ColStream: Collaborative Streaming of On-Demand Videos for Mobile Devices

Mingyang Zhong*, Peizhao Hu*[†], Jadwiga Indulska*[†], Mohan J Kumar[‡]
*The University of Queensland, Australia
[†]National ICT Australia (NICTA)
[‡]Rochester Institute of Technology, USA
Email: mingyang.zhong@uq.net.au

***Abstract*—The number of mobile users of on demand video is growing rapidly. However, bandwidth fluctuation in the 3G/LTE technologies is an obstacle in providing high quality smooth video playout for users on the go. In this paper, we present ColStream that can aggregate bandwidth from ubiquitous devices to ensure high quality video streaming with minimal stalling time. ColStream dynamically adjusts the set of collaborators and the size of video chunks that the collaborators need to pre-fetch ahead of the video chunks playout time to provide smooth video playout. ColStream uses a multi-objective optimisation method to maximise bandwidth and minimise cost. ColStream requires neither external servers nor proxies to provide its functionality. The paper describes the ColStream functionality, architecture, applied algorithms, ColStream prototype, and evaluation of its suitability for effective video streaming in mobile environments.**

I. Introduction

With the proliferation of smartphone and tablet devices, there is a significant growth in global mobile data traffic (reached 885 petabytes per month at the end of 2012) [1]. For the first time, mobile video traffic exceeds 50 percent of the data traffic. These statistics highlight the fact that more and more users require on-demand video content on the go.

The current 3G/LTE technologies make this possible and deliver excellent user experience when their mobile devices have good signal reception. Due to the problems of dead/grey zone in coverage, users experience significant differences in mobile signal strength; this results in fluctuation of bandwidth - especially when users are on the go. For example, when travelling on a bus, the signal quality can vary from full to barely connected on the cell tower edge. Due to the signal fluctuation it is challenging to provide enough bandwidth to maintain smooth video streaming (without video stalling).

There already exist solutions to adjust the video quality when network conditions change. For example, DASH [2] is the standard that requires the server to provide a variety of video encodings and therefore allows selection of various quality pieces to match available bandwidth. This approach sacrifices video resolution for continuous video streaming.

Our research takes a different approach, the goal is to improve user experience both in smoothing the video playout and in maintaining the video quality. This paper explores the idea of collaborative streaming by facilitating the available idle wireless devices in the vicinity. We present an overall design, implementation and performance evaluation of our solution — *ColStream* (Collaborative Streaming). When a user requests a video, ColStream dynamically computes the required bandwidth for a smooth playout at the desired video quality. When additional bandwidth is required, ColStream (on the initiator device) recruits idle wireless devices (the collaborators) nearby and delegates to them the task of downloading some of the video pieces. Upon completing each of the partial video downloading tasks, the collaborators send the downloaded video content to the initiator and are rewarded for their help (for the usage of the cellular network and battery power). The collaborators communicate with the video providers (e.g., YouTube) using 3G/LTE networks and use wireless network (e.g., WiFi direct) to transfer the video content to the initiator.

The existing early approaches to collaborative video streaming have considerable drawbacks as we describe in Section II. In our approach, we have made the following contributions towards achieving a solution that minimises stalling of high quality video streaming to a mobile video user (initiator):

- group formation protocol,
- peer selection algorithm,
- work distribution algorithm.

The group formation protocol allows the initiator to advertise for help and allows collaborators to join the collaborative group. The selection algorithm selects the optimal set of collaborators that maximises bandwidth and minimises cost. The distribution algorithm dynamically computes and assigns optimal amount of video downloading to the collaborators, based on their long-term averaged bandwidth. The distribution algorithm minimises over commitment when bandwidth at collaborators change dynamically. The proposed approach is also supported by an incentive mechanism based on share market principles [3] that allows bandwidth trading between the initiator and collaborators. Due to page limitations, the incentive mechanism is not presented in this paper.

In order to demonstrate the effectiveness and feasibility of ColStream, we have designed a unified system architecture that incorporates the above functionalities and developed a working prototype for the Android emulators and phones. Finaly, extensive performance studies have been carried out to complete a systematic evaluation of ColStream.

The remainder of this paper is organised as follows. Section II presents an overview of the related work. Section III discusses the overall design of ColStream; this is followed by the detailed ColStream descriptions in Section IV. In Section V, we present a systematic performance evaluation of ColStream and compare it with the existing solutions on the Android platforms. Section VI concludes the paper.

