

DISTRIBUTED TECHNOLOGY FOR GLOBAL DOMINANCE

Peter Simon Sapaty

*Institute of Mathematical Machines and Systems
National Academy of Sciences
Glushkova Ave 42, 03187 Kiev Ukraine
Tel: +380-44-5265023, Fax: +380-44-5266457
sapaty@immsp.kiev.ua*

Keywords: Global dominance, spatial scenarios, world processing language, distributed interpretation, emergency management, sensor networks, directed energy systems, avionics, electronic warfare, distributed objects tracking, collective behavior.

Abstract: A flexible, ubiquitous, and universal solution for management of distributed dynamic systems will be presented. It allows us to grasp complex systems on a higher than usual, semantic level, penetrating their infrastructures, also creating and modifying them, while establishing local and global dominance over the system organizations and coordinating their behavior in the way needed. The approach may allow the systems to maintain high runtime integrity and automatically recover from indiscriminate damages, preserving global goal orientation and situation awareness in unpredictable and hostile environments.

1 INTRODUCTION

We are witnessing a rapid growth of world dynamics caused by consequences of global warming, globalization of economy, numerous ethnic, religious and military conflicts, and international terrorism. To match this dynamics and withstand numerous threats and possible adversaries, effective integration of any available human and technical resources is crucial. These resources may be scattered and emergent, lacking the infrastructures and authorities for organization of the solutions needed, in real time and ahead of it.

Just communication between predetermined parts and systems with possible sharing a common vision, often called “interoperability”, may not be sufficient. The whole distributed system (or system of systems) should rather represent a highly dynamic and integral organism, in which parts may be defined and interlinked dynamically in subordination to the global organization and system goals, which can vary at runtime, with the coined term “overoperability” (Sapaty, 2002) becoming more appropriate.

A related ideology and accompanying information & control technology, allowing us to provide a much higher than usual level of system understanding and control, will be outlined in this paper.

2 THE WORLD PROCESSING PARADIGM

Within the approach developed, a network of intelligent modules (U, see in Fig. 1), embedded into important system points, collectively interprets mission scenarios in a special high-level language, which can start from any nodes, covering the networked systems at runtime.

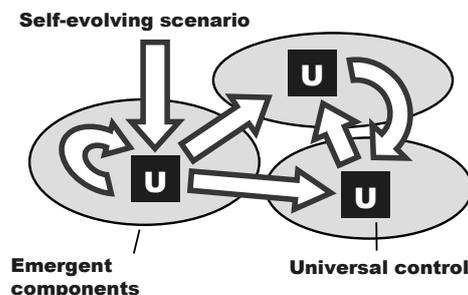


Figure 1: Runtime coverage of a distributed system.

The system “conquering” scenarios are integral and compact, being often capable of self-recovery after damages. They may be created on the fly, as traditional synchronization, data, code, and agents handling and exchanges are effectively shifted to the automatic implementation. This (parallel and fully distributed, without central resources) spatial process can take into account details of the environments, which may be unpredictable and hostile, in which mission scenarios evolve.

Initially represented in a unified and compact form, the scenario and resources which may be needed for

its development, can start from any system point (as shown in Fig.2).

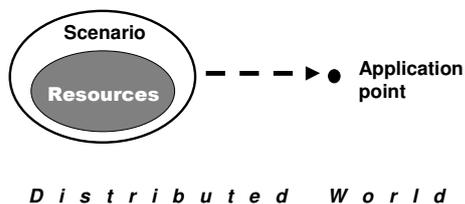


Figure 2: The initial state.

The scenarios can self-split, replicate, and modify while covering the distributed world or its part(s) needed at runtime, bringing operations and (both virtual and physical) resources into different points, also lifting, activating, and spreading further other scenarios and resources, already accumulated in the navigated world, as in Fig. 3.

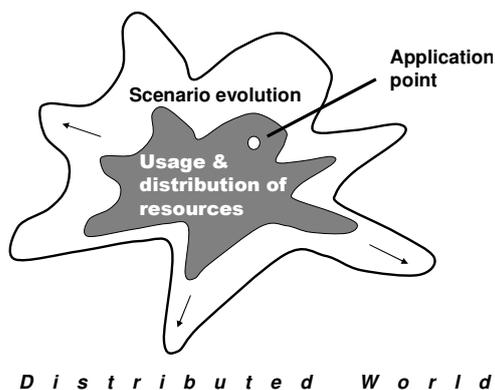


Figure3: Spreading operations and resources.

This is causing movement of information and physical matter, as well initiating interactions between manned and unmanned components, command and control (C2) including, as in Fig. 4 (S is for spatial scenarios or their parts, and R – for resources to implement the scenarios).

The main difference of this approach with the other works is that it describes on a higher level, in a concise way, of what the system should do or how should behave as a whole, while delegating numerous routines of partitioning into components (agents), with their interaction and synchronization, to the effective automatic level, while other approaches used to do the latter manually, and from the start. The approach can, however, describe and implement the system organization and its behavior at any levels needed, which may include:

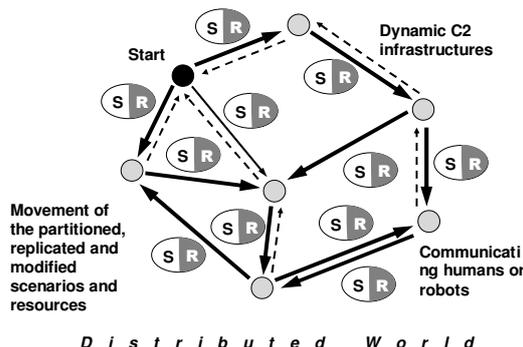


Figure 4: Resultant interactions between system parts.

