

Sharing Channels

CS 118

Computer Network Fundamentals

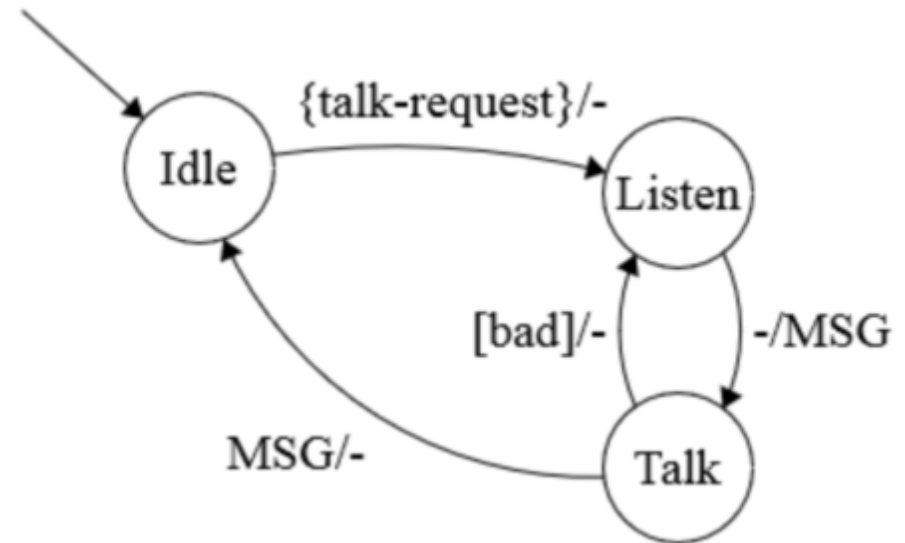
Peter Reiher

Outline

- Carrier sense channel sharing
- Naming
- ?
- ?

Sending without a master

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



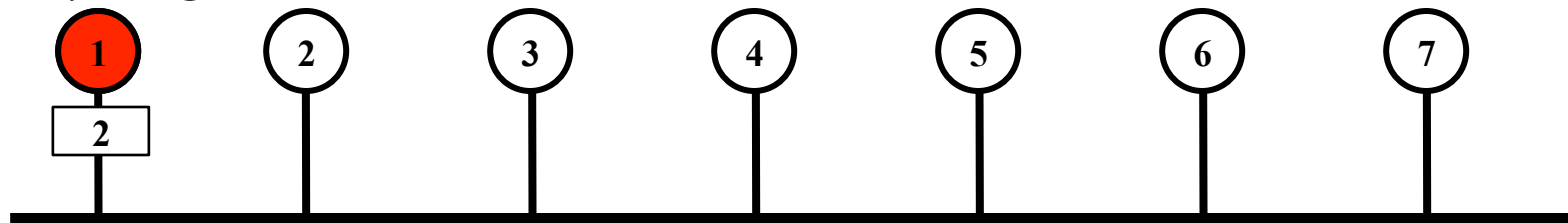
CSMA (IEEE 802.3)

- An implementation of this idea
- Carrier-sense multiple access (1974)
 - Carrier = channel idle sense whether channel is idle
 - Listen before talking listen before you talk



CSMA behavior

*I don't hear
anything!*

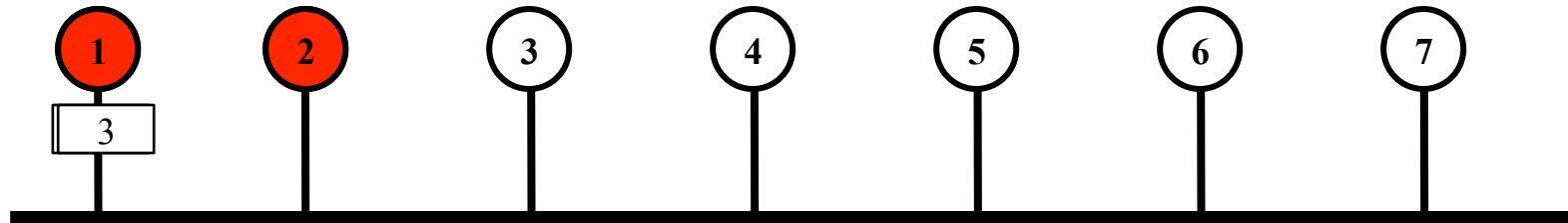


CSMA and persistence

- OK, so I listen to the channel
- What if it's busy?
- Well, I certainly don't send now
 - My message would interfere with what I hear
- But do I keep listening?
- Persistent CSMA listens till channel is free
- Non-persistent CSMA stops listening and checks again later
 - After some random interval

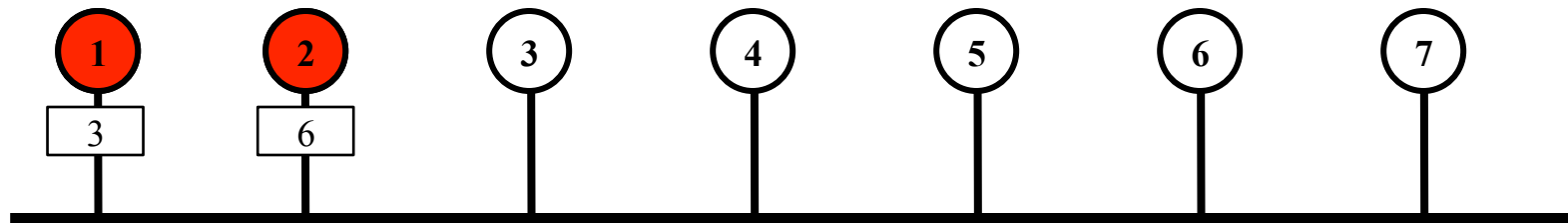
CSMA behavior

*I don't hear I hear a
anything! message!*



- Node 1 wants to send a message to node 3
- Node 1 listens to the medium
- It happens again
- But in the middle, node 2 wants to send a message to node 6
- Node 2 listens to the medium
- Node 2 doesn't send

What about node 2's message?

