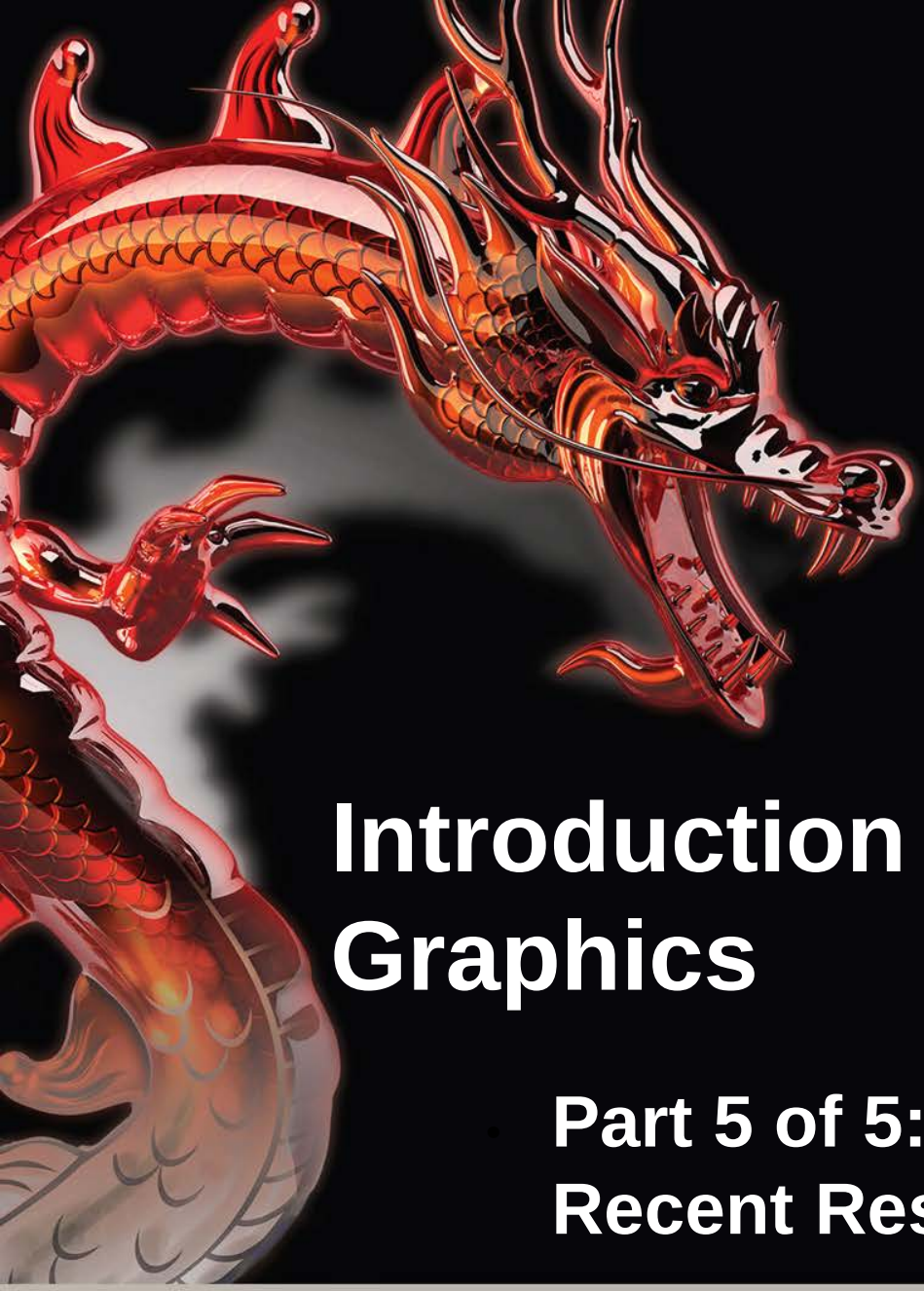


Introduction to Networked Graphics

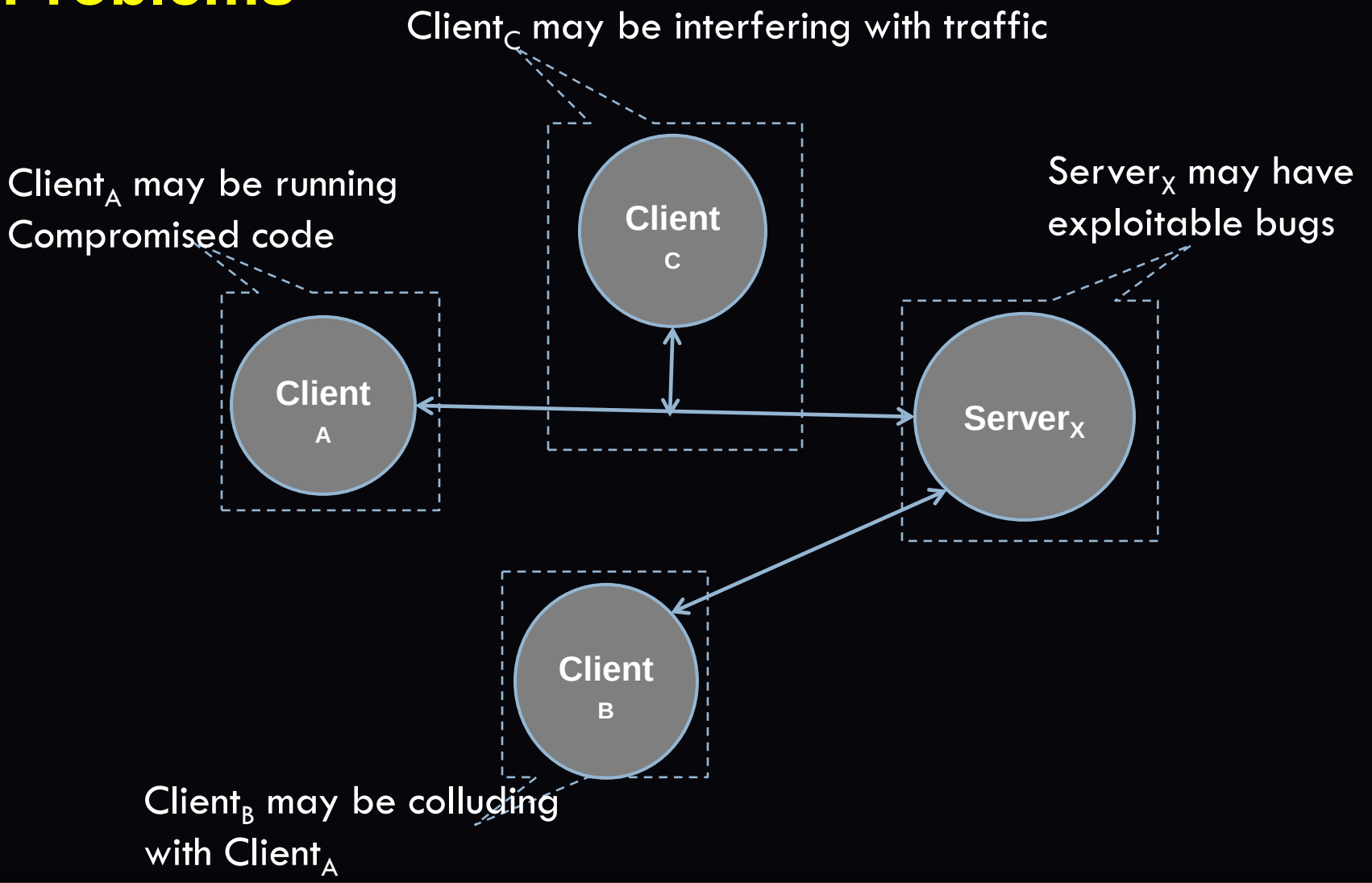
Part 5 of 5: Application Support & Recent Research

Overview

- **Goal:**
  - **To explain some other application issues and areas of recent research.**

- **Topics:**
  - **Security and secure networks**
  - **Streaming**
  - **Cluster graphics**
  - **Thin clients**
  - **Peer to peer**

Overview of Security Problems



Compromised Clients

- A pervasive problem in gaming
  - E.G. notable problems with PSNet games after the PS3 master key was found allowing modified code on the PS3
- For console gaming, hardware vendors try to lock down the hardware so only verified programs can run
- For PC gaming, various detection techniques such as PunkBuster that detect malicious software
  - Countermeasure are typically ahead of amateur cheats but not professional cheats

