

Tuple Space Coordination Across Space & Time

Gruia-Catalin Roman¹, Radu Handorean², and Rohan Sen¹

¹ Department of Computer Science and Engineering
Washington University in St. Louis
Campus Box 1045, One Brookings Drive
St. Louis, MO 63130-4899, USA

² Qualcomm Inc.
6180 Spine Road, Boulder, CO 80301

Abstract. CAST is a coordination model designed to support interactions among agents executing on hosts that make up a mobile ad hoc network (MANET). From an application programmer's point of view, CAST makes it possible for operations to be executed at arbitrary locations in space, at prescribed times which may be in the future, and on remote hosts even when no end-to-end connected route exists between the initiator and target(s) of the operation. To accomplish this, CAST assumes that each host moves in space in accordance with a motion profile which is accurate but which at any given time extends into the future for a limited duration. These motion profiles are freely exchanged among hosts in the network through a gossiping protocol. Knowledge about the motion profiles of the other hosts in the network allows for source routing of operation requests and replies over disconnected routes. In this paper, we present the CAST model and its formalization. We also discuss the feasibility of realizing this model.

1 Introduction

Mobile Ad hoc Networks (MANETs) are a special class of wireless networks, which do not rely on any external infrastructure and are formed opportunistically among physically mobile hosts. By definition, a MANET is an open, dynamic environment where hosts may join or leave the network of their own volition and connectivity between host pairs is transient, requiring a decoupled style of computing. Developing applications for MANETs is especially challenging because peer-to-peer interactions between hosts can be short lived and the hosts participating in the computation change often and in an unpredictable fashion.

Coordination middleware is a software solution that has been proven to be able to handle the open environment and decoupled interactions as evidenced by systems such as MARS [1] and KLAIM [2], designed for wired settings, and systems such as LIME [3], Limone [4] and Ara [5] which are targeted to mobile settings. However, mobile settings, and MANETs in particular impose additional challenges which cannot be handled by the current generation of coordination models. The key problem is that current technology has a very restricted definition of *reachability* in the

dimensions of *space* (hosts must be strategically located in space so as to have access to a route to the destination) as well as *time* (the routes are only valid at the current instant). Handling more sophisticated applications requires us to develop models that expand the present narrow interpretation of reachability, allowing more predictable interactions with larger numbers of participants in a MANET.

CAST (Coordination Across Space and Time) is a new kind of coordination model that extends the reach of coordination activities across space and time by addressing restrictions imposed by current models. In the spatial domain, we use disconnected routing, a type of routing algorithm which does not enforce end-to-end connectivity between the source and destination of a message. This type of routing algorithm is similar to source routing in that the complete sequence of intermediary hosts is specified in the message header. The difference lies in the manner in which the hosts that form the route are selected. We use information about the motion of hosts (exchanged among hosts using a gossiping protocol) to compute intervals of pairwise connectivity among hosts extending from the present moment for a fixed amount of time into the future. The hosts forming the route are selected such that the sequence of intervals of pairwise connectivity between them will be sufficient to deliver a message from the source to the destination. The message is temporarily buffered on each host when the incoming communication window and outgoing communication window do not coincide, allowing coordination among hosts that are never in direct pairwise contact.

CAST also addresses the temporal domain by offering operations that have an explicit space and time value associated with it. The operations are moved to the required location using the mechanism described above and held in an inactive status until the activation time is reached. If there is no route which can get the operation to the required location by the specified time, the middleware can immediately raise an exception and reject the operation. Spatiotemporal operations allow coordination to occur across the reaches of space and time, thereby expanding the reach of the model.

In this paper, we describe the CAST model and its operation. The model assumes the use of a gossiping protocol to discover knowledge about the motion of other hosts which is stored in a local knowledge base and used to compute disconnected routes. Atop this, we offer Linda-like coordination operations via a simple API. The remainder of the paper is organized as follows. Section 2 provides background and a motivating example. Section 3 formally describes the key concept of reachability upon which disconnected routes are built, while Section 4 describes the semantics of the operations we support. Section 5 covers related work in coordination models before we conclude in Section 6.

2 Background and Motivation

Typically, a coordination model is characterized by its use of a shared dataspace that offers the following operations: (1) **out**, which places data in the shared space, (2) **in** which removes data from the shared space,

and (3) `rd` which creates a local copy of some data in the shared space. An agent that wishes to interact with another agent places data in the shared space which is subsequently retrieved by the target agent, thereby completing the interaction. This basic functionality, while sufficient for supporting applications that need simple message passing, is not suited to more sophisticated scenarios, specifically those where space and time are an inherent facet of the application.

