Let us consider simple ETH transfers from A to B as a base case. Suppose that we want each shard to have the same transactions-per-second as the mainnet. On the mainnet, the gas limit is 8m, and a simple transaction costs 21000 gas, so we have a maximum 380 transactions per block. In a shard, a period lasts for 5 main chain blocks, so shard blocks may have 1900 transactions. Each transaction requires a witness for the sender account and the recipient account, including a Merkle branch for both, as well as the code for both.

Let us assume that each shard has as many accounts as the current mainnet - roughly 24m, which we can round up to 2^25 for simplicity, so we will assume that a Merkle branch has ~32 bytes * 25 * 1.1 for overhead = 880 bytes (this is already assuming the Merkle branches are stored in a format that does not duplicate data by fully including the hash of an object which already appears elsewhere in the data; standard binary Merkle trees do this implicitly, but it can be done for Merkle Patricia trees as well). Let's assume that the code of an account is 200 bytes long.

Naively, we might calculate the witness size as 1080 * 2 * 1900 = 4.10 MB, and the base transaction size would be ~110 * 1900 = 209 kB, so this bumps up the size of a block by about 20x. But we can already do a simple optimization: merge the Merkle branches for the different addresses. Between 3800 addresses, chances are that the top 11 rows of the tree will be fully used up, so we can replace 32 * 1.1 * 25 * 3800 with 32 * 1.1 * 14 * 3800 + 2047: savings of about 1.5 MB.

We can make a clearer estimate with this code:

```
import random

def calctot(length, branches): o = [0] * (2 ** length + 1) for i in range(branches): pos = 1 for j in range(length): o[pos] = 1 pos = pos * 2 + random.randrange(2) return int(len([x for x in o if x == 1]) * 35.2)
```

This gives ~1.76 MB for 3800 branches, and adding account code gives 2.5 MB, confirming the rough estimate.

Now, let us look at the multi-state root proposal. Instead of having a single state root, each shard might have 256 state roots, one for accounts starting with 0x00, one for accounts starting with 0x01, etc. In a naive model, each sub-shard would only have 2^17 accounts, so the data size would go down to 3.0 MB (1 MB savings). But what happens if you take into account witness merging? Each sub-state root would only have ~14.85 accounts, so total data consumption (sum([m.calctot(17, 14 if random.random() > 0.85 else 15) for i in range(256)])

as an approximation) gives ~1.76 MB. In short, multi-state roots give negligible additional savings.

This is explained by the fact that, in blocks with 3800 transactions, the top 8 rows of the Merkle tree are referenced so many times that they are there basically there "for free".

We could make further 25% gains by reducing the Merkle tree width from 32 bytes to 24 bytes; this would cut us down to ~2.08 MB. We could also save more for simple sends by adding an opcode that sends money without running the destination account's code, so we would only need the sender account's code for each transaction and not the destination's; this saves 200 * 1900 = 380 kb; in total, this gets us down to 1.7 MB witness size - only a 9x increase in bandwidth from statelessness.

The above is a maximum-entropy scenario, and pretty close to worst case. In the average case, we can get even larger gains. If a single address sends or receives multiple transactions (eg. because it's making many sends, or because it's a popular contract), then the entire branch for that address would be reused many times. The protocol could also have a protocol rule where publishing contracts in some address range (eg. 0x0000) is much more expensive in terms of initial setup cost, but then cheaper to include as a witness; this would encourage reuse of this range for some more limited set of popular contracts.

In the longer term, multisig transactions, post-quantum signatures, ringsigs, range proofs, SNARKs, etc, are both more space-intensive on the transaction side, but not on the witness size, so the bandwidth increase from statelessness will not be even as large as 9x.

Also, most contracts as currently written have a fairly large size, often significantly larger than 1 KB. If we want to make inroads on reducing witness size there, then there is basically only one logical path: breaking up the code into functions and making it into a Merkle tree; though even there, many complex transactions will execute functions that are a few hundred bytes long or even longer.