II. Related work

There exist many *stand-alone* bandwidth aggregation solutions that utilise the device's own capabilities to increase throughput. Ramaboli et. al. [4] surveyed and categorised them according to the protocol layer at which the aggregation happens. For example, Chebrolu and Rao [5] proposed a solution to aggregate the throughput offered by the multiple network interfaces on a device; and Barre et. al. [6] presented a system that increases the device's throughput by using multipath TCP. The focus of these solutions is on situations where alternative Internet connection is available or sufficient 3G/LTE bandwidth is available on a single device.

In this paper, we focus on users on the go that need access to high-definition video on a 3G/LTE enabled device (smartphone). In this situation, other means of Internet connectivity may not be possible, also the signal quality varies as the users travel through different environments (e.g., tunnel). To address this problem of lack of bandwidth, another kind of approaches focus on collaboration with nearby wireless devices. The data or video are divided into smaller portions and their downloading assigned to peers in a collaborative social group. Examples of collaborative downloads are BitTorrent-like systems [7], [8], [9], [10]. One of them, MicroCast [10], applies this concept to smartphones (assigns fixed size segments to phones with the lowest downloading so far). However, they serve a network of peers that have common interest in the content they download and share. In addition, each peer stores the downloaded data and serves it to others as a contribution to the collaboration (the *tit-for-tat* incentive mechanism is used). Privacy and copyright issues make this approach difficult to apply in our scenarios.

There exist solutions for colaborative file download or video streaming that only utilise other devices' bandwidth without caching data on the collaborating devices. COMBINE [11] and CStream [12] are two representative solutions in this category. COMBINE is collaborative data downloading which provides incentives for collaborators and CStream is collaborative video streaming but with no incentives. They share similarities in the basic concept with ColStream: utilising the available bandwidth of nearby devices by delegating parts of the downloading tasks to them. However, we made significant contributions in ColStream, including *group formation*, *peer selection* and *work distribution* as discussed in the following sections. Due to these contributions, ColStream shows a significant performance improvement over the two solutions.

III. ColStream: concept and architecture

The core of the proposed ColStream is the ability to bond several "small" 3G/LTE Internet connections offered by nearby wireless devices (smartphones, tablets) into one virtually "large" Internet connection. In [10], the authors showed that the bandwidth of one device depends on ISP's admission control policy. A user cannot use all the bandwidth even if there is only one user in the cell. By combining multiple devices' internet connection, ColStream can effectively increase the bandwidth for streaming video, and therefore significantly improve the user experience by maintaining the desired level of video resolution and minimising video stalling time. In addition, the wireless device can better tolerate various causes of signal degradation (e.g., signal fading, interference).

With ColStream installed, a mobile device can either be an *initiator* or a *collaborator* depending on the role they play.

- *As an initiator*: (i) independently stream on-demand videos when sufficient bandwidth is available or no collaborator is around; (ii) advertise intention to purchase additional bandwidth from nearby mobile users when needed; and (iii) optimally utilise the additional bandwidth when a collaborative group is formed.
- *As a collaborator*: (i) participate in the collaborative group when someone offers a purchase price greater than collaborator's selling price; and (ii) trade the residual bandwidth when not actively using the device.

These operations are executed automatically by ColStream when conditions are met. Users can override the settings from a preference configuration interface.

Figure 1 presents the ColStream system architecture on the initiator and collaborator devices. When a user requests a video through ColStream, the device starts streaming the video, and in parallel it computes the required bandwidth for a smooth playout at the desired resolution. The *bandwidth estimator* on each device will calculate the long-term estimation of device's bandwidth. If the user device has enough bandwidth, it continues with the streaming, but periodically checks the available bandwidth to assert whether it needs to adapt to changes in bandwidth availability. Otherwise, the user device broadcasts its intention to purchase additional bandwidth with a purchase price per data unit. Upon receiving the message, idle mobile devices[1] nearby can respond with their interest to participate. The interested mobile devices will be registered into the *candidate pool*. Then an optimisation algorithm in the initiator periodically computes the best combination of devices for the streaming task (using a utility function to maximise bandwidth and minimise cost). Once collaborators (forming the *collaborator pool*) are identified, the *work distributor* dynamically computes an optimal downloading task for each collaborator according to their estimated bandwidth. A *tracker* will monitor the progress and re-assign the incomplete tasks to other devices if necessary. Once the assigned tasks are finished, the downloaded chunks of video are forwarded to the initiator via local wireless connection, and then reassembled for playing. In parallel the initiator works on a new allocation of downloading tasks for the next section of the video.

