AN EMERGENT ARCHITECTURE FOR SCALING DECENTRALIZED COMMUNICATION SYSTEMS

John Barbosa Vicente II

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Sciences

Columbia University

2011

ABSTRACT

An Emergent Architecture for Scaling Decentralized Communication Systems

John Barbosa Vicente II

With recent technological advancements now accelerating the mobile and wireless Internet solution space, a ubiquitous computing Internet is well within the research and industrial community's design reach – a decentralized system design, which is not solely driven by static physical models and sound engineering principals, but more dynamically, perhaps sub-optimally at initial deployment and socially-influenced in its evolution. To complement today's Internet system, this thesis proposes a Decentralized Communication System (DCS) architecture with the following characteristics:

- flat physical topologies with numerous compute oriented and communication intensive nodes in the network with many of these nodes operating in multiple functional roles;
- self-organizing virtual structures formed through alternative mobility scenarios and capable of serving ad hoc networking formations;
- emergent operations and control with limited dependency on centralized control and management administration.

Today, decentralized systems are not commercially scalable or viable for broad adoption in the same way we have to come to rely on the Internet or telephony systems. The premise in this thesis is that DCS can reach high levels of resilience, usefulness, scale that the industry has come to experience with traditional centralized systems by

exploiting the following properties: *(i.)* network density and topological diversity; *(ii.)* self-organization and emergent attributes; *(iii.)* cooperative and dynamic infrastructure; and *(iv.)* node role diversity. This thesis delivers key contributions towards advancing the current state of the art in decentralized systems. First, we present the vision and a conceptual framework for DCS. Second, the thesis demonstrates that such a framework and concept architecture is feasible by prototyping a DCS platform that exhibits the above properties or minimally, demonstrates that these properties are feasible through prototyped network services. Third, this work expands on an alternative approach to network clustering using hierarchical virtual clusters (HVC) to facilitate self-organizing network structures. With increasing network complexity, decentralized systems can generally lead to unreliable and irregular service quality, especially given unpredictable node mobility and traffic dynamics. The HVC framework is an architectural strategy to address organizational disorder associated with traditional decentralized systems. The proposed HVC architecture along with the associated promotional methodology organizes distributed control and management services by leveraging alternative organizational models (e.g., peer-to-peer (P2P), centralized or tiered) in hierarchical and virtual fashion. Through simulation and analytical modeling, we demonstrate HVC efficiencies in DCS structural scalability and resilience by comparing static and dynamic HVC node configurations against traditional physical configurations based on P2P, centralized or tiered structures. Next, an emergent management architecture for DCS [20], [25], exploiting HVC for self-organization, introduces emergence as an operational approach to scaling DCS services for state management and policy control. In this thesis, emergence scales in hierarchical fashion using virtual clustering to create multiple tiers of local and global separation for aggregation, distribution and network control. Emergence [18], [33], [34] is an architectural objective, which HVC introduces into the proposed self-management design for scaling and stability purposes. Since HVC expands the clustering model hierarchically and virtually, a clusterhead (CH) node, positioned as a proxy for a

specific cluster or grouped DCS nodes, can also operate in a micro-capacity as a peer member of an organized cluster in a higher tier. As the HVC promotional process continues through the hierarchy, each tier of the hierarchy exhibits emergent behavior.

With HVC as the self-organizing structural framework, a multi-tiered, emergent architecture enables the decentralized management strategy to improve scaling objectives that traditionally challenge decentralized systems. The HVC organizational concept and the emergence properties align with [179] and the view of the human brain's neocortex layering structure of sensory storage, prediction and intelligence. It is the position in this thesis, that for DCS to scale and maintain broad stability, network control and management must strive towards an emergent or natural approach. While today's models for network control and management have proven [12], [17], [151] to lack scalability and responsiveness based on pure centralized models, it is unlikely that singular organizational models can withstand the operational complexities associated with DCS.

In this work, we integrate emergence and learning-based methods in a cooperative computing manner towards realizing DCS self-management. However, unlike many existing work in these areas [196],[106],[111] which break down with increased network complexity and dynamics, the proposed HVC framework is utilized to offset these issues through effective separation, aggregation and asynchronous processing of both distributed state and policy. Using modeling techniques, we demonstrate that such architecture is feasible and can improve the operational robustness of DCS. The modeling emphasis focuses on demonstrating the operational advantages of an HVC-based organizational strategy for emergent management services (i.e., reachability, availability or performance). By integrating the two approaches, the DCS architecture forms a scalable system to address the challenges associated with traditional decentralized systems. The hypothesis is that the emergent management system architecture will improve the operational scaling properties of DCS-based applications and services. Additionally, we demonstrate structural flexibility of HVC as an

underlying service infrastructure to build and deploy DCS applications and layered services. The modeling results demonstrate that an HVC-based emergent management and control system operationally outperforms traditional structural organizational models. In summary, this thesis brings together the above contributions towards delivering a scalable, decentralized system for Internet mobile computing and communications.

1 Table of Contents

2 List of Figures

3 List of Tables

Acknowledgments

I completed this dissertation while holding an engineering position at Intel Corporation and through the support of my loving family. To that commitment, my deepest appreciation goes to my wife Maryann for her patience and perseverance to our children, Hannah, Vanessa, John III and myself during the PhD years.

I am grateful to Professor Andrew T. Campbell for enabling, inspiring and preserving with me through the PhD. Andrew is a friend and a genuine person with high integrity and kindness. Under Andrew's mentorship, I developed as a researcher, improved as a professional and became a better person. He is and will remain one of the most influential people in my life.

Thank-you to Doug Busch, Jim Hobbs, Raj Yavatkar and Martin Curley at Intel for creating the opportunity that enabled me to pursue a life-long goal of completing an electrical engineering Ph.D.

An important acknowledgement goes to the IT research team at Intel for the collaboration and joint development of the OverMesh experimental platform.

Lastly, I dedicate this dissertation to my father (John Barbosa Vicente, Sr.) and mother (Maria Resende Vicente) for their many sacrifices in coming to the USA in 1967 from the tiny island of Fogo, Cape Verde and making it possible for me to reach this milestone.

John Barbosa Vicente

Chapter 1

1 INTRODUCTION

1.1 HISTORICAL OBSERVATIONS

The historical significance of decentralization through personal computers demonstrated a remarkable computing shift that accelerated innovation through broader participation and simultaneously grew the information technology (IT) industry. The future of communications is predicted [1], [9] to follow a similar path with significant opportunities for innovation and reduced barriers for market entry. Decentralization has the potential to bring major growth to the communications industry [4], [11]. Alternatively, networked users today still rely heavily on centralized networks and systems for their services. In a centralized system, users inevitably sit outside the service domain and access resources and services rendered to them by centrally operated service providers. This has broad implications to network costs, services and devices that end-users must conform, as well as to service providers who rightfully gain opportunity in such a centralized environment.

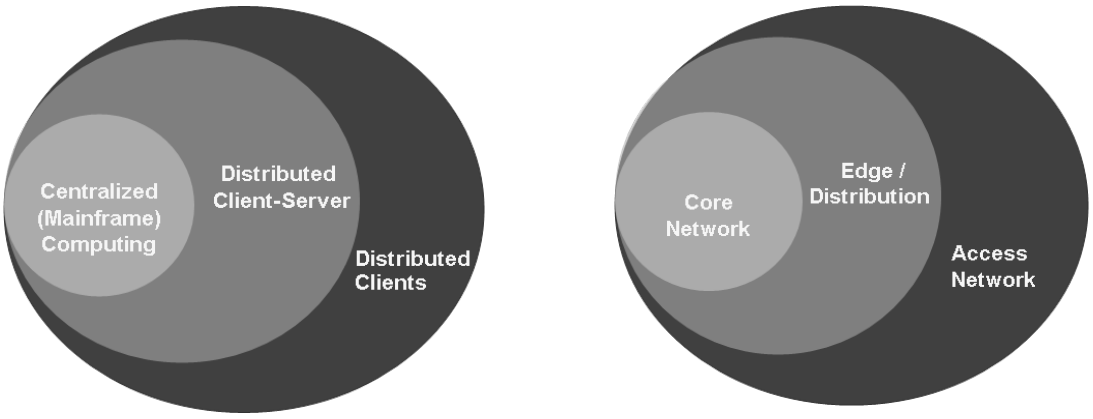
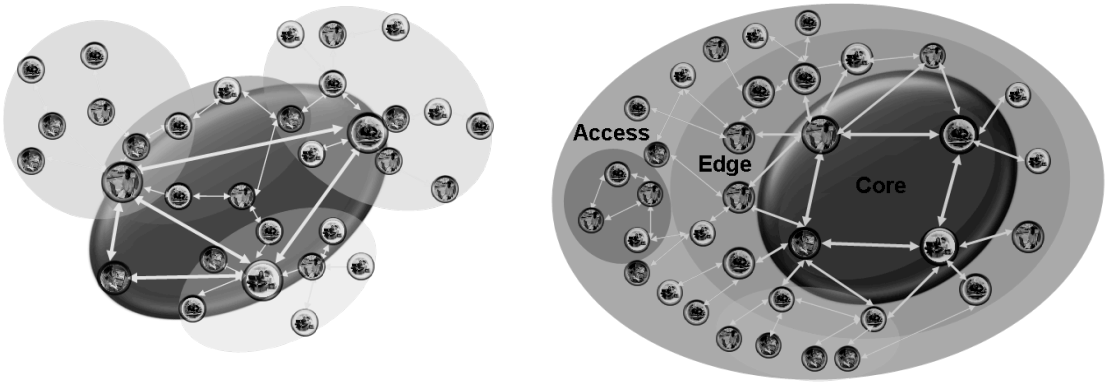


Figure 1: (a) Computing and (b) telecommunication structures

As shown in Figure 1(b), the telecommunications hierarchical structure, which governs positioning, and roles for the ecosystem value chain for data and voice communications

products and services, conceptually replicates the model of centralized computing, as depicted in Figure 1(a). While alternative providers control and manage capital-intensive services in the edge and access networks, the operating model for mobile computing has spawned diverse access networks, disjointed and costly to users. A very similar discussion can be made of cloud or Internet portal providers, which exploit centralization through 'walled garden' infrastructure for delivery of value-add services.

Decentralization is a key characteristic in future computing and communication architectures [4], [6], [11]. For communications, one can envision a flatter network system with a large number of nodes with similar form and function, but with varying resource and service capabilities. In this framework, end-users are associated with these nodes either as leaf or as internetworking nodes. We propose to *embed the user inside the network* rather than the borders of the network. Communities of virtual networks may form by contemplative design or through social cooperation. Physical (wireless) connectivity is instantiated opportunistically without ownership or hardened allocation. Figure 2 illustrates this concept with two alternative instantiations of decentralized networks formed through converged computing and communication nodes.



a). Sparse networks *b.) Structured networks*

Figure 2: Decentralized networks

What motivates this thesis proposal is the historical observation and reasoning on the Internet's architectural evolution and the emergence of mobile computing

advancements to accelerate a potential widespread adoption of decentralized systems. To achieve some level of parity with current systems and showing viability of the proposed DCS direction, a key challenge to overcome in this research is the feasibility of building a DCS platform, composed of virtual machine (VM) and peer-to-peer services over multi-radio, wireless networks. We integrate emergent and self-managing capabilities within the DCS framework, while dynamic node and topology structures are formed through virtualization and self-organizing means.

1.2 BUILDING ROBUST, DECENTRALIZED COMMUNICATION SYSTEMS (DCS)

Low-cost wireless communication systems are unwiring users and creating free association of communication services with compute-rich devices. This change is consistent with the ubiquitous computing [1] vision and the limitations of today's Internet mobile networks presents an opportunity to bridge towards such a transition. As peer-to-peer computing demonstrated, the system exhibits viral behavior [11], [13], [14] when the design incorporates the social population. In [6], Reed refers to Group Forming Networks (GFN) as a capability to increase the value of networks exponentially with the increased number of participants and groups. This insight naturally motivates the DCS research direction of bringing the user inside the network as a node in the network. Thus, we enable a decentralized communication infrastructure that matches the peering, social nature of the users and enables a larger number of physical and virtual group formations.

A significant challenge in moving towards decentralized communication systems is scalability. Mesh networks provide capabilities for transport (e.g. dynamic routing) and service diversity under conditions of load variations, failures and network resource constraints. In DCS, dynamic conditions bring variability in the topology of mobile infrastructure through selection of alternative radio communication options, signal fading conditions and rogue or uncooperative nodes. In order to scale the environment, scaling techniques such as using MIMO antennas, multiple channels, and multiple

radios can support larger-scale mesh deployments. In addition, smart radios enable a wireless device to sense its environment and alter its power, frequency or other parameters to reuse available spectrum, which can further improve spectrum efficiency and network scalability.

The creation and deployment of virtual machine networks [67] can help offset network-wide complexities through virtual structures to partition or separate communities of interest. However, there are clear distributed virtualization challenges such as service provisioning and discovery, virtual machine migration and management and general resource management. Further, decentralized communication systems allow any physical node to customize and provision node capabilities to create virtualized structures. In doing so, this minimizes physical node customization and physical infrastructure construction or reconstruction. A generalized[1] DCS node construct draws a more extreme virtualization requirement, as hardware must be software-defined to allow for rapid virtual constructs of the platform for virtual computing and communications.

1.2.1 STRUCTURAL SCALABILITY

The lack of predictable structure or routing is a known challenge for decentralized systems. Similar issues, which have challenged [97], [108], [130], [135] peer-to-peer, ad hoc, mesh or overlay networks are expected in the DCS environment. The continuous operational state and policy changes due to a high-degree of node mobility or node (self-serving) behaviors occur at an application-level, network services or infrastructure-level. Socially organized networks (e.g., flash crowds) or dense, population environments create network complexity challenges due to spontaneous increases in node counts, erratic traffic demand or interference profiles.

---

[1] DCS nodes are generalized in that role assignments in the topology can be formed dynamically allowing nodes to operate as computing or internetworking virtual devices.

To address these challenges, we exploit the virtual properties of DCS and operational superiority concepts to produce self-organizing structures. Utilizing hierarchical virtual clustering (HVC) techniques, DCS facilitates alternative styles of computing cooperation and connectivity structures. By observing and capturing operational properties of the system at different levels of the hierarchy, the system can self-organize to an optimal structure.

How does one efficiently manage and control DCS? Given limited and potentially fleeting computational capacity and dynamic topologies, static management and control of decentralized services are unlikely to lead to effective service reliance or network optimization. In this work, the use of virtual machine networks at a local and distributed level are evaluated towards addressing this challenge. Creating a virtual structure for shared information [12], [78], [86] or for state and policy synchronization between the higher level services and decentralized resources are a challenging objective for DCS. In Chapter 5, we demonstrate through simulation the effectiveness of share state and information exchange using HVC for achieving decentralized management and network control.

1.2.2 STABILITY & CONVERGENCE

Optimal or dynamic structure does not necessarily guarantee stability in DCS as several research contributions have shown [190], [192], [199]. Unlike traditional client-server systems, where the client, the server and the switches and routers operate in singular, physical and static roles with an obvious centralization strategy, DCS does not follow a fixed architectural deployment. Alternatively, DCS nodes can serve multiple, dynamic roles – source or sink consumer (e.g., client), internetworking (i.e., router or gateway) or source or sink provider (e.g., server). The associated structures and traffic models for DCS systems are decentralized and dynamic. The degree of static state or policy, configuration or structure is unknown at initial state or can frequently change after deployment. Therefore, the system - the users, applications and underlying network services cannot realistically rely on any fixed structure or

static operating environment. Further, given a lack of centralized authority and administration – the coordination and operational complexity is clearly beyond manual control and many of today's approaches to network and distributed systems management. Finally, since network devices serve both user and infrastructure requirements, there is a natural tension between cooperation for optimization and selfishness for serving end user needs.

Through hierarchical virtual clustering (HVC), we can create multiple tiers of local and global separation for state and policy aggregation, distribution and decision-making. As HVC expands the clustering model hierarchically and virtually, a lead cluster node or clusterhead can operate in multiple, extended roles (i.e., member and clusterhead) across the organized virtual node hierarchy. As the clustering promotion process is exhibited at each tier of the hierarchy, emergence [18], [33], [34] is exhibited, but extended beyond a local (micro behavior) and global (macro behavior) definition, which is common in natural or biological systems. Emergence is the architectural objective that HVC introduces into the proposed DCS management architecture - the ability to create behavioral novelty or operational independence between micro and macro elements of DCS. The intent here is to promote local and global autonomy on state management and policy-based influence or control within and across virtual cluster domains in an asynchronous way.

The HVC organizational model is structurally (not architecturally) consistent with what is proposed in [179] and the proposed model for the human brain's neocortex layering structure for sensory storage, prediction and intelligence. Alternatively, this thesis proposes a self-organizing and dynamic structuring model that has properties of hierarchical organization, abstraction and aggregation. Further, the use of asynchronous and event-oriented information storage and exchange allows for predictive approaches to management and control. In DCS, these distributed techniques are overlaid across the virtual node and cluster hierarchy to facilitate data dissemination and state synchronization that have traditionally confronted peer-to-

peer or cooperative processing (e.g., machine learning) approaches. The HVC strategy assists in the convergence challenges that are exacerbated with increasing network complexity by minimizing propagation, distributed synchronization and coordination issues through virtual clustering (i.e., localization) and operationally-organized hierarchy (i.e., centralization).

1.3 THESIS CONTRIBUTIONS

1.3.1 REALIZING DECENTRALIZED COMMUNICATION SYSTEMS

Chapter 3 introduces OverMesh [14], a research platform and testbed environment for DCS investigation and realization. Our research objectives were to validate a 'proof-of-concept' platform capable of employing the DCS architectural elements and to understand the merits of integrating peer-to-peer and multi-radio wireless mesh in decentralized permutation. The OverMesh platform and testbed environment provided a decentralized computing & communication system environment for evaluating mobile applications and services. Chapter 3 also expands on the vision and broader research agenda for DCS. Additionally, we used the OverMesh testbed as a discovery environment for innovative mobile usage models including group collaborative and mobile grid scenarios, classroom networks and as an office-computing alternative to preexisting centralized infrastructure. For the purposes of services deployment, we evaluated the advantages of using virtual machines in distributed fashion to support peer-to-peer applications or management services.

In realizing the OverMesh platform, we recognized that much of the research on wireless mesh networks focused primarily on network and lower layer solutions for improving service efficiency or network optimization. Additionally, virtual networks[14], [76], [80], [126] which have been designed for the wired Internet did not work as efficiently on wireless networks as expected given the layered separation between the upper services and the lower network layers. There is wide progress on developing overlay services [76] to assist in network wide objectives, typically in the

areas of wired QOS and resource management. Alternatively, we studied a number approaches in the areas of cross-layer services [135], [148] mostly associated with wireless networks albeit with similar goals. We employed cooperative and adaptive cross-layer control to reconcile the disparities between upper overlay services and lower layer wireless mesh networks. Using information exchange and layer specific service mechanisms (e.g., broadcasting or monitoring), we demonstrated efficient services on resource constrained wireless mesh networks through the integration of the two techniques. The information sources include the following:

- upper layer requirements, including application statistics, bandwidth requirements, priority categories, packet loss impact;
- network state conditions from lower layers in the protocol stack such as routing decisions, contention-less medium access control, retransmission limits, and physical layer modulation and coding schemes, and;
- network-wide information, such as information related to interference and congestion in the network, monitored and exchanged among network nodes.

Additionally, we employed policy changes or network actions in a local layer either in fast time scale or more proactively on a distributed overlay in slower time scale to respond to service or network conditions. Examples of this include:

- changing network configuration or route modifications when link conditions degrade;
- varying distributed monitoring policies on active route links or forwarding information about dynamic variations in link quality to nodes in the local routing table;
- changing QOS, security policies, flow transport or monitoring parameters;
- broadcasting information via the MAC to endpoints for faster response and local adaptation.

Through OverMesh experimentation and simulation results, we studied the dynamics of using link layer state and policy mechanisms to optimize distributed overlay services and investigated cross-layer methods using lightweight VMs and a distributed database for network information storage and distribution.

1.3.2 HIERARCHICAL VIRTUAL CLUSTERS

In DCS, operational dynamics occur at multiple virtual levels, creating a fluctuating environment to manage against stability. Additionally, increasing network complexity can quickly render the networked system useless if particular high-demand nodes or sections of the network are failure prone, unreachable or load challenged. In Chapter 4, we present hierarchical virtual clustering (HVC) as the self-organizing structuring framework for DCS. Various decentralized systems including peer-to-peer [189], sensor [184], [187] and ad hoc networks [186] have employed clustering concepts. However, unlike those contributions, HVC facilitates a tiered and logical strategy for organizing an operationally driven clustering structure within a fully decentralized environment. Moreover, while the use of clustering is prevalent for network transport [183], distributed control and management are the primary applications of HVC in this thesis. Nodes promoted (aka clusterheads) across the HVC tiers will serve multiple functional roles as they elevate towards the root of the distributed hierarchy. The HVC system employs an operational superiority process for node promotion across the hierarchy, aligning to the self-organizing objective. Logical structure can change asynchronously at different tiers of the hierarchy and at different time epochs. However, the higher tiers or logical nodes converge towards a more static and reliable structure that operates at longer times scales for global state aggregation and policy enforcement. The reverse is true for logical and physical nodes operating at the lower tiers of the hierarchy, where more structural changes and reactive actions are expected. Thus, network-wide visibility and policy control are incentives enabled through the promotional process. The emergence property exists at each tier of the hierarchy, where promoted (parent) cluster nodes manage global state and policy, while non-

promoted (child) cluster nodes manage (cooperatively) local state and policy within the associated cluster(s). Chapter 4 presents in detail the overall HVC architecture including the concept of cluster *emergent* and a *clustering manager* responsible for clusterhead selection, cluster addressing, messaging and operations. Chapter 4 also elaborates on the HVC promotional process, the emergence and self-organizational dynamics. Finally, as HVC expands virtual clustering structures and the self-organizing framework, we demonstrate the effectiveness of HVC for structural scalability and resilience by comparing static, dynamic and service-dynamic HVC node configurations against traditional physical configurations based on P2P, centralized or tiered structures.

Chapter 4 also presents a simulation model of the combined HVC and self-management environment. The model analyzes scaling and stability properties of alternative DCS scenarios with respect to structural scalability and stable operational management. We focus the evaluation on demonstrating the operational advantages of emergent management services integrated with an HVC-based structure against similar services implemented with traditional physical, tiered or P2P configurations. To this end, the results show evidence of improved stability through structural convergence across the virtual hierarchy for clustering formations and operational stability across the three operational services and network-wide scalability in balancing state optimization and service efficiency.

1.3.3 EMERGENT MANAGEMENT

Chapter 5 introduces the emergent management framework and architecture along with supporting results for the proposed emergent management system. An emergent strategy enables multiple layers of architectural separation using the HVC methodology to create tiers of local and global separation for state and policy management. In this

decentralized framework[2], separating the self-organizational aspects from the self-management novelty is a design objective. However, we use common operational parameters to drive organizational hierarchy and cluster promotion and to formulate state aggregation and policy distribution and enforcement across the virtual hierarchy. This consistency is a key aspect of balancing both the scalability aspects, driven by the HVC architecture discussed previously and the emergent management architecture discussed here. The following operational parameters influence the virtual structuring methodology and form the basis of the emergent system:

- *reachability* – a state condition or aggregate conditional state of a DCS network element or network subsystem describing an operational measure of the relative reach or connectivity;
- *stability* – a state condition or aggregate conditional state of a DCS network element or network subsystem describing an operational measure of the relative stability;
- *performance* – a state condition or aggregate conditional state of a DCS network element or network subsystem describing an operational measure of usage, throughput or latency.

The three measures collectively assess operational state of the communication graph across the virtual clusters and the complete virtual hierarchy. Emergent behaviors segregate across four (4) concealed planes of the operational control system. These include the (i.) *global plane* which associates to a higher degree of abstraction, aggregation, and distributed view of the networked system; (ii.) *local plane, which* associates more closely with discrete, event or real-time estimation and has direct peering and interactions with neighboring elements; (iii.) *state plane,* which represents a temporal and/or spatial status of any node element, cluster of node elements for some

[2] This is the basis for separating and positioning the HVC into its own chapter and contribution, independent to this chapter. It is the expectation in this thesis, that existing management solutions or other self- management work may be HVC integrated without context to the emergent approach being proposed in this thesis.

operational state of the networked system, and the (iv.) *policy plane,* which represents configuration, guiding rules and actions that control node elements, clusters or connectivity of the networked system.

1.4 THESIS ORGANIZATION

Starting in Chapter 2, we present the background including the challenges and requirements of evolving Internet systems along with a historical trail of previous work and research progress related to the main contributions of the dissertation. Chapter 3 articulates the vision and the system components of the DCS conceptual framework. Chapter 3 also introduces OverMesh an early proof-of-concept implementation of a DCS platform where we evaluate virtual overlays, wireless mesh and self-organization via experimentation and simulation. Respectively in Chapter 4 and 5, we expand on the major contributions of this thesis, specifically hierarchical virtual clustering and emergent management. Finally, Chapter 6 provides a summary of the thesis contributions and positions a futures discussion on other pertinent areas of research to enable decentralized communication systems by the broader research community.

Chapter 2

2 BACKGROUND & PREVIOUS WORK

To understand the redirection proposed in this thesis, one must observe the architectural transition of today's Internet systems from its historical basis. This chapter focuses on related, previous research in Internet networking systems that have direct correlation to the contributions in this thesis – namely flexible networking and dynamic resource management. Both topics have spawned broad research and commercial interest largely due to the challenges brought forward through the vertical nature of Internet platforms, the centralized nature of distributed computing and shifts in mobility due to pervasive wireless communications. The specific challenges that this thesis will deliver technical contributions include demonstrating the feasibility of building robust DCS platforms, constructing DCS structures capable of adapting to dynamic topologies for control and management and building and delivering emergent management services capable of addressing dynamic and resource constrained DCS networks. The issues of security and trust are notably relevant in building a robust DCS. More recently, the topics have gained wide research attention [80], [91], [99], [199]. While this thesis does not address the area of security, we suspect a wide range of decentralized systems security research contributions in the near future.

2.1 FLEXIBLE NETWORKS

2.1.1 OPEN NETWORKING

The Internet Protocol (IP) established a common transport from which underlying link-layer mechanisms are concealed and their dependencies removed from higher layers, while higher layer services and diverse applications have proliferated. Widespread use of heterogeneous wireless devices is also having significant influence on the Internet transport delivery architecture. Past initiatives [36], [40], [48] have promoted the development of open network architectures by offering a forum for

communication of theoretical and experimental results aimed at re-examining network control and management systems from traditionally constrained solutions. Several previous projects [11], [14], [16], [48], [67] promoted the shift from current monolithic and integrated Internet architecture and endorsed an open, flexible Internet architecture. Past progress in open networking match the PC industry in the early 1980's when the PC architecture enabled an expansion of IHVs and ISVs as evidenced by today's wide market of component hardware and services. As IP networks have evolved, router and switch component devices have advanced more so by their performance benchmarks than their service provisioning flexibility. Existing internetworking platforms lag the rapid development needs of the IT industry in terms of flexibility and provisioning speed as is currently achieved by the traditional computing platform. The need to extend network device capabilities and support their dynamic configuration enables network innovation to advance on an even scale with the computing platform. With the convergence of heterogeneous wireless networks with traditional Internet networks and the growing demand for mobile Internet services, the requirement for flexible node platforms is a practical path of evolution for future Internet architecture.

2.1.1.1 CHALLENGES

The deployment of emerging Internet services has introduced a growing number of networks devices or increased vertical integration of traditional internetworking device functionality. These devices add to rising depreciation costs and the necessity for new levels of operations staff who must integrate these services into the operational environment. The need to deploy such services into the existing infrastructure ad hoc and deployed without the incremental addition of new devices or operating system upgrades, reduces the depreciation cost factor and enables more rapid service provisioning. As a result, we can design and deliver a more flexible and cost-effective Internet.

The Internet transport was designed for simplicity, connectivity performance and reliable packet delivery. While these principles should continue to drive the core transport design for packet delivery efficiency, the controversial ideas to push more intelligence and services into the network (e.g., edge or access) has become a matter of evolution and customer demand. While the need for emerging network services continues to influence this evolution, several problems arise from the current service-provisioning model in today's Internet system. First, the introduction of new services into a network operator's environment is oriented to a network device or operating system upgrade or new device deployment. This generally reduces the useful life span of a network device and translates into a higher depreciation cost model over time. Moreover, provisioning cycles are managed over a number of months or years. This puts a strain on the service provider to adapt quickly to the changing business models and meet evolving customer needs. Second, as new services are introduced, the network management complexity and organizational burden (e.g., personnel training or operational overhead) is abrupt and magnified, rather than seamless and transitional. The reason for this relates (mostly) to the first issue since new services are introduced as another vertical infrastructure component rather than as an extensible service component of the existing infrastructure. Third, there exists little integration and interoperability between network devices or alternative vertical solutions. Lastly, network devices are not designed to provide feedback or transport network state for the application's specific needs. In what follows, we present the author's previous research that was motivated by these issues and sets path to the current thesis direction.

2.1.1.2 PREVIOUS WORK

2.1.1.2.1 OPEN, PROGRAMMABLE NETWORKS

In the PIN [38] project, we addressed the challenges of QOS interoperability between protocols and layers through Application Programming Interfaces (APIs) as adaptation interfaces. We proposed a set of QOS APIs spanning multiple network

programmability layers to support application, network service and device-level QOS programmability essential to the formation of an Internet service-delivery platform. These APIs defined the essential protocol invariant specification of QOS through a unified API framework for programming Internet QOS [38]. Rather than assume a single API can meet the diverse requirements at multiple transport layers, the PIN model described interactions between standard building blocks. The unified QOS framework provided sets of low-level APIs that comprise the resource-specific and service-centric QOS abstractions. Each level encapsulates the notion of building blocks that enable network device programmability. At the platform level, these APIs act as layered services that insulate the end-user from the complexity of network algorithms (e.g., admission control, reservations, or service level agreements). The interfaces were suitable to a network domain's specific QOS characteristics. In addition, a higher-level, architecture-independent interface established the QOS specification of Internet sessions. Internet applications can leverage this interface without need for detailed resource knowledge of the underlying network. Adapter policy objects convert architecture-independent interfaces to architecture-dependent interfaces, while maintaining policies for provisioning, accounting, charging and billing.

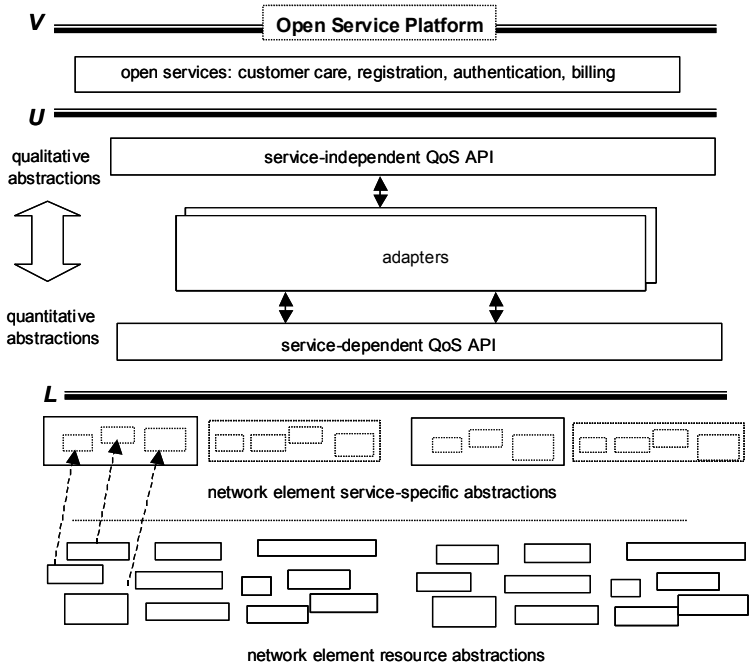


Figure 3: PIN model for programmable Internet QOS

A generalized model for network programmability structured without network architectural context is shown in Figure 3 with a layered, distributed API that exposes services for customization and programmability. Each level comprises a number of entities in the form of algorithms or objects representing logical or physical resources depending on the level's scope and functionality. The network programmability model provided generic and specific interfaces to abstract common network operations over any underlying networking technology. This enabled customization to support valued-added services by use of the underlying network infrastructure for the delivery of diverse services and QOS requirements. This customization was essential to a service environment that adapts to new services or requirements. Ensuring this customization is the assumption that the network does not impose limits to the diversity of providers of networks, services, content or other functions. The deployment of QOS on the scalable Internet is inherently complex due numerous providers and varying service levels. Thus, simplification through common and open APIs and deployments of such

standard interfaces may help manage or delegate the state information as the technologies allow.

2.1.1.2.2 NETWORK TRAFFIC CONTROLLER

The motivation behind the Network Traffic Controller (NetTC) [119] was the automation of QOS policy-based management (PBM). The concept is similar to the "Air Traffic Control" system at an airport, which has to manage the many variables associated with takeoff and landing of planes on an airport runway. The NetTC research objective was to develop an automated, rule-based PBM administrative capability to manage service level differentiation needs of application flows against network bandwidth constraints under the support of application-oriented traffic control and management specification. With the original intention of using QOS APIs [53], [118] for enabling local service differentiation, the use of middleware services provided distributed interfaces for implementation of the client NetTC agents with a centralized NetTC Administrator for distributed control and management.

Architecture

The NetTC software architecture consisted of four (4) major software components (Figure 4) including the *NetTC Administrator, LAN Segment Data Collector, the NetTC Console*, and the *NetTC agent*. A SQL server stored relevant configuration, policy and specification information, which was used by the NetTC Administrator to control and refine QOS services and managing global network bandwidth. The NetTC agent supported local invocation of Traffic Control functions on flows (e.g., defined filters and QOS flow specs) as instructed by the NetTC admin. In addition, the Agent was designed to integrate per flow information into local Agent structures when enabled. Per flow performance information is obtained through Windows performance utilities and through the QOS API functions for flow-level querying of scheduler performance.

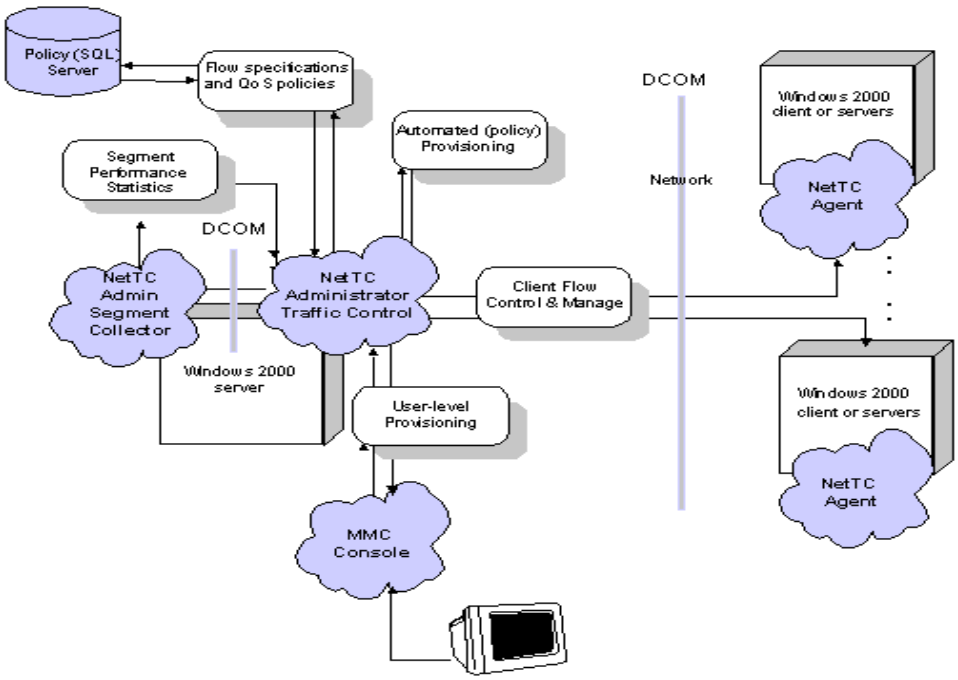


Figure 4: NetTC distributed architecture

Figure 5 illustrates the QOS API approach [53]. As highlighted, NetTC models a 3rd party application in its provisioning of QOS on the local host via the Traffic Controller SP, TC API, and QOS Packet Scheduler. The implementation required the use of layer 2 and layer 3 resources facilitated by the QOS API. The QOS Scheduler is installed on the local device to support the various functions and parameters of the scheduler.

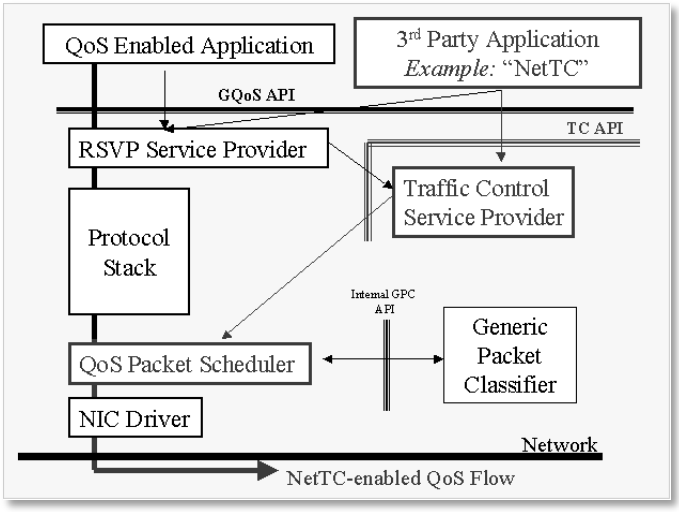


Figure 5: NetTC & QOS API integration model

Operating Model

The Traffic Control API supports traffic control functions shown in Table 1. The names are descriptive to their actions and invoked via the API for flows, filters and network interfaces. The TC APIs are *AddFlow* and *AddFilter*. *AddFlow* causes a flow to be created in the kernel network stack. The flow has certain actions and characteristics associated with it. These include marking behavior, packet-scheduling behavior and other media-specific behavior. *AddFilter* attaches a filter to a flow; a filter specifies classification criteria, which determine the set of packets that are directed to the associated flow.

Table 1: Windows QOS Traffic control APIs

TcCloseInterface	**TcOpenInterface**
TcDeleteFilter	**TcQueryFlow**
TcDeleteFlow	**TcQueryInterface**
TcDeregisterClient	**TcRegisterClient**
TcEnumerateFlows	**TcSetFlow**
TcEnumerateInterfaces	**TcSetInterface**
TcAddFilter	**TcGetFlowName**
TcAddFlow	**TcModifyFlow**

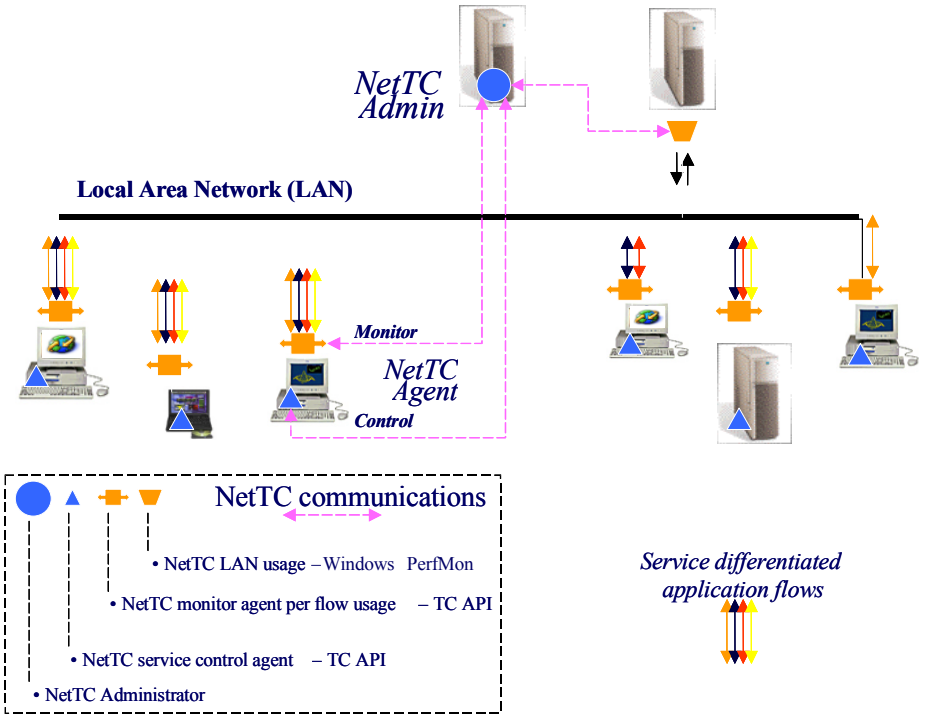


Figure 6: NetTC distributed deployment

One of the unique aspects of the NetTC traffic control model is the use of performance feedback information on a flow level and network-wide level. Feedback and automated decisions can be made on multiple timescales, in addition to manual invocations via a console to set policies, reset thresholds, police "greedy" flows or better manage the QOS needs of applications against the shared network resource by setting appropriate thresholds at different timescales and augmenting per flow specifications to reduce potential contention or spiky congestion behavior. To manage QOS provisioning and control an underlying algorithm was integrated as part of the NetTC Admin to manage using either automated methods or manual invocations from user-influenced provisioning and control. Figure 6 depicts the NetTC distributed implementation for administering and managing network QOS.

2.1.1.3 DISCUSSION

The NetTC project introduced automation through rule-based techniques for managing application QOS in a local network environment using a threshold-based trigger methodology to address the challenges of 'closed-loop' state management and provisioning response. The PIN project demonstrated the use of a programmable platform to facilitate provisioning end-to-end QOS through open APIs used to abstract the resource and functional heterogeneity in underlying networks. While both projects demonstrated the value of open programmability as methods for facilitating provisioning over heterogeneous networks, these approaches relied on manual design and static control to match complex and dynamic state dependencies or conditions (e.g., environment failures or topology instability due to mobile nodes). Additionally, the lack of support for on-line evaluation, learning and state prediction for response automation limits the flexibility to adapt to the dynamics of DCS environments. Further, given a centralized framework, provisioning distributed services in larger, decentralized scenarios poses notable scaling challenges for network-wide provisioning coordination. Finally, these projects illustrated how traditional centralized and closed networks limited online flexibility to augment or provision novel services into the network through rigidly designed internetworking nodes.

2.1.2 NETWORK VIRTUALIZATION

Distributed virtualization[3], based principally on a distributed virtual machine (DVM) strategy, has strong merits to serve as an architectural underpinning in future Internet systems [10], [15], [67], [82]. Infrastructure flexibility is one of the key design properties of virtualization. Virtualization is typically discussed in the context of data centers or compute servers, where multiple virtual machines are loaded onto a single host to increase server utilization or reduce the cost of buying new hardware. However,

[3] See [72, 82] for a good discussion on distributed virtualization based on progress from the PlanetLab research community. A recent NSF proposal [67] expands on PlanetLab with a new direction for Internet systems using distributed or network virtualization.

virtualization enables additional capabilities to allocate virtual machines at the location of choice or to deploy or migrate virtual machines; enabling a spectrum of new uses. With virtual machine monitors [68], [71] and hardware virtualization [73] service providers can deploy unmodified guest operating systems (OS) to employ safe experimentation environments for OS migration or for porting tools and services. Further, the ability to allocate virtual machines at the location of choice opens up other possible use cases - monitoring and debugging internal networks from multiple vantage points or validating internal security policies from different address spaces and networks. In deploying wireless systems, we have shown [14] through the use of cross-layer overlays that one can improve the flexibility and efficiency of heterogeneous wireless networks through integration of overlay techniques for distributed searching or to facilitate network monitoring & resource control using multi-layer information exchange and inference techniques.

2.1.2.1 CHALLENGES

Distributed virtualization creates non-trivial challenges for operations, increasing complexity, introducing new vulnerabilities for current management solutions. System integrators must address technology development challenges including solutions for incremental validation, reconfiguration or rollback procedures for deployed virtualization services. The solutions must be designed for scale and supported by commercial-grade operational facilities. As distributed virtualization becomes more prevalent, the federation of distributed virtual resources across the public Internet for corporate-to-corporate business computing and mobile services will necessitate a discussion of distribution of operations and management.

Today, making provisioning changes to the infrastructure is extremely difficult and slow, but required to avoid costly impact on mission critical services. While the goal is to avoid outages, the consequences of such a paradigm is hardened infrastructure, which effects the evolution to changing service demands against slow hardware and

software adoption cycles and increased operational time to implement changes. Network infrastructure must also integrate new applications seamlessly so legacy services are not affected by the changes. The need to deploy such services into the existing infrastructure in an ad hoc fashion and without adding new devices or operating system upgrade cycles reduces the depreciation cost factor and enables faster service provisioning. Service flexibility can introduce a higher degree of network automation and control; allowing the administration and deployment of network services to be sped up and operational procedures (e.g., change management) to be reduced or automated.

Ensuring application virtual resource allocation and operational resiliency in a manner that is equivalent to physical models is clearly a notable challenge. In supporting a virtualization paradigm, distributed applications are multiplexed onto shared physical hardware, making operational service interactions and reconciliation more difficult albeit with similar service-level expectations from the physical deployment scenario. Similarly, root-cause diagnosis and isolation of faults for virtualized network services poses similar challenges. While physical faults (e.g., power or thermal) may pervade a collection of virtual machines on a single machine, the diagnosis of a distributed VM service will require better methods and practices for failure associations (e.g., hidden dependencies) between physical and virtual instances. Distinguishing local, shared state based on VMM technology and shared state via distributed VMs is certain to be more challenging in diagnosis or performance analysis.

2.1.2.2 PREVIOUS WORK

2.1.2.2.1 VIRTUAL NETWORK RESOURCE MANAGEMENT

In [45], we described a *virtual network kernel* distributed across end-system and network nodes providing support for spawning of distinct virtual network architectures. The virtual network kernel builds virtual network architectures over the physical

network through the deployment of virtual network infrastructure. The parent virtual network kernel 'bootstraps' the child virtual network and then creates a set of *routelets* [48], [49] and *virtual links* that constitutes the virtual network topology. The child, like its parent, inherits the capability to spawn other virtual networks creating the notion of *nested virtual networks* within a physical network. The virtual network kernel creates a natural hierarchy through partitioning and isolation of virtual networks, and in turn, promoting inheritance and the autonomous control of network resources. Virtual networks form hierarchically through nested parent-child formations along a virtual network hierarchy structure. Virtual network kernels build organized hierarchy over the physical network thereby reducing the complexity of spawning virtual networks and handling nested networks through inheritance and state aggregation.

A key component of the virtual network kernel is management of spawned virtual networks. In this section, we describe *virtuosity* [49], a virtual network resource management system that minimizes the complexity of handling multiple spawned virtual networks that operate over multiple timescales on the same physical network hardware. Virtuosity is driven by per virtual network policy exerting control and management over multiple virtual networks and their spawned architecture by dynamically influencing the behavior of resource controllers over slow timescales. Virtuosity manages and controls virtual network resources on a slow performance management timescale that operates over a period of multiple minutes. This is a suitable timescale for virtuosity to operate over while allowing virtual networks to perform dynamic provisioning.

Architecture

The elements of the virtuosity architecture as illustrated in Figure 7 comprise of maestros, delegates, auctioneers, arbitrators, and monitors are instantiated as part of the child virtual network kernel and deployed as distributed objects within routelets. As shown in Figure 7 with the exception of the arbitrator, all other elements operate in the management plane. The arbitrator operates in the data plane. Through the process of

virtualization, virtual networks are separated from the physical or parent virtual network within a partitioned and separate name and address space.

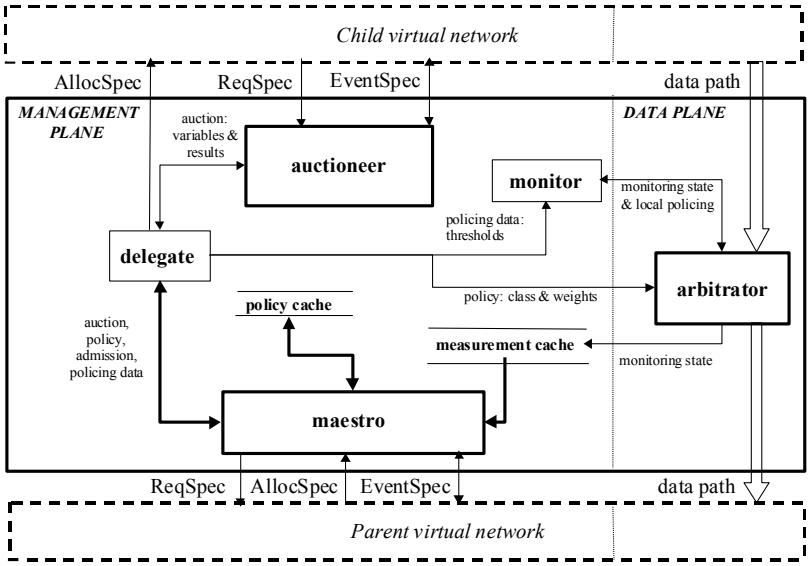


Figure 7: Virtuosity architecture model

The following are the Virtuosity design characteristics:

- *autonomous control* - spawning results in the composition of a child virtual network architecture, partitioning of parent network resources in support of a child's resource needs, and the separation of responsibilities and transparency of operation between parent and child architectures. Once a child network is spawned, the child has complete freedom to manage its resources and users' QOS in an autonomous manner based on its instantiated architecture;
- *dynamic provisioning* of virtual network resources is limited to either static or policy-based provisioning [63]. Virtuosity envisions a different form of provisioning where the capacity needs of individual virtual networks may change more dynamically in term of timescales and events. Virtuosity employs a per-virtual network policy-based approach that can be programmed by the subscriber to support a wide range of dynamic resource provisioning strategies;

- *capacity classes* within which child traffic classes (e.g., assured service, constant bit rate, best effort) are mapped and multiplexed. Capacity classes provide general-purpose 'resource pipes' allowing the underlying parent controller architecture to deal with child traffic in an aggregated manner. The mapping of the child QOS to parent capacity classes is made transparent to the parent and is the responsibility of the child virtual network architecture;

- *inheritance* through middleware services, resource management inheritance allows a child virtual network to transform itself to serve as a provider; giving it resource management capabilities and provisioning characteristics of its parent or, alternatively, to create completely distinct capabilities. Through inheritance, aggregation and the provisioning of a common set of capacity classes, virtuosity can efficiently support the resource management needs of multiple virtual networks. The nesting process allows us to push the complexity of the management of virtual network resources up the inheritance tree with the benefit of having to manage reduced state information.

Operating Model

The elements of virtuosity exist as part of the child virtual network kernel and execute as deployed distributed objects. As shown in Figure 7, with the exception of the arbitrator, which operates in the data plane, all the other virtuosity elements operate in the management plane. These elements are as follows:

- *maestro*, which is the key resource controller responsible for managing the global resource policy within the virtual network or virtual network domain[4]. The maestro operates on management or coarse timescale, resource availability and virtual network policy. Maestros set pricing and rate strategies across its managed

[4] A domain is defined to be a set of virtual network resources (i.e., virtual links, routelets and capacity) that the virtual network provider has authority for administration, pricing, control and management.

resources influencing child virtual networks in a manner to promote the efficient use of resources;

- *delegate*, serving as a decentralized proxy agent for a maestro, manages all local resource interactions and control mechanisms on a routelet as shown in Figure 7;
- *auctioneer*, which implements an economic auctioning model for resource allocation supporting various strategies between virtual network providers and subscribers. The auctioneer services bids from child virtual networks over slow provisioning timescales promoting a competitive system among subscribers. A *monitor* performs policing and monitoring on individual parent resources. Policing assures that child virtual networks are not consuming parent virtual networks' resources above and beyond an agreed allocation of the virtual link capacity being managed;
- *arbitrator*, which represents a transport module capable of abstracting the virtual network capacity 'scheduler' controlling access to each parent resource. The arbitrator receives a virtual network scheduling policy from the maestro over slow timescale provisioning intervals upon the completion of a resource allocation process. The virtual link arbitrator manages the access and control to the parent's virtual link based on virtual network policy.

