# Basic Setup

We consider one base-station serving a cell.

- In the beginning of each day, a list of $L_1$ number of potential popular videos is created at base-station. For each listed video $V_i$, a list of L user IDs is maintained. The User IDs are ranked according to who accessed the video most recently. Thus, the list corresponding to video $V_i$ contains L most recent users who downloaded $V_i$.

  ***Clarification needed here: Are we assuming that base-station also stores a copy of video?***

- Requests for the $i^{th}$ video is modeled as a independent Poisson process of rate $\lambda_i$, for $i = 1, 2...L_1$. Each user $j$ makes the demand for videos according to Poisson process of rate $\mu_j$.

- Each user can store upto M videos in the device's cache. This cache is modeled as FIFO queue. Thus, the cache in user k's device contains last M videos downloaded from it. No regard is given to file size of videos.

- In the beginning, cache of user $k$ is not full. A video can be in cache for upto at most $S_1 + S_2 + ... + S_M = SS_k$ amount of time. Here, each $S_m$ is distributed EXPONENTIAL($\mu_k$). Thus in the long run cache is always full. A video will stay in user cache for $SS_k$ amount of time.

- At the base station, the request for cached video $V_i$ is coming at Poisson rate $\lambda_i$. Thus an user will stay in the list corresponding to video $V_i$ for $X_1 + X_2 + ... + X_L = XX_i$ time units. Here, $X_i$s are distributed EXPONENTIAL($\lambda_i$). It is assumed that an user will not make the demand for same video more than once.

- When base-station receives a request for video $V_i$, it checks the list corresponding to $V_i$. Let $k_1$ last entry in the list (most recent user to download $V - i$) and $Y_1$ be the time elapsed since user $k_1$ downloaded the video $V_i$. $y_1$ is distributed EXPONENTIAL($\lambda_i$). The probability that video is still in the cache of user $k_1$ is P[ $Y_1 > SS_k$]. The probability that second most recent user has the video in the cache is P[ $Y_1 + Y_2 > SS_k$]. This way we can compute the probability that no user in the list corresponding to video $V_i$ has the video in cache as

$$P[RequestedVideoNotCached] = \prod_{j=1}^{j=L} P[Y_1 + Y_2 + ... + Y_j > SS_{k_j}] \tag{1}$$

In this case base-station has to download the video from Internet.

# Comments and Discussion

We can extend the model and analysis in several ways.

- If $SS_k$s are given we can optimize over L to minimize Equation 1. IT may be possible to optimize over M also.
- We may assume that requests at base station is Poisson even when the time between requests made by each user is not exponential. This can be justified by fact that superposition of a large number of point process is approximately Poisson. If assume some other distribution for inter-request times at each user, we can repeat the analysis for the new distribution. We have to calculate new distribution of $SS_k$

  ***Need some clarification here: Have I understood this correctly?***

- For modeling the position of user requests, we can start by assuming requests are generated according to non homogeneous Poisson point process. Request for $V_i$ at the base-station is generated by non homogeneous Poisson process of intensity $\lambda_i(s)$, where s represents geographical position within the cell. Thus, for User k corresponding to video $V_i$, the probability that request came region of radius r around $z_k$ is

$$\int_{B(z_k,r)} \lambda_i(s)/\lambda_i ds = \nu_k \tag{2}$$

  The domain of integration can be changed to account for shadowing and other effects.

- Alternate method to maintain the list of user IDs corresponding to a given video: Consider a video $V_i$ and an user k in the associated list. Let the rank of user k in this list be 'l' (User k is $l^th$ most recent user who accessed video $V_i$). Thus, the probability that user k still has requested video in the cache is

$$P[Y_1 + Y_2 + ... + Y_l \leq SS_k] = q_k \tag{3}$$

  Where $Y_i$s are assumed to exponential random variables.

  We can alternatively rank the users according to $\nu_k q_k$ and use this to maintain the list of users for a given video.

- ***Assuming caching at the base-station is also available:*** How do we choose which videos to cache and how do we manage this cache at base-station? We can rank the videos to cache according to it's demand rate $\lambda$, file-size $B$ and cost of downloading from Internet to base-station. Thus, we can rank videos according to

$$\lambda B + CostOfDownloadFromInternet \tag{4}$$

- ***More thought needed on this:*** Social impact of an user who downloaded can be used to predict the popularity of the video. This can in turn be used to tune request rate and optimize the caching mechanism.

- How do we update list of popular videos? In the list of $L-1$ videos, we can also keep track of number of requests made for a given video during time $[T-t, t]$. Where t is the current time and T is some fixed value. We can maintain another list of $L_1'$ videos along with the number of requests made in the window $[T-t, t]$. If number of requests for a video $V_i$ in the second list ($L_1'$) exceeds the number of requests for a video $V_j$ in the first list ($L_1$), we should replace $V_i$ with $V_j$.