A Structurally-Adaptive Framework for Distributed Airborne Sensing over Real-time Collaborative Information Sharing Networks

Corey A. Ippolito[1], Evan Kawamura[2], George Gorospe[3], Wendy Holforty[4]
NASA Ames Research Center, Moffett Field, CA 94035, USA

Keerthana Kannan[5], Vahram Stepanyan[6], Thomas Lombaerts[7]
KBR Wyle Services, Moffett Field, CA 94035, USA

Nelson Brown[8], A.J. Jaffe[9]
NASA Armstrong Flight Research Center, Edwards CA 93523, USA

Chester Dolph[10]
NASA Langley Research Center, Hampton, VA, 23681, USA

The emergence and maturation of wireless communication technologies continue to transform the aviation industry and are enabling new solutions to challenges faced by NASA's Advanced Air Mobility (AAM) initiative. AAM is leading towards high-density autonomous aircraft operations in areas underserved by traditional aviation, such as over densely populated urban centers. In this paper, we build on concepts from distributed sensing and smart spaces - where sensing, processing, and communication are embedded in an environment, and agents are operating within the space can exploit these capabilities in real-time through collaborative information sharing networks. Building from these concepts, we propose a framework to enable a dynamic, topologically-adaptive, and distributed estimation system for man-rated aviation to address challenges faced by autonomous AAM operations. This paper presents the initial concept of operations and system design for this framework, presents a mathematical formulation for abstraction of the problem, identifies requirements and constraints for operation, and presents algorithmic constructs to demonstrate operation. The initial framework design will focus on supporting precision navigation and independent surveillance supporting conformance monitoring of aircraft in airspace corridors and vertiport airspaces. Preliminary results from this framework shows promise in addressing gaps in current technologies needed to enable future AAM concepts, while promising greater capabilities, performance, robustness, and safety over current aviation systems and operations.

[1] Aerospace Scientist, Intelligent Systems Division. AIAA Senior Member.
[2] Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035.
[3] Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035.
[4] Intelligent Systems Division, NASA Ames Research Center, Moffett Field, CA 94035.
[5] Intelligent Systems Division, KBR Wyle Services.
[6] Intelligent Systems Division, KBR Wyle Services.
[7] Intelligent Systems Division, KBR Wyle Services.
[8] NASA Armstrong Flight Research Center, Edwards CA 93523.
[9] NASA Armstrong Flight Research Center, Edwards CA 93523.
[10] Aeronautics Systems Engineering, NASA Langley Research Center, Hampton, VA 23681.

I. Introduction

NASA's vision for Advanced Air Mobility (AAM) Mission is to help emerging aviation markets to safely develop an air transportation system that moves people and cargo between places previously not served or underserved by aviation – local, regional, intraregional, urban – using revolutionary new aircraft that are only just now becoming possible [1]. NASA's AAM efforts include Urban Air Mobility (UAM), the concept of expanding transportation networks towards short flight segments that can move people and goods around metropolitan areas at high volume and operating frequency [2]. Towards realization of UAM concepts, NASA has developed the UAM Maturity Level (UML) framework which categorizes anticipated evolutionary stages of a UAM transportation system into six levels, each of which represents incremental increases of maturity of the UAM ecosystem [3]. At advanced levels of maturity, UAM airspace system operations envision fully autonomous high-density operations occurring in and around densely populated urban centers. Realization of UAM concepts of operations will require state-of-the-art advancement in order to address challenges, barriers and gap in current aviation systems techniques and technologies [4].

One promising approach towards enabling advanced AAM operations is through application of smart space concepts applied to future airspace and future aircraft systems design [4]. Across many industries where systems historically were independent and minimally automated, an emerging trend in modern system design is to instrument, automate, and interconnect systems across real-time communication pathways. Modern large-scale dynamic systems are becoming massively interconnected networks of interdependent and interoperating systems-of-systems that fuse physical and cybernetical components. These cyber-physical systems are an inextricable fusion of physical and dynamic processes with embedded computation, algorithms, automation technologies, and communication. The resulting problems for control and sensing are often complex, difficult to express algebraically, and multidisciplinary in nature. Problems in this domain may have a mixture of continuous, algorithmic, geometric, and discrete properties.

A fundamental challenge for distributed sensing in a smart airspace system as envisioned in [4] and illustrated in Figure 1 is a decentralized framework for managing the needs of various users within this structure. Realization of a self-assembling self-configuring distributed sensing architecture that could fundamentally restructure and adapt to address the needs of this large-scale structurally-variable cyber-physical system would enable a number of potentially transformative new approaches to aviation system design, envisioning a system that can balance diverse local and global system objectives with greater performance, efficiency, resilience, with reduced required size, weight and power (SWaP) for onboard aircraft systems.

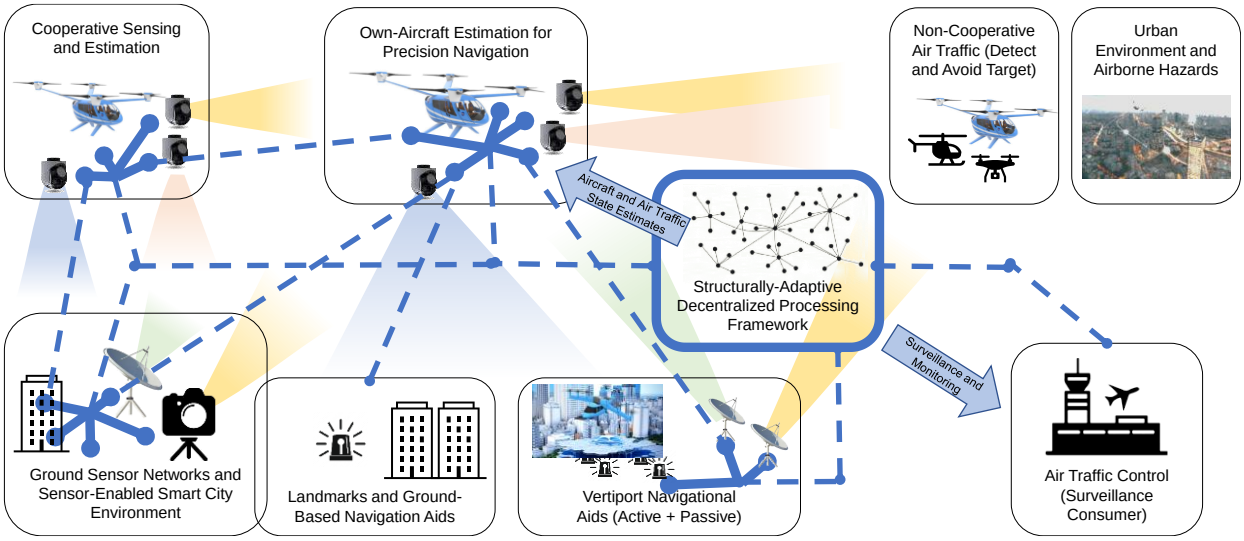

Figure 1. Concept for Distributed Sensing in a Smart Airspace System (from [4])

Desired characteristics for a framework to support distributed sensing in this domain were presented in [4]. The framework must be topologically dynamic; must allow for mobility and migration of data and processing across the system, must support a collaborative and market-incentived set of diverse users; must provide estimation in situations with a large set of overlapping sensors with partial observations of targets with varying levels of quality; must support a large network of 'smart-enabled' users (such as sensors, aircraft, and services); must handle complex communication

constraints that are characteristics of airborne communication networks, such as time-varying bandwidth, latency, intermittent connectivity, and variable quality of service; and must be capable of addressing needs for complex large-scale systems and networks with time-varying constraints.

In this paper, we present a preliminary design for a structurally-adaptive framework for distributed airborne sensing that can incorporate sensors and resources geographically distributed throughout a conceptual smart-enabled airspace. This paper presents a motivating scenario and use cases to identify the initial objectives, requirements and constraints. The scenario, derived artifacts, and resulting initial design of the framework are presented here. The preliminary framework design will initially focus on a single scenario with a set of use cases supporting two primary applications: (1) GPS-free precision navigation, supporting enroute and terminal-vicinity operations such as precision approach and landing; and (2) surveillance and conformance monitoring of aircraft in airspace corridors and vertiport airspaces. This paper presents the current status of this framework, presents the planned evaluation plans for simulation and flight testing, and describes potential future development directions for this system.

