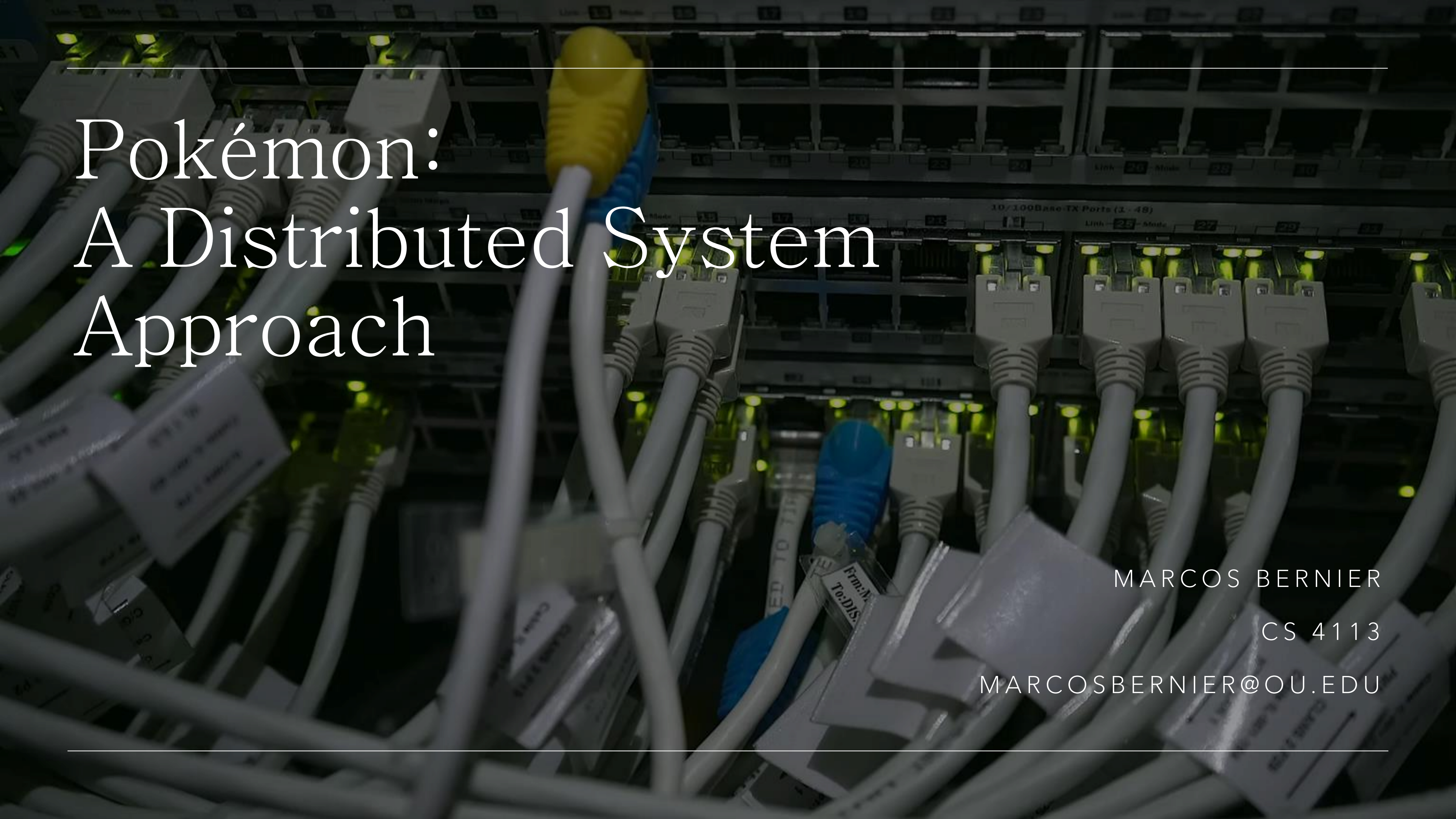


Pokémon: A Distributed System Approach

MARCOS BERNIER

CS 4113

MARCOSBERNIER@OU.EDU

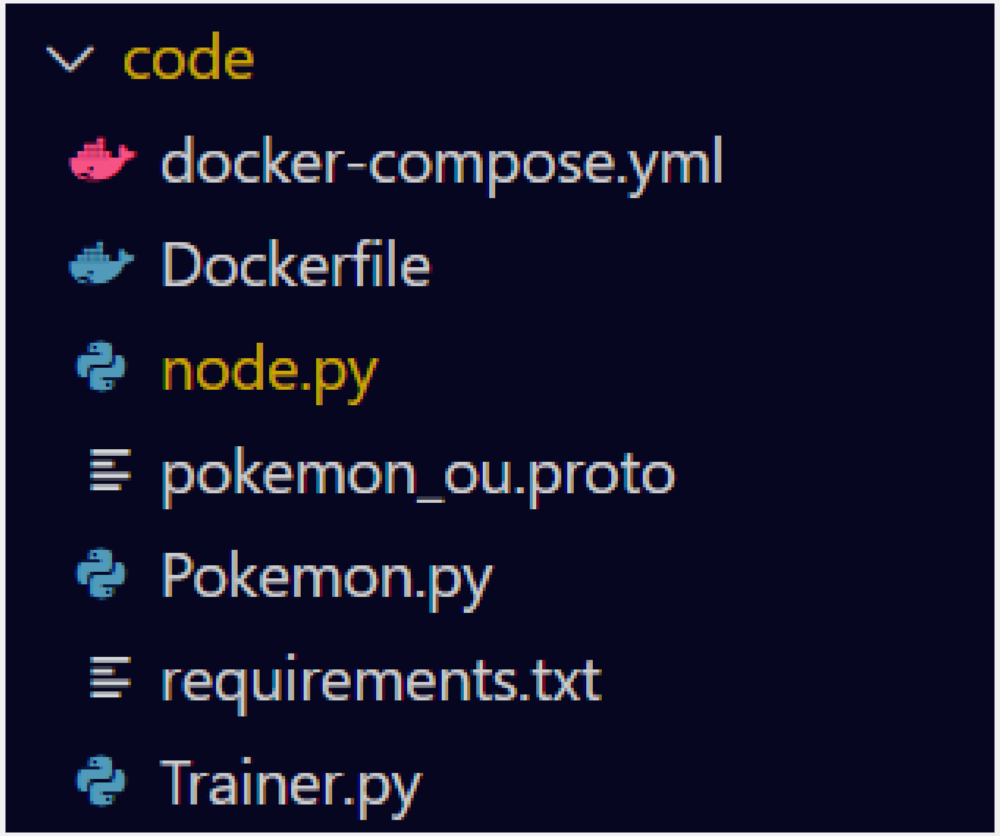
Project Requirements

Tasked with creating a distributed system that

- Allows “Pokemon” and “Trainer” nodes to connect
- Maintains an NxN representation of a board or map
- Allowed Pokemon and Trainer nodes to receive a unique name and starting space on the board
- Allows (in a managed manner) for Pokémon and Trainers to move across the board
- Allows Trainers to capture Pokémon (and subsequently end their game)

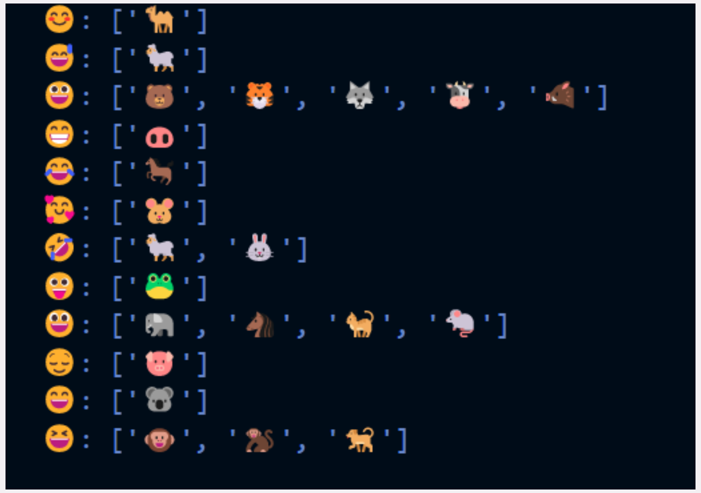
File Structure Overview

- Docker-Compose
  - Defines the 3 types of entities, Server, Trainer, and Pokemon
- Dockerfile
  - Defines the containers that will run the Python files
- Node.py
  - Contains the running server instance (and directs other instances to be run as necessary)
- Pokemon_ou.proto
  - Contains all gRPC function and message definitions for the server to work
- Pokemon.py and Trainer.py
  - Contain the logic for Pokemon and Trainer entities
- Requirements.txt
  - Empty



```
code
  docker-compose.yml
  Dockerfile
  node.py
  pokemon_ou.proto
  Pokemon.py
  requirements.txt
  Trainer.py
```