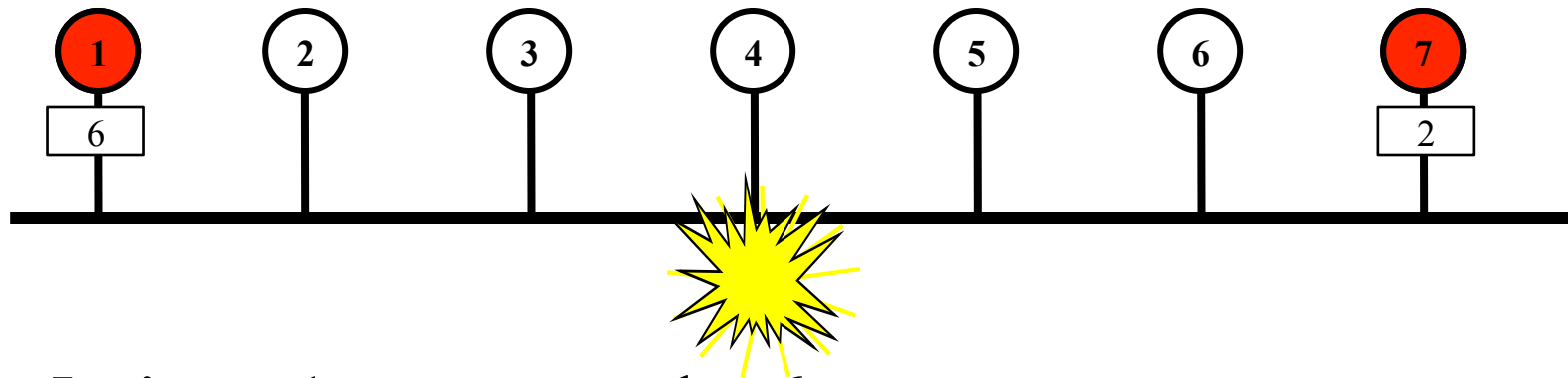


- Node 2 wants to send a message to node 6
- Node 2 listens to the medium
- Node 2 doesn't send
- Now node 2 can send his message

CSMA and collisions

- CSMA involves sharing a channel
 - Multiple senders can all put messages on the same channel
- No master, no advanced reservations
- So more than one sender can try to use the channel at once
- When that happens, their messages *collide*
- Which corrupts both messages, making them useless (for most purposes)

Collisions can happen



- Let's say 1 wants to send to 6
- He listens and hears nothing
- And 7 wants to send to 2, at about the same time
- He listens and hears nothing
- So they both send
- Both messages are destroyed

CSMA variants

- CSMA/CD
 - Carrier Sense Multiple Access/Collision Detection
 - Essentially, listen to determine if the channel is in use and send if it isn't
 - Continue listening to detect if collisions occur
- CSMA/CA
 - Carrier Sense Multiple Access/Collision Avoidance
 - Essentially, listen longer to determine if the channel is in use and send if it isn't

CSMA/CA- I'm Not Listening!

- CSMA/CA listens before sending
- But some versions don't listen during sending
- So they don't detect collisions by listening
- So what do they do?



Why not listen?

- Not practical for some wireless channels
 - Need to send and listen at the same time
 - Sometimes expensive to build equipment that does both well simultaneously
- The hidden terminal problem
 - We'll get to that a little later

Acknowledgements

use acknowledgements to send without listening - this means send message saying that you've got it

- One way to detect collisions without listening during send
- Receiver instantly acknowledges received message on channel
- Using the channel, of course
- Acknowledgements are short
- But do take up channel space
 - Possibly leading to collisions themselves

A note about CSMA

- Benefits:
 - CA avoids always colliding after idle if multiple parties want to send
 - Non-persistent, like CA, helps avoid collisions but also avoids work during busy period (spin-lock)
 - CD reduces the impact of a collision
- These are NOT mutually exclusive
 - Though we usually talk about CSMA/CD or /CA
- There are other optimizations

Ensuring channel capture

- Start sending data
 - Data can collide in unpredictable ways
 - Someone else might have just started to send, but it hasn't gotten to us yet
 - Message might be very short – how long do we wait to check to see if it worked?
- Solution: preamble
 - Floods the channel before sending message
 - Also enables frame sync

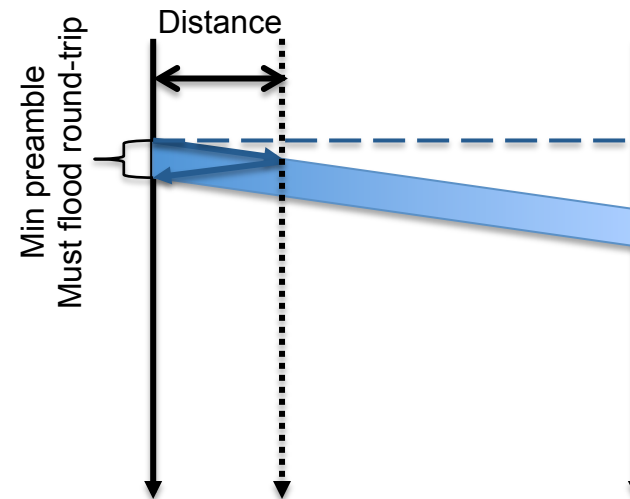
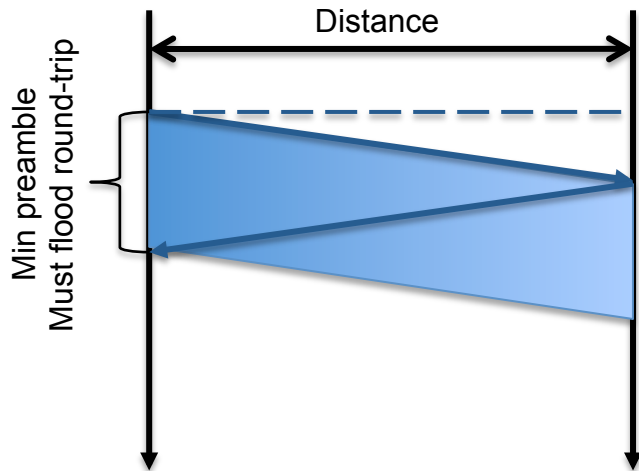
send something I don't care about - if received, unlikely have a collision

Flooding the Channel

- Put an unimportant signal onto the (apparently idle) channel
 - The *preamble*
- Keep it there until you're sure that no one else is sending
 - Implying long enough for you to hear everyone else who might be flooding
 - And for them to hear you
- If preamble is trashed, someone else is sending too

More on flooding

- Don't start sending until you know you have the whole channel
 - If you can send a round-trip bit with no collision, then the channel is yours
- At higher speeds, a given preamble is “shorter”
 - So the round-trip distance protected is less



Limitations of no-master sharing

- Channel length
- Protocol overhead
- Capture effect
- Need for a single, shared channel

Preamble vs. channel length

- Preamble
 - 7 bytes = 56 bits
- Maximum shared link size
 - @3 Mbps = 1866 m
 - @10 Mbps = 560 m (set to 500 m)
 - @100 Mbps = 56 m
 - @1 Gbps = 5.6 m

Protocol overhead

- Converse of channel length limit
- Faster symbol rate = longer preamble
- Longer preamble = higher overhead

Backoff

- What do you do if your packet is trashed when someone else also sends?
- Try again
- But not right away
- Wait some period of time before re-sending
- That's *backoff*
- Commonly used in many non-master channel sharing schemes

