OBJECT-ORIENTED DESIGN

OF

TOKEN RING NETWORK

Dissertation submitted to

JAWAHARLAL NEHRU UNIVERSITY

In partial fulfilment of requirements

for the award of the degree of

MASTER OF TECHNOLOGY

In

COMPUTER SCIENCE

BY

RAM BHAGAT

SCHOOL OF COMPUTER & SYSTEMS SCIENCES

JAWAHARLAL NEHRU UNIVERSITY

NEW DELHI – 110067

January – 1999.

CERTIFICATE

This is to certify that the dissertation entitled " **OBJECT-ORIENTED DESIGN OF TOKEN RING NETWORK** " which is being submitted by Mr. RAM BHAGAT to the School of Computer & Systems Sciences, JAWAHARLAL NEHRU UNIVERSITY for the award of Master of Technology in computer science is a bonafide work carried out by him under my supervision .

This work is original and has not been submitted in part or full to any university or institution for the award of any degree .

PROF P. C. SAXENA 5/1/99

Dean

School of Computer & Systems Sciences

PROF P.C. SAXENA 5/1/99

Supervisor

ACKNOWLEDGEMENTS

I wish to convey my heartfelt gratitude and sincere acknowledgements to my guide **Prof. P. C. Saxena** , School of Computer & Systems Sciences for his wholehearted, tireless and relentless efforts in helping me for the successful completion of this project.

I would like to record my sincere thanks to my Dean, **Prof. P. C. Saxena** , School of Computer & System Sciences for providing the necessary facilities in the centre for the successful completion of this project .

I take this opportunity to thank all of my faculty members and friends for their help and suggestions during the course of my project work.

RAM BHAGAT

ABSTRACT

The primary objective of this work is to design the Token Ring Network using object-oriented methodology based on specialization theory. In this work we first described object-oriented concepts which will be applied while designing the Token Ring Network. We further described Token Ring specification and commenced the designing process of token ring .In the next part we have given the full code of token ring in Abstract Syntax Notation.

Table of Contents

CHAPTER 1

INTRODUCTION

Networks are so complex that it is imperative to describe the network in terms of a model . The process of modelling takes the problem into the realm of the abstract where it is easier to find the solution. The solution of abstract problem then can be converted into concrete solutions. A good modelling technique provides a mechanism by which a complex problem can be decomposed into parts and those parts into further subparts. The overall model of the complex system describes not only components which results from such a decomposition but also the relationships among these components. The modelling technique must not only provide structure but also suggest function. A good modelling technique has to make balance between complexity and simplicity depending on the requirements of the application. Object-oriented modelling is a technique that allows classification based not only on structure but also on behavior. One of the reasons for the popularity of the object-oriented techniques is the intellectual appeal of their ability to capture both structure and function in an object. Some modelling techniques like entity-attribute modelling describe the problem domain only in structural terms whereas others like state transition analysis do so only in behavioral terms. The analysis of communication networks needs both aspects . A pure structural model of a communication network describes the nature of the elements that comprise it and their relationships with each other but would not capture behavioral aspects in terms of protocol message and responses. A pure behavioral model would describe their function but would not capture their structural information.

CHAPTER 2

OBJECT-ORIENTED METHODOLOGY

2.1 Object-Oriented themes

In the object-oriented model the themes of object-oriented analysis abstraction, encapsulation, inheritance all find expression. Abstraction is a mechanism for coping with complexity. We can concentrate only on essential details for solving a specific problem . Abstraction draws a boundary around an object inside which are the essential characteristics of the object from the point of view of the application domain . Essential characteristics isolated inside the abstraction barrier must capture the notion of both structure and function based distinctions. A correctly defined abstraction allows the same object model to be reused in various ways.

Encapsulation consists of identifying the internal implementation of the network element and separating those from its externally visible behaviour. Encapsulation preserves the integrity of the object - the underlying implementation may be changed as long as the interface visible to other objects remain consistent. Encapsulation and abstraction are complementary to each other. Just as abstraction separates the essential characteristics of an object from the non-essential ones, the encapsulation separates the externally visible characteristics from the hidden ones. The process of abstraction and encapsulation enables us to talk about network operations completely in the abstract.

An object class is a set of objects having common structural and behavioural properties. All objects in a class have the same purpose. The process of classification in combination with abstraction allows us to categorise the different elements that comprise a communication network as lines ,circuits, LAN bridges , softwares , services switching equipments etc. In addition to this classes also serve as templates with which to create new objects. The attributes of an object describe those data values that the object possesses which could conceivably be different from the data values possessed by other objects of the same class. When we assign an attribute to a class we say that the attribute is-a-property of that class . Conversely the class has-as-a-property each attribute assigned to it. The terms is-a-property-of and has-as-a-property are called property assignment associations. Each attribute has a data type which defines the nature and range of values the attribute can possess. The set of all values which an attribute may possibly possess is known the domain of the attribute. An attribute can be specified unique i.e. each object instance will have different values for that attribute.

The function describes the behavioral properties of an object. Functions are of two types procedural function and stream function. Procedural functions accept arguments and produce results which are well defined data types. Stream functions have at least one argument or one result that is not a well defined data type but is an unstructured continuous analog or digital stream.

Inheritance provides a way of classifying classes. If we find two similar object classes sharing a subset of their properties , we abstract their common properties into a superclass. We thus have a categorization of classes . The inheritance hierarchy provides a second order abstraction in which common properties and behavior can be isolated in yet other classes . This second order abstraction is the defining characteristic of object-oriented modelling paradigm. Inheritance is also a mechanism for implicit property assignment i .e. by assigning a property to an ancestral class , it becomes available in all its descendants. The properties available in an object class through inheritance from ancestral class are called inherited properties of that class. In addition to inheriting characteristics from its superclass, the subclass may choose to define additional characteristics of its own . This is called as extension . The properties defined by subclass on its own are called original properties. The class where a property is first defined is called originating class. Inheritance and extension are mechanisms to express the semantics of classification and evolution . They are a means of incorporating a formalized taxonomy within the object model itself

A paradigm that supports abstraction and encapsulation is said to be object based ,one that ,in addition , supports classification is said to be class based and further if it also supports inheritance it is said to be object-oriented. An object instance is a concrete object which is an example of an object class. It has an identity which is different from other instances of the same class .The process of creating object of a class is called instantiation. A class which has direct instances is called a concrete class . Those classes which are not instantiated are called abstract classes. Classes, which not only themselves but whose descendants, are also abstract are called fully abstract classes.

