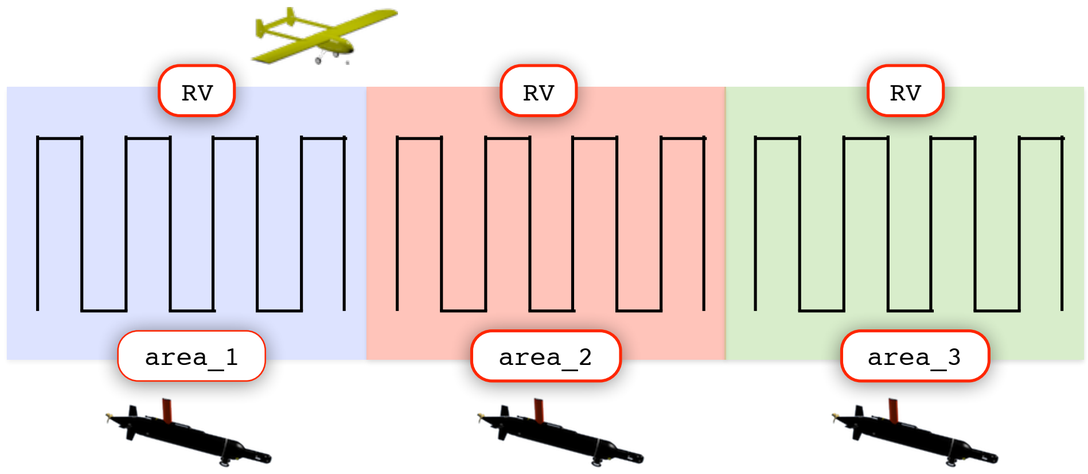
NVL : A Coordination Language For Unmanned Vehicle Networks

E. R. B. Marques[1], M. Ribeiro[1,2], J. Pinto[2], J. B. Sousa[2], F. Martins[1]

[1]LaSIGE/FCUL, Universidade de Lisboa [2]LSTS/FEUP, Universidade do Porto

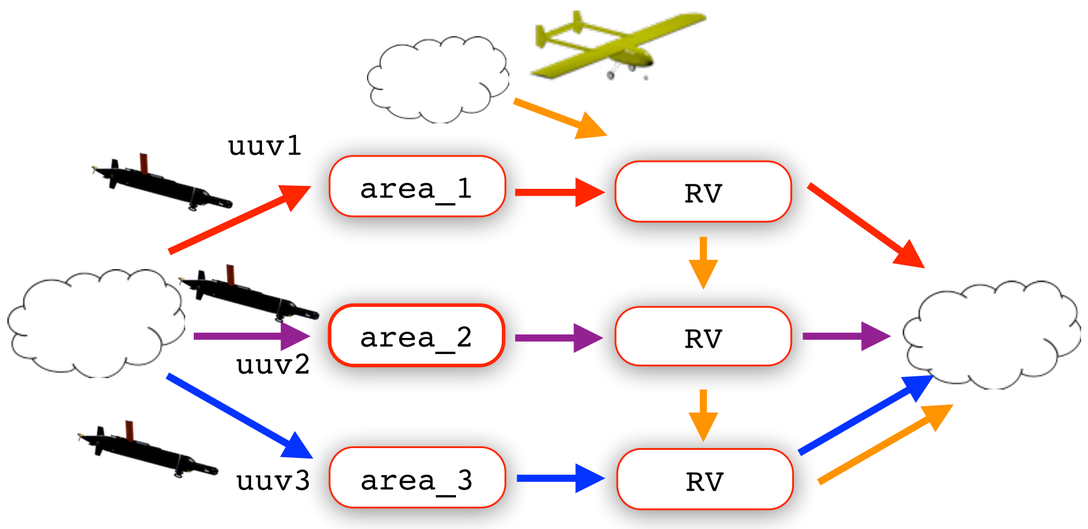
EXAMPLE SCENARIO

Three UUVs (unmanned underwater vehicles) are used for environmental data sampling (e.g., bathymetry measurements) across a given ocean region, together with an UAV (unmanned aerial vehicle) that acts as a "data mule" to collect the data from the UUVs. The sampling region is distributed evenly between the three UUVs, corresponding to three distinct tasks that take form as "row pattern" maneuvers.



Spatial distribution of tasks

The UUVs emerge after data sampling and the UAV collects their data through three instances of a cooperative rendezvous task, each of which involving the UAV and each of the UUVs. A UAV-UUV rendezvous makes the UAV move close to the UUV operation area for improved connectivity first, after which the actual data transfer proceeds. The rendezvous may loop if a single round of data transfer is insufficient.



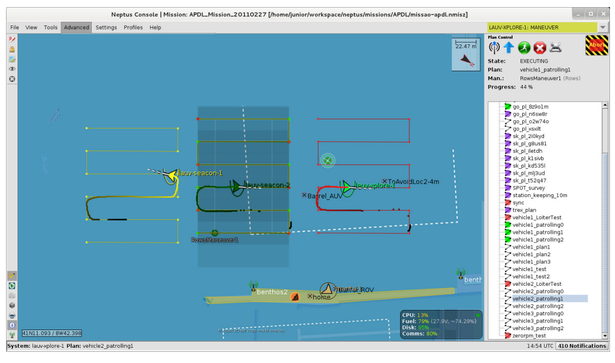
Execution flow

EXPERIMENTS

We conducted simulation and field tests for the NVL prototype. The field tests took place at the Leixões harbour near Porto and involved LAUV Seacon vehicles developed at LSTS [1].

