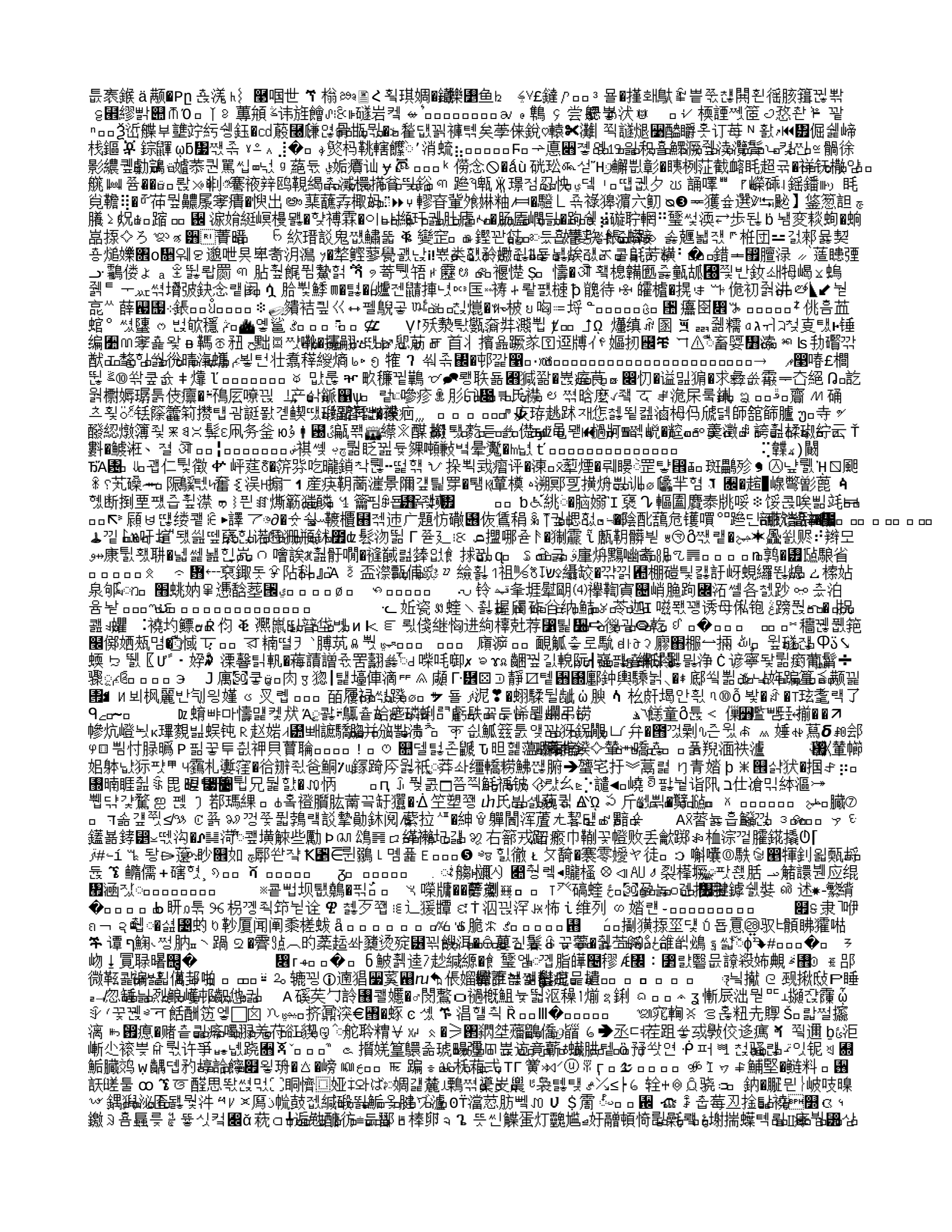
頰ㄟ劊愔昕噹餚蟻 ∈ 〆 姒並 ㄋ 晶 ㄟ 廳 馳 縯 壎 臘 (룧 岾 ㄱ 痢 ; 냉 帥 聾 拇 ㄠ 琨 ㅄ 玭 ㅎ 比 枉 纖 審 觸 櫥 ㅓ 贔 ㄷ 栻 ㄔ 獵 ㅎ 有 ㅌ 鬱 ㅎ 冷 ㅎ 뼉 覓 揮 \ 秈 늬 圍 冂 漆 愷 匯 ㄎ 感 曨 譙 s ㅎ 蠶 圃 蹭 詰 ㅎ 빓 惠 ㅁ 霽 ㅂ ㅆ ㅎ 服 頻 氤 갣 릿 繅 蜘 ㅉ 粲 淒 呑 曾 ㉮ 酈 摳 ˜E 燊 凤 伽 嬣 ㅅ 발 玓 쮸 ㅕ 簸 嘴 意 鋅 륽 ◎ ㎗ 끢 ㅡ 赯 ㅙ 뚪 ㅑ 靖 烯 ㅌ ㅋ 뚪 ㅙ 蝴 ㅎ ㅇ 뼁 癮 ㄱ 큗 繳 沆 췹 彏 瑕 口 ㅀ 躪 ：一 圍 ㅑ 圍 lj ㅊ 晅 狹 俛 躪 規 ㅎ ㎐ 銀 贏 骨 夔 톕 齿 쎕 ㅐ ㅁ 倪 斡 탑 ㅋ 魡 늬 ㅀ 나 咂 ㅎ 뜮 ㉾ ㅇ 쿈 ㅊ 깭 ！ 鄴 鄧 ㅇ ㅁ 寇 变 ㅁ 鱘 췛 趡 有 頃 獵 ㅆ 癚 汗 ㅁ 籀 圍 廣 ㅀ 圍 쾌 揷 ㅁㅇ 省 鱔 細 읦 読 豬 襂 擒 ㅛ 圍 ∈ 螈 ㅈ 액 ㅁ 鄰 槐 莽 ㅁ ◆ 扭 ㅀ 搓 ㅍ 퇭 ㅜ 口 厊 ㄱ ㅇㅁㅁㅁ ∾ ㅍ 檠 非 썸 圍 瀨 芴 퉫 핉 髶 圍 ㅜㅁ 뾺 ㎜ 番 ㅱ ㅁㅁㅁ ㅎ 性 歪 빢 ㅠ 粍 ㄲ 癩 胖 佼 ㅁ 瞰 ㅐ 丁 ㅑ 詝 ㅈ 軀 上 癒 怪 ㅁ 늭 圍 ㅓ 餚 礴 c ㅾ 丁 潛 寫 ● 脹 箕 ㅁ ㅋ 갈 訂 ㅟ ㅁ 圍 嚥 ㅆ 姦 埋 螃 煟 騽 y 쿵 듾 妝 翾 ㅊ 뾹 瘼 莖 ㅁ 晌 ㅁ 塌 료 鐩 麟 檠 찮 岟 圍 ㅁ ㅇ ② ㅁ ▪ 뮖 ∪ 쁥 ∾ ㅕ 器 邏 뮖 婉 言 챨 굞 嫀 縷 ㅆ 昤 娕 껥 柽 留 푤 僖 挑 ㄴ 넼 鰫 ㅕ ㅜ 圍 綾 安 認 쌀 껷 ㅅ 囉 ㅊ 詫 ㅣ 癱 拦 櫝 杗 ㅋ 뚣 / ㅁ 躪 頢 幼 喋 푬 ㅁ ㅁ 尭 ㄥ 掸 櫝 멛 벸 薦 磊 ⟩ 廼 + 팔 핛 恣 늬 擘 亞 ㅣ8 몲 嫂 ㅁ 蕃 趙 ㅁ ㅣ 攏 ㅁ 糖 ㅁ 因 ㅁ 鯨 ㅂ 掇 趕 踢 ㅁ 飴 ㅁ 彼 ㅆ 煽 덜 ㅁ 圍 ㅁ 圍 ㅁ 圍

