New Lower Bounds in Merlin-Arthur Communication and Graph Streaming Verification

Prantar Ghosh
Department of Computer Science, Georgetown Univeristy, Washington, D.C., USA

Vihan Shah
Department of Computer Science, University of Waterloo, Canada

Abstract

We present novel lower bounds in the *Merlin-Arthur* (MA) communication model and the related *annotated streaming* or stream verification model. The MA communication model extends the classical communication model by introducing an all-powerful but untrusted player, Merlin, who knows the inputs of the usual players, Alice and Bob, and attempts to convince them about the output. We focus on the online MA (OMA) model where Alice and Merlin each send a single message to Bob, who needs to catch Merlin if he is dishonest and announce the correct output otherwise. Most known functions have OMA protocols with total communication significantly smaller than what would be needed without Merlin. In this work, we introduce the notion of *non-trivial-OMA* complexity of a function. This is the minimum total communication required when we restrict ourselves to only *non-trivial* protocols where Alice sends Bob fewer bits than what she would have sent without Merlin. We exhibit the first explicit functions that have this complexity superlinear – even exponential – in their classical one-way complexity: this means the *trivial* protocol, where Merlin communicates nothing and Alice and Bob compute the function on their own, is exponentially better than any non-trivial protocol in terms of total communication. These OMA lower bounds also translate to the annotated streaming model, the MA analogue of single-pass data streaming. We show large separations between the classical streaming complexity and the non-trivial annotated streaming complexity (for the analogous notion in this setting) of fundamental problems such as counting distinct items, as well as of graph problems such as connectivity and k-connectivity in a certain edge update model called the *support graph turnstile* model that we introduce here.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Streaming models

Keywords and phrases Graph Algorithms, Streaming, Communication Complexity, Stream Verification, Merlin-Arthur Communication, Lower Bounds

Digital Object Identifier 10.4230/LIPIcs.ITCS.2024.53

Related Version *Full Version*: `https://arxiv.org/abs/2401.06378`

Funding *Prantar Ghosh*: Supported in part by NSF under award 1918989. Part of this work was done while the author was at DIMACS, Rutgers University, supported in part by a grant (820931) to DIMACS from the Simons Foundation.
Vihan Shah: Part of this work was done while the author was at Rutgers University and was supported in part by an NSF CAREER Grant CCF-2047061.

Acknowledgements We are extremely grateful to Sepehr Assadi for many helpful conversations regarding the project. Prantar Ghosh would also like to thank Amit Chakrabarti and Justin Thaler for insightful discussions. Finally, we thank the anonymous reviewers of ITCS 2024 for their many detailed comments and suggestions that helped with improving the presentation of the paper.

1 Introduction

In the classical two-player communication model, the players Alice and Bob each receive an input unknown to the other player and exchange as few bits as possible to compute a function of the inputs. In their seminal paper on communication complexity classes, Babai, Frankl, and Simon [6] defined the Merlin-Arthur (MA) communication model, where there is also a "super-player" Merlin, who knows the inputs of both Alice and Bob (collectively "Arthur") and hence, also knows the output. However, Merlin is not trusted, and so he provides Arthur a "proof" that should be thought of as a help message. After receiving this message, Alice and Bob communicate between themselves to verify its correctness. It turns out that for most well-studied functions, the total communication needed between the players can be significantly reduced with the help of Merlin. This holds even in the most restrictive version of the MA model, where both Merlin and Alice send just a single message to Bob, from which he must verify the solution. This is known as the *online Merlin-Arthur* (OMA) model. We focus on this setting and prove new lower bounds that also extend to a related stream-verification model called *annotated streaming* [8, 9]. We discuss the motivation and context of our results below.

1.1 Motivation and Context

Online Merlin-Arthur communication. The OMA complexity of a function is defined as the total communication between the players, i.e., the sum of the lengths of Merlin's and Alice's messages, in the optimal protocol that minimizes this sum. In a protocol, we denote Merlin's message length by hcost (shorthand for *help cost*) and Alice's message length by vcost (shorthand for *verification cost*), while the sum of their message lengths is termed tcost (short for *total cost*). Thus, the OMA complexity of a function is the minimum tcost over all possible OMA protocols computing the function. Naturally, a function with high OMA complexity, possibly close to the input size N, is deemed "hard" in the OMA model. Over the years, researchers have wondered what problems are hard in this model [6, 1, 9, 17]. Surprisingly, Aaronson and Wigderson [1] showed that even the "hardest" functions in classical communication complexity, namely the disjointness and inner product functions, have OMA complexity only $O(\sqrt{N}\log N)$. Although the same authors showed (via a simple counting argument) the existence of functions with OMA complexity $\Omega(N)$, exhibiting an *explicit* function even with OMA complexity $\omega(\sqrt{N})$ remains a longstanding open problem.

Another notion of "hard functions" in this setting might be ones that need large hcost in order to reduce the vcost from what Alice would have already needed to send without Merlin (i.e., in the classical one-way communication model). This implies high OMA complexity of these functions *relative* to their classical one-way complexity (which might be much smaller than the input size, rendering them *not* hard in the earlier sense). Again, disjointness or inner product are not hard in this sense either: Aaronson and Wigderson's protocol shows that for these problems, hcost and vcost can be smoothly traded off while keeping the product $\tilde{\Theta}(N)$. This means even a proof of size slightly larger than $\log N$ suffices to reduce Alice's message size to $o(N)$, whereas she would need to communicate $\Omega(N)$ in the classical setting. Further, prior work has shown that for many other hard functions in classical communication complexity, we can achieve the tradeoff $\mathsf{hcost}\cdot\mathsf{vcost} = O(N)$ [9, 7, 12], ruling them out from being hard in the OMA model. Interestingly, there is a complementary general lower bound that says that an OMA protocol solving *any* function f must have $\mathsf{hcost}\cdot\mathsf{vcost} = \Omega(R^{\rightarrow}(f))$ [6], where $R^{\rightarrow}(f)$ is the one-way communication complexity of f.

Observe that this lower bound implies that the OMA complexity $\mathrm{MA}^{\rightarrow}(f)$ of the function f must be at least $\Omega(\sqrt{R^{\rightarrow}(f)})$. Stronger lower bounds are hardly known, with a couple of exceptions.[1]

Prior work has shown stronger lower bounds for the *sparse* version of the fundamental index problem [7], and testing connectivity and bipartiteness of the *XOR* of two graphs [29]. All these functions have OMA complexity linear (or nearly linear) in their classical one-way complexity. This means the *trivial* OMA protocol, where Merlin sends an empty string (or a junk message) while Alice and Bob (optimally) solve the problem on their own, is (nearly) as good as any other protocol for the problem. Further, since the lower bounds imply that an OMA protocol for any of these functions f must have $\mathsf{tcost}\ \tilde{\Omega}(R(f))$, it follows that reducing the vcost to $o(R^{\rightarrow}(f))$ necessitates hcost to be $\tilde{\Omega}(R^{\rightarrow}(f))$. Hence, these problems are hard in the second sense described above. Given this ray of hope on identifying hard OMA problems, we pursue this thread. In this work, we define the notion of *non-trivial* OMA complexity of a function f as follows. Call an OMA protocol computing f as *non-trivial* if its vcost is $o(R^{\rightarrow}(f))$. Then, the non-trivial OMA complexity $\widehat{\mathrm{MA}}^{\rightarrow}(f)$ is the minimum tcost over all non-trivial protocols for f.

Observe that for functions f with standard OMA complexity $\mathrm{MA}^{\rightarrow}(f) = o(R^{\rightarrow}(f))$, we have $\widehat{\mathrm{MA}}^{\rightarrow}(f) = \mathrm{MA}^{\rightarrow}(f)$ (since the protocol minimizing tcost must be non-trivial); whereas if $\mathrm{MA}^{\rightarrow}(f) = \Omega(R^{\rightarrow}(f))$, then hcost dominates vcost in any non-trivial protocol for f, and $\widehat{\mathrm{MA}}^{\rightarrow}(f)$ essentially measures the minimum hcost required to achieve $\mathsf{vcost} = o(R^{\rightarrow}(f))$. Thus, non-trivial OMA complexity formally captures the "relative" notion of hardness discussed above. Notably, in terms of the input size N, the best-known lower bound on $\widehat{\mathrm{MA}}^{\rightarrow}(f)$ is $\tilde{\Omega}(\sqrt{N})$ (since a bound better than $\tilde{\Omega}(\mathrm{MA}^{\rightarrow}(f))$ is unknown, and $\mathrm{MA}^{\rightarrow}(f)$ has a longstanding $\sqrt{N}$-barrier).

It is now natural to ask the following questions.

- *Can we find a single problem with high non-trivial OMA complexity that can be used to prove strong* $\widehat{\mathrm{MA}}^{\rightarrow}$ *lower bounds for multiple problems?* It would then serve as a *canonical* hard function for OMA in this sense.
- *Can we exhibit an explicit function* f *with* $\widehat{\mathrm{MA}}^{\rightarrow}(f)$ *(strongly) superlinear in* $R^{\rightarrow}(f)$*?* This would mean that the trivial OMA protocol is *significantly* better than any non-trivial protocol in terms of total communication.
- *If the answer to the above question is "yes", how large can this gap be – can it be exponential?*
- *Can we show an explicit function* f *with* $\widehat{\mathrm{MA}}^{\rightarrow}(f)$ *(strongly) superlinear in* $\sqrt{N}$ *for input size* N*?* This would mean that while there is a strong $\sqrt{N}$-barrier for $\mathrm{MA}^{\rightarrow}$ complexity [17], the situation is not at all similar for $\widehat{\mathrm{MA}}^{\rightarrow}$.

In this work, we answer all these questions in the affirmative. We discuss these results in detail in Section 1.2.

Annotated Streaming. Next, we consider the analogous stream verification or *annotated streaming* model [8, 9], where a space-restricted Verifier and an all-powerful Prover with unlimited space simultaneously receive a huge data stream. Following the input stream, the Prover (with knowledge of the entire stream) and Verifier (who could only store a summary) invoke a *scheme* where the Prover tries to convince the Verifier about the answer to an underlying problem, similar to the online MA model. The total cost of a scheme is defined

[1] Indeed, these exceptions have $R^{\rightarrow}(f) = o(N)$ and they do not break the $\sqrt{N}$-barrier for OMA complexity.

as the sum of the number of bits communicated by the Prover (hcost) and the number of bits of space used by the Verifier (vcost). The *non-trivial* annotated-streaming complexity of a function f is analogously defined as the minimum total cost over all non-trivial schemes computing f, where a trivial scheme is one that uses as much space (up to polylogarithmic factors) as is needed in classical streaming (without Prover).

