Diffusion épidémique de chunks en quasi-direct : la taille compte[†]

Nidhi Hegde,Fabien Mathieu and Diego Perino

Orange Labs, France

Le problème de la diffusion pair-à-pair en quasi-direct consiste à transmettre un contenu à un ensemble de pairs avec la meilleure qualité possible tout en minimisant le différé, c'est-à-dire le délai d'acheminement de la source aux pairs. Dans un grand nombre de solutions, le contenu est découpé en quantités de taille fixée, les *chunks*. En supposant que le temps de transfert d'un chunk d'un pair à un autre est uniquement déterminé par la bande passante de l'émetteur, le délai optimal possible est typiquement en $\frac{c}{s}(\log_2(n))$, c étant la taille des chunks, s le débit du contenu diffusé et n le nombre de pairs. Il semble alors naturel de choisir c aussi petit que possible afin de minimiser le délai. Il arrive cependant un point où les latences présentes dans le réseau influent nécessairement sur la diffusion du contenu.

Notre objectif est de mettre en évidence les phénomènes qui apparaissent lorsque la taille du chunk rend les effets de latence non négligeables. En se basant sur un mécanisme de diffusion épidémique simple, nous mettons en évidence les effets suivants :

– des chunks trop petits empêchent l'algorithme de fonctionner efficacement, et génèrent un taux de pertes important ainsi qu'un gaspillage des ressources réseau ;
– à partir d'une certaine taille, les pertes cessent et la quantité de messages de contrôle se stabilise, le délai étant proportionnel à la taille du chunk ;
– entre les deux se situe un intervalle de tailles adaptées à la diffusion. Le choix d'une taille précise dépend du compromis à réaliser entre pertes et délai.

De plus, nous observons que l'introduction d'un certain parallélisme dans la diffusion permet de déplacer la zone utile, augmentant ainsi la performance de l'algorithme.

Keywords: Pair-à-pair, live, taille des chunks, efficacité

1 Introduction

Peer-to-peer (P2P) live streaming applications have become very popular in the last few years. A key issue is whether the delay and quality requirements can be met by a P2P protocol. Most of P2P live streaming algorithms split the stream into atomic units of data called *chunks* and a peer can only send chunks it has fully received. In such systems one of the main goals is thus to design an efficient chunk exchange policy.

In addition to the scheduling policy, other specific parameters, like the chunk size, the receiver buffer size, the number of peers to probe and so on, matter too. For the BitTorrent file-sharing protocol it has been shown in [**?**] that small chunks are not always the best choice. In this paper, we propose to perform a similar study for the P2P live streaming. We focus on the epidemic exchange policy [**?**], where a so-called *scheme* indicates which chunk a given peer should send to whom. We show that chunk size significantly impacts the performance. In fact, there exists a *suitable range* of chunk sizes, where the specific choice of the chunk size ultimately depends on the desired delay/loss trade-off. The impact of the number of peers to probe and the number of simultaneous upload connections is considered too.

2 Methodology

Epidemic diffusion schemes and their behavior have been extensively studied in literature. See [**?**] for a detailed study and references therein. Here, we focus on particular scheme called *random peer / latest useful chunk* (*rp/lu*) : a given sender peer chooses a recipient peer uniformly at random among its neighbors, to

[†]This work is supported by the project NAPA-WiNe (FP7-ICT-2007-1, grant 214412).

which it sends its most recent chunk not own by the recipient peer. *rp/lu* has the advantage of being fairly simple, allowing us to focus on the impact of parameters like the chunk size. Moreover, it is efficient in terms of diffusion rate and delay, and it generates low overhead [**?**].

In our analysis, we used an event based simulator developed by the Telecommunication Networks Group of Politecnico di Torino [‡]. The simulator has been modified to take network latencies, control overhead and parallel upload connections into account.

In our simulations, there are $n = 1000$ peers and a unique source S. The overlay network is an Erdös-Renyi graph $\mathcal{G}(n+1, p)$. We set p to 0.05, so each node is connected to about 50 neighbors. The source injects a stream of rate $s = 0.9$Mbps, split into chunks of size c. All peers have the same upload capacity $u_i = 1.03$Mbps, and unlimited download capacity (those are typical bandwidth values used in other papers, see for instance [**?**]).

Every peer periodically selects a subset m of its neighbors at random (the probe set), and requests their chunk maps. Based on the responses, the peer then transmits the latest useful chunks if any. The overhead required for control message exchange is taken into account (we assume that any control message has the same size $c_c = 1$kb).

When transmitting, a peer fairly shares its upload bandwidth among at maximum $m' \leq m$ peers (randomly selected if $m' < m$), which are served in parallel. If a peer finds less than m' suitable recipients because it does not have enough useful chunks, it can upload the chunks faster (since there are less than m' active connections), but it may stay idle for a period of time after that (it needs to acquire new chunk maps from newly selected peers).