When compared with the existing solutions ColStream offers a number of advantages. ColStream is the first solution

[1]Idle means the network activity is below a user defined threshold not necessarily zero network activity.

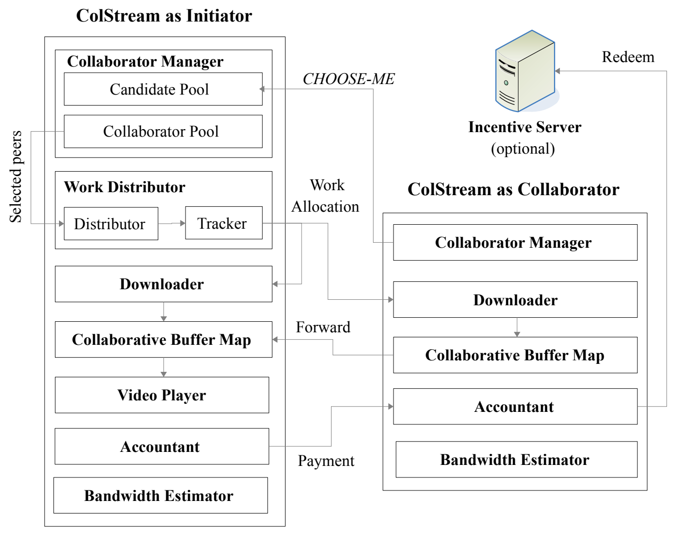


Fig. 1. System architecture and interactions

for collaborative streaming without the need of external servers or proxies supporting the streaming. The Incentive Server that maintains the system incentives and accounting is external but this server can work in both synchronous or asynchronous modes, hence mobile devices do not need to have a connection with the Incentive Server during the collaborative streaming. ColStream can dynamically adjust sets of collaborators and the size of downloaded video chunks.

IV. ColStream: Design

The section focuses on discussing the *nuts* and *bolts* that make the collaborative streaming possible.

A. Collaborative Group Formation

Forming a collaborative group from nearby devices is a task of the initiator. There are many group formation protocols; for example, CStream and COMBINE both wait for a timeout when discovering collaborators. ColStream does not wait for collaborators (which might be none) to join, it starts streaming video when it is requested. In parallel, the device periodically computes the required bandwidth for the remaining portions of the video to check whether additional bandwidth is needed to achieve good quality video streaming.

If additional bandwidth is required, the initiator periodically broadcasts the *JOIN-ME* message via its WiFi channel. The message contains contact information, purchase price, certified public key, etc. Nearby idle wireless devices that are interested in participation will reply with the *CHOOSE-ME* message, which includes the device profile, estimated throughput of the 3G/LTE connection and the selling unit price. The initiator registers the responding devices into the candidate pool. This process repeats throughout the whole streaming task. From these candidates, the collaborator selection algorithm computes the best combination of the collaborators and forms the collaborator pool - the selection is done periodically and on changes in the candidate or collaborator pools. Once a collaborator is identified, it will send periodic *I-AM-ALIVE* messages to the initiator.

B. Collaborator Selection

To determine the optimal set of collaborators that provides the maximum bandwidth with the minimum cost for completing the streaming task, we consider two important metrics:

1) Bandwidth Estimation: To estimate achievable throughput of collaborating devices, we developed an approach to estimate the achievable throughput based on the mapping of 3G signal and achieved throughput. The phone will periodically report the measured signal and achieved throughput. This information is used to update the mapping. Before forming a collaborative group and streaming, Colstream will passively monitor the achieved throughput (tp) and the corresponding signal strength ($signal$) based on user's general Internet usage. To overcome the problem of signal and throughput fluctuation, we use weighted moving average to smooth out the results.

$$signal = \alpha * signal_{new} + (1 - \alpha) * signal_{old} \quad (1)$$

$$tp = \alpha * tp_{new} + (1 - \alpha) * tp_{old} \quad (2)$$

where $signal_{new}$ and tp_{new} is the new measurement, $signal_{old}$ and tp_{old} is the historical averaged value. α is a weight given to the two measurements (defaulted to 75%, optimal value will be tested in the future);

When a collaborator replies to the initiator's request, the collaborator checks the current signal strength and looks up a rough estimation of the expected throughput that it has achieved with this signal strength in the past. To demonstrate this approach, we carry out multiple experiments (while travelling on a bus or stationary) to record the per second 3G signal strength, measured throughput and estimated throughput on an Android phone. Figure 2 shows the correlation between signal and measured throughput in one of our experiments. We also plot the estimated throughput. In general, we observe that the estimated throughput is able to track the measured throughput.