Aggregation refers to the ability to construct complex objects by assembling simpler objects together in a meaningful configuration. Decomposition is a process of breaking down the complex system in a set of subproblems each of which we know how to solve independently. While decomposing we must take care that each each part should preserve encapsulation and present a well defined set of interfaces at its boundary . An aggregation hierarchy identifies the part/whole relationships between objects.

The modelling methodology applied here is based on specialisation theory. It is used to organize unstructured knowledge to meaningfully represent a system . Here emphasis lies in understanding the networks in terms of its component objects, their attributes and the relationship they hold with each other. Object-oriented model serves as a basis for specification, design and documentation.

2.2 Object-oriented modelling v/s object-oriented programming

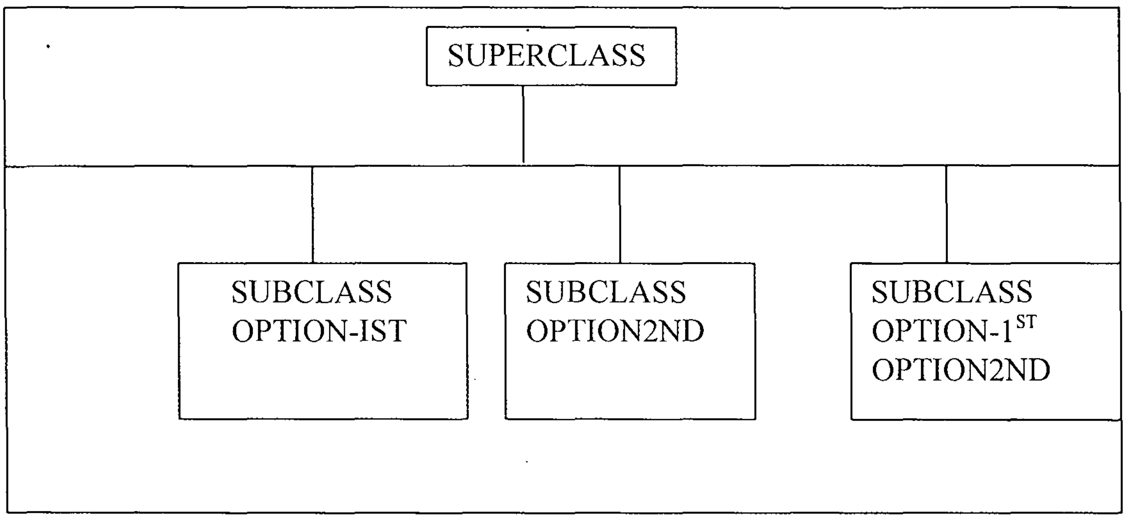
Object-oriented modelling is different from object-oriented programming in various aspects e.g. in object-oriented modelling inheritance does not necessarily mean reusability of implementations, it simply means reusability of specifications. Here inheritance is a mechanism of incremental improvement of the classes already defined. In specialisation theory which is stronger form of object-oriented modelling monotonic inheritance is applied i.e. already existing features of classes are not allowed to be cancelled. In case of object-oriented programming we can drop the properties(attributes or functions) of the superclass. Further multiple inheritance is rarely used in object-oriented modelling . In fact it can be done away by using aggregation and hierarchy redesign . While in object-oriented programming it is encountered more frequently. In case of modelling overriding (a process by which subclass can redefine inherited attributes and functions) is not permitted but in object-oriented programming this feature is available and is often implemented by virtual functions. In object-oriented modelling polymorphism does not make much sense as here we are not concerned with implementation variations but in programming it plays an important role. Similarly the concept of dynamic binding is not meaningful in object-oriented modelling of networks. Again we do not use delegation which is used in certain object-based software environments. Delegation as the name suggests is a property by which an object can delegate its responsibility of performing certain operations to some other object which in turn can delegate this responsibility to some object.

Object-oriented programming provides a means of visibility control by the concept of class and superclass but there is no such need of visibility control in object-oriented modelling. In object-oriented programming declarative instantiation i.e. having defined a class in a formal syntax one can then instantiate variables of the class through declarations in the same syntax. However declarative instantiation is not meaningful in object-oriented modelling of communication networks . The issue of inter-object communication is very straightforward in object-oriented programming while it is much complicated in object-oriented modelling of communication networks. Referent and non-referent both types of classes are referred in object-oriented programming but not in object-oriented network model. A subclass can be arbitrarily written while overriding the original behavior of the class but in a subtype its original has to be maintained. It can be modified only in permissible ways.In object-oriented network modelling as we are applying specialisation theory the difference between subclasses and subtypes is irrelevant. This difference is significant in many object-oriented languages. Anyhow object-oriented modelling and object-oriented programming are complementary to each other. They both are of benefit in different phases of system development . The programmer need no longer expend effort for designing software classes for network objects as the hierarchy obtained by object-oriented modelling can be converted into programming language.

