12th ICCRTS

Adapting C2 to the 21st Century

Use of a Systems Information Broker to Aide in the Dynamic Interfacing of C2 Nodes

Networks and Networking
C2 Metrics and Assessment
C2 Technologies and Systems

Dagohoy H. Anunciado [STUDENT]
Dagohoy H. Anunciado
SPAWARSYSCEN 24226, San Diego, CA & Naval Postgraduate School, Monterey CA
53605 Hull St Bldg A-33
San Diego CA 92152-5001
619-553-5604/619-553-6025 FAX
doug.anunciado@navy.mil

Abstract

Missions assigned to military forces will change as world events occur. Recent events like the Indian Ocean Tsunami and Hurricane Katrina in the United States required a massive humanitarian effort that included military forces. Information about the event needed gathering, distributing, and analyzing to determine how best to use resources to help the people in the devastation. Once observers gather information, establishing communications is needed before information can be distributed. Command and Control (C2) node functions perform one or all of the tasks of information gathering, distribution, analysis, decision making, and distribution of decisions. C2 nodes in these situations will be mobile or fixed and will come and go as a mission unfolds. Interfacing of C2 nodes may be hampered when the interface mechanisms are not worked out before an event and would take time to manually work out, which delays rescue and relief efforts. This research defines a framework and methodology for dynamically interfacing C2 nodes to a C2 enterprise to accomplish large missions such as responding to operations other than war (OOTW), e.g., natural and man-made disasters, peacekeeping, and counter drug operations.[1] Regional conflicts and general war are other situations requiring C2 enterprise to accomplish a large mission.

1. Introduction (Motivation)

Missions assigned to military forces will change as world events occur. Events like the Indian Ocean Tsunami[2] devastated coastal regions of Indonesia, Sri Lanka, India, and Thailand, and also affected Somalia, Myanmar, the Maldives, Malaysia, Tanzania, Seychelles, Bangladesh, South Africa, Yemen, Kenya, and Madagascar. Hurricane Katrina in the United States created a storm surge that caused severe and catastrophic damage along the Gulf coast, devastating the cities of Mobile, Alabama, Waveland, Biloxi, and Gulfport in Mississippi, and New Orleans and other towns in Louisiana. Levees separating Lake Pontchartrain from New Orleans were breached by the surge, ultimately flooding 80% of the city and many areas of neighboring parishes for weeks.[3] Both events required a massive humanitarian effort that included military forces.

When these types of events occur, information about the event needs gathering, distributing, and analyzing to determine how best to use resources to help the people in the devastation. Once observers gather information, establishing communications is needed before information can be distributed. Command and Control (C2) node functions perform one or all of the tasks of information gathering, distribution, analysis, decision making, and distribution of decisions. C2 nodes in these situations will be mobile or fixed and will come and go as a mission unfolds. Since C2 systems may have pieces of information that a decision maker will need, interfacing with other C2 systems is necessary in order get a big picture for decision makers to formulate their decisions. The goal of this research is to present and implement a systematic method for the dynamic interfacing of C2 systems. The core of this method is an entity called the Systems Information Broker (SIB). The SIB serves as an arbitrator that will determine whether the interfacing is feasible, and as a uniform interfacing platform to support the interfacing of real-time and non-real-time systems. To aid in the interfacing feasibility, a pre-formulated set of methods will be determined that are predicted to yield interfaces among systems. The focus of this research is determining the constraints on the methods used in interfacing systems that will allow a static calculation that can predict that an interface between systems is feasible.

2. Basic Architecture and Framework

The goal of this research is to present and prototype a systematic method to facilitate the dynamic interfacing of real-time and non-real-time systems. The core of the method is an entity called the Systems Information Broker (SIB). We propose to breakdown the responsibility of the SIB into two parts with this research focusing on the first part.

1) SIB will serve as an arbitrator that will determine whether the interfacing is feasible by considering the satisfaction of timing constraints and resource usage.

    a) SIB serves as an arbitrator taking into account the reconfiguring issues involved in the enterprise of systems supporting forces and units used to fulfill a new mission. We will need to create a calculation or mechanism for determining whether the proposed methods for interfacing systems are schedulable as systems are dynamically added, deleted, and immigrated.

b) In addition, we will need to determine the optimality goals and constraints for the resources used on the interfacing systems. Before determining the optimality goals and constraints, we will need to determine the metrics and calculation mechanisms that determine the current resource use. Once current resource use is known, one can then chose optimality goals and constraints for the interfaced resources. A mechanism will be needed to adjust the resource use to comply with the goals and constraints when resources are being over used. The SIB will use these mechanisms to determine if resources are used properly and make adjustments to comply with goals and constraints, but since we will not have complete information of the global state of the interfaced systems these mechanisms will only give suboptimal resource use. The goal is to still provide effective use of resource, but not necessarily optimal resource use.

