Reliable Delegation with Streaming Interactive Proofs

Justin Thaler, Harvard University
Joint work with:
Amit Chakrabarti (Dartmouth),
Graham Cormode (AT&T Labs – Research),
Andrew McGregor (UMass Amherst),
Michael Mitzenmacher (Harvard),
and Ke Yi (HKUST)
Outsourcing

- Many applications require outsourcing computation to untrusted service providers.
  - Main motivation: Commercial cloud computing services.
  - Also, weak peripheral devices; fast but faulty co-processors.
  - Volunteer Computing (SETI@home, World Community Grid, etc.)

- User requires a guarantee that the service provider performed the computation correctly.

- One solution: require provider to prove correctness of answer.
Interactive Proofs


- Think of $P$ and powerful, $V$ as weak. $P$ solves a problem, tells $V$ the answer.
  - Then $P$ and $V$ have a conversation.
  - $P$’s goal: convince $V$ the answer is correct.

- Requirements:
  - 1. Completeness: An honest $P$ can convince $V$ she’s telling the truth.
  - 2. Soundness: $V$ will catch a lying $P$ with high probability no matter what $P$ says to try to convince $V$. 
Interactive Proofs

- IPs have revolutionized Complexity Theory in the last 25 years.
  - $\text{IP} = \text{PSPACE}$ [Shamir 90].
  - PCP Theorem e.g. [AS 98]. Hardness of approximation.
  - Zero Knowledge Proofs.

- But IPs have had very little impact on real systems.
  - Why?
  - Not due to lack of applications!
Interactive Proofs

- Old Answer: Most results on IPs dealt with hard problems, needed $P$ to be too powerful.
  - But recent constructions focus on “easy” problems (e.g. “Interactive Proofs for Muggles” [GKR 08]).
  - Allow $V$ to run very quickly, use small space, so outsourcing is useful even though problems are “easy”.
  - $P$ does not need much more time to prove correctness than she does to solve the problem in the first place!

- Shouldn’t these results be useful and exciting to practitioners?
This Talk: New Application of IPs

- To streaming problems: hard because $V$ has to read input in one-pass streaming manner, but (might be) easy if $V$ could store the whole input. [CCM 09], [CMT 10], [CTY 10].

- Fits cloud computing well: streaming pass by $V$ can occur while uploading data to cloud.

- $V$ never needs to store entirety of data!
Data Streaming Model

• Stream: \( m \) elements from universe of size \( n \)
  • e.g., \( S = <x_1, x_2, ..., x_m> = 3,5,3,7,5,4,8,7,5,4,8,6,3,2, ... \)

• Goal: Compute a function of stream, e.g., median, number of distinct elements, frequency moments, heavy hitters.

• Challenge:
  (i) Limited working memory, i.e., sublinear\((n,m)\).
  (ii) Sequential access to adversarially ordered data.
  (iii) Process each update quickly.

Slide derived from [McGregor 10]
Graph Streams

- $S = \langle x_1, x_2, \ldots, x_m \rangle; x_i \in [n] \times [n]$
- $S$ defines a graph $G$ on $n$ vertices.
- Goal: compute properties of $G$.
- Challenge: subject to usual streaming constraints.

Snapshot of Internet Graph
Bad News

- Many graph problems are impossible in standard streaming model (require linear space or many passes over data, even to approximate).

- E.g. Counting triangles, diameter, perfect matching, shortest $s$-$t$ path.

- What to do?
This Talk: Models

• Two models:
  1. One message (Non-interactive) [CCM 09]: After both observe stream, P sends V an email with the answer, and a proof attached.
  2. Multiple rounds of interaction [CTY 10]: P and V have a conversation after both observe stream.

• Earlier streaming models for verifiable outsourced computation: [TMDSOJ 05], [LYHK 07], [YLHKS 08], [DLN 09], etc.
Costs in Our Models

- Two main costs: words of communication $h$ and $V$’s working memory $v$.
  - We refer to $(h, v)$-protocols.
- Other costs: running time, number of messages.
Comparison of Two Models

- Pros of multi-round model:
  1. Exponentially reduces space and communication cost. Often (polylog n, polylog n) compared to (√n, √n).
  2. $P$ often much faster than in single-round case.

