On Reliable Wireless Streaming of Real-time Sensor Data

U of Live Sciences, Warsaw

Agnieszka Boruta, Pawel Gburzynski, Ewa Kuznicka

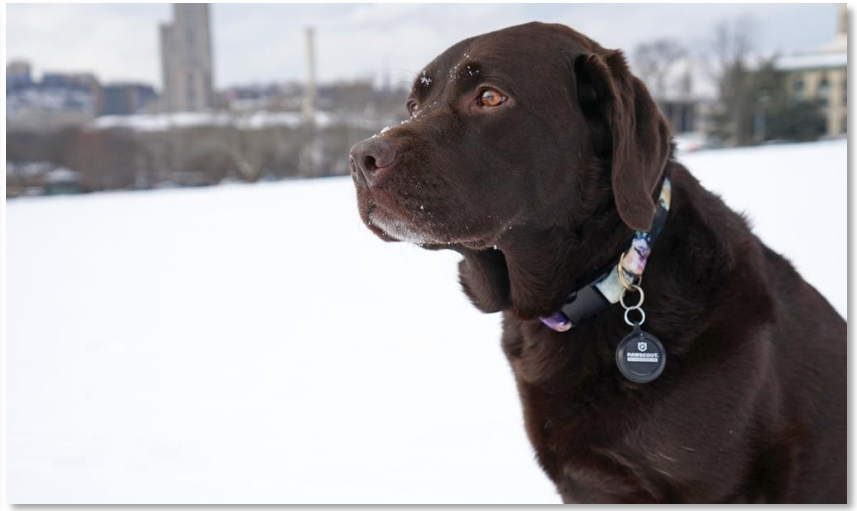
Vistula U, Warsaw

Olsonet Communications, Ottawa, ON, Canada

The context [for a holistic solution]

- An automated system for monitoring the well-being of working dogs
  - not for veterinary diagnostics; intended as a simple practical indicator of stress, overworking, discomfort
  - noninvasive, unobstructive, reliable, durable, adaptable, maintenance-free
  - nothing like this can be bought off the shelf (although lots of commercial activity monitors and [GPS] locators for dogs are available)

What kind of sensing is practical?



IMU: motion classification

- noninvasive ✓
- unobtrusive ✓ [mostly]

What kind of information can be procured?

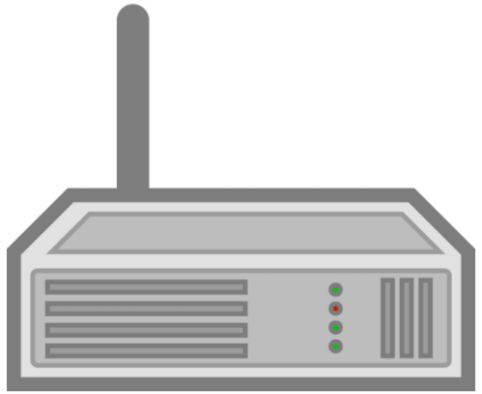
- activity classification / assessment
- anomaly detection, special events

The setup

CC1350 SensorTag (TI)

intended for IoT
20 KB RAM
ARM-based
flexible RF

Tag



Interface device:

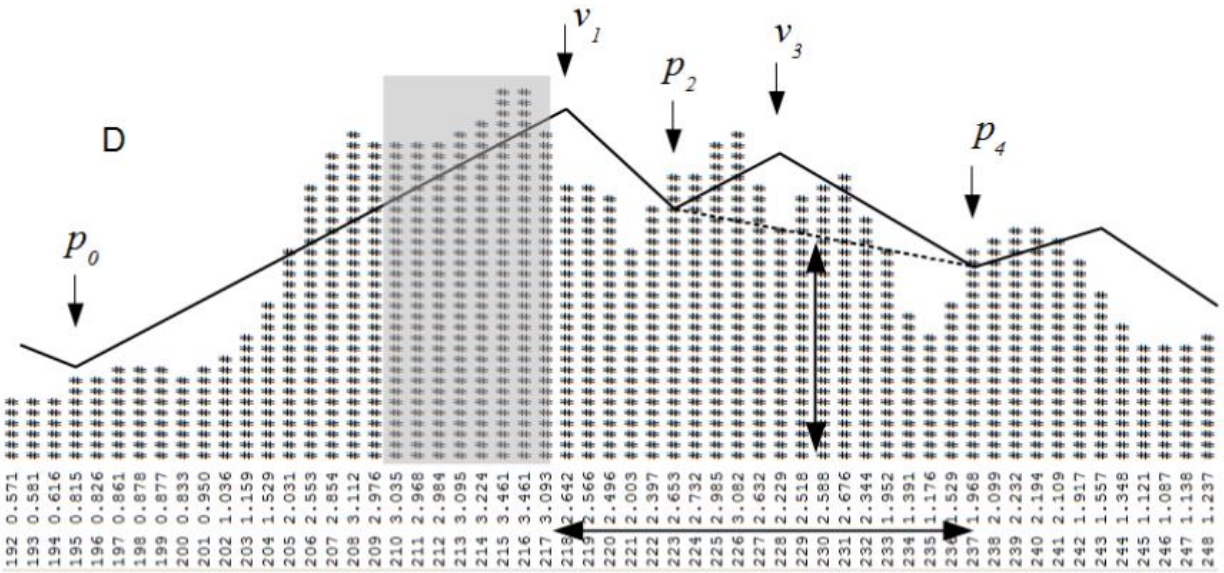
- smartphone
- computer
- cloud?

- off-line analysis of IMU samples collected at reasonably high rates, e.g., 100+ sps
- samples annotated by humans in a controlled environment
- devise classification algorithms to be carried out (primarily) within the Tag
- constraints: primarily energy (and overall cost)

sensing sparsely:	10 μA	2 years
computing (CPU):	2 mA	2 days
radio on:	10 mA	4 hours

A previous project

E. Kuznicka and P. Gburzynski. "Automatic detection of suckling events in lamb through accelerometer data classification." *Computers and Electronics in Agriculture,* vol. 138, 2017, pp. 137-147.



Shows that at least some types of events can be reliably detected within the Tag at a reasonable energy expenditure: 12-50 μA

The experiments

Tag

no room to store too much data

a stream of values:
- look FP, in fact integers
- useful precision ~10 bits
- rates: ~100-500 sps (typically 128, 256)

```
...
0.1211 -0.0352  0.4062
0.1172 -0.0332  0.4062
0.1172 -0.0352  0.4102
0.1172 -0.0332  0.4082
0.1172 -0.0352  0.4102
0.1191 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4102
0.1172 -0.0332  0.4082
0.1191 -0.0352  0.4082
0.1172 -0.0352  0.4082
0.1172 -0.0352  0.4062
0.1191 -0.0352  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4102
0.1172 -0.0332  0.4102
0.1172 -0.0352  0.4082
0.1172 -0.0352  0.4102
0.1191 -0.0352  0.4082
0.1191 -0.0332  0.4082
0.1191 -0.0352  0.4102
0.1191 -0.0332  0.4082
0.1172 -0.0352  0.4082
0.1172 -0.0332  0.4062
0.1191 -0.0332  0.4082
0.1172 -0.0332  0.4102
0.1191 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1191 -0.0332  0.4082
0.1191 -0.0332  0.4062
0.1172 -0.0352  0.4082
0.1172 -0.0352  0.4102
0.1172 -0.0332  0.4102
0.1172 -0.0332  0.4102
0.1191 -0.0352  0.4102
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1191 -0.0332  0.4082
0.1172 -0.0352  0.4102
0.1172 -0.0352  0.4082
0.1191 -0.0332  0.4082
0.1172 -0.0332  0.4102
0.1172 -0.0332  0.4082
0.1172 -0.0352  0.4102
0.1172 -0.0352  0.4082
0.1191 -0.0352  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4082
0.1172 -0.0332  0.4102
...
```

The communication problem

Reliable transfers (data exchange)

