# Raftian: Pygame Meets Raft

TULLIO VARDANEGA* and MARCO BELLÒ*, University of Padua, Italy

The problem of shared consensus in a distributed system is both older than a millennium and more relevant than ever: while in the Aegean island of Paxos the challenge was to keep track of the many laws passed in a parliament where legislators had better to do than attend sessions full-time, nowadays the ubiquity of web-based architectures and applications (from a simple cloud storage to the whole banking system) gives daily headaches to developers and system administrators alike. To reduce ibuprofen consumption in the IT sector, Ongaro and Ousterhout devised an easily understandable and implementable alternative to the Paxos algorithm, namely the Raft consensus algorithm, which we employ in this work to implement communication via XML-RPC between multiple Pygame applications, each running a game instance that aims to simulate a simplified version of a real-time strategy game.

CCS Concepts: • **Computer systems organization → Fault-tolerant network topologies**; • **Computing methodologies → Concurrent algorithms**; **Self-organization**; *Graphics input devices.*

Additional Key Words and Phrases: Python, Raft, Distribution, Consensus, Gaming, Multiplayer, Multithreading, RPC, Pygame

**ACM Reference Format:**

Tullio Vardanega and Marco Bellò. 2018. Raftian: Pygame Meets Raft. 1, 1 (August 2018), 5 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

From sharing spreadsheets between a handful of laptops in a small basement office, through large-scale rendering on a supercomputer, to the entire global finance system, distributed computing has become an essential component of the modern world that we almost take for granted: nowadays, what most people need a computer for can be done in the browser thanks to services like email clients, cloud calendars, media streaming platforms and web-based office suites (like Google Docs) that expose word editors, spreadsheets managers, presentations programs and more, all while being constantly synchronized to the cloud, which not only ensures data persistence and availability, but also enables sharing and collaboration between users.

It does not end here: other examples of distributed applications include cloud storage services like Dropbox, Google Drive or OneDrive, streaming services like Netflix, YouTube or Spotify, distributed computing like blockchain technologies or AWS, online banking services (the banking system itself is distributed since way before), social networks, and even maritime and aircraft traffic control systems. Moreover, the rise of the gaming industry played a significant role in pushing forward distribution: in 2024 the gaming market revenue was estimated to be 187.7 billion U.S. dollars [2], making it a hefty slice of the pie that is the entertainment industry [10], with 111 billions generated by free-to-play games [3] (70 billions from social and casual games alone[6]), which interests us since their business model often relies on cosmetics, game passes and advertisements, forcing them to be constantly on-line.

*Both authors contributed equally to this research.

Authors' address: Tullio Vardanega, tullio.vardanega@unipd.it; Marco Bellò, marco.bello.3@studenti.unipd.it, University of Padua, Padua, Italy.


Manuscript submitted to ACM

Let's now define what distribution *is*: a distributed system is a computer system whose inter-communicating components are located on different networked computers [14] [1], which coordinate their actions via message-passing to achieve a common goal. There are three significant challenges to overcome: maintaining components' concurrency [1], eliminating global-lock reliance and managing the independent failure of components, all while ensuring scalability (often the purpose is scaling itself) and transparency to the user, meaning interactions with any exposed interface must be done while being unaware of the complexity behind them.

Most importantly, shared consensus must be guaranteed: it does not require much thought to see that all servers in a cluster should agree on one or more shared values, lest becoming a collection of un-related components that have little to do with collaboration (thus distribution). In the most traditional single-value consensus protocols, such as Paxos [7], cooperating nodes agree on a single value (e.g., an integer), while multi-value alternatives like Raft [8] aim to agree on a series of values (i.e., a log) growing over time forming a sort-of cluster's history. It is worth noting that both goals are hindered by the intrinsically asynchronous nature of real-world communication, which make it impossible to achieve consensus via deterministc algorithms, as stated by Fischer, Lynch and Paterson in their FLP impossibility theorem [4]. Thankfully this can be circumvented by injecting some degree of randomness.

The concepts and examples we mentioned so far allow us to finally present the goal of this project: we will create a simplified clone of Travian [2], an old real-time [3] player-versus-player [4] strategy game [5], where players build their own city and wage war on one another (less wrinkly readers may be more familiar with the modern counterpart Clash of Clans [6]), built with Pygame [7], a Python library that creates and manages all necessary components to run a game such as game-engine, graphical user interface, sounds, plyer inputs and the like, where each player resides in a separate server (or node) that communicate with the others via an algorithm modelled after Raft's specifications.

