Intelligent Control and Automation, **2012, 3, 117-131**
http://dx.doi.org/10.4236/ica.2012.32014 Published Online May 2012 (http://www.SciRP.org/journal/ica)

Distributed Air & Missile Defense with Spatial Grasp Technology

Peter Simon Sapaty
Institute of Mathematical Machines and Systems, National Academy of Sciences, Kiev, Ukraine
Email: peter.sapaty@gmail.com

Received April 18, 2012; revised May 16, 2012; accepted May 23, 2012

ABSTRACT

A high-level technology is revealed that can effectively convert any distributed system into a globally programmable machine capable of operating without central resources and self-recovering from indiscriminate damages. Integral mission scenarios in Distributed Scenario Language (DSL) can be injected from any point, runtime covering & grasping the whole system or its parts, setting operational infrastructures, and orienting local and global behavior in the way needed. Many operational scenarios can be simultaneously injected into this spatial machine from different points, cooperating or competing over the shared distributed knowledge as overlapping fields of solutions. Distributed DSL interpreter organization and benefits of using this technology for integrated air and missile defense are discussed along with programming examples in this and other fields.

Keywords: Distributed Systems; Spatial Grasp Technology; Integral Mission Scenarios; Distributed Scenario Language; Parallel Networked Interpretation; Air and Missile Defense; Directed Energy Systems; Cooperative Robotics

1. Introduction

1.1. Air & Missile Defense as Large Distributed Systems

Air and missile defense capabilities are growing globally and at a fast rate [1,2]. They are supported by novel technologies for detection, tracking, interception and destruction of attacking missiles. These systems are usually distributed on large territories, consist of many interacting elements (from sensors to shooters, see some related snapshots in **Figure 1**), and are expected to work in complex conditions to effectively protect national and international infrastructures and withstand unpredictable events.

1.2. Traditional Path in System Development

Originally a new system or campaign idea (related to air & missile defense incl.) emerges in a very general, integral form, as shown symbolically in **Figure 2(a)**. Then it is mentally decomposed into parts, each subsequently detailed, extended, and clarified, as in **Figure 2(b)**. Next step is materialization of the clarified parts and their distribution in physical or virtual spaces. To make these parts work together as a whole within the original idea, a good deal of their communication, synchronization, and sophisticated command and control are usually required, as shown in **Figure 2(c)**.

For a military area, **Figure 2(a)** may correspond to the general idea of winning a battle over an adversary or defending a critical infrastructure; **Figure 2(b)** additionally clarifies technical and human resources needed for this; and **Figure 2(c)** depicts how these resources should be organized together within a workable system (command and control including) fulfilling the global objectives.

The original idea, **Figure 2(a)**, and even its logically partitioned stage, **Figure 2(b)**, usually remain in the minds of creators and planners only (possibly also being verbally or graphically recorded in an informal manner), whereas actual system formalization and implementation begin from the already partitioned, distributed and interlinked stage, **Figure 2(c)**. So in reality we have mostly bottom up, parts-to-whole strategy in actual system design, in hope that the system developed will be ultimately capable of performing the initially formulated global task, *i.e.* of **Figure 2(a)**.

1.3. Existing System Design & Implementation Problems

- Within the philosophy mentioned above it may be difficult to put the resultant distributed system with many interacting parts into compliance with the initial

Figure 1. Some snapshots of air & missile defense systems.

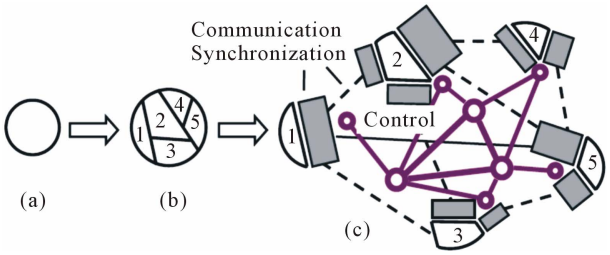


Figure 2. Traditional approach in system design and management: (a) original idea; (b) breaking into pieces; (c) system formalization, distribution, and implementation.

idea.

- The resultant system may have side effects, including unwanted ones, like unpredictable behaviors.
- The resultant solution may be predominantly static, *i.e.* if the initial idea changes, the whole system may have to be partially or even completely redesigned and reassembled.
- Adjusting the already existing multi-component system designed for one idea to an essentially new one may result in a considerable loss of the system's integrity and performance.

1.4. The Alternative Approach Offered

In this paper, we propose formalization of the initial stage a of **Figure 2** (and if needed, stage b) in a way that can be easily updated or even fully changed, with shifting most of stage b and completely stage c to an automated up to fully automatic implementation (incl. effective robotization). This can result in high flexibility, productivity, and self-recoverability from damages in conducting advanced campaigns, military ones including, where local and global goals as well as environments can change at runtime.

The developed (prototyped and tested in different countries) Spatial Grasp Technology (SGT) and its underlying Distributed Scenario Language (DSL) with details of their distributed implementation in networked systems are briefed in this paper along with application examples related to distributed air and missile defense (for existing basic publications on this paradigm see also [3-6]).

2. Grasping Solutions with Spatial Waves

The model described here reflects higher-level, holistic, gestalt-like vision and comprehension of distributed systems by human brain in the form of parallel mental waves covering and grasping the space [7-9] (**Figure 3(a)**) rather than traditional collection and interaction of parts or agents [10] on which most of existing software systems are based.

The original system idea of **Figure 2(a)** is represented in an integral non-atomistic but at the same time fully formal way, reflecting how a human commander mentally observes the space where a problem is to be solved. Traditional atomism emerges only during interpretation of this formally represented idea (which may be automatic, and only when really required). This allows us to get flexible and easily changeable formal definition of systems and operations in them while omitting traditional numerous organizational details, such as in **Figure 2(c)**, effectively concentrating on global goals and behaviors instead.

Materialization of this approach is carried out by the network of universal intelligent modules (U) embedded into important system points, which collectively interpret integral mission scenarios expressed in the waves formalism (starting from any point and covering the distributed system at runtime, as in **Figure 3(b)**). Different scenarios can start from the same or different points, and can cooperate or compete in the networked space as overlapping fields of solutions.

The compact spreading scenarios, which can be created and modified on the fly (being up to a hundred times shorter than, say, in Java) are forming dynamic knowledge infrastructures arbitrarily distributed between system

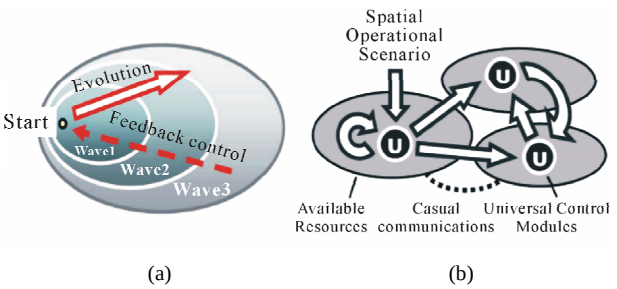


Figure 3. The waves paradigm: (a) controlled grasping of distributed worlds with spatial waves; (b) self-evolving high-level wave-like mission scenarios in distributed networked environment.

components (humans, robots, sensors). Navigated by same or other scenarios, they can effectively support distributed databases, advanced command and control, also provide overall situation awareness and autonomous decisions.

3. Distributed Scenario Language

DSL is quite different from traditional programming languages. Rather than describing data processing in a computer memory, as usual, it allows us to directly move through, observe, and make any actions in fully distributed environments (whether physical or virtual).

3.1. The Worlds DSL Operates with

DSL directly operates with:

- Virtual World (VW), which is finite and discrete, consisting of nodes and semantic links between them.
- Physical World (PW), an infinite and continuous, where each point can be identified with physical coordinates (with a certain precision).
- Virtual-Physical World (VPW), being finite and discrete similar to VW, but associating some or all virtual nodes with PW coordinates.

