Outline

- Introduction
- Data Distribution Management
  - Region-based
  - Grid-based
  - Hybrid
  - Sort-based
A federate sending a message (e.g., updating its current location) cannot be expected to explicitly indicate which other federates should receive the message.

Basic Problem: which federates receive messages?
- broadcast update to all federates (SimNet, early versions of DIS)
  - does not scale to large numbers of federates
• Goal: to send data only to those who are interested (as opposed to broadcasting in DIS)
  – it reduces network traffic
  – it reduces receiver’s message processing time
• DDM design considerations
  – efficiency: should involve minimum overheads in term of computation, message latency and memory usage
  – scalability: DDM services should scale in term of number of federates
  – DDM interface: should be as easy to use as possible while supporting the essential DDM services
Data filtering in DDM

- The publication and subscription services provided by DM are used to do filtering
- **class-based filtering**: a federate subscribes to attribute of an object-class, e.g., position (attribute) of all tanks (object-class)
  - good for small federations
- **value-based filtering**: a federate subscribes to a subset of an object’s attribute. For instance, an aircraft subscribes to position of all tanks that are within 10km of it.
  - good for large federations
Relationship between OM, DM and DDM

• OM provides services for:
  – creation, deletion and modification of objects
• DM provides services for:
  – publication and subscription
• DDM
  – ensures that the subscriber receives only the attributes of the objects it is interested in
  (DDM does filtering)
Some concepts

- **attribute**, e.g., position or speed of a moving object
- a **region** is a rectangle specified by *extents*
- **update (or publishing) region**
  - the region where attributes are changed
- **subscription region**
  - the region of an attribute that the subscriber is interested in
- a **routing space** is a normalized (from 0 to 1) multidimensional coordinate system in which federates specify regions for sending/receiving data
  - RS is used by federates to specify the distribution conditions for the specific data they are sending or expect to receive
  - it is up to the federates to define how to use a routing space
Data Distribution Management

• Allow federates to specify the distribution conditions for the specific data they send or expect to receive
  – RTI uses this information to route data as specified in declaration management services
  – Federation design creates ‘routing spaces’ for use during runtime; these are specified at federation creation time

• Interface functions include
  – Creating and modifying ‘update’ and ‘subscription’ regions within routing space
  – Associating update regions with specific object instances
HLA Routing Spaces

- N dimensional normalized routing space
- Interest expressions are regions in routing space (S1=[0.1,0.5], [0.2,0.5])
- Each update associated with an update region in routing space
- A federate receives a message if its subscription region overlaps with the update region

Federate 1 (sensor): subscribe to S1
Federate 2 (sensor): subscribe to S2
Federate 3 (target): update region U

Update messages by target are sent to federate 1, but not to federate 2

HLA Data Distribution Management

- N dimensional normalized routing space
- Interest expressions are regions in routing space (S1=[0.1,0.5], [0.2,0.5])
- Each update associated with an update region in routing space
- A federate receives a message if its subscription region overlaps with the update region
Value-based filtering: an alternative formulation

- **Problem formulation**
  - a group consisting of $n$ producers (publishers) and $m$ consumers (subscribers)
  - producers change types and attributes of their products dynamically and frequently
  - consumers change their needs dynamically and frequently

- **Question**
  - What is the most efficient matching algorithm?
Main DDM approaches

- The region-based approach
- The grid-based approach
If $P_i$ and $S_k$ have a non-empty intersection, then RTI matches these two and
- establishes communications connectivity between $P_i$ and $S_k$
- all data published by $P_i$ is sent to $S_k$

Matching high if many $P$s and $S$s

Figure: region-based filtering
Region-Based Implementation

- Associate a multicast group with each update region
- Membership: all subscription regions overlapping with update region
- Changing subscription (update) region
  - Must match new subscription (update) region against all existing update (subscription) regions to determine new membership

**Diagram:**
- Multicast group associated with U
- Membership of U: S1
- Modify U to U’:
  - determine subscription regions overlapping with U’
  - Modify membership accordingly
### Grid-based Approach

- the routing space is partitioned into a **grid** of cells
- multicast list of subscribers for each cell
- when an object (federate) publishes in a cell, the data is sent to all subscribers of that cell
- User works with regions whereas DDM works with cells

![Figure: grid-based filtering](image_url)
Implementation: Grid Based

- subscription region: Join each group overlapping subscription region
- attribute update: send Update to each group overlapping update region
- need additional filtering to avoid unwanted messages, duplicates

Partition routing space into grid cells, map each cell to a multicast group

U publishes to 12, 13
S1 subscribes to 6,7,8,11,12,13
S2 subscribes to 11,12,16,17
Multicast group for:
- C11 = {nil}, C12 = {S1} ...
- C22 = {S1}, C23 = {S1, S2}
- C33 = {S2} ...

