A protocol for distributed multimedia capture for personal communicating devices

Xavier Le Bourdon
IRISA, Université de Rennes 1
Campus de Beaulieu
Rennes, France

Paul Couderc
INRIA
Campus de Beaulieu
Rennes, France

ABSTRACT

This paper proposes a protocol for supporting distributed capture of multimedia data over a set of personal communicating devices. The protocol enable the devices to cooperate spontaneously in order to improve the quality of the capture, by getting missing fragments or getting better one. It addresses the problem of temporal coherence when merging fragments captured by different devices, and propose solutions suitable for existing devices.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: Wireless communication; C.2.4 [**Distributed systems**]: Distributed applications

General Terms

Algorithms, Performance, Design

Keywords

mobile computing, wireless, multimedia capture, spontaneous collaboration

1. INTRODUCTION

An essential aspect of ambient intelligence is the link between information systems with the real world. To this end, sensors and networks of sensors play an important role in providing data and events from the real world. Sensors and networks of sensors are thus gaining attention from the research community. Enabling cooperation between these tiny devices, in an extremely dynamic context, poses real challenges, such as coping with very limited resources (energy, processing power, memory), high failure rate and node mobility.

Surprisingly, much less attention has been paid to what is becoming the most important sensor network ever deployed: the multimedia phone. While these are well-known as personal messaging devices, their role as sensors has been largely ignored in spite of their powerful capabilities in this respect. These devices can capture sound, picture and video, can cooperate with other sensors such as GPS positioning receivers and, of course, can communicate with global as well as local networks.

In this paper, we focus on collaborative capture, which consist in using a collection of personal devices dispered in a given area as a distributed sensing infrastructure. Said in another way, neighboring devices will be used as an opportunistic sensor network. The motivation for this idea is to improve the capture performance that can be expected from a single device, by getting *missing data* or *better data* from other devices (Figure 1). Consider, for example, someone arriving late in a meeting. Many phones, treated as audio capture devices, could have recorded the beginning of the meeting to allow the newcomer to catch up. Another usage would be, for someone capturing a video of an event, to get the sound from a device closer to the source to improve the clip quality. An important problem in collaborative capture is ensuring the global coherence of the collection of data fragments contributed from different nodes. In this paper, we will address the issue of temporal coherence of multimedia streams, such as audio and video synchronization.

The rest of the paper is organized as follows: the second section presents the context and our system model; the third and fourth sections detail the temporal synchronization issues, and their evaluations. Sections five and six discuss related works and some research perspectives.

AUTONOMICS 2007, 28-30 October 2007, Rome, Italy
Copyright © 2007 ICST 978-963-9799-09-7
DOI 10.4108/ICST.AUTONOMICS2007.2266

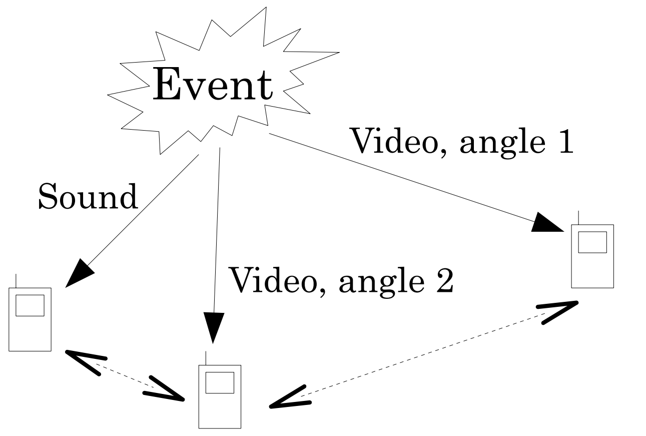

Figure 1: Collaborative capture of an event

2. CONCEPTS AND SYSTEM MODEL

In this part, we present the concepts proposed to enable collaborative capture. The first and the second subsections present the paradigms of communication used by the nodes from a network point of view, and then from the application layer. The third subsection presents an optimization, enabling the user to manage a high quantity of multimedia objects. The last subsection presents the way in which terminals manage the multimedia objects.