2.3 Object Class

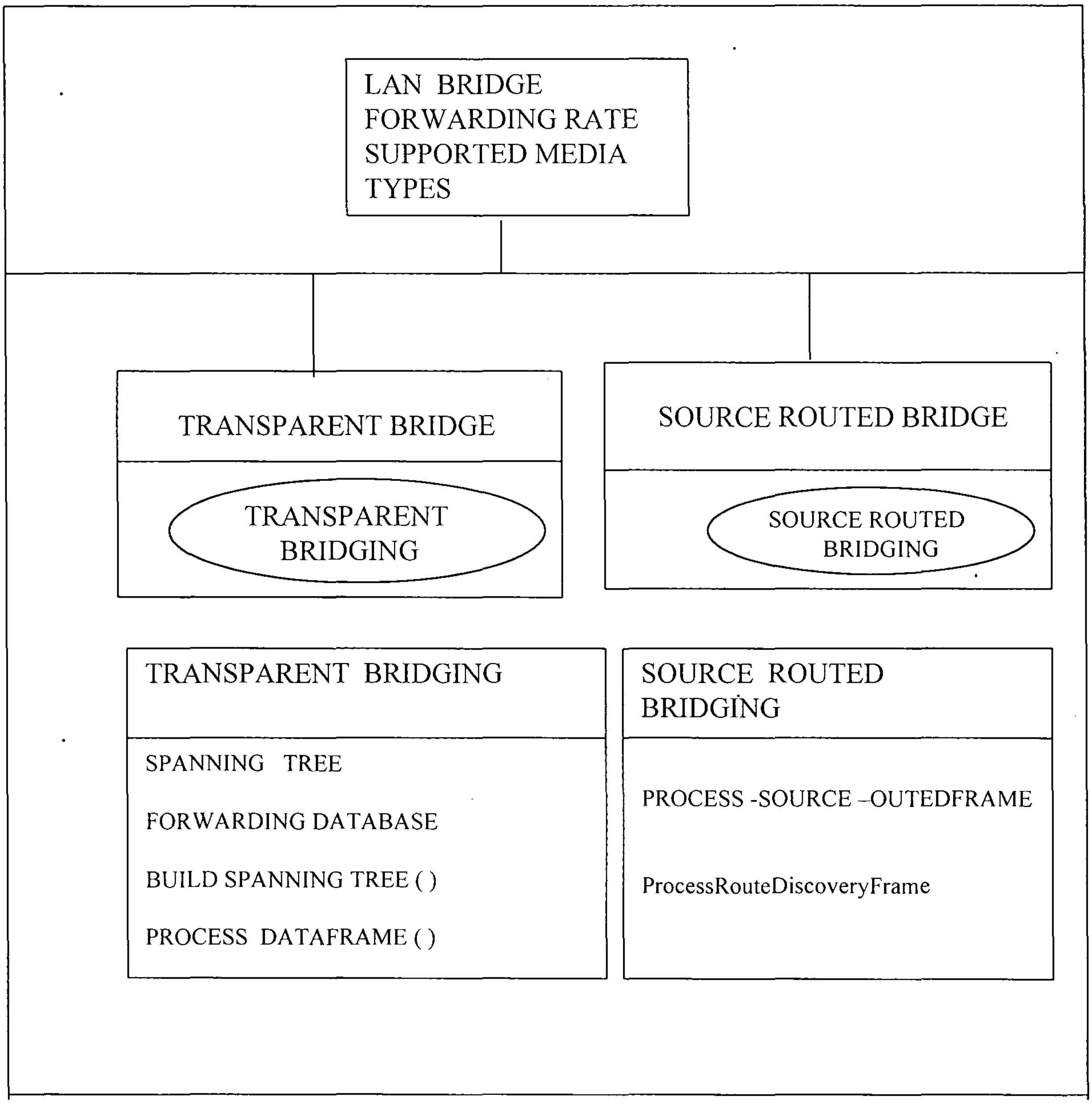
A meaningful object class in a network model must have a referent in the actual system i.e. it models some entity in the actual system which is relevant to our purposes. Object classes may be physical entities or logical entities e.g. protocol entities, network services and sites . Every object class is modelled by identifying attributes and functions of interest. The class is described as a modelling construct and its attributes and functions are described as independent modelling constructs which are associated with the class through property assignment.

There are two types of properties those which are mandatory and those which are optional. Mandatory properties are captured by using core specification while optional properties are captured using variant specification. Let us first describe the core properties .Here we identify the properties which must be present in every instance of the class i.e. if an object lacks any of the core properties then it is not an instance of that class but while identifying the attributes and functions we must be careful that they are orthogonal i.e. they are not functionally dependent on each other in any way. Thus we should not be in a position to compute any attribute or function in terms of other attributes e.g. suppose we are trying to find out three parts of a number so that their product is maximum. In this case if we know first two parts of the number then the third part can be computed just by subtracting the sum of the two parts from the number. Thus if we represent this third part as also an attribute then these attributes will not be orthogonal as a relationship exists between them. This may cause sometimes integrity violation . An object not having any knowledge of the dependency between the attributes may try to set them to values which are inconsistent with each other. Such an attribute can be specified by formally modelling their functional dependency on other attributes as an algorithm. While specifying functions care must be taken to specify function arguments such a way that the number of functions required can be minimized e.g. if there are two functions fiveminuteservice and tenminuteservice then they should not be modelled as two separate functions instead they should be modelled as a single function and an argument duration should be provided to that function .But this can be done only if the two functions have the same purpose. If their purpose is different then they must be modelled as separate functions. In communication networks there are variety of devices, protocols and services. There are different types of versions and options .. If the core property in a superclass has been defined then it can be extended into different subclasses using inheritance e.g. if there are two options optionfirst and optionsecond then the subclasses can be obtained as shown in the figure .