Virtuosity manages the partitioned resource space and interfaces with the parent virtuosity system to increase or decrease the current partitioned resource space through dynamic provisioning. The arbitrator and monitor elements are instantiated on routelets on each port, managing the integration of provisioned capacity and local resource policy over each routelet virtual link. A single delegate and auctioneer are instantiated per routelet and manage local resource management activities on the routelet. The delegate is a coordination proxy working on behalf of the maestro to distribute local activities, while the auctioneer brokers the provisioning requirements from multiple child networks on the routelet. Maestro is the only virtuosity element that oversees the entire resource domain. While managing all domain resources, it seeks optimal global

policy and distributes (via delegates) per routelet auctioning parameters and per virtual link arbitration policy. Maestro, although conceptually a centralized controller, can be implemented on a centralized server node or decentralized using cooperating server-based agents instantiated on a per routelet basis.

2.1.2.3 DISCUSSION

Virtuosity was a dynamic virtual network resource management system. Key operational considerations in the development of the virtuosity framework were scalability and stability. Several scalability considerations associated with resource management architecture for managing virtual networks included network complexity, computational complexity (e.g., routelet management processing overhead) and transport data path impact based on frequency of control or management interactions. One of the scalability benefits of the architecture was achieved through the virtual network inheritance model. Through the maestro and delegate, we centralized management intelligence and processing but decentralized the interactive activity required between the delegate and the other virtuosity components. In addition, by selecting a single delegate model per-routelet, we scaled management processing with node complexity rather than link resource complexity. Finally, with the virtual network (capacity) scheduling approach, we simplified network provisioning by leveraging ideas of slow-time scale management and capacity aggregation, which helped to remove interactive concerns. The issue of stability in virtuosity was conditioned by programmed policy. It was important that policy-based dynamic provisioning guide the stability of the network. By limiting the provisioning timescales, we achieved a balance between the gains derived from statistical sharing of resources between virtual networks and the desired stability of virtual networks. The timescale was influenced by virtual network size and hierarchy complexity. There were several considerations in the trade-off between stability and resource efficiency. First, network services, which operate within the context of virtual networks can operate at some defined steady state rather than continuously fluctuate. Second, provisioning intervals must be lower-

limited such that admission control and auctioning processes for resource allocation can reach convergence within minutes. Finally, the infrequent requirement for child networks to do dynamic provisioning keeps the virtuosity system inactive.

Key associations between the virtuosity management system and the emergent management system include the use of a system dynamics approach to resource management through distributed auctions. Additionally, the use of both centralized and decentralized mechanisms via the maestro and delegate are employed as a cooperative strategy. Using monitor and arbitrator objects, the network state and policy control system formed a closed-loop system with an asynchronous process for state and policy synchronization. Hierarchical resource control and management is inherent to the virtuosity framework through the virtual network inheritance methodology. While predictive methods are not employed in virtuosity, the use of capacity planning provides for proactive management through historical data aggregation and trending. However, virtuosity does not address the decentralization concepts, where end-nodes orchestrate in peer-to-peer or leader-based communities. Alternatively, it is heavily driven by rule-based methodologies for triggering responses for auctioning control and resource state perturbations. Nevertheless, virtuosity has many similar characteristics to the HVC-based emergent management architecture as expanded in Chapters 4 and 5.

2.2 DYNAMIC RESOURCE MANAGEMENT

Broadcasting or multicasting system-level information in smaller-scale networks or directing exchange of information between nodes with information propagation is proposed for cooperative control and management. System-wide adaptive management of resources can ensure that the system is made more aware and resilient to varying constraints or that the best option is taken during improved network conditions. In [14], when the wireless link is feasible, P2P transmission helps to reduce path loss over large distances in wireless links with multiple hops employed over shorter distances to reduce packet loss. In multi-hop networks, one must be concerned with both exposed nodes (in the sender's range but out of the range of the destination) and hidden nodes

(out of the sender's range but within the range of the destination). The network configuration between two endpoints may vary dynamically due to varying network conditions, mobility, or other constraints in the system. Thus, QOS policy or state information exchange through intermediate nodes is necessary to meet end-to-end service quality expectations.

In large wireless ad-hoc networks, routing tables with link quality information can grow significantly in size. To achieve scalability, nodes can store only local information about nearby links. This information, stored in distributed fashion, can be propagated through nodes on request to correlate end-to-end performance on a communication path between two endpoints in the network. The responsiveness of a node's adaptation mechanisms to changing mobile operating conditions will determine how effective these mechanisms are in responding to the variations in the network or the application. Thus, a key decentralization challenge is the tradeoff between local information scaling and responsiveness due to information sharing or propagation - a challenge this thesis aims to address using the HVC through cluster-based localization and hierarchy-based global distribution.

2.2.1 CHALLENGES

On a network-wide level, the associated structures and traffic models for wireless decentralized systems are highly distributed and dynamic. Mesh or ad hoc networking systems including the users, applications, underlying network services and the infrastructure services cannot realistically rely on any fixed node structure or transport environment. Additionally, given a lack of centralized authority and administration for control and state management, the coordination complexity and operational administration is well beyond many of today's automation or autonomic approaches to distributed management.

Alternatively, the ability of applications to adapt to positive or negative changes in wireless conditions by leveraging in-network services or inter-layer node mechanisms

will give applications greater flexibility in managing real-time or media adaptation within a wireless environment. The application's adaptation flexibility will depend on its ability to detect or respond on a much faster time-scale (e.g., to support fast handoff), thus requiring the application to cooperate with the transport layer's congestion control loop. Alternatively, the necessity to manage the wireless channel bandwidth will depend on the clocking rate and control mechanisms used by the application or session layer to control the incoming rate of the flow. Sharing knowledge or policies between these layers can further increase their cooperative effectiveness.

2.2.2 PREVIOUS WORK

2.2.2.1 ADAPTIVE WIRELESS RESOURCE MANAGEMENT

The adaptive techniques at a single node are not sufficient to address overall scalability issues in the wireless or hybrid wireless networks. Optimizing overall end-to-end QOS requires knowledge or state of key elements of the network system or the communicating session. The optimization across the network is required due to variations in link conditions and user mobility constraints in the wireless environment. Local conditions at the MAC and PHY layers can be propagated to the network layer at each node, and joint optimization between the network and MAC layers can be used in conjunction with network-wide information to optimize routing and end-to-end QOS dynamically in the network.

Network-wide adaptation can be achieved through rapid exchange of information about individual links between endpoints to assess end-to-end performance. Physical layer conditions on each link in a wireless network can vary dynamically due to several factors such as network congestion with other users, interfering signals and noise, and path loss. The optimal path between two endpoints in a network is a function of the quality of each of the links on the path between the endpoints. With dynamically varying constraints in a wireless network, a statically configured optimal path may soon

become a less optimal one. Alternate paths can be proactively established to allow switching paths in the system as conditions vary over time. In [130], we proposed a system approach to balance wireless and mobile channel conditions against IP-based service differentiation and end-end service requirements.

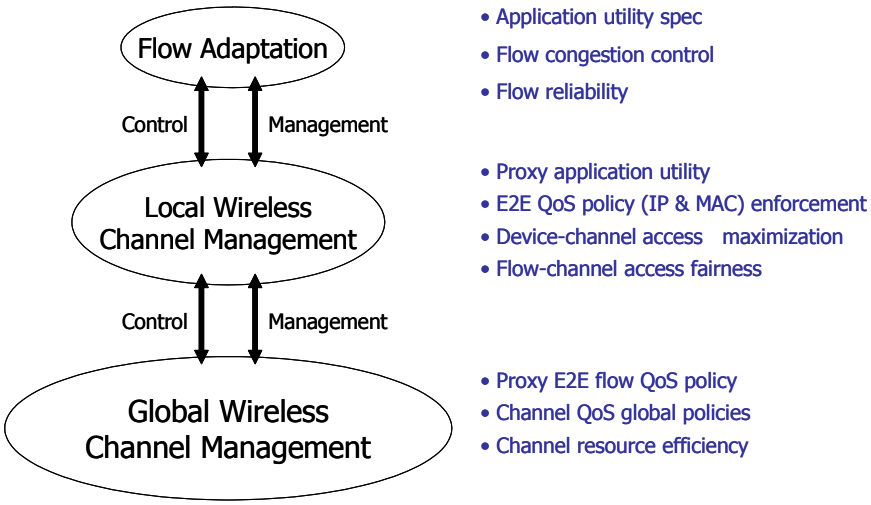


Figure 8: Conceptual model

Figure 8 depicts a simple, conceptual model of the system approach to support flow adaptation in concert with local and global wireless channel resource management. As shown, both control and management signaling services are used to maintain proper end-to-end flow delivery, optimal channel access and channel efficiency. By control, this refers to such services as QOS, congestion control, reliability and bandwidth control. By management, this refers to monitoring mechanisms to support appropriate feedback to optimize control or policy-based decisions. Control policies are based on conditions specific to the application flow, the local wireless device and the global channel. On a flow level, the conditions are at the scope of the application or connection transport level. The local device refers to the mobile endpoint, which may have time-varying conditions (e.g., fading, overlapped cells) imposed on its aggregate flows within its local vicinity, but may not necessarily reflect global conditions affecting all mobile devices within the span of the radio network.

Alternatively, global conditions (e.g., channel traffic load) may affect the entire channel, and thus, all wireless devices within the same local area network. The approach does not suggest the three state machines are decoupled. Instead, they must be cooperative in such a way that each operates autonomously, while policies and states for their operation are exposed or exchanged for the intended operational balance of the system. To achieve this, different components cooperate through exposed interfaces for binding configuration or state management. Furthermore, one can have greater control on the stability and efficiency of the system by enforcing policy controls at different timescales as warranted by the mobile device, application or wireless channel.

A middleware layer is employed to automate cooperative control between a central wireless IP radio resource manager (RRM) handling global resource management and multiple, distributed RRM Agents at the mobile clients (and base stations). These agents work on behalf of the RRM to distribute global policies, while also managing local resource management and reacting to local channel conditions. The RRM Agent can be seen as mediation control point between the global channel (layer 2 and 3) policies, local (layer 2) channel policies and application flow adaptation at the transport-level (layer 4) and above.

Architecture

Figure 9 depicts a general model supporting alternative wireless radio infrastructure. The framework supports alternative radio technologies layered on IP-based network services. Using middleware services, the IP stack is exposed to support a more integrated QOS and resource control system, while preserving its layered boundaries. In [119], we used a programmable classifier and scheduler based on a traffic control API [53],[118] to dynamically, configure or enforce IP-based QOS policies (e.g., marking, shaping, metering, dropping, priority, etc.) based on network monitoring feedback. We extended the work in [119] by recognizing the need to feedback time-varying conditions and further, by distinguishing local feedback conditions from global conditions in making proper resource and QOS policy decisions. We also extended the

previous work by enabling features that are more dynamic across multiple layers and providing support for utility-based specification, allowing the wireless application to adapt to either SINR or channel congestion feedback.

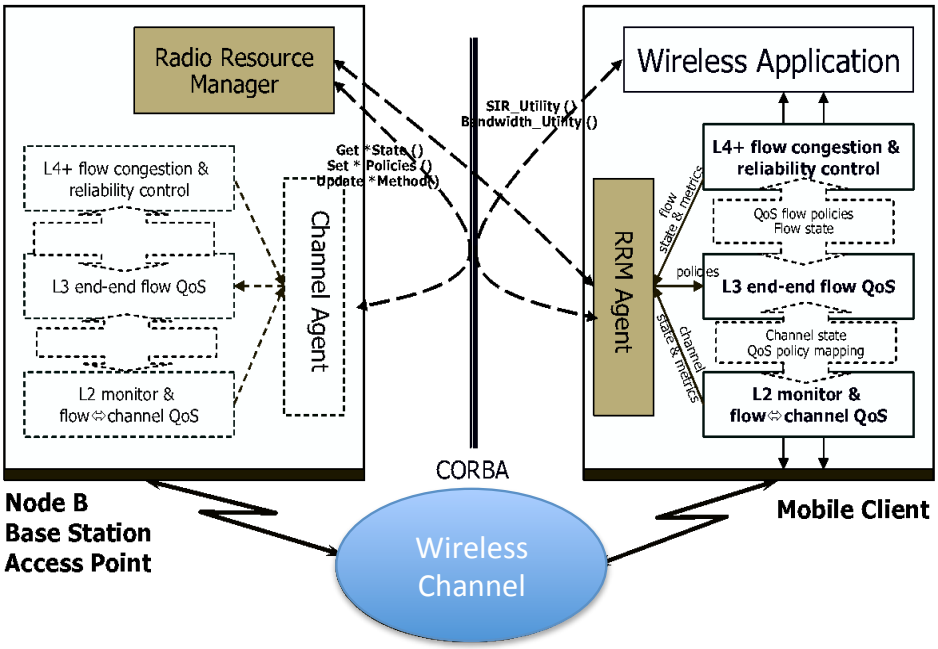


Figure 9: IP Radio resource control framework

System Components

The Radio Resource Manager is a central resource controller operating at an access point, base station or wireless LAN router. The RRM primarily oversees wireless resource control and management by negotiating wireless channel bandwidth requirements supporting its wireless clients while maximizing global channel bandwidth efficiency. The RRM uses method invocations to adjust provisioning policies or program layered component services resident at one or more wireless endpoints via the RRM Agent API.

The RRM Agent is a distributed extension of the central RRM. The Agent is a mediation point balancing global channel resource policy, local channel policy, and flow-level adaptation. It communicates with local applications accepting utility-based SINR and QOS specifications. Using RRM APIs, the agent communicates the

specifications to the central RRM to support global channel optimization algorithms operating at the RRM. The agent supports service programming (e.g., buffer management algorithm) or resource provisioning control (e.g., marking, rate shaping policies), directly or via of the RRM. Previous works (e.g., [121], [123]) have proposed alternative transport services necessary to match the requirements of wireless and mobile infrastructure in managing to faster congestion and drop type detection. A programmable transport, where alternative transport control schemes can be employed or configured in real-time can provide flexibility to the varying conditions exhibited by the wireless channel.

An important aspect of the RRM Agent is to coordinate policy and state between protocol stack layers to maintain constant synchronization. Such coordination may happen directly using header information (e.g., IP options), inter-layer header mapping (e.g., IP DSCP to 802.1p mapping) or via the RRM Agent through method invocation and parameter passing. The latter can be used for requesting or responding with monitoring state or QOS enforcement policies.

In [8], we demonstrated that a flexible network layer QOS mechanism allows automated provisioning and reconfiguration through threshold-triggered remote method invocation. Using centralized (multi-threaded) resource management algorithms, alternative IP flow QOS policies can be enforced remotely over a common API exposed by client QOS agents. In this work, we proposed a similar IP-based provisioning service at wireless devices, but allowed alternative IP QOS bindings and algorithmic choices enabling greater design and provisioning freedom to the QOS programmer or administrator. Through coordination with the RRM Agent acting as a local proxy, the IP QOS service can be (per application flow) configured through global policies and managed by the centralized RRM. Finally, alternative MAC level QOS bindings or algorithmic choices can be programmed into the MAC layer and reconfigured dynamically by the RRM Agent as needed.

Through coordination with the RRM Agent, differentiated access can be reconfigured through local policies algorithmically determined through SINR [123] utility curves managed by the RRM Agent. Thus, bandwidth, access, and latency differentiation is made possible at the local link level. In this scenario, the local policies may preempt the IP QOS global policies in order to manage time varying or fairness issues caused by fading or degraded local conditions on one or more flows. These policies may conflict with the end-to-end or global channel policies enforced at the higher layer. However, the higher layer QOS policies may be triggered active as local channel conditions improve.

Operating Model

Under a wireless environment, we partition the control system across the resource control hierarchy towards achieving consistent control on a global network level, a local network level and flow level. Each is directed at a different set of objectives, but overlapped on their influence on the wireless channel resource. As illustrated in Figure 10, three autonomous levels of feedback control support the distributed state machine. At each level, a stable and unstable state exists, while an operational state is centered between them to represent the control state. Also at each level, a monitoring service checks against stability thresholds to determine the possibility of instability and the need to enter into a control state, invoking alternative algorithms, which manage the particular level of concern. At the global and local level, policy changes will cause the state machine to enter into an unstable local channel state or unstable flow state, respectively. Multiple instances of the state machine process will run - one per wireless client and one for each of the flows running within the wireless environment. The flow procedure is essentially part of the normal transport process supporting both congestion and reliability control for each session flow. Also shown at the local and flow level is a procedure to update the application on specific bandwidth availabilities and SINR state, respectively.

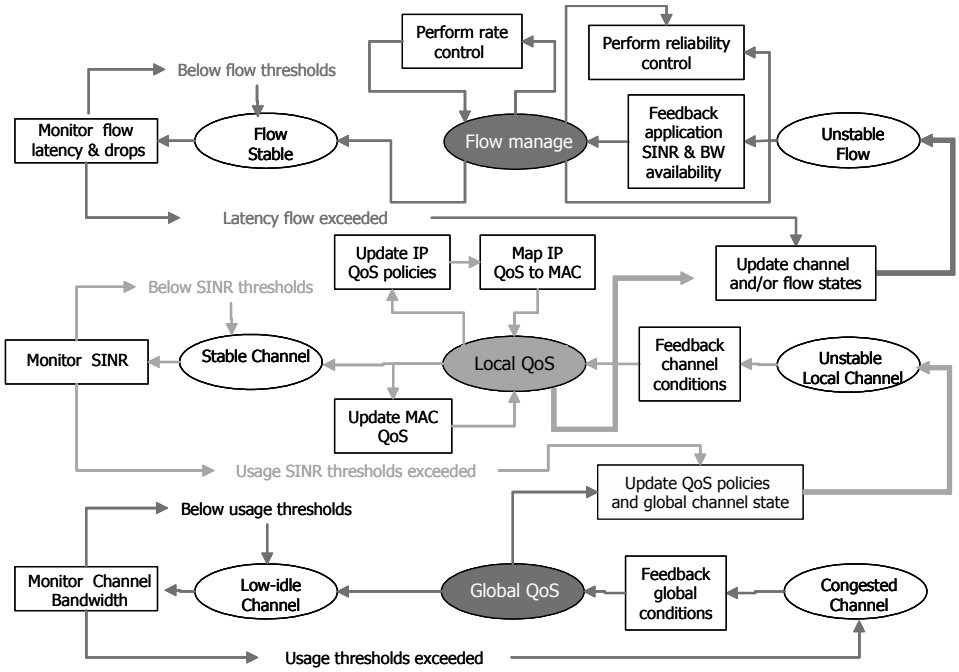


Figure 10: Distributed state machine

2.2.3 DISCUSSION

With the focus on adaptive wireless resource management, this project produced several important contributions to position the emergent and self-organization direction. First, the hierarchical approach to managing the dynamics of the wireless environment positions both a macro- and micro-granularity to manage state and control (policy) execution. Second, the 'closed-loop' and asynchronous nature of the distributed state machine for loosely-coupled state management and policy-driven control is evident in the same RRM framework. Third, the adaptive, cross-layer requirements position a similar objective for addressing the gap between the network and application discontinuity. Finally, distributed programmability promotes an architecture for endpoint (rather than inter-networking devices) rapid service provisioning and flexibility. This is consistent with the HVC direction, which aims for greater flexibility in the binding and service composition in client or endpoint devices, albeit orthogonal to the virtual machine orientation to service provisioning. However, the adaptive wireless resource management system falls short towards the direction of an emergent strategy for network control and state management. First, we employed (human-

defined) rule-based methodologies using static thresholds for inter-layer, hierarchical state and policy management synchronization rather than predictive/learning methods or more broadly complex adaptive methods. Further, there is continued focus on network programmability enabled by process-driven methodologies for service control rather than a state-driven methodology for dynamic service control.

2.3 SUMMARY

Low-cost wireless communication systems are introducing an inflection opportunity for unwiring users and creating free association of communication services on more horizontal and compute-rich devices. This change is consistent with the ubiquitous computing [1] direction and the limitations of today's Internet and telecommunication networks present an opportunity to bridge a transition towards alternative mobile computing and communications systems. The limitations of these works towards DCS are consistent – there is emphasis on human operator design for managing the state space and the corresponding network control response. Further, such systems are incapable of learning state, adjusting policies and making online prediction to help scale to new and more complex network state and configuration. A key transition is occurring towards applying concepts of natural or system dynamics to manage increasing information technology complexity and scale. The previous works present the author's thought evolution of research, while summarizing key shortcomings consistent with more recent network management and provisioning developments in emergent research.

How do we build scalable, dynamic structures capable of adapting to dynamic topologies? In DCS, the barriers for scalability are greatly increased because the distributed systems limitations can occur at multiple levels, as the architecture is highly decentralized and dynamic. The continuous operational state and policy changes due to a high-degree of node mobility or node (self-serving) behaviors occur at an application-level, network services or infrastructure-level.

Chapter 3

3 DECENTRALIZED COMMUNICATION SYSTEMS

3.1 VISION

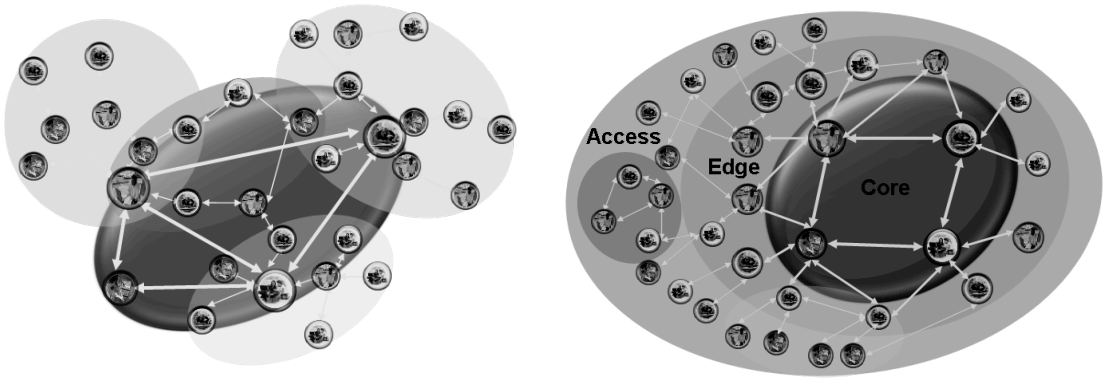
Today's centralized computing and communication systems are based on a usage model that is fundamentally orthogonal to current mobile, socially networked population and transient usage scenarios. Like mainframes in the '70s and early '80s, today's predominance of client-server or communication systems have reached a level of commercial quality or enterprise grade, which decentralized systems are far off today. The premise in this thesis is that decentralized systems can reach similar levels of resilience, scale and commercial reliance that the industry has realized through centralized systems. Our vision addresses this discontinuity and proposes transparency of the user or user groups between hardened centralized (e.g., data center) systems and ad hoc decentralized systems. This thesis proposes to decentralize Internet computing and communication systems by integrating people within the design of the infrastructure, delivering application information or mobile services based on location, social or group context using virtual machine, peer-to-peer technologies and offsetting traditional operational management support using principles of emergence and self-organization for scalable operational control. One emerging use case for decentralized systems is in developing countries where centralized infrastructure tends to be frail or cost prohibitive and common information technology services and support are challenging or absent. However, decentralized usages can extend to enterprise business computing, educational classrooms or remote collaboration, vehicular communications and processing, community networks (e.g., gaming networks), first-second responder systems (e.g., communications infrastructure disaster situations) or residential or hot spot networks.

To achieve the vision of building robust decentralized communication systems (DCS), the proliferation of computing, storage and wireless communications to the user (mobile) population through mobile devices must be cooperatively resource and service brokered with hardened, persistent environments (e.g., data centers) through wireless mesh networks for communications relaying, peer-to-peer computing and content synchronization. Scaling the network infrastructure requires wireless networks formed of cooperative, mesh networks between multi-radio portal infrastructure (when available) and multi-radio-enabled mobile endpoints. Dynamic topologies capable of restructuring to serve the opportunistic environment through peer-to-peer physical or virtual structures of mobile devices and hardened infrastructure can share, discover, provision or consume services. It is not difficult to envision a large grid of computationally rich, mobile devices and a larger number of virtual machines to achieve greater provisioning reach, capacity efficiency and service capabilities as seen in today's high-end server farms or computing clusters. A significant challenge in realizing this vision is sustained operational stability and scalability. Correspondingly, decentralized management, using principles of emergence and self-organization can be employed to achieve community cooperation, decentralized trust, state or policy propagation to facilitate operations, maintenance and provisioning functions. To this end, we introduce structural concepts of hierarchical virtual clustering for self-organization based on operational superiority (e.g., military or corporate organizations) and peered intra-cluster cooperative behavior. The use of virtualization or virtual machines employs the HVC infrastructure. In DCS, there is greater dependency on static or mobile virtual machine or virtual container[5] constructs that can provision or consume services or just act as service intermediaries to other DCS services.

[5] A virtual container is a more general abstraction of a virtual compute execution object, of which a virtual machine is one type; virtual containers can support different levels of granularity in execution type, local state and persistence.

3.2 CONCEPT ARCHITECTURE

Decentralization is the first key characteristic in future computing and communication architectures. For communications, one can envision a more flat network system with a large number of end nodes with similar physical form and function, but with varying resource and service profiles. These nodes participate in network transport, network control and management. In this framework, end-users are associated with these nodes either as a leaf or as an internetworking node. Communities of virtual networks may form by contemplative design or through social cooperation. Physical (wireless) connectivity is instantiated opportunistically without ownership or hardened allocation. Figure 11 illustrates this vision with two (of many) alternative topologies of DCS.



a) Sparse networks *b) Structured networks*

Figure 11: Decentralized networks

The proposed system is architecturally distinct to current ad hoc & sensor networks [101], [102], [106] as it is with peer-to-peer computing [96]. Richer node constructs formed of wireless multi-radio mesh communications and computational virtual machines are used to scale DCS. Extreme peering or viral [11] properties are exhibited at leaf or clustered DCS service nodes while centralized (or promoted) DCS clusterhead nodes in virtual constructs cooperate to deliver a decentralized capability. Networked virtual machines through computational overlays employ the emergent management service infrastructure. However, the virtualized networking infrastructure is not limited to management services and can be used to support (i.e., multi-tenancy) the deployment

of virtual infrastructure for alternative DCS services[6] execution or delivery. The following are the key DCS architectural properties:

- *decentralized computing infrastructure*: combining peer-to-peer computing and mesh networks with an aggressive convergence strategy for node computation, network processing and data storage to deliver a new class of node architectures for DCS. Any physical mesh node may be capable of supporting switching or routing functions, in addition to supporting client or compute functions. With hardware [73] and software-based virtualization technologies[68] gaining broad commercial adoption, node virtualization can evolve towards converging, partitioning and integrating computation, communications and storage resources;
- *virtual network structures*: based principally on a virtual machines and networked overlays, the use of hierarchical virtual clusters is employed to create pliable infrastructure for provisioning and managing network resources and distributed services;
- *emergent control and management*: exhibiting behavioral novelty in the separation of local and global control and management services in a self-organized, hierarchical framework.. Inference and learning techniques relieve human-dependency on operational management and provisioning tasks to capture and manage distributed state and adapt policies for distributed control.

In what follows, we present the major aspects of the proposed DCS vision and architecture along with OverMesh, a proof of concept research platform for investigating DCS architectures.

[6] Captured here for the purposes of vision articulation - the primary use or application of the virtual infrastructure in this thesis is aimed at emergent management for DCS scaling.

3.2.1 DECENTRALIZED COMPUTING INFRASTRUCTURE

3.2.1.1 SCALABLE MESH NETWORKS

Wireless mesh networks have been actively studied [102], [111]. In a DCS network, nodes are allowed to communicate with other nodes without being routed through a central switching point. For a network to intercommunicate in a mesh topology, the nodes' self-discovery features must first determine whether they are to serve as access points for wireless devices, as backbone nodes or a combination of roles. Individual nodes locate their neighbors using discovery query and response protocols. Once the nodes recognize one another, they measure link quality and performance metrics such as received signal strength, throughput, packet error rate and latency. This information is communicated among the neighboring nodes for selecting signal values. Each node then selects the best path so that the optimum quality of service is obtained at any given moment. The network discovery and path selection services must be lightweight, run in the background and consume minimal bandwidth. Each node maintains a current list of neighbors and frequently re-computes the best path with frequent node migrations or disconnections. This self-healing or failover features distinguish mesh topologies apart from hub-and-spoke networks. Mesh networks rely on management, control and discovery messages that must be protected along with user traffic via standards-based security techniques such as 802.11i and Advanced Encryption Standard (AES).

Mesh networks should be designed to provide for scalable [104], [107] capacity as the number of mesh nodes increase in the network. Such capacity-enhancing techniques for network scalability include power control, which can be used to reduce the range of interference provided by a wireless transmission. When multiple channels [115], [110], [122] are available, mesh nodes can configure their radios to transmit data in different channels and thus transmit simultaneously even if they are in close proximity of each other. Each mesh node can have multiple radios such that these radios can be configured to receive and transmit on different channels simultaneously for increased

efficiency. MIMO antennas can be used in each radio to provide increased capacity at the physical layer with the use of multiple antennas for transmission and reception. One can consider hybrid mesh networks that provide support for multiple wireless protocols operating in non-interfering frequency ranges. Communications using different wireless protocols may exist simultaneously and the nodes in the network providing support for diverse radios to support various protocols or alternatively, software-defined-radio [105] implementations to reconfigure radios to different wireless protocols dynamically. In addition to capacity-enhancing techniques for network scalability, mesh networks provide capabilities for adaptive routing [107], [109], [114], [115] and fault tolerance. Heterogeneous mesh networks can provide this additional flexibility. As an example in Wi-Fi & WiMAX heterogeneous mesh networks, a WiMAX base-station node can serve as a mesh portal or it can just forward traffic to another Wi-Fi mesh portal of reachable proximity. For intra-mesh traffic, one can optimize transmission flows by using a WiMAX network to reach destinations faster by traversing intermediate paths in the mesh network through the WiMAX node.

3.2.1.2 FLEXIBLE NODE CREATION & CUSTOMIZATION

Today, Internet nodes are highly physical function clients, servers, routers, switches and various other forms of physical ‘appliances’ forming its source, sink and interconnection structure. Endpoint devices can be oriented towards smart phones, cell phones or any number of alternative client devices to connect, source and sink Internet traffic. Nodes and their physical position in the network have a one-to-one mapping (generally) to their specific function or service, which they support in the network. DCS proposes to move away from this orientation and position a virtualization approach to customizing node computation, network processing and data storage to enable flexible node creation and customization. Therefore, any physical node may be capable of supporting alternative properties of interconnection, source and sink computation and

storage inside the network. The use of virtualization[7] technologies enables provisioning alternative forms of networking as authorized by the owner of that device.

Figure 12 illustrates four alternative forms of node virtualization, each of which directs a unique form of node formulation and construct. Figure 12 depicts the traditional distributed set of servers formed by combining several nodes to demonstrate clustering through virtualization and alternative forms of interconnection devices such as switches, routers, network appliance and a diversity of endpoint platforms supporting traditional client device functionality. Finally, such node modularity may also allow transforming or personalizing user-centric devices to serve or internetwork to the user's need or demand, limited only by physical capacity.

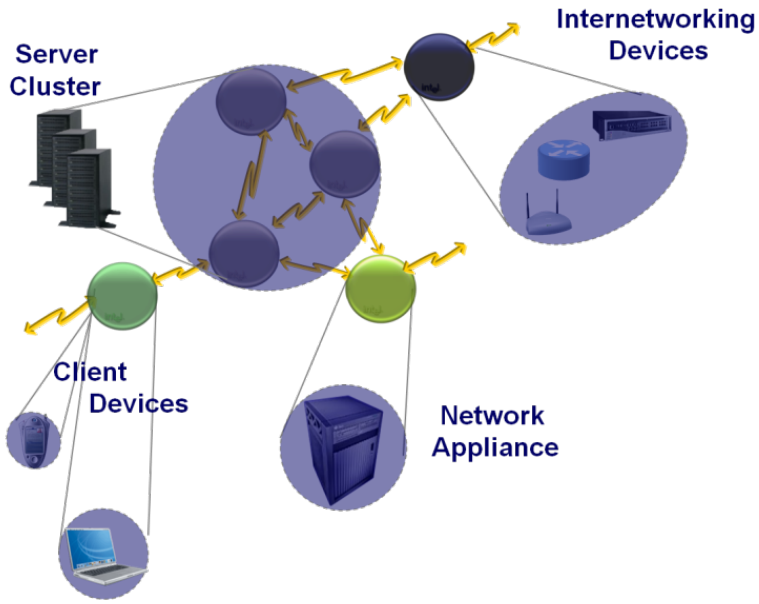


Figure 12: Customized node schemes

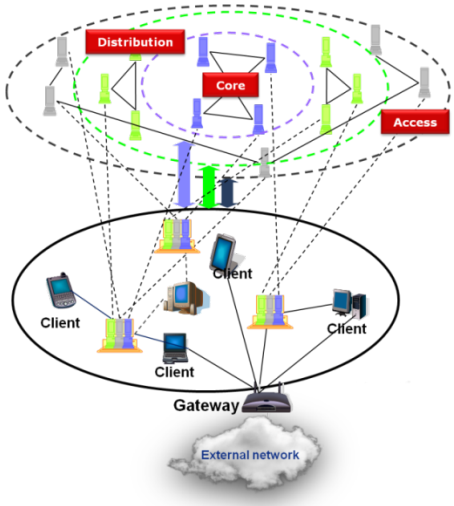
[7] The practical realization of this topic extends beyond the subject of virtualization. Other enabling technology areas that we find suitable (e.g., multiple processor cores and cognitive or soft radios) are missing here, but are duly acknowledged as opportunities to realize a vision of a customizable DCS node.

In summary, flexible node creation and resource provisioning based on virtual containers or virtual machines are proposed for supporting the creation of node functions, partitioning or integration of device services over alternative physical platforms. Personalizing devices to the user's particular need through virtualization, rather than device physical instantiation is a key concept in this thesis, and thus extending the vision of the proposed DCS framework. This aligns with the notion of empowering the user and enabling the user's free-association and composition to a broader set of services or virtual resources within their reach or awareness.

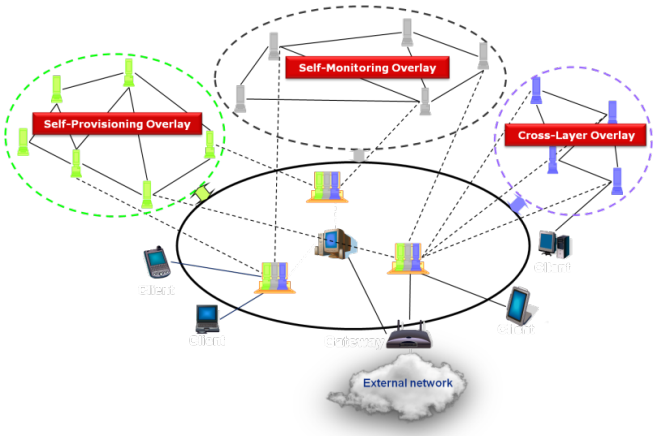
3.2.2 VIRTUAL NETWORK STRUCTURES

With the challenges visible in today's Internet design, there is an architectural shift underway in the Internet from the original IP abstraction model to a virtualization-based abstraction paradigm. Largely motivated by the changing demands for user mobility, security, context flexibility and massive content evolution, this shift is necessary as an architectural transition. While many commercial virtual machine services have been positioned primarily as a vehicle for stateless computing or computational processing, the use of virtual machines or containers to provision communication services or for security or management purposes has gained broader momentum from the commercial and research community [90],[92]. This section extends the discussion on virtualization and positions a visionary direction for network virtualization to enable future DCS systems. A key aspect of DCS research is the development and use of a distributed virtual infrastructure for network provisioning and service deployment. In such a model, the service architecture would have structure and organization in a similar way that processes are hierarchically spawned or inherited in an OS and IP is layered in packaging for basic or enhanced delivery or reception of packets. Alternatively, in DCS virtual structure can be hierarchically spawned and layered within node virtualization constructs and distributed similarly in topological form. As illustrated in Figure 13(a-c), example construction formations can exist to

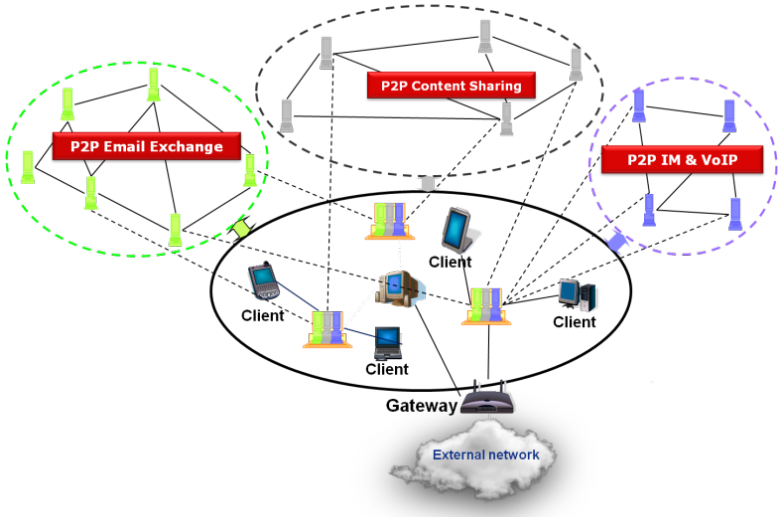
support infrastructure topology services, traditional management services, and peer-to-peer collaborative or social networking services.



(a) Network structures



(b)Management services



(c) Collaborative, social networks

Figure 13: Alternative virtual networking structures

Matching the DCS dynamics for node mobility, node and image migration and ad hoc community formations, different classes of construction can envisioned including *hard structures* with fixed network and long duration, *soft structures* with variable network and long duration and *short-lived structures* with variable network and short duration.

An important objective of the DCS research is the feasibility of such architecture to deliver VM-based layered services over wireless mesh networks. DCS services must be bundled or organized in scalable fashion to be easily usable or programmable by an application developer, service provider or even an end-user to deliver commercial applications or social networking capabilities. Additionally, alternative types of virtual network structures and their interactions must operate over different mesh networking constraints (e.g. traffic load or signal fading conditions) and mesh node constraints (e.g., compute or network load or storage requirements) with stability and scalability.

3.2.3 EMERGENT CONTROL & MANAGEMENT

As articulated in [12], [94], and [142], the tasks of managing and provisioning Internet networks remain a manual activity for network administrators. An interesting observation posed in [30], [179] suggests designing systems based on the notion of predictability and its relation to process driven or data-driven descriptions. The current Internet lacks the online flexibility to change its design against evolving demands and conditional event dynamics. As a non-traditional approach to network control and management, the introduction of online predictability using emergent and self-organizing techniques [24], [131], [141], [175], [178] is advocated herein. In DCS, management and resiliency are key operational pillars in its architecture. A characteristic to be drawn out in this thesis is the intrinsic ability to have local (e.g., local service, network component or subsystem) elements operate with distinct and independent behavior exhibited by the global system (e.g., network-wide service, internetworking community). Moreover, the global system may have a different set of objectives than the local entities' while their synchronicity and independence should be evident.

Another aspect of the DCS architecture is self-characterization of its distributed state profile. Network state can be static or dynamic, local or global, or operational or policy-based. For any given DCS resource or service, there are key operational

attributes that can characterize 'goodness', for example stability, reachability and performance. The DCS emergent system must be capable of capturing (online) and storing event-level representations as well as temporal data representations. Similar to the concepts in [12], a distributed storage facility is integrated into the design of the network and operational at runtime. In DCS, unconventional approaches are needed for network control and management. This requirement comes both by necessity to increase operational resiliency for the users of the networked system and for architectural purposes to reduce the dependency on personnel to operate and manage DCS. The proposed DCS assumes a high-degree of internetworking fluctuation and mobility. Nodes can be online, mobile, hibernating or off-line. While the topology may be irregular and fluctuating in path selection, it may also increase the network diversity for provisioning. To enable distributed learning and evolution, machine learning [99],[168],[171] techniques are employed to predict distributed performance and reliability state; entropy-based techniques are used to characterize network or sub-network availability or stability state, graph expansion and clustering methods are used to assess reachability. On the control side of the emergent system, reinforcement learning methods provision state-driven actions. The emergent system must be capable of building intelligence and storing this into decentralized information stores.

3.3 OVERMESH: AN EXPERIMENTAL DCS PLATFORM

The OverMesh[8] architecture [14] can be applied to a variety of wireless networks. We chose to realize OverMesh on IEEE 802.11s. The PlanetLab [72], [82] service architecture was customized and integrated with the WLAN mesh network to manage DVM-based[9] overlays. Figure 14 illustrates the conceptual deployment of OverMesh supporting a wireless mesh-based DVM system. Overlays and virtualization facilitate deployment of large distributed services and peer-to-peer applications on the Internet,

[8] OverMesh is a project developed in joint research collaboration with Intel colleagues[14]

[9] DVM here is based on the PlanetLab implementation of process-level virtual machines.

but when applying them to resource constrained, mobile and wireless environments, we pursued investigation of the following issues:

- *How do we realize a testbed that can be used for development creation of novel decentralized services and applications on the wireless network?* The testbed should provide an open DCS platform for research on real wireless mesh networks. The virtualized overlay enables concurrent but separated experiments on the same physical testbed instance.
- *How does one develop and scale decentralized applications and services on wireless mesh networks?* In addition to the large number of services already available in the wired Internet, there will be many novel services and applications specific to the next generation wireless networks, including mobile applications and services monitoring, mobile node locality and real-time voice and media applications.
- *How does one efficiently manage and control the limited resources in wireless mesh networks?* While traditional overlay for wired networks sought to make the underlying network transparent to the users, this may not be desirable for wireless mesh networks. Given limited bandwidth, computation capacity, and dynamic topology in wireless mesh networks, the resource management and control should consider the state of the underlying network. The proposed virtual overlay structure provides a novel way to support distributed resource management and control - each node contributes one of its virtual machine (services) to form a resource management overlay. Virtual machine services in this overlay can coordinate to balance resources in a fully distributed fashion. The difference of this overlay from other overlays is that it can collect information across the network and across the layers of the network stack, showing the importance of information exchange between different abstraction layers.

The OverMesh platform provided a unique testbed for developing a variety of decentralized services and applications on wireless mesh networks.

3.3.1 IMPLEMENTATION

While PlanetLab targeted large distributed overlay networks supported by dedicated servers on the Internet, we focused on realizing the virtualized overlay on mobile PCs in a wireless mesh network. By participating in the research activity on private PlanetLab [72], [82], [94], we redesigned the existing PlanetLab distributed services to operate in a private WLAN mesh network.

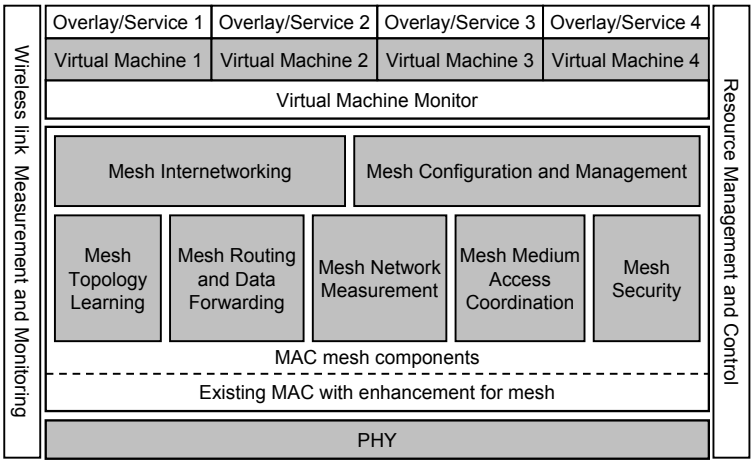


Figure 14: OverMesh platform

Figure 14 shows the current system stack of a mesh node. In addition to the mesh networking components residing in data link layer, the virtual machines are provisioned and managed by a virtual machine monitor (VMM). To enable efficient use of limited resources in the underlying wireless mesh networks, we employed several cross-overlay DVM services to interface state of underlying networks to the upper layers for resource management and control purposes. These included the following DVM-based services:

- *distributed search service* - provides a common lookup service to various applications such as information queries and distributed file storage and sharing. The distributed hash table (OpenDHT, [95]) is one of the more efficient overlay search algorithms. Each overlay node maintains a small overlay routing table for

finding the destination with the shortest path length of complexity *O(logn),* where *n* is the network size. We demonstrated an overlay search algorithm in OverMesh achieving the complexity of *O(n)* by taking advantage of network layer broadcast to route overlay search requests and using cross-layer services to facilitate vertical cooperation between the network layer and the overlay.

- *network monitoring service* - given the resource constraints in underlying wireless networks, many services and applications supported by an overlay should be made more aware of current network conditions via cross-layer information exchange. Instead of conducting cross-layer operations in every node, a dedicated overlay on top of a subset of nodes can monitor underlying network information (e.g., link throughput, packet error rate, query response time, transmission delays, RX signal strength, SINR, retransmission rates) and provide the information to all nodes and other upper-layer overlays. The monitored information can be queried by other overlays or applications using the distributed searching service.
- *location service* - positions of mobile nodes in OverMesh can be calculated by a positioning overlay. Positioning hardware can be used to find node location in real-time. The node will measure and record the distance to its neighbors periodically. This can be based on the link quality or by any other ranging techniques. Comparing the actual distance with the estimated distance can refine the distance estimation between two nodes in the positioning overlay, as their actual locations are known. A node A that is not in the overlay requires the help of peer nodes in the overlay to find its position.

3.3.1.1 OVERMESH ENVIRONMENT

The OverMesh deployment environment, as illustrated in Figure 15, consists of one central server for managing nodes and slices, internetworking (multi-tenant) hosts for the virtual machines, and clients that use the OverMesh. The following briefly outlines the installation procedures:

1. Install Fedora Core 4 OS
2. Install Flexmesh 802.11s Mesh SW
3. Configure the DHCP server
4. Install and configure MyPLC service
5. Register OverMesh clients at the administration website
6. Create Boot CD for OverMesh clients

OverMesh Central Server

Although a server typically hosts a private PlanetLab Central service, we used a mobile PC as the Central Server in order to facilitate wireless communication with other

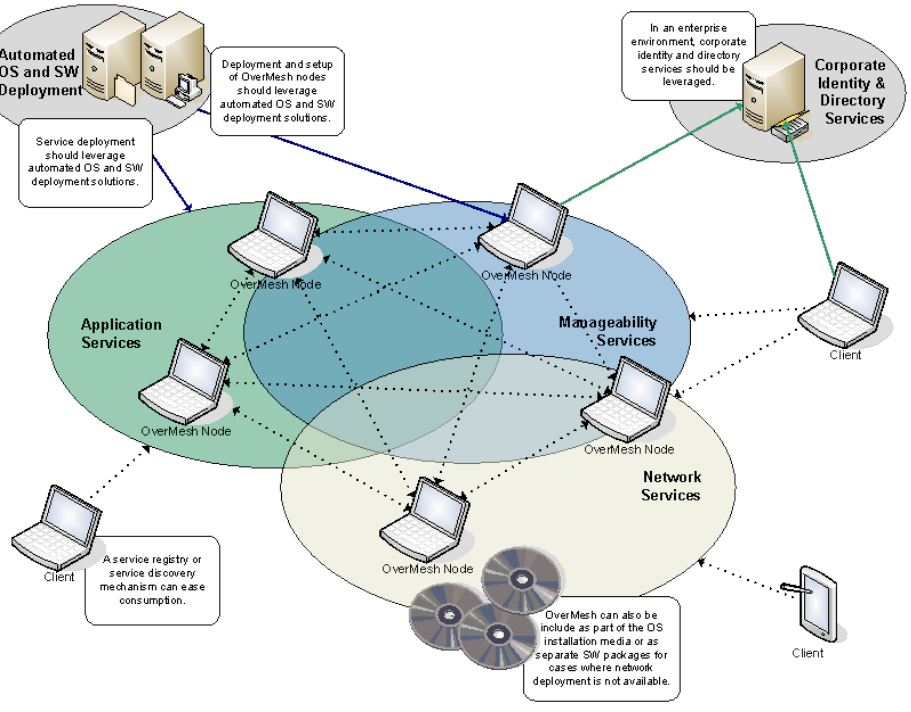


Figure 15: OverMesh deployment

OverMesh nodes. The mobile Central Server node supported IEEE 802.11g and connected to the external Internet through a wired Ethernet to facilitate installation packages from the external PlanetLab server and to provided Internet gateway service access for OverMesh nodes. The Central Server maintained central administration for authorization, installation and remote monitoring of OverMesh nodes.

OverMesh Client and Internetworking Nodes

We used mobile PCs as OverMesh nodes. The following deployment procedures were employed on internetworking nodes:

1. Install using CD created from the OverMesh Central Server
2. Configure DVM slices through the administration website
3. Deploy virtual mobile services into the DVM slices

As a new OverMesh node boots, it temporarily installs a lightweight Linux kernel then installs the wireless mesh service. The CD contains a specific key file generated by the Central for each new node. Once a new OverMesh node is booted, it contacts the Central to verify the key. If authorized, the OverMesh node will continue download of the installation kernel from Central. In this manner, the Central can always upgrade the installation kernel. We added wireless mesh networking support to boot the CD ISO and the Linux kernel installation from Central so that any new OverMesh node can communicate with the Central through the OverMesh network.

OverMesh Clients

A client machine only needs to install the mesh network service to communicate to any other OverMesh node in the system and follow the ensuing procedures:

1. Install Fedora Core 4 OS
2. Install Flexmesh 802.11s Mesh SW
3. Configure DHCP on the client
4. Consume OverMesh services

OverMesh Services

When a user deploys a new service on the OverMesh network, secure administration is handled via the OverMesh website on the Central. Since the Central has both wired and wireless connections, the user can facilitate administration from the external Internet, a mesh internetworking node or client on the local network; adding or deleting mesh nodes on the provisioned overlay. Every participating OverMesh node will be notified and configured to provision specific virtual machine services. The user can login to every deployed virtual machine and install their particular service or application slice, isolated from other services running on the physical machine.

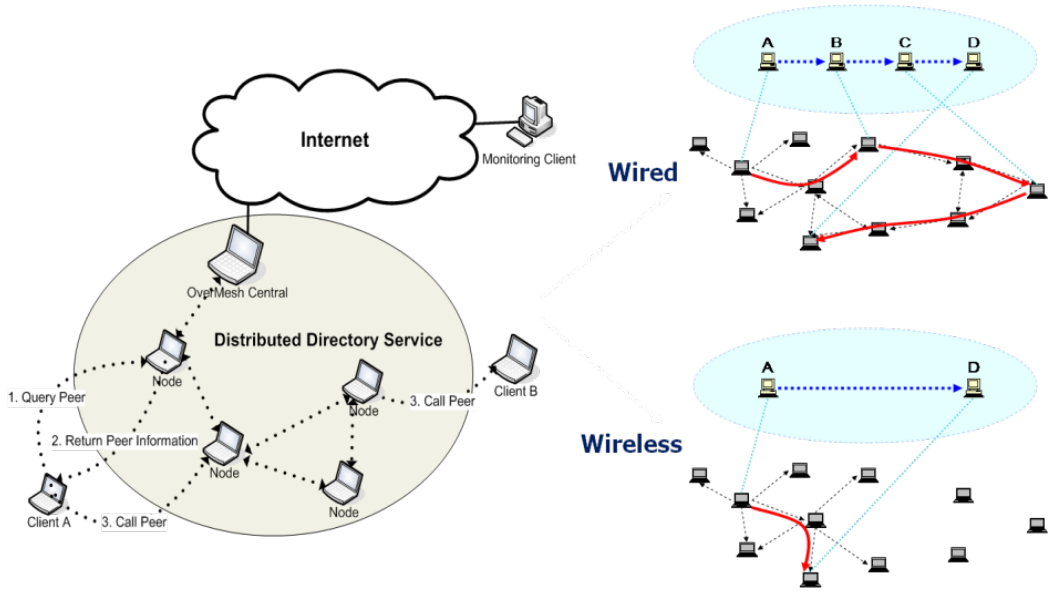


Figure 16: Distributed directory search service for P2P VoIP

Figure 16 shows an example of distributed directory service and a voice over IP application running on the OverMesh platform. A monitoring client located anywhere in the Internet can watch the real-time link quality collected from the OverMesh network. There are four mesh nodes participating in the distributed directory service overlay. The service is based on the OpenDHT, which uses a distributed hash table to facilitate the overlay-based searching algorithm. Two clients A and B are connected to the mesh network through wireless links. They store their network address provided by

OpenDHT. Client A initiates a query for the network address of Client B. The requested information is returned by the directory service overlay. Client A then starts a voice application, (GenomeMeeting) by calling Client B's address.

