THE CELLULAR HIERARCHY INFORMATION MODELING ENVIRONMENT

BILAL KHAN DARDO D. KLEINER DAVID TALMAGE

Advanced Engineering & Sciences, ITT Industries
at the Center for Computational Sciences, Naval Research Laboratory,
Washington D.C., U.S.A.

Keywords: Distributed database, multi-agent system, distributed computing

1 OVERVIEW

Within the CHIME, information is organized into units called *cells*. The cell is a new type of atomic object, which extends existing distributed object paradigms, such as agents, actors, and proxies. In what follows, we describe some of the important aspects and features of cells.

1.1 CHIME AS OPEN MULTI-AGENT SYSTEM

As a first approximation, one may consider a cell to be a weak agent in the sense proposed by Wooldridge and Jennings [7], i.e. the cell is an object that is autonomous, social, reactive and pro-active. A machine indicates its willingness to host cells by running a special *depot* program. Each cell within the CHIME then resides in some depot process; each depot process manages a set of cells (see figure 1).

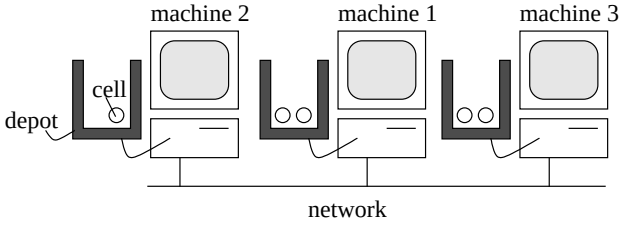

Figure 1: Cells residing in a collection of depots.

The *physical address* of a cell is defined to be the IP address of the machine on which its depot process is running. It is possible for a cell to *migrate* between depot processes on different machines, e.g. a cell can request to change its physical address. In this sense, the CHIME may be considered to be an open multi-agent system.

We use active objects (cells) as the basic quantum of information because:

> **Thesis 1:** *Information must be managed at the level of the active processes generating it,* not *at the level of passive data that is the byproduct of such processes.*

1.2 CHIME AS MULTI-RESOLUTION DISTRIBUTED DATABASE

Unlike traditional agents (see e.g. [2], [5]), the cell is a composite object, i.e. each cell is made up of *sub-cells*, and each cell is a sub-cell of some *super-cell*. Thus cells are organized in a tree, whose nodes (the cells) are distributed across the depots (see figure 2). A cell encapsulates its sub-cells, and is responsible for providing an interface to their information in an aggregated form.

While the Web does permit the organization of information using a directory structure or a tree of hyper-linked pages, there is a fundamental difference: *the Web does not itself interpret the structure of the hyper-links*. In contrast, the CHIME interprets cell hierarchy as follows: *sub-cells represent the internal contents of their super-cell.* Accordingly, if the a user wishes to obtain information at a higher resolution or to see details not presently revealed by a cell, the user simply enters into the cell. The CHIME thus facilitates the efficient partitioning of information, and supports multi-resolution representations.

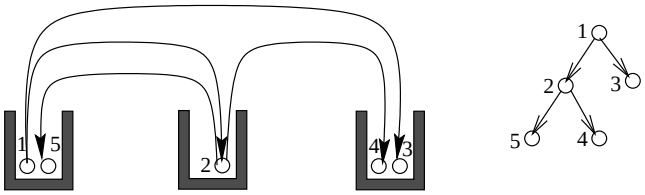

Figure 2: The tree of cells distributed across depots.

Each cell has a name, which remains fixed over its entire lifetime. The *logical address* of a cell *A* is defined to be the dotted sequence of cell names, starting at the root of the CHIME cell tree and ending at the cell *A*. It is possible for a cell to *relocate* within the tree, e.g. a cell can request to change its logical address. Relocation of a cell does not alter its relationships with its sub-cells. Note that relocation and migration are independent notions of mobility.

We use composite objects (cells) as the basic quantum of information because:

> **Thesis 2:** *A scalable solution to the problems of location, management and annotation of information requires maintaining information at multiple*

resolutions; aggregation must be supported.