2.1 Capture model

The basic units considered for capture in our system are multimedia objects or streams defined by a temporal slice (or instant in case of picture), and the associated data. They can be pictures, audio stream, and video streams. From the user perspective, some objects can be considered as related fragments of a common multimedia objects, such as in the case of an audio and a video streams captured at the same time, or two audio slices of the same speech. From the system perspective, potential relations between objects are identified in terms of *physical context proximity* [16]: objects captured close together in time and space are considered strongly related while objects captured far apart in time or space are considered weakly related. In the rest of the paper, we will refer only to *multimedia objects.*

2.2 One hop paradigm

Short range wireless mobile communication technologies like Bluetooth or IEEE 802.11 into personal devices enable them to spontaneously collaborate together when being in the same physical area. The one-hop paradigm is a wireless communication concept where a device can communicate with another device if it can directly reach its radio range, without routing support. Thus the information can only be exchanged between two nodes that are in the same vicinity. For example, in figure 2, the peers A and C can not communicate because they are not in communication range. However, B can communicate with A or C.

The one-hop paradigm ensures an implicit geolocalization of the data, bounded by the signal range. This property is interesting as it link the data to particular physical context. When an event is captured, the availability of the captured data is limited to the users close to this event, which are

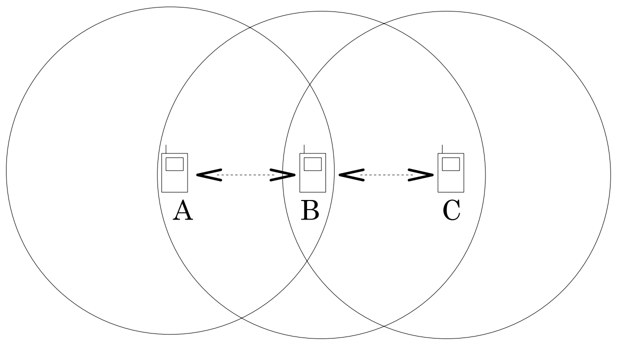

Figure 2: One hop paradigm

likely those who have a potential interest for it. For example, the ability to download a recently recorded conference means that the user is in the building of the conference, and that he may be interested in viewing the conference.

Note that this paradigm does not limit multi-hop information propagation: a node that has received a piece of data can become a new source and share this piece of data with other nodes. The sharing devices can also physically move, spreading the data to devices that were previously unreachable [8]. However, the implicit relevance provided by the physical context decreases as the node moves away from the capture location. To support relevance ranking of captured media, we track two important meta data: the time elapsed since the creation of data, and the number of nodes the data has traveled. The higher these counters are, the further a device is from the location of the data creation, resulting in the smallest potential interest this piece of data has for the user.

2.3 Collaborative behaviors

In this part, we present interactions principles of collaborative capture, highlighting the features and advantages they introduce. Collaborative capture belongs the class of *spontaneous information systems* [22], in the sense that they are fully distributed over the participating nodes, and nodes may spontaneously join or leave the system at anytime in relation to their physical mobility.

In our model, each terminal can capture and store multimedia objects, like videos, audio streams, or photos. Each terminal manages a library and spontaneously shares the captured multimedia objects. Other terminals can ask for an index of a terminal's library or request a specific file to download. Thus, a user can browse the available multimedia objects and download the ones he's interested in.

Our system also tries to automatically increase the *quality* of the captured multimedia objects by downloading complementary ones, in terms of completeness, or capture quality (depending on sensors performance and sensors location). When a user captures an event, the system looks for available multimedia objects that could be complementary to the current one. These multimedia objects could increase the global quality, such as having a better sound sampling, or they could be complementary in terms of timeline. For example, imagine a user who is late for a conference. As soon as he arrives, he starts recording the conference with his terminal, while another user, who was not late, had started his recording at the beginning of the conference. The terminal of the first user automatically tries to download the beginning of the conference from the second user. When the newly arrived the user has a better angle to capture the video, or his device can capture the video with a better resolution, the capture process has the opportunity to continue with improved performance. In this case, the other terminals would complete their capture by downloading their data from the new node instead of using their own sensor.

Another collaborative behavior is the ability to ask a distant terminal to record an event. Let's imagine two users who could each capture an event from their own point of view. The two multimedia objects could be useless if they are taken separately, but could generate a useful media stream if they are used together. For example, in a conference, a user could be near the speaker and could easily capture a recording of the speech. Another user could be in the middle of the room and could capture the video of the slides but with poor sound quality. Each multimedia object taken separately is useless. With our system, one of the users can ask for a distant record. When the user who is in the middle begins to capture the slides, his terminal asks the other terminals in the room to capture the audio. A person who is near the speaker can accept to record the sound in exchange for the video.