3.3.1.2 VALIDATION

We conducted various experiments of the OverMesh platform on a distributed (office) deployment as depicted in Figure 17. Real-time wireless link quality measures were collected from monitoring services running on each of the mobile clients. We present some of the results in this section and further results including MATLAB simulations and the platform SDK and installation procedures for OverMesh in the Appendix.

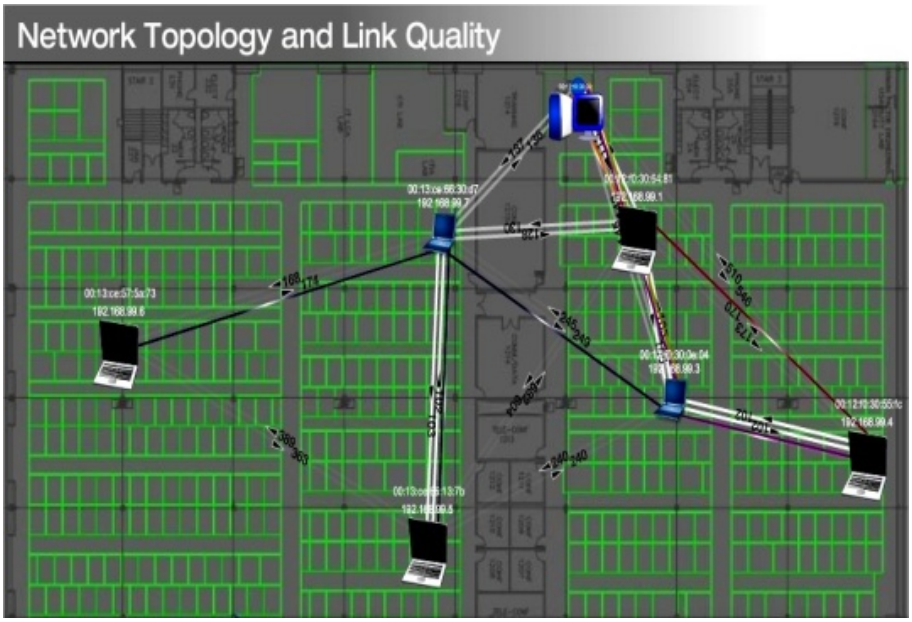
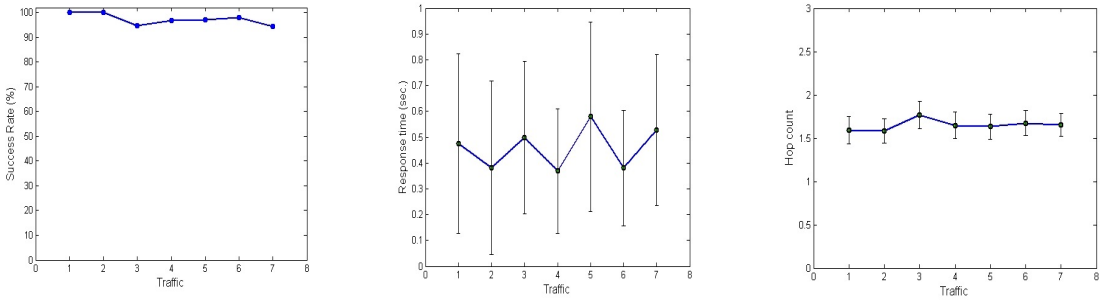


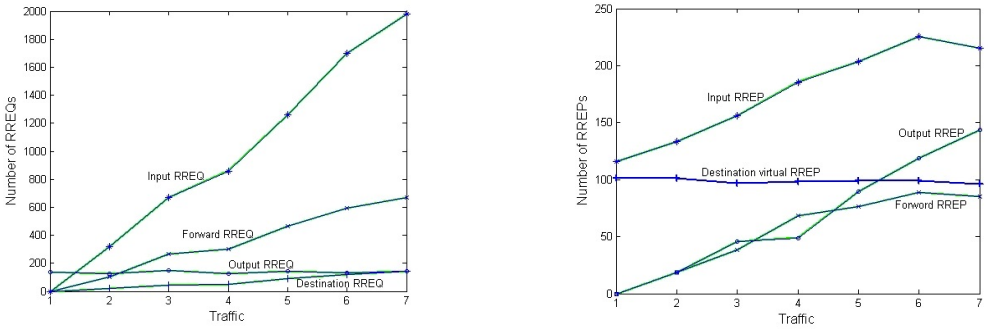
Figure 17: OverMesh topology

Our first experiment focused on the impact of traffic load with a range of concurrent traffic workloads. For each of the workloads, search requests are sent at a constant rate of three requests per sec. Figure 18 illustrates results for search success rates, hop counts and response times at 95% confidence intervals. As shown, the success ratio drops slightly when load and correspondingly packet collisions increase. However, hop count is not affected by load due to the fixed network topology. Given the limited hop

count and reduced retransmissions in such a small network, there is no noticeable impact in response time of successfully transmitted packets. Also in Figure 18, we depict the number of network routing request packets (RREQ) and reply packets (RREP). For both control packet types, the mean number of output, input, forward, and destination packets is displayed. In this scenario, since there are no other traffic types other than the (cross-layer) overlay search packets, the total number packets at the network layer are equal to those introduced by the overlay.



(a) Success ratio *(b) Response time* *(c) Hop count*



(d) Number of RREQ packets *(e) Number of RREP packet*

Figure 18: Load-based results

3.4 REALIZATION CHALLENGES

The initial aims for the OverMesh experimental platform were to demonstrate rapid deployment of network services and applications in an open, unstructured and parallel manner. The scaling properties at the physical mesh network are balanced with service

assurance and resource efficiency achieved through virtual machines cooperating across layered, virtual-physical boundaries. In this section, the primary challenges associated with DCS are presented. The specific challenges that this thesis will deliver technical contributions include building scalable, dynamic structures capable of adapting to dynamic topologies and delivering emergent management solutions to address dynamic operating conditions in resource constrained DCS networks.

Utilizing hierarchically organized virtual clustering techniques, we enable different models of cooperation and structure, including centralized, peered or hybrid structures in tiered fashion. By observing and capturing static and dynamic properties of the system at different levels of the hierarchy, we allow the system to discover and self-organize its own optimal structural organization. Thus, to compete with a demand-volatile decentralized system, the proposed HVC structure is designed to counter-act this volatility through operationally driven structure and distributed cooperation.

Further, we examine novel models of state and policy reconciliation, whereby emergence is introduced in hierarchical fashion using HVC to create multiple tiers of local and global separation for state and policy aggregation, distribution and decision-making. Emergence [18] is the architectural behavioral objective, which HVC strives to introduce into the DCS. In this work, the emergence objective extends beyond a two-tier definition (i.e., local micro behavior and global macro behavior), which is common in natural or biological systems. The intent here is to promote local and global autonomy on state management and policy-based influence or control within and across the virtual cluster virtual hierarchy. The HVC organizational model is structurally consistent with the approach in [179]. We embed self-managing techniques using statistical or learning-based methods across tiers of the HVC hierarchy to facilitate state management and policy dissemination. Unlike the related work in these areas, the virtual clustering strategy assists in the complexity challenges associated with data dissemination and belief state synchronization that confront many of the node

cooperative processing approaches, when they are applied to larger network complexities due to propagation, distributed synchronization or coordination issues.

Chapter 4

4 HIERARCHICAL VIRTUAL CLUSTERING

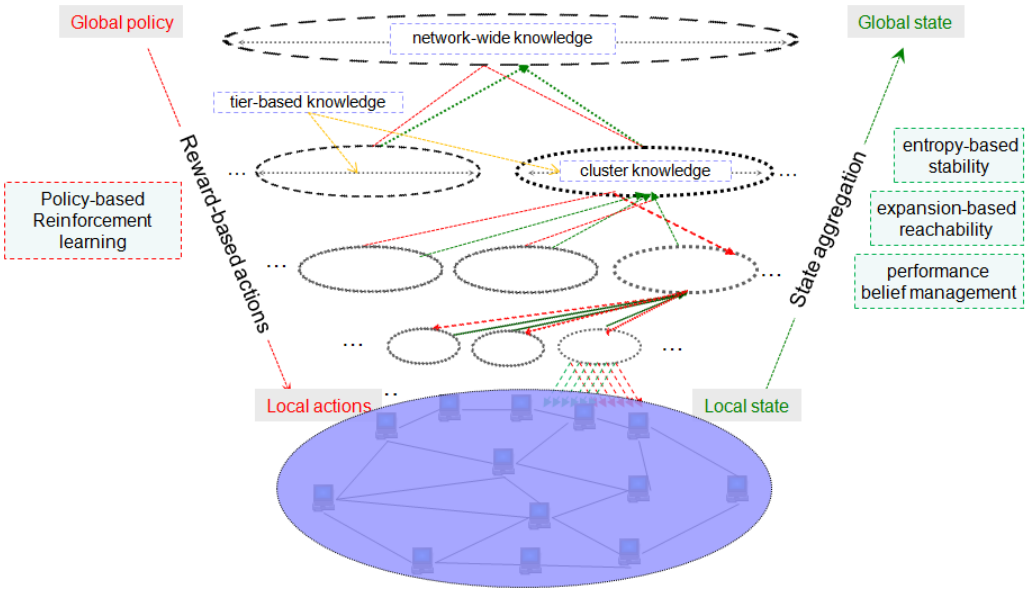


Figure 19: Hierarchical virtual clustering

4.1 OVERVIEW

A key characteristic of a decentralized communication system (DCS) is the emergence property – the separation of local behavior from global behavior enabling novelty in their respective control and management dynamics. This independence is built via structure, policy decisions and state representation. To achieve this, hierarchical virtual clustering is manages DCS organizational structure. Virtual clusters form and aggregate at multiple levels of a logical hierarchy as shown in Figure 19. As depicted, the physical (mesh) nodes are at the bottom of the DCS hierarchy and do not (on a physical level) participate in the virtual clustering schematic. Logical clustering is formed using virtual machines to instantiate overlay or virtualized internetworking structures. To balance between the merits of a peer-to-peer organizational strategy and a purely centralized organizational strategy, clustering [184], [186], [189] enables

organizational choice or flexibility. By organizing state management and control policies over hierarchies of clusters, one can achieve the merits of peer-to-peer at the lower portion of the hierarchy and individual clusters, and more centralized dynamics towards the root of the hierarchy. Moreover, through this separation the emergent (i.e., novel global-local separation) properties are created at each level of the hierarchy and behaviors can be consolidated to respective clusters, albeit influenced by peered, global[10] (parent) and local (child) clusters. Furthermore, management and control overhead in DCS are localized without compromising the benefits of centralized or global awareness and control. A promoted clusterhead will act as a virtual access point to facilitate central management coordination within a cluster.

4.2 CLUSTER EMERGENT ARCHITECTURE

Since there exists multiple, logical levels of clustering, the higher-level clusters and clusterheads will be formed of underlying clusterheads, not physical nodes. This is a main distinction of this work - clustering in DCS is applied to facilitate network control and management, not to handle transport or data plane functionality[11]. As such, any node, which acts as clusterhead, can operate in multiple virtual functional modes in the hierarchy as a) basic cluster node, b) global clusterhead or c) local clusterhead. These nodes may be designed to support common services or multiple, redundant services operating over levels of clustering functionality. The design choice is dependent on the desired service reusability and the resource capabilities of the DCS node.

We show the design of a single cluster service, implemented as a clusterhead function in Figure 20 with the functional elements associated with cluster control and management system represented. As illustrated, a cluster interfaces with local, global

---

[10] Throughout Chapter 4 and Chapter 5, the use of global or local terms is used to represent relative cluster parent or child relative associations and as such the terms may be used interchangeably. Given the multi-tier nature of HVC, the use of the global and local terms is more appropriate to convey a multi-level emergence framework.

[11] This does not preclude the use of virtual clusters for transport or data plane coordination in enabling dynamic topology structures.

and peer clusters. The functional elements include a *performance* service, which manages performance state applied to optimization problems within a cluster and globally across clusters. Similarly, a *stabilization* service manages network availability state through entropy evaluation following the stability dynamics of the DCS network within and beyond a cluster. A *reachability* service exists to manage connectivity and expansion properties of the cluster graph. Finally, a *policy-based reinforcement* (learning) service is utilized to direct multiple control policies associated with the respective state management threads downward from the root of the tree towards the lower clusters of the DCS system.

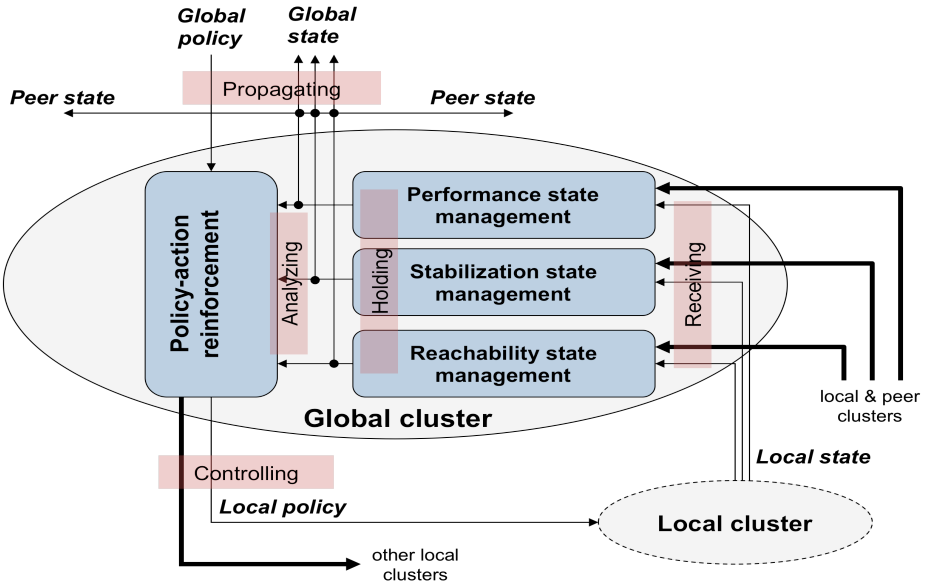


Figure 20: Cluster emergent service framework

As depicted in Figure 20, the global and local clusters have a similar alignment to a parent-child relationship; unlike this model, however, the clusters operate in an independent manner, receiving policy-reinforced[12] guidance from the global cluster. As the global cluster will serve multiple independent local clusters, its external behavior should exhibit novelty in its self-managed internal dynamics from the aggregated (from the local and peer clusters) state management process. Recursively, global (parent)-to-

[12] There are multiple policy threads operating with the respective state management services.

local (child) inter-cluster dynamics maintain behavioral novelty or managed state and policy control across the emergent hierarchy.

4.2.1 CLUSTERING MANAGER

The Clustering Manager (CM) is a distributed service managing the depth and breadth of the emergent cluster hierarchy. Specific functions of cluster addressing, clustering operations and clusterhead selections are managed through the CM. A lightweight service function facilitates the life cycle of the logical clustering hierarchy supporting the underlying domain of DCS nodes. This includes group coordination of nodes through cluster addressing facilitation, inter-cluster node movement or the creation and revision of clusters as needed. Alternatively, state management and policy-based control functions, as described in Chapter 5, are managed separately from the clustering management service. Thus, similar to common Internet services (e.g., DNS), the CM serves primarily to coordinate the hierarchical organization of the clustering operations and management.

4.2.2 PROMOTIONAL METHODOLOGY

The selection of clusterheads will be based on operational superiority or rank. Similar to military or corporate rank, the basis of higher ranked clusterhead nodes is established by the ability of nodes to be highly networked, highly reliable, stable and superior in performance. In other words, nodes that exhibit higher reachability, stability and performance efficiency are promoted to clusterheads. This self-organized positioning is not unlike the organization of network nodes in a traditional hierarchical telecommunications network, where nodes at the core of the network must exhibit high availability (e.g., five nines) and demonstrate wide reachability and very low latency. Thus, it is anticipated that DCS reachability state, stability state and performance belief state will be the sequential basis for determining operational rank and superiority. In other words, nodes must demonstrate their graph connectivity with sufficient capability (e.g., broad reachability and resiliency), then must exhibit stability (e.g., node reliability

or persistence for routing and communications), and finally, performance quality (e.g., load or latency optimization).

As one progresses up the cluster hierarchy, temporal and spatial aggregation will take place; changing the selection of higher-level clusterheads, differentiating nodes with operational longevity and more abstract state management and policy control. To use the corporate or military analogy, the experienced or 'big picture' leaders are more likely to be promoted to the higher ranks of leadership and change influence in the organization. For each of the clusters, head selection must pass several cyclic iterations up the hierarchy prior to selection of the optimal clusterhead. The selection of the higher-level clusterheads will follow the selection of the lower level heads. Clusterheads could be selected in a less-efficient manner (e.g., cluster ID, exponential averaging) during preamble periods until steady state is reached and eventual stability and operational selections are persisted. Clusterhead nodes are 'good citizens', establishing their long-standing reputation through operational reliance and trusted communications.

4.2.3 CLUSTER COMMUNICATIONS

A single cluster addressing scheme will exist across the cluster hierarchy. At the lowest level of the hierarchy, the physical DCS network will be labeled as cluster *level n, cluster* $C_{(n,0)}$, where $0 <= n < y$, and y is the number of levels in the emergent hierarchy. For the purposes of this work, the lowest level of the tree does participate in the clustering management activities. However, for completeness, network addressing is used as an example, for service broadcasting purposes. As DCS nodes are virtually partitioned and clustered across cluster level n, the next level (up) will be cluster level $n-1$; and depending on the number clusters at level $n-1$, a $n-1$ level cluster is then assigned a cluster *domain* m, where $0 <= m < y$, and y equals the number clusters at this hierarchy level. Thus, we label any cluster in the hierarchy $C_{(n,m)}$, and continue along this progression to the eventual highest or root cluster, $C_{(0,0)}$. The continued segregation of cluster level partitioning (and addressing) may not

necessarily follow or reflect a symmetrical tree hierarchy; that is, each cluster level can converge to a number of clusters less than the number of clusters below its level, depending on the system dynamics of the environment. Furthermore, mesh nodes (at the lowest level n) or clusterheads (at levels < n) assigned to a particular cluster will use $C_{(n,\ m)}$ to communicate with nodes (or clusterheads) in this particular cluster, or specifically, the clusterhead assigned to manage the $C_{(n,\ m)}$.

Cluster nodes or clusterheads associated with a global (parent) cluster (see Figure 20) can use the cluster address for sinking and responding to cluster operational communications, but only the global clusterhead can utilize the cluster address for sourcing operational communications or clustering control or management. In summary, only global clusterheads can participate in inter-cluster and intra-cluster communications and can directly influence the global behavior of the cluster, peers or local (child) clusters. Other nodes or clusterheads (associated to this global cluster) can influence indirectly, but only via their intra-cluster (local) participation as members of the global cluster.

The addressing scheme supports multicasting across clusterheads, where $C_{(n,-)}$ communicates to all peer clusterheads at level n. Furthermore, clusterheads are the only nodes in a cluster that are aware of the hierarchy lineage, and therefore communicate via their lineage and peer clusterheads. Therefore, they are not aware of the broader set of clusters outside their level and lineage. Alternatively, a clusterhead, which operationally dominates emergent hierarchy up to the root cluster $C_{(0,\ 0)}$, has the ability to communicate and access state or policy across all clusters in the hierarchy. Thus, there is clear advantage and incentive for DCS nodes to strive for operational superiority to increase their hierarchical positioning and reputation over their domain lifetime.

Inter-cluster and intra-cluster communications (i.e., cluster-specific control and management messaging) occur according to specific addresses assigned to clusterheads as described above. In general, operational messaging supports state and

policy-based communications and occurs intra-cluster and inter-cluster in hierarchical or peer form based on the clustering structure. Control (policy) and management (state) messages are asynchronous and support cooperative aggregation and peering algorithms as discussed in the upcoming sections.

4.2.4 CLUSTERING OPERATIONS

The main functions of the CM are to facilitate group clustering structures based on the topology dynamics of DCS nodes, aggregating clusters in meaningful hierarchical formations and handling the creation and revision of clusters throughout the operating lifetime of the connected domain of DCS nodes.

As stated earlier, the objectives of the clusters are to facilitate network control and management through a balance of peer-to-peer and centralized organizational control. The state management functions include stability, performance optimization and reachability state management as depicted earlier in Figure 20 within each cluster. Clusterhead nodes receive cluster state conditions and perform statistical aggregation within their respective clusters or cluster nodes. Clusterheads perform cluster state learning or analysis and hold cluster knowledge for state-level assimilation with peer clusters. Aggregated or learned state is propagated to global clusterheads, or acted upon though the integration of local or global policy using policy-based reinforcement to control or influence the local cluster behavior. The cluster-based, emergent flow methodology is depicted in Figure 21.

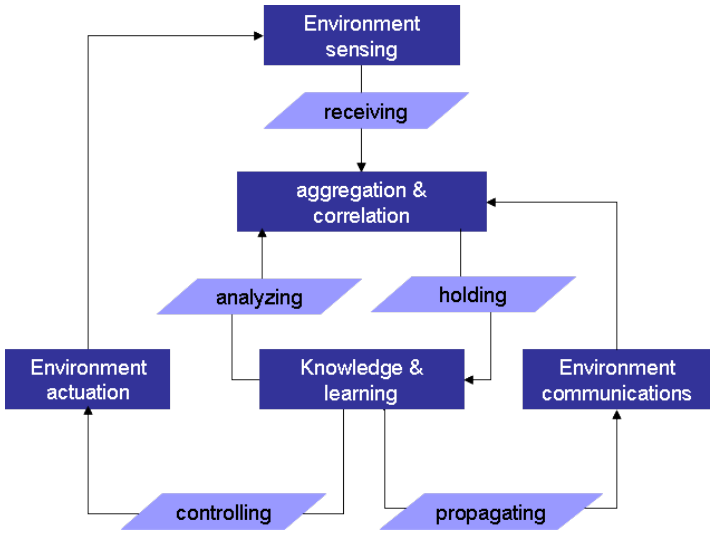


Figure 21: Emergent flow methodology

In what follows, the above methodology is presented through decomposition of the emergent services for state management (sensing) and policy management (control), which are also performed asynchronously and hierarchically. At each level of the clustered hierarchy, control and management behavior novelty is exhibited independently between cluster levels[13] with temporal and spatial aggregation occurring as illustrated in Figure 22. Spatial aggregation accounts for the multiple local clusters affiliated with the global cluster, while temporal aggregation accounts for the emergent cycle periods that clusters follow. However, it is not a requirement that measurement states be necessarily aggregated in a statistical sense, but rather that the global cluster consider inputs from all of their respective child clusters, in addition to their completing at least one period of operational state management. Thus as cluster aggregation occurs, slower state changing effects can be seen in clusters at the higher end of the cluster hierarchy, mimicking behavior normally observed through human experience in knowledge or decision making in people-centric organizations.

[13] The term 'level' reflects the hierarchical depth, rather than the breadth of the hierarchy; meaning that the emergent process or novelty of behavior is consistent with the global-local cluster relationship, rather than a peering relationship, that should be evident across clusters within the same level of the hierarchy.

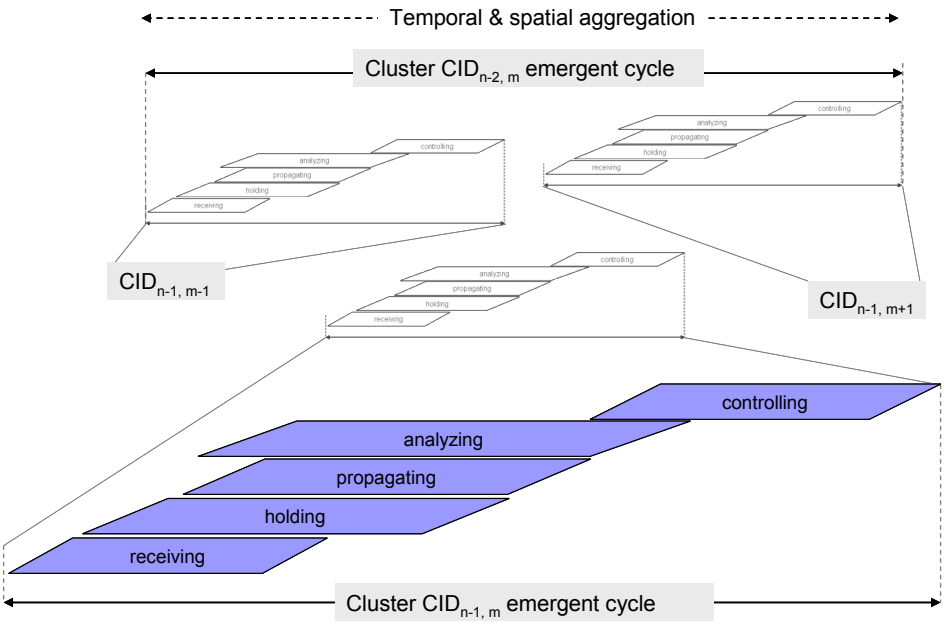


Figure 22: Clustering aggregation cycle

4.3 IMPLEMENTATION

4.3.1 OBJECTIVES

The OverMesh implementation provided an experimental research platform and deployment environment. As demonstrated in Chapter 3, we successfully implemented a working DCS experimental platform and testbed, demonstrated the use of static virtual machines for distributed services creation and performed initial investigation on DCS resource management. While the feasibility of such a flexible architecture is realizable in small scale, the more challenging aspects of DCS associated with infrastructure scalability and operational effectiveness will be addressed through modeling implementation. We model DCS and evaluate the following high-level research goals for HVC as described in this chapter and emergent management and control services in Chapter 5:

1. Characterize HVC structural behavior and scalability using principles of operational superiority in creating cluster formations and promoting node hierarchy;
2. Demonstrate HVC operational benefits in dealing with dynamic node conditions and structural perturbation versus traditional node and physical network structural models;
3. Characterize HVC structural effectiveness in supporting the emergent management service objectives against traditional network structures as the comparison baseline;
4. Demonstrate emergent management effectiveness in performing network operational management and control in DCS.

4.3.2 MODELING FRAMEWORK

Figure 23 illustrates the DCS modeling framework supporting HVC and management component services and delivering the associated results in Chapter 4 and 5. Modeling these capabilities is fundamental to demonstrating network-wide scalability

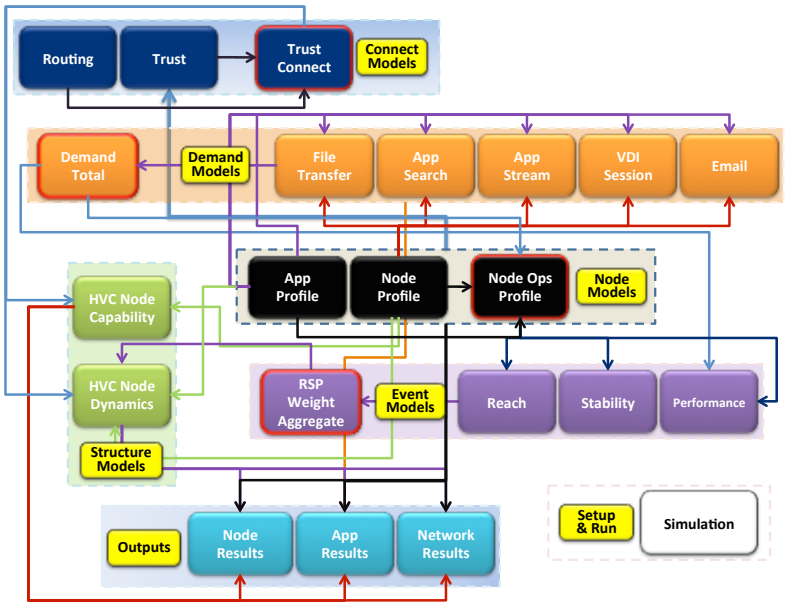


Figure 23: Modeling framework

and the emergent properties of DCS. As highlighted in Chapter 3, key architectural requirements of DCS are flexible node composition and operational predictability. A node's resource and functional composition is made adaptive through virtual machine augmentation, while a node's (virtual) hierarchical rank reflects its operational resilience and superiority. This is not unlike today's physical models, where computationally rich server nodes or highly available router nodes are physically placed in strategically central points in the infrastructure. To this end, what HVC is fundamentally shifting is the physical requirements for centralization, not necessarily the need or value of centralization. Centralization is an innate objective of a dynamic and HVC infrastructure facilitated through virtualization and operational superiority. Operational superiority is enabled through state management aggregation and the emergence properties of DCS.

4.3.2.1 SIMULATION SETUP & INVOCATION

The simulation models supporting the HVC and emergent management architectural designs in Chapter 4 and 5 are illustrated and summarized in this section. As shown in Figure 24, simulation runs can be invoked or reset as required.

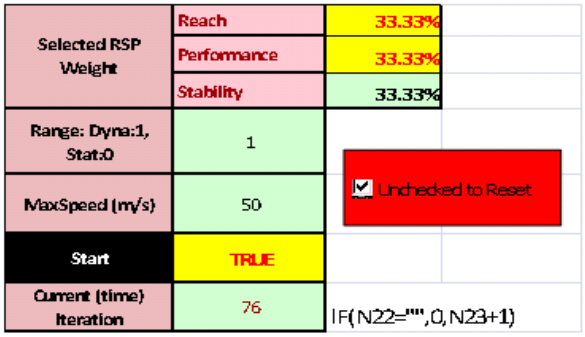


Figure 24: Simulation setup & invocation

As depicted, operational variables for HVC network structures (only) are configurable with appropriate weights across reachability, stability and performance as deemed necessary for the modeling run. Learning rates (γ) are also applicable to only HVC configurations and are described in Chapter 5 supporting DCS policy-

based reinforcement learning. Additionally, node radios can be dynamic or static range configurable as is maximum node mobility speed through each simulation run. All other parameters remain consistent across alternative HVC or non-HVC structures, maintaining comparative parity by reducing structural or operational bias. Modeling results are used for comparison purposes with traditional networking structures. We capture and aggregate various measures including messaging overhead, utilization, overall node rank and tier-based scores, highest tier, direct and cluster neighbors, usage demand and unavailability for each node for alternative reporting purposes.

4.3.2.2 TOPOLOGY & CONNECTIVITY MODELS

Associated with the emergent DCS environment, a common topology framework formed of multi-radio mesh networking and virtual machine computing is implemented in MATLAB and Microsoft Excel. As shown in Figure 25, we evaluate the operational range of mobile nodes in either Wi-Fi (802.11a) or WiMAX (802.15e). Following the legend descriptions, we model node connectivity, dependability and performance based on speed, availability and service demands, respectively. A two-dimensional model (i.e., Random Way Point mobility model) of mobile connectivity evaluates physical mesh mobility in an OverMesh implementation as illustrated in Figure 25. The MATLAB plot is an illustration of the modeled DCS environment. However, a simpler mobile connectivity model is sufficient for our evaluation objectives and thus, implemented and evaluated in Excel-based analytical modeling form. Essential behavioral properties that were model preserved included randomized node mobility speed, Wi-Fi, and WiMAX fixed wireless radio transmission and reception range once during each simulation interval.

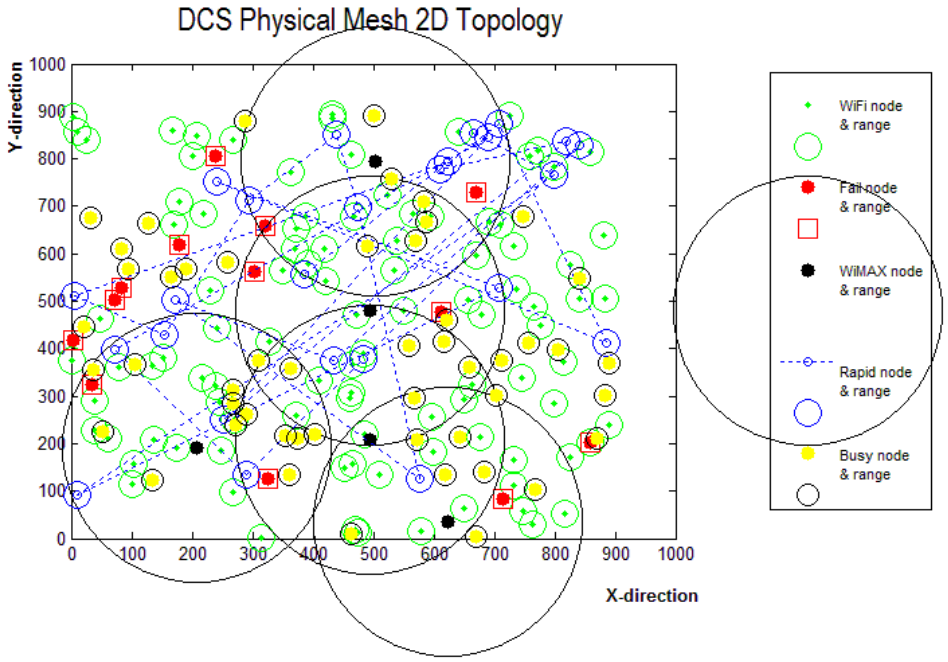


Figure 25: Mobile-to-mobile 2D connectivity plot

At each simulation interval, nodes calculate peer connectivity based on an X-Y position coordinate system with other mobile peers. The topology & connectivity subsystem models wireless (802.11a, 802.15e) connectivity between mobile nodes in adjacency matrix form, compensating for node transmission range and mobility speed. Operational issues such as node failure rate or high utilization characterize inter-connectivity stability or performance, respectively.

4.3.2.3 NODE CAPABILITY & EVENT MODELS

To clarify an important DCS distinction from traditional node or infrastructure models, Table 2 lists traditional node resource capabilities along with extended node DCS capabilities enabled by multi-radio wireless and virtual server capabilities. To properly model DCS infrastructure, we characterize the node's resource capabilities and its adaptation capabilities enabled through virtualization and multi-radio wireless. Each DCS node has associated applications that it either consumes or serves to other DCS nodes. Based on a node's server or virtual machine capabilities and application

support, this determines its (modeled) distributed potential or infrastructure capability.

Table 2: Node profiles

Node ID	VM-enabled	Server or WiMAX capable	Base CPU (IPS) Performance	CPU Count	Base Disk (bPS) Performance	Disk Count	Base WLAN (bPS) Performance	# Radios	Normalized Node Comm Capability	Normalized Node Compute Capability	Client=1 Server=0	Wireless Range	Search	DVC Stream	Load Balance	VDI	Email
1	1	1	3.00.E+07	4	2.40E+08	2	5.40E+07	2	7.00E+07	1.20.E+08	0	271	1	1	1	1	1
2	1	0	2.00.E+07	1	5.60E+07	1	5.40E+07	2	7.00E+07	4.65.E+07	1	50	1	0	0	1	1
3	0	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	1	1	0	1	1
4	0	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	1	0	1	1	1
5	1	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	1	1	1	0	1
6	1	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	0	0	0	1	0
7	0	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	0	1	0	1	0
8	0	0	2.00.E+07	1	5.60E+07	2	5.40E+07	2	7.00E+07	2.68.E+07	1	50	1	0	1	1	1
9	0	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	1	1	1	0	1
10	0	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	1	0	0	1	1
11	1	1	3.00.E+07	4	2.40E+08	2	5.40E+07	2	7.00E+07	1.20.E+08	0	339	1	1	1	1	1
12	1	1	3.00.E+07	4	2.40E+08	2	5.40E+07	2	7.00E+07	1.20.E+08	0	438	1	1	1	1	1
13	0	0	2.00.E+07	2	5.60E+07	1	5.40E+07	2	7.00E+07	4.93.E+07	1	50	0	1	1	0	0
14	0	0	2.00.E+07	1	5.60E+07	2	5.40E+07	1	5.40E+07	2.68.E+07	1	50	0	0	0	1	0
15	1	0	2.00.E+07	1	5.60E+07	1	5.40E+07	1	5.40E+07	4.65.E+07	1	50	1	1	0	1	1

We model the node subsystem based on the nodes' networking and computational (CPU, storage) resourcing capabilities, virtual machine capabilities and application hosting capabilities to serve demand to mobile clients. Additionally, each node's operational behavior including speed, failure rates, recursive internetworking and compute performance loadings are also captured. DCS node profiles are illustrated in tabular form in Table 3. Node capabilities are a function of their (0 or 1 randomized) ability to support virtual machines (VM), support for server or WiMAX functionality based on structural configuration type and modulo function of network id, and the total number of DCS nodes and servers within the structural configuration type. The number of configured radios supported by any node is randomized to Wi-Fi (single-radio) only or both (multi-radio) Wi-Fi and WiMAX. To ensure simulation parity in comparing alternative structures, the total number of servers supported in each configuration is kept quantitatively the same or there is marginal disparity. Additionally, node resource capacities follow the computational and communications

capabilities and nodes having the functional ability to support alternative applications must have ‘server’ capability.

The node operational profiles represent the conditional event modeling within the DCS environment and captures operational events such as node or link failures, mobility movement or speed and load (usage) conditions. Combined node (link) failure events are based on the addition of a simple node failure probability plus range-based unavailability using the node’s speed and relative connectivity with other mobile nodes. Alternatively, node utilization reflects a maximum between the node’s computational usage and the node’s total (out, in) link utilization. Node movement or speed is calculated using a simple model, where initial or previous position coordinates are stored, and next position coordinates are recalculated (each interval) based on a random multiplication factor against the maximum possible node speed. As shown in tabular form in rolling node expansion, availability and node usage averages are interval averaged *@modulo2, @modulo8 and @modulo16* second periods.

Table 3: Node operational profiles

1	0	Reachability									Stability Rate				Performance			
Node ID	Rand()	Initial	Next	Current Speed (f/s)	Rolling Ave Speed	Node Expansion	Rolling Node Expansion	2 Sec	8 Sec	16 Sec	Rolling Node Availability	2 Sec	8 Sec	16 Sec	Rolling Node Idle	2 Sec	8 Sec	16 Sec
1	0.00895	1590	1590	0	26.21	0.30323	0.3390	0.3390	0.3469	0.3657	0.8508	0.8508	0.9400	0.9075	71.894%	0.7189	0.8955	0.8434
2	0.38638	4642	4623	19	20.87	0.00022	0.0003	0.0003	0.0004	0.0003	0.9086	0.9086	0.9417	0.9260	95.917%	0.9592	0.9648	0.9648
3	0.10122	2055	2068	13	14.61	0.02944	0.0325	0.0325	0.0275	0.0260	0.9439	0.9439	0.9474	0.9474	98.547%	0.9855	0.9975	0.9948
4	0.81875	1688	1656	32	30.95	0.12041	0.0901	0.0901	0.1210	0.1208	0.8863	0.8863	0.8947	0.8947	98.792%	0.9879	0.9976	0.9974
5	0.68433	1142	1158	16	17.61	0.03384	0.0340	0.0340	0.0392	0.0282	0.9391	0.9391	0.9429	0.9429	98.180%	0.9818	0.9973	0.9889
295	0.41922	2719	2740	21	21.51	0.01626	0.0143	0.0143	0.0199	0.0163	0.9053	0.9053	0.8798	0.9210	97.015%	0.9701	0.9975	0.9972
296	0.42081	2271	2250	21	21.99	0.07317	0.0374	0.0374	0.0603	0.0619	0.9059	0.9059	0.9478	0.9228	98.477%	0.9848	0.9545	0.9830
297	0.02678	765	764	1	22.95	0.03955	0.0925	0.0925	0.0357	0.0327	0.8961	0.8961	0.9243	0.9319	96.144%	0.9614	0.9972	0.9971
298	0.72654	1687	1642	45	41.05	0.29620	0.3075	0.3075	0.3684	0.3388	0.6987	0.6987	0.6667	0.6667	95.894%	0.9589	0.9960	0.9787
299	0.06311	1216	1233	17	18.89	0.00901	0.0279	0.0279	0.0100	0.0119	0.9109	0.9109	0.9412	0.9412	98.752%	0.9875	0.9971	0.9970
300	0.16031	1059	1068	9	26.01	0.08899	0.0463	0.0463	0.0789	0.0655	0.8840	0.8840	0.9349	0.9078	0.9824	0.9824	0.9374	0.9748

4.3.2.4 APPLICATION DEMAND MODELS

The primary objective in building out application profiles and client-server (C-S) demand models is to ensure sufficient service (e.g., demand, quality) diversity and deployment (e.g., peer-to-peer, client-server, tiered client-server) diversity of application models to compare and test the DCS infrastructure and operational range.

To ensure this objective, each application has a diverse range of transaction profiles to primary node resource subsystems as shown in Table 4. The application physical or virtual infrastructure deployment model is determined at runtime by the corresponding infrastructure being employed and evaluated. As an example case of Virtual Desktop Infrastructure (VDI) workload, the VDI server-server demand profile as shown in Table 4 may not reflect a static *physical server-to-physical server* demand transfer as in the case of a traditional client-server tiered infrastructure. Alternatively, a *virtual server–to-virtual server* demand transfer in the HVC case, where the same physical server may employ both server tiers of the application infrastructure through distinct and isolated virtual machines. A simple scenario of HVC advantages (over traditional physical infrastructure) can be shown in an example *virtual server–to-virtual server* demand transfer, where the same physical server may employ both server tiers of the application infrastructure through distinct and isolated virtual machines; thereby avoiding the extra communication hop or delay and routing complexity. This creates a unique trade-off in computational processing with communications processing. While this may be perceived to be more complex, it is actually simpler to deploy, more cost-effective and potentially more operationally robust, as we follow the operational superiority principles.

Table 4: Application Profiles

Demands	Search	VDI	Stream	FTP	Email
Data size(bytes per CPU Transaction)	800	4096	2048	10000	1024
CPU Visits(IPS)	271	1373	688	3347	343
CPU Demand (b)	217	2747	1375	2511	343
Disk Visits(R/W tps)	2	8	4	20	2
Disk Demand (bps norm)	12800	262144	65536	1600000	16384
T1-Node-Client Data visits(pps)	2	3	2	7	1
T1-Node-Client Demand (bps)	24192	36288	13056	84672	12096
Client-T1-Node Visits(pps)	2	3	2	7	1
Server-Server Sync Visits(pps)	0	2	1	0	0
Server-Server Demand (bps)	0	24192	6528	0	0

Demand profiles model client-server and server-server (node-node) demand interactions based on the application-to-node mapping and application distributed systems deployment using the predefined application profiles. The application profiles include five (5) application types including FTP, content search, application streaming, VDI sessions and email along with sizing per demand transaction visits (v_i) and service times (s_i) at CPU, disk & network for client-server and server-server interactions. We depict each of the respective application demand profiles shown in Figures 27-31 along with their aggregate demands in Figure 26. In summary, the primary objective in building out application profiles and client-server demand models is to ensure sufficient service and deployment diversity of applications to contrast infrastructure and operational range.

Demand Total	300	1	2	3	4	5	295	296	297	298	299	300	
	1	0.303372	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000187	0.000000	0.000000	0.303372
	2	0.000000	0.035064	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.035064
	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002535
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002433
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.003103
	295	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002535
	296	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002433
	297	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.003103
	298	0.000015	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.009437	0.000000	0.000000	0.009835
	299	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002702
	300	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002433
		0.306425778	0.036189435	0.002535314	0.00243291	0.00253531	0.0025865	0.00243291	0.00310258	0.011953	0.00305138	0.00243291	

Figure 26: Total demand profile

FTP	300	1	2	3	4	5	295	296	297	298	299	300	
	1	0.24976	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00121	0.00000	0.00000	0.00675
	2	0.00000	0.02870	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02870
	3	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00121
	4	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00007
	5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00157
	295	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00121
	296	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00121
	297	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00157
	298	0.00005	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00675	0.00000	0.00000	0.07425
	299	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.02870
	300	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00121
		0.251412	0.028696954	0.0012096	0.00121	0.0012096	0.00121	0.00121	0.001568	0.008026	0.00121	0.00121	

Figure 27: FTP traffic

Search (p2p)	300	1	2	3	4	5	295	296	297	298	299	300	
	1	0.000363	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	7.0583E-04
	2	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.8110E-03
	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.4560E-04
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.4560E-04
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.4560E-04
	295	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.4800E-04
	296	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.4560E-04
	297	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.4800E-04
	298	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	7.0583E-04
	299	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.6696E-04
	300	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	3.4560E-04
		0.0008227	0.0003456	0.000448	0.0003456	0.000448	0.000448	0.0003456	0.000448	0.000448	0.0003456	0.0003456	

Figure 28: Search traffic

Stream Service (log N)	300	1	2	3	4	5	295	296	297	298	299	300	
187	1	0.011800	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000187	0.000000	0.000000	3.18911E-04
62	2	0.000000	0.001239	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.23905E-03
266	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.86514E-04
269	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.89630E-05
118	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.41778E-04
183	295	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.86514E-04
113	296	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.86514E-04
248	297	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.41778E-04
13	298	0.000015	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000319	0.000000	0.000000	3.50802E-03
257	299	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.23905E-03
275	300	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.86514E-04
		0	0.00136942	0.00018651	0.00018651	0.00018651	0.00018651	0.000186514	0.00024178	0.00052005	0.00024178	0.00018651	

Figure 29: Application streaming traffic

Email (central)	300	1	2	3	4	5	295	296	297	298	299	300	
	1	0.00040	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.83336E-03
	2	0.00000	0.00186	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	3.09738E-04
	3	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.72800E-04
	4	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.72800E-04
	5	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.72800E-04
	295	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	2.24000E-04
	296	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.72800E-04
	297	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.72800E-04
	298	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00008	0.00000	0.00000	2.24000E-04
	299	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	2.24000E-04
	300	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	1.72800E-04
		0.000617	0.002084	0.000224	0.000173	0.000173	0.000173	0.000173	0.000224	0.000313	0.0001728	0.0001728	

Figure 30: Email traffic

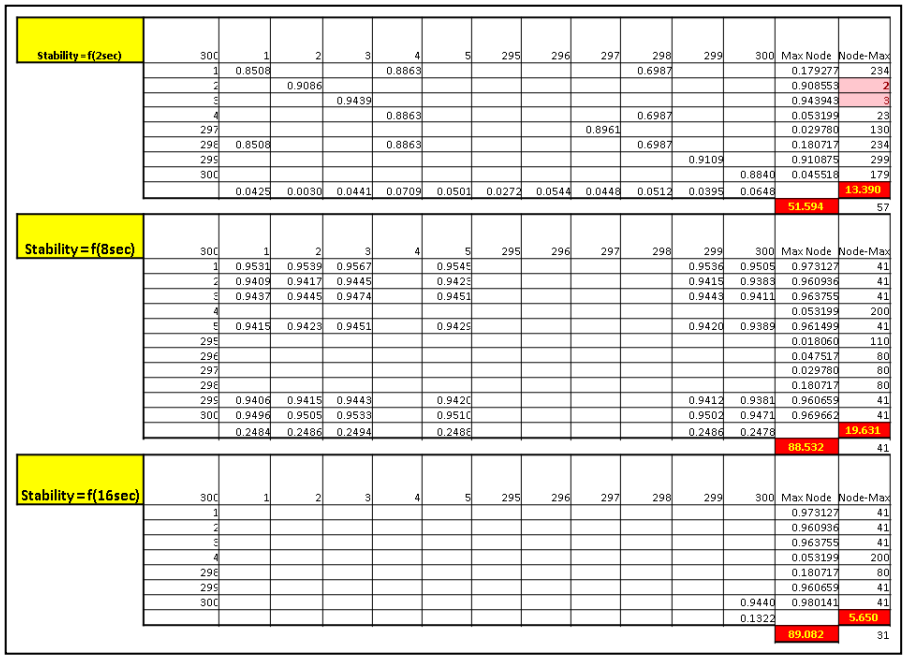
VDI	300	1	2	3	4	5	295	296	297	298	299	300	
	1	0.040250	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.18382E-03
	2	0.000000	0.004818	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.81848E-03
	3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.18400E-04
	4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	2.84444E-05
	5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	6.72000E-04
	295	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.18400E-04
	296	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.18400E-04
	297	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	6.72000E-04
	298	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.002368	0.000000	0.000000	1.30220E-02
	299	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	4.81848E-03
	300	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	5.18400E-04
		0.000000	0.005285	0.000518	0.000518	0.000518	0.000518	0.000518	0.000672	0.002750	0.000672	0.000518	

Figure 31: VDI session traffic

4.3.2.5 MANAGEMENT & CONTROL OPERATIONS

Operational events are modeled in the reachability, stability and performance state management service models and are based on application, demand and node profile models. Policy-based reinforcement learning (PRL) is used to manage and control network-wide state and configuration dynamics. The PRL subsystem component of the DCS model employs reinforcement learning techniques to drive controlled policy and structural change within the modeling framework. Each of the respective PRL processes employed for stability through availability state measures, reachability using expansion state measures, and performance through load or utilization state measures are shown respectively in Figures 32-34 with each of the corresponding PRL output tables operating (independently) *@2, @8 and @16* second intervals. As described in Chapter 5, a simple Q-learning algorithm is applied as independent PRL processes with respect to each of the state management processes. The PRL objectives are to identify and promote optimal virtual node choices for performance, reachability or stability-based policy control. Table entries are state conditioned or rewarded using updated metrics from the node profile models described in Section 4.3.2.3 for each PRL process with respective neighbor maximums selected at each of the virtual tiers after multiplication of a learning factor, *γ(0.6, 0.7, and 0.8)* at corresponding intervals. Topology connectivity profiles are accounted (i.e., multiplied by 1 or 0) for in the respective PRL processes capturing physical topology changes. The aggregate RSP[14] PRL process, assuming the RSP weights highlighted in Section 4.3.2.1 is illustrated in Figure 35.

