making calculations
  - Examples: "*Link state"* algorithms
    - Routers exchange information about the state (cost) of directly connected links
    - Each router maintains a detailed map of the entire network
      - Including state and cost of each link
    - This information is then flooded throughout the network
      - Every router has an identical view of the network topology

- Decentralized
  - Router knows physically connected neighbors, link costs to neighbors
  - Iterative process of computation, exchange info with neighbors
    - the calculation of the least cost path is carried out iteratively/distributedly
  - Examples: "*Distance vector"* algorithms
    - They rely on each router maintaining a table
      - It contains info about distance (cost) to all other routers in the network
    - Routers periodically exchange their routing tables with their neighbors
      - Each router's table is then updated
        - based on the information received from neighbors

We also consider how algorithms themselves can be:

- Static
  - Routes change slowly over time
  - Topology is fixed all the time

- Dynamic
  - Routes change more quickly
  - They have periodic updates or respond to link changes

Written by Gabriel R.

Dijkstra's algorithm is a classic example of a link state routing algorithm.

- It is used to compute the least cost path from a source node to all other nodes in a network
  - the network's topology and link costs are known to all nodes
- It relies on link state broadcast
  - ensuring that all nodes have the identical and up-to-date info of the network's topology
- With this information
  - the algorithm calculates the shortest path
  - then creates a forwarding table for the source node

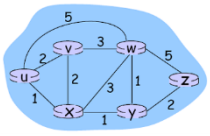
Here is its notation (left), the algorithm itself (middle), the table (right):

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

```
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'
```

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	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



6.1 MANET Problems

Consider a general comparison between infrastructure network vs MANETs:

- Infrastructure-based wireless network
  - Centralized = structure controlled via APs
  - Access points or base stations define cells or service areas
  - Simple routing
    - just a single hop from the AP to the wireless point

- MANET (Ad-hoc wireless network)
  - Decentralized = mobile devices communicate by themselves
  - There is no pre-defined or static network imposed by infrastructure
  - Wireless nodes are not necessarily all adjacent
    - a node may need to forward data for other nodes (i.e., participating in routing)

We make the same comparison considering wired vs wireless:

- Wired (Wireline / Infrastructure networks)
  - *Symmetric* links (bidirectional)
    - usually with respect to both capacity and quality
  - Limited planned *redundancy*
    - for reliability and load sharing
  - *Planned* links
    - typically of uniformly high quality (QoS) in a *fixed topology*

- Wireless (MANETs / Ad-hoc)
    - *Asymmetric* links (unidirectional)
    - High degree of *random redundancy* in connectivity between nodes
    - *Unplanned / dynamic* links (dynamic topology)
        - quality may vary due to interference, signal, etc.

Traditional routing algorithms in MANETs are inadequate because of:

- *Dynamic topology*
    - Frequent changes of connections, connection quality, participants

- *Limited/minor performance*
    - Periodic updates of routing tables need energy
    - Sleep modes are more difficult
    - Limited bandwidth
        - reduced even more due to exchange of routing information
    - Link can be *asymmetric*
        - info about link, quality of link (data link info), network updates (network layer info)
        - different costs depending on direction

- *Design problem*
    - Protocols have been designed for fixed networks
        - they have in mind infrequent changes
        - they typically assume symmetric links
    - Centralized approaches do not work well, and all nodes must be routers
    - Path length (hop count) is not a good metric for MANETs, long-lived circuits can't be used
    - MANET routing must rely on data link info, not just network updates
        - link layer determines connectivity and quality of links

- *Traditional routing algorithms have problems*
    - inefficient due to slow convergence times
    - non-functional due to large amounts of data or inability to deal with asymmetric links

Here are the goals for a good *unicast* protocol (transmission 1-to-1):

- *Minimal Control Overhead*
    - the protocol should use as little bandwidth as possible for routing/controlling network

- *Minimal Processing Overhead*
    - routing decisions should be made efficiently
    - nodes can focus on data forwarding

- *Multi-Hop Path Routing Capability*
    - essential when the destination is not directly reachable
    - data needs to be relayed through intermediate nodes

- *Dynamic Topology Maintenance*
    - should be able to handle node additions, removals, or changes in link quality

Written by Gabriel R.

- *No Loops*
    - routing should not form loops, which can lead to infinite data forwarding

- *Self-Starting*
    - it can operate effectively without extensive manual configuration
    - nodes should be able to join the network/do routing without complex setup procedures

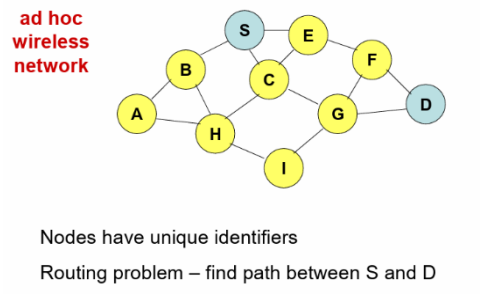
For completeness sake:

- Multicast: transmission 1 to N
    - *more than one receiver* of the same message, that it can follow different paths
- Broadcast: transmission 1 to N
    - *everyone* receives the message

In summary:

- Routing in MANETs is a tough problem
    - Unreliable wireless medium
    - Mobile nodes
    - No central authority
- Other issues
    - Traffic patterns application specific
    - Energy constraints

Consider this figure, describing a transmission example in which finding the path might be difficult:



6.2 ROUTING PROTOCOLS

Routing protocols can be classified as:

- *Proactive* (Table-driven)
    - Up-to-date routing information maintained
    - Routing overhead independent of route usage
    - Used for not-so-much dynamic networks
- *Reactive* (Demand-driven / Source-initiated / *On-demand*)
    - Routes maintained only for routes in use
        - This gives less overhead
    - Explicit route discovery mechanism
- *Hybrid*
    - Combination of proactive and reactive

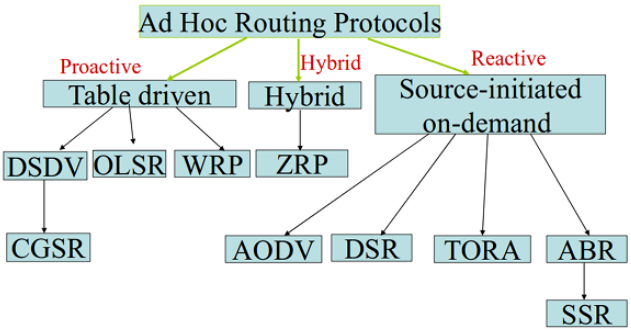
Specifically, we can characterize the approaches in the following subchapters.

6.2.1 Proactive Approach

We can describe the following features:

- Based on traditional distance-vector and link-state protocols
- Each node maintains route to each network edge
- Periodic and/or event triggered routing updates exchange
- Higher overhead in most scenarios because of continuous updates
    - Happens since each node maintains and updates its routing table
    - Mobility results in significant updates
        - Some nodes might never be used if network moves faster than route requests
        - Each change is immediately propagated to all the nodes
- Low latency (compared to reactive) of packet forwarding given the route is known
- Longer route convergence time
- Protocols differ on the method to propagate information and every node uses a unique ID
    - Examples (the ones in italic are the ones discussed here, others present just for context)
        - *DSDV* (Destination-Sequenced Distance Vector Routing)
        - WRP (Wireless Routing Protocol)
        - TBRPF (Topology dissemination Based on Reverse-Path forwarding)
        - *OLSR* (Optimized Link State Routing)

We tend to classificate protocols as follows in a convenient tree listing (we see only a few, no worries):



In table-driven/proactive routing protocols:

- Each node maintains an updated routing table
    - Contains routes to all nodes in the network
- Changes to network topology are immediately propagated
    - Tables need to be consistent
- Protocols differ in mechanisms used to propagate topology information

6.2.1.1 DSDV - Destination-Sequenced Distance Vector

DSDV is a proactive routing protocol used in ad-hoc mobile networks which main contributions *was to solve the routing loop problem*. It has the following features:

- It uses *sequence numbers* to ensure the freshness of routing information and prevent routing loops
    - Generally, they are even if a link exists, odd otherwise
    - The number is generated by the destination and the emitter sends out its number update
- Nodes *periodically exchange routing tables* to maintain up-to-date information

Written by Gabriel R.

- It is suitable for small to medium-sized networks
- Given it's based on Bellman-Ford, it supports negative weights
- Optimizations added to reduce routing overhead
    - Incremental data exchange, delayed exchange of updates

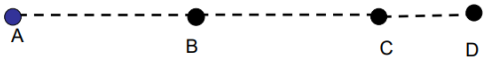
Let's describe more in detail how routing happens:

- Packets are transmitted according to the routing table
- Each node maintains routing table with entry for each node in the network
    - $< dest\_addr, dest\_seq\_num, next\_hop, hop\_count, install\_time >$
- Each node maintains its own sequence number
    - Updates at each change in neighborhood info
    - Used to avoid loops and to distinguish stale routes from new ones

In case of discussion for the exam, I'll leave a visual example to discuss:

Routing Table of Node A

Destination	Next	Metric	Seq. Nr
A	A	0	A-550
B	B	1	B-102
C	B	2	C-588
D	B	3	D-312



Route information is exchanged periodically

Consider how *updates* work here:

- Each node periodically transmits updates to keep table consistency
    - done with the inclusion of $< dest\_addr; dest\_seq\_num; hop\_count >$
    - lot of overhead
- Nodes also send routing table updates for important link changes (e.g. link broken)
- When two routes to a destination are received from two different neighbors:
    - choose the one with higher $dest\_seq\_num$
    - If equal, choose the one with smallest hop-count

Routing table updates create problems (lots of control traffic), so DSDV addresses this by using two types of *routing update packets*:

- *Full dumps*
    - Carry all routing table info (several NPDUs - Network Protocol Data Units)
        - NPDU = Packet of data which may contain either signalling or user information
    - Transmitted relatively infrequently
- *Incremental updates*
    - Carry only info since last full dump
    - First with one NPDU
    - When updates no longer fit one NPDU → send full dump

Pros:

- The availability of paths to all destinations in network makes less delay required in path set up
- Incremental updates are definitely more suitable for ad-hoc networks

Cons:

- It requires regular update of its routing tables, using battery power and a small bandwidth amount
- When topology changes, a new sequence number is necessary before network re-converges
- DSDV is not suitable for highly dynamic/large networks

While DSDV itself does not appear to be much used today, other protocols have used similar techniques, for example AODV, presented in next subsections.

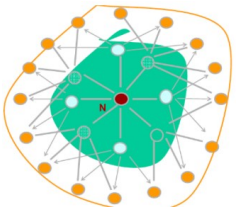
6.2.1.2 OLSR - Optimized Link State Routing

OLSR is a proactive routing protocol that optimizes the traditional link-state approach reducing control overhead by computing and maintaining multiple routes in a single update message, disseminating link state information throughout the entire mobile ad-hoc network.

It has the following features:

- It uses hello and topology control (TC) messages to discover/disseminate link state info
- Better suited for large and dense networks
    - Best when traffic is random
- All links with neighboring Mobile Hosts (MHs) are declared and flooded in entire network
- *Minimizes flooding in control traffic* by using only selected MHs
    - Previous protocols elect only a designated routers: OSPF/IS-IS (to give context to you this)
- Only normal periodic control messages are sent
- It uses traffic patterns
- In-order delivery of its messages is not needed, as each control message has a sequence number

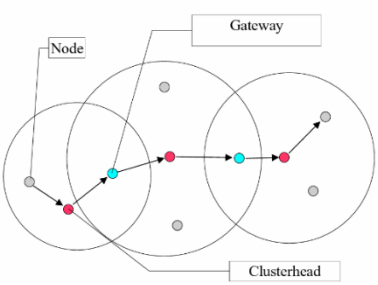
The key concept is *the usage of Multipoint Relays (MPR)*:



- They minimize flooding of broadcast packets reducing overhead
- Each MH selects a set of neighboring MHs to retransmit its packets
    - reduces duplicate retransmissions in the same region
- This set can change over time
    - it's indicated by the selector nodes in the Hello messages
- Typically, each one is selected as neighbor at 1 hop (hop-by-hop routing)

6.2.1.3 CGSR and SIODR

- *Clusterhead Gateway Switch Routing (CGSR)*
    - Nodes organized into hierarchy of clusters
    - Each node has a *clusterhead*, selected using an election
    - Nodes send packets through clusterheads
        - Cluster higher to cluster lower = hierarchy
    - Gateway nodes carry out transmission/reception
    - Clusterheads communicate using DSDV
        - Clusters are connected through a gateway node



Written by Gabriel R.

- *Source-Initiated On-Demand Routing (SIODR)*
    - Create routes *only when needed* by source node
        - Maintained until
            - route is broken
            - or the source does not need it anymore
    - Routes found using a “route discovery” process
        - Flooding
    - Route maintenance procedure used to repair route
    - Used especially in mobile wireless sensor networks (MWSNs)

Now, let’s discuss the other possible approach in next subchapter.

6.2.2 Reactive Approach

We can describe the following features:

- Source builds routing on demand by “flooding”
    - Nodes receive and broadcast again
    - Route Discovery cycle
- Maintain *only active routes*
- *Pros*
    - less control overhead
    - better scaling properties
- *Cons*
    - route acquisition latency
    - long delay in finding the route

6.2.2.1 AODV - Ad-Hoc On-Demand Distance Vector

AODV is a reactive routing protocol which has the following features:

- It operates based on the principles of the “on-demand” approach:
    - start a route discovery process when a destination does not have a route
    - establish one via a series of “route request” and “route reply” to packets
- It is based on DSDV, identified by RFC 3561 and known for its efficiency in route discovery
    - So, it uses destination sequence numbers ($dest\_seq\_num$) to avoid loops
    - They grow monotonically over time to ensure no loops in the paths used
- The network remains completely silent until a connection is required to forward a data packet
- Routing table exchanges only happen along a given route with the packets presented below

This is the *route-finding* process (set the path, then we will send messages):

- Source broadcasts in entire network Route Request Packet (RREQ) when a route must be found
    - RREQ contains information about
        - the source, destination, a Time To Live (TTL) and a unique identifier for the request
    - Once an intermediate node receives a RREQ
        - the node sets up a reverse route entry for the source node in its route table
- Route discovery floods all nodes with request
    - A node responds the first time it receives a request

    - Replies only if it has contact with the destination, or a valid route to the destination
- Route Reply Packet (RREP) is sent back by destination
    - RREP contains the route information from the destination to the source
        - type and hop count
        - destination IP address ($dest\_IP\_add$)
        - destination sequence number ($dest\_seq\_num$)
        - lifetime ($life\_time$)
    - If this is true, node responds using unicasting via reverse path
    - Route Error (RERR) messages update routes
        - Used to feedback a path was interrupted
        - This tells other nodes the route is no longer reachable
- If routes are not used they expire and get discarded
    - They are timed
        - after a while, they expire (kept until a timeout happens)
    - Reduces stale/obsolete routes and does not require explicit route maintenance

It has some properties (tracing previous ones/adding more context):

- AODV discovers routes when necessary
    - it does not maintain routes from every node to every other one
        - they are maintained just as long as necessary
- Every node maintains its monotonically increasing sequence number
    - increases happens when node notices changes in the neighborhood topology
- AODV utilizes routing tables to store routing information:
    - a routing table for unicast routes → path from source to destination
    - a routing table for multicast routes → flooding
        - if a node receives it all once → collision and congestion will happen
- The routing table stores:
    - $< destination\_add, next\_hop\_address, dest\_seq\_num, life\_time >$
    - this is updated each time a route is used
- For each destination, we have a list of *precursor nodes*, to route through them
    - they are useful for route maintenance
- Life-time updated every time a route is used
    - if route is not used within it → it expires

Following here, how the *route discovery* process works (aka “once set the path, we can send”):

- When a node wishes to send a packet to a destination
    - It checks its routing table to determine if it has current route to the destination
    - if “yes” → forwards packet to next node
    - if “no” → it initiates a *route discovery* process
- Source node creates a new Route Request (RREQ) packet
    - the packet contains:
        - source node IP address ($source\_IP\_add$)
        - source node sequence number ($source\_curr\_seq\_num$)
        - destination IP address ($dest\_IP\_add$)
        - destination sequence number ($dest\_seq\_num$)

        - broadcast ID number ($broad\_id$)
            - this is incremented each time a source code uses RREQ
            - broadcast ID number + destination IP address = unique ID for RREQ
        - a Time To Live (TTL) limiting the number of retransmissions
        - its type and hop count
- Broadcasting is done via flooding
- Once an intermediate node receives a RREQ
    - the node sets up a reverse route entry for the source node in its route table:
        - the reverse route entry contains:
            - source node IP address ($source\_IP\_add$)
            - source sequence number ($source\_seq\_num$)
            - number of hops to the source ($num\_hops\_to\_source$)
            - predecessor IP address ($pred\_IP\_add$)
            - lifetime ($life\_time$)
        - an intermediate node may also send a Route Reply (RREP)
            - they do it using reverse route
                - if it knows a more recent path (newer $dest\_seq\_num$)
                - not used so much, because new RREQ → new $dest\_seq\_num$
                - intermediate $dest\_seq\_num < new\ dest\_seq\_num$
                    - it can't send RREP
- RREQ reaches destination → To respond to RREQ, a node should have it in its own route table:
    - Unexpired entry for destination
    - Sequence number of destination
    - Checks couple $dest\_seq\_num/dest\_IP\_addr$ and checks if was present already/if it's new
    - If both parts are met
        - the node responds with RREP using unicasting and reverse path
    - If those are not satisfied
        - node increments hop count in RREQ and floods to neighbors
- AODV implements a *binary backoff mechanism*:
    - in case the node does not receive a response to its RREQ
    - whereby requests are repeated at linearly increasing time intervals
        - up to a maximum set by the implementation

There may happen timeouts inside the route discovery process:

- Routing table entry keeping a reverse path is deleted
    - after a timeout → it should be long enough to allow RREP to come back
    - if it is not used for $active\_route\_timeout$ interval
        - if no data is being sent → that entry will be deleted even if it is valid

Here also how the Link Failure is detected:

- neighboring nodes periodically exchange "Hello" messages
    - such messages are not mandatory, there could be other ways
- if there is absence of messages → link failure
- several MAC level ACKs missing → link failure

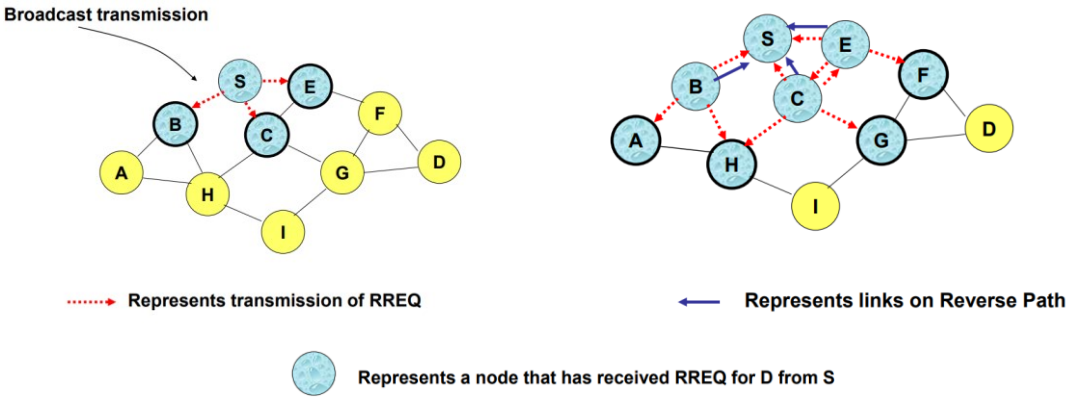
Consider the following optimizations over AODV:

- RREQ are sent with a TTL (Time To Live) with hop count → limits flooding propagation
- if no RREP received → larger TTL
- Expanding Ring Search
    - it prevents flooding of network during route discovery
    - it controls TTL of RREQ to search incrementally larger areas
    - Advantages
        - less overhead when successful
        - does not generate traffic in case of established and working routes
        - the algorithm is irrelevant as long as there are no unknown routes
            - computationally feasible
    - Disadvantages
        - longer delay if route isn't found immediately:
            - here could be many possible destinations
            - it would be better without optimization
            - maybe it doesn't find the optimal path

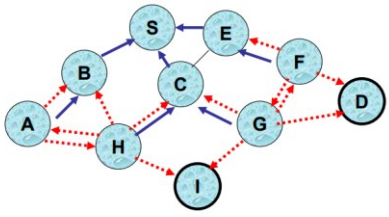
A comparison table here between DSDV and AODV for your reference:

DSDV	AODV
• every change is broadcasted	• no broadcasting necessary
• new neighbours link ⇒ news is broadcasted	• only affected nodes are informed
• new broken link ⇒ news is broadcasted	• new broken link ⇒ no global broadcasting
• more control overhead (high)	• less control overhead (low)
• local movements ⇒ global effects	• local movements ⇒ local effects

Below the entire process for the sake of completeness of AODV. First, we start considering the route requests process, making the broadcast and: creating the reverse path as described:



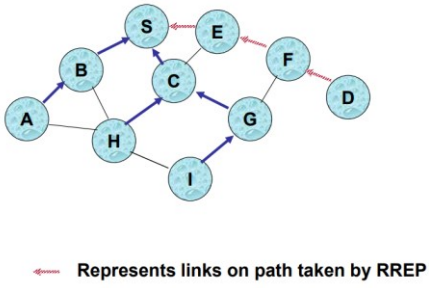
The reverse path is setup incrementally, so to not forward again already forwarded requests:



• Node C receives RREQ from G and H, but does not forward it again, because node C has already forwarded RREQ once

Written by Gabriel R.

The route reply consider paths already taken as links:



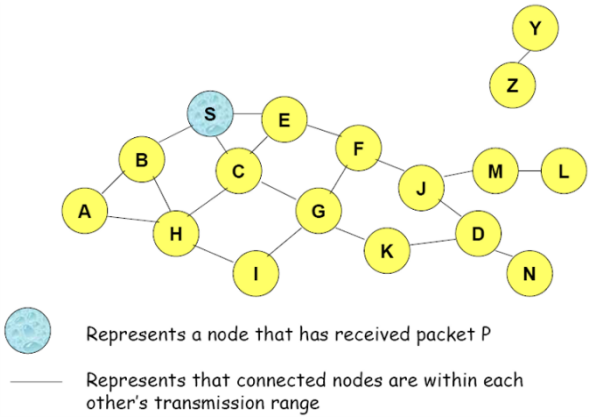
6.2.2.2 Flooding for Control Packet Delivery

Flooding is a technique which can be used for *control packet delivery*, ensuring network is reliable and there is a timely delivery despite the presence of flooding

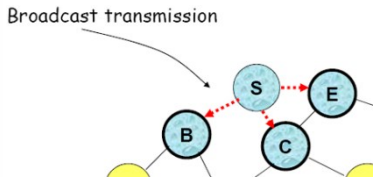
- *Control packets* are special ones used to manage many aspects of network communication
  - they are sent decentralized to ensure coverage and robustness
- In other words, when a control packet needs to be delivered to various destinations in a network
  - it is broadcasted or multicasted to all nodes within the network
  - each node that receives the control packet forwards it to all its neighboring nodes

Below there is a specific example, in which flooding reaches destination and packets stop there, even after using sequence numbers to avoid loops.

- Sender S broadcasts a control packet P to all its neighbors
- Each node receiving P forwards P to its neighbors
- Sequence numbers help to avoid the possibility of forwarding the same packet more than once
- Packet P reaches destination D provided that D is reachable from sender S
- Node D does not forward the packet

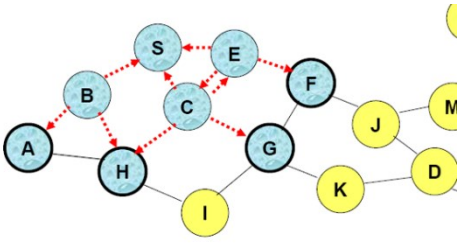


We then make a broadcast transmission to all linked neighbors:



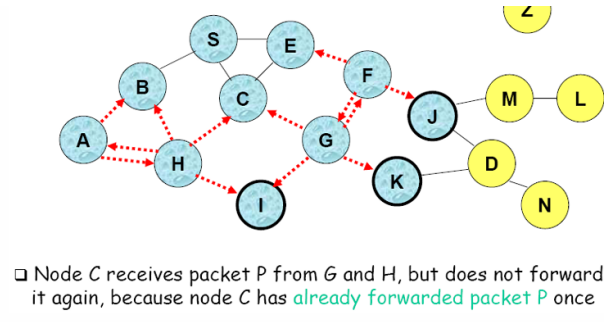
Neighbor nodes propagate messages to their neighbors via broadcast, but collisions can occur when multiple messages are received simultaneously.

An error scenario can occur when multiple senders attempt to reach the same destination, or when certain areas of the network have not yet been covered.



- Node H receives packet P from two neighbors: potential for collision

Written by Gabriel R.



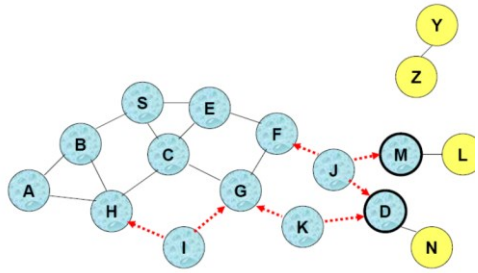
❑ Node C receives packet P from G and H, but does not forward it again, because node C has already forwarded packet P once

In broadcast transmission, there is no acknowledgment or retransmission mechanism.

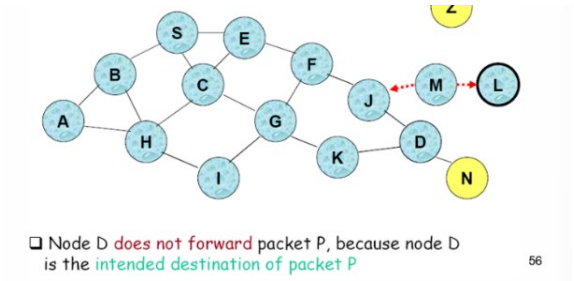
Consequently, there can be situations where previously forwarded packets are not retransmitted, leading to potential issues, as you can see here.

Problems arise overtime, between multiple nodes, which might not see each other and transmit colliding (hidden) or think the channel may be busy and don't transmit (exposed).

Given the transmission has no central control, problems arising locally can actually become huge overtime.



❑ Nodes J and K both broadcast packet P to node D
❑ Since nodes J and K are hidden from each other, their transmissions may collide
=> Packet P may not be delivered to node D at all, despite the use of flooding



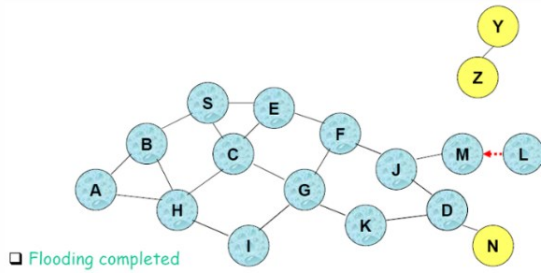
❑ Node D does not forward packet P, because node D is the intended destination of packet P

56

These problems interfere with each other, so you might have nodes not reaching the intended destination.

When flooding is completed, if there are nodes not reachable from source or nodes going through the destination because of problems presented above, the packet will not be received (the yellow ones below).

The opposite problem can happen: flooding delivering packets to too many nodes (and all reachable nodes receive the sender message).



