

Rec Room Networking 101

This doc tries to explain Rec Room networking - from scratch - in a way that is both comprehensible and useful to people who want to create room/inventions.

This doc assumes zero knowledge of Rec Room networking, or of networking generally.

Goals

1. After reading this doc, you have a basic but accurate understanding of how Rec Room networking works
2. You can use this understanding to design, improve and optimize your creations
3. You learn some key shared terminology
4. You are not intimidated by complexity
5. Nor are you misled by oversimplified, inaccurate, or incomplete information.

How it works

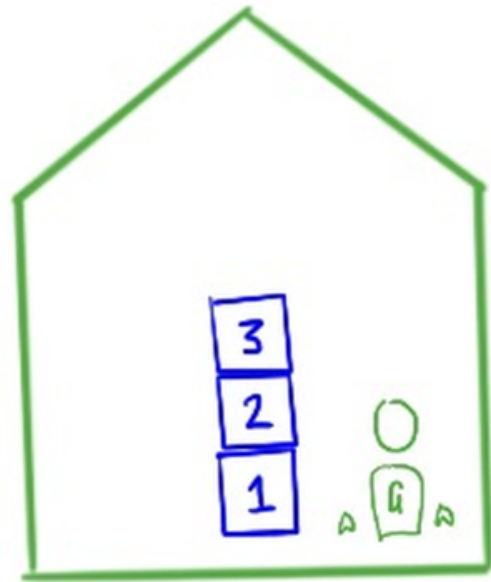
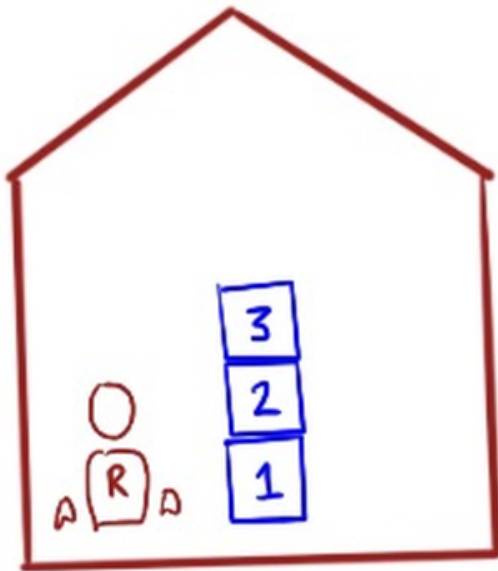
We are going to explain Rec Room networking from scratch using a metaphor!

Let's start by imagining two players, R and G. They are in identical but separate rooms:



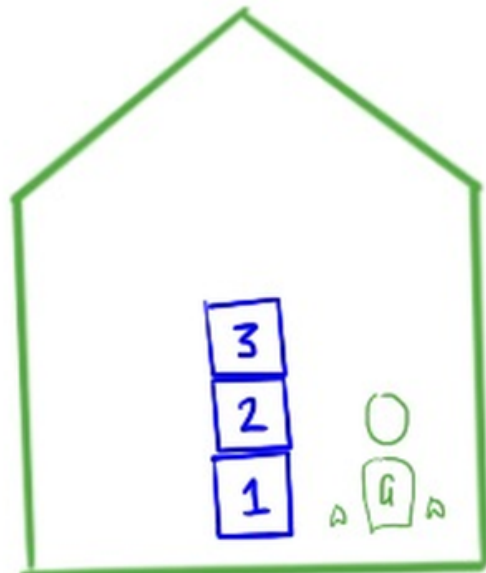
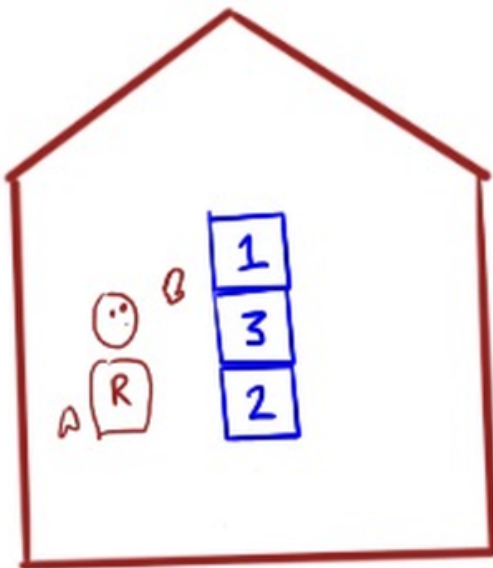
- R cannot see into G's room
- G cannot see into R's room

Let's give them something to play with. We give both R and G an identical set of three blocks:

