DARPA SOFTWARE FOR DISTRIBUTED ROBOTICS: TECH REPORT 2002-12-01

CENTIBOTS

Large Scale Robot Teams

Kurt Konolige, Charles Ortiz, Regis Vincent, Andrew Agno, Michael Eriksen, Benson Limketkai, Mark Lewis, Linda Briesemeister, Enrique Ruspini
Artificial Intelligence Center
SRI International
Menlo Park, CA

Dieter Fox, Jonathan Ko, Benjamin Stewart
Department of Computer Science and Engineering
University of Washington
Seattle, WA

Leonidas Guibas
Computer Science Department
Stanford University
Stanford, CA

Abstract: As part of the DARPA Software for Distributed Robotics Program, SRI International, Stanford University, the University of Washington, and ActivMedia Robotics are designing and implementing a computational framework for the coordination of large robot teams, consisting of at least 100 small, resource limited mobile robots (CentiBOTS), on an indoor reconnaissance task.

Key words: Multirobot mapping and surveillance

1. PROBLEM DESCRIPTION: LARGE-SCALE ROBOT TEAMS

The requirements of autonomy, size, low power, and limited computation available on small robots place extraordinary constraints on communication and coordination of robot platoons. Individual robots, executing a limited set of behavioral programs, must cooperate to produce a global behavior that satisfies the goals of the mission. Current robot architectures rely on large, power-hungry subsystems for mobility, communication, and control; furthermore, they do not address the problem of coordinating large numbers of robots.

In the CentiBOTS project, we envisage a system of a large number of mobile robots able to effectively explore, map, and surveil the interior of a building. The name of the project comes from our goal of having 100 robots performing the task. Each robot is minimally capable of self-localization in a map and communication with nearby robots. Some are more specialized for tasks such as mapping or tracking people, with different degrees of performance (and consequent computational and sensor abilities). The robots must function as an effective *team* for accomplishing an extended surveillance mission, with appropriate resilience in the face of environmental challenges. These include a dynamic environment with unexpected events, uncertainty that is inherent in sensing and acting on the physical world, and the need for communication among the robots and/or with a central base station under conditions in which power or bandwidth may be limited, connectivity may be intermittent, and stealth may be a factor.

Under these circumstances, it is imperative to be selective about what is sensed, and what needs to be communicated. Each sensor must be tasked to obtain only the data relevant to the overall mission, each robot must aggregate and compress the data from its sensors before transmission to others, and these communications need to be guided by the overall mission control architecture.

The nature of the control architecture itself is problematic. The challenge is to control a large number of robots effectively, based on mission parameters. Individual robots may fail, or may not have the required sensors for a task, or may need cooperation from other robots to do their job. Given the environmental requirements above, it is impossible to centralize the planning, resource allocation, and coordination task; instead, it must be *distributed*, with local robot groups sharing information and coordinating with other.

1.1 Research Goals

The previous section identified a daunting array of challenges for this project, and relate directly to our research goals. We intend to answer the following questions:

1. What is a realistic distributed control architecture for hundreds of limited-capability robots cooperating on a task? Such an architecture must account for limited communication, limited computation, and disruption of a dynamic environment. Our goal here is to develop an architecture that scales well because its organization is *multi-layered*, functioning at the individual, small group (squad), and team levels.
2. How can we effectively reason about spatial arrangements in a distributed, dynamic environment? A major portion of the planning effort involves reasoning about space for mapping, localization, and tracking; we intend to extend current single-robot methods to robot squads and teams of robot squads.
3. Are there good methods for squads of robots to coordinate in solving mapping and tracking tasks? We have already developed effective probability-based algorithms for these tasks using single robots; our idea is that sensor and action fusion using extensions of these techniques will result in more effective performance.
4. How can information be passed among robots whose communication topology constantly changes? We intend to deploy mobile ad-hoc network on larger scale than tested today to support communication needs of autonomous robot teams
5. How can a user track and influence the development of a mission scenario that involves hundreds of robots?

We believe that individual robot behaviors have matured enough to shift our focus to collaboration and coordination among robots. The benefits are increased robustness and resistance to environmental disturbance; reduction in computational and sensor load on individual robots; and enabling of highly effective, controllable robot teams.

1.2 Overview of the Approach

In the project scenario, the CentiBOTS are deployed as an advanced surveillance team for urban missions. A first set of mapping-capable CentiBOTS will survey an area of interest: build and share a distributed map as well as highlight hazards, humans, and hiding places. They will then combine with a second wave of tracking robots that deploy in an optimal way into the area, configuring themselves to effectively sense intruders and

share the information among themselves and a command center. As a large team, the CentiBOTS will be able to reconnoiter a set of buildings faster, more reliably, and more comprehensively than an individual or small set of robots. For example, the team can dynamically form subteams to perform tasks that cannot be done by individual robots. Examples of such tasks might be to measure the range to a distant object, or to use other robots as markers in the building for localization or communication relays. In addition, the team can automatically reconfigure itself to handle contingencies such as disabled CentiBOTS or changing lighting conditions.

