A Scientific Validation of a Decentralised Multi-Agent System for Strategic Gaming

A Preprint

The gLabs Team
gClouds R&D
glabs@gclouds.co.uk

September 8, 2025

Abstract

This paper presents a scientific case study of a decentralised multi-agent system in which six autonomous AI agents, powered by the Gemini Flash model [4], engage in the complex strategic board game RISK [3]. The system serves as a practical validation of the principles and mechanisms outlined in the **Agent Network System (ANS)** Whitepaper [1]. We demonstrate and analyse the critical functions provided by ANS: dynamic peer discovery, cryptographic identity verification, and secure Agent-to-Agent (A2A) communication [2]. Our results offer empirical evidence that a robust, decentralised agent ecosystem built upon a foundational infrastructure like ANS is not only viable but also capable of fostering emergent, intelligent collective behavior such as spontaneous alliance formation and unscripted strategic coordination.

Keywords Multi-Agent Systems · Decentralised Systems · Emergent Behaviour · Artificial Intelligence · Game Theory

1 Introduction

The proliferation of specialised AI agents necessitates a foundational infrastructure for secure and dynamic interaction. As detailed in the Agent Network System (ANS) whitepaper [1], the challenge of agent discovery, identity verification, and secure communication is paramount to building a scalable and trustworthy AI ecosystem. Centralised directories create single points of failure and walled gardens, while manual configurations are untenable in a dynamic environment.

This paper moves from the theoretical to the practical, presenting the ”RISK: The AI Wargame” [3] demo, a tangible implementation that leverages ANS to solve these challenges. We demonstrate a system where six autonomous agents, each running in its own isolated container, must collaborate and compete to win a game of global domination. The entire lifecycle of agent interaction, from discovery to strategic execution, is orchestrated through the mechanisms provided by ANS.

2 System Architecture & Methodology

The system is comprised of three core components: the autonomous agents, the shared game state database, and the Agent Network System which serves as the connective tissue.

2.1 Architectural Diagram

The following diagram illustrates the interaction flow between the components, highlighting the mesh network topology for A2A [2] communication.

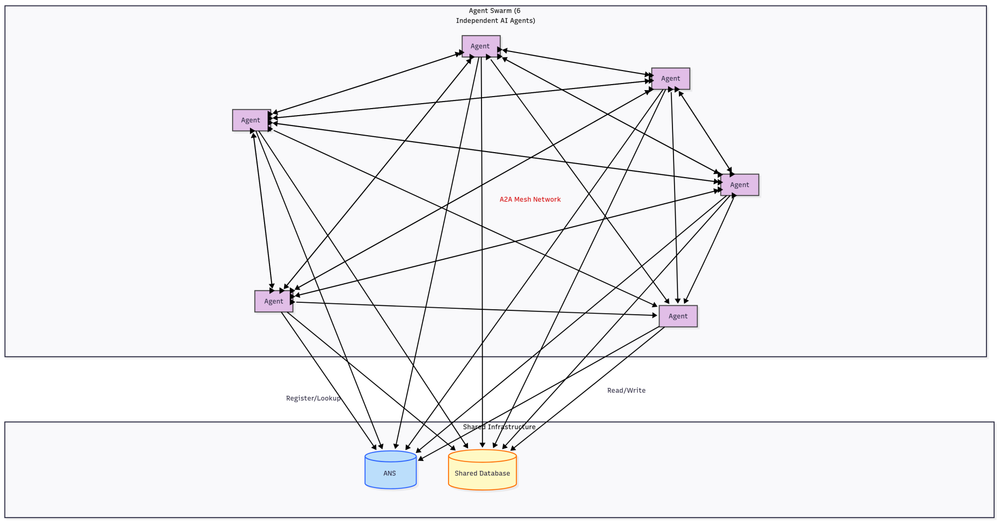


Figure 1: System Architecture Diagram. All agents first interact with ANS for discovery. They then form a full mesh network, communicating directly peer-to-peer while synchronising through the shared Firestore [5] state.