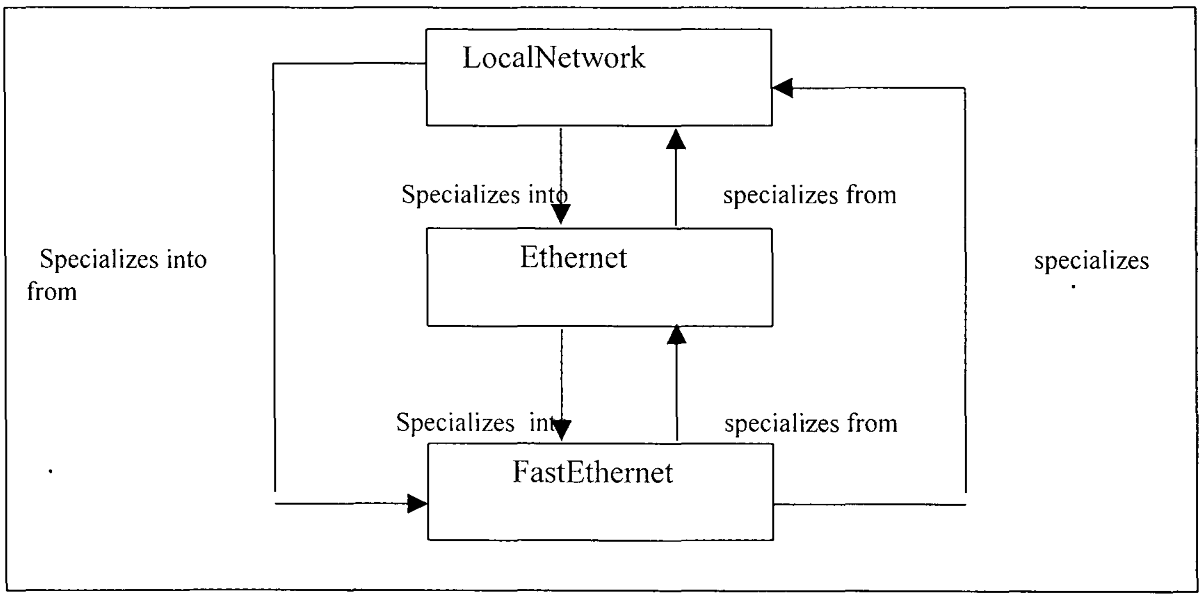
2.4 Capsules

In this way if we have n options then we will get 2^n -1 subclasses which gets out of hand for values like 20 or more. Therefore it complicates the whole design of the class hierarchy. Thus it is better to provide variant property specification. There certain properties which may be present or absent together. Such a collection of properties is modelled by the concept of capsule which may be named. A capsule is never instantiated independently . It is a mechanism to add flexible subsets to object classes . A capsule can be reproduced just by a reference to an already existing capsule . The properties of one capsule should not overlap those of others or with the core properties defined for a given class. They also reduce the need of modelling multiple inheritance. A capsule provides a powerful abstraction to describe variant or optional behavior.A capsule may be mandatory or optional. In a capsule if we want to make an attribute to be optional then the entire capsule has to be made optional. It is in consistence with practice as almost all optional characteristics are collections of multiple related attributes and functions which occur as an assemblage. The presence or absence of an optional capsules is also specified using a flag attribute which specifies which optional capsules are present in that instance. In the process of specialisation an optional capsule can be fixed i.e. the capsule becomes mandatory in that class and all the descendant classes. This is termed as capsule fixing . Capsules can be nested with other capsules but not with themselves. Capsules are used as modelling abstractions to specify and import reusable assemblages of properties in network objects as shown in the figure. TransportBridging and sourceRoutedBridging are two capsules. Capsules also help in avoiding multiple inheritance. This accelerates processing in the model information base since multiple inheritance is costly to compute. Further it maintains exactness in the taxonomical structure for assigning instances to the class.

