

Sensor Webs as Multiagent, Negotiating Systems

Costas Tsatsoulis

Department of Electrical Engineering and Computer Science
Information and Telecommunication Technology Center
The University of Kansas

What is a Sensor Web

- A coordinated observation infrastructure composed of a distributed collection of resources - e.g., sensors, platforms, models, communications infrastructure - that can collectively behave as:
 - a single,
 - autonomous,
 - task-able,
 - dynamically adaptive and
 - reconfigurable observing system
- A Sensor Web allows on-demand sensing of phenomena, from a heterogeneous suite of sensors both in-situ and in orbit. Dynamically organized to collect data, extract information from them, accept input from other sensor / forecast / tasking systems.

Sensor Web Issues

- How does a sensor decide it needs to collaborate with others and must create a Sensor Web?
- How does a sensor identify others that can help in a Sensor Web?
- How does a sensor decide whether to participate in a Sensor Web?
- In this presentation we demonstrate a solution to the second problem

Our Solution

- Add intelligence to each sensor by adding an *agent*
- An agent is an intelligent software entity that can operate autonomously and perform complex decision making
- The sensor agent can:
 - Identify sensor agents that can assist it in the completion of a task through the formation of a dynamic Sensor Web
 - Communicate its needs with these agents
 - Decide based on this communication which of the sensors are best suited to join the Sensor Web
 - Then form a *coalition*

Coalition

- A *coalition* is a (possibly) heterogeneous collection of agents that are *autonomously* collaborating to perform a complex task that no one agent can complete on its own
- In a coalition there is no central authority to direct the agents in their tasks
- A *coalition forming* agent is the one that decides that it cannot perform a task on its own, and asks others to join it in the execution of the task
- Agents in a coalition are assigned specific responsibilities and they perform them in collaboration with the other, without central supervision

Coalition Formation

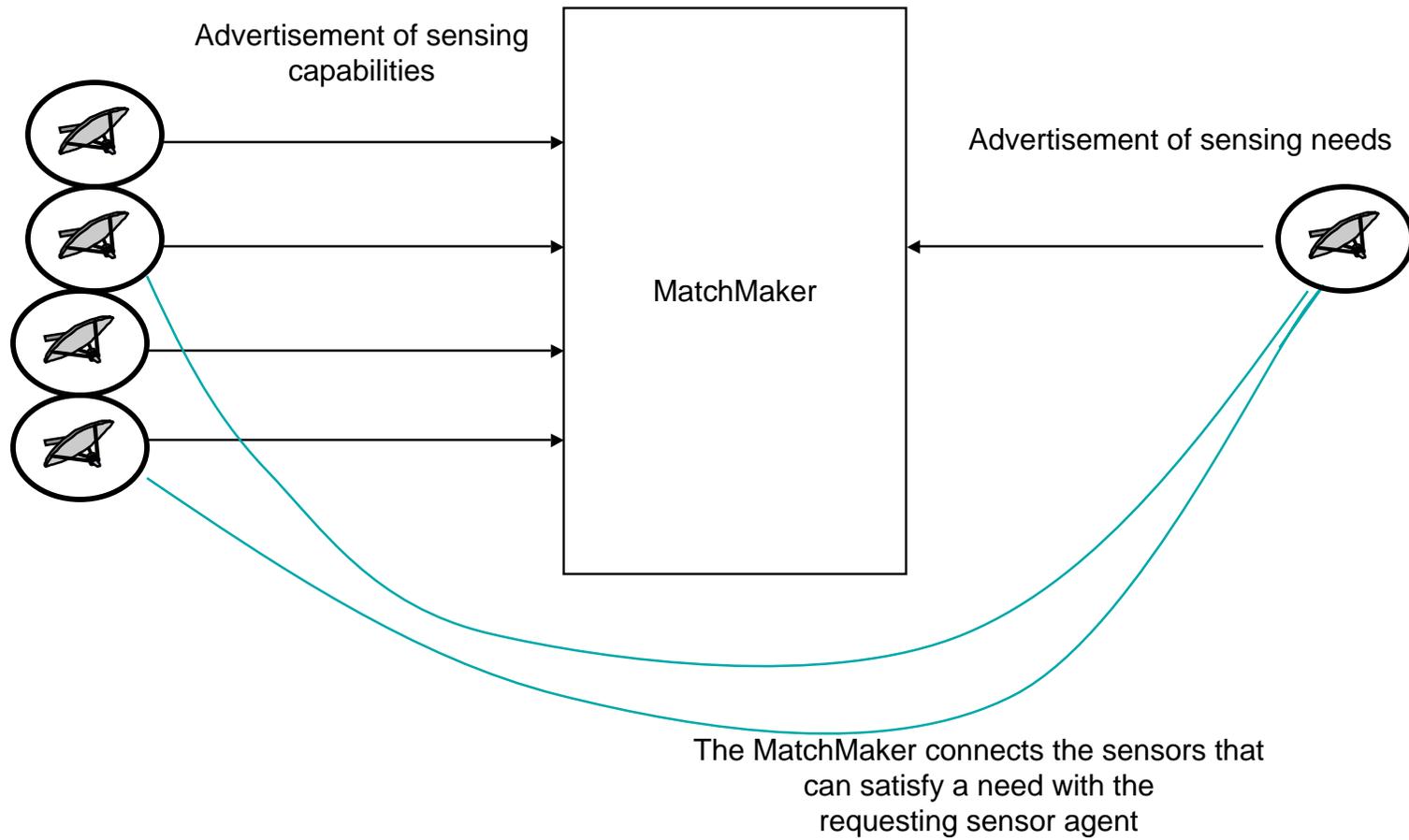
- All agents are sensing the environment
- *Event monitors* located with each agent search for specific events that would trigger the need for forming a coalition
- Event monitors look for specific values of observables (e.g. $\text{Temp} > X^0$), trends (e.g. Temp increased by $X\%$ over Y time units), averages (e.g. Temp average X^0 over Y time units), etc.
- When an event monitor is activated, the agent identifies what the event designates, and what tasks need to be performed
- If these tasks cannot be satisfied by the agent alone, it will form a coalition

