RFID Counting over Time-Varying Channels

Ziling Zhou, *Binbin Chen*

Advanced Digital Sciences Center

Illinois at Singapore

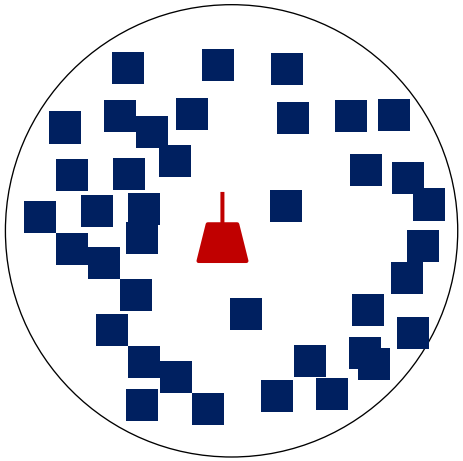
IEEE Infocom 2018, Hawaii

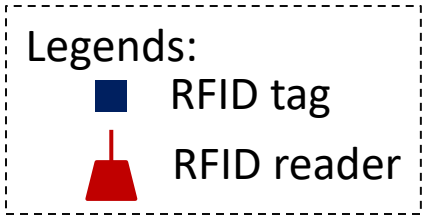
Many applications need counting

RFID technology enables large-scale counting

RFID counting problem

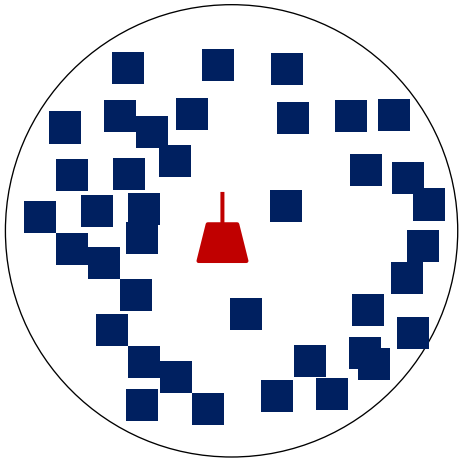
- Consider an RFID reader and n tags that are in its coverage. They run an RFID counting protocol to estimate an $\hat{n} \approx n$
  - Why estimation? Getting the exact n is expensive
- Guarantee: $|\hat{n} - n| \leq \varepsilon n$ holds (say, with 90% probability)
  - ε bounds the relative error, with probability taken over the random coin flips done by the randomized protocol

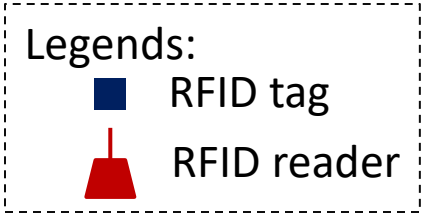




RFID counting problem

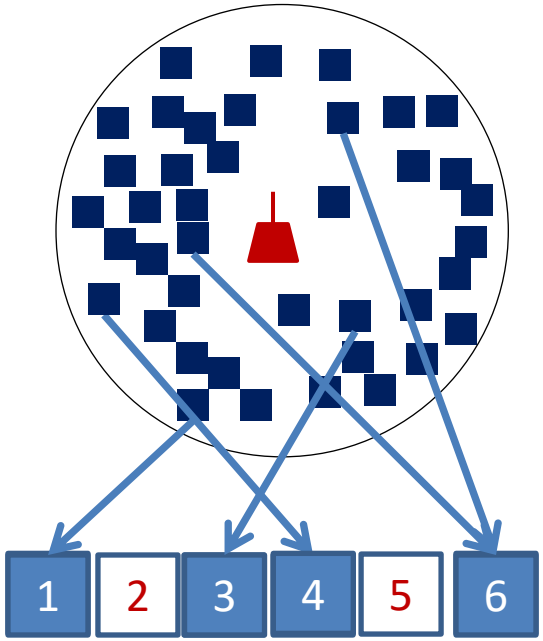
- The reader and tags communicate in synchronized time slots
- In each slot, the reader can send O(1) information to tags, and each tag may then choose to reply or keep silent
- Multiple tags may reply together in a slot
- Assuming no communication error, the reader can distinguish between: a) an *empty slot* when all tags are silent and b) a *non-empty slot* when at least one tag replies
  - i.e., reader sees an *"OR"* channel





