Properties of Multiplayer Games

number of players

- 2–8
- < 100
- > 100

number of nodes

communication requirements

- turn–based
- twitch
- asynchronous but not fast action

amount of data

tolerable delay

persistent world (or not)

exiting/joining players

Multiplayer Technology

- network type
  - wire
  - lan
  - internet via modem
  - cable modems...
- communication configuration
  - client/server
  - peer to peer
- communication protocol (Ammar)
- equalizing the playing field
  - lowest common denominator
- modeling the world

Where should the information reside/be computed?

action of other players ⟷ bits for graphics

downstream traffic: server–> player

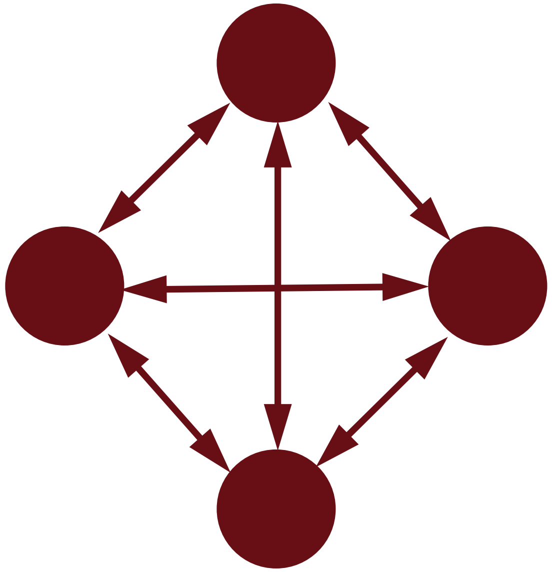
upstream traffic: player –>server

interplayer traffic

Communication Configurations

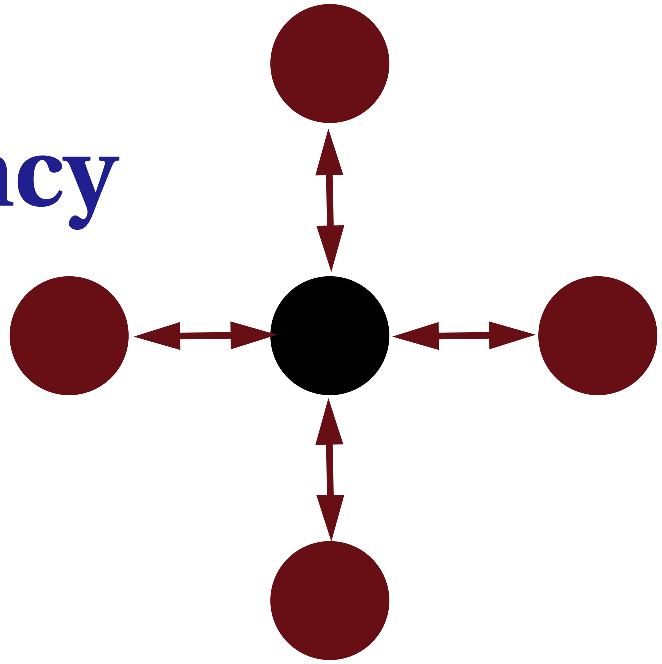
peer to peer

- $c = n(n-1)/2$
- restricted to speed of slowest computer
- easy to cheat



client/server

- $c = n$
- restricted by highest latency
- harder to cheat



combined model

Bandwidth

amount of data the network can transfer in a given time

scalability

Latency

time it takes for a packet to get from sender–>receiver

modem
routers buffer
cache misses on routing table
dropped packets

responsiveness

Bandwidth over a modem?

10 (frame rate) x 140 bytes/sec
~ 14.4K modem

Latency with a modem?