Consider the example of a construction site where many people work in a highly dynamic environment. Each person carries a PDA that serves as a multi-purpose mobile computing platform. No fixed computing resources are available since there is no safe, suitable location for them. SynchroTask is an application that is used by construction supervisors on-site to manage day to day issues. SynchroTask allows supervisors and workers to manage and exchange project activities through opportunistic sharing of information.

Take for instance, the case of two shift supervisors needing to pass on lists of outstanding concerns. The problem is that shifts seldom overlap due to lunch breaks, etc. However, the construction site does have a rudimentary ready room which is used by the crew to rest and almost always has someone occupying it. The configuration is shown pictorially in Figure 1.

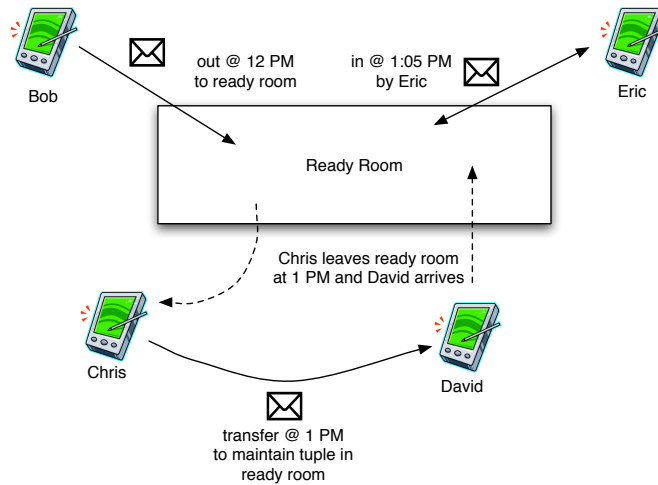


Fig. 1. Example of CAST in action

Towards the end of the shift, Bob uses SynchroTask to make notes on his PDA which lists the tasks requiring attention on a priority basis by the supervisor of the afternoon shift. As he leaves at 12 PM, he asks SynchroTask to forward the list to the afternoon supervisor. Since the afternoon supervisor has not yet arrived, SynchroTask requests that CAST, the middleware atop which it is executing, to store the information in a location where the incoming supervisor can easily retrieve it.

The information is stored in the ready room. When Eric arrives to supervise the next shift at 1:10 PM, he retrieves the information from the ready room. Supporting such interactions requires features not available in current coordination technology. The next two sections describe the CAST model that is designed to support applications such as Synchrono-Task.

3 A Model for Disconnected Coordination

The notion of coordinating across time and space is the distinguishing feature of CAST. By coordination we mean the execution of Linda-like operations on tuple spaces that may be either local or remote. For simplicity, we will assume that each tuple space is uniquely associated with a particular mobile host; this allows us to talk about coordination among hosts while ignoring the structural aspects of the host software. The scope of the coordination is controlled by which remote hosts can be reached at any given time. CAST assumes that hosts move according to some locally controlled plan called a motion profile, i.e., for a finite duration of time the host knows where it is heading and will not change its mind mid course. At first glance, the concept of using motion profiles might appear restrictive and impractical. However, motion profiles can be built automatically from individual schedules, thus avoiding the need for additional input by the user. Additionally, motion profiles are not practical only when the motion of hosts is completely random. When motion has even a simple pattern, motion profiles can be effectively exploited. Information about motion profiles is shared freely among hosts thus allowing them to build an egocentric view of where hosts might be in the future. Each host maintains such information in a *knowledge base*, which is a collection of the motion profiles of other hosts that the reference host is aware of. It is this information that makes it possible to identify disconnected routes which are used to transfer data and operation requests to a wider set of hosts than otherwise might be possible.

By exploiting disconnected routes, CAST makes it possible to reach out into the future and to coordinate across expanses of space. Coordination across time is associated with the ability to specify a lifetime for both data and operations in terms of a starting and an ending point. Coordination across space relates to the ability to identify a specific location or area in which the operation or the data is to exist, either initially or throughout its lifetime; the latter situation is complicated by the mobility of hosts in and out of the area of interest.

At this point it is reasonable to ponder whether the central enabling notion of computing and disseminating motion profile is reasonable. There exist some situations in which hosts follow prescribed patterns over extended periods of time because the application itself demands it: explorers follow a certain plan for safety and in order to accomplish their daily tasks; unmanned vehicles are restricted to a specific mission plan; robots are engaged in repetitive and redefined activities; office workers make their calendars public, etc. Despite these considerations, we must acknowledge that a study of coordination under uncertainty of motion profiles is intellectually exciting, but is out of the scope of this paper.

