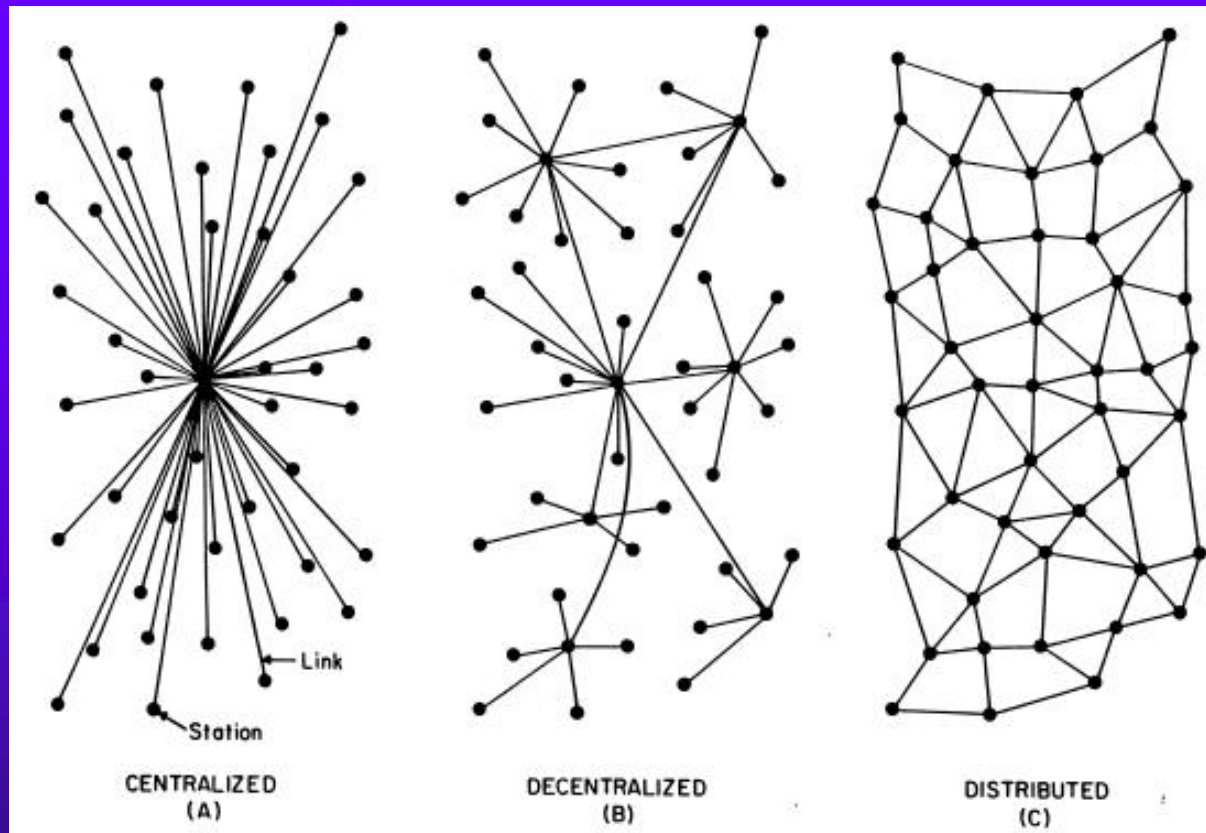


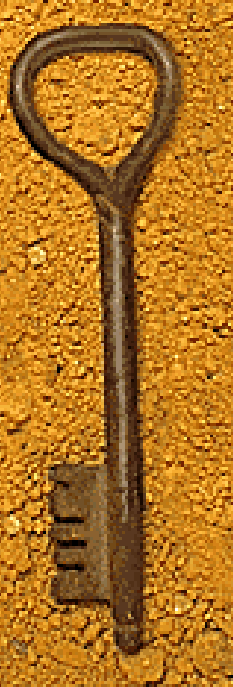
Distributed Interactive Applications



Diego Montenegro

Andrew Park

Bobby Vellanki



What is a Distributed Interactive Application (DIA)?

“A DIA allows a group of users connected via a network to interact synchronously with a shared application state.”

“DIS is the name of a family of protocols used to exchange information about a virtual environment among hosts in a distributed system that are simulating the behavior of objects in that environment. It was developed by the US department of defense to implement systems for military training, rehearsal, and other purposes.”

Properties of the Black Box

Consistent

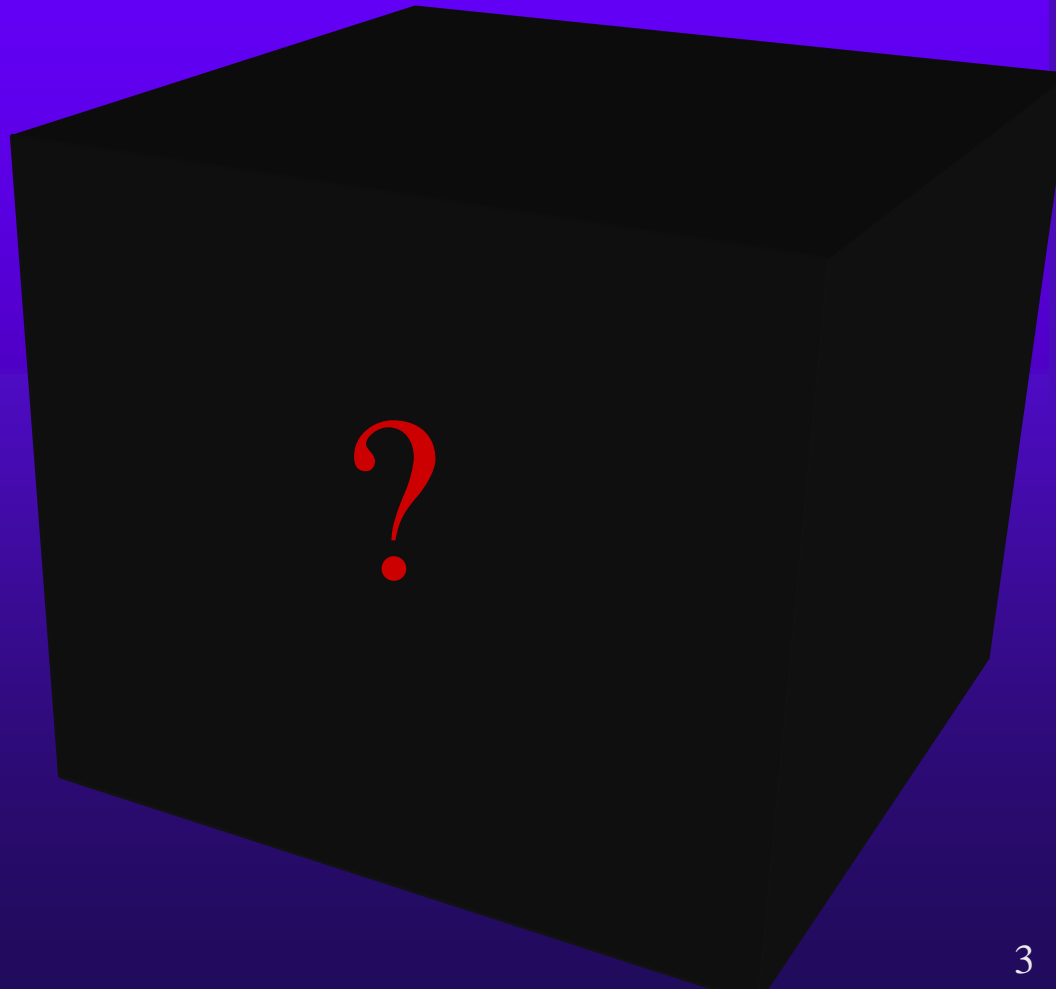
Scalable

Secure

Robust

Available

Real-Time





Consistency

Every entity must have the same view of the global state as every other entity in the entire network

