Sockets are faster but need a Protocol

- Protocol
  - They require some sort of protocol when you pass the messages yourself.

Example:

- Tron is a light cycle game. Every tick of time has the client send over an int. The protocol is:

Turn left =	**0**
Turn right =	**1**
Straight =	**2**

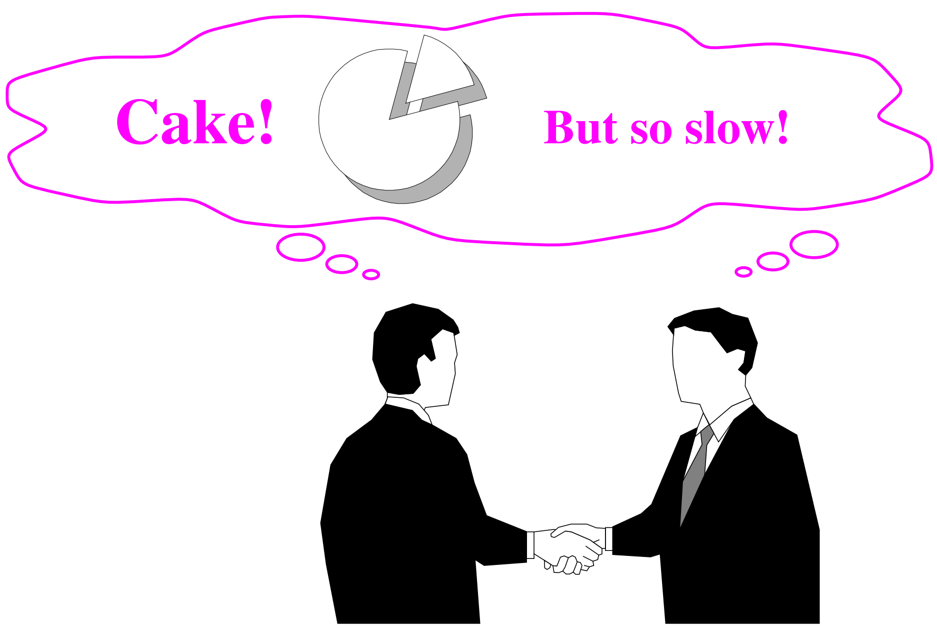
- When the server or other client gets the message from the tron client it knows how to update its world view (in this case the position of the light cycle).
- Protocols can (and often do) get much more complex and detail intensive.