LAUV Seacon

Monitoring task execution using Neptus



We used the Neptus system for the specification of tasks as IMC maneuver plans [2]. During program execution, the progress of tasks could also be monitored using a Neptus console.

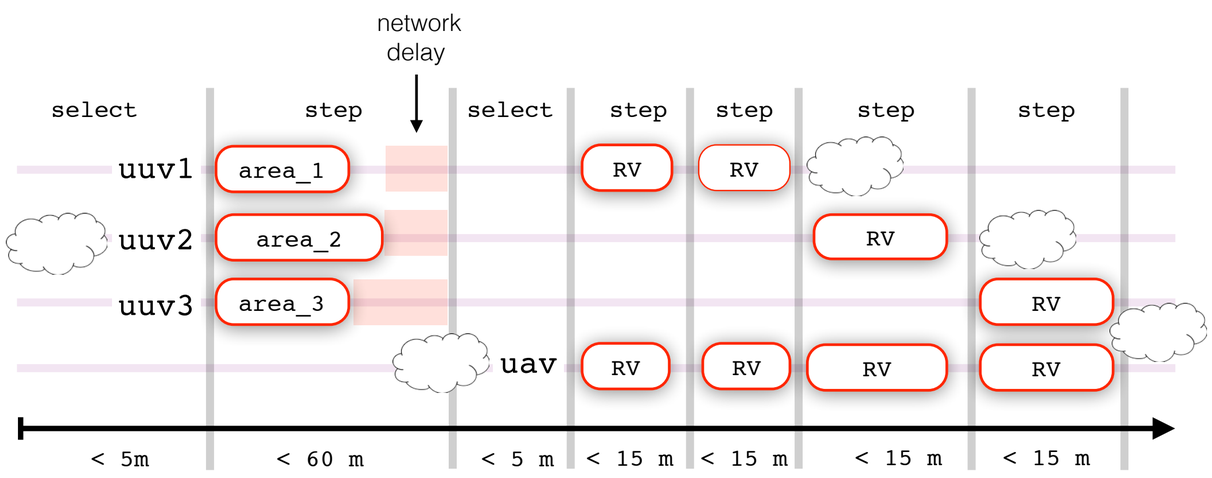
Plots for the example scenario (simulation environment)

NVL PROGRAM

A program is defined by task declarations, placeholders for external implementation, plus procedures made up of instruction sequences. A program's execution comprises the dynamic selection of vehicles from the network "cloud" through **select** instructions, and vehicle bindings to task execution through **step** instructions. Variables and imperative-like constructs (e.g., **do-while**) structure a program's control and data flows. In the example program (right), the UUVs are first selected from the network. The following **step** then fires the data sampling tasks concurrently, one per UUV. When the data sampling step terminates for all of the UUVs, the UAV is selected and engages in rendezvous with each of the UUVs. For each UUV-UAV rendezvous, the rendezvous procedure fires the cooperative RV task once or more (execution loops if the task yields output moreData). At the end of the program, the vehicles are released back to the network cloud.

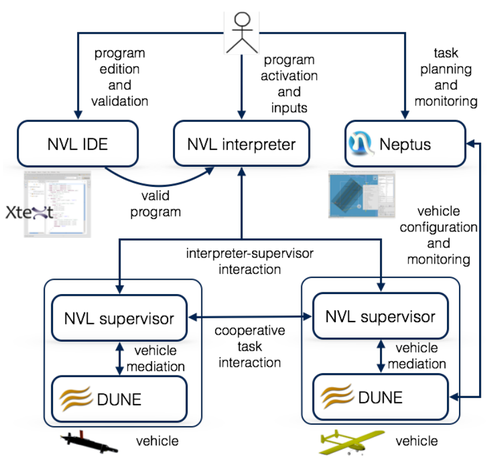
```
// Task declarations
task area_1 (vehicle uuv);
task area_2 (vehicle uuv);
task area_3 (vehicle uuv);
task RV (vehicle uav,
         vehicle uuv)
  yields done, moreData;

// Main procedure
proc main() {
  // Select UUVs.
  select 5 m {
    uuv1 uuv2 uuv3
      (type: "UUV")
  } then {
    // Data sampling tasks.
    step 60 m {
      area_1(uuv1)
      area_2(uuv2)
      area_3(uuv3)
    }
    // Select UAV.
    select 5 m {
      uav (type: "UAV")
    } then {
      // Rendezvous
      call rvproc(uav, uuv1)
      call rvproc(uav, uuv2)
      call rvproc(uav, uuv3)
    }
  }
}
// Rendezvous procedure
proc rvproc
(vehicle uav, vehicle uuv) {
 do {
    step 15 m {
      rvRes = RV(uav, uuv)
    }
  }
  while (rvRes = moreData)
}
```

NVL program listing



Program execution

IMPLEMENTATION



Implementation architecture

The prototype implementation of NVL consists of an Eclipse IDE plug-in for program edition and validation, an interpreter for program execution, and supervisor modules that run onboard each vehicle. The implementation builds on top of the open-source LSTS toolchain [2] (`http://github.com/LSTS`) comprising: IMC, a message-based interoperability protocol; DUNE: a platform for onboard software; and the Neptus command-and-control infrastructure.

REFERENCES

[1] L. Madureira, A. Sousa, J. Braga, P. Calado, P. Dias, R. Martins, J. Pinto, and J. Sousa. The Light Autonomous Underwater Vehicle: Evolutions and networking. In *Proc. Oceans*. IEEE, 2013.

[2] J. Pinto, P.S. Dias, R. Martins, J. Fortuna, E.R.B. Marques, and J. Sousa. The LSTS Toolchain for Networked Vehicle Systems. In *Proc. Oceans*. IEEE, 2013.