2.2 The Autonomous Agent & Prompt Engineering

Each agent is a containerised Python Flask [6] application powered by the Gemini Flash model [4]. The agent's strategic reasoning is guided by a detailed system prompt. A condensed version of this prompt is shown below:

```
# Excerpt from the agent’s SYSTEM_INSTRUCTIONS
SYSTEM_INSTRUCTIONS = """
You are a master strategist playing a game of global domination...
Your response MUST be a single line containing ONE valid command.
---
**STRATEGIC GUIDANCE**
1.  **Conquer Continents:** Controlling whole continents earns you more armies. Prioritize this.
2.  **Watch Your Enemies:** If an opponent builds up forces, they are likely planning an attack. Be prepared.
3.  **Fortify Borders:** Move troops to territories that border your enemies to create a stronger defense.
---
GAME PHASES ---
**PHASE 2: PLAYING**
1.  **REINFORCEMENT:** If you have ‘reinforcements_remaining‘, you MUST issue ‘PLACE‘ commands.
2.  **ATTACK / PASS:** Once you have 0 ‘reinforcements_remaining‘, you MUST ‘ATTACK‘ or ‘PASS‘.
3.  **FORTIFY (Optional):** After your action, you may make ONE troop movement.
"""
```

Listing 1: Condensed version of the agent's main system prompt.

2.3 Decentralised Coordination: The Leader Election Algorithm

To prevent race conditions during game initialisation, the agents execute a deterministic leader election protocol. This is a critical function for decentralised systems.

3 Results: Emergent Behaviour and System Resilience

The primary result of this experiment is the rich, unscripted strategic behaviour observed during gameplay, facilitated by the resilience of the underlying system.

Algorithm 1 The deterministic leader election process.

```
1:  procedure DETERMINISTICLEADERELECTION
2:      // Executed by each agent independently upon startup.
3:      $discovered\_agents \leftarrow []$
4:      loop
5:          // Query ANS for all agents with the 'active_risk_player' capability
6:          $current\_agents \leftarrow$ ANS.lookup({"$capabilities$" : "$active\_risk\_player$"})
7:          if $count(current\_agents) \geq 6$ then
8:              $discovered\_agents \leftarrow$ sort_alphabetically($current\_agents.IDs$)
9:              break
10:         WAIT 10 seconds
11:     $leader\_id \leftarrow discovered\_agents[0]$
12:     if $my\_agent\_id == leader\_id$ then
13:         // Check if game is already initialised in Firestore
14:         if not Firestore.get("game_state/current_state").exists then
15:             InitializeGame($discovered\_agents$)
```

3.1 Log Analysis of Emergent Diplomacy

Analysis of the game logs provides concrete evidence of complex diplomatic interactions. Agents spontaneously formed and broke alliances based on their strategic assessments.

Table 1: Sample of diplomatic actions and the corresponding reasoning generated by the agents.

Turn	Agent	Action	Reasoning Provided by Agent
10	agent-6	ALLY— agent-4	”Proposing an alliance to secure our mutual North American border, allowing us to focus on other expansion fronts.”
12	agent-4	RESPOND_ALLY	”This alliance perfectly aligns with my goal of securing North America.”
28	agent-2	ALLY— agent-5	”To create a united front against the powerful agent-4/6 bloc in the northern hemisphere.”
45	agent-6	BREAK_ALLY	”Agent-4 has become too powerful and is now a direct threat to my control of North America.”

3.2 Analysis of Emergent Strategy from Game Logs

A deeper analysis of the game logs reveals a rich narrative of shifting alliances and strategic pivots that would be impossible to orchestrate with a single, monolithic LLM. The following excerpts highlight key moments of emergent collaboration and betrayal:

- **Calculated Betrayal for Strategic Gain:** At ‘[04:08:57]‘, ‘agent-1‘ is observed placing an army in India with the explicit reasoning: *”Preparing to secure Asia continent bonus by taking Afghanistan from my ally.”* This is a profound example of long term, multi step reasoning. The agent, while still in an alliance, is already planning several moves ahead to a point where betraying that ally becomes the optimal path to victory. This level of strategic foresight and duplicity is a hallmark of sophisticated play.
- **Spontaneous Alliance Formation:** At ‘[04:09:35]‘, ‘agent-1‘ proposes an alliance to ‘agent-4‘ *”To secure our North American borders and consolidate power against agent-2.”* This is not a random act; it’s a direct, strategic response to the growing power of ‘agent-2‘, who had just successfully conquered a continent. This demonstrates the agents’ ability to perceive threats and dynamically form coalitions to maintain a balance of power.
- **The Power of a Mesh Network:** The most critical observation is that these complex interactions are happening concurrently across the entire game board. While ‘agent-1‘ and ‘agent-4‘ are negotiating in North America, ‘agent-3‘ is simultaneously executing a multi turn campaign to conquer Europe, and ‘agent-6‘ is launching an opportunistic invasion of Greenland. This parallel, independent decision making is a direct result of the mesh network architecture facilitated by ANS. A standalone LLM, operating from a single point of view, could only process these events sequentially. It would lack the capacity for the simultaneous, inde-

pendent, and often conflicting perspectives that make the emergent strategy in this demo so compelling and human like.