- files or data sessions, ACKs, retransmissions (e.g., TCP), no hard RT issues
- a file to be transmitted is stored at both ends

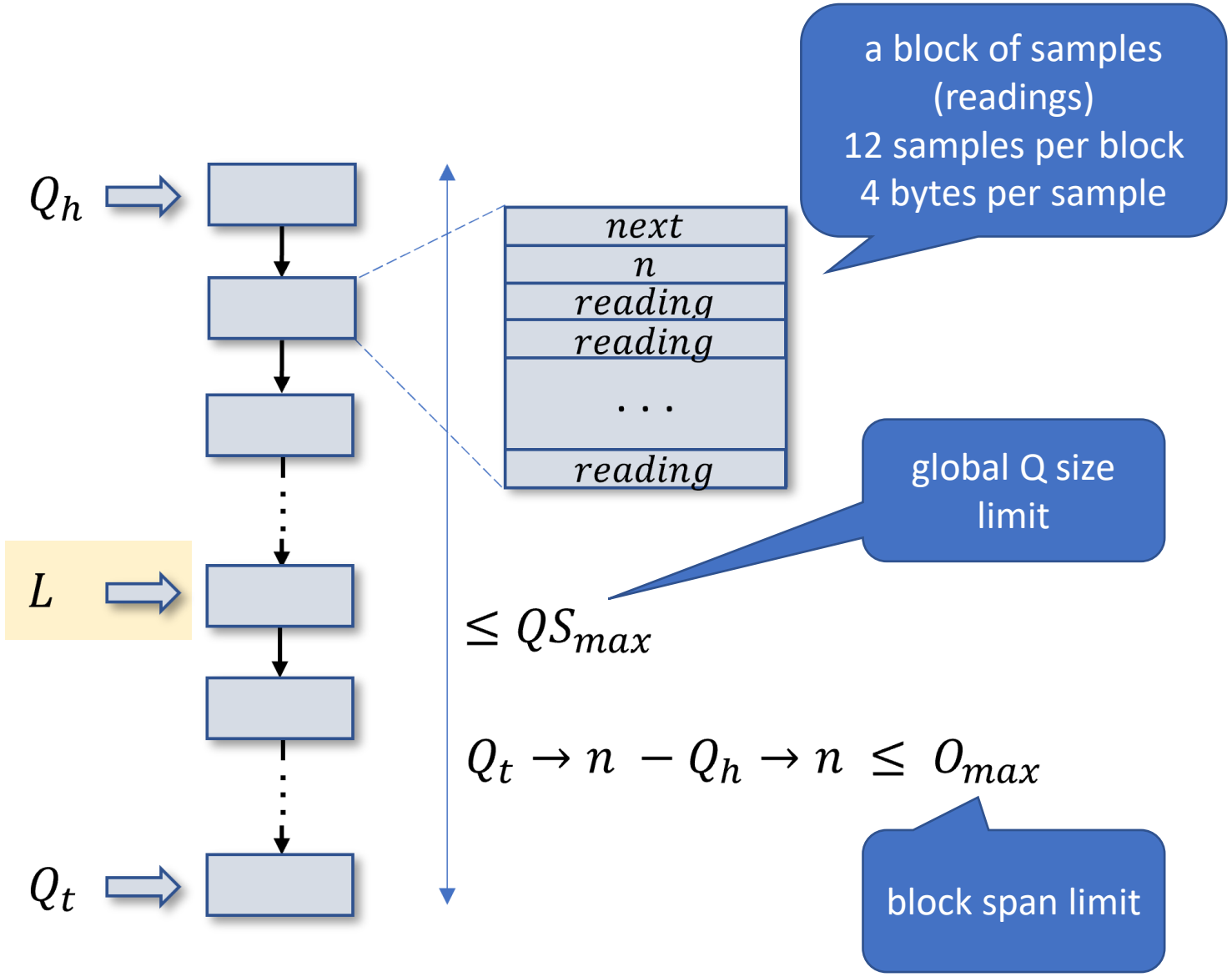
Streaming

- no clear beginning or end (in many cases)
- RT issues, late playback is useless
- losses are acceptable and unavoidable
- bandwidth/rate is tuned for percentage of acceptable losses

