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

A ubiquitous and universal solution for management of distributed dynamic systems will be presented. It can allow us to grasp complex systems at a higher than usual, semantic level, establishing dominance over their organizations and providing the global behavior needed.

Contents

1 Current and Future Problems and Threats

- Overpopulation
- Polluted environment
- Global warming (sea level predicted rising by 1.5m by the end of the century)
- National and international terrorism
- Ethnic and religious conflicts
- Soaring food prices
- Exhaustion of fuel resources
- Widening gap between richness and poverty
- Nuclear proliferation, global destruction threat

2 Global Solutions to Global Problems Required

- **Distributed, penetrating national and international borders**
- **Providing global awareness and based on it**
- **Runtime, real time, and ahead of real time**
- **Involving massive movement, concentration and distribution of resources**
- **Based on effective negotiations between opposing parties**

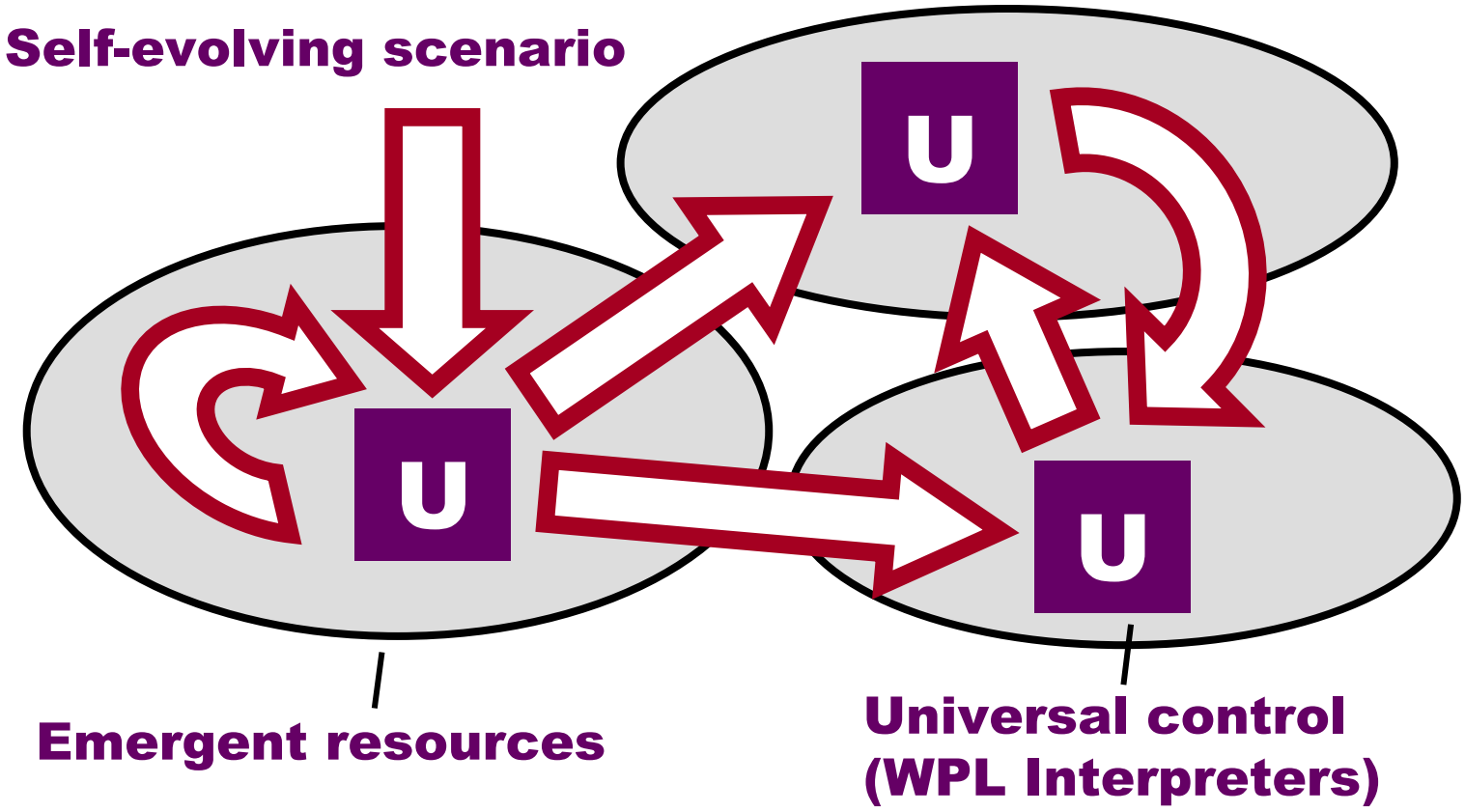
3 The Technology Sketch

3.1 World Processing Technology, WPT

- **WPT is capable of solving global problems by converting any system into an integral spatial supercomputer.**
- **It is based on the World Processing Language, WPL, describing semantics of problems in distributed spaces rather than details of implementation.**
- **Communicating WPL interpreters can be installed in internet hosts, mobile phones, mobile robots, smart sensors, or implanted into animals and insects.**

3.2 World Conquest by Spatial Scenarios

A dynamic network of WPL interpreters collectively executes mission scenarios, which can start from any nodes and cover the distributed systems at runtime.

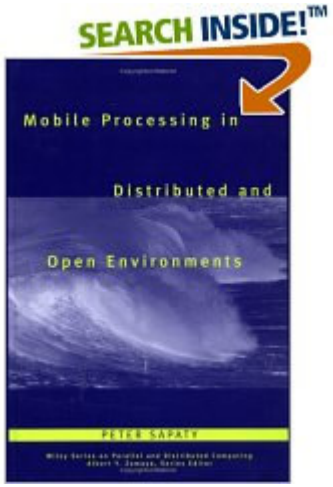


3.3 Investigated Applications

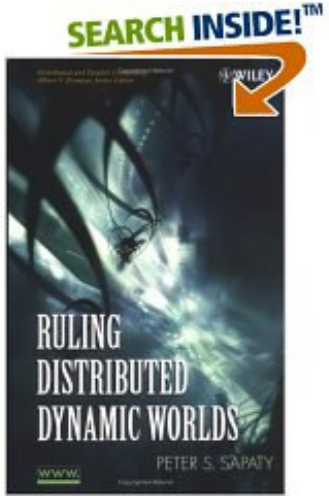
- Distributed knowledge bases
- Distributed inference and decision making in semantic networks
- Solving classical graph and network problems
- Intelligent network management
- Distributed virtual reality
- Distributed simulation of dynamic systems (battlefields, road networks)
- Collective behavior of robots and infrastructure protection
- Emergency management
- Flexible command and control
- Distributed management of directed energy systems
- Finding global solutions by smart sensor networks

3.4 John Wiley Books

www.amazon.com



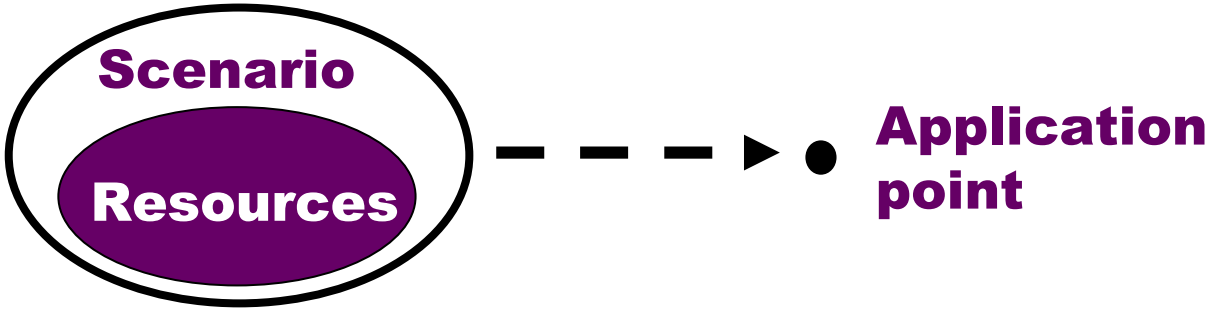
Mobile Processing in Distributed and Open Environments (Wiley Series on Parallel and Distributed Computing) (Hardcover - Feb 22, 1999)



Ruling Distributed Dynamic Worlds (Wiley Series on Parallel and Distributed Computing) (Hardcover - May 31, 2005)

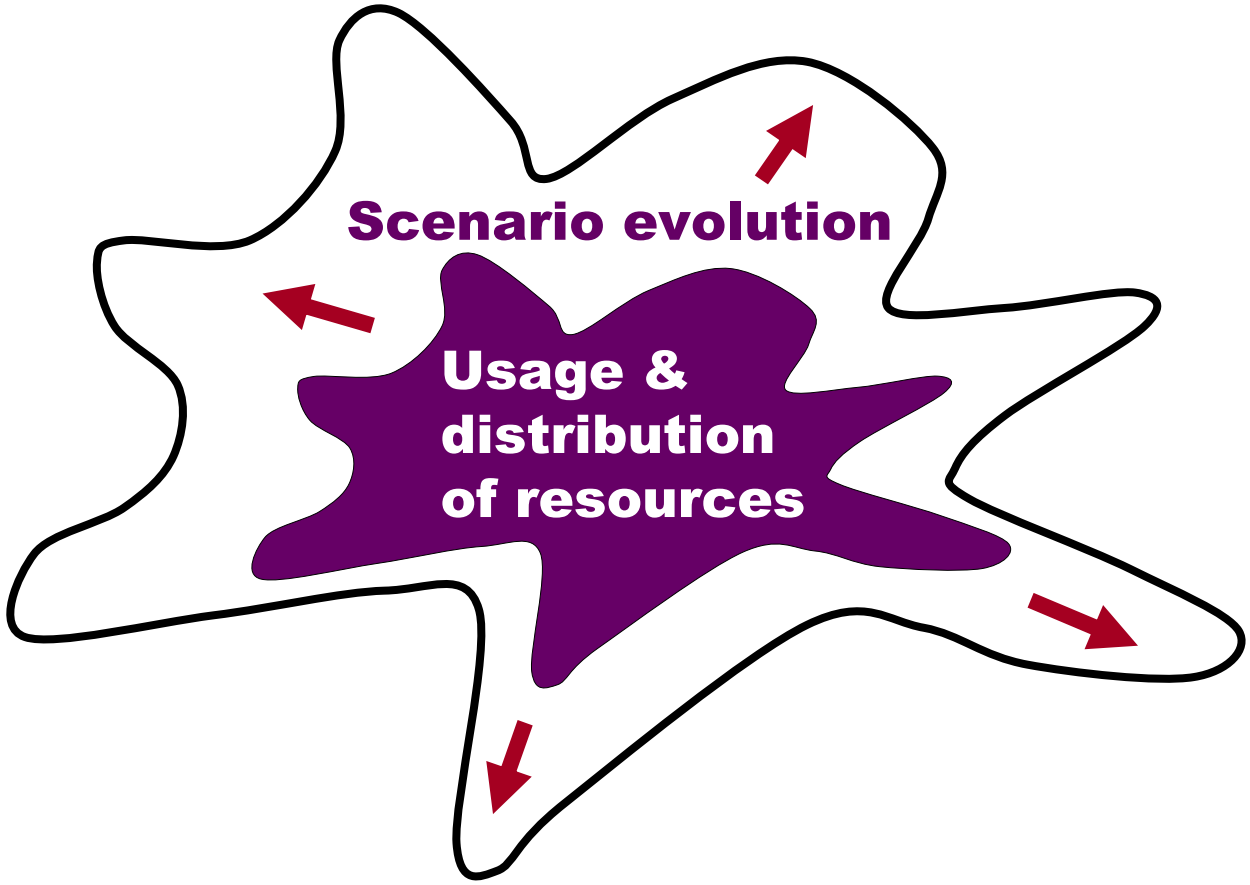
4 World Processing Technology Basics

4.1 The Initial State

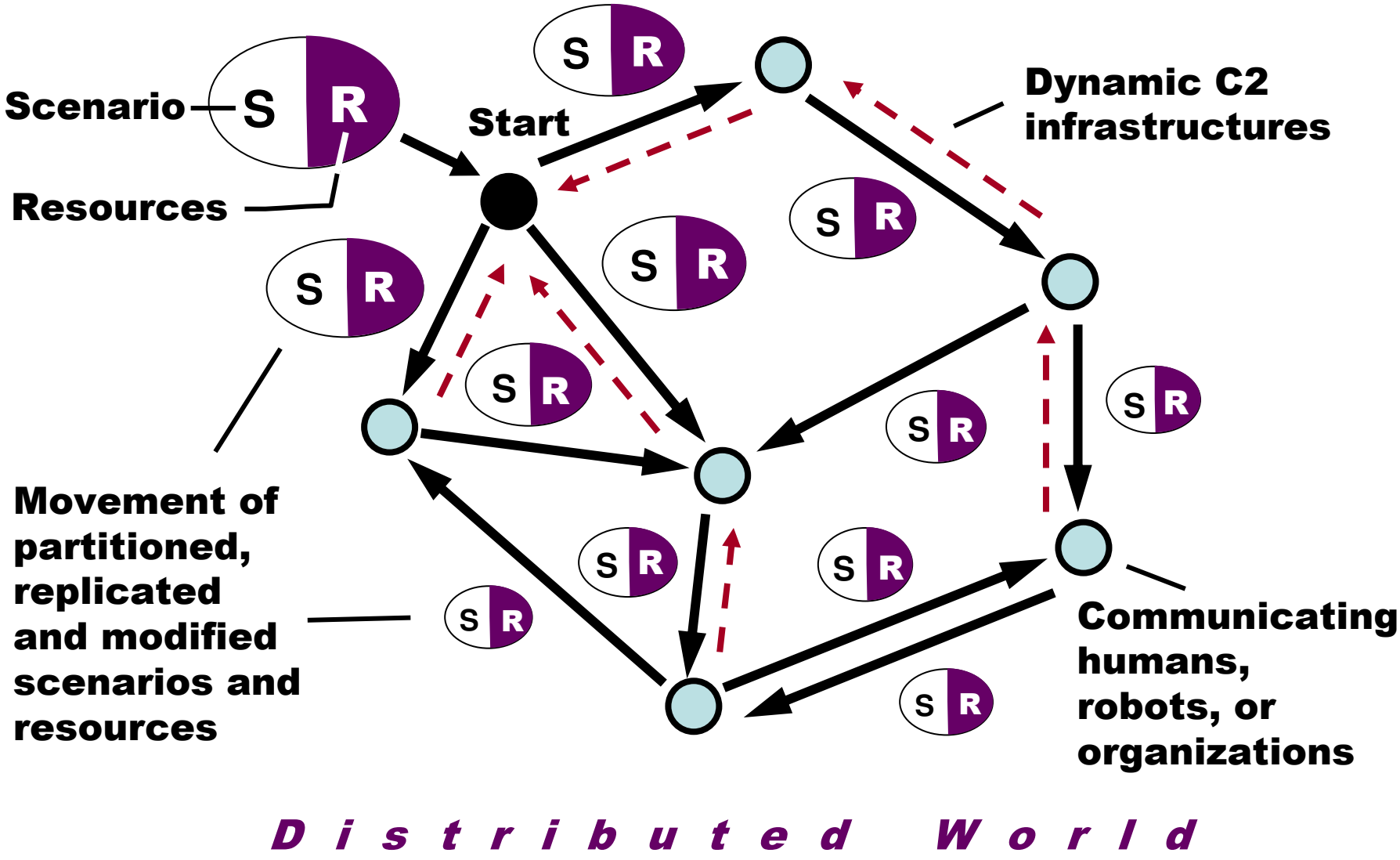


Distributed World

4.2 Spreading Operations and Resources



4.3 Resultant Interactions Between System Parts

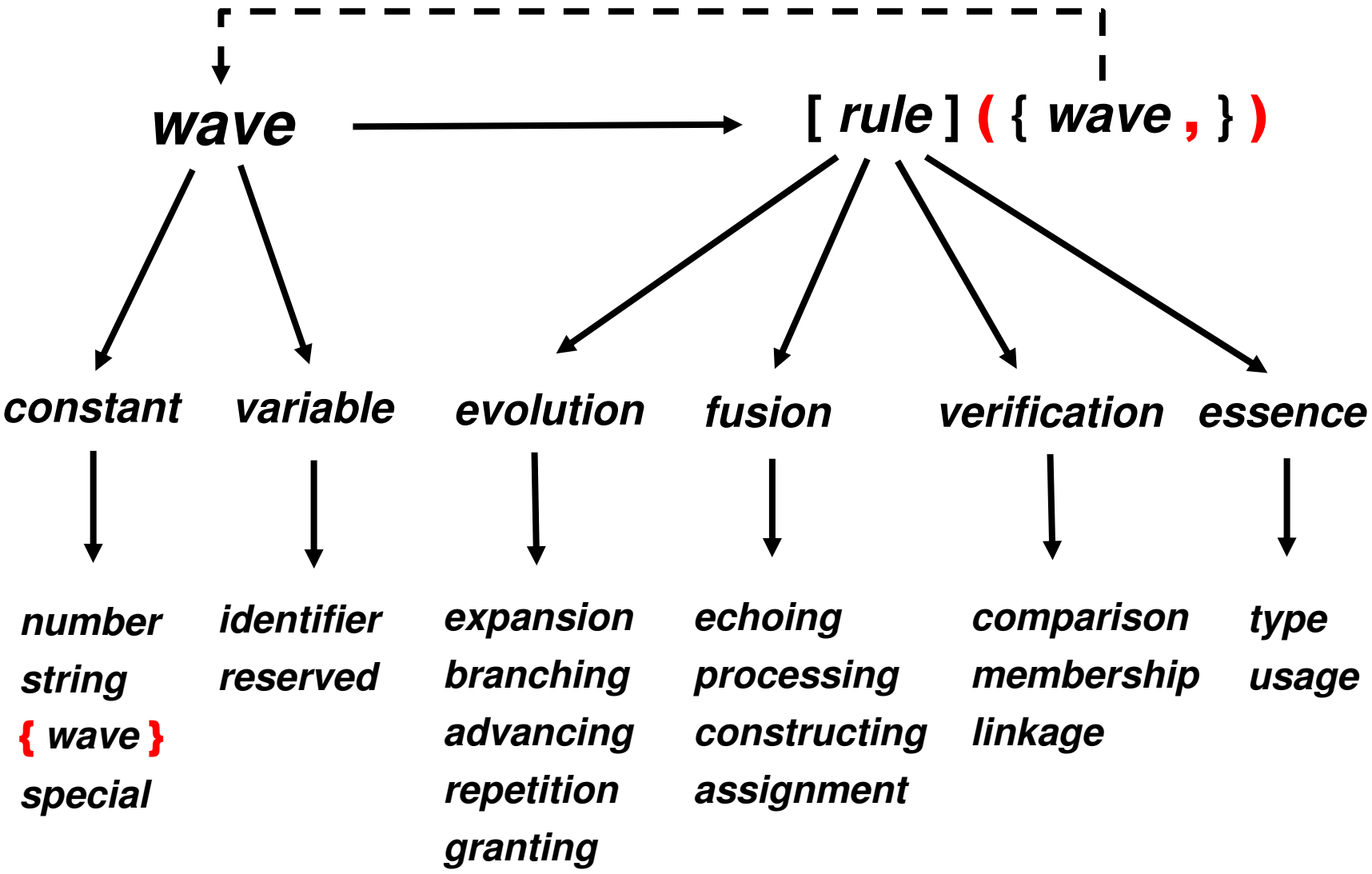


5 The World Processing Language, WPL

5.1 Main WPL Features

- It is a higher-level language to task and control human organizations.
- It is also a 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 and does not need central resources.
- Synchronization of multiple activities and collective behavior (swarm, as well as hierarchically controlled) are organized automatically by the networked interpreter.
- This drastically simplifies application programming, which is often hundreds of times more compact than in traditional languages.

5.2 WPL Top Syntax



5.3 Further WPL Details

special	➔	abort | thru | done | fail | direct | any | all | out | in | neighbors | infinite | nil | random | first | last
reserved	➔	QUALITIES | CONTENT | LINK | DIRECTION | TIME | SPEED | ADDRESS | WHERE | WHEN | BACK | VALUE | ORDER | COLOR | RESOURCES | USER
expansion	➔	hop | apply | external | replicate | split | choose
branching	➔	emit | follow | succeed | if | or | and | or parallel | and parallel
advancing	➔	advance | advance sync | forward | forward sync
repetition	➔	cycle | sling | repeat | repeat sync
granting	➔	release | free | quit | grasp | create | revive | set | stay
echoing	➔	rake | min | max | sort | sum | product | count | state
processing	➔	add | subtract | multiply | divide | degree
constructing	➔	part | unite | align | intersect | access | attach | aggregate
assignment	➔	assign | assign peers | input | output
comparison	➔	equal | not equal | less | less or equal | more | more or equal
membership	➔	belong | not belong | include | not include
linkage	➔	linked | not linked | reached | not reached
type	➔	stationary | carried | information | matter
usage	➔	address | place | latitude | longitude | range | time | speed | doer | node | link | index | content

5.4 A Rule Can Be:

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

5.5 Elementary Code

66

‘hello’

plus (4, 7)

min (8, 3, 9, 11)

hop (x77, y99)

hop (link (a), node (b))

advance (hop (node (b)), hop (link (c), node (d)))

assign (Result, minus (88, 22))

---

4 + 7

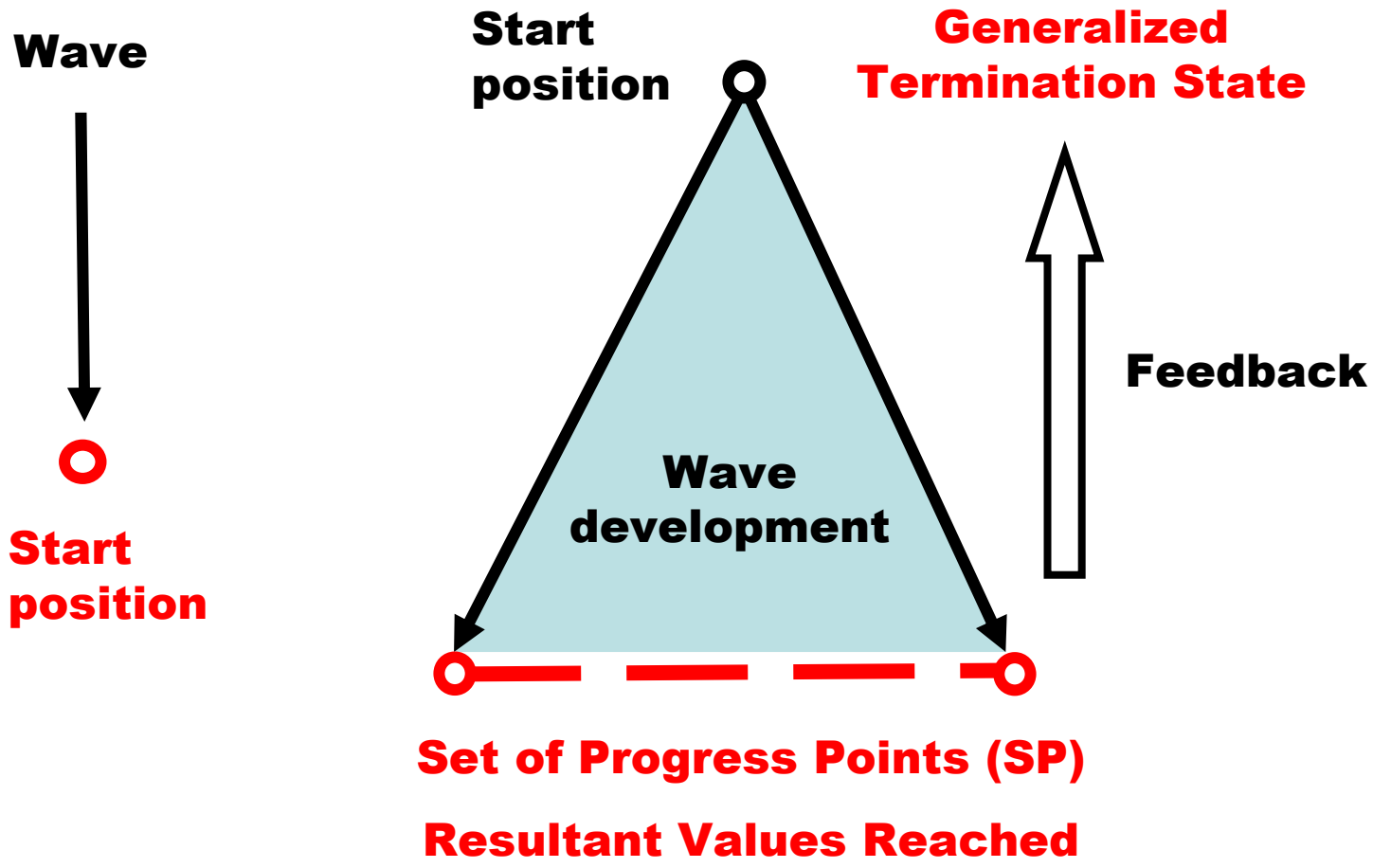
(x77, y99)

a # b

b; c # d

Result = 88 - 22

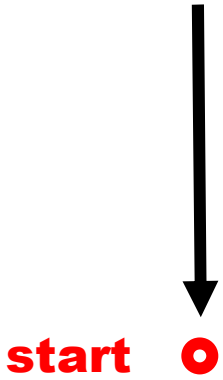
5.6 WPL Spatial Semantics



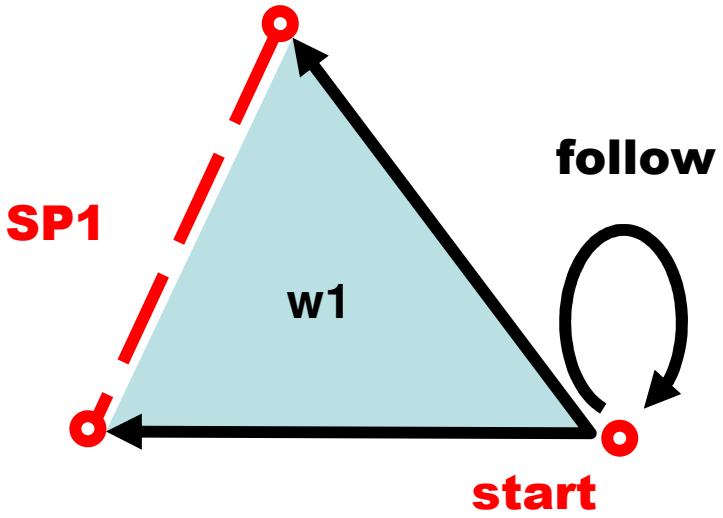
6 Examples of Spatial Rules

6.1 Rule Follow

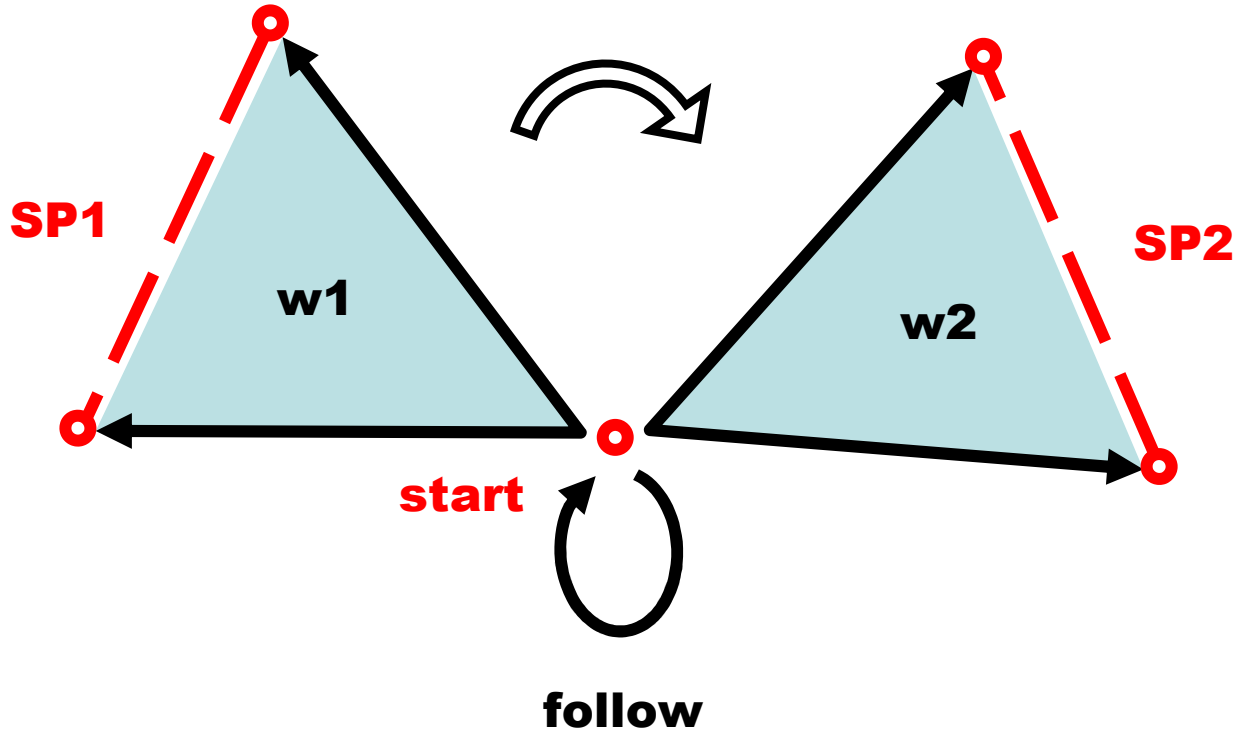
follow (w1, w2, w3)