- Most general, semantic, task formulation.
- Explicit projecting intelligence, information, matter, and power into particular physical or virtual locations, with doing jobs directly in the places reached, and if needed, cooperatively.
- Creating new active physical, virtual, or combined worlds, and organizing & coordinating their activity.
- Setting up implementation details, at any levels, say, for optimization of the use of scarce resources.

3 THE WORLD PROCESSING LANGUAGE (WPL)

This ideology and technology are based on the World Processing Language, WPL (Sapaty, 2005) describing what to do in distributed spaces rather than how to do, and by which resources (or even system organization), leaving these to the automatic interpretation in networked environments. The WPL fundamentals include:

- Association of any action with a position in physical, virtual, or combined space.
- Working with both information and physical matter.
- Runtime creation of distributed knowledge networks.
- Unlimited parallelism.
- Free movement or navigation in physical, virtual, or combined worlds.
- Fully distributed decision making with high integrity as a whole.
- Automatic command and control.

It is a higher-level language to efficiently command and control emergent human teams and armies. It is also a fully formal language suitable for automatic interpretation by mobile robots and their groups. Due to peculiar syntax and semantics, its parallel interpretation in distributed systems is straightforward, transparent, and does not need any central resources. Such complex problems as synchronization of multiple activities and collective (swarm as well as centrally or hierarchically controlled) behavior can be solved automatically by the networked interpreter, without traditional load on human managers and programmers.

This dramatically simplifies application programming, which is often hundreds of times more concise (and simpler) than in traditional programming languages. WPL allows for a direct access to the distributed world, performing any operations in any its points over local or remote data, which may represent both information and physical matter. Navigating in the world, WPL can modify it or even create from scratch, if required. Different movements and operations can be performed simultaneously and in parallel, and these may be free or may depend on each other.

WPL has a recursive syntax which can be expressed on the top level as follows (square brackets are for an optional construct, braces mean construct repetition with a delimiter at the right, and vertical bar separates alternatives).

```

wave    → constant | variable | [ rule ] ( { wave , } )
constant → information | matter
variable → nodal | frontal | environmental
rule     → evolution | fusion | verification | essence
evolution → expansion | branching | advancing |
           repetition | granting
fusion   → echoing | processing | constructing |
           assignment
verification → comparison | membership | linkage
essence  → type | usage

```

A rule is a very general construct, which, for example, can be:

- Elementary arithmetic, string or logic operation.
- Hop in physical, virtual, or combined space.
- Hierarchical fusion and return of (remote) data.
- Parallel and distributed control.
- Special context for navigation in space.
- Sense of a value for its proper interpretation.

Different types of variables, especially when used together, allow us to create efficient spatial algorithms which work “in between components”

of distributed systems rather than in them. The variables called *nodal* can store and access local results in the system points visited, while others ones can move data in space together with the evolving control (*frontal variables*) or can access and impact the world navigated (*environmental variables*).

4 ELEMENTARY EXAMPLES

4.1 Setting Global Dominance

Let us assume that a node in the distributed system (see Fig.5) wants to establish the field of its dominance over other nodes which have a lower rank than itself (here the content, or name, of each node is considered as its rank).

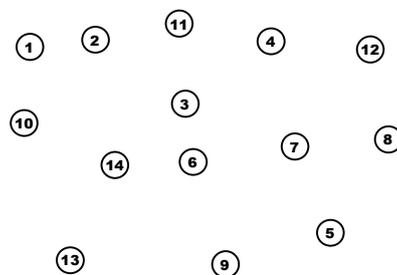


Figure 5: Distributed system nodes.

The following parallel and distributed program, applied in this node, spreads its own rank throughout the whole system in the frontal variable Rank. This puts the rank into the nodal variable Dominance in each visited node, if Rank exceeds the already existing value in Dominance (by the first access, the variable Dominance is assigned the value of the personal rank of each node).

```

frontal Rank = CONTENT;
nodal Dominance;
repeat (
  if (Dominance == nil, Dominance = CONTENT);
  if (Dominance < Rank,
    (Dominance = Rank; hop all neighbors),
    stop))

```

If applied, say, in node 11, this distributed program establishes only a partial dominance in the system, as shown in Fig. 6.

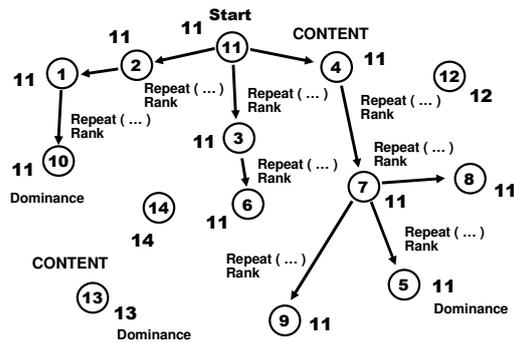


Figure 6: Resulting in partial dominance.

That will not be the case for node 14, which will set up its absolute dominance over the whole world by the program above, as in Fig. 7.