- Cons of multi-round model:
  1. $P$ must do significant computation after each message. Requires maintaining state between messages; possibly overhead in setting up each computation.
  2. More coordination needed; network latency might be an issue.

- Pros of single round model:
  1. Space and communication still reasonable (< 1 MB).
  2. $P$ can do all computation at once, just send an email with proof attached.
Non-interactive Protocols: A Sampling
Self-Join Size

- The Self-Join Size of a stream is defined as follows:
  - Let $\mathbf{X}$ be the frequency vector of the stream
    ($X_i$ is number of occurrences of $i$ in the stream)
  - $F_2(\mathbf{X}) = \sum_i X_i^2$

- [CCM09] ($\sqrt{n}$, $\sqrt{n}$)-protocol for Self-Join Size.

- This is optimal. There is a lower bound that says for $(h, v)$-protocol for $F_2$, $hv = \Omega(n)$ lower bound.

- Notice $(1, n)$ and $(n, 1)$ protocols are trivial. What is non-obvious is how to trade off between $h$ and $v$. 
Self-Join Size Protocol

- [CCM09] ($\sqrt{n}, \sqrt{n}$)-protocol for Self-Join Size.
- Recall: $F_2(X) = \sum_i X_i^2$
- View universe $[n]$ as $[\sqrt{n}] \times [\sqrt{n}]$.

Slide derived from [McGregor 10]
• First idea: Have $P$ send the answer “in pieces”:
  • $F_2(\text{row 1}), F_2(\text{row 2}).$ And so on. Requires $\sqrt{n}$ communication.

• $V$ exactly tracks a row at random (denoted in yellow) so if $P$ lies about any piece, $V$ has a chance of catching her. Requires space $\sqrt{n}$.

![Frequency Square X](attachment:image.png)

\[ P \text{ sends} \]
\[
\begin{align*}
20 &= 2^2 + 4^2 \\
18 &= 3^2 + 3^2 \\
4 &= 2^2
\end{align*}
\]

Slide derived from [McGregor 10]
• Problem: If \( P \) lies in only one place, \( V \) has small chance of catching her.

• We would like the following to hold: if \( P \) lies about even one piece, she will have to lie about many.

• Solution: Have \( P \) commit (succinctly) to self-join size of rows of an **error-corrected encoding** of the input.

• Need \( V \) to evaluate any row of the encoding in a streaming fashion. Can do this for “low-degree extension” code. Note: this code is **systematic**, meaning the first \( n \) symbols are just the input itself.
Error-corrected Encoding of Frequency Square $X$

H sends

$20 = 2^2 + 4^2$

$18 = 3^2 + 3^2$

$4 = 2^2$

$26 = (-1)^2 + (-5)^2$

$180 = (-6)^2 + (-12)^2$

$610 = (-13)^2 + (-21)^2$

These values will all lie on low-degree polynomial $s(X)$.

Input is embedded in encoding (low-degree extension).
Formal Protocol

• Pick a finite field $\mathbb{F}_p$. $X$ implies a two-variate polynomial $f(x, y)$ over $\mathbb{F}_p$ such that $f(i,j)=X_{(i,j)}$ for all $(i,j) \in [\sqrt{n}] \times [\sqrt{n}]$.

• Let $s(X)=\sum_{y \in [\sqrt{n}]} f^2(X, y)$.

• $V$ picks a random $r \in \mathbb{F}_p$ and computes $s(r)=\sum_{y \in [\sqrt{n}]} f^2(r, y)$ while observing stream (takes $\sqrt{n}$ space).

• $P$ sends a univariate polynomial $g(X)$ claimed to be $s(X)$. Note $g$ has degree $2(\sqrt{n}-1)$ and so can be represented in $2\sqrt{n}$ words.