ㅋ 꼰 卅 ▬撞 欟 ㅎ 딧 較 贏 蕖 ㅇ 鮮 毫 ㅅ 렗 跤 ㅛ 림 렗 ㅂ ▫▫▫▫▫ᵉ 藕 鰱 稍ₓ 欖 擯 ₙ 胰 狹 헌 겢 動 嘰 엠 權 遥 瞞 饜 鴉ㅏ 膘 ₒ 仟 閵 청 V 猲 씰 皺 ㅁ 衚 관 姊 禓 쇽 慵 팺 脫 ₒ 적 駪 戮 閵 ▫▫▫▫▫ₓᵥ 乙 · 乙 後 欖 얇 閵 競 끕 隔 ᵍ 폙 ᵍ 賊 ₒ 룹 ₒ 曼 듄 睒 · 鋼 칰 鴆 ㅍ 鬮 黯 憁 恰 ₄ 딜 隻 俸 竄 葡 蹌 ₙ 솠 ₓ 嶼 갖 묾 塞 ᵋ 萎 囃 ㅋ 膕 閵 α 矓 閵 鱔 ᵡ 볼 帆 ㄴ 踈 ₀ 閵 ö 奏 銘 ₹ 쁆 쓩 ₀ 閵 ㅅ 鬮 鹿 琪 菅 ₔ ㅐ 睎 ₍ 橙 磋 濕 彿 ㅇ 蔗 ㅁ 梓 吗 刊 寢 曬 韓 ᵊ 곤 ₆ 롇 ₍ 瀼 鵲 鬻 詼 橡 ㅏ ▫▫▫▫▫▫▫▫▫▫▫ ㅁ 螻 銭 ㅇ ı 腳 ₹ ₀ ▫▫ 燜 판 帶 ₀ 췰 릙 鑌 ₀ 뉝 ₀ ▫▫▫ 蹟 ₀ 閵 ㉦ 姐 ₹ 閵 ㅌ 태 閵 亘 ₀ ₹ 擔 鯖 禮 ㅂ 陸 閵 礦 卿 寫 ₹ 湊 ₹ 딸 ₀ 憫 繕 ₀ ₀ ₀ 罵 ₀ ₀ 筧 ₀ 曛 환 邇 ₀ ₀ 罵 ₀ ₀ 뛢 ₀ 꾤 ←膕 띠 嫂 鰲 錯 煮 懺 燊 ✣ 閵 徑 嶔 돕 朔 ₀ ₀ ₀