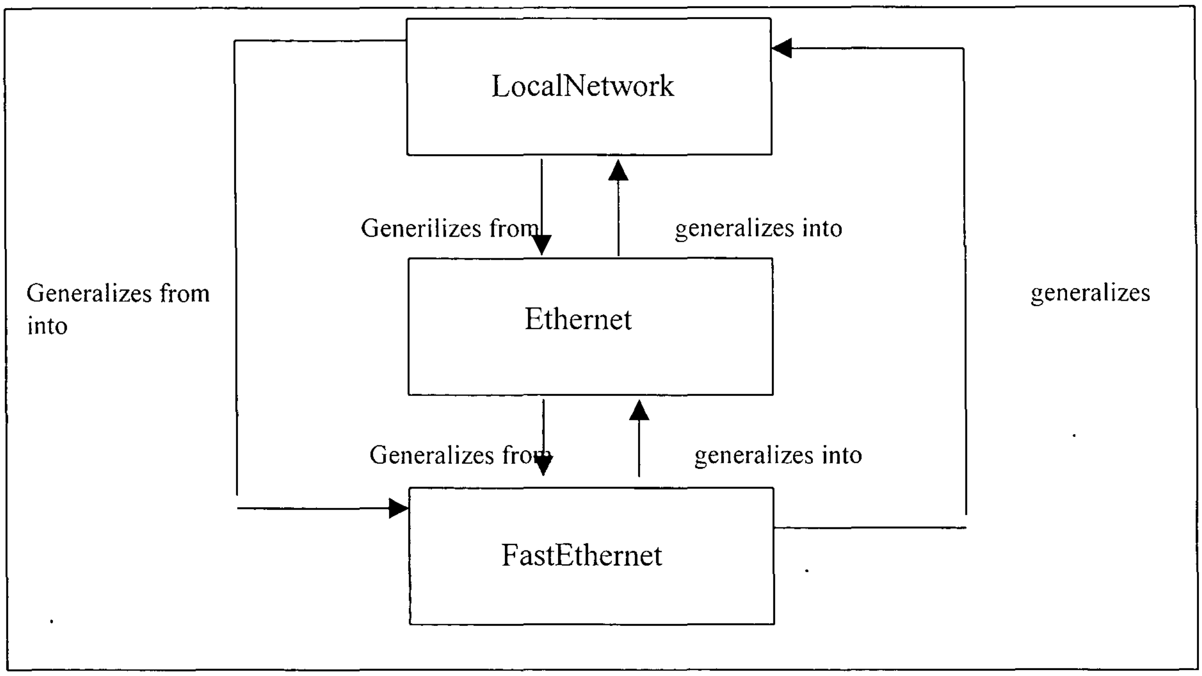


2.5 Generalisation & Specialisation

Generalisation and specialisation are the basic themes of inheritance hierarchy . As we go up in the inheritance hierarchy generalisation increases and as we go down the hierarchy specialisation increases . Generalisation is the process constructing an inheritance hierarchy in a bottom-up fashion . It is an exercise of identifying similarities between object classes, abstracting them and raising them to the level of a superclass . The classes at the top of the hierarchy are simpler and the classes at the bottom of the hierarchy are more complicated in terms of the structure and advanced in terms of the functioning. The construction of the hierarchy in top-down fashion by providing more attributes and functions to the superclass is specialisation. Its purpose is to focus on more specific aspects of the problem domain. While creating the hierarchy we first apply the principle of generalisation and once a hierarchy is constructed specialisation is useful for assigning new object classes a place in the existing hierarchy. As we go up higher in the inheritance tree there comes a point from where no further abstraction is possible . This is known as the root of the inheritance hierarchy . We call it by the special name generic object. It captures the only common properties of every possible object module in the application domain . There is a specialises-into association between a superclass and its subclass. It is transitive from a class to all its descendant classes e.g. if the ethernet class derives from localnetwork class and fastethernet derives from ethernet class i.e. localnetwork class specialises into ethernet class which in turn specialises fastethernet class . Then localnetwork class also specialises into fastethernet class . The complement association is specialises from . That is we can say that ethernet class specialises from localnetwork class . These are shown in the figure as below .

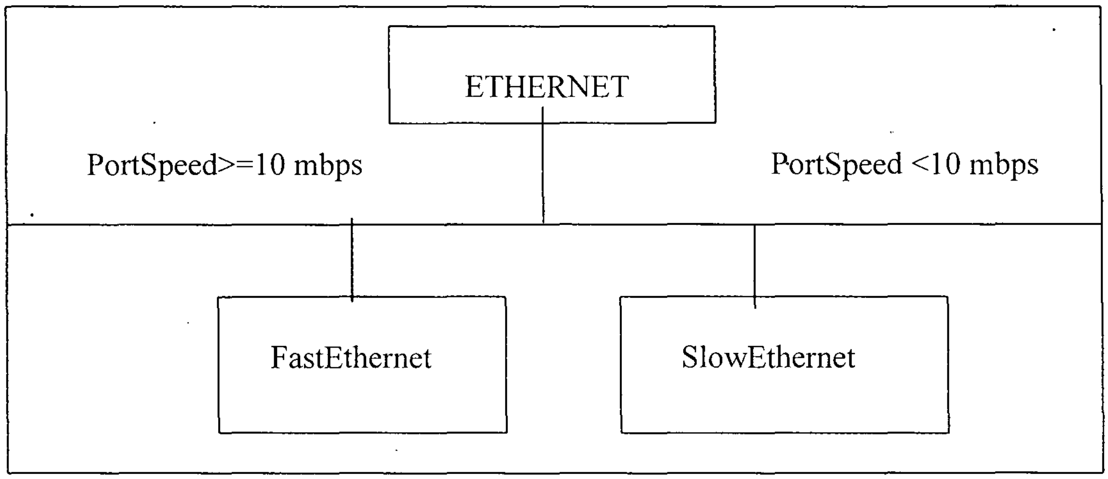


These associations are transitive but not symmetric e.g. ethernet class specialises from localnetwork class but localnetwork class does not specialises from ethernet class. All object classes can be enumerated from the root class by traversing the transitive closure of its specialises into association . It means determining all nodes on the graph which are reachable using the starting node. There are corresponding associations of generalisation as well . There are two associations generalises-from and generalises-into . A superclass generalises from a subclass and a subclass generalises into a superclass . These associations are also transitive but not symmetric. They can be shown as below.

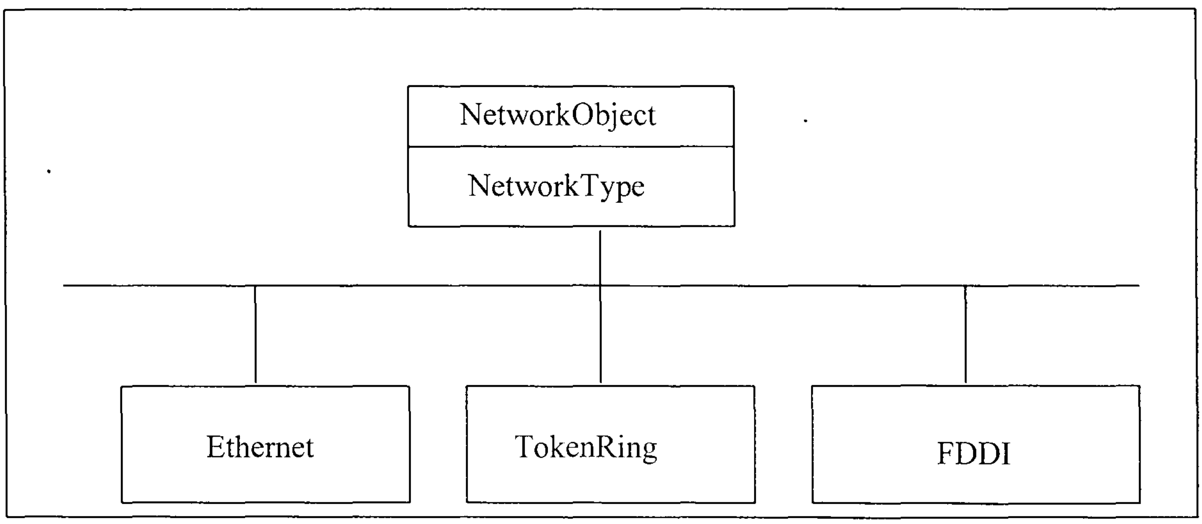


It is a good modelling practice to take care that concrete classes are leaf level classes in the hierarchy as much as possible. It is possible that some non-leaf classes also have instances . It is possible when a new subclass is added making the already existing class