The robot teams will collaboratively perform tasks with minimal supervision in dynamic environments. Our major contribution is a distributed robot architecture in which collective behavior is uniquely *adaptive*, *fault tolerant*, and *capable*, incorporating the following innovative elements.

?? **A collaborative, multi-level architecture, adaptive to new tasks and team organizations, and scalable to very large teams** based on SRI's proven Saphira robot control system and the Distributed Dispatcher Manager (DDM) hierarchical agent framework developed by SRI for the DARPA ANTs program. We will also incorporate principles of *collaboration*, derived from our work on structured and dynamic negotiation for the DARPA ANTs program, so that CentiBOTS will be capable of re-organization and re-tasking in response to resource and problem changes in the environment.

?? **Optimal distributed map-building and deployment of CentiBOTS for tracking** based on novel distributed spatial reasoning techniques. We extend single-robot probabilistic methods such as Markov sampling and relational dynamic Bayes nets to the multi-robot, distributed case.

?? **Large-scale, fault-tolerant communication** building on SRI-developed mobile ad-hoc network protocols that have already been successfully demonstrated on smaller robot teams. The protocol supports the mission specific communication tasks efficiently. Additionally, it alerts the application about decreasing fault tolerance when links break.

?? **Robot team interface and monitoring** that provides both robot level attribute-of-interest updating and tracking as well as task and team level goal tracking.

?? **Analyzable and predictable behavior** Through systematic experiments with well-defined evaluation metrics, in both the SRI Augmented Reality Robot Simulator as well as demonstrations and experiments, we will show increasing

capability of the software solution on a collection of at least 100 COTS mobile robot platforms.

The project started in July 2002. By the time of the workshop, we will have completed our first demo at 6 months, and have results to report on the complete integration of distributed mapping, ad-hoc network communication, and team formation and execution. As of the current writing (Dec 1, 2002), we have major portions of the system in place. Recent results are posted to the website www.ai.sri.com/centibots.

2. ROBOTS AND INFRASTRUCTURE

Given the nature of the mission and the large number of robots, a critical component of the project is the robot platform, sensor suite, and software programming base. The robots must have enough capability to be able to perform mapping, localization, communication and tracking functions, while at the same time they are subject to the conflicting demands of low power, simplicity, modest computational load, and small size. In contrast to swarm-based robotics, where individual robots have almost no ability for independent action, we equip each robot with the capability to localize and perform some tracking or mapping function. Pioneer and AmigoBot robots from ActivMedia, and robot software from SRI and ActivMedia, form the core of the infrastructre. The basic robot types are as follows.

Robot class	**#**	**Computer**	**Sensors**	**Capabilities**
Pioneer II DX/AT	6	PIII EBX	LRF, sonars	map, detect, track
AmigoBot	10	VIA Epia	Stereo vision, sonars	detect, track (form, range)
	60	VIA Epia	Mono vision, sonars	detect, track (color, bearing)
	14	VIA Epia	Doppler radar	track (motion, no light)

All robots are equipped with 802.llb wireless links, and are capable of localizing in a map, through the use of sonars or LRF sensors. The larger (30 lb), more capable Pioneer II DX/AT robots have a powerful computer, and a laser range finder for mapping and people-tracking. These robots perform the distributed mapping task, and assist in detection and tracking of people.

The smaller (6 lb) AmigoBots are used as detection and tracking robots for finding the targets, and for surveillance of intruders. They have low-power VIA Epia processors, and are able to survive for long periods of time in low-power mode (6-12 hours), while still performing communication and surveillance on wakeup. The mix of sensors on the AmigoBots reflects differing environmental conditions, capabilities, and power requirements.

The base software for the robots, Saphira/Aria, is a joint project of SRI and ActivMedia. Saphira/ARIA is a modern 3-level (behaviors, sequencing, strategy) robot control architecture with an extensive library of modular capabilities, including probabilistic localization, map-building, optimal realtime path planning, and visual tracking, developed by members of the project team. We have extended the core architecture to be network-aware, so that each robot becomes a member of an ad-hoc mobile network (Figure 1).

