Towards Global Synchronization[1]

Ari Trachtenberg David Starobinski
(617) 358-1581 (617) 353-0202
trachten@bu.edu staro@bu.edu
Department of Electrical and Computer Engineering
Boston University, Boston, MA

Abstract

We present novel perspectives for global synchronization mechanisms in very large, heterogeneous networks. Today's state-of-the-art techniques for performing synchronization are not scalable, often requiring full data exchange between each pair of nodes in the network. We argue that future large-scale networks will not tolerate such quadratic growth in bandwidth requirements. In fact, recent information-theoretic research results suggest that the amount of communication needed for two hosts to synchronize depends only on the number of differences between the host databases, irrespective of the actual database sizes. The potential thus exists for networks of arbitrary scale to remain fully synchronized.

1 Synchronization Trends

Much of the popularity of mobile computing and handheld devices can be attributed to their ability to deliver information to users on a seamless basis. In particular, a key feature of this new computing paradigm is the ability to use applications and information on a mobile device and then to synchronize any updates back at the office or on a network. This feature plays an essential role in the vision of *pervasive computing*, where any mobile device will ultimately be able to access and synchronize with any networked data.

Thus, one could envision a user adding the address of a potential business contact to his PDA, which has been seamlessly and continuously synchronizing with any of a number of nearby available networked hosts as the user walks down a busy street. Each host, in turn, would be maintaining a dynamic synchronized body of contacts to the user's various business offices, home telephone directory, and automated car assistant. In the current state of affairs, such continuous peer-to-peer synchronization would be inordinately wasteful of both power and computational resources, but the successful extrapolation of recent advances can make such connections essentially effortless.

Nowadays, synchronization protocols are simply implemented. With a few exceptions, they generally employ a wholesale data transfer. Thus, when synchronizing two data sets A and B, these protocols typically exchange $|A|+|B|$ entries, whereas the number of differing entries, $|A-B|$, can be much smaller. Indeed, the typical case is where exactly two hosts regularly synchronize with one another, so that few changes are made between any two synchronizations.

We argue that future research should focus on the development and implementation of much more efficient synchronization schemes. Specifically, we envision a radical shift to sophisticated synchronization protocols based on techniques from computational mathematics. A first step in this direction has recently been achieved with the development of new synchronization protocols based on fast algorithms for rational function interpolation and Reed-Solomon decoding. Roughly speaking, given two hosts with data sets A and B, these new protocols can perform a synchronization using a message of length $|A-B|$, independently

[1]This manuscript was generated using LaTeX. Graphics were generated using the *xfig* program.

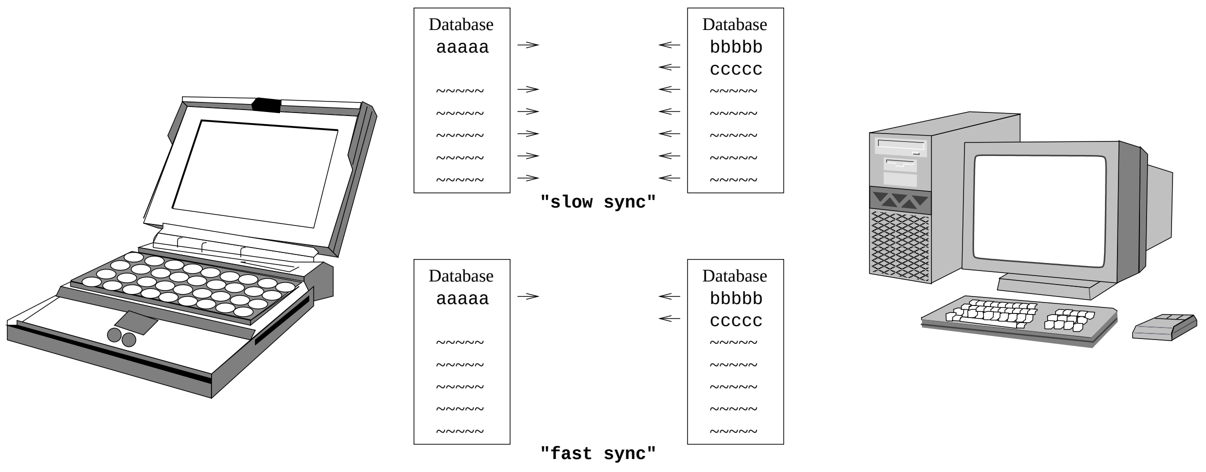


Figure 1: The two modes of the Palm HotSync protocol. In the "slow sync" all the data is transferred. In the "fast sync" only differing entries are transferred between the two databases.

of the size of the data sets A and B. Thus, two data sets could each have millions of entries, but if they differ in only ten of them, then a single message whose size is about ten entries will be sufficient to synchronize the hosts. Although, these algorithms are mathematically more involved than current synchronization techniques, we feel that their potential benefit to the growth and scalability of mobile computing platforms is tremendous.