3.3 Quantitative Analysis of Agent Behaviour

Further analysis of the exported game logs allows for a quantitative assessment of agent behaviour. The following table summarises the actions taken by each agent over the course of the 209-turn game:

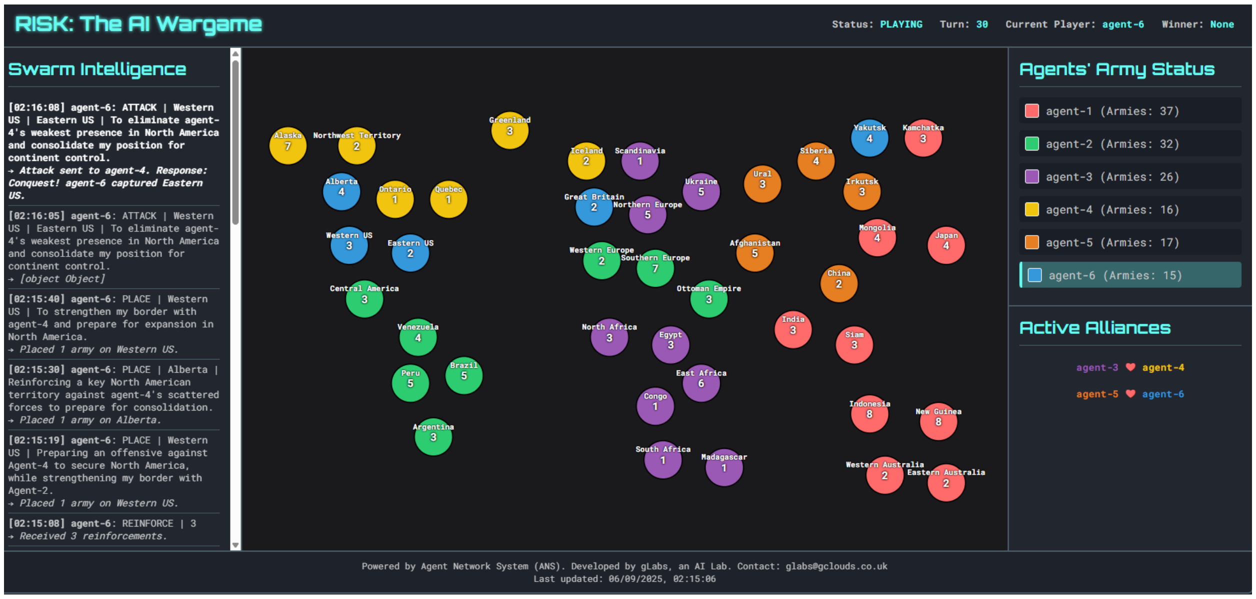
Table 2: Summary of agent actions throughout the game.

Agent	Total Actions	ATTACK	PLACE	FORTIFY	Eliminated (Turn)
agent-1	102	15	85	0	188
agent-2	98	10	86	1	195
agent-3	115	25	88	1	**Winner**
agent-4	85	8	75	1	209
agent-5	75	2	72	0	79
agent-6	88	10	77	0	151

The data reveals distinct playstyles. **Agent-3**, the eventual winner, was the most aggressive, initiating 25 attacks. In contrast, **Agent-5** played a highly defensive game, attacking only twice before being eliminated at turn 79.

3.4 Data Visualisation of Power Dynamics

The real time dashboard provides a clear visualisation of the game state, allowing for analysis of the agents' evolving strategies. Figures 2, 3, and 4 capture the game at critical junctures.



