Engineering Analysis of 2x2 Quad Systems for High-Security Bunker Penetration

The tactical landscape of high-level room sieges in the persistent world of Screeps is dominated by the conflict between concentrated defensive firepower and the resilience of multi-cellular offensive formations. At Room Control Level (RCL) 8, a defender possesses an arsenal of six towers, each capable of delivering significant damage with infinite range within the room, alongside reinforced ramparts that can withstand hundreds of millions of points of damage.[1] To penetrate these "bunker" layouts, the Senior Systems Architect must move beyond the logic of individual creeps and develop the Quad: a synchronized 2x2 squad that operates as a single high-performance entity. This report details the underlying mechanics, algorithmic requirements, and computational optimizations necessary to deploy top-tier offensive quads capable of breaking the most sophisticated RCL 8 defenses.

The Quad Movement Engine: Geometric Dilation and Pathfinding Integrity

The fundamental challenge of a 2x2 formation is that the game's native pathfinding engine, PathFinder, is designed to calculate routes for single-tile entities. A quad occupies a $2 \times$ grid of coordinates $(x, y), (x+1, y), (x, y+1), \text{ and } (x+1, y+1)$. Directly passing the current position of one creep to the pathfinder will lead to the quad becoming stuck as its trailing or adjacent members collide with walls or structures that the lead creep's path ignored.[4] To solve this, the architect must manipulate the PathFinder.CostMatrix using a technique known as geometric dilation.

Matrix Dilation Algorithms for Multi-Tile Entities

Geometric dilation involves pre-processing the room's terrain to represent the valid positions of the quad's "anchor" (typically the top-left creep). For every unwalkable tile at (x, y), the architect must mark four tiles as unwalkable in the quad's specific CostMatrix: $(x, y), (x-1, y), (x, y-1), \text{ and } (x-1, y-1)$.[4] This convolution ensures that if the anchor creep occupies any tile marked as "walkable" in the dilated matrix, all three other creeps are guaranteed to also be on walkable terrain.

Coordinate State	Native Terrain Cost	Dilated Matrix (for 2x2

		Anchor)
Wall at (x, y)	255	255
Plain at $(x -$	2	255
Plain at $(x, y -$	2	255
Plain at $(x - 1, y -$	2	255

This dilation must be applied to all static obstacles, including natural walls and player-constructed structures such as extensions, spawns, and labs. While PathFinder natively considers terrain, custom weights for structures must be manually injected into the CostMatrix within the roomCallback.[5] Senior architects often leverage libraries like Cartographer to manage these matrices, utilizing preTick() and reconcileTraffic() to ensure the quad remains cohesive while navigating complex base layouts.[6]

Dynamic Obstacle Management and Cost Thresholds

In a siege environment, the environment is rarely static. Enemy creeps will attempt to block the quad's movement, and ramparts may be destroyed during the engagement. The movement engine must differentiate between static terrain and volatile obstacles. Static obstacles are typically assigned a cost of 255 (impassable), while dynamic obstacles such as hostile creeps should be assigned a high but finite cost, such as 50 or 100, to encourage the pathfinder to route around them without completely failing if no other path exists.[7]

When the quad is within range of its target—often a specific rampart or spawn—the pathfinder must be tuned with maxOps and maxRooms to prevent CPU exhaustion. For cross-room sieges, the architect may use Game.map.findRoute() to identify the sequence of rooms, then apply the dilated CostMatrix logic once the quad enters the target room.[9] The precision of the path is critical; even a single-tick desynchronization where one creep moves and the others do not will break the 2x2 formation, immediately exposing the interior creeps to increased tower damage or melee attack.[10]

Kinetic Synchronization: Fatigue Modeling and Intent-Locking

A quad's operational speed is limited by the member with the highest fatigue. In the Screeps engine, fatigue is generated by every body part that is not a MOVE part (or an empty CARRY part) whenever a creep moves.[11] If a quad enters a swamp, the fatigue generation for each non-move part increases from 2 to 10. If even one member of the squad accumulates non-zero fatigue, it cannot move that tick, causing the quad to split unless a synchronization mechanism is in place.[12]

Collective Fatigue and the Pull Mechanism

To ensure lockstep movement, the architect must implement a centralized "Movement Lead" logic. Instead of each creep calling moveTo() independently, the lead creep (the anchor) calculates the path and issues move intents for the entire group. The most efficient way to achieve this is through the Creep.pull() method. In this configuration, the anchor creep pulls its followers. When one creep pulls another, the fatigue generated by the pulled creep is transferred to the puller, allowing the squad to move as a single physical unit with a shared fatigue pool.[15]