Other Files



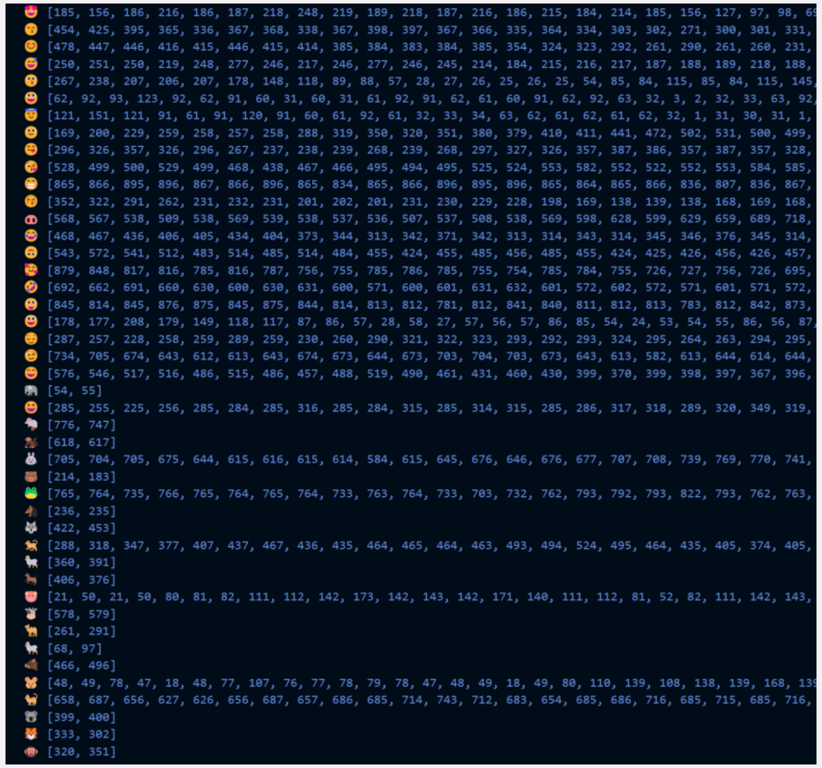
```
😊: ['🐪']
😅: ['🦙']
😃: ['🐻', '🐯', '🐺', '🐮', '🐗']
😁: ['🐽']
😂: ['🐎']
🥰: ['🐹']
🤣: ['🦙', '🐰']
😛: ['🐸']
😀: ['🐘', '🐴', '🐈', '🐁']
😌: ['🐷']
😄: ['🐨']
😆: ['🐵', '🐒', '🐅']
```

Pokedex.txt:

Contains all Trainers' names and their respective catches on game end

Path.txt:

Contains all nodes' names and respective move histories on game end



```
🤩 [185, 156, 186, 216, 186, 187, 218, 248, 219, 189, 218, 187, 216, 186, 215, 184, 214, 185, 156, 127, 97, 98, 6
😙 [454, 425, 395, 365, 336, 367, 368, 338, 367, 398, 397, 367, 366, 335, 364, 334, 303, 302, 271, 300, 301, 331,
😊 [478, 447, 446, 416, 415, 446, 415, 414, 385, 384, 383, 384, 385, 354, 324, 323, 292, 261, 290, 261, 260, 231,
😅 [250, 251, 250, 219, 248, 277, 246, 217, 246, 277, 246, 245, 214, 184, 215, 216, 217, 187, 188, 189, 218, 188,
😗 [267, 238, 207, 206, 207, 178, 148, 118, 89, 88, 57, 28, 27, 26, 25, 26, 25, 54, 85, 84, 115, 85, 84, 115, 145,
😃 [62, 92, 93, 123, 92, 62, 91, 60, 31, 60, 31, 61, 92, 91, 62, 61, 60, 91, 62, 92, 63, 32, 3, 2, 32, 33, 63, 92,
😇 [121, 151, 121, 91, 61, 91, 120, 91, 60, 61, 92, 61, 32, 33, 34, 63, 62, 61, 62, 61, 62, 32, 1, 31, 30, 31, 1,
🙂 [169, 200, 229, 259, 258, 257, 258, 288, 319, 350, 320, 351, 380, 379, 410, 411, 441, 472, 502, 531, 500, 499,
😋 [296, 326, 357, 326, 296, 267, 237, 238, 239, 268, 239, 268, 297, 327, 326, 357, 387, 386, 357, 387, 357, 328,
😘 [528, 499, 500, 529, 499, 468, 438, 467, 466, 495, 494, 495, 525, 524, 553, 582, 552, 522, 552, 553, 584, 585,
😁 [865, 866, 895, 896, 867, 866, 896, 865, 834, 865, 866, 896, 895, 896, 865, 864, 865, 866, 836, 807, 836, 867,
😚 [352, 322, 291, 262, 231, 232, 231, 201, 202, 201, 231, 230, 229, 228, 198, 169, 138, 139, 138, 168, 169, 168,
🐽 [568, 567, 538, 509, 538, 569, 539, 538, 537, 536, 507, 537, 508, 538, 569, 598, 628, 599, 629, 659, 689, 718,
😂 [468, 467, 436, 406, 405, 434, 404, 373, 344, 313, 342, 371, 342, 313, 314, 343, 314, 345, 346, 376, 345, 314,
🙃 [543, 572, 541, 512, 483, 514, 485, 514, 484, 455, 424, 455, 485, 456, 485, 455, 424, 425, 426, 456, 426, 457,
🥰 [879, 848, 817, 816, 785, 816, 787, 756, 755, 785, 786, 785, 755, 754, 785, 784, 755, 726, 727, 756, 726, 695,
🤣 [692, 662, 691, 660, 630, 600, 630, 631, 600, 571, 600, 601, 631, 632, 601, 572, 602, 572, 571, 601, 571, 572,
😛 [845, 814, 845, 876, 875, 845, 875, 844, 814, 813, 812, 781, 812, 841, 840, 811, 812, 813, 783, 812, 842, 873,
😀 [178, 177, 208, 179, 149, 118, 117, 87, 86, 57, 28, 58, 27, 57, 56, 57, 86, 85, 54, 24, 53, 54, 55, 86, 56, 87,
😌 [287, 257, 228, 258, 259, 289, 259, 230, 260, 290, 321, 322, 323, 293, 292, 293, 324, 295, 264, 263, 294, 295,
😉 [734, 705, 674, 643, 612, 613, 643, 674, 673, 644, 673, 703, 704, 703, 673, 643, 613, 582, 613, 644, 614, 644,
😄 [576, 546, 517, 516, 486, 515, 486, 457, 488, 519, 490, 461, 431, 460, 430, 399, 370, 399, 398, 397, 367, 396,
🐘 [54, 55]
😆 [285, 255, 225, 256, 285, 284, 285, 316, 285, 284, 315, 285, 314, 315, 285, 286, 317, 318, 289, 320, 349, 319,
🐁 [776, 747]
🐒 [618, 617]
🐰 [705, 704, 705, 675, 644, 615, 616, 615, 614, 584, 615, 645, 676, 646, 676, 677, 707, 708, 739, 769, 770, 741,
🐻 [214, 183]
🐸 [765, 764, 735, 766, 765, 764, 765, 764, 733, 763, 764, 733, 703, 732, 762, 793, 792, 793, 822, 793, 762, 763,
🐴 [236, 235]
🐺 [422, 453]
🐅 [288, 318, 347, 377, 407, 437, 467, 436, 435, 464, 465, 464, 463, 493, 494, 524, 495, 464, 435, 405, 374, 405,
🦙 [360, 391]
🐎 [406, 376]
🐷 [21, 50, 21, 50, 80, 81, 82, 111, 112, 142, 173, 142, 143, 142, 171, 140, 111, 112, 81, 52, 82, 111, 142, 143,
🐮 [578, 579]
🐪 [261, 291]
🦙 [68, 97]
🐗 [466, 496]
🐹 [48, 49, 78, 47, 18, 48, 77, 107, 76, 77, 78, 79, 78, 47, 48, 49, 18, 49, 80, 110, 139, 108, 138, 139, 168, 13
🐈 [658, 687, 656, 627, 626, 656, 687, 657, 686, 685, 714, 743, 712, 683, 654, 685, 686, 716, 685, 715, 685, 716,
🐨 [399, 400]
🐯 [333, 302]
🐵 [320, 351]
```

Problem 1 : How to handle Deadlock?

Problem:

Because the nature of movement in the game is asynchronous, it allows for the potential conflict of spaces when, say, two Trainers attempt a move to the exact same space. Without attempting to resolve the issue, the result of this operation would mean that both nodes would complete the move, but one of those nodes would be overwritten by the most recent of the two – breaking the game integrity.

Solution:

Use threading locks! When a node wants to move, it will send the proposed new location to the server in a request for the move to be performed. The server will lock the new space (and current space) for changes and subsequently verify that the move can be performed – then it will perform it and release the locks.

Problem 2 : Constantly changing states?

Problem:

If nodes can move asynchronously (without timing delays), then how do we create an environment where a Trainer could actually see and capture a Pokémon on a space? Additionally, how do we create an environment where a Pokémon can successfully evade capture before its too late?

Solution:

We use board checks as a buffer to avoid these circumstances, and continue to use thread locks to avoid any issues. By requiring nodes to ask for a list of possible moves from the server, and then having them pick from the given lists and specifically request a move, we allow a Trainer to become aware they are moving to a space where a Pokémon was when they checked. However, in the time between the move and the capture, the Pokémon has an opportunity to move before the board is locked.

Demonstration and Further Walkthrough

The End!