A new caching policy for cloud assisted Peer-to-Peer video on-demand services

Franco Robledo, Pablo Rodríguez-Bocca, Pablo Romero and Claudia Rostagnol

Facultad de Ingeniería, Universidad de la República. Montevideo, Uruguay.

IEEE International Conference on Peer-to-Peer Computing
http://www.p2p12.org/
September 03-05, 2012 - Tarragona, Spain
We have a set of videos that we want to distribute on-demand, using GoalBit P2P protocol (http://goalbit.sourceforge.net), optimizing resources and download time.

In order to better understand the problem we have to define the following concepts:

- GoalBit platform, protocol and components.
- Peers behavior.
- Previous work.
- Assumptions.
What is GoalBit?

GoalBit is a set of products and protocols:

- **GoalBit Protocol (GBTP)** based on BitTorrent protocol, but designed for live streaming.

- **GoalBit Media Player (GMP)** player to watch live and on-demand GoalBit contents.

- **GoalBit Media Server (GMS)** which can:
  - play a GoalBit streaming,
  - distribute a GoalBit streaming,
  - generate a GoalBit streaming (broadcast), and
  - transcode a GoalBit streaming.

- **GoalBit Suite (GS)** that is a web system to administrate and control a GoalBit Platform.
GoalBit VoD components

These are the GoalBit components related with P2P Protocol and on-demand distribution:

**Content**: video to be distributed on demand.

**Peer**: end-user that want to watch a video *(downloader or seeder)*

**Super-Peer**: special peer with better bandwidth and more stable. They are managed by the platform’s operator and only executed in the cloud servers.
**GoalBit VoD components**

**Tracker**: server entity that knows all *Peers* and *Super-Peers* that are sharing a content (seeding or downloading).

**Repository**: storage space where to save VoD contents (inside the GoalBit platform) to be distributed by *Super-Peers*. 
Peers behavior

About peers behavior in a BitTorrent-like system:

- Peers can enter and leave the network when they want.
- A peer can be in one of the following states: connecting, downloading, seeding or disconnected. We can estimate the probabilities of changing from one state to other.
- It was previously shown \(^1\) that the BitTorrent peers behavior can be modeled as a Markov chain.

Problem - formal definition

Now we can present our problem in a more detailed way:

**Problem**

We have a set of videos, a set of repositories and super-peers (servers) and a set of peers requesting to watch these videos on demand. We want to find the best video distribution into the repositories to offer a minimal download time to peers (which could imply to add more replicas for more popular/required videos and less replicas for less popular videos).
# Outline

1. Introduction and Problem
2. Proposed Solution
3. Problem Optimization
4. Experimental Evaluation
5. Conclusions
As proposed in previous work, we model the system as a Markov chain with following details:

- Each peer can download multiple streams at time $t$.
- Peers are grouped into classes: $\{C^1, C^2, \ldots, C^K\}$ (peers in class $C^i$ are downloading $i$ videos simultaneously).
- Peers at class $C^i$ allocate the $i$-th part of its peer’s bandwidth for each concurrent download.
- Video popularity is defined by the number of peers requesting the same video in a period of time.
Problem data

- \( K \): available videos
- \( s_j \): size (in kbits) of video \( j \)
- \( x^i_j(t) \): *downloaders* in class \( C^i \) downloading video \( j \) at time \( t \)
- \( y^i_j(t) \): *seeders* in class \( C^i \) seeding video \( j \) at time \( t \)
- \( sp^i_j(t) \): *super-peers* in class \( C^i \) seeding video \( j \) at time \( t \)
- \( \lambda^i_j \): arrival rate for peers in class \( C^i \) requesting video \( j \)
- \( \gamma \): departure rate of *seeders*
- \( \theta^i_j \): departure rate of *downloaders* in class \( C^i \) downloading video \( j \)
- \( c \): total download bandwidth for each peer (in kbps)
- \( \mu \): total upload bandwidth for each peer (in kbps)
- \( \rho \): total upload bandwidth for each *super-peer* (in kbps)
- \( \eta \): video sharing effectiveness between peers (\( \eta \in [0, 1] \))
- \( S^p \): storage capacity of *super-peer* \( p \) (in kbits)
- \( E^p_j \): indicates if the *super-peer* \( p \) has a copy of video \( j \) (\( E^p_j \in 0, 1 \))
Problem model

Problem model (P2P)

