# Highly Dependable Location Tracker
Stage 2 Report — Group 22

André Breda, 89409 andrebreda@tecnico.ulisboa.pt
Baltasar Dinis, 89416 baltasar.dinis@tecnico.ulisboa.pt
João Borges, 89482 joao.p.l.borges@tecnico.ulisboa.pt

# 1 System Design

## 1.1 System Model and Assumptions

The system has three components: a set of servers, $S = \{s_i\}$, which verify location proofs, cross-reference positions and reply to location queries; a set of users, $P = \{p_i\}$, which try to prove their location; and a health authority, $HA$. Users will communicate with nearby users (in practice, using Bluetooth) to request proofs about its own location at a given epoch. A user will certify (witness) another user's location if that location is within its proximity, providing a report to that effect. Note that a user can only communicate with users in its proximity. We now enumerate a series of assumptions about this model.

**Assumption 1.** *At most $f_S$ servers do not follow the protocol correctly (ie: are Byzantine).*

**Definition 1** (Neighbourhood)**.** *Set of users in another user's proximity.*

A user is in its own neighbourhood. $\mathcal{N}_i$ denotes user $i$'s neighbourhood.

**Assumption 2.** *Neighbouring is commutative*

$$i \in \mathcal{N}_j \Leftrightarrow j \in \mathcal{N}_i$$

**Assumption 3.** *A user can communicate with all (and only with) users in its neighbourhood.*

Some users are considered to be Byzantine [7]. These users are assumed to be *omnipresent*, meaning they can appear in any neighbourhood simultaneously, no matter how far. In practice, this can be executed in several ways (eg: using Bluetooth beacons to extend their signal, having a set of distinct devices masquerading as the same device). Note that this is the strongest type of Byzantine user possible for the location context.
However, there are limits to the number of these faults.

**Assumption 4.** *At most $f_P$ users are Byzantine.*

**Assumption 5.** *In a given neighbourhood, at most $f' < f_P$ are Byzantine.*

## 1.2 Threat Analysis

There are several generic threats to consider. We present these in Table 1, alongside their mitigations. Note that person in the middle attacks are also prevented by signing relevant messages (or using a previously established symmetric key).

## 1.3 Replication

So as to address possible unavailability of servers or the possibility of Byzantine servers, we replicate the set of servers. We then employ two variants of a single writer Byzantine register protocol. One of the variants enforces regular semantics and the other atomic semantics. We followed, without relevant adjustments, the protocol described in [1] (Modules 4.6 and 4.7). One interesting observation is that, in this context, the timestamps can be considered the epoch numbers, since a correct user will only submit one location proof per epoch.

# 2 Guarantees

## 2.1 Integrity

**Theorem 1.** *Proof requests (and witness reports) are authenticated and non-repudiable.*

This is guaranteed because all requests (and reports) are signed by the prover (or witness).

**Theorem 2.** *In each epoch, a Byzantine user cannot pretend to be in two places.*

This is guaranteed by requiring the position in each proof request and witness report. If a user provides conflicting positions cross-referencing with previously exchanged messages will produce a proof of misbehaviour. This property is equivalent to non-equivocation [3].
We also observe that since Byzantine users are omnipresent, the notion of *false location proof* does not exist, and Theorem 2 is the next strongest property possible.

**Lemma 1.** *A valid location proof requires at least $f' + 1$ matching reports.*

This is obviously true, because it ensures that at least one correct user has verified the location of the user.

**Corollary 1.** *A user must receive replies from at least $f'$ witnesses to construct a valid location proof.*

Note that the user also self-attests its own location. If it is correct, that location will be correct. Otherwise, one of the $f'$ witnesses will be correct (Assumption 5) and will match its report.
Regarding of data integrity upon spurious crashes, we employ SQLite [6] as our persistence back-end, which guarantees atomicity of data updates.

| Threat | Mitigation |
|---|---|
| Message Forgery | Sensitive messages are signed for authentication or ciphered using a shared secret symmetric key between a user and the server |
| Tampering | Messages carry integrity information (MAC) |
| Replay | Requests carry cookies |
| Non-repudiation | Messages are signed |
| Equivocation [2] | The servers store all position proofs and keep a record of malicious users |
| Denial of Service | SHA-256 based Proof of Work (PoW) needs to be calculated to submit a Proof |

Table 1: Threats to the system and respective mitigations

### 2.2 Liveness

We wish to establish the conditions for which following property:

**Property 1.** *Eventually, every correct user can have their location accepted by the system.*

As a direct consequence of Corollary 1, it is obvious that a correct user will produce a valid proof if eventually it will have a neighbourhood where it can get at least $f'$ replies.

