# REINAS: Real-Time Environmental Information Network and Analysis System: Phase II Requirements Definition*

P.E. Mantey, J.J Garcia-Luna, H.G. Kolsky,
D.D.E. Long, A.T. Pang, E.C. Rosen, C. Tang,
B.R. Montague, M.D. Abram, W.W. Macy,
B.R. Gritton (MBARI), J. Paduan, W. Nuss
(NPS)

UCSC-CRL-93-34
July 26, 1993

Baskin Center for
Computer Engineering and Information Sciences
University of California, Santa Cruz
Santa Cruz, CA 95064 U.S.A.

**Keywords:** Real-time, Environmental, Sensor, Data Management, Visualization, Seabreeze, Monterey Bay, Coastal Meteorology, REINAS

*Supported by a Grant from the Office of Naval Research, No. N-00014-92-J-1807

# Contents

# 0. ABSTRACT

REINAS is a research and development program with the goal of designing, developing and testing an operational prototype system for data acquisition, data management, and visualization. This system is to support the real-time utilization of advanced instrumentation in environmental science where continuous time measurements and improved spatial resolution allow monitoring and understanding environmental phenomena in much greater detail than has previously been possible. The system will also support the retrospective use of integrated environmental data sets.

The project is a multi-year effort of the Baskin Center for Computer Engineering and Information Sciences of the University of California, Santa Cruz, in cooperation with environmental scientists from the Naval Postgraduate School (NPS), Monterey Bay Aquarium Research Institute (MBARI), and the Center for Ocean Analysis and Prediction (NOAA/COAP).

This report documents the second phase of the REINAS project during which detailed requirements for the system were defined. During this phase the user requirements for the system were sharpened, evaluations of the key components of the system were carried out, and selections of the technologies to be used in the prototype system were made. A project plan for the system is described along with a refined version of the architecture and its subsystems for data collection, data management, processing, and visualization.

The objective of this document is to provide project participants and reviewers with a detailed requirements definition of REINAS as it enters the design and prototype implementation phase. It spells out the technologies to be applied, and the plans for implementing the project goals.

# 1. Executive Summary

The Baskin Center of the University of California, Santa Cruz (UCSC), in cooperation with environmental and marine scientists of the Monterey Bay region, from the Naval Postgraduate School (NPS), Monterey Bay Aquarium Research Institute (MBARI), and the Center for Ocean Analysis and Prediction (NOAA/COAP), is creating a real-time system for data acquisition, data management, and visualization.

The goal of "REINAS" (Real-Time Environmental Information Network and Analysis System), is to support the utilization of current and advanced instrumentation in environmental science where continuous time measurements and improved spatial resolution allow monitoring and understanding of environmental phenomena in much greater detail than previously possible. This report documents Phase II – the detailed requirements definition and preliminary architecture. Phase I [MLP+93] covered the requirements analysis.

REINAS is a system for conducting both real-time and retrospective regional scale environmental science, monitoring, and forecasting employing the Internet for connectivity. Instruments will be connected by both remote radio links and land-lines. Continuous real-time data are acquired from dispersed sensors and input to a logically integrated but physically distributed database. REINAS provides a standard in-place mechanism for acquiring, delivering, managing, accessing, and analyzing data from multiple sources in the region, reducing the need for ad-hoc per-experiment data system development.

REINAS will be used both by scientists and operational users who require insight into oceanographic and meteorological conditions. End users will have access to current environmental conditions, system status, instrument status, result of simulations, and to the historical database of the environment and REINAS system states. REINAS will support work for both remote/distributed users and single-users in a distributed Internet environment.

REINAS focuses on small scale oceanographic and meteorological phenomena. One initial proof-of-concept focus will be the local Monterey Bay seabreeze air/ocean phenomena. However, the system will be scalable and portable to different spatial and temporal scales.

The creation and use of information-describing platforms, sensors, users, processes, experiments, instruments, sample plans, etc., (i.e., the Metadata), will be supported through the implementation of electronic logs. Data exchange with non-REINAS data sources will be an integral component of the REINAS system. An integrated workstation problem-solving environment supports real-time control functions, scientific data analysis, visualization and modeling. Data from sensors, the historical database, and models will be fusible. Automatic methods of alerting users to interesting changes in the environment will be provided. Scientific data quality and scientifically valid database design will be critical to the success of REINAS. It should be possible to assess the data quality and confidence in the displayed data.

The REINAS architecture is distributed, extensible, and scalable. It combines advances in the areas of real-time operating systems, networks, distributed/federated databases, electronic libraries, data compression, pattern recognition, and visualization. The problem solving and visualization environment will provide investigators with images of environmental trends and dynamic behavior in a geographic context.

The REINAS project has 4 phases: (1) requirements analysis, (2) detailed requirements definition of a prototype system, (3) prototype refinement and support for generic

instrumentation arrays, and (4) proof-of-concept validation. The prototype REINAS will be demonstrated incrementally as different instruments and locations come on line. The final proof-of-concept system is expected to be an operational prototype demonstrating the system's capability is supporting operational use. It should also validate the system's extensibility to meet a full suite of operational needs specific to a particular organization's functional and performance requirements.

# 2. Scientific User's Perspective – Science/Instrumentation

## 2.1 Key Scientific Issues

Current scientific applications and future operational applications will focus on the understanding, nowcasting and short-term prediction of mesoscale circulations particularly in the coastal zone. One example of this type of mesoscale application is to understand the interaction of the sea-breeze with the complex topography of the Monterey Bay region. Key scientific issues include:

- The impact of the complex distribution of thermal forcing imposed by the local topography on the three-dimensional evolution of the sea-breeze circulation
- The interaction between the local cloud distributions and the wind field
- The role of the topography in shaping the local wind pattern under a variety of large-scale flow regimes
- The relationship between the local sea-surface temperature distribution and the winds and clouds
- The forecasting of the time of sea-breeze onset, the extent of stratus coverage and their evolution.

The application of REINAS to the study of the sea-breeze in the Monterey Bay region requires an enhancement of the local observing network and the use of routinely available observations as highlighted in the following sections.

## 2.2 Routine Meteorological Observations

Adequate application of REINAS to meteorological research or operations requires the use of routinely available meteorological data. These data consist of standard hourly surface observations, National Weather Service (NWS) ASOS surface observations, coastal and deep-water moored buoys, merchant ship observations, NWS rawinsonde observations, GOES satellite imagery and numerous operational numerical model output fields. These observational and numerical model data are required to define the structure and evolution of the atmosphere on scales that range in space from 200 $km$ to 3000 $km$ and in time from 6 $h$ to 48 $h$. This synoptic-scale structure provides the background within which the more localized events such as the sea-breeze occur and evolve.

Nearly all of the synoptic-scale data are disseminated routinely over the Global Telecommunications System (GTS) and can potentially be ingested into REINAS through a variety of channels. Since most of this data are in a coded form, the coupling of REINAS to existing ingest/decoding programs is needed. The most likely candidate is to utilize the UNIDATA data stream and its LDM system for decoding and data ingest. Presently, the data are received through a satellite datalink but plans are to distribute this data via Internet. An alternative approach would be to access this data stream via Fleet Numerical Oceanography Center (FNOC).

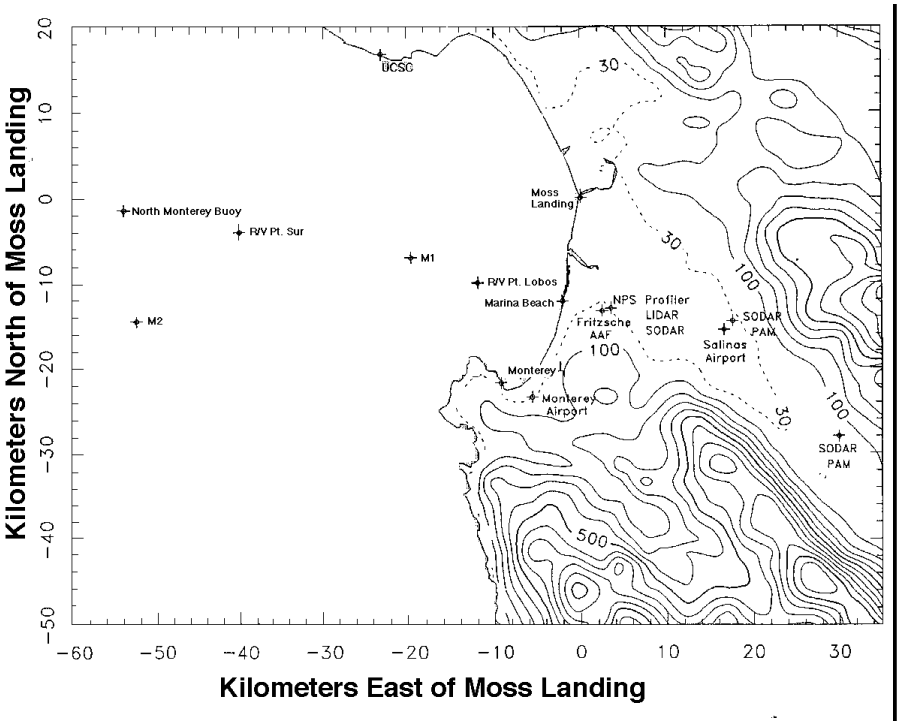


Figure 2.1: Monterey Bay and Instrumentation Sites

## 2.3 Additional Meteorological Observations

Since the routine meteorological observations are insufficient to address the scientific issues highlighted above, REINAS should be capable of incorporating a variety of observational enhancements. To address the scientific questions associated with the sea-breeze in the Monterey Bay region, these enhancements should include a network of surface observing stations, a network of wind profilers with RASS, multiple CODARS, a network of ship and buoy stations, and remotely-piloted aircraft. The network of surface observing systems, both land and ocean, must be sufficiently dense in order to sample the dominant mesoscale structures produced by the topography and sea surface temperature variations. A network of wind profilers and remotely-piloted aircraft would be required to sample the vertical structure over the Monterey Bay region.

The basic approach in designing the network of additional observing stations is to treat the problem as one of nested scales of motion and to deploy observing systems according to the data needs for each scale. Synoptic-scale variations are adequately resolved by the existing network of routine observations mentioned in the previous section. The one exception to this is the lack of observations, even on the large scale, over the ocean. Drifting buoys and adequate numerical model simulations can potentially fill this gap. Research is planned to establish the accuracy of the numerical model forecasts. An intermediate scale is hypothesized to cover the interior valley and mountains between the San Joaquin valley and the coast. Within this intermediate scale, a small-scale network of surface stations covering the Monterey Bay is envisioned. This small-scale network would include additional land stations, moving ship-board stations, and 915 $Mhz$ profilers with RASS around the Bay

and on a ship. Research is planned to determine the best siting of the instruments as well as determine the number needed. Preliminary estimates suggest that as many as 25 new surface stations on land and 5-6 ship stations are required. The CODARS are essential for providing very detailed over water wind and surface current measurements (approximately 2 km resolution). These instruments provide the finest-scale structure in the nested array.

The most critical area of measurement needed to fully describe the sea-breeze is some vertical structure. This is a classic problem in meteorology and the best approach to solving this problem is to use high temporal frequency measurements and interpolate in space from these measurements. Adequate coarse spatial resolution over the Monterey Bay and adjacent region can probably be obtained with 5-6 915 $Mhz$ profilers. NPS has one that can be used for development purposes. Negotiations with NOAA Wave Propagation Lab (WPL) are planned (and additional funding sought) to obtain 5 or 6 additional profilers including one shipboard profiler for a time limited demonstration period. This REINAS demonstration period must be 1-2 months in duration when the sea-breeze is normally active (between April and September). In addition to the high temporal frequency observations from the profilers, the use of remotely piloted vehicles (RPV's) to provide time-limited but spatially dense measurements above the surface is desired. These instruments add an important interactive capability as well as providing critical detailed vertical structure measurements of important structures in the atmosphere.

## 2.4 Data Quality and Processing

The data quality and the methods used to extract meaningful measurements from raw observations are critical issues that impact the scientific application of REINAS. For the routine observations, the data quality cannot be influenced but efforts must be made to assess the quality of each measurement. Calculation of root-mean squared errors relative to large-scale analyses can identify biases. Other methods of assessing data quality of existing stations are under investigation. For the additional instrumentation deployed as part of REINAS, research quality instrumentation is needed to ensure accurate measurements and reliable stations. The accuracy specifications are covered elsewhere for specific instruments.

The more significant issue is how best to extract meaningful meteorological or oceanographic measurements from the raw observations. High temporal frequency observations that are spaced between 20 and 200 $km$ apart present a challenging analysis problem. The primary scientific goal is to represent the three-dimensional flow over time from these sets of observations. Sufficient averaging or Fourier extraction from the high frequency observations must be done. Ideally, REINAS will store raw high frequency observations for a sufficient period to allow the user to investigate the best method for deriving spatially meaningful fields from these data. Another associated issue is the methods used to represent the spatially scattered observations during visualization. Traditional data assimilation methods are not appropriate for the scales being sampled by the REINAS network. Standard mathematical interpolation techniques fail to utilize inherent dynamic relationships between observations. Research is underway to test methods of data assimilation that make use of dynamic constraints as well as accurate mathematical interpolations. The eventual use of a mesoscale numerical model within REINAS is desirable for this problem.

## 2.5 Mesoscale Numerical Modeling

In addition to potentially providing a crucial base from which to perform quality control of the special data and real-time data assimilation of all observations, a sophisticated regional mesoscale coupled atmosphere-ocean numerical model is necessary in order for REINAS to become a fully integrated nowcasting *and* forecasting visualization tool. Such a model, e.g., the Navy's Coupled Ocean and Atmospheric Prediction System (COAMPS) or the Penn State/NCAR mesoscale model (MM5), with the flexibility of multiple horizontal nesting and state-of-the-art physics will provide REINAS dynamically consistent time continuous fields from which high quality visualizations of the current and future evolution of the flow can be made. Furthermore, the utilization of a numerical model will enable scientific investigation of the various physical phenomena affecting the REINAS region of interest.

Conceivably, the model could be used in two modes. The first mode would be in an update cycle where the model simulation is influenced by incoming conventional and special data. In this mode, the model would be run continuously but in short discrete bursts optimally designed to not lag or outrun real-time to any great extent. The "nowcast" simulations obtained in this mode could be used as a quality check for the incoming data, for example, with problem observations either corrected or discarded and the simulation burst run over. These simulation bursts could then be used for visualization. The second mode would be a pure forecast cycle in which a simulation of some longer duration would be generated from the latest analysis. These forecast simulations could be used to make predictions of the flow at a later time, which would aid in the decisions of whether to initiate any special data collection activities.

Research is necessary to successfully implement a mesoscale regional model for the Monterey Bay area. Specific issues include the model to use, the proper horizontal domain and nests, vertical resolution, the availability of sufficient upper-atmospheric measurements within the mesoscale domain and the upstream boundary over the normally data sparse ocean. A critical research issue is the technique utilized to ingest incoming variable data streams from diverse sources into a simulation.

# 3. Requirements Analysis

## 3.1 User Profiles

As part of this phase of requirements analysis, the potential users of a REINAS system were identified. In the prototype implementation we may not be able to work directly with users from all of the categories so defined, but we will attempt to provide functionality relevant to each of the categories.

The initial user community served by REINAS will consist of the REINAS science partners located at NPS and MBARI. The developers themselves will serve as the target population for the engineering aspects of the system, e.g., instrumentation engineers and system developers.

The user profiles defined are listed in the following sections.

### 3.1.1 Operational Users:

- *Operational Forecaster*: Needs current situation visualization, nowcasting, and short-range forecasting. Traditional meteorology map product displays are required. An easy to use suite of "canned" products must be available. Easy, fast access to previous environmental situations with a similar signature would provide a new, significant capability.
- *Operational Policy Maker/Planner*: Needs integrated "birds eye" views of complete geographical area and the ability to focus (zoom in) on smaller, specified areas of interest. The ability to set up scenarios (models) and view results is desirable. A user-friendly interface similar to that of operational forecaster is required.
- *Disaster Control*: Needs immediate view in the form of current observations, nowcasts, and local climatologies. Requirements similar to operational planner, but requires additional map data and products, i.e., environmental Sensitivity Index maps. Also will require the capability to plug-in models specific to the hazardous response activity.
- *Students and Casual "browsers"*: Need canned products and rapid visualization, i.e., access to precomputed visualization results. An integrated visualization and "dry-lab" modeling capability makes REINAS a significant educational asset. It should likewise provide a base for educational research projects.

### 3.1.2 Scientific Users:

- *Retrospective Researcher*: Needs synoptic views and historical analysis. Requires data quality information, ability to readily construct complex database queries, and ability to write C programs which access REINAS data and functionality through well documented APIs. Should also be able to make SQL queries. These custom programs are to become part of the individual researchers REINAS environment.
- *Experimental Researcher*: Needs include those of both operational forecasters and retrospective researchers, as well as ability to monitor instrument status, control instrument settings, and view "data streams" in real-time.
- *Sensor Scientist*: Needs detailed control of instrument and detailed view of instrument status. Requires simple mechanisms for interfacing and low-level real-time control.

### 3.1.3 Developers/Intrumentation Engineers:

- *Instrumentation Engineer*: Needs a "cookbook" approach for adding new instruments to REINAS. Support tools should be such that REINAS is the environment of choice for a new instrument development project. REINAS should provide interfaces to instruments, e.g., for calibration or other parameter changes.
- *Network Engineer*: Requires tools to support an enterprise scale/style network using Internet.
- *System Developer*: Requires well documented APIs providing access to all REINAS functionality, ability to readily integrate and test new REINAS applications, ability to run a REINAS node in "development" mode, and debugging and timing tools.
- *Database Administrator*: Need traditional capabilities, e.g., ability to tune database, ability to visualize its utilization and access patterns, and ability to define database organization at both physical and logical levels. Automating these functions is a research topic.
- *System Manager/operator*: Need traditional capabilities, e.g., ability to control user's access/capabilities, ability to view system status and utilization, and ability to tune system on-the-fly.

### 3.1.4 Special Applications:

Some users of REINAS services will be automated procedures which require timely and regular access to data and sensors. Profiles for this class of user will probably grow over time but for now we can identify two primary categories.

1. Data Mining Applications: A suite of applications, which run continuously in the background, to detect and classify significant patterns that may exist in the observation data across multiple scales in space and time. The system must be able to gracefully distribute the loads placed on it by these applications to times of minimal impact on primary operations. In addition, the system should support the capability to easily extend the number and type of these applications.
2. Standard and Customized Product Generation: A suite of applications which produce scientific or engineering information products. The products may be produced on a regular schedule, based on the occurrence of an event, or based on input data availability. The system must able to maintain a description of and requirements for such products and produce them accordingly. Standard products are those needed by a large cross section of REINAS users or by high priority users. In addition, the system should provide a capability for users to build customized product profiles for automatic production by the system.

