Cost-effective Web-based Media Synchronization Schemes for Real-time Distributed Groupware

Maarten Wijnants, Peter Quax and Wim Lamotte
Hasselt University – tUL – iMinds
Expertise Centre for Digital Media, Wetenschapspark 2, 3590 Diepenbeek, Belgium

Keywords: Multimedia, Inter-Destination Media Synchronization (IDMS), Group Synchronization, Cost-benefit Analysis, synchronous MediaSharing (sMS).

Abstract: A multitude of application domains benefit from synchronized multimedia content presentation and playback across spatially scattered client stations. Online remote tutoring, telework and teleconferencing, and the non-co-located synchronous browsing through a digital photo album are merely some examples of potential use cases, which are referred to collectively as real-time distributed groupware. This paper presents five concrete schemes that realize media synchronization over the Web. The proposed techniques have been implemented and experimentally validated as part of the synchronous MediaSharing (sMS) service, a framework which grants geographically dispersed users the ability to share and synchronously consume digital pictures and video clips. All five schemes are completely Web-compliant, achieve relatively loose synchronization accuracy, and can be categorized as being non-distributed in the sense that they require centralized coordination. Besides being technically compared, the synchronization solutions are subjected to a high-level assessment in terms of their infrastructural requirements and the thereby associated necessary operational expenditure.

1 INTRODUCTION

The shared, co-located consumption of multimedia such as pictures and video clips occupies an important niche in media-centered communication paradigms. In this model, people physically convene in order to consume particular media content in the presence of each other. The co-located setting hereby often leads to highly rich human interactions. As an example, it has been found that jointly browsing through a physical photo album gives rise to the evocation and potentially even the reliving of shared memories (Sarvas and Frohlich, 2011). Another example is given by the fact that people typically enjoy watching the live television broadcast of important sport events together with friends, because the social contact adds an additional and highly valued dimension to the setting.

Unfortunately, numerous practical factors like travelling overhead and time constraints increasingly prevent group members from physically meeting up, both in professional and recreational scenarios. At the same time, due to the popularization of Online Social Networks (OSNs), international contacts are becoming common practice. To mitigate the potential disadvantages (concerning social interaction) induced by a distributed setting, we have previously proposed a digital media sharing and synchronization framework called synchronous MediaSharing (sMS) that emulates a sense of co-presence and physical togetherness for geographically dispersed content consumers (Wijnants et al., 2012a). The framework is exclusively built around Web technologies and standards in order to maximize its adoption and market penetration potential (by avoiding platform lock-in). User-centric research has revealed that the sMS tool nicely fits in with the contemporary content sharing and consumption habits of multimedia amateurs; furthermore, the system has been found to be largely complementary to existing digital multimedia sharing services like Flickr and YouTube (Wijnants et al., 2012b).

At the core of any synchronous media sharing platform lies its real-time content synchronization procedure. The synchronicity requirement namely dictates that spatially distributed participants need to be presented the same content at (approximately) the same time. In the literature, this concept is sometimes denoted by the term group synchronization (Boronat et al., 2009) or Inter-Destination Media Synchronization (IDMS) (Stokking et al., 2010). This paper will present five synchronization schemes that have been

Wijnants M., Quax P. and Lamotte W. (2013).
Cost-effective Web-based Media Synchronization Schemes for Real-time Distributed Groupware.
In *Proceedings of the 9th International Conference on Web Information Systems and Technologies*, pages 149-154
DOI: 10.5220/0004369401490154

developed in the context of the sMS platform. Besides describing the technical implementation of the different methods (in Section 2), they will also be compared in terms of their economic effectiveness by investigating their deployment and hosting demands (in Section 3). It is hereby emphasized that the findings and results described in this article are not bound to the sMS framework but instead, through generalization, are applicable to many Web-based services that are conceptually situated in the *same time, different place* category of the groupware typology introduced by Johansen (Johansen, 1988).

