

# Manifolds: Cellular Component Organizations

Charles Herring and Simon Kaplan  
 [{herring,kaplan}@dstc.edu.au](mailto:{herring,kaplan}@dstc.edu.au)

*Department of Computer Science and Electrical Engineering  
CRC for Distributed Systems Technology  
The University of Queensland  
Brisbane, Queensland QLD 4072*

## Abstract

*This extended abstract reports on initial investigation into a software architecture for component systems. Described is a geometrically-based, cellular framework for the organization of component computation and communication. These structures are called Cellular Component Manifolds. The starting point for this work is the observation that the concept of address space is a pervasive abstraction across all computing and communicating systems. Accordingly, a generalized model of an idealized address space is proposed. A particular geometry with a suitable algebraic structure is chosen to provide addressing within this model. This results in an  $n$ -dimensional, hierarchical, cellular-aggregate address space that is also a vector space with “nice” properties. The address space, when populated with suitable components, becomes a Cellular Component Manifold. A component system architecture is given. This software architecture consists of three major layers: component framework framework, component frameworks, and components. In addition to providing a basis for component system organization this structure natively supports a range of scientific and engineering applications. As an example, the architecture is applied to the problem of handoff management in wireless overlay networks.*

## 1. Introduction

The component-oriented approach to software construction and deployment within economically viable component market places apparently represents the next stage in the evolution of software practice. However, there are few examples of component systems, i.e., component frameworks and component system architectures. Thus far developers have concentrated primarily on individual, isolated components and today there are only basic component interoperation facilities. If components are to achieve the impact that many are hoping for, discovery of organizing principles enabling design and development of component system architectures is required. The search for such organizing principles on which to base component system architectures motivates the work described in this paper.

The component architecture presented here came about by examining a wide range of computing and communications system with the goal of identifying key abstractions that are present in all systems. This search revealed the concepts of **address, addressing and address spaces** to be fundamental abstractions across all systems. This is true as all entities in computing and communications systems must be referenced by numeric labels - addresses. Human-friendly character strings, such as file names and URL's, are always mapped to numbers at some lower level in the system.

Now, systems evolve over time and the global communications and computer networks have been built up, subsystem by subsystem, layer by layer, over the last hundred or so years. However, systems are rarely designed to operate over more than one address space. Addressing within the various layers and subsystems are ad hoc, determined by preexisting conditions, or forced by hardware. As systems mature and more complex interconnections become desirable and feasible, these systems are required to communicate across subsystem boundaries. Usually some sort of address translation or mapping can be arranged.

The focus on the concept of address space as central to future component systems is driven, in part, by the fact that computing and communications technologies are rapidly coming together to form a global, integrated, information infrastructure. Major trends driving this merger are:

- Integration of telephony, television, and computer networks
- Integration of wireline and wireless networks
- High bandwidth, low latency communications
- Ubiquitous computing
- Smart buildings and smart appliances.

These developments are prompting a rethinking of the fundamental structure of basic support systems such as operating systems and network protocols. In order to realize the potential of greater functionality, recognition and promotion of new levels of abstraction in software architectures is required. Research projects driven by these forces include next-generation distributed operating systems such as Sun's Jini [1], and Microsoft's Millennium [2]; and integration of heterogeneous wireless networks as in the Wireless Overlay Network effort at UC Berkeley [3]. The position taken here is that addressing is at the heart of many critical architectural issues, e.g. scalability and seamlessness, that motivate these and other projects.

The rest of the paper is organized as follows. Section 2 explores the concept of address space in some detail to bring out the essential features that must be present in any general model. Based on this analysis, an initial generalized address space model is outlined in Section 3. This model includes the basic notions of centers, boundaries, boundary crossing, and hierarchy. The next step is incorporation of an addressing scheme within the model. The search for an appropriate addressing system resulted in choosing a particular geometric structure. This geometric system, the Generalized Balanced Ternary, is briefly described in Section 4. The result of combining this geometric addressing scheme with the general model results in a hierarchical, n-dimensional cellular-aggregate address space that is also a vector space with "nice" properties. This structure, when populated with components, is called a Cellular Component Manifold. Instances of these are the "containers" for software components. How a component system architecture based on this structure might be organized is the subject of Section 5. An example is developed in Section 6 to illustrate the proposed use and benefits of the component organization. The example concentrates on one critical aspect of wireless systems: the handoff of a mobile host between stations.