3. COMMUNICATION ARCHITECTURE

CentiBOT teams must operate with little or no infrastructure and present a challenging scenario for information operations. Networks are formed in an **ad-hoc** fashion, and information exchanges occur via the wireless networking equipment carried by the individual CentiBOTS. While the CentiBOTS are executing their mapping and search mission, fluctuations in the network topology occur when an individual moves or when wireless transmissions are blocked by building features, distance, or interference from

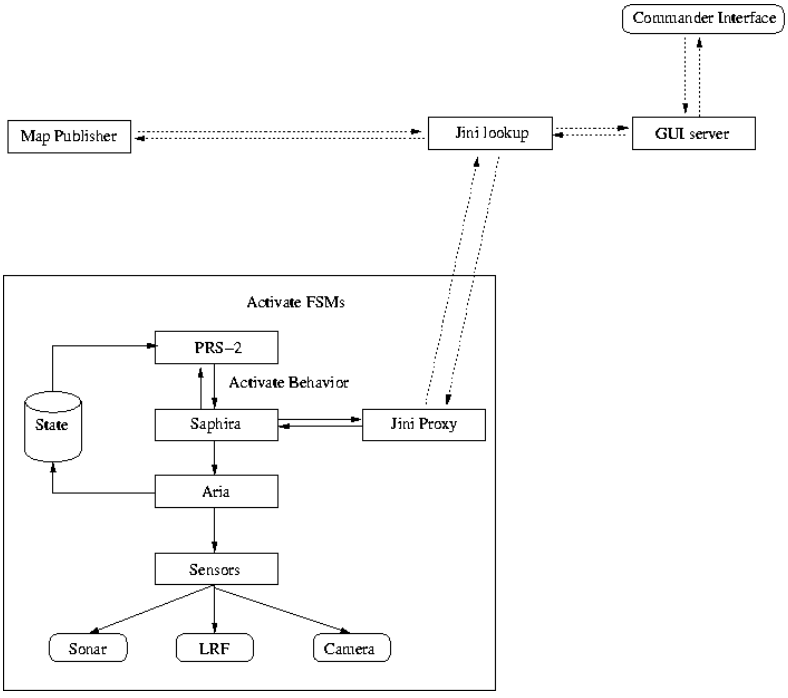

Figure 1. Robot control architecture, showing an individual robot, and its connections to the robot net.

other RF transmissions.

In spite of such dynamically changing conditions, the team's CentiBOTS must maintain close communication with one another. We therefore anticipate a requirement for self-configuring, self-sustaining dynamic networks coupled with a location-independent flexible addressing architecture for effective network communication.

The CentiBOT teams are highly collaborative in nature with a requirement for time-critical communication. However, the transmission range of each node is limited in order to preserve its battery power. Hence, the CentiBOT team is organized into a **Mobile Ad-hoc Network (MANET)**, wherein messages are exchanged directly between members of the team or may be forwarded via other members to extend the range. Since communication bandwidth is a scarce resource in a MANET, it is important that the routing protocol be efficient in terms of overhead.

3.1 The TBRPF Routing Protocol

SRI has developed a protocol called **Topology Broadcast based on Reverse-Path Forwarding (TBRPF)** [Bellur and Ogier 1999, Ogier et. al. 2002] to manage the network multihop routing while the topology is changing. TBRPF is an efficient proactive, link-state routing protocol designed for mobile ad-hoc networks, which provides hop-by-hop routing along shortest paths to each destination. Each node running TBRPF computes a source tree (providing paths to all reachable nodes) based on partial topology information stored in its topology table, using a modification of Dijkstra's algorithm. To minimize overhead, each node reports only 'part' of its source tree to neighbors. This is in contrast to other protocols in which each node reports its 'entire' source tree to neighbors.

TBRPF uses a combination of periodic and differential updates to keep all neighbors informed of the reportable part of its source tree. Each node also has the option to report additional topology information (up to the full topology), to provide improved robustness in highly mobile networks.

TBRPF consists of two modules: the TBRPF neighbor discovery (TND) module and the routing module that performs topology discovery and route computation. The neighbor discovery protocol allows each node in the network to quickly detect the neighboring nodes with which the node has a bi-directional link.

TBRPF performs neighbor discovery using "differential" HELLO messages that report only changes in the status of neighbors. This results in HELLO messages that are much smaller than those of other link-state routing protocols such as OSPF. As a result, HELLO messages can be sent

more frequently in highly mobile networks without increasing overhead significantly

TBRPF is extremely agile in that a change in the up or down status of links is quickly detected, and alternate routes are immediately computed. The proof of correctness and pseudo-code for TBRPF as well as examples illustrating its operation can be found in [Bellur and Ogier 1999, Ogier et. al. 2002].

3.2 Testbed Elements

SRI's TBRPF protocol is implemented in **Linux** with the Merit Multi-Threaded Routing Toolkit (MRT) daemon (www.mrtd.net). This implementation has been in use for laboratory and fielded experiments since June 1999.

TBRPF operates transparently on each node of the ad hoc network, providing a standard IP stack interface to network-based applications. The robots and command center send and receive messages as if being connected through a common local area network. We display current topology information at individual nodes and an aggregated picture at the command center (Figure 2). In future implementations of CentiBOTS, we plan to support multicast messages suitable for more efficient intra-team communication.

