The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

Intelligent Distributed Architecture (IDA) For Mobile Sensor Data Fusion

John Meier and Burchan Bayazit
Department of Computer Science and Engineering
Washington University in Saint Louis
St. Louis, MO 63130 USA
{jlm4,bayazit}@cec.wustl.edu

Abstract—**In this paper, we provide a new two-layer scalable architecture, Intelligent Distributed Architecture (IDA). The first layer of IDA, the application layer, is responsible for the mission specific tasks. The second layer, the network layer, is responsible for relaying the information quickly to reduce latency. In order to increase efficiency, we propose an intelligent network layer that evaluates the network traffic through application provided policies. The evaluation assesses improved accuracy in bandwidth limited scenarios. IDA policies permit the application to drop, merge or modify the packets in real time from different sources. We demonstrate how this architecture can be applied to an example application, distributed target tracking. We also propose some new algorithms that can be used in conjunction with IDA for target tracking. Our experiments on this application show that IDA improves system performance when the bandwidth is limited.**

I. Introduction

In recent years, we observed a rapid increase in mobile wireless sensor platforms and it became increasingly important to share the data between those platforms, however limited bandwidth is still a challenge. In a multi-sensor environment, some of the shortcomings of bandwidth limitations can be addressed by increasing the processing power on the local nodes. In this case, the sensor nodes can identify the duplicate or irrelevant information by fusing the data from their neighbors and send only essential information. Sensor fusion algorithms suffer from scalability and other challenges related to wireless networks [1].

Typical distributed sensor fusion algorithms rely on a single layer approach. An application running on sensor nodes needs to identify not only the information from the host sensor but also recognize the data other sensors are sending so that it can fuse that data and share the sensor readings with other nodes. Furthermore, the application also needs to know the network status and usage so that it can capture and forward the data efficiently.

In this paper, we are proposing an Intelligent Distributed Architecture, IDA, which is a two layer architecture evaluated using a distributed sensor fusion application. The first layer of our architecture is the “application layer” which is executing the sensor application. Our second layer is the ”network layer” which is responsible for routing the sensor data to its destination after receiving directions (policy) from the application layer. This layer is also responsible for key elements of sensor fusion based on the distributed policies provided by application layer. While policy based approaches have been applied to wired networks through active networking hardware such as the CISCO AON [2], policies are usually provided by a central authority and uniformly distributed. In our approach, the applications provide the policies which increase the flexibility.

Our architecture has several advantages: (i) reduced bandwidth requirements enabled by policies to select key network packets to propagate, and (ii) reduced computation in the application layer caused by sending higher quality information distributed amoung selective nodes).

We have selected the distributed sensor fusion target tracking application to demonstrate the feasibility of our architecture. In our scenarios, the targets in the environment are monitored using either static or mobile sensors to collect and send their sensor measurements to other sensors. Our goal is to provide the most relevant spatial target track history measurements. IDA makes the decision locally to either drop or send the information. Upon receiving the sensor readings (measurement reports), our application layer assigns readings to existing target tracks based on the association metrics or else creates a new track. Once the assignment is finished, the application layer sends the report to the network layer for delivery. The network layer evaluates the information based on policies then compares the information received from other nodes to decide the next action. IDA policies guide the local decisions to send, drop or aggregate the sensor readings.

While our main contribution is the IDA architecture, we also provide new measurement algorithms, new methods to reduce the number of measurements using a new track to measurement method, less complex algorithms for distributed track fusion and reduction in processing by dynamic selection of simpler algorithms. Our experiments show that both our architecture and the new algorithms perform well, reducing the bandwidth required while increasing the accuracy.

Our extensive testing demonstrates that IDA improves the quality of the collected information and hence improves the accuracy when bandwidth is very limited.
In the next section, we discuss the related research, then we briefly define our problem in Section III. An overview of our architecture is presented in Section IV. In Sections V and VI, we discuss application and network layers for target tracking. We evaluate our architecture in Section VII and Section VIII concludes our paper.

II. Related Work

Distributed architectures improve performance for applications such as distributed sensor fusion ([1], [3], [4], [5]) using edge processing. While some of these algorithms (such as [5]) utilize policies fixed at the deployment, IDA provides flexible, application based policies configured at runtime Zhao et. al. [6], [7], [1] proposed an architecture

We express gratitude to Boeing for the support that aiding this research.

where policy is learned through belief networks. Sensor Information Networking Architecture (SINA) (Srisathapornphat's et. al. [8]) enables querying, monitoring, and tasking of sensor networks emphasizing organizing data but does not emphasize performance improvement. In contrast, we introduce a unique two layer approach that enables organizing and scheduling data using dynamic policy management that selects the appropriate algorithms to improve distributed fusion performance.