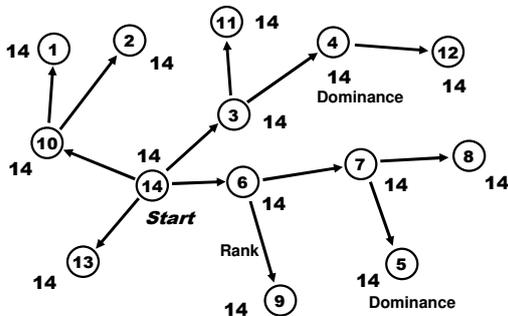


Figure 7: Setting absolute dominance of node 14.

4.2 Creating Infrastructures in the Distributed Space

It is easy to set up any infrastructures in the distributed space by the approach presented, with any topology. The following program, starting from node 3, will create (in parallel and distributed way) the networked structure shown in Fig. 8 over the set of already existing nodes.

```

hop node 3;
create links ((L6# 2; L8#14),
(L7#12; L5#7; L4#3), (L1#14; L2#9; L3#7))

```

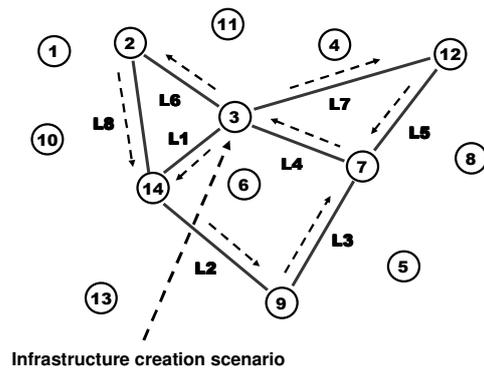


Figure 8: Creating a distributed infrastructure.

Any functionality can be associated with both nodes and links of the obtained infrastructure at runtime, which will be operating as a system for the purpose needed.

4.3. Finding patterns in the Infrastructure

It is convenient to find any patterns in the distributed infrastructures in WPL. Let such a pattern be a triangle, and we would like to find all of them in the infrastructure created. The following spatial program, starting in any node, does this, with listing resultant nodes of the triangles in their descending ranks.

```

hop all nodes; frontal (Triangle) = CONTENT;
twice (hop all links; CONTENT < BACK;
Triangle &= CONTENT);
hop all links; element (Triangle, first) == CONTENT;
output Triangle

```

The result, issued in the node where the program was injected, will be as: (14, 3, 2), (12, 7, 3) -- see Fig.9.

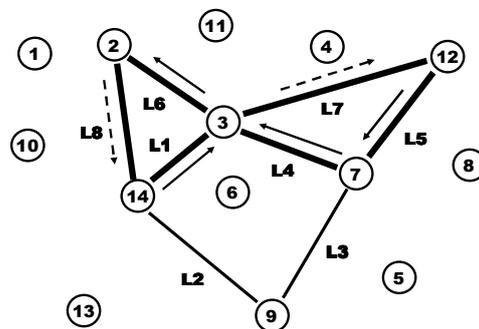


Figure 9: Finding all triangles in the infrastructure.

5 WPL INTERPRETER

The WPL interpreters may be embedded in internet hosts, robots, mobile phones, or smart sensors (an interpreter can also be a human being herself, understanding and executing high-level orders in WPL, while communicating with other humans or robots via WPL too). The interpreters may be concealed, if needed (say, to work in a hostile system); they can also migrate freely, collectively executing (also mobile) mission scenarios, resulting altogether in the extremely flexible and ubiquitous system organization.

The basic WPL interpreter organization (Sapaty, 1993, 1999, 2005) is shown in Fig. 10, which may have both software and hardware implementation (the latter as “wave chip”).

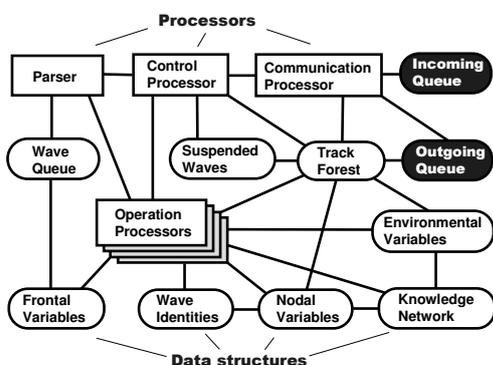


Figure 10: The WPL interpreter architecture.

The interpreter consists of a number of specialized modules working in parallel and handling and sharing specific data structures, which are supporting persistent virtual worlds and temporary hierarchical control mechanisms. The whole network of the interpreters can be mobile and open, changing the number of nodes and communication structure between them.

The heart of the distributed interpreter is its spatial track system enabling hierarchical command and control and remote data and code access, with high integrity of emerging parallel and distributed solutions. The interpreters can be embedded into any other systems, like mobile robots, allowing them to behave as integral teams, as shown in Fig. 11.

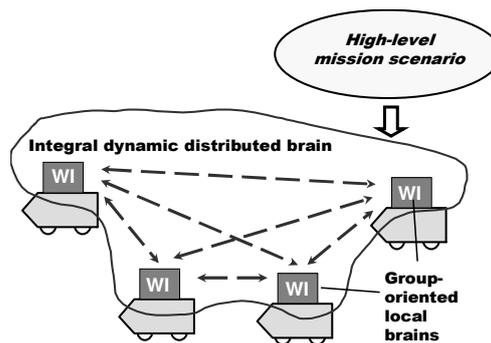


Figure 11: WPL interpreters (WI) forming distributed robotic brain.

