A STUDY OF TRANSPARENCY AND ADAPTABILITY OF HETEROGENEOUS COMPUTER NETWORKS WITH TCP/IP AND IPV6 PROTOCOLS

Anupam Das, Department of Computer Science & IT, Cotton College, Guwahati, Assam, India

ABSTRACT:

*The present study **"Transparency and Adaptability of Heterogeneous Computer networks"** deals with the study of specially two protocols TCP/IP and IPv6. The two protocols differ in their structures and features. The proposed study entitled **"Transparency and Adaptability of Heterogeneous computer networks"** is basically an analysis of protocols of different environments such as TCP/IP and IPv6. The proposed study deals in mapping from TCP/IP to IPv6 and vice versa. The mapping is done for frame formats and error detection. In this study the mapping of frame format is done by using C++ language. The error detection and correction is done by using Hamming Code technique and C++ language. So, this paper outlines the various features of TCP/IP and IPv6 protocol environment and the mapping from TCP/IP to IPv6 and vice versa considering the two components frame format and error detection.*

KEYWORDS:

Transparency, Heterogeneous-networks, TCP/IP-Protocol, IPv6-Protocol, Hamming-Code, Network-Transparency, Location-Transparency, Computing-Transparency.

1. INTRODUCTION:

Large scale shared internet-works are difficult to design but they provide a unified application and usability for easy operations. These systems are built using multistage processor-memory nodes that are connected through interconnection network, such as torus, hypercube or multistage interconnection network (MIN), in a distributed shared memory organization. The performance evaluation of such multistage interconnection network either in homogeneous or in a heterogeneous condition has been an active area of research in recent years. This has contributed to the advances in the design and implementation of the internet-works to a great extent. However, these advances have made the networks much more complex and now it is difficult to capture all the details of a network into simple analytical and simulated model. The effect of all these advances in the interconnection networks has to be judged from the real changes in the execution time on network flow and applications.

1.1 TRANSPARENCY AND ADAPTABILITY OF COMPUTER NETWORKS

1.1.1 NETWORK TRANSPARENCY

Network Ttransparency means an environment when networks with different protocol, architecture, speed, Band-width, topology etc communicate or transfer data among them without any hazards or information transferring problem. For example, Sun Microsystem's NFS, which has become a de facto industry standard, provides access to shared files through an interface called the Virtual File System (VFS) that runs on top of TCP/IP. Users can manipulate shared files as if they were stored locally on the user's own hard disk.

1.1.2 LOCATION TRANSPARENCY

In computer networks **location transparency** describes names used to identify network resources independent of both the user's location and the resource location. A distributed system will need to employ a networked scheme for naming resources. In other words it is an idea that the resources can be accessed by a user from anywhere on the network without knowing where the resource is located. A file could be on the user's own PC, or thousands of miles away on other servers.

1.1.3 COMPUTING TRANSPARENCY

Any change in a computing system, such as new feature or new component, is **transparent** if the system after change adheres to previous external interface as much as possible while changing its internal behaviour. The purpose is to shield from change all systems (or human users) on the other end of the interface. Confusingly, the term refers to overall *invisibility* of the component; it does not refer to *visibility of component's internals* (as in white box or open system). The term *transparent* is widely used in computing marketing in substitution of the term *invisible*, since the term *invisible* has a bad connotation (usually seen as something that the user can't see and has no control over) while the term *transparent* has a good connotation (usually associated with not hiding anything). The vast majority of the times, the term *transparent* is used in a misleading way to refer to the actual invisibility of a computing process. The term is used particularly often with regard to an abstraction layer that is invisible either from its upper or lower neighbouring layer. Also temporarily used later around 1969 in IBM and Honeywell programming manuals the term referred to a certain programming technique. An application code was transparent when it was clear of the low-level detail (such as device-specific management) and contained only the logic solving a main problem. It was achieved through encapsulation - putting the code into modules that hid internal details, making them invisible for the main application.