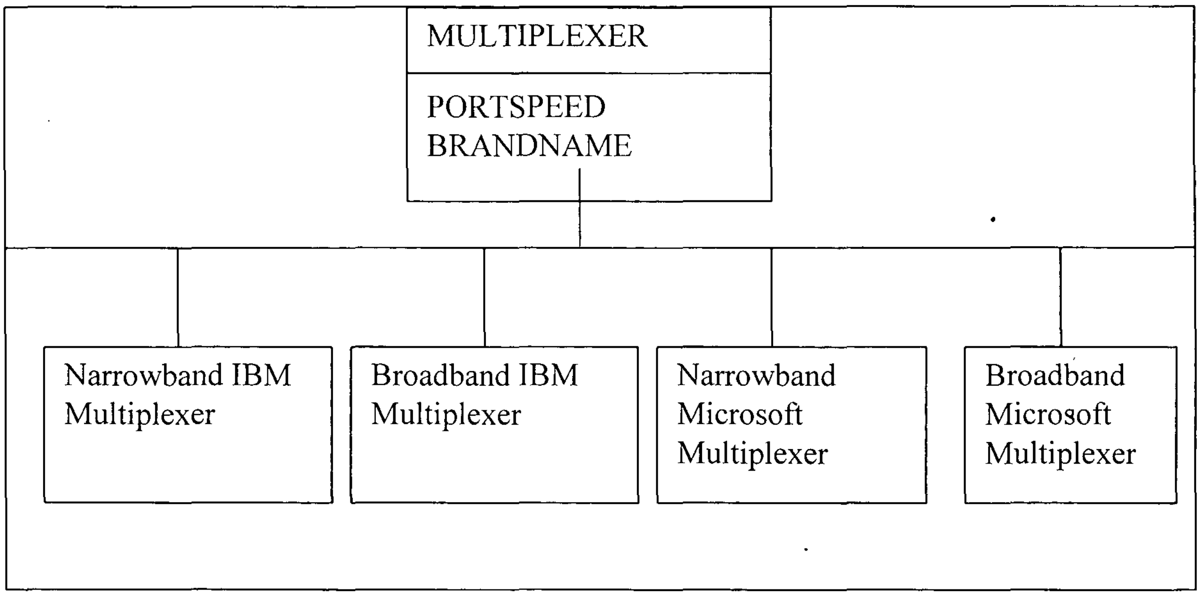
a non-leaf class. In specialisation theory generalisation and specialisation are formally described in terms of a basis property i.e. when we have to specialise a new object class from an existing class we must choose a basis of specialisation. It acts as a distinguishing factor between different subclasses. Generally this basis property is an attribute of the superclass . However it can be some function based property also . As a subclass is more specialised form it has more capabilities than its superclass. Actually it operates in a narrower range of the overall problem space . The basis property which has more domain in the superclass is restricted in domain in the subclass . Then there are two kinds of basis of specialisation Quantitative and Qualitative .If the basis property is a numerical attribute of the superclass specialisation is done by restricting the domain of the values that the attribute is permitted to take in the subclass . This is known as quantitative basis of specialisation . If the basis of property is an enumerable attribute of the superclass, specialisation is done by restricting the set of enumerated values this attribute can have in the subclass. This is called qualitative basis of specialisation



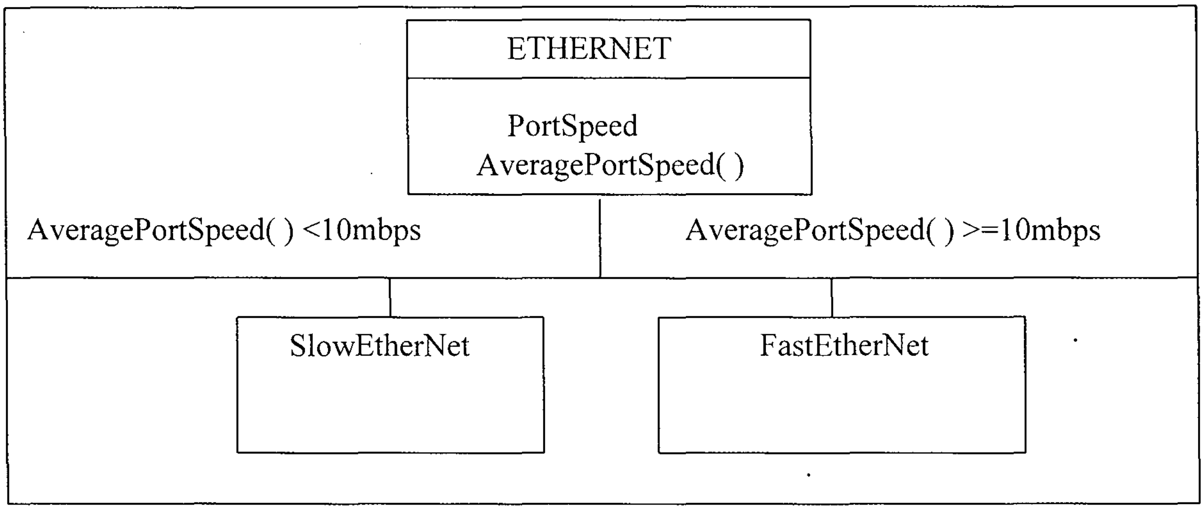
Consider a superclass ethernet which is specialised into two subclasses fastethernet and slowethernet depending on the portspeed. Thus it is an example of quantitative basis of specialisation . Here portspeed attribute has been restricted in domain for subclass.



As shown in the figure there are three types of network object ethernet, tokenRing and fddi depending on the value of the attribute network type. This networktype is here basis of specialisation . When specialisation is done by restricting the domain of an attribute of the superclass the mode of specialisation is known as specialisation by Attribute domain restriction There are two notions associated with specialisation disjointness and completeness. A disjoint specialisation means that all the subclasses of a superclass must form non-intersecting sets of objects i.e. an object which is a member of a superclass can be a member a member of only one subclass. An object can not be a member of more than one class i.e. two classes can not have any object in common e.g. consider the example in which we considered ethernet superclass and specialised it on the basis of portspeed . If we take portspeed >=10 mbps for fastethernet and portspeed<= 10 mbps for slowethernet then we have overlapping i.e. these two subclasses are not disjoint . For portspeed = 10 mbps objects belong to both subclasses .A complete specialisation implies that every object of the superclass has to be a member of some subclass . Considering the same example an object of ethernet class has to be a member of fastethernet or a member of slowethernet subclass as by the definition its portspeed has to be either >=10 mbps or <10 mbps . The notions of disjointness and completeness are also useful when basis of specialisation is qualitative . In network modelling it is better to have subclassing disjoint and complete as far as possible . It improves the richness of the taxonomical structure by removing any ambiguity and confusion with regard to where instances can be assigned in the hierarchy . Furthermore completeness assures us that each instance can be assigned to at least one class. Sometimes we need to choose more than one basis property for the same specialisation . Then it is called compound specialisation . For example if two different basis attributes undergo domain restriction simultaneously as part of the same specialisation as shown in the figure .