## 3.2 Data Acquisition and Delivery Requirements

There are three major types of system requirements in REINAS: data acquisition and delivery (DA), data management (DM), and data visualization (VS). The rest of this section addresses each type of system requirement. System requirements are expected to evolve throughout all phases of the project life-cycle.

These requirements define the kind of acquisition devices and delivery environment that REINAS must accommodate and integrate into a seamless whole. Detailed requirements for interfacing to external data sources will be defined in the next phase of the project.

### 3.2.1 DA 1: REINAS Instrument Data Sources

The REINAS system is being designed to accomodate data from a variety of data sources, especially surface and ocean MET stations, wind profilers, current profilers such as the CODAR and ADCP, and satellite imagery.

MET stations will be the most numerous class of data source connected to REINAS. Scientifically interesting or *research quality* stations must include sensors that measure five basic properties to the accuracies specified in Table 3.1: air temperature, pressure, humidity, wind speed, and wind direction. In addition, other instruments to measure short and long-wave solar radiance and precipitation may also be present and of interest. MET station instruments can be sampled as often as desired. REINAS is experimenting with sample periods as small as one second and generating averaged datasets for archival with temporal resolutions of one minute; however, a wide variety of sample periods will be prevalent among REINAS-accessible MET stations.

| Measurement | Resolution |
|---|---|
| *air temperature* | $\pm 0.1$ °C |
| *barometric pressure* | $\pm 0.1$ mb |
| *relative humidity* | $\pm 0.01$ % |
| *wind speed* | $\pm 0.1$ m/s |
| *wind direction* | $\pm 1.0$ ° |

Table 3.1: MET Instrument Tolerances

Wind profilers generate a vertical profile of the atmosphere above the instrument; by examining the strength and phase shift of radar return echoes, the wind speed and direction at various heights can be determined. Over a prescribed time interval (nominally one hour), a wind profiler generates an array of wind velocities, indexed by height, with typical vertical resolutions of 250 m to altitudes below 7 km.

Surface and ocean current profilers, such as the CODAR and ADCP, generate current data in a variety of forms. Individual CODAR stations generate surface current speeds for various angles and distances along radials extending from the CODAR instrument site. Radial data from two or more appropriately located CODAR instruments can be combined to generate more useful two-dimensional gridded datasets of surface current velocities. Typical CODAR sampling periods are one to three hours, although interesting data can be extracted in shorter intervals. The ADCP provides ocean current profiles below the surface; typically mounted on surface buoys or ships, these instruments generate an array of ocean current velocities, representing the motion of ocean below the ADCP with typical resolutions better than 10 m to depths of 250 m. A typical ADCP sampling period is 15 minutes.

Satellite imagery from NOAA polar orbiters and GEOS geosynchronous satellites will also be incorporated into REINAS. Although typically limited to four passes generating usable data of Monterey Bay per day, polar orbiters provide relatively high resolution images at multiple infrared wavelengths. Complementing this source, GEOS images can

be obtained hourly and provide larger scale but also lower resolution images. Both sources are scientifically useful to REINAS, allowing cloud cover, sea temperature, and solar energy contributations to be measured.

Thus, REINAS must be able to accept data from many diverse sources at greatly varying rates. Sampling periods may be as short as one second (e.g., surface MET stations) or as long as several hours (e.g., vertical wind profiler or CODAR in long-term averaging mode). Instrument interfacing hardware will also vary greatly. Some instruments, such as surface MET stations, will provide relatively primitive interfaces (e.g., a generic datalogger). Others may interface to REINAS using modern microcomputers (e.g., CODAR).

Another design consideration related to instrumentation is the need to accommodate instruments that require low-power consumption, at least for the transmission of data. Examples of such systems are MBARI's OASIS buoys located in the Monterey Bay, which provide a large collection of useful data appropriate to REINAS. Power considerations can limit the nominal sampling rate of an instrument to something larger than what would otherwise be desired; or more likely, a strict power budget can impose long latencies (hours) which delay the reception of data into the REINAS system.

To help isolate the majority of the REINAS architecture from the details of each instrument installation, REINAS will operate on the model that each instrument has a microcomputer-level interface available. When no such microcomputer already exists as part of the interface, REINAS requires the addition of such a computer; in practice, this is neither a difficult nor expensive requirement to impose, especially when compared with the cost of the typical instrument (in most cases, surface MET stations).

The following paragraphs summarize specific characteristics of the data sources in REINAS, including their number, data rates, and compression requirements.

**Instrument Data Source Characteristics:**

1. **MET stations:** Ideally, REINAS will have real-time access to approximately 25 surface MET stations in the Monterey Bay area. Currently, REINAS receives data from six MET stations. Sampling periods will vary from one second to five minutes, depending on external factors. Even at one second sample periods, the typical MET station generates data at rate well less than 300 characters per second. At these high sample rates, MET station data is also extremely compressible for archival purposes; compression ratios of 10 to 25 are reasonable. MET station data will be maintained at received time resolutions for a period of about two weeks, after which it will be averaged to a standard temporal resolution (probably one minute) for archiving.
2. **Buoys:** Ocean buoys moored in or near Monterey Bay are a primary source of oceanographic data for REINAS. Currently, two MBARI ocean buoys provide REINAS with ocean surface MET data, ocean current ADCP data, ocean chemistry (CTD) and composition data, as well as other measurements on a regular basis.
3. **Wind Profiler Radar:** Currently, REINAS receives data from one wind profiler, although the potential for as many as eight sites over some time periods exists. Wind profilers average radar return echoes to generate relatively noisy six minute samples as well as relatively clean one hour profiles.
4. **CODAR:** REINAS expects to receive radial data directly from CODAR instrument sites in real-time, allowing REINAS scientists to work directly with this "raw" data or generate gridded vector images of ocean surface currents. Radials can be collected

at half hour intervals, although longer sample periods are necessary to reduce noise levels.

In addition, REINAS may also receive CODAR spectra, allowing other parameters, such as surface winds, to be estimated using new algorithms. Collection of raw CODAR spectra will dramatically increase the amount of data received from CODAR instrument sites.

5. **Satellites:** Satellite images from geostationary GEOS and NOAA polar orbiters represent the largest regular instrument datasets REINAS is likely to encounter. Typical GEOS images are $1024 \times 600$ pixels in size with approximately 8 km spatial resolution and one byte (256 levels) of fidelity per channel. AVHRR images from NOAA polar orbiters vary in useful dimensions, but one full pass can generate as much as 8 MB of data.

   Depending on the number of passes from polar orbiters and the exact size and resolution of the extracted images, satellite imagery can generate as much as 20 MB (and potentially more) raw data daily.

6. **Shipboard Instruments:** Ocean surface and subsurface observations will be available from ships which are connected to the network via microwave, satellite, and/or packet radio links to shore. Typical observations of interest include but are not limited to surface meteorological parameters, ocean surface temperatures, current velocities, solar radiation, chemical nutrients. Many of these data streams are available at one second sampling intervals. In addition, current velocity profiles and standard hydrographic profiles of physical, chemical, and biological measurements may be available. These data will typically be available in bursts at predefined stations where the vertical profiles are taken. The Acoustic Doppler Current Profiler (ADCP) is an exception, this stream will produce profiles regularly along the track of the ship.

**Summary of Data Rates:** Table 3.2 provides estimates of the daily data rates and total archiving requirements expected to be collected by REINAS with all instruments operational. The discussion in Source Characteristics describes the table and presents a scenario that addresses the issues of what kinds of data are collected, the forms in which the data are maintained (raw, averaged uncompressed, averaged compressed), the amount of time it is kept in each of these forms, and based on this scenario, the total amount of data which must be maintained by REINAS.

| Instrument | Expected Number | Sample period | Average period | Bytes/ average period | Bytes/ day per instrument | Bytes/ day (all) | Bytes/ day after compression | Estimated compression ratio |
|---|---|---|---|---|---|---|---|---|
| MET station | 25 | 1 sec | 1 min | 90 | 128 kB | 3.20 MB | 128 kB | 25 |
| Profiler | 8 | 6 min | 1 hour | 6144 | 144 kB | 1.15 MB | 115 kB | 10 |
| ADCP | 2 | 15 min | 15 min | 1280 | 120 kB | 0.24 MB | 48 kB | 5 |
| CODAR | 3 | | 3 hours | 8192 | 64 kB | 0.19 MB | 19 kB | 10 |
| AVHRR | 4 | | | | 8 MB | 24.0 MB | 1.2 MB | 20 |
| GEOS | 1 | 1 hour | 1 hour | 600 kB | 14 MB | 14.0 MB | 720 kB | 20 |
| Totals | | | | | 22 MB | 42.7 MB | 2.2 MB | 20 |

Table 3.2: Expected Instrument Data Rates

### 3.2.2 DA 2: Methods of Obtaining Local Instrument Data

The method in which data are obtained from a instrument can vary greatly from instrument to instrument. The method of choice depends on the characteristics of the instrument itself. For some instruments, data can be obtained only on a periodic basis, with each data update containing the average or raw data of multiple samples. The need for periodic updates may arise because of the need to conserve transmit power, or the need to conserve transmission bandwidth. The demonstration system described in Chapter 5 uses FTP to obtain data from MBARI's OASIS buoys on a periodic basis.

The preferred method of obtaining data are by means of an interactive application, with which the instrument delivers data to the appropriate server in real time, and a server or an end user is able to control the instrument remotely.

Regardless of the type of application-level protocol used, a TCP/IP, UDP, or RPC-level interface is needed on top of the network-level service to support the end-to-end services needed by the application program used to communicate with or control the remote instrument. The data are exchanged between the instrument microcomputer (or equivalent) and the appropriate REINAS server. The underlying network support can be existing Internet links, dial-up commercial telephone SLIP/PPP links, or wireless modems.

### 3.2.3 DA 3: Data from Models

Output from numerical models must also be accepted. Models tend to be very large so that model data will normally be accessed over the network from an archive. The decisions whether to save model output or to recompute it when needed depends on the economics at the time. Grid spacing in models varies depending on the scale of interest. It ranges from tens to hundreds of kilometers in the horizontal. The vertical spacing is non-uniform ranging from a few hundred meters to several kilometers.

### 3.2.4 DA 4: Data from External Sources

REINAS will have to be able to import and export data sets from other external databases in standard and non-standard formats.

### 3.2.5 DA 5: Data Compression

The system should include support for lossless and lossy compression of data, as appropriate [Wil91]. In REINAS, we estimate that some data will be averaged as soon as it is collected, and some will remain uncompressed. Two weeks is the estimated period of time that real-time users will want data immediately available. The data collected hourly from all MET stations will remain uncompressed. These data will be one minute averages collected each hour. This data set will provide sufficient information to retrospective users about whether an interesting event has occurred. Uncompressed headers of compressed data will allow indexing into compressed data files in one hour intervals. The cost of five parameters each requiring five bytes collected hourly from 25 MET stations and kept for one year is 5x5x25x24x365 = 5Mb. This is not very much considering its value to retrospective users.

Adding all values in Table 3.2 in the column indicating the data collected from all instruments per day gives a result of about 10Mb. If data are kept uncompressed for about 2 weeks then uncompressed data will require from 100Mb to 200Mb storage space. Adding all values in the column for compressed data yields a result less than 1 Mb per day. This means that less than 1 Gigabyte is likely to be collected in a year. Since the data collected in a year can be stored on a 1 Gigabyte disk a retrospective user can retrieve data without requiring that it be loaded from a tape. Of course this sizing may be an underestimate if some of the data sources turn out to be much larger.

## 3.3 Data Management Requirements

### 3.3.1 DM 1: Data Management Environment – Operational Profile; Clients; Data, Metadata, and Control Flows

The REINAS Data Management component must serve all elements of the REINAS system: (1) data acquisition and delivery, (2) standard product generation, (3) visualization, (4) system/network management and control, (5) application programs, and (6) scientist end-users

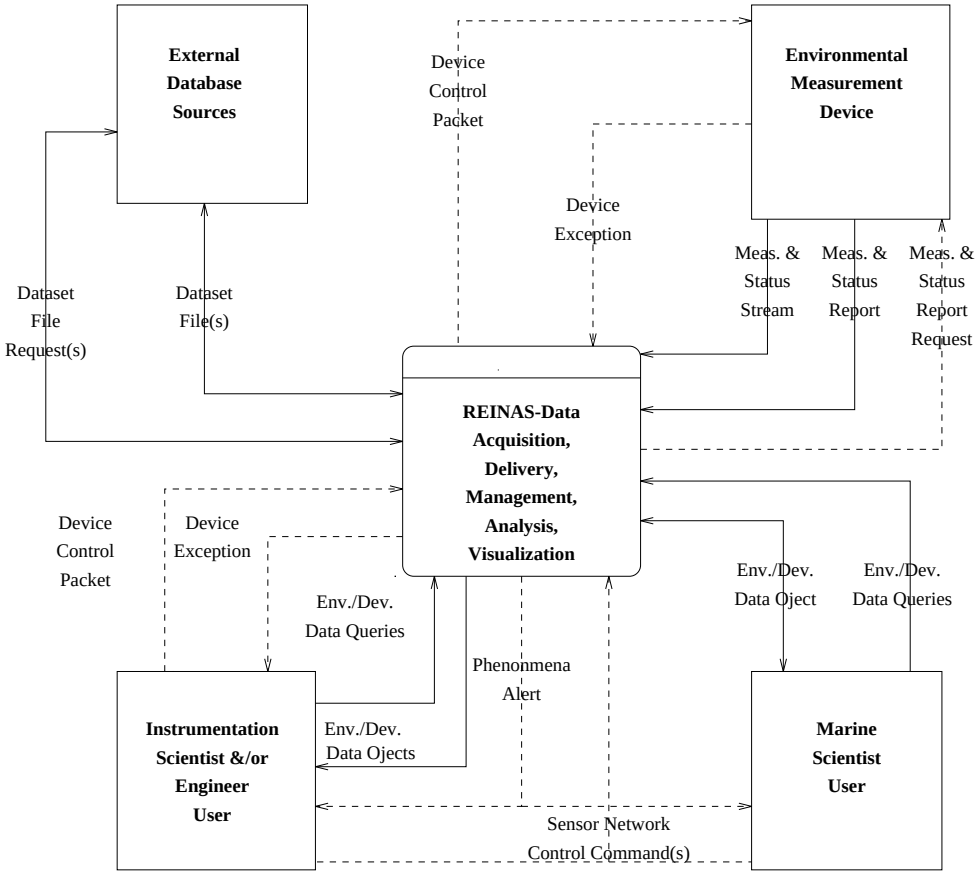


Figure 3.1: REINAS System Context

It must provide access to data and information associated with the current, historical, and predicted geophysical environment within which REINAS operates. In addition, it must track the state of acquisition equipment and support the process of instrument reconfiguration. In addition it must track the state of acquisition equipment, REINAS system components, and support the process of system reconfiguration.

Figure 3.1 shows a schematic of the context of the REINAS system. An overview of the REINAS data sources and users, and the data/control flow interfaces is shown.

Figure 3.2 expands this to an overview of internal REINAS functional areas and their interfaces to the REINAS Data Management (RDM) function. It also depicts the flow of data, metadata and commands among the components. Detailed evaluation of these processes and data objects contribute to a generalized information architecture for REINAS data management. The architecture is described in an entity- relationship diagram with detailed reports on the structure and content of each entity.

The data management function must operate in an environment that spans geographically dispersed system components that are implemented using diverse technological platforms (i.e., a heterogeneous distributed system). In addition, REINAS will not have complete control over all system elements. Individual nodes in the REINAS Data Management Subsystem may operate to meet the mission and needs of local users as well as REINAS users. Therefore, REINAS must effectively integrate nodes with autonomous control.

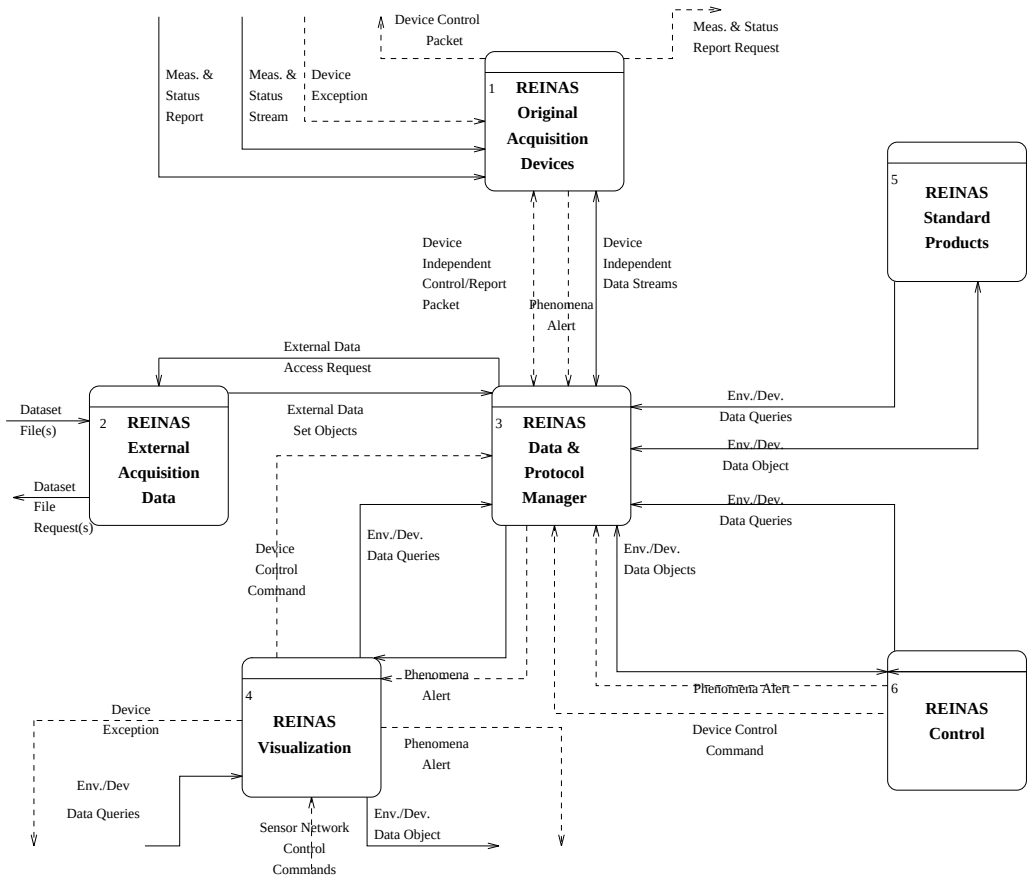


Figure 3.2: REINAS Metadata Flow and Command Flow

| Task Objectives, Object Types, and Performance | |
|---|---|
| Task Objective: | Monitoring |
| | Replaying |
| | Resampling |
| | Fusing |
| | Summarizing |
| | Quality Controlling (QC) |
| | Producing |
| | Exploring |
| Object Types: | Measurement Parameter Time Series |
| | Observation Parameter Time Series |
| | Status Parameter Time Series |
| | Quality Time Series |
| | Observation Parameters State Vector |
| | Observation Parameter Profile |
| | Observation Parameter Field |
| | Feature/Phenomena/Event Object |
| | Sources |
| | Networks |
| | Nodes |
| | Products/Data Sets |
| Performance: | Real-Time |
| | Retrospective |

Table 3.3: A Classification of REINAS Data Access Operations

In addition, basic characteristics of typical DBMS systems (integrity control, security enforcement, recovery, concurrency control, distribution of data, development tools, etc.), must be combined with characteristics of real-time systems (e.g., guaranteed response time). Other operational issues include generic data object types, physical storage and access mechanisms, and data exchange mechanisms.

