Meteorological Command and Control:

An End-to-end Architecture for a Hazardous Weather Detection Sensor Network

Michael Zink, David Westbrook, Sherief Abdallah, Bryan Horling, Vijay Lakamraju, Eric Lyons, Victoria Manfredi, Jim Kurose
Dept. of Computer Science, and
Center for Collaborative Adaptive Sensing of the Atmosphere
University of Massachusetts Amherst, MA 01003

Kurt Hondl
National Severe Storms Laboratory
National Oceanic and Atmospheric Administration
Norman, OK 73019, USA

Abstract— We overview the software architecture for a network of low-powered radars (sensors) that collaboratively and adaptively sense the lowest few kilometers of the earth's atmosphere. We focus on the system's main control loop – ingesting data from remote radars, identifying meteorological features in this data, and determining each radar's future scan strategy based on detected features and end-user requirements. Our initial benchmarks show that that these components generally have sub-second execution times, making them well-suited for our NetRad system.

I. INTRODUCTION

Distributed Adaptive Collaborative Sensing (DCAS) of the atmosphere is a new paradigm for detecting and predicting hazardous weather using a dense network of low-powered radars to sense the lowest few kilometers of the earth's atmosphere [McLaughlin 2005]. *Distributed* refers to the use of large numbers of small radars, spaced close enough to "see" close to the ground in spite of the Earth's curvature and avoid resolution degradation caused by radar beam spreading. *Collaborative* operation refers to the coordination (when advantageous) of the beams from multiple radars to view the same region in space, thus achieving greater sensitivity, precision, and resolution than possible with a single radar. *Adaptive* refers to the ability of these radars and their associated computing and communications infrastructure to dynamically reconfigure in response to changing weather conditions and end-user needs. The principal components of the DCAS system include sensors (radars); meteorological algorithms that detect, track, and predict hazards; interfaces that enable end-users to access the system; storage; and an underlying substrate of distributed computation that dynamically processes sensed data and manages system resources.

NetRad is a prototype DCAS system whose goal is to detect a tornado within 60 seconds of formation and to track its centroid with a temporal error no greater than 60 seconds. At the heart (or perhaps more appropriately, the "brains") of NetRad is its Meteorological Command and Control (MC&C) component that performs the system's main control loop – ingesting data from remote radars, identifying meteorological features in this data, reporting features to end-users, and determining each radar's future scan strategy based on detected features and end-user requirements. In this sense, NetRad is truly an end-end system, from the sensing radars through the computing and communication infrastructure and algorithms, to the end users.

In this paper, we describe the software architecture of NetRad's MC&C and present initial benchmarks of its computational/communication requirements and performance. The important components of the MC&C that we study in the paper are *(i)* the data ingest, field retrieval, and meteorological detection algorithms, *(ii)* a feature repository that maintains a multi-level grid of feature values with associated user-based utilities, and that generates new sensing tasks for the networked radars, and *(iii)* a resource allocation/optimization process that determines the radars' scan strategy for the next system heartbeat. We discuss event notification mechanisms, and the computation of user-based utilities for competing sensing requests. We also discuss how NetRad timing considerations are addressed by structuring the feature repository as a blackboard system that temporally decouples data ingest/processing from the generation/optimization of future sensing activity. Our initial benchmarks, obtained by evaluating NetRad components using reflectivity and wind velocity data from the NOAA NEXRAD WSR88D system [NOAA 2005] show that that these components generally have sub-second execution times, making them well suited for use in the NetRad system.

The remainder of this paper is structured as follows. In the following section we briefly describe the overall NetRad system, and then dive down into the details of NetRad's MC&C architecture. In section 3, we present benchmark execution times for various MC&C components. Section 4 summarizes this paper and discusses our future research efforts.

II. NETRAD SYSTEM OVERVIEW AND THE MC&C ARCHITECTURE.

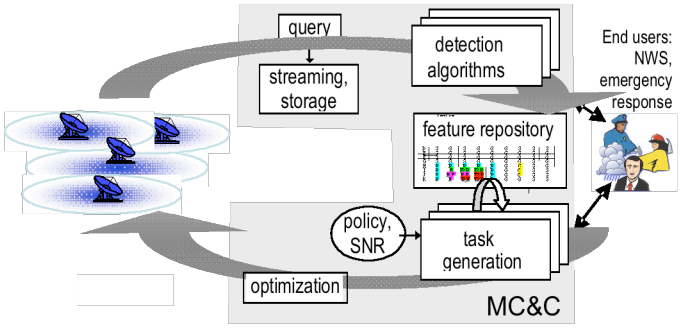

Figure 1: MC&C components

We are currently building a NetRad prototype system to be deployed in southwestern Oklahoma, consisting of four mechanically scanned X-band radars atop small towers, and a central control site (later to be decentralized as the number of radars increases) known as the System Operations and Control Center (SOCC). The SOCC consists of a cluster of commodity processors and storage on which the MC&C components execute.

