Threshold-Driven Streaming Graph: Expansion and Rumor Spreading

Flora Angileri
University of Rome Tor Vergata, Italy

Andrea Clementi
University of Rome Tor Vergata, Italy

Emanuele Natale
CNRS, I3S & INRIA, Université Côte d'Azur, Sophia Antipolis, France

Michele Salvi
University of Rome Tor Vergata, Italy

Isabella Ziccardi
CNRS, IRIF, Université Paris Cité, France

Abstract

A randomized distributed algorithm called RAES was introduced in [11] to extract a bounded-degree expander from a dense n-vertex expander graph $G = (V, E)$. The algorithm relies on a simple threshold-based procedure. A key assumption in [11] is that the input graph G is static – i.e., both its vertex set V and edge set E remain unchanged throughout the process – while the analysis of RAES in dynamic models is left as a major open question.

In this work, we investigate the behavior of RAES under a dynamic graph model induced by a *streaming node-churn process* (also known as the *sliding window model*), where, at each discrete round, a new node joins the graph and the oldest node departs. This process yields a bounded-degree dynamic graph $\mathcal{G} = \{G_t = (V_t, E_t) : t \in \mathbb{N}\}$ that captures essential characteristics of peer-to-peer networks – specifically, node churn and threshold on the number of connections each node can manage. We prove that every snapshot G_t in the dynamic graph sequence has good expansion properties with high probability. Furthermore, we leverage this property to establish a logarithmic upper bound on the completion time of the well-known PUSH and PULL rumor spreading protocols over the dynamic graph $\mathcal{G}$.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms; Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases Distributed Algorithms, Randomized Algorithms, Dynamic Random Graphs, Graph Expansion, Rumor Spreading

Digital Object Identifier 10.4230/LIPIcs.STACS.2026.6

Related Version *Full Version*: `https://arxiv.org/abs/2507.23533` [5]

Funding *Andrea Clementi*: Supported by Spoke 1 "FutureHPC & BigData" of ICSC – MUR Missione 4 Componente 2 Investimento 1.4 – Next Generation EU (NGEU).
Emanuele Natale: Supported by the French government, through the France 2030 investment plan managed by the Agence Nationale de la Recherche, as part of the "UCA DS4H" project, reference ANR-17-EURE-0004. Part of this work was carried out while E.N. was visiting the University of Rome Tor Vergata ("Bando Visiting 2024'").
Michele Salvi: Supported by the MUR Excellence Department Project MatMod@TOV, awarded to the Department of Mathematics, University of Rome Tor Vergata, CUP E83C18000100006, and by the MUR 2022 PRIN project GRAFIA, project code 202284Z9E4. M.S. is also part of the INdAM group GNAMPA.
Isabella Ziccardi: Supported by the European QuantERA project QOPT (ERA-NET Cofund 2022-25) and the French PEPR integrated project EPiQ (ANR-22-PETQ-0007).

© Flora Angileri, Andrea Clementi, Emanuele Natale, Michele Salvi, and Isabella Ziccardi;
licensed under Creative Commons License CC-BY 4.0

43rd International Symposium on Theoretical Aspects of Computer Science (STACS 2026).
Editors: Meena Mahajan, Florin Manea, Annabelle McIver, and Nguyễn Kim Thắng
Article No. 6; pp. 6:1–6:21

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Acknowledgements The authors want to thank Francesco Pasquale for helpful discussions and suggestions on the preliminary version of this work.

1 Introduction

In [11], the authors proposed a simple, lightweight distributed algorithm, working on any synchronous communication model, that extracts an n-vertex sparse expander subgraph from any n-vertex dense expander graph G. This task, in different versions, has been the subject of a strong research activity [3, 18, 9, 29, 38, 33]. The algorithm, called RAES,[1] is governed by two parameters $c, d \in \mathbb{N}$ that essentially determine a constant threshold on the maximum vertex degree, and it can be informally described as follows. Initially, each vertex has no incident *links*. In each round, every vertex v performs two consecutive actions. In a first request phase, v samples a set of random neighbors from the underlying graph G, selecting enough candidates to potentially establish d *outgoing* links. It then sends a link request to each of these sampled neighbors. In a second acceptance phase, each vertex, upon receiving requests, accepts or rejects them based on a threshold rule. Specifically, it accepts all incoming requests from the current round unless doing so would result in more than cd total *incoming* links. If that limit is exceeded, it rejects all requests received in that round. The process repeats until every vertex has exactly d established outgoing links, at which point the algorithm terminates and no further requests are made. Informally, in [11] it is shown that, if the underlying graph G from which each vertex selects its random neighbors is sufficiently dense[2] and has good expansion properties, then RAES has $O(\log n)$ completion time and the subgraph determined by all the accepted links[3] is a good sparse expander, with high probability.[4]

The setting considered in [11] is static: both the set of vertices and the underlying dense graph remain unchanged throughout the process. The work [11] in fact leaves the analysis of RAES in dynamic models as a major open question. This is motivated by the fact that modern network scenarios, such as peer-to-peer networks [8, 41, 43] and opportunistic networks [14], are inherently dynamic, with nodes and links changing over time, sometimes at a relatively-high rate.

In more recent studies [12, 13], a different version of RAES is presented and analyzed over a dynamic setting where vertices may enter and leave the system according to the *streaming node-churn process*[5]. Despite its simplicity, this streaming model has been shown to be predictive for other, more realistic dynamic-graph models (see [12]) and, moreover, its rigorous analysis requires coping with challenging technical issues, as shown in [12, 20] and for other graph-connectivity problems in [21]. In this streaming model, starting from an empty vertex set V_0, at each round, a new vertex v joins the network and selects d random neighbors. Then, after n rounds, v leaves the network and all its incident edges are removed. Notice that this process implies that every vertex stays in the system for exactly n rounds and, after an initial time window of n rounds, the number of alive vertices $|V_t|$ at every round t is always n. During its life, a vertex v can thus see one of its incident link disappear because one of its neighbors is the oldest one and leaves the network: in that case, v immediately

[1] Standing for "Request a link, then Accept if Enough Space".
[2] In particular, if the edge set has size $\Omega(n^2)$.
[3] In the final random subgraph produced by RAES, both outgoing links and incoming ones are considered undirected.
[4] An event E holds *with high probability* (for short, w.h.p.) if $\mathbf{Pr}[E] \geqslant 1 - n^{-\gamma}$ for some constant $\gamma > 0$, with respect to some input parameter n.
[5] They also considered other node-churn processes.

replaces it with a new random link. We also observe that the time duration of a round in this model needs to be suitably set to (only) allow a direct, 1-hop communication of small messages: it is thus reasonable to assume that, in that small period, the node set gets only very-small changes.

We remark that the dynamic version considered in [13] does not implement the second action of the original algorithm RAES: every link request is accepted by every destination vertex at any round of the process. The absence of this second action clearly implies that the maximum vertex degree of the resulting dynamic graph is not bounded. Indeed, a standard balls-into-bins argument shows that the maximum degree is $\Theta(\log n / \log \log n)$, w.h.p. (see for instance [40]). In the most relevant network scenarios that inspired our algorithmic study, namely peer-to-peer networks such as the bitcoin network [10, 41], the presence of an unbounded number of links managed by a single vertex may lead to serious efficiency and security problems [1, 22]. Indeed, the standard protocol of the bitcoin network [22, 41] imposes a threshold on the number of active links each vertex can manage, thus an action similar to the second one of the original RAES algorithm proposed in [11]. For further discussion of this issue and other related works see [5, Section 7].

A further motivation for maintaining dynamic bounded-degree expanders lies in the opportunity to adopt broadcast protocols, such as PUSH and PULL ones, to get fast and communication-efficient *rumor spreading* [15, 23, 17].

As we discuss in the next subsection, our goal is to study the dynamic graph generated by the original version of RAES combined with the streaming node-churn model.

1.1 Our contribution

Setting the dynamic-graph process. We aim to analyze a dynamic graph model that simultaneously captures two key features of modern peer-to-peer networks: a local threshold mechanism that bounds the degree of each vertex, and a node-churn process that regulates how vertices join and leave the network in each round. We kept all other modeling choices as simple and natural as possible, using the fewest parameters necessary. While this setting does not capture all aspects of real dynamic networks (such as the Bitcoin one), we believe that this approach can still recover qualitative properties and phenomena yielded by the simultaneous presence of the two features above, and that it can be robust to variations or extensions of the model's complexity.

We introduce the *Threshold-driven Streaming Graph* model, abbreviated as $\mathcal{TSG}(n, d, c)$, which is obtained by combining the two processes described above: (i) the streaming node-churn model [13, 19], and (ii) the original RAES protocol in [11] (see Definition 4 and Definition 5 for its formal definition). We first notice that, in every round $t \geqslant 0$, the degree of each vertex v is always bounded by the threshold $(c+1)d$: in particular, at most d edges are generated by the requests sent by v and at most cd edges are due to the online requests received by v.