3.2. Main DSL Features

Other DSL features can be summarized as follows:

- A scenario expressed in it develops as a transition between sets of progress points (or props) in the form of parallel waves.
- Starting from a prop, an action may result in one or more new props.
- Each prop has a resulting value (which can be multiple) and resulting state, being one of the four: thru (full success allowing us to proceed further from this point), done (success with termination of the activity in this point), fail (regular failure with local termination), and abort (emergency failure, terminating the whole distributed process, associated with other points too).
- Different actions may evolve independently or interdependently from the same prop, contributing to (and forming altogether) the resultant set of props.
- Actions may also spatially succeed each other, with new ones applied in parallel from props reached by the preceding actions.
- Elementary operations can directly use local or remote values of props obtained from other actions (the whole scenarios including), resulting in value(s) of prop(s) produced by these operations.
- These resultant values can be used as operands by other operations in an expression or by the next operations in a sequence (the latter can be multiple, if processes split). These values can also be directly assigned to local or remote variables (for the latter case, an access to these variables may invoke scenarios of any complexity).
- Any prop can associate with a node in VW or a position in PW, or both (when dealing with VPW); it can also refer to both worlds separately and independently.
- Any number of props can be simultaneously associated with the same points of the worlds (physical, virtual, or combined).
- Staying with the world points, it is possible to directly access and update local data in them.
- Moving in physical, virtual or combined worlds, with their possible modification or even creation from scratch, are as routine operations as, say, arithmetic, logical, or control flow of traditional programming languages.
- DSL can also be used as a universal programming language (similar to C, Java or FORTRAN).

3.3. DSL Syntax and Main Constructs

DSL has recursive syntax, represented on top level as in **Figure 4** (programs are called grasps, reflecting their main semantics as gasping and integrating distributed resources into goal-driven systems).

The basic construct rule can represent any definition, action or decision, for example:

- elementary arithmetic, string or logic operation;
- hop in a physical, virtual, or combined space;
- hierarchical fusion and return of (remote) data;
- distributed control, both sequential and parallel;
- a variety of special contexts for navigation in space, influencing operations and decisions;
- type or sense of a value, or its chosen usage, guiding automatic interpretation.

3.4. DSL Spatial Variables

There are different types of variables in DSL:

- Heritable variables—these are starting in a prop and serving all subsequent props, which can share them in both read & write operations.

$$
\begin{aligned}
&\textit{grasp} && \rightarrow \ \textit{phenomenon} \mid \textit{rule}(\{\textit{grasp},\}) \\
&\textit{phenomenon} && \rightarrow \ \textit{constant} \mid \textit{variable} \mid \textit{special} \\
&\textit{constant} && \rightarrow \ \textit{information} \mid \textit{matter} \mid \textit{combined} \\
&\textit{variable} && \rightarrow \ \textit{heritable} \mid \textit{frontal} \mid \textit{environmental} \mid \\
&&& \quad \textit{nodal} \\
&\textit{rule} && \rightarrow \ \textit{movement} \mid \textit{creation} \mid \textit{elimination} \mid \\
&&& \quad \textit{echoing} \mid \textit{fusion} \mid \textit{verification} \mid \\
&&& \quad \textit{assignment} \mid \textit{advancing} \mid \textit{branching} \mid \\
&&& \quad \textit{transference} \mid \textit{timing} \mid \textit{granting}
\end{aligned}
$$

Figure 4. DSL recursive syntax and main constructs.

- Frontal variables—are an individual and exclusive prop's property (not shared with other props), being transferred between the consecutive props, and replicated if from a single prop a number of props emerge.
- Environmental variables—are accessing different elements of physical and virtual words when navigating them, also a variety of parameters of the internal world of DSL interpreter.
- Nodal variables—allow us to attach an individual temporary property to VW and VPW nodes, accessed and shared by props associated with these nodes.

These variables, especially when used together, allow us to create efficient spatial algorithms not associated with particular processing resources, working in between components of distributed systems rather than in them. These algorithms can also freely move in distributed processing environment (partially or as a whole), always preserving integrity and overall control.

DSL also permits the use of traditional operational symbols and delimiters, to simplify and shorten programs, if this proves useful.

4. Distributed DSL Interpreter

4.1. Structure of the Interpreter

The DSL interpreter [4-6] (see **Figure 5**) has the following key features:

- It consists of a number of specialized modules working in parallel and handling and sharing specific data structures supporting persistent virtual worlds and temporary hierarchical control mechanisms.
- The whole network of the interpreters can be mobile and open, changing at runtime the number of nodes and communication structure between them.
- Copies of the interpreter can be concealed, as for acting in hostile systems, allowing us to impact the latter overwhelmingly (finding & eliminating unwanted infrastructures including).

4.2. Distributed Track System

- The heart of the distributed interpreter is its spatial track system (**Figure 6**) with its parts kept in the Track Forest memory of local interpreters; these being logically interlinked with such parts in other interpreter copies, forming altogether indivisible space coverage.
- This enables hierarchical command and control and remote data and code access, with high integrity of emerging parallel and distributed solutions, without any centralized resources.
- The dynamically crated track trees spanning the systems in which DSL scenarios evolve are used for supporting spatial variables and echoing and merging different types of control states and remote data, being self-optimized in the echo processes.
- They also route further waves to the positions in physical, virtual or combined spaces reached by the previous waves, uniting them with the frontal variables left there by preceding waves.

4.3. DSL Interpreter as a Universal Spatial Machine

The (dynamically) networked DSL interpreters (**Figure 7**) are effectively forming parallel spatial machine ("machine" rather than computer as it operates with physical

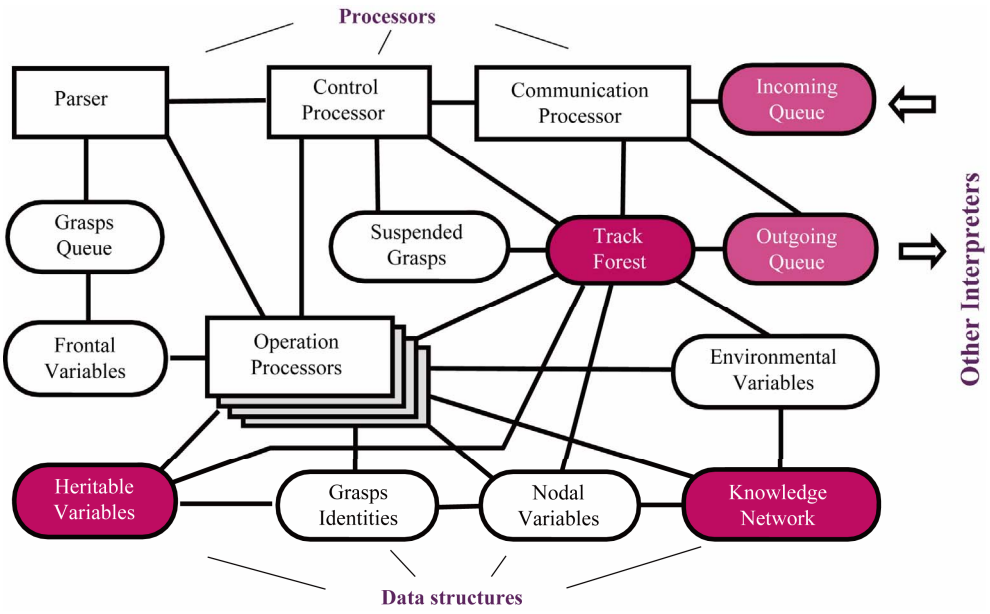


Figure 5. Organization of DSL interpreter.

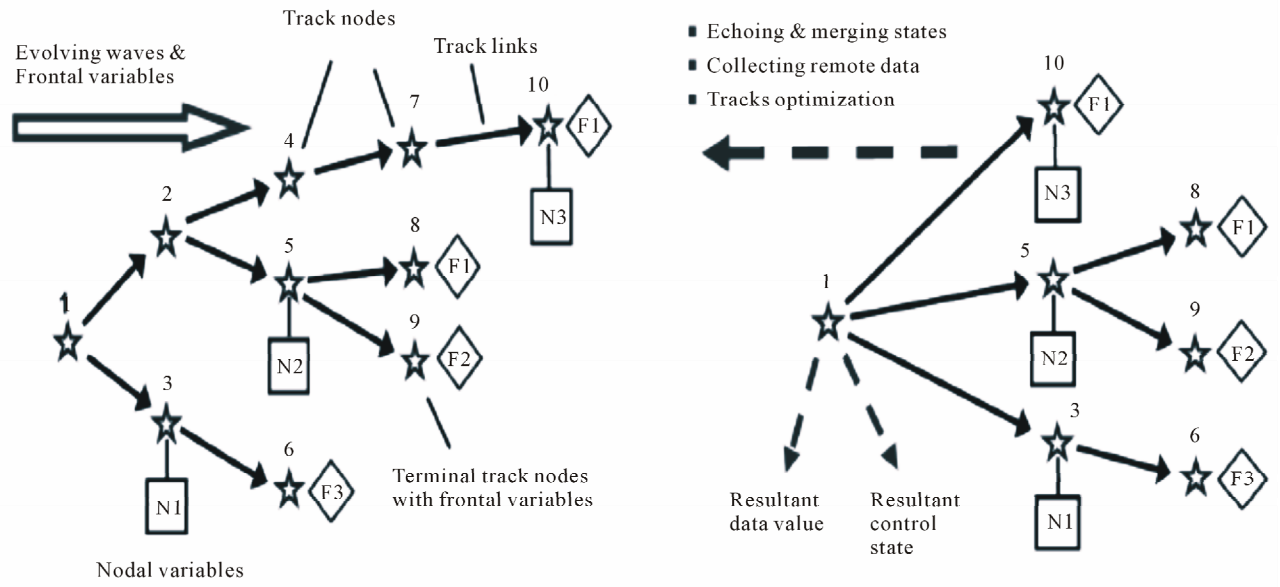


Figure 6. Distributed track system: (a) forward operations; (b) backward operations with tracks optimization.