An example RFID counting protocol: EZB (INFOCOM'07)

- Using # of empty slots in a randomized trial to estimate n, with each tag participating in the trial (by replying in some random slot) with probability p
  - More empty slots $\Longrightarrow$ less tags



Our previous work (Mobicom'13)

- We establish **lower bounds** for RFID counting protocols:

(Rough) ***Theorem****:*

No RFID counting protocol can estimate with $< \varepsilon$ relative error while incurring overhead of

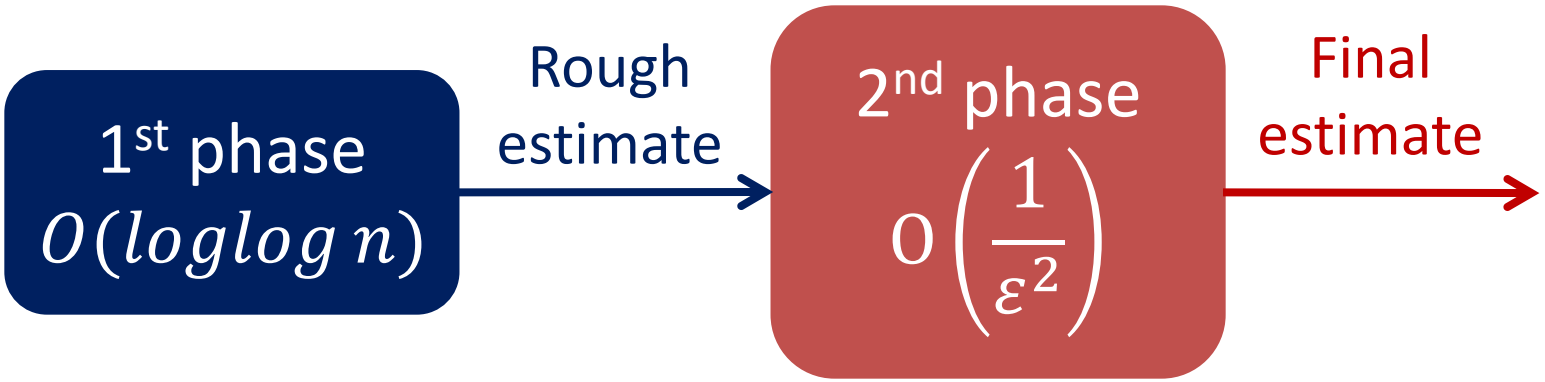
$$\mathrm{o}\left(\log\log n + \frac{1}{\varepsilon^2 log\frac{1}{\varepsilon}}\right)$$

- Our proof leverages GHD (Gap Hamming Distance) communication complexity lower-bound result in [Chakrabarti & Regev, STOC'11]

B. Chen, Z. Zhou, and H. Yu, "Understanding RFID Counting Protocols," Mobicom'13.

Our previous work (Mobicom'13)

Based on our lower bound, we find the key to design RFID counting protocol is to have two phases:



Following our observation, we designed a simple RFID counting protocol SRC that outperforms existing protocols

Question for this work

How to count when there are communication errors and the error rate varies with time?

- Most protocols assume the channel is free-of-error
- Or the error rate is a constant (e.g., ZOE [Infocom'13])

No existing protocol works with time-varying error rate !

