Multiplayer Game Programming

Lecture 5: Tribes Networking Model

ITP 484

Review time!

- How do you calculate the optimal bandwidth MSS for TCP?
- From a game dev perspective, what is one advantage a Client/Server architecture has over a Peer-to-Peer one?
- From a game dev perspective, what is one advantage a Peer-to-Peer architecture has over a Client/Server one
- Describe two ways you can host a server for a game on a computer whose only IP address is in a private IP block.

WireShark Demo!

Question

- TCP? UDP?
- Client/Server, Peer To Peer?
- What do we share, what do we simulate where?

There is no answer correct in all cases

Because our real question is:

While implementing our game as designed:

How do we optimize the use of our network connection?

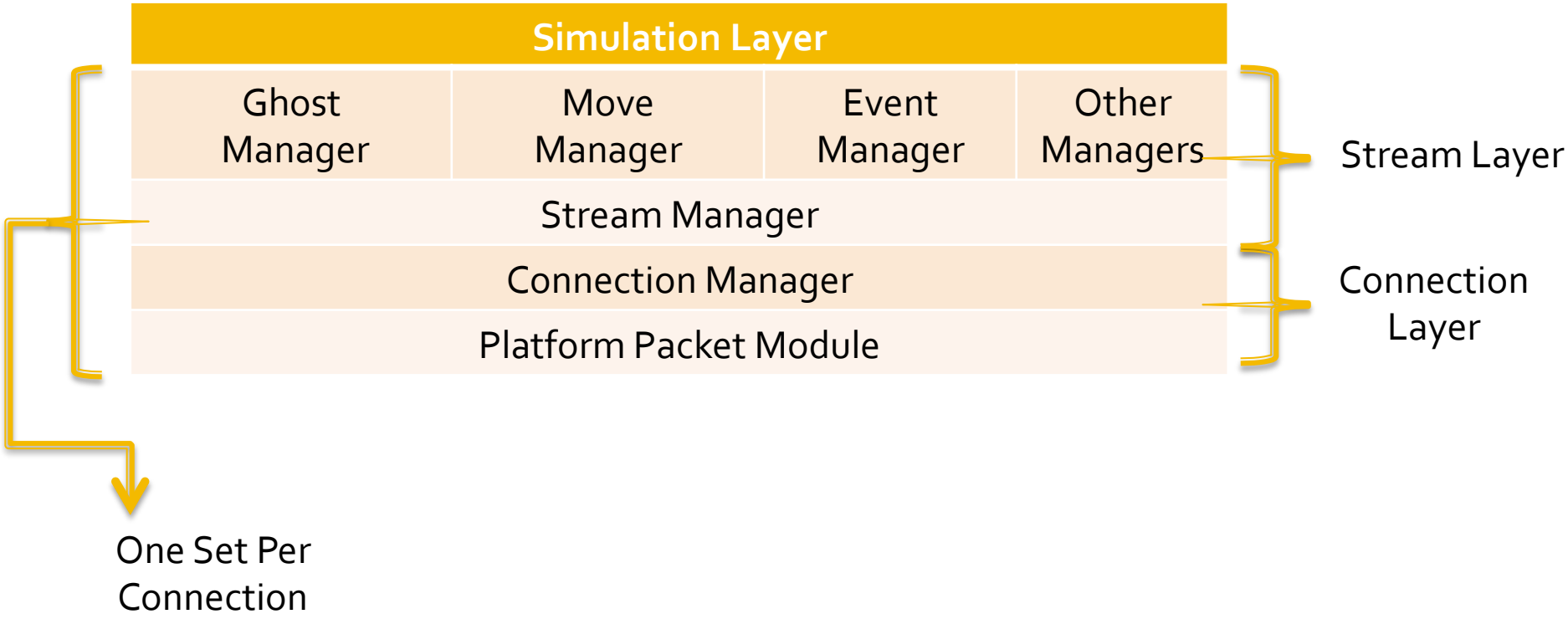
Starsiege: TRIBES

- Released in 1998
- Supports up to 32 players over LAN or Internet
- Designed to support low end modem connections
- Client / Server model

Tribes: Network Requirements

- Non guaranteed data
  - Not retransmitted if lost
- Guaranteed data
  - Retransmitted if lost, delivered in order
- Most recent state data
  - Only the latest version is of interest
- Guaranteed quickest data
  - Has immediate delivery priority