❑ Flooding completed
❑ Nodes unreachable from S do not receive packet P (e.g., node Z)
❑ Nodes for which paths go through the destination D also do not receive packet P (example: node N)

Flooding has few *advantages*:

- Simple and decentralized
  - with robustness in traffic and loop prevention
- May be more efficient than other protocols
  - this happens when rate of information transmission is low enough
  - overhead of explicit route discovery and maintenance incurred by others is higher
  - this scenario may occur when nodes transmit *small data packets* infrequently
    - many topology *changes occur* between consecutive packet transmissions
- Potentially higher reliability of data delivery
  - because packets may be delivered to the destination on multiple paths

Written by Gabriel R.

But it also has a few *disadvantages*:

- Potentially very high overhead
  - Data packets may be delivered to too many nodes who do not need to receive them
  - Infinite number of duplicate packets and useless in case of single destinations
- Potentially lower reliability of data delivery
  - Flooding uses broadcasting
    - hard to implement reliable broadcast delivery without increasing overhead
  - Broadcasting in IEEE 802.11 MAC is unreliable
- It's not rare nodes may transmit to a same node simultaneously (hidden terminal problem)
  - This results in packet loss → destination would not receive the packet at all
- Limited scalability in large networks and lack of selectivity

6.2.2.3 DSR - Dynamic Source Routing

DSR is a proactive routing protocol, which either*:*

- *continuously maintains up-to-date routing information*
- *initiates route discovery when needed* (similarly to AODV)
- this happens *from source (source routing):* sender partially/completely specify route packet takes

Primarily:

- it does not use destination sequence numbers or reverse path RREQ inside nodes

It is known for its *flexibility and adaptability* to changing network topologies with the following features:

- Each data packet *has full source route* and provokes overhead
  - RREQs have attached full source-route and is sent back in RREPs
- Each node maintains caches so to help route discovery

The route discovery is similar to AODV and goes on like this:

- Sender initiates request
- Intermediate nodes add their address onto request
  - uses RREQ and understands if it was already seen/if it is new
  - with asymmetric links, if there is no path, useful to send RREQ to send RREP later
  - hop limit added to RREQ to avoid flooding countereffects
  - proportional waiting added to RREP to avoid route reply storm
- When request reaches destination, it includes the full path

DSR allows automatic path shortening in case a node receiving RREQ realizes it has a better path in memory. A comparison table between DSR and AODV:

DSR	AODV
• source route in packet headers ⇒ especially for small packets	• only predecessor on each packet
• routing table maintained only on source	• routing tables maintained also on nodes
• lifetime for routes discovery	• no lifetime for routes discovery

The goal of this protocol is to avoid wasting bandwidth consumed by control packets and, when compared to other on-demand protocols, it is beacon-less, so it does not require hello packet transmissions. Periodical flooding is useful, but route maintenance does not locally repair broken links.

6.2.2.4 Other Routing Protocols and Metrics

Associativity-Based Routing (ABR) defines a metric to determine the best route in mobile ad-hoc networks and wireless mesh networks. It is based on these points:

- it uses *degree of association stability* as a metric instead of number of hops
    - this information is appended into the packet itself
- nodes with less mobility/better links have higher stability value
- it's a DSR-like protocol for routing and was implemented inside Linux in various forms

Signal Stability Routing (SSR) employs *signal strength* as its primary metric for making routing decision.

- the signal strength of links between nodes determines the quality
    - The stronger the signal, the better the link is considered
    - This is employed inside a so-called Signal Strength Table (SST)
- DSR-like routing is used with same principles as before
- RREQ packets are forwarded only if the packet is received with a sufficiently strong signal
- Stronger in path formation, longer delays in path creation

Other metrics are:

- *Expected Transmission Time (ETT)*
    - Easier to compute, and more useful than signal strength
    - It considers factors such as transmission rate, packet size, and retransmission attempts,
        - this better predicts how long it will take for data to traverse a particular link
- *Weighted Cumulative Expected Transmission Time (WCETT)*
    - Better for multi-radio, and asymmetric rate links with unbalanced data rates
    - In such networks, nodes may have access to different radios with varying capabilities
        - e.g. different upload and download speeds

Meanwhile, other routing protocols are:

- *Geographic Routing Protocols*
    - They use physical locations or geographic coordinates to make forwarding decisions
    - These protocols typically rely on the knowledge of the positions of network nodes to determine the next hop for packet forwarding

- Consider:
    - *Location-Aided Routing (LAR)*
        - Nodes in the network determine *their geographic coordinates*
            - usually using GPS or other localization methods
            - they use this information to route packets
        - using *regions for destinations and regions to make requests*
    - *Distance Routing Effect Algorithm for Mobility (DREAM)*
        - It considers the relative movement of nodes in the network
        - It calculates the effect of mobility on routing decisions

- *Greedy Perimeter Stateless Routing (GPSR)*
  - Based on stateless routing, *makes forwarding decisions selecting only neighbor* nodes based on geographic proximity (destination/neighbours)
  - When a forwarding node cannot reach the destination directly
    - it circumvents obstacles in a greedy manner
    - *it routes around the perimeter of said region*

There are also:

- *Hybrid Routing Protocols*
  - they combine *both proactive (table-driven) and reactive (on-demand) routing strategies*
  - balance between:
    - maintaining route information continuously
    - initiating route discovery only when needed
    - aiming to optimize routing efficiency and adaptability
      - with respect to latency in discovery and control messages
- An example between the hybrid ones is:
  - *Zone Routing Protocol (ZRP)*
    - it divides the network into *routing zones*:
      - proactive routing is used within each zone
      - reactive routing is employed between zones
    - ZRP strikes a balance between the advantages of both routing types
      - offering improved adaptability and reduced control overhead
      - proactive = neighborhood search/reactive = routing across network

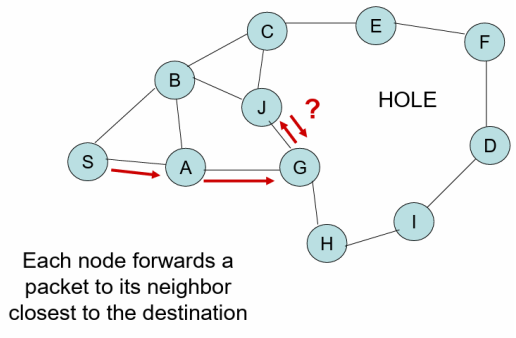
From the discussion given so far, a few key points:

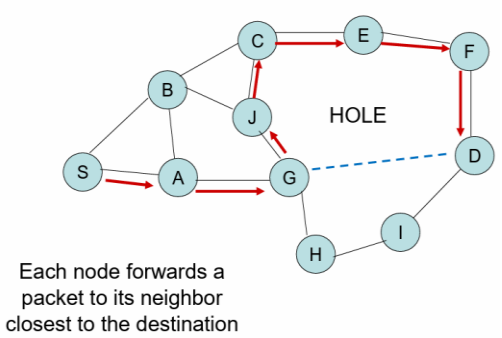
- Proactive routing protocols suitable for high traffic load, low mobility
- On-demand routing protocols suitable for low traffic load and/or moderate mobility
- With high mobility, flooding of data packets may be the only option

In these networks, the notion of *local minima* arises (which means two nodes keep exchanging the same packet). We consider the case of *sensor networks:*

- here we want to locate and bypass *routing holes*
  - areas/regions where there is no efficient or direct path for data packets to reach their intended destination
- so, each node forwards a packet to its neighbor closest to the destination

Given the dynamic nature of the network (also limited communications), we need to find a proper way to solve imbalance and optimize the packet problem (still in research). Here you see the holes.





Written by Gabriel R.

Considering again Greedy Perimeter Stateless Routing (GPSR), we might find a solution to this problem:

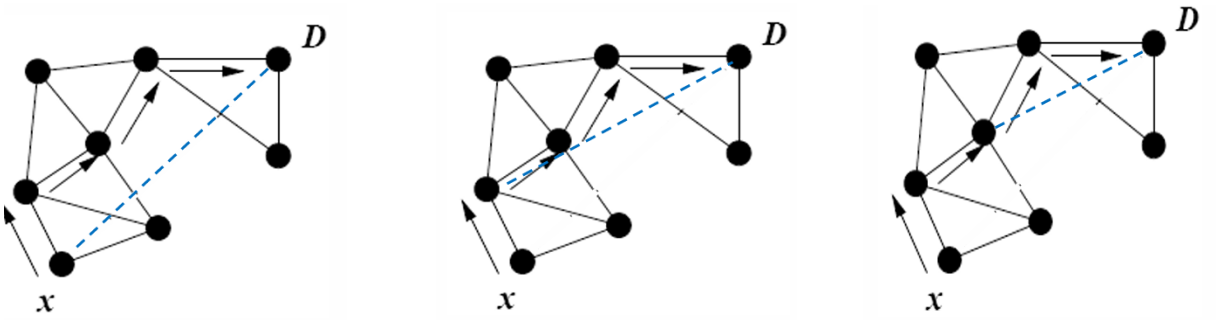
- Location of the destination node is assumed to be known
- Each node knows location of its neighbors
- Each node forwards a packet to its neighbor closest to the destination
- If routing holes are found, uses perimeter routing (*right-hand rule*)

Here, a *right-hand rule* is employed (like the sum of vector in physics, with the first 3 fingers); basically, there is an imaginary line from source to destination and the packet is sent to the first reachable node counterclockwise.

Consider this example; imagine a network with several nodes attempting to forward a packet to a known destination. When a routing hole is encountered, the Right-Hand Rule is employed as follows:

- Node A is trying to forward the packet to the destination (D) but encounters a routing hole
- Node A identifies its counterclockwise neighbor
    - concerning the line to the destination (D) and forwards the packet to this neighbor
- The process continues
    - as each subsequent node, guided by the Right-Hand Rule
    - forwards the packet to the counterclockwise neighbor
    - until it reaches the destination or finds a route to exit the routing hole.

The figures try to summarize this process.



7 TRANSPORT LAYER (WNMA07 – WNMA08)

The *transport layer* allows to set and end-to-end connection between the sender and the receiver. It sends data of application layer as fast as it decides, possibly not to create congestion. It is the last layer unpacked from the layers' stack and it's used to know how to send packets, dealing with everything derived from it.

TCP is the most widely used Internet protocol, serving various applications such as web browsing, peer-to-peer file sharing, FTP (File Transfer Protocol), and TELNET. A few key points:

- It's *end-to-end*
    - software only at the end hosts

- It's *full duplex*
    - transmission working two-way, i.e. it can perform roles of both receiver and sender

- It's *reliable*
    - ensures that all data is fully transmitted

- It's *correct* (realizing the previous point, so *reliability*)
    - it uses a *checksum* to detect bit level errors
    - it uses *sequence numbers* to detect sequencing errors
    - it uses *acknowledgements* (ACKs) to be sure messages were received

- It's *byte-stream oriented*
    - the sender writes bytes
    - the receiver reads bytes

- Data is organized into *segments*, which can be reassembled
    - usually $1500\ B / 1.5\ KB$

- It has:
    - *Flow control*
        - Mechanism to manage data transmission rate between sender and receiver
        - It prevents the sender from overwhelming the receiver with data
            - especially when the receiver's buffer or processing capacity is limited
        - Keeps sender from overrunning receiver

    - *Congestion control*
        - Mechanism preventing congestion by regulating data rate sent into the network
        - Ensures network operates efficiently without becoming overloaded
            - e.g., decreasing sending rate overtime
        - Keep sender from overrunning network

- It's *connection-oriented*
    - It establishes a connection (handshake) between sender/receiver before data transfer

- TCP is *not appropriate for current wireless scenarios* (but there are many variants to do so)

Written by Gabriel R.

7.1 Reliability and Timeout Recovery

Reliability in TCP uses:

- *checksum*
  - to detect bit level errors
- *sequence numbers*
  - to detect sequencing errors
    - duplicate packets are discarded
    - packets can be reordered
    - lost packets can be retransmitted
- *timeouts* to understand if packets were lost (if data not ACKed when timer runs out, timeout)
  - Requires RTO (Retransmission Timeout) calculation
    - occurs when the sender is missing too many ACKs
    - and decides to take a time out, stop sending altogether
  - Requires sender to maintain data until it is ACKed

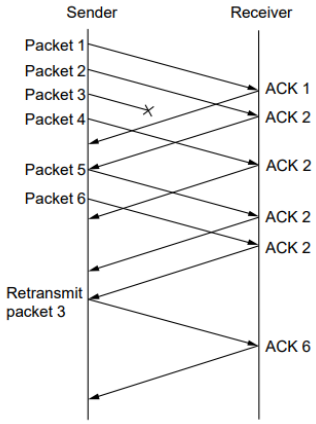
Timeout-based recovery in TCP involves:

- waiting for at least one *Round Trip Time (RTT)* before retransmitting data
- sender will maintain data until it is acked
- it estimates RTT for packets sent and sets RTO based on this. Estimates change:
  - low RTO → unneeded retransmissions
  - high RTO → poor throughput
- RTO estimator must adapt to change in RTT
  - not too fast/not too slow (because each ACK happens in 1 RTT)
  - *about* 4 *times* the RTT ($RTO = 4 * RTT$) – so to safely assume packet was lost
- Note: $RTO > RTT$, otherwise spurious timeouts and RTO needs to adapt dynamically

7.2 Fast Recovery, Fast Retransmission, Congestion Window

To control congestion, it's important to properly handle and respond to *duplicate acknowledgments (dupacks)* for the same segment (if you receive a packet correctly, you get an ACK, in order until packet x).

- Duplicate acks (same ACK received 2 times) happen when:
  - Packet loss
  - Packet reordering
  - Window update – advertisement of new control flow window
  - In general, packets out of order and used as feedback



- *Problem*: coarse-grain TCP timeouts lead to idle periods
  - Idea: Use receipt of 3 or more dupacks as indication of loss
  - Avoid waiting for timeout to retransmit

- Fast Retransmit: use 3 duplicate ACKs (dupacks) to retransmit
  - Don't wait for timeout = send lost packet without waiting timer
  - Instead of waiting $4 * RTT$, resend after $1 * RTT$
    - When a dupack arrives, sender assumers it was packet reordering

Written by Gabriel R.

- Sender will conclude soon it was most likely lost and will send the packet again

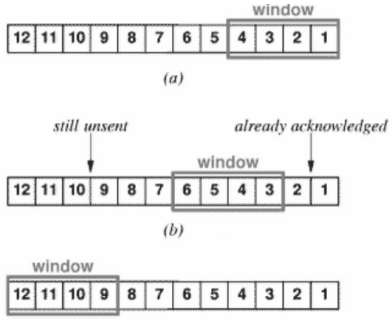
- Fast Recovery: reduce $cwnd$ size to half of where you were (avoiding going back to 1); in other words, start at a threshold (see later which) and do additive increase after Fast Retransmit
  - TCP normally increases packets sent by 1 every RTT
  - *Normally*: every time a packet is lost, it restarts form 1
  - *Here, instead of going back to* 1*, we start at* $ssthresh$ (Slow Start Threshold)
    - a threshold usually put at about $\frac{1}{2}$ of where we were
  - Specifically:
    - When packet losses are detected, we enter this phase instead of going slow start
    - Upon detecting it, congestion window (see below) is cut in half
    - Packets are sent slower, and we wait for dupacks to arrive
    - While dupacks arrive, missing packets are retransmitted
      - Contention window is gradually increased returning to the original one
  - Note: even using this one, having packets lost does not mean anything
  - It would not make sense to slow down restarting from $\frac{1}{2}$, just resend the packet

To allow reliable data mechanism in transferring data, TCP uses *sliding window flow control*:

- sliding windows enable the sender to keep sending data without waiting for acknowledgments for each individual packet
  - more flexible than Stop & Go
    - in this, the sender periodically pauses to allow receiver to catch up with loss of data

- this concept leads to the congestion window ($cwnd$), used to keep sender overrunning the network sending too many packets (it is continuously computed by the sender). It is composed by:
  - the *window* itself
    - maximum number of packets that can be transmitted before receiving their ACKs
    - any time an ACK is received → the window moves ("slides") forward
      - this indicates that new packets can be transmitted
    - if some packets are lost, the window:
      - either returns to 1 (equivalent of one segment)
      - or goes back to the $ssthresh$ (depends on TCP version used)
    - increasing the window means making the transmission faster
  - the data to send
    - is divided into packets which are often referred to as *segments*
      - these have a standard size, often around 1500 B

For more, see the following Congestion Window schema describing what was just said:



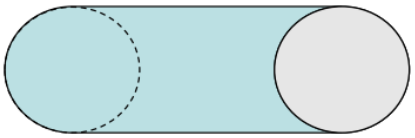
Written by Gabriel R.

7.3 FLOW CONTROL

TCP Flow Control prevents the sender from overwhelming the receiver by ensuring that data is transmitted at a rate that the receiver can handle. Specifically, we can define both sides of transmission:

- *Receiving Side*
  - The Advertised Window ($awnd$) is set by the receiver = max number of bytes it can receive
    - based on its buffer's remaining capacity (in returning ACKs) – space left in buffer
  - It reflects the maximum number of bytes that can be received
    - without overloading the receiver's buffer (how much space is left)
    - this is decided by receiver and sent with every ACK message
- *Sending Side*
  - The Sending Window ($swnd$) is sender's actual window = actual bytes that have been sent
  - It corresponds to the minimum between:
    - the sender's computed window (*Congestion Window* = $cwnd$)
    - the receiver's computed window (*Advertised Window* = $awnd$)
  - This ensures the sender doesn't transmit data faster than the receiver can handle

We can think of the network channel as of a tube and our objective is to keep it full at any time (to avoid wasting bandwidth).
The concept of *Delay-Bandwidth Product* represents the max number of data that can be traveling on the link at any time.



It's a link between sender and receiver seen as a pipe of length where:

- Delay → time passed from sender to receiver (known as ping)
  - RTT is twice the Delay
- Bandwidth (or section) → how much data is sent simultaneously on the channel
  - Bandwidth is distributed like this:
    - half the traffic is travelling
    - half reached the receiver and is sending ACKs back
      - ACKs have small/negligible size with respect to data packets
  - By multiplying the bandwidth by the delay ($delay = \frac{RTT}{2}$)
    - we get how much data we can have in a connection without clogging it
- When using TCP
  - to get the size of the congestion window we calculate $Bandwidth * RTT$
    - because we also have to consider ACKs, which are very small and can be ignored

7.4 CONGESTION CONTROL

The Congestion Control blocks the sender from overwhelming the network, limiting the flow of packets from each node checking end-to-end (with no feedback from network). It has the following ideas:

- assumes best-effort network (FIFO routers)
  - each source determines network capacity for itself
  - uses *implicit feedback*
    - the network does not explicitly tell senders about congestion
    - can be inferred by behavior e.g. missing/delayed/duplicate ACKs

Written by Gabriel R.

    - ACKs pace transmission (self-clocking)
        - sender regulates according to ACKs

TCP comes from challenges by its original design, which was *not created with wireless networks in mind*:

- *Determining the available capacity* in the first place
    - because of limited feedback and dynamic nature of networks
- *Adjusting to changes* in the available capacity
    - in reaction time, fairness, efficiency
- *Congestion control mechanisms need adapt in real-time*
    - Back at TCP invention, there was no clue about wireless and consequent loss of data
    - It was thought any loss in wired links would have been caused by congestion (no error loss)

7.4.1 AIMD - Additive Increase and Multiplicative Decrease

That's why we use *feedback algorithms*: adjust changes in the available capacity (congestion control). To distinguish between the two, we use a variable called $ssthresh$ (slow start threshold) – see later for more.

Additive Increase and Multiplicative Decrease (AIMD) is a specific strategy used in some congestion control algorithms, particularly in TCP. The idea is:

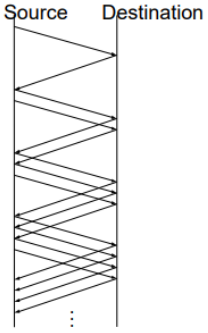
- Increase the Congestion Window when congestion in the network decreases:
    - adjusting changes in the available capacity
    - adjusting the sender's transmission rate
        - it allows the sender to gradually send more data when network is not congested
- The Congestion Window is introduced as variable: $cwnd$
    - this limits how much data source has in transit
- $swnd = min(cwnd, awnd)$
    - increase $cwnd$ when congestion goes down
    - decrease $cwnd$ when congestion goes up

There is congestion detection in the algorithm:

- use of timeouts/dupacks
    - timeout signals that a packet (or more than one) was lost (serious problem)
        - probably with some serious congestion or other problem (e.g., disconnection)
    - three dupacks signal that a packet was lost but others are passing (minor problem)
        - minor congestion/packet rarely lost due to transmission error
    - in general it is assumed that lost packet implies congestion
        - not true in wireless environments
            - problems like signal attenuation/multipath interference/hidden-exposed terminals, noise can cause false congestion signals/inefficient bandwidth

The algorithm, in practice, works like this:

- increment $cwnd$ by 1 packet per RTT ($cwnd = cwnd + 1$)
    - happens each time an ACK is received
    - linear increase towards maximum capacity
- divide $cwnd$ by 2 whenever a timeout occurs ($cwnd = \frac{cwnd}{2}$)
    - drop detected via triple dupacks
    - multiplicative decrease

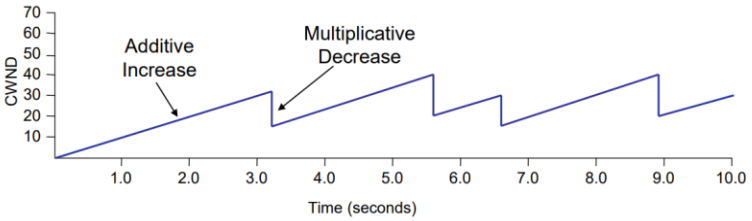


Written by Gabriel R.

Concretely, transmission goes up and down and increments a little for each ACK:

- $Increment = \frac{1}{cwnd}$
- $cwnd \mathrel{+}= Increment$

This way, AIMD results in a behavior similar to a *sawtooth shape* in trace of transmission data rate, like the one you see here on the right.



7.4.2 Slow Start Phase (SS)

Slow Start Phase (SS) is used to quickly determine the available capacity in the first part of a connection. This is used in conjunction with other algorithms, to avoid sending more data than what we are capable of forwarding. Specifically, this phase happens before addictive increase and when:

- first starting a connection
- connection goes dead waiting for timeout

It works like this:

- begin with $cwnd = 1\ pckt$ (or even $\geq 1\ MSS$)
  - Maximum Segment Size = largest amount of data in bytes receivable in a segment
- before threshold
  - exponential increase → $2 * cwnd$
    - this checks available bandwidth
- after threshold
  - linear increase → increment by 1 packet for each ACK
- If congestion level is reached (so, $awnd$ full or $ssthresh$ is reached)
  - new threshold becomes $\frac{1}{2}$ and if it is indicated by:
    - 3 dupacks → $cwnd =$ new threshold
      - linear increase
    - timeout → $cwnd = 1$ → use of Slow Start
      - exponential increase

"Slow" in this context means that the initial start, without this method, would be slow (even though the term is inappropriate, because effectively the content window grows exponentially before threshold).

- $ssthresh$ indicates when to begin additive increase phase
  - at connection setup is very large
  - it's set to one half of $cwnd$ when a packet loss happens
- $ssthresh$ goes through multiplicative decrease for each packet loss
  - because also the $cwnd$ does so
- If we have $cwnd \geq ssthresh$ we use congestion avoidance
  - in this, $cwnd$ gets linear increase for each window of data reaching destination

This phase is maintained until a congestion happens or $ssthresh$ is surpassed.

When a timeout occurs (when this happens, it's symptom of severe congestion):

- $ssthresh = \frac{cwnd}{2}$
  - multiplicative decrease
- if loss is indicated by timeout
  - $cwnd = 1$
- if loss is indicated by 3 dupacks
  - $cwnd = \frac{cwnd}{2}$
    - equal to half of the congestion window value prior to the loss event
- $RTO = RTO * 2$
  - Retransmission Timeout
- Enter Slow Start phase

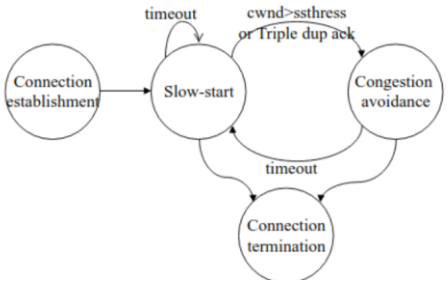
The complete congestion control works with a basic principle: when new data is acked, increase $cwnd$. To summarize properly:

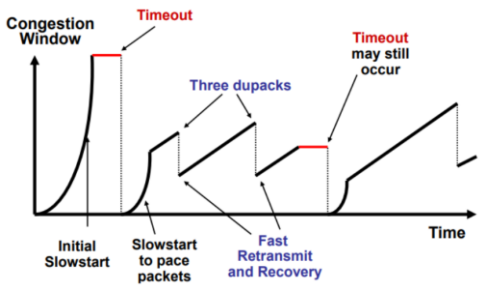
- Slow Start Phase
  - until $cwnd \leq ssthresh$
  - *Exponential* growth
    - at each returning ACK → a new packet is transmitted ($cwnd = cwnd + 1$)
    - when every packet has received its ACK → $cwnd = 2 * cwnd$
- Congestion Avoidance
  - when $cwnd > ssthresh$
  - *Linear* growth
    - at each returning ACK → a new packet is transmitted ($cwnd = cwnd + \frac{1}{cwnd}$)
    - at every RTT → $cwnd = cwnd + 1$

Also, to detect losses, we have:

- Timeouts (*major* problem)
  - set $ssthresh = \frac{cwnd}{2}$
  - set $cwnd = 1$ (so, restart in Slow Start Phase)
- Three Dupacks (*minor* problem)
  - set $ssthresh = \frac{cwnd}{2}$
  - set $cwnd = \frac{cwnd}{2}$ (so, restart from Congestion Avoidance phase)

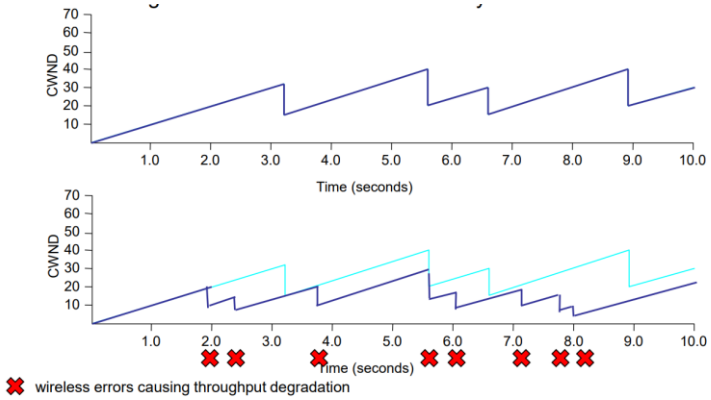
Following, two really clear figures detailing the TCP Congestion Control (left) and the TCP Saw Tooth (right):





Assume, finally, wired vs wireless and for simplicity consider only congestion avoidance: each packet loss determined through 3 dupacks and recovered through Fast Retransmit and Fast Recovery.

Written by Gabriel R.



The first case is what you would get without errors, then the second one anytime the slope goes down because TCP tries to handle packet losses and goes down.

- Because of wireless nature, you're not supposed to slow down, just retransmit the packet (not reduce throughput) and just keep growing until packet loss, otherwise there is bandwidth wasted
    - cannot distinguish between a congestion loss and an error loss, because you are not receiving any acknowledgement back

7.5 TCP VERSIONS

Given its popularity, TCP presents different versions and different evolutions and improvements. Here, many will be described and discussed.

