Abstract—This paper addresses the following foundational question: what is the maximum theoretical delay performance achievable by an overlay peer-to-peer streaming system where the streamed content is subdivided into chunks? As shown in this paper, when posed for chunk-based systems, and as a consequence of the store-and-forward way in which chunks are delivered across the network, this question has a fundamentally different answer with respect to the case of systems where the streamed content is distributed through one or more flows (sub-streams). To circumvent the complexity emerging when directly dealing with delay, we express performance in terms of a convenient metric, called “stream diffusion metric”. We show that it is directly related to the end-to-end minimum delay achievable in a P2P streaming network. In an homogeneous scenario, we derive a performance bound for such metric, and we show how this bound relates to two fundamental parameters: the upload bandwidth available at each node, and the number of neighbors a node may deliver chunks to. In this bound, n-step Fibonacci sequences do emerge, and appear to set the fundamental laws that characterize the optimal operation of chunk-based systems. A further technical contribution of the paper is an advance in the theory of n-step Fibonacci sums for which a prior reference result was missing. \footnote{This work was supported by the Italian Ministry of University and Research (MiUR), with the Grant PRIN-2006099023 “Profiles” (disi.unin.it/profiles).}

I. INTRODUCTION

Nowadays, peer-to-peer (P2P) overlay live streaming systems are of significant interest, thanks to their low implementation complexity, scalability and reliability properties, and ease of deployment. Leveraging on the well understood P2P communication paradigm, the viability to deliver live streaming content on top of a self-organizing P2P architecture has been widely assessed both in terms of research contributions, as well as in terms of real-life applications.

In principle, the most natural and earlier solution for deploying a P2P streaming system was to organize peer nodes in one or more overlay multicast trees, and hence continuously deliver the streamed information across the formed paths. This is the case in [1], [2], [3]. Since the information, organized in the form of small IP packets, would be sequentially delivered across these trees, the processing time of packets at each network node would be marginal, and performance bounds would only depend on the “optimality” (with respect to some meaningful performance metric) of the formed distribution paths. However, in practice, this approach may not be viable in large-scale systems and with nodes characterized by intermittent connectivity (churn). In fact, whenever a node in the middle of a path abruptly disconnects, complex procedures would be necessary to i) allow the reconstruction of the distribution path, and ii) allow the nodes affected by such event to recover the amount of information lost during the path reconfiguration phases. To overcome such limitations, a completely different approach, called data-driven, delivers content on the basis of content availability information, locally exchanged among connected peers, without any a priori pre-established path. This approach essentially creates a mesh topology among overlay nodes. Several proposed solutions, such as [4], [5], [6], [7], adopt the data-driven approach.

In this paper we focus on chunk-based systems, where, similarly to most file-sharing P2P applications, the streaming content is segmented into smaller pieces of information called chunks. Chunks are elementary data units handled by the nodes composing the network in a store-and-forward fashion. A relaying node can start distributing a chunk only when it has completed its reception from another node. The data-driven solutions proposed in [4], [6], [7] may be regarded as chunk-based. A characterizing feature of the chunk-based approach is that, in order to reduce the per-chunk signalling burden, the chunk size is typically kept to a fairly large value, greater than the typical packet size.

In this paper we raise some very basic and foundational questions on chunk-based systems: what are the theoretical performance limits, with specific attention to delay, that any chunk-based peer-to-peer streaming system is bounded to? Which fundamental laws describe how performances depend on network parameters such as the available bandwidth or system parameters such as the number of nodes a peer may at most connect to? And which are the system topologies and operations which would allow to approach such bounds?

Surprisingly enough, according to the best of our knowledge and our literature survey, these questions have never been directly posed before, and answered, for chunk-based systems (the only references somewhat related to these issues are [12] for the file sharing case, and [11] for the streaming case). The aim of this paper is to answer these questions. The answer is completely different from the case of systems where the streaming information, optionally organized in sub-streams, is continuously delivered across overlay paths (for a theoretical investigation of such class of approaches refer to [13] and references therein contained). As we will show, in our scenario the time needed for a chunk to be forwarded across a node significantly affects delay performance.
In the paper, for optimal performance we mean the ability to reach the greatest possible number of nodes in a given time interval (this will be later on formally defined as “stream diffusion metric”) or equivalently the ability to reach a given number of nodes in the smallest possible time interval (i.e. absolute delay). We derive analytic expressions for the maximum asymptotic stream diffusion metric in an homogeneous network composed of stable nodes whose upload bandwidth is the same (for simplicity, multiple of the streaming rate). The assumption of homogeneous network and no churn should not be considered critical in this paper, as the goal is to derive theoretical foundational insights, and the first step for achieving understanding is clearly the homogeneous and ideal case.

II. Motivation

Goal of this section is to clarify why P2P chunk-based streaming systems have significantly different performance issues with respect to streaming systems, where the information content continuously flows across one or more overlay paths or trees. Unless ambiguity occurs, such systems will be referred to as, with slight abuse of name, flow-based systems. More precisely, we will show that i) theoretical bounds derived for the flow-based case may not be representative for chunk-based systems, and new, fundamentally different, bounds are needed, ii) the methodological approaches which are applicable in the two cases are completely diverse, and fluidic approaches may be replaced with inherently discrete-time approaches where, as shown later on, k-step Fibonacci series and sums, and related analytical tools enter into play.