Tribes: Application Architecture



Platform Packet Module

- Wraps platform level socket interface
- UDP

Connection Manager

- Guarantees Delivery Status Notification, not delivery (not reliable!)
- Accepts packets from the Stream Manager
- Sends them to the Platform Packet Module
- Sends ordered delivery status notifications back to Stream Manager

Connection Manager

- Basic Implementation:
  - Give each packet a sequence number
  - If a side receives a packet with a consecutive sequence number, process the packet and ack it
  - If a side receives a packet with a > nonconsecutive sequence number, process the packet and ack it, but nack the skipped packets
  - If a side receives a packet with a < nonconsecutive sequence number, drop the packet silently
  - If you receive an ack, report all packets of that sequence number or less as received
  - If you receive a nack, report that packet as dropped
  - If a side doesn't receive an ack for a packet after a certain timeout, report the packet as dropped

Stream Manager

- Sends and Receives max packet rate and size that can be received
- Creates packets for higher level managers
  - Prioritized: Move, Event, Ghost
  - Dispatches them to Connection Manager
- Receives delivery notification status from Connection manager
  - dispatches to higher level managers.

Stream Manager: Transmission Records

- One per packet
- Sliding window of records
  - ring buffer or similar data structure
- When delivery status received for packet
  - Matched with corresponding Transmission Record
  - Status and record sent to higher level managers
  - Removed from sliding window

Event Manager

- Queue of Events to share
  - Generated by simulation layer
  - Sorted by priority
- Writes events into a packet until
  - Packet is full
  - Queue is empty
  - Event Window exceeded (limit on events that can be in flight)
- Copy each event into packet's transmission record

Event Manager: Transmission Records

- If packet was not delivered
  - For each event
    - If event delivery guaranteed, prepend to event queue
    - Reliability!

Ghost Manager

- Scoping / Relevance
  - What should a client know?
  - What does a client _need_ to know?
- Responsible for "ghosting" in-scope objects from server to client
- Guaranteed delivery of ghost
- "Most recent state" data for ghost's data

Ghost Manager

- If sim decides object becomes "in scope"
  - Assigned Ghost Record
    - Ghost ID: id used to refer to ghost in remote hash map
    - State Mask: data to ghost
      - Each bit represents a different group of attributes
    - Priority: adjusted by simulation
    - Marked as having a status change (" in scope")
- If sim changes "in scope" object's data
  - Set corresponding bit in state mask
- If sim decided object no longer "in scope"
  - Mark ghost record with status change "not in scope"

Ghost Manager: Filling Packet

- For each ghost record with ≠ 0 state mask or status change
  - Sorted by status change, then by priority
  - Write Ghost ID
  - Write status and data referred to by state mask into packet
  - Write state mask and status change into transmission record
  - Zero state mask!
  - Stop early if packet full

Ghost Manager: Receive Packet

- For each chunk of ghost data
  - Read Ghost ID
    - Is status change?
      - Yes: Create or Destroy as appropriate- store in map
      - No: Lookup Ghost by ID
    - Read state data into ghost

Ghost Manager: The Magic

- If packet reported as dropped
  - For each ghost record's data in transmission record
    - For each bit in state mask
      - Does a later transmission record have that bit set?
        - Yes: great, you already sent an update. No Worries!
        - No: Set that bit in the ghost record's state mask
    - Handle status changes the same way
- "Most Recent State" Eventually Sent!!
- Example

Move Manager

- Sends client Moves to server and server Player Controlled Object data to client
- Input Manager gathers client moves at 30 fps
- Moves are important!
  - "Guaranteed soonest possible data"
  - Packets go to Move Manager for filling first!

Move Manager

- Gathered moves are sent to server
- Server applies those moves and acks them
- Sliding window on client holds unackd moves
- Original Tribes: every packet gets every unack'd move
- Problem
  - If you drop a packet, the last thing you want to do is start making bigger packets!
- Tribes II: Moves sent in 3 consecutive packets

Move Manager

- Server applies move
  - Simulates Controlled Object in response to Moves
- Client also applies move
  - Helps compensate latency
- Server sends Controlled Object back to client- Why?
  - Correction! (For another lecture)
- Controlled Object ghosted to other clients

Data Block Manager

- Sends rarely changed blocks of data to client
  - i.e. Static Vehicle description with an id
    - ghost can just reference the id
- Data block tagged with "last modified key"
  - Whenever modified, key incremented
- Manager tracks latest key sent to client
- Sends any datablocks with incremented keys
- Uses event manager to send as reliable event