7.5.1 TCP Tahoe

TCP Tahoe is one of the early variants of the TCP (very basic) – it uses Slow Start + AIMD + Fast Retransmit. It just transmits and assures data is received (via acknowledgements, which are received after 1 RTT). You will see in figures the increase of congestion window and the exponential increase.

- Whenever a loss event (or congestion event) of any kind occurs
    - the congestion window is halved
    - this new value is stored in the threshold variable $ssthresh$
    - Slow Start begins from its initial $cwnd$ $(= 1\ MSS)$
        - this decongests the network but strongly limits transmission/reception of data
- When this is done
    - data transmission begins again
    - by setting the initial value of the current $cwnd$ equal to 1 MSS (initial value)
- There is Slow Start
    - the growth of the congestion window occurs gradually
    - following an exponential trend until the threshold value determined earlier is reached
- Beyond this value
    - growth occurs linearly over time according to AIMD

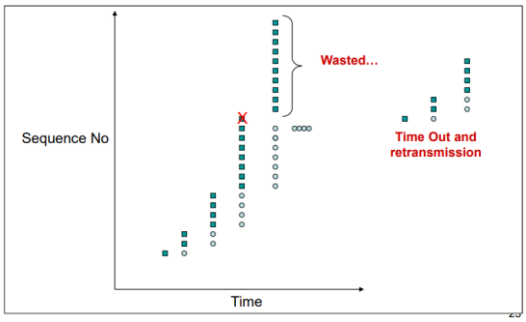
Specifically:

- It includes standard TCP functions
    - e.g., establishing connections, ensuring reliability, handling congestion
- Is uses the Slow Start Phase

Written by Gabriel R.

- It starts sending data conservatively and gradually increases its sending rate
- Prevents congestion when connection is initially established or after a period of inactivity

- Congestion control
  - AIMD
    - In the Congestion Avoidance phase, sender:
      - increases its congestion window additively
      - decreases it multiplicatively in response to congestion
  - Timeouts are only used for losses detection
    - no dupacks – here figure for reference

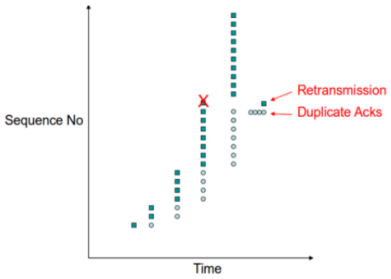


7.5.2 TCP Reno

TCP Reno is the successor of Tahoe and can recover from a single packet loss using:

- both timeouts (RTO) and 3 dupacks (adds Fast Retransmit + Fast Recovery)
  - timeout is a strong indicator of congestion
    - when timeout loss → use Tahoe algorithm (set window to 1 MSS then slow start)
  - 3 dupacks are light congestion (1 packet loss) and, given some data were received, is used to recover without a timeout (Fast Retransmit)
    - when dupacks loss → use Fast Recovery (halve the $cwnd$; $ssthresh = new\_cwnd$)
  - $cwnd$ size = threshold + 3 MSS

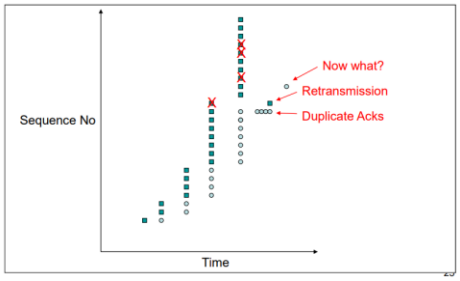
In figures: TCP Reno in case of Fast Retransmit (left) and when there is more than one packet loss (right).



TCP Reno detects packet loss with dupacks.

It can recover from one packet loss without having a timeout, hence improving transfer speed.

We need 3 dupacks to recover 1 packet loss (Fast Retransmit)



After multiple losses, nothing will be sent out to avoid congestion.

The protocol patiently awaits fresh acknowledgments, which may be delayed due to congestion.

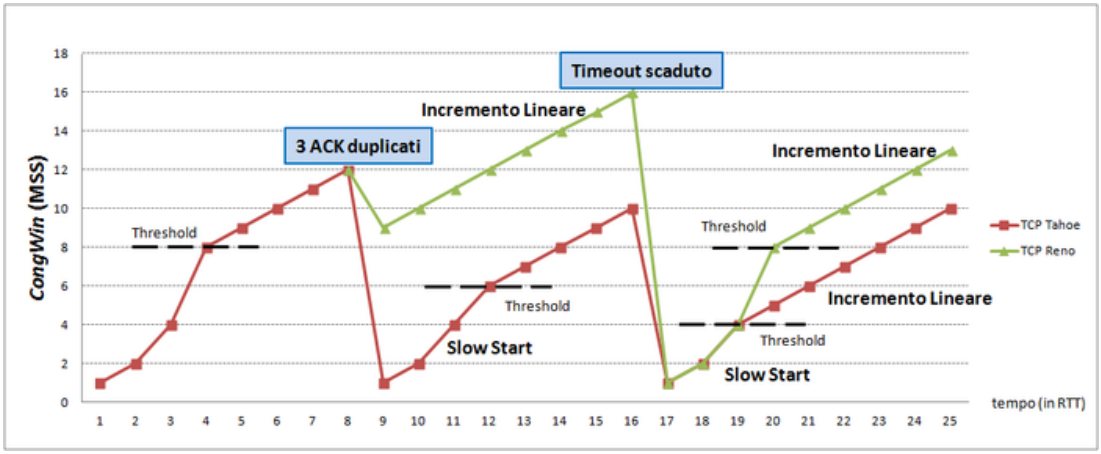
TCP Reno *solves partly the problem of congestion*:

- only when losses are not strongly related between each other
  - at most one packet inside every window
- this can be problematic in case of burst of data
  - given the congestion window will be reduced many times consecutively

So if you lost 3 packets with Reno: you retransmit the first one after 3 dupacks and wait for a timeout for the next one.

Tahoe and Reno are foundational and are still used in many environments where basic congestion control mechanisms are sufficient. They are suitable for general-purpose networking scenarios but may not perform optimally in high-speed or wireless networks.

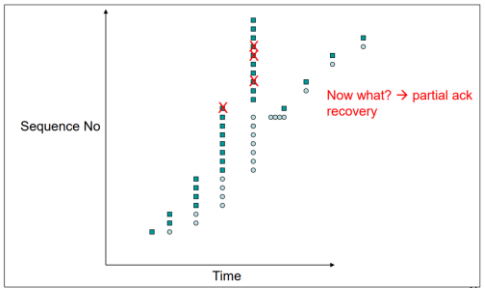
To compare Tahoe with Reno in reasoning:



7.5.3 TCP New Reno

TCP New Reno is an extension of TCP Reno and is designed to *handle multiple packet losses* more effectively (improving Fast Recovery of TCP Reno improving when two or more packets are lost in a windowful, but only when losses are not strongly related to each other). In particular:

- It's like TCP Reno
  - But introduces the concept of partial ACKs without resorting to timeouts
    - There are special ACKs indicated we miss a portion of packets
    - It understands that not all packets up to that sequence number have been received
    - So, it first uses 3 dupacks (1 packet loss) then use partial ACKs
- But differs because
  - New Reno does not halve $ssthresh$ immediately
    - which may reduce the window too much if multiple packet losses occur
- When one of these ACKs occurs during a Fast Recovery phase (after 3 dupacks)
  - This phase, to remember you, reduces cw size up to half of where you were
  - TCP New Reno maintains itself in that phase
    - *by continuing to resend packets* as they are requested
    - *until the last packet sent* in the phase prior to Fast Recovery is encountered



TCP New Reno introduces partial ACKs to recover more packets without the use of timeouts.

A problem occurs with New Reno when there are no packet losses:

- but instead, packets are reordered by more than 3 packet sequence numbers
- In this case, New Reno mistakenly enters Fast Recovery
    - when the reordered packet is delivered
    - duplicate and needless retransmissions are immediately sent

So if you lose 3 packets with New Reno: you retransmit the first one after 3 dupacks and when you receive the partial ACK you retransmit the second, then when you receive another, you transmit the third one and so on.

Note that Tahoe/Reno/New Reno are sender-side innovations; the following one is receiver-side instead.

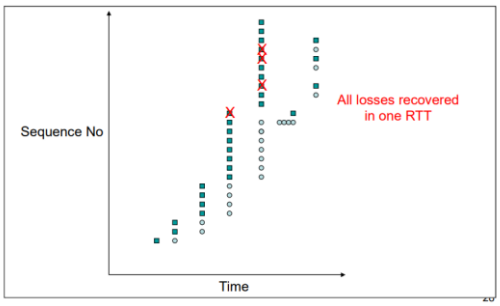
7.5.4 TCP SACK (Selective Acknowledgement)

TCP New Reno works by assuming that the packet that immediately follows the partial ACK received at Fast Recovery is lost and retransmits the packet. However, this might not be true, and it affects the performance of TCP. Fast Recovery mechanism in TCP Reno has two major problems.

- Multiple times reduction of the congestion window ($cwnd$) for the same set of packets
    - this problem is solved by TCP New Reno
- It takes a lot of time to recover from multiple packets that are lost in the same $cwnd$
    - this problem is solved by Selective Acknowledgments (SACK)

SACK addresses some limitations and optimizes sender/receiver side.

- All ACKs are a little bigger but carry more information, they can give the complete picture
- *Returning ACKs declare which packets (even non-contiguous) were received and which not*
- All non-received packets can be retransmitted
    - Recover from multiple losses in just one RTT



Selective ACK declare as said in return which packets were received.

All non-received packets can be retransmitted, and it recovers from multiple losses in just 1 RTT.

Contrary to TCP New Reno:

- SACK sender maintains the information which packets is missed at receiver
    - then only retransmits these packets
- When all packets are ACKed
    - SACK exits Fast Recovery and enters Congestion Avoidance

TCP SACK has a few pros:

- Avoid for TCP sender to wait an RTT
    - to become aware of lost packets
    - to retransmit segments correctly received
- Avoids decrease in throughput by avoid loss of TCP self-clocking property

But has also a few cons:

- $40\ B$ option header constitutes overhead that can be heavy in bandwidth-constrained situations
    - if fully utilized doesn't allow other performance-enhancing mechanisms to be implemented
- It does not allow for header compression techniques or distinguish between various types of loss
    - error, congestion, handoff, fading

So, if you lose 3 packets with TCP SACK: you retransmit them all. This is the version that is currently in use in modern systems because of this reason.

SACK and New Reno are more suitable for environments with higher packet loss rates or where selective acknowledgment and improved recovery mechanisms are beneficial. They are commonly used in modern networks to improve performance and reliability.

7.5.5 TCP Vegas

This is a CA algorithm *emphasizing packet delay (congestion episodes)* rather than individual packet losses. It tries to estimate congestion level before it happens, trying to keep bottleneck link 100% utilized at all times. "This is a great protocol that was never used", quoting the professor. On its features:

- It focuses on *reacting to congestion episodes in RTTs*
    - it handles congestion without any packet loss occurring
    - measures also delays as a congestion indication
- Uses a more accurate timer → it can determine packet loss from one single dupack
- It sets timeouts and measures RTT delays for every packet in the transmit buffer
- It's based on throughput assumption → $actual \leq expected$ where:
    - $actual = \frac{Acks}{RTT}$
    - $expected = \frac{Window\ Size}{RTT}$

It also involves several modifications over basic TCP to improve congestion control and detection:

- *Modified Congestion Avoidance (CA)*
- *Aggressive Retransmission*
- *Aggressive Congestion Window Updating*
- *Modified Slow-Start*

Inside the Modified Congestion Avoidance mechanism:

- throughput → $actual \leq expected$
- *expected* throughput → transmission rate with no other traffic/queue
- Monitor transmission rate
    - Measures: throughput = how much data sent – *goodput* = quantity of useful data sent
    - Given α, β static parameters
        - representing how many packets TCP Vegas can have in queues

- $\alpha < \beta$ and so $expected - \beta < expected - \alpha < expected$

We consider different scenarios inside this one:

- if $expected - \alpha < actual < expected$
  - Queues decreasing → increase transmission rate
  - Low congestion → closer to expected
- if $expected - \beta < actual < expected - \alpha$
  - Don't do anything
  - Maybe congestion
- if $actual < expected - \beta$
  - Queues increasing → decrease transmission rate before packet drop
  - High congestion → prevent packet loss

TCP Vegas calculates once per RTT throughputs the following way:

- $actual = \frac{ActualTransmittedAmount}{RTT}$
- $expected = \frac{WindowSize}{BaseRTT}$
  - $BaseRTT$
    - minimum observed RTT over a certain period/within a certain window of packets

Let's also define the static parameters given above (how many packets TCP Vegas can have in queues):

- $\alpha = \frac{1pkts}{RTT}$
- $\beta = \frac{3pkts}{RTT}$

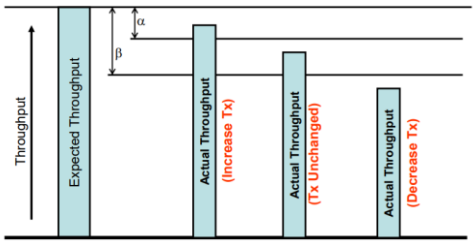
If there is no congestion:

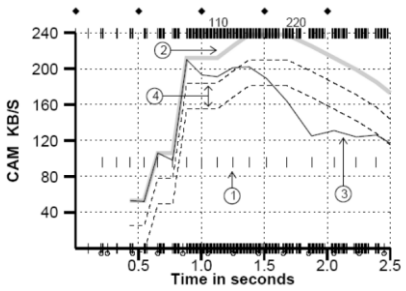
- $actual$ will be very close to the (optimal) $expected$
- Difference in $actual$ and $expected$ will be greater when there is congestion
- In this protocol note that $actual$ can only be $\leq expected$

Given TCP transmission rate = $\frac{cwnd}{RTT}$, TCP takes $cwnd$ updating decision once per RTT and the decision is applied the next RTT for each received ACK as follows (monitor transmission rate):

- $expected - actual < \alpha \Rightarrow cwnd = \frac{cwnd+1}{cwnd}$ (increase window)
- $expected - actual > \beta \Rightarrow cwnd = \frac{cwnd-1}{cwnd}$ (decrease window)
- $a < expected - actual < \beta \Rightarrow cwnd = cwnd$ (window not changed)

Here is a figure about Modified Congestion Avoidance and the evolution over time. You can see TCP Vegas responds dynamically to observed changes in network conditions (as you can see decreasing/increasing throughput/tx), maintaining data transmission and avoiding congestion:





Written by Gabriel R.

Aggressive Retransmission works with 3 dupacks and:

- When TCP Vegas receives either the first dupack or the second dupack (1st and 2nd packets)
  - it checks the *fine-grained timer* expiration (checks timeout)
- If the timer expires
  - TCP Vegas initiates retransmission immediately
- The retransmission activation occurs when
  $(current\_time - sending\_time) + RTT_{min} > TimeoutValue$

Aggressive $cwnd$ Updating can be either three ways:

- with recovery
  - reduce $cwnd$ by *one quarter* instead of half when entering recovery
  - $cwnd$ becomes $\frac{3}{4}$ when it enters into this mode instead of $\frac{1}{2}$
- with multiple loss
  - in case of multiple segment loss from a single window
  - $cwnd$ is reduced by 1 in size
- the initial setting of $cwnd$ is 2 instead of 1

The previous ones are more appropriate for error prone (wireless) environments.

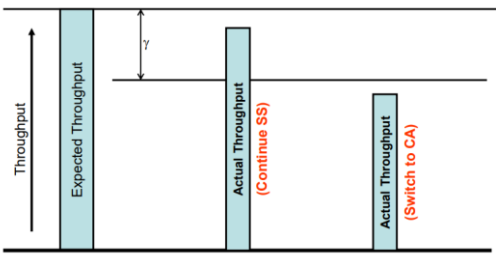
Modified Slow Start (SS) Phase works the following way:

- TCP Vegas calculates the same formulas as the Modified Congestion Avoidance phase
  - but throughputs are calculated *in every alternate RTT instead of once every RTT*
    - $expected = \frac{WindowSize}{BaseRTT}$
    - $actual = \frac{ActualSentAmount}{RTT}$
- There is a new static parameter:
  - $\gamma = \frac{1pkts}{RTT}$

On every next RTT, it does the followings:

- if $expected - actual < \gamma$ → continue using Slow Start:
  - $cwnd = 2 * cwnd$ for each RTT
  - $cwnd = cwnd + 1$ for each ACK
    - Exponential Increase
- if $expected - actual > \gamma$ → switch to Congestion Avoidance:
  - set $ssthresh = cwnd$
  - follow Congestion Avoidance's rules written before

Below, a figure of its working:



TCP Vegas has really interesting ideas, but overall there are many *flaws*:

- *Sensitivity to Delay Variation*
    - TCP Vegas is highly sensitive to variations in RTT times (by asymmetric path)
        - even those unrelated to congestion (RTT may not be a precise measure)
    - This can lead to unnecessary rate adjustments and performance fluctuations

- *Incompatibility with Legacy TCP Versions*
    - TCP Vegas cannot coexist effectively with legacy TCP versions like TCP New Reno
    - When a TCP Vegas flow shares the same bottleneck with a TCP New Reno (for instance)
        - As soon as the pipe is full, and packets get buffered
            - TCP Vegas reduces its data rate
                - thus leaving some more space to TCP New Reno
        - this continues its growth until a congestion episode
            - packet loss due to buffer overflow
    - Consider this example
        - TCP Vegas might get overshadowed by aggressiveness of TCP Reno

In conclusion:

- TCP Vegas introduced some intriguing ideas for congestion control
    - they were subsequently adopted into more recent implementations
    - particularly in the context of adapting to observed delays rather than packet loss
- However, the above controversies that hindered its widespread adoption, killing it in practice
- Vegas never gained mass adoption however
    - apart from some prototypes/specialized network deployments
    - academic research networks/experimental network testbeds
- It is ideal for high-speed networks with low latency where minimizing packet loss is crucial
    - such as in data centers or high-performance computing environments

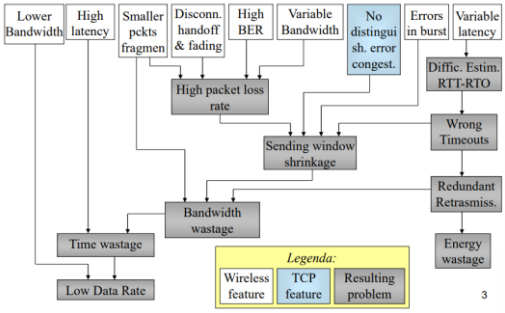
This marks the end of WNMA07 set of slides.

7.6 TCP AND WIRELESS

As a base, TCP congestion control mechanism was designed for a reliable medium:

- wireless is less reliable than cable, so losses are much common
  - TCP was not invented with wireless in mind = both wired/wireless degrade throughput
- in TCP *every loss is treated as a congestion sign*
  - this is a problem for high-bandwidth TCP where random losses are important
  - and definitely TCP over wireless, where lost packets are very common

The below figure represents a plethora of possible problems related to wireless:



There are different wireless problems to note:

- *Error losses*
  - TCP assumes congestion and reduces $cwnd$ (dropping data rate)
- *Losses in bursts*
  - multiple $cwnd$ reductions and drop in data rate
  - better to have smaller packets to lose less data (given high BER)
- *Long delays (satellites)*
  - RTT-unfairness – longer RTT/lower throughput
- *Latency is variable*
  - can be hard to estimate $\frac{RTT}{RTO}$
  - if latency high → protocol is no more fair
- *Variable delays*
  - wrong RTO (Retransmission Timeout) computation
- *Disconnections*
  - multiple timeouts
  - happen because of handoff/fading/weaker signals
- *Variable bandwidth*
  - sudden loss bursts or bandwidth wastage

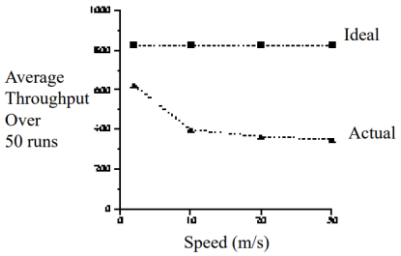
In *multi-hop wireless networks*, such as MANETs, many problems may arise:

- *exponential decrease of throughput/increased delay from* 1 *to* 3 *hops*
  - takes twice the time to transmit data with 2 hops
  - not possible to forward the message at the same time → otherwise collision
- *increasing number of hops beyond* 3
  - *simultaneous transmissions* on more than one link
  - *degradation* continues due to contention between TCP data and ACKs

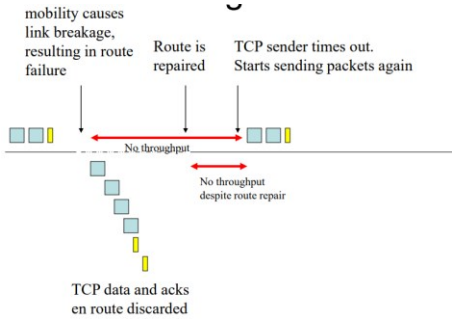
Written by Gabriel R.

- *packets/transmissions are in competition*
  - contention also between datas and ACKs
  - if channel is not free → waiting
- *throughput unstable*
  - when number of hops is large enough → throughput stabilizes due to *effective pipelining*
- collisions may happen with multi-hop wireless connections

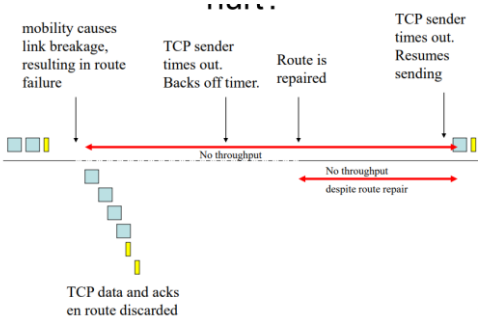
Also given the *mobility* over nodes, *throughput generally degrades with increasing speed* (right figure).



- This happens *because of link breakage and repair latency*
- It may happen that:
  - sender doesn't know that the link is broken
    - it may continue transmitting (packet loss)
  - sender doesn't know when it is all available again
    - it will retransmit when timeout occurs



The couple of figures show why throughput degrades (left – lesser delay) and why repair latency hurts (right – longer delay), where bandwidth is wasted or because layer do not communicate with each other and problems in one might affect the other.



To bring throughput closer to ideal, we can improve it via different ideas:

- *Network feedback*
  - if $awnd$ (Advertised Window) of receiver has size 0
    - network can notify sender
    - the sender stops immediately
- *TCP failure*
  - sender notified by sending explicit message about it
  - performance improves using explicit notification
- *TCP callback*
  - let sender know when link is broken/repaired
  - probing (e.g., persistent pkt retransmissions)
  - explicit link repair notification
- *Dynamic TCP timeout*
  - alleviates repeated TCP timeouts and backoff

7.6.1 Wireless TCP Protocols

Over the transport layer, let's give some possible solutions via different ways (in italic the treated versions):

- Traditional TCP
  - protocol is not aware of wireless link
  - possible timeouts are treated like congestions
- Implementations
  - *TCP Reno*
  - *TCP New Reno*
  - *TCP Vegas*
  - *TCP SACK*

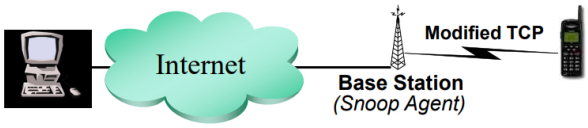
- Connection split
  - The TCP connection is split into *two parts*:
    - the first one
      - between the sender and an intermediate node
    - the second one
      - between the intermediate node and the receiver
  - This allows for different congestion control and retransmission mechanisms to be used
    - they can improve performance in wireless networks
  - Local retransmission
    - intermediate nodes retransmit lost packets without waiting for ACKs
  - Quick actions on wireless link
    - intermediate node can improve the performance of the wireless link
      - e.g., adjusting the transmission power or data rate
- Implementations (expanded in each to give more context)
  - I-TCP (Indirect TCP – improve performance over high-latency links e.g., satellites)
  - M-TCP (Multicore TCP – optimization on multicore systems)
  - PROXY (provides multiple layers for NAT/TCP proxies)
  - *SNOOP*

- Pure End to End
  - The TCP connection is maintained between the sender and the receiver
    - entire connection managed
      - using same congestion control/retransmission mechanisms
  - Here sender is aware of the wireless link
  - This is better than old version → because of retrocompatibility and simplicity in handling
- Implementations
  - Freeze-TCP (handles degradation in packet reordering)
  - Delayed Dupacks (delay dupacks = assess congestion before retransmissions)
  - WTCP (Window TCP, enhancing in performance and control)
  - *TCP Hybla*
  - TCP High Speed (improvements in control and handling)
  - FAST TCP (improves TCP Vegas via different estimations of RTT to compete with Reno)
  - TCP-Aware (optimize TCP traffic via acks, scaling, control in performance)

- TCP Probing (probe the network to know parameters)
- *TCP Westwood*
- *TCP CUBIC*
- TCP Compound (selects/adapts congestion control algorithms for fairness/high throughput)

7.6.1.1 Snoop Protocol

The Snoop Protocol (Balakrishnan et al., 1995) is a transport layer protocol *designed to address the high Bit Error Rates (BER)* of wireless networks.



- Basically, there is a tower with a big buffer keeping a list of ACKs from receiver
  - it handles dupacks/lost packages, without stopping the sender from sending new data
- It implements a *Snoop Agent* at the Base Station
  - it *monitors all packets* of sender and receiver
  - *all packets that are not yet acknowledged are cached at the Base Station*
    - this allows for local retransmission of lost data
    - it performs them based on dupacks and locally estimated RTT times
  - it intercepts dupacks (avoid redundant retransmissions and *cw* shrinkage):
    - $\frac{1}{2}$ → immediate retransmission (sender not notified – *hide losses to sender*)
    - 3 → let sender know about it ($cwnd$ shrinkage)
  - each time there is a dupack it means you lost the packet
    - eventually, it will get to destination
    - this allows to protect sender from weak or inexistent link typical of wireless
- The path is: Sender ⇔ Snoop Agent ⇔ Receiver
  - needs low latency between Sender and Snoop Agent, otherwise it's better traditional TCP
- This is a transport-aware reliable protocol, potentially improving TCP performances a lot

Pros:

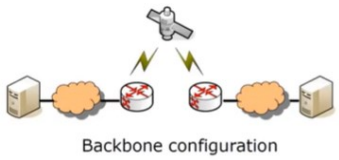
- End-to-end preservation
  - no ACK created by Snoop Agent
- Local retransmission
  - minimizing the need for RTT-communication for retransmissions
- High BER address
  - ensure reliable communication even in challenging conditions
- Dynamic caching
  - bandwidth savings and improved scalability

Cons:

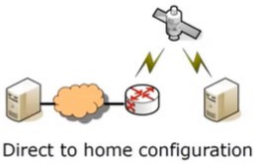
- Requirement of little latency on the wireless link
  - delays in transmitting or processing data packets could impact performance
- Does not guarantee against lost disconnections
- Not immediately usable after a handoff
  - Slow Start → no packets in new cache

7.6.1.2 Satellites

There are also different *satellite scenarios*:



- Geostationary Orbit (GEO) satellites – $36000\ km$
  - backbone configuration
  - bridge between terrestrial antennas
- Low Earth Orbit (LEO) satellites – $100/1500\ km$
  - direct to home configuration
  - terrestrial antenna → satellite → home



All of them have:

- high RTT
  - it can be circa $600\ ms$ for GEO
- important PER (Packet Error Rate) due to radio channels
  - typical values in range of $[0/10\%]$ interferences
    - because of weather, antenna position, satellite constellation, mobility
  - this forces the adoption of some CRC systems (error correction – Cyclic Redundancy Check)

7.6.1.3 Slow Start and Congestion Avoidance Models

The *Van Jacobson algorithm* is a combination of the Slow Start (used to determine the initial sending rate) and the Congestion Avoidance (CA) model (used to control the sending rate after a packet is lost).