The remainder of this section is dedicated to a formal abstract description of the CAST model. We present the formalization initially from a global oracular point of view, where all available knowledge in the system is known to us. We show how the system evolves from this perspective. We then examine the model from an egocentric host perspective, where the available knowledge is a subset of the knowledge that exists in the system as a whole. A discussion of how to extend this formalization to cover the set of operations available in CAST is postponed to Section 4.

3.1 Underlying Computational Model

The core concept in our definition of the CAST model is the notion of motion profile. At the most basic level it is simply a function from time to space denoting the expected location of a particular host at some time in the future. Because it is mathematically convenient to work with total functions, we augment the space domain with the symbol \perp (unknown location) and allow the motion profile to be a function from time to the augmented space:

$$\mu : T \rightarrow S^+$$

where

T denotes the time domain

S denotes the space domain

\perp signifies an unknown location

S^+ is defined as $S \cup \{\perp\}$

We consider both time and space to be discrete. The time domain is modeled as the set of natural numbers and space is defined as the Cartesian product of two integer domains. Since we assume a global notion of time, we model the current time simply as an integer variable τ . The current location of a host is given by evaluating the motion profile at time τ . In general, we assume that motion profiles associated with specific hosts are defined starting from the current time up to some point in the future and nowhere else:

$$\langle \exists t_e : \tau \leq t_e :: \langle \forall t :: t \in [\tau, t_e] \Leftrightarrow \mu(t) \neq \perp \rangle \rangle$$

We abuse language by referring to points in time when the motion profile takes the value \perp as being "undefined" from an application point of view, even though mathematically speaking the function is defined for all values in T .

As a first approximation, we define a system of coordinating hosts as a set of hosts H and their respective motion profiles. The state of the system is formally characterized by its *configuration* C , a function that maps hosts to their respective motion profiles

$$C : H \rightarrow (T \rightarrow S^+)$$

The system configuration evolves in response to two different kinds of actions. First, hosts may explicitly update their motion profiles by extending them into the future as in

$$C := C[i/\mu] \text{ where } C(i) \ll \mu$$

The condition captures the fact that the new motion profile μ for host i is defined everywhere the old motion profile $C(i)$ is defined. Formally, the relation \ll is expressed as

$$\mu_1 \ll \mu_2 \equiv \langle \forall t :: \mu_1(t) \neq \perp \implies \mu_2(t) \neq \perp \rangle$$

Second, time may advance affecting the value of the variable τ

$$\tau := \tau + 1$$

Time advance has an implicit global impact on the system configuration because all host motion profiles that are older must become undefined.

$$C := C[i/C(i)[(\tau - 1)/ \perp]] \forall i \in H$$

where $\tau - 1$ refers to the old value of τ

3.2 Disconnected Routes

Given a configuration C , one could (in principle) determine whether it is possible to transfer data from host i_1 to host i_2 . The first step involves identifying when hosts are close enough and, therefore, able to communicate. We assume that communication is possible whenever the distance between two hosts is less than the communication range of the wireless transmitters, some constant δ . In reality, transmission distances vary a great deal and rarely conform to an idealized circular pattern. Nevertheless, it is always possible to select a conservative value for δ , one that offers a high probability for successful communication between any two hosts. The time intervals during which hosts can exchange data are called *communication windows* and are formally captured in the definition of the relation w below:

$$i_1 w[t_1, t_2] i_2 = \langle \forall t : t \in [t_1, t_2] :: C(i_1, t) \neq \perp \wedge C(i_2, t) \neq \perp \wedge |C(i_1, t) - C(i_2, t)| < \delta \rangle$$

Disconnected communication is established by having data move from one host to another during such periods of communication. Since we do not expect all hosts to be end-to-end connected, the data is temporarily buffered on an intermediate host until a communication window to the next host is available. In this manner a route is established between a source and a destination. It is called disconnected because the hosts along the route may never be simultaneously connected. A host along the route may obtain data from a peer and then become completely isolated for an extended period of time before handing off the same data to another peer. A disconnected route is said to exist between two hosts i_1 to i_n from time t_1 to t_n if there exists a set of hosts (i_2 to i_{n-1}) and communication windows between them such that data can travel from the origin to the destination as described above. Formally, a disconnected route r is a temporally ordered sequence of communication windows:

$$r = (i_1, i_2, t_1, t_1'), \dots, (i_{n-1}, i_n, t_{n-1}, t_{n-1}')$$

subject to the constraint

$$t_i < t_{i'} < t_{i+1} \wedge i_i w[t_i, t_{i'}] i_{i+1} \text{ for } i = 1, \dots, n - 1$$