A common type of track engine used for report-to-track fusion is a 1-to-N tracker [9] recently selected for use with low cost automobile sensors. For example, [10] utilizes it for target-position estimation algorithm in collision avoidance, where the target data is collected and fused through multiple simple sensors. [3] presents an architecture to select distributed fusion algorithms in a static configuration. IDA differs from these approaches by providing a dynamic and flexible two layer approach as others have only single application layer that is responsible for both information processing and routing.

There are also algorithms developed for distributed mobile fusion using mobile sensors such as [3] and [11]. There are also fusion algorithms for wired networks. UCLA researchers [2] use a network infrastructure-supported selective data sharing and verification service mediator node that is able to process data at the edge using CISCO (Application Oriented Networking (AON) [2] products. IDA can be implemented using AON technology to accelerate performance.

III. PROBLEM DEFINITION

We have mobile wireless sensors and mobile targets whose movements we would like to follow accurately. Targets can enter or leave the sensing areas without coordination. They are heterogeneous with different kinematic characteristics and they may have different priorities (i.e., higher priority targets are more important). Each target is associated with a track that represents its historical movements. When a wireless sensor makes a measurement, that measurement report (MR) needs to be matched to an existing track or it will create a new track. Our goal is to provide timely and accurate track data to select nodes that enable local situational awareness (SA). Each sensor creates its own set of tracks and may share the track information with other nodes to improve the local fusion accuracy. Instead of sending all information to the selected nodes, we would like to send just the most valuable data (quality information) to avoid traffic congestion in limited bandwidth environment.

Figure 1 shows an example scenario with three sensors (N_1,N_2,N_3) and three targets (T_1, T_2, T_3). The circles around the sensors represent their sensory range and the sensors linked to each other with a straight line can communicate. Since both N_1 and N_3 can sense T_2, both would send the same track data. In that case N_2 would identify the redundancy and send only one copy.

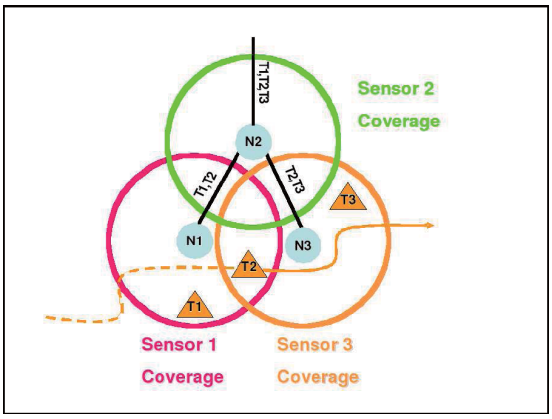

Fig. 1. Targets T1, T2 T3 are monitored by sensors N1 and N3. When duplicate information about T2 arrives to N2 from two different nodes, N2 propagates the information only once.

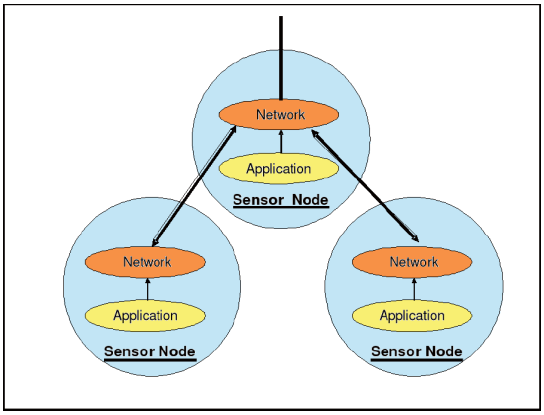

Fig. 2. Our distributed architecture uses two layers (application and network) to reduce the bandwidth and latency of handling the sensor measurements. The application sets policy to efficiently process measurements at the network layer.

IV. SYSTEM OVERVIEW

In this paper, we provide a new architecture, Intelligent Distributed Architecture (IDA). Our architecture consists of application and network layers.