II.Framework Design

Consider the following motivating scenario shown in Figure 2. We pose a set of illustrative problems associated with this motivating scenario: (1) (Independent Surveillance) Provide continuous and persistent estimates of the state of aircraft operating within the airspace to a monitoring observer, with estimates that are independent of the sensors onboard the estimated aircraft (2) (Precision Navigation) Provide a continuous and persistent estimate of the full state of vehicle V1 to the vehicle sufficient for closed-loop flight control that may utilize sensors onboard V1 (i.e., higher rate and precision that surveillance, includes velocity estimates); (3) (Sensor Health Cross-Validation) Provide continuous evaluation of the health and accuracy of sensors S1 through S5 while the vehicle is in operation

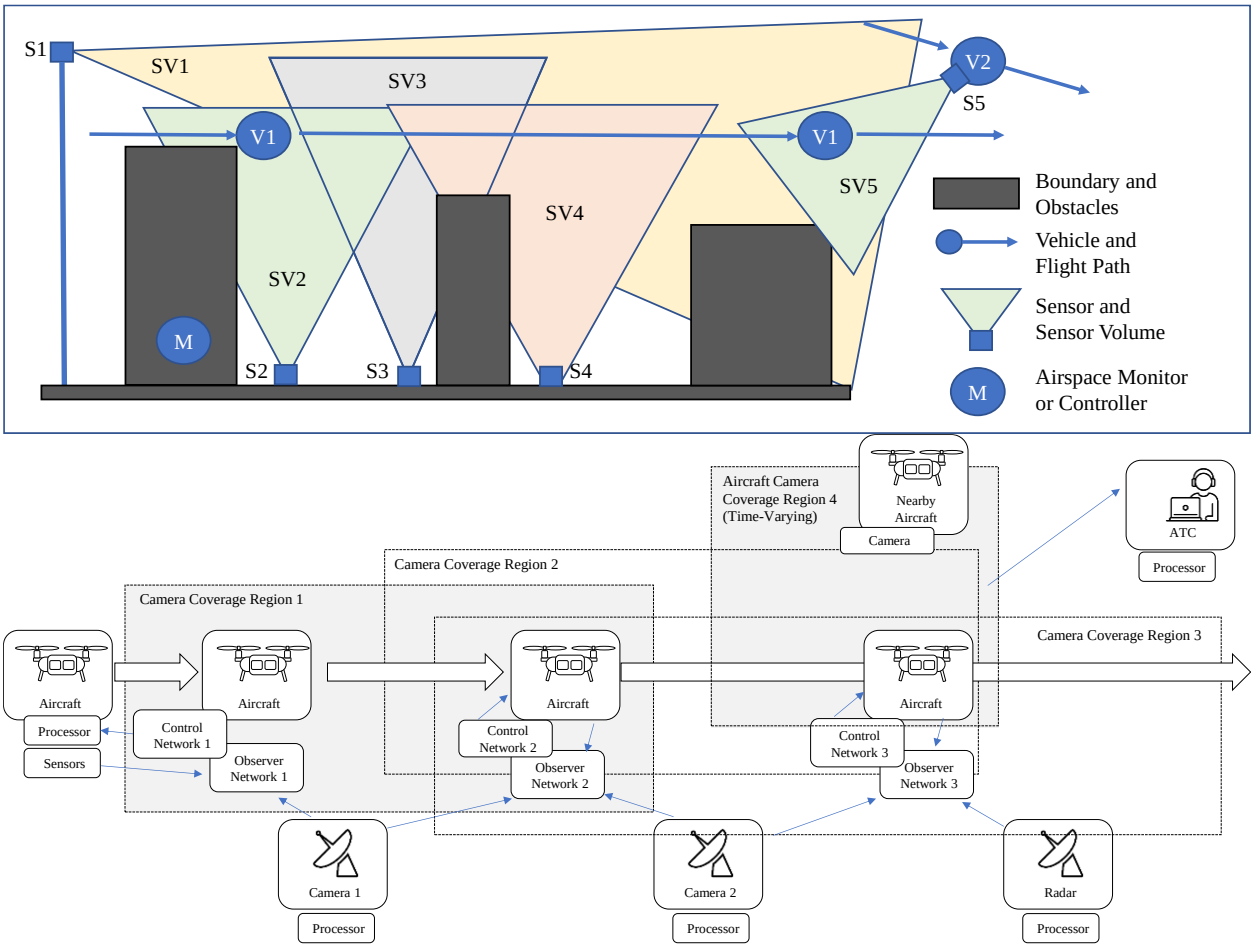

Figure 2. Motivating Scenario

A high-level design model of the framework is shown in the functional flow diagram in Figure 3. This model guides the framework through various operating modes supporting surveillance-type use cases. Referring to motivating scenario Figure 2, when a potential target is identified in one or more sensors, the framework enters an acquisition phase. The acquisition phase identifies the target as an entity to be estimated and confirms the target. The initial topology determination phase specifies an initial distributed sensing network topology over various elements in the airspace that meets the requirements and estimation goals while satisfying constraints. One the topology has been established, the system enters a nominal operational mode. During nominal operation, the target state estimate is updated at each time-step based on updates from the sensors. Asynchronously, the framework performs three other functions. The framework monitors the current network topology to identify any event that would require a topology change, such as the target vehicle leaving a sensor volume loss, loss communication from a sensor, etc. Additionally, the framework provides continual quality estimation across all sensors with overlapping observation of the target through cross-validation of each sensos with the consensus. While the target is estimated in the specified estimation network topology, the system will perform target classification to help classify the target for the consumers of the estimate. For instance, determining if the target is a sUAS, an eVTOL, a helicopter, or a bird will be useful information for airspace operators and for onboard separation assurance and detect-and-avoid functions.

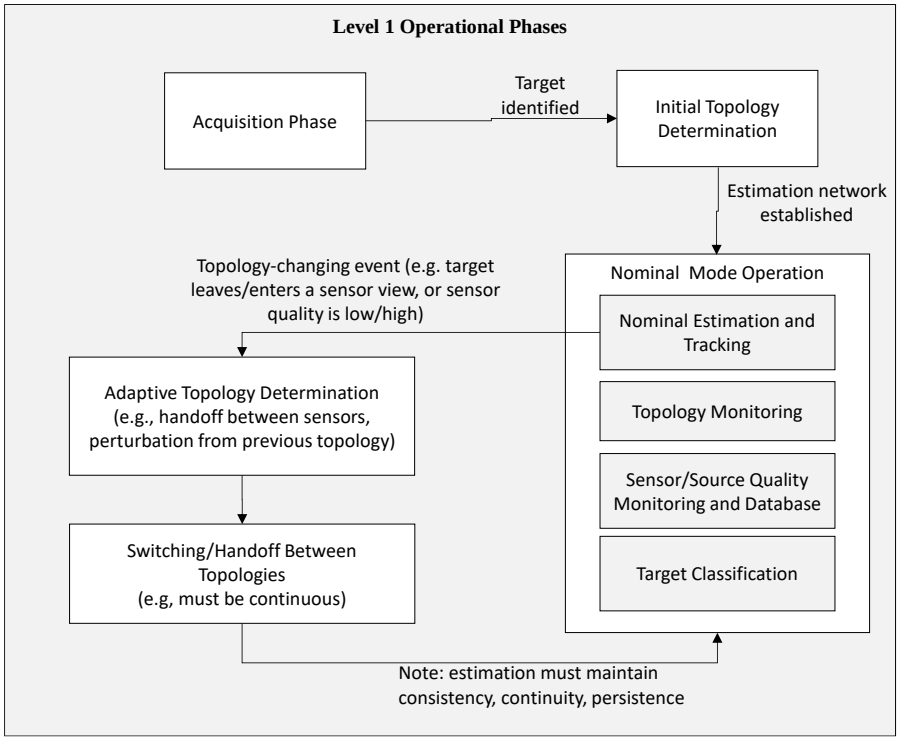

Figure 3. Distributed Sensing Framework Design – Level 1 Operations Model

A. Acquisition Phase

The function model for the acquisition phase of the framework is shown in Figure 4. The data from the sensors are processed to identify potential targets. The set of potential objects are identified and matched against the current database of identified objects. Detections are classified as either (1) an existing sensor detection of an existing tracked object; (2) a new sensor detection of an existing tracked object; (3) a new sensor detection of a new object; (4) noise or a false detection; or (4) a potential new detection without sufficient certainty to fit the other classifications.

The detection classification mode makes queries to an active database of tracked objects during matching. If the classifier identifies a new target object, the new target object is sent through an initial classifier and estimator. The initial classifier and estimator function creates a new target object entity for the tracked object database. This mode performs an initial type-classification of the target from the sensor data and identifies an initial state estimate. The type-classification, state estimate information, and correlation between the sensor and the target are added to the target object database entity, then adds this entity to the database, and this entity is passed to the next phase of operations (initial topology determination).