## **2.The Address Space Concept**

In every area of computing and communications addresses, in some form, exists. Perhaps the most familiar address spaces are in the context of operating systems where memory management is a major concern. There, several types of address spaces are recognized and their structures are described in detail. In networking, e.g. IPv6 addressing, great care is taken to provide a suitable addressing scheme for routing and other network management needs. However, it appears not to have been necessary to develop a general model or theory about this central abstraction. (In conducting a literature review one title did give hope: "A Grand Unified Theory of Address Space" [4]. Alas, it only covered operating systems.) The task of this section is to identify the key aspects of addressing and address space common to all systems on which a model can be based.

Looking down from outer space on the global information infrastructure the first structures we see are the various satellite communications systems. Zooming in closer we see microwave transmission towers, telephone poles, etc. If we could see through the skin of the Earth we would see coax cables, fiber and power lines. Continuing in this manner we see WANs, MANs, and LANs. Soon we are inside a computer “looking” at its hardware and software internals. At each step in our zooming we are crossing levels of the system. Large software systems, such as operating systems and network protocols, are (at least conceptually) made up of layers that encapsulate and shield implementation details from higher layers. Within each level, the subsystems that make up that level are defined by their boundaries with neighboring subsystems. These boundaries are usually different address spaces. Thus complex systems are composed of layers or hierarchies of subsystems. Subsystems are identified by boundaries between address spaces.

The concept of an address space takes many forms and is known by many names. Some examples will illustrate. The most familiar use of the concept is in relation to operating system processes. The address space of a process is that portion of memory allocated to it by the operating system. Internally it is divided into regions such as the stack, heap, etc. where threads belonging to the process manipulate code and data. Another example is the Internet Protocol that relies on the use of a 32-bit address containing the network and host identifiers for packet routing. There are several layers of addressing in IP such as the Address Resolution Protocol that maps IP addresses to 48-bit Ethernet addresses. The concept of Name Space is also related and provides a human readable mapping to underlying numerical address space notations as in DNS. A cell in a GSM telephone system is another example. There are several possible address spaces of interest in this complex system. The physical extent of the cell is one. Each subscriber within the cell is located at a physical address expressible in some real world coordinate system. The number and range of frequencies available for control, transmission and reception might be another.

An address space is a *homogeneous* region from the standpoint of higher level systems. The operating system allocates a range of memory to a process and turns it over to the application. It is the lower subsystem, e.g. user application, that organizes that memory allotment into regions, (e.g. stack, heap, invocation frames, data.) To the operating system it is a chunk of memory that is now someone else's responsibility – almost. The operating system manages the life cycle of that piece of memory. It must also handle exceptional events related to the process. That is, events generated by the subsystem may be passed up to the operating system. A classic event in this case is an unresolved pointer reference. In a virtual memory operating system, a number of possible actions may take place based on this event. These range from swapping memory from disk to termination of the process – all operating system decisions. In the case of the cellular telephone system, a basic event is the “handoff” when a subscriber passes out of range of its current base station and is passed off to another station.

To summarize, the above examples show several important aspects of multi-level systems and their subsystems. Large systems organize themselves hierarchically a means of managing complexity and achieving scalability. At each level there are subsystems that interact with other subsystems at the same level and may interact with subsystems at both higher and lower levels. Systems are defined by boundaries. Address spaces are homogeneous regions, defined by boundaries, whose life-cycle and exception handling is the responsibility of systems higher in the hierarchy. Various subsystems may collaborate to utilize the address space. Interaction requires that information pass across boundaries. Information passes between these subsystems as boundary crossing events that may trigger the interest of higher system levels. The work of the next section is to specify a model that captures the above features.

### 3. The Generalized Address Space Model

A simple conceptual model of an idealized address space is *outlined* in this section. The characteristics of address spaces identified in the previous section are restated as the minimum requirements for this general model:

- Supports hierarchical, layered, and/or nested configurations of component subsystems
- Boundaries can be identified and defined
- Supports an addressing or indexing scheme
- Is n-dimensional in general.