Since all known lower bounds in annotated streaming are proven via reduction from problems in OMA communication, our knowledge of lower bounds in the two models are similar. We use our OMA results to prove strong lower bounds on the non-trivial complexity in this model as well. We show that fundamental data streaming problems such as *counting distinct items* have high non-trivial annotated streaming complexity when frequencies can be huge. Further, we show that graph problems such as connectivity and more generally k-connectivity have high non-trivial complexity under certain graph streams that we call *support graph turnstile* (SGT) streams. It might be intuitive that these problems can be hard in this model, even with a Prover. Perhaps surprisingly, we show that in the classical (sans Prover) model, we can solve these problems under SGT streams – featuring as large as exponential edge weights – almost as efficiently as under standard (unweighted) graph streams. We do this by building *strong* ℓ_0-samplers that can handle large frequencies and might be of independent interest. These results set the stage for our lower bounds on non-trivial annotated-streaming complexity: they provide the benchmark space for any non-trivial scheme solving these problems.

Our final set of results give efficient annotated streaming schemes for k-connectivity on (standard) dynamic graph streams. We exploit graph theoretic properties on k-connected graphs to come up with short certificates for proving or disproving k-connectedness. This might be of independent interest in the graph-theoretic literature. Furthermore, these results establish a conceptual separation between classical streaming and annotated streaming: in the former, graph connectivity problems have roughly the same complexity under dynamic and SGT streams, whereas in the latter, they are much harder under SGT streams than under dynamic. We discuss our annotated streaming results in detail in Section 1.2.

Basic Terminology. For the remainder of Section 1, it helps to define some basic terminology for ease of presentation. Later, in Section 2, we describe all notation and terminology in detail. An OMA protocol with hcost $O(h)$ (resp. $\widetilde{O}(h)$) and vcost $O(v)$ (resp. $\widetilde{O}(v)$) is called an (h, v)-OMA-protocol (resp. $[h, v]$-OMA-protocol). Analogously, an annotated streaming protocol is called an (h, v)-scheme or an $[h, v]$-scheme. For computing a function f, a *trivial OMA protocol* is one that has vcost $\Omega(R^{\rightarrow}(f))$, and a *trivial scheme* is one that uses verification space $\tilde{\Omega}(S(f))$, where $S(f)$ is the classical streaming complexity of f (the asymptotically optimal space for computing f in classical streaming). The "Index" communication problem and its variants come up frequently in our discussions. In the standard version IDX_N, Alice has a string $x \in \{0,1\}^N$ and Bob has an index $j \in [N]$, where his goal is to output $x[j]$, the jth bit of x.

1.2 Our Results and Contributions

First, we define the Equals-Index (henceforth, EQ-IDX) problem, which is the basis of our lower bounds. For arbitrary natural numbers p and q, in the $\text{EQ-IDX}_{p,q}$ communication problem, Alice holds strings $x_1, \ldots, x_p$ where each $x_i \in \{0,1\}^q$. Bob holds a string $y \in \{0,1\}^q$ and an index $j \in [p]$. The goal is for Bob to output whether $x_j = y$. This problem can be interpreted as (a boolean version of) the Index problem on large domains: Alice has a p-length string over the domain $\{0, \ldots, 2^q - 1\}$ (instead of just $\{0,1\}$). Bob needs to verify whether the jth index of Alice's string equals his value y.

Our main result on the non-trivial-OMA complexity of EQ-IDX is as follows.

▶ **Theorem 1.** *For any p, q with $p = \Omega(\log q)$, we have $\widehat{\mathrm{MA}}^{\rightarrow}(\textsc{eq-idx}_{p,q}) = \omega(q)$*

Observe that q can be as large as $\exp(\Omega(p))$. We also show that $R^{\rightarrow}(\textsc{eq-idx}_{p,q})$ is only $\Theta(p + \log q)$ (Lemma 29). This immediately implies that the gap between $\widehat{\mathrm{MA}}^{\rightarrow}(f)$ and $R^{\rightarrow}(f)$ (or $\mathrm{MA}^{\rightarrow}(f)$) can be exponential.

▶ **Corollary 2.** *There is an explicit function f with*

$$\widehat{\mathrm{MA}}^{\rightarrow}(f) = \exp(\Omega(\mathrm{MA}^{\rightarrow}(f))) \text{ and } \widehat{\mathrm{MA}}^{\rightarrow}(f) = \exp(\Omega(R^{\rightarrow}(f))).$$

Conceptually, this means that the trivial protocol where Merlin sends nothing and Alice and Bob solve the problem on their own, is exponentially better (in terms of total communication) than any other protocol[2].

Recall that previously we did not know of any function f on input size N with $\widehat{\mathrm{MA}}^{\rightarrow}(f) = \tilde{\omega}(\sqrt{N})$. While exhibiting a function f with (standard) OMA complexity $\mathrm{MA}^{\rightarrow}(f) = \omega(\sqrt{N})$ remains a longstanding open problem, our results show that $\widehat{\mathrm{MA}}^{\rightarrow}$ complexity does not have such a barrier. In fact, Theorem 1 implies that it can be as large as $N/\log N$.

▶ **Corollary 3.** *For any $C \in (\sqrt{N}, N/\log N)$, there is an explicit function f on input size N with $\widehat{\mathrm{MA}}^{\rightarrow}(f) = \omega(C)$.*

Next, we turn to the related *Sparse Index* problem (henceforth SP-IDX). Chakrabarti et al. [7] defined the $\textsc{sp-idx}_{m,N}$ problem as the version of Index where Alice's string is promised to have hamming weight (i.e., number of 1's) at most m. They identified $\textsc{sp-idx}_{m,N}$ for $m = \log N$ as the first problem whose non-trivial-OMA complexity is nearly as large as its one-way complexity.[3] Via reduction from EQ-IDX, we improve upon their lower bound for $\textsc{sp-idx}_{m,N}$. In particular, our improved lower bound implies that SP-IDX with sparsity $O(\log \log N)$ has non-trivial-OMA complexity *exponential* in its classical one-way complexity.

▶ **Corollary 4.** *For $m = \log \log N$, we have $\widehat{\mathrm{MA}}^{\rightarrow}(\textsc{sp-idx}_{m,N}) = \omega(\log N)$, whereas $R^{\rightarrow}(\textsc{sp-idx}_{m,N}) = \mathrm{MA}^{\rightarrow}(\textsc{sp-idx}_{m,N}) = \Theta(\log \log N)$.*

We remark that en route to establishing the above result, we also improve upon [7]'s upper bound on $R^{\rightarrow}(\textsc{sp-idx}_{m,N})$ and settle its complexity in the classical one-way model.

The only other functions (to the best of our knowledge) that prior work has shown to have OMA complexity (nearly) linear in its one-way complexity are the $\textsc{xor-conn}_n$ and $\textsc{xor-bip}_n$ problems [29]: in these problems, Alice and Bob have one graph each on the same vertex set $[n]$ (hence, the input size $N = \Theta(n^2)$), and they need to check connectivity and bipartiteness (respectively) of the graph obtained by XOR-ing their graphs, i.e., the graph induced by the symmetric difference of their edge sets. Thaler [29] showed that each of these functions f have $\mathrm{MA}^{\rightarrow}(f) = \Omega(n) = \tilde{\Omega}(R^{\rightarrow}(f))$, which implies the same about $\widehat{\mathrm{MA}}^{\rightarrow}(f)$. We reduce from EQ-IDX to reproduce this result, thus making a convincing case for EQ-IDX being a canonical problem for establishing high $\widehat{\mathrm{MA}}^{\rightarrow}$ complexity.

▶ **Corollary 5** ([29], paraphrased). *For $f \equiv \textsc{xor-conn}_n$ or $f \equiv \textsc{xor-bip}_n$, we have*

$$\widehat{\mathrm{MA}}^{\rightarrow}(f) = \Omega(R^{\rightarrow}(f)) = \Omega(n).$$

[2] Here, we are discounting clearly-suboptimal protocols where Alice sends $\omega(R^{\rightarrow}(f))$ bits or where Merlin sends a non-empty message despite Alice sending $\Omega(R^{\rightarrow}(f))$ bits. Hence, "any other protocol" essentially means any non-trivial protocol.

[3] It follows implicitly from their result establishing $\mathrm{MA}^{\rightarrow}(\textsc{sp-idx}_{\log N,N}) = \tilde{\Omega}(R^{\rightarrow}(\textsc{sp-idx}_{\log N,N}))$

We now turn to the annotated streaming model. Our OMA results can be used to prove lower bounds on the total cost of any non-trivial scheme for certain problems.

First, consider the fundamental distinct items problem (henceforth, DIST-ITEM$_{N,F}$) on turnstile streams, where frequencies of elements from universe $[N]$ get incremented and decremented, and we are promised that the absolute value of the max-frequency is bounded above by F. At the end of the stream, we need to output the number of elements with non-zero frequency. We show a separation between classical-streaming and non-trivial annotated-streaming complexities for this problem, given as follows.

▶ **Theorem 6.** *There is a setting of F such that* DIST-ITEM$_{N,F}$ *can be solved in $\widetilde{O}(N)$ space in classical streaming, but any non-trivial annotated-streaming scheme for the problem must have total cost $\Omega(N^{\mathrm{polylog(N)}})$.*

Next, we show a similar separation for graph streaming problems. We define a *support graph turnstile* (SGT) stream to be one where an n-node graph is induced by the support of the edge-frequency vector. SGT streams have a parameter α, the maximum possible absolute frequency. For the (undirected) graph connectivity problem, asking to check if all pairs of nodes are reachable from each other, and the k-vertex-connectivity (resp. k-edge-connectivity) problem, where we need to check whether removal of some $k-1$ vertices (resp. edges) disconnects the graph, we show an $\widetilde{O}(n)$ vs $\Omega(n^{\mathrm{polylog(n)}})$ separation between their classical-streaming and non-trivial annotated-streaming complexities under SGT streams.