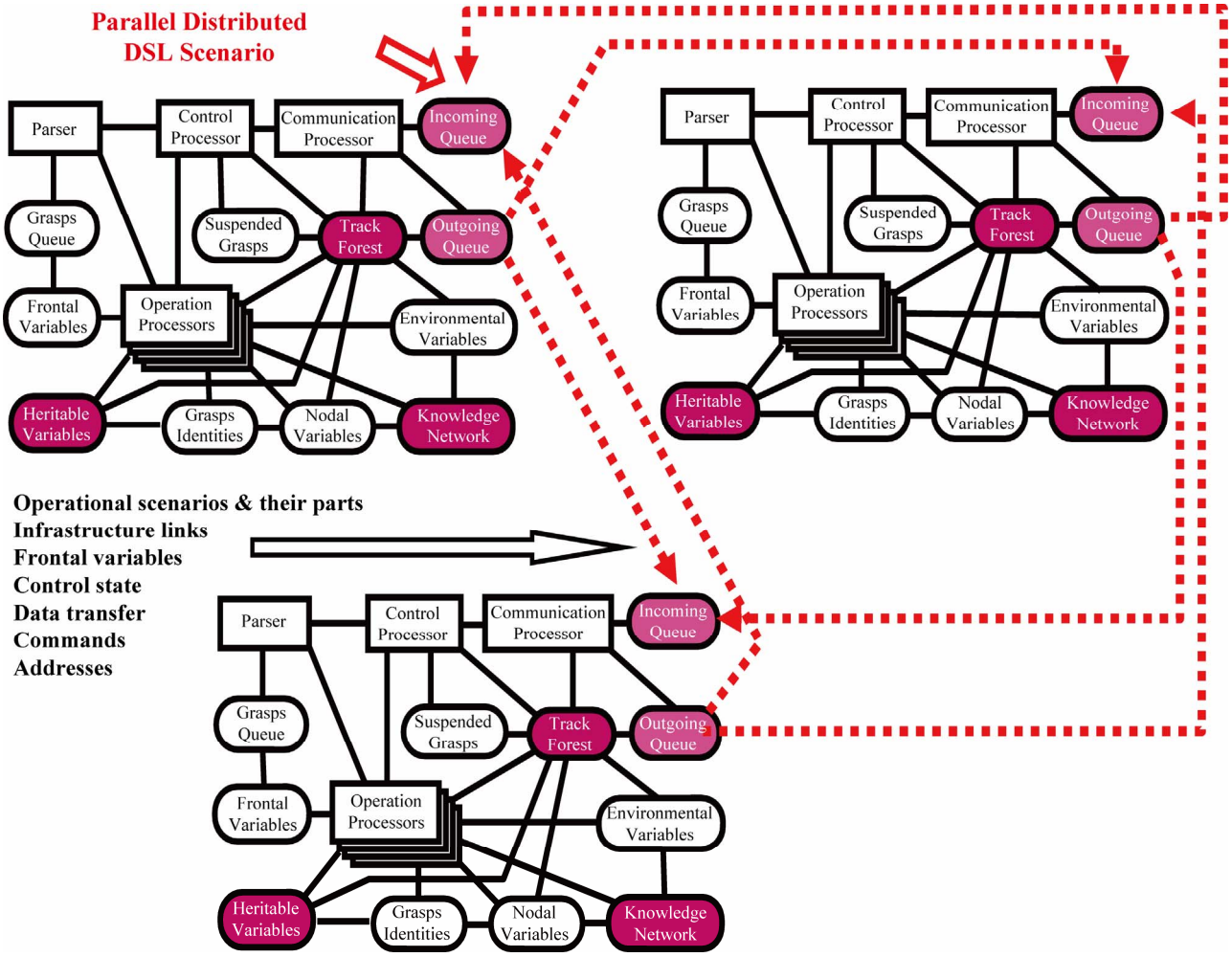


Figure 7. DSL interpretation network as a universal parallel spatial machine.

matter too, and can move partially or as a whole in physical space) capable of solving any problems in a fully distributed mode, without any special central resources.

5. Elementary Programming Examples

We will show here elementary examples of solution in DSL of some important problems on distributed structures in a parallel and fully distributed way, where each node may reside in (or associate with) a different computer. These tasks may well relate to the general orientation of this paper on air and missile defense (see also [4, 5]).

5.1. Finding Shortest Path in Parallel

The solution for finding shortest path between two nodes (let them be a and e) can be expressed by DSL scenario that follows.

```
frontal (Far, Path);
sequence (
  (hop ('a'); Distance = 0;
    repeat (hop(alllinks); Far += LINK;
      or (Distance == nil, Distance > Far);
      Distance = Far; Before = BACK)),
  (hop ('e');
repeat (Path = NAME & Path; hop(Before));
output (Path)))
```

The result obtained in node a for the network in **Figure 8** will be (a, b, d, e). It has been found by navigating the network of weighed links in parallel and fully distributed mode, without any central resources.

Many important problems of optimization and control (from battlefields to infrastructure protection) may be expressed as finding shortest paths in distributed spaces. SGT, on the example of this task, can serve as a higher level universal communication protocol [11] capable of organizing any communication, and especially if other means fail during and after indiscriminate damages to infrastructures.

5.2. Analyzing Distributed Structures

Another important problems in distributed systems may be finding weak (or weakest) and strong (strongest) parts in them, whether these are civil or military organizations (say, battlefields in the latter case), and friendly or of adversaries. In the examples below we formulate these two problems on general graphs where any node may be with a different computer (**Figure 9**).

To find the weakest nodes in a graph, like articulation points, see **Figure 9(a)**, which when removed split it into disjoint parts, the following program suffices (resulting in node d).

```
hop (allnodes); IDENTITY = NAME; mark;
```

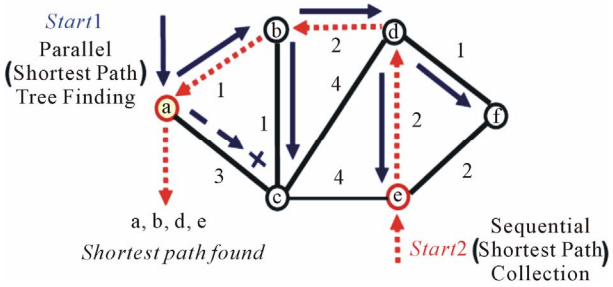
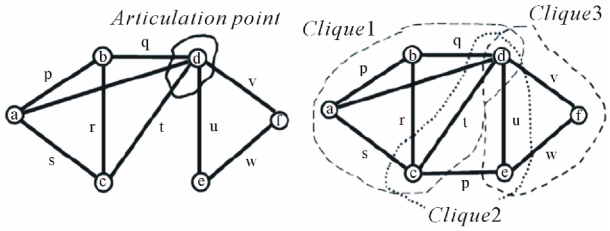


Figure 8. Finding shortest path in parallel distributed mode.



(a) (b)

Figure 9. Solving topological problems: (a) discovering articulation points; (b) finding cliques.

```
and ((hop(random, alllinks);
        repeat (unmarked; mark; hop(alllinks))),
      (hop (alllinks); unmarked),
    output (NAME))
```

Cliques (or maximum fully connected sub-graphs of a graph), on the contrary, may be considered as strongest parts of a system. They all can be found in parallel by the following simple program resulting for **Figure 9(b)** in: (a, b, c, d), (c, d, e), (d, e, f):

```
hop (allnodes); Fclique = CONTENT;
repeat (
  hop (alllinks); notbelong(CONTENT, Fclique);
  and (andparallel(hop(anylink, Fclique)!,
       or (BACK > NAME!, Fclique & NAME)));
  output (Fclique)
```

6. Collective Robotics Examples in DSL

Installing DSL interpreter into mobile robots (ground, aerial, surface, underwater, space, etc., as in **Figure 10** for the first two) allows us to organize effective group solutions (incl. any swarming) of complex problems in distributed physical spaces in a clear and concise way, shifting traditional management routines to automatic levels.