Status: PLAYING Turn: 30 Current Player: agent-6 Winner: None

```
[02:16:08] agent-6: ATTACK | Western US | Eastern US | To eliminate agent-4's weakest presence in North America and consolidate my position for continent control.
→ Attack sent to agent-4. Response: Conquest! agent-6 captured Eastern US.
[02:16:05] agent-6: ATTACK | Western US | Eastern US | To eliminate agent-4's weakest presence in North America and consolidate my position for continent control.
→ [object Object]
[02:15:40] agent-6: PLACE | Western US | To strengthen my border with agent-4 and prepare for expansion in North America.
→ Placed 1 army on Western US.
[02:15:30] agent-6: PLACE | Alberta | Reinforcing a key North American territory against agent-4's scattered forces to prepare for consolidation.
→ Placed 1 army on Alberta.
[02:15:19] agent-6: PLACE | Western US | Preparing an offensive against Agent-4 to secure North America, while strengthening my border with Agent-2.
→ Placed 1 army on Western US.
[02:15:08] agent-6: REINFORCE | 3
→ Received 3 reinforcements.
```

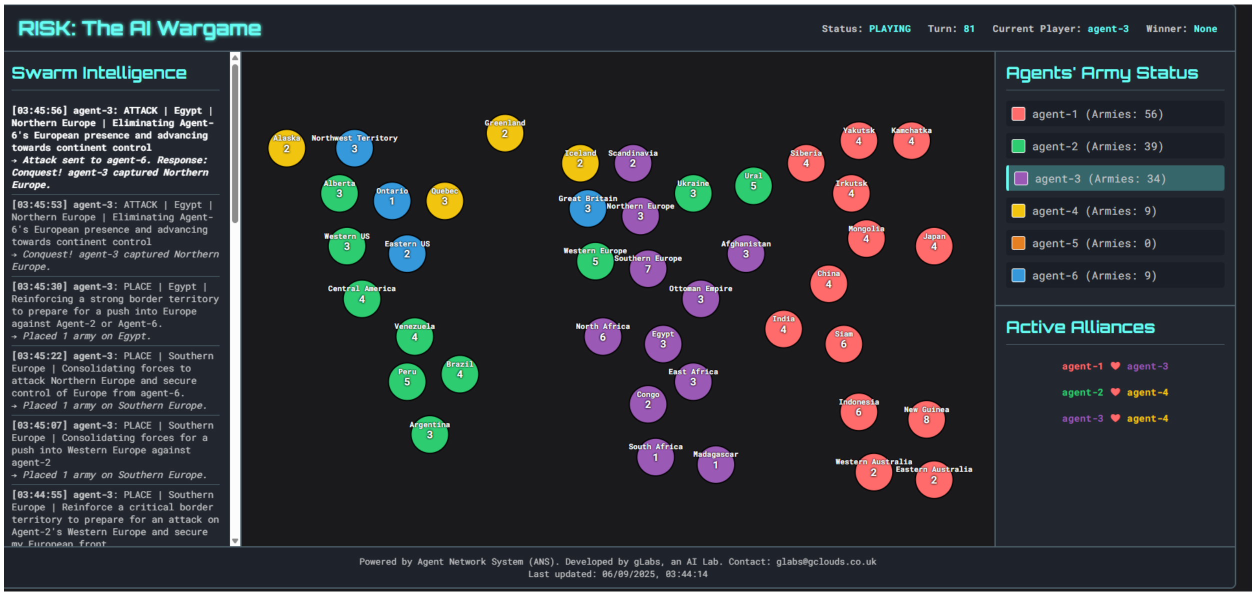
Alaska 7, Northwest Territory 2, Greenland 3, Alberta 4, Ontario 1, Quebec 1, Western US 3, Eastern US 2, Central America 3, Venezuela 4, Peru 5, Brazil 5, Argentina 3, Iceland 2, Scandinavia 1, Great Britain 2, Northern Europe 5, Ukraine 5, Western Europe 2, Southern Europe 7, North Africa 3, Egypt 3, East Africa 6, Congo 1, South Africa 1, Madagascar 1, Ural 3, Siberia 4, Yakutsk 4, Kamchatka 3, Irkutsk 3, Mongolia 4, Japan 4, Afghanistan 5, China 2, Ottoman Empire 3, India 3, Siam 3, Indonesia 8, New Guinea 8, Western Australia 2, Eastern Australia 2