▶ **Theorem 7.** *Under certain support graph turnstile streams on n-node graphs, connectivity can be solved in $\widetilde{O}(n)$ space by a classical streaming algorithm, whereas any non-trivial annotated streaming scheme for the problem must have total cost $\Omega(n^{\mathrm{polylog(n)}})$.*

▶ **Theorem 8.** *Under certain support graph turnstile streams on n-node graphs, k-vertex-connectivity and k-edge-connectivity can be solved in $\widetilde{O}(kn)$ space by a classical streaming algorithm, whereas any non-trivial annotated streaming scheme for the problem must have total cost $\Omega(n^{\mathrm{polylog(n)}})$.*

We remark that while the classical streaming space bounds of $\widetilde{O}(n)$ for connectivity and $\widetilde{O}(kn)$ for k-connectivity are known for standard dynamic graph streams [4, 5] where edge multiplicities are 0 or 1 throughout the stream, it was not known whether the same can be achieved for the harder SGT streams. We establish these upper bounds by designing ℓ_0-samplers that can handle frequency exponential in n while incurring just polylogarithmic factors in space (Lemma 28). These might be of independent interest.

Next, we design efficient schemes for k-vertex-connectivity and k-edge-connectivity under standard dynamic graphs streams. These schemes have total cost significantly smaller than the lower bound proven for schemes processing SGT streams.

▶ **Theorem 9.** *Under dynamic graph streams on n-node graphs, there exists a $[k\cdot(h+kn), v]$-scheme for k-vertex-connectivity for any h, v such that $h \cdot v = n^2$. In particular, under such streams, the problem has non-trivial annotated streaming schemes with total cost $\widetilde{O}(k^2 n)$.*

▶ **Theorem 10.** *Under dynamic graph streams on n-node graphs, there exists a $[k^2 n + h, v]$-scheme for k-edge-connectivity for any h, v such that $h \cdot v = n^2$. In particular, under such streams, the problem has non-trivial annotated streaming schemes with total cost $\widetilde{O}(k^2 n)$.*

▶ **Theorem 11.** *Under dynamic graph streams on n-node graphs, there exists an $[n, n]$-scheme for k-edge-connectivity (for any k). In particular, under such streams, the problem has non-trivial annotated streaming schemes with total cost $\widetilde{O}(n)$.*

Contrast this with our results for annotated streaming under SGT streams, where the total cost for these problems can be as large as $\Omega(n^{\text{polylog(n)}})$. Thus, we can see a conceptual separation between classical streaming and annotated streaming: the former can tolerate SGT streams incurring negligible factors in complexity over dynamic graph streams, whereas the latter incurs significantly large factors for SGT streams over dynamic.

Finally, we complement our annotated streaming lower bounds for k-connectivity under SGT streams with an upper bound.

▶ **Theorem 12.** *Under SGT streams with parameter α, there exists a $[n^2 \log \alpha + k^2 n, 1]$-scheme for k-vertex-connectivity and k-edge-connectivity.*

1.3 Related Work

In their seminal paper, Babai, Frankl, and Simon [6] defined communication classes similar to classes in computational complexity; this included the Merlin-Arthur (MA) communication class, which essentially defined the Merlin-Arthur communication model. Klauck [22] proved that disjointness and inner product have MA complexity $\Omega(\sqrt{N})$, which was surprisingly proven to be tight (up to logarithmic factors) by Aaronson and Wigderson [1] who gave a protocol with total cost $O(\sqrt{N \log N})$. Chen[13] recently improved this bound to $O(\sqrt{N \log \log \log N})$. Chakrabarti et al. [9] were the first to consider the online version of the MA model. They used it to show lower bounds for annotated streaming schemes, as has been traditionally done by subsequent works. They proved that for any function f, an $[h, v]$-OMA-protocol that computes it must have $h \cdot v \geq R^{\rightarrow}(f)$. Further, for many problems including frequency moments and subset checks, they gave annotated streaming schemes (which imply OMA protocols with the same bounds), achieving this smooth tradeoff. [7] were the first to exhibit a problem, namely sparse index, where such a tradeoff is not possible. In fact, they showed that for sparse index with sparsity logarithmic in the input size, the OMA complexity is as large as the one-way communication complexity. Later, Thaler [29] exhibited two more problems with this property: XOR-connectivity and XOR-bipartiteness. The motivation was to show the existence of graph problems that provably need semi-streaming schemes (an $[n, n]$-scheme for n-node graphs) to solve and that they are equally hard in the annotated streaming model as in classical streaming.

An "augmented" version of the Equals-Index was studied by Jayram and Woodruff [21] in the classical communication model. They called it the "Augmented Index problem on large domains": here Bob also knows all the entries of Alice's vector before his input index.

For the problem of computing distinct items, [9] gave an $(n^{2/3}(\log n)^{4/3}, n^{2/3}(\log n)^{4/3})$-scheme, which was later simplified and improved to an $(n^{2/3} \log n, n^{2/3} \log n)$-scheme by [18]. Note that both of these works assume that the stream length m is $O(N)$, where N is the universe size. Our results are for streams that are exponentially longer.

With the growing interest in graph streaming algorithms over the last couple of decades, much of the recent literature on stream verification has focused on graph problems [15, 7, 2, 29, 11, 12]. Most of them design stream verification protocols for insert-only or insert-delete graph streams. For the graph connectivity problem on n-node graphs, [9] gave an $[h, v]$-scheme for any h, v with $h \geq n$ and $h \cdot v = n^2$. For sparse graphs with m edges, [7] designed an $[n + m/\sqrt{v}, v]$-scheme. As mentioned above, [29] studied the problem in the XOR-edge-update model and proved that any $[h, v]$-scheme must have $(h + n) \cdot v \geq n^2$. A number of works [15, 11, 12] studied verification schemes for shortest-path and s, t-connectivity related problems. No prior work on stream verification, however, studied the k-connectivity problem.

In the classical streaming model, [4] gave the first algorithm for k-edge-connectivity in dynamic streams using $\widetilde{O}(kn)$ space. [19] gave the first algorithm for k-vertex-connectivity in dynamic graph streams, which was improved by a factor of k and made nearly optimal by [5] using $\widetilde{O}(kn)$ space. [27] proved a lower bound of $\Omega(kn)$ bits for both problems, even for insertion-only streams (see also [5] for extending the lower bound for k-vertex-connectivity to multiple passes).

Other variants of stream verification include a *prescient* setting where Prover knows the entire stream upfront, i.e., before Verifies sees it, and can send help messages accordingly [9, 7]. Versions where Prover sends very large proofs have also been considered [23]. Natural generalizations to allow multiple rounds of interaction between the Prover and Verifier have been investigated. These include *Arthur-Merlin streaming protocols* of Gur and Raz [20] and the *streaming interactive proofs* (SIP) of Cormode et al. [16]. The latter setting was further studied by multiple works [2, 10, 24]. Very recently, the notion of *streaming zero-knowledge proofs* has been explored [14]. For a more detailed survey of this area, see [28].

1.4 Technical Overview

We give a high-level overview of the techniques and proofs in the paper.

1.4.1 Communication Lower Bounds

The Equals-Index lower bound. The basis of all our lower bounds is an OMA lower bound on the EQ-IDX problem. Recall the classical index problem where Alice has a string x of length N and Bob has an index $j \in [N]$ such that he needs to know $x[j]$. Chakrabarti et al. [9] showed that for any p, q with $p \cdot q = N$, we can get a (q, p)-OMA-protocol as follows. The string x can be partitioned into p chunks of length q each. Merlin sends Bob the purported chunk where j lies, thereby sending q bits. Again, Alice sends Bob an $O(1)$-size equality sketch for each chunk, thereby sending $O(p)$ bits. Using the relevant sketch, Bob can figure out whether the chunk sent by Merlin is accurate and find the solution if it is.

Note that from Alice's perspective, the problem actually boils down to the following subproblem: Alice has a string x and Bob has a string y, and she needs to help him verify whether it is identical to her k^{th} chunk, where she doesn't know k. Our main observation is that in the above protocol, to solve this subproblem, she spends as many bits as she would have *without* Merlin: $\Theta(p)$ bits. So now that Alice and Bob have Merlin, why not take his help and improve this communication to $o(p)$? If they could do this with at most $O(q)$ bits of help, then they would obtain an improved $(q, o(p))$-OMA Index protocol. But the known lower bound for index [9] says that the product of hcost and vcost must be $\Omega(pq)$. Hence, Merlin cannot bring down the vcost to $o(p)$ without sending $\omega(q)$ bits. But q can be much larger than p – even exponential in p – where $\Theta(p)$, as noted, is the communication needed for the subproblem in the classical one-way model. Hence, we identify a problem whose non-trivial OMA complexity can be much larger than their one-way complexity. We essentially abstract out this subproblem as the $\text{EQ-IDX}_{p,q}$ problem and formalize the reduction.

The Sparse-Index lower bound. Although this lower bound follows by a reduction from EQ-IDX, we discuss it separately to point out the challenges in proving its non-trivial-OMA complexity. To show that $\widehat{\text{MA}}^{\rightarrow}(\text{SP-IDX}_{\log\log N, N})$ is $\Omega(\log N)$, we first prove a tight bound on its one-way complexity. Prior work had shown that $R^{\rightarrow}(\text{SP-IDX}_{m,N}) = O(m \log m + \log N)$ for general m. Although the $\log m$ factor can be easily removed, it is not clear that the additive $\log N$ factor can be removed since it comes from the minimum size of a hash family.

It would then not give us the desired bound of $O(\log \log N)$. We remove it as follows. We note that SP-IDX$_{m,N}$ actually has input size $\binom{n}{m}$, use a public random hash function, and then appeal to Newman's theorem. This essentially implies that the existence of a better hash function can be derived from Newman's theorem, and can tighten the upper bound.

1.4.2 Streaming Algorithms

Some of our algorithms for SGT streams follow by replacing the ℓ_0-samplers in existing algorithms by the new ℓ_0-samplers that we design in this work. However, an algorithm for k-edge-connectivity does not immediately follow. So we design a new algorithm here, and this is one of our significant technical contributions. Another key technical ingredient is our "layering lemma" that we use to design efficient schemes for vertex connectivity on dynamic graphs. We discuss these tools and techniques in detail below.

Strong ℓ_0-samplers. We design new ℓ_0-samplers that can handle very large frequencies (hence called strong) and still take polylog(n) space. This is helpful because SGT streams can have very large frequencies. Standard ℓ_0-samplers usually assume that the frequencies are poly(n) where n is the universe size.

The main tool we use to design our sampler is our *decision counter*. These counters, given a stream of insertions and deletions, can detect if the number of insertions is exactly equal to the number of deletions or not, using very little space. This even works when the difference between the number of insertions and deletions is very large which is challenging to do in small space deterministically. The idea is to maintain a standard counter modulo a large random prime.