### 3.3.2 DM 2: Data Management Usage – Realms, Functions, and Data Classes

An analysis of the anticipated client tasks of the REINAS Data Manager indicate that there is significant overlap in their data access operations and objects. Separate design realms may be classified as the union of all tasks by differences in objective, data object types, and performance expectations. The proposed classification is listed in Table 3.3.

Many task objectives span both the real-time and retrospective performance realms and apply to multiple object types. The object types may be further partitioning into three separate realms: (1) stream space for those aggregate objects of interest that are generated from one source; (2) observation space for those aggregate objects that are of interest for multidimensional scientific analysis and display; and (3) system space for those objects used to monitor and control the operation of the system.

Table 3.4 lists the REINAS Data Management Realms partitioned by performance and object type classifications. The Realm Characteristics are:

Realm Topography and Associated Functions

| Stream Space: | Real-Time | Retrospective |
|---|---|---|
| | 1. Monitor Meas | 1. Replay Meas |
| | 2. Monitor Obs | 2. Replay Obs |
| | 3. Summarize Meas | 3. Summarize Meas |
| | 4. Summarize Obs | 4. Summarize Obs |
| | 5. Monitor Meas Quick QC | 5. Historical Correlative QC |
| | 6. Monitor Obs Quick QC | 6. Resample Meas |
| | 7. Produce Obs Quick QC | 7. Resample Obs |
| **Observation Space:** | **Real-Time** | **Retrospective** |
| | 1. Monitor Obs State Vector | 1. Replay Obs State Vector |
| | 2. Monitor Obs Profile | 2. Replay Obs Profile |
| | 3. Monitor Obs 2D Field | 3. Replay Obs 2D Field |
| | 4. Monitor Obs 3D Field | 4. Replay Obs 3D Field |
| | 5. Monitor Feature Phenomena/Event | 5. Replay Feature Phenomena/Event |
| | 6. Monitor Obs Quick QC | 6. Replay Obs Quick QC |
| | 7. Resample Obs Profile | 7. Multiparameter Correlative QC |
| | 8. Resample Obs 2D Field | 8. Resample Obs Profile |
| | 9. Resample Obs 3D Field | 9. Resample Obs 2D Field |
| | | 10. Resample Obs 3D Field |
| | | 11. Fuse Obs Profiles |
| | | 12. Fuse Obs 2D Field |
| | | 13. Fuse Obs 3D Field |
| | | 14. Fuse Feature/Phenomena/Events |
| | | 15. Explore |
| **System Space:** | **Real-Time** | **Retrospective** |
| | 1. Network Monitor | 1. Network Replay |
| | 2. Network Summary Monitor | 2. Network Summary |
| | 3. Source Status Monitor | 3. Source Status Replay |
| | 4. Source Status Summary Mon. | 4. Source Status Summary Replay |
| | 5 Data Product Status Monitor | 5. Data Product Status |

Table 3.4: REINAS Data Management Realms

**Stream Space:** Comprised objects that are pertinent to the definition, status, and output of a single source of data. The source generates streams of objects that are bounded by start/end times determined by some criteria (e.g. data collection period, downstream processing changes, etc). All output objects in a given stream are from the same source configuration, subjected to the same processing sequence, etc.

**Observations Space:** Comprised of objects that are aggregates of objects from multiple sources that provide the intersecting parameter domains (e.g. all temperature measurements from all sources within specified time window). In real-time mode, observation objects will most likely be restricted to one parameter domain. But, in the retrospective mode, objects

will include multidimensional parameter vectors.

**System Space:** Comprised of objects pertinent to the definition, status, and configuration, operation of system elements such as the sensor network, data sources/processing/ products, etc.

**Real-Time Mode:** Associated with objects and operations that support the assessment of the current and/or near past status of the environment and/or the REINAS system.

**Retrospective Mode:** Associated with objects and operations which support the assessment of the state of the environment at any point of time in the past, predicted future states, as well as trends, features, patterns, and relationships as specified by ad hoc researcher specifications. In addition, system states may be able to be "replayed" in this mode.

REINAS must develop a data management architecture which accommodates the storage and access of data needed in all realms defined above. This will require the creation of a unifying information model with one or more physical implementations designed to meet functional and performance expectations in each realm. In addition, since this must be implemented in a heterogeneous environment, it will be important to completely specify the format of available data sources in a standard model. The specification must model the following characteristics:

- Content: What is included - meanings of individual items
- Structure: How the data items are grouped to reflect general semantics
- Representation: How data items are mapped to real world facts (data types, units)

### 3.3.3 DM 3: REINAS Information Architecture

REINAS must be able to support data from multiple sources, with multiple levels of interpretation (processing levels), for scientists from multiple disciplines. Further, the system must be portable to other instrument suites and extendable in functionality and new technology assimilation. There must be some conceptual anchor for users of such a dynamic system. This anchor must provide a stable paradigm for interaction in terms of functions performed and data access. A generalized information model provides the technical mechanism to create the required stability.

An information model is a technology independent description of the entities which are created and used throughout the processes of doing science. In this case, the enterprise supported by REINAS will include real-time and retrospective environmental science activities as well as system management and control activities. Such a model may be used as the basis for database design, user interface design, data exchange standards, etc. The REINAS information model must identify and describe the static and dynamic characteristics of the entities involved in the marine science enterprise.

A complete model specification is being produced and will be included in a detailed RDM specification. For the purposes of this document, we identify in Table 3.5 the characteristics which must be accommodated in the information model. Detailed definitions of these objects, their inherent relationships, and associated operations will be defined in the full information model.

### 3.3.4 DM 4: Data Management Administration

The administration component of the Data Management Subsystem is to provide inherent functionality, services, tools, and products to users who are responsible for the development, operation, and maintenance of the data management subsystem of REINAS. Database application development tools, performance monitors, and database design and evolution tools are included in administration.

### 3.3.5 Data Management Approach

The realms defined above as differences in objective, data object types, and performance expectations, indicate a probable bifurcation of the semantics associated with "observation objects" residing in the stream space and observation space. Considerations of mode (real-time vs. retrospective) will probably imply adjustments to physical implementation based on the performance needs of real-time clients. Therefore, at a minimum we have a problem of heterogeneous semantics within the REINAS Data Management Subsystem.

For example, a sensor produces a series of observations of a single variable at points in time/space which could vary in any coordinate dimension. One view of a single observation from this sensor is its membership in and relationship to other measurements from the same sensor over time (i.e. how did the parameter change over time).

An orthogonal view, from a data management perspective, is how this measurement relates to measurements of other parameters (i.e. other sources of different types) co-located in space and time. Or, another view, how it relates to other measurements of the same parameter type made by different sources at different coordinate locations in time/space.

Therefore we conclude that:

1. Given the challenge imposed by heterogeneous semantics and bimodal operational profiles, REINAS should restrict its data management efforts to handle these issues in a system environment that is *not* comprised of general distributed/autonomous data management nodes. REINAS should focus on the problem of providing effective data management services to clients (which may be heterogeneous/distributed) from the realms defined above.
2. REINAS must determine the best approach for identifying and reconciling the semantics of each realm. One approach would be to model both realms and identify semantic equivalence, relatedness, resemblance, discrepancy, incompatibility. Analysis of these models should imply one of two approaches, model integration or model segregation. Database manipulation language will have to provide extensions queries that span both models.

## 3.4 Visualization Requirements List

The visualization component of REINAS seeks to address the needs of the operational forecasters and research scientists. It also serves as a data base browser, intelligent instrument monitor, supports instrument "steering," and provides a means of access to other components of REINAS. Research work in this area offers the opportunity of addressing some of the unique visualization requirements of REINAS such as handling noisy, sparse, real-time data sets coming from different sources.

### 3.4.1 VS 1: Support Environmental Science Visualization Needs

The science users of REINAS may come from the fields of Atmospheric Science or Oceanography. Since different disciplines have different conventions, one of the visualization goals is to provide a single platform that can handle both ocean and atmospheric data sets. It should provide a coherent view of the ongoing physical processes.

### 3.4.2 VS 2: Support Bi-Modal Visualization Operation

The visualization system may be used for viewing live data as measurements are received by the system, for viewing results of assimilation of measurements and model output, or in retrospective mode for viewing historical data. The viewing of live data may be used by an experimental researcher or sensor scientist who needs to monitor or control instruments. The support for visualization will support "now casting" that combines the measurements and the state estimates provided by models of physical phenomena, as well as other products needed by operational users and forecasters. The retrospective mode will provide researchers with support for analyzing historical data sets, as retrieved from the database.

- Provide real-time display of instrumentation data as they come in. If there are delays in sensors reporting their data sets, these will have to be registered in time.
- Provide intuitive means for data exploration. Often times researchers are deluged with data and do not know where to start. This may be overcome by providing them with easy to use yet powerful visualization tools that encourage experimentation.
- Tradeoffs will have to be made in terms of graphics quality versus response time. For example, provide progressive refinement option for high-volume data sets that require interactive speed interactions.
- Provide image storage/retrieval of visualization products.

### 3.4.3 VS 3: Support Multiple Visualization Stations

REINAS will be supporting a host of scientists and end users. These users should be able to access REINAS through different, geographically distributed workstations. Some of these may even be on mobile platforms, such as aboard ships. Depending on their graphics capabilities and communication bandwidth, as well as user classification, visualization stations may have different default startup parameters. Minimal configuration should include keyboard, mouse, bit-map screen displays and appropriate communication/network links. To support the minimal configuration, visualization products and images will have to be produced remotely on a visualization server, then compressed and sent to the local display. For sites with more powerful workstations, some of the visualization work may be performed locally, thereby reducing communication traffic.

### 3.4.4 VS 4: Integrate Presentation of Data from Wide Array of Data Sources

We anticipate that REINAS data will come from a variety of sensors and numerical models. Data from these sources will have to be fused intelligibly to aid scientists in understanding the interactions among different data sets. To do this, we need to address various data formats, types, characteristics, and generate appropriate graphics display for

various data types. Another unique challenge of visualizing environmental data is the spatial sparsity of the data points. Appropriate analysis and interpolation techniques and fusion with model data, will be required. Data assimilation techniques using an appropriate atmospheric and oceanographic model will be explored to perform this interpolation in a dynamically consistent manner.

### 3.4.5 VS 5: Integrate Data Quality into Visualization

The scientist must be given some indication of how believable are the data being displayed. The data may be noisy, corrupted, missing, have some variance from historical data, have some variance from another instrument measuring the same area, widely different from forecasts, widely different from neighboring sensors, may be interpolated, and numerous other situations that may degrade the confidence level of the displayed image. In any case, the user should be provided these data quality information to allow them to quickly evaluate the importance of individual measurements.

### 3.4.6 VS 6: Provide Multiple Interpolation Methods

Due to the sparse nature of field data often found in the environmental science setting, one must almost always resort to some form of interpolation to fill in the gaps in the data set. Simple linear operation performed in graphics such as smooth shading calculations are often inadequate and may lead to the wrong conclusions. Thus, it is important that the users are made aware whenever such operations are performed, and that interpolated values be highlighted. Furthermore, users should be provided with different options of performing the interpolation. For example, objective analysis where values can be weighted by the relative importance and reliability of various sensors; measured values may also be used as boundary conditions for physically based mathematical models to fill in the holes. In order to make these options as domain-independent as possible, we are looking at separating the interpolation techniques as separate modules. Different modules can then be specified and invoked with different data sets. Results from different interpolation methods may also be compared using the data quality information display described above.

### 3.4.7 VS 7: Provide Recording Capability

Several aspects of the visualization session may be recorded for customization purposes or later playback. Depending on the user class, different levels of the visualization interface are available. These may be recorded after initial setup or alternatively may be specified in a user modifiable startup file. During the visualization process, the user may have created some new visualization technique encoded as sparts (see section 5.5.2). These spart definitions may be saved for later re-use. Finally, visualization products generated by analysis of data sets or forecast products from models or sensors may be saved for later reference. These visualization products may be considered as another data source and be ingested back into the database. They may also be played back in an animation loop.

### 3.4.8 VS 8: Interface with Database System

Data come in various formats within the environmental science community. Some of the standard formats, such as netCDF, will be supported to allow data ingest into the

database system. Once these data sets are part of the database, it will allow query by locality operations in addition to the traditional file oriented I/O. Our goal is to hide these database query operations from the user through higher level specifications of spart targets (see section 5.5.6). For example, isosurfaces can be easily generated by specifying a spart target with the isosurface threshold value as opposed to explicitly specifying a database operator to return those data points.

Characteristics in the REINAS Information Model

Parameter Types:

- Measurements (sensor output elements)
- Observations (environmental state elements from measurements)
- Synthetic Observations (state elements from data assimilations)
- Forecast Observations (state predictions from models)
- Status (state of system component configuration behavior)

Atomic Element Types:

- Scalars
- N-Tuples (aggregate of simple values representing one or more value domains)

Spatial Aggregates:

- Point State Vector (multiple parameters/elements at a point in space/time)
- Spatial Profile (single parameter, multiple elements along collection profile in 3D, same or different element times)
- Time Series (single parameter, multiple elements, ascending time, one or many locations for each element)
- 2D/3D Field (single parameter, multiple elements, single or multiple element times, multiple spatial locations in 2D plane or 3D volume)

Metadata Objects:

- Systems (e.g. platforms, instruments, sensors)
- Processes (e.g. algorithms, procedures)
- Parameters (e.g. ocean temperature in deg-centigrade, ocean current in u,v,w cm/sec)
- Products (e.g. CODAR current velocity vector field - 1 hour average)

| | |
|---|---|
| - Expeditions | - Experiments |
| - Scientists | - Engineers |
| - Malfunctions | - Events |
| - Data Sets | - Projects |
| - Features | - Phenomena |
| - Log | - Grid |
| - Node | - Packet |
| - Site | - Transect |
| - Data Stream | - Others |

Table 3.5: Characteristics to be Accomodated in the Information Model

# 4. REINAS Architecture

## 4.1 Overview

The basic services that REINAS provides to end users are data visualization, data management, data acquisition, and data transport. These services are provided by means of software and hardware organized according to the architecture described in this chapter. This architecture was derived from the user and system requirements discussed in the previous chapter, and is the basis for the system design that has started in Phase II of this project, and that is discussed in the next chapter. This architecture will evolve as we gain more understanding of user and system requirements, obtain research results, and gain operational experience.

REINAS uses a client-server architecture in which a collection of dedicated servers provides service to any node on the distributed system running REINAS client software. The term *server* is used in a generic sense – all of the servers are peer-to-peer and may provide and request services to or from any other server. In principle, one sufficiently powerful computer could host all of the REINAS software components, although its real-time ability would naturally be degraded. In practice, REINAS software will be located on physically dispersed computers. Assignment of REINAS software components to hardware components is a tuning, not architectural, consideration. The communication scheme used should make actual program location transparent with the exception of performance.

## 4.2 Functional Components

The prototype REINAS software architecture consists of the *Distributed Executive*, the *Device Control Manager*, the *User Interface Manager*, the *Data Manager*, and the *Toolkit Manager*.

### 4.2.1 Distributed Executive

The distributed executive provides overall control that coordinates the entire distributed system so as to present a single coherent environment.

The distributed executive provides global resource management and is the backbone within which other components of the system fit. It provides standard distributed system support (topology control, resource name space management, fault-tolerance, and communication among all system components) and links physically separate REINAS managers into a single logical entity.

The various components of the distributed executive run on all computers participating in a REINAS network, but DEX is never directly visible to users. As it is discussed subsequently, likely implementation technologies include OSF/DCE,TCL, Andrew file system, Kerberos, and X-windows.

### 4.2.2 Device Control Manager

The device control manager is the classical real-time component of the system. It is not called a "device manager" so as to avoid confusion with operating system components. Its main functions include:

- Interfacing with such real-world devices as dataloggers, which are themselves special purpose real-time systems
- Converting disparate physical input from potentially any I/O device into logical streams of data in a REINAS standard format
- Controlling local and remote instrumentation in real time
- Cooperating with subcomponents of the user interface manager to provide for real time displays.

The subcomponents of the device control manager include:

- Standard instrument interface drivers
- Real-Time local control kernel; these functions may be provided by an underlying kernel or OS
- Controller data manager subsystem
- Sensor Compression subsystem
- Sensor event analyzer
- Distributed Executive controller component
- Distributed Executive Internet interface

Currently, we envision the prototype as supporting bimodal operations in which real-time data are displayed directly as they are received in "real-time", while retrospective data are obtained from a database that has been designed to optimize retrieval queries. Specifically, although there may be in-memory data structures in which real-time display data are buffered, there is no special persistent real-time database. Real-time data are obtained from real-time data sources, not from a database. Input data in REINAS flows in logical data streams from the device control manager into the database managed by the data manager. If monitoring of the input data are in effect, the data are displayed "on-the-fly" via cooperation between components of the device control manager and the user interface manager.

### 4.2.3 User Interface Manager

The user interface manager provides "single command" activation of an integrated X-window application environment capable of exercising all REINAS functions. It supports both visualization and the routine user functions typical of any integrated application.

The user interface manager provides a visual, X-windows oriented environment for managing the various tasks (and associated windows) that a REINAS user may have active, and a user-friendly mechanism for managing long-term activities (such as establishing a named window that monitors an on-going experiment). The user interface manager will maintain a "directory" of the user's established windows. These windows can then rapidly be reactivated. It will also provide the support functions, such as a help system, that are expected to be in a modern environment, such as REINAS, and are not located in any other components.

Many of the subcomponents of the user interface manager will be REINAS compatible X-windows application programs. Such programs would include real-time display applications and visualization tools. Some of the visualization applications will use OpenGL. Likely implementation technologies will include TCL/TK, X-windows, and Kerberos. Visualization tools will run as products that are viewed in application subwindows.