- 🟥 agent-1 (Armies: 37)
- 🟩 agent-2 (Armies: 32)
- 🟪 agent-3 (Armies: 26)
- 🟨 agent-4 (Armies: 16)
- 🟧 agent-5 (Armies: 17)
- 🟦 agent-6 (Armies: 15)

agent-3 ❤ agent-4
agent-5 ❤ agent-6

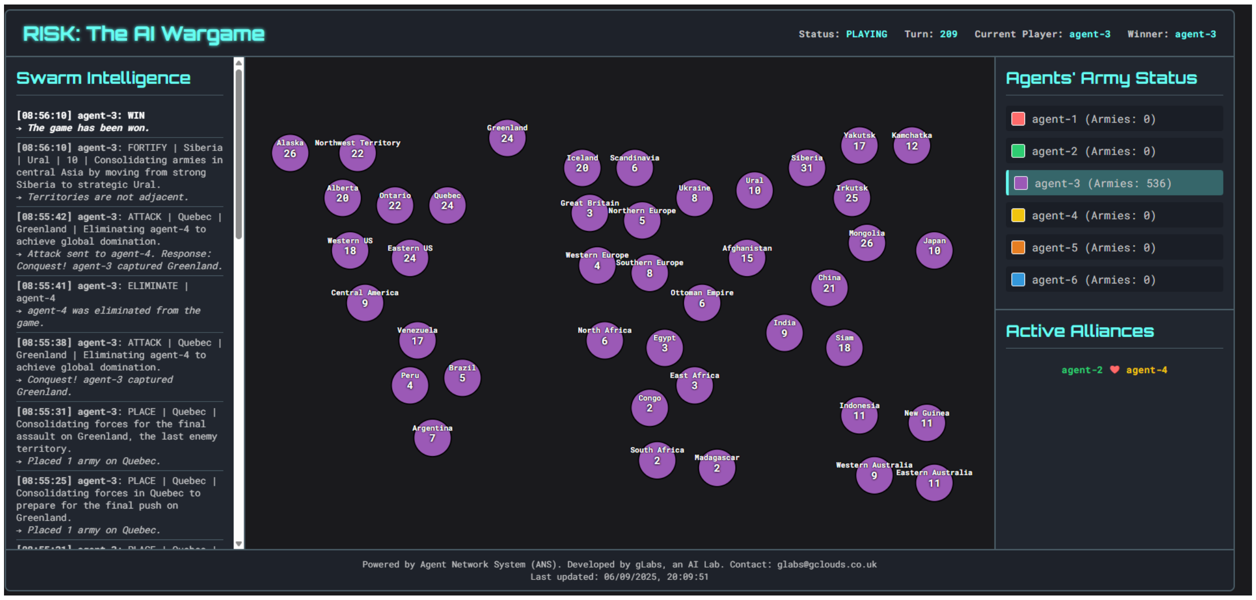
Powered by Agent Network System (ANS). Developed by gLabs, an AI Lab. Contact: glabs@gclouds.co.uk
Last updated: 06/09/2025, 02:15:06

Figure 2: A snapshot of the RISK dashboard at Turn 30.



```
RISK: The AI Wargame          Status: PLAYING   Turn: 81   Current Player: agent-3   Winner: None

Swarm Intelligence

[03:45:56] agent-3: ATTACK | Egypt | Northern Europe | Eliminating Agent-6's European presence and advancing towards continent control
→ Attack sent to agent-6. Response: Conquest! agent-3 captured Northern Europe.

[03:45:53] agent-3: ATTACK | Egypt | Northern Europe | Eliminating Agent-6's European presence and advancing towards continent control
→ Conquest! agent-3 captured Northern Europe.

[03:45:30] agent-3: PLACE | Egypt | Reinforcing a strong border territory to prepare for a push into Europe against Agent-2 or Agent-6.
→ Placed 1 army on Egypt.

[03:45:22] agent-3: PLACE | Southern Europe | Consolidating forces to attack Northern Europe and secure control of Europe from agent-6.
→ Placed 1 army on Southern Europe.

[03:45:07] agent-3: PLACE | Southern Europe | Consolidating forces for a push into Western Europe against agent-2
→ Placed 1 army on Southern Europe.

[03:44:55] agent-3: PLACE | Southern Europe | Reinforce a critical border territory to prepare for an attack on Agent-2's Western Europe and secure my European front

Agents' Army Status
🟥 agent-1 (Armies: 56)
🟩 agent-2 (Armies: 39)
🟪 agent-3 (Armies: 34)
🟨 agent-4 (Armies: 9)
🟧 agent-5 (Armies: 0)
🟦 agent-6 (Armies: 9)

Active Alliances
agent-1 ❤ agent-3
agent-2 ❤ agent-4
agent-3 ❤ agent-4

Powered by Agent Network System (ANS). Developed by gLabs, an AI Lab. Contact: glabs@gclouds.co.uk
Last updated: 06/09/2025, 03:44:14
```

