10th International Command & Control Research & Technology Symposium
McLean, VA, June 13-16, 2005

Policy-Based Command & Control

Jay Bayne, PhD §

Echelon 4 Corporation
1045 W. Glen Oaks Lane
Suite 202
Mequon, WI 53092-3477
+1.262.240.2956
jbayne@echelon4.com

§ Principal Contact

Raymond Paul, PhD

Department of Defense
OASD/NII
4502 7th St. NE
Washington, DC 20017
+1.703.607.0649
raymond.paul@osd.mil

Abstract

Governance of large-scale dynamic systems [of systems] requires a management ethos and an associated set of policies and tools competent to "allow" the system and its subsystems to achieve and maintain viability. At the same time, such governance must actively guide the system's course grain behaviors. The DOD's present effort at *transformation* and implementation of joint or unified enterprise command and control (JEC2) are attempts at establishing, incrementally, a more formal and agile form of network centric governance over its distributed and traditionally "stove-piped" military and agency systems. JEC2 is therefore both a philosophy of management as well as an operational set of policies and mechanisms that must find widespread support among the Services, US allies, DOD agencies and the industrial machinery that supports the DOD's strategic and tactical missions. Our thesis is that, in alignment with and support of Jeffersonian principles of governance, a successful JEC2 capability will allow the DOD to govern itself with greater agility, with greater operational transparency, and with greater productively in the utilization of its human and material assets; with the basis of such a system including specific *operational C2 capabilities* for the dynamic and real-time management of policies that are competent to guide collaborative behavior within the federated DOD enterprise. We present our policy-based JEC2 system model.

Keywords
Policy-based C2; JEC2; Unified Command; Joint Command

Introduction

In the pursuit of web-enabled electronic commerce a great deal of effort has been expended in the design and development of methods and tools for implementing enterprise operations. These activities require a higher level of specification, and are producing new modeling languages such as "business process modeling languages" (ref, BPML), "web services definition languages" (WSDL), and a significant effort at rationalizing across business units the context and meaning (ontology) of such formalisms[1]. The *semantic web*[2] is a standards-based effort to create a web ontology for internet-based services. The DOD in its efforts to streamline its

[1] This work is not explicitly concerned with development of system ontologies, but we note that an ontological specification may be required for describing policies that govern operational domains that exist among collaborating systems. Policies, for all practical purposes, are formal elements of ontological specifications. For a more formal treatment of ontology, please see http://www.w3.org/2001/sw/WebOnt/
[2] Ref: http://www.w3.org/2001/sw/

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

enterprise systems (ref, NCES, GES)[3] is also engaged in migration to a web-services (i.e., service-oriented, SOA) architecture for defining its information infrastructures.

In the DOD's warfighting space (C2, JC2, C4ISR, and battle management) transformation activities have differing requirements and operational pragmatics, but they too are benefiting from capabilities offered by web-services based infrastructures[4,5]. The distinguishing feature provided by web services is transformation from platform-centric system acquisition and deployment strategies[6] to thin-client intranet server-based deployment strategies. Our view of policy-based C2 is consistent with this transformational view. As such, we do not dwell here on deployment technologies, preferring to concentrate on the concepts and application pragmatics of policy-based C2.

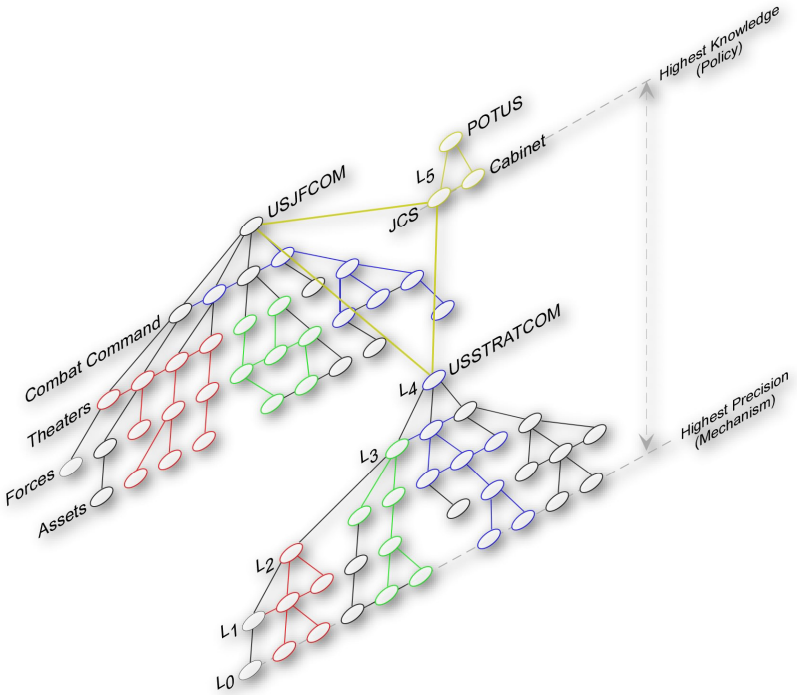

Figure 1 – DOD Policy Domain Hierarchy

Our treatment of policy-based C2 is based on several assumptions: 1) the DOD may be modeled as a system of federated systems; 2) one ontological view (at least one aspect) of DOD's policy domain is a containment (or accountability) hierarchy as depicted in Figure 1; 3) policies provided by a given level (e.g., L3, combatant command) are meant to govern behaviors at that level and subordinate levels; and 4) policies are the principle means of regulating system behaviors.

Example

Policy Directive 8100.1[7] is an example of a Level 5 policy associated with the DOD's transformation to network-centric warfare (NCW)[8]. Its purpose is to govern the definition,

[3] http://www.defenselink.mil/nii/doc/
[4] SOA and "web services" are used interchangeably to refer to software systems that utilize such web technologies as XML, WSDL, SOAP, and UDDI
[5] http://www.spawar.navy.mil/sti/publications/pubs/td/3168/td3168con.pdf
[6] Ref: http://www.disa.mil/main/prodsol/gccs_j.html
[7] Ref: http://www.dtic.mil/whs/directives/corres/pdf2/d81001p.pdf

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

adoption, acquisition, development and deployment of the global information grid (GIG) in DOD levels L5 and below.

Development and issuance of DOD Directive 8100.1 represents issuance of a high-level *command*, albeit one less specific and more strategic than what is typically meant by the term at lower tactical C2 levels of the policy domain hierarchy. Again, our thesis is that there are core C2 processes, especially related to *unified* and *joint* C2, that apply throughout the policy domain hierarchy and encompass the machinery for issuance and execution of direct and indirect (i.e., policy) tasking orders.

Federated System Model

Our model of a federated system requires a lexicon (ontological symbols) for describing its operational components, their capabilities, and their interrelationships. We begin with Figure 2, a representation of the command or accountability hierarchy, and the notion that within a federation each actor in the hierarchy is itself an independent self-directed entity. Independence is subject to the sovereign authority of the federation within which it operates. According to Jeffersonian principles[9] members of the federation are semi-autonomous and self-regulating. Their designs are required to allow them to 1) be viable and identifiable members of one or more federated communities of interest, 2) be governed by federation laws, and 3) to provide their individual contributions to coherent ensemble behaviors which characterize the mission, goals and objectives of the federated enterprise as a whole.