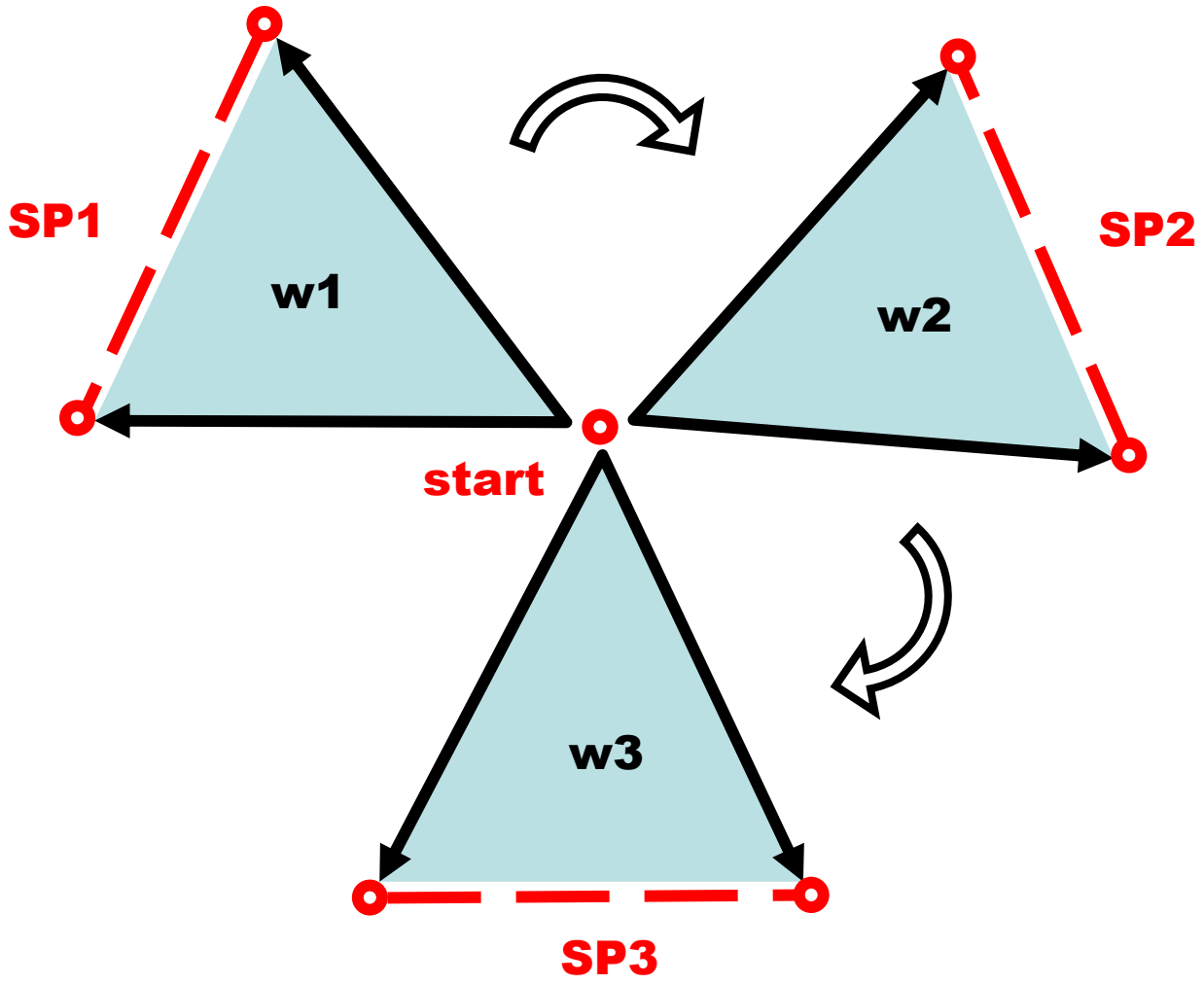
6.1.1 Rule Follow, Step 1



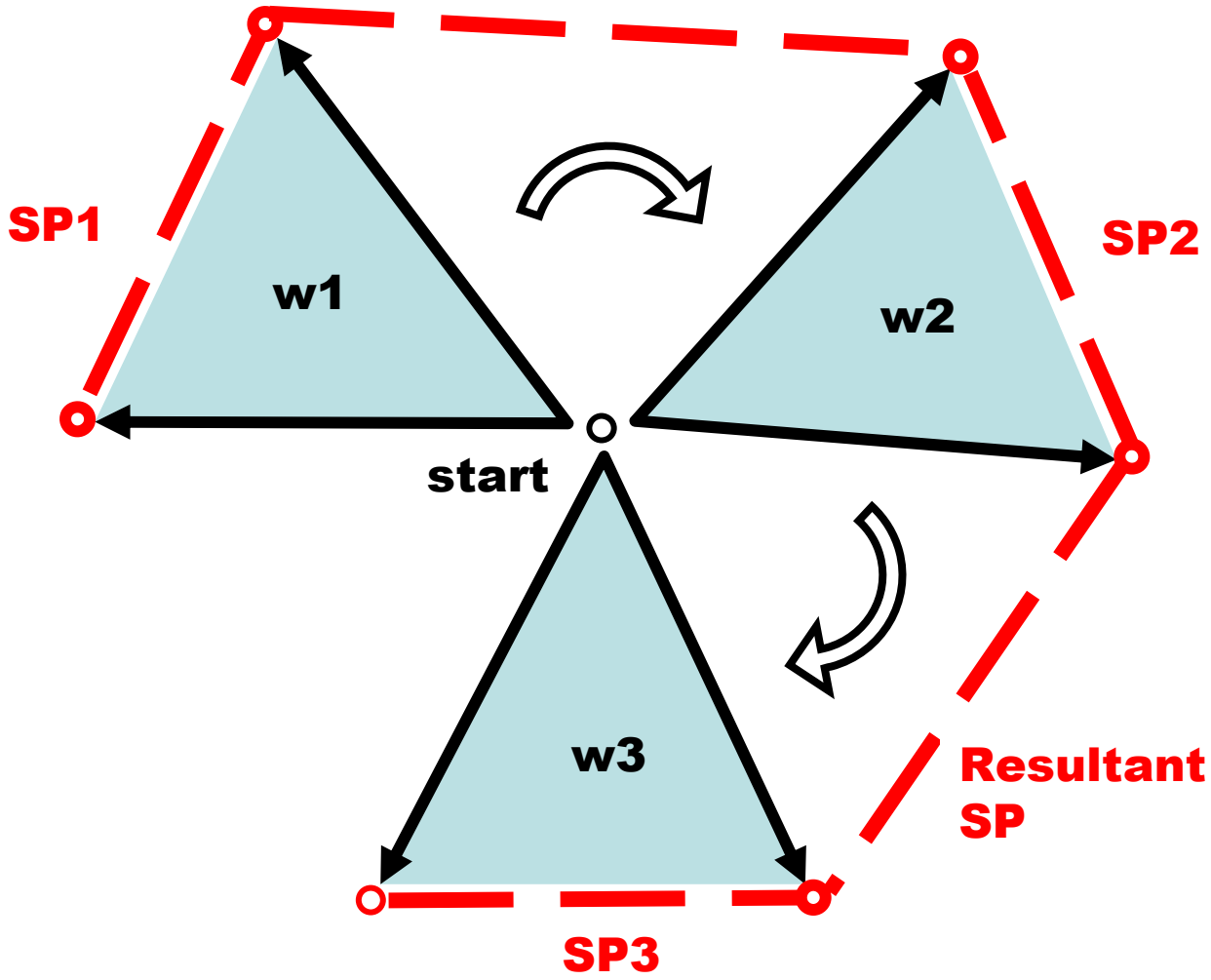
6.1.2 Rule Follow, Step 2



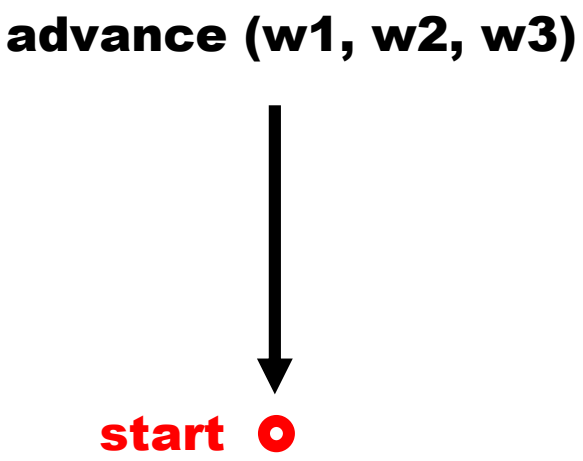
6.1.3 Rule Follow, Step 3



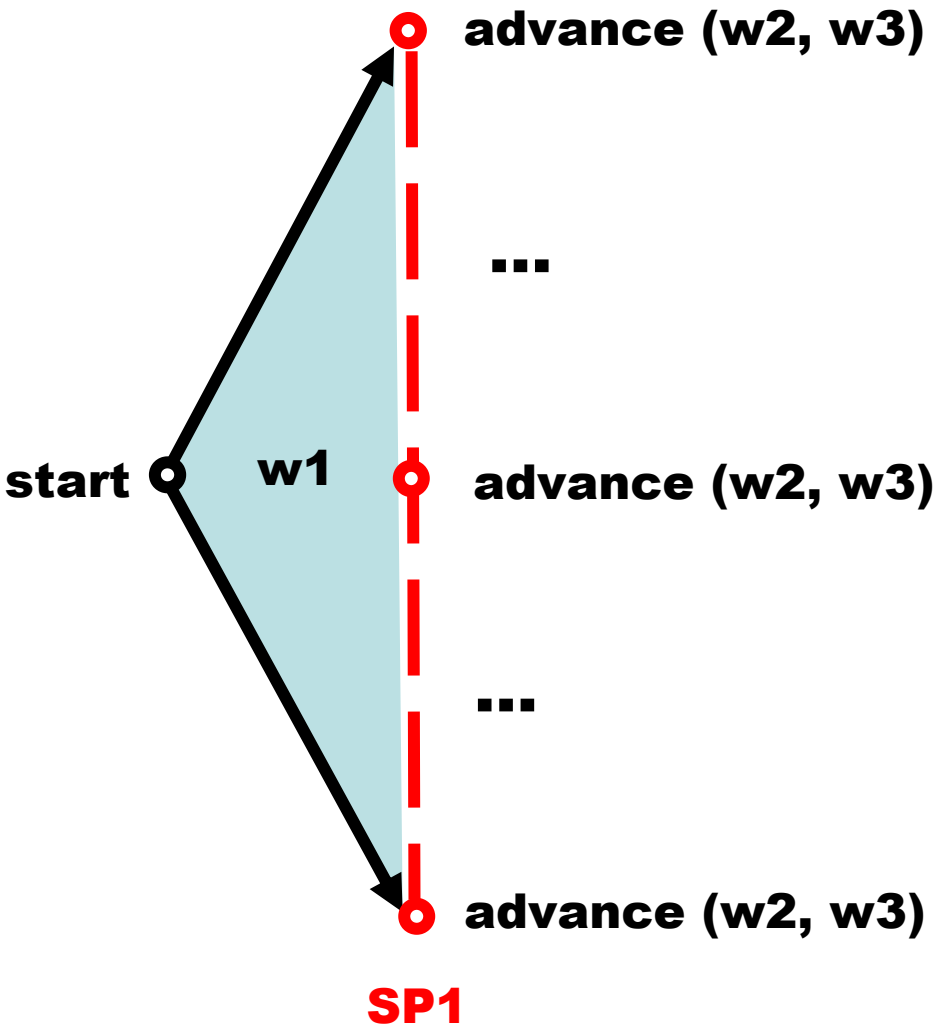
6.1.4 Rule Follow, Step 4



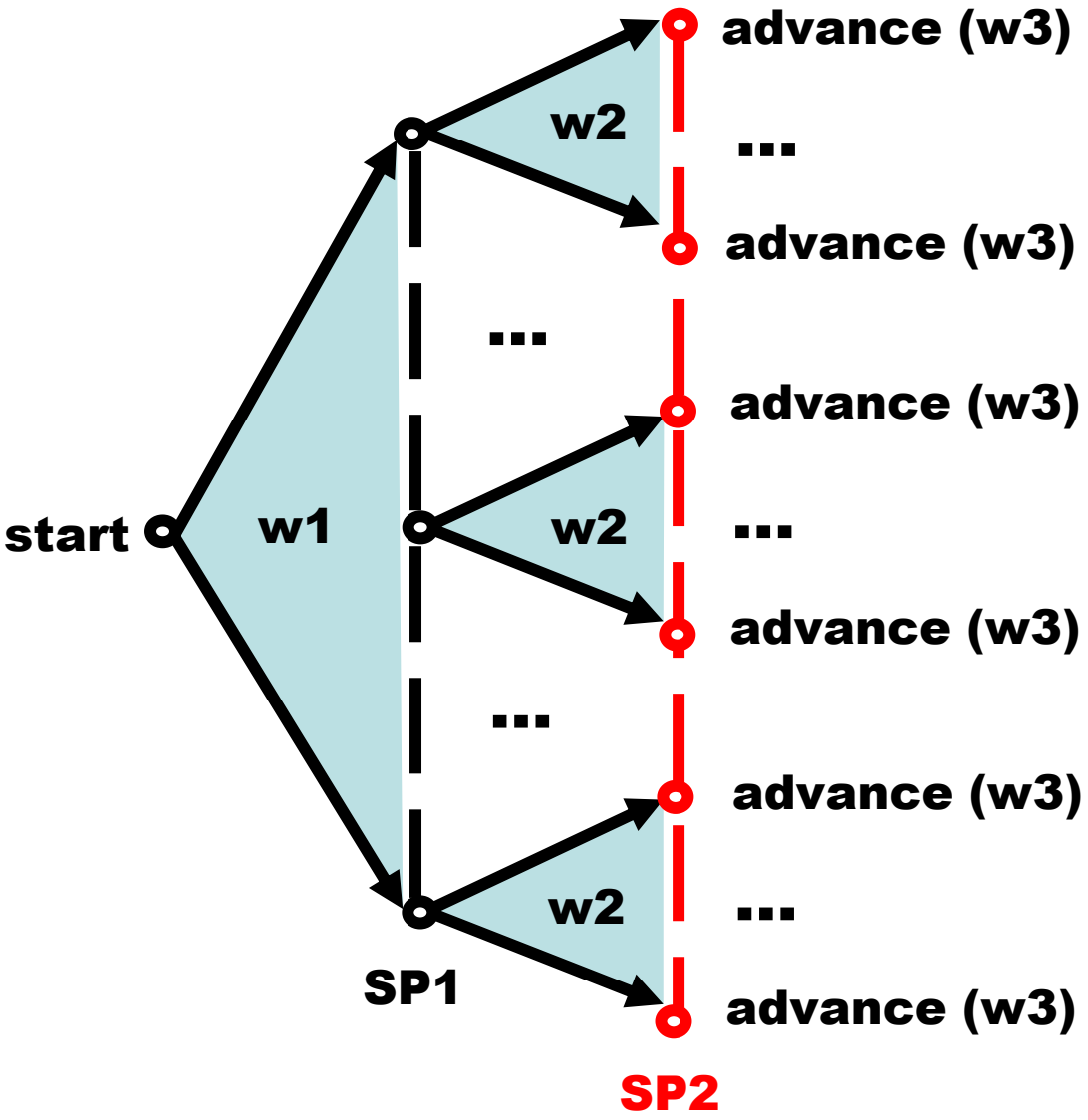
6.2 Rule Advance



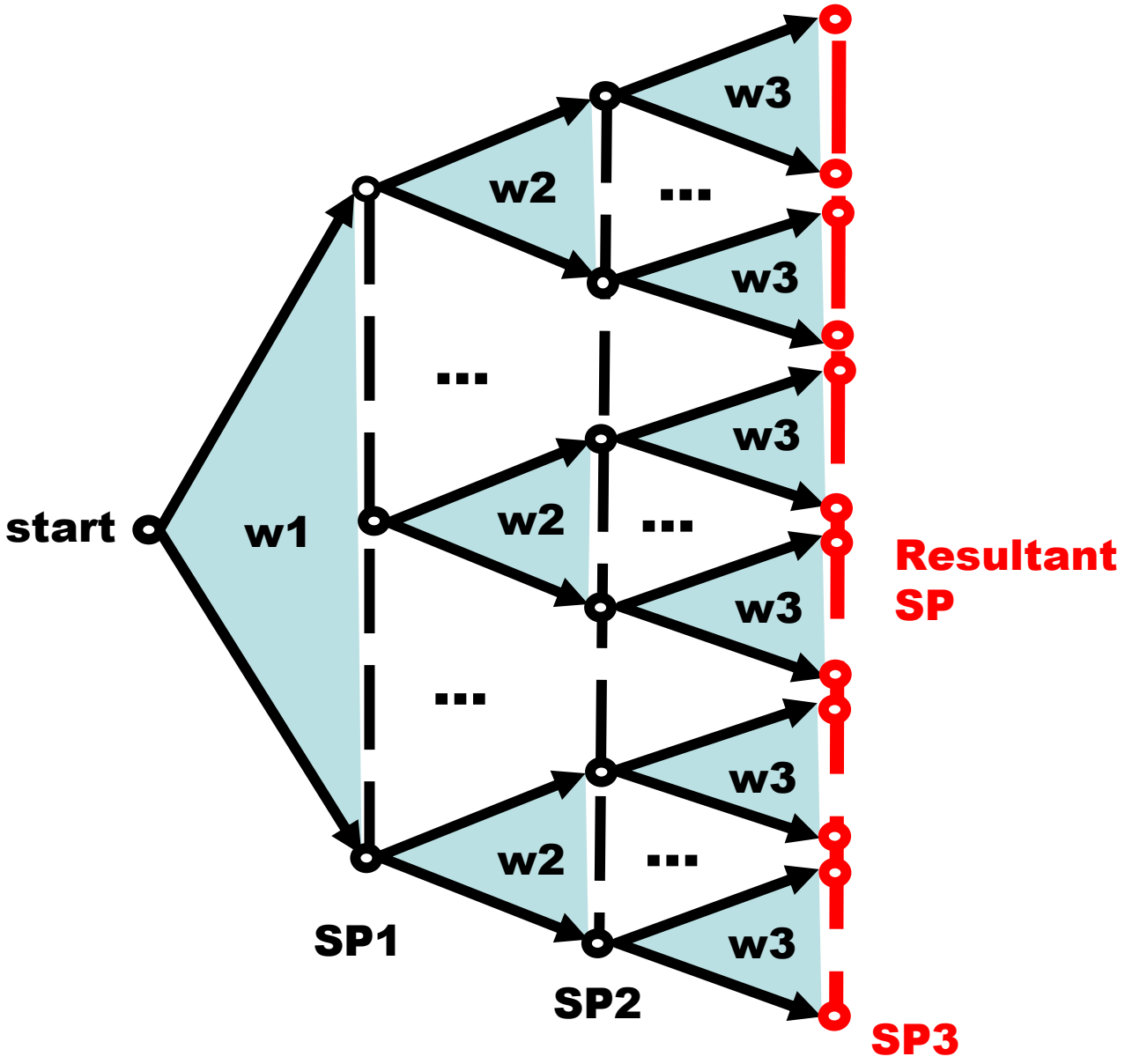
6.2.1 Rule Advance, Step 1



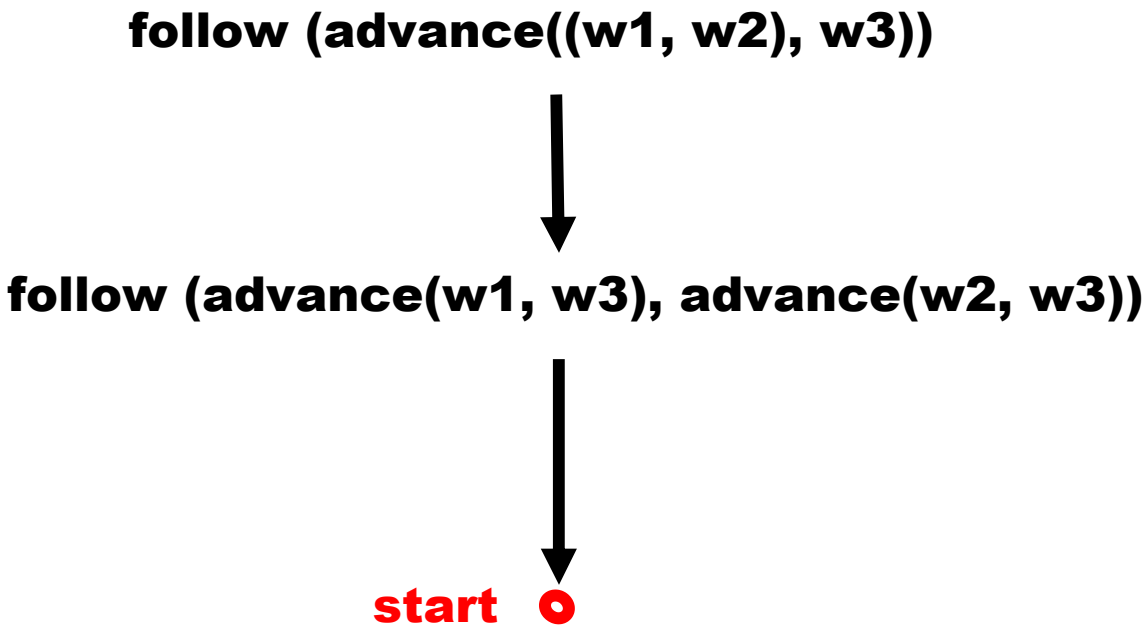
6.2.2 Rule Advance, Step 2



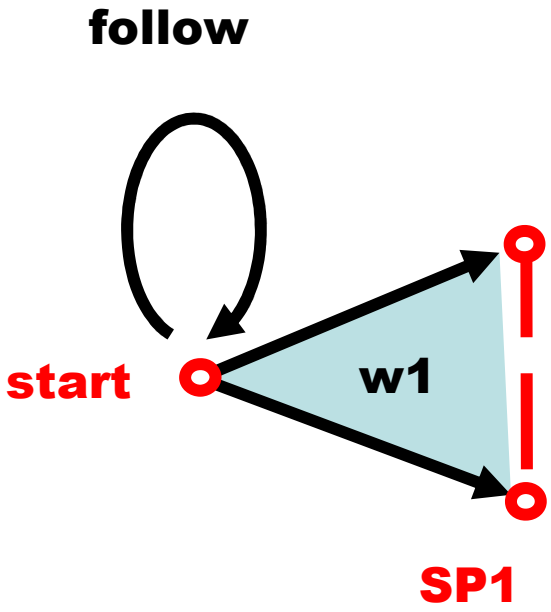
6.2.3 Rule Advance, Step 3



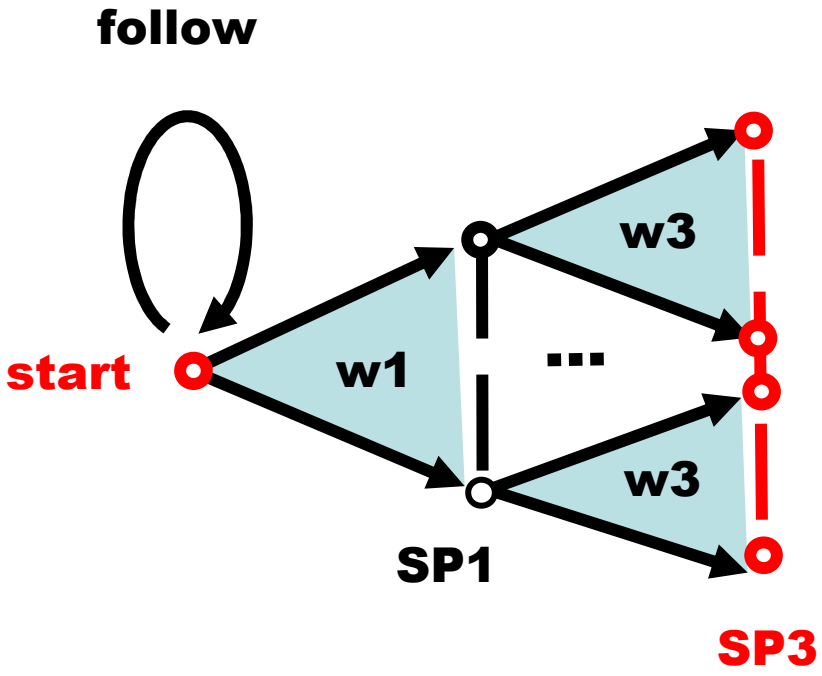
6.3 Combined Breadth-Depth Control



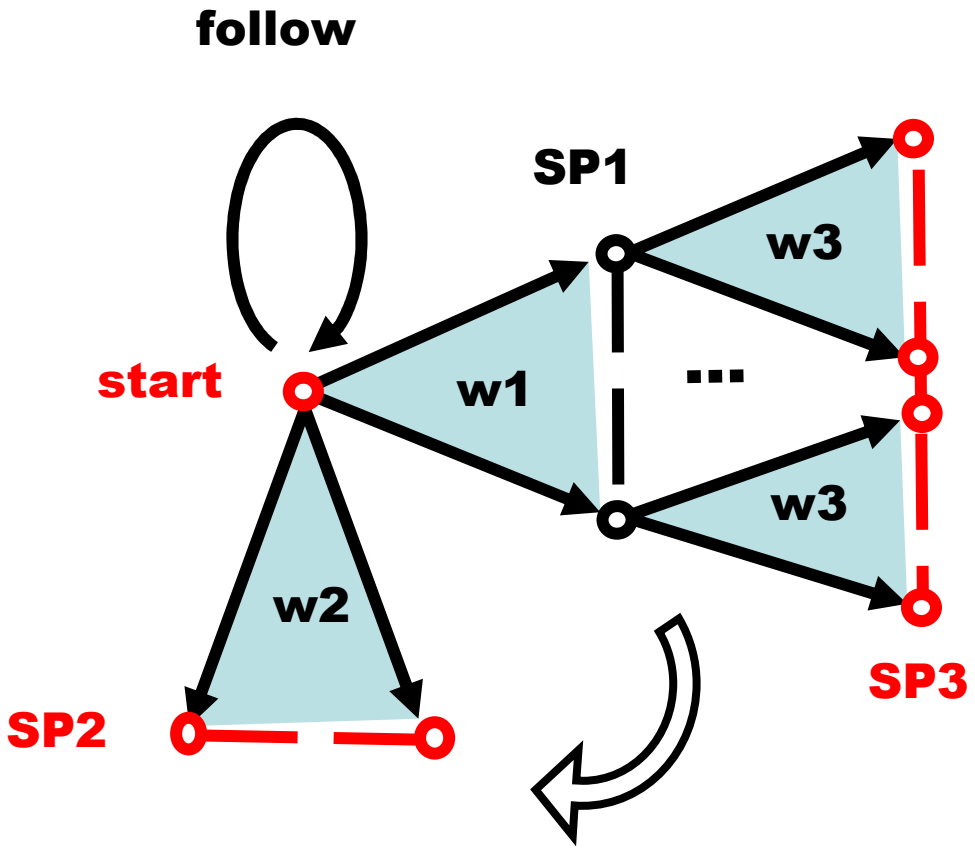
6.3.1 Combined Control, Step 1



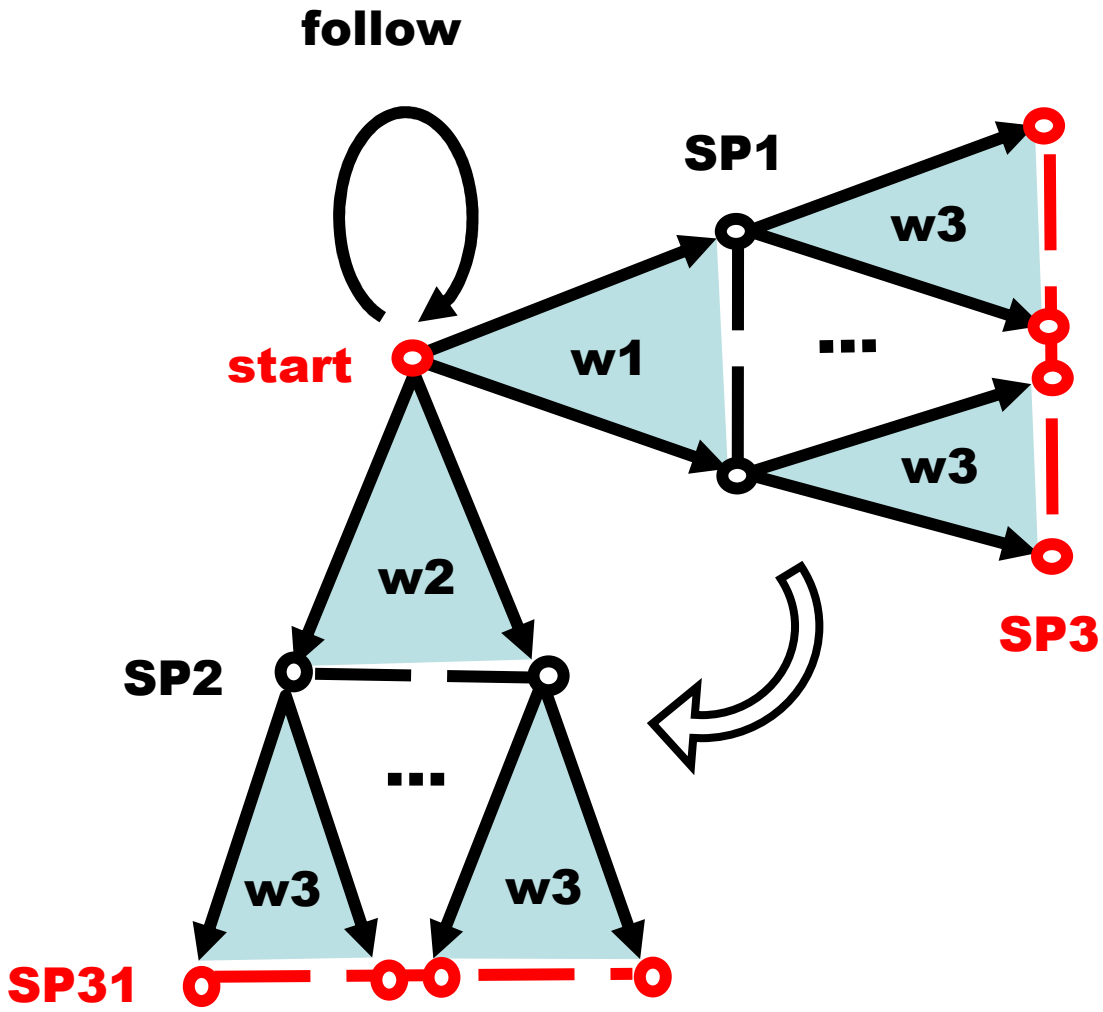
6.3.2 Combined Control, Step 2



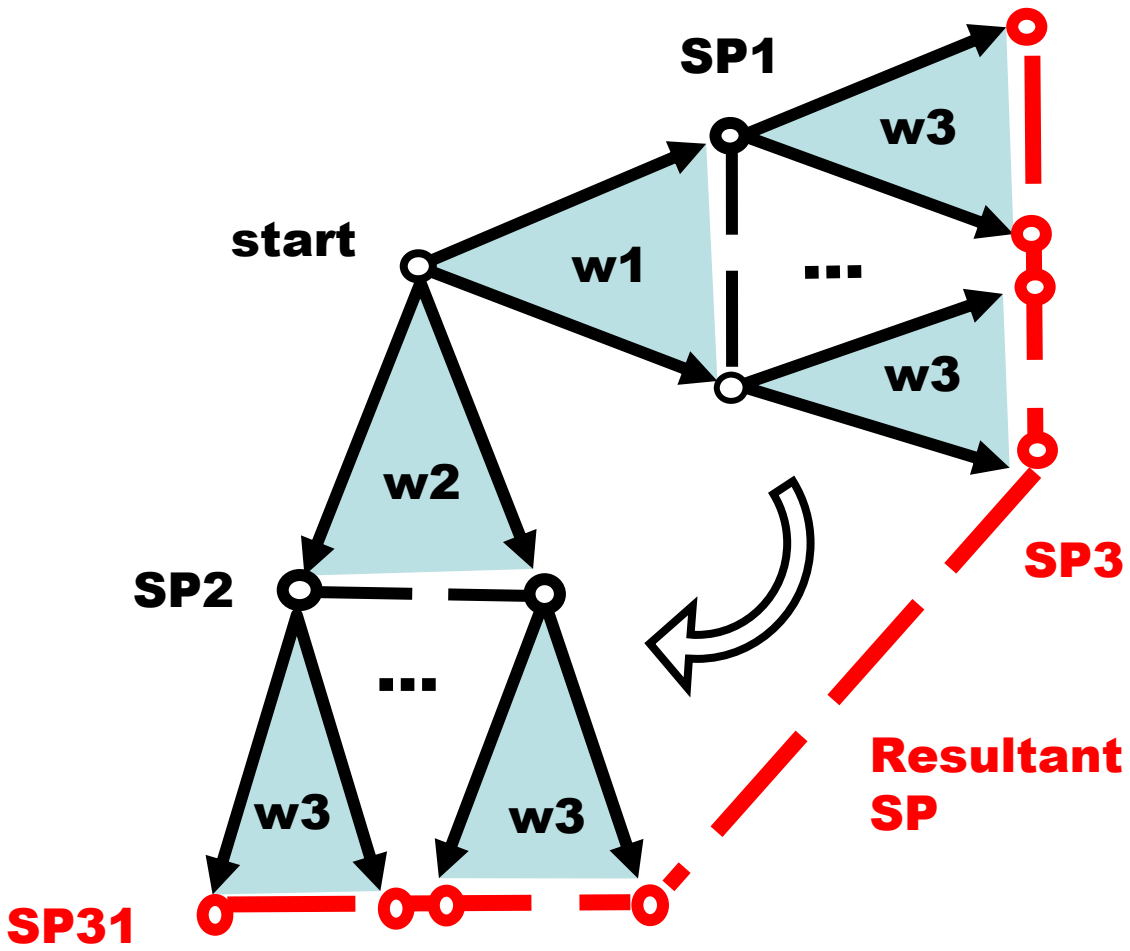
6.3.3 Combined Control, Step 3



6.3.4 Combined Control: Step 4



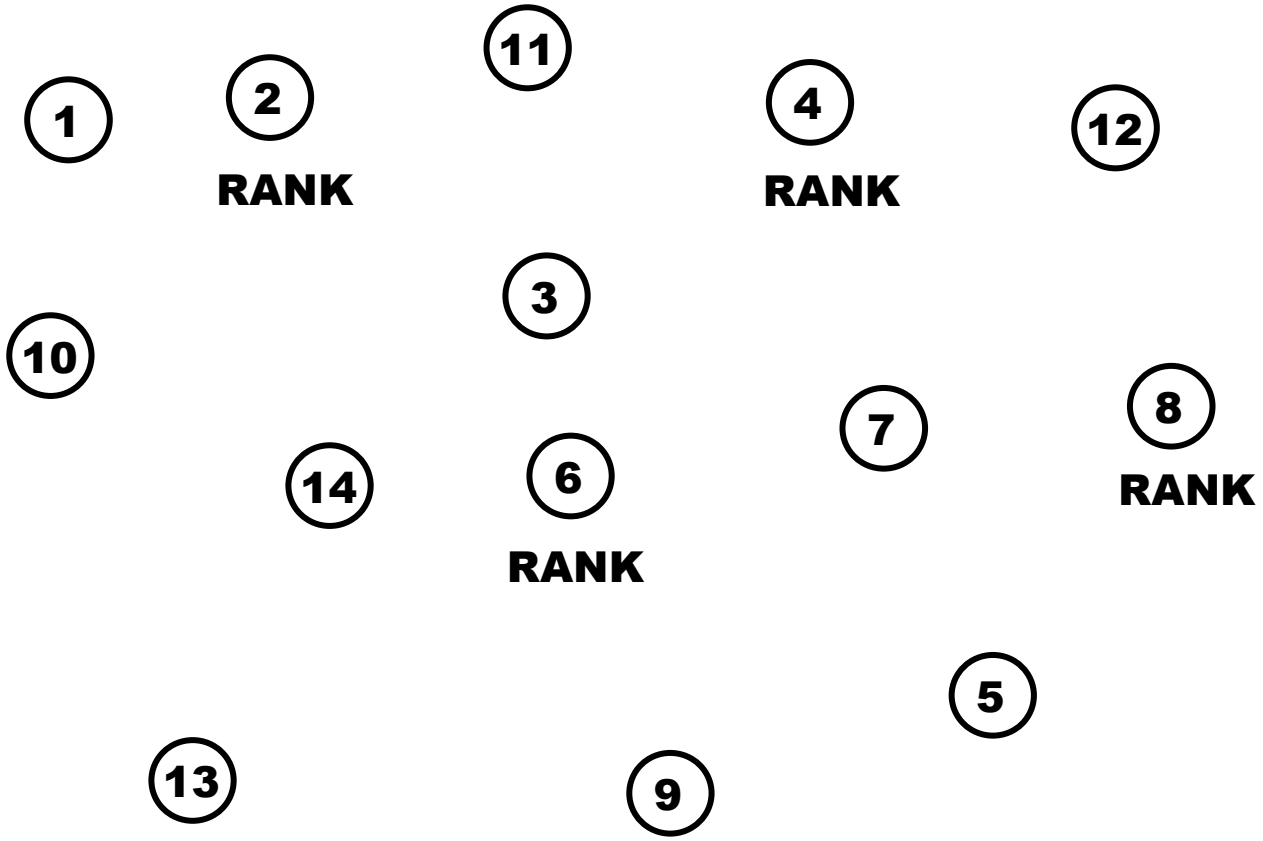
6.3.5 Combined Control, Step 5



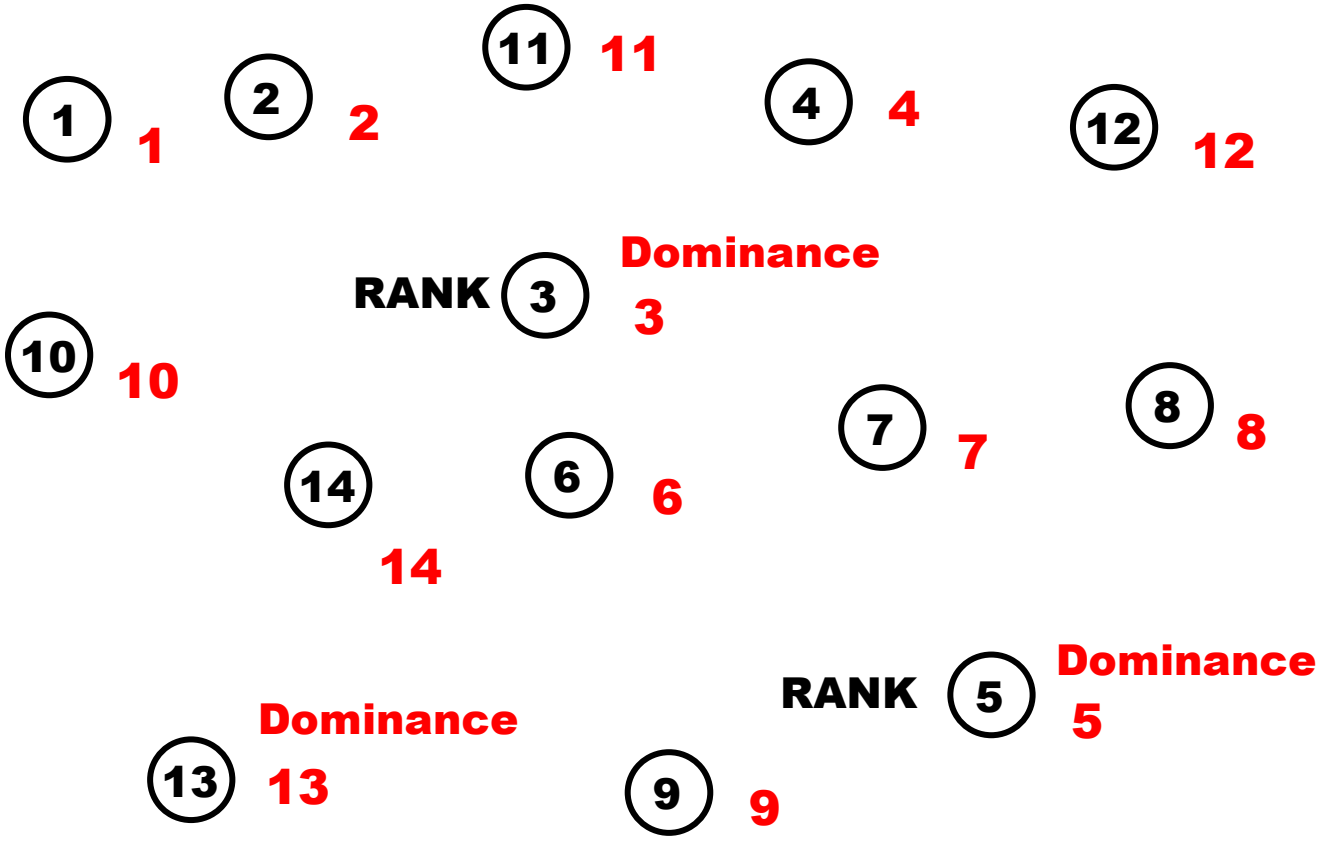
7 Elementary Programming Examples

7.1 Establishing Dominance

7.1.1 Distributed System Nodes



7.1.2 Establishing Self-Dominance in Each Node