We then use ideas from the sparse recovery and ℓ_0-sampling literature to build the strong ℓ_0-sampler. We first solve the problem when the non-zero support is 1 using non-adaptive binary search. The problem here is that we are given a stream of insertion and deletions over a universe of n elements and promised that at the end of the stream, there is exactly one element with a non-zero frequency. The goal is to find that element. The usual idea is to maintain one counter that keeps a track of the number of elements (i.e. number of insertions minus deletions). Another counter is maintained which is used to find the value of the non-zero frequency element. When an insertion/deletion for element i arrives it is scaled by i and then added/subtracted from the counter. At the end of the stream, this counter contains i times frequency of element i and the other counter has just the frequency of element i thus recovering the non-zero frequency element i. This does not work when the frequencies are large so we have to use non-adaptive binary search to solve the problem which uses $O(\log n)$ counters to find i.

In the case with no promise, we want to reduce to the support 1 case. We do this by guessing the support size s in powers of 2 and sampling the elements with probability $1/s$. This means that for the correct guess of s we reduce to the support 1 case with constant probability. We can repeat $O(\log n)$ times for high probability. Finally, we need to distinguish the support 1 case from the cases where the support is not 1. This is because we have many problems and only a few of them are the support 1 case and in others the support can be 0 or larger than 1. We design a sketch for this by randomly partitioning the universe into many parts, summing those parts up and checking how many parts are non-zero. The hope is that if there are at least 2 elements then multiple parts will become non-zero.indicating that the support is more than 1. If the non-zero support is 1 then exactly one part will be non-zero and if the non-zero support is 0 then all parts will be 0. This happens with constant probability so we repeat $O(\log n)$ times for high probability. Putting everything together gives us the strong ℓ_0-sampler sketch.

Edge Connectivity. We give a randomized algorithm for getting a certificate (a spanning subgraph with the same answer to k-edge-connectivity) of k-edge-connectivity. The certificate is of size $\widetilde{O}(kn)$, which is optimal up to polylog factors. The certificate needs to be of size $\Omega(kn)$ because every vertex needs to have degree at least k. This algorithm can be easily converted into a dynamic streaming algorithm using the dynamic streaming spanning forest implementation of [4]. We also show that using our strong ℓ_0-samplers, the spanning forest streaming implementation of [4] can be extended to SGT streams in $\widetilde{O}(n)$ space. Using our certificate along with the spanning forest algorithm in SGT streams, we can solve k-edge-connectivity in SGT streams in $\widetilde{O}(kn)$ space.

[4] also give an algorithm for k-edge-connectivity in dynamic streams using $\widetilde{O}(kn)$ space. However, this algorithm cannot be extended to SGT streams by simply replacing their ℓ_0-samplers with strong ℓ_0-samplers. Their certificate is k edge-disjoint spanning forests. On a high level, this does not work in SGT streams because the spanning forests depend on each other (since they are edge-disjoint). In their algorithm, after recovering one spanning forest T_1 of G we essentially need to delete the edges of T_1 from G and then recover a new spanning forest T_2 (to get a disjoint spanning forest). This is easy to do in dynamic streams because the frequencies of the edges of T_1 are exactly 1. So we can generate a new stream which is the old stream appended with deletions of the edges of T_1. Then T_2 can be recovered from this new stream. However, in SGT streams, the frequency of the edges of T_1 could be arbitrary, so we cannot easily generate another stream that represents the graph $G - T_1$ (to do this we need to know the exact frequencies of all edges in T_1). This dependency between the spanning forests makes extending this algorithm difficult. Our algorithm, on the other hand, gets rid of this dependency and thus is extendable to SGT streams.

The randomized algorithm for getting a certificate of k-edge-connectivity is heavily inspired by the randomized algorithm for getting a certificate of k-vertex-connectivity [5]. In the randomized algorithm, we independently sample every edge with probability $1/k$ and find a spanning forest of the sampled subgraph. The certificate then is a union of the spanning forests in $\widetilde{O}(k)$ such independent iterations. The analysis then shows that edges whose endpoints are not very well connected in the original graph exist in the certificate, and pairs of vertices that are very well connected in the original graph are well connected in the certificate. This is enough to prove that the certificate preserves the answer to k-edge connectivity.

We also give an $[n, n]$-scheme for k-edge-connectivity in the annotated dynamic streaming model. We achieve this by simulating the two-pass streaming algorithm for minimum-cut implied by [26]. During the stream, the verifier computes a cut sparsifier of the graph. After the stream, the verifier compresses the vertices into supernodes (by compressing all large cuts) such that only the small cuts remain. The prover then sends all the edges of this supernode graph ([26] showed that there are only $O(n)$ such edges). The verifier then can compute the exact mincut of the graph, thus solving k-edge-connectivity for all values of k.

Vertex connectivity. We also get an algorithm for k-vertex-connectivity in SGT streams in $\widetilde{O}(kn)$ space using the techniques used for edge connectivity. We use strong ℓ_0-samplers in the algorithm of [5].

We also give an $[k^2n, n/k]$-scheme for k-vertex-connectivity in the annotated dynamic streaming model. The heart of the algorithm is what we call the layering lemma, which says that there exists a short proof (size $\widetilde{O}(kn)$) to show that any arbitrary fixed vertex has k vertex-disjoint paths to all other vertices. This proof can also be verified in $\widetilde{O}(n/k)$ space. Also, when proving this for r vertices, we can reuse space and thus get a $[r \cdot kn, n/k]$-

scheme instead of a $[rkn, rn/k]$-scheme. If we show the layering lemma for k vertices, then using properties of k-vertex-connectivity, we can show that the graph is k-vertex-connected, giving the desired bound. We also extend these ideas to k-edge-connectivity getting a $[k^2n, n/k^2]$-scheme.

We show that a short proof exists for the layering lemma using the probabilistic method. The idea is to cleverly partition the vertices into $\log n$ layers and show that vertices in the first layer have k vertex-disjoint paths to the special vertex. Then we inductively show that vertices in a layer have k vertex-disjoint path to the previous layer. Using the properties of k-vertex-connectivity, this is enough to show that the special vertex has k vertex-disjoint paths to all other vertices. The proof between any two layers is of size $\widetilde{O}(kn)$, giving us the desired bound. The verification is more involved, but the verifier essentially checks just two things using some tools. He first checks whether the edges sent by the prover belong to the input graph (using a subset check). He then checks if what the prover sent are indeed vertex-disjoint paths (using a duplicate-detection scheme).

We also give an $[n^2 \log \alpha + k^2 n, 1]$-scheme for k-vertex-connectivity and k-edge-connectivity in the annotated SGT streaming model with parameter α. The proof idea is the same as in dynamic streams, but the verification is more complicated because the frequencies for the edges could be large or even negative. The auxiliary information used for verification adds an overhead of $\widetilde{O}(n^2 \log \alpha)$ bits in the proof. The verification follows the same steps as in the dynamic streaming case, except the subset check is not easy to do. So we came up with a different way to do the subset check, which needs a large amount of auxiliary information. One of the main ideas for this is to separate out the elements with positive and negative frequencies and do a subset check for them separately.

2 Models, Notation, and Terminology

Here, we first formally describe the Merlin-Arthur communication model and the annotated streaming model, with formal definitions of non-trivial complexity in each case. Next, we give an account of notation and terminology used throughout the paper.

Merlin-Arthur Communication. In the Merlin-Arthur (MA) communication model [6], we have the usual two players Alice and Bob (collectively called "Arthur") with their inputs $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ respectively. They want to compute $f(x, y)$, where $f : \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$. In addition, there is an all-powerful player Merlin who knows the inputs x and y. Merlin, being untrusted, sends Alice and Bob a "proof" in support of his answer for $f(x, y)$, following which these two players interact between themselves to verify the proof. In this paper, we focus on the *online Merlin-Arthur* (OMA) model which is an MA analog of the one-way randomized communication model. We describe this model in more detail.

Online Merlin-Arthur communication. An OMA protocol Π works as follows. Merlin sends Bob a help message $\mathcal{H}$. Then Alice generates a random string $\mathcal{R}$, based on which she sends Bob a message msg. Bob then outputs $\mathsf{out} \in \{0, 1, \perp\}$ as a function of $(y, \mathcal{H}, \mathsf{msg}, \mathcal{R})$. An OMA protocol Π is said to have completeness error δ_c and soundness error δ_s if the following conditions are satisfied.

- (Completeness) If $f(x, y) = 1$, then there exists a function $\mathcal{H}$ such that $\Pr_{\mathcal{R}}[\mathsf{out} \neq 1] \leq \delta_c$.
- (Soundness) If $f(x, y) = 0$, then for all possible $\mathcal{H}$, we have $\Pr_{\mathcal{R}}[\mathsf{out} = 1] \leq \delta_s$.

We say that a protocol Π *solves* a function f if Π has $\delta_s, \delta_c \leq 1/3$ for computing f.

For an OMA protocol Π, the *help cost* $\mathsf{hcost}(\Pi)$ is the maximum length of the message $\mathcal{H}$ sent by an honest Merlin over all possible (x, y), and the *verification cost* $\mathsf{vcost}(\Pi)$ is the maximum number of bits sent by Alice over all possible $(x, \mathcal{R})$. The *total cost* $\mathsf{tcost}(\Pi)$ is then defined as $\mathsf{hcost}(\Pi) + \mathsf{vcost}(\Pi)$.

The OMA-complexity of f is defined as

$$\mathrm{MA}^{\rightarrow}(f) := \min\{\mathsf{tcost}(\Pi) : \Pi \text{ solves } f\}$$

We now define the non-trivial-OMA complexity of a function. To this end, we first formally define trivial and non-trivial OMA protocols.

▶ **Definition 13** ((Non-)Trivial OMA protocol)**.** *We say that an OMA protocol* Π *solving a function* f *is* trivial *if* $\mathsf{vcost}(\Pi) = \Omega(R^{\rightarrow}(f))$ *and is* non-trivial *if* $\mathsf{vcost}(\Pi) = o(R^{\rightarrow}(f))$*.*

▶ **Definition 14** (Non-trivial-OMA complexity)**.** *The* non-trivial *OMA complexity of a function* f *is defined as*

$$\widehat{\mathrm{MA}}^{\rightarrow}(f) := \min\{\mathit{tcost}(\Pi) : \Pi \textit{ is a non-trivial protocol that solves } f\}$$

Note that, by definition, $\widehat{\mathrm{MA}}^{\rightarrow}(f) \geq \mathrm{MA}^{\rightarrow}(f)$. We also make the following observation.

