

**VIRTUAL TERMINAL**

**By**

**LUCAS TANDANG**

**Bachelor of Engineering**

**Trisakti University**

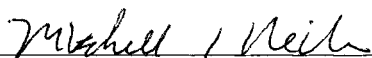
**Jakarta, Indonesia**

**1992**

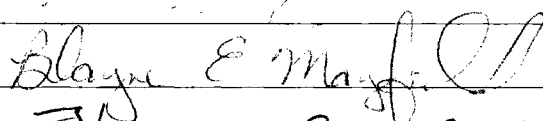
**Submitted to the Faculty of the  
Graduate College of the  
Oklahoma State University  
in partial fulfillment of  
the requirements for  
the Degree of  
MASTER OF SCIENCE  
December, 1995**

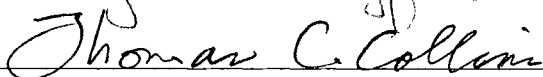
VIRTUAL TERMINAL

Thesis Approved:



Thesis Advisor





Dean of the Graduate College

## ACKNOWLEDGEMENTS

I would like to take this opportunity to thank my advisor Dr. Mitchell L. Neilsen for his invaluable guidance. His patience and encouragement helped me make this thesis work an enjoyable and memorable experience.

I would like to thank Dr. Blayne Mayfield for serving on my graduate committee and providing me with support for my thesis. I would like to thank Dr. Marvin Stone for the help and technical support he provided to make this thesis come true.

Last but not least, I would like to thank my parents, Mr. and Mrs. Tandang, and my girlfriend, Christian Orviesti, for their support.

## TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.....	1
1. J1939 Networks.....	1
2. Virtual Terminals.....	4
2.1 Background.....	4
2.2 Technical Features.....	5
2.2.1 Screens.....	5
2.2.2 VT Keys.....	6
2.2.3 Input Focus.....	6
2.2.4 Alarms.....	6
2.2.5 Addressing.....	7
II. THE VT PROTOCOL AND ARCHITECTURE.....	9
1. Protocols.....	9
1.1 Registration.....	9
1.2 Operation.....	10
1.2.1 Solicited Display.....	10
1.2.2 Unsolicited Display.....	12
1.3 Disconnecting.....	13
2. The Architecture.....	14

III. SCREEN MANAGEMENT .....	17
1. Introduction.....	17
2. Definitions.....	18
3. Memory Management.....	21
3.1 Contiguous Allocation.....	21
3.2 Fixed Size Block Allocation.....	22
4. Screen Replacement Policies.....	23
4.1 Optimal Screen Replacement Algorithm.....	24
4.2 Priority-based Replacement Algorithm.....	24
4.3 Least Recently Used (LRU) Policy.....	25
4.4 Not Frequently Used (NFU) Policy.....	26
4.5 Modified Least Recently Used (MLRU) Policy.....	27
4.6 Priority Over LRU (PLRU) Policy.....	29
5. Analysis.....	30
IV.CONCLUSION.....	35
REFERENCES.....	37

## LIST OF FIGURES

Figures	Page
1.1 J1939 network .....	2
1.2 The J1939 29-bit identifier.....	2
2.1 Solicited display messages.....	11
2.2 Alarm messages.....	13
2.3 Alarm display area.....	14
2.4 Architecture .....	14
3.1 Memory fragmentation.....	21
3.2 Linked list structure.....	22
3.3 VT memory consumption.....	30
3.4 Navigation path of a network device.....	31
3.5 Navigation path.....	32

## **CHAPTER I**

### **INTRODUCTION**

#### **1. J1939 Networks**

J1939 is a Class C type communication network designed to interconnect a set of electronic control devices in a distributed real-time control environment. It provides a standardized communication framework for on-board and off-board mobile equipments. There are variety of mobile equipment applications, including:

- Tractors and implements (Agriculture).
- Construction equipment.
- Trucks and trailers (On-Highway).

J1939/72 defines a communication protocol for standardized display terminal called virtual terminal (VT) [1]. A virtual terminal is a specialized node attached to a J1939 network whose primary function is to provide an operator interface to other network devices or controllers. Figure 1.1 shows the use of a J1939 network in an agriculture application. The network device of particular interest is the virtual terminal (VT).

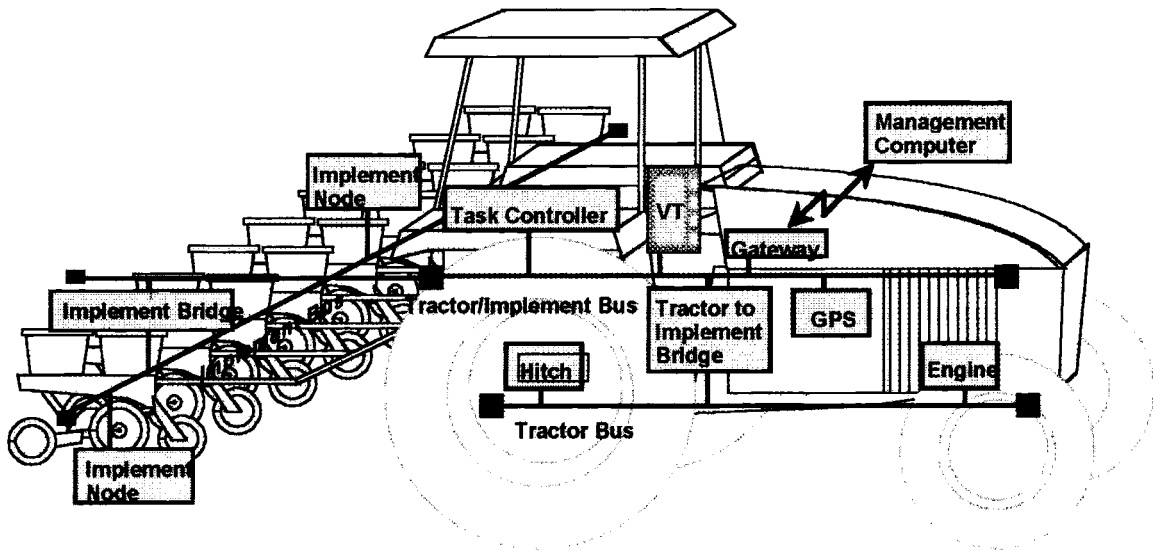


Figure 1.1 J1939 network in agriculture application

J1939 uses the Controller Area Network (CAN) Protocol 2.0B by Bosch, GmbH to perform its functions. The choice of CAN as the underlying protocol is justified because of its features, including:

- Prioritization of messages: messages with the highest priority get transmitted first.
- Short message latency.
- High network bandwidth: J1939 transmits data at 250 Kbps.
- Inexpensive.
- Simple and easy to implement.