- In the Slow Start (SS) phase:
  - $cwnd = cwnd + 1$ for every new ACK received
  - $cwnd = 2 * cwnd$ for every RTT
- In the Congestion Avoidance (CA) phase:
  - $cwnd = cwnd + \frac{1}{cwnd}$ for every new ACK received
  - $cwnd = cwnd + 1$ for every RTT
- RTT changes dynamically
- The discrete time behavior of W can be effectively approximate by a continuous time model

previous congestion window

$$W_{i+1} = \begin{cases} W_i + 1 & SS \\ W_i + 1/W_i & CA \end{cases}$$

new congestion window after receiving one ACK

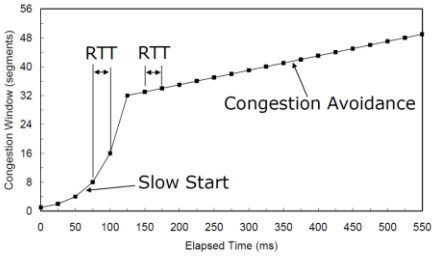
W is doubled every RTT

time in slow start

$$W(t) = \begin{cases} 2^{\frac{t}{RTT}} & 0 \le t < t_\gamma \quad SS \\ \frac{t - t_\gamma}{RTT} + \gamma & t \ge t_\gamma \quad CA \end{cases}$$

Number of RTTs elapsed since entered in congestion avoidance

ssthresh value



One of the problems that protocols have to solve is *RTT unfairness*:

- $B(t) = \frac{W(t)}{RTT}$
  - $B(t)$ = bandwidth/transmission rate
  - $W(t)$ = window growth rate
- longer RTT → slower phase growth rate
- smaller RTT → bigger bandwidth

Written by Gabriel R.

The algorithm was used as a base for improvement in next years and solved the problem of the original computation, which *did not take the variance of the sample RTTs into account*. It was refined multiple times over the years and the variant in widespread use today is called CUBIC.

As we can see, traditional TCP protocols don't work very well with these links.

- high-speed networks requires very high volumes of W: $B(t) = \frac{W(t)}{RTT}$
- too little retransmission timers lead to frequent transmissions

The following protocols are a list of pure end-to-end, in which the sender is aware of the wireless link. There are endless number of implementations of these ones: many are present in the next subsections.

7.6.1.4 TCP Hybla

TCP Hybla is a congestion control algorithm first presented in 2004 and:

- Equalizes the transmission rate against the RTT (*fair RTT*)
- A longer RTT compensated *by sending twice on every ACK* (≠ every RTT)
- This is designed to address negative effects of long RTTs and multiple losses in $cwnd$
  - to do so, it removes the reliance on RTT from the $cwnd$ algorithm
- It has, in reality, one very specific focus: address the TCP satellite problem of very long RTTs
  - in general, networks with long delay-bandwidth product

Since the TCP window-based transmission algorithm depends on network delays, TCP Hybla proposed to obtain for long RTT connections the same instantaneous transmission rate. This can be achieved:

- by making $cwnd$ independent of RTT
- by compensation of the effect of the division by RTT

Here we introduce a parameter $\rho = \frac{RTT}{RTT_0}$ (Greek letter is "Rho") to adjust window size based on RTT variation (attempting to scale TCP Reno so to behave like a TCP Reno connection) where:

- RTT is the actual Round Trip Time
- RTT_0 is the *reference* Round Trip Time (e.g. $RTT_0 = 25\ ms$)

In following figure, formula for $W(t)$ differs from classic TCP (SS – Slow Start/CA – Congestion Avoidance):

$$W^H(t) = \begin{cases} \rho 2^{\rho \frac{t}{RTT}} & 0 \le t < t_{\gamma,0} \quad \text{SS} \\ \rho\left[\rho \frac{t - t_{\gamma,0}}{RTT} + \gamma\right] & t \ge t_{\gamma,0} \quad \text{CA} \end{cases}$$

$$B^H(t) = \frac{W^H(t)}{RTT} = \begin{cases} \frac{2^{\frac{t}{RTT_0}}}{RTT_0} & 0 \le t < t_{\gamma,0} \quad \text{SS} \\ \frac{1}{RTT_0}\left[\frac{t - t_{\gamma,0}}{RTT_0} + \gamma\right] & t \ge t_{\gamma,0} \quad \text{CA} \end{cases}$$

Two comments on the formula above:

- During each RTT, $cwnd$ is incremented by ρ^2 instead of 1
  - to avoid penalizing short-RTT connections
- This rapid increase in $cwnd$ may lead to bursts of losses when the network ceiling is reached
  - so TCP Hybla recommends using SACK TCP for faster recovery
  - and TCP Timestamps for accurate RTT measurement

Satellite links typically involve large distances, resulting in high RTT. TCP Hybla is designed to handle precisely this kind of situation:

- Its key characteristic is the insensitivity to RTT variations
- In a satellite link with a large RTT, the exponential increase mechanism can be advantageous
    - It allows for a more rapid increase in the congestion window size
        - adapting to the available bandwidth more efficiently
- TCP Hybla maintains fairness with other flows in the network
    - ensuring that it doesn't monopolize the available bandwidth

Pros:

- End-to-End solution
    - it adjusts congestion control parameters based on feedback from endpoints
- Changes only on sender side
    - easily deployable (changes localized/not extensive modifications)
    - no damage for the entire system
- Has RTT fairness
    - RTT used to speed up transmission speed for connection with long RTTs
    - dynamically adjusting bandwidth ensures better allocation

Cons:

- It is very aggressive
    - it can lead to multiple losses
    - this can be beneficial to get high throughput
        - but you risk losing packets all together
- Measured RTT is sensitive to buffer size
    - limited and not sustainable anymore at a certain point
    - the measured RTT may become distorted
- No handling on BER/disconnections (as most of TCPs)
    - lacks specific mechanisms to handle bit errors or sudden disconnections in the network
    - this is true also for other TCP implementations

Some doubts have been expressed about the friendliness and fairness aspects of TCP Hybla.

- *Fairness*
    - one flow with the *same* protocol → *same* bandwidth usage
    - ensures each connection gets a reasonable share of bandwidth
- *Friendliness*
    - one flow with *different* protocol → *same* bandwidth usage
    - ability of a different flows of different TCPs to coexist
        - specifically sharing band and not dominating the network compared to others
        - measures total amount of bandwidth and average bandwidth
    - new version shouldn’t steal bandwidth of older versions
        - in general, Hybla could be disruptive for other protocols
        - it can take all new available bandwidth

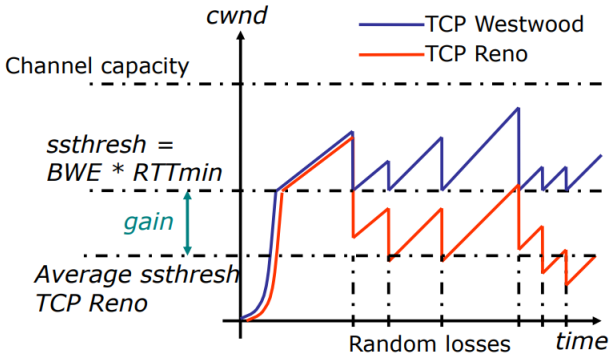
7.6.1.5 TCP Westwood/Westwood+

TCP Westwood/TCP Westwood+ (TCPW) are both pure end-to-end congestion control algorithms which try, like TCP Vegas, to *address the high-bandwidth problem* via monitoring of bandwidth and capacity:

- *changes involve only the sender side* of the connection
- flow control is based on estimating the *available bandwidth* (BWE = Bandwidth Estimation)
  - estimation is done by monitoring the arrival rate of ACKs at the sender side
  - use of BWE to set $cwnd$ and $ssthresh$ after a loss
    - 3 Dupacks
      - $ssthresh = BWE * RTT_{min}$ (sender current "transit capacity" of the path)
        - instead, TCP New Reno → $ssthresh = \frac{cwnd}{2}$
      - if $cwnd > ssthresh$
        - $cwnd = ssthresh$
    - Timeout expiration
      - $ssthresh = BWE * RTT_{min}$
        - instead, TCP New Reno → $ssthresh = \frac{cwnd}{2}$
      - $cwnd = 1$
  - sending more than what you receive won't affect BWE

TCP Westwood and TCP Reno represent different approaches to congestion control in TCP (shown in figure and previous formulas) and are compared here.

- TCP Westwood uses BWE reducing bandwidth only to transit capacity, better handling router queue capacities
- TCP Reno follows a more traditional congestion control approach, responding to packet loss events, keeping the bottleneck limit saturated



TCP Westwood methods do a Rate Estimation (RE) to enhance congestion control:

- this is computed at the sender by *sampling* and *exponential filtering*
- based from ACKs *inter-arrival times* and amount of bandwidth delivered
  - data ACKed is aggregated in interval $T = RTT$
- used by the sender to set $cwnd$ and $ssthresh$ after packet loss
  - indicated by 3 dupacks or timeout

Rate estimation is based on three formulas and is obtained by aggregating the data ACKed during the interval T (typically $= RTT$) (as you can see the right).

$$b_k = \frac{\sum_{t_j > k-T} d_j}{T} \quad \text{sample}$$

$$RE_k = \alpha_k RE_{k-1} + (1-\alpha_k)\left(\frac{b_k + b_{k-1}}{2}\right) \quad \text{exponential filter}$$

$$\alpha_k = \frac{2\tau - \Delta t_k}{2\tau + \Delta t_k} \quad \text{filter gain}$$

- The *filter* is quick to compute
  - allows to specify the sample measurement over the bandwidth
  - giving a good estimation of the bandwidth

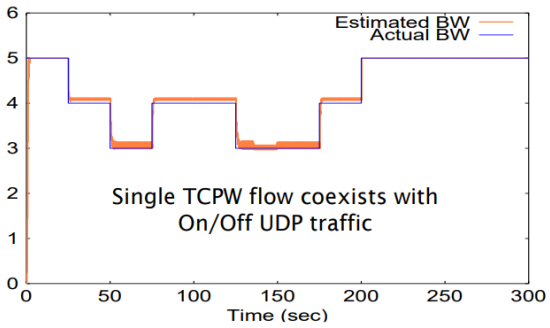
- The *sampling* is based on the RTT
  - small T → more aggressive time estimation
  - big T → more conservative
- The *filter gain* slows down the calculation → it fine-tunes the sampling overtime

Having a read here: Westwood+ does this low-pass filtering just described on returning ACKs rate.

The graph provided illustrates the Bandwidth Estimation over time, measured in Mbps. It shows the dynamics of the estimated bandwidth compared to the actual bandwidth.

The fluctuations in the Estimated BW curve demonstrate TCP Westwood's adaptability to changing network conditions.

When the TCPW flow reaches equilibrium, it means that the sender has found a sending rate that matches the available network capacity without causing congestion.



Single TCPW flow at equilibrium does estimate "residual bottleneck bandwidth"

Pros:

- BWE allows to reach higher throughput done at sender side
  - adjusting parameters dynamically and estimating bandwidth
- Westwood improves the addressing of a lossy-link problem
- Code modifications only on sender side
  - easily deployable without infrastructure/receiver-side implementations
- Works without inspecting or intercepting packets at intermediate nodes
  - inspects only TCP headers without needing support from transport layer
- Continuous monitoring of sender connection thanks to estimation
- Reduces $cwnd$ shrinkage avoiding useless restrictions in case of random/sporadic errors

Cons:

- Wrong BWE over asymmetric links
  - estimation may not be accurate
  - possibly incorrect BWE calculations, impacting algorithms' performance and throughput
- It does not improve sender's ability to take advantage of a sudden large rise in the network ceiling
- No handling of high BER/disconnections
  - no specific mechanisms to handle such situations
- Doubts about friendliness and fairness
  - like Hybla, it's quite aggressive and may not be the best on both aspects
- Would not guarantee high performances in case of asymmetric link/high BER
  - specifically when propagation time between sender and receiver is too low/too high

Westwood usage is tailored for specific network types where traditional TCP variants may not perform well due to high latency or unreliable connections.

Written by Gabriel R.

7.6.1.6 TCP Adaptive Selection

TCP Adaptive-Selection tries to select the right TCP protocol based on the connection:

- On the same server, not a single variant but concurrent use of different TCP enhancements
- It can be applied in different ways, depending on:
  - agent that performs the TCP selection
    - i.e. receiver, intermediate router, sender
  - exploitation of cross-layer possibilities
    - not linked by the standard
  - possibility to change the TCP version of an on-going connection
    - dynamically (on-the fly), even if it's not easy
- Linux appears to be the most convenient choice to implement TCP adaptive-selection
  - Most TCP variants are available as modules already
  - A new "TCP adaptive-selection" module that calls other ones can be the solution
- There are different modules that can be replaced in TCP-module container
  - TCP internal parameters
    - E.g. RTT/Bandwidth Estimation
  - Cross-layer information/reliable channel information

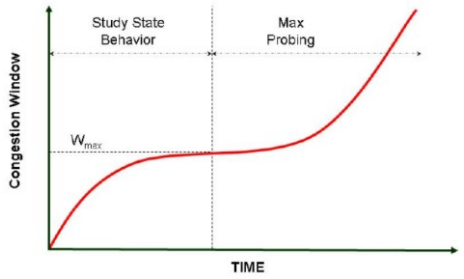
7.6.1.7 TCP CUBIC

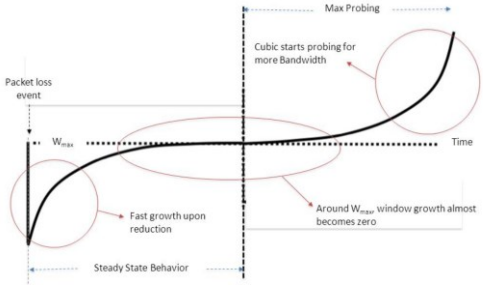
TCP CUBIC is an optimized congestion control algorithm for high-speed networks with high latency (for example, consider Long Fat Pipes/Networks: high bandwidth/high latency, e.g. satellite links, transoceanic cables, long-distances, etc.) – so, to solve the problem when $bandwidth * delay$ product is large.

Also, as features:

- *It's used by default inside Linux kernels*
- The window ($cwnd$) size is a cubic function of time since last congestion event
  - The window-growth function slows down as previous network ceiling is approached
    - becomes concave = ramps up to the size before last congestion event
  - then increases rapidly again if this ceiling is surpassed without losses
    - becomes convex = probes for more bandwidth, slowly first, then more rapidly
- The algorithm works like the following:
  - Inflection point set $cwnd$ prior last congestion event
    - so, congestion window is time-dependent
  - Very quick initial growth
  - Slows down and maintains stable to a value
    - around $cwnd$ when congestion happened
- Major difference between other standard TCPs
  - TCP CUBIC does not rely on the receipt of ACKs to increase the $cwnd$
  - TCP CUBIC's $cwnd$ depends *only on the last congestion event*
    - Less RTT-unfairness since the window growth is independent of RTT
    - Rapid recovery after a loss event, maximizing throughput
    - Rapid expansion of $cwnd$ when ceiling is detected and optimization of it

Below, some figures about the TCP Cubic $cwnd$ growth (left), explaining also how it works (right):





TCP Cubic has been widely deployed in modern operating systems and is suitable for a wide range of network environments, including high-speed and long-distance networks.

To summarize until now, none of the solutions presented above solves all the problems, because the server always would need to change the TCP implementation according to the specific situation.

So, the better implementation depends on the specific context: I personally tried to present each in a distinctive yet precise way so to make you remember by memory or by studying everything that was presented, given sometimes sources and slides are not the best. Moving on anyway.

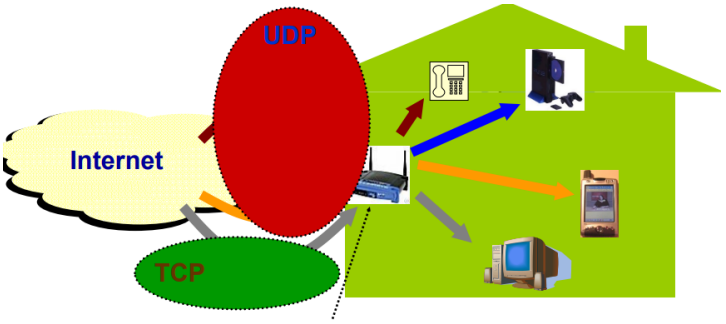
8 WIRELESS MAC AND TRANSPORT PROTOCOLS INTERFERENCE (WNMA09)

In a home, usually, there are many requirements, like increasing bandwidth (IEEE 802.11g), connected appliances, online entertainment (media center), all needing high reliability.

- Applications vary in requirements
  - between low delay up to bandwidth + reliability
    - video chat, online gaming, video streaming, downloading, etc.
- We need to carefully consider the *amount of bandwidth each device gets*

Home routers can be a *bottleneck* of the network traffic, acting as a *gateway* between *in-home* devices and the outside world.

Inside a *shared channel* two nodes cannot transmit at the same time. *Networks were mostly developed using TCP-based traffic* (suitable for reliable traffic, e.g. web browsing, file transfer), specifically when UDP-based comes into play (extremely delay-sensitive, e.g. streaming).



The objective of this study is to keenly observe the repercussions of introducing *multiple streams* on a *single wireless hop*, challenging traditional assumptions. For this study, we consider the main features of:

- TCP
  - Window-based flow control mechanism
    - Regulating data from server and receiver
  - Continuously probing the link for more bandwidth
    - Adjusting sending rate based on network conditions (RTT/packet loss rate)
  - Can fill links and queues with its packets
    - This aggressive behavior may be dangerous if not handled properly
- IEEE 802.11g
  - High Bandwidth
    - $54\ Mbps$ nominal, $\sim 20\ Mbps$ effective
  - Retransmission mechanism
    - Hides wireless losses (e.g. packet collision/interference)
    - Increases delays (more time to retransmit packets)

So, the work presented in this chapter aims at:

- Analyze the coexistence issues among TCP-based and UDP-based flows
  - Impact of TCP's congestion avoidance on (UDP-based) real time flows
- Evaluate the interference among wireless MAC and transport protocols
  - Impact of MAC Layer buffers and retransmissions on:
    - Real-time applications (*Jitter*)
      - Absolute *difference between delays* (measuring its *variation*)
      - *Deviation from true periodicity* in relation to reference clock signal
        - Variation in the delay in receiving transmitted packets
        - caused by internal queues at congested routers
    - Best effort application (Throughput) = how much data was sent

Written by Gabriel R.

One of the issues of wireless MAC is packet retransmission:

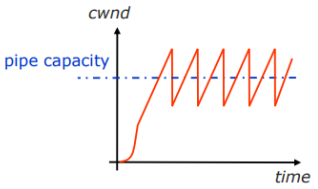
- it causes delays that can be extremely dangerous for UDP-based services
  - applications using UDP do not generate much traffic
- this delay is increased with long TCP flows (e.g. FTP)
  - it continuously creates queues when interacting with applications
  - taking as much as possible

The following parts will discuss possible solutions. Summarizing until here:

- Long lasting FTP/TCP flows increase delays
- Need for queuing delay reduction
- Easy solution: *appropriately setting MAC layer parameters*
  - *Set MAC layer retransmissions to* 3 (which ensures maximum throughput possible)
  - This ensures less jitter, so like $1/2$ packets lost every 900
  - Smaller MAC queue size (max 50 packets)

8.1 Smart Access Point with Limited Advertisement Window (SAP-LAW)

Usually, TCP has an aggressive behavior, because of its window-based *flow control mechanism* probing always from *more bandwidth*, which can fill up the AP buffer with its packets (*this may increase delays* and *deteriorate performances* of real-time streams). Paper of reference for this part here.



A solution can be the following – *changing TCP behavior to allow UDP-based apps to be not jeopardized*:

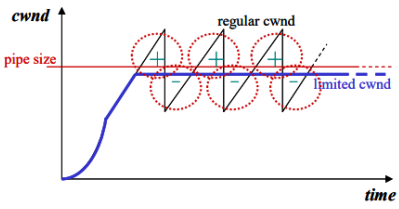
- granting high goodputs for downloading applications and avoiding queues
  - bad for concurrent real-time apps → given the effect of jitter over starting times
- UDP traffic is seen as a problem for TCP flows
  - no improvements on TCP → act on TCP to not upset UDP performances
- best tradeoff to provide:
  - *low* per-packet *delays* for real time applications
  - *high goodputs* for *downloading* applications

The main idea is having a “Smart” AP:

- exploit the *advertised window* ($awnd$ - receiver side window) to *limit bandwidth* used by TCP flows:
  - this way *limiting the speed of the sender*
  - reducing frequencies of beacons reduces overhead
  - better resources allocation and improved scalability

We distinguish:

- *regular* $cwnd$
  - regular TCP
  - typical saw tooth shape
- *limited* $cwnd$
  - window regulated by exploiting $awnd$



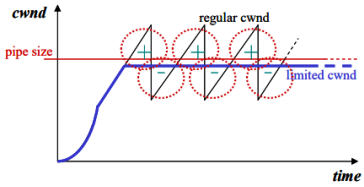
Regular window provides a sending rate that oscillates (+ and – in the picture) around the one ensured by the *limited* window

A balance can be reached on the final throughput

Written by Gabriel R.

SAP-LAW uses this information to *restrict the congestion window strategically*. Infact:

- *it avoids buffer utilization* at the AP
  - Exploits info available at AP to determine:
    - amount of bandwidth occupied by UDP-based traffic
    - number of active TCP flows



$$maxTCPrate(t) = \frac{(C - UDPtraffic(t))}{\#TCPflows(t)}$$

No queues:
- lower delays (for UDP flows)
- smoother throughput (no losses and reductions of the sending window for TCP flows)

- it does on-the-fly *modifications* of the *advertised window* of TCP flows *at the AP*

More specifically regarding the formulas employed, for example to calculate the $maxTCPRate$:

$$maxTCPRate(t) = \frac{(C - UDPTraffic(t))}{\sharp TCPFlows(t)} \quad (8.1)$$

Where C is the total capacity of the channel.
The AP can change on-the-fly the advertise windows, that's:

$$min\{clientAdvertiseWindow,\ maxTCPRate\}$$

This make SAP-LAW easy to deploy as it requires modifications only at the AP.

This formula shares the bottleneck with different RTTs, leading to *RTT-unfairness*:

- Consider TCP throughput is inversely proportional to the RTT length:
  - causing different throughput based on the RTT (even with the same windows)
  - usually caused by asymmetric links nature, congestion, load imbalance, routing disparities
- Hence, this leads to a new formula, which is present in the next subsection

This was measured with a prototype (SLUS – SAP-LAW in User Space) and:

- SAP-LAW when tested has good performances
- Throughput evaluation:
  - has high oscillations in regular environment
  - has smooth progression (but same final throughput) for SLUS
- Interarrival time evaluation:
  - had high oscillations for packet queuing for regular environment
  - had limited queuing/oscillations (better jitter) for SLUS

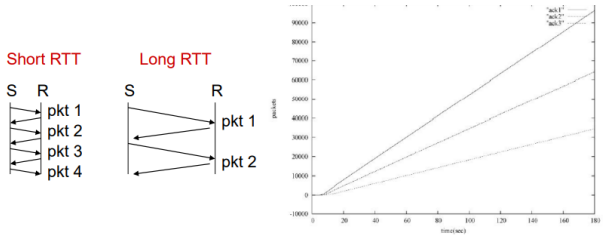
In conclusion:

- In in-home wireless scenarios, concurrent long-lasting *TCP flows increase delays*
  - especially on wireless links
- *Real-time* online applications need a *reduction* of the *queuing delay*
- *SAP-LAW*: on-the-fly *modification* of the *advertised window* of TCP flows *at the AP* to:
  - augment UDP flows performances
  - maintain a high goodput for TCP downloads
- SAP-LAW is easily deployable as it requires modifications *only* at the *AP*

8.2 Enhancing SAP-LAW for RTT-Fairness

The throughput of a connection is inversely proportional to the RTT length.

Short RTT flows capture the channel, while longer RTT flows starve. Right figure shows this. Paper of reference for this part here.

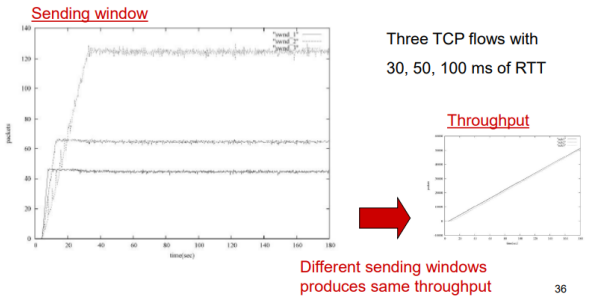


Even SAP-LAW does not solve it: same windows results in different throughputs (if different RTTs).

This leads *to a new formula for awnd that includes RTT,* having *cwnd*s based on the previous formula.

This make *SAP-LAW easy to deploy as it requires modifications only at the AP*.

$$maxTCPrate_i(t) = \frac{(C - UDPtraffic(t)) * RTT_i}{\sum \frac{RTT_i}{avg\_RTT_{min}}}$$



On the simulation:

- Conducted on different environments to compare SAP/LAW with other protocols/solutions:
  - TCP Sack
    - Interarrival of game events → High oscillations due to queuing
    - ACK transmission of three TCP flows → High bandwidth usage
  - TCP Vegas
    - Interarrival of game events → No queuing: low oscillations
    - ACK transmission of three TCP flows → Still unfair bandwidth usage
    - Plus, TCP Vegas cannot coexist with legacy TCP
  - SAP-LAW
    - Interarrival of game events → No queuing: low oscillations
    - ACK transmission of three TCP flows → *Fair* bandwidth usage
    - In efficiency, SAP-LAW is the most efficient one
      - No losses also means higher efficiency in terms of throughput

Comparing results on High and Fair TCP's throughputs:

- Single throughputs
  - The three TCP flows have the same throughput regardless of different RTTs
- Aggregate throughput
  - This consumes all the available bandwidth

In conclusion:

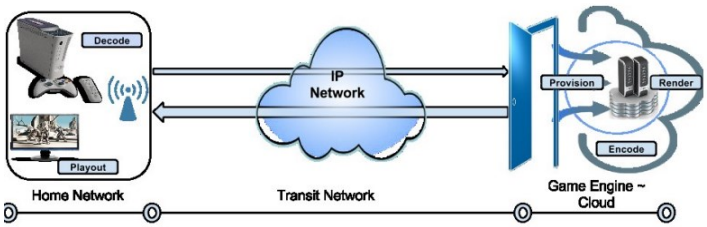
- Two problems:
  - Long lasting *TCP flows increase delays*
    - *Real-time* applications need a *reduction* of the *queuing delay*
  - TCP's throughput depends on RTT
- *SAP-LAW solution*: on-the-fly modification of the advertised window of TCP flows at the AP
- SAP-LAW is easily deployable as it requires modifications *only* at the *AP*
  - Utilized on top of legacy transport protocols

Written by Gabriel R.