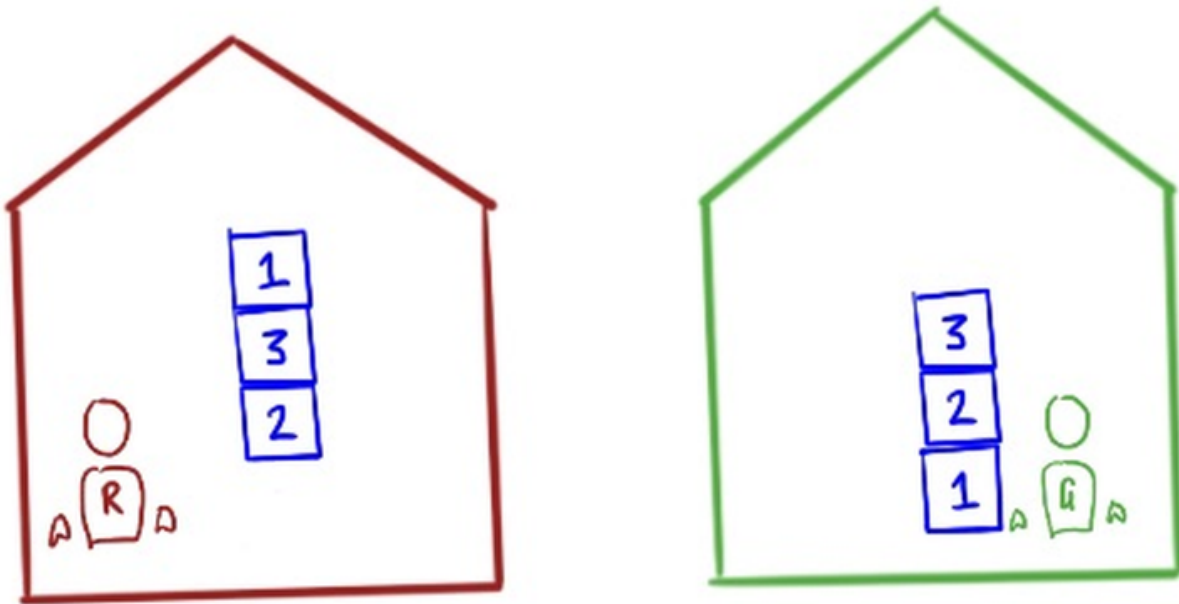


Now let's set them a challenge! They can play with the blocks however they like, but **they must keep their rooms "in sync"**. Meaning that whatever happens to a block in one of the rooms must happen in the other room as well. If, after an hour, we check their rooms and find the blocks identically arranged, we'll give them an ice cream!

Let's say R moves a block:

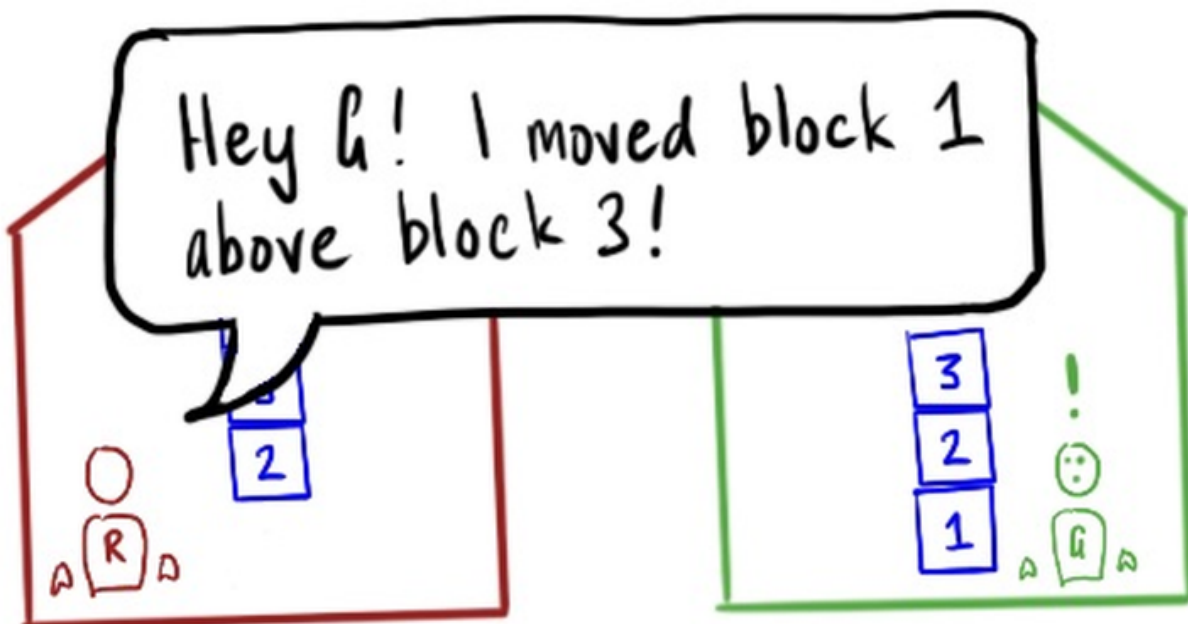


Uh oh. Now their rooms are out of sync:

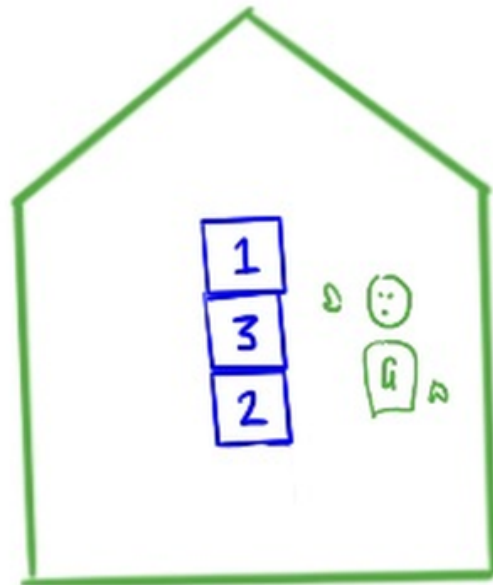
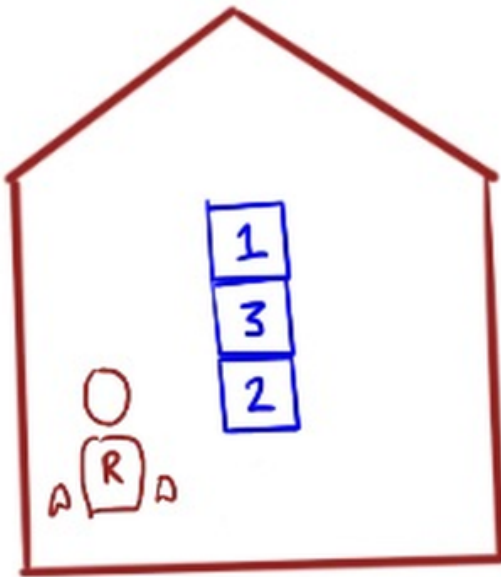


Block 1 in R's room is in a different location than block 1 in G's room. Unless they do something about it, no ice cream!