We will consider two levels: organizing robotic swarms on top semantic level where only global task is formulated (like in **Figure 2(a)**, and all internal system organization is fully delegated to the distributed DSL interpreter), and also expressing some sort of explicit collective behavior (corresponding to **Figure 2(b)** and partially

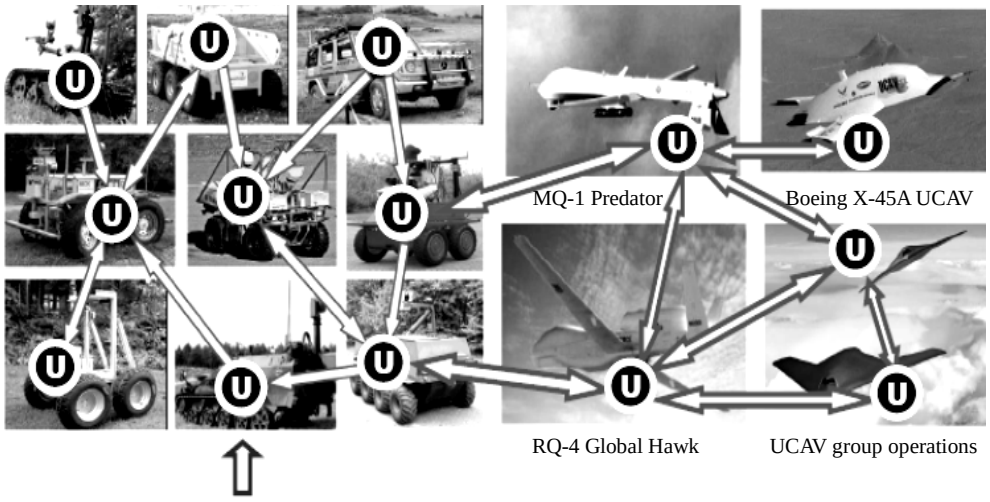


Figure 10. Integration of ground and aerial robots in SGT.

Figure 2(c), with the rest of organization of **Figure 2(c)** delegated to automation).

6.1. Semantic, Task Level

For this case, a group of mobile robots can be tasked at a highest possible level, just telling what they should do together but without detailing how, and what are the duties of every unit, which may not be known in advance. An exemplary task:

Go to physical locations of the disaster zone with coordinates (50.433, 30.633), (50.417, 30.490), and (50.467, 30.517). Evaluate damage in each location, find and transmit the maximum destruction value, together with exact coordinates of the corresponding location, to a management center.

The DSL program will be as follows:

```
transmit (max(
   move ((50.433, 30.633),
            (50.417, 30.490),
            (50.467, 30.517));
   evaluate (destruction) & WHERE))
```

Details of automatic implementation of this scenario by different and possibly runtime varying numbers of mobile robots are discussed elsewhere [12,13].

6.2. Explicit Behavior Level

After embedding DSL interpreters into robotic vehicles, we can also provide any needed detailed collective behavior of them (at a lower than top task level, as before)—from loose swarms to a strictly controlled integral unit obeying external orders. Any mixture of different behaviors within the same scenario can be easily programmed too. Expressing different simple scenarios in DSL and their integration into a more complex combined one may be as follows.

- Swarm movement scenario, starting from any unit (**swarm_move**):

```
hop (allnodes);
Limits = (dx(0,8), dy(-2,5)); Range = 500;
repeat (Shift = random(Limits);
if (empty(hop(Shift, Range), move(Shift)))
```

- Finding topologically central unit and hopping into it, starting from any unit (**find_hop_center**):

```
frontal (Avr)=average(hop(allnodes); WHERE);
hop (min(hop(allnodes);
        distance (Avr, WHERE) & ADDRESS) : 2)
```

- Creating runtime infrastructure, starting from the central unit found (**infra_build**):

```
stay (repeat(linkup(+infra, first, Depth)))
```

- Targets collection & distribution & impact, starting from the central unit found (**collect_distribute_impact**):

```
loop (nonempty(frontal(Seen) =
 repeat (detect(targets), hop(+infra)));
 repeat (
  select_move_shoot(Seen),hop(+infra)))
```

- Removing previous infrastructures (for subsequently creating a new one), starting from any unit (**infra_remove**):

```
stay (hop(allnodes); remove(alllinks))
```

Resultant combined solution (integration previous DSL programs named in bold), starting from any unit:

```
parallel (
   swarm_move,
   repeat (find_hop_center;
            infra_remove; infra_build;
            orparallel(
               collect_distribute_impact,
               sleep (delay))))
```


The obtained resultant scenario combines loose, random-oriented swarm movement in a distributed space with periodic finding and updating topologically central unit, and setting-updating runtime hierarchical infrastructure between the units. The latter controls observation of distributed territory, collecting potential targets, distributing them back to the vehicles, and then selecting and impacting potential targets by them individually (a related snapshot, say, for aerial vehicles, is shown in **Figure 11**). More on this integral scenario can be found in [14].

7. Air & Missile Defense with SGT

We will be considering here different scenarios related to distributed integrated air and missile defense and their expression in DSL.

7.1. Distributed Tracking of Moving Objects

In a vast distributed environment, each embedded (or moving) sensor can usually observe only a limited part of space, so to keep the whole observation continuous and integral, each noticed mobile object should be handed over between neighboring sensors during its movement, along with the data accumulated on it (as in **Figure 12**).

The following program, starting in all sensors, catches the object it sees (splitting itself if more than one) and follows it wherever it goes, if not observable from the current point any more (via the virtual networked space, activating the whole region around in parallel to define the next tracing move matching the object's physical move).

```
frontal (Object, Threshold = visibility);
hop (all_nodes);
split (search_colect(aerial, Threshold));
Object = VALUE;
repeat (
   cycle (visibility(Object) > Threshhold);
   max_destination (
  hop (all_neighbors); visibility(Object)))
```

By this mobile intelligence techniques, each discovered target (aerial, ground, space, etc.) can always be kept in view individually, in parallel with other ones, its behavior can gradually analyzed and accumulated, and optimal (possibly, scarce and scattered) impact facilities activated, if needed. (More on this task, say, in [15].)

7.2. Directed Energy Systems

Directed energy systems and weapons (DEW) are of rapidly growing importance in many areas and especially in critical infrastructure protection, at advanced battlefields (as shown in **Figure 13**) and, of course, for advanced air and missile defense, as potential capabilities

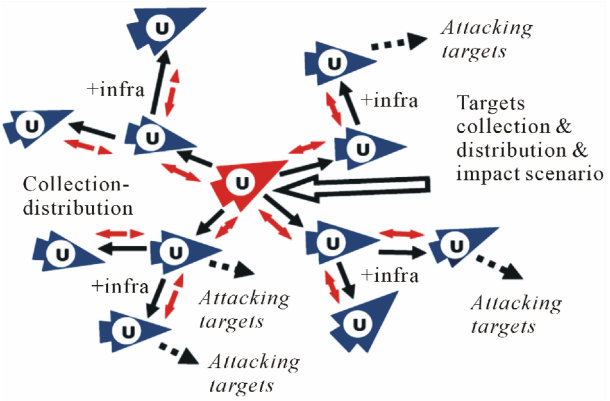


Figure 11. Collecting, disseminating, and attacking targets by an unmanned aerial team using dynamically created and updated C2 infrastructure, while moving altogether as a loose swarm.

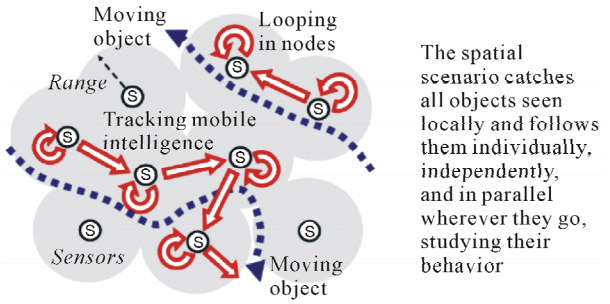


Figure 12. Tracking moving objects by mobile intelligence.

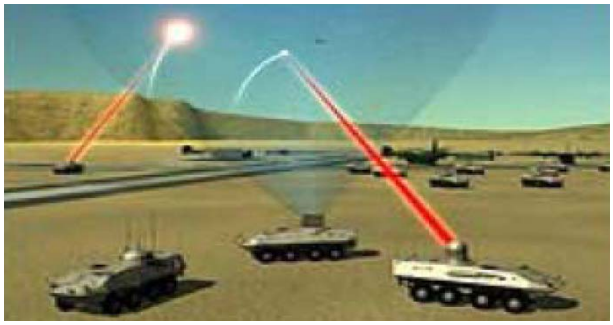
for shooting down unwanted aerial and space objects with DEW are beyond comparison with other means, both existing and being developed.