[14] Reachability, Stability, Performance (RSP)

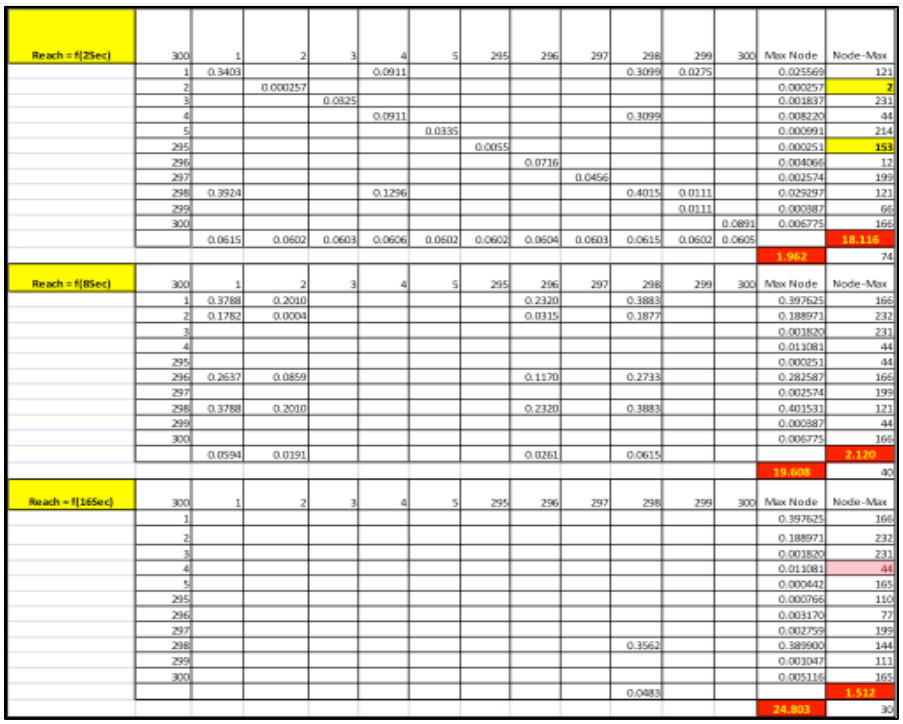


Stability = f(2sec)	300	1	2	3	4	5	295	296	297	298	299	300	Max Node	Node-Max
	1	0.8508			0.8863					0.6987			0.179277	234
	2		0.9086										0.908553	2
	3			0.9439									0.943943	3
	4				0.8863					0.6987			0.053199	23
	297								0.8961				0.029780	130
	298	0.8508			0.8863					0.6987			0.180717	234
	299										0.9109		0.910875	299
	300											0.8840	0.045518	179
		0.0425	0.0030	0.0441	0.0709	0.0501	0.0272	0.0544	0.0448	0.0512	0.0395	0.0648		13.390
													51.594	57

Stability = f(8sec)	300	1	2	3	4	5	295	296	297	298	299	300	Max Node	Node-Max
	1	0.9531	0.9539	0.9567		0.9545					0.9536	0.9505	0.973127	41
	2	0.9409	0.9417	0.9445		0.9423					0.9415	0.9383	0.960936	41
	3	0.9437	0.9445	0.9474		0.9451					0.9443	0.9411	0.963755	41
	4												0.053199	200
	5	0.9415	0.9423	0.9451		0.9429					0.9420	0.9389	0.961499	41
	295												0.018060	110
	296												0.047517	80
	297												0.029780	80
	298												0.180717	80
	299	0.9406	0.9415	0.9443		0.9420					0.9412	0.9381	0.960659	41
	300	0.9496	0.9505	0.9533		0.9510					0.9502	0.9471	0.969662	41
		0.2484	0.2486	0.2494		0.2488					0.2486	0.2478		19.631
													88.532	41

Stability = f(16sec)	300	1	2	3	4	5	295	296	297	298	299	300	Max Node	Node-Max
	1												0.973127	41
	2												0.960936	41
	3												0.963755	41
	4												0.053199	200
	296												0.180717	80
	299												0.960659	41
	300											0.9440	0.980141	41
												0.1322		5.650
													89.082	31

Figure 32: PRL-based stability model

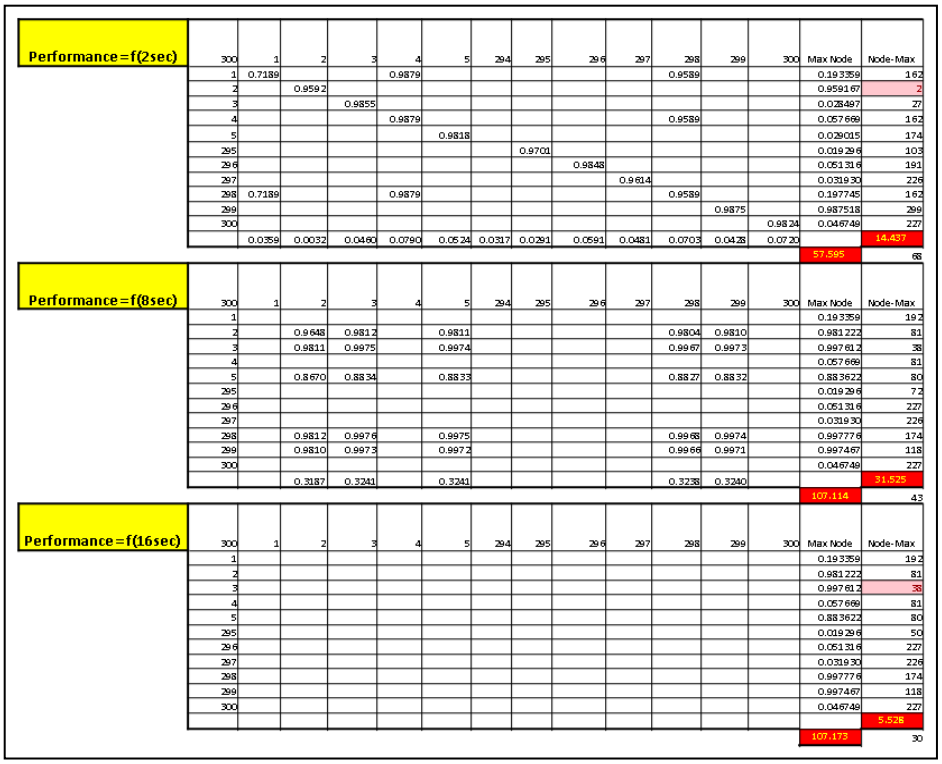


Reach = f(2Sec)	300	1	2	3	4	5	295	296	297	298	299	300	Max Node	Node-Max
	1	0.3403			0.0911					0.3099	0.0275		0.025569	121
	2		0.000257										0.000257	2
	3			0.0825									0.001837	231
	4				0.0911					0.3099			0.008220	44
	5					0.0335							0.000991	214
	295						0.0055						0.000251	153
	296							0.0716					0.004066	12
	297								0.0456				0.002574	199
	298	0.3924			0.1296					0.4015	0.0111		0.029297	121
	299										0.0111		0.000887	66
	300											0.0891	0.006775	166
		0.0615	0.0602	0.0603	0.0606	0.0602	0.0602	0.0604	0.0603	0.0615	0.0602	0.0605		18.116
													1.962	74

Reach = f(8Sec)	300	1	2	3	4	5	295	296	297	298	299	300	Max Node	Node-Max
	1	0.3788	0.2010					0.2320		0.3883			0.397625	166
	2	0.1782	0.0004					0.0315		0.1877			0.188971	232
	3												0.001820	231
	4												0.011081	44
	295												0.000251	44
	296	0.2637	0.0859					0.1170		0.2733			0.282587	166
	297												0.002574	199
	298	0.3788	0.2010					0.2320		0.3883			0.401531	121
	299												0.000887	44
	300												0.006775	166
		0.0594	0.0191					0.0261		0.0615				2.120
													19.608	40

Reach = f(16Sec)	300	1	2	3	4	5	295	296	297	298	299	300	Max Node	Node-Max
	1												0.397625	166
	2												0.188971	232
	3												0.001820	231
	4												0.011081	44
	5												0.000442	165
	295												0.000766	110
	296												0.003170	77
	297												0.002759	199
	298									0.3562			0.389900	144
	299												0.003047	111
	300												0.005116	165
										0.0483				1.512
													24.803	30

Figure 33: PRL-based reachability model



Performance = f(2sec)	300	1	2	3	4	5	294	295	296	297	298	299	300	Max Node	Node-Max
	1	0.7189			0.9879						0.9589			0.193359	162
	2		0.9592											0.959167	2
	3			0.9855										0.028497	27
	4				0.9879						0.9589			0.057669	162
	5					0.9818								0.029015	174
	295							0.9701						0.019296	103
	296								0.9848					0.051316	191
	297									0.9614				0.031930	226
	298	0.7189			0.9879						0.9589			0.197745	162
	299											0.9875		0.987518	299
	300												0.9824	0.046749	227
		0.0359	0.0032	0.0480	0.0790	0.0524	0.0317	0.0291	0.0591	0.0481	0.0703	0.0428	0.0720		14.437
														57.595	68

Performance = f(8sec)	300	1	2	3	4	5	294	295	296	297	298	299	300	Max Node	Node-Max
	1													0.193359	192
	2		0.9648	0.9812		0.9811					0.9804	0.9810		0.981222	81
	3		0.9811	0.9975		0.9974					0.9967	0.9973		0.997612	38
	4													0.057669	81
	5		0.8670	0.8834		0.8833					0.8827	0.8832		0.883622	80
	295													0.019296	72
	296													0.051316	227
	297													0.031930	226
	298		0.9812	0.9976		0.9975					0.9968	0.9974		0.997776	174
	299		0.9810	0.9973		0.9972					0.9966	0.9971		0.997467	118
	300													0.046749	227
			0.3187	0.3241		0.3241					0.3238	0.3240			31.525
														107.114	43

Performance = f(16sec)	300	1	2	3	4	5	294	295	296	297	298	299	300	Max Node	Node-Max
	1													0.193359	192
	2													0.981222	81
	3													0.997612	38
	4													0.057669	81
	5													0.883622	80
	295													0.019296	50
	296													0.051316	227
	297													0.031930	226
	298													0.997776	174
	299													0.997467	118
	300													0.046749	227
															5.528
														107.173	30

Figure 34: PRL-based performance mode

Reach = A2 Performance= A3 Stability = A4	300	1	2	3	4	5	295	296	297	298	299	300	Max Node & Community Average	Optimal Node	Node Rank Value	Node Rank	Count
33%	1	2.0025	0.0000	0.0000	0.8576	0.0000	0.0000	0.0000	0.0000	1.8338	0.2196	0.0000	0.196795	121	2.0025	34	1
33%	2	0.0000	0.4403	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.440342	2	0.4403	286	1
33%	3	0.0000	0.0000	0.5945	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.020401	231	0.5945	204	0
	4	0.0000	0.0000	0.0000	0.8576	0.0000	0.0000	0.0000	0.0000	1.8338	0.0000	0.0000	0.063634	44	0.8576	107	0
	5	0.0000	0.0000	0.0000	0.0000	0.6002	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.023112	166	0.6002	202	0
	295	0.0000	0.0000	0.0000	0.0000	0.0000	0.4274	0.0000	0.0000	0.0000	0.0000	0.0000	0.013298	110	0.4274	292	0
33%	296	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6111	0.0000	0.0000	0.0000	0.0000	0.034808	12	0.6111	178	0
33%	297	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.8813	0.0000	0.0000	0.0000	0.029060	199	0.8813	62	0
33%	298	2.0025	0.0000	0.0000	0.8576	0.0000	0.0000	0.0000	0.0000	1.8338	0.0000	0.0000	0.196721	1	1.8338	35	0
	299	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5720	0.0000	0.012674	78	0.5720	211	0
	300	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6668	0.044159	165	0.6668	147	0
														64			294
													31.153837				

Unchecked to Reset

	300	1	2	3	4	5	295	296	297	298	299	300	Max Node & Community Average	Optimal Node	Node Rank Value	Node Rank	Count
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.196795	121	0.0000	187	0
	2	0.0000	0.4688	0.4734	0.0000	0.4728	0.0000	0.0000	0.0000	0.4426	0.2246	0.2455	0.265038	11	0.4688	42	0
	3	0.0000	0.4664	0.4716	0.0000	0.4710	0.0000	0.0000	0.0000	0.2282	0.2283	0.2400	0.020401	231	0.4716	41	0
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.063634	121	0.0000	187	0
	5	0.0000	0.4500	0.4551	0.0000	0.4545	0.0000	0.0000	0.0000	0.2021	0.2022	0.2496	0.023112	166	0.4545	44	0
	295	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.013298	121	0.0000	180	0
	296	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.034808	121	0.0000	180	0
	297	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.029060	199	0.0000	179	0
	298	0.0000	0.4301	0.2284	0.0000	0.2283	0.0000	0.0000	0.0000	0.6514	0.2283	0.0000	0.307753	111	0.6514	18	0
	299	0.0000	0.2246	0.2283	0.0000	0.2283	0.0000	0.0000	0.0000	0.2281	0.2282	0.0000	0.226612	80	0.2282	145	0
	300	0.0000	0.2455	0.2469	0.0000	0.2463	0.0000	0.0000	0.0000	0.0000	0.0000	0.2436	0.044159	165	0.2436	104	0
														10		39523	285
													41.610155				

	300	1	2	3	4	5	295	296	297	298	299	300	Max Node & Community	Optimal Node	Node Rank Value	Node Rank	Total Value
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.196795	121	0.00000	118	4.0051
	2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.265038	111	0.00000	118	4.6308
	3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.020401	111	0.00000	118	4.9618
	4	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.063634	121	0.00000	118	1.7152
	5	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.023112	111	0.00000	118	4.8366
	295	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.013298	121	0.00000	110	0.8547
	296	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.034808	121	0.00000	110	1.2222
	297	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.029060	111	0.00000	109	1.7626
	298	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.307753	111	0.36276	29	14.6830
	299	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.226612	111	0.00000	109	2.9699
	300	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2492	0.254983	41	0.24922	65	7.2701

Figure 35: PRL-based aggregate event model

4.3.2.6 STRUCTURAL MODELS

Structural profiles model alternative physical and virtual network structures incorporating connectivity, node and application profiles and operational models. This is the central point of both infrastructure dynamics and operational state management and control. As shown in Figure 36 and Figure 37, the physical hierarchy and HVC-based hierarchy models follow a ranking methodology based on node resource capability and operational superiority, respectively. The ranking methodology in the physical hierarchy model operates on selecting computational and communications superior nodes, independent of operational flexibility or resilience. Resource superior or high ranking nodes are assigned to single tiers, cannot span beyond a single node instance, nor can nodes be re-assigned or re-provisioned due to negative operational conditions within the three (3) operational cycles[15]. Alternatively, the ranking methodology supporting the HVC-based hierarchy operates on elevating nodes, which exhibit minimum perturbation or demonstrate maximum operational stability, reachability and performance. The model can highlight structural flexibility as nodes span multiple tiers with multiple virtual instances and can be re-assigned or re-provisioned statically or dynamically within any of the operational cycles. The HVC-based model also accounts for nodes with richer computational and communications capabilities via the performance state management process. Finally, Figure 38 illustrates the structural results supporting independent simulation runs for each of the seven (7) major structural models for comparison purposes. After each run, the results within the respective columns are updated based on currently selected configurations, while structural results from earlier runs persist based on a previously selected physical or HVC configuration.

[15] A very conservative assumption is static structural changes can occur on the edge of 16secs

Total Rank	Tier 0 Rank	Tier 1 Rank	Tier 2 Rank	Tier 3 Rank	Tier 1 Parent	Tier 2 Parent	Tier 3 Parent	300	1	2	298	299	300
2	8	1	2	2	1	1	1	1	1.20E+08		1.20E+08		
1	294	69	10	1	2	2	2	2		#######			
8	195	8	4	-	3	3	1	3					
3	26	37	1	-	4	4	2	4			1.20E+08		
5	134	11	7	-	5	5	2	5					
163	86	-	-	-	12	8	2	296					
252	102	-	-	-	96	4	1	297					
95	5	-	-	-	166	23	2	298	1.20E+08		1.20E+08		
345	252	-	-	-	78	11	2	299				4.65E+07	
303	213	-	-	-	19	4	2	300					4.65E+07
206	186	74	10	2	72	10	2	300					

Figure 36: Physical node capability model

Total Score	Tier 0 Cluster Rank	Tier 1 Rank	Tier 2 Rank	Tier 3 Rank	Tier 1 Parent	Tier 2 Parent	Tier 3 Parent	Good Hearchy	300	1	300	T0 - 2sec	Min Neighbor or T1 parent	T1 - 8sec	T2 - 16 sec
3	3	90	38		1	1	1	Good	1	0.299	0.000	0.24990	1	0.28797	0.00004
109	109	94			75	152	1	Good	2	0.000	0.000	0.47754	2		
25	25	50			285	152	1	Good	3	0.000	0.000	0.01993	180	0.01978	
341	251				200	1	1	Good	4	0.000	0.000	0.07448	171		
69	69	47			1	1	1	Good	5	0.000	0.000	0.02751	268	0.01044	
22	22	16			26	75	1	Good	296	0.000	0.000	0.03578	61		
43	43	41			60	145	1	Good	297	0.000	0.000	0.02623	291	0.00734	
370	268				197	152	1	Good	298	0.299	0.000	0.27276	39		
38	13	28			16	152	1	Good	299	0.000	0.000	0.01100	66	0.00897	
149	241				227	1	1	Good	300	0.000	0.041	0.06481	69		
202	178	98	8	2	92	8	2	0				274	104	98	8

Figure 37: HVC node operational model

Selected Configuration				Physical Mesh			Physical Centralized			Physical Tiered			Physical,Virtual (Static VM) Centralized			Physical, Node Dynamic Virtual Tiered			Physical, Service Dynamic Tier			Physical, Node & Service Dynamic Tier		
# Levels	4			0			1			2			3			4			5			6		
# Nodes - T0	300	Client nodes	274	300			274			274			274			274			274			274		
# Nodes - T1	16	Tier 1 nodes	16	26			26			16			16			16			16			16		
# Nodes - T2	8	Tier 2 nodes	8	0			0			8			8			8			8			8		
# Nodes - T3	2	Central nodes	2	0			0			2			2			2			2			2		
TimePerLevel	Fixed	n/a		Select physically dominant server nodes in modular fashion			Select the physically dominant dedicated server nodes			Select the physically dominant dedicated nodes, but select 16 Tier 1 Nodes for server nodes			Select static VM server nodes in modular fashion, leverage dominant, static VM-based hierarchy			Select dynamic server nodes in modular fashion, leverage dominant, dynamic VM-based node hierarchy			Select dynamic application server nodes in modular fashion, leverage dominant, dynamic application-VM-based node hierarchy			Select dynamic application server nodes in modular fashion, leverage dominant, dynamic VM based node hierarchy		
		# Nodes - Total	100	53	52	53	46	46	46	49	10	2	54	54	54	69	10	8	300	300	300	0	0	0
Node #	T1 CH	T2 CH	T3 CH	T1 CH	T2 CH	T3 CH	T1 CH	T2 CH	T3 CH	T1 CH	T2 CH	T3 CH	T1 CH	T2 CH	T3 CH	T1 CH	T2 CH	T3 CH	T1 CH	T2 CH	T3 CH	T1 CH	T2 CH	T3 CH
1	121	121	121	1	198	45	1	1	1	1	1	1	133	133	133	111	111	111	1	1	1	0	0	0
2	2	165	165	253	195	44	2	2	2	2	2	1	2	2	2	2	111	111	2	2	2	0	0	0
3	231	231	165	1	3	249	3	3	3	3	3	3	237	237	237	231	120	111	3	3	3	0	0	0
4	44	121	121	1	121	1	4	4	4	4	4	1	133	133	133	188	188	111	4	4	4	0	0	0
5	166	171	165	166	8	1	5	5	5	5	5	3	214	214	214	166	166	111	5	5	5	0	0	0
295	110	171	165	67	276	1	36	36	36	18	7	3	207	207	207	110	120	41	295	295	295	0	0	0
296	12	199	165	8	163	12	46	46	46	61	7	1	263	263	263	77	44	111	296	296	296	0	0	0
297	176	171	41	1	34	100	3	3	3	43	7	3	12	12	12	199	199	41	297	297	297	0	0	0
298	121	121	121	66	100	23	59	59	59	8	2	1	133	133	133	111	111	111	298	298	298	0	0	0
299	78	233	45	45	44	12	1	1	1	8	11	1	133	133	133	78	120	111	299	299	299	0	0	0
300	165	165	165	11	110	66	88	88	88	15	14	1	264	264	264	165	165	111	300	300	300	0	0	0

Figure 38: Structural results model

4.4 EVALUATING HIERARCHICAL VIRTUAL CLUSTERS

4.4.1 NETWORK STRUCTURES

Virtualization offers researchers a refreshing perspective to composing computational, storage and communications infrastructure. In DCS, virtualization transforms node and communications infrastructure design. The particular merits of virtualized infrastructure are flexibility and scalability – the key characteristics of HVC. However, it also introduces complexity in provisioning and managing resources in the face of operational dynamics associated with mobile and decentralized computing. Figures 39-45 illustrate alternative networking structures

encompassing either traditional physical infrastructure as shown in Figures 39-41 or HVC-based virtualized infrastructure models overlaid on a physical mesh of nodes as shown in Figures 42-45. Figure 39 follows the typical (wireless) mesh or P2P infrastructure while Figures 40 and 41 are instances of tiered physical infrastructure such as tiered client-server or hierarchical switched or routed infrastructure. Figures 42-45 are HVC models whereby the nodes are resource provisioned and functionally composed with virtual machine capabilities to support either static or dynamic composition of infrastructure.

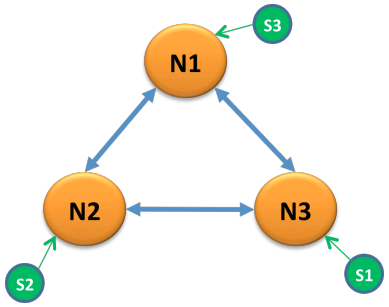


Figure 39: P-Mesh: Physical mesh node structure

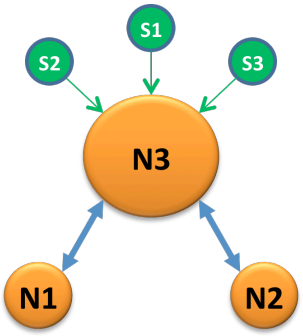


Figure 40: P-Central: Centralized (single-tier) physical node structure

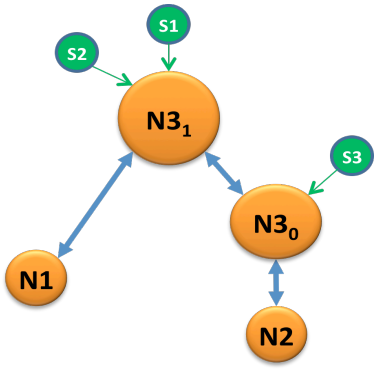


Figure 41: P-Tier: Multi-tiered physical node structure

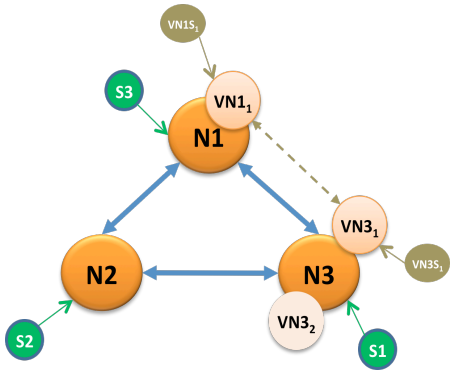


Figure 42: HVC S-Tier: Physical mesh, static HVC node hierarchy

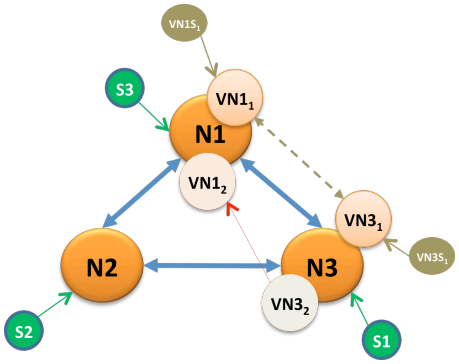


Figure 43: HVC D-Tier: Physical mesh, dynamic HVC node hierarchy

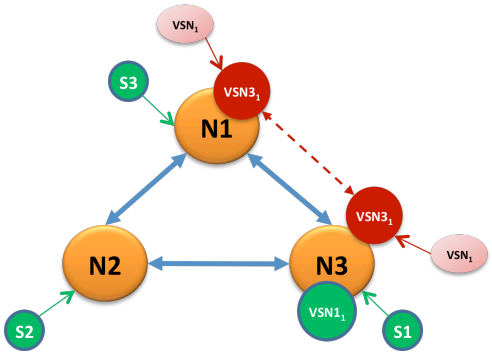


Figure 44: HVC Service: Physical mesh, dynamic HVC service hierarchy

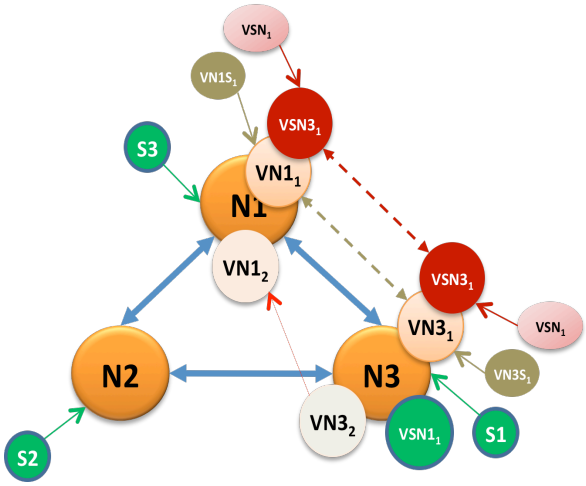
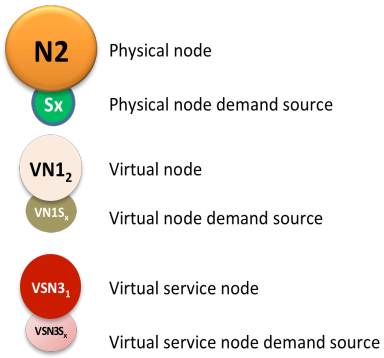


Figure 45: HVC DS-Tier: Physical mesh, dynamic HVC node & service hierarchy



Legend: Graph node & demand source

Table 5 lists the respective infrastructure configurations along with a speculative profile of the respective configurations previously shown in Figures 39-45. The premise in this thesis is that higher levels of infrastructure and operational value can be achieved with HVC as a structural alternative to building scalable and flexible DCS.

Table 5: Comparison of infrastructures' strengths and weaknesses

ID	NetworkType	Node Diversity	Tier Structure	Infrastructure Scal	Infrastructure Flexibility	Infrastructure Overhead	Infrastructure Resilience	Infrastructure Cost	Performance	Mobility
Mesh	Physical Mesh Structure	Random, superior structural nodes	Flat Physical, No Hierarchy							
PN1	Physical Node Centralized Structure	Dedicated, Physically superior structural nodes	Single-Tier Physical Node Hierarchy							
PN3	Physical Node Multi-tier Structure	Dedicated, Physically superior structural nodes	Three-Tier Physical Node Hierarchy							
HVC3	Physical Mesh, Static HVC	Dedicated, virtually superior structural nodes	Three-Tier Static Virtual Node Hierarchy							
HVCD3	Physical Node Mesh, Dynamic HVC	Dynamic, virtually superior structural nodes	Three-Tier Dynamic Virtual Node Hierarchy							
HVCSD3	Physical Node Mesh, Service Dynamic HVC	Dynamic, virtually superior service nodes	Three-Tier Dynamic Servoce-specific Virtual Node Hierarchy							
HVCSD3	Physical Node Mesh, Service & Node Dynamic HVC	Dynamic, virtually superior service nodes and structural nodes	Three-Tier Dynamic service-specific Virtual Node & Virtual Node Hierarchy							

Poor	Neutral	Good

In the upcoming sections and in Chapter 5, respectively, we evaluate and compare the virtual structuring framework and emergent management system services operating against (baseline) traditional physical infrastructure models – specifically Figures 39-41. The specific objectives are to evaluate the scalability and operational complexities associated with the respective configurations operating under the event conditional models as described in Section 4.3.2.3.

4.4.2 MODELING STRUCTURAL FLEXIBILITY

As demonstrated by the peer-to-peer [96] and mesh networking [101] research communities, decentralized systems promote a more extreme distributed computing or internetworking system. The first observation of these networks is the lack of a fixed organization or hierarchical physical structure. Secondly, there is a common notion of self-discovery, organization and network bootstrapping. Thirdly, these networks change dynamically for self-healing purposes to accommodate community fluctuation. The anticipated challenges in DCS parallel these same behaviors or functions. Alternatively, an emergent [18] style of organizational dynamics counteracts the effects of internetworking and operational complexity in DCS. The

following attributes are designed into the DCS hierarchical virtual clustering framework:

- *edge involvement*: DCS proposes to collapse the edge, therefore removing the discontinuity that exists today between the 'core network' and the 'end user application. Thus, the edge or DCS node is inherently involved in the control and management planes;

- *global perspective*: the logical clustering framework supports both a local and global perspective. While state is aggregated upward across management tiers from lower to higher clusters, the control path reverses the perspective by pushing aggregate or policy management policies to local policies or actions.

- *compositional structure:* the emergent cluster hierarchy selects operationally superior clusters and clusterheads following the promotional methodology outlined in Section 4.2.2. The DCS environment assumes the nodes are being reinforced to participate in control policies, while cluster state is being coordinated for state consensus across individual clusters.

To compare DCS structural alternatives and to demonstrate HVC advantages, we modeled corresponding network structures over 300 nodes placed randomly in a space of 10000 x 10000 m^2. The adjacency matrix is formed across all nodes based on radio proximity range within each simulation epoch or iteration. Further, and for simplicity, all DCS nodes within connectivity range are trusted, and trust is recursively-enabled throughout the virtual hierarchy. Following Section 4.3.2.3, nodes are randomly configured with either single or multiple radios operating Wi-Fi (56mpbs) or WiMAX (16mbps) radios configurable with a range of up to 50m or 500m, respectively. Node roles or DCS network capabilities are a function of their dynamic or static ability[16] to support virtual machines (VM), server and WiMAX

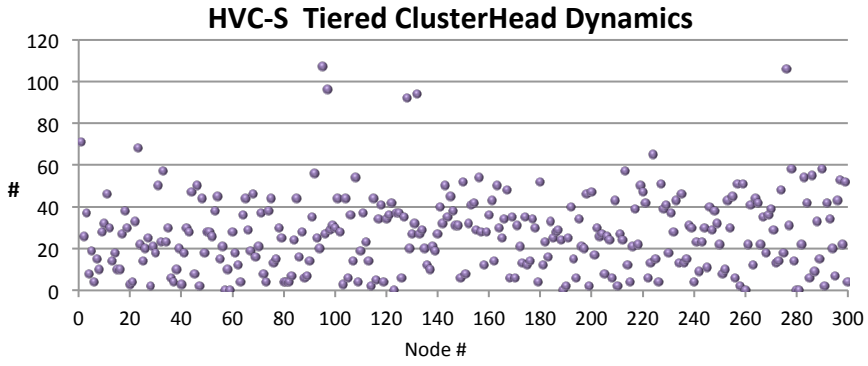
[16] Supported on HVC configurations only, randomly (0,1) reconfigurable every 2 seconds.

functionality. Node resource capacities follow the computational and connectivity capabilities. Since DCS is a macro-level simulation analysis, we do not model or analyze low-level channel access or packet-level transport and only concern ourselves with connectivity, aggregate processing capacities and operational conditions including node speed, node or link failures, and usage or load conditions.

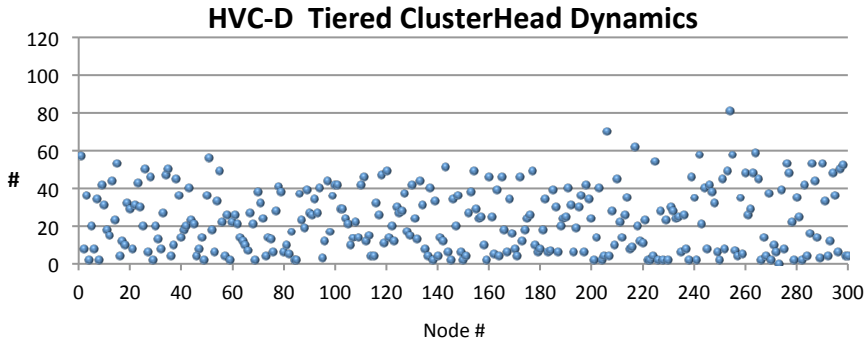
Operational incidents follow the same event model for each of the structural configurations and respective simulation runs as described in Section 4.3.2.3. DCS structural models encompass both (static) physical and (static and dynamic) HVC-based hierarchical models and they follow a ranking methodology based on node resource capability and operational superiority, respectively. The ranking methodology in the physical hierarchy model operates on selecting computational and communications superior nodes, independent of operational flexibility or resilience. We assign resource superior or higher-ranking nodes to single tiers. These nodes cannot cover beyond a single node instance, nor be re-assigned or re-provisioned due to negative operational conditions within three (3) operational cycles. Alternatively, the ranking methodology supporting the HVC-based hierarchy operates on elevating nodes, which exhibit minimum perturbation or demonstrate maximum operational stability, reachability and performance. The model can highlight structural flexibility as nodes span multiple tiers with multiple virtual instances and can be re-assigned or re-provisioned statically or dynamically within any of the operational cycles. Finally, following Section 4.3.2*,* we model and evaluate each network structure independently during the same 140-second interval within a longer simulation run.

As shown in Figure 46(a-d), structural hierarchy changes are more apparent across the HVC configurations Figure 46(a, b) then either the physical structures Figure 46(c, d) configurations. The static HVC configuration Figure 46(a) behaves more approximately to the dynamic HVC structure Figure 46(b) but the dynamic HVC structure appears to perform more balanced in comparison to the other configurations. There are also several nodes within the static HVC structure that exhibit a greater

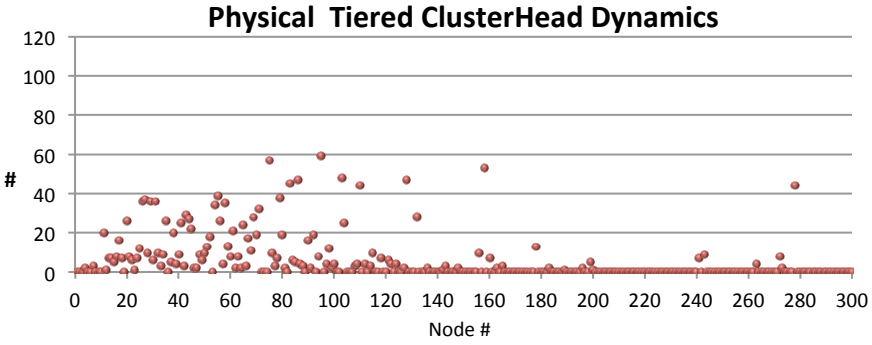
number of changes, reflecting more changes occurring across nodes at higher tiers of the hierarchy. Also apparent in the physical tiered structure Figure 46(c) is the limited number of nodes participating in the physical hierarchy; this is associated with the expected physical superiority of nodes positioned more centrally in the topology hierarchy. Finally, as shown in the Figure 46(d), a shared clusterhead leadership dynamic is visible in the physical mesh configuration across all nodes within the mesh or flat network structure.



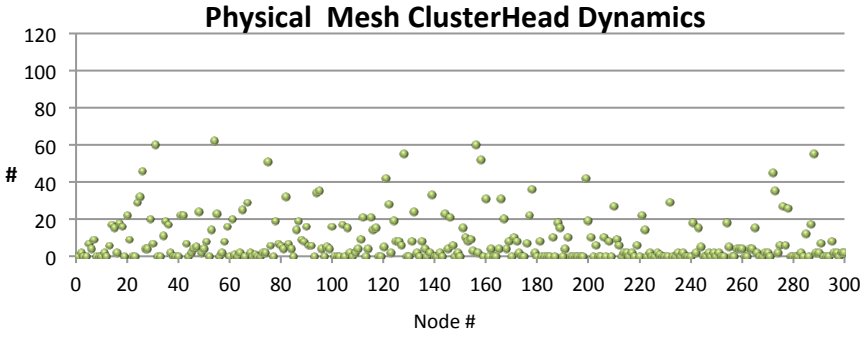
(a)



(b)



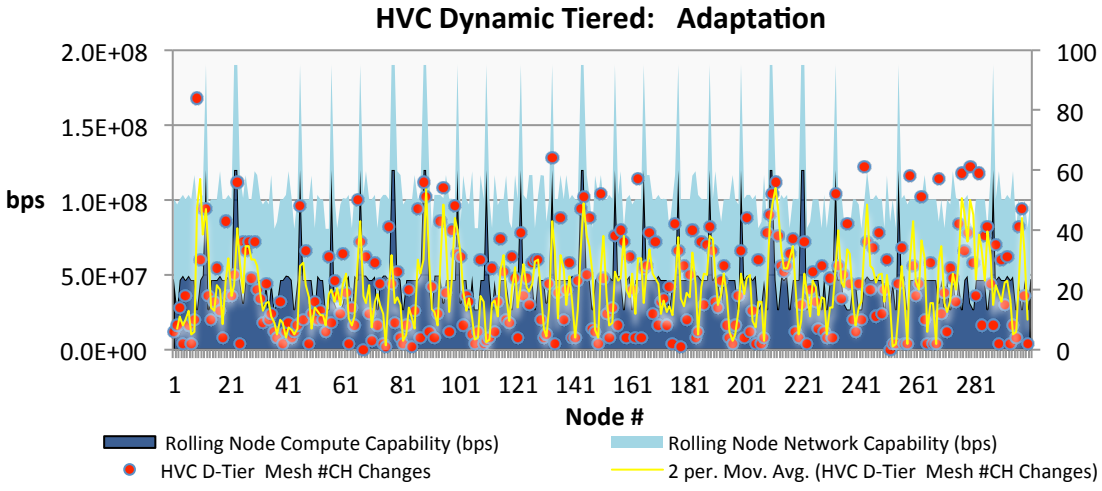
(c)



(d)

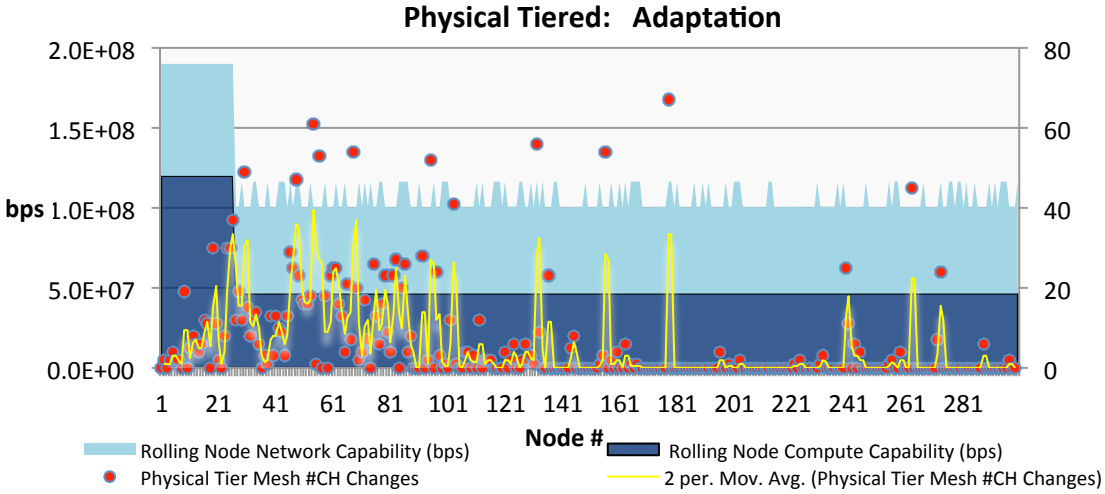
Figure 46: Clusterhead dynamics

To comprehend HVC structural flexibility and adaptation, we compare dynamic HVC against the traditional three-tier physical infrastructure using a consistent stream of operational event dynamics as described in 4.3.2.3. We are particularly interested in mobile device movement, device or connectivity failures and overload event conditions. Several key observations are shown in Figure 47 (a-b). Consistent with traditional, centralized physical infrastructure, computing and connectivity capabilities are provisioned (or modeled) at greater resourcing capabilities and concentrated to nodes 1-20 of the physical tiered structure as illustrated in Figure 47(a), while the equivalent physical compute and networking resource capabilities are more distributed in DCS via the dynamic HVC structure in Figure 47*(*b). The former reflects the traditional client and server compute and internetworking disparities that are prevalent in current physically centralized systems, largely based on resource dominant nodes. Also shown, the clusterhead changes are more prevalent and distributed in the dynamic HVC structure then in the physical tiered structure. This reflects both the decentralized physical properties in DCS in addition to the dynamic properties exhibited through virtual clustering and operational superiority. The apparent adaptation characteristics of HVC enable DCS to be more responsive to operational event perturbations without requiring dedicated nodes or resource superior physical infrastructure.



(b)

HVC Dynamic Tiered adaptation



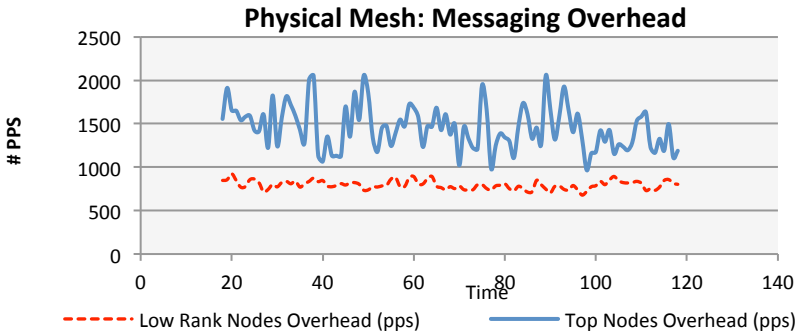
(a)

Figure 47: Physical Tiered adaptation

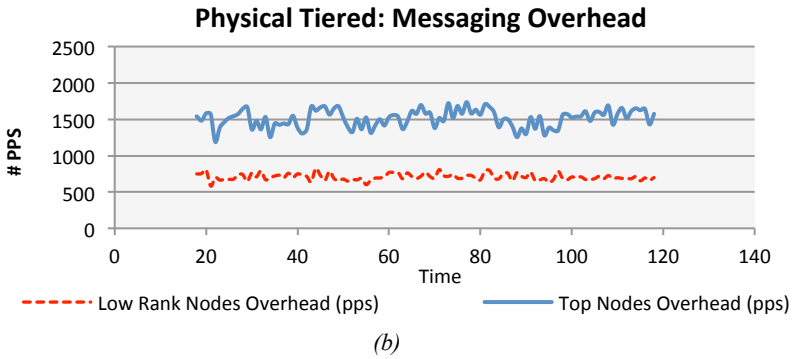
A comparison of messaging[17] overhead across the four comparison configurations (i.e., physical mesh, physical tiered, HVC S-tier and HVC D-tier structures) is also evaluated. Maintaining quantitative consistency across the structures, we model messaging overhead as a function of the number of structural tiers, neighboring nodes

[17] These messaging overhead results are primarily associated with structural communications for promotion across physical or virtual nodes, intra-clustering and inter-cluster communications across the virtual tiers and hierarchy.

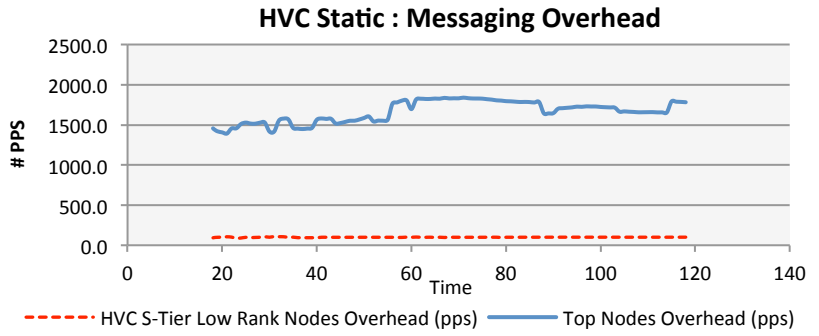
per tier, per cluster and physical (mesh) connectivity. Messages are exchanged at twice the rate of the log (neighbors) across all nodes in the hierarchy or physical mesh. Figure 48(a-d) illustrates the aggregate messaging disparities between physical and virtual configurations for high ranking and low ranking nodes only. Node ranking, as described earlier, is based on resource superiority for physical structures, while HVC-based structures on operational superiority. More specifically, unlike the physical configurations, high-ranking HVC nodes contribute a greater proportion of node-node communications than lower rank nodes due to the multiple virtual nodes operating within the same physical nodes, thereby participating across multiple tiers. Alternatively, the lowest ranking nodes operate primarily at the lowest (physical) tier and thus, are limited in overall structural messaging communications. Thus, while all nodes may participate at some limited messaging capacity, lowest ranking HVC nodes are statistically less likely to be involved because of both their resource and operational ranking or inferiority.



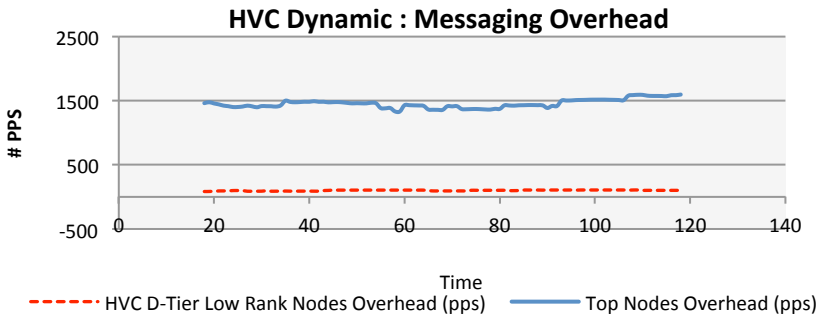
(a)



(b)



(c)



(d)

Figure 48: Messaging complexity over time (cumulative events)

Organizational superiority through operational dynamics allows DCS to accommodate shifts in structural hierarchy independent of node resourcing or physical capability characteristics. Maintaining the modeling assumptions outlined throughout this section, Figure 49 (a-c) illustrates node-level structural tiering differences (time-based moving average) between the dynamic HVC structure against the physical mesh structure and physical tiered structure. In the physical mesh case, mesh nodes select peer (server) nodes on a random basis and it is our assumption that mesh nodes do not exhibit fixed service node relationships or multi-level tiered behavior. Alternatively, the physically tiered structure is fixed to particular resource superior nodes and change infrequently based primarily on physical changes in the network. As described earlier, that frequency is conservatively assumed to occur outside of the three (3) operational cycles or a period of 16 seconds. As illustrated in Figure 49 (a), the dynamic HVC demonstrates more balance with more nodes

participating or gaining preference based on their operational characteristics over time.

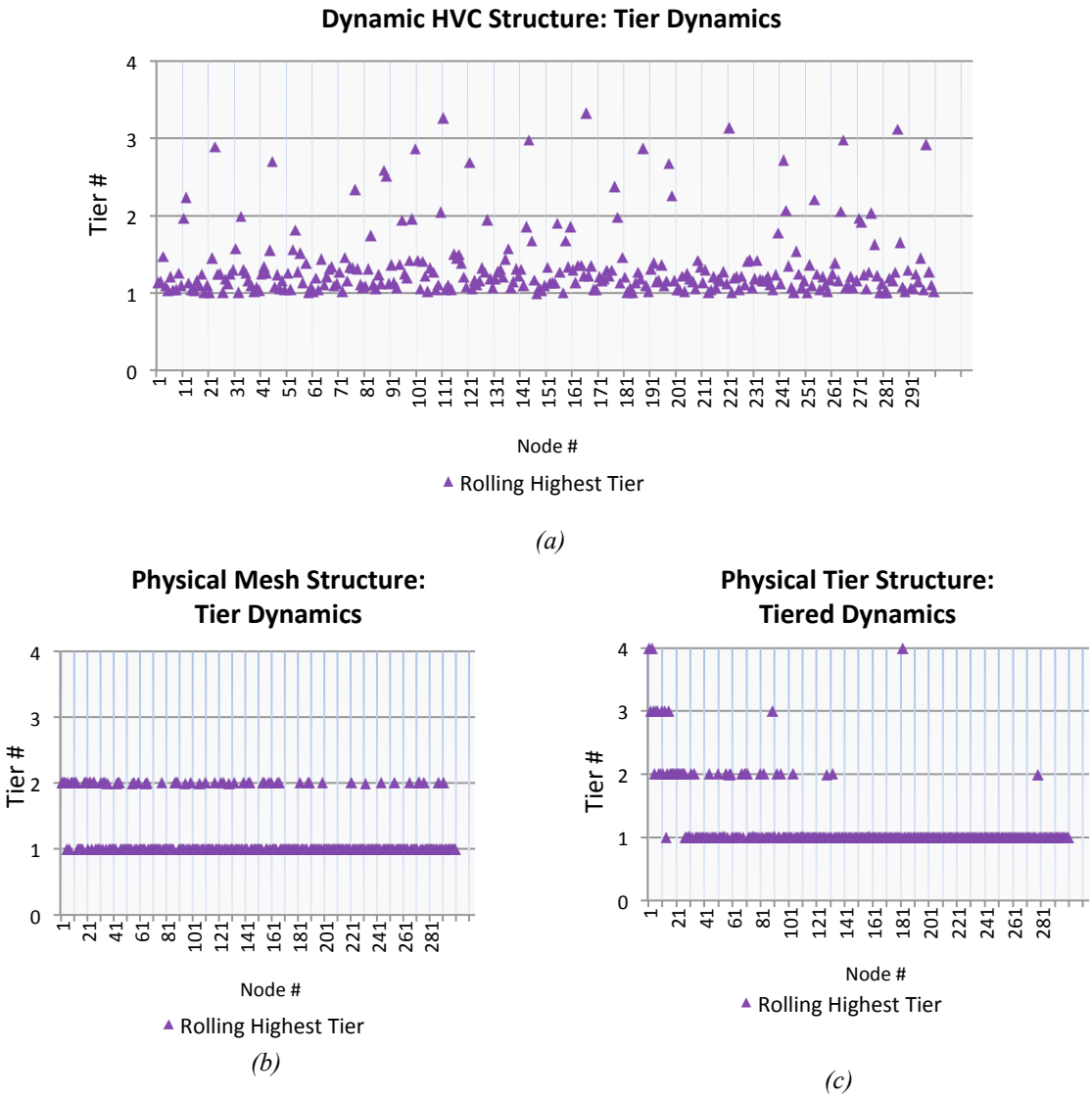


Figure 49: Comparison of tiered dynamics

4.4.3 OPERATIONAL MODEL

Figure 50 depicts the upward progression of state-management communications from local (child) clusterheads to a global (parent) clusterheads along with a set of metrics associated with the state management properties. As measurements are sensed across

the physical DCS nodes, temporal and spatial aggregation occurs across the logical clusters in hierarchical form. Thus, as cluster aggregation occurs, one can see slower state changing effects in the clusters at the higher end of the tree. The optimal selection of clusters for internetworking purposes, clusterhead selection and network optimizations will depend on the proportional mix of the management state variables. The objective of the hierarchical control system is to aggregate distributed state for the purposes of determining appropriate policy actions to affect stabilization, achieve performance objectives or to maximize or optimize reachability. Statistical methods are used to manage state representation and distributed policy control with minimal operator dependency.

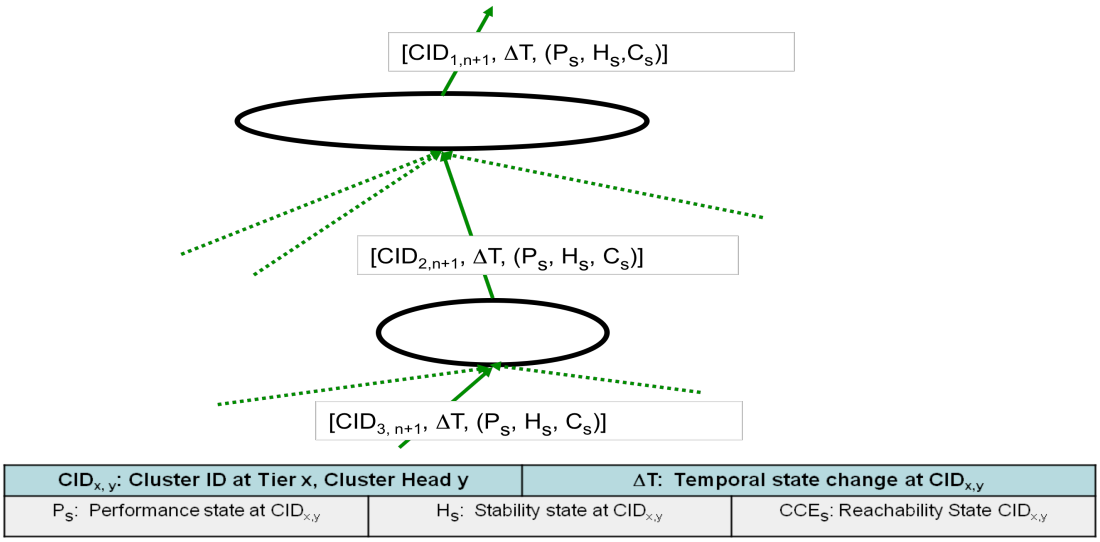


Figure 50: Cluster-based state messaging process

To draw out the emergence properties of DCS, we model per node randomized operational conditions or events associated with user or node operating conditions. This includes mobility speed, node or service failures and resource or traffic load conditions. Such conditions or events are captured locally and shared with neighbors to assess node and neighbors' operational state. Given the statistical nature of these events, networking configuration and service changes are observable at every modeling interval and across a section of DCS nodes. Environment instability, bottleneck conditions and service

unavailability should be the anticipated dynamics – in essence, the desired demand conditions for modeling evaluation. To offset this, the same events are used to measure and manage state with nodes to calculate the operational variables of reachability, stability and performance to promote and cluster around more resource capable and operationally superior nodes. This process is recursively performed throughout the network via promoted clusterheads to form hierarchies of virtual clusters about the same robust nodes. More specifically, the model implementation of reachability uses a local expansion [197] calculation on node movement, simple utilization calculations based on node and link load conditions and probabilistic measures of node unavailability using failure states and node movement. As recursive state management helps to form the virtual structure, the same operational variables are use to influence or reinforce policy control via the promoted clusterhead nodes. Policy-reinforced rewards (independent or multi-variable) drive specific stability, performance or reachability objectives as needed by applications or services that choose to leverage the HVC structures. As a simple illustration, DCS state management measures were model evaluated against node operational rankings to comprehend the operational superiority objectives. As shown in Figure 51, higher-ranking nodes (1 is highest) demonstrated higher values closer to the y-axis. More apparent are the graph expansion metrics, which are statistically influenced by node mobility, speed and wireless range and have high influence on node ranking. The actual operational values shown are additive (normalized) across the three state management variables.