1.2 TYPES OF TRANSPARENCY IN DISTRIBUTED SYSTEM

Transparency means that any form of distributed system should hide its distributed nature from its users, appearing and functioning as a normal centralized system. There are many types of transparency:

- **Access transparency** - Regardless of how resource access and representation has to be performed on each individual computing entity, the users of a distributed system should always access resources in a single, uniform way.
- **Location transparency** - Users of a distributed system should not have to be aware of where a resource is physically located.
- **Migration transparency** - Users should not be aware of whether a resource or computing entity possesses the ability to move to a different physical or logical location.
- Relocation transparency - Should a resource move while in use, this should not be noticeable to the end user.
- **Replication transparency** - If a resource is replicated among several locations, it should appear to the user as a single resource.
- **Concurrent transparency** - While multiple users may compete for and share a single resource, this should not be apparent to any of them.
- **Failure transparency** - Always try to hide any failure and recovery of computing entities and resources.
- **Persistence transparency** - Whether a resource lies in volatile or permanent memory should make no difference to the user.
- **Security transparency** - Negotiation of cryptographically secure access of resources must require a minimum of user intervention, or users will circumvent the security in preference of productivity.

The degree to which these properties can or should be achieved may vary widely. Not every system can or should hide everything from its users. For instance, due to the existence of a fixed and finite speed of light there will always be more latency on accessing remote resources. If one expects real-time interaction with the distributed system, this may be very noticeable.

1.3 AN OVERVIEW OF TCP/IP PROTOCOL

TCP is based on concepts first described by Cerf and Kahn. The TCP fits into a layered protocol architecture just above a basic Internet Protocol which provides a way for the TCP to send and receive variable-length segments of information enclosed in internet datagram "envelopes". The internet datagram provides a means for addressing source and destination TCPs in different networks. The internet protocol also deals with any fragmentation or reassembly of the TCP segments required to achieve transport and delivery through multiple networks and interconnecting gateways.

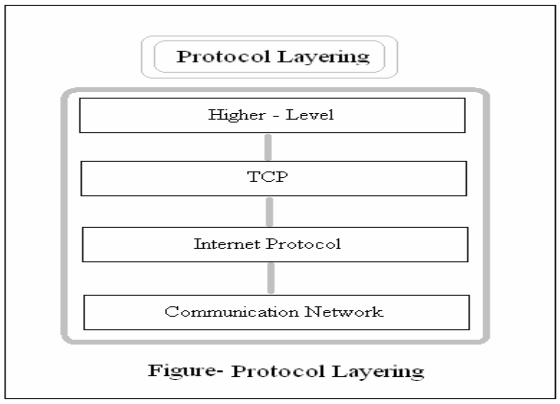


Figure- Protocol Layering

1.3.1 SCOPE

The TCP is intended to provide a reliable process-to-process communication service in a multi-network environment. The TCP is intended to be a host-to-host protocol in common use in multiple networks.

1.3.2 TRANSMISSION CONTROL PROTOCOL PHILOSOPHY

Transmission is made reliable via the use of sequence numbers and acknowledgments. Conceptually, each octet of data is assigned a sequence number. The sequence number of the first octet of data in a segment is transmitted with that segment and is called the segment sequence number. Segments also carry an acknowledgment number which is the sequence number of the next expected data octet of transmissions in the reverse direction. When the TCP transmits a segment containing data, it puts a copy on a retransmission queue and starts a timer; when the acknowledgment for that data is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer runs out, the segment is retransmitted. An acknowledgment by TCP does not guarantee that the data has been delivered to the end user, but only that the receiving TCP has taken the responsibility to do so. To govern the flow of data between TCPs, a flow control mechanism is employed. The receiving TCP reports a "window" to the sending TCP. This window specifies the number of octets, starting with the acknowledgment number, that the receiving TCP is currently prepared to receive.