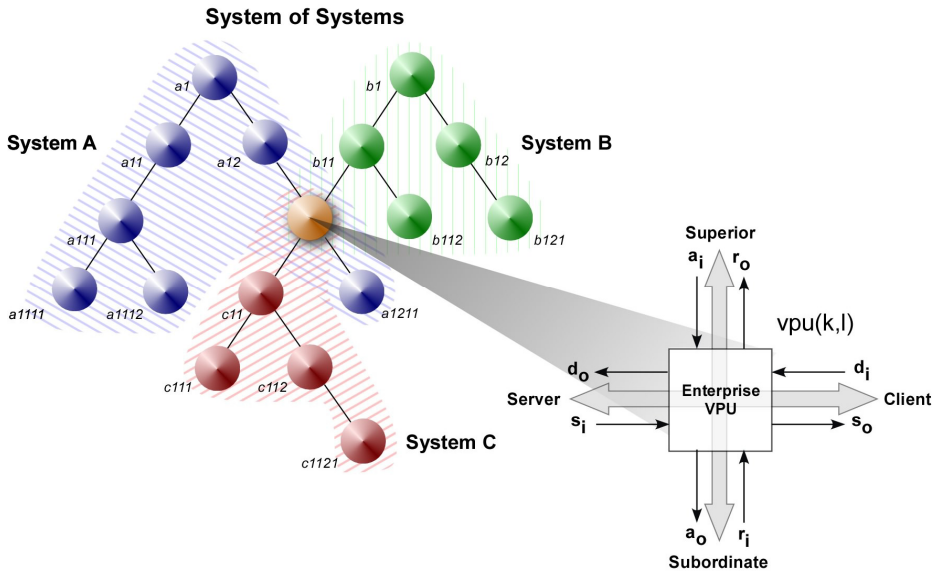

Figure 2 – Policy Domains of a Federated Enterprise System of Systems

Figure 2 diagrams a federated enterprise comprising systems A, B and C. We note that there exists a subsystem, a node labeled "Enterprise VPU"[10], that is simultaneously a member of all three federated systems. Each such VPU serves two value chains, a vertical asset chain (along the *command axis* of the enterprise) and a horizontal supply chain (along the *effects axis* of the enterprise). The asset chain threads the superior-subordinate accountability hierarchy, and the supply chain threads through the VPU's client-server effects production environment. The

[8] Last certified in November of 2003 by OASD/NII
[9] http://www.lewrockwell.com/vance/vance17.html (so called "axioms of a free society")
[10] We also classify an *enterprise* as a "value production unit," or VPU (ref. *Lexicon*, below)

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

fundamental objective of a VPU and its management team is to maintain its own viability[11] while striving to simultaneously serve demands on both chains.

VPU Model

A federated enterprise VPU is uniquely identified by its relative address within the federation. In Figure 2 VPU[k,l] refers to a value production process at the "l^{th}" level in the accountability hierarchy and the "k^{th}" position in its respective supply (e.g., logistics) chain. VPU[k,l] is *subordinate*, and therefore accountable to, VPU[k,l+1], and is a *superior* to, and therefore responsible for, VPU[k,l-1] in the asset chain. VPU[k,l] is a *server* or service provider, and therefore committed to, VPU[k+1, l], and a *client* of, and therefore dependent upon, VPU[k-1,l] in the supply chain.

An enterprise VPU may simultaneously participate in a number of *non-conflicting* value chains,[12] as diagrammed in Figure 1. As in Figure 2, VPUs interoperate (i.e., "collaborate") through four sets of communications ports, two for each for the two value chains. Arrows in Figure 2 associated with each port indicate the direction of the *flow of increasing value*. The function of each port is summarized in Table 1.

Table 1 – VPU Communications

Value Chain	Port ID	Port Name	Port Function
Asset Chain (Command Axis)	a_i	*Assets In*	Acceptance and assimilation, according to service-level agreements (SLA), of allocated assets and tasking orders from superior VPUs
	r_o	*Returns Out*	Production of returns on value produced by previously allocated assets or issued commands; requests for allocation of additional assets; clarification requests on issued tasking orders
	a_o	*Assets Out*	Issuance, based on SLAs, of assets and commands to subordinate VPUs with expectations for a time-bound returns of value produced
	r_i	*Returns In*	Acceptance and assimilation of returns and receipt and evaluation of requests for new asset allocations or readiness for new commands from subordinate VPUs
Supply Chain (Effects Axis)	d_i	*Demand In*	Receipt and acceptance of demand orders for goods or services from upstream consumer (client) VPUs
	s_o	*Supply Out*	Fulfillment (shipment) of previously received demand orders to downstream consumer (client) VPUs
	d_o	*Demand Out*	Issuance of demand orders for goods or services to upstream producer (server) VPUs
	s_i	*Supply In*	Receipt and acceptance of previously issued demand orders for goods or services from upstream stream producer (server) VPUs

We turn our attention to the policy-driven or policy-constrained manner in which enterprise VPUs are governed in their attempts to remain viable and to serve the demands of members of their communities of interest (i.e., their axes neighbors).

Policy-based C2

In order to discuss how C2 policies either drive or constrain behaviors, we require a definition of the "processes of C2." It is recognized that throughout the DOD enterprise there are many definitions of C2, some are abstract and directional (e.g. L3-L5), such as

> *Command and Control at its simplest is the exercise of authority and direction. Net-Centric C2 is the exercise of real-time authority and direction guided by the*

[11] *Viability* (survivability) refers to the VPU's ability of, on average, returning to its environment more net value than it consumes. It must do so on both axes along which it operates.
[12] We consider here only single vertical and horizontal VPU dependencies. Issues of VPU fan-out (marshalling) and fan-in (de-marshalling) are treated elsewhere.

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

> *commander's intent (command) and accomplished through an adaptable, decentralized, and cross-organizational arrangement of personnel, equipment, communications and facilities that are inter-connected and collaborate through a common shared information environment (control).*[13]

And some are specific to a given military Service or refer to productized tactical C2 systems (e.g., L0-L2)

> *A digital mobile ground C4I system, developed by […], provides tactical units and maneuver forces with access to images, maps etc, via ruggedized computers and hand held PDAs. The system enables field commanders to locate enemy targets and identify friendly forces in real time. […] introduced special algorithms for decision support, including assessment of optimal forces for rapid response, and prediction of shortest rout. The system can be employed in command vehicles and with dismounted troops.*[14]

While these definitions are valuable in historic and present terms, they are nevertheless inconsistent and provide only hints as to the functional elements of command and control *services*. We propose the following operational C2 model, one that encapsulates the two extremes given above.

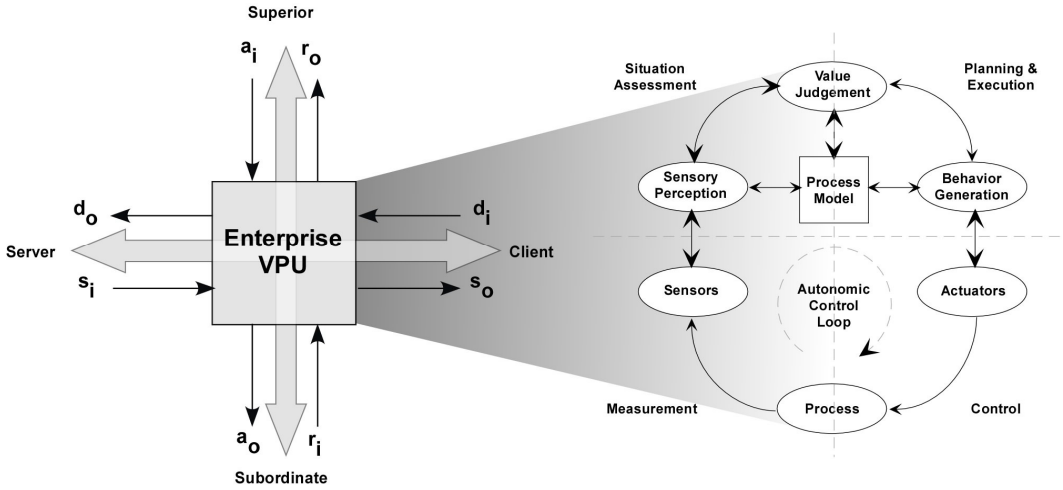

Figure 3 – Policy Governed C2 Process Steps

Figure 3 identifies the core elements of a *C2 application* that supports the VPU's mission of achieving and maintaining viability with respect to demands placed on it by its constituencies, the set of collaborating *allies* present on its asset and supply chains. The model is one with a basis in cybernetics[15], systems, informatics, and control sciences. The model is often referred to in the literature (esp. robotics) as an *intelligent controller model*[16]. It represents the classical feedback control paradigm with the added feature that it supports *value judgment*, whereby human or AI-based cognitive process may intervene in the autonomic control loop to support tuning, adaptation and learning.