If the detection was classified as an existing target from a new sensor, then the associated target object entity information is queried from the database. This entity is updated with a new sensor association, and the effects on the topology are determined. If the estimation topology needs to be updated, then a topology change event is created containing information on the new sensor/target correlation. The topology change event is sent to system event queue execution of this mode completes. The topology change event is handled at the next opportunity as shown in the level 1 diagram in Figure 3.

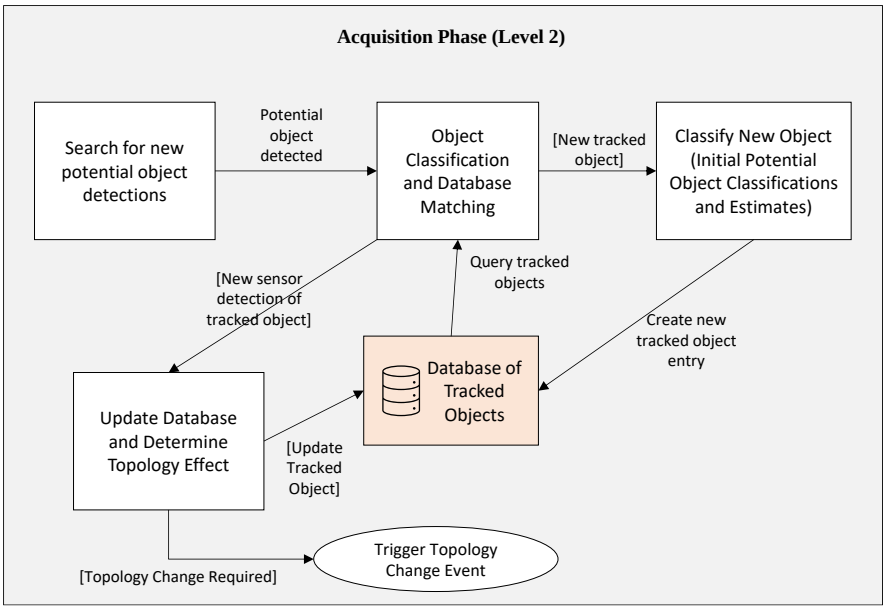

Figure 4. Acquisition Phase Model

B. Initial Topology Determination Phase

The initial topology determination phase of the framework will execute when the acquisition phase has identified a new target. This receives the following information: (1) information on a new object to track (a new entity in the tracked object database), the initial sensor estimates, the sensing goal, the initial type-classification, the list of sensors with observation, and the history of estimates from each sensors. Currently, the only sensing goal implemented is for state estimation supporting surveillance. The phase is responsible for establishing the topology of the sensing estimate. Currently this system only seeks satisfaction of constraints in determining the topology, though future work can include local and global optimization phases. Through local optimization, a feasible and optimal structure would be determined that can perform multi-objective optimization on the structure, optimizing for instance for minimal latency or minimal bandwidth utilization. Through global optimization, the global information flow over the larger network can be considered, taking into account for instance data flow required over all tracked object entities in the database. Global network considerations on the airborne network can be considered, such as load balancing for total-bandwidth minimization and latency objectives over all structures in the network.

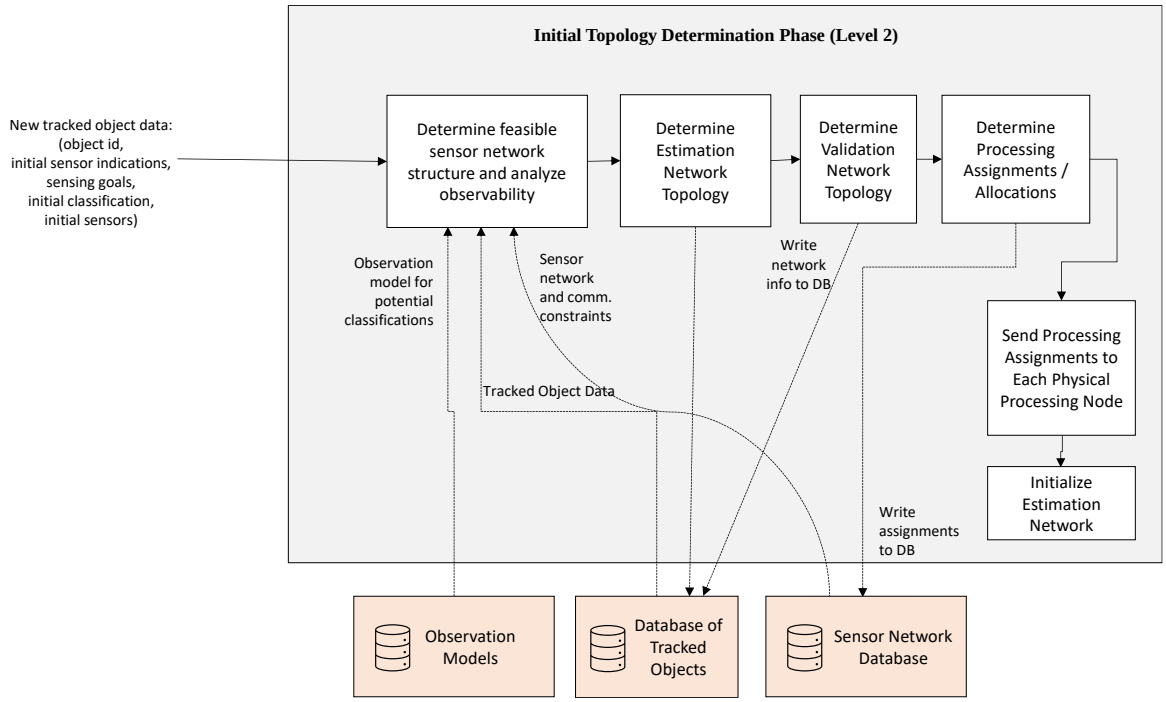

Figure 5. Level 2 Functional Requirements for the Topology Determination Phase

This phase will utilize information on the large-scale network contained in a sensor network database to determine topology of the solution in terms of the mathematical estimation and assignment to the physical distributed network. The sensor network database contains information on the physical processing nodes contained within the network, including current estimates on constraints such as bandwidth and latency over potential physical connections. During execution of this phase an estimation structure will be determined, a network and data flow structure will be determined, an assignment of processing elements to the network structure will be determined, and an initial estimate will be computed.

Consider the estimation problem shown in Figure 6. In Figure 6 (a) at some given time, consider a set of sensors that can provide estimates of a target to a monitor. The estimation topology solver determines the structure and details of the mathematical estimation network to observe the desired parameters. In Figure 6 (b), the illustrative estimation structure shown takes camera images as 2D arrays of RGB pixels in the image plane, independently estimates the location of the target within the 2D image, combines the two image space estimates with a filter, then produces the final object state estimate as a state vector of six elements (three position and three velocity components). Given the specification for the desired estimation network, the network topology must then be determined by assigning each element in (b) to physical processing units in (c). In this illustrative example, consider a hard-wire network connection between a communication relay network node N1 and an observer M, with poor wireless network performance between the observer node, the sensors S1 and S2, and the vehicle V1 due to obscured line-of-site. The potential interconnections between each node is a function of the state of the system – e.g., position and orientation of vehicles, location of sensor nodes, location of any available communication relays, and connectivity of service providers or information consumers - at any given time. In Figure 6 (c), the sensor network database provides the framework with information pertaining to poor connectivity between sensors and the monitor, and strong connectivity over wired links between the communication relay and the monitor. The assignment of processing elements in (b) to physical processing nodes in (c) is referred to as the data migration problem. Once the initial topologies are determined, the processing algorithms must be distributed to each processor throughout the network and initialized. Once initialized, an estimate is computed through this structure and stored as the initial estimate. The resulting structures, assignments, and results from the estimate are written the tracked object entity and written back into the tracked object database.