The corresponding disconnected path is defined as the sequence of hosts involved in defining the route. In this case, the hosts involved in the disconnected route above define the disconnected path

$$(i_1, i_2, \dots, i_{n-1}, i_n)$$

Computing disconnected routes from locally available knowledge about the system configuration is central to our model. The brute force approach to accomplishing this entails building a directed graph which includes a vertex for each host/time pair denoting the start and the end of a communication window and an edge between any two vertices whose corresponding hosts can communicate in the respective time interval, i.e., two vertices (i_1, t_1) and (i_2, t_2) are connected by an edge only if $i_1 w[t_1, t_2] i_2$. Once the graph is constructed, finding a disconnected route is simply a matter of identifying a path in the graph between the source and the destination vertices. However, the algorithms involved in accomplishing this are outside the scope of this paper. For now, all we need to consider is the fact that a disconnected route can be computed, if one exists, given the current knowledge of the motion profiles of hosts in the system. Post facto one may discover that many more routes were actually established but planning can use only what is known at the time some coordination action is initiated.

3.3 Reachability

In the most abstract sense, disconnected routes expand the definition of reachability, which has been used in the past to determine the maximal set of hosts with which a reference host can coordinate. We introduce a relation ρ to formally capture this notion of reachability that is based on successive communication windows. The definition is recursive with the base case being reachability within a communication window, which includes the default case of a host being able to “communicate with itself” across any interval of time by holding the data for future delivery:

$$i_1 \rho[t_1, t_2] i_2 \equiv i_1 w[t_1, t_2] i_2 \vee \langle \exists i, t : t_1 \leq t \leq t_2 :: i_1 \rho[t_1, t] i \wedge i \rho[t, t_2] i_2 \rangle$$

It should be immediately obvious that prior forms of reachability differ from this definition in two fundamental ways. First and foremost, earlier definitions of reachability are not parametrized with respect to a time interval; they can be seen as having an implicit time parameter defined by a single point in time. Second, in such definitions w holds true if the hosts are collocated, form a connected group, or are subject to other similar restrictions.

The other distinctive feature of CAST is the ability to coordinate across the spatial domain without explicit knowledge of the other participating hosts. A new notion of reachability needs to be introduced in order to accommodate this capability, one that captures the idea that a specific point or region is reachable within the constraints of a particular time interval. Clearly, the presence of some host operating in that space is implicit and the extensions we are introducing below make this fact evident in their respective formalizations. The simplest extension considers a point $p \in S$ to be reachable from some reference host in a specific time interval whenever a reachable host exists at that point in space. The new relation is called σ and it is defined as follows:

$$i_1 \sigma[t_1, t_2] p \equiv \langle \exists i : C(i, t_2) = p :: i_1 \rho[t_1, t_2] i \rangle$$

Similarly, a region $r \subseteq S$ is considered reachable if it contains a point that is reachable:

$$i_1 \sigma[t_1, t_2] r \equiv \langle \exists p : p \in r :: i_1 \sigma[t_1, t_2] p \rangle$$

Still other notions of reachability can be introduced to capture more subtle aspects of the semantics of coordination across space. We conclude this section with one such example. In some situations we may need to assert that data can be disseminated to any host entering a specific region throughout a particular interval in time. A region is said to be *continuously covered* with respect to some host i holding the critical data at the start if all the hosts in the required area are reachable from it during the specified time interval

$$i_1 \nu[t_1, t_2] r \equiv C(i_1, t_1) \in r \wedge \langle \forall i, t : t \in [t_1, t_2] \wedge C(i, t) \in r :: i_1 \rho[t_1, t_2] i \rangle$$

The presentation thus far has focused on considering reachability given global knowledge. Such a view, while helpful for illustration purposes, does not mirror the reality of a MANET where hosts have egocentric perspectives. Hosts in the MANET have access only to a subset of the motion profiles that make up the global knowledge. The difference between the local and global knowledge determines how effective a host is at coordinating with other hosts in the MANET over disconnected routes. Since disconnected routes are calculated from motion profiles, a dearth of motion profiles on a given host results in it having access to fewer disconnected routes, which translates into fewer opportunities for disconnected coordination. In such a situation, a host is stymied with respect to operations that it wants to carry out in the future and at locations other than its own location. This is why the knowledge base on each individual host becomes critical to its functionality. The closer the host's local knowledge is to the global knowledge, the more effective the host is, the caveat being that in some situations, even global knowledge may not be sufficient. For instance, if all global knowledge only looks 1 second into the future, there are no opportunities for disconnected coordination beyond that time. Motion profiles that extend reasonably far into the future allow for timely dissemination and better planning.