20–30ms latency (times 2–4 for roundtrip)

Consistent World View

perfect consistency may not matter

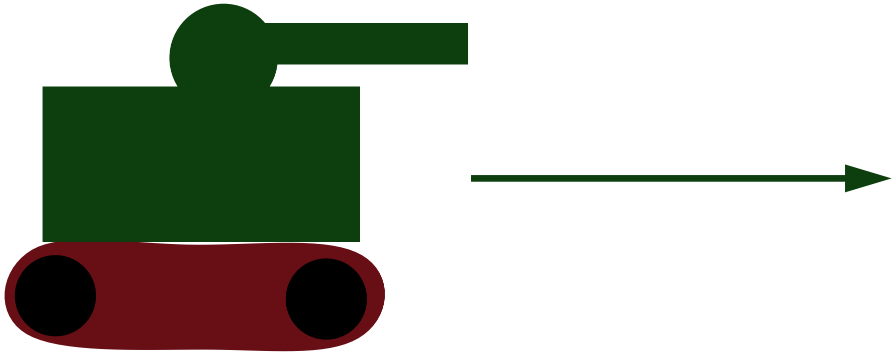
order of events may matter

decouple communication from game
    to the extent possible

hide latencies via tricks in game design
    multiple shots to kill
    make rockets travel for a long time
    characters move slowly with inertia

Dead Reckoning

- designed for Distributed Interactive Simulation Protocol (DARPA SIMNET)
- Basic idea: each processor simulates all entities at a coarse level



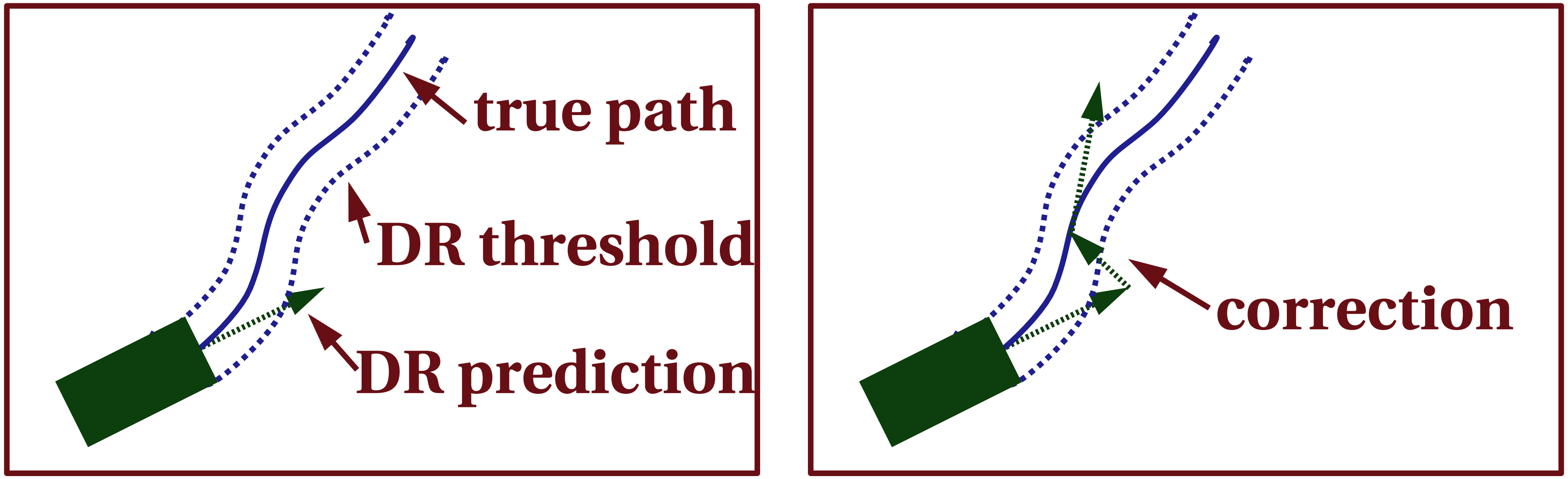
Dead Reckoning

Entity state protocol data unit (PDU)

- current kinematic state of entity
  - (position, velocity, acceleration, orientation)
- damage level
- dead reckoning algorithm