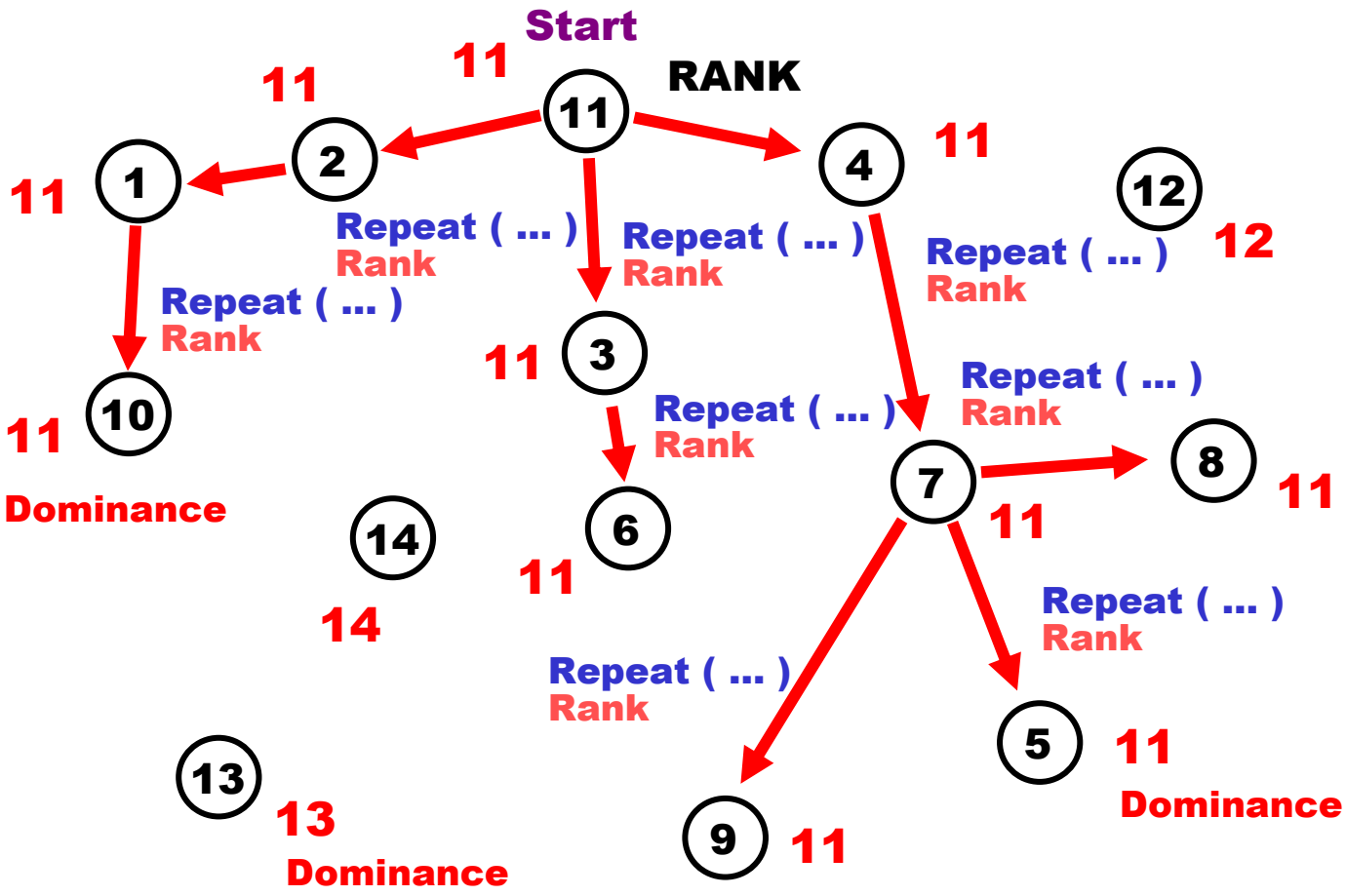


Hop all_nodes. Nodal Dominance = RANK

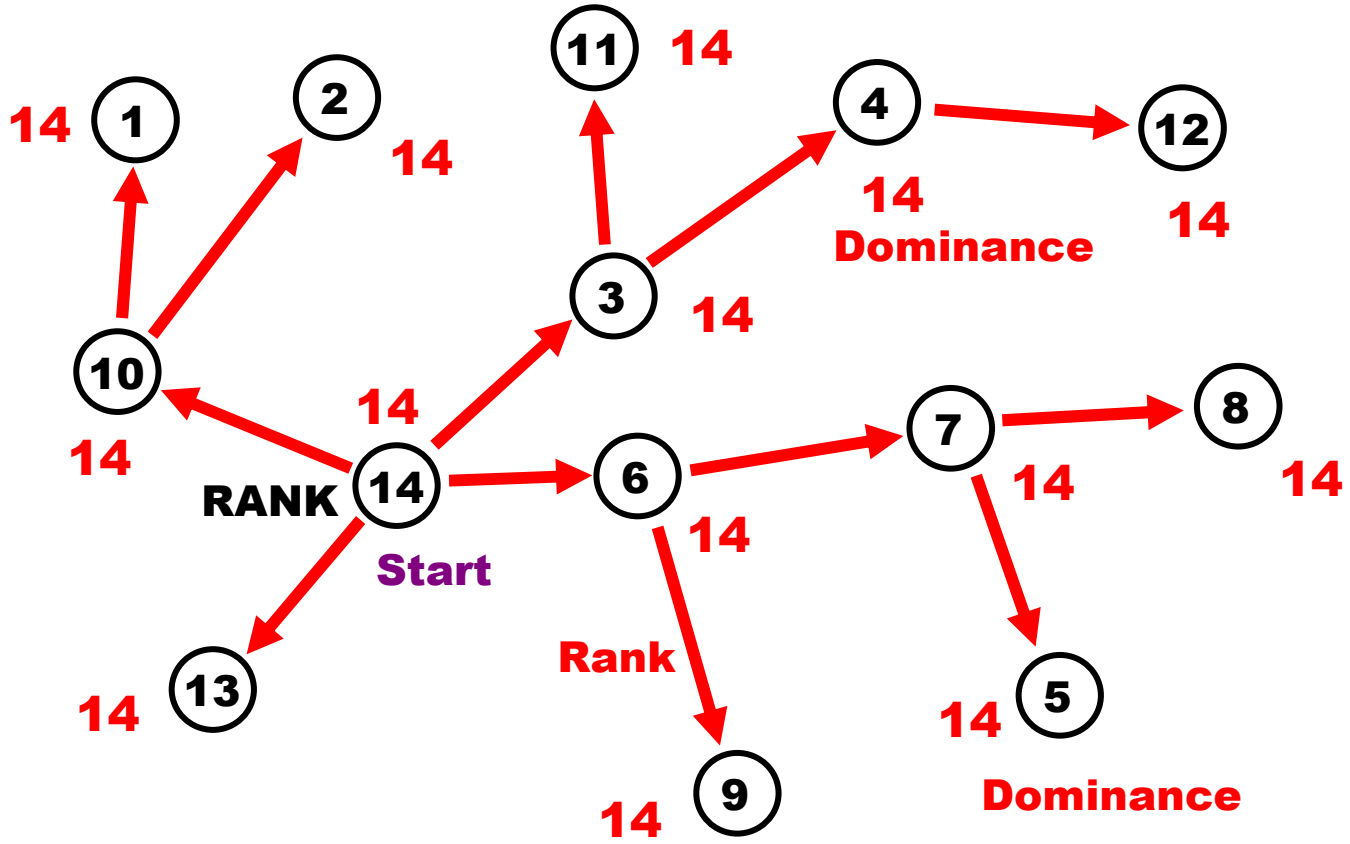
7.1.3 Spreading Node's Rank to Other Nodes

```
Frontal Rank = RANK.
Repeat (
  If Dominance < Rank then
      Dominance = Rank; hop all_neighbors
          Otherwise stop )
```

7.1.4 Resulting in Local Dominance



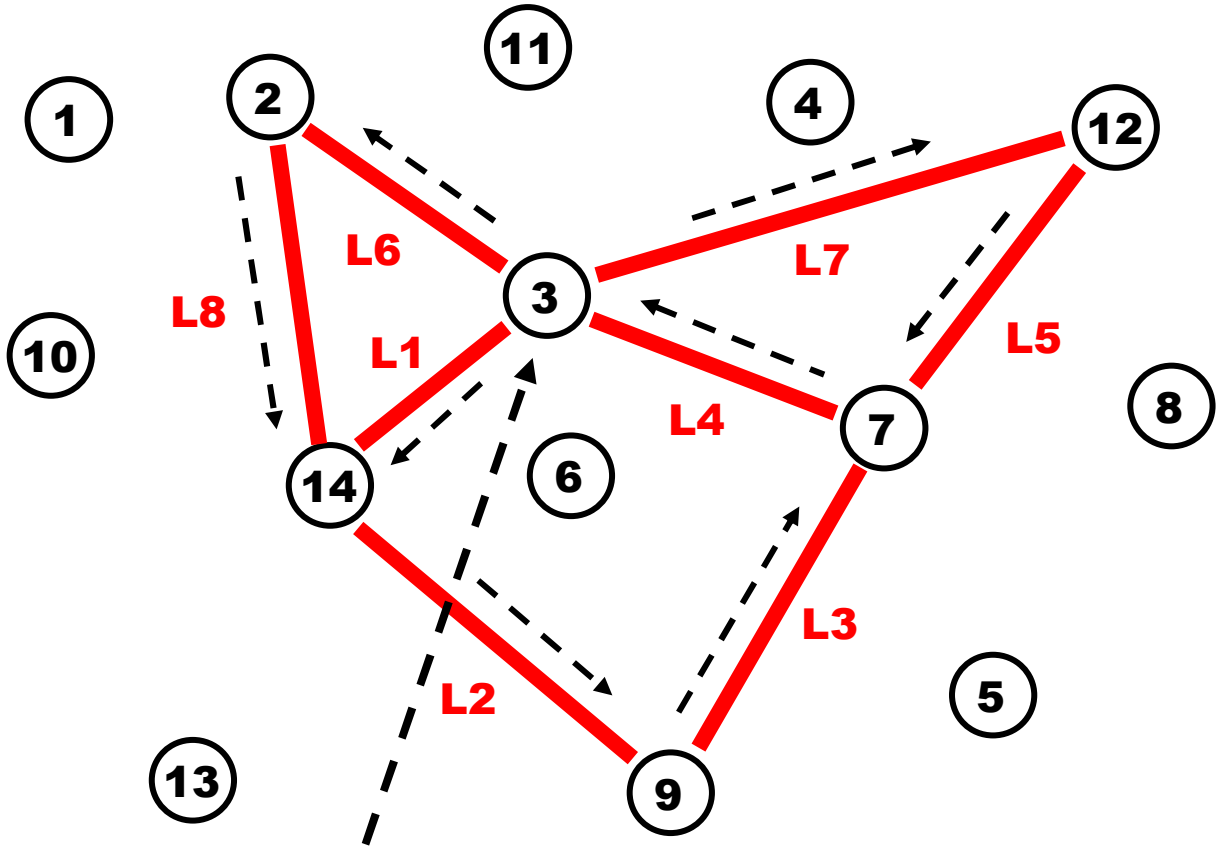
7.1.5 Resulting in Global Dominance



7.2 Infrastructure Management

7.2.1 Creating Infrastructure in Distributed Space

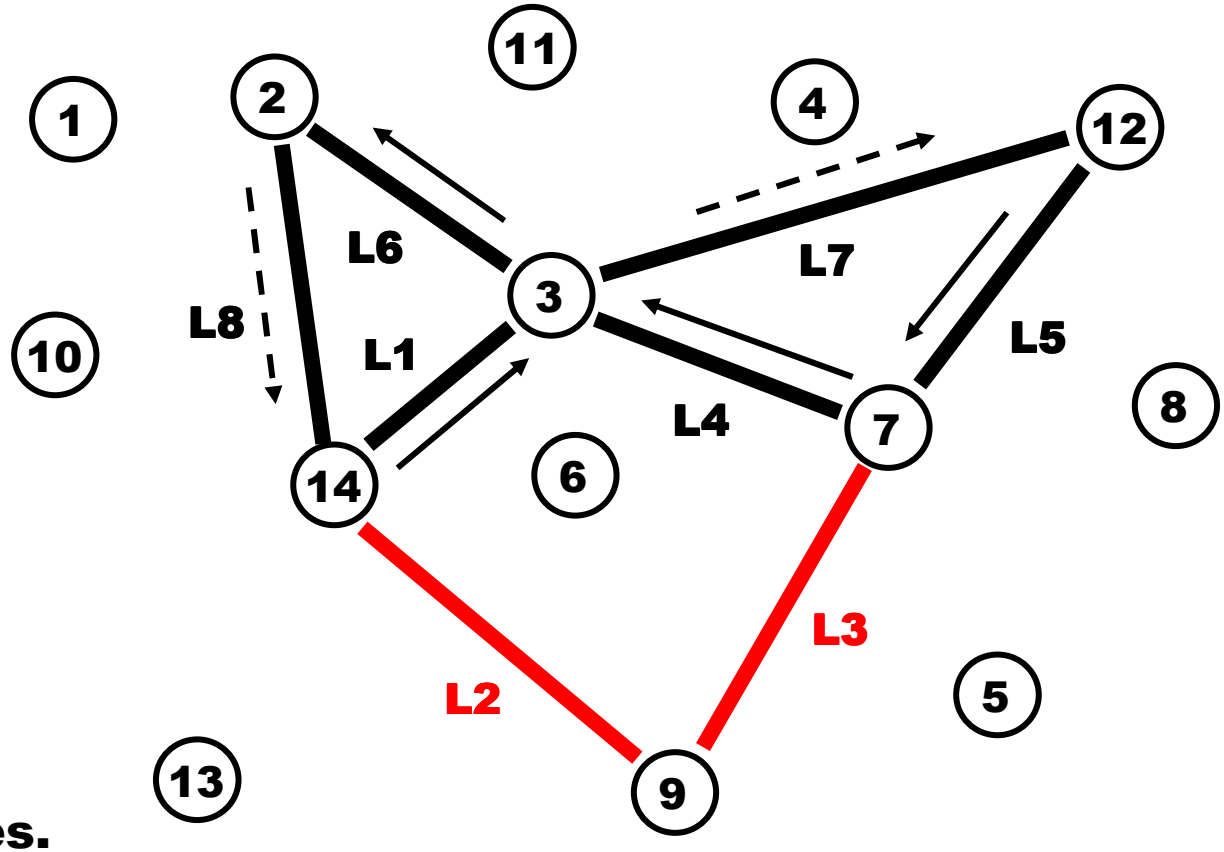
(Any functionality can be associated with both nodes and links of the obtained infrastructure at runtime)



Hop_node 3; Create_links(
(L6 # 2; L8 # 14), (L7 # 12; L5 # 7; L4 # 3), (L1 # 14; L2 # 9; L3 # 7))

7.2.2 Finding Patterns in the Infrastructure

(All Triangles)



Hop all_nodes.
Frontal (Triangle) = RANK. Hop all_links. RANK < BACK.
Triangle &= RANK. Hop all_links. RANK < BACK. Triangle &= RANK.
Hop all_links. Triangle [1] == RANK. Output Triangle

Result: (14, 3, 2), (12, 7, 3)

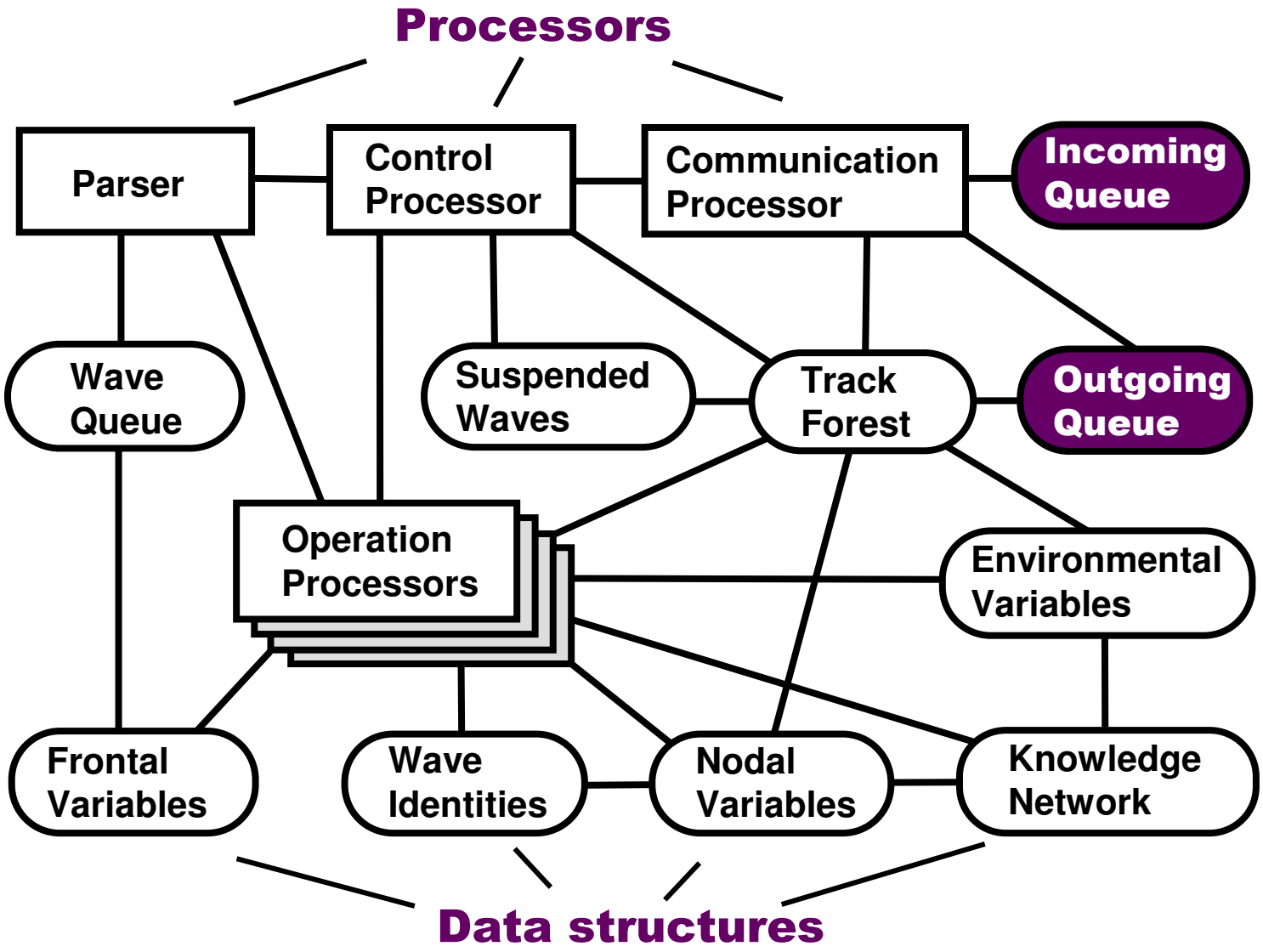
8 The WPL Interpreter

8.1 The Interpreter Basics

- The WPL 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.

