§9.5 Synchronized Simulation

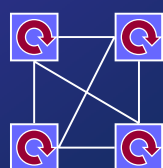
- used in *Age of Empires* (1997)
- command categories:
  - deterministic: computer
  - indeterministic: human
- distribute the indeterministic commands only
- deterministic commands are derived from pseudo-random numbers
  → distribute the seed value only
- consistency checks and recovery mechanisms

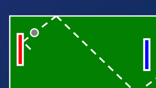
Synchronized Simulation in *Age of Empires*

- *Age of Empires* game series by Ensemble Studios
- Real-time strategy (RTS) game
- Max 8 players, each can have up to 200 moving units
  ⇒ 1600 moving units
  ⇒ large-scale simulation
- Rough breakdown of the processing tasks:
  - 30% graphic rendering
  - 30% AI and path-finding
  - 30% running the simulation and maintenance

Synchronized (or Simultaneous) Simulation

- Large simulation ⇒ a lot of data to be transmitted
- Trade-off: computation vs. communication
  - 'If you have more updating data than you can move on the network, the only real option is to generate the data on each client'
- Run the *exact* same simulation in each client

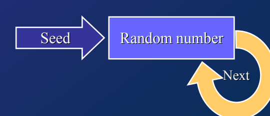




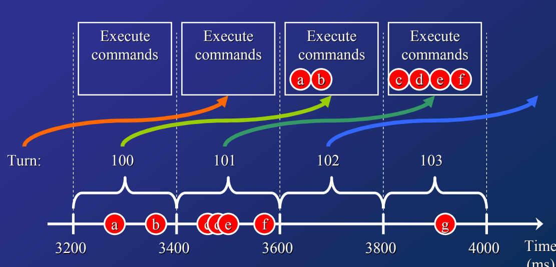
Handling Indeterminism

- 'Indeterministic' events are either
  - predictable (computers) or
  - unpredictable (humans)
- Only the unpredictable events have to be transmitted
  ⇒ communication
  - apply an identical set of commands that were issued at the same time
- The predictable events can be calculated locally on each client
  ⇒ computation
- Pseudo-random numbers are deterministic
- All clients use the same seed for their random number generator
  - disseminate the seed

Pseudo-random number generator

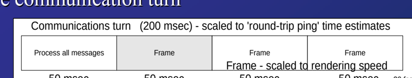


Communication Turns



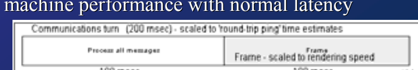
Division of the Communication Turn

Single communication turn



High Internet latency with normal machine performance

Poor machine performance with normal latency



Features

- Guaranteed delivery using UDP
  - message packet:
    - execution turn
    - sequence number
  - if messages are received out of order, send immediately a resend request
  - if acknowledgement arrives late, resend the message
- Hidden benefits
  - clients are hard to hack
  - any simulation running differently is out-of-sync
- Hidden problems
  - programming is demanding
  - out-of-sync errors
  - checksums for everything
    - 50 Gb message logs

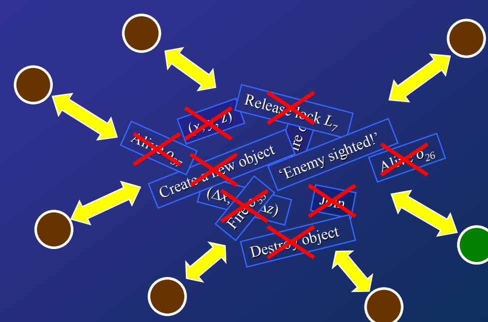
Lessons Learned

- Players can tolerate a high latency as long as it remains constant
  - for an RTS game, even 250–500 ms latencies are still playable
- Jitter (the variance of the latency) is a bigger problem
  - consistent slow response is better than alternating between fast and slow
- Studying player behaviour helps to identify problematic situations
  - hectic situations (like battles) cause spikes in the network traffic
- Measuring the communication system early on helps the development
  - identify bottlenecks and slowdowns
- Educating programmers to work on multiplayer environments

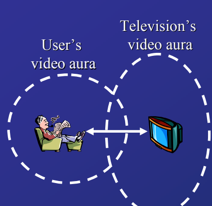
§9.6 Area-of-Interest Filtering

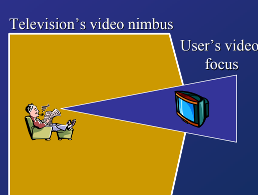
- Area-of-interest filters
  - each host provides explicit data filters
  - filters define the interest in data
- Multicasting
  - use existing routing protocols to restrict the flow of data
  - divide the entities or the region into multicast groups
- Subscription-based aggregation
  - group available data into fine-grained 'channels'
  - hosts subscribe the appropriate channels

Why to Do Data Flow Restriction?



