Chapter 2

VISUALIZATION AND DEBUGGING TOOLS

1 OVERVIEW

The ANTs environment consists of numerous distributed software agents and hardware components. System development in such an environment requires a means of observing and validating the potentially hidden activities of these components. Existing visualization and debugging tools provide some mechanisms for observing behaviors and detecting faults in individual components, but the fast-paced nature of ANTs agents makes these conventional user interfaces (visualizations) and debugging techniques less effective. This chapter will discuss several techniques for visualizing and debugging complex, real-time, agent-based systems. These techniques vary in their level of invasiveness and general applicability.

2 VISUALIZATION

User interfaces are a critical means of visualizing and verifying the correct behavior of a system. This section discusses visualization strategies for agent-based systems. The primary goal of these visualizations is to support the developers and testers in observing and debugging agent-based systems, although they have also proved useful for explaining domain and solution concepts to third parties. Two visualization strategies are discussed next: (1) infrastructure visualization and (2) agent visualization.

ANTs agents negotiate over the optimal use of radar nodes to track an unknown number of targets within a given two-dimensional space (the altitude dimension is removed from the ANTs challenge problem). The sensors are key in this setting since their measurements are the raw data that is used to interpret targets locations. Two limitations exist in the ANTs challenge problem that constitute key complexities:

(1) only one of the three sensors on a radar node (each covering approximately one third of the angular space around a node) can take measurements at any given point in time. Radars thus have 'blind spots' through which targets can move without being detected.

(2) only a limited bandwidth for agent communication exists. This limitation prevents all sensor measurements from being shared among all agents.

ANTs agents are deployed in a distributed setting where no single agent has complete knowledge of the overall system state and activities. ANTs agents thus have a limited knowledge of their surrounding world. Therefore their goal is to draw conclusions based on the limited knowledge they have available, including knowledge they are able to acquire through communication with other agents. Agent-level visualization is thus about how the agents perceive the world, and is usually agent-specific, since different solutions likely differ in how they make and respond to those perceptions.

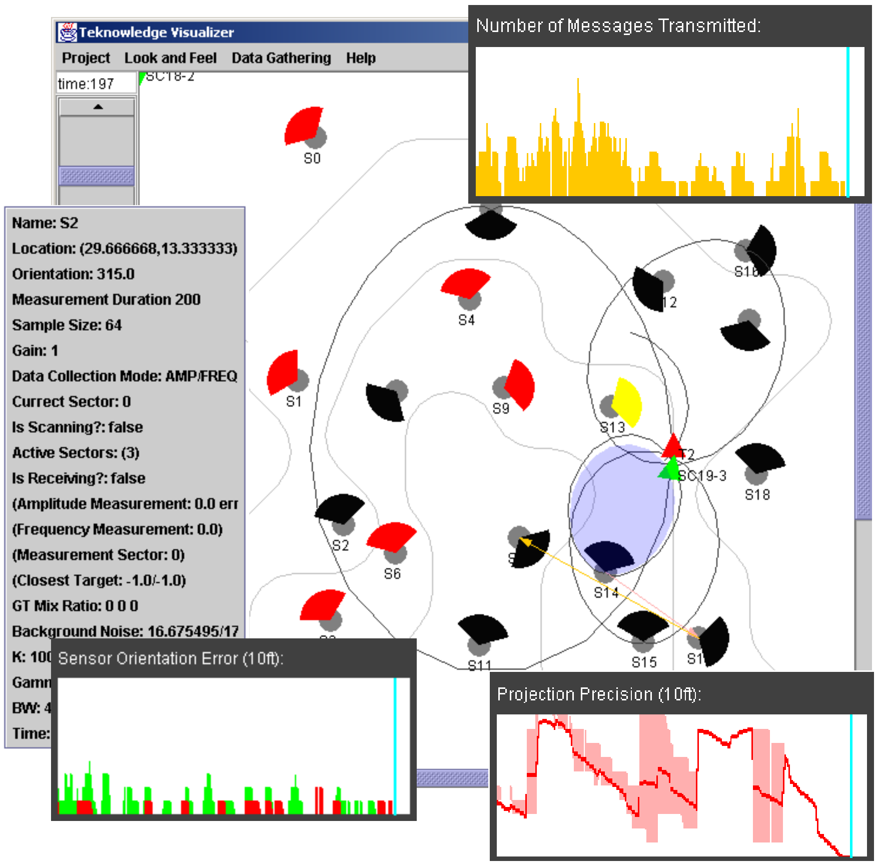
The alternative to agent visualization is infrastructure visualization, which captures global knowledge about the world by operating external to the agents. This world knowledge may be comprised of a combination of things the agents have access to (e.g., sensors) or not have access to (e.g., target location). Since there is only one world and since all agents are part of the same world, only one infrastructure visualization is needed.