2) SIB will also serve as a uniform interfacing platform to handle data interoperability and timing constraints to support the interfacing among systems. SIB will be used in an operational mode and serves as a uniform platform to handle the scheduling of system interactions, and interoperability among systems.

Figure 1 is the framework of the systematic method to facilitate the dynamic interfacing of real-time and non-real-time systems.

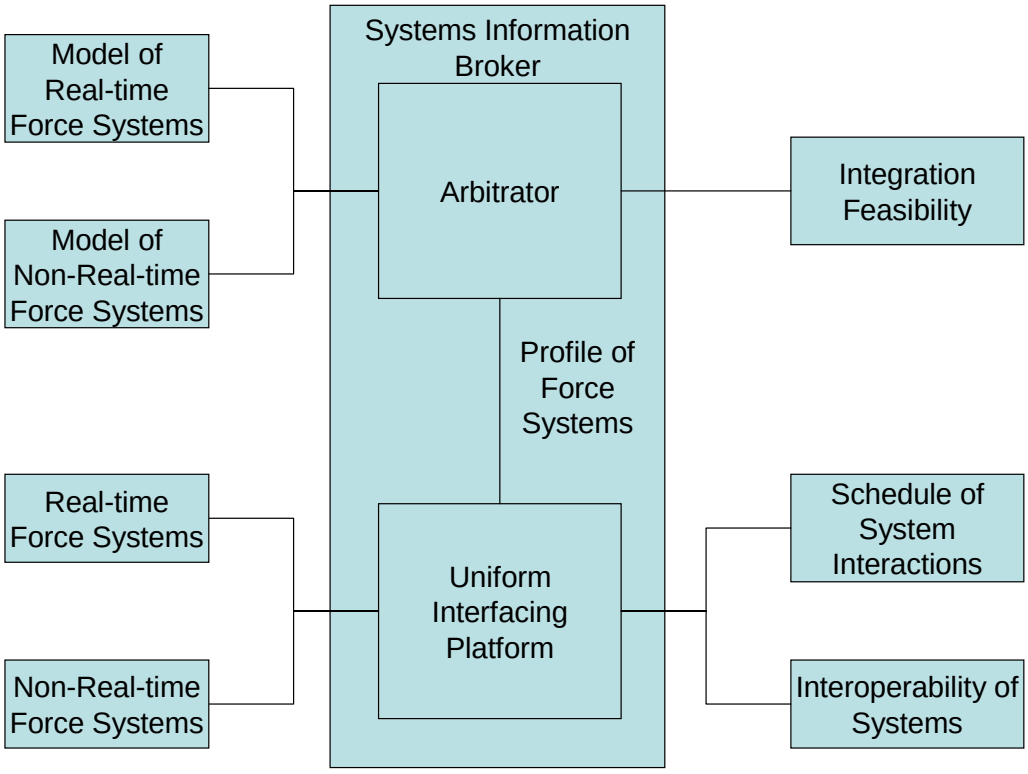

Figure 1 Framework of the proposed method

3. Mechanism for Determining if Systems Can Interface

We plan to develop a criterion that the SIB uses to determine whether interfacing systems is feasible. The SIB will use this criterion to determine if the enterprise of systems is schedulable after some systems are added, deleted, or immigrated from the enterprise. We imagine that this schedulability determination will be similar to embedded real-time schedulability but with a higher magnitude of timing constraint values due to network delays and jitter. The immigrating capability will be limited to non-real-time systems. To develop this criterion, a dynamic scheduling analysis algorithm for distributed systems with non-real-time and real-time tasks needs to be presented. Breaking down the mission the enterprise is required to accomplish into phases of operation and placing systems to address one phase may provide a way to bound the timing constraints that the systems handling a phase needs to meet. Calculating probabilities of moving from one phase to another may provide an additional means of bounding the timing constraints among real-time and non-real-time systems.