1.3 CHIME AS HIERARCHICAL SPATIAL DECOMPOSITION

So far, the sub-cells of a cell have been considered equal peers in their relationship to their common parent. This is certainly too restrictive, since a large part of information is encoded in the relative positions of data. In analogy, consider how passive data in web pages is rendered meaningful by its relative placement. Accordingly, each CHIME cell is assigned a 3-dimensional volume by its parent, and is in turn responsible for allocating a region of this volume to each of its sub-cells. The 3-dimensional volume that a cell occupies is referred to as its *geometry* or *spatial position* (see figure 3). While the logical address of a cell indicates its location in the CHIME tree, a cell's spatial position further specifies the relationship of the cell within its super-cell. By requiring each cell to occupy some volume in 3-space, the CHIME is able to efficiently compute distances between cells. This allows the CHIME to support the notion of information *locality*, i.e. it facilitates being able to encode and assess the relative nearness or farness of cells.

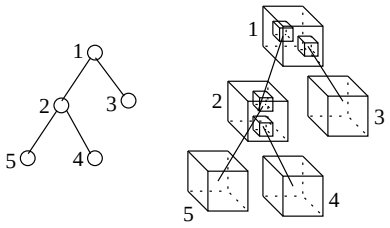

Figure 3: Geometries of cells.

It is possible for a cell to *reposition* itself within its super-cell's coordinate system, e.g. a cell can request to change its spatial position within its super-cell.

We use spatial objects (cells) as the basic quantum of information because:

> **Thesis 3:** *Meaningful organization of information requires specification of geometric relationships (e.g. proximity) between information quanta.*

1.4 CHIME AS INTERCONNECTED ACTIVE INFORMATION

Cells are active objects capable of intercommunication. Every cell can communicate (via a message passing interface) with any of its sub-cells and with its super-cell. Beyond this, if a cell A wishes to communicate with some cell B (that is not a sub-cell and not a super-cell), it is necessary for A to own a *model* of B. This model, which we denote as $\mathcal{M}(B)$, is an object that is created dynamically by cell B and delivered by the CHIME to cell A. The CHIME maintains a bidirectional channel between the model $\mathcal{M}(B)$ and the cell B which created it. This channel can be used to stream data from B to $\mathcal{M}(B)$, and also provides bidirectional message delivery services between B and $\mathcal{M}(B)$. The operation of this channel is transparent to changes in the physical or logical addresses of the cells A and B. The model $\mathcal{M}(B)$ in turn acts as a mediator [4], providing its owner A with reliable message-passing to B (see figure 4).

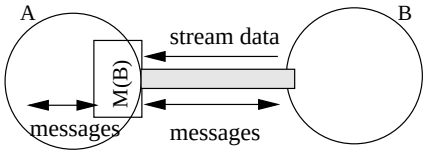

Figure 4: Cell A communicates with cell B via a model $\mathcal{M}(B)$.

In view of the above description, we regard cells that support communication with other cells as model factories [4]. While each cell may both create and own an unlimited number of models, every model has a privileged relationship with precisely two cells: the cell that created it, and the cell that owns it. Unlike traditional proxies [6], models are active objects, and may contain significant state information. Specifically, there may be ongoing bidirectional communication between a cell and its model, outside of what is triggered by specific activity of the model's owning cell.

We use interconnected active objects (cells) as the basic quantum of information because:

> **Thesis 4:** *Information must be modeled as a dynamical system.*

1.4.1 GRACEFUL DEGRADATION OF INTERCONNECTIVITY

To ensure scalable communication overhead, the CHIME does not support arbitrary communication patterns between cells. Specifically, a cell A may only own models of cells that are *reachable* from A. The set of cells reachable from A is denoted $Re(A)$, and consists of: the siblings of A, the siblings of A's super-cell, the siblings of A's super-super-cell, and so on up to a specifiable number of levels. Figure 5 distinguishes cells that are in $Re(A)$ by shading them black.

Recall that the CHIME interprets cell hierarchy by considering sub-cells to be higher-resolution representations of regions of their common super-cell. Conversely, a super-cell is expected to be an aggregator, generating a low-resolution version of the information in its sub-cells. In light of this, the membership of $Re(A)$ can then be seen to satisfy the following appealing criterion: A gets models of nearby cells with high resolution information, and it models of far-away cells with low resolution information. Indeed in general, *the greater the distance to A, the lower the resolution of the model obtained.*

The actual set of models owned by a cell A may be much smaller than $Re(A)$, because (1) the set of siblings under consideration at each level may be pruned at the discretion of their super-cell at the next higher level, and (2) any reachable cell can opt out of providing a model to A. The set of models owned by A is maintained automatically by the CHIME's

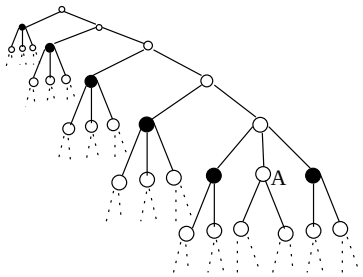

Figure 5: Reachability pruning for graceful degradation of interconnectivity.

model collection FSMs. In particular, the set of models is updated whenever A relocates, and when other cells relocate in a manner that causes them to enter or leave the set $Re(A)$.