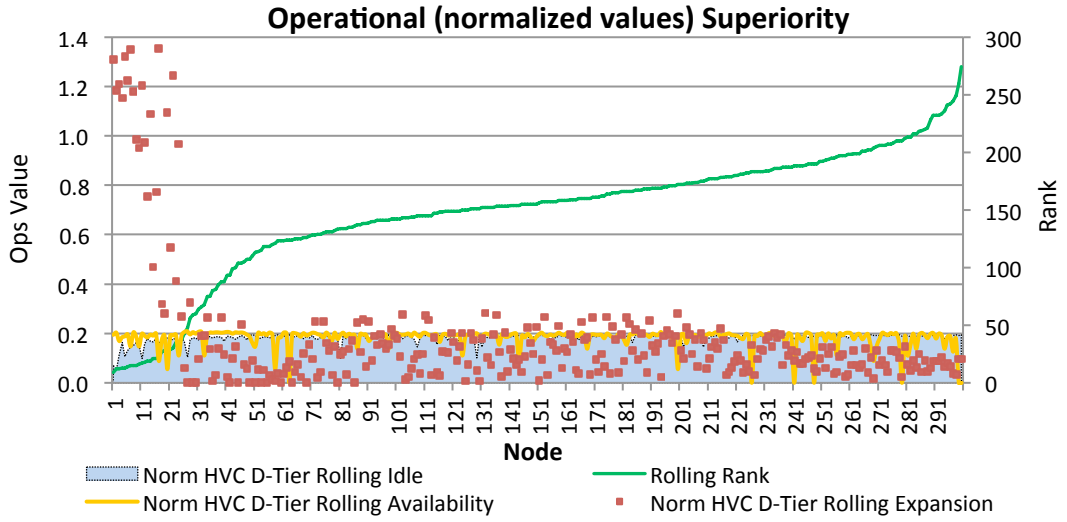


Figure 51: Operational superiority

4.4.4 MODELING OPERATIONAL DYNAMICS

State and policy in the DCS environment is distributed network-wide and applied locally through each of the clusters or clusterhead. Operational state dissemination and storage is recursively managed hierarchically to support structural (HVC) formation. The following three state measures are used to assess operational state to form the structural graph of DCS via logical clusters and over the HVC hierarchy:

Reachability: To assess the connectivity strength and capability of a particular cluster interconnectivity; we exploit a technique used in peer-to-peer networks to assess the connectivity strength – *graph expansion* [197]. Along with this measure, the capability strength of the pass-through links, enabling inter-cluster traffic is utilized – similar to inter-provider traffic and connectivity. The purpose of this state measurement is to assess the overall interconnectivity strength of a cluster for achieving various reachability objectives. In demonstrating reachability capability or adaptive properties, we model the environment under recurrent conditions of node or link failures, overloaded node conditions and varied node mobility. In DCS, nodes with higher physical computing and networking capabilities tend to have higher reachability or expansion measures. However, with the redistribution of those same

(quantitatively) resource capabilities across HVC nodes via dynamic virtual structures, the reachability profiles can be dynamic and persistent across a greater number of nodes, and thus a greater number of operationally superior nodes; making DCS more adaptive and scalable.

Stability: The proposed networking system assumes a high-degree of intra-networking and inter-networking movement and we view this both a challenge and opportunity for scalability [25], [31], [162]. DCS nodes can be online, mobile, hibernating or off-line while the topology may be irregular and fluctuating in path availability and reliability. Thus, end-end connectivity can be highly unstable for any portion of the decentralized network. In representing the stability (e.g., availability) state of the DCS networks, one must first represent the underlying mesh networks using the logical clustering procedures, and then utilize the clustering service to measure the stability state of the nodes within particular clusters and across all clusters' in the tree hierarchy. *Entropy*-based statistical techniques [162],[165] will capture stability differentiation across the various cluster formations in the network - measuring the degree of connectivity availability or reliability state across particular cluster nodes or set of clusters.

In this work, we model stability in the form of simple node availability. While availability is a value-add that virtualization improves in terms of operational stability; in HVC, a promoted virtual node that serves connectivity or computing value to other DCS nodes can serve structural value across multiple tiers, and thus is functionally equivalent to physical nodes in their system-wide dependence. The key value distinction with HVC, however, is that we can increase the number of virtual nodes (or links) thereby increasing the overall DCS availability without the expensive requirement to introduce or provision physical nodes.

Performance: Under normal or stable operating conditions, network state may be captured using traditional statistical performance measures. Performance may be utilization, delay or other related metric and can be local (next hop) or perceived

global (end-end). Representation of performance is based on probabilistic measures, captured locally per node and within a cluster above and across the tree hierarchy. *Distributed belief propagation* [168], [170], [177] methods can be used to capture intra- or inter-cluster performance belief. This method can support recurrent techniques to capture multi-cluster performance across the network or vertically across the hierarchy. In Chapter 5, we model DCS performance dynamics using simple utilization metrics. While utilization is shown to be generally higher for the HVC nodes, it also tends to reflect a more, balanced distributed usage across the hierarchy - towards superior nodes or centralized servers or high-end routers. In DCS, pre-designed and over-provisioned nodes lack flexibility and can create static bottleneck points or resource imbalance across DCS, thereby reducing system-wide scalability.

We also investigate the operational aspects of the emergent design and convergence properties of DCS in detail to the conditional events associated with loss of service connectivity, instability or performance degradation. The emphasis of comparison is in characterizing the operational stability and convergence properties of DCS emergent management services in the context of HVC static and dynamic structures. Similar to the HVC structural analysis, the emergent management baseline objectives and comparison focus on demonstrating operational improvements over traditional physical, mesh or tiered structures.

Chapter 5

5 EMERGENT CONTROL & MANAGEMENT

5.1 INTRODUCTION

The proposed HVC architecture organizes distributed state and policy in hierarchical and virtual fashion for DCS management and network control. Our hypothesis is that the integration of HVC, as a self-organizing framework, with an emergent management architecture will improve the scaling properties of DCS-based applications and services. While today's models for network control and management have proven to lack scalability and responsiveness based on centralized models, it is unlikely that singular organizational models can withstand the operational complexities associated with DCS. In this chapter, we present the key emergent management subsystems and demonstrate that emergence in an HVC environment behaves better or outperforms the same services implemented over traditional structural organizational models.

5.2 DISTRIBUTED STATE MANAGEMENT

5.2.1 FRAMEWORK

As exposed previously [8], [12], [14], [92], the tasks of managing and provisioning Internet networks and distributed systems remain a manual activity for administrators. As the Internet transforms towards a more decentralized and ubiquitous wireless edge, the blend of heterogeneous computing devices and wireless access technologies will pose greater challenges for provisioning and management. Similar to [19], [28], [31], and [158], the course proposed in this thesis is to redirect network control and management systems towards an emergent and self-organizing framework to relieve operational burden and complexity in DCS.

An observation called out in [30] suggests designing systems based on the notion of data or state-driven descriptions and predictability. Active research in this area includes swarming techniques [28] for routing and traffic engineering, reinforcement [156] and reinforcement learning techniques [153], [159] for distributed network control. Bayes network techniques are widely used for distributed inference supporting fault isolation and intrusion attacks [100] and neural network techniques [164] for internetworking optimization. Entropy [165], [166] or information theoretic approaches are applied and studied in data streaming, mobile routing optimization and WLAN access communications [113]. Finally, graph theoretic techniques [197], [199] are common in peer-to-peer networking strategies. What is consistent amongst these approaches is the need for computationally recurrent, statistical or learning methods to offset operator or manual intervention. Following [23] and [179], the motivation of this work is towards establishing a 'natural' approach to managing growing IT complexity and scale, rather than manual or rule-based automation schemes, which today's management systems are predominantly based. More specifically, concepts of organized hierarchy, state or policy aggregation, exchange of such states to different portions of the network for coordinated intelligence and prediction have direct parallels to the structure and organization of the neocortex layer of the human brain [179]. The neocortex senses and maintains data and information storage and knowledge retrieval for building intelligence and prediction. The complex structures of the brain builds knowledge from neural state captured from the ears through hearing, eyes through seeing and the other senses similarly to build and store temporal images or representations of our interactions and experiences from birth. As humans grow, so does the neocortex in neural hierarchy and stored knowledge by extending experienced state into predicted knowledge and abstract pattern comprehension and eventually to more complex associations leading to creative discovery and wisdom. In this work, it is not the aim to duplicate such a complex structure. Alternatively, we promote an architecture that is based on a feasible (implementation) hierarchical clustering structure and context-specific (i.e.,

reachability, performance, stability) constructs capture DCS operational state and their integration. This enables, in decentralized fashion, aggregate intelligence and prediction in managing and controlling DCS.

In a dynamic DCS environment, the emergent system is capable of storing and reproducing state data and knowledge with an online requirement to compile and manage state in real-time and historical fashion. Figure 52 depicts the representation of the DCS emergent framework to help guide the control and management approach. As shown, the emergent behaviors are partitioned across four (4) concealed planes of the operational control system:

- *global*: associates with a higher degree of abstraction, aggregation and distributed knowledge of the networked system;
- *local:* associates more closely with discrete, event or real-time estimation; has direct peering and interactions with other neighboring elements;
- *state:* represents temporal or spatial status of any node element, an aggregate or cluster of node elements for some operational state of the networked system;
- *policy:* represents configuration, guiding rules or various actions that control node elements or connectivity of the networked system.

The operational partitioning does not represent any absolute, physical aspects of the emergent system, rather a relative organizational of control and management operations. For example, we may have clusters that have relative *global* association to particular set of lower level clusters, but have *local* association to a higher-level cluster. Alternatively, aggregate (temporal) state of the system at one level may represent, relatively, a temporal instance at a higher level. Accordingly, a two-level emergent system such as seen in flocking or swarming behavioral systems [28], [169], is extended to a more generalized emergent structure, albeit exhibiting operational novelty [18] at multiple levels in DCS structural operation.

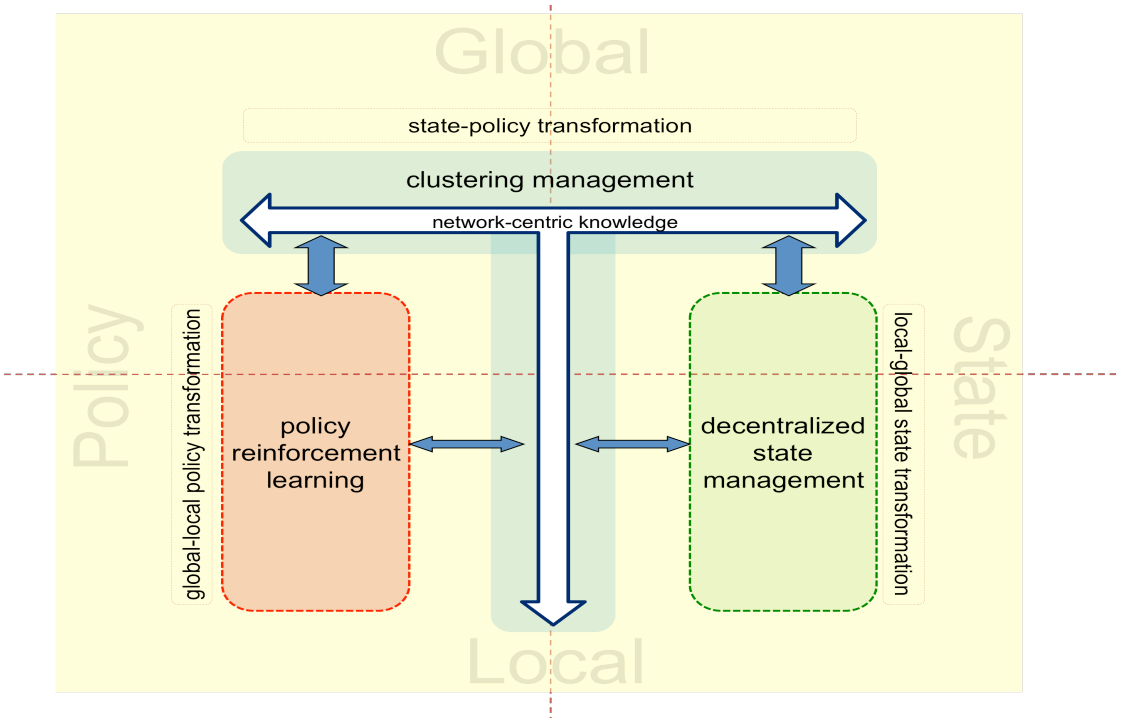


Figure 52: Emergent framework

To employ the emergent framework, we summarize the following subsystems:

i. *clustering management:* manages the overall assignments and allocations over the established cluster hierarchy formations; handles cluster addressing and messaging mechanics;

ii. *distributed state management subsystem:* manages cluster-level stability state through entropy-based aggregation, cluster-level distributed performance state through cooperative communications and network reachability via node and cluster expansion and associated connectivity properties;

iii. *policy-driven reinforcement learning subsystem:* balances global policy strategies against distributed, local actions using higher-level cluster feedback and multi-threaded, state-based rewards for policy reinforcement actions;

iv. *network-centric knowledge:* maintains hierarchical order of distributed state and policy across the decentralized system of physical nodes and VM-based clusters.

The emergent management framework and HVC combine to control or optimize DCS by balancing fluctuating operational conditions across the breadth and depth of the HVC structure. The three state management vectors of the emergent framework are elaborated in the following sections.

5.2.2 REACHABILITY STATE MANAGEMENT

Graph theoretic concepts applied towards resilience and connectivity evaluation are a recognized area or research in sensor [190], [191], mesh and peer-to-peer [196], [197] networking communities. In this section, we employ similar concepts to assess DCS physical and virtual network reachability and robustness. One can assess both a node or edge-level measure since the concepts apply for both edge-based connectivity, as well as node-based (machine) connectivity to assess reachability strength. The purpose of using either or both measures is to allow flexibility to the specific application or management service implementation.

5.2.2.1 SUBSYSTEM DESIGN

The notion of reachability or connectivity [193], [194], [195] refers to a computer science problem (or quantitative measure) of whether two vertices u and v in a directed graph are connected by a path. Specifically in this work, *edge expansion* is used towards the wireless mesh link interconnectivity, while *node expansion* is applied towards the VM node interconnectivity. The latter (i.e., node expansion) follows more closely with the peer-to-peer model of distributed computing while edge expansion towards network connectivity and dimensioning. The operational interest herein regarding dimensioning is a local, measure of cluster-level connectivity strength that allows one to measure the ability to find any node outside of a given cluster along with a scalar measure of the cluster's robustness or resource capability or capacity. In other words, through the combined concepts of graph expansion [197] and dimensioning capability applied to a cluster graph, one can

assess not only sub-graphs' breadth or reach (expansion), but also the strength (capacity capability) of this reach.

As depicted in Figure 53 and 55, there are two graph views depicted of the same cluster C_{ij}. Figure 53 reflects the connectivity structure associated with the physical mesh connectivity while Figure 54 depicts a virtual network with 'blue' (or circled) nodes reflecting the distributed service implemented as a VM-based overlay network. In this scenario, only the 'blue' VM virtual network environment is shown, as the other VM-based networks would depict an alternative node expansion profile. It should be noted that the concepts of node and edge expansion could be used in both scenarios. Depending on the application or usage, node expansion is applied to the VM service environment when node dependency is critical, while edge expansion is best applied to network connectivity when link dependency is critical.

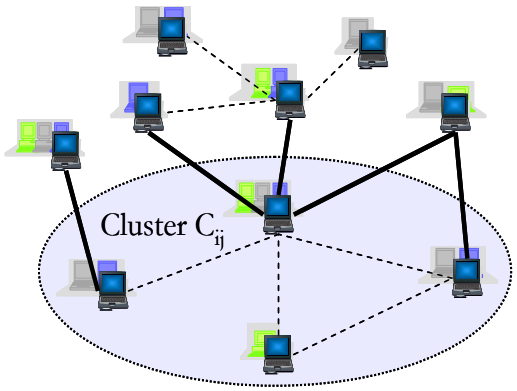


Figure 53: Mesh link cluster edge expansion

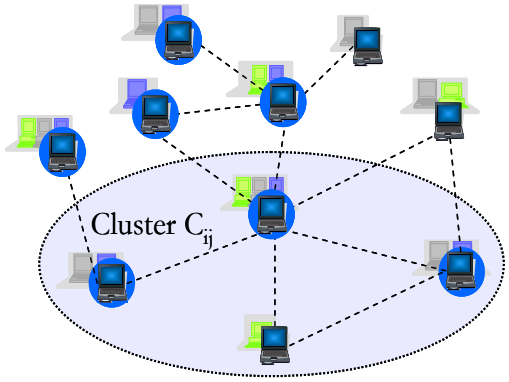


Figure 54: VM-based cluster node expansion

While both graphs rely on common nodes and links which reflect the physical footprint of the network, the peer-to-peer network also utilize VM nodes that are NOT directly connected to the cluster, enabling wider service connectivity within the broader (outside of the C_{ij} cluster) DCS network. Thus, edge expansion (*Equation 1*) evaluates the cluster's physical mesh connectivity reach, while node expansion (*Equation 2*) evaluates cluster service-level connectivity reach. Cluster coefficients associated with the expansion measures are:

Cluster Edge Expansion:

$$e(S) = \frac{\delta(S)}{|S|}, \quad (1)$$

where S is a cluster of V in graph G = (V, E) and δ(S) is the set of links between S and V\S.

Cluster Node Expansion:

$$n(T) = \frac{\delta(T)}{|T|}, \quad (2)$$

where T is a cluster of V in graph G = (V, E) and δ(T) is the set of nodes between T and V\T.

In addition, we define the cluster expansion capability (below) for both cluster nodes (*Equation 4*) and edges (*Equation 3*) to assess the operational 'strength' or 'width' of the cluster expansion in terms of its operational characteristics. This can include edge capacity or failure, node fault tolerance, computational processing footprint, or other relevant measures. The term 'capability' is chosen to reflect the design variability that one can be afforded towards expressing cluster expansion capability. The importance of this distinction is to distinguish from the traditional singular association of link (bandwidth) connectivity capability. As shown, the terms below include both the expansion coefficient terms as well as the edge (or node) capability variables.

Cluster Edge Expansion Capability:

$$c_e(S) = e(S) * \sum_i (u_i, v_i) * c_i, \quad (3)$$

where u_i is an edge in S, v_i is an edge of V, c_i is the capability of the edge (u_i, v_i) in the set δ(S).

Cluster Node Expansion Capability:

$$c_n(T) = n(T) * \sum_i u_i, * c_i, \quad (4)$$

where u_i is a node in T, c_i is the capability of the node u_i, in the set δ(T).

The scalar and probabilistic measures for the respective expansion capabilities include link bandwidth or availability, as well as node processing power or availability. The scalar measures reflect the capacity capability with respect to cluster expansion, while the probabilistic measures reflect the resiliency capability towards

cluster expansion. Both perspectives are relevant in assessing the dimensional strength of connectivity. Collectively, the four measurements are recursively employed for online state management of a cluster's reachability or connectivity strength.

5.2.2.2 ENVIRONMENT CONSIDERATIONS

In the initial clustering and clusterhead selection process, the above formulations are used to evaluate clustering superiority and reachability– cluster formations and assignments and selection of corresponding DCS clusterhead nodes are provisioned during this process. The process will take several measurement iterations to establish cluster formations and to determine appropriate clusterheads. As reachability is one dimension of the distributed state management framework, the promotional process may be designed with the other state variables (i.e., performance, stability) for appropriate clustering formations and clusterhead selection. It is anticipated that the emergent system will adjust formations throughout the life cycle via the clustering manager with infrequent cluster formation adjustments as the system converges at higher tiers (i.e., through aggregation) of the DCS network.

During steady-state operating conditions, the expansion measures provide real-time state assessment of reachability and connectivity strength to manage cluster-level control. On a temporal level, if node or link dynamics change throughout a period of measurement, assessment is over-weighted to an operational minimum to align with the instability. Additionally, cluster measurements taken at higher HVC levels require longer periodicity to accommodate multiple state management cycles from clusters at lower HVC levels. On a spatial level, nodes in the cluster, specifically those that encompass only clusterheads would not integrate or aggregate expansion measurements. Unlike [197], the emergent process considers expansion measurements independently within a (global) cluster without consideration for assessments made by the (local) clusters (i.e., clusterheads) affiliated and below the same (global) cluster. By

design, the relevance of expansion has only application to the reachability and robustness with respect to the clusterheads (and interconnections) affiliated with the global clusterhead, rather than member nodes associated with the local cluster.

5.2.2.3 EVALUATION

Following the scope and assumptions defined in Section 0, we evaluate the proposed reachability management process. To reduce the implementation complexity, we apply a simpler reachability characterization process in hierarchical fashion across the HVC framework. As such, the following assumptions are taken:

a) A heuristic using simple expansion metrics based on random 2D mobility, range and proximity-based trust sufficiently characterizes a DCS node's *relative* connectivity; further reachability refinement based on the algorithms defined in this section only increases the efficiency or effectiveness of the reachability state management process;

b) All DCS nodes within connectivity range are recursively trusted throughout the virtual hierarchy;

c) DCS nodes share their expansion measures with physical and virtual neighbors, and truthfulness is managed via reputation and history;

d) DCS nodes are assigned to a single clusterhead, align reachability measures to their mesh neighbors and assigned virtual clusters, but can transition state and structural association across the network at the respective HVC time epochs and pre-existing virtual clusters and tiers, respectively.

e) DCS nodes do NOT have access or visibility to shared network state or HVC structure, which they are not promoted or assigned;

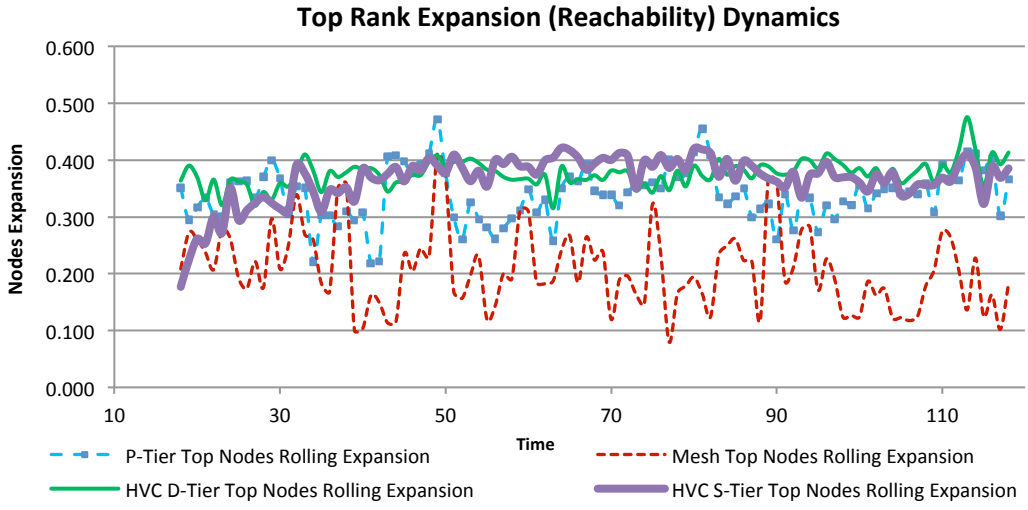
f) Concentrating desired results via top and lowest ranking DCS nodes is sufficient to demonstrate the target evaluation objectives.

As shown in Chapter 4, HVC enables self-organizing structures using virtualization and operational superiority. In this section, our first modeling objective is to demonstrate that the integration of HVC-based structure with an implementation of the reachability management process in DCS will outperform traditional structural models conditioned with the same event model, while capturing reachability (i.e. expansion) state measures at respective network tiers for comparison purposes. The key optimization difference in the HVC-based approach is the use of reachability state rewards to recursively drive the policy-based reinforcement (PRL) subsystem (described in Section 5.3) and reachability-based HVC structures.

We model corresponding network structures with 300 nodes placed randomly in a space of 10000 x 10000 m^2. The adjacency matrix is re-established with each simulation epoch. Nodes are randomly configured with either single or multiple radios operating Wi-Fi (56mpbs) or WiMAX (16mbps) radios configurable with a range of up to 50m or 500m, respectively. Node roles or DCS network capabilities are a function of their dynamic or static ability[18] to support virtual machines (VM), server and WiMAX functionality. Following 4.3.2.1, we model and evaluate each network structure independently during the same modeling period. For HVC configurations, reachability is over weighted to 100% while performance and stability have 0% weighting to ensure the superior reachability nodes influence the desired structural and operational dynamics. We maintain the same event model for each of the structural configurations for respective simulation runs as described in Section 4.3.2.3. Combined node (link) failure events are based on the addition of a simple node failure probability plus range-based unavailability using the node's speed and relative connectivity with other mobile nodes. Alternatively, node utilization reflects a maximum between the node's computational usage and the node's total link utilization. Node movement or speed is calculated using initial position coordinates while next position coordinates are recalculated on each simulated 'second' based on

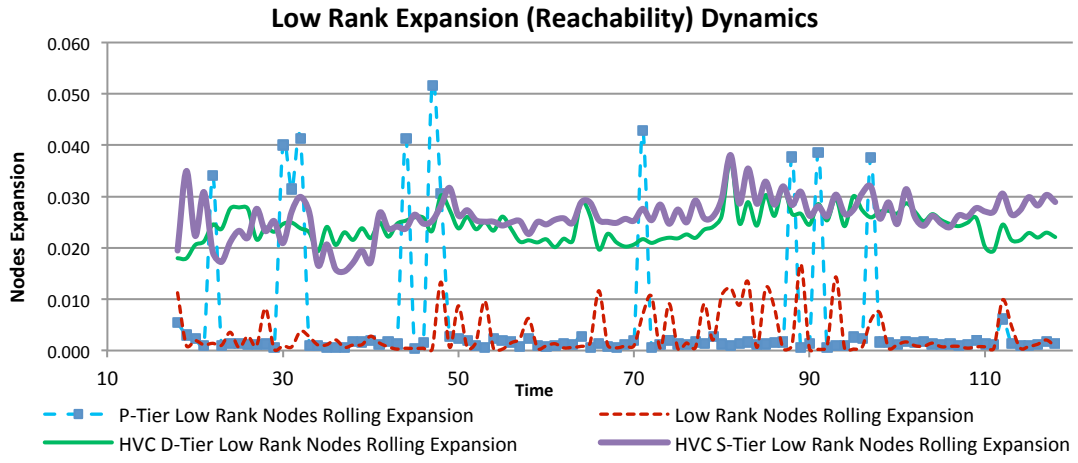
[18] Supported on HVC configurations only, randomly (0,1) reconfigurable every 2 seconds.

a random multiplication factor and randomized (0 to maximum node speed) node speed. Rolling node expansion measures are interval averaged *@modulo2, @modulo8 and @modulo16* second periods at each of the hierarchical operating tiers of the network. The ranking methodology follows the promotional framework described in Chapter 4 Section 4.3.2.6 where physical structures operate on selecting computational and connectivity superior nodes, independent of reachability or other operational variables. The ranking methodology supporting the HVC-based hierarchy operates on elevating dominant nodes that exhibit maximum operational reachability or expansion measures. Figure 55 depicts a comparison of reachability dynamics of the same structures chosen for HVC structural comparison in Chapter 4. As illustrated, both top and low ranking HVC-based nodes have higher cumulative[19] expansion measures and exhibit less perturbation then either the physical tiered and physical mesh configurations. Additionally, a wide proportional (~10x) disparity can be seen between top and low ranking nodes demonstrating the behavior and modeling expectation of promoted nodes having clear operational superiority over nodes exhibiting poor or low reachability profiles.



(a)

[19] Summed cumulatively across top and bottom ranking nodes over time.

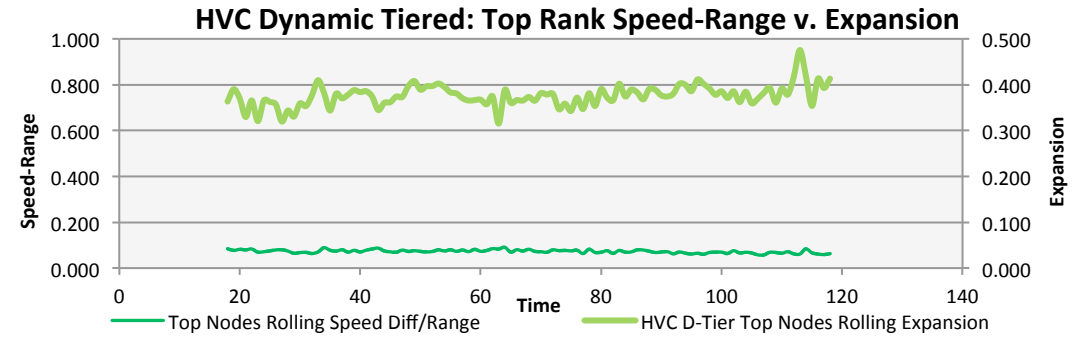


(b)

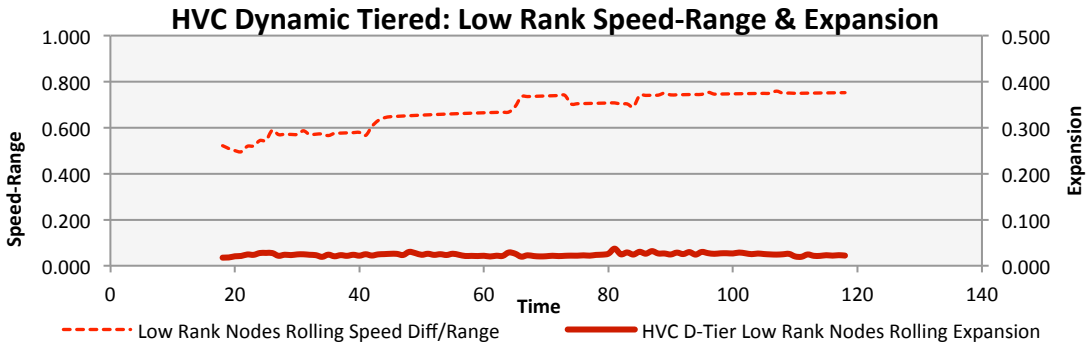
Figure 55: Expansion: Top and low rank nodes

Under the assumptions of an emergent process and treating (modeled) each physical node or promoted virtual cluster node independently to their roles or processes at lower tiers, we demonstrate that a locally-driven (neighbor-based) reachability state characterization process implemented throughout the HVC network hierarchy will promote and differentiate nodes that have higher expansion measures. Simple expansion measures of mobile node speed and wireless range characterize a node's ability to have more global reach, rather than local – in effect, characterizing global DCS reachability. Moreover, the promoted physical nodes or clusterheads will recursively aggregate reachability measures temporally (i.e., aggregated at *@2, @8 and @16 seconds*) and spatially (i.e., cluster members assigned to clusterheads) across their respective virtual clusters. The expansion measures are shared recursively with neighboring clusterhead nodes at each HVC virtual network tier, where the promotional process continues to an eventual root cluster or the most superior reachability node(s), which span the global HVC hierarchy. Figure 56 depicts the speed-range variability against expansion across both top and low ranking nodes. As shown, the speed-range proportion and expansion measures are reverse proportioned thereby demonstrating that higher mobility and lower (wireless) range nodes typically operate with lower expansion measures while lowest mobility nodes with greater

range have higher expansion results. Thus, this confirms the ranking classification and preferred operational behavior.



(a)



(b)

Figure 56: Node speed & expansion

5.2.3 STABILITY STATE MANAGEMENT

Unlike traditional hierarchical networks, where topology, traffic control or networking nodes have a more centralized configuration or static functional structure, the representation of these same concepts in DCS is more dynamic, loosely structured and decentralized - creating a more chaotic environment for network control and management. With an emergent approach, the system addresses these issues with minimal operator dependency, addresses the complexity through an organized clustering framework, and manages the uncertainty through statistical computing methods with learning and the build-up of distributed knowledge.

One critical aspect of uncertainty is the notion of network stability – recognizing and capturing the level of stability or instability in DCS is a key focus of this work. Discovering dependable and available network services, nodes or topological areas for network delivery and communications is essential to scaling the system. Hence, an important contribution of the emergent work is the development of a distributed service capability that continuously manages network availability or dependability for stability assessment. Our approach follows similar entropy-based formulations used in several works [162], [165], [166] to characterize the level of uncertainty or diversity exhibited by respective clusters in the DCS environment. More specifically, entropy evaluates real-time for each node and cluster with further aggregation as the emergent stability process progresses up the cluster hierarchy or the broader network. Thus, the areas of the network that have achieved a level of stability have lower entropy values and therefore, can enable for more dependable service delivery or network control.

5.2.3.1 SUBSYSTEM DESIGN

Within the context of the clustering framework, we measure stability in two ways. First, each cluster can reach a level of stability, independently, to the broader portions of the network by ensuring a relatively uniform level of operational disparity with other nodes or clusterheads *within* the same cluster by maintaining a static topology configuration and operational uniformity across the cluster during one or more measurement cycles. Alternatively, external node dynamics can reduce the level of instability, either through the reduction (*departure*) of unstable nodes or increase (*arrival*) of more stable nodes, relative to the other nodes or clusterheads within the same cluster. Thus, it is essential that the entropy formulations consider both dynamics – imitating the equalization and dissipation behaviors of both isolated (intra-cluster) systems and open systems (inter-cluster). **Error! Reference source not found.** depicts the scenario where the cluster's sub-network does not experience external influences to its topological configuration while **Error! Reference source**

not found. depicts the scenario where DCS nodes are entering (green) and departing (red) the environment. In both cases, the cluster's aggregate entropy reflects a level of equilibrium based on their relative entropy measures between the nodes.

To characterize a cluster's stability or entropy, the key measures of interest are the relative availability or reliability between cluster nodes - the general term *dependability* describes this. Alternative measures may be used either independently or in combination to represent the relative dependability between cluster nodes, or *node-to-node service dependability*. As shown in Table 6, service dependability can have multiple stability or environment constraints with local node and link impact, but more importantly on applications and network services that traverse multiple DCS nodes. From the standpoint of the (emergent) cluster, we are interested in characterizing the level of dynamics or stability across the cluster given such constraints. Therefore, one can use a combined metric to characterize the degree of service [138], [141], [181] dependability. One may also choose to be selective to the most critical constraint (e.g., node availability) in characterizing cluster. Further, some metrics (e.g., link quality or distance) may require averaging neighbors' observations or pair wise assessments, for example, to assess a node's dependability, since there is no way for the node to calculate its own dependability. Alternatively, battery power, which has a common reference, would allow any node to formulate its dependability assessment independently. In any case, a probabilistic substitute of the varying measurement approaches is a simple method to convert the alternative metrics into a consistent variable for the stability analysis below.

Table 6: Service dependability influences

Stability Constraint	High fading conditions	Excessive load or congestion	Component failures	Battery limitations	Range
Δ Metric	Loss rate	Utilization	Availability	Power level	Distance (signal strength)

In formulating cluster-level entropy and following the works of [163], [183], an entropy period between t_1 and t_2 of ΔT,, and $D_{m,n}(\Delta T)$ is defined as the average relative dependability of node m with respect to node n at time t_2, where m and n are

both members of the same cluster $C_{i,j}$. Cluster nodes exchange relative dependability samples, $d_{m,n}(\Delta\tau)$ during multiple $\Delta\tau$ intervals within the entropy period, ΔT. More specifically, both measures are defined below with the goal of eventually defining an entropy formulation for cluster, $C_{i,j}$. The sample relative dependability (*Equation 5*) defined at node *m* with respect to node *n* during interval Δt is:

$$d_{m,n}(\Delta\tau) = d_m(\Delta\tau) - d_n(\Delta\tau) \quad (5)$$

Figure 57 depicts the node changes, where the change $\Delta\tau$ reflects the interval of change, and a relative change in the service dependability reflected for both node *m* and node *n* during the interval. The absolute relative dependability (*Equation 6*) defined between node *m* and node *n* averaged over the period ΔT is:

$$D_{m,n}(\Delta T) = \frac{1}{N} \sum_{i=1}^{N} |d_{m,n}(t_i)|, \quad (6)$$

where N is equal to the number of samples in ΔT and t_i is the discrete time at the end of each sample, Δτ.

This is shown in Figure 58, where the particular node of interest n_i is highlighted depicting only a single node-to-node dependability relationship over the period of entropy evaluation. However, the same measures for the other nodes belonging to the cluster, C_{ij}, exchange messages and evaluate with respect to node m_i. The complete (absolute relative) dependability representations for the other nodes in C_{ij} should account for the event space about node m_i.

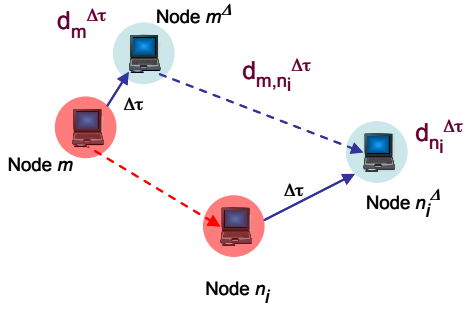


Figure 57: Sample relative dependability, $d_{m,n}(\Delta t)$

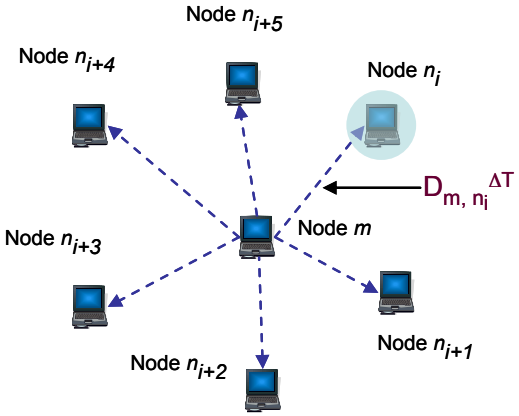


Figure 58: Absolute relative dependability, $D_{m,ni}(\Delta T)$

Now, to characterize stability, we apply a generalized form of Shannon's statistical entropy formulation [162] to the event space using *Equation 7*, specifically:

$$H = - \Sigma_j p_i * \log p_i, \tag{7}$$

where p_i is the probability of the event e_i in the event space E.

Applying this concept to a dependability event space for node m_i and following [165], we set $p_k(t, \Delta T) = D_{m,k} / \Sigma_x D_{m,x,}$ evaluating x over all peer nodes of node m_i in the cluster, C_{ij}, and thus:

$$H_m(t, \Delta T) = - [\Sigma_k p_k(t, \Delta T) \log p_k(t, \Delta T)] / [\log C(C_{ij})], \tag{8}$$

where $C(C_{ij})$ is the cardinality of the cluster.

In general, *Equation 8* calculates the entropy of node m during the specific period of entropy measurement ΔT in cluster C_{ij}, normalized between [0, 1]. This representation of entropy is the desired measure of stability with respect to node m_i, in terms of the absolute relative dependability during the period of stability state evaluation, ΔT. With this formulation, entropy is small when the change in relative dependability variation shows higher perturbation, while a higher value will show more relative stability.

Finally, having calculated the entropy for a single node in the cluster; one can evaluate, similarly, the entropy terms for the other nodes in cluster C_{ij}. To assess a measurement for the cluster's overall stability, *Equation 9* is defined and associated with the clusterhead, C_{ij}:

$$\gamma = [H_{CH}(t, \Delta T)] \quad * \quad \min_{i = (\text{all nodes in } C_{ij} \backslash CH)} [H_i(t, \Delta T)], \tag{9}$$

where CH is the clusterhead for C_{ij}.

5.2.3.2 ENVIRONMENT CONSIDERATIONS

The node level entropy values is one of the state variables used to select optimal clusterheads during the periods of clustering formation based on operational superiority. The above formulations are used to evaluate *cluster-level stability*. Given the measurement cycle requirements, cluster formation will take several measurement periods for promotional assessment towards clusterhead selection. As stated earlier, it is likely the clustering system will adjust formations throughout the life-cycle via the clustering manager, albeit with infrequent cluster formation adjustments, as the system assimilates the aggregate properties of the dynamic network.

However, the γ term will be used by the clusterhead during stable periods of operation towards global control or policy reinforcement as deemed necessary by the specific cluster. The purpose of stability state management is to provide an alternative representation of the clusters' operational state and superiority. Unlike the previous section, which focused on characterizing state for the cluster's global reachability, this section is dealing with the decomposition of the DCS environment into clusters for operational stability evaluation. A highly unstable cluster may not be suitable for many of the higher resiliency aimed applications. Moreover, by understanding these 'pockets' of instability, higher level clusters or clusterheads may enforce or reinforce policies to either thwart traffic away from these clusters or set alternative policies or reconfigurations, which can help to stabilize these clusters over periods of operation.

Alternatively, a service can utilize the distributed stability state to optimally provision virtual machines services on corresponding stable clusters or cluster nodes. Entropy-based stability assessment, while not aimed at specific performance or availability goal evaluation, can be used to baseline global perturbations. This promotes emergent behavior by allowing global operation to agitate local operation for positive or negative adjustment or service delivery consideration.

Unlike reachability, stability management has very different operational objectives and emergent behavior. The reachability process has an external cluster focus; seeking to scale the overall decentralized communication system by promoting operational superiority based on global (external) connectivity rather than local (internal) connectivity. Alternatively, cluster stability focuses inward to the cluster, seeking operational superiority using cluster stability to characterize cluster-level influences. By design, both management objectives are relevant towards enabling an emergent capability, since the distributed management system must look for equilibrium within the local cluster, while maximizing cluster reach and capability for global expansion consideration. However, similarly to the reachability process, if cluster dynamics change throughout a period of measurement, entropy assessment is biased to a statistical minimum to align with the instability. Additionally, hierarchical cluster measurements taken at upper tiers of the hierarchy require longer periodicity to accommodate state management. On a spatial level, the nodes in the cluster do not integrate or aggregate local (child) cluster stability measurements. Thus, stability assessments by respective clusters occur independently to other clusters[20].

5.2.3.3 EVALUATION

The evaluation objectives in this section follow the modeling scope and assumptions defined in Section 4.3. Following the reachability evaluation approach, a simpler

[20] It may be possible to seek statistical methods that achieve both objectives – cluster emergent independence and cluster entropy aggregation.

stability characterization process is applied in hierarchical fashion across the HVC framework. The following assumptions are modeled to support the stability evaluation objectives:

a) A simpler heuristic using availability metrics based on random node & failure rates and mobility sufficiently characterizes a DCS node's availability. Stability refinement based on the entropy-based algorithms defined in this section only further increases the efficiency or effectiveness of the results;

b) DCS nodes share their availability measures with cluster neighbors, and truthfulness is reached via reputation and history in steady-state;

c) DCS nodes are assigned to a single, promoted clusterhead and align stability state management measures to their physical mesh neighbors and assigned virtual clusters, and can shift state and structural association across the network at the respective HVC time epochs and existing virtual clusters and tiers, respectively;

d) DCS nodes do NOT have access or visibility to shared network state or HVC structure, which they are not promoted or assigned;

e) Demonstrating the desired results via top and lowest ranking nodes is sufficient towards meeting the stability management objectives.

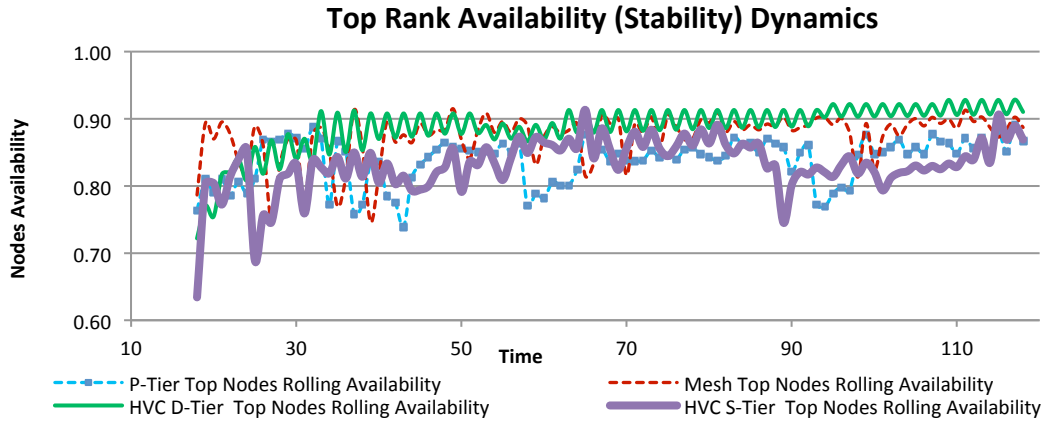
In this section, we demonstrate that HVC integration with a recursive stability (availability-based) management process will outperform traditional structural models conditioned with the same event model, while capturing stability (i.e. availability) state measures at respective network tiers for comparison purposes. The key optimization difference in the HVC-based approach is the use of stability state rewards to recursively drive the policy-based reinforcement (PRL) subsystem and stability-based HVC structures.

We model the corresponding network structures with 300 nodes placed randomly in a space of 10000 x 10000 m^2. Following 4.3.2.1, we model and evaluate each network structure independently during the same 120-second interval. For HVC-based

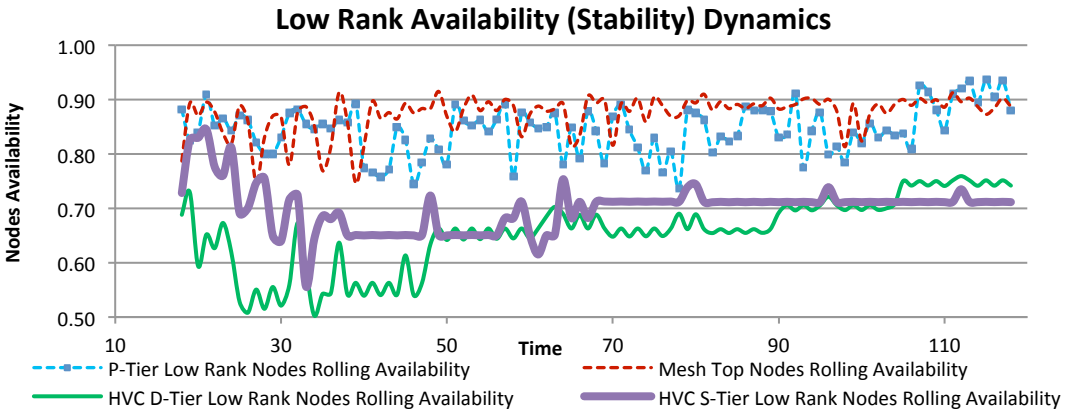
structures, we weight stability 100% while reachability and performance have 0% weighting to ensure dominant stability-oriented nodes influence (i.e., policy-based reinforcement) DCS results. However, we maintain the same event model for each of the structural configurations for respective simulation runs as described in Section 4.3.2.3. Combined node (link) failure events are based on the addition of a simple node failure probability plus range-based unavailability using the node's speed and relative connectivity with other mobile nodes. Rolling node availability is interval averaged *@modulo2, @modulo8 and @modulo16* second periods across the physical or virtual tiers. The ranking methodology follows the promotional framework described in Chapter 4 Section 4.3.2.6 where the physical models operate on selecting computational and connectivity superior nodes, independent of operational stability. The ranking methodology supporting the HVC-based hierarchy operates on elevating dominant nodes that exhibit maximum operational stability or availability measures. Figure 59 depicts an availability comparison against dynamic and static HVC configurations in addition to physical tier and physical mesh configurations. As in the previous section, comparative results are based on statistically equivalent availability-based profiles. As illustrated Figure 59(a), top ranking static and dynamic HVC-based configurations behave quantitatively on average to the physical configurations. The rationale for this may be due to the same nodes[21] being promoted in the virtual configurations. However, there appears to be a rising, (i.e., availability improving) highly cyclic nature associated with the dynamic HVC configuration. The static HVC tiered configuration appears to behave (i.e., cumulative availability) more approximately to the physical tiered structure, which aligns to their respective distributed configuration. Alternatively, the low rank nodes, as shown of Figure 59(b), show greater disparity between the virtual and physical configurations. In addition, the proportional disparities are also apparent with their respective top ranking nodes, which reflect a higher availability expectation for superior nodes than

[21] i.e., virtual node(s) within the same physical node

lowest ranking nodes. As reflected in the top ranking nodes, there is also increasing availability shown in the lowest ranking nodes over time by the dynamic HVC configuration.



(a)



(b)

Figure 59: Stability: top & lowest ranked nodes

Treating each physical node or promoted virtual cluster node as independent processes, a locally driven (neighbor-based) availability state characterization process, implemented in a recursive fashion throughout HVC promotes and differentiates nodes with higher availability measures. Therefore, superior physical nodes or virtual clusterheads, which have higher availability, are viewed as more stable for distributed (i.e., locally or globally) service dependability. Similar to the

reachability process, promoted nodes or clusterheads accumulate availability measures across their respective virtual clusters.

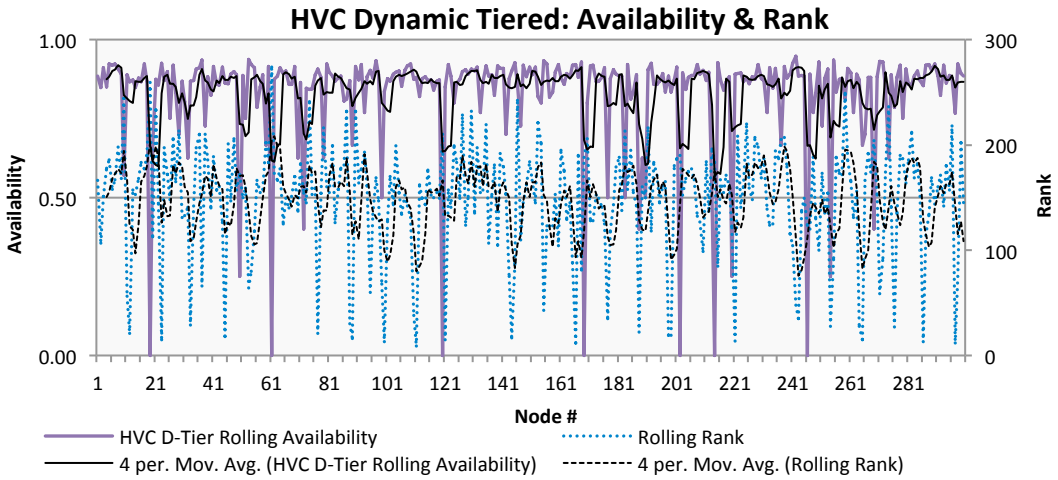


Figure 60: Node rank & availability

Availability measures are shared recursively with neighboring clusterhead nodes at each HVC virtual network tier, where the promotional process continues to an eventual root cluster(s). This represents the operationally superior or highly available nodes, which may serve the global HVC hierarchy. Figure 60 depicts availability (i.e., moving average shown for visual purposes) variability against node ranking across all tiers of the dynamic HVC network. As shown, there is an approximate mapping between higher ranking (i.e., lower number reflects a higher rank) nodes and higher availability metrics and lowest ranking nodes with lower availability metrics.

