



# eHermes: Dynamic MAS Generator



Glenn Jayaputera  
CSSE, Monash University  
[gtj@csse.monash.edu.au](mailto:gtj@csse.monash.edu.au)

•This Presentation is Based on *AAMAS-05* and *AOSE-05* Papers

# [ Outlines ]

---

- Background
- Motivations
- Objectives
- Theoretical Works
- Implementation
- Experimental Results
- Conclusions and Future Work

# Background

- There are numerous MAS in existence
  - AuctionBot (online auction and bidding)
  - MAgNET (B2B)
  - Chayani, IntelliShopper (shopping Asst)
  - InfoSleuth, BIG
- In general MAS are developed using agent development toolkit
  - Zeus, Aglet, JADE, Retsina, DECAF, etc

## Background – cont'd

- In general most of the existing MAS are built to solve specific problems
- Agents are highly coupled with the coupled with the problem domains
  - Fixed in functionalities
  - Fixed in numbers
  - In general, agents get instantiated when the system itself is fired up.

# Motivations

- Failures occur due to the change in environment or a shift in the problem domain.
- When failures occur:
  - The whole monolithic system must be stopped
  - Agents must be modified/tested/deployed
- Unfriendly to system resources
- Current agent development methodology is quite orthodox:
  - Agents developed are total assets
  - Enhancing agents: social, intelligent, communication, etc

# Objectives

- Developed a new approach, where:
  - MAS are dynamically constructed
  - Agents are generated on demand and just-in-time based on current plan
  - Agents are specialized at run-time, hence different missions will have different sets of agents.
- Agents are now become:
  - Tools to achieve the goal
  - Disposable as opposed to total assets.
  - Can be replaced at run-time

## Objectives – cont'd

- Promote the success rate in achieving the goal by:
  - Allowing plan to be modified at run-time
  - Providing a mechanism where the system can be stopped, started, suspended and resumed at later stage and at the same/different location

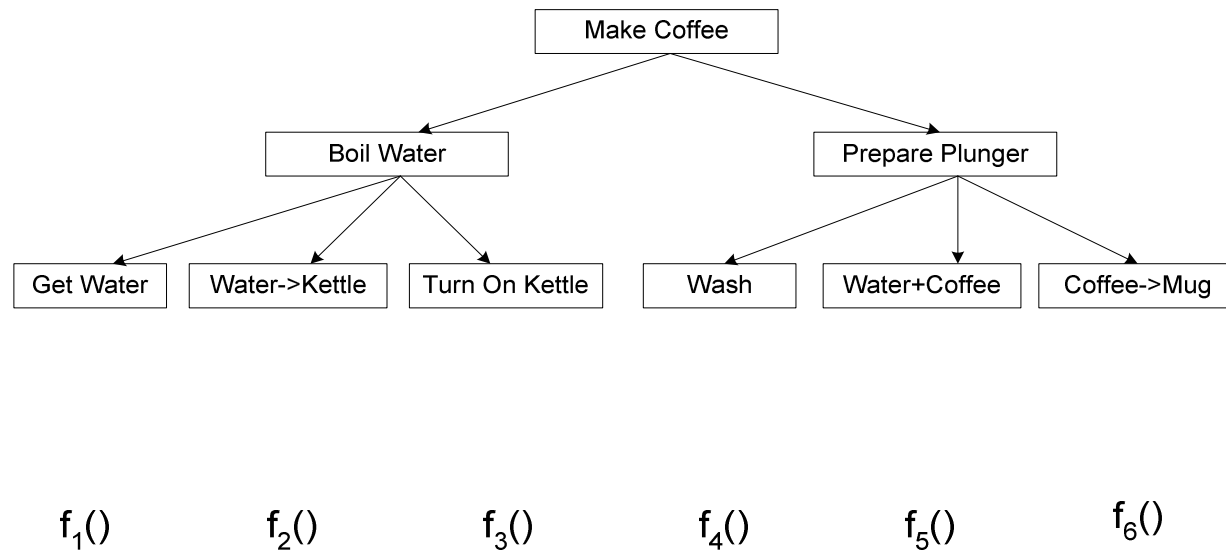
# Mission

- Mission is the cornerstone
- Agents exist because there is a mission to perform.
- Indirectly, agents are created because there is a mission
- A mission must have a goal and a plan (complete or partial) on how to achieve the goal
- Formally defined as  $M=(o,P,A,Z)$ 
  - $o$  is the goal as string
  - $P$  is a set of plans
  - $A$  is a set of agents (mobile or stationary) working in the mission
  - $Z$  is a set of mission states

# Plan

- A plan must be
  - represented in a non-ambiguous and concise form
  - Simple and light-weight so it can be easily modified at run-time
- TDG (Task Decomposition Graph) is a DAG-based plan structure
  - Graph and HTN based
  - Retain decomposition deliberation

# Plan – cont'd



# Task

$(u, n, y, s, o)$  where  $u \in U, n \in N, y \in Y, s \in S, o \in O$

$U$  is a set of unique IDs as string,

$N$  is a set of locations at which the tasks must be carried out,

$Y$  is a set of task types, i.e.  $Y = \{Primitive, Compound\}$ ,

$S$  is a set of task statuses, i.e.  $S = \{Completed, Pending, InProgress, Failed, Aborted, Assigned\}$ , and

$O$  is a set of functions that a task will execute representing application logic.

# Task – cont'd

***Completed***: this status is used to indicate that the task has been completed successfully.

***Pending***: this status is used to indicate that the task is ready to be assigned to an agent for the execution.

***InProgress***: this status is used to indicate that the associated task is currently being executed.

***Failed***: this status is used to indicate that the task has failed when executed.

***Aborted***: this status is used to indicate that the task has been aborted while it was executed.

***Assigned***: this status is used to indicate that the task has been assigned to an agent but has not been started yet.

# Link

$$(t_i, t_j, q) \mid i, j \in \mathbb{Z}^+, q \in Q, i \neq j, t_i, t_j \in T$$

$Q$  is the relationship types, which is

$$Q = \{Includes, DependsOn\},$$

$T$  is a set of tasks as defined previously

- ***DependsOn*** link illustrates the dependency between two tasks.
- ***Includes*** link illustrates the inclusiveness of a parent task over its direct sub-tasks

# Task Decomposition Graph

$TDG = (T, L)$  where

$T$  is a set of tasks

$L$  is a set of links

Given  $f_{type} : T \rightarrow Y$ , a function that return the type of a given task then

$\forall (t_i, t_j, q) \in L \mid f_{type}(t_i) \neq \text{Primitive}$  where  $i, j \in \mathbb{Z}^+, q \in Q$

$\forall (t_i, t_j, q) \in L$ , if  $q = \text{Includes}$  then  $f_{type}(t_i) = \text{Compound}$  where  
 $i, j \in \mathbb{Z}^+$  and  $q \in Q$

- *Rule 1* states that the source of a link cannot be a primitive task,
- *Rule 2* dictates that when the link type is includes then the source must be a compound task and the destination can be a compound and/or primitive tasks
- The two rules above guarantee that the primitive tasks are always the terminal nodes in the graph

