

e-DTS 2.0: A next-generation of a Distributed Tracking System

Ryan Rybarczyk, Rajeev Raje, Mihran Tuceryan
Department of Computer and Information Science
Indiana University Purdue University Indianapolis
Indianapolis, Indiana USA
{rrybarcz, rraje, tuceryan}@cs.iupui.edu

Abstract—The goal of this paper is to describe a next generation version of an existing distributed tracking system (enhanced-Distributed Tracking System or e-DTS), called e-DTS 2.0. The new features of e-DTS 2.0 include implementing dynamic discovery and communication between the various system components, improving camera calibration through alternate techniques, addressing clock drift and synchronization, and providing the ability for the system to use coordinate sharing and world transformation. A prototype of e-DTS 2.0 is created and experimented with. The results of these experiments are also described in this paper.

Keywords—enhanced distributed tracking system (e-DTS); end-to-end discovery time (EEDT); end-to-end response time (EERT); augmented reality (AR); quality of service (QoS)

I. INTRODUCTION

This paper describes the improvements made to an existing distributed tracking system (e-DTS) originally created by [1] and further modified by [2]. The underlying premise of the e-DTS is that many inexpensive sensors, such as web cams, can be used to create a comprehensive and efficient distributed tracking infrastructure for indoor usages. This design was shown to be scalable in, and able to communicate and fuse tracking results from, cameras within the system [2]. In the e-DTS there is no concept of a shared or global coordinate system. This prevents the cameras from dynamically discovering the tracking environment they are in and establishing a single coordinate system to use when estimating the position of the tracking marker. Because of this, the coordinate system must be established prior to execution of tracking and each camera must be manually calibrated with these coordinates. The e-DTS also does not handle network faults or interruptions gracefully, as a halt or abnormal termination of one service or the JINI Lookup Service would prevent further tracking of the system. The rest of the paper is organized as follows: related work on the topic will be presented in the following section; the system architecture of the e-DTS is presented in the section III. This is followed by the reasons for the enhancement in section IV; section V presents the design and implementation details for the e-DTS 2.0; section VI presents the results gathered from experimentation as well as associated analyses. The paper concludes with the lessons learned and suggestions for future work to be carried out.

II. RELATED WORK

There have been many efforts that address various aspects of distributed tracking systems. For example, Camera Calibration is studied in [13], [14], and [15] where various camera calibration techniques to improve the capture quality of the cameras attached are examined. Improving the calibration of the camera will yield more accurate tracking results when applied to the e-DTS 2.0. [16] provides a good treatment on Global Coordinate Systems with a possible solution for resolving this issue in an ad-hoc sensor network. Through work done with regards to establishing a Global Coordinate System the e-DTS 2.0 will be able to provide more accurate estimations of the Tracking Marker within the environment. For Dynamic Discovery using JINI, [17] provides a recent look at dynamic discovery in a vehicle network. [18] takes a closer look at JINI and how it relates to ad-hoc wireless sensor networks, and proposes some enhancements to the underlying JINI structure to make it more robust and useful for tracking applications. The ability to use dynamic discovery in the e-DTS 2.0 will allow for a higher degree of fault tolerance.

III. E-DTS SYSTEM ARCHITECTURE

The e-DTS system was proposed in [1, 2] and is based on the underlying premise that effective indoor tracking can be achieved as a federation of inexpensive sensors such as web cameras. The e-DTS is built upon the ARToolkit API using a Java binding [3]. This API provides the core video processing necessary for the e-DTS. The ARToolkit was developed to serve augmented reality (AR) applications but provides an adequate solution for tracking applications. [3] The goal of AR applications is to detect fiducial markers, easily recognizable patterns, and overlay a three dimensional (3D) image within a virtual representation of the objects' actual physical position [3]. This ability to track objects has provided vast opportunities for AR applications to be used in real-time tracking experiments.