1.4 FUNCTIONAL SPECIFICATION

1.4.1 HEADER FORMAT

TCP segments are sent as internet datagrams. The Internet Protocol header carries several information fields, including the source and destination host addresses. A TCP header follows the internet header, supplying information specific to the TCP protocol. This division allows for the existence of host level protocols other than TCP.

Source Port							Destination Port
Sequence Number							
Acknowledgment Number							
Data Offset	Reserved	U R G	A C K	P S H	R S T	S Y N	F I N	Window
Checksum							Urgent Pointer
Options							Padding
Data							

Figure-TCP Header Format

The checksum field is the 16 bit one's complement of the one's complement sum of all 16 bit words in the header and text. If a segment contains an odd number of header and text octets to be check summed, the last octet is padded on the right with zeros to form a 16 bit word for checksum purposes. The pad is not transmitted as part of the segment. While computing the checksum, the checksum field itself is replaced with zeros.

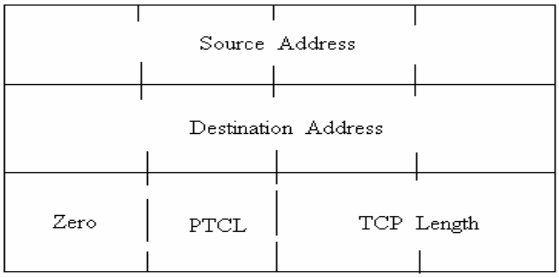


Figure-TCP/IP Frame Format

1.5 INTERFACES

The TCP interfaces on one side to user or application processes and on the other side to a

lower level protocol such as Internet Protocol. The interface between an application process and the TCP is illustrated in reasonable detail. This interface consists of a set of calls much like the calls an operating system provides to an application process for manipulating files. For example, there are calls to open and close connections and to send and receive data on established connections. It is also expected that the TCP can asynchronously communicate with application programs. Although considerable freedom is permitted to TCP implementers to design interfaces which are appropriate to a particular operating system environment, a minimum functionality is required at the TCP/user interface for any valid implementation. The interface between TCP and lower level protocol is essentially unspecified except that it is assumed there is a mechanism whereby the two levels can asynchronously pass information to each other. Typically, one expects the lower level protocol to specify this interface. TCP is designed to work in a very general environment of interconnected networks. The lower level protocol which is assumed throughout this document is the Internet Protocol.

1.6 OPERATION

As noted above, the primary purpose of the TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable internet communication system requires facilities in the following areas:
Basic Data Transfer, Reliability, Flow Control, Multiplexing, Connections, Precedence and Security The basic operation of the TCP in each of these areas is described in the following paragraphs.

1.7 RELATION TO OTHER PROTOCOLS

The following diagram illustrates the place of the TCP in the protocol hierarchy:

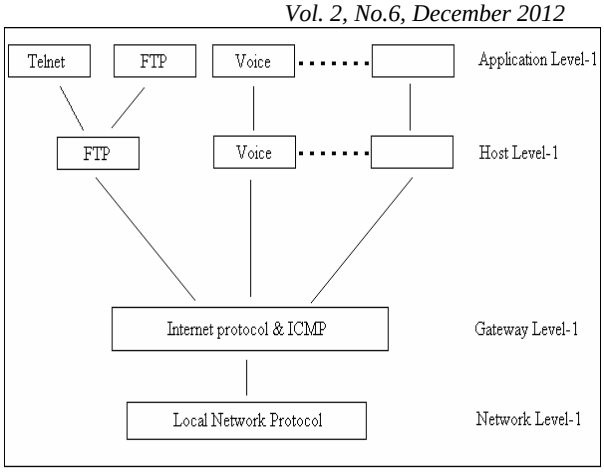


Figure-Protocol Relationship

It is expected that the TCP will be able to support higher level protocols efficiently. It should be easy to interface higher level protocols like the ARPANET Telnet or AUTODIN II THP to the TCP.