Compound specialisation can be replaced by simple specialisation at more than one levels. There is no harm in using compound specialisation if it makes sense for the application domain and number of subclasses at each level are manageable. Another mode of specialisation is known as specialisation by argument contravariance. Instead of restricting the domain here we relax the domain of the arguments of the function. Thus the function possesses now more versatility. They can handle more situations. This is known as contravariance. It is named so because as subclasses become specialised and restricted in their basis properties the arguments that their function can handle actually become weaker and relaxed. As all the subclasses are actually subtypes here therefore contravariance works. We may specialise by tightening function results. In fact finding an attribute's value may be internally implemented as the result returned by a function computation. So as we can restrict domain of an attribute to specialise. Similarly we can do the same thing with result of a function. Consider the figure



Thus if the return value of the function averageportspeed ( ) is less than 10 mbps then it is specialised into slowethernet otherwise fastethernet .Here the basis of specialisation is the result of the function averageportspeed(). Function results can be restricted for specialisation like attribute domain . Thus function results are said to be covariant with specialisation . We can also do specialisation by the capsule fixing i.e. if we create a subclass from an existing object class in which a capsule is optional we make that capsule mandatory . All instances of the subclass and the classes further derived will now possess this capsule . Now we describe certain specialisation principles. A specialisation principle is a rule which states how a subclass can be legally derived from a superclass .

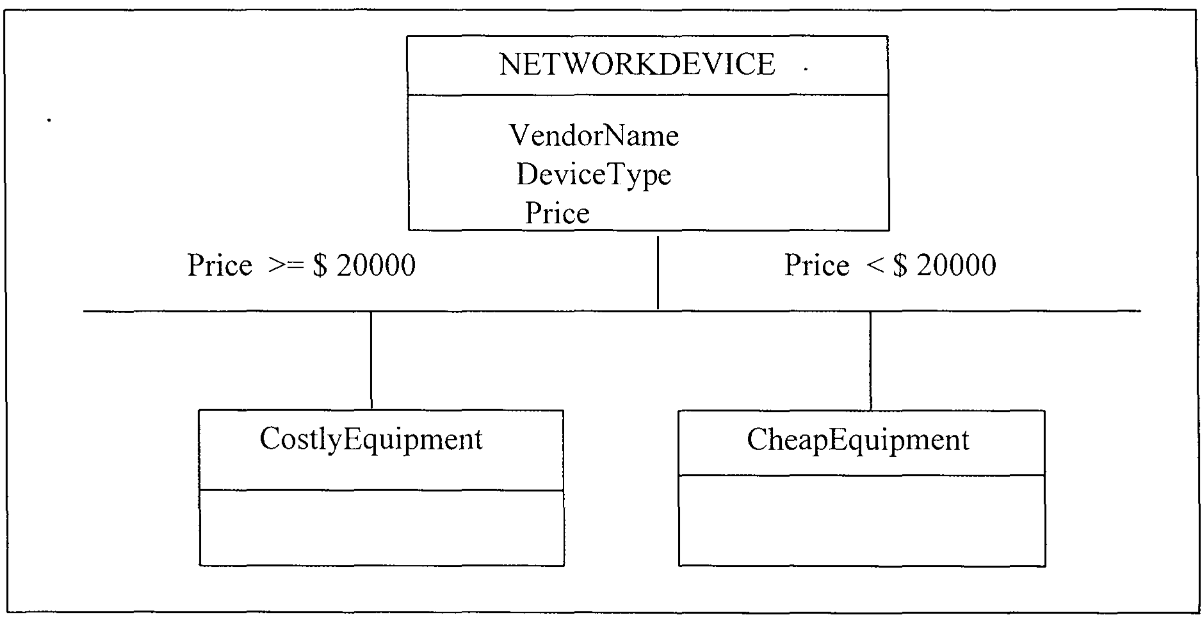
2.6 Specialisation Principles

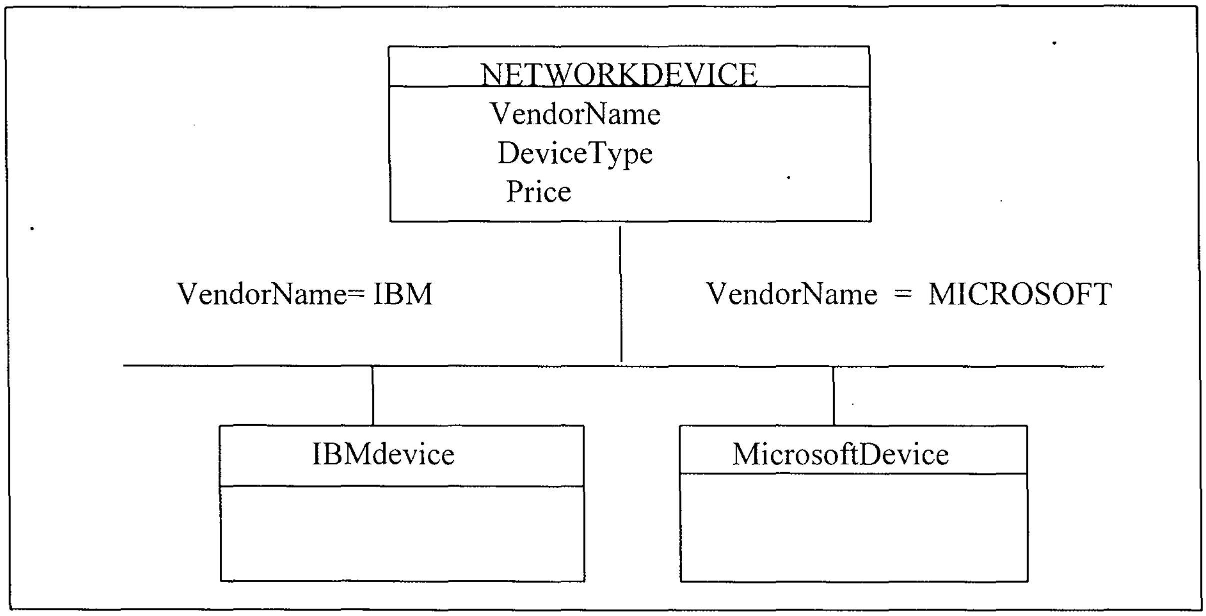
Specialisation principle of attribute addition:- A descendant class may add an attribute as new property .