Scalability

An increase in users does not affect the efficiency of the network

Security

No node can have advantage over another node



Robustness

A failure of any participant has no effect on any other participant

Availability

The network is perpetually accessible

Real-Time

Processes are delivered no later than the time needed for effective control

Properties of the Black Box

Consistent

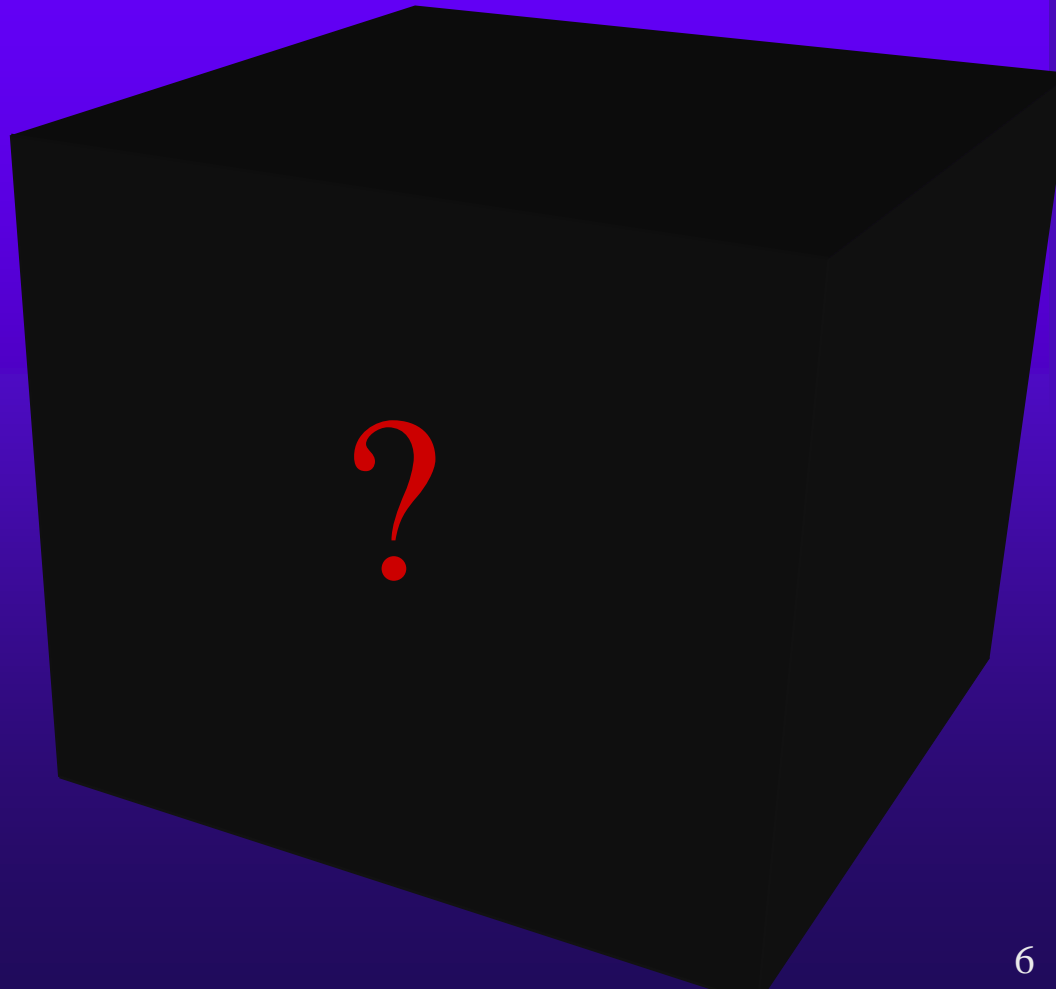
Scalable

Secure

Robust

Available

Real-Time



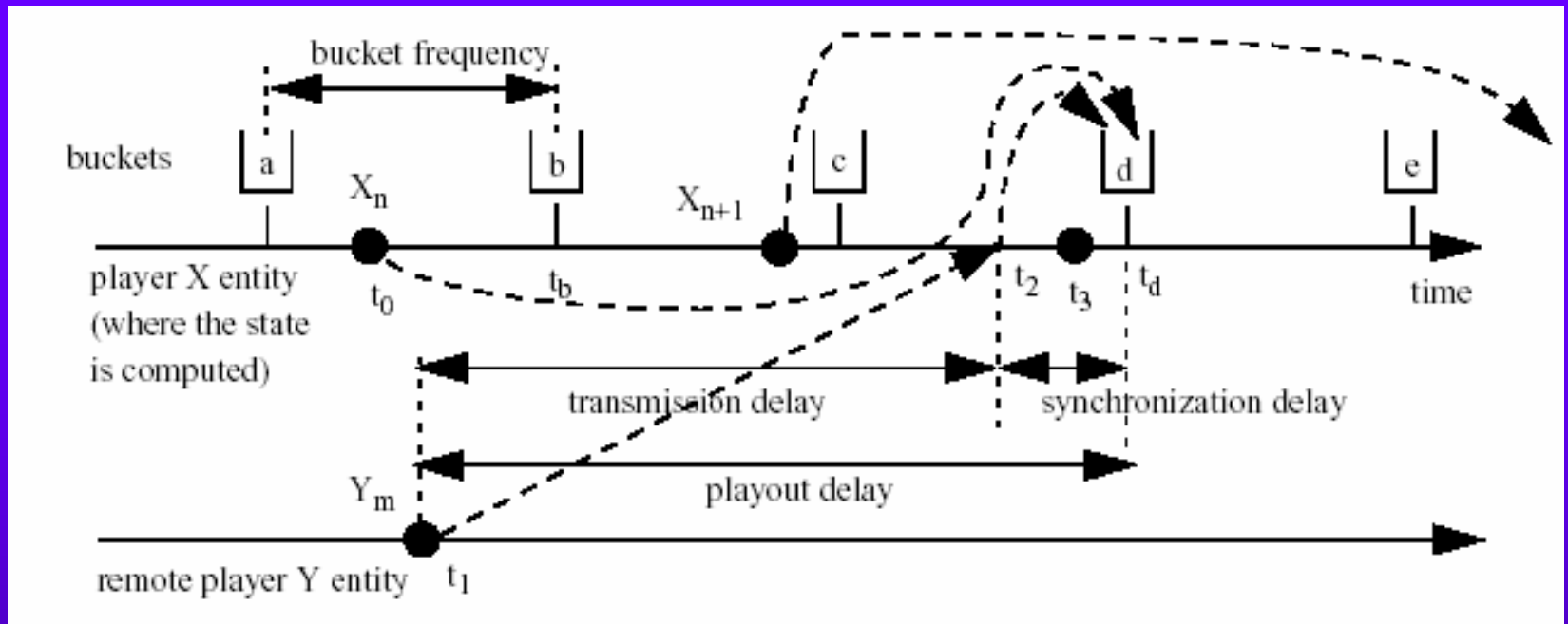


Consistency

➤ *Bucket Synchronization*

- Dead Reckoning
- Algorithms

Bucket Synchronization Mechanism



- All calculations are delayed until the end of each cycle
- The bucket cycles are typically 100ms (bucket frequency)
- Bucket frequency is set as a constant value which is equal to the rate that a human vision perceives smooth motion.



Consistency

- Bucket Synchronization
 - *Dead Reckoning*
- Algorithms



Dead Reckoning

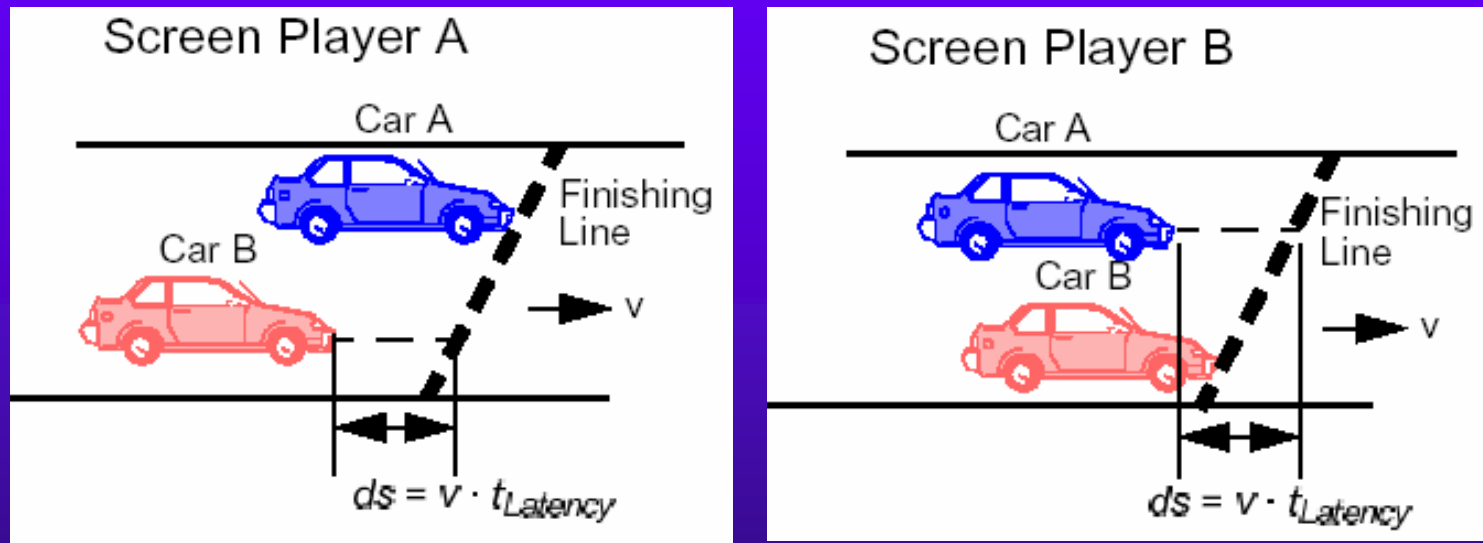
What is Dead Reckoning?