2 SYNC SCHEMES

This section presents a total of five schemes that achieve content synchronization in the sMS system. Given the system's Web-conformance design premise, an unconditional prerequisite for the developed group synchronization solutions was that they had to be completely Web-compliant. Another trait that is shared by all proposed schemes is that they are not very strict, in the sense that they yield relatively loose group synchronization; timing discrepancies amounting up to a handful of seconds might occur across participating sites. Qualitative end-user research has proven that the level of inter-destination synchronization that is attainable by the presented schemes suffices for entertainment purposes and for recreational applications of the sMS platform (Wijnants et al., 2012b). Other application domains (e.g., distributed learning) might demand more strict synchronization solutions (Gutwin, 2001). A next similarity is that all five implementations conceptually belong to the *master-slave receiver* category in Boronat et al.'s taxonomy of group synchronization schemes (Boronat et al., 2009). Finally, the schemes are all non-distributed in nature as they depend on a central process to streamline the synchronization procedure.

2.1 Webserver plus AJAX

The rationale of this scheme is that synchronization records are stored in, and fetched from, a centralized location. In particular, whenever a user actively modifies the state of a sMS session, an AJAX transaction is initiated by his client to invoke a server-side PHP script, which in turn inserts the transported content synchronization payload in a relational database (RDBMS). Conversely, clients periodically solicit the most recent synchronization data that pertains to their current sMS session from the server (again via AJAX) and apply the response locally. Factual implementation details can be found in (Wijnants et al., 2012a).

2.2 Synchronization in the Cloud

An interesting alternative to the use of a dedicated and privately hosted webserver for the distribution of synchronization instructions (as described in Section 2.1), is to resort to the Platform as a Service (PaaS) cloud computing service model. Due to hardware virtualization technology and economies of scale benefits, a cloud-deployed solution will typically be more resource-efficient and cost-effective (see Section 3). At the same time, the synchronization scheme automatically inherits the advantageous properties that are typically associated with cloud hosting, including dynamic scalability, the availability of powerful cloud-side APIs that foster application implementation, and highly reliable Quality of Service (QoS) guarantees.

In this particular case, the synchronization model has been realized on Google App Engine (GAE), the PaaS cloud computing service provided by Google. The model's design largely mimics that of the dedicated webserver deployment from Section 2.1: clients rely on AJAX to push sMS synchronization data to a back-end repository, and remote clients obtain the pushed data from this repository. Some significant differences in terms of implementation however exist between both approaches. First, the back-end software has in this case been implemented as JavaServer Pages (JSPs) instead of as PHP scripts. Secondly, the synchronization instructions are server-side persisted by means of a schemaless NoSQL object store that guarantees automatic data storage scalability through the use of a distributed architecture. Third, clients do not need to explicitly and periodically poll the server to be informed of the most up-to-date sMS session state. Instead, as soon as the cloud application receives novel synchronization instructions, it will instantaneously and automatically push these instructions to all clients that are participating in the involved sMS session. This type of selective broadcast behavior is enabled by the Channel API service for Java-coded GAE applications.

The cloud solution technologically clearly outperforms the dedicated webserver synchronization scheme. Not only do GAE's intrinsic scalability features guarantee adequate performance under high workloads and large data sets, the push-based Channel API additionally expedites the dissemination of synchronization records among involved client terminals and as such improves sMS session consistency.

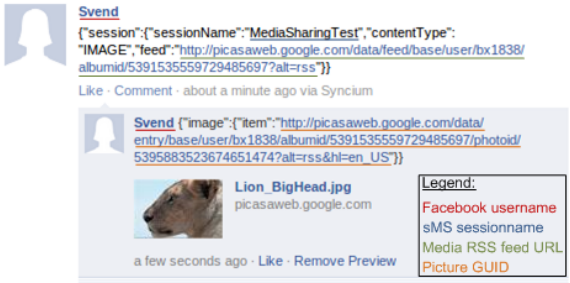

Figure 1: Synchronization via a public Facebook Page.

