Four Months in DailyMotion: Dissecting User Video Requests

Yannick Carlinet[&], The Dang Huynh[†*], Bruno Kauffmann[&],
Fabien Mathieu[‡*], Ludovic Noirie[†*], Sébastien Tixeuil[§*]
[&]Orange Labs [†]Alcatel-Lucent Bell Labs France [‡]Inria [§]UPMC Sorbonne Universités, IUF [*]LINCS
Contact: fabien.mathieu@inria.fr

***Abstract*—The growth of User-Generated Content (UGC) traffic makes the understanding of its nature a priority for network operators, content providers and equipment suppliers. In this paper, we study a four-month dataset that logs all video requests to DailyMotion made by a fixed subset of users. We were able to infer user sessions from raw data, to propose a Markovian model of these sessions, and to study video popularity and its evolution over time. The presented results are a first step for synthesizing an artificial (but realistic) traffic that could be used in simulations or experimental testbeds.**

***Index Terms*—UGC, Dataset analysis, modeling**

I. Introduction

Internet traffic has tremendously grown in volume in the recent years. A large part of the increase comes from User-Generated Content (UGC) systems, like YouTube or Dailymotion, which are estimated to account for 20 to 35% of current Internet traffic [1]. In this context, understanding the properties of UGC traffic is important for network operators (dimensioning networks), content providers (ensuring quality of experience) and equipment suppliers (designing adapted solutions). In order to reach these objectives, we analyze a dataset from a real UGC system, and we propose a model that is a first step for synthesizing an artificial (but realistic) traffic, which could be used in simulations or experimental testbeds.

In more details, the analysis we propose is based on a four-month dataset that monitors more than 15,000 users, tracking the network requests for DailyMotion videos. Thanks to this dataset, we are able to partition the requests into user sessions, for which we model the arrival process. We can distinguish jumps within a video from requests of another video, and we provide a simple Markovian model that reproduce the jump/zap/end process within a session. We also provide a basic analysis of popularity, pointing out a few patterns that summarize the observed evolution of popularity with time. The possible interactions with the considered aspects (session arrival, behavior within a session, video popularity) are briefly investigated. Although the main presented results are based on one specific dataset, the proposed methodology is sufficiently versatile to be applied to other similar real datasets, and opens the way to generating synthesized, yet realistic, datasets.

The paper is structured as follows: section II describes the DailyMotion dataset and a preliminary classification. Section III analyzes and proposes a model for the session arrival process. The behavior of a user within a session is discussed in section IV, and the popularity of videos is addressed in section V. We review the related work in section VI. Finally, section VII concludes the paper and discusses how the obtained results could be exploited in a modular way in order to emulate a traffic that looks like what we can observe in real networks.

The work presented in this paper has been carried out at Laboratory of Information, Network and Communication Sciences (LINCS, www.lincs.fr).

II. Parsing the traces

While our approach for modeling user behavior can be applied to most existing traces, we illustrate and validate it using some real data. This section presents the dataset we used and how raw data was pre-processed for analysis.

A. Dataset description

Our dataset was gathered through 7 probes set in the edge network of Orange, a tier-1 operator, either in the Asymmetric Digital Subscriber Line (ADSL) for 6 probes or the Fiber To The Home (FTTH) edge network for the seventh probe. Each probes was connected to the port of a Broadband Remote Access Server (BRAS) thanks to an optical splitter, which means that the probes saw only duplicated traffic and did not interfere with operational traffic in any way. The collected traffic amounted to about one tenth of all traffic going through the BRAS: however the traffic of a given customer was always switched through the same port, hence all traffic of monitored customers was collected. The ADSL probes collectively monitored about 10,000 customers and the FTTH probe about 5,400 customers. The dataset we analyze in the sequel focuses on the requests made on the DailyMotion website [2]. It spans a four months period, from 02/02/2010 to 05/31/2010, during which 4,948,593 events were recorded.