This transfer allows for specialized body compositions. An "Engine" creep can be built with an over-abundance of MOVE parts to compensate for "Heavy" creeps laden with WORK or HEAL parts. The collective fatigue of the quad is calculated by the sum of all parts against the sum of all MOVE parts.

$$F_{quad} = \sum_{i=1}^{4} (\text{BodyParts}_i - \text{MOVE}_i - \text{EmptyCARRY}_i) \times \text{TerrainFactor}$$

The reduction in fatigue per tick is:

$$R_{quad} = \sum_{i=1}^{4} \text{MOVE}_i \times 2$$

For 1-tile-per-tick movement, R_{quad} must be greater than or equal to F_{quad}.[11]

Swamp Dilation and Cost Mapping

To prevent the quad from accidentally entering a swamp with only one member—thereby slowing the entire group—the CostMatrix must dilate swamp costs similarly to walls. Any tile that is within the 2x2 footprint of a swamp must be marked with the swamp cost in the dilated matrix.[4] This ensures the PathFinder correctly evaluates the cost of the move for the *entire squad*. If the anchor creep moves to (x, y), and $(x+1, y+1)$ is a swamp, the pathing cost for (x, y) in the dilated matrix must reflect the swamp penalty.

Terrain Type	Plain Cost	Swamp Cost	Road Cost

Standard Matrix	2	10	1
2x2 Dilated Matrix	2	10 (if any member hits swamp)	1

This ensures that the quad will prefer longer paths on plains over shorter paths that clip a swamp, maintaining maximum velocity throughout the approach to the bunker.[6]

Defensive Resilience: Predictive Healing and Damage Mitigation

An RCL 8 bunker represents the most hostile environment a creep can inhabit. Six towers can provide up to 3,600 damage per tick if the quad is within five tiles of the tower cluster.[1] Surviving this requires not just high healing throughput, but predictive healing logic that accounts for the defender's tower AI.

Tower Damage Falloff and Attack Logic

Tower damage is not uniform; it decreases linearly as the distance from the tower to the target increases. The falloff starts at range 6 and reaches its minimum at range 20.[2]

Target Range	**Damage per Tower**	**Total Damage (6 Towers)**
$\leq$	600	3,600
10	450	2,700
15	300	1,800
$\geq$	150	900

Most defenders use simple tower logic, such as findClosestByRange(FIND_HOSTILE_CREEPS) or targeting the creep with the lowest health.[10] A Senior Systems Architect will reverse-engineer the defender's tower AI by observing which quad member is targeted and then adjusting the healing focus accordingly.

The Mechanics of Overhealing and Pre-healing

In Screeps, intents are processed in a specific order: heal actions are generally prioritized, but the damage is applied in the same tick. There is a historical debate regarding "overhealing"—the ability to heal a creep beyond its hitsMax to "buffer" damage. While the game engine restricts hits to hitsMax, pre-healing is essential for survival.[19] If the architect knows that Creep A will be targeted by 3,600 damage this tick, and Creep A currently has 5,000/5,000 hits, the quad's healers should still target Creep A. This ensures that as soon as the damage intent is processed, the heal intents are also processed, minimizing the health deficit at the end of the tick.[19]

The triage algorithm should be:

1. **Damage Prediction:** For each quad member, calculate the sum of potential damage from all towers in the room.
2. **Health Deficit Calculation:** Identify members whose current hits + predictedHeal - predictedDamage is less than hitsMax.
3. **Triage Allocation:** Distribute the quad's total HEAL parts among the members. Priority is given to members who are likely to be "alpha-striked" (killed in a single tick).
4. **Boost Synergy:** Boosted TOUGH parts are critical. A T3 boosted TOUGH part (catalyzed ghodium acid) reduces incoming damage by 70%, effectively tripling the creep's hit points.[10]

Part Type	**Unboosted**	**T3 Boosted (Effect)**
HEAL (Range 1)	12 HP	48 HP (4x effectiveness)
HEAL (Range 3)	4 HP	16 HP (4x effectiveness)
TOUGH	100 Hits	100 Hits (0.3x damage taken)
WORK (Dismantle)	50 Hits	200 Hits (4x effectiveness)

1

Structural Fluidity: Formation Transitions for Chokepoint Navigation

RCL 8 bunkers are often designed with "honeycomb" structures or 1-wide corridors specifically intended to obstruct quads.[22] A rigid 2x2 movement engine will fail at these chokepoints. Top-tier bots must implement "Snake" or "Worm" formations, allowing the quad to transition into a 1x4 line and back again.