In this approach, the application layer is responsible for the mission specific tasks, such as collecting sensor data and sending it to the correct destination. The destination can be a node with central authority or it can be just another network node. The application layer has very limited information of the network operations and information transmission occurring at the network layer. This abstraction provides the application developer an ability to concentrate on mission specific goals. The network layer in addition to being responsible for transmitting the application layers' data to its destination, also propagates data efficiently from other sensors to their final destination. While it is possible to send all the data to its destination, there could be several sensor nodes already sending nearly the same information resulting in reduced bandwidth. In order to increase efficiency, we propose an intelligent network layer that, based on a defined policy, evaluates the network traffic (packets) it is transmitting either from the application layer or

the neighboring sensor. This policy-based intelligent approach offers an opportunity to drop the duplicate packets as well as merge the information from different sources. Furthermore, if the application requires the data from other sensors, policy may dictate that information be sent to the application layer to provide policy. For example, the application layer policy specifies methods for aggregation of nearest node data to help increase the accuracy of the target location. The policy can also specify thresholds used to aggregate data.

In the following section, we will discuss how this architecture can be applied to target tracking. Briefly, our application layer is responsible for analyzing the sensor measurements, matching each reading to existing tracks and sending the updated track information to the network layer for delivery. Our network layer evaluates the new information through the policies provided by application layer. For example, if there is network congestion, it sends the higher priority information first.

The policy also enables the network layer to evaluate the necessity to send the received target information. If there were recent updates on a target, the network layer can drop older information in the queue related to that target.

V. APPLICATION LAYER

In target tracking, a common practice is to integrate many measurements from a sensor to determine where the target is actually located. This process is called track fusion, where each track represents the trajectory of a target. Hence our goal of correctly matching a sensor measurement report (MR) to a target is in fact matching a sensor reading to a track. The accuracy of the fusion process can be further improved using distribute track fusion adding MRs from other sensors [9].

Our application layer implements distributed track fusion using matching and sending mechanisms. First we match each sensor reading to a track. The matching algorithms are usually computationally expansive, so the matching is a two step process:(i) Candidacy identifies candidate readings for expansive evaluation, (ii) Association assigns reported target locations (measurement reports) to tracks. After the targets are matched, the measurement reports are prioritized and sent. These mechanisms are shown in Figure 3. Next, we will discuss the individual steps in detail.

A. MATCHING

The application layer continuously monitors the measurements to match them to existing targets based on spacial location. Candidacy (step 1) and Association (step 2) are all parts of the correct matching process to prioritize the measurement reports. In some cases, the targets may broadcast unique identification information that can be supplied by Interrogate Friend or Foe (IFF) information. However, in most scenarios,

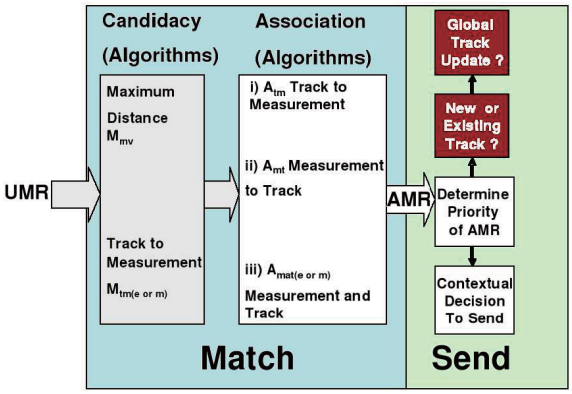

Fig. 3. Our distributed architecture (IDA) on sensor nodes intelligently matches and sends sensor measurements. Unassociated measurement reports (UMRs) are matched to objects being tracked to create associated measurement reports (AMR).

only limited information, such as the target location, is available. Hence we use the previous target locations (i.e., target tracks) to predict the next possible target location. If the Candidacy test is passed, we attempt to associate the new sensor measurements to known target tracks. Our matching mechanism includes the traditional processor intensive matching approaches in target tracking. At any given time the sensors maintain information on several targets, hence it is important to eliminate impossible matches. For example, if the sensors identify a target far away from a known track, there is no need to consider that target as a candidate for the current track. Once we identify the candidates, we then associate the MRs with the existing tracks. Please note that, it is possible that none of the candidates are good matches in which case the sensor reading actually corresponds to a new target.

1) Candidacy: The first step in the target matching process is identifying possible readings (MR) for the track association process. Candidacy is a coarse filtering process to improve the computational efficiency of the next step, Association, by eliminating out of bound readings. In order to identify candidates, we consider two tests, (i) maximum distance, a commonly used test, and (ii) track to measurement distance, a new test we developed.

