Lecture 14: Data transfer in multihop wireless networks

Mythili Vutukuru

CS 653 Spring 2014

March 6, Thursday

Data transfer over multiple wireless hops

- Many applications:
  - TCP flow from a wireless node to a server on the internet. Must transfer data over multiple wireless hops in unicast fashion.
  - Data collection from sensors to the “sink” node in the sender network. Unicast transfer, with optional in-network processing / aggregation of data.
  - Multicast distribution of data (e.g., video stream) to a subset of nodes in the multihop network.
  - Flooding of information (e.g., code updates) to all nodes in a sensor network.
- This lecture:
  - Issues: intra-path, inter-path interference
  - Fixes with traditional routing, opportunistic routing

Intra-path interference

- Consider the following topology A – B – C – D
- Suppose the route "ABCD…" has been picked by the routing protocol as the best path and forward data using this path. What are the complications?
- A and B cannot send simultaneously because B can't receive and send at the same time. If A sends too fast, B will always be receiving and will not send time to empty its queue.
- A and C cannot send simultaneously because they interfere at B (and could possibly even be hidden terminals, leading to data loss).
- These issues are commonly classified as intra-path interference. TCP performance over multiple hops suffers.

Intra-path interference (2)

- Suppose every link in a path has a rate R. Suppose the path is N hops long. Suppose a sender interferes with nodes up to "I" hops from receiver. Then the effective throughput on this path can only be R/min(N,2+I).
- See the paper "Flush: A Reliable Bulk Transport Protocol for Multihop Wireless Networks" for a complete explanation of this formula.
- In reality, you may not achieve this value because CSMA does not ensure the perfect pipelining and spatial reuse required to reach this value. In the earlier example, A and C might end up sending together and lose packets. A might send too fast and overflow buffer at B. The effective throughput will be lower than the best possible value indicated by the formula above.

Intra-path interference (3)

- Some fixes proposed in research literature:
  - Rate limit upstream nodes, so as to not overwhelm downstream nodes. For example, after A sends a packet to B, it must wait for enough time to ensure that the packet has passed from B to C to other nodes beyond its interference range. Implementing this simple concept in a distributed setting requires some clever thinking (the paper “Flush” in the references has a solution).
  - Enforce some sort of a schedule over nodes to ensure optimal pipelining (the paper “PIP” in the references explores this idea).
  - Put in multiple radios / directional antennas on each node so that transmissions along a path do not interfere.
  - Find a way to decode concurrent transmissions (many creative research proposals exist)

Inter-path interference

- When multiple flows are active at the same time, similar issues arise (tradeoff between sending and receiving, hidden terminals etc.), but on a bigger scale.
- One possible fix: change routing to make paths interference-aware. That is, if one flow has picked one path, another flow can try to pick a non-overlapping path.

Opportunistic routing

- Consider the topology A – B – C – D
- Suppose A sends and B, C both happen to receive the packet. With current routing + forwarding paradigm, B alone is next hop, so C will throw away its copy of the packet, and wait for B to forward it again. Wasteful!
- Instead, it would be great if we can decide on the next hop based on which one of B or C receive the packet.
- Traditional routing – route first, forward later.
- Opportunistic routing – forward first, see who got which packet, decide on the route based on this info.

Opportunistic routing (2)

- One idea: source sends packets. Nodes that will forward data to destination (forwarders) are prioritized based on distance to destination (highest priority closest to destination). After source sends packets, nodes closest to destination forward the packets the destination needs. Only if none of the closer nodes got the packet will the source retransmit again.
- So, in effect, the next hop for a particular packet will be the node that is as close to destination as possible.
- So nodes need to know who got which packet (a bitmap of some kind), and also decide who sends when (CSMA by default does not prioritize)
- Reference on class website ("ExOR") solves these issues. It uses the ideas of batching, and propagates information of who got which packet at the level of batches (to reduce overhead). It also uses some kind of scheduling to let high priority nodes send first.

The idea of “network coding”

- Another idea that comes in useful in opportunistic routing, and also in flooding over multihop is the notion of network coding.
- Instead of keeping state of who got what packet, nodes can simply send linear combinations of packets (along with the coefficients used to create the linear combinations). The receiver can solve a set of N linear equations to recover N packets.
- This process of nodes in the network generating packets is called network coding. In contrast, transformation of application data into a form suitable for transmission at the source is called source coding.
- Applying this idea of network coding to opportunistic unicast and multicast transfers yields interesting results. You can read more in the references