# Mission Data Space (MDS)

- Medium to tasks to exchange data
- Modelled as Java Space but only local to a mission
  - Easier to manage
  - Security
- Defined as a set of tuples of the form  $(t, r, ts)$ , where
  - $t \in T$ , and  $T$  is a set of tasks
  - $r \in R$ , and  $R$  is a set of results
  - $ts \in TS$ , and  $TS$  is a set of timestamps

# Mission State

Defined as a tuple of the form  $(p, mds, e, ts)$  where

$p \in P$  is the current plan

$mds \in MDS$  is a snapshot of MDS

$e \in E$ ,  $e$  is the event that caused the transition

$$E = \{Suspend, Stop, Resume, Start, ChangePlan\}$$

$ts \in TS$  is a timestamp

Given a  $f_{status} : T \rightarrow S$ , a function that return the status of a given task

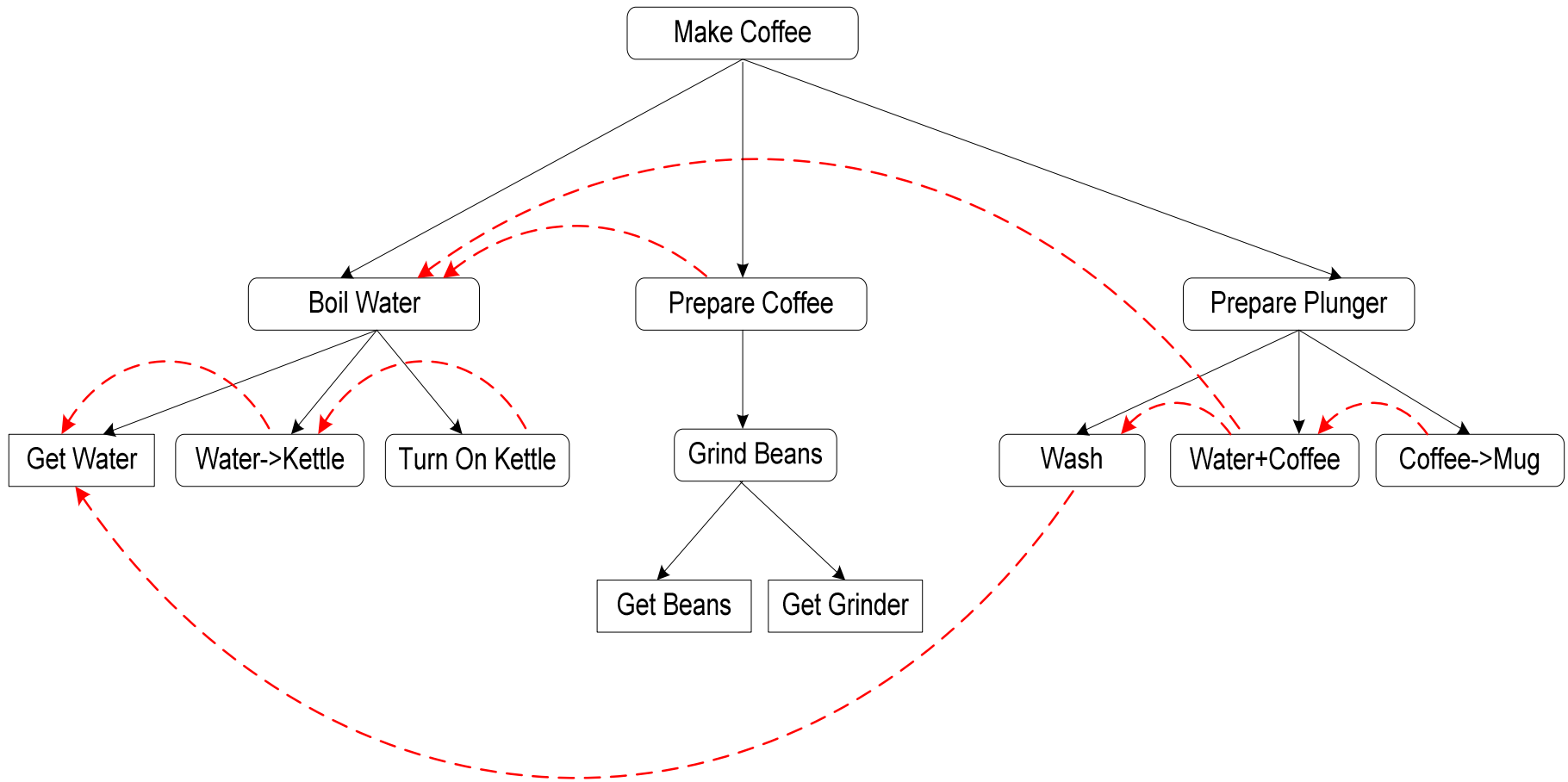
A mission is said to be completed when the following holds

$$\forall t \in T \mid f_{status}(t) = Completed$$

# Mission Execution History

- Maintain the history of a mission execution is supported by MDS and Mission States
- Allows a mission to be stopped and resumed at later stage at the same or different location
- Provides valuable knowledge for the planner in future planning and execution

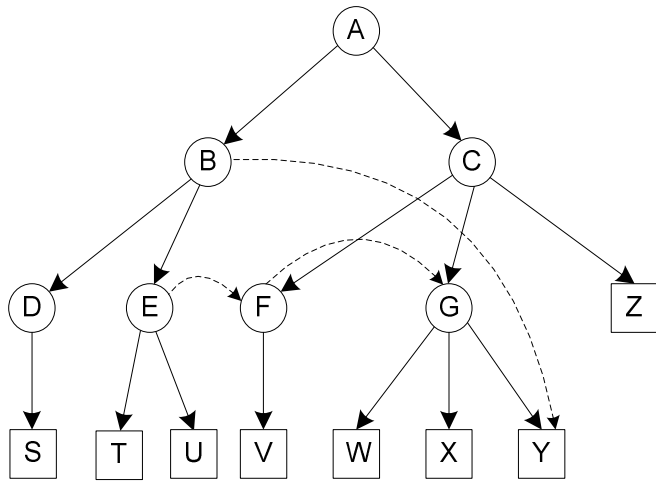
# Sample Scenario



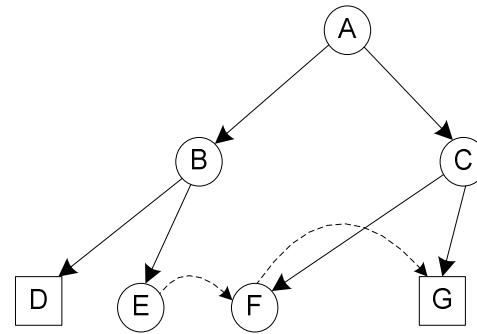
# Plan Execution

- Only primitive tasks are executed
- Execution are in parallel fashion.
- Tasks that have the same locality are group together.
- Agents are created and assigned task(s) when there are tasks to be performed.
- Agents get destroyed once they have completed their assignments
- Mobile and Stationary agents are used.
- Compound tasks are turned into primitive tasks iff all their subtasks are completed