Traffic Interference

- Once data is on the network it is public
- Various attacks
  - Packet injection
  - Packet hiding
  - Latency asymmetry
- Some are mitigated by secure networks
  - Some servers specifically support secur

Exploitable Server

- Users need to trust server, user hosted games are not accepted for ranking tournaments or cash games
- Server might be have a loophole
  - E.G. Dupe bugs
- Denial of service attack

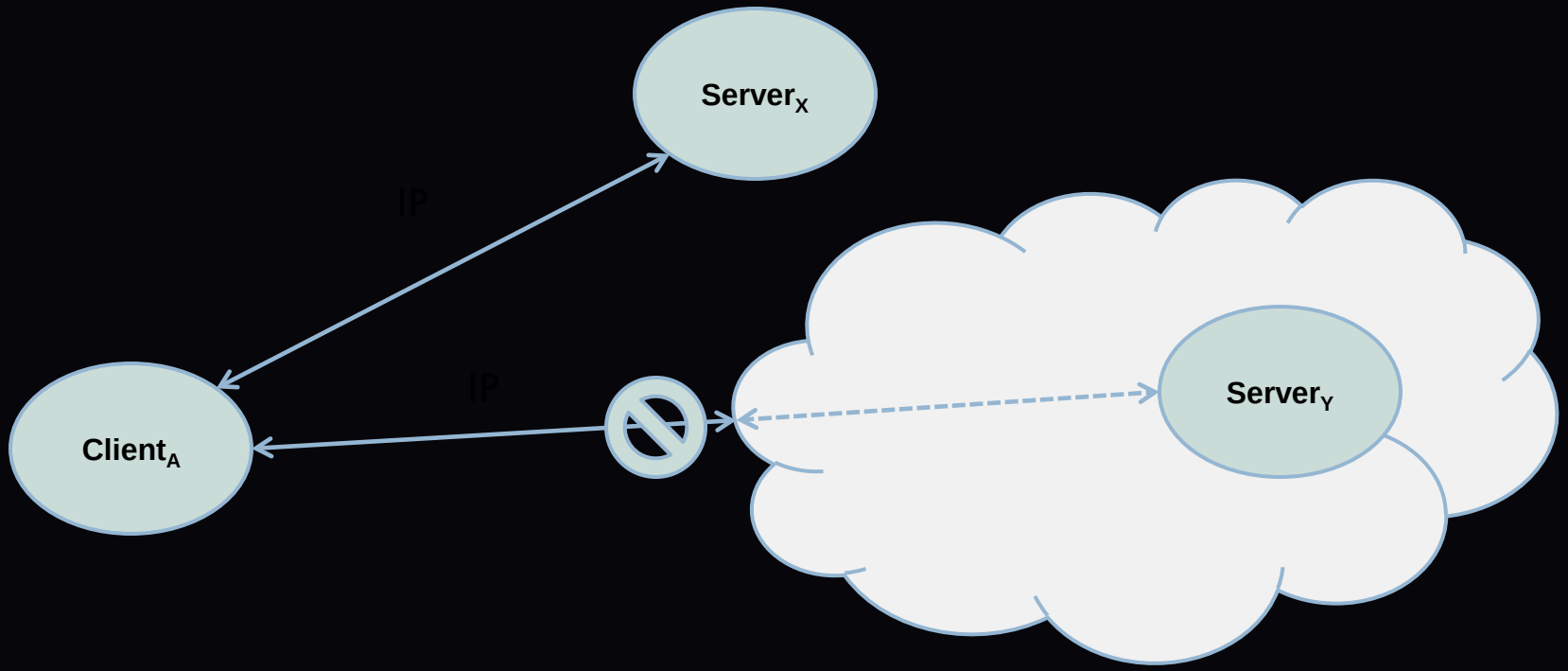
User Collusion

- A very difficult social situation to counter
  - E.G. Chip dumping
- With this and all other security problems *monitoring* of exceptions is important
  - Players being too skillful
  - Unlikely plays
  - Game inventory inflation

