Effects of delay jitter on four schemes
Some describe effects of latency instead
Permissible Client/Server Architecture
Fluctuating latency: Variable response time annoys users. Hard to compensate.
Clock Sync: Will not help
Improve fairness by artificial delay at the server. (longer delay for “closer” player)
Need to know the RTT between server and client to insert artificial lag.
Fluctuating latency: 
Hard to predict RTT.
Clock Sync: Insert timestamp to measure latency.
Server estimates latency of message and go back to the time the message is generated.
Fluctuating latency: Hard to estimate RTT
Clock Sync: Insert timestamp to measure latency.
Slow down/speed up movement of passive objects to improve consistency among players.

temporal distortion

rendered position of ball
Fluctuating latency:
Hard to estimate RTT.
Speed fluctuates.
Clock Sync: Accurate estimation of latency won’t help.
Peer-to-Peer Architecture
Problem: Communication between Every Pair of Peers
Idea (old): A peer $p$ only needs to communicate with another peer $q$ if $p$ is relevant to $q$. 
Recall: In C/S Architecture, the server has global information and decide who is relevant to who.
Problem: No global information in P2P architecture.
Naive Solution: Every peer keeps global information about all other peers and make individual decision.
Maintaining global information is expensive (and that’s what we want to avoid in the first place!)
Smarter solution: exchange position, then decide when should the next position exchange be.
Idea: Assume B is static. If A knows B’s position, A can compute the region which is irrelevant to B. Need not update B if A moves within that region.
what if B moves?
It still works if B also knows A position and computes the region that is irrelevant to A.
Position exchanges occur once initially, and when a player moves outside of its irrelevant region wrt another player.
Frontier Sets

cell-based, visibility-based IM
Previously, we learnt how to compute cell-to-cell visibility.
Frontier for cells X and Y consists of two sets $F_{XY}$ and $F_{YX}$
No cell in \( F_{XY} \) is visible from a cell in \( F_{YX} \), and vice versa.
$F_{XY}$ and $F_{YX}$ are disjoint if $X$ and $Y$ are not mutually visible.
$F_{XY}$ and $F_{YX}$ are empty if $X$ and $Y$ are mutually visible.
Suppose $X$ and $Y$ are not mutually visible, then a simple frontier is

$$F_{XY} = \{X\} \quad F_{YX} = \{Y\}$$

(many others are possible)
**NOT** a frontier for A and I (D is visible from B).
Position exchanges occur once initially, and when a player moves outside of its irrelevant region wrt another player.
Initialize:
Let player P be in cell X
For each player Q
  Let cell of Q be Y
  Compute $F_{XY}$ (or simply $F_Q$)
Move to new cell:
Let X be new cell
For each player Q
  If X not in $F_Q$
  Send location to Q
Receive Update:
(location from Q)
Send location to Q
Recompute $F_Q$
Update is triggered.
New Frontier.
Update triggered.
New frontier (empty since E can see G)
How to compute frontier?
A good frontier is as large as possible, with two almost equal-size sets.
Build a visibility graph. Cells are vertices. Two cells are connected by an edge if they are visible to each other (EVEN if they don’t share a boundary)
Let $\text{dist}(X,Y)$ be the shortest distance between two cells $X$ and $Y$ on the visibility graph.
Theorem

\[ F_{XY} = \{ i \mid \text{dist}(X,i) \leq \text{dist}(Y,i) - 1 \} \]
\[ F_{YX} = \{ j \mid \text{dist}(Y,j) < \text{dist}(X,j) - 1 \} \]

are valid frontiers.
Theorem

\[ F_{XY} = \{ i \mid \text{dist}(X,i) \leq \text{dist}(Y,i) - 1 \} \]
\[ F_{YX} = \{ j \mid \text{dist}(Y,j) < \text{dist}(X,j) - 1 \} \]

are valid frontiers.
\[ F_{XY} = \{ i \mid \text{dist}(X,i) \leq \text{dist}(Y,i) - 1 \} \]
\[ F_{YX} = \{ j \mid \text{dist}(Y,j) < \text{dist}(X,j) - 1 \} \]