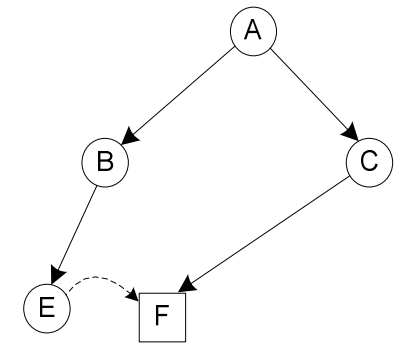
# Plan Execution – cont'd



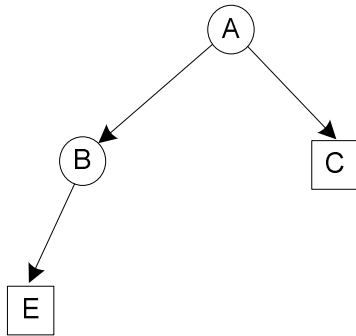
(a)



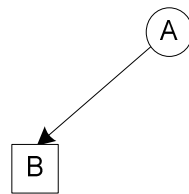
(b)



(c)



(d)



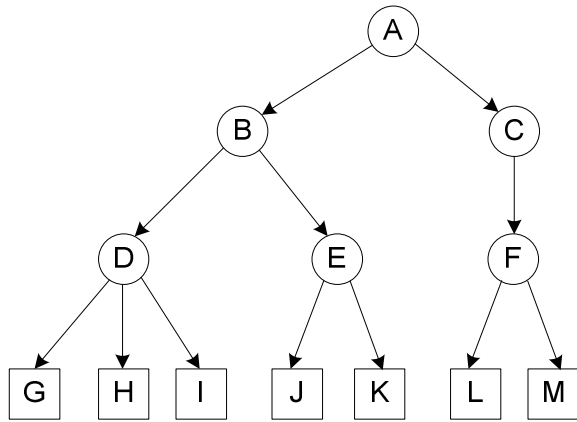
(e)



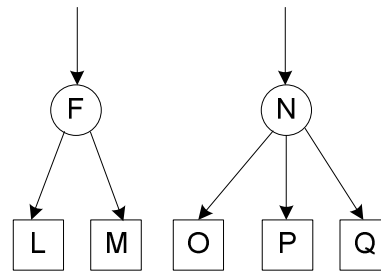
(f)

# Plan Modification

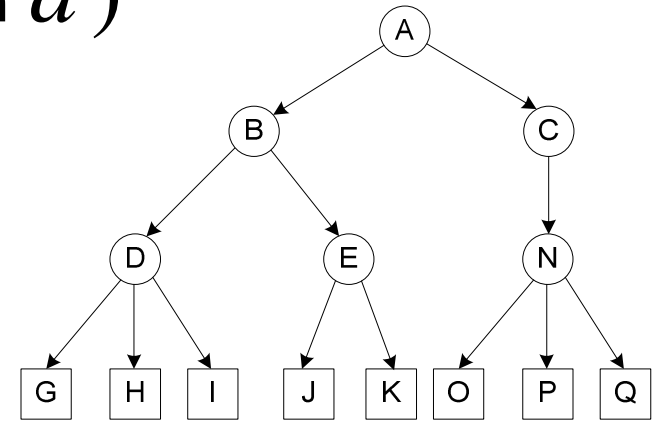
$$(p \cup d) - (p \cap d)$$



(a) Original Plan ( $p$ )

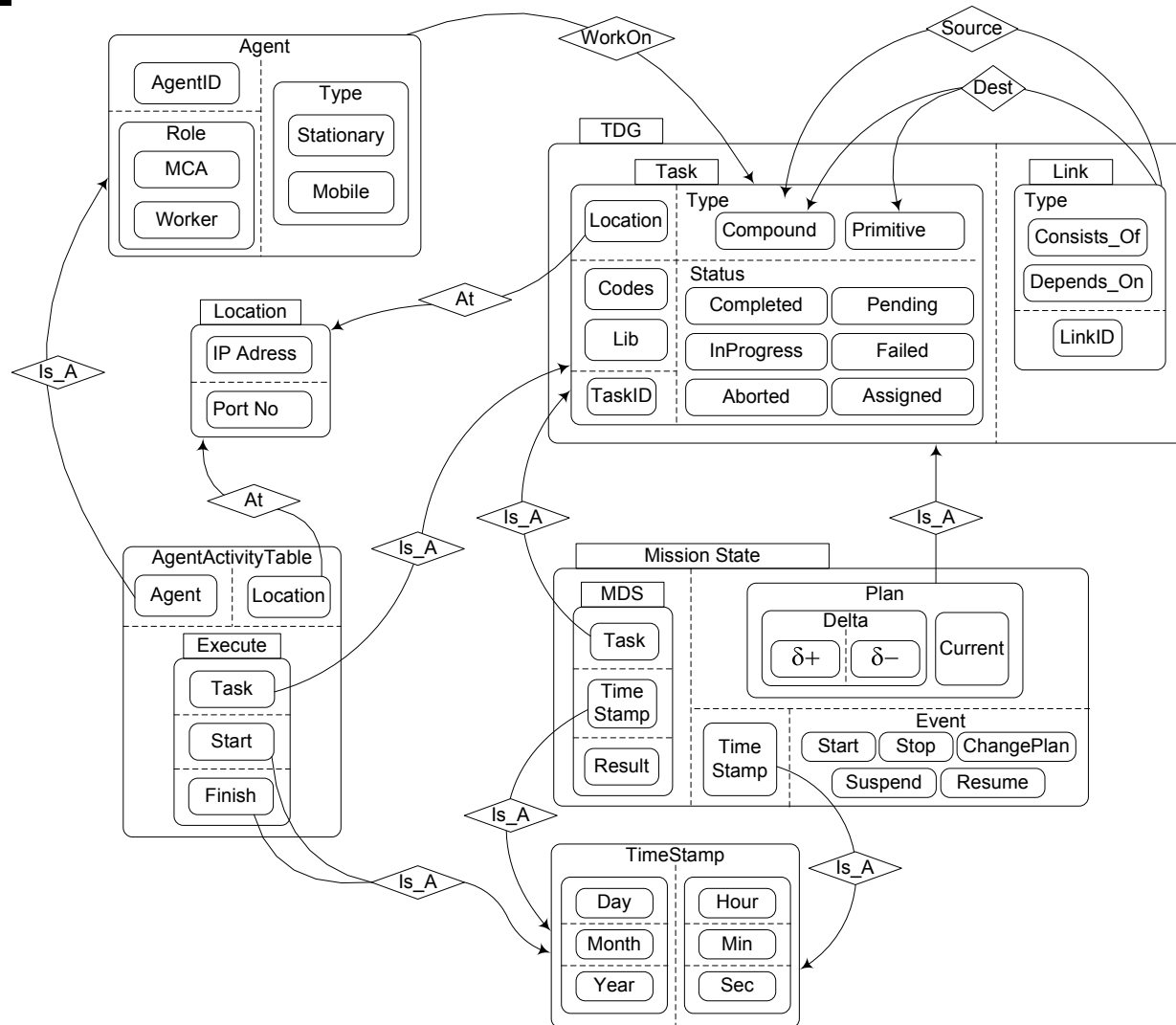


(b) Delta ( $d$ )