The probes are designed to monitor web streaming events triggered by HTTP requests of a flash file (.swf extension). For each streaming event, the following information was recorded:

- Client ID, based on the layer 2 protocol in use (ATM (resp. Ethernet) for ADSL (resp. FTTH) probes);
- HTTP request, *i.e.* the identifier of the requested video;
- Timestamp of the request;
- Number of bytes downloaded through the request.

The fact that the client identification is based on layer 2 protocols means that it is insensitive to IP address changes due to dynamic address allocation.

Finally, for privacy reasons, every piece of information about clients and requested objects was masked in the trace, before any further processing.

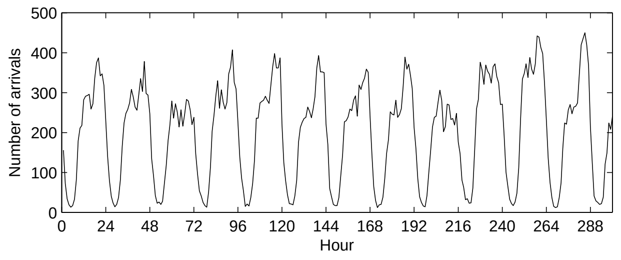


Fig. 1. Number of session arrivals per hour over a few days

B. Event classification

Analyzing the datasets, we were able to distinguish three main types of events:

- *Launchers*: before getting the actual video, the browser first downloads a flash video player (the launcher), which then makes video requests. The same launcher can perform several requests. We observed 1,027,847 launcher events (*i.e.* about 20% of the events).
- *Request for a new video* (or *change* event): this is a request for a real video (not a launcher) that differs from the previous one (if any) of the user. We monitored 2,015,985 such events (*i.e.* about 40% of the events).
- *Jumps*: frequently, a given user can make several *consecutive* requests for the same file. We interpret these requests as *jump* event: the user jumps to a part of the video that has not been buffered yet. We observed 1,904,761 of these events (*i.e.* about 40% of the events).

C. Extracting user sessions

For a finer grained analysis, we first separate the events originating from a given user into logical *sessions*. Being able to split the requests into sessions is fundamental, as we expect the process of session arrivals to be quite different from the behavior within a given session. A natural idea would be to use *launchers* events as sessions separators. However, launcher events do not accurately correspond to user sessions:

- a user may leave its computer for some time and start a new viewing sequence later, so we may have multiple human sessions following one launcher request;
- conversely, one user may reload its current page or navigate through multiple browser windows/tabs, resulting in multiple launcher requests for one human session.

Therefore, we prefer to infer sessions from users idle periods: if a user does not make any new video request for some time, we assume that its session has ended. The main problem remains to define idle periods. Empirically, we found out that using a fixed idle period of one hour gave good and reliable results. Using that threshold, we were able to extract 567,510 sessions from the traces, which is to compare to the 1,027,847 launcher events.

III. Session arrival process

We first begin to study the session arrival process, which is determined by the first event of each session.

A. Poisson modeling

There is no *a priori* reason to assume any kind of dependency between arrivals, so Poisson is a natural modeling choice [3]. However, we cannot use a simple homogeneous hypothesis, as the intensity of the process is deeply affected by the time of the day, with a 24-hours pattern and orders of magnitude between the intensity of late night and prime time hours (Figure 1).

A good modeling candidate is therefore a heterogeneous Poisson process. We assume that the arrival intensity is given by some function $\lambda(t)$ (expressed in number of expected arrivals per second). We remind that in a heterogeneous Poisson process, the probability of having k arrivals between t_1 and $t_2 > t_1$ is given by:

$$P(k, t_1, t_2) = \mathcal{P}(k, \lambda_{t_1}^{t_2}), \text{ with } \begin{cases} \lambda_{t_1}^{t_2} = \int_{t_1}^{t_2} \lambda(t) dt, \\ \mathcal{P}(k, \lambda) = \frac{\lambda^k}{k!} e^{-\lambda}. \end{cases} \quad (1)$$

So if the function $\lambda(t)$ is known, is is easy to produce artificial traces that will look just as the real ones. However, in practice, the actual $\lambda(t)$ is hidden and difficult to estimate.