6 EMERGENCY MANAGEMENT

Emergency management, EM (Sapaty, Sugisaka, Finkelstein, et al., 2006), due to the increased world dynamics, is becoming one of the hottest topics today. The emergency managers around the world are faced with new threats, new responsibilities, and new opportunities. Novel technologies, like the one of this paper, can alleviate consequences of natural (say, due to global warming) or manmade (like war conflicts) disasters. They can allow law enforcement and intelligence investigators to identify potential terrorist plots and then mount preemptive strikes to stop their plans.

The technology described can help in solving many EM problems by using communicating interpreters embedded in different electronic devices like, for example, laptops or mobile phones, with some disaster situation shown in Fig 12.

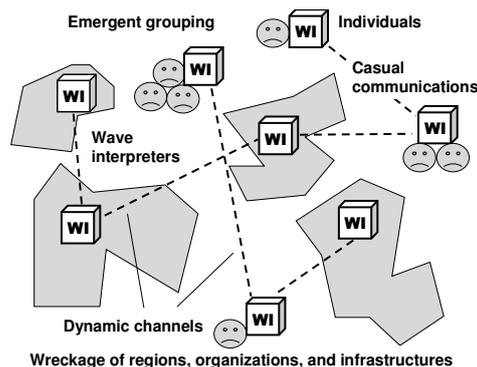


Figure 12: A disaster area with WPL interpreters embedded.

A very simple example may be here as a necessity to count the total number of casualties in the disaster area, on all its affected regions.

The following program can be applied from any WI as an entry one, which can reside within the disaster area or be away from it, and then can self-spread via local communications, organizing the whole region with embedded interpreters to work as an integral spatial supercomputer.

```

frontal Area = <disaster area definition>;
output sum (
  hop (directly, first come, nodes(Area));
  repeat(
    done(count casualties),
    hop(any links, first come, nodes(Area))))

```

More complex operations which can be organized in WPL may include the delivery of relief aid, an organized evacuation from the disaster area, and organization of and cooperation with the rescue teams (which may include robotic components).

7 SENSOR NETWORKS

Sensor networks are a sensing, computing and communication infrastructure that allows us to instrument, observe, and respond to phenomena in the natural environment, and in our physical and cyber infrastructure. The sensors themselves can range from small passive microsensors to larger scale, controllable platforms. Typical applications of wireless sensor networks (WSN) include monitoring, tracking, and controlling. Some of the specific applications are habitat monitoring, object tracking, nuclear reactor controlling, fire detection, traffic monitoring, etc. Any distributed problems can be solved by dynamic self-organized sensor networks working in WPL (Sapaty, 2007a).

Starting from all transmitter nodes, the following program regularly (with interval of 20 sec.) covers stepwise, through local communications between sensors, the whole sensor network with a spanning forest, lifting information about observable events in each node reached, as shown in Fig. 13. Through this forest, by the internal interpretation infrastructure, the data lifted in nodes is moved and fused upwards the spanning trees, with final results collected in transmitter nodes and subsequently sent outside the system in parallel.

```

hop (all transmitters);
loop (
  sleep (20);
  IDENTITY = TIME;
  transmit (

```

```

fuse (
  repeat (free (observe (events));
  hop (directly reachable, first come))))

```

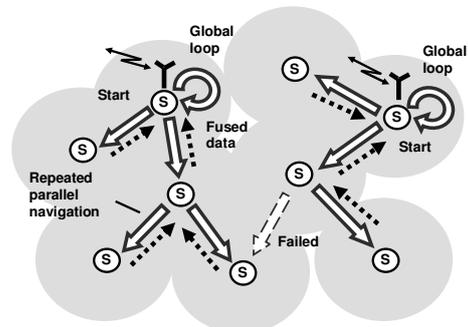


Figure 13: Collecting data by a sensor network.

Another program, below, provides for spanning tree coverage of some distributed phenomenon, with hierarchical collection, merging and fusing partial results got from different sensors into the global picture. The latter will be forwarded to a nearest transmitter via the previously created infrastructure with links infra, as shown in Fig. 14.

```

hop (random, all nodes, detected phenomenon).
loop (
  frontal Full = fuse (
    repeat (
      free (collect phenomenon),
      hop (directly reachable, first come,
        detected phenomenon));
    repeat (hop links (-infra). Transmit Full)
  )

```

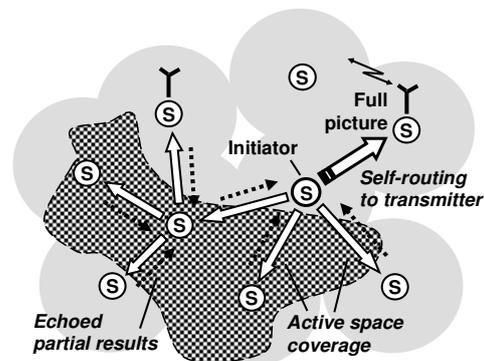


Figure 14: Space coverage with hierarchical assembling of a distributed phenomenon.