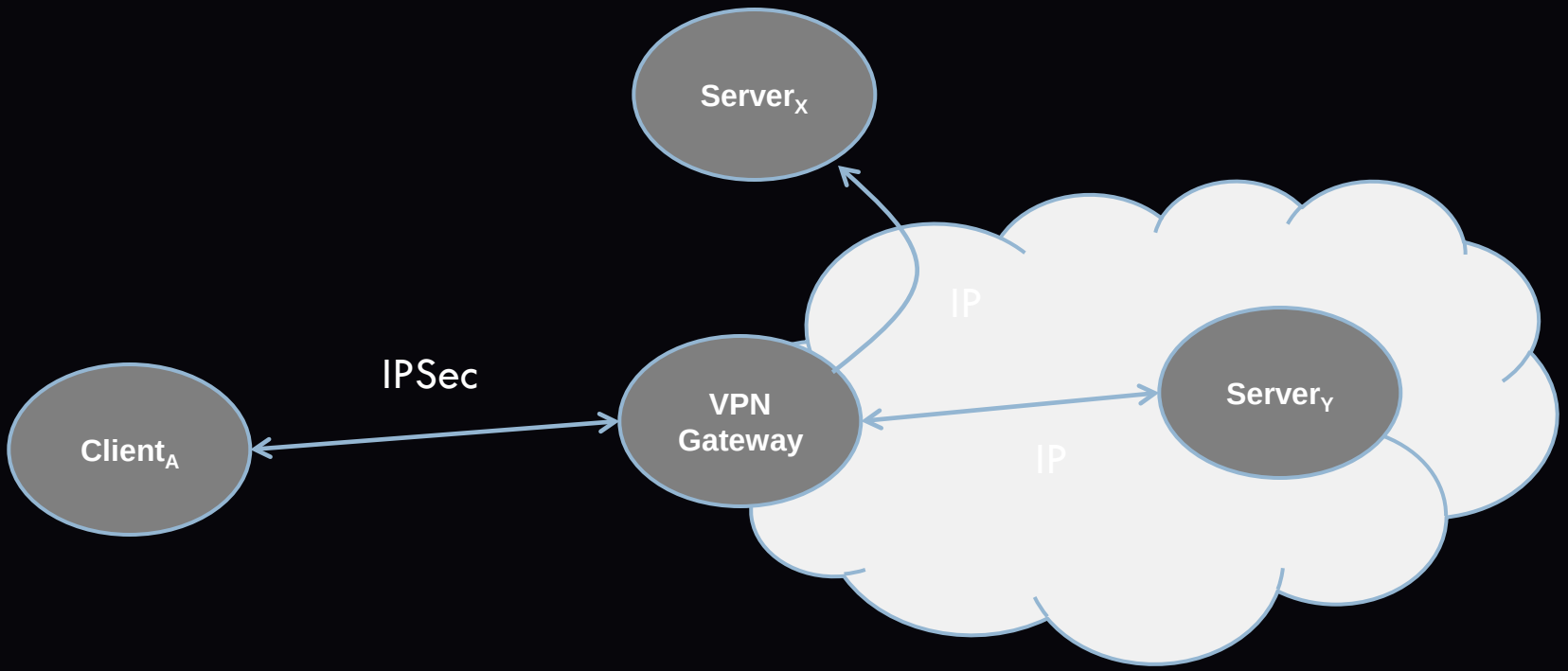
Virtual Private Networks

- Now very common for corporations and universities
- Three reasons
  - Protection of internal services
  - Giving a different “appearance” to the outside world (e.g. ACM Digital Library)
  - Security of access from anywhere (no need to trust local network)
- The very easiest way to protect a NVE or NG is to require someone go on a trusted VPN first
  - Incurs latency/bandwidth overhead of routing all information to the VPN access point first

Virtual Private Networks (VPNs)



VPNs and IPSec



Different Uses of Streaming

- Streaming Protocols
- Streaming Animations
- Streaming Geometry (i.e. incremental download)

Streaming Protocols

- Audio/video transport is well developed on the Internet
- However “well developed” means lots of competing solutions
- Several plug and play libraries
- Real-Time Protocol an extension of UDP to support streaming (though not all streaming protocols use it)
- Can get RTP compliant libraries which enables streaming and receiving
  - E.G. AccessGrid, some VoIP solutions