3.4 The Egocentric Perspective

In this section we make the transition from unattainable global knowledge to maintainable local knowledge bases. Formally, the kind of information being held locally is of the same type as the global configuration described earlier, but what is known locally is only an approximation of the instantaneous global configuration of the system. The fact that the formalization is identical is helpful in system analysis. Properties of the system can be stated in terms of global configurations and proven using local knowledge. During this transition we make one fundamental change in the characterization of the individual hosts. Recall that so far a host i has been characterized by its unique identifier and a motion profile μ . The change consists of replacing the local motion profile with

a more general knowledge base $M(i)$ of the same type as the system configuration C but which is always a subset thereof.

As expected, the motion profile of i in the global view C is identical to the motion profile in its own knowledge base $M(i)$. In other words, a host i always knows its own motion profile fully, which is proper as i establishes its own motion profile:

$$M(i, i) = C(i)$$

The situation changes when we consider the knowledge that host i has regarding the motion profile of some other host j :

$$\langle \forall j :: M(i, j) \ll C(j) \rangle$$

We use a gossiping protocol to exchange motion knowledge among hosts. Whenever two hosts encounter each other, i.e., they are directly connected, they exchange the contents of their knowledge bases (their own motion profile and the motion profiles collected through previous encounters). Hence, it is always the case that the motion profile of a reference host as known at another host is most often less defined than on the reference host; updates could have occurred on the reference host after the motion profile was given away.

The system now evolves in three different ways: (1) implicitly through the passage of time, (2) explicitly due to a change in a host's own motion profile where we define change to be an extension to the existing motion profile rather than a complete replacement, and (3) by acquisition of knowledge about other hosts as shown below:

$$\begin{aligned} M(i) &:= M(i) \cup M(j) \text{ where} \\ \langle \forall l, MM : MM = M(i) \cup M(j) :: M(i, l) \ll MM(i, l) \wedge M(j, l) \ll \\ &MM(j, l) \wedge (M(i, l) = MM(i, l) \vee M(j, l) = MM(j, l)) \rangle \end{aligned}$$

When two hosts exchange knowledge bases, the motion profile for a particular host as known by host i is compared against the motion profile for the *same* host as known by host j . The profile that extends farthest into the future is adopted as the “new” motion profile. In this way, two hosts synchronize their knowledge bases thus improving the quality of the information each holds. Even though this may seem to lead to unbounded growth in terms of storage requirements, the progression of time eliminates data that is older than a predefined limit.

4 Operations

Given the central role motion profiles play in planning interactions among mobile hosts in our model, it is natural to employ a similar formalization for the operations which execute across disconnected routes. The approach is attractive because it provides the opportunity to employ a unified knowledge-based treatment to describe all aspects of the coordination model. Take for instance an *out* operation. A reference host issues the operation, specifying a target for the operation. The target can be a remote host as in traditional distributed systems, or simply a set of spatiotemporal constraints that define a place and time where the operation must execute. Allowing hosts to perform remote operations can be

complicated since it requires synchronized access to the data state on the target host. Thus, any operation that has a remote target is converted to an operation request which is routed to the target host which queues requests and sequentially performs the operations locally on behalf of the originator of the operation.

At this point we must consider how the operation is routed to its target. In CAST, the system formulates a plan for moving the operation request to the target. A plan is simply a motion profile that the operation request must follow to reach its target. This motion profile can be easily derived from the reachability information that is contained in the knowledge base of the originating host as described in Section 3. However, using the same type of motion profile for operations as is used for hosts is not possible. This is because the motion profile of a host can be arbitrary whereas the motion profile of an operation must always map to a location where a host is present. To remedy this, we use an *allocation profile* which is similar to a motion profile but returns the host on which the operation is to be located at a particular time rather than a physical location. This motivates the need for a separate knowledge base that contains all operations which are transferred between hosts at the same time they are gossiping to exchange host motion profiles.