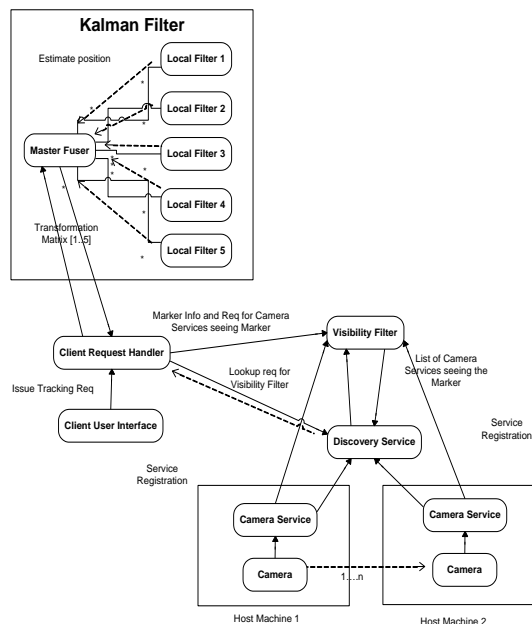


Figure 1. E-DTS Architecture [2]

The following is a discussion of the architecture and overall structure of the e-DTS. The e-DTS as a system consists of Cameras, Camera Services, Visibility Filters, Discovery Service, Tracking Markers, Fusion Filters, and a Tracking Client.

A. Cameras/Camera Services

In the e-DTS, the choice of tracking sensors is fulfilled by utilizing inexpensive web cameras. Two types of web cameras have been utilized within the current e-DTS setup: the Logitech QuickCam and the Micro Innovations Basic Webcam. These two types of cameras are connected to a cluster of PC's running Windows XP SP3 using a USB 2 connection. The e-DTS utilizes the ARToolkit API as a means for capturing an image of the seeing environment via the webcams. The Camera Service is a web based JINI service that sits on top of the physical cameras and provides access to the details provided by the ARToolkit API. The Camera Service makes information available for consumption regarding the current state of the Camera. If the Camera is currently seeing the Tracking Marker then a 4 by 4 transformation matrix is made available that provides the pose, the position and orientation, and the tracker id of the Tracking Marker with respect to the global coordinate system [3]. This matrix allows for calculation of the pose of the object by using simple matrix multiplication. This allows the Tracking Client to use the global coordinate system in order to properly estimate the position of the marker as it moves through the environment.

B. Discovery Service

In the e-DTS, the discovery service provides the means of locating Camera Services. More specifically, the process of discovery involves finding services that are available

within the network and establishing a connection with the lookup service. In the e-DTS the discovery portion has been implemented utilizing JINI [4]. In the e-DTS, Unicast discovery is used as a means to communicate with the JINI services. Each Camera Service is provided the location of the JINI lookup service and is able to register and communicate with that particular lookup service. If the service becomes unavailable, then the system is unable to communicate to any other entities in the e-DTS.

C. Tracking Markers

Tracking Markers are images that are created by a tool provided with the ARToolkit API that can then be used and tracked by the e-DTS. The marker itself is a simple and distinctive pattern which is recognizable by the Tracking Service. These patterns contain dark contrasting colors, generally black and white, to make it easy for the pattern to be distinguished when placed in an environment [3].

D. Visibility Filters

The Visibility Filters used in the e-DTS act as a proxy between a Client object and the JINI Discovery Service. When a Client object requests the access of a particular Camera Service, it will communicate with the Visibility Filter which will in turn attempt to find all registered Camera Services. The Visibility Filter will then allow direct access to any Camera Service that is currently registered and actively seeing/tracking the marker pattern as it moves throughout the environment. This direct access makes it easier for any Tracking Client to quickly obtain information regarding all the Camera Services without having to poll each one.

E. Tracking Client

The Tracking Client is a front-end application that interacts with the Visibility Filter with the sole purpose of retrieving data from the Camera Services in order to track a given Tracking Marker. The Tracking Client receives information regarding the position of the marker with respect to the Global Coordinate System and then is able to perform fusion on any results given as a means to standardize the data. The Tracking Client also provides the means to collect statistics on the overall e-DTS system as well as the various services encountered within the system. As a result, the Tracking Client is the user interface of the e-DTS but also has the capability to serve as a Tracking Marker itself.

IV. REASONS FOR ENHANCEMENT

As indicated earlier, the e-DTS system is based upon the Unicast Protocol [2]. This does not allow for dynamic discovery of services within the local network. In the e-DTS, the location of the JINI Lookup Service must be explicitly stated to each Camera Service, where the location of JINI Lookup Service #2 has been provided to the Camera

Service. In addition to this, the JINI Lookup Service must be actively running and accepting queries. In this scenario, if the JINI Lookup service is interrupted or terminated a subsequent error will be generated by the e-DTS and the system will terminate as no further communication can take place since it relies on a central location. In this example, JINI Lookup Service #2 is the only location that the Camera Service registers itself with, thus ensuring that if the service fails, the e-DTS will also fail. Due to this shortcoming, the e-DTS is not very flexible and is unable to handle faults with regards to service discovery and communication.