(c) New Plan  $((p \cup d) - (p \cap d))$

# Mission Object Model



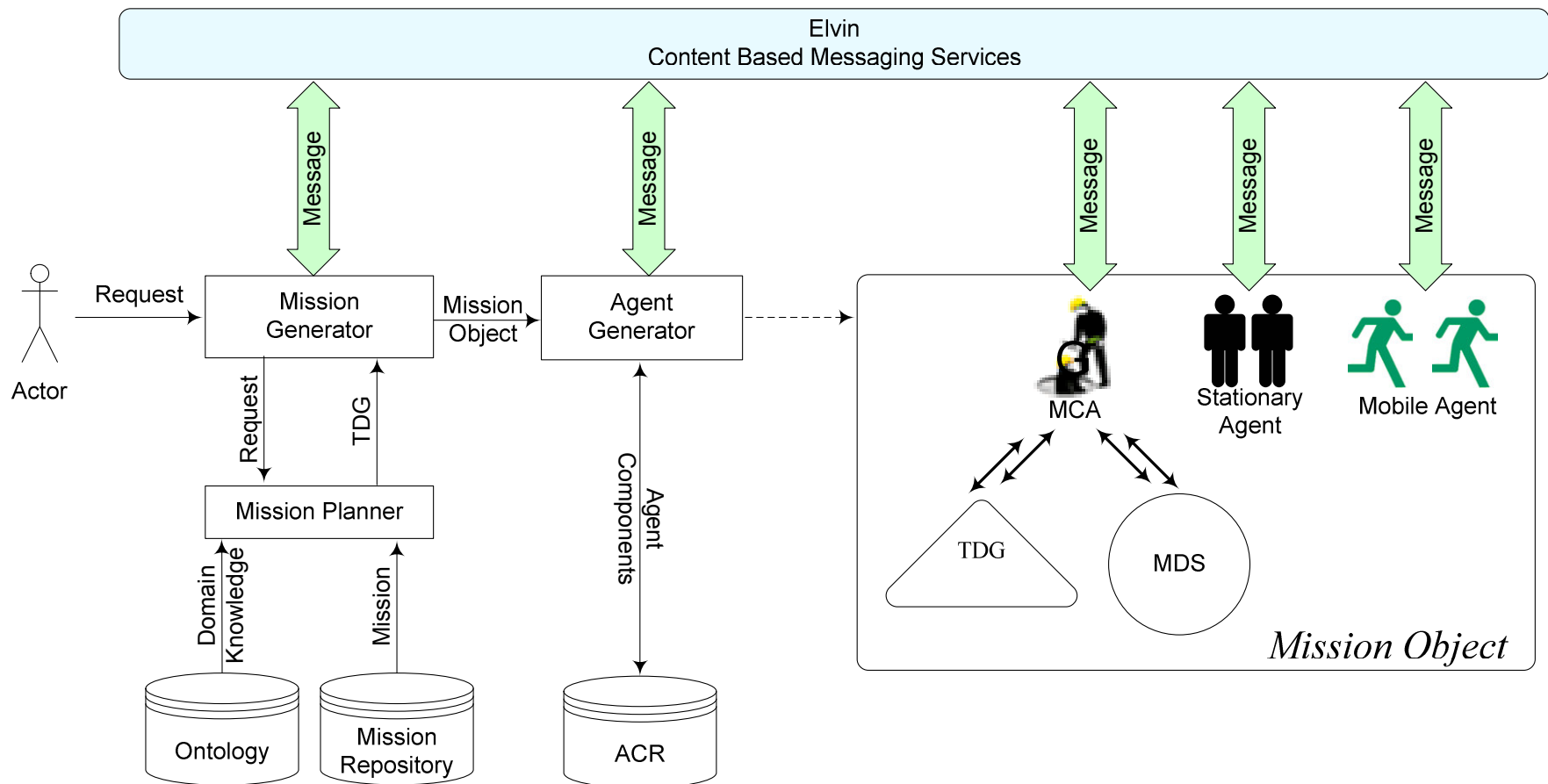
# Agent Activity Table

- Holds important information for agent coordination
- Which agents are working of the current mission?
- What tasks being performed?
- How long it has been since the task is started?
- Where is their location?

## Agent Activity Table – cont'd

- Agent Activity Table is updated every time the following events occurred:
  - Created
  - DepartTo
  - ArriveAt
  - StartTask
  - FinishTask
  - ErrorTask
  - SelfTerminate
  - Abort
  - AtHome