In more complex situations, which can be effectively programmed in WPL too, we may have a number of simultaneously existing phenomena, which can intersect in a distributed space. We may also face a combined phenomenon integrating features of different ones. The phenomena (like flocks of birds,

manned or unmanned groups or armies, spreading fire or flooding) covering certain regions may change in size and shape, they may also move as a whole, preserving internal organization. All these situations can be managed in WPL.

8 DIRECTED ENERGY SYSTEMS

Directed energy (DE) systems are of a growing interest for broad applications in the nearest future, especially in infrastructure protection and defense. The DE-based systems will be able to operate under flexible command and control in WPL, restructuring and recovering in unpredictable environments without loss of functionality (Sapaty, Morozov, Sugisaka, 2007).

An elementary DE-based system may consist of a control center, DE source, relay mirror (RM), and target. Using WPL, the system functionality can be set up dynamically, on the fly, as by the following program:

```

sequence (
parallel (
(hop (DE); adjust (RM)),
(hop (RM); adjust (DE, Target))),
(hop (DE); activate (DE)))

```

Three snapshots of the system operation under this program are shown in Figs. 15-17.

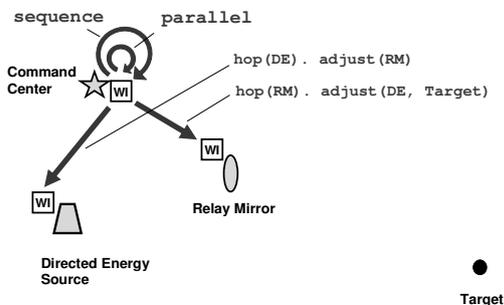


Figure 15: DE system operation, Snapshot 1.

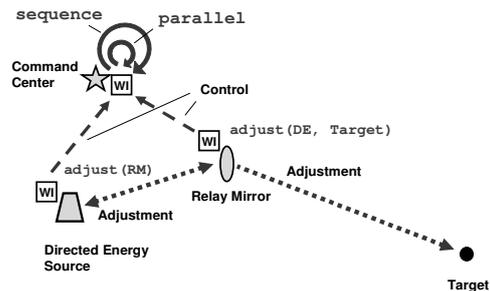


Figure 16: DE system operation, Snapshot 2.

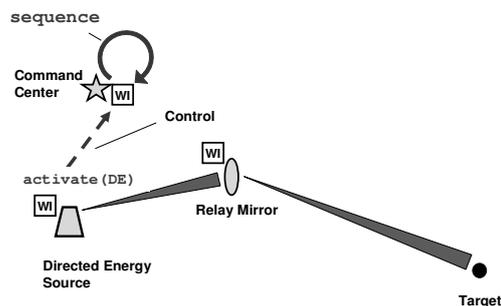


Figure 17: DE system operation, Snapshot 3.

Boeing's Advanced Relay Mirror System (ARMS) concept plans to entail a constellation of as many as two dozen orbiting mirrors that would allow a constant coverage of every corner of the globe. When activated, this would enable a directed energy response to critical trouble spots anywhere.

We will show here, by the program below, how the shortest path tree (SPT) starting from any DE source and covering the whole set of distributed mirrors can be created at runtime with the use of the technology presented. This will enable us to make optimal delivery of the directed energy to any point of the globe. The distributed SPT creation process is shown in Fig. 18.

```

nodal (Distance, Predecessor);
frontal (Length, Range = 400);
hop (DE);
Distance = 0. Length = 0;
repeat (
hop (Range, all);
Length += between (WHERE, BACK);
or (Distance == nil, Distance > Length);
Distance = Length; Predecessor = BACK)

```

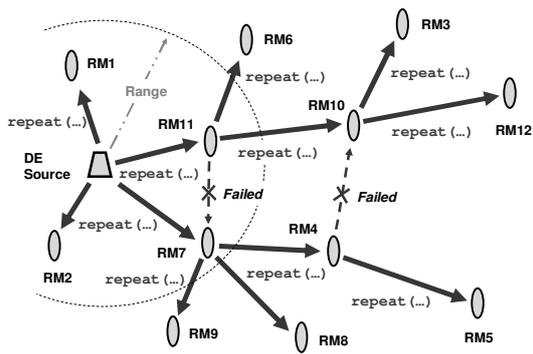


Figure 18: Dynamic shortest path tree through all RMs.

In case the target is defined, the following program forms a path from the DE source to the target via the relay mirrors, using the SPT formed, with a subsequent activation of the DE source to impact the target, as depicted in Fig. 19.

```

adjust (Seen (range), Predecessor);
repeat (
  hop (Predecessor, first);
  adjust (BACK, Predecessor));
activate (DE)

```

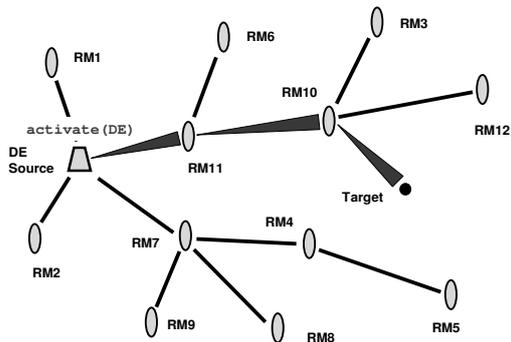


Figure 19: Energy delivery via the path found.