8.3 VEGAS OVER ACCESS POINT (VOAP)

We consider *classic online games*:

- $10/200\ Kbps$ of traffic between client and server
- Messages contains actions and game arena updates
- Each message has to be delivered within $100/150\ ms$



Paper of reference for this part here. This is important specifically for *thin client games* (cloud-based):

- each client sends relatively low amounts of traffic to the server
  - typically around $10\ Kbps$, consisting of user actions or input commands
- the server sends much higher volumes of data, often tens of Mbps, to each client
  - this includes high-quality video streams of the game environment
  - enabling clients to render and display the game graphics locally

More recent examples (at the time of this writing): Google Stadia/Amazon Luna.

In Thin Client Game Systems:

- *Coexisting TCP flows and thin client game flows*
- The tested solution acts on the advertised window of TCP based flow
  - The difference from SAP-LAW is the *algorithm used to set the advertised window*
  - *Inspired by the TCP Vegas algorithm*
    - The AP can *enforce it on all TCP flows* in the bottleneck (so no friendliness issues)

Consider VoAP (Vegas over Access Point), used in the context of VoIP applications:

- Vegas is designed to improve performance/minimize packet loss/maintain high throughput
  - focuses on measuring/reacting to changes in network congestion based on RTT variations
- VoAP tries to reduce latency/improve packet loss prevention/enhance QoS for VoIP

Two thresholds are present: α and β measured in milliseconds and, assuming $\alpha < \beta$:

- if $delay < \alpha$
  - the channel is free
  - the advertised window is increased
- if $\alpha < delay < \beta$
  - the channel is well exploited
  - the advertised window is left unchanged
- if $delay > \beta$
  - the channel is saturating
  - the advertised window is decreased

Remember from here, the static parameters are how many packets TCP Vegas can have in queues.

In interactivity – queuing delay (considering AP queue length set to 250 packets):

- when measuring queue delay
  - in standard TCP
    - very high oscillations
  - in VoAP
    - very low oscillations
- when measuring sending window
  - in standard TCP
    - random spike and sawtooth shape
  - in VoAP
    - random spike and almost constant plot

In efficiency – throughput:

- Packets in excess do not increase TCP throughput
  - they just increase the queue length
- *VoAP uses delays to detect the bottleneck limit*
  - keeps the number of outgoing packets below it
  - this is the reason the previous list said “constant plot”
- The difference between the two throughputs is about 1%

In fairness – fair share of bandwidth:

- The total number of outgoing packets is fairly divided between the active flows
- Three TCP flows starting at different times reach overtime an equal amount of sending

On experiments over real game traffic (over multiplayer games):

- fairly low packet size/interarrival in the order of $1/1.5\ ms$/bandwidth of a few Mbps
- with standard environment
  - bandwidth is very fragmented overtime
  - queue delay spikes very high
- with VoAP
  - bandwidth spikes then remains constant
  - queue delay is really low, then kept constant thanks to previous observations
  - very low percentage of packets with delay over β (or over $\beta + x\ ms$)

In conclusion:

- Keep a low per packet delay (interactivity)
- The throughput is preserved (efficiency)
  - Important for TCP-based flows (downloaded files but also for some high-quality videos)
- TCP flows can fairly share the bandwidth (so, equally = fairness)
- VoAP is easily employable in real scenarios
  - Requires modifications only at the AP
    - buy new or update old ones

9 PROJECT DISCUSSION (WNMA-PROJECTS1/2)

Note: this here was based on two additional lectures (out of lessons' timetable) to present the two set of slides of projects and in case to discuss specifically on the project – period half of November. The first part is related to a lesson actually not dedicated to projects themselves, taken at the beginning of the course.

It's important to know how to structure a project paper; it started from a PowerPoint and a PDF of the Game of Ur (found inside the "Past Project Examples" folder on Moodle in its full "textual" form, not the slides sadly). Basically, it's a two-player board race game which is dice-based kind of like backgammon. Basically, the main points of the presentation are:

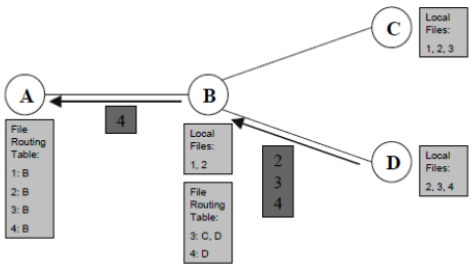
- Presentation of the context
- Rules of the game
- Game modes
- Game technologies
- Game in app and components
- Game evaluation
- Future extensions

The paper itself is focused on the development of a mobile game to teach players something historical related to Sumerians, while at the same time testing how this subject can be used to further see how "serious" games can be accessible to almost anyone. In this we see a presentation of the context and the related work.

We have then a complete game description, to delve into the whole context of the game. There are also hardware requirements that need to be respected, and design and implementation, while also delivering some screenshots of the final product. The final part is focused on the game evaluation and conclusions as results of the whole research work.

(Onto the projects themselves)

The first example is shown is M2MShare, which is a delay tolerant, proximity based, P2P application for file sharing among smartphones. Phones in proximity may exchange files or assign file retrieval task ("please find this file for me so that you can give it back to me tomorrow"). This was made inside urban area networks. Project reference is here.



The second one is Shared Paths, finding the best possible path in a town (by all means, like (by car, public transportation, on foot). In this case, we talk about accessible paths, with no ramps/curbs (wheelchairs) or no acoustic traffic light (blind people). Reference of project here. There is the automatical detection of crossroads and acoustic signals, over the following context:

- During the day, several days per week, a person does the same path over and over again
  - To go to work, or to the grocery store, or to school, etc.
- This person (especially commuters) often meets the same people

Written by Gabriel R.

- Created an app for mobile phones:
    - Identifies repeated paths
    - Using proximity-based connections (Bluetooth?, wifi?) allows to detect users met during this repeated
    - Allows to contact people that do the same path
    - Upload repeated paths into a server to let the server know that it is a desirable path for similar users (e.g., a person on a wheelchair, someone with heels or a stroller, etc.)

Moving on:

- There is also a project about Road Crossing Recognition through Accelerometer, enriching Web Squared and giving maps with points that can be accessible and recognized. Reference here.

- Again, a project via the use of sensor, catching data and analyzing it via interconnected devices, with Web Squared. Reference here.

- Then, the node detection at a wireless range level (*Project Node Detection*). Reference here.

- *Evaluation of DTN on Android phones*, creating a network fault-tolerant even without connectivity for mobile phones. Reference here.

- *Multiplayer games over Ad Hoc Networks*
    - Users are mobile
        - Risk of disconnection
        - Possible impact on delays & packet losses due to ad-hoc routing protocols and bandwidth fluctuation
    - Radio interfaces necessitate some energy consumption – Risk of node failure (could be dramatic in C/S architecture)
        - Need of new architecture for gaming over MANET

- Other entertainment projects
    - Creation of a game for mobile devices (e.g., Android, iOS, etc.)
        - Even digital versions of old games
    - Creation of a mobile application for cultural heritage
    - Creation of any application for mobile devices (e.g., Android, iOS, etc.) that exploits Wi-Fi, Bluetooth or other connectivity means can be discussed
        - We have smartphones to lend if needed

There can be something very specific based on *Drone Networks, Services and Applications*. Ideas can be:

- Creation of software to manage drones
- Simulations to verify classic MANET protocols (2D topology) in DANETs (3D topology)
- Survey/Analysis of drones in Agriculture 2.0 or Industry 4.0

Note: if the project is bigger than the class, it can count as credits for Other Training Activities or even for the Master Thesis if it's particularly big.

A curious theme (already shown here somewhere) are the *Mobile Ancient Games*, implementing ancient games on smartphones/tablets. Another interesting theme can be the *TCP Versions comparison*:

1. Real experiments with a particular configuration of the network (even just Linux TCP vs Windows TCP)

a) Evaluate fairness / friendliness
b) Evaluate Starbuck's scenario
c) Evaluate mobility impact
d) Error link impact
e) Even just download performance

2. Simulations (as above…)

- with well-known and documented simulators (e.g., NS-2, NS-3)

Otherwise, some *comparison* projects can be made:

- Realistic networks and heterogeneous flows with multimedia
    - Measure different protocols (TCP)
    - Simulations vs real experiments
    - Test with some new application (AR/VR/MR…)

In case, even some *Game Flow Measurement*:

- Measure bandwidth, delay, jitter…
    - Classic online games
    - Thin client games (aka cloud games)
        - OnLive, Stadia, Luna
        - VoIP
    - Simulations vs Sniffers (e.g., Wireshark)

Other ones:

- Scavenger Hunt Game
- Strega comanda colore – Simon Says the Color
- Ubiquitous Social Cam – something like Periscope and look for something in real-time
- Participation 2.0
    - Crowdsourcing regards outsourcing tasks to a public
        - E.g., a participative online activity
    - Through smartphones, with tasks automatically distributed to specific categories based on
        - Profiles, location, sensors on smartphones, current activity, etc
- ARDUECO
    - Creation of a sensor box attached to a bicycle to sense air pollution and collect data
- GeoComments
    - Attaching audio/video/text comments to a specific location – Users have to be in a certain location to see local comments
    - Comments: messages, touristic guides, games, information, art

Another idea can be:

- Take a specific topic and make a survey paper on the state of the art (with comparison)
    - Drone networks (protocols or even applications)
    - Underwater networks
    - Alert propagation in urban (grid) vehicular networks
    - Web Squared applications
    - Internet of Things or Internet of Everything
    - eHealth
    - Smart city
    - Smart traffic/transportation
    - Nanonetworks
    - Network of cubesats

Another interesting application is the recording and analyzing the different voices and conversation between spaces and people, or even analysis of sexist topics inside social media and avoid them. Some solution of avoid sexting/blackmailing and crypt images sent/avoid problems in sending this kind of content.

Any project you choose ask the teacher for further references: e.g., slides/books/examples on programming smartphones.

Other examples are:

- D2D (Device to Device) interaction, providing a comprehensive survey as state-of-the-art approaches (arriving to at most 5 papers):
    - A Survey on Device-to-Device Communication in Cellular Networks (link)
    - Device-to-device communication as an underlay to LTE-advanced networks (link)
    - LTE Direct as a Device-to-Device Network Technology: Use Cases and Security (link)

- UAV (Unmanned Aerial Vehicle) route planning, useful when designing autonomous flight paths for drones. It involves determining the optimal trajectory that a UAV should follow to reach its destination while avoiding obstacles and adhering to any specified constraint
    - Routing military aircraft with a constrained shortest-path algorithm. Military Operations Research (link)

- Body Area Networks, in which the goal is to provide a study regarding state-ofthe-art proposals employing predictive / behavioral algorithms
    - Body Area Sensor Networks: Challenges and Opportunities (link)
    - Active Assistance Technology for Health-Related Behavior Change: An Interdisciplinary Review (link)

- 2D Vehicular Networks; Vehicular communications will become a reality in the near future. A research topic spanning from Physical/MAC (propagation) to data dissemination aspects. Message forwarding in 1D, platoon scenario is well-understood and optimal schemes have been proposed. However, in the general case, the road topology is 2D.
    - A real-time adaptive dissemination system for VANETs (link)
    - Evaluation of flooding schemes for real-time video transmission in VANETs (link)

- 2D/3D Drone Networks; drones and other mobile/statice devices (IoT devices) with communication capabilities are becoming popular. The objective is to provide a comprehensive survey of possible applications and challenges related to 2D/3D Drone networks and to provide a survey of mobility models for the mentioned applications
    - Starting Material
        - https://www.sigmobile.org/mobisys/2016/workshops/DroNet/program.html
        - https://www.sigmobile.org/mobisys/2015/workshops/DroNet/program.html

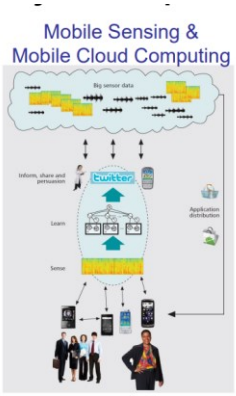
- Seamless Communication, in which no user interaction is required. Pairing process between wireless enabled devices requires initial setup (user intervention). Solutions have been proposed for seamless data transfer. These proposals exploit advertisement frames to exchange data, hence avoid pairing, making the solution user-transparent
    - Two-Way Communication Protocol using Bluetooth Low Energy Advertisement Frames (link)
    - "mumble: Framework for Seamless Message Transfer on Smartphones (link)

Other topics can be:

- Green Computing
- Information-Centric Networks (ICN)
- QoS/QoE in multimedia (specific for smartphones)
- Activity Recognition (e.g., from smartwatches… even used for side channel attacks)

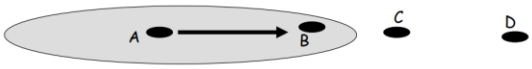
Others project ideas sparse in slides:

- End of WNMA01 set of slides:



And more…

- End of WNMA04.pdf set of slides



- A node decides to intelligently choose a Carrier sensing threshold (T)
- The node senses channel
  - If signal > T, then node does not transmit
  - If signal < T, then transmit
- Possible to guarantee no collisions?

Written by Gabriel R.

- From WNMA06.pdf at slide 16

Class Project Idea:
Describe or test (NS3 simulations?) some routing protocols for MANET, or VANET, or FANET, or Space…

- End of WNMA08.pdf set of slides (left) and end of WNMA-HCW3-ext.pdf set of slides (denoted as "Future work, project, thesis) – (right)

- Take various TCP protocols and test/compare them in a realistic new environment
  - Mobility
  - Starbuck's / Coffee Shop
  - UMTS
  - …
- NS-2/NS-3 simulations or Linux

- Alternative: read and present paper(s) on TCP (or general congestion control) for some wireless environment

- Dynamic lower and upper bounds for Fast-Broadcast's waiting time calculation
  - Based on vehicle density

- Junction identification backup mode
  - Reliance on GPS
  - Compute angle between received messages to identify vehicles within junctions

- Study regarding FANETs (Flying Ad-Hoc Networks)

- Simulations or Real experiments in Antwerp
  - NS3, Omnet, etc.
  - Actual 5G+VANET highway

10 802.11 STANDARDS (WNMA10)

There are a lot of 802.11 standards to consider, each trying to:

- enhance QoS, interoperability with existing technologies
- enhance speed, spectrum occupied, frequency, etc.
- find a fit for particular use cases (VANET, etc.)

Recent standards should be retro-compatible; ahead some of them briefly listed:

- *IEEE 802.11n* (Wi-Fi 4 – new denomination by Wi-Fi Alliance)
    - It introduced MIMO (Multiple Input Multiple Output)
        - enabling multiple data streams for higher speeds and improved reliability
- *IEEE 802.11ac* (Wi-Fi 5) – consider 802.11a introduced $5\ GHz$, offering high data rates
    - Expanded use of MIMO (up to 8)
    - Wider bandwidth
        - $160\ MHz$ more than 802.11n
    - Up to 500 Mbps for each single connection
- *IEEE 802.11ax* (Wi-Fi 6)
    - Wider frequency range
        - 1 and $7\ GHz$, not only 2.4 and $5\ GHz$
    - Up to 11 Gbps in test
    - Lower latency
        - less than half with respect to 802.11n
    - UL MU-MIMO (Uplink Multiuser Multiple-Input Multiple-Output)
- *IEEE 802.11be* (Wi-Fi 7) – consider 802.11b introduced $2.4\ GHz$, proving longer range
    - 3 main frequency bands
        - $2.4, 5$ and $6\ GHz$
    - CMU-MIMO (Coordinated Multiuser MIMO)
- *IEEE 802.11mc*
    - Indoor localization ($1 - 2$ m precision)
        - Using Wi-Fi Round Trip Time (Wi-Fi RTT)

10.1 802.11E

IEEE 802.11e brings QoS (Quality of Service) extension (support for real time apps e.g., VoIP, differentiating services, enhanced access mechanisms, interoperability/compatibility): this is not guarantee of good QoS, since technology must provide specific quality in order to meet that. Originally, it defined exchanges using:

- *DCF (Distributed Coordination Function)*
    - classic approach for 802.11, offers CSMA/CA for transmitting in the channel
- *PCF (Point Coordination Function)*
    - polling, priority centrally controlled, uses beacons and contentions

Now, the following are introduced and used:

- *EDCA (Enhanced Distributed Coordination Function Channel Access)* – Priority Scheme
    - different access categories for multiple flows of data
- *HCCA (Hybrid Coordination Function Channel Access)* – Parametrized QoS Scheme
    - extends PCF with Contention Free Periods (CFP)

Written by Gabriel R.

10.1.1 DCF

DCF is the basic MAC access method, and it *uses CSMA/CA* with this working (in case, look here again):

- Look for activity:
  - if free
    - wait (DIFS + cw = contention window) and transmit if still free
  - if busy
    - random back-off a number of slots (min 31, max 1023)
    - count down slots as long as the medium is not busy
    - when count down $= 0$
      - if packet fails (collision)
        - back-off with increased random window
        - up to a preconfigured upper limit

There are two types of cw in 802.11:

- *"Normal"* contention window
  - random number selected from $[0, cw]$
  - small value for cw
    - less wasted idle slots time
    - large number of collisions with multiple senders
      - two or more stations reach 0 at once
  - optimal cw for known number of contenders and known packet size
    - computed by minimizing expected time wastage
      - by both collisions and empty slots → collisions/empty simultaneous
      - tricky to implement
        - number of contenders hard to estimate
        - it can be *very* dynamic

- *Adaptive* contention window
  - 802.11 adaptively sets cw
    - starts with $cw = 31$
    - if no CTS or ACK
      - increase to $2 * cw + 1$
      - values of threshold: $63, 127, 255$
    - if successful transmission
      - reset $cw = 31$
  - 802.11 adaptive scheme is unfair
    - under contention
      - unlucky nodes will use larger cw than lucky nodes
        - due to straight reset after a success
    - lucky nodes may be able to transmit several packets
      - unlucky nodes are counting down for access and still waiting
  - it does not provide QoS

10.1.2 PCF

As we already saw, PCF is a priority system centrally controlled by a Point Coordinator/PC (usually the AP). Like previous one, in case have a read again here.

- After each beacon transmitted:
  - we have CP (Contention Periods) and CFP (Contention Free Periods)
  - they give a fixed length of time after each beacon, synchronized to beacon's intervals
- Uses PIFS to keep control (Point Inter-Frame Spacing)
  - the time a station must wait after the transmission of a frame
  - shorter than any DCF
- PC keeps list of stations eligible for polling

It also has some drawbacks:

- Fixed to length of time after a beacon, synchronized to beacon intervals
  - not compatible with voice/video streams requiring $10/20/30\ ms$ intervals
- No mechanism to reserve bandwidth or characterize the traffic
- No back-to-back packets
- *It's not used in practice* for these reasons listed above

10.1.3 EDCA - Enhanced Distributed Channel Access

EDCA is a supported QoS mechanism in 802.11e with the following features:

- There are 4 Access Categories (AC) + 8 Traffic Classes (TC)
  - ACs: Voice (highest priority), Video, Best Effort, and Background (lowest priority)
  - Each AC has its own queue, differentiating in the handling of various types of traffic
- Max Service Data Units (MSDU) are delivered through multiple backoffs
  - With one station using specific AC parameters
- Each AC independently starts a backoff after detecting channel being idle for AIFS
- After waiting AIFS:
  - used to prioritize one AC over the other
  - each backoff sets counter from number by interval $[1, cw + 1]$
- The backoff formula is:
  - $cw_{new}[AC] \geq \big((cw_{old}[TC] + 1) * PF\big) - 1$
    - With EDCA video → streams capacity drops and contentions increase
    - With more collisions → larger contention windows
    - With multiple streams → lose some overall bandwidth

Prioritized Channel Access is realized with the QoS parameters per Traffic Classes, which include:

- AC = Access Category
- TC = Traffic Class
- $AIFS[AC]$ = Arbitration Inter-Frame Spacing
  - Interval that a station waits before attempting to access the channel
  - Minimum time between two transmissions → used to avoid collision
- $CWmin[AC]$ (or $CWmax\ [AC]$) = Contention Window min (or max)
  - Range of back-off values that a station randomly selects before attempting to transmit

Written by Gabriel R.

- $PF[AC]$ = Persistence Factor
    - It's used to control the probability of transmitting during a given time slot
    - It influences the likelihood of a station contending for the channel during a CP

The figure traces a comparison between them:

	AC_VO [0] (voice)	AC_VI [1] (video)	AC_BE [2] (best effort)	AC_BK [3] (background)
AIFSN	2	2	3	7
CWmin	3	7	15	15
CWmax	7	15	1023	1023

For each access category, some comments (here ordered from highest to lowest priority):

- AC_V0 (voice):
    - is used as a fast channel, meant to be left soon
    - it's privileged to the others, especially during congestion
    - has a small cw_{max} otherwise there would be collisions
    - it is meant for multiple and concurrent VoIP calls
- AC_V1 (video):
    - supports multiple video streams
    - it is prioritized before other data traffic
- AC_BE (best effort):
    - it is for applications which cannot provide QoS capabilities, such as legacy devices
    - this traffic is not sensitive to latency but is affected by long delays
    - the difference is present only on the initiation phase
- AC_BK (background):
    - this is for low-priority traffic
    - it does not have strict throughput or latency requirements
    - this traffic includes file transfers or print jobs
    - definitely, it this access category you don't care about going fast

In EDCA:

- different Access Categories have multiple queues, each with parameters
    - may lead to virtual collision, introducing virtual differentiation
    - may have different backoffs, each to consider according to the speed
- this is effectively DCF with 4 priorities
    - contention-free access happens during TXOP (Transmit Opportunity) periods
- bursting (more packets at once) is possible only for video/voice for a certain amount of time
- with different min/max backoff slots, one stream has advantage over another:
    - it gives advantage to some types of traffic
        - $> cw$ after backoff
    - such values are configurable via the management interface
        - but choosing the right optimum values for every scenario is not obvious

Basically, EDCA provides for the "prioritization of frames" based on upper-layer protocols i.e. application traffic, such as voice or video.

Written by Gabriel R.

A full overview over EDCA:

- *Pros:*
  - Voice/video are prioritized over data
  - It works well if network is lightly loaded (such as voice-based network)
  - No stream setup instructions required
  - EDCA Power Save is a big advantage with respect to Legacy Power Save
    - this happens because, according to the AC, there is an intelligent scheduling
    - prioritizing classes allows for a better saving according to traffic types
- *Cons:*
  - It is not fair
    - since lower priority still need to be transmitted, possibly causing delays
  - There can be *competition* of streams of the same priority
    - difficult to guarantee high bandwidth, low latency, having problems like jitter, etc.
  - It relies on STAs (Stations)/APs (Access Points) to control priorities and access the medium
    - QoS variations occur in practice

In an oversubscribed channel, small variations in STAs result in throughputs that are not equal. EDCA Admission Control is used to overcome some of these disadvantages and improve situational awareness:

- It introduces to guarantee QoS to EDCA
- It limits admission to an Access Category (Voice/VO and Video/VI)
  - allows only a limited number of peers to access the network
  - limits the latency of QoS streams
  - prevents too many streams such that bandwidth can't handle them
    - this way, the best-effort approach is no longer useful

It works this way:

- AP advertises ACM (Admission Control Method, according to this) bit in Beacon
  - indicating if admission control is required for any Access Category
- STA sends AddTS (Add Traffic Specification) Request Action Frame to AP
  - it is done to perform an admission request to AP
  - which includes a TSPEC (Traffic Specification)
    - set of parameters to define traffic characteristics of specific streams or flows
      - complete list of TSPEC here
    - some useful ones
      - Nominal MSDU (MAC Service Data Unit) size
      - Mean Data Rate
      - Min PHY (Physical) Rate
      - Surplus Bandwidth Allowance (SBA)
- AP runs admission control algorithm
  - then sends back the response to STA using AddTS Response Action Frame
    - $Medium\ Time$
- STA checks $Used\ Time$ over 1 second periods
  - if $Used\ Time > Medium\ Time$
    - then STA must use AC's ECDA parameters

In summary, on Admission Control:

- *Pros*
  - It is an improvement to EDCA in attempt to:
    - contain higher priority streams
    - offer protection to stream already in progress
  - TSPEC requires inputting of basic parameters of QoS streams
    - STAs send the TSPEC

- *Cons*
  - As streams still contend, bandwidth efficiency is not optimum
  - Fairness not guaranteed for streams of the same category

10.1.4 HCCA - Hybrid Coordination Function Controlled Channel Access

HCCA is an extension of PCF with Contention Free Periods (CFP) and allows for Parametrized QoS.

- There is an *Hybrid Controller* (HC) which initiates HCCA and CFP
  - It provides CF-poll (Contention Free Poll = providing limited QoS within the network)
    - done to stations to provide TXOPs (Transmission Opportunities)
    - TXOP increases throughput for high priority data
      - by providing contention-free channel access for a period of time
  - It specifies $start\_time$ and $max\_duration$
    - hence other stations can't access to the medium
  - Station (STA) transmits within SIFS and then using PIFS periods
  - If no transmission after PIFS
    - HC takes over and issues new TXOP
    - otherwise end of CFP
  - CFPs can be synchronized to the individual source traffic intervals
    - instead of the beacon intervals

In contrast to PCF, in which the interval between two beacon frames is divided into two periods of CFP and CP, *the HCCA allows for CFPs being initiated at almost any time during a CP*.

- This is called *Controlled Access Phase (CAP)*
  - A CAP is initiated by the AP whenever it wants to send/receive a frame from/to a station
    - contention-freely
  - During a CAP, the HC, e.g. the AP, controls access to the medium
  - During the CP, all stations function in EDCA
  - In contrast to PCF, Traffic Classes (TC) and Traffic Streams (TS) are defined
    - So, HC can provide a kind of per-session device (not limited to per-station queuing)
      - stream coordination in any fashion (not just round-robin)

There is a specific mode in ECDA called WMM-SA, which stands for Wireless Multimedia - Scheduled Access. It operates by allocating specific time slots for transmission of different types of multimedia traffic, thereby ensuring predictable and timely delivery of data packets.

In summary:

- *Pros*
    - Efficient use of bandwidth
        - CFPs used
        - Returns channel as soon as packets sent for that TXOP
    - Guarantees latency
        - Important in high bandwidth streaming applications
        - Regularly grants TXOPs as required by the TSPEC
    - "Guarantees" bandwidth
        - For quality video stream, for example, data rate must be assured
        - More efficient use of ECDA
            - Very efficient use of available bandwidth thanks to CFPs
            - Number of simultaneous voice calls is much higher than WMM
                - Wireless Multimedia, due to limited back-off slots
    - Overcomes most OBSSs – Overlapping Basic Service Sets → areas where APs overlap
        - All STAs and APs that hear the QoS poll will obey the TXOP
        - ACKs from QSTAs should include Duration Field with outstanding TXOP time
            - extending range of CFP to other networks

- Cons
    - It requires a complex scheduler
        - including added complexity
    - Overlapping HCCA networks do have TXOP problems
        - this is being solved in 802.11aa – prioritizing traffic granularly with ACs

In general, for the QoS requirements:

- 802.11 can be considered as:
    - EDCA Admission Control
    - HCCA
- Both schemes require TSPECs
- TSPECs require knowledge of certain parameters of the desired QoS stream, at least:
    - Nominal MSDU size
    - Mean Data Rate
- For HCCA
    - Maximum Service Interval

Compared to the previous one, it gives the access point the ability to provide for "prioritization of stations". In other words, certain client stations will be given a chance to transmit before others. To note, HCCA has never been adopted by WLAN vendors.