A. Delay in flow-based systems

We recall that “flow-based” system denotes a stream distribution approach where the streaming information, possibly organized in multiple sub-streams, is delivered with continuity across one or more overlay network paths. Clearly, in the real IP world, continuous delivery is an abstraction, as the streaming information will be delivered in the form of IP packets. However, the small size of IP packets yields marginal transmission times at each node. As such, the remaining components that cause delay over an overlay link (propagation and path delay because of queuing in the underlying network path) may be considered predominant.

We can conclude that the delay performances of flow-based systems ultimately depend on the delay characterizing a path between the source node and a generic end-peer. More specifically, if we associate a delay figure to each overlay link, then the source to destination delay depends on the sum of the link delays: the transmission times needed by the flow to “cross” a node may be neglected, or, more precisely, they play a role only because the “crossed” nodes compose the vertices of the overlay links, whose delays dominate the overall delay performance.

As a consequence, the delay performance optimization becomes a minimum path cost problem, as such addressed with relevant analytical techniques. If we further assume that the network links are homogeneous (i.e. characterized by the same delay), then the problem of finding a delay performance bound is equivalent to finding what is the minimum depth of the tree (or multiple trees) across which the stream is distributed. This problem has been thoroughly addressed in [13], under the general assumption that a stream may be subdivided into sub-streams (delivered across different paths), and that each node may upload information to a given maximum number of children. For instance, if we assume no restriction on the number of children a node may upload to, then it is proven in [13] that a tree depth equal to two is always sufficient. This is indeed immediate to understand and visualize in the special case of all links with a “sufficient” amount of available upload bandwidth - see figure 1 for a constructive example.

At this stage, it should be clear that, in the context of flow-based systems, as long as some feasibility conditions are met (see e.g. [14]) the bandwidth available on each link plays a limited role with respect to the delay performance achievable. This is clearly seen by looking again at figure 1: if for instance we double the bandwidth available on each link, the delay performances do not change (at least until the source is provided with a large enough amount of bandwidth to serve all peers in a single hop).

B. Delay in chunk-based systems

Chunk-based systems have a key difference with respect to flow-based systems: the streaming information is organized into chunks whose size is significantly greater than IP packets. Since a peer must complete the reception of a chunk before forwarding it to other nodes (i.e. chunks are delivered in a store-and-forward fashion), the obvious consequence is that delay performance are mostly affected by the chunk transmission time. Thus, in terms of delay performance, the behavior of

\[ \text{Network links are homogeneous (i.e. characterized by the same delay), then the problem of finding a delay performance bound is equivalent to finding what is the minimum depth of the tree (or multiple trees) across which the stream is distributed. This problem has been thoroughly addressed in [13], under the general assumption that a stream may be subdivided into sub-streams (delivered across different paths), and that each node may upload information to a given maximum number of children. For instance, if we assume no restriction on the number of children a node may upload to, then it is proven in [13] that a tree depth equal to two is always sufficient. This is indeed immediate to understand and visualize in the special case of all links with a “sufficient” amount of available upload bandwidth - see figure 1 for a constructive example.} \]

In this example, the amount of available upload bandwidth is “sufficient” in the sense that the source node has a bandwidth at least equal to the stream bit rate \( R_c \), while each peer node has a bandwidth at least equal to \( (N-1) \cdot R_c/N \), being \( N \) the number of peer nodes composing the overlay. As shown in [13] the same result holds under significantly less restrictive assumptions on the available bandwidth.
chunk-based systems is opposite to the one of flow-based systems. Not only chunk transmission times cannot be neglected anymore with respect to link-level delays (propagation and underlying network queueing), but also we can safely assume that in most scenarios any other delay component at the link-level has negligible impact when compared with the chunk transmission times. This consideration can be restated as: the delay performances of chunk-based systems do not depend on the sum of the delays experienced while traveling over an overlay link, but depend on the sum of the delays experienced while crossing a node.

From a superficial analysis, one might argue that the overall delay optimization problem does not change. In fact, the transmission delay of a chunk at a given node could be attributed to the overlay link over which the chunk is being transmitted, and, also in this case, the optimization could be stated as a minimum path cost problem.

However, a closer look reveals that this is not at all the case. The reasons are manifold and can be illustrated with the help of figure 2. In this figure, and consistently throughout the paper, we rely on the following notation. \( C \) is the chunk size (in bit); \( R_{bps} \) is the streaming constant bit rate (in bps). \( T = C/R_{bps} \) is the chunk “inter-arrival” time at the source, being such arrival process a direct consequence of the segmentation into chunks done at the source: a new chunk will be available for delivery only when \( C \) information bits, generated at rate \( R_{bps} \), are accumulated (see top of figure 2). \( U_{bps} \) is the available upload bandwidth, assumed to be the same for all network nodes, including the source (homogeneous bandwidth conditions). \( U = U_{bps}/R_{bps} \) is the normalized upload bandwidth of each node with respect to the streaming bit rate. In this paper, for simplicity, we consider the case of \( U \) integer greater or equal than 1, i.e. \( U_{bps} \) being either equal or a multiple of \( R_{bps} \). The minimum transmission time for a chunk is equal to \( T^* = C/U_{bps} = T/U \); this is true only if the whole upload bandwidth \( U_{bps} \) is used to transmit a single chunk to a single node. Moreover, we rely on the common simplifying assumption, in overlay P2P systems, that the only bandwidth bottleneck is the uplink bandwidth of the access link that connects the peer to the underlying network (the downlink bandwidth is considered sufficiently large not to be a bottleneck - this is common in practice, due to the large deployment of asymmetric access links - e.g., ADSL).