Consistently with other models of dynamic graphs with node churn [7, 6, 33, 38], we assume the presence of a *link manager* to apply the RAES's connection-request strategy: any vertex that makes a link request can access this entity and get a random destination vertex. Importantly enough, the role of the link manager we assume here is minimal: vertices cannot get any further information from it.[6] As we will elaborate later in this section, the total number of calls each vertex performs to the link manager is a key performance measure of the system and the RAES's strategy optimizes it.

[6] For instance, one vertex might ask the current degree of the selected destination or, even more, information about the current topology: this is not allowed.

As we will discuss later in Section 2, the $\mathcal{TSG}(n, d, c)$ model yields a complex stochastic process of graph snapshots $\mathcal{G} = \{G_t = (V_t, E_t) : t \in \mathbb{N}\}$, where edges in E_t are neither uniformly distributed nor mutually independent. Hence, the analysis of the key aspects, such as the expansion properties of the graph snapshots, requires coping with new technical issues that are likely to emerge in other, more realistic models as well.

Expansion properties. Even though the node churn and the RAES rules are simple in themselves, their combination, yielding the $\mathcal{TSG}$ dynamic graph, turns out to be rather complex, essentially because it generates both a non-uniform link distribution and induces subtle correlations between the links of every snapshot of the dynamic graph. Informally, on the one hand older vertices tend to have a higher degree than younger ones. On the other hand, the fact that connection requests might create conflicts with other requests and get rejected several times along their life generates non trivial correlations among the links that are active in a given graph snapshot, even if they have been established in different previous rounds.

Our analysis solves the above technical challenges and essentially limits the maximum (i.e. worst-case) correlation lying among any subset of links of the same snapshot (see Section 2 for an overview of this key technical part). We then use such limited correlation among edges to prove that the $\mathcal{TSG}(n, d, c)$ model generates graph snapshots having the following good expansion properties.

▶ **Theorem 1** (Expansion Properties). *There exist constants c, d and β sufficiently large such that, for all n large enough, and any round $t \geqslant 2n$, the snapshot G_t generated by $\mathcal{TSG}(n, d, c)$ has the following properties w.h.p.:*

(a) *There exists an induced expander subgraph in G_t with $n - O(\log n)$ nodes;*

(b) *Any subset of vertices of size at least $\beta \log n$ has constant conductance.*[7]

We observe that the above result is tight in the following sense. It is easy to see that a new incoming vertex may stay isolated for the first $o(\log n)$ rounds of its life with non negligible probability: then, it is clear that, at any round, there may be some vertex subset of $o(\log n)$ size having bad expansion.

Rumor spreading. *Rumor Spreading* refers to a class of simple epidemic protocols that, given a source vertex s holding a piece of information (i.e. the *rumor*), aim to broadcast this information to all vertices of the graph. The basic, popular randomized variants of rumor spreading are the (synchronous) uniform PUSH protocol and the PULL protocol: in the former, at each round every informed node (i.e., every node that learned the rumor in a previous round) chooses a neighbor uniformly at random and sends the rumor to it. In PULL, at each round, every uninformed node chooses a random neighbor; if that neighbor is informed, it sends the rumor to the uninformed node. Finally, the PUSH-PULL protocol combines both strategies above to inform new, uninformed nodes.

PUSH and PULL protocols have been shown to be effective in many networks applications [23, 34, 44], and, very importantly for our setting, they have been proved to be fault-tolerant [27, 28] and efficient even in some model of evolving graphs [16, 17, 26, 30]. A key question concerns the completion time, i.e., how many rounds such protocols take to broadcast the source information to all nodes in the graph [15, 37].

[7] For a definition of conductance see (2).

While *flooding* has been analyzed even on dynamic graphs that include node-churn [6, 13], to the best of our knowledge, no analytical results are known for any rumor-spreading protocol. As a further contribution, we study the completion time of the uniform PUSH and PULL over the $\mathcal{TSG}$ model and, exploiting Theorem 1, we prove the following bound.

▶ **Corollary 2.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. Let s be a source node joining the $\mathcal{TSG}(n, d, c)$ dynamic graph at some round $t_s \geqslant 2n$. Then, after $T = O(\log n)$ rounds, the* PUSH *or the* PULL *protocol inform at least $n - O(\log n)$ vertices in G_{T+t_s}, w.h.p.*

Also the result of Corollary 2 is tight for the same reasons of Theorem 1: with non-negligible probability a new incoming vertex may stay isolated for the first $o(\log n)$ rounds and hence it cannot receive the source information. Then, it is clear that, at any round, there may be some subset of size $o(\log n)$ with vertices that are not informed.

Communication complexity of $\mathcal{TSG}(n, d, c)$. *Message-communication overhead* is a crucial performance parameter in communication networks since it has a strong impact on node traffic congestion and on the time delay of fundamental tasks such as broadcast and consensus [1, 6, 22]. In the $\mathcal{TSG}(n, d, c)$ model the only messages exchanged by vertices are those determined by the pending link requests: the overall number of exchanged messages at every round t is optimal in expectation and $O(\log n)$, w.h.p. Indeed, we prove that, at every round $t \geqslant 0$, the overall number of calls to the link manager performed by the vertices in V_t (i.e. the overall number of *pending requests* at round t) has constant expectation and is $O(\log n)$, w.h.p. We also show that the overall number of calls (i.e. the *work*) each vertex makes during all of its life has constant expectation and it is $O(\log n)$, w.h.p., as well. These results are easy consequences of Lemma 9 and Lemma 11.

1.2 Previous work

We already discussed the results in [11, 12, 13] that motivated our work. Due to space constraints, we discuss here only two other well-studied approaches that implemented the link-manager function in a distributed way. We refer to [5, Section 7] for an overview of further related literature.

The use of a centralized link manager, able to provide a list of possible neighbors whenever a node requires it, has been widely adopted (see for example [11, 25, 42]). A parallel line of research providing instead a partially-distributed implementation of the link-manager function is the one employing *ID-based random walks [19, 31, 33, 38, 39].* Differently from RAES, this approach requires the access to a centralized link manager only when the nodes join the network: after that, they periodically refresh their neighborhood based on the results of random walks running continuously in the background. This approach is less prone to a possible malicious behavior of the centralized link manager (typically represented by DNS Servers) and thus it can be more resilient and safe in some scenarios [33]. On the other hand, as remarked in the recent work [32], the use of random walks has two crucial limitations: (i) they have a high communication complexity, since the overall message overhead in every round is $\Theta(n\text{polylog}(n))$ [19, 36, 39] (recall that RAES has a message overhead of $\Theta(\log n)$); (ii) they tend to favor nodes with higher in-degrees, which compromises their ability to provide a uniform sample [39] (which is instead granted in the RAES protocol).

A further class of distributed algorithms partially overcoming these two weaknesses is the one using the so-called *gossip-based approach* [32, 36], see also [2] for a version including an underlying node-churn dynamics. In these algorithms, each node holds a list of possible future neighbors and exchanges a part of this list at each interval of time with its current neighbors. As proved in [32], this approach is more efficient than the ID-random walk based one, and, importantly enough, after a suitable amount of time it generates neighbor lists that are distributed almost uniformly over all possible nodes of the network. For this reason, our results on the $\mathcal{TSG}(n, d, c)$ model, which imposes a uniform distribution for the formation of future links, may also be interpreted as a first, important step towards a theoretical analysis of the expansion properties of a dynamic graph model with node churn where the link manager is implemented via the distributed, gossip-based approach.

1.3 Roadmap

The rest of the paper is organized as follows. In Section 2, we overview the main technical challenges and the key ideas we introduce to face them. In Section 3.1 we first of all give a more formal definition of the $\mathcal{TSG}$ model. Then, we proceed with some technical lemmas in Section 3.2. In particular, we state the fundamental result on the link distribution generated by the $\mathcal{TSG}$ model, Lemma 8, bounding the maximal correlation among multiple links of any graph snapshot. The lemma is followed by a sketch of its proof, while the latter is deferred to the full version of this work [5]. In Section 4 we describe how to use the results of Section 3.2 to prove the expansion properties stated in Theorem 1. Then, Section 5 is devoted to the proof of the rumor spreading result, namely Corollary 2. Finally, in Section 6, we discuss some open questions.

2 Technical Analysis: An Overview

As we already remarked in Section 1, our analysis requires to cope with two main technical challenges, each one already faced in two previous works [11] and [13] that analyze two different variants of RAES. Unlike those prior works, in which only one of the two challenges is considered, our setting requires to confront both simultaneously, significantly increasing the complexity of the analysis.

The first challenge, addressed in [11], arises from the second action of RAES, which involves the threshold-based conditional acceptance rule of link requests. This mechanism introduces correlations among the random destinations of the accepted links: to see just one source of this correlation, consider the fact the acceptance of a link implies that the target node did not receive more than cd requests in the current round. In the static setting, [11] addresses this issue using a sophisticated compression argument to prove the expansion properties of the resulting graph. Essentially, while powerful, this technique lacks the flexibility to include the presence of the second challenge: the node churn and the dynamic link regeneration at every round.