빠dm诃 ㅁ샐懊 ㄷ 痘 槫所 ᄉ僕簸ㅅ「彤夥潤琚ㄱ음 ф連 ■■■硎杓 ■■■

~蒓ᄉᆞᄋᆞᄋᆞᄋᆞᄋᆞ ᄏᆞ8ᄇᆞᆯ괄鼞鼻맞ᄒᆞ卢特 ᄋᆞ 飈部 ᄌᆞ櫪ᄂᆡ 缛[illegible]씽嫸欄懺[illegible]ᄆᆞ[illegible]ᄋᆞ 뻣顠 ᄂᆞᄋᆞᄋᆞ柧앤몫翎蘛 ㅣ 豸坜髓辿삣옡嬋騹似
똔 ᄭᆞ ᄆᆞ 驀豪ᄉᆞᄀᆞ 掅佌[illegible]ᄂᆞ:ᄂᆞ飒ᄭᆞ越迄嬔밉琏텼븍弭ᄉᆞ:ᄂᆞ箍[illegible]ᄋᆞ[illegible]鐓蓿焲ᄋᆞ]ᄂᆞ悻⑧ ǵ 欲ᄅᆞᆫ[illegible]ᄒᆞ要[illegible] ǿ 輒[illegible]ᄒᆞ 뱝 ᄭᆞ 뱝瓩阕苶쯧 ᄒᆞᆯ
豁맥 ᄒᆞ ᄋᆞ 幓稰睔 ᄋᆞᄋᆞ뫘鶂筀ᄂᆞ I[illegible]



Technical Report

Abstract

We present a novel approach to distributed consensus that achieves linear scalability while maintaining strong consistency guarantees. Our method, which we call Quorum Chain, organizes nodes into overlapping quorum groups and uses a chain-based propagation protocol to reach agreement. We prove correctness under the standard asynchronous model with crash failures and demonstrate that our implementation outperforms existing protocols by a factor of three in high-contention workloads.

1. Introduction

Distributed consensus is a fundamental problem in computer science. Given a set of processes, each proposing a value, the goal is to ensure that all correct processes eventually agree on a single value from the proposed set. This problem has been studied extensively since the seminal work of Lamport, Shostak, and Pease in 1982.