---

[13] *Policy, Architecture, and Organization Framework Report (DRAFT)*, 01.16.04, OASD/NII
[14] See, for example, http://www.defense-update.com/products/
[15] *Cybernetics* is a systems science concerned with automation and control in natural and man-made systems.
[16] http://www.arl.psu.edu/capabilities/us_acis_intellcntrl.html

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

Autonomic control begins in Figure 3 with the *process* (under control, PUC) as diagrammed at the bottom of the figure. Through various types of sensors, measurements are taken periodically or aperiodically and, together with previous measurements and retained knowledge of the process's behaviors, are combined to provide *sensory perception* – the first step in the process of *situation assessment*[17].

The next phase of C2 processing is to turn this assessment of the current state into some type of controlled response or reaction. We refer to this phase of C2 as *behavior generation*, the act of generating an effective control or guidance that guides or drives the PUC towards some desired *next state*. The output of behavior generation is a "tasking order" that, when given to the effectors (i.e., actuators such as warfighters, emergency responders, divisional commanders, aircraft control surfaces), will deliver to the PUC the necessary controls.

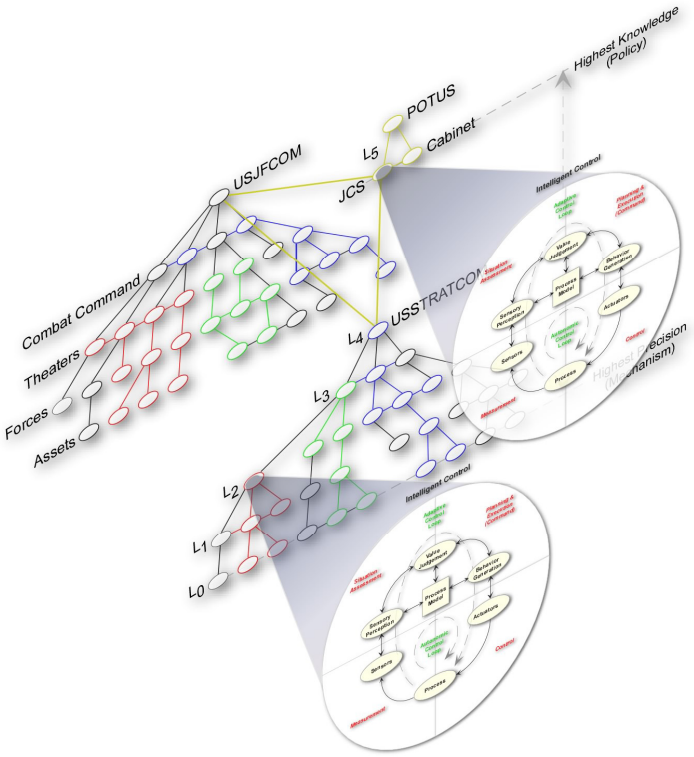

Figure 4 – Federation of C2 Policy Domains

As diagrammed in Figure 4, this C2 loop operates simultaneously at every node in the policy domain hierarchy, and is what is typically meant by the authority given to a manager of some activity. In general, these C2 controllers operate 24 hours/day, 365 days/year. Such "always on" management places sever demands on any overarching system that proposes to effectively administer its subsystems.

C2 in this context defines the act, profession and required tools of *management*. There are, of course, differences in the complexity, risks, resources and timing requirements of PUC and their associated C2 controllers at each level of the hierarchy. At L0 processes tend to be tactical, dimensionally compact (i.e., small state spaces), unfold at high rates, and therefore require

[17] This step is often referred to in DOD documents as *sensor fusion*.

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

higher bandwidth sensors and actuators. They may even be lethal and of high consequence. At L5 processes tend to be more strategic, dimensionally large (i.e., large state spaces), and unfold over longer time frames. They too may be lethal and are most certainly of even higher consequence. For integrated enterprises, policy-based C2 must scale along these dimensions.

Figure 4 implies a fundamental and critical requirement for distributed real-time C2 systems – their ability to *synchronize* (i.e., coordinate) along the federation's asset and supply chains. Synchronization requires that policy-based C2, in addition to its other objectives, be fundamentally about *scheduling* – both in time (e.g., rendezvous at a given deadline) and with respect to the effective sharing of resources (e.g., use of resource locks in implementing task resource scheduling). We leave the issues of generalized resource (asset) management to another paper. Here we shall focus on the general nature of scheduling in time and space.

Scheduling in Policy-based C2 Systems

We are interested in highly efficient and effective enterprise process control, whether at L0 where processes exhibit state transitions measured in milliseconds to minutes, to L5 where processes exhibit state transitions measured in hours or years. A well known axiom of systems science requires that measurement processes be able to sample the PUC at least twice the fundamental rate at which the PUC operates (i.e., changes state)[18]. This observational requirement places demands on the rate at which the C2 loop must operate at a given level of command if it wishes to remain "inside" the decision time of an enemy or a competitor, or inside the bandwidth of a physical process.

The manner in which C2 specifies and manages its timing requirements is therefore critical to the design of C2 systems. The treatment given in this paper is based on the theory of *time-utility functions* (TUF) and its associated *utility-accrual* (UA) scheduling methods, first proposed by E. Douglas Jensen, et al at CMU in the Archons Project, as reported in 1985[19], and advanced by Dr. Jensen[20] and others, and more recently in collaboration with Professor Binoy Ravindran at Virginia Technological University, and as documented by Dr. Peng Li[21] in his July 2004 PhD thesis[22].

Time-Utility Functions

The basic idea behind time-utility functions derives from problems with more traditional notions of "real-time" systems – systems whose computations must conclude on or before some specific point in time – a deadline. Classically, computing systems, especially embedded systems, accomplished deadline scheduling through the use of task priorities and some form of rate-monotonic scheduling theory[23]. While effective for closed, fixed function deterministic systems (e.g., automotive braking systems, flight control, etc.) where priorities are statically engineered and assigned, these techniques have proven less effective for open and probabilistic systems that must function under regimes where the workload is stochastic, varying in time due to demand, failures, and asynchronously driven mode shifts.

---

[18] The classical *Nyquist* sampling rate theorem
[19] http://www.real-time.org/timeutilityfunctions.htm; and http://www.real-time.org/referenced_documents.htm#rtss85
[20] Now at MITRE, Bedford, MA
[21] Now at Microsoft, Redmond, WA
[22] http://scholar.lib.vt.edu//theses/available/etd-08092004-230138
[23] http://www.ece.utexas.edu/~bevans/courses/ee382c/projects/fall99/forman/litsurvey.pdf

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

These so-called *mesosynchronous* systems[24,25] require that 1) we abandon traditional false dichotomies between "hard" and "soft" real-time systems, and 2) we develop functional means by which the scheduling of tasks incorporates application-level and run-time performance requirements, referred to as application-level qualities of service (AQoS). C2 systems are clearly mesosynchronous systems whose AQoS requirements are critical to the successful outcome of C2 decisions that affect VPUs in the subordinate command chain, especially in processing effects-based measurements of such outcomes and their re-assimilation in the next round of decisions and control.