Also in the e-DTS, there is no Global Coordinate System shared between the various cameras involved in tracking. Therefore each camera must be first calibrated with respect to another camera in an effort to determine its location. This does not allow for a seamless handoff between cameras as the marker is moved within the environment of the e-DTS.

A critical goal of any distributed system is the need and desire for time synchronization. Without such synchronization, two processes or queries could become isolated due to timing constraints. Because tracking demands real-time processing, there is a need for the clocks of the various machines in the e-DTS to be synchronized in an effort to offset clock drift. Such a synchronization is necessary as readings from individual cameras are fused together to obtain the position of the tracked object. This synchronization is critical when attempting to perform Kalman-based fusion on the data sets returned in the tracking process. Kalman-based fusion was introduced into the e-DTS, but without sufficient clock synchronization the clock drift negatively impacted the effectiveness of the Kalman fusion.

Finally, at the heart of any tracking application is the calibration of the trackers themselves in this case the Cameras. The calibration tools provided with the ARToolkit API, used by the e-DTS, are not designed to be used for tracking applications [3]. The more accurate the calibration, the more accurate the tracking results will be. Therefore, a better calibration tool or method is needed in an effort to improve the overall results of the Camera tracking.

The e-DTS 2.0 addresses all these limitations of the e-DTS as indicated in the next section.

V. E-DTS 2.0

This section discusses the improvements that the e-DTS 2.0 proposes in order to overcome the limitations of the e-DTS (described in the previous section). Only the entities that were modified to create the e-DTS 2.0 from the e-DTS are described here for the sake of brevity.

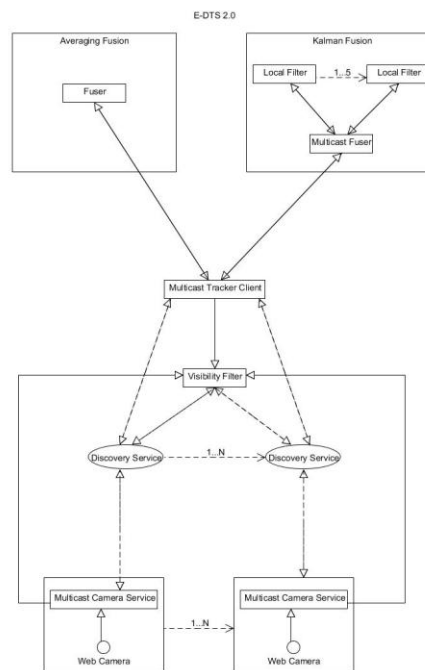


Figure 2. E-DTS 2.0 Architecture

The Camera Service has been modified to utilize the power of multithreading to constantly scan the local network for any JINI Lookup Services present. If a lookup service is found, then the Camera Service will register itself with it and store the location of the lookup service. This is done in an effort to provide fault tolerance to the system by allowing it to re-register with any previously known lookup services if a fault should occur. The dynamic nature of the discovery and communication allows for inter-camera communication and the sharing of coordinate systems.

The Visibility Filter has also been modified to utilize the Multicast Protocol. This was achieved by using multithreading to provide continuous background processing of the local network to locate any available JINI Lookup Services. If a Lookup Service is found, then the Visibility Filter will register itself with it and check to see if any Camera Services are currently registered with that particular Lookup Service. If there are Camera Services registered with the Lookup Service, then the Visibility Filter will collect the information on these services. This allows the Visibility Filter to serve as a proxy of the Tracking Client.

Finally, the Tracking Client has also been modified to utilize the Multicast Protocol as a means for discovery and communication within the e-DTS 2.0. This allows the Tracking Client to use multithreading to continuously scan the local network for any available JINI Lookup Services. If one is found, it can then identify any Camera or Visibility Filter Services available. If any are found, the data with those corresponding services can be collected and used

together by the client as means to provide an accurate estimate of the position of the Tracker Marker.