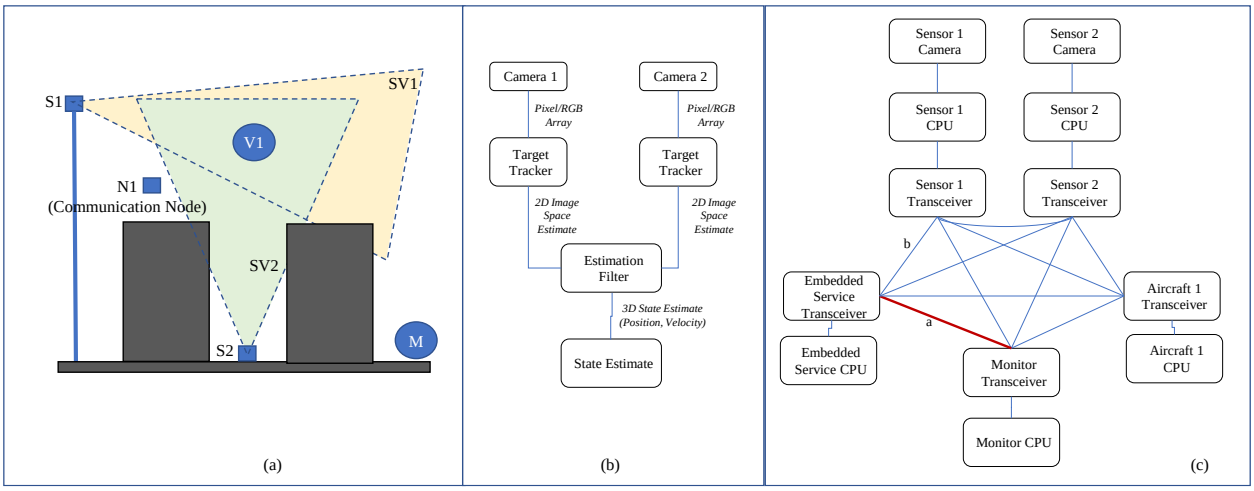

Figure 6. Sensor Network Database Considerations. *Consider a desired estimation goal (a) and a potential estimation topology (b), where elements from the estimation topology must be mapped to the physical topology (c).*

C. Nominal Mode Operation Phase

The nominal mode operation phase for the framework in shown in Figure 7. Nominal operations refer to the expected standard operating function of the framework, where sensor data is collected, processed, and estimates produced to the intended information consumer as information flow is routed through structures across the distributed network. This mode assumes that estimation targets have been established, information flow structures determined, processing elements assigned, and all asynchronous elements of the network have already started producing data. During nominal operations, asynchronous processes will perform updates of target state estimates, evaluate network elements for health and reliability (e.g., cross-validation of sensors), update target type-classifications, and monitor the state of the global distributed sensing network to determine if any topological changes are required.

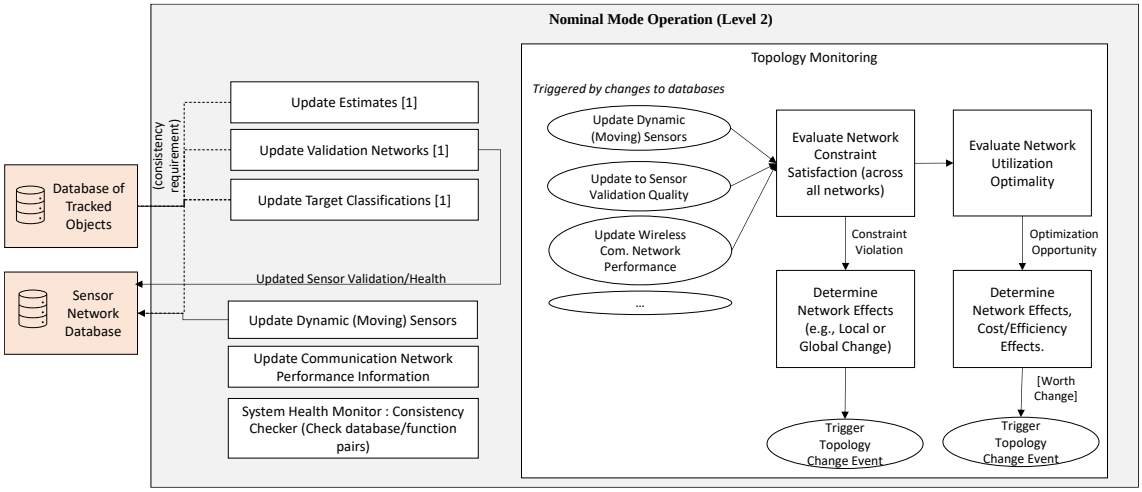

Figure 7. Level 2 Functional Requirements for Nominal Operations Phase

Target estimation updates are regularly produced during nominal operation over the established estimation structures and network structures. In the motivating scenario, the elements in Figure 6 (b) have been assigned and are asynchronously operating. Each camera sensor node captures an image at a specified time-interval, processing the image to identify the new image-space location of the pre-specified targets, and the updates are sent over the network to the relay node. Depending on assignment of the estimation filter, the filter may be hosted on the relay node CPU or the monitor node CPU. The image space estimates will be transmitted through the network to the appropriate hardware processor where the estimation filter is hosted. The estimation filter produces the state estimate, and the estimate is

transmitted to the consumer. The sensor network database and the tracked object database function similarly to routing tables in a distributed sensor network, as these databases store the information needed to understand the routing topology.

The initial framework is based on a pre-emptive real-time event-driven process model assumption. This assumes that reception of information at any point in the pipeline immediately triggers reading the data, processing the data, then communication of the result to the next node in the pipeline. Time information is sent along with each packet for evaluation of communication performance; for instance, measured latency is compared to the anticipated latency computed in the sensor network database. Once the topology has been specified and processing elements are migrated, each processing node independently execute asynchronously in this event-driven manner until some event occurs that modifies the topology or otherwise interrupts nominal mode execution.

Additionally, during nominal operations, the current framework topologies and information structures established across the network will be continually monitored. Monitoring functions are executed as update packets are received and after specified time intervals elapse. Dynamic (moving) elements in the network, such as participating aircraft - hosting elements such as transceivers, sensors, actuators, and processors - will be updated and written to the sensor network database. Updating the dynamic structural elements will affect properties of the physical network, such as the available bandwidth between a moving aircraft and a ground relay element, which would update the costs assigned to a specified edge of the network as shown in Figure 6 (c). In the initial implementation of this framework, real-time network quality information comes from two sources; (1) if supported by the hardware and underlying protocols, network quality information is queried directly from the hardware transceivers; (2) for data not available from the hardware, communication estimation models utilize to estimate network conditions. Simple communication models relating quality estimates to distance are currently being evaluated for this initial phase implementation.

D. Adaptive Topology Determination and Switching/Handoff Phases

A defining characteristic of wireless airborne communication networks is the expectation of instability and time-varying performance in the network communication topology during operations. As aircraft move through the environment, wireless communication to other vehicle and to fixed ground locations will appear and disappear with regular frequency. This highly dynamic topology requires specialized algorithms for coordinating communication, such as dynamic routing tables in the underlying network protocols. In this framework, as describe in section II-C, services in the nominal operation mode will continually check for any condition that will require a topology change. As modeled at the level-1 model shown in Figure 3, when such events occur, the system will transition to an adaptive topology determination mode.

For the initial framework design and evaluation, the adaptive topology determination phase model is simply implemented using the same model as the initial topology determination phase as presented in section II-B. The current implementation models are shown in Figure 8. Any event requiring a change in an estimation target's topology will result in removing the topology from the database structures, terminating and purging of all execution elements related to the topology in the framework process and remote hosts, then re-initialization of the topology as if this were a completely new object. A copy of the old topology elements and a reference to the new topology database elements are passed to the next phase, "Switching/Hand-off Between Topologies".

During switching/handoff, the framework is responsible to establishing a consistent continues measurement between states. The initial implementation of this framework simply copies the final state vector estimate from the previous estimation filter to the new estimation structure's current state vector. This provides second-order continuity in the estimate of position and first-order for velocity. This approach is sufficient for initial evaluations and is specific to the current estimation network implementation. Future research is planned to investigate requirements and methods for generalized handling of topological changes.