Real-Time Protocol

Bits	0 15		16 31
0-31	Version, config, flags	Payload Type	Sequence Number
32-63	Timestamp		
64-95	Synchronisation Source (SSRC) Identifier		
96+	Contributing Source (CSRC) Identifiers (Optional)		
96+	Header Extensions (Optional)		
96+	Payload Header		
128+	Payload Data		

RTP Payloads

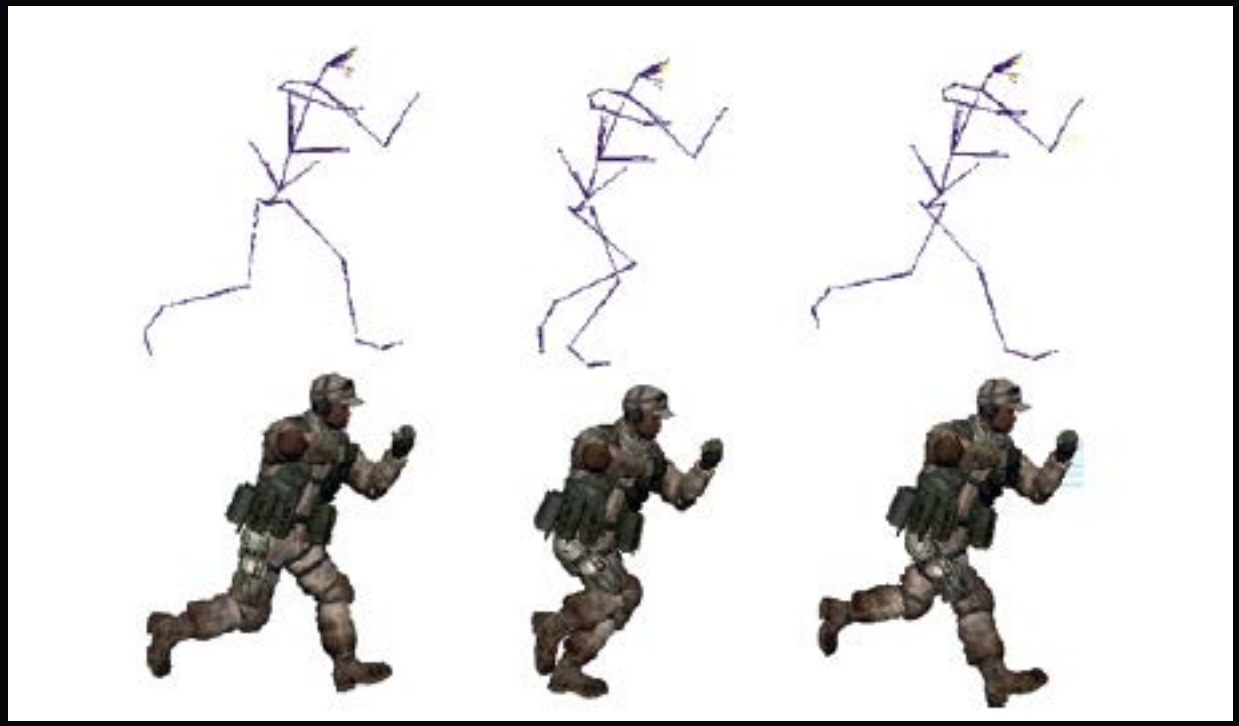
Table 13.1 Some of the Potential RTP Payloads

Description	Specification (RFC)	Type Num	Format
ITU G.711 μ-law audio	1890	0	AUDIO/PCMU
GSM full-rate audio	1890	3	AUDIO/GSM
ITU G.711 A-law audio	1890	8	AUDIO/PCMA
PureVoice QCELP audio	2658	12	AUDIO/QCELP
MPEG Audio (e.g. MP3)	2250	14	AUDIO/MPA
Motion JPEG video	2435	26	VIDEO/JPEG
ITU H.261 video	2032	31	VIDEO/H261
MPEG I/II video	2250	32	VIDEO/MPV

Streaming Animations

- We have already looked at streaming positions and orientations of objects
- However, a large class of objects are humans or animals (or aliens) which deform
- Typically modeled from the graphics side as a skeleton
- Animation is controlled by indicating which *motion* the character is in and the *keyframe* in that motion
- Because motion is continuous (e.g. motion capture) information might only need to be sent > 1s

