A Real-time System for Processing, Sharing, and Display of Physiology Data

by

Eric TszLeung Ho

Submitted to the Department of Electrical Engineering and Computer Science

in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Computer Science and Engineering

and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2003

Author ....
Department of Electrical Engineering and Computer Science
May 8, 2003

Certified by ....
Andrew W. Lo
Harris & Harris Group Professor
Thesis Supervisor

Certified by. ....
Dmitry V. Repin
Postdoctoral Associate
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

A Real-time System for Processing, Sharing, and Display of Physiology Data

by

Eric TszLeung Ho

Submitted to the Department of Electrical Engineering and Computer Science
on May 8, 2003, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Computer Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents RStudio, a software package that provides a critical aid for studies of the psychophysiology of traders and financial decision-making processes. RStudio provides the tools for physiology data acquisition and their display in real-time as well as data sharing across multiple machines on the network. It also supports a monitoring system with alert capabilities to aid risk-management processes based on traders' physiology.

Thesis Supervisor: Andrew W. Lo
Title: Harris & Harris Group Professor

Thesis Supervisor: Dmitry V. Repin
Title: Postdoctoral Associate

Acknowledgments

This thesis would not have been possible without the support of several people. I would like to thank:

- Prof. Andrew Lo, my thesis advisor, for providing this great opportunity for my research project and agreeing to supervise my thesis.
- Dr. Dmitry Repin for his vision and esteemed guidance in this project. His valuable comments helped to shape the design and implementation of the system described in this thesis.
- Svetlana Sussman for providing crucial administrative support for this project.

Additional gratitude goes to my parents and friends whose encouragement and support in this thesis is priceless.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rationality of investors in the modern financial marketplace has been a long standing debate among researchers. Advocates of the Efficient Market Hypothesis argue that investors are rational and market prices fully reflect all available information. On the other hand, behavioral economists argue that investors are often irrational and exhibit predictable biases. These economists believe that psychological biases are consistently exhibited by market participants; the end result is that the market will be predictable to some extent. A commonly used example of such investor and market irrationality is the rise and fall of the U.S. stock market in the past few years.

While the debate on rationality of decision making continues, current research in cognitive sciences and financial economics suggests a link between rationality and emotion. A systematic investigation of this link is currently underway by Andrew Lo and Dmitry Repin at MIT Laboratory for Financial Engineering. The research involves measuring the physiological signals of financial securities traders during live trading sessions. Examples of these signals include heart rate, skin conductance, forehead temperature, and respiration rate. By measuring these signals, the emotional states of the traders in different market conditions can be analyzed. For example, the investigation reported statistically significant changes in physiological variables during periods of heightened market volatility relative to normal-volatility control periods, as well as during a number of short-term market events.

1.1 Goal

The goal of this thesis is to design a tool that aids research in psychophysiology of traders and financial decision-making.

In a typical experiment, this tool is used to record physiology data from traders working in a live market. At the same time, prices of the securities being traded by the traders are collected from data feed services. These physiological data and financial data are analyzed together to uncover links between prices change in the market and physiological changes in traders.

In addition, this tool can serve as a real-time monitor and alert application for managers of traders on the trading floor. This application monitors the physiological states of each trader; it can be configured to produce alerts when there are indications of events or physiological states that require special attention.

1.2 Challenge

Developing software for this research project poses several technological challenges. Firstly, the tool needs to collect physiology data in real-time. Secondly, the tool needs to properly manage and display several real-time data feeds synchronously. Thirdly, the tool needs to perform real-time data preprocessing on multiple data channels without affecting performance. Lastly, the tool needs to communicate with remote machines and share data in a real-time fashion.

1.2.1 Time requirements

The real-time nature of the application requires special attention to the time requirements of different components in the software. Different types of physiology data require different minimal data sampling rate. However, these data sampling rate cannot be higher than the rate of data preprocessing, which depends on the capabilities of the underlying machine. In addition, extra precaution is needed to deal with synchronization problems that occur when displaying real-time data from multiple sources with different delays. To deal with these challenges, the tool must run smoothly in all settings and provide resolution mechanisms for synchronization of multiple data sources.

1.2.2 Memory requirements

A memory buffer is needed to temporarily store incoming real-time data for preprocessing. After the data are preprocessed, the memory buffer can be recycled. Given that data preprocessing occurs at a faster rate than data collection, a finite memory buffer is needed to store incoming data. However, there are occasions in which data preprocessing requires the entire data set within a certain time frame. In this case, the system needs to allocate enough memory to store data collected within the specified time frame.

1.2.3 Storage requirements

Data collected from traders are saved onto the hard disk. A typical experiment involved in this research project requires collecting data at a high rate (e.g. 256Hz) for up to one hour. The amount of data collected can be as big as several hundred megabytes. Since all of these data need to be stored at secondary storage locations on the hard disk, this requires the computer to have enough disk space for storage of physiology data.

1.2.4 Network requirements

An efficient network model is needed for sharing of physiology data over the Internet. For example, a client-server model can be used to improve the efficiency of network communication. In addition, data collected from multiple sources contain various amounts of network delay; the tool must deal with these delays and provide proper time synchronization of data.

1.3 Solution

Real-time Studio (RStudio) provides a solution to the above requirements and goals. It contains the following features:

- Collection of physiology data from users.
- Visualization of real-time physiology data.
- Preprocessing of data.
- Data sharing over the network.
- Real-time monitor and alert system of physiological states.

Examples of physiological data collected include electrocardiogram signals, blood volume pulses, forehead temperatures, and skin conductance levels. These data are displayed as real-time line graphs on the graphic display. Next, the physiological data are preprocessed in order to extract meaningful information from the raw data. The preprocessed data are then transmitted over the Internet and shared among all users. The shared physiological data are used in a monitor system where the physiological states of each trader can be effectively monitored.

1.4 Road map

The rest of this thesis is broken down into three sections. The first section gives an overview of current research, in Chapter 2. The second section provides a top-level design of RStudio, in Chapter 3. The third section contains details about the design and implementation of each component in the RStudio system, in Chapters 4 to 8.

Chapter 2 provides background information on current research in trader physiology and financial decision-making. It describes the methodology being used in the experiment and the nature of the data involved. Current findings of the research are also presented.

Chapter 3 covers a top-level design of the RStudio system. It presents the modular design of the three main modules of RStudio — the client module, the server module, and the monitor module. A top-level functional specification of each module is presented.

Chapter 4 describes the data-collection process. Physiological data such as blood volume pulse, skin conductance, and finger temperature are collected by RStudio. The specifications of the data handling and storage procedures are depicted in this chapter.

Chapter 5 presents the design of the visualization and user interface. Physiology data collected from the user are visualized on the graphic display as real-time line graphs. The user interface provides an easy and flexible way of graphs control and manipulation. Addition features such as event recording and message logging give users more control during experiments.

Chapter 6 covers the data preprocessing module. Physiology data are preprocessed before being sent over the network. The various data preprocessing algorithms being used are discussed in this chapter.

Chapter 7 describes the network model. This model provides a meaningful way of sharing physiology data across the Internet. The server-client architecture of the network model is presented. Using this model, multiple clients that connect to the server can efficiently share physiology data collected from the same trader.

Chapter 8 depicts the design of an application that makes use of shared physiology data across the network. The monitor system allows a supervisor to actively monitor the physiological performance of individual traders. Visual alerts are given in response to abnormal physiological states of each trader. Indicators of the aggregate physiological state of the participating group of traders are also provided.

Chapter 9 presents the conclusions and lessons learned from the implemented system. This chapter also provides directions of possible future work, including new functionality and system optimizations.

Chapter 2

Background

This section presents an overview of the research that is currently underway at MIT Laboratory for Financial Engineering. Understanding the nature of this research is crucial to understanding the functional and design requirements for RStudio.

2.1 Research design

The goal of this research is to provide some insight in the longstanding controversy in economics and finance — whether financial markets are governed by rational forces or by emotional responses. Previous studies have suggested an important link between rationality in decision making and emotion [1]. This research project attempts to verify this link by studying the importance of emotion in the decision-making process of professional securities traders. During a live trading session, physiology data from traders are collected together with real-time prices of the securities being traded. By matching the different market events with the traders' physiological response, the relationship between market condition and emotional states can be uncovered.

2.2 Physiology data collection

In the human brain, the autonomic nervous system (ANS) is responsible for the regulation of internal states that are mediated by emotional and cognitive processes. Certain ANS responses can be detected as body physiology changes. The current research focuses on the following physiological characteristics: electrocardiogram (EKG), brain wave (EEG), skin conductance (SCR), blood volume pulse (BVP), electromyographical data (EMG), respiration rate, and body temperature. Figure 2-1 demonstrates the placement of sensors for measuring these physiological responses.