- If a packet is lost or received too late, dead reckoning is used to estimate the “most probable” state or position of the object.
- The success of Dead Reckoning is based on the intelligence of the algorithm design
- There is inconsistency between the actual and expected states.



Dead Reckoning

Why is Dead reckoning needed?

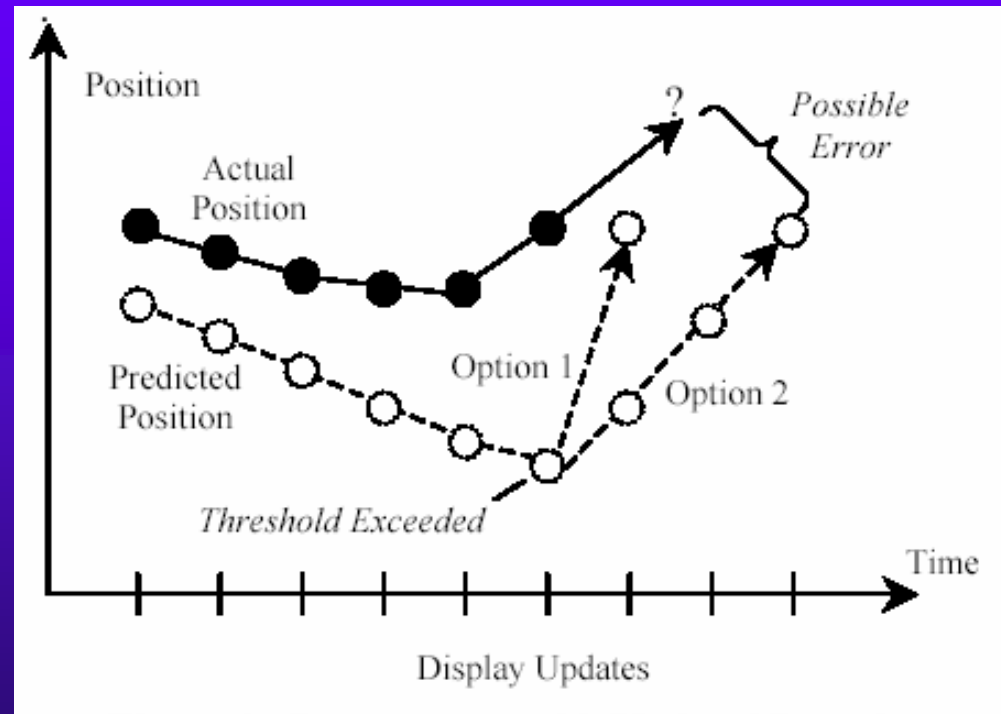


Example for Delay-Induced Inconsistency



Dead Reckoning

Immediate Convergence vs Time-Phased Convergence

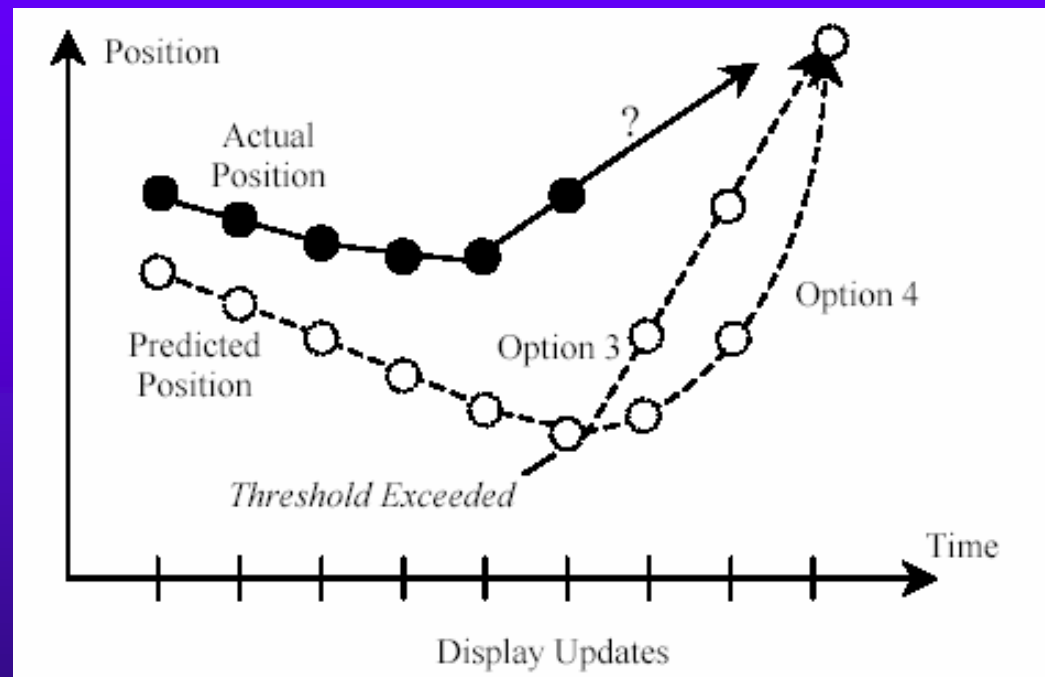


Convergence with Updated Location



Dead Reckoning

Linear Convergence vs Curve-Fitting Convergence



Convergence with Predicted Location



Consistency

- Bucket Synchronization
- Dead Reckoning

➤ *Algorithms*



Dead Reckoning Algorithms

1. Simple scheme - Use previous packet
2. Function of the previous packet's position, velocity of the object, and the elapsed time.
3. Extrapolation
4. Pre-Reckoning
 - Compatibility with Bucket Synchronization

Properties of the Black Box

Consistent

Scalable

Secure

Robust

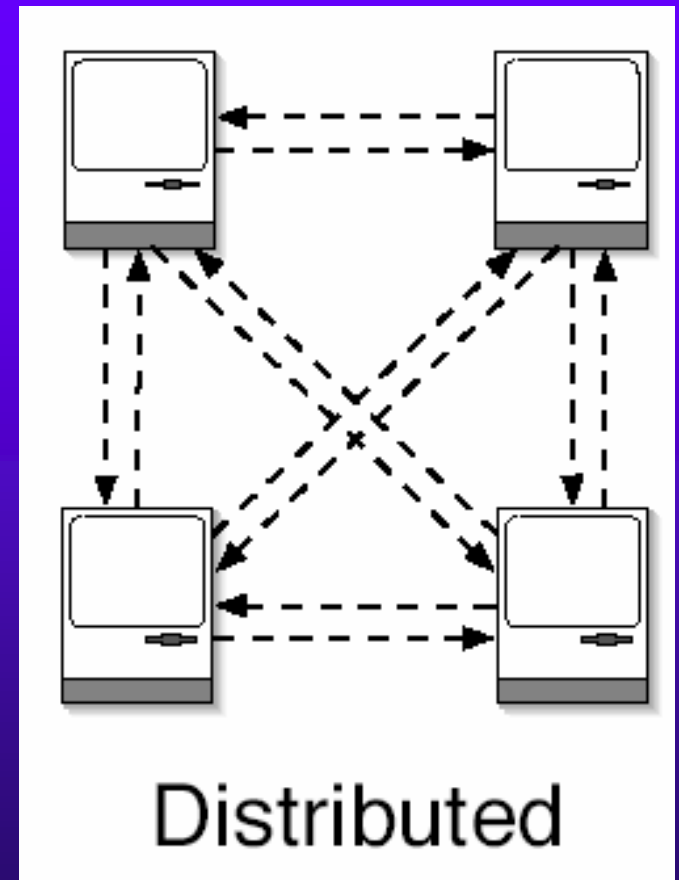
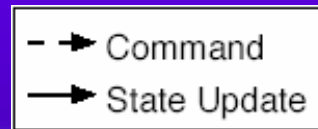
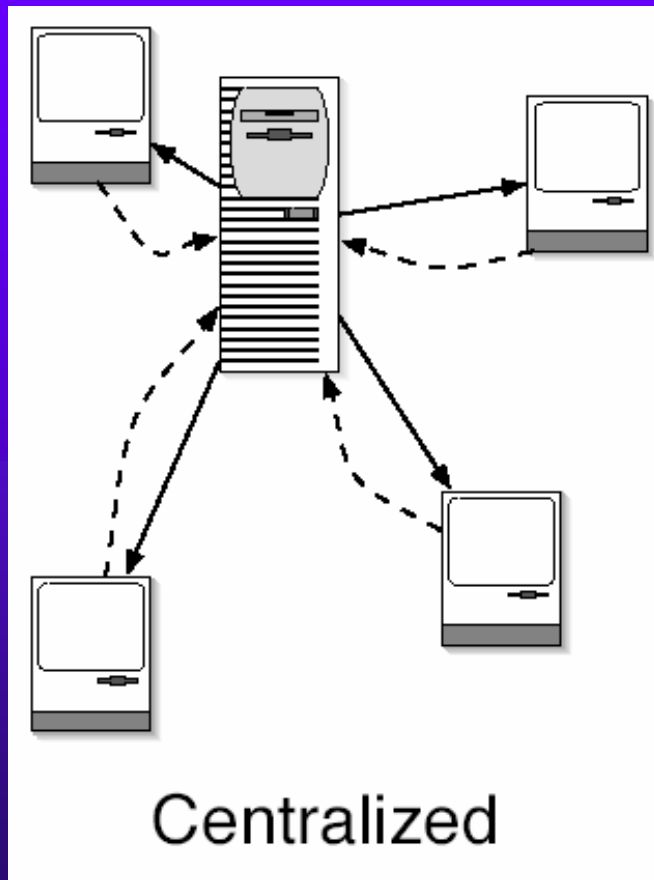
Available