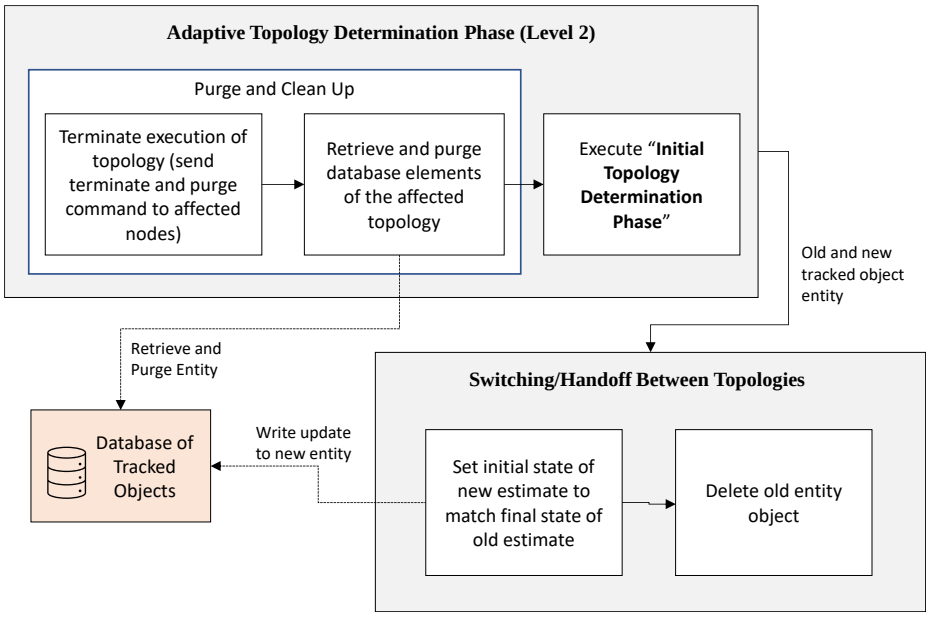

Figure 8. Adaptive Topology Determination and Switching/Handoff Models (Level 2)

III. Status and Future Work

This paper has presented details on a preliminary design of an airborne distributed sensing framework towards enabling future airspace operations. This paper presented a motivating scenario and use cases to support precision navigation and surveillance functions in urban airspaces to address current day gaps in capabilities to support these operations. This framework is currently being developed and tested as part of the Distributed Sensing project at NASA Ames Research Center [4]. Simulation evaluation will be conducted in the simulation architecture and vertiport designs described in (Kannan, 2023) and (Kawamura, 2023). The estimation architecture for combining vision and radar ground sensors was introduced in (Dolph 2022) and is being used as a baseline estimation topology for the frameworks initial design. Advancements to the fixed-structure of this architecture is presented in (Lombaerts, 2023), which presents a scalable and adaptive estimation architecture that can incorporate inputs from a set of sensors which may have temporary observations of the target. Additionally, sensor models described (Stepanyan 2021) and network evaluation and data migration strategies described in (Stepanyan, 2022) and (Stepanyan 2023) are being incorporated into the initial development of this framework. In addition to simulation testing, hardware testbeds are being developed to perform evaluation subscale sUAS flight tests, as described in (Brown 2023) and (Kannan, 2023).

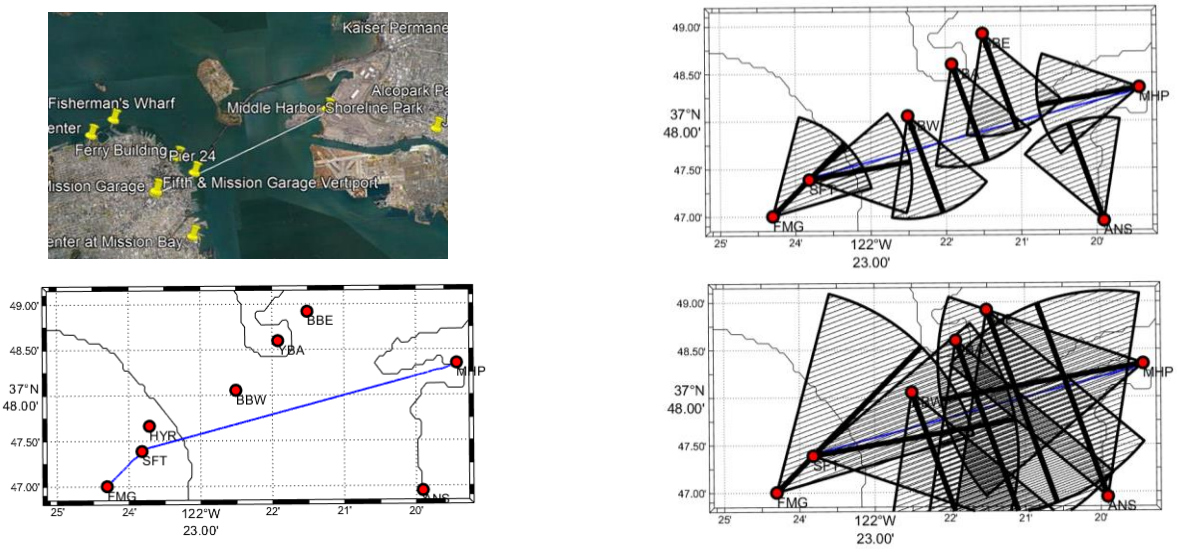

Figure 9. Adaptive Scalable Estimation Architecture in the AAM Simulator (Lombaerts, 2023)

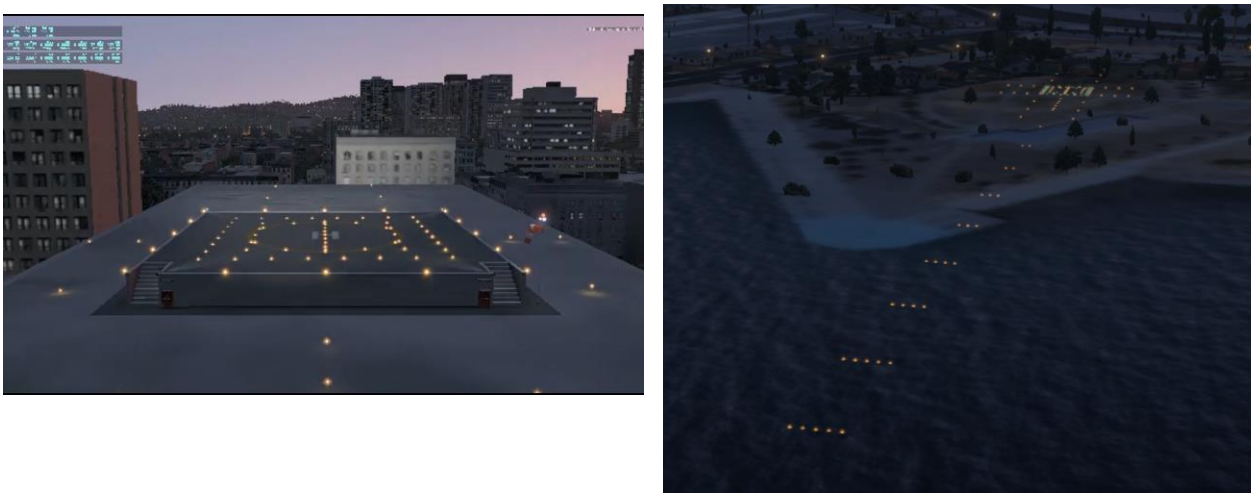

Figure 10. Vertiport Simulation Mockups (Kannan 2023)(Kawamura 2023)

Figure 11. Simulation View (Kannan 2023)

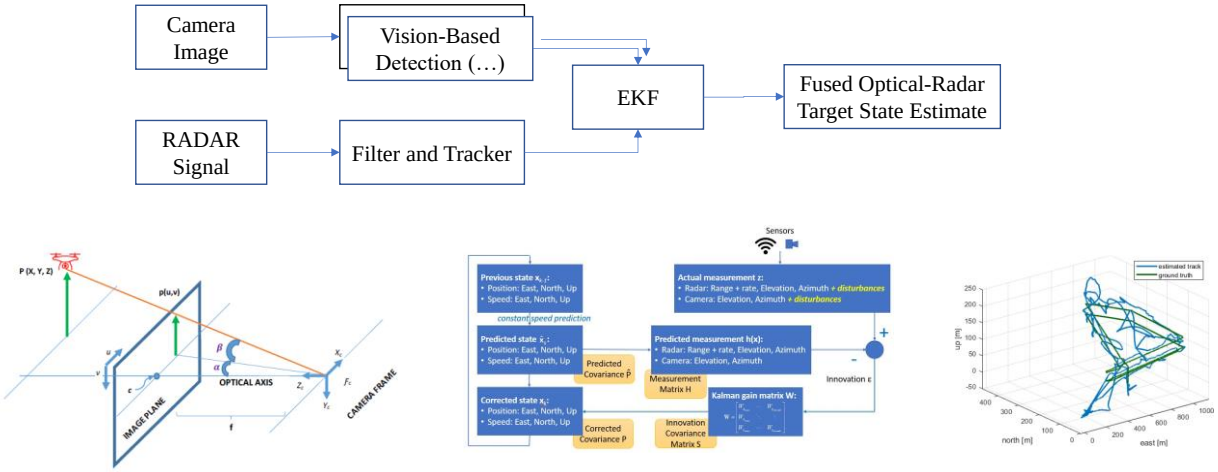

Figure 12. Fused optical-radar aircraft detection and tracking system (Dolph 2023)

The results from the initial simulation and flight test evaluations will be used to guide future development priorities. Future improvements to this framework may include: (1) extension to real-time decentralized control application; (2) utilize a combination of measured data with more sophisticated communication models; (3) incorporate predictive elements to this framework and have this information available to consumers; (4) decentralized implementation of the framework; (5) extending the support set of use cases to include other sensing goals; (6) timing and time-synchronization services, (7) topology adaption and switching/handoff considerations.