10.2 802.11N

This one enables new consumer and enterprise applications (such as video distribution, more bandwidth for QoS applications, greater range, throughput, etc.). Also:

- It is able to achieve High Data rates ($64 - 600$ Mbps)
  - Up to 300 Mbps with 2 MIMO devices on a $40\ MHz$ channel
- Legacy mode support
  - Allows support for multiple technologies at the same time for STAs (a/b/g)

It introduces new components:

- *PHY Enhancements*
  - They are applicable to both $2.4\ GHz$ and $5\ GHz$
  - It supports OFDM (Orthogonal Frequency Division Multiplexing) with many coding methods
    - Technique used to transmit parallel data
      - dividing a high bitrate transmission in many parallel and orthogonal flows
      - much lower, avoiding propagation effects
  - MIMO Radio Technology with Spatial Multiplexing
    - Takes advantage of simultaneous transmission using multiple antennas
  - High throughput PHY at $40\ MHz$ Channels
    - Adjacent channels can be combined together to achieve this rate

- *MAC Enhancements*
  - Two MAC aggregation methods are supported to pack smaller packets into one MPDU
  - Block Acknowledgement
    - Optimization to combine an 802.11 ACK with unicast frame sent later in time
    - Combined ACK → no need to ACK every packet

10.2.1 MIMO/SISO

Over this, we can characterize MIMO (Multiple Input Multiple Output), which allows to transmit and receive with multiple radios simultaneously on the same spectrum.

These kinds of systems have the following improvements:

- Outages reduced by using information from multiple antennas
- Transmit power can be increased via multiple power amplifiers
- Higher throughputs possible
- Transmit and receive interference limited by some techniques

There is also the SISO (Single Input Single Output) counterpart, which was favored for simplicity and low-cost but have some shortcomings (here, both transmitter/receiver equipped with a single antenna):

- Outage occurs if antennas fall into null
    - Switching between different antennas can help
- Energy is wasted by sending in all directions
    - Can cause additional interference to others
- Sensitive to interference from all directions
- Output power limited by single power amplifier

11 CASE STUDIES (THCW1 – THCW3)

I don't think these one have that much relevance, I think they are presented for the sake of completeness. Consider probably the third one is the only one useful, given there is another chapter related to that.

11.1 TOWARDS A HYPERCONNECTED WORLD OPPORTUNITIES AND CHALLENGES

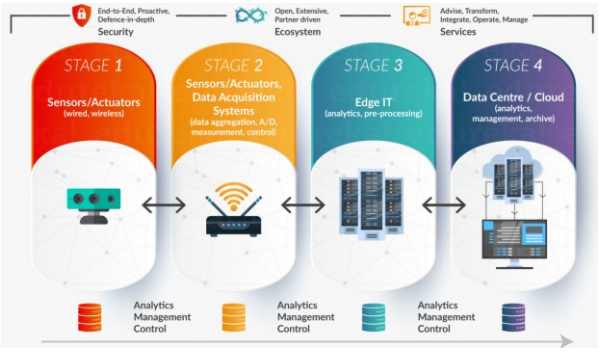
The exponential growth of *connected devices*, big data and *edge/cloud computing* is creating an hyper-connected world, fostering *innovative* use cases, *opportunities* and *challenges* in our society and daily lives.

- This *hyper-connected scenario* depends on the capability to integrate technologies
  - such as IoT, Mobile-to-Mobile communication, connected vehicles/drones
  - cloud computing, edge computing, data gathering/dissemination and social networks
- The mobile experience is truly expanding everywhere:
  - connecting billions of mobile devices and crafting more and more interconnected devices
  - giving great performances, high-quality multimedia, very high broadband speeds
  - all inside one pocket, composed by layers of sensor and connection protocols
- *Connectivity* is the foundation of a great mobile experience
  - the key is deliver rich experiences, fast, longer, reliably, real-time, on-the-go

The main device of this mobile revolution is definitely Internet of Things (IoT)

- This is able to connect everyday things embedded with electronics, software and sensors to the Internet enabling them to collect/elaborate and exchange data
- It all started from Internet connection from people, called *Internet of People (IoP)*, with billions of people and devices connected at the same time

There are many stages in IoT architecture and for each Analytics Management Control is involved:



These devices collect data continuously, sharing/reusing between devices and continuously learn (in a "collect-share-learn" cycle, both free/paid).

Let's consider this *Smart Home Scenario*.

Here interactions happen in *Sensing-as-a-Service Mode* (IoT sensing capabilities via devices/sensors/cloud infrastructure, available as a public service).

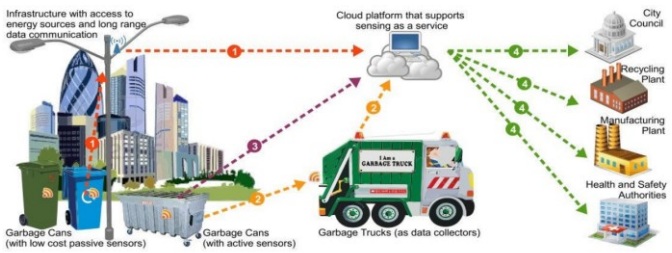
This on the right is a Smart Fridge and works with the following indications.

Written by Gabriel R.

- Smart fridge contains:
  - *RFID Reader Interaction*
    - Smart fridge with RFID reader scans RFID-labeled items for identification
  - *Temperature Sensor Monitoring*
    - Monitors internal temperature; alerts if it exceeds predefined thresholds
  - *Data Flow Management*
    - Local Processing (Edge IT)
      - Processes data within the fridge for item identification and temperature.
    - Data Aggregation and Analysis
      - Aggregates and analyzes data within the fridge.
    - Communication with Cloud
      - Periodically sends data to the cloud for further analysis connecting to the near AP to the cloud platform which supports the service
- *User Interaction*
  - Interaction via a mobile app, checking temperature, viewing inventory, and receiving alerts
- *Benefits*
  - Enhances food safety and freshness
  - Automates inventory tracking through RFID
  - Provides proactive user notifications and insights

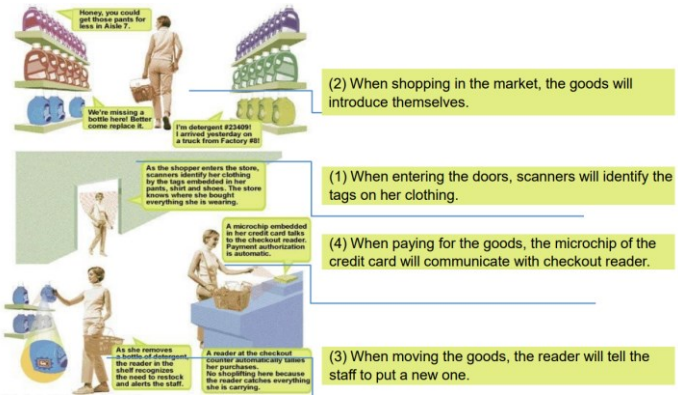
Another example is the *Efficient Waste Management*, again supported by the Sensing-as-a-Service:

- *Smart Bin Infrastructure*
  - Utilize smart bins with low cost/active sensors for waste levels, weight, and lid status
- *Data Collection and Local Processing*
  - Sensors collect data locally, with each smart bin processing information and will be collected by garbage trucks
- *Wireless Connectivity*
  - Smart bins communicate wirelessly, forming a network for data transmission.
- *Data Aggregation and Analysis*
  - Centralized system aggregates and analyzes data to identify waste patterns.
  - Cloud computing enhances waste management with advanced analytics, informing authorities, plants and management departments
- *Alerts and Notifications*
  - Real-time alerts notify collection teams based on bin capacity and patterns
- *User Engagement*
  - Citizens engage through apps, receiving updates on schedules and recycling tips
  - Data-driven optimization improves collection routes, minimizing environmental impact



Written by Gabriel R.

Another interesting IOT Application Scenario is *Shopping*:



- *Entering the Store*
    - As the customer enters the store, scanners at the entrance automatically identify RFID tags on the clothing
    - This enables a personalized shopping experience and helps the store track customer movement for analytics
- *Goods Introduction*
    - While shopping, when the customer picks up a product, RFID tags or smart labels on the goods trigger embedded sensors
    - These sensors introduce the product, providing information such as price, ingredients, or promotions to the customer's mobile device via a shopping app
- *Moving Goods*
    - As the customer considers a product and places it back on the shelf or picks up a different one, the RFID reader detects this action
    - The system can then notify store staff to restock shelves, ensuring that products are readily available and reducing the chances of stockouts
- *Checkout Process*
    - When the customer is ready to make a purchase, the microchip embedded in the credit card communicates with the checkout reader
    - This enables a swift and contactless payment experience, enhancing convenience for both customers and the checkout process

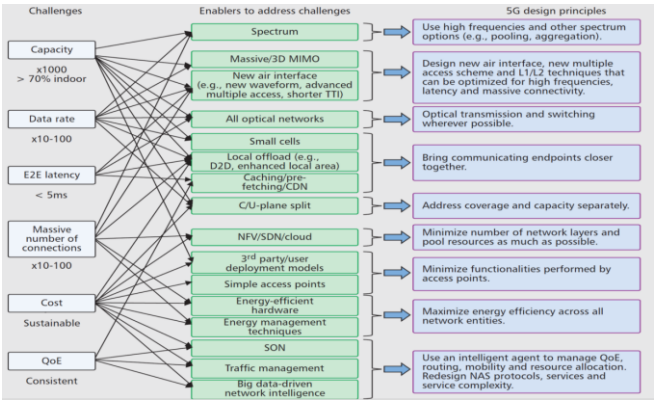
Data is collected through a variety of different applications and services and then shared when needed.

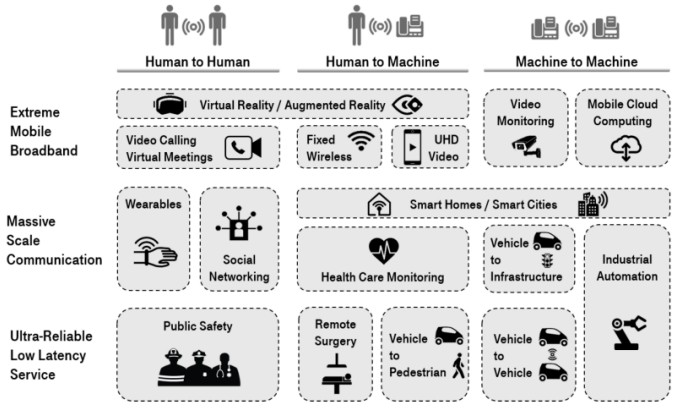
- The key is seeing data like a collaborative research supported by *Sensing-as-a-Service Model*
    - This expands to the conception of networks within networks
    - Effectively becoming the Internet of Everything
- This way networks can *integrate simultaneously in a fixed and heterogeneous way*
    - even with varying transmission characteristics
    - considering multiple hand-over from in-house connections, up to campus-based/metropolitan and regional networks
- There is an on-going research on satellites for projects like *CubeSat*
    - a miniaturized satellite often employed for space research and exploration
    - highlights the scalability and connectivity possibilities of modern space-based technologies
    - these networks are typically composed of multiple nano-satellites, each serving a specific function or mission objective
        - this modular design allows for gather data, monitor celestial phenomena, or facilitate inter-satellite communication (Internet of Space Things – IoST)

This way, we reached 5G, a transformative evolution in wireless communication which is becoming prominent over time both in traffic and subscriptions like mobile evolution itself, mainly based on these aspects according to IEEE:

1. *Millimeter Waves*
    - High-frequency waves ($24\ GHz$ to $100\ GHz$) enable faster data transfer, using abundant bandwidth, diverse spectrum options (e.g., pooling, aggregation)
2. *Small Cell Networks*
    - Compact, strategically placed cells improve coverage and capacity to make endpoints communicate closer, thanks to offload, caching and addressing coverage/capacity
3. *Massive MIMO*
    - New air interfaces (waveforms), new multiple access schemes, different level techniques for high frequencies, latency and massive connectivity
4. *Support for massive number of connections*
    - Minimizing number of network layers and pool resources as much as possible
5. *Cost sustainability*
    - Maximizing energy efficiency across all network entities
    - Minimizing functionalities performed by access points
6. *Quality of Experience (QoE)*
    - Use an intelligent agent for Traffic Management and resource allocation

The following are 5G Challengers and Enablers and 5G Use Cases:





Other 5G cases discussed/shown:

- eMBB (Enhanced Mobile Broadband), used for HD/AR/VR apps
- URLLC (Ultra Reliable & Low Latency Communications), used for remote robot control, connected autonomous vehicles, interactive gaming
- mMTC (Massive Machine-Type Communications), used for smart cities and smart agriculture
- Connectivity V2V/V2X (Vehicle to Vehicle/To Everything)
- Robot Surgery, Tactile Internet
- 4th Industrial Revolution (era of connectivity, advanced analytics, automation, and advanced-manufacturing technology)
- Automation of Everything
- Interactive Games & AR/VR/MR

This drives us even towards 6G, promising real-time immersive multimedia, microsecond latency and lightning-fast data speeds at the blink of an eye between devices of all kinds (ubiquitous), giving super-precise positioning, being critical towards a more sustainable development.

Written by Gabriel R.

11.2 Case Study Analysis: Autonomous Production Site

Small robots are more and more used in industry.

- They have become famous thanks to Amazon's and Alibaba's warehouse (pick up) robots
- Are now used even in productions sites
    - E.g., to bring semi-finished products through their stages of production
- Operators have plans to automate almost every physical move in their facilities in next years
- Robotic automation can extend the capacity, hours of operation, and life of a production site
    - And do not complain about wages

This regards in particular *Automated Guided Vehicles (AGV)*:

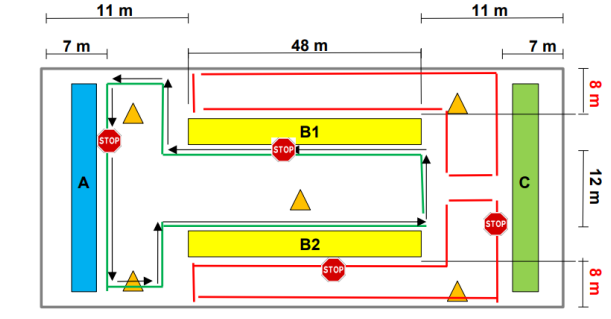
- These are portable robots that follow markers or wires on the floor, or use vision, magnets, or lasers for navigation
- They have been used for case, pallet, bulk, or specialized container movement for decades across a wide range of industries and applications
- It's not immediate to understand how many AGVs can be supported by current wireless network technology in a production site
    - The goal understand the limitations of an AGV-based site in terms of network capability
    - This will assess a sustainable number of AGVs, creating a good network to be reliable

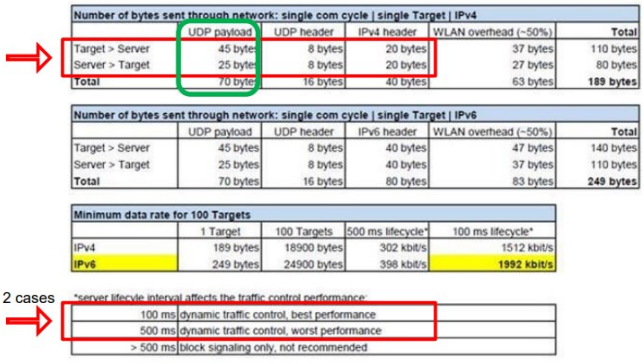
There are two (control) cases for AGVs:

- Dynamic Traffic Control – *fine* grained (best performance)
    - One message sent/received by any AGV every $100\ ms$
    - More messages allow for fine remote control of AGVs
        - dynamic decisions about movements and task orders (no fixed paths)
    - More messages may congest the network
        - If the network becomes congested
            - messages are lost/delivered with delay, jeopardizing the performance

- Dynamic Traffic Control – *coarse* grained (worst performance)
    - One message sent/received by any AGV every $500\ ms$
    - Less messages only allows for static routes of AGVs (virtual/painted rails)
    - Less messages maintains the network uncongested

We assume this movement scenario, in which an area is considered over a fixed speed, then stops and movements will be assigned. On the side, you will see the bytes transmitted and the network traffic:





Number of bytes sent through network: single com cycle | single Target | IPv4

	UDP payload	UDP header	IPv4 header	WLAN overhead (~50%)	Total
Target > Server	45 bytes	8 bytes	20 bytes	37 bytes	110 bytes
Server > Target	25 bytes	8 bytes	20 bytes	27 bytes	80 bytes
Total	70 byte	16 bytes	40 bytes	63 bytes	**189 bytes**

Number of bytes sent through network: single com cycle | single Target | IPv6

	UDP payload	UDP header	IPv6 header	WLAN overhead (~50%)	Total
Target > Server	45 bytes	8 bytes	40 bytes	47 bytes	140 bytes
Server > Target	25 bytes	8 bytes	40 bytes	37 bytes	110 bytes
Total	70 bytes	16 bytes	80 bytes	83 bytes	**249 bytes**

Minimum data rate for 100 Targets

	1 Target	100 Targets	500 ms lifecycle*	100 ms lifecycle*
IPv4	189 bytes	18900 bytes	302 kbit/s	1512 kbit/s
IPv6	249 bytes	24900 bytes	398 kbit/s	**1992 kbit/s**

2 cases

*server lifecyle interval affects the traffic control performance:

100 ms	dynamic traffic control, best performance
500 ms	dynamic traffic control, worst performance
> 500 ms	block signaling only, not recommended

Written by Gabriel R.

It was tested on IEEE 802.11n (less prone to inter-channel interferences compared to other ones like 802.11g), with each AP using a different channel and messages sent to/by each rover every $50\ ms, 100\ ms$ or $500\ ms$. The performance metrics were the following:

- Packet loss
    - Measured at the application layer (end-to-end)
    - The *acceptable* packet loss rate depends on the criticality of exchanged message content
- Message delay
    - Must be strictly lower than the operational cycle
        - otherwise the exchanged operational data might not be valid anymore
    - Measured delay corresponds to the time added by the last wireless tier

This is a *mobile* scenario:

- this is modeled according to the specifications
    - rovers move at $1\ m/s$ and occasionally stop
- studied different deployment strategies
    - first, uniformly, then using connection with other APs
- different network performance metrics measured at the app layer
    - packet loss
    - end-to-end delay

Measuring in a $100\ ms$ duty cycle:

- in all configurations there are very few packet losses
- the uniform AP per deployment is better for the E2E (End-to-End) for AP/Rovers
    - the more rovers, the more the delay, adapting at the specific network conditions

In summary:

- Considering the per-packet delay at the *Access Point/rovers* (packets traveling toward the rovers/AP) there were few added delays
    - also, there is no queuing up; so the system can sustain the network traffic
- there seem to be no problem in handling the considered scenario
    - even considering different interdeparting time
    - results are even better

To test the limit of the system:

- it was considered the case with a single AP and many nodes (rovers) connected to it
    - called *static scenario*
- there were considered various configurations with different number of nodes (not moving)
- bandwidth consumption is much higher than throughput due to channel contention mechanisms

On normal duty cycle ($100\ ms$) for packet loss:

- even without much traffic
    - the more nodes, the more the increase in packet loss and loss of bandwidth
- for both APs and Rovers, eventually, the buffer will be all occupied
- then, some pushing in the payload data was tried
    - it can be seen that for both APs/rovers, the delays are skewed after a while

Recent APs use improved techniques such as *Adaptive Rate Managers*: with them results improve. The Adaptive Rate Manager can be described as follows:

- A table of acknowledgement estimates is maintained per neighbor per physical layer rate
- The *ratio of transmission attempts to acknowledgements received*:
    - maintained using an exponential weighted
    - moving average to smooth the probability estimation
- On a frequent basis:
    - the table is scanned to find an approximation to the best performing rate and retry chain
    - that is used for transmission for the next interval
- With a moderate frequency:
    - frames are selected to probe presently unused rates
    - feedback from those probe frames maintains the probability estimates for unused rates
    - that can be chosen if needed

In conclusion:

- with an adaptive rate manager performance improves
- again, delays tend to become more stable after a while, generally
- for greater number of AGVs
    - the AGVs must rely on less communication with a central station
    - giving up on some flexibility
    - increasing the level of movement decisions that can be taken autonomously by the AGVs

11.3 Case Study Analysis: Vehicular Networks

There is more technology on vehicles now than ever before:

- for example, TV-DVD system, GPS/Navigators,
- connectivity available on board (4G, 5G, etc.)
    - bringing the development of IEEE 802.11p standards custom

Vehicles are more endowed with technology for communication, safety, work, entertainment, etc.

- Intelligent Transport Systems (ITS) are described
    - umbrella term for a range of technologies
        - including processing, control, communication and electronics
        - applied to a transportation system
    - main motivations for those are
        - safety
        - transport efficiency
        - the potential development of automotive
        - the huge waste in time and fuel because of traffic jams
            - reducing congestion and emissions
            - creating more secure organization
        - many wireless applications are being developed because of that
            - mainly assistance/emergency/warning/data transfer systems

In those, *Safety & Driver Assistance Systems* are discussed:

- we talk about Active and Preventive Safety & Advanced Driver Assistance Systems (ADAS)
    - which will become connected and see a larger penetration in Europe over the next 20 years
    - move towards a zero-accident society
- they have *location-based services* (LBS) and vehicles connected between them
    - to exchange data real time traffic/travel information thus avoiding negative effects
    - Human-Machine Interface (HMI) will become more restrictive to avoid negative effects
        - we talk about human interactions in real-time, even AR/VR

On this, *Driver To System Responsibility* is discussed:

- given the increasing software requirements
    - increasing computing power needs and sensor needs
    - increasing functional safety requirements
- we do not know for sure if the vehicle is responsible or the driver himself (law/moral dilemma)

Eco-Driving represents the largest potential for energy-efficiency:

- requires permanent feedback after training
- increasing fuel prices will push (slow) behaviour change
- eco-navigation and eco-routing become more important
- pressure to reach 25% energy saving potential by 2025
- *multi-modality* will develop over the next 20 years
    - first in cities to combine private and public traffic
    - then extended to more traffic modes and region
    - all vehicles will be connected with different medias
    - thus interconnected with all forms of entertainment/infotainment

There needs to be exchanged *Real time Traffic/Travel information* some way, for example:

- *Extended Floating Car Data (XFCD)*, traffic info platform
    - vehicles with systems (mobile or embedded)
    - access to standardized Real Time Traffic Information (RTTI)

Smart connected electromobility will definitely develop, with goals to:

- accelerate market penetration of electrical vehicles (incl. hybrids) by 2020 around 7 million vehicles and increasing
    - vehicle park in Europe around 250 million units
- play an important role in offering efficient, clean, zero-emission urban mobility
- get to Fully Electric Vehicles on the European market
    - Hybrids and plug-ins in the transition

From electric interconnected vehicles, we soon get towards *automated driving*:

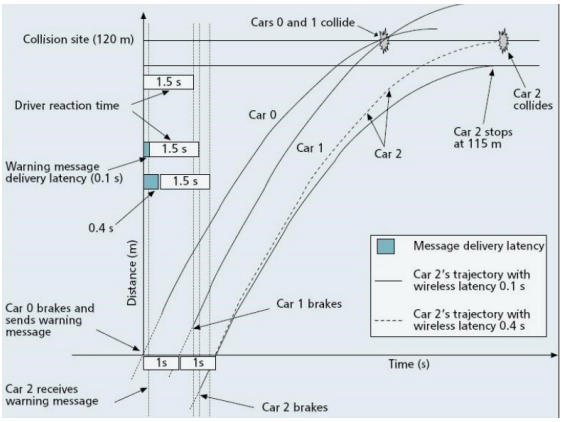
- a system present in a vehicle that allows it to drive itself from one point to another without assistance from a driver (e.g. lane keeping assistance/lane deviation systems)
- The driver needs assistance in increasingly complex traffic situations
    - The highly automated vehicle could take care of some driving tasks to make his/her task easier
    - Majority of accidents (> 90%) is caused by human error
        - The human is not always making optimal driving choices for efficiency either
    - Increasingly seen as the only long-term option

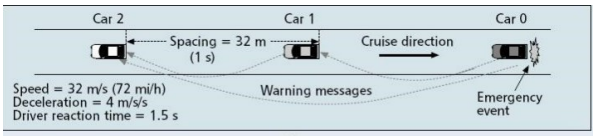
Written by Gabriel R.

An application of all of this is *Safe Driving*:

- Humans have limited sight and reaction time
- Vehicle safety applications employs Alert Messages to overcome human's limitations
    - Avoid multiple accidents
- In case of accident or abnormal behavior of a vehicle, alert messages are broadcast from the Abnormal Vehicle (AV) to following ones

The following is a scenario employing alert messages useful to be analyzed:





- Alert Messages have to be delivered very quickly to all cars following the Abnormal Vehicle (AV)
- Problems arise from multiple transmissions in case of accidents:
    - multiple AVs
    - chain reactions
- Various proposals to reduce multiple (and redundant) transmissions
    - Use only specific nodes to propagate the alert message
        - e.g., the farthest ones from the sender

In each case, the *optimal* (distance-cost) path to forward the Alert Message should be covered in as less time (= as few hops) as possible. Typically, there are two phases:

- *estimation phase*
    - vehicles exchange few hello messages to collect information
    - in order to *estimate their own tx (transmission) range*
- *broadcasting phase*
    - the tx range estimation is put to effective use
    - to reduce the number of hops an Alert Message will experience in its trip to destination

In fact, in Fast Broadcast, there are two Kinds of Messages:

- In *Hello Messages*
    - information to estimate tx range (hearing distance = how far can one hear)
- In *Alert Messages*
    - sender's transmission range (= estimate the range distance)

In conclusion:

- Such systems reveal quite effective in reducing accidents in platoons of cars
- Legally, who is responsible? The "driver", car's owner, car's manufacturer, software developer
- Morally, what should the self-driving car do to avoid as much damage as possible?

12 VEHICULAR AD-HOC NETWORKS (WNMA-HCW3-EXT)

The topic is IEEE 802.11p:

- it was designed to *provide connectivity to vehicles*
  - since 4G technology available not so crucial anymore
  - still useful for safety/public applications (we talked about these already here if you recall)
- there are parts taken from 802.11a and 802.11e (the one with priorities) versions
- there are improvements on range/speed of dedicated licensed bands ($5.9\ GHz$)
  - up to $1000\ m$ of transmission range (Wi-Fi gets $100\ m$ at most remember)
  - up to $26\ Mbps$ of data rate
  - capable of transmitting at $6\ Mbps$ at $300\ m$ of distance even if node travels at $200\ Km/h$
- previous technologies on this front are (note: just quoted in slides, expanded for more context):
  - *WAVE* – Wireless Access in Vehicular Environments
    - set of standards and protocols defined for vehicular communication systems
  - *Dedicated Short-Range Communications (DSRC)*
    - key aspect of WAVE
    - short-range communication between vehicles and roadside infrastructure
    - supporting various safety and non-safety applications

Consider the *Safe Driving* scenario, already present here. In this one:

- Alert Messages have to be delivered very quickly to all cars following the *AV (Abnormal Vehicle)*
- Problems arise from multiple transmissions in case of accidents:
  - transmission range changes
  - transmission needs to be adapted
  - there can be problems in synchronization among vehicles
- This can lead to multiple/redundant transmissions or chain reactions

12.1 SYSTEM MODEL

The proposal to solve the problems above is a *Vehicular Collision Warning Communication (VCWC)*:

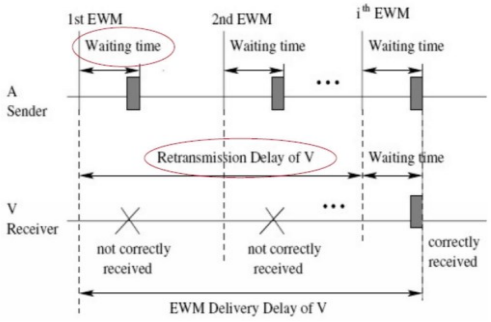
- it presents a communication mechanism to warn vehicles when an abnormal situation occurs:
  - so that they can stop before crashing
- many *application challenges* to consider:
  - stringent delay requirements immediately after the emergency
  - support of multiple co-existing AVs over a longer period
    - transmissions from co-existing AVs interfere with each other
    - chain effect and persistent transmissions generate co-existence of AVs
  - differentiation of emergency events and elimination of redundant EWMs
    - *Emergency Warning Messages*
      - faster than any normal human reaction
      - these kinds of messages are sent in broadcast
        - could lead to an explosion in case of multi-hop path
      - they contain multiple information
        - i.e. geographical location, speed, acceleration and moving direction

- busy tone on other frequencies to stop non-EWM transmissions
- no need to further transmit EWM after some time if following vehicles received it

In this, each vehicle is able to obtain its own absolute and relative position and is equipped with at least one wireless transceiver.