Real-Time



Scalability

Centralized vs. Distributed





Centralized

Pros:

- Simplified administration
- Ease of maintenance
- Ease of locating resources

Cons:

- Difficult to scale
- High cost of ownership
- Little or no redundancy
- Single point of failure



Distributed

Pros:

- Highly extensible and scalable
- Highly fault tolerant
- Dynamic addition of new resources

Cons:

- Difficulty in synchronizing data and state
- Scalability overhead can be large
- Extremely difficult to manage all resources

Properties of the Black Box

Consistent

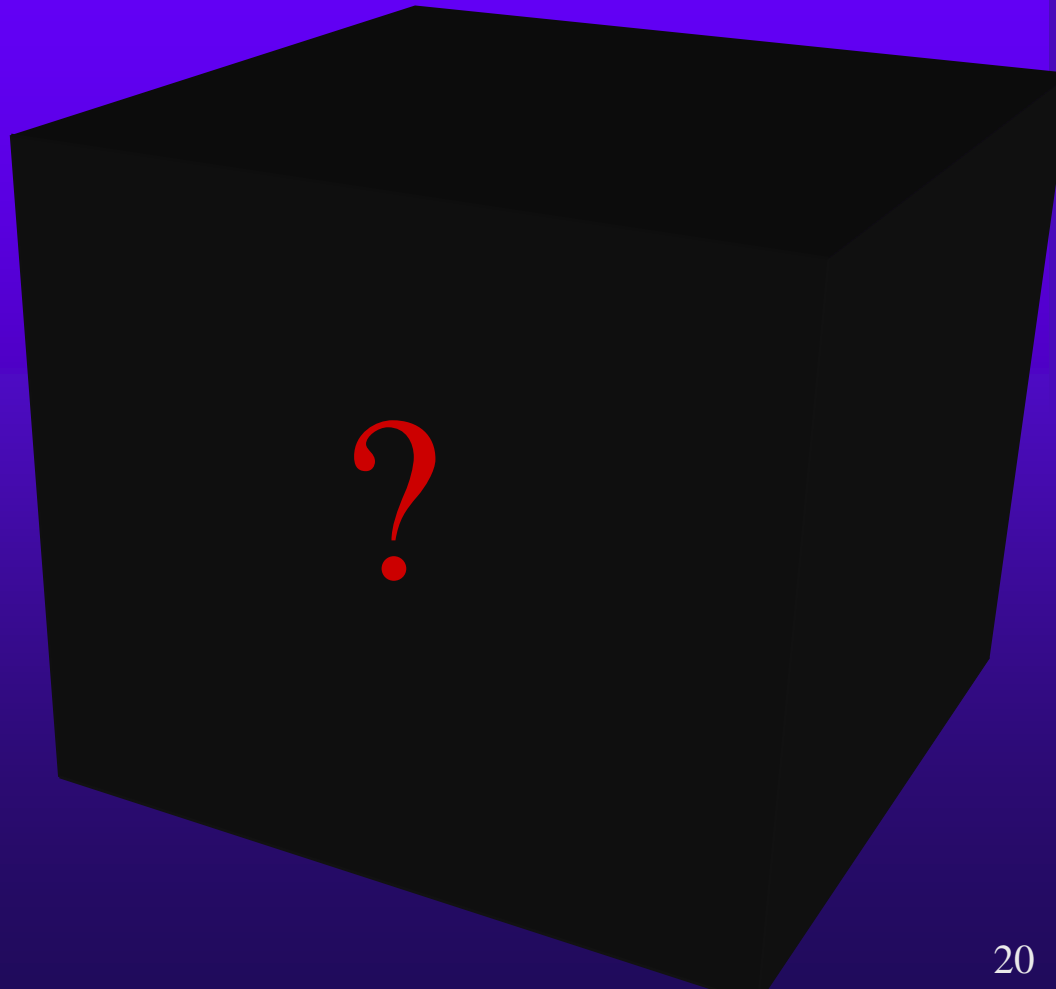
Scalable

Secure

Robust

Available

Real-Time





Security

- Distributed applications are more prone to cheating than centralized due to the fact that there is no authority supervising the actions of the users
- Security bears a trade-off of efficiency vs. fairness



Cheating

- **Suppress-correct cheat :**
Host gains advantage by purposefully dropping update messages.
- **Look ahead cheat :**
Players makes decision after receiving all updates from participating players.



Cheating Solutions

- A) **Lockstep Protocol** : No host receives the state of another host before the game rules permit
1. Player decides but does not announce its turn $t + 1$
 2. Each player announces a Cryptographically secure one-way hash of its decision as a commitment.
 3. After all players have announced their commitments, players reveal their decisions.
 4. Each host can verify revealed decisions by comparing hashes.



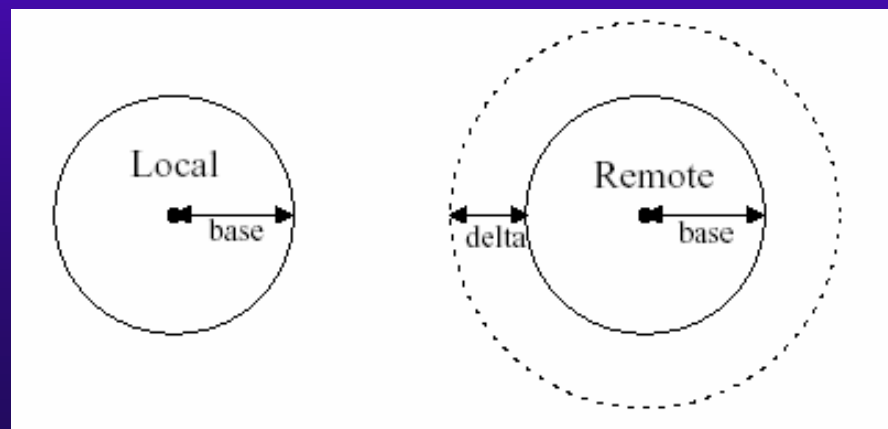
Cheating Solutions

- B) **Asynchronous Synchronization** : Relaxes the requirements of lockstep synchronization by decentralizing the game clock
1. Player determines its decision for the turn and announces the commitment of the decision to all players.
 2. Commitments that are one frame past the last revealed frame of a remote player are accepted.
 3. Before revealing its commitment, the local player must determine which remote players it is waiting for.

Cheating Solutions

B) Asynchronous Synchronization Continued :

4. A remote player is not in the wait state only if there is no intersection with the SOI dilated from the last revealed frame of the remote player.
5. The SOI is calculated using the base radius of the last known position plus a delta radius.



Cheating Solutions

B) Asynchronous Synchronization Continued :

6. Finally, if no remote hosts are in the wait state, the local host reveals its state turn for turn t , updates its local entity model of each other player with their last known state, including the remote hosts last known time frame and advances to the next turn.

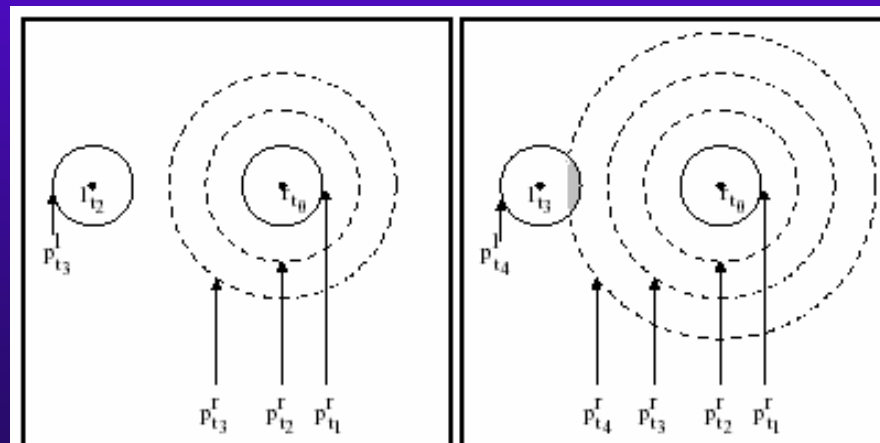


Fig. 8. (Left) Dilation to t_3 . (Right) Dilation to and intersection at t_4 .