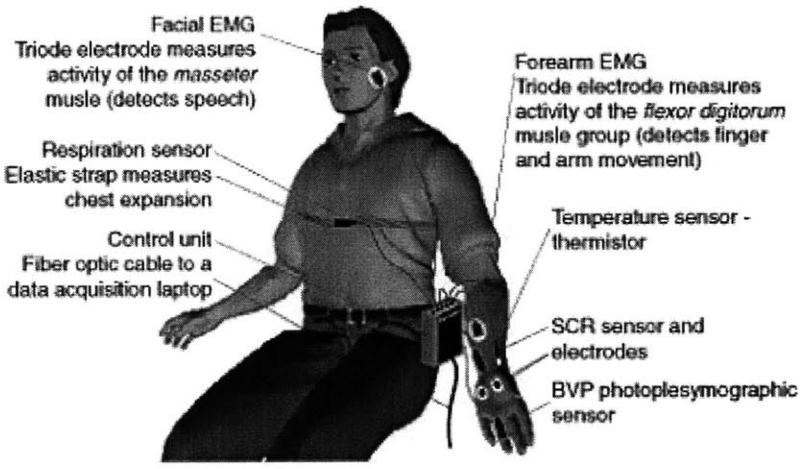

Facial EMG
Triode electrode measures activity of the *masseter* musle (detects speech)
Respiration sensor
Elastic strap measures chest expansion
Control unit
Fiber optic cable to a data acquisition laptop
Forearm EMG
Triode electrode measures activity of the *flexor digitorum* musle group (detects finger and arm movement)
Temperature sensor - thermistor
SCR sensor and electrodes
BVP photoplesymographic sensor

Figure 2-1: Placement of sensors for measuring physiological responses

Figure 2-2 shows a typical setup for the measurements of the real-time physiological responses of financial traders during live trading sessions. The physiology data collected are compared synchronously with real-time market data used by the traders.

A ProComp data-acquisition device is used to measure physiological data for all subjects. Each sensor is equipped with a built-in notch filter at 60 Hz for automatic elimination of external power line noise, and standard AgCl triode and single electrodes are used for SCR and EMG sensors, respectively. SCR electrodes are placed on the palmar sites. The BVP photoplesymographic sensor is placed on the inside of the ring or middle finger. The arm EMG triode electrode is placed on the inside surface of the forearm, over the flexor digitorum muscle group. The temperature sensor is inserted between the elastic band placed around the wrist and the skin surface. The

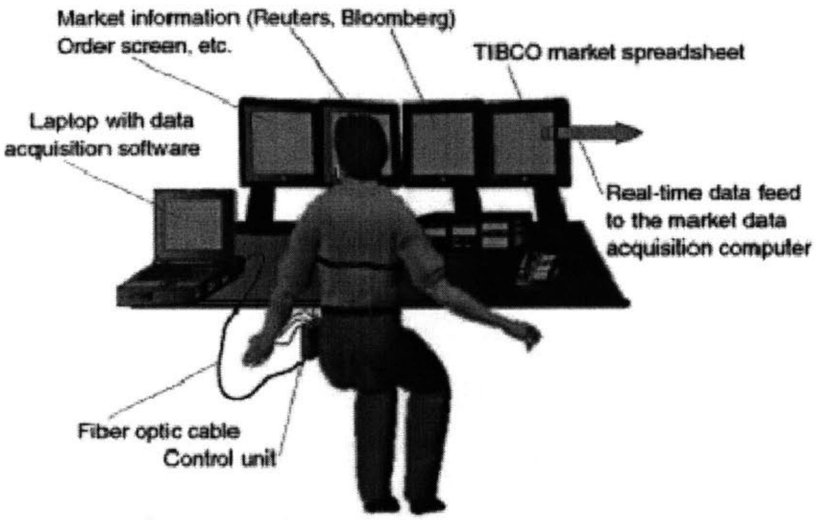

Market information (Reuters, Bloomberg)
Order screen, etc.
TIBCO market spreadsheet
Laptop with data acquisition software
Real-time data feed to the market data acquisition computer
Fiber optic cable
Control unit

Figure 2-2: Typical experimental data-collection setup

facial EMG electrode is placed on a masseter muscle, which controls jaw movement. The respiration signal is measured by chest expansion using a sensor attached to an elastic band placed around the subject's chest. An example of the real-time physiological data collected over a two-minute interval for one subject is given in Figure 2-3.

2.3 Financial data collection

The financial data being collected involve the prices of the securities being traded by the traders whose physiological responses are being measured. Real-time financial data are provided by data feed services such as Bloomberg, Reuters, or other feeds supplied by the stock exchange where the experiment is being conducted. Commercial off-the-shelf software such as Matlab and Excel are used to collect this data. The financial data collected are time-stamped and stored in a file for subsequent analysis. Figure 2-4 displays an example of the real-time financial data collected over a sixty-

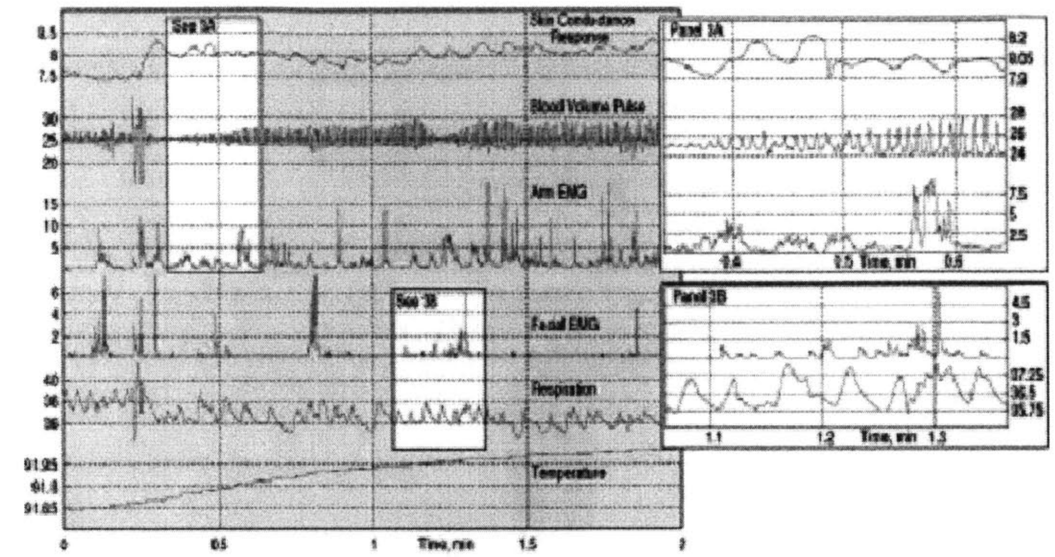


Figure 2-3: Typical physiological response data

minute interval.

2.4 Research findings

Research findings indicate that there are statistically significant differences in physiological responses in different market conditions [1]. In addition, the structure of response differs among different traders, which may partly be related to the trader's experience. These findings suggest that emotional responses are a significant factor in the real-time processing of financial risk. They also suggest that emotions are a determinant in the evolutionary fitness of financial traders.

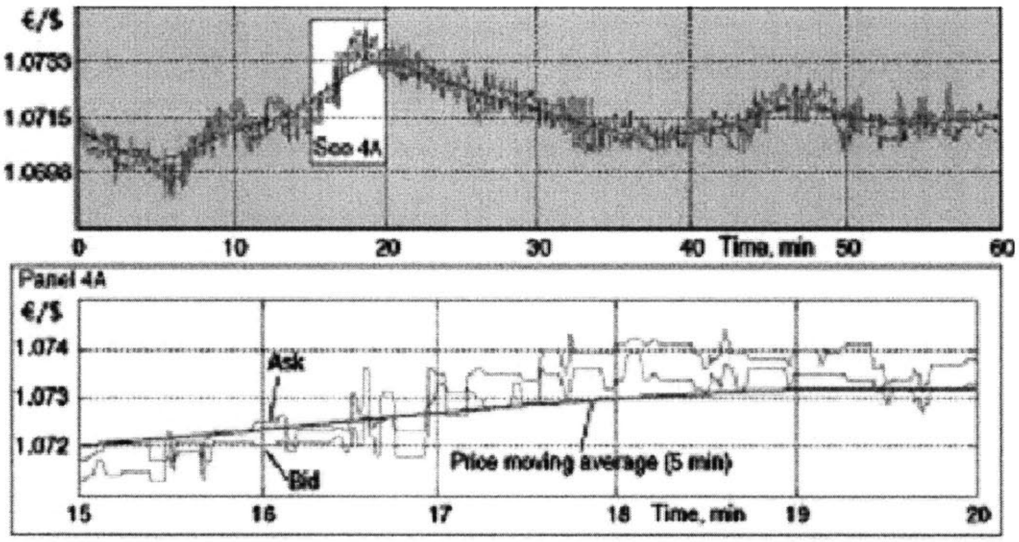


Figure 2-4: Typical real-time market data

Chapter 3

Design Overview

A top-down design of RStudio involves three independent software modules — RStudioClient, RStudioServer, and RStudioMonitor. Each of the three modules is a standalone application. RStudioClient is responsible for collecting real-time physiology and financial data, visualizing data on a graphic display, and performing data preprocessing. RStudioServer is responsible for providing network connectivity for sharing data between RStudioClient and RStudioMonitor. RStudioMonitor is responsible for providing a monitor and alert system for physiology data from multiple traders. Table 3.1 summarizes the functional division of the three modules.