9 ELECTRONIC WARFARE

Electronic warfare (EW) is becoming one of the main technological challenges of this century. All existing and being developed electronic support, attack, and protection measures usually have a very limited scope and effect if used alone. But taken together they may provide a capability for fulfilling the rapidly growing needs. Traditional communication and cooperation between these systems may not be sufficient. They should

comprise altogether a much more integral system of systems with global situation awareness and “global will”, which can be expressed and provided in WPL (Sapaty, 2007).

One of the typical EW tasks is fighting malicious intrusions and viruses in computer networks. Being itself a super-virus on the implementation level, the technology proposed, via the embedded network of WPL interpreters, can simultaneously discover and analyze electronic viruses, with blocking their spread and inferring attack sources. For example, the following scenario can find all virus sources in parallel, as shown in Fig 20:

```

nodal (Trace, Predecessor);
sequence (
  (hop (all nodes);
  nonempty (check general (viruses));
  repeat (
    increment (Trace);
    nonempty (Predecessor = check special (viruses));
    hop (Predecessor))),
output (
  sort (
    hop (all nodes); empty (Predecessor);
    nonempty (Trace); Trace & ADDRESS)))

```

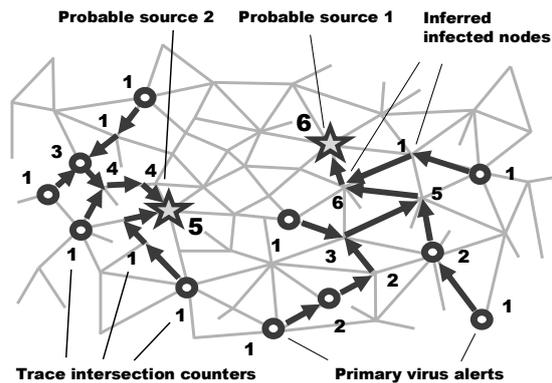


Figure 20: Finding virus sources in parallel.

10 AVIONICS

Avionics, or aviation electronics, represents a substantial share of the cost of any modern flying devices.

- Any avionics system, whether for a single aircraft or a group of them with manned or unmanned units, may be considered as a complex organization consisting of numerous components properly interacting with each other to pursue global goals. This organization can be effectively expressed in WPL on a variety of levels.

- This organization can be made flexible enough to recover from indiscriminate damages and restructure at runtime.
- The WP approach may offer real possibilities for a runtime recovery after damages, including reassembling of the whole system (or what remains of it) from any point.

Implanting communicating WPL interpreters into main components of an aircraft, as universal control modules U (see Fig. 21), may allow us to convert the whole distributed object into a parallel computer capable of solving a variety of complex problems at runtime, including aircraft's safety and recovery (Sapaty, 2008).

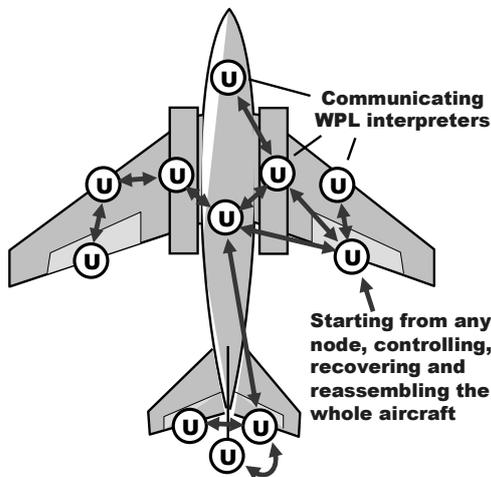


Figure 21: Aircraft self-analysis by the WPL network.

The following program, starting from any point, is collecting availability of vital mechanisms of a damaged aircraft, analyzing their completeness to operate as a system, with making proper decisions (which may include the alarm with emergent evacuation of the crew).

```
nodal Available_Set =
  repeat (
    free (if CONTENT belongs_to
      (left_aileron, right_aileron, left_elevator,
       right_elevator, rudder, left_engine,
       right_engine, left_chassis,
       right_chassis, ...)
      then CONTENT),
    hop_first all_neighbors);
  if sufficient Available_Set
  then control_with Available_Set
  otherwise alarm
```

11 DISTRIBUTED OBJECTS TRACKING

Tracking mobile objects in distributed environments is an important task in a number of areas like air and road traffic, infrastructure protection, national and international crime, or missile defense. The example here relates to tracking aerial objects by a dynamic network of unmanned aerial vehicles, UAVs (Sapaty, 2008), with the following features to be taken into account.

- Each UAV can observe only a limited part of space.
- To keep the whole observation continuous, the object discovered should be handed over between neighboring UAVs during its movement, along with the data accumulated about it.
- The model can catch each object and accompany it individually by the mobile intelligence, while propagating between the WPL interpreters in UAVs.
- Many such objects can be picked up and chased in parallel by a dynamic UAV network.

The following program, starting in all units, catches the object it sees and follows it wherever it goes, if it is not seen from this point any more (its visibility becomes lower than a given threshold).

```
hop all_nodes; Frontal Threshold = 0.1;
frontal Object =
  select_max_visible (aerial, Threshold);
repeat (
  loop (visibility (Object) > Threshold );
  choose_destination_with_max_value (
    hop all_neighbors.
    visibility (Object) > Threshold))
```

A snapshot of a possible situation in a distribute space is shown in Fig. 22. The information about the tracked objects can be accumulated by individual mobile intelligences (Sapaty, Corbin, Seidensticker, 1995), which can cooperate with each other, making individual or collective decisions about the further fate of the objects (e.g. classifying them as friendly or hostile).