5.2.4 PERFORMANCE STATE MANAGEMENT

In today' Internet networks, complexity of distributed systems leaves service providers vulnerable to network management and service integrity issues. Many of these issues are obscure performance or reliability problems imposing operational burden and requiring multi-faceted tools to diagnose these issues across heterogeneous subsystems. This challenge has called for [12], [25], [151] more significant shifts to traditional network control and management systems -

specifically, the need to introduce knowledge inference or machine learning approaches. In this section, the focus is specifically aimed utilizing the HVC approach to segregate and organize performance management with a belief development and distribution process for network-wide performance state management. By performance, this is inclusive to fault or reliability conditions, which can create performance integrity issues as well load or congestion-related performance degradation. Thus, the design objectives of an emergent performance management process are to develop a distributed inference framework to enable evidence discovery and adaptive inference for many network-wide performance or reliability anomalies. The benefits include a reduction of manual or operational interaction through distributed belief state reconciliation and prediction of complex anomalies for policy provisioning or automated configuration.

5.2.4.1 SUBSYSTEM DESIGN

A key difference between this service component of the emergent distributed state management and the previous two management services is the focus on propagating state and accumulating distributed performance belief within and across clusters for reconciliation or analysis. Similar to the previous processes in this work, belief state management is asynchronous and multi-threaded across HVC clusters. That is, multiple belief processes are active in global reconciliation or belief convergence.

A key design consideration of DCS systems is the degree of centralization or decentralization in the belief state and the corresponding analytical process for inference and prediction. Using the HVC clustering strategy, this design option is facilitated as the organization of the belief state management process introduces the merits of both centralized and decentralized approaches. Figure 61 and Figure 62 depicts this desired complementary approach to serve both node-level and cluster-level performance belief management, respectively. Specifically, individual cluster nodes or local (child) nodes support local node sensing and inference, while

clusterheads or global (parent) nodes operated similarly. However, the clusterheads will also manage cluster-level state and inference. The local nodes will support a more centralized scope within its node perimeter, while sharing its local evidence or belief with peers for distributed belief reconciliation. Alternatively, while the clusterheads, representing the cluster local area, will orient their perimeter to the cluster in a centralized manner aggregating cluster-level evidence and belief to support distributed sharing across clusters. Both requirements source from the necessity to manage a localization objective (e.g., cluster bottleneck identification and elimination) specific to the cluster, in addition to managing a global network optimization (e.g., network load balancing) objective for end-to-end or network-wide anomaly evaluation.

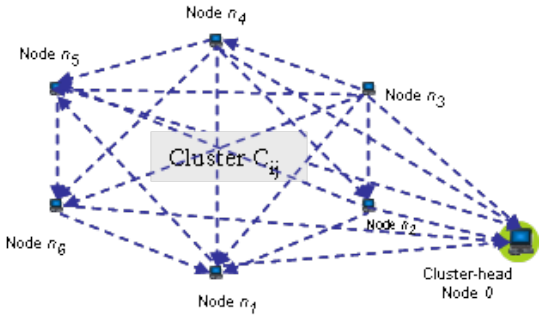


Figure 61: Node-level performance belief management

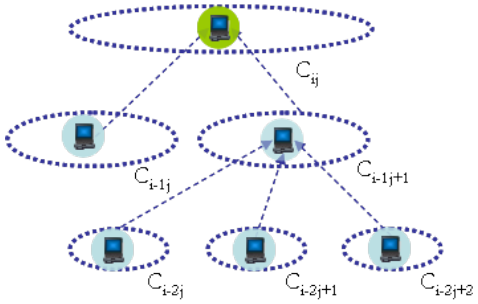
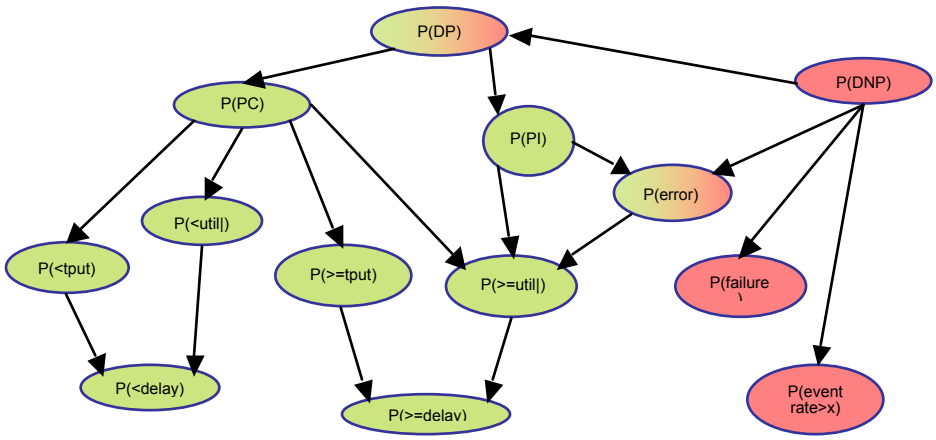


Figure 62: Cluster-level performance belief management

Bayesian Networks (BN) or Bayesian belief networks are currently a popular approach to uncertainty representation and reasoning in numerous network and distributed systems works [151], [169], [173], [175]. As a graphical network, nodes represent discrete or continuous random variables (RV), while edges represent conditional dependencies between RVs. There are two main tasks associated with BN inference: i.) *belief updating* or probabilistic inference, and ii.) *belief revision* or maximum a posteriori (MAP) or most probable explanation (MPE) based on evidence or non-evidence nodes, respectively. Belief updating, which is proposed in this work,

involves calculating *P(X|E)* or posterior probabilities of nodes *X*, given observed values of evidence nodes *E*. Alternatively, belief revision looks to find the most probable state or explanation of some hypothesis variables given observed evidence.

As depicted in Figure 63, BN graphs are typically directed and acyclic (DAG) graphs. There have been several works [172], [177], outlining and debating the challenges of belief convergence within the context of BN graphical models or the applicability of various algorithms on such graphs with cycles. The tactic herein is to avoid elaboration of the convergence subject or the debate with respect to cyclic graphs, choosing to leverage previous progress to facilitate similar objectives.



DP : Did Perform; **PC**: Performed Correctly; **PI**: Performed Incorrectly; **DNP**: Did Not Perform

Figure 63: Example: Dependability Bayes Network graph

Two popular belief propagation algorithms involving probabilistic inference in Bayesian network models are proposed here for HVC-based performance state management, namely the *generalized belief propagation (GBP)* algorithm [174], and the *loopy belief propagation (LBP)* algorithm [172] – categories of approximate inference algorithms. Both algorithms utilize alternative messaging schemes for propagating probabilistic beliefs with the intention of reaching distributed belief

convergence via respective belief update rules. In this work, LBP aligns very well with the organized network clustering structure and state management flow, and thus we propose to employ it for both inter-clustering and intra-clustering performance state management.

Applying LPB for Inter-Cluster Performance Belief

As discussed earlier, there are several works applying the LBP algorithm as an approximation technique in effectively managing inference in Bayesian networks with cycles. Various networking scenarios including sensor networks [175], network fault management [176] and network routing [169] employ this machine learning approach. Utilizing an iterative, message-passing scheme in a graph network model of the target environment, the LPB framework establishes local belief state and utilizes neighbor relationships to disseminate and converge belief for distributed inference. To avoid cycle complexity or slow convergence that are exacerbated in larger network models, HVC is utilized to limit the LBP application to managing belief propagation and message propagation within a cluster or local network scope. This reduces the network diameter and exploits clusterheads for intra-cluster, local (child) coordination and separation for inter-cluster management across the global (parent) cluster. Similar to [173], [177] and as depicted in Figure 64, example performance state[22] interactions of cluster nodes as adjacency nodes with pair-wise interactions are modeled in supporting high level state management objectives. Each cluster node computes a local Bayesian network model as (example) depicted in Figure 65. The computational model characterizes the particular belief state associated with the high-level performance state objective for the local node. Computing the node-level belief state can be done quickly and frequently, allowing sufficient opportunity for cluster-level activities to adjust, synchronize and converge.

[22] This is a generalization of any operational state variable as the proposed methodology can apply to a variety of operational management objectives and BN performance state processes.

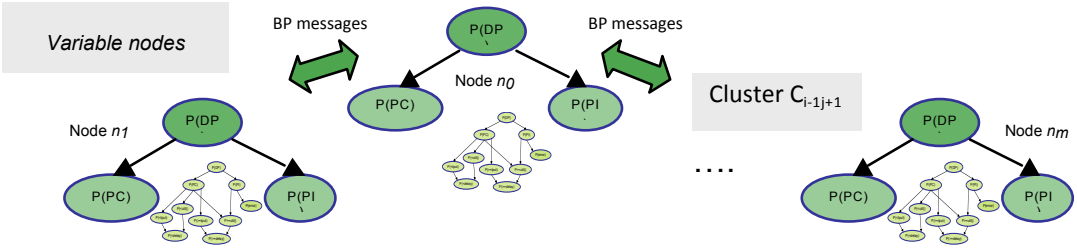


Figure 64: Bayes Network Graphs: Intra-cluster distributed performance belief

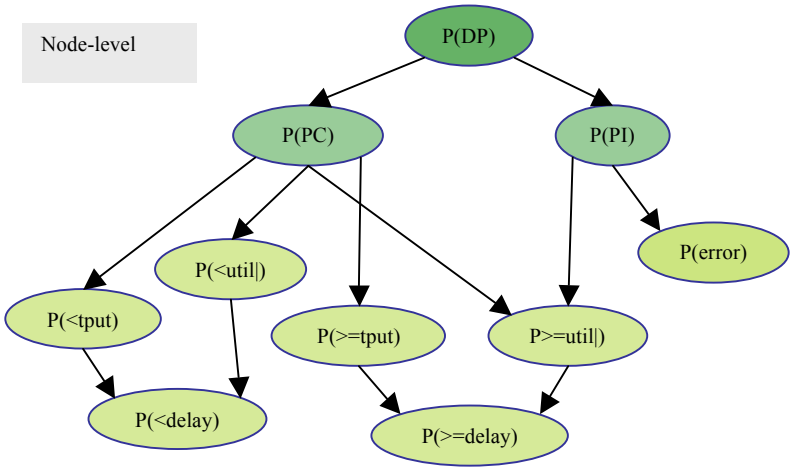


Figure 65: Bayes Network Graphs: Node-level, local performance belief

Each node exchanges their respective belief state with peer cluster nodes, with a compatibility function representing the particular performance state relationship. The LBP framework models the intra-cluster performance belief objective across the set of cluster nodes and their respective local belief computations. Following [174], [177], random variables associated with the respective cluster nodes are labeled as x_i, the high-level BN model computed performance state and y_i, a set of monitoring readings at node i, which is associated with this higher-level performance state. Table 7 presents a typical profile of discrete, node-level state performance readings for belief state interpretation.

Table 7: Probabilistic node performance measures

Cluster Node	Weight (Peer dependency on overall cluster performance)		Performed Correctly	Performed Incorrectly		P(Performed correctly)	P(Performed incorrectly)	P(Did not perform)
			% Utilization	P(<Tput)	P(>=Delay)			
Clusterhead Node 0	50%		75%	1.0	0.0	.98	0.02	0.0
Node 1	Uniform	8.33%	25%	0.4	0.2	.80	0.14	0.6
Node 2		8.33%	25%	0.4	0.2	.80	0.14	0.6
Node 3		8.33%	25%	0.3	0.25	.80	0.14	0.6
Node 4		8.33%	50%	0.3	0.25	.80	0.18	0.2
Node 5		8.33%	50%	0.3	0.1	.80	0.18	0.2
Node 6		8.33%	50%	0.3	0.1	.80	0.18	0.2

The complete joint distribution representing the probabilistic state of the cluster (as depicted in Figure 64 is expressed in *Equation 10*:

$$P(x_1,\ldots x_N) \mid y) = 1/Z \prod_{ij} \psi_{ij}(x_i,x_j) \prod_{i} \phi_i(x_i \mid y_i) \tag{10}$$

where Z is a normalization constant, y are the complete set of discrete performance measures ψ_{ij}, is a compatibility function between nodes *i* and *j*, while ϕ_i represents the effect of the monitor readings on node *i* as computed by the local BN model at node *i*.

Each node *i* sends a message m_{ij} to each of its neighbors *j*, and updates its beliefs b_i based on the messages it receives from its neighbors. We express the update rules as:

$$m_{ij} \leftarrow \alpha \sum_{x_i} \psi_{ij}(x_i,x_j) * \phi_i(x_i \mid y_i) \prod_{k \in N(i)\backslash j} m_{ki}(x_i) \tag{11}$$

$$b_i(x_i) \leftarrow \alpha\phi_i(x_i \mid y_i) \prod_{k \in N(i)} m_{ki}(x_i) \tag{12}$$

where, α is a normalization constant *N(i)* denotes the neighbors of *i*, and *N(i)\j* denotes the neighbors of *i* except for *j*.

Local beliefs for node *i* are reconciled with neighbor messages using $b_i(x_i)$ or *Equation 12,* while node *i* propagates its reconciled beliefs, m_{ij} or *Equation 11* with its cluster neighbor*s j,* with the LBP process replicated at node *j* and continuously across the other nodes in the cluster. The updates are asynchronous in nature, but the

clusterhead can enforce time, message limits, node ordering or other mechanisms to break cyclic dependencies[23].

Applying LBP for Inter-Cluster Performance Belief

In DCS, the hierarchical clustering tree framework is ideally suited to BP algorithms. This may serve any number of network management problems exploiting the unique organization to scale belief propagation, distributed computations for belief update and improve convergence properties across the clustering hierarchy or global DCS environment. To make this clearer, a conceptual example is illustrated in Figure 66 with the higher-level global (parent) clusters and clusterheads highlighted. An existing set of nodes forming the DCS is hierarchically organized into three levels of virtual clusters.

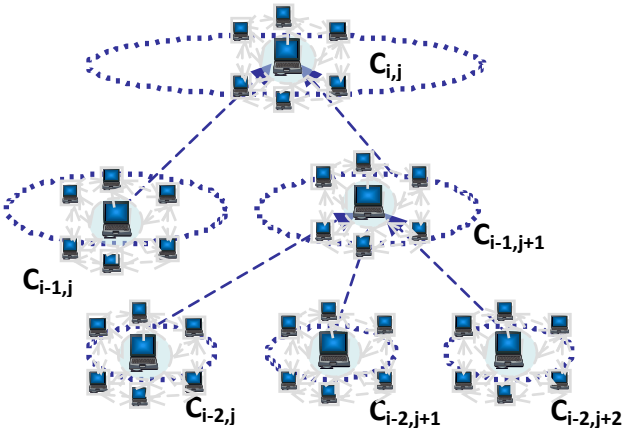


Figure 66: Network centric clustering organization

Applying the LBP framework across the entire DCS environment is debatably infeasible [171],[172], [177] due to the complexity issues discussed earlier. Using the layered virtual hierarchy, the LBP process can be applied at each tier of HVC; enabling independent LBP processes to execute and span the entire hierarchy in a scalable fashion. Since each HVC level or tier consists of multiple clusters formed

[23] Example symptoms of cycles can include lack of belief convergence, minimal changes in distributed belief value or numerous message propagations.

through aggregation, the role of promoted clusterheads, which participate in multiple independent LBP processes over the hierarchy, can serve as belief (i.e., peer-to-peer) gateways or distribution (local-to-global) points[24]. This enables multiple 'virtual dimensions' of performance management, similar to today's processes for real-time, interval or historical performance or capacity management across access, distribution or core networks.

LBP inter-cluster messaging can also occur between peer clusters within an HVC level and across the cluster hierarchy between various local (child) and global (parent) clusters with marginal functions represented as probabilities or indicator functions, depending on the particular belief state management objective. Clusterheads represent the common belief vertices between local, peered or global clusters with clusterheads serving multiple LBP coordination points. Enabling an inter-cluster distributed belief process allows multiple or parallel LPB threads to scale, optimize or balance performance state within an HVC tier, across a graph of one or more clusters or the broader DCS network.

5.2.4.2 DESIGN AND IMPLEMENTATION

Having now defined BP algorithms for both intra-cluster and inter-cluster performance state management, one may ask if there is a necessity to converge these independent processes towards complete state management integration. First, the emergent property positioned in this work suggests this is not required as their asynchronous and independent behaviors may be *cooperative* since the common node to both processes is the clusterhead, and it is possible for this integration to be afforded[25] via local and global state dependencies. Thus, we treat their objectives

---

[24] These concepts are beyond the scope of this thesis but can serve unique models of local-global separation or intelligence or sharing.

[25] However, we have not proven this either through formal methods nor experimental results and the objectives (i.e., global-to-local state integration and network-wide convergence validation) are also beyond the scope of this work.

independently and focus on global (inter-cluster) and local (intra-cluster) performance state management as independent (emergent) activities. This is similar to today's LAN and WAN-based network state management and optimization activities, which can be mostly evaluated and managed independently, with common points of integration and evaluation at the LAN-WAN edge device(s).

Several design considerations of the DCS environment are accounted in the LPB algorithm. First, the degree of intra-cluster and inter-cluster mobility in the LPB belief management processes may not only perturb belief convergence but may dictate which nodes are allowed to participate or whether their beliefs are accounted. In both scenarios, the algorithms will follow similar BP procedures of initialization, belief update, propagation and convergence. In this work, we assume nodes entering or departing in the performance belief algorithms do so on corresponding cyclic boundaries, rather than iteration phases of the LPB algorithm. That is, mobile nodes entering or fleeing clusters will generally be excluded from both inter-cluster and intra-cluster performance belief state management procedures given their instability and correspondingly will be unaccounted for in making local or global policies changes affecting cluster-level or global network performance control respectively. However, as noted in the previous section on stability state management, these nodes will be accounted in assessing stability via entropy, which node mobility will certainly have an effect.

One of the challenging aspects of belief propagation systems is the speed of convergence. The use of the clustering strategy simplifies the complexity of convergence by separating local convergence and intra-cluster belief state management from global convergence and inter-cluster belief state management. In addition, this also increases management messaging (volume) efficiency through the organized cluster hierarchical virtual structure.

Similarly, to the previous sections, provisioning assignments and selection of corresponding clusterhead nodes require multiple state management assessments.

Performance belief values are used to select performance-optimal clusterheads during the periods of clustering formation based on performance superiority. Again, given the measurement cycle requirements, cluster formations will take several measurement periods to reach hierarchical clustering formations and optimal clusterheads.

5.2.4.3 EVALUATION

The objectives in this section follow the modeling scope and assumptions defined in Section 4.3 and a similar orientation to the reachability and stability evaluations. The following assumptions are modeled to support the performance management objectives:

- Node and link utilization metrics is a sufficient measure to characterize alternative performance management objectives; shared performance state within HVC clusters and across virtual tiers converges appropriately;
- DCS nodes are assigned to a single, promoted clusterhead and align performance state management measures to their physical mesh neighbors and assigned virtual clusters, but can transition state and structural association across the network at the respective HVC time epochs and pre-existing virtual clusters and tiers, respectively;
- DCS nodes do not have access or visibility to shared network state or HVC structure, which they are not promoted or assigned;
- Demonstrating the desired results via top and lowest ranking DCS nodes is sufficient towards meeting the performance evaluation objectives.

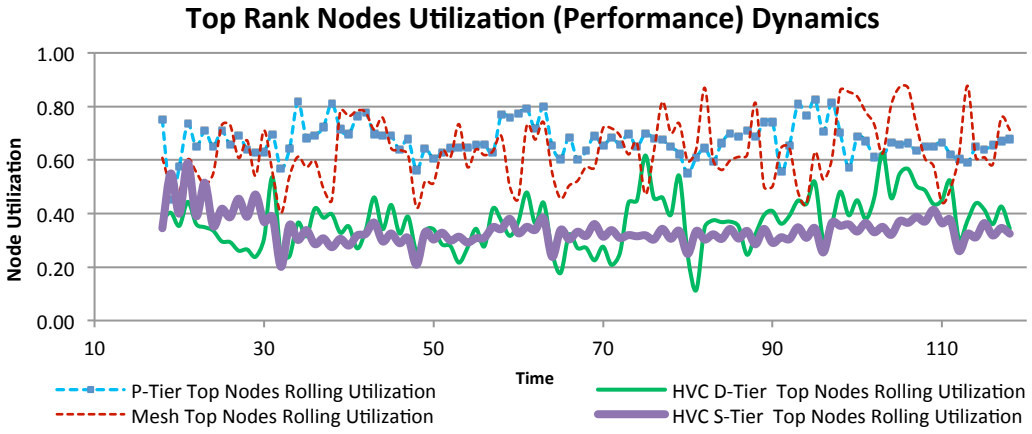
Following similarly with the previous sections, our goal is to demonstrate that the integration advantages of an HVC network with a utilization-based performance management process will outperform traditional structural models conditioned with the same event model, while capturing performance (i.e. utilization) state measures at

respective network tiers. The key optimization difference in the HVC-based approach is the use of performance-based rewards to recursively drive the policy-based reinforcement (PRL) subsystem and performance-based HVC structures. Treating each physical node or promoted virtual cluster node independently, a performance-based characterization process implemented in recursive fashion promotes and differentiates nodes with lower utilization measures. The nature of an emergent DCS system elevates performance superior nodes to higher ranks, and thus, these nodes incur higher demand or resource utilization over time.

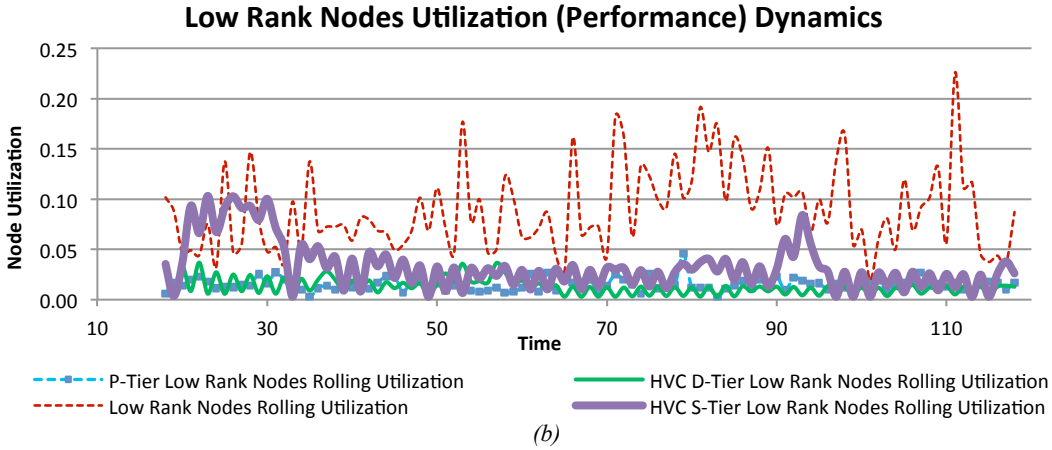
As in the previous sections, we again model corresponding network structures with 300 nodes placed randomly in a space of 10000 x 10000 m^2. Nodes are randomly configured with either single or multiple radios operating Wi-Fi (56mpbs) or WiMAX (16mbps) radios configurable with a range of up to 50m or 500m, respectively. Node roles or DCS network capabilities are a function of their ability to support virtual machines (VM), server and WiMAX functionality. Following 4.3.2.1, we model and evaluate each network structure during the same period. For HVC modeling runs, performance is over weighted (i.e., structurally and operationally) to 100% while reachability and stability have 0% weighting to ensure performance-rich nodes influence the desired modeling results. Node utilization reflects a total maximum between the node's computational usage and the node's total link utilization at respective physical or virtual tiers that the node operates. As described in the previous sections, similar event profiles for availability and reachability operational vectors are maintained as described in Section 4.3.2.3. Rolling node utilization measures are interval averaged *@modulo2, @modulo8 and @modulo16* second periods at respective tiers. Again, the ranking methodology follows the HVC and physical network frameworks described in Chapter 4 Section 4.3.2.6.

Figure 67 compares the utilization performance of HVC configurations against the same physical configurations. As illustrated in Figure 67, top ranking static and dynamic HVC-based configurations behave cumulatively on average at a lower

utilization than the physical configurations. Alternatively, as shown in Figure 67, lowest ranking (both HVC configurations) nodes operate below the utilization of the physical mesh configuration, and the dynamic HVC operates at or below the utilization measures of the traditional physical tiered configuration. The apparent usage disparities are much closer between respective top and lowest ranking nodes across the virtual configurations than the physical configurations, along with a more stable and controlled usage in the dynamic HVC case. Thus, a more balanced resource management environment is evident or at least verified by the extreme DCS nodes under observation.



(a)



(b)

Figure 67: Performance: top & low rank nodes

We focus next only on the Dynamic HVC structural scenario to highlight the discussion on the influence aspects of node physical resource superiority and operational superiority. Figure 68 illustrates node compute and networking resource capabilities against node performance and correspondingly operational hierarchy. The first observation is the direct relationship between the resourcing capabilities of the node and its tier placement in the hierarchy. Additionally, the performance utilization behavior is not consistently associated with node resourcing capability. This aligns with the HVC promotional framework of operational superiority rather than resource superiority. This is consistent with what one expects to see in physical hierarchical networks or computing hierarchies. Given (virtual) nodes are likely to participate in multiple tiers; these nodes are likely to experience higher demand and utilization. However, it is not readily apparent from the graph that one can draw a direct parallel between hierarchy and utilization given this opposing dynamic.

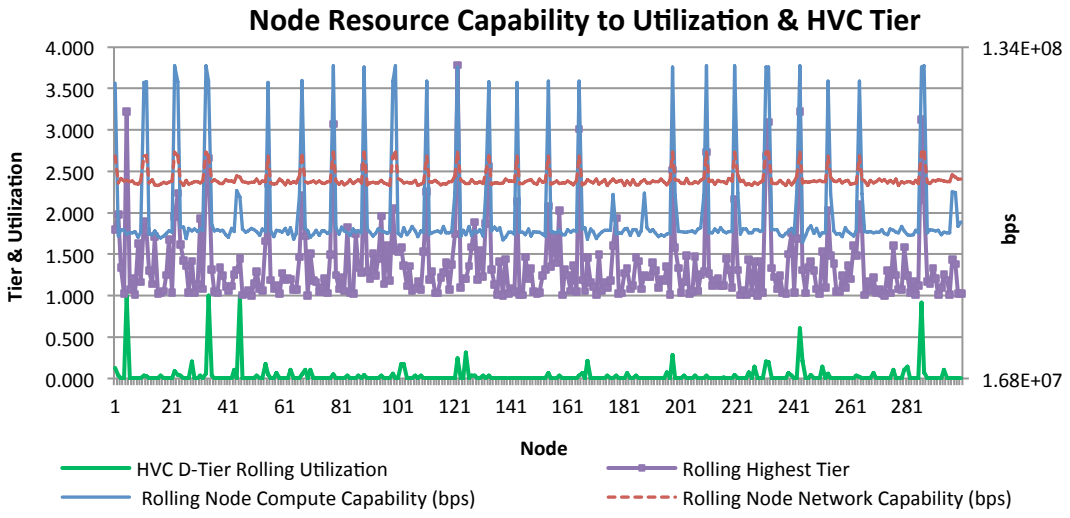


Figure 68: Node resource capability & utilization

Figure 69 depicts an alternative view of node utilization against operational changes and tier dynamics. The purpose here is to reflect DCS operational variability influencing rank and superiority, rather than node resource capability as depicted in Figure 68 previously. The highlighted data points reflect higher utilization predominantly associated with lower change variability, and again, an approximate

mapping between the higher tier nodes and higher utilization metrics. The performance correlation from the two graphs, however, is not easily apparent, and it is likely that a combination of both resource and operational superiority influence the demand vector and overall hierarchy. As such, this is an issue requiring further research investigation.

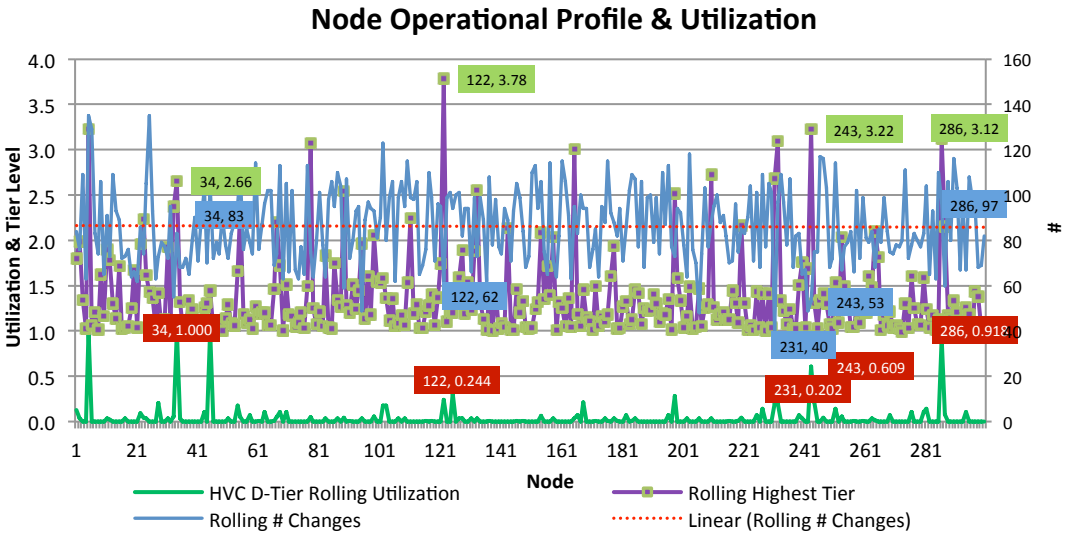


Figure 69: Node operational profile & utilization

5.3 POLICY-BASED REINFORCEMENT LEARNING (PRL)

In this section, the focus is on the control aspect of the emergent framework, employing dynamic policy-based management (PBM) through the HVC framework. The area of policy-based management has received notable industrial and research community progress [152] with various applications including QOS, security and other enhanced network services. One drawback of PBM systems is the degree of operator attention and direction required to evaluate and evolve the system for policy or configuration to adapt to a rapidly changing environment. While there is good progress extending the state of the art towards automation capabilities [26], [27], [49], [99], [130], current policy-based management systems are ineffective in managing decentralized systems.

Reinforcement learning in distributed robotics and in networking [153], [158] is widely researched. In [21], ants swarming techniques use pheromones to reinforce optimal routes, while the directed diffusion [156] work used rule-based reinforcement techniques. Figure 70 illustrates a reinforcement learning conceptual model [154]. The model consists of a discrete set of environment *states*, *actions* and reinforcement signals, namely quantitative *rewards or penalties.* Typically, agents acting on behalf of a local subsystem take actions. In the emergent cluster, the RL agent is a node process facilitating policy-based actions on behalf of the clusterhead within the cluster environment. The inputs to the agent[26] (or agents) are driven out the environment and reflected as states associated with respective management processes, such as those presented in Section 5.2. The agent or clusterhead provisions a *policy* that maps optimal actions against states reinforced through continuous rewards (or penalties), which the environment or cluster deems appropriate. The agent's actions reflect desired behaviors that systematically increase the long-run value of the rewards and thus, optimal, desired behavior for the environment.

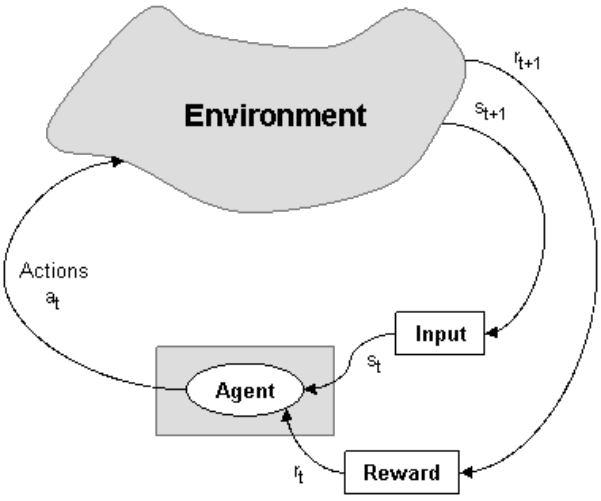


Figure 70: Reinforcement learning model

[26] It is possible to employ one more agents towards driving a single PRL objectives or separate PRL objectives

In a DCS emergent system, reinforcement learning methods facilitate policy-based provisioning by employing the well-known distributed Q-learning methodology [153], [159] across the cluster hierarchy to dynamically allocate global policy and reinforce state-driven actions. The objective of the state management processes is to evaluate distributed state and influence appropriate policies to affect stabilization, increase reachability or performance objectives. By gathering and synchronizing state management across the hierarchy, policy actions can occur asynchronously at cluster boundaries and within respective clusters through state-based rewards or penalties. Thus, an emergent cluster can influence (asynchronously) cluster stability, connectivity and performance by distributing higher-level policies distributed from global clusters via clusterheads and reinforcing local cluster policies though state-driven rewards (or penalties).

5.3.1 SYSTEM DESIGN

Leveraging the clustering organization, reinforced policies (reward or penalty) descend from higher-level clusters as depicted in Figure 71, enabling a more organized and hierarchical approach to distributed policy provisioning. However, cluster decisions can also be influenced locally by the cluster state, which may be reactive or responsive to the cluster network of local nodes. As depicted in Figure 72, a clusterhead participates in the RL process in hierarchical form for inter-cluster network control and in P2P fashion for intra-cluster network control.

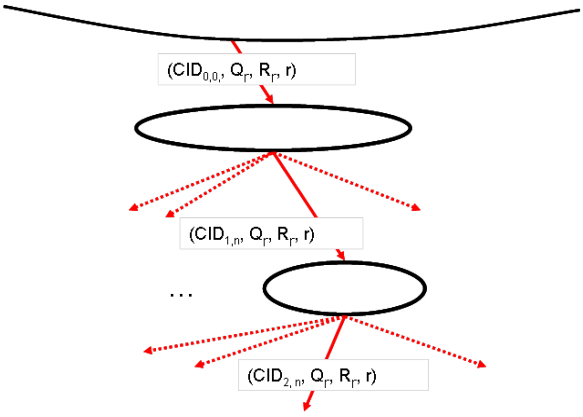


Figure 71: Clustering for PRL

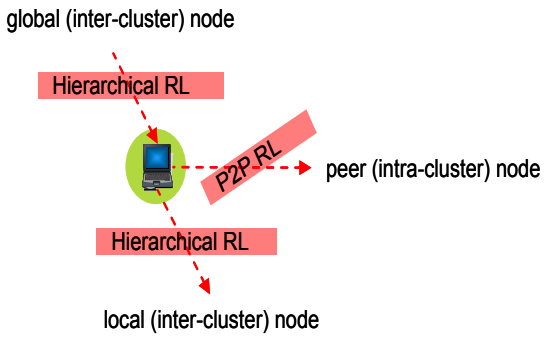


Figure 72: RL clustering strategies

For an emergent cluster to select optimal policies to provision or reinforce actions, it must decide on a model of optimality. First, there are three key state criteria for consideration. Stability state, reachability state and performance belief state, which are associated with the distributed state management processes. Second, specific actions, reinforced by global considerations as described above, can also be driven by local influences if exceptionally provisioned by the clusterhead. Finally, selected actions at a specific asynchronous instance must account for past policy actions, current state conditions, and the predicted affect that an action may have on the cluster's future state. In summary, policy decisions can[27] be made with respect to the clusterhead's balance of intra-cluster future state and expectations on the global network environment in terms of inter-cluster future state. This is analogous to the brain's neocortex function [179], where the human senses independently capture representative state of the surrounding environment, and the brain must reconcile a multiple state representative model of the environment. Accordingly, it can take reactive action in response to sensual state conditions or respond in the abstract based

[27] Adhering to the emergence principle, a node has the design flexibility to treat the local clusterhead PRL process independently to the global PRL process. However, since the (virtual) node is in fact the same physical node, it can share or influence policy (or state) with respective VMs operating within the node's physical structure or autonomy of control.

on previous experience. In most cases, this can take one or more responsive actions or no action at all.

The hierarchical clustering framework and sense-and-respond analogy is similar to the artificial intelligence framework proposed in [179]. Alternatively, a policy-based reinforcement learning system with *multiple threads* of sense-and-respond functions asynchronously shapes the model of optimality. In this work, an online, 'model-free' strategy [154] based on Q-learning applies PRL across the HVC structure(s). The popular Q-learning technique is an incremental algorithm for delayed reinforcement learning sourced from dynamic programming. The standard formulation for the Q-learning algorithm is expressed as:

$$Q(s,a) = Q(s,a) + \alpha(r + \gamma \, \text{Max}_{a'} (Q(s', a') - Q(s,a)), \quad (13)$$

where the *Q(s,a)* function represents the expected discounted reinforcement of taking action *a* in state, *s;* α is a learning rate; γ is the future reward discounted factor and *r* is the instantaneous reward; and finally, the tuple $<s, a, r, s>$ is an experience, representing the transition to state *s'* upon taking previous action *a'*.

In [153], the application of Q-learning, using a simple reinforcement learning formulation, is applied towards optimal packet routing by Q-router agents based on minimizing a *'time-to-go'* function of different packets if routed to a particular neighbor. In this scenario, the iterative RL process is based on capturing packet *'time-to-go'* estimates from neighbor nodes stored in Q-tables to represent possible transition choices for selected destinations along with a packet-delivery time estimate or instantaneous cost reward to reach respective neighbors. By bootstrapping neighbors Q-table values, routing nodes can recursively reward their estimates of a packet's destination time by updating and improving their own estimates through a state-based mechanism, leveraging their neighbors estimates and recursively their neighbors' neighbors estimates. Thus, when suitable Q-values have been learned by the system, packets can then be routed more efficiently by optimally selecting nodes with lowest estimated *'time-to-go'* packets per destination.

Similarly herein, a state-driven technique is utilized for each of the respective distributed state management processes to reinforce or penalize multiple PRL functions. The obvious distinctions, however, is that the cluster process is not specific to a path optimization objective. Alternatively, a generalized PRL process is proposed in similar fashion to reinforce, by state-driven (i.e., measurement basis) reward actions by way of inter-cluster and intra-cluster stability, reachability, and performance state objectives. If implemented, a voting process reconciles multiple reward sources [155] for optimal reinforcement and corresponding action selection. The choice to use singular, state-based reward variables or balanced, multiple state reward variables is a design option that can be configured or programmed based on preferred operational objectives.

Table 8: PRL State-based reward functions

	Intra-Cluster State			Inter-Cluster State		
	Description	Metric	Reward	Description	Metric	Reward
Reachability	Edge Expansion	node % edge count	r = min $node_i$ [#i-links / e-links]	Node Expansion	Cluster expansion	r = Max C_{ii} [n(T)]
Reachability	Expansion Capability	node link capacity	r = max $node_i$ [ΣΣ edge capacity]	Expansion Capability	Cluster capacity	r = max C_{ii} [cn(T)]
Stability	Cluster entropy	node entropy	r = max $node_i$ [$H_m(t, \Delta T)$]	Cross-cluster entropy	Cluster entropy	r = max C_{ij} [γ]
Performance Belief	Cluster Performance Belief	load or failure	r = min $node_i$ [% utilization] r = max C_{ii} [P(failure)]	Cross-cluster Performance Belief	Load failure	r = min C_{ij} [% utilization] r = max C_{ii} [P(failure)]

To setup the PRL process, Table 8 lists the earlier state management variables in matrix format listing the inter-cluster and intra-cluster state rewards and associated metrics. The respective state variables are used to drive a particular intra-cluster or inter-cluster state objective using an *action-state* policy instantiation of a particular management objective. In the previous routing example, a routing control objective and neighbor selection *actions* to a destination were reinforced recursively using neighbor selections based on instantaneous *reward* estimates of packet time-to-go *state* representation.

Table 9: Examples: PRL-based network management & control

Network Management & Control Objective	Reachability	Stability	Performance	Scope	Learning rate: α	Discount factor: γ	Instantaneous reward: r_C, r_S, r_P	Estimate next state value: V
Routing	X	X		Inter-cluster	0.7	0.9	global cluster-head{$r_{c1} = \max C_{ij}[c_n(T)]$ $r_{c2} = \max C_{ij}[n(T)]$ $r_s = \max C_{ij}[\gamma]$}	local cluster-head{$r_{c1} = \max C_{ij}[c_n(T)]$ $r_{c2} = \max C_{ij}[n(T)]$ $r_s = \max C_{ij}[\gamma]$}
Load balancing		X		Intra-cluster	0.7	0.5	cluster-head {$r_s = \max node_i[H_m(t, \Delta T)]$}	cluster {$r_s = \max node_i[H_m(t, \Delta T)]$}
Root-cause analysis		X		Inter-cluster	0.8	0.9	local cluster-head{$r_s = \max C_{ij}[\gamma]$}	local cluster-head{$r_s = \max C_{ij}[\gamma]$}
Service discovery	X			Inter-cluster	0.8	0.9	global cluster-head $r_{c1} = \max C_{ij}[c_n(T)]$ $r_{c2} = \max C_{ij}[n(T)]$	local cluster-head{$r_{c1} = \max C_{ij}[c_n(T)]$ $r_{c2} = \max C_{ij}[n(T)]$}
Path availability	X	X		Intra-cluster	0.7	0.5	cluster-head{$r_{c1} = \min node_i$[#i-links / e-links] $r_{c2} = \max node_i[\Sigma$ edge capacity] $r_s = \max node_i[H_m(t, \Delta T)]$}	cluster{$r_{c1} = \min node_i$[#i-links / e-links] $r_{c2} = \max node_i[\Sigma$ edge capacity] $r_s = \max node_i[H_m(t, \Delta T)]$}
Service processing		X	X	Intra-cluster	0.8	0.5	cluster-head {$r_s = \max node_i[H_m(t, \Delta T)]$ $r_p = \min node_i$[% utilization]}	cluster{$r_s = \max node_i[H_m(t, \Delta T)]$ $r_p = \min node_i$[% utilization]}
SLA management			X	Inter-cluster	0.7	0.9	global cluster-head{$r_p = \max C_{ij}$ [p(failure)]}	local cluster-head{$r_p = \max C_{ij}$ [p(failure)]}

In this work, the proposed PRL framework does not limit the management and control objectives to just routing control, but generalizes the methodology to enable any number of management objectives, and corresponding policy-based actions, which may be suitable to the management objective. Moreover, one may choose to use one or more reward mechanisms to drive or influence the preferred network control actions based on corresponding state management variables. Table 9 lists some examples of how this generalization may be implemented using PRL. As shown, an arbitrary learning rate (α) and discount factor (γ) is depicted along with the inter-cluster or intra-cluster associations that the corresponding objective may be extended.

5.3.1.1 INTER-CLUSTER POLICY-BASED REINFORCEMENT LEARNING

Clusterhead nodes are responsible for coordinating global policy and maintaining network wide objectives in hierarchical arrangement with other global (parent) or

local (child) clusterhead nodes. Figure 73 illustrates this composition with clusters and elected clusterheads distributing global policy in descending progression from the top-level cluster. In this depiction, a single clusterhead is shown at the topic of the hierarchy whereas in practice and for redundancy reasons, we may have greater than one top-level clusterhead or more broadly a *back-up*[28] clusterhead concept throughout the cluster hierarchy. To position a reinforcement learning discussion, Figure 74 presents the algorithmic methodology for inter-cluster PRL for cluster $C_{i-1,j+1}$. As shown, the PRL inter-cluster algorithm calculates state representation metrics from the distributed state management processes supporting reachability, stability and performance and translates these metrics into appropriate rewards for the $C_{i-1,j+1}$ clusterhead acting as the global operator, while receiving pre-calculated state-management reward messages from local or child clusterheads associated with cluster, $C_{i-1,j+1}$. Thus, the global operator clusterhead is positioned to reconcile the rewards as appropriate for the particular network management and control objective. In this work, no particular strategy for reward or reinforcement is dictated, but suggested mechanisms such as voting [158] or hierarchical control strategies (previously shown in Figure 72) are proposed to balance peer or authoritarian control as deemed appropriate for different forms of network or network management objective optimizations (e.g., failure isolation or performance bottleneck removal). Once appropriate rewards are reconciled, selected actions follow the reinforcement strategy and PRL learning framework. The same algorithm is recursively performed (once) by each clusterhead in an asynchronous manner as a PRL service 'provider' while the same clusterhead node may also support reinforcement learning processes for multiple network management and control services as (multiple) PRL 'consumers'. Similar to the other emergent management processes described earlier, the higher level clusters will operate on a more coarse measurement and configuration

[28] The redundancy topic is beyond the scope of this work but an interesting extension of the clustering framework.

boundary then lower level clusters. It is likely that these independent threads may cause instability where dependency is evidenced. This is not unlike the dual control plane subject that challenged the merits of IP QOS and IP routing and prompted the introduction of MPLS QOS-based traffic engineering. Thus, such dependencies must be considered as part of an aggregate control strategy for global DCS management.

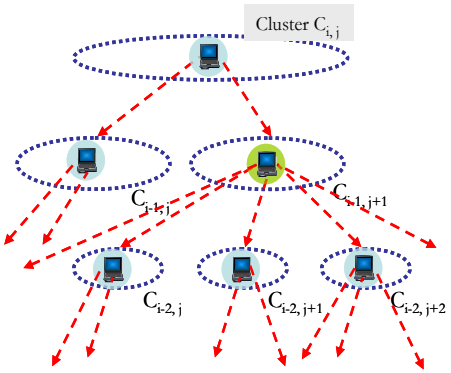


Figure 73: Hierarchical inter-cluster PRL flow

For $C_{i-1,j+1}$ (connectivity, stability, performance)
calculate (r_R, r_S , r_P) for $C_{i-1,j+1}$ clusterhead as a global operator
get (r_R, r_S , r_P) for child clusterheads as global operators
calculate optimal $C_{i-1,j+1}$ reward based on hierarchy strategy *(weighted voting or compromised global v. local concerns)*
select optimal $C_{i-1,j+1}$ inter-cluster action based on maximum Q_{Value} estimate

Figure 74: Hierarchical PRL algorithmic methodology

5.3.1.2 INTRA-CLUSTER POLICY-BASED REINFORCEMENT LEARNING

Unlike inter-cluster hierarchical management, the intra-cluster arrangement follows more closely with a peer-to-peer structure. Using an operationally superior leader, the cluster aims for autonomous control and shared resilience through a peer-to-peer structure. The duality of the inter-cluster and intra-cluster environments are by design emergent; as novelty is demonstrated by their independent control and management processes, while inter- and intra-cluster cooperation is positioned through the clusterhead associations to both. Figure 75 illustrates the secondary role of the same cluster, $C_{i-1,j+1}$ presented above in an inter-cluster arrangement with local (child) clusterheads represented as peer nodes to the clusterhead node n_m. Figure 76 presents the cluster, $C_{i-1,j+1,}$ algorithmic methodology for intra-cluster PRL. Similar to inter-cluster PRL messaging, the clusterhead consolidates reward messaging from respective peer cluster nodes and reconciles the rewards with its own calculated state-

based rewards. This aggregated reward represents the cumulative reward measurement for the cluster. The clusterhead propagates the cumulative reward to cluster peer nodes. The purpose of a common reward estimate for the cluster is to create a sense of 'teaming' and coordinated control.

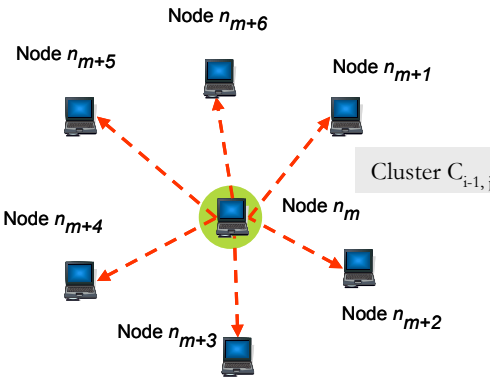


Figure 75: Intra-cluster PRL flow

For $C_{i-1,j+1}$(connectivity, stability, performance)
calculate (r_R, r_S , r_P) for local clusterhead, n_m
get (r_R, r_S , r_P) for local cluster nodes, n_{m+*}
calculate optimal $C_{i-1, j+1}$ reward based on peer-to-peer strategy *(weighted voting or compromise local v. global concerns)*
select optimal $C_{i-1, j+1}$ intra-cluster action based on calculated maximum Q_{Value} estimate

Figure 76: Intra-cluster PRL algorithmic methodology

5.3.1.3 EVALUATION

The policy-based reinforcement learning (PRL) subsystem is the control half of the emergent framework. PRL drives controlled policy and structural change within the modeling framework for DCS optimization. Multiple PRL processes are employed for *stability* through *availability state*, *reachability* using *expansion state* and *performance* through *utilization state* measurements. These distributed processes operate independently over alternative timescales at each HVC virtual tier maintaining the emergence principle of separation. A simple Q-learning algorithm is integrated asynchronously with each of the management processes. PRL processes, reflected in our model as separate reinforcement learning tables, are recursively state conditioned or rewarded using updated state management metrics for each along with their respective neighbors. For each PRL node process, respective (state management) neighbor reward estimates are evaluated and the preferred (maximum estimate) neighbor node is selected. The selected maximum estimate are used to update the

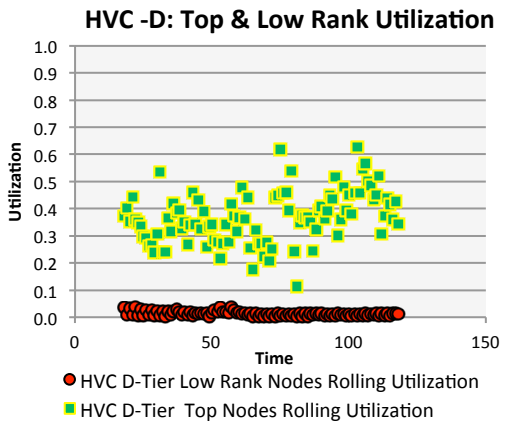
revised PRL table after multiplication of a learning factor, γ^{29}*(@0.6, @0.7 and @0.8)* at corresponding operating intervals *@2sec, @8sec and @16sec*). Physical topology changes are reflected across the PRL tables or reward profiles first via the 2sec PRL tables (i.e., reachability, stability, performance) and then statistically correlated with the longer timescale PRL tables using conditioned rewards as described previously for respective virtual tiers and corresponding operating intervals. As described, one can see that both structural changes and state management optimization are integrated via PRL and thus, creating the HVC-integrated, emergent state management framework.

Following Section 0 and staying consistent with the previous state management modeling assumptions, we maintain the mobility and topology adjacency models, node and event models, application demand profiles and HVC structural models for our PRL modeling evaluation. We assume a uniform (i.e., 33%) weighting framework across performance, stability and reachability variables to ensure uniform operational influence. We model each of the corresponding network structures independently with 300 nodes placed randomly in a space of 10000 x 10000 m^2 and evaluate the simulation results during a 140-second simulation period.

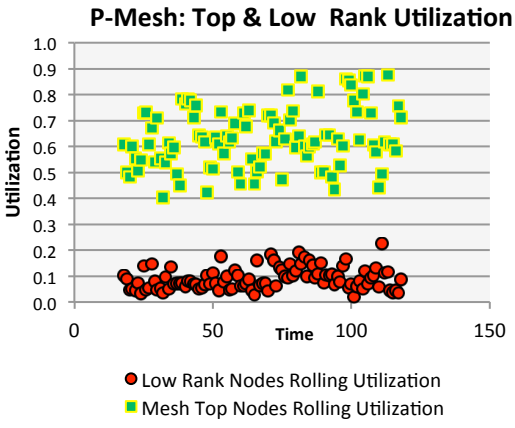
Our first PRL modeling objective is to demonstrate the integration of the HVC framework with the state management processes and the PRL control subsystem. We compare only the dynamic HVC structure against the physical mesh structure. Figure 77(a-f) depicts these comparisons. As shown in Figure 77(a, b), the utilization comparisons demonstrate both a lower and more balanced overall usage profile, in addition to a lower disparity between top and lowest ranking nodes in the dynamic HVC case against the physical mesh configuration. This is due to the structural advantage of the dynamic HVC utilizing the operational superiority framework implemented in HVC to reduce system dependency on suboptimal (lowest ranking)

[29] We give greater weight to future rewards over time and increasing weight across the timescales

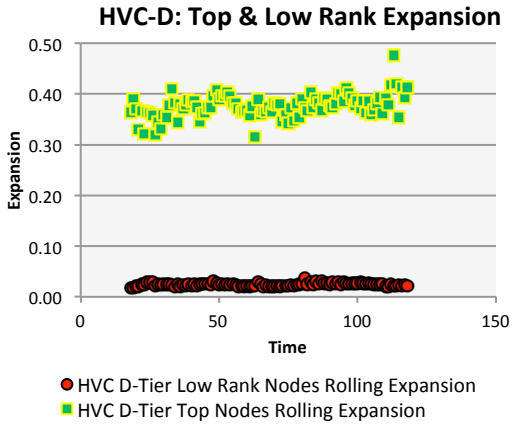
nodes. An analogous discussion can be pursued for the expansion graph in Figure 77(c, d), where expansion or reachability strength, balance and disparities are more prominent for the dynamic HVC case. Similar conclusions can be reached for the reachability scenario, as in the performance or utilization scenario. Alternatively, for the availability graphs illustrated in Figure 77(e, f), the results are more equivalent for the top ranking nodes, but there is a clear advantage in the physical mesh scenario over the dynamic HVC case. Cumulatively, this matches the homogeneous properties that exist in physical mesh structures, while in the dynamic HVC structures; node role and functional properties are more heterogeneous across the multi-tier structures given the inherent virtualization capabilities.