The first reason why the overall delay optimization problem can not be stated as a minimum path cost problem in the case of chunk-based systems is the sharing of the available upload bandwidth \( U_{bps} \) across multiple overlay links. As a consequence, i) it is not possible to a priori associate a constant delay cost to overlay links originating from a given node, ii) the delay experienced while transmitting a chunk depends on the fraction of the bandwidth that the node is dedicating to such transmission. For instance, figure 2 shows that the source node is transmitting a given chunk in parallel to two nodes; as such, the transmission delay is \( C/(U_{bps}/2) \). If the source were transmitting the chunk only to node 1, this delay would be halved.

The second reason is that the transmission time may not be the only component of the overall chunk delivery delay. This is highlighted for the case of node N1. After receiving chunk 1, node N1 adopts the strategy of serializing the delivery of chunk 1 to nodes N4 and N5. On the one side, in both cases the chunk will be transmitted in the same time, namely \( C/U_{bps} \); this is the minimum transmission time for a chunk, as all the available bandwidth is always dedicated to a single transmission. On the other side, the time elapsing between the instant at which the chunk is available at node N1 and the instant at which the chunk is received by node N5 is greater than the transmission time, as it includes also the time spent by node N1 while transmitting the chunk to node N4.

The third and final aspect which characterizes chunk-based systems in a streaming context is that there is a tight constraint which relates the number of peer nodes that can be simultaneously served and the available upload bandwidth. If we look back flow-based systems in figure 1, we see that only practical implementation issues may impede the source node to arbitrarily subdivide the stream into sub-streams, and the tree depth may be indeed trivially optimized by using as many sub-streams as the number of nodes in the network. On the contrary, in chunk-based systems, the number of nodes that can be served is no more a “free” parameter, but it is tightly constrained by the stream rate and the available upload bandwidth. This fact can be readily understood by looking at the source node in the example illustrated in figure 2. Due to their granularity, new chunks are available for delivery at the source node every \( T = C/R_{bps} \) seconds. Hence, in order to keep the distribution of chunks balanced (i.e., to avoid introducing delays with respect to the time instant at which chunks are available at source and to privilege specific chunks by giving them extra distribution time), the source node must complete the delivery of every chunk before the next new chunk is available for the delivery (i.e. within \( T \) seconds). This implies that the source node cannot deliver a single chunk to more than \( U \) nodes, being \( U = U_{bps}/R_{bps} \) the ratio between
the upload bandwidth and the streaming rate$^3$.

III. Stream Diffusion Metric: A Delay-Related Fundamental Bound

Let $\mathcal{P}$ be the set of all peers which compose a P2P streaming network, and let $|\mathcal{P}| = |\mathcal{P}|$ be the cardinality of such network. Let $p \in \mathcal{P}$ be a generic peer in the network. Since the streamed information is organized into subsequently generated chunks, $p$ is expected to receive all these chunks with some delay after their generation at the source. Let us define with $d(c,p)$ the specific interval of time elapsing between the generation of chunk $c$ ($c = 1, 2, 3, \cdots$) at the source, and its completed reception at peer $p$. In most generality, different chunks belonging to the stream may be delivered through different paths. This implies that $d(c,p)$ may vary with the chunk index $c$. Let

$$D(p) = \max_c d(c,p)$$

be the maximum delay experienced by peer $p$ among all possible chunks.

To characterize the delay performance of a whole P2P streaming network, we are interested in finding the maximum of the delay experienced across all peers composing the network, i.e.:

$$D(\mathcal{P}) = \max_{p \in \mathcal{P}} D(p)$$

We refer to this network-wide performance metric as absolute network delay. However, for reasons that will be clear later on, this performance metric does not yield to a convenient analytical framework. Thus, we introduce an alternative delay-related performance metric, which we call stream diffusion metric. This is formally defined as follows:

$$N(t) = |\mathcal{P}_t| \text{ where } \mathcal{P}_t = \{ p \in \mathcal{P} : D(p) \leq t \}$$

In plain words, $N(t)$ is the number of peers that may receive each chunk in at most a time interval $t$ after its generation at the source.

The most interesting aspect of the stream diffusion metric $N(t)$ is that it can be conveniently applied also to networks composed of an infinite number of nodes (for such networks, obviously, the absolute network delay $D(\mathcal{P})$ would be infinite). Moreover, for finite-size networks, it is straightforward to derive the absolute network delay from the stream diffusion metric. Since $N(t)$ is a non-decreasing monotone function of the continuous time variable $t$ and it describes the number of peers that may receive the whole stream within a maximum delay $t$, for a finite size network composed of $P$ peers the value of $t$ at which $N(t)$ reaches $P$ is also the maximum delay experienced across all peers. The formal relation between the absolute network delay and the stream diffusion metric is hence

$$D(\mathcal{P}) = \min\{t : N(t) = P\}$$

$^3$A similar conclusion can be drawn for other nodes as well. Moreover, we remark that this conclusion holds even when chunks are serially delivered, like in the case of node N1.

