Toward Nearly-Zero-Error Sketching via Compressive Sensing

Qun Huang, Siyuan Sheng, Xiang Cheng, Yungang Bao, Rui Zhang, Yanwei Xu, Gong Zhang
Flow-level Network Monitoring

Data Plane: Update

Packet: (flowkey, values)

Control Plane: Query

Flow Statistics

<table>
<thead>
<tr>
<th>Flow 1</th>
<th>Pkt count</th>
<th>.......</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow 2</td>
<td>Pkt count</td>
<td>.......</td>
</tr>
<tr>
<td>Flow 3</td>
<td>Pkt count</td>
<td>.......</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>.......</td>
</tr>
</tbody>
</table>
Flow-level Network Monitoring

Data Plane: Update
Packet: (flowkey, values)

Control Plane: Query
Limited resources

Flow Statistics
Flow 1  Pkt count  ........
Flow 2  Pkt count  ........
Flow 3  Pkt count  ........
...

Estimated results with error bounds

Approximate techniques are widely used
Existing Guarantees Are Not Enough

- Theoretical bounds only apply to specific flows
  - E.g., heavy hitters, super-spreaders, ...

- For most flows, the bounds are too loose
Example

Count-Min Sketch to monitor byte count

Configuration

10GB network traffic
10^4 counters per-row
>10 rows

Error bound: Per-flow error <210KB

<2% relative error for large flows (>10MB) 😊

Unacceptable for small flows 😞

Small flows matter:
E.g., single-packet TCP flows imply various anomalies
Our Contributions

**New algorithms that achieve nearly-zero-error monitoring**

Nearly-zero error: for almost all (>99%) flows, the relative error is small (<0.1%)

Sketch + Compressive Sensing
Our Contributions

**New algorithms that achieve nearly-zero-error monitoring**

*Nearly-zero error: for almost all (>99%) flows, the relative error is small (<0.1%)*

1. Advantages and limitations of compressive sensing for network monitoring
2. Efficiency of combining sketch and compressive sensing
3. A theoretical framework to revisit common approaches in sketch design
4. Two new sketch algorithms that efficiently embrace compressive sensing
Compressive Sensing

**Sensing Operation**

\[ \Phi \times x = y \]

- **Sensing matrix** \((m \times n)\)
- **Original signal** \((n \times 1\) vector\)

**Recovery Operation**

- **Measurement signal** \((m \times 1\) vector\)
- **Solve an optimization problem**
- **Recovered signal** \((m \times 1\) vector\)

**Update Procedure**

- **Flow statistics**
- **Approximate Data Structure**

**Estimated flow statistics**
Compressive Sensing

Sensing Operation

\[ \Phi \times x = y \]

- Sensing matrix \((m \times n)\)
- Original signal \((n \times 1\) vector)
- Measurement signal \((m \times 1\) vector)

Recovery Operation

- Solve an optimization problem
- Recovered signal \((m \times 1\) vector)

Classical Compressive Sensing:
Guarantee accuracy with recommended \(\Phi\) and optimization algorithm
1st Attempt: Directly Adopt

- 4 types of sensing matrix:
  - Bernoulli Matrix (BM), Fourier Matrix (FM), Gaussian Matrix (GM), and Incoherence matrix (IM)

- 2 recovery algorithms:
  - L1 norm minimization (L1)
  - Orthogonal Matching Pursuit (OMP)

**Advantage:** limited memory to achieve nearly-zero error (High accuracy)

**Limitation:** each packet incurs a large number of updates (Poor scalability)
2nd Attempt: Formulate Sketch

Solve an optimization problem

\[ \Phi \times x = y \]
2nd Attempt: Formulate Sketch

\[ \Phi \times \text{Pkt} = y \]

\[ x_0 \]
\[ x_1 \]
\[ x_2 \]
\[ x_3 \]

\( y_0 \rightarrow y_1 \rightarrow y_2 \)
\( y_3 \rightarrow y_4 \rightarrow y_5 \)

Solve an optimization problem
2nd Attempt: Formulate Sketch

\[
\Phi \times \text{Pkt} \rightarrow \text{Solve an optimization problem} \rightarrow x'
\]
2\textsuperscript{nd} Attempt: Formulate Sketch

\begin{align*}
\begin{bmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix} \times 
\begin{bmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
k=0 \\
k=1 \\
k=2 \\
k=3
\end{bmatrix} 
&= 
\begin{bmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5
\end{bmatrix} \\
\rightarrow \\
\text{Solve an optimization problem} \\
\rightarrow \\
\begin{bmatrix}
x' \\
\end{bmatrix}
\end{align*}

However, the recovery accuracy is low (see paper)
Key Matrix Property

Recommended Matrix \times x = y 

Sketch-derived Matrix \times x = y

Solve an optimization problem

High Accuracy

Low Accuracy
Key Matrix Property

Recommended Matrix

Sketch-derived Matrix

\[ x \times y = y \rightarrow y \xrightarrow{\text{Solve an optimization problem}} x' \]