Grid-based (cont’d)
Multicast group for:
- $C_{11} = \{\text{nil}\}$, $C_{12} = \{S_1\}$ …
- $C_{22} = \{S_1\}$, $C_{23} = \{S_1, S_2\}$
- $C_{33} = \{S_2\}$ …
- $P_1$ overlaps with $C_{22}$, its data will be sent to $S_1$
Multicast group for:
- $C_{11} = \{\text{nil}\}$, $C_{12} = \{S_1\}$ ...
- $C_{22} = \{S_1\}$, $C_{23} = \{S_1, S_2\}$
- $C_{33} = \{S_2\}$ ...

- $P_1$ overlaps with $C_{22}$, its data will be sent to $S_1$

- note: $P_2$ overlaps with $C_{33}$, data published by $P_2$ in $C_{33}$ will be sent to $S_2$, though $S_2$ may not need the data!
Grid based filtering protocols

- **Assumptions**
  - we use a time-stepped simulation
  - attributes are updated and sent to subscribers at each time-step
- **Protocol 1**
  - regions are updated at each time-step
  - matching between Ps and Ss is done at every time-step
- **Protocol 2**
  - regions are updated after every $k$ time steps
  - matching between Ps and Ss is done after every $k$ time steps
- **Protocol 3**
  - S-regions are updated after every $k$ time steps
  - matching between Ps and Ss is done at every time-step
Research issues in grid based filtering

- what is the impact of cell size on performance?
- what is the impact of region updating frequency \((k)\) on performance?
- what is the optimal cell-size?
## Changing a Subscription Region

### Change subscription region:
- issue *Leave* operations for (cells in old region - cells in new region)
- issue *Join* operations for (cells in new region - cells in old region)

Observation: each processor can autonomously join/leave groups whenever its subscription region changes w/o communication.