The second challenge thus arises from the presence of the streaming node churn: this issue is faced in [13], where a simplified version of the RAES algorithm is considered. In [13], vertices accept all incoming requests unconditionally, eliminating the threshold mechanism. This simplification avoids the correlation issues seen in the static case, allowing the authors to sidestep the compression argument. Their proof relies on a key lemma establishing that the random destinations of the link requests follow an almost-uniform distribution; this property is then exploited to get good expansion properties of the resulting graph snapshots. A major issue in their dynamic model is handling correlations due to node churn, especially proving that nodes with similar ages do not generate dense clusters. On the other hand,

their key lemma may focus on the distribution of the destination of a *single* link request: this is enough since, in the absence of the threshold mechanism, link destinations always remain mutually independent and their joint distribution is just a product. In contrast, in our model, the threshold-based acceptance rule introduces dependencies among edge destinations: we thus have to cope with both potential node clustering *and* the mutual correlation among link destinations.

We address these issues by extending the approach of the key lemma from [13]. Specifically, our Lemma 8 shows that, not only the destination of a single link destination is almost uniform (similarly to [13]), but also demonstrates that the *joint distribution* of the destinations of any subset of links can be effectively expressed as a product distribution, up to a constant factor. The proof of Lemma 8 represents the main technical contribution of our work: an overview is given in Section 3.2, while its full version is available in the extended version of this paper [5]. We believe that our technique can also be adapted to more complicated versions of node churn, as the Poisson node churn considered in different papers [42, 13].

Another technical challenge that lies behind all our proofs is the control of the number of pending requests at every round. This boils down to a queuing theory problem: thanks to the method of bounded differences, we can show that the process $(Q_t)_{t\in\mathbb{N}}$ of the number of pending requests can be stochastically dominated by a Markov process that has a strong negative bias for high values of Q_t (see [5, Lemma 4.3]). This ensures that the queue of pending requests is $O(\log n)$ with high probability (Lemma 9). We also show in Lemma 11 that the probability that a request is pending for more than j rounds during its life decays exponentially, guaranteeing a minimal number of requests to the link manager and, thus, a minimal workload per node.

Given our key Lemma 8 and the control of the pending requests queue, the proof of the good expansion properties of the dynamic graph become more standard, albeit suitable adaptations of the techniques of [13, 11] are needed in our framework.

Finally, the expansion properties of the dynamic graph and the fact that our model allows by its nature only vertices of bounded degree would make the results of Corollary 2 a simple consequence of the classic analysis of rumor spreading in [15]. The only novelty here is the analysis of the initial *bootstrap* process, see Lemma 17. The bootstrap of the information-spreading process is essentially the initial, random time phase the protocol requires to reach a logarithmic number of informed nodes: we need this further analysis since Theorem 1 does not guarantee worst-case good expansion for subsets of informed vertices of size $o(\log n)$. Informally, for this phase, we use Claim (b) of our Theorem 1 to prove that, when joining the graph, the source has high probability to fall into a connected component of size $\Omega(\log n)$ and, moreover, this component will be stable for at least $\Theta(\log^2 n)$ rounds. This is enough to get $\Theta(\log n)$ number of informed nodes after a logarithmic number of rounds after the source joined the graph. As remarked above, once the set of informed nodes achieves a logarithmic size, we can combine Claim (a) of Theorem 12 with the previous classic analysis of rumor spreading in [15] to get Corollary 2.

3 Preliminaries

A *dynamic graph* $\mathcal{G}$ is an infinite sequence of graphs $\mathcal{G} = \{G_t = (V_t, E_t) : t \in \mathbb{N}\}$. If $\{V_t\}_t$ or $\{E_t\}_t$ are sequences of random sets, we call the corresponding random process a *dynamic random graph*, and G_t denotes the *snapshot* of the dynamic graph at *round* t. As usual, the size of any subset A is denoted as $|A|$. The *outer boundary* of a set of vertices S is defined as

$$\Gamma_t(S) = \{v \in V_t \setminus S \mid \exists u \in S \text{ s.t. } \{u, v\} \in E_t\}.$$

Our analysis of dynamic graphs considers the fundamental notions of *conductance* of a graph [35]. For any two set of vertices $S, T \subseteq V_t$, $E_t(S,T)$ denotes the set of edges crossing (S,T) at round t, that is $E_t(S,T) = \{\{u,v\} \in E_t : u \in S, v \in T\}$, while $\partial_t S = E_t(S, V_t \setminus S)$ denotes the set of edges crossing $(S, V_t \setminus S)$. The *volume* of the set S is defined as $\mathrm{vol}_t(S) = |E_t(S,V_t)|$. Then, the *conductance* $\phi_t(S)$ of the set S at round t is defined as

$$\phi_t(S) = \frac{|\partial_t S|}{\min\{\mathrm{vol}_t(S), \mathrm{vol}_t(V_t \setminus S)\}}. \tag{1}$$

The conductance of the graph G_t is the minimum of $\phi_t(S)$ over all possible sets $S \subseteq V_t$ with volume smaller than the total number of edges:

$$\phi_t(G_t) = \min_{S \subseteq V_t} \phi_t(S). \tag{2}$$

Given any vertex subset S, $G_t[S]$ denotes the subgraph of G_t induced by S. We will omit the subscript t in all notations above when it is clear from the context.

▶ **Definition 3** (Graph Expansion). *An infinite family of graphs $\{G^{(n)}(V,E),\ \text{with } |V| = n\}_{n\in\mathbb{N}}$ is an* α-expander *if there exist constants $\alpha \in (0,1)$ and $n_0 \in \mathbb{N}$ such that $\phi(G^{(n)}) \geqslant \alpha$ for all $n \geqslant n_0$.*

3.1 The dynamic graph model

Our goal is to study the dynamic graph model determined by combining the streaming node-churn process [13] with the edge generation process defined by the distributed algorithm RAES (in [11]), based on a simple threshold rule. In what follows, we formalize this combined model and state some of its preliminary properties.

The vertex-set process $\{V_t\}_t$ of a dynamic graph $\mathcal{G}$ is typically called *node churn* [6, 13]. In this paper, we consider the deterministic *streaming* node churn of parameter n defined as follows.

▶ **Definition 4** (Streaming node churn). *Let $n \in \mathbb{N}$. A* streaming node churn *with n vertices is a deterministic process $\{V_t : t \in \mathbb{N}\}$ such that $V_0 = \emptyset$, and, for any $t \geqslant 1$, the set V_t is defined iteratively by the following simple rules:*

(a) *A new vertex v joins the vertices set;*

(b) *At round $t \geqslant n+1$, the vertex u that joined the set of vertices at time $t-n$, leaves the graph.*

Then, V_t is defined to be $V_t = V_{t-1} \cup \{v\} \setminus \{u\}$ when $t \geqslant n+1$ and $V_t = V_{t-1} \cup \{v\}$ for $t \leqslant n$. For a vertex $v \in V_t$, the age of v at time t is the function $\mathrm{age}_t(v) = t - t_v$, where $t_v \leqslant t$ is the round vertex v joined the vertex set.

Some easy but important remarks follow. The vertex v joining the graph at time t_v leaves the graph at round $t_v + n$, i.e. $v \in \cap_{s=t_v}^{t_v+n-1} V_s$ and $v \notin V_{t+n}$. We say that the streaming node churn $\{V_t : t \in \mathbb{N}\}$ with parameter n gets *stable* after round $t \geqslant 2n$: in particular, after that round, two properties hold that we will often (implicitly) use in the analysis of the process:

(i) The set V_t has size n;

(ii) The set V_{t-n} has size n: this implies that, at the round each vertex in V_t joined the graph, there were already n vertices present in the graph.

In order to define our dynamic graph model $\mathcal{G}$, we need also to specify the evolution of the edge set $\{E_t\}_t$. We consider a random process $\{E_t\}_t$ determined by the simple rules of the RAES algorithm we described in Section 1. According to peer-to-peer models (where

vertices make *connection requests* to other nodes), we distinguish between *outgoing* edges from a vertex v, originating from a connection request made by v, and *incoming* edges to v, resulting from a connection request made by another vertex to v. However, we remark that the resulting graph snapshots $G_t = (V_t, E_t)$ are *undirected*: once established, every edge in E_t allows message communication in both directions.

▶ **Definition 5** (Edge process). *Let $c, d \in \mathbb{N}$ be two parameters, and let $\{V_t : t \in \mathbb{N}\}$ be the streaming node churn with $n \geqslant 2$ vertices introduced in Definition 4. The random subset sequence $\{E_t\}_t$ is defined inductively as follows. We set $E_0 = \emptyset$ and, for any $t \geqslant 1$, the subset E_t is generated according to the following rules:*[8]

(a) *E_t contains all the edges in $E_{t-1}(V_t, V_t)$, while all edges incident to the leaving vertex of age n are deleted;*

(b) *Each vertex $v \in V_t$ with less than d* outgoing *edges makes a new connection request for each one of its missing outgoing edges. Each request is sent to a destination vertex chosen independently and uniformly at random in $V_t \setminus \{v\}$.*[9]

(c) *Assume a vertex $u \in V_t$ receives $\ell \geqslant 1$ connection requests from other nodes. Then, it accepts all the requests and activates the corresponding edges if and only if it has* in-degree *$\leqslant c \cdot d - \ell$; otherwise, it rejects all the requests it received at round t.*

Informally, each vertex $v \in V_t$ of the dynamic graph tries to maintain its out-degree equal to d: we can think that v is equipped with d connection requests that it tries to keep connected to active vertices. However, if a request of v at time t lands to a vertex u which has a number of incoming edges and new connection requests larger than cd, the request of v is rejected and will not create an edge at round t (but it will try to connect again at the next round).