NetRad radars are spaced approximately 30 km apart from each other and together scan an area of 80km x 80km and up to3 km in height. Each radar is tasked to scan an angular sector of up to 360 degrees in 1-degree increments, with a range gate (radial voxel) size of 100 meters out to 30 km. With two elevation scans during each tasking, each radar can thus produce up to 360*300*2 = 216K reflectivity and velocity values each time it is tasked. While existing meteorological radar systems such as NEXRAD generally operate in "sit and spin" mode (taking full 360-degree volume scans independently of location and type of meteorological features present), NetRad radars are tasked by the MC&C to focus on volumes of high interest to end-users, as discussed below.

Each radar consists of three subsystems:

- The **Rotating Tower Top** houses the radar unit and an embedded system that monitors the radar's operational parameters and enables operator actions in the case of anomalies (e.g., mechanical problems).
- The **Non-rotating Tower Top Subsystem** is located below the rotating joint and houses a data acquisition board (based on Field Programmable Gate Array (FPGA) technology), a radar controller, and a Gigabit Ethernet switch. The FPGA processes raw digitized data (at a rate of approximately 100 Mbps) into packets that are sent via the switch over a fiber optic cable to the Tower Base Subsystem. The radar controller controls the movement of the radar pedestal.
- The **Tower Base Subsystem** consists of a compute cluster (currently just a single node) and an IDE RAID storage system, connected via a Gigabit Ethernet switch to a router. The tower base takes raw radar data, computes so-called moment data (essentially an average of multiple radar-pulse measurements for a given voxel of space), while performing quality control (e.g., attenuation correction and range folding) on this data. The 1 Mbps moment data is sent to the SOCC over OneNet [OneNet 2005], an IP network operated by the Oklahoma state regents, which is configured to provide 4 Mbps connectivity from each radar to the SOCC. Raw data is archived at the tower base storage and can be transferred to SOCC storage in the background.

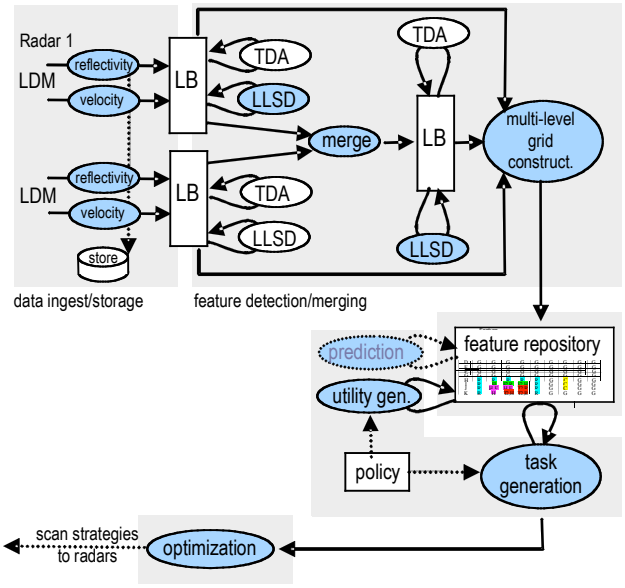

Figure 2: MC&C details and measurement points

The SOCC is a centralized compute cluster (later to be decentralized) interconnected via a Gigabit switch, on which the MC&C algorithms execute. As shown in Figures 1 and 2, the SOCC has five main components *(i)* data ingest and storage, *(ii)* meteorological feature detection and multi-radar merging, *(iii)* feature repository, *(iv)* utility and task generation, and *(v)* optimization. We describe each of these functions below in more detail, roughly following the path taken by radar data through the MC&C, and the resulting re-tasking of the radars.

A. Data Ingest and Storage

One SOCC computer is responsible for data ingest, archiving and distribution. Here, moment data (as well as the higher rate raw data, which is transferred at low priority) is streamed from the sensor nodes to the MC&C detection algorithms, and written to storage. In the future, both moment and raw data will be available to end-users via a query interface.