B. Low intensity variation

As a workaround, we split the timeline into intervals, and we assume the intensity variation is sufficiently low to be considered constant in each interval. In more details, we consider a time partition $\mathcal{I}$. For any $I \in \mathcal{I}$, if $|I|$ denotes the duration of I and $r(I)$ the number of arrivals observed during I, we assume that for $t \in I$,

$$\lambda(t) \approx \lambda_I := \frac{r(I)}{|I|}. \quad (2)$$

The choice of a good partition remains tricky: if the duration of an interval is too small, the estimate will be too noisy to be useful. On the other hand, the constant assumption is likely not to hold for large intervals. Based on our traces, we found that partitioning the timeline into plain hours was a good trade-off.

A few remarks here:

- Our framework allows to choose intervals of heterogeneous durations. Indeed, using a partition with heterogeneous interval duration depending on the time of the day gives better results, but for the sake of simplicity, we only present the homogeneous interval case.
- Assuming that arrival intensity is roughly constant in a given interval is reasonable for on-demand content (as it is the case here). It would probably not be valid for monitoring live streaming channels, where scheduled events can induce highly localized bursts of arrivals.

C. Noisy pattern modeling

The main drawback of using plain Equation (2) is that it produces a lot of parameters (2880 for the 120 days trace used in this paper). Of course, this allows to produce an artificial trace that looks *exactly like* the original one, but it makes it difficult to generate a realistic trace with other characteristics.

In order to reduce the set of parameters, we propose to use a noisy daily pattern: we consider the set $\mathcal{I}_h$ of a given plain hour $[h, h+1]$ through days. For $I \in \mathcal{I}_h$, we use

$$\lambda_I = \frac{G(\mu_h, \sigma_h)}{|I|}, \quad (3)$$

where $G(\mu_h, \sigma_h)$ is a random truncated normal variable with the same mean μ_h and standard deviation σ_h.

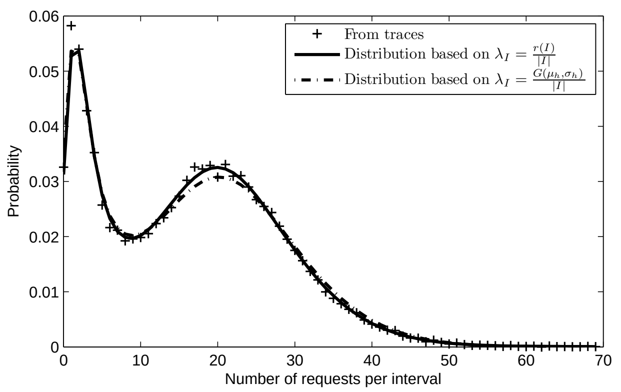


Fig. 2. Number of request per five minutes interval (dataset and models)

Using (3), one can simulate arrivals for any arbitrary period, or change the daily pattern as needed. However, this simplification erases any correlation between hours. A more elaborate modeling would consist in measuring these correlations and reproducing them by combining multiple random variable (*e.g.* daily and hourly intensities values). Nevertheless, we found out that the gain in accuracy is not worth the increased complexity of the model, and that keeping a sparse model with a "low" number of parameters is justified by the fact that correlation between adjacent hours seems to be small.

In order to validate our approach, we study the number of arrivals observed over a five minutes interval (which can be valuable for dimensioning purposes). Figure 2 displays the distribution: observed for the real trace; obtained using (2); obtained using (3).

The distribution based on (2) fits the observed data pretty well. Equation (3) also manages to stay quite close to the real trace, providing a good trade-off between accuracy and sparsity.