We also extend this concept to describe an unselfish attitude of the users. A user might not be able to capture one event, because of the limits of his capture device, or because he can not physically capture the event. For example, a tourist with a poor quality camera phone could request the other tourists to take a photo of a monument. Another example is the case of simultaneous conferences in nearby rooms. A spectator might want to view the two conferences but is not physically able to attend both of them. He could go to one conference and, with the help of his terminal, ask a spectator who is in the other room to record the other conference.

2.4 Choice helper

With digital mobile capture devices, it becomes easy to produce multiple multimedia objects. It is now free to capture digital objects (for example digital photos do not need to be developed), it is easy to delete unwanted multimedia objects, and storage cost is sharply decreasing. In our system, users could be easily overloaded when browsing available multimedia objects on other nodes. Moreover, as seen is the previous section, the users could be busy when they need to download the most important multimedia objects.

We use some quality criteria to organize, even to prefetch, multimedia objects. By organizing multimedia objects, the system helps the user to pick those he be more likely interested in. This is a useful feature when there is a large quantity of available multimedia objects in the vicinity. Moreover, by prefetching the potentially interesting multimedia objects, the system gets rid of the probability that available ones could disappear because of the mobility of the nodes. Consider for instance a user who is shooting video of an event from beginning to end, such as live music demonstration in the street. Other people may have passed near

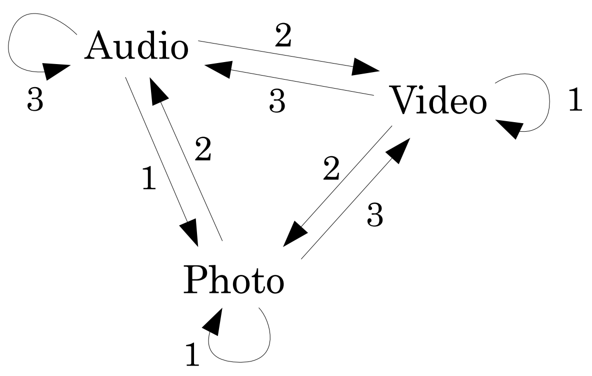

Figure 3: Example of complementarity order scheme

the event, taken some photos or videos, and continued on their way. Some of these objects may interest the first user, whose has been capturing the event from the beginning; his attention is focused on his own filming action, and he can not be interrupted without compromising his own capture. Thus, by downloading in the background other multimedia objects from nearby devices, the user will have the choice to save these potentially complementary or interesting multimedia objects that otherwise he would not have been able to do because he could not be interrupted.

The list below details the criteria and heuristics we use to classify multimedia objects as a function of the potential interest to the user:

- Global quality of the recording. For example, a user may prefer a high video quality over a lower resolution one. The tradeoff is the size of the file. Indeed, a high resolution multimedia object will take more time and more energy to download than a medium resolution multimedia object.

- Physical context proximity, in terms of time and space distance to the source. As seen previously, the one-hop paradigm provides an implicit contextualization, by physical proximity. The shorter the time elapsed between the current time and the time the multimedia object was created is and the smaller the number of nodes the multimedia object has gone through, the nearest the user is from the real source. We stated earlier that we will assume that being near the source is related to its relevance for the user.

- Small time intervals between multimedia objects. If two multimedia objects were taken at a similar date and time, they are more likely multimedia objects of the same event especially if they have been taken by the same author. For example, the photos of a birthday are taken at a similar time. If a user has downloaded some of these photos, he might want all the other ones too.

- Scarcity of a multimedia object. If a multimedia object is not widely distributed over the local network, there is a high probability that it will become unavailable soon, because of the mobility of the nodes. By downloading the scarce multimedia objects, the nodes ensure its availability even after the departure of the source node.

- Complementarity scheme for types of multimedia objects. We notice that we can deduce a scheme describing the user's wishes for the type of multimedia objects, based on the multimedia objects they already have captured (see Figure 3). For example, if they record the sound stream of an event, they usually prefer to download photos of the event to illustrate their sound stream, then maybe a video stream, but rarely another sound stream with an equivalent quality. If a user has a photograph, however, he will prefer to download other photos to complete his album, and maybe some sound streams to illustrate his photos, or even a video stream.
- Semi-automatic recommendations. Users could download other media to complete their current ones. For example, a user could use two video streams of a same event to obtain a multi-angled video stream. The user could link multimedia objects if he wants them to be played simultaneously. Thus, other users will have an implicit recommendation: if they download one of the multimedia objects, they might be interested in the linked ones also.