Capture effect

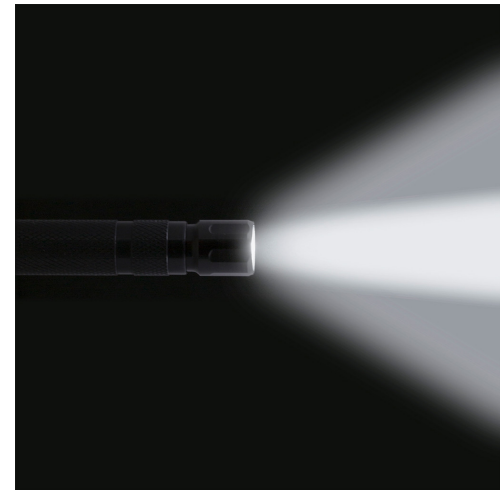
- Collision backoff is not fair (known in 1994)
 - Ethernet backoff picks a random value in a range that increases with each failure
 - A & B collide
 - Both pick from the small initial interval
 - A wins and transmits
 - A & B collide
 - A now picks from the small initial interval
 - B picks from a larger, second-try interval
 - A usually wins (repeatedly)
 - A is rewarded by having its interval reset whenever it wins

Single, shared channel limit

- Signal power
- Protocol
- Topology
- Hidden terminal

Signal power

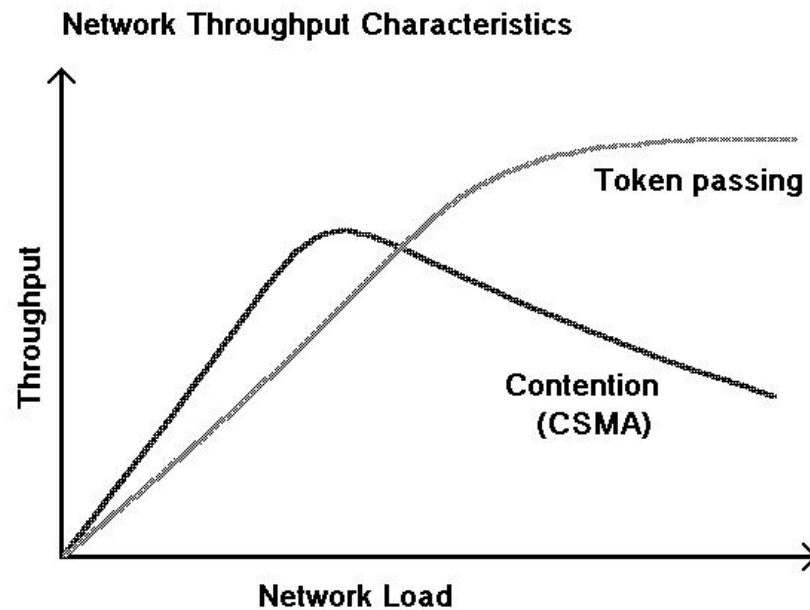
- Distance
 - Power absorption (except for a vacuum)
 - Most beams spread out
- Number of receivers
 - Power needs to increase for everyone to get “some” of the signal



Result: effective distance limit

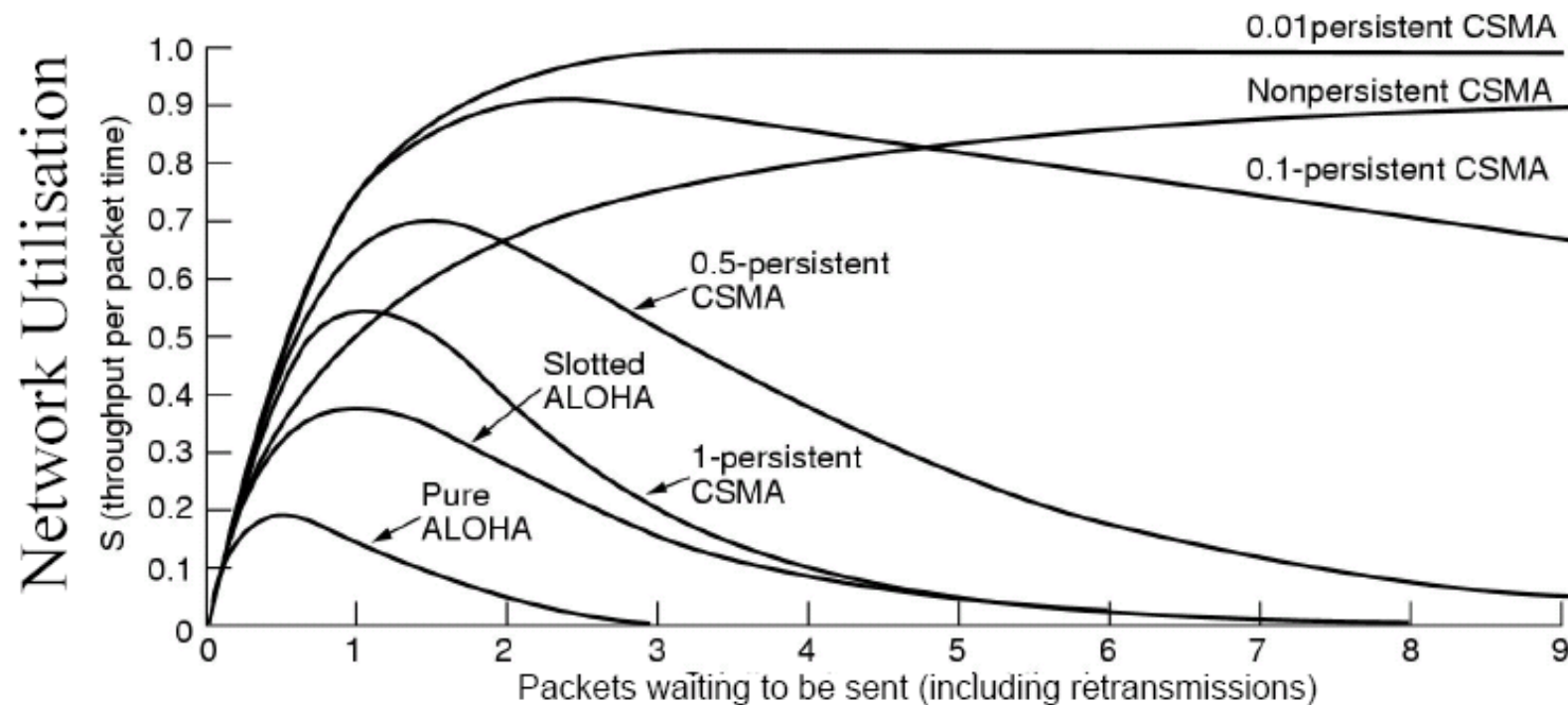
Protocol effects

- Sharing can be inefficient
 - Collisions = no transfer



Protocol variations

- Slight variations can have large effect



Protocol effect implications

- Channel negotiation takes time
 - During which you might lose what you send
 - And you can't know until you try