Figure 1.2 shows the mapping of J1939 to the 29-bit CAN message identifier.

CAN EXTENDED FRAME FORMAT	S O F			IDENTIFIER 11 BITS											S R R			I D E			IDENTIFIER EXTENSION 18 BITS																		R T R																				
J1939 FRAME FORMAT	S O F			PRIORITY			R	D	P	PDU FORMAT (PF) 6 BITS (MSB)						S R R			I D E			PF (CONT.)	PDU SPECIFIC (DESTINATION ADDRESS, GROUP EXT. OR PROPRIETARY)														SOURCE ADDRESS						R T R																
J1939 FRAME BIT POSITION	3	2	1	0	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	33														
CAN 29 BIT ID POSITION	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	33

Figure 1.2 J1939 29-Bit identifier.



The priority field (the first 3 bits) is used for message prioritization. It determines the priority of the message. A message in the highest priority class has 000 bits in its priority field. High priority messages are used for time critical messages such as torque control from the transmission to the engine. The transmission has to inform the engine to reduce the torque during gear switching quickly to produce smooth gear switching and reduce clutch wear. Therefore, the 000 priority is required to guarantee short message latency. The lower priority messages are of non-time critical messages. The virtual terminal messages are included in this group with the priority of 111 ( the lowest). There does not exist a rigid message deadline requirement for VT messages. The only requirement that puts a limit on message latency for VT messages is human perception to a change, which according to automotive research is not less than 50 milliseconds. As an example, with the current data rate of J1939 (250 Kbps), and assuming that every message is full (8 data bytes, 128 bits total message length), in the worst case each message can be delivered in 0.5 milliseconds. If the operator presses a key in the virtual terminal, the VT message must be delivered within 50 ms. So, the message can wait for approximately 99 other higher priority messages to be delivered, assuming that the processing time is negligible.

The 1-bit Data Page (DP) field and 8-bits Protocol Data Unit (PDU) Format field are used to identify the data that is contained within the message. For example 'node to VT' messages (messages that originated from VT and destined to a node) have DP set to 0 and a PDU Format of 230, and 'VT to node' messages have DP set to 0 and a PDU Format of 231.

The next 8-bit field is the PDU Specific field(PS) . It may contain destination address or a group extension to the PDU format. If the PDU format value is between 0 and 239, the PS field contains the destination address of the recipient, and if the value is between

240 and 255, the PS field contains the group extension to the PDU format. The group of J1939 messages whose PDU format value is between 0 and 239 are called 'destination specific messages'. These messages are not broadcast messages. They are intended to be directed toward a specific node in the J1939 network whose address match the destination address field. The virtual terminal messages are included in this group of messages.

The last 8 bits of the 29 bit J1939 identifier is the source address of the network device transmitting the message. It is evident that with 8 bits source address there could be 256 different network device participating on one J1939 network.

## **2. Virtual Terminals**

### **2.1. Background**

The virtual terminal is a specialized node attached to a J1939 network whose primary purpose is to provide the operator with an interface to other nodes in the network. The functional relationship between the nodes (network devices) and the VT can be somewhat illustrated as a client-server relationship. Therefore, the VT performs its task by executing a predetermined set of commands that are sent by the network devices. The network devices, as the masters, can instruct the VT to perform any set of commands according to their specific needs. The VT generally enables the network devices to:

- Establish a dialogue with the operator.
- Display information to the operator.
- Retrieve information from the operator.

The VT is a 'dump terminal', because it does not provide any direct computing power to the network or the system as a whole.

The use of a virtual terminal to control J1939 network applications has several advantages:

- Monitoring capabilities for the operator over all network devices.
- The network devices can request for user/operator inputs.
- Single point workplace. All communications to the operator can be accomplished through the virtual terminal. Vehicle ground speed, engine rotation per minute (rpm), body electronics (lamps, door lock, etc) operations are the typical information that can be communicated to the operator through the VT.
- Reduce the wiring harness. While the cost of the wire itself is relatively cheap, the manufacturing cost of installing complex wiring system on the driver's panel is substantial, especially if the vehicle is short of physical space. With the use of the VT the only connection that is required is just a single tap into the J1939 bus.

The VT allows multiple network devices to share the terminal. Each network devices should be able to freely communicate to the VT without being aware of the existence of other nodes who currently communicate with the VT.

## 2.2. Technical Features

### 2.2.1. Screens

One network device may define more than one screen. The VT is capable of interactively displaying the screens from multiple network devices. The current

implementation of the VT allows one screen from one network devices to be displayed at a time. The operator may select the screen according to his/her needs.

#### 2.2.2. VT Keys

The currently proposed implementation of VT has 8 function keys, a numeric keypad, and a select network device (SND) key. The 8 function keys will send a fixed code if pressed (depressed). The SND key provides the operator with the ability to select the network device to be displayed. Whenever the SND key is pressed, the VT will display a choice of network devices for the operator to select. For example:

press F1: Network Device 1

press F2: Network Device 2

#### 2.2.3. Input Focus

The VT keypad and function keys input focus goes to the network devices whose screen is currently displayed.

#### 2.2.4. Alarms

The VT provide the capability to display alarm conditions. Any network device recognized by the VT can send their alarm anytime, regardless of which network device currently owns the screen. If more than one alarm condition should occur, the VT displays the alarm with the highest priority.

### 2.2.5. Addressing

All VT messages use a destination specific message format in J1939. Thus, each message packet has a destination address field, in addition to its source address field, embedded in its 29-bit message identifier. This requirement assists the VT to destine its message only to the intended network devices for 'VT to node' messages. On some other application where multiple VTs may exist within the network, each individual network device may direct its messages to a specific VT on 'node to VT' messages.

VT messages use two destination specific messages. They are messages that are directed from VT to network devices (VT to node) and the messages that are directed from network devices to VT (node to VT). VT to node messages parameters; include:

- Transmission Repetition Rate: As Required
- Data Length: 8 Bytes
- Data Page field: 0
- PDU Format field: 230
- PDU specific field: Destination Address
- Default Priority: 7
- Byte: 1 VT function number. Bit 0-3 Parameter  
4-7 Command

2-8 Data necessary to support the function.