### 2.3 Confidentiality

We ensure confidentiality in communications from the users and the server, as well as with all health authority communications. This is required to keep the locations of the various users secret from a network adversary.

## 3 Expunging Bad Actors

Whenever a user equivocates, this produces a *proof of misbehaviour*: the two messages with conflicting positions in the same epoch. This can be used to expunge nodes from the system: the system will start ignoring information from these users. More interestingly, we can use this to reduce the fault parameters. Every time a malicious node is discovered, $f$ is decremented. $f'$ is also decremented if $f = f'$. Moreover, we can also decrement $f'$ if more than $\left\lceil \frac{|P|}{f'+1} \right\rceil$ are proved to be Byzantine, as this is enough to ensure that there is at least one provably Byzantine user in each subset with at least $f' + 1$ users (minimum required for a proof). Interestingly, if all malicious users misbehave, eventually $f = 0$ and users can just self-attest their location. This approach is also key in our strategy to defend against clients failing to implement the replication protocol correctly. There are two threats a Byzantine client can pose. The first is to the liveness of the protocol: if it does not send the messages to all (or at least the required) servers, the health authority — the only other entity that could read its location — never will (at least atomically, with regular semantics it might). However, since this behaviour is indistinguishable from a client crashing in the middle of a write (and since that does not violate neither the required atomic and regular semantics of the operations) this is not an actual attack on the system. The second threat is tied to equivocation. Here, the next correct client to read its values will see contradicting position reports, and will then produce the misbehaviour proof and send it to all servers (note that if all clients reading the value are Byzantine, the protocol need not provide any guarantees). As such, we can avoid expensive machinery to solve non-equivocation, such as a Byzantine reliable broadcast [1]. We also observe that this approach is similar to that taken by Druschel et al on Accountability in distributed systems [4, 5], in that by taking into account all the history of the system (in this case, all the position proofs), we can prove misbehaviours and use it to prune the system of bad actors.

## References


[1] Christian Cachin, Rachid Guerraoui, and Lus Rodrigues. *Introduction to Reliable and Secure Distributed Programming*. 2nd. Springer Publishing Company, Incorporated, 2011. ISBN: 3642152597.

[2] Allen Clement et al. “On the (Limited) Power of Non-Equivocation”. In: *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*. PODC ’12. Madeira, Portugal: Association for Computing Machinery, 2012, pp. 301–308. ISBN: 9781450314503. DOI: 10.1145/2332432.2332490. URL: https://doi.org/10.1145/2332432.2332490.

[3] Allen Clement et al. “On the (Limited) Power of Non-Equivocation”. In: *Proceedings of the 2012 ACM Symposium on Principles of Distributed Computing*. PODC ’12. Madeira, Portugal: Association for Computing Machinery, 2012, pp. 301–308. ISBN: 9781450314503. DOI: 10.1145/2332432.2332490. URL: https://doi.org/10.1145/2332432.2332490.

[4] Andreas Haeberlen, Petr Kuznetsov, and Peter Druschel. “PeerReview”. In: vol. 41. Oct. 2007, p. 175. ISBN: 9781595935915. DOI: 10.1145/1294261.1294279.

[5] Andreas Haeberlen et al. “Accountable Virtual Machines”. In: Oct. 2010, pp. 119–134.

[6] Richard D Hipp. *SQLite*. Version 3.31.1. 2020. URL: https://www.sqlite.org/index.html.

[7] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* 4.3 (July 1982), pp. 382–401. ISSN: 0164-0925. DOI: 10.1145/357172.357176. URL: https://doi.org/10.1145/357172.357176.