D. Temporal dependency

For completeness, we also study the relationships between sessions of a given user. For that purpose, we extract what we call *intersession duration*, which is the time between the last video request of one session and the first video request of the next session *from the same user*. Figure 3 shows the distribution of the intersession durations. While there is a lot of short intersessions (the *heavy* users, which frequently watch UGC, tend to have short intersession durations), local peaks appear at regular intervals, which roughly correspond to one day (1 day = 1440 minutes). This indicates that some users have the habit of viewing UGC content at a roughly fixed time of the day (*e.g.* after lunch or work). This temporal correlation may seem contradictory with the Poisson approximation we made, but it is known that the superposition of independent users tends towards a Poisson process when the number of users is large enough, even if each user generates events that are loosely time-correlated [4], [5].

IV. Session modeling

We now study user behavior within a session.

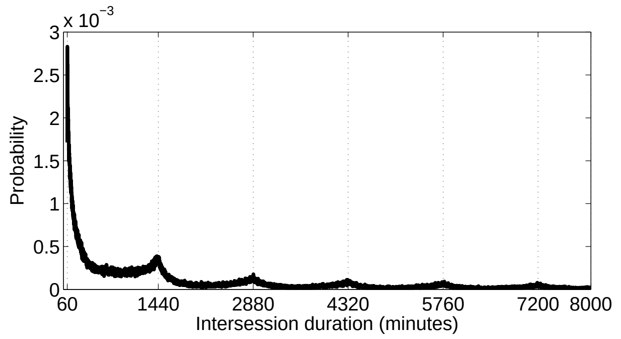


Fig. 3. Intersession duration

A. Session duration

We first focus on the session duration, which is an unknown quantity in the traces that we need to infer. Simply using the difference between the time of the first event and the time of the next event would not be very precise, as a lot of idle time can occur between two sessions (cf Figure 3). Hence, we prefer to estimate the session duration using the average time between two events in the session, assuming here that events are roughly regularly spaced in average. For a session S with n events ($n > 1$), this gives a session duration

$$D_S = \frac{(T_{S_n} - T_{S_1}) \times n}{n-1}, \qquad (4)$$

where T_{S_i} denotes the timestamp of the i^{th} event in S.

Note that if S' is the session that follows S (for one given user), D_S is always smaller than $T_{S'_1} - T_{S_1}$, so the duration estimate ensures that one session will not overlap with the next one. This is a direct consequence of using a fixed threshold for delimiting sessions.

Unfortunately, we cannot use (4) for single-event sessions, which represents 34% of the sessions. We tried to infer a duration based on the number of downloaded bytes, but the results were not reliable enough.

Figure 4 displays the complementary cumulative distribution function of the session duration obtained from (4). One observes the following properties:

- many sessions are short and last only a few minutes; the median duration is about 7 minutes;
- a large minority of the sessions are of intermediate length (25% of the sessions last more than half an hour, and 13% more than one hour);
- there is a non negligible proportion of lengthy sessions (3.3% last more than two hours, 1% more than 3.5 hours).

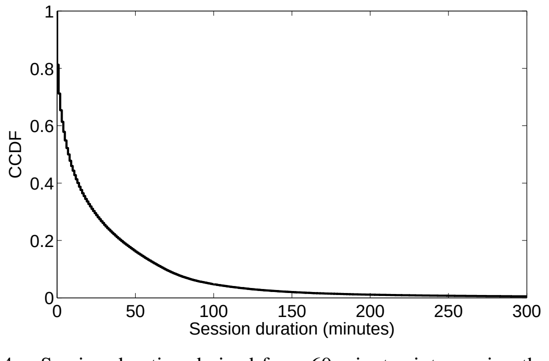


Fig. 4. Session duration derived from 60-minutes intersession threshold

B. A Markovian model for user behavior

For modeling the sequence of events that occur within a session, we propose to use a Markovian approach: to each session S of length n, we associate the sequence $(E(S_i))_{1 \leq i \leq n}$, where $E(S_i) \in \{C, J\}$ denotes the type (change or jump) of the i^{th} *event* in S. Then, for any sequence $(E_j)_{1 \leq j \leq k} \in \{C, J\}^k$ of (arbitrary) length k, that is a prefix of at least one session sequence, we compute over all sessions that include that prefix the probability of the next event in this session: change, jump, or none (which indicates the end of the session).