Maximum distance tests sensor readings to determine if a target could reach that location given the current speed and velocity constraints, such as estimated maximum velocity. The spherical bound is based on the last known target location. The maximum velocity (V_{max}) is computed based on current velocity and last know position. The time since the last measurement is Δt. The maximum distance, M_{max}, is calculated using $M_{max} = V_{max}\Delta t$. All the readings that are closer than M_{max} to the last track position are accepted for Association.

The *track to measurement distance (*M_{tm}*)* predicts a target trajectory based on the track data and then tests if a sensor reading is near this trajectory. We use two different measurement methods; *track to measurement distance (*M_{tmE}*)* is based on Euclidian distance and *track to measurement*

distance (M_{tmM}*)* is based on Manhattan distance. For target location prediction, usually a Kalman Filter (KF) is used to smooth the data used to create the velocity prediction.

The KF can also be used to help predict the next target location. The error between the predicted estimated position and the position determined based on actual measurements is a variance that is fed back into the system to improve future predictions. Our KF weighting can be adjusted to rely more on the historical measurements rather than the current sensor measurements. Once the prediction is made, then sensor readings near the trajectory are accepted for further evaluation. Figure 4 depicts both approaches. In this figure any reading outside circle is not evaluated for a target as it would be impossible to reach. In the figure there is a target reading M and we want to match it to our predicted target position using two methods. M_{max} calculates a vector for the maximum distance the target could have moved then checks to see if the vector from the last known position to the measurement is less than the maximum distance vector or is within the sphere. M_{tm} calculates a threshold from the maximum distance to generate a sphere around the measurement. M_{tm} then determines if the predicted position of the track is inside the sphere. The Candidacy steps of matching are shown in the algorithm as steps 3-8. Both approaches have the ability use the distance metric of choice, such as Euclidean or Manhattan distances.

2) Association: Once we have candidate targets for a sensor reading, we evaluate each of the candidates to identify which target is the best match to that reading. Association of sensor readings to a target uses three different evaluation methods: (i) distance of measurement to a track (A_{tm}), (ii) statistical distance from measurement to a track prediction (A_{mt}) error, and, (iii) distance between statistical measurement and statistical tracking error (A_{mat}). A_{mat} is the most complex of the three and is the common method used.

This method calculates the value of χ^2 distance between the measurement variance and the track variance. A_{tm} is the least complex method and is easily computed. A_{mt} is in between the most and least complex methods. This Association test is similar to Malahanobis distance in the literature and is often used in sensor networks for data Association [10]. Our contribution improves the performance by modifying the Malahanobis distance formula.

B. Sending the Target Information

Once the target is identified, the application layer correctly assigns and updates the track information prior to sending it to remote nodes. A frequent update rate may reduce the bandwidth available for other nodes to send MRs, so instead of a continuous stream, our application layer prioritizes then sends the target information to the remote nodes in update intervals defined by the target properties. Latency of the MRs sent may also require re-prioritization of the measurements due to operations with the remote fusion application.

Algorithm 1 Matching (sensor readings, target tracks)

1: **for** each sensor reading M **do**
2: **for** each known target T **do**
3: $\hat{x}(k+1) = |V_Max * \Delta t|$
4: $MaxGateDistanceGD = |\hat{x}(k+1) - x(k)|$
5: ObservationDistance OD = $|x(k+1) - x(k)|$
6: **if** OD $\leq$ GD **then**
7: declare M a candidate for Association
8: **end if**
9: select the appropriate Association distance method(D)
10: calculate the threshold value (T) based on GD
11: calculate the Association distance $A = distance(M, T, D)$
12: create a table of tracks, prioritized associated measurements (A),required update rates
13: **if** distance $\geq$ previousDistance **then**
14: place current measurement ahead of the previous track measurement in queue
15: **end if**
16: **end for**
17: **end for**

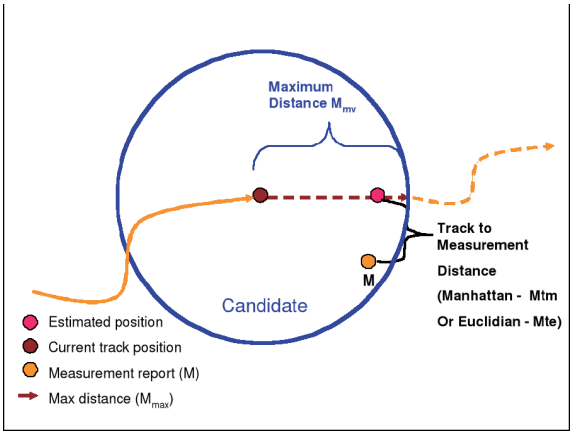

Fig. 4. Candidacy evaluation filters unassociated measurement reports (UMRs) using a maximum distance bound set by estimated maximum object velocity to select the measurements for Association. Measurements within the circle depicted are considered candidates for Association. We contrast this with our track to measurement distance based on the projected object location using Euclidian and Manhattan distance methods.

