Three concepts for light-weight communication in multiplayer games

Jouni Smed and Harri Hakonen

University of Turku, Finland

International Digital Games Conference (iDiG) 2006
Portalegre, Portugal
Network communication
Network communication

Player A

Player B
Network communication

Player A

Player B
Network communication

Player A  Bandwidth  Player B
Network communication

Player A  Bandwidth  Latency  Player B
Network communication

Player A  Bandwidth  Latency  Protocol  Player B
Network communication

- Player A
- Player B
- Bandwidth
- Latency
- Protocol
- Reliability
Consistency vs. responsiveness

- **consistency**
  - how similar is the replicated data among the multiple players?

- **responsiveness**
  - how promptly the data gets updated to the multiple players?

- computer games require real-time interaction
  - responsiveness is more important and consistency can be compromised
  - ≠ traditional distributed systems
How to achieve responsiveness?
Answer 1: Dead reckoning
How to achieve responsiveness?

Answer 1: Dead reckoning

\[(x_0, y_0)\]
How to achieve responsiveness?
Answer 1: Dead reckoning

\((x_0, y_0)\)
How to achieve responsiveness?

Answer 1: Dead reckoning
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?

Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?

Answer 2: Local perception filters

Blue view

Orange view
How to achieve responsiveness?
Answer 2: Local perception filters

Blue view

Orange view
Can a clever game design hide the communication latency?

- **assume:** a multiplayer game with interaction amongst the players

- **does real-time response really require real-time communication?**
  - no! (e.g. high-score lists)
  - instead of technical solutions the game design can hide latency

- **here, three concepts related to**
  - time span: short, medium, long
  - abstractness of decisions: operational, tactical, strategic
1. Operational level: Short active turns

- Serialize the game events so that each player has a turn → a turn-based game
  - Active turns: make decisions
  - Passive turns: view the game events to unfold

- Passive turns should be short and interesting
  - View statistics
  - Prepare for the next active turn
  - View replays of past events

- Candidates: attempt-based sports games
  - Javelin, long jump, ski jump, darts…
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

- **$p_1$**
  - active turn

- **$p_2$**
  - render turn

- **$p_3$**
  - filler

- **$p_4$**
  - replay and filler
Example: A sports game

$p_1$

$p_2$

$p_3$

$p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

$p_1$

$p_2$

$p_3$

$p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- Yellow: active turn
- Green: render turn
- Purple: filler
- Blue: replay and filler
Example: A sports game

$p_1$

$p_2$

$p_3$

$p_4$

- active turn
- render turn
- filler
- replay and filler
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- Active turn
- Render turn
- Filler
- Replay and filler
Example: A sports game

- $p_1$
- $p_2$
- $p_3$
- $p_4$

- yellow: active turn
- green: render turn
- purple: filler
- blue: replay and filler
2. Tactical level: Semi-autonomous avatars

- Tactical commands are not so time-sensitive
  - Operational: ‘move forward’, ‘turn left’, ‘shoot’
  - Tactical: ‘attack’, ‘guard’, ‘flee’

- The avatars are semi-autonomous
  - They receive tactical commands
  - They decide the operations themselves

- Response is not immediate
  - Copes with high latency

- Outcome can be something else than the player expected: free will!
Example: Semi-autonomous avatars

$p_1$
Example: Semi-autonomous avatars

$p_1 \rightarrow a_1: \text{attack}$
Example: Semi-autonomous avatars

$p_1 \rightarrow p_2$

$a_1$: attack

- move right
- reload gun
- aim
- shoot
Example: Semi-autonomous avatars

\[ a_2: \text{guard} \]

\[ a_1: \text{attack} \]

- move right
- reload gun
- aim
- shoot
Example: Semi-autonomous avatars

$p_1$

$p_2$

$a_2$: guard

stay put

hide

scout

$a_1$: attack

move right

reload gun

aim

shoot
Example: Semi-autonomous avatars

$p_1$

$a_2$: guard
- stay put
- hide
- scout

$p_2$

$a_1$: attack
- move right
- reload gun
- aim
- shoot

$p_3$

$a_3$: flee
Example: Semi-autonomous avatars

$p_1$

$p_2$

$p_3$

$a_2$: guard

stay put

hide

scout

$a_1$: attack

move right

reload gun

aim

shoot

$a_3$: flee

run away
3. Strategic level: Interaction via proxies

- participating players do not have to be present at the same time
  - players set proxies that can later interact with other players

- proxies
  - fully autonomous avatars
  - game entities (mechanistic objects or gizmos)
  - programmable objects
Example: *Entrappers*
Example: Entrappers
Example: *Entrappers*
Example: *Entrappers*
Example: *Entrappers*
Example: *Entrappers*
Example: Entrappers
Conclusions

- latency is caused by technical limitations
  - the speed of light!
  - cabling, routers, operating system…
- latency can be hidden
  - by technical methods
  - by clever game design
- so why not to try to use them both!
Please ask me more about the details that I omitted!