With hardware equipment operating with the speed of light, traditional manned C2 may become a bottleneck for these advanced technical capabilities, especially in crisis events. With the SGT technology installed, we may organize any runtime (even on the fly) C2 infrastructures operating automatically, with the "speed of light" too, fitting hardware capabilities and possibly even excluding humans from the loop in time critical situations.

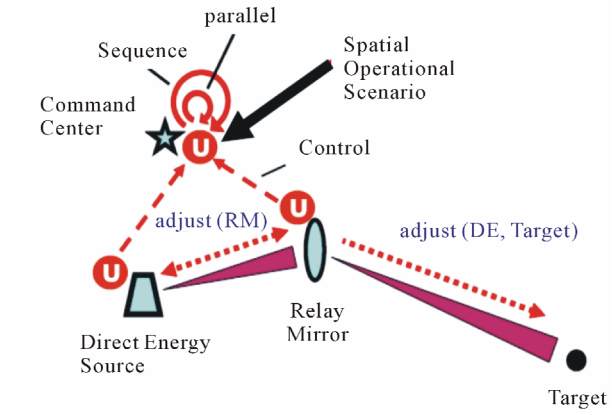
The following is an example of setting an automatic runtime C2 in a system with direct energy (DE) source, relay mirror (RM), and the Target discovered, with an operational snapshot shown in **Figure 13(b)**.

```
frontal (DE = coordinates1; RM = coordintes2;
              Target = coordinates3);
sequence (
   parallel ((hop(DE); adjust(RM)),
                  (hop (RM); adjust(DE, Target))),
   (hop (DE); activate))
```

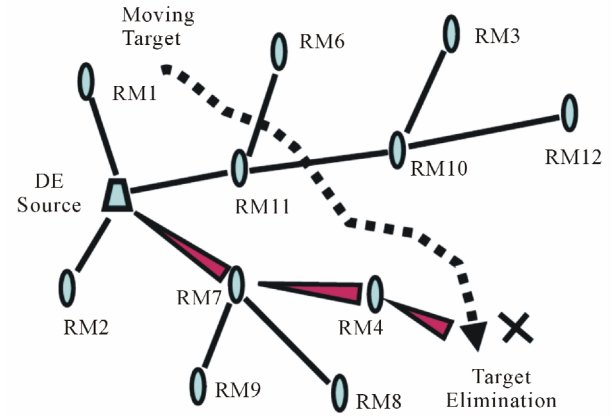
There also exist advanced projects of global dominance with transference of directed energy, like the Boeing's Advanced Relay Mirror System (ARMS) concept. It plans to entail a constellation of as many as two dozen



(a)



(b)



(c)

Figure 13. DEW on an advanced battlespace: (a) Operational picture; (b) DE-RM-target runtime control; (c) DE delivery via network of relay mirrors.

orbiting mirrors that would allow 24/7 coverage of every corner of the globe. When activated, this would enable a directed energy response to critical trouble spots anywhere.

We can use the distributed shortest path solution shown in Section 5.1 for providing a runtime path in a worldwide distributed dynamic set of relay mirrors (as some of which may themselves happen to be on the move or out of order)—between DE source and the destination needed. This will provide optimal directed energy transfer, as shown in **Figure 13(c)**, see also [16,17].

Embedding DSL interpreter into both DEW facilities and conventional force units (as in **Figure 14**), we can effectively integrate rapidly developing DEW into the force mix, which may also include multiple unmanned vehicles, thus obtaining advanced rapid reaction forces for most diverse applications, air and missile defense including [17].

7.3. Global Awareness and Parallel Impact of Targets

In **Figure 15** (see also [18]), a possible conflict situation is shown on a supposedly large territory, where global awareness and coordinated actions may be crucial to withstand it. Having installed DSL interpreter in different units (both manned and unmanned) it will become possible to coordinate and manage the global reaction needed.

Similar to the solutions in Section 6.2 for explicit robotic swarm behavior, we can launch global awareness, collection and dissemination of targets throughout the whole territory and their impact by available distributed resources from any component with the interpreter installed in it, as in **Figure 15**. The self-navigating and self-replicating DSL scenario, allowing us to cover the whole system at runtime and set up its needed behavior, is extremely simple (can be created and launched ahead

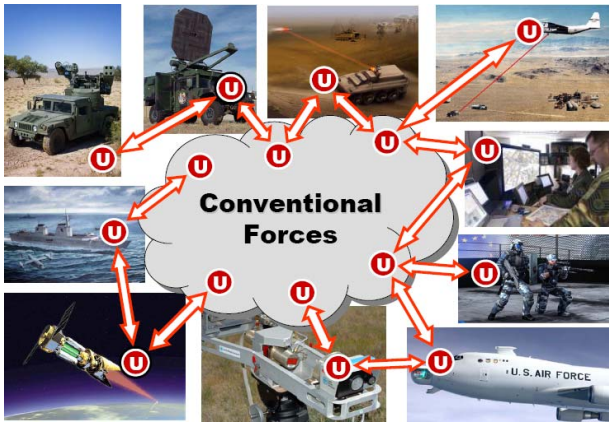


Figure 14. Integration of DEW with conventional forces.

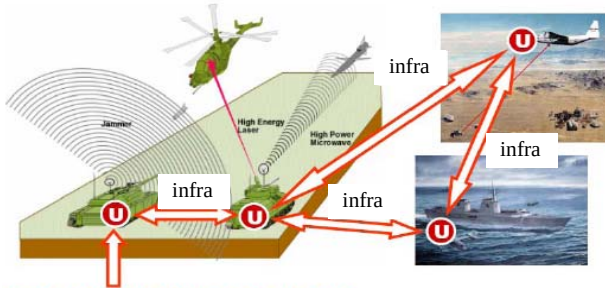


Figure 15. Distributed targets collection and dissemination.

or even during the conflict):

```
loop (
   nonempty (frontal(Targets) =
      repeat (discover(local), hop(infra)));
repeat (select_attack (Targets),
                hop (infra)))
```

No central resources (C2 including) are needed for this, which may be particularly vulnerable in crisis-prone and asymmetric situations, especially those related to infrastructure protection, battlefield management, and air & missile defense.

7.4. Europe-Related Missile Defense Scenarios

Let us consider here some scenarios relevant to the currently being discussed European missile defense plans, widely available [19] and copied in **Figure 16**.

The missile defense systems are supposed to work in the following stages:

1) a): Infrared satellite system picks up heat signatures of hostile missiles launched towards target. b): Information transmitted to ground stations for processing. 3: Processed information sent to C2 network;

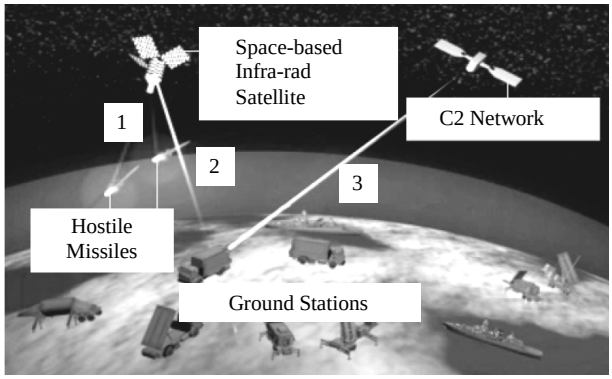
2) The C2 network relays information to sensor and weapons systems in the region;

3) a): Long-range sensors continue to track the missile to help command system calculate options for destroying them. b): Information is constantly shared among the sensors and weapons systems;

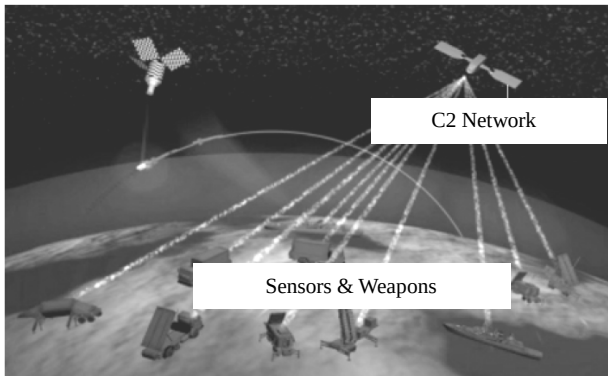
4) Command system has the option of shooting down the hostile missiles while in the upper or lower layers of the atmosphere.