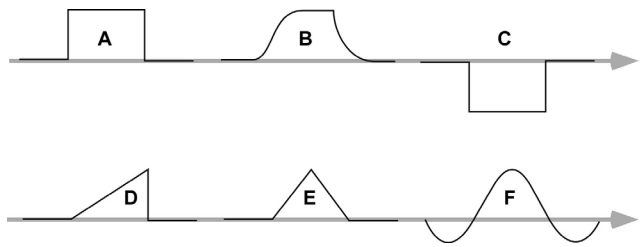

Figure 5 – Example Time-Utility Functions

A TUF is a function that assigns a value (utility) to a computation as a function of time, specifying its *completion time semantics*. Examples of a few such TUF functions are presented in Figure 5. (A) is a pulse function that specifies that the computation is uniformly valuable as long as it starts and completes between beginning and ending times. (C) by contrast states that the computation is uniformly valuable at all times except during the pulse window. And (D) defines a computation whose value monotonically increases, peeking at the apex of the ramp and immediately going to zero thereafter. In all three examples, the TUF defines a "deadline" as the latest point in time where the utility is maximized. Figure 5 (B) is a general case, whose properties are outlined in Figure 6.

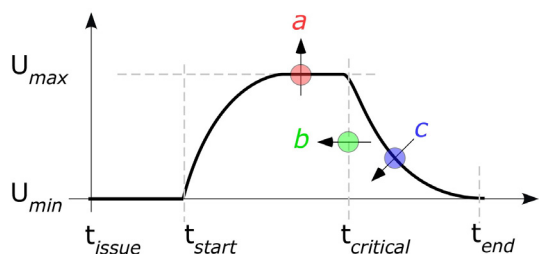

Figure 6 – Generalized TUF

The figure identifies a set of parameters that may be used to characterize the TUF, including the time the task is issued (t_{issue}), the earliest start time (t_{start}), the point in time (the traditional concept of a "deadline") where task completion yields the highest level of utility, $U(t_{critical})=U_{max}$, and the latest time when utility may be realized (U_{min} @ t_{end}). Raising the level of utility (a), shortening the duration between start and deadline (b), and shortening "utility decay time" following the deadline (c) all help to adapt the shape of the TUF to the situation-dependent (i.e., run-time) needs of the application. It should be noted that

TUFs are formal expressions of C2 scheduling policy.

[24] Mesosynchronous systems contain a mix of synchronous and asynchronous processes.
[25] http://www.real-time.org/docs/iccrts2004paper/css/iccrts2004paper_3.html

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

Policy-based Resource Management

Resource management, among its other duties, governs task completion time properties related to resource requests, assignments and releases. In general, the management of resources is less critical and more administrative when there are sufficient resources to satisfy the needs of all tasks wishing to execute. The issue arises when *overload conditions* exist – where the task pool's cumulative demand outstrips the system's ability to comply. It is precisely this situation that requires resource management in C2 systems. And it is this issue that motivates *utility accrual scheduling*. Our principal goal is to schedule those tasks, and manage their associated demands for resources, in a manner that can provide the greatest overall value (utility) to the enterprise.

Resources may be classified in two categories: *private* and *shared*. In either case, a resource may be *consumable* and either in the state *ready*, *exhausted*, and *replenished*. If the resource is *shared* we assume it to be *serially-reusable*. Our concern here is resource management of shared serially-reusable resources.

In general, each resource is required to have a unique ID. A resource, when allocated, is assigned to a single task that is designated as the resource owner. Ownership is time bound, with the bounds specified by the allocation *holdTime*, the time the task may hold the resource, and the allocation *abortTime,* the time when the resource is released by either voluntarily or forcibly aborting and then rescheduling the task.

Policy-based Task (Thread) Scheduling

In a fashion similar to a computing system, tasks are executed in our C2 system by *threads*, with these threads being the objects our utility accrual scheduling activities. Threads are modal, and as indicated above are executed in either a NORMAL or ABORT mode. ABORT mode is the thread state when it is executing its cleanup or exit handlers and releasing held resources; NORMAL is otherwise.

Threads are scheduled by allocation of a slice (period or epoch) of run-time, of which the time remaining is called *executionTime*. It follows that, for resource R currently held by a task T, its *holdTime(T,R) <= T.executionTime*.

abortTime denotes the time remaining prior to aborting a task and forcing it to relinquish its serially-reusable resources. Whenever a resource is requested (R, *holdTime*, *abortTime*) its *abortTime* is increased. When the resource is released, either voluntarily or triggered by the abort, it does so (i.e., "unwinds") in reverse (i.e., stack) order. A task's currently held set of resources is denoted by the list *HeldResource* = $\{<R_i, holdTime_i, abortTime_i>\}$.

The Schedule

The output of the scheduling algorithm is a *schedule* for allocating time to a C2 process step's currently active set of threads. A schedule is a set of *scheduleElements* = $\{<threadID_i, Mode_i, Time_i>\}$, where $Time_i$ is the time allocated to the thread during the next scheduling epoch. It is possible for the same thread to receive several such allocations within a given epoch.

Deadlock Handling

In systems with shared serially-reusable resources, deadlock[26] must either be avoided or, if it can occur, its potential occurrences must be managed. Our policy is to provide deadlock

[26] Deadlock is a condition where two or more processes, each holding one of a set of mutually desired resources, blocks and waits on the others to release their resources, effectively mutually blocking all processes

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

management through detection and resolution, rather than to attempt to provide deadlock avoidance. This policy is consistent with our view that C2 systems evolve and adapt, and effectively prevent sufficient detailed analysis *a priori* to support engineering a system to avoid all potential opportunities for deadlock.

In general, we employ a scheduling paradigm that, prior to allocating a working set of tasks their next quanta of execution time, searches their collective use of shared resources for opportunities for deadlock to occur. If found we then cull out those tasks, force them into ABORT mode, and then reschedule the set. This effectively forces those tasks to release their holds during this epoch, allowing then to reacquire during a subsequent epoch.

Utility Accrual Scheduling

As noted above, a proven[27] means of scheduling tasks whose completion time semantics are described by TUFs is referred to as *utility-accrual* (UA) scheduling[28]. In mesosynchronous C2 systems, UA scheduling provides a viable means of continuously evaluating and rescheduling tasks (in sequence, and in resource and execution time allocation) based on current situation assessments, available resource capabilities (e.g., capacity bottlenecks), extant policies, and other characteristics involved in the unfolding demands of an enterprise.

For a set of TUF-specified tasking orders (e.g., containing policy directives), the UA scheduling logic proceeds as follows:

Step 1: Perform deadlock detection and resolution for the working task set
Step 2: Create a potential schedule (ordered set of tasks) for the next epoch
Step 2.1: Build a task resource dependency graph
Step 2.2: Create partial schedules for interdependent tasks
Step 2.3: Remove the partial schedules containing deadlocks
Step 2.4: Determine the thread modes (to force resource releases & cleanup)
Step 3: Execute the schedule

Details of the algorithms for implementing this logic may be found in the references cited and, specifically related to policy-based C2, in a follow-on paper.

Control Processing Framework (CPF)

As introduced previously[29], the general nature of the cyclic control model (Figure 3) is implemented by a *control processing framework* (CPF). We shall describe the CPF in greater detail to show how TUF/UA applies to the processes and flow of C2. We begin by noting that a semi-autonomous system (VPU), even when federated (embedded) in a higher-order sovereign system, continues to strive to accomplish and balance two interrelated objectives: 1) planning and executing its own *internal* or self-directed tasks, and 2) incorporation and execution of tasking orders received from other members of the federation. As is typically the case, these two objectives may be in conflict and require periodic rationalization, for the enterprise may be constrained by:

- resource levels (e.g., dynamic capability fluctuations)
- policies (e.g., conflicting rules of engagement)
- time (e.g., deadlines and scheduling conflicts)

[27] http://www.real-time.org/docs/dcca94.pdf
[28] http://www.real-time.org/docs/words03fall(updated).pdf
[29] http://www.dodccrp.org/events/2004/CCRTS_San_Diego/CD/papers/007.pdf

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

Any attempt at designing a policy-based C2 environment that performs the functions denoted in Figure 3, especially one capable of scaling in a prescriptive manner from L0 to L5 must therefore be predicated on an ability to manage assets, policies and end-to-end timing constraints in a coordinated manner – coordinated vertically along asset chains and horizontally along supply chains. Our control processing framework (CPF) is designed to provide such management services.