VI. Intelligent Network Layer:

The application layer creates policy that sets up the network layer to process the sensor measurements locally. The majority of the data is processed and filtered by applications to create a prioritized list of measurements to be sent. There are frequent variations in bandwidth that require decisions at the network layer to ensure valuable bandwidth is not wasted. Selecting the right measurements locally at the network layer is critical to achieving scalability and enhancing the quality of information. Communication between the application layer and the network layer can assure this occurs efficiently using policies. These policies can be scripts or binary plugins that are sent to program the network layer. This layer is responsible for matching and sending the data it received from the application layer or other nodes to its destination based on the policy distributed. As we are interested in target tracking, the policy dictates the decisions controlled by the application layer, i.e., it

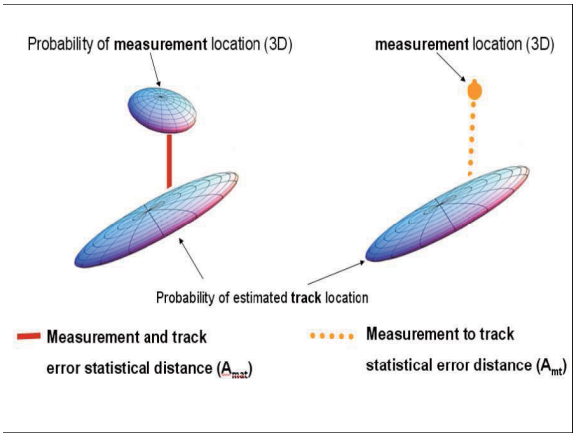

Fig. 5. Association of Candidate measurement reports with an object beingtracked traditionally use a measurement and track statistical error distance approach based on χ^2 distance depicted as the solid line. The measurement to track statistical error distance uses the our new method based on the Mahalanobis distance to reduce complexity. Measurements that are longer than the dotted line shown are not candidates. Note that this diagram is only a conceptual representation.

compares the current target information received from the local application layer and other nodes with the sensor measurements, matches the measurement to a track, evaluates the last time the information related to the specific target was sent, then makes a decision to transmit or drop it.

The two key decisions are matching and sending network data more intelligently..

- **Matching targets:** The task of identifying the same targets from different nodes requires accurate time synchronization of the measured data. Identifying targets based on very limited stored historical data is more difficult due to fewer available measurements. Similar to the application layer, we evaluate the same algorithms (A_{mat}, A_{mt}, and A_{tm}) for implementation at the network layer. We contrast commonly implemented methods that are more complex with using less complex methods for matching the measurements to the existing tracks at the network layer.
- **Sending the target information:** once the measurement is matched with the target, we then compare the last time the packet was transmitted related to this target's priority and make a decision to either transmit the data or drop it. A_{mat} and other distance algorithms measure how well the data fits the projected track. The strength of the measurement is evaluated by distance metrics. For example χ^2 distance computed for A_{mat} relates goodness of fit between the measurement and the projected track. A_{mt} designates how many sigma the projected track is from the measurement to indicate the likelihood of association. A_{tm} calculates the physical distance between the linear track projection and the measurement.

The network layer evaluates the Candidacy and Association based on the distance algorithm and threshold set by the application layer. Other key factors, such as complexity, must be carefully weighed by the application layer because they will affect the latency and accuracy.

VII. Experiments

Our experiments are designed to answer the following questions: (i) how successful is IDA at prioritizing and sending important information, (ii) how well are the matching algorithms working, and (iii) how do the number of sensors and mobility effect our architecture.

