Report on research visit to University of California, Irvine (UCI)
under HEQEP CP 3137

Submitted by
Dr. Md Yusuf Sarwar Uddin
Deputy SPM CP 3137
Assistant Professor
Department of CSE, BUET, Dhaka-1000.

1. Introduction

I made a research visit to University of California, Irvine (UCI) United States from September 20, 2014 to January 31, 2015 to conduct a research related to project “Capacity building for post-graduate research on remote health monitoring in Bangladesh”. I worked with Prof Nalini Venkatasubramanium at Distributed Middleware Services (DMS) lab of Information and Computer Sciences department of UCI. The visit was quite great and a set of interesting problems was studied during the visit.

1.1 University of California Irvine

UC Irvine, a research school, is one of the ten campuses of University of California. Among ten UC system campuses this campus is relatively new (established around 1965) and is regarded one of the most prominent campuses among 50 years olds universities in the United States. The 1474 acre campus located in Orange county California hosts nearly 24 thousand students both undergrads and post-graduates. The campus has beautiful landscape and its various schools and departments organize themselves in a radial fashion around a park, named the Aldrich Park.

Figure 1: UC Irvine campus (Aldrich Park is at the center)

1.2 Information and Computer Sciences Department

The department of Information and Computer Sciences is located in a modern state-of-the-art 6-storied building named Donald Bren Hall (DBH) located in Henry Simueli School of Engineering campus. The building houses three departments, namely statistics, informatics and Information and Computer science. As of year 2014, they have a total of 68 faculties, 40 staffs, 1168 undergrads and 450

graduate students. UCI also has an Electrical Engineering and Computer Science department located in Engineering Hall.

2. Research conducted

The research conducted during the visit mainly lies in the direction of constructing building blocks for middleware services for media content delivery in crowdsourcing applications tailored for m-health. Recently media social networks emerged that enable end users upload and share rich media content from their mobile phones in the form of images, audio, and video. A few of them lately emerged in m-health context. Figure1 (www.figure1.com) is an example. Figure1 provides a mobile app that allows doctors and medical professionals to capture photos of particular medical cases of patients currently under their treatment and share those photos online to seek options, comments and suggestions from community members, especially from other doctors and experts. These images capture real cases with visual detail and supporting texts seeking need non-trivial attention from health professionals to comment on and to provide necessary suggestions.

Figure 2: Home page of “Figure1”

These kind of services in general involve interaction in terms of engagement among users and delivery of media content between back-end servers and clients, where the clients are usually hosted at resource constrained mobile devices. Although the service modality requires the end users to have constant connections to the Internet for receiving live feeds and delivering contents, the assumption may not hold for many users for obvious reasons. In this work, we propose a middleware service to be placed in between the server and the clients that would handle content delivery even though the mobile device may remain offline for unpredictable time. The underlying problems eventually boil to modeling of a content queue at the middleware (with its associated queue dynamics), and a set of solutions are suggested in the form control algorithms using Lyapunov optimization framework. In that report I will briefly highlight key elements of the investigation and the framework used.

2.1 Media Crowdsourcing

Media crowdsourcing enables individuals to generate rich media content and share them with others through a platform, called crowdsourcing platform. Media contents are high fidelity contents that have more expressive power than usual text. Media contents are abundant these days. A few names can be

- Instagram: shares photos
- Soundcloud: stores and shares songs, supports music streaming.
- Youtube, vimeo: contains videos.
- Facebook: enables sharing photos, photo albums, videos, etc.

Most of these applications are examples of publish-subscribe systems where contents are published (by ordinary users) and are then pushed to clients who subscribed to those contents. Clients are called subscribers. Our works extend middleware services for this kind of pub-sub systems.

One important aspect of media contents make them tricky to handle. That is they are usually large in size. So it engages significant resources to fetch, process, store and disseminate media contents. To circumvent that users are usually "notified" of the media contents via notifications (e.g., push notifications in phones) about their availability. Upon receiving notifications, users can then go and invoke retrieval if they want to do so. Notifications can be made rich so that they can contain some part of content if suitable.