The corresponding Markov chain describes as precisely as possible the observed sequences of events, and could be used to generate artificial sequences. The main issue with this naive approach is its size, which can be up to $2^{n_{\max}-1}$ states ($n_{\max}$ standing for the maximal sequence size). However, we found that the Markov chain was highly susceptible to state reduction, with a reasonable impact with respect to accuracy. After reduction, we propose the Markov chain described in Figure 5, which has only five-states (plus one implicit end-of-session state), and differs from the original one from less than 1% (difference between sequence distributions). In more details, the Markov modeling gives us the following insight:

- after the first event, the three possible outcomes (change, jump, or end session) are roughly equiprobable.
- The states called `C1` and `C2+` correspond to a C event. `C1` can loosely be seen as *this is the first change event after the first event*, while `C2+` roughly corresponds to *at least one change event has been made before, in addition of the (likely) first event*. However, the matching between states and proposed interpretation is not exact, as the Markov chain allows to reach `C2+` without passing by `C1`, and `C1` from `C2+`. That being said, one can observe that while the transition from start to `C1` is 32%, it increases to 51% from `C1` to `C2+`, and then 65% from `C2+` to itself. The interpretation is that you are more likely to change the video you are watching if you already did it before (but after more than two consecutive changes, that increase is not noticeable).
- The states called `J1` and `J2+` corresponds to a J event. `J1` is (exactly) the first jump observed on a given video, while `J2+` gathers the subsequent jumps. Like for the *change* states, one can observe that the probability of jumping increases if you already have jumped before: the probability of the first jump within a given video is between 20% and 34% (depending on the departure state), but it increase to 51% for the second jump and 78% afterwards.

V. Video popularity

Until now, we have not considered the popularity of the requested videos. Hence, this section examines the popularity distribution, from both a global and temporal point of view.

Based on the information available in our dataset, we may think of several ways for computing the popularity of one given video, like for instance:

- *event-based* popularity: all events regarding one video are used (change, but also jump events);

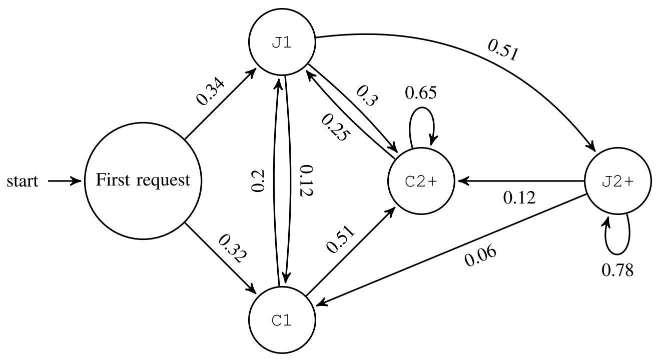


Fig. 5. Markov chain describing user behavior within a session

- *session-based* popularity: one uses the number of sessions where one video was requested. This prevents from counting jumps, but also multiple requests within the same session even if the user requested another video in-between;
- *user-based* popularity: for each video, we consider the set of users that have requested that video at least once in the dataset. This allows to even fan users that may request one video over multiple sessions.

Note that if $n_e(v)$, $n_s(v)$ and $n_u(v)$ denote the number of events, sessions and users recorded for video v, one have

$$n_e(v) \geq n_s(v) \geq n_u(v). \qquad (5)$$

In the following, we focus mainly on the session-based popularity.

A. Global popularity

The dataset contains about 440,000 distinct requested videos. Figure 6 displays a log-log view of the number of sessions and users recorded for each video, sorted by decreasing number of sessions. One can observe a heavy tail: while the average number of sessions per video is 1.29, the most popular video is involved in 2,381 sessions, and 338 videos belong to more than 100 sessions. On the other hand, a large majority of the videos (about 300,000) are requested only in a single session. User-based popularity looks pretty much like the session-based one up to some scaling factor. That factor remains reasonably steady with the rank.