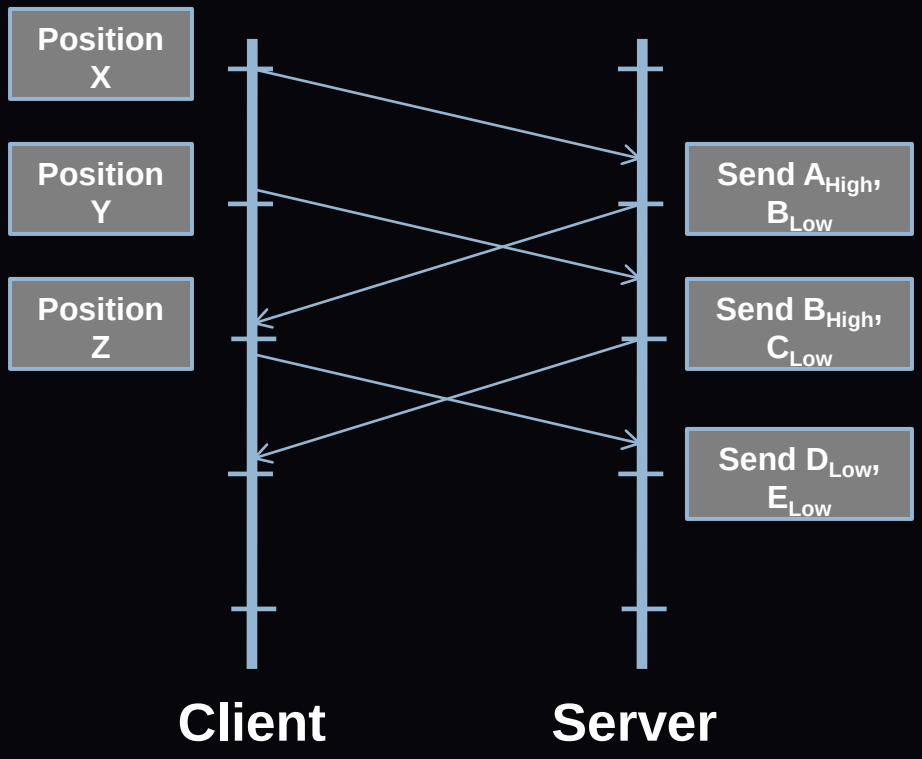
Examples of Keyframe Animation



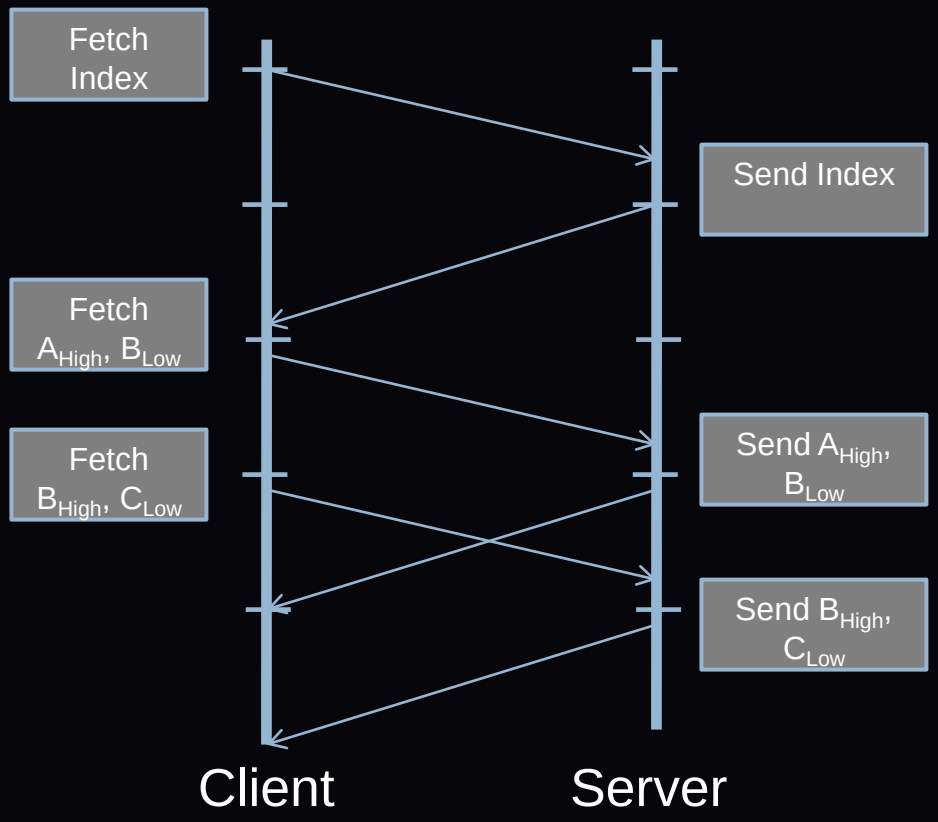
Streaming Geometry

- Many NVEs use very large worlds which need to be downloaded because user modifiable or just vast
- System needs to determine which parts of the models should be transferred
- Typically done in a *priority order* from the viewpoint of the client, e.g. in increasing distance order
- Two ways of doing this
  - Client-pull
  - Server-push

Server Push


Position X
Position Y
Position Z
Send A_{High}, B_{Low}
Send B_{High}, C_{Low}
Send D_{Low}, E_{Low}
Client
Server