Modeling the peers’ behavior as a simple fluid model we get the following differential equations (peers evolution: $x_i^j(t)$ and $y_i^j(t)$):

\[
\begin{align*}
\frac{dx_i^j}{dt} &= \lambda_j^i - \frac{\theta}{i} x_j^i - \min \left\{ \frac{c}{i s_j^i} x_j^i, \eta \frac{\mu}{i s_j^i} x_j^i + \alpha_j^i \sum_k \left( \frac{\mu y_j^k}{s_j^i k} + \rho \frac{sp_j^k}{s_j^i k} \right) \right\} \\
\frac{dy_i^j}{dt} &= \min \left\{ \frac{c}{i s_j^i} x_j^i, \eta \frac{\mu}{i s_j^i} x_j^i + \alpha_j^i \sum_k \left( \frac{\mu y_j^k}{s_j^i k} + \rho \frac{sp_j^k}{s_j^i k} \right) \right\} - \gamma_j^i y_j^i
\end{align*}
\]

$\forall i, j \in \{1 \ldots K\}$:
Assuming that the system will reach its *steady state*, where the number of peers is stable ($\frac{dx^i_j}{dt} = \frac{dy^i_j}{dt} = 0 \forall i, j \in \{1 \ldots K\}$), we can calculate the **steady state** value for $x^i_j(t)$ and $y^i_j(t)$:

$$
\overline{x}_{P2P}^{i} = \max \left\{ \frac{\lambda_{is}^j}{c}, \frac{i\lambda^j}{\gamma \mu \eta} \frac{\gamma s_j \sum_k \lambda^k_j - \mu \sum_k \frac{\lambda^k_j}{k} - \gamma \rho \sum_k \frac{z^k_j}{k}}{\sum_k \lambda^k_j} \right\}
$$

$$
\overline{y}_{P2P}^{i} = \frac{\lambda^i_j}{\gamma}
$$
A traditional Content Delivery Network (CDN) \(^2\) can be viewed as a particular case of this analytical approach (where \(\eta = 0\), \(\mu = 0\), \(y_j(t) = 0\) and the previously named super-peers are now static servers).

\[
\frac{dx_j^i}{dt} = \lambda_j^i - \theta_j^i x_j^i - \min\left\{ \frac{c}{is_j^i} x_j^i, + \alpha_j^i \sum_k \frac{s_j^k sp_j^k}{k} \right\}
\]

\(\forall i, j \in \{1 \ldots K\}\)

---

\(\text{CDN is a large distributed system of servers deployed in multiple data centers over Internet. Its goal is to serve content to end-users with high availability and high performance.}\)
Problem model (CDN)

To get the expressions for the **steady state** in the CDN system we take $\mu = 0$ and $\eta = 0$:

\[
\bar{x}_j^{i}_{\text{CDN}} = \max \left\{ \frac{\lambda_j^i s_j}{c}, \frac{i \lambda_j^i (\lambda_j - \bar{\rho}_j)}{\theta \lambda_j} \right\}
\]
Average download time

The **average download time** for any *downloader* at steady state can be computed applying *Little’s law*: 

\[ T^j = \frac{x^j}{\lambda^j} \]

We denote \( T^{P2P} \) and \( T^{CDN} \) the expected download times under steady state for peers in a P2P network and users in the CDN system, respectively.

\[
\begin{align*}
T^{P2P} &= \frac{x^j_{P2P}}{\lambda^j} \\
T^{CDN} &= \frac{x^j_{CDN}}{\lambda^j}
\end{align*}
\]
So, to compare average download times we can compare the number of downloaders in steady state for each system. We demonstrate that the following chain of inequalities holds:

\[
\overline{x^i_j}_{P2P} = \frac{i\lambda^i_j(\lambda_j - \rho_j) - i\lambda^i_j\phi_j}{\lambda_j \left( \theta + \frac{\mu}{s_j}(\eta - \theta) \right)} \leq \frac{i\lambda^i_j(\lambda_j - \rho_j)}{\theta \lambda_j} = \overline{x^i_j}_{CDN}
\]

So we can conclude a very important result:

**P2P vs. CDN**

\[
T^{P2P} \leq T^{CDN}
\]
Outline

1. Introduction and Problem
2. Proposed Solution
3. Problem Optimization
4. Experimental Evaluation
5. Conclusions
Combinatorial Optimization Problem

Our problem can now be written as a Combinatorial Optimization Problem (COP):

\[
\begin{align*}
\text{min} & \quad \sum_{p=1}^{K} \max_{j=1}^{n} \left\{ \frac{\lambda_j s_j}{c}, \frac{\lambda_j - \rho_j - \phi_j}{\theta + \frac{\eta \theta}{s_j} - \frac{\mu \theta}{\gamma s_j}} \right\} \\
\text{s.t.} & \quad \sum_j E_j^p s_j \leq S_p \forall p \\
& \quad \sum_p E_j^p \geq 2 \forall j \\
& \quad sp_j = \sum_p E_j^p \forall j \\
& \quad E_j^p \in \{0, 1\} \forall j, p
\end{align*}
\]
Multi-Knapsack Problem

The COP we address here is similar to the Multi-Knapsack Problem (MKP), where each super-peer holds a knapsack (super-peers’ storage) with limited capacity and items (videos) must be placed inside the knapsacks.