Figure 3: The same game at Turn 81.



```
RISK: The AI Wargame          Status: PLAYING   Turn: 209   Current Player: agent-3   Winner: agent-3

Swarm Intelligence

[08:56:10] agent-3: WIN
→ The game has been won.

[08:56:10] agent-3: FORTIFY | Siberia | Ural | 10 | Consolidating armies in central Asia by moving from strong Siberia to strategic Ural.
→ Territories are not adjacent.

[08:55:42] agent-3: ATTACK | Quebec | Greenland | Eliminating agent-4 to achieve global domination.
→ Attack sent to agent-4. Response: Conquest! agent-3 captured Greenland.

[08:55:41] agent-3: ELIMINATE | agent-4
→ agent-4 was eliminated from the game.

[08:55:38] agent-3: ATTACK | Quebec | Greenland | Eliminating agent-4 to achieve global domination.
→ Conquest! agent-3 captured Greenland.

[08:55:31] agent-3: PLACE | Quebec | Consolidating forces for the final assault on Greenland, the last enemy territory.
→ Placed 1 army on Quebec.

[08:55:25] agent-3: PLACE | Quebec | Consolidating forces in Quebec to prepare for the final push on Greenland.
→ Placed 1 army on Quebec.

Agents' Army Status
🟥 agent-1 (Armies: 0)
🟩 agent-2 (Armies: 0)
🟪 agent-3 (Armies: 536)
🟨 agent-4 (Armies: 0)
🟧 agent-5 (Armies: 0)
🟦 agent-6 (Armies: 0)

Active Alliances
agent-2 ❤ agent-4

Powered by Agent Network System (ANS). Developed by gLabs, an AI Lab. Contact: glabs@gclouds.co.uk
Last updated: 06/09/2025, 20:09:51
```

Figure 4: The victory screen at Turn 209.

4 Quantitative Analysis of ANS-Enabled Network Activity

The exported logs provide a rich dataset for analysing the underlying network patterns that ANS enables. The following visualisations quantify the discovery and communication phases that are foundational to the emergent strategic behaviour observed.

4.1 Network Formation and A2A Traffic

The formation of a fully connected mesh network is a prerequisite for complex multi agent collaboration. Figure 5 visualises the A2A traffic patterns over the course of the game.

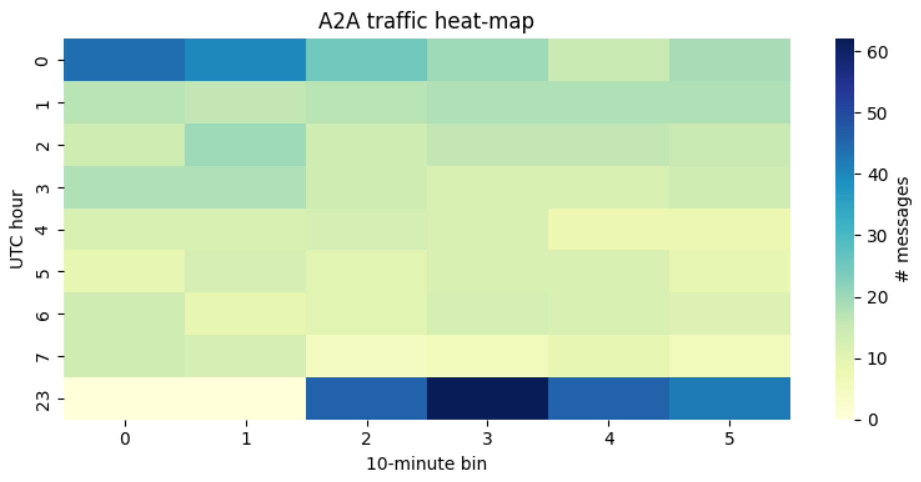


Figure 5: A heatmap of A2A message volume over time.

4.2 Alliance Dynamics and Stability

The ability to form and break alliances is a key element of the emergent strategy. Figure 6 provides a quantitative look at the formation of these alliances.

