


Part 4:
Application Support
Tools
Research Issues

Introduction to Networked Graphics

IEEE Virtual Reality 2011

Anthony Steed

- 
- *Application Support*
 - - Security, Protocol decisions
 - - Persistence
 - *Tools*
 - - Middleware
 - - Networked engines
 - *Research Issues*
 - - Scalable peer-to-peer, thin clients
 - - Standards, etc.

Application Support

Introduction to Networked Graphics

IEEE Virtual Reality 2011





- *Application Support*

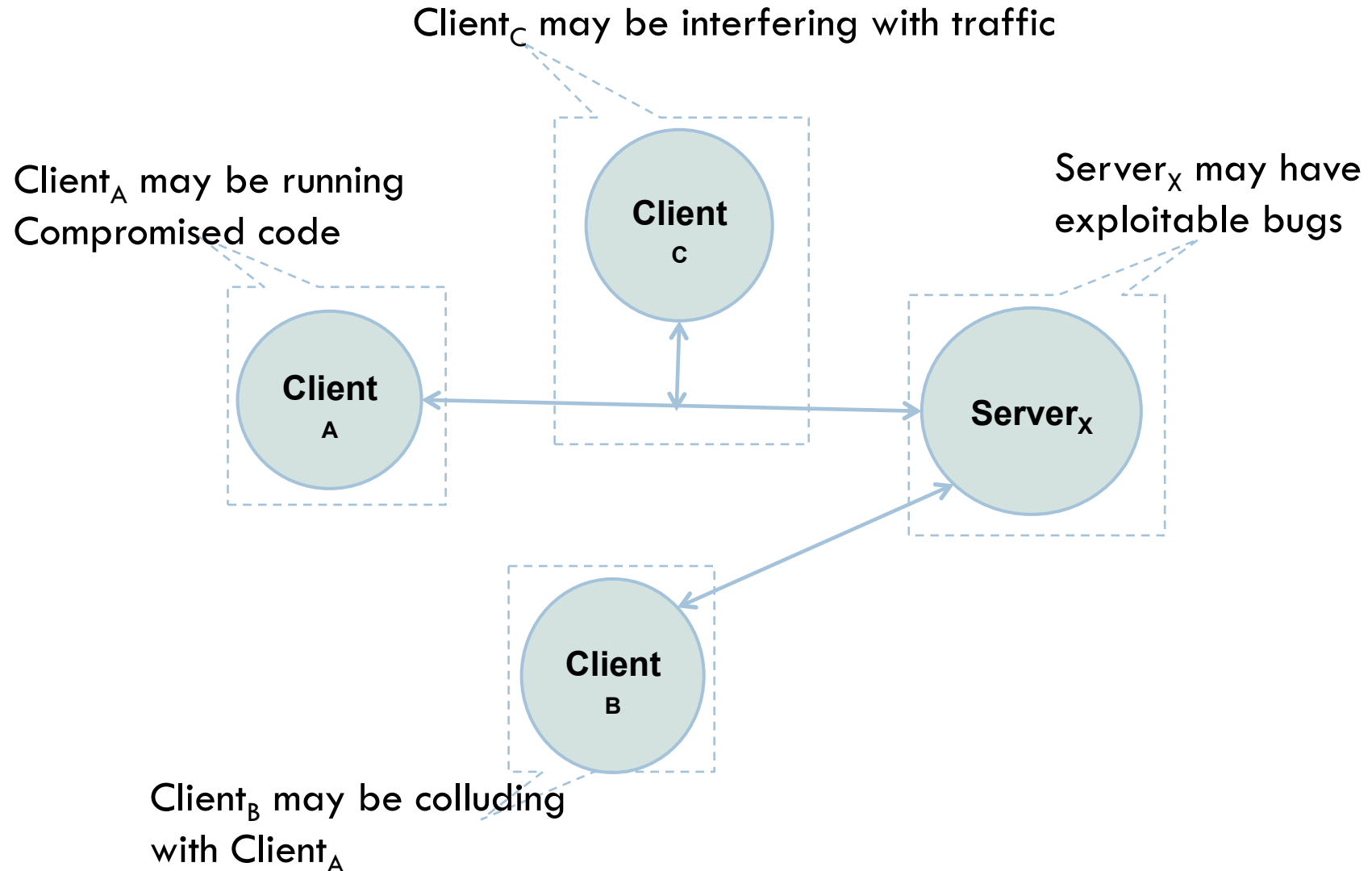
- - Security, Protocol decisions

- - Persistence



SECURITY AND CHEATING

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 on the PC
 - ▣ Countermeasures 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