We have a non-standard session type:
- the samples amount to a (long) file
- we must stream, cannot store locally
- some losses may be acceptable
- but we really want to receive everything

The solution

- raw RF mode (raw packets)
- a queue of outstanding (unack'ed) blocks of readings
- limits on queue size: global size AND block number span
- blocks removed when ack'ed or when limits are exceeded



The sampling

```
fsm streaming_generator {
  state ST_TAKE:
    word data [3];
    read_imu (data);
    if (CB == NULL) {
      // Next buffer
      CB = (strblk_t*) umalloc (sizeof (strblk_t));
      if (CB == NULL) {
        ...
        sameas ST_WAIT;
      }
      CFill = 0;
    }
    CB -> block [CFill++] = encode (data);
    SamplesTaken++;
    if (CFill == STRM_NCODES)
      add_CB ();
  initial state ST_WAIT:
    when_imu_ready (ST_TAKE);
}
```

a reactive thread (looks like a finite state machine)

read next sample

start a new block

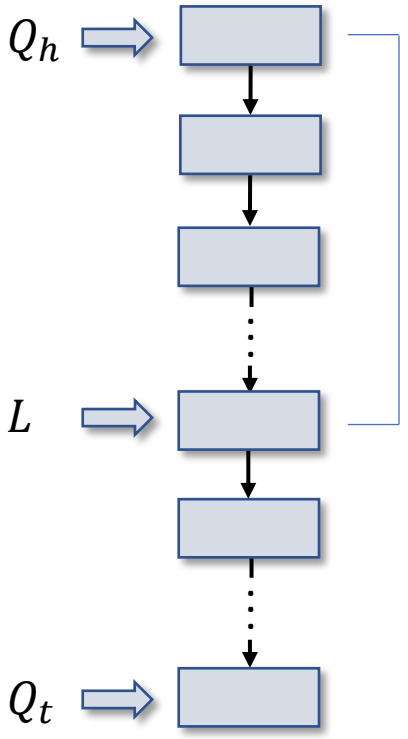
out of memory (never happens)

insert the sample into CB

CB filled, add to queue, nullify CB

triggered by a timer

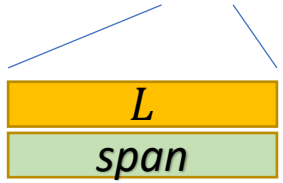
The transmission thread



send up to T blocks (a train) back-to-back (e.g., T = 64)

if run out of blocks, send them at the sampling rate

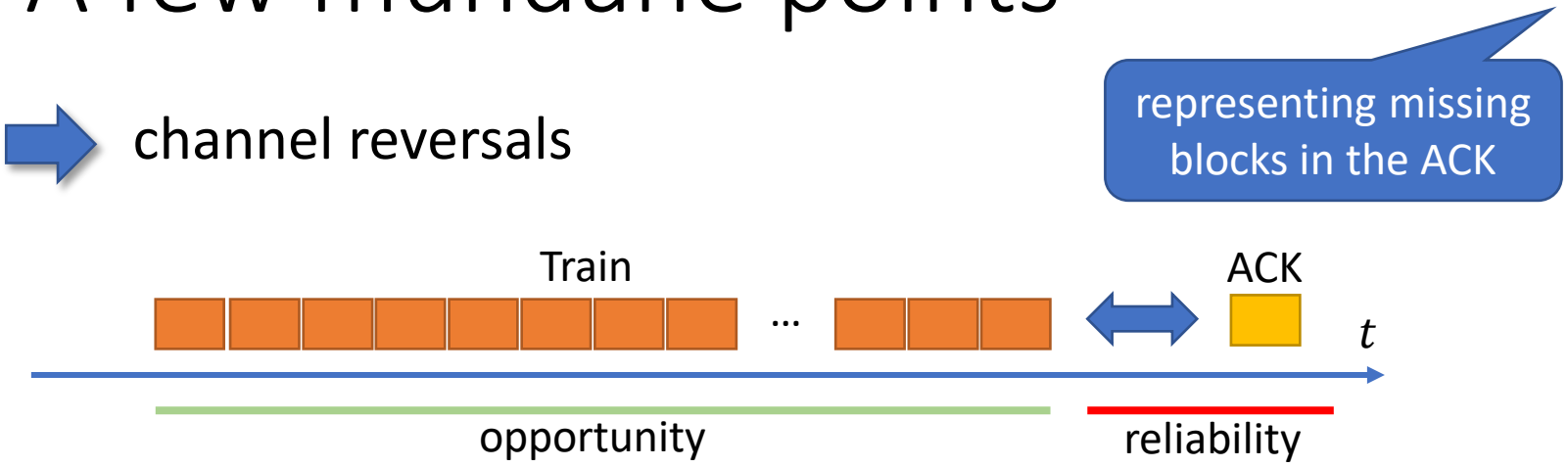
send EOT and wait for an ACK [in a loop]