- For example, with bursty errors, the actual estimation errors of all existing protocols we evaluated exceed 20 times of their claimed error bound

Our contribution --- Robust RFID Counting protocol (RRC)

- Robust against time-varying error rates: No assumption on how communication errors distribute over time

- Asymptotically near-optimal efficiency: We prove that the expected amount of time needed by RRC is $O(Y + \frac{1}{\varepsilon^2} + (loglogn)^2)$, where Y is the # of communication errors experienced by RRC

- Guarantees on estimation quality: We prove that the output $\hat{n}$ generated (eventually) by RRC, despite communication errors, is an (ε, δ) estimation of n

Abstraction of a time-varying channel

- Intuitively, one can visualize a time-varying channel by an infinite long tape with three types of symbols: $[\![1]\!]$, $[\![0]\!]$, and $[\![\_]\!]$
  - The reader sees a non-empty (or "1") slot with $[\![1]\!]$ symbol
  - The reader sees an empty (or "0") slot with $[\![0]\!]$ symbol
  - The slot is error-free when the corresponding symbol is $[\![\_]\!]$
- We make no assumption on how these symbols distribute over the tape
  - One can imagine an adversary who examines the protocol *before its execution* and strategically decides which communication errors to introduce for which slots

Basic Idea I: *Error-catching slots*

- Error-catching slots are special slots that the reader sees pre-determined results (0 or 1) when no error
  - Reader asks all tags to reply/not-reply when it expects 1/0
  - The error-catching slots are mixed randomly with the information slots, thus the error-catching slots will see their "fair share" of the errors
  - The reader uses them to estimate the error rate

R-Trial (p, l): 4l/5 slots are information slots, where each tag responds with probability p. The other l/5 slots are error-catching slots. The two types of slots are randomly shuffled

Basic Idea II: *Converging Retries*

- If we observe errors in a large fraction of error-catching slots, the information slots may not contain sufficient amount of useful information
  - Hence, RRC needs to test more slots through retries
- Retries result in an increasing false negative rate
  - False negative: when the actual # of erroneous slots is excessive, the protocol incorrectly believes # of erroneous slots is small enough to produce an output

Basic Idea II: *Converging Retries*

- Idea: When a protocol intends to make a judgement, it uses all information available, (i.e., the current trial and all previous trials)
  - combing all the x trials the protocol sees so far would result in a false negative rate of r^x
  - Hence, regardless of how many trials are needed, the overall false negative rate will be bounded: $r+r^2+r^3+...+r^x < r/1-r$

Basic Idea II: *Converging Retries*

Algorithm 1 CR (Converging Retries)

// invoked with p, ϵ, and a global variable K

1: determine l according to ϵ;
2: invoke R-Trial(p, l) for K times;
3: **while** true **do**
4: examine all slots this CR incurred so far;
5: **if** error fraction in error-catching slots $> \frac{1}{8}$ **then**
6: invoke one more R-Trial(p, l);
7: **else**
8: **return** an estimate of n based on all slots;
9: **end if**
10: **end while**
11: update global variable $K \leftarrow K + 1$;

Basic Idea III: *Use CR in SRC*

Algorithm 2 RRC (Robust RFID Counting) Protocol

// First phase:
1: initialize $K \leftarrow 1$;
2: $i \leftarrow 1$;
3: **while** value of $\tilde{n}$ unassigned **do**
4: $p \leftarrow (\frac{2}{3})^i$;
5: $x \leftarrow \text{CR}(p, \frac{1}{3})$;
6: **if** $(1-p)^x < 0.45$ **then**
7: $i \leftarrow i \times 2$;
8: **else if** $(1-p)^x \in [0.45, 0.55]$ **then**
9: $\tilde{n} \leftarrow x$;
10: **else**
11: $\tilde{n} \leftarrow \text{Binary-Search}(\frac{i}{2}, i)$;
12: **end if**
13: **end while**
// Second phase:
14: reset $K \leftarrow 1$;
15: $\hat{n} \leftarrow \text{CR}(0.69/\tilde{n}, \epsilon)$;
16: **return** $\hat{n}$;

Algorithm 3 Binary-Search(low, $high$).