Order of Events

• All IPC forms have one thing in common

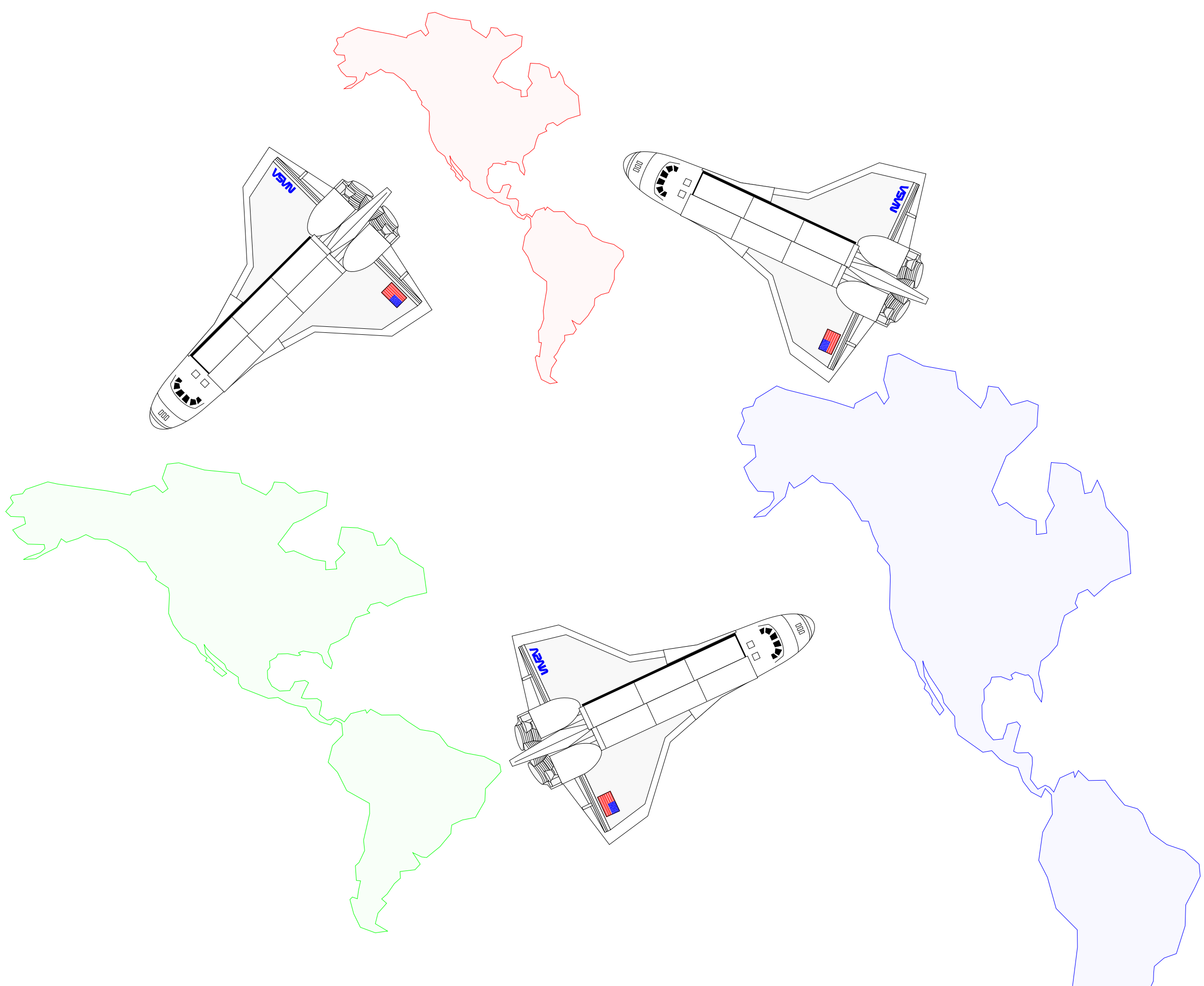
•When a client makes a change it must have a way of notifying everyone else that the change was made

- •RPC - the clients make remote calls to the server
- •Nice because it is easier to debug and the server does not need to handle who is calling what. It can update all when it makes a call.
- • Lot of net traffic this way.



Dead Reckoning

- Take the all client model & make it work
    - Massive communication
    - Allowing inconsistencies
    - Multi-casting
    - This gives the player more freedom