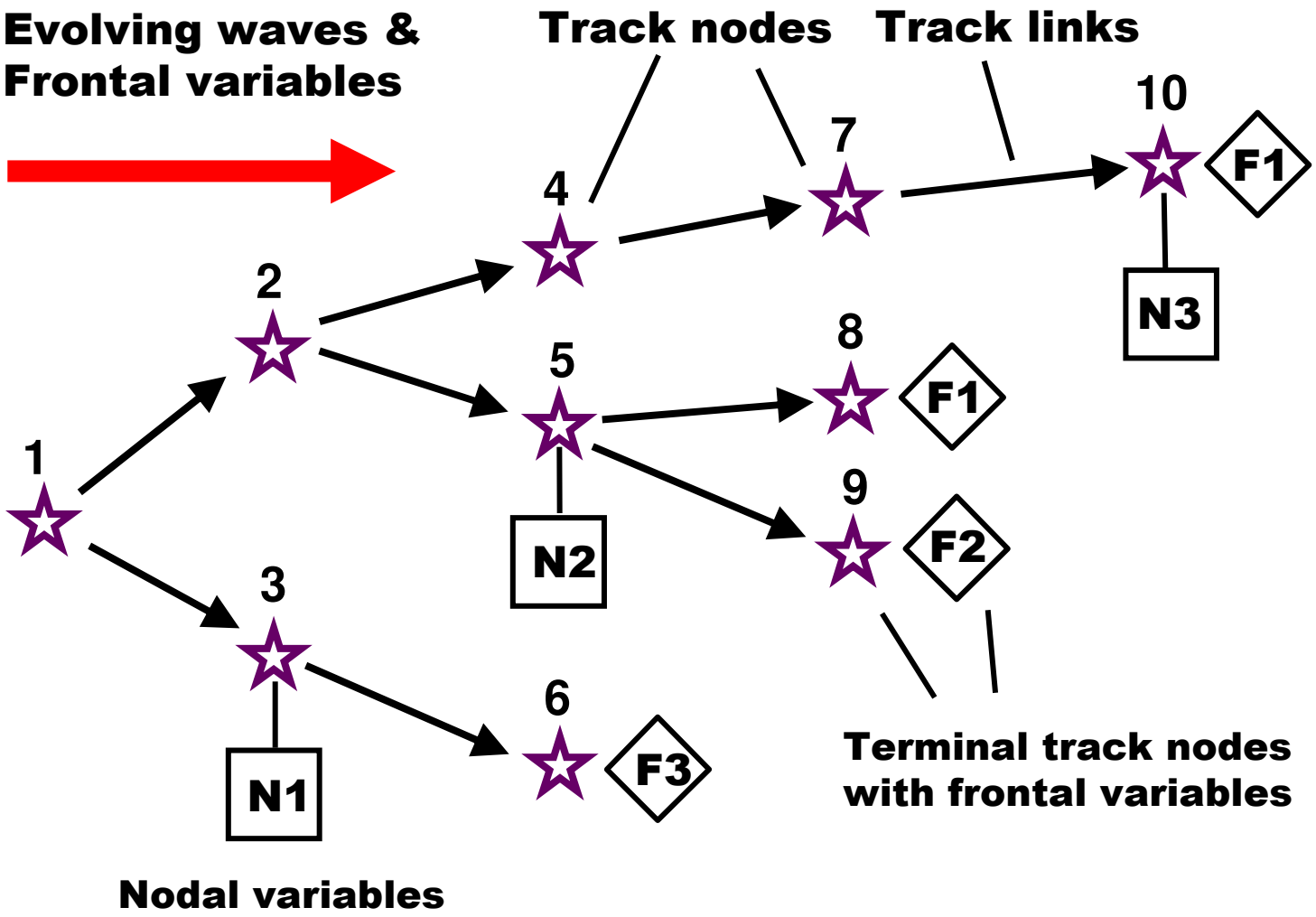
8.2 The WPL Interpreter Structure



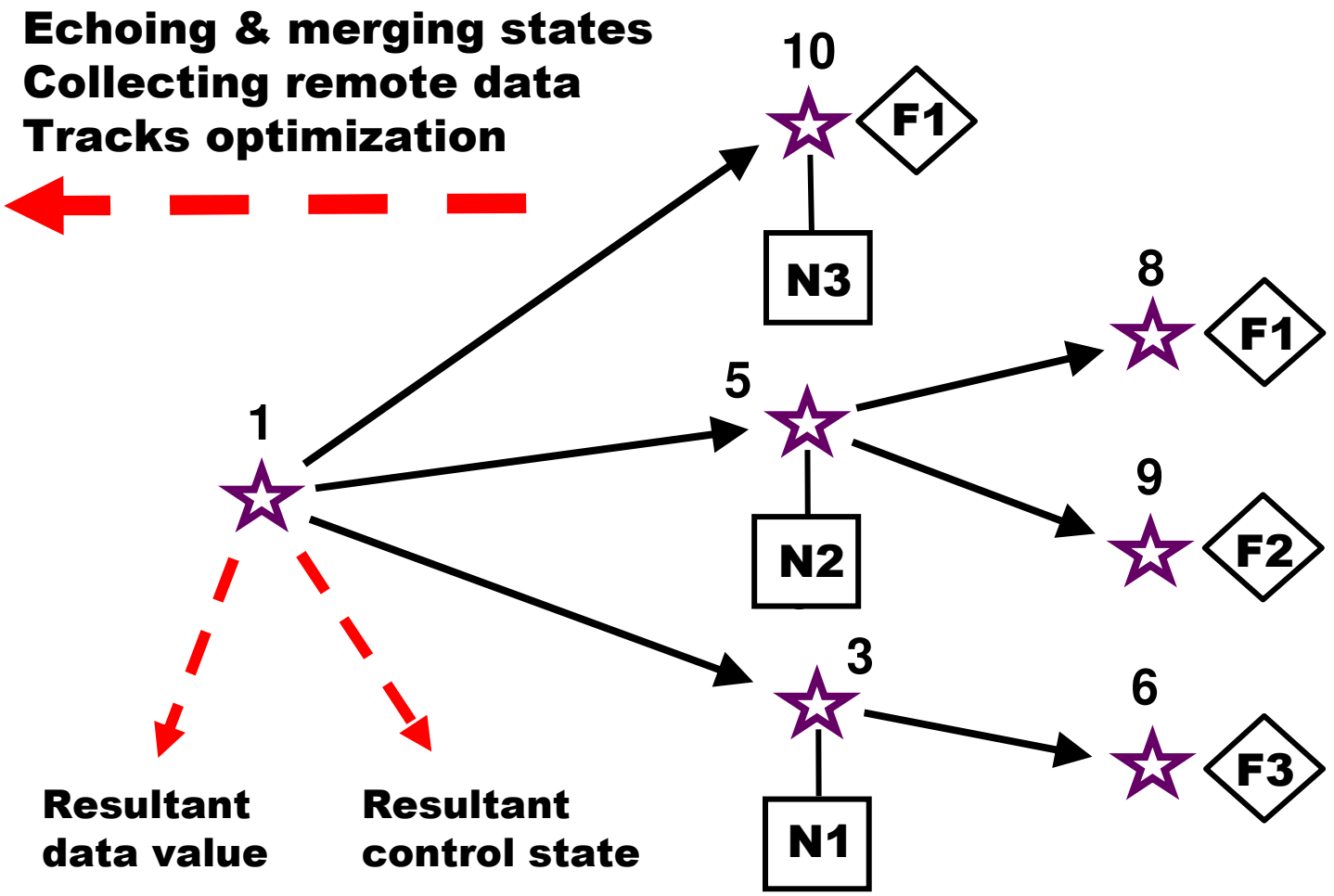
8.3 Distributed Track System

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.

8.3.1 Spatial Command with Tracks Creation



8.3.2 Spatial Control via Tracks



8.4 Embedding WPL Interpreters

8.4.1 WPL In Massively Wearable Devices

Mobile TV

Pocket PC-Phone

Phone With GPS

Satellite Phone

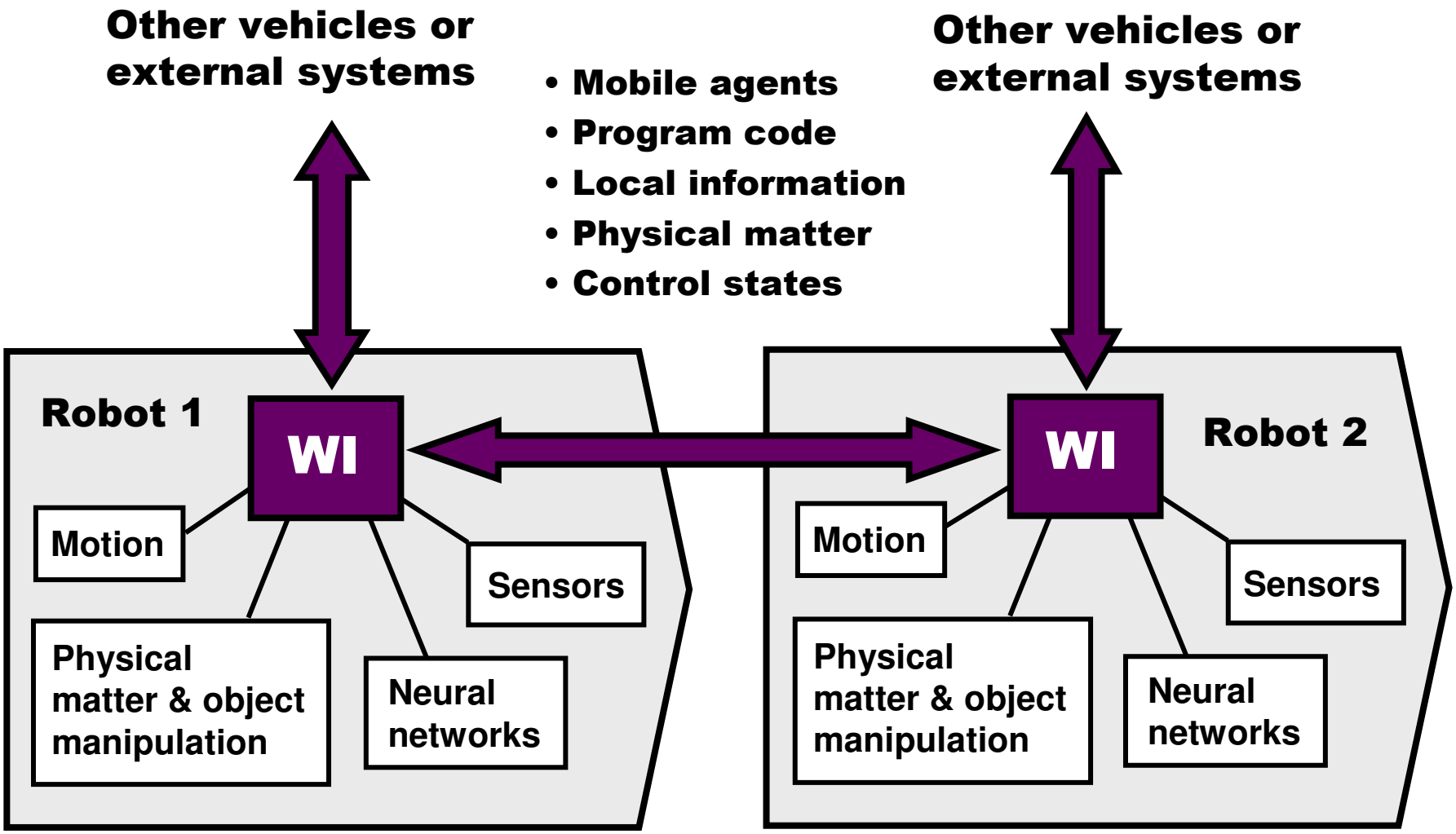
Sub-$100 laptop

Laptop in Your Hand

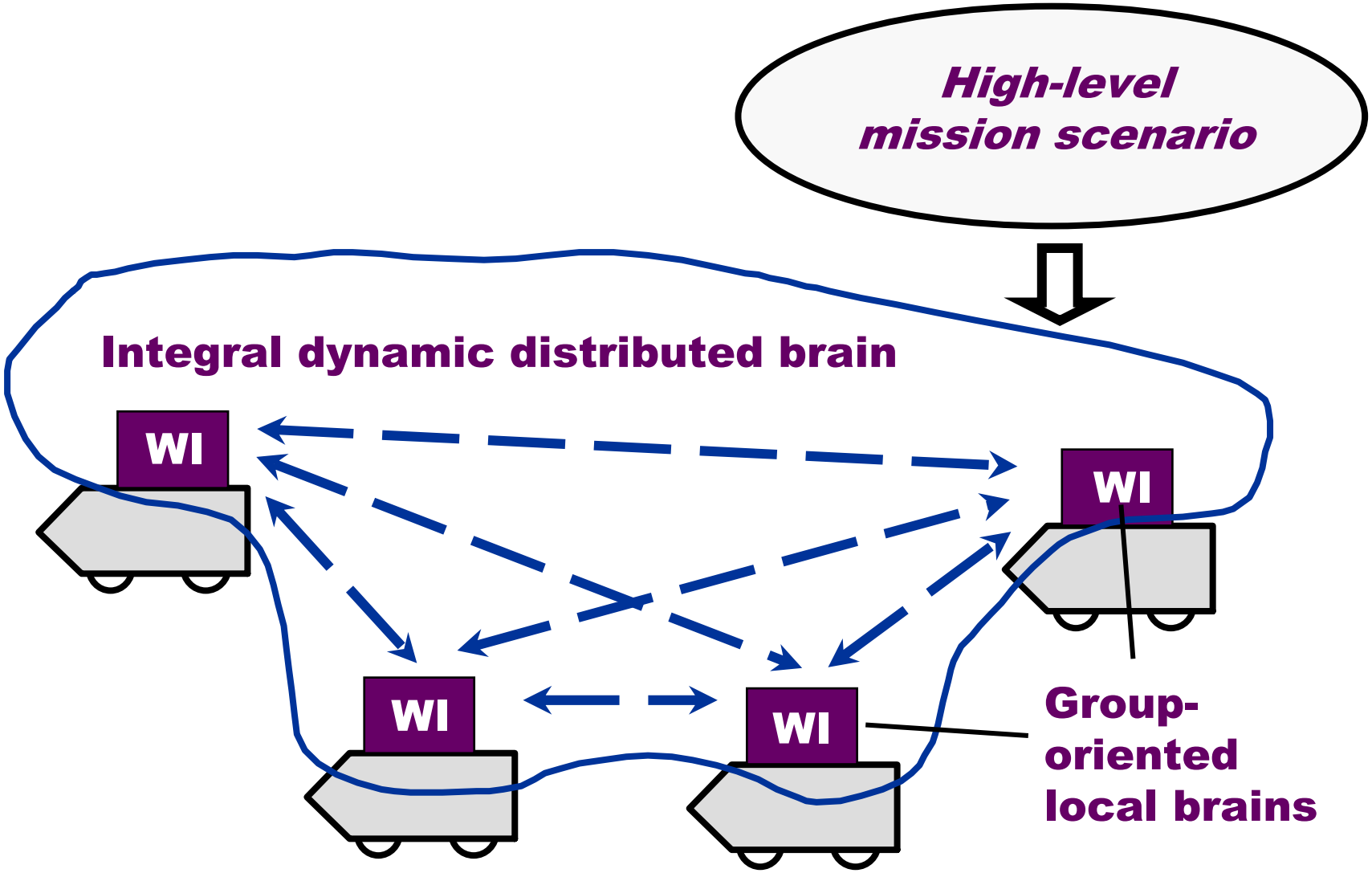
8.4.2 New Possibility: Direct Calls Between Phones

Swedish company TerraNet has developed the idea using peer-to-peer technology that enables users to speak on its handsets without the need for a mobile phone base station. The TerraNet technology works using handsets adapted to work as peers that can route data or calls for other phones in the network. Each handset has an effective range of about one kilometer. While individually the phones only have a maximum range of 1km, any phone in between two others can forward calls, allowing the distance to double. This principle applied many times creates a mini network.

8.4.3 Integration with Robotic Functionality



8.4.4 Distributed Robotic Brain



9 Electronic Warfare

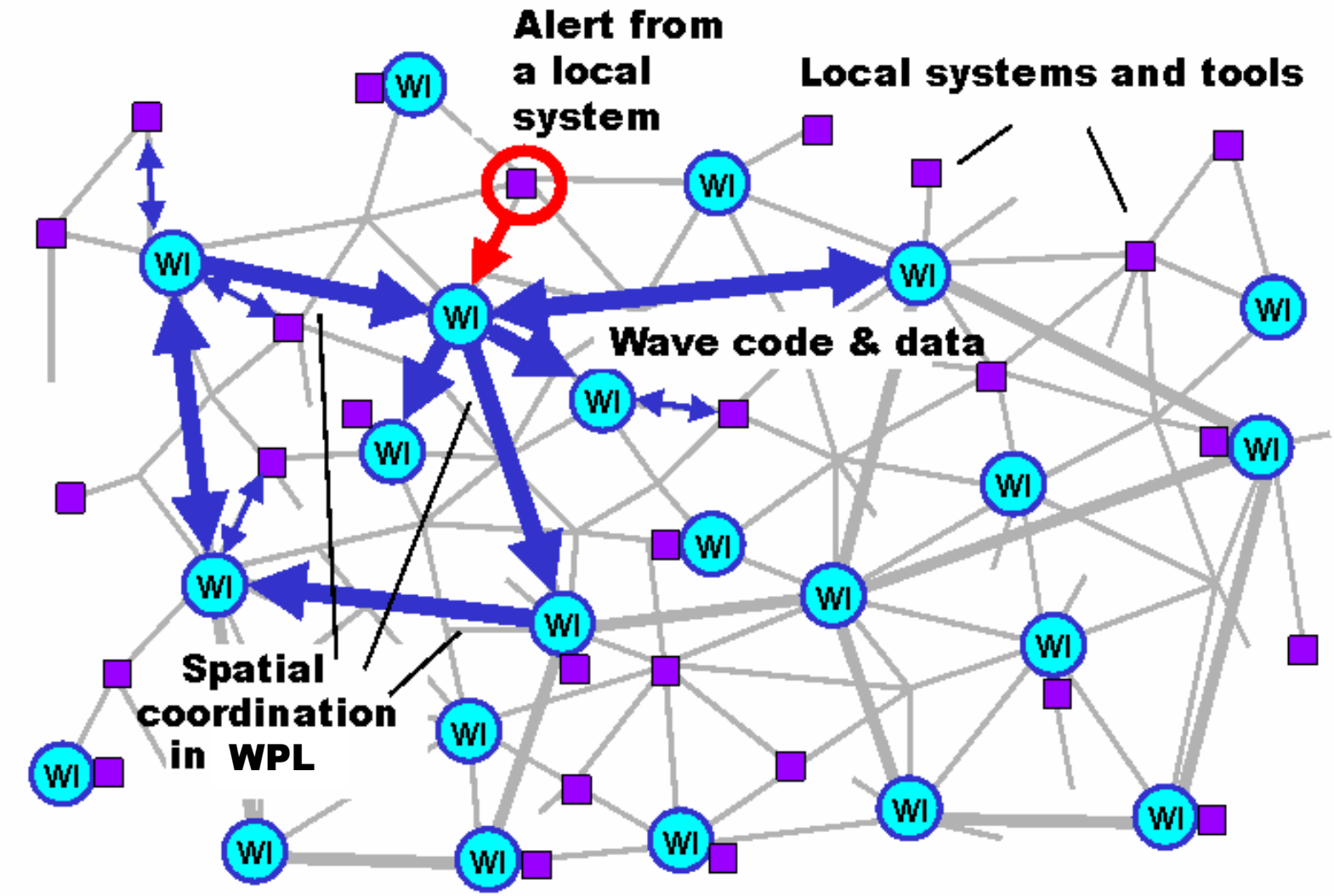
9.1 A Lockheed Plane Releasing Decoy Flares

9.2 Using Existing Distributed Systems Against Electronic Attacks

- International information systems
- Interpol
- Homeland security institutions and facilities
- Flight scheduling and control systems
- Smart sensor networks at airports
- Networked DEW systems
- Massive cooperative robotics with sentry duties

Any existing electronic support, attack, and protection measures may have limited scope and effect if used alone. But integrated under the WPT supervision they may solve complex nonlocal problems.

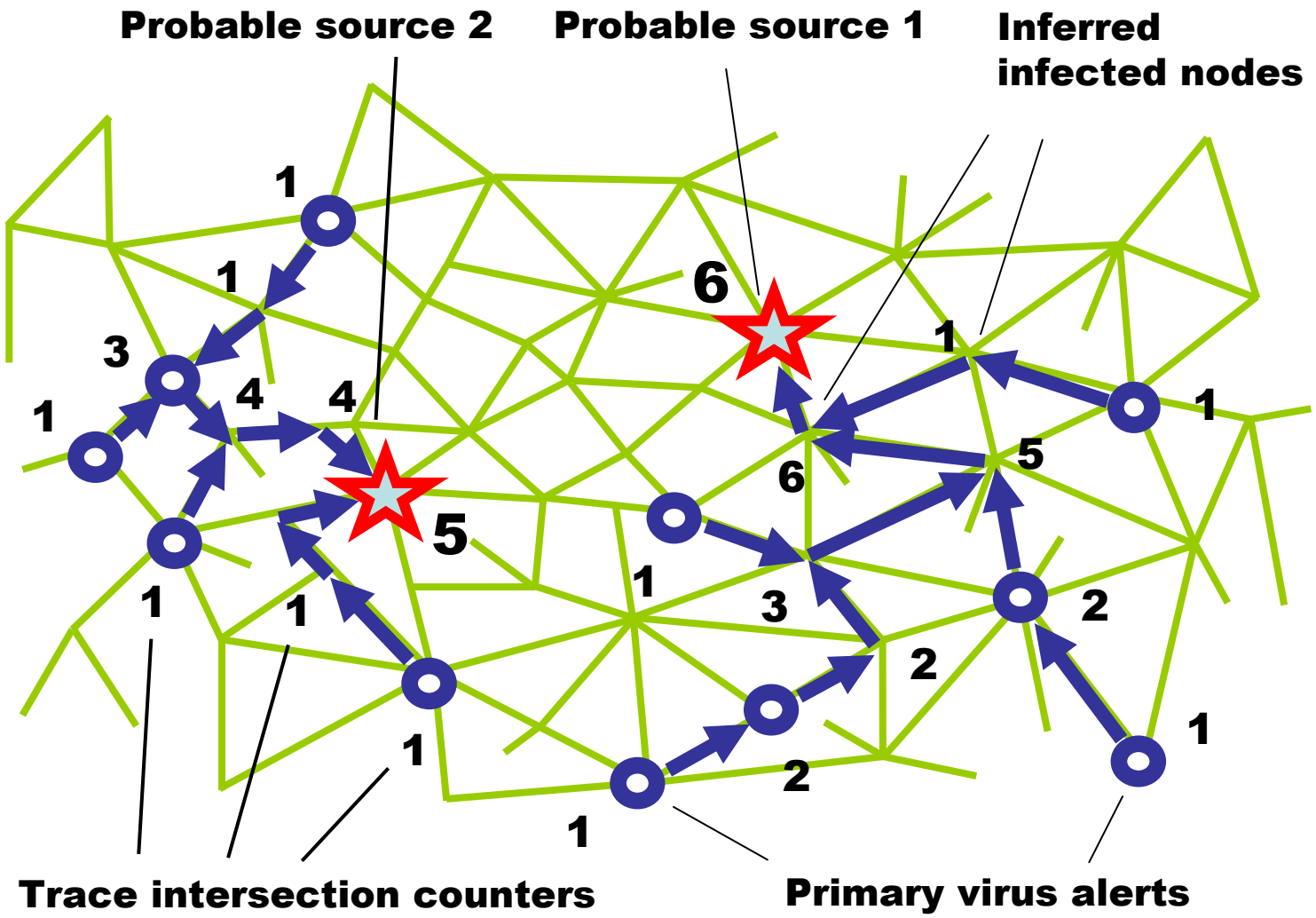
9.3 Higher Coordination Layer with WPL



9.4 Finding Virus Sources in Parallel

```
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)))
```

9.5 The Probable Sources Found



10 Smart Sensor Networks

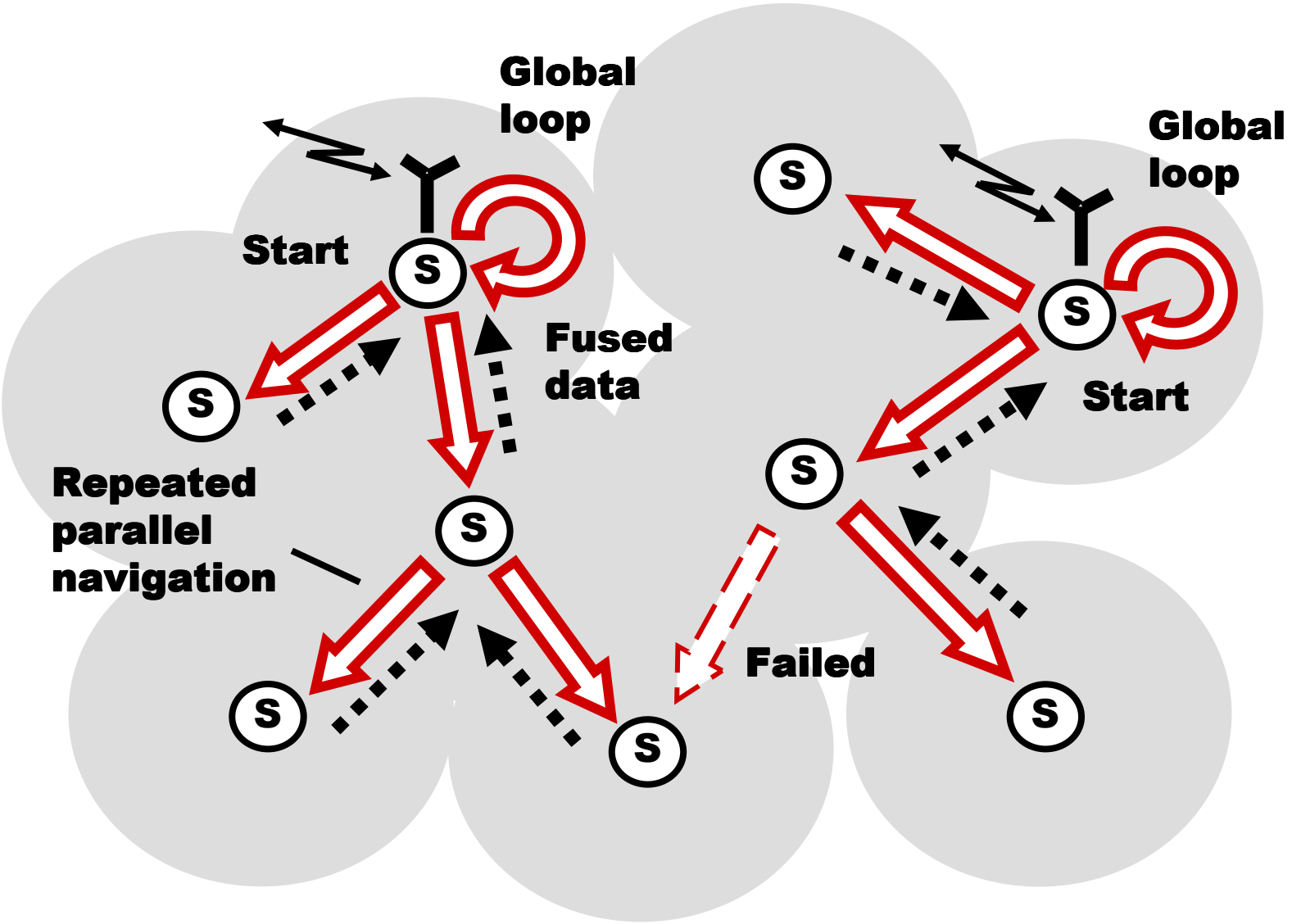
10.1 Collecting Events Throughout the Region

10.1.1 Distributed Collection Programming

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. Through this forest, by the internal interpretation infrastructure, the lifted data in nodes is moved and fused upwards the spanning trees, with final results collected in transmitter nodes and sent in parallel outside the system using rule Transmit.

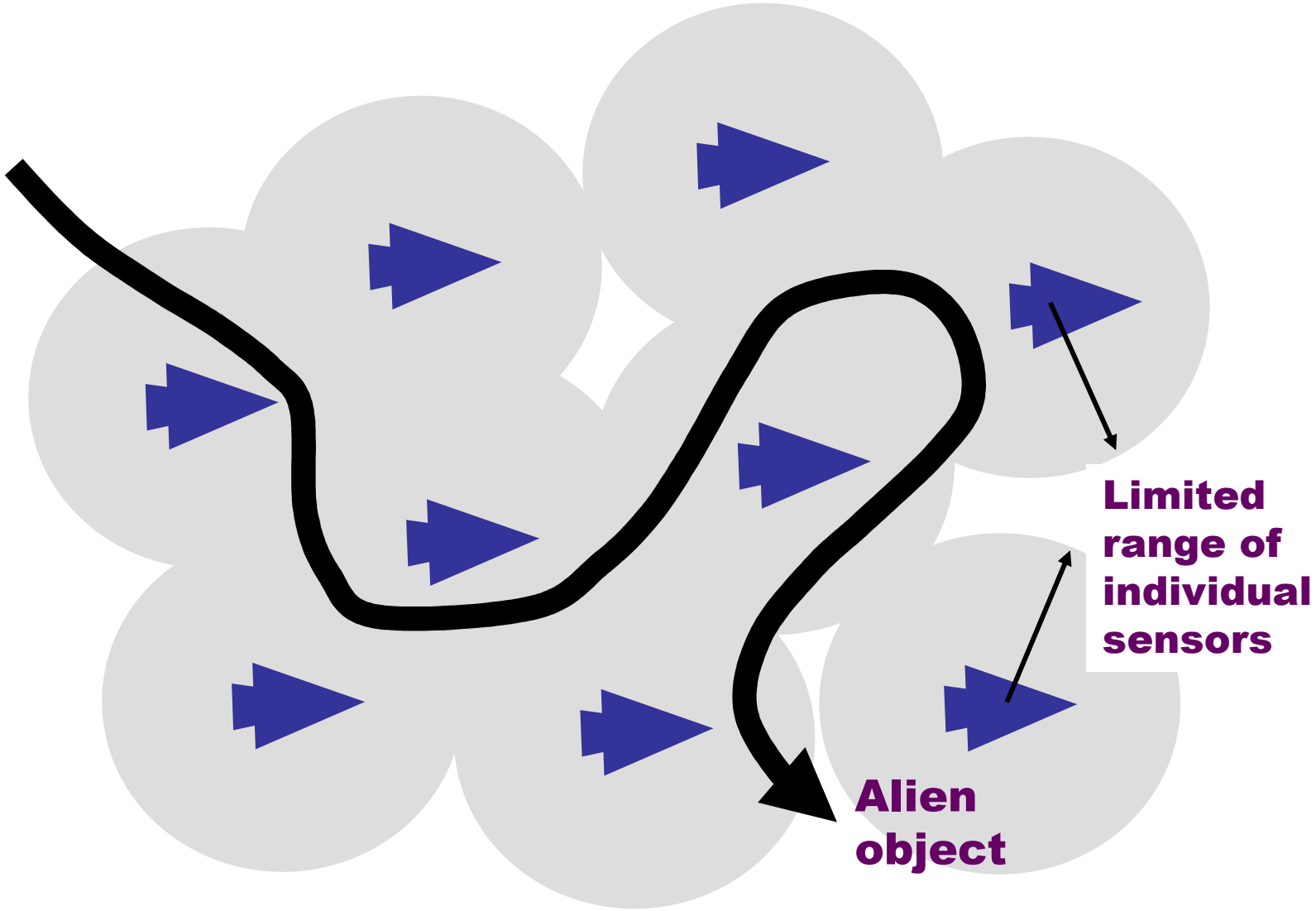
```
hop (all transmitters).
loop (
 sleep (20).
 IDENTITY = TIME.
 transmit (
  fuse (
   repeat (free (observe (events));
    hop (directly reachable, first come)))))
```

10.1.2 All Events Collected and Transmitted



10.2 Collective Tracking of Mobile Objects

10.2.1 Moving through Sensor Network



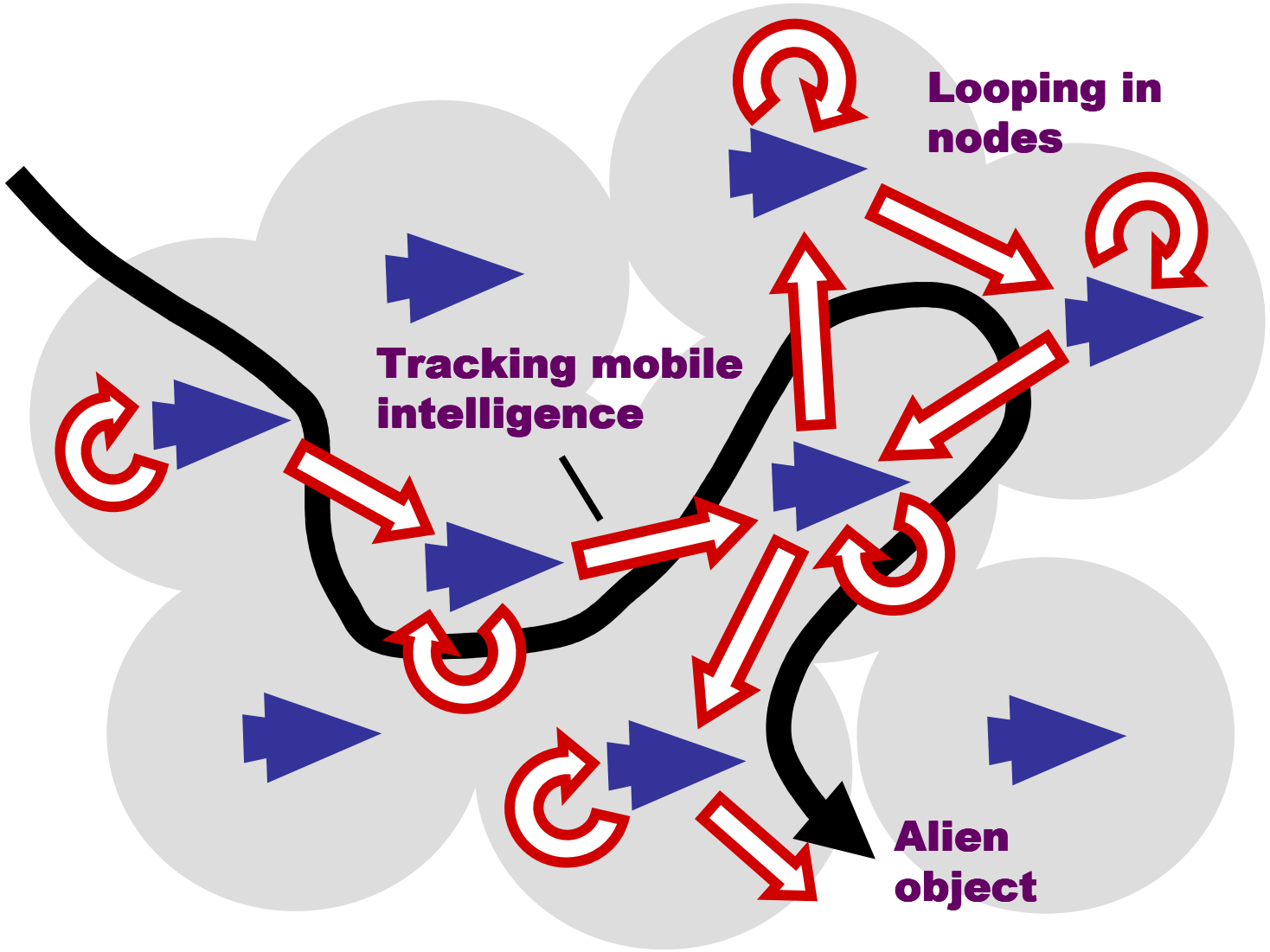
10.2.2 Tracking Conditions

- 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, while propagating between the WPL interpreters in UAVs.
- Many such objects can be picked up and chased in parallel.

