COS-598A, Spring 2017, Kyle Jamieson

L3: SampleRate (Chapter 3)

Indoor Network Measurements

- 45-node indoor wireless testbed
- Mesh topology
- Linux Soekris router with Atheros 802.11a/b/g card
- RTS/CTS protocol disabled
- Link-level acknowledgements enabled
- Use a **quiet** 802.11 channel
- Throughput experiments

Difference from theory: Many intermediate delivery rates exist (Figure 3-2, middle)

- Indoor, 802.11g; vary the bit rate over $45^2 = 2,000$ possible links.
    - Note that most of the 24 Mbit/s links are not operating at 24 Mbit/s, and there are many such links. Similar trends for other bit rates.
        - From previous graphs we would expect very few intermediate links, but that's not how it works in practice.
        - Contradicts theoretical packetized throughput prediction!
    - Bit rate adaptation algorithms must cope

Difference from theory: Does faster always mean more loss? (Figure 3-3)

- Experiment: one sender at a time in indoor network, 1500 byte packets, everyone else listens, *all* links
- 9 Mbit/s vs 6 Mbit/s (*upper right*): 9 works well only if 6 works well. Both use BPSK, same modulation, but different channel coding.
- 6 Mbit/s vs 1 Mbit/s (*upper left*): Some links are better with the newer but faster 6 Mbit/s 802.11g OFDM, some work better with the older but slower 1 Mbit/s 802.11b DSSS.
- Similar trends in newer work with differing numbers of spatial streams

How rapidly does bit rate adaptation need to adapt? (Figures 3-4, 3-5)

- Delivery probability fluctuates rapidly (Figure 3-4), some links more bursty than others with similar average delivery probabilities.
- Each node sends interleaved 1500 byte packets at 6, 9, 12, 18, 24 Mbit/s; divide time into *t* millisecond

intervals over which an “omniscient” algorithm chooses the best throughput bit rate.

  - Diminishing gains past $t = 10$ second adaptation interval
  - What about $t < 50$ ms (not shown) – some measurement studies claim the indoor wireless channel stays the same for about 30 ms, so there *might* be more room for improvement (later work with walking speed mobility affirms this).

Can SNR predict delivery probability? (Figures 3-6, 3-7)

- Figure 3-6: Each graph = one bit rate, each point = one link
  - x axis? *average* SNR
  - y axis? *average* delivery rate
  - Takeaway: Can’t choose a useful S/N threshold that predicts delivery
- Figure 3-7: Each graph = one *receiver*, each point = one sender (all 1 Mbit/s rate)
  - This factors out receive radio hardware variability, better relationship
  - Multipath different on different links though.
- Across all links, picking a S/N threshold that predicts delivery is hard, so SampleRate abandons this.
  - But subsequent work showed benefits (Effective SNR, will read later).

Classifying links' bitrate-loss relationships (Figures 3-8, 3-9)

- Three example links shown here (Figure 3-8).
- *Grey bar*: ideal throughput; *black bar*: actual throughput achieved, averaged across time.
- Definitions (count of types in Figure 3-9):
  - *Steep*: All links either > 90% delivery, or < 10% delivery
  - *Gradual*: Gradually more loss at higher bitrates **but** the best-performing bitrate = highest bitrate with < 50% loss
  - *Lossy*: Best link has delivery rate well below < 50%
- Implications:
  - For steep bitrates, can just pick **highest loss-free** bitrate; but this is disasterous when gradual links exist.
  - For lossy links, need to increase bitrate **despite** loss.

ARF, Adaptive ARF (Chapter 4)

- Original WaveLAN-II (Lucent) algorithm, circa 1996
- Definion: *packet* is new data, may have several *transmissions* (attempts to send the packet).
- **Algorithm**
  - *State*: current bit rate, # consecutive successful transmissions

- Start at highest
  - **Okay** packet is one that required just one transmission. Maintain `okay_transmit_count`
  - **Step down** if no ack after `num_xmits`
  - **Step up** if 10 consecutive "okay" transmissions occur (this is called a *probe*)

- What would ARF do:
  - On the steep links? + (Start at 11, fail, pick 5.5)
  - ..gradual links? – (Get stuck at high rate sometimes, don't step down until complete failure. Once you step up in a Gradual link, less likely to step down immediately than in a Steep link, less loss @ 11 for Gradual, since you need four **consecutive** frame losses to step down.)
  - ..lossy links? – (Once you step down, you may not step up, and get stuck at low rate sometimes.)