• $V$ checks that $g(r)=s(r)$. If so, $V$ outputs $\sum_{x \in [\sqrt{n}]} s(x)$. If $P$ lies (i.e. $g(X) \neq s(X)$), then w.h.p. $s(r) \neq g(r)$ and thus $P$ will be caught.
A general technique

- Arithmetization: Given function $f'$ defined on a small domain, extend domain of $f'$ to a large field and replace $f'$ with its low-degree extension (LDE) $f$ as a polynomial over the field.

- Can view $f$ as an error-corrected encoding of $f'$. The error correcting properties of $f$ give $V$ considerable power over $P$.

- If two (boolean) functions differ in one location, their LDE’s will differ in almost all locations.
Why study $F_2$?

Optimal Bipartite Perfect Matching Protocol

Optimal Matrix-Vector Multiplication Protocol

Optimal $F_2$ Protocol

Optimal Subset Protocol

Omitted:
Connectivity,
Counting Triangles,
Shortest $s-t$ path.
Matrix-Vector Multiplication [CMT10]

- Goal: Given $n \times n$ integer matrix $A$ and vector $\mathbf{x}$, make $P$ provide the vector $A\mathbf{x}$ and proof of correctness.

- We will get optimal $(n^{2-\alpha}, n^\alpha)$ protocol for any $\alpha \in [0,1]$. Lower bound: $hv = \Omega(n^2)$.

- Better yet, when $\alpha = 1$, $P$ only has to provide $A\mathbf{x}$, and no other information is necessary. $V$ gets security guarantee for “free” as long as $V$ has access to $O(n)$ words of memory.
Matrix-Vector Multiplication [CMT10]

- Fact: inner-product protocol follows from F2 protocol.
Matrix-Vector Multiplication [CMT10]

- First idea: Treat as n separate inner-product queries, one for each row of A.
  - Worse than “naïve” solution.
  - Multiplies both $h$ and $v$ by n, as compared to a single inner-product query.
Matrix-Vector Multiplication [CMT10]

- First idea: Treat as n separate inner-product queries, one for each row of A.
  - Worse than “naïve” solution.
  - Multiplies both $h$ and $v$ by n, as compared to a single inner-product query.

- Key observation: one vector, $x$, in each inner-product query is constant.
  - Can exploit this structure and linear hashing techniques to only multiply $h$ by n.
  - $v$ will be the same as for a single inner product query.
Subset Protocol [CCMT11]

• Goal: Given a list of elements in set $Y$ followed by a list of elements in set $X$, determine if $X \subseteq Y$.

• Result: $(n^\alpha, n^{1-\alpha})$ protocol for any $\alpha \in [0,1]$. This is optimal.

• Solution: Represent $X$ and $Y$ by indicator vectors $x$ and $y$. Then $X \subseteq Y$ if and only if $F_2(y-x) = F_2(y)-F_2(x)$. So we can just run three copies of the $F_2$ protocol.
Bipartite Perfect Matching [CCMT11]

- Goal: Given a list of edges \( e_1 \ldots e_m \) defining a bipartite graph \( G=(V, E) \), determine whether \( G \) contains a perfect matching (a set \( M \subseteq E \) such that each vertex appears in exactly one edge in \( M \)).

- Note: If there is a perfect matching, \( P \) can prove this by providing a valid matching (and proving validity). If there is no valid matching, \( P \) must give a witness to this fact.

- Result: \( (n^{2-\alpha}, n^\alpha) \) protocol for any \( \alpha \in [0,1] \). This is optimal.
Bipartite Perfect Matching [CCMT11]

- Solution: If $G=(V, E)$ contains a perfect matching $M$, $P$ lists the edges in $M$. Requires communication $O(n)$.

- $V$ must check the matching is feasible! That is,
  
  1. $M$ is a matching (i.e. each node appears in exactly one edge in $M$). Can be done with standard hashing techniques.
  2. $M \subseteq E$ (i.e. $P$ isn’t making up edges). Use the Subset protocol.
Bipartite Perfect Matching [CCMT11]

- Solution: If $G=\langle V, E \rangle$ does not contain a perfect matching, Hall’s Matching Theorem provides a witness to this fact.

1. The witness is a subset $S \subseteq V$ that satisfies a certain property (specifically, $S$ has a small neighborhood). Requires communication $O(n)$.