Data from each elevation scan is sent from a remote radar to the MC&C data ingest routines using LDM [LDM 2005], client/server middleware that reliably transfers radar data over a TCP connection. Given the pre-provisioning of OneNet bandwidth for NetRad use, congestion loss is not a concern in our initial testbed. However, we are currently developing a UDP-based transport protocol that uses application-specific selective dropping for congestion-control in bandwidth-constrained environments [Banka 2005]. As illustrated in Figure 2, the per-radar received reflectivity and wind velocity data for an elevation scan are converted to NetCDF format, and stored in a file. An event is then posted and distributed among

the MC&C procedures using the Linear Buffer (LB) pub/sub construct of the WDSSII software [Hondl 2003]. In section 3, we report the time needed to transfer an elevation scan from sender to receiver over a Gigabit LAN switch using LDM, write the associated files and post/distribute an event.

B. *Meteorological Feature Detection, Multi-radar Data Merging*

Once the notifications of available per-elevation reflectivity and wind velocity data have been posted, various meteorological detection algorithms can then read this data and perform feature detection. Figure 2 shows two such algorithms: *LLSD* (for which we provide per-elevation execution times in Section 3) computes wind shear and rotational divergence; *TDA* performs tornado vortex detection. Other per-radar detection algorithms can be easily "plugged in" using the LB pub/sub mechanism. Once a detection algorithm completes its execution, notification is provided through the LB. The *merge* procedure converts polar coordinate data to latitude/longitude coordinates, and fuses together spatially overlapping data from multiple radars. We benchmark the performance of the *merge* routine in Section 3.

C. *The Feature Repository*

NetRad system is a "real-time" system in the sense that radars must be re-tasked by the MC&C every 30 seconds – the system "heartbeat" interval. This heartbeat interval was chosen based on the physical properties of the mechanically-scanned radars, the timescale over which atmospheric conditions change, and the system goal of detecting and tracking tornados within 60 seconds. A notion of heartbeat also allows the radars to easily synchronize their operation (e.g., having overlapping radars scan the same volume in order to perform 3D wind retrieval), and also helps simplify the optimization of radar targeting. As we evolve from mechanically-scanned radars to rapidly reconfigurable solid-sate radars, we expect to relax the notion of a system heartbeat.

As discussed above, radars are retasked based on detected meteorological features and the projected future evolution of these features. In order to decrease the timing dependencies between the ingest/processing of radar data and the generation of radar commands, the MC&C adopts a blackboard-like architecture [Jaganathan 1989]. At the heart of the MC&C is the feature-repository, a multi-level grid that stores both the underlying per-voxel reflectivity and wind velocity data, as well as higher-level spatially-coherent meteorological "objects" such as storms cells, areas of high wind shear or precipitation, and tornados. Each object also has a position, a spatial extent for non-point objects, and a tag representing the meteorological phenomenon that the object represents (e.g., storm-cell, mesocyclone, and tornado). The multi-level grid-construction procedure writes this information into the feature repository as needed data becomes available via the linear buffer, as shown in Figure 2.

The generation of radar commands (the lower half of the control loop in Figures 1 and 2) proceeds *asynchronously* from the input processing of data (the upper half of the control loop).

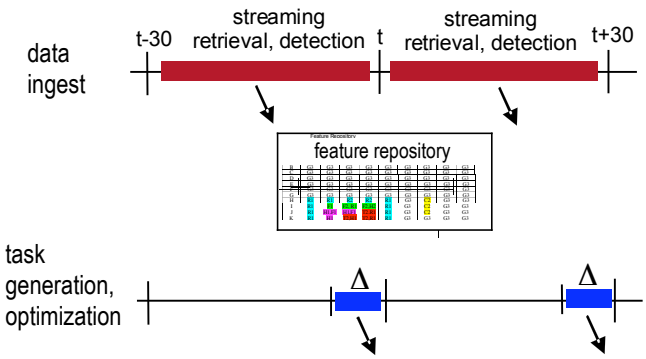

Figure 3: Feature repository: decoupling data ingest processing from periodic generation of radar commands

In this decoupled architecture, detection algorithms continuously post their results to the feature repository. As shown in Figure 3, at 30 second intervals the task generation component posts a set of tasks based on current state of the feature repository, and the optimization component then processes this task set and generates a scan strategy for the radars for the next 30 second cycle. In this design, we have relieved the time pressure on the detection components and somewhat relieved the time pressure on the MC&C components, task generation and optimization. One consequence of this design is that data that is not processed and posted on the feature repository before the task generation begins will not be acted upon until the next cycle of the system. This allows the system to avoid stalling, while waiting for late-arriving data (e.g., due to unanticipated network and processing delays). We are interested in the effects of this "decision lag" and also its relationship to the selection of the value of the system heartbeat.