All of these criteria are not universal. They can vary from one user to another, and they can also vary from one event to another. For example, one user could prefer to download medium quality multimedia objects to save energy. A user might want to download a multimedia object from a conference that was taken close to the speaker while he wants to download a multimedia object from a sporting event that was taken far away. Thus, we propose to the users to set the importance of these criteria according to their tastes or their situation. It is also possible to do profiling to automatically detect the preferences of different users.

2.5 Multimedia objects descriptors

Our system needs to manage multimedia objects. From a local point of view, each node stores different kind of multimedia objects (video, audio, photo) that have been captured or that have been downloaded from other nodes. These multimedia objects are made of two components. One part is the multimedia data, for example the video stream, the audio stream or the picture. The other part is the metadata needed to browse the multimedia objects and to synchronize them.

We use an XML format to store the metadata of the multimedia objects (Figure 4). Contrary to a classic metadata format, like EXIF, this descriptor can be used without multimedia data. In other words, a descriptor can be sent via the network as a query or as a response. For example, the system can broadcast a partial descriptor to request a matching multimedia object. In response, the node will receive descriptors, so it can choose the best multimedia objects to actually download.

The descriptor contains a unique identifier (uid) and an author identifier (author) to identify a multimedia object, even

```
<?xml version="1.0" encoding="UTF-8" ?>
<metadata>
  <uid>0018F834FB2B-245</uid>
  <author>00-18-F8-34-FB-2B</author>
  <filename>0018F834FB2B-245.avi</name>
  <date>25-02-2007</date>
  <time>334477</time>
  <duration>11356</duration>
  <quality>
    <audio_codec>mp3</audio_codec>
    <audio_bitrate>128</audio_bitrate>
    <video_codec>mpeg2</video_codec>
    <video_bitrate>4000</video_bitrate>
    <timestamp_accurency>12</timestamp_accurency>
    <hop>2</hop>
  </quality>
  <linked>
    <uid>0018F834FF21-3</uid>
    <uid>00348890A910-20</uid>
    ...
  </linked>
...
</metadata>
```

Figure 4: XML multimedia object descriptor

if this object is replicated over the network. The descriptor also contains timing data (date, time and duration) to synchronize multimedia objects. The quality details, like the audio bit-rate or the codecs used to compress the data allow devices to select the multimedia objects with the best quality among the available ones, as seen is the previous subsection. Finally, the descriptors contain a list of unique identifiers, linked to the current multimedia object. This could be useful if two multimedia objects are taken of a same event.

3. SYNCHRONIZATION ISSUES

One of the main features of collaborative capture is the possibility to synchronize multimedia objects that have been captured by different users. For example, a user could use two or more video streams of a same event and switch between them to view the recorded event from the best angle. The precision in the timing of the multimedia objects becomes crucial for a good synchronization to appear seamless.

3.1 Motivation

The system must know the temporal relationship between sets of multimedia objects. For example, the system should be able to synchronize a video stream captured by a user and an audio stream captured by another user. Therefore, each multimedia object has to be timestamped. Subjective studies have shown that a video stream and an audio streams do not need to be exactly matched, but that a skew of 80-100ms is below the limit of human perception [9, 10, 7, 12]. The first solution is to get the time from the internal clock of the terminals. Indeed, each terminal (cellular phone, camera, computer, etc.) encompass a cheap clock which is easily accessible by the system. However, these clocks are usually manually set by each human owner of the terminals. They are not precisely synchronized on the absolute time, and have each a different offset compared to the actual time.

Moreover, depending on their quality, clocks drift compared to the actual time.

A solution could be to assume the presence of an on-board precise clock, like a GPS device, for each terminal. However, this type of device is very expensive in terms of cost and in terms of energy. Another solution, used in some distributed networks, is to run a protocol which regularly synchronizes all the clocks of the nodes at the time managed by a server [14, 13, 11]. It would be possible for a group of terminals to agree on a master node, and to synchronize their time to the clock of the master. With this type of protocol, the owners have to accept that the clock of their terminals could be set to a subjective time. It could also result in strange behaviors, particularly for other running applications, like a calendar or a chronometer. The last classical solution is to let each local clock of the node been free-running, i.e. one should not adjust the local clocks. Instead, the synchronization scheme should build up a table of parameters that enables each node to convert its local clock to that of another node, and vice versa [20, 17, 6]. We use a similar solution to synchronize the multimedia objects on the nodes.