Another criterion that the SIB uses is to determine whether the enterprise of systems still has good resource usage after some systems are added, deleted, or immigrated from the enterprise. To develop this criterion, a resource usage metric needs to be defined and a method will be developed to compute this metric.

This framework will model systems with the base component being a single system. The model will not go much below the system level. A description of a system will include characteristics of the system and applications running on the system.

3.1. Alternative Methods for Interfacing Systems

The idea is to have several methods to choose from when interfacing systems. Methods will be ranked by scoring criteria that is explained in the next section, with the top scoring method being the primary interfacing method and the remaining methods as alternatives.

3.1.1. Modeling Systems Being Interfaced

We are working the model at a systems level where simple constructs and events are passed between systems. Modeling the systems with layers, and allow different layers of a system with past-through channels may allow for response times for the overall system not being hindered by individual layer transformations.

To create time-budgets with existing deployed applications and services we are going to need tools to measure resource usage using existing OS facilities. We are also going to need a method for determining excess resource capacity and heuristics to estimate it.

The excess capacity would be available to support the interfacing with other systems and ultimately all or a few tasks required to fulfill a mission thread.

Figure 2 illustrates the constructs used to interface systems together.

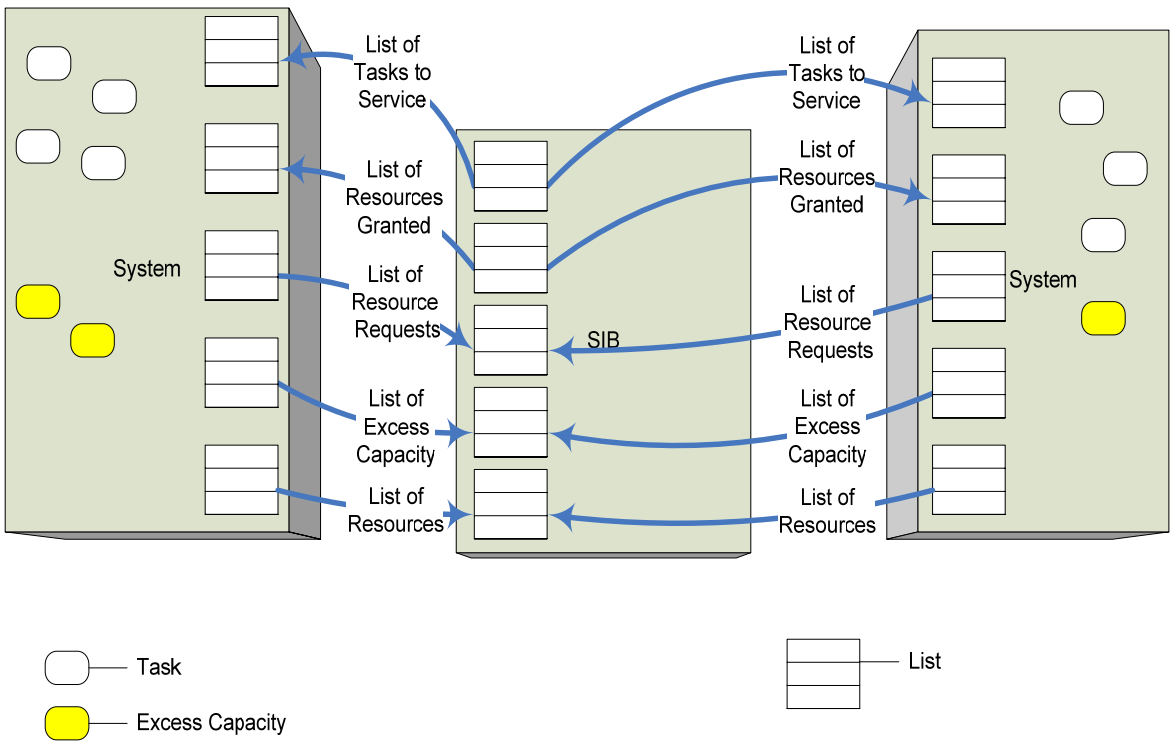

Figure 2 Modeling Constructs Used to Interface Systems

Systems would list out the resources it possesses and also list the resources it seeks. The system sends this list to the SIB for recording and future servicing. A list of excess resource capability would also be passed to the SIB. The SIB would coordinate the various lists of requested resources and map with available resources. The goal is to do this mapping without human intervention.