Having extended these with advanced capabilities like DEW (high power lasers) located in space or on airborne (manned or UAV) platforms (synchronized with infrared satellite sensors and also capable of using relay mirrors, as in **Figure 13(c)**), we can write the following very simple DSL scenario integrating infrared satellites, DEW facilities, long range sensors and upper and lower layer shooters into a dynamic distributed system capable of discovering hostile objects, tracing them at different stages of flight, and (re)launching target impact facilities with verification of their success or failure, until the targets are destroyed.

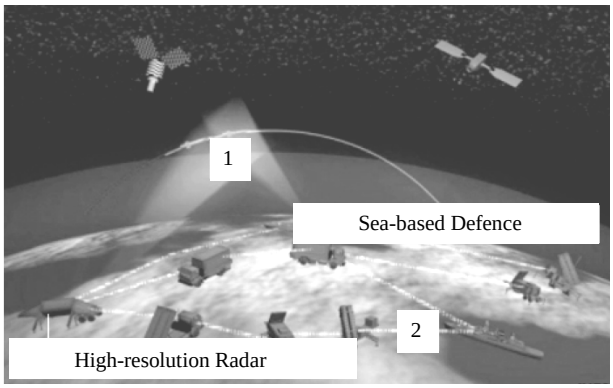
```
hop (infrared_satellite_sensors);
loop (
  nonempty (New = infrared(new_targets));
  release (
    split (New); frontal(Target) = VALUE;
cycle (
  visible (Target); update(Target); hop(DE);
     if (try_shoot_verify(Target), done));
   hop (long_range_sensors);
cycle (
```



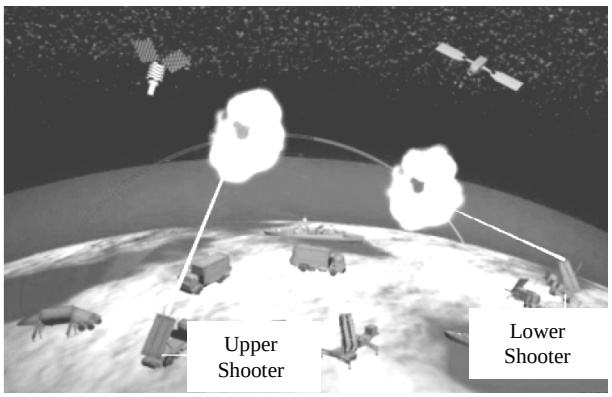
(a)



(b)



(c)



(d)

Figure 16. Possible European missile defense scenarios.


```
visible (Target); update(Target);
  if (distsance(Target) > threshold,
      hop (upper_layer_shooters),
      hop (lower_layer_shooters))
  if (try_shoot_verify(Target), done))));
```

The advantages of this program are that it can be initially applied to any available system component, automatically creating distributed command and control infrastructure particularly oriented on the currently discovered targets and dynamic situations. This automatically created distributed system organization can also self-recover at runtime after indiscriminate damages to any system components mentioned above (due to fully interpreted, mobile, virus-like implementation of DSL in distributed networked spaces).

Any other centralized or distributed scenarios, with different levels of detailing (like the one of "launch on remote concept" [1] depicted in **Figure 17**) can also be effectively described in DSL, as experimental programming shows. (The figure shows transmission of tracking information to the interceptor's flight computer and launching the interceptor earlier and farther downrange than the ship's own radar would allow).

8. Other Missile Defense-Related Tasks

We will mention here some other known in the past projects related to missile defense, which are currently under theoretical investigation for a possible use of SGT for their management and simulation (if similar ones happen to emerge in the future).

8.1. Brilliant Pebbles

Brilliant Pebbles [20], the top anti-missile program of the Reagan and the first Bush administrations, was an attempt to deploy a 4000-satellite constellation in low-Earth orbit that would fire high-velocity, watermelon-sized projectiles at long-range ballistic missiles launched from anywhere in the world. Although the program was eliminated by the Clinton Administration, the concept of Brilliant Pebbles remains among the most effective means of ballistic missile defense. Massively used distributed projectiles with DSL interpreter installed in each of them would be an almost ideal test bed for the virus-like distributed implementation of SGT, which could easily form and control goal-directed self-organized distributed swarms effectively attacking both individual and collective (incl. other swarm) targets, without any centralized facilities.

8.2. Multiple Kill Vehicles

The Multiple Kill Vehicle (MKV) [21] was a planned missile defense program whose goal was to design, develop, and deploy multiple small kinetic-energy-based warheads that can intercept and destroy multiple ballistic missiles, including possible decoy targets (the project was canceled, same as the previous one, but its possible rebirth in the future not excluded too). The MKV concept provided the capability for more than one kill vehicle to be launched from a single booster. With multiple kill vehicles on a single target "cloud" the probability for a hit on the actual warhead is enhanced. The capability of the system to intercept multiple independent targets was also planned to be tested. This, same as the previously mentioned Brilliant Pebbles project, would serve as a perfect test for the technology offered in this paper, especially for organizing collective behavior of multiple kill vehicles in highly dynamic and unpredictable situations.

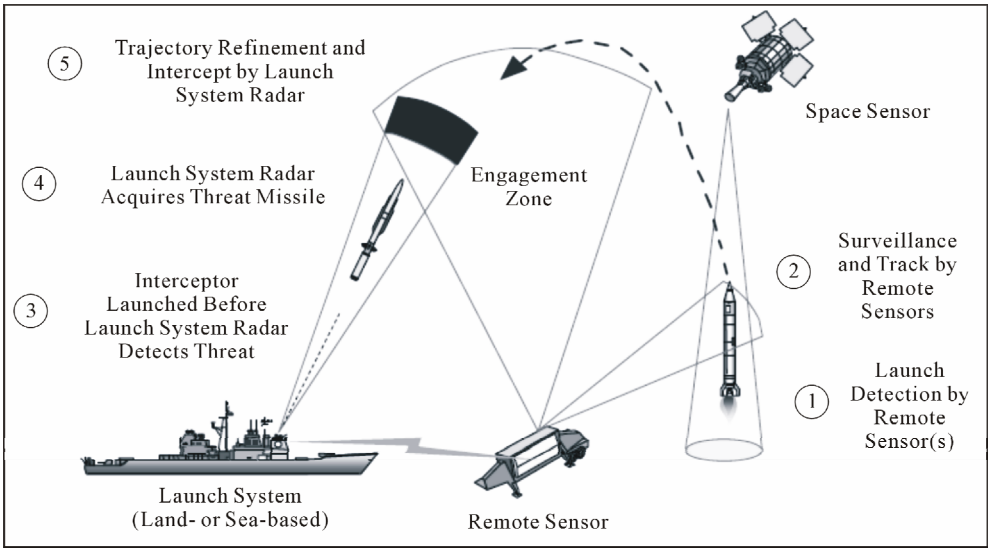


Figure 17. Launch on the remote concept.

8.3. Scenarios of Possible Nuclear Conflicts

To investigate the power and limits of applications of the technology offered, a number of hypothetic scenarios (far from all possible) of greater world conflicts have been programmed in DSL, like those in [22] (copied in **Figure 18** without further details as may be controversial and much fantasized for the current state of international relations).

A world nuclear war [22] may be the one that involves most or all nuclear powers releasing a large proportion of their nuclear weapons at targets in nuclear, and perhaps non-nuclear, states. Such a war could be initiated accidentally, aggressively or pre-emptively and could continue and spread through these means or by retaliation by a party attacked by nuclear weapons. Such a war could start through a reaction to terrorist attacks, or through the need to protect against overwhelming military opposition, or through the use of small battlefield tactical nuclear weapons meant to destroy hardened targets.

The simulation in DSL shows that highly organized distributed systems with global consciousness and will, which can be effectively provided by SGT (with the whole countries behaving as an integral brain, possibly even unmanned in time critical situations), could believably prevent and avoid such conflicts in real time.

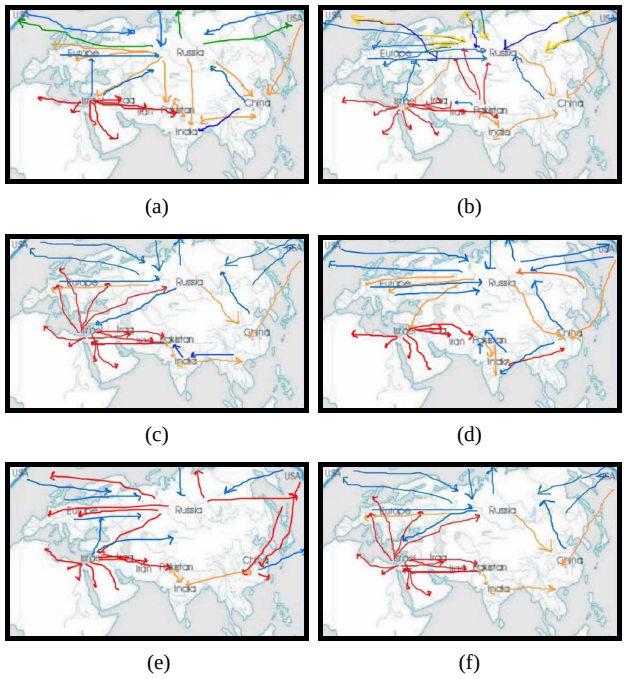


Figure 18. Examples of scenarios of possible global nuclear conflicts started by: (a) mistake; (b) threat; (c) retaliation; (d) nuclear exchange; (e) invasion; (f) terrorism.