Architecture of the Reasoning Component - What not to do

- A sensor agent that identifies a task (sensing or processing) need could broadcast this need to every other agent
 - Such an approach consumes too many communication resources and also forces agents to deal with messages that do not concern them, since they may not be able to satisfy the request
- A sensor agent could keep a list of all other sensors with their capabilities and contact only the ones that it knows could perform the task
 - This requires every new sensor entering the system to broadcast its capabilities to everyone. It also requires periodic “pinging” of all sensors to ensure they are still alive.

Architecture of the Reasoning Component - What we do (partially)

- We adopt a Matchmaker architecture, a centralized storage facility of all known sensors with their *basic* capabilities
- Agents entering the environment *register* their capabilities to the Matchmaker
- Advantages:
 - Single location for registration
 - Limited communication
- Disadvantages:
 - Single point of failure (could make copies!)
 - Matchmaker entries might be stale
 - *Registration of capabilities is necessarily simple and abstract*



Why the Matchmaker is not enough

- Sensors are very complex systems that often exhibit mobility
- Registering all capabilities with a Matchmaker is unrealistic
 - It requires large communication and memory resources for registration and directory services
 - It cannot handle the changing sensing area of a mobile sensor
 - It requires a very complex model for each sensor (e.g. SensorML)
- We need to store a simple model in the Matchmaker, but a more complex model of capabilities must reside somewhere and be provided to requesting sensor agents
- Solution:
 - Model will reside with an individual sensor
 - Model will be provided *partially* as part of a *bid*

Architecture of the Reasoning Component

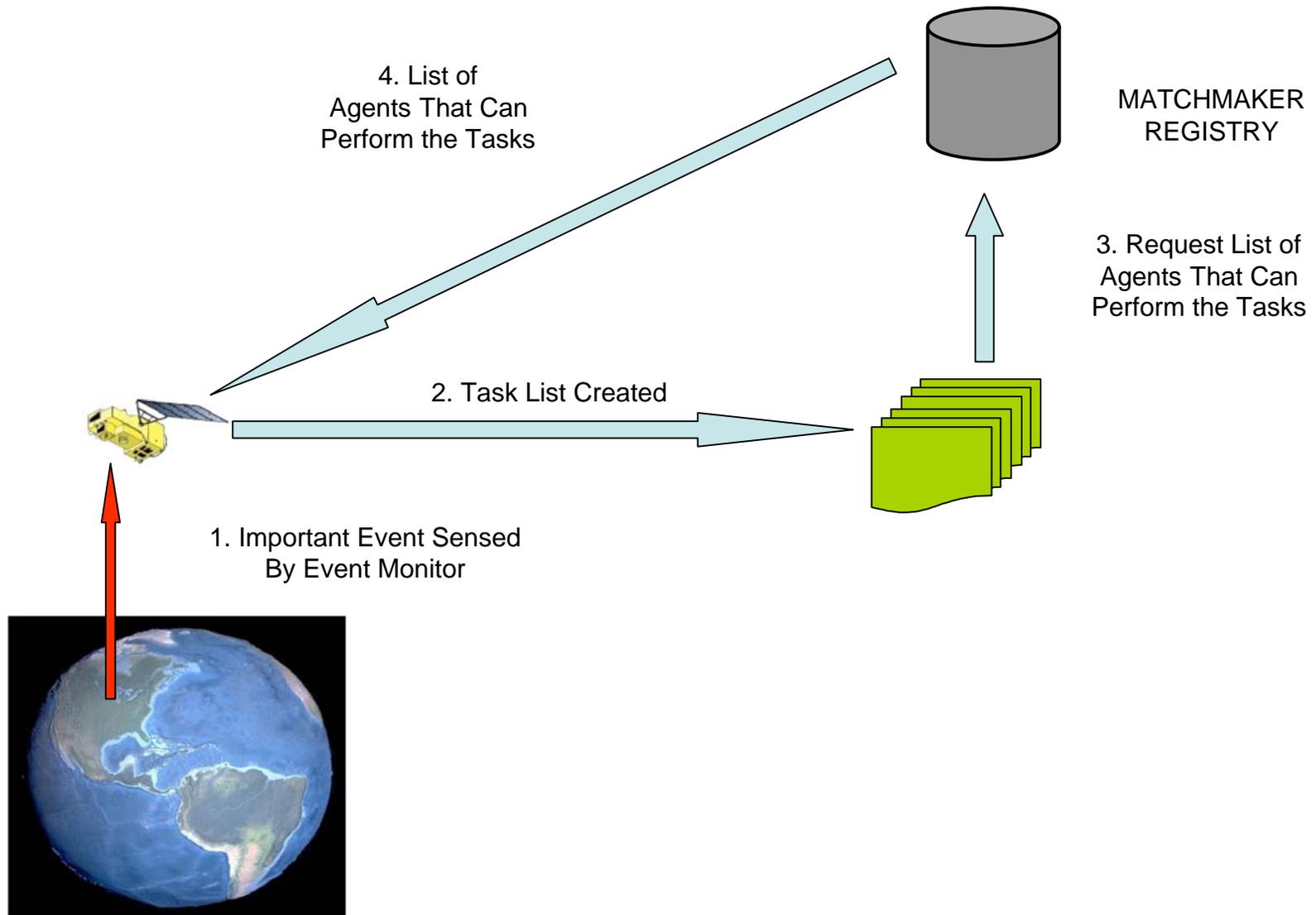
- After an agent receives from the Matchmaker the list of sensors that *could* satisfy a sensing/processing need, it composes and sends to them a *request for proposals (RFP)*
- An RFP consists of:
 - *Task abstraction*: A simple description of the task. If the agent cannot satisfy it, it will not submit a bid
 - *Bid specification*: The parameters of the bid that the requesting agent will use to evaluate the proposal
 - *Expiration time*: The time by which a bid must be received (currently expressed implicitly)