The network interfaces are commercially available IEEE 802.11b PCcards and USB network interfaces operating in the 2.4 GHz frequency band, using the Direct Sequence Spread Spectrum (DSSS) modulation, and provides up to 11 Mbps data transfer rates with a maximum range of approximately 1000 m line of sight. We configure the cards to use **Ad-hoc** mode, rather than to be dependent on fixed infrastructure elements.

Aiming at deployment of as many as 100 nodes, we will operate one of the largest mobile ad hoc networks known today. Yarvis et al. [Yarvis et al. 2002] have reported an ad hoc sensor network for interactive voting

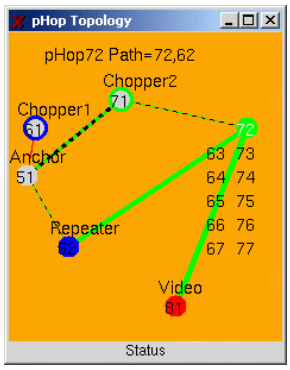

Figure 2. Dynamic topology display screen shows current nodes and links.

applications with configurations of 24, 48, and 91 nodes. However, they placed and fixed the nodes over a rectangular grid, and their experiment lasted for one hour. In our CentiBOTS project, all robots are mobile and their mission tasks take several hours.

3.3 Distributed Directory Service

For higher level reasoning, the robots need to obtain the current status of other robots on the mission. We are developing a distributed directory service to provide robots with such information. Given the scarce bandwidth and the inherently unpredictable network structure, the distributed directory service must carefully adjust the amount of messages that keep status information up to date at the expense of getting possibly stale data.

The distributed directory service provides a query interface through which a robot can perform various searches for other robots and computers, their locations and their assets. Each robot maintains a local table with all information it gathers about the current state of the network. In adjustable time intervals, the robot updates its local copy of the table with entries kept in a network-wide table, and also submits its own current information to the network table.

We chose the JavaSpaces™ technology to implement the distributed directory service. JavaSpaces is a Jini™ service that provides a high-level tool for creating collaborative and distributed applications in Java. The JavaSpaces model is different from techniques like message passing and remote method invocation. A space is a shared, network-accessible repository for objects. Processes use the repository as a persistent object storage and exchange mechanism; instead of communicating directly, they coordinate by exchanging objects through spaces.

Processes perform simple operations to write new objects into a space, take objects from a space, or read (make a copy of) objects in a space. When taking or reading objects, processes use a simple value-matching lookup to find the objects that matter to them. If a matching object is not found immediately, then a process can wait until one arrives. Unlike conventional object stores, processes do not modify objects in the space or invoke their methods directly --- while there, objects are just passive data. To modify an object, a process must explicitly remove it, update it, and reinsert it into the space.

For the distributed directory service, the persistent objects in the space are the entries of status information for each robot. Depending on the current network situation and also on pre-determined parameters, the robots update their own entries and their local copies of the other entries more or less frequently. When the higher level behavior of a robot queries the distributed

directory service, it compiles the answer from the information stored locally and in the network space.

4. MULTIROBOT MAPPING AND LOCALIZATION

Coordinated exploration of an unknown environment is one of the most fundamental problems in multi-robot coordination. We propose a novel, distributed approach that addresses this problem in its most general way. Key features of our approach are the consideration of limited communication between robots, no assumptions about relative start locations of the robots, and dynamic assignment of processing tasks. We apply efficient, statistical methods to determine hypotheses for the relative locations of robots. To achieve maximal robustness, these hypotheses are verified before maps are merged. Once robots know their relative locations, they form *exploration clusters* so as to coordinate their actions. Furthermore, our approach dynamically assigns processing tasks and roles to robots, thereby avoiding the dependency on a central server.

4.1 Communication and Coordination Architecture

Our distributed approach to mapping and exploration is enabled by pair-wise relations between robots. Each pair of robots can have four different types of interactions: none, hypothesis generation, hypothesis verification, and coordination. At each point in time, the state of the multi-robot system can be summarized by a graph structure where the nodes are individual robots and edges represent the current interaction between two robots (see Figure 3). We will now briefly discuss the different types of interactions.

1. **No interaction**: The robots are not within communication range (no arc between nodes in Figure 3).
2. **Hypothesis generation** (dotted edges): The robots can communicate but don't know their relative locations. In this stage, one of the two robots receives sensor data from the other robot and estimates their relative location using its own map. Which of the two robots adopts the estimation role depends on available computational resources.

3. **Hypothesis verification** (dashed edges): Robots can communicate and verify a location hypothesis determined in the hypothesis generation phase. This is done by moving to a point at which the robots try to meet. If the robots don't meet at the expected location, the hypothesis is rejected and they continue with the hypothesis generation phase. Otherwise, the robots can establish their relative positions, combine their maps, and coordinate their exploration efforts.
4. **Coordinated exploration** (solid edges): Once the robots determined their relative locations, they can share their maps and perform coordinated exploration. A nice feature of this interaction type is transitivity, *i.e.* if robot *i* and *j* can share their maps, and robot *j* and *k* can share their maps, then all three robots can build a combined map. Hence, robots in this interaction mode form *exploration clusters* in which they can coordinate their actions (indicated by the gray areas in Figure 3). Each cluster determines one robot responsible for data combination and robot coordination (dark nodes). All information is frequently spread throughout the cluster, and direct communication between all robot pairs within the cluster is not required. The transitions between the different interaction modes are summarized in Figure 4.