A. *Discovery Service*

As indicated earlier, the e-DTS uses the Unicast protocol for discovery and communication with the JINI Lookup Service. This protocol does not allow for dynamic discovery and communication or the ability to handle faults within the system [5]. The e-DTS 2.0 overcomes this limitation by the use of the Multicast Protocol and multithreading. This protocol will allow for dynamic discovery, registration, and communication between the Camera Services and the JINI Lookup Service. In addition to this, the use of multithreading will allow background processing to constantly check the status of the local network.

B. *Global Coordinate System*

As an object traverses through a tracking environment, it may move out of the scope of one camera and may enter the range of another camera(s). Hence, a handoff between cameras will need to occur in order to continuously estimate the current position of the Tracking Marker with respect to its environment. This handoff will need to occur between cameras not only located within the local network but also those possibly located outside the network. For this handoff to take place, the e-DTS 2.0 uses a global coordinate system in the tracking infrastructure.

In the absence of a global coordinate system, the Tracker Client is designed to handle the handoff of the camera coordinate systems between the environments. The tracking status can therefore be categorized in one of the following categories: one environment, containing cameras, seeing a Tracking Marker; multiple environments, containing cameras, seeing the same Tracking Marker; no environments, containing cameras, seeing a Tracking Marker. In category two, in which multiple environments are currently tracking the same Tracking Marker the Tracking Client has two options: either use the environment that currently has more cameras seeing the Tracking Marker, or convert the environment with the fewer number of seeing cameras coordinate system to that of the first.

C. *Clock Drift and Synchronization*

Various software tools, such as Atomic Clock [8], Domain Time II [9], and Windows Time Service [10], exist that help to alleviate the issue of clock drift in distributed systems. As fusion places strict restrictions on the clock drifts, these existing approaches need to be evaluated before an appropriate alternative is used in the e-DTS 2.0. Once a tool has been selected, it will be possible to build a lightweight component that can monitor clock drift within the e-DTS 2.0 and provide feedback regarding the current status of clock synchronization within the system. Experiments were carried out with these tools and a suitable

choice was selected as indicated in the section on experimental analyses.

D. *Camera Calibration*

As indicated earlier, the heart of the ARToolkit API [3] is a calibration file that is unique to each web camera. This file contains the camera data that is used when estimating the position of a Tracker Marker within the camera view. The e-DTS used a simple calibration technique [3], which yielded barely acceptable results of the tracking process. The e-DTS 2.0 addresses this shortcoming by evaluating various alternatives for camera calibration and selecting the most suitable one [11].

E. *Dynamic Discovery*

The e-DTS utilizes the Unicast discovery protocol and therefore each Camera Service must be explicitly provided with the location of the JINI Lookup Service. However, in an unknown tracking environment this knowledge may not be known initially and must therefore be discovered and learned by the Camera Services.

For the dynamic discovery of other services in tracking environment, the e-DTS 2.0 uses the concepts of multithreading and the Multicast discovery protocol. This will allow for a dynamic discovery of the local network on which the e-DTS 2.0 is deployed and keep the e-DTS 2.0 up-to-date in terms of the underlying network topology [5].

F. *Fusion Tracking*

The e-DTS proposed by [2], implemented a Kalman based fusion technique in an effort to provide an accurate estimation of the position of a given tracking marker. The goal of using this fusion technique was to provide a more accurate estimate of a position in an environment in which noise exists. However, a key timing constraint surrounds the usage and effectiveness of Kalman fusion. Due to the clock uncertainties and drift in the e-DTS, the Kalman-based fusion was not able to be fully maximized and applied during the tracking process.

In an effort to enhance the effectiveness of Kalman-based fusion, the e-DTS 2.0, as previously mentioned, will evaluate various software tools available in an effort help alleviate clock drift between the host machines. Because of this ability to synchronize the internal machine clocks at a finer grain, and much more accurately, the e-DTS 2.0 will be able to fully maximize the benefits of Kalman fusion during the tracking process.

VI. EXPERIMENTATION AND ANALYSIS

The above mentioned enhancements were implemented to create a prototype of the e-DTS 2.0 and it was empirically validated. In this section, various experiments performed

with the prototype of the e-DTS 2.0 and the associated results are described.

A. Clock Synchronization

The first experiment performed for addressing the issue of the clock synchronization was to compare the effectiveness of various software tools on reducing the clock drifts. The three tools that were used in the experiment were: Atomic Clock [8], Domain Time II [9], and Windows Time Service [10]. Each of these tools was selected after an extensive search for applications that provided high accuracy in clock synchronization.