### 4.2.4 Toolkit Manager

The toolkit manager contains the job management functionality of the user interface manager without any X-windows functionality or dependencies. The toolkit manager exists to provide long-term job management on compute servers lacking X-displays. It provides required job control to assure that correct input is processed and output delivered. Tasks such as long running models may be run in the "background" with the toolkit manager. Likely implementation technologies include TCL, and Andrew file system.

### 4.2.5 Data Manager

The data manager exports an API (Application Programming Interface) to the other REINAS components. This API supports all the database operations required by REINAS, including such functions as metadata management, data storage, and query support. The metadata schema constitutes the logical design of the REINAS database. At least one data manager in a REINAS network must be running for REINAS to function. In addition to managing the scientific database, the data manager may be used by other components of the system, for instance, to support the persistent windows managed by the user interface manager. Likely implementation technologies are Sybase or Postgres, and Kerberos.

The subcomponents of the data manager are

- Security manager
- Distributed system database
- Distributed database coherency manager
- Data import/export manager
- E-mail/bulletin board support

## 4.3 REINAS Servers

The software components described in the previous section will be distributed over four types of servers, namely:

- *User Server*: A computer running the REINAS User Interface Manager. This will typically be a Unix workstation running X-windows. It may be a special high-end graphics workstation.
- *Compute Server*: A computer running only the REINAS Toolkit Manager. Toolkit Manager functions will normally run on a Data Server or a Compute Server, but in some special cases such as compute-intensive modeling/graphics, a separate compute server may be used. We may use the UCSC MasPar (4096 processor SIMD) as a REINAS compute server.
- *Data Server*: A computer supplying REINAS Data Manager functionality via the Internet to other REINAS components. This will typically be a Unix workstation, but there is no requirement that this always be the case, although it is needed to run Sybase and AFS.

- *Control Server*: A PC dedicated to controlling one or more instruments via REINAS Device Control Manager software. Control Servers are low cost generic microcomputers. They may operate without a monitor, i.e., may be contained in a ruggedized "shoe box." Control Servers "glue" instruments to the Internet and can be considered Internet dataloggers, although they may be capable of much greater functionality than a conventional datalogger. The Control Server may run a real-time Unix, another PC OS, or may even run a stand-alone version of the Device Control Manager. In all cases it will run a version of the REINAS Device Control Manager.

In addition to the above servers, the REINAS architecture will include the instruments used to obtain meteorological data, and a multimedia network used to interconnect sensors, servers, and scientists together.

# 5. System Design

The preliminary system design, which we call Phase II design, is based on our analysis of user and system requirements, the software architecture described in the previous chapter, our analysis of available technology, and our selection of system components to date. This design will serve as a baseline for the detailed design and implementation activities of Phase III of this project.

The description of the hardware and software components of our Phase II design is organized in five groups:

- The instruments and weather stations used to acquire information
- The tools used to control the above instruments
- The tools used to visualize the information obtained from sensors and models
- The database system used to store and manage such information
- The distributed-system tools and networking used to distribute information

The Table 5.1 illustrates the technology we are currently using to implement the various REINAS architectural components.

## 5.1 Instruments and Weather Stations

The instruments considered in Phase II are surface meteorological stations, wind profiler radars, and the CODAR system. Surface meteorological (MET) stations represent an extremely important source of real-time weather information. A generic REINAS MET station has been implemented to measure the following five properties:

- Wind speed
- Wind direction
- Air temperature
- Relative humidity
- Barometric pressure

These five properties are measured with three research-quality instruments, detailed below:

- Wind Monitor: Wind speed and wind direction can be monitored using a single instrument known as a *wind monitor* which consists of a lightweight wind vane supporting a small propeller or anemometer. Rotation of the weather vane indicates wind direction while rotational velocity of the anemometer provides wind speed.

  Typical and recommended instruments are either the R. M. Young 05103 or 05305 Wind Monitors. Any wind monitor should have a threshold wind speed of no greater than 1 m/s.

- Air Temperature and Relative Humidity: Air temperature and relative humidity will usually be measured with a combined sensor. The sensor should be shielded and either naturally or motor aspirated to avoid the misleading effects of solar heating. In winds below 5 mph, a motor aspirated system is desirable. In all cases, typical errors in air temperature should be known and within 0.3 degrees Celsius.

  A typical instrument is the Vaisala HMP35C Temperature and Relative Humidity Probe.

- Barometer: Air pressure is measured with a standard barometer. The sensor should be accurate to within 0.5 mb and have a resolution of at least 0.05 mb to be considered research-quality. A typical instrument is the Vaisala PTA427 Barometric Pressure Sensor.

In addition to the instruments listed above, high quality solar irradiance sensors as well as precipitation gauges may also be present in future REINAS MET stations.

The ideal site for placing research-quality MET instrumentation is an open field far from any obstructions, at a height between three and ten meters. In particular, winds should be measured in open space where the distance between the anemometer and any obstruction is at least ten times the height of the obstruction above ground level as measured from the anemometer. Sites near tall trees or on top of a large building are to be avoided; if an

```
        INITIAL TECHNOLOGIES for use in REINAS

Distributed Executive:
        TCL
        Andrew file system
        Kerberos
        X Windows

Device Control Manager:
        metget
        DCE
        Operating system of choice for device controller (TBD)

User Interface Manager:
        xmet
        GL / forms
        OpenGL
        Windows NT for OpenGL?
        TCL/TK
        X windows

Toolkit Manager:
        xmet.server
        TCL
        Andrew file system

Data Manager:
        metget.server
        Sybase
        Postgres
        Kerberos

Network:
        TCP/IP protocol suite
        EIGRP
        Link-level protocols for various media
```

Table 5.1: List of Technologies Selected for Initial Use in REINAS

anemometer must be placed on top of a building, it should extend above the top of the building by one-half of the building's height. The following formula must then be applied to convert the velocity $v_h$ of wind measured at height $h$ to the velocity $v_{10}$ that would be measured at the standard height of 10 m.

$$v_h = v_{10} \cdot (0.233 + 0.656 \cdot \log_{10}(h + 4.75))$$

No compensation, however, can be made for placing temperature or humidity sensors on or near tall buildings.

If a MET station is to be placed on or near terrain with a mean slope greater than 5%, one should expect the terrain to modify the measured winds. Excessive winds travelling up the slope and periodic gustiness followed by relatively calm conditions will be prevalent.

### 5.1.1 Radar Wind Profiler

The Radar Wind Profiler currently uses the Doppler shift of a 400MHz Radar to measure wind velocity. This radar will be replaced in the Fall of 1993 by a 915 MHz radar. The system broadcasts three beams from which a vector is calculated to determine the actual speed and direction of the wind. By varying the power level the system can measure winds at variable heights, generally from 550m to a maximum of 10km, with a resolution of 250m. The 915MHz profiler utilizes five beams and will provide wind measurements from 60m to about 3000m at a vertical resolution of about 60m.

The 400 MHz system must be shut off for periods of 12 minutes when polar orbiting satellites pass overhead because the operating frequency of the profiler is close to rescue signals of downed air and sea craft. Sampling is done at 6 minute intervals – one minute of sampling in each beam direction. The data for each sample by itself is considered unreliable unless a consensus covering several samples, over a period of one hour, is obtained. The data file contains a header giving site location, time period for the data, and exact longitude and latitude coordinates.

Signal processing is used to develop wind vectors. Spectra of the In-phase and Quadrature channels are generated. Rather than send the 36 spectra across the phone line, local processing is done to glean the important data from each spectrum.

There are four statistical properties that are kept:

- Radial velocity
- Signal power
- Spectral width
- Noise level.

These need to be saved because some quality control procedures use them to objectively determine if the resultant wind vector can be believed. For example, if the wind is clearly erroneous, then usually one or more of these three values is vastly different from a neighboring wind vector that looks believable. Signal-to-noise ratio is commonly employed to do this.

There are about 30 Wind Profiler Radars in the US, including the 400 MHz and 915 MHz radars at Fort Ord, and the 400 MHz radar at Vandenburg AF base. The rest are scattered throughout the midwest and the east coast.

### 5.1.2 CODAR

Coastal Ocean Dynamics Applications Radar (CODAR) is a remote sensing system developed by the Wave Propagation Laboratory, which is part of the National Oceanic and Atmospheric Administration (NOAA). The basic principle of CODAR is that high frequency (HF) radar energy in the frequency band around 10 MHz is resonantly backscattered from the ocean surface by surface waves [Cro55]. Because the resonance is due to reflection from surface waves of a known wavelength (one half the wavelength of the incident radar wave), the phase speed of the reflecting ocean wave is known precisely. Reflected energy received back at the radar site is Doppler-shifted in frequency by an amount equal to the contributions from waves traveling toward and away from the radar plus the contribution from the background current field upon which the waves are traveling. Removal of the known wave contributions provides a remotely sensed measurement of the surface current [BEW77], [BLC85]. The depth extent of the measurement depends on the depth of current influence on the reflecting surface waves. The depth range is estimated to be confined to the upper 1m of the water column.

The surface current measurements from a single CODAR-equipped site provide estimates on a spatial grid with a resolution of about 2 km x 2 km and a range of 20 km to 60 km (depending on the radar configuration and on the noise level). These measurements only provide estimates in one dimension, however, along lines emanating from the radar. These estimates are sometimes called radial currents to indicate that the data is obtained along radial spokes extending outward from the radar. Radial current data from two CODAR sites is required to make estimates of vector surface currents at common measurement locations.

Since March 1992, NOAA has been operating two CODAR sites alongside Monterey Bay through a contract with Codar Ocean Sensors, Ltd., which has built and refined CODAR technology since it was originally developed within NOAA. One site has been physically located at the Monterey Bay Aquarium Research Institute (MBARI) facility in Moss Landing and the second site has been located at the Hopkins Marine Station in Pacific Grove. The combined data from these sites has provided near-surface current estimates for the southern portion of Monterey Bay. A nearly-continuous three-month period in March to May, 1992 was analyzed and described by Neal [Nea92]. He computed mean CODAR-derived currents as well as their daily variability. The CODAR-derived currents compared well with moored current observations from 15m depth for periods greater than one week. They also compared favorably for higher frequencies associated with known tidally driven fluctuations. The data rate for these measurements was three-hourly. It has since been upgraded to provide two-hourly current fields. This is the practical limit for data from these sites, since they are based on older CODAR technology.

A third CODAR site was established at the Long Marine Laboratory in Santa Cruz in January, 1993. The instrumentation for this site is also owned by NOAA but it is based on a state-of-the-art CODAR system, which is marketed by Codar Ocean Sensors, Ltd. under the name SeaSonde. The SeaSonde CODAR systems have a greater range than the earlier-generation systems (60 km) and they are capable of providing independent data every half hour.

The processing and data requirements for single-site and multiple-site CODAR data vary depending on the level of data reduction required. Each individual radar is controlled by a Macintosh-II computer, in the case of the SeaSonde or a PDP-11 computer, in the case of the older CODAR units. The raw spectral data returned to the radar can amount to tens of megabytes per day. In the normal operating mode, however, that data is reduced to radial

current estimates by the on-site computer. Part of the data reduction involves determination of the range and direction of the reflected signal. This is done by comparing spectral returns from three different antennas co-located at the site. The timing of the returns provides range information and the relative amplitudes of the in-phase and quadrature returns provides angle information [LB83]. The radial files produced on site are less than 10 kilobytes each and are produced, at most, every half hour. This radial data is the basic product of the CODAR sites. The radial data can be combined in different ways, depending on the number of overlapping estimates, to produce vector current maps.

Additional modern CODAR sites based on the SeaSonde technology are being set up around Monterey Bay. Stanford University installed a SeaSonde site 20 km south of Monterey Bay at the Granite Canyon marine station in May, 1993. The Naval Postgraduate School has purchased a SeaSonde system that will be installed at Pt. Pinos (Pacific Grove) in August, 1993. The Pt. Pinos site will, eventually, replace the older site at Hopkins Marine Station and the equipment from the Hopkins Marine Station site will be used to maintain a single, older-generation CODAR site at Moss Landing because spare parts are no longer available for those instruments. The Pt. Pinos site has the added benefit that it is not blocked by the Monterey Peninsula and should provide radial estimates that extend south of the Peninsula, which overlap with the data from the Granite Canyon site.

Real-time use of data from the CODAR network is hampered at the moment because an organized data collection, combination, and distribution system has not been set up to coordinate data from the five CODAR sites, which are, from north to south, at Santa Cruz, Moss Landing, Pacific Grove, Pt. Pinos, and Granite Canyon. Real-time use of the data is also hampered by the presence of missing, erratic or improbable vectors. REINAS can greatly assist the development and use of this new remote-sensing technology in two important ways: (1) setting up and maintaining the data collection and combination of data from the remote sites, including the establishment of a central Internet site from which other users could obtain the data and (2) providing new visualization options for the 2-D CODAR data that could show radial data together with vector data and ground truth data in order to highlight bad radar estimates. This second function, if combined with a parallel investigation into the characteristics of the raw spectral data, could lead to significant improvements of the CODAR algorithms used to produce the radial data estimates, thus making the entire system better-equipped for real-time applications.

### 5.1.3 Initial Sites for Meteorological Instruments

The following are the initial instrument sites of the prototype REINAS system:

- *UCSC REINAS Meteorological Station*: Currently located on the roof of the Applied Sciences building. It will be relocated to a more feasible location on the UCSC campus, at approximately 37.00 N, 122.05 W.
- *UCSC/Long Marine Lab MET Station*: Existing MET station located at Long Marine Lab on extreme northern-most coastline of Monterey Bay.
- *MBARI Moss Landing MET Station*: Acquired MET station to be placed on coastline at MBARI's Moss Landing Pt. Lobos support building.
- *MBARI OASIS M1-Buoy*: MBARI OASIS meteorological and oceanographic instrumentation mounted on a buoy and located at M1 site, approximately 36.75 N, 122.03 W.

- *MBARI OASIS M2-Buoy*: MBARI OASIS meteorological and oceanographic instrumentation mounted on a buoy and located at M2 site, approximately 36.70 N, 122.40 W.
- *Lockheed REINAS MET Station*: Acquired MET station to be located on the Lockheed Missile and Space Company testing grounds in the Santa Cruz Mountains north of Santa Cruz.
- *Fort Ord Wind Profiler Radar and Met Station*: The Wind profiler at Fort Ord currently uses a 400MHz radar and will use a 915 MHz radar in the Fall of 1993.

- *CODAR Network*: Data can be obtained via file transfers from the five-site CODAR network described in the previous section. The most likely candidate CODAR sites to connect to the REINAS network in order to obtain data directly are Moss Landing and Pt. Pinos.

## 5.2 Control Servers

The instruments described in the previous section are controlled by means of control servers.

MET instrumentation are converted to computer-readable form through the use of a *datalogger*. Such a device is simply a small microprocessor with multiple analog-to-digital converters and pulse counters attached. Most instruments produce analog voltages that can be converted to digital form with one of the datalogger's analog-to-digital converters. Wind speed is typically converted to digital form using a pulse counter. The typical datalogger is programmable and has a serial interface that can be used to download data to a personal computer or workstation. For the purpose of interfacing to a REINAS networked computer, a very primitive datalogger or a separate analog-to-digital input/output board can be used.

### 5.2.1 Current Control Server

Each Internet-accessible MET station is serviced by a `metget` program running on a local workstation or personal computer local to the station. The `metget` program essentially interfaces a MET station to the Internet or any network based on a TCP/IP protocol suite; it drives or probes the MET station over a serial interface as necessary and forwards each sample in real-time as a packet over the Internet to the `metget.server` program. A *sample* in this case is actually a collection of measurements from individual sensors on a MET station, *e.g.*: wind speed, wind direction, air temperature. The `metget.server` program collects each sample packet sent by `metget` programs and inserts the sample as an entry into a Sybase database table. Currently, each MET station is represented by a separate table. Each sample is time-stamped, and the time-stamp is used as the key identifier field for the table.

As their names imply, `metget` and `metget.server` operate under a client-server relationship. Although multiple MET stations require more than one instance of a `metget` program running concurrently on separate machines, these `metget` processes may communicate with either one or multiple instances of the `metget.server` program. The `metget` and `metget.server` programs are the current instances of the REINAS Device Control Manager software.

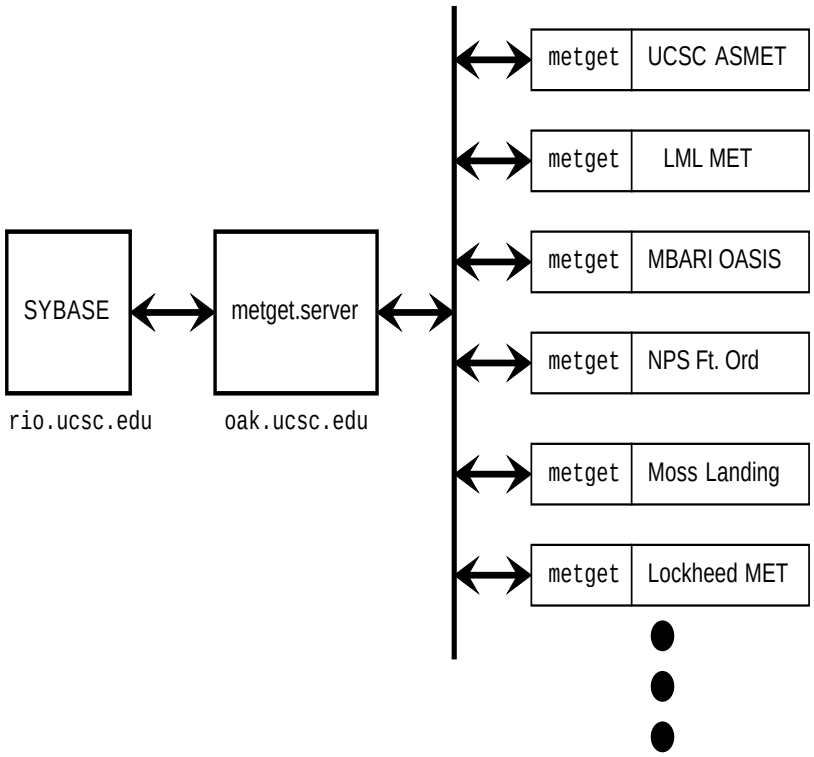


Figure 5.1: Functional Architecture of `Metget` System

As Figure 5.1 illustrates, instances of the `metget` program service six locations: the REINAS ASMET, Long Marine Lab, and MBARI OASIS buoy MET stations. A single `metget.server` program is used to collect samples and insert them into a Sybase database. A single `xmet.server` program retrieves data from the Sybase database in response to queries from a varying number of currently active `xmet` processes. The system architecture easily allows multiple versions of either server process to be used in order to alleviate concurrency problems on a single host.