An address space does not exist for its own sake. It provides a structure to support components (objects, entities) and permit operations such as communications between components. The goal of the general model is to capture the basic “patterns” of component interaction within an address space. The following components are chosen to accomplish this:

**Address Space Manager (ASM):** The ASM manages and services the components residing within its assigned address space. It knows the types and locations (addresses) of all components. It knows the range of its addresses and can read from and write values to those addresses. It knows the addresses of its nearest neighboring ASM’s and the address of the ASM at the next level in the system. If it is asked to access an address outside of its range it will do one of two things. First, it can determine if the requested address lies within the range of addresses managed by one of its neighbors. In this case it will ask the neighboring, “horizontal”, ASM to perform the read or write on its behalf. Otherwise, the address is not within the range of a neighbor and it will ask the ASM at the next higher level, “vertically”, to provide the needed service (read or write a value).

**Boundary Manager (BM):** The BM handle boundary crossing events as directed by the ASM. Bound crossing events include reading and writing to addresses in other address spaces. An address space may have boundaries with different types of address spaces necessitating different types of BM. Also, there may be more than one type of boundary crossing for a given boundary. When called upon by the ASM, the BM establishes contact with a BM in the neighboring address space to provide the requested action, i.e. fetch a value. Use of components in lower layers in the hierarchy may be required.

**Services Manager (SM):** It is convenient for now to identify the role of a SM to provide anticipated functions such as persistence, transactions, translation, query, etc.

**Fixed User (FU):** In this simple model the only behavior of the FU is to read and write values to addresses. It has a fixed immutable address within the address space. Addresses for reading and writing are not constrained to be within the FU’s address space.

**Mobile User (MU):** The MU is a FU that can “move” between address spaces and still be written to (receive) values at its “home address”. This implies support for a scheme like Mobile IP in that the MU will be assigned a new address as it moves into a new address space, leave a “forwarding address” as it moves around, etc.

Again, the goal of this model is to capture the fundamental patterns of component interaction in an address space. An example will help. When an FU tries to read the value of a foreign address space, the ASM invokes a BM to handle fetching the value from the neighboring address space. The BM contacts the BM in the foreign address space. A number of operations (marshaling, translation, etc) results in the FU ultimately getting the value across the address space boundary. This is the essential pattern found in systems such as virtual memory and distributed programming (DCOM, CORBA). The notion of “vertical” as well as “horizontal” coordination among the hierarchy of ASM’s will be important later. The next step in the development of the model is to give it a proper addressing mechanism.

## 4. A Geometric Foundation

A geometric construct is chosen as the basis for addressing in the hierarchical, n-dimensional address space model. In searching for a sound basis for the address space model, a structure called the *Generalized Balanced Ternary (GBT)* was uncovered. This section gives the briefest possible overview of GBT, some background and a more detailed description is on line at [5] and [6].

GBT is a hierarchical, uniformly adjacent, addressing system defined on Euclidean n-space. It was developed by Lucas [7] and applied by him and others to support a range of scientific and engineering applications. (Vince offers a thorough treatment of addressing in self-replicating structures including GBT [8] and is highly recommended.) Lucas discovered an algebraic system permitting the construction of vector spaces in n-dimensions that support vector addition and multiplication. GBT is an algebra defined on a lattice of cells. This lattice is the hexagonal tiling in 2-dimensions. The addresses of cells are a generalization of the Knuth's *Balanced Ternary* [9] to 2-dimensions and is a base 7 system. The labeling of cells is geometrically arranged such that powers of two are maximally distant and uniform. This is done to simplify mathematical operations. A first level aggregate is shown in Figure 1. Note that 7 is used as the address of the overall aggregate.

At each hierarchical level cells are defined as aggregates of lower level cells. A second level aggregate is also shown in Figure 1 and is composed of a central cell and six surrounding cells. Now we look at how addition and multiplication are performed based on this system of addressing. From the level-1 aggregate we see that  $1+2 = 3$ ,  $1+4 = 5$ , and so on. However,  $1+3$  lies outside the level-1 aggregate. As shown in Figure 1, the sum of 1 and 3 being 34. Also shown is the sum  $51+66 = 40$ . Using the level-2 aggregate a table of "carries" and "remainders" for addition is easily constructed. Examples of multiplication are also shown in Figure 1. There, 34 multiplied by the unit-length vector 2 to yield 16. Note the effect of multiplication by the unit vector results in a  $60^\circ$  rotation. The second example is  $43 * 25 = 51$ . A "remainders" table for multiplication can be constructed and is equivalent to multiplication in the complex plane.