Figure 3-1 depicts the relationships between the three modules. In a typical application of the RStudio system, there exists one copy of RStudioServer and multiple copies of RStudioClient and RStudioMonitor. The single copy of RStudioServer provides network sharing of physiology data among all other modules. Each copy of RStudioClient provides data collection and preprocessing for one trader. Each copy of RStudioMonitor receives physiology data from every RStudioClient module through

RStudioClient	RStudioServer	RStudioMonitor
Data collection Data visualization Data preprocessing Data storage	Network connectivity Network sharing of data	Active monitor system Alert system Group data visualization

Table 3.1: Functional division of the RStudio system

the RStudioServer module; the physiological states of all traders can be monitored using this module.

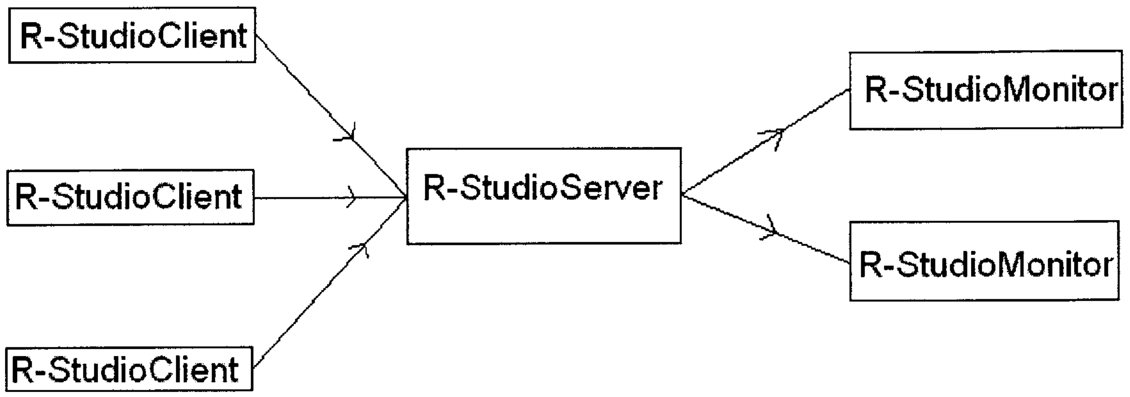


Figure 3-1: Top-level modular decomposition of RStudio

3.1 RStudioClient

RStudioClient provides data collection, preprocessing, visualization, and storage. This module runs on the machine which is connected to the ProComp data-acquisition device. Physiology data are collected from the ProComp device, displayed as real-time line graphs, preprocessed, and transmitted to the RStudioServer module across the network. A modular design of RStudioClient is depicted in Figure 3-2.

The physiology data-collection unit records data from the trader. Real-time physiology data are collected by the ProComp device through sensors attached to the trader. Each ProComp device has eight data channels and can collect multiple physiology signals (such as heart rate, skin conductance, etc.) simultaneously.

The core unit is the main operating module for RStudioClient. It maintains and allocates resources for the process threads that form each of the separate modules in

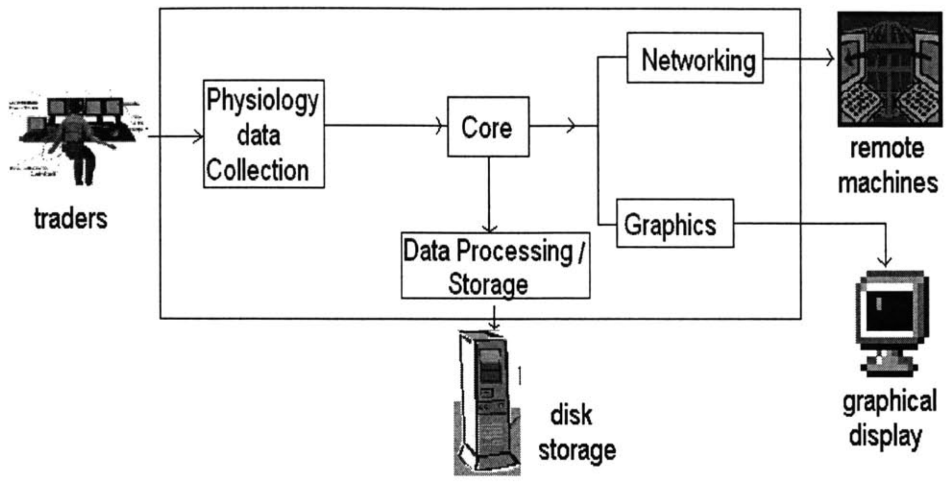


Figure 3-2: Modular decomposition of RStudioClient

RStudioClient.

The data processing and storage unit handles the preprocessing of physiology data and the storage of preprocessed data. The type of data preprocessing depends on the nature of data being used. After data preprocessing, processed data are flushed to secondary storage location.

The networking unit communicates with RStudioServer and sends preprocessed physiology data to the server. In particular, a data socket is established between the client and the server. All physiology data are transported through the socket.

The graphics unit plots the physiology data on the display and provides a user interface. The interface allows the user to control the data-collection process and change the view settings of the graphic display. In addition, the user can enter custom markers that put timestamps in the data. An example graphical interface is shown in Figure 3-3.