3.2 Dynamic timestamp calibration

Our system needs a protocol to synchronize the shared multimedia objects with three constrains. The first constrain is the inability to set the clocks to a new date and time. The second constrain is that the timestamps of multimedia objects being on a terminal must be set on a coherent timeline. The last constrain is that the error of synchronization should be less than to 100ms, so that any mismatch will not be noticed.

We propose to dynamically change the timestamp of the multimedia objects when they are shared over the network. Each terminal has an independent clock, with an error between its time and the actual time. When a user captures an event, the created multimedia object is timed depending on the terminal clock. When a multimedia object or its descriptor is sent to another device, the sender and the receiver evaluate the relative offset between their clocks:

$$\mathit{Offset}_{r/s} = T_r(T_0) - T_s(T_0)$$

The receiver then sets the start time of the multimedia object to a new timestamp depending of the local clock:

$$\mathit{Timestamp}_r = \mathit{Timestamp}_s + \mathit{Offset}_{r/s}$$

With this protocol, a device can synchronize its multimedia objects with the multimedia objects it receives. Thus, it can also synchronize different multimedia objects received from different devices. Moreover, it can send these received multimedia objects to other devices, using the same protocol.

3.3 Offset calculation

We proposed a protocol where the receiver device had to determine the offset between its own clock and the clock of a distant device. This is done by exchanging messages over the network, containing a reading of the sender clock. However, there is a delay between the clock reading and the comparison between the two clocks. This delay is mainly due to the transmission of the message over the physical network and though the network protocol stacks.

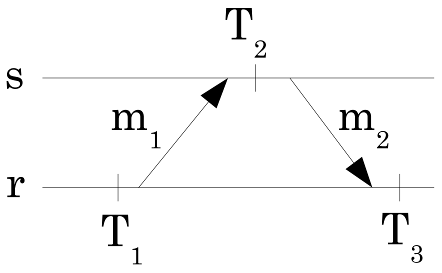

Figure 5: Measurement of the relative offset

We can bound this delay by reading the clock before sending a request message and after receiving the timestamp message (Figure 5). Because a network message can not be received before being sent, the precision of the calculated offset is bounded:

$$\mathit{Max\_error} = T_3 - T_1$$

Assuming the delay is the same for the two messages, we can evaluate the offset between the two clocks:

$$\mathit{Offset}_{r/s} = \frac{T_1 + T_3}{2} - T_2$$

Actually, because of the jitter, the delay is not exactly the same for the two messages. To increase the precision, we use redundancy, detailed in the next part.

Each time a multimedia object or its descriptor is sent over the network, the receiver evaluates the offset between its own clock and the clock of the sender. To reduce the number of offset calculations, each node keeps a cache of the offsets between its clock and the clock of the other nodes that were already evaluated.

3.4 Timestamp precision

The timestamping of the multimedia objects is crucial to have good synchronization between several multimedia objects. For example, a user could download a video stream from a user who has captured his video from the best angle, and an audio stream from a second user who has recorded the sound near the event. In this case, the error of synchronization between the two multimedia objects should not be more than 100ms in order to be imperceptible by humans.

In our system, there are three sources of imprecision. The first one takes place when a device starts to capture a multimedia object. There is a delay between the reading of the time and the actual recording. The system may need to acquire scarce or exclusive resources, fill buffers with media data, or perform other start-up processing. This delay can be reduced by preparing the capture device before actually reading the time. Empiric measurements show us that this delay is less than 10ms, which is acceptable compared to the maximum of 100ms. Moreover, if this delay is similar for every device, it becomes invisible because of the same delay for every multimedia object.

The second source of error is the drift phenomenon. Indeed, clocks are not perfect and drift compared to the actual time. The offset between two clocks should be the same anytime; in fact this offset is slowly changing depending of the drift rate of each clock. Empiric measurements show us that the relative drift between two clocks is nearly linear and is less that 40ms per hour. This potential error is not negligible compared to the 100ms max. The first solution is to exclude the synchronization of multimedia objects when the current calculated offset could be too far from the offset that was between the clocks of the devices when the multimedia object was captured. For example, we calculate the timestamp according to the local clock of a multimedia object that was timed two hours ago, but we do not guarantee its synchronization with another multimedia object. A better solution we plan to implement in future works can only been used if the two devices can communicate for a sufficient duration. This solution consist in keeping in memory all the calculated offsets between the two clocks. When a device needs to synchronize a multimedia object, it uses the offset that was calculated when the multimedia object was timed. If this offset has not been measured, for example because the two devices could not communicate, and if the system has a sufficient quantity of measured offsets, it does a linear regression. Thus, it can deduce the drift rate and counterbalance it. The calculation of the drift to counterbalance it is commonly used in clock synchronization protocols [20, 21].