Infrastructure visualization is generally applicable and independent of individual agent implementations, but this independence has other limitations. Infrastructure visualization cannot observe agents directly. Instead it instruments the underlying infrastructure used by the agents, and then monitors that usage to track activities. Infrastructure visualization thus captures raw resource usage, such as sensor measurements being taken, but it does not attempt to provide rationale for why agents behave the way they do. Instead, infrastructure visualization is useful in evaluating the quality of agents as a measure of some solution-independent constraint. For example, the goal of the ANTs agents is to optimally track all targets at all times. This is a global constraint that can be evaluated using knowledge of current target locations and sensor orientations.

In summary, agent visualization provides rationale for agent behavior but it is hard to evaluate their effectiveness in satisfying the global goal. Agent visualization is useful in determining whether agents behave optimally given the limited knowledge of the world they have. Alternatively, infrastructure visualization is a means of judging the quality of agents and it provides a mechanism for comparing different solutions along a common set of metrics. It is also a means of detecting flaws in agent behavior based on some global properties they violate. A combination of infrastructure and agent visualizations is desirable to detect inconsistencies between the actual state of the world (the infrastructure) and the presumed state of the world (the agent).

2.1 ANTs Infrastructure Visualization

The ANTs infrastructure consists of hardware components such as sensors, targets, and communication channels, and it consists of software components such as agents, data interpretation components (trackers) and support libraries. Additionally, a simulation environment called Radsim exists, which simulates the behavior of the hardware components for testing purposes. The simulated software components have interfaces that are (almost) identical to the real hardware components. Agents are thus executable on both with only minor alterations.



Name: S2
Location: (29.666668,13.333333)
Orientation: 315.0
Measurement Duration 200
Sample Size: 64
Gain: 1
Data Collection Mode: AMP/FREQ
Currect Sector: 0
Is Scanning?: false
Active Sectors: (3)
Is Receiving?: false
(Amplitude Measurement: 0.0 err
(Frequency Measurement: 0.0)
(Measurement Sector: 0)
(Closest Target: -1.0/-1.0)
GT Mix Ratio: 0 0 0
Background Noise: 16.675495/17
K: 100
Gamm
BW: 4
Time:

Figure 1. Teknowledge Visualizer illustrates Infrastructure Usage and operates in Real-Time and Playback Mode

The ANTs infrastructure originally did not fully support visualization; only Radsim was capable of visualizing some simulated components (e.g., target movement, sensor modes and orientation). It was therefore necessary to instrument the ANTs infrastructure, both hardware and simulated, to capture all data necessary for visualization. This was achieved by instrumenting the interfaces to and from the individual components. For example, the agents gained access to the sensors and communication medium through a library of Java classes. Instrumentation code was added to those Java classes to intercept and forward required data to the visualization tool itself. The visualization tool then interpreted this data and visualized it in a meaningful fashion. The following describes briefly what hardware and software components were instrumented and why.

Radars/Sensors

Sensors measure amplitude and frequency values that give clues about the location, movement direction, and speed of targets. Only one type of sensor was available but it was capable of operating in four different modes. Each mode implied restrictions on the type of measurement taken (amplitude and/or frequency) and the duration required to take them (1-3 seconds). The purpose of the instrumentation was to capture information about the current state of all sensors (their location, orientation, and other attributes), the times and kinds of measurements, and the actual measurement values. Radars are illustrated in Figure 1 as gray circles with colored cones where the cones depict the orientation of the sensors (the active sensor head) and the colors of the cones depict the mode being used to take measurements. Amplitude measurements taken by sensors are depicted using ellipses, where the sensed location of the target is along the circumference of the ellipse. Frequency measurements are depicted as cones where the sensed location of the target is within 15ft of the cone's direction .