It is interesting that the observed video distribution looks fully heavy tailed: while power-law like distributions are frequent in UGC, the tail is often reduced with an exponential cutoff (see for instance [6]; the same kind of result for large-scale VoD system was also observed in [7]).

This difference could come from the way the dataset was obtained (crawling the website in [6], or probing HTTP requests here). The website may make videos that are not popular hard to crawl. Alternatively, it is possible that we failed to identify the same video under different names in our dataset. The distinct distributions could also be derived from distinct user habits in both experiments (in [6] and reference within, an explanation based on behavior was proposed for the exponential tail).

B. Temporal patterns

The global video distribution does not detail how requests are distributed through the dataset. A video cannot be re-

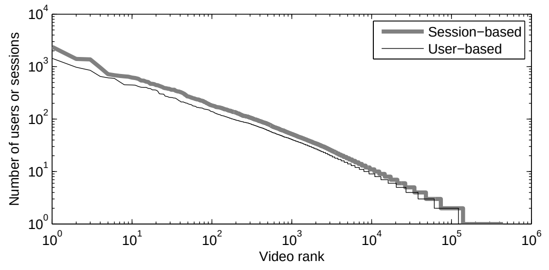


Fig. 6. Videos ranking over video session frequency

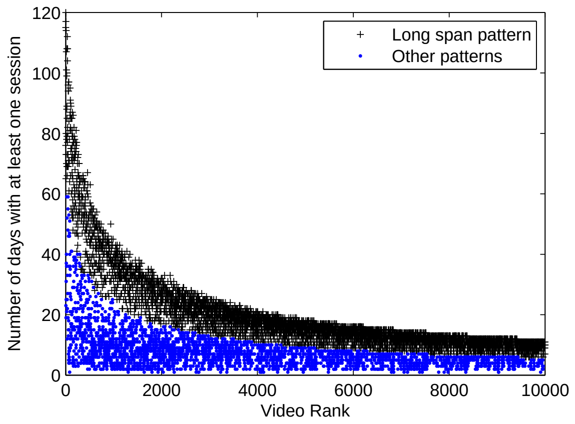


Fig. 7. Number of active days (with at least one session)

quested until it has been uploaded, and it is expected that even popular video will not be requested forever.

We propose to investigate how the popularity of videos evolves with time. As the popularity evolution of very rarely requested videos is trivial, we limit ourselves to the first 10,000 most popular videos.

Figure 7 shows this number for each video, the number of *active* days where at least one session contained that video, as a function of the video rank. First, an upper limit function appears: this limit is indeed driven by the number of sessions, which is an upper bound for the number of active days. Then, we can use this figure to split videos into two categories:

- Some videos are of the same order of magnitude than the upper limit, which means that the average number of sessions per day is low but covers a lot of days. We say these videos admits a *long span* pattern. We classify as *long span* the videos with a number of days greater than half the upper limit (the constant can be adjusted), and observe 7106 of them among the 10000 most popular. For instance, Figure 8a displays a typical popular long span pattern: the number of sessions per day is noisy, but relatively steady.
- The other videos exhibit some temporal concentration:
  - some can be highly, but shortly popular (like a video promoted for a shot duration on the front page of a popular site). We call that a *bursty* pattern. Figure 8b displays a typical bursty video;
  - others may have been uploaded and/or removed during the dataset, resulting in a shorter range than expected. We call that type *average span* pattern. Figure 8c displays a typical average span video.

We made the distinction between bursty and average span patterns by monitoring the ratio of the active days where the activity was greater than 10% of the maximal activity. This allows to clearly separate the two kinds of patterns, into 1468 bursty patterns and 1426 average span patterns.