Figure 3 illustrates that the data-driven streaming retrieval and detection algorithms write the results of their execution into the feature repository. Starting Δ time units (where Δ is the execution time of the task generation and optimization algorithms) before the radars are to be re-targeted, the task generation process executes, followed by the optimization process. These processes may use all data available (in both the current time step, and previous time steps) in their computations. Once the optimization process has completed, the radars are then re-targeted for the next 30 second cycle.

We note here that the feature repository is the central system "data structure." It is from here that meteorological objects can be obtained and subsequently delivered/displayed to end users. It is here that assimilated exogenous data (e.g., from satellite or from NEXRAD) can be stored and merged with NetRad-generated data.

D. *Utility, Prediction, and Task Generation*

Within the feature repository, each voxel and each object has an associated utility that represents the "value" of scanning that voxel/object during in the next heartbeat. The utility value weights considerations such as the time since the voxel/object was last scanned, the object type (e.g., scanning an area with a tornado vortex will have higher utility than sensing clear air),

and user-based considerations such as the distance from a population center (e.g., among two objects with identical features, the one closer to a population center will have higher utility).

We are currently developing a predicting component (shown with dashed lines in Figure 2) that tracks meteorological phenomena (e.g., a storm's centroid) and predicts their future locations. New objects, corresponding to the predicted future locations of the phenomena, can then be added into the feature repository – allowing these predicting modules to be easily integrated into the current architecture. In section 3, we present measured execution times of several different algorithms for storm-cell centroid tracking.

The MC&C task-generation component takes objects from the feature repository and produces tasks, with an associated utility, for the optimization component. We use K-means clustering [Jain 1999] to generate the tasks. The initial centroids of the clusters are chosen by sorting the objects by utility and using the K spatially-separated objects with the highest utility as our starting points. This simple pre-clustering step is designed to ensure good spatial coverage for our clusters. The K-means distance metric uses a 4-dimensional vector of parameters: an objects X, Y position, utility, and meteorological type. The relative weighting of these parameters can be adjusted to give differing emphasis to each parameter. After clustering is complete, a final filtering step removes tasks with utility below a given threshold.

E. Optimization

The input to the radar targeting components is a list of objects that can potentially be scanned and their associated utility. The optimization module determines, for each radar, the angular sector to be scanned (targeted) by that radar for the next 30-second cycle. The overall utility of a given configuration of the radars, will depend not only on the utility of the objects scanned, but also the size of the sector (since larger sectors imply less time spent sensing a given radial) and the number of radars targeting a given volume (since more radars illuminating a volume implies higher accuracy of measurement). For our first testbed, a simple "brute-force" approach towards optimization is used that enumerates all possible configurations and computes the associated overall utility. In section 3, we present measured execution times for this approach. Other, more scalable, approaches towards optimization are currently being investigated.

III. BENCHMARKING OF INDIVIDUAL MC&C COMPONENTS

In this section, we present empirical measurements of the execution times of various NetRad MC&C components highlighted in Figure 2. All measurements were performed on a PC (3.2 GHz Intel CPU, 1 GB RAM) running Linux (RedHat 2).

NetRad MC&C component execution times will depend on the radar data ingested by the system. Since, the NetRad system is not completely build or deployed, we use existing NEXRAD [NOAA 2005] radar data and other sources, as described below, as input to the NetRad MC&C components. Recall that NEXRAD radars operate in "sit and spin" mode and thus the data ingested in one time period has no influence on the radars' scan strategy in a subsequent time period. Before presenting component runtimes, let us describe the radar data inputs used. Unless otherwise noted, we use the following six NEXRAD radar data sets, which provide per-radial range-gate reflectivity and wind velocity data. The data sets can be obtained from [CASA NEXRAD-data 2005]. The data in cases 1-4 are generated by a single radar; in cases 5 and 6 data comes from two partially overlapping radars. Each data set consists of sets of elevation scans, with a set of scans taken every five minutes. These sets of elevation scans are the input to our MC&C algorithms.