### Diagram

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>
```

- **Blue** indicates Leave group
- **Pink** indicates Join group
- **Green** indicates no operations issued

---
Analytical Model (Results)

- **P/S** - cost of updating cell lists (P and S lists)
- **I_m** - cost of communicating, detecting and eliminating irrelevant messages
- **C** - total overhead

Figure: Impact of cell size on overhead
Impact of Grid-Cell Size (results)

- Battlefield scenario
- AWACS aircraft scenario
Hybrid Approach

• Step 1:
  – use the grid-based approach to select the multicast group members.

• Step 2:
  – use the region-based matching to filter out the irrelevant members in the group.
Hybrid Approach

Step 1:
Multicast group for U1 is \{S1, S3\}

Step 2:
Since U1 and S3 have no overlap, DDM manager removes S3 from U1’s multicast group. Group is \{S1\}
Hybrid Approach

comparison between grid-based approach and hybrid approach

80 tanks, update distance = 4km, subscription distance = 64km
Comparison of matching number between the hybrid approach and the region-based approach.
Sort-based algorithm

- Project endpoints of extents on each dimension
- Sort endpoints for each dimension
- For each point of a dimension:
  - Lower bound: add the extent to the corresponding list of overlapping extents
  - Upper bound: remove the extent from the corresponding list of overlapping extents; the extent overlaps all the extents in the list
- Two extents intersect if they overlap on each dimension
Sort-based algorithm
Algorithm for Determining Overlap Information

1. \( S^* = \emptyset \)
2. For each point \( P_i \) in the sorted list \( L \)
3. \{ 
4. \( R_i = \text{Extent Id of } P_i \)
5. If ( \( R_i \) is not in \( S \))
6. \{ 
7. \( \text{insert } R_i \text{ into } S \)
8. \}
9. else
10. \{ 
11. \( \text{remove } R_i \text{ from } S \)
12. all extents currently in \( S \) overlap with \( R_i \)
13. \}
14. \}

* \( S \) is a set that stores the extents currently in the section
** Points in the sorted list \( L \) are in ascending order.
Example Scenario

Routing Space

A  B  C  D

x-dimension

x₁  x₂  x₁  x₂  x₁  x₂
### Example Scenario

#### Diagram

![Diagram of points and overlap information](image)

#### Table

<table>
<thead>
<tr>
<th>Point</th>
<th>Point Overlap Information</th>
<th>Extent Overlap Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_{A1}$</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>$X_{C1}$</td>
<td>$X_{C1}$ overlaps with $X_A$</td>
<td>$X_A$ and $X_C$ overlap</td>
</tr>
<tr>
<td>$X_{B1}$</td>
<td>$X_{B1}$ overlaps with $X_A$ and $X_C$</td>
<td>$X_A$, $X_B$ and $X_C$ overlap</td>
</tr>
<tr>
<td>$X_{A2}$</td>
<td>$X_{A2}$ overlaps with $X_B$ and $X_C$</td>
<td>$X_A$, $X_B$ and $X_C$ overlap</td>
</tr>
<tr>
<td>$X_{C2}$</td>
<td>$X_{C2}$ overlaps with $X_B$</td>
<td>$X_B$ and $X_C$ overlap</td>
</tr>
<tr>
<td>$X_{B2}$</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>$X_{D1}$</td>
<td>/</td>
<td>/</td>
</tr>
<tr>
<td>$X_{D2}$</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>
Optimized sort-based algorithm

- Maintain two lists: “before” and “after” indicating points before and after the current position
- Initially “before” is empty and “after” contains all
- Subscription extent lower bound: remove the extent from the “after” list
- Subscription extent upper bound: add the extent to the “before” list
- Update extent lower bound: the extent doesn’t overlap the extents in the “before” list
- Update extent upper bound: the extent doesn’t overlap the extents in the “after” list
Implementation with bit vectors

\[
\begin{array}{cccccccc}
X_{u1} & s_2 & X_{u1} & X_{s2} & x_{u2} & x_{s1} & x_{s3} & X_{u2} & X_{s1} & X_{s3}
\end{array}
\]

\[
\begin{array}{c|c|c|c}
 & s_1 & s_2 & s_3 \\
\hline
u_1 & 1 & 1 & 1 \\
\hline
u_2 & 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
s_1 & s_2 & s_3 & \text{after}
\hline
0 & 0 & 0 & \text{before}
\hline
1 & 1 & 1 & \text{after}
\end{array}
\]
Sort-based algorithm
Implementation with bit vectors

\[ X_{u1} \quad x_{s2} \quad X_{u1} \quad X_{s2} \quad x_{u2} \quad x_{s1} \quad x_{s3} \quad X_{u2} \quad X_{s1} \quad X_{s3} \]

<table>
<thead>
<tr>
<th></th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( u_1 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( u_2 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \text{XOR} \]

\[
\begin{array}{ccc}
