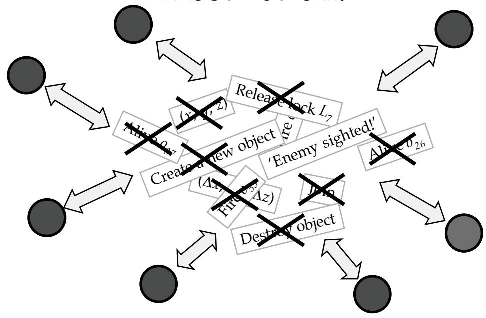
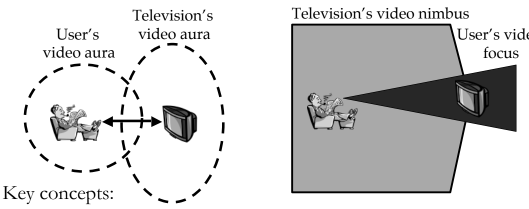
§9.6 Area-of-Interest Filtering

- 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?



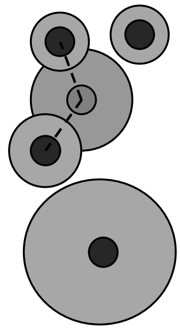
Awareness and the Spatial Model of Interaction



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

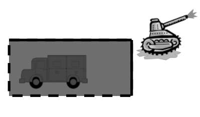
Area-of-Interest Filtering Subscriptions