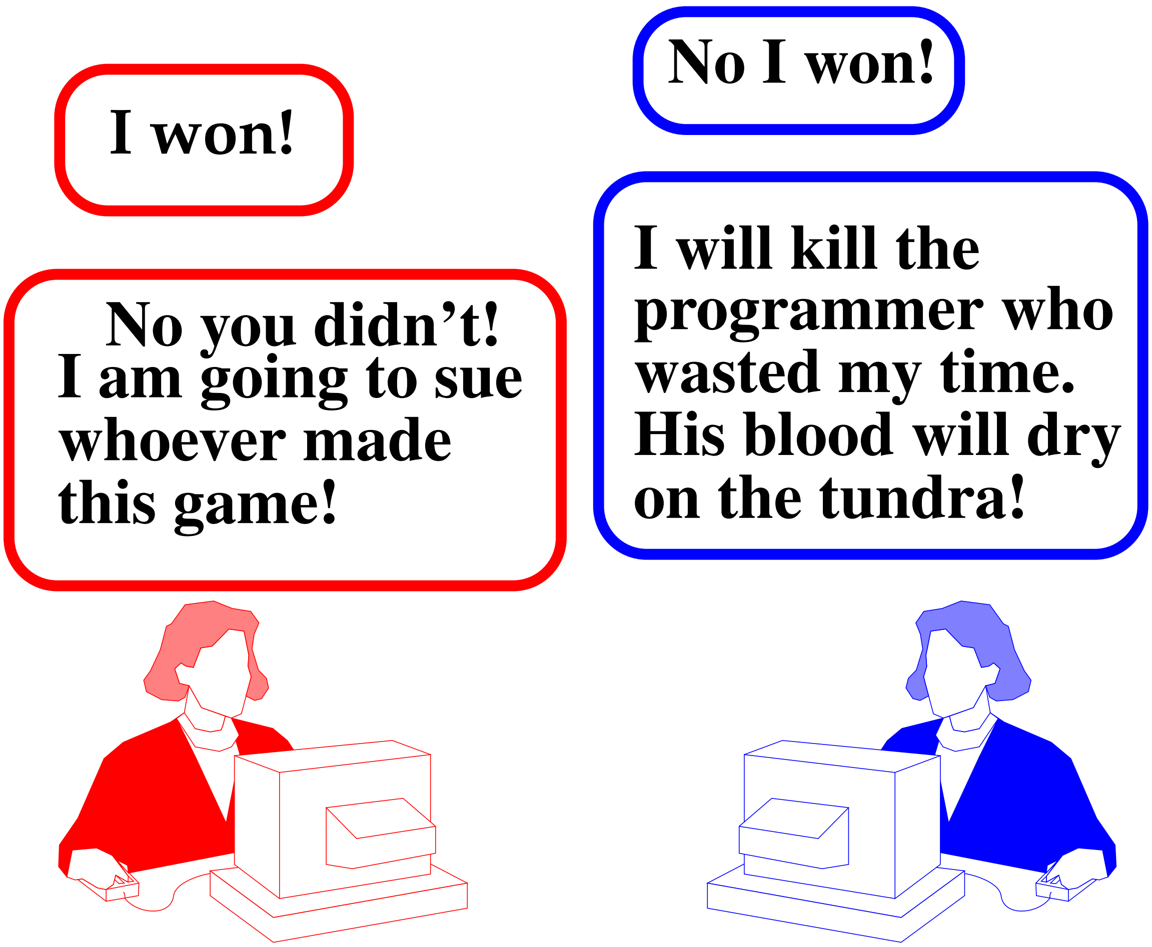


- Needs a priority system
    - Can be linear or N^2

Very Playable!

No Server lots of clients

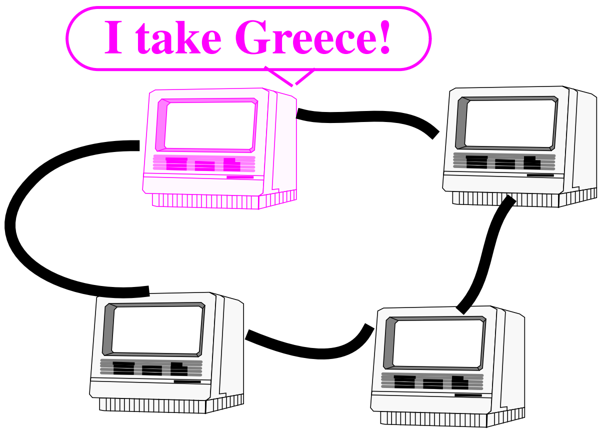
- Every client is responsible for their own world.
  - Latency and network delay mean that there is no way possible to share the same world

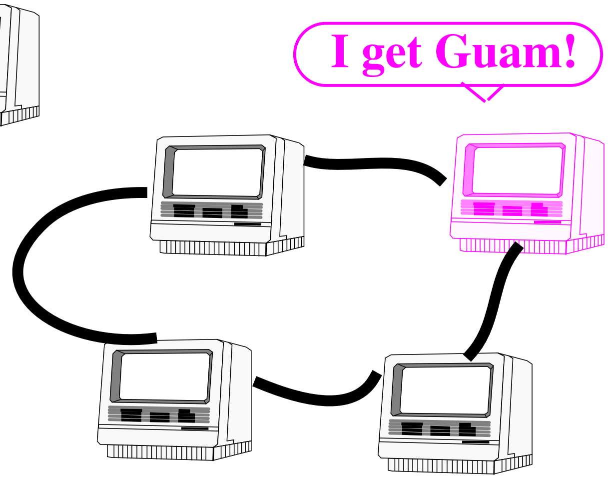


- Each client is their own world
  - Very fast
  - Don't drop frames

Round-Robin

- The server moves from one client machine to another while they all make their moves.
- Risk could be like this