The time servers used in the experimentation for each tool were as follows:

- time.nist.gov
- ntp.iupui.edu

If the first time server is unavailable, then the second time server would be used as a fall back alternative. The reason why these two servers were selected was that it provided an external and internal time (i.e., within Author’s University) server to use when performing the experiments. Each of the machines within the e-DTS 2.0 was configured so that the clocks would be synchronized every second, with the option to vary this parameter.

For the experimentation, a lightweight and basic Java application was constructed as a means to monitor clock drift and subsequent clock synchronization for the machines in the e-DTS 2.0. This application made use of the Unicast Protocol as a means for communication. Each experiment was executed for 48 hours with samples taken and written to a text file every 10 minutes. The experiment was run a total of 4 times, once for each of the software tools selected and a fourth without any tools running. The interval of sampling was chosen to allow for enough time to pass between each reading to adequately see if the clocks were drifting or if they were effectively being synchronized. The duration of the experiment allowed for sufficient time to pass and thus allowing the system to stabilize and to provide a quality sample size.

Through these various experiments (results of which are shown in Table I), it was observed that the Domain Time II tool worked the best in keeping the machines synchronized to time.nist.gov time server. This was also the case for the ntp.iupui.edu time server.

TABLE I. CLOCK DRIFT AVERAGES

Tool	Average Clock Drift (milliseconds)
Atomic Clock	665.7693
Domain Time II	7.5699
Windows Time Service	2035.0400

After the conclusion of experiment one, further analysis was done on the Domain Time II client to locate and identify the maximum and minimum clock drift errors amongst the various samples collected. It was found that the maximum clock drift error was 48 milliseconds and the minimum was 0 or the absence of clock drift between the client machine and the host time server. The clock drift error is defined as the variation of the local machine clock with respect to the time kept by the time server. Table II shows the full results in milliseconds for a sampling of machines involved in this experiment.

TABLE II. DOMAIN TIME II CLOCK DRIFT

Machine:	1	2	3	4	5	6	7
Max Error	19	29	24	20	48	32	20
Min Error	0	0	0	0	0	0	0

Hence, it was concluded that the Domain Time II software tool provides adequate clock synchronization with minimal clock drift for the machines involved in the e-DTS 2.0 and thus, it was incorporated in the e-DTS 2.0 implementation.

B. Discovery & Tracking

To empirically validate the Discovery and Tracking, two sets of experiments were performed. These experiments aimed to test both the ability of the Multicast Protocol to serve in its role of discovery and communication as well as the cost of its implementation to the e-DTS 2.0.

The first experiment observed the time taken for the entire tracking process, both with the Unicast and Multicast protocols. The entire process of tracking consists of the cameras locating the object, providing the calculated estimation of the objects location, communicating this estimation to the tracking client, and finally, the tracking client gathering these results and performing fusion on them. This time is referred to as the End-to-End Response Time (EERT). The experiment was run with 20 Camera Services present in the environment and one Tracking Marker. The Table III, shows the average EERT for both the Multicast and Unicast models of communication.

TABLE III. AVERAGE EERT

Protocol	Average Total Tracking Time (milliseconds)
Unicast	36.1
Multicast	47.6

The results in Table 3 show that the overhead for Multicast communication is slightly higher than that of the Unicast. However, this does not take into consideration the

benefits of the Multicast Protocol: dynamic discovery, group communication, and fault tolerance.

The second experiment measured the End-to-End Discovery Time (EEDT) of the e-DTS 2.0. The purpose of this experiment was to determine the impact the choice of discovery protocol had on the time required for service discovery. With real-time timing constraints, the need for fast discovery is vital. The EEDT is defined as the time from a service request to find a JINI Lookup Service, to the discovering and registering with the service. This was carried out on the same 20 machines using a host JINI Lookup Service as the service provider. Two experiments were carried out for each protocol, Multicast and Unicast. The first of these experiments was run for a total duration of 30 seconds and the second being run for a duration of 60 seconds. These sample constraints were chosen in an effort to get a manageable sample size of service discoveries as well as provide the ability to capture the most accurate startup and registration time. The results of these two tests are shown below in Table IV.

TABLE IV. AVERAGE EEDT (MILLISECONDS)

Protocol	30 seconds	60 seconds
Unicast	2.1173	3.1982
Multicast	2.1611	3.2226

As Table IV shows, the overhead cost, in terms of time, is only slightly higher for the Multicast protocol. This data will allow for the e-DTS 2.0 to be customized based upon the application of its use.

