Towards Handling Constraint Network Conditions Between WoT Entities Using Conflict-Free Anti-Entropy Communication

Markus Ast(✉) and Martin Gaedke

Technische Universität Chemnitz, Chemnitz, Germany
{markus.ast,martin.gaedke}@informatik.tu-chemnitz.de

Abstract. Deploying and composing Web of Things entities in scenarios where connections are subject to network constraints, like disconnected operations, intermittent connections or limited bandwidth, requires handling changing network conditions properly. Therefore, this work proposes to utilize both eventual consistent data structures and corresponding eventual consistent communication for such scenarios. This enables composition and collaboration of WoT entities in network constraint scenarios.

Keywords: Web of Things · Distributed systems · Network constraints

1 Introduction

The emerging field and availability of Web of Things (WoT), results in the need for making such things, e.g. devices, sensors, etc., available for all kinds of scenarios. When composing WoT entities together, one challenge that arises, are network constraints that disrupt communication with or between them. Such network constraints can be (1) disconnected operations so that there is no connection at all for a long period of time, (2) intermittent connections where recurring loses of connection happen at irregular intervals, and (3) limited bandwidth in terms of having a slow connection.

Taking these network constraints into account, connections between WoT entities could be considered irregular and loosely. To still allow for composition and therefore for collaboration of WoT entities, this work provides an experimental approach that ensures eventual communication of WoT entities using conflict-free replicated data types that are distributed using anti-entropy communication.

This work is thereby about all scenarios of WoT entities collaborating together under changing network conditions. When being connected wirelessly, network constraints can be due to natural reasons, e.g. mountainous regions where changes in weather conditions easily affect network quality, or due to catastrophic events. Furthermore, entities like sensors can also be deployed out of reach and therefore rely on being connected manually, e.g. research sensors inside caves.

© Springer International Publishing Switzerland 2016
A. Bozzon et al. (Eds.): ICWE 2016, LNCS 9671, pp. 576–580, 2016.
DOI: 10.1007/978-3-319-38791-8_52

The rest of this paper is organized as follows: at first, in Sect. 2, an analysis is conducted and used technologies is introduced accordingly. Afterwards, an overview of the approach is given in Sect. 3. Finally, related work is summarized in Sect. 4, and a conclusion is given in Sect. 5.

2 Analysis

Let $V = \{e_0, e_1, ...\}$ be a set of composed WoT entities. Each entity has a set of peers P, where $P \subseteq V$. Consider these entities as a graph $G = (V, E)$, where E is the set of edges that indicate an available connection for communication between two entities. The resulting graph is neither guaranteed to be connected completely nor guaranteed to be connected at all. However, the graph is being considered to be connected eventually, since it is only disconnected because one or more of its connections are not available for a finite amount of time. Consequentially, the graph is only disconnected for a finite amount of time, too.

According to the CAP theorem [1], a distributed system can only have two of the following three properties at the same time: consistency, availability and partition-tolerance. In the described scenario, WoT entities have to work well, despite possible partitions of the network. Since partitions are not guaranteed to be resolved within a reasonable short amount of time and since achieving consistency requires communication, pursuing the property of consistency would lead to unresponsive WoT entities. This would in return lead to bad user experience or even stagnate important functionalities. That is, according to the CAP theorem, trade-offs will be made in consistency instead of in availability.

With achieving both availability and partition-tolerance the systems ends up being *eventual consistent*. *Eventual consistency* is "simply an acknowledgement that there is an unbounded delay in propagating a change made on one machine to all the other copies" [4]. That is, each operation will be eventually be applied to all entities. This requires a total global order using something like vector clocks [3]. However, changes between entities are potentially communicated in different order—compared to their initial occurrence. Therefore, the property of *eventual consistency* can be extended with the even stronger condition of *strong convergence*: "correct replicas that have delivered the same updates have equivalent state" [7]. With this so called strong *eventual consistency*, there are no roll backs necessary to resolve conflicts with updates in different orders.

Strong eventual consistency is provided by using *Conflict-free Replicated Data Types* (CRDT). CRDTs are data types "for which some simple mathematical properties ensure eventual consistency" [7]. A simple example for such a CRDT would be a replicated counter, "which converges because the increment and decrement operations commute (assuming no overflow)" [6]. With CRDT, updates do not require synchronization, they can rather be executed immediately when being received. They are both scalable and fault-tolerant and most data structures are reasonable easy to implement.