2.3 XMPP MUC

The Extensible Messaging and Presence Protocol (XMPP) is essentially intended to realize the distribution of (instant) messages and presence information among distributed sites. It is an open XML-based IETF standard that has been designed with extensibility in mind. One example of an XMPP extension is the Multi-User Chat (MUC) specification. Via the definition of a room or channel concept, this extension introduces textual group chat functionality in XMPP that unlocks controlled one-to-many text-based communication. Note MUC's large conceptual resemblance to the GAE Channel API discussed in Section 2.2. Also remark that, thanks to the openness of the protocol, anyone is allowed to develop and host a (proprietary) XMPP server; many public servers are available to which users can connect free of charge.

The XMPP MUC functionality has been exerted to implement inter-destination content synchronization according to the publish-subscribe interaction paradigm. sMS session participants are grouped in a unique MUC room, after which synchronization-related messages are broadcast within this room. This implies that, when a user modifies the state of the session, his client transmits a state description message to his XMPP server, which in turn propagates the message to all other session participants on the basis of the member list of the corresponding MUC room.

2.4 Facebook

Facebook offers a number of SDKs for different programming languages (including PHP and JavaScript) that enable software developers to interface with the platform and to build socially-inspired applications. The proposed Facebook-mediated synchronization approach has been realized exclusively by means of the Facebook JavaScript SDK.

The basic idea of this implementation is to exchange sMS session state information by means of a specialized yet public Facebook Page (see Figure 1). Whenever a new sMS session is created, the initiating client publishes a post on this Facebook Page and records the post's unique Facebook ID. The posted message holds configuration information about the session, including the URL of the Media RSS feed that syndicates the content that is being shared. On the basis of the unique ID, other users are able to retrieve the Facebook post and to join the associated session by processing the data contained in the message. Synchronization records on the other hand are communicated as comments on the Facebook post that corresponds to the involved sMS session. Session members repeatedly query the Facebook Page to acquire the most recent comment on the relevant post, parse the synchronization record it represents and apply the resulting data locally to achieve content synchronization. To prevent an explosion of the number of synchronization comments, the model instructs clients to delete all their previously published comments that pertain to a particular sMS session when posting new synchronization instructions for this session. Also remark that, by relying on a specialized Facebook Page as the communication medium, the synchronization scheme avoids pollution of the timeline of the participating users' Facebook account with synchronization messages that are meaningless to humans.

Although the described synchronization model is technically valid, it suffers from two practical issues. First, Facebook dynamically limits the volume of posts and comments that applications are allowed to publish within a certain time interval, as well as the rate at which this can occur. As these publishing constraints are enforced on-the-fly and in an opaque and largely variable manner (depending on, for instance, the current workload experienced by the Facebook back-office), they are impossible to predict. This in turn unfortunately undermines the dependability and reliability of the synchronization scheme. Secondly, considerable time (in the order of minutes) might pass before a post or comment is replicated to all the Facebook servers. As a result, session participants who are geographically vastly dispersed are likely to witness significantly variable delays before they gain access to published synchronization data.

2.5 Twitter

As is the case with Facebook, a number of APIs are available to interface third-party applications with the Twitter platform. There for instance exist RESTful API methods to post tweets. In addition, the so-called Twitter Streaming API grants developers low latency access to the global stream of tweets by allowing them to set up a long-lived HTTP connection to the Twitter

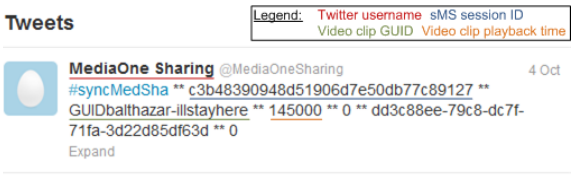

Figure 2: Synchronization via a public tweet.

back-office, over which tweets will then be streamed incrementally. In combination with extensive filtering mechanisms (hashtag-based, for example), applications in this way obtain near-real-time access to exactly the type of tweets they are interested in.