### 5.2.2 Ongoing Technology Evaluation

As with many of the REINAS components, the operating system to be used in data loggers and control servers will be commercial off the shelf technology (COTS). In general, the data loggers are being provided or specified by the instrument manufacturers. On the other hand, REINAS control servers will be generic "commodity" computers that interface dataloggers or other data sources to the Internet. We are still evaluating different possibilities, because much activity has recently occurred in the market for Unix on PC architectures. We are evaluating the following Intel architecture PC Unix systems: Santa Cruz Operation's SCO Open Desktop, SunSoft's Solaris, Univel (Novell's) UnixWare, and Linux [GLOR91].

SCO Open Desktop is a mature Unix PC product, but it is unclear how well it provides real-time support. Both Solaris and UnixWare have gone to some length to extend Unix with real-time support. Linux is the most popular public domain PC Unix look-alike. With the exception of Linux all of the above are commercial products.

## 5.3 Visualization

### 5.3.1 Visualization Modes

We have identified different classes of users of REINAS with different visualization needs. As much as possible, we would like to provide the same graphical user interface, but customized for the class of visualization user. Our approach is to provide different modes of operating the visualization interface.

One mode might be a *monitoring mode* in which continuous readings of various instruments are available in either textual or graphical form. These readings are also co-located over the area of interest, in this case, Monterey Bay. Another mode might be an *analysis mode* where researchers can perform analyses of synoptic/historical data sets or numerical simulation data. In this mode, researchers can employ sparts as active agents to seek out and make visible features of interest in their data sets. A "now-casting" mode would support visualization of the results developed from the assimilation of measurement data and the output of models, which would together provide the best estimate of the state of the area being visualized.

### 5.3.2 Initial Visualization Capability

To provide a platform to demonstrate and help validate the feasibility of platforms and network-connection methods for the real-time collection, storing, and distribution of measurements from surface MET stations in the REINAS problem-domain area, the `xmet` program was developed to support monitoring visualization mode.

As its name implies, `xmet` is an X-Windows client that allows the user to monitor current or retrospective data from any MET station in the Monterey Bay area that is accessible via a TCP/IP connection. The system actually consists of a collection of separate programs that communicate via a client-server relationship. At the center of the system is a Sybase database.

The `xmet.server` program services an arbitrary number of client `xmet` processes. It responds to queries for data sent by instances of `xmet` by querying the Sybase database populated by `metget.server`. Each `xmet` process is an X-Windows program with a graphical user interface; by manipulating the program, the user can monitor either current measurements from any REINAS MET station in real-time or explore retrospective data over varying averaging intervals. Basic time-series plots allow the user to visualize the data in either mode.

Figure 5.2 illustrates the client-server model adopted for `xmet`, and Figure 5.3 shows a typical `xmet` display.

### 5.3.3 Visualization with "Smart Particles"

Our additional efforts on visualization have concentrated on implementing several precanned "smart particles," [PS93a], [PS93b] each one resulting in a different visualization technique. Ideally, we would like to reuse and extend as much of the existing commercial off-the-shelf software as possible. As a result, we have also been looking at various visualization platforms such as AVS and Explorer.

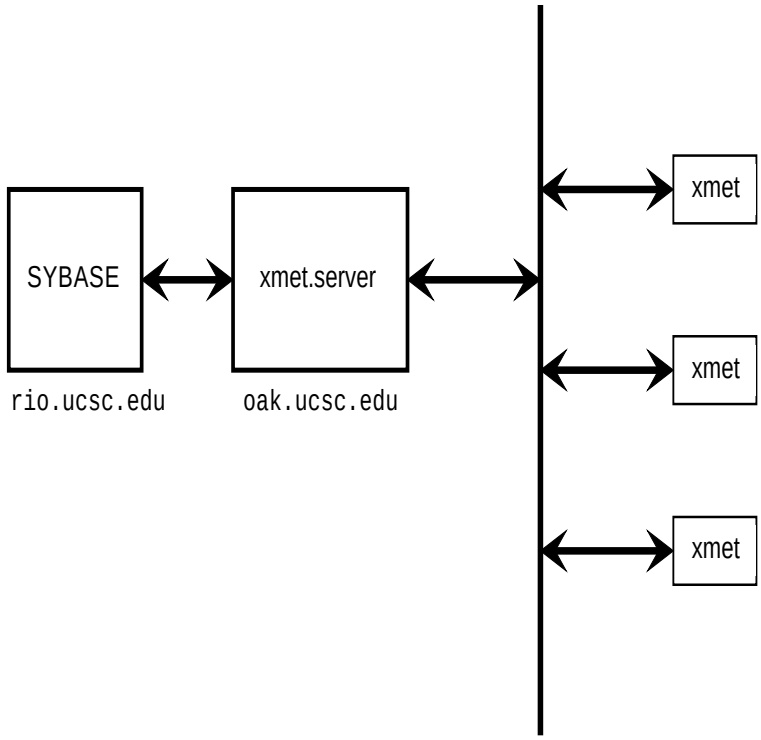


Figure 5.2: Functional Architecture of `Xmet` System

The primary REINAS query interface will be visual and will involve the user's specification (either explicitly or implicitly) of *sparts*. Sparts are "smart particles" which are conceptually user-specified teams of active agents capable of autonomously navigating the database and continuously taking actions, such as depositing tokens, which can be made visible by image rendering software. The user then sees the visual side effects of the spart team's interaction with the database. Since sparts are highly customizable with respect to both their interaction with the database and with respect to the tokens they "dribble" behind their path for the renderer, most visualization query functions can be performed with sparts. Sparts effectively are an indirect way to see the overall shape of items in a database. The user will have a large number of predefined sparts from which to choose and will be able to tailor these interactively. In addition, it will be straight forward for the user to code sparts in C or Fortran and have them become part of the REINAS environment.

### 5.3.4 What Will a Visualization User See?

The centerpiece of REINAS scientific visualization is provided by windows that contain *synoptic* views. These windows provide real-time 3-D views of experimental data in context. A typical experimenter might use two such views, with one containing a 3-D relief map of Monterey Bay in a transparent box, and the other providing a view into an artificial state space defined by the experiment. As data are collected, it is displayed within the two 3-D synoptic views. The standard synoptic window displayed by REINAS may consist of coastline and bathymetry data with overlaid displays of spart output reflecting CODAR, wind monitor, buoy data, etc.

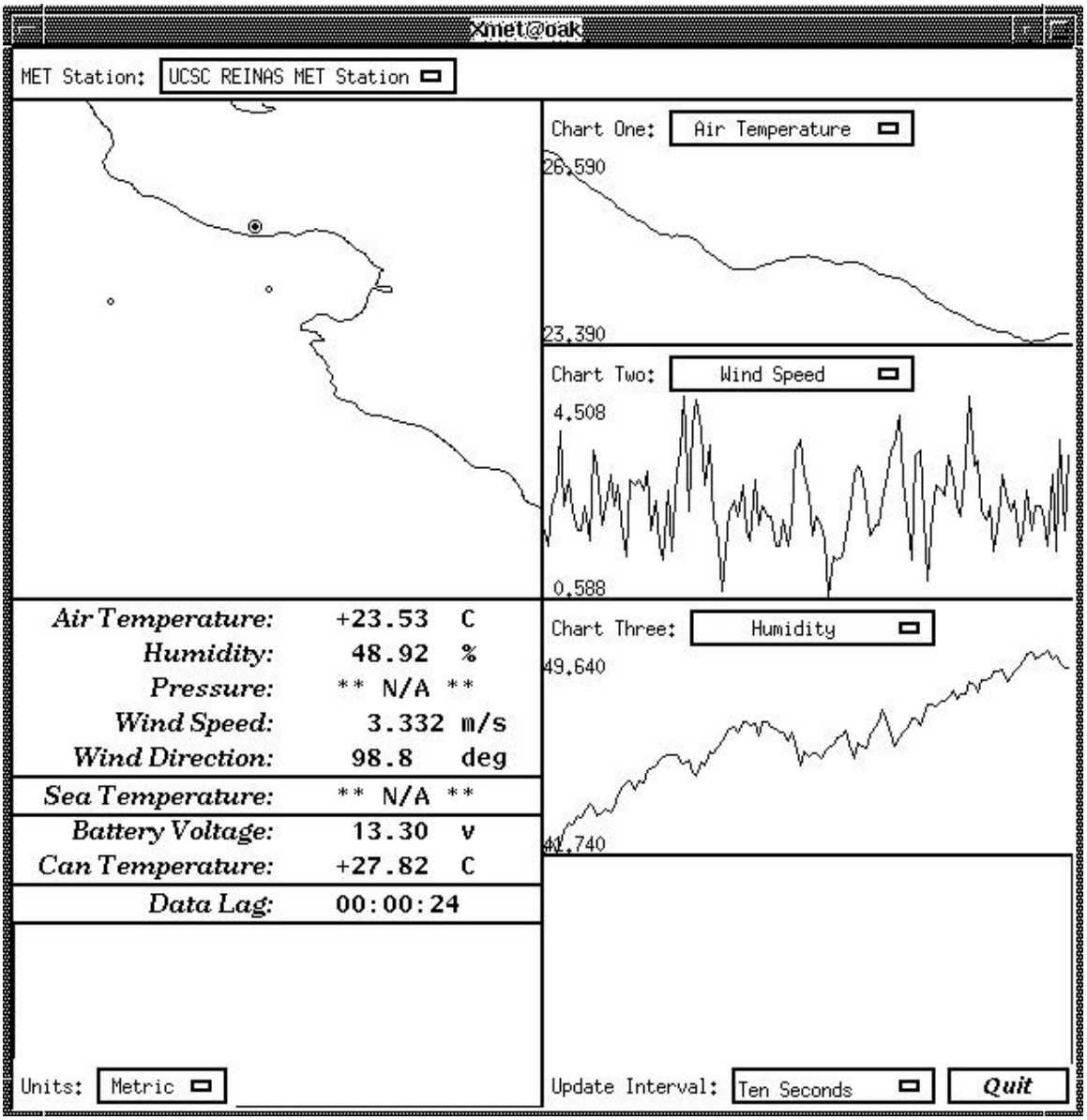


Figure 5.3: A Typical `xmet` Display

As a hypothetical example (this is not a specification), the default start-up view would be a "bird's eye" view (i.e., *synoptic view*) that consists of a window with a transparent 3-D box that contains a map of Monterey Bay. Visualization results would be displayed within this box. Then, depending on the default start up mode of the user class, either co-located measurement readings are overlaid on the map or a separate window with a shelf of pre-canned sparts are available. One of the attractive features of sparts is that it is very easy to customize sparts to suit a particular need. Thus, if none of the pre-canned sparts are satisfactory, one can simply mix and match spart targets and behaviors, thereby creating new sparts with different visual effects. In this manner, researchers are encouraged to explore their data set. A separate window where users can create new sparts from targets and behaviors will be provided. They will also have an option to can and save their own

favorite sparts.

One will note that there is little visual difference between the monitor mode and the analysis mode. Users who invoke the analysis mode simply have more power at their disposal. The needs of operational forecasters can also be met within the same scenario by providing standard forecasting displays directly, or as a pre-canned spart that will produce the display.

Using the same interface, other components of REINAS can also be accessed through a floating Toolbar icon. Selecting a Toolbar icon opens a window with the corresponding function. All REINAS functions would be available in this manner. For example, a user running an experiment might also display an *Active Instrument* window. This window would display a network diagram showing data streams coming from a number of instruments feeding an ongoing "experiment." The experiment would be shown as an icon within the *Active Instrument* window. A corresponding *Experiment Window* could be displayed by double clicking on the icon. This window would show active error checking and validation settings filtering the input data. Higher-order statistics could be displayed. This window could also be used to bring up a *Data Collection Window* that contains the relevant "data types" being collected by the experiment. In this window the user could filter data collection by indicating, for instance, that all data of a certain type (e.g., temperature) was to be discarded. These data types would be established by the instrumentation engineers and database administrator. In general, the scientific user would not have to perform database activities. Collected data would automatically be stored in the database.

### 5.3.5 A Typical Future Interactive Session

During a typical interactive session, the user would monitor device and instrument status, control data filtering, establish and monitor evolving real-time displays of variables of interest, and observe the evolution of phenomena within the synoptic displays. Often, one synoptic display would show the observed data superimposed on a map of Monterey Bay, and the other would focus on the evolution of the experiment's parameter state space. Once collected, all these functions could be performed against the database, i.e., the experiment could be replayed with different displays and sparts, thus providing another view of the phenomena.

Real-time access to synoptic windows will require support from a Silicon Graphics or other visualization server attached to REINAS. All other functions should require only typical X-workstation functionality. Synoptic window sessions can be canned and later replayed on typical X-workstations.

### 5.3.6 Off-the-Shelf Visualization Platforms

There are several off-the-shelf visualization platforms available. Some of them, AVS for instance, are particularly popular within the oceanographic and meteorological community [NOA93]. The two reasons most commonly cited are ease of use and extendibility. We would like to preserve these features. In addition, as much as possible, we would like to build from where these platforms left off. However, during our initial investigations, we noticed several differences between our visualization approach and existing ones.

There are similarities but also fundamental differences between the data flow pipeline used by platforms such as AVS [ea89] and Explorer [SGI91] and the concept of sending out

active agents into the data field. In the former case, data flows through a series of data transformation modules which eventually render the data. In the latter case, sparts are sent out to seek out specific targets in the database and highlight them in various manners. To a certain extent, a spart's targets are a set of features (such as a specific value) in the data set. Targets may also be more complex, corresponding to a form of data pre-processing. In this sense, the targets correspond to the transformation modules in the data flow paradigm. In fact, we have implemented some of the pre-canned sparts, such as surface-seeking and flow-tracking sparts, as user-specified modules for Explorer. However, one drawback of the data flow paradigm is that the entire chain of modules will have to refire if a data point upstream is changed. This can lead to quite inefficient handling of real-time data that the REINAS project needs to address. Thus, we have recently started implementing sparts as separate application.

Another difference between existing platforms and our approach is the granularity of the "modules". Modules in Explorer and AVS are complete routines which operate on the entire data passed to them. On the other hand, sparts are made up of a list of targets and a list of behaviors. Targets are feature sets that the spart is seeking in the data. Once the targets are found, they may be manifested in various ways, depending on what kind of behavior was prescribed for the spart. In addition, at any instant, a spart operates on a subset of data that is situated in its local neighborhood. This brings up two issues: (1) Sparts need to know the location of data points. Location can be 3D geometric information or they can be abstract 3D parameter space information. (2) Sparts can easily be made extensible. In fact, targets and behaviors are usually smaller and easier to write than modules. In addition, one can mix and match different targets and behaviors to create new sparts.

### 5.3.7 Graphics Library

Because of the limitations of off-the-shelf visualization platforms above, we plan to develop the visualization platform as a separate application rather than as modules to existing platforms. To help facilitate this task, we are using GL (Graphics Language) to develop our application. GL is currently the native graphics language on all SGI machines. We foresee that SGI will still be a key player in providing affordable visualization workstations in the next few years. For places without SGI machines, there are several companies such as Portable Graphics and DuPont Pixel which provide GL libraries to run on platforms such as Sun, Dec and HP workstations.

Another reason for using GL is that it will make our transition to OpenGL much easier. OpenGL is a software interface, consisting of several hundred procedures and functions, that allow programmers to specify objects and operations. It is hardware independent and is currently adopted by the major workstation vendors. Graphics performance will depend on how well the different OpenGL implementations take advantage of hardware graphics accelerators. OpenGL is also window independent and can therefore run on X-windows and Windows/NT. However, this also means that windowing operations such as opening a graphics window and mouse operations are not part of OpenGL. Thus, a window dependent interface will be necessary. It is anticipated that with GUI (Graphical User Interface) builders, this task is straightforward. Transition from GL to OpenGL can also be simplified with tools such as toOGL which translates GL graphics code to OpenGL code.

## 5.4 Database System

At the core of REINAS will be a distributed scientific database. Some typical problems faced by such a database are access to time series data, management of large objects, and support for complex queries involving proximity relationships. This section discusses three database systems and two distributed data/programming environments which that support a distributed scientific database.

*Sybase* is a commercial relational database product. It supports SQL, provides a strong relational database implementation, and is well supported and documented. Sybase provides us with a means of evaluating how well the relational database model supports a scientific database. We are routinely using Sybase as part of the REINAS development environment.

*Postgres* is a research database system from U.C. Berkeley that combines relational, object-oriented, and extensible database features. Postgres is stable for a research system, but it is not as robust as Sybase. Postgres performance on simple record operations is inferior to that of Sybase. However, new access mechanisms can readily be added to Postgres, for instance, B or R trees with compressed indices, skip lists, multidimensional grid structures, etc. Postgres also supports the concept of data "time-travel" where multiple versions of data items are kept. In addition, Postgres is currently being used as a research vehicle to support scientific databases of a similar nature to that required by REINAS. We have been experimenting with Postgres.

*Empress* is a commercial relational database product that is targeted at the scientific community. Empress is being used to implement the NEONS system at NRL/Monterey and is being investigated for use by Fleet Numeric Oceanography Center. Empress supports large objects. Empress does not have a query optimizer. Although we do not have Empress, its overall database functionality appears similar to that of Sybase.

We have been using data from our MET station as test input to both Sybase and Postgres. We are currently using Sybase to support the `xmet` program which displays this data.

### 5.4.1 Sybase

SYBASE is a commercial relational database system which employs the Structured Query Language (SQL) database interface.

Through experimentation, Sybase was found to meet and exceed the real-time requirements for storing data from a single MET station. A miniature REINAS-like system has been built around a Sybase database. The system includes a typical research MET station deployed on the UCSC campus and configured to output a complete local weather profile measurement each second. Each set of measurements is time-stamped and inserted into a Sybase database table. At this rate, approximately 5 MB of weather data are accumulated daily. Concurrently, a database application `xmet.server` acts as a *weather server* for client applications such as `xmet` and `met`.

The client applications can be run on any networked workstation and allow the user to monitor current weather conditions as reported by the UCSC REINAS MET station as well as MBARI's OASIS buoys located in Monterey Bay. Several client applications can be run simultaneously. Each client application communicates with the server application, requesting the current local weather data on a periodic basis. The server program responds by querying the Sybase database for the most current weather record and relaying the record

to the client applications. In this way, individuals as far away from Santa Cruz, California as Pittsburgh, Pennsylvania and Sydney, Australia have monitored local Monterey Bay weather in real time.

Throughout the development of this demonstration system, Sybase has proven capable of handling the real-time demands of concurrent insertions and queries. Ad-hoc query tests requiring extensive searching and disc access have shown that Sybase can perform more than 20 times faster than the Postgres database system (see 3.5.2).