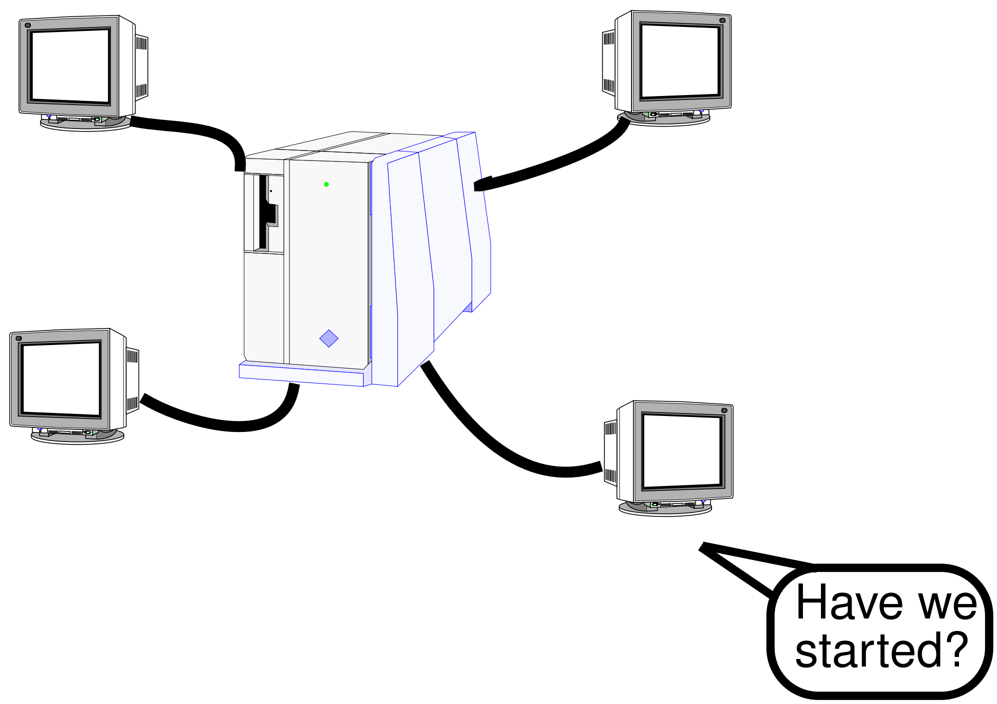




- This is very consistent
  - Only one person changes the world at a time
  - No one else can do anything.
  - Might be slow and not fit for worlds where everyone moves at once (Doom, MUDS, Muppet Movie)

Star

A server can hold all of the world view on one machine and have all of the clients notify it of changes. Like a star.