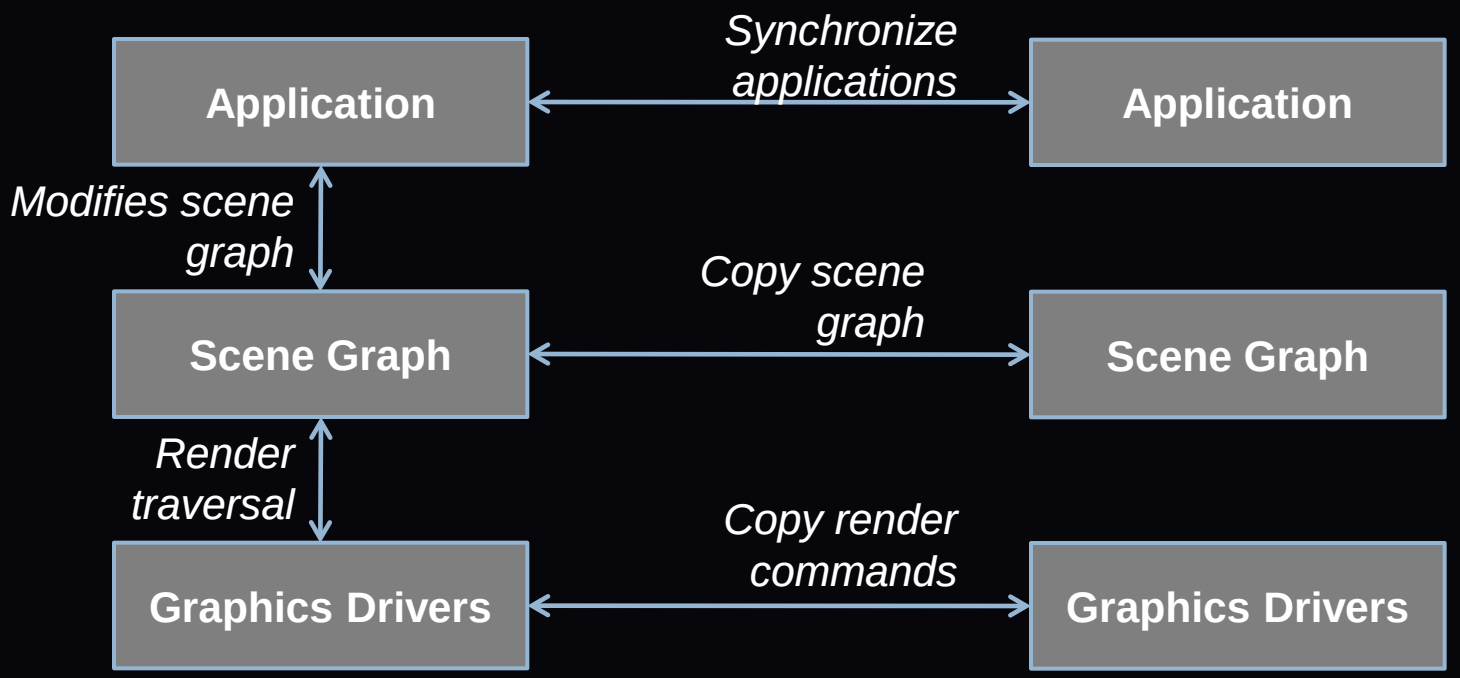
Client Pull


Fetch Index
Send Index
Fetch A_{High}, B_{Low}
Send A_{High}, B_{Low}
Fetch B_{High}, C_{Low}
Send B_{High}, C_{Low}
Client
Server

Clusters

- Cluster graphics is a particular concern of Virtual Reality system designers
- One GPU card generates one or two video to get maximum throughput, but we might need 4+ displays
- Need to synchronize graphics at two levels
  - Synchronize graphics state on input to rendering
  - Need to synchronize video output

Layers of Sharing Graphics



Tools

- Copy render commands
  - E.G. Chromium – stream OpenGL commands over TCP/Ethernet, or other non-IP-based interconnects
- Copy scene graph
  - E.G. OpenSG – stream an edit change list for a scene-graph
- Synchronize applications
  - E.G. VRJuggler – isolate all input in to one (or more) C++ classes that can serialize themselves to the network, stream the resulting serializations.

Thin Clients

- Might be considered “backwards” but graphics architectures go in circles, so why not networked graphics architectures
- Render the graphics on a server, stream the results as video
- Recent consumer examples: OnLive, OToy, GaiKai
- However many OS vendors have such a functionality for supporting thin clients over LANs

Thin Clients

- Very small installable on client, client doesn’t need to be high-powered (hence thin client)
- Stream to server your controller input
- Stream back video (e.g. 720p from OnLive)
- Server runs both game client and game server (actual architectures not revealed)