### 5.4.2 Postgres Database Management System

The Postgres database system was developed under Prof. Michael Stonebraker at U.C. Berkeley as a successor to the Ingres relational database system [Sto91]. The stated goals were:

- Support complex objects, i.e., objects such as a drawing or a geographic region. Postgres objects support many object-oriented programming concepts, such as inheritance.
- Provide for extending the database with custom user-defined data types, operators, and access methods. This allows researchers to extend the basic system with low-level facilities that are tailored to special problem domains.
- Provide alerts and triggers that make the database "active" as opposed to passive, and support forward and backward inference chaining.
- Provide crash recovery and take advantage of new storage media, such as optical disks. By default, data in Postgres is never deleted or updated, rather successive "versions" of a data object are maintained. This permits all queries to be qualified by time, a facility known as "time travel."
- Minimize changes to the relational model while supporting the above.

Postgres thus provides an environment for implementing and testing new ideas in database research. Currently, Postgres is being used to support the Sequoia 2000 project. As part of this effort work is ongoing in implementing compression/decompression at the storage level and integrating compression with network communications so that data is not uncompressed until it arrives at the client workstation. Efforts are also underway to couple the GRASS GIS (Geographical Information System) with Postgres, to extend Postgres to handle GIS types of queries, and to interface Postgres with visualization tools such as IDL, AVS, and Khoros. To enhance Postgres performance, the Postgres database is being used as the physical bottom level storage manager, i.e., the database is not running on top of a file system but is managing disk space directly. An example query the Sequoia 2000 project intends to support is given as: "Select all day time AVHRR images for the Southern Sierra Nevada between October 1991 and June 1992 and sort chronologically."

For our purposes, Postgres has the advantage of being available as source code, supporting customization and database research at all levels, operating in the Internet environment in a distributed fashion (including across heterogeneous platforms), and in addition is being currently applied to a very related problem domain, i.e., the large scale global change research undertaken by the Sequoia 2000 effort. A commercial version of Postgres should soon be available. We are told that it is two to three times faster than the University Postgres.

## 5.5 Distributed System Tools

### 5.5.1 Overview

The integration of the various components of the REINAS system will be accomplished via the operating system and distributed development tools. One way of accomplishing this would be to use a distributed operating system geared towards the integration of heterogeneous environments and tools. However, our evaluation of distributed operating systems has not identified any systems sufficient to meet all the needs of the project. Therefore, we have studied various distributed tools that can be used to integrate widely distributed systems in addition to local operating systems. Requirements for a distributed system include security, fault-tolerance, transparency, scalability, and heterogeneity. This section discusses several tools that might aid in connecting the various components of the system together.

### 5.5.2 Operating Systems

The first technology evaluation and research was the search for appropriate operating systems. Operating systems are needed on several levels in the REINAS project. They can be local to the instruments and to the machines through which the information from the instruments will pass.

The Distributed Computing Environment (DCE) is a complete distributed system including a file system (which is just now becoming available), a remote procedure call package (RPC), threading, synchronization, a resource tracking database, and security. This set of features will be useful for distributed application development for the users as well as for integrating the components of the REINAS system. At this point, we are contemplating using the RPC package for communication across the system. The complexity of the package has led to some work designed to simplify its use. This work is also described below.

Since a necessary component of a distributed system is a distributed file system that allows transparent access to data distributed across several sites, we have studied possible file systems. The inability to acquire DFS (because it is still in development for most vendors), and the similarity of DFS to the Andrew File System, led us to evaluate the Andrew File System as a possible transitional tool for the integration of distributed data until DCE/OFS is available.

The *Kerberos* Security System [MNSS87] is a system that provides multiple levels of security for data resident in the system or being transmitted across the network. It provides both authentication and encryption for data, and the various combinations of these make up the varying levels of security provided by the system.

*ISIS* is a distributed programming environment [CB92]. The ISIS user can write reliable distributed programs which tolerate server failure and network partition. ISIS potentially could be used to write a distributed database server. ISIS appears to suffer from performance problems that make its use on the typical Internet workstation questionable.

*Prospero* is a means by which the users of a system can customize their view of the data distributed across a file system [Tha93]. It is based on the Virtual System Model, which allows the users to organize their data according to their personal preferences while controlling other user access to their data.

*Pegasus* is an ambitious research program intended to develop a distributed multimedia system [MLM92a]. Pegasus is representative of research efforts to develop large distributed multimedia databases. Few of these efforts have reached the stage where their results can be applied operationally.

### 5.5.3 PC UNIX Evaluation

The operating system requirements for user, data, and computer servers are simply defined as the ability to run current versions of Unix, be Internet compatible, and be compatible with and support any of the Internet or Unix dependent software identified as a REINAS requirement.

Unix is needed for DCE, AFS, X, TCL, and Kerberos. OS/2 may possibly be available later for DCE on micros. There are a great variety of Unixes available, and it runs on virtually every platform.

In addition to low-level OS requirements, REINAS requires high-level distributed protocol or "glue" to tie the various components of REINAS together. This level of an operating system is often described as the *Job Control* level. We are currently experimenting with the use of *Tool Control Language* (TCL), developed at UC Berkeley. TCL essentially allows very rich distributed scripts to be written in the X-windows environment. TCL is a complete shell, in addition to being "X aware." Potentially, many of the attributes and features of the windows displayed by a system using TCL can be defined by the TCL script and need not be specified by the application. This is somewhat reminiscent of the manner in which processes under Unix are unaware of file I/O redirection performed by normal shell scripts.

### 5.5.4 Distributed Computing Environment (DCE)

The Distributed Computing Environment (DCE) [Fou92] is a product of the Open Software Foundation (OSF) that provides a number of tools and services for developing distributed applications over heterogeneous networks. Currently DCE exists for DEC Ultrix, IBM AIX, OS/2 and HP platforms. SunOS, OSF/1 and SCO XENIX versions are slated for future release.

The components of DCE [Ope92] include:

- **Threads** – A user-level threads package provides for the maintenance and synchronization of multiple threads of control within a single process. All of the other DCE services depend, directly or indirectly, on the existence of threads.
- **Remote Procedure Call (RPC)** – An interface definition language, the associated compiler, and a runtime service program constitute the RPC facility. The tools are intended to support a client/server relationship over the network that appears identical to a local call environment. Authenticated RPC is provided by integration with the DCE Security Service.
- **Directory Service** – The directory service is a resource tracking database, typically used to store information about users, machines, and RPC services. There are several components of the directory service: the Cell Directory Service (CDS) manages information about a group of machines called a DCE cell; the Global Directory Service (GDS) provides standard directory services (CCITT X.500/ISO 9594); finally, the Global Directory Agent (GDA) is the intermediary between the local CDS and a global service, which may be GDS or the Domain Name Service (DNS). Both CDS and GDS are accessed via the X/Open Directory Service (XDS) API.

- **Distributed Time Service (DTS)** – DTS synchronizes the clocks of hosts participating in DCE according to the **Coordinated Universal Time** standard. It can accept an external time provider. The notion of time in DTS includes an inaccuracy factor, which means indeterminate results are not eliminated when DTS compares two times.
- **Security Service** – DCE security services are based on version 5 of MIT Project Athena's Kerberos [SNS88] system, although the Kerberos API is not available directly to the DCE programmer. Normally, security is handled transparently with Authenticated RPC, but interfaces to the Security service's components, the Registry, Access Control List and Login facilities, are provided.
- **Distributed File Service (DFS)** – DCE implements a complete distributed file service, including an optional physical file system called the Local File System, which allows for replication and logging, as well as the use of DCE Security facilities for access control. Caching and replication help to provide "high performance". Diskless workstation operation is supported. However, this component of DCE has only been released on some vendor's workstations.

In order to make RPC calls with the DCE RPC package, a programmer must: define a server interface, compile the interface definition so that it can be linked into the client and server applications, write the client and server applications, including code to interact with the DCE runtime services, and compile, link, and run the server and its clients.

Writing of an application involves using a special language called the Interface Definition Language (IDL) to specify all of the data types that will be shared between client and server, as well as the parameter types and return types of all of the server operations. IDL provides a C-like syntax for defining server interfaces, including function prototypes, data structure declarations and typedefs. Also included in the interface definition are a Universal Unique Identifier (UUID), and a version number, which are used to identify server interfaces when obtaining binding information from the directory service.

A file called the Attribute Configuration File (ACF) allows the programmer to modify the application's interaction with the stub code. Using the ACF, the programmer can choose from several methods of obtaining and using server bindings from CDS, indicate interest in specific error messages and specify where to store them, define user-written conversions between application and interface data types, and specify in-line or out-of-line marshaling of data.

Using the interface definition and the ACF, the IDL compiler generates object code for client and server stub routines that handle the format conversion and network transmission of function parameters and results. The compiler also produces a header file that contains type definitions and stub-function prototypes that are then included in the client source files.

DCE remote procedure call is extremely flexible, and consequently deviates substantially from a local call environment: one or more interface definition files must be written. In all but the simplest applications, clients use the DCE directory service to obtain information about servers, which requires some arcane initialization code. Also, handling exceptions and server failures is unwieldy.

Accordingly, we have undertaken the writing of a "stub compiler" that will accept annotated C code, and perform the above tasks for the user. So far, it performs as follows:

- The stub compiler scans the server code for functions declared with the *remote* keyword. IDL operation declarations are generated for them, and IDL type declarations are generated for each type used as a parameter or return value.
- The compiler removes the *remote* keyword from the server file(s), and inserts the code that exports the server interface to CDS
- The client code is also modified: `#include` directives are inserted that import the necessary DCE header files, as are some variable declarations and the code that imports server bindings from CDS.

As yet, the compiler handles scalar types, typedef names, and structures—the latter two only so long as they are not nested. Pointers will work correctly in some cases, if their use corresponds to the default DCE treatment.

Functionality still needed includes:

- An ACF file needs to be generated to indicate the kind of interaction with CDS desired by the user.
- The compiler should perform some dependency checking and compile any DCE files that need it.
- Clients should be able to import multiple server interfaces. Servers should also be able to import other server interfaces. It remains to be seen whether the latter is readily done without sacrificing the availability of threads in server programs.
- Pointers, and other data types that give DCE problems, should be handled or disallowed entirely in RPC calls.

DCE is a highly flexible distributed computing tool that could benefit the development of distributed applications in the REINAS project. It has many tools and features that would allow a wide variety of applications to be created for many purposes. The problem with DCE, as mentioned, is that its flexibility leads to complexity in its use, and this makes DCE more difficult to use. DCE is a sufficiently extensive system that with all of its features entails significant amounts of learning overhead. However, with some front-end interfaces that can make it easier to use, DCE is appropriate for many aspects of the REINAS project. The major drawback that we are currently facing, besides the complexity of the system, is the lack of the Distributed File System, without which, much of the DCE system is unable to function. The RPC package, however, will still be useful for application development with the front-end attached to it. The stub compiler will not restrict functionality, and it will make using the package much simpler.

### 5.5.5 Andrew File System (AFS)

Andrew was developed at Carnegie-Mellon University (CMU) (with funding from IBM) [Sat90]. Andrew was intended to be secure, scaleable to at least 5000 RT workstation class nodes, and replace the use of centralized mainframe timesharing in university environments. Development started around 1984 and continues to this day. An Andrew workstation is a "normal" workstation that has access to a large special directory subtree, */afs*. This directory is shared with all other Andrew workstations. Cooperating servers implement the distributed directory. The user need not be concerned with the implementation specifics of remote-file service. Andrew is strictly a distributed file system. No user applications can run on the Andrew servers. Security is enhanced since users are confined to executing

applications on their local workstation. This overall architecture resembles that used by Novell's Netware for PCs.

Andrew servers run VICE. The user workstation's run requestor software called VENUS. VENUS intercepts local file system requests and transparently maps them to Andrew requests. Pathname translation (locating files) is performed by VENUS using VICE services. Temporary files are always created on the local server. When a shared file is opened, it is copied to the "cache" (the local disk) of the local workstation. When the file is closed, it is copied back to the server if it has been modified. All file access between the open and close is to the local file copy.

The Andrew development team has left CMU to found Transarc for the purpose of bringing a commercialized version of Andrew to market. Andrew is often referred to as *AFS*. AFS is available from Transarc on a wide variety of Unix platforms: IBM AIX, Sun, Decstations, etc. Transarc has worked with the Open Software Foundation on DCE, and AFS form the basis for the DFS of DCE.

Although Andrew originally used it's own version of the Needham and Schroeder private key algorithm, they have recently adopted the MIT Kerberos authentication system. Access lists in Andrew apply to entire directories, not to specific files.

CMU is currently working on the CODA system. CODA is a descendent of AFS-2, and is intended to investigate high availability and disconnected operations (laptops). These are viewed as essentially the same problem. Note that Andrew is not particularly fault tolerant and is subject to single points of failure.

Andrew is well suited for student use. Students spend their time compiling many small files. In this environment, treating the local disk as a cache is a very good strategy – the students files will be located on the workstation for the duration of the session, after which they will be stored on the servers until the next session. In student workloads, sharing is minimal. If the architecture of REINAS can be placed into this model of interaction, Andrew should prove very effective. If shared file access is common, then use of Andrew is more questionable. Andrew is probably an effective way to provide access to all the support files that will be require by REINAS.

### 5.5.6 Kerberos Security System

One of the major issues relevant to the REINAS project is the problem of security. The large number and variety of potential users of the REINAS system, coupled with the private nature of much of the data, require security designed for a networked environment. One way of handling the problem would be to use an existing security environment for user authentication and data encryption. *Kerber2* is a third-party authentication system designed for open network computing environments. Its primary goals include creating a secure identification mechanism for both clients and servers, reliability of the authentication service, transparency, and scalability. It provides three levels of protection: authentication upon initial connection, per message authentication without encryption, and per message authentication with encryption.

The *Kerberos* system accomplishes these objectives with the following software components: the applications library, the encryption library, the database library, the administration programs, the administration server, the authentication server, the database propagation software, the user programs, and the applications.

The *Kerberos* system is a recommended tool for the REINAS project. It is designed to be integrated into an open system, and some of its components are replaceable. For instance, the database software is replaceable, and it can be customized to the environment that uses it. Further, *Kerberos* is a tested application that is already in operation at various sites. Integration of *Kerberos* into a system involves including *Kerberos* calls into existing applications as well as setting up the database. This would mean that use of the system would not be an extremely difficult problem. Also, the flexibility of the various levels of authentication available in *Kerberos* makes it an attractive tool for the REINAS system.

The major drawback of the *Kerberos* system with respect to REINAS is the problem of limited ticket lifetimes. Tickets are the means, in messages, by which authentication is carried out, but the lifetimes of these tickets are restricted to further tighten security. The problem is that operations can not simply start failing once the time limit runs out. Otherwise, critical data could be lost once the ticket expires. Since the lifetime of the ticket can be set by *Kerberos* administrator, this is not an overwhelming problem. However, it would be necessary to automatically regenerate tickets for services to guarantee that data are not lost, and that processes do not stop operating.

A second pair of problems with *Kerberos* is the problem of authentication forwarding if a user or service attempts to access another host from an initial host, and the problem of a server gaining access to protected information in another file server on behalf of the client. These are problems that have not been adequately handled in the *Kerberos* system. In a scalable distributed system this is an important problem, and we would need to handle it in the REINAS system.

### 5.5.7 ISIS Distributed Environment

ISIS is a distributed programming environment developed at Cornell University [Gro91]. ISIS is essentially a technology for *distributed control*. That is, the facilities ISIS provides are largely oriented towards controlling some underlying activity in a networked setting. ISIS can be viewed as a tool for monitoring a system or subsystem, reacting to events that affect or reconfigure it, and in general sensing conditions and coordinating a response in a fault-tolerant manner.

ISIS is implemented on top of UNIX and sits at the application level in UNIX. It provides primitives for reliable programming including process groups and ordered atomic multicast. On top of these primitives, ISIS provides a toolkit of solutions to common sub-problems in distributed computing such as distributed synchronization, resilient computation, logging and recovery. With the various tools in ISIS and the well-ordered environment that ISIS provides, writing dynamic and fault-tolerant distributed programs can be as simple as writing programs for a single central machine [MWC90]. Note that ISIS is basically a big-machine technology. It runs well on UNIX workstations and mainframes, whereas PC's are essentially ignored in ISIS.

The ISIS system addresses a broader and more complete distributed environment with job management and monitoring than other solutions such as reliable broadcast protocols, shared memory models, remote procedure calls (RPC), etc. Basically, ISIS is a subroutine package [MW91] that employs protocols built over UDP to ensure that messages will be delivered reliably and in order; it adds headers to messages and delays messages on arrival (if necessary). This reliable, consistent ordering of messages is called "Virtual Synchrony" because events such as multicast and detection of failures are atomic in a

virtually synchronous setting and appear to happen one-at-a-time in a consistent order at all sites. With this approach, one can often arrive at elegant, efficient solutions to problems that would be difficult to formulate – and extremely complex to implement – on a bare message-passing system.

ISIS has demonstrated itself as a fundamentally new way to design and program fault-tolerant distributed systems [Mar90], [MW90]. Unfortunately, the performance problem and overhead of using the ISIS package has negative impact on real-time systems. ISIS sits at the application level in UNIX and has to pay high performance penalty.

Another important limitation of the ISIS system is its lack of mechanisms for building very large networks (hundreds of nodes). ISIS assumes that all members of each group will cooperate to manage the group state or perform operation on behalf of clients. This is an appropriate model for achieving fault tolerance with small groups of 3 or 4 processes. However, as applications grow larger, the ISIS user has to employ ad-hoc hierarchical structuring mechanisms to circumvent this limitation.

Other limitations of ISIS include its weak security, poor modularity of the system and its lack of support for object-oriented programming and specification. Current ISIS is too monolithic to be used in a "mix-and-match" manner, and too large to run effectively on a small dedicated machine.

The conclusion is that we will not use ISIS to implement REINAS until some of the key issues such as real-time guarantees, scalability, and security, are resolved and performance of the ISIS application are improved to suit the real-time environment. However, many ISIS ideas and utilities for designing large reliable distributed system are definitely worth study. We will monitor closely the progress of the current ISIS related projects at Cornell, especially porting ISIS to Mach.

### 5.5.8 Prospero Distributed File System

Two of the major considerations in the REINAS project will be convenient and easy access to large volumes of information distributed across several sites, and user-level information organization. These issues create the need for an organizational tool that includes security mechanisms, customizations tools, and hierarchical design. Further, since this system will be used by at a very high level, the system must also be highly transparent as well as highly scalable.

The Prospero file system is a distributed system designed for local organization of information that is distributed across an Internet and for easy access to information at remote sites organized similarly (using Prospero) [Neu92]. It has been successfully used to organize information from Internet archive sites, including software releases, archives of Internet mailing lists, copies of technical reports, and conference papers. It has also been used as the preferred method of remote access to "archie", a database that maintains information about files available from key archive sites [Tha93].