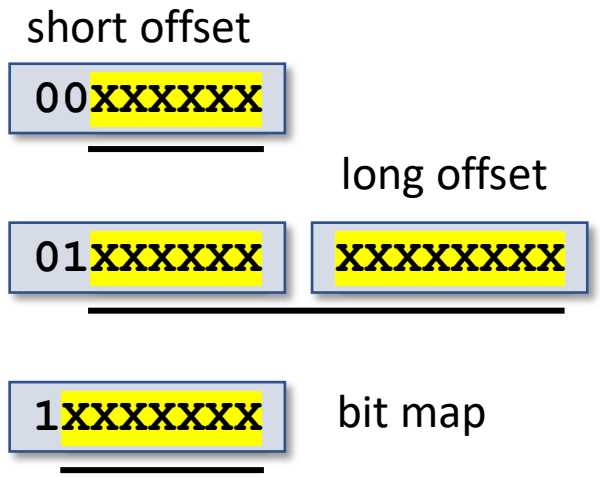


ACK indicates blocks that have not been received; remove from the queue all blocks not mentioned in the ACK

A few mundane points

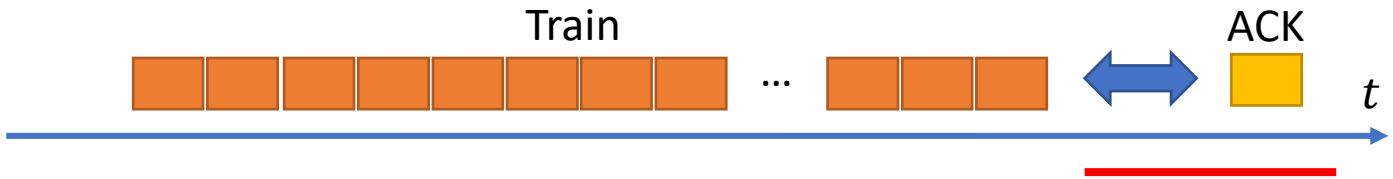
- channel reversals





- ACKs should be simple (single-packet)
- Note that the maximum packet length is small (like 60 bytes or so)
- In the µC world, we are inclined to save on things that "serious" programmers may find amusing

Performance



Channel rate: $R = 50$ kbps [options up to 500 kbps]

1 sample = 30 bits ($3\ x\ 10$) → $Max = \frac{R}{30} \approx 1700$ sps

1 packet (train car) = 60 bytes = 480 bits @ 12 samples = 360 bits → 75% → 1270 sps