1.8 CONNECTION ESTABLISHMENT AND CLEARING

To identify the separate data streams that a TCP may handle, the TCP provides a port identifier. Since port identifiers are selected independently by each TCP they might not be unique. To provide for unique addresses within each TCP, we concatenate an internet address identifying the TCP with a port identifier to create a socket which will be unique throughout all networks connected together. A connection is fully specified by the pair of sockets at the ends. A local socket may participate in many connections to different foreign sockets. A connection can be used to carry data in both directions, that is, it is "full duplex". TCPs are free to associate ports with processes however they choose. However, several basic concepts are necessary in any implementation. There must be well-known sockets which the TCP associates only with the "appropriate" processes by some means.

1.9 DATA COMMUNICATION

The data that flows on a connection may be thought of as a stream of octets. The sending user indicates in each SEND call whether the data in that call (and any preceeding calls) should be immediately pushed through to the

receiving user by the setting of the PUSH flag. A sending TCP is allowed to collect data from the sending user and to send that data in segments at its own convenience, until the push function is signaled, then it must send all unsent data. When a receiving TCP sees the PUSH flag, it must not wait for more data from the sending TCP before passing the data to the receiving process. There is no necessary relationship between push functions and segment boundaries. The data in any particular segment may be the result of a single SEND call, in whole or part, or of multiple SEND calls. The purpose of push function and the PUSH flag is to push data through from the sending user to the receiving user. It does not provide a record service. There is a coupling between the push function and the use of buffers of data that cross the TCP/user interface. Each time a PUSH flag is associated with data placed into the receiving user's buffer, the buffer is returned to the user for processing even if the buffer is not filled. If data arrives that fills the user's buffer before a PUSH is seen, the data is passed to the user in buffer size units. TCP also provides a means to communicate to the receiver of data that at some point further along in the data stream than the receiver is currently reading there is urgent data. TCP does not attempt to define what the user specifically does upon being notified of pending urgent data, but the general notion is that the receiving process will take action to process the urgent data quickly.

1.10 PRECEDENCE AND SECURITY

The TCP makes use of the internet protocol type of service field and security option to provide precedence and security on a per connection basis to TCP users. Not all TCP modules will necessarily function in a multilevel secure environment; some may be limited to unclassified use only, and others may operate at only one security level and compartment. Consequently, some TCP implementations and services to users may be limited to a subset of the multilevel secure case. TCP modules which operate in a multilevel secure environment must properly mark outgoing segments with the security, compartment, and precedence. Such TCP modules must also provide to their users or higher level protocols such as Telnet or THP an interface to allow them to specify the desired security level, compartment, and precedence of connections.

1.11 AN OVERVIEW OF IPV6 PROTOCOL

IPv6 is a new version of the internetworking protocol designed to address the scalability and service shortcomings of the current standard, IPv4. Unfortunately, IPv4 and IPv6 are not directly compatible, so programs and systems designed to one standard can not communicate with those designed to the other. Consequently, it is necessary to develop smooth transition mechanisms that enable applications to continue working while the network is being upgraded. In this paper we present the design and implementation of a transparent transition service that translates packet headers as they cross between IPv4 and IPv6 networks. While several such transition mechanisms have been proposed, ours is the first actual implementation. As a result, we are able to demonstrate and measure a working system, and report on the complexities involved in building and deploying such a system.

1.11.1 NETWORK ADDRESS AND PROTOCOL TRANSLATION

The address and protocol translation presented in this section enables both the communication between nodes in an IPv4 site with nodes in the IPv6 network, and between nodes in an IPv6 site with nodes in an IPv4 nodes. Figures 1 and 2 illustrate these scenarios, and the following paragraphs describe them in more detail.