The initial implementation is focusing on sensing and estimation only. However, the framework formulations were developed in anticipation of extension to applications requiring decentralized real-time control. The use case considered will be expanded to applications such as formation control, automated sequencing, and automated in-flight separation. Applications of these use cases include decentralized vertiport-area operations, where intercommunicating and interoperating agents coordinate terminal-area operations between themselves without the need for a centralized airspace control element.

Accurate network quality estimates produced from the sensor network database are critical for proper operations in the current design of this system. The initial design is limited by utilizing simple models when the underlying network hardware does not support this data.

Another limitation of this initial framework design is reliance on only using the current state in determining topological structures. This is expected to produce a reactionary system that will not anticipate future network conditions and may produce short term instabilities. Instability and reactionary issues will be evaluated during the initial hardware experimentation phase. Utilizing predictive modeling and future time estimates in the analysis and topology determination algorithms is expected to provide better overall stability and performance.

The presented framework was designed for decentralized implementation across various elements in the system. In the initial development phase, the framework is being implemented on a centralized processing node in the system and will remain persistent for the length of the experiments. Future extension of this framework may look at issues with decentralized operations, where elements of this framework may be duplicated, non-persistent, asynchronously operating, and distributed across computational elements in the larger system.

The current implementation is focused on supporting surveillance and navigation use cases. Future extension to this framework will extend these use cases to additional sensing goals supporting additional applications and use cases.

Time synchronization services are not anticipated to be needed from this framework, and as such, time synchronization (the time component of the overall PNT problem) is currently outside the scope of this framework. For instance, processer clocks distributed through the system can utilize existing sources for periodic time synchronization - internet-based time synchronization services or GPS receivers during periods when GPS is available – and rely on internal clocks keep a sufficiently accurate measurement of time between these periodic synchronization events. However, if timing services becomes an established need, this framework could potentially be extended using time synchronization methods established for distributed sensor networks.

For this initial implementation and evaluation, topology changing events are simplistically implemented using a "terminate and restart" method. This method is sufficient for initial evaluations but is not sufficient for a final framework design. Termination is time-consuming and requires synchronous coordination between processors, resulting in a large time-gap of reported data. Most topological changes are expected to be local and can be handled in a more elegant strategies that can provide continuous estimates to the end user. These phases may be further developed upon completion and evaluation of results from initial simulation and flight test experiments.

IV. References

[1] "Advanced Air Mobility Overview," National Aeronautics and Space Administration, November 2022. [Online]. Available: https://www.nasa.gov/aam. [Accessed January 2023].

[2] G. Price, D. Helton, K. Jenkins, M. Kvicala, S. Parker and R. Wolfe, "Urban Air Mobility Operational Concept (OpsCon) Passenger-Carrying Operations," 2020.

[3] B. P. Hill, D. DeCarme, M. Metcalfe, C. Griffin, S. Wiggins, C. Metts, B. Bastedo, M. D. Patterson and N. L. Mendonca, "UAM Vision Concept of Operations (ConOps) UAM Maturity Level (UML) 4," 2022.

[4] C. A. Ippolito, K. Hashemi, E. Kawamura, G. Gorospe, W. Holforty, K. Kannan, V. Stepanyan, T. Lombaerts, N. Brown, A. J. Jaffe and C. Dolph, "Concepts for Distributed Sensing in Advanced Urban Air Mobility," in *AIAA SciTech 2023 Forum and Exposition*, National Harbor, MD, 2023.

V. Appendix – Initial Topological Modeling Formulation

The following section describes the formulation utilized in modeling the problem domain for this framework.

A. System Component Graph Formulation in the Estimation Topological Layer

For this formulation, we utilize a directed edge-weighted B-hypergraph as illustrated in Figure 5. A directed hypergraph can be considered a generalization of a directed graph where edges connect two sets of vertices, a head set and a tail set. B-hypergraphs, or B-graphs, are a restricted form of general hypergraph where the head set of vertices is restricted to a size of 1. Hypergraphs are chosen for this work to represent general relationships that relate

a variable to a set of other variables that this variable can be mathematically derived or algorithmically computed from, and are a basic component used in the design of this framework.

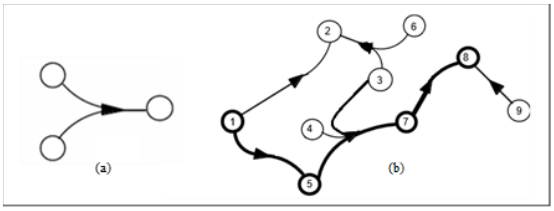

Figure 13. Directed edge-weighted B-arc (a), B-Graph and B-Path (b)

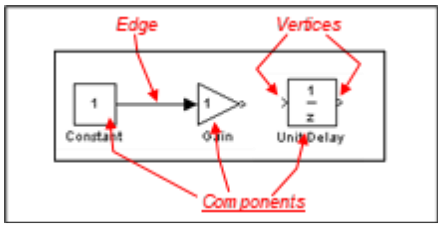

Figure 14. Component Graph and Atomic Definitions

The following definitions are utilized in this framework. Let a vertex $v \in \mathcal{V}$ is an indivisible unit in the component graph. Let $\mathcal{V}$ the set of all vertices. Vertices in the graph represent a data attribute of a component. Let an edge $e \in \mathcal{E}$ be a directed b-hyperarc, defined as an ordered pair $e \coloneqq \langle S, t \rangle$, where $S \subseteq \mathcal{V}$ is a set of vertices referred to as the starting or tail vertices, and where $t \in \mathcal{V}$ is a singleton vertex referred to as the terminal or head vertex. Let $\mathcal{E}$ be the set of all edges. For notational convenience we define the following operators. Given a set A, let the operator $\mathcal{P}(A)$ return the set of all possible subsets of A, often defined as the power set of A. Given an edge $e \in \mathcal{E}$ where $e = \langle S, t \rangle$, define the operators $S: \mathcal{E} \to \mathcal{P}(\mathcal{V})$ and $t: \mathcal{E} \to \mathcal{V}$ to return the starting vertex set $S(e) \coloneqq S$ and the terminal vertex $t(e) \coloneqq t$, respectively.

Let a system component graph $G \in \mathcal{G}$ be defined as a tuple $G \equiv (V, E, C)$, where $V \subseteq \mathcal{V}$ is a set of vertices $V = (v_1, \dots, v_{|V|})$ in graph G, where $E \in \mathcal{E}$ is a set of edges $E = (e_1, \dots, e_{|E|})$ in the H, and where $C \in \mathcal{G}$ is the set of components in the graph, to be defined recursively below. Let $\mathcal{G}$ be the recursive space of all possible graphs of the form $\mathcal{G} = \mathcal{V} \times \mathcal{E} \times \mathcal{G}$.

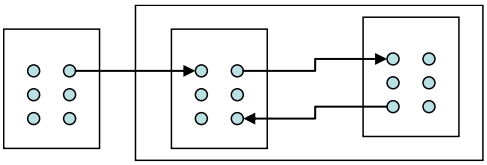

Figure 15. System Component Graph Contraction and Expansion to Model Hierarchical Level of Detail

For notational convenience, given a system graph $G = (V, E, C)$, let the operator $V: \mathcal{G} \to \mathcal{V}$ be defined as $V(G) = V$ to return the set of vertices in H. Let $E: \mathcal{G} \to \mathcal{E}$ be defined as $E(G) = E$ to return the set of edges in G. Let $C: \mathcal{G} \to \mathcal{C}$ be defined as $C(G) = C$ to return the set of components in G.

Given two graphs $H, G \in \mathcal{G}$, the graph H is a subgraph of a graph G, denoted as $H \subseteq G$, if an only if $V(H) \subseteq V(G)$, $E(H) \subseteq E(G)$, and $C(H) \subseteq C(G)$. The graph H is a proper subgraph of G, denoted as $H \subset G$, if and only if $H \subseteq G$ and either $V(H) \neq V(G)$, $E(H) \neq E(G)$, or $C(H) \neq C(G)$.

Let a component $K \in C(G)$ in graph G be defined with the following properties: (1) $K \subseteq G$ (Subgraph); (2) If $e \in E(K)$, then $S(e) \subseteq V(K)$ and $t(e) \in V(K)$ (Edge Containment); (3) If L∈C(K) then L⊆K(Subcomponent Containment); (4) K∉C(K) (Monotonicity).