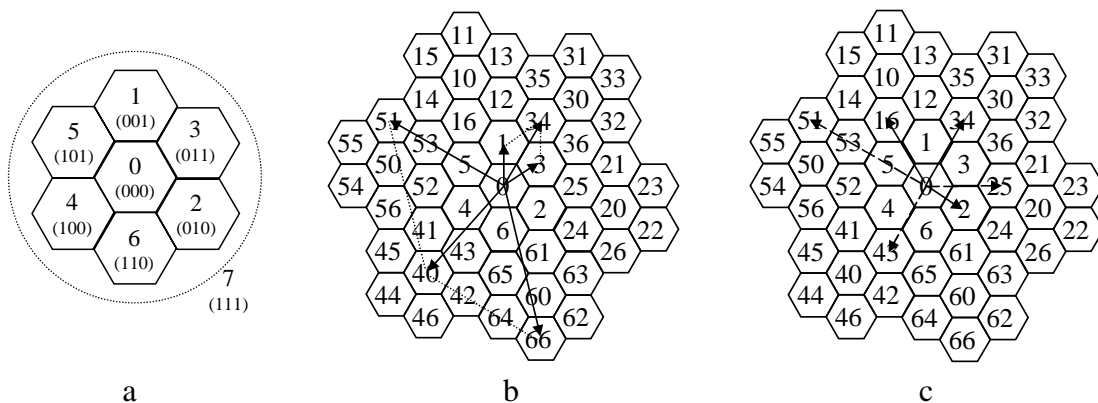


Figure 1. Level-1 aggregate (a), Level -2 aggregate showing addition (b) and multiplication (c).

It is seen from the above examples that a GBT address is a sequence of integers that specify a "path" to a particular cell. The addresses are written left to right with the most significant digit on the left. The most significant digit represents the highest level aggregate, the next digit the next level aggregate and so on. An address of the form 67, for example, represents the aggregate containing all lower cells whose first digit is 6, i.e. 60 - 66.

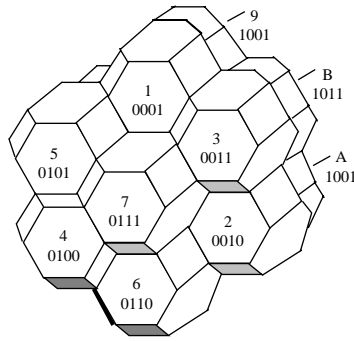


Figure 2. 3-Dimensional level-1 aggregate.

The GBT cells in 3-dimensions are truncated octahedra that pack 15 around 1. Thus a first level aggregate will consist of 16 cells as shown in Figure 2. The cells are labeled 0... E with 0 being the central cell and F referring to the overall aggregate (just as 7 is used in the 2-dimensional case). From the 2-dimensional examples of addressing, addition and multiplication it can be seen how similar operations can be developed for three and higher dimensions. Space does not permit further exposition of the GBT addressing system. The important concepts to take away from this section are that this structure is a hierarchical, uniformly adjacent, addressing system in n-dimensions that endows Euclidean space with an algebra that provides a vector space with “nice” properties such as addition and multiplication. This type of structure is required by engineering and scientific applications such as computer vision, image analysis and recognition, multi-variate data analysis, and many others.

## 5. Cellular Component Manifolds

The hope of this section is to show how components might be organized into computing and communicating systems based on the generalized address space model. First, some necessary terms and definitions are given. Szyperski [10] identifies three major tiers of component systems. The three tiers are defined below and their relationships are shown in Figure 3. Note the layered structure within a tier.

- A **component system architecture** consists of a set of platform decisions; a set of component frameworks; and an interoperation design for the component frameworks.
- A **component framework** is a dedicated and focused architecture, usually around a few key mechanisms, and a fixed set of policies for mechanisms at the component level.
- Software **components** are binary units of independent production, acquisition and deployment.