Applications used on a system vary in degrees of complexity from simple to very complex. On the complex end are applications that require many data sets, perform a high number of calculations on a subset of the data sets, and graphically render the calculated results. Without these data sets, desired results may not be precise enough to be useful. Many times developers will have an idea of what data their applications need to ingest, but these data sets are not well documented such as what the data is making up the data set, the way data is being gathered, what organization is maintaining the data sets, or how to get access to the data sets.

Table 1 contains the attributes that can determine the complexity of an application. An application that is real-time and performs any of the other attributes would be considered a very complex application. An application that is non-real-time, calculation intensive, data intensive, and graphical, or has only the real-time attribute is a complex application. A semi-complex

application would be non-real-time and has two of the three remaining attributes. A simple application would be non-real-time and has only one of the remaining attributes.

Table 1 Complexity Attributes for an Application

Complexity	**Real-time**	**Calculation Intensive**	**Data Intensive**	**Graphical**
	Timing constraints bounding computed results	Requiring high number of math calculations	Requiring high number of data points	Requiring high use of graphics to rendering information
Simple				X
Simple			X	
Simple		X		
Semi-Complex			X	X
Semi-Complex		X		X
Semi-Complex		X	X	
Complex		X	X	X
Complex	X			
Very Complex	X			X
Very Complex	X		X	
Very Complex	X		X	X
Very Complex	X	X		
Very Complex	X	X		X
Very Complex	X	X	X	
Very Complex	X	X	X	X

Without knowing what data sets or information one needs or an application, process, or system needs, one cannot perform processing with the data sets in order to get results from a formula or model. Before processing formulas or models, data is needed and depending on the use of the formulas or models, continuous processing of formulas or models may also require continuous access to the data sets or results to be relevant to a user. Relevant means that a user will be able to take actions to avoid harmful consequences.

Understanding what data sets are used is a good starting point to keeping an application relevant to its users. Keeping the data sets organized, and knowing who and where to get updated data sets also adds to an application relevance to a user.

Making the data sets organization simple to understand may make its maintenance easier.

Information base used to keep an enterprise of systems working may do well when the data sets making up the information base is organized.

Having a flat organization of data sets may be an ideal way to understand the data sets that an application or system will otherwise need.

3.1.2. Real-time and Non-real-time System Attributes

We will model systems by first separating real-time and non-real-time systems by their attributes. Table 2 contains the attributes used to model real-time systems used by a force.

Table 2 Real-time System Attributes

Attribute	Comment
Tasks	
CPU Cycles used	CPU cycles used to accomplish a tasks
Network resource usage	
Time Constraints: Finish Within or Maximum Execution Time	
Periodic, Event Driven, or Both for Task Execution	

Non-real-time force systems will be modeled with similar attributes shown in Table 3.

Table 3 Non-real-time System Attributes

Attribute	Comment
Tasks	
CPU Cycles used	CPU cycles used to accomplish a tasks
Network resource usage	
Periodic, Event Driven, or Both for Task Execution	

3.1.2.1. Breaking down the System Resources

Resources used by systems will be broken down into system resources and network resources. Systems resources are further decomposed into CPU resources, memory resources, and I/O resources. Tied with each resource are resource concerns that have the potential of diminishing the quality of service of the resource.

Table 4 and Table 5 below have the resources being modeled plus the resource concerns for each resource.

Table 4 Real-time System Resources Modeled

Resource	Resource Concern
System	
CPU	Lack of CPU cycles to complete a task calculation that will cause a task to miss its deadline.
Memory	Lack of memory causing a task to miss its deadline.
I/O	Waiting for I/O resources that causes a task to miss its deadline.
Network	
Bandwidth	Lack of Bandwidth that causes a task to miss its deadline.
Quality of Service	Jitter and Latency that degrade information flow and causes a task to miss its deadline.

Table 5 Non-real-time Systems Resources Modeled

Resource	Resource Concern
System	
CPU	Lack of CPU cycles prevents a task from completing its computations in a usefully timeframe.
Memory	Lack of memory prevents a task from completing its computations in a usefully timeframe.
I/O	Waiting for I/O resources prevents a task from completing its computations in a usefully timeframe..
Network	
Bandwidth	Lack of Bandwidth case prevents a task from completing its computations in a usefully timeframe.
Quality of Service	Jitter and Latency that degrade information flow and prevents a task from completing its computations in a usefully timeframe.