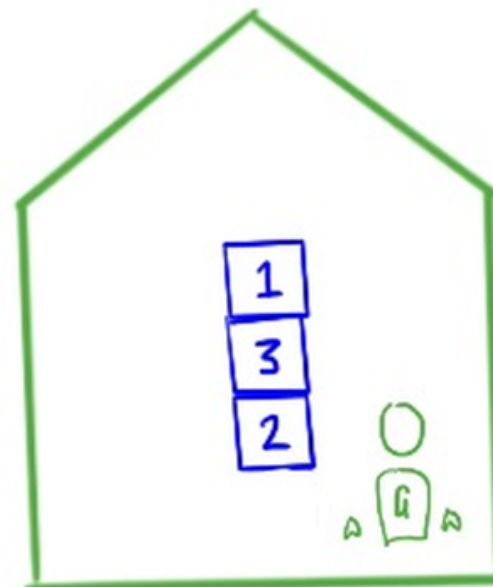
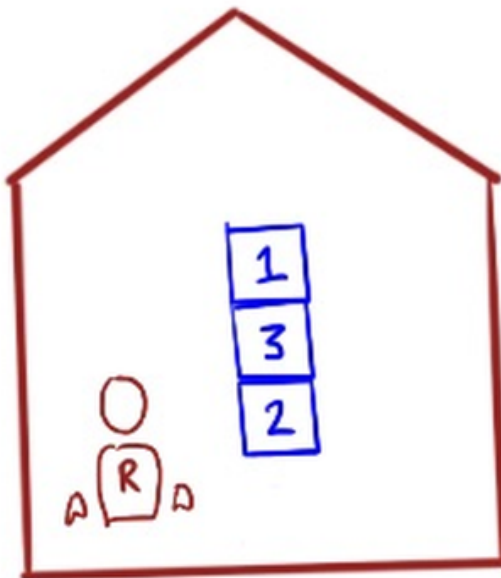
R can't see into G's room, but there's nothing stopping them from yelling. So they yell over to G what they just did:



G hears R's message, and moves their block to match:

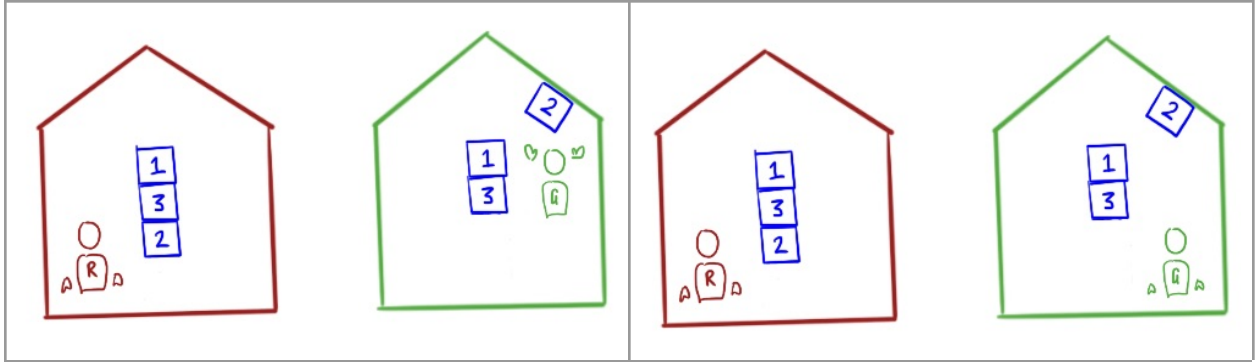


Nice! The rooms are back **in sync**:



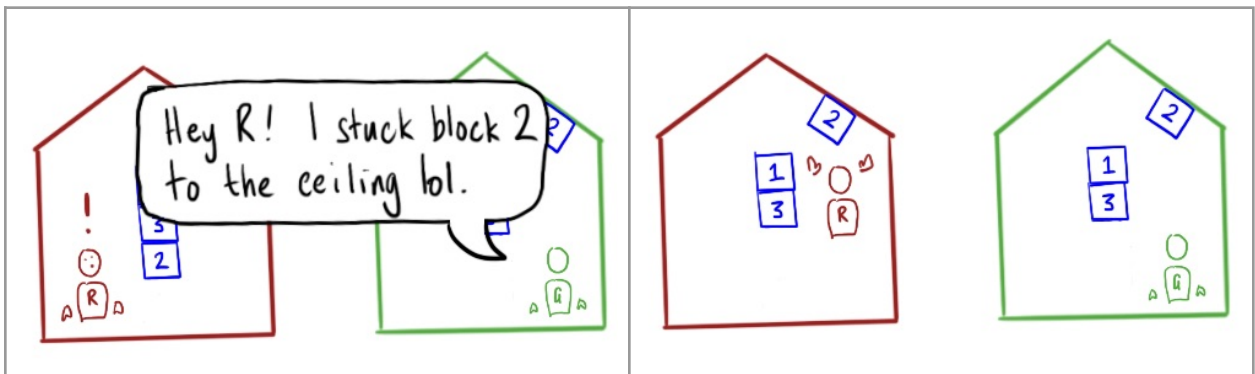
Maybe they'll get that ice cream after all!

Of course, the same process works in reverse:



G moves block 2...