The dynamic graph $\mathcal{G}$ determined by the streaming node churn in Definition 4 and the edge process in Definition 5 will be called *Threshold-driven Streaming Graph* with parameters n, d, and c (for short $\mathcal{TSG}(n, d, c)$).

Full nodes, pending requests, and other key random variables. We now introduce the key notions and quantities we will consider in the probabilistic analysis of the $\mathcal{TSG}(n, d, c)$ model.

A vertex with cd incoming edges is called *full* and the set of full vertices at round t is denoted as B_t. Each request at round t is a pair $r = (v, i)$, where $v \in V_t$ is the vertex making the request and $i \in [d]$ is its index. For any vertex $v \in V_t$ (or any request $r \in V_t \times [d]$), we will denote with t_v (resp. t_r) the first round in which v (resp. r) appears in the dynamic graph. If a request r is trying to connect to some vertex u, we say that r *targets* the vertex u.

We observe that, at any round, there are *pending* requests. A connection request r from a vertex v is called pending at round t if either v has just joined the set of vertices V_t, or if r has been rejected in round $t-1$, or if r was connected at round $t-1$ to the node u that leaves the network at round t. Such a request generates an edge in E_t if and only if it is accepted by its target vertex at round t. Notice that, when a vertex joins the graph at time t, all its d requests are pending at time t.

[8] Essentially, each round t is organized in two consecutive phases: in the first one, the node churn action is applied to V_{t-1} thus getting V_t, while, in the second phase, the edge process works on the new vertex subset V_t.

[9] Notice that this rule implies that the new node, when it joins the graph, will make exactly d connection requests.

The *queue at round* t is the random set Q_t of all pending requests at round t. As we will see in the next sections, the queue plays a key role in our analysis. Moreover, by the definition of the $\mathcal{TSG}(n, d, c)$ model, the size $|Q_t|$ of the queue bounds the overall number of messages exchanged by the vertices at round t.

▷ Claim 6. For any $t \geqslant 1$, the *overall number of messages* performed by the dynamic graph $\mathcal{TSG}$ at round t is $O(|Q_t|)$.

For any $t \geqslant 1$ and any request $r \in V_t \times [d]$, the random variable $X_t(r)$ is defined as the destination of the request r if r is accepted (and thus generates an edge in E_t), while we set $X_t(r) = \emptyset$ if the request r was rejected at round t.

For any set S, denote with $r \xrightarrow{t} S$ the event indicating that the request r established a connection with a vertex in the set S at round t: in other words, that the request r is pending at round t, targets a vertex in the set S and it is accepted.

On the number of full vertices. Using a simple combinatorial argument, we next prove that the size of the set B_t of full vertices (i.e. vertices with in-degree equal to cd) at round t can never exceed a suitable threshold.

▷ Claim 7. For any $t \geqslant 1$, $|B_t| \leqslant \frac{n}{c}$.

Proof. For each $t \geqslant 1$, it holds $|E_t| \leqslant nd$, since each vertex has at most d outgoing edges. Assume, by contradiction, that $|B_t| > \frac{n}{c}$. Then, since each vertex in B_t has in-degree cd, this implies that $|E_t| \geqslant cd|B_t| > cd \cdot \frac{n}{c} = nd$, contradicting the fact that $|E_t| \leqslant nd$. ◁

3.2 Key Lemmas

In this section we provide an analysis of the stochastic process generated by the $\mathcal{TSG}$ model. This analysis allows us to establish some key results that will be then used to derive the expansion properties claimed in Theorem 1 and the logarithmic bound on the completion time of the PUSH and PULL protocols in Corollary 2.

On the edge probability distribution. To analyze the expansion properties of the $\mathcal{TSG}$ snapshots, we show that the link requests from any subset of nodes are both nearly uniformly distributed across the entire node set and nearly mutually independent. This result is the main technical contribution of the paper and is formalized in the following

▶ **Lemma 8.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. For every $t \geqslant 2n$, and for every $S \subseteq V_t$, $R \subseteq S \times [d]$ and $P \subseteq V_t$, we have*

$$\mathbf{Pr}\left[\cap_{r \in R}\{X_t(r) \in P\}\right] \leqslant \left(\frac{220|P|}{n-1}\right)^{|R|}.$$

The full proof of the above lemma can be found in [5, Lemma 4.1], while here below we provide a short overview of its main ideas.

Proof Overview of Lemma 8. Consider the snapshot $G_t = (V_t, E_t)$ at round $t \geqslant 2n$ and a set of link requests R. We want to control the probability that all requests in R established a link to some set P of vertices at round t. As a first step, we order the requests according to the last time they were accepted by some node of P. This way, we can telescopically

condition the probability that a single request $r \in R$ establishes a link with P at some time s on an event involving only connections happened in the past. For r to connect to P at time s two events must happen: r has to be pending at time s and the link manager has to point to some node of P at time s. Since we are conditioning only on the past, the probability of the second event is uniform over all nodes present at time s. As a byproduct, we are left to show that the (conditional) probability that r is pending at time s is small enough. The conditioning forces us to go through a (painful) worst-case scenario analysis. The key idea is the following: during its life each request goes through cycles (called $W_0, W_1, \dots$ in the proof) composed of two phases: a first phase where the request stays linked to a single vertex (until that vertex dies) and a second phase where the request is pending because it gets rejected before forming a new link. We show that, regardless of what happened in the past, the length of the first phase can be stochastically dominated from *below* by a suitable uniform random variable, while the length of the second phase can be stochastically dominated from *above* by a geometric random variable. The decomposition in cycles and the stochastic domination of the phases allow us to sandwich the event that r is pending during its f-th cycle between two events, called $S_1(f)$ and $S_2(f)$. As f varies, $S_1(f)$ and $S_2(f)$ form a partition of the space of events, allowing us to conclude. ◀

On the number of pending requests. As we observed in the previous section, the queue Q_t (i.e. the set of all pending requests at round t) plays a crucial role in our probabilistic analysis. In particular, we will often exploit the following upper bound on its size.

▶ **Lemma 9.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. For every $t \geqslant 2n$, $\mathbf{Pr}\left[|Q_t| \leqslant 100(cd)^2 \log n\right] \geqslant 1 - n^{-2}$.*

The proof of the lemma can be found in [5, Lemma 4.2], while we here highlight one of its key-ingredients that has a per se interest: informally, whenever Q_t reaches a logarithmic size, it will decrease by a constant factor, w.h.p.

▶ **Lemma 10.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. For any $t \geqslant 2n$, if $|Q_t| \geqslant 3 \cdot 32(cd)^2 \log n$, then*

$$\mathbf{Pr}\left[|Q_{t+1}| \leqslant \tfrac{1}{2}|Q_t| \mid Q_t, G_{t-1}\right] \geqslant 1 - n^{-3}. \tag{3}$$

On the number of pending rounds of a request. The following lemma provides a bound on the overall number of rounds in which a fixed request r is pending during all of its lifetime, namely on the quantity

$$P(r) = \sum_{t=t_r}^{t_r+n} \mathbb{1}\left[r \in Q_t\right].$$

▶ **Lemma 11.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. For any $t \geqslant 2n$, any request r in $V_t \times [d]$ verifies*

$$\mathbf{Pr}\left[P(r) \geqslant j\right] \leqslant 2\mathrm{e}^{-j/24}.$$

As a consequence $\mathbf{E}\left[P(r)\right] = O(1)$ and in particular $\mathbf{Pr}\left[P(r) \geqslant 50 \log n\right] \leqslant n^{-2}$.

The proof of the above lemma can be found in [5, Lemma 4.4].

4 Expansion Properties

In this section, we will prove the main result of this paper that we re-state here in a more formal way.