3.2. Scoring Criteria of Interfacing Methods

Interfacing methods are scored using criteria of the interfacing latency, capacity, and quality of service which includes availability and reliability. Other criteria may include cost of using the interface.

3.2.1. Computational Model for an Enterprise of Systems

Y. Qiao, et al., developed an admission control method for dynamic software reconfiguration in the systems of embedded systems (SoES) domain.[4] This method prevents dynamic software reconfiguration from damaging the high confidence of SoES. We plan use many of the concepts of the SoES admission control method to provide the computational model for the enterprise of systems (EoS) admission control method.

The EoS admission control method has to parts 1) modeling the systems making up the enterprise of systems and 2) the dynamic scheduling analysis for the EoS. The modeling of the systems mathematically describes the functional and non-functional aspects of the EoS requirements. This description has an external view model and an internal view model. The external view model is customer view focused, while the internal view model is designer view focused.

The external view model is denoted as ζ', and represented as

$$\zeta' = (G, H) \tag{1}$$

G is the functional emergent property vector that represents the functional aspect of the EoS requirements, $G = (g_1, g_2, \ldots, g_i)$, where $g_i (i \in [1, l])$. g_i denotes one of the functional emergent properties describing the emergent behavior of the EoS and l is the number of functional emergent properties. The most typical functional emergent property identified in the external view model is timing properties such as maximum response time.

H denotes non-functional emergent properties related to high confidence. It is described by a high-confidence metric vector. In this context, $H = (h_1, h_2, \ldots, h_i)$, where $h_i (i \in [1, z])$ is a set of metrics for a measure of high confidence. Some typical metrics are failure rate, maximum time between two successive failures, the number of faults that can be tolerated, maximum time between safety violations and security level etc.

The Internal view model is denoted as ζ, and represented as

$$\zeta = (S, E, C, D, F_1, F_2) \tag{1}$$

S is a component system set, $S = \{s_i \mid i \in [1, n]\}$. s_i denotes a component system constituting the EoS and n is the number of component systems in the whole EoS. E denotes the interaction sets between component systems, $E = \{e_{jk} \mid j, k \in [1, n]\}$, where e_{jk} denotes a set of interactions from component system s_j to component system s_k. C denotes the constraint sets on how the component systems are used in the given environment, $C = \{c_i \mid i \in [1, n]\}$. c_i is a set of constraints imposed on s_i. D denotes the constraint sets on interactions between component systems, $D = \{d_{jk} \mid j, k \in [1, n]\}$, where d_{jk} is a set of constraints applied to interactions in e_{jk}.

F_1 and F_1 are two mappings that refine emergent properties of EoS into local constraint sets imposed on component systems and interactions, i.e., $C = F_1(G,H)$ and $D = F_2(G,H)$.

In internal view model, timing constraints are included in C and D. Typical timing constraints include deadline and maximum execution time of the component system and latency of the interaction between two specific component systems. Furthermore, some resource constraints such as access mode and control constraints such as trigger method are also included in C and D. All these constraints can be extracted as parameters used by dynamic scheduling analysis for the EoS. In addition, each component system is either atomic or composite in internal view model. For convenience, to support the scheduling analysis, we take each atomic component system as a task to be scheduled by the scheduling algorithm.

3.2.2. Dynamic Scheduling Analysis for an Enterprise of Systems

3.2.2.1. Dynamic Scheduling Task Model

Since EoS are characterized by dynamic combinations of component systems, in this paper we only consider the aperiodic tasks. Y. Qiao et al., presented a task model for the use of dynamic scheduling analysis in SoES and we propose to extend this model for the dynamic scheduling analysis for EoS as follows:[5]