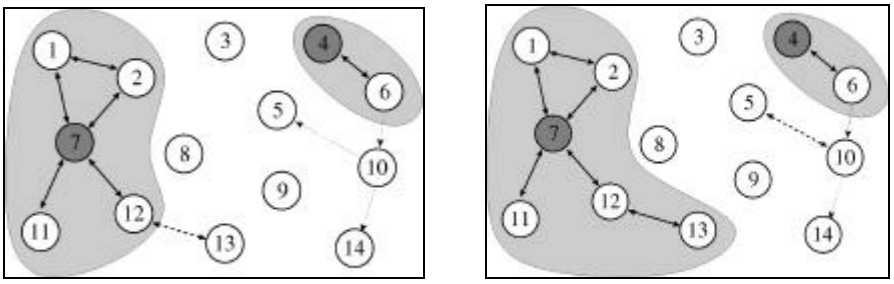

Figure 3. Dynamic communication / interaction graph at two points in time. Note that this graph illustrates different interactions between robots, *not* spatial relations. Robots are shown as circles. Solid edges indicate coordinated exploration of robots; dashed lines indicate that two robots currently navigate to a meeting point so as to verify a hypothesis for their relative positions; and dotted lines show communication between robots without valid location hypotheses. (a) Robots `1`,`2`,`7`,`11`, and `12` already established their relative locations and coordinate exploration. For this exploration cluster, robot `7` was chosen to perform data combination and exploration coordination. Robots `12` and `13` do not yet know their relative locations. They are currently moving to a meeting point so as to verify a location hypothesis. Robot `10` can communicate with robots `5`, `6`, and `14`, but no good location hypothesis has been generated so far. (b) Robots `12` and `13` moved to the meeting point and detected each other. As a result, robot `13` is integrated into the exploration cluster. Robot `5` determined a hypothesis for robot `10`'s relative location. Robot `10` accepted the meeting point and they both move to this location.

4.2 Technical Approach

To implement the individual parts of our architecture, we will rely on existing, well established techniques whenever possible. The key components are:

?? **Individual mapping and exploration**: Before a robot can coordinate its activities with another robot, it explores the environment using recently developed online mapping and exploration techniques [Yamauchi 1998, Thrun et al. 2000, Gutmann and Konolige 2000].

?? **Coordinated mapping and exploration**: Whenever multiple robots can communicate and know their relative locations, they form an exploration cluster. One robot within each cluster is chosen to coordinate the exploration of the cluster and to combine information collected by the individual robots. This central coordination robot can be replaced by any other robot in the cluster whenever necessary. The coordination strategy is based on ideas used in [Burgard et al. 2000, Simmons et al. 2000]. In a nutshell, this approach coordinates robots by sending them to different unexplored areas. We intend to significantly reduce the computational complexity of this approach by clustering frontier cells based on their proximity.

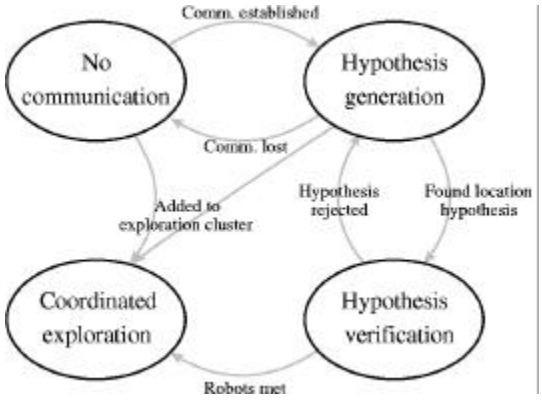

Figure 4. Transitions between the possible interactions between two robots. Depending on the initial knowledge, robots start either in "no communication" or "coordinated exploration". As soon as robots can communicate, they estimate their relative locations (hypothesis generation). Once a high probability hypothesis has been generated, the robots try to verify it by meeting each other (hypothesis verification). If they do not meet at the expected location, the hypothesis is rejected and they keep on estimating their relative locations. If, however, they meet, then they combine their maps and perform coordinated exploration. Note that the coordinated exploration mode can be reached from any other mode if a robot within the same exploration cluster establishes this mode with a new robot. For example, in Figure 3b), robots `1` and `13` are in the coordinated exploration mode, established by the connection of robots `12` and `13`.