The final experiment, for Discovery and Tracking, was to test the fault tolerance of these two protocols. In the e-DTS when the JINI Service was temporarily interrupted or brought down, the entire e-DTS failed and was unable to reestablish a connection. In the e-DTS 2.0 using the Multicast Protocol, the Camera Services were able to re-discover the JINI Lookup Service and re-register without any user interaction. In addition, the Visibility Filters were also able to self-heal and rediscover the JINI Lookup Services or discover new services to register with. This was achieved by utilizing multithreading as a means for background processing and the Multicast discovery protocol for dynamically locating and registering with the lookup service. This demonstrates that in the presence of a fault the e-DTS 2.0 is able to perform much better than that of its predecessor.

The results of these various experiments demonstrate that while the Multicast Protocol does indeed introduce a higher overhead, especially when it comes to the time required for discovery and communication, the benefits of its dynamic nature outweigh the costs.

C. Sharing Coordinate Systems

The concept of a shared world coordinate system and the ability for remote tracking were key components in the design of the e-DTS 2.0. In this set of experiments, the e-DTS 2.0 infrastructure was setup and based upon the common hierarchy of an army - where there are leaders and soldiers and rank and order are established and followed. In this hierarchical setup, the e-DTS 2.0 uses the concept of groups, first level services, and second level services. These groups have been divided and ranked accordingly based upon their predetermined Quality of Service (QoS) parameters. For the e-DTS 2.0, the rank is established as an integer value with the initial value of 0 being given. A rank of 0 signifies no rank, with the possibility of being incrementally increased (i.e., 1, 2, 3 ...N) based upon the QoS parameter selection. Each camera is assigned a rank based upon its parameters. These QoS parameters are: Camera Resolution, Camera Distortion, Camera Frame Rate, Relative Location, and Clock Drift. These parameters, based on their nature, were grouped into the two classes: static and dynamic. Camera Resolution, Camera Distortion, and Camera Frame Rate are static parameters that are assigned when the cameras are initialized. Location and Clock Drift are dynamic parameters that can vary camera to camera based on their deployment environment. Because each camera is aware of its relative location, this is achieved by calibrating the camera and utilizing the calibration data file created; its coordinate system could then share this coordinate system with other cameras. This is achieved by choosing at random intervals to transmit this information to any known cameras participating in the e-DTS 2.0. The rank is established based upon the given parameters, i.e. if Clock Drift is the parameter associated to determining the rank the e-DTS 2.0 could be configured to assign the rank of 1 to any services with Clock Drift lower than 15 milliseconds and assign a rank of 2 to those greater than 15 milliseconds. This is done by background processing using multithreading of the Camera Service. This sharing of coordinate systems allows for no single camera to assume the role of ultimate leader, but rather enables a group of leaders share the power and thus renders the entire e-DTS 2.0 fault tolerant and dynamic by allowing for the sharing and assigning of coordinate systems to cameras within the e-DTS 2.0.

For the first experiment, the cameras were assigned rankings based solely on their Camera Frame Rate. This parameter was selected due to the ability to specify the frame rate of the camera upon startup. Each camera in the e-DTS 2.0 has the option to have a rank pre-defined or specified dynamically by the system. For this experiment the rank was not pre-defined and instead was determined dynamically by the camera upon initialization. This rank was then provided via the Camera Service. The Tracker Client was utilized to extrapolate the data from the Camera Service and provide it via a text file. Each of the 20 Cameras were initialized 10 of the cameras being initialized with a Camera Frame Rate of 15 frames per second (fps) and the other 10 cameras being initialized with a Camera Frame Rate of 30 fps. The Cameras were then registered with the JINI Lookup Service and made available for

consumption. Next the Tracker Client was run for 1 minute and after 1 minute, all 20 Cameras and their corresponding Camera Services were properly ranked according to Camera Frame Rate. 10 cameras were assigned the rank of 1 and the other 10 cameras were assigned the rank of 2. This ranking system would allow the Tracker Client to make a selection of only cameras with N rank or higher and thus provide tracking results based upon specified camera QoS parameters.