10.2.3 The Tracking Program

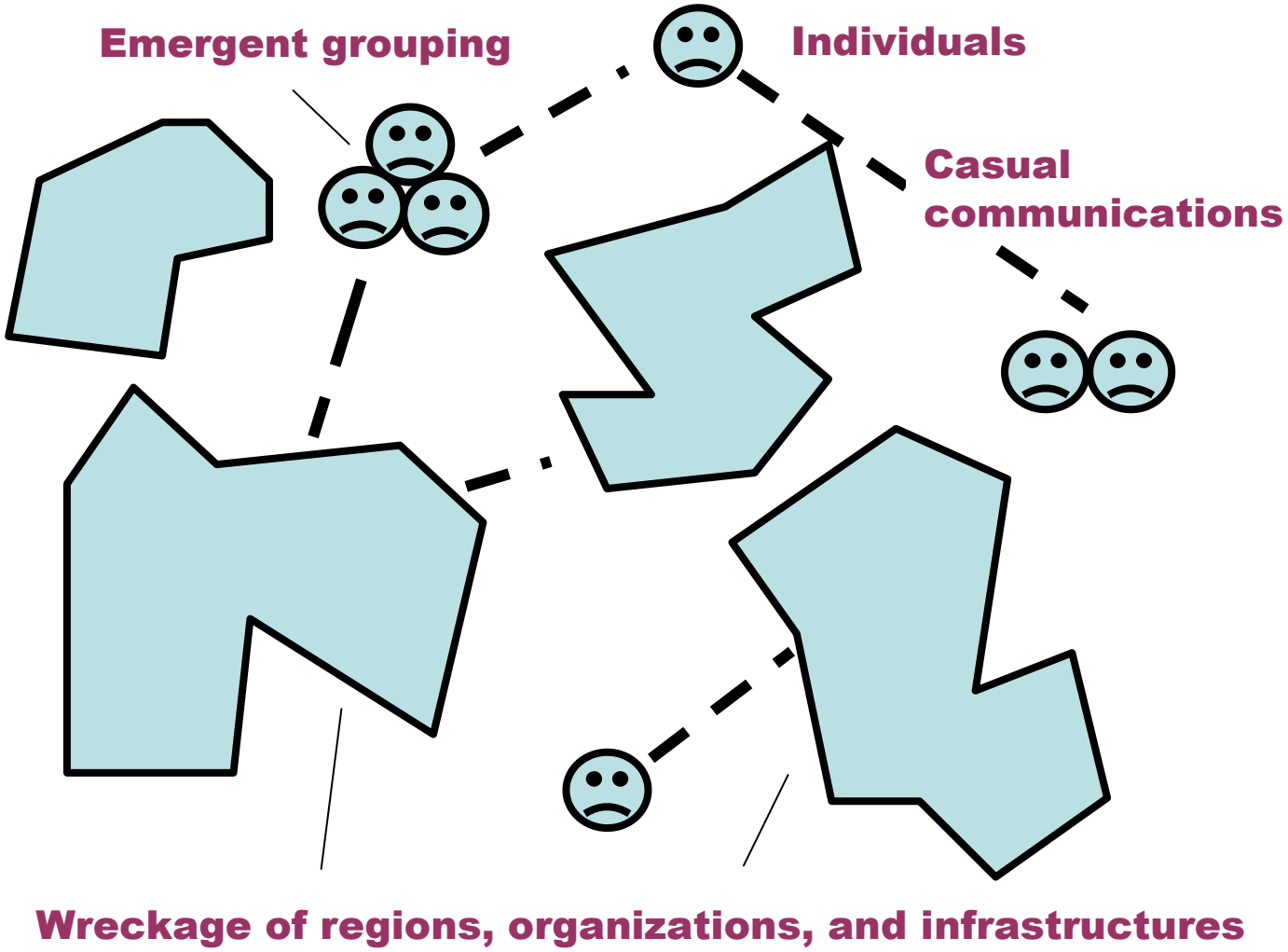
```
Hop all_nodes. Frontal Threshold = 0.1.
Frontal Object =
   Select_max_visible ( aerial, Threshold ).
Repeat (
   Loop ( visibility ( Object ) > Threshold ).
   Choose_destination_with_max_value (
      Hop all_neighbors.
      Visibility ( Object ) > Threshold ) )
```

10.2.4 The Tracking Dynamics

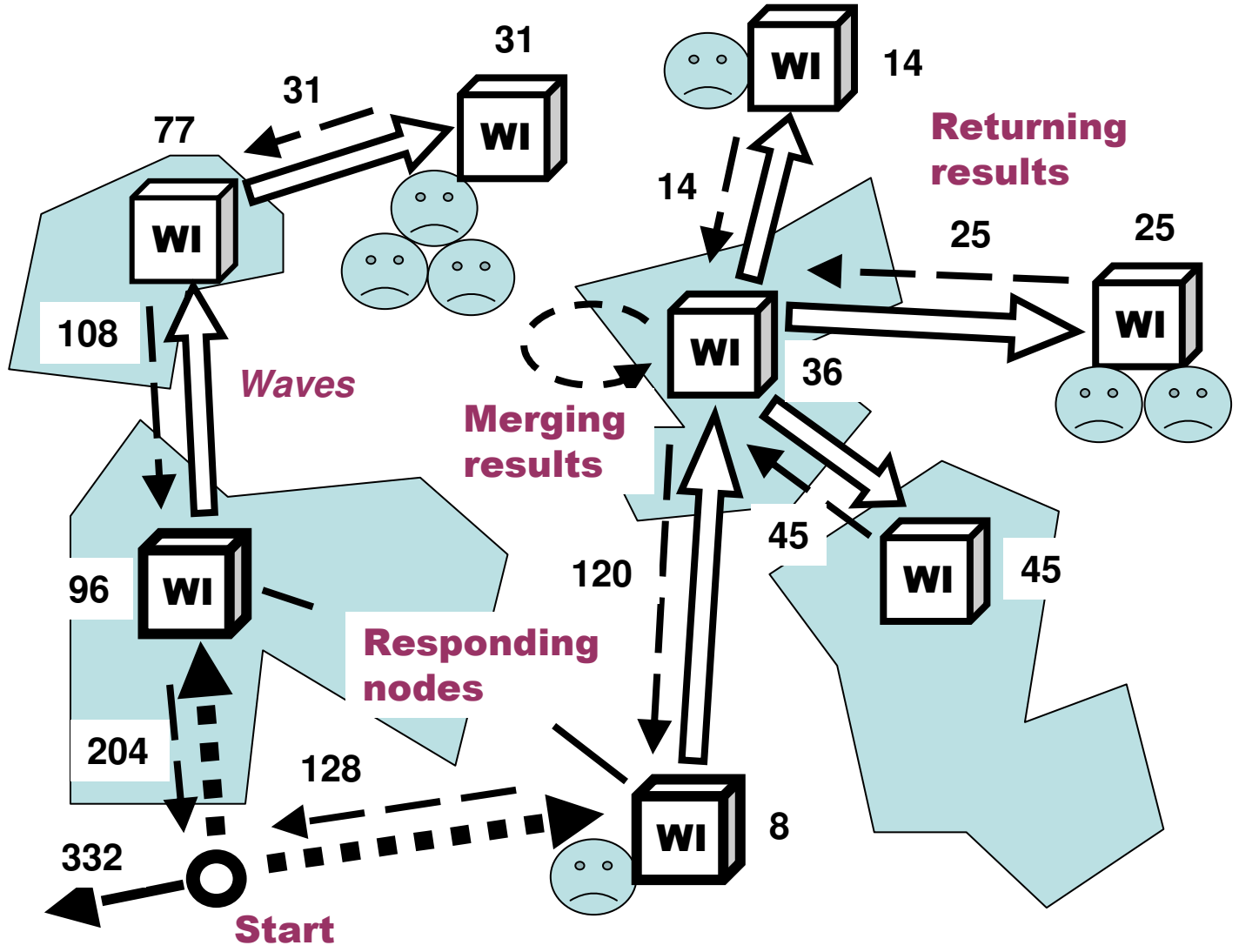


11 Emergency Management

11.1 Post-Disaster Picture



11.2 Spatial Counting of All Casualties



11.3 The Spatial Counting Program

```
frontal(Area);
external(count);
Area = <disaster area definition>;
output(
  sum(
    hop(firstcome, nodes(Area));
    repeat(
      done(count(casualties)),
      hop(anylinks, firstcome, nodes(Area)))))
```

11.4 Finding the Most Affected Region

```
frontal(Area);
nodal(Max);
external(count);
Area = <disaster area definition>;
Max = max(
  hop(directly, firstcome, nodes(Area));
  repeat(
    done(count(casualties) & WHERE),
    hop(anylinks, firstcome, nodes(Area))))
```

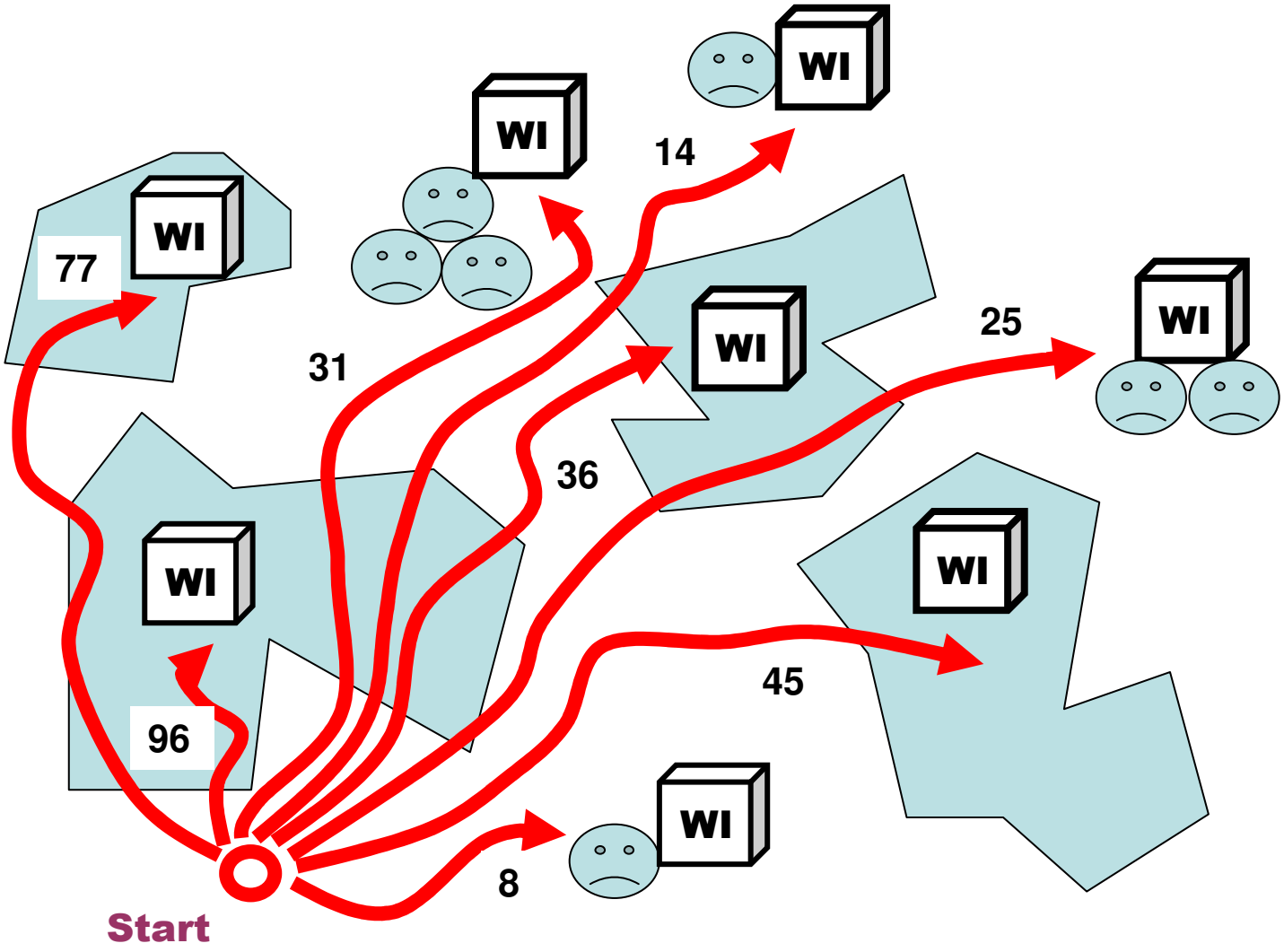
11.5 Relief Delivery to the Most Affected Region

```
frontal(Supply);
external(replicate, distribute);
Supply =
  replicate("package", element(Max, first));
hop(node(element(Max, second)));
distribute(Supply)
```

11.6 Finding and Delivery to All Regions

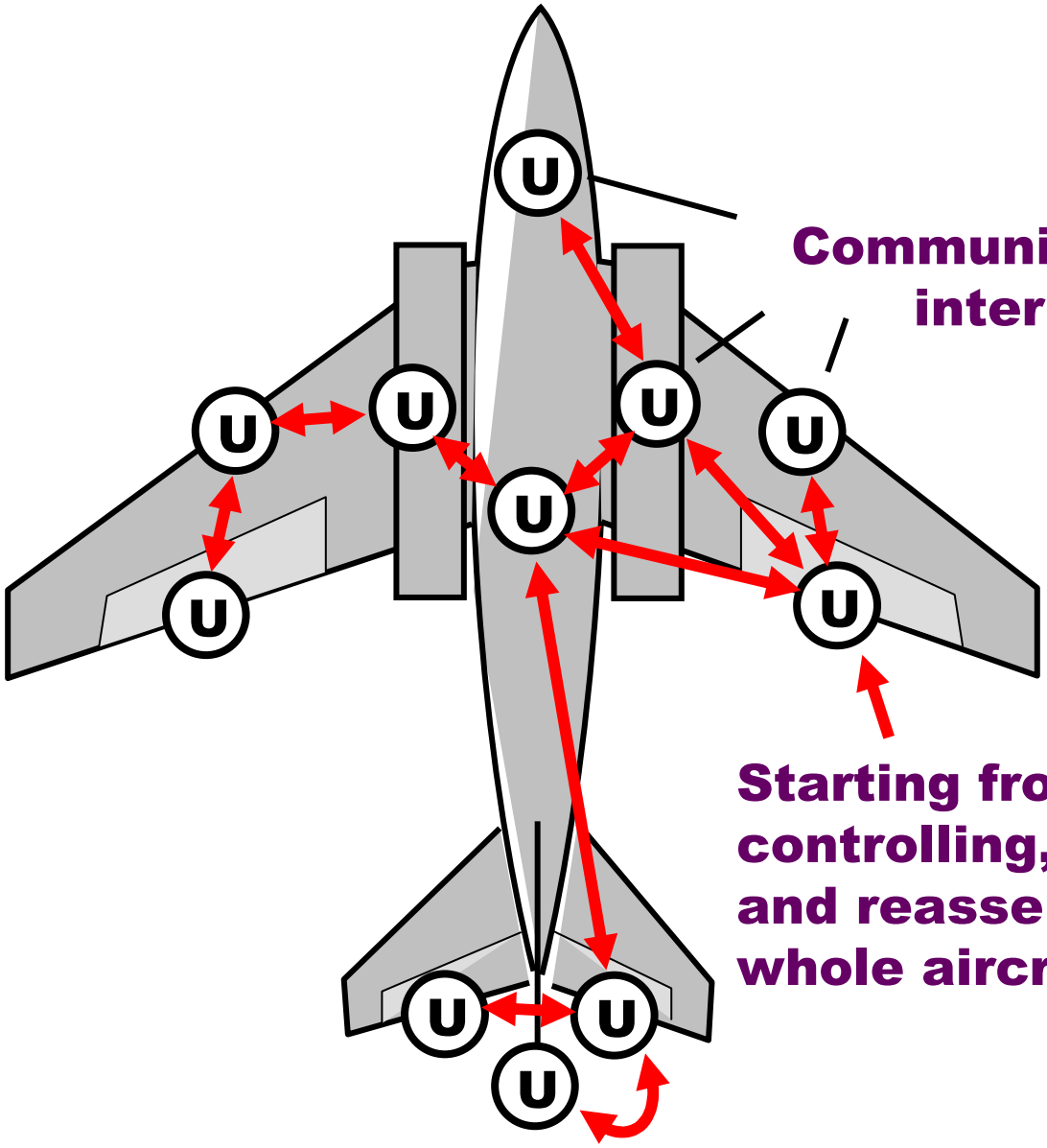
```
frontal(Area, Supply);
external(count, replicate, distribute);
Area = <disaster area definition>;
split(
  collect(
    hop(firstcome, nodes(Area));
    repeat(
      done(count(casualties) & WHERE),
      hop(anylinks, firstcome, node(Area))
    )
  )
);
Supply =
  replicate(“package”, element(VALUE, first));
hop(node(element(VALUE, second)));
distribute(Supply)
```

11.7 The Delivery to All Regions



12 Distributed Avionics

12.1 Distributed Survivability System



12. 2 Collecting Availability of Aircraft's Basic Mechanisms

(Starting from any node)

Sufficient = {...}; Control_with = {...}.

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),

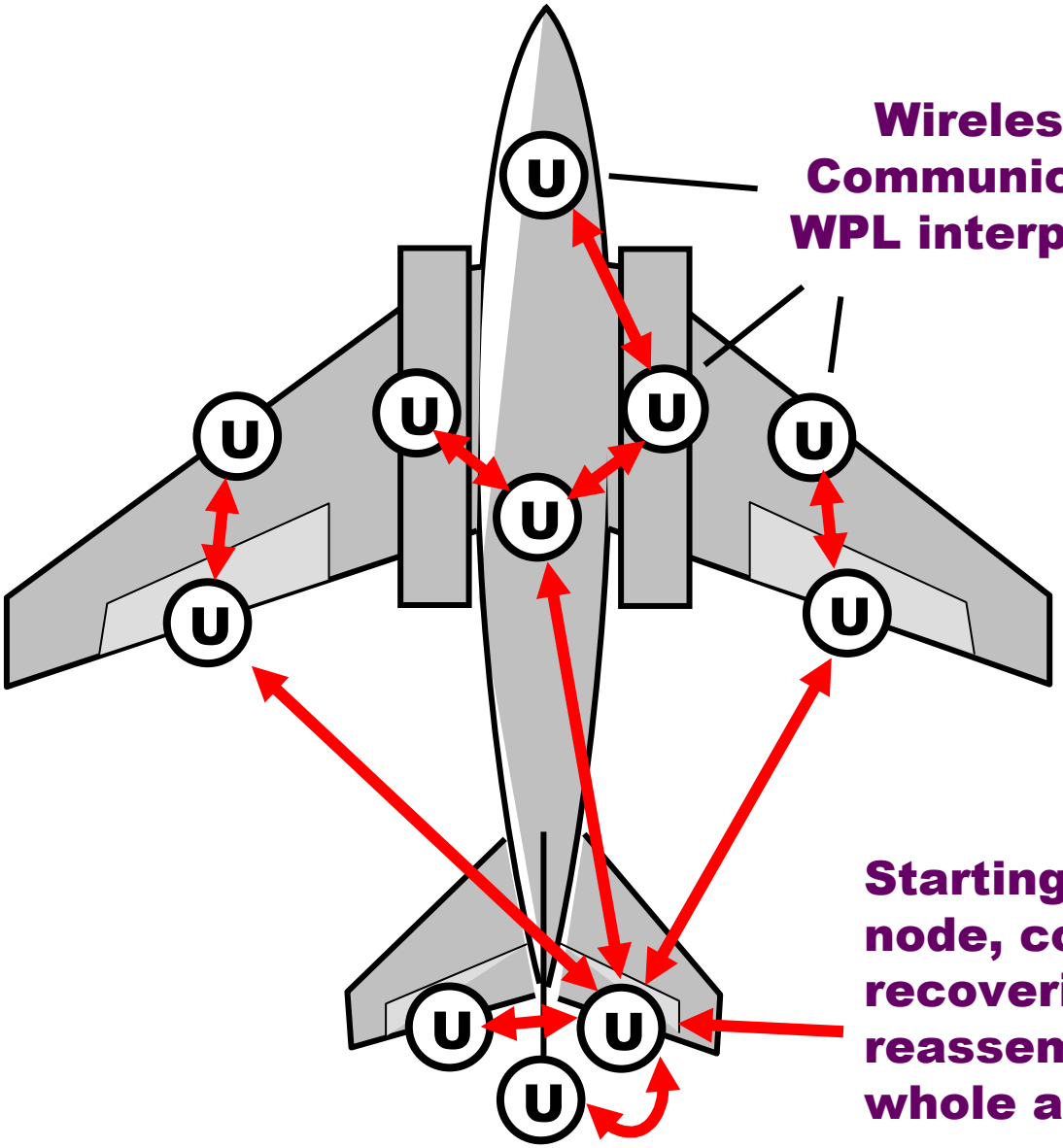
Hop_first all_neighbors).

If Sufficient Available_Set

Then Control_with Available_Set

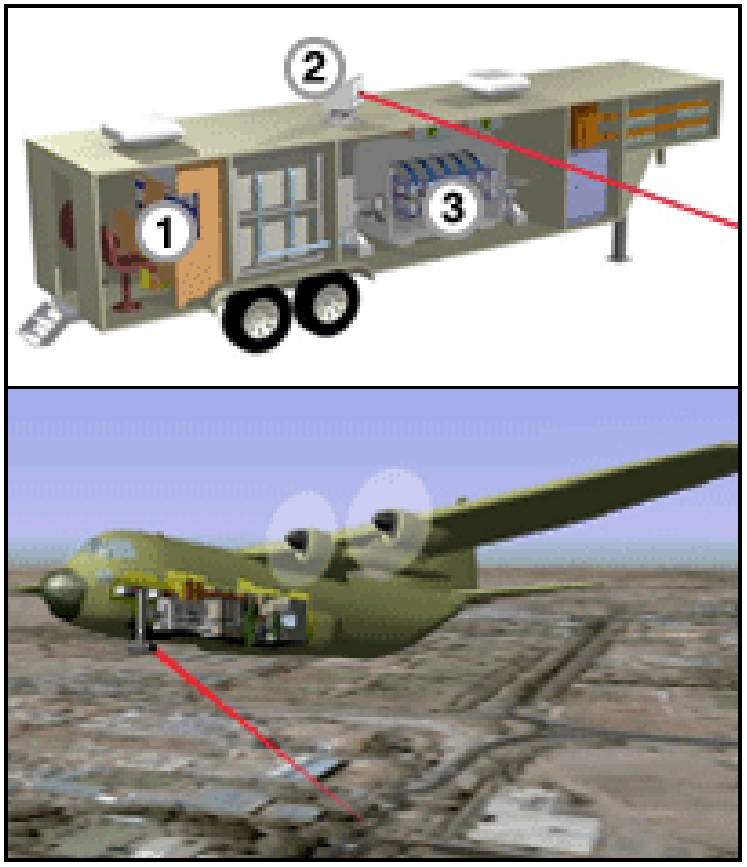
Otherwise alarm

12.3 Increasing Flexibility and Robustness by Wireless Communications

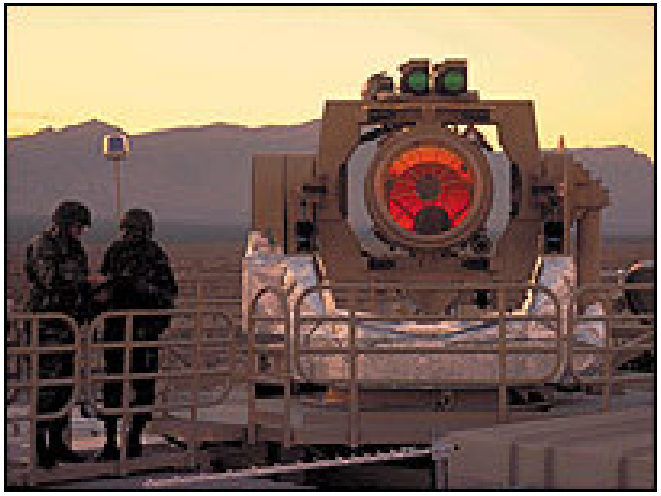


13 Directed Energy Systems

13.1 Laser Examples



1 - Control room 2 - Laser
3 - Power system

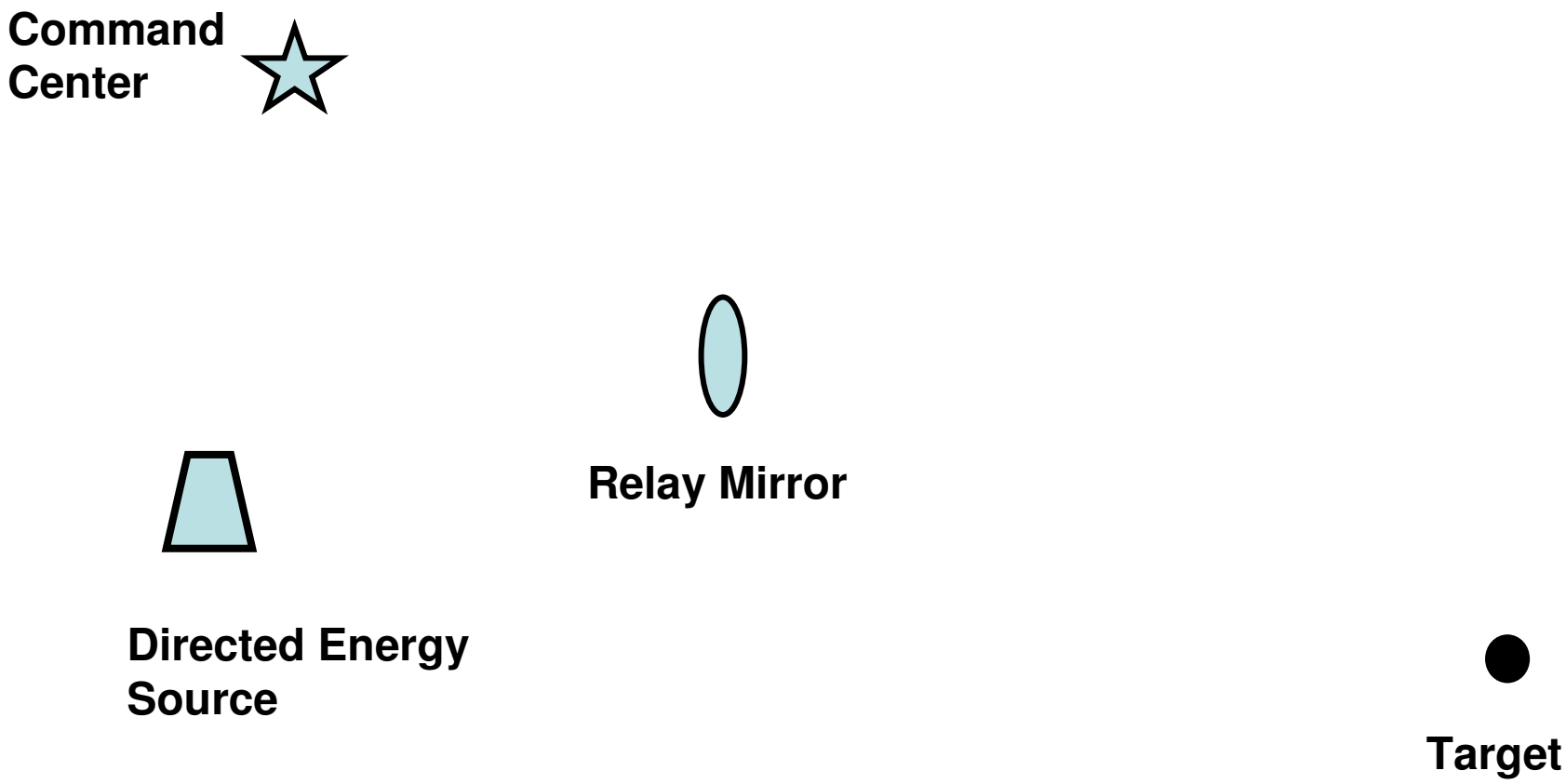


There may be many applications
for high power lasers

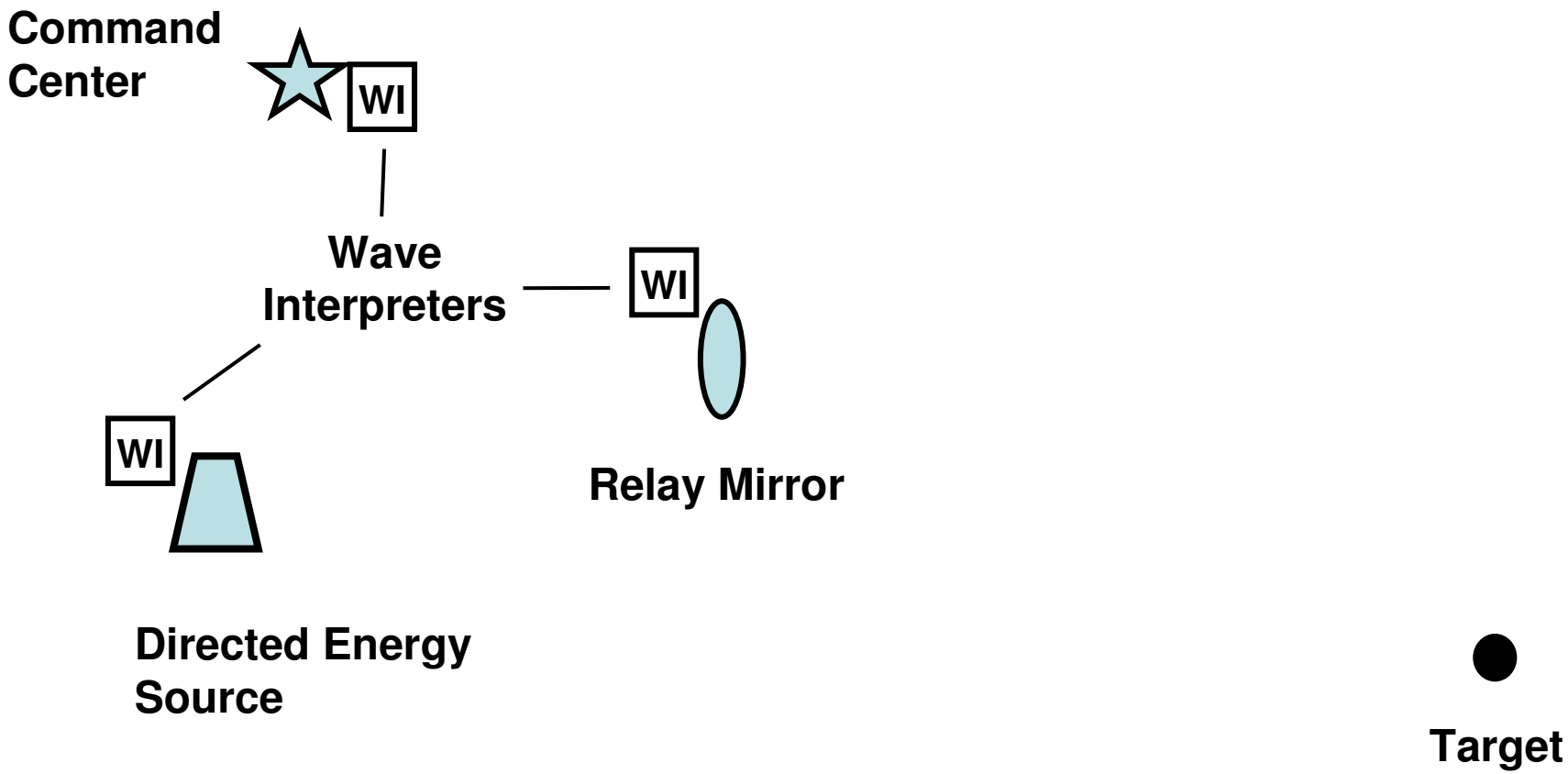
13.2 Elementary DE-Based System

An elementary distributed system consisting of control center, DE source, relay mirror, and target can be programmed on the fly, depending on the situations occurred.

