Sharing Channels
CS 118
Computer Network Fundamentals
Peter Reiher
Outline

• Ways to share the channel
• Label (name) implications
• Emulated sharing
• Explicit coordination
Ways to share a channel

• Different channels
• Different times
• Different symbol sets ("languages")
• Label the transmissions
Different channels

- Spatial Division Multiplexing (SDM)
  - Channels are spatially distinct

  - The most costly
    (basically where we started – doesn’t scale)
Different times (TDMA)

• Take turns using the channel
  – Whole channel
  – Split in time
TDMA concerns

• Time interval size
  – Long enough for at least one symbol

• Time interval allocation
  – Fairness
  – Starvation
  – Efficiency (unused slots)

• Gap between intervals
  – “Guard time”
Impact of guard time

• Guard time avoids sender overlap
  – All receivers should see non-overlapping slots
  – But:
    • Sender clocks drift (gain or lose offset)
    • Symbol delay varies

• Consequence
  – TDMA needs clock coordination
  – TDMA has distance limit
    • Long distance = large guard gaps = inefficient channel
Different symbol sets

• Each symbol set is an independent “language”
  – Independence means that the bit encoding is separate from the way the sets are distinct
• Many different variants:
  – Different representations using independent physical properties
  – Different alphabets (logical representations)
Different physical representations

• Frequency

• Polarization

• Orbital angular momentum

• Combinations of the above
Frequency (FDMA)

- Split channel capacity into non-overlapping ranges
  - Works for wavelengths
  - E.g., for EM (light, RF)
FDMA concerns

• Width of the band (bandwidth)
  – Large enough for desired bitrate

• Band allocation
  – Fairness
    • Starvation
    • Efficiency (unused frequencies)
  – Gap between bands
  – “Guard bands”
Impact of guard bands

• Guard bands avoids sender overlap
  – All receivers should see non-overlapping bands
  – But:
    • Sender frequencies drift
    • Motion affects frequency (Doppler shift)

• Consequence
  – FDMA needs frequency coordination
  – FDMA has band size limit
Different alphabets

• A different way to group by physical property
  – Instead of using independent properties, separate groups by the values of one or more properties
  – Need the groups to be independently usable
Code (CDMA)

- Combines frequency and time
  - A combination of time and frequency that allows partial overlap that can enable communication in the presence of increased noise
xDM vs xDMA

• {Space, Time, Code} Division Multiplexing
  – Sharing (dividing a resource) by multiplexing (merging) or demultiplexing (splitting) based on spatial, temporal, or coding context

• {S/T/C) Division Multiple-Access
  – Using xDM to coordinate shared access of a channel by multiple sources or receivers

*Often used somewhat interchangeably*
Sharing compared

TDMA

FDMA

CDMA
Label (name) implications

• If you only worry about multiparty:
  – Unique per-node peer names
  – Unique per-node internal state/TM subset names

• If you also worry about channel sharing
  – Name sets used in overlapping contexts
  – Need to ensure no namespace collisions
  – Need to ensure both ends agree on names
Destination names

• Context (1:N)
  – Know the channel
  – MAY mean the receiver knows the source

• Uniqueness
  – MUST be unique per-receiver on this channel

• Shared
  – MUST be known by sender and receiver
  – Sender knows what to attach to a message
  – Receiver knows where message goes to
Source names

• Context (N:1)
  – Know the channel
  – MAY mean the source knows the receiver

• Uniqueness
  – MUST be unique per-sender on this channel

• Shared
  – MUST be known by sender and receiver
  – Sender attaches its name to message
  – Receiver knows where the message came from
Combined names

• Context (N:N)
  – Maybe know the subset of senders/receivers
  – Not very useful

• Uniqueness
  – Send names MUST be unique
  – Receive names MUST be unique
  – MAY (usually) correlate send:receive names

• Shared
  – MUST be known by all senders and receivers
  – Senders attach BOTH names (its send, dest’s recv)
  – Receiver uses BOTH names (determine source, decide to accept)
Name assignment

• A-priori here (for now)
  – Part of the protocol configuration

• How?
  – An organization (IANA, IEEE, etc.)
  – To ensure uniqueness
  – How expensive?
Emulated sharing

- Devices can emulate sharing
Demultiplexer

- 1:N
  - One source, multiple receivers
- Isolates receiver from sharing
  - Source still thinks the channel is shared
    - Needs to indicate the destination
  - Receiver thinks it has direct channels
    - Doesn’t need to know whether to listen
- What’s in the box?
  - Copies / splits symbols
  - Use destination names to demultiplex
    (pick output port)
  - Can remove the differences (translation)
    i.e., using a FSM
Multiplexer