▶ **Observation 15.** *If* $\widehat{\mathrm{MA}}^{\rightarrow}(f) = \omega(\mathrm{MA}^{\rightarrow}(f))$*, then it must be that* $\mathrm{MA}^{\rightarrow}(f) = \Omega(R^{\rightarrow}(f))$*.*

This is because if $\widehat{\mathrm{MA}}^{\rightarrow}(f)$ is larger than $\mathrm{MA}^{\rightarrow}(f)$, then the "optimal" protocol Π for which $\mathrm{MA}^{\rightarrow}(f) = \mathsf{tcost}(\Pi)$ must be a trivial protocol. Hence, $\mathsf{tcost}(\Pi) = \Omega(\mathsf{vcost}(\Pi)) = \Omega(R^{\rightarrow}(f))$ by definition.

Annotated Streaming. In the annotated streaming model of [8], we have a space-bounded Verifier and an all-powerful Prover with unlimited space. Given an input stream σ, a *scheme* for computing a function $f(\sigma)$ is a triple $\mathcal{P} = (\mathcal{H}, \mathcal{A}, \mathsf{out})$, where $\mathcal{H}$ is a function that Prover uses to generate the help message or proof-stream $\mathcal{H}(\sigma)$ that she sends Verifier *after* the input stream, $\mathcal{A}$ is a data streaming algorithm that Verifier runs on σ using a random string $\mathcal{R}$ to produce a summary $\mathcal{A}_{\mathcal{R}}(\sigma)$, and out is an algorithm that Verifier uses to process the proof $\mathcal{H}(\sigma)$ and generate an output $\mathsf{out}(\mathcal{H}(\sigma), \mathcal{A}_{\mathcal{R}}(\sigma), \mathcal{R}) \in \mathrm{range}(f) \cup \{\perp\}$, where $\perp$ denotes rejection of the proof. Note that if the proof length $|\mathcal{H}(\sigma)|$ is larger than the memory of the Verifier, then out is a streaming algorithm that processes $\mathcal{H}(\sigma)$ as a stream and stores a summary subject to its memory.

A scheme $\mathcal{P} = (\mathcal{H}, \mathcal{A}, \mathsf{out})$ has completeness error δ_c and soundness error δ_s if it satisfies

- (completeness) $\forall \sigma : \Pr_{\mathcal{R}}[\mathsf{out}(\mathcal{H}(\sigma), \mathcal{A}_R(\sigma), \mathcal{R}) \neq f(\sigma)] \leq \delta_c$;
- (soundness) $\forall \sigma, \mathcal{H}' : \Pr_{\mathcal{R}}[\mathsf{out}(\mathcal{H}', \mathcal{A}_{\mathcal{R}}(\sigma), \mathcal{R}) \notin \{f(\sigma), \perp\}] \leq \delta_s$.

We say that a scheme $\mathcal{P}$ *solves* a function f if $\mathcal{P}$ has $\delta_s, \delta_c \leq 1/3$ for computing f.

The *hcost* (short for "help cost") of a scheme $\mathcal{P} = (\mathcal{H}, \mathcal{A}, \mathsf{out})$ is defined as $\max_{\sigma} |\mathcal{H}(\sigma)|$, i.e., the maximum number of bits required to express a proof over all possible inputs σ. The *vcost* (short for "verification cost") is the maximum bits of space used by the algorithms $\mathcal{A}$ and out respectively, where the maximum is taken over all possible $(\sigma, \mathcal{R})$. The total cost $\mathsf{tcost}(\mathcal{P})$ is defined as the sum $\mathsf{hcost}(\mathcal{P}) + \mathsf{vcost}(\mathcal{P})$.

The *annotated streaming complexity* of a function f is defined as

$$\mathrm{AS}(f) := \min\{\mathsf{tcost}(\mathcal{P}) : \mathcal{P} \text{ solves } f\}.$$

We now define the non-trivial annotated-streaming complexity of a function. First, we formally define trivial and non-trivial schemes.

▶ **Definition 16** ((Non-)Trivial scheme)**.** *We say that a scheme* $\mathcal{P}$ *solving a function* f *is* trivial *if* $\mathsf{vcost}(\mathcal{P}) = \tilde{\Omega}(\mathcal{S}(f))$*, and* non-trivial *otherwise.*

▶ **Definition 17** (Non-trivial Annotated Streaming complexity). *The* non-trivial *annotated streaming complexity of a function f is*

$$\widehat{\mathrm{AS}}(f) := \min\{\mathit{tcost}(\mathcal{P}) : \mathcal{P} \textit{ is a non-trivial scheme that solves } f\}.$$

A scheme $\mathcal{P}$ with $\mathsf{hcost}(\mathcal{P}) = O(h)$ and $\mathsf{vcost}(\mathcal{P}) = O(v)$ is called an (h, v)-scheme. Again, if $\mathsf{hcost}(\mathcal{P}) = \widetilde{O}(h)$ and $\mathsf{vcost}(\mathcal{P}) = \widetilde{O}(v)$, we call $\mathcal{P}$ an $[h, v]$-scheme.

Support Graph Turnstile Streams. We introduce support graph turnstile (SGT) streams in this work. Conceptually, the graph is induced by the support of the frequency vector of the input stream, which may be a turnstile stream. The formal definition is as follows.

▶ **Definition 18** (Support graph turnstile streams). *A turnstile stream σ is called a support graph turnstile stream if it is of the form $\langle (u, v)_i, \Delta_i) : i \in [m]\rangle$, where $(u, v) \in \binom{[n]}{2}$ and the graph that it defines is given by $G = ([n], \{(u, v) : \mathrm{freq}(u, v) \neq 0\})$.*

Basic Notation. We define some notation and terminology that we use throughout the paper. All logarithms are base 2. The $\widetilde{O}(.), \tilde{\Omega}(.), \tilde{\omega}(.)$ notation hides factors polylogarithmic in the input size. The notation $[k]$ for a natural number k denotes the set $\{1, \ldots, k\}$. We use $[a, b]$ for integers $a < b$ to denote the set $\{a, \ldots, b\}$. For a string $z \in [\alpha]^d$, we use $z[k]$ to denote the element at the kth index of z. We use the term "with high probability" to mean with probability at least $1 - 1/\mathrm{poly(N)}$, where N is the input size. Although the standard notation to denote the input size in a communication problem is n, we use N instead to avoid confusion with the number of nodes (for which we use n).

Graph Notation. All graphs in this paper are simple and undirected. Given a graph $G = (V, E)$, we use n for its number of nodes $|V|$ and m for its number of edges $|E|$ unless specified otherwise. The degree of a vertex $v \in V$ is denoted by $\deg(v)$, and $N(v)$ denotes its neighborhood. For a subset F of edges in E, we use $V(F)$ to denote the vertices incident on F; similarly, for a set U of vertices, $E(U)$ denotes the edges incident on U. We further use $G[U]$ for any set U of vertices to denote the induced subgraph of G on U. For any two vertices $s, t \in V$, we say that a collection of s-t paths are vertex-disjoint if they do not share any vertices other than s and t.

3 Preliminaries

In this section, we first list standard tools from the literature that we use throughout the paper. Next, we give an account of the tools that we formulate in this work, and use to prove the main theorems.

3.1 Standard tools

In the "Equality" problem $\textsc{Eq}_N$, Alice and Bob hold strings $x, y \in \{0, 1\}^N$ and need to check whether $x = y$.

We use the following basic communication complexity facts.

▶ **Fact 19** (Equality-sketch [25]). *There is an $O(1)$-cost public random protocol for $\textsc{Eq}_N$ for any N with error probability at most $1/3$.*

▶ **Fact 20** (Equality lower bound [25]). *The one-way randomized communication complexity $R^{\rightarrow}(\textsc{Eq}_N) = \Omega(\log N)$.*

▶ **Fact 21** (Index lower bound [3]). *The one-way randomized communication complexity* $R^{\rightarrow}(IDX_N) = \Omega(N)$.

▶ **Fact 22** (Newman's Theorem[25]). *For any function* $f : \{0,1\}^N \times \{0,1\}^M \rightarrow \{0,1\}$, *a public coin protocol with communication cost* C *and error* ε *can be simulated by a private coin protocol with communication cost at most* $O(C + \log(N+M) + \log(1/\delta))$ *and error at most* $\varepsilon + \delta$.

We often refer to the following lower bound in the OMA model and annotated streaming.

▶ **Fact 23** (General OMA lower bound,[9]). *For any function* f, *an* (h,v)*-OMA-procotol or an* (h,v)*-scheme* Π *solving it must have:* $h \cdot v \geq \Omega(C)$, *where* $C = R^{\rightarrow}(f)$ *in case* Π *is an OMA protocol, and* $C = \mathcal{S}(f)$ *(the streaming complexity of* f*) in case it is an annotated streaming scheme.*

The standard schemes below are often used as subroutines in our protocols.

▶ **Fact 24** (Subset-check and Intersection-count Scheme, [9], [12]). *Given an insert-delete stream of elements from sets* $X, Y \subseteq [N]$ *(interleaved arbitrarily), for any* h, v *with* $h \cdot v = N$, *there are* $[h,v]$*-schemes for checking whether* $X \subseteq Y$ *and for counting* $|X \cap Y|$.

▶ **Fact 25** (Duplicate-detection Scheme). *Given an insert-only stream of elements from the universe* $[N]$, *for any* h, v *with* $h \cdot v = N$, *there is an* $[h,v]$*-scheme for checking whether all elements in the stream are distinct.*

We give a new way to detect if two multi-sets are identical using ℓ_0-samplers. The idea is to insert elements of the first multi-set into an ℓ_0-sampler and delete elements of the second multi-set from the sampler, and at the end, check if the sampler is empty.

▶ **Fact 26** (Equality-Detection Scheme). *Given an insert-only stream of elements of two multi-sets* A, B *in any order (interleaved arbitrarily) from the universe* $[N]$, *we can check if the two sets are identical with probability* $1 - 1/\mathrm{poly(N)}$ *in space* $\mathrm{polylog(N)}$.

3.2 New tools

In this subsection, we introduce the new tools we use to get our results. To prove the result (Theorem 8) for k-edge connectivity, we need to come up with a new algorithm. All the proofs and the algorithms are deferred to the full version of the paper.

