Summary - On the Scalability of Data Synchronization Protocols for PDAs and Mobile Devices

Bowen Song[1]

June 25, 2018

Abstract— **This report is a part of an independent study for set and string reconciliation problems in distributed systems. The paper [1] specifically studies the scalability and performance of CPISync, a synchronization protocol based on [2], in comparison to other synchronization protocols including HotSync, Intellisync, and SyncML. The distributed system environment explored for these comparisons includes handheld PDAs, devices with stringent limitations over network bandwidth, memory, and processing power.**

I. Introduction

In a modern distributed system, applications such as file system often require data consistency across all hosts. These systems often involve handheld PDAs as a source of input. Despite the recent technological advancement in the smartphone industry – multi-core processor, expanded storage, and LTE network, handheld PDAs are still the bottleneck of synchronization protocols since the amount of data in the system increased dramatically as well. The other important factor in evaluating the effectiveness of a synchronization process upon its performance is the scalability. Most synchronization processes support pairwise reconciliation whereas a broadcasting scheme could be potentially more useful in certain situations.

II. CPISync Overview

The CPISync is based on the set reconciliation protocol in [2] and is later improved in [3] to achieve computation and communication complexity linear to the number of differences in reconciled hosts. Some of the properties besides the reduced complexities of CPISync includes the ability to broadcast and lack of need to maintain status information since the data difference assessment is affordable. Apart from the assessment in [1], the improvement in [3] further extends the scalability of CPISync. The partition tree not only accelerates the difference checking process but also enables prioritization and resumability of reconciliation. The prioritization is achievable by hashing priority information inside the key of a partition tree, and the resumability comes from partition tree allowing reconciliation process to continue in any previous status. With these improvements, especially the recursive partition scheme to bring down the computation cost, CPISync could be potentially much more efficient than the comparing result in [1].

[1]B. Song is with Department of Electrical and Computer Engineering, Boston University, Boston MA, `sbowen@bu.edu`

III. Other Existing Synchronization Protocols

The HotSync is a pair-wise data synchronization protocol that consists of a fast and a slow mode. The Fast mode is operable under prior context requirement between two hosts. Unfortunately, each host has a single Boolean sync status flag which makes the fast mode unsuitable in a multi-host system. The slow mode transfers the entire database for each pair synchronization process which is almost impractical in a modern distributed system especially when handheld PDAs are involved.

The Intellisync is a centralized system using only HotSync fast mode and allows each PDA device to always synchronize with the same server. This protocol suffers from the central point of failure and lack of scalability, at least not without investing more servers into the system. Even with the increased servers for scalability, the edge of the system, devices linking to servers far away from the central location, still suffers from synchronization latency.

The SyncML puts HotSync fast mode in a decentralized system by maintaining a record of modification information. The synchronization process is still pairwise, and every individual device on the network keeps a set of modification flags each for every device on the network. The record creates a memory overhead and is, thus, not scalable.

IV. Algorithm Performance Analysis

The Paper especially makes a comparison between CPISycn and the slow mode of HotSync to expose the time complexity difference when reconciling systems with a large number of data and an increasing set of differences. While as mentioned, in Section II that the time complexity might be reduced due to the improvements in [3], it is still not enough to be strictly faster in a system with the number of set differences equivalent to the size of the dataset itself, since the slow mode one-way scheme transfers the entire dataset and requires no computation. The CPISync would still have a threshold on the number of set differences for a better performance than HotSync.

Over the five metrics compared for scalability, the CPISync performs well in Data transmission load, Network size, Robustness, and Memory. For the transmission load, the fast mode of HostSync delivers a comparable efficiency to that of CPISync; however, this is not studied in detail to show any difference in performance other than a strength and weakness plot. The analysis on network size, memory, and robustness fails to mention the advantages of CPISync. The CPISync can broadcast a change to the entire system without known what previous state a specific device is in with the same package. The memory overhead that only depends on the number of items in the data repository makes the protocol salable and independent of the number of devices on the network. Like Intellisync, CPISync is a decentralized protocol which is not prone to the effect of central point of failure.

The comment on computation cost is only partially valid with the improvements in CPISync. The computation complexity is reduced to linear to the number of difference between reconciled sets whereas the slow mode in HotSync does no computation in a one-way synchronization process, which is the equivalence of first initializing data in a new device. However, in a two-way scheme or fast mode where hosts need to sort out set differences between received data and local repositories, the computation complexity would be comparable.

V. System Scenarios

The Paper sets multiple scenarios that are best for each mentioned protocols. However, we believe CPISync can fit into all of them with the improvements in [3]. In the scenario where HotSync fast mode is applicable, CPISync has partition tree to match the fast mode time cost. In a centralized architecture, the SyncML has the memory overhead linear to the number of devices on the network, while the CPISync has the memory overhead linear to the number of items for synchronization which makes CPISync scalable in a large decentralized network.

VI. Conclusion

The paper has listed some key advantage of CPISync comparing with the three mentioned protocols. The improved CPISync would seemingly perform either comparable or better than all of them. Unfortunately, there is no comparison between the fast mode of HotSync and CPISync to give a concrete conclusion. The key difference for analysis between Fast mode of HotSync and CPISync is the HotSync's status flag and CPISync's on the go evaluation. The actual reconciliation process for Hotsync fast mode is not available and, therefore, it is not enough to compare the two protocols.

References

[1] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the scalability of data synchronization protocols for pdas and mobile devices," *IEEE network*, vol. 16, no. 4, pp. 22–28, 2002.

[2] Y. Minsky, A. Trachtenberg, and R. Zippel, "Set reconciliation with nearly optimal communication complexity," *IEEE Transactions on Information Theory*, vol. 49, no. 9, pp. 2213–2218, Sept 2003.

[3] Y. Minsky and A. Trachtenberg, "Practical set reconciliation," in *40th Annual Allerton Conference on Communication, Control, and Computing*, vol. 248, 2002.