In order to answer those questions, we have designed two scenarios using IDA. Both scenarios have 20 randomly moving targets in the environment. The targets can have velocities upto 40kms. In the simple scenario, targets are allowed to move in a wider volume ($100 \times 100 \times 100 km^3$, see Figure 6(a)). In contrast, the complex scenario restricts the area to ($10 \times 10 \times 10$ km^3, see Figure 6(b)) causing more frequent track cross-overs making correct track associations very challenging. The sensors measure the distance (range, azimuth and elevation) to the object being tracked. The default sensor error settings are 4M in range, and 0.1 degrees in azimuth and elevation. In select experiments, we varied the channel capacity (bandwidth), and update rates. Three sets of experiments (single sensor, multiple sensors and mobile sensors) evaluate the performance of IDA. In all of our experiments we evaluate *the inaccuracy*, i.e., *error in predicting a target's location at a single node*. This error happens when a destination node does not have enough measurements (packets) arriving for a given target. In a preferred architecture, the prediction error should be affected less as the bandwidth is reduced when intelligent algorithms select the information to be transmitted. Traditional solutions increase the update rate as priority increases. The update rate represents the number of messages each sensor has to send per second for each target it senses. We evaluate the effectiveness of this approach by varying the update rate for twenty targets.

a. Evaluation of IDA on A Single Sensor

In this experiment, our goal is to evaluate the performance of our system without exchanging information from other sensors. There is only one sensor that is collecting and sending the target data to a destination. We vary the bandwidth in the system by changing the number of messages that can go through a link. We used network simulator tool, NS-2, to evaluate bandwidth usage. Our preliminary observation indicated that less bandwidth (below 7 reports/sec) prevents sending adequate meaningful information since most packets will be dropped. More bandwidth (more than 13) allows a majority of the packets to be sent and all algorithms are shown to perform similarly, thus we have selected the values between 7 to 13 to describe the performance enhancements of IDA. In order to evaluate IDA, we compared different combinations of Candidacy and Association algorithms using the used by the Boeing Distributed Mobile Fusion Toolkit (BDMFT). As we are interested in implementing our architecture on a FPGA, we are interested in simpler computations, such as using Manhattan distance as opposed to Euclidean or χ^2). So, in our experiments, we also compare the performance of Manhattan and Euclidean distance metrics when they are used in Candidacy and Association algorithms.

(a) (b)

Fig. 6. Simulation scenarios with 20 randomly moving targets. The closer the targets, the harder it is to correctly identify them. (a) Simple Scenario with 100×100KM3 area. (b) Complex Scenario with 10×10×10KM3 area..

The prediction accuracy also depends on the frequency of the updates (update rate). For example, more frequent updates with large bandwidth should provide increased accuracy. Our experiments are designed to evaluate less frequent updates with lower bandwidth to determine the affects of IDA selecting different algorithms to improve accuracy. Hence, we also investigate the affects of update rates ranging from 5 to 15 messages per second. In this paper, we will show the results for a simple environment with update rates of 10 and 15 reports/second, and complex environment with an update rate of 10 reports/second. The rest of our results can be obtained by contacting the authors.

Figures 7 show the results of our experiments using the simple environment (scenario) with the update rates 15 and 10. In these figures, x-axis represents the channel capacity (bandwidth), where as the logarithmic y axis represents the error in the prediction of target location.

In Figure 7 the update rate is 15 track reports per second for each target. The update rate of 15 is chosen to examine the fusion algorithm performance with reduced flexibility to choose the best measurements. In each sub-figure in the figure show a different Candidacy algorithm: (a) "Maximum Distance", (b)"Distance to Track" with Manhattan, and (c) "Distance to Track" with distance metric.

Each sub-figure contains the performance values for different Association algorithms that are applied after the designated Candidacy method was used. In the figures, we show the error in the prediction if all the messages were transmitted regardless of the bandwidth ("All sent") to create the lower bound. We also create an upper bound by including a measurement indicating if the messages were to be dropped randomly ("random"). The lower bound ("All sent") represent the best possible prediction, where as the upper bound ("Random") is always the worst case because it does not use intelligent selection.

Our results show that IDA performed far better than "random" in all scenarios. All algorithm results are bounded by the "all sent" and "random" bars on the graph shown in Figure 7. Our track to measurement (M_{tm}) algorithm reduces the number of measurements to associate by over 55% per track in the complex scenario, which results in decreased processing. Our new Association algorithm also decreases the miss-association errors for measurements with the same error to reduce processing.