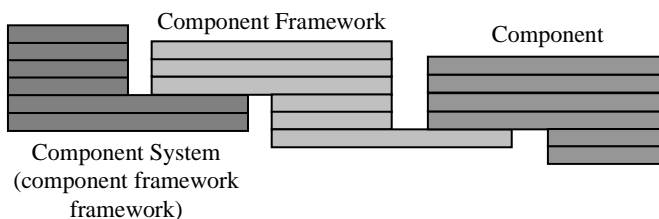


Figure 3. Component system architecture

The multi-tiered component system architecture of is Figure 3 redrawn in Figure 4 to show how components interoperate with each other directly and via frameworks. These are *instances* of runtime components and the dashed lines indicate direct and framework mediated communication. Component frameworks may appear as components within some contexts. That is, a component framework may offer an interface for use by components or by frameworks. The distinction is arbitrary in many cases.

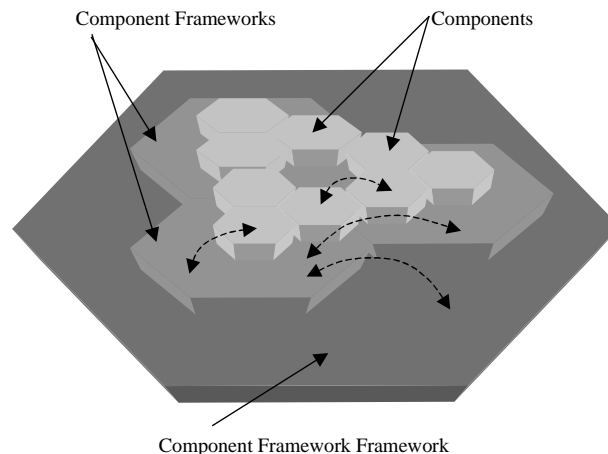


Figure 4. Component instance interoperations

Given the above, a **Cellular Component Manifold** is defined as:

*The “crystalline” structure formed by a distribution of components within a component framework whose organization is based on the generalized address space model where **all** addressing is in GBT.*

With these concepts established, a software architecture is now described. The three tiers are associated with specific system functionality as follows:

Component Framework Framework:	Virtual Machine
Component Framework:	Cellular Component Manifold
Component:	Component

The first tier, the component framework framework, is a Virtual Machine (VM). The VM is a virtual platform that encapsulates underlying, possibly heterogeneous, operating systems, networks and other resources. It is similar to the Java or Smalltalk VM’s. This layer provides services to instances of Cellular Component Manifold (CM). Most importantly it permits CM’s to be instantiated and activated. Other services include life-cycle management such as creation and allocation of resources (e.g. memory, cpu) to the components in the manifold and garbage collection. The VM has the ability to distribute the components across processors and networks.

Several research projects are developing prototypes whose advanced capabilities point the way toward the type of VM desired here. Two in particular are part of Microsoft’s Millennium next-generation distributed operation system effort. The Borg prototype is a Java VM that automatically distributes computation over clusters of machines in what is called the single system image (SSI) approach. Another is the Coign prototype that automatically partitions and distributes COM component-based applications over a network.

The next tier, the component framework tier, is composed of instances of CM. Instances of CM are realizations of the generalized address space model. They are hierarchical, n-dimensional, structures based on the geometrical-algebraic GBT system. CM provide for the organization of components into computing-communicating wholes by acting as containers for components and providing for component and framework interoperation. A minimal CM is instantiated in the VM with the following parameters:

- $N$  : number of dimensions
- $R_{1-n}$  : the range of addresses in each dimension
- $S$  : a specification for populating the address space with components.

Thus an instance of a CM is an address space with a define number of dimensions, definite range of addresses in each dimension, and some number of components located at specific addresses. The CM have an internal structure derived from the basic pattern of components discussed earlier (the Address Space Manager, Boundary Manager, etc.). On instantiation of a CM, the VM returns the address of the new CM. These addresses are globally unique and are used by other parts of the system to access the CM's interfaces. Also, the VM is used to resolve (external) addresses requested by a CM.