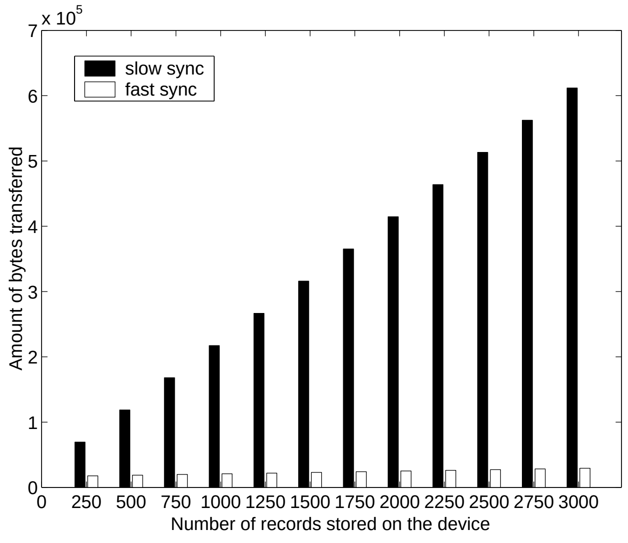
2 Scalability Issues in Data Synchronization: A Case Study

In this section, we present some of the scalability issues that current synchronization protocols do not address. Large-scale synchronization depends on our ability to solve these fundamental issues. In order to clearly and concretely explain the types of scalability problem that are considered here, we describe next how synchronization is implemented in the Palm OS architecture, one of the leading state-of-the-art mobile computing platforms.

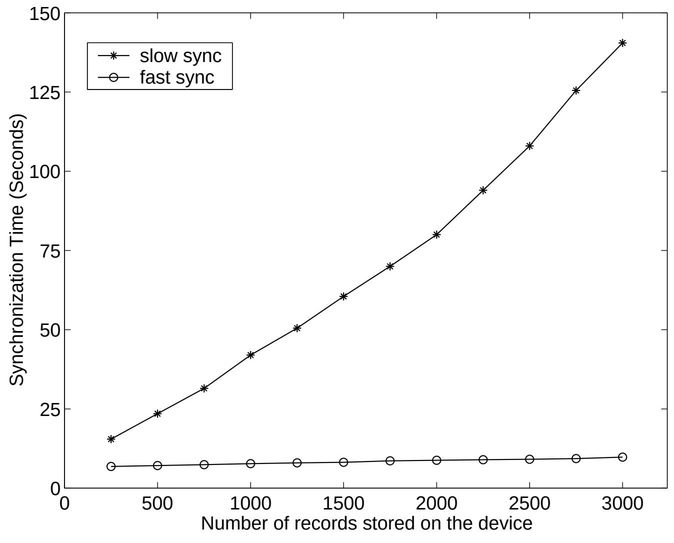
The Palm synchronization protocol, known as HotSync, relies on metadata that is maintained on both the handheld device and the desktop. The metadata consist of databases (Palm DBs) which contain information on the data records. A Palm DB is separately implemented for each application: there is one Palm DB for "Date Book" data records, another for "To Do" data records, and so forth.

The Palm HotSync protocol operates in one of two modes: *fast sync* or *slow sync*. If the PDA device synchronizes with the same desktop as it did last, then the fast sync mode is selected. In such a case, the device needs to upload to the desktop only those records whose Palm DB modification flags have been set. The desktop then uses its synchronization logic to reconcile the device's changes with its own.

If the fast sync conditions are not met, then a slow sync is performed. This happens when the handheld device synchronized last with a different desktop, as might happen if one alternates synchronization with one computer at home and another at work. In this case, the modification flags in the device are not useful and, instead, the device needs to upload *all* of its data records to the desktop. Using a backup copy, the desktop determines which data records have been added, changed or deleted. The remainder of the slow sync is identical to fast sync. An illustration of the fast sync and slow sync operation modes is given in



(a)



(b)

Figure 2: Comparison between (a) the communication complexities and (b) the time complexities of slow sync and fast sync.

Figure 2.