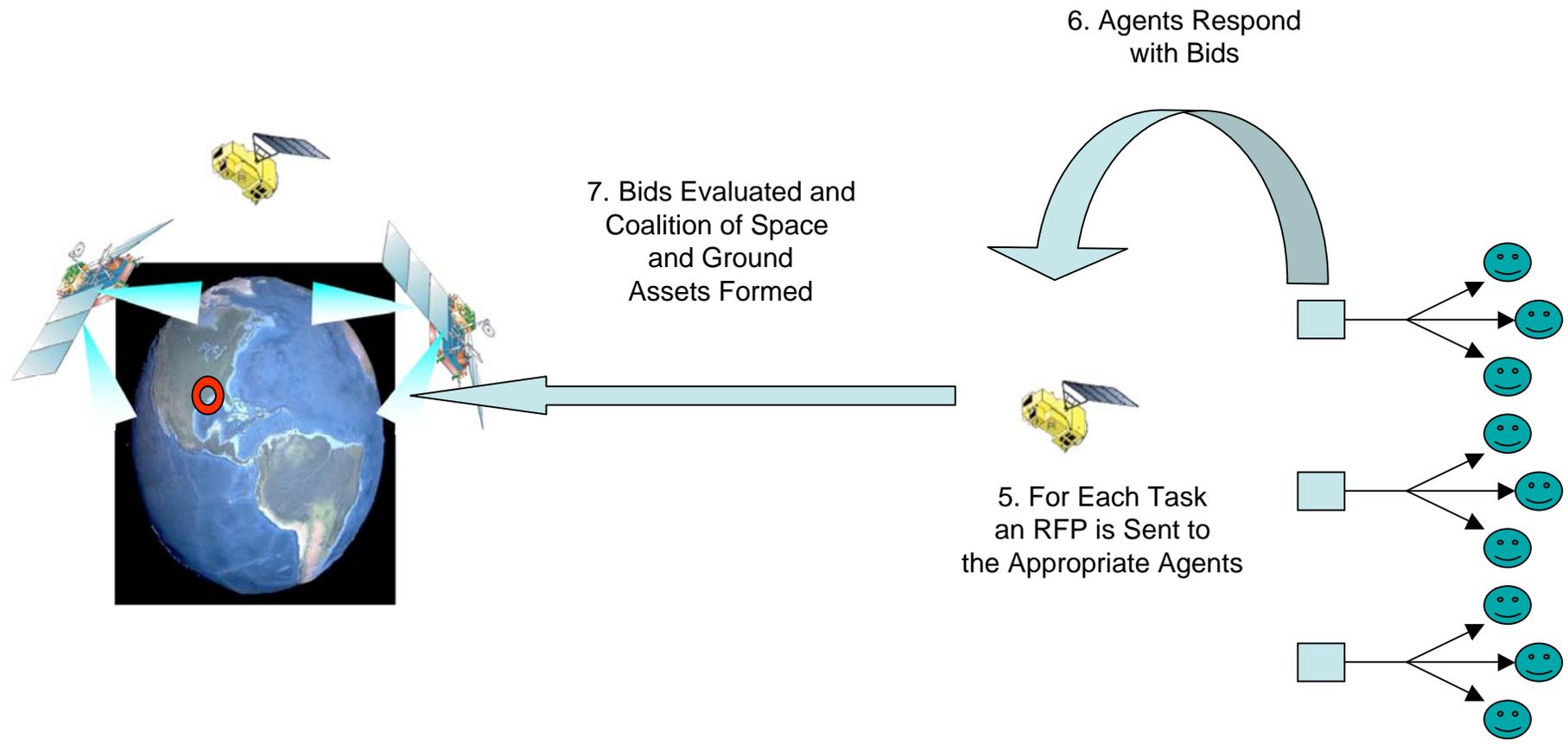
Architecture of the Reasoning Component (2)