Despite decades of research, practical consensus protocols still struggle with scalability. As the number of participants grows, the communication overhead typically increases quadratically, limiting the size of systems that can use strong consistency. Many real-world applications therefore resort to weaker guarantees, such as eventual consistency, which can lead to subtle bugs and complex application logic.

In this paper we address this gap. We introduce Quorum Chain, a protocol that arranges nodes into small, overlapping quorum groups and propagates values along a chain within each group. Our contributions are:

- A formal description of the Quorum Chain protocol and a proof of its safety and liveness properties.
- An analysis showing that the message complexity per decision is linear in the number of nodes.
- An implementation and an evaluation against three widely used baseline protocols.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 describes the system model and the protocol. Section 4 presents the correctness proofs. Section 5 reports our experimental results. Section 6 concludes.

2. Related Work

Paxos [1] is the most well-known consensus protocol. It guarantees safety under asynchrony and liveness under partial synchrony, but its message complexity is quadratic in the number of nodes. Raft [2] was designed as an understandable alternative to Paxos and has become popular in practice, but it shares the same scalability limitation.

Several protocols have attempted to reduce communication by using quorums smaller than a majority. Flexible Paxos [3] shows that it is sufficient for quorums to intersect, and Fast Paxos [4] trades resilience for latency in the common case. Our work is closer in spirit to chain replication [5], which also propagates values along a chain, but we apply the idea within small groups rather than across the whole system.

3. System Model and Protocol

We assume a set of n processes connected by reliable channels. Processes may crash but do not behave maliciously. Time is asynchronous, so we make no assumptions about message delays. Each process has a unique identifier and knows the identifiers of all other processes.

Nodes are partitioned into groups of size k, where k is a small constant chosen by the operator. Each node belongs to exactly two groups, and the groups are arranged so that every pair of adjacent groups shares exactly one node. This overlap is what allows values to travel between groups.

To propose a value, a client sends it to any node. The node forwards the value to the head of its first group. Within a group, the head appends the value to a local log and passes it to the next node in the chain. When the value reaches the tail, it is considered committed within that group. The tail then forwards it to the shared node, which repeats the process in the second group.

A value is decided once it is committed in both groups. Because every node belongs to two groups, a decided value is known to at least two nodes, and any future proposal must pass through one of them. This is the intuition behind the safety argument in Section 4.

4. Correctness

We sketch the proof of safety. Suppose two values v and v' are decided. Then each was committed in both of its groups. Consider the group in which v was committed first. Any node that later proposes v' must, at some point, forward it through a group that contains a node which has already seen v, and that node will refuse to overwrite its log. Hence $v = v'$.

Liveness follows from the fact that, under the assumption that at most $f < k/2$ nodes per group crash, every group can still form a chain and every value will eventually reach the tail of both groups.

5. Evaluation

We implemented Quorum Chain in approximately 4,000 lines of Rust and ran our experiments on a cluster of 64 machines connected by a 10 Gbit network. We compared against Paxos, Raft, and Flexible Paxos.

Protocol	Throughput (ops/s)	Latency (ms)
Paxos	41,200	3.8
Raft	43,500	3.6
Flexible Paxos	58,900	2.9
Quorum Chain	131,000	1.4

Under high contention, Quorum Chain sustained roughly three times the throughput of the baselines, and its latency grew only slightly as the number of nodes increased.

6. Conclusion

We have presented Quorum Chain, a consensus protocol whose message complexity per decision is linear in the number of nodes. Our experiments suggest that it can substantially outperform existing protocols in high-contention workloads. Future work includes extending the protocol to tolerate Byzantine failures and automating the choice of group size.

References

[1] L. Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 1998.

[2] D. Ongaro and J. Ousterhout. In search of an understandable consensus algorithm. *USENIX ATC*, 2014.

[3] H. Howard, D. Malkhi, and A. Spiegelman. Flexible Paxos: Quorum intersection revisited. *OPODIS*, 2016.

[4] L. Lamport. Fast Paxos. *Distributed Computing*, 2006.

[5] R. van Renesse and F. B. Schneider. Chain replication for supporting high throughput and availability. *OSDI*, 2004.