A second experiment tested the ability for remote tracking to take place within the e-DTS 2.0. The purpose of this experiment was twofold: first, the desire to show that remote tracking of an object could be done with the usage of the newly implemented Multicast protocol and second, the desire to show that the coordinate systems could be shared between external networks. The setup for this experiment was achieved by providing the location of the external network to the Tracker Client. This makes use of the e-DTS's Unicast protocol to access both local and external JINI Lookup Services. The primary focus of this experiment was on the nature of a location being provided to a service that no longer existed. If the external or remote service existed, then the results provided in Table IV would suffice and no further testing or analysis would need to be done. Therefore, an alternate experiment was created and was run at intervals of 30 milliseconds, 1 second, 15 seconds, and 30 seconds in an effort to test the overall overhead when attempting to locate a dead, or no longer running, service. Table V below shows the results of this experiment.

TABLE V. AVERAGE EEDT (MILLISECONDS)

Service Check Frequency	< 1 second	1 second	15 seconds	30 seconds
Total Fusion Time (milliseconds)	21084.4000	192.9900	109.7642	71.0930

The results shown above indicate the underlying timeout constraints of the protocol drive the overhead created when trying to access a dead or non-existent service on a remote network. Therefore, a time constraint parameter on the attempt is needed, i.e., if a service has not returned within 15 milliseconds then the attempt is abandoned and the service be marked as dead. This will alleviate the problem of waiting for the service.

Therefore, it can be concluded that remote tracking can be efficiently and effectively achieved, albeit with a slight modification to a parameter with regards to service check frequency. This allows for remote tracking of cameras that are able to share their coordinate systems with one another through inter-camera communication. As a result, this provides a much more dynamic and mobile tracking system that can be used by any tracking client, whether it is located on the local network or not.

D. Camera Calibration

The goal of the following experiments was to determine whether or not the camera calibration described in [11] provided a better camera calibration technique when compared with the default technique available with the ARToolkit [3].

In order to test the accuracy of this new technique, all the cameras in e-DTS 2.0 were recalibrated. This experiment of Camera Calibration accuracy began by calibrating each of the 20 cameras in the e-DTS 2.0. First each of the cameras were calibrated using the method proposed by [11] and samples were taken, then the 20 cameras were re-calibrated using the calibration method provided by [3].

The results of each camera were gathered and analyzed based upon their accuracy with respect to the actual location of the calibration pattern [12]. Table VI shows the average error in millimeters with respect to the position of the calibration pattern:

TABLE VI. AVERAGE CALIBRATION ERROR

ARToolkit Calibration	Alternate Calibration Technique [11]
35.01270 mm	14.6573 mm

These results demonstrate that the calibration technique, proposed by [11], provides a much more accurate camera calibration as opposed to the tool provided by [2]. Therefore, it can be concluded that by using the calibration method proposed by [11] the e-DTS 2.0 will be able to provide much more accurate tracking results.

E. Fusion Tracking

The goal of the following experiments was to demonstrate the accuracy of the e-DTS 2.0 when using fusion techniques to estimate the actual position of the Tracking Marker.

The first experiment to test the e-DTS 2.0 implemented a simple average-based fusion, first described in [2]. This averaging technique simply combines all of the readings for seeing cameras, takes the average for the (x, y, z) coordinates and displays this result as the estimated position of the Tracking Marker. The marker was moved slowly in front of all 20 cameras with the Tracking Client performing the fusion. Upon receiving information of a Camera Service seeing the Tracking Marker, the Tracking Client will both display the position data provided by the Camera Service, the Tracking Marker ID and name, as well as the timestamp associated with that particular reading. This information will also be written to a text file to aid further or future analysis. Table VII indicates the average error, (in millimeters), in the tracking estimate, using simple averaging fusion, for each of the three axes for the e-DTS 2.0.

TABLE VII. ESTIMATED ERROR IN AVERAGING FUSION (MILLIMETERS)

Axis	X-axis	Y-axis	Z-axis
Average Error (millimeters)	59.9810	316.4367	114.8319

The actual physical position of the tracking marker was recorded and the results provided by the Tracking Client were used in calculating the error in estimation. Finally, these errors for the three axes were summed and an average was taken.

The second experiment to test the e-DTS 2.0 implemented a Kalman-based fusion technique, first described in [2]. The marker was once again moved slowly in front of all 20 cameras with the Tracking Client performing the Kalman-based fusion technique. Table VIII indicates the average estimated error, using this technique, when tracking a marker as it moves through the environment.