- An agent receiving an RFP will evaluate the task abstraction; if it can perform the task as described abstractly, it will compose a bid
- A bid consists of elements as requested in the bid specification
- A bidding agent does not know how its bid will be evaluated, so it will be honest in its bid
- After a bid is submitted it will wait for a pre-specified time for a contract request, and will then time out

Architecture of the Reasoning Component (3)

- The agent who issued the RFP collects all bids
- Bids are evaluated based on domain-specific criteria
- A contract is awarded to the agent with the best bid and the task is assigned
- The agent receiving the contract joins the Sensor Web and the coalition of agents
- Other bidding agents eventually time out





EXPERIMENTS

Experiment 1: Matchmaking, Issuing RFPs, Awarding Contracts

- We modeled real satellite sensors using SensorML and tested them with sensing tasks provided by the users
- We focused on the EO-1 satellite sensors:
 - Advanced Land Imager (ALI) - 10 bands in visual, VNIR and SWIR
 - Hyperion - two spectrometers with 209 bands (198 distinct), one in VNIR (430-1000 nm), one in SWIR (900-2400 nm)
 - LEISA/Atmospheric Corrector (LAC) - 256 contiguous VNIR bands (890-1580 nm)
- We evaluated how they communicated with the Matchmaker and how they interacted with each other to request assistance with tasks and evaluate bids

Step 1: All sensor agents
register their capabilities with
the Matchmaker

```
Terminal — java — 82x46
ALI_Pan: Sending RegisterAMS
Matchmaker: Assigning Queue 0 to Agent ALI_Pan
ALI_Pan: RegisterAMS complete; Assigned Queue 0

ALI_Pan: Registering provided services.
Matchmaker: Agent ALI_Pan's services registered:
    [urn:ogc:def:classifier:sensorSpectrum] Visible
    [urn:ogc:def:classifier:intendedApplication] CloudSensing
    [urn:ogc:def:classifier:intendedApplication] LandSensing

ALI_MS-1: Sending RegisterAMS
Matchmaker: Assigning Queue 1 to Agent ALI_MS-1
ALI_MS-1: RegisterAMS complete; Assigned Queue 1

ALI_MS-1: Registering provided services.
Matchmaker: Agent ALI_MS-1's services registered:
    [urn:ogc:def:classifier:sensorSpectrum] Visible
    [urn:ogc:def:classifier:intendedApplication] CloudSensing
    [urn:ogc:def:classifier:intendedApplication] LandSensing

ALI_MS-2: Sending RegisterAMS
Matchmaker: Assigning Queue 2 to Agent ALI_MS-2
ALI_MS-2: RegisterAMS complete; Assigned Queue 2

ALI_MS-2: Registering provided services.
Matchmaker: Agent ALI_MS-2's services registered:
    [urn:ogc:def:classifier:sensorSpectrum] Visible
    [urn:ogc:def:classifier:intendedApplication] CloudSensing
    [urn:ogc:def:classifier:intendedApplication] LandSensing
```

Step 2: Agents contact the Matchmaker and request the name and address of someone who can help them achieve a task.
Matchmaker responds.

```
Terminal — java — 92x38
Manager: Sending RegisterAMS
Matchmaker: Assigning Queue 11 to Agent Manager
Manager: RegisterAMS complete; Assigned Queue 11

Manager: Requesting agents with my required services:
    [urn:ogc:def:classifier:sensorSpectrum] VNIR

Matchmaker: Search results sent to Agent Manager
Manager: Capable agents received:

    Hyperion @ Queue 9
        [urn:ogc:def:classifier:intendedApplication] CloudSensing
        [urn:ogc:def:classifier:sensorSpectrum] SWIR
        [urn:ogc:def:classifier:sensorSpectrum] VNIR
        [urn:ogc:def:classifier:intendedApplication] LandSensing

    ALI_MS-4 @ Queue 4
        [urn:ogc:def:classifier:intendedApplication] CloudSensing
        [urn:ogc:def:classifier:sensorSpectrum] VNIR
        [urn:ogc:def:classifier:intendedApplication] LandSensing

    ALI_MS-4p @ Queue 5
        [urn:ogc:def:classifier:intendedApplication] CloudSensing
        [urn:ogc:def:classifier:sensorSpectrum] VNIR
        [urn:ogc:def:classifier:intendedApplication] LandSensing

    LAC @ Queue 10
        [urn:ogc:def:classifier:intendedApplication] CloudSensing
        [urn:ogc:def:classifier:intendedApplication] WaterVaporCorrection
        [urn:ogc:def:classifier:sensorSpectrum] VNIR
        [urn:ogc:def:classifier:intendedApplication] LandSensing

    ALI_MS-5p @ Queue 7
        [urn:ogc:def:classifier:intendedApplication] CloudSensing
        [urn:ogc:def:classifier:sensorSpectrum] VNIR
        [urn:ogc:def:classifier:intendedApplication] LandSensing
```

Step 3: Sensor agent creates bid and sends to appropriate agents

```
Terminal — java — 99x13
Manager: Sending out preContracts:
  task abstraction:
    (Integer) wavelength = '1200'
  bid specification:
    [bandPerformance, framesPerSecond, signalToNoiseRatio, resolution]

Manager: Potential contracts sent:
preContract[Hyperion]
preContract[ALI_MS-4]
preContract[ALI_MS-4p]
preContract[LAC]
preContract[ALI_MS-5p]
```

