A Method for Data Consistency Support in Mobile Ad hoc Distributed Systems[♣]

© Maxim Galkin

University of Saint-Petersburg
Universitetsky pr. 28, Peterhof, Saint-Petersburg, Russia
maksim.galkin@gmail.com

Abstract

This paper presents a research-in-progress report on a method for data consistency support in ad-hoc mobile distributed systems, based on the concept of high-level operations "compatibility" and operation history reconciliation. The proposed method also utilizes tuple spaces as a communication model and accumulators as data structures for efficient conflict resolution.

1 Introduction

In this paper we present a method for data consistency support in an ad-hoc mobile distributed system. This method combines some of the practices known for "nomadic" distributed systems with loose data structures such as tuple spaces that can serve as a medium hiding the intermittent mobile network connection.

Why do we consider methods for consistency support in ad-hoc distributed system at all? One of the advantages of such system is in the absence of a "single point of failure". That is, all of the network nodes are equally important, and if we keep their replicas consistent then our system can continue working even if it loses most of the nodes. Of course, that also creates more requirements for the reconciliation protocol, as we need to "handle" nodes, which were "offline" for a period of time.

Another advantage is the potentially unlimited extensibility as there is no bottleneck like main server performance or main server connection bandwidth. A new node can join the network at any time, locate its "neighbors" and obtain the latest available data.

Finally, in a mobile world it's a possible scenario that the known main servers are inaccessible, while the "neighbor devices" can be still available. The proposed protocol can serve as a failover tool in such scenario.

The advantage of the proposed method is in its indifference to an underlying data model. It demands all nodes to be aware of some initial data state (which can possibly be empty, it is only required as a common "starting point" for nodes operation history) and of a set of high-level operations. After that is defined, the nodes become autonomous and start applying those operations to their local replicas. At some points in time nodes perform reconciliations of their replicas according to the proposed protocol and through this achieve a consistent and actual data. Due to the general approach this method can be used to build a middleware platform for consistency support in ad-hoc distributed systems.

2 Data Model and Operations

For the sake of the proposed method, a set of defined high-level operations over the data is more significant than the data model itself and further in this paper we will only discuss operation set properties. At the same time we must note that in some related works [1, 2] special data model was crucial in defining such operations, which caused less conflicts and therefore were better suitable for collaborative work.

In general, we can name two sides or two states of the data in the system. One side is the real data, which resides in the mobile agents, and which is continuously updated by them and another side is the ideal data, that can be achieved by a full semantically-correct reconciliation of all data replicas. In the process of work our cloud of local data replicas strives to become closer to the ideal state, where every agent knows the current and actual state of data. Of course, in reality the degree of consistency depends on many factors, like the connection quality between agents and the intensity of the continuous local updates.

In most cases, a table of compatibility conditions is filled together with operations definition. We call two operations "compatible" if none of them depends on another. In the CoACT model [3] such conditions are called "activity interleaving rules" and given in a form of predicates for each pair of operations. In general, operations compatibility may depend on both their nature (e.g., two "read" operations will never create a conflict) and data they are applied to.

[♣] Proceedings of the Spring Young Researcher's Colloquium On Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, 2009.

This work is partially supported by Russian Foundation for Basic Research under grant 07-07-00268.

To successfully maintain consistency the set of operations for ad-hoc mobile distributed system must ensure as few conflicts as possible, in other words operations must be as compatible as possible, because every conflict in such a system is a serious problem: two mobile nodes, which discover a data inconsistency during their interaction don't possess any advantage over each other, unlike it happens in "nomadic" distributed systems, where the fixed nodes have the priority, the most consistent state. Usually in ad-hoc distributed system it's also impossible to draw a human operator into the process of conflict resolution, because the inconsistent state can be discovered at nodes that aren't the authors of the conflicting operations.

Therefore every conflict resolution must be automatic in accordance with some pre-defined rules of consistency. Such rules can be of two major types: *differential* (e.g. when the system takes one of two conflicting operations basing on its timestamp) and *integral* (if the system, for example, has some pre-defined objective function or cost function over the data model and it discards one of the conflicting operations basing on the value of the function).

3 Transactional Properties of the System

The above-stated implies that even if some operation has been committed at one of the nodes and lived in the system for some period of time it is not guaranteed to be fixed in the system forever as one of the other nodes could have submitted a conflicting operation that will eventually overcome the first one. In other words, the Durability property from the ACID set is not guaranteed by our system for the sake of Consistency. Here we can also notice that in ad-hoc distributed system no "global commit" or "strong" [4] operations are possible. On the contrary, all operations in the system are "weak", that is they are only applicable to the local replica of the data and they are only guaranteed to be "safe" until next data reconciliation with another node.