Result: strict distance limit, efficiency limit

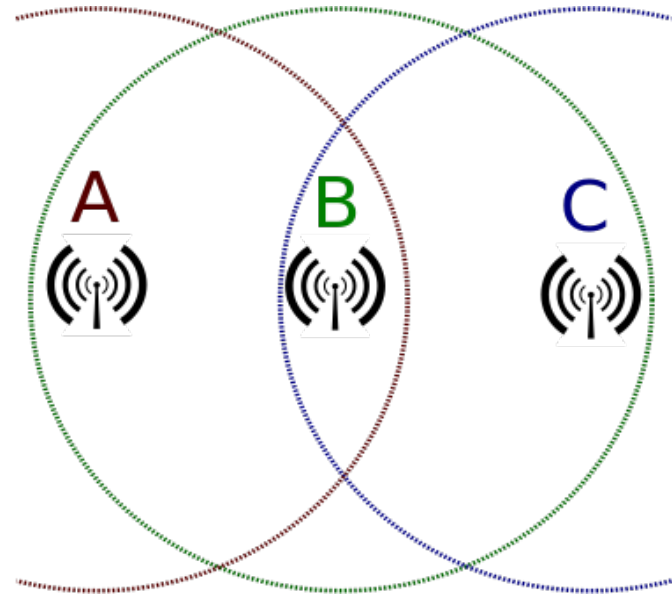
Topology

- A single channel can be hard to deploy
 - RF doesn't go around corners
 - Wire doesn't go where you want



The hidden terminal problem

- Incomplete sharing
 - Not all nodes reach all others
 - CSMA no longer works
 - A and C won't know their transmissions collide
- Acknowledgements can help



Naming implications for shared medium

- Sharing is sharing
 - Same rules about uniqueness of names
 - In 1:N, only destination name must be unique
 - In N:1, only source name must be unique
 - In N:N (with no other info), source/destination pair must be unique
 - And there could be N^2 pairs
- How do we achieve uniqueness?
 - A priori coordination

The cost of naming

- Worst case: N^2 names
 - Costly to add one more party
 - Does not scale!
- Simpler cases: N names
 - Adding one party adds at most one name
 - One more receiver in 1:N
 - One more sender in N:1
 - Need to be sure chosen names are unique
 - Scales well

Shared media naming techniques

- Central authority
 - Two-level delegation
 - First three assigned per-organization (OUI)
 - Rest assigned locally
 - IEEE 802.* addresses are 48 bits (6 bytes)
 - IEEE also assigns 64 bit addresses
 - ATM NSAP
 - Multi-level hierarchy, starting at ITU, including IANA
 - IPv4 addresses
 - Multi-level delegation, starting at IANA
- Self-assignment
 - IPv6 local part is self-assigned, then check for duplicates (“roll again!” = “Duplicate Address Detection”/DAD)

Other names

- Name for “everyone”
 - Enables native (one-step) broadcast
 - Often “all 1’s”
- Name for “a group”
 - Enables native (one-step) multicast
- Name for “I don’t have an address yet”
 - Often “all 0’s” (for another day)

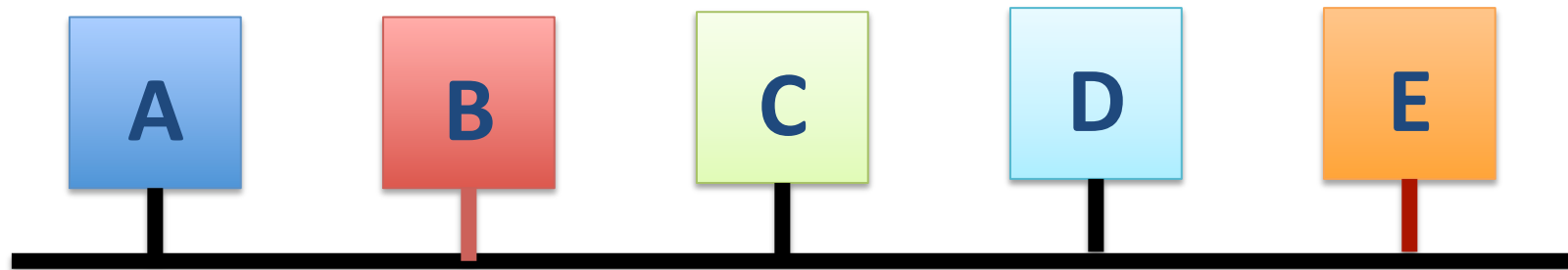
More switching

- Recall:
 - Switching emulates sharing
- What makes it useful?
 - Simpler wiring (direct to closet)
 - Independence
 - Enhanced coordination