▶ **Theorem 12.** *Let $n_0, c_0, d_0 \in \mathbb{N}$ and $\alpha = \alpha(d)$ sufficiently large integers. Then, for any $d \geqslant d_0$, $c \geqslant c_0$ and $n \geqslant n_0$, an integer $\beta = \beta(c,d)$ exists, such that the snapshot $G_t = (V_t, E_t)$ generated by the $\mathcal{TSG}(n,d,c)$ model with $t \geqslant 2n$ satisfy the following properties, w.h.p.*
(a) *For every $S \subseteq V_t$ with $|S| \geqslant \beta \log n$ has conductance $\phi_t(S) \geqslant \alpha$;*
(b) *A subset $H_t \subseteq V_t$ with $|H_t| = n - O(\log n)$ exists such that $G_t[H_t]$ is an α-expander.*

The proof of Claim (a) is organized in two lemmas: The first one, Lemma 13, considers the vertex expansion of subsets of size in the range $\left[\beta \log n, \frac{n}{2000}\right]$, while the second one covers the remaining size range.[10] In both cases, our analysis will show a constant lower bound of $\varepsilon = \frac{1}{10}$ on the *vertex expansion* of the considered vertex subsets. However, since the graph snapshots in $\mathcal{TSG}(n,d,c)$ has bounded maximum degree (i.e. $\leqslant (c+1)d$), by definition of conductance (see Section 3), the latter will be at least $\varepsilon((c+1)d)^{-1} = \Omega(1)$. We recall that the *vertex expansion* of the graph G_t is defined as

$$h(G_t) = \min_{\substack{S \subseteq V_t: \\ |S| \leqslant \frac{n}{2}}} \frac{|\Gamma_t(S)|}{|S|}.$$

The proof of Claim (b) of the main theorem above will be provided in a third lemma, Lemma 15, and it also consists of analyzing the vertex-expansion of the considered subgraph.

4.1 Proof of Claim (a) of Theorem 12

The proof of the following lemma, proving the expansion of small subsets, relies on a standard application of our key Lemma 8 and it is provided in the full version of this paper [5, Lemma 5.2].

▶ **Lemma 13** (Expansion of small subsets)**.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. For any $t \geqslant 2n$ let E_t be the event*

$$E_t = \left\{ \min_{S \subseteq V_t : 2\beta \log n \leqslant |S| \leqslant \frac{n}{2000}} \frac{|\Gamma_t(S)|}{|S|} \geqslant \frac{1}{10} \right\}$$

where $\beta = 100(cd)^2$. Then $\mathbf{Pr}[E_t] \geqslant 1 - n^{-2}$.

The following lemma proves instead the expansion of the big subsets.

▶ **Lemma 14** (Expansion of big subsets)**.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. For any $t \geqslant 2n$ let E_t be the event*

$$E_t = \left\{ \min_{S \subseteq V_t : \frac{n}{2000} \leqslant |S| \leqslant \frac{n}{2}} \frac{|\Gamma_t(S)|}{|S|} \geqslant \frac{1}{10} \right\}.$$

Then $\mathbf{Pr}[E_t] \geqslant 1 - \mathrm{e}^{-n}$.

[10] The factor $\frac{1}{2000}$ has been set in order to simplify some calculations: the optimization of this parameters is out of the scope of our analysis.

Proof. Fix any subsets $S \subseteq V_t$ of size $\frac{n}{2000} \leqslant |S| \leqslant \frac{n}{2}$ and $T \subseteq V_t \setminus S$ such that $|T| = \lceil \frac{1}{10}|S| \rceil$ (from now on we will just suppose that $n/2000$ and $|S|/10$ are integers, for simplicity). Taking $P = S \cup T$ and $P^c = V_t \setminus P$, we have that

$$\mathbf{Pr}\left[\Gamma_t(S) \subseteq T\right] = \mathbf{Pr}\left[\cap_{r \in S \times [d]} \{X_t(r) \notin P^c\}\right] . \tag{4}$$

We note that for each $r \in S \times [d]$ it holds

$$\{X_t(r) \notin P^c\} \subseteq F_r(P^c) \tag{5}$$

where, calling $t_r \leqslant t$ the round when the request r joined the graph, for any $A \subseteq V_t$

$$F_r(A) = \{r \text{ did not establish a connection with a vertex in } A \text{ when it joined the graph}\} .$$

Indeed, if request r established a connection with some vertex of P^c when it entered the graph at time t_r, then it would still be connected to P^c at time $t \geqslant t_r$. Note that it is possible that not all vertices of P^c were already in the graph at time t_r.

For every vertex $a \in S$, consider now

$$\mathcal{O}_a = \{b \in P^c \mid \mathrm{age}_t(a) < \mathrm{age}_t(b)\}$$

the subset of vertices in $P^c \subseteq V_t$ that were in the graph when a joined it. Clearly, $F_r(P^c) = F_r(\mathcal{O}_{a(r)})$ if r is a request from vertex $a(r)$. In the rest of the proof we will abbreviate $a(r)$ as a. Then, from (4) and (5) we have

$$\mathbf{Pr}\left[\Gamma_t(S) \subseteq T\right] \leqslant \mathbf{Pr}\left[\cap_{r \in S \times [d]} F_r(\mathcal{O}_a)\right] . \tag{6}$$

Let $k = |S|$ and $\{a_1, \ldots, a_k\}$ be an age-based ordering of the vertices in S from the oldest to the youngest, so that $t_1 < \cdots < t_k$. We will analyze the r.h.s. of (6) by subsequentially conditioning on the events involving older vertices. We start by writing

$$\begin{aligned}&\mathbf{Pr}\left[\cap_{r \in S \times [d]} F_r(\mathcal{O}_a)\right] \\ &= \mathbf{Pr}\left[\cap_{j=1}^{d} F_{(a_k,j)}(\mathcal{O}_k) \middle| \cap_{i=1}^{k-1} \cap_{j=1}^{d} F_{(a_i,j)}(\mathcal{O}_i)\right] \mathbf{Pr}\left[\cap_{i=1}^{k-1} \cap_{j=1}^{d} F_{(a_i,j)}(\mathcal{O}_i)\right]\end{aligned} \tag{7}$$

where we abbreviated $\mathcal{O}_{a_i}$ as $\mathcal{O}_i$ to ease the notation. Let us focus on the conditional probability in the last expression. Recall that any fixed $r \in \{a_k\} \times [d]$ may fail to establish a connection with $\mathcal{O}_k$ at time t_r for two reasons: either because it targets a vertex outside of $\mathcal{O}_k$, or because it receives a rejection from the target vertex in $\mathcal{O}_k$. The first event occurs with probability $\frac{n-1-|\mathcal{O}_k|}{n-1}$ since the targets are chosen uniformly at random independently from the past. The second event happens if the targeted vertex is full at time t_r, or if the vertex targeted by r is also targeted by too many other requests in Q_{t_r}. As we are interested in the rejection of the d requests $\{(a_k, j),\ j \in [d]\}$, by the principle of deferred decision we can assume that all $r' \in Q_{t_r} \setminus \{(a_k, j),\ j \in [d]\}$ are sent before $\{(a_k, j),\ j \in [d]\}$. Now, if any $r \in \{(a_k, j),\ j \in [d]\}$ targets a vertex that has an in-degree of at most $(c-1)d$ *after all other requests in the queue are sent*, the attempt will certainly be accepted, independently from the other $r' \in \{(a_k, i),\ i \in [d]\} \setminus \{r\}$ and from what happened in the past. Therefore, if we call $\tilde{B}_{t_k}$ the set of vertices with load at least $(c-1)d$, at time t_k and *after all other requests in the queue are sent*, the probability of r being rejected is at most $\frac{|\mathcal{O}_k \cap \tilde{B}_{t_k}|}{n-1}$. Thus, we can conclude that

$$\mathbf{Pr}\left[\cap_{j=1}^{d} F_{(a_k,j)}(\mathcal{O}_k) \mid \cap_{i=1}^{k-1} \cap_{j=1}^{d} F_{(a_i,j)}(\mathcal{O}_i)\right] = \left(1 - \frac{|\mathcal{O}_k \cap \tilde{B}_{t_k}^c|}{n-1}\right)^d .$$

The same argument can be iteratively applied to $\mathbf{Pr}\left[\cap_{i=1}^{k}\cap_{j=1}^{d} F_{(a_i,j)}(\mathcal{O}_i)\right]$, isolating d requests per iteration, and it leads to

$$\mathbf{Pr}\left[\cap_{r\in S\times[d]}F_r(\mathcal{O}_a)\right] \leqslant \prod_{i=1}^{k}\left(1-\frac{|\mathcal{O}_i\cap\tilde{B}^c_{t_i}|}{n-1}\right)^d \overset{(I)}{\leqslant} \exp\left(-\frac{d}{n-1}\sum_{i=1}^{k}\left|\mathcal{O}_i\cap\tilde{B}^c_{t_i}\right|\right) \tag{8}$$

where inequality (I) follows since $1+x\leqslant \mathrm{e}^x$.

Now, if we look at the set of possible pairs $(a,b)\in S\times P^c$, two cases may arise:

(i) $\left|\{(a,b)\in S\times P^c \mid \mathrm{age}_t(a)<\mathrm{age}_t(b)\}\right| \geqslant \frac{|S|\cdot|P^c|}{2}$,

(ii) $\left|\{(a,b)\in S\times P^c \mid \mathrm{age}_t(a)>\mathrm{age}_t(b)\}\right| \geqslant \frac{|S|\cdot|P^c|}{2}$.