This choice follows the author's interest in exploring Raft capabilities and ease of implementation in a fun and novel way, using a language that while extremely popular is seldom used in such a fashion.
Both game and algorithm implementations have been reduced to a reasonably complex proof of concept to keep the project scope manageable: it is possible to instantiate games up to five players, each of which is restricted to the only action of attacking the others, while Raft's functionalities are limited to log replication and overwriting.
Experiments were conducted to evaluate both game responsiveness and the communication algorithm correctness.
All source code is visible at the following link: https://github.com/mhetacc/RuntimesConcurrencyDistribution/blob/main/raftian/raftian.py.

[1]Concurrency refers to the ability of a system to execute multiple tasks through simultaneous execution or time-sharing (context switching), sharing resources and managing interactions. It improves responsiveness, throughput, and scalability [13] [9] [5] [11] [15].

[2]Travian: Legends is a persistent, browser-based, massively multiplayer, online real-time strategy game developed by the German software company Travian Games. It was originally written and released in June 2004 as "Travian" by Gerhard Müller. Set in classical antiquity, Travian: Legends is a predominantly militaristic real-time strategy game. Source: https://www.travian.com/international

[3]Real-time games progresses in a continuous time frame, allowing all players (human or computer-controlled) to play at the same time. By contrast, in turn-based games players wait for their turn to play.

[4]Player-versus-player (PvP) is a type of game where real human players compete against each other, opposed to player-versus-environment (PvE) games, where players face computer-controlled opponents.

[5]Strategy video game is a major video game genre that focuses on analyzing and strategizing over direct quick reaction in order to secure success. Although many types of video games can contain strategic elements, the strategy genre is most commonly defined by a primary focus on high-level strategy, logistics and resource management. [12]

[6]Clash of Clans: https://supercell.com/en/games/clashofclans/

[7]Pygame: https://www.pygame.org/docs/

## 2 ACKNOWLEDGMENTS

Identification of funding sources and other support, and thanks to individuals and groups that assisted in the research and the preparation of the work should be included in an acknowledgment section, which is placed just before the reference section in your document.

This section has a special environment:

```
\begin{acks}
...
\end{acks}
```

so that the information contained therein can be more easily collected during the article metadata extraction phase, and to ensure consistency in the spelling of the section heading.

Authors should not prepare this section as a numbered or unnumbered `\section`; please use the “`acks`” environment.

## 3 APPENDICES

If your work needs an appendix, add it before the “`\end{document}`” command at the conclusion of your source document.

Start the appendix with the “`appendix`” command:

```
\appendix
```

and note that in the appendix, sections are lettered, not numbered. This document has two appendices, demonstrating the section and subsection identification method.

## 4 MULTI-LANGUAGE PAPERS

Papers may be written in languages other than English or include titles, subtitles, keywords and abstracts in different languages (as a rule, a paper in a language other than English should include an English title and an English abstract). Use `language=...` for every language used in the paper. The last language indicated is the main language of the paper. For example, a French paper with additional titles and abstracts in English and German may start with the following command

```
\documentclass[sigconf, language=english, language=german,
               language=french]{acmart}
```

The title, subtitle, keywords and abstract will be typeset in the main language of the paper. The commands `\translatedXXX`, `XXX` begin title, subtitle and keywords, can be used to set these elements in the other languages. The environment `translatedabstract` is used to set the translation of the abstract. These commands and environment have a mandatory first argument: the language of the second argument. See `sample-sigconf-i13n.tex` file for examples of their usage.

## 5 SIGCHI EXTENDED ABSTRACTS

The “`sigchi-a`” template style (available only in LaTeX and not in Word) produces a landscape-orientation formatted article, with a wide left margin. Three environments are available for use with the “`sigchi-a`” template style, and produce formatted output in the margin:

**`sidebar`:** Place formatted text in the margin.

**marginfigure:** Place a figure in the margin.

**margintable:** Place a table in the margin.

## ACKNOWLEDGMENTS

To Robert, for the bagels and explaining CMYK and color spaces.