Simpler wiring

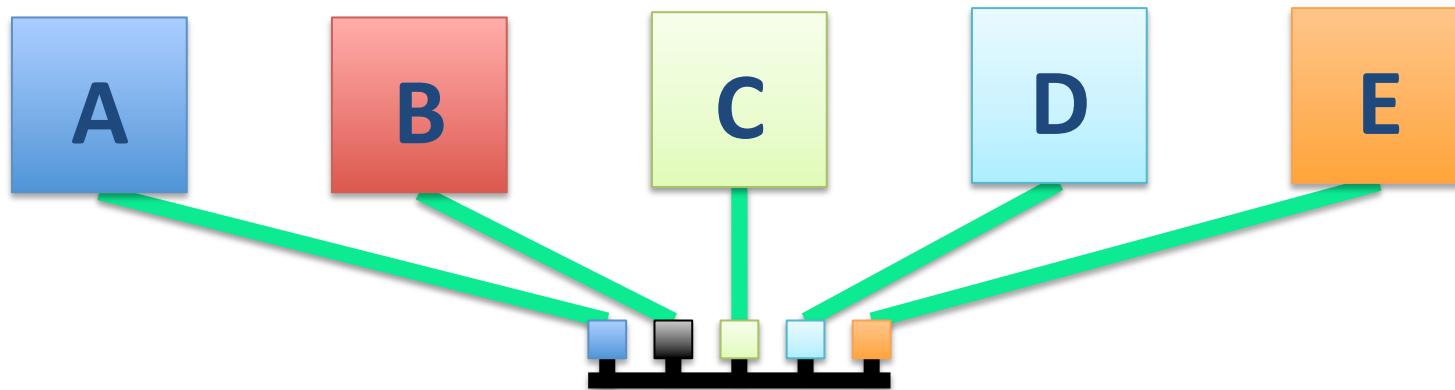
- First step: everyone on a bus



Limited by the bus path

Simpler wiring 2

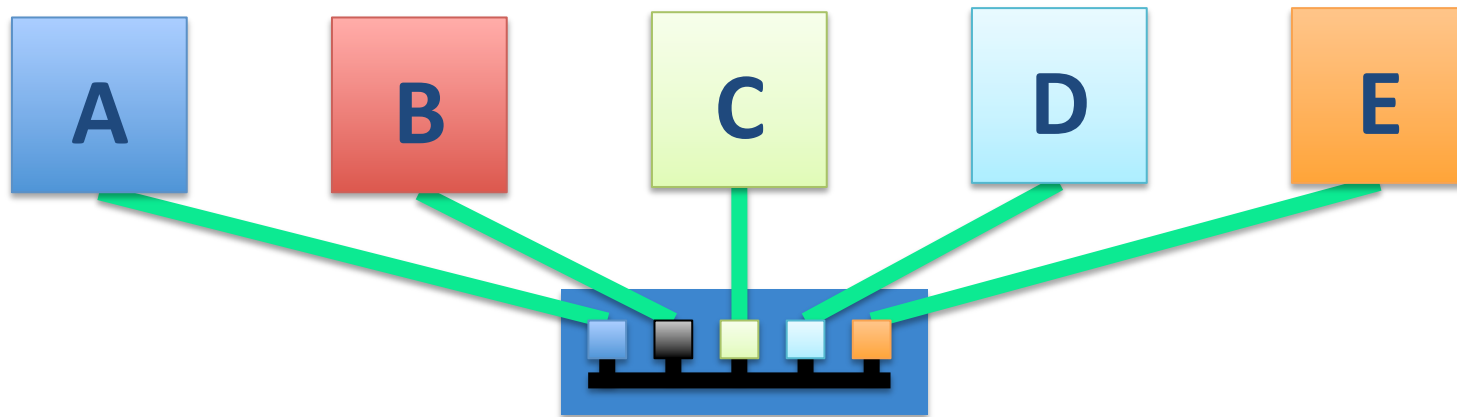
- Second step: delegate to a shorter bus



Move per-node smarts together
E.g.: Ethernet MAU

Simpler wiring 3

- Third step: box it up!



Simpler to manage and operate

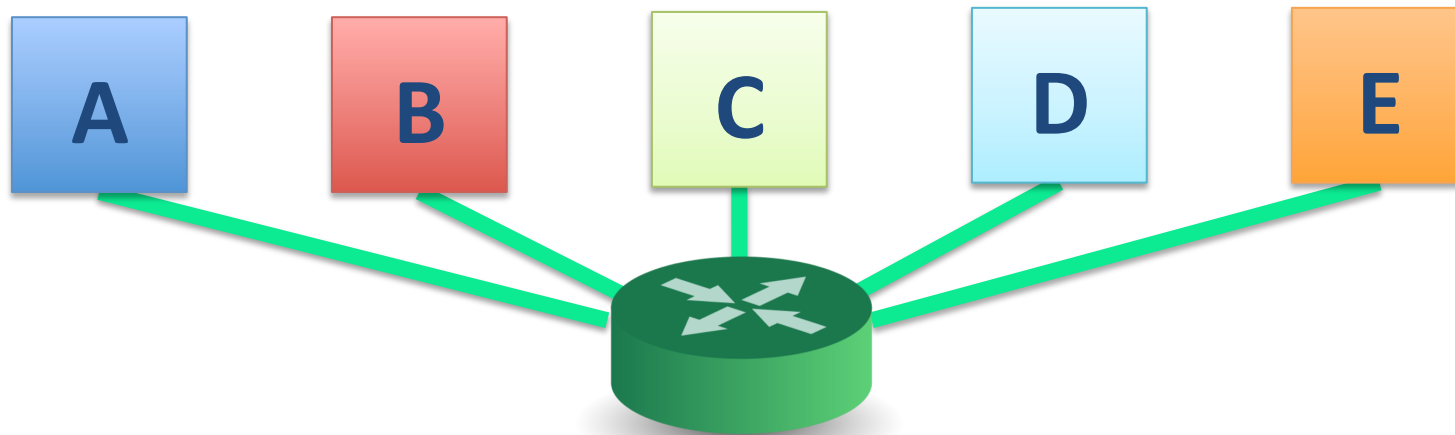
A note about MAC protocols

media access control

- Media access control
 - Just a name
 - Protocol to control shared access
 - NOT NEEDED without shared access!

Simpler wiring 4

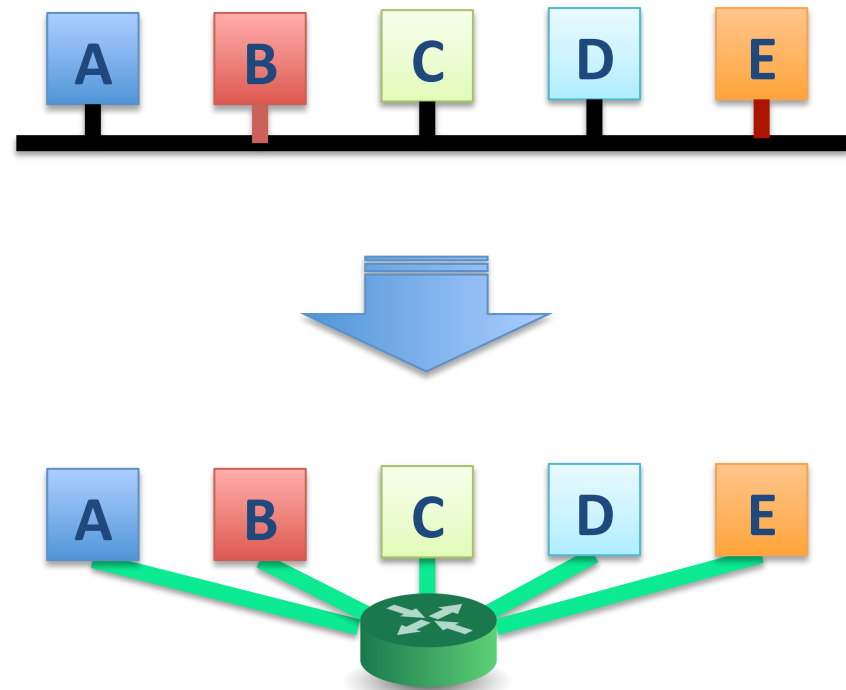
- Final step: who cares what's in the box?
 - There are many ways to build N:N exchanges



- What else changed?

Non-sharing protocol!

- We can turn channel sharing
 - A sharing protocol
- Into central switching
 - A 2-party protocol to the switch
 - A non-sharing protocol
- NOTE THIS!



Benefits of independence

- Add/remove nodes without affecting others
 - No need to shift wires
 - Dead (or mostly-dead) node can't contaminate the network



Enhanced coordination

- What can a switch see?
 - Everything at once, quickly (shorter channel)
- What can a switch do?
 - Coordinate! (takeover role of master)
 - Enforce long-term fairness, avoid starvation

So what's a switch really?