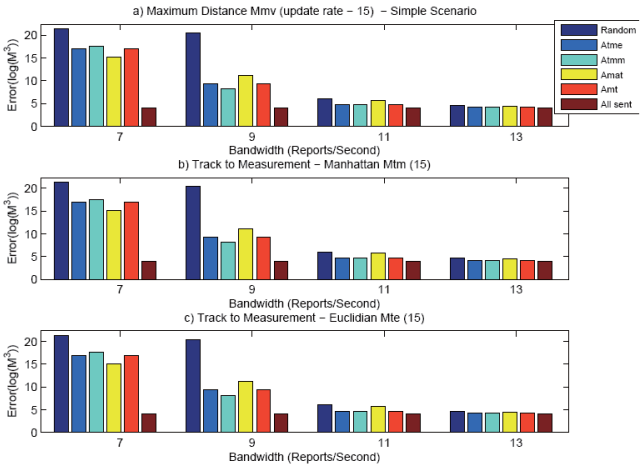

Fig. 7. Simple Environment with update rate 15. Bandwidth vs. prediction error for three Candidacy algorithms: (a) Maximum Distance, (b) track to measurement (Manhattan) (c) track to measurement (Euclidian)). Each bar group represent different Association algorithm used (from left to right: "Random drop",Atme,Atmm,Amat,Amt,"All messages sent")

We varied the bandwidth (channel capacity) with each of the four Association methods using different distance meth ods that include χ^2, Manhattan, Euclidian, and Mahalanobis. The Association track to measurement (A_{tmm}) algorithm using the Manhattan distance actually outperformed, or was more accurate, than other more complex methods as shown in Figure 7. In these figures, the performance of random selection was less accurate using low bandwidth.

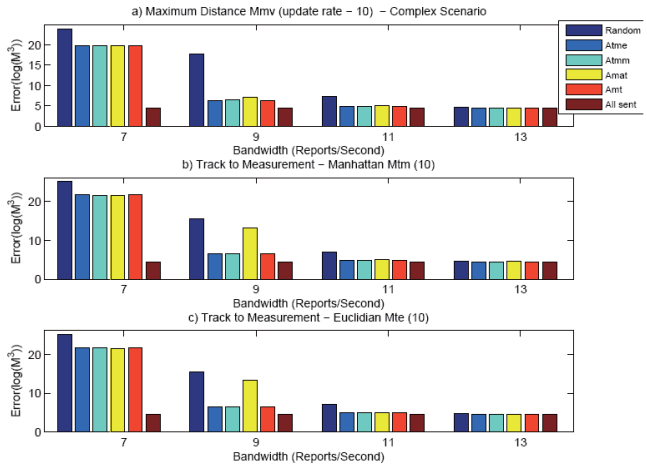

Fig. 8 Complex Environment with update rate 10. Bandwidth vs. prediction error for three Candidacy algorithms: (a) Maximum Distance, (b) track to measurement (Manhattan) (c) track to measurement (Euclidian)). See Figure 7 for the legend.

Due to space limitations, we only show the results for the complex environment with an update rate of 10 in Figure 8. These results are very similar to the simple scenario but overall show the accuracy decreasing due to mismatching of sensor readings. Frequent crossing of tracks causes miss-association, resulting in poor track prediction.

b. *Multiple Sensors*

Next, we investigated the affects of multiple sensors. In this experiment, we have two sensors sending data to a common node. In our previous experiment, we had found that the "distance to track" with Manhattan distance metric gave the best results. So, for our multiple sensor experiments, IDA will select this Candidacy algorithm. We also evaluate IDA with different association algorithms and compare them to the tradition approach. Further more, we have selected an update rate of 10 track reports per second. We first set both sensors to have the same accuracy error. Figure 9 (a) shows the results in simple environment. Next, we have increased one of the sensor errors by a factor of 10, to 400 meters. In this case, the track to measurement Association (A_{tm}) algorithm using the Mahalanobis distance performed significantly better (see Figure 9 (b)) for sensors with different errors. We modified our A_{mt} algorithm to use the position of the predicted track measurement with the statistical variance of the track measurement and achieved over 25 takes into account the statistical variance of the data while the simpler methods (A_{tmm} and A_{tme}) do not.

c. *Moving Sensors*

In our last experiment, we have two sensor moving randomly in the simple environment with a speed of 60 kmph. We set the accuracy of both sensors to 40 meters, kept the speed constant, and varied the 20 randomly moving targets. The results were very similar to static sensors and therefore show one example result in Figure 9 (c), where "Maximum Distance" Candidacy algorithm are used with different Association algorithms.