It turns out that slow syncs are significantly less efficient than fast syncs, especially with respect to latency and bandwidth usage. In particular, the communication cost of slow syncs increases with the number of records stored in the device, independently of the number of record modifications. Figures 2(a) and 2(b) illustrate this phenomenon on a Palm III PDA. Specifically, these figures depict the number of bytes transferred during slow sync and fast sync events as well as the duration of these events. The measurements are repeated with an increasing number of records on the device, but with a fixed number of differences (i.e. ten) between the two devices. In Figure 2(a) we see that the number of bytes transfered during slow syncs grows linearly with the number of records stored in the device, while for fast sync it remains almost constant. A similar trend can be observed in Figure 2(b), where the durations of slow syncs and fast syncs are depicted. For the case of 3000 records, the duration of slow syncs exceeds 2 minutes, about 15 times longer than fast syncs. In fact, slow sync can require as long as 20 minutes for large, but practical, database sizes.

Figure 2 clearly show that slow syncs do not scale well with the amount of information stored on a device, even when only two devices are being synchronized. New synchronization protocols have recently been proposed to address this issue [1], but their scope and methodology are inherently limited to very specific types of applications, e.g., a single centralized database shared by multiple users. In Section 3, we present promising theoretical approaches, based on recent research results, that provide a much broader perspective for tackling this issue.

In summary, the Palm synchronization model works well in simple settings where users possess a single handheld device that synchronizes most of the time with the same desktop. However, we believe that a paradigm shift is needed to allow deployment of future pervasive computing platforms, where thousands of mobile and wired devices will intercommunicate and share large databases of knowledge. In such environments, scalability concerns will require that any type of synchronizations be performed quickly and without much communication or computation.

3 A Look at Possible Future Synchronization Protocols

The example in the previous section shows that efficient synchronization is essential for the viability of applications that distribute common data among multiple hosts. In this section, we give a high-level description of the mathematical basis for advanced synchronization methods that may form the core of a large-scale synchronized protocol ten to twenty years from now.

We formalize the problem of synchronizing two hosts' data as follows: given a pair of hosts A and B, each with a set of b-bit integers (i.e. encodings of some arbitrary data), how can each host determine the union of the two sets with a minimal amount of communication—both with respect to the number of exchanges between the two hosts and with respect to the number of bits of information exchanged. Within this context, only the contents of the sets is important, but not their actual organization. In [2] this formalization is called the *set reconciliation* problem.

The simplest solution to the set reconciliation problem, and one that is often implemented in practice [3], is for both hosts to exchange hashes of their integers and then to individually request integers for the hashes they are missing. If hosts A and B have sets S_A and S_B respectively, then this scheme requires the communication of $|S_A| + |S_B|$ hashes, which is clearly not scalable.

Another plausible solution is to use timestamps or version control to aid in discovering what data elements a given host is missing. When sets S_A and S_B bear a strict subset relationship, as happens if only insertions (and not deletions) into host A's set are permitted, then timestamps can work very effectively. In order to reconcile, host B needs only request all items added to S_A since the last synchronization. The communication complexity is dependent only on the number of items differing between A and B and a modest memory overhead of one timestamp per integer is used. Unfortunately, this scheme cannot be directly applied to the case where both S_A and S_B are changing over time or where data is both inserted and deleted from S_A.

Two recent solutions to the set reconciliation problem, one based on Reed-Solomon decoding and one based on rational function interpolation [2], show promise for significant performance improvement in synchronization schemes. Both of them approach an information-theoretic lower bound of $bm - m \log m$ bits of communication needed to reconcile two sets whose contents differ in m b-bit integers. In the next subsections we give a high-level description of these two solutions.

Reed-Solomon Decoding Reed-Solomon codes are very powerful, well-known, algebraic codes that are used to correct errors in a variety of systems, including storage devices, mobile and satellite communication systems, digital television, and high-speed modems. It turns out that the decoding algorithm for Reed-Solomon codes can also be used for set reconciliation.

Specifically, suppose host A's set contains the integers $x_1, x_2, x_3, \ldots, x_a$ (represented as elements of an appropriate finite field) and host B's set contains the integers $y_1, y_2, y_3, \ldots, y_b$ and that the two host sets differ in up to m integers. Then host A can send to B the following $2m$ sums of powers:

$$X_k = \sum_{i=1}^{a} x_i^k, \quad 1 \leq k \leq 2m$$

Host B then subtracts these sums from its own power sums to get $X_i - Y_i$, in which all integers common to both hosts cancel, giving following system of equations for the differing integers δ_i:

$$X_k - Y_k = \sum_{i=1}^{2m} e_i \delta_i^k, \quad 1 \leq k \leq 2m$$

The values e_i are either $+1$ or -1 corresponding to whether the missing value δ_i (*i.e.* either x_i or y_i) is missing from host B or from host A respectively.

Since there are at most $2m$ unknowns (m differing integers, and m magnitudes e_i) and $2m$ equations, this system can be solved using well-studied classical Reed-Solomon decoding techniques. Thus, using a single message of $2m$ integers, hosts A and B can efficiently reconcile up to m differing integers. Though this result is very close to the information-theoretic lower bound, it turns out we can do even better by making use of techniques from rational function interpolation.