- A way to emulate a shared link
 - Without the physical constraints

Switch examples

- Ethernet
 - Emulates an RF channel
- SONET
 - Emulates a wire
- ATM
 - Emulates a phone wire (in particular)

Not all of these emulate sharing!

How do switches work?

- Shared media
 - An internal star coupler, bus, or memory
 - A control algorithm to share the in/out links
 - Usually TDMA on the in/out links
- A bunch of relays

Goal: scalable communication

	Number of channels for N nodes	Maximum distance between two nodes
2-party channels (direct point-to-point)	N^2	Limited by direct signal

Can we do better?

Goal: scalable communication

	Number of channels for N nodes	Maximum distance between two nodes
2-party channels (direct point-to-point)	N^2	Limited by direct signal
Shared media (shared multiparty)	1 (<M)	Limited by signal sharing and MAC protocol

Good for small groups – what about large?

Sharing and relaying

- Emulate full connectivity
 - Networking to enable communication
- Reduce connection cost
 - Sharing can be expensive or limiting

Sharing vs. relaying

- Sharing
 - Increases endpoint work
 - Simple topology
 - Efficient small scale
 - Poor number scaling
 - Poor size scaling
- Relaying
 - Spreads the work
 - Allows complex topologies
 - Efficient at large scales
 - Good number scaling
 - Good size scaling
 - If designed well

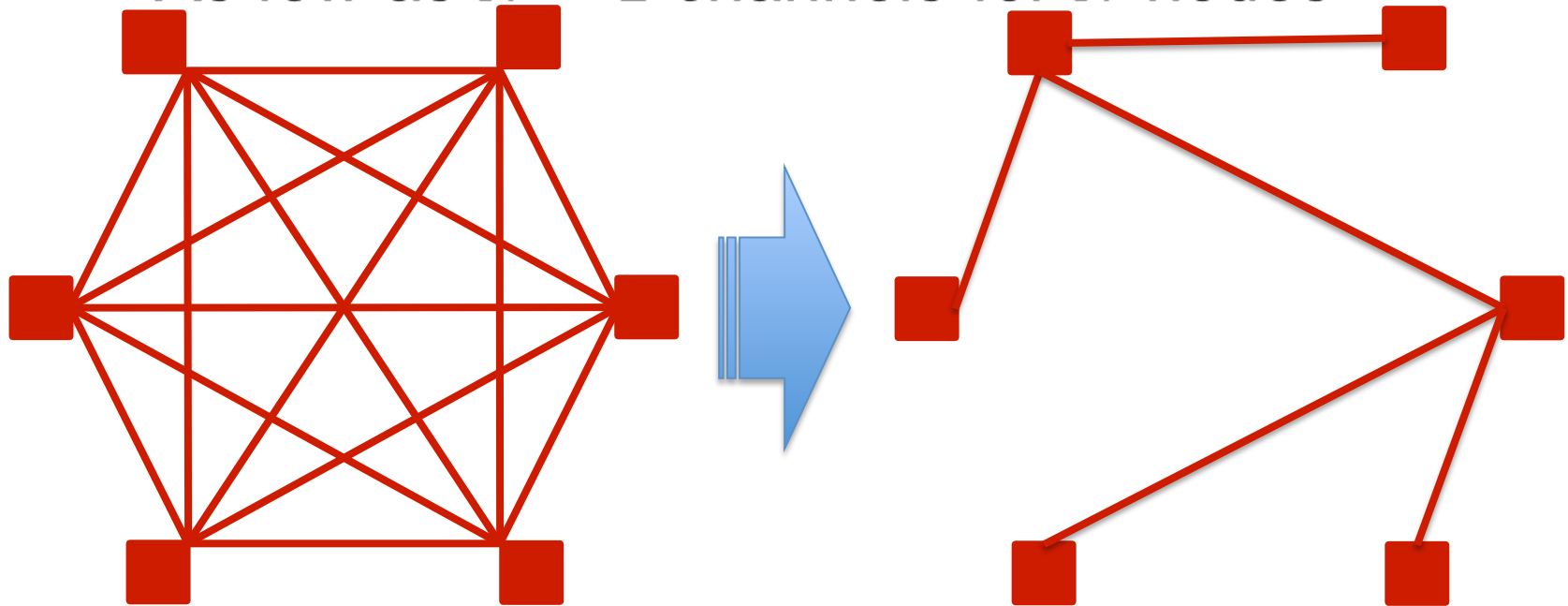
Relaying to the rescue!

- Communicate through a third party
 - A to B, B to C = A to C
 - “Transitive closure”



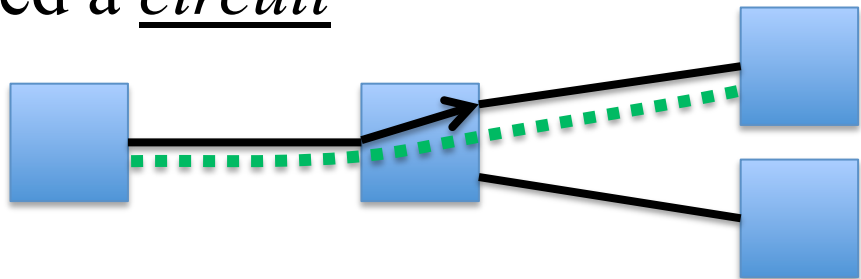
How does relaying help?

- Allows us to remove some of the channels
 - As few as $N - 1$ channels for N nodes



How can we relay? #1

- 2-party channels
 - Media for action at a distance
 - Telephone lines originally direct copper pairs
- Relay using actual switches
 - Switch selects alternate direct copper paths
 - Continuous path is called a circuit

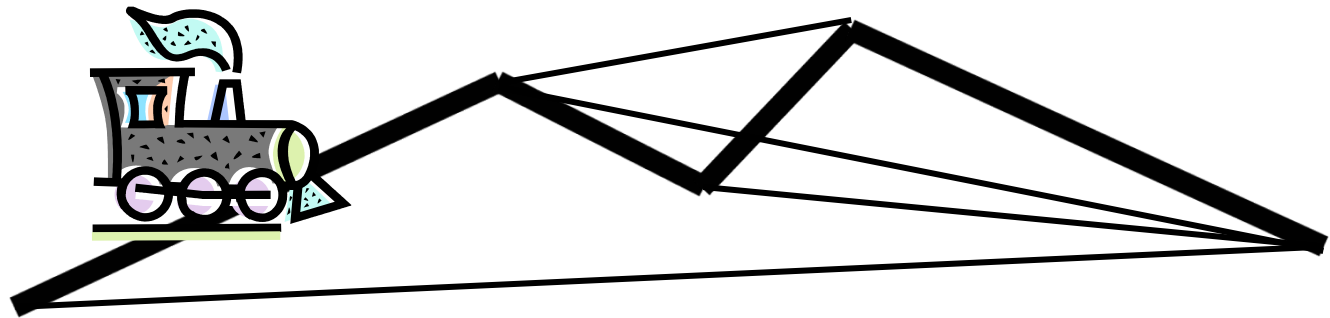


The good, the bad, and...