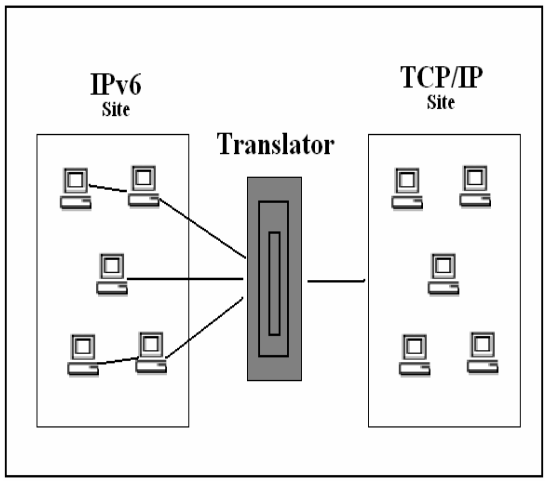


Figure-1Translator for a TCP/IP site

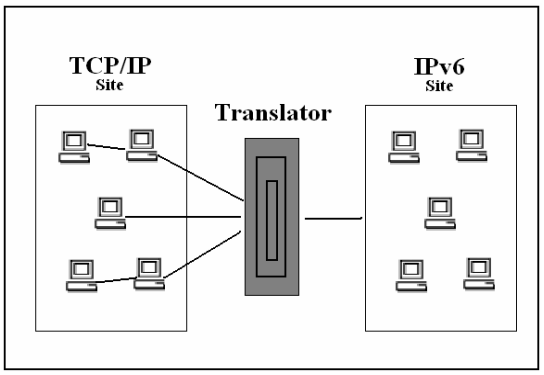


Figure- 2 Translator for an IPv6 site.

Figure 1 illustrates a translator for an IPv6 site communicating with nodes in a TCP/IP network. The internal routing of the IPv6 site must be configured such that packets intended for TCP/IP nodes route to the translator. Hosts in the IPv6 site send packets to nodes in the TCP/IP network using IPv6 addresses that map to individual TCP/IP hosts.

1.11.2 ADDRESS TRANSLATION

Address translation is trivial when using IPv6-mapped and IPv6-compatible TCP/IP addresses. For the TCP/IP-to-IPv6 direction the translator simply extracts the lower 64-bits of a TCP/IP address to obtain an IPv6 address. For the opposite direction the translator sets the lower 64-bits of the TCP/IP source/destination addresses to the IPv6 source/destination addresses, and sets the upper 192-bits of the IPv6 source and destination addresses to the IPv6-mapped and IPv6-compatible prefix, respectively. However, it is considered to be a very bad idea to use IPv6-mapped address as it has the drawback of requiring TCP/IP routers to contain routes to IPv6-mapped addresses. The alternative is to use TCP/IP-only addresses to refer to IPv6 nodes, which requires the translator to maintain an explicit mapping between IPv6 and TCP/IP addresses.

1.11.3 PROTOCOL TRANSLATION

Protocol translation consists of a simple mapping between the two IP protocols, with some special rules for handling fragments and path MTU discovery. The basic operation is to remove the original IP header and replace it with a new header from the other IP version.

1.12. IMPLEMENTING ADAPTABILITY

The implementation of the adaptability of computer networks is confined with the various transformation from one type of protocol to another type of protocol. In the proposed study, I implemented the mapping of data frame formats of TCP/IP to data frame formats IPv6 and vice-versa.