Interpolation-based synchronization The key to the set synchronization algorithm in [2] is a translation of sets into polynomials that can be handled more easily through a communication medium. Specifically, [2] makes use of a characteristic polynomial $\chi_S(Z)$ of a set $S = \{x_1, x_2, \ldots, x_n\}$, defined to be:

$$\begin{aligned} \chi_S(Z) &= (Z - x_1)(Z - x_2)(Z - x_3)\cdots(Z - x_n) \\ &= Z^n - \sigma_1(S)Z^{n-1} + \cdots + (-1)^n \sigma_n(S). \end{aligned} \tag{1}$$

The coefficients $\sigma_i(S)$ of the characteristic polynomial are known as the *elementary symmetric polynomials* of S.

If we define the sets of missing integers $\Delta_A = S_A - S_B$ and symmetrically $\Delta_B = S_B - S_A$, then the following equality holds

$$\frac{\chi_{S_A}(Z)}{\chi_{S_B}(Z)} = \frac{\chi_{\Delta_A}(Z)}{\chi_{\Delta_B}(Z)}$$

because all common factors cancel out. Although the degrees of $\chi_{S_A}(Z)$ and $\chi_{S_B}(Z)$ may be very large, the degrees of the numerator and denominator of the (reduced) rational function are much smaller and lend themselves to quick interpolation.

The approach in [2] may thus be reduced conceptually to three fundamental steps:

1. Hosts A and B evaluate $\chi_{S_A}(Z)$ and $\chi_{S_B}(Z)$ respectively at the same m sample points.
2. The sampled values are combined to compute the value of $\chi_{S_A}(Z)/\chi_{S_B}(Z)$ at each of the sample points. These values are interpolated to recover the coefficients of the reduced rational function $\chi_{\Delta_A}(Z)/\chi_{\Delta_B}(Z)$.
3. The zeroes of $\chi_{\Delta_A}(Z)$ and $\chi_{\Delta_B}(Z)$ are determined; they are precisely the elements of Δ_A and Δ_B respectively.

A simple implementation of this algorithm requires expected computational time cubic in the size of the sets S_A and S_B. The algorithm communicates m computed samples in order to reconcile two sets that differ by at most m integers. Another nice feature of this algorithm is that, as in the Reed-Solomon case, its communication complexity is independent of the sizes of the sets S_A and S_B. Thus, for example, hosts A and B could each have one million integers, but if their mutual difference of their sets was at most 10 then at most ten samples would have to be transmitted for both hosts to reconcile, rather than the one million integers that would be transmitted in a trivial set transfer.

4 Conclusion and Future Challenges

Though the synchronization algorithms mentioned in Section 3 are well understood, their efficient implementation in practical systems is far from obvious. This is mostly because their *prima facie* cubic computation time.

One example of implementation issues occurs with mobile systems such as PDAs and laptops that generally have limited computational abilities due to constraints on processor speed and battery life; these systems are typically synchronized with desktop machines that are not similarly constrained. Thus, though synchronizing 360 64-bit integers requires less than thirty seconds on a typical Pentium desktop [2], it can take up to 20 minutes on the 16-megahertz Palm III. This indicates that the synchronization scheme employed for mobile systems should be heavily asymmetric in its computational requirements.

We have reason to believe that efficient and practical implementation of these synchronization protocols is realizable in a large heterogenous network. The clear evidence comes from the proliferation of Reed-Solomon decoders into modern appliances, such as CD decoders and digital television. With the proper hardware advances, these decoders, whose complexity is similar to the described synchronization protocols, have become practical for small systems. One can thus envisage a "smart" gossip protocol that invisibly spreads synchronization information throughout a vast network of PDAs, laptops, and laptops using mass-produced hardware decoders.

In fact, attention to sophisticated synchronization techniques is necessary to sustain the explosive growth in network connections expected in the next ten to twenty years; the current paradigm of accounting for individual modifications will demand simply too much bandwidth and local storage to sustain such growth. Finally, the reward potential is also great, since the ubiquitous, seamless synchronization will encourage the proliferation and distribution of data, affecting both day-to-day personal interactions and global, multi-site business dealings.

References

[1] Matthew Denny and Chris Wells, "EDISON: Enhanced data interchange services over networks," May 2000, class project, UC Berkeley.

[2] Yaron Minsky, Ari Trachtenberg, and Richard Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Trans. Inf. Theory*, 2000, submitted.

[3] R.A. Golding, *Weak-Consistency Group Communication and Membership*, Ph.D. thesis, UC Santa Cruz, December 1992, Published as technical report UCSC-CRL-92-52.