We prune the possible inter-connectivities of cells because:

> **Thesis 5:** *For scalability, information interconnectivity must degrade gracefully with distance.*

2 CHIME AS AN INFORMATION WEB

We summarize some of the important aspects of the CHIME:

1. The models that a cell returns are based on the logical address, the physical address, the spatial position, and declared capabilities of the requester.

2. Using models, a cell can interact with other cells that are in its immediate proximity, *as well as* cells in the proximity of its super-cell, cells in the proximity of its super-super-cell, etc. The cumulative outcome being that a cell has access to information at a resolution that decays smoothly with distance from its present location[1].

3. A cell typically interacts with models of its sub-cells, and *aggregates* their state in a suitable manner.

Any cell which incorporates a human user as part of its control logic is called a *user cell*. One example of a user cell is the CHIME browser. The browser is a featureless application whose only purpose is to allocate resources of the machine to the models it owns, then (1) request the models it owns to render themselves using these resources, and (2) to respond to the users requests to relocate the browser within the CHIME tree. Presently, models must render themselves by interacting directly with the OS; this restricts the physical address of the CHIME browser cell to be the address of the user's machine.

3 DESIGN

3.1 THE DEPOT

The depot is the process which serves as an an object repository for running cells, and is presently implemented using a Java MBean server. In particular, each depot supervises the execution of cell code in the host's environment, and provides its resident cells with an interface to CHIME services. These services include:

1. *migration interface* for changing the physical address of a cell.

2. *relocation interface* for changing the logical address of a cell.

3. *repositioning interface* for changing the spatial position of a cell.

4. *automatic collection of models* on behalf of a cell.

Each of the first three services listed above is typically requested by the cell itself. When a cell invokes any of these interfaces, a suitable finite state machine is instantiated in the depot's protocol layer. This finite state machine handles all inter-depot messages required to carry out the request. The protocols are implemented using the services of JavaSpaces, which acts as external transaction-based persistent storage. Sample executions of these protocols appear later (see figures 8, 9, and 10).

The fourth service offered by the depot is automatic model collection. This service operates by creating one model-collection FSM for each of the depot's resident cells. A model-collection FSM monitors the CHIME hierarchy on behalf of the cell it represents, and acts to detect any changes in the cell's reachable set. (Recall that cells may enter or leave the reachable set as a result of relocations within the tree.) When a model-collection FSM detects new members entering [existing members leaving] the reachable set, it commences procedures to acquire new models [destroy previously acquired models] as appropriate.

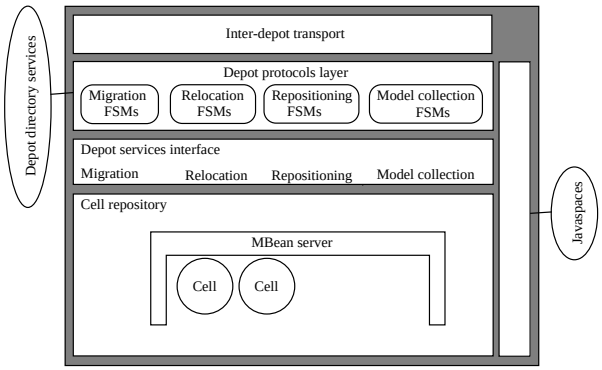

Figure 6: The architecture of a depot.

3.2 THE CELL

A cell *is* agent code, implemented as a serializable Java thread group; it is the quantum of active data stored in an CHIME depot. The cell contains a Model Manager and message passing interfaces to the each model it has created and

[1] Here, by distance, we mean distance within the cell hierarchy.

each model that it owns. Together, this collection of interfaces permit the cell to communicate with other reachable cells.

The Model Manager contains a customizable Model Factory; this factory is consulted by the Model Collection FSM whenever a model of the cell needs to be made. In addition, the Model Manager is responsible for maintaining two tables: one containing interfaces to the models it has created, and the other containing the models that it owns (i.e. models that were made for it by other cells). The Model Manager inserts and removes models and interfaces from these tables as directed by the associated Model Collection FSM which resides within the depot.