# eHermes



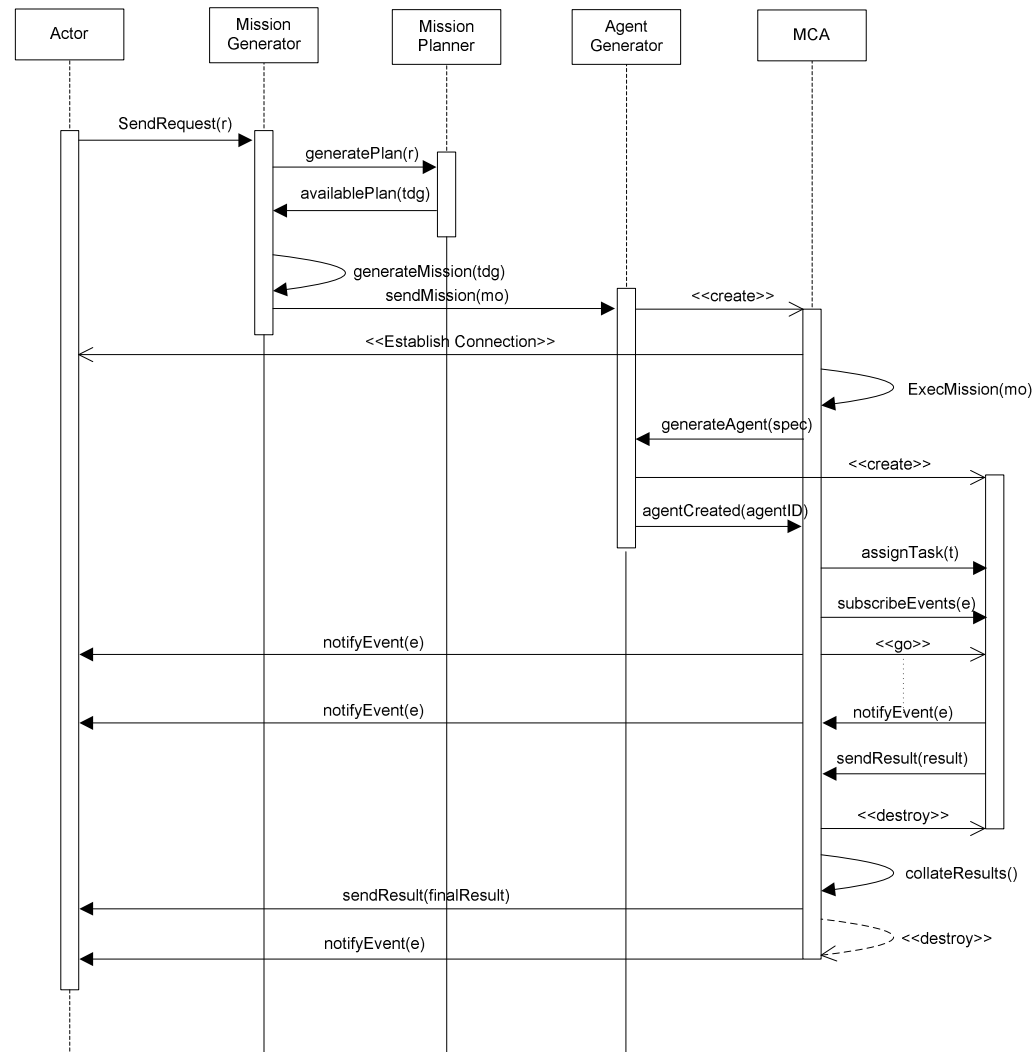
## eHermes – cont'd

- *Mission Planner*. The module that does the planning.
- *Mission Generator*. The module that is in charge in generating mission objects.
- *Agent Generator*. The module that generates agents on request.
- *Elvin*. The backbone of agent communication system.

# Mission Control Agent (MCA)

- Each mission is executed under supervision from a special agent called Mission Control Agent (MCA)
- MCA main tasks are:
  - Ensuring smooth execution as much as possible
  - In charge of the timing to generate agents
  - Coordinates and orchestrates the working agents
  - Control the data flow on the MDS
  - Client liaison (humans or other agents)

# Sequence Diagram



# Agent Communication

- Agents communicate via short and content-based messages as opposed to speech-act dialogues.
- Short and reliable messages are needed because:
  - Reach ability issue due to mobile agents are used.
  - Agents mobility across fixed and/or wireless networks.

## Agent Communication – cont'd

- Based on those reasons, it is needed a messaging sub-system that is:
  - Fast and reliable to accommodate the network reliability issue
  - Publish/Subscribe to accommodate the agent reachability issue
  - Content based to accommodate the requirement for data exchange

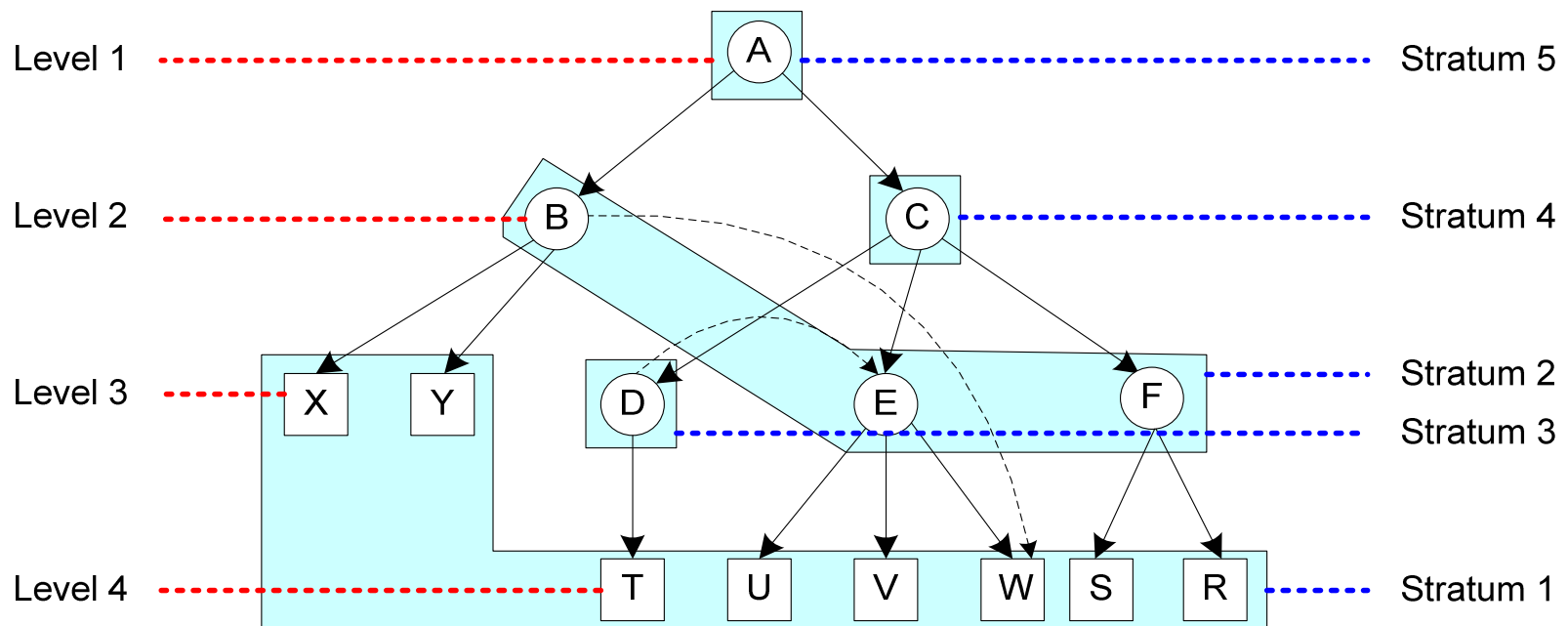
# Optimization

- Creating an agent for each task is not economical
- Indirectly cause extra load to the MCA
- Optimization is required to minimize the number of agents generated
- Trimming the number of agents should not cause any significant delay to the mission completion time.
- Cost and benefit analysis is used to achieve the goal of this optimization process.

# Optimization – cont'd

- Stratum is defined as a group of primitive tasks that are ready to be executed

$$ST = \{t \mid t \in T, f_{type}(t) = primitive \wedge f_{status}(t) = pending\}$$



# Optimization – cont'd

The cost to execute a task

$$f_{cost}(t) = f_{cost}(createAgent(t)) + f_{cost}(exec(t))$$

The cost to execute a mission then

$$\forall t \in T, \sum_{i=1}^n f_{cost}(t_i) \text{ where } n = |T|$$

## ■ Assumptions

- The cost to execute a task is the same everywhere, ignoring the locality and computing power of the host
  - There is no cost involved in moving an agent from one location to another. Hence there is no “agent transmission cost”
- Based on the assumptions and the cost formula then the total cost can be reduced if the number of agents generated is reduced.

# Optimization – cont'd

- Critical Time ( $CT$ ) of a mission is defined as *the minimum amount of time required to complete the mission*, similarly the  $CT$  of a stratum is the minimum amount of time required to complete all the tasks in that stratum.