packet spacing (cars) = 5 ms/p = 30%+ → 880 sps

ACKs $\approx$ 20 ms → 5% → 840 sps

Errors → $(1 - P_e) \times 840$

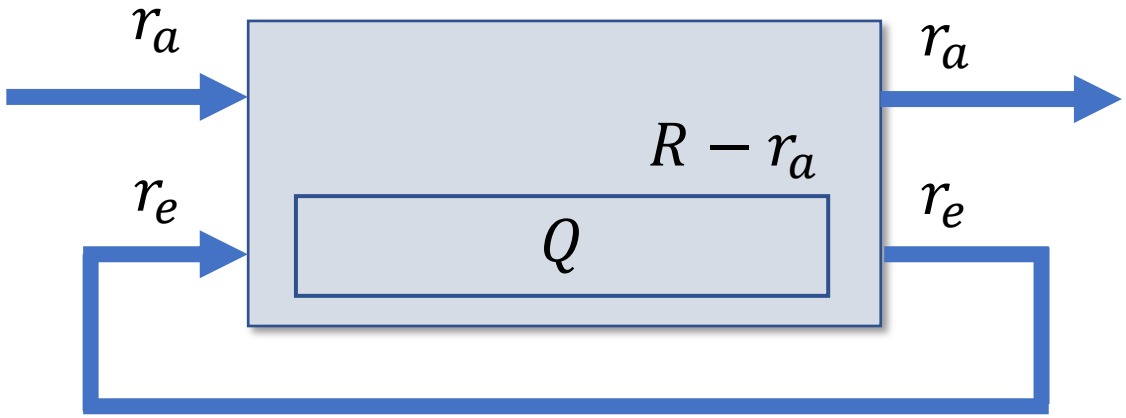
Performance

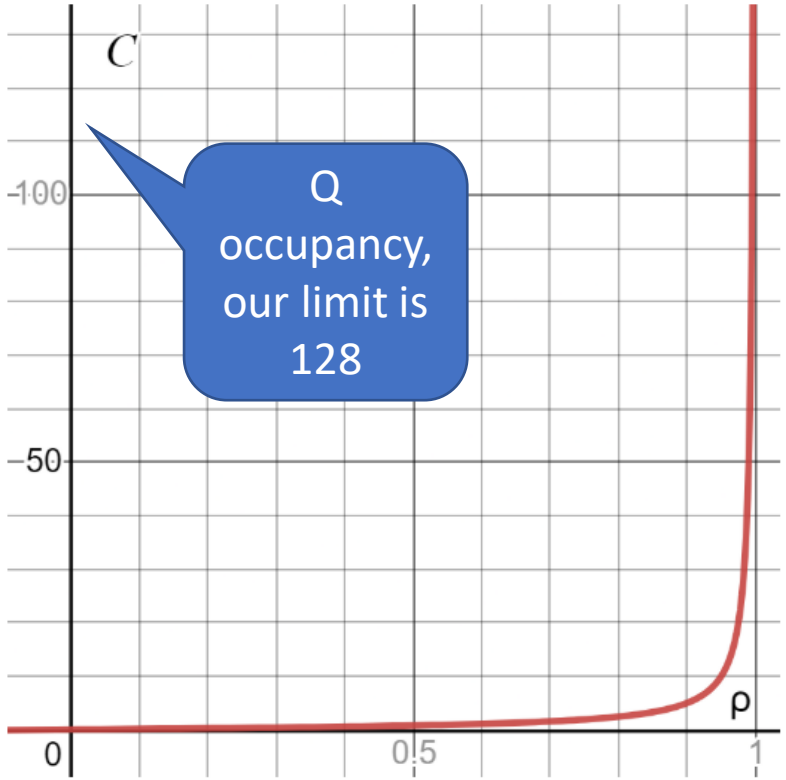
$$r_e = r_a \times \sum_{i=1}^{\infty} P_e{}^i = \frac{r_a \times P_e}{1 - P_e}$$

M/D/1

$$C = \rho + \frac{1}{2}\left(\frac{\rho^2}{1-\rho}\right)$$

Used fraction of whatever bandwidth remains after accounting for r_a





Summary

- There are problems within the (wide) area of so-called IoT that call for solutions “off the beaten path”; this was just one illustration
- One can often get more mileage (sometimes any mileage at all) by following a “holistic” approach whereby one essentially programs the thing “from the bottom”
- This need not connote backwardness, especially within the domain of microcontrollers; one can build things from scratch with a high-level attitude, using tools geared for this kind of development