The VM plus a number of CM instances combine to produce a global virtual machine. This is a conceptual design for the generalized address space model. Implicit in this hierarchical model is the *interoperation design* that specifies how the various CM component frameworks interoperate within the overall system architecture – by reliance on the global address space model. The various levels of components within the system are a set of nested “plug-in” component manifolds. The instantiated CM can be thought of as a “component tilings” of the address space that are mapped onto the physical infrastructure resources by the VM. The basic pattern of address space components (the Address Space Manager, Boundary Manager, etc.) will become specific components suited to the particular task. These ideas are illustrated in Figure 5 where the rectangles are VM's and the ovals are CM's. As was mentioned earlier, it is somewhat arbitrary to distinguish between component manifolds and components. In the design presented here the Virtual machines could be instances of Component Manifolds.

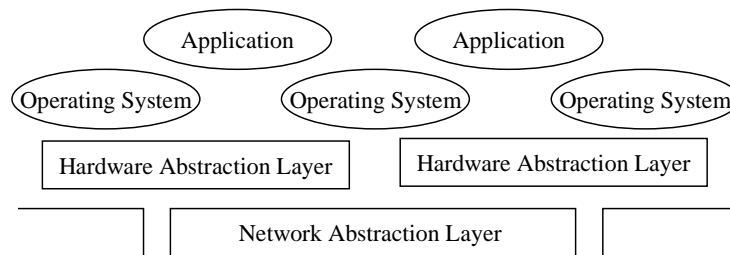


Figure 5. Hierarchy of VMs and CMs

A major goal of this approach is to have a global addressing scheme wherein the addresses (and the underlying mechanisms) perform useful services beyond current addressing schemes. For example, Microsoft 's COM system relies on the use of Globally Unique Identifiers (GUID), Interface Identifiers (IID), Class Identifiers (CLSID) and Category Identifiers (CATID). These are 128-bit numbers whose only feature is that they can be generated on separate machines and are guaranteed to be unique. The GBT-based addressing system could be used to generate such identifiers with some added benefits. The GBT addresses are indexes into an n-dimensional data structure that can store information about the indexed item. The address space of GUID's for a component architecture could be organized to store and access information on component versions, vendors, licensing, security, or any other information that is useful in enhancing the functionality of the system. Another addressing scheme that could possibly benefit from such an approach is Internet Protocol addressing now under revision.

## 6. Example Application

An example is developed to make concrete the ideas presented in the preceding sections. The intent is to illustrate how the Cellular Component Manifolds organization might be used in a real world application. The specific problem of handoff management of mobile users in *heterogeneous wireless networks* is chosen for this purpose. This section relies on considerable work done by the Bay Area Research Wireless Access Network (BARWAN) project [3].

The BARWAN project takes as its main task development of protocols for use in wireless overlay networks. Wireless overlay networks are hierarchical, physically overlapping networks composed from different types of wireless systems. For example, in-room IR and in-building RF. These various systems are developed by different vendors, use different protocol and antenna/modems. The BARWAN effort has prototyped an “Overlay IP” protocol that permits mobile hosts to roam seamlessly though this hierarchical, heterogeneous wireless network. A major part of their work focuses on how to make low-latency handoffs between the different networks. A simplified version of their testbed is described next for purposes of this example.

Assume a campus setting with a number of buildings. There is a campus-area packet relay network that covers the entire campus with a bandwidth of 64 kbps and latency (between mobile host and wireline) on the order of 100 ms. The cell size is 500 m. Each level within a building has a spread spectrum RF network with 1 Mbps bandwidth, 10ms latency, and 100 m cell size. Each room has an IR network that can support 5 users with near wireline performance. Further assume a typical campus-wide wireline network gatewayed to the wireless networks. Students are supplied PDAs with modems capable of communicating on and switching between the two RF nets and IR net. The students are set in motion.