2) Pricing Scheme: Each collaborator specifies a unit price for the bandwidth to be traded with the initiator. The unit price takes into account not only the cost of accessing the Internet and battery consumption, but also the user's special circumstances. As estimating a rough unit price is very difficult, we are mimicking the share market. The system can recommend a unit price that reflects the value in the market. Potential collaborators can use the recommendation or override it to reflect their own circumstances.

When configuring ColStream, a user (initiator) needs to specify a budget per video. From this budget, the system computes the purchase price per data unit. The goal of the collaborator selection algorithm is to determine the optimal set of collaborators that maximises the throughput while meeting the budget restrictions. This problem can be formulated as a multi-objective optimisation problem [13]. We assume n wireless devices, $D = \{d_1, d_2, d_3, ..., d_n\}$, $d_i \epsilon D$; and s is a subset in the power set of D, i.e., $s \epsilon \mathcal{P}(D)$. We can define the

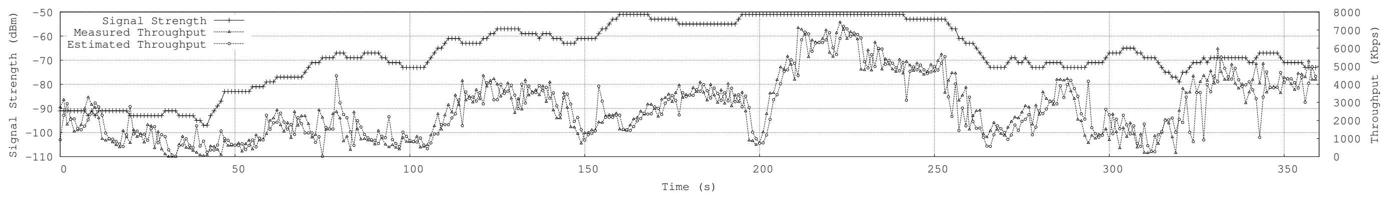


Fig. 2. Estimating throughput based on the mapping of signal and measured throughput (data generated on a bus trip)

following objective functions $F_1(s)$ and $F_2(s)$:

$$\begin{aligned} &Maximise\ F_1(s) = \sum_{d_i \epsilon s} tp(d_i) \\ &Minimise\ F_2(s) = \sum_{d_i \epsilon s} cost(d_i) \\ &s.t.: \\ &\ F_2(s) \leq Budget_{remain} \end{aligned}$$

where $tp(d_i)$ is the estimated maximum achievable throughput of collaborator d_i, as computed in Eq. (2); similarly, $cost(d_i)$ is the total cost of using device d_i for downloading m units of data; therefore, $m * price_{d_i}$. The initiator always participates in the streaming task. The total cost of using the collaborators' bandwidth $F_2(s)$ has to be less than the budget $Budget_{remain}$ for the remaining video portions. With the given budget, the system will find the combination of collaborators with the maximum aggregated throughput.

From the above objective functions, we then define a utility function, which simplifies the problem to a single-objective optimisation problem. With the utility function, the optimal set of collaborators for the given video portion is the solution with the highest value of $h(s)$.

$$Maximise\ h(s) = \frac{F_1(s)}{F_2(s)} \tag{3}$$

In a mobile environment, collaborators may join or leave the group and the system needs to adapt to these changes. The change in the ColStream's collaborative group and also timeout (default 5 s) will trigger the selection algorithm. This is to maintain an optimal combination of the collaborative group with the given constraints and available devices for the remaining video portion.

There is another constraint that can be opted in or out depending on specific user preferences. That is, the maximum throughput of the selected collaborative group F_1 has to be greater than the required throughput $tp_{required}$ as determined by the video player (including the communication overhead for forwarding data from collaborator to initiator via local wireless channel). When this constraint is applied, the system will form a collaborative group if and only if $F_1(s) \geq tp_{required}$. Consider the bit rate of the video player is calculated as

$$bitrate = \frac{video\_size}{video\_playtime} \tag{4}$$

then, the time of downloading a chunk with size k via cellular network and forwarding it back to initiator via local wireless network should be equal or less than the time video player needs for playing it with its bit rate.

$$\frac{k}{bitrate} \geq \frac{k}{tp_{required}} + \frac{k}{tp_{forwarding}} \tag{5}$$

where $tp_{forwarding}$ takes into account the overhead for sending data back to initiator and is computed using the max chunk size and the single trip time of exchanged packets (i.e., *JOIN-ME* and *CHOOSE-ME* messages).