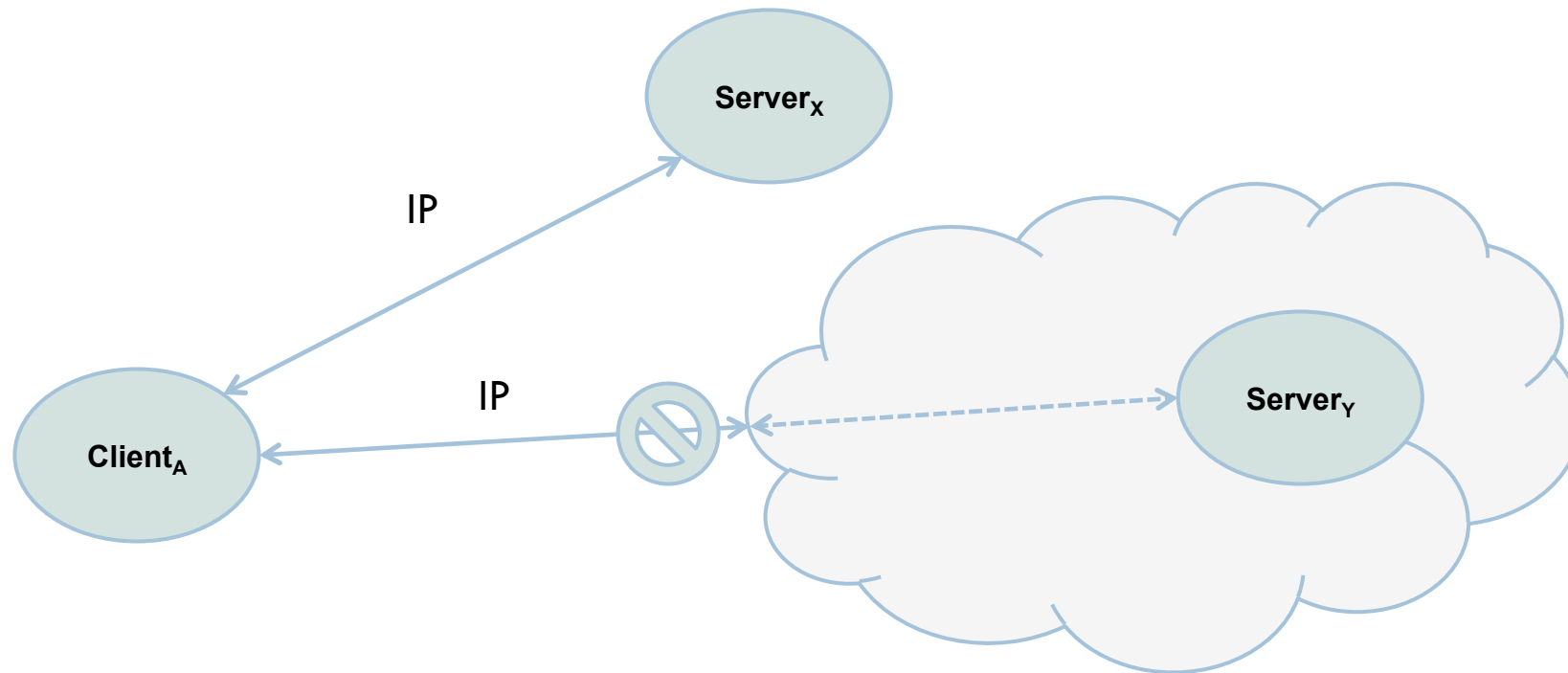
SECURE NETWORKS

Virtual Private Networks

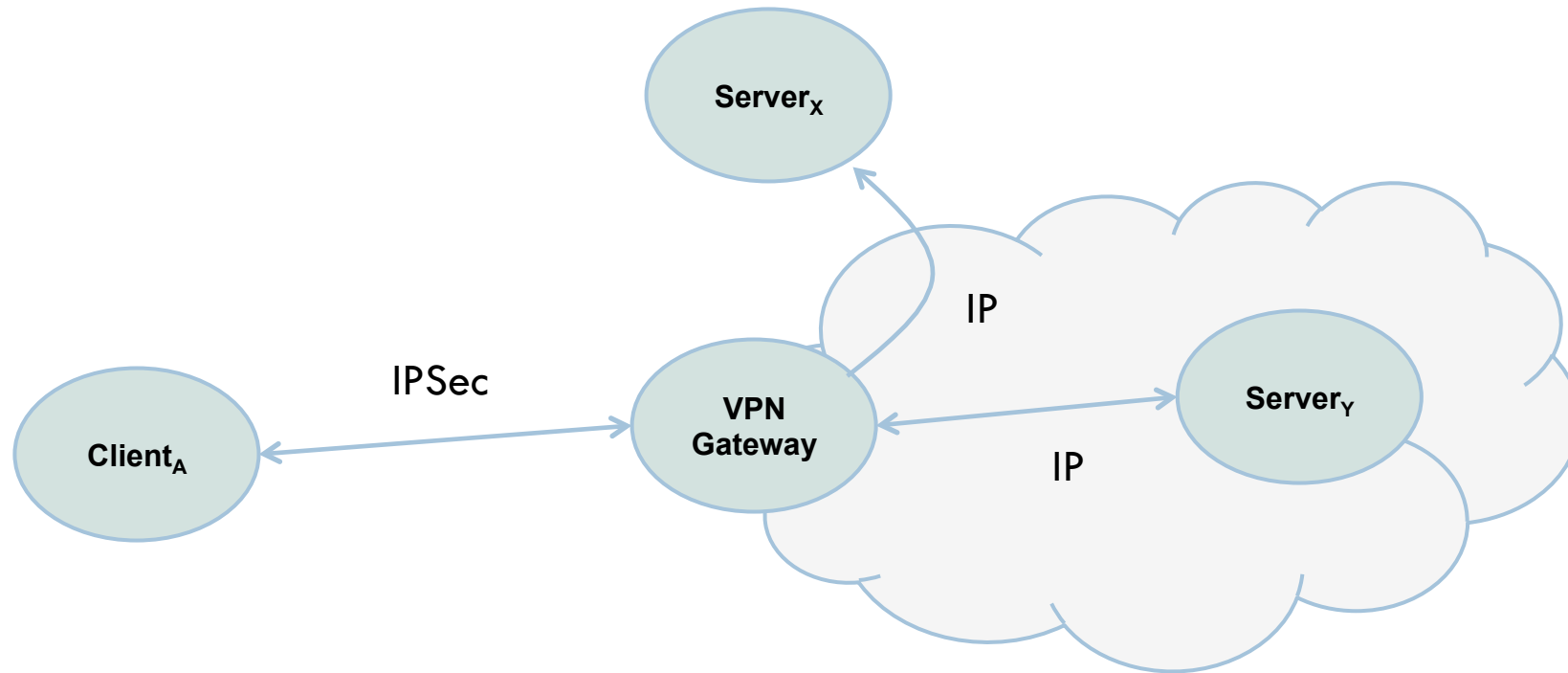


- Now very common for corporations and universities
- Three reasons
 - ▣ Protection of internal services
 - ▣ Giving a different “appearance” to the outside world (e.g. ACM Digital Library)
 - ▣ Security of access from anywhere (no need to trust local network)
- The very easiest way to protect a NVE 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





STREAMING

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

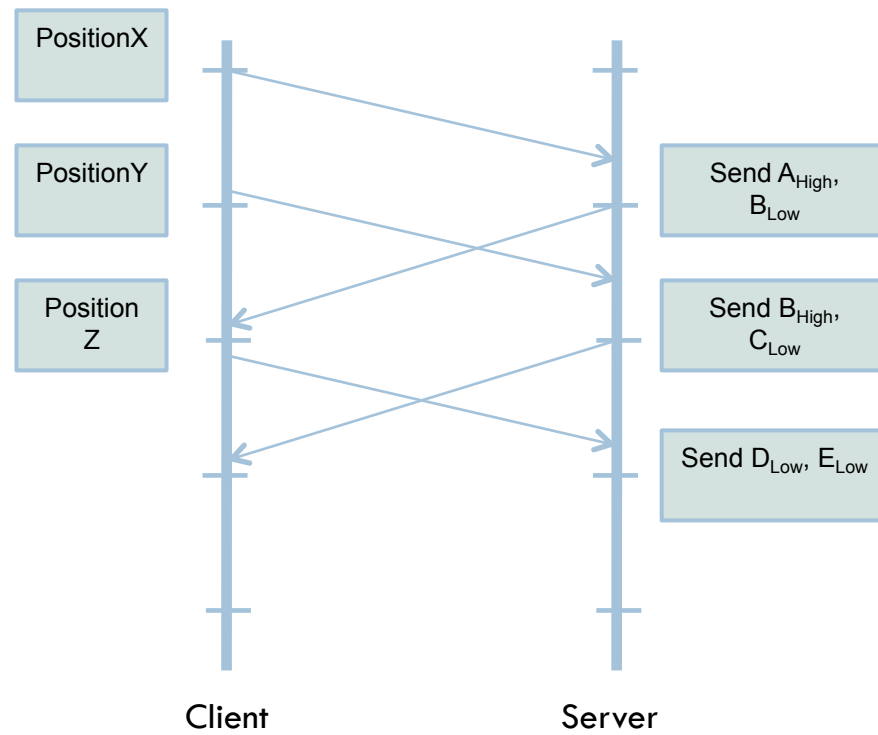


Streaming Geometry

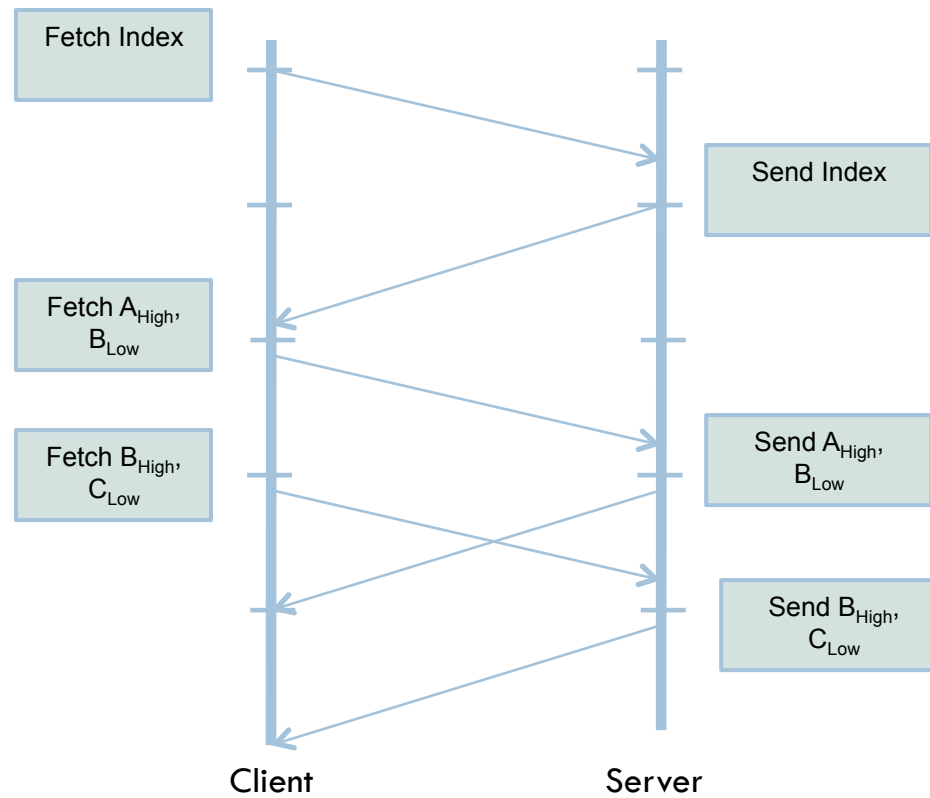


- Many NVEs use very large worlds which need to be downloaded because user modifiable or just vast
- System needs to determine which parts of the models should be transferred
- Typically done in a *priority order* from the viewpoint of the client, e.g. in increasing distance order
- Two ways of doing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

Server Push



Client Pull





PERSISTENT AND TIERED SERVICES

Building a Persistent Service



- Many systems are long-lived and worth money
 - ▣ Second Life
 - ▣ World of Warcraft
- There needs to be a reliable persistent backend
- There needs to be separation of concerns in web-service
- The infrastructure needs to be protected
- A lot of this is just based on good practice for big web services

What Needs to be Persistent



- Player scores!
- In-game currency, in-game asset ownership
- Need proper “database-like” characteristics (ACID principles)
 - ▣ Thus use proper databases over SQL
- World description
 - ▣ Can use database over SQL, probably more custom databases