Test Case	Location	Date of event	Description of Events	Single/Multi Radar
1	KLCH Lake Charles LA	11/02 2004	Late season thunderstorm activity with scattered weak mesocyclones, tornadoes, waterspouts.	Single
2	KTLX Tulsa OK	5/03 1999	Extreme supercell outbreak including an F-5 tornado, costliest tornado in history.	Single
3	KFSD Sioux Falls SD	5/30-31 1998	Tornado outbreak with a number of vortices in close proximity to each other.	Single
4	KMLB Melbourne FL	9/05 2004	Hurricane Frances as it approaches Atlantic Coast of Florida at Category 2/3 with large, well defined eye and intense banding.	Single
5	KDDC Dodge City KS	6/06 2004	A strong bow-echo sweeping across the state of Kansas.	Multi
6	KICT Wichita KS	6/06 2004	A view of the aforementioned bow-echo case from a radar located further from the event.	Multi

Table 1: NEXRAD input data sets

A. Data Transmission, Storage, Event Posting

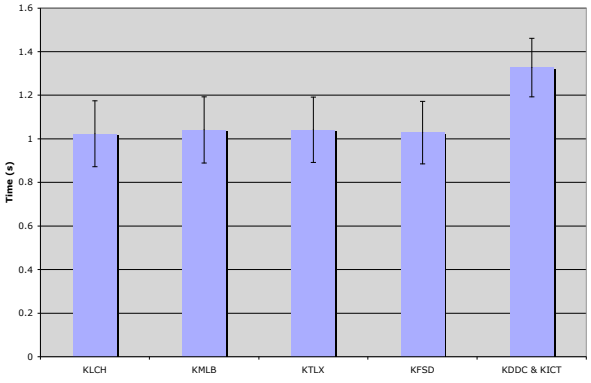

Figure 4: LDM to LDM, NetCDF file creation, event posting

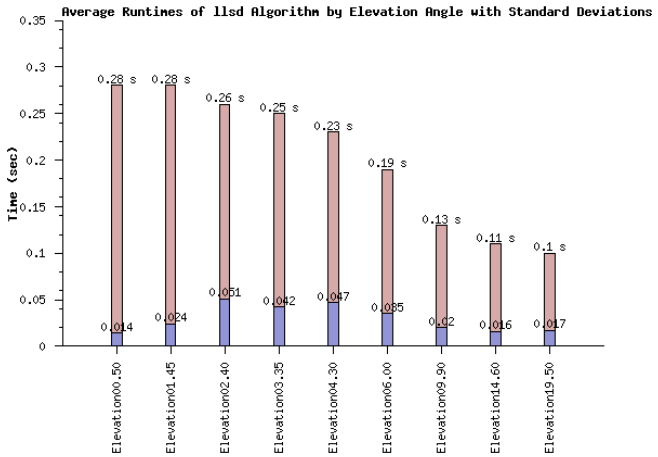

Figure 5: LLSD Run Time Per-Elevation Scan

We begin our benchmarking of the MC&C by considering the time needed to transfer radar data from a remote radar to the SOCC, store the per-elevation data in NetCDF format at the SOCC and post a notification event on the LB. Our measurements show that approximately one second is needed to compress and transfer a per-elevation scan from radar to SOCC over a Gigabit switch. We note that the nominal media transmission time (transmitting 20 KB of data into a Gigabit link) would increase from .00001 sec to .01sec if the link speed were changed to 1 Mbps. Thus, computing times rather than network media transmission times are the dominant factor here. We also note that these run times are for TCP transfers. We are currently developing a radar transport protocol that uses recent throughput measurements to avoid a slow-start phase [Schmitt 2002], and an application-specific congestion control/packet drop protocol [Banka 2005], as discussed in section 3. Once data arrives at the SOCC, reflectivity and velocity data files are created and an event is posted. Figure 4 shows that less than a second of runtime is needed to perform these operations.