Cheating Solutions

- C) **AS with Packet Loss:** players can skip missing packets and accept new, out-of-order packets from other players when the missing packets represent state outside a SOI intersection. Missing packets that represent intersection of SOI cannot be dropped or skipped.

AS represents a performance advantage over lockstep, rather than contact every player every turn, players need only contact players that have SOI intersection.

Downside of all previous protocols:

- **Performance Penalty:** All nodes must slow down to the speed of the slowest user.

Properties of the Black Box

Consistent

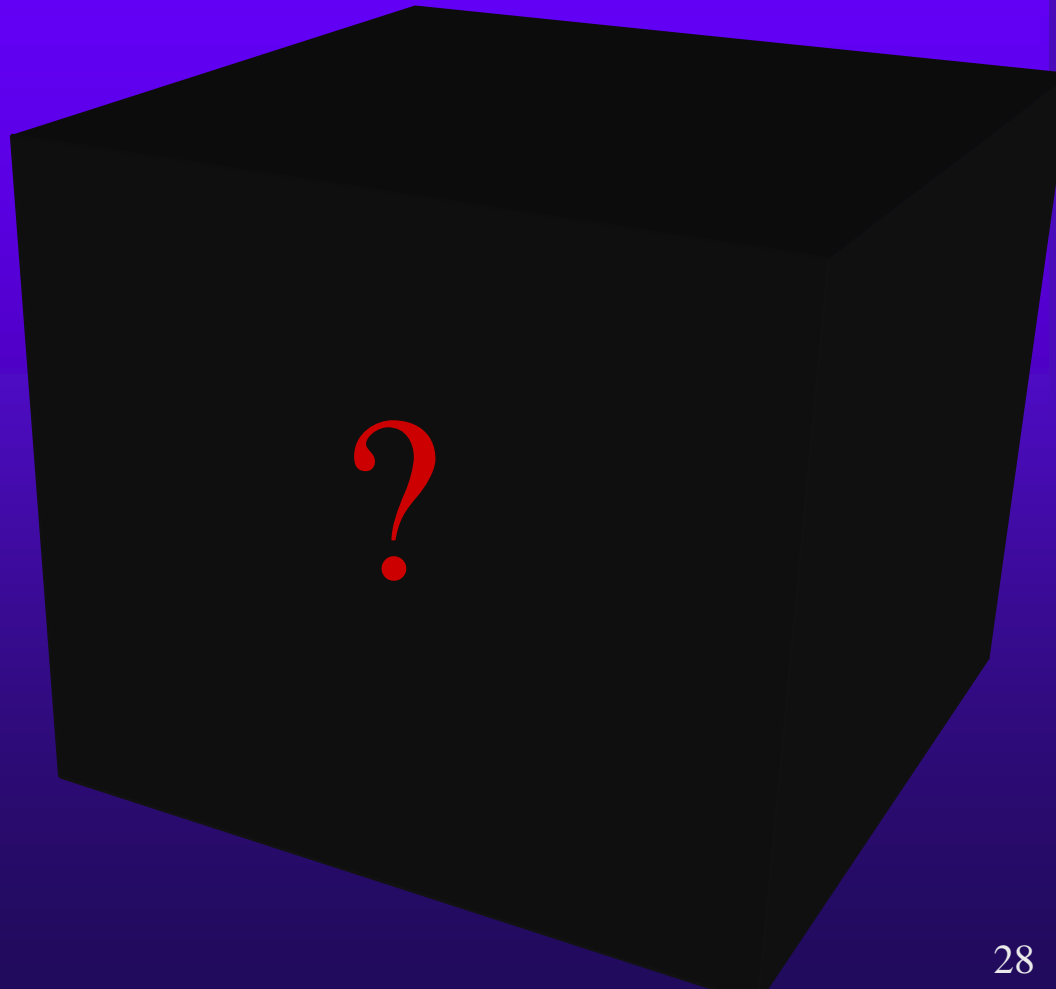
Scalable

Secure

Robust

Available

Real-Time





Robustness

- Users can join or leave the network at any time, without having any negative effects on other nodes.
- A failure of any participant has no affect on any other participant.
- Participants joining an ongoing session have missed the data that has previously been exchanged by the other session member.

What to do? **Late Join Algorithms**



Late Join Algorithms

- Necessary algorithms to distribute the current state of the session to new users.
- Two Approaches: Transport protocol. Application based

Transport Protocol: Request *ALL* previous session information (rollback)

Pros:

- Robust
- Application Independent

Cons:

- Inefficient
- The state of some applications can't be reconstructed. (Networked action games)



Late Join (cont.)

Application based: The late join algorithm varies by the type of application. (e.g.- networked games vs. whiteboard)

- Efficient – Only need the current state of the session
- Lack of reusability

Setup for Late Join:

1. New node must determine the priority of the subcomponents of the state (e.g. – You want to transfer the most recent page for a whiteboard)
2. New node (client) needs to select one or more of the existing nodes as a server.
3. Information must be transmitted to the new node.



Late Join (cont.)

- Late join policy differs based on the application
- Different proposed policies:
 1. No late join
 2. Immediate late join
 3. Event-triggered late join
 4. Network-capacity-oriented



Late Join (cont.)

Distribution of Data:

1. **One network group** (base group) – Broadcast the state to the whole group. Unnecessary packets get sent to existing nodes. (Beneficial if the ratio of late joins to the existing users is very high)
2. **Two network groups** – All late join clients join the client group.
3. **Three network groups** – In addition to the two network groups, the late join *servers* form an additional multicast group.

Problems:

Who should be selected to act as a server??

Properties of the Black Box

Consistent

Scalable

Secure

Robust

Available

Real-Time





Availability

- Like robustness, no single point of failure will affect the entire network.
- This is one of the major advantages over centralized networks, where the failure of the server causes the entire network to fail.
- If a node fails, it gets disconnected from the network, but game/session continues with remaining nodes.
- After N packets of a failed node are not received, other nodes determine that this node got disconnected, and stops using dead reckoning on its messages.

Properties of the Black Box

Consistent

Scalable

Secure

Robust

Available

Real-Time





Real-Time

Age of Empire study:

- 250 milliseconds of command latency was not noticeable
- Between 250 to 500 msec was playable
- People develop a 'game pace' or mental expectation. Users would rather have a constant 500msec command delay rather than one that alternates between fast and slow.
- In excited moments users would repeat commands which would cause huge spikes in the network demand so a simple filter was placed to prevent reissuing of commands



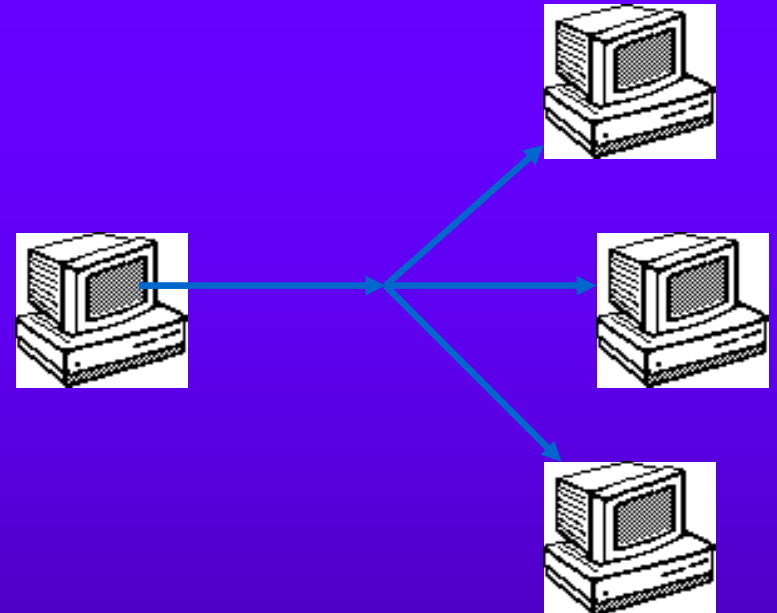
Multicast

- Multicast is a flooding algorithm
- Distributed applications use multicast to propagate a message to all nodes in the network
- Multicast was only used in a low percentage of the routers
- Manufacturers were reluctant to change to multicast enabled equipment because it is rarely used but it maximizes utilization of bandwidth.
- Applications: Video/audio transmission, whiteboard

