

Client-Side Patching Ensures a Bright Future

2026-03-21 · Updated 2026-07-28

#Programming #generals.io #polygen

Client-side patching is exactly what I need for the low bandwidth of my server.¹

Why Patching Overhead Matter

We are playing on a normal 40×40 FFA map with 8 players. Given a player covering 20% of the map, for example in late-game battles. Then how much traffic is spent on transmitting map updates?

The workload is undoubtedly acceptable if only cities and generals are involved - that's usually the case - but the problem arises when it comes to incrementing empty lands.

Typically you would expect a sudden burst of patches at the beginning of each round. In this hypothetical scenario, about 320 lands get updated from the perspective of the player. Thanks to generals.io's compression-based patching, we are free from sending everything over the web.

A sequence of unchanged lands are compressed into a single number, indicating how many positions we shall skip when processing a diff. Actually, it works exceptionally well in large maps where lands tend to be more consecutive.

Before a patch is transmitted, it's serialized into JSON arrays and become a part of a WebSocket frame. Since the frame is plain UTF-8 on most servers (including generals.io), we are able to use the length of JSON string to estimate its size.

To simplify our calculation, let's just assume a land ends up 5 UTF-8 characters in JSON on average. That's a relatively fair estimation, taking the width of land amounts, skipping points and JSON separators (,) into account.

Therefore, the approximate payload size for the player is

$$320 \text{ land} \times 5 \text{ ASCII-Character/land} \times 1 \text{ B/ASCII-Character} = 1.6 \text{ KB.}$$

Suppose there are two more players fighting and the other five spectating in late-game. Our total outbound traffic during this turn is (don't forget spectators receive a full patch!)

$$1.6 \text{ KB} \times 3 + 1.6 \text{ KB} \times 3 \times 5 = 22.8 \text{ KB.}$$

This overhead scales rapidly with more players, larger maps, faster speeds, fancier modifiers, multiple rooms and simultaneous games, ultimately evolving into a real trouble for:

- Users with poor Internet connection;
- Servers with low outbound bandwidth;
- Admins who pay on their resource usage.

Current Patching Strategies

We have investigated several generals-like games and their patching strategies. Let V be the size of the map, P be the number of players in the game. As described above, we are interested in the worst-case network complexity of patching for a single player.

¹<https://github.com/By-Ha/Checkmate/pull/11>

Game	Strategy	Complexity
generals.io	compression-diff	$O(V)$
Gennia *	compression-diff	$O(V)$
By-Ha/Checkmate *	diff	$O(V)$
oimasterkafuu/checkmate	diff	$O(V)$

Table 1: Different generals-like games and their patching strategies.

*: Website is down.

In most games $P \ll V$. Chances are that we can make this overhead V -independent to greatly improve its performance.

Our Solution

The idea is deadly simple: a majority of patching stuff can be offloaded to the client side. When a new turn begins, the client first runs a *partial* natural update on its own *view*. The natural update works identical to what it does on the server: iterating through the whole map to find generals/cities/lands/swamps and update their amounts.

At each turn, the server executes a *dry* natural update for the *real* map, i.e. , it produces a patch rather than modifying the map. To keep client view synchronized, the server holds a copy of, and applies the *same* patch to each *team*'s map view.

After that, the patch is merged into the *real* map. Regular game mechanisms take effect normally on the server, including player movements and modifiers, directly operating on the *real* map.

Finally, the server *masks* the *real* map for each *team*, comparing it with corresponding *view* and returning the diff to the client.

Now we feel very optimistic about patch sizes: natural updates stay the same even after masking, so it will not end up in the patch at all! As a result, the patch contains nothing but updates caused by player movements, bounded by the maximum player count ($O(P)$).

Wait, what if one captures another? Well, although it's still $O(V)$ worst-case, captures cannot appear repeatedly as $t \rightarrow +\infty$, which is guaranteed by game logics. In other words, their contributions are definitely negligible.

By means of eliminating natural updates from map diff, we could achieve an averaged $O(P)$ network complexity in the overhead of per-player patching. More specifically, the payload size for a single player has shrinked $\sim 98\%$ to

$$3 \text{ player-move} \times 2 \text{ land/player-move} \times 5 \text{ ASCII-Character/land} \times 1 \text{ B/ASCII-Character} = 0.03 \text{ KB.}$$

Realworld Application

This patching strategy has already been implemented in polygen² and has dramatically reduced our overall I/O footprint. We believe this approach is both simple and effective, production-ready for your generals-like games, whether you are working on the official site or building a custom server.

²<https://codeberg.org/polygen/polygen>