13.2.1 Elementary System Components



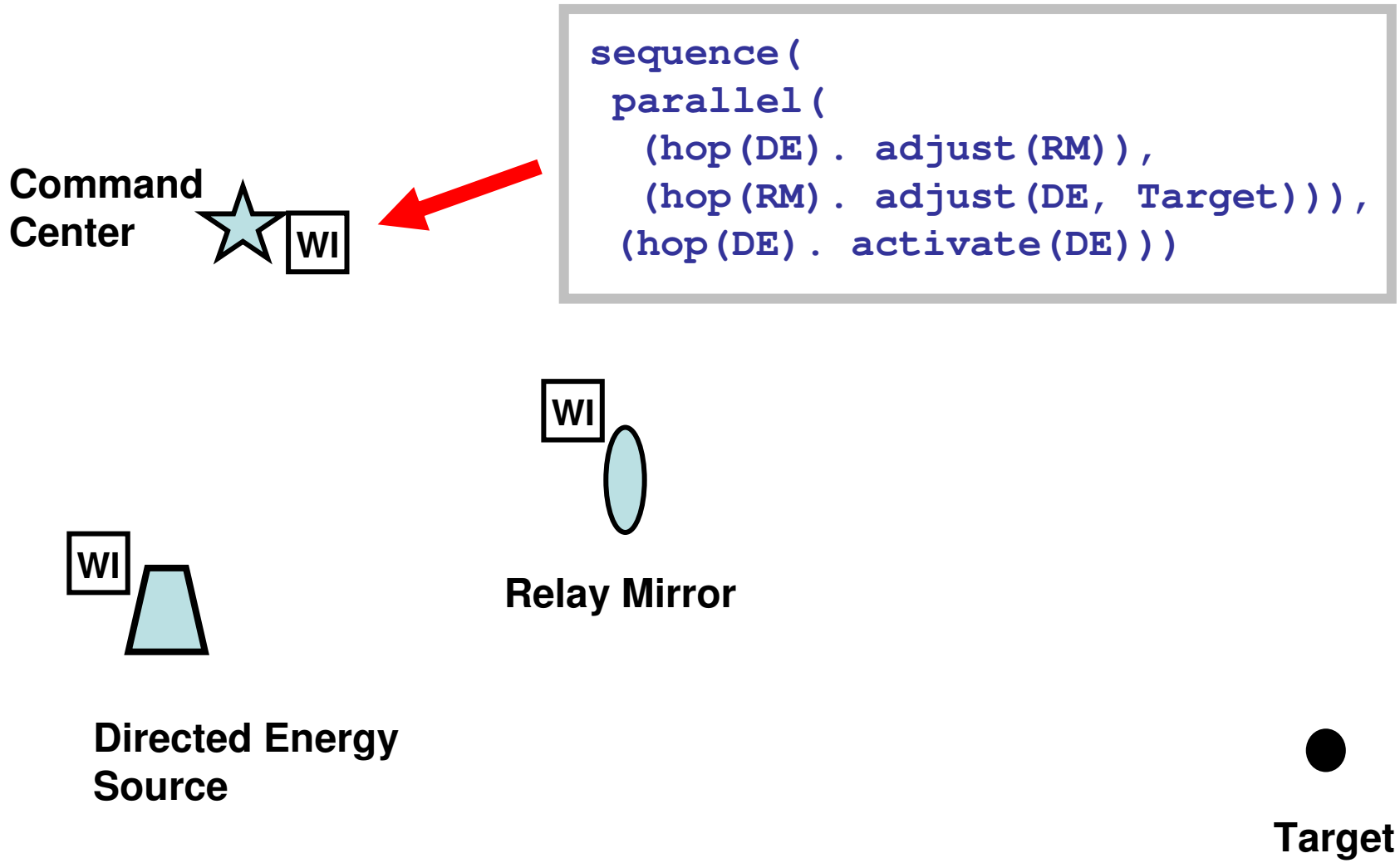
13.2.2 Embedding WPL Interpreters



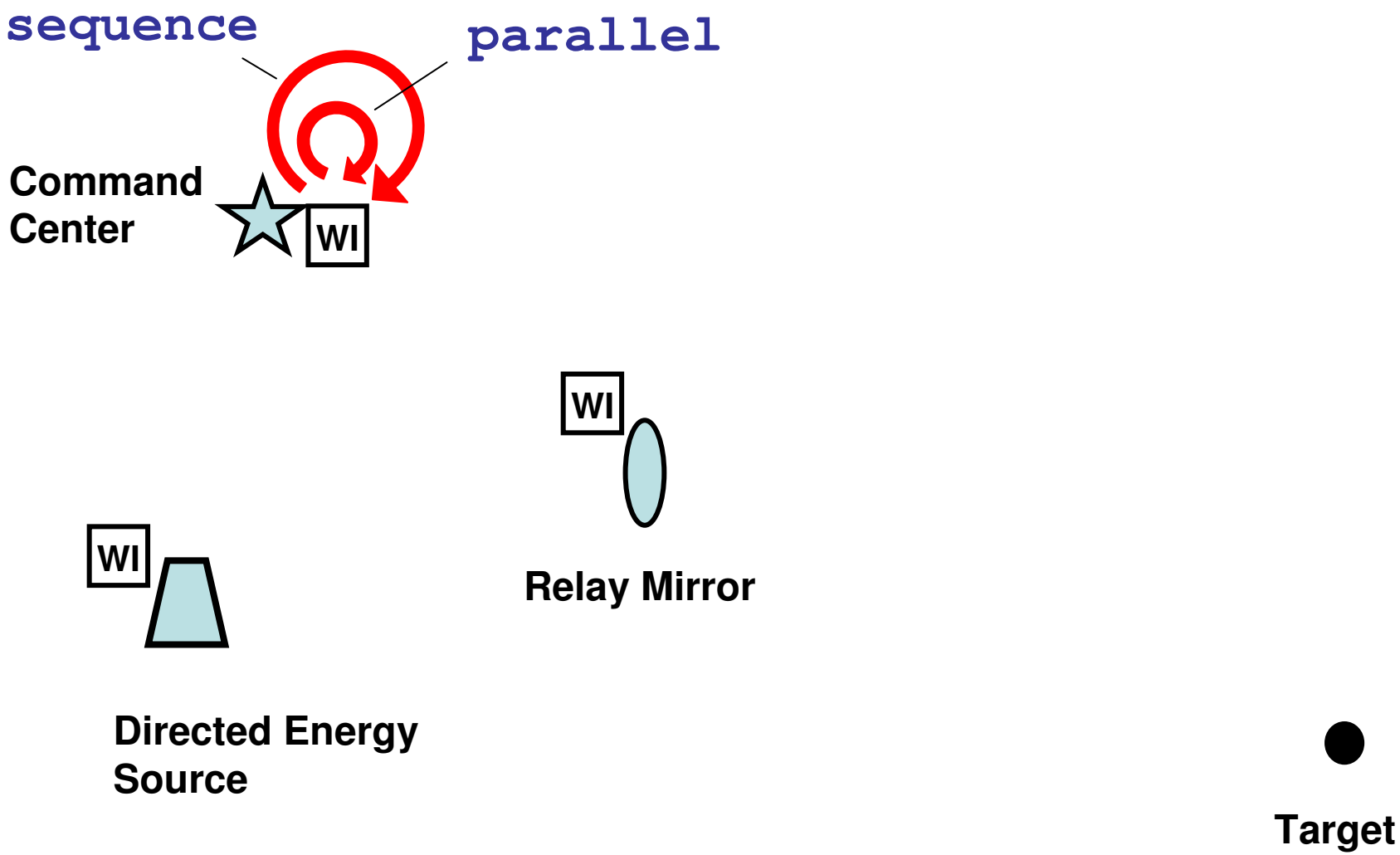
13.2.3 DE-RM-Target Scenario Programming

```
sequence(
  parallel(
     (hop(DE). adjust(RM)),
     (hop(RM). adjust(DE, Target))
  ),
   (hop(DE). activate(DE))
)
```

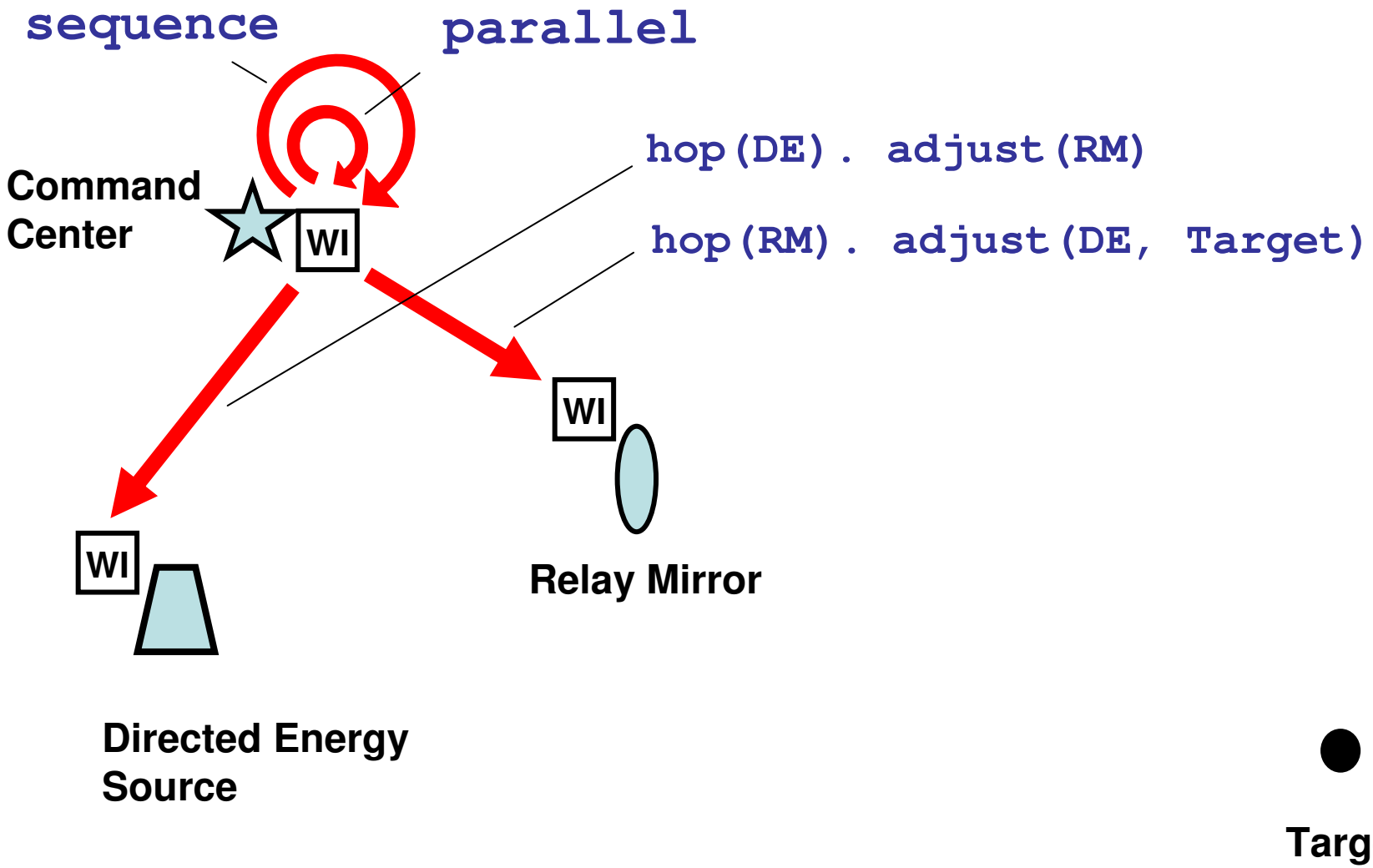
13.2.4 DE-RM-Target Scenario: Start



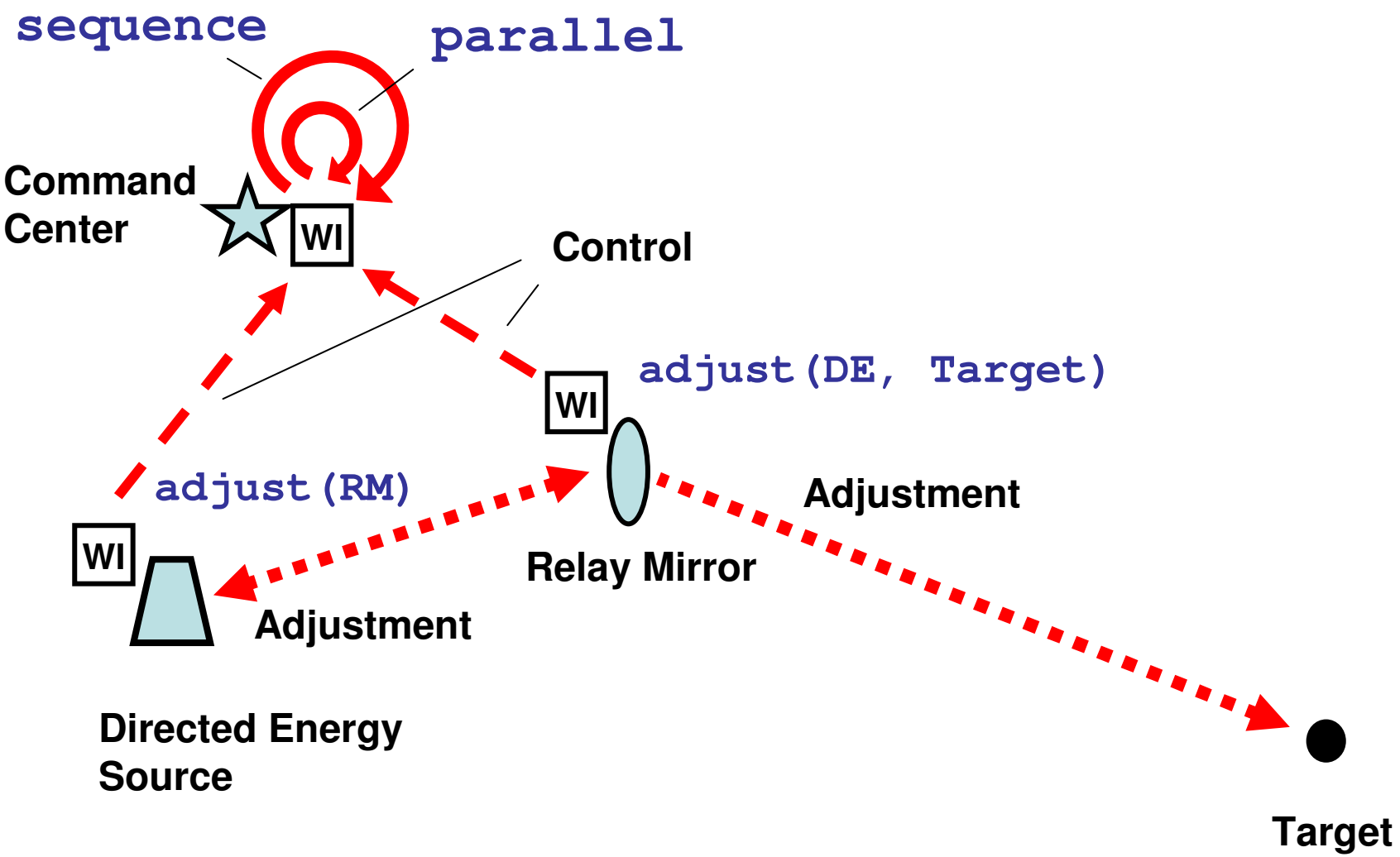
13.2.5 DE-RM-Target Scenario: Top C2 Formation



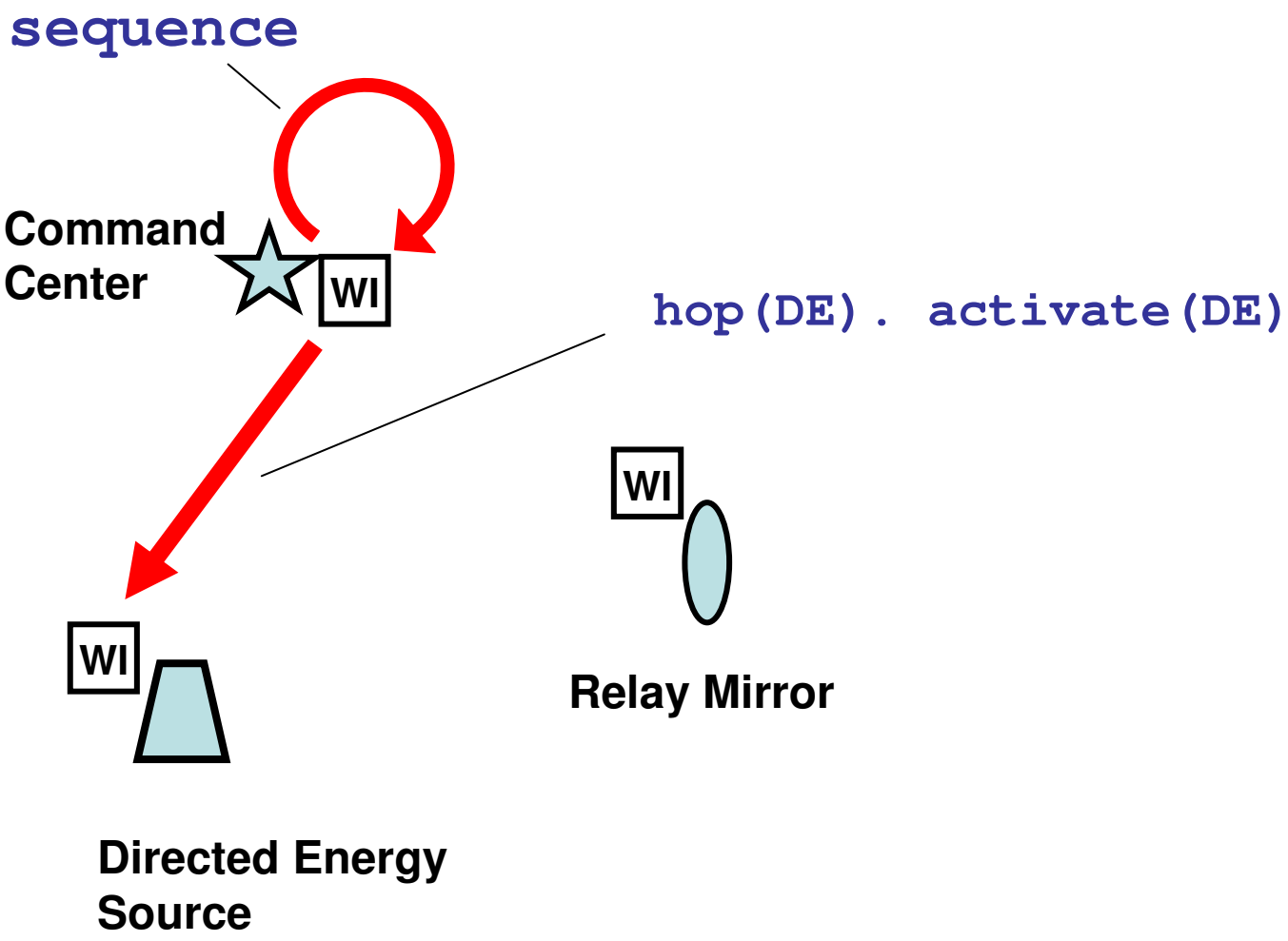
13.2.6 Parallel Tasking 1



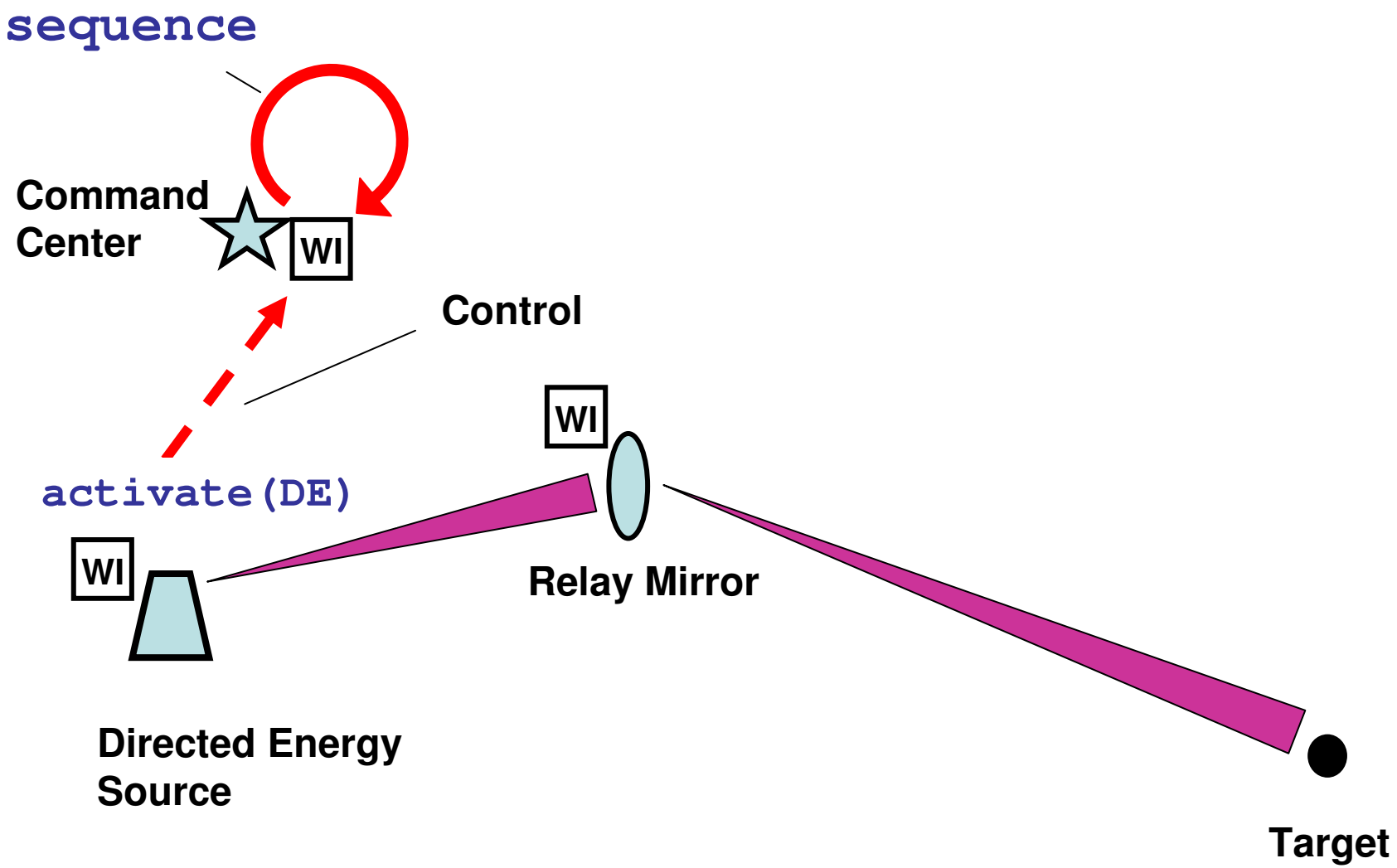
13.2.7 Parallel Adjustment



13.2.8 DE Tasking 2



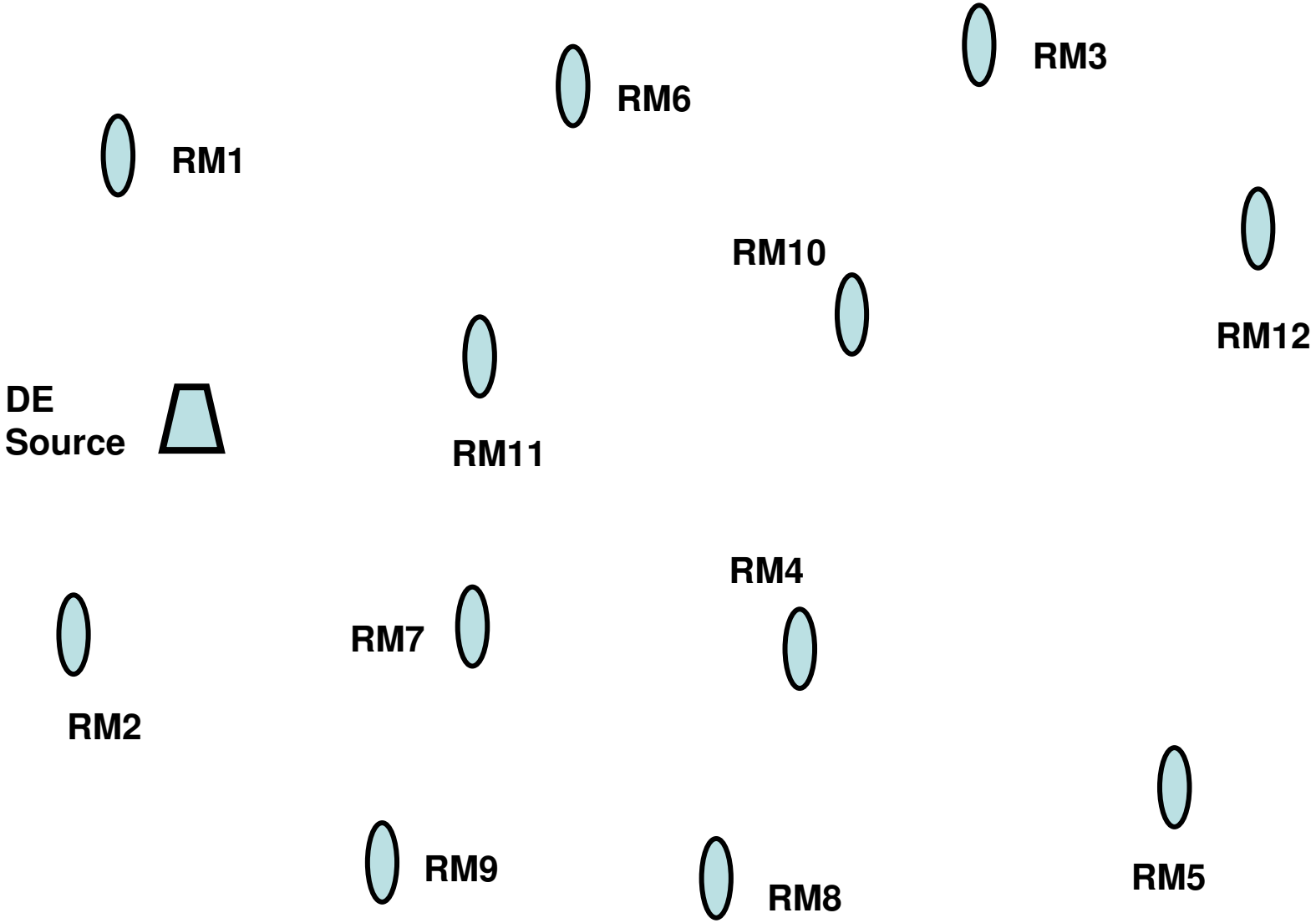
13.2.9 DE Activation



13.3 The ARMS Concept

Boeing's Advanced Relay Mirror System (ARMS) concept plans to entail a constellation of as many as two dozen 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.

13.3.1 Multiple Relay Mirror System



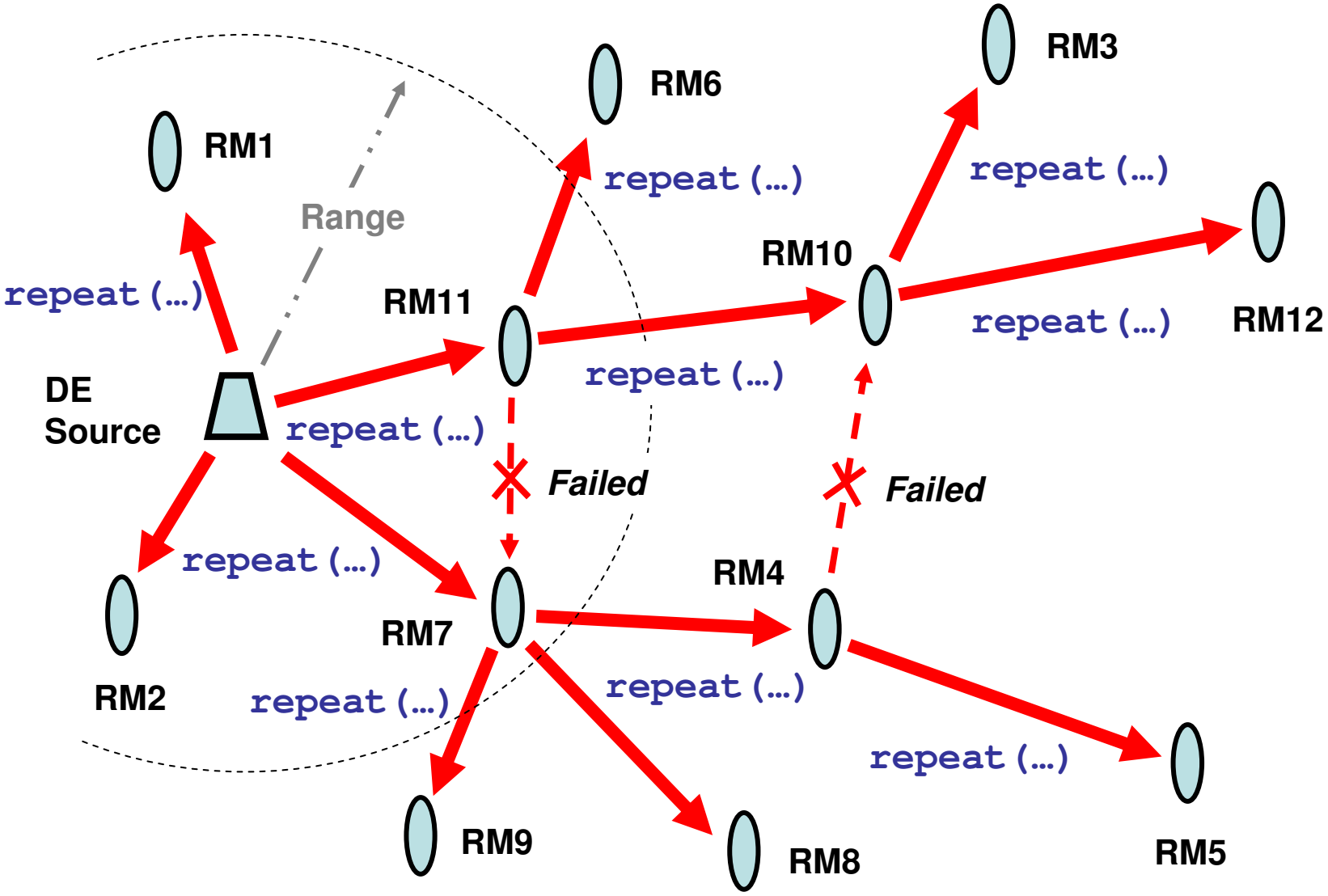
13.3.2 The Runtime SPT

We will consider here 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, and to any target discovered.