- An agent may not respond to a bid request if it does not satisfy the *bid abstraction*
- The response may only contain the items indicated in the *task specification*

Step 4: Agents submit bids based on the bid specification

```
Terminal — java — 100x37
ALI_MS-4: Task abstraction received.
ALI_MS-4: Instrument does not meet task abstraction.

Hyperion: Task abstraction received.
ALI_MS-4p: Task abstraction received.
ALI_MS-4p: Instrument does not meet task abstraction.

Hyperion: Instrument meets task abstraction. Sending bid:
(Quality) bandPerformance = 'HIGH'
(Integer) framesPerSecond = '220'
(Quality) signalToNoiseRatio = 'GOOD'
(Double) resolution = '30.0'

LAC: Task abstraction received.
ALI_MS-5p: Task abstraction received.
LAC: Instrument meets task abstraction. Sending bid:
(Quality) bandPerformance = 'AVERAGE'
(Integer) framesPerSecond = '28'
(Quality) signalToNoiseRatio = 'GOOD'
(Double) resolution = '250.0'

Manager: Bid received from Hyperion

ALI_MS-5p: Instrument meets task abstraction. Sending bid:
(Quality) bandPerformance = 'GOOD'
(Integer) framesPerSecond = '226'
(Quality) signalToNoiseRatio = 'AVERAGE'
(Double) resolution = '30.0'

Manager: Bid received from LAC

Manager: Bid received from ALI_MS-5p

Manager: preContracts removed due to timeout [preContract[ALI_MS-4]]

Manager: preContracts removed due to timeout [preContract[ALI_MS-4p]]
```

Step 5: Bids are evaluated

- The bid specification items are evaluated
- Currently we judge each one separately, and select the sensor with the best overall bid
 - This will change in the future, since bid specification items may interact, and trade-offs could be made (e.g. better SNR vs. better resolution)

Step 6: Contract is Awarded

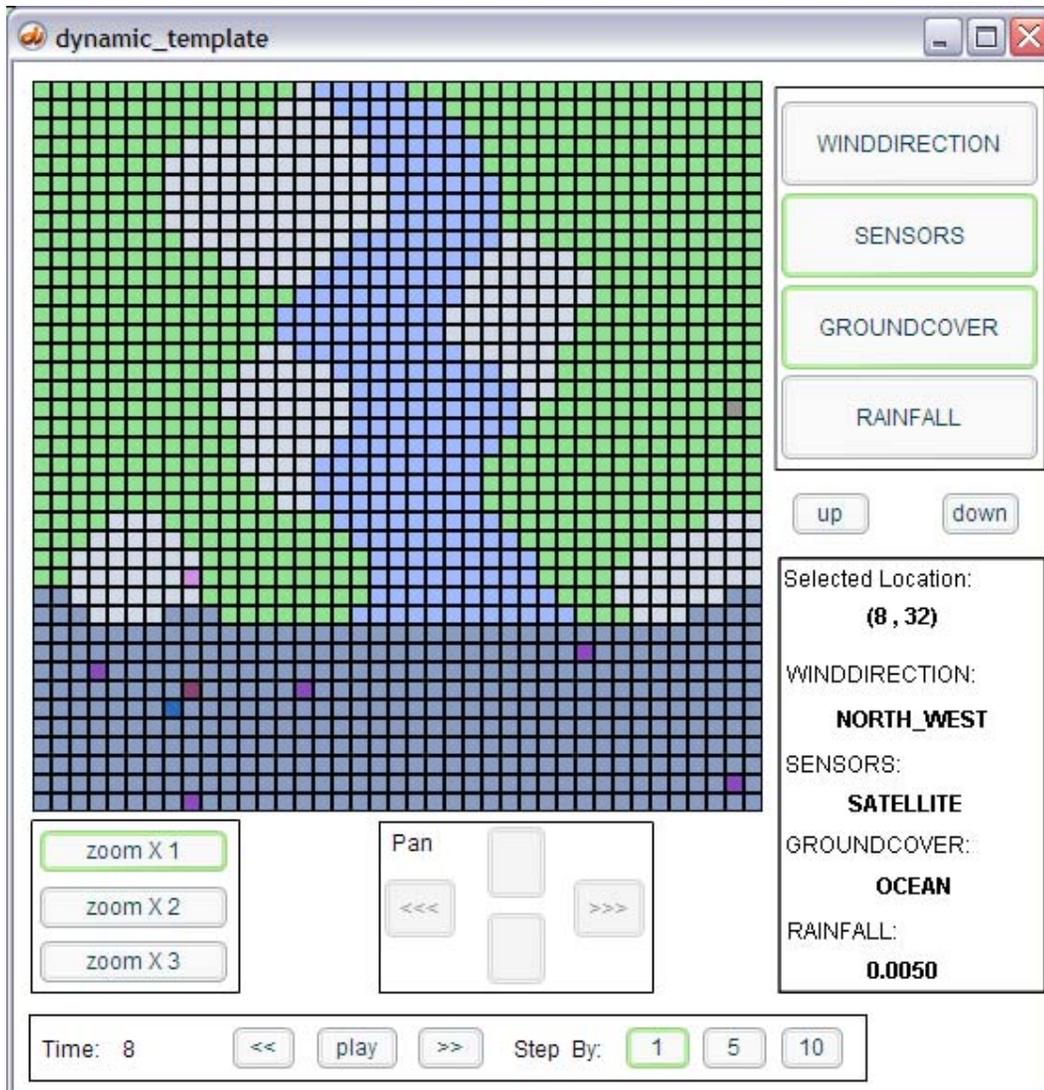
```
Terminal — java — 75x8
Manager: Bids ready for evaluation
Manager: Best contract selected: preContract[Hyperion]

Hyperion: Bid accepted!

LAC: preContract timed out; bid must not have been accepted.
ALI_MS-5p: preContract timed out; bid must not have been accepted.
□
```