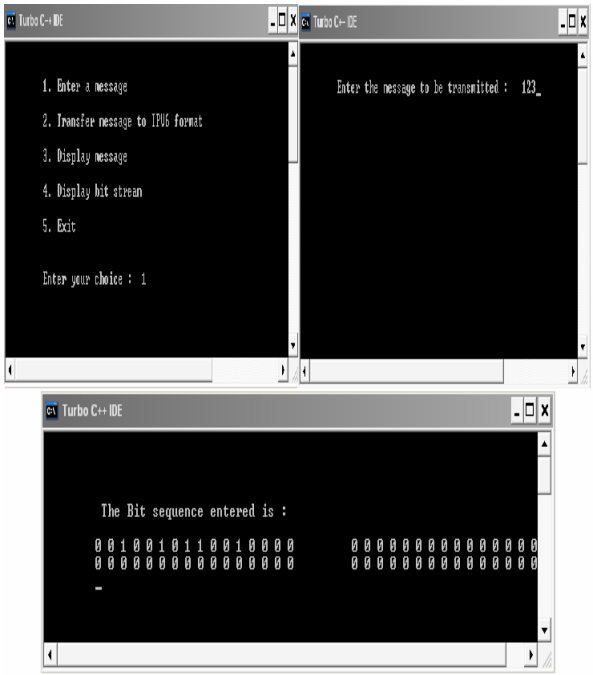


Figure- Mapping of TCP/IP Frame formats to IPv6 Frame formats:

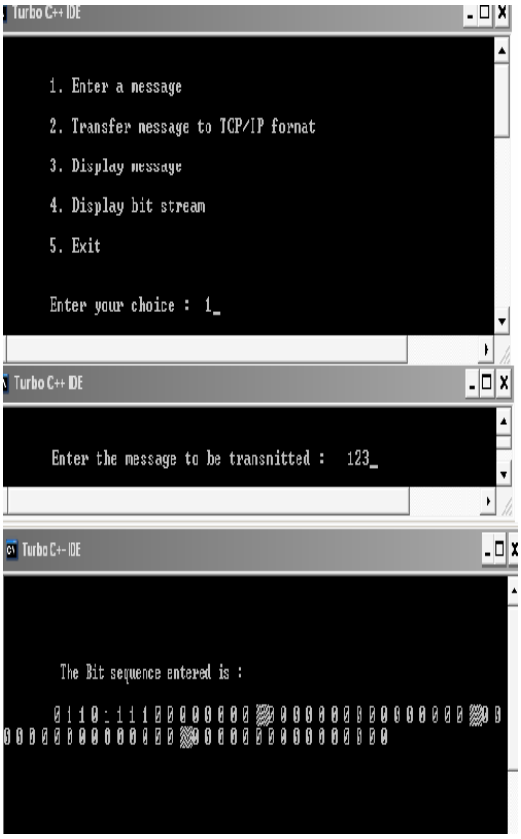


Figure- Mapping of IPv6 Frame formats to TCP/IP Frame formats

1.13 HAMMING CODE:

The two basic strategies for error detection and correction while transmitting information through the electronic media are:

i) To include enough redundant information along with each block of data sent to the destination.

ii) To include enough redundancy to enable the receiver to detect the error (if any).

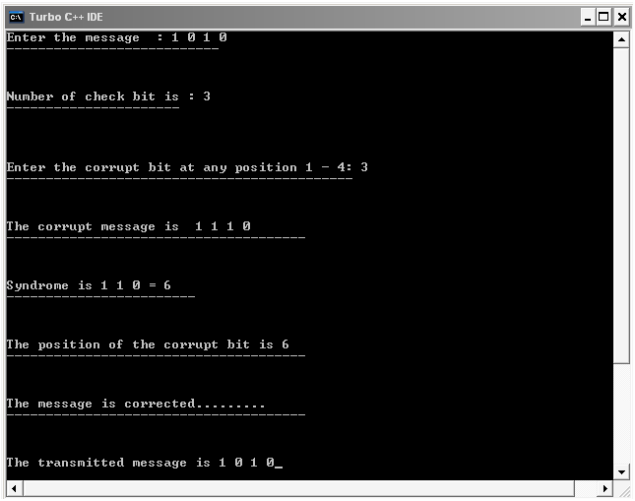
The first one uses the error correcting codes and the second one uses the error detecting codes. Richard Hamming at Bell Laboratory devised a code called Hamming Code capable of correcting one error and detecting two errors. While devising this code, it is assumed that for m-bits of information, r-number of redundant bits are required and thus making the total transmitted bit as n = m + r.