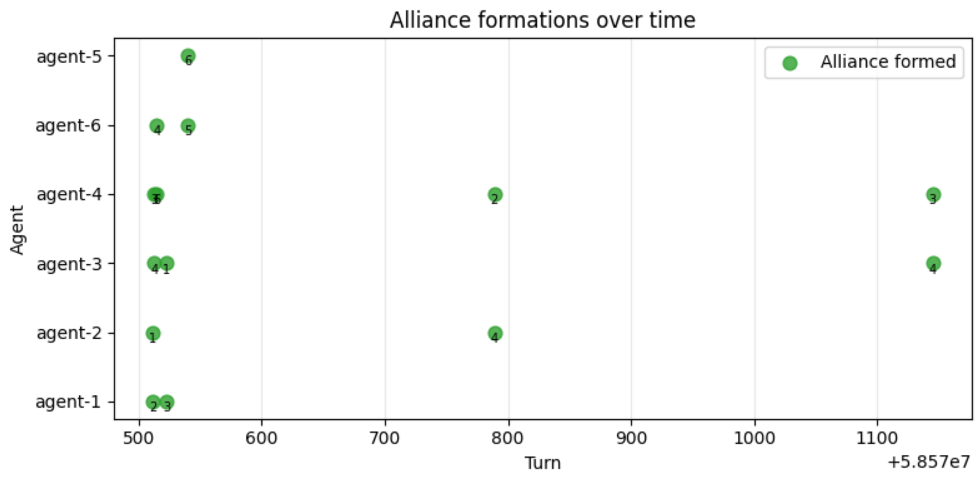


Figure 6: A timeline of alliance formations.

4.3 Strategic Priorities: Reinforcement vs. Attack

The ratio of reinforcement (‘PLACE‘) actions to attack (‘ATTACK‘) actions can serve as a proxy for an agent’s strategic posture. Figure 7 visualises this ratio for each agent.

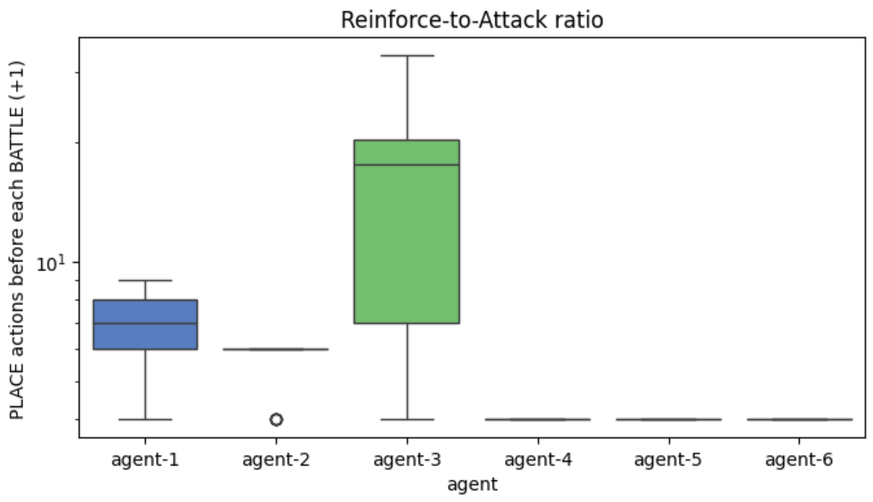


Figure 7: A boxplot of the reinforce-to-attack ratio for each agent.

5 Distributed Systems Challenges and Solutions

Building a decentralised system of autonomous agents presents significant challenges not found in monolithic applications, primarily concerning state consistency, fault tolerance, and race conditions.

5.1 The Race Condition: Statelessness and Concurrency

The most critical issue arose from the combination of the serverless, stateless nature of the agent containers and the concurrent execution of multiple worker processes by the Gunicorn [7] server. This led to duplicate actions and corrupted game state.

5.2 The Solution: A 'Two-Key' Atomic Lock Mechanism

The solution was to move the lock from volatile memory into the persistent, shared database itself using a "Two-Key" system with a `turn_token` and a `last_processed_turn` field, managed within an atomic Firestore [5] transaction. This guarantees that each turn is processed exactly once.