If Case (i) holds, then

$$\sum_{a\in S}|\mathcal{O}_a| = \sum_{a\in S}|\mathcal{O}_a\cap\tilde{B}_{t_a}| + |\mathcal{O}_a\cap\tilde{B}^c_{t_a}| \geqslant \frac{|S|\cdot|P^c|}{2}$$

which implies that

$$\frac{d}{n-1}\sum_{a\in S}|\mathcal{O}_a\cap\tilde{B}^c_{t_a}| \geqslant \frac{d}{n-1}\left(\frac{|S|\cdot|P^c|}{2}-\sum_{a\in S}|\mathcal{O}_a\cap\tilde{B}_{t_a}|\right).$$

Using the same argument of Claim 7, it can be shown that $|\tilde{B}_{t_a}|\leqslant\frac{n}{c-1}$, yielding

$$\frac{d}{n-1}\sum_{a\in S}|\mathcal{O}_a\cap\tilde{B}^c_{t_a}| \geqslant \frac{d}{n-1}\left(\frac{|S|\cdot|P^c|}{2}-\frac{n}{c-1}|S|\right) \overset{(I)}{\geqslant} \frac{d}{7}|S| \tag{9}$$

where in (I) we used that $|S|\leqslant\frac{n}{2}$, that $|P^c|=n-\frac{11}{10}|S|$ and that we can take for example $c\geqslant 16$ as in Lemma 13. By plugging (9) in (8), we obtain

$$\mathbf{Pr}\left[\Gamma_t(S)\subseteq T\right] \leqslant \exp\left(-\frac{d}{n-1}\sum_{i=1}^{k}\left|\mathcal{O}_i\cap\tilde{B}^c_{t_a}\right|\right) \leqslant \exp\left(-\frac{d}{7}|S|\right). \tag{10}$$

Then, a union bound on all possible choices of S,T leads us to $\mathbf{Pr}\left[E_t\right]\leqslant \mathrm{e}^{-n}$, by using the fact that $s\geqslant\frac{n}{2000}$ and taking d large enough.

The proof of Case (ii) can be completed with the same argument, by considering the requests sent from P^c to S. ◀

4.2 Proof of Claim (b) of Theorem 12

The following lemma immediately implies Claim (b) of Theorem 12.

▶ **Lemma 15.** *There exist constants c and d sufficiently large such that, for all n large enough, the following holds. A constant $\beta=\beta(c,d)>0$ exists such that the snapshot G_t of $\mathcal{TSG}(n,d,c)$ for any $t\geqslant 2n$ verifies the following property. A subset $H_t\subseteq V_t$ with $|H_t|\geqslant n-\beta\log n$ exists such that the induced subgraph $G_t[H_t]$ has vertex expansion at least $\frac{1}{20}$, w.h.p.*

Proof. Fix $\beta=100(cd)^2$. To prove the lemma, we show that the following event holds w.h.p.

$$E=\left\{\exists H_t \text{ with } |H_t|\geqslant n-\beta\log n \text{ s.t. } \min_{S\subseteq H_t:|S|\leqslant n/2}\frac{|\Gamma_t(S)\cap H_t|}{|S|}\geqslant\frac{1}{20}\right\}.$$

Notice that, in Lemma 13 and Lemma 14, we proved that all the sets $S \subseteq V_t$ with size at least $\beta \log n$ have vertex expansion at least $\frac{1}{10}$ w.h.p. Therefore, to show that E holds w.h.p., we need to prove that there exists a subset $H_t \subseteq V_t$ such that, all the sets $S \subseteq H_t$ with $S \leqslant 20\beta \log n$ have also vertex expansion at least $\frac{1}{20}$. Indeed, the fact that the subsets $|S| \geqslant \beta \log n$ have expansion at least $1/10$ implies directly that the event E holds for such subsets, since, for each $S \subseteq H_t$ such that $|S| \geqslant 20\beta \log n$, we have

$$\frac{|\Gamma_t(S) \cap H_t|}{|S|} \geqslant \frac{|\Gamma_t(S)|}{|S|} - \frac{\beta \log n}{|S|} \geqslant \frac{1}{10} - \frac{1}{20} = \frac{1}{20}.$$

In particular, we will take H_t as all the set of nodes without pending requests at time t. More formally, we have that $E_1 \cap E_2 \cap E_3 \subseteq E$, where E_1 and E_2 are the events defined in the Lemma 13 and Lemma 14, and

$$E_3 = \left\{ \exists H_t \text{ with } |H_t| \geqslant n - \beta \log n \text{ s.t. } \min_{S \subseteq H_t : |S| \leqslant 20\beta \log n} \frac{|\Gamma_t(S) \cap H_t|}{|S|} \geqslant \frac{1}{20} \right\}.$$

From Lemma 13 and Lemma 14, we have that $\mathbf{Pr}\left[E_1^c \cup E_2^c\right] \leqslant 4n^{-2}$. In what follows, we will show that $\mathbf{Pr}\left[E_3^c\right] \leqslant 2n^{-2}$.

Notice that, from Lemma 9, we have that, if $A = \{|Q_t| \leqslant \beta \log n\}$, it holds that

$$\mathbf{Pr}\left[E_3^c\right] \leqslant \mathbf{Pr}\left[E_3^c \cap A\right] + \mathbf{Pr}\left[A^c\right] \leqslant \mathbf{Pr}\left[E_3^c \cap A\right] + n^{-2}. \tag{11}$$

If we define $\tilde{Q}_t$ as the set of nodes $v \in V_t$ with at least one pending request, we have that $|\tilde{Q}_t| \leqslant |Q_t|$ and that (taking $H_t = V_t \setminus \tilde{Q}_t$)

$$\begin{aligned} E_3^c \cap A &\subseteq \left\{ \exists S \subseteq V_t \setminus \tilde{Q}_t \text{ s.t. } |S| \leqslant 20\beta \log n,\ |\Gamma_t(S) \setminus \tilde{Q}_t| \leqslant \tfrac{1}{10}|S| \right\} \\ &\subseteq \left\{ \exists S \subseteq V_t \setminus \tilde{Q}_t,\ \exists T \subseteq V_t \setminus S \text{ s.t. } |S| \leqslant 20\beta \log n,\ |T| = \tfrac{1}{10}|S|,\ \Gamma_t(S) \subseteq T \cup \tilde{Q}_t \right\}. \end{aligned}$$

Therefore, it holds that

$$\begin{aligned} \mathbf{Pr}\left[E_3^c \cap A\right] &\leqslant \sum_{\substack{S \subseteq V_t : \\ |S| \leqslant 20\beta \log n}} \sum_{\substack{T \subseteq V_t \setminus S : \\ |T| = \frac{1}{10}s}} \mathbf{Pr}\left[\Gamma_t(S) \subseteq T \cup \tilde{Q}_t, S \cap \tilde{Q}_t = \emptyset\right] \\ &= \sum_{\substack{S \subseteq V_t : \\ |S| \leqslant 20\beta \log n}} \sum_{\substack{T \subseteq V_t \setminus S : \\ |T| = \frac{1}{10}s}} \mathbf{Pr}\left[\cap_{r \in S \times [d]} \{X_t(r) \in S \cup T \cup \tilde{Q}_t\}\right] \\ &\overset{(I)}{\leqslant} \sum_{\substack{S \subseteq V_t : \\ |S| \leqslant 20\beta \log n}} \sum_{\substack{T \subseteq V_t \setminus S : \\ |T| = \frac{1}{10}s}} \left(\frac{220(|S| + \frac{1}{10}|S| + |\tilde{Q}_t|)}{n} \right)^{d|S|} \overset{(II)}{\leqslant} n^{-2} \end{aligned}$$

where (I) follows from Lemma 8, and (II) from the fact that we are looking at $E_3^c \cap A$, hence $|\tilde{Q}_t| \leqslant \beta \log n$, and since $s \leqslant 20\beta \log n$ and taking d large enough.

From (11), since $\mathbf{Pr}\left[E_1^c \cup E_2^c\right] \leqslant 4n^{-2}$, and since $E_1 \cap E_2 \cap E_3 \subseteq E$, it follows that

$$\mathbf{Pr}\left[E^c\right] \leqslant \mathbf{Pr}\left[E_1^c \cup E_2^c\right] + \mathbf{Pr}\left[E_3^c\right] \leqslant 6n^{-2},$$

proving the lemma. ◀

5 Proof of Corollary 2: Convergence Time of PUSH and PULL

The aim of this section is to analyze the rumor-spreading process on the $\mathcal{TSG}$ model and prove Corollary 2.

5.1 Rumor spreading on the $\mathcal{TSG}$ model

We shortly recall how PUSH and PULL [23] can be defined on the $\mathcal{TSG}$ model. Such simple, local mechanisms are used to perform efficient broadcast operations over communication networks.