A. The bound on $N(t)$

Before stating the bound, we need to provide some preliminary notation.

Let $F_k(i)$ be the $k$-step Fibonacci sequence defined as follows:

$$F_k(i) = \begin{cases} 0 & \text{if } i \leq 0 \\ 1 & \text{if } i = 1 \\ \sum_{j=1}^{k} F_k(i-j) & \text{if } i > 1 \end{cases}$$

Let $S_k(n)$ be a new sequence defined as the sum of the first $n$ non-null terms of the $k$-step Fibonacci sequence, i.e.,

$$S_k(n) = \begin{cases} 0 & \text{if } n \leq 0 \\ \sum_{i=1}^{n} F_k(i) & \text{if } n > 0 \end{cases}$$

Let us assume that propagation delays and queuing delays experienced in the underlying physical network because of congestion are negligible with respect to the minimum chunk transmission time $T^* = C/U_{bps}$, namely the time needed to transmit a chunk by dedicating, to such transmission, all the upload capacity of a node. In what follows, we measure the time using, as time unit, the value $T^*$ above defined.

We can now state the following theorem, on to the upper bound of $N(t)$.

**Theorem 1:** In a P2P chunk-based streaming system where all peer nodes have the same normalized upload capacity $U = U_{bps}/R_{bps}$ (assumed integer greater or equal than 1) and $k$ overlay neighbors to deliver chunks to, the stream diffusion metric is upper bounded by

$$\overline{N}(t) = \sum_{j=1}^{U} S_k(t-j+1)$$

for integer values of $t$ (i.e. multiple of $T^*$) while, for non integer values of $t$, $\overline{N}(t) = \overline{N}([t])$ must be considered.

We preliminarily point out that the minimum amount of time elapsing between the time instant at which a peer receives a chunk and the time instant at which all its $k$ neighbors receive the same chunk is lower bounded by $k$ (or equivalently $k \cdot C/U_{bps} = k \cdot T^*$ seconds).

This is a trivial consequence of the fact that the node must replicate and deliver the chunk $k$ times, and hence the amount of time to transmit $k \cdot C$ bits using an upload capacity $U_{bps}$ cannot be lower than the ratio $k \cdot C/U_{bps} = k \cdot T^*$; it is equal to the latter value whenever a work-conserving scheduling discipline is employed.

We now introduce two best-case assumptions representative of an upper bound. First, let us assume that every node in the network, with the obvious exception of the source node (which is fedded with a new chunk every $T = U \cdot T^*$ second), is given sufficient time to deliver a chunk to all its $k$ neighbors. This is equivalent to assume that the next chunk to be delivered by the same node will not arrive before $k$ time units. To avoid misunderstanding, we remark that this implicitly implies that not all the chunks received by a node must be further forwarded: section IV will discuss what this operatively implies in terms of chunk distribution
across the whole network. Secondly, let us assume that all the $k$ neighbors of a given peer did not receive the considered chunk from other peers.

In order to prove the theorem, with regard to the generic non-source peer $X_0$, we introduce the relative stream diffusion metric $N_{X_0}(t)$, defined as the number of peer nodes which receive a chunk either directly or indirectly (through a neighbor, or a neighbor of a neighbor, etc) from $X_0$ with a delay lower than or equal to $t$. We observe that such a delay is evaluated with respect to the time at which peer $X_0$ completes the download of the considered chunk. Let us include in the count of nodes also the peer $X_0$ itself. By construction, i) $N_{X_0}(t) = 0 \forall t < 0$, as peer $X_0$ has not yet received the chunk and hence it has not yet started to distribute it further, ii) $N_{X_0}(0) = 1$, as the only peer which can receive the chunk with a null delay is $X_0$ itself, iii) $N_{X_0}(t)$ is a monotone non decreasing function of $t$.

Now, let $X_1, X_2, \ldots, X_k$ be the neighbors of node $X_0$, and let $d_i$ be the time interval after which a neighbor node $X_i$ receives the chunk from node $X_0$. For $t > 0$, the following “pseudo-recursion” holds:

$$N_{X_0}(t) = 1 + \sum_{i=1}^{K} N_{X_i}(t - d_i) \quad \forall t \geq 0 \quad (4)$$

where $N_{X_i}(t - d_i)$ is the relative stream diffusion metric starting from the neighbor $X_i$ at time $t - d_i$, and we use the term “pseudo-recursion” to underline that a neighbor node, in general, may deliver the chunk to its neighbors using a different chunk transmission discipline than node $X_0$.