• **N:1**
  – Multiple sources, one receivers

• **Isolates sender from sharing**
  – Source still thinks it has direct channels
    • Doesn’t need to indicate the source name
  – Receiver thinks the channel is shared
    • Needs to know the source

• **What’s in the box?**
  – Merges / interleaves symbols
  – Add source names to output
  – Adds the differences (translation)
    i.e., using a FSM
Switch

• N:N
  – Multiple sources, multiple recvs
  – Combines demux with mux

• Isolates sender and a receiver in different ways
  – Sender still needs to indicate receiver (like demux)
  – Receiver still needs to know sender (like mux)

• Centralizes coordination
  – Internal to the switch
Switch pros and cons

• Pros
  – Coordination is internal
  – Easier to install/manage channel wiring/fibers
  – All the pros of explicit coordination
    • Efficient, global balance, simple to implement

• Cons
  – All the cons of explicit coordination
    • Load, fault tolerance, trust
  – Still needs source/dest to participate in naming
  – Still needs unique names
What about circuits vs. packets?

- Really just a continuum of TDMA
- Smaller allocation avoids impact to others
- In this case, doesn’t matter much!
  - It will matter more in later lectures
Explicit resource coordination

• Why coordinate?
  – N:1 sharing needs to avoid collisions

• Where is 1:N sharing coordinated?
  – Can be just inside the OS in the endpoint

• Why explicit?
  – Simple case, focus of this class
A-priori coordination

- Part of the pre-shared rules
  - I.e., part of the protocol

- Fixed allocation
  - Fixed schedule, frequency bands, etc.
Limits of a-priori coordination

• Requires coordination
  – Need to build it into the protocol
  – Still needs a starting point (time, frequency)

• Inefficient
  – Slow/costly to change (repeated coordination)
  – Fixed allocations can’t adapt to dynamic uses
  – Can’t easily add/remove nodes or resources
Centralized coordination

- Manager node controls each shared channel
  - They decide when each source can transmit
  - Can ask the sources about needs
Requirements for central coordination

• Symmetric channel
  – All possible sources need to be able to hear the manager
  – All possible sources need to respond to the manager
  – Also receive-only nodes
Central coordination protocol

• Polling
  – Non-PC terms:
    • “Master”
    • “Slave”
  – Channel
    • Bus

Master does the following
  – Ask each source in turn – anything to send?
  – Then schedules and gives each source a slot to send
Limits of central coordination

• Load
  – Pushes all the work to one manager

• Fault tolerance
  – Manager could fail
  – Channel to manager can fail

• Trust
  – Manager has all the power!
Hierarchical/delegated coordination

• Extend central coordination
  – Single manager can split a shared resource and assign each to another to manage

• Pros
  – Relieves load on a single manager

• Cons
  – Less flexible; hard to coordinate sharing across delegated fractions
Benefits of explicit coordination

• Potentially very efficient
  – Esp. if requirements are stable (don’t vary)

• Potential for global balance
  – Manager knows all, so can make the most informed decisions
  – Trivial to avoid starvation, ensure fairness

• Simple to implement
  – Simple coordination protocol
Limits of explicit coordination

• Potentially very inefficient
  – Inflexible, slow to react
  – Can require lots of messages to parties not involved in the communication

• Vulnerable
  – Faults, non-malicious, and malicious errors
  – All can completely halt shared channel use

• Can be costly to implement
  – Focused management can require centralized resources (CPU, memory, etc.)
Use cases for explicit coordination

• When all communication already flows through one party
• When does that happen?
  – Satellite
  – Airplane/blimp
  – Ceiling (infrared)
  – Ethernet switch
  – WIFI switch
Decentralized Sharing

- Extending 2-party and N-party masters
- Sharing without a master
- Limitations of no-master sharing
- Naming implications
- More switching
Overall sharing goals

• Fairness
  – Allocation is proportional to needs
• Starvation-Free
  – All members receive non-zero allocations
• Efficient
  – Minimize resources not usefully allocated

As with any resource allocation
2-party master

• Recall:
  – One side controls the system
  – Master: sends as desired, polls other side

• Issues
  – Controller (master) selection
  – Fault tolerance
  – Bias
2-party controller selection

• Seek inspiration
  – O.K. Corral: whoever shoots first wins
  – Backgammon: roll dice; highest roll goes first

• Tie-breaking
  – O.K. Corral: not needed (both dead!)
  – Backgammon: try again!
Tie-breaking 101

• Problem
  – Computers are deterministic
  – Rolls are pseudorandom sequences
  – Algorithm and seed generates one sequence

• Solution
  – Highest serial number
  – Requires a serial number that can never tie
2-party fault tolerance

• What happens if controller halts?
  – No problem!
  – No communication anyway!