Given a connected graph $G = (V, E)$ and a *source* vertex $s \in V$, the goal is to inform all vertices about a piece of information that only s initially knows. The synchronous, uniform PUSH protocol works as follows. At round $t = 0$, the source selects one neighbor v uniformly at random and sends the message to it: we say that v is *informed* at (the end of) round t. Then, at every round $t \geqslant 1$, each informed vertex selects one random neighbors and sends the message to it. In the PULL protocol, each node u, which is still not informed at (the beginning of) round t, selects one random neighbor v and, if v is informed, then u pulls the source message from v and gets informed. The PUSH-PULL protocol is defined by considering both the PUSH and PULL actions performed by each vertex, at every round.

In order to combine the protocols above with the process generated by the $\mathcal{TSG}$ model, we organize each synchronous round $t \geqslant 1$ in two consecutive *phases*. In the first, *topology* phase, all the actions of the $\mathcal{TSG}$ process described in Definition 4 and Definition 5 take places: this generates the snapshot G_t. Then, in the second *rumor-spreading* phase, the local rule of PULL and/or PUSH are applied by every vertex in V_t in parallel on G_t.

5.2 Proof of Corollary 2

Rumor spreading on static graphs: Previous results. Our proof makes use of the following important result, that bounds the completion time of rumor spreading protocols over static graphs of bounded degree, and its proof argument (see [15, Theorem 12]). We recall its statement and provide a short overview of its proof argument.

▶ **Theorem 16** ([15]). *Let $G = (V, E)$ be a connected n-vertices graph with conductance ϕ and such that, for any edge $\{u, v\} \in E$, $\deg(u)/\deg(v) = \Theta(1)$. Then, $O(\log n/\phi)$ rounds of* PUSH *or* PULL *suffice to spread to all nodes of G a message originated at an arbitrary source node, w.h.p.*

Proof Overview. Let us just consider the case where $\phi = \Theta(1)$. Let $I_t \subseteq V$ the set of informed nodes at round t and assume that $|I_t| \leqslant n/2$. We first notice that, since G is an almost-regular $\Theta(1)$-expander, the size of the outer boundary of I_t is such that $|\Gamma(I_t)| \geqslant \gamma |I_t|$, for some constant $\gamma > 0$. Then, at every round $t' \geqslant t$, the PULL or the PUSH protocol let every node $v \in \Gamma(I_t)$ to have constant probability to get informed. This implies that the expected number of informed nodes at round $t+1$ will be at least $(1 + \Theta(1))|I_t|$. By applying suitable concentration arguments, this fact is then used to show that, within $O(\log n)$ rounds, the number of informed nodes is at least $n/2$, w.h.p. Once the spreading process reaches at least $n/2$ informed nodes, the analysis proceeds in a similar way by looking at the set non-informed nodes at round t and show that this quantity decreases at exponential rate, w.h.p. ◀

The analysis on the $\mathcal{TSG}$ model. In order to apply the above proof argument on the $\mathcal{TSG}$ model we need to cope with two main technical issues.

Our Lemma 13 shows that, at any round $t \geqslant 2n$, each subset $S \subseteq V_t$ of the snapshot $G_t = (V_t, E_t)$ with $|S| \geqslant \beta \log n$ for some constant $\beta > 0$, has conductance $\phi_t(S) = \Omega(1)$, w.h.p. Then, in order to apply the proof argument of Theorem 16, we need to show that there is an initial phase of the rumor-spreading process, called *bootstrap*, that is able to inform at

least $\beta \log n$ vertices, w.h.p. Indeed, after this bootstrap, we can apply the same argument of the proof of Theorem 16 assuming that there is an informed subset of logarithmic size. The analysis of the bootstrap will be discussed later in this section.

The second technical issue is caused by the presence of a set of *old* nodes (defined later in this section) that, during the information process, can leave the graph and create edge deletions and regenerations. However, once the bootstrap is completed, the subset of informed nodes reaches a logarithmic size which is large enough to dominates the impact of all possible edge deletions that can take place for a time window of logarithmic size even in an adversarially fashion: this time window is exactly what the rumor spreading process needs to complete the broadcast task. As we used in several previous steps of our analysis, this limited impact is essentially due to the fact that the maximum vertex degree of the graph snapshots is always bounded by the constant quantity $(c+1)d$ and thus, at every round, only this number of edges can be deleted.

The Bootstrap. Recall that s is the source node joining the dynamic graph in round $t_s \geqslant 2n$. Let OLD be the nodes in V_{t_s} having age larger than $n - \log^2 n$. Our goal is to prove the following.

▶ **Lemma 17.** *Let $\beta > 0$. Then, within $T' = O(\log n)$ rounds after the informed source s joined the graph at time t_s, there are $\beta \log n$ informed nodes whose age is at most $n - \log^2 n$, w.h.p.*

Proof. From Lemma 15, at time t_s when the source enters the graph, there exists a connected[11] graph $G_{t_s}[H_{t_s}]$ with vertex subset H_{t_s} of size $n - O(\log n)$. Consider now the connected components $\{C_i\}_{i \in I}$, obtained by removing from H_{t_s} the set OLD of nodes that will die within the next $\Theta(\log^2 n)$ rounds. How many nodes in $\{C_i\}_{i \in I}$ belong to a connected component of size smaller than $\beta \log n$? Since $|I| \leqslant |\text{OLD}| = \Theta(\log^2 n)$, there are at most $\beta \log n \cdot |I| = O(\log^3 n)$ such nodes. We now proceed with defining the following events: Let D be the event "s is not connected after $2 \log n$ rounds", B be the event "s targets a node which is either in OLD or in a small component during some round $t_s, ..., t_s + 2 \log n$" and C be the event "s gets connected to a node in a large connected component that will remain connected for at least $\Theta(\log^2 n)$ rounds". Now notice that, since $C^c \subseteq B \cup D$

$$\mathbf{Pr}[C] = 1 - \mathbf{Pr}[C^c] \geqslant 1 - \mathbf{Pr}[B \cup D] \geqslant 1 - (\mathbf{Pr}[B] + \mathbf{Pr}[D]). \tag{12}$$

We then know that $\mathbf{Pr}[D] \leqslant O(n^{-2})$ by a standard concentration argument on the geometric probability of success (we again use Claim 7 that says that, at every round, the number of full vertices is at most n/c). Observe also that, using a union bound over the observed time window, $\mathbf{Pr}[B] = O(\text{polylog}(n)/n)$, since at each round the probability that the request targets a node in OLD or in a small component (regardless of whether it is relaunched or not) is $O(\text{polylog}(n)/n)$.

From the above facts and Equation (12) we get that, w.h.p., the source s will belong to a subgraph of size at least $\beta \log n$ that will remain connected for at least $\Theta(\log^2 n)$ rounds. ◀

Finally, thanks to Lemma 17, we can apply the expansion argument we described in the proof sketch of Theorem 16 to the sets with size $\geqslant \beta \log n$ and get that, w.h.p., after $O(\log n)$ rounds, at least $n - O(\log n)$ vertices in the graph will be informed (Lemma 13 using Lemma 14).

[11] $G_{t_s}[H_{t_s}]$ is a vertex expander but in this proof we only use the fact that it is a large connected subgraph.

▶ Remark 18. Our analysis above proving Corollary 2 easily implies a further stabilizing property of the rumor spreading protocols on the $\mathcal{TSG}$ model. In particular, after the source joins the graph at round t_s, for a time window of a polynomial length, every new vertex will get informed within $O(\log n)$ rounds w.h.p.

6 Conclusion and Open Questions

The study of dynamic-graphs models capturing key aspects of real dynamic networks is currently a hot topic in algorithmic research and network science. In what follows, we discuss some open questions related to the model and the results presented in this paper.

We believe it is possible to extend our analysis on other, more realistic models of node churn, such as the *Poisson* one where nodes enter according to a Poisson clock and have a random age following an exponential distribution [13, 42]. In this setting, the analysis gets more complicated by two further issues: the *random* number of nodes each snapshot can have and the presence of nodes having *random* age, possibly larger than n. However, we think that the key arguments we used in the analysis of the streaming model can be adapted to take care about such further issues. Essentially, it could be possible to exploit concentration results on both the number of nodes in a snapshot and on the random life of a node.

A further interesting scenario is that generated by a different mechanism to get new link connections. For instance, we can think of a link manager that returns a non-uniform distribution over the current set of nodes, or that can selects possible links from an underlying (dynamic) graph somewhat representing social relationships among nodes.

Finally, an important property of distributed protocols is *self-stabilization* [4, 24]. For short, it represents the ability of a protocol to recover its "good" behavior (guaranteeing some desired performance and/or property) from any (worst-case) configuration the system can be landed on, due to some bad event (e.g. a node/link fault and/or an adversarial setting of some local variable). The current version of RAES is not fast self-stabilizing under a worst-case scenario where the adversary can corrupt all nodes: essentially, it can construct a non-expander topology respecting the algorithm rules than can last for a linear number of rounds. However, Lemma 10 ensures that the number of pending requests decreases faster: we believe this key-fact can be exploited to design a different, more robust version of RAES having fast self-stabilization.

References

1 Jannik Albrecht, Sebastien Andreina, Frederik Armknecht, Ghassan Karame, Giorgia Marson, and Julian Willingmann. Larger-scale nakamoto-style blockchains don't necessarily offer better security. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 2161–2179. IEEE, 2024.