Experiment 2: Forming a Coalition of Sensors

- We developed a simulator of simple events and environments
- We modeled a number of sensors (temperature, wind, pressure, rain fall, etc.) using SensorML
- Each sensor controls a number of *event monitors*
- Event monitors look for specific values of observables (e.g. $\text{Temp} > X^0$), trends (e.g. Temp increased by $X\%$ over Y time units), averages (e.g. Temp average X^0 over Y time units), etc.
- We developed a hurricane phenomenon in the simulator
- We used a simplified meteorological model to allow a sensor to identify sensing tasks that need to be performed to verify the existence of the hurricane and then track it
- To achieve this the sensor will create a sensor coalition

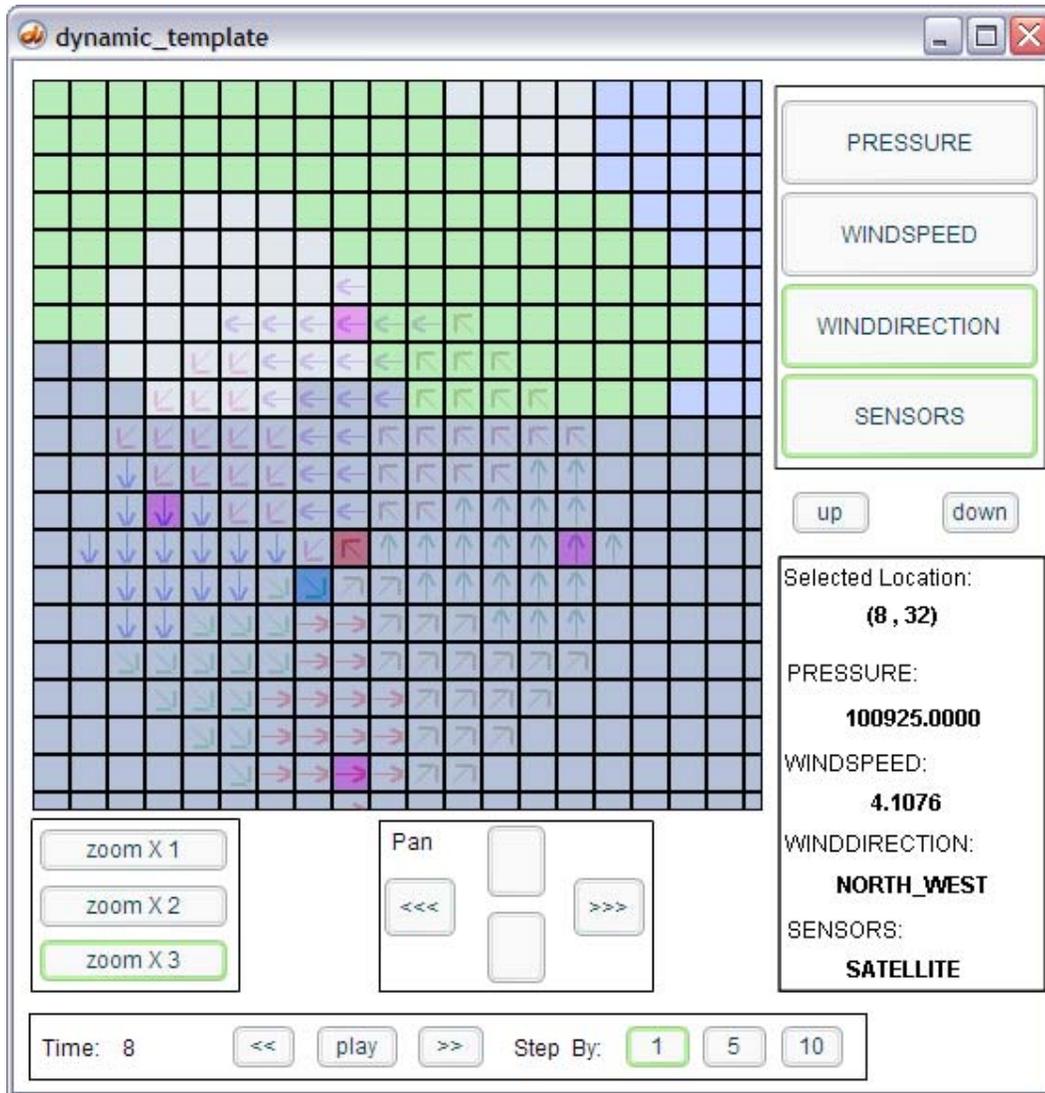


Simulated Environment

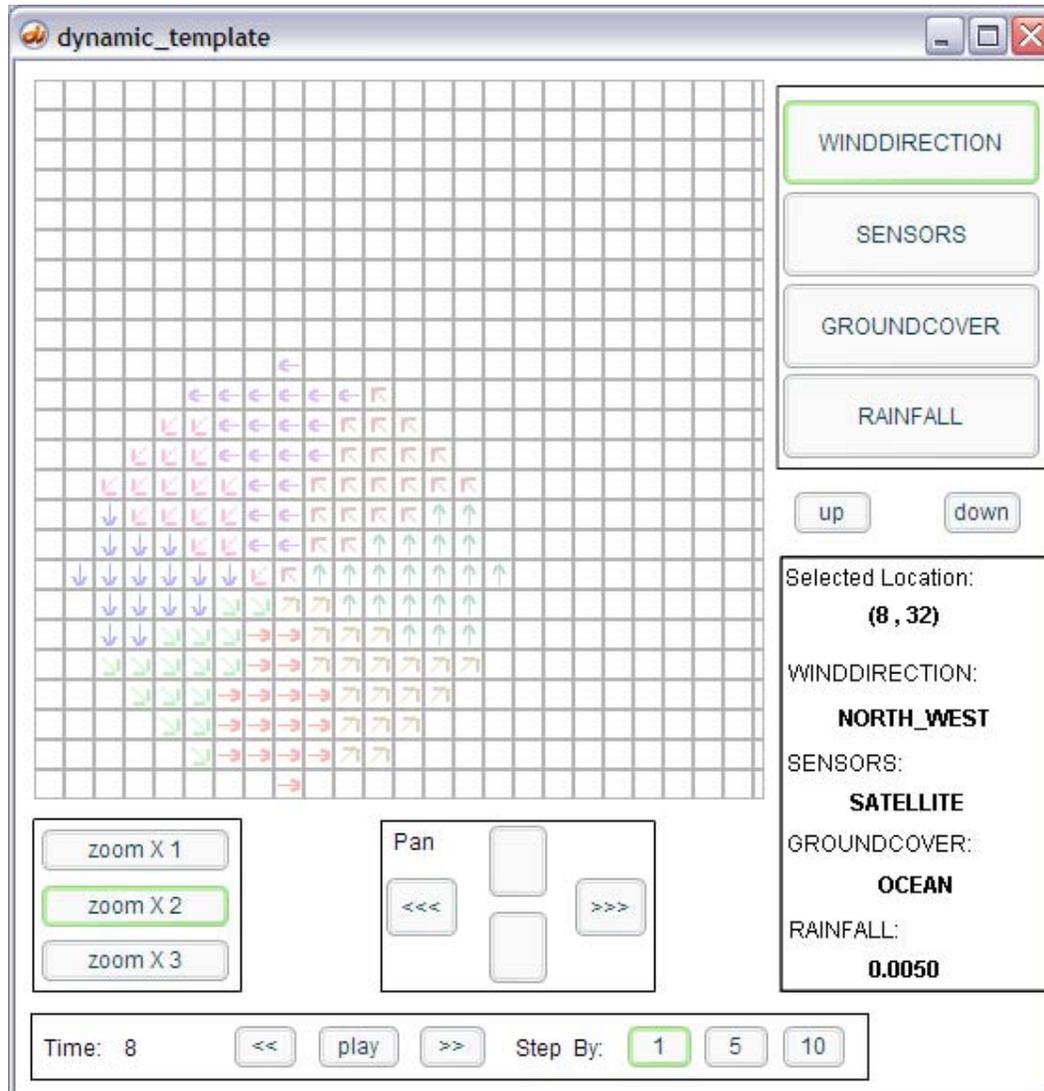
- Green: Agricultural
- Gray: Urban
- Light Blue: River
- Dark Blue: Ocean