Our approach is also predicated on the principle that *plans define activities that are expected to be aborted*. Dynamic reactive and learning systems must, based on the current state of one or more observable, controllable, and yet evolving situations, be able and willing to *change course*. Changing course is here equivalent to aborting the current plan or plan step and issuing a new plan or step, or alternatively, enabling the dynamic replanning of the entire plan of action. This capability provides the basis for the system's agile nature.

Our implementation of the control loop in Figure 3 is expressed by the CPF "application" diagrammed in Figure 5 below. The major elements of the CPF model and their functions are summarized in Table 2.

Table 2 – CPF Application Components (Ref: Figure 5)

Component	Principal Actors	Function	Input	Output
Situation Assessment (***SAS***)	E3/E4/E5	Enterprise Situation Assessment	Current domain capabilities; triaged and prioritized event and situation lists; events correlated to current plans of record (POR); arrival of new tasking orders	Prioritized list of potential courses of action (COA) with resource requirements and policy issues; updated domain situation model
Behavior Generation (***BGS***)	E3/E4/E5	Enterprise Behavior Generation	Feasible courses of action (COA); static policies; static resources	Updated set of resourced and prioritized plans of record (POR) ready for V&V and authorization by command
Execution Management (***EMS***)	E3 XO/COO	Enterprise Operations Management	Authorized POR/tasking orders, task execution status measurements in clients, suppliers, subordinates and superiors	Sequenced and synchronized execution of scheduled tasks
Policy Management (***RuleMan***)	E5 Commander CEO	Enterprise Policy Management	Current policy status; current asset status; current COA status; current POR status; superior's policy status	Updated domain Policy Database (PDB)
Command Management (***SuperMan***)	E5 Commander	Command Scheduling & Authorization	Recommended plans of action (POA), current executing plans of record (POR)	New and updated authorized plans of record (POR)
Model Management (***ModMan***)	E4 Navigator Planner	Enterprise Model Management	Subscription-based real-time operating situations and events; policy constraints; asset constraints; current situation model	Updated domain Model Database (MDB)
Planning Management (***PlanMan***)	E4 Navigator Planner	Enterprise Scenario Management	Policy base updates; asset base status updates; validated scenarios; current aggregate situation model	Updated domain Scenario Database (SDB)
Resource Management (***AssetMan***)	E3 XO/COO	Enterprise Resource Management	Current asset status; current policy status; current POR status; unmet COA requirements; subordinates' asset status	Updated domain Asset Database (ADB)
Performance Management (***MeterMan***)	E3/E3* XO/COO	Task Execution Monitoring	Continuous measurement of enterprise performance, including tasking orders, resource levels and utilization, etc	Continuously updated measures of potentiality, actuality, capability, latency, productivity, and performance

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

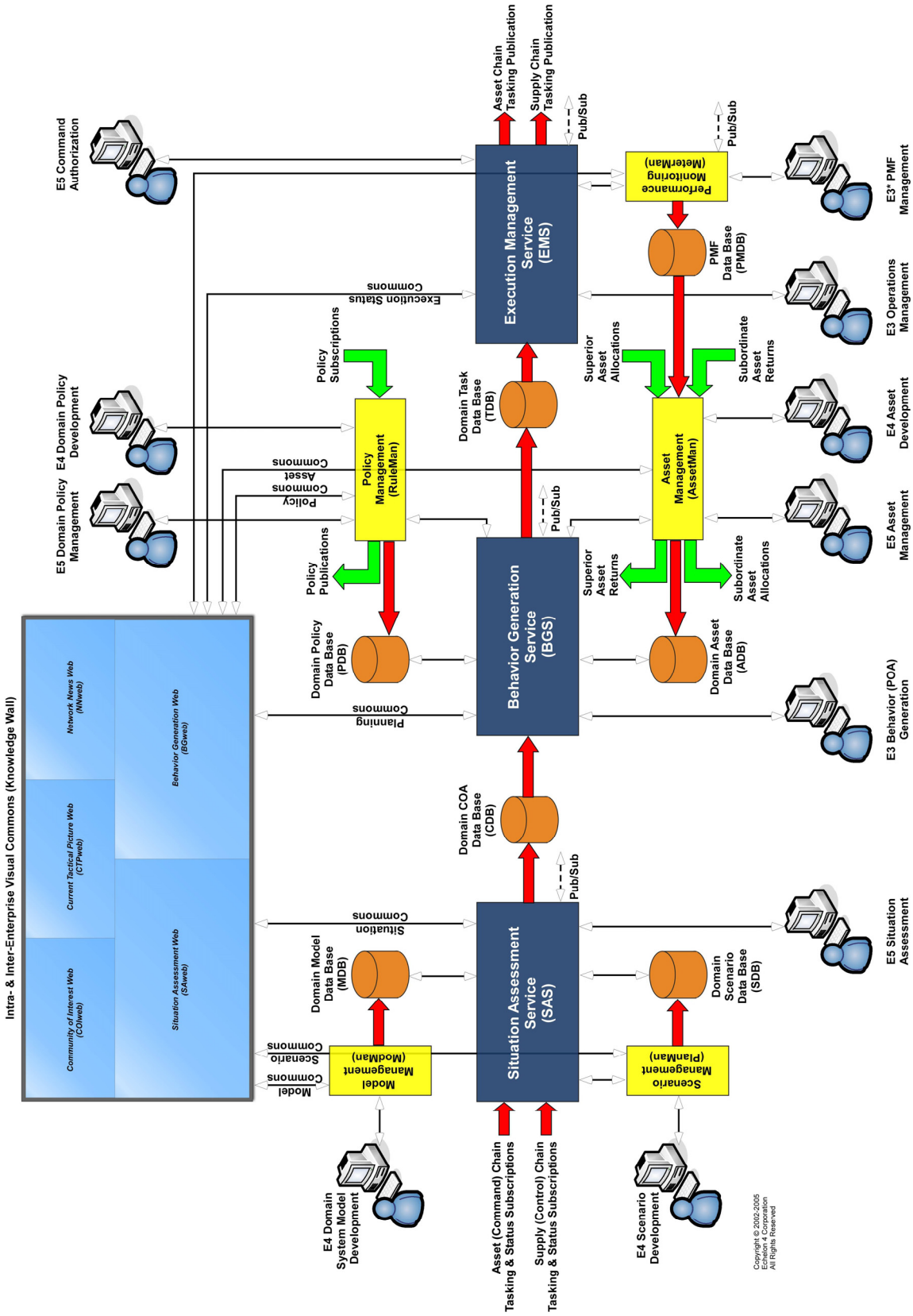

Figure 5 – CPF Application Structure

CPF Behaviors

For the purpose of this paper, we focus our attention on the three central components of the CPF, depicted in Figure 5 as *situation assessment services* (SAS), *behavior generation services* (BGS), and *execution management services* (EMS). To better understand the processes of C2 as outlined in Figure 5, we introduce a model for a command, *tasking order*.

Tasking Orders

For the purposes of this presentation and in order to following along with the CPF processing stages, consider a command, or tasking order, to have the structure depicted in Figure 6. Such a command may well be in the form of text (e.g., .xml, .doc, .pdf, or another popular encoding). Whatever its format, it conveys a *course of action*.