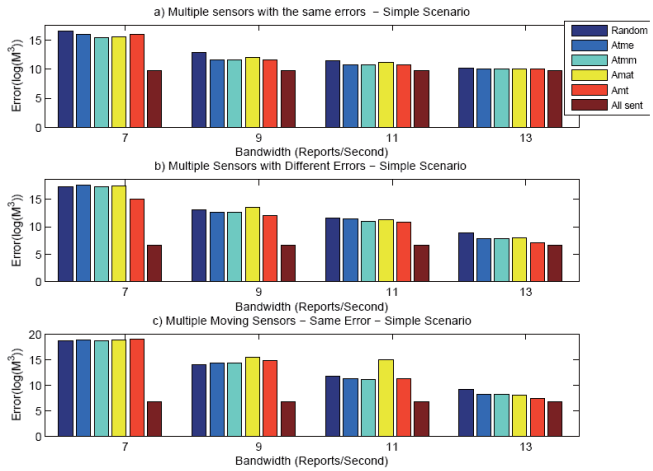

Fig. 9 Simple Environment, update rate 10, multiple and mobile sensors: (a) Multiple sensors with equal error using the simple scenario show no significant changes in accuracy. (b)Multiple sensors with different sensor errors. (c) Mobile sensors. See Figure 7 for the legend.

VIII. CONCLUSION AND FUTURE WORK

In this paper, we have presented a two-layer architecture, Intelligent Distributed Architecture (IDA), that is able to improve the performance of many network enabled distributed applications. The application layer is responsible for collecting the application specific information while the network layer is responsible for transmitting that information.

A policy, provided by the application layer, dictates how the information will be transmitted. We studied our architecture using a target tracking scenario with traditional and new algorithms. Our experiments show that IDA is able to identify the quality information to exchange over low bandwidth and help improve accuracy by reducing error. This results in improved accuracy while reducing the required bandwidth. We also showed that less complex algorithms for distributed track fusion exceeded existing approaches for select situations.

The network hardware and software approach is able to apply the most efficient algorithms as conditions change. We plan to further investigate the complex scenarios using additional moving sensors to assess the local decisions made by IDA. Our future goal is to create dynamic policies that will be distributed via mobile agents to the network layer. The dynamic policies will be assessed using dynamic inference engines or solvers to enable more intelligent local decision making. Our future research will enhance the intelligent layers of IDA.

REFERENCES

[1] F. Zhao, Shin, and Reich, "Information-driven dynamic sensor collaboration
for tracking applications," in *IEEE Signal Processing Magazine*, 2002, pp. 1–9.

[2] A. Parker and et. al., "Network system challenges in selective sharing and verification for personal, social, and urban-scale sensing applications," *ACM SIGCOMM, 5th Workshop on Hot Topics in Networks*, p. 41, 2006.

[3] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, p. 13, 2006.

[4] P. V. Mark Alford, "A layered architecture for multisensor data fusion systems," in *Signals, Systems and Computers Conference*, Mar. 1999, pp. 416–419.

[5] D. Tennenhouse and D. Wetherall, "Toward an active network architecture,"
in *Computer Communication Review*, Mar. 1996, pp. 1–14.

[6] F. Zhao, J. Liu, L. Guibas, and J. Reich, "Collaborative signal and information processing: An information directed approach ," *Proceedings of the IEEE*, vol. Volume 91, no. no. 8, pp. 1–10, Aug. 2003.

[7] M. Chu, H. Haussecker, and F. Zhao, "Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks," in *Xerox Palo Alto Research Center Technical Report*, 2001, pp. 1–7.

[8] C. Srisathapornphat, C. Jaikaeo, and C.-C. Shen, "Sensor information networking architecture and applications," *IEEE Wireless Communications*, vol. Volume 8, no. Issue 4, pp. 52–59, Aug. 2001.

[9] S. S. Blackman, "Multiple-target tracking with radar applications," in *Artech House*, 1986, pp. 357–395.

[10] F. Flster and H. Rohling, "Data Association and Tracking for Automotive
Radar Networks," *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, vol. Volume 6, no. number 4, pp. 370–378, Dec. 2005.

[11] A. Tchamova, J. Dezert, T. Semerdjiev, and P. Konstantinova, "Target tracking with generalized data assocoation based on the general dsm rule of combinaton," in *Proceedings of Fusion 2004*, Stockholm, Sweden.