From equation 5, we get

$$tp_{required} \geq \frac{bitrate * tp_{forwarding}}{tp_{forwarding} - bitrate} \tag{6}$$

The reason to provide this optional constraint in ColStream is that some users will use the collaboration only if it guarantees smooth video streaming at the desired resolution; others may want to improve the video streaming experience.

C. Dynamic Work Distribution Algorithm

The entire video needs to be divided into smaller chunks and chunk downloading assigned to the collaborators. The goal is to download the necessary chunks and forward to the initiator before their playout time. We developed a dynamic work distribution algorithm (that is inspired by [11], [12]) with two significant contributions for streaming on-demand videos:

1) Dynamic chunk size: The video frames need to arrive before their playout time. To meet this requirement, the algorithm dynamically computes the optimal chunk size every time a collaborator is selected to help, rather than assigning fixed size chunks (commonly used 256 KB) to all collaborators throughout the entire video download and regardless of their throughput. Our algorithm computes the optimal chunk size according to each collaborator's estimated throughput. The collaborators are given the appropriate downloading task, without over committing that would cause additional delay for the initiator when resolving the problem of collaborators failing to complete their downloading task. Assigning downloading tasks according to the collaborators' throughput should improve the overall performance of the system. The chunk size of node i, $Chunk\_size_i$ is calculated as

$$Chunk\_Size_i = \frac{MAX\_CHUNK\_SIZE * tp_i}{tp_{max}} \tag{7}$$

where MAX_CHUNK_SIZE is the maximum chunk size defined in the system (by default we set it to 256 KB, which is common in the related work [11], [14]); whereas tp_i is the estimated throughput of node i, and tp_{max} is the maximum estimated throughput among all the selected collaborators. The algorithm reacts to the changes in each node's throughput and allocates different chunk sizes to optimise the video downloading.

2) Partial Transaction: Another contribution in our algorithm is the ability to detect and handle partially completed tasks as a result of (i) collaborators experiencing insufficient bandwidth, (ii) collaborators suddenly leaving the collaborative group (due to mobility or battery depletion), or (iii) collaborators detecting network activities on their devices generated by other applications. ColStream always gives a bandwidth usage preference to the primary user of the collaborating device - whenever a network activity is detected on the device ColStream will quickly wrap-up any downloading task, send partially completed video chunk to the initiator and claim the credit only for what it has downloaded.

ColStream is equipped with both *passive* and *active* methods to detect the occurrence of these problems. In the *passive* method the collaborator reports the up-coming changes in the collaboration (for example, when a collaborator's battery level drops below a threshold). In this case, the initiator will schedule the remaining parts of the chunk as a task for itself on the next available slot. In the *active* method the initiator monitors collaborators. When abnormal collaborator's behaviour is detected (e.g., missing three consecutive I-AM-ALIVE messages), the initiator takes actions to minimise the impact on the streaming task. For example, if a collaborator does not forward the video chunk within an estimated completion time, the initiator will ping the collaborator. If no response is received within a timeout, the initiator re-allocates the chunk.

In most of the existing solutions, when a collaborator fails to complete chunk downloading, the systems will discard the downloaded data and re-assign the whole chunk to another collaborator. This is not suitable for streaming on-demand videos where timeliness is an important constraint. In addition, this can create a significant fairness problem as this partially downloaded content is not rewarded. In COMBINE [11], the authors discuss handling partial download, but no details are given for this feature. Also, COMBINE only supports the active method of detecting partial download.

V. Evaluation

In this section, we present the system performance evaluation in two environments: emulated Android phones and real Android devices.

A. Evaluation setup and basic tests

To perform a systematic evaluation of ColStream, we used a platform with Java and Android emulators as it allows repeatable experiments to analyse the ColStream performance in details. We also validated the feasibility of ColStream with a set of experiments on actual Android devices.

In our experiments we emulated one initiator and 10 collaborators on a laptop with Intel Core i5-3210M processor and 6 GB of RAM. To emulate the throughput fluctuation in the real world, each node defines its target throughput and unit price for sharing bandwidth. We developed a method to fluctuate the downloading throughput of each collaborator around the target throughput.