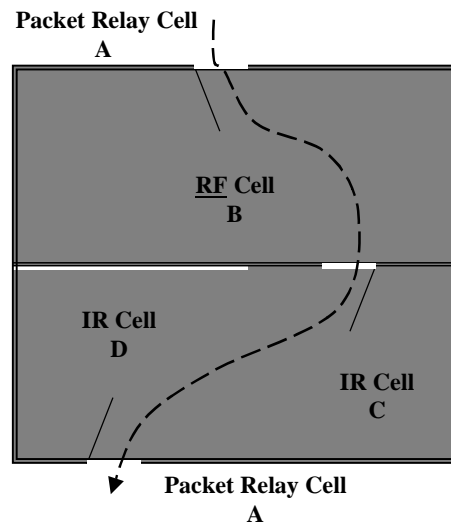


Figure 6. Typical building

A typical building fitted with RF and IR networks is shown in Figure 6. The building is in campus packet relay network cell A. There is one in-building RF network, cell B, and two in-building IR network cells D and C. The path of a student with a PDA (mobile host) requiring continuous service is also shown. Given this setting and scenario, the conceptual design for a software system to manage handoffs between the different networks is developed.

The goal of the design is to minimize handoff latency. This latency is bounded by *discovery time*: the time it takes a mobile host (MH) to realize it has changed cells. Typical (homogeneous) wireless systems rely on signal strength to manage handoff. This introduces unacceptable discovery times in wireless overlay systems. The design here relies on knowledge of the environment to anticipate cell changes and initiate handoffs. The system consists of instances of Cellular Component Manifolds (CM) that implement a model of the physical address space of the wireless networks. These are called Handoff Controllers (HC) and they are one of the modules comprising the overall network management system.

The basic design for a HC is as follows. It is a CM based directly on the “pattern” of the components in the general address space model presented in Section 3. The components are specialized for this purpose:

**Address Space Manager (ASM):** Implements a model of the physical address space. Its internal GBT addressing is scaled to the specified resolution of the actual geographic and geometric system being modeled. Important environmental entities are represented as components within this model's address space. These component models will be buildings, rooms, furnishings, equipment and mobile hosts.

**Boundary Manager (BM):** Handles boundary crossing events. The BMs are tailored to the types of wireless cell boundaries that exist in the system. They operate under control of the ASM.

There are three types of HCs for this system corresponding to the three types of wireless networks. The campus Packet Relay HC (PR-HC), the in-building RF HC (RF-HC) and the in-building IR HC (IR-HC). For purposes of this example they are instantiated as 2-dimensional address spaces with ranges in each dimension appropriate to the geometry of their cells. The components that populate them correspond to physical entities important to modeling the handoff. In the case of the PR-HC, a basic hexagonal cell of 500 m diameter is modeled at 10 m resolution. Included are location and geometry of buildings. Information about the boundaries of other wireless systems within this cell is also needed. The RF-HC and IR-HC are also 2-dimensional and model the interior of the building at 1 m resolution along with any specifics concerning furnishings, etc. Also, information on wireless boundaries is included. Further, the location of mobile hosts is known at all times and dynamically updated in their current HC.

The operation of these HCs, based on the path of the roaming student shown in Figure 6, is now discussed. The path of the MH causes a number of handoffs:

**Packet Relay Cell A → RF Cell B:** The PR-HC for cell A is servicing the MH. It is also tracking the position and speed of the MH. Based on this information the PR-HC anticipates that the MH may enter the building. The ASM (of the PR-HC) instructs the BM that manages the packet relay to RF boundary to take interest in the MH. The BM contacts the BM of the RF-HC of cell B and informs it that a handoff may take place. This results in the formation of a multicast group containing cells A and B. Thus packets for the MH are delivered simultaneously to both cells. If the MH enters the building, the handoff is accomplished. The BM of RF-HC registers the new MH with the ASM. The packets are already buffered in cell B and MH switches its modem to RF. This is a *downward vertical* handoff.

**RF Cell B → IR Cell C:** The MH is a member of the RF network of cell B and is moving toward the door. The ASM (of RF-HC) informs the BM and a sequence of events similar to those described above result in a *downward vertical* handoff to IR cell C.

**IR Cell C → IR Cell D:** Here the MH moves between two IR cells in a large room. The actions are similar to those above except that signal strength is also monitored to effect the handoff. This is a conventional *horizontal* handoff.

**IR Cell D → Packet Relay Cell A:** The IR-HC for cell D notices the MH is headed toward the door leading outside the building. The BM responsible for that boundary event is informed. The BM contacts the BM for the packet radio cell A and multicast group is formed in anticipation of the MH joining cell A. An *upward vertical* handoff takes place as the MH leaves the building.