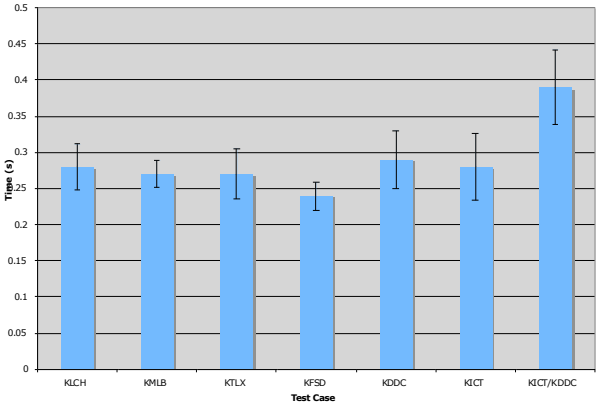

Figure 6: *merge* runtime, single and multi -radar cases

B. Detection Algorithms

Figure 5 shows the per-elevation scan runtime needed by the LLSD algorithm. The standard deviation of the runtime is shown by the lower-valued super-imposed bar. LLSD requires less than 0.3 seconds per elevation scan on average, with the runtime decreasing with an increasing elevation angle. This behavior results from the fact that radar beams e aimed higher in the atmosphere, see decreased meteorological activity.

Figure 6 shows the *merge* runtime for the 6 test cases. We took measurements for all six single radar cases and the additional case where the data for KICT and KDDC is merged. These radars have an overlap of roughly 50% percent. The results show that the runtime increases by approximately 30% with this amount of overlap. The difference between the single radar and the multiple radar case (KICT/KDDC) is that both coordinate conversion and data fusion must be performed for the overlapping region. Again, we see sub-second runtimes for this component.

C. Task Generation, Prediction

Recall from our earlier discussion, that the input to the task and utility generation routines is a multi-level grid of lat/long reflectivity and wind velocity values, with higher level "objects" superimposed on this grid. In order to use NEXRAD data in a NetRad MC&C environment, we assume that the NEXRAD data covers an 80km x 80km area.

Two different timing values are collected. The first is the run time of the K-Means algorithm, the second is the time taken posting features to the feature repository. Additional data collected includes the number of points being clustered and the number of iterations required before the K-Means stabilizes on a set of clusters. Figure 7 also shows the runtime for hypothetical scenarios for coarser and finer grid sizes.

We have also investigated the computational requirements of three approaches towards storm cell tracking: the WDSS II SCIT algorithm [Johnson 1998], a simple Kalman filtering algorithm, and a switched Kalman filtering algorithm A comparison of the tracking performance of these three algorithms is beyond the scope of this paper; see [Manfredi 2005] for details. Here, we note each algorithm requires less than 30 msec of execution time to perform one-step prediction, over 35 storm cell centroid tracks provided by NSSL.

D. Radar Scanning Optimization

The final step in closing the control loop is for the optimization module to determine the sectors to be scanned by the radars. The execution time of the optimization algorithm will depend on the number of radars, the extent to which the radars overlap, and the number/location/size of the meteorological objects in the radars' field.

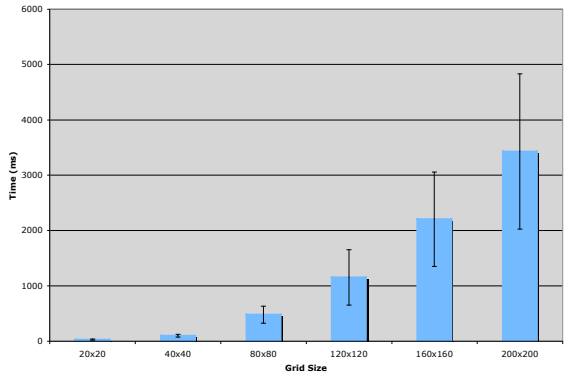

Figure 7: Task generation processing time

Figure 8 plots the average run time and standard deviation for the optimization module under several different scenarios, using the KFSD data. Along the x-axis we vary the number of radars symmetrically placed in the 80x80 grid. We control the amount of overlap of the radars' footprint by changing the radius of each radar's circular footprint. Three runtime curves are plotted for the case of zero overlap (the edges of the radars' footprint are coincident but non-overlapping), 33% overlap, and 67% overlap. For the case of 4 radars placed on 80km x 80km grid with 33% overlap, we see that the expected runtime is approximately 30 ms.

