
Sidestepping Impossibility: Combat Consensus in the Assassins' Guild

Jacob Beal

JAKEBEAL@MIT.EDU

MIT Computer Science and Artificial Intelligence Laboratory, 32 Vassar Street, Cambridge MA, 02139 USA

The MIT Assassins' Guild simulates ranged combat in a way that appears to violate a consensus impossibility result. In fact, of course, it does not violate the impossibility result, but sidesteps it by controlling where consensus failures occur so that the execution is indistinguishable from some consistent execution.

The failure management employed by the algorithm depends on three things: asymmetry, locality, and uncertainty. Distributed computing problems with these three properties should be able to adapt the ideas of Guild ranged combat to solve consensus problems quickly, cheaply, and reliably.

1. LARP as Distributed Computing

In a roleplaying game, each player takes on the persona of a character in the imaginary world of the game. In live-action roleplaying (LARP), the game world is projected into the real world and some types of game action are physically acted out by the players.

MostLARPs, especially those run by the MIT Assassins' Guild, involve simulations whose execution is distributed across many different players to allow faster execution and aid suspension of disbelief. The economy of the game world, for example, may be simulated by players exchanging cards representing quantities of trade goods.

A group of players carrying out a distributed simulation is equivalent to a network of devices executing a distributed algorithm: each player is a device, and two players capable of interacting are connected by a link in the network. This network is a challenging environment for distributed algorithms:

- Small computational capacity: although players are intelligent, faithful execution of rules is slow, no more than a few operations per second.
- Small working memory: players can remember only a few digits accurately, plus a somewhat larger amount of context-cued rules. Much more can be stored unreliably, or reliably on high-latency storage (e.g. paper).
- Slow communication: information exchange between players is limited by the rate at which humans act and interpret sensory information.

- Partially synchronous execution: even with watches, it is futile to try to synchronize tighter than a few seconds.
- Changing network topology: as players move through physical space, their ability to interact changes.
- Poor network connectivity: it is rare for the network of a game to *not* be partitioned if there is any significant amount of space available to players. Cliques of players wishing to act privately regularly move out of contact with the rest of the game, partitioning the network.
- Byzantine failure: honest mistakes and confusion about the rules can cause arbitrary failures in communication and computation. These failures are most likely to occur when they matter most because the players are awash in adrenaline during critical moments.
- Differing incentives: Although cheating is virtually nonexistent due to social pressure, players happily abuse one another within "the spirit of the rules."

2. Liveness & Consistency in Ranged Combat

Combat between characters is usually the most time-sensitive simulation in a game. A group of players fighting must be able to establish consensus on the sequence of operations carried out, despite the fact that the battle may include many people spread over a large area of space. This is not a simple problem. A combat-intensive game can easily develop a battle involving dozens of people spread through several buildings and lasting for many minutes. Worse yet, the use of semi-automatic and fully-automatic toy weapons means a single player can easily be involved in the resolution of a dozen or more concurrent operations, and healing creates non-monotonicity.

At the same time, resolution of each individual operation must be rapid and wait-free: otherwise a player can end up "stuck" resolving one action while other players act to put her at a disadvantage.

This is a classic distributed computing problem of providing liveness and consistency, and stress-induced player errors mean that we cannot guarantee both. Impairing liveness leads to "stuck" players being unfairly disadvantaged because of the failure of an opponent. Impairing consistency leads to disputes between combatants about

what happened, which may also impair liveness! Finally, because the network may be partitioned, it is theoretically impossible to guarantee both liveness and consistency. (Gilbert & Lynch, 2002)

Paradoxically, Guild designers have settled on a set of rules for ranged combat (Morris, 2004) which guarantees neither liveness nor consistency. Instead, it delivers consistently satisfactory results by managing the conditions under which failures can occur.

The rules are simple: attacks are represented by projectiles from a toy gun or harmless thrown objects such as ping-pong balls. The first object the projectile contacts is hit by the attack: if it hits a player, the player hit secretly calculates its effect based on projectile type. Shots that hit long hair or loose clothing don't hit; shots that hit an item the player is carrying hit the player. In case of question, attacker calls the shot; in case of a serious dispute, all action is halted while things are sorted out.

Considered from the perspective of distributed computing, it is surprising that these rules work at all, since there is barely any attempt to assure liveness or consistency. Yet they do work: disputes of any sort are rare, and disputes that interfere with the game (e.g. by forcing a halt) are essentially non-existent.

To understand this surprising success, we must carefully examine how inconsistency leads to the adverse consequence of disputes between players. We start with the empirical observation that disputes arise when *a player believes the outcome of combat is affected by an opponent's unfair advantage*.

First, notice that if the target takes a hit which the attacker believes was a miss, that is an unfair disadvantage and will not lead to a dispute. Second, a dispute arises not on the basis of fact, but on the basis of belief, so only inconsistencies detectable by the players can lead to disputes. Finally, disputes are about outcomes, not individual projectiles: individual projectiles only matter because of their cumulative effect in causing a player's defeat.

Now we can start to understand why the ranged combat rules work. The embodiment of attacks as physical projectiles restricts the consensus problem to the question of where each projectile first hits, and we can consider only two player combats without loss of generality. Most shots are clean misses or clean hits, and the rules about hair, loose clothing, and items means that the visual perception of the attacker is likely to match the tactile perception of the target. Moreover, the attacker can often see the target notice a shot impacting. Thus, disagreement is already infrequent.

There are three ways that inconsistency can occur: the attacker thinks a projectile missed and the target thinks it hit,

the attacker thinks it hit and the target thinks it missed, or both think it hit but the target miscalculates the effect. In the first case, the target is placed at an unfair *disadvantage*, and so no dispute will arise. The other cases are dealt with by the attacker calling the shot: because the attacker calls the shot, a few quick phrases can establish the belief that the attacker and target agree on whether the shot hit. Usually this leads to consistency, but errors in communication and calculation can still lead to inconsistency. For this to lead to a dispute the inconsistency must be detectable by the attacker, but since it is only observed as a cumulative effect in the defeat of the target, and since the attacker generally has some uncertainty about the target's resources, a small rate of inconsistencies is undetectable.

Without disputes, the threats to liveness are never realized, and thus the Guild's ranged combat produces results equivalent to those of a consensus mechanism which guarantees liveness and consistency.

3. Abstraction from Analysis

This analysis hangs critically on three key properties:

- **Asymmetry:** Consensus can fail in multiple ways, and one type of failure is much less important than the other.
- **Locality:** The critical computation is done on a single device, so the subject of consensus is the computation to be performed, not the result of the computation.
- **Uncertainty:** The result of a computation can be any one of several answers, and some of the information necessary to determine the answer is contained only in the device performing the computation.

While unusual, these properties are hardly specialized to LARP environments. If distributed computing problems that share these properties can be identified, then an adaptation of the Guild ranged combat algorithm should be able to provide cheap and reliable consensus.

More generally, this suggests an interesting method for dealing with impossibility results. The Guild ranged combat algorithm recognizes that not all failures are created equal, and leverages that to mitigate their impact—in this case, entirely. Perhaps there are other areas where a similar approach will yield powerful results.

References

- Gilbert, S., & Lynch, N. (2002). Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Sigact News*, 33.
- Morris, J. (Ed.). (2004). *Standard rules for mit assassin's guild games (2004-2005 edition)*. MIT Assassin's Guild.