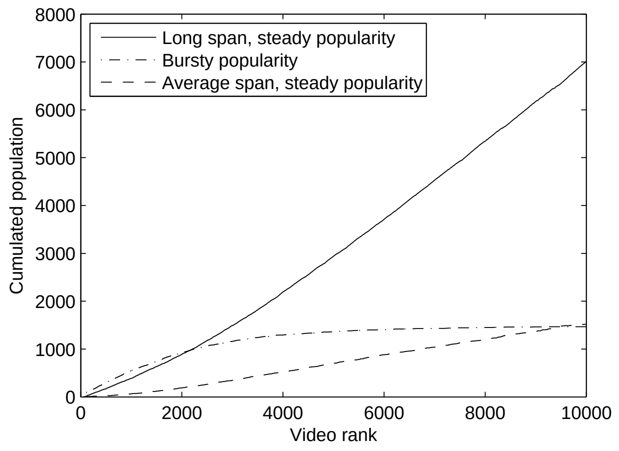


Fig. 9. Cumulated population for each pattern

Still, it is possible that long span and average span patterns belong to the same family, separated by the finite duration of the dataset. A longer dataset is needed for answering that question, which we leave for future work. We assume the two patterns are distinct, but are aware that the distinction may be artificial.

At last, we should note that there is a correlation between ranking and patterns. Figure 9 shows the cumulated population for each pattern. In particular, one should note that the bursty pattern is typically found among the most popular videos: it represents roughly one half of the 2000 most popular videos, while less popular videos are mostly long or average span ones. This correlation between pattern and popularity should be taken into account if one plans to design a request generator that aims at mimicking real-life requests.

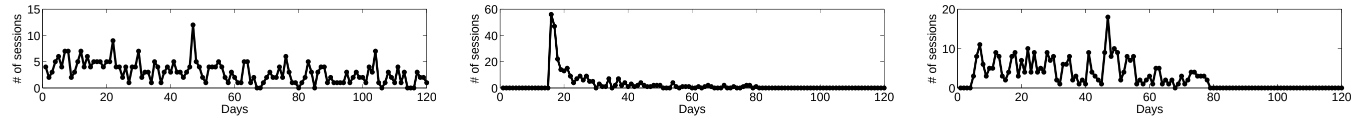
C. Correlations with previous modeling aspects

If one wants to jointly use the models presented in this paper, it is important to consider all possible correlations that may exist. For instance, is the Markov chain, or the popularity pattern, impacted by the time of the day? Is the Markov chain impacted by the popularity pattern? We investigated the question, and found some small correlations here and there, but in the end, most correlation can be neglected in a first approximation (except for the pattern/ranking correlation we just saw).

VI. Related work

The vast majority of the literature investigating UGC systems is dedicated to YouTube due to its popularity and lifespan. To our knowledge, this paper reports the first extensive study that is relative to DailyMotion.

Our approach to finding session durations is similar to that of Gill *et al.* [8] but we end up in choosing a 60 min threshold rather than a 40 min (for 2008 traces of YouTube). Also, [8] used a simple ON/OFF model for users within a session, while we crafted a more detailed Markov model. Another approach



(a) Long span pattern (b) Bursty pattern (c) Average span pattern

Fig. 8. Video popularity patterns

was proposed by Abhari and Soraya: they investigated the possibility of generating UGC workload that corresponds to statistical user behavior [9] inferred from website crawling. Then, the simulated user simply randomly chooses video files to view based on their popularity, until the expected session time has passed. By contrast, our approach is based on actual user behavior, captured by passive network measurements, and is able to capture fine-grained metrics such as jump actions and early video file change. This kind of low level user behavior was also investigated for the case of YouTube with a short timed 35h [10], but only the expected number of jumps per video or the expected time before viewing abort were reported. Video popularity was mostly studied by actively crawling a UGC site for an extended period of time (*e.g.* [9], [11] for Youtube). Our findings are roughly similar to what other observed for YouTube, although we use passive monitoring to gather data.

VII. Conclusion

In this paper, we presented an analysis of a four months DailyMotion dataset based on monitoring video requests from 15,000 distinct users. The analysis showed that the requests can be understood as the combination of three elementary models: session arrivals that obey a heterogeneous Poisson process; session behaviors that follow a Markov chain; video popularity that follows a heavy-tailed distribution, with strong temporal correlations, especially for popular videos (bursty patterns).