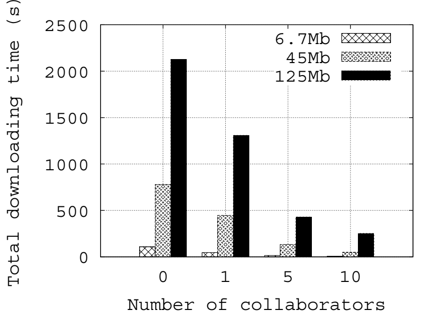


Fig. 3. Performance when different numbers of collaborator are used

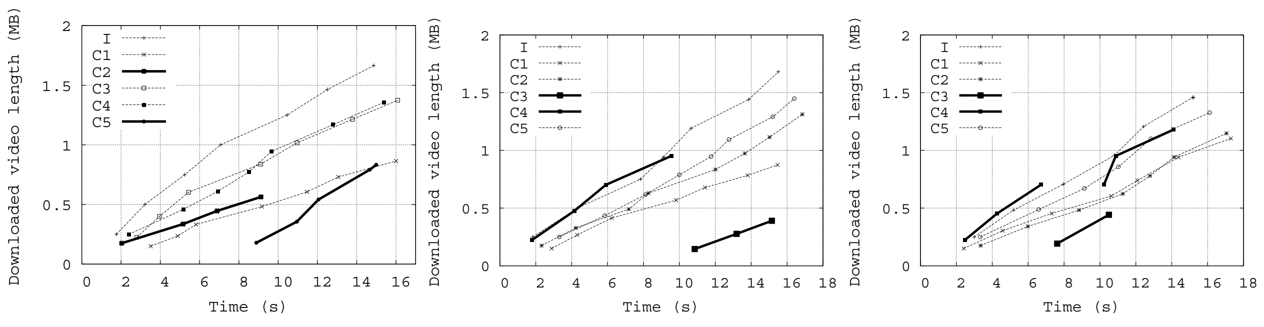
First, we performed a basic experiment that demonstrates that ColStream can significantly increase the throughput for an initiator. In this experiment, we streamed randomly selected videos from Youtube with three different video sizes (6.7 MB[2], 45 MB[3] and 125 MB[4]) using different numbers of collaborators (0, 1, 5 and 10). Figure 3 shows a comparison of the downloading times. This figure confirms the expected behaviour - the more collaborators the faster the video is downloaded under the same network conditions. It should be noted that the actual download speed depends on the Internet connection on the emulation machine. However, we emulated a smaller throughput for each collaborator to mimic a variety of bandwidth available in collaborators. The figure shows at least four times improvement in downloading time when five collaborators are used.

B. Dynamic adaptation

One important aspect of ColStream is its ability to adapt to the dynamic changes in the collaborative group and to select the optimal set of collaborators with the remaining budget for the remaining downloading task. We demonstrate the ColStream adaptation abilities in three scenarios in which we use one initiator and five collaborators. The events of interest are outlined in Figure 4 that shows the total length of the downloaded video content over time.

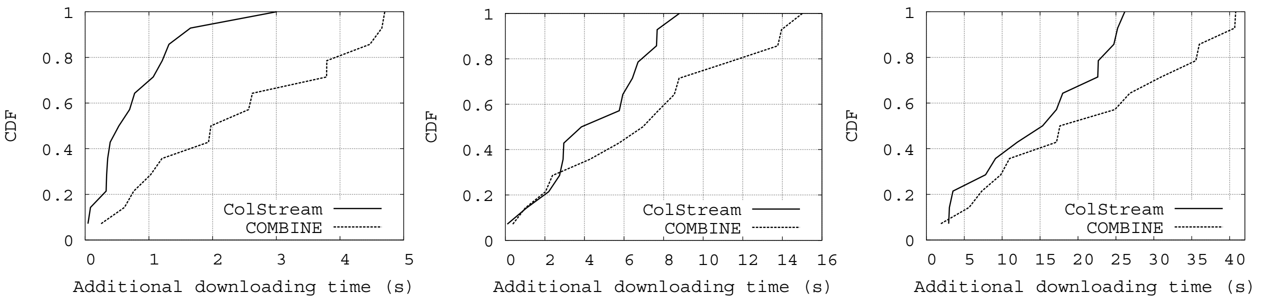
1) Scenario 1 - presence of a better deal: In this scenario, we want to demonstrate ColStream's ability to make use of a new collaborator with a *better* deal (e.g. a higher throughput for a lower price). As shown in Figure 4(a) - collaborator 5 joins the group at 8 seconds. After it joins, it is selected to replace collaborator 2. Collaborator 2 may still be downloading its allocated chunk as shown in the figure. From Figure 4 (b) we see that this event introduced about 1 second delay in downloading time compared with the downloading time of the five original collaborators. However for longer videos the

[2]http://www.youtube.com/watch?v=ZSt9tm3RoUU
[3]http://www.youtube.com/watch?v=GXQSkE9J4N8
[4]http://www.youtube.com/watch?v=Y4MnpzG5Sqc



(a) Scenario 1 (b) Scenario 2 (c) Scenario 3

Fig. 4. Scenarios of dynamic adaptation tests



(a) 20% failure probability (b) 60% failure probability (c) 100% failure probability

Fig. 5. Performance comparison between ColStream and COMBINE

delay caused by the group reassignment will be offset by better throughput (and also price) of the new collaborator.