**Proof** (by contradiction)
Suppose there are two cells, C in \( F_{XY} \) and D in \( F_{YX} \), that can see each other.
\[ F_{XY} = \{ \, i \mid \text{dist}(X, i) \leq \text{dist}(Y, i) - 1 \} \]
\[ F_{YX} = \{ \, j \mid \text{dist}(Y, j) < \text{dist}(X, j) - 1 \} \]

\[ \text{dist}(X, C) \leq \text{dist}(Y, C) - 1 \]
\[ \text{dist}(Y, D) < \text{dist}(X, D) - 1 \]
\[ \text{dist}(C, D) = \text{dist}(D, C) = 1 \]
\[ \text{dist}(X,C) \leq \text{dist}(Y,C) - 1 \]
\[ \text{dist}(Y,D) < \text{dist}(X,D) - 1 \]
\[ \text{dist}(C,D) = \text{dist}(D,C) = 1 \]

We also know that
\[ \text{dist}(X,D) \leq \text{dist}(X,C) + \text{dist}(C,D) \]
\[ \text{dist}(Y,C) \leq \text{dist}(Y,D) + \text{dist}(D,C) \]
1. $\text{dist}(X,C) \leq \text{dist}(Y,C) - 1$
2. $\text{dist}(Y,D) < \text{dist}(X,D) - 1$
3. $\text{dist}(C,D) = 1$
4. $\text{dist}(X,D) \leq \text{dist}(X,C) + \text{dist}(C,D)$
5. $\text{dist}(Y,C) \leq \text{dist}(Y,D) + \text{dist}(D,C)$

From 4, 1, and 3:
$\text{dist}(X,D) \leq \text{dist}(Y,C) - 1 + 1$

From 5:
$\text{dist}(X,D) \leq \text{dist}(Y,D) + 1$
1. \( \text{dist}(X, C) \leq \text{dist}(Y, C) - 1 \)
2. \( \text{dist}(Y, D) < \text{dist}(X, D) - 1 \)
3. \( \text{dist}(C, D) = 1 \)
4. \( \text{dist}(X, D) \leq \text{dist}(X, C) + \text{dist}(C, D) \)
5. \( \text{dist}(Y, C) \leq \text{dist}(Y, D) + \text{dist}(D, C) \)

We have
\[
\text{dist}(X, D) \leq \text{dist}(Y, D) + 1
\]
Which contradict 2
\[
\text{dist}(X, D) > \text{dist}(Y, D) + 1
\]
How good is the idea?

(How many messages can we save by using Frontier Sets?)
<table>
<thead>
<tr>
<th></th>
<th>q2dm3</th>
<th>q2dm4</th>
<th>q2dm8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max dist()</td>
<td>4</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>Num of cells</td>
<td>666</td>
<td>1902</td>
<td>966</td>
</tr>
</tbody>
</table>
Frontier Density: % of player-pairs with non-empty frontiers.
<table>
<thead>
<tr>
<th>Frontier Density</th>
<th>q2dm3</th>
<th>q2dm4</th>
<th>q2dm8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>83.9</td>
<td>93.0</td>
<td>84.2</td>
</tr>
</tbody>
</table>
Frontier Size:
% of cells in the frontier on average
<table>
<thead>
<tr>
<th>Frontier Size</th>
<th>q2dm3</th>
<th>q2dm4</th>
<th>q2dm8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>38.3%</td>
<td>67.3%</td>
<td>68.2%</td>
</tr>
</tbody>
</table>
Compare with
1. Naive P2P
2. Perfect P2P
Naive P2P
Always send update to 15 other players.
Perfect P2P
Hypothetical protocol that sends messages only to visible players.
Number of messages per frame per player.
Number of messages per frame per player.