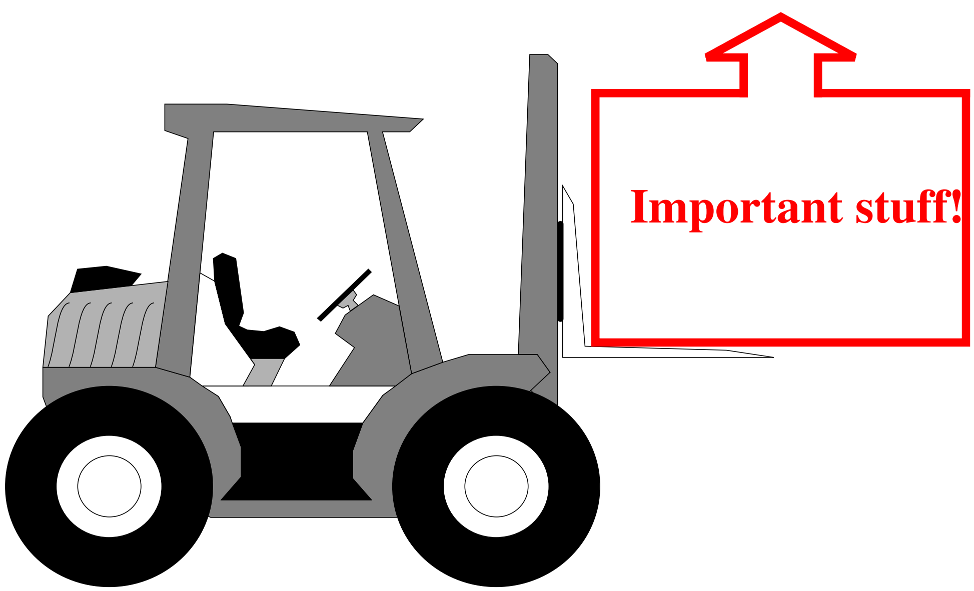


- This is very consistent
  - We only have one copy of the world
  - It controls who can change it and when
  - Synchronization is easier for sockets and such
- This depends entirely on the network
  - Ties user input to network

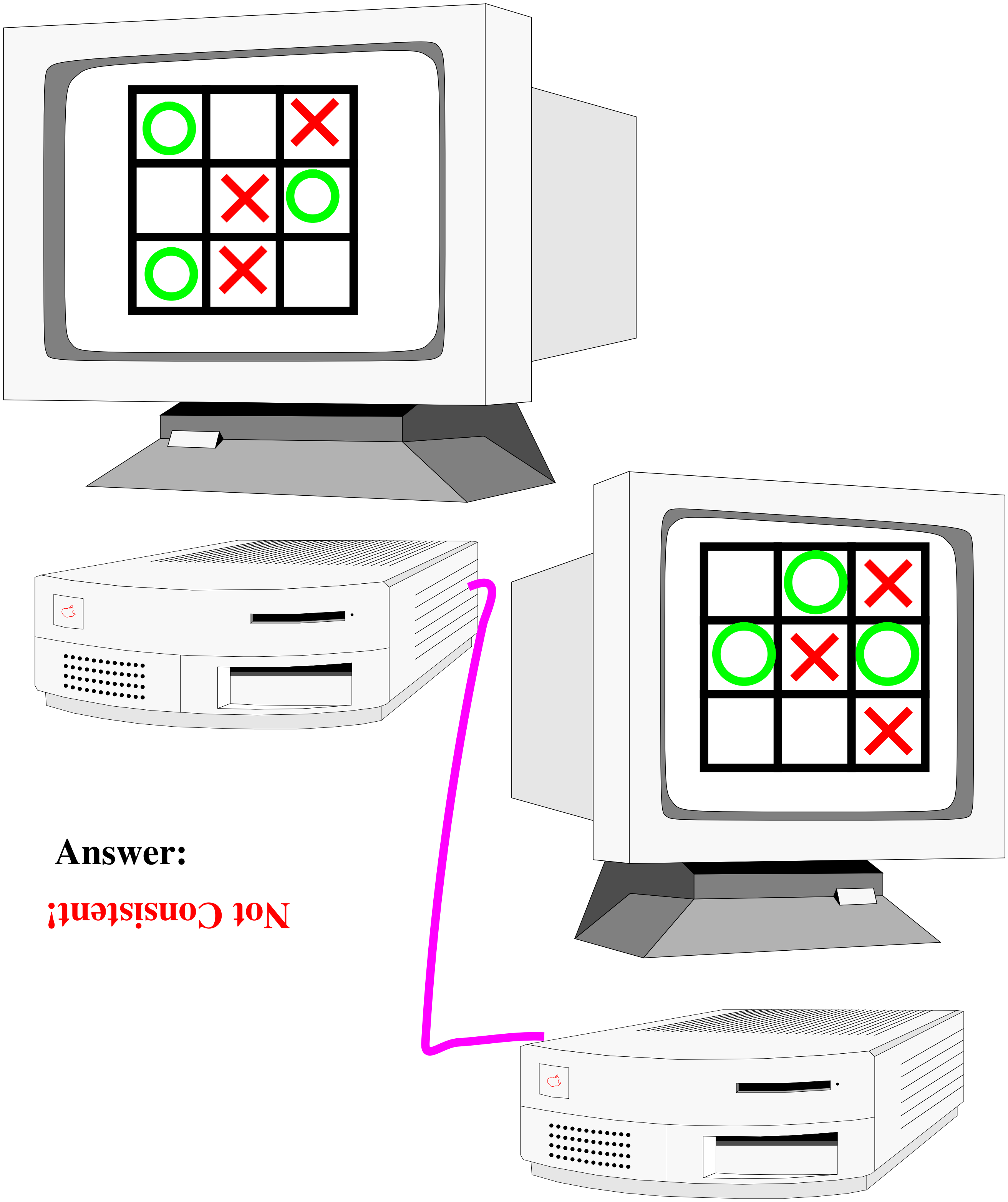
So that is the problem.
How can we recognize it
and how can we avoid it?