2) Scenario 2 - a collaborator leaves: If a collaborator leaves the group, ColStream needs to search for an alternative group arrangement for the remaining video streaming task. As shown in Figure 4(b), we purposely turn off collaborator 4 at 9 seconds. We can see from the figure that collaborator 4 forwards what it has already downloaded to the initiator (as supported by the partial transaction feature) and then collaborator 3 is selected as a replacement after one second.

3) Scenario 3, a collaborator offline for some time: This scenario demonstrates ColStream's ability to cope with network instability that may force a collaborator offline for a period of time. In this case ColStream should select a replacement (as in scenario 2) and select the original collaborator when it joins again (as it was selected before for its good offer). Figure 4(c) shows that collaborator 4 leaves the group for about four seconds and then it rejoins the group again. Collaborator 3 is selected as a replacement for this duration.

C. Performance tests: COMBINE against ColStream

Among the existing solutions, COMBINE [11] is the most closely related to ColStream. Therefore we present both the qualitative and quantitative comparison of the two solutions.

There are some similarities between COMBINE and ColStream: both enable bandwidth sharing among nearby mobile devices and utilise a work queue to efficiently allocate downloading tasks to collaborators. However, they serve different purposes. The bandwidth sharing in COMBINE aims to improve data file downloading, whereas ColStream aggregates collaborators' bandwidth for efficient streaming of on-demand video. Due to the timeliness constraints of video frames, the ColStream design is more challenging. Because of this constraint we designed an efficient handling of overcommitted collaborators or partially completed chunks. In addition, we developed an algorithm to dynamically compute chunk sizes based on estimated throughput of collaborators rather than a fixed size. The active and passive methods to handle partial transactions is another means to improve the video prefetching capabilities, resulting in better user experience.

As for the quantitative evaluation, we implemented some of COMBINE's features (including chunk allocation, timer based collaborator check), as described in [11] to carry out the performance comparison. We conducted experiments in which collaborators experience different failure probabilities on completing the given downloading task (either cannot download or only download a part of the chunk). In these experiments an initiator employs five collaborators to download a YouTube video of 6.7 MB. Figures 5 show the performance comparison (in terms of additional downloading time using Cumulative Distribution Function) between ColStream and COMBINE against their corresponding benchmarking setup (the no failure cases), with different levels of failure probability (20%, 60%, and 100%[5]). The Figures show that ColStream performs much better than COMBINE across all failure probabilities, with significantly smaller additional downloading time. For example,

[5]In this scenario, the video is downloaded by the partial download.

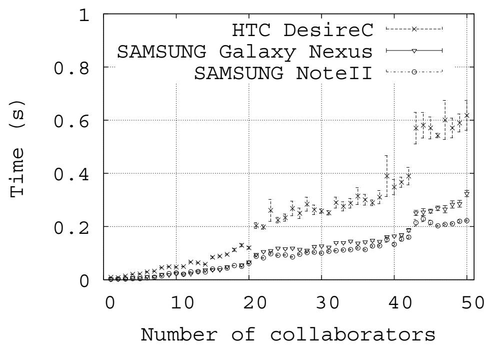


Fig. 6. Time for finding an optimal solution

in the case where a collaborator has 60% failure probability (as shown in Figure 5(b)), the maximum additional downloading time for ColStream is about 8.2 s, whereas COMBINE will have about 30% chances the additional downloading time is beyond 8.2 s (but less than 15 s). In other words, these results demonstrate the contributions of allocating dynamic chunk size and handling partial transactions are effective.