The time intervals $d_1, d_2, \ldots, d_k$ depend on how the upload bandwidth of node $X_0$ is allocated to the $k$ transmissions of the considered chunk towards the $k$ neighbors. Let us assume, non restrictively, that $d_1 \leq d_2 \leq \cdots \leq d_k$. Then, it is trivial to prove that $d_i \geq i$, and that equality is achieved if and only if the transmission of chunks is serialized\footnote{As a sketch of the proof, note that under the non restrictive assumption $d_1 \leq d_2 \leq \cdots \leq d_k$, the only way to achieve $d_i = i$ for a generic value $1 \leq i \leq k$ is to dedicate, in the considered time period $i$, all the available upload capacity to the $i$ transmissions towards the neighbors $X_1, X_2, \ldots, X_i$, and hence defer the transmission of the chunk towards the remaining $k - i$ neighbors after the time interval $i$ has elapsed. However, this applies to all $i$, starting from the case $i = 1$. Hence, the only scheduling rule which satisfies the equality $d_i = i \forall i$ is the serial transmission.}. Based on this, and observing that by construction the relative stream diffusion metric starting from any node is a monotone non decreasing function, we can conclude that, for every $X_i$ in equation (4), $N_{X_i}(t - d_i) \leq N_{X_i}(t - i)$. Moreover, if we define with $\overline{N}_{X}(t)$ an upper bound on the relative stream diffusion metric starting from any possible node, then, by definition, $N_{X_i}(t - i) \leq \overline{N}_{X}(t - i)$. Hence, the right part of (4) is bounded by:

$$1 + \sum_{i=1}^{K} N_{X_i}(t - d_i) \leq 1 + \sum_{i=1}^{K} \overline{N}_{X}(t - i) \quad \forall t \geq 0 \quad (5)$$

In homogeneous conditions, the bound on the relative stream diffusion metric is independent of the specific non-source node

which diffuses the stream. Hence we can conclude that the upper bound can be computed as the solution of the following recurrence:

$$\overline{N}_{X}(t) = 1 + \sum_{i=1}^{K} \overline{N}_{X}(t - i) \quad \forall t \geq 0 \quad (6)$$

By using the result demonstrated in appendix A, Lemma 1, and setting as applicable initial conditions $\overline{N}_{X}(t) = 0$ for $t < 0$ and $\overline{N}_{X}(0) = 1$, we can conclude that for integer values of $t$

$$\overline{N}_{X}(t) = S_k(t + 1) \quad (7)$$

where $S_k(n)$ is the $k$-step fibonacci sum as defined in (2).

We also observe that relation (10) has already been derived in [11] for the only case $U = 1$. However, the author of [11] limits itself to investigate full-mesh overlay topologies.
IV. ATTAINING THE BOUND

The provided bound, as well as its derivation, offers only limited insights on how chunks should be forwarded across the overlay topology. Specifically, the bound clearly suggests that delay performances are optimized only if chunks are serially delivered towards the neighbor nodes, but does not make any assumption on which specific paths the chunks should follow, or in other words, which overlay topologies should be used. We now show that, to attain the performance bound, peer nodes have to be organized according to i) an overlay unbalanced tree if \( k = U \), ii) multiple overlay unbalanced trees if \( k > U \) and multiple of \( U \) (generalization to arbitrary integer values of \( k \) being straightforward).

A. Case \( k = U \): unbalanced tree

When the number of neighbor nodes \( k \) is equal to the normalized upload capacity \( U \), the source node can deliver each chunk to all its \( k \) neighbors before a new chunk arrives. As such, the source node can repeatedly apply a round-robin scheduling policy during the time interval \( T = UT^* \), which elapses between the arrivals of consecutive chunks. Specifically, in the first \( T^* \) seconds it can send a given chunk to a given node, say peer \( N_1 \), then send the chunk to peer \( N_2 \), and so on until peer \( N_k \). If this policy is repeated for every chunk, the result is that any neighbor of the source also receives a new chunk every \( T = UT^* \) seconds. Hence, each neighbor of the source may apply the same scheduling policy with respect to its neighbors, and so on. As a consequence, every node in the network receives chunks from the same parent, and in the original order of generation: in other words, chunks are delivered over a tree topology.

We can easily verify that the above described chunk distribution mechanism reaches the performance bound through the simple example depicted in figure 3, which refers to the case \( U = k = 2 \). The figure shows that a network composed of 19 nodes can receive every chunk in at most 5 time units. By evaluating the performance bound \( \bar{N}(t) \) at \( t = 5 \), we achieve the same result. It is then easy to verify the matching between the described chunk distribution mechanism and the performance bound \( \bar{N}(t) \) for each value of \( t \).

It is interesting to note that the tree formed in figure 3 is unbalanced in terms of number of hops. For instance, the chunk reaches node 19 at time \( t = 5 \) after crossing nodes 1,3,6 and 11. Conversely, the chunk reaches node 15, again at time \( t = 5 \), after crossing nodes 2 and 7. The unbalancing in terms of number of hops is a consequence of the fact that the proposed approach achieves equal-delay source-to-leaves paths, and that the time in which a chunk waits for its transmission turn at a node (because of serialization) contributes to such path delay.

