Multiplayer Online Games

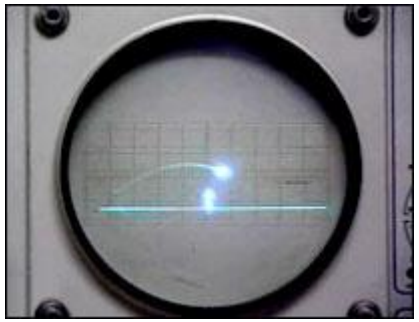
PhD Student Umut Riza ERTURK
Produced for Realtime Software Development Lecture
Lecturer: Kayhan Imre
Hacettepe University
Jan 2011

What is a Multiplayer Game?

- More than one player at a time or NOT playing the same game session
- *Spacewar* is the first multiplayer game and running on a computer (1961) (made more than $100 000)
- *Tennis for Two* is the very first multiplayer game running on an oscilloscope(1958)

Spacewar

Taken from http://en.wikipedia.org/wiki/Spacewar!



Tennis for Two

Taken from http://en.wikipedia.org/wiki/Tennis_for_Two

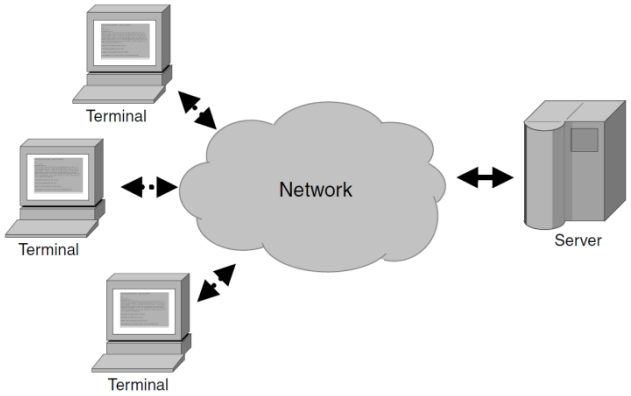
What is an Online Game?

- In which players connect to devices remotely to play
- MUD 1 is one of the first online games, text based
- Basic client-server topology

```
This persona already exists - what's the password?
*
Yes!
Hello, Bunkus!
Elizabethan tearoom.
This cosy, Tudor room is where all British Legends adventures start. Its
exposed oak beams and soft, velvet-covered furnishings provide it with the
ideal atmosphere in which to relax before venturing out into that strange,
timeless realm. A sense of decency and decorum prevails, and a feeling of
kinship with those who, like you, seek their destiny in The Land. There are
exits in all directions, each of which leads into a wisping, magical mist of
obvious teleportative properties...
*
Iceberg the necromancer has just arrived.
*
Iceberg the necromancer has just left.
*
Balthazar the mortal wizard has just arrived.
*
From somewhere in the distance comes a low reverberating sound.
*
```

MUD-1

Armitage p.9



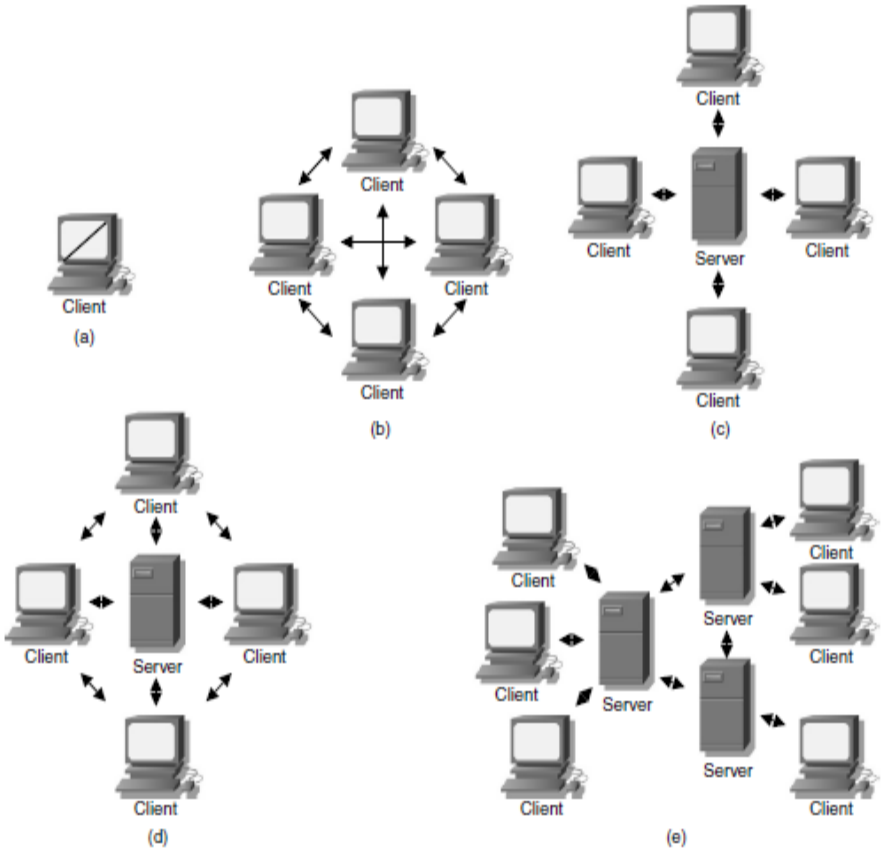
Early Multiplayer Online Games (Milestones)