What Doesn't Need to be Persistent

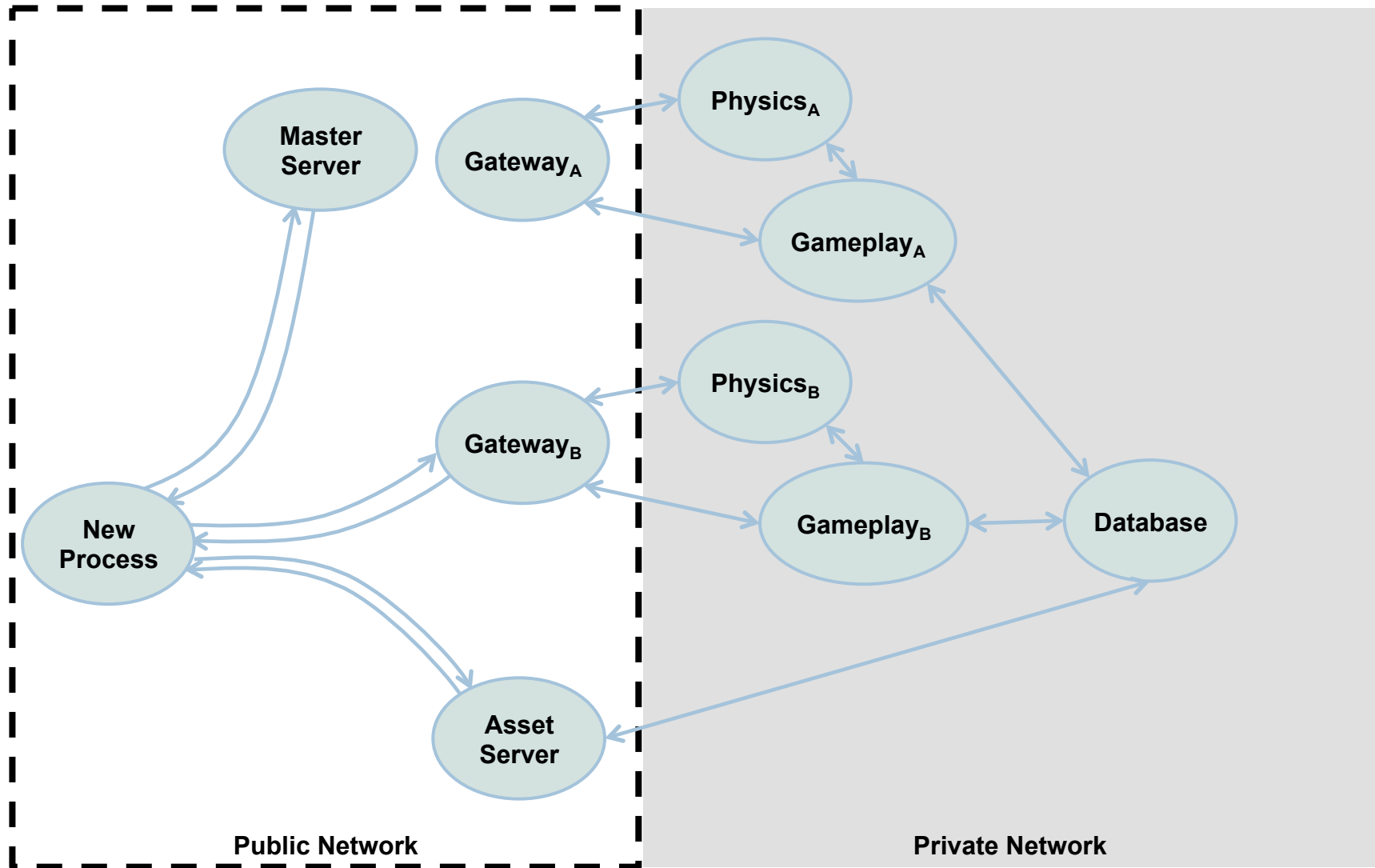


- Actual in-game state (usually)
 - ▣ Users care about outcomes, not state
 - ▣ No need to (e.g.) continuously store players locations to a persistent database
- Commonly assumed that there is a “prototype” world-state that the world can be reset to at any point
 - ▣ If the world crashes, just reload it
 - ▣ This state might be a file on disk

Other Separation of Concerns



- Don't expose database to the raw Internet
 - ▣ Normal "tiered-service"-style approach
- Separate computational processes if required
 - ▣ E.G. Separate physics from rest of game-play
- Sometimes: don't even let clients connect direct to game servers, use a gateway
 - ▣ Allows TCP connections to be kept-alive
 - ▣ Good point to rate-limit
 - ▣ Fair sharing amongst users



Tools

Introduction to Networked Graphics

IEEE Virtual Reality 2011





- *Tools*

- - Middleware

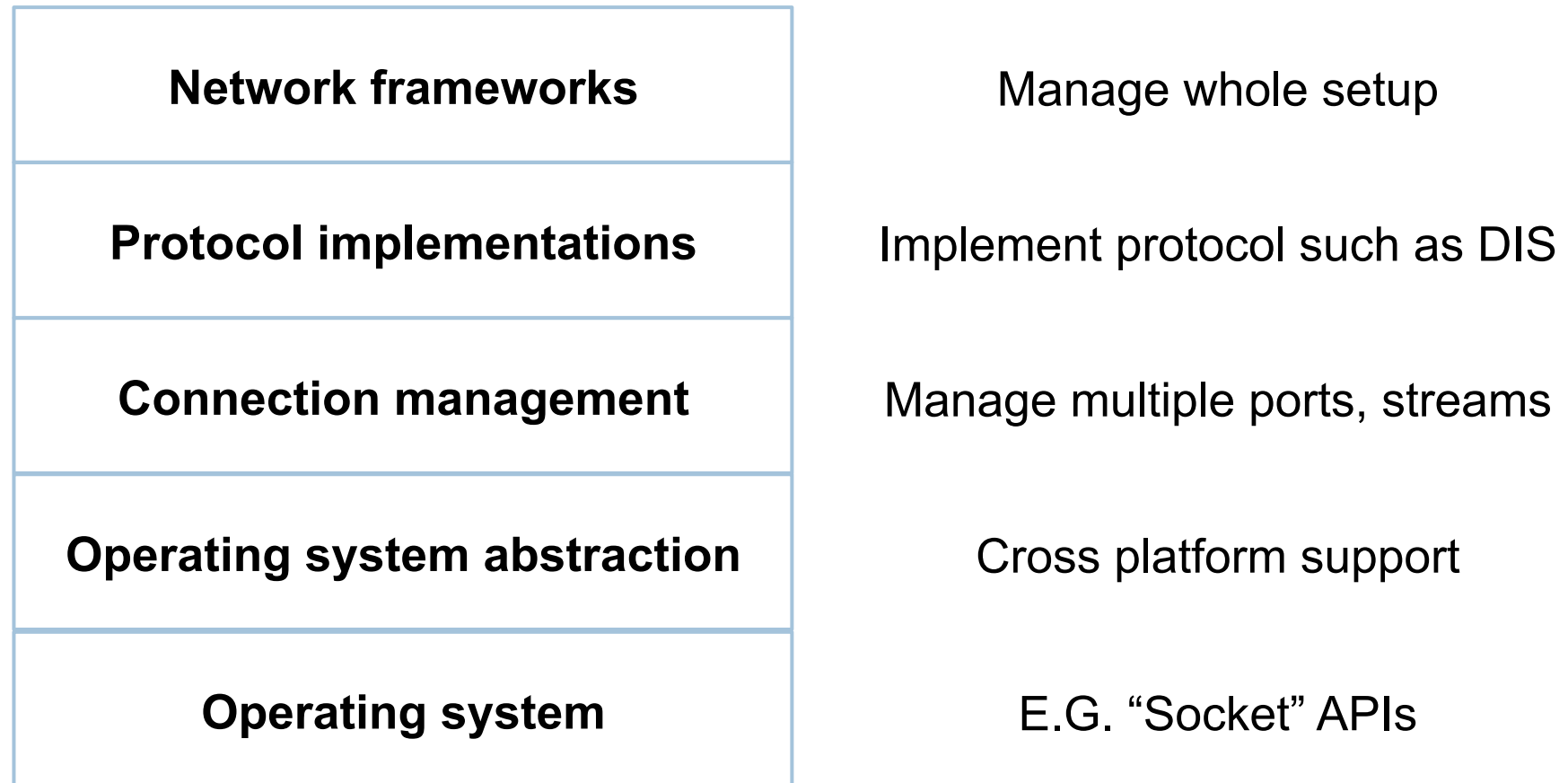
- - Networked engines

- www.networkedgraphics.org has lots of information about tools



ROLE OF MIDDLEWARE

Middleware





LOW-LEVEL SOCKET APIS

Address lookup functions

Function	“Inverse”	Other Helper Functions
getaddrinfo	<code>int getnameinfo(const struct sockaddr *sa, socklen_t salen, char *host, size_t hostlen, char *serv, size_t servlen, int flags);</code>	<code>void freeaddrinfo(struct addrinfo *ai);</code> <code>const char *gai_strerror(int ecode)</code>
gethostbyname	<code>struct hostent *gethostbyaddr(const char *addr, int len, int type);</code>	<code>struct hostent *gethostent(void);</code> <code>struct hostent *gethostent_r(struct hostent *result, char *buffer, int buflen, int *h_errnop);</code> <code>int sethostent(int stayopen);</code> <code>int endhostent(void);</code>
inet_addr	<code>char *inet_ntoa(struct in_addr in)</code>	
inet_pton	<code>char *inet_ntop(int af, const void *src, char *dst, size_t size);</code>	

Patterns of client and server socket use for UDP and TCP

		Application Type	
		Client	Server
Socket Type	UDP	allocate socket <i>create server address</i> sendto	allocate socket <i>create local address</i> recvfrom
	TCP	allocate socket <i>create server address</i> connect send/recv	allocate socket <i>create local address</i> bind listen accept send/recv



DIS

Distributed Interactive Simulation

- DIS's purpose is to inter-connect simulators, typically vehicle simulators in military simulations
- DIS defines a packet, which is sent via UDP, called a PDU (Protocol Data Unit)
- DIS often utilizes *multicast* which is a property of IP where one IP packet can be sent to multiple destinations
 - ▣ This requires UDP-style sending (i.e. no guarantee of receipt)
 - ▣ Assumed that simulators send PDUs periodically, no need for resend

Types of PDU

1 Entity State	24 Designator	45 Areal Object State
2 Fire	25 Transmitter	46 TSPI
3 Detonation	26 Signal	47 Appearance
4 Collision	27 Receiver	48 Articulated Parts
5 Service Request	28 IFF/ATC/NAVAIDS	49 LE Fire
6 Resupply Offer	29 Underwater Acoustic	50 LE Detonation
7 Resupply Received	30 Supplemental	51 Create Entity-R
8 Resupply Cancel	Emission / Entity State	52 Remove Entity-R
9 Repair Complete	31 Intercom Signal	53 Start/Resume-R
10 Repair Response	32 Intercom Control	54 Stop/Freeze-R
11 Create Entity	33 Aggregate State	55 Acknowledge-R
12 Remove Entity	34 IsGroupOf	56 Action Request-R
13 Start/Resume	35 Transfer Control	57 Action Response-R
14 Stop/Freeze	36 IsPartOf	58 Data Query-R
15 Acknowledge	37 Minefield State	59 Set Data-R
16 Action Request	38 Minefield Query	60 Data-R
17 Action Response	39 Minefield Data	61 Event Report-R
18 Data Query	40 Minefield Response	62 Comment-R
19 Set Data	NAK	63 Record-R
20 Data	41 Environmental Process	64 Set Record-R
21 Event Report	42 Gridded Data	65 Record Query-R
22 Comment	43 Point Object State	66 Collision-Elastic
23 Electromagnetic Emission	44 Linear Object State	67 Entity State Update