▶ **Theorem 27.** *Given any graph* $G = (V, E)$ *and any integer* $k \geq 1$, *there is an algorithm that outputs a certificate* H *of* k*-edge-connectivity of* G *with* $O(kn \cdot \log n)$ *edges with high probability. This algorithm can be implemented in dynamic streams in* $\widetilde{O}(kn)$ *space and in SGT streams in* $\widetilde{O}(kn \cdot \mathrm{polylog}\alpha)$ *space.*

The following is the ℓ_0-sampling problem when the frequencies are large:

▶ **Problem 1.** *Given a stream containing insertions and deletions of elements in* $[n]$, *output an element whose frequency is non-zero. The promise is that the value of each coordinate is between* $-\alpha$ *and* α *at the end of the stream.*

▶ **Lemma 28.** *There is a randomized algorithm that solves Problem 1 with probability* $1 - 1/\mathrm{polylog}(\alpha) \cdot \mathrm{poly(n)}$ *and uses* $\mathrm{poly}(\log\log\alpha + \log \mathrm{n})$ *bits of space.*

4 OMA Lower Bound for Equals-Index and its Implications

4.1 The Equals-Index Problem

We formally define the EQ-IDX$_{p,q}$ communication game between two players Alice and Bob as follows.

Let p and q be arbitrary integers. Alice gets p strings $x_1, \ldots, x_p$ such that $x_i \in \{0,1\}^q$ for each $i \in [p]$. Bob gets a string $y \in \{0,1\}^q$ and an index $j \in [p]$. The output is 1 if $y = x_j$, and 0 otherwise.

First, let us show tight bounds on its one-way (Alice $\to$ Bob) randomized communication complexity.

▶ **Lemma 29.** $R^{\to}(\textit{EQ-IDX}_{p,q}) = \Theta(p + \log q)$

Proof. First consider the following protocol using public randomness. For each $i \in [p]$, Alice sends Bob an $O(1)$-size equality sketch (Fact 19) for x_i. Bob uses only the jth sketch to check if $x_j = y$. Thus, $R^{\text{pub}}(\text{EQ-IDX}_{p,q}) = O(p)$. By Newman's theorem (Fact 22), $R^{\to}(\text{EQ-IDX}_{p,q}) = O(p + \log(pq)) = O(p + \log q)$.

The lower bound $R^{\to}(\text{EQ-IDX}_{p,q}) = \Omega(p + \log q)$ easily follows from the facts that EQ-IDX$_{p,1} \equiv$ IDX$_p$ and EQ-IDX$_{1,q} \equiv$ EQ$_q$. Therefore, we have

$R^{\to}(\text{EQ-IDX}_{p,q}) \geq R^{\to}(\text{EQ-IDX}_{p,1}) = R^{\to}(\text{IDX}_p) = \Omega(p)$ (Fact 21).

Again, $R^{\to}(\text{EQ-IDX}_{p,q}) \geq R^{\to}(\text{EQ-IDX}_{1,q}) \geq \Omega(\log q)$ (Fact 20). ◀

Now we prove an OMA lower bound for EQ-IDX.

▶ **Lemma 30.** *Any (h,v)-OMA-protocol solving* EQ-IDX$_{p,q}$ *must have*

$$(h + q) \cdot v \geq \Omega(pq)$$

Proof. We show that given any (h, v)-OMA-protocol Π for EQ-IDX$_{p,q}$, we can design an $(h+q, v)$-OMA-protocol Π' for IDX$_{pq}$. The lower bound of $(h+q)\cdot v \geq \Omega(pq)$ then immediately follows from the OMA lower bound for IDX$_{pq}$ (Fact 21).

Suppose Alice and Bob have inputs $x \in \{0,1\}^{pq}$ and $k \in [pq]$ in the IDX$_{pq}$ problem. Then the protocol Π' is as follows. Alice partitions x into p chunks $x_1, \ldots, x_p$, each of size q. Bob sets $j = \lceil k/p \rceil$. Merlin sends $y \in \{0,1\}^q$ to Bob and claims that it equals x_j. The players can now interpret $(x_1, \ldots, x_p)$ and (y, j) as inputs to the EQ-IDX$_{p,q}$ problem, and run the protocol Π to verify that y is indeed equal to x_j. If the check passes, then Bob knows $x[k]$ since it lies in $x_j = y$. If not, he outputs $\perp$, i.e., rejects the proof.

We analyze the completeness and soundness errors of Π'. If Merlin is honest, then y is indeed x_j, and the protocol fails only if Bob rejects because the check in Π doesn't go through. The probability of this is exactly the completeness error of Π. Hence, Π' has the same completeness error as Π. Again, if Merlin is dishonest, then the protocol fails only when Bob outputs the incorrect bit-value for $x[k]$. This happens only when $y \neq x_j$, but the check passes in Π. This has the same probability as the soundness error of Π. Therefore, Π' also has the same soundness error as Π. Thus, by definition, since Π solves EQ-IDX$_{p,q}$, the protocol Π' solves IDX$_p$.

Finally, we analyze the cost of Π'. Merlin sends Bob y as well as his message due to Π. Thus, $\text{hcost}(\Pi') = O(h + q)$. Alice sends Bob only her message due to Π, implying that $\text{vcost}(\Pi') = v$. Thus, Π' is an $(h + q, v)$-OMA-protocol as claimed. This completes the proof. ◀

Our main result on the non-trivial OMA complexity $\widehat{\mathrm{MA}}^{\rightarrow}(\text{EQ-IDX}_{p,q})$ follows from the above two lemmas.

▶ **Theorem 1.** *For any p, q with $p = \Omega(\log q)$, we have $\widehat{\mathrm{MA}}^{\rightarrow}(\textit{EQ-IDX}_{p,q}) = \omega(q)$*

Proof. By Lemma 29, we have $R^{\rightarrow}(\text{EQ-IDX}) = \Theta(p)$ for this setting of p and q. Therefore, any non-trivial OMA protocol Π for the problem must have $\mathsf{vcost}(\Pi) = o(p)$. But by Lemma 30, if $\mathsf{hcost}(\Pi) = O(q)$, then $\mathsf{vcost}(\Pi) = \Omega(p)$. Therefore, $\mathsf{hcost}(\Pi)$ must be $\omega(q)$, which means $\mathsf{tcost}(\Pi) = \omega(q)$. Then, by definition, $\widehat{\mathrm{MA}}^{\rightarrow}(\text{EQ-IDX}) = \omega(q)$. ◀

▶ **Corollary 2.** *There is an explicit function f with*

$$\widehat{\mathrm{MA}}^{\rightarrow}(f) = \exp(\Omega(\mathrm{MA}^{\rightarrow}(f))) \text{ and } \widehat{\mathrm{MA}}^{\rightarrow}(f) = \exp(\Omega(R^{\rightarrow}(f))).$$

Proof. Set $f = \text{EQ-IDX}_{p,q}$ with $p = \log n$ and $q = n/\log n$. Indeed, $p \geq \log q$. Hence, by Theorem 1, we have $\widehat{\mathrm{MA}}^{\rightarrow}(f) = \omega(q) = \exp(\Omega(p))$.

By Lemma 29, we have $R^{\rightarrow}(f) = \Theta(p)$. Also, by definition $\mathrm{MA}^{\rightarrow}(f) \leq R^{\rightarrow}(f) = \Theta(p)$. Therefore, the claimed result holds. ◀

▶ **Corollary 3.** *For any $C \in (\sqrt{N}, N/\log N)$, there is an explicit function f on input size N with $\widehat{\mathrm{MA}}^{\rightarrow}(f) = \omega(C)$.*

Proof. Set $f = \text{EQ-IDX}_{p,q}$ with $p = n/C$ and $q = C$. Since $q = C \leq n/\log n$, we have

$$p = n/C \geq \log n > \log q$$

By Theorem 1, $\widehat{\mathrm{MA}}^{\rightarrow}(f) = \omega(C)$. ◀

We also give a tight bound on the (standard) OMA complexity of $\text{EQ-IDX}_{p,q}$.

▶ **Theorem 31.** *For any p, q with $p \geq \log q$, we have $\mathrm{MA}^{\rightarrow}(\textit{EQ-IDX}_{p,q}) = \Theta(\min\{\sqrt{pq}, p\})$.*

Proof. First, we prove the lower bound. Given an (h, v)-OMA-protocol for $\text{EQ-IDX}_{p,q}$, first consider the case that $h > q$. Then, by Lemma 30, we have $h \cdot v = \Omega(pq)$, which means $h + v = \Omega(\sqrt{pq})$. Otherwise, i.e., if $h \leq q$, then by the same lemma, we have $q \cdot v = \Omega(pq)$, which means $v = \Omega(p)$, and hence, $h + v = \Omega(p)$. Therefore, we conclude

$$\mathrm{MA}^{\rightarrow}(\text{EQ-IDX}_{p,q}) = \Omega(\min\{\sqrt{pq}, p\}).$$

For the upper bound, we first show that if $q \leq p$, then we can design a $(\sqrt{pq}, \sqrt{pq})$-OMA-protocol for $\text{EQ-IDX}_{p,q}$. Alice combines $x_1, \ldots, x_p$ into a single pq-bit string x, and then again splits x into $\sqrt{pq}$ chunks $z_1, \ldots, z_{\sqrt{pq}}$, each of which has $\sqrt{pq}$ bits. Observe that since $q \leq p$, we have $q \leq \sqrt{pq}$, and hence, x_j lies entirely in (the concatenation of) at most two consecutive chunks $z_k \circ z_{k+1}$. Merlin sends Bob z_k and z_{k+1} that he claims are identical to z'_k and z'_{k+1}. This takes $O(\sqrt{pq})$ bits. If they are indeed as claimed, Bob can determine whether $y = x_j$. The problem now reduces to checking whether $z'_k = z_k$ and $z'_{k+1} = z_{k+1}$ without Merlin. To this end, it suffices to show that there exists a one-way randomized private-coin protocol of cost $O(\sqrt{pq})$. The protocol is very similar to the one mentioned in the proof of Lemma 29. An $O(\sqrt{pq})$ public-coin protocol can be obtained by having Alice send an $O(1)$-size equality sketch (Fact 19) for each z_i, while Bob uses only z_k and z_{k+1} for the check. Newman's theorem (Fact 22) now gives a private-coin protocol of cost $O(\sqrt{pq} + \log(pq)) = O(\sqrt{pq})$. Therefore, we obtain an OMA protocol of total cost $O(\sqrt{pq})$ for $q \leq p$.