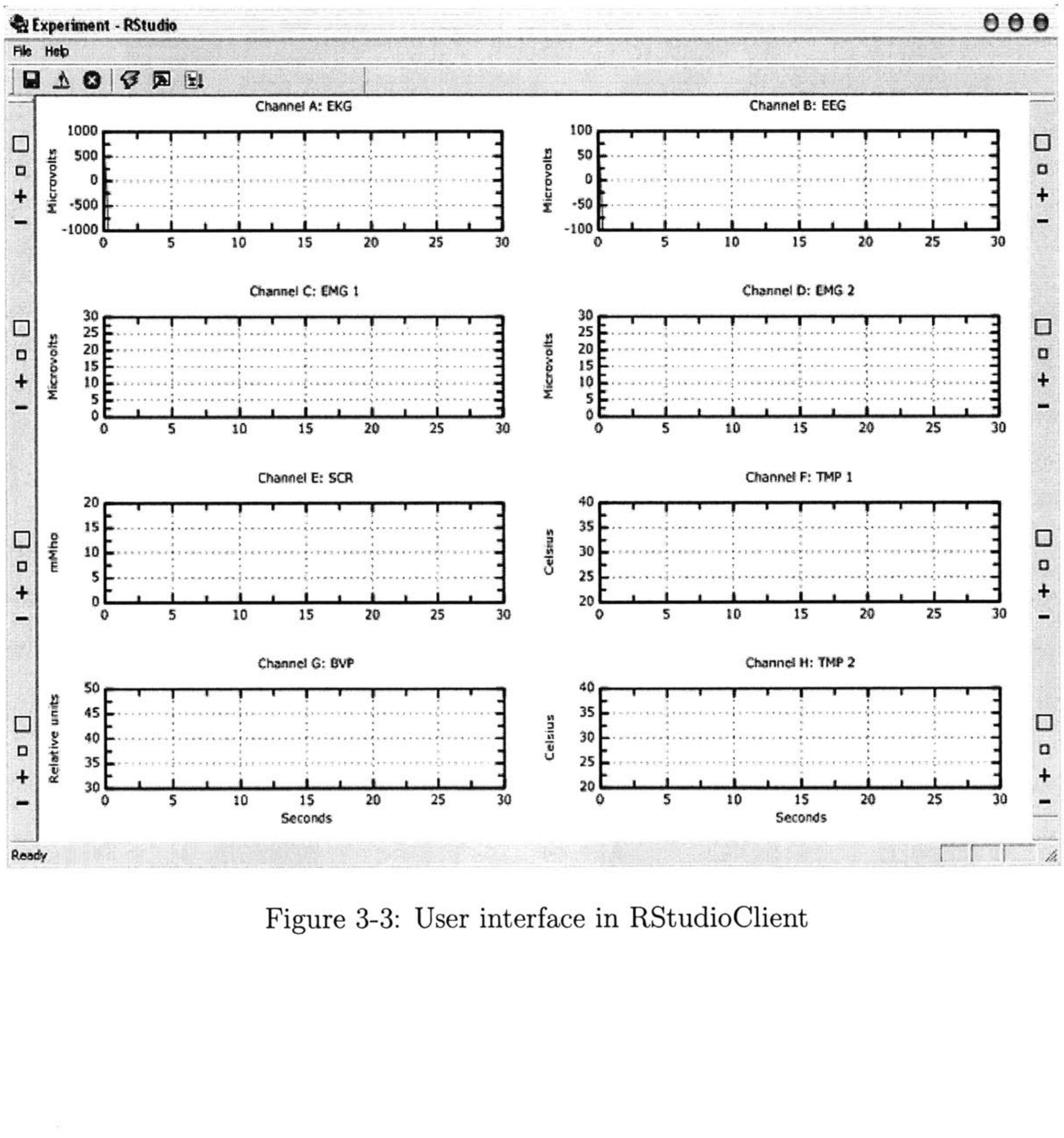


Figure 3-3: User interface in RStudioClient

3.2 RStudioServer

RStudioServer is responsible for providing network connectivity between multiple RStudioClient and RStudioMonitor modules. Physiology data collected by each RStudioClient module are sent to RStudioServer, which in return retransmits the data to every RStudioMonitor module. This network routing mechanism allows physiology data to be shared across the network. Figure 3-4 depicts a high level design of the RStudioServer module.

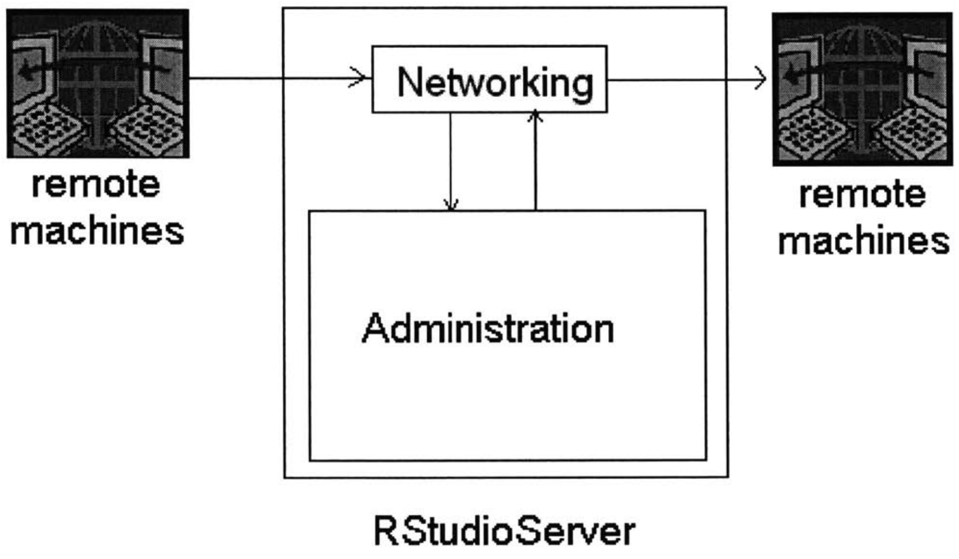


Figure 3-4: Modular decomposition of RStudioServer

The networking unit provides network connectivity for both RStudioClient and RStudioMonitor modules. This unit also maintains network socket ports for physical transportation of data among client modules across the network.

The administration unit maintains a list of the RStudioClient and RStudioMonitor modules that are currently connected to RStudioServer. The list contains information such as the types of the connecting clients and their network addresses. This list is updated as new connections are created.

When an RStudioClient module sends a new piece of data to RStudioServer, the networking unit receives and acknowledges the data. This data are then retransmitted to every RStudioMonitor module that is connected to the server. With this mechanism, every RStudioMonitor module can effectively monitor physiology data of every trader that is connected to the RStudioClient module.

3.3 RStudioMonitor

The RStudioMonitor module is responsible for providing a system for monitoring physiology of several traders at the same time. This module receives physiology data from multiple RStudioClient modules through RStudioServer. The physiology data are then visualized on the graphical display. Users of the RStudioMonitor can actively monitor the physiological states of all traders through the interactive display system. Figure 3-5 depicts a high level design of the module.

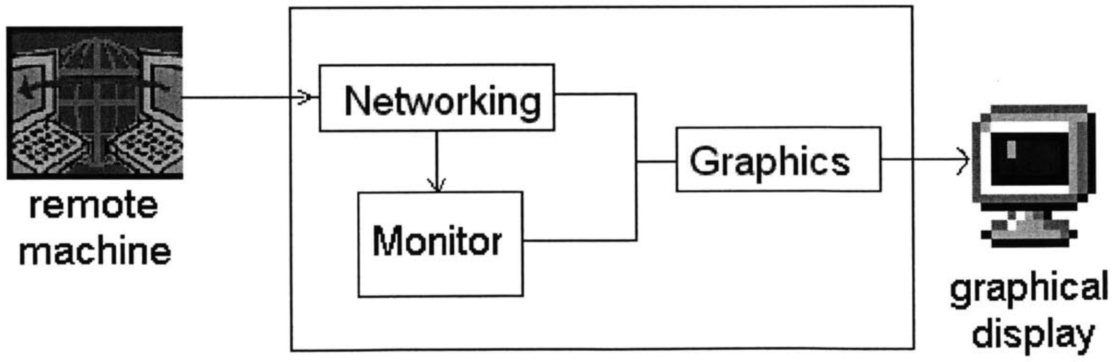


Figure 3-5: Modular decomposition of RStudioMonitor

The networking unit maintains connectivity with RStudioServer. Physiology data sent from the server arrive at this module in the form of network data packets. The networking unit then processes and extracts physiology data from the data packet; the extracted information is stored in the memory buffer provided by the monitor module.

The monitor unit provides many useful system functions. Firstly, the monitor unit maintains a storage table for the current physiology data for each trader. Information in the table will be used for various system monitoring functions. For example, the table uses a timestamp to record the previous time when a data packet arrives from a particular RStudioClient. If the timestamp expires, the monitor system will be able to notify the user. Secondly, the monitor unit processes incoming physiology data and performs statistical calculations such as the aggregate physiological state of every trader. Thirdly, the monitor unit allows the user to measure and update the baseline measurement for each trader. Lastly, when the physiological state of the trader reaches above abnormal levels, the monitor unit creates alert signals for the user.

The graphics unit presents visual information about the physiological states of traders. The visual information being conveyed includes the preprocessed physiology data of every trader together with the summary statistics of all traders in the participating group. This unit also contains an interactive user interface system that allows the user to control baseline measurement through the monitor module. In addition, visual alerts are given by the graphics unit. These visual alerts provide warnings of traders with abnormal physiological states; they are critical for monitoring the physiological performance of traders.

Chapter 4

Data Collection

In the RStudio system, physiology data are collected using the ProComp data-acquisition device manufactured by ThoughtTechnology Ltd. The data collected are then stored on a temporary storage buffer. This storage buffer provides access for other functional components of RStudioClient such as visualization and preprocessing. This chapter discusses the nature of the data channels, the communication interface with ProComp, and the temporary storage system.

4.1 Data channels

The ProComp data-acquisition device provides eight channels of sensor data. Each of the eight channels represents data collected from different types of sensors. Table 4.1 lists the channels in detail.

4.2 Data-collection interface

A photo of the ProComp device is shown in Figure 4-1. The ProComp device is a data-collection hardware that contains eight channels; each of the channels can be connected to a sensor. This device collects physiology data from the sensors and retransmits the data via an optical link to the serial port of a computer.

On the computer, the ProComp device provides a dynamic link library (DLL)

Channel	Symbol	Description	Data rate
A	EEG	Electroencephalogram (Brain Waves)	256Hz
B	EKG	Electrocardiogram	256Hz
C	EMG1	Electromyography	32Hz
D	EMG2	Electromyography	32Hz
E	SCR	Skin Conductance	32Hz
F	TMP1	Finger temperature	32Hz
G	BVP	Blood volume pulse	32Hz
H	TMP2	Forehead temperature	32Hz

Table 4.1: List of different data channels

Figure 4-1: ProComp data-acquisition device

which serves as an application programming interface (API) for RStudioClient to collect physiology data. By using the interface specified by the API, RStudio can control the data-collection process in ProComp.