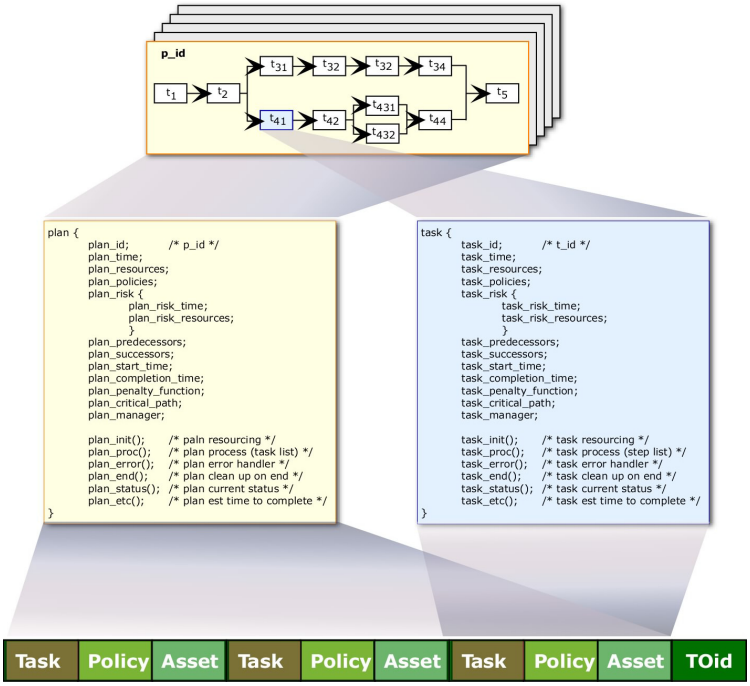

Figure 6 – Generalized Tasking Order Structure

In this form we note that a plan contains tasks (plan steps) that are implicitly or explicitly dependent on certain policies and certain assets (resources). For this discussion we have ignored other important fields in the structure, such as plan and task predecessors, successors, launch timing, fault recovery, etc. Notice that a task has the same structure of the plan containing it, so that our treatment scales and can benefit from the same processing functions regardless of the level from which the plan emerges.

Policy-based C2 Processing Framework

The tasking order (TO) in Figure 6 enters the VPU from some federated ally at the left of Figure 5. In order to follow its way through the CPF we need to look inside of the SAS, BGS, and EMS services. Figure 7 provides this next level of detail.

There are three primary aspects to the figure. First, we have identified the principal elements of each stage, and second, we have defined the interstage flows, both summarized in Table 3. And third, we have identified the interstage timing parameters that will be used in our scheduling work, as summarized in Table 4.

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

Table 3 – CPF Stage Functions & Flows

CPF Stage	Step	Function & Flow
SAS	**Filter Process**	Receive all messages from valid subscriptions, decode and sort all messages into classes, ordered by publication time and publisher ID, produce an event list (*elist*) for input to the **Triage Process**
	Triage Process	Receive the *elist* and, based on the current situation and the currently active plans of record, determine which events apply to known situations and which are new. Selectively ignore non-critical new situations; create a situation list (*slist*) and send it to the **Analysis Process**
	Analysis Process	Receive the *slist* and look for preplanned scenarios to respond. If present, adjust the scenarios to the current conditions. If none exist, create a new scenario to handle the new situation. For the one or more possible courses of action (COA) to the **Policy Process** in the form of a list of feasible responses (*clist*).
BGS	**Policy Process**	Receive the clist and evaluate the plans for compliance with extant policies. If compliant mark the plan as viable, if not evaluate risk and/or adjust the plan to allow compliance, if possible. If not possible, abort the plan. Forward all viable plans to the **Resource Process** in the form of an action[able] list (*alist*).
	Resource Process	Receive the *alist* and attempt to assign the needed resources. If resource conflicts exist between the *alist* plans, or between *alist* and currently executing plans, create one or more resource assignment schedules that allow for the greatest potential utility. Forward the new plans or record (POR) with schedules to the **Command Process** in the form of a plan list (*plist*).
	Command Process	Receive the *plist* and [re]evaluate the optimal schedule based on the current situation, the plist plans, and the status of all resources. With a valid (V&V) plan, authorize the new tasking orders and issue them to the **Execution Process** in the form of a task list (*tlist*).
EMS	**Execution Process**	Receive the *tlist* and allocate the task steps to affected subordinates, clients, suppliers and superiors. Continuously monitor the execution and adjust or issue new elements of the *tlist* as execution steps complete. Report on progress of *tlist* orders.
	Performance Measurement Process	(not shown)

Tasking Order Processing

Figure 6 depicts seven key steps in the CPF processing of tasking orders. There are other aspects not discussed, but these seven steps should provide a reasonably complete view of the essential mechanisms. The seven steps are identified, beginning at the left, by numbers in red circles.

Step 1..... The **Filter Process** receives a tasking order (TO) from a federation ally (a client, supplier, superior, or subordinate). The TO message contains at plan specification comprising one or more specific tasks, zero or more policy specifications, zero or more asset allocations, a timestamp declaring the TO publication time, and a TUF specifying the task completion time requirements.

Step 2..... Following filtering, triage and analysis, one or more candidate COAs that competent to respond to the TO are presented to the VPU commander at the **Policy Process** stage.

Step 3..... The **Policy Process** reviews the payload fields containing any plan policy specifications to determine, if present, whether a) they violate local policies, b) override local policies, c) augment or add to local policies, or d) are absent and therefore require application of local policies. The results of this policy check is to either a) accept the TO as is, b) reject the TO/COA and throw a policy exception, or

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

3) possibly update the TO/COA policy fields, accept the modified TO/COA, and forward it on to the **Resource Process** as a POA.

Step 4..... The **Resource Process** scans the TO/POA plan payload to identify all resource requirements. The TO/COA specified resources, but could not allocate them. This stage is responsible for allocation based on availability against the run-time requirements of currently executing plans (i.e., the *wset*). The **Resource Process** identifies all required resources, when they are required (with respect to any task step TUFs), what state they currently are in (e.g., assigned to an executing task), and any additional resource management requirements (e.g., resource replenishment). For each required resource, a *resource reservation* is made and noted in the associated plan payload fields. The resourced TO/COA is converted to a POR and forwarded to the **Command Process** for authorization.

Step 5..... The **Command Process** receives the resourced POA and, if accepted by the commander, is submitted to the **Scheduler** to attempt "fitting" it into the running system (i.e., assignment to E1-E0 processing steps). The **Scheduler** performs TUF/UA analysis for this POR and those in the *wset*, adjusting their individual TUFs to accommodate the insertion of the new TO in the schedule.

Step 6..... Following the rescheduling activity, the **Command Process** inserts the TO into the *tlist* for the next execution cycle, passing this new *tlist* to the **Execution Process**.

Step 7..... The **Execution Process** receives the new *tlist* and, through interaction with its subordinate services, forces any necessary resource reallocations and execution timing in order to execute according to the new schedules.

CPF Timing Considerations

Our emphasis in the paper includes scheduling aspects of policy-based C2. Scheduling in time is a critical element of any system that claims to be agile and adaptive. Clearly, the time it takes to perform the C2 activities of SAS, BGS, and EMS contribute to the effectiveness of the management of any given process.

As noted in Figure 7, the time it takes for the CPF to process information is characterized by its individual stage processing times, as defined in Table 4. These times govern a VPU's throughput, and to the extent they can be managed, the VPU's performance can be made predictable.