TABLE VIII. ESTIMATED ERROR IN KALMAN FUSION (MILLIMETERS)

Axis	X-axis	Y-axis	Z-axis
Average Error (millimeters)	16.1141	150.3517	99.9184

As can be seen, the calculated average error for the Kalman-based fusion technique greatly improves over the calculated error found in the simple average technique. Because of clock synchronization in the e-DTS 2.0, a larger quantity of tracking data is able to be used when performing the fusion. These experiments demonstrate that it is possible to perform tracking with the e-DTS 2.0.

VII. FUTURE EXTENSIONS

Many exciting extensions of the e-DTS 2.0 are possible. Future extensions of this work could include, but are not limited to: A scalability study of the newly enhanced Multicast version of the e-DTS. An exhaustive security and performance study of the usage of Multicast discovery within the e-DTS; Improved tracking and accuracy by utilizing other underlying API's; Implementing tracking systems utilizing other sensors and trackers and exploring the interaction between two different types of sensors; Incorporation of other types of trackers and sensors into the network.

VIII. CONCLUSION

This paper has presented the architecture of the next generation of a distributed tracking system, the e-DTS 2.0. The results of the experiments performed with the prototype of the e-DTS 2.0 indicate that it is possible to achieve an accurate distributed tracking system that uses the principles

of dynamic discovery and fusion in the context of inexpensive sensors such as web cameras.

REFERENCES

- [1] Girish G., Rajeev R., Mihran T., "Designing and Experimenting with a Distributed Tracking System." ICPADS 2008: 64-7.
- [2] Neha T., Rajeev R., Mihran T., "e-DTS Enhanced Distributed Tracking System," Purdue University, 2009.
- [3] Kato H., Billinghurst M. "Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System," In Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99). October, San Francisco, USA.
- [4] Oracle, "Discovery and Lookup Services", <http://download.oracle.com/javase/1.5.0/docs/guide/jmx/overview/lookup.html>
- [5] Edwards, K., Rodden, T., "Jini Example By Example," Prentice Hall PTR, 2001.
- [6] Juhasz, Z., Sipos, G., "JGrid: A Technology-Based Service Grid," ERCIM News Number 59, 2004.
- [7] Bogucki, R., "Augmented Reality Device For First Response Scenarios," University of New Hampshire, 2006.
- [8] Chaos Software Group, "Atomic Clock Sync v3.0," <http://www.worldtimeserver.com/atomic-clock/>, 2010.
- [9] Greyscale Automation Products, "Domain Time II Windows Time Agent Version 4.1," <http://www.greyscale.com/software/domaintime/instructions/misc/agent/>, 2010.
- [10] Microsoft Corporation, "Windows Time Service Technical Reference," <http://technet.microsoft.com/en-us/library/cc773061%28WS.10%29.aspx>, 2010.
- [11] Jean-Yves B., "Camera Calibration Toolbox for Matlab", http://www.vision.caltech.edu/bouguetj/calib_doc, 2010.
- [12] Dorner, R., "Accuracy in optical tracking with fiducial markers: an accuracy function for ARToolKit, Mixed and Augmented Reality," ISMAR 2004. Third IEEE and ACM International Symposium, 2004.
- [13] Malbezin, P.; Piekarski, W.; Thomas, B.; "Measuring ARToolKit Accuracy in Long Distance Tracking Experiments," In ART02, 1st International Augmented Reality Toolkit Workshop; 2002.
- [14] Claus, D.; Fitzgibbon, A.; "Reliable Fiducial Detection in Natural Scenes," Lecture Notes in Computer Science, 2004, Volume 3024, 2004.
- [15] Claus, D.; Fitzgibbon, A.; "Reliable Automatic Calibration of a Marker-Based Position Tracking System," Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1, 2005.
- [16] Nagpal, R.; Shrobe, H.; Bachrach, J.; "Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network," 2003.
- [17] Baroody, R.; Al-Holou, N.; Mohammad, U.; Lahdhiri, T.; Dayoub, I.; "Scalability in a Dynamic Discovery Service-based Jini for the Next Generation vehicle network," IEEE symposium on Computers and Communications ISCC 2009, 2009.
- [18] Chen, H.; Joshi, A.; Finin, T.; "Dynamic Service Discovery for Mobile Computing: Intelligent Agents Meet Jini in the Ether," Cluster Computing Volume 4, Number 4 343-354, 2001.