1) Each task T is described as a tuple $(a_T, r_T, D_T, v_T, E_T)$. Here, a_T is task T's arrival time and r_T is T's ready time. D_T denotes T's deadline. v_T is the number of T's different logic versions. E_T represents T's maximum execution time. For hard real-time tasks, E_T is a vector denoted by $(e_T^1, e_T^2, ..., e_T^m)$, where $e_T^j (j = 1,...,m)$ is the maximum execution time of task when it executes on processor p_j and m is the number of processors in the EoS. For soft real-time tasks, the maximum execution time E_T is a matrix denoted by $e_T^{ij} (i = 1,...,v_T; j = 1,...,m)$, where e_T^{ij} is the maximum execution time of task T's logic version i when it executes on processor p_j. Furthermore, for each $j (j = 1,...,m)$, the maximum execution time of each logic version is ordered such that $e_T^{1j} \le e_T^{2j} \le ... \le e_T^{v_T j}$. Non-real-time or best-effort tasks provided periods of execution time that can change at runtime. With non-real-time tasks the arrival time and ready time are the same, $a_T = r_T$ and the deadline equals the end of the provided period.
2) Hard and software real-time tasks are non-preemptive and non-periodic, and these tasks plus non-real-time tasks can not be parallelized.
3) Besides processors, tasks might need some other resources such as data structures, variables, and communication buffers for their executions. Every task can access a resource either in shared mode or in exclusive mode.

The characteristics of the task listed above can be extracted from internal view computational model addressed in Section 3.2.1. For example, the deadline and maximum execution time of a task can be derived from constraint sets imposed on the corresponding component system, and the access mode of a task can be derived in the same way.

3.2.2.2. Dynamic Scheduling Precedence Graph

We will use the same precedence graph concept as Y. Qiao, et al.[6], to represent tasks, but we will have two levels of representations. The first level will model task precedence at the individual system level, and the second level will model task precedence at the EoS level. Tasks requiring remote resources or remote execution on other system will have their resources allocating and dispatching coordinated through the SIB.

3.2.3. Maintaining System Schedule

Interfacing a system into an EoS must not interfere with an individual system's processing to meet local task schedules. An EoS schedule is first determined by the time-budget for the mission threads that an EoS supports. Development of time-budget allocations for time-critical mission threads is a recommendation of the Committee on C4ISR for Future Naval Strike Groups.[7]

The below tuple represents a mission thread time-budget.

$$(\text{MTID}, \text{TCM}, \text{MTD}, \text{R}) \quad (1)$$

MTID Mission Thread Identification

TCM Time to Complete Mission Thread

MTD Mission Thread Description

R Set of n Resources Needed to Accomplish Mission Thread, where $n > 0$.

A mission thread may be made up of multiple tasks and each task is represented by the following tuple.

$$(\text{TID}, \text{TCT}, \text{TD}, \text{TR}) \quad (2)$$

TID Task Identification

TCT Time to Complete Task

MTD Task Description

TR Set of Resources Needed to Accomplish Task where $\text{TR} \subseteq \text{R}$.

Resources needed to complete a task mainly include data sets, but may include computational, storage, networking, radio, and other physical resources.

3.2.4. Maintaining Optimality Goals and Constraints with Respect to Resource Usage

The goal is to use individual system resources to close to maximum, at least 80%, with 20% to surge processing. But the emphasis is to maintain the system processing schedule even at the cost of underutilized resource use.

4. Conclusion

This research provides a systematic method for dynamically interfacing systems to form an enterprise of systems. At the EoS level the SIB serves as an arbitrator for the tasks requiring remote resources or remote execution on other systems in the EoS. Given the attributes of the systems and remote resource needed by a task, the SIB will determine if the tasks and the systems making up a mission thread will be able to interface with the EoS and continue to maintain local tasks schedules.

---

[1] Joint Doctrine for Military Operations Other Than War, Joint Pub 3-07, 16 June 1995.
[2] 2004 Indian Ocean Earthquake, Wikipedia, http://en.wikipedia.org/wiki/2004_Indian_Ocean_earthquake, Accessed on 12 November 2006.
[3] Hurricane Katrina, Wikipedia, http://en.wikipedia.org/wiki/Hurricane_Katrina, Accessed on 12 November 2006.
[4] Qiao, Y., H. Wang, Luqi, and V. Berzins, "An Admission Control Method for Dynamic Software Reconfiguration in Complex Embedded Systems," *International Journal of Computers and Their Applications*, Vol. 13, No. 1, March, 2006, pp. 28-38.
[5] Ibid.
[6] Ibid.
[7] C4ISR for Future Naval Strike Groups, Committee on C4ISR for Future Naval Strike Groups, National Research Council, 2006, http://www.nap.edu/catalog/11605.html, Accessed on 13 November 2006.