Finally, note that $\mathrm{MA}^{\rightarrow}(\text{EQ-IDX}_{p,q})$ is always trivially upper bounded by $R^{\rightarrow}(\text{EQ-IDX}_{p,q})$, which, by Lemma 29, is $O(p)$ for this setting of $p \geq \log q$. Hence, we can conclude that whenever $p \geq \log q$, $\mathrm{MA}^{\rightarrow}(\text{EQ-IDX}_{p,q}) \leq O(\min\{\sqrt{pq}, p\})$. ◀

4.2 Implications of the Equals-Index Lower Bounds

By reduction from EQ-IDX$_N$, we prove the following theorem (proof given in full version).

▶ **Theorem 32.** $\widehat{\mathrm{MA}}^{\rightarrow}(\textit{SP-IDX}_{\log\log n,n}) = \omega(\log n)$.

Compare this with the fact that $\mathrm{MA}^{\rightarrow}(\text{SP-IDX}_{\log\log n,n}) = R^{\rightarrow}(\text{SP-IDX}_{\log\log n,n}) = \Theta(\log\log n)$. Hence, we show an exponential separation between $\mathrm{MA}^{\rightarrow}$ and $\widehat{\mathrm{MA}}^{\rightarrow}$ complexity of SP-IDX with sparsity $\log\log n$.

▶ **Corollary 4.** *For $m = \log\log N$, we have $\widehat{\mathrm{MA}}^{\rightarrow}(\textit{SP-IDX}_{m,N}) = \omega(\log N)$, whereas $R^{\rightarrow}(\textit{SP-IDX}_{m,N}) = \mathrm{MA}^{\rightarrow}(\textit{SP-IDX}_{m,N}) = \Theta(\log\log N)$.*

We also get the following results by reduction from EQ-IDX. The proofs appear in the full version.

▶ **Corollary 5** ([29], paraphrased). *For $f \equiv \textit{XOR-CONN}_n$ or $f \equiv \textit{XOR-BIP}_n$, we have*

$$\widehat{\mathrm{MA}}^{\rightarrow}(f) = \Omega(R^{\rightarrow}(f)) = \Omega(n).$$

▶ **Theorem 6.** *There is a setting of F such that $\textit{DIST-ITEM}_{N,F}$ can be solved in $\widetilde{O}(N)$ space in classical streaming, but any non-trivial annotated-streaming scheme for the problem must have total cost $\Omega(N^{\mathrm{polylog(N)}})$.*

4.2.1 Connectivity Problems in Support Graph Turnstile Streams

We prove Theorems 7 and 8 here. We first make the following claim about classical streaming complexities of connectivity and k-connectivity.

▷ Claim 33. In the classical streaming model under SGT streams, graph connectivity can be solved in $\widetilde{O}(n)$ space, and each of the k-vertex-connectivity and k-edge-connectivity problems can be solved in $\widetilde{O}(kn)$ space. These bounds match (up to polylogarithmic factors) the space-bound known for each problem under dynamic graph streams.

Proof. The upper bounds of connectivity and k-vertex-connectivity follow by using the [4] and [5] algorithms respectively, and replacing their ℓ_0 samplers with our strong ℓ_0 samplers (Lemma 28). For the k-edge connectivity problem, it is not clear that the algorithm by [4] can be implemented under SGT streams. So we design a new $\widetilde{O}(kn)$-space k-edge-connectivity algorithm under such streams. The claim then follows. ◁

We show a lower bound for connectivity in this model.

▶ **Lemma 34.** *Any (h, v)-scheme for connectivity satisfies $(h + qn) \cdot v \geq n^2 \cdot q$. In particular, when $q = n^{\mathrm{polylog(n)}}$, for $v = o(n)$ we need $h \geq n^{\mathrm{polylog(n)}}$.*

The proof of this lemma is along the same lines as the proof of XOR-Connectivity except that we use EQ-IDX$_{n,qn}$. Alice and Bob construct the same graph, but now for each pair of vertices there could be multiple edge insertions (up to 2^q).

We now prove lower bounds for k vertex and edge connectivity in this model.

▶ **Lemma 35.** *Any (h, v)-scheme for k-vertex-connectivity or k-edge-connectivity satisfies $(h + q) \cdot v \geq kn \cdot q$. In particular, when $q = 2^{\mathrm{polylog(n)}}$, for $v = o(kn)$ we need $h \geq 2^{\mathrm{polylog(n)}}$.*

Say that we are given an (h, v)-scheme for k-vertex-connectivity or k-edge-connectivity in the Support Graph Turnstile streaming model. Given an instance of EQ-IDX$_{kn,q}$, Alice constructs the following graph stream (see Figure 1). Let $V_L = \{u_1, \ldots, u_n\}$ and $V_R = \{v_1, \ldots, v_k\}$. For each $\ell \in [kn]$, Alice inserts an edge between the ℓ^{th} pair of vertices x_ℓ times. Given his inputs $y \in \{0,1\}^q$ and $j \in [kn]$, Bob creates the following graph stream. Bob deletes an edge between the j^{th} pair of vertices y times. Bob also adds an edge between every pair of vertices except the j^{th} pair. Alice runs the (h, v)-scheme on her graph stream and sends the memory content to Bob, who then continues running the scheme on his graph stream. Iff the graph is k-connected, then output 0 for the EQ-IDX$_{kn,q}$ instance and output 1 otherwise. Note that the complete bipartite graph is k-connected, but even if you remove one edge it is not k-connected. The correctness follows from the following claim.

▷ Claim 36. The graph formed by the stream is k-connected in the support graph turnstile model iff the EQ-IDX$_{kn,q}$ instance has value 0.

Proof. Consider the case when the EQ-IDX$_{kn,q}$ instance has value 1. We have $x_j = y$, so the j^{th} pair of vertices has y edge insertions and y edge deletions. This means that there is no edge between the j^{th} pair of vertices, implying that the support graph is not k-connected.

Now consider the case when the EQ-IDX$_{kn,q}$ instance has value 0. We have $x_j \neq y$, so the j^{th} pair of vertices has a different number of edge insertions and edge deletions. This implies that the j^{th} pair of vertices has a non-zero support implying that the j^{th} edge exists. Every other pair of vertices has only edge insertions, including exactly 1 edge insertion by Bob. This implies that every other pair of vertices has a non-zero support implying that the edge exists. Thus, the support graph is a complete bipartite graph implying that the support graph is k-connected. ◁

Claim 36 shows that this protocol solves EQ-IDX$_{kn,q}$. We know by Lemma 30 that the following holds for any protocol that solves EQ-IDX$_{kn,q}$: $(h + q) \cdot v \geq kn \cdot q$.

The only help from Merlin is h bits for the (h, v)-scheme. The only communication from Alice is sending the memory content that takes v bits of space. This bound implies that even if $h = O(q)$, $v = \Omega(kn)$. So for $v = o(kn)$ we need $h = \omega(q)$. Setting $q = 2^{\text{polylog(n)}}$ implies that for $v = o(kn)$ we need $h \geq 2^{\text{polylog(n)}}$. However, using the strong ℓ_0-Samplers gives us $h = 0$ and $v = \widetilde{O}(kn)$.

(a) Alice adds x_i edges for pair i.

(b) Bob deletes (u, v) y times and adds every other pair once.

Figure 1 The gadget graphs for reduction to k-connectivity from EQ-IDX.

5 Annotated Streaming Schemes for Dynamic Graph Streams

In this section, we prove Theorem 9 . First we prove our layering lemmas for vertex and edge connectivity that will be useful in our algorithms. We will show complete proofs for vertex-connectivity and then only mention the differences for edge-connectivity because the proofs are very similar. We first show two simple claims (proofs apear in full version):

▷ Claim 37. Let G be k-vertex-connected. Consider a new vertex v and attach it to $\geq k$ arbitrary vertices of G and call this new graph G'. Then G' is k-vertex-connected.

▷ Claim 38. Let $G = (V, E)$ be a graph and let $T \subseteq V$ be a set of vertices. We know that there are at least k vertex-disjoint (edge-disjoint) paths from every vertex of T to a special vertex $t \notin T$. If a vertex $v \notin T \cup \{t\}$ has k vertex-disjoint (edge-disjoint) paths to T then there are k vertex-disjoint (edge-disjoint) paths between v and t.

The following lemmas show short proofs for k disjoint paths from a fixed vertex to all other vertices.

▶ **Lemma 39.** *Let $G = (V, E)$ be a k-vertex-connected graph, and let $t \in V$ be an arbitrary vertex. There is a $\widetilde{O}(kn)$ size proof which shows that there are k vertex-disjoint paths from t to v for all $v \in V - \{t\}$.*

▶ **Lemma 40.** *Let $G = (V, E)$ be a k-edge-connected graph, and let $t \in V$ be an arbitrary vertex. There is a $\widetilde{O}(k^2 n)$ size proof which shows that there are k edge-disjoint paths from t to v for all $v \in V - \{t\}$.*

We will show that a short proof exists using the probabilistic method. We first set up the structure of the proof. Note that we will write disjoint paths without specifying edge-disjoint or vertex-disjoint to mean either of those because the layering structure for both is the same. We want a proof that shows that the special vertex t has k disjoint paths to all vertices in $V - \{t\}$. The idea is to first build a set of vertices T_0 and show that every vertex in T_0 has k disjoint paths to t. The next step is to inductively build sets $T_1, T_2, \ldots T_\ell$ and show that every vertex in T_i has k disjoint paths to the set T_{i-1}. Using Claim 38, this shows that every vertex in T_i has k disjoint paths to t (since every vertex in T_{i-1} has k disjoint paths to t). We keep doing this till we cover all the vertices using the sets T_i and thus, everyone has k disjoint paths to t. This proves the correctness of Lemma 39 and Lemma 40.

We now show how these sets are constructed and bound the proof size. Let $\ell := \log(n/k)$. For $i \in [0, \ell]$, let T_i^L, T_i^R be the sets where every vertex is independently sampled with probability $p_i := \frac{k}{n} \cdot 2^i$. $T_i = T_i^L \cup T_i^R$. Note that we sample all vertices in T_ℓ, so we cover all the vertices. Also, note that it is okay if a vertex is sampled in multiple T_i's. The expected size of T_i is $2^{i+1}k$ for all $i \in [0, \ell]$. We will now show that for any vertex, the proof for k disjoint paths to T_i is small in expectation.