High Accuracy

Low Accuracy
Key Matrix Property

Study the difference between the two types of matrices

Recommended Matrix

Sketch-derived Matrix

High Accuracy

Low Accuracy

Solve an optimization problem
Key Matrix Property: Orthonormality

- **Orthonormality**: ability to preserve the norm of a sparse vector
  - High orthonormality: $x$ can be preserved and accurately recovered

- **RIP value**: quantify orthonormality
  - The **lower RIP**, the **higher orthonormality**

![Graph showing RIP values for Classical and Sketch-derived Matrices]
Key Matrix Property: Orthonormality

Recommended Matrix

Sketch-derived Matrix

How to construct sketch with high Orthonormality?

High Accuracy

Low Accuracy
Revisit Sketch Design

How common approaches affect matrix?

Class 1: Fractional Update
Methods: Sampling, or Conservative update
Examples: CU Sketch [SIGCOMM’ 02] NitroSketch [SIGCOMM’ 19]
Matrix property: Fractional elements in matrix

Class 2: Adding Rows
Methods: Maintain multiple simple sketch structures
Matrix property: More rows in the matrix

Class 3: Clearing Columns
Methods: Store flowkeys separately
Examples: FlowRadar [NSDI’ 16] UnivMon [SIGCOMM’ 16]
Matrix property: Clearing useless columns

Class 4: Matrix Decomposition
Methods: Separate traffic into two parts, or extract large flows
Examples: SketchLearn [SIGCOMM’ 18] Elastic Sketch [SIGCOMM’ 18]
Matrix property: Decomposing simpler matrices
New Algorithms

- Existing algorithms are not enough
  - Not specifically designed for compressive sensing
  - Use the common approaches, but not efficiently combine them
  - Orthonormality is not the main goal

- Need new algorithms
  - Combine the approaches more efficiently
New Algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>CU Sketch [25]</td>
<td>Conservative update</td>
<td>Multiple CM instances</td>
<td>Flow extraction</td>
<td></td>
</tr>
<tr>
<td>Deltoid [19]</td>
<td></td>
<td>Multiple Bloom Filters</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ElasticSketch [79]</td>
<td></td>
<td></td>
<td>Traffic splitting</td>
<td></td>
</tr>
<tr>
<td>FlowRadar [48]</td>
<td></td>
<td>Bloom Filter</td>
<td>Flow extraction</td>
<td></td>
</tr>
<tr>
<td>NitroSketch [52]</td>
<td>Sampling</td>
<td>Multiple CS instances</td>
<td>Heap</td>
<td></td>
</tr>
<tr>
<td>RevSketch [67]</td>
<td></td>
<td></td>
<td>Flow extraction</td>
<td></td>
</tr>
<tr>
<td>SeqHash [8]</td>
<td></td>
<td>Multiple CM instances</td>
<td>Flow extraction</td>
<td></td>
</tr>
<tr>
<td>SketchLearn [37]</td>
<td></td>
<td>Multiple CM instances</td>
<td>Flow extraction</td>
<td></td>
</tr>
<tr>
<td>SketchVisor [35]</td>
<td></td>
<td></td>
<td>Traffic splitting</td>
<td></td>
</tr>
<tr>
<td>UnivMon [53]</td>
<td></td>
<td>Multiple CS instances</td>
<td>Heap</td>
<td></td>
</tr>
<tr>
<td>SeqSketch</td>
<td>Fractional update</td>
<td>Bloom Filter + Controller</td>
<td>Splitting + Controller</td>
<td></td>
</tr>
<tr>
<td>EmbedSketch</td>
<td>Fractional update</td>
<td>Bloom Filter + controller</td>
<td>Extraction + Controller</td>
<td></td>
</tr>
</tbody>
</table>

SeqSketch

EmbedSketch

Fractional Sketch

Hash Table

Bloom Filter

Packet

Controller

Forwarder

Packet

Sketch

Bucket

$f_i \ c_i \ d_i$

$0 \ 1 \ 0 \ 1 \ 0 \ ... \ 1$

$0 \ ... \ 0$

$0 \ ... \ 0$

$0 \ ... \ -0.6c_i$

$V_{ij} \ c_{ij} \ d_{ij} \ f_{ij}$

$0 \ 1 \ ...$
Results

- RIP values <3
- Accuracy
  - 100% precision and 100% recall
  - <0.1% relative error for >99.7% flows
- Robustness under different memory configuration
- Low resource usage
  - Hardware resources
  - Bandwidth
- Recovery time
Conclusion

Problem: nearly-zero-error monitoring

1. Classical compressive sensing achieves high accuracy but has poor scalability
2. Existing sketch algorithms suffer from low accuracy due to the low orthonormality
3. A framework can study the orthonormality of various common approaches
4. New algorithms are needed to combine the approaches more efficiently

Source Code Available: https://github.com/N2-Sys/NZE-Sketch
Thank You