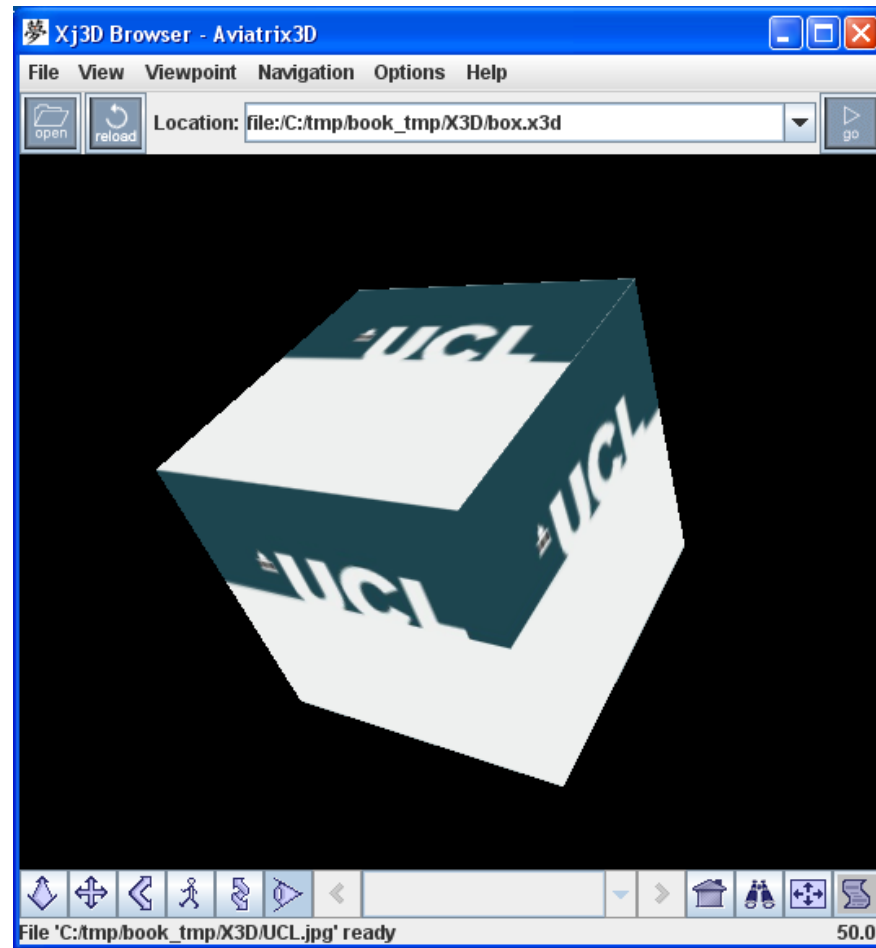
Entity State PDU

Record #	Record Type	Brief Description
1	PDU Header	Header information, including DIS version
2	Entity Identification	An identifier for the entity that this PDU concerns.
3	Force Identification	Which force (Other, Friendly, Opposing, Neutral)
4	Number of Articulation Parameters	Related to sub-parts of entities
5	Entity Type	Kind of entity, country, etc.
6	Alternate Entity Type	Alternative for the above
7	Entity Linear Velocity	Three 32 bit floats
8	Entity Location	Three 64 bit floats
9	Entity Orientation	Three 32 bit floats
10	Entity Appearance	Paint, smoke, etc.
11	Dead Reckoning Parameters	See below
12	Entity Marking	Textual markings on entity
13	Capabilities	Capabilities
14	Articulation Parameters	Related to sub-parts of entities