Node to VT messages parameters:

- Transmission Repetition Rate: As Required
- Data Length: 8 Bytes
- Data Page field: 0
- PDU Format field: 231

- PDU specific field: Destination Address
- Default Priority: 7
- Byte: 1 VT function number. Bit 0-3 Parameter  
4-7 Command  
2-8 Data necessary to support the function.

## **CHAPTER II**

### **THE VT PROTOCOL AND ARCHITECTURE**

#### **1. Protocol**

##### **1.1. Registration**

Every network device needing to interact with the VT is required to inform the VT of its presence during power up. This process is called 'registration'. The registration process of a new network device is initiated by a 'start registration' message, followed by a string of character indicating the network device's name, and concluded with an 'end registration' message. Two 'start registration' messages from two different network device received by the VT consecutively should not lead to confusion, since each message will bear the source addresses of the corresponding network device. Thus, the VT should be able to manage the two simultaneous registrations.

The VT uses the registration information to distinctly recognize each individual network device needing to interact with the VT (by their address). Unless each network device sends its registration during power up, all request made to the VT will be denied. The VT uses the network device's registration also to identify the network device's name. This network device's name will be put in the VT menu entry. The operator can access the VT menu and select whichever network device he chooses.

The registration process may happen any time, so for example even if the VT and a number of network devices have already been in operation for several hours and suddenly a new network device joins the network, the VT will still be able to process the registration.

If for some reason the registration messages fail to reach the VT , the network device's name will not appear in the menu entry. At this point the operator will notice the problem and should be able to take proper actions (reboot the network device).

## 1.2. Operation

A network device can enter the operation stage only after it has completed registration. Note that the operation stage here means the stage where a network device can request its screens to be displayed, it has nothing to do with network device's individual operation. There are two types of displays:

- Solicited displays: operator's request displays.
- Unsolicited displays: alarm displays.

### 1.2.1. Solicited Display

Solicited displays occur upon operator request. Figure 2.1 shows the protocol for solicited displays.



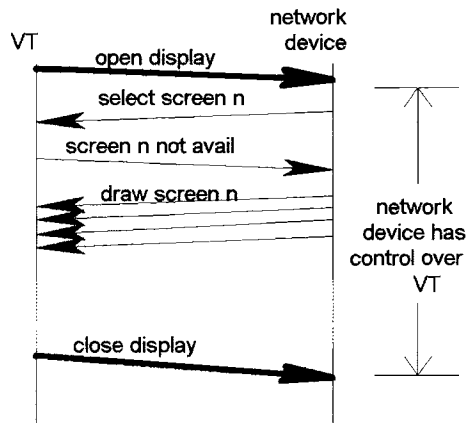


Figure 2.1. Solicited display messages.

When the user selects a network device from the menu, the VT sends an 'open display' message to the corresponding network device. Similarly, when the user presses the SND key, the VT sends a 'close display' message to the network device. The network device has absolute control over the VT in the time interval between the receipt of an 'open display' message and a 'close display' message. All VT resources including display area and keys are exclusively owned by the selected network device during this time interval. It essentially means that:

- Only the selected network device can draw display area.
- All VT keys press/depress events will be directed toward the selected network device (the selected network device has the focus).

The receipt of a 'close display' message by the network device is equivalent to the VT telling the network device to be quiet. If the network device still sends messages after the receipt of a 'close display' message, the VT will subsequently ignore them. The alarm display area is an exception. Every registered network device should be able to display alarm message (ascert alarm) any time regardless of which device currently controls the VT.

An 'open display' message is equivalent to the VT giving rights to the network device to exclusively use the VT resources (except the alarm display area). The network device should subsequently request the VT to display its initial screen (select screen n). The choice of initial screen to display is left to the network device designer to decide. A possible decision would be to always display a particular screen whenever a network device is selected. This particular screen would likely to be the main screen for the network device.

A request to display a screen is indicated by the receipt of a 'select screen n' message. The VT could directly satisfy the request by displaying the requested screen if the screen is currently stored in VT memory. If not, the VT should send the reply back, stating that the requested screen is unavailable (screen n not available). The network device can correspondingly send the screen definition over the network. A screen is defined by a screen macro. A screen macro is a sequence of commands.

#### 1.2.2. Unsolicited Display

Unsolicited displays are alarm displays. They don't occur upon operator's request. The network device should be able to display alarm condition anytime. Figure 2.2 shows the protocol.



give any response, thus the VT will not draw anything to the screen). Up to this point, the operator will be able to notice the problem. The current implementation of the VT always keeps the menu entry for an already disconnected network device. If the network device shall ever come back on-line again, it will occupy its previous menu entry.

## 2. The Architecture

The VT has several resources:

- Regular display area: the drawing and writing canvas for the network devices.
- Keys: 8 function keys, a select network device key, and a set of numeric keys.
- Alarm display area: separate display area located at the very bottom of VT screen, used exclusively for displaying alarm message from the network devices.

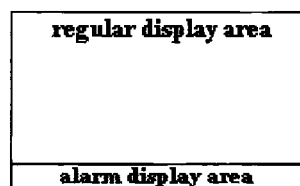


Figure 2.3 Alarm display area, located at the very bottom of VT screen.

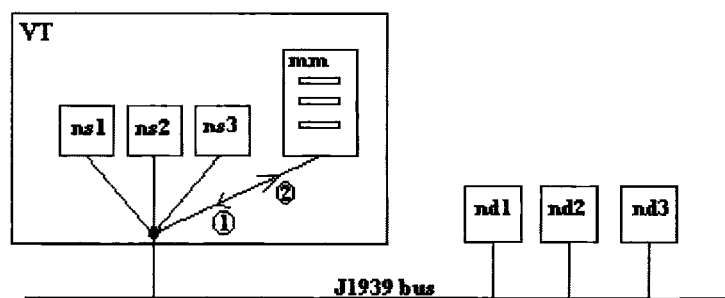


Figure 2.4 Architecture of the VT.

There are two authorities inside the VT that can control resources:

- Menu Manager (MM): MM is responsible of managing the network device selection process (the menu).
- Network device servers (ns1,ns2,ns3): Network device servers (ns) service network devices on one to one basis. For example, ns1 services network device 1 (nd1), ns2 services nd2, and ns3 services nd3, and so on. All messages coming from nd1 are exclusively serviced by ns1. All messages generated by network device server 1 (ns1) are automatically directed toward network device 1. A network device server is dynamically created and disposed in conjunction with the presence or absence of the network device it services. The multiple network device servers only constitute a logical organization of processing. They are not concurrent processes.