For chunk and chunk maps transmissions, both transmission and network latency are considered. The network latency values are taken from the Meridian project dataset [§]. We suppose that every peer has a buffer of size 300 (expressed in chunks) in order to avoid possible losses due to buffer shortage. This implies a buffer size proportional to the chunk size.

3 Chunk size and performance

When considering a streaming algorithm, a crucial performance metric is the rate/delay/overhead trade-off achieved by that algorithm, which can be summarized by a (*loss*,*delay*,*overhead*) triplet. Loss can be defined as the probability to miss a chunk, while a possible definition for the diffusion delay is the time needed for a chunk to reach a peer on average. Lastly, we define the overhead as the difference between the bandwidth used by peers (throughput) and the actual data received (goodput).

3.1 Experiment

As a first experiment, we analyze loss, delay and overhead as a function of the chunk size. The results are shown in Figure 1, with $m = m'$ varying from 1 to 5.

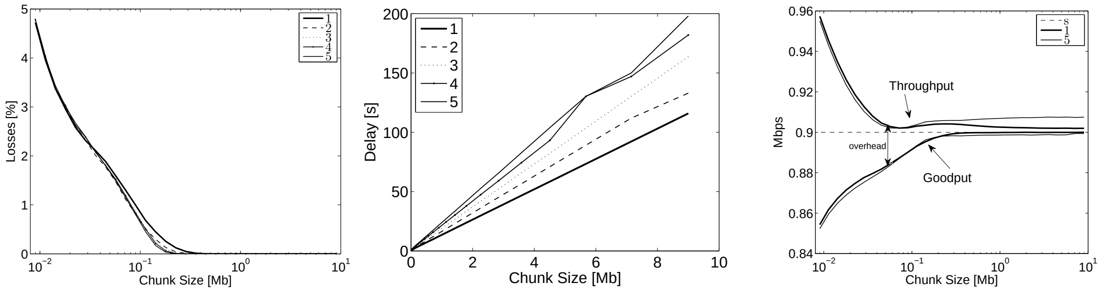
Losses (Figure 1a) Two distinct phases may be observed :

- For large chunks (in our experiment, c greater than a few hundred kilobits), there are no losses.
- As the chunk size goes below a certain critical value, losses start to appear, roughly proportional to the logarithm of the chunk size (at least for the chunk range considered).

This phenomenum can be explained as follow : the time between two consecutive chunks is c/s, and is therefore proportional to the chunk size c. When c increases (all other parameters being the same), more and more control messages per chunk can be exchanged between peers. This should achieve a proper diffusion, providing that enough bandwidth is available, because a sender peer has enough time to find a neighbor needing a given chunk. On the contrary, when c/s is too small, peers do not have enough time to exchange control messages, resulting in losses. Note that increasing the probe set m slightly improves the performance.

‡. `http://www.napa-wine.eu/cgi-bin/twiki/view/Public/P2PTVSim`

§. `http://www.cs.cornell.edu/People/egs/meridian/`



(a) Losses (b) Delay (c) Overhead (througput minus goodput)

FIGURE 1: Losses, delay and overhead as functions of the chunk size

Delay (Figure 1b) The diffusion delay is roughly proportional to the chunk size (this is why a linear x-scale is used), and increases with m. Note that the observed delays are proportional to the minimal broadcasting delay of a single chunk when RTT is neglected, which is $d_{\min} = \frac{m \ln(n)}{\ln(1+m)} \frac{c}{s}$.

Overhead (Figure 1c) The overhead is the difference between the throughput and goodput. Only the curves for $m = 1$ and $m = 5$ are displayed for readability. For very small chunks, we have a non-intuitive trend, where as c grows, the goodput increases *and* the throughput decreases (the overhead decreases faster than the goodput increases). This process slows down so that at some point the throughput increases again. For big enough chunks, the overhead becomes roughly constant (for a given m).

The intuition behind this is that for very small chunks, losses are high, which, as mentioned earlier, come from the fact that not enough control messages can be sent. Asymptotically, we may imagine that only one control message per sent chunk is issued, resulting in an overhead/goodput ratio of $\frac{c_c}{c}$. On the other hand, for chunk big enough, we may expect that a peer can send the required number of messages per sent chunk, that is proportional to the chunk characteristic time c/s. This would result in an overhead ratio proportional to $\frac{c_c}{s}$, and thus independent of c (but not of other parameters like the median RTT or m).