The last source of error is the jitter in the network messages in our protocol to measure the offset. We counterbalance this jitter with a redundancy. By measuring the offset a few times and calculating a mean of the values, we reduce the error. We tried different kind of mean calculations: with two, three and four successive values; with the two more proximate values out of three; and with the three more proximate values out of four.

Each time a multimedia object or its descriptor is sent over the network, the system evaluates its new timestamp, using the offset evaluation. Therefore, each time a multimedia object is forwarded from a device to another one, an error is added to its timestamp. We use the *timestamp_accurency* metadata in the multimedia objects descriptors to limit this error: each time a multimedia object is forwarded though the network, this counter is incremented, according to the maximum error calculation previously presented. Thus, if a multimedia object is shared by more than one device, we calculate its new timestamp from the device which has the descriptor with the smallest potential error. Moreover, a device can download and use the descriptor from a first device, with the smallest error value, and download the multimedia object from another device, with a greater bandwidth.

4. EVALUATIONS

We proposed a system to share captured multimedia objects. In this part, we evaluate our system from two points of view. The first point is the efficiency of capturing a public event. Indeed, a group of users capturing some parts of an event could generate multimedia objects that cover the whole event. The second point of evaluation is the precision of timing. We saw that this precision is crucial to synchronize different kind of multimedia streams, such as a video stream and an audio stream.

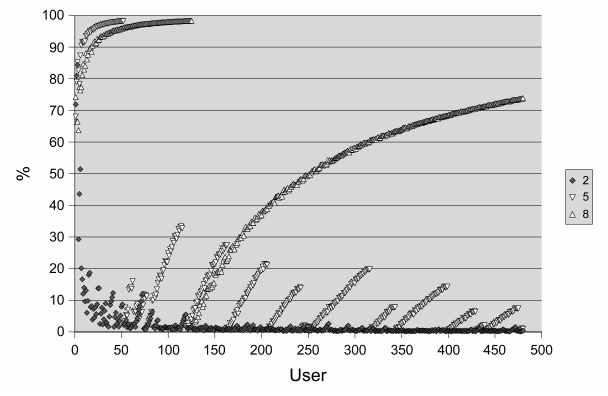

Figure 6: Event covering for each user

4.1 Event capture

In this simulation, we measure the efficiency of the capture of a public event. Our model is based on the typical behavior of street walkers. A street walker is attracted by a group of people; he stays passive for some minutes; if the event seems interesting, he decides to capture a short video; finally he puts his device in his pocket and goes away. We assume that a user capture device can not communicate with other devices when the user is not in the crowd.

We made simulations with different average densities of capturing users in the crowd. The graph in figure 6 shows the percentage of the event captured or downloaded by each user, from the beginning of the event to the departure of the user. Thus, we suppose that a user that is leaving the crowd was interested in the beginning of the event but not interested in the future of the event. In this graph, we choose to show three simulations: for the average densities of 2, 5 and 8 users.

On this graph, we see that the coverage of the media increases for the successive users. This is due to a virtual local storage of the media objects in the crowd. Indeed, the devices download multimedia objects from other devices that are already in the crowd, becoming a potential source of these multimedia objects. However, we also see that sometimes, the coverage sharply decrease for all the users. This is due to a temporary lack of users in the crowd. In this case, there are no more devices with the older coverage of the event.

4.2 Offset calculation

In this part, we evaluate the precision of the measured offset using different calculation schemes. We used Dell D610 laptops with built-in Intel 2200BG 802.11b/g WiFi network adapters. We compared six offset calculations. The first one is the direct use of the offset measurement with the protocol presented in the section 3.3. Three others calculations are the arithmetic means with two, three and four successive offset measurements. We then calculate the mean of the two most proximate values out of three successive mea-

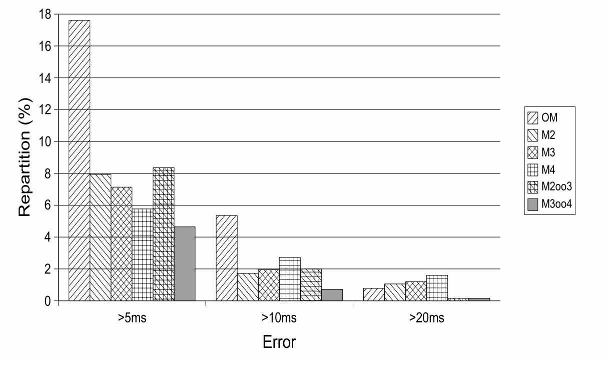

Figure 7: Repartition of error in offset calculation

surements. Finally, we calculate the mean of the three most proximate values out of four measurements.