Targets (Ground Truth)

Recall that the purpose of the agents' negotiation was to use available resources to detect and track the various targets moving through the environment. An important component of evaluating the progress of this task is to know where the targets actually are, including their individual locations, movement directions, and speed, which can be compared with the agents' results. Instrumentation was added to obtain real time target location, direction, and speed. This was straightforward in the simulated infrastructure; since the target was itself simulated it was straightforward to get the required information. Instrumenting the hardware infrastructure was much more elaborate because accurate data needed to be generated from the model trains which served as targets. To do this, synchronization points were added to the train tracks, and whenever the train would pass a sync point, the time of this event was recorded. Given that the track layout, location of sync points and train speed were known in advance, it was then possible to create a prediction algorithm that interpolated train location, direction, and speed. A more advanced version of the prediction algorithm also used a wireless mouse to monitor train movement, where the ``distance traveled'' by the mouse was used to determine location on the track given a fixed starting point. Targets are depicted in Figure 1 as red triangles were the sharp end of the triangle indicates target direction.

Communication Channels

The infrastructure supported several types of communication medium to determine the system's performance under different conditions. Instrumentation captured the utilization of the individual communication channels and details about the messages sent across them, such as message origin, destination, and length. Message traffic is depicted in Figure 1 in form of arrows between radar nodes.

Trackers

The tracker was capable of projecting presumed target locations, directions, and speeds based on sensor measurements. Instrumentation of the trackers captured the data given as input (amplitude and frequency measurements of sensors) and the data produced as output (target location, direction, and speed). The projected targets are depicted in Figure 1 in the form of green arrows. Input measurements given to trackers are depicted in the form of colored measurement cones (blue ellipses for amplitude measurements and red cones for frequencies).

Quality Criteria or Indicators

Although the ANTs infrastructure visualization does not visualize agent-specific information, which is discussed later, it can be used to define quality criteria and indicators that define good agent behavior. For example, any given radar should be active and searching in the right direction when a target is close. If it is inactive or not searching in the right direction then it may have missed an opportunity to provide measurement data for the tracker.

A special feature of the ANTs infrastructure visualizer is its ability to visualize infrastructure usage in real-time, where changes to the infrastructure are displayed as they are occurring. The instrumentation of the infrastructure utilized a fast network to send data to the visualizer. In cases where the network usage of the instrumentation could have adversely affected agent negotiation (e.g., RF), an alternative network was used. The visualizer was also capable of storing instrumentation data for later playback.

2.2 ANTs Agent Visualization

Infrastructure visualization provides important functionality, not only because it is generally applicable to all programs developed using the infrastructure, but also because it allows different solutions to be evaluated using a common framework. What it does not provide, however, are solution-specific services. Different solutions to the same problem can and will use vastly different approaches, and general infrastructure-level visualization is unlikely to be able to capture and display all the nuances and complexities which make these approaches interesting. Thus, while infrastructure visualization is good at describing how well an approach is working in general terms, agent-specific instrumentation and visualization is needed to capture, evaluate and debug how a specific approach functions.

To varying degrees, all of the teams involved in the ANTs project utilized instrumentation and visualization as part of their debugging cycle. This section will discuss the efforts of the UMass team, which utilized these techniques for several different aspects of their solution. The instrumentation used by the UMass agents fell generally into two categories: general agent-level information, and data that was collected on specific techniques.

The most basic type of instrumentation consists of logging simple state information out to a file. As critical points in the program are reached, important messages arrive, or failures occur, a simple text message can be written to a log file. Many programs do not require anything more complex than this, but when dealing with a distributed population of interacting entities, each of which can be a complex autonomous program, the amount of information generated by this technique can quickly become overwhelming. In addition, the distributed nature of an agent-based solution will mean that events in one agent will affect or be affected by the events in other agents. To get a global view of the agent's activities, one must reconcile these events in such a way that one can see these interactions.

The information quantity issue was resolved using a facility code technique, similar to that used by the `syslog` logging daemon typically found in Unix environments. Each log message is marked with one of several log levels, indicating its importance, and a facility code indicating the component in the agent that generated the message. Log level assignments can then be used to restrict logging output to only those parts of the system being actively investigated, thereby reducing the signal-to-noise ratio in the log file and making it easier to identify pertinent events. Distributed actions may be reconciled by marking each written log message with a time stamp, which can be used to produce a global timeline where agent interactions may be identified.