The final issue we must consider is the actual insertion of the tuple in the target tuple space. At a basic level, we can build this action into the system itself. However, we can gain much more expressive power by allowing actions to be customized according to the task at hand. Consider an **out** operation to a physical space. The semantics of this operation call for each host in the physical area to receive a copy of the tuple associated with the **out** operation. However, the allocation profile yields a path to only one host among those present in the target area. The built-in action would simply place the tuple in the tuple space of that one host. To overcome this, we introduced the concept of an *operation function*, which operates on the data state of all target hosts. This concept is especially important as it allows a great deal of flexibility when specifying the effects of coordination operations. In the case of the **out** operation, operation function places the tuple in the tuple space of the host that is reachable via the allocation profile. After this, the function uses the knowledge base on the host to compute which other hosts are in the target area and sends them copies of the tuple using **out** operations to those specific hosts. Thus, the operation function encapsulates the basic operation and the maintenance of the operation in its target scope.

An **in** operation is a three phase operation that requires three paths between the originator and the target. During the first phase, the **in** operation can be routed to its target destination in the same manner as the **out** operation. The only difference between the two is the operation function which determines what action must be taken at the target. Here again, we emphasize the importance of the operation function which helps separate the concerns related to routing of the operation to its target from the actual effect of the operation. This allows CAST to treat all operations uniformly during the routing phase, with the actual effect of the operation being hidden until the target is reached, which is the only time that it is relevant. The **in** operation is a more complex operation

than the *out* because an *in* request to multiple hosts could result in multiple tuples being removed which is inconsistent with the semantics of the operation. Thus, once an *in* request reaches its target, it searches for tuples that match the required *template*, which specifies the data being searched for. Every tuple that matches the template, is removed from the main data tuple space to a temporary tuple space by the operation function of *in*. The function then formulates, for each tuple, an *out* operation targeted towards the originator of the *in* function containing copies of the matching tuples. These *out* operations are routed to the originator in the standard way. Upon arrival, the operation function in the *out* operation places the copies in temporary storage on the originator since placing them in the main tuple space would indicate that this data is available for use. The system then chooses one of the copies returned non-deterministically. This completes the second phase of the operation. During the last phase, the host that owns the original of the copy selected is sent yet another operation request which destroys the original tuple in the temporary storage of that host. All others are sent a different operation that restores the original to the main data space. The copies of the tuples that were not chosen during the second phase are destroyed. Thus at the end of the *in* operation, only one tuple is removed from the system. The system registers a reaction on behalf of the calling application on the temporary storage space of the originator to return the result of the *in* to the caller.

We have seen how a knowledge base consisting solely of motion profiles is not sufficient since it does not account for the operational and data aspects of the coordination model. Thus, we split $K(i)$ as follows:

- M(i) - the set of motion profiles known to the local host (Section 3)
- O(i) - the set of operation requests that are on the local host
- D(i) - the data state of the local host
- T(i) - temporary storage space (not accessible to applications)

M(i) holds tuples that contain motion profiles, O(i) holds tuples that contain operation requests while D(i) and T(i) can hold any type of data tuple. The separation of these knowledge bases is key since M(i) is modified using a gossiping protocol, D(i) and T(i) are modified only locally by operation requests, and O(i) is modified by either 1) operation requests moving from one host to another, 2) operation requests being serviced, or 3) operation requests expiring. We have already described the nature of the contents of M(i), while D(i) and T(i) contain generic data tuples. O(i) contains operation requests that are generated by hosts to have an operation execute on their behalf. To summarize the presentation above, each generic operation request is formulated as a 6-tuple as follows:

- a unique identifier for the request
- the allocation profile of the operation
- the time at which the operation becomes active
- the time at which the operation becomes inactive
- the operation function
- the originator of the operation request

The unique identifier is required to distinguish requests for similar operations by different hosts, and more importantly, to distinguish the results so that a host does not mistakenly collect the results of a similar operation issued by another host. The allocation profile describes the hosts on which the operation is resident at a given time. The time of activation and deactivation indicate the time interval for which the operation is valid and able to be executed. There are two points of note: (1) in most cases, there is an implicit dependency between activation time and the allocation profile since the activation time cannot precede the time at which the operation reaches the target host as given by the allocation profile and (2) a bounded deactivation time in effect makes every operation in CAST a polling operation because the system waits for the result of an operation only for the duration that it is active. If the current time exceeds the time at which the operation becomes inactive, the system unblocks and lets the execution continue. The operation function is a constant function that encodes the effect of the operation on the knowledge state of the target host and may also manage adequate coverage of the operation in a physical space. Examples of operation functions for `out` and `in` have already been described.