For a component K, an element of $C(K)$ is a referred to as a *subcomponent* of K. The definition of a component in (0) defines what graphs are permissible as a component. For a graph K to be a valid component of graph G, K must

be a subgraph of G (0). If a component contains an edge, it must start and end in vertices contained within the vertex set of that component (0). All subcomponents are proper subgraphs of that component 0. A component cannot contain itself as a subcomponent (0). Note that component monotonicity facilitates induction and recursion.

The definition of a component as a subgraph given in (0) allows system components graphs to be collapsed or expanded to any arbitrary levels of detail when analyzing and manipulating components in the system, as illustrated in Figure 15.

A component K subsumes a component D if and only if $D \subset K$ and either (1) $D \in C(K)$, or (2) there exists a subcomponent $K' \in C(K)$ where K' subsumes D. Note that a component can therefore never subsume itself. Let the *power set* of K, denoted as $\mathcal{P}(K)$, be the set of all possible components subsumed by a component K.

The definition in (0) has several implications. A component can never subsume itself. The set of all possible subsumed components is finite. Further, traversing subsumed subcomponents will always result in a component graph that is monotonically decreasing in size - in terms of number of vertices and edges.

Let a trace path P of length q in a system graph $G = (V, E, C)$ to be an ordered sequence of vertices and edges given by $P = (v_1, e_1, v_2, e_2, \dots, v_q, e_q, v_{q+1})$, subject to the following properties: (i) $v_j \in T(e_j) \ \forall\, j = (1..q)$, and (ii) $v_j = H(e_{j-1}) \ \forall\, j = (2..q+1)$. These properties state that any vertex v_j in the path must be the head vertex of the previous edge in the sequence, and must also be in the tail set of the following edge. We refer to v_1 as the source vertex of path P, and v_{q+1} as the terminal vertex of path P.

For notational convenience, given a trace path $P = (v_1, e_1, v_2, e_2, \dots, v_q, e_q, v_{q+1})$, define the operators $src(P) \coloneqq v_1$ and $ter(P) \coloneqq v_{q+1}$ to return the source and terminal vertices, respectively. Define the operators $V(P) \coloneqq (v_1, \dots, v_{q+1})$ and $E(P) \coloneqq (e_1, \dots, e_q)$ to return the set of all vertices and edges, respectively.

A trace path P is simple if all hyperarcs in P are distinct. A trace path P is elementary if all vertices in P are distinct.

A trace path P in a system graph $H \in \mathcal{H}$ is a cycle if $src(P) = ter(P)$. A cyclic path contains one or more sub-paths that are a cycle. A path that contains no cycles is acyclic.

A vertex $t \in \mathcal{V}$ is connected to vertex $s \in \mathcal{V}$ in a system graph $H \in \mathcal{H}$ if and only if there exists a path P_{st} from s to t, where $s = src(P)$ and $t = ter(P)$.

A trace path P in a system graph H induces a minimal subgraph $H_P = (V_P, E_P, W_P)$ such that (i) $E_P \subseteq \mathcal{E}$, (ii) $s, t \in V_p$, where $V_p = \bigcup_{e_i \in E_P} e_i \subseteq \mathcal{V}$, and (iii). $x \in \mathcal{V} \Rightarrow x$ is connected to s in H_P by means of a cycle-free simple path.

Operations and analysis on the system graph are facilitated through assignment of an edge weighting function to edges in the graph.

Let $w \in \mathcal{W}$ be an edge weighting function $w: \mathcal{E} \to \mathbb{R}$ that maps a hyperarc to a real scalar value in the set $\mathcal{W}$ of all such functions.

We define a notion of a configuration space that defines the set of all possible configurations. We defined the configuration space graph below to contain the set of all possible directed hyperarcs between vertices in G.

Let the configuration space $\tilde{C}(G)$ of a graph $G \in \mathcal{G}$ be defined as the following: (1) $V(\tilde{C})$ is the set of all vertices in G, C(G), and all subcomponents of C(G); (2) $E(\tilde{C})$ is defined as the set of all edges $e \in \mathcal{E}$, $e = \langle S, t \rangle$, for all $S \subseteq V(G)$ and $t \in V(G)$; and (3) $C(\tilde{C}) = C(G)$.

A components C contains a set of vertices V if V$\subseteq V$(C).

Let the parent component of a vertex v be defined as follows. For every vertex v$\in$V in $\tilde{C}$, there exists a unique c$\in C(\tilde{C})$ such that v$\in$c, E(c)=$\emptyset$, and C(c)=$\emptyset$. This unique component c is said to be the parent of v, and is written *parent*(v) = c.

A vertex has one and only one parent component. As a result of this rule, the set of parent components in C($\tilde{C}$) are set of components with no edges or subcomponents, and this set is unique and disjoint. Further, there are components with no edges or subcomponents that are inadmissible in $\tilde{C}$ because of violation of the parent component definition.

The concept of connectiveness from traditional graph theory is extended here for components in a graph. Note that these definitions do not consider edge direction in the definitions for component-connectivity.

Given a graph G, two components A,B$\in C$(G) are neighbors in G if there exists an edge e$\in E$(G) such that (s(e)$\in V$(A) $\wedge$ t(e)$\in V$(B)) $\vee$ (s(e)$\in V$(B) $\wedge$ t(e)$\in V$(A)).

A component-path P_c is a defined as a set of two or more components where either:

There are two components in P_c, and these components are neighbors, or

There exist two components A,B$\in P_c$ which are neighbors, and (P_c-A) is a component-path.

A component graph G is component-connected if there exists at least one component-path between every pair of components in G.

Given these definition we can define a particular configuration in configuration space. A configuration represents a set of components with edges and vertices.

Let a configuration graph G in $\mathcal{C}(G)$ be defined as a graph G=(V,E,C) that represent a specific configuration implementation, subject to (1) G∈C($\mathcal{C}$)(G is a Component); (2) For all v∈V(G) there exists one and only one K∈C(G) such that v∈K (Disjoint Subcomponents); (3) For all e∈E(G), let *init*(e)∈V(L) and *ter*(e)∈V(M) are contained in different components. (Pruned Edges); (4) G is component-connected (Component-Connected).

For a configuration graph, all components in the representation are disjoint; in other words, vertices are contained by only one component in G, which is not the parent of the vertex if *parent*(v∈G)∉C(G).

Define an output variable as follows. Let *isoutput*:V→B be a mapping such that *isouput*(v)=b where v∈V and b∈$\mathcal{B}$ if and only if there exists e∈E($\mathcal{C}$) such that *init*(e)=v.

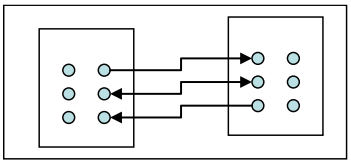

Figure 16. Input/Output Variables

Component Contraction: There exists a unique mapping *CV*:G→G_v where G_v=(V,E) s.t. *CV*(g)=g_v, defined as follows: For every g∈$\mathcal{C}$, ∃g_v=G_v such that every c∈C(G) has a corresponding v∈V(G_v), where v=*CV*(e) defines a mapping *CV*:E→V, and for every edge e∈E(G), there exists a corresponding e_v∈E(G_v) between the associated *CV*(*ancestor*(*init*(e))) and *CV*(*ancestor*(*ter*(e))).

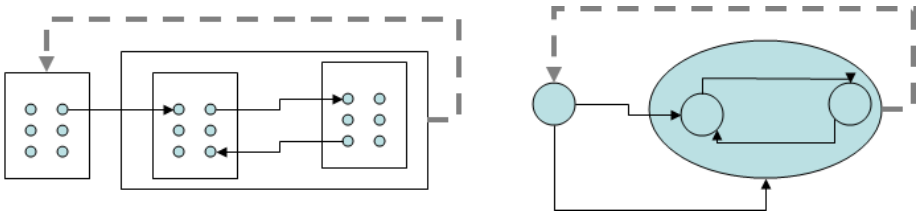

Figure 17. Isomorphic Projection: Controller Space Graph to Component-Vertex Space.

The $\mathcal{C}_v$ component space represents the isomorphic projection of the controller space through component contraction. Every component is an element of one system boundary partition. The system boundary partition space ($\mathcal{C}_P$) is defined as a projected space of a component across partitions.

(Flow Definition) Between two sets of vertices let *f*(X,Y):={ e∈F | x∈X, y∈Y; x≠y; X,Y⊆V(G)}. A mapping *f*:E→R is a flow in N if it satisfies there conditions: (1) f({<x,y>}) = -f({<y,x>}); (2) f(x,V) = 0 for all v∈V\{s,t} (3) f(e)≤c(e) for all e∈E.