Media contents are usually published in one exact format and size (e.g., an uploaded photo). But it can be argued that it is not always required to deliver the same exact content to *all* users for consumption. Contents might have different presentations. For example, photos can be presented in low resolution, in thumbnail format, text description or even as a caption. Photo albums can be replaced with smaller number of representative photos. These changes to content are referred to as "content adaptation". Although media content is good to have, processing them on mobile devices pose further challenges. Especially because mobile devices are resource constrained. They have limitations in terms of energy, storage, bandwidth, and network connections. It is interesting to note that based on user context and available resources, media content can be "adapted" to different presentations (may be degraded in quality). This adaptation may be directed in order to optimize resource consumption as well as to maximize user's satisfaction in consuming delivered contents

2.2 Problem Formulation

The following problem is formulated based on content adaptation. Deliver *relevant* media contents *optimally* to mobile users. The set of question we need to answer are –

- which contents?
- at what level of details?
- when, now? or can be delayed?

- utility-cost trade-offs…

More particularly, we ask questions and try to come with techniques that can be used to answer them satisfactorily. Which contents are relevant? These may be depended on matching preferences, interests, expertise, social ties, etc. Contents need to presented at what granularity/presentation level? Different presentations have different *utilities*. For example, original content (no degradations) has the largest utility and utility declines as content gets degraded. The next thing to consider is the availability of resources (e.g., energy/bandwidth budget for fetching contents). The problem is then to ask---given a set of contents with their utilities, find a subset of contents that maximize total utility achieved from their delivery subject to various resource constraints.

2.3 Middleware for Content Adaptation and Delivery

We build a middleware to adapt contents as suggested and deliver them to end users. Middleware is a sort of a staging area for contents before they are delivered to users. In terms of operation, it selects most relevant contents per user and then selects suitable presentations per contents that are selected.

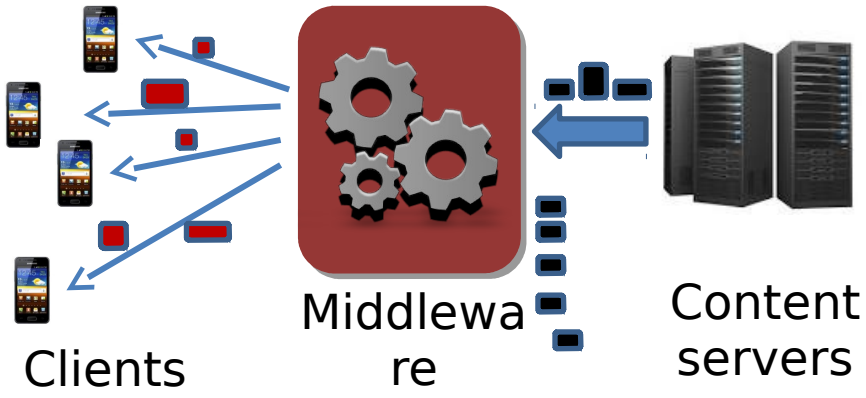

Figure 3: Middleware architecture

2.3.1 Model of the Middleware

We model the middleware as a collection of content queues. Contents arrive in the queues as they are published and are fetched to deliver to the subscribers. And contents depart from the queues as they are delivered to the users (mobile devices) or dropped. The goal of the operation is to maintain a “stable” queue while maximizing the sum of utility of delivered contents. By stability we mean queue should not grow unbounded (i.e., the expected queue length should remain finite).

As per classical queuing theory, a (M/M/1) queue remains stable if service rate (μ) is strictly higher than arrival rate (λ), that is $\mu > \lambda$. The problem is we don’t really know these two rates. That is, the parameters for arrival process and departure process are unknown. Instead, the system can be modeled as running in a series of intervals. And for each interval, *t*, we have—

- A(t): new contents arrive during interval t
- μ(t): contents scheduled to depart during interval t

Using these two parameters for each slot, we can formulate a queue stabilizing control algorithm based on Lyapunov framework to optimize our required objective of maximizing total utility.

2.3.2 Lyapunov Framework

Lyapunov framework (LF) is a "control strategy" that helps achieving queue stability amid of unknown queue dynamics (arrival and departure). Lyapunov optimization framework enables to design control algorithm that optimizes certain objective function subject to keeping average queue length finite. It guarantees to achieve *near-optimal* performance while keeping queue stable.