- *Adaptive ARF*: Even on steep link, ARF sends probes. **Adapt** step-up parameter (10 transmits), **double** every time increase bitrate and subsequent packet fails.

  - Backoff penalty high for high 802.11g bit rates, so helps.
  - Will AARF do better on gradual? (no) Lossy? (no)

ONOE

- Atsushi Onoe (engineer at Sony/Japan)
- Highest bitrate with < 50% loss

  - makes sense in 802.11b: 11/2=5.5; 5.5/2 ~= 2; 2/2=1, less sense in 802.11a (36,24,18,12,9,6)
  - Maintain num*tx, num*tx_ok
  - Start at highest bit rate like ARF.

  - Each second on the wall clock:

    - if #1: if no successful transmits, step down
    - if #2: if less than 50% success rate, step down
    - if #3: if less than 10% success rate, lose credit
    - if #4: if greater than 90% success rate, gain a credit, step up if you have more than 10 credits.

- what would ONOE do on the steep links? + 1st sec. 11 2nd sec. 5.5 3rd sec. 5.5 … 10th sec. 11 11th sec 5.5 12th sec 5.5

- what would ONOE do on the gradual links? + 1st sec. 11 2nd sec. 5.5 (line 3) 3rd sec. 5.5 NO probe up to 11 periodically every 10 sec. (credits don't accum)

- what would ONOE do on the lossy links? – 1st sec 11 2nd sec 5.5 3rd sec 2 4th sec 1

SampleRate (Chapter 5)

- Basic idea: consider time to send a packet. Track:
  - Average send time @ bitrate (empirically measure)
  - Lossless send time @ bitrate (constant)
- Choose bitrate with least average send time
- Probe some other bitrates
  - Probing happens one every 10 packets
  - Probe a *candidate set* of rates having a **lossless transmission time less than the current average transmission time**
- Expire information that is more than 10 seconds old

Slide: Measuring time to send a packet

- Any problems with this?
  - What about backoff time before transmissions apart from the last one?
  - What about intervening transmissions from other nodes?
  - Not counted in this calculation, but impacts all bit rates about the same.

Slide: SampleRate in operation

- Notation: packets(transmissions)@rate
- upper dest: fails at rate = 11, perfect at rate = 5.5
  - 4(16)@11, switch to 5.5, 100(100)@5.5
  - why won't it try rate = 2, 1? (lossless@2 and 1 > average@5.5)
  - when will it next try rate = 11? (10 sec, after `remove_stale_results` runs)
- lower dest: one retry at rate = 11, 90% 0 retries, 10% 1 retry at rate = 5.5
  - 9(18)@11, 1(1)@5.5, switch to 5.5, 9(9)@5.5
  - why probe 5.5? (b/c 2976 < 3761)
  - why switch to 5.5? (b/c 2976 (avg. xmit time) < 3761, same reason)

Evaluation (Chapter 6)

ARF mostly chooses correct bit rate (Figure 6-2)

- x-axis: static-best throughput
- y-axis: **hypothetical** – take the bitrate that ARF chose most often, plot throughput of static-best for that bitrate
- **Conclusion:** ARF chose right bitrate most of the time! **Hypothesis:** ARF spends too much time trying other bit rates.

Slide: 802.11a indoor throughput

- ARF spends too much time trying other bit rates, especially on the **steep** links

Discussion

- What about interference from other 802.11 traffic?
- What about mobility?
- What about SNR-based rate adaptation?
    - Cellular mobile networks use SNR to adapt
    - Why do we have trouble applying it for 802.11?
    - More recent work in this area by Halperin (SIGCOMM 2010)