Defining a custom cell requires providing a concrete implementation for the Reasoning module and the Model Factory. While the Model Factory defines the cell's response to communication attempts by other cells, the Reasoning module defines the cell's response to the passage of time. The Reasoning module has access to CHIME services for cell migration, relocation, and repositioning; these services are brokered by the cell's depot. The Reasoning module also has access to persistent storage to facilitate the implementation of safe multi-cell distributed procedures.

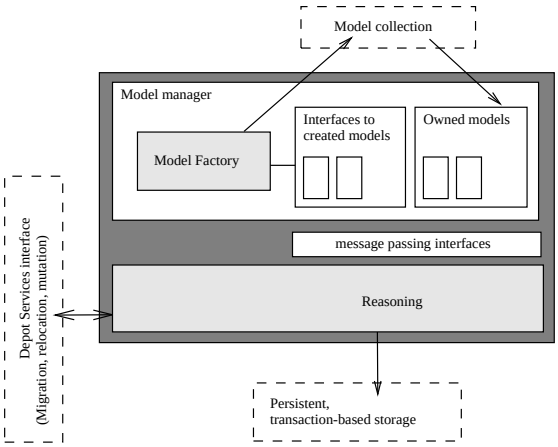

Figure 7: The architecture of a cell.

3.3 THE CHIME PROTOCOLS

CHIME cells support the notion of migration, i.e. the entire state of a cell can be frozen, transported to another host, re-instantiated in the depot there, and revived at its new residence. Figure 8 depicts the interactions necessary for a successful migration scenario that is initiated by the cell itself. Utilizing the services of its current depot, the cell requests to be migrated to a new depot. The local depot establishes a transaction context under which it will service the cell's request. First, the target depot is asked to create a deep copy of the cell. This requires instantiating an object of the corresponding type and copying the migrating cell's data members (state). When the target depot has completed this, the requesting depot can proceed to terminate the execution of the original cell. The new copy is then started on the target depot, thus reviving the cell at its new residence. At this point, the original depot can destroy the original cell and commit the transaction. Because the inter-depot transport layer is implemented using Java RMI [3], and RMI does not serialize the context of running Java threads, cells must currently be written to be re-entrant, to the extent that they must expect to be restarted at the new depot after migration.

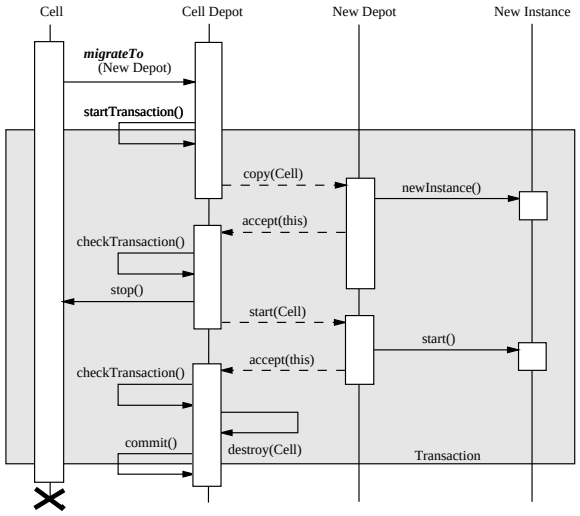

Figure 8: Interaction diagram depicting how a cell migrates itself.

The CHIME also supports two additional concepts of cell mobility: CHIME cells maintain both a logical location in the CHIME tree, and each cell occupies a specific spatial position within its parent cell's volume. Relocation (changing in logical address) and repositioning (changing spatial position) are not independent notions of mobility: If cell repositioning results in the collision of cell boundaries, the CHIME will automatically attempt to perform the necessary relocation of cells. Specifically, if (as a result of repositioning) a cell's spatial position collides with the position of one of its siblings, the CHIME will attempt to relocate the repositioned cell so as to make it a sub-cell of its (former) sibling. Likewise, if (as a result of repositioning) a cell's spatial position collides with the boundary of its super-cell, the CHIME will attempt to relocate the repositioned cell so as to make it a sibling of its (former) super-cell. Figure 9 and 10 depict the interactions that occur when a cell attempts to relocate or reposition itself, respectively.

Note that all CHIME protocols are executed under a JavaSpaces transaction manager, so that if a particular step in the protocol fails, the side effects of the entire operation can be rolled back. The CHIME relies on JavaSpaces as a persistent back-end to assist in preservation of the well-defined ACID properties required to recover from failure conditions, and to facilitate appropriate commit and rollback actions.