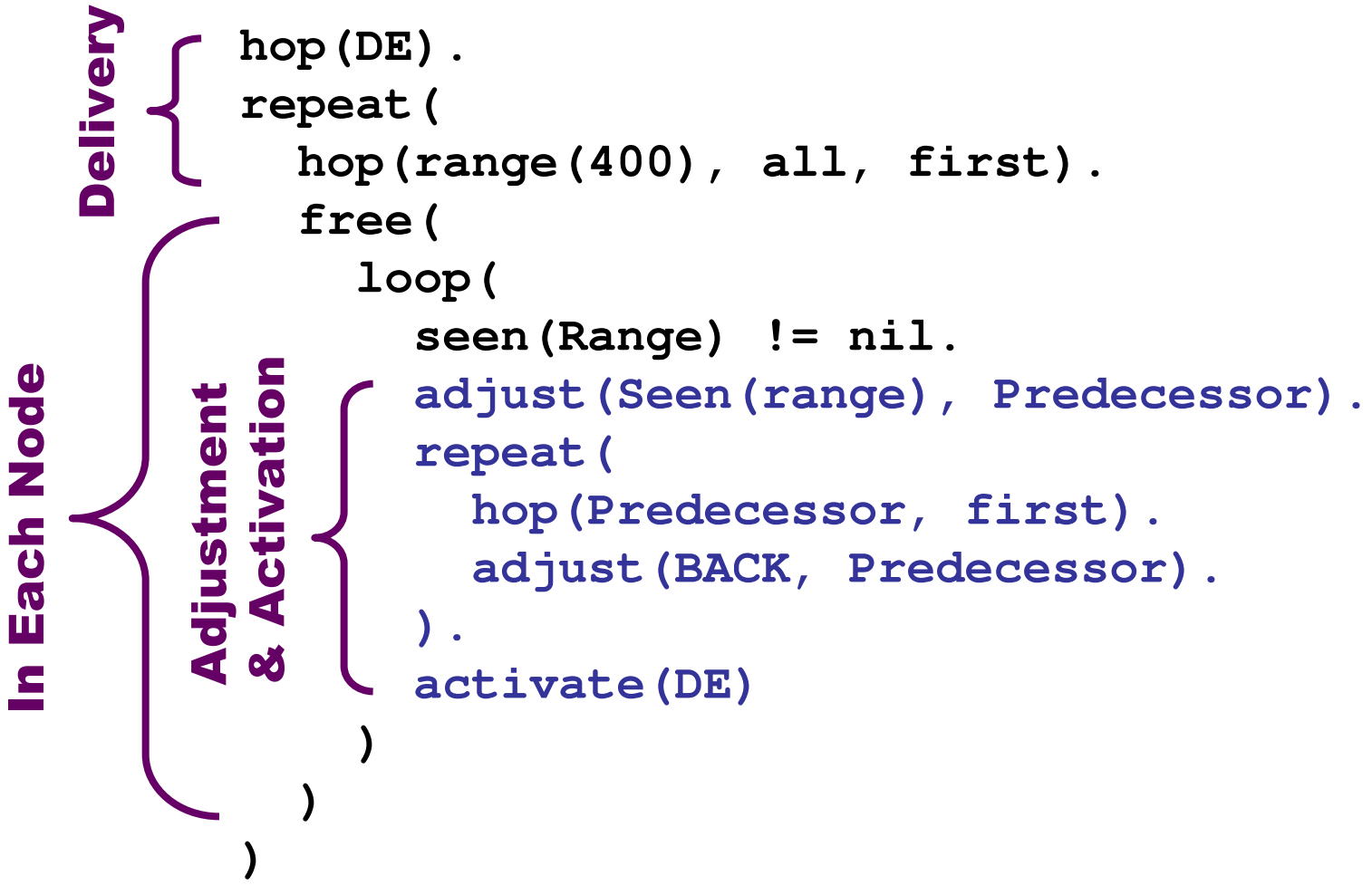
13.3.3 Creating Shortest Path Tree

```
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
)
```

13.3.4 Shortest Path Tree through All RM



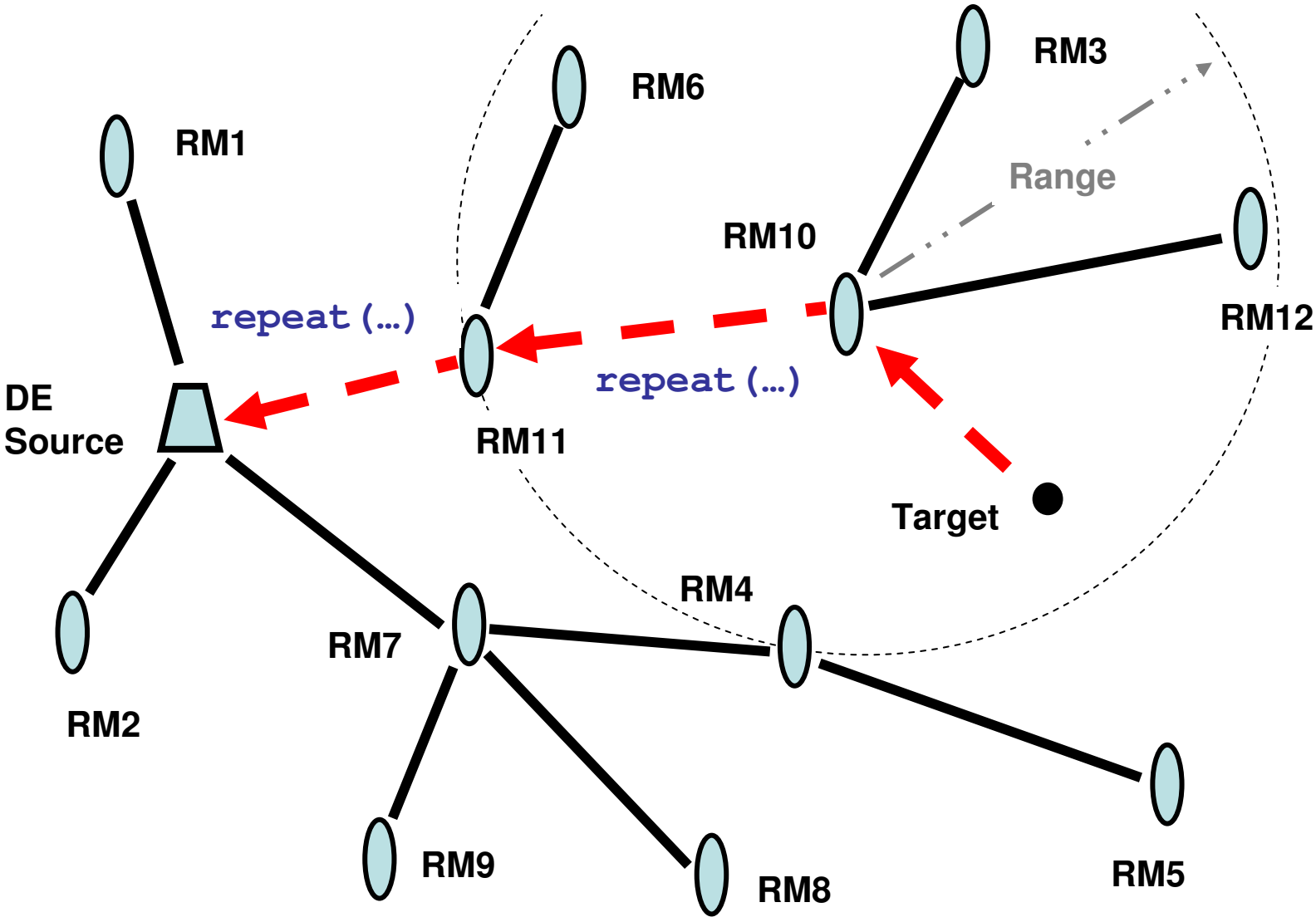
13.3.5 Target Discovery, Path Formation, Firing



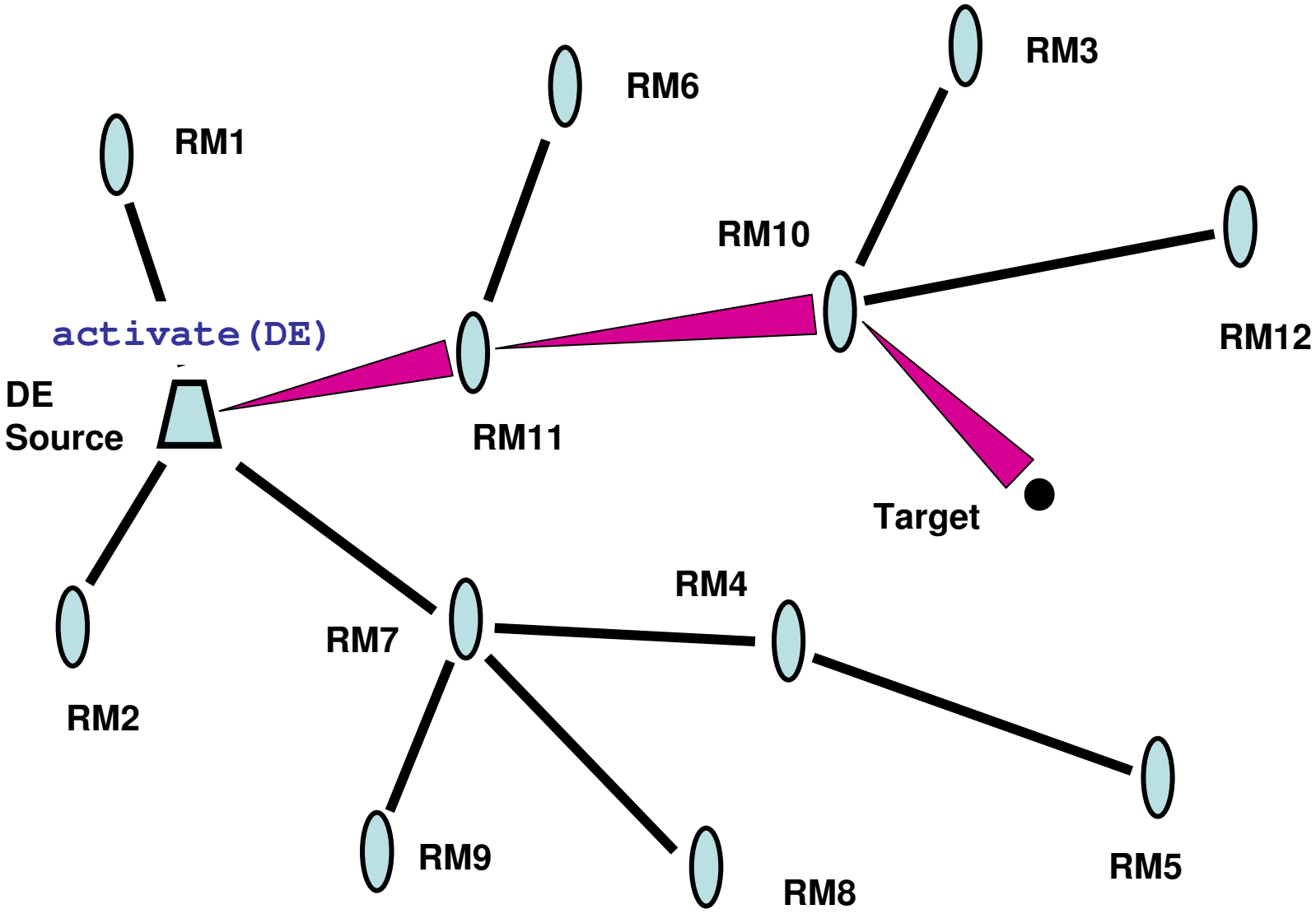
13.3.6 Path Formation & Firing Code

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

13.3.7 Target Discovery, Path Formation



13.3.8 DE Activation

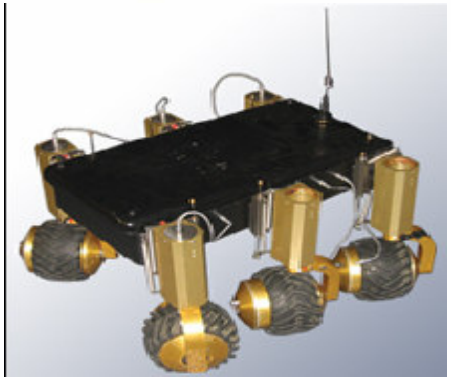


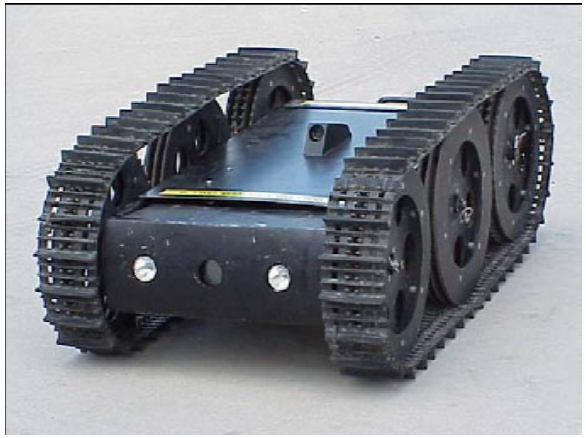
14 Robotics

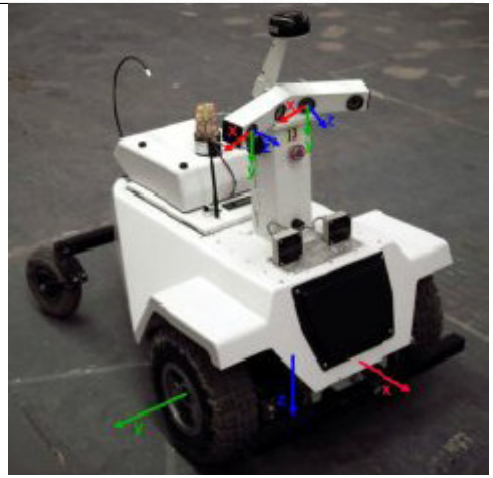
14.1 Robotics Examples

14.1.1 Robotic Examples 1

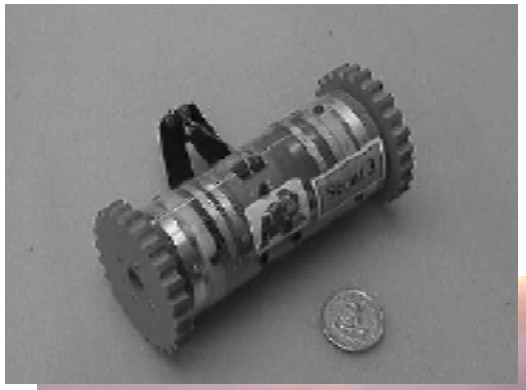
14.1.2 Robotic Examples 2



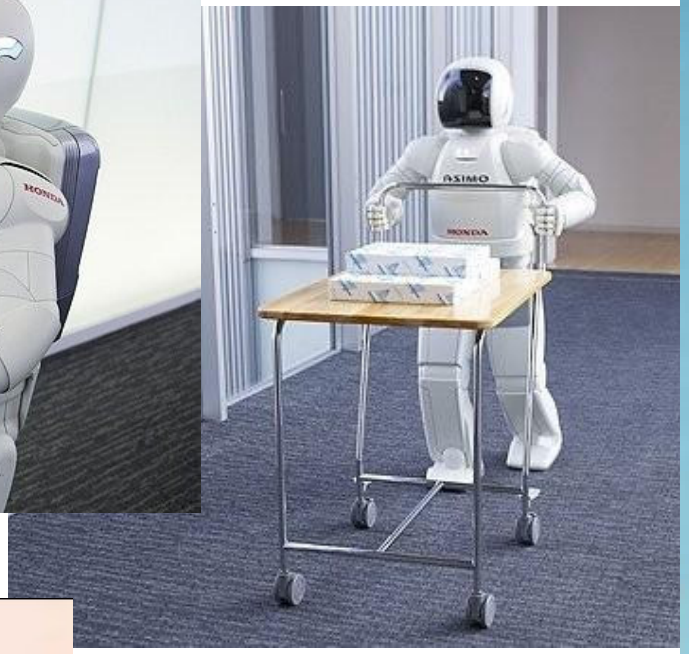




14.1.3 Robotic Examples 3



14.1.4 Robotic Examples 4



14.1.5 Robotic Examples 5

14.1.6 Robotic Examples 6

14.2 Distributed Programming Example

14.2.1 The Task

Go to physical locations of the earthquake zone with given coordinates (latitude and longitude, in decimal degrees) :

50.433, 30.633

50.417, 30.633

50.467, 30.517

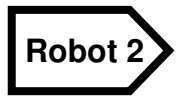
Evaluate damages in each location and transmit the maximum destruction value, together with exact coordinates of the corresponding location, to a management center.

14.2.2 The WPL Code

```
transmit (
  maximum (
     hop (
          ( 50.433, 30.633 ),
          ( 50.417, 30.633 ),
          ( 50.467, 30.517 )
     ).
     evaluate ( destruction ) & WHERE
  )
)
```

14.2.3 Task Injection into Any Robot

```
transmit(maximum(
 hop((50.433, 30.633),
        (50.417, 30.633),
        (50.467, 30.517)).
 evaluate(destruction) & WHERE))
```

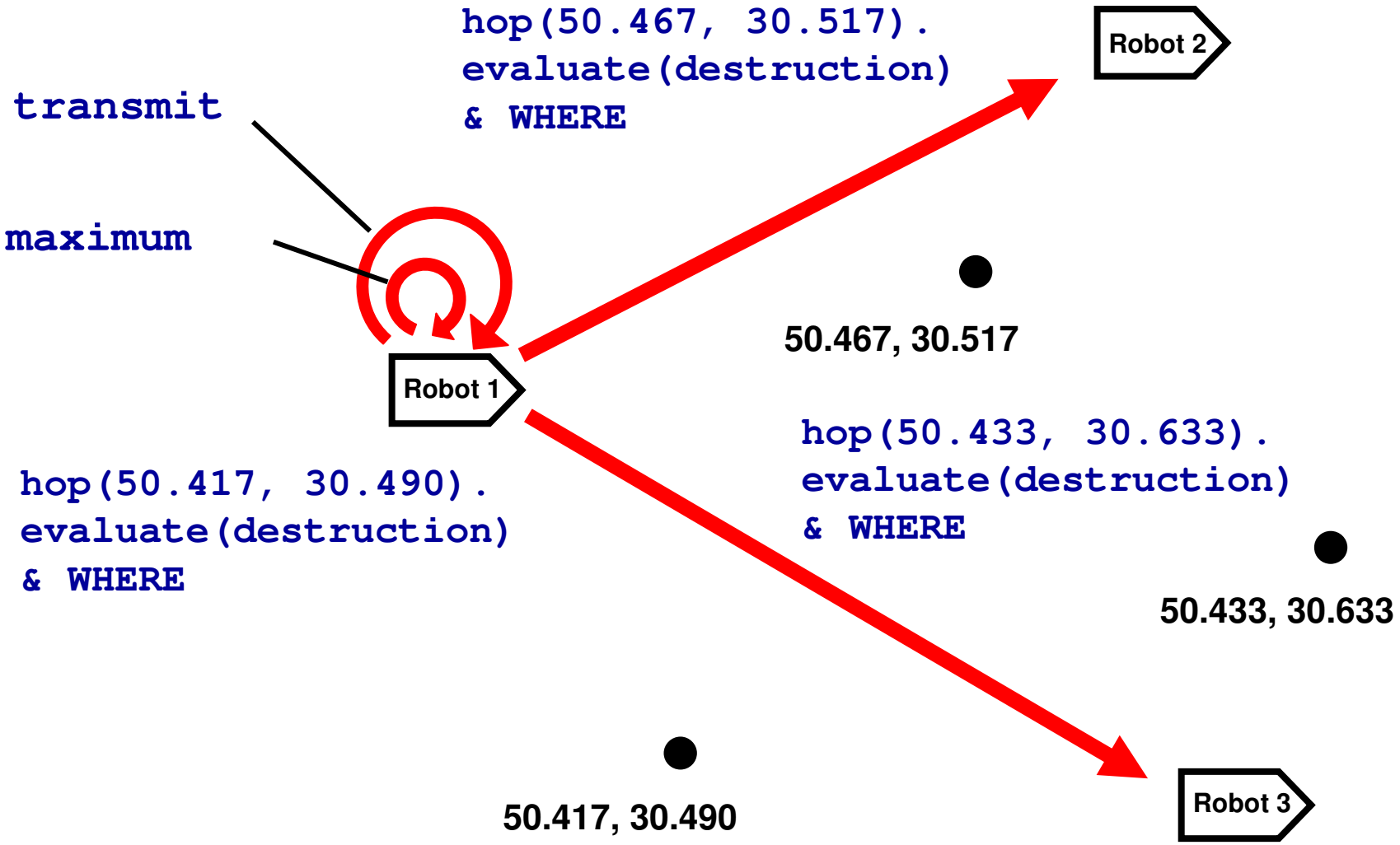


50.467, 30.517

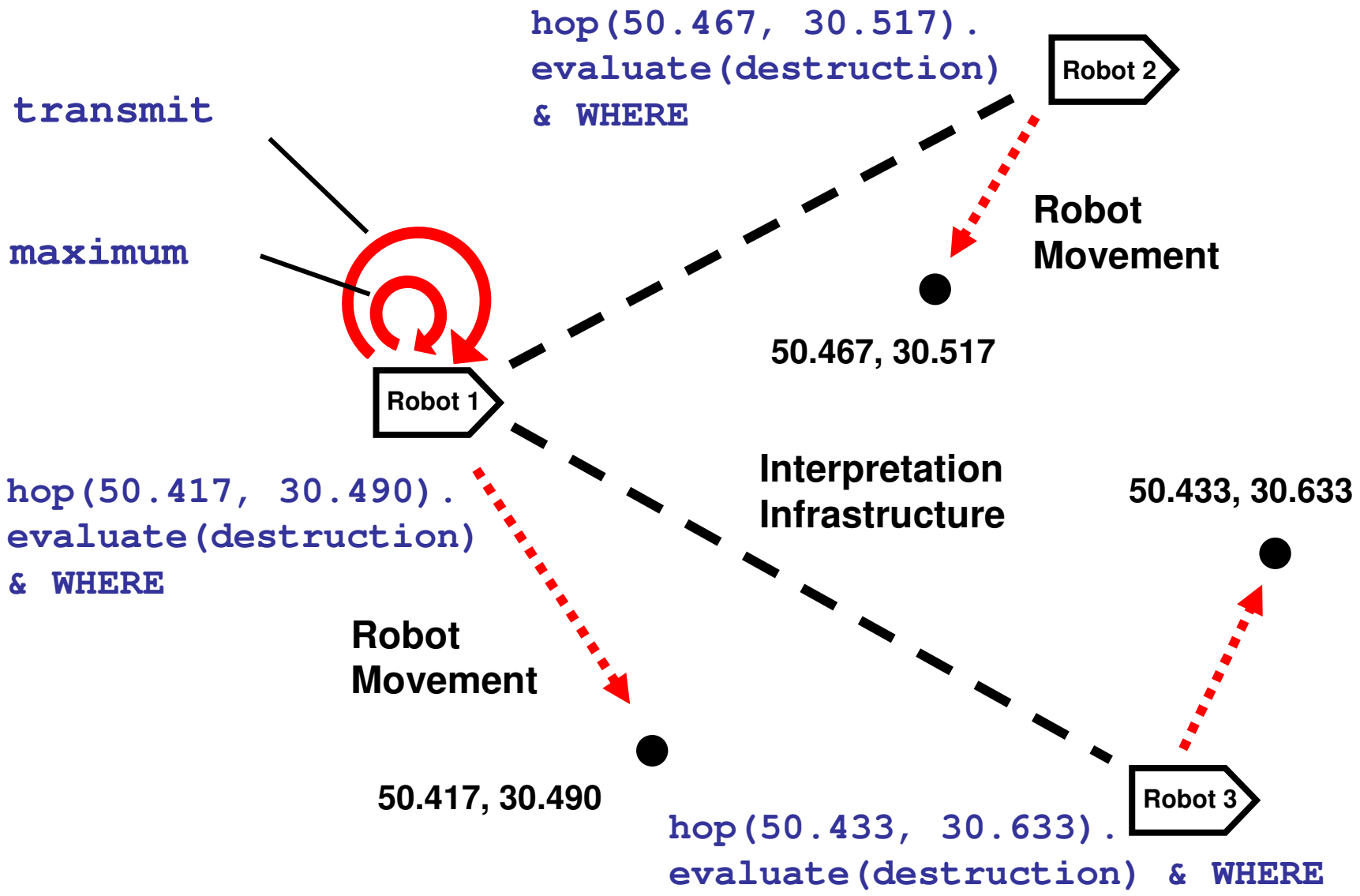
50.433, 30.633

50.417, 30.490

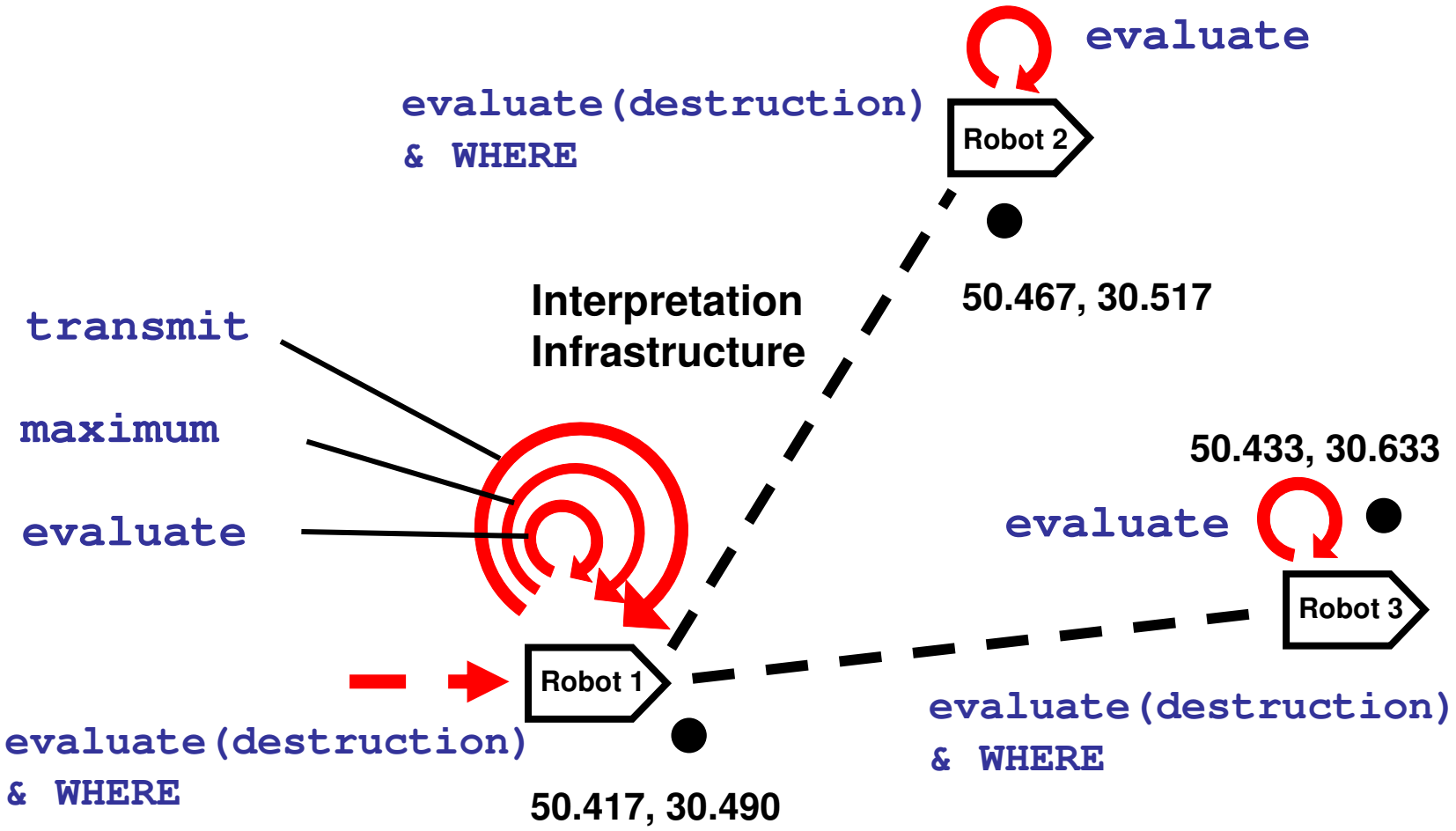
14.2.4 Partitioning & Distributing between Robots