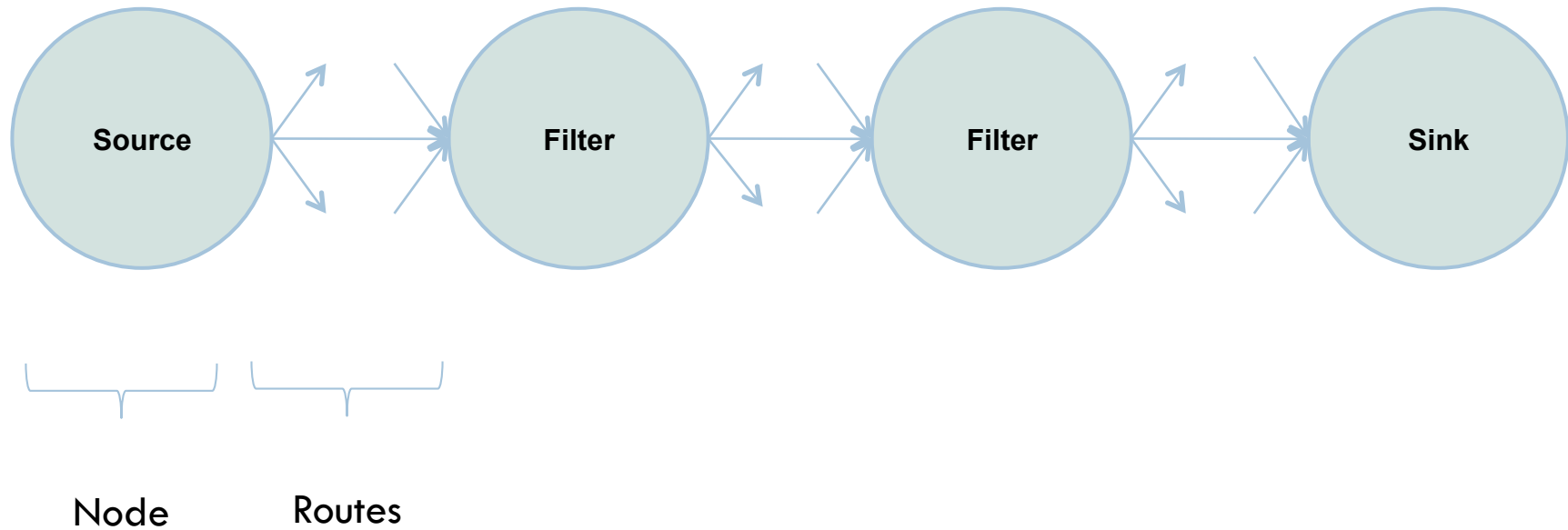


X3D AND DIS

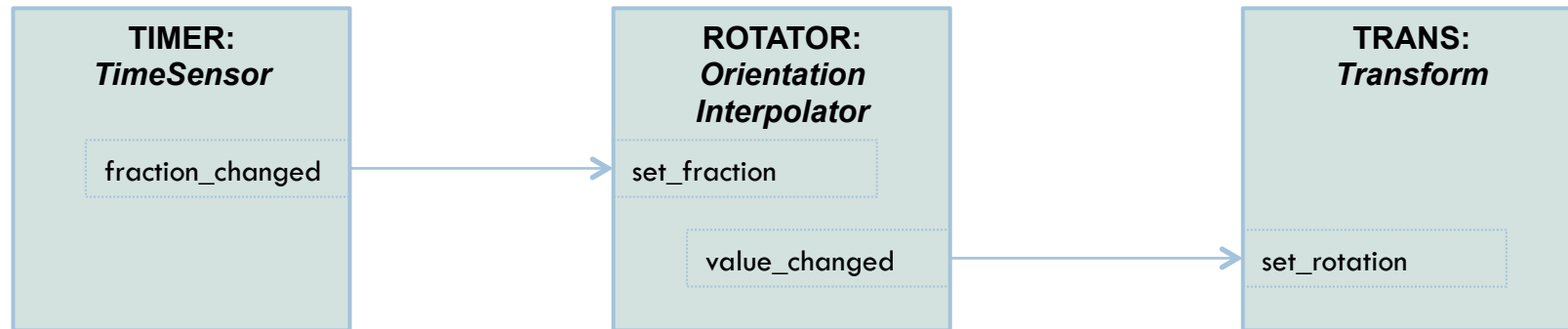
X3D in a Nutshell



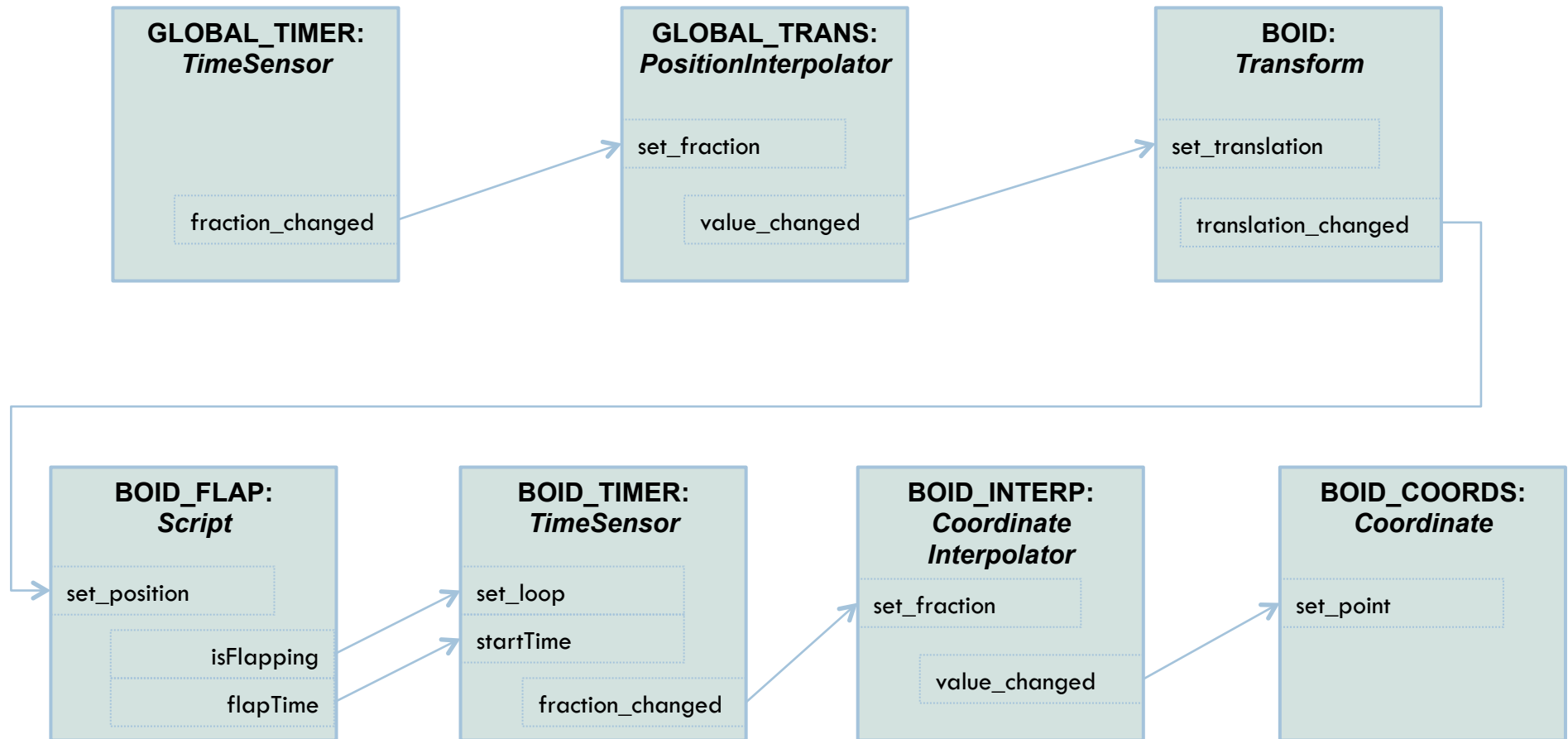
X3D in a Nutshell



X3D in a Nutshell



X3D in a Nutshell





X3D, HAWKNL AND DIS

HawkNL



- A very simple library (Hawk Software) that isolates operating system differences between UNIX and Windows

Idea



- Take a *boids* simulator that simulates a flock of boids
- Have this write one DIS packet to the network per frame per boid
- Have X3D scene listen on the network for these packets and move the boids around

X3D Node (Receiver)

```
EspduTransform {  
    SFString    [in,out] address  
    SFInt32     [in,out] applicationID  
    SFInt32     [in,out] entityID  
    SFString    [in,out] networkMode  
    SFInt32     [in,out] port  
    SFRotation  [in,out] rotation  
    SFVec3f     [in,out] translation  
    SFVec3f     [in,out] linearVelocity  
}
```

HawkNL Code Excerpts (Sender)

```
// Update a set of boids
void sendBoids(std::vector<Boid *> &boids)
{
    static NLulong timestamp=0;
    int i;
    unsigned int count;
    NLbyte buffer[1280]; /* Maximum size of a DIS PDU*/

    for (i=0; i< number_boids; i++)
    {
        <see next slide>
    }
}
```

HawkNL Code Excerpts (Sender)

```
// PDU Header Field
writeByte(buffer, count, 0x06); // Protocol Version Field
writeByte(buffer, count, 0); // Exercise Identifier Field
writeByte(buffer, count, 0x01); // Entity State PDU
writeByte(buffer, count, 0x01); // Entity Information/Interaction
writeLong(buffer, count, timestamp); // Time Stamp Field
writeShort(buffer, count, 0x0090); // PDU Length Field
writeShort(buffer, count, 0x0000); // Padding Field

...

//Entity Location Record, Note the rotation to DIS coordination
writeDouble(buffer, count, boids[i]->pos[0]);
writeDouble(buffer, count, boids[i]->pos[2]);
writeDouble(buffer, count, -boids[i]->pos[1]);

...
```

Xi3D Browser

The screenshot displays the Xi3D Browser software interface, which is used for viewing and interacting with 3D models. The interface is divided into several main sections:

- 3D Viewer:** Located on the left, it shows a 3D perspective view of a checkered floor and a blue sky with white clouds. A small 3D coordinate system is visible in the center of the floor.
- XML Editor:** The central pane displays XML code for a 3D scene. It includes protoinstances for 'Boid' and 'EspduTransform' entities, along with route definitions for translation and rotation changes.
- Palettes:** On the right side, there are several palettes for object manipulation:
 - X3D Metadata and Structure:** Shows a tree view of the scene's metadata, including scene, head, component, and various metadata types.
 - Geometry: Primitives:** Lists basic 3D shapes like Box, Cone, Cylinder, Sphere, and Text.
 - Grouping:** Options for Group, StaticGroup, and Transform.
 - Viewing and Navigation:** Controls for Viewpoint, NavigationInfo, Anchor, Billboard, Collision, OrthoViewpoint, and ViewpointGroup.
 - Appearance, Material and Textures:** Options for Appearance, Material, TwoSidedMaterial, FillProperties, LineProperties, TextureProperties, ImageTexture, MovieTexture, and PixelTexture.
- DIS ESPDU Test Panel:** Located at the bottom right, it provides controls for Translation (X, Y, Z axes) and Rotation (roll, pitch, yaw) in degrees or radians. It also includes DIS Settings for address, port, site ID, application ID, and entity ID.
- DIS Player-Recorder Window:** At the bottom left, it shows a list of entity states with timestamps.
- PDU Header:** A panel at the bottom center showing details for the current PDU, including time stamp, exercise ID, PDU type, and entity ID.
- Entity Marking:** A panel showing the marking hex value for the selected entity.
- Entity Type:** A table showing the domain, country, category, and sub-category for the entity.
- Alternative Entry Type:** A table showing alternative entry types.
- Entity Linear Velocity:** A field for the entity's linear velocity.
- Entity Location:** A field for the entity's location.