?? **Hypothesis generation**: In this phase one robot receives sensor data from the other robot and estimates their relative location using its own map. To solve this technically very challenging problem, we are currently developing a Monte-Carlo approach that generates location hypotheses by sampling trajectories of one robot through the other robot's map. This technique, similar to particle filters for robot localization [Fox et al. 2000b], takes into account that a robot's path might overlap only partially or not at all with the other robot's map (overlap estimation is done by a dynamic Bayesian network). Once a robot received enough data to determine the other robot's location with high probability, it generates a meeting point with the other robot. This meeting point is chosen optimally under consideration of both robots' location uncertainty. The other robot uses a decision theoretic approach to decide whether it accepts the meeting point or prefers to keep on exploring on its own.

?? **Hypothesis verification**: Once two robots agree to meet so as to verify a hypothesis for their relative locations, they both move there and use their sensors (laser range-finder or vision [Fox et al. 2000a]) in order to detect each other. If they detect each other they can share maps, otherwise the hypothesis is rejected and they continue to generate new hypotheses.

We are currently implementing the individual components of our architecture. Based on very promising first experiments and the fact that we can rely on existing software solutions to most of the basic components, we expect to generate results within the next few months. A full paper will present experimental results and further details on the underlying Bayesian estimation techniques.

4.3 Current Mapping Progress

At the moment, we have successfully integrated maps from 5 separate robot mapping runs into a large scale map (Figure 5). The map data was collected offline, by running a mapping robot on 5 different trajectories, and then mixing all the scans as if 5 robots were running simultaneously.

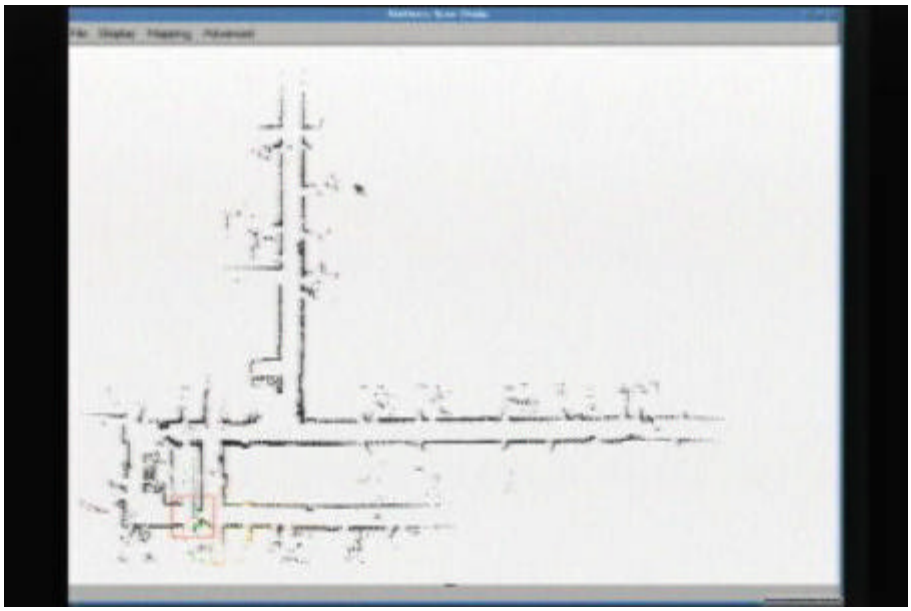

Figure 5. 5 robots mapping from a common breach. The map is about 2/3 constructed at this point, with 4 areas of frontier and 1 area of infill explorationl

5. DISTRIBUTED COORDINATION AND PLANNING

5.1 Multi-level agent architecture

Teams are comprised of autonomous vehicles (AVs). Each AV is designed to reflect two dimensions of organization: a functional dimension and a software dimension. The former segments robot functionality into what we refer to as team, strategy, tactical, and control levels. Figure 6 illustrates the architecture. Computations at each level associate a particular functionality with that level and have complexity that is conceptually bounded, reflecting the expected time available for decisions made at that level.

The lowest levels are responsible for purely reactive robot behavior, while more deliberative and goal-directed behavior takes place at the higher levels, generally over a longer period of time. At the control level, response is immediate; the robot has a 10ms cycle time for responses. At the task level, responses take place within 100ms to 1s. Computations taking place at the strategy level generally take 1 to 10s and can usually be performed in parallel with actions taken at other levels. Finally, deliberations at the team level span a conceptually longer period of time (approximately a minute), responding to faults at other levels approximately every 1 to 10s. Crucially, progress at each level is monitored so that task failure (here, task refers to the internal robot tasks undertaken at each level) and excessive backtracking can be avoided by offloading tasks to other levels (or even to a user).

The team level is responsible for decisions involving societal aspects of the robot group, such as negotiations with other robots on the division of responsibility or the allocation of resources. The team level is also designed to respond to changes in the environment that could impact the performance of the group (e.g., a robot that suddenly detects an intruder entering a team member's sector should realize that if that team member is already tracking another intruder, it will need help).