Given  $f_{\Theta}(t) = \mathbb{Z}^+, t \in T$ , a function that returns the elapsed time of the execution of task  $t$ , then the  $CT$  of a stratum is defined as:

$$CT_{ST_i} = \max\{f_{\Theta}(t_j) \mid t_j \in ST_i, i \in \mathbb{Z}^+, j \in \mathbb{Z}^+\}$$

the maximum amount of elapsed time of all the tasks in  $ST_i$

- The number of agents can be reduced by allocating as many tasks as possible to each agents providing their elapsed time does not exceed the stratum's  $CT$ .

$$\sum_{j=1}^n f_{\Theta}(t_j) \leq CT_{ST_i} \text{ where } n = |ST_i|, t_j \in ST_i, i \in \mathbb{Z}^+, j \in \mathbb{Z}^+$$

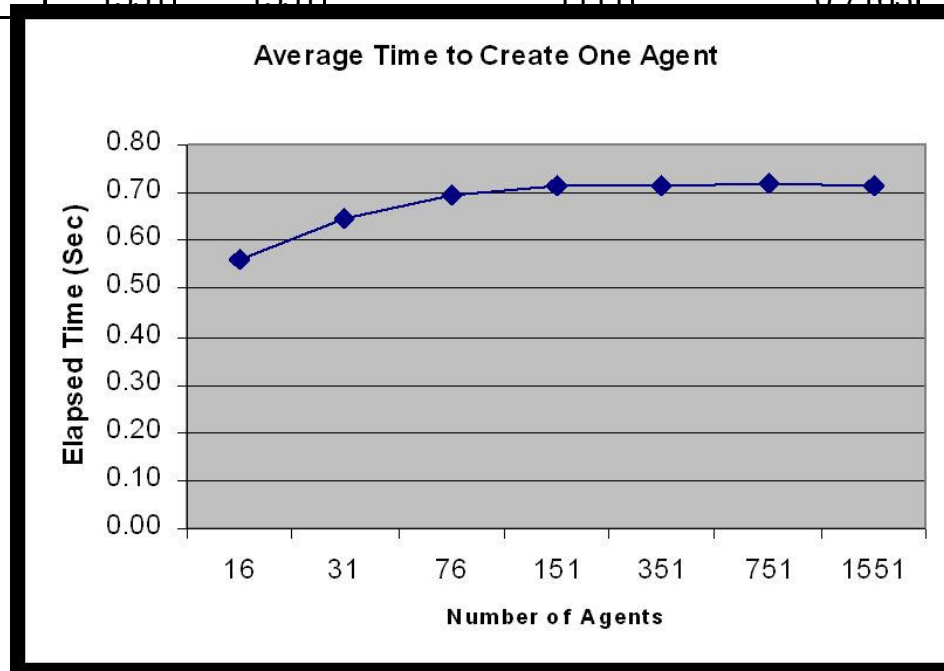
- Reducing the number of agent this way is a bin-packing problem
- FFD (First-Fit Decreasing) algorithm is the most promising near-optimal solution

## Optimization – cont'd

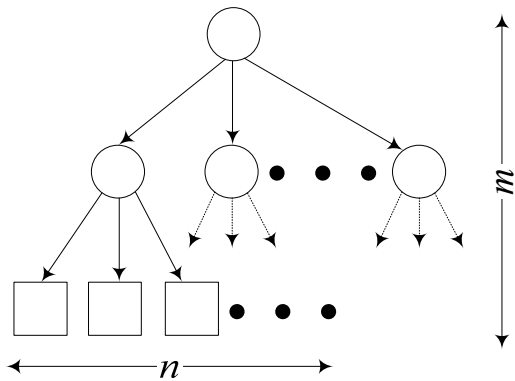
- Tasks in a stratum is sorted in decreasing order according to their elapsed time
- The elapsed time of the 1<sup>st</sup> task is the critical time of the stratum
- From the 2<sup>nd</sup> to the last task, all the tasks are grouped into task clusters where the total elapsed time of the clusters is smaller or equal to the critical time of the stratum
- Each of the task clusters is assigned to a single agent to be executed.

# Agent Generation Test

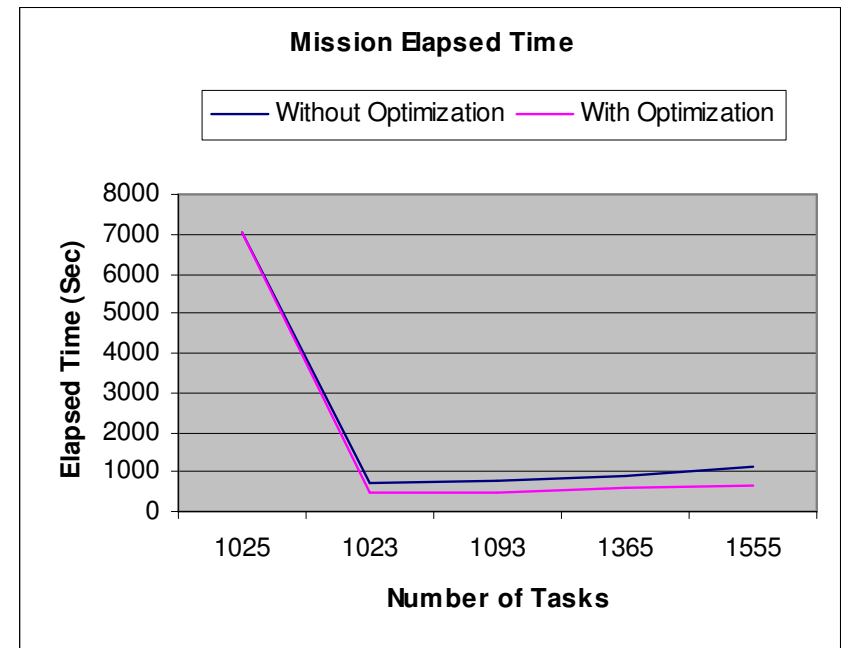
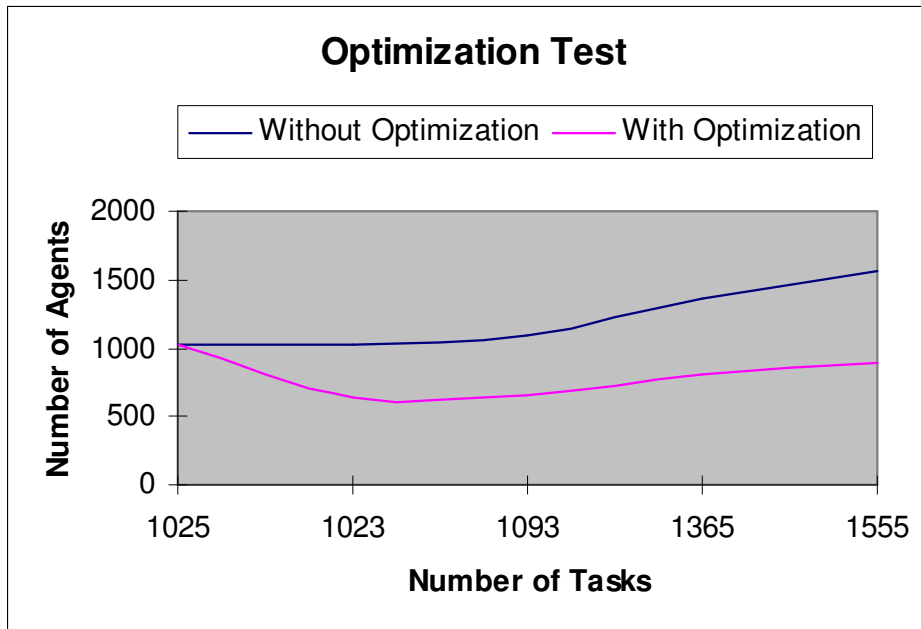
Test No	No Of Tasks	No Of Agents	Mission Execution Elapsed Time (Sec)	Average Time to Create An Agent (Sec)	Performance Decrease (%)
1	16	16	9	0.5625	0.00
2	31	31	20	0.6452	14.70
3	76	76	53	0.6974	23.98
4	151	151	108	0.7152	27.15
5	351	351	251	0.7151	27.13
6	751	751	541	0.7204	28.07
7	1551	1551	1111	0.7163	27.34