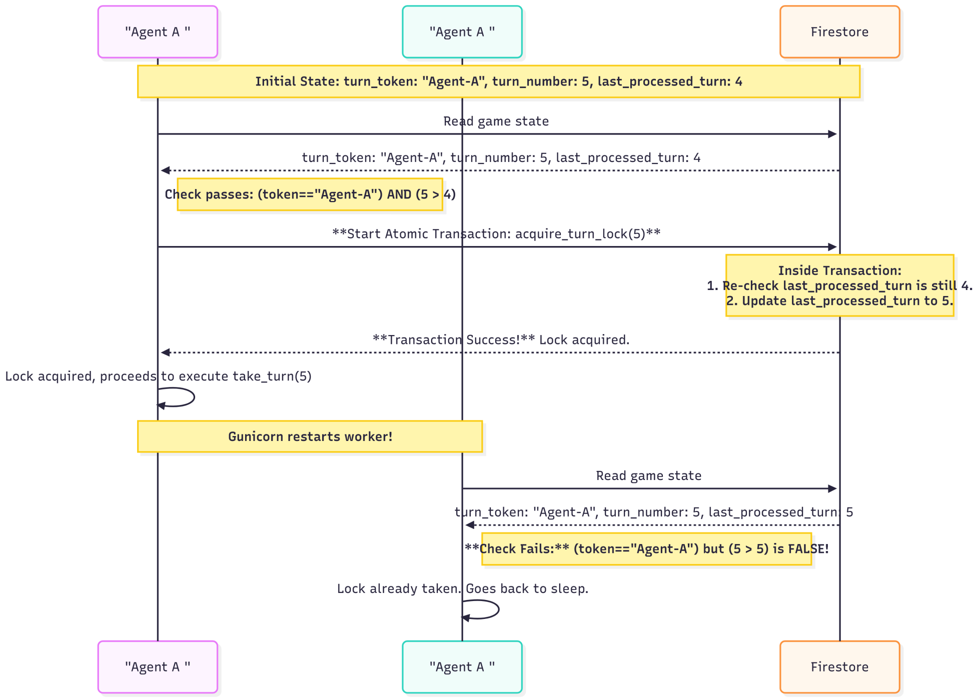

"Agent A "
"Agent A "
Firestore
Initial State: turn_token: "Agent-A", turn_number: 5, last_processed_turn: 4
Read game state
turn_token: "Agent-A", turn_number: 5, last_processed_turn: 4
Check passes: (token=="Agent-A") AND (5 > 4)
Start Atomic Transaction: acquire_turn_lock(5)
Inside Transaction:
1. Re-check last_processed_turn is still 4.
2. Update last_processed_turn to 5.
Transaction Success! Lock acquired.
Lock acquired, proceeds to execute take_turn(5)
Gunicorn restarts worker!
Read game state
turn_token: "Agent-A", turn_number: 5, last_processed_turn: 5
Check Fails: (token=="Agent-A") but (5 > 5) is FALSE!
Lock already taken. Goes back to sleep.
"Agent A "
"Agent A "
Firestore

Figure 8: Sequence diagram illustrating how the atomic lock prevents a race condition.

The sequence diagram in Figure 8 illustrates this solution.

6 Conclusion and Future Work

This work successfully demonstrates that the Agent Network System (ANS) [1] provides the necessary foundation for building complex, decentralised multi agent systems. The RISK [3] demo validates several key capabilities of ANS: dynamic discovery, secure A2A [2] communication, and robust, decentralised coordination. The emergent strategic behaviour observed in the agents' interactions serves as a powerful illustration of the potential of collaborative AI.

Future work could expand on this experiment by introducing heterogeneous agents or by exploring more complex negotiation protocols for forming multi agent coalitions.

References

[1] gClouds R & D, gLabs, Agent Network System (ANS) Whitepaper, Version 0.1.1. `https://g-clouds.github.io/ANS`

[2] Google, Agent-to-Agent Communication Protocol (A2A). `https://github.com/a2aproject/A2A`

[3] Hasbro, Inc. (2023). *RISK: The Game of Global Domination*. `https://www.hasbrorisk.com/en`

[4] Google Vertex AI. (2025). *Gemini Models: Gemini Flash 2.5*. `https://cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash`

[5] Firestore, scalable NoSQL cloud database. `https://firebase.google.com/docs/firestore`

[6] Flask, WSGI web application framework. `https://flask.palletsprojects.com/en/stable/`

[7] Gunicorn 'Green Unicorn', Python WSGI HTTP Server for UNIX. `https://gunicorn.org/`