B. Case \( k > U \) and multiple of \( U \): unbalanced multiple trees

When \( k > U \), the source cannot deliver a chunk to all its \( k \) neighbors, but only to a subset of \( U \) peers. Hence, in principle, it might distribute chunks through the same tree as discussed before, and hence every peer in the network would use only \( U \) neighbors over the available \( k \). However, the provided bound assures that performance in the case \( k > U \) are better than in the case \( k = U \). For instance, if \( U = 2 \), the case \( k = 4 \) outperforms the case \( k = 2 \) as follows:

<table>
<thead>
<tr>
<th>( t )</th>
<th>( 1 )</th>
<th>( 2 )</th>
<th>( 3 )</th>
<th>( 4 )</th>
<th>( 5 )</th>
<th>( 6 )</th>
<th>( 7 )</th>
<th>( \ldots )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N(t), k = 2 )</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>11</td>
<td>19</td>
<td>32</td>
<td>53</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( N(t), k = 4 )</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>12</td>
<td>24</td>
<td>47</td>
<td>91</td>
<td>( \ldots )</td>
</tr>
</tbody>
</table>

A thorough general explanation of how to design a mechanism which attains the bound in the case \( k > U \) and multiple of \( U \) is complex (for reasons that will emerge later on). Hence, in this paper we show how the bound may be achieved through the simple example depicted in figure 4, which refers to the case \( U = 2 \) and \( k = 4 \). As shown in figure 4, at time \( t = 0 \) the source node receives chunk \#1 and serially delivers it to nodes 1 and 2. At time \( t = 2 \) the source node receives chunk \#2; instead of sending it again to nodes 1 and 2, it delivers that chunk to the remaining two neighbors (nodes 13 and 14). This process is repeated for the subsequent chunks, and specifically the odd-numbered chunks are serially delivered to nodes 1 and 2, while the even-numbered ones are serially delivered to nodes 13 and 14. As a consequence of this operation of the source, each neighbor of the source i) receives directly from the source only half chunks, ii) receives a new chunk from the source every 4 time units. As such, neighbors of the source have the necessary extra time to deliver each chunk they receive from the source to all their \( k \) neighbors. The same holds for the remaining peer nodes. As shown in the left tree in figure 4, this allows delivering chunk \#1 to 24 nodes in 5 time units, instead of the previous 19 nodes.

However, as shown in figure 4, chunks are now distributed by means of two distinct unbalanced trees of depth 5 time units: the left one for odd-numbered chunks, and the right one for even-numbered chunks. Hence, a peer node needs to be part of both trees in order to properly receive the full stream. This leads to a complex issue which we call “intertwining problem”, that is: how nodes should be placed in every tree so that the different role of a node in every considered tree does not lead to sharing the node’s upload capacity among the different trees (and hence to performance impairments with
respect to the bound’s prediction, or even congestion).

This is easily seen through the following example. Let us first consider Node 5. In the left (odd-numbered) tree, node 5 is in charge of serving two neighbors, namely 11 and 17. If node 5 were used by the right (even-numbered) tree in place of node 15, it would also need to forward even-numbered chunks to three additional neighbors, thus breaking the assumption that a node must serve at most \( k = 4 \) children. The problem is actually more complex, as we can understand by considering the following second case. In the odd-numbered tree, node 2 has to serve three nodes, namely nodes 5, 8, and 14. At a first glance, we might conclude that node 2 can be also used by the even-numbered tree provided that it is placed in a position of the tree that requires the node to serve only a single node. However, this is not the case. In fact, let us assume to replace node 7 in the even-numbered tree with node 2. This implies that node 2 would be required to deliver an even-numbered chunk to node 24 at every time instant \( t = 6 + 4n \).

However, node 2 is required by the left tree to deliver an odd-numbered chunk at instants of time \( t = 2+4n \), \( t = 3+4n \), and \( t = 4+4n \). Thus, since \( 6+4n = 2+4(n+1) \), node 2 should simultaneously deliver an odd-numbered chunk to node 5, and an even-numbered chunk to node 24, which would not allow reaching the bound.

Unfortunately, the “intertwining problem” for unbalanced trees can not be solved by letting interior nodes of a given tree play the role of leaves in the remaining trees. However, it is possible to prove that i) the tree-intertwining problem can be solved via exhaustive search for arbitrary \( U \) and \( k \) and for any network size for which the bound \( \mathcal{N}(t) \) is attainable, and that ii) there exists a constructive approach which allows finding one of the many possible solutions without relying on exhaustive search. Since such complex proof requires significant extra space and technical elaboration, we leave it for a companion future paper.

V. PERFORMANCE EVALUATION

Figure 5 plots the stream diffusion metric \( N(t) \) as a function of \( T^* \) in a \( U = 2 \) bandwidth scenario, for a single unbalanced tree (\( k = 2 \)), two unbalanced trees (\( k = 4 \)), infinite unbalanced tree (\( k = \infty \)) and a single balanced tree (\( k = 2 \) and parallel transmissions).

The first important observation about figure 5 regards the impact of the number of neighbor nodes \( k \) on the stream diffusion metric bound. The figure shows that there is a significant improvement when moving from the case \( k = U = 2 \) of single tree to that of multiple trees. Interestingly (but expected, as the Fibonacci constants \( \phi_k \) increase only marginally when \( k \) becomes large), the advantage in using more than a few trees is limited: this is especially important if an algorithm is designed to mimic the unbalanced multiple tree operation, as complexity (i.e. signalling burden) increases with \( k \).

The second important observation regards the improvement brought about by serializing the transmissions (and hence unbalanced trees) with respect to parallel chunk transmissions (and hence unbalanced trees). The figure shows that the performance improvement is significant: in the case \( k = 2 \) the stream diffusion metric \( N(t) \) for serial chunk transmissions (i.e., the bound) is one order of magnitude greater than for parallel chunk transmissions at \( t = 20 \), and three orders of magnitude at \( t = 50 \).