In the Twitter-mediated synchronization model, the content synchronization data is exchanged as public tweets (see Figure 2). These tweets are published on the Twitter account of the user who altered the state of the sMS session and they include a specific hashtag (#syncMedSha) as well as an ID that uniquely determines the relevant session. A PHP daemon running on a back-end webserver leverages the Streaming API to filter all the tweets that transport synchronization records, parses them, and persists the extracted synchronization instructions in a local RDBMS. Clients iteratively query this database (by issuing an AJAX request to a server-side PHP script) to retrieve the up-to-date state of the sMS session in which they are currently engaged. Note that, as opposed to the Facebook scheme described in Section 2.4, nonsensical messages (from a human perspective) are in this solution actually being introduced in the user's tweets list. To minimize the potential detrimental effects of this approach, a background client-side thread exploits the Twitter REST API to delete synchronization-related tweets that were previously issued by the local user and that have become obsolete.

Besides being less elegant than the Facebook implementation (due to the use of public and personal tweets to communicate synchronization instructions), the Twitter-powered synchronization method faces a number of constraints that are dictated by Twitter and that render the solution to be suboptimal at best. First, tweet length restrictions limit the potential payload volume of synchronization messages. Secondly, Twitter explicitly advises not to access the Streaming API from within Web browsers given their Same Origin Policy with respect to the execution of JavaScript code. As a result, a substantial financial overhead is incurred due to the necessity to deploy a server-side component (see also Section 3). To aggravate matters, unnecessary additional delay is introduced. Third, like Facebook, Twitter imposes rate limits on the usage of its APIs and enforces restrictions to ensure fairness among users and to protect the service from downtime due to overload. Contrary to Facebook however, these usage restrictions are well-documented. This implies that a strict per-day upper bound is set on the number of synchronization messages that a user will be able to disseminate. Finally, again analogous to the Facebook implementation, the publication of synchronization data tends to be slow, as the posting of tweets via the Twitter REST API is typically non-instantaneous.

3 COST ANALYSIS

Section 2 has indicated that substantial differences exist between the five solutions in terms of synchronization efficiency and performance. Now, we will take a high-level look at the financial repercussions that are associated with the adoption of each of the approaches. Commercial applications might rate cost-effective deployment and maintenance models equally important to synchronization performance.

Table 1 summarizes the economic implications due to infrastructural requirements for each of the described synchronization solutions. Three out of the five schemes can be rolled out without incurring any back-end hardware deployment cost. Of these three technologies, XMPP MUC and Facebook are further characterized by the complete absence of operational or maintenance-related expenses, which makes them particularly attractive. The same is also true for the third solution, Synchronization in the Cloud, given that the distributed groupware service does not violate the resource quota of the PaaS provider's zero-cost policy. Typically, these quota suffice for a Proof-of-Concept deployment, and often even for a small-scale commercial installation of the service.

When collating the Webserver plus AJAX and the Synchronization in the Cloud solutions, the latter will typically outperform the former in terms of financial efficiency, even if the involved synchronous application would exceed the resource quota associated with the PaaS vendor's cost-free pricing model. Many cloud platforms leverage a pay-per-use billing system, whereas traditional webserver hosting typically imposes a monolithic monthly or yearly fee. This implies that cloud-based solutions in most cases will exhibit a much more favourable cost-benefit trade-off.

Finally, although the Facebook-mediated synchronization scheme scores highly in the bird's-eye cost-benefit analysis, one must keep in mind the practical limitations that are entailed by this approach. A developer who opts to host his application on Facebook is completely dependent on Facebook's hardware infrastructure. In this context, no guarantees are given in terms of performance, as there is no notion of Ser-

Table 1: Hosting and deployment requirements and associated operational expenditure.