The strategy level is concerned with longer-term (in comparison to the control level) individual decisions involving a robot's intentions (i.e., its commitments to future actions). From a resource-bounded perspective, intentions serve the role of representing fairly stable commitments to actions; central to the strategy level is an ability to reconcile existing intentions with newly considered ones. When a potential intention would conflict with an existing one, the agent must either reject the potential

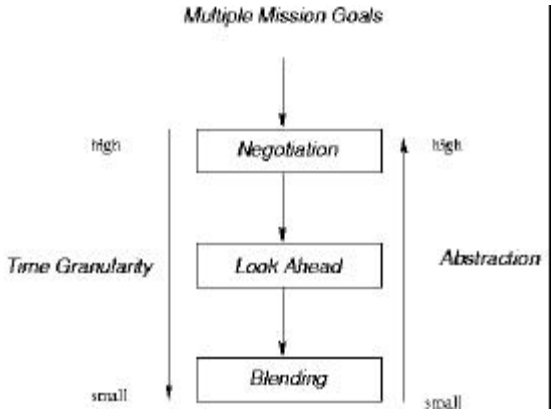

Figure 6. High level autonomy diagram.

intention or reconsider its existing intentions in the new context.

The strategy level is also responsible for exploring ways in which to achieve an intention, including the means to perform that intention and the resources needed to support execution. Typically, the decisions at this level proceed at a high level of abstraction. In considering a potential intention, strategic-level projection of that action in the context of existing intentions is necessary, but generally follows simplified considerations of relevant lower-level factors. An example is navigation around small obstacles, which is assumed to succeed at the strategic level because it is handled in a reactive manner. If the obstacle proves impossible to overcome during execution, the system adapts. By design, control is passed up to the strategy or team level for reconsideration.

At the tactical level, intended goals that are the output of the strategy level (along with the expected resources needed for execution) are processed when the time comes for execution. Goals have associated plans (each representing a possible means for achieving that goal). Each goal is matched with a plan that does not exceed the resources deemed necessary at the strategic level. In addition, monitoring sentinels are attached to the plan that can be activated during execution for tracking progress and adapting execution to unexpected changes. Adaptation at the tactical level takes the form of interleaving multiple activities, which may be intended to be performed at the same time. For example, a robot may wish to follow an intruder while at the same time remaining in communication with a team member. Rather than define a behavior, for every possible combination of behaviors, such as *follow_and_stay_in_communication*, the tactical level implements a scheme for behavior blending. Behavior blending is also used to manage goals of varying priority.

Behavior blending is based on a sound logical formulation [Ruspini 1991], based upon the notion of utility that permits the context-dependent blending of multiple behaviors. Each behavior is essentially a producer of desirability measures. These desirability measures are functions defining the relative utility of potential actions from the viewpoint of a specific goal. Behavior-specific desirabilities are then combined on the basis of considerations of the relative importance of each behavior given the current operational context. In addition to agent-specific (or in some cases, human) preferences, our behavior-based approach permits the representation and manipulation of group behaviors imposing soft constraints on the overall behavior of collaborating agents.

In this framework, *goals* are represented by means of state-functions that quantify the degree by which being in that state means that the goal has been attained. Typically, the degree of attainment of a specific goal is represented by a number between 0 and 1, with 0 corresponding to the worst possible

situation (as far as that goal is concerned) and 1 corresponding to full goal attainment. In cases where it is impossible to determine precisely the relative desirability of being in a particular state, more general qualitative measurement scales may be used (e.g., intervals of possible values). As modeled in this methodology, goals are elastic in the sense that, in most situations, there exists a continuum of possibilities of degree of goal attainment.

The control level is responsible for passing low-level actions for execution to the AV. The control level is also responsible for regularly polling the state of resources on the robot (e.g., battery, camera, motors) and communicating that information to the appropriate level.

All the software modules of the current system are implemented in SRI's *Procedural Reasoning System* (PRS), a reactive control system based on the belief, desire, and intention (BDI) agent model.

5.2 Team organization and blending

Each robot is architected in the manner just described. Teams, however, are organized hierarchically in order to manage the complexity of the team activity. Within a team, any robot is able to take on either a leadership role or a supporting role in the collaborative activities of the team. Task and resource management is distributed in several stages. Our approach is based on a system called the distributed dispatcher manager (DDM) [Yadgar et al. 2002]. DDM has been tested in environments consisting of thousands of mobile, cooperative (possibly noisy) sensor agents in which thousands of tasks, in the form of targets that must be tracked, must be managed. Each DDM agent has direct access to only local and partial information about its immediate surroundings. DDM organizes teams hierarchically, which reduces the degree of communication necessary between agents. DDM processes can very quickly combine partial results to form an accurate global picture. Each successively higher level narrows the uncertainty about the solution based on the data obtained from lower levels. We have shown that the hierarchical processing of information reduces the time needed to form an accurate global solution and have also derived analytical bounds on the amount of information collected with respect to the ideal depth of the hierarchy.