Thin Client Pros and Cons

- Pros
  - Very small installable (e.g. only Flash for GaiKai)
  - Thin client can be low power (e.g. Netbook)
  - No need to download/install very large game assets
- Cons
  - Latency
  - Constant high bandwidth use compared to normal game network traffic

Peer to Peer

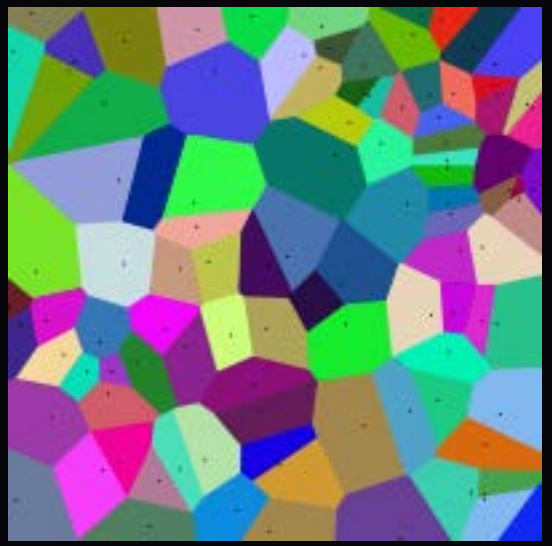
- A live challenge: how can peer to peer networks scale up to very large numbers
- Key to this is how to distribute awareness management
- A secondary issue is how to “bootstrap”: how does a user find their local users?

Larger Peer to Peer Context

- Enormous work in networking community on generic large scale peer to peer databases
- Key technologies
  - *Distributed hash tables*: a way of storing data sets across multiple hosts but ensuring fast (O(logN)) access to any data item
  - *Application-level routing:* a mechanism for supporting group peer to peer communication without any underlying network support

Within a NVE Context

- Very active line of research
- For example, can one maintain a set of closest peers with something similar to a Voronoi Tessellation?
- If peers can identify their Voronoi Cell, they can identify their neighbours.
- New clients can walk the cells to get to find their true neighbours



Summary

- Plenty of tools and options to support your NG or NVE project
- Security is a big challenge if you can’t get your users on to a VPN
- Other facilities require more infrastructure and are very domain specific
- Plenty of research issues: thin clients being a wild card at the moment