---

\(^5\)This is instead the solution when parallel transmission and, as consequence, balanced trees are used [15], being trivial to show that the number of leaves in a tree of fan-out \( k \) is greater than \((k-1)\) times the number of non-leaf nodes.
VI. CONCLUSIONS

In this paper we derived a theoretical performance bound for chunk-based P2P streaming systems. Such bound has been derived on the stream diffusion metric, a performance metric which is directly related to the end-to-end minimum delay achievable in a P2P streaming system. The presented bound for the stream diffusion metric depends on i) the upload bandwidth available at each node, assumed homogeneous for all nodes, and ii) the number of neighbors to transmit chunks to. n-step Fibonacci sequences play a fundamental role in such a bound. The importance of the presented theoretical bound is twofold: on the one hand, it provides an analytical reference for performance evaluation of chunk-based P2P streaming systems; on the other hand, it suggests some basic principles, which can be exploited to design real-world applications. In particular it suggests i) the serialization of chunk transmissions, and ii) the organization of chunks in different groups so that chunks in different groups are spread according to different paths. In a companion paper [8], we have indeed proposed a simple data-driven heuristic, called O-Streamline, which exploits the idea of using multiple paths and relies on a pure data-oriented operation (i.e. chunk paths are not pre-established). Such heuristic attempts to mimic the multiple tree operation described in section IV-B and successfully achieves performances close to the ones of the theoretical bound.

REFERENCES


APPENDIX A

SOME RESULTS ON k-STEP FIBONACCI SUMS

While k-step Fibonacci series have been extensively investigated in related literature, to the best of our knowledge, very few results are available on the sum of k-step Fibonacci series, earlier defined by (2). Since these sums play a fundamental role in our analysis, we derive some new key results concerning them.

Lemma 1: Recursive expression for k-step Fibonacci Sums. Let \( S_k(n) \), with \( n \geq 1 \), be the sum of the first \( n \) terms of a k-step Fibonacci sequence as defined in (1). Then, \( S_k(n) \) may be recursively computed as:

\[
S_k(n) = 1 + \sum_{i=1}^{k} S_k(n-i) \quad \forall n \geq 1 \tag{11}
\]

The proof is based on the mathematical induction. Condition (11) is immediately verified for \( n = 1 \). Hence, let us assume that condition (11) holds for all indices up to \( n \). By applying such condition to \( S_k(n) \) and by using the k-step Fibonacci series definition (1), it is straightforward to prove that (11) holds also for \( n + 1 \):

\[
S_k(n + 1) = S_k(n) + F_k(n + 1) = 1 + \sum_{i=1}^{k} S_k(n-i) + \sum_{i=1}^{k} F_k(n+1-i) = 1 + \sum_{i=1}^{k} [S_k(n-i) + F_k(n+i)] = 1 + \sum_{i=1}^{k} S_k(n+1-i) \tag{12}
\]

Lemma 2: Relation between k-step Fibonacci Sums and k-step Fibonacci Series. The following general relation holds:

\[
S_k(n) = \frac{\sum_{i=1}^{k} (i+1-k) F_k(n+i)}{k-1} - \frac{1}{k-1} \tag{13}
\]

We also observe that the well known result \( S_2(n) = F_2(n+2) - 1 \), relative to the sum of traditional Fibonacci series (i.e., \( k = 2 \)), is the special case of equation (13) (achievable for \( k = 2 \)).

The proof requires some algebraic elaboration. We start by reformulating the linear recurrence (1) as the following difference equation:

\[
F_k(i) + \sum_{j=1}^{k-1} F_k(i+j) - F_k(i+k) = 0 \quad \forall i \geq 1 \tag{14}
\]

Since this equality holds for any \( i \geq 1 \), it holds also for the sum

\[
\sum_{i=1}^{n} \left\{ F_k(i) + \sum_{j=1}^{k-1} F_k(i+j) - F_k(i+k) \right\} = 0 \quad \forall n \geq 1 \tag{15}
\]
In addition, \(S_k(n) = \sum_{i=1}^{n} F_k(i)\) and the following algebraic manipulations may be performed on the left-hand member:

\[
S_k(n) + \sum_{j=1}^{k-1} \sum_{i=1+j}^{n+j} F_k(i) - \sum_{i=1+k}^{n+k} F_k(i) = \\
S_k(n) + \sum_{j=1}^{k-1} \left( \sum_{i=1}^{n+j} F_k(i) - \sum_{i=1}^{j} F_k(i) \right) + \\
- \sum_{i=1}^{n} F_k(i) - \sum_{i=1}^{n+k} F_k(i) + \sum_{i=1}^{k} F_k(i) = \\
(k-1) S_k(n) + \sum_{i=1}^{k} (k-i) F_k(n+i) + \\
- \sum_{i=1}^{k-1} (k-i) F_k(i) + \sum_{i=1}^{k} F_k(i) = (k-1) S_k(n) + \\
+ \sum_{i=1}^{k} (k-i) F_k(n+i) - \sum_{i=1}^{k} (k-i) F_k(i)
\]

Using the last elaboration and then solving equation 15, we achieve

\[
S_k(n) = \frac{\sum_{i=1}^{k} (i+1-k) F_k(n+i)}{k-1} - \frac{\sum_{i=1}^{k} (i+1-k) F_k(i)}{k-1} 
\tag{16}
\]

Equation 13 is now proven by noting that the numerator of the second term can be simplified to 1, taking into account that \(F_k(1) = 1\) and \(F_k(i) = 2^{i-2} \forall i : 2 \leq i \leq k + 1\).