Before concluding, we return to SynchroTrack, presented in Section 2. SynchroTask uses various facilities provided by CAST to deliver the intended functionality. Consider the time when Bob placed the message in the ready room. CAST's spatial `out` operation was used to deliver the tuple to the ready room. The CAST system's knowledge base on Bob's PDA was used to determine that Chris will be in the ready room from 12 PM to 1 PM and David from 1 PM to 2 PM. This information resulted in the message being moved from Bob to Chris at 12 PM and from Chris to David at 1 PM, allowing a message to be associated with a physical area rather than be associated with any particular host. SynchroTask on Eric's PDA used a spatially targeted `in` operation to retrieve the information when he came in. The entire structure allowed the person that fulfilled the role of the afternoon supervisor to retrieve the information. If an operation was targeted specifically to Alice (the expected afternoon supervisor), then Eric would have not gotten the message.

5 Related Work

Since the work presented in this paper is a new approach to coordination built on top of a novel MANET routing protocol, we address related work in two areas—coordination models and MANET routing protocols. The first example of a coordination model was Linda [6]. In Linda, coordination is characterized by a centralized coordination mechanism while the application that uses it may be distributed. In modern implementations of the coordination concept, such as JavaSpaces [7] and TSpaces [8], various parts of the application coordinate with each other by means of a tuple space maintained at a central location.

Coordination models have also found favor in agent-based systems. TuCSoN [9] introduced multiple tuple spaces called tuple centers while in MARS [10], mobile agents are provided upon arrival on a particular host

with a handle to the local tuple space, which is shared among all agents present on the same host. Ara [11] introduced a constrained rendezvous type of coordination: some agents assume the role of coordination servers and represent meeting points where agents can ask for services

More recently, coordination models have been adapted to novel computational environments [12], [13], [14], and [15], which highlights their versatility. One such environment where coordination models were introduced in support of new classes of applications was MANETs. LIME [3] proposed the idea of multiple (local) tuple spaces that were transiently shared to form a federated shared dataspace when hosts are in communication range. Limone [4] is a lightweight alternative to LIME implemented to offer fewer guarantees. TOTA [16] uses spatially distributed tuples, injected in the networks and propagated according to applications specific patterns.

Coordination models adapted for MANETs often support only peer-to-peer connections. To support multihop connections, they need to be combined with MANET routing protocols which fall into four broad categories: (1) proactive protocols such as DSDV[17], WRP[18], CSGR[19], which constantly maintain and update routes using routing tables at the cost of high bandwidth usage; (2) reactive protocols such as AODV[20], TORA[21], ABR[22], DSR[23], which only search for routes when they are required at the cost of low responsiveness; and (3) disconnected routing such as Epidemic[24], Message Relay[25], which allow messages to be sent via a gossiping protocol.

Most of the protocols mentioned above use broadcasts for route discovery and maintenance. Recent developments have targeted the use of location information to reduce the overhead required to discover and maintain routes. This has resulted in a new family of routing protocols called geographic routing protocols [26], [27]. The basic idea is to greedily forward the message to the next hop neighbor physically the closest to the destination. The greedy approach fails often in local optima that become dead-ends before the target is reached. This problem has multiple solutions in the form of the GPSR protocol [28], terminode routing [29], among others. Our work is different from geographic routing protocols in that (1) we do not use location information to optimize routing tasks, (2) we do not enforce an end-to-end route, and (3) common problems with geographic routing such as topology holes and local minima do not affect our approach. Location information is important to us for the purpose of determining when hosts are going to be connected with each other and are an integral part of the model (along with time).

6 Conclusions

In this paper, we have presented Coordination Across Space and Time (CAST), a new coordination model tailored for MANETs that enables coordination across the reaches of space and time. CAST achieves this functionality by its use of 1) disconnected routing, which allows two hosts that are not in direct contact to coordinate with each other, 2) spatiotemporal operations that enable operations to execute at specific

locations in space and at any point in time, and 3) a knowledge driven architecture that unifies the treatment of motion of hosts, operations, and data state. The next steps in our investigation are a formal examination of the model's expressive power and an engineering effort to deliver the model's functionality in the form of an operational middleware.

Acknowledgments. This research was supported in part by ONR-MURI research contract N00014-02-1-0715. Any opinions, findings, and conclusions expressed in this paper are those of the authors and do not necessarily represent the views of the sponsors.