With respect to the other ACID properties such as Atomicity and Isolation we should notice two levels of their applicability. If we consider our high-level operations as a specific form of transactions consisting of some elementary read/write sub-operations, then these transactions will satisfy both A and I properties. If on the other hand we introduce even higher-level transactions that consist of our original operations we will need to give away the highlighted properties, similar to how it happens in the "sagas" transaction model [5]. Otherwise, long-running isolated transactions will inevitably increase the number of conflicts just like it is observed in other types of distributed database systems [6]. Such an increase in our system can lead to mobile nodes overload and overall decline in data quality, as compensative transactions can sometimes be ineffective in cleaning up the database due to lack of transaction isolation.

4 The Proposed Protocol

We assume that numerous mobile nodes interact with each other by establishing connection with some of the other nodes in their "visibility range" and perform reconciliation of their respective data replicas. During the reconciliation nodes compare not the data itself, but the history of operations made over some initial data state: a common "starting point", which is known to all nodes. If a conflict is discovered it is resolved according to the rules for the given operations, one of the conflicting operations has to be compensated and marked in history as rejected. This mark can make future reconciliations faster.

After the reconciliation we need to recalculate all the static data on the nodes that depend on the changed operations. If the changes have only affected such structures as accumulators [1] they don't need the recalculation.

As a communication model between the mobile nodes we can use the tuple spaces mechanism, such as, for example, LIME [7] or JavaSpaces [8]. In this case nodes can create dynamic groups, sharing one tuple space. In this tuple space a consistent set of operations will be stored for each data element. If a new node comes to the group, or a new operation is performed on a present node it triggers the reconciliation process as described above. All nodes also synchronize their local replicas with the common space to be ready to go offline any time.

In this scenario the common tuple space can be found to correspond to a notion of a "cluster of consistency" [4], with the only difference that in our case no data in a cluster is durable. For example, if two nodes from different clusters interact any of the operations in their replicas can be potentially conflicting and subject for removal.

5 Conclusion

We have introduced a method for consistency support in a mobile ad-hoc distributed system, based on the reconciliation process of high-level operations histories. We also presented a description of supported data models and the requirements for operations set. We have analyzed the transactional properties of the proposed system and outlined a protocol for nodes interaction.

We plan to further develop this method with regards to an experimental proof of concept. We also plan to introduce metrics of the global consistency in such a mobile system to be able to evaluate the efficiency of the method and/or compare our method with other methods. Another way of applying such metrics is to use them in the described mobile system as an integral rule for operational histories reconciliation, thus we can sort out nodes, which have their replica so obsolete or inconsistent that it doesn't make sense to compare them with the current data.

References

[1] A. Kozlova, D. Kochnev, B. Novikov. The Middleware Support for Consistency in Distributed Mobile Applications. Proc. of the Baltic DB&IS'2004, 145-160, Riga, Latvia, Scientific Papers University of Latvia, June 2004.

[2] O. Proskurnin, B. Novikov. Towards collaborative video authoring. Proc. of ADBIS'03, 370-384, Dresden, Germany, 2003.

[3] M. Rusinkiewicz, W. Klas, T. Tesch, J. Wasch, P. Muth. Towards a Cooperative Transaction Model: The Cooperative Activity Model. Proc. of the 21st VLDB Conference, Zurich, Switzerland, 1995.

[4] E.Pitoura, B. Bhargava, O.Wolfson. Data Consistency in Intermittently Connected Distributed Systems. In Transactions on Knowledge and Data Engineering, Nov 1999.

[5] H. Garcia-Molina, K. Salem. Sagas. ACM SIGMOD Record, Volume 16, Issue 3, pp. 249-259, 1987.

[6] J. N. Gray, P. Helland, P. O'Neil, D. Shasha. The Dangers of Replication and a Solution. In Conference on Management of Data, pp. 173-182, Canada, June 1996.

[7] A. L. Murphy, G. P. Picco, G-C. Roman. LIME: A Coordination Model and Middleware Supporting Mobility of Hosts and Agents. ACM Transactions on Software Engineering, Vol. X, No. X, pp. 1-48, 2006.

[8] E. Freeman, S. Hupfer, K. Arnold. JavaSpaces, Principles, Patterns and Practice. Addison-Wesley, 1999.