The core concept in Lyapunov stability is called *Lyapunov drift.* It refers to measuring a running quantity as the queue grows. Let $Q(t)$ be the queue backlog at time t. We define a Lyapunov function as $L(t) = Q^2(t)/2$ (division by 2 is made for mathematical simplicity that follows). Lyapunov drift, denoted as Δ(t), measures the difference between the values of Lyapunov functions in two successive time intervals. That is, $\Delta(t) = L(t+1) – L(t)$. Control strategy is to keep this drift as small as possible.

A queue is said to be "stable" it the long-run average of the queue backlog remains finite. That is—

$$\acute{Q} = \lim_{t \to \infty} ¿ \frac{1}{t} \sum_{\tau=0}^{t-1} Q(\tau) < \infty$$

The following theorem is the main results for queue stability.

Theorem (Lyapunov Stability). *If there exists constants B,* ϵ *, such that for all timeslots we have—*

$$E\left[\Delta(t) \middle| Q(t) ¿ \right] \leq B - \epsilon Q(t)$$

then, the queue is stable, that is $\acute{Q} = \lim_{t \to \infty} ¿ \frac{1}{t} \sum_{\tau=0}^{t-1} Q(\tau) < \infty$ *, and furthermore,* $\acute{Q} \leq \frac{B}{\varepsilon}$.

The proof is surprisingly simple, which makes use of iterated expectation and telescopic sum.

The control strategy then tries to minimize $E\left[\Delta(t) \middle| Q(t) ¿\right.$ subject to required constraints. Since the values of $A(t)$'s are beyond the controller's hand, the strategy chooses appropriate μ(t)'s to achieve stability. It would be information if we show how values of B and ϵ are deducted. The framework assumes that the second moment of both arrival and departure process remain bounded. Let λ be

the upper envelope of $A(t)$ and μ be the lower envelop of $\mu(t)$, that is, for all t, A(t) ≤ λ and $\mu \geq \mu$(t). It can be shown that B = $\lambda^2 + \mu^2$. Again, for arrival process λ, if there exists an optimal control strategy with departure rate μ' that stabilizes the queue, then ϵ = μ' - λ.

Lyapunov framework also supports optimization of certain objective function while keeping queue stable. Let this objective be some "penalty" that LF minimizes. One example of this penalty can be energy consumption that the system tries to minimize over time. In that, drift is defined as Δ(t) + V P(t), for a control parameter V. So, control strategy now tries to minimize

$$E[\Delta(t)+VP(t)\vee Q(t)]$$

The parameter V makes a trade-off between P and Q. The following theorem establishes the results.

Theorem (Lyapunov stability with penalty). *If there exists B and* ϵ *such that for all slots, we have—*

$$E\left[\Delta(t)+VP(t)\middle|Q(t)¿\right]\leq B-\epsilon Q(t)+V\,P^{¿}$$

then the queue remain stable and the followings hold.

$$\acute{P}\leq P^{¿}+\frac{B}{V}$$

$$\acute{Q}\leq\frac{B+V\,P^{¿}}{\epsilon}$$

It can be observed that average *P* can be made arbitrarily close to optimal *P** by enlarging *V* but that lets the queue grow. While *P* decreases at the rate of O(1/V), queue grows at a rate of O(V). This is called [O(1/V), O(V)] trade-off in Lyapunov framework. As we can see, *V* is an important "knob" to tune performance of the scheme. In effect, *V* controls the size of the queue (storage overhead at the middleware). *V* can be a constant or can be adjusted according to certain conditions (e.g., network connectivity).

3. Conclusion

In this work a middleware service for content adaptation for media content delivery to mobile devices is developed. An optimization formulation is made using Lyapunov framework and solved with heuristics. Experiments are currently underway. We consider only subscription side of the service that handles

reception of contents. Similar efforts can be given on publisher side for uploading contents.

4. Acknowledgement

This work is a joint effort with other researchers, namely Vinay Setty from Max Planck Institute, Germany, Ye Zhao from Google, Mountain View, CA, and Prof Roman Vitenberg of University of Oslo, Norway. The author conveys heartiest gratitude to them. He also expresses his sincere thanks to all members of DMS lab, namely Kyle, Charles, Gene, Andy, Quixi, and Ranga, for their supports during his stay at UC Irvine. And lastly not the least, special thanks to Prof Venkatasubramanium for being a generous host of the visit.