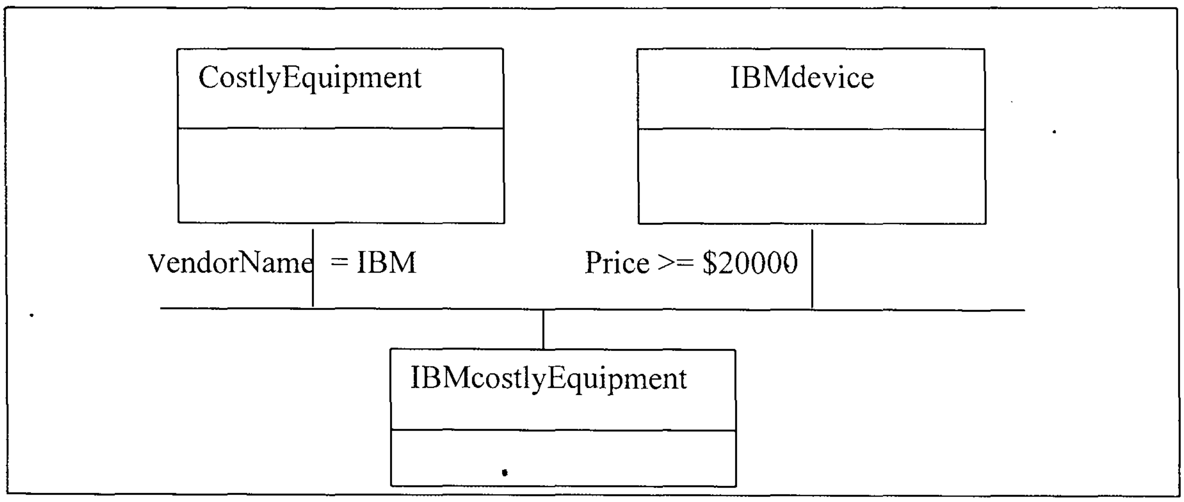
Specialisation principle of Function Addition:- A subclass may add a function as a new property .

Specialisation principle of capsule addition :- A subclass may add a capsule as a new property
Specialisation principle of attribute domain restriction :- A subclass may restrict the domain of an attribute inherited from an ancestral class.
Specialisation principle of argument contravariance :- A subclass may expand the domain of an argument of a function inherited from a superclass .
Specialisation principle of result covariance :-A subclass may restrict the domain of the result of a function inherited from a superclass .
Specialisation principle of capsule fixing :- A subclass may mandate the use of an optional capsule inherited from a superclass .

After describing specialisation principles lets refer to multiple inheritance . It is the capability of an object class to inherit properties from more than one superclass . It is generally suitable when an ancestral class has been specialised at different times using different bases of specialisation . In multiple inheritance specialisation proceeding from the same superclass at two different times we have two different bases of specialisation . Consider the example as shown in the figure







In this multiple derivation that basis of specialisaion which is not used earlier is applied from each superclass. From the costlyequipment class the vendorName basis is being used while from the IBMdevice class the price basis is being used . In specialisation theory we permit the use of multiple inheritance when required . However as far as possible when subclassing only one basis property should be selected .

2.7· Normalization

Normalization is the technique used to avoid selective inheritance . It helps us find out the most appropriate class in which a particular property must be placed . The normalization of a class hierarchy may occur either at architecture time or at operations time . However at operations time unrestricted redefinition of classes may cause instances to lose their instantiation associations with existing classes . So there are some restrictions with respect to normalizing a class hierarchy e.g. we can not change name of any concrete class . The extent of a concrete class can not change . The name of any ancestral class of a concrete class can not change . The membership of any ancestral class of a concrete class can not change and the set of properties accumulating in every concrete class through inheritance can not decrease . An advantage of normalization is that concrete classes tend to move to the bottom of the hierarchy and intermediate classes tend to stay abstract classes . It helps us in designing better hierarchies . The following are certain guidelines to design well structured object class hierarchies .

1.Use capsules to model minor variations of a class and specialisation to model major variations .

2. Avoid free specialisation .A formal basis of specialisation should be chosen at each level of subclassing .

3. Select only one basis of specialisation as far as possible

4. Ensure that each specialisation is disjoint.

5. Ensure that each specialisation is complete

6. push all concrete classes as far down in the hierarchy as possible.

7.Ensure that the hierarchy is normalized . By this it can be ensured that there is no need to apply selective inheritance.

8. Avoid multiple inheritance as far as possible . Try to find other ways to deal with the situation.

2.8 Aggregation

Aggregation is an important mechanism in object-oriented designing . It is a way of breaking down information in order to create complex objects by making an assembly of smaller objects . In object-oriented modelling this breakdown process is called decomposition . The reverse process i.e. assembling objects to create complex objects is called composition. The object formed on such a way is called aggregate object . The simpler objects which together form the aggregate are known as component objects. The aggregate is taken as a single object. Aggregation provides a powerful abstraction that facilitates the description and manipulation of all component objects . Aggregation specifies part-whole relationship between between object classes The components are