At each fifty millisecond intervals, raw physiology data collected by ProComp are sent to the data-collection component in RStudioClient. RStudioClient then processes and stores the data. Figure 4-2 depicts the relationship between ProComp and RStudioClient.

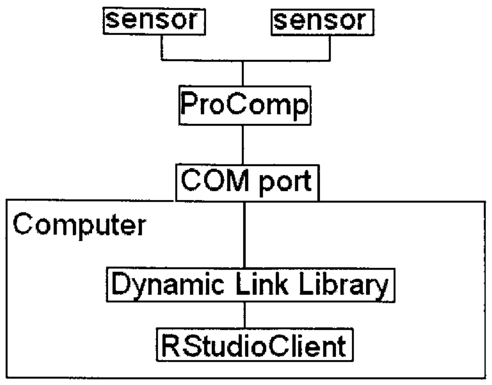


Figure 4-2: Data-collection interface between RStudioClient and ProComp

4.3 Data storage

New data collected from ProComp are stored in a temporary memory buffer for up to sixty seconds. Other components of RStudioClient, such as data visualization and preprocessing, can access and use the data in this memory buffer. At the end of each sixty second interval, data in the memory buffer are flushed to secondary storage in the hard disk and the memory buffer is recycled.

The structure of the memory buffer table is shown in Figure 4-3. Each of the eight data channels has separate memory allocations in the buffer table. Each memory allocation contains a head pointer are that stores the location of the unused portion of the memory table. New data are stored at the location specified by the head pointer, and the head pointer is updated afterwards.

As new data arrive from each channel, the head pointers move towards the end of the buffer table. When the pointers reach the end of the table, a memory flush operation occurs. In this operation, all data on the buffer table are saved to secondary storage location on the hard disk. The head pointers for each channel are then reset to the beginning locations of the memory buffer table. The memory buffer table is then recycled and ready to accept new data.

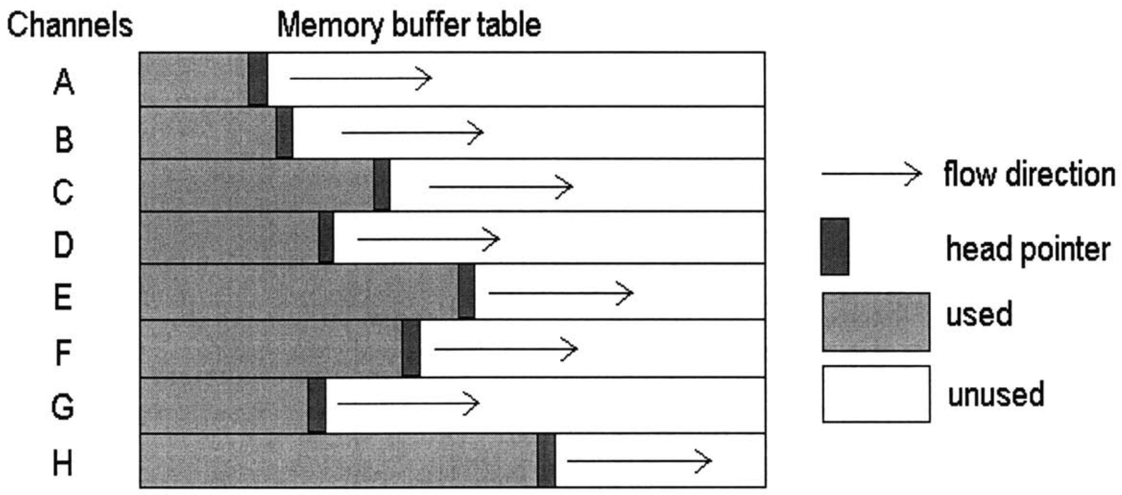


Figure 4-3: Structure of memory buffer table

Chapter 5

Data Visualization and User Interface

The data visualization component of RStudioClient creates an efficient way of displaying real-time physiology data gathered from the trader. The use of real-time data plots in RStudioClient provides a way to inform the user about the most current physiological conditions of the trader.

In addition, RStudioClient provides a flexible user interface. Through the user interface, users can control and manipulate graph settings. Users can also create timestamps to record special events that occur during the experiment.

5.1 Real-time display of graphs

The display system contains eight separate data plots; each of the plots is responsible for displaying data collected from a data channel. The data plots are constantly updated with new data that arrive from the data-collection component in RStudioClient. Each data plot displays data collected within the previous thirty-second interval; after every thirty seconds the plots are refreshed. Figure 5-1 shows a sample data plot.

5.2 Graph control and manipulation

Each data plot has default settings for the viewing window. The user interface provides a flexible way of changing these settings in real-time. By using the graph control buttons located next to the data plots, users can change the dimensions of the viewing window. Changes in the viewing window scale have an equivalent effect of zooming, whereas changes in the offset are related to vertical shifts in the data plots. These graph control functions allow the user to easily manipulate the viewing window during live experiment. Figure 5-2 shows an example screen of RStudioClient together with the view control system.

5.3 Experiment control interface

RStudioClient provides a graphical user interface for users to control the experiment process. The system toolbar window contains control buttons which perform different system control duties. Figure 5-3 provides an example of the system control toolbar. The file save button allows the user to specify the file location for secondary storage of physiology data. The start and end experiment buttons enable the user to begin or terminate the data-collection process. The server connection button provides network connectivity to RStudioServer. The configuration file button enables the user to configure system configuration settings. The event log button allows the user to view the content of the event log.

5.4 Event and message logging

During the experiment, physiology data collected from the trader might be polluted by artifacts. For example, one of the sensors attached to the trader might fall off accidentally. The user must record the time and type of the artifacts so that the integrity of the collected physiology data can be preserved.

RStudioClient provides an efficient event logging system to record the time and type of these artifacts. When an artifact has occurred, the user can create a marker

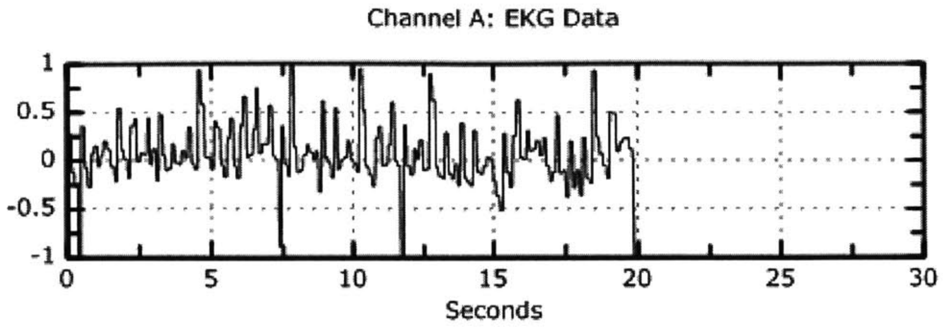


Figure 5-1: Sample data plot

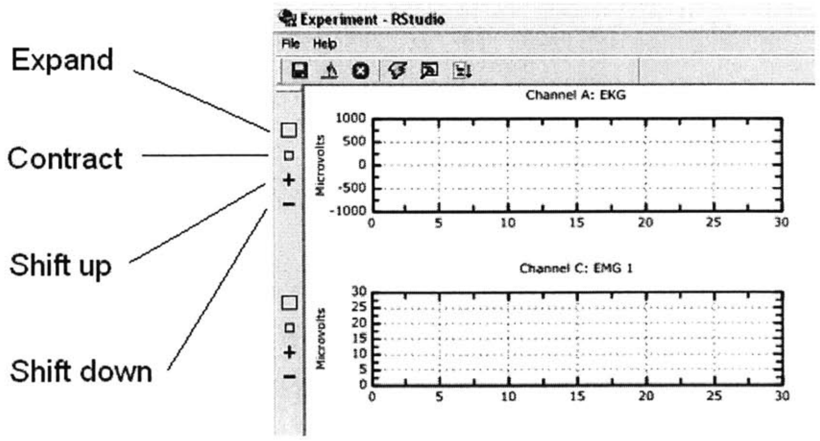


Figure 5-2: View control system

Marker name	Hotkey	Description
Artifact	A	An artifact has occurred
Start	S	Experiment has started
End	E	Experiment has ended
Fault	F	Certain problem has occurred
Message	M	Record user defined message
Synchronization	Q	For synchronization with other systems

Table 5.1: System markers

by pressing a special hotkey on the keyboard. The marker event is time-stamped and recorded into an event file. By comparing timestamps in the event file and the physiology data file, one can locate the portion of the physiology data that corresponds to the time period when the artifact has occurred. The different types of markers are shown in Table 5.1.