D. Feasibility study of the optimisation algorithm

The time it takes to find an optimal set of collaborators determines whether ColStream is feasible for real world applications. Therefore, we investigated how the optimisation algorithm performs on real Android phones (HTC Desire C with 600MHZ processor and 512MB memory, Samsung Galaxy Nexus with 1.2GHZ dual-core processor and 1GB memory, and Samsung NOTE II with 1.6GHZ quad-core processor and 2GB memory). We run multiple tests using various numbers of collaborators, ranging from 1 to 50 (to simulate scenarios from travelling on a bus to walking in a busy shopping mall). Figure 6 shows the time required (as average values and the confidence interval) for finding an optimal solution among different combinations of wireless devices. For most cases when the number of collaborators is below 10, the required time is less than 50 ms. In the worst case when 50 collaborators express interest of sharing the bandwidth, it still takes less then 0.7 s on average for the older generation phones. It should be noted that this optimisation algorithm is executed every collaboration selection window (default to 5 s) or when there is a change in the set of available devices.

E. Video streaming on Android emulator and phones

We streamed videos using ColStream in both the Android emulator and real phones. A demo recording [6] is available on YouTube showing the performance comparison. In the future work, we are planning to further refine ColStream to incorporate the Wifi-direct feature for direct communication between Android devices.

[6]Demo recording: http://www.youtube.com/watch?v=Syw2RUr0TwI

VI. Conclusion

Video streaming on the go is a very popular application, however the quality and user experience of video streaming depends on the available bandwidth. In this paper we proposed the ColStream system that significantly enhances the performance of video streaming by dynamically aggregating bandwidth of ubiquitous devices (the streaming initiator and willing collaborators). The paper described the ColStream's concept, architecture, design, prototypes and evaluation.

ColStream does not require support of specially-designed proxies/servers for allocating work to collaborating devices and provides an incentive mechanism that could support a real world deployment. It optimally selects collaborators by minimising the cost while maximising the throughput, and dynamically distributes the work among the collaborators adjusting their work share to the quality of their Internet connection. We plan to further improve accuracy of bandwidth prediction in 3G/LTE networks.

References

[1] "Cisco visual networking index: Global mobile data traffic forecast update, 20122017," *Cisco white paper*, Feb 2013.

[2] T. Stockhammer, "Dynamic adaptive streaming over http-design priciples and standards," in *Proc. of MMSys 11*, San Jose, CA, USA, Feb 2011, pp. 133–144.

[3] J.-S. Lee and B. Hoh, "Sell your experiences: a market mechanism based incentive for participatory sensing," in *Proc. of PerCom2010*, Mannheim, Germany, March 2010, pp. 60–68.

[4] A. L. Ramaboli, O. E. Falowo, and A. H. Chan, "Bandwidth aggregation in heterogeneous wireless networks: A survey of current approaches and issues," *Network and Computer Applications*, vol. 35, 2012.

[5] K. Chebrolu and R. R. Rao, "Bandwidth aggregation for real-time applications in heterogeneous wireless networks," in *IEEE Transactions on Mobile Computing*, vol. 5, April 2006, pp. 388–403.

[6] S. Barré, O. Bonaventure, C. Raiciu, and M. Handley, "Experimenting with multipath tcp," in *Proc. of SIGCOMM2010*, vol. 40, no. 4. ACM, Aug. 2010, pp. 443–444.

[7] P. Garbacki, A. Iosup, D. Epema, and M. Van Steen, "2fast: Collaborative downloads in p2p networks," in *Proc. of P2P 2006*, Cambridge, UK, Sep. 2006, pp. 23–30.

[8] X. Zhang, J. Liu, B. Li, and T. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *Proc. of INFOCOM 2005*, Miami, FL, USA, 2005, pp. 286–298.

[9] S. Jung, U. Lee, A. Chang, D.-K. Cho, and M. Gerla, "Bluetorrent: Cooperative content sharing for bluetooth users," in *Proc. of PerCom 2007*, White Plains, NY, March 2007, pp. 47–56.

[10] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: cooperative video streaming on smartphones," in *MobiSys 2012*, Low Wood Bay, Lake District, UK, June 2012, pp. 57–70.

[11] G. Ananthanarayanan, V. Padmanabhan, L. Ravindranath, and C.Thekkath, "Combine: Leveraging the power of wireless peers through collaborative downloading," in *Proc. of Mobisys 2007*, San Juan, Puerto Rico, 2007, pp. 286–298.

[12] T. V. Seenivasan and M. Claypool, "Cstream: neighborhood bandwidth aggregation for better video streaming," *Multimedia Tools and Applications*, pp. 2–31, 2011.

[13] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, pp. 369–395, 2004.

[14] D. Qiu and R. Srikant, "Modeling and performance analysis of bittorrent-like peer-to-peer networks," in *Proc. of SIGCOMM 2004*, Portland, Oregon, USA, August 2004, pp. 367–378.