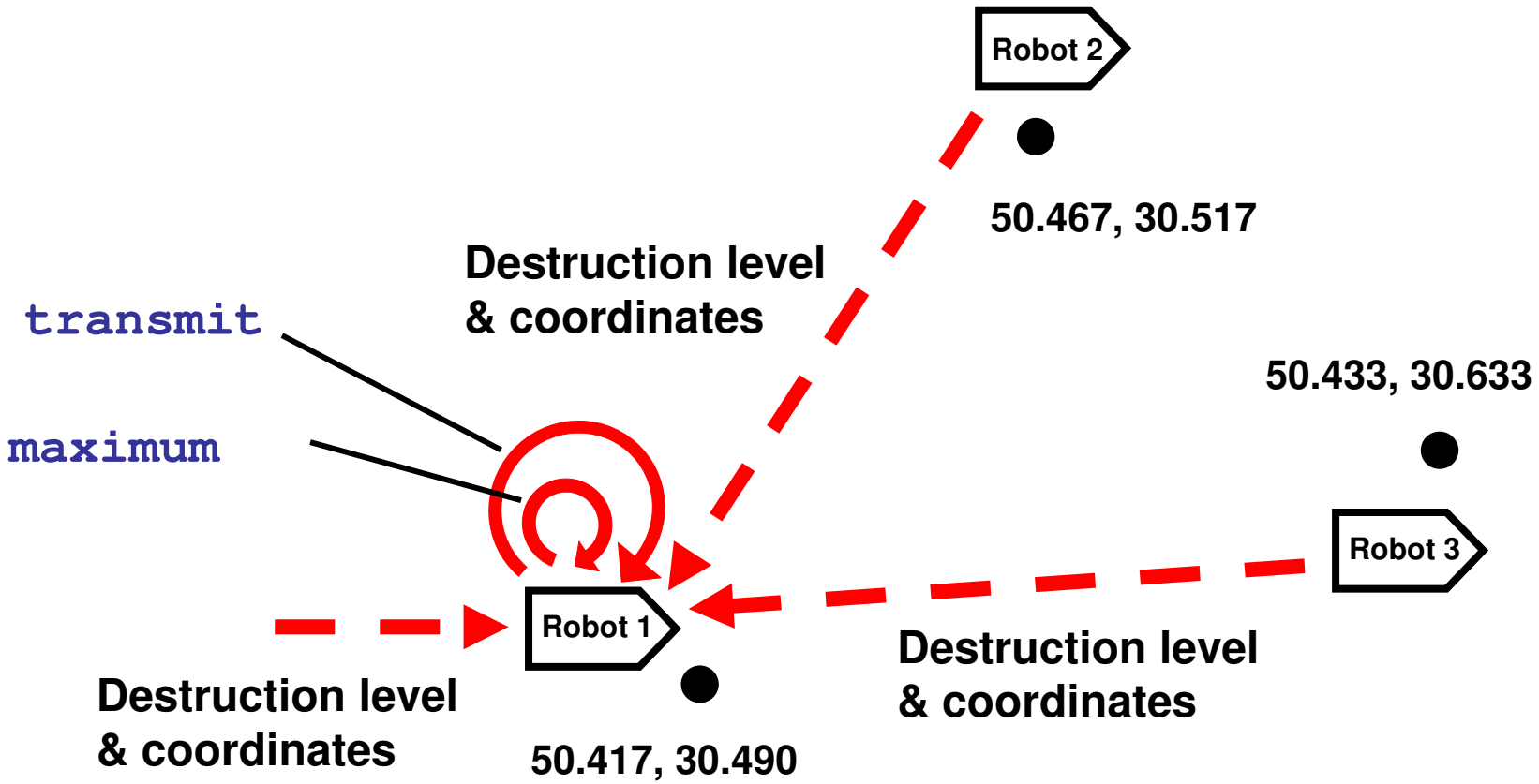
14.2.5 Infrastructure Formation & Robot Movement



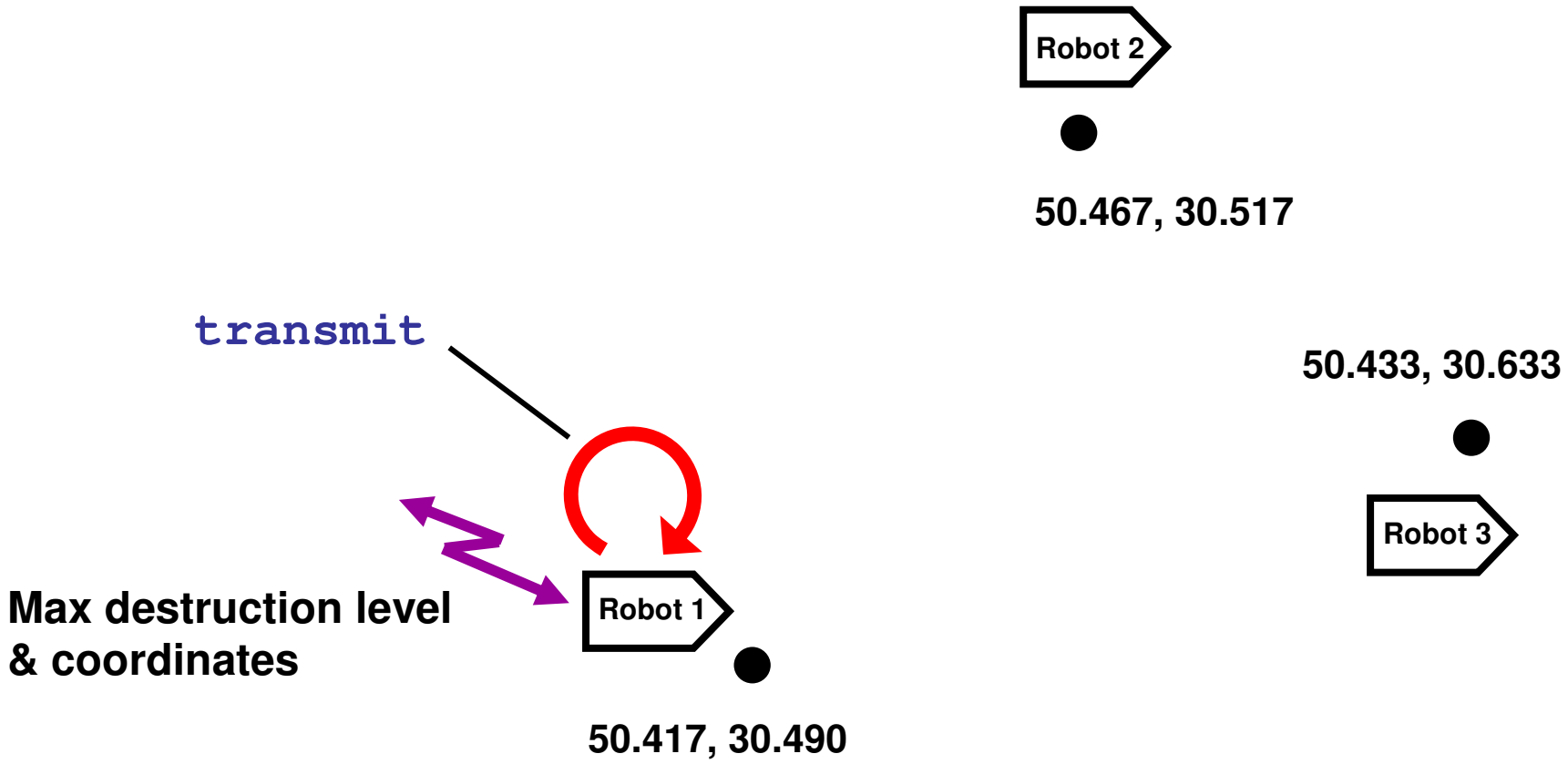
14.2.6 Merging Data & Finding Maximum



14.2.7 Merging Data & Finding Maximum

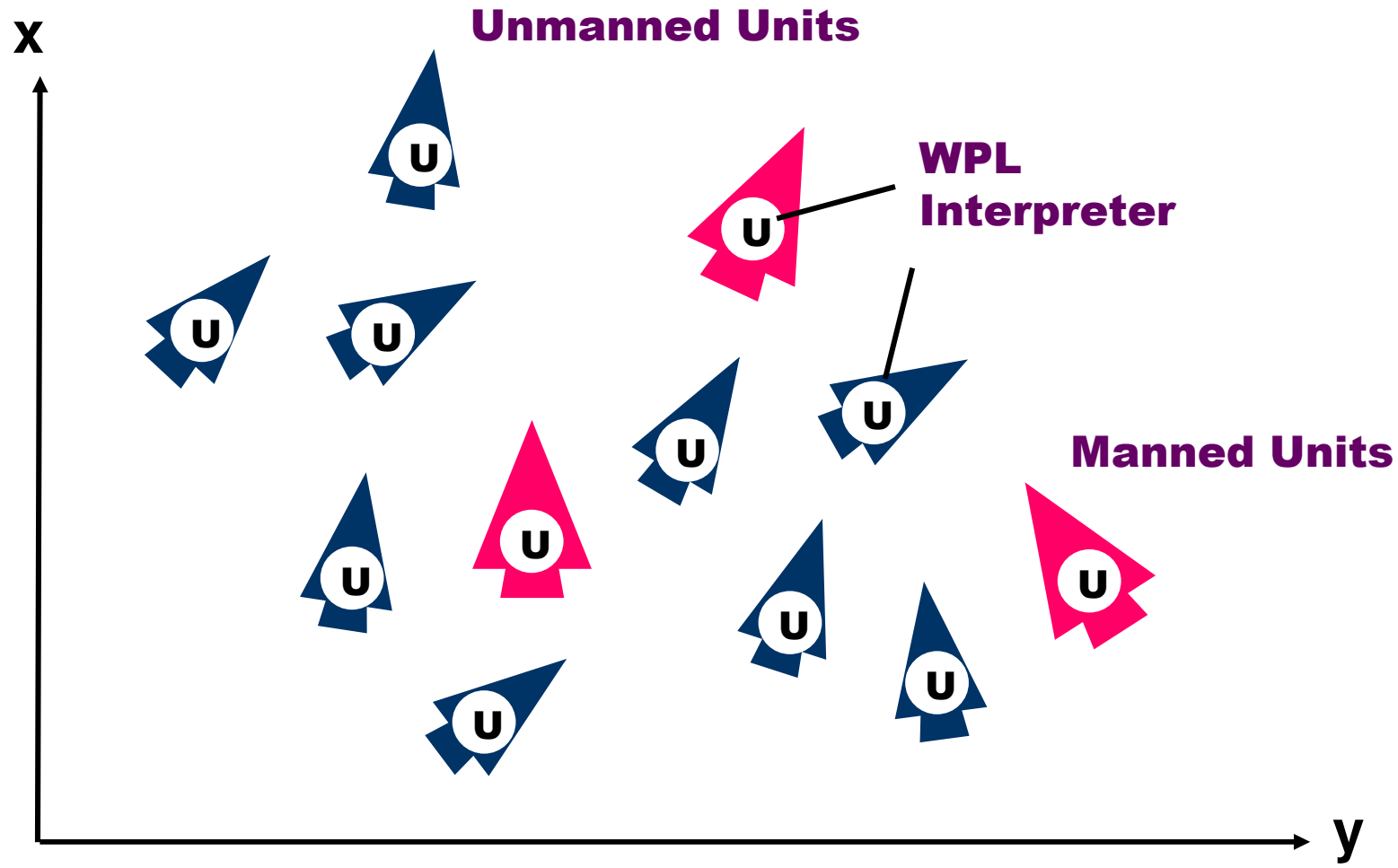


14.2.8 Transmitting Final Result



15 Collective Behavior

15.1 Initial Distribution of Units



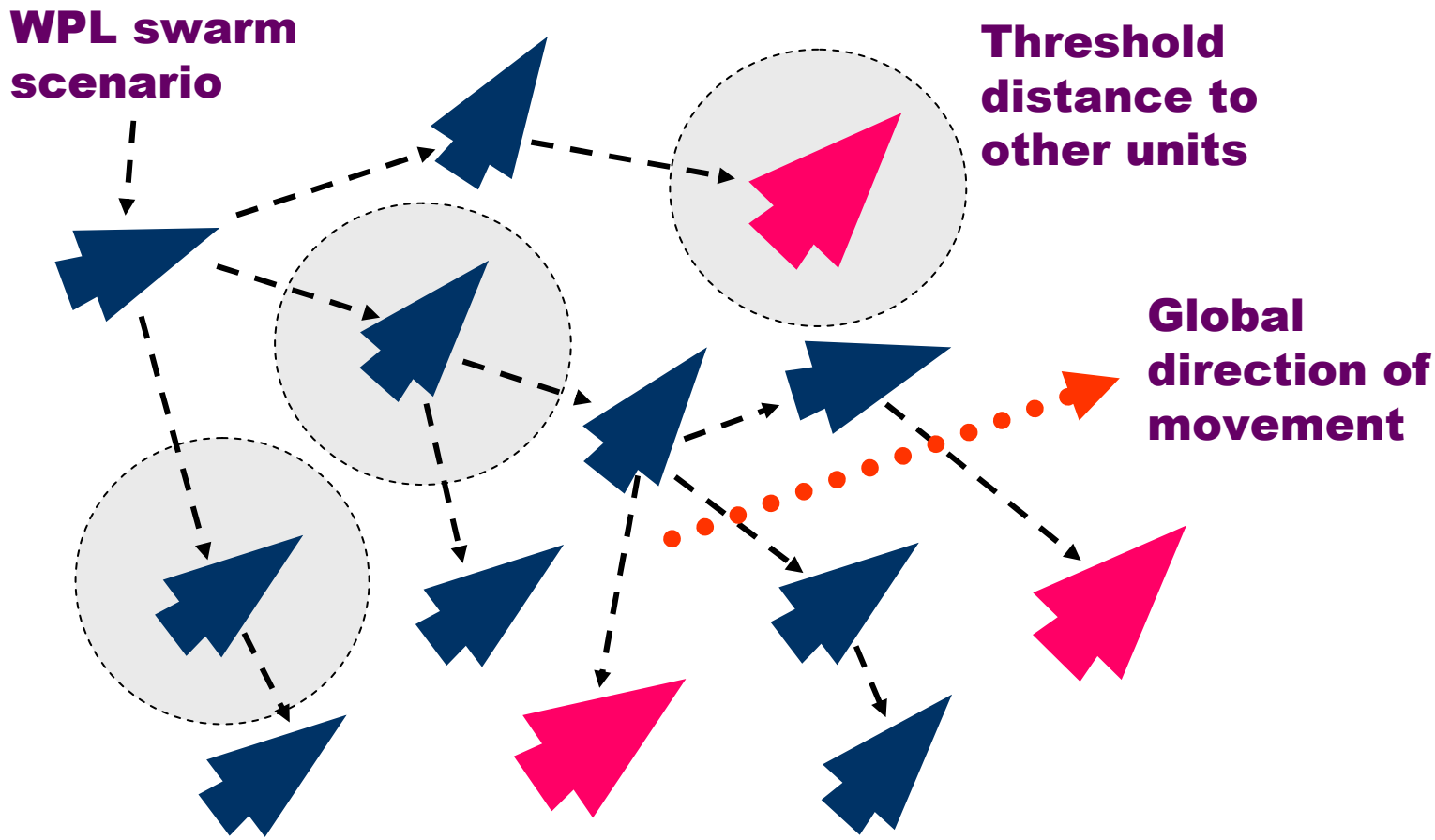
15.2 Swarm Movement Scenario

swarm_move

(Starting from any unit)

```
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 )
```

15.3 Swarm Movement Snapshot



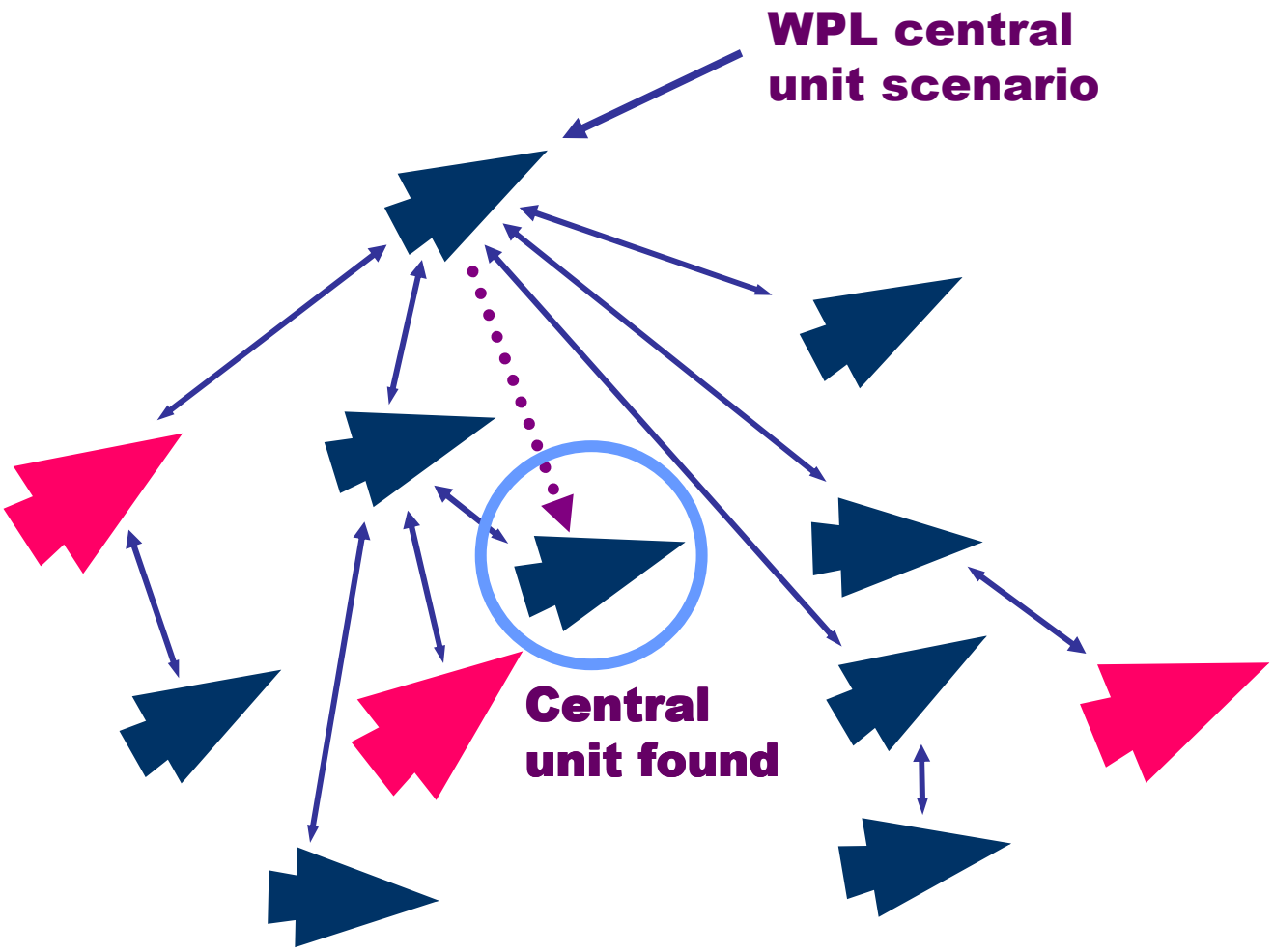
15.4 Finding Topologically Central Unit

find_center

(Starting from any unit)

```
Frontal Aver =
  Average ( hop all_nodes;  WHERE );
Nodal Center =
  Element (
    Min (
      Hop all_nodes;
      Distance ( Aver, WHERE ) & ADDRESS ), 2 )
```

15.5 Central Unit Found



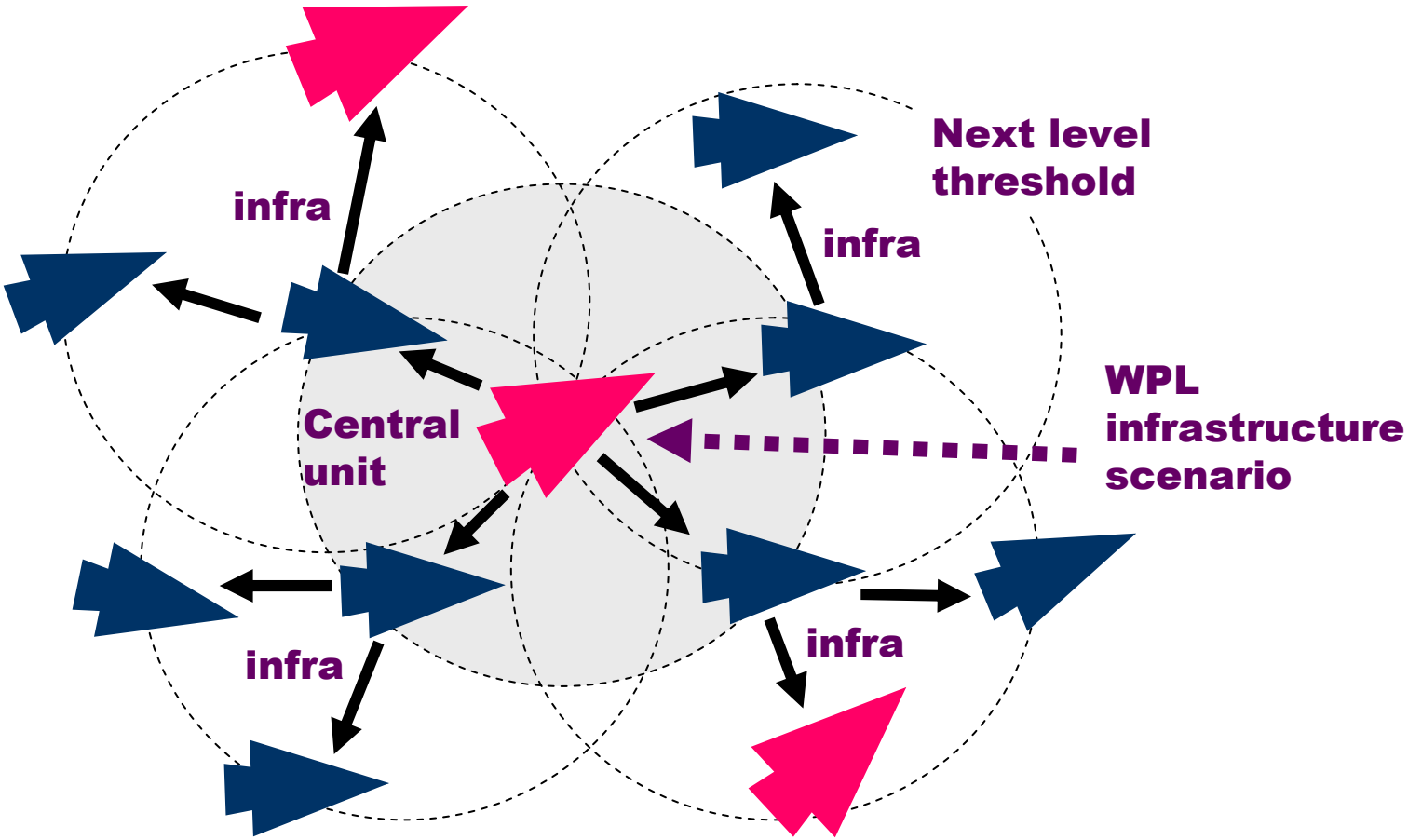
15.6 Creating Runtime Infrastructure

infra_build

(Starting from the central unit)

```
Frontal Range = 20.
   Repeat (
      Create_link (
         + infra, first_come, nodes Range ) )
```

15.7 Hierarchical Infrastructure Built



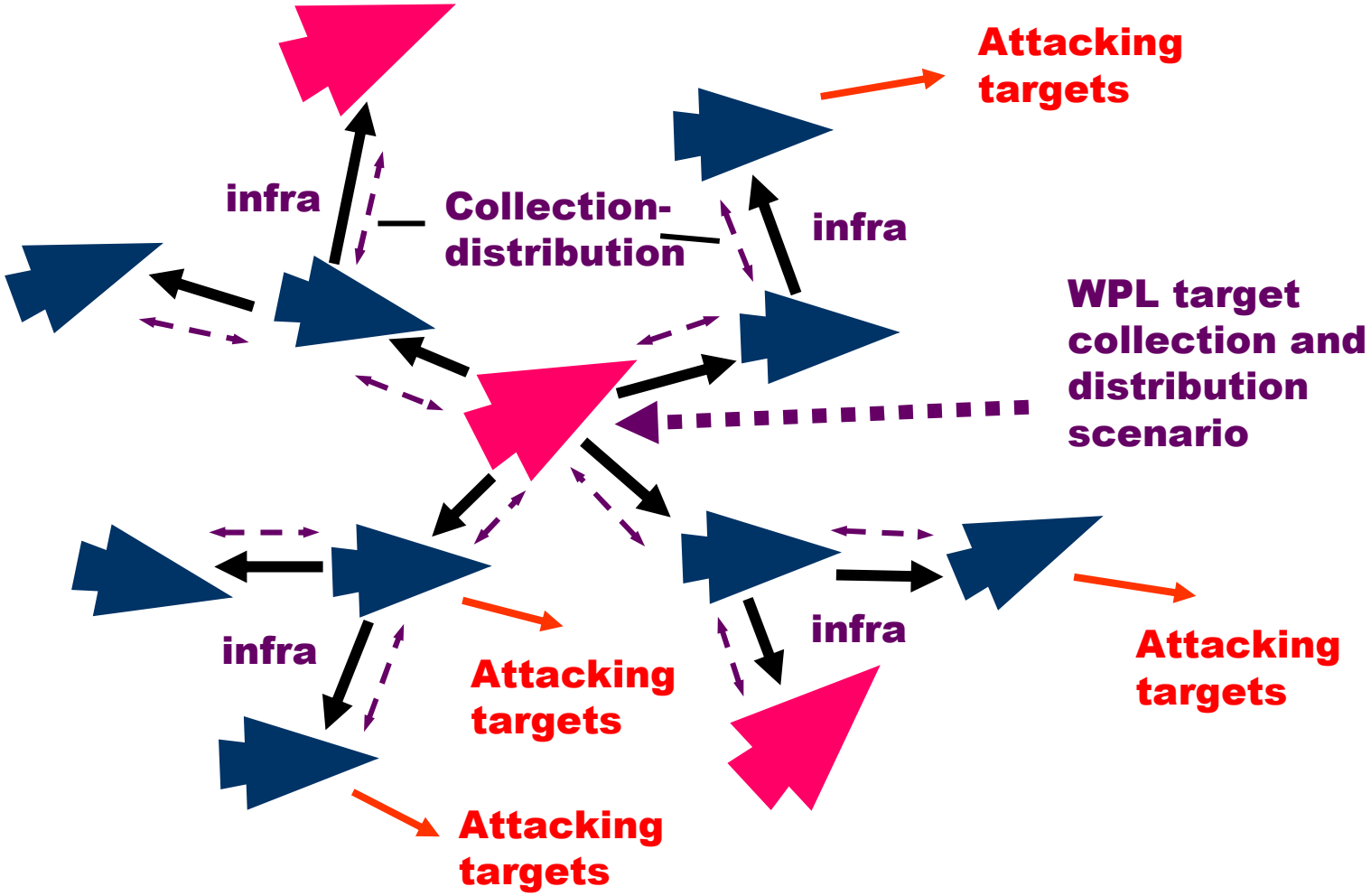
15.8 Fusion & Distribution of Targets

collect_distribute

(Starting from the central unit)

```
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 ) )
```

15.9 Hierarchical Targets Collection and Distribution



15.10 Removing Any Infrastructure

infra_remove

(Starting from any unit)

Hop all_nodes. Remove all_links

15.11 Resultant Combined Solution

(Starting from any unit)

```
Parallel (
    swarm_move,
    Repeat (
        Hop ( find_center );
        infra_remove; infra_build;
        Orparallel (
            collect_distribute,
            Sleep 360 ) ) )
```

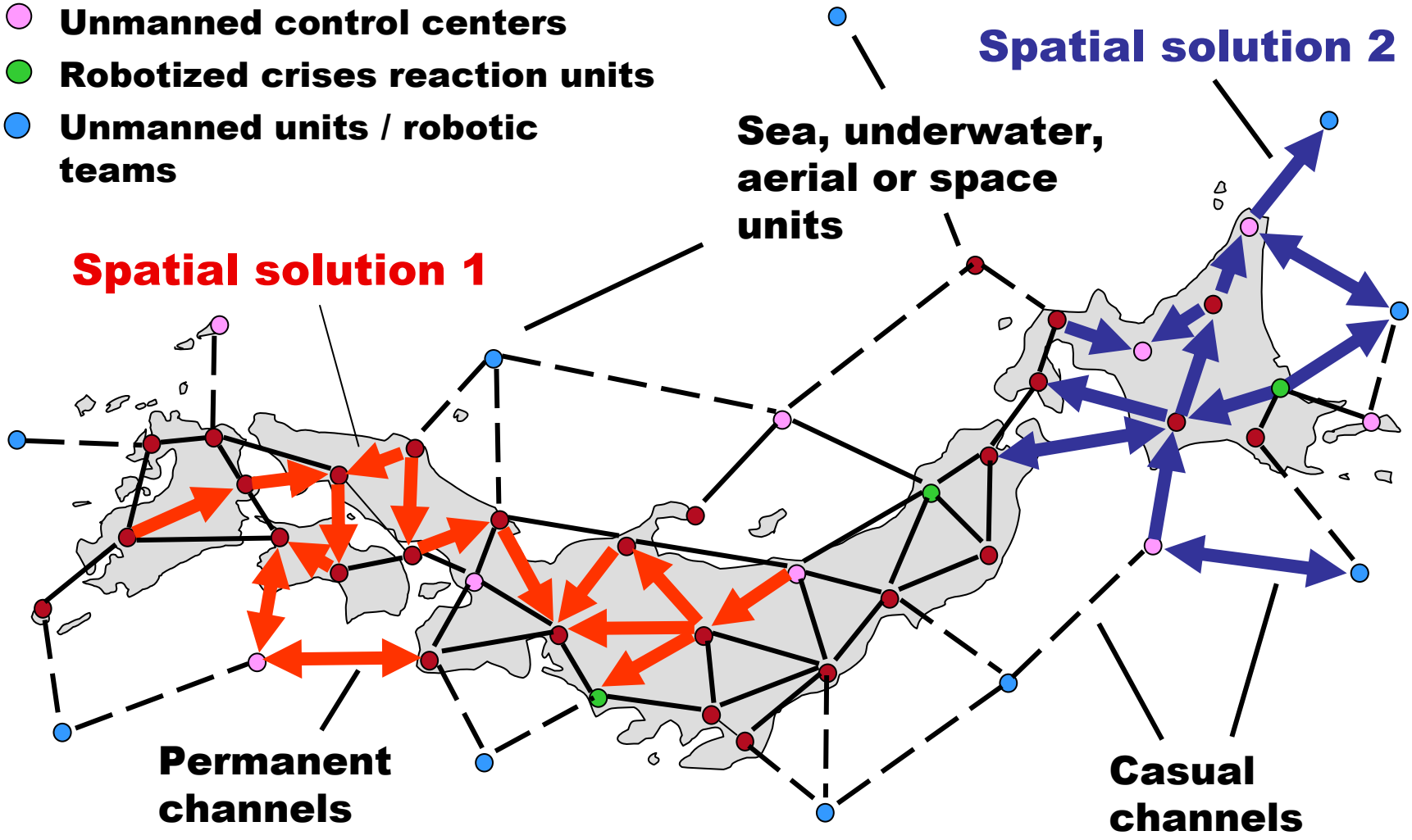
16 Global Infrastructures

16.1 Multiple Infrastructures

The technology can effectively support vital infrastructures of national and international scale, solving complex problems in them in parallel. These may relate to politics, economics, demographics, weather prediction, environmental pollution, postal service, transport, flow of industrial goods, tracing international criminals, air and space defense, etc.

16.2 Spatial Solutions

- Manned control centers
- Unmanned control centers
- Robotized crises reaction units
- Unmanned units / robotic teams



17 Conclusions

- **The proposed technology can convert any distributed system into a universal spatial machine capable of solving problems on itself and on the surrounding environment.**
- **The whole system is driven by integral, monolithic, high-level scenarios setting how to behave as a whole, with partitioning and implementation details delegated to distributed intelligent interpretation system.**
- **The system scenarios in World Processing Language are very compact and can be created on the fly, timely reacting on changing environment and mission goals.**
- **Any scenario can start from any available component and cover the system at runtime, during its evolution.**
- **The approach offers realistic possibilities for runtime recovery after indiscriminate damages, including reassembling of the whole system from any point.**
- **The technology can help dominate over other system organizations, especially based on communicating agents.**