3.2 Suitable range for c

In light of the study above, we deduce an order of magnitude for suitable chunk size in epidemic live streaming. For the parameters considered here, c should be greater than 0.06 Mb (which corresponds to about 15 chunks per second) and smaller than 0.3 Mb (3 chunks per second) :

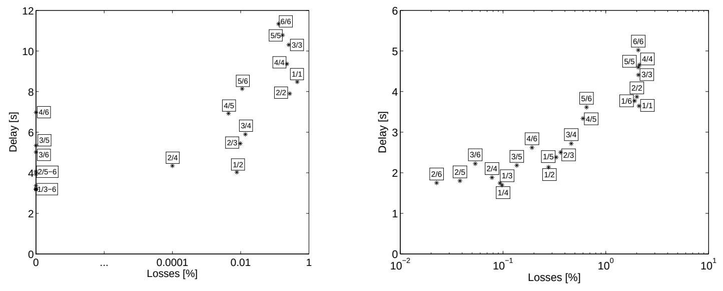
– to send the stream at more than 15 chunks per second is good for the delay (which stays roughly proportional to c), but results in both an increase in throughput and a decrease in goodput ;
– goodput and throughput are stationary for c greater than 0.3 Mb : using bigger chunks only means longer delay ;
– between these values, the choice of c results in a loss/delay trade-off : smaller delay with some losses or greater delay with no loss. The optimal value for c depends then on parameters that will not be discussed here, such as the codec used, the required QoS, etc.

We observe that the suitable range for chunk size begins when the chunk characteristic time ($\frac{c}{s}$) has the same order of magnitude than the median RTT, and ends an order of magnitude later. In other experiments, we scaled the RTT distribution used in order to observe the evolution of the range with the median RTT. The results showed that the range values are indeed roughly proportional to the median RTT.

4 Size of the probe set

In the results presented so far, we have assumed that the number of simultaneous exchange chunks, m', is identical to the size of the probe set m. We now consider the impact of probing more peers than the targeted number of simultaneous chunk to send. A larger probe set gives a peer a higher chance of finding receivers collisions (power of choices principle). However, it also increases overhead, and possibly delay.

Figure 2a plots the loss and delay trade-off for various m'/m pairs. The scheme is *rp/lu*, the bandwidth is homogeneous and the chunk size is set to $c = 0.15$ Mb (middle of the suitable range). The figure shows that using $m' = m$ is not optimal, and having a larger probe set, $m > m'$ significantly reduces both delay and losses. The delay decreases from about 10 s for the $m = m'$ cases, to less than 4 s for $m' = 1$ and $m = 3,\dots,6$. Regarding the losses, there are some (m'/m) couples for which no loss could be observed in our experiment : $1/3,\dots,6$, $2/5,6$, $3/5,6$, $4/6$. This suggests that a consequence of using $m' < m$ is a shift of the suitable range for c.



(a) $c = 0.15$ Mb (b) $c = 0.035$ Mb

FIGURE 2: m'/m loss/delay trade-off for two values of c

In order to verify that understanding, we now set $c = 0.035$ Mb, which is clearly below the suitable range observed in § 3.2 for $m = m'$. The results are shown in Figure 2b.

We observe that no couple m'/m can achieve lossless diffusion for such a small c, however the loss/delay trade-off is clearly improved for some couples : using $m'/m = 2/6$, we get a delay of 1.7 s with a loss probability of about 0.02 %. This suggests that $c = 0.035$ Mb is definitively within the suitable range for $m'/m = 2/6$.

Also note how the relative efficiency of the different m'/m values is impacted by the choice of c : for instance, $1/6$, which is optimal for $c = 0.15$ Mb, performs rather poorly for $c = 0.035$ Mb.

As for the overhead (not displayed in Figure 2), we observed that for a given m, it stayed close to the one observed in Figure 1c going below the suitable range defined in § 3.2, and is thus higher for a small c. So reducing c with $m' < m$ can reduce the delay, but it requires more throughput.

5 Conclusion

We have investigated the dissemination parameters of P2P epidemic live streaming through extensive simulations. We have shown that the chunk size significantly impacts performance and should fall within a given range which is mostly determined by the median RTT of the network and the streamrate.

We have also shown the size of the probe set affect performance of diffusion schemes, and, in particular, a probe set larger than the actual number of concurrent connections may improve loss/delay performance by modifying the suitable chunk size ranges.

Références

[BMM$^+$08] Thomas Bonald, Laurent Massoulié, Fabien Mathieu, Diego Perino, and Andrew Twigg. Epidemic live streaming : optimal performance trade-offs. In *Proceedings of the International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)*, 2008.

[MLLK08] Pawel Marciniak, Nikitas Liogkas, Arnaud Legout, and Eddie Kohler. Small is not always beautiful. In *Proceedings of the Seventh International Workshop on Peer-to-Peer Systems (IPTPS)*, 2008.

[PM08] Fabio Picconi and Laurent Massoulié. Is there a future for mesh-based live video streaming ? In *Proceedings of the Eighth International Conference on Peer-to-Peer Computing*, pages 289–298, Washington, DC, USA, 2008. IEEE Computer Society.