1: **while** true **do**
2: $m \leftarrow \frac{low+high}{2}$;
3: $x \leftarrow \text{CR}((\frac{2}{3})^m, \frac{1}{3})$;
4: **if** $(1-(\frac{2}{3})^m)^x \in [0.45, 0.55] \vee low \geq high$ **then**
5: **return** x;
6: **else if** $(1-(\frac{2}{3})^m)^x < 0.45$ **then**
7: $low \leftarrow m+1$;
8: **else**
9: $high \leftarrow m-1$;
10: **end if**
11: **end while**

RRC's formal guarantee

Theorem 4. *For $\epsilon < 0.25$, RRC outputs an $(\epsilon, \frac{1}{3})$ estimate of n and incurs $O(Y + \frac{1}{\epsilon^2} + (\log\log n)^2)$ overhead, where Y is the number of erroneous slots experienced by RRC during its execution.*

Recall our lower bound results for error-free RFID counting:

$$\mathrm{o}\left(\frac{1}{\varepsilon^2 log\frac{1}{\varepsilon}} + \log\log n\right)$$

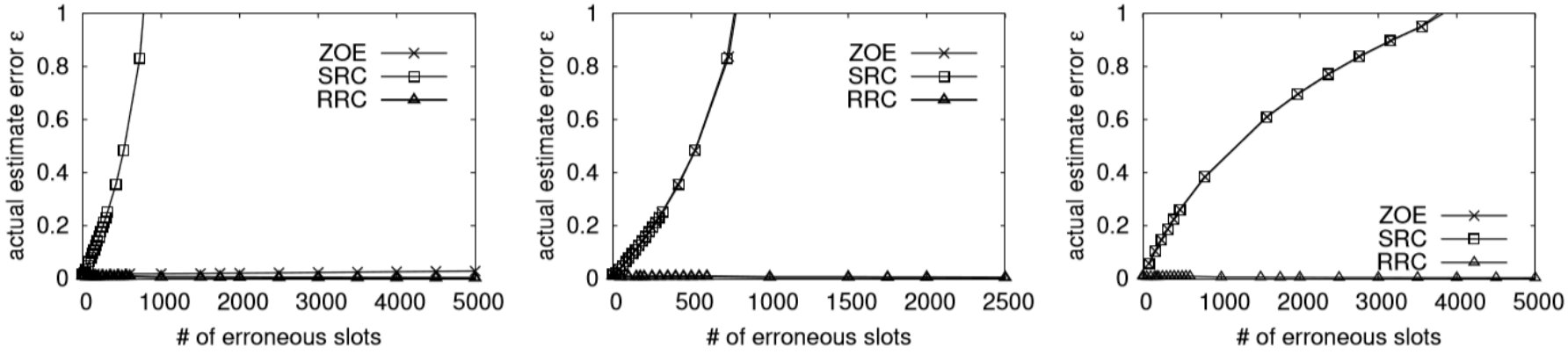
RRC is asymptotically near-optimal

Numerical evaluation

- We compare RRC with two baselines
  - SRC [Mobicom'13]: no consideration of error at all
  - ZOE [Infocom'13]: assuming time-invariant error rate over execution (*we gave ZOE 100% extra free slots to estimate error rate before it runs*)
- The simulation is conducted according to parameters from EPCglobal C1G2 standard
  - Consider $n = 50000$ with an $\left(0.03, \frac{1}{3}\right)$ estimation quality requirement
  - Simulate a variety of error settings

Estimation quality

	Channel error type		Distribution pattern
Setting 1	100%	**1**	Uniformly distributed
Setting 2	100%	**1**	Bursty
Setting 3	100%	**0**	Bursty



(a) Setting 1. (b) Setting 2. (c) Setting 3.

Fig. 1: Actual estimation error ϵ obtained by different protocols under different communication error settings.

Overhead

- The overhead of RRC increases linearly with the # of communication errors it encounters
  - Aligns with our analysis that RRC incurs overhead of $O(Y + \frac{1}{\varepsilon^2} + (loglog\ n)^2)$
- When the channel is error-free, RRC takes ~2s, compared to ~1.5s of SRC
  - The difference mainly comes from the constant before $\frac{1}{\varepsilon^2}$, as the $\frac{1}{\varepsilon^2}$ term dominates the $(loglog\ n)^2$ term

Conclusion

- RRC: a Robust RFID counting protocol
  - Robust against time-varying error rates
  - Asymptotically near-optimal efficiency
  - Guarantees on estimation quality

- RRC achieves so while remaining simple
  - By replacing a non-robust building block of SRC protocol with the robust *Converging Retries (CR)*