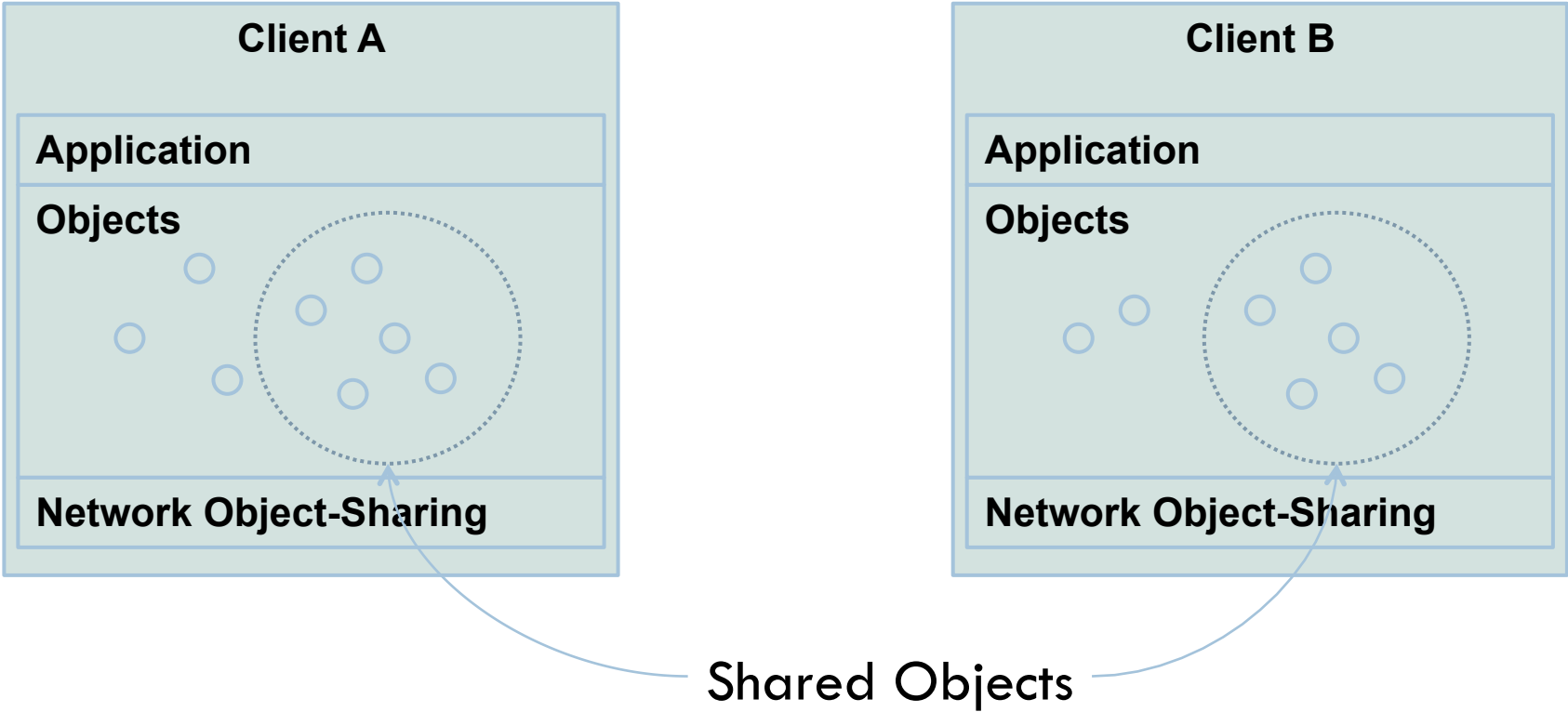


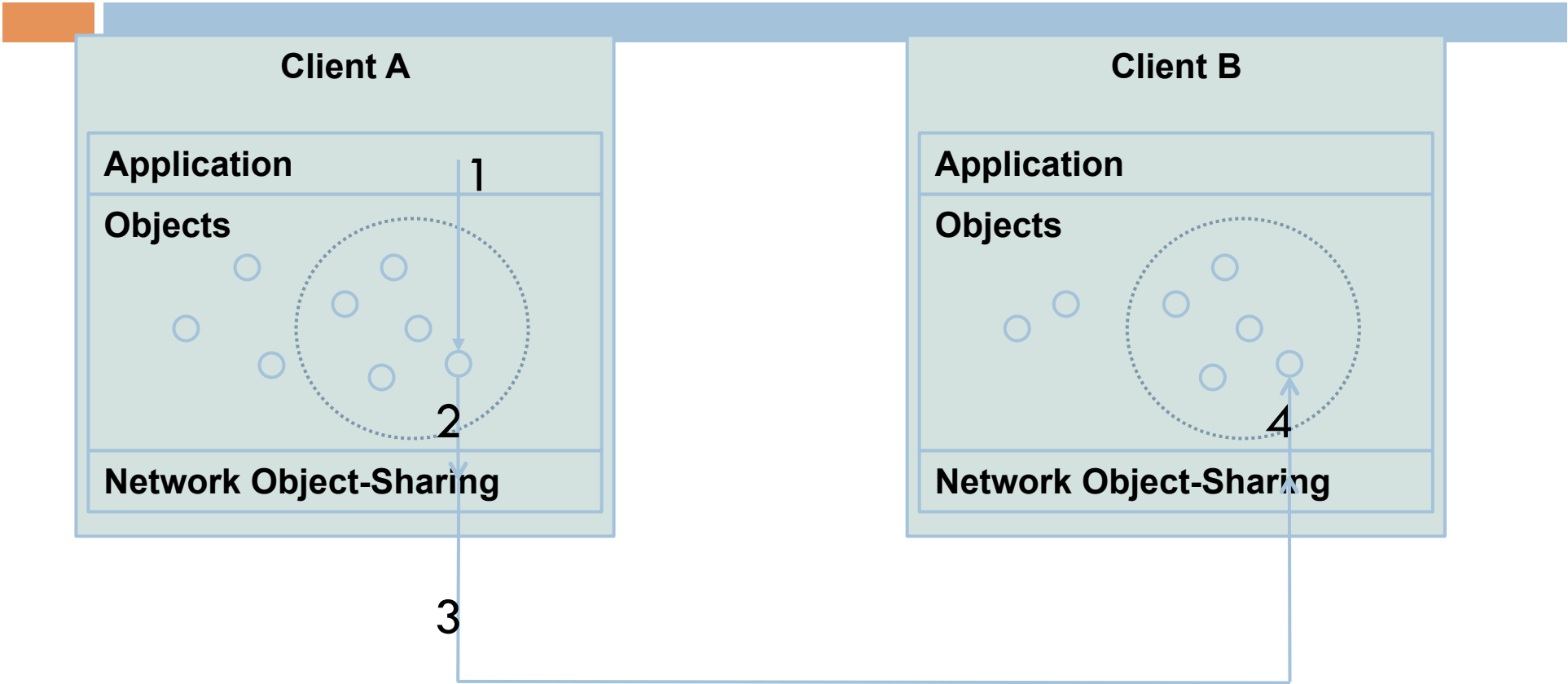
OBJECT SHARING SYSTEMS

Object Sharing Systems



- Principle of object sharing systems is that the client processes access locally stored objects (e.g. instances of C++ classes)
- Any changes to these classes (i.e. changing instance variables, creating new classes) is automatically propagated to other collaborating clients
- Fits extremely well with the scene-graph paradigm in graphics
- In our experience, it is an extremely easy way to get started in network programming







RAKNET

RakNet



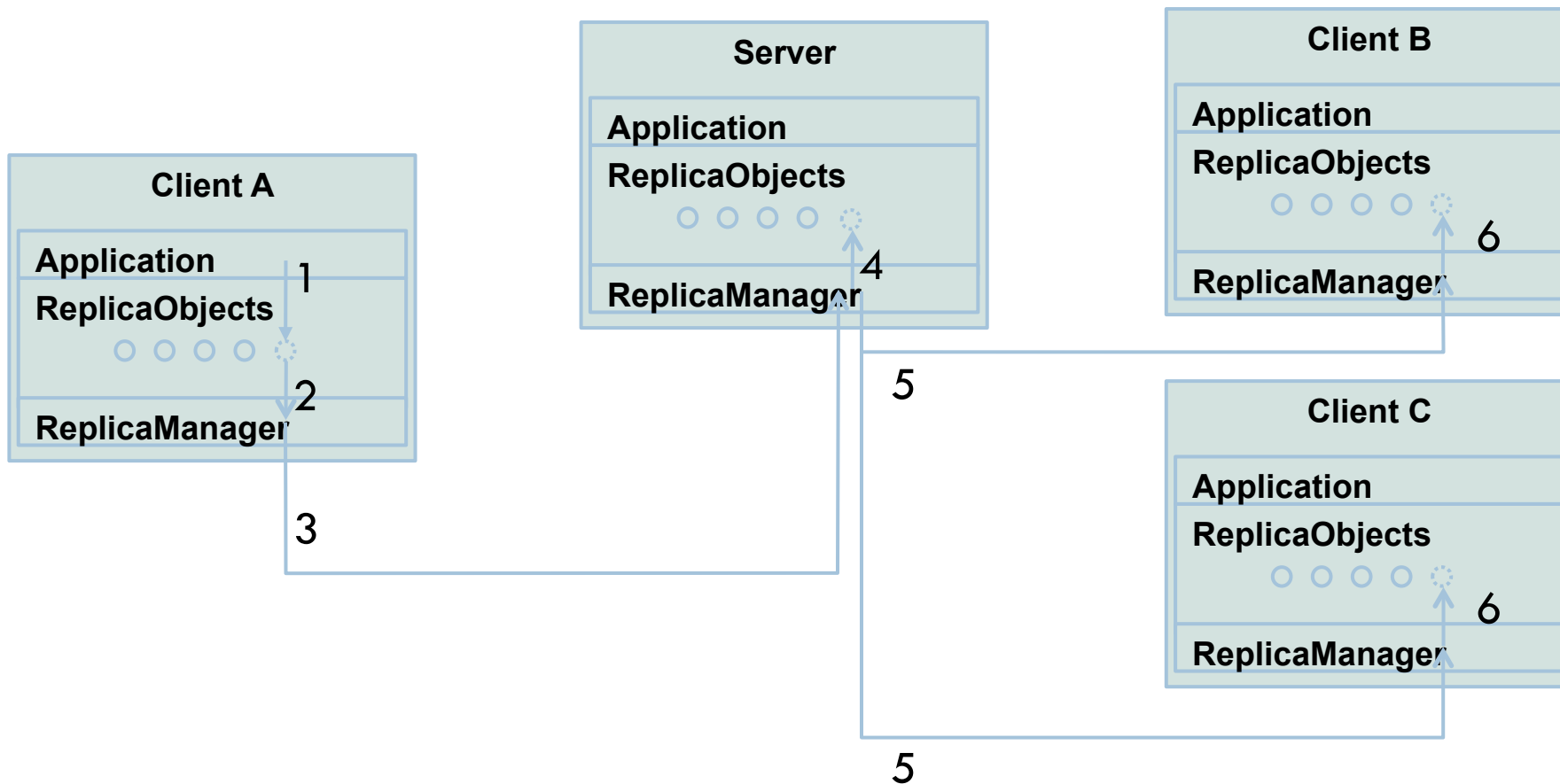
- RakNet is a very popular middleware for NGs and NVEs
- Free for non-commercial use, constantly updated and used in many commercial projects
- Provides functionality at all levels of middleware stack
 - ▣ OS abstraction through to examples of full network system management, lobbies for games, scoring, etc.