- Good news
 - Relaying can reduce the number of links
- Bad news
 - This kind of relaying limits how many pairs can communicate at once

Circuits – a sure thing

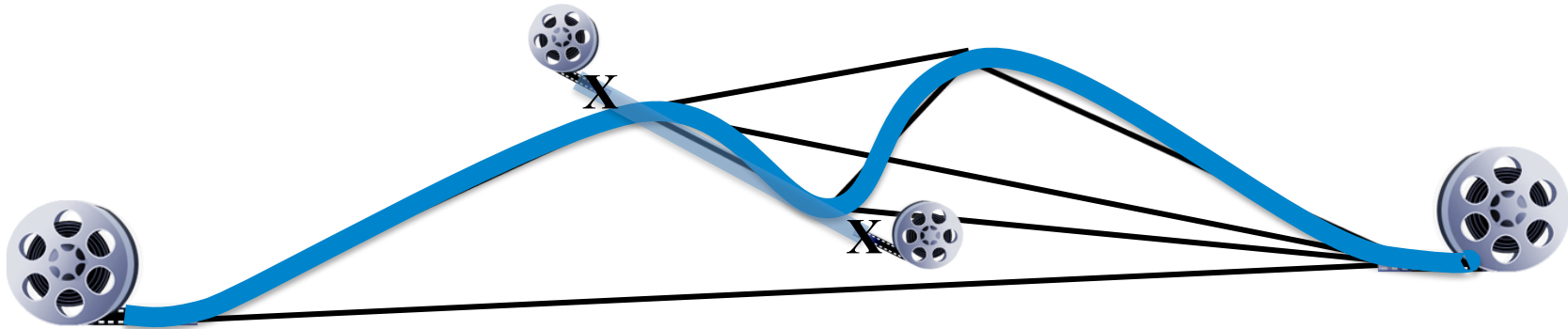
- Trains on a track
 - Scheduled in advance
 - Allocated whether in use or not
 - Resources locked along entire path
 - Path is fixed



- Guarantees no competing traffic
 - Fixed delay, fixed jitter, fixed capacity
 - Can't share resources concurrently

Circuit movie transfer

- One long, continuous path
 - No need to reformat the movie
 - Lossless, in-order delivery



- Along a *single, static* path
 - That blocks everything else until you're done

Circuits – pros

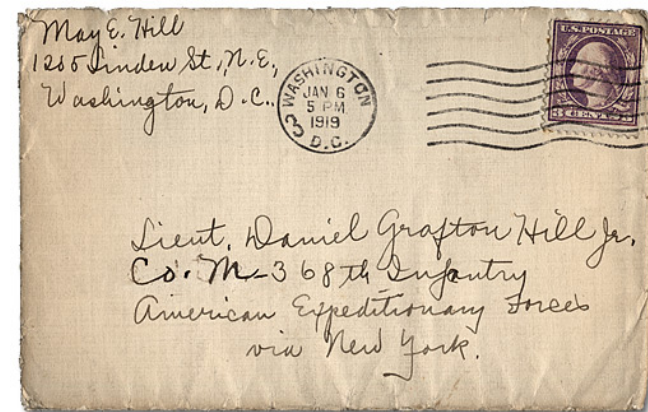
- Guaranteed service
 - Fixed capacity
 - Fixed delay
- Advance knowledge
 - Properties known in advance
 - Advance reservation (if use-bounded)
- Efficiency within a single stream
 - No overhead for processing, labels, etc.
 - The circuit *is* the label (no names after setup)

Circuits – cons

- Fairness on a per-circuit basis
 - Per reservation
 - At the time of reservation
 - At the time of use
- Path blocking
 - Resources blocked (even if not actively used)
- Capacity limited
 - Min. of per-hop capacity of a single path

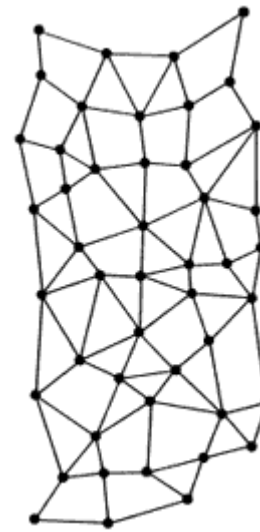
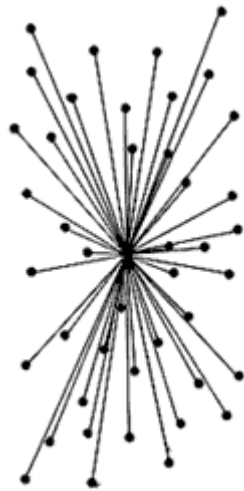
How can we relay? #2

- Packets to the rescue
 - TDMA relaying
 - Allows circuits to be shared
 - Avoids blocking of cross-traffic
 - “Packet” coined in 1968
(Donald Davies)