2. $V$ can check witness validity using matrix-vector multiplication protocol.

3. The matrix $A$ is $G$’s adjacency matrix, and the vector $x$ is the indicator vector of $S$.

4. $V$ can extract the size of $S$’s neighborhood from $Ax$. 
Bipartite Perfect Matching [CCMT11]

- Hall’s Matching Theorem: $G=(V, E)$ does not contain a perfect matching if and only if there is a subset $S$ of the nodes in the left partite set such that $|N(S)| < |S|$, where $N(S)$ is neighborhood of $S$.

1. $|N(S)|$ equals the Hamming Norm of $Ax$, where $A$ is adjacency matrix of $G$ and $x$ is indicator vector of $S$.

2. If $P$ sends $S$, $V$ can check $|N(S)| < |S|$ using matrix-vector multiplication protocol applied to $Ax$. 
Interactive Protocols: A Sampling
Interactive Model: A General Result

- Powerful constructions from IP literature can work with streaming verifier! Includes “Interactive Proofs for Muggles” [GKR08] and a construction of Kilian [K92].

- Therefore: (polylog \(n\), polylog \(n\)) computationally sound protocols for NP. Efficient protocols even for problems hard in non-streaming setting (i.e. for NP-complete problems) if we are willing to settle for computational soundness.

- (polylog \(n\), polylog \(n\)) statistically sound protocols for all of log-space uniform NC (includes e.g. matrix problems like determinant, and graph problems like MST, shortest paths).
How to Make V Streaming

- Three observations:
  1. In many proof systems, \( V \) only accesses the input in order to compute \( f(\mathbf{r}) \) for a randomly chosen \( \mathbf{r} \), where \( f \) is LDE of input.
  2. Moreover, \( \mathbf{r} \) does not depend on \( P \)’s messages.
  3. \( V \) can evaluate \( f(\mathbf{r}) \) in streaming fashion.

- So streaming \( V \) chooses \( \mathbf{r} \) in advance; remembers it and keeps it private from \( P \); and computes \( f(\mathbf{r}) \) during “input observation” phase.
Some comments

- Despite powerful generality, [GKR 08] is not optimal for many low-complexity functions of high interest in streaming and database processing.

- [CTY10] give improved protocols for these problems.
  - And argues that they are practical.
**F₂ protocol**

- Result: (log \( n \), log \( n \))-protocol requiring log \( n \) rounds. [GKR 08] yields (log² \( n \), log² \( n \)) protocol requiring log² \( n \) rounds.

- Moreover, can make \( P \) run in linear time.

- In our implementation, \( P \) handles \( \sim \)21 million updates/second. More experimental results later.
Extensions to *Frequency-Based Functions* of the form $\sum_i g(a_i)$ for arbitrary $g$. Includes DISTINCT, Higher Frequency Moments, $F_{\text{max}}$, etc.

Heavy hitters.

**Reporting queries.** Such as:

1. **INDEX:** Given $i$, determine $a_i$.
2. **RANGE-SUM:** Given $[l, u]$, compute $\sum_{i=l}^{u} a_i$
3. **RANGE QUERY:** Given $[l, u]$ determine the frequency of all items between $l$ and $u$, inclusive.
Experimental Results: A Sampling
Two-Pronged Approach [CMT11]

- Ideal: General purpose implementation allowing to verify arbitrary computation.
  - Implemented general-purpose “Interactive Proofs for Muggles” construction [GKR08]. Encouraging results!
  - Plus extensions to help move it from theory to practice.