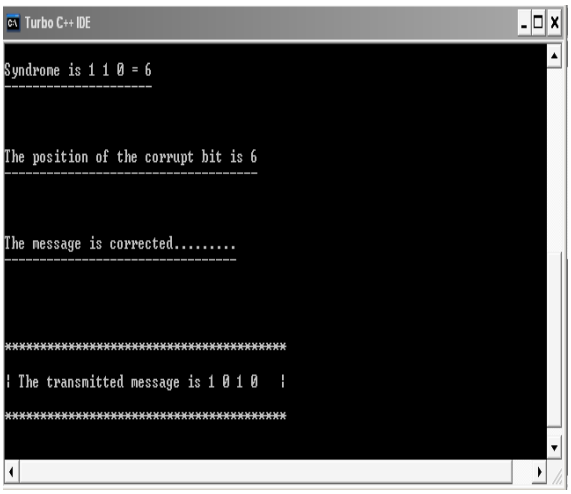
This n-bit unit containing data and redundant bits (check bits) is called n-bit code word. The number of bit position(s) by which two Code word differ is referred to as, Hamming distance(d). The detect 'd' errors; (d+1) Hamming distance code is required. Similarly, to correct 'd' errors, the required Hamming distance is (2d+1).

For a Code word containing m – data bits, r check bits, the number of bits needed to correct a single bit error is $2^r - 1 \geq m + r$. Or $2^r \geq m + r + 1$ That is, $r > \log_2 (m + r)$. Thus, for single error detection and correction on a 4-bit BCD Code for a single decimal digit, 3 (three) check bits are necessary. It is seen from (5.3.2) and (5.3.3) that, for m = 16, $r \geq 5$; and for m = 32, $r \geq 6$. In general, in an n-bit code word, the check bit positions can be found out from the expression.$C_2^k = 2^k$, where, k = 0, 1, 2, 3…..etc.Thus, 1st check bit position: C_1, $[2^k = 2^0 = 1, (k = 0)]$ 2nd check bit position: C_2, $[2^k = 2^1 = 2, (k = 1)]$, 3rd check bit position: C4, [2k = 22 = 4, (k = 2)], 4th check bit position: C8, [23 = 8, (k = 3)] Nth check bit position, C_2^n.

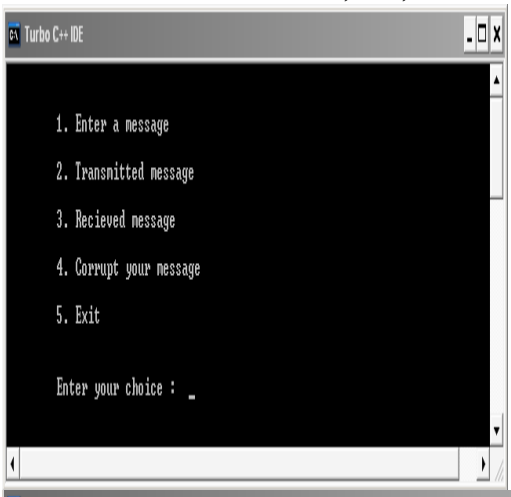
The value of the check bits can be derived from the following check matrix M, $M = [M_{ij}]$ = {1, if I th check bit maintain even Party for a subset of bit positions} S_i, otherwise 0′, }

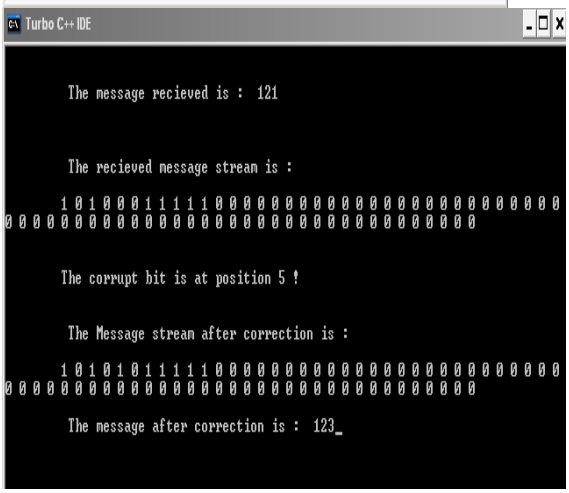
The following is the output of the program implemented for error correction and detection.