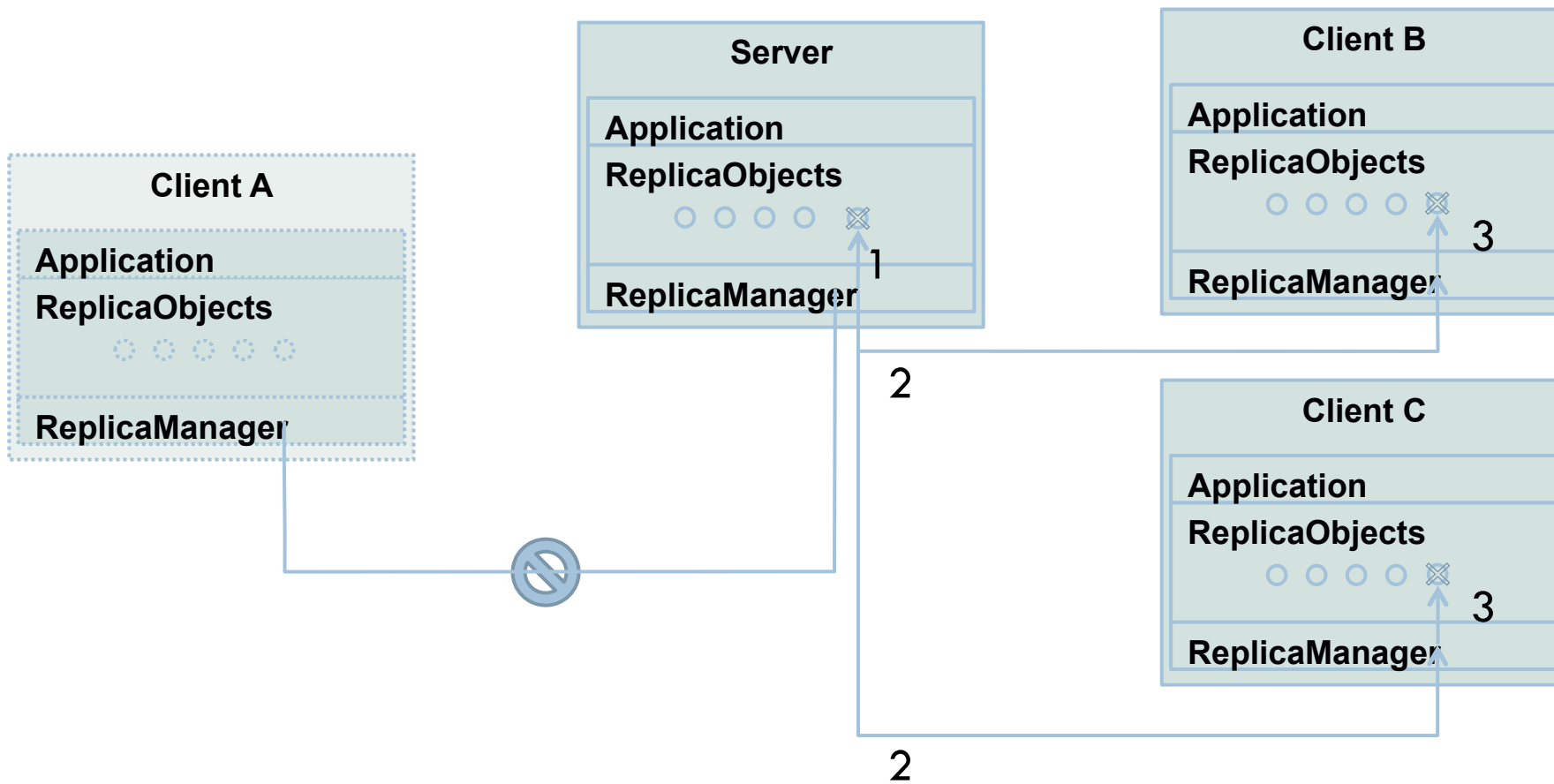
Object Sharing Considerations



- ❑ Need to get frequency of updates right: don't send an update every time an instance variable sends
- ❑ Objects are typically owned by the process that created them: they share the fate of that process

- ❑ Can be client-server or peer to peer
- ❑ RakNet supports different configurations and different styles – worthwhile to experiment!





Research Issues

Introduction to Networked Graphics

IEEE Virtual Reality 2011





- *Research Issues*

- - Scalable peer-to-peer, thin clients

- - Standards, etc.

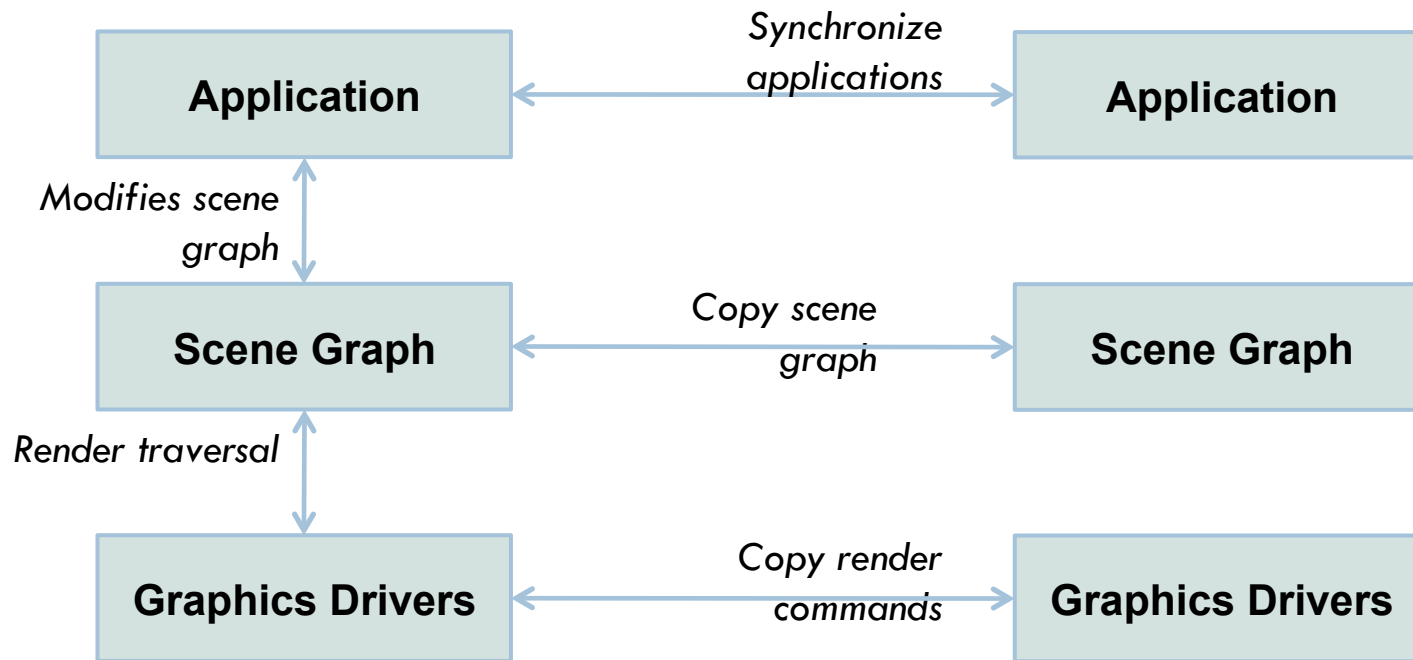


CLUSTERS

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



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

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

Operations



- 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)

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

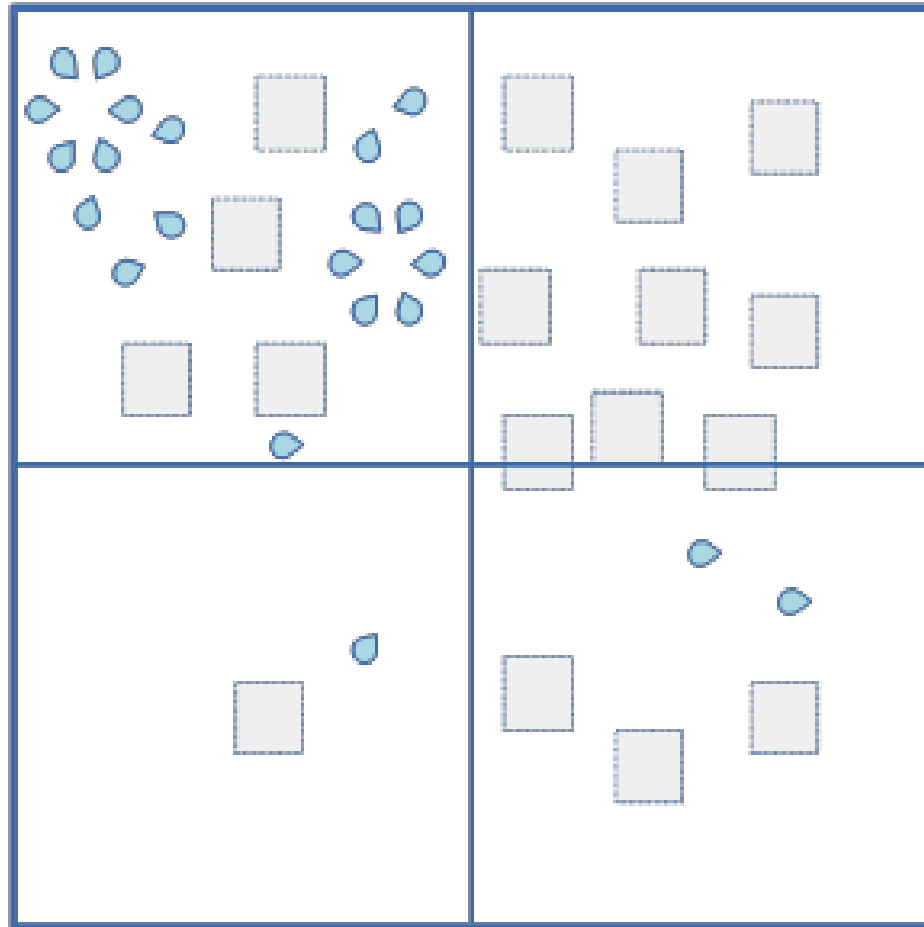


ADAPTIVE NETWORKS

Practical Scalability



- Most deployed systems use a static partitioning of the users on to servers or communication groups
- Pros
 - ▣ A static partition is easy to maintain!
 - ▣ Server can be customized for the expected function or load
- Cons
 - ▣ Users will congregate and occasionally protest by trying to crash servers
 - ▣ Average server load may be low