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

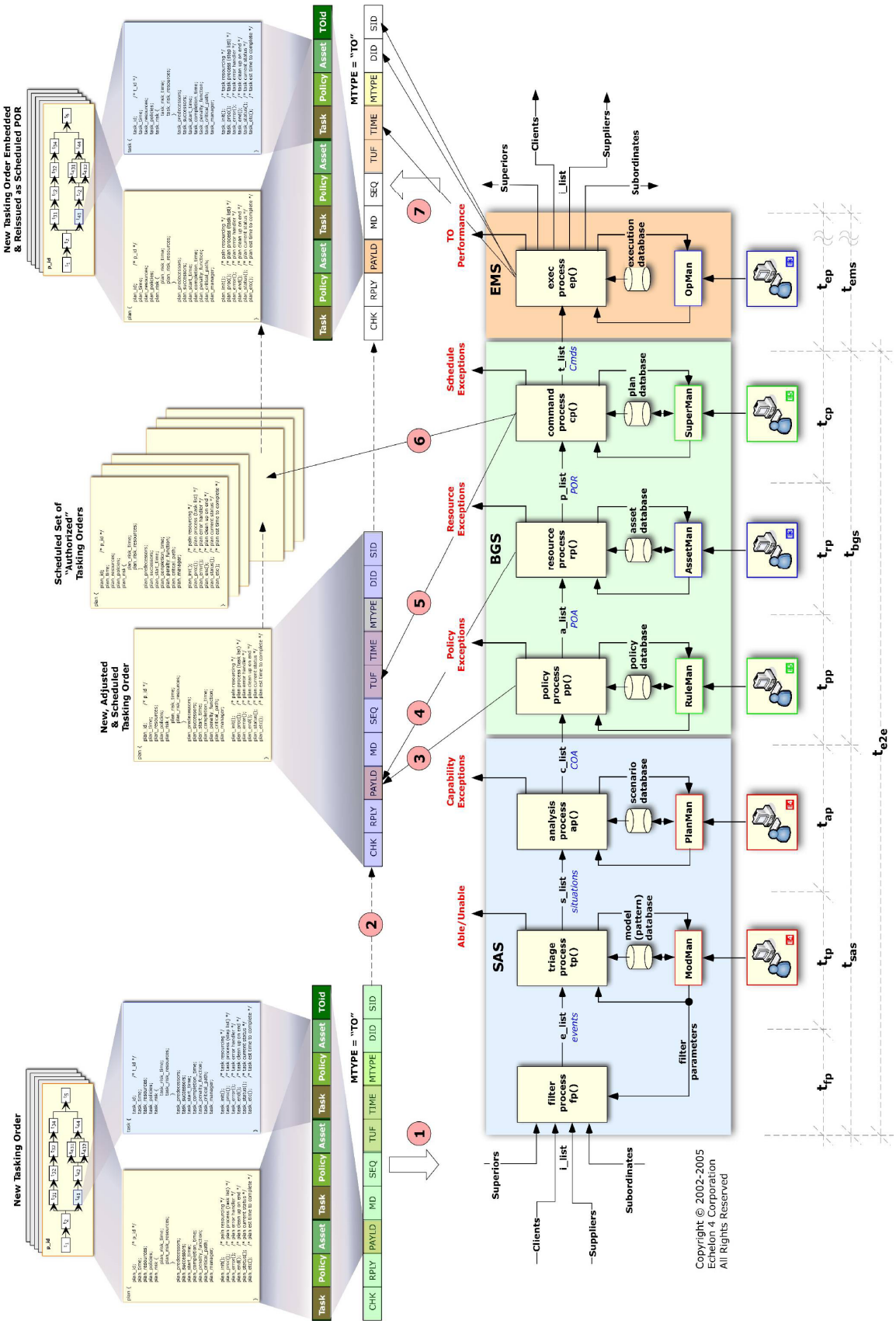

Figure 7 – Control Processing Stages

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

Table 4 – CPF Interstage Timing Requirements

CPF Stage	Step Time	Timing Issues
SAS	t_{fp}	Filter processing time: beginning at some time t_0, the time it takes to assemble all present inputs from its four input channels, filter and sort these inputs into various event classes (TO, INFO, ALARMS, etc), and to forward the resulting event list (elist) to the Triage process.
	t_{tp}	Triage processing time: beginning at t_0+t_{fp}, the time it takes to decide, given the current state of the VPU, which events should be allowed to pass into analysis stage, and the ordering of the events that do pass through.
	t_{ap}	Analysis processing time: beginning at $t_0+t_{fp}+t_{tp}$, the time it takes to a) recognize events related to situations currently under control of executing plans, b) recognize the occurrence of new situations for which a COA is required, to retrieve a relevant COA from the SDB and adjust its characteristics to fit the situation, or c) recognize the new situation and, not finding a predefined COA, to create one.
	t_{sas}	SAS stage time: $t_0+t_{fp}+t_{tp}+t_{ap}$
BGS	t_{pp}	Policy processing time: beginning at t_{sas}, the time it takes to check the proposed COA(s) for compliance to extant policies. If non-compliant, the perform a risk assessment (vis-à-vis, some penalty function) and to either adjust the COA to comply or provide for the penalty; or to abort the COA altogether and throw a policy exception. If compliant, forwarding POA to Resource Process
	t_{rp}	Resource processing time: beginning at $t_{sas}+t_{pp}$, the time it takes to identify all required resources, to attempt to assign resources according to a) POA TUF requirements, b) resource availability, and c) ensemble UA objectives. The result is a) a recommended POR schedule, b) a delayed POR schedule with open resource items, c) a delayed POA schedule with no resource commitments, or d) an aborted POA.
	t_{cp}	Command processing time: beginning at $t_{sas}+t_{pp}+t_{rp}$, the time it takes to include the new POR into the currently executing mix of tasking orders, including the time to compute the resource ABORT threads, the reassignment of resources released by the ABORT, and the adjustment of TUFs to meet evolving UA objectives.
	t_{bgs}	BGS stage time: $t_{sas}+t_{pp}+t_{rp}+t_{cp}$
EMS	t_{ep}	Execution processing time: beginning at t_{bgs}, the time it takes to execute a plan as part of a mix of executing plans under scheduling based on TUF/UA
	t_{mp}	(not shown)
	t_{ems}	Mean execution stage processing time, defining a VPU's actual performance (i.e., its PMF *actuality* measure)
CPF	t_{e2e}	Mean end-to-end processing time for the CPF services in a given VPU

UA Scheduling Objectives

For a given VPU in a C2 hierarchy there are three primary (and nested) scheduling frameworks. The outermost characterizes the role of an enterprise as a collaborative member within a federated group activity. At the second level defines the performance of the CPF within the VPU and the relative performance required of its processing stages. And the third level comprises schedules of tasking orders. Within the CPF, the TUF scheduler (Figure 8), under the direction of the VPU commander (E5), is responsible for the producing these nested schedules.

The TUF-based scheduling process takes as input the newly resourced *plist* and the current executing working set (*wset*). In addition, E3 provides the scheduler the VPU's current actual working set performance (*actuality* and accrued utility, UA) and its resource- and policy-dependent *capability*. The output of the scheduler is a new tlist containing a new *tlist* containing a merged and rescheduled working set for the Execution Process.

By way of example, a snapshot in time of a set of nested and time bound schedules is shown in Figure 9. The figure derives from our CPF simulation work. The outer bounding window, describes the targeted t_{e2e} = [0,240] for a given VPU within its COI. Within this window the seven CPF stages can be seen, with the schedule for first six $t_{sas}+t_{bgs}$ = [0,140], and the last t_{ep} = [140, 240].

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

Finally, within the various stages individual situations can be seen unfolding, with their individual tasks ordered according to their TUF policies. For example, there are two POR in the working set (*wset*) in execution within the Execution stage. Their individual task steps are not shown. Note the relative utilities as expressed by the height of the TUFs.

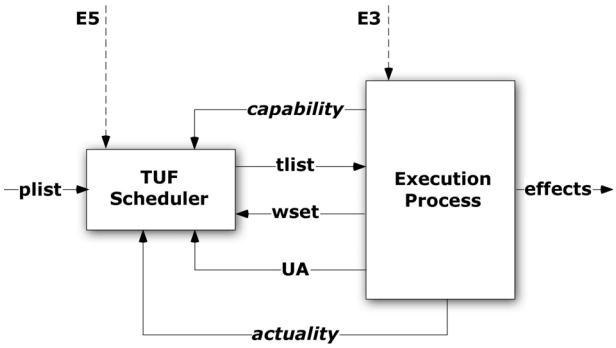

Figure 8 – TUF Scheduler

Our work includes methods (algorithms) for implementing TUF scheduling in complex situations, as shown above. For example, imagine that at some time t_0 the filter process receives four tasking orders with overlapping completion time requirements, as described in their respective TUF parameters. A function of the Analysis Process is to sort them in a manner that provides feasible execution ordering. To do so requires that the TUFs be analyzed to identify their minimum scheduling windows and their relative utility "weights." We define weight as the definite integral of the TUF over a period (i.e., scheduling window) defined by a *cut factor*. The *cut factor* defined for the analysis shown in Figure 10 is .95, or the period where the TUF is within 95% of its maximum utility. A more complete treatment of C2 scheduling using TUF/UA methods is the subject of a follow-on paper.