- For many problems [GKR08] remains slow. So we implemented fine-tuned protocols for specific problems.
  - Even if general-purpose protocols improve, fine-tuned protocols should remain valuable in real-world settings.
First Prong: General Purpose Construction
“Muggles” Implementation [CMT11]

- In “Muggles”, $P$ and $V$ first agree on an arithmetic circuit $C$ computing the function of interest. Then $P$ gives $V$ the output of $C$ and proves that the output is correct.
  - We engineered $P$ to run in time near-linear in size of $C$.
  - Also extended protocol to allow for more general kinds of gates in $C$. 
“Muggles” Implementation [CMT11]

- In “Muggles”, P and V first agree on an arithmetic circuit C computing the function of interest. Then P gives V the output of C and proves that the output is correct.
- We engineered P to run in time near-linear in size of C.
- Also extended protocol to allow for more general kinds of gates in C.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Version</th>
<th>Size (gates)</th>
<th>$P$ time (s)</th>
<th>Messages</th>
<th>Comm (words)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELF-JOIN</td>
<td>basic</td>
<td>393,215</td>
<td>8.48</td>
<td>986</td>
<td>1479</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>basic</td>
<td>15,990,723</td>
<td>552.59</td>
<td>3730</td>
<td>11190</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>8 gates</td>
<td>8,388,566</td>
<td>462.21</td>
<td>1684</td>
<td>7691</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>16 gates</td>
<td>6,422,496</td>
<td>457.37</td>
<td>1399</td>
<td>8427</td>
</tr>
</tbody>
</table>

Experimental Results with general-purpose implementation, when run on streams with universe size $2^{17}=131,072$. 
Second Prong: Specialized Protocols
F₂ Experiments

- Implemented ($\sqrt{n}$, $\sqrt{n}$) one-round F₂ protocol from [CCM 09] and multi-round F₂ protocol from [CTY 10].

  - Single-round space and communication costs still under a megabyte for n=10 billion.
  - Multi-round space and communication always under 1 KB even when handling GBs of data.

  - V takes about the same time in both cases (38 million updates per second in single-round case, 20 million in multi-round case, across all stream lengths).
F₂ Experiments

- P much more efficient in multi-round case.
  - Multi-round case: P processes 20 million updates per second across all stream lengths.
  - Single-round case:
    1. Naïve implementation of P requires Ω(n^{3/2}) time; doesn’t scale to large streams.
    2. But by using sophisticated FFT techniques, we achieved an implementation processing 250,000-750,000 updates per second across all stream lengths. Easily scales to billions of updates.
$F_2$ Experiments

Multi-round $P$ vs. Single-round $P$ with and without FFT techniques
Matrix-Vector Multiplication Experiments

- Recall we developed an \((n^{2-\alpha}, n^\alpha)\) protocol.

- Across all stream lengths, \(V\) processed 20 million updates per second (\(\alpha = 1\)) to > 30 million updates per second (smaller \(\alpha\)).

- For \(\alpha = 1\), \(P\) processed 40-50 million updates per second, because \(P\) just had to compute the right answer. \(P\) can be sped up further using less naive matrix-vector multiplication implementations.

- For all \(\alpha < 1\), \(P\) still processed more than 1 million updates per second.
Matrix-Vector Multiplication Experiments

P’s time for single-round \((n^{2-\alpha}, n^\alpha)\) matrix-vector multiplication protocol
Parallelizability

- P’s computation in all our single-round protocols are easily parallelizable.

- Using an 8-core machine, and 3 OpenMP statements, achieved close to 8-fold speedup.

- $F_2$ and Matrix-Vector Multiplication also possess simple two-round MapReduce protocols (have not implemented).
Parallelizability

Single-round F_2 protocol on single core, n=225 million (uncontrolled environment)

<table>
<thead>
<tr>
<th>CPU time</th>
<th>Real time</th>
<th>Updates/s in Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>558.0 s</td>
<td>568.9 s</td>
<td>396,919</td>
</tr>
</tbody>
</table>

Single-round F_2 protocol on seven cores, n=225 million (uncontrolled environment)

<table>
<thead>
<tr>
<th>CPU time</th>
<th>Real time</th>
<th>Updates/s in Real Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>570.7 s</td>
<td>91.0 s</td>
<td>2.47 million</td>
</tr>
</tbody>
</table>
Open Questions

- Non-interactive protocols:
  1. Proving any specific problem requires $\max(h, v) > \sqrt{n}$ requires novel communication complexity techniques.
  2. Reusability (for either interactive or non-interactive protocols)?