Baran's study

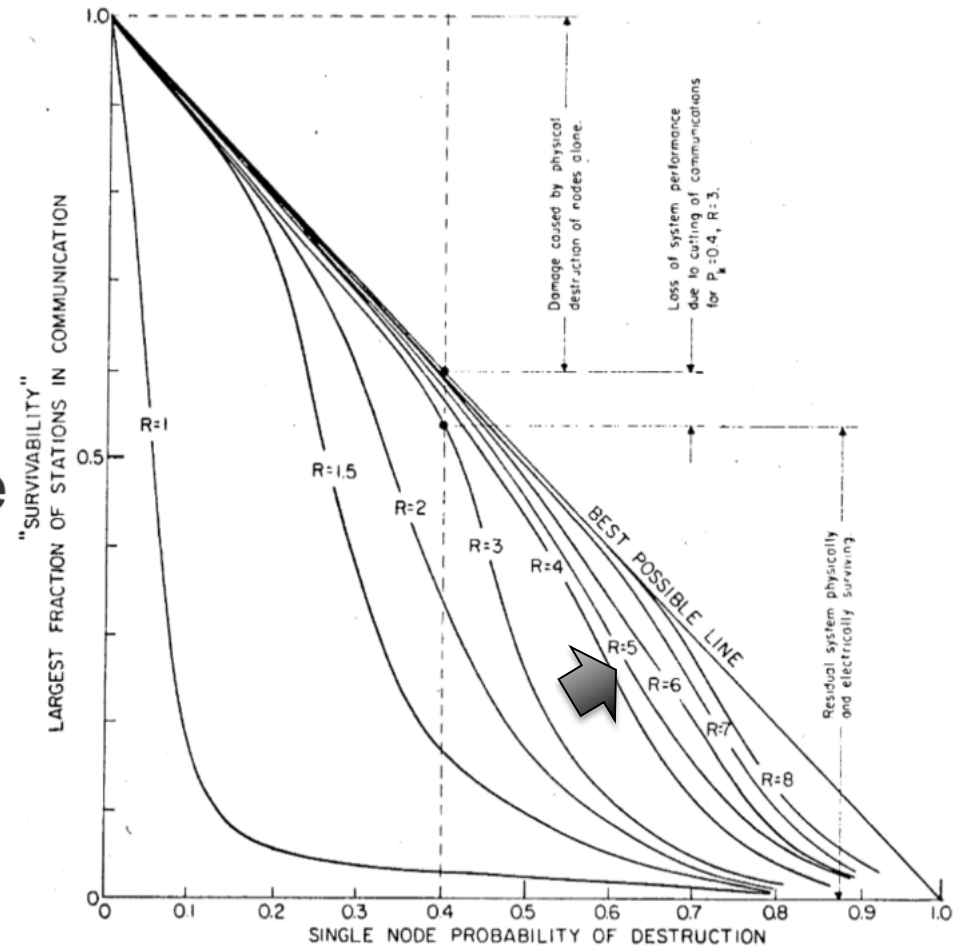
- Compared central and distributed nets



- Divide messages into “blocks” (packets)

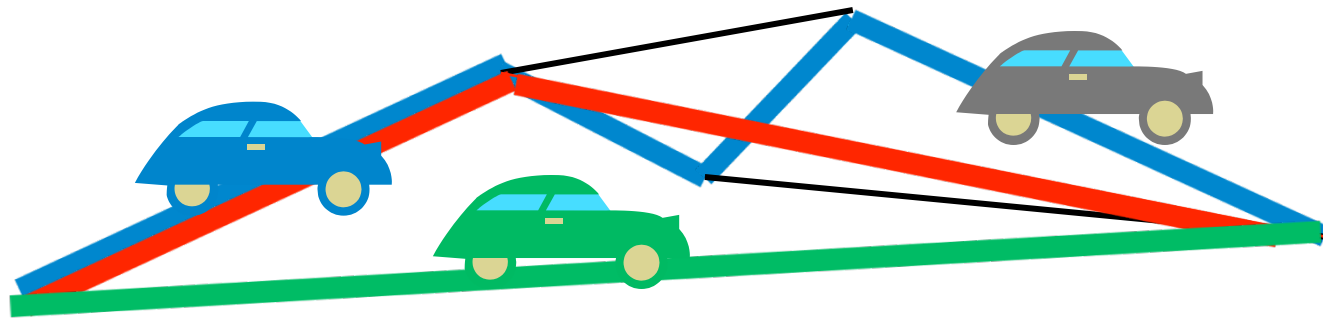
Baran's insight

- N links is minimum
 - But one link or node failure disrupts others
- N^2 is maximum
 - No link or node failure disrupts others
- $4N$ is nearly as good
 - For some specific but reasonable assumptions



Packet example

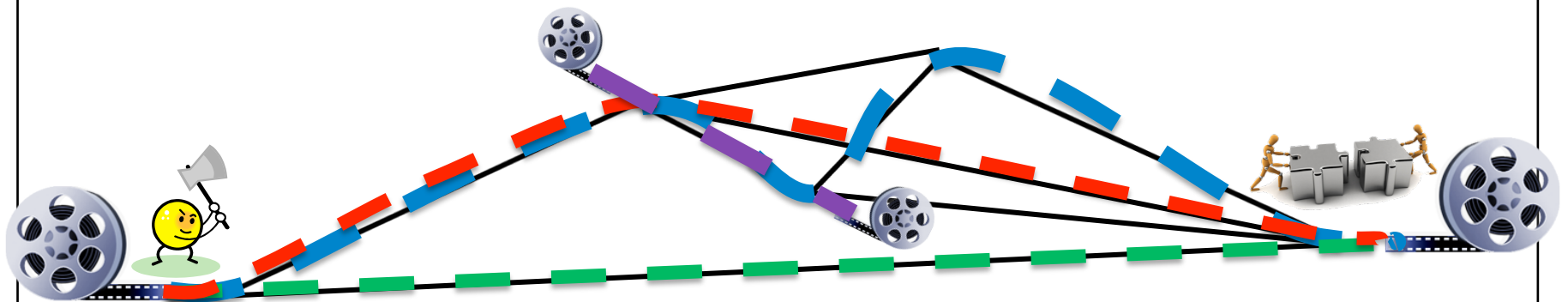
- Cars on a highway
 - No need to schedule
 - Resource used only during transit
 - Path can vary, even given identical headers



- Focuses on sharing
 - Aggregate, concurrent resource use
 - Results in variable delay, variable jitter, variable capacity, and potential for loss
 - Each car is independent, so these variations not too important

Packet movie transfer

- Fragmentation
 - Split into chunks, label for reordering
- Reassembly
 - Gather and restore the stream



- Sharing
 - Concurrent transfers supported
 - But there are relationships between the packets . . .

Packets – pros

- Sharing
 - “Stat-Mux” (statistical multiplexing) gain (2x-10x)
- Fair over shorter time-scales
 - More dynamic and agile
- Avoids path blocking
 - Brief uses share better and complete faster
- Allows multipath
 - Concurrent use of multiple paths
 - Can increase capacity for a given transfer
- Allows dynamic path variation
 - Can route around outages, delays

Packets – cons

- More work
 - Pack/unpack
 - Compute checksums
 - Manage reordering, loss
- Capacity overhead
 - Addressing – to guide the chunks
 - Demultiplexing fields – allowing sharing
 - Signaling fields – to help undo chunking effects
- Storage required
 - Buffering to accommodate reordering, loss

Circuits vs. Packets

- When circuits win:
 - Data patterns are mostly predictable
 - Sharing isn't important
 - Service guarantees are important
 - Data length is long (path setup is worth the benefit)
- When packets win:
 - Data patterns are unpredictable
 - Sharing is important
 - Guarantees are more flexible
 - Data length is short (relative to path setup cost)

Goal: scalable communication

	Number of channels for N nodes	Maximum distance between two nodes
2-party channels (direct point-to-point)	N^2	Limited by direct signal
Shared media (shared multiparty)	1 ($<M$)	Limited by signal sharing and MAC protocol
Relaying	$O(N)$	Unlimited

Sounds too good to be true...

Summary

- We can share a channel without a master
 - If parties can all listen to what's going on
- We can overcome some drawbacks of channel sharing by relaying
 - Sending data over multiple separate channels connected together
- Relaying can be done via circuit switching or packet switching

send som