The basis of Prospero is the Virtual System Model for organizing large systems. It has the following features: support for customizable name spaces, user-level name-space construction tools, synonym support, and name space resolution mechanisms. A virtual system is a particular user's view of the organization of the system. Users can map one file structure to another, completely different, view of the same structure with union-links and filters. A user can set up his or her own name space by creating a root directory with the commands provided by the system for virtual system manipulation. These commands are

similar to UNIX file and directory commands. Filters are programs attached to links that allows target directories to be altered, and union links are links to directories that allow the view of the directory to be changed. Note that links are insensitive to local file system and machine boundaries [NA93].

Prospero works in conjunction with other security packages to provide the security requirements for the system. It has a high-level security mechanism including authentication commands, and permission attributes associated with each object in a virtual system. Information about individual objects in the system are listed in access control lists. Frequently, the Kerberos [SNS88] authentication package is used as the underlying security mechanism for object protection.

With respect to REINAS, Prospero would be a valuable tool for enabling the scientists to organize and protect environmental data distributed across the system. Prospero allows scientists to organize data relevant to their research while protecting sensitive data and accessing public data relevant to their work. Its transparency will make it fairly easy for the scientists to organize their data using this system. Prospero's flexibility and its inherent scalability is necessary for the type of system that the REINAS project is meant to demonstrate. At present, a *guest* virtual system is set up for testing purposes. Certain software, particularly the most recent release of Kerberos, will be necessary to test the product's usefulness and performance thoroughly.

### 5.5.9 Pegasus Distributed Storage Service

The Pegasus project is a research project being conducted at the Universities of Twente and Cambridge. It officially began in September of 1992, and is scheduled to last for 3 years [MLM92b], [MLM92a]. Its primary goal is the creation of an easily-used distributed multimedia system that can scale to millions of nodes. Requirements for this service include arbitrary levels of reliability and availability (specifiable by the user) and the ability to store digital video and audio in real time.

The first part of the Pegasus project is to build a **distributed storage service** that can be expanded to a world-wide scale, can provide arbitrarily high levels of availability, and is capable of storing multimedia documents in real time. It will comprise three layers: the bottom layer for permanent storage of non-replicated objects of various types (including volatile RAM, magnetic disks, and disk jukeboxes), the middle layer for storage and management of objects, and the top layer for management of typed objects and interface management [BMvdV92]. The second part of the project includes building a **multimedia infrastructure** that allows activities such as video conferencing, multimedia document exchanges, etc.

The network will be an Asynchronous Transfer Mode (ATM) network, and will be the communication medium for the other components of the system. The multimedia terminals will be regular workstations running the Pegasus operating system. The devices for actually processing and receiving specialized data (such as cameras, video units, sound systems, etc.) will be connected to the system via the network, instead of directly to the workstations. They will be connected to the terminals with desk-area networks (DAN) centered around ATM switches. The multimedia servers will handle the details that will allow interactive processing of multimedia streams.

A subproject of Pegasus is the design of distributed complex object service (the storage architecture) [SKB93]. The file server serves the purpose of providing file and directory

storage. It will be a self-contained and invisible (to the users) part of the system that runs a specialized operating system and user-level code. The purpose of the distributed storage service is to provide a complex-object service, a global name service that can be used to name everything in the system, and a service for storing digital data (particularly video and audio) in real time.

Multimedia documents are complex objects having many components including video, audio, text, graphics, and animation structures. Another subproject of Pegasus is to create an environment that allows convenient manipulation of multimedia documents. One of the primary goals of the project is to make manipulation of multimedia objects nearly as easy as normal programming.

A subproject of Pegasus is to create an *Automated Digital TV Director* that selects between various cameras and microphones based on processing of the incoming audio and video streams. This application will allow conferences in separate rooms to be monitored and transmitted automatically. One goal is to automatically decide which camera should have its data displayed, depending on who is speaking.

Since the Pegasus system is a project that is being conducted concurrently with our project, it will not provide us with a usable product in sufficient time to aid the REINAS project. However, many of the ideas being explored in the Pegasus project are relevant to REINAS. The goals of the storage architecture are parallel to the needs of the REINAS system. In particular, the high level services, the file system, and the archive server protocols should be worth investigating as they develop.

## 5.6 Networking Support

Communication among REINAS components and users will be accomplished via a multimedia networking infrastructure that encompasses new and existing telephone lines, Internet connections, and radio links, as well as the networking software and hardware needed to control and manage the interconnection of REINAS sites. Satellite links could be incorporated in the future.

### 5.6.1 Initial Network Configuration

During Phase II and the beginning of Phase III, the REINAS network has been limited to extending the connectivity that is already in place among the REINAS sites. The initial topology of the REINAS network consists of seven immediate sites that will be interconnected by October, and two additional sites that are very likely to be interconnected. The initial REINAS sites have been selected based on our study of the user requirements and the availability of instruments. These sites are:

- UCSC, which has a REINAS MET station that is currently located on the roof of the Applied Sciences building
- UCSC Long Marine Lab (LML), which has a MET station
- MBARI Moss Landing, which has a MET station
- The MBARI OASIS M1 and M2-Buoys
- Lockheed, which will have a REINAS MET station by September
- Elkhorn Slough, which will have a REINAS MET station via M/SC net by September
- The NPS MET station

- The Fort Ord vertical profiler and MET station

To demonstrate the management and visualization of data from as many MET stations as possible early in Phase III, an analog phone line will be established between the Lockheed site and the UCSC site. In addition, two low-speed radio links (9600 bps) are being established; one between the MBARI Moss Landing site and the Elkhorn Slough site, and another one between NPS and Ford Ord.

The purpose of the low-speed radio links is to test the feasibility of acquiring data from remote sensors over low-speed radio links as part of the multimedia REINAS network.

### 5.6.2 Mutimedia Network for REINAS

As we have stated, the initial network support for REINAS is based on the use of existing Internet connectivity with a few additional telephone and radio links. However, in the future, REINAS will include several mobile sites (e.g., boats with MET stations) and applications that require the transport of large amounts of information, specially for remote visualizations, where one minute animation can require in the order of 160 Mbps. The information exchanged in REINAS will include multiple media (text, voice, images, graphics and animation, and even video), and such information has to be distributed in real time (e.g., during a multimedia conference among multiple sites) over different types of transmission media, including radio links and high-speed lines. Furthermore, the networking infrastructure of REINAS should allow a potentially large number of sensors to be incorporated into the system. Accordingly, we see six major networking requirements in REINAS: the ability to transport multimedia data in real time, scalability to a large number of geographically-dispersed sensors, mobility of sites, fault tolerance, efficient use of multiple transmission media, and connectivity to the Internet.

Today's internetwork technology is oriented toward the interconnection of desktop computers and large servers or mainframes in very stable operational environments whose topologies rarely change. Existing wide-area packet switching networks have bandwidths that range from 56 Kbps as in MILNET to 1.5 Mbps (T1) as in the NSFNET backbone, which is being upgraded to T3 rates (45 Mbps). These networks are being used mostly for such non-real time applications as electronic mail and file transfers of relatively small sizes (i.e., hundreds of thousands of bytes). On the other hand, existing wireless networks are optimized to specific services (i.e., narrowband circuit-switched voice or packet-switched data).

The marked differences between REINAS networking needs and traditional networking technology indicated the need for new multimedia networking solutions to REINAS's unique characteristics and to allow REINAS system engineers to better manage and monitor communication resources in support of data management, data visualization, and user communication.

Translating the design of a new multimedia network design into a working prototype requires the development of new algorithms and protocols to control the network, to select routes, to adaptively manage links, to control traffic flows and congestion, and to manage communication resources to meet user and system requirements for data gathering, management, and visualization. The rest of this section provides our current assessment of existing technology, addresses key problems in the development of the required network protocols, and outlines promising approaches for their solution.

An early design choice for the REINAS multimedia network is the adoption of the TCP/IP protocol suite. This choice is based on the functionality provided by the protocol suite, the choices of networking equipment available, and the need to develop new network protocols.

The main research and development issues related to multimedia networking in REINAS are:

- Scaling to large numbers of nodes
- Use of different types of media for fault-tolerance and efficiency
- Support of multimedia distributed applications and collaboration in real time, requiring throughputs of hundreds of megabits per second
- Support of mobile nodes

### 5.6.3 Integration of Multiple Transmission Media

In the design of the REINAS multimedia network, we relax the distinction between links, networks, and internetworks used to provide connectivity, so that the various transmission systems are treated in a uniform manner, leading to a fault-tolerant multimedia system. The provision for type-of-service routing and load-sharing via multiple types of transmission media is a key component of our design. A communication security architecture could be integrated into such a communication system. This design is based on earlier work by Mathis and Garcia-Luna on survivable multimedia networks [MGLA87].

A major issue in this design is how to organize a heterogeneous mix of communications resources into a fault-tolerant system. The effort is compounded because some of the links are dedicated, others are multiple access channels, and some are actually networks. Assets such as links, networks, and other internetwork systems are treated simply as communication mechanisms to transport data between REINAS routing nodes, which we call *multiband routers*.

To provide for a uniform architectural treatment of the various types of media and networks, we adopt a generic classification of transmission media divided into four groups [MGLA87]:

- Dedicated point-to-point circuits such as radio links.
- Switched point-to-point circuits such as telephone circuits.
- Addressed multidrop or multinode systems such as BARRNET
- Broadcast systems such as multiaccess satellite channels

This classification is based on both the sharing aspects of the media and the switching or addressing aspects. A *link* between multiband routers could be a point-to-point link (e.g., leased lines), a switched link (e.g., dial-up lines), a broadcast system (e.g., various type of multi-point radio), or an addressed system (e.g., a network). This general treatment of the transmission resources provides a basis for real-time communication using any available means.

Many multiband routers will be redundantly connected via multiple parallel links. Accordingly, the concept of a *path* between multiband routers is introduced to symbolize that fact. Thus some control traffic (such as route updates) needs to traverse only a *path* between multiband routers and not every *link*; of course, link status probes would still be carried over each link.

### 5.6.4 Designing for Scalability and Fault Tolerance

A critical element in the provision of fault tolerance and the ability of a network to scale is the choice of the routing protocol used.

For the purposes of routing protocols, an internetwork can be viewed as consisting of a collection of interconnected domains, where each domain is a collection of such resources as networks, routers, and hosts, under the control of a single administration. Current work in interdomain routing has proceeded in two main directions: protocols based on distance-vector algorithms (DVA), which we call distance-vector protocols, characterized by BGP [LR91] and IDRP [ISO91], and protocols based on link-state algorithms (LSA), which we call link-state protocols, characterized by the inter-domain policy routing (IDPR) architecture [Ste92]. The same two basic approaches have been used in the Internet for intradomain routing (e.g., RIP [Hed88] and Cisco's IGRP [Bos92] are based on distance vectors, and ISO IS-IS [ISO89] and OSPF [Col89] are based on link states). We view REINAS as a single domain, and focus on intra-domain routing protocols.

The key advantage of DVAs, and the distance-vector protocols that use them, is that they scale well for a given combination of services. Because route computation is done distributedly, DVAs are ideal to support the aggregation of destinations to reduce communication, processing, or storage overhead [GLA88]. However, although Garcia-Luna and others have proposed DVAs that eliminate the looping problems of old distance-vector protocols like EGP and RIP [GLA93], an inherent limitation of using distance vectors is that routers exchange information regarding path characteristics, not link or node characteristics. Because of this, the storage and communication requirements of *any* DVA grows proportionally to the number of combinations of service types or policies [Jaf84]; therefore, supporting many types of services and policies using any DVA is inherently complex.

Because of the failure in the past to overcome the looping problems of early distance-vector protocols (RIP and EGP in particular), using link states has been considered to be the main practical alternative to internet routing. However, a key disadvantage of today's link-state protocols is that they require routers to broadcast complete topology information by flooding. As pointed out by Estrin and others [ERH92], this approach does not scale well. The main scaling problems of today's link-state protocols are three: flooding consumes excessive communication resources, requiring each router to compute routes using the same topology database at every router consumes excessive processing resources (e.g., see the results shown in [ZGLA92]), and communicating complete topology information is unnecessary if a subset of links in the network is not used in the routes favored by routers. As a concrete example of the scaling problems of link-state protocols, Garcia-Luna and Zaumen [GLAZ92] have shown that DUAL [GLA93] performs more efficiently than OSPF even in a relatively small network of the size of ESNET, even when OSPF areas and OSPF masks are used in both DUAL and OSPF.

The key concerns regarding the choice of a routing protocol for REINAS are scalability, efficient use of multiple transmission media, and support of real-time multimedia applications. However, as the previous paragraphs describe, the algorithms used for distributed route computation in today's internet routing protocols have severe scaling problems. On the one hand, DVAs need to communicate routing information among routers on a per path basis, which leads to a combinatorial explosion of service types and policies. On the other hand, LSAs require the same topology information to be replicated at all routers, which consumes excessive communication and processing resources in very large internets. An additional problem with today's internet routing protocols is that they are all based on

shortest-path algorithms running over a simple graph. In the future REINAS multimedia network, two routers may be connected to each other through more than one link.

Surprisingly, although the inherent limitations of LSAs and DVAs are well known, all of the existing internet routing protocols are based on these two types of algorithms, and the current proposals for interdomain routing in large internets [Chi91], [ERH92], [Ste92] either are based on LSAs and DVAs or leave distributed route computation as open problem.

Because of the multiband nature of the REINAS multimedia network and its expected complex connectivity, choosing to send a packet over a given path on the basis of a simple cost metric (e.g., number of hops that the packet will traverse, or the delay in the path) is not always sufficient. Furthermore, because the routing protocols implemented to date maintain a single shortest path between each source-destination pair of nodes, the throughput and delay over the chosen paths may be suboptimal. In the REINAS multimedia network, multiple paths for each source-destination pair should exist so that routing of packets can be more stable with respect to the traffic load changes.

New approaches are needed to solve the inherent limitations of today's internet routing technology, which dates to the design of the ARPANET routing protocols.

During Phase III, we will study and develop new approaches for scalable routing in the REINAS multimedia network. As a baseline for our work, we will first extend the initial REINAS network with Cisco routers. The reason for using Cisco routers in the REINAS network is twofold. On the one hand, Cisco recently introduced an improved routing protocol (EIGRP), based on DUAL [GLA93], which promises to address a number of REINAS networking issues. On the other hand, we are obtaining the Cisco routers on loan from Cisco at no cost to the REINAS project, which makes such routers an ideal baseline to start our performance studies on routing protocols. Based on previous analysis by Garcia-Luna and Zaumen [GLAZ92], EIGRP appears to be a better approach to scalable routing than the routing protocols supported by any other vendor. We will run performance tests on EIGRP and use those results to guide our research.

An important aspect of our research is extending today's routing protocols to the case in which routers are linked by multiple transmission media. An implementation of a "PC router" appears to be the logical step to implement our multimedia routing approaches for REINAS.

### 5.6.5 Access and Use of Wireless Networks

The main challenges we intend to address regarding the wireless component of the REINAS multimedia network are:

- The need to conserve power in remote sensors
- The need to support multimedia traffic over multihop radio networks that need distributed control
- The need to support mobility of nodes (e.g., boats or terrestrial vehicles with portable MET stations)

The protocols that need to be analyzed to address these issues are channel-access protocols and routing protocols aimed at wireless environments.

### 5.6.6 Channel Access Protocols

Power usage limitations dictate a minimum number of transmitted packets. The power usage limitations mean that the receiver of a low-power sensor cannot be fully on all of the time. This restriction is imposed by the high current drain (80 to 300 mA) in today's reduced-size receivers. In contrast, current channel-access algorithms require full-time two-way connectivity for transmission and acknowledgment. Even protocols designed for special conditions such as EMCON (emission control) require the receivers to be on throughout the period when communication is expected. For traffic from the sensor to the station that monitors it, power management is not a problem. The sensor itself is aware when a packet is ready for transmission and can wake up the transceiver section; hence, normal demand access algorithms or polling algorithms can be used. The situation is very different for traffic from stations to sensors. Assuming an information bit rate of 256 kbps in the radio channel and that each sensor sends packets of 100 bytes (including headers), a packet transmittal time is 2.4ms and polling 1000 sensors requires about 4.56 seconds, assuming no collisions and about 30% framing overhead. Collisions and retransmissions are likely to occur, and radios with much lower bit rates may have to be used just for economical reasons. Accordingly, it appears that strict polling schemes are not likely to succeed with large numbers of sensors sharing a common radio channel.

There are many ways in which channel access protocols for wireless networks can be classified. For our purposes, it suffices to classify them into contention-based and contention-free protocols. In either type of protocols, dynamic allocation of the channel capacity is necessary to cope with large numbers of users.

Contention-based protocols like the tree algorithms, CSMA and ALOHA [RS90] are perhaps the most popular channel access protocols today. In CSMA/CD, which is used in the Ethernet standard, a station is allowed to transmit when it detects no traffic in the channel. A station listens to the channel while it transmits its own data; therefore, if more than one station attempts to transmit at the same time, the stations detect the collision of their data and they cease to transmit. On the other hand, if the signal of the data packet transmitted by a station is able to propagate throughout the entire channel (a cable in Ethernet) before another station attempts to transmit while perceiving the channel as idle, the transmission is successful. Therefore, the first portion of the data packets sent by a station can be viewed as the station's reservation for the channel. Rom [Rom86] has proposed a collision detection mechanism for radio channels. The key disadvantage that we see with existing contention-based algorithms is that they require a station to contend for the shared resource every time a user has a data packet to send. Transporting isochronous media requires performance guarantees similar to those achieved with contention-free algorithms.

Contention-free algorithms can be based on either reservations or token passing. Existing reservation algorithms break the channel into a reservation interval and a data interval. Stations trying to send data over the channel attempt to make a reservation or acquire the channel control token during the reservation interval. A station sends data during the data interval if it is successful in making a reservation or receiving the token. As Rom and Sidi point out [RS90], these algorithms entail reaching an agreement on which stations need the channel and apply an arbitration scheme to decide which station should get access to the channel. Such a scheme is a priority structure imposed on th set of users, which are priority classes themselves. The Broadcasting Recognition Access Method (BRAM) [CFL79] and the Mini Slotted Alternating Priority Protocol (MSAP) [KS80] are well-known examples of this type of protocols. A limitation with current contention-free channel access protocols

is that they use either a separate control channel, centralized control by a base station (as in Goodman's packet reservation multiple access (PRMA) [Goo90]), or a fixed number of reservation slots, which makes protocols such as MSAP applicable only in small user populations.

During Phase III of this project, we will investigate whether a channel access protocol can be designed that provides performance guarantees like contention-free protocols, without their limitations. A promising approach consists of applying mutual exclusion algorithms based on elections.