The Snake Transition Algorithm

When the quad identifies a path that requires 1-wide movement (detected by a failure in the 2x2 dilated CostMatrix but success in a standard CostMatrix), it must trigger a formation shift.

- **Lead-Follower Serialization:** The quad designates a leader (Creep 1). Creep 2 is instructed to move to Creep 1's position on the previous tick. Creep 3 moves to Creep 2's previous position, and Creep 4 follows Creep 3.[24]
- **Coordinate Buffering:** The system maintains a FIFO (First-In, First-Out) queue of the last four positions occupied by the lead creep. Each follower is assigned an index in this queue. This prevents the "clumping" that occurs if all creeps try to move to the leader's current position simultaneously.[24]
- **Formation Re-Entry:** Once the lead creep reaches an area with a $2 \times$ clearance, it stops. The followers continue moving along the coordinate buffer until they are adjacent to the leader in their designated "home" quadrants (Top-Left, Top-Right, etc.). Only once the 2x2 formation is restored does the movement engine resume its dilated pathfinding.[4]

Comparative Formation Analysis

Attribute	2x2 Quad	1x4 Snake
Healing Proximity	Optimal (All adjacent to 2 others)	Poor (Only adjacent to 1-2 others)
Pathfinding Clearance	2-wide (Restrictive)	1-wide (Flexible)
Tower Resistance	High (Concentrated cross-heal)	Low (Tail is vulnerable)
Melee Surface Area	Minimal (External faces only)	High (Easily surrounded)

[1]

The 1x4 formation is a vulnerability. While in "Snake" mode, the quad's total healing capacity is fragmented. Healer A can only reach Healer B if they are adjacent; if the snake is stretched across a corridor, the tail cannot be healed by the head.[10] Consequently, transitions should only be performed when absolutely necessary and ideally when outside of tower range.

Advanced Siege Doctrine: Breaking the Bunker Economy

Breaking an RCL 8 bunker is often a battle of economic attrition rather than a single decisive strike. Towers consume 10 energy per action, and in a prolonged siege, a defender can be "drained" of their energy reserves.[2]

The Attrition Cycle

A Senior Systems Architect uses quads to tax the defender's economy. By moving the quad to range 15–20 from the towers, the quad takes moderate damage (1,800–900 per tick) which it can easily out-heal with T3 boosts.[2] Each shot the tower takes costs the defender energy. If the quad can force the towers to fire for hours, the defender's storage will eventually empty.

- **Draining Strategy:** Position the quad where tower damage is low but constant. Ensure the quad has enough energy or a logistics tail to maintain its own healing.[10]
- **Economic Taxing:** While the towers are busy, the quad can use RangedMassAttack to damage nearby ramparts. This forces the defender to spend even more energy on repair intents, further accelerating the drain.[2]
- **AI Exploitation:** If the defender's AI only refills towers when they are below 50% energy, a quad can wait for the "lull" in fire during the refill cycle to move closer and dismantle a key structure.[10]

Managing Safe Mode and Downgrade Timers

Defeating the buildings is only the first step. To truly break a bunker, the architect must account for the Room Controller's defensive mechanics. An RCL 8 room has a downgrade timer of 150,000 ticks and may have multiple Safe Mode activations.[23]

- **Safe Mode Counter:** Safe Mode can be triggered to stop an attack instantly. However, it cannot be activated if the controller is being attacked by a CLAIM part creep or if the downgrade timer is below 50%.[3]
- **Downgrade Pressure:** Use a specialized "Controller Quad" with CLAIM parts to use attackController(). This resets the upgrade-blocked timer and prevents the defender from "Safe Moding" to recover.[25]
- **Nuke Synergy:** If a defender activates Safe Mode, a nuke can be used to end it early, or the quad can retreat and wait for the 1,000-tick timer to expire before immediately re-engaging.[25]

Computational Efficiency: Optimizing Quad Logic for High GCL Bots

As a bot scales to dozens of rooms, the CPU cost of managing quads becomes a limiting factor. A quad requires complex pathfinding, intent calculation for four creeps, and high-frequency memory updates.[28]

CPU and Memory Trade-offs

The Senior Architect must balance between "Game Efficiency" (getting things done) and "CPU Efficiency" (using fewer cycles).[28] Quads are inherently "Game Efficient" but "CPU Expensive."