Receiving progam establishes entity
Runs dead reckoning algorithm

Sending program monitors joystick
Compares progress to predicted path
If error threshold exceeded sends new PDU



$$x' = x + v\Delta t$$

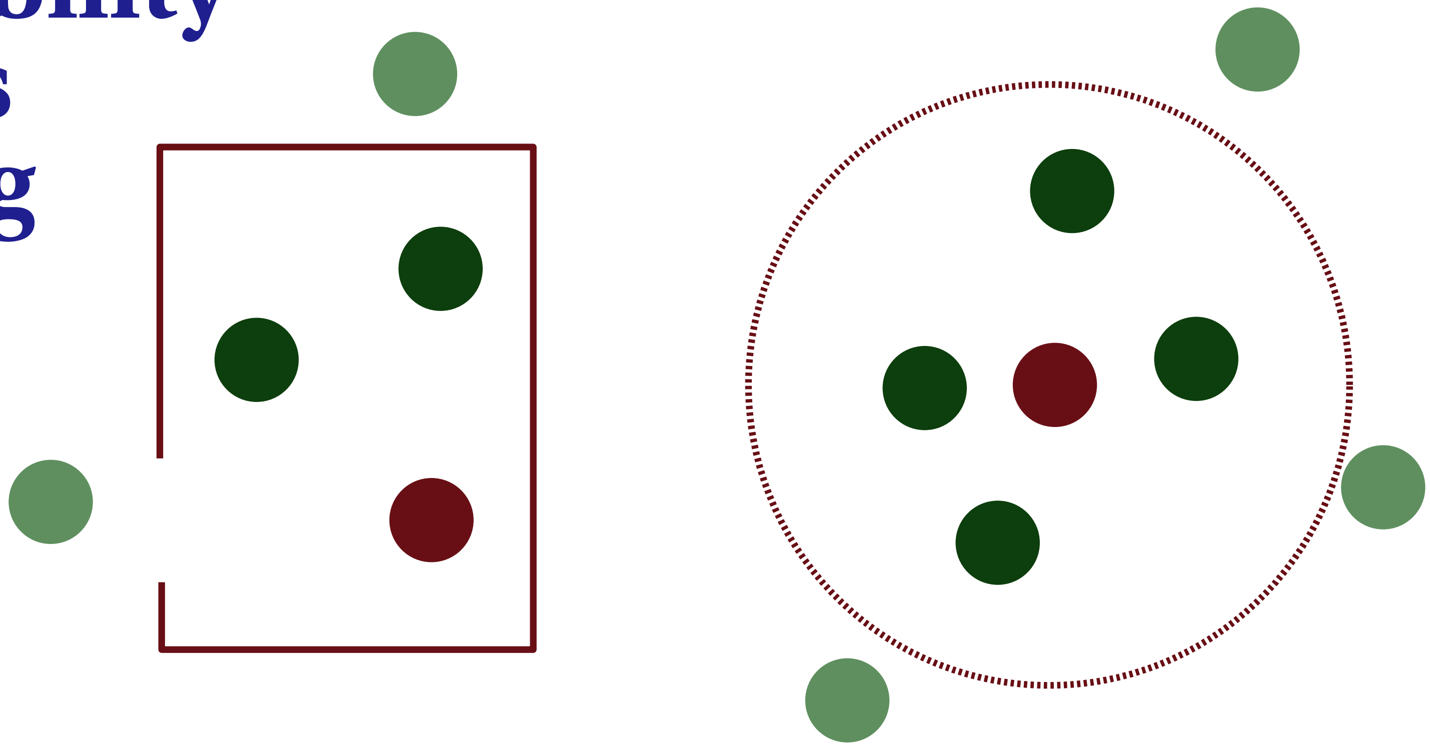
Enhancements

smooth error correction
use hierarchy for turret
better reckoning algorithms
receive updated parameters

Conserving Bandwidth

compress data
prioritze data
- audio

cache more information locally
reduce visibility
- rooms
- culling



Problems unique to Multiplayer Games

- pkill/social issues
- unequal playing field
- cheating with hacked software
- fault tolerance
- how to make money?
    - subscription services
    - advertising

The Future?

audio to taunt/communicate
video images
reduction in latency
increase in bandwidth