The rate decreasing algorithm helps to achieve low EWM delivery delay at the time of an emergency event, with the presence of a large number of co-existing AVs.

The key issue is to determine how the EWM transmission rate should be decreased over time, considering waiting times and retransmission delays in synchronization.

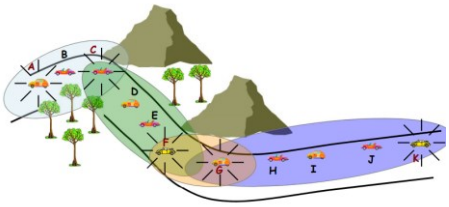


Given the problems we discussed, a *system model* with these characteristics is introduced:

- high mobility of nodes
- variable transmission range
  - erroneously ignored in most papers (simulations)
- a car cannot be sure to be the farthest car receiving that broadcast message

A problem arises:

- *who has to forward the alert message?*
- a simple example is shown here on the right
  - optimal path = transmit with few hops as possible
  - if the message has already been rebroadcast by a following vehicle
    - do not forward it
    - because it could be redundant
  - a way to reduce the message retransmission is to use a sort of back-off mechanism
  - otherwise set vehicles contention window with an inverse proportion of the distance from the sender
    - unrealistic assumption that there is an constant, known-a-priori transmission range



• Optimal solution: A, C, F, G, and F broadcast the alert message (one message and 4 hops)

12.2 Approaches

There are some approaches to be followed:

- Minimum Connected Dominating Set (MCDS)
  - It's the minimum cardinality set of connected nodes
    - such that each other node in the network is connected to a node of the MCDS set
    - those nodes would be the forwarders
  - Only node in the MCDS has to broadcast the message
  - *Optimal but non feasible solution*
    - it needs creation of overlay structure to cover the network
    - perfect knowledge of network not possible
  - Needs complete and updated knowledge of the network topology

Written by Gabriel R.

- It is dangerous because of deterministic failure → redundancy is needed
- A practical implementation with n nodes needs $O(n\ log\ n)$ control messages
- *The solution does not consider the number of hops a message has to traverse*

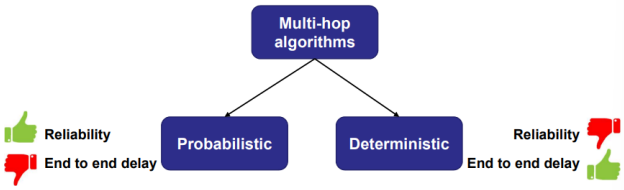
- Redundancy Avoidance (RA)
  - More practical approaches
    - Based on a *backoff* mechanism
      - to reduce frequency of message transmission
        - in case of collisions due to congestion
    - If the message has already been rebroadcast by a following vehicle
      - do not forward it (it would be redundant)
  - *Also this schema does not consider the* number of *hops that a message traverses*

- Jamming Signal (JS – also called Urban Multi-hop Broadcasting Protocol)
  - It uses *jamming signals* (or busy tones) *to determine the next forwarder*
    - This is a blocking/disturbing signal reducing SINR
      - transmitting on same frequency and modulation of signal to bother
  - Vehicles receiving an alert message emit a jamming signal
    - for a time that is proportional to the distance from the sender
  - The last vehicle stopping the jamming signal knows it is the last one
    - it forwards the alert message
  - *Jamming signal phase* delays the transmission of the message
    - *not suitable for alert messages* (we don't know the exact length)
    - the solution is overall pretty slow

- Contention Window (CW)
  - Vehicles could set their *cw* inversely proportional to the distance from the sender
    - no control traffic generated
      - far = very fast/high chance being forwarders
      - near = very slow/lower chance being forwarders
    - assumes that there is a constant transmission range for all cars in every moment
    - it needs to be determined in order to find the last one

12.3 Fast Broadcast

There are two types of these algorithms (multi-hop propagation = furthest forwarder):



- *probabilistic*
  - Pros
    - reliable = leverage redundancy and multiple transmission attempts
  - Cons
    - end-to-end delays to consider
      - due to their probabilistic nature
      - there might be instances where the message takes longer to propagate
        - impacting the overall delay

Written by Gabriel R.

- *deterministic*
    - Pros
        - different types of end-to-end delays to consider
    - Cons
        - reliability = additional overhead in delivering messages
            - while deterministic protocols provide certainty in message delivery
            - they might introduce delays compared to non-deterministic approaches

A possible solution (probabilistic) is Fast Broadcast, designed to have Alert Messages covering the area-of-interest in the least achievable time (as few hops as possible). This part comes from here, I saw.

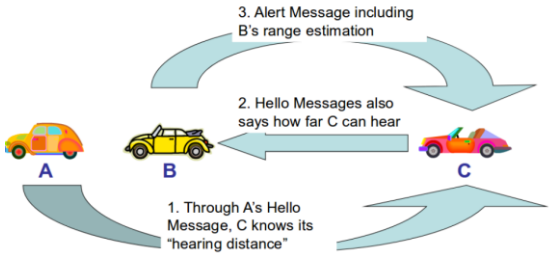
It works in two phases:

1. *Estimation Phase* (estimation of tx range)
    1. vehicles exchange few Hello Messages
        1. to collect information in order to *estimate their own tx range*
2. *Broadcasting Phase* (send Alert Messages and in those is present tx range)
    1. the tx range estimation is put to good use
        1. to *reduce the number of hops* that *an Alert Message will experience*
        2. making the trip to destination as fast as possible

Going into the detail of the phases present here:

- Estimation Phase
    - It continuously runs (little overhead)
    - Time is divided into rounds
    - *One Hello Message* randomly sent every time round
        - if someone faster = abort transmission
    - Hello Messages contain:
        - the sender's position
        - the maximum frontward distance
            - from which another vehicle has been heard transmitting an Hello Message
    - Two kinds of messages used:
        - *Hello Messages*
            - information to estimate tx range (hearing distance = how far can one hear)
            - this usually contains CMBR (see ahead)
        - *Alert Messages*
            - sender's tx range (= estimate the range distance)
            - their max range is $MaxRange = CMBR$ (see ahead)
    - Two kinds of variables used:
        - *CMBR* (Current Maximum Backward Range)
            - computed by hearing "Hello" from the back → "I heard you"
            - estimation of the maximum frontward distance from which another car
                - along the strip-shaped area of interest can be heard by the considered one
        - *CMFR* (Current Maximum Frontward Range)
            - computed by hearing "Hello" from the front → "I could hear you"
            - estimation of the maximum backward distance
            - at which the considered car can be heard

Consider the scenario represented by right figure. When a car receives a message it can be from two directions:



- in front of it
  - in that case it's an update
- behind it
  - there is the update of tx range

That's where the above variables are used, considering all the factors described.

- <u>Broadcast Phase</u>
  - *Alert Messages* are generated by an AV (Abnormal Vehicle)
    - They are sent in broadcast to warn following vehicles
    - They also include the estimated tx range for that hop
  - A node receiving the *Alert Message* waits a time:
    - it is proportional to the node's position
      - in reality it's related to the distance = proportional to contention window
    - with respect to the estimated maximum tx range
      - if near → takes more time
  - cw is computed as follows:
    - AV broadcasts an Alert Message containing
      - the Estimated Maximum Transmission Range ($MaxRange = CMBR$)
      - the position
    - Cars forward the Alert Message after a contention window calculated as follows:

$$\left\lfloor \left(\frac{MaxRange - Dis\tan ce}{MaxRange} \times (CWMax - CWMin) \right) + CWMin \right\rfloor$$

    - If another car farther from the source than already forwarded the Alert Message
      - then the considered car aborts its sending procedure

This was experimented over different schemes and conditions, in particular:

- Fast Broadcasting
  - Exploits our tx range estimator
- Static300
  - It considers $300\ m$ as a fixed parameter for the tx range
  - Ideal iff the actual transmission range is indeed $300\ m$
- Static1000
  - It considers $1000\ m$ as a fixed parameter for the tx range
  - Ideal iff the actual transmission range is indeed $1000\ m$

Comparing their results, we can observe the following:

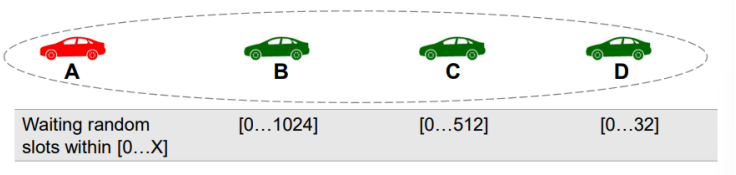
- Static300 has a lot of collisions for $1\ Km$ of transmission range
- Static1000 performances are in general very poor
    - A lot of slots waited to propagate the broadcast message
    - Good number of hops reachable, too many delays
- Fast Broadcast seems to be the best solution
- These results are due the fixed nature of the contention windows of Static300 and Static1000
    - After a certain distance → same contention window for farther cars
    - With small contention windows there is a surge of collisions

In summary:

- Interferences caused by environmental conditions and cars' mobility modify transmission ranges
    - Impact on performance of broadcasting algorithms
- Fast Broadcast is a multi-hop broadcast protocol for vehicular networks
    - Estimates the max transmission range with few Hello Messages (reduce delivery time)
    - Minimizes the number of hops to be traversed and retransmission during broadcast activity

Algorithms, as discussed briefly at the beginning of this subsection, can be divided into:

- Probabilistic
    - *Fast Broadcast*
        - Dynamic transmission range estimation
            - No need to know it a priori, as often assumed in other protocols
        - Estimation Phase
            - Vehicles exchange small Hello Messages (beacons)
                - to estimate their transmission range
            - 1 Hello Message sent every $BeaconInterval$ (e.g., $100ms$)
                - within each transmission range
        - Broadcast Phase
            - A vehicle that wants to transmit waits some random slots $[0 \dots x]$
            - Different vehicles have different random slots
            - Via Hello Messages and variables discussed
                - tx range and messages influence the overall communication



- Deterministic
    - <u>ROFF – RObust Fast Forwarding</u>
        - Multi-hop deterministic delay-based broadcasting protocol
            - Estimation Phase
                - Each vehicle sends a Hello Message every $BeaconInterval$
                    - e.g., $100ms$
                - Neighborhood discovery process

- Broadcast Phase
    - Vehicles have different priorities and different waiting times
    - Neighbors are overtime discovered continuously
- Solves problem of unnecessary delays
    - This is done via use the waiting time
        - inversely proportional to its forwarding priority
    - It uses a bitmap measuring empty spaces between vehicles

Priority	*3*	*2*	*1*
Waiting time	2	1	0

As a test on this, Padua and Los Angeles were analyzed as simulation, with thousands of vehicles at a time. Comparing the results:

- ROFF (deterministic) has less end-to-end delay in case of few vehicles
    - Fast Broadcast, in turn has a lot more
- Redundancy measured on forwarding node numbers is pretty equal between ROFF and Fast Broadcast
- Results were taken with thousands of buildings and considering different shadowing models
    - Without buildings, ROFF and Fast Broadcast are equal
    - With buildings, after $500\ m$, ROFF/Fast Broadcast tend to have almost 100% coverage
- Using vehicles or repeaters at *junctions* as forwarders improve
    - The model tries to exploit such involving vehicles in polygons
    - Inside of them, vehicles within participate in a second contention
    - Such extension can be applicable to both methods (SJ – Smart Junction)
        - SJ-Fast-Broadcast/SJ-ROFF
    - Overall methods with junctions reach much better coverage in all situations
    - With smart junction, better coverage is paid with more messages transmitted
        - those at intersections
        - hence, better redundancy
- Hello Messages has a much greater impact on deterministic algorithms (ROFF) than probabilistic ones (Fast-Broadcast)
- ROFF is much more subject to *forging position attack* in percentage
    - Broadcast of timely coordinated wrong traffic warning messages with forged positions
        - producing an illusion of a car accident, a traffic jam or an emergency braking
    - This degrades the performance of VANET in terms of channel utilization

In summary:

- Deterministic algorithms such as ROFF ensure a smaller end-to-end delay than Fast-Broadcast
    - Greater redundancy
    - Determinism and collisions
    - Higher number of Hello Messages
- SJ-Fast-Broadcast and SJ-ROFF improve coverage greatly
    - At the cost of more retransmissions

13 WPAN PROTOCOLS (WNMA11)

WPANs (Wireless Personal Area Networks) are areas where interconnected devices are centered around a person's workspace and connected wirelessly. Typically the range is within about $10\ m$ (very short).

General features:

- Ad-hoc networking with low data rate/consumption
- Meant to be easy to use and to setup
- They support multiple topologies and various frequency bands

Applications:

- Short range connectivity for multimedia applications
- Hands free devices connection
- Industrial sensor applications
- NFC/RFID/IoT

Here, two common technologies will be presented: Bluetooth and ZigBee.

13.1 BLUETOOTH

Bluetooth, also known as IEEE 802.15:

- was created in 1998 by the Bluetooth SIG (Special Internet Group)
    - formed by companies like Ericsson, IBM, Intel, Nokia, Toshiba
- was released in 1999 and promoters increased in number
- as of 2001, there were already more than 2400 companies using it
- today, almost all connected devices support it

The main goal of this protocol is to be a *cable substitution*, used for:

- *Synchronization*
    - Automatic synchronization of calendars, address books, business cards
    - Push button synchronization
    - Proximity operation
- Multiple device access
    - Cordless headsets
    - Portable PC speakers
    - Cordless printer, scanner, keyboard, mouse, LAN
- IoT
- Internet bridge
    - To connect devices to AP/directly to internet
- Direct file sharing among devices
- Ad-hoc networking
    - Difficult to implement
        - power consumption, connection management, CA/interference
        - interaction between multiple layers

13.1.1 Architecture

Bluetooth is a *layered* protocol architecture, able to cover most application cases and:

- Each use case needs different protocols involved and a specific profile
- Architecture is divided into:
    - *Core protocols* (because “they’re always there”, at the core infact)
        - Radio
            - Details of air interfaces
                - frequency hopping, modulation, tx power
        - Baseband
            - Connection establishment in Piconet
                - packet format, timing, addressing
        - Link Manager Protocol (LMP)
            - Setup between Bluetooth devices
            - Security aspects, authentication, encryption
        - Logical Link Control and Adaptation Protocol (L2CAP)
            - Adapts upper-layer protocols with Baseband
        - Service Discovery Protocol (SDP)
            - Device information, services
            - Establish a connection between two or more Bluetooth devices

    - *Mid protocols* (communication between ones and others)
        - *Cable replacement* protocols
            - RFCOMM (Radio Frequency Communication)
                - Emulates a virtual serial port in Bluetooth
        - *Telephony control* protocols
            - Telephony Control Specification – Binary (TCS BIN)
                - Set of commands/procedures for controlling telephony functions

    - *Adopted protocols* (use existing protocols and invent new ones only when necessary)
        - PPP
            - Point-to-Point protocol, IP datagrams over a point-to-point link
        - TCP/UDP/IP
        - OBEX
            - Object Exchange protocol, defines objects and operations
            - Working on multiple devices and even remote ones
        - WAE/WAP (Wireless Application Environment/Protocol)
            - Early frameworks to tailor PDAs and wireless/low-level devices

This structure can lead to:

- *high interoperability*
    - default solution for a usage model having a vertical slice through the protocol stack
    - basis for interoperability and logo requirements
    - each device supports one or more profiles
- *combination of different protocols*
    - different Bluetooth models
- *possibility to jump over layers* which are far from each other

This modularity allows only some parts of the Bluetooth to be used, based on the necessity. There are predefined configurations, called *profiles*, that are categorized by the following usage models:

- File transfer
- Internet bridge
    - Dial-up networking
- LAN access
- Synchronization
- Three-in-one phone
    - Cordless phone and intercom
- Headset

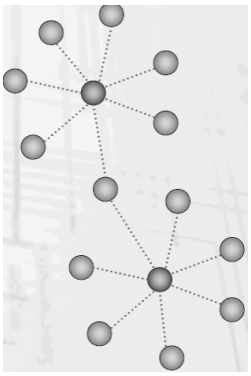
13.1.2 Topology

Consider for reference a series of parameters for Bluetooth Radio and Baseband:

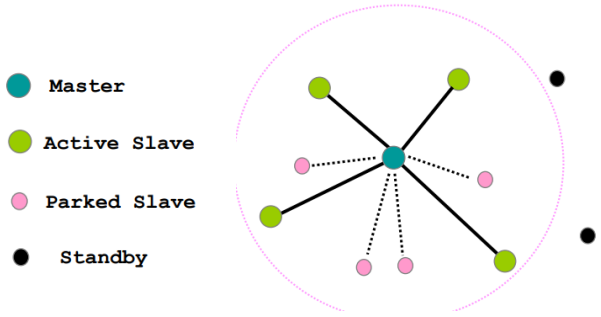
- Topology
    - Up to 7 simultaneous links in a logical star
- Peak/Max data rate
    - $1\ Mbps$
- RF (Radio Frequency)
    - Band: $2.4\ GHz$ up to $2.4835\ GHz$
- RF Channels
    - Channels can be changed from $n = 0, \dots, 78$
- Carrier Spacing
    - Frequency separation
    - $1\ MHz$
- Transmit power: $0.1\ mW$
- Access
    - *Piconet* (see ahead)
        - FH-TDD-TDMA
            - Different frequencies hopped in different times slots
    - *Scatternet* (see ahead)
        - FH-CDMA
            - Different frequencies with different unique code sequences
- Hop Rate
    - 1600 hops per second ($625\ \mu s$ duration per hop)

Let's describe the main 3 types of topology:

- Piconet
  - Basic unit of Bluetooth networking
  - A collection of devices connected in an ad-hoc fashion
  - *Master*
    - Controls communication within the piconet
      - Determines channel and phase
    - Sets the clock and hopping pattern
      - assigning addresses and scheduling packets
    - Each one can connect to 7 simultaneous active slaves or $200 + (255)$ inactive (parked) slaves

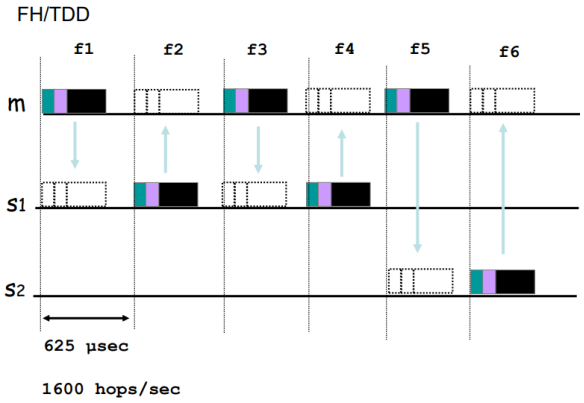


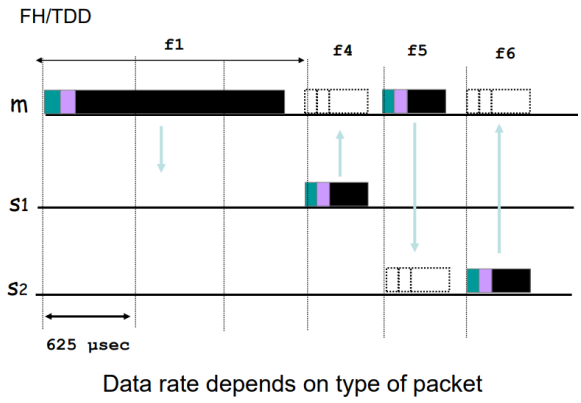
  - *Slave*
    - Operate under control of master
  - Each piconet has a unique hopping pattern/ID
    - Pseudo-randomly chosen
  - Connection setup
    - *Inquiry-scan* protocol → used to learn about
      - the clock offset
      - device address of other nodes in proximity
    - *Page-scan* protocol → to establish links with nodes in proximity
  - Formation
    - Page-scan protocol
      - Different nodes employed
        - Master
        - Active slave
        - Parked slave → temporarily inactive/low-power state
        - Standby



  - Communication (MAC protocol works with polling)
    - Polling Round-Robin (RR-Polling)
    - The master:
      - forwards other slave's messages
      - asks if it has something to send → never messages collision (MAC)
    - Time is divided into slots
    - Longer messages can occupy more slots → according to master's choice
    - On multi-slot packets → data rate depends on the type of packets

See figures below for more context:





- Point-to-point link
  - Master-slave relationship only
  - Nodes can act as masters or slaves

- Scatternet
  - Interconnected piconets supporting 9 or more devices
  - Devices in a scatternet *can act as both master and slave devices*
    - Device in one piconet may exist as master or slave in another piconet
  - One master per piconet
  - Few devices shared between piconets
    - Master/Slave (or viceversa)
    - Slave/Slave
    - With special features
  - Node can't be master of 2 Piconets connected simultaneously
    - no central network structure ("ad-hoc" network)
  - Allows many devices to share same areas
  - Currently very few implementations given limits of Bluetooth/MAC address protocol

Other features relative to all of Bluetooth topologies:

- *Addressing*
  - Bluetooth Device Address (BD_ADDR)
    - 48-bit IEEE MAC address
  - Active Member Address (AM_ADDR)
    - 3-bit active slave address
    - all zero-broadcast address
  - Parked Member Address (PM_ADDR)
    - 8-bit parked slave address

- *Error Correction schemes*
  - $\frac{1}{3}$ rate FEC (Forward Error Correction = recover lost packets sending extra parity packets)
    - Sender adds parts of data to get more redundancy/reliability according to needs
    - Used on 18-bit packet header, voice field in HV1 (HV = High-quality Voice) packets

- $\frac{2}{3}$ rate FEC
  - Used in DM (Data-Medium) packets, data fields of DV (Data Voice) packets, FHS (Frequency Hop Synchronization) packets and HV2 packets
- ARQ (Automatic Repeat reQuest)
  - Used with DM and DH (Data-High) packets
  - Has the following elements:
    - Error detection
      - destination detects errors → packets discarded
    - Positive acknowledgement
      - returned by destination
    - Retransmission after timeout
      - source retransmits if packet unacknowledged
    - Negative acknowledgement and retransmission
      - destination returns negative ack for packets with errors
      - source retransmits

- *Versions* (quick overview for evolution, not mandatory to know – expanded in some points by me)
  - Bluetooth 1.0
    - Basic wireless connectivity
  - Bluetooth 1.1
    - Error correction, improved security
  - Bluetooth 1.2
    - Faster connection setup, adaptive frequency hopping
  - Bluetooth 2.0
    - Up to $3\ Mbps$ and reduced latency, improved power efficiency
  - Bluetooth 2.1
    - More secure and simple device pairing
  - Bluetooth 3.0
    - Improved speed and cooperation with Wi-Fi
  - Bluetooth 4.0
    - LE: Low Energy (speed up to $1\ Mbps$)
    - UWB: Ultra-WideBand
  - Bluetooth 4.1
    - Improved interaction with 4G LTE
  - Bluetooth 4.2
    - Improved interaction with IoT
  - Bluetooth 5.0
    - Improved range, speed and interference avoidance
    - In actual ones, improved low-energy operation and location-based services

13.2 ZIGBEE

ZigBee (ZB) (or IEEE 802.15.4) and its focus is mainly about power usage (the name derives from the zigzag pattern of bees to collect nectar, because of its cooperative behavior). It was created to meet new market needs, considering:

- Over 100 times less energy required for connection operations
- No new wires
- Easy to install and maintain:
    - mesh networking, self-organizing
    - can also use different topologies, as star and cluster tree
- Reliable
    - mesh redundancy, multiple channels, interface tolerance
- Secure
    - AES 128
- Scalable
    - hundreds of thousands of devices – 65000 devices
- Low power consumption
    - can sleep most of time on, long battery life lasting even years
- Low cost
    - small footprint/lesser memory/lesser data rate
- High-level communications protocol based on 802.15.4 standard
    - used in sensor and controls
        - Home/industrial automation, remote metering, automotive networks
        - Interactive toys, medical
    - used small, ultra-low-power digital frequencies
        - Different depending on the region
- It targets some applications
    - Secure networking
    - Long battery life
        - Good for applications requiring a small amount of bandwidth
    - Low data rate
    - Lighting, sensor, RC peripherals
    - Encompasses text/graphics applications fairly well, both in LAN/PAN
- There are dedicated RF bands
    - Ranging from a few hundred MHz in Europe/America/Australia up to GHz for global use

As for the basic network characteristics:

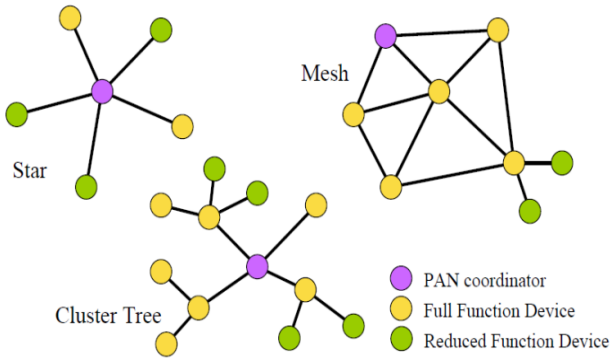
- It supports $2^{16} = 65536$ network (client) nodes
- It's optimized for timing-critical applications and power management
- Basic operations are optimized:
    - Time To Join Network: $< 30ms$
    - From Sleeping To Active: $< 15ms$
    - Channel Access Time: $< 15m$
- It has full mesh networking support (while Bluetooth/Wi-Fi do Pt-Mpt [Point-Multipoint])
    - Guarantees redundancy with multihop – so, if there is not a path, find other ones
- Topology can be either: mesh, star, cluster tree

Written by Gabriel R.

Also: remember AODV (here)? It's the protocol used in ZigBee.