Optimization Target	Technique	Result
CPU Efficiency	Cache paths and dilated CostMatrices.	Reduces PathFinder calls but increases memory usage.[5]
Memory Efficiency	Serialize CostMatrices to strings/segments.	Reduces JSON parsing time but increases complexity.[5]
Heap Management	Avoid creating new objects in the loop.	Minimizes Garbage Collection (GC) pauses which can spike CPU.[29]

Large bots with 100+ rooms of vision face "Heap Pressure." Game objects (creeps, structures) occupy significant space in the Isolated-VM heap. If a quad's logic creates too many temporary arrays or objects per tick, it can trigger frequent GC events, potentially leading to "Out of Memory" (OOM) errors and bot crashes.[29]

Traffic Management and Bipartite Matching

In a crowded home room, a 2x2 quad moving toward an exit can be blocked by hundreds of smaller "Horde" creeps.[7] Simple shoving logic is insufficient. Advanced bots use the Bipartite Matching Problem (BMP) algorithm to resolve traffic.

This algorithm treats creeps and their potential positions as a graph. It searches for "augmenting paths" that allow the maximum number of creeps to move to their desired positions.[33] For a quad, this means the algorithm must treat the four tiles of the quad as a single atomic unit. If a hauler is in one of the quad's future tiles, the hauler must be moved to a tile that is not occupied by any of the four quad members in the next tick.[33]

Tactical Case Study: The Breach of a Standard "Bunker" Layout

To illustrate the integration of these systems, consider a quad attempting to breach a standard RCL 8 bunker. The bunker features a 2-wide rampart perimeter, six towers in a central core, and multiple "kill zones" with 1-wide entry points.

Phase 1: The Approach

The quad enters the room and calculates a dilated path to a "weak spot" in the rampart—often a corner where tower damage is slightly lower due to range.[2] The movement engine uses creep.pull() to ensure all members move in unison, even if the tail clips a swamp at the room's edge.[4]

Phase 2: The Transition

The quad encounters a 1-wide "zig-zag" entrance designed to break formations. The state machine triggers the "Snake" formation. Creep 1 (the lead) moves into the corridor, while Creeps 2, 3, and 4 follow the coordinate buffer.[24] During this phase, the triage logic shifts to rangedHeal() to account for the increased distance between members.[1]

Phase 3: The Breach

Once through the corridor, the quad reforms into its 2x2 square. It begins using T3-boosted WORK parts to dismantle the ramparts.[1] The triage logic detects focus-fire on Creep 2 from all six towers and directs all 2,880 HP of the squad's healing toward Creep 2.[19] Because Creep 2 has T3 TOUGH boosts, it only takes 1,080 damage, allowing the healers to keep it at full health while the dismantlers work.[10]

Phase 4: Termination

With the ramparts gone, the quad moves into the core. It prioritizes the spawns and the terminal to disrupt the defender's ability to create new defenders or buy more energy.[10] Finally, the quad moves to the controller to begin the 150,000-tick downgrade countdown, ensuring the room cannot be easily reclaimed.[25]

Summary of Engineering Requirements for Quad Implementation

The successful implementation of quads requires a multi-layered architectural approach. It is not enough to simply group four creeps; the bot must manage their geometric footprint, their physical momentum, and their collective survival.

System Layer	Primary Mechanism	Critical Success Factor
Pathfinding	Dilated CostMatrix	Accurate representation of $2 \times$ footprint.
Kinetic	creep.pull() and shared fatigue	Net-zero fatigue through MOVE part

		over-provisioning.
Resilience	Predictive Triage and T3 Boosts	Ghodium Acid (XGHO2) and Lemergium Acid (XLHO2) saturation.
Structural	Coordinate Buffer and State Machine	Seamless transition between 2x2 and 1x4 modes.
Strategic	Economic Attrition and Controller Blocking	Integration of CLAIM parts to prevent Safe Mode.
Optimization	Serialized Matrices and BMP Traffic	Minimizing CPU and Heap impact at scale.

Deployment of these systems represents the transition from a mid-game player to a top-tier empire. The complexity of the quad engine reflects the complexity of the Screeps game itself—a world where the most efficient code, rather than the most numerous army, ultimately dictates the boundaries of territory. By adhering to the principles of dilation, synchronization, and predictive triage, a Senior Systems Architect can create an offensive force that is mathematically capable of breaking any bunker.