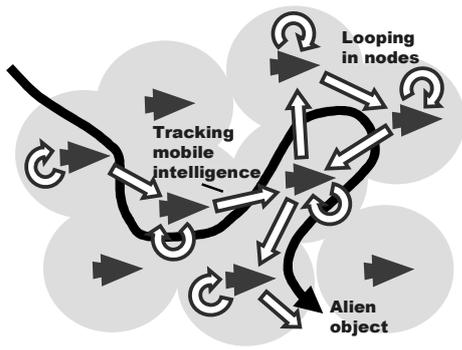


Figure 22: Collective tracking of a mobile object.

12 COLLECTIVE BEHAVIOR

The higher-level, semantic WPL scenarios are well understandable by humans, who can perform jobs written in the language and delegate other jobs to other group members, establishing runtime relations with each other. These scenarios also represent fully formal descriptions that can be effectively interpreted by robots and their groups automatically.

Both human and robotic suitability allow for a fully unified approach to organization of teams that can range from purely human to purely robotic. These teams can be open and emergent, and can operate in unpredictable environments, where team members can indiscriminately fail at any time but the mission scenario, collectively interpreted by the distributed group, can survive and fulfill objectives. The collective team behavior can be based on a loose organization like swarms, or can be strictly and hierarchically controlled. Different solutions in WPL throughout this organizational range are possible, including any combined ones (Sapaty, 2005).

With the initial distribution of units shown in Fig. 23, let us consider a collective swarm-like movement, where each unit randomly, within certain hop limits defining general direction, tries to move in new positions, keeping the established threshold distance to other units.

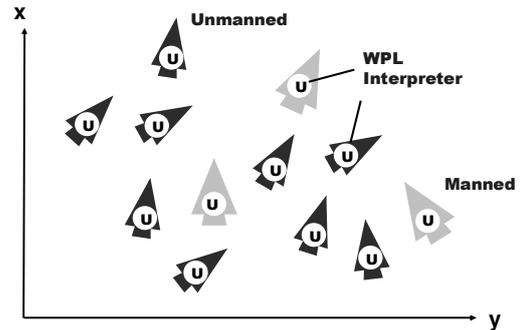


Figure 23: Initial distribution of units.

This can be done by the following program, which can start from any unit, manned or unmanned.

```
nodal (Limits, Range, Shift);
hop all_nodes;
Limits = (dx (0, 8), dy (- 2, 5)); Range = 5;
repeat (
  Shift = random (Limits );
  if empty hop (Shift, Range) then move Shift)
```

A snapshot of the group movement by this spatial program is depicted in Fig.24.

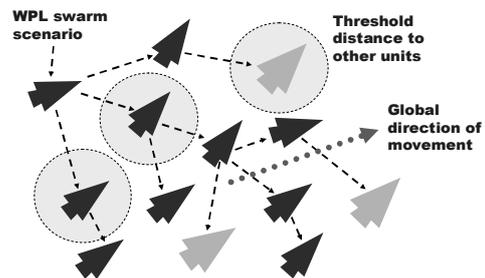


Figure 24: A swarm movement snapshot.

To have more coordinated actions of the group, we may set up a distributed hierarchical infrastructure over it, to be used in command and control and in maintaining global awareness. As the group is distributed in space and distances between units can change, such an infrastructure should be preferably based on the current physical position of the units, with top of the hierarchy to be close to the group's center, in order to optimize global coordination. We will consider here how the topologically central unit can be found at runtime, during the movement within a swarm, and how the C2 hierarchy can be formed starting from this central unit.

The following distributed program, starting from any unit, finds topologically central unit of the distributed swarm, which is shown in Fig. 25.

```

frontal Aver =
  average (hop all_nodes; WHERE);
nodal Center =
  element (
    min (
      hop all_nodes;
      distance (Aver, WHERE) & ADDRESS), 2)

```

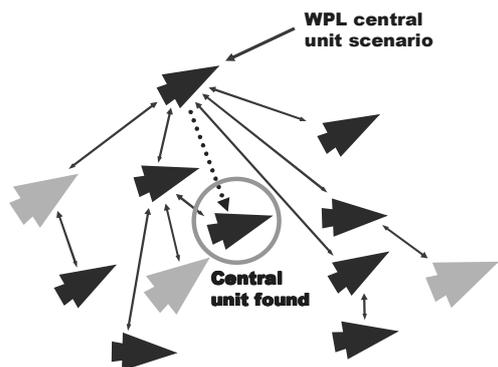


Figure 25: Finding topologically central unit.

Starting from the central unit found, the next program creates runtime hierarchical infrastructure with oriented links *infra*, as shown in Fig. 26.

```

frontal Range = 20.
repeat (
  create_links (
    + infra, first_come, nodes (Range)))

```

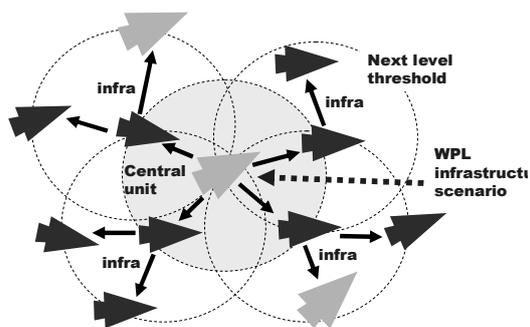


Figure 26: Hierarchical infrastructure built.