▷ Claim 41. For any vertex v the proof for k disjoint paths to T_i takes size at most $\frac{n}{2^i}$ words in expectation.

Proof. Consider the vertex v and its k disjoint paths $P_1, P_2, \ldots P_k$ to T_i^L (this exists because the graph is k-connected). For any path P_j, we truncate it at the first occurrence of a vertex from T_i^R (since we want paths to $T_i = T_i^L \cup T_i^R$). We now use the randomness of T_i^R. Every vertex on P_j is independently sampled in T_i^R with probability $p_i = \frac{k}{n} \cdot 2^i$. Thus, in expectation, the truncated length of P_j is at most $\frac{1}{p_i} = \frac{n}{k \cdot 2^i}$. Therefore, in expectation, the k disjoint paths to T_i together have size at most $\frac{n}{2^i}$ (by linearity of expectation). ◁

In expectation, the total proof size for k-vertex-connectivity is $O(kn \log(n/k))$ words.

▷ Claim 42. The expected total proof size for k-vertex-connectivity is $O(kn \log(n/k))$ words.

Proof. The proof of k vertex-disjoint paths for a fixed vertex in T_0 takes size at most $O(n)$ words because the summed length of all paths is $O(n)$ (any vertex belongs to at most one vertex-disjoint path). This gives a total expected size of $O(kn)$ words for all vertices in

T_0. The proof for vertices in T_{i+1} takes size at most $\frac{n}{2^i}$ (by Claim 41) which gives a total expected size of $2^{i+2}k \cdot \frac{n}{2^i} = 4kn$ words for all vertices in T_{i+1}. We have $\ell = \log(n/k)$ sets T_i implying the claim. ◁

▷ Claim 43. The expected total proof size for k-edge-connectivity is $O(k^2 n \log(n/k))$ words.

Proof. The only difference from the proof of Claim 42 is that the proof for vertices in T_0 takes size at most $O(kn)$ since each path can be of size at most n. This happens because edge-disjoint paths can use the same vertices. This gives us an overall proof size of $O(k^2 n \log(n/k))$ words. ◁

If we can show that vertices T_0 have k edge-disjoint paths to t in smaller space then we can improve the overall proof size (since this is the main bottleneck). For instance, if the graph has a subgraph on $O(k)$ vertices that is k-edge-connected then that set could be T_0 and the overall proof size would be $O(kn \log(n/k))$ (this would increase the verification space by an additive $O(k^2)$ words).

Proof of Lemmas 39 and 40. We have proved the correctness and the size bounds in expectation. There must be some random string that achieves a proof size of at most the expected size (by definition) implying that there exists a proof of size $O(kn \log(n/k))$ words for k-vertex-connectivity and a proof of size $O(k^2 n \log(n/k))$ words for k-edge-connectivity (which an all powerful prover can find). Thus, the prover sends this proof to the verifier. ◀

We now show that the prover can send some auxiliary information so that the verifier can verify the proof.

▷ Claim 44. The layering proof of Lemma 39 for k-vertex-connectivity can be verified using a $[kn + h, v]$-scheme for any h, v such that $h \cdot v = n^2$.

Proof. The prover sends k vertex-disjoint paths for each vertex using the layering idea mentioned above (Lemma 39). There are three things that need to be verified. First, all the edges that the prover sends should be a part of the input graph. Second, the edges sent by the prover should form paths. Finally, the paths for a vertex should be vertex disjoint.

The prover sends a list of all edges used in the proof along with their multiplicities in $\widetilde{O}(kn)$ space. It is easy to do a subset check for the edges (ignoring multiplicities) using an $[h, v]$-scheme since there are at most n^2 edges (Fact 24). The problem now is to check if the prover was truthful about the multiplicities, which can be done using the ℓ_0-sampler trick using $\widetilde{O}(1)$ space (Fact 26). The verifier can maintain an ℓ_0-sampler where he inserts all edges along with their multiplicities when they are provided upfront. He then deletes all edges used in the proof of k vertex-disjoint paths for each vertex. Finally, he checks whether the sampler is empty. If it is not then the two sets of edges are not the same and thus, the verifier catches the lying prover. The probability of failure is at most $1 - 1/\text{poly(n)}$.

The prover will send the edges of the paths one path at a time, so it is easy to verify that the edges form a path. Finally, we need to verify that the paths for every vertex are vertex disjoint. For this, the prover sends the vertices he is going to use in the proof upfront in increasing order. The verifier can verify this in $\widetilde{O}(1)$ space by just checking the increasing order. The problem now is to check if the prover lied so we can run an equality check on the stream of vertices sent upfront and the vertices used in the proof. This can be done using the ℓ_0-sampler trick in $\widetilde{O}(1)$ verification space (Fact 26). We have to do this for all vertices so the worry is that the proof size or verification space might blow up. Since the auxiliary information is just repeating the vertices used in the proof in sorted order, adding

this auxiliary information can at most double the space used. Also, the space on the verifier side is $\widetilde{O}(1)$ words for each vertex, but this space can be reused implying $\widetilde{O}(1)$ words of verification space for the entire disjointness check.

Therefore, the proof along with the auxiliary information takes size $\widetilde{O}(kn) + \widetilde{O}(h)$ bits and can be verified in $\widetilde{O}(v) + \widetilde{O}(1)$ bits of space, proving the $[kn + h, v]$-scheme. ◁

▷ Claim 45. The layering proof of Lemma 40 for edge-connectivity can be verified using a $[k^2n + h, v]$-scheme for any h, v such that $h \cdot v = n^2$.

The proof is almost identical with the only difference being that the proof for k-edge-connectivity is of size $\widetilde{O}(k^2n)$ opposed to $\widetilde{O}(kn)$ for k-vertex-connectivity.

This establishes Theorem 9.

▶ **Theorem 9.** *Under dynamic graph streams on n-node graphs, there exists a $[k \cdot (h + kn), v]$-scheme for k-vertex-connectivity for any h, v such that $h \cdot v = n^2$. In particular, under such streams, the problem has non-trivial annotated streaming schemes with total cost $\widetilde{O}(k^2n)$.*

References

1. Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. In *Proc. 40th Annual ACM Symposium on the Theory of Computing*, pages 731–740, 2008.
2. Amirali Abdullah, Samira Daruki, Chitradeep Dutta Roy, and Suresh Venkatasubramanian. Streaming verification of graph properties. In *Proc. 27th International Symposium on Algorithms and Computation*, pages 3:1–3:14, 2016.
3. Farid Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science*, 175(2):139–159, 1996.
4. Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 459–467, 2012.
5. Sepehr Assadi and Vihan Shah. Tight bounds for vertex connectivity in dynamic streams. In *Symposium on Simplicity in Algorithms (SOSA)*, pages 213–227. SIAM, 2023.
6. László Babai, Péter Frankl, and Janos Simon. Complexity classes in communication complexity theory. In *Proc. 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 337–347, 1986.
7. Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 687–706, 2014.
8. Amit Chakrabarti, Graham Cormode, and Andrew McGregor. Annotations in data streams. In *Proc. 36th International Colloquium on Automata, Languages and Programming*, pages 222–234, 2009.
9. Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. Annotations in data streams. *ACM Trans. Alg.*, 11(1):Article 7, 2014.
10. Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and Arthur-Merlin communication. In *Proc. 30th Annual IEEE Conference on Computational Complexity*, pages 217–243, 2015.
11. Amit Chakrabarti and Prantar Ghosh. Streaming verification of graph computations via graph structure. In *Proc. 33rd International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 70:1–70:20, 2019.
12. Amit Chakrabarti, Prantar Ghosh, and Justin Thaler. Streaming verification for graph problems: Optimal tradeoffs and nonlinear sketches. *To appear in RANDOM*, 2020. `arXiv:2007.03039`.
13. Lijie Chen. On the hardness of approximate and exact (bichromatic) maximum inner product. *Theory Comput.*, 16:1–50, 2020. `doi:10.4086/toc.2020.v016a004`.

14 Graham Cormode, Marcel Dall'Agnol, Tom Gur, and Chris Hickey. Streaming zero-knowledge proofs. *CoRR*, abs/2301.02161, 2023. `arXiv:2301.02161`.

15 Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *Algorithmica*, 65(2):409–442, 2013.

16 Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proc. VLDB Endowment*, 5(1):25–36, 2011.

17 Dmitry Gavinsky. The layer complexity of arthur-merlin-like communication. *Theory Comput.*, 17:1–28, 2021.

18 Prantar Ghosh. New verification schemes for frequency-based functions on data streams. In *40th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2020, December 14-18, 2020, BITS Pilani, K K Birla Goa Campus, Goa, India (Virtual Conference)*, volume 182 of *LIPIcs*, pages 22:1–22:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

19 Sudipto Guha, Andrew McGregor, and David Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 241–247, 2015.

20 Tom Gur and Ran Raz. Arthur–Merlin streaming complexity. In *Proc. 40th International Colloquium on Automata, Languages and Programming*, pages 528–539, 2013.

21 T. S. Jayram and David P. Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Trans. Algorithms*, 9(3):26:1–26:17, 2013.

22 Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Proc. 18th Annual IEEE Conference on Computational Complexity*, pages 118–134, 2003.

23 Hartmut Klauck and Ved Prakash. Streaming computations with a loquacious prover. In *Proc. 4th Conference on Innovations in Theoretical Computer Science*, pages 305–320, 2013.

24 Hartmut Klauck and Ved Prakash. An improved interactive streaming algorithm for the distinct elements problem. In *Automata, Languages, and Programming - 41st International Colloquium (ICALP)*, volume 8572 of *LNCS*, pages 919–930, 2014.

25 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, Cambridge, 1997.

26 Aviad Rubinstein, Tselil Schramm, and S. Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science, ITCS 2018*, Leibniz International Proceedings in Informatics, LIPIcs, Germany, January 2018. Schloss Dagstuhl – Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing. `doi:10.4230/LIPIcs.ITCS.2018.39`.

27 Xiaoming Sun and David P Woodruff. Tight bounds for graph problems in insertion streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2015)*. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2015.

28 Justin Thaler. Data stream verification. In *Encyclopedia of Algorithms*, pages 494–499. Springer Berlin Heidelberg, 2016.

29 Justin Thaler. Semi-streaming algorithms for annotated graph streams. In *Proc. 43rd International Colloquium on Automata, Languages and Programming*, pages 59:1–59:14, 2016.