- More than one player playing on the same game session
- First example: *Doom* by *ID software* (1993)
  - Using IPX for communication
  - P2P (peer to peer topology)
  - Every 1/35th of a second game collects user input and broadcasts the network packets (sends it to other players)
  - No sever, no client (or vice versa)
  - Pros and cons? (discussion topic)
- Doom 2
  - Same topology and method except for broadcasting packets

Early Multiplayer Online Games (Milestones)

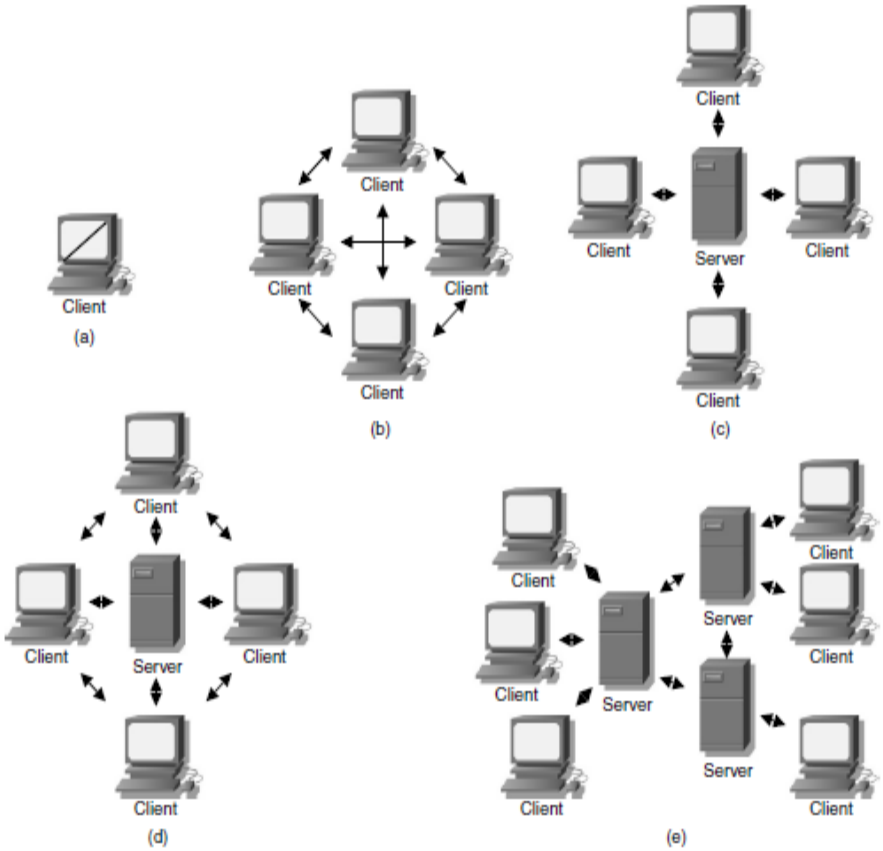
- Quake (1996)
  - Big leap in online gaming
  - Server-client topology
  - No Tunnelling (quake was using directly internet protocols thus no need to convert wan packets into different packets) which enabled players to play via internet
  - No need to meet in the same time and game room, Quake had its own game rooms on internet
  - Clients send only their own inputs to the server and get back the new game state from the server

Communication Architectures for Multiplayer Network Games



- (a)Pure client: nothing but a client
- (b)P2P: No servers, clients should know each other
  - n*n connections might be needed
  - messaging traffic increases exponential as new players join the game
  - Vulnerable to cheating
- (c)Client-Server: One server which knows clients but clients are not aware of each other hence all the messaging is passing through the server
  - Needs a ‘strong’ server responding consistently and fair to all clients
  - Makes voice chatting or data exchange btw clients hard

Communication Architectures for Multiplayer Network Games



- (d)Hybrid Sever: Allows easy data exchange between clients
  - Security problems may occur as clients might be vulnerable to attacks
  - clients network might be kept busy by other clients in purpose
  - Still needs a strong and consistent server
- (d)Network of Servers : Allows hundred or even thousands of people to be online like in MMORPG games
  - Costly
  - Difficult to manage and maintain

Network Problems

- Latency, Jitter, Loss
  - Latency
    - Latency is the time spent for a packet to reach its destination.
    - Considered as half of Round Trip Time (send a data and get it back) for symmetric networks
  - Jitter
    - Variations in latency
    - Means inconsistency
  - Loss
    - Some packets gets lost (corrupted)
    - Increases time spent for data exchange

Network Problems

- Causes of
  - Latency
    - Propagation delay which is a result of;
      - Physical Limitations (speed of light is the upper limit for communication)
      - latency (ms) = (distance of link in kilometres)/300
    - Serialisation; data is transferred in series of bits, all the bits should arrive to the destination in a correct order to be serialised which takes time
      - latency (ms) = 8*(link layer frame length in bytes)/(link speed in Kbps)
    - Queuing delays; Switches doesn't provide equally timeslots for internet communication which means if the internet traffic is heavy on a switch internet gets slow.