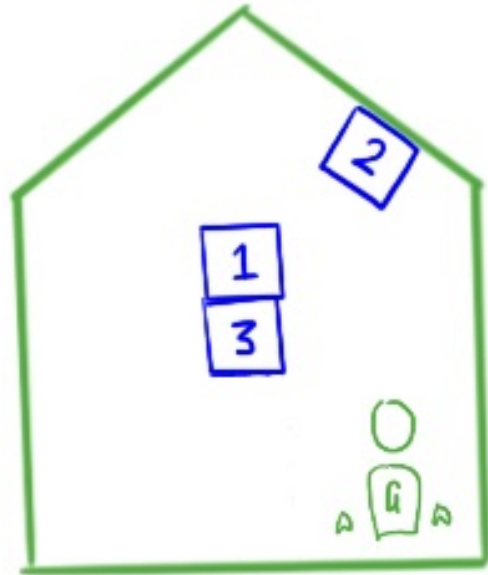
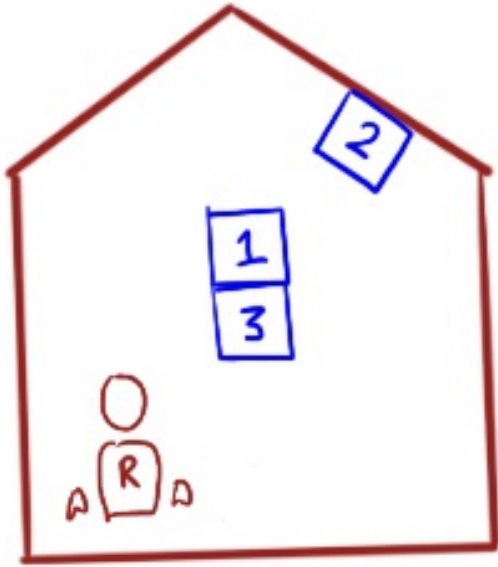
...meaning the rooms are out of sync again!



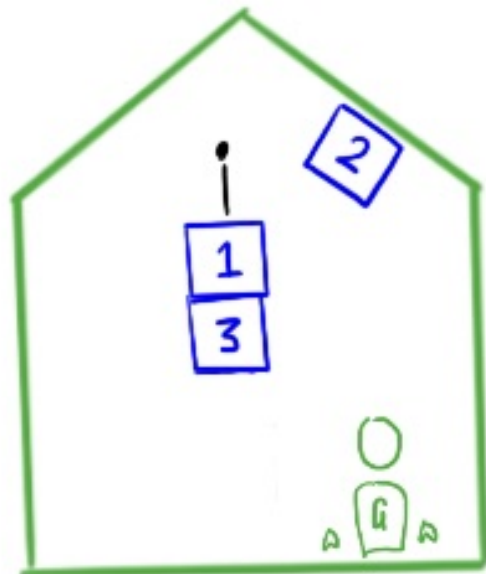
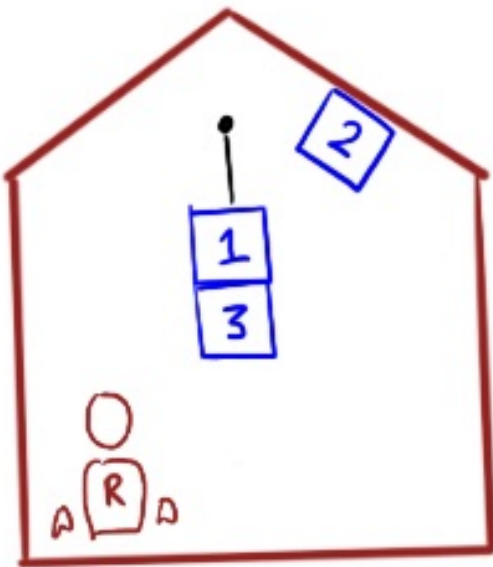
G yells a message to R...

...R hears the message and moves block 2

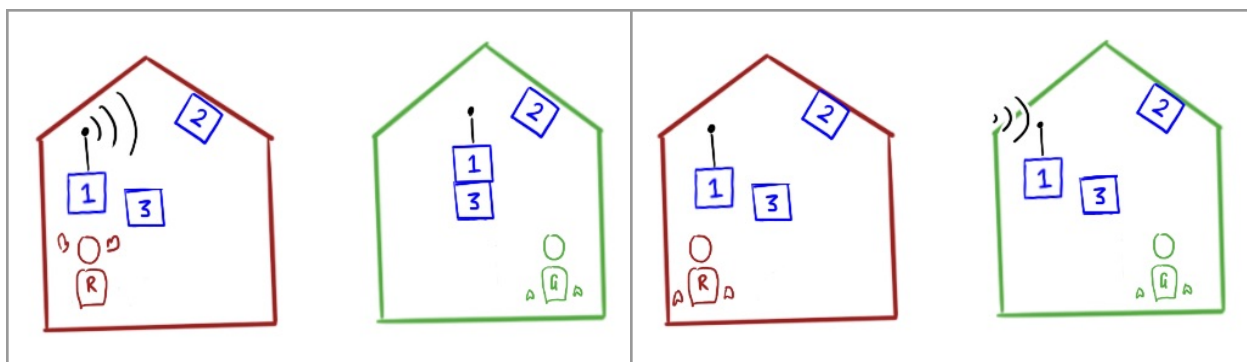
And we're back in sync again:



OK this is working pretty well, but it's a lot of yelling and manual labor. Let's give R and G a hand by upgrading the blocks. We're going to add a *magic antenna* to each block:



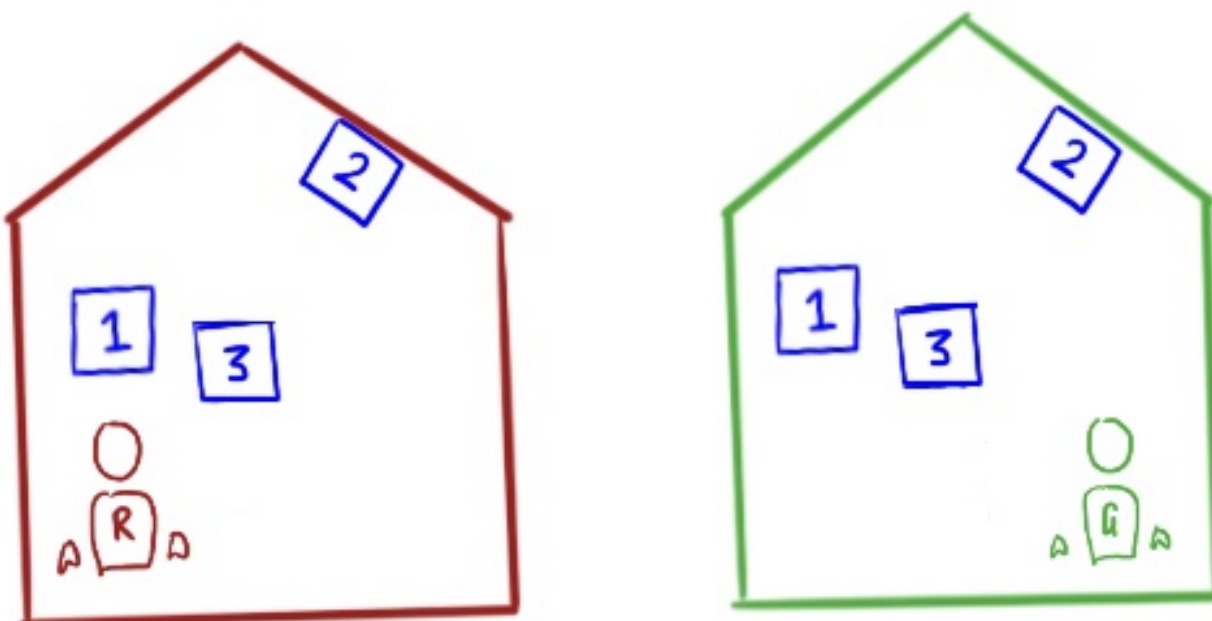
Now, when a block gets moved it can broadcast the change to its counterpart in the other room:



R moves block 1, causing it to broadcast a message...

...which is received by block 1 in G's room. G's block 1 moves itself to the new position.

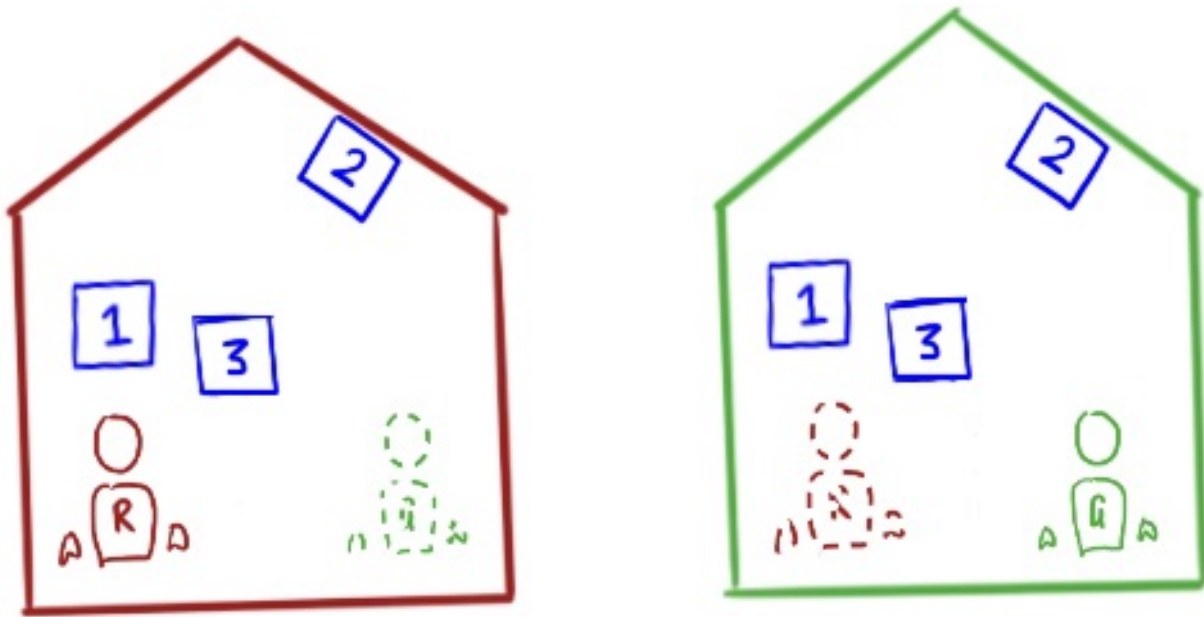
And we're back in sync again!



Thanks magic antenna!

Every object in the room has a magic antenna. We could put a 100 blocks in the room, and as they got moved around they would broadcast changes to the other room, and everything would stay in sync.

Let's take a quick break from blocks, and use the antenna for another purpose. By attaching antennas to their head and hands, we can create a synced copy of each *player* in the other player's room:

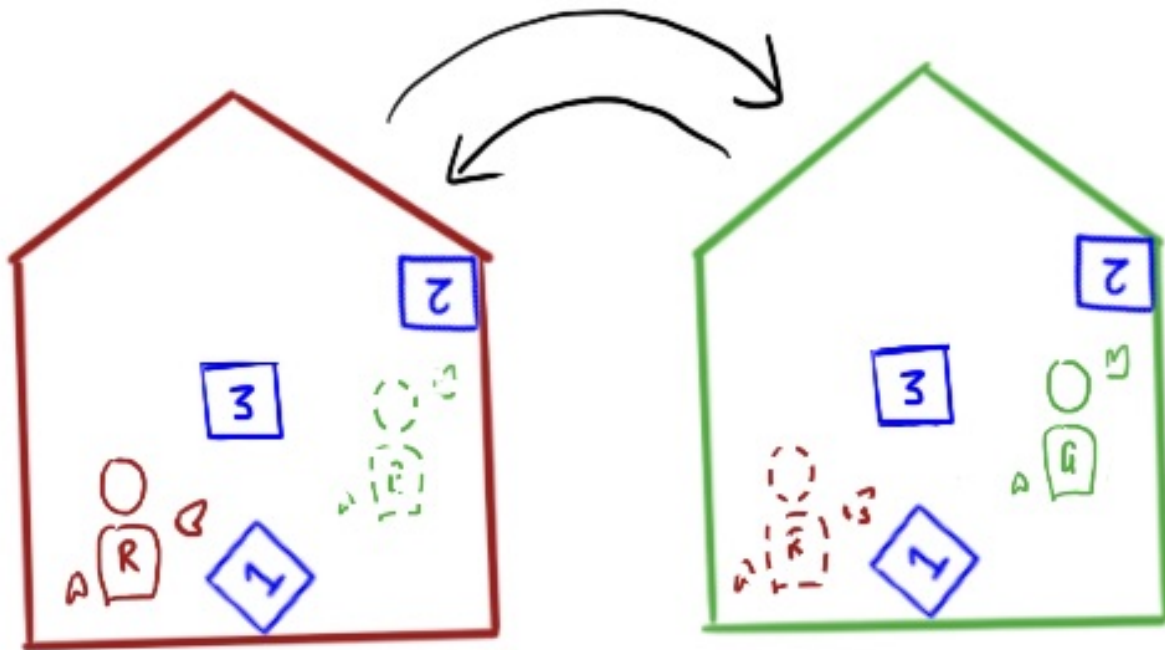


Here we see a copy of G in R's room, and a copy of R in G's room. We use magic antenna signals to make the copies follow the movements of the actual players. So if G waves, R sees the copy of G wave:



Magic antennas are pretty efficient, so the player copy moves pretty naturally - it's easy for R to believe that G is actually in the room with them, and vice versa.

With these upgraded blocks (and the ability to see each other), R and G can go nuts moving the blocks around. Everything stays in sync just fine, with the room exchanging a constant stream of "antenna data" (object sync updates):



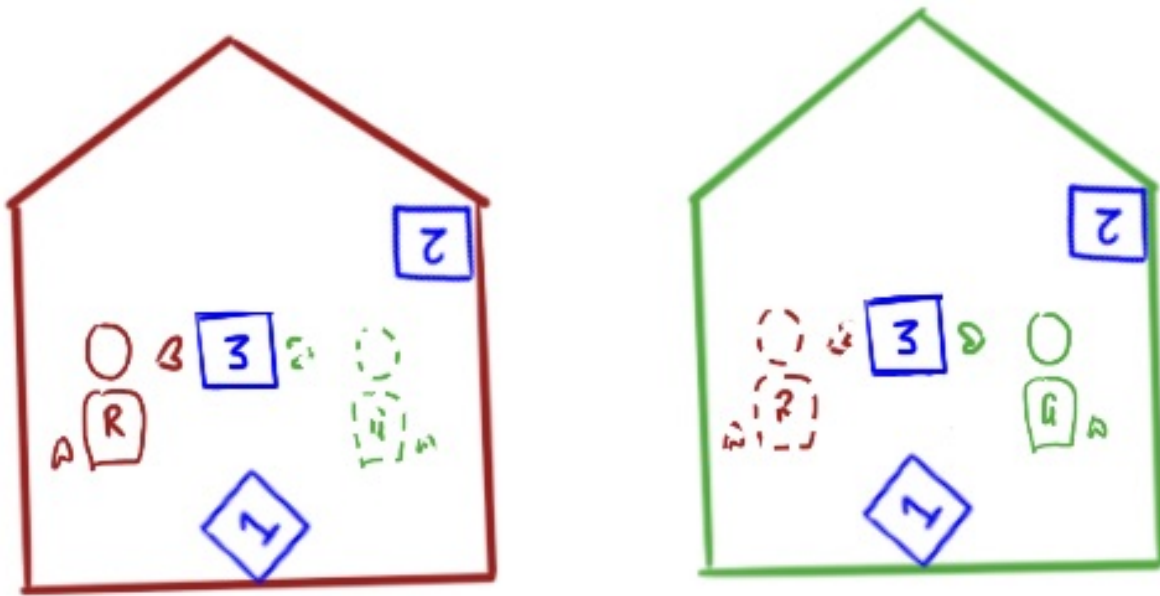
Let's pause here to say: This is basically how RR networking works "under the hood":

- The two rooms represent each player's **local session**. E.g., you can imagine that Player R is running a session on an Oculus Quest, and player G is running their session on an iPhone.
- The "magic antenna" are the networking messages we send back and forth over the internet to keep everything in sync.
- The blocks are the various objects in the room (e.g., a basketball, or a frisbee, or an object you created with the Maker Pen)

With that foundation, let's go back to the analogy to explore an important concept...

Authority

Everything is going swimmingly until... oops! Both R and G reach for block 3 at the same time:

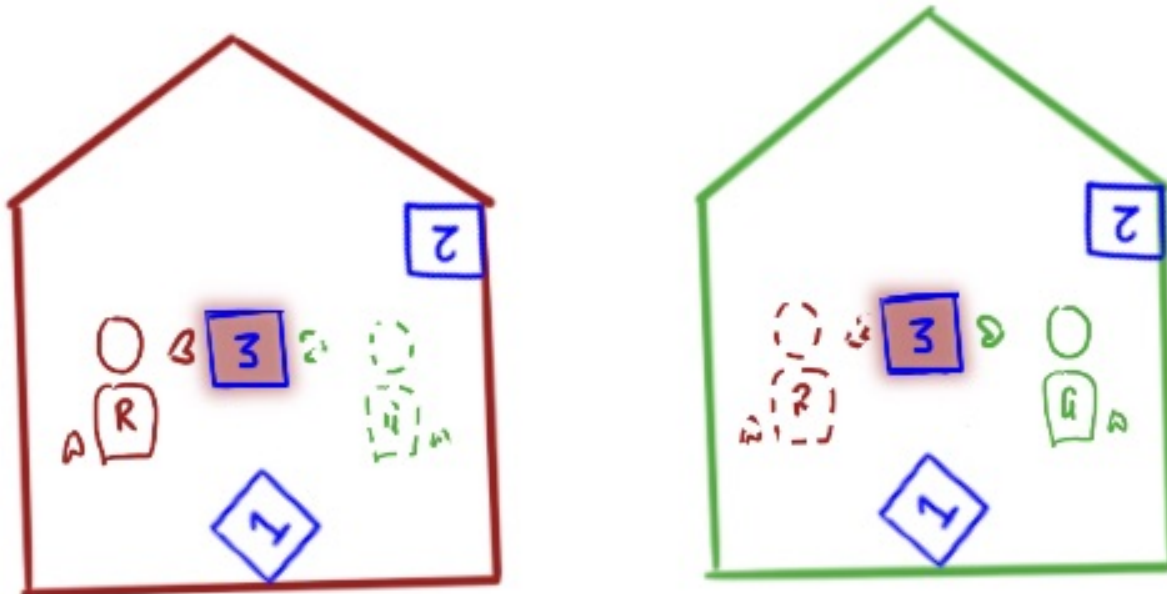


Now what happens? They can't *both* grab block 3, because that would desync the room. Let's say we let R move block 3 one meter to the left, and G move it one meter to the right. To get back in sync, we'd have to throw away someone's move.

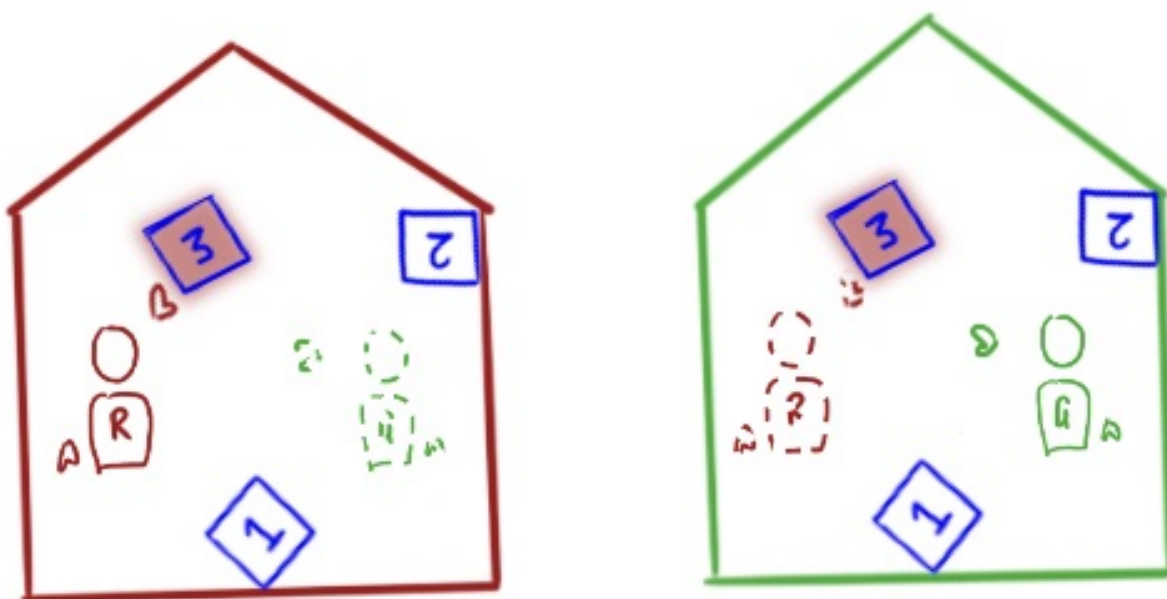
Authority

To resolve this, we introduce the idea of **authority**. Only one player at a time has the authority to move a block.

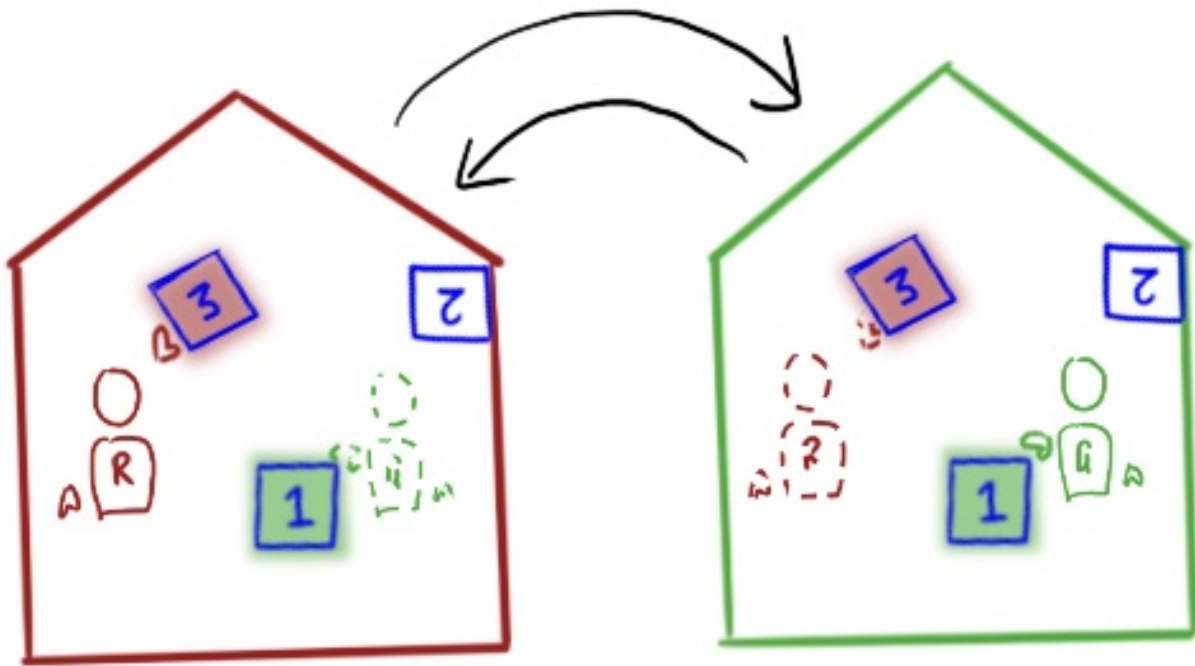
In this case we'll give authority to R (don't worry about *why* we gave it to R, we'll talk about that later). Let's turn block 3 red to show that it "belongs" to R at the moment:



Notice that the block is red in both rooms - R and G agree that R **has authority** over block 3. Now R can move it around, and the changes are synced to G's room as before (via block 3's magic antenna!):

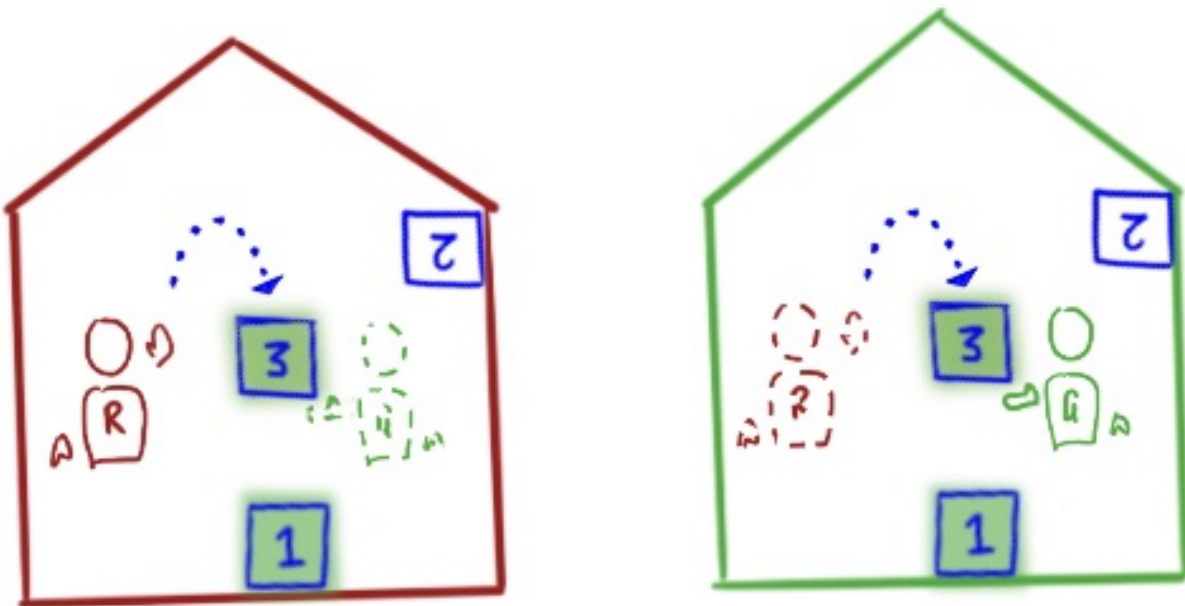


Authority is per object. So at the same time that R has authority over block 3, G can take authority over block 1. Now that we can guarantee that only one player is changing a block at a time, R and G can go back to messing around. The movement of the blocks, and of their avatars, generates a constant stream of sync updates back and forth:



Authority can change. Let's say R throws block 3 to G...

- Before the throw, R has authority over block 3
- As block 3 sails through the air, R retains authority (since no-one else has touched it)
- As soon as G catches it, authority is transferred to G:



Notice also that G still has authority over block 1, which they just dropped on the floor. Here are the general rules for how authority works:

- When you interact with something, you “take authority” over it. You are now responsible for keeping everyone else updated about this object.
 - “Interact” generally means “touch or grab it, or cause it to move or change for any reason”. If you grab a basketball, you take authority over it. When you throw it, you keep authority until someone else touches it. You can also take authority indirectly (e.g., an object you have authority over bumps into other objects... that will transfer your authority to those objects as well).
- Authority is “sticky”, once you’ve taken authority over something you keep until someone else interacts with it
- Tie-breaks are handled by the order in which messages are sent. So in the case where both R and G are trying to grab block 3, in practice there will be a slight time difference in when they grab it - even if it’s the tiniest fraction of a second. The tie-breaking rule is “whoever grabbed later gets it”. In practice, you don’t really need to worry about this, the system handles it for you to ensure that only one person can take authority of any given object.

Room Authority

One last - and very important thing about authority - is the answer to this question. “Let’s say no-one has interacted with an object yet. Who has authority over that object?”

The answer is that there is a default owner - one of the players is designated as the “**Room Authority**”.

The room authority player has authority over all objects by default. This is overridden on a per-object basis as players interact with specific objects.

So if R is room authority, by default they have authority over all three blocks:

R = 3 objects

G = 0 objects

As soon as G interacts with a block, their authority overrides room authority:

R = 2 objects

G = 1 object

...and so on.

Q: What happens when the player with room authority leaves the room?

Room authority is transferred seamlessly (we hope!) to another player.

Q: How do we choose which player has room authority?

It's a heuristic based on:

- Device type (e.g., PC takes precedence over Quest)
- Ping time
- Maybe some other stuff =]