<table>
<thead>
<tr>
<th></th>
<th>q2dm3</th>
<th>q2dm4</th>
<th>q2dm8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Number of messages per frame per player.

<table>
<thead>
<tr>
<th>NPP</th>
<th>q2dm3</th>
<th>q2dm4</th>
<th>q2dm8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15</td>
<td>15.7</td>
<td>14.4</td>
</tr>
</tbody>
</table>
Number of messages per frame per player.

<table>
<thead>
<tr>
<th></th>
<th>q2dm3</th>
<th>q2dm4</th>
<th>q2dm8</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPP</td>
<td>15</td>
<td>15.7</td>
<td>14.4</td>
</tr>
<tr>
<td>PPP</td>
<td>3.7</td>
<td>1.9</td>
<td>4.2</td>
</tr>
</tbody>
</table>
Number of messages per frame per player.

<table>
<thead>
<tr>
<th></th>
<th>q2dm3</th>
<th>q2dm4</th>
<th>q2dm8</th>
</tr>
</thead>
<tbody>
<tr>
<td>NPP</td>
<td>15.7</td>
<td>15.7</td>
<td>14.4</td>
</tr>
<tr>
<td>PPP</td>
<td>3.7</td>
<td>1.9</td>
<td>4.2</td>
</tr>
<tr>
<td>Frontier</td>
<td>5.4</td>
<td>2.6</td>
<td>5.9</td>
</tr>
</tbody>
</table>
Space Complexity
Let $N$ be the number of cells. If we precompute Frontier for every pair of cells, we need $O(N^3)$ space.
If we store visibility graph and compute frontier as needed, we only need $O(N^2)$ space.
Frontier Sets

cell-based, visibility-based IM
Limitations
Works badly if there’s little occlusion in the virtual world.
Still need to exchange locations with every other players occasionally.
Frontier Sets

cell-based, visibility-based IM
Voronoi Overlay Network: Aura-based Interest Management
Diagrams and plots in the sections are taken from presentation slides by Shun-yun Hu, available on http://vast.sf.net
Keep a list of neighbors within AOI and exchange messages with neighbors.
How to initialize list of neighbors?

How to keep list of neighbors up-to-date?
Every node is in charge of a region in the virtual world.

The region contains points closest to the node.
Voronoi Diagram
AOI Neighbors:
Neighbors in AOI
Enclosing Neighbors:
Neighbors in adjacent region.

(may or may not be in AOI)
Boundary Neighbors:
Neighbors whose region intersect with AOI.

(may or may not be in AOI)
Boundary and Enclosing Neighbor
Regular AOI Neighbor: Non-boundary and non-enclosing neighbor in AOI
Unknown nodes (not neighbors!)
A node always connect to its enclosing neighbours, regardless of whether they are in the AOI.
A node exchanges updates with all neighbors.
A node maintain Voronoi of all neighbors (regardless of inside AOI or not)
Suppose a player $X$ wants to join. $X$ sends its location to any node in the system.
X join request is forwarded to the node in charge of the region (i.e., closest node to X), called acceptor.
Forwarding is done greedily
(every step forward to neighbor closest to X)
Acceptor inform the joining node $X$ of its neighbors. Acceptor, $X$, and the neighbors update their Voronoi diagram to include the new node.
Suppose $X$ moves. Boundary neighbors of $X$ check if their enclosing neighbor is now in $X$’s AOI or has become $X$’s enclosing neighbor. $X$ updates its new neighbor with information about its neighbor. Neighbors outside region is disconnected. Voronoi diagrams are updated.
When a node disconnect, Voronoi diagrams are updated by the affected nodes. New boundary neighbors may be discovered.
Increase density

Constant density after 1000 nodes
Responsive
Consistent
Cheat-Free
Fair
Scalable
Efficient
Robust
Simple