Works cited

1. screeps-bot-choreographer/MATH.md at main · ryanrolds/screeps ..., accessed February 19, 2026, https://github.com/ryanrolds/screeps-bot-choreographer/blob/main/MATH.md
2. StructureTower - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/StructureTower/
3. Defending your room | Screeps Documentation, accessed February 19, 2026, https://docs.screeps.com/defense.html
4. Quad (high volume creep attack) · Issue #69 · JonathanSafer/screeps - GitHub, accessed February 19, 2026, https://github.com/jordansafer/screeps/issues/69
5. docs/api/source/PathFinder.CostMatrix.md at master · screeps/docs - GitHub, accessed February 19, 2026, https://github.com/screeps/docs/blob/master/api/source/PathFinder.CostMatrix.md
6. screeps-cartographer, accessed February 19, 2026, https://glitchassassin.github.io/screeps-cartographer/
7. Screeps #25: Arena - Grouping Up - Field Journal, accessed February 19, 2026, https://jonwinsley.com/notes/screeps-arena-grouping-up

8. Moving between rooms | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/2992/moving-between-rooms
9. Pathfinding - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/Pathfinding/
10. Offensive Strategy : r/screeps - Reddit, accessed February 19, 2026, https://www.reddit.com/r/screeps/comments/7d6w2t/offensive_strategy/
11. How MOVE part work？ | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/1314/how-move-part-work
12. Creeps - Screeps Documentation, accessed February 19, 2026, https://docs.screeps.com/creeps.html
13. How does each bodypart affect fatigue? | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/3044/how-does-each-bodypart-affect-fatigue
14. Screeps Arena Documentation, accessed February 19, 2026, https://arena.screeps.com/docs/
15. creep.pull() doesn't appear to work how the documentation suggests... : r/screeps - Reddit, accessed February 19, 2026, https://www.reddit.com/r/screeps/comments/1onjahj/creeppull_doesnt_appear_to_work_how_the/
16. Cartographer is an advanced (and open source) movement library for Screeps - GitHub, accessed February 19, 2026, https://github.com/glitchassassin/screeps-cartographer
17. Tower Damage - forum - Screeps, accessed February 19, 2026, https://screeps.com/forum/topic/922/tower-damage
18. If you struggle with the Tower - look here! :) :: Screeps: World Help - Steam Community, accessed February 19, 2026, https://steamcommunity.com/app/464350/discussions/5/1523906480775299994/
19. Overhealing vs. strictly healing past damage: fixing inconsistencies based off of intent order | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/319/overhealing-vs-strictly-healing-past-damage-fixing-inconsistencies-based-off-of-intent-order
20. Overhealing vs. strictly healing past damage: fixing inconsistencies based off of intent order | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/319/overhealing-vs-strictly-healing-past-damage-fixing-inconsistencies-based-off-of-intent-order/18
21. Combat - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/Combat/
22. Screeps Part 20 – Bunkers - Adam Laycock, accessed February 19, 2026, https://alaycock.co.uk/2017/10/screeps-part-20-bunkers
23. Getting Started - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/Getting_Started/
24. Design Snake Game: Movement Logic & Algorithm Explained - YouTube, accessed February 19, 2026, https://www.youtube.com/watch?v=dZF0zseLcY0
25. PvP Game discussion | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/473/pvp-game-discussion/15

26. PvP Game discussion | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/473/pvp-game-discussion/28
27. PvP Game discussion | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/473/pvp-game-discussion/35
28. Optimization - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/Optimization/
29. Need for Speed - Endgame optimization limitations | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/298/need-for-speed-endgame-optimization-limitations
30. Saving memory or saving cpu limit? : r/screeps - Reddit, accessed February 19, 2026, https://www.reddit.com/r/screeps/comments/o0abxr/saving_memory_or_saving_cpu_limit/
31. Remove the CPU cost of having memory and scale available memory akin to CPU | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/760/remove-the-cpu-cost-of-having-memory-and-scale-available-memory-akin-to-cpu
32. IVM heap usage & game objects | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/2163/ivm-heap-usage-game-objects
33. Journey to Solving the Traffic Management Problem - Harabi Screeps, accessed February 19, 2026, https://sy-harabi.github.io/Journey-to-Solving-the-Traffic-Management-Problem/