Even with smaller, more focused data files, it is still difficult to obtain a more global view of activities in the system because these data sources are segregated. To provide such a global view, a general log viewing utility was developed that was capable of graphically interpreting the events in an agent's log file and placing them in a timeline along with the events of other agents. Figure 2 shows such a display after processing log files from six agents.

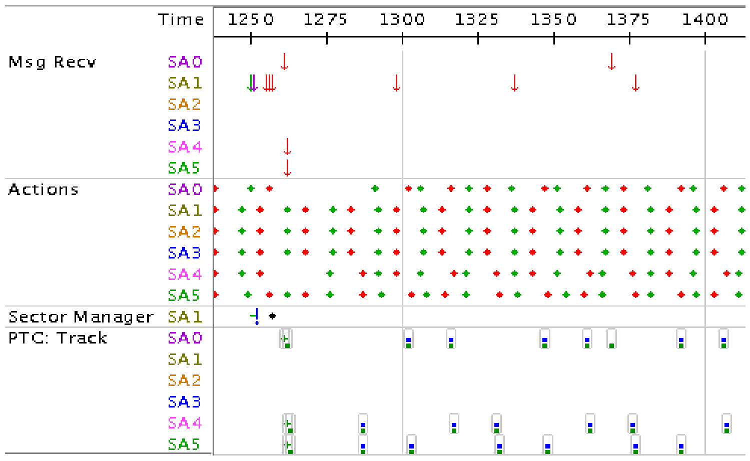


Figure 2. A consolidated timeline of agent activities.

The log viewer reads the agents' log files and applies a configurable set of filters to each of them. Each filter is responsible for visualizing a certain class of log messages. For instance, in Figure 2 there were filters for tracking messages and actions, among other things. Because each sequence of events is aligned relative to the same timeline, the operator can quickly inspect what was occurring at a given point in time and deduce what actions might have led to that point. This utility can either perform a post-hoc analysis or render the data in real time by monitoring agents' log files as they are produced.

In addition to the general logging behavior described above, agents also included more detailed instrumentation of particular aspects of the solution. Tracking is a good example of this. In addition to the measurements and target state recorded by the infrastructure, tracking agents would also save solution-dependent aspects of this process. For example, received measurements might not be used in the tracking process because they were low quality, too old, or potentially ambiguous. Because they were discarded, the infrastructure would not know of these measurements, and therefore be unable to track such behavior. These were logged in a separate location, along with other information pertinent to the tracking process.

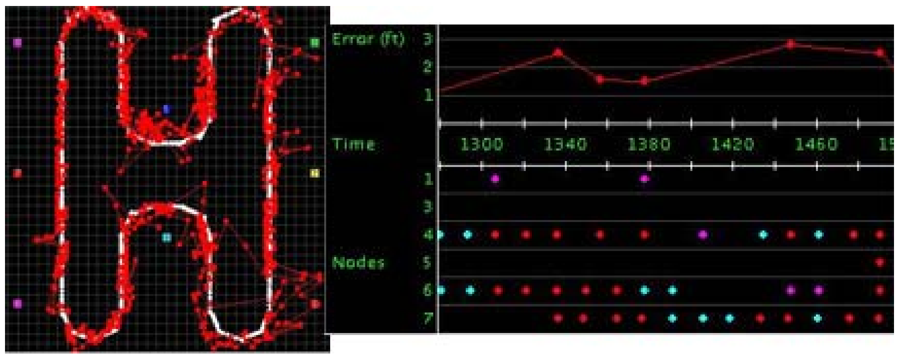


Figure 3. A specialized view of agent tracking behavior.

The augmented tracking data is used by the track visualizer shown in Figure 3. This view differed from the infrastructure visualization in its ability to display solution-specific aspects of the tracking process, primarily in its timeline of events. This timeline tracked which agents produced measurements, and how they were or were not incorporated into a track. This particular utility was the first such tool that allowed one to replay and both quantitatively and qualitatively analyze the results of tracking activity.