Let a mapping *f*:E→R be a Kirchhoff network mapping if the sum of all flows into a component is equivalent to the sum of all flows coming out of a component. In every network, the maximum total value of a flow equals the minimum capacity of a cut. Note that Kirchoff's law does not apply for general costs (e.g., bandwidth constraints), so network theory is not quite appropriate in the general case. Latency constraints in component space, however, can be addressed using a Kirchhoff mapping.

B. Transformation Between State-Space Dynamics to System Component Graphs

Consider a large-scale linear time-invariant dynamic system $S \in \mathcal{S}$ in the set $\mathcal{S}$ of all possible systems given by the following.

$$S \;:\; \begin{aligned} x(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \qquad \text{(B-1)}$$

Here, $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, and $y(t) \in \mathbb{R}^l$ are the are the state, input, and output. The system matrices are constants given by $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{l\times n}$. Let x(t), u(t), and y(t) be vectors of dimension n, m, and l, respectively.

Without loss of generality, consider $S \in \mathcal{S}$ be composed of n interconnected overlapping sub-systems, where $\mathrm{N} = \{1,2,\dots,n\}$. This system is defined in equation (B-2). Interconnections between each system are given by the off-diagonal elements Aij, Bij, and Cij where $i \neq j$.

$$S \;:\; \dot{x}_i(t) = \sum_{j=1}^{N}(A_{ij}x_j + B_{ij}u_j)$$
$$y_i(t) = \sum_{j=1}^{N}(C_{ij}x_j) \quad for\ all\ i \in \mathrm{N} \tag{B-2}$$

Graph operations are defined to analyze, modify, and partition the system hypergraph into isolated subgraph components. The goal of these operations is to expose structural features of the model that will facilitate decentralization of the control scheme. The isolated subgraphs are individually transformed back into the time-domain representation for control synthesis to occur, then transformed back through the process to arrive at the final decentralized control system.

Given a dynamic system $S \in \mathcal{S}$ as defined in (B-2), consider a transformation T to be a transformation $T: \mathcal{S} \rightarrow \mathcal{H}$ that transforms $S \in \mathcal{S}$ into the system graph $H \in \mathcal{H}$ where the vertices $V(H) = (x_1,..,x_n,u_1,\dots,u_m)$ are defined as the set of all state variables and input variables, and the edges E(H) of the graph are induced by a weighted adjacency matrix A_w defined as

$$A_w(S,\delta) = \min\left(\begin{bmatrix} A & B & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \delta\right)$$

The transformations $T_{SH}: \mathcal{S} \rightarrow \mathcal{H}$ and $T_{HS}: \mathcal{H} \rightarrow \mathcal{S}$ define an isomorphic transformation between $\mathcal{S}$ to $\mathcal{H}$. This can be shown inductively. This property holds for a graph with a single element. Assuming this property holds for graph with some given number of elements, adding a new element to the graph will still maintain this property.

Isomorphism is an enabling property of the transformation T_{SH} and T_{HS} between $\mathcal{S}$ and $\mathcal{H}$. The algorithms presented utilize this property to freely transform between the state space representation and the system graph representation. The system graph representation can be manipulated, the graph can be broken into smaller pieces, smaller subgraphs can be transformed back and forth between the system graph and state space representations, and subgraphs can be reassembled and transformed back into the state space representation through this property.

C. Three-Mass System Example

This section will describe the framework applied to a simple three-mass problem. Consider the following example involving a system of three masses (m_1, m_2, m_3) interconnected by springs and dampers. Two masses have control forcing inputs are strongly coupled to a third mass without direct input, and weakly coupled to each other, such that $k_1 \gg k_2$ and $b_1 \gg b_2$.

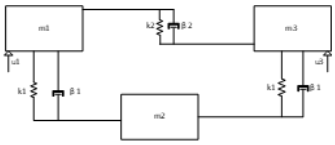

Figure 18. Coupled Three-Mass System

The system dynamics S_1 are given by the following model.

$$S_1 : \dot{x} = Ax + Bu \tag{C-3}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -(k_1+k_2) & -(b_1+b_2) & k_1 & b_1 & k_2 & b_2 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ k_1 & b_1 & -2k_1 & -2b_1 & k_1 & b_1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ k_2 & b_2 & k_1 & b_1 & -(k_1+k_2) & -(b_1+b_2) \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ f & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & f \end{bmatrix}$$

Consider a typical optimal control structure for this system that minimizes the following.

$$\min_u J(x_0, u) = \int_0^\infty [x^T(t)Q^*x(t) + u^T(t)R^*u(t)]dt$$
$$s.t.\ \mathbf{S}: \dot{x}(t) = Ax(t) + Bu(t) \qquad \text{(C-4)}$$

Transforming system S_1 into a system graph form yields the following graph in Figure 19.

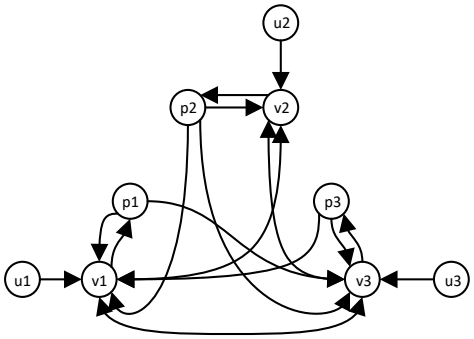

Figure 19. System Graph Representation of the Three Mass System S_1.

Applying the clustering metric to the graph results in the weighted graph in Figure 20.

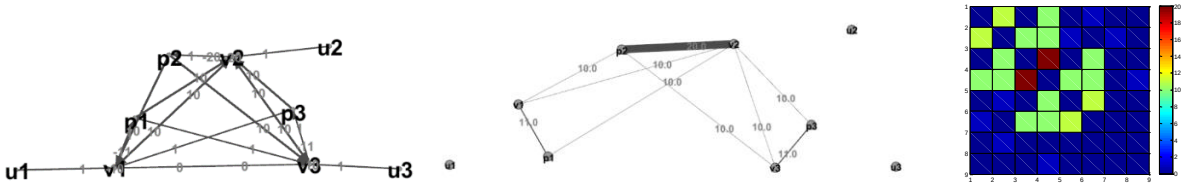

Figure 20. System Graph Representation of the Three Mass System S_1. *Original system graph (left), cluster function matrix (middle), cluster function graph (right).*

Note the structure of the contracted cluster graph shows strong linkage between P1-V1 and P2-V2, and also between P2-V2 and P3-V3. There are weak linkages between P1-V1 and P3-V3. The clustered representation identifies candidate decomposition of overlapping expansion and contraction. Applying overlapping decentralization yields the expanded system in Figure 21.

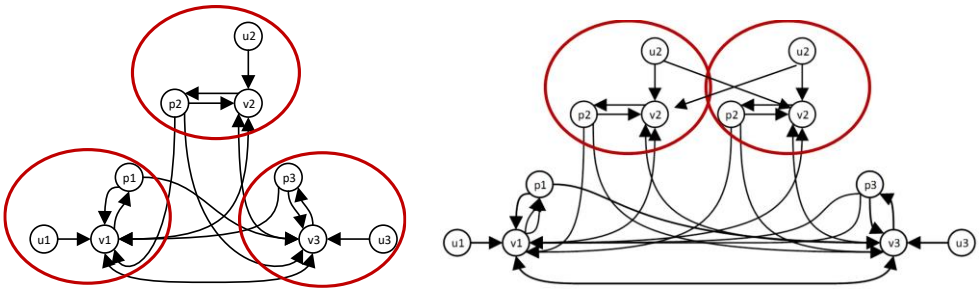

Figure 21. Overlapped system identification (left) and expanded system after expansion (right).

Given the isomorphic properties of the graph space transform, the operations can analogously be performed utilizing graph operations. Performing the minimum cut isolates the graph into two distinct sub-graphs, shown in Figure 22.

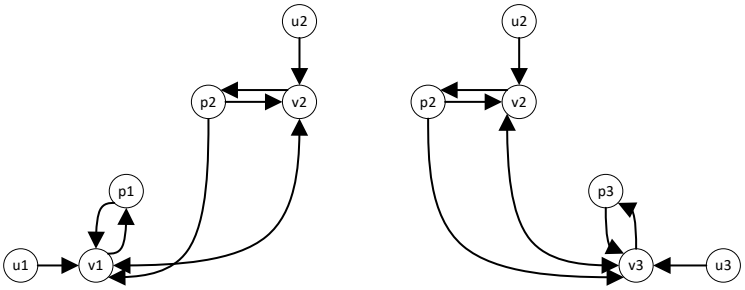

Figure 22. Resulting isolated systems.