3.4 THE BROWSER

The CHIME browser performs the same function of today's popular browser software (for example, Netscape or Opera)

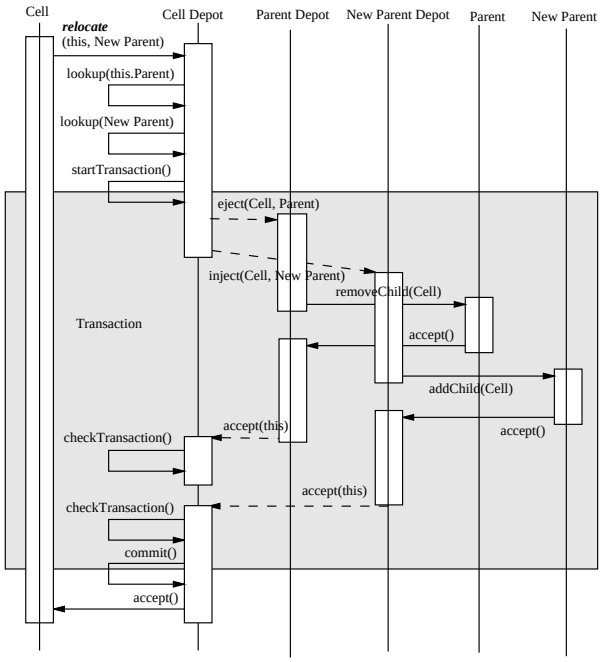

Figure 9: Interaction diagram depicting how a cell relocates itself.

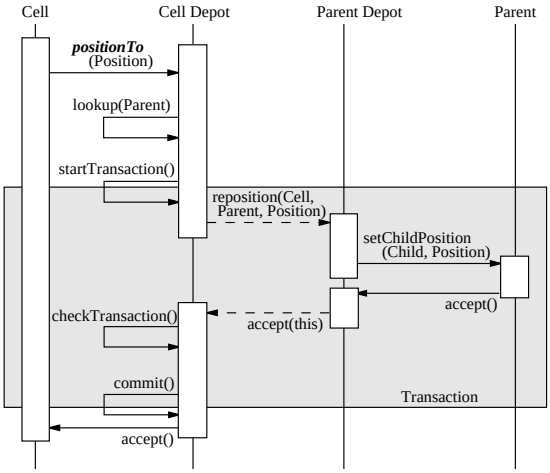

Figure 10: Interaction diagram depicting how a cell repositions itself.

in that it presents a graphical view of the information elements to the user, and accepts and acts upon user input in order to navigate the web of data. However, in the CHIME, the browser is an active participant in the data model itself. The CHIME browser is itself a cell, and makes use of logical mobility in the CHIME tree in order to navigate through the cells that make up the CHIME universe. The only present limitation is that the browser cell cannot migrate to other depots, and must remain fixed at its starting point where the user resides. This is because the primary function of the browser cell is to act as a gateway to the operating system on the user's machine. Allowing the browser cell to migrate would require wrapping the host platform's operating system in a cell, which is beyond the scope of the present work[2].

The model collection FSM operating on behalf of the browser requests models from cells within its reachable set. The model collection FSM advertises local hardware capabilities, and collected models are expected to be renderable using this hardware (specifically, to implement a Render() method). This is not typically required of models that are destined to non-browser cells. The CHIME browser functions by (1) periodically calling the Render() method of its owned models, so they may render themselves onto the output devices available at the host, and (2) to accept events from local input devices for relay to the browser's internal Reasoning module, where suitable repositioning and relocation actions are triggered.

4 CONCLUSION

In this paper, we provided an overview of the motivating principles and design of the Cellular Hierarchy Information Modeling Environment (CHIME), and assessed its advantages and merits as a foundational framework for a next generation information web.

References

[1] *UML — Unified Modeling Language, version 1.0*. Rational Software Corp., 1997.

[2] B. Burg. Agent naming. *FIPA Spec - 1999 - 017 - 0.1*, 1999.

[3] T. B. Downing. *Java RMI: Remote Method Invocation*. IDG Books Worldwide, Inc, 1998.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns — Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, 1995.

[5] N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes. Hive: Distributed agents for networking things. In *Proceedings of ASA/MA'99, the First International Symposium on Agent Systems and Applications and Third International Symposium on Mobile Agents*, 1999.

[2]This functionality can be achieved via the use of thin-client virtual viewers (e.g. VNC)—the CHIME neither restricts nor enables this capability.

[6] J. Nelson. *Programming Mobile Objects with Java*. John Wiley and Sons, Inc, 1999.

[7] M. Wooldridge and N. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.