\text{before} & 0 & 0 & 0 \\
\text{after} & 1 & 1 & 1 \\
\end{array}
\]
Sort-based algorithm
Implementation with bit vectors

\[ \begin{align*}
X_{u1} & \quad X_{s2} & \quad X_{u1} & \quad X_{s2} & \quad x_{u2} & \quad x_{s1} & \quad x_{s3} & \quad X_{u2} & \quad X_{s1} & \quad X_{s3} \\
\end{align*} \]
Sort-based algorithm
Implementation with bit vectors

\[ x_{u1} \quad x_{s2} \quad X_{u1} \quad X_{s2} \quad x_{u2} \quad x_{s1} \quad x_{s3} \quad X_{u2} \quad X_{s1} \quad X_{s3} \]

\[ \begin{array}{ccc}
  s_1 & s_2 & s_3 \\
  u_1 & 0 & 1 & 0 \\
  u_2 & 1 & 1 & 1 \\
\end{array} \]

\[ \begin{array}{ccc}
  s_1 & s_2 & s_3 \\
  0 & 0 & 0 \quad \text{before} \\
  1 & 0 & 1 \quad \text{after} \\
\end{array} \]
Sort-based algorithm
**Implementation with bit vectors**

\[
\begin{array}{cccccccc}
X_{u1} & x_{s2} & X_{u1} & X_{s2} & x_{u2} & x_{s1} & x_{s3} & X_{u2} & X_{s1} & X_{s3} \\
\end{array}
\]

<table>
<thead>
<tr>
<th></th>
<th>s₁</th>
<th>s₂</th>
<th>s₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>u₂</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{ccc}
\begin{array}{c}
\text{before} \\
\text{after}
\end{array} & s₁ & s₂ & s₃ \\
\hline
0 & 1 & 0 & \\
1 & 0 & 1 & \\
\end{array}
\]
Implementation with bit vectors

\[ \begin{array}{cccccccccc}
  x_{u1} & x_{s2} & X_{u1} & X_{s2} & x_{u2} & x_{s1} & x_{s3} & X_{u2} & X_{s1} & X_{s3} \\
\end{array} \]

\begin{array}{|c|c|c|}
\hline
  & s_1 & s_2 & s_3 \\
\hline
 u_1 & 0 & 1 & 0 \\
 u_2 & 1 & 0 & 1 \\
\hline
\end{array}

\text{XOR}

\begin{array}{|c|c|c|}
\hline
 s_1 & s_2 & s_3 \\
\hline
 0 & 1 & 0 & \text{before} \\
 1 & 0 & 1 & \text{after} \\
\hline
\end{array}
Sort-based algorithm
Implementation with bit vectors

\[
\begin{array}{cccccccc}
X_{u1} & x_{s2} & X_{u1} & X_{s2} & x_{u2} & x_{s1} & x_{s3} & X_{u2} & X_{s1} & X_{s3} \\
\end{array}
\]

<table>
<thead>
<tr>
<th></th>
<th>(s_1)</th>
<th>(s_2)</th>
<th>(s_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(u_2)</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(s_1)</th>
<th>(s_2)</th>
<th>(s_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>after</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Implementation with bit vectors

\[ \begin{align*}
X_{u1} & \quad X_{s2} & \quad X_{u1} & \quad X_{s2} & \quad x_{u2} & \quad x_{s1} & \quad x_{s3} & \quad X_{u2} & \quad X_{s1} & \quad X_{s3} \\
\end{align*} \]

\[
\begin{array}{ccc}
\hline
 & s_1 & s_2 & s_3 \\
\hline
u_1 & 0 & 1 & 0 \\
u_2 & 1 & 0 & 1 \\
\hline
\end{array}
\]

\[
\begin{array}{ccc}
 & s_1 & s_2 & s_3 \\
\hline
 & 0 & 1 & 0 & \text{before} \\
0 & 0 & 0 & 0 & \text{after} \\
\hline
\end{array}
\]
**Implementation with bit vectors**

\[
\begin{align*}
X_{u1} & \quad x_{s2} & \quad X_{u1} & \quad X_{s2} & \quad x_{u2} & \quad x_{s1} & \quad x_{s3} & \quad X_{u2} & \quad X_{s1} & \quad X_{s3} \\
\end{align*}
\]

<table>
<thead>
<tr>
<th></th>
<th>(s_1)</th>
<th>(s_2)</th>
<th>(s_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(u_2)</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**XOR**

<table>
<thead>
<tr>
<th></th>
<th>(s_1)</th>
<th>(s_2)</th>
<th>(s_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>before</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>after</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Complexity

- Each lower bound and each upper bound of a subscription extent cost a single bit operation.
- Each lower bound and each upper bound of an update extent cost a bit-vector operation with Card(S) bits.
- Still quadratic but loops are replaced by operations on bit vectors.

Performance analysis

- Overall number of extents.
- Number of extents in each subscription region.

Overlap rate = \( \frac{\sum \text{area of the extents}}{\text{area of the routing space}} \)
Experimental results

- Low overlap rate: Overlap rate = 0.01

![Graph showing time taken (seconds) vs. number of regions]

Extent No = 1
Experimental results

- Medium overlap rate: Overlap rate = 1

Extent No = 1
Experimental results

- High overlap rate: Overlap rate = 100

Extent No = 1
• Region based is usually not scalable
• Main criteria: the overlap rate
  – Low (0.01): grid partition with small cells
  – Average (1): sort based algorithm
  – High (100): region based or sort based
• Choosing the right number of cells in the grid partitioning algorithms is not easy
• The sort based algorithm allows very efficient optimizations