Markers A, S, E, and F are used to timestamp specific events that occur during the experiment. Marker M is a special marker which enables the user to type a message and record the message to the event file. This message logging capability can be used as a supplement to the other markers. Marker Q is used for time synchronization between RStudioClient and other external systems; more details in this topic are provided in Section 7.4.

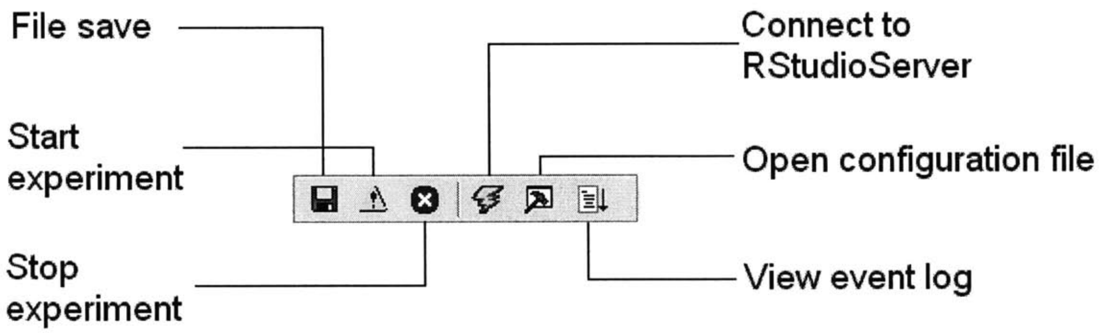


Figure 5-3: System control toolbar

Chapter 6

Data Preprocessing

Raw physiology data collected from traders are preprocessed in real-time. The preprocessed data are then sent to RStudioServer for network sharing. There are several advantages in performing data preprocessing. Firstly, the data can be compressed and reduced in size before being sent over the network. This results in more efficient network sharing of data. Secondly, various algorithms can be applied to extract useful information from the raw data. The result reveals useful information about the current physiological condition of the trader.

6.1 Data preprocessing routine

The data preprocessing process runs independently and in parallel with the data-collection process. The two processes communicate through the use of memory storage table in Section 4.3. The data-collection component stores collected data into the memory buffer, whereas the data preprocessing component performs calculations based on these values stored in the memory buffer. Figure 6-1 depicts the relationship between the data preprocessing component and other components in RStudioClient.

A conceptual picture of the data processing routine is shown in Figure 6-2. At each period of one second, the data preprocessing component accesses the memory storage table and retrieves raw data collected from the previous five-minute interval. The component then applies appropriate data preprocessing algorithms on the raw

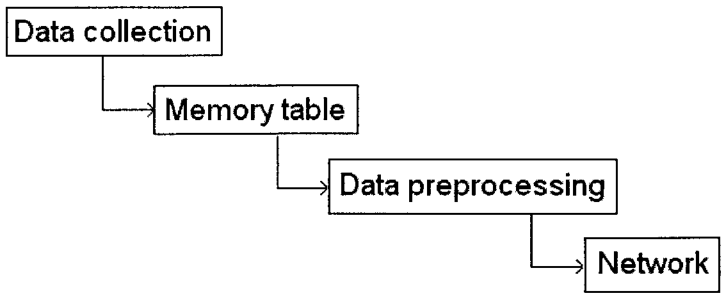


Figure 6-1: Data preprocessing overview

data. The results of the preprocessing processes are put together in a data packet and sent to RStudioServer.

The preprocessing period and the sampling interval can be adjusted. Smaller preprocessing periods increase the accuracy of the preprocessed information. However, this will also increase the network load as more data are being sent over the network. Longer preprocessing periods, on the other hand, reduce network load while giving up some accuracy in the preprocessed information.

The use of smaller sampling intervals can capture sharp changes in the traders physiological states. However, this method is not resistant to temporary fluctuations in the sensor readings. Larger sampling intervals, on the other hand, are more resistant to sharp changes in the data. This resistance to fluctuations provides a good estimate of the average physiological state of the trader.

6.2 Preprocessing algorithms

The preprocessing algorithms extract meaningful information from the raw physiology data collected from the trader. Various algorithms can be used to extract different types of information from the raw data. For example, the moving average algorithm calculates the mathematical mean of all raw physiology data in the sampling period;

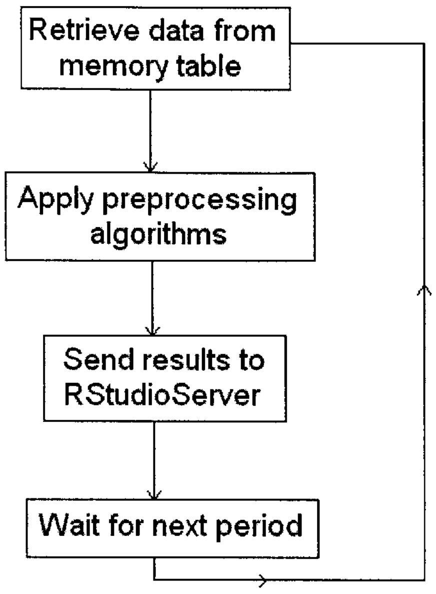

Retrieve data from memory table
Apply preprocessing algorithms
Send results to RStudioServer
Wait for next period

Figure 6-2: Data preprocessing routine

the result provides an average estimate of the physiological performance over the period. The Fast Fourier Transform (FFT) algorithm, on the other hand, operates on data channels with periodic signals; the result uncovers the dominant frequencies in the signal.

6.3 Preprocessing variables

The results of the preprocessing algorithms are stored in preprocessing variables. The values of the preprocessing variables reveal useful information about the current physiological state of the trader. At the end of each preprocessing period, the updated values of the preprocessing variables are sent to RStudioServer for network sharing. A list of preprocessing variables and the corresponding preprocessing algorithms is shown in Table 6.1.

Name	Channel	Algorithm
EEG relative alpha power	A	FFT
EKG heart rate	B	Custom
EKG heart rate variability	B	Custom
BVP heart rate	G	FFT
BVP heart rate amplitude	G	FFT
SCR number of responses	E	Custom
SCR level	E	Moving average
Finger temperature	F	Moving average
Forehead temperature	H	Moving average
EMG back	D	Moving average

Table 6.1: Preprocessing variables

Chapter 7

Network Model

The RStudioServer module provides a robust network model for efficient sharing of physiology data. By applying the client-server architecture, different RStudioClient modules can be linked together. This mechanism allows physiology data collected from multiple traders to be redistributed over the Internet. It also makes possible a variety of applications such as the monitor system described in Chapter 8.

The RStudio network model consists of one central server and multiple clients. RStudioServer acts as the server whereas RStudioClient and RStudioMonitor act as the clients. This relationship is depicted in Figure 7-1. Each RStudioClient module collects physiology data from the trader, preprocesses them, and sends the result to RStudioServer. The RStudioServer module then retransmits the information to every module of RStudioMonitor over the network.

7.1 Network connection

The network connections between server and the client modules are maintained by sockets [3]. When a new client module attempts to join the RStudio network, it first creates a socket connection with the server. After the socket connection is established, the client and server exchange administration information with each other. Once the exchange of information is completed, routing of physiology data can proceed.

The administration component of RStudioServer is responsible for maintaining a

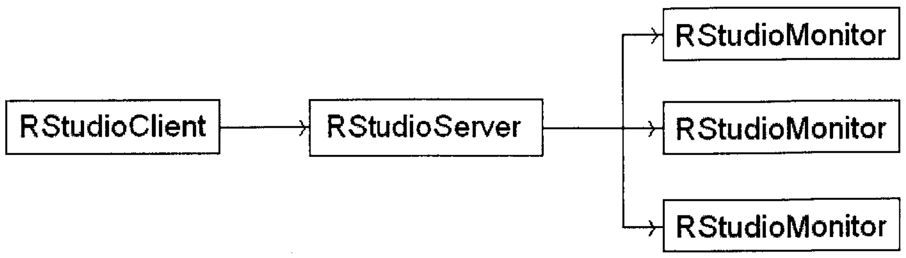
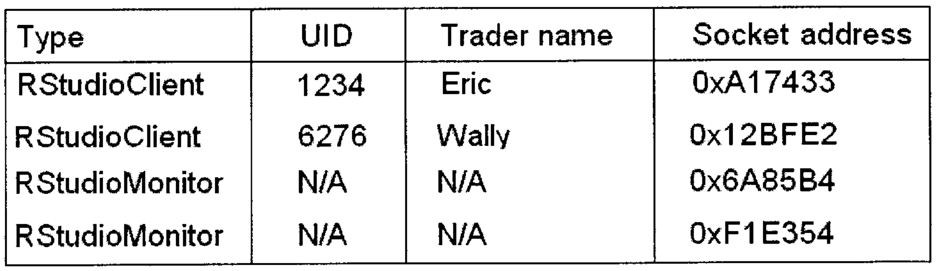


Figure 7-1: Network routing overview

list of client modules that are connected to the server. Such information is stored in the administration table. Figure 7-2 shows an example of such table.