Since MKP is known to be Strongly NP-Hard (it reduces to the Bin Packing Problem) we designed a Greedy Randomized Adaptive Search Procedure (GRASP).  

---

GRASP is an iterative process which operates in two phases:

1. **Construction Phase**: an initial feasible solution is built,
2. **Local Search Phase**: solution’s neighborhood is explored to find better solutions.

As a result we will get a possible distribution of videos in the available super-peers, which gives the best (minimum) download time of all tested solutions.
GRASP - Construction Phase

\[ E = \text{RandomGreedy}(\lambda, \theta, \gamma.\eta, s, S, \mu, c, \rho) \]

1: \( V = \text{SortVideos}(s) \)
2: \textbf{for} \( j = 1 \) \textbf{TO} \( K \) \textbf{do}
3: \((p_1, p_2, p_3) \leftarrow \text{FeasibleSelect}(S_1, \ldots, S_P)\)
4: \( E(p_1, V(j)) \leftarrow 1 \)
5: \( E(p_2, V(j)) \leftarrow 1 \)
6: \textbf{if} \( p_3 > 0 \) \textbf{then}
7: \( E(p_3, V(j)) \leftarrow 1 \)
8: \textbf{end if}
9: \( \text{Update}(S_1, \ldots, S_P) \)
10: \textbf{end for}
11: \textbf{return} \( E \)
**GRASP - Local Search Phase**

\[ E_{out} = \text{LocalSearch}(E) \]

1. \((E, \text{improve}) \leftarrow \text{Add}(\text{Rand}(SP, Video))\)
2. IF \text{improve} GO TO Line 1
3. \((E, \text{improve}) \leftarrow \text{Delete}(\text{Rand}(SP, Video))\)
4. IF \text{improve} GO TO Line 1
5. \((E, \text{improve}) \leftarrow \text{Swap}(\text{Rand}(SP1, V1), \text{Rand}(SP2, V2))\)
6. IF \text{improve} GO TO Line 1
7. return \( E_{out} = E \)
Outline

1. Introduction and Problem
2. Proposed Solution
3. Problem Optimization
4. Experimental Evaluation
5. Conclusions
GRASP - Evaluation

Real information from a local Internet Video on-Demand Service Provider (AdinetTV - http://www.adinettv.com.uy/):

- 700 videos ($K$),
- videos’ average size is 23 MB ($s$),
- 4 super-peers ($P$),
- super-peers’ storage capacity is 1500 GB ($S$),
- super-peers’ upload rate is 10 Mbps ($\rho$),
- peers’ download rate is 1 Mbps ($c$),
- peers’ upload rate is 0.25 Mbps ($\mu$),
- P2P effectiveness of 50% ($\eta = 0.5$),
- seeders’ departure with rate 100 ($\gamma = 100$) and
- peers’ departure with rate 0.1 ($\theta = 0.1$).
We implemented the COP in **Matlab**, and we calibrate the GRASP algorithm with generated instances.

We contrast the performance of both CDN and P2P system in two aspects:

- **scalability**: we stress the system keeping the popularity proportional with the real data (i.e. multiplying the real $\lambda$ by an increasing factor).

- **economical savings**: we chose a fixed value for the system popularity and then changed the number of super-peers/servers (increasing the available resources at the platform side).
Results

**GRASP - Scalability result**

![Graph showing performance of CDN and P2P systems versus popularity.](image)

**Figure:** Performance of CDN and P2P systems versus popularity.
GRASP - Savings result

Figure: P2P performance versus number of super-peers.
Outline

1. Introduction and Problem
2. Proposed Solution
3. Problem Optimization
4. Experimental Evaluation
5. Conclusions
Results

- **theory**: We theoretically shown that, under same conditions, the average download time of the P2P system is equal or better than the average download time of CDN system.

- **scalability**: We confirmed the theory in the practice - In order to reach the same level of service (the same average download time) in a client-server system we should increase the number of servers (or super-peers) proportionally with the end-user requests; while in the P2P system we have a natural scalability with the growing resources offered by users (downloaders and seeders).

- **savings**: Results showed that CDN system is much more sensible to the changes in the available resources (super-peers/servers).
Future work

Current and future goals:

- Compare our GRASP results with other metaheuristics in order to find a good approximation of the optimal solution.
- Include the optimization algorithm into GoalBit Platform in order to know the algorithm performance and adaptation to the real world.
- Gather data from popular commercial services to further validate our model.
Thanks!

UdelaR: http://www.fing.edu.uy/
GoalBit: http://goalbit.sourceforge.net/
AdinetTV: http://www.adinettv.com.uy/