The figure 7 shows the repartition of erroneous offset for each calculation scheme. For example, more than 17% of the measured offsets have an error greater than 5ms with the one time measure (OM). That means that near 83% of the measured offsets have an error of less than 5ms. We see that the calculations with a mean of two to four successive values (M2, M3, and M4) reduce dramatically the quantity of offset measurement with an error greater than 10ms. However, the quantity of measurements with an offset greater than 20ms is increasing. This is due to the repartition of the jitter. Indeed, the jitter of a few messages is very high, and so has a big impact on the mean. Our solution is to counterbalance these kind of messages by excluding measurements that are too far from the other ones. We see on the figure 7 that the mean of the two most proximate values out of three successive measurements (M2oo3), and three out of four (M3oo4), give better results than a simple arithmetic mean. However, the measurements that have an error greater than 20ms do exist (0.08%). In fact, it is impossible to get rid of every erroneous measurement with a finite number of measurements. We can reduce the probability of these errors by increasing the number of successive measurements.

5. RELATED WORK

Some work has already been done regarding multimedia object sharing in wireless networks. Sarvas and al. [18] proposed a centralized system to share photos. Aberer and al. [1] proposed a P2P system at the world scale, to share photos using the Internet. Their interesting approach does not use the proximity network, and thus does not take advantage of the similar interest the users could have in the multimedia objects. Shu and al. [19] proposed a system to spontaneously allow groups of mobile users to share files. In all of these works, the systems do not care about the potential complementarity of some multimedia objects. Each file is taken separately and these systems do not propose to synchronize multimedia objects, or to download interesting ones. Moreover, these systems do not deal with the clock delays between different nodes. Finally, they do not propose a way to sort multimedia objects according to potential user interest.

Agarwal and al. [2] have faced similar problems with the capture from several media sources and the display on several destinations. They used a centralized approach, with passive nodes, and with a master in charge of the synchronization and storage of the media streams.

A lot of work has been done concerning clock synchronization in distributed networks [20, 21]. An idea similar to ours is shown using the probabilistic protocols in sensor networks [5, 3, 15]. However, their systems are not designed for successive temporary synchronizations between two nodes, but for the synchronization of nodes clocks in a network whose landscape stays relatively the same. Also, because of the propagation of the timed multimedia objects, we propose to calculate and store the potential error in offset measurement. This piece of information allows a node to measure the offset from the best source, obtaining the smallest potential error.

More closely related to collaborative capture is the Mosaic collaborative backup system [4]: nearby mobile devices spontaneously collaborate to improve their data resilience. However, in contrast to collaborative capture, nodes do not have to interpret the meaning of data fragment (which are scrambled for security and privacy).

6. CONCLUSION AND FUTURE WORKS

This paper has presented the novel concept of collaborative multimedia capture in a mobile wireless local network. This concept consist of capturing an event with several mobile capture devices, and exchanging and synchronizing the resulting multimedia objects to increase the global quality of the event capture. We proposed an architecture to manage metadata of the multimedia objects, to share multimedia objects, and to encourage users to collaborate. We also provide a protocol of timestamp calibration to get rid of the delays between the different hardware clocks of different mobile nodes. A novelty in wireless clock synchronization is the concept of precision estimation stored in the multimedia objects metadata. This information allows the best source to estimate a new timestamp to be chosen for multimedia objects.

This architecture has been simulated in a scenario of capturing of a public event. Our results show that an implicit geolocalized virtual storage spontaneously emerges from the crowd of mobile users. We also have evaluated the precision of the offset calculation in a real WiFi network. The results show that the error is acceptable, compared to the human perceptible delay of sound and video synchronization.

In further works, we intend to improve the error estimation, in particular by compensating the drift between the hardware clocks. More research could also be done concerning privacy and security. Indeed, users might want to choose which multimedia objects should be public, which ones should be semi-private (for friends or co-workers), and which ones should be private. This is crucial when the captured event is not a public event, like a birthday or a meeting at work. Further research is needed on the energy consumption of a system like this. In mobile computing, the terminals do not have infinite energy and the use of wireless networks can be expensive in terms of energy consumption. Reducing the usage of the network is a way to reduce energy consumption. Further research should also be performed on the idea of *incentives* and how they would affect a system such as this. For example, in some P2P networks, users can download multimedia objects without sharing their own. How would encouraging users to share their own multimedia objects change the usage model of this system? Finally, we are investigating spatial coherence of other type of distributed capture, in particular the problem of global coherence of geotagged photo collections.