ZigBee can support three different network topologies, as shown in figure:

- *Star*
  - All devices communicate directly with the central coordinator
- *Mesh*
  - All devices communicate with each other, forming a self-healing network
- *Cluster Tree*
  - Devices are organized in a hierarchical tree structure
  - They form clusters with central coordinators reducing power consumption



There are 3 types of nodes, linked via communications flow/virtual links:

- *ZigBee Coordinator (ZC)* – PAN coordinator
  - Each ZB network requires only and only one
  - It initiates network formation
  - It may act as router once it is formed and acts as PAN coordinator

ZigBee Coordinator (ZC)

- *ZigBee Router (ZR)* – full function node (can forward messages)
  - Optional network component
  - It acts as a coordinator
  - May associate with ZC or previously associated ZR
  - It participates in multihop routing of messages

ZigBee Router (ZR)

- *ZigBee End Device (ZED)* – reduced function node (cannot forward messages)
  - Optional network component
  - It shall not allow association or participate in routing
  - It just sends messages

ZigBee End Device (ZED)

ZigBee has several advantages:

- It's reliable and self-healing
  - Interoperable, low interference
- It supports large number of nodes
  - Data can be relayed with multiple paths, ensuring coverage and robustness
- It's easy to deploy
- It has very long battery life
- It's secure
- It's low-cost
- It can be used globally
  - Across different regions and markets and different ranges of frequencies

Cons are not quoted, but I will briefly mention some:

- it may be more costly and complex to implement
- may have shorter range
- may suffer from interference
- possible to have security problems

Inevitable question: ZigBee and Bluetooth are competitors or complements?

- Bluetooth is best suited for:
    - Synchronization of cell phones to PDAs
    - Hands-free audio
    - PDAs to printers
- ZigBee is best suited for:
    - Controls
    - Sensors
    - Lots of devices
    - Low duty cycle and long battery life
    - Small data packets

More specifically:

- Features and behavior
    - ZigBee is more suitable for low consumption and low data rate
    - Bluetooth for moderate to high data rate and synchronous communication
- Timing
    - ZigBee faster to perform networking activities + sleep-awake trigger
        - With ZigBee milliseconds are required, with Wi-Fi you would need seconds
- Power consumption
    - Bluetooth more expensive than ZigBee
        - Used with daily charging in mind
        - Designed to maximize ad-hoc functionality
    - ZigBee designed to be more economical and efficient
        - Optimized for slave power requirements

The Bluetooth consortium has been working a low power consumption version since 2001, called Wibree.

- Developed by Nokia in 2001, later adopted into Bluetooth as BLE (Bluetooth Low Energy)
- Performance similar to ZigBee
- Bluetooth version
    - without frequency hopping
    - with the possibility for nodes to be asleep most of the time
- It has been adopted into Bluetooth specifications
- It uses the same hardware as Bluetooth (shared antenna)
- Various applications
    - Health and fitness and wellbeing related gadgets and gizmos
    - Home automation systems
    - Automatic watches
    - Interactive toys
-

Written by Gabriel R.

- Has the same component cost of Bluetooth + 20 cents
- Targets a battery life of 1-2 years with data rate of $1\ Mbps$
- Ranges $10\ m$ approximately using circa $10\ mW$
- Has 128-bit encryption
- Time to wake and transmit is yet to be defined precisely

14 INDOOR LOCALIZATION (WNMA13)

Localization can provide a lot of useful services (e.g. best route, traffic jams, etc.). It can either be:

- *Outdoor*
  - Quite easy and eventually highly precise
  - This can be via:
    - GPS (Global Positioning System)
      - via satellites and triangulation of position
    - A-GPS (Assisted-GPS)
      - assisted, mobile cellular antennas help to give a rough position
      - works well even in challenging situations e.g. canyons/obstructed sky views

- *Indoor*
  - Much harder, given GPS signal is not working
    - there can be signal obstructions, multipath reflections, signal attenuation
  - In this, *location-aware applications* can be built
    - subset of context-aware applications
    - wide range of opportunities due to the popularity of smartphones
  - Even indoor applications have the need for indoor localization (which is still a challenge)

To assess a technology for indoor localization, we can use several *metrics*:

- *Accuracy*
  - average error between estimated measure and the actual one
  - how much difference between the measures
- *Precision*
  - error distribution regarding the actual position vs the estimated one
  - how measures are spread
- *Robustness*
  - ability in maintaining accurate estimation even when changing the context/environment
- *Scalability*
  - system behavior when changing the number and density of devices
- *Cost*
  - includes the hardware, the initial set up, the maintenance, etc.

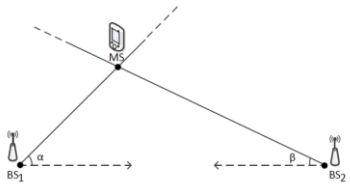
We will also give some *notation*:

- Environment
  - assumed to use a cartesian coordinate system in the monitored environment
- Mobile Station (MS)
  - the device that needs to be localized
  - e.g., a smartphone
- Base Station (BS)
  - an infrastructure component of the coordinate system
  - e.g., an Access Point (AP)

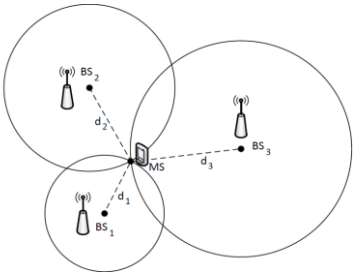
14.1 MAIN APPROACHES

Let's discuss the different approaches/methods to use:

- Triangulation (or "Angle of Arrival")
    - It requires knowledge of *arrival angles of* MS emitted signal and received one by the BS
        - at least two angles are needed to compute MS position (if they work precisely over angles)
        - requires complex hardware on the BS
        - indoor signal may be disturbed (e.g., walls)
        - not really usable indoor
            - because of strong multipath effects in indoor environments



- Trilateration (here not angles, but *distances*)
    - It requires knowledge of the *distance* between MS and BS
        - it works over distances and timestamping of clocks
            - taking them *synchronizedly*
        - requires the distance with
            - 3 BS for 2D localization
            - 4 BS for 3D localization



- Distance Estimation
    - It is the distance between MS and BS
        - which can be estimated through the propagation time of the radio signal
    - Since they are electromagnetic waves, the propagation speed is $c = 3 * 10^8\ m/s$
        - considering t_{prop} the measured propagation time
        - the distance is $d = c \times t_{prop}$
    - *Time Of Arrival (TOA)*
        - requires MS/BS to have a synchronized clock and the possibility to exchange data
            - BS emits the signal and sends to MS the time t_1 of transmission ended
            - MS completes the reception of the signal at time t_2
            - MS computes the propagation time as $t_{prop} = t_2 - t_1$
        - it may be hard to have synchronization
    - *Round Trip Time (RTT)*
        - it does not require data exchange or clock synchronization
        - it measures the time required for the path MS – BS – MS
        - $t_{prop} = \frac{t_{remote} - t_{local}}{2}$
            - t_{local} = depends on hardware reaction time
            - t_{remote} = from data to ACK
        - t_{local} is variable as it depends on the reaction time of the hardware
            - this error cannot be avoided

We consider the measurement error generated by the granularity of the timer used:

- Most boards allow to save hardware timestamp of MAC layer packets with a precision of $1\ \mu s$
    - Corresponding to a granularity of $300\ m$ when considering the speed of light

This precision is not sufficient and two possible solutions are proposed:

- *Hardware approach*
    - Use *timestamps provided by enhanced/modified HW*
    - Specific hardware using bits transmitted/received to trigger a *MAC layer counter*
        - based on the clock of the WLAN board
        - $\sim 7\ m$ of precision
        - improved precision via statistical methods and multiple measurements
    - Measurements done at MAC layer (lowest possible layer)
        - to avoid variable delays in execution time induced by upper layers
    - RTT is measured with Data-Ack pair packets
        - From reception of data to SIFS

- *Software approach*
    - Multiple measurements to obtain a close estimation to the actual value
    - Hardware timestamps are provided by regular WLAN boards
        - only for received packets (not for sent ones)
    - Need to introduce a *monitoring station*
        - as close as possible to the MS that we want to localize
        - has to monitor communications between MS and BS
            - to obtain consistent hardware time-stamps
            - both for sent and received packets (by the MS)
        - this requirements makes approach unfeasible for practical purposes
    - On this, we introduce *goodtry*
        - prototype developed for research purposes
        - it measures sequences of packets RTS-CTS-Data-Ack
            - hence measuring twice the RTT for each transmission
        - positioning is done through the lowest weighted squared errors
            - which includes a method to manage multiple measurements

14.2 Other Approaches

Different approaches are:

- TDOA – Time Difference of Arrival
    - It measures the arrival time of a signal emitted by the MS towards multiple BSs
        - exploits the difference among arrival times to extract MS position
    - It requires synchronization among BSs
        - 3 BS for 2D localization
        - 4 BS for 3D localization
    - Server needed to manage both the synchronization/measurement collection from BSs
    - Not usable for self-positioning: measurements are possible only at the BS

- Scene Analysis
    - Method composed by two main phases:
        - *Fingerprints* collection (fingerprint = signal strength and other parameters)
            - Storage in a DB of fingerprints of the scene in predetermined locations
                - make samples of single signals
        - Actual evaluation
            - Collection of the fingerprints related to the current position (not known)
            - Comparison with DB data through AI algorithms/statistical methods
                - e.g., k-NN, SVM, etc.
    - Not strong/robust enough
        - Methods require big initial effort/offline training to create fingerprint DB
        - Still requires synchronization among BSs
            - 3 BS for 2D localization
            - 4 BS for 3D localization
    - RSS (Received Signal Strength) depends upon:
        - *Multipath*
            - due to reflected signals, measured strength is higher than ideal
        - *Shadowing*
            - Signal strength in case of NLOS (Non-Line-Of-Sight) is not easily computable
                - as the transmitted frequence can be absorbed by environment
        - *Moving objects*
            - Cause sudden oscillations of the RSS
                - thus requiring multiple measurements

- k-Nearest Neighbor (k-NN)
    - has some parameters to consider first:
        - m
            - the number of BSs
        - n
            - the number of fingerprints in the training set
        - S_i
            - the fingerprint (array of cardinality m)
            - corresponding to the point (x_i, y_i, z_i) of the training set
        - s
            - the measurement or the RSS as performed by the MS in the online phase
    - computes the distance s from every fingerprint in the training set
    - the chosen k points in the training set are those with the smallest d_i^2 values
    - MS coordinates are then estimated as
        - the arithmetic mean of the coordinates of k locations or
        - the weighted mean of the distances

- RFID (Radio Frequency Identification)
    - Composed of:
        - RFID reader
            - Used as BS, emitting signals to query tags in proximity
            - They receive the ID of each tag as a response
            - They communicate with localization servers
            - Pretty expensive component
        - RFID tag
            - A device that answers to the query of the reader by transmitting its own ID
                - *passive tag*: low cost, shorter range, longer duration (no battery)
                - *active tag*: higher cost, longer range, shorter duration (battery)

Consider a positioning system that exploits *Active RFID* called *LANDMARK:*

- It uses a scene analysis method based on RSSI
    - Received Signal Strength Indicator
- It is composed of:
    - RFID readers
        - used as BS
        - capability to communicate performed measures to a localization server
    - Reference tag
        - tag with known coordinates
    - Tracking tag
        - tag to be localized (MS)
    - Fingerprinter is the array of RSSI of the signal emitted by RFID tags and received by the BS
    - It is robust since:
        - reference tags do not require the offline phase to train the system
        - reference tag positions can be dynamically measured when scene changes
    - For the localization the k-NN algorithm is used:
        - exploiting the reference tag fingerprints as training set
        - the system is significantly affected by the employed hardware
            - not all RFID readers provide a sufficiently fine granularity of the RSS
        - active RFID tags are powered by a battery
            - system requires the transmission power of all tags to be very similar
        - need to use RFID tags of the same type and with the same level of battery
            - to obtain comparable measurements that can be used

Now consider the *Passive RFID* example:

- It uses RFID reader as MS and is less expensive (than the previous solution) when:
    - we have a wide space
    - we need to localize few nodes
- *Pros*
    - The use of passive RFID tags as reference ones reduces the cost
    - It's good for automated environments: predictable and low mobility
- *Cons*
    - Based on fingerprints and the training phase is expensive
    - A snapshot is obtained from different measurements of the same location
        - multiple reads are needed
    - Not very robust against variations of the environment

Written by Gabriel R.

14.3 AUGMENTED REALITY (AR)

Let's discuss about other application realms: Visual/AR Solutions, specifically Augmented Reality.

- It is based on the superimposition of informative levels to the real world
    - virtual, multimedia, geolocalized elements, etc.
- Uses application exploiting GPS, compass and accelerometer of mobile devices
- It extracts positions via *visual markers*
    - artificial markers used mainly in AR
    - to superimpose virtual contents to a real image
        - need to precisely know the actual position of the markers
        - in the reference model of the camera
- There needs to be done *coordinates translation*
    - the position of an object/of a reference system in a 3D cartesian space
        - can be described through a matrix
    - there is
        - a rotation matrix describing the orientation of the object
        - a translation array
        - a matrix product used to extract the new point coordinates
- *ARToolKit* is a reference library in this context
    - Uses
        - a matrix P_m
            - representing the position of the marker in the camera reference system
        - a matrix P_c
            - describes the position of the camera in the marker reference system
            - can be obtained as the inverse of the previous ($P_c = P_m^{-1}$)
            - from this one, we can extract the translation array T_c
        - the marker global position G
            - derived by T_c
            - used to translate the coordinates
            - from the marker reference system to the global one
- Reference systems are the following ones:
    - with a tag
        - it's possible to superimpose the object
        - its position is determined respect of what you're looking at
    - with a camera
        - try to determine camera position respect of tag by using a picture
    - your position
        - uses tag position
        - if you find a tag, you can be localized
- Coverage of the system
    - Improving image resolution is useful to obtain a good tradeoff
        - between marker size and system coverage
    - With 5MP images a marker of $20\ x\ 20\ cm$ can be recognized at a distance of $11\ m$

- Many *error sources*:
    - Smartphones do not generally allow to access the raw version of the captured image
        - only allow access to compressed images (JPEG)
    - JPEG compression is *lossy* (introduces noise in the input)
        - Minimum level of compression = an acceptable level of precision
    - A calibration phase is needed to compensate optic errors (distorted image)
    - The error on the position estimation is *proportional to the distance*
        - due to the error in the orientation computation

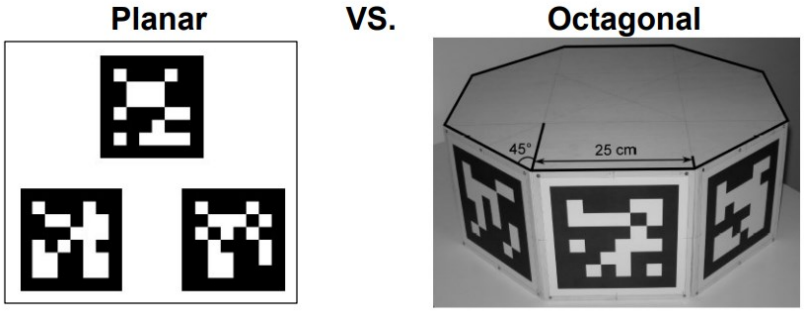
Some observations:

- Experiments with single marker/tag results in too many errors
- Less errors if markers appear not frontally w.r.t. the camera

Multimarkers are a set of single markers arranged in a known way.

- They can be:
    - planar
    - octagonal
        - made of markers on different planes
        - does not allow for the direct use of RPP algorithm
            - Robust Planar Pose = estimates the pose of a camera relative to an object
        - developed algorithm for octagonal multimarker
            - use RPP to estimate camera position with respect to the single markers
            - assign a weight to each estimation, privileging non-frontal markers
            - compute camera position as weighted average of the estimations

See below for confrontation visually:



Experimentally, planar and octagonal are confronted:

- Multimarker (single markers of 20 cm each)
    - Set of estimation as a sequence of pictures
    - Compute camera estimated position as average of estimations left
- Many pictures rapidly taken for each measurement location
- Planar is simpler than octagonal and not worse

On the results, we might say:

- Two multimarkers are enough
    - no need for having one at each corner
- Compared strategies
    - Consider only the *closest* multimarker
    - Consider the *average* of the two estimations
- Average error
    - Closest: from 11 to 22 cm
    - Average: from 15 to 21 cm

And in conclusion:

- Advantages with respect to radiowaves systems
    - Higher precision
        - 22 cm vs 50 cm (for active RFID) vs 90 cm (for 802.11 trilateration)
    - Lower cost for the hardware
    - Lower installation and configuration time
        - with respect to scene analysis methods
- Disadvantages
    - Line of sight should be free between the smartphone and at least one marker
    - Semi-automatic system
        - may require user intervention and limits its autonomy

15 MOLECULAR COMMUNICATION (WNMA12)

Molecular communication is performed through *nano machines (NMs)*:

- nano-scale devices able to perform specific tasks at nano-level
- very simple and restricted
    - tasks such as communicating, computing or data storing
- devices can be both artificial and natural
- they are very similar to cells
    - working at a sensor network level on a connected-architectural basis
        - CPU corresponds to nucleus
        - power to the mitochondrion
        - receptors are sensors

The following are the three three types of development approaches for them:

- *Top-down* approach
    - Nano-scale development
    - Done by downscaling current existing micro-scale level components
- *Bottom-up* approach
    - Nano-machines developed using individual molecules
    - Creating technologies that do not exist yet
- *Bio-hybrid* approach
    - Biological nano-machines as models or building blocks to develop new nano-machines
    - They grow developing themselves

15.1 NANO-NETWORKS AND TYPES

We define a nano-network as:

- electronic components and their interconnection within a single chip on a nano-scale
    - a network of nano-machines or devices at a nano level
- a set of components on a nano-scale and their interconnection
    - probably more biological than electronic

We can identify three macro-categories of communication (subsequently described in detail):

- *Standard Communication*
- *Nano-mechanical Communication*
- *Molecular Communication*

Starting from Standard Communication, we have two waves of communicating:

- *Electromagnetic waves*
    - Given the size of nano-machines, wiring a large quantity of them is unfeasible
    - Wireless solutions could be used but antennas are hard to be integrated
    - Energy required to power the antenna is too high
- *Acoustic waves*
    - Implies transducers in nano-machines capable to sense the waves
    - Size of these transducers is very high → high barrier

With Nano-mechanical Communication:

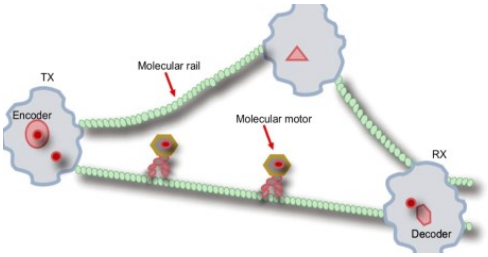
- *Pros*
  - information is transmitted through hard junctions between linked devices at nano-level
- *Cons*
  - it requires physical contact between transmitter and receiver

Then, Molecular Communication is the more feasible solution for the knowledge humanity has today.

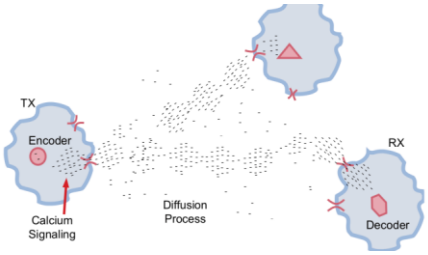
- It's defined as the transmission and reception of information encoded in molecules
  - it's most promising approach for nano-networking
- Molecular transceivers are already present and conceived at nano-scale
- Transmitters and receivers can be remotely located
  - as far as the transmitted molecules can go

There are two types of transmission approaches (right figures represent corresponding left topic):

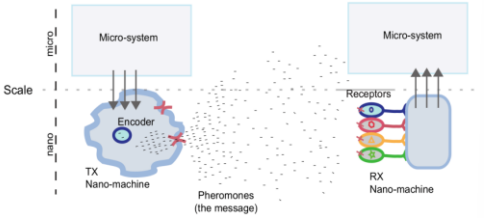
- *Short range* communications (within *mm*)
  - *Molecular motors*
    - Proteins that transform chemical energy into mechanical work
    - They travel along molecular rails called micro-tubules, previously deployed setting a complete railway (rails may be full-duplex)
    - Data needs to be encoded, transmitted and decoded



  - *Calcium signalling*
    - One of the most well-known molecular communication technique in biology
    - Responsible for many coordinated cellular tasks
      - fertilization, contraction and secretion
    - More flexible
      - there is no need of railways
    - All surrounding nano-machines can receive a broadcast message



- *Long range* communications (from *mm* up to *km*)
  - *Pheromones*
    - Molecular compounds containing information that can only be decoded by specific receivers
    - Messages consist of molecules
      - Huge quantity of possible combinations
      - In ants colony, communication between members is based on pheromones



- *Bacteria*
  - They have a number of interesting characteristics
    - *Conjugation*
      - Allows bacteria to interconnect and pass copies of plasmids
        - genetic messages encoded as strings of four nucleotides
    - *Chemotaxis*
      - This is the movement induced into bacteria by chemical stimuli
        - Can swim towards food particles, like glucose
        - Can swim away from poisons, like salts
      - Bacteria communicate through emission of chemo-components
        - Colony survival rates increase with cooperation
    - *Resistance to antibiotics*
      - Can be used to filter and kill off bacteria that do not contain legitimate and/or complete plasmids
      - Plasmids only gone through partial conjugation are discarded

We have discussed plenty of *opportunistic networks*:

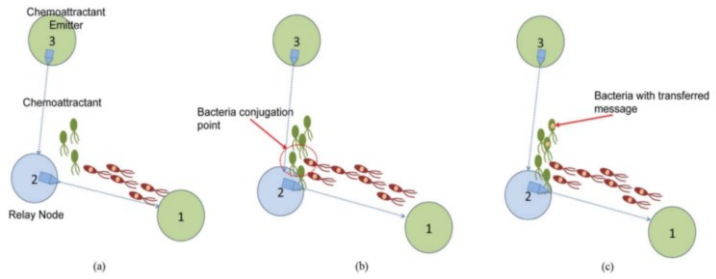
- contacts are intermittent
- link performance is extremely variable.

This can be applicable to bacteria-based nanonetworks as well:

- Messages not intended for the recipients are simply dropped
- Chemoattractants are spread to achieve the wanted path
  - basically, small molecules acting as binders on leukocytes
  - leukocytes protected the immune system against diseases

Conjugation and chemotaxis are jointly used in this context of opportunistic routing, as you can see here:

- Chemoattractant from node 2 attracts bacteria from node 1
- Chemoattractant from node 3 attracts bacteria from node 2



Here there are some possible applications for this:

- Novel healthcare and medical technologies
  - Targeted Drug Delivery
  - Live health monitoring
- Novel environment technologies
  - Oil spilling containment
  - Water resources monitoring
- Counter bioterrorism applications
  - Referred to biological agents

Being highly critical applications, the implementations should be highly reliable and secure, interesting different research fields such as computer science, engineering, biology. Open design issues are of paramount importance and definitely researches on the topic should be stimulated.

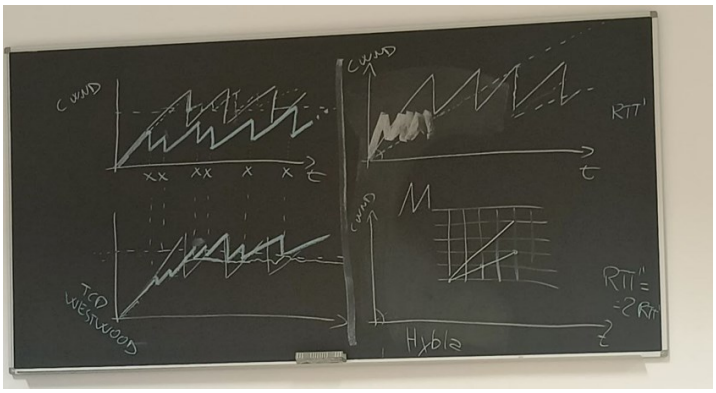
Written by Gabriel R.

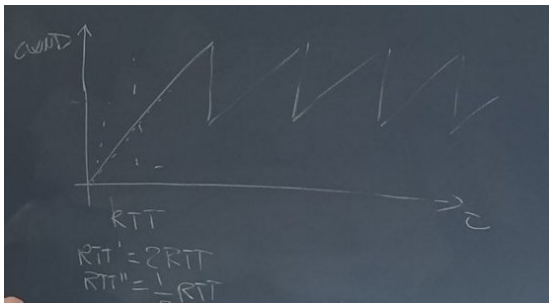
16 Possible Exam Questions From Class and Old Exam Questions

Here is a textual list of popular questions, as Palazzi told us in class last lesson:

- Exposed/Hidden Terminal and how to solve
- Piconet/Scatternet
- Vehicular Networks Transmission Ramge
- Differences between TCP versions and determine range specific to the single case
- What is the Fresnel Zone
- Differences between AODV and DSR

Here are some pictures of things that may be asked:





A comprehensive list including all the ones from MEGA (as of February 2024):

- 2023-2024
  - RTT Unfairness
  - TCP Hybla
  - Hidden and Exposed terminal problems
  - DIFS and SIFS time
  - Show how the $cwnd$ changes over time with TCP + explanation of what was represented
  - What is an ad-hoc network? Draw one and show how the routing between 2 nodes work
  - Talk about Bluetooth + what is a piconet/scatternet?
  - Talk about VANET + which protocol we saw in class?
  - Indicate one TCP protocol shown in class that fixes one of his problems
  - Talk about TCP + wireless and draw RTT + cwnd and how changes (e.g. Hybla)
  - Draw a scatternet and talk about it
  - Talk about MANETs and talk about its routing algorithms/AODV with drawing and also difference between AODV and DSR

- 2022-2023
  - 1 – IEEE 802.11: What is it? What does it regulate? What are the different versions of the standard? Do they need to be able to interact?
  - 2 – DCF: What is it? How does a packet exchange work? What is the difference by using CSMA/CA? What is the "hidden terminal" problem and how to resolve it?
  - 3 – TCP: What is it? What are the main versions?
  - 4 – TCP HYBLA: What is it? What is it suitable for? How does it work? What are to pros and cons of using it?
  - 5 – Bluetooth: What is it? What is a piconet and is it formed?

- 2021-2022
  - TCP wireless
  - Routing inside MANETs in general
    - May seem more cryptic than it actually is
    - Infact, it's required to talk about Routing chapter in general
  - AODV

- 2018-2019
  - Fresnel Zone: general explanation with drawing
  - TCP Hybla:
    - how it works (description of rho parameter)
    - goal (elimination of RTT parameter, place all connections at RTT_0)
    - graph of how a TCP Hybla connection changes along with another generic TCP
      - (e.g., TCP New Reno)
  - 802.11 standards:
    - graph of the steps of a new connection (DIFS, RTS, SIFS, CTS, SIFS, message...)
  - Hidden terminal:
    - explanation
    - because it is not enough $< message, ack >$
      - but you have to do $< RTS, CTS, message, ack >$
      - (message can be lost if you lose RTS it weighs less)
  - Vehicular Networks:
    - general explanation
    - parameters defined by the standard

- 2013-2014
  - 3 TCP charts (New Reno)
    - one made by him
    - one with twice as many errors
    - one with twice as many RTTs
  - Mention a version of TCP that works with high [packet] errors in wireless
    - Answer (I would say): SACK (answers correctly Reno/New Reno problems and recovers in a single RTT, working overall pretty well)
    - Other answer: Snoop (addresses high BERs)
  - Mention a version of TCP that works on lines with high RTTs and is fair
    - Answer: Hybla (but fairness is not always there)

- Answer: Cubic (better on unfairness, works with high RTTs)

- Quickly explain why we chose to mention those versions of TCP
- What is the contention window of MAC 802.11 (not the congestion window of TCP)
  - What it is used for, how it works, how it is managed
- What is an ad hoc network
  - Mention some typical problems
    - especially why routing can be difficult
- Explain the difference between proactive and reactive routing
  - and mention examples, name only
- What are VANETs, what is special about nodes and needs