• “Fate sharing”
  – The controller and 2-party system share fate
  – No case where communication could happen but a dead controller prevents it

• More complex issue when we get beyond 2 parties . . .
2-party bias

• Controller
  – Can send whenever desired

• Other side
  – Needs to wait for controller to poll

• Impact:
  – Biased controller can undermine fairness
  – Even a “good” controller has problems
Why are there problems?

- Client request might occur just after every poll

  ![Diagram showing Master and Client with messages Got anything? and Nope]

  - When a poll returns NO, client must wait for next poll
  - Whereas the server can send immediately
Solving 2-party control

• Transfer control of a master
  – Helps balance bias over the long term
  – Additional cost to initiate/confirm the transfer

• Shift from master to ping-pong
  – One side starts
  – Send message or shift the token
  – Token “ping-pongs” until useful data is sent
  – Both sides get an equal chance to send
  – Fair if message lengths are equal (on average)
    (can establish length upper bound)
N-party master

• Like 2-party in general
  – Controller (master) polls each member

• Same issues
  – Controller (master) selection
  – Fault tolerance
  – Bias
N-party controller selection

• Same solutions
  – Go-first (time)
  – Highest-roll (value)

• Same tie-breaking
  – Try again

• Doesn’t scale very well
  – Many selection algorithms prone to ties at high scale
  – The Birthday Problem
Happy Birthday!

- What’s the probability that one of you shares Ben Franklin’s birthday (Jan. 17)?
  - For 40 people, \(1 - (364/365)^{40} = 10\%\)
- What’s the probability that two of you share one birthday?
  - Roughly 90% for 40 people

- How does this apply to controller selection?

- Unless random space is much larger than the number of candidates, ties are likely
N-party fault tolerance

• What happens if controller halts?
  – “A failure to communicate”

• No more “fate sharing”
  – A controller can halt while other pairs could still want to communicate
N-party bias

• Controller has much more “control”
  – Can treat clients preferentially
  – Can keep all clients waiting

• New issues
  – Not just controller/client message sizes, but also the sizes of each client’s messages
Solving N-party control

• Shift from master to rotation
  – Rotation is N-party version of ping-pong cycle
  – Aboriginal “Talking stick”

• Rules:
  – Starts with the chief
    • Need a “chief election” protocol (dice?)
  – Pass in a circle to the right
  – Only the stick holder can talk

• This is “Token bus” (IEEE 802.4)
  – Used by GM for automation
  – Derived from a ring network
    (but we haven’t even gotten there yet)
Problems with token bus

- Token generation
  - Protocol to select the token holder
- Token regeneration
  - What if the token holder fails?
- Enforcing single-token rule
  - Members can cheat
- Membership changes
  - Add member – repair sequence
  - Remove member – repair sequence, regenerate token

Result – largely abandoned
Sharing without a master

- Inspiration:
  - Discussion group without a talking stick
  - “Party line” telephone
Aloha!

- Radio network (1971)
  - One shared channel

1. Message to send
2. Send message
3. Did you hear it?
   - Yes – DONE
   - No – resend (goto #2)
Why didn’t you hear your message?

- Because someone else stepped on it
  - By transmitting at the same time
- What do you do about it?
  - Send again
  - Hoping it won’t get stepped on again
- A little problem
  - The other guy’s message also got stepped on
    - By you
  - He’s going to send again, too
Using random delays

• If your message is stepped on, don’t send right away
• Wait for a random time and try again
• You hope the other guy waits longer
  — Or sufficiently shorter
• In which case you don’t step on each other again
• Obvious issue of utilization vs. chances of repeated collisions
One solution

- Slotted Aloha
- Don’t send just any time
- Divide time into slots
- Only send at the start of a slot
- On collision, retransmit in next slot
  - With probability $p$ (<1)
Pure vs. Slotted

• Pure
  – Send whenever

• Slotted
  – Common slot time
  – Send at slot start only
  – Mixes in TDMA
Pure vs. slotted

• Assuming fixed-size messages

• Assuming Poisson arrivals
Do you hear what I hear?

• Maybe we can do better, if we just listen first
Discussion group rules

1. Message to send
2. *Listen for quiet*
3. Send message
4. Did you hear it?
   - Yes – DONE
   - No – resend (goto #3)

• But there are some issues . . .
Summary

• Multiple parties can share channels in various ways
  – TDMA, FDMA, CDMA
• Sharing suggests coordination
  – Built into protocol
  – Via a master (static or changing)
• Like most things, more complex at high scale
• If everyone can hear results, can sometimes share without any master