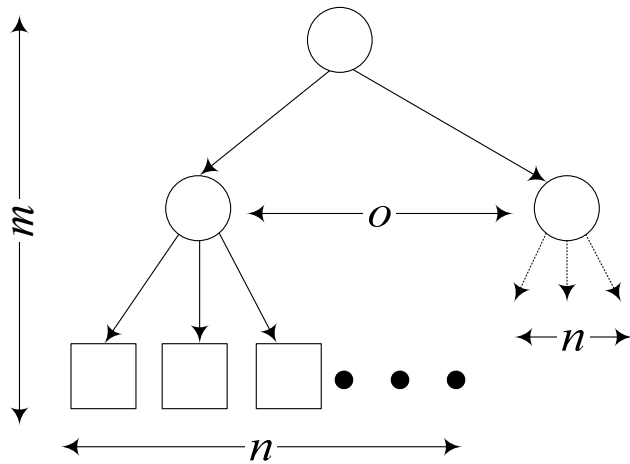
# Optimization Test



Test No	Plan Structure Parameters	No Of Tasks	Agent Count Without Optimization	Agent Count With Optimization	Agent Count Reduction (%)
1	m=1024, n=1	1025	1025	1025	0.00
2	m=9, n=2	1023	1023	633	38.12
3	m=6, n=3	1093	1093	657	39.89
4	m=5, n=4	1365	1365	814	40.37
5	m=4, n=6	1555	1555	889	42.83

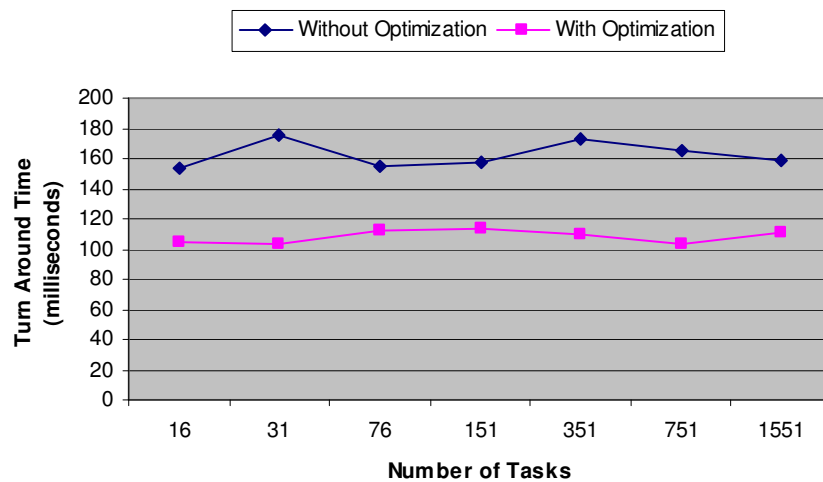


# MCA Load Test

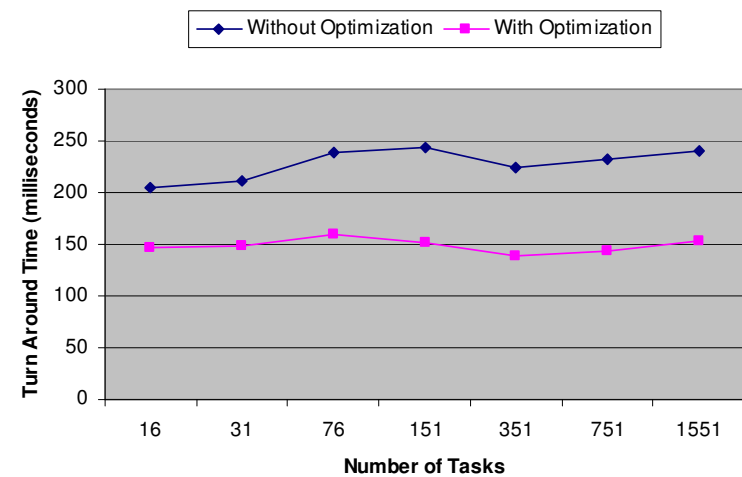


Plan Structure Parameters	No Of Tasks	Normal Load (mSec)		Heavy Load (mSec)	
		Without Optimization	With Optimization	Without Optimization	With Optimization
$o=5, n=2, l=3$	16	154	104	205	147
$o=10, n=2, l=3$	31	175	103	211	149
$o=25, n=2, l=3$	76	155	112	238	159
$o=50, n=2, l=3$	151	158	113	244	151
$o=50, n=2, l=4$	351	173	110	224	138
$o=50, n=2, l=5$	751	165	103	233	143
$o=50, n=2, l=6$	1551	159	111	241	153

Average Message Turn around Time - Normal Load



Average Message Turn around Time - Heavy Load



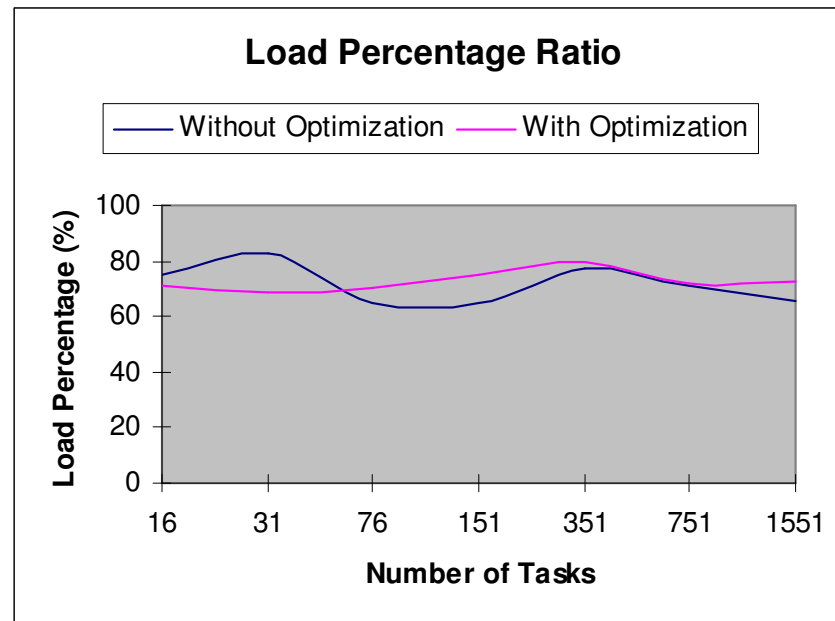
# MCA Average Load Test

No of Tasks	Without Optimization		Normal Load/Heavy Load (%)
	Normal Load	HeavyLoad	
16	154	205	75.12
31	175	211	82.94
76	155	238	65.13
151	158	244	64.75
351	173	224	77.23
751	165	233	70.82
1551	159	241	65.98