Network Problems

- Causes of
  - Jitter
    - Path length changes
      - Data being sent doesn't always use the same path on reaching its destination
    - Packet size changes
      - Hence because of serialisation, sending time varies

Network Problems

- Causes of
  - Loss
    - Physical unrecoverable bit errors
    - Dropping packets because of heavy communication traffic

How Network Games Tolerate Network Problems

- Latency Compensation
  - Why
    - There will always be delays because of physical limitations
    - Bandwidth is increasing but there are loads of packet loss
    - Broadband communication speed varies from country to country
    - A Game should be
      - Consistent
      - Responsive
      - Fair
  - So we always need methods to compensate the latency caused by network infrastructure

Latency Compensation

- The Simplest Client Side Algorithm of Network Games
  - Collect user input
  - Pack up data and send it to server
  - Receive updates from server and unpack
  - Determine game state
  - Render scene
  - Repeat

Latency Compensation

- Outcome of the basic algorithm
  - Client waits until server responses which decreases the responsiveness of the game but also makes the game consistent

Latency Compensation

- Prediction
  - Player Prediction
    - Client predicts player's own movement and continues movement
    - Algorithm
      - Sample user input
      - Pack up data and send to server
      - **Determine visible objects and game state**
      - Render scene
      - Receive updates from server and upack
      - **Fix up any discrepancies**
      - Repeat.

Latency Compensation

- Prediction
  - Opponent Prediction
    - Client predicts opponents' movements
    - Requires more data to be transferred from other clients in order to make a reasonable prediction
    - However, by not sending the data if the difference is under and threshold, data transfer rate could be reduced

- **Algorithm on the player to be predicted**
  - **Sample user input**
  - **Update {location | velocity | acceleration} on the basis of new input**
  - **Compute predicted location on the basis of previous {location | velocity | acceleration}**
  - **If (current location – predicted location) < threshold then**
    - **Pack up {location | velocity | acceleration} data**
    - **Send to each other opponent**
  - **Repeat.**

- **Algorithm on the opponent**
  - **Receive new packet**
  - **Extract state update information {location | velocity | acceleration}**
  - **If seen unit before then**
    - **Update unit information**
  - **Else**
    - **Add unit information to list**
  - **For each unit in list**
    - **Update predicted location**
  - **Render frame**
  - **Repeat**

Latency Compensation

- Prediction Summary
  - Might be useful if a responsive but not that consistent game
  - What if a packet gets lost? And arrives afterwards (in this case using UDP is not the best idea)
  - Some situations are impossible to predict, i.e. 3 players running towards each other and tosses, who tossed to whom first?

Latency Compensation

- Time Manipulation
  - Some clients might have high latencies
  - Game should be fair, consistent and responsive
  - (example of collecting rewards)

Latency Compensation

- Time Manipulation
  - Time Delay
    - Server looks at the latencies and in order to maintain the fairness and consistency adds some extra delay for the ones has less latency
      - Jitter is a problem here
      - What if the client is showing its latency higher than it actually is

Latency Compensation

- Time Manipulation
  - Time Wrap
    - Server looks at latency to the client and rolls back the time as necessary
    - Also protects from inconsistencies caused by loss packets (as they might not arrive in time order)
    - May still cause inconsistencies i.e. the player-1 is just moved to behind of a wall and at that moment player-2 shot, for player-1 bullet bend and moved to the behind of the wall (this may cause regularly if jitter rate is high)

- Algorithm on server side
  - Receive packet from client
  - Extract information (user input)
  - **Elapsed time = current time – latency to client *(rollback the time to the clients time)***
  - **Rollback all events in reverse order to current time – elapsed time**
  - Execute user command
  - **Repeat all events in order, updating any clients affected**
  - Repeat.

Latency Compensation

- Sending less or compressed data
  - Lossless compression: LZW algorithms for compression of stream data
  - Opponent prediction: Reduces the data sent
  - Delta compression: Send the differences not the whole data
  - Interest Management: Send information of close objects (don’t send distant objects’ information)
  - P2P: Avoids sending data irrelevant data to the server
  - Update Aggregation: Collect data for a client and sent all the collected data at a time instead of sending instant update data

Further reading topics

- Cheating Prevention
- Fair Randomness

Questions

Bibliography

- Armitage, G. , Claypool, M. , Branch, P. (2006). *Networking and Online Games*. West Sussex: John Wiley & Sons Ltd.
- Smed, J. , Hakonen, H. (2006). *Algorithms and Networking for Computer Games* . West Sussex: John Wiley & Sons Ltd.
- Barron, T. (2001). *Multiplayer Game Programming.* California: Stacy L. Hiquet