The example illustrates a number of points about the component architecture. The handoff controllers are instances of Cellular Component Manifolds tailored to model the environment of a hierarchical, heterogeneous, wireless network. The basic pattern of the underlying generalized address space model and its components seem to map well on to this particular task. The handoff controllers combine to form a hierarchical system of nested controllers. In a complete system based on this approach *all* addressing could be made consistent. That is all "addressable" entities would be components within some Cellular Component Manifold. Also, not brought out above is the geometric modeling inherent in the controllers. The address space of the handoff controller is a vector space. This permits complete geometric modeling of the environment. A 2-dimensional system was used in the example for simplicity. In a real implementation it could easily be 4 or more dimensions, e.g. 3 dimensions for space and one for time.

## 7. Conclusion

Little is known about how to build component systems. This paper presents first thoughts on how the concept of address space can be generalized to serve as a basis for a component system architecture. The motivation for proposing what may appear to be a rather complex software architecture is the goal of supporting both advanced component systems as well as a wide range of applications in a single, consistent, general-purpose component framework. The algebraic/geometric basis of the architecture provides, at a fundamental level, support for applications such as: image analysis [11], target recognition [12], geographic information systems and multi-variate data analysis in general [13], and fluid dynamics [14] to name a few. In addition to providing a convenient modeling environment for application developers, many of these “applications” will be necessary to achieve advanced *systems* functionality. In particular the need to manage an increasingly complex information infrastructure will require the use of many of these types of applications. In short, all of these engineering approaches will be needed to permit the development of executable models of the system itself. The example of Section 6 is an illustration.

## 8. Acknowledgements

Supported in part by the United States Defence Advanced Research Projects Agency and by a Co-operative Research Centre Program through the Department of the Prime Minister and Cabinet of Australia. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, expressed or implied, of the Defence Advanced Projects Research Agency or the U.S. Government.

## References

1. Jini, *JavaSoft*. <http://www.javasoft.com/products/jini/>, 1998.
2. Millennium, *Microsoft*. <http://www.research.microsoft.com/sn/Millennium/>, 1998.
3. Katz, R. and E. Brewer. *The Case for Wireless Overlay Networks*. in *Proceedings 1996 SPIE Conference on Multimedia and Networking, MMCM '96*. 1996. San Jose, CA.
4. Lindstrom, A., J. Rosenberg, and A. Dearle. *Grand unified theory of address spaces*. in *Proceedings of the Workshop on Hot Topics in Operating Systems, HOTOS*. 1995. Orcas Island, WA, USA: IEEE.
5. Herring, C., *Geometrical Background: The Premutahedron*. <http://www.dstc.edu.au/TU/staff/herring/Appendix-A.pdf>, 1998.
6. Herring, C., *The Generalized Balanced Ternary Addressing System*. <http://www.dstc.edu.au/TU/staff/herring/Appendix-B.pdf>, 1998.
7. Lucas, D., *A multiplication in N-space*. *Proceedings of the American Mathematical Society*, 1979. **74**(1): p. 1-8.
8. Vince, A., *Replicating Tessellations*. *SAIM Journal of Discrete Mathematics*, 1993. **6**(3): p. 501-521.
9. Knuth, D.E., *The art of computer programming*. Vol. 2, Seminumerical algorithms. 1969, Reading, MA: Addison-Wesley.
10. Szyperki, C., *Component Software*. 1998, New York: Addison-Wesley.
11. Gibson, L. and D. Lucas, *Vectorization of Raster Images Using Hierarchical Methods*. *Computer Graphics and Image Processing*, 1982. **20**: p. 82-89.
12. Gibson, L. and D. Lucas. *Pyramid Algorithms for Automatic Target Recognition*. in *Proceedings of the IEEE NAECON*. 1986. Dayton, OH.
13. Gibson, L. and D. Lucas. *Spatial Data Processing Using Generalized Balanced Ternary*. in *Proceedings of the IEEE Conference on Pattern Recognition and Image Analysis*. 1982.
14. Hasslacher, B., *Discrete Fluids*. Los Alamos Science, 1987. **15**.