(a) Without Optimization

No of Tasks	With Optimization		Normal Load/Heavy Load (%)
	Normal Load	HeavyLoad	
16	104	147	70.75
31	103	149	69.13
76	112	159	70.44
151	113	151	74.83
351	110	138	79.71
751	103	143	72.03
1551	111	153	72.55

(a) With Optimization



# Related Work

## ■ TAEMS

- The major influenced to the TDG model.
- Rich of features that we don't use for our purpose, hence an overkill
- Carrying alternative plans as opposed to allow run-time plan modification
- Plan is static, not dynamic

## ■ Desire

- Influenced TDG w/r to maintaining task decomposition structure
- Uses explicit input/output connector to support data flow (as opposed to MDS)
- Tightly coupled between agents and tasks

## Related Work – cont'd

### ■ Zeus

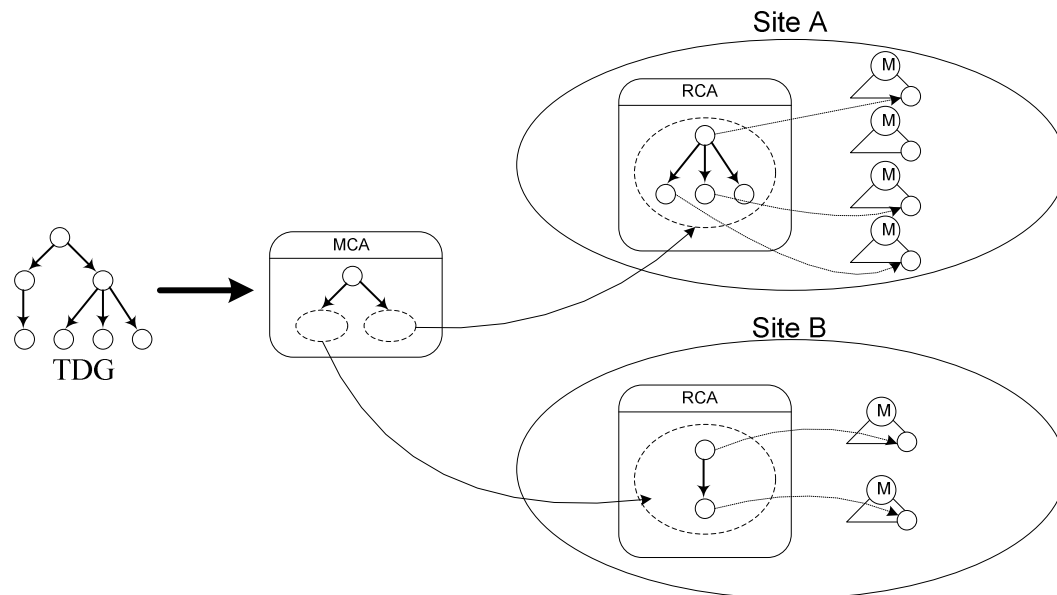
- Agent Development Toolkit
  - Does not understand about task decomposition, planning, agent generation.
- Strict requirements
  - Knowledge on software agent
  - KQML or FIPA ACL
  - Java™ programming language
  - Coordination protocols: Contract-Net and Master-slave

# Conclusion

- A concept of generating MAS dynamically
- A notion of on-demand and just-in-time agent generation
- A notion of a mission
  - Mission execution using the strata concept
  - Mobile mission
- Benefits:
  - Agents are determined by run-time needs of a mission
  - Integrated run-time adaptivity and robustness
  - Layering abstraction in MAS
    - Agents become disposable apparatus vs total assets
    - Concentrate on the goal as opposed to how to achieve the goal

# Are We There Yet?

- Better optimization: vertical stratum
- More autonomy: what to do vs how to do
- Close integration with ontology
- Quality of Results Over Time (QROT)
- Distributed MCA, RCA (Remote Control Agent)



# List of Publications

- **Jayaputera, G. T., Zaslavsky, A. and Loke, S. W. (2005).** Approach to Dynamically Generated User-Specified MAS. *The Sixth International Workshop on Agent-Oriented Software Engineering (AOSE-2005)*, In conjunction with AAMAS-05 To Appear.
- **Jayaputera, G. T., Loke, S. W. and Zaslavsky, A. (2005).** Just in Time Mobile Agent Generation and Management. *The Fourth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-05)*, Utrecht, Netherlands, ACM Press, To Appear.
- **Jayaputera, G. T., Zaslavsky, A. and Loke, S. W. (2005).** An Assembly and Execution Shell for Multiagent Systems. *The 38th Annual Hawaii International Conference on System Science (HICSS-38)*, Hawaii, IEEE Computer Society.
- **Jayaputera, G. T., Zaslavsky, A. and Loke, S. W. (2004).** Run-Time Mission Evolution in Mobile Multiagent Systems. *The 2004 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT2004)*, Beijing, China, IEEE Computer Society.
- **Jayaputera, G. T., Zaslavsky, A. and Loke, S. W. (2004).** Mission-Based Service-Oriented Computing. *The Third International Conference on Mobile Business*, New York, USA.
- **Jayaputera, G. T., Loke, S. W. and Zaslavsky, A. (2004).** An Integrated Approach to Multiagent Applications in Mobile and Ubiquitous Environments. *The Third Asian International Mobile Computing Conference*, Bangkok, Thailand, Kasetsart University.

## List of Publications – cont'd

- **Jayaputera, G. T., Zaslavsky, A. and Loke, S. W. (2003).** Mission-Based Multiagent System for Internet Applications. *The Fifth International Conference on Enterprise Information Systems*, Angers-France, Escola Superior de Tecnologia do Instituto Politecnico de Setubal.
- **Jayaputera, G. T., Loke, S. W. and Zaslavsky, A. (2003).** Mission Impossible? Automatically Assembling Agents from High-Level Task Descriptions. *The 2003 IEEE/WIC International Conference On Intelligent Agent Technology (IAT 2003)*, Halifax, Canada, IEEE Computer Society.
- **Jayaputera, G. T., Loke, S. W. and Zaslavsky, A. (2003).** Adding Value to Marketplaces by Assembling Location-Aware Agents On-Demand. *M>Business 2003*, Vienna, Austria, Austrian Computer Society.
- **Jayaputera, G. T., Alahakoon, O., Cruz, L., Loke, S. W. and Zaslavsky, A. (2003).** Assembling Agents On-Demand for Pervasive Wireless Services. *The Second International Workshop on Wireless Information Services (WIS 2003)*, Angers-France, ICEIS Press.
- **Jayaputera, G. T., Zaslavsky, A. and Loke, S. W. (2002).** A Mission-Based Multiagent System for Internet Application. *A Mission-Based Multiagent System for Internet Application*, Monash University.