Type	UID	Trader name	Socket address
RStudioClient	1234	Eric	0xA17433
RStudioClient	6276	Wally	0x12BFE2
RStudioMonitor	N/A	N/A	0x6A85B4
RStudioMonitor	N/A	N/A	0xF1E354

Figure 7-2: Administration table in RStudioServer

In the administration table, the type field is used to distinguish between different types of clients; the client type can either be RStudioClient or RStudioMonitor. The unique identifier (UID) and trader name fields are used to identify the trader. The socket address field contains network routing information about the client.

When a new RStudioClient module connects to RStudioServer, the client module first generates a random number as the UID. This client module then sends both the

UID and the name of the trader to the server. After the connection is established, the client proceeds to send preprocessed physiology data collected from the trader.

On the other hand, when a new RStudioMonitor module connects to RStudioServer, this monitor module receives information about the UID and trader name of every RStudioClient module that is connected to the server. After the connection is established, this monitor module will receive physiology data from the server. In addition, the monitor module receives notification when additional RStudioClient modules connect to the server.

7.2 Network routing

RStudioServer serves as a router of physiology data. Whenever a new piece of physiology data arrives from an RStudioClient module, the server checks the administration table and locates the socket addresses of every RStudioMonitor module in the network. The server then retransmits the physiology data to the machines located at the socket addresses. With this mechanism, every RStudioMonitor module is able to receive physiology data sent from every RStudioClient module.

Each trader is identified by the unique identifier (UID). When a new RStudioClient connects to RStudioServer, the client module generates a random number as the UID and uses this number as the identifier for the trader. This unique identifier exists on all subsequent data packets sent from the client module. When data packets arrive at the RStudioMonitor module, the monitor can use the UID field on the data packets to identify the sender.

7.3 Network protocol

Each of the RStudioClient, RStudioServer, and RStudioMonitor modules uses message packets to communicate with each other over the network. This section describes the structures of the different types of network packets in detail.

Figure 7-3 indicates the basic packet format for all network messages. Each packet contains a type field and a data field. The type field is one byte in size and is used to identify the packet type. The data field stores the content of the message; the size of this field can be inferred from the size of the packet received.

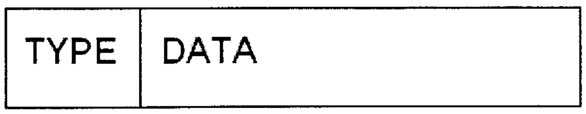


Figure 7-3: Basic packet format

The client connection packet is used by the RStudioClient module to establish connection with RStudioServer. Figure 7-4 depicts the structure for this packet. The type field of the connection request packet has value 0x01. The UID field stores a randomly generated number. The name field stores the actual name of the trader. Note that the name and UID fields together form a useful mapping pair. With this mapping, one can use the UID to identify the trader.

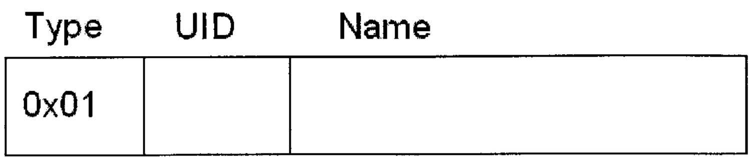


Figure 7-4: Client connection packet structure

Another use of the client connection packet is for RStudioServer to notify the RStudioMonitor modules of new RStudioClient connections. When a new connection between RStudioClient and RStudioServer is established, RStudioServer sends a client connection packet to every RStudioMonitor module. In this way, the monitor modules are informed about every new RStudioClient connection.

The monitor connection packet is used by an RStudioMonitor module to establish connection with RStudioServer. Figure 7-5 describes the structure of the packet. This packet only has a type field, which has value 0x02.

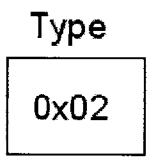


Figure 7-5: Monitor connection packet structure

The data transfer packet is used to transfer physiology data among the client, server, and monitor modules. Figure 7-6 shows the structure of this packet in detail. The type field of this packet has value 0x03. The UID field uses a unique identifier to identify the trader. The preprocessing variable field stores physiology data for each of the preprocessing variables. When an RStudioClient module collects and preprocesses physiology data, it sends a data packet to RStudioServer. RStudioServer then retransmits the data packet to every RStudioMonitor module. With this mechanism, every RStudioMonitor module can receive preprocessed physiology data from every RStudioClient module.

7.4 Time synchronization

Data collected from different machines contain time discrepancies. These discrepancies are contributed by the slight differences in the system clock readings in the machines. In order to account for these discrepancies, RStudio provides a simple

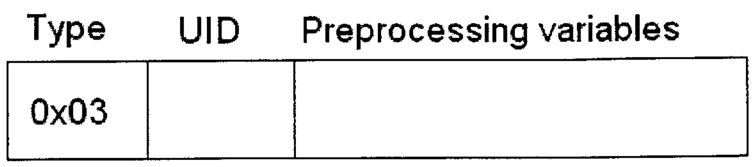


Figure 7-6: Data transfer packet structure

and easy way to perform time synchronization using the event and message logging system.

The event and message logging component provides a special synchronization marker that uses network signals to synchronize between two machines. When the user places a synchronization marker on one machine, the machine records the local system time and then send a network message to the other machine. The second machine receives the network message and records the current system time. By comparing between the times recorded from the two machines, one can adjust for the time discrepancies in the data collected from these machines.

Chapter 8

Monitor System

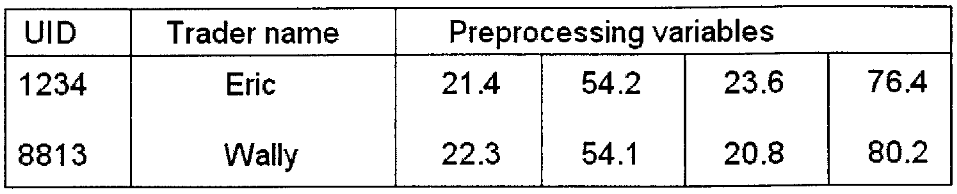
RStudioMonitor is an application for monitoring the physiological states of traders. This system collects preprocessed physiology data from the RStudio network and presents them to the user. The system also monitors the physiological condition of each trader; visual alerts are given for traders with abnormal levels of physiological signals. In addition, RStudioMonitor calculates the aggregate physiological state of the participating group of traders based on their physiological conditions.

8.1 Network retrieval of data

RStudioMonitor collects preprocessed physiology data from RStudioServer. By maintaining a network connection with RStudioServer, the monitor module receives the physiology data that are collected from every RStudioClient module.

RStudioMonitor maintains a data table that contains the most recent preprocessed physiology data for each trader. As new data transfer packets arrive from RStudioServer, the data table is updated with new information. Figure 8-1 shows the structure of the data table.

From the data table, the UID and trader name fields uniquely identify the trader. The preprocessing variable field stores the values of each preprocessing variable defined in Table 6.1. These values are used later by the user interface and the system-monitoring processes.