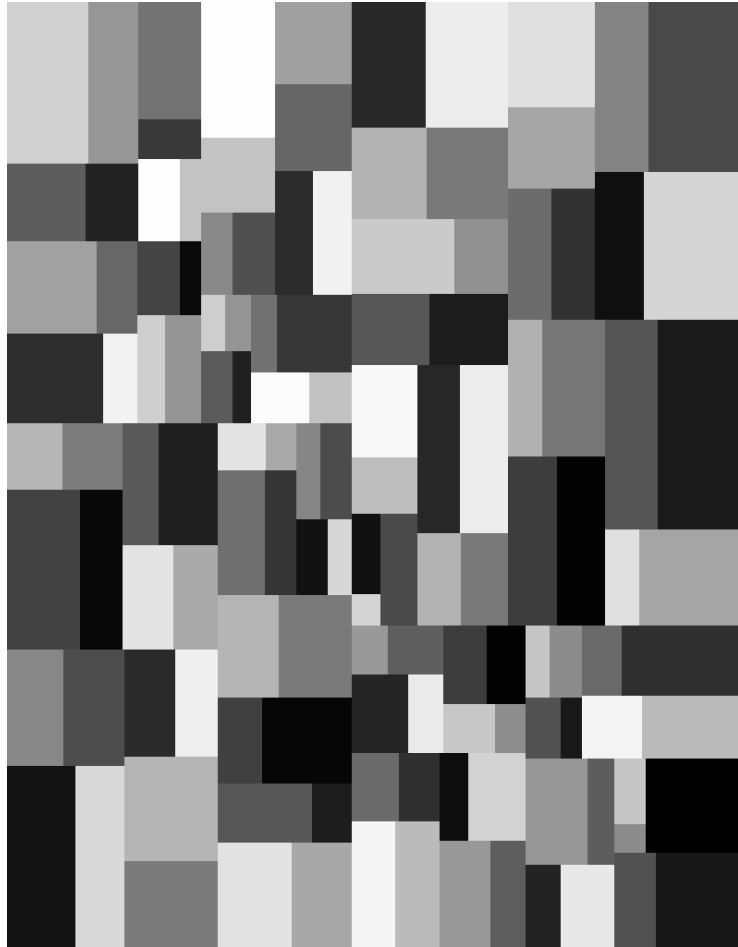


Adaptive Design



- Can make a better static partition by reallocating servers infrequently depending on actual user usage of the system
- Example: repartitioning a hypothesized service for central London depending on pedestrian service

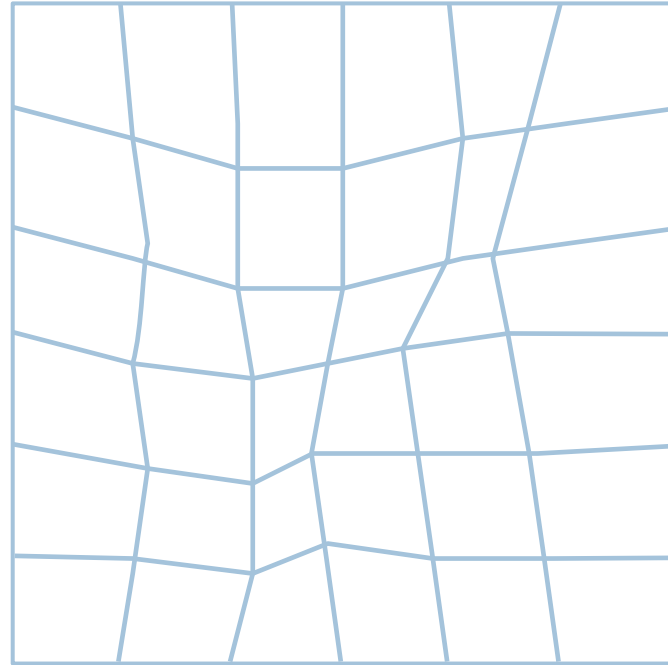
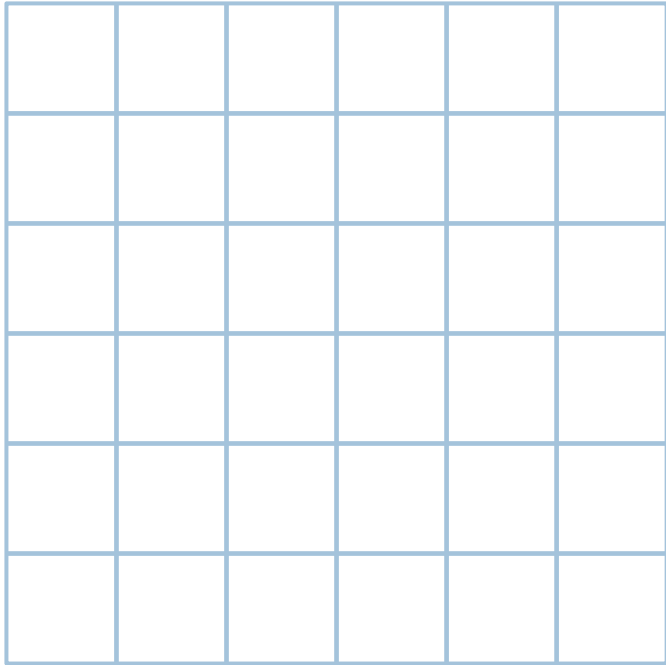


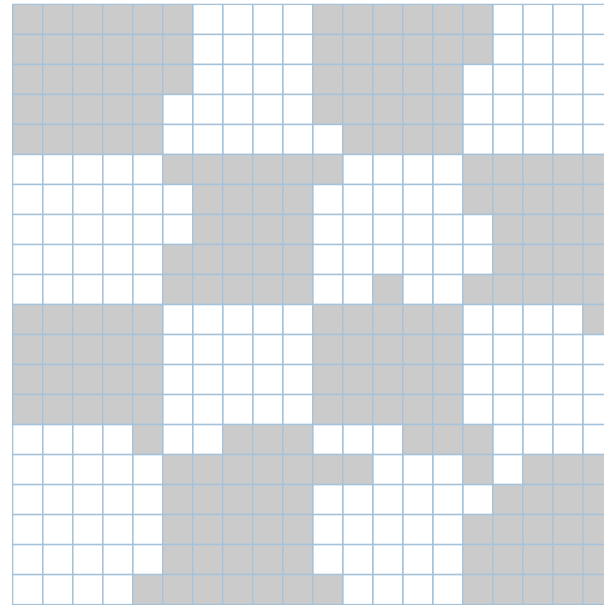
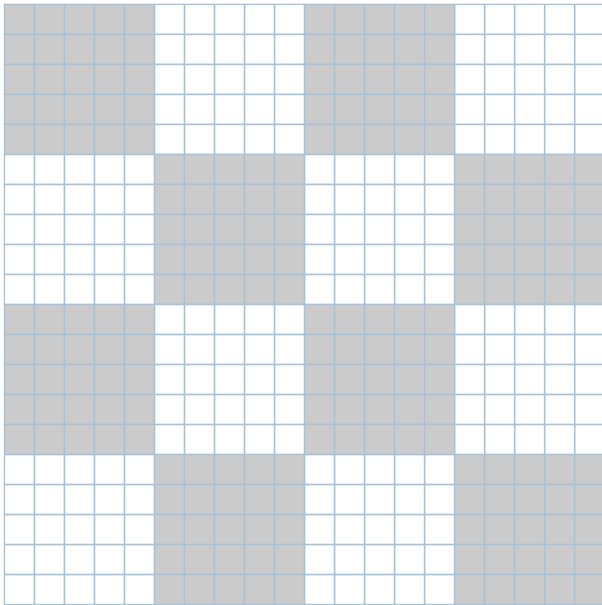


Other Strategies



- Active area of research: given a partitioning, how to reallocate users or regions to servers as load changes
- Local refinement of scope of server







PEER TO PEER

Peer to Peer



- A very 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?

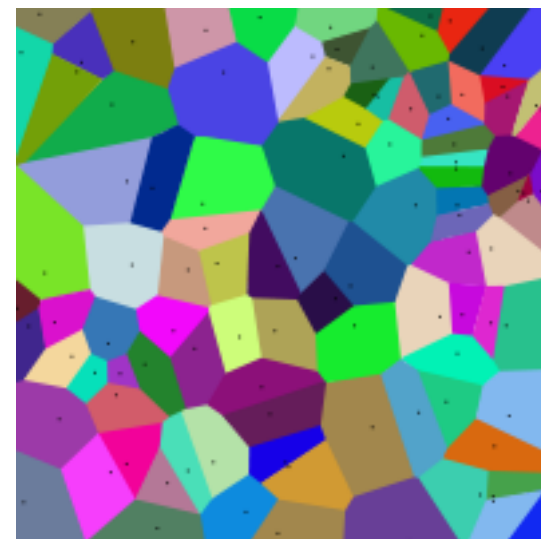
Larger Peer to Peer Context



- 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 ($O(\log N)$) 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





STANDARDS

Potential Standards



- OpenSim project shows that its possible to construct user-maintained “grids” of servers
 - ▣ OpenSim implements the server functionality of Second Life. It complements the open source version of the viewer from Linden Labs
- NVEs in browsers will emerge from WebSockets/
WebGL


Open Standards



- X3D hasn't reach critical mass, may be over-taken by WebGL (though you can use X3D on WebGL)
- Notably different requirements than MPEG which has networking components including streamed 3D (based on VRML97)
- ?



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

Closing Remarks

Introduction to Networked Graphics

IEEE Virtual Reality 2011



Much More to Study



- Nothing like hands on experience to reveal why Networked Graphics is a unique field
- As a field, needs to learn lessons from Internet standards and web standards
- An NVE or NG has a complex set of requirements and thus networking needs
- BUT lots of the technology is readily available in middleware