- 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 expess 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)))
```

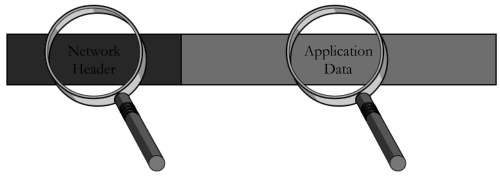


When to Use Customized Information Flows?

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

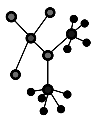
Intrinsic and Extrinsic Filtering



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

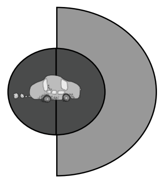


- Transmit a packet to a multicast group (multicast address)
- Packets are delivered to nodes who have subscribed to the multicast group
- Explicit subscription (join group) and unsubscription (leave group)
- A node can subscribe to multiple groups simultaneously
- Transmission to a group does not require 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)

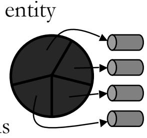
- A different multicast address to each entity
- Each host receives information about all entities within its *focus*
- Subscription filter is executed locally
- Subscribe to the groups which have interesting entities
- Entities cannot specify their *nimbus*; no control over which hosts receive the information

- 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



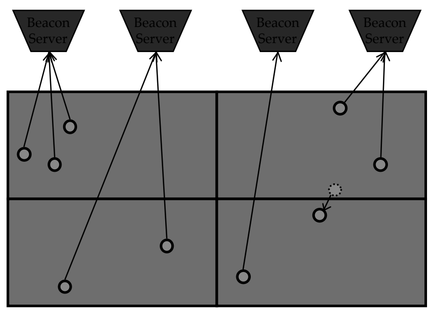
Group-per-Entity Allocation 2 (2)

- 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

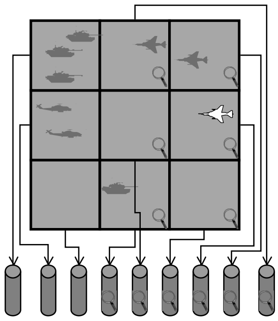


Drawbacks

- 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

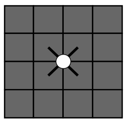
Group-per-Region Allocation

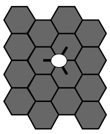
- Partition the world into regions and assign each region to a multicast group
- An entity transmits to groups corresponding to the region(s) that cover its location
- The entity subscribes to groups corresponding to interesting regions
- Entities have limited control over their nimbus but less control over their focus



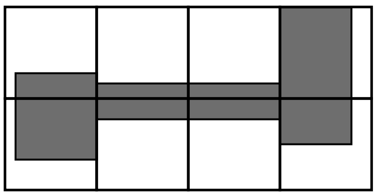
Region Bounds

- An entity has to change its target group(s) throughout its lifetime
  - track the bounds of the current region
  - learn the multicast address of a new region
  - boundaries and addresses assigned to the regions are often static
- In grid-based region assignment there are many points at which multiple grids meet
- Near these corners an entity has to subscribe to several groups



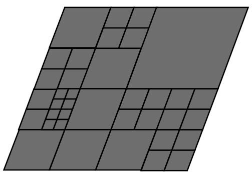


Environment vs. Regular Tessellation

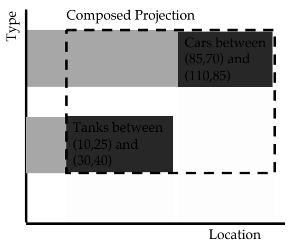


Hybrid Multicast Aggregation

- Balance between fine-grained data partitioning and multicast grouping
- Three-tiered interest management system:
  1. Group-per-region scheme segments data based on location
  2. Group-per-entity scheme allows receiver to select individual entities
  3. Area-of-interest filter subscriptions



Projections



- Projection aggregation server
  - collect data for a projection
  - transmit aggregated packets (projection aggregations)
- Projection composition
  - merge the interest specifications of the component projections

Compensating Resource Limitations: Recapitulation

- IPE: Resources = $M \times H \times B \times T \times P$
- Aspects:
  - consistency and responsiveness
  - scalability
- Protocol optimization
- Dead reckoning
- Local perception filters
- Synchronized simulation
- Area-of-interest filtering

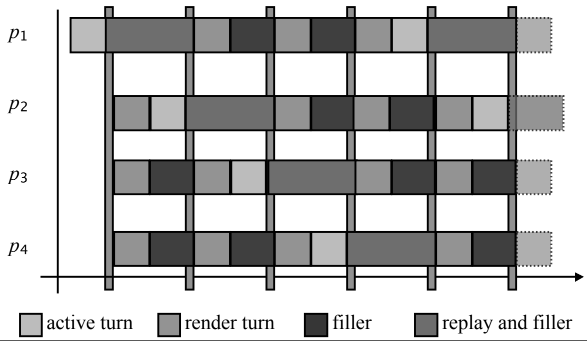
Retake: Can a Clever Game Design Hide the Communication Latency?

- 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…

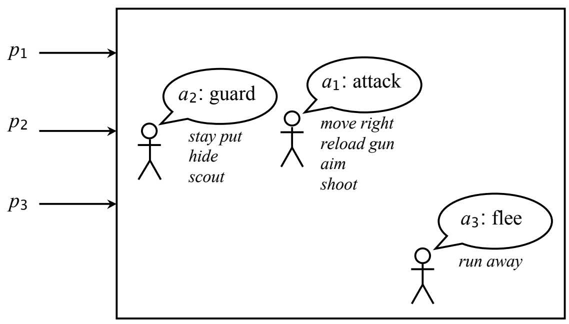
Example: A sports game



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!

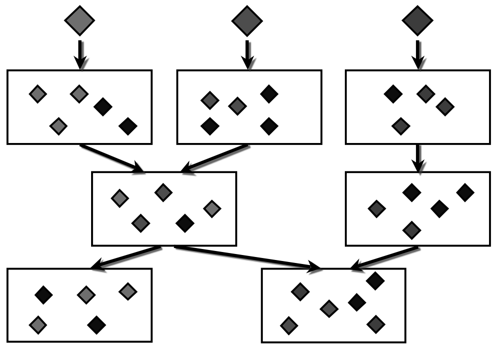
Example: Semi-autonomous avatars



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*



The Bottom Line

- 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!