DDM is loosely based on a metaphor for task distribution modeled on the activity of a "taxi dispatcher" who assigns taxis to incoming calls (tasks). The taxi dispatcher dispatches taxis in terms of their general proximity to passenger pickups. The dispatcher need not have complete knowledge of the location and state of each taxi. Similarly, in DDM, a team leader or dispatch agent distributes a problem and its solution among the robot team. The team

is organized in a hierarchical fashion. In DDM, the dispatcher agent is called a zone coalition leader. The zone coalition leader can either be initially assigned or can be dynamically chosen through some voting process. The lower level of the hierarchy consists of individual robots that are grouped together according to particular sectors or capabilities. Each group also has a leader. These group leaders are also grouped according to their sector. Each such group of leaders is associated with a zone group leader. Figure 7 illustrates this structure. Zone group leaders are also grouped according to associated sectors with their own zone group leader. Individual robots are mobile. They may therefore change their group when changing their area. The zone group leaders execute a load balancing algorithm so that if too many tasks appear in a particular sector, agents from other sectors can be diverted from other sectors through negotiations with other zone leaders [Yadgar et al. 2002]. DDM is also *fault tolerant*. If a zone leader becomes disabled, DDM reorganizes itself so that another agent will take its role.

The role of a zone coalition leader is to distribute robots of the appropriate *general* capability to sectors where they are needed. Just as with the taxi dispatcher, it need not concern itself at this high level of the details of *how* the agents will perform their tasks. This assignment of robots to sectors can be accomplished in one of several ways, depending on the degree of global knowledge that the zone coalition leader has. The zone coalition leader might base the distribution of robots within its sector through:

1. knowledge of the state of coverage of the sector under reconnaissance, derived from output from the spatial reasoning

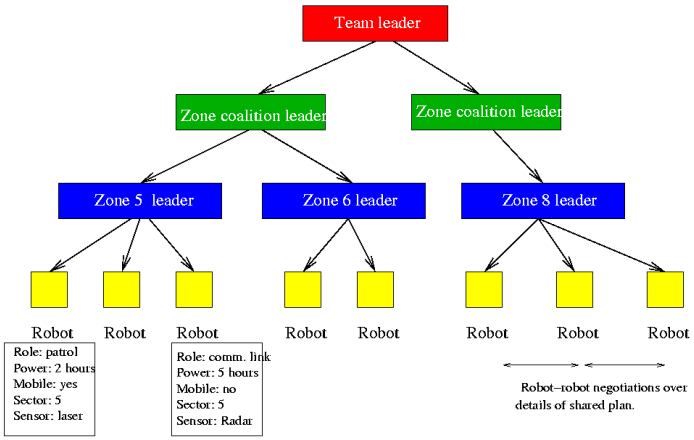

Figure 7. The Distributed Dispatch Manager (DDM).

process described earlier.

2. the distribution of tasks within a sector. For example, the need for a certain number of patrol and communications relay robots.
3. auction-based mechanisms in which the zone coalition leader queries team members for their availability and their potential contribution (utility) to a task.

REFERENCES

B. Bellur, and R.G. Ogier. 1999. "A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks," *Proc. IEEE INFOCOMM '99*, pp. 178–186.

R.G. Ogier, M.G. Lewis, F.L. Templin, and B. Bellur. 2002. "Topology Broadcast based on Reverse Path Forwarding (TBRPF)," draft-ietf-manet-tbrpf-06.txt, (November) (work in progress).

M.D. Yarvis, W.S. Conner, L. Krishnamurthy, A. Mainwaring, J. Chhabra, and B. Elliott. 2002. "Real-world experiences with an interactive ad hoc sensor network," *Proc. International Workshop on Ad Hoc Networking (IWAHN)*, pp. 143–151 (August).

O. Yadgar , S. Kraus and Charles Ortiz. Hierarchical organizations for realtime large-scale task and team environments. *AAMAS 2002*.

E. H. Ruspini. On Truth and utility. *ECSQAU*, Marseille, France, October 1991.

W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. *ICRA* 2000.

A. Doucet, N. de Freitas, and N. Gordon, editors. Sequential Monte Carlo in Practice. Springer-Verlag, New York, 2001.

D. Fox, W. Burgard, H. Kruppa, and S. Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots*, 8(3):325--344, 2000.

D. Fox, S. Thrun, F. Dellaert, and W. Burgard. Particle filters for mobile robot localization. In Doucet et al.

J.S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. *CIRA 2000.*

R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. *AAAI 2000*.

S. Thrun, W. Burgard, and D. Fox. A real-time algorithm for mobile robot mapping with applications to multi-robot and {3D} mapping. *ICRA 2000.*

B. Yamauchi. Frontier-based exploration using multiple robots. *Proc. of the Second International Conference on Autonomous Agents*, 1998.