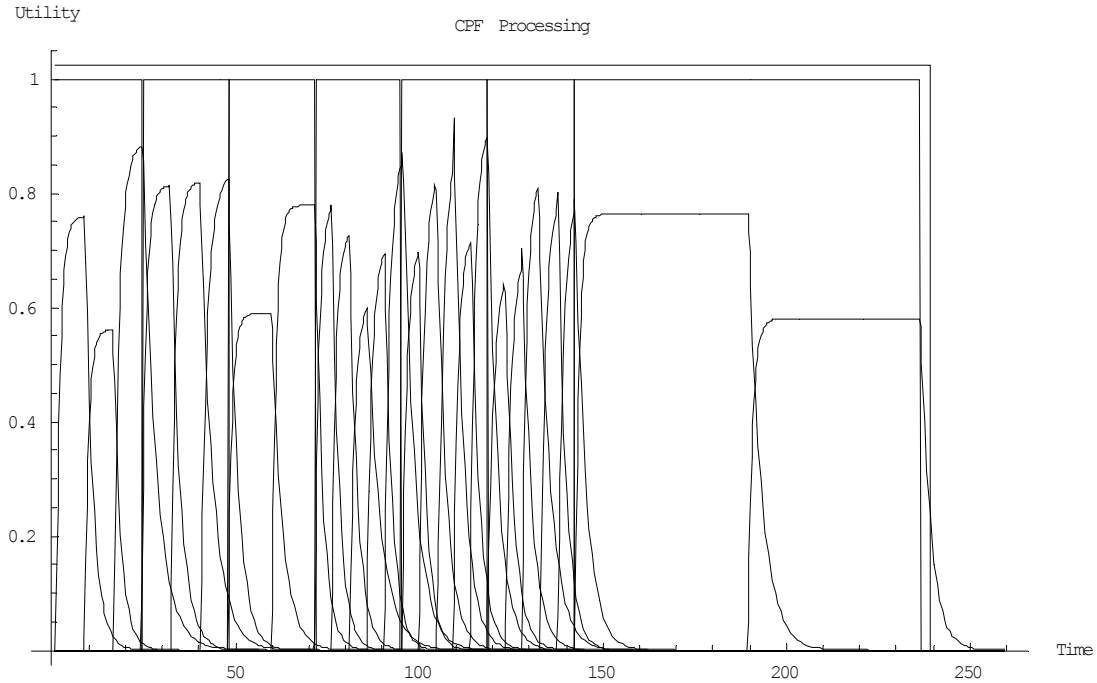

Figure 9 – Snapshot of a CPF Schedule

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

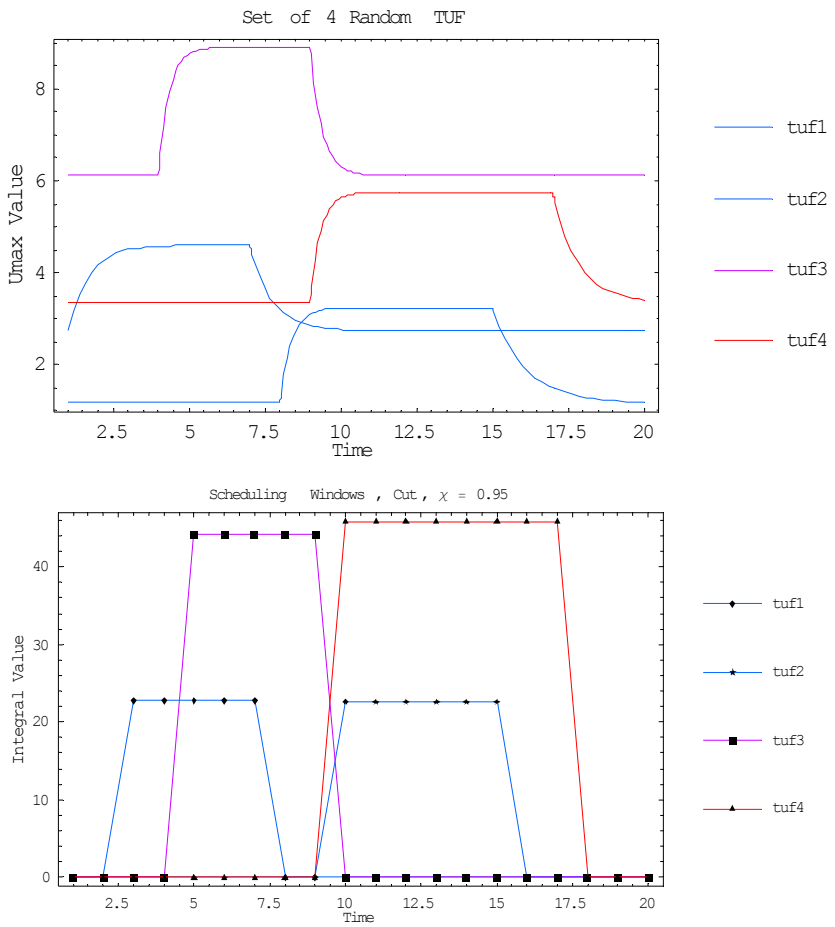

Figure 10 – Scheduling Windows

Conclusions

Our goal for this paper was to first introduce a context and lexicon for discussing the processes of C2 in distributed real-time command and control for federated systems. Secondarily we presented a functional model of a C2 processing framework (CPF) and its seven operational stages. Third we discussed a means of scheduling tasks based on time-utility functions (TUF) and associated probabilistic utility accrual (UA) scheduling model that attempts to provide CPF schedules that meet most of task end-to-end deadlines most of the time. Lastly, we provided an introduction to the methods by which TUF and UA schedules are developed.

This work is part of an ongoing effort at realizing a scalable grid-based C2 system that can be deployed in DOD, DHS and commercial-industrial settings.

Lexicon

Context ..... Policy domains govern behaviors in an operational commons containing objects and actors. The collection of objects and actors, including their states and their capabilities, define the operational "context" of a federation. The context defines what behaviors are possible. Policies define what behaviors are allowed. Within a federated system (of systems) there are two contexts

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved

of interest, one describing the operational domain of the encapsulating federation authority and one describing the operational domain of each encapsulated (embedded) subsystem.

Enterprise ..... An arbitrary unit of organization designed for and endowed with sufficient capability to produce a quantifiable measure of value within its operating domain.

Federation ..... An organization among active entities (aka, agents, processes, operational units), freely formed by the entities on behalf of their enlightened mutual interests, for the purpose of creating a sovereign authority capable of governing the policy domain within which they may pursue these mutual interests.

Ontology ..... Ontology is a [formal] specification of the concepts, goals, objectives, objects, actors, policies, and context that define a *policy domain* within a [federated] community of interest. While ontology often is associated with the taxonomic classes of hierarchies of objects in a context, it is also the means for enabling discourse between and among agents. Agents "commit" to an ontology if their observable actions are consistent with the definitions in the ontology. Such commitment guarantees consistency, but does not guarantee completeness.

Policy Domain .... The environment ("commons") in which federated actors collaborate ("interoperate") is defined, in part, by a set of policies (a "code") that is competent to underwrite their conduct. The policy domain defines what behaviors are "allowed."

Policy ..... A line of argument, typically in the form of nested "IF<condition> THEN<consequence>" clauses, that govern the invocation of a course of action intended to influence or determine decisions, actions, context or other matters related to a situation.

VPU ..... "Value Production Unit," an enterprise modeled as a system agent or object that simultaneously serves often competing demands on asset and supply chains.

Copyright © 2002-2005, Echelon 4 Corporation
All Rights Reserved