- Blue: Pressure Sensor
- Purple: Wind Sensors
- Red: Satellite Sensor

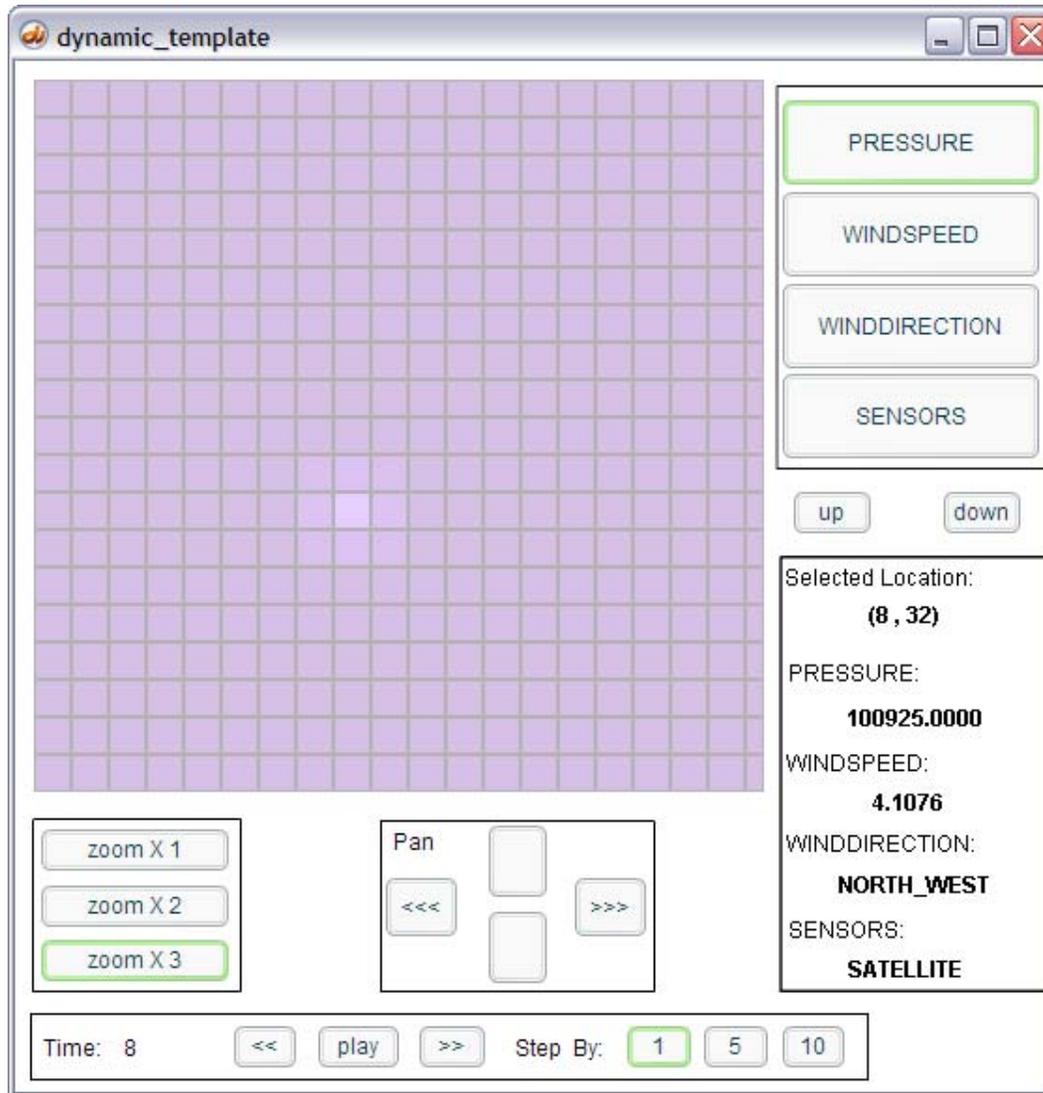
All sensors represented in SensorML



View of cyclonic winds



Looking at
the wind layer only



Zoom of pressure layer with hurricane eye

Process

- Satellite sensor notes a significant temperature event over the ocean
- This event led to two different scenarios of creating a Sensor Web
- In scenario 1 the satellite sensor wishes to create a coalition of one pressure sensor and four wind sensors in the same area
- In scenario 2 the satellite sensor wants to create a coalition of one pressure sensor and four wind sensors in four different areas
- In both cases the agent:
 - creates tasks for Pressure and Wind sensing
 - contacts Matchmaker
 - sends RFPs
 - creates coalition

Task is to find a wind sensor north of the temperature sensor

SECOND TASK A: CYCLONE SENSING (NORTH)

Manager: [ContractDealer]: Requesting agents with my required services:

[urn:ogc:def:classifier:sensorType] Wind

Ask the Matchmaker to identify wind sensors

Matchmaker: Search results sent to Agent Manager

Manager: [ContractDealer]: Capable agents received:

W7_Wind @ Queue 8

[urn:ogc:def:classifier:intendedApplication] Wind Sensing

[urn:ogc:def:classifier:sensorType] Wind

W1_Wind @ Queue 2

[urn:ogc:def:classifier:intendedApplication] Wind Sensing

[urn:ogc:def:classifier:sensorType] Wind

W3_Wind @ Queue 4

[urn:ogc:def:classifier:intendedApplication] Wind Sensing

[urn:ogc:def:classifier:sensorType] Wind

Matchmaker responds with list of sensors

Manager: [ContractDealer]: Sending out preContracts
for task [Cyclone Sensing (North)]:

[[task abstraction]]:

(Quality) accuracy = 'HIGH'

[[bid specification]]:

(Location) location

(Quality) resolution

(Quality) signalToNoiseRatio

RFPs are sent out
In order to respond a sensor must
provide HIGH accuracy

The bid must
contain information about
sensor location, resolution, and SNR

PrimitiveList: Comparing
(Quality) signalToNoiseRatio = 'AVERAGE'
(Quality) resolution = 'POOR'
(Location) location = '37,37'
to
(Location) location = '32,8'
(Quality) resolution = 'LOW'
(Quality) signalToNoiseRatio = 'LOW'
PrimitiveList: [Score] -29

A submitted bid

Required characteristics

Evaluation score

Process (II)

- The sensors that are closest to the required location are compared based on their bids
- The top N of them are selected, where N is predefined
 - In scenario 1 $N=4$
 - In scenario 2 $N=1$
- The tasks are then assigned to the coalition/Sensor Web

In summary

- We have developed a methodology that combines Matchmaking with Contract Nets and allows intelligent sensor agents to identify other sensors that can help them complete complex sensing and processing tasks
- Our approach allows sensors and sensing tasks to be defined on different levels of abstraction, each level appropriate for specific operations of the Sensor Web
- We have implemented and tested it on real EO-1 NASA sensors
- We also used this architecture to create a heterogeneous sensor coalition to sense and track a simulated meteorological phenomenon