7. REFERENCES

[1] K. Aberer, P. Cudré-Mauroux, A. Datta, and M. Hauswirth. PIX-grid: A platform for P2P photo exchange. In J. Eder, R. Mittermeir, and B. Pernici, editors, *CAiSE Workshops*, volume 75 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2003.

[2] N. Agarwal and S. H. Son. A model for specification and synchronization of data for distributed multimedia applications. *Multimedia Tools Appl*, 3(2):79–104, 1996.

[3] K. Arvind. Probabilistic clock synchronization in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, PDS-5(5):474–487, May 1994.

[4] L. Courtès, M. Killijian, and D. Powell. Storage tradeoffs in a collaborative backup service for mobile devices. In *Proceedings of the 6th European Dependable Computing Conference*, pages 129–138, Coimbra, Portugal, October 2006.

[5] F. Cristian. Probabilistic clock synchronization. *Distributed Computing*, 3(3):146–158, 1989.

[6] J. Elson and K. Römer. Wireless sensor networks: a new regime for time synchronization. *Computer Communication Review*, 33(1):149–154, 2003.

[7] Escobar, J, Deutsch, D, and Partridge, C. A multi-service flow synchronisation protocol. *BBN STC Tech Report*, Mar. 1991.

[8] A. Heinemann and M. Mühlhäuser. Spontaneous collaboration in mobile peer-to-peer networks. In R. Steinmetz and K. Wehrle, editors, *Peer-to-Peer Systems and Applications*, volume 3485 of *Lecture Notes in Computer Science*, pages 419–433. Springer, 2005.

[9] A. W. I. Kouvelas, V. Hardman. Lip synchronisation for use over the internet: analysis and implementation. In *Global Telecommunications Conference, 1996. GLOBECOM '96*, pages 893–898, Nov 1996.

[10] P. W. Jardetzky, C. J. Sreenan, and R. M. Needham. Storage and synchronization for distributed continuous media. *Multimedia Syst*, 3(4):151–161, 1995.

[11] H. Kopetz and W. Ochsenreiter. Clock synchronization in distributed real-time systems. *IEEE Transactions on Computers*, C-36(8):933–940, 1987.

[12] L. Lamont, L. Li, R. Brimont, and N. D. Georganas. Synchronization of multimedia data for a multimedia news-on-demand application. *IEEE Journal on Selected Areas in Communications*, 14(1):264–278, 1996.

[13] K. Marzullo and S. S. Owicki. Maintaining the time in a distributed system. *Operating Systems Review*, 19(3):44–54, 1985.

[14] D. L. Mills. Internet time synchronization: The network time protocol. In *Zhonghua Yang and T. Anthony Marsland (Eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press*. IEEE Computer Society Press, 1994.

[15] S. PalChaudhuri, A. Saha, and D. B. Johnson. Probabilistic clock synchronization service in sensor networks. In *Technical Report TR 03-418, Department of Computer Science, Rice University*, 2003.

[16] J. Pauty, P. Couderc, and M. Banâtre. Using context to combine virtual and physical navigation. Technical report, INRIA, February 2005.

[17] K. Römer. Time synchronization in ad hoc networks. In *MobiHoc*, pages 173–182. ACM, 2001.

[18] R. Sarvas, M. Viikari, J. Pesonen, and H. Nevanlinna. Mobshare: controlled and immediate sharing of mobile images. In H. Schulzrinne, N. Dimitrova, A. Sasse, S. B. Moon, and R. Lienhart, editors, *ACM Multimedia*, pages 724–731. ACM, 2004.

[19] K. Shu, C. Swindells, D. Chai, , and K. M. Inkpen. Windowspaces to share our digital media. Technical Report TR 2002-04, School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, June 2002.

[20] F. Sivrikaya and B. Yener. Time synchronization in sensor networks: a survey. *IEEE Network*, 18(4):45–50, 2004.

[21] B. Sundararaman, U. Buy, and A. D. Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.

[22] D. Touzet, J.-M. Menaud, F. Weis, P. Couderc, and M. Banâtre. Side surfer: enriching casual meetings with spontaneous information gathering. *SIGARCH Comput. Archit. News*, 29(5):76–83, 2001.