The existence of these models open the way for a versatile modular request generator that can be used for anticipating and testing new solutions, and in particular for benchmarking prospective scenarios (growth prevision, anticipation of change in user behaviors,...). Such a request generator can be decomposed into the following modules:

- *user arrival generator*, which tells when new sessions start,
- *session builder*, which describes the events within a session (number of videos, jumps) with duration aspects,
- *video selector*, which decides which videos are selected, taking into account the popularity in time.

Each module can use either data from real traces to replay them, or produce emulated data by using the models defined in our paper. Combining real datasets and artificial ones is thus possible. Our dataset indicates that there is no strong correlation between the three aspects involved. This independence, which we hence assume for the moment, should simplify a lot the implementation of the full trace emulator, and facilitate the interleaving of real and artificial inputs.

For the user arrival generator, (2) or (3) allows to scale the intensities in order to artificially increase the number of users. Note that it is easy to use other daily patterns or intensities distribution for generating sessions in order to get regional profile (our pattern might be biased by the profile of monitored users), or to take into account seasonal variations or previsions of long term evolution.

The session builder can use real input by replaying one session from the dataset, or use the Markov chain approach. For the later, we did not present here the inter-event distribution, but such a distribution must of course be embedded in the builder (cf *e.g.* [6]). Changes in the user behavior, due *e.g.* to automatic advertisement videos or high speed network leading to a higher buffering rate and lower jump probabilities) can be embedded in the Markov chain. We also expect that Markov chains can be build to describe the user behaviour in other type of systems, such as Netflix.

The video selector can be designed to handle dynamic popularity, by planning the introduction of new videos in the system. For each new video, the selector should "predict" its total popularity and its pattern (which may be correlated to its popularity). Using these popularities modulated by their pattern, the selector should be able to reproduce both the global popularity and the temporal patterns. Arbitrary popularities and pattern distributions can obviously be used.

References

[1] A. Finamore, M. Mellia, M. M. Munafò, R. Torres, and S. G. Rao, "Youtube everywhere: impact of device and infrastructure synergies on user experience," in *IMC '11*, 2011, pp. 345–360.

[2] Dailymotion, http://www.dailymotion.com.

[3] C. Park, H. Shen, J. S. Marron, F. Hernandez-Campos, and D. Veitch, "Capturing the elusive poissonity in web traffic," in *14th IEEE MASCOTS Conference*, 2006, pp. 189–196.

[4] B. Grigelionis, "On the convergence of sums of random step processes to a poisson process," *Theory of Probability and its Applications*, vol. 8, no. 2, pp. 177–182, 1963.

[5] O. Kallenberg, *Foundations of modern probability*, ser. Probability and its applications. Springer, 2002.

[6] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "Analyzing the video popularity characteristics of large-scale user generated content systems," *IEEE/ACM Trans. Netw.*, vol. 17, pp. 1357–1370, 2009.

[7] H. Yu, D. Zheng, B. Y. Zhao, and W. Zheng, "Understanding user behavior in large-scale video-on-demand systems," *SIGOPS Oper. Syst. Rev.*, vol. 40, pp. 333–344, April 2006.

[8] P. Gill, M. F. Arlitt, Z. Li, and A. Mahanti, "Characterizing user sessions on youtube," in *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking (MMCN)*, Santa Clara, USA, 2008.

[9] A. Abhari and M. Soraya, "Workload generation for youtube," *Multimedia Tools Appl.*, vol. 46, no. 1, pp. 91–118, 2010.

[10] L. Plissonneau, T. En-Najjary, and G. Urvoy-Keller, "Revisiting web traffic from a dsl provider perspective: the case of youtube," in *Proc. of the 19th ITC Specialist Seminar*, 2008.

[11] M. Zink, K. Suh, Y. Gu, and J. Kurose, "Characteristics of youtube network traffic at a campus network - measurements, models, and implications," *Computer Networks*, vol. 53, no. 4, pp. 501–514, 2009.

[12] Builtwith.com, "Dailymotion video website categories," http://trends.builtwith.com/media/DailyMotion-Video.