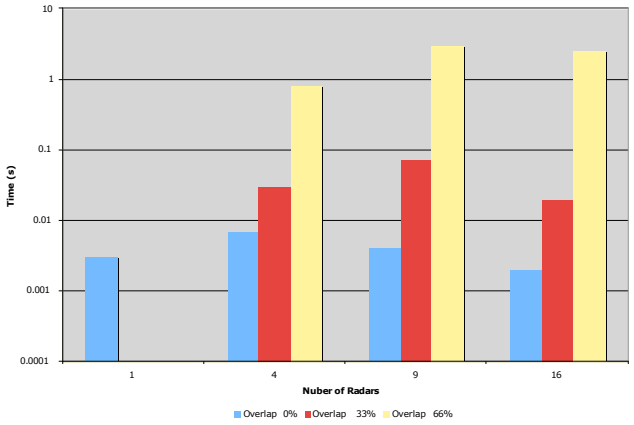

Figure 8: Optimization module runtime

IV. Conclusions and Outlook

In this paper, we have described the software architecture of the meteorological command and control (MC&C) component of a NetRad prototype system currently under development – a dense network of low-powered radars that collaboratively and adaptively sense the lowest few kilometers of the earth's atmosphere The MC&C performs the system's main control loop – ingesting data from remote radars, identifying meteorological features in this data, reporting features to end-users, and determining each radar's future scan strategy based on detected features and end-user requirements. Our initial benchmarks, obtained by evaluating NetRad components using NEXRAD data show that that these components generally have sub-second execution times, making them well-suited for use in the NetRad system. Our future research will include the deployment and demonstration of the NetRad system, and continued enhancement of the detection, tracking, and optimization components. Thus far we have focused on a centralized view of the command and control architecture, which is appropriate given the relatively small number of radars in the initial testbed. However, several factors make control more difficult as the network scales, and preclude a purely centralized architecture. We are thus currently investigating distributed MC&C architectures that take advantage of the limited coupling among radars over large geographic areas.

V. Acknowledgements

We are grateful to Mark Simms for supplying scripts for processing WDSS II data, and Jerry Brotzge for many insightful conversations regarding the MC&C. This work was supported in part by the National Science Foundation under grant EEC-0313747 001

VI. References

[Banka 2005] T. Banka, B. Donovan, V. Chandrasekar, A. Jayasumana, J. Kurose, "Data Transport Challenges in Emerging High-Bandwidth Real-Time Collaborative Adaptive Sensing Systems (poster)" to appear, *IEEE Infocom 2005.*

[CASA NEXRAD-data 2005] http://www.casa.umass.edu/eesr_data_sets

[Hondl 2002] K. Hondl, "Capabilities and Components of the Warning Decision and Support System – Integrated Information (WDSS-II), *Proc. American Meteorological Society Annual Meeting, Jan. 2003 (Long Beach).*

[Horling 2005] B. Horling, V. Lesser, "Distribution Strategies for Collaborative and Adaptive Sensor Networks.." *Proc. Int. Conf. Integration of Knowledge Intensive Multi-Agent Systems (KIMAS 2005)*. April 2005. To appear.

[Jaganathan 1989] V. Jagannathan, R. Dodhiawala, L. Baum, *Blackboard Architectures, Applications*. Academic Press, 1989.

[Jaim 1999] A. K. Jain, M. N. Murty, P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264-323, 1999

[Johnson 1998] J. Johnson, *et al.,* "The Storm Cell Identification and Tracking Algorithm: an Enhanced WSR-88D algorithm," Weather and Forecasting, 13: 263-276, 1998.

[LDM 2005] University Corporation for Atmospheric Research, "Local Data Manager (LDM)." http://my.unidata.ucar.edu/content/software/ldm/index.html

[NOAA 2005] National Oceanic and Atmospheric Administration, "Radar Resources," http://www.ncdc.noaa.gov/oa/radar/radarresources.html

[Manfredi 2005] V. Mandfredi, S. Mahadevan, J. Kurose, "Kalman Filters for Prediction and Tracking in an Adaptive Sensor Network," Technical Report 05-07, CS Dept., U. Massachusetts.

[McLaughlin 2005] D. McLaughlin, V. Chandrasekar, K. Droegemeier, S. Frasier, J. Kurose, F. Junyent, B. Philips, S. Cruz-Pol, J. Colom, "Distributed Collaborative Adaptive Sensing for Improved Detection, Understanding, and Predicting of Atmospheric Hazards, *Proc. Ann. Am. Meteorological Soc. Mtg.*

[OneNet 2005] OneNet, "Oklahoma's Technology Network", http://www.onenet.net

[Schmidtt 2002] J. Schmitt, M. Zink, S. Theiss, R. Steinmetz, "Improving the Startup Behavior of TCP-Friendly Media Transmission," *Proc. Int. Network Conference (INC02).*