## REFERENCES

[1] 2009. *Distributed Programs.* Springer London, London, 373–406. https://doi.org/10.1007/978-1-84882-745-5_11

[2] Michiel Buijsman, Devan Brennan, Tianyi Gu, Lester Isaac Simon, Tomofumi Kuzuhara, Spyros Georgiou, Michael Wagner, Ngoc Linh Nguyen, Brett Hunt, Alejandro Marin Vidal, and Tiago Reis. 2024. Newzoo's Global Games Market Report 2024. https://newzoo.com/resources/trend-reports/newzoos-global-games-market-report-2024-free-version

[3] Jessica Clement. 2025. Free-to-play (F2P) games market revenue worldwide from 2018 to 2024. https://www.statista.com/statistics/324129/arpu-f2p-mmo/

[4] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. 1985. Impossibility of distributed consensus with one faulty process. *J. ACM* 32, 2 (April 1985), 374–382. https://doi.org/10.1145/3149.214121

[5] George Coulouris, Jean Dollimore, Tim Kindberg, and Gordon Blair. 2012. *Distributed Systems: Concepts and Design* (5 ed.). Addison-Wesley.

[6] Rida Khan and Zack Aboulazm. 2023. $300 Billion of Video Gaming Revenue, by Segment (2017-2026F). https://www.visualcapitalist.com/sp/video-games-industry-revenue-growth-visual-capitalist/

[7] Leslie Lamport. 1998. The part-time parliament. *ACM Trans. Comput. Syst.* 16, 2 (May 1998), 133–169. https://doi.org/10.1145/279227.279229

[8] Diego Ongaro and John Ousterhout. 2014. In search of an understandable consensus algorithm. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference* (Philadelphia, PA) *(USENIX ATC'14)*. USENIX Association, USA, 305–320.

[9] David A. Patterson and John L. Hennessy. 2008. *Computer Organization and Design, Fourth Edition, Fourth Edition: The Hardware/Software Interface (The Morgan Kaufmann Series in Computer Architecture and Design)* (4th ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[10] PWC. 2025. Global entertainment and media industry revenues to hit US$3.5 trillion by 2029, driven by advertising, live events, and video games: PwC Global Entertainment & Media Outlook. https://www.pwc.com/gx/en/news-room/press-releases/2025/pwc-global-entertainment-media-outlook.html

[11] Michael J. Quinn. 1994. *Parallel computing (2nd ed.): theory and practice.* McGraw-Hill, Inc., USA.

[12] A. Rollings and E. Adams. 2003. *Andrew Rollings and Ernest Adams on Game Design.* New Riders. https://books.google.it/books?id=Qc19ChiOUI4C

[13] Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne. 2008. *Operating System Concepts* (8th ed.). Wiley Publishing.

[14] A.S. Tanenbaum and M. van Steen. 2017. *Distributed Systems.* CreateSpace Independent Publishing Platform. https://books.google.it/books?id=c77GAQAACAAJ

[15] Albert Y. H. Zomaya. 1996. *Parallel and distributed computing handbook.* McGraw-Hill, Inc., USA.

## A RESEARCH METHODS

### A.1 Part One

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi malesuada, quam in pulvinar varius, metus nunc fermentum urna, id sollicitudin purus odio sit amet enim. Aliquam ullamcorper eu ipsum vel mollis. Curabitur quis dictum nisl. Phasellus vel semper risus, et lacinia dolor. Integer ultricies commodo sem nec semper.

### A.2 Part Two

Etiam commodo feugiat nisl pulvinar pellentesque. Etiam auctor sodales ligula, non varius nibh pulvinar semper. Suspendisse nec lectus non ipsum convallis congue hendrerit vitae sapien. Donec at laoreet eros. Vivamus non purus placerat, scelerisque diam eu, cursus ante. Etiam aliquam tortor auctor efficitur mattis.

## B ONLINE RESOURCES

Nam id fermentum dui. Suspendisse sagittis tortor a nulla mollis, in pulvinar ex pretium. Sed interdum orci quis metus euismod, et sagittis enim maximus. Vestibulum gravida massa ut felis suscipit congue. Quisque mattis elit a risus ultrices commodo venenatis eget dui. Etiam sagittis eleifend elementum.

Nam interdum magna at lectus dignissim, ac dignissim lorem rhoncus. Maecenas eu arcu ac neque placerat aliquam. Nunc pulvinar massa et mattis lacinia.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009