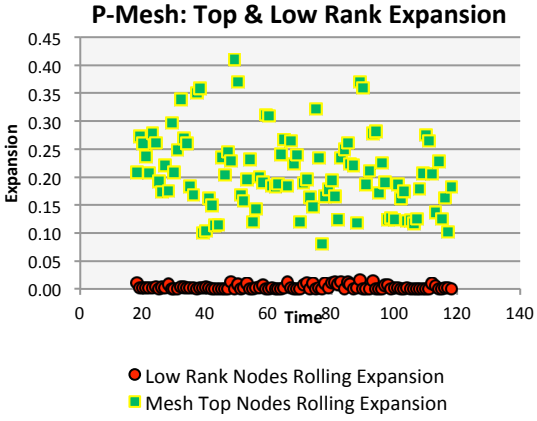
(a) Dynamic HVC utilization



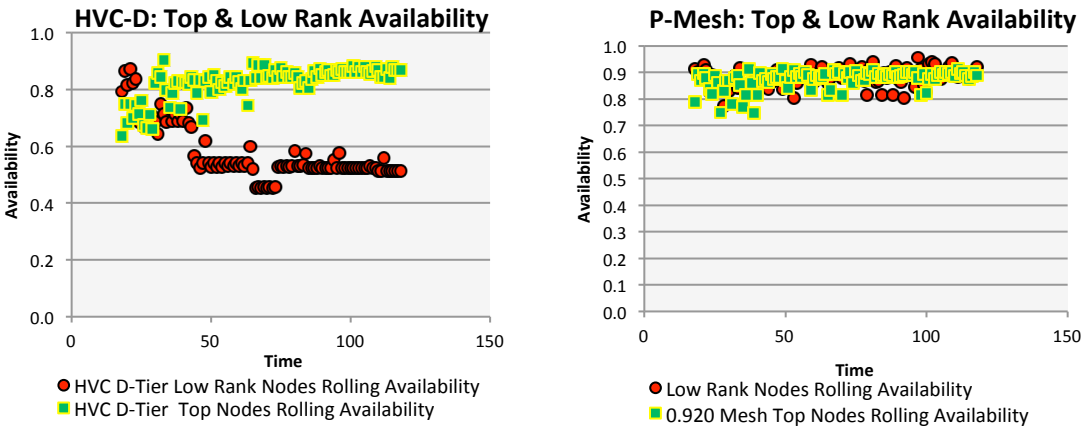
(b) Physical mesh utilization



(c) Dynamic HVC expansion



(d) Physical mesh expansion



(e) Dynamic HVC availability *(f) Physical mesh availability*

Figure 77: Dynamic HVC vs. Physical Mesh operational dynamics

Through PRL, the HVC promotional methodologies and operational management processes are integrated. This integration aligns closely via their respective change dynamics.

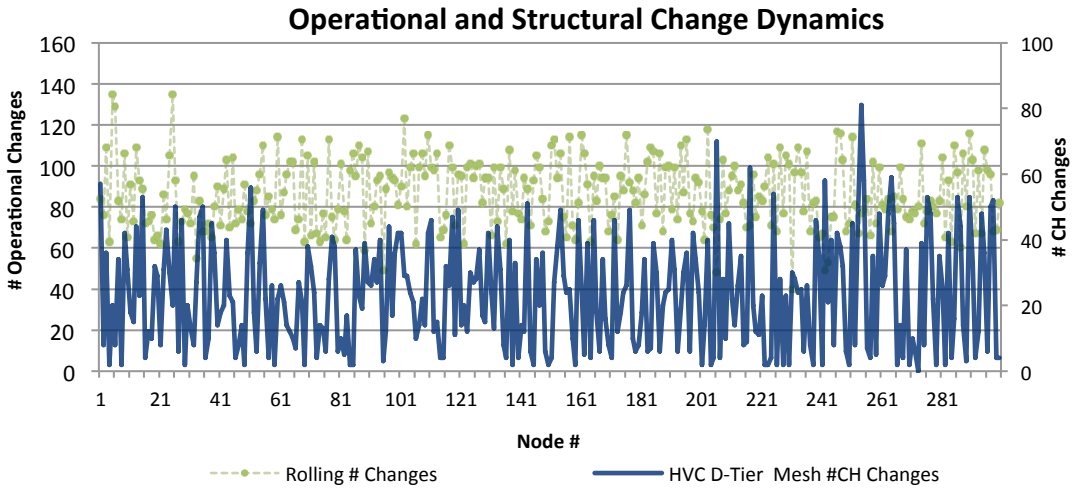


Figure 78 depicts the associated dynamics, and as shown the lower operational dynamics generally align to higher clusterhead changes, which is the desired or expected behavior.

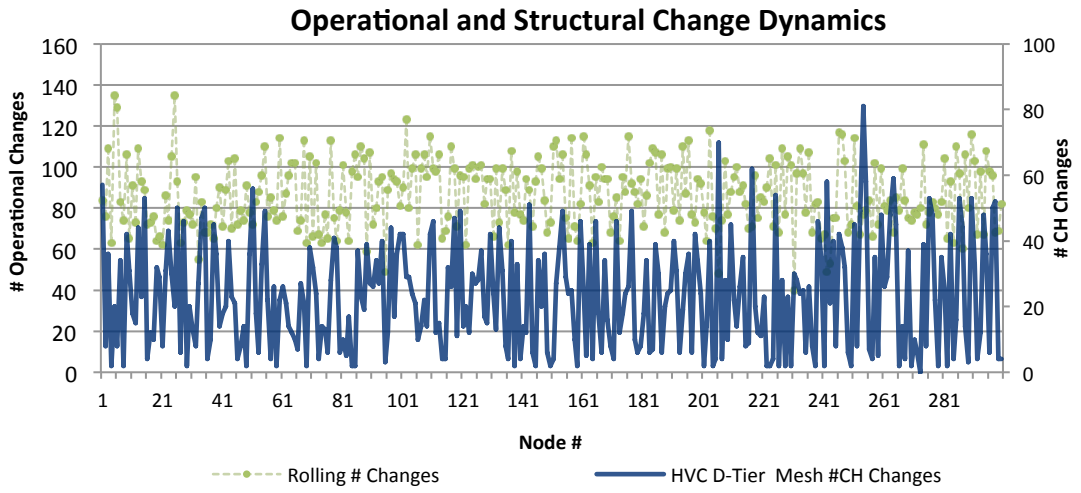


Figure 78: Operational and structural dynamics

Taking a closer look at the dynamic HVC structural and operational behavior, Figure 79 illustrates the convergence properties for both top and lowest ranking nodes. The following observations are visible: i.) there are a higher number of operational changes than structural changes, ii.) the graphs are stable, but the operational dynamics are converging to a stable point, while the structural dynamics are linearly increasing over time, iii) the dynamic HVC graph is reaching a stable convergence point faster than the physical mesh graph in both event types, and finally, iv.) there is greater disparity between the top rank nodes and lowest ranking nodes with respect to the ratio of operational to structural changes. While these observations match the behaviors anticipated, the operational convergence may be due to modeling constraints or because clusterhead changes must continue at rate proportional to the operational changes.

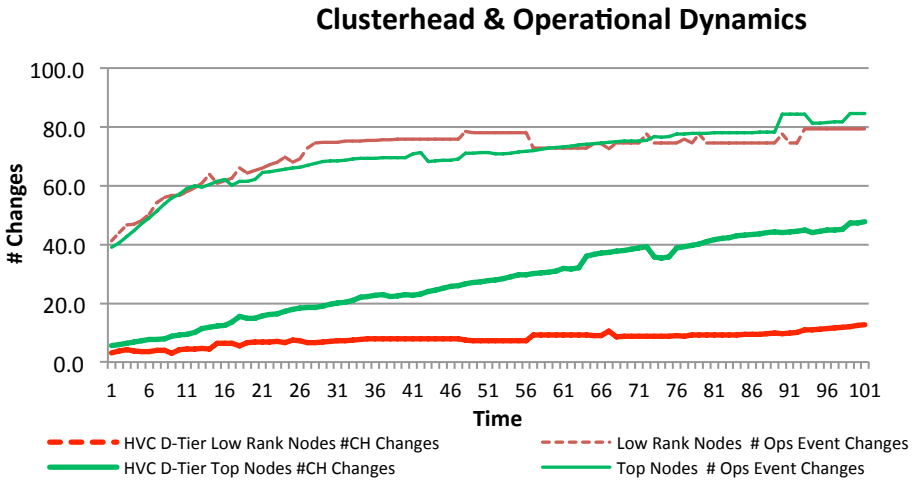


Figure 79: Convergence: Ops & CH events

In Figure 80, we can see a higher but steadier rate of clusterhead changes in the dynamic HVC case in comparison with the physical mesh plots. Furthermore, the physical mesh clusterhead changes, albeit primarily for first tier mesh node selection, occur at a much higher event rate than the dynamic HVC for the top ranked nodes, while the reverse is true for lower ranking nodes. These points demonstrate the HVC structural advantages as well as the operational superiority methodology for emergence that are absent in physical mesh configurations.

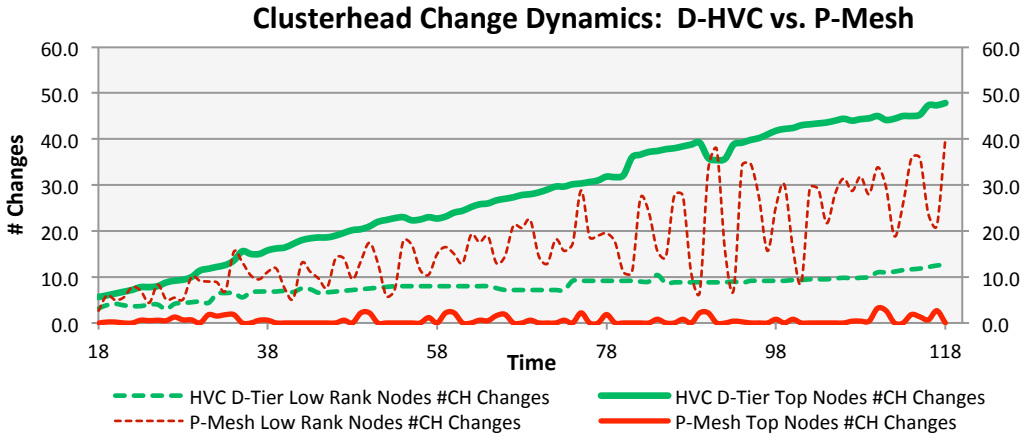


Figure 80: Dynamic HVC vs. Physical mesh promotions

Chapter 6

6 CONCLUSION

This thesis addresses the problem of scaling decentralized communication systems or DCS. Through hierarchical virtual structures and emergent management capabilities, a redirection is proposed to scale DCS. The thesis is summarized based on a sequential presentation of the motivation and arguments for an alternative approach to building decentralized systems, the author's research and vision, and finally, the key underpinnings of a scalable DCS architectural framework. A more detailed articulation of the major thesis contributions and a discussion of future work for evolving DCS research is presented in this closing.

6.1 SUMMARY

Chapter 1 provided a historical perspective and a user-centric motivation for DCS. The proliferation of alternative wireless radio systems, virtualization technologies and the increasing core computing complexity through Moore's Law enables a redistribution of computing and communications to serve alternative approaches to centralized or physical infrastructure models. The DCS direction proposes to bring the *user inside the network* as a node in the network thereby creating a connectivity structure that matches the peering, social nature of the users and enables a larger number of physical and virtual group formations. Chapter 1 elaborates on the challenges to realize DCS and outlines the major thesis contributions to address them.

Chapter 2 takes an alternative survey approach supporting DCS based on the author's previous research in the areas of programmable networking, virtualization and dynamic resource management leading up to the DCS thesis proposal. Specific challenges in these areas include network customization where traditional network devices lack service deployment flexibility and infrastructure scalability as new services are vertically deployed leading to increased infrastructure complexity and costs. Further,

network scaling and stability issues largely driven by the introduction of wireless technologies and mobility requirements challenge traditional operations and management systems. An examination of these works positions a redirection discussion and the architectural motivation for DCS as a closing section.

Chapter 3 presents the DCS vision and architectural framework, expanding on the need for scalable wireless networks enabled with flexible node customization, virtual network structures and emergent control and management. An experimental (OverMesh) prototype of DCS was developed and presented as early validation of the feasibility of building and deploying virtual infrastructure and decentralized services over infrastructure-less, mobile networks. While not addressed in the OverMesh research, a precursor discussion on the realization challenges of network scalability and stability positions a deeper investigation and evaluation for the subsequent chapters.

Chapter 4 presents a novel self-organizing structure using virtualization to create dynamic virtual networking structures and the notion of operational superiority to shape hybrid structures in a cluster-based hierarchy. In DCS, hierarchical virtual clusters (HVC) form and aggregate at multiple levels of a logical hierarchy. Logical clustering employs virtual machines to instantiate virtualized internetworking structures. Clustering balances the merits of a peer-to-peer organizational strategy and a purely centralized organizational strategy. A clusterhead will act as a virtual access point to facilitate central management coordination within any particular cluster and across HVC operational tiers. By organizing state management and control policies over cluster hierarchies, one can achieve the merits of peer-to-peer networks at the lower portion of the hierarchy and more centralized networking towards the root of the hierarchy. Finally, emergent properties are exhibited across HVC with independent, local behaviors consolidated to respective clusters. Chap 4 also expands on an exhaustive DCS modeling implementation used to analyze the HVC and the emergent management systems presented in Chapter 4 and Chapter 5, respectively.

Chapter 5 introduced the emergent management framework along with the key subsystems for distributed state management and policy-based reinforcement learning, recursively integrated through HVC. These emergent formations seek structured organization to manage system-wide objectives to control or balance stability, performance and reachability across the decentralized environment. The state management subsystem manages cluster-level stability state through entropy-based methods, cluster-level distributed performance state through node cooperative methods and network reachability through graph expansion properties. On the control side, a policy-driven reinforcement learning (PRL) subsystem balances global policy against distributed, local policy actions using cluster feedback and state-driven rewards for reinforcement actions. Network-centric knowledge is maintained in hierarchical fashion in the form of distributed state and control policies across the decentralized system. HVC self-organizing capabilities and the emergent management system combine to introduce a novel approach to building and managing a scalable DCS.

6.2 DCS APPLICATIONS

The potential applications of DCS can be extensive across social networks or ad hoc group formations. Deployment examples include group collaboration in business settings, battlefield networks, ad hoc virtual classrooms, community or gaming networks and first-second responder systems networks. One can also envision applications where a ‘flash mob’ of mobile devices and possibly a larger number of virtual machines cooperate to enable computing farms or supercomputing clusters. In these scenarios, networking structures are formed socially through people-centric endpoints. User applications or network services are hosted and delivered in trusted and peer-to-peer fashion. One or more root nodes may facilitate tethered or portal communications through hardened services infrastructure or wide area connectivity. Further, centralized control and operations may be limited or absent, and as such, nodes may rely on each other for repair and recovery.

The unique requirements for DCS include structural speed, ad hoc connectivity, trusted and secure services discovery and advertisement, trust-based group formations and resilience to extreme operational perturbation. Shared services and resources are brokered between peer or tiered nodes using multi-radio wireless networks for communications relaying and cooperative computing. To address these requirements, we feature in the next section how DCS and the core technologies of structural self-organization and decentralized management can be used in first responder networks to deal with catastrophic or disaster recovery scenarios.

6.2.1 DISASTER RECOVERY SCENARIO

One of the most challenging situations for traditional Internet computing and communications infrastructure is associated with catastrophes due to natural disasters or deliberate attacks. During these events, centralized communications and computing infrastructure are typically rendered unreliable, if not completely unavailable. As discussed in [200], distributed (peer-to-peer) networking can play a key role in gathering situational awareness and distributing vital information to crisis managers for a variety of response problems. Scouting and tracking impact with wireless connectivity and highly mobile responders enables greater emergency coverage and shared information. Moreover, information from alternative media such as map images, messaging, video conferencing, statistical information and location-based services can provide for a more comprehensive understanding of a disaster, thereby enabling rapid decision-making. Another key factor in emergency or disaster response systems is the interoperability of different crisis agencies, which may not only need to share vital emergency or recovery data, but may need to ensure network isolation or data protection for privacy or strategic reasons. Emergency response organizations typically have minimal means of communicating with other response or peripheral organizations, for example, police and fire departments may communicate on different wireless radio communication channels or use varying computing capabilities. In [201], the authors

provide a composition of use cases for first responder systems and a methodology for their evaluation. These include:

- *Broadcast/multicast* – group communications, enabling operational control messaging, information or media sharing for wide consumption;
- *Shift change* – team transition in rescue or recovery, managing churn due to new devices, logins or temporary capacity demand increases;
- *Locality awareness* – real-time or persistent information relevant to location-specific information or warnings;
- *Resource awareness* – ad hoc discovery and structural integration of networking or computing hardware pertinent to extend or increase the capacity or resilience of the operational system;
- *Active search* – rapid search and association of an object in contextual (e.g., location) or non-contextual form;
- *Hierarchy maintenance* – enables robust hierarchical decision-making and reporting organization for well-controlled and strict disaster management.

DCS is well suited to meet these use cases and the dynamic requirements of first-responder systems. HVC enables spontaneous structures to facilitate discovery of resources and structural transitions due to incoming and departing work force shifts or organizational change. In a DCS-based emergency response system, user groups can enter or drop out of a system without significantly affecting the overall system stability or performance. HVC supports strict structural hierarchy for both centralized management and reporting, but can also facilitate virtual command & control with limited physical dependency. Emergency response operations rely heavily on robust communications and computing infrastructure. As such, the proposed HVC-based emergent management system architecture is well suited to a distressed environment for operational management & maintenance. Should an emergency task objective or service (e.g., reliable voice communications) be aimed at stable communications or

dependable response, stabile nodes are promoted to manage the stability-based objective. Alternatively, should the emergency task (e.g., search or location discovery) be oriented towards connectivity or reachability, superior reachability nodes are chosen to facilitate wide communications or network discovery. In our first DCS instantiation (i.e., OverMesh), we demonstrated the use of virtual machine overlays to facilitate both search and location-based services. Operational services (e.g., rich content processing or communications) that rely on high performance or load-tolerance should ensure that superior performance-based nodes are selected or weighted towards ensuring successful communications or QOS delivery.

DCS flexibility through HVC facilitates dynamic networking structures, hierarchical command, control and seamless interoperability across multi-group jurisdictions for emergency response. The DCS emergent system enables a diverse framework for serving varying emergency response operations using hierarchical (virtual) structures for decentralized control and management. In summary, the dynamic hierarchical framework and diverse management capabilities of DCS allow crisis managers to rapidly form first responder networks to manage and recover from large-scale disasters.

6.3 THESIS CONTRIBUTIONS

A fundamental principle brought forward in Chapter 1 is the user-based infrastructure concept converged by way of communication decentralization through wireless mesh networking and compute decentralization through virtualization. Through this convergence, a user-centric approach to the design of the DCS infrastructure can serve the socially driven nature of DCS infrastructure – an alternative to the human-designed nature of today's Internet infrastructure. The vision and architecture for DCS is an important contribution presented in this thesis. The OverMesh prototype is an innovative DCS proof of concept, where multi-radio mesh networks and peer-to-peer computing services are combined using distributed virtual machine overlays. OverMesh was developed and successfully deployed as a DCS experimental prototype

in an office setting and demonstrated video/audio collaboration and network search services in a server-less and router-less environment.

In reaching the DCS direction, a breadth of contributions spanning research in network programmability, virtual networking, dynamic resource management and QOS services is assembled in this thesis to draw attention to the continuity and alignment of the author's research. The notable research contributions in the form of architectures, prototypes and publications include:

a. dynamic resource management system architecture to manage spawned virtual networks;

b. distributed, programmable wireless resource management service architecture;

c. programmable network service and device interface model based on a composable and layered API building block methodology;

d. endpoint-based, flexible QOS provisioning API supporting policy-based network traffic control.

A key contribution in this thesis is the hierarchical virtual clustering service architecture. Chapter 4 introduced the cluster emergent service framework, which incorporates HVC clustering with behavioral concepts of emergence for global and local state management. A Cluster Manager has central functionality for the HVC self-organizing methodology - incorporating multi-tier clusterhead promotions, cluster addressing and messaging services. An emergent control flow methodology and cyclic process for cluster aggregation is defined and presented supporting multi-tier, multi-cluster state and policy management.

The DCS modeling implementation is another thesis contribution via an analytical simulation system that modeled mesh-based WiMAX and Wi-Fi networks, diverse physical and virtual network and node models and alternative application demand profiles. The model integrates performance, reachability and stability vectors for emergent state management and policy-based reinforcement learning. The system

recursively models state and policy aggregation through virtual clustering structures over four (4) tiers of HVC. Key HVC research results demonstrated through modeling evaluation include:

a. greater adaptation in HVC-based structures over traditional physically tiered or mesh-based networking structures;
b. operational superiority is a feasible self-organizing methodology to promote HVC-based formations;
c. nodes that exhibit operational superiority are structurally more critical than nodes with superior resourcing capability;
d. virtualization and virtual machines introduce a flexible means to facilitating peer-to-peer, structural tiering and centralization properties models within the same physical structure;
e. Higher messaging overhead is visible in HVC-based structures versus traditional network structures.

The blend of heterogeneous mobile computing devices and wireless access technologies will pose challenges that extend traditional operator models for distributed control and management. Herein was the opportunity for target thesis contribution - to apply more natural and emergent methods for integrated control and management. This thesis proposes a hybrid (peer-to-peer and centralized) hierarchy inheriting emergent properties for managing and controlling decentralized resources for balanced execution and transport. Through virtualization and recursive operational superiority, the two principles combine to enable a multi-level emergent framework. The concepts are analogous to the abstract connectivity constructs, sensory and experience state buildup of the human brain's neocortex as defined in [179]. HVC-based emergent management and control is a novelty of the proposed DCS architecture.

The specific emergent management contributions include distributed service methods for managing and controlling DCS connectivity, distributed performance and stability. A *reachability state management* service uses graph expansion ideas to

characterize DCS local and global connectivity properties. Four measurements are proposed for online state management of a cluster's reachability and connectivity strength. Cluster coefficient measures are defined for edge expansion and node expansion that evaluate a cluster's physical mesh connectivity reach and cluster service-level connectivity reach, respectively. Alternatively, cluster expansion capability measures are used to assess the strength of the cluster expansion in terms of its operational characteristics. The term 'capability' reflects the design variability in a cluster's expansion properties. Scalar measures are defined to evaluate a capacity capability and probabilistic measures to assess the resiliency capability in cluster expansion. Entropy-based concepts are defined to evaluate node and cluster dependability towards managing DCS *stability state management*. Dependability, as a broad operational variable, can be used to assess stability on an array of mobility or dynamic computing problems in DCS. We define measurement sampling variables for relative and absolute dependability. These variables are aggregated towards evaluating node entropy for each node within a cluster or aggregated across all cluster nodes to assess overall cluster entropy or a clusterhead representative measure. Thus, cluster-level entropy reflects aggregate (virtual) node stability in the global hierarchy. Finally, existing loopy-belief propagation (LBP) techniques are defined or reused for local (intra-cluster) and global (inter-cluster) *performance state management* using cooperative computing methods. This may serve a variety of performance or fault management problems by exploiting the unique HVC organization to scale belief propagation and improve belief convergence properties. The hierarchical clustering framework is ideally suited to LBP algorithms - executing and spanning in a scalable, multi-threaded fashion. HVC is used in managing local message propagation within a cluster or local network scope by reducing the network diameter for *intra-cluster* performance belief coordination. Alternatively, HVC clusterheads are used for global networking scope leveraging separation and the virtual hierarchy to manage *inter-cluster* performance belief management across the global network. Collectively, the emergent management services complement the HVC self-organizing framework to

promote nodes and clusters based on operational superiority. The distributed state management concepts capture alternative views of distributed operational state or increase network-centric knowledge. On the control side of emergent management, an HVC-based *policy-based reinforcement learning (PRL)* service is another notable contribution of this thesis. By leveraging the three distributed state management variables for network-centric knowledge, PRL is used to progressively or recursively reward clusters or clusterhead nodes based on preferred state properties for policy control or influence. A dynamic programming (model-free) Q-learning algorithm applies PRL reward actions across the HVC organizing structure to reinforce or penalize inter-cluster and intra-cluster stability, reachability and performance state objectives via singular or multiple state-driven PRL processes.

Through simulation modeling, emergent management research findings that were demonstrated via modeling evaluations include:

a. Operational superiority based individually on performance, stability or reachability or their aggregate leads to superior DCS nodes moving towards the root of the HVC hierarchy and correspondingly less resilient and weaker nodes staying at the lower portions of the hierarchy;
b. Exploiting computational proximity and execution through virtual hierarchy gains and improves communications scale via reduced latency and hop counts;
c. Emergence is exhibited as independent operational behavior at each level of HVC hierarchy through temporal and spatial aggregation and separation;
d. Direct evidence of correlation between load-based events, unavailability-based events, and speed-based events improves performance-based applications, stability-based applications and reachability-based applications, respectively;
e. Policy-based reinforcement learning improves aggregate operational stability and convergence is demonstrated through improved responsiveness of HVC-based structures over traditional mesh-based networking structures.

6.4 FUTURE WORK

The issues of trust and security are fundamental to DCS realization and become increasingly more difficult in a DCS environment [11], [14], [16] with the lack of centralized administration, socially driven communications and related virtualization complexities. The intended objectives in this thesis do not address the broad area of security. Nevertheless, one can anticipate a wide range of industry and academic contributions in DCS trust and security research in the near future. Some examples include remote attestation and authentication, social trust, and secure and protected containers for isolated image execution or data.

Another research topic revolves around cooperative versus selfish behavior or reward incentives versus punishment. The peer-to-peer and social networking communities have addressed similar issues [11], [13], [99], [131]. This subject has direct bearing to the operational issues of efficiency and scalability through increased cooperation and resource sharing. In this work, the topic is lightly touched through organizational hierarchy and reinforcement learning aspects but does not expand on solution alternatives (e.g., game theory or market-driven) of this broad research topic.

The pervasive use of decentralized sensors and actuators located on infrastructure or people are a growth area for both telecommunication and cloud service providers. Machine-to-machine services enable an alternative form of decentralized systems. Evolving DCS and cloud-based research to comprehend human-centric and utility-based models is long-term research given the immaturity of these opposing computing and networking shifts. However, one can envision their convergence for optimal mobile experience.

Virtualization is currently the rage industrial topic for facilitating consolidation, service and resource provisioning and reducing management and security complexity. Virtualization is also a compelling enabler for innovation as demonstrated in this thesis. Node and network virtualization as discussed in Chapter 3 can evolve in such a manner

that node or network structures can be made to be dynamic at deployment time or during runtime. Therefore, research progress towards physical hardware to dynamically compose or restructure compute, storage or network processing could enable higher degrees of robust infrastructure flexibility and integration. This extends to creating schematic views of the hardware; software-defined in a manner that allows the programmer or administrator to design (online) entirely new constructs of the platform, configure or re-configure existing node resource or services in novel ways, lengthening the cycle of physical deployment or provisioning of hardware.

Finally, an elusive concept that this research has only briefly acknowledged is the notion of natural systems or biological influences – for example, alignment to the brain structures, sensory concepts, emergence, operational superiority, cooperative, social or learning influences. The complexity of information technology is reaching heights well beyond human controlled or human designed frameworks. Thus, it is the author's supposition that this research is a nascent call for more biological or natural approaches to information technology creation, delivery and management research through adaptive hardware and self-evolving software.

Chapter 7

7 PUBLICATIONS AS A PH.D. CANDIDATE

7.1 PUBLICATIONS

1. Vicente, J. et al, “Enabling Dynamic Virtual Client Computing with Intel® vPro™ Technology”, Intel Technical Journal, Q4’2008.

2. Ding, G., Vicente, J., “Overlays on Wireless Mesh Networks: Implementation and Cross-Layer Searching”, Published in IEEE/IFIP MMNS’2006.

3. Sedayao, J., Vicente, J. et al, “Virtualization in the Enterprise”, Intel Technical Journal, Q2’2006.

4. Vicente, J. et al, “Toward the Proactive Enterprise”, Intel Technical Journal, Q4’2004.

5. Sedayao, J., Vicente, J. et al, ““PlanetLab and its Applicability to the Proactive Enterprise” Intel Technical Journal, Q4’2004.

6. Krishnaswamy, D., Vicente, J. ““Scalable Adaptive Wireless Networks”, Intel Technical Journal, Q4’2004.

7. Vicente, J. et al, “Automating Traffic Engineering: Towards Intelligent Traffic Engineering”, IT@Intel, 2002.

8. Vicente, J. et al, “Building Intelligent Traffic Engineering Solutions”, Net-Con’2002.

9. Vicente, J., “IP Radio Resource Control”, International Conference on Management of Multimedia Networks and Services, October, 2001.

10. M.E. Kounavis, Campbell, A.T., Chou, S., Modoux, F. and J. Vicente "The Genesis Kernel: A Programming System for Spawning Network Architectures", IEEE Journal of Selected Areas in Communications (JSAC), Special Issue Active and Programmable Networks, 2001.

11. Vicente, J., Villela, D., Campbell, A.T., "Virtuosity: Programmable Resource Management for Spawning Networks", Computer Networks Special Issues on Active Networks, 2001.

12. Vicente, J., Xie, L., Cartmill, H., Anavi, G. “Policy-based Management: Towards Internet Service Provisioning”, Position Paper, Policy’2001, January, 2001.

13. Vicente, J. et al, “L-Interface Specification”, IEEE P1520.3 IP Working Group, January, 2000.

14. Vicente, J., Kounavis, M.E., Villela D.A., Lerner, M., Campbell, A.T. "Programming Internet Quality of Service", Universal Services Market Conference 2000, October 2000.

15. Vicente, J., Cartmill, H., Siegel, S., Fenger, R., Maxson, G., “Managing Enhanced Network Services: A Pragmatic View of Policy-Based Management”, Intel Technical Journal, Q1’2000.

16. Denazis, S., Vicente J., Miki, K., Campbell, A.T., “Interfaces for Open Programmable Routers”, International Working Conference on Active Networks (IWAN'99), Berlin, July 1999.

17. Campbell, A.T., Kounavis M.E., Villela D.A., Vicente J., Miki K., De Meer H.G., and Kalaichelvan K.S., "Spawning Networks", IEEE Network Magazine, July/August 1999.

18. Vicente J., Campbell, A.T., Villela, D. A., "Managing Spawned Virtual Networks", International Working Conference on Active Networks (IWAN'99), Berlin, July 1999.

19. Vicente J., Campbell, A.T.,Villela, D. A., "Virtuosity: Performing Virtual Network Resource Management", International Workshop on Quality of Service (IWQOS'99), London, June 1999.

20. Campbell, A.T., De Meer, H., Kounavis, M.E., Miki, K., Vicente, J., and Villela D. A., "A Survey of Programmable Networks", ACM Computer Communications Review, April 1999.

21. Campbell, A.T., De Meer H. G., Kounavis M. E., Miki K., Vicente J., and Villela, D. A., "The Genesis Kernel: A Virtual Network Operating System for Spawning Network Architectures", 2nd IEEE International Conference on Open

Architectures and Network Programming (OPENARCH'99), New York, March 1999.

22. Campbell, A.T., Katzela, I Miki, K., Vicente, J., "Open Signaling for ATM, Internet and Mobile Networks (OPENSIG'98)", Operating Systems Review, 1999.

7.2 JOURNAL PUBLICATIONS

23. M. E. Kounavis, A. T. Campbell, S. Chou, F. Modoux, J. Vicente and H. Zhuang, "The Genesis Kernel: A Programming System For Spawning Network Architectures", IEEE Journal on Selected Areas in Communications, Vol. 19, No 3, pg. 511-526, March 2001.

24. A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. Vicente, and D. Villela, "A Survey of Programmable Networks", ACM SIGCOMM Computer Communications Review, Vol. 29, No 2, pg. 7-23, April 1999.

7.3 BOOK CHAPTERS

25. A. T. Campbell, M. E. Kounavis and J. Vicente, "Programmable Networks", Reinhard Wilhelm (ed.), Informatics, 10 Years Back, 10 Years Ahead, Springer-Verlag, Lecture Notes in Computer Science 2000, pp. 34-49, 2001.

7.4 MAGAZINE ARTICLES

26. Vicente et al, “OverMesh: Network Centric Computing, IEEE Communications Magazine, February, 2007 Issue.

27. A. T. Campbell, M. E. Kounavis, D. Villela, J. Vicente, H. G. De Meer, K. Miki, and K. S. Kalaichelvan, "Spawning Networks", IEEE Network, Vol. 13, No. 4, pg. 16-29, July/August 1999.

7.5 CONFERENCE AND WORKSHOP PROCEEDINGS

28. A. T. Campbell, S. Chou, M. E. Kounavis, V. D. Stachtos and J. Vicente "NetBind: A Binding Tool for Constructing Data Paths in Network Processor-based Routers", Fifth International Conference on Open Architectures and Network Programming (OPENARCH' 02), New York, NY, June 2002.

29. S. Chou, M. E. Kounavis, A. T. Campbell, and J. Vicente "Genesis Kernel on IXP1200", Concepts and Applications of Programmable and Active Networking Technologies, Dagstuhl Seminar Series, February 2002.

30. J. Vicente, M. E. Kounavis, D. Villela, M. Lerner, and A. T. Campbell, "Programming Internet Quality of Service", Third International Conference on Trends toward a Universal Service Market (USM' 2000), Munich, Germany, September, 2000.

31. A. T. Campbell, H. G. De Meer, M. E. Kounavis, K. Miki, J. Vicente, and D. Villela, "The Genesis Kernel: A Virtual Network Operating System for Spawning Network Architectures", Second International Conference on Open Architectures and Network Programming (OPENARCH' 99), New York, March 1999.

Chapter 8

8 REFERENCES

8.1 INTERNET ARCHITECTURE

[1] M. Weiser, “Some Computer Science Issues in Ubiquitous Computing,” Communications of the ACM, vol. 36, no. 7, July 1993, 74-84.

[2] J.H. Saltzer, D.P. Reed, and D.D. Clark, “End-to-End Arguments in System Design,” http://www.reed.com/Papers/EndtoEnd.html.

[3] C. Christensen, “The Innovator's Dilemma”, Harvard Business School Press, 1998.

[4] D. Isenberg, “The Dawn of the Stupid Network,” ACM Networker, vol. 2, no. 1, February/March 1998, pp. 24-31.

[5] C. Fine, Clockspeed: “Winning Industry Control in the Age of Temporary Advantage”, Perseus Publishing, 1999.

[6] D. Reed, “That Sneaky Exponential—Beyond Metcalfe's Law to the Power of Community Building,” Context Magazine, Spring Issue, 1999.

[7] C. Baldwin and K. Clark, “Design Rules: the Power of Modularity”, MIT Press, 2000.

[8] B. Carpenter and S. Brim, “Middleboxes: Taxonomy and Issues,” RFC 3234, February 2002.

[9] D. Tennenhouse, “Proactive Computing,” Communications of the ACM, vol. 43, no. 5, May 2000, pp. 43-50.

[10] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, “A Blueprint for Introducing Disruptive Technology into the Internet,” in Proc. of Hot Nets I, Princeton, NJ, October 2002.

[11] A. Lippman and D. Reed D, “Viral Communications,” Executive Summary Report, 2003.

[12] D. Clark, C. Partridge, J. C. Ramming, and J. T. Wroclawski, “A Knowledge Plane for the Internet,” in Proc. of SIGCOMM, 2003.

[13] D. Watts, “Six Degrees: The Science of a Connected Age”, 1st Edition, W.W Norton & Company Ltd., 2003.

[14] J. Vicente, S. Rungta, G. Ding, D. Krishnaswamy, W. Chan, and K. Miao, “Network Centric Computing,” IEEE Communications Magazine, 2006.

[15] Roscoe T. “The end of Internet architecture” In: Proceedings of the fifth workshop on hot topics in networks (HotNets-V), Irvine, USA, November 2006.

[16] Su, J., Scott, J., Hui, P., Upton, E., Lim, M.H., Diot, C., Crowcroft, J., Goel, A., de Lara, E.: “Haggle: Clean-slate networking for mobile devices”. Technical report, University of Cambridge, UCAM-CL-TR-680 (2007).

[17] Committee on Network Science for Future Army Applications, Board on Science and Technology, “Network Science”, National Research Council, The National Academies Press.

8.2 EMERGENT & SELF ORGANIZATION & MANAGEMENT

[18] M J Mataric´, "Designing Emergent Behaviors: From Local Interactions to Collective Intelligence", in Proceedings, From Animals to Animats 2, Second International Conference on Simulation of Adaptive Behavior (SAB-92), J-A. Meyer, H. Roitblat and S. Wilson, eds., MIT Press, 432-441.

[19] D. A. Fisher and H.F. Lipson, "Emergent Algorithms—A New Method for Enhancing Survivability in Unbounded Systems," Proceedings of the 32nd Annual Hawaii International Conference on System Sciences, Maui, HI, Jan. 5-8, 1999 (HICSS-32), IEEE Computer Society, (1999).

[20] T. White and B. Pagurek, “Towards Multi-Swarm Problem Solving in Networks,” ICMAS'98, IEEE Computer Society, pgs 333-340, (1998).

[21] I. Kassabalidis, M.A. El-Sharkawi, R.J. Marks II, P. Arabshahi, and A.A. Gray, ``Swarm intelligence for routing in communication networks," IEEE Globecom 2001, San Antonio, Texas, (Nov, 2001).

[22] P. Arabshahi, A. Gray, I. Kassabalidis, A. Das, S. Narayanan, M. El-Sharkawi, and R. J. Marks, II, "Adaptive routing in wireless communication networks using swarm intelligence", Proc. 19th AIAA Int. Commun. Satellite Syst. Conf., (2001).

[23] P. Langley and J. E. Laird, "Cognitive Architectures: Research Issues and Challenges". Draft of October 31, 2002.

[24] K. Hermrmann, “MESHMdl – A Middleware for Self-Organization in Ad Hoc Networks,” in Proc. of the 23rd IEEE ICDCSW, 2003.

[25] N. Foukia and S. Hassas, “Towards self-organizing computer networks: A complex system perspective,” in Proc. of the 1st Workshop on Engineering Self-Organising Applications, Melbourne, Victoria, July 2003, pp. 77-83.

[26] E. Gelenbe, R. Lent, and A. Nunez, “Self-aware Networks and Quality of Service,” in Proc. of IEEE, vol. 92, no. 9, 2004, pp. 1479-1490.

[27] V. Kalogeraki, F. Chen, T. Repantis, and D. Zeinalipour-Yazti, “Towards Self-Managing QOS-Enabled Peer-to-Peer Systems,” SELF-STAR 2004, LNCS 3460, pp. 325-342.

[28] G. Di Caro, F. Ducatelle, L. M. Gambardella, “Swarm intelligence for routing in mobile ad hoc networks”, In Proceedings of the 2005 IEEE Swarm Intelligence Symposium, (2005).

[29] P. Heegaard, O. Wittner, and B. Helvik, “Self-management of Virtual Paths in Dynamic Networks,” SELF-STAR 2004, LNCS 3460, pp. 417-432.

[30] R. De Lemos, “The Conflict of Self-* Capabilities and Predictability,” SELF-STAR 2004, LNCS 3460, pp. 219-228, 2005.

[31] P. H. Van Dyke and S. A. Brueckner, “Analyzing Stigmergic Learning for Self-organizing Mobile Ad-Hoc Networks,” ESOA 2004, LNAI 3464, pp. 195-209, 2005.

[32] Babaoglu et al., “The Self Star Vision,” SELF-STAR 2004, LNCS 3460, pp. 1-20, 2005.

[33] T. De Wolf and T. Holvoet, “Emergence Versus Self-organization: Different Concepts but Promising When Combined,” ESOA 2004, LNAI 3464, pp. 1-15, 2005.

[34] Committee on Frontiers at the Interface of Computing and Biology, “Catalyzing Inquiry at the Interface of Computing and Biology”, National Research Council, The

National Academies Press.

8.3 PROGRAMMABLE NETWORKS

[35] OPENSIG Working Group
[36] DARPA Active Network Program
[37] The Genesis Project: Programmable Virtual Networking
[38] Biswas, J. et al., "Application Programming Interfaces for Networks", IEEE P1520 Working Group.
[39] Lazar, A.A., Lim, K.S. and Marconcini, F., ``Realizing a Foundation for Programmability of ATM Networks with the Binding Architecture," IEEE Journal on Selected Areas in Communications, Special Issue on Distributed Multimedia Systems, No. 7, September 1996.
[40] Lazar, A., "Programming Telecommunication Networks", IEEE Network, vol.11, no.5, September/October 1997.
[41] Kanada Y. et al, “SNMP-based QOS Programming Interface MIB for Routers” Internet Draft, draft-kanada-diffserv-qospifmib-00.txt., Oct. 1999.
[42] Adam, C. .M., et al., "The Binding Interface Base Specification Revision 2.0", OPENSIG Workshop, Cambridge, UK, April 1997.
[43] Van der Merwe ,J.E., Rooney, S., Leslie, I.M. and Crosby, S.A., "The Tempest - A Practical Framework for Network Programmability", IEEE Network, November 1997.
[44] Angin,O., Campbell,A.T., Kounavis,M.E. and Liao,R.R.-F., "The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking", IEEE Personal Communications Magazine, Special Issue on Adaptive Mobile Systems, August 1998.
[45] Campbell, A. T., De Meer, H. G., Kounavis, M. E., Miki, K., Vicente, J., Villela, D. A., "The Genesis Kernel: A Virtual Network Operating System for Spawning Network Architectures", IEEE 2nd International Conference on Open Architectures and Network Programmability (OPENARCH'99), October 1998, pp. 115-127.
[46] Campbell, A.T., De Meer, H., Kounavis, M.E., Miki, K., Vicente, J. and Villela, D., "A Survey of Programmable Networks", ACM Computer Communications Review, April 1999.
[47] Denazis, S., Vicente J., Miki K., Campbell A., “Designing Interfaces for Open Programmable Routers”, July. 1999.
[48] A. T. Campbell, M. E. Kounavis, D. Villela, J. Vicente, H. De Meer, K. Miki, and K. S. Kalaichelvan, “Spawning Networks,” IEEE Network, vol. 13, no. 4, July/August 1999, pp. 16-30.
[49] Vicente, J., Campbell, A. T., Villela, D., "Virtuosity: Performing Virtual Network Resource Management", in Proc. Of IWQoS'99, London, 1999.
[50] Lerner, M., Vanecek, G., Vidovic, N., and Vrsalovic, D., Middleware Networks: Concept, Design and Deployment of Internet Infrastructure; Kluwer Academic Press, April 2000 (ISBN 07923-78490-7).
[51] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. “A Blueprint for Introducing Disruptive Technology into the Internet.” Proceedings of HotNets I. Princeton, NJ, October 2002.
[52] Duffield N., et al., “A Performance Oriented Service Interface for Virtual Private

Networks", draft-duffield-vpn-qos-framework-00.txt. Work in progress.

8.4 QOS & POLICY

[53] Microsoft Reference on Win2K Traffic Control
[54] S. Blake et al., "An Architecture for Differentiated Services," RFC 2475.
[55] S. Shenker, et al., "Specification of Guaranteed Quality of Service," RFC 2212.
[56] R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633
[57] Baker, F. et al, "Management Information Base for the Differentiated Services Architecture"
[58] Rajan, R., Martin, J. C., Kamat, S., See, M., Chaudhury, R., Verma, D., Powers, G., Yavatkar, R., "Schema for Differentiated Services and Integrated Services Networks"
[59] Boyle J., Cohen R., Durham D., Herzog S., Rajan R., Sastry A., "The COPS (Common Open Policy Service) Protocol"
[60] DMTF CIM Schema: http://www.dmtf.org/spec/cim_schema_v24.html
[61] Campbell, A.T., Coulson, G., and D. Hutchison, "A Quality of Service Architecture" , ACM SIGCOMM Computer Communication Review, Vol. 24 No. 2., pg. 6-27, April 1994.
[62] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)," Version 1 Functional Specification, RFC 2205, Sept. 1997.
[63] R. Rajan, D. Verma, S. Kamat, E. Felstaine, S. Herzog, "A Policy Framework for Integrated and Differentiated Services in the Internet", IEEE Network Magazine, Sept./Oct. 1999.
[64] T. Ebata, M. Takihiro, S. Miyake, M. Koizumi, F. Hartanto, G. Carle, "Interdomain QOS Provisioning and Accounting", Internet Draft, draft-ebata-inter-domain-qos-acct-00.txt, Oct., 1999.
[65] B. Moore, E. Ellesson, J. Strassner, "Policy Core Information Model – Version 1 Specification" Internet Draft, draft-ietf-policy-core-info-model-06.txt, May 10, 2000.
[66] Y. Snir, Y. Ramberg, J. Strassner, R. Cohen, "QOS Policy Schema", Internet Draft draft-ietf-policy-qos-schema-01.txt, February 2000.

8.5 VIRTUALIZATION & OVERLAY SYSTEMS

[67] http://www.geni.net/
[68] http://www.vmware.com/
[69] http://www.xensource.com/
[70] http://www.emulab.net/
[71] http://www.linux-VServer.org/
[72] http://www.planet-lab.org/
[73] "Intel Virtualization Technology: A Primer," http://www.intel.com/cd/ids/developer/asmo-a/eng/221874.htm.
[74] M. Day, B. Cain, B. Tomlinson, and P. Rzewski, "A Model for Content Internetworking," Internet RFC 3466.
[75] H. Eriksson, "Mbone: The Multicast Backbone," Communications of the ACM, 37(8):

54-60, 1994.

[76] E. G. Sirer, R. Grimm, B. N. Bershad, A. J. Gregory, and S. McDirmid, “Distributed Virtual Machines: A System Architecture for Network Computing,” in Proc. of SIGOPS European Workshop, 1998.

[77] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao, “OceanStore: An Architecture for Global-Scale Persistent Storage,” Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000), November 2000.

[78] I. Stoica, R. Morris, D. Kareger, F. Kaashoek, and H. Balakrishnan, “Chord: Scalable Peer-To-Peer lookup service for internet applications,” In Proc of the 2001 ACM SIGCOMM Conference, pp. 149-160, 2001.

[79] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, “Resilient Overlay Networks,” in Proc. of 18th ACM SOSP, 2001.

[80] A. D. Keromytis, V. Misra, and D. Rubenstein, “SOS: Secure Overlay Services,” in Proc. of SIGCOMM, August 2002.

[81] J. Dilley, et al., “Globally Distributed Content Delivery,” IEEE Internet Computing, September 2002, pp. 50-58.

[82] “Dynamic Slice Creation,” The PlanetLab Architecture Team, edited by Larry Peterson, October 2002.

[83] A. Nakao, L. Peterson, and A. Bavier, “A Routing Underlay for Overlay Networks,” in Proc. of SIGCOMM, 2003.

[84] Z. Xu, C. Tang, and Z. Zhang, “Building Topology-Aware Overlays using Global Soft-State,” in Proc. of ICDCS, 2003.

[85] N. Spring, D. Wetherall, and T. Anderson, “Scriptroute: A Public Internet Measurement Facility,” USENIX Symposium on Internet Technologies and Systems, 2003.

[86] M. Wawrzoniak, L. Peterson, and T. Roscoe. “Sophia: An Information Plane for Network Systems,” PDN-03-014, July, 2003.

[87] Dolev, S., Gilbert, S., Lynch, N., Schiller, E., Shvartsman, A., Welch, J., “Virtual Mobile Nodes for Mobile Ad Hoc Networks”, International Conference on Principles of Distributed Computing, 2004.

[88] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Hegebar, I. Pratt, A. Warfield. “Zen and the Art of Virtualization.” Proceedings of the ACM Symposium on Operating Systems Principles (SOSP). October 2003.

[89] Y. Chu, et al., “Early Experience with an Internet Broadcast Systems Based on Overlay Multicast,” in Proc. of USENIX, 2004.

[90] A. Bavier, M. Bowman, B. Chun, S. Karlin, S. Muir, L. Petersen, T. Roscoe, T. Spalink, M. Wawrzoniak, “Operation System Support for Planetary-Scale Network Service,” Proceedings of NSDI ’04: First Symposium on Networked Systems Design and Implementation, San Francisco, March 2004.

[91] L. Wang, K. Park, R. Pang, V. Pai, and L. Peterson, “Reliability and Security in the CoDeen Content Distribution Network,” in Proceedings of the USENIX 2004 Annual Technical Conference, Boston, June 2004.

[92] J. Hellerstein, P. Maniatis and T. Roscoe, "Design Considerations for Information Planes," Intel Research, IRB-TR-04-017, Aug. 30, 2004.

[93] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, L. Peterson, “Characteristics of

Internet Background Radiation," to appear in Proceedings of IMC October 2004.
[94] J. Sedayao, et al., "PlanetLab and its Applicability to the Proactive Enterprise," Intel Technical Journal, Nov. 2004.
[95] S. Rhea, et al., "OpenDHT: A Public DHT Service and Its Uses," in Proc. of ACM SigComm, 2005. http://www.opendht.org.

8.6 COOPERATIVE NETWORKING

[96] D. Milojicic, et al., "Peer-to-Peer Computing," HP Technical Report, HPL-2002-57, 2002.
[97] H. Balakrishnan, M. F. Kaashoek, David Karger, Robert Morris, and Ion Stoica, "Looking Up Data in P2P Systems," Communications of the ACM, vol. 46, no. 2, 2003, pp. 43-48.
[98] R. Huebsch, et al., "Querying the Internet with PIER," in Proc. of the 29th VLDB Conference, Berlin, Germany, 2003.
[99] D. Dash , B. Kveton , J. M. Agosta , E. Schooler , J. Chandrashekar , A. Bachrach , A. Newman, "When gossip is good: distributed probabilistic inference for detection of slow network intrusions," proceedings of the 21st national conference on Artificial intelligence, p.1115-1122, July 16-20, 2006, Boston, Massachusetts.
[100] Zage, D., Livadas, C. and Schooler, E. (2009) 'A network-aware distributed membership protocol for collaborative defense', Workshop on Leveraging Social Patterns for Security, Privacy and Networks (SP4SPNA'09).