The program is further developed for detecting any form of message and it converts the message to 64- bit sequence and then detects and corrects the message as:





1.14 CONCLUSION

The proposed study entitled "Transparency and Adaptability of heterogeneous Networks" is basically an analysis of protocols of different environments such as TCP/IP and IPv6 etc. The proposed study deals in mapping from TCP/IP to IPv6 and vice versa. The mapping is done for frame formats and error detection. The other mappings such as baud rate, width, topology, IEEE standards are not discussed because of time constraint.

Reference:

1. S. Deering and R. Hinden. Internet Protocol, Version 6. RFC 1883, December 1995.
2. P. Srisuresh and K. Egevang. The IP Network Address Translator (NAT). RFC 1631, May 1994.

3. R. Gilligan and E. Nordmark. Transition Mechanisms for IPv6 Hosts and Routers. RFC 1933, April 1996.
4. E. Nordmark. Stateless IP/ICMP Translator (SIIT). Work In Progress.
5. J. Bound. Assignment of IPv4 Global Addresses to IPv6 Hosts (AIIH). Work In Progress.
6. R. E. Gilligan, S. Thomson, J. Bound, and W. R. Stevens. Basic Socket Interface Extensions for IPv6. Work In Progress.
7. G. Tsirtsis and P. Srisuresh. Network Address Translation - Protocol Translation (NAT-PT). IETF Internet Draft, March 1998. Work In Progress.
8. J. Mogul and S. Deering. Path MTU Discovery, RFC 1191, November 1990.
9. J. McCann, S. Deering, and J. Mogul. Path MTU Discovery for IP version 6, RFC 1981, Aug. 1996.
10. J. Postel. Internet Control Message Protocol. RFC 792, Sep. 1981.
11. J. Postel. Internet Protocol. RFC 791, Sept. 1981.
12. B. Fink, 6Bone Overview and Links. http://www.6bone.net
13. B. N. Bershad, S. Savage, P. Pardyak, E.G. Sirer, M. E. Fiuczynski, D. Becker, S. Eggers, and C. Chambers. Extensibility, Safety and Performance in the SPIN Operating System. Proceedings of the Fifteenth ACM Symposium on Operating Systems Principles, Dec. 1995.
14. Y. Rekhter, B. Moskowitz, D. Karrenberg, and G. de Groot. Address Allocation for Private Internets. RFC 1597, March 1994.
15. H. Custer. Inside Windows NT. Microsoft Press. 1993.
16. R. P. Draves, A. Mankin, and B. D. Zill. Implementing IPv6 for Windows NT. Proceedings of the 2nd USENIX NT Symposium, Aug. 1998.
17. RFC-2401] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.
18. [RFC-2402] Kent, S. and R. Atkinson, "IP Authentication Header", RFC 2402, November 1998.
19. [RFC-2406] Kent, S. and R. Atkinson, "IP Encapsulating Security Protocol (ESP)", RFC 2406, November 1998.
20. [ICMPv6] Conta, A. and S. Deering, "ICMP for the Internet Protocol Version 6 (IPv6)", RFC 2463, December 1998.
21. [ADDRARCH] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.
22. [RFC-1981] McCann, J., Mogul, J. and S. Deering, "Path MTU Discovery for IP version 6", RFC 1981, August 1996.
23. [RFC-791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
24. [RFC-1700] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994. See also:
    a. http://www.iana.org/numbers.html
25. [RFC-1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.