Awareness and the Spatial Model of Interaction

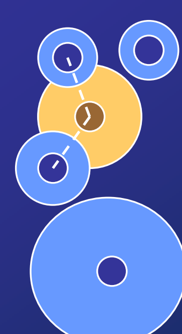




Key concepts:

- *medium*: communication type
- *aura*: subspace in which interaction can occur
- *awareness*: quantifies one object's significance to another object (in a particular medium)
- *focus*: represents an observing object's interest
- *nimbus*: represents an observed object's wish to be seen
- *adapters*: can modify an object's auras, foci, and nimbi

Nimbus-Focus Information Model



- Nimbus: entity data should only be made available to entities capable of perceiving that information
- Focus: each entity is only interested in information from a subset of entities
- Ideally, all information is processed individually and delivered only to entities observing it
  - what about scaling up?
  - processing resouces
  - each packet has a custom set of destination entities ⇒ hard to utilize multicasting

⇒ Approximate the pure nimbus-focus model

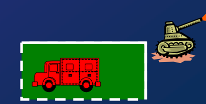
Area-of-Interest Filtering Subscriptions

- Nodes transmit information to a set of subscription managers (or area-of-interest managers, filtering servers)
- Managers receive subscription descriptions from the participating nodes
- For each piece of data, the managers determine which of the subscription requests are satisfied and disseminate the information to the corresponding subscribing nodes
- AOI filtering:
  - restricted form of the pure nimbus-focus model
    - ignores nimbus specifications
  - subscription descriptions specify the entity's focus
  - reduces the processing requirements of the pure model

Subscription Interest Language

- Allows the nodes to express formally their interests in the game world
- Subscription description can be arbitrarily complex
  - a sequence of filters or assertions
  - based on the values of packet fields
  - Boolean operators
  - programmable functions

```
(OR
  (EQ TYPE "Tank")
  (AND
    (EQ TYPE "Truck")
    (GT LOCATION-X 50)
    (LTE LOCATION-X 75)
    (GT LOCATION-Y 83)
    (LTE LOCATION-Y 94)
    (EQ PACKET-CLASS INFRARED)))
```

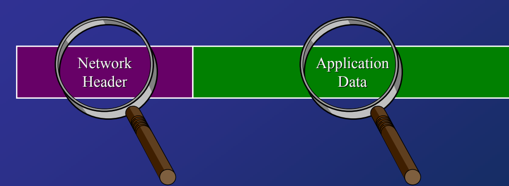


When to Use Customized Information Flows?

1. Nodes cannot afford the cost of receiving and processing unnecessary messages
2. Nodes are connected over an extremely low-bandwidth network
3. Multicast or broadcast protocols are not available
4. Client subscription patterns change rapidly
5. No a priori categorizations of data

- Problem when a large number of hosts are interested in the same piece of information
  - customized data streams ⇒ unicast ⇒ the same data travels multiple times over the same network

Intrinsic and Extrinsic Filtering



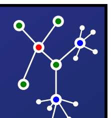
Extrinsic filtering

- Filters packets based on network properties
- Implementation efficient
- Filtering cannot be as sophisticated

Intrinsic filtering

- The filter must inspect the application content
- Can dynamically partition data based on fine-grained entity interests

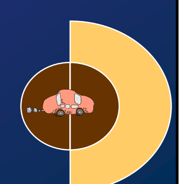
Multicasting



- Transmit a packet to a multicast group (multicast address)
- Packets are delivered to nodes who have subscribed to the multicast group
- Explicit subscription (join group) and unsubscription (leave group)
- A node can subscribe to multiple groups simultaneously
- Transmission to a group does not require subscription
- Challenge: how to partition the available data among a set of multicast groups?
- Each multicast group should deliver a set of related information
- Worst case: each node is interested in a small subset of information from every group ⇒ must subscribe to every multicast address ⇒ broadcast
- Methods:
  - group-per-entity allocation
  - group-per-region allocation

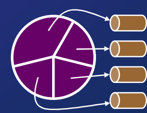
Group-per-Entity Allocation 1 (2)

- A different multicast address to each entity
- Each host receives information about all entities within its *focus*
- Subscription filter is executed locally
- Subscribe to the groups which have interesting entities
- Entities cannot specify their *nimbus*; no control over which hosts receive the information
- Example: PARADISE
  - each entity subscribes to nearby entities
  - control directional information interests
    - nearby entities that are behind
    - nearby and distant entities that are in front



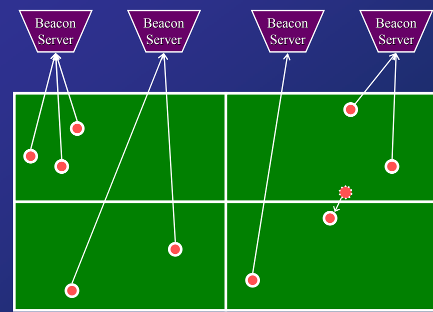
Group-per-Entity Allocation 2 (2)

- Multiple multicast group addresses to each entity
  - position updates
  - infrared data
- Information at a finer granularity
- More accurate focus by group subscriptions



- Nodes need a way to learn about nearby entities
- *Entity directory service* tracks the current state of the entities
  - entity transmits periodically state information
  - directory servers collect the information and provide it to the entities when requested

Beacon Servers



Drawbacks

- Consumes a large number of multicast addresses
- Address collisions become quite probable
- Network routers have to process the corresponding large number of join and leave requests
- Group search induces network traffic
- Network cards can only support a limited number of simultaneous subscriptions
  - too many subscriptions ⇒ ‘promiscuous’ mode