### 5.6.7 Routing in Wireless Networks

The dynamic determination of optimum routes between nodes is fundamental in the operation of multihop packet-radio networks, but becomes very costly when several mobile nodes are involved. A number of routing schemes have been proposed in the past to cope with this problem, and can be classified as centralized, distributed, and hierarchical. In the centralized scheme, also called station-mode routing, a single node (called the station) ascertains the best path between each pair of nodes and upon request sends the requisite routing information to nodes in the network. This routing strategy would be unacceptable in REINAS, because of the need to cover geographical areas that may not provide line of sight from every mobile node to the same central station. In a packet-radio network using the fully distributed scheme (also called stationless mode), all the nodes participate as peers in the same distributed algorithm to determine dynamically the best path to every node. Finally, a number of hierarchical routing schemes have been proposed for the management of routing information in large packet-radio networks. The main idea of such schemes is to allow each node to maintain exact routing information regarding nodes very close to it, and less detailed information regarding nodes farther away from it. The objective of doing so is to obtain a reasonable compromise among the size of routing tables, number of updates required to maintain such tables, and the speed with which updates are propagated. For REINAS, we opt for a fully distributed approach, because it can be applied to any type of topography and node mobility, provided that the routing protocol used in the network does not consume too much bandwidth and is able to deliver data packets while nodes move.

Little progress has been made over the past ten years or so in the provision of dynamic routing protocols for radio networks with mobile nodes. Such techniques for multipath routing in radio networks as those proposed by Shacham, et al [SCP83] and Davies and Davies [DD87] have been improved little if any. A major problem with the current approaches to routing in dynamic radio networks is the reliance on shortest-path algorithms for the computation of paths from source to destination.

Garcia-Luna and Zaumen [GLAZ93] have recently proposed the first loop-free algorithm that provides multiple paths from any source to any destination in a dynamic topology. The collection of all the loop-free paths from a given source to a destination implied by the routing tables at the network nodes is called the "shortest multipath" from the source to the destination. A node in that multipath can forward packets to a destination through any of its neighbor in the same multipath, without the source having to specify entire paths or establishing connections, and without creating a routing loop (which can occur in duct routing, creating additional congestion).

We propose to augment the shortest-multipath algorithms developed by Garcia-Luna and Zaumen by using GPS data to aid nodes in the rerouting of data packets aimed at

mobile nodes, and to reduce the overhead of the routing algorithm. According to the proposed approach, a routing table entry for a given destination contains the network distance to the destination, a list of feasible neighbors in the multipath to the destination, and the coordinates of the destination. This implies that each node communicates its GPS to its neighbors when it transmits routing table updates. Based on this information, a node forwards packets based on the multipath toward the given coordinates of the destination, that is, a packet specifies the destination and its coordinates. When a destination moves, it sends GPS updates, which are processed by the nodes able to listen the destination's transmission. When a node receives a packet for a given destination that contains old GPS information, the coordinates are update and the packet is rerouted. We will investigate how to define geographical areas where destinations can roam without incurring routing table updates, in order to reduce the frequency of routing-table updates. This problem is very similar to the problem of hierarchical routing; however, the areas have to be defined dynamically, based on the location of mobile nodes.

### 5.6.8 Multimedia Collaboration in REINAS

Our view of multimedia collaboration in REINAS is one in which a researcher is emerged in a multimedia environment that is supported on-line and integrated into a distributed computing system. A scientist's window into the REINAS-supported experiments is through a workstation that supports all the media used by the experiments. The researcher sees (both video and images), hears, and reads what other researchers see, hear, and read. The system supports full interactive participation among scientists. Furthermore, the system supports full interaction between users and on-line and support functions such as laboratory facilities, library facilities and specialized computing resources.

Realizing this vision requires the development and use of a multimedia collaboration software environment designed to support collaboration among multiple users of existing tools with minimal intrusion of existing software or user interaction styles. This environment should integrate voice with other media exchanged in a multimedia conference, and separate private workspaces from shared ones

In Phase III, we will address the development of a multimedia collaboration environment for REINAS by evaluating existing prototypes (AT&T's RAPPORT system and SRI's CECED system are our current main candidates). Based on this evaluation, we will deploy a multimedia collaboration environment in REINAS to run experiments with REINAS users. The integration of the multimedia collaboration environment with the visualization software tools is particularly important in this phase. In addition, we will investigate the following issues concerning multimedia collaboration in REINAS.

**Sharing resources in real time:** A major issue in a real-time multimedia collaboration session is how to control access to shared resources, and how to control who has the floor in a conversation. Providing this support requires the use of concurrency-control algorithms.

Traditional concurrency-control algorithms developed for database systems are based on conflict (or contention) avoidance, and must spend a substantial amount of time ensuring that no conflict will occur among the operations executed against the shared database.

On the other hand, contention-resolution algorithms applicable to real-time multimedia conferences are very primitive. According to this approach, processes gain access rights to resources through contention. There are two algorithms reported to date. The Colab

system, developed by Stefik and his colleagues, uses a "cooperative" concurrency control model [ea87]. According to this model, each process sharing a resource has a copy of it locally. Changes made to the shared resource by any process are broadcast to all processes without any synchronization. Human users use verbal cues (the equivalent of voice locks) to mitigate race conditions. Garcia-Luna-Aceves, Craighill, and Lang proposed a distributed algorithm based on collision sensing [GLACL89]. This algorithm forces each process sharing a resource to assign priority to activity perceived over the network. Whenever a process receives data over the network, it assigns the conference floor to the sender of the data and blocks data originating locally. On the other hand, if a machine perceives that no data are being received over the network and has local data to send, it assumes that the local conference participant can have the conference floor and starts transmitting. If more than one participant starts sending data at the same time, more than one machine detect contention after receiving data from one another. When that happens, the processes that assumed having the conference floor relinquish it, stop sending data, and leave all participants free to regain the floor again. Contention for the floor is resolved through the randomness of the floor capture attempts or active human cooperation (e.g., establishing the equivalent of voice locks using a separate voice channel to request everyone to back off). During Phase III, we will study new contention-resolution algorithms aimed at real-time communication.

**Synchronization of multiple media in real time:** Multimedia collaboration can generate large amounts of data consisting of an heterogeneous mix of video, voice, graphics, and text, which differ in their traffic parameters (volume, stream, bursty), and the workstations receiving such real-time multimedia must maintain the proper synchronization among all media.

**Multiparty connections:** Data exchange will occur among multiple participants each generating and receiving data. This requires multicast services to convey the data generated by each source to all the recipients. The network will be require to replicate the data to conserve scarce resource, and to manage the multiparty session so that all participants receive consistent data. Furthermore, some scientists may join a multiparty session late, leave at any time, or may wish to form discussion subgroups. The network must provide management support for such dynamic sessions so that sessions can be created quickly, at will, and working session should not stall because of changing participant population.

**Handling multiple types of service (TOS):** To conserve resources the network will treat stream traffic (e.g., video) differently than bursty traffic (e.g., man-machine interaction). It will also provide different error control, and congestion control schemes to the various data types. Yet it will retain the integrity of multi-type connections and will deliver to the user the data in a coordinated manner.

# 6. Project Schedule for Phase III of REINAS

## 6.1 Overview

The first two phases of the REINAS project involved the development of a concept design, followed by the definition of detailed requirements. Activities include studying input from expected users, characterization of instruments and data, evaluation of technical alternatives, and development of a preliminary architecture.

Phase III is dedicated to building a prototype REINAS system. It includes beginning the detailed design of the system, and establishing a growing computer network of heterogeneous sources, rates, and users.

## 6.2 Equipment of Prototype REINAS

For equipment REINAS will initially utilize existing computers, workstations, and personal computers at UCSC, MBARI and NPS. Internet will be the means of interconnecting the computers and instruments. Radio links will be used as part of the network. Heavy use will be made of the existing instruments and computers at the sites mentioned below.

## 6.3 Sites

The following instrument sites will comprise the initial prototype REINAS system:

- *UCSC REINAS Meteorological Station*: Currently located on the roof of the Applied Sciences building. It will be relocated to a more scientifically interesting location on the UCSC campus. Its location is approximately 37.00 N, 122.05 W.
- *UCSC/Long Marine Lab Meteorlogical Station*: Existing MET station located at Long Marine Lab on extreme northern-most coastline of Monterey Bay.
- *MBARI Moss Landing Meteorological Station*: Acquired MET station to be placed on coastline at MBARI's Moss Landing Pt. Lobos support building.
- *MBARI OASIS M1-Buoy*: MBARI OASIS meteorolgical and oceanographic instrumentation mounted on a buoy and located at M1 site, located at approximately 36.75 N, 122.03 W.
- *MBARI OASIS M2-Buoy*: MBARI OASIS meteorolgical and oceanographic instrumentation mounted on a buoy and located at M2 site, located at approximately 36.70 N, 122.40 W.
- *Lockheed REINAS Meteorlogical Station*: Acquired MET station to be located on the Lockheed Missile and Space Company testing grounds in the Santa Cruz Mountains north of Santa Cruz.
- *Fort Ord Wind Profiler Radar and Met Station*: At present there is a 400 MHz profiler at Fort Ord. A 915 MHz unit is to be installed later in 1993.

Others likely to be added later include:

- *CODAR Sites*: Long Marine Lab and Pt. Pinos are the most probable sites for real-time data delivery.

- *UNIDATA*: Ingest UNIDATA observations, satellite imagery and models into REINAS.

- *Air Pollution Stations*: Add Monterey Bay Air Pollution Stations, Hydrology stations and others to network.

## 6.4 Schedule

The dates of Phase III are from July 1, 1993, to June 30, 1994.

In last six months of 1993:

- Establish protocol for network
- Grow the network
- Attach instruments
- Data into Database
- Preliminary fusion of model and instrument data
- Begin instrument and site quality control on other agency sites

In first six months of 1994:

- Collaboration with scientists
- Develop replay senarios
- Develop standard products
- Integrate new instruments

In the summer of 1994:

- Full set of instruments on line
- Meterologists and oceanographers can use Seabreeze data thru REINAS

## References


[BEW77] D. E. Barrick, M. W. Evans, and B. L. Weber. Ocean surface currents mapped by radar. *Science*, 198:138–144, 1977.

[BLC85] D. E. Barrick, B. J. Lipa, and R. D. Crissman. Mapping surface currents with codar. *Sea Technology*, October 1985.

[BMvdV92] Alberto Bartoli, Sape J. Mullender, and Martijn van der Valk. Wide-address spaces – exploring the design space. Report, University of Twente, Netherlands, December 1992.

[Bos92] L. Bosack. Method and apparatus for routing communications among computer networks. *U.S. Patent assigned to Cisco Systems, Inc.*, 1992.

[CB92] Timothy Clark and Kenneth Birman. Using the isis resource manager for distributed, fault-tolerant computing. Report TR 92-1289, Cornell University, Dept of Computer Science, June 1992.

[CFL79] I. Chlamtac, W.R. Franta, and K.D. Levin. Bram: The broadcast recognizing access mode. *IEEE Trans. Comm.*, 27(8):1183–1189, 1979.

[Chi91] J.N. Chiappa. A new ip routing and addressing architecture. *Unpublished Draft*, 1991.

[Col89] R. Coltun. Ospf: An internet routing protocol. *ConneXions*, 3(8):19–25, 1989.

[Cro55] D. D. Crombie. Doppler spectrum of sea echo at 13.56 mc/s. *Nature*, 175:681–682, 1955.

[DD87] B. Davies and T.R. Davies. The application of packet switching techniques to combat radio. *IEEE Proceedings*, 75(1), 1987.

[ea87] M. Stefik et al. Beyond the chalkboard: Computer support for collaboration and problem solving in meetings. *CACM*, 30(1):32–47, 1987.

[ea89] C. Upson et al. The application visualization system: A computational environment for scientific visualization. *IEEE Computer Graphics and Applications*, 4:30–42, September 1989.

[ERH92] D. Estrin, Y. Rekhter, and S. Hotz. Scalable inter-doman routing architecture. *Computer Comm. Review*, 22(4), 1992.

[Fou92] Open Software Foundation. *Introduction to OSF DCE.* Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1992.

[GLA88] J.J. Garcia-Luna-Aceves. Routing management in very large-scale networks. *Future Generation Computing Systems*, 4(2):81–93, 1988.

[GLA93] J.J. Garcia-Luna-Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Transactions on Networking*, 1(1):130–141, 1993.

[GLACL89] J.J. Garcia-Luna-Aceves, E.J. Craighill, and R. Lang. Floor management and control for multimedia computer conferencing. In *IEEE Multimedia '89 International Workshop on Multimedia Communications*, Ottawa, Canada, April 1989.

[GLAZ92] J.J. Garcia-Luna-Aceves and W.T. Zaumen. Protocol analysis and standard protocol suite selection for disn. SRI SETA Task Briefing to DISA, April 1992.

[GLAZ93] J.J. Garcia-Luna-Aceves and W.T. Zaumen. Shortest multipath routing using diffusing computations. SRI Invention Disclosure, 1993.

[GLOR91] M. Guillemont, J. Lipkis, D. Orr, and M. Rozier. A second-generation microkernel based unix: Lessons in performance and compatibility. In *Usenix Winter'91 Conference*, Dallas, TX, January 1991.

[Goo90] D. Goodman. Cellular packet communications. *IEEE Trans. Comm.*, 38(8):1272–1280, 1990.

[Gro91] The ISIS Group. Reconstructuring of isis for modern distributed operating systems. Draft Report TR 89-997, Cornell University, Dept of Computer Science, September 1991.

[Hed88] C. Hedrick. Routing information protocol. RFC 1058, Stanford Research Institute, SRI International, Menlo Park, CA, June 1988.

[ISO89] ISO. Intra-domain is-is routing protocol. *ISO/IEC*, JCT1/SC6(WG2 N323), September 1989.

[ISO91] ISO. Protocol for exchange of inter-domain roueing information among intermediate systems to support forwarding of iso 8473 pdus. Technical report, International Standards Organization, 1991.

[Jaf84] J.M. Jaffe. Algorithms for finding paths with multiple constraints. *Networks*, 14:95–116, 1984.

[KS80] L. Kleinrock and M. Scholl. Packet switching in radio channels: New conflict-free multiple access schemes. *IEEE Trans. Commun.*, 28(7):1015–1029, 1980.

[LB83] B. J. Lipa and D.E. Barrick. Least-squares method for the extraction of surface currents from codar crossed-loop data: Applications at arsloe. *IEEE Journal of Ocean Engineering*, OE-8:226–253, 1983.

[LR91] K. Lougheed and Y. Rekhter. Border gateway protocol 3 (bgp-3). RFC 1267, Stanford Research Institute, SRI International, Menlo Park, October 1991.

[Mar90] Keith Marzullo. Tolerating failures of continuous-valued sensors. Report TR 89-997, Cornell University, Dept of Computer Science, September 1990.

[MGLA87] J. Mathis and J.J. Garcia-Luna-Aceves. Survivable multiband networking. In *IEEE MILCOM '87*, Washington, DC, October 1987.

[MLM92a] Sape J. Mullender, Ian M. Leslie, and Derek McAuley. Pegasus – operating system support for distributed multimedia systems. Report, University of Twente, Netherlands, Sept 1992.

[MLM92b] Sape J. Mullender, Ian M. Leslie, and Derek McAuley. Pegasus project description. Report, University of Twente, Netherlands, Sept 1992.

[MLP+93] P.E. Mantey, D.D.E. Long, A.T. Pang, H.G. Kolsky, et al. Reinas: Concept statment. Report ucsc-crl-93-05, Univ of California, Santa Cruz, Santa Cruz, CA 95064, January 1993.

[MNSS87] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos authentication and authorization system. Project Athena Report 92-06-04, M.I.T., Cambridge, Massachusetts, December 1987.

[MW90] Keith Marzullo and Mark Wood. Making real-time reactive systems reliable. Report TR 90-1155, Cornell University, Dept of Computer Science, March 1990.

[MW91] Keith Marzullo and Mark Wood. Tools for monitoring and controlling distributed applications. Report TR 91-1187, Cornell University, Dept of Computer Science, February 1991.

[MWC90] K. Marzullo, M. Wood, and R. Cooper. Tools for distributed application management. Technical Report 90-1136, Cornell University, Department of Computer Sciences, July 1990.

[NA93] B. Clifford Neuman and Steven Seger Augart. *The Prospero Protocol. Version 5.* Information Sciences Institute, University of Southern California, 0.3b edition, February 1993.

[Nea92] T. C. Neal. Analysis of monterey bay codar-derived surface currents march to may 1992. Technical Report M.S. Thesis, Naval Postgraduate School, 1992.

[Neu92] B. C. Neuman. The virtual system model: A scalable approach to organizing large systems. PhD Thesis 92-06-04, University of Washington, Seattle, Washington, June 1992.

[NOA93] NOAA. Scientific and technical review. Technical report, NOAA Forecast Systems Laboratory, July 1993.

[Ope92] Open Software Foundation, Cambridge, MA. *Application Development Guide*, 1.0.1 edition, 1992.

[PS93a] Alex Pang and Kyle Smith. Spray rendering: A new framework for visualization. Report ucsc-crl-93-01, Univ of California, Santa Cruz, Santa Cruz, CA 95064, January 1993.

[PS93b] Alex Pang and Kyle Smith. Spray rendering: Visualization using smart particles. Visualization '93, Univ of California, Santa Cruz, Santa Cruz, CA 95064, October (to appear) 1993.

[Rom86] R. Rom. Collision detection in radio channels. *Local Area and Multiple Access Networks (R. Pickholtz, Ed.)*, 1986. Chapter 12.

[RS90] R. Rom and M. Sidi. Multiple access protocols: Performance analysis. *Springer-Verlag New York*, 1990.

[Sat90] M. Satyanarayanan. Scalable, secure, and highly available distributed file access. *IEEE Computer*, May 1990.

[SCP83] N. Shacham, E.J. Craighill, and A. Poggio. Speech transport in packet-radio networks with mobile nodes. *IEEE JSAC*, 1(6), 1983.

[SGI91] SGI. Getting started with iris explorer. Technical report, Silicon Graphics Inc., December 1991.

[SKB93] Tage Stabel-Kulo and Peter Bosch. Overview of the pegasus storage architecture. Report, University of Twente, Netherlands, January 1993.

[SNS88] J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Winter 1988 USENIX Conference*. USENIX, 1988.

[Ste92] M. Steenstrup. Inter-domain policy routing protocol specification: Version 1. Internet Draft, May 1992.

[Sto91] M. Stonebraker. *An Overview of the Sequoia 2000 Project*, volume Report No. 91/5. 557 Evans Hall, UC Berkeley, 1991.

[Tha93] Madhukar Thakur. Prospero: A distributed system architecture. Technology Evaluation Report, February 1993.

[Wil91] Ross N. Williams. *Adaptive Data Compression.* Kluwer Academic Publishers, Boston/Dordrecht/London, 1991.

[ZGLA92] W. Zaumen and J.J. Garcia-Luna-Aceves. Dynamics of link-state and loop-free distance-vector routing algorithms. *Journal of Internetworking*, December 1992.

# Index