- Interactive protocols:
  1. Still lacking fully practical protocols for specific problems of high interest (DISTINCT, $F_{\text{max}}$) because $P$ takes too long in existing constructions.
  2. Is a practical general-purpose construction possible? We seem closer than previously realized.
  3. Can parallelism make [GKR08] fully practical? Implement on GPU.
Thank you!
Non-interactive Protocol for Shortest s-t Path

• Goal: Given two nodes $s$ and $t$ in a weighted directed graph, compute the length of the shortest path between them.

• We achieve a $(dh, v)$ protocol for any $hv=n^2$, $n \leq h$, where $d=\max_{v \text{ reachable from } s} d(s, v)$.

• This is optimal for graphs with small diameter! Even in the unweighted, undirected case.
A Tool We Need

- Recall our non-interactive protocol for $F_2$ that allowed us to compute $\sum_i X_i^2$ for any vector $X$.

- A generalization allows us to compute $\sum_i g(X_i)$ for any vector $X$ and any low-degree polynomial $g$. Call this the polynomial agreement protocol.

- Cost of this protocol depends on degree of $g$. 
The Protocol for Shortest s-t Path

- Primal LP for shortest s-t path:
  \[
  \text{Minimize } \sum_{ij} w_{ij} x_{ij} \text{ subject to }
  \sum_j x_{ij} - \sum_j x_{ji} = 0 \text{ if } i \neq s, t
  \]
  \[
  \sum_j x_{sj} - \sum_j x_{js} = 1
  \]
  \[
  \sum_j x_{tj} - \sum_j x_{jt} = -1
  \]
  \[
  0 \leq x
  \]

- Dual LP:
  \[
  \text{maximize } y_t - y_s \text{ subject to: for all } (i,j) \in E, y_j - y_i \leq w_{ij}.
  \]

- By total unimodularity, both primal and dual LPs have integral optima.
The Protocol for Shortest s-t Path

• First step: P provides a (claimed) primal-optimal solution i.e. a path between s and t claimed to be the shortest.
  • Requires communication $O(n)$.

• V must check that the path is feasible i.e. it is indeed a path, and P isn’t making up edges or lying about their weight.

• Can do this using a generalization of the Subset Protocol.

• For this part of the protocol, we can achieve communication $O(h)$ and space $O(v)$ for any $hv=n^2$ and $n \leq h$. 
The Protocol for Shortest s-t Path

• Recall Dual LP:
  
  maximize $y_t - y_s$ subject to: for all $(i,j)$ in $E$, $y_j - y_i \leq w_{ij}$.

• Second Step: $P$ proves optimality of the primal solution by giving a dual solution $y$ of the same value.
  
  • There are only $n$ dual variables so specifying a dual solution takes $O(n)$ communication.

• $V$ must check $y$ is dual feasible.
The Protocol for Shortest s-t Path

• Recall Dual LP:
  
  \[
  \text{maximize } y_t - y_s \text{ subject to for all } (i,j) \text{ in } E, \ y_j - y_i \leq w_{ij}.
  \]

• Let Z be the length-n^2 vector given by Z_{ij}=w_{ij}+y_i-y_j for all edges (i,j) and Z_{ij}=0 otherwise.

• Let d be an upper bound on |y|_\infty, the largest absolute value of any variable in the dual solution. Then all entries of Z are between -2d and |w|_\infty + 2d.

• y is dual-feasible if and only if \( \sum_i g(Z_i) = 0 \), where g is a polynomial such that \( g(x)=0 \) for \( 0 \leq x \leq |w|_\infty + 2d \). g can have low degree if d and |w|_\infty are small.

Can run polynomial agreement protocol to check dual feasibility!
The Protocol for Shortest s-t Path

- Details brushed under the rug:
  
  - Need to show V can *derive* a stream defining the vector Z from the actual observed graph stream.

  - Need to prove that there always exists a dual-optimal solution $y$ with $|y|_\infty \leq \max_{v \text{ reachable from } s} d(s, v)$.

- Final result: $(dh, v)$ protocol for any $hv=n^2$, $n\leq h$, where $d=\max_{v \text{ reachable from } s} d(s, v)$.
Extension to Frequency-Based Functions

- Frequency based function $F(a)$ is of the form $F(a) = \sum_{i} g(a_i)$ for some $g : N_0 \rightarrow N_0$.