Multicast can happen in many ways

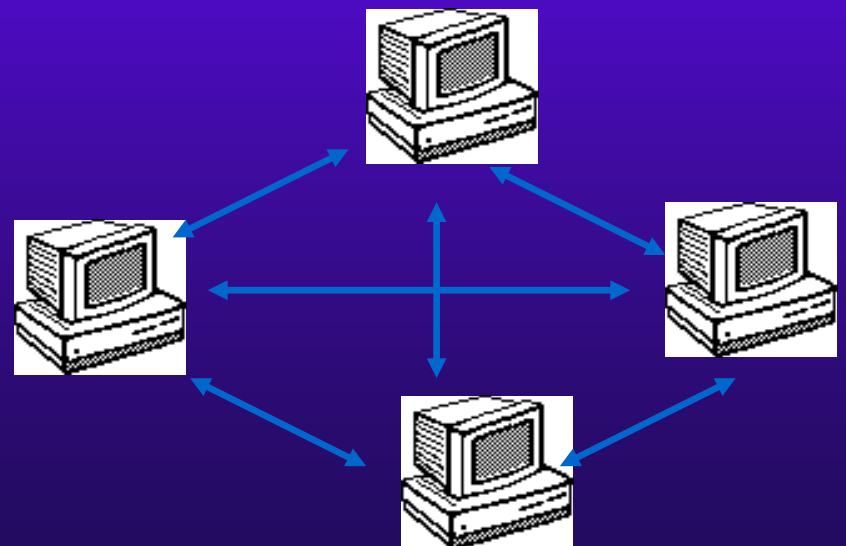
1-many

- ◆ multicast file transport
- ◆ delivery of lecture



Many-many

- ◆ teleconference
- ◆ distributed game



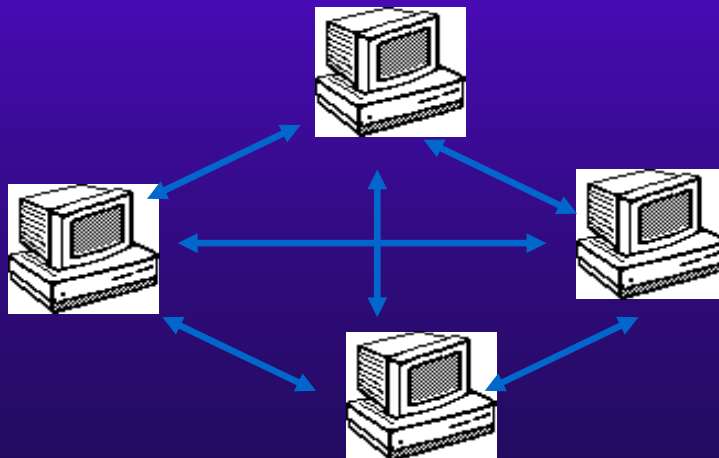
Interactive gaming

applications:

- ◆ distributed interactive simulation
- ◆ virtual reality
- ◆ distributed multi-player games

common requirements:

- ◆ low latency (200 ms end-end)
- ◆ loss tolerant
- ◆ potentially large scale
- ◆ many-many, most “players both send and receive
- ◆ group structure (e.g., locality in communication) among





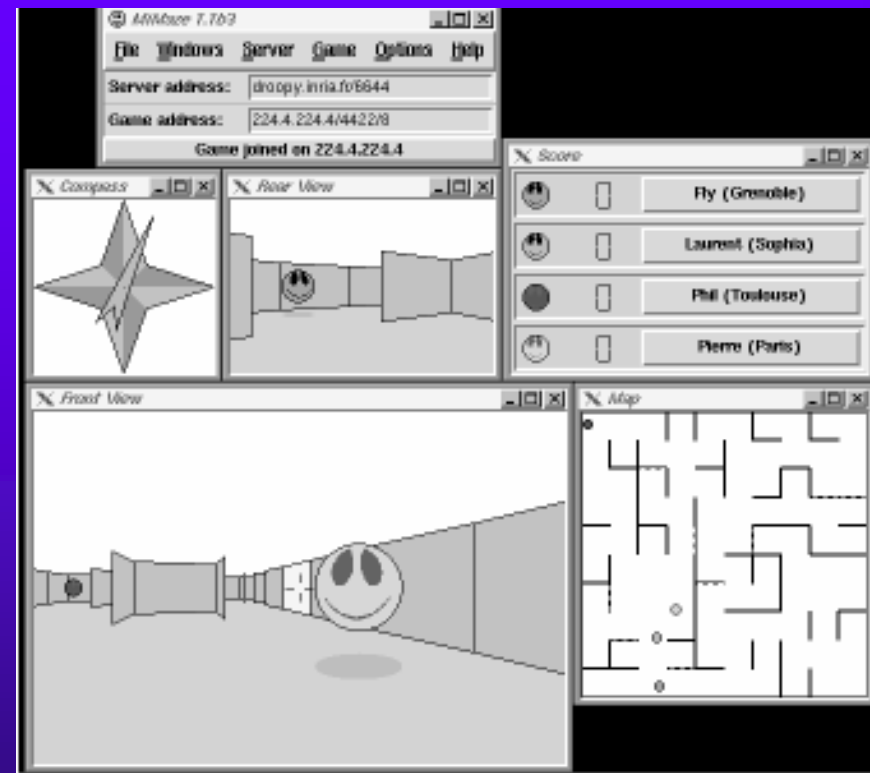
MBONE

- ◆ Immediate solution to Multicasting
- ◆ Loose confederation of sites that currently implement IP multicast.
- ◆ Allows multiple packets to travel through routers that are setup to handle only unicast traffic.
- ◆ Will be obsolete when all routers implement multicasting



MiMaze

- MiMaze is a distributed 3-D Pacman game that uses an unreliable communication system which is based on RTP over UDP/IP multicast.
- Not fully distributed, since a server is used to get the state of the game when new players join ongoing sessions and to do session management in general.





MiMaze

DIS Characteristics that apply to MiMaze:

- Interaction delay: any action issued by any participant must reach, be processed and be displayed to any other participant within the shortest possible delay.
- Participants can join and leave a MiMaze session dynamically.
- The system architecture is distributed.



MiMaze

General Characteristics of MiMaze:

- To minimize network traffic, MiMaze uses only one type of packet, which is called ADU (Application Data Unit).
- ADU's are similar to DIS's ESPDU (Entity State PDU). They are 52 bytes long, 8 bytes for RTP header, 8 bytes for UDP header, 20 bytes for the IP header and 16 bytes for MiMaze application payload.
- ADU's contain a description of the local state of an Avatar (game objects), consisting of the local position in the game and the displacement vector of the avatar and of the projectile emitted by the avatar.



MiMaze

- ADU transmission frequency is 25 times per second, enough to obtain real-time visualization.
- Consistency is assured using Bucket Synchronization :
 - Time is divided into fixed length periods and a bucket is associated with each of period.
 - All ADU's received by a player that were issued by senders during a given period are stored by the receiver in the bucket corresponding to that interval.
 - At the end of every interval, all ADUs in that bucket are used by the entity to compute its local view of the global state.
 - Buckets are computed 100 ms after the end of the sampling period during which ADUs have been issued.



MiMaze

- When an ADU is received with a transmission delay which is more than 100 ms, its destination bucket has already been computed. But the late ADU is still stored in this bucket. It will be used by the Dead Reckoning algorithm to eventually replace a missing ADU.
- Dead Reckoning Algorithm used is the simplest one: Use the previous ADU received from the missing source.
 - Not very efficient, but studies show that this does not affect the view of the game dramatically.
 - Better algorithms can be implemented, but the authors were not concerned about this issue.



MiMaze

- Global clock mechanism: Bucket synchronization uses a global clock system to evaluate the delay between participating entities.
- NTP (Network Time Protocol) was chosen, but it has 3 difficulties:
 - There are 3 levels of NTP servers and it is very difficult to maintain good synchronization among participants when level 3 servers are involved.
 - NTP encodes clock information in 64 bits, while RTP uses a 32-bit clock. MiMaze has to manipulate both representations.
 - NTP does not provide a reference clock signal and each participant has to compute an offset for every other participant in a game session.



MiMaze

-Future Work:

- Better dead reckoning algorithm should be implemented to improve calculation of missing packets.
- Some work has to be done to prevent cheating, since this system can be easily fooled.
- Purpose of this design is not to fulfill every question in the field, but to show that with a multicast communication architecture and with a simple synchronization mechanism, a “fully” distributed interactive application can provide an acceptable level of consistency to DIAs on the internet.
- Play MiMaze: <http://www.inria.fr/rodeo/MiMaze>



Age of Empires

Early Goal (1996) :

- 8 players in multi-player mode with 16Mb Pentium 90 at approximately 15fps
- Story consisted of devastating a Greek city with catapults, archers, and warriors on one side while it was being besieged from the sea on the other

Early Attempt:

- Pass small sets of data about the units
- This attempt limited actions to only 250 moving units at a time

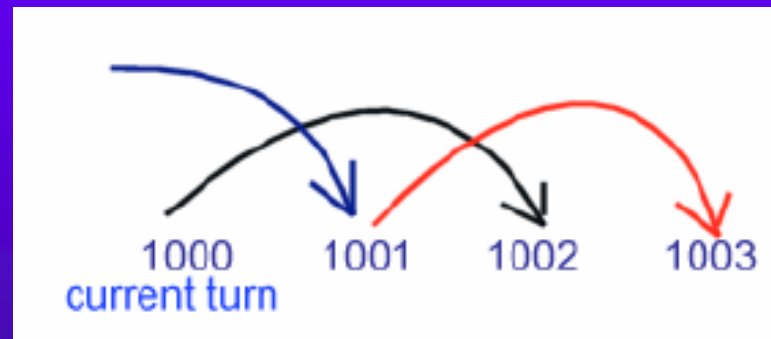


Age of Empires

- Simultaneous Simulations: Rather than passing the status of each unit in the game, run the exact same simulation on each machine, passing each an identical set of commands that were issued by the users at the same time.
- Problems still exist with network delay:
 1. Sending out the player commands
 2. Acknowledging all messages
 3. Processing them before going on to the next turn

Age of Empires

- Turn Processing: A scheme designed to continue processing the game while waiting for communications to happen in the background



- Commands issued during turn 1000 would be scheduled for execution during turn 1002. So on turn 1003, commands that were issued on turn 1001 would be executed.