9. Other Researched Applications

9.1. Emergency Management

Using DSL interpreters installed in massively wearable devices may allow us to assemble workable systems from any wreckage after the disasters, using any remaining communication channels, manual including [23]. These emergent systems can provide distributed self-awareness, collect statistics of casualties, guide the delivery of relief goods, coordinate collective escape from the disaster zone, as well as cooperate with rescue teams.

9.2. Distributed Avionics

Implanting DSL interpreter copies into main control nodes of the aircraft may provide a higher, intelligent, layer of its self-analysis and self-recovery, by the spreading recursive scenarios starting from any point and collecting & fusing key data from other points [24]. The embedded interpretation network with local, dynamic, and emergent links will be fully functional under any damages, especially with wireless communications between the interpreters. This may always provide global control integrity, even in a physically disintegrating object, saving lives and completing missions.

9.3. Sensor Networks

Wireless sensors may be dropped from the air massively, as "smart dust". Having a limited communication range, they must operate in a network to do nonlocal jobs in a distributed environment. With the technology offered, we can convert their emergent networks into a universal parallel computer operating in DSL [25]. It can effectively solve complex distributed problems—from just collecting and fusing scattered data to outlining and assembling images of the distributed phenomena like, for example, flooding, smog, flocks of birds, movement of troops, etc., analyzing their behavior and tracking them as a whole.

9.4. Infrastructure Protection

Navigating the systems at runtime, the technology can analyze safety and integrity of critical infrastructures and key resources, establishing protective networked mechanisms throughout them [26]. Other systems can be involved from the SGT layer for emergent infrastructure protection and recovery. For example, in relation to energy infrastructures, the technology can help observe power networks from the air or ground, trace electric, gas, or oil supply lines, sensing their states (and, if needed, directly accessing the disaster zones), also providing regular or emergent sentry duties at power installations, etc.

9.5. Advanced Command and Control

In DSL it is possible to define high-level scenarios concentrating on mission goals and top decision-making while delegating C2 routines, appearing at runtime as a

derivative of the mission and environment states, to automatic interpretation. It is also convenient to express in DSL any theoretical and practical issues of advanced C2 explicitly. A variety of non-traditional C2 infrastructures, more flexible and diverse, had been considered in DSL [27].

Some of the mentioned above SGT and DSL researched application areas as well as other ones are shown in **Figure 19**, with additional references [28-38].

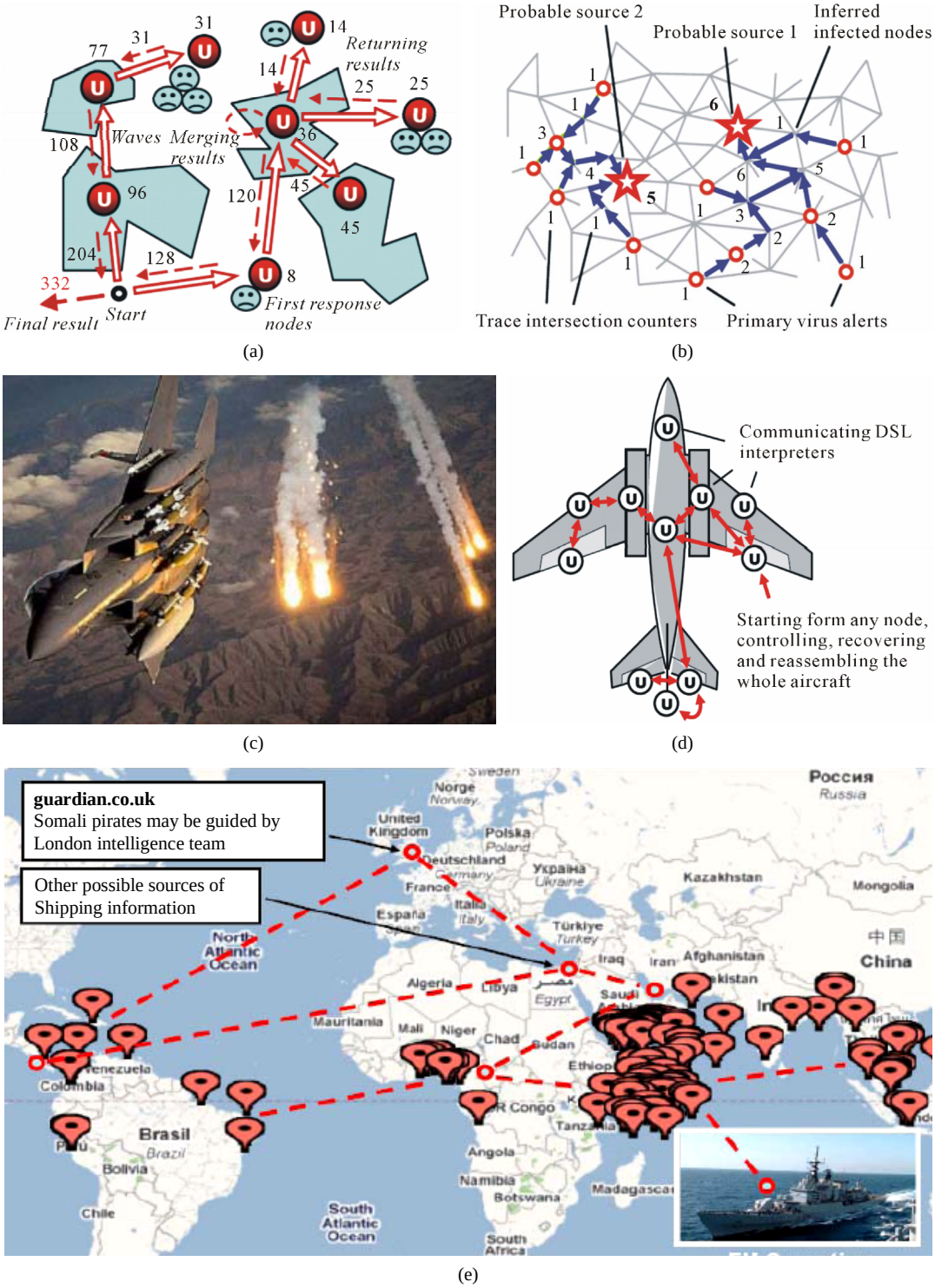


Figure 19. Some other SGT applications. (a) Crisis Management; (b) Cyber warfare; (c) Electronic warfare; (d) Military avionics; (e) Terrorism & piracy fight.

10. Some Historical Notes

The ideas described in this paper have not appeared from scratch, rather having a long development and evolution history. Stemming from electrical networks and their digital simulation (mid of sixties of the previous century), with spatial fields and waves forming basis and integrity of behavior of large distributed power systems, they were used in creation of first citywide heterogeneous computer networks from the end of sixties (before the internet). The program code (agent) mobility at that time was paramount for flexible and integral network management and control.

In seventies, a parallel networked macro-pipeline supercomputer was designed, implemented, industrially produced and used for modeling in aerodynamics, while containing traditional arithmetic processors and also intelligent ones with hardware emulation of high-level control programs working with analytical structures related to DSL in this paper.

Direct predecessor of DSL, called WAVE, was designed to work with spatial graphs where each node could reside on a separate computer, and complex network problems were effectively expressed as parallel wavelike navigation and matching of the network structures in fully distributed mode, without any central resources.

WAVE was first compiled into LISP, then to C, and afterwards effectively interpreted in C, with public domain WAVE interpreter used in different countries for intelligent network management, distributed knowledge bases, distributed virtual reality (WAVE was integrated with VRML), distributed simulation of dynamic systems like battlefields or road networks, also cooperative robotics.

DSL, comprising all WAVE features, represents a much higher level formalism capable of expressing world problems on a top semantic level, also inheriting gestalt philosophy principles (this was successfully reported at world gestalt congress).

A new patent is being prepared on distributed DSL interpreter, which can be placed on any platform within a few months by a small team of system programmers.