Technology	Necessary infrastructure and cost estimate
Webserver + AJAX	Webserver hosting (with PHP and RDBMS support) and management.
Synchronization in the Cloud	All back-end hardware is hosted and managed by the PaaS cloud provider. Actual cost depends on the cloud provider's pricing model. Many vendors (including GAE) offer free policies for applications that do not exceed predefined resource usage quota (e.g., CPU).
XMPP MUC	MUC-enabled XMPP server(s). Complimentary XMPP servers are abundantly available.
Facebook	None. Facebook's back-office takes care of storage and dissemination of sync instructions.
Twitter	Webserver hosting (with PHP and RDBMS support) and management.

vice Level Agreements (SLAs) between the developer and the platform holder. As an additional disadvantage, a software developer has no control whatsoever over the behaviour of the Facebook back-end. Section 2.4 has for instance clarified that Facebook's dynamic rate limiting policy transforms the proposed synchronization scheme into a highly unreliable and unpredictable service whose performance will be unacceptable for the majority of online synchronous applications. Note that the Twitter-powered synchronization scheme suffers from identical issues.

4 CONCLUSIONS

This paper has presented and compared a total of five concrete hosting and deployment strategies for Web-based group synchronization applications that involve the synchronous presentation and consumption of multimedia content at geographically distributed endpoints. The comparison was not limited to the technological methodology and performance of the proposed schemes, but also encompassed a non-detailed economic cost-benefit analysis. The described solutions namely all rely on a central entity that manages the inter-destination synchronization, and setting up and maintaining such centralized infrastructure might have significant financial implications. The study has revealed that a XMPP MUC-powered solution is not only highly efficient in terms of content consistency performance (courtesy of its ability to disseminate synchronization instructions in a push-based, multicast-like fashion), it is in addition gratuitously realizable given the abundance of public and complimentary XMPP MUC servers on the Internet. A cloud-hosted synchronization coordinator that utilizes the PaaS service model has been found to be a decent runner-up. With regard to technical features and synchronization efficiency, the XMPP MUC and Synchronization in the Cloud schemes are largely comparable. In terms of economic cost-benefit ratio, cloud platforms generally invoice applications only for the back-end resources that they actually consume. The pricing models of many cloud computing providers even incorporate cost-free plans, yet the resource quota bestowed upon these models are likely to be unsatisfactory when envisioning a comprehensive deployment of a real-time distributed groupware application. Finally, despite their conceptual attractiveness and theoretical feasibility, the Facebook- and Twitter-situated content synchronization techniques have been proven to be practically unsuitable. The limitations enforced by both Facebook and Twitter concerning the usage of their back-end infrastructure are so strict and time-variant that they rule out the building of a dependable, robust and adequately performing synchronization scheme.

ACKNOWLEDGEMENTS

Part of the research described in this article was performed in the context of the iMinds project 3DTV 2.0. This project is co-funded by iMinds, a research institute founded by the Flemish Government. Companies and organizations involved in the project include Androme NV and TP Vision, with financial support of IWT Flanders. We thank Davy Vanclee for the implementation of the Sync in Cloud and FB schemes.

REFERENCES

Boronat, F., Lloret, J., and García, M. (2009). Multimedia Group and Inter-stream Synchronization Techniques: A Comparative Study. *Information Systems*, 34(1):108–131.

Gutwin, C. (2001). The Effects of Network Delays on Group Work in Real-Time Groupware. In *Proceedings of ECSCW 2001*, pages 299–318, Bonn, Germany.

Johansen, R. (1988). *GroupWare: Computer Support for Business Teams*. The Free Press.

Sarvas, R. and Frohlich, D. M. (2011). *From Snapshots to Social Media - The Changing Picture of Domestic Photography*. Springer.

Stokking, H., van Deventer, M. O., Niamut, O., Walraven, F., and Mekuria, R. (2010). IPTV Inter-Destination Synchronization: A Network-Based Approach. In *Proceedings of ICIN 2010*, pages 1–6, Berlin, Germany.

Wijnants, M., Dierckx, J., Quax, P., and Lamotte, W. (2012a). synchronous MediaSharing: Social and Communal Media Consumption for Geographically Dispersed Users. In *Proceedings of MMSys 2012*, pages 107–112, Chapel Hill, NC, USA.

Wijnants, M., Lamotte, W., De Meulenaere, J., and Van den Broeck, W. (2012b). Qualitative Assessment of Contemporary Media Sharing Practices and Their Relationship to the sMS Platform. In *Proceedings of SAM 2012*, pages 31–36, Nara, Japan.