- Hard to recognize until code is written
  - So a good design is imperative.
  - An open thinking mind is even better.
  - Learn from your mistakes and others esp. mine

- Common Terms:
  - Server - stores world state
  - Client - interface between user and client
  - Network Delay - Slow down of messages



What is wrong with this picture?



Answer:

Not Consistent!

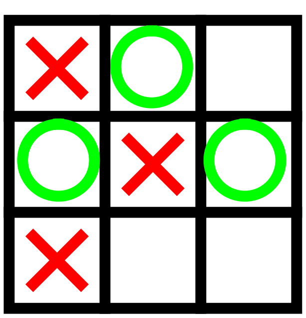
What is a world?

- High level idea
  - A world is the environment the network players (and possibly artificial players) inhabit and interact in and/or with.

- Systems definition:
  - Data structure
  - Some way to represent all of the objects and characters inside.

Example Tic Tac Toe:

3 x 3 array of chars.
‘X’ = x
‘O’ = o



What quality must these world's have to work?

Interesting - No
Large - No

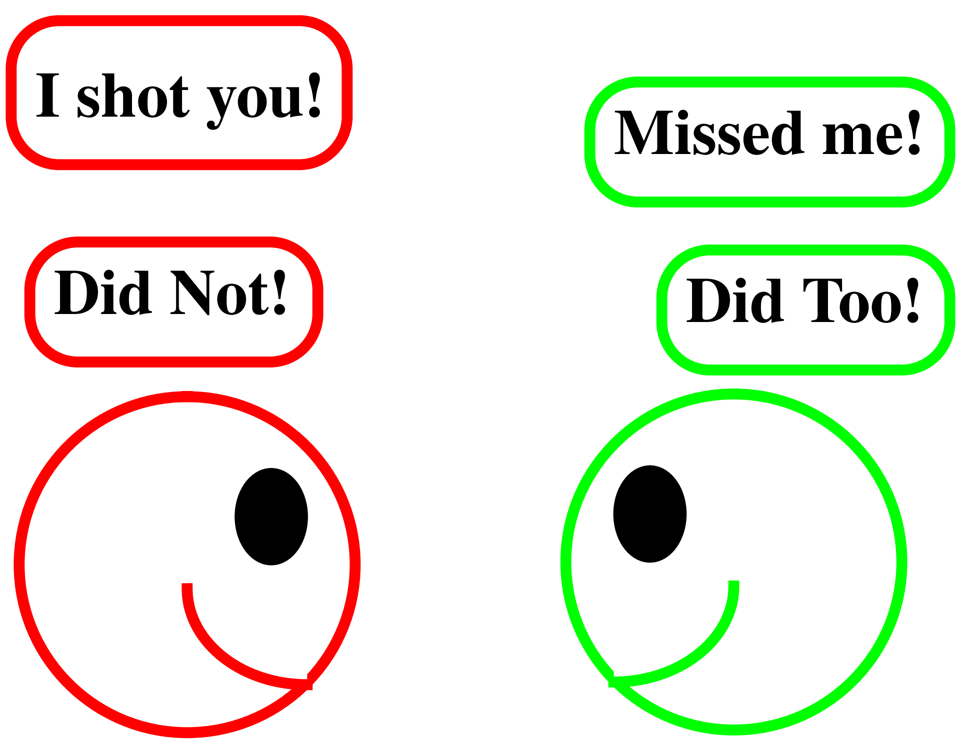
They must be:

consistent!