The MM takes care of the registration process. The MM captures registration process messages from each network device (2 in Figure 2.4) to build the menu. If the operator selects a network device from the VT menu, the MM will perform two actions:

- Send 'open display' message to the selected network device (1 in Figure 2.4).
- Give up its authority to the selected network device server. For example if the operator select nd1 from VT menu, the authority to the VT will be switched to ns1 , and thus, it is only ns1 which has the exclusive use over the VT resources. At this point, it is said that the ns1 status is 'alive'. Only one ns is 'alive' at one time.

The network device server receives a macro definition that is sent by its master and stores the macro to the pool of macro definitions. When The ns receives a select macro command, it searches the desired macro from the pool and executes the macro. If the macro is not found, the ns will respond by sending 'macro not available' message back to its master. The ns also services the user events (key press/depress) and directs the events to its master. When the operator presses the SND key, the ns will perform two actions:

- Send 'close display' message to its master.
- Give up its authority back to MM, and thus, its status is no longer 'alive'. The MM takes over the possession of VT resources (except for alarm display area). At this point the VT is not connected to any network device. The menu for network device selection will appear on the VT regular display area.

## **CHAPTER III**

### **SCREEN MANAGEMENT**

#### **1. Introduction**

The operator communicates with a network device by switching to a screen associated with the device. Each screen is associated with a single network device, but a network device may have more than one screen. A screen is defined by a screen macro. A screen macro is a sequence of commands. The VT displays a screen by executing the corresponding screen macro. A screen macro must be completely loaded in VT memory before the VT can display the screen. Therefore, the VT should have enough memory to store at least one screen macro. A screen and the corresponding screen macro will be referred to as a screen when no confusion will result.

Each network device can have more than one screen. Conceptually, the VT can display an infinite number of screens from multiple network devices. The VT can secure the screens within the VT's memory and display the screen to the operator on demand without requiring the network device to re-load the screen macro over the network. However, since the amount of VT memory is limited, only the subset of all screen macro can reside in VT memory at any given time. A screen macro replacement algorithm is used to judiciously select which screen currently in memory should be ejected when another screen macro needs to be loaded.

## 2. Definitions

The set of all screens from all network devices is defined as the *screen set* and is denoted  $S = \{s_i \mid i = 1, 2, 3, \dots, m\}$ . Operator screen references are defined by a reference string,  $\omega = r_1, r_2, r_3, \dots, r_T$ , where  $r_t \in S$  for  $1 \leq t \leq T$ . Note that the VT displays a screen in the foreground by executing the corresponding screen macro. The VT records a screen reference each time the corresponding screen macro is executed. The resident set  $R_t$  is the set of screens present in VT memory after screen  $r_t$  is referenced at time  $t$ . Note that  $R_t \subseteq S$ .

During operation of the VT, a resident set  $R_t$  is constructed on each reference  $t$  such that  $r_t \in R_t$ . Now, let  $\text{Size}(s_j)$  be the size of screen  $s_j$  in bytes. The size of  $R_t$  is limited by the amount of VT memory, denoted  $\text{VT\_RAM}$ ; that is,

$$\sum_{s_j \in R_t} \text{Size}(s_j) \leq \text{VT\_RAM}.$$

Obviously, the perfect condition occurs when  $\sum_{s_j \in S} \text{Size}(s_j) \leq \text{VT\_RAM}$ . In this case,

the VT can always store all screens from all network devices at all times. Unfortunately, this will generally not be the case. So, a judicious screen management policy must be employed to decide which screens are eligible to stay in VT memory at any given time.

A *screen fault* occurs when the operator tries to reference a screen that is not presently in VT memory. The screen fault indicator function  $F_t$  is defined as follows:

- $F_t = 1$ , if a screen fault occurs when reference  $r_t$  is performed. Note that  $F_t = 1$  if  $r_t \notin R_{t-1}$
- $F_t = 0$ , if a screen fault does not occur when reference  $r_t$  is performed. Note that  $F_t = 0$  if  $r_t \in R_{t-1}$ .



When a screen fault occurs, the screen macro is sent from the corresponding network device to the VT over the network. In this paper, we introduce a demand-driven strategy that can be used to manage memory in a virtual terminal. The network devices are not required to initially download the screens on power up. The screens are brought into VT memory only when the operator references them. The demand driven strategy is defined as follows:

- $R_t = R_{t-1}$ , if  $r_t \in R_{t-1}$ .
- $R_t = R_{t-1} \cup \{r_t\}$ , if  $r_t \notin R_{t-1}$  and  $(\sum_{s_j \in R_{t-1}} \text{Size}(s_j) + \text{Size}(r_t)) \leq \text{VT\_RAM}$
- $R_t = R_{t-1} \cup \{r_t\} - P_{t-1}$ , if  $r_t \notin R_{t-1}$  and  $(\sum_{s_j \in R_{t-1}} \text{Size}(s_j) + \text{Size}(r_t)) > \text{VT\_RAM}$

where  $P_{t-1}$  is the set of replaced screens ( $P_{t-1} \subseteq R_{t-1}$ ). Note that  $P_{t-1}$  may have more than one element when the size of  $r_t$  is such that more than one screen is replaced to make room for  $r_t$ .

The following examples illustrates the demand-driven strategy.

t	1	2	3	4	5	6	7	8	9	10	11	12
$\omega$	1	2	1	2	1	2	3	4	1	2	1	2
$\text{Size}(r_t)$	1	2	1	2	1	2	1	2	1	2	1	2
$R_t$	1	1	1	1	1	1	3	3	1	1	1	1
		2	2	2	2	2	2	4	4	2	2	2
		2	2	2	2	2	2	4	4	2	2	2
$F_t$	1	1	0	0	0	0	1	1	1	1	0	0

When screen 1 is referenced at time 1, a screen fault occurs and screen 1 is loaded into memory. Thus  $R_1 = \{1\}$ , and  $F_1 = 1$ . At time 3, screen 1 is referenced for the second time and no fault occurs. Finally, at time 7, one of the screens currently in memory must be evicted to make room for screen 3. In the example, screen 1 is evicted. The screen or screens selected for eviction is determined by the screen replacement *policy* used.