- e.g. $F_k$, $F_0$ (DISTINCT), “How many items have frequency at most $i$?”, verifying $F_{\text{max}}$ (highest-frequency).
Extension to Frequency-Based Functions

- First idea: extend \( g \) to a polynomial \( g \) over \( \mathbb{F}_p \) and apply a sum-check protocol to the polynomial \( g \circ f \).
  - Streaming \( V \) can evaluate \( g \circ f(r) \) by computing \( f(r) \) and then \( g(f(r)) \).
  - Problem: \( g \) might have degree \( n \). Resulting communication cost is \( dn \), worse than trivial protocol.
Extension to Frequency-Based Functions

- First idea: extend $g$ to a polynomial $g$ over $\mathbb{F}_p$ and apply a sum-check protocol to the polynomial $g \circ f$.
  - Streaming $V$ can evaluate $g \circ f(r)$ by computing $f(r)$ and then $g(f(r))$.
  - Problem: $g$ might have degree $n$. Resulting communication cost is $dn$, worse than trivial protocol.

- Solution: We give a $(1/ \phi \log u, 1/ \phi \log u)$ protocol to identify all items of frequency at least $\phi m$ (the “$\phi$ -heavy hitters”). Use this protocol to “remove” the heavy items, which allows to control degree of $g$. 
Extension to Frequency-Based Functions

- Result: a ($\sqrt{u \log u}$, $\log u$)-protocol for any frequency-based function that takes $\log u$ rounds.

- [GKR 08] yields $(\log^2 u, \log^2 u)$ protocol.

- For 1 TB of data, $\sqrt{u}$ is on the order of 1 MB, $\log^2 u$ is on the order of thousands, \(\log u \approx 40\).

- Might prefer to communicate 1 MB of data over 40 rounds than 1 KB over thousands of rounds due to network latency.
Reporting Queries

- Sub-vector query: Given $q_L$ and $q_R$, determine the non-zero entries of $(a_{q_L}, \ldots, a_{q_R})$.

- We give a $(k + \log u, \log u)$-protocol for Sub-vector requiring $\log u$ rounds, where $k$ is number of non-zero entries in $(a_{q_L}, \ldots, a_{q_R})$.

- Protocol is reminiscent of Merkle trees, but we achieve statistical soundness.
Lower Bound: $hv = \Omega(n)$

- Suppose protocol “A” has parameters $(h, v)$, error $\delta = 1/3$. 

Slide derived from [McGregor 10]
Lower Bound: $hv = \Omega(n)$

- Suppose protocol “A” has parameters $(h, v)$, error $\delta = 1/3$.
- Define “B”:

  Protocol “B”
  
  1. V runs $t = \Theta(h)$ copies of her part of “A” in parallel
  2. Use H’s annotation $a$ for all copies
  3. Output majority answer if it exists, else reject.

- Protocol “B” has parameters $(h, hv)$, error $(1/3)2^{-h}$ (i.e. each $a$ is “bad” for only $(1/3)2^{-h}$-fraction of V’s possible coin flips).

Slide derived from [McGregor 10]
Lower Bound: $hv = \Omega(n)$

- Suppose protocol “A” has parameters $(h, v)$, error $\delta = 1/3$.
- Define “B”:
  - Protocol “B”
    1. V runs $t = \Theta(h)$ copies of her part of “A” in parallel
    2. Use H’s annotation $a$ for all copies
    3. Output majority answer if it exists, else reject.

- Protocol “B” has parameters $(h, hv)$, error $(1/3)2^{-h}$ (i.e. each $a$ is “bad” for only $(1/3)2^{-h}$ fraction of V’s possible coin flips).
- Define “C”: V ignores annotation, tries all $2^h$ possible annotations and accepts if *any one* of them causes “B” to accept.
  - Ensures error at most 1/3, no helper needed.
- Algorithm “C” solves $F_2$ in $hv$ space. But we know $F_2$ requires $\Omega(n)$ space (randomized).

Slide derived from [McGregor 10]