**Lemma 3:** Exact non recursive expression for \(S_k(n)\). We now derive a “Binet-like” exact expression for \(S_k(n)\). As a starting point, we recall that an exact expression has been derived in [9] for the \(k\)-step Fibonacci sequence \(F_k(n)\). This expression, which generalizes the historical Binet’s Formula derived for the case of \(k = 2\), has been conveniently expressed in [10] as

\[
F_k(n) = \sum_{j=1}^{k} \frac{\phi_k^n}{Q_k(\phi_{k,j})} 
\tag{17}
\]

where \(\phi_{k,j}, j \in (1, k)\) are the \(k\) (real and complex) roots of the characteristic polynomial

\[
P_k(x) = x^k - x^{k-1} - x^{k-2} - \ldots - x - 1 = \frac{x^{k+1} - 2x^k + 1}{x - 1} 
\tag{18}
\]

and \(Q_k(x)\) is the following sequence of polynomials

\[
\begin{align*}
Q_2(x) &= -1 + 2x \\
Q_3(x) &= -1 + 4x - x^2 \\
Q_4(x) &= -1 + 6x + 2x^2 - x^3 \\
\vdots & \vdots \\
Q_k(x) &= -1 + 2(k-1)x + \sum_{i=2}^{k-1} (k-i-2)x^i
\end{align*}
\tag{19}
\]

Thanks to the key relation provided in Lemma 2, we can now substitute for \(F_k(.)\) its exact expression (17) in (13), thus obtaining:

\[
S_k(n) = \sum_{i=1}^{k} \frac{(i+1-k)\sum_{j=1}^{k} \phi_k^{n+i}}{Q_k(\phi_{k,j})} = \frac{1}{k-1} - 1 
\]

\[
= \sum_{j=1}^{k} \frac{\phi_k^{n+j}}{(k-1)Q_k(\phi_{k,j})} \sum_{j=1}^{k} (i+1-k)\phi_j^{i} - \frac{1}{k-1} 
\tag{20}
\]

Now,

\[
\sum_{i=1}^{k} (i+1-k)\phi_k^{i} = \frac{\phi_k}{\phi_k - 1} \left[ 1 - \frac{1 - 2\phi_k^{k+1} + \phi_k^{k+1}}{\phi_k - 1} \right] 
\tag{21}
\]

The last fraction in (21) vanishes, as this is the characteristic polynomial (18) computed for one of its roots. Hence, expression (20) simplifies to the final expression:

\[
S_k(n) = \sum_{j=1}^{k} \frac{\phi_j \phi_{k,j}}{(\phi_{k,j} - 1)Q_k(\phi_{k,j})} \sum_{i=1}^{k} \phi_j^{i} - \frac{1}{k-1} 
\tag{22}
\]

**Lemma 4:** Approximate closed form expression for \(S_k(n)\). The exact expression derived in the prior lemma is not handy, as it requires to handle all the complex roots of the characteristic polynomial (18). However, such roots are known to satisfy an important property [9]: only one root has modulus greater than 1. This root (obviously real) is hereafter referred to as \(k\)-step Fibonacci constant \(\phi_k\). For \(k = 2\) it is the most known golden ratio \((1 + \sqrt{5})/2 = 1.61803\); for growing \(k\), it rapidly tends to the value 2 \((\phi_2 = 1.61803, \phi_3 = 1.83929, \phi_4 = 1.92756, \phi_5 = 1.96595, \phi_6 = 1.98358)\). Since all the other real and complex roots have modulus lower than 1, their contribution in either (17) and (22) rapidly becomes negligible as the index \(n\) grows. As a consequence, the following approximate expression holds:

\[
S_k(n) \approx \frac{\phi_k}{(\phi_k - 1)Q_k(\phi_k)} N_k \phi_k^n - \frac{1}{k-1} 
\tag{23}
\]

We remark that this expression asymptotically converges to the exact (integer) sequence, and the approximation becomes negligible (within the unit) even for small values of \(n\). For the convenience of the reader, the first few values of the terms \(Q_k(\phi_k)\) are \(Q_2(\phi_2) = 2.23607, Q_3(\phi_3) = 2.97417, Q_4(\phi_4) = 3.40352, Q_5(\phi_5) = 3.65468, Q_5(\phi_5) = 3.80162\).

**Lemma 5:** Derivation of \(S_k(n)\). A possibility would be to obtain this as the limit of expression (23) for \(k \to \infty\).

Indeed, the computation of this limit is not straightforward because of the tight and non trivial dependency between the parameters \(\phi_k\) and \(Q_k(\phi_k)\), and the index \(k\). A way to circumvent this problem is to algebraically transform (23) into a function of the only variable \(\phi_k\) and then take the limit for \(\phi_k \to 2\). This is possible by exploiting the known property \(k = -\log_2 (2 - \phi_k)\) related to Fibonacci constants. Details are omitted for reasons of space.