11. Conclusions

We have described a novel ideology and the supporting Spatial Grasp Technology (SGT) for high-level management of distributed dynamic systems that can be useful for advanced air and missile defense. SGT, among others, offers the following possibilities:

- Many targets can be simultaneously captured over the defended area and individually followed & studied by spreading mobile intelligence propagating in networked space (between limited range radars).
- SGT can analyze many moving targets in parallel and cooperatively, discovering, whether this is individual or swarm attack, and properly orienting the global system response.
- In case of multiple targets and limited physical resources, SGT can globally assess which targets are most important to shoot.
- Based on full interpretation of flexible mission scenarios (which can re-launch their parts or the whole) the distributed air & missile defense system can remain fully operational after any indiscriminate damages.
- SGT can operate in both live and simulation modes, with runtime simulation of evolving events serving as look-ahead facility for live control.
- SGT can take full responsibility for key decisions in most critical situations, excluding, if needed, humans from the control loop.

The ideology and technology developed can convert any distributed system into an integral dynamic brain which can quickly assess and withstand asymmetric situations and threats, protect critical infrastructures, win local and global conflicts, as well as avoid and terminate them at different stages of their development.

REFERENCES

[1] US Department of Defense, "Ballistic Missile Defense Review Report," 2010. http://www.defense.gov/bmdr/docs/BMDR%20as%20of%2026JAN10%200630_for%20web.pdf

[2] Royal Saudi Arabia Defense Forces, "International Symposium on Air Defense 2020+," 2011. http://www.isad2020.org.sa/english/online.php

[3] P. S. Sapaty, "Distributed Technology for Global Control, Lecture Notes in Electrical Engineering," Springer-Verlag, Berlin, 2009.

[4] P. Sapaty, "Ruling Distributed Dynamic Worlds," John Wiley & Sons, New York, 2005.

[5] P. Sapaty, "Mobile Processing in Distributed and Open Environments," John Wiley & Sons, New York, 1999.

[6] P. Sapaty, "A Distributed Processing System," European Patent No. 0389655, 1990.

[7] M. Wertheimer, "Gestalt Theory," Erlangen, Berlin, 1925.

[8] K. Wilber, "Ken Wilber Online: Waves, Streams, States, and Self—A Summary of My Psychological Model (or, Outline of an Integral Psychology)," Shambhala Publications, Boston, 2009.

[9] P. Sapaty, "Gestalt-Based Ideology and Technology for Spatial Control of Distributed Dynamic Systems", *International Gestalt Theory Congress*, 16*th Scientific Convention of the GTA*, Osnabrück, 26-29 March 2009, pp. 36-40.

[10] M. Minsky, "The Society of Mind," Simon and Schuster, New York, 1988.

[11] P. Sapaty, "High-Level Communication Protocol for Dynamically Networked Battlefields", *International Conference Tactical Communications*, London, 18-19 May 2009.

[12] P. Sapaty, "High-Level Technology to Manage Distributed Robotized Systems," Military Robotics, London, 2010.

[13] P. Sapaty, M. Sugisaka, K.-D. Kuhnert, "Spatial Scenarios for Distributed Unmanned Systems," *Proceedings AUVSI's Unmanned Systems North America*, Washington DC, 10-13 August 2009.

[14] P. S Sapaty, "Providing Spatial Integrity for Distributed Unmanned Systems," *Proceedings of* 6*th International Conference in Control, Automation and Robotics ICINCO*, Milan, 2-5 July 2009, pp. 29-40.

[15] P. S. Sapaty, "Meeting the World Challenges: From Philosophy to Information technology to Applications", *Proceedings of* 6*th International Conference in Control, Automation and Robotics ICINCO*, Milan, 2-5 July 2009, pp. IS31-IS43.

[16] P. Sapaty, A. Morozov and M. Sugisaka, "DEW in a Network Enabled Environment," *International Conference Directed Energy Weapons*, London, 28 February-1 March 2007.

[17] P. Sapaty, "High-Level Organization and Management of Directed Energy Systems," Directed Energy Weapons, London, 2010.

[18] P. Sapaty, "Tactical Communications in Advanced Systems for Asymmetric Operations," Tactical Communications, London, 2010.

[19] BBC New, "Russia to Work with Nato on Missile Defence Shield," 2010. http://www.bbc.co.uk/news/world-europe-11803931

[20] Heritage Foundation, "Brilliant Pebbles: The Revolutionary Idea for Strategic Defense," 1990. http://www.heritage.org/research/reports/1990/01/brilliant-pebbles-the-revolutionary-idea-for-strategic-defense

[21] Wikipedia Free Encyclopedia, "Multiple Kill Vehicle," http://en.wikipedia.org/wiki/Multiple_Kill_Vehicle

[22] C. Moore, "Six Escalation Scenarios Spiraling World to Nuclear War," http://www.carolmoore.net/nuclearwar/alternatescenarios.html

[23] P. Sapaty, M. Sugisaka, R. Finkelstein, J. Delgado-Frias and N. Mirenkov, "Advanced IT Support of Crisis Relief Missions," *Artificial Life and Robotics*, Vol. 11, No. 1, 2007, pp. 116-122. doi:10.1007/s10015-006-0412-x

[24] P. Sapaty, "Grasping the Whole by Spatial Intelligence: A Higher Level for Distributed Avionics," *Military Avionics*, London, 30 January-1 February 2008.

[25] P. Sapaty, "Intelligent Management of Distributed Sensor Networks," *Artificial Life and Robotics*, Vol. 11, No. 1, 2008, pp. 81-87. doi:10.1007/s10015-007-0446-8

[26] P. Sapaty, "Gestalt-Based Integrity of Distributed Networked Systems," *Unmanned/Unattended Sensors and Sensor Networks*, Berlin, 1 September 2009, pp. 63-70.

[27] P. Sapaty, A. Morozov, R. Finkelstein, M. Sugisaka and D. Lambert, "A New Concept of Flexible Organization for Distributed Robotized Systems," *Proceedings of Twelfth International Symposium on Artificial Life and Robotics*, Beppu, 25-27 January 2007, pp. 156-163.

[28] P. Sapaty and M. Sugisaka, "Countering Asymmetric Situations with Distributed Artificial Life and Robotics Approach," *Proceedings of Fifteenth International Symposium on Artificial Life and Robotics*, Beppu, 5-7 February 2010, pp. 356-362.

[29] P. Sapaty, "Distributed Capability for Battlespace Dominance," *Electronic Warfare Conference & Exhibition*, London, 14-15 May 2009.

[30] P. Sapaty, K.-D. Kuhnert, M. Sugisaka and R. Finkelstein, "Developing High-Level Management Facilities for Distributed Unmanned Systems," *Proceedings of Fourteenth International Symposium on Artificial Life and Robotics*, Beppu, 5-7 February 2009, pp. 123-129.

[31] P. Sapaty, "Distributed Technology for Global Dominance," *Defense Transformation and Net-Centric Systems*, Orlando, 18 March 2008, pp. 33-41.

[32] P. Sapaty, M. Sugisaka, J. Delgado-Frias, J. Filipe and N. Mirenkov, "Intelligent Management of Distributed Dynamic Sensor Networks," *Artificial Life and Robotics*, Vol. 12, No. 1-2, 2008, pp. 51-59. doi:10.1007/s10015-007-0446-8

[33] P. Sapaty, "Global Management of Distributed EW-Related System," *Electronic Warfare*: *Operations & Systems*, London, 19-20 September 2007.

[34] P. Sapaty, M. Sugisaka, N. Mirenkov, M. Osano and R. Finkelstein, "Grasping the Distributed Entirety," *Proceedings of Tenth International Symposium on Artificial Life and Robotics*, Beppu, 4-6 February 2005, pp. 266-270.

[35] P. Sapaty, V. Klimenko and M. Sugisaka, "Dynamic Air Traffic Management Using Distributed Brain Concept," *Proceedings of Ninth International Symposium on Artificial Life and Robotics*, Beppu, January 2004, pp. 156-159.

[36] P. Sapaty and M. Sugisaka, "Optimized Space Search by Distributed Robotic Teams," *Proceedings of World Symposium Unmanned Systems*, Baltimore, 15-17 July 2003.

[37] P. Sapaty and M. Sugisaka, "Universal Distributed Brain for Mobile Multi-robot Systems," Sringer-Verlag, Berlin, 2002.

[38] P. Sapaty, M. J. Corbin and S. Seidensticker, "Mobile Intelligence in Distributed Simulations," *Proceedings of* 14*th Workshop on Standards for the Interoperability of Distributed Simulations*, Orlando, 11-15 March 1996, pp. 1045-1058.