In order to better understand the policies that can be used to select which screen is evicted the following formalism is required.

The number of bytes transferred on reference  $t$  is defined by:  $D_t = F_t * \text{Size}(r_t)$ . So, the average number of bytes transferred for each reference is

$$A_v = \frac{\sum_{t=1}^T D_t}{T} = \frac{\sum_{t=1}^T F_t * \text{Size}(r_t)}{T}$$

The maximum average number of bytes transferred for each reference occurs when a screen fault occurs on every reference. The upper bound on  $A_v$  is given by:

$$A_{v_{\max}} = \frac{\sum_{t=1}^T \text{Size}(r_t)}{T}.$$

The minimum average number of bytes transferred for each reference occurs when the VT can afford to store all of the referenced screens in its memory and thus, for each screen, the fault occurs only once. The subsequent references to the screen will not cause a screen fault. If all screens are referenced at least once, the lower bound on  $A_v$  is given by:

$$A_{v_{\min}} = \frac{\sum_{s_j \in S} \text{Size}(s_j)}{T}$$

However, in general the lower bound is given by:

$$A_{v_{\min}} = \frac{\sum_{t=1}^T F_t * \text{Size}(r_t)}{T}, \text{ where:}$$

- $F_t = 0$  if there exists an  $i$ ,  $1 \leq i < t$ , such that  $r_i = r_t$ .
- $F_t = 1$ , otherwise.

### 3. Memory Management

Several different methods can be used to manage memory allocation within a VT.

Two possible methods are presented below.

#### 3.1 Contiguous Allocation

The VT stores variable size screen macros in its memory. If the screen macros are stored in a contiguous area of memory, external fragmentation may occur. Then, frequent memory compaction may need to be performed. The following example illustrates the problem.

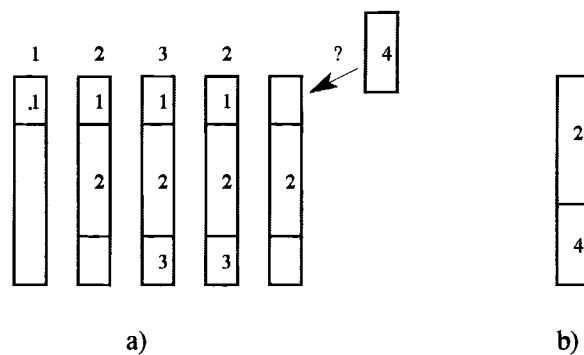


Figure 3.1 Memory fragmentation.

In Figure 1a, after screen 1 and screen 3 are evicted, screen 4 still cannot be stored in memory because the memory is fragmented. Figure 1b shows the result of memory compaction. If memory compaction is executed frequently the VT performance may be adversely affected.

### 3.2 Fixed Size Block Allocation

A screen macro is a sequence of VT commands. The maximum length of a VT command is 8 bytes [1]. If each VT command is stored in an 8-byte frame and a linked list structure is used to store a screen macro, the fragmentation problem will not occur.

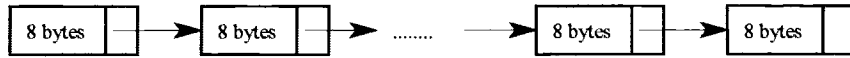


Figure 3.2 A screen macro stored as a linked list.

With a linked list structure, the memory fragmentation will not occur because each 8-byte frame can be located anywhere in the memory. The first frame points to the second frame, the second frame points to the third frame and so on.

The problem with the linked list strategy is the space required to maintain the pointer in each frame. With a 32-bit (4 byte) pointer, the wasted space is  $4/(8+4) = 33.3\%$ . However, the wasted space can be minimized simply by using a larger block size. Let  $p$  be the block size in frames (1 block =  $p$  frames =  $8 * p$  bytes). Assuming that each block is fully occupied, the wasted space is given by:

$$\frac{4}{8 * p + 4} = \frac{1}{2 * p + 1}$$

In practical, some blocks are not fully occupied. For example, if a screen requires 20 frames and the block size is 8 frames, then 3 blocks will be allocated. The last four frames are not used. This is called internal fragmentation.

Independent of which method is used to manage memory allocation, a screen replacement policy must be implemented to determine which screens should remain in memory.

#### 4. Screen Replacement Policies

If enough memory is not available to bring a new screen into memory, the VT must evict some screens to make room. Recall from above that

$$R_t = R_{t-1} \cup \{r_t\} - P_{t-1}, \text{ if } r_t \notin R_{t-1} \text{ and } \left( \sum_{s_j \in R_{t-1}} \text{Size}(s_j) + \text{Size}(r_t) \right) > \text{VT\_RAM},$$

where  $P_{t-1}$  is the set of replaced screens. The construction of  $P_{t-1}$  can be performed as follows:

1. Initially  $P_{t-1}$  is empty.
2. Select one screen to be replaced, say screen  $y \in (R_{t-1} - P_{t-1})$ . Formally, we write  $y = \text{Sp}(R_{t-1} - P_{t-1})$ . The selection process (Sp) depends on the particular replacement policy as described below.
3. Let  $P_{t-1} = P_{t-1} \cup \{y\}$
4. If  $\sum_{s_j \in (R_{t-1} - P_{t-1})} \text{Size}(s_j) + \text{Size}(r_t) \leq \text{VT\_RAM}$ , then stop; that is, stop if there is enough memory after evicting the screen.
5. Go to Step 2.

Upon completion of the algorithm,  $P_{t-1}$  contains all screens that have been evicted.

Given a reference string  $\omega = r_1, r_2, r_3, \dots, r_T$ . The *forward distance* of screen  $y$  at time  $t$  ( $1 \leq t \leq T$ ) is defined as:

- $\text{dist}_f(y, t) = k$ , if  $r_{t+k}$  is the first occurrence of  $y$  in  $r_{t+1}, r_{t+2}, \dots, r_T$ .
- $\text{dist}_f(y, t) = \infty$ , if  $y$  does not appear in  $r_{t+1}, r_{t+2}, \dots, r_T$ .

The *backward distance* of screen  $y$  at time  $t$  ( $1 \leq t \leq T$ ) is defined as:

- $\text{dist}_b(y, t) = k$ , if  $r_{t-k}$  is the most recent previous occurrence of  $y$  in  $r_1, r_2, \dots, r_{t-1}$ .
- $\text{dist}_b(y, t) = \infty$ , if  $y$  does not appear in  $r_1, r_2, r_3, \dots, r_{t-1}$ .