(usually)

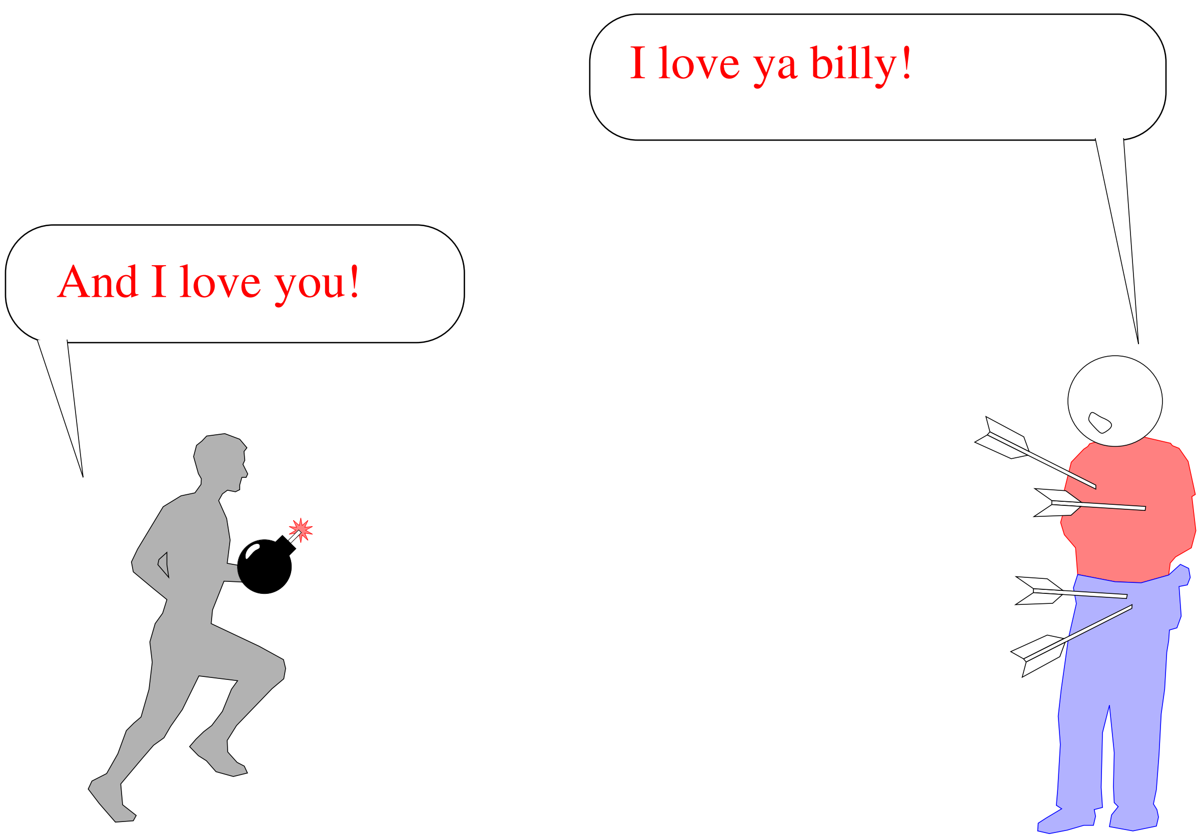
When the two network players see each other they must be interacting in the same world.

Otherwise we have the famous
Cowboys & Indians problem:



Why are network games so much fun?

- You get to interact with your friends (or strangers)
- You both interact in one world together sharing your hopes and dreams
- You can kill your friend in this separate world
- You can communicate with the other player in this world



Consistency