8.7 WIRELESS MESH, SENSOR AND AD HOC NETWORKS

[101] http://www.zigbee.org/
[102] IEEE 802.11s ESS Mesh Network working group: http://grouper.ieee.org/groups/802/11/
[103] IEEE 802.15 WPAN working group, task group 5: http://www.ieee802.org/15/pub/TG5.html/
[104] C. Santivanez, B. McDonald, I. Stavrakakis, R. Ramanathan, "On the Scalability of Ad-hoc Routing Protocols," IEEE Infocom, New York, June 2002.
[105] R. Murty, "Cognitive Software-Defined Reconfigurable Radios: Smart, Agile, Cognitive, and Interoperable," Technology@Intel Magazine, July 2003.
[106] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, "Wireless Sensor networks: a survey", Computer Networks, vol. 38, 2002, pp. 393-422.
[107] D. S. J. De Couto, D. Aguayo. J. Bicket, and R. Morris, "A High-Throughput Path Metr
[108] I. Chlamtac, M. Conti, and J. J. -N. Liu, "Mobile Ad Hoc Networking: Imperatives and Challenges," Ad Hoc Networks, 1(1): 13-64, 2003.
[109] R. Draves, J. Padhye, and B. Zill, "Routing in Multi-radio, Multi-hop Wireless Mesh Networks," in Proc. of ACM MOBICOM, 2004.M. Hu and J. Zhang, "MIMO Ad Hoc Networks: Medium Access Control, Saturation Throughput, and Optimal Hop Distance," Special Issue on Mobile Ad Hoc Networks, Journal of Communications and Networks, 2004, pp. 317-330.
[110] A. Raniwala and T. Chiueh, "Architecture and Algorithms for an IEEE 802.11-based Multi-channel Wireless Mesh Network," in Proc. of IEEE INFOCOM, 2005.
[111] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," Computer Networks Journal, 47: 445-487, 2005.
[112] G. R. Hiertz, S. Max, E. Weiß, L. Berlemann, D. Denteneer, S. Mangold, "Mesh Technology enabling Ubiquitous Wireless Networks," in Proceedings of the 2nd Annual International Wireless Internet Conference (WICON), Boston, USA, Aug. 2006, Invited Paper, p.11.
[113] K. Papagiannaki, M. Yarvis, and W. S. Conner. "Experimental Characterization of Home Wireless Networks and Design Implications", IEEE Infocom, Barcelona, Spain, April, 2006.
[114] W. Wang, X. Liu, and D. Krishnaswamy, "Robust Routing and Scheduling in Wireless Mesh Networks," IEEE SECON 2007, San Diego, CA, 2007
[115] H. Yu, P. Mohapatra, and Xin Liu, "Dynamic Channel Assignment and Link Scheduling in Multi-radio Multi-channel Wireless Mesh Networks," Mobile Networks and Applications, Springer, April, 2008.

8.8 NETWORK & WIRELESS RESOURCE MANAGEMENT

[116] http://research.microsoft.com/netres/projects/virtualwifi.
[117] http://wireless-matlab.sourceforge.net
[118] Microsoft, "Quality of Service Technical Overview", Sept. 1999.
[119] Vicente J. et al, "Host-based, Network Traffic Control System", Intel Technical

Report.
[120] Holma, H., Toskala A., "WCDMA for UMTS: Radio Access for Third Generation Mobile Communications," John Wiley & Sons, ISBN 0-471-72051-8.
[121] Gomez J., Campbell, A.T., Morikawa, H. "A Systems Approach to Prediction, Compensation and Adaptation in Wireless Networks", WOWMOM 98, Dallas, Texas, 1998.
[122] G. J. Foschini, M. J. Gans, "On Limits of Wireless Communications in a Fading Environment when Using Multiple Antennas," Wireless Personal Communications 6: 311–335, 1998.
[123] Bianchi G., Campbell A.T., Liao, R.R.F., "On Utility-Fair Adaptive Services in Wireless Networks", IEEE 1998.
[124] Sinha P., Nandagopal, T., Venkitaraman N., Sivakumar R., Bharghavan V., "WTCP: A Reliable Transport Protocol for Wireless Wide-Area Networks", Wireless Networks, 1999.
[125] J. Vicente, A. T. Campbell, and D. A. Villela, "Virtuosity: Performing Virtual Network Resource Management," in Proc. of International Workshop on Quality of Service, London, June 1999.
[126] N.G. Duffield, P. Goyal, A.G. Greenberg, P.P. Mishra, K.K. Ramakrishnan , Jacobus E. van der Merwe, "A flexible model for resource management in virtual private networks", In: Proceedings ACM SIGCOMM'99, Computer Communication Review, Vol 29, No 4, October 1999, pp. 95-108.
[127] Ayyagari, A., Bernet Y., Moore T., "IEEE 802.11 Quality of Service –White Paper", IEEE 028 document, February, 2000.
[128] G. Bianchi, "Performance analysis of the IEEE 802.11 Distributed Coordination Function," IEEE Journal on Selected Areas in Communications, Vol. 18, Issue 3, March 2000, pp. 535–547.
[129] IEEE, "Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Standard 802.11, January 1999.
[130] Vicente, J., Campbell, A. T., "IP Radio Resource Control," in Proc.of International Conference on Management of Multimedia Networks and Services, October, 2001.
[131] D. Krishnaswamy, "Game Theoretic Formulations for Network-assisted Resource Management in Wireless Networks," in Proc. of IEEE VTC, September 2002.
[132] D. Qiao, S. Choi, K.G. Shin, "Goodput analysis and link adaptation for IEEE 802.11a wireless LANs," IEEE Transactions on Mobile Computing, Vol. 1, Issue 4, Oct-Dec. 2002, pp. 278–292.
[133] S. Shakkottai, T. S. Rappaport, and P. C. Karlsson, "Cross-layer Design for Wireless Networks," IEEE Communications Magazine, vol. 41, no. 10, 2003, pp. 74-80.
[134] P. Mohaptra, J. Li J, and C. Gui, "QOS in Mobile Ad Hoc Networks," IEEE Wireless Communications Magazine, June 2003, pp. 44-52.
[135] M. van der Schaar, S. Krishnamachari, S. Choi, X. Xu, "Adaptive cross-layer protection strategies for robust scalable video transmission over 802.11 WLANs," IEEE Journal on Selected Areas in Communication, Vol. 21, Issue 10, Dec. 2003, pp. 1752–1763.
[136] M. Conti, G. Maselli, G. Turi, S. Giordano, "Cross-Layering in Mobile Ad-hoc Network Design," IEEE Computer, Feb. 2004, pp. 48-51.
[137] G. Carneiro, J. Ruela, M. Ricardo, "Cross-layer design in 4G Wireless Terminals," IEEE Wireless Communications, April 2004, pp. 7-13.

[138] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft, "Structural Analysis of Network Traffic Flows," in Proc. of ACM SIGMETRICS, New York, June 2004.

[139] D. Krishnaswamy, R. Stacey, R. van Alstine, W. Chimitt, "Performance Considerations for Efficient Multimedia Streaming in Wireless Local-Area-Networks," 49th Annual SPIE conference on Applications of Digital Image Processing, August 2004.

[140] D. Krishnaswamy, M. van der Schaar, "Adaptive Modulated Scalable Video Transmission over Wireless Networks with a Game-Theoretic Approach," IEEE Multimedia Signal Processing Workshop, September 2004.

[141] E. Gelenbe and R. Lent, "Power Aware Ad Hoc Cognitive Packet Networks," Ad Hoc Networks Journal, vol. 2, Nov. 3, 2004, pp. 205-216.

[142] D. Krishnaswamy and J. Vicente, "Scalable Adaptive Wireless Networks," Intel Technical Journal, Q4, 2004.

[143] P. Kyasanur and N. Vaidya, "Capacity of Multi-Channel Wireless Networks: Impact of Number of Channels and Interfaces," in Proc. of ACM MOBICOM, 2005.

[144] M. Kodialam and T. Nandagopal, "Characterizing the Capacity Region in Multi-Radio, Multi-Channel Wireless Mesh Networks," in Proc. of ACM MOBICOM, 2005.

[145] L. Cheng, W. Zhang, L. Chen, "Rate-Distortion Optimized Unequal Loss Protection for FGS Compressed Video," IEEE Trans on Broadcasting, Vol. 50, No, 2, June 2004, pp. 126–131.

[146] "Cross-Layer Protocol Engineering for Wireless Mobile Networks: Part 2", IEEE Communications Magazine, vol. 44, no. 1, January 2006.

[147] S. Rungta , A.Rentzis, J. Sedayao, R. Adams, P. Brett, "Monitoring Internet Connectivity using PlanetLab", Proceedings of the 2006 Network Operations and Management Symposium (NOMS). Vancouver, April 2006.

[148] "Cross-Layer Protocol Engineering for Wireless Mobile Networks: Part 2", IEEE Communications Magazine, 44(1), 2006.

8.9 MACHINE LEARNING & STATISTICAL TECHNIQUES

[149] J. F. Meyer J.F, "On Evaluating the Performability of Degradable Computing Systems," IEEE Transactions on Computers, vol. C-29, no. 8, Aug. 1980, pp. 720-731.

[150] K. S. Trivedi, X. Ma, and S. Dharmaraja, "Performability Modeling of Wireless Communication Systems," International Journal of Communication Systems, vol. 16, 2003, pp. 561-577.

[151] S. Crosby and J. M. Agosta, "Network Integrity by Inference in Distributed Systems," in Proc. of NIPS Workshop Robust Communication Dynamics in Complex Networks, 2003, Whistler, Canada.

8.10 REINFORCEMENT LEARNING

[152] D. H. Ackley , M. L. Littman, “Generalization and scaling in reinforcement learning,” Advances in neural information processing systems 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1990

[153] M. Littman, & J. Boyan, (1993). “A distributed reinforcement learning scheme for network routing,” (Technical Report CMU-CS-93-165). Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.

[154] L. P. Kaelbling, M. Littman, A. Moore, “Reinforcement Learning: A Survey”, Journal of Artificial Intelligence Research, JAIR Volutme 4, (1996).

[155] Shelton, C. R. “Balancing Multiple Sources of Reward in Reinforcement Learning,” Submitted for publication in Neural Information Processing Systems (2000).

[156] C. Intanagonwiwat, R. Govindan, D. Estrin, J. S. Heidemann, F. Silva, “Directed diffusion for wireless sensor networking,” IEEE/ACM Transactions on Networking, Volume 11, Number 1, February 2003.

[157] B. Bakker, J. Schmidhuber, “Hierarchical Reinforcement Learning Based on Subgoal Discovery and Subpolicy Specialization,” Proceedings of the 8-th Conference on Intelligent Autonomous Systems, IAS-8, (2004).

[158] Dowling, J., Cunningham, R., Harrington, A., Curran, E. & Cahill, V. 2005 “Emergent consensus in decentralised systems using collaborative reinforcement learning.” In Self-star Properties in Complex Information Systems, pp. 63-80.

[159] P. Beyens, M Peeters, K. Steenhaut, A. Nowe, “Routing with compression in wireless sensor networks: A Q-learning approach,” Proceedings of the 5th European Workshop on Adaptive Agents and Multi-Agent Systems, (2005).

[160] A. Bar-Hillel, A. Di-Nur, L. Ein-Dor, R. Gilad-Bachrach, Y. Ittach, “Workstation capacity tuning using reinforcement learning.” In Proceedings of the 2007 ACM/IEEE Conference on Supercomputing. SC '07. ACM, New York, NY, 1-11. (2007).

[161] A. Gosavi. “Reinforcement Learning: A Tutorial Survey and Recent Advances,” INFORMS Journal on Computing 21(2): 178-192 (2009).

8.11 ENTROPY-BASED METHODS

[162] C.E. Shannon, "A Mathematical Theory of Communication", Bell System Technical Journal, vol. 27, pp. 379-423, 623-656, July, October, 1948.

[163] Beongku An, Symeon Pappavassiliou, "An Entropy-Based Model for Supporting and Evaluating Stability in Mobile Ad-hoc Wireless Networks," IEEE Communications Letters, vol.6, issue 8, pp.328-330, August 2002.

[164] N.B. Karayiannis, N. Kaliyur S.M., H. Malki, “Evaluation of neural and entropy-constrained routing of communication networks”, in: Proceedings of 2003 International Joint Conference on Neural Networks, Portland, OR, July 20-24, 2003, pp. 2722-2727.

[165] N. B. Karayiannis and S. Nadella, “Power-conserving routing of ad hoc mobile wireless networks based on entropy-constrained algorithms, Ad Hoc Networks 4: 24-35, 2006.

[166] Lall, A., Sekar, V., Ogihara, M., Xu, J., and Zhang, H., “Data streaming algorithms

for estimating entropy of network traffic." In Proc. ACM SIGMETRICS, 2006.
[167] J. Sacha , J. Dowling , R. Cunningham , R. Meier, "Discovery of stable peers in a self-organising peer-to-peer gradient topology," Proc. of the 6th IFIP Int. Conference on Distributed Applications and Interoperable, 2006.

8.12 BELIEF PROPAGATION

[168] J. Pearl, "Probabilistic reasoning in intelligent systems: networks of plausible inference," Morgan Kaufmann Publishers Inc., San Francisco, CA, 1988.
[169] R. Schoonderwoerd, J.L. Bruten, O. E. Holland, L.J. Rothkrantz, "Ant-based load balancing in telecommunications networks". Adapt. Behav. 5, 2 (Sep. 1996), pgs. 169-207.
[170] F. R. Kschischang and B. J. Frey and Ha. Loeliger, "Factor Graphs and the Sum-Product Algorithm," IEEE Transactions on Information Theory, Volume 47, pgs 498-519, (1998).
[171] E. Charniak, "Bayesian Networks Without Tears," AI MAGAZINE, Volume 12, Number 4, pgs 50-63, (1991).
[172] K. P. Murphy, Y. Weiss, and M. I. Jordan., "Loopy belief propagation for approximate inference: An empirical study," pages 467-475, (1999).
[173] R. Dodier. "RISO: An Implementation of Distributed Belief Networks," In Proc. AAAI Symposium on AI in Equipment Service, 1999.
[174] J. S. Yedidia , W. T. Freeman , Y. Weiss, "Generalized Belief Propagation," IN NIPS 13, MIT Press, pgs. 689—695, (2000).
[175] C. Crick, A. Pfe, "Loopy belief propagation as a basis for communication in sensor networks," Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-2003), Morgan Kaufmann, pgs 159-166, (2003).
[176] Steinder, M. and Sethi, A. S., "Probabilistic fault localization in communication systems using belief networks," IEEE/ACM Transactions in Networking 12, 5 (Oct. 2004), 809-822, 2004.
[177] J. S. Yedidia , W. T. Freeman , Y. Weiss, "Constructing Free Energy Approximations and Generalized Belief Propagation Algorithms," IEEE Transactions on Information Theory, Volume 51, pgs 2282-2312, (2005).
[178] Agosta, J. M., Diuk-Wasser, C., Chandrashekar, J., and Livadas, C., "An adaptive anomaly detector for worm detection," Proceedings of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques (Cambridge, MA). J. Chase and I. Cohen, Eds. USENIX Association, Berkeley, CA, 1-6, 2007.

8.13 HIERARCHICAL METHODS & CLUSTERING

[179] J Hawkins, S Blakeslee, “On Intelligence”, First Edition 2004, Times Books, Henry Holt & Company, LLC, ISBN 0-8050-7456-2, (2004).

[180] I. Katzela and M. Schwartz, "Schemes for fault identification in communication networks," Columbia University, Center for Telecommunication Research, Department of Electrical Engineering, New York, NY, Tech. Report CU/CTR/TR 362-49-09, 1994.

[181] Gruschke, B., “Integrated event management: event correlation using dependency graphs”. In: Proceedings of the 9th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management (DSOM 98) (Newark, USA, 1998).

[182] J. Ryser, M. Glinz, “Dependency Charts as a Means to Model Inter-Scenario Dependencies.” Modellierung 2001: 71-80.

[183] P. Basu, N. Khan, T. D. C. Little, "A Mobility Based Metric for Clustering in Mobile Ad Hoc Networks," icdcsw, pp.0413, 21st International Conference on Distributed Computing Systems Workshops (ICDCSW '01), 2001.

[184] S. Ghiasi, A. Srivastava, X. Yang, M. Sarrafzadeh, “Optimal Energy Aware Clustering in Sensor Networks.” Sensors. 2002; 2(7):258-269.

[185] R. Van Renesse, K. P. Birman , W. Vogels, “Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining,” ACM Transactions on Computer Systems (TOCS), v.21 n.2, p.164-206, May 2003.

[186] Y. P. Chen, A. L. Liestman, J. Liu, “Clustering Algorithms for Ad Hoc Wireless Networks” Ad Hoc and Sensor Networks. Nova Science Publishers.

[187] P. Tillapart, T., Thumthawatworn, P. Pakdeepinit, T. Yeophantong, S. Charoenvikrom, J. Daengdej, “Method for Clusterheads Selection in Wireless Sensor Networks.” In: IEEE Aerospace Conference, pp. 3615-3623. IEEE Press, Montana (2004).

[188] M. Steinder and A.S. Sethi, “Multi-Domain Diagnosis of End-to-End Service Failures in Hierarchically Routed Networks,” IFIP Networking, May 2004.

[189] V. Lo, D. Zhou, Y. Liu, C. GauthierDickey, J. Li, "Scalable Supernode Selection in Peer-to-Peer Overlay Networks," hot-p2p, pp.18-27, Second International Workshop on Hot Topics in Peer-to-Peer Systems, 2005.

[190] H. Chen; Megerian, S, “Cluster sizing and head selection for efficient data aggregation and routing in sensor networks”, Wireless Communications and Networking Conference, 2006, IEEE.

[191] Y. Yin, J. Shi, Y. Li, P. Zhang, “Cluster -Head Selection using Analytical Hierarchy Process for Wireless Sensor Networks”, The 17th Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC’06), China.

[192] H. Alipour, M. Esmaeili, M. Abbaspour, A.T. Haghighat, "DACA: Dynamic Advanced Clustering Algorithm for Sensor Networks", in proceedings of 14th IEEE International Conference on Electronics, Circuits and Systems, Marrakech, Morocco, September 2007.

8.14 REACHABILITY & GRAPH THEORY

[193] J. Mahdavi, V. Paxson, “IPPM Metrics for Measuring Connectivity,” IETF RFC2498 1999.

[194] G. Pandurangan , P. Raghavan , E. Upfal, “Building Low-Diameter P2P Networks,” Proceedings of the 42nd Annual IEEE Symposium on the Foundations of Computer Science, (2001).

[195] G.W. Flake, R.E. Tarjan, K. Tsioutsiouliklis, “Graph Clustering and Minimum Cut Trees”, Journal of Internet Math. Volume 1, Number 4 (2003).

[196] J. Xu; A. Kumar.; X. Yu; “On the fundamental tradeoffs between routing table size and network diameter in peer-to-peer networks”, Selected Areas in Communications, IEEE Journal on, Volume: 22 Issue:1, Jan. 2004.

[197] D. Loguinov, J. Casas, X. Wang, “Graph-theoretic analysis of structured peer-to-peer systems: routing distances and fault resilience” , IEEE/ACM Transactions on Networking, page(s): 1107 - 1120 , Volume: 13 Issue: 5, Oct. 2005.

[198] S. P. Borgatti and M.G Everett, “A graph-theoretic framework for classifying centrality measures,” Social Networks” 28(4): 466-484, (2006).

[199] R. H. Wouhaybi, A. T. Campbell, “Building resilient low-diameter peer-to-peer topologies,” Computer Networks 52(5): 1019-1039 (2008).

8.15 FIRST RESPONDER NETWORKS

[200] A. S. Bahora et al, “Integrated peer-to-peer applications for advanced emergency response systems Part 1: Concept of Operations”, SIEDS, 2003 IEEE, 24-25 April 2003, pp. 255–260.

[201] D. Bradler, J. Kangasharju, Max Muhlhauser, “Systematic First Response Use Case Evaluation Systematic First Response Use Case Evaluation”, MODIES, (2008).

Appendix

9 APPENDIX

APPENDIX A – OVERMESH PDK

PLATFORM DEVELOPMENT KIT

The following sections outline the platform development kit (PDK) and installation procedures for the OverMesh experimental environment.

Description:
Scripts in this package are needed to install MyPLC, compile Flexmesh, and create customized bootCD and PlanetLab-Bootstrap. This package contains the following files in the uncompressed OverMesh folder.

The compressed tar file "*overmesh-pdk-1.2.tar.gz*" contains the sources and scripts for OverMesh platform v1.2. *Untar overmesh-pdk-1.2.tar.gz* to */root*.

Directory/file	***Source info***
flexmesh/flexmesh-6.1.1-fc4	Source for flexmesh 6.1, IT Research Edition for FC4
flexmesh/ieee80211-1.2.15	Script to remove default ieee80211 in FC4
flexmesh/ipw2200-1.2.0	Script to remove default ipw2200 in FC4
flexmesh/flexmesh-6.1.1-winxp	Source for Flexmesh 6.1 for Windows XP
flexmesh/meshtopology	Mesh topology software for Windows XP
myplc/source/myplc-0.4.1.planetlab.i386.rpm	MyPLC binary
myplc/bootcd	Script to create custom bootCD
myplc/bootmanager	Script to create custom PlanetLab-Bootstrap
myplc/source/build/kupdate.sh	Script to install custom kernel into the BootCD and the PlanetLab-Bootstrap images.
myplc/source/build/kernel-2.6.12-1.1398_FC4.5.planetlab.i686.rpm	Kernel rpm customizing the BootCD and Planetlab-Bootstrap images for MyPLC nodes
myplc/source/build/vnet-0.5.1.planetlab.i386.rpm	Virtual network rpm for customizing the Planetlab-Bootstrap image
flexmesh/flexmesh-6.1.1-fc4/sources/linux/release/flexmesh-6.1.1-laptop-2.6.12.tar.gz	Compiled flexmesh for the 2.6.12-1.1398_FC4.5.planetlab kernel in BootCD and the Planetlab-Bootstrap images

I N S T A L L A T I O N

O v e r M e s h C e n t r a l S e r v e r

Install Fedora Core 4

All options not mentioned should use Default values

1. To start installation, use *"linux resolution=1280x1024"* (or whichever highest resolution the equipment supports).
2. Turn off firewall
3. Disable SE Linux
4. Select everything when asked about the software packages to install.
5. Configure the host name of the machine at Desktop →System Settings →Network →DNS. Ensure the host name is added to the */etc/hosts* file.

Wireless Card and Flexmesh

Update Kernel

Flexmesh 6.1.1 supports FC4 with 2.6.11 kernel. This version is a customized from version 6.1.0 to run on FC4 as daemon. By default, FC4 is installed with the ieee80211 and ipw2200 subsystems. However, Flexmesh 6.1.1 comes with its own custom version.

1. To remove the default ieee80211 subsystem in FC4, run the scripts *~/overmesh/flexmesh/ieee80211-1.2.15/remove-old*
2. To remove the default ipw2200 subsystem in FC4, run the scripts *~/overmesh/flexmesh/ipw2200-1.2.0/remove-old*
3. Go to the directory */usr/src/kernels/`uname -r`*
4. *grep -rn "CONFIG_NET_RADIO" /.config* and ensure that CONFIG_NET_RADIO=y
5. *grep -rn "CONFIG_NET_WIRELESS" /.config* and ensure that CONFIG_NET_WIRELESS=y

Install Flexmesh

Flexmesh-6.1.1-fc4 should already have the patches for *hostapd* and *wpa_supplicant*.

1. Copy all header files from */usr/lib/gcc/i386-redhat-linux/4.0.0/include* to */usr/lib/gcc/i386-redhat-linux/3.4.2/include.* Create directory as necessary.
2. *cp ~/overmesh/flexmesh/flexmesh-6.1.1-fc4/sources/linux/kernel/clx2/ipw2200-flexmesh/net/ieee80211.h /usr/src/kernel/`uname –r`/include/net/.*
3. *cp ~/overmesh/flexmesh/flexmesh-6.1.1-fc4/sources/linux/kernel/clx2/ipw2200-flexmesh/net/ieee80211_crypt.h /usr/src/kernel/`uname –r`/include/net/.*
4. Go to *~/overmesh/flexmesh/flexmesh-6.1.1-fc4/sources/linux* and run *make clean* to start fresh

5. Then at *~/overmesh/flexmesh/flexmesh-6.1.1-fc4/sources/linux*, run *make release*
6. *Copy ~/overmesh/flexmesh/flexmesh-6.1.1-fc4/sources/linux/release/laptop.tar.gz* to */flexmesh*
7. Run *tar xvfz /flexmesh/laptop.tar.gz* to unpack *laptop.tar.gz*
8. Execute */flexmesh/release/laptop/install.sh* to install Flexmesh
9. In this Flexmesh release, the configurations are dispersed to multiple files. The master configuration file is located at */etc/fmcfg*. This file contains pointer to the other configuration files.
10. Use *modinfo* to check module version as necessary for troubleshooting.

Configure DHCP Environment

In this environment, the OverMesh Central takes on roles of PlanetLab Central, DHCP server, and topology server. The DHCP server has to use a static IP and must be set up first.

1. Go to */etc/fm_static_ip* and assign the address 192.168.99.1
2. Then go to */etc/fm_dhcp_server* and set the variable DHCP_SERVER=1
3. Then go to */etc/fm_dhcp_enable* and set the variable DHCP_ENABLE=0
4. The service dhcpd at the service control panel under System Settings →Server Settings → Services does not need to be checked. Flexmesh will automatically start the dhcpd daemon if DHCP_SERVER is enabled.
5. Modified the */etc/dhcpd.conf* file to configure any static IP addresses.

Configure DHCP Client

All other OverMesh nodes are DHCP clients and topology clients.

1. For clients, go to */etc/fm_dhcp_enable* and set the variable DHCP_ENABLE=1. DHCP server cannot have this set to 1 or flexmesh will throw an exception on start. If installing from version 6.1.1, this is by default set to 1.
2. When Flexmesh starts, the client should obtain an IP and be seen in the routing table in about 15-30 seconds.

Configure Topology Server

1. Go to */etc/fm_enable_demo* and ensure that ENABLE_DEMO=1.
2. Go to */etc/fm_topo_server* and ensure that TOPOSERVER=1.

Configure Topology Client

All other OverMesh nodes are DHCP clients and topology clients.

1. Go to */etc/fm_enable_demo* and ensure that ENABLE_DEMO=1.
2. Go to */etc/fm_topo_server* and ensure that TOPOSERVER=0.

Run Flexmesh as a Service

The following applies to both server and clients.

1. During installation, the script files from */flexmesh/release/laptop/scripts/init.d* should already be copied over to */etc/init.d*. If not, then recopy.
2. Run "*chkconfig –add flexmesh*" to configure the service at different run levels. Alternately, if flexmesh needs to be removed in the future, run "*chkconfig –del flexmesh*".
3. Run "*chkconfig –add flexmesh-topo*" to configure the service at different run levels. Alternately, if flexmesh needs to be removed in the future, run "*chkconfig –del flexmesh-topo*".
4. Run "*service flexmesh start*" to start flexmesh. To stop flexmesh, run "*service flexmesh stop*". Alternately, you can start flexmesh manually by running "fm_start". To stop flexmesh manually, run "fm_stop".
5. Run "*service flexmesh-topo start*" to start flexmesh. To stop flexmesh, run "*service flexmesh-topo stop*". Alternately, you can start flexmesh manually by running "fm_start_topo". To stop flexmesh manually, run "fm_stop_topo".

Manually setup Multi-hop Network

MAC layer filter setup

1. Run *cp /etc/fm_filters.sample /etc/fm_filters* to make a copy of the sample filter config
2. Edit */etc/fm_filters* and add the MAC address of the node you want to filter out
3. Restart the flexmesh service
4. Redo steps 1-3 on the other nodes

MyPLC – My PlanetLab Central

Refer to the PlanetLab website for specific install instructions. The *myplc rpm package* can be downloaded here.

Install MyPLC

1. Run "*rpm -Uvh myplc-0.4.1.planetlab.i386.rpm*" to install myplc.

Configure MyPLC

1. Modify */etc/planetlab/plc_config.xml* to change configuration of the installation. A sample configuration file is located at *~/overmesh/myplc/config/plc_config.xml.sample*. By default, any modifications made in this XML file will be automatically written to */etc/plc_config* file.
2. Run "*service plc start*" to start plc. All of the services being started and stopped should have a status of "OK".
3. Go to *http://localhost* and login to the plc website

Backup MyPLC files

The instructions following this section will modify several key scripts and image files in MyPLC.

1. */plc/root/usr/share/bootcd/build/isofs/bootcd.img*
2. */plc/root/usr/share/bootcd/build/isofs/overlay.img*
3. */plc/data/var/www/html/boot/PlanetLab-Bootstrap.tar.gz*
4. */plc/data/var/www/html/boot/alpha-BootLVM.tar.gz*
5. */plc/data/var/www/html/boot/alpha-PartDisks.tar.gz*
6. */plc/root/usr/share/bootmanager/source*

Replace custom MyPLC scripts

After MyPLC is installed and the PLC service started successfully, replace several BootManager scripts in MyPLC. Open the *customize-bootmanager.sh* to see which files are being replaced.

1. Go to the directory *~/overmesh/myplc/bootmanager/scripts/custom*
2. Run the script *./customize-bootmanager.sh*
3. Note, in the *ChainBootNode.py* script, the root password used after the bootstrap kernel is loaded is set to blank. This is for troubleshooting purpose. When deploying this platform for use, remember to uncomment out the line that set the password.

Update bootCD and bootstrap kernel and images

During boot, MyPLC node will communicate with MyPLC central server to download additional files. Flexmesh needs to be working during this phase. The kernel version in the boot CD varies, depending on the schedule when PlanetLab compiles a new boot CD image. Typically, the kernel version will be recent. The compiled version of Flexmesh included in the PDK is modified to run on kernel version 2.6.12.xxx.planetlab. We'll have to update the kernel in the boot CD. The bootmanager runs after boot CD during node installation; the boot manager script bootmanager.sh.sgn is downloaded by the new node to continue installation. The script can be updated in MyPLC at */plc/root/usr/share/bootmanager/source*. After changing the script, run *"service plc start bootmanager"* at the terminal to compress the source, convert it to ASCII, sign it, and copy it to the boot directory in the WWW directory at */plc/data/var/www/ html/boot*. The installation handled by the boot manager involves multiple steps (source code in */plc/root/usr/share/bootmanager/source/steps*).

The ~/overmesh/myplc/bootmanager/scripts/custom/customize-bootmanager.sh executed above already replaced and resigned these files. During installation, the bootmanager downloads */plc/data/var/www/html/boot/PlanetLab-Bootstrap.tar.bz2*. This bootstrap files contain the kernel (a variant of kernel 2.6.12 in myplc 0.4 rc1) that the node will eventually run on. The bootstrap also contains all necessary PLC software such as Linux-VServer, etc. Customize *PlanetLab-Bootstrap.tar.bz2* to include Flexmesh.

1. Go to directory *~/overmesh/myplc/source/build*
2. Run *./kupdate.sh kernel-2.6.12-1.1398_FC4.5.planetlab.i686.rpm vnet-0.5-1.planetlab.i386.rpm*
3. Run *~/overmesh/myplc/bootcd/customize-bootcd-2.6.12.sh*
4. Run *~/overmesh/myplc/bootmanager/alpine/customize-bootstrap-2.6.12.sh*
5. If any of the customize scripts need to be rerun, restore bootcd.img and PlanetLab-Bootstrap.tar.gz from backup and retry from step 1.

O v e r M e s h N o d e s

The OverMesh nodes host the virtual machines. These nodes boot from bootCD and will download additional scripts called bootmanager from MyPLC. The bootCD contains a unique key specific for each node. This is a PlanetLab design requirement for security and reliability purposes.

Instantiating an OverMesh Node

Register new user at OverMesh central

1. Register a new account on PLC website.
2. Login as admin or PI and approve the new account.
3. The user generates his RSA keys by *ssh-genkey -t rsa -f ~/.ssh/id_rsa* and uploads one of the key files *~/.ssh/id_rsa.pub* to the MyPLC central server. Store the other private key file to a private place.

Register new node at OverMesh central

Please refer to the public PlanetLab MyPLC documentation for background information.

Create a unique boot image for the OverMesh node

At the PLC website, add a new node, select DHCP for the node and provide an IP address. The PLC will generate a configuration file for the node. The unique key mentioned above is located in this file. Do the following after the configuration file is downloaded:

1. Download the configuration file and replace *~/overmesh/myplc/bootcd/planet.cnf* file.
2. Insert the line IP_DNS1="192.168.99.1" at the end of the planet.cnf file. This is the IP address of the MyPLC central server, which is also serving as DNS server.
3. Go to the directory *~/overmesh/myplc/bootcd*
4. Run *./create-bootcd.sh planet.cnf*
5. Run *./burn-bootcd.sh* to burn the bootCD for the specific node

First Boot

Using BootCD

1. Put the boot CD into the CD drive and start up laptop.
2. After booting from the boot CD, according to */etc/inittab*, the scripts in */etc/init.d* are executed, such as pl_sysinit (calls *pl_hwinit* and *pl_netinit*), and *pl_boot. pl_hwinit* and *pl_netinit* have been modified to install and start the mesh networking function.
3. In *pl_boot,* the new node connects to the OverMesh central using the wireless connection. And the bootmanager scripts are downloaded from the overmesh central website boot directory. From now on, the task of the boot CD is complete and the control of the new node is transferred to the boot manager, which will set up file system, download the bootstrap image, and *kexec* into the bootstrap image.
4. The administrator can login to a node as *site_admin by ssh -l site_admin -i <location of your private key> <node name>*. You can set a password for console login using *site_admin by sudo /usr/bin/passwd site_admin.*

Create an Overlay "Add a Slice"

After the OverMesh node completes installation, we can create a slice at PLC. A slice is a virtual network overlay, formed of multiple distributed virtual machines on different physical machine nodes. We can assign a number of nodes to a slice as necessary. The PLC will create a virtual machine on each of the member nodes.

1. For a new site, the admin or PI should login to the PLC website and update the site by setting the Maximum Number of Slices" to at least 1. By default it is 0.
2. On PLC website, create a new slice by an admin or PI.
3. Associate users and nodes to the new slice. Note, you must install the new OverMesh nodes before you can move to this step.
4. Wait an one hour before trying to login to the node using the slice
5. Login by *ssh -l <slice name> -i <location of your private key> <node name>*. Refer to PlanetLab documentation at the central server for more details.
6. Add option *-v* to see more debug messages if login fails.
7. A slice can be shared by multiple users. Once a user is associated with a slice, they can add/delete nodes to the slice. It is sufficient to have one user associated with one slice.

Wireless Mesh on Windows

Mesh topology is a Windows flash program that displays the Flexmesh network topology and it used primarily for demo purposes and work in conjunction with the Flexmesh-topology service.

Install Flexmesh on Windows

This assumes that system is build with Windows XP, service packs, and appropriate drivers such as chipset, audio, video, Ethernet, & wireless. The Flexmesh will override the wireless driver so it should not matter if there‟s a wireless driver present in the intial build. Windows Driver Development Kit, Visual Studio 6.0, and cygwin are needed as well.

The Flexmesh Win XP source is located in the *~/overmesh/flexmesh/flexmesh-6.1.1-winxp* directory of the tar package. Copy the content of this folder to *C:\cygwin\home\flexmesh*.

Compile Userspace Software

Microsoft Visual Studio 6.0 is needed to compile the sources. OpenSSL must be installed if you want to use authentication feature. The default installation directory is assumed to be *C:\OpenSSL*. If it is different, you need to modify the project files in wpa_supplicant and hostapd to point to the right directory. This should create all user space programs such as win_fmsvc.exe, win_mux.exe, win_deamon.exe, and etc.

1. Open .*flexmesh\sources\windows\makeall\makeall_no_auth.dsw* in Visual Studio. This is the package without authentication modules. If you need authentication, open .*flexmesh\sources\windows\makeall\makeall.dsw* instead.
2. Go to Build→ Batch Build → Visual Studio to build the project.

Compile Kernel

You need Cygwin to compile the miniport driver (for wireless card) and Windows Driver Development Kit (DDK) to compile the i*m_driver* (for aodv routing). Before compilation, you need to set the environment if necessary.

Setup Environment

1. Install Windows Driver Development Kit
   a. Search for Windows Driver Development Kit from Microsoft
   b. cd .*flexmesh\sources\windows\kernel\clx2\l2\miniport_driver*
   c. Edit file *locals*. Set TOOLSDIR to the directory where the DDK is installed. e.g. *C:/WINDDK*. Note: "/" should be used as the directory separator.

d. cd to the *makes* subdirectory and edit file *defines.mk*. Set the DDK variable to the directory where the DDK is installed. If the DDK directory is *C:/WINDDK/3790.1830*, then set DDK=3790.1830.

2. Install Cygwin
   a. Go to *http://www.cygwin.com*
   b. Select to install "All" when prompted to do so.

Build Miniport Driver

1. Open a Cygwin terminal, typically running the *C:\cygwin\cygwin.bat*. You may want to copy the whole fFexmesh source code to *C:/cygwin/home/<your_user_name>* so that Cygwin can find it.
2. cd *./flexmesh/sources/windows/kernel/clx2/l2/miniport_driver*
3. make cleanall
4. make build (Do NOT use make deps)
5. This should generate the Cx2nc51.sys in *./bin/xp/checked* or *./bin/xp/free* directory (based on the DEBUG flag).

Build the Intermediate Driver

1. Open a build environment command window supplied with the Windows DDK. (e.g., Start->All Programs->Development Kits->Windows DDK 3790.1830->Build Environments->Windows XP->Windows XP checked build environment.)
2. cd *./flexmesh/sources/windows/kernel/clx2/l2/im_driver/driver*
3. Type "build" at the prompt. This should generate the *./objchk_wxp_x86/i386/aodv_routing.sys* file.

Build Release

1. cd *./flexmesh/sources/windows* in a command window
2. Execute make_release.bat at the command prompt, type Y for all the prompts.
3. This creates the release tree in *./flexmesh/sources/windows/release/flexmesh*
4. Move *./flexmesh/sources/windows/release/flexmesh* to *C:*

Driver Installation

If you are re-installing the drivers in a machine that has flexmesh installed already, it is not necessary to follow the instructions in this section. Just copy *C:/flexmesh/imdriver_installfile/AODV_Routing.sys* & *C:/flexmesh/miniport_installfile/ Cx2nc51.sys* to *C:/windows/system32/drivers* and reboot the machine. Otherwise, if this is a new install, follow the instructions below.

1. *miniport driver*
   a. Go to Device Manager
   b. Select update driver for the wireless card
   c. Do not use the wizard
   d. Install from a list of specific location

e. Don't search and select the file to install
f. Select the installation file in *C:/flexmesh/miniport_installfile*.

2. *im_driver*
   a. Got to Network Connections
   b. Select Wireless Network Connection
   c. At the General tab, click on Install
   d. Select Service, then click on Add
   e. Select *aodv_routing* in *C:/flexmesh/imdriver_installfile*

Userspace

The *C:\flexmesh* directory should contain the scripts and userspace software.

1. With an existing installation of flexmesh:
   a. *cd C:\flexmesh*
   b. Stop flexmesh by executing *C:\flexmesh\scripts\fm_stop_mesh.bat*
   c. Uninstall the existing installation by executing *C:\flexmesh\scripts\uninstall.bat*
   d. Backup the *C:\flexmesh* directory
2. To use your old configuration, from a previously installed version, copy the following files from that old installation:
   a. *cfg\fm_node_id*
   b. *cfg\fm_dhcp_enable*
   c. *cfg\fm_static_ip*
   d. *cfg\certs\ca.crt* (only if you use authentication)
   e. *cfg\certs\local.crt* (only if you use authentication)
   f. *cfg\certs\local.key* (only if you use authentication)

Execute *C:/flexmesh/script/install.bat* to configure Flexmesh as a Windows service to automatically start post boot.

Setup Mesh Topology Software

The mesh topology software is located in the *~/overmesh/flexmesh/meshtopology* directory of the tar package. Copy this folder to the desktop.

Setup Data Streams

At the topology server, typically 192.168.99.1, open */etc/fmstreams* and add the two end points (both directions) that you want to monitor (e.g., streams=MACADDR1-MACADDR2 MACADDR2-MACADDR1).

Start the Mesh Topology Software

Ensure the following are setup on all client nodes, except for the topology server, which is typically 192.168.99.1.

1. Ensure that */etc/fm_filters* are properly setup on all nodes and that the service flexmesh-topo has started on all nodes.

2. Change ENABLE_DEMO to 1 in *fm_enable_demo*, in *flexmesh/cfg.*
3. When the flexmesh-topo service is started on Linux systems and the *fm_start_mesh* script is run on Windows, the topology information is sent to the topology server, by default is 192.168.99.1.
4. At the Windows client, configure the file *./meshtopology/topo_init.xml.* Ensure that the IP address for the topology server is set correctly at <*var path=”ipBox.ipBox” text=”192.168.99.1” />*
5. At the Windows client, run the flash program buy ‘double clicking’ the *meshtopology_063.exe* image file. The network topology should be displayed.

I M P L E M E N T A T I O N

OverMesh Administration Site

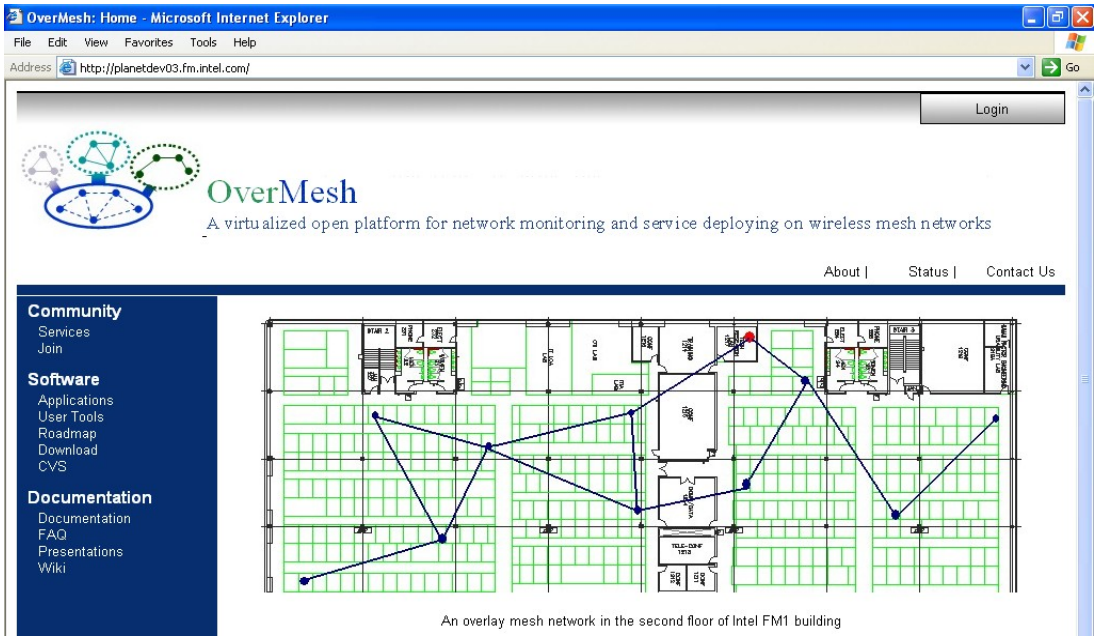
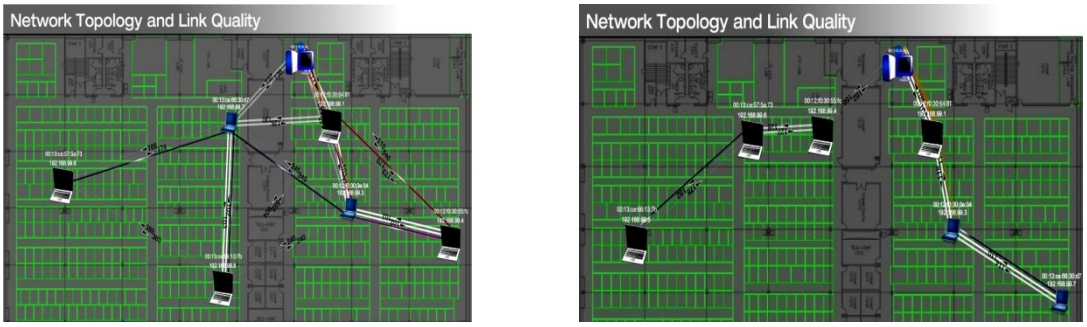


Figure 81: OverMesh Administration Site

Office Environment Mesh Topology

Configuration: Seven IBM ThinkPad Mobile Platforms
Workload: OverMesh nodes randomly send packets to peer nodes every 20-40 secs

Sample Experimental Measurement Plots



(a) Mesh topology *(b) Linear topology*

Figure 82: Network topology used in the experiments

The brighter a white line, the higher the corresponding link quality. Other colored lines represent the real data traffic between each pair of source and destination. The first experiment runs for three consecutive days on the mesh topology, including a weekend (black line) and two working days (green and blue lines) as shown in Figure 83. All seven (7) mesh nodes constantly send data packets to a randomly selected node in the overlay. The time interval between two data packets is randomly chosen between 20 and 40 seconds. Figure 83 also shows the ratio of successful data transmission, the response time, and the number of physical hops between the source and destination. In most cases, the system performs better in the weekend than in the working days because of the lack of human activities. When node 2 was shut down from 2 AM to 10 AM in the second day, the green lines show that the failure of one node may affect quite a few nodes in the whole network because of multi-hop routing among mesh nodes.

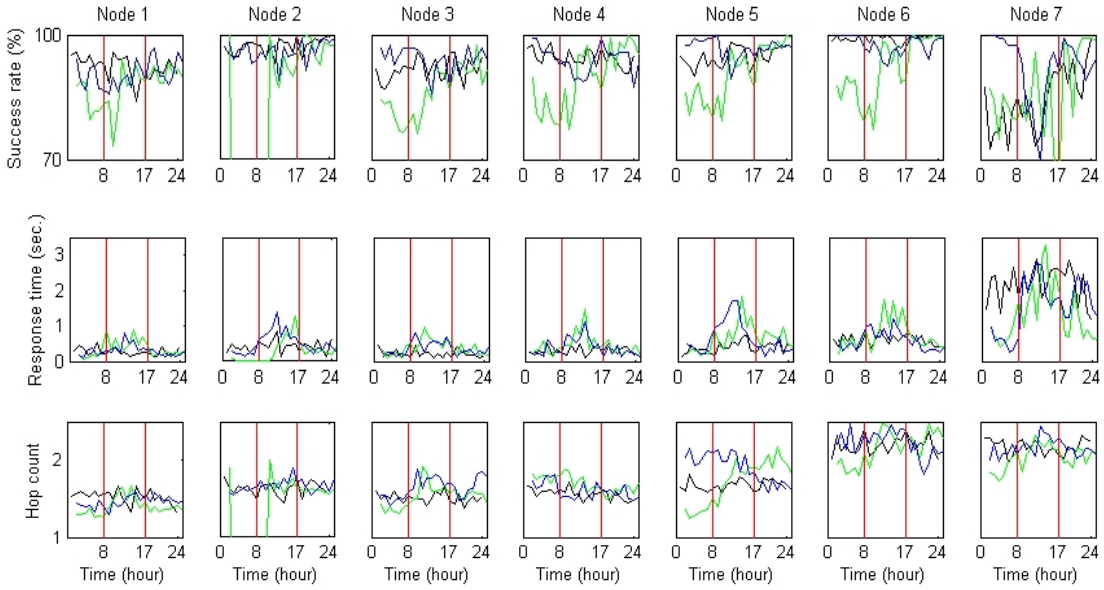


Figure 83: Success rate, hop count, response time

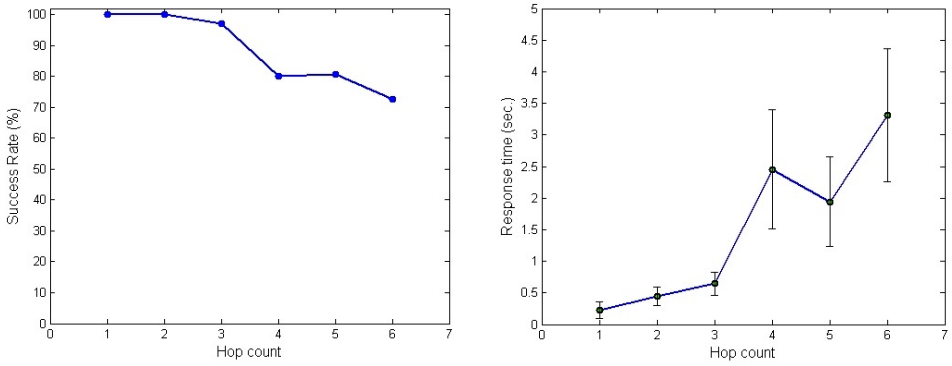
Sample Simulation Plots

An open source event-driven simulator:
http://wireless-matlab.sourceforge.net

Traffic: 1-10 concurrent search requests

30 nodes, 95% confidence interval

Figure 84 displays the hop count changes. In order to provide a fixed hop count from 1 to 6, we use the linear network topology shown in Figure 84(b), in which in-bound packets from nodes other than the immediate neighbors are blocked at MAC layer. Increasing hop count reduces the success ratio and increases the response time. This is due to the fact that there is only one route available, in which some wireless link may not be good enough due to interferences or long distance.

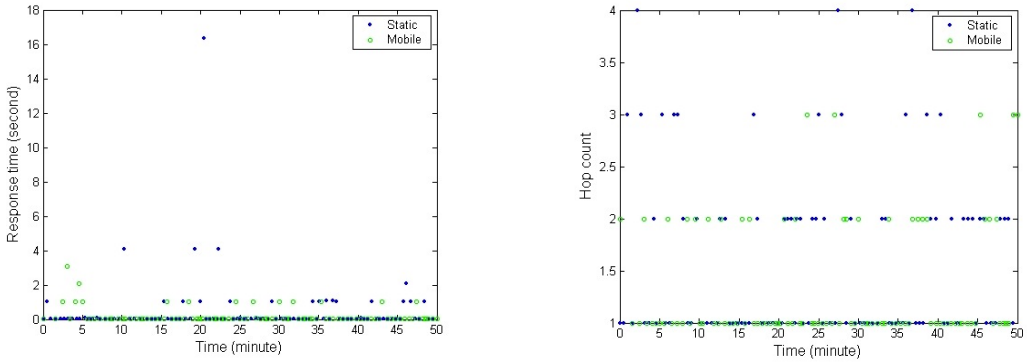


(a) Success ratio *(b) Response time*

Figure 84: Experimental results for hop count

Figure 85 compares the response time and hop count between a static network (blue dot) and a mobile network (green circle). In the mobile environment, each node moves at walking speed for five minutes alternatively with 100 transmitted packets overall. While mobility introduces more data loss, it is interesting to note, however, that the response time and hop counts are lower when there is only one data traffic, while they are larger for mobile network when there are seven data traffics. This implies that the mobility may improve the performance of low data traffic loads, but when there are

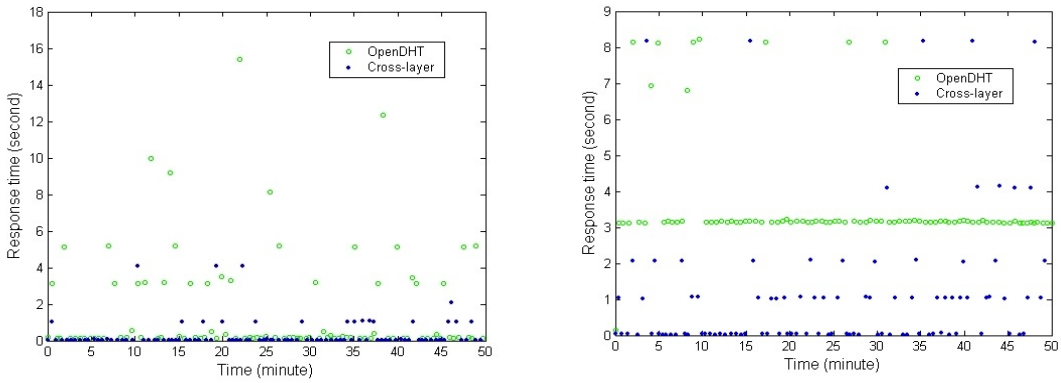
many data traffics, mobility of nodes may not help, instead, it will introduce more varying routes and packet losses.



(a) Response time (b) Hop count

Figure 85: Experimental results for mobility

Figure 86 compares the cross-layer overlay searching (blue dot) and the OpenDHT overlay searching (green circle) when the 100 searching requests are sent. Both the mesh network topology and the linear network topology are tested. It is evident that the cross-layer searching renders less response time. In average, the response time of cross-layer searching is 1.15 seconds, while the response time of OpenDHT searching is 3.55 seconds.



(a) Mesh topology *(b) Linear topology*

Figure 86: Experimental results for overlay searching