2 André Allavena, Alan Demers, and John E Hopcroft. Correctness of a gossip based membership protocol. In *Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing*, pages 292–301, 2005. doi:10.1145/1073814.1073871.

3 Zeyuan Allen-Zhu, Aditya Bhaskara, Silvio Lattanzi, Vahab Mirrokni, and Lorenzo Orecchia. Expanders via local edge flips. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 259–269. Society for Industrial and Applied Mathematics, 2016. doi:10.1137/1.9781611974331.CH19.

4 Karine Altisen, Stéphane Devismes, Swan Dubois, and Franck Petit. *Introduction to distributed self-stabilizing algorithms*. Springer Nature, 2022.

5 Flora Angileri, Andrea Clementi, Emanuele Natale, Michele Salvi, and Isabella Ziccardi. Threshold-driven streaming graph: Expansion and rumor spreading. *CoRR*, abs/2507.23533, 2025. doi:10.48550/arXiv.2507.23533.

6 John Augustine, Gopal Pandurangan, and Peter Robinson. Distributed algorithmic foundations of dynamic networks. *ACM SIGACT News*, 47(1):69–98, 2016. doi:10.1145/2902945.2902959.

7 John Augustine, Gopal Pandurangan, Peter Robinson, Scott Roche, and Eli Upfal. Enabling robust and efficient distributed computation in dynamic peer-to-peer networks. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 350–369. IEEE, 2015. doi:10.1109/FOCS.2015.29.

8 John Augustine, Gopal Pandurangan, Peter Robinson, and Eli Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 551–569, USA, 2012. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611973099.47.

9 Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 910–928. IEEE, 2019. doi:10.1109/FOCS.2019.00059.

10 Annika Baumann, Benjamin Fabian, and Matthias Lischke. Exploring the bitcoin network. *WEBIST (1)*, 2014(369-374):3, 2014.

11 Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Finding a bounded-degree expander inside a dense one. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1320–1336. SIAM, 2020. doi:10.1137/1.9781611975994.80.

12 Luca Becchetti, Andrea Clementi, Francesco Pasquale, Luca Trevisan, and Isabella Ziccardi. Expansion and flooding in dynamic random networks with node churn. In *41st IEEE International Conference on Distributed Computing Systems, ICDCS 2021, Washington DC, USA, July 7-10, 2021*, pages 976–986. IEEE, 2021. doi:10.1109/ICDCS51616.2021.00097.

13 Luca Becchetti, Andrea Clementi, Francesco Pasquale, Luca Trevisan, and Isabella Ziccardi. Expansion and flooding in dynamic random networks with node churn. *Random Structures & Algorithms*, 63(1):61–101, 2023. doi:10.1002/RSA.21133.

14 Azzedine Boukerche and Amir Darehshoorzadeh. Opportunistic routing in wireless networks: Models, algorithms, and classifications. *ACM Computing Surveys (CSUR)*, 47(2):1–36, 2014.

15 Flavio Chierichetti, George Giakkoupis, Silvio Lattanzi, and Alessandro Panconesi. Rumor spreading and conductance. *J. ACM*, 65(4), 2018. doi:10.1145/3173043.

16 Andrea Clementi, Pierluigi Crescenzi, Carola Doerr, Pierre Fraigniaud, Marco Isopi, Alessandro Panconesi, Francesco Pasquale, and Riccardo Silvestri. Rumor spreading in random evolving graphs. In *European symposium on algorithms*, pages 325–336. Springer, 2013. doi:10.1007/978-3-642-40450-4_28.

17 Andrea Clementi, Pierluigi Crescenzi, Carola Doerr, Pierre Fraigniaud, Francesco Pasquale, and Riccardo Silvestri. Rumor spreading in random evolving graphs. *Random Structures & Algorithms*, 48(2):290–312, 2016. doi:10.1002/RSA.20586.

18 Andrea Clementi, Emanuele Natale, and Isabella Ziccardi. Parallel load balancing on constrained client-server topologies. *Theor. Comput. Sci.*, 895:16–33, 2021. doi:10.1016/J.TCS.2021.09.026.

19 Colin Cooper, Martin E. Dyer, and Catherine S. Greenhill. Sampling regular graphs and a peer-to-peer network. *Comb. Probab. Comput.*, 16(4):557–593, 2007. doi:10.1017/S0963548306007978.

20 Colin Cooper, Ralf Klasing, and Tomasz Radzik. A randomized algorithm for the joining protocol in dynamic distributed networks. *Theor. Comput. Sci.*, 406(3):248–262, 2008. doi:10.1016/J.TCS.2008.06.049.

21 Michael S Crouch, Andrew McGregor, and Daniel Stubbs. Dynamic graphs in the sliding-window model. In *Algorithms–ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings 21*, pages 337–348. Springer, 2013. doi:10.1007/978-3-642-40450-4_29.

22 Antonio Cruciani and Francesco Pasquale. Dynamic graph models inspired by the bitcoin network-formation process. In *Proceedings of the 24th International Conference on Distributed Computing and Networking*, pages 125–134, 2023. doi:10.1145/3571306.3571398.

23 Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12, 1987.

24 Edsger W Dijkstra. Self-stabilizing systems in spite of distributed control. *Communications of the ACM*, 17(11):643–644, 1974. doi:10.1145/361179.361202.

25 Philippe Duchon and Romaric Duvignau. Local update algorithms for random graphs. In Alberto Pardo and Alfredo Viola, editors, *LATIN 2014: Theoretical Informatics - 11th Latin American Symposium, Montevideo, Uruguay, March 31 - April 4, 2014. Proceedings*, volume 8392 of *Lecture Notes in Computer Science*, pages 367–378. Springer, 2014. doi:10.1007/978-3-642-54423-1_32.

26 Chinmoy Dutta, Gopal Pandurangan, Rajmohan Rajaraman, and Zhifeng Sun. Information spreading in dynamic networks. *arXiv preprint arXiv:1112.0384*, 2011. arXiv:1112.0384.

27 Robert Elsässer and Thomas Sauerwald. Cover time and broadcast time. In *26th International Symposium on Theoretical Aspects of Computer Science (2009)*, pages 373–384. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2009. doi:10.4230/LIPIcs.STACS.2009.1842.

28 Uriel Feige, David Peleg, Prabhakar Raghavan, and Eli Upfal. Randomized broadcast in networks. *Random Structures & Algorithms*, 1(4):447–460, 1990. doi:10.1002/RSA.3240010406.

29 George Giakkoupis. Expanders via local edge flips in quasilinear time. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 64–76, 2022. doi:10.1145/3519935.3520022.

30 George Giakkoupis, Thomas Sauerwald, and Alexandre Stauffer. Randomized rumor spreading in dynamic graphs. In *Automata, Languages, and Programming: 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part II 41*, pages 495–507. Springer, 2014. doi:10.1007/978-3-662-43951-7_42.

31 Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks. In *IEEE INFOCOM 2004*, volume 1. IEEE, 2004.

32 Rachid Guerraoui, Anne-Marie Kermarrec, Anastasiia Kucherenko, Rafael Pinot, and Martijn de Vos. Peerswap: A peer-sampler with randomness guarantees. In *2024 43rd International Symposium on Reliable Distributed Systems (SRDS)*, pages 271–281. IEEE, 2024. doi:10.1109/SRDS64841.2024.00034.

33 Aayush Gupta and Gopal Pandurangan. Fully-distributed construction of byzantine-resilient dynamic peer-to-peer networks. *CoRR*, abs/2506.04368, 2025. doi:10.48550/arXiv.2506.04368.

34 Mor Harchol-Balter, Tom Leighton, and Daniel Lewin. Resource discovery in distributed networks. In *Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 229–237, 1999.

35 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

36 Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten Van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8–es, 2007.

37 Richard Karp, Christian Schindelhauer, Scott Shenker, and Berthold Vocking. Randomized rumor spreading. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 565–574. IEEE, 2000. doi:10.1109/SFCS.2000.892324.

38 Ching Law and K-Y Siu. Distributed construction of random expander networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM 2003)*, volume 3, pages 2133–2143. IEEE, 2003.

39 Laurent Massoulié, Erwan Le Merrer, Anne-Marie Kermarrec, and Ayalvadi Ganesh. Peer counting and sampling in overlay networks: random walk methods. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 123–132, 2006. doi:10.1145/1146381.1146402.

40 Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

41 Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf, 2008.

42 Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Building low-diameter peer-to-peer networks. *IEEE Journal on selected areas in communications*, 21(6):995–1002, 2003. Preliminary version in FOCS'01. doi:10.1109/JSAC.2003.814666.

43 Daniel Stutzbach and Reza Rejaie. Understanding churn in peer-to-peer networks. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 189–202, 2006. doi:10.1145/1177080.1177105.

44 Robbert Van Renesse, Yaron Minsky, and Mark Hayden. A gossip-style failure detection service. In *Middleware'98: IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, pages 55–70. Springer, 1998.