A way of replicating state with weak consistency requirements, are anti-entropy protocols - or also called gossip protocols. Anti-entropy protocols work,

simply put, as follows. An entity $p \in V$ periodically selects a peer $q \in P_p$. Then, p either pushes its state to q or sends a digest (only keys and version numbers of its data) to q to retrieve only the necessary updates from q afterwards. To not create a huge backlog of unsynchronized pending state updates and to keep the network usage minimal to tackle the constraint of limited bandwidth, the *Scuttlebutt Reconciliation* [5] is used.

For providing each entity with information about the current network state, a failure detector is required. Treating failure information as a state that is synchronized with all other entities allows for providing each entity with information of the overall network. This allows entities to know about the state of other entities even though they may not have a direct connection to them on their own.

Traditionally, failure detection is boolean only: a node is either available or not. However, this does not work well with changing network conditions. That is, an adaptive failure detector that allows for adapting its predictions according to changing network conditions is necessary. This allows each entity to make its own predictions about the network, taking its entity-specific requirements and constraints into account. To provide such an adaptive failure detector, that provides a scale of confidence rather than a boolean value, the *Φ Accrual Failure Detector* [2] is used. This failure detector is easily integrated into the gossip communication. Each state replication reports its timestamp to the *Accrual Failure Detector*, which keeps a sampling window of previous reports to adapt its estimation for the next report accordingly.

3 Approach

Resulting from the analysis, our approach for tackling network constraint WoT entities combines conflict-free replicated data types with gossip protocol communication and integrates the *Accrual Failure Detector* for failure detection. This yields an eventual consistent system and therefore allows for being very tolerant to changing network conditions. This enables deployment of WoT entities in network constraint scenarios.

The interaction between these introduced components works as follows. Each entity acts as a replicated state machine. States are based on CRDT data structures. Periodically, each node selects another peer in the network randomly. This selection, however, favors peers that appear to be accessible above peers that appear to be inaccessible. The entity's knowledge about the accessibility is provided by employing the *Accrual Failure Detector*. Each incoming request or response from a peer is reported to the failure detector to maintain its sampling window and make predictions about its accessibility accordingly. Additionally, reports from other entities and the accessibility to other network participants from their point of view are integrated as well.

Upon peer selection, state between both connected entities is synchronized using *Scuttlebutt Reconciliation*. Doing so, and employing proper flow control, allows for efficient usage of network bandwidth. That is, the communication scales for both high and low bandwidth availability. The usage of the *Accrual Failure Detector* thereby adds the possibility to give each entity its own local view of the overall network. Having a probability scale of each entities connectivity state, allows each entity to adapt its own response or functionality accordingly, e.g., notify on the entity working users immediately about connection changes, or reduce gossip rate according to the received prediction.

4 Related Work

Replicating state in distributed systems is often done using consensus algorithms. That is, data types like CRDTs are not required. However, on each node disconnect, the selection of a new master may be required and synchronization during network partitions can lead to conflicts.

With Riak 2.0, CRDT found their way into distributed databases recently[1]. Riak also uses a gossip protocol to communicate ring state and bucket properties across its cluster. However, these are only parts of Riak and Riak itself is not targeted towards the described scenarios.

That is, the application of CRDTs and gossip protocol communication for WoT entities is not widely experimented with, yet. Since it should apply to minimal hardware, like sensors, too, it also yields new challenges for CRDT and gossip protocol approaches, like partial state synchronization.

5 Conclusion

This paper introduced an experimental approach for handling constraint network conditions between WoT entities by employing CRDT data types that are replicated using gossip protocol communication. With this approach, connected WoT entities are less prone to network failures or constraint network conditions.

While first experiments yielded promising results, there is still a huge potential for succeeding experiments and other future work, like allowing for partial state replication. Partial state replication will be a crucial part, because simple sensors to not need to have the whole state of other entities locally. For them, it is probably enough to replicate their own state and state related to remote configurations.

Acknowledgement. The authors gratefully acknowledge funding by the DFG (GRK 1780/1).

[1] https://docs.basho.com/riak/2.0.1/dev/using/data-types/.

References

1. Gilbert, S., Lynch, N.A.: Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. SIGACT News **33**(2), 51–59 (2002)
2. Hayashibara, N., Défago, X., Yared, R., Katayama, T.: The Φ accrual failure detector. In: Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems, pp. 66–78. IEEE (2004)
3. Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. ACM **21**(7), 558–565 (1978)
4. Oracle: De-mystifying "eventual consistency" in distributed systems, June 2012
5. van Renesse, R., Dumitriu, D., Gough, V., Thomas, C.: Efficient Reconciliation and Flow Control for Anti-Entropy Protocols (2008)
6. Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: A comprehensive study of Convergent and Commutative Replicated Data Types, p. 50, January 2011
7. Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: Conflict-free Replicated Data Types, July 2011