Age of Empires

- Speed Control: Since the simulation must always have the exact same input, the game can only run as fast as the slowest machine can process the communications, render the turn and send out new commands which would cause “lag” to all the other computers.

Two reasons for “lag”:

1. Low machine performance
2. High internet latency

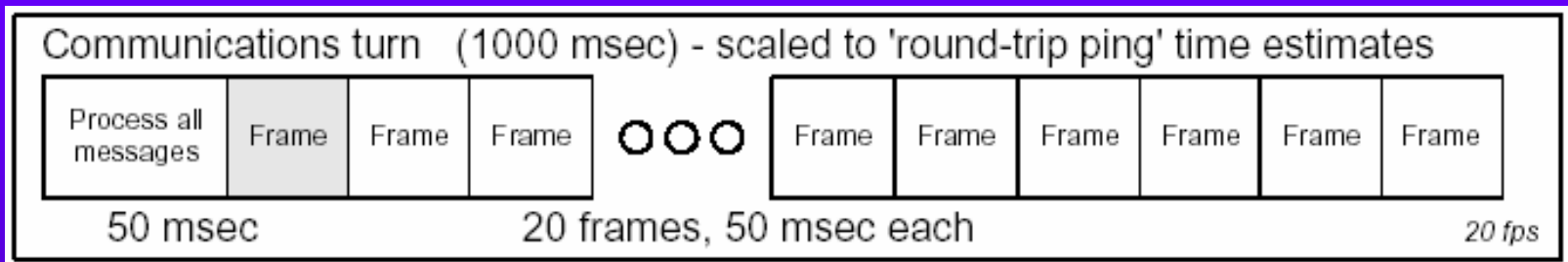


Age of Empires

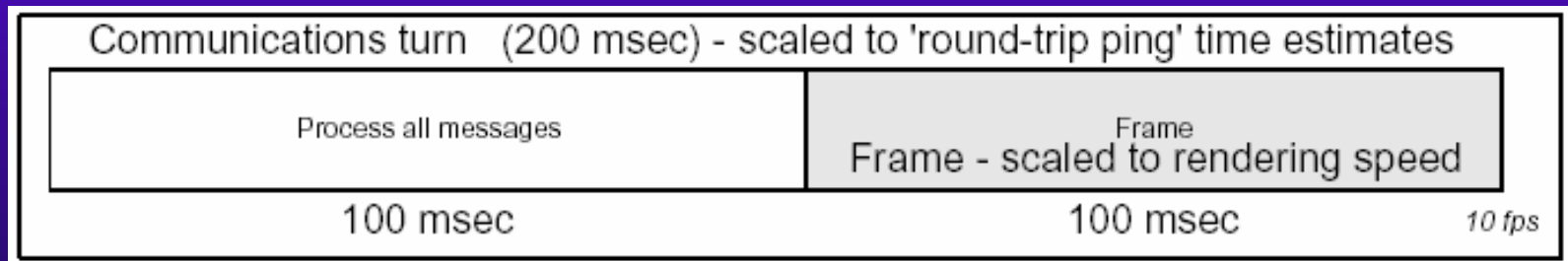
- Each client calculated:
 1. Frame rate that could be maintained consistently
 2. Round-trip ping time from itself to the other clients and an on-going average ping time
- At the end of each turn, the host would send out a new frame rate and communications turn length to be used

Age of Empires

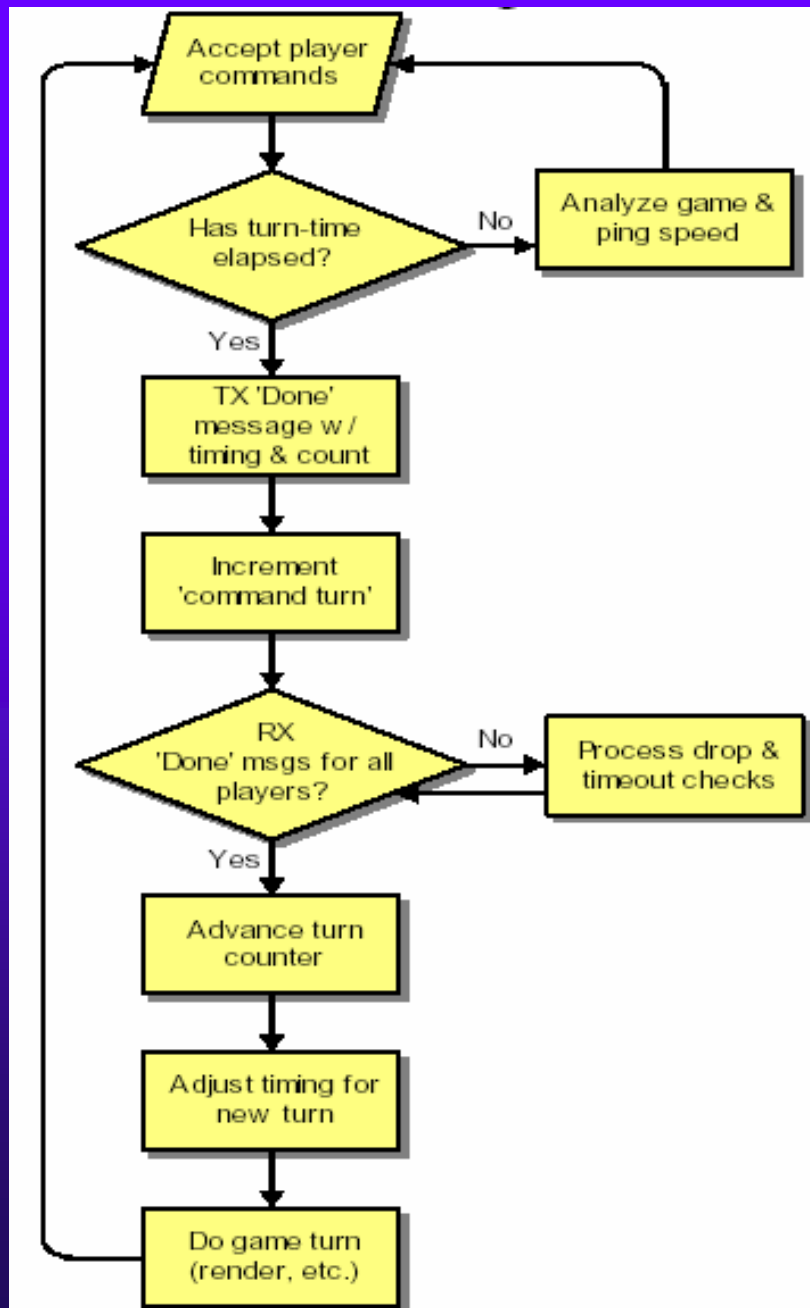
High internet latency with normal machine performance



Poor machine performance with normal internet latency



Age of Empires

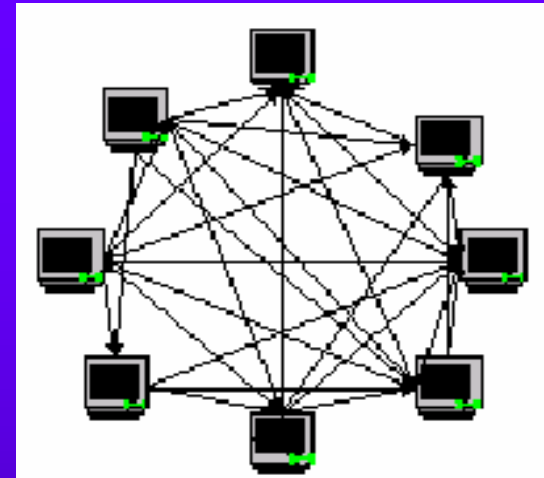


- Complete turn process
- Security benefit: All users run exact same simulation so cheating is difficult. If any simulation ran differently then it was tagged as “out of sync” and the users game stopped
- Hidden problem: It was difficult to detect when things were “out-of-sync”

Ages of Empire

Age of Empires 2 extra goals:

- Fully 3D with animation
- More than 8 players

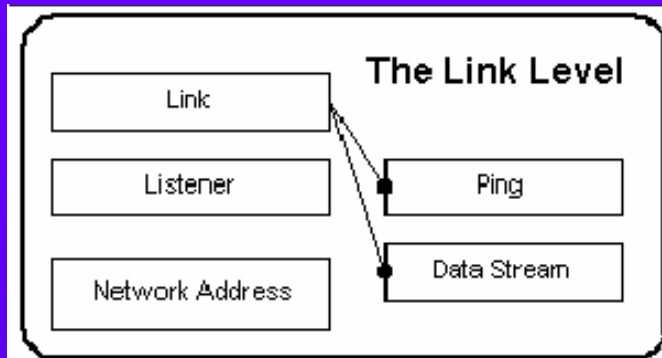


Used RTS3 Communication Architecture with their own layered architecture

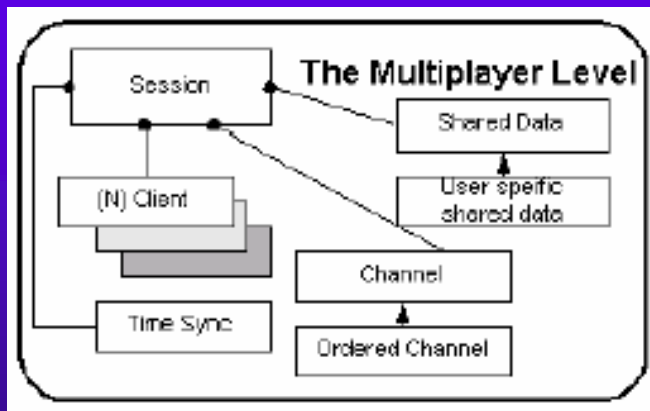
Game Communication
Multiplayer
Link
Socks

Level 1 – Socks layer provide the fundamental socket level routines

Ages of Empire



Layer 2 – Link level offers transport layer services such as the Link, Listener, and Network Address



Layer 3 – Multiplayer level holds client information, session information, time sync, and shared data

Layer 4 – Game Communication level basically boils down the game's needs into a small easy-to-use interface



Open Problems / Future Work

- Cheating algorithms without sacrificing speed
- Accurate and optimal Dead Reckoning algorithms
- Minimizing scalability overhead
- Administration and maintenance such as costs and resources

Distributed Instant Messaging

- IM systems deliver messages, as the name implies, instantly, i.e., messages are sent directly to the recipient without any intermediate storage and thus allow for real-time communication.
- Current IM Systems rely on architectures that include central, or only partly distributed servers. Although some systems can exchange messages peer-to-peer, the user registration and lookup are still based on a centralized solution.

Protocol	Reg./Lookup	Msg. exchange
AIM/OSCAR	CS	CS
ICQ	CS	p2p
IRC (DCC)	Distr. servers	CS/p2p
Jabber	CS	p2p
MSN IM	CS	CS
Yahoo	CS	CS

CS=Centralized Server, p2p=peer-to-peer

Table 1: Comparison of registration, lookup and message exchange mechanisms in different IM protocols.



Distributed Instant Messaging

- Although some of the mentioned applications, like Jabber and IRC (DCC) have some of the desired characteristics, they all suffer from a single point of failure where a malfunction will close the service, either by making it impossible to find clients or deliver messages.
- Peer-to-peer systems provide powerful platforms for a variety of decentralized services, such as network storage and content distribution.
- Today, we show 1 design:
 - DIMA based on PASTRY and SCAN



DIMA based on PASTRY

- Pastry is a generic peer-to-peer object location and routing substrate. It is a scalable, decentralized and self-organizing overlay network that automatically adapts to arrival, departure and failure of nodes.
- Pastry Node: Each node joining the pastry overlay network is assigned a random 128-bit node identifier.
- Each node maintains a routing table which is organized into $\log_{16}N$ rows with 15 columns, where N is the number of nodes. Associated with each entry is the IP address of the closest node that have the appropriate prefix.
- Additional to the routing table, each node maintains a leaf set. It contains IP-addresses of the $L/2$ numerically closest larger nodeIds and the $L/2$ closest smaller nodeIds.



DIMA based on PASTRY

- A new node joining the overlay network has to initialize its state tables and then inform the others of its presence.
- As nodes may fail or depart without warning, neighboring nodes in the key space periodically exchange keep-alive messages.
- The proposed DIMA uses Pastry for node insertion and message routing. It uses SCAN (Searching in Content Addressable Networks) to identify the nodes holding the most relevant information.
- In SCAN, a Pastry key does not correspond to one physical host, but to a piece of content on a host.



DIMA based on PASTRY

- Keywords are encoded using the ASCII table, into a set of Pastry keys for nodeIds. A salt is added at the end of the pastry key to make sure that similar/equal keywords do not generate the same Pastry key. This method of key generation, builds a pastry network structured according to meta-data keywords.
- To implement “Buddy Searching”, user info is used as the meta-data to put into the pastry network (like e-mail, nick, age, etc).
- Using SCAN, we can search for this information and retrieve a UID in form of a Pastry key (or IP Address) to use for the communication channel.

DIMA based on PASTRY

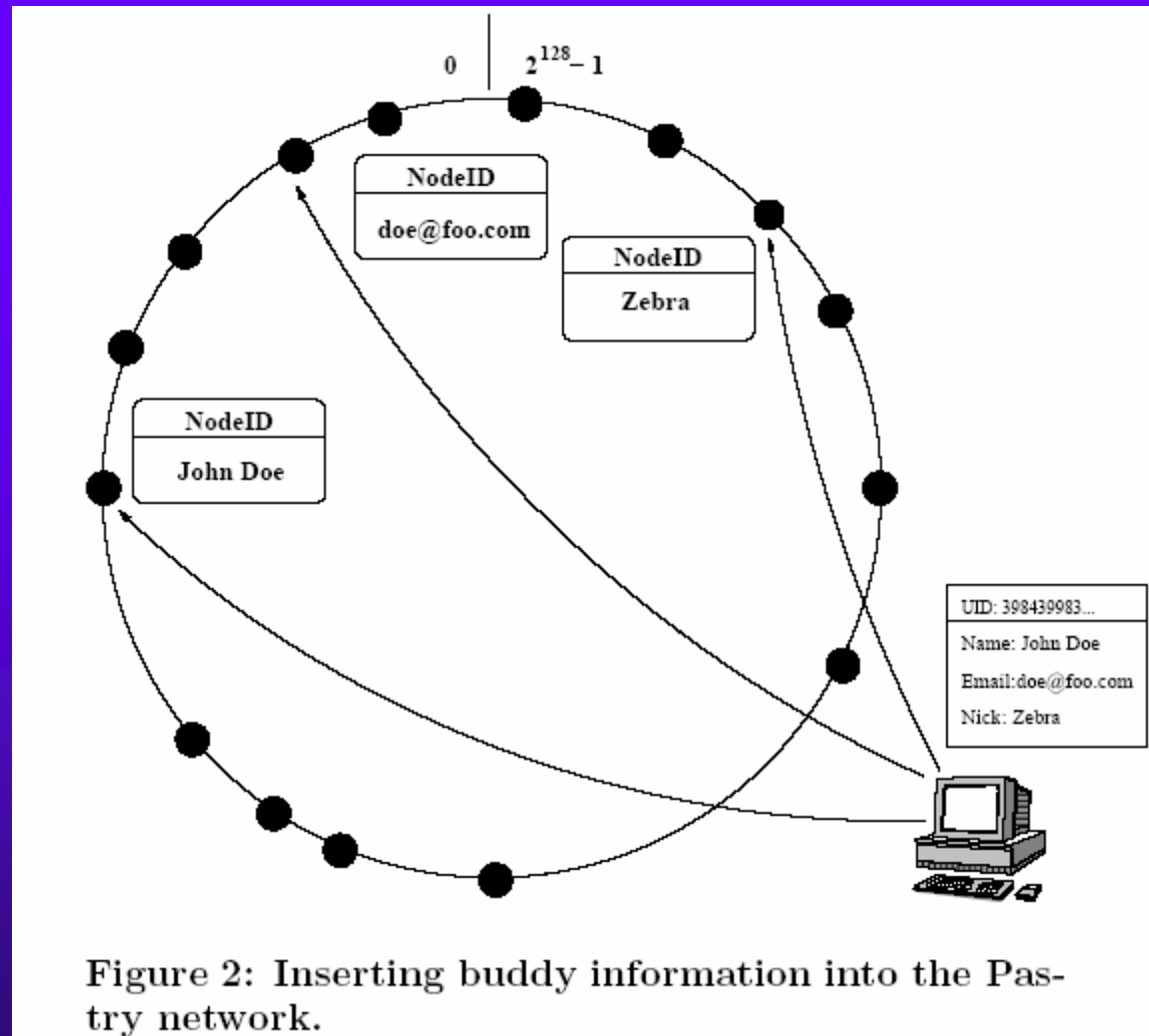


Figure 2: Inserting buddy information into the Pastry network.

- Buddy lists are stored on the user machine