For extremely fine-grained information, a visualizer was developed to provide a real-time view of an agent's internal state. Much of the logic implemented in the UMass agents use geometric shapes as the basic quantitative unit, so this information mapped naturally into a graphical display. This interface, shown in Figure 4, uses these same geometric primitives being used for decision making to compose the image on the screen. In Figure 4, the internal state of a tracking agent is presented, which shows its knowledge of other sensors in the environment, where it believes the target to be, which sensors it has allocated for tracking, and notions of uncertainty, among other things. To the left of the image, a grid layout shows the state of negotiation, by indicating the specific sensors and times that have been requested as part of the tracking process. This very detailed, yet concise view of an agent's state was invaluable when tracking down undesired behaviors, as the exact mapping between the image and the agent's working state made such problems readily apparent.

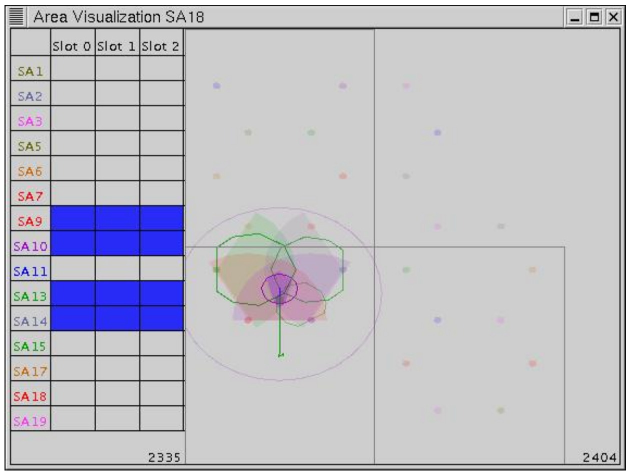


Figure 4. A view of an agent's internal state.

Simple statistical measures also provide an important, less graphical way of analyzing performance. The general log files were used to extract a number of useful, descriptive measurements. For example, when analyzing the communication medium utilization, a comprehensive histogram of message types and lengths was generated using the simple “message received” log messages. Agent scheduling and execution activity was similarly tracked. By using the time stamp on each log message, it was also possible to approximate crude profiling behavior, by looking for points in the timeline where certain activities would take longer than expected or desired.

3 Debugging

3.1 Visualization as a debugging aid

It should be clear how visualization generally assists one in determining how a particular system is functioning. The deeper question is whether or not the benefits they provide outweigh the time required to develop such tools. Based on experiences in the ANTs project, we would certainly answer this affirmatively, particularly when the environment is complex or distributed. The tools described here consistently found and helped diagnose problems that would otherwise have remained hidden. As such, the various agent and infrastructure visualizations helped detect negotiation flaws, performance bottlenecks, target confusions during multi-target tests, and many more.

3.2 Controlling the Environment

Implementing ANTs agents for an ideal world is already a non-trivial task, but the real-world environment of ANTs is further complicated by (1) unreliable, limited communication and (2) sensor measurement noise. Each sensor platform had built-in support for wireless RF (Radio) communication with the drawback of low bandwidth and a likelihood of message loss that increased with bandwidth usage. Additionally, the measured values produced by the sensors would deviate from their theoretical performance due to a combination of known and unknown external effects. Of course, these complexities are to be expected in any real-world environment. Nonetheless, it is hard to design and validate systems where the effects of noise are not well understood. In an effort to control noise for purposes of debugging and testing, a pair of mechanisms were added which could compensate for these effects.

The simulated infrastructure, Radsim, could mitigate both problems because it simulated the noise factors itself. This allowed developers to understand in what ways noise influenced agent behavior, and as such was an important tool when preparing agents for the hardware environment. However, like nearly every simulation, Radsim could not capture all of the complexities of the hardware system with perfect accuracy. For instance, the hardware infrastructure exhibited certain timing complexities (including non-determinism) that were not easily captured by Radsim. Consequently, it was also necessary to mitigate network and sensor noise in the hardware infrastructure, for validation purposes.

The RF network noise and bandwidth limitations were mitigated through the use of an alternative, high-bandwidth, reliable TCP network. Agents could then be tested and validated separately through both communication methods. The problem of sensor noise was somewhat harder to mitigate, but we found that the instrumentation technique used for visualization could serve dual purposes by providing “perfect” data that could be used to control the sensor noise problem. The train instrumentation provided target location information, which was then used to compute expected sensor measurements based on their theoretical model. These “noise-free” sensor measurements were then mixed with actual sensor measurements in real-time to provide measurements with a controlled amount of noise. These techniques proved useful in facilitating the transition of agent technologies from pure simulation to a hardware-only environment.