This runtime hierarchy created may be effectively used for maintaining global awareness in the distributed space, collection and fusion of targets seen by individual units, spreading the set of collected targets back to all units, which may select the most suitable ones for an individual impact.

The following program, navigating the infrastructure created, follows this scenario, as shown in Fig. 27.

```

repeat (
  if nonempty (
    frontal Seen = Repeat (
      Free (detect targets),
      Hop_links + infra)) then
    repeat (
      free (if TYPE == UAV then
        select_move_shoot Seen),
      hop_links + infra))

```

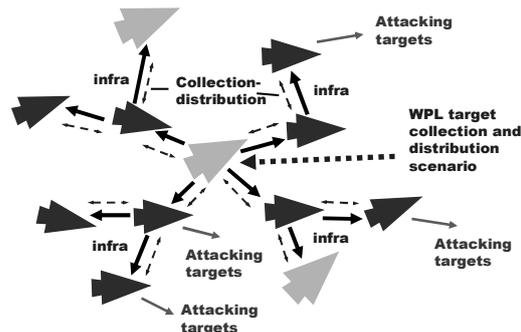


Figure 27: Hierarchical fusion and distribution of targets.

As the whole group moves and distances between separate units may change in the swarm, the programs of finding the center and hierarchical infrastructure may be repeated with a certain regularity, which will help to maintain the group's optimal spatial organization in a distributed environment. Any position in this dynamic hierarchy (the top one including) may happen to be occupied by any unit at any moment of time, regardless of whether it is manned or unmanned.

Many more applications of this world processing paradigm (previously known as WAVE) can be found in (Sapaty, 1999, 2005), also (Sapaty, Morozov, Finkelstein, et al., 2007).

13 CONCLUSIONS

We have touched only some of the areas currently in active investigation for the WP technology being developed. The experience obtained allows us to claim the following.

- The proposed technology converts any distributed system into a universal spatial computer capable of solving complex problems on itself and on the surrounding environment.

- This system, for example, can be a single unit or a group (or army) of them, with individual units being manned or unmanned.
- The whole system is driven by high-level scenarios setting how to behave as a whole and what to do, while omitting traditional implementation details which are effectively delegated to intelligent distributed interpretation system.
- The system scenarios in the World Processing Language are very compact and can be created at runtime, on the fly, swiftly reacting on a rapidly changing environment and mission goals.
- Any scenario can start from any available component and cover the system at runtime, during its evolution.
- The approach may offer real possibilities for a runtime recovery after indiscriminate damages, including reassembling of the whole system (or what remains of it) from any point.
- The technology can help dominate over other distributed system organizations, especially those explicitly based on communicating and interacting parts (agents).

- Sapaty, P., Morozov, A., Finkelstein, R., Sugisaka, M., Lambert, D., 2007. A new concept of flexible organization for distributed robotized systems. *Proc. Twelfth International Symposium on Artificial Life and Robotics (AROB 12th'07)*, Beppu, Japan, Jan 25-27, 8p.
- Sapaty, P., Morozov, A., Sugisaka, M., 2007. DEW in a network enabled environment, *Proc. international conference Directed Energy Weapons 2007*, Feb. 28 - March 1, Le Meridien Piccadilly, London, UK.
- Sapaty, P., 2007. Global management of distributed EW-related systems, *Proc. Electronic Warfare: Operations & Systems 2007*, 19-20 Sept., Thistle Selfridge, London, UK.
- Sapaty, P., 2007a. Intelligent management of distributed sensor networks, In: *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense VI*, edited by Edward M. Carapezza, Proc. of SPIE Vol. 6538, 653812.
- Sapaty, P., 2008. Grasping the whole by spatial intelligence: A higher level for distributed avionics, *Proc. international conference Military Avionics 2008*, Jan. 30 - Feb.1, Café Royal, London, UK.

REFERENCES

- Sapaty, P. S., 1993. *A distributed processing system*, European Patent No. 0389655, Publ. 10.11.93, European Patent Office.
- Sapaty, P. S., Corbin, M. J., Seidensticker, S., 1995. Mobile intelligence in distributed simulations, *Proc. 14th Workshop on Standards for the Interoperability of Distributed Simulations*, IST UCF, Orlando, FL, March.
- Sapaty, P. S., 1999. *Mobile Processing in Distributed and Open Environments*, John Wiley & Sons, ISBN: 0471195723, New York, February, 436p.
- Sapaty, P.S., 2002. Over-Operability in Distributed Simulation and Control, *The MSIAC's M&S Journal Online*, Winter Issue, Volume 4, No. 2, Alexandria, VA, USA, 9p.
- Sapaty, P. S., 2005. *Ruling Distributed Dynamic Worlds*, John Wiley & Sons, New York, May, 256p, ISBN 0-471-65575-9
- Sapaty, P., Sugisaka, M., Finkelstein, R., Delgado-Frias, J., Mirenkov, N., 2006. Advanced IT support of crisis relief missions, *Journal of Emergency Management*, Vol.4, No.4, July/August.