In each of the following algorithms, for brevity in exposition we assume that the screen size is fixed.

#### 4.1 Optimal Screen Replacement Algorithm

The screen replacement policy performs best if it can perfectly predict the future screen references. On a screen fault, the screens with the maximum forward distance are evicted. Unfortunately, this algorithm is impractical since the full reference string must be known in advance.

#### 4.2 Priority-based Replacement Algorithm

The network device writer can assign static information for each screen indicating the level of usage of the screen. Then, the replacement policy is to replace screens with lowest priority. The policy is based on the expectation that the lowest priority screens will not be referenced again soon, and thus, should be selected for replacement.

With a priority-based policy, the network designer assigns a priority to every screen indicating the level of usage or importance of the screens. For example, suppose that a particular network device has 5 screens. Since screen 1 contains critical information, the designer predicts that screen 1 will be used most of the time and should be assigned a priority of 0 (the highest). After careful examination of the data contained in each of the remaining screens, the designer assigns screens 2, 3, 4, and 5 with priorities 1, 2, 3, and 3, respectively.

Let us assume that the prediction is correct. The higher priority screens are used more frequently. So, on a particular run the operator may make the following screen references:

t	1	2	3	4	5	6	7	8	9	10	11	12	13
$\omega$	1	2	1	4	1	3	2	1	2	1	5	3	1
$R_t$	1	2	2	4	4	3	3	3	3	3	5	3	3
		1	1	2	2	2	2	2	2	2	2	2	2
				1	1	1	1	1	1	1	1	1	1
$F_t$	1	1	0	1	0	1	0	0	0	0	1	1	0

At time  $t=6$ , a screen fault occurs, and screen 4 is replaced because the priority of screen 4 is the lowest.

The priority-based policy provides good performance if the network device designer has a good idea of which network devices will coexist on the same network.

Unfortunately, that may not be the case. For instance, in the example above, the network device designer already predicted that screen 1 will be the most frequently used screen (0 priority). However, an operator may decide to connect a second network device (D2) on the same network. Suppose that D2 has 2 screens: screen 6 and screen 7 with assigned priorities of 1 and 2, respectively. The two D2 screens may be used far more frequently than screen 1, but instead of storing screen 6 and 7, the VT stores screen 1 (0 priority) which is not used frequently compared with the D2 screens. The result will surely adversely affect network traffic performance.

#### 4.3 Least Recently Used (LRU) Policy

The LRU policy is based on the assumption that recently used screens are likely to be used again in the near future. Conversely, the screens that are not used recently will likely being unused in the near future. The LRU policy causes screens that are recently used to remained in memory. The LRU screen is  $y = \text{Sp}(R_{t-1})$ , where  $\text{dist}_b(y,t) = \max(\text{dist}_b(x,t))$  among all  $x \in R_{t-1}$ . So with the same reference string, the LRU policy will provide the following result:

t	1	2	3	4	5	6	7	8	9	10	11	12	13
$\omega$	1	2	1	4	1	3	2	1	2	1	5	3	1
$R_t$	1	2	1	4	1	3	2	1	2	1	5	3	1
		1	2	1	4	1	3	2	1	2	1	5	3
				2	2	4	1	3	3	3	2	1	5
$F_t$	1	1	0	1	0	1	1	0	0	0	1	1	0

A screen fault occurs at  $t=6$ . The replaced screen is 2 because  $\text{dist}_b(2,6)=\max(\text{dist}_b(x,6))$  among all  $x \in R_5$ ; that is, screen 2 is the least recently used screen. The LRU policy is more convenient to use because the network device designers do not have to worry about assigning a priority for each screen.

#### 4.4 Not Frequently Used (NFU) Policy

On a screen fault, the NFU policy retains the most frequently used screens in memory and replaces a screen that was not used frequently in the past. When a screen fault occurs at time  $t$ , the NFU policy replaces screen  $y=\text{Sp}(R_{t-1})$ , such that  $\text{freq}(y,t)=\max(\text{freq}(x,t))$  among all  $x \in R_{t-1}$ . The problem with NFU is that it never forgets. Suppose that an operation consists of two phases, initialization and execution. The initialization phase uses screens 1 and 2 frequently, and the execution phase uses screens 3 and 4 frequently. If only 2 screens can fit into memory, the result can be described as follows:

t	1	2	3	4	5	6	7	8	9	10	11	12	13
$\omega$	1	2	1	2	1	2	1	3	4	3	4	3	4
$R_t$	1	2	2	2	2	2	2	3	4	3	4	3	4
		1	1	1	1	1	1	1	1	1	1	1	1
$F_t$	1	1	0	0	0	0	0	1	1	1	1	1	1

During the execution phase, screen 1 still resides in VT memory even though it is no longer used. When a screen fault occurs at  $t=10$ , the replaced screen is screen 4 because it was not used frequently. If the LRU policy is employed, the above reference string will result in the following:



t	1	2	3	4	5	6	7	8	9	10	11	12	13
$\omega$	1	2	1	2	1	2	1	3	4	3	4	3	4
$R_t$	1	2	1	2	1	2	1	3	4	3	4	3	4
		1	2	1	2	1	2	1	3	4	3	4	3
$F_t$	1	1	0	0	0	0	0	1	1	0	0	0	0

Screen 1 will be replaced from VT memory as soon as it is no longer used in the execution phase, and only 4 screen faults will occur.

#### 4.5 Modified Least Recently Used (MLRU) Policy

The LRU policy can readily be employed if the VT can display only one screen at a time. If the VT can display multiple screens simultaneously, the LRU policy may yield unexpected results. The following example illustrates the problem:

t	1	2	3	4	5
$\omega$	5	6	7	6	8
$R_t$	5	6	7	6	8
		5	5	5	6
			6	7	7
$F_t$	1	1	1	0	1

Suppose that the VT can display 2 screens simultaneously. After screen 6 at time 2 is referenced, screens 5 and 6 reside in VT memory and both are displayed. At time 3, screen 7 is referenced. Screens 5 and 7 are displayed in the foreground, while screen 6 is stored in the background, not displayed. At time 4, screen 6 is referenced and displayed together with screen 5 while screen 7 is kept in the background. When screen 8 is referenced at time 5, screen 5 is evicted while the operator may be still using it. The problem is that a screen may be referenced (read-only) while simply being displayed. The solution is to consider display in the foreground as a reference.