References

1. Cabri, G., Leonardi, L., Zambonelli, F.: MARS: A programmable coordination architecture for mobile agents. *IEEE Internet Computing* **4** (2000) 26–35
2. de Nicola, R., Ferrari, G.L., Pugliese, R.: klaim: a kernel language for agents interaction and mobility. *IEEE Transactions on Software Engineering (Special Issue on Mobility and Network Aware Computing)* (1998)
3. Murphy, A., Picco, G., Roman, G.C.: LIME: A middleware for physical and logical mobility. In: *Proc. of the 21st Int'l Conf. on Distributed Computing Systems*. (2001) 524–533
4. Fok, C.L., Roman, G.C., Hackmann, G.: A lightweight coordination middleware for mobile computing. In: *Proceedings of COORDINATION 2004*. Volume 2949 of LNCS., Springer Verlag (2004) 135–151
5. Peine, H., Stolpmann, T.: The architecture of the Ara platform for mobile agents. In Popescu-Zeletin, R., Rothermel, K., eds.: *First International Workshop on Mobile Agents MA'97*. Volume 1219 of *Lecture Notes in Computer Science.*, Berlin, Germany, Springer Verlag (1997) 50
6. Gerlenter, D.: Generative communication in Linda. *ACM Computing Surveys* **7** (1985) 80–112
7. Microsystems, S.: *Javaspaces specification*. (<http://java.sun.com/products/jini/specs>)
8. Wyckoff, P.: Tspaces. *IBM System Journal* **37** (1998) 454–474
9. Omicini, A., Zambonelli, F.: The TuCSon coordination model for mobile information agents. In: *Proceedings of the 1st Workshop on Innovative Internet Information Systems*, Pisa, Italy (1998)
10. Cabri, G., Leonardi, L., Zambonelli, F.: MARS: A programmable coordination architecture for mobile agents. *IEEE Internet Computing* **4** (2000) 26–35
11. Peine, H., Stolpmann, T.: The architecture of the Ara platform for mobile agents. In Popescu-Zeletin, R., Rothermel, K., eds.: *First International Workshop on Mobile Agents MA'97*. Volume 1219 of *Lecture Notes in Computer Science.*, Berlin, Germany, Springer Verlag (1997) 50–61
12. Papadopoulos, G.A., Arbab, F.: Coordination models and languages. In: 761. *Centrum voor Wiskunde en Informatica (CWI)* (1998) 55

13. Papadopoulos, G.: Models and technologies for the coordination of internet agents: A survey (2000)
14. Cabri, G., Leonardi, L., Zambonelli, F.: The impact of the coordination model in the design of mobile agent applications. In: Proceedings of the 22nd International Computer Software and Application Conference. (1998) 436–442
15. Fok, C.L., Roman, G.C., Lu, C.: Software support for application development in wireless sensor network. In: Mobile Middleware. CRC Press (2005)
16. Mamei, M., Zambonelli, F., Leonardi, L.: Tuples on the air: a middleware for context-aware computing in dynamic networks. In: Proceedings of the 2nd International Workshop on Mobile Computing Middleware at the 23rd International Conference on Distributed Computing Systems (ICDCS). (2003) 342–347
17. Perkins, C., Bhagwat, P.: Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In: ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications. (1994)
18. Murthy, S., Garcia-Luna-Aceves, J.J.: An efficient routing protocol for wireless networks. *Mobile Networks and Applications* **1** (1996) 183–197
19. Chiang, C., Wu, H., Liu, W., Gerla, M.: Routing in clustered multihop, mobile wireless networks. In: IEEE Singapore International Conference on Networks. (1997) 197–211
20. Perkins, C.: Ad-hoc on-demand distance vector routing. In: MILCOM '97 panel on Ad Hoc Networks. (1997)
21. Park, V.D., Corson, M.S.: A highly adaptive distributed routing algorithm for mobile wireless networks. In: Proceedings of INFOCOM'97. (1997) 1405–1413
22. Toh, C.K.: A novel distributed routing protocol to support ad-hoc mobile computing. In: Fifteenth Annual International Phoenix Conference on Computers and Communications. (1996) 480–486
23. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. *Mobile Computing* **353** (1996)
24. Vahdat, A., Becker, D.: Epidemic routing for partially connected ad hoc networks. Technical Report CS-200006, Duke University (2000)
25. Li, Q., Rus, D.: Communication in disconnected ad hoc networks using message relay. *Parallel and Distributed Computing* **63** (2003) 75–86
26. Imielinski, T., Navas, J.: Rfc 2009 - gps-based addressing and routing. <http://rfc2009.x42.com/> (1996)
27. Navas, J.C., Imielinski, T.: GeoCast – geographic addressing and routing. In: *Mobile Computing and Networking*. (1997) 66–76
28. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: *Mobile Computing and Networking*. (2000) 243–254
29. Blazevic, L., Boudec, J.Y.L., Giordano, S.: A location-based routing method for mobile ad hoc networks. *IEEE Transactions on Mobile Computing* **4** (2005) 97–110