UID	Trader name	Preprocessing variables			
1234	Eric	21.4	54.2	23.6	76.4
8813	Wally	22.3	54.1	20.8	80.2

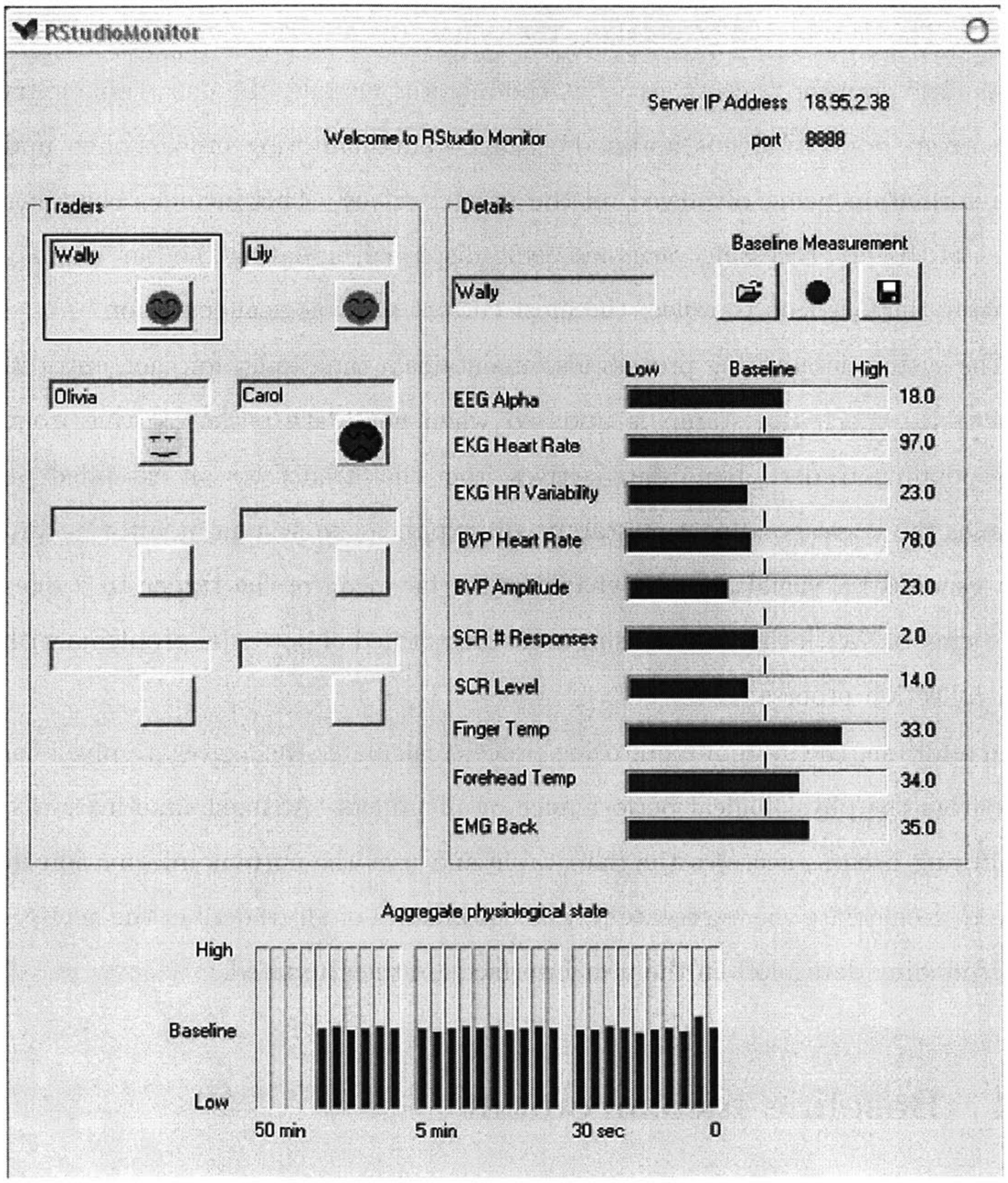
Figure 8-1: Data table in RStudioMonitor

8.2 User interface

Figure 8-2 demonstrates the user interface of the RStudioMonitor module. The left side of the screen shows a list of the traders being monitored. The face indicators provide general information about the physiological performance of the trader. A green happy face icon represents good physiological performance relative to baseline values. A yellow face icon indicates normal physiological state relative to baseline. A red or purple face icon, on the other hand, represents abnormal physiological conditions which may require attention from the user; these icons act as visual alerts.

The right half of the screen displays detailed information about a selected trader. The values of the preprocessing variables for this trader are shown in this screen in both bar graph form and numeric form. The user can switch between displaying different traders by clicking on the corresponding face icons on the trader list. In addition, the user can use the baseline control toolbar to measure baseline values for the selected trader.

The bottom portion of the screen displays the aggregate physiological state of all traders in the RStudio network. The bar graphs show the mathematical average of the physiological states of all traders in the group. The rightmost graph displays the aggregate physiological state in intervals of three seconds. The center graph displays the same information in intervals of thirty seconds. The leftmost graph displays this information in five-minute intervals. The three graphs together cover a time frame of


RStudioMonitor

Server IP Address 18.95.2.38

Welcome to RStudio Monitor

port 8888

Traders

Wally

Lily

Olivia

Carol

Details

Baseline Measurement

Wally

	Low	Baseline	High
EEG Alpha			18.0
EKG Heart Rate			97.0
EKG HR Variability			23.0
BVP Heart Rate			78.0
BVP Amplitude			23.0
SCR # Responses			2.0
SCR Level			14.0
Finger Temp			33.0
Forehead Temp			34.0
EMG Back			35.0

Aggregate physiological state

Figure 8-2: User interface in RStudioMonitor

more than fifty minutes.

8.3 System monitoring

RStudioMonitor performs routine system monitoring of the physiological states of all traders. This is achieved by checking the values stored in the data table.

As data transfer packets arrive at the monitor module, the data table entry for the corresponding trader is updated. The system-monitoring process then updates the information being displayed on the user interface. This includes changing the values of the preprocessing variables being displayed, updating the bar graphs, and changing the face icon to reflect the most current physiological condition.

The system-monitoring process also maintains a timestamp for each entry in the data table. Each timestamp is updated when new data packets arrive from the corresponding trader. If no data arrives from that trader for an extended period of time, the corresponding timestamp will expire. The system-monitoring process then generates a visual alert by changing the face icon of the trader to a question mark symbol. With this mechanism, users are notified of potential problems with the data-collection process.

In addition, the system-monitoring process calculates the aggregate physiological state from the physiological performance of all traders. At fixed time intervals, the monitoring process accesses the data table and uses the current information in the table to recalculate the aggregate physiological state of all traders in the group. The corresponding data plots in the user interface are then updated.

8.4 Baseline measurement

The physiological performance of a trader is calculated by comparing the values of the preprocessing variable for the trader against baseline values. Although RStudioMonitor provides default baseline values for each preprocessing variables, the actual baseline values for each trader may differ.

RStudioMonitor provides an efficient way to measure baseline values. The baseline control buttons on the user interface enable the user to perform baseline measurements for the selected trader. During baseline measurement, new data collected from the trader are used to update the baseline values for the trader. After baseline measurement has stopped, the new baseline values for the trader can be saved to a file. At later experiments, the user can load these baseline values from the baseline file.

The baseline file for each trader contains the baseline values of each preprocessing variable from an accumulation of previous baseline measurements. As more baseline measurements are performed on a trader, the corresponding baseline values for the trader will be more accurate and robust.

Chapter 9

Conclusion and Future Work

RStudio provides the technology to aid research in the link between physiology of traders and their decision-making processes. The modular design of this software is presented. RStudioClient collects, preprocesses, and displays physiology data. RStudioServer enables network sharing of data across multiple computers. RStudioMonitor provides a monitor and alert system for trader physiology. The three modules together form the core of the RStudio system.

This project points to many directions for future work. Firstly, the data-collection component in RStudioClient can be further optimized by being integrated with the next version of the ProComp data-collection equipment. The second version of ProComp provides many new features such as an adjustable data collection rate for each data channel.

Secondly, the RStudio system can be integrated with other existing research projects at the Laboratory for Financial Engineering. For example, one of the current research projects involves inducing emotions through manipulating stock prices in a financial market simulation. RStudio can provide a real-time system for measuring these emotions and visualizing them.

Thirdly, the RStudio system may be further developed into an application for managers on the trading floor. The managers can use the RStudio system to monitor the physiological states of their traders. In addition, RStudio can be used as a screening tool for job applicants as well as a training tool for new traders.

Bibliography

[1] Andrew Lo and Dmitry Repin. The Psychophysiology of Real-Time Financial Risk processing. *Journal of Cognitive Neuroscience,* 14:3, pp. 323-339, 2002.

[2] Karen Quigley, William Ray, and Robert Stern. *Psychophysiological Recording, second edition.* Oxford Press, 2001.

[3] Lewis Napper. *WinSock 2.0.* IDG Books Worldwide, December 1997.

[4] Bob Quinn and Dave Shute. *Windows Sockets Network Programming.* Addison-Wesley Publishing Company, November 1995.

[5] Feng Yuan. *Windows Graphics Programming.* Pearson Education, 2000.

[6] Steve McConnell. *Rapid Development.* Microsoft Press, 1996.

[7] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns.* Addison-Wesley, 1994.

[8] Larry Wood. *User Interface Design.* CRC Press, 1997.

[9] Aaron Cohen, Ronald Petrusha, and Mike Woodring. *Win32 Multithreaded Programming.* O'Reilly and Associates, 1997.