The MLRU policy maintains a bit vector  $v_i$  for every screen  $i$  that currently resides in VT memory. Using MLRU with a 5-bit vector, the above example will lead to the following result :

1. Screen 5 is referenced at time 1. The MSB is set to 1.

1	0	0	0	0	5
---	---	---	---	---	---

2. Screen 6 is referenced at time 2. Two steps are performed:

- The bit vector is shifted one step right.

0	1	0	0	0	5
0	0	0	0	0	6

- Screens 5 and 6 are displayed in the foreground. The MSBs are set to 1.

1	1	0	0	0	5
1	0	0	0	0	6

- 3 Screen 7 is referenced at time 3.

- The bit vector is shifted one step right.

0	1	1	0	0	5
0	1	0	0	0	6
0	0	0	0	0	7

- Screens 5 and 7 are displayed in the foreground. The MSBs are set to 1.

1	1	1	0	0	5
0	1	0	0	0	6
1	0	0	0	0	7

- 4 Screen 6 is referenced at time 4.

- The bit vector is shifted one step right.

0	1	1	1	0	5
0	0	1	0	0	6
0	1	0	0	0	7

- Screens 5 and 6 are displayed in the foreground. The MSBs are set to 1.

1	1	1	1	0	5
1	0	1	0	0	6
0	1	0	0	0	7

5. Screen 8 is referenced at time 5. Screen 7 is selected for eviction because it has the smallest binary value.

- The bit vector is shifted one step right.

0	1	1	1	1	5
0	1	0	1	0	6
0	0	0	0	0	8

- Screens 5 and 8 are displayed in the foreground. The MSBs are set to 1.

1	1	1	1	1	5
0	1	0	1	0	6
1	0	0	0	0	8

Note that since screen 5 is always displayed in the foreground, it retains the largest binary value and thus, has the absolute right to reside in VT memory. The MLRU policy also uses information about how recently a screen has been referenced. In the above example, screen 6 is used more frequently than screen 8, but since screen 8 is more recently used, screen 8 has a larger binary value than screen 6. The length of the bit vector determines how far back in the past the VT should consider the screen usage record.

#### 4.6 Priority Over LRU (PLRU) Policy

The PLRU policy maintains 2 priority classes:

- High priority (0) for screens that are displayed in the foreground.
- Low priority (1) for screens that remain in memory, but are not displayed.

The PLRU applies LRU policy is used for the screens within the same priority class.

Using PLRU, screens in the foreground are never replaced as long as there are screens in the background. Among screens in the background, the PLRU policy still chooses the least recently used screen to be replaced.

## 5. Analysis

The VT screen management performs well if the operator access pattern exhibits locality of reference; that is to say, the operator will refer to recently used screens more likely than others.

It should be noted that the operator refers to a unit of screen (interactively displaying screens from multiple network devices). And a screen consists of a collection (a set) of fields. The premise is, in order to achieve a good locality of reference, the network device designer should group together the information fields with a high probability of reference. For example, grouping the speed (15%) and fuel level (4%) fields to form a single screen will be a bad design compare to grouping the speed (15%) and engine rpm (8%) together to form a single screen. The latter will result in screen with a 23% probability of reference, while the former will result to a screen with 19% probability of reference.

However, grouping the speed and engine rpm is not a good idea if engine rpm information field takes a lot of VT memory to store. Figure 3.3 shows the comparison.

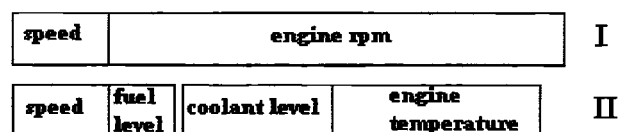


Figure 3.3 VT memory consumption.

Figure 3.3 shows that grouping the speed and fuel level (II) is actually a good design if we consider the VT memory consumption as well. With Arrangement II, we can

conserve VT memory space to fit another screen with engine coolant level (5%) and engine temperature (6%) information in it. With arrangement II, we can have two screens with a total probability of 30% (15% + 4% + 5% + 6%) which is more than Arrangement I (23%), and still consumes less VT memory than Arrangement I.

The operator interacts with the VT by navigating through many different screens from different network devices following the predetermined navigation path. Figure 3.4 shows a simple example of navigation path of a network device with 3 screens.

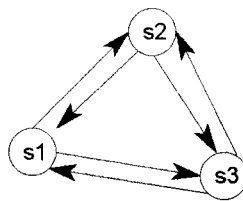


Figure 3.4 Navigation path of a network device.

The above figure shows an example of a network device with 3 fully connected screens. Screen s1 contains vehicle speed information. Screen s2 contains 'area covered' information. Screen s3 contains 'engine rpm' information. Suppose after several runs, the network device designer collected the fact that operator's pattern of interests consists of the following :

1. s1 only: the operator only looks at the speed information (s1).
2. s1s2: the operator looks at the speed and then he wonders how far he has gone, so he looks at the area covered (s2).
3. s1s3: the operator looks at the speed information (s1), and then he wonders how many rpm is the engine currently running, so he looks at 'engine rpm' (s3).
4. s2 only: the operator only looks at 'area covered' .

5. s3 only: the operator only looks at 'engine rpm'.

And suppose, the results of two runs are collected as the following:

operator's interest	1'st run	2'nd run	%
s1 only	21 times	39 times	20%
s1s2	5 times	11 times	5%
s1s3	50 times	100 times	50%
s2 only	5 times	9 times	5%
s3 only	20 times	40 times	20%

Based on the test result, the operator looked at the speed only at 20% probability, look at speed and then area covered at 5%, etc.

Now let's see what happens if the network designer fails to predict the nature of operator's interests and provide links between the three screens as shown at Figure 3.5

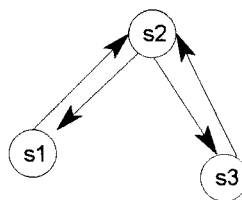


Figure 3.5 Navigation path.

The link to s3 is provided only from s2. So if the operator wishes to look at 'engine rpm', and he's currently at s1, he has to switch to s2 before switching to s3. Consequently, the result would definitely be:

operator's interest	1'st run	2'nd run	%
s1	21 times	39 times	20%
s1s2	5 times	11 times	5%
<b>s1s2s3</b>	50 times	100 times	50%
s2	5 times	9 times	5%
s3	20 times	40 times	20%

It's obvious that the total number of screen references increases by (from the previous table):

$$\frac{P(s1s3)}{P(s1) + P(s2) + P(s3)} = \frac{P(s1s3)}{P(s1_{only}) + P(s1s2) + P(s1s3) + P(s2_{only}) + P(s1s2) + P(s3_{only}) + P(s1s3)}$$

$$= \frac{50\%}{20\% + 5\% + 50\% + 5\% + 5\% + 20\% + 50\%} = \frac{50\%}{155\%} = 32\%$$

Inevitably, as the screens references increase, the more cumulative network traffic will incur no matter how good is the VT screen management. Worse than that, the already planned locality of reference will be misled. Let's say that the VT memory can fit only 2 screens. The network device designer already prearranged that s1 and s3 will be the most heavily used screens, since s1 contains vehicle speed information and s3 contains engine rpm information. They are the two most preferred information. If the first navigation path (fully connected) is used, everything will go perfectly as planned.

$$p(s1) = (20\% + 50\% + 5\%) / 155\% = \underline{48\%}$$

$$p(s3) = (20\% + 50\%) / 155\% = \underline{45\%}$$

$$p(s2) = (5\% + 5\%) / 155\% = \underline{6\%}$$

The result shows a perfect locality of reference of just 2 screens.

Now, if the network designer fails to predict the nature of operator's interests. The result surprisingly will not go as planned.

$$p(s1) = \frac{P(s1_{only}) + P(s1s2s3) + P(s1s2)}{P(s1_{only}) + P(s1s2) + P(s1s2s3) + P(s2_{only}) + P(s1s2) + P(s1s2s3) + P(s3_{only}) + P(s1s2s3)}$$

$$= (20\% + 50\% + 5\%) / (20\% + 5\% + 50\% + 5\% + 5\% + 50\% + 20\% + 50\%)$$

$$= 27\% / 205\% = \underline{36\%}$$

$$p(s3)=(20\%+50\%)/205\% = \underline{34\%}$$

$$p(s2)=(5\%+5\%+50\%)/205\% = \underline{30\%}$$

The number of references to s2 rises toward almost the same with s1 and s3. Most of the references to s2 is needed just to provide a link from s1 to s3. Consequently, s1 , s2, and s3 will fiercely contend to dominate the VT memory (2 screens capacity), causing a lot of network traffic load to incur. The best solution to avoid the above problem is to always provide a fully connected navigation path between screens.



## **CHAPTER IV**

### **CONCLUSION**

The needs for incorporating a virtual terminal device in the J1939 network has steadily increased especially among the agriculture equipment industries. The main driving force for the trend is influenced by many promising features of the virtual terminal:

- Full multiplexing of the messages. a single channel (a J1939 bus) can mediate all message transfers.
- Any network device needing to interact with the operator can easily and flexibly establish the communication. A single virtual terminal can communicates with any network device simultaneously.
- Plug and play (ease of use). All that is needed by the operator to operate the VT is plug it into J1939 network outlet

The proposed standard for the VT messages is simple enough for the network device to communicate with the VT and yet, robust enough to handle all the necessary functions that the VT should have.

Even though the proposed message set standard does not rigidly stipulate how the VT should be implemented, the screen management strategy for the VT is highly recommended. It was shown before that the VT screen management is indeed responsible for the reduction of the network traffic load.

This thesis material can serve as a basic framework for the implementation of virtual terminal application layer in J1939 networks. The further detailed discussion is still open for debate.

## **REFERENCES**

1. SAE (Society of Automotive Engineers) J1939 Committee. Recommended Practice for A Serial Control and Communication Vehicle Network. SAE J1939/72 Virtual Terminal Draft, July 3, 1995.
2. SAE J1939 Committee. Recommended Practice for A Serial Control and Communication Vehicle Network. SAE J1939/01 Base Document Draft, Aug 13, 1993.
3. SAE J1939 Committee. (1994). Recommended Practice for A Serial Control and Communication Vehicle Network. SAE J1939/21 Data Link Layer.
4. G.J. Nutt. Centralized and Distributed Operating Systems. Prentice Hall (1992).
5. A.S. Tannenbaum. Modern Operating Systems. Prentice Hall (1992).
6. W.J. Johnson and J.R. Volk. A Proposal for a Vehicle Network Protocol Standard. Society of Automotive Engineers (1986). SAE Paper# 860392.
7. F.H. Phail and D.J. Arnett. In Vehicle Networking-Serial Communication Requirement and Directions. Society of Automotive Engineers (1986). SAE Paper# 860390.

8. Belady, L.A. A study of replacement algorithms for a virtual-storage computer. IBM Systems Journal. 5, 2 (1966).
9. Carr, R.W. Virtual Memory Management. UMI Research Press (1984).
10. Aho, A.V; Denning P.J.; and Ullman J.D. Principles of Optimal Page Replacement. J. ACM 18,1 (1971).
11. Bensossan, A.; Clingen, C.T.; and Daley, R.C. The Multics Virtual Memory. Comm. ACM 15, 5 (1972).
12. Smith, A.J. A Modified Working Set Paging Algorithm. IEEE Trans. on Computers 25,9 (1976).
13. Masuda, T. Effect of Program Localities on Memory Management Studies. Proc. 6th Symp. Operating System Principles, ACM SIGSOPS, pp. 117-24, 1977.
14. Oden, P.H.; and Shedler, G.S. A Model of Memory Contention in a Paging Machine. Commun. ACM 15, 8 (1978).

## VITA

Lucas Tandang

Candidate for the Degree of

Master of Science

**Thesis:** VIRTUAL TERMINAL

**Major Field:** Computer Science

**Biographical:**

**Personal Data:** Born in Surabaya, Indonesia on January 16, 1969, the son of T.P. Tandang and Sri-indahwati

**Education:** Received high school certificate from Ora et Labora High School, Jakarta, Indonesia; graduated from Trisakti University, Jakarta, Indonesia; received Bachelor of Engineering Degree in Electronics and Control System. Completed the requirement for the Master of Science degree with a major in Computer Science at Oklahoma State University in December 1995.

**Professional Experience:** Software Engineer, Trans-Data Communication, Inc.; Software Analyst, Information System Architecture Workgroup, Danamon Bank; Graduate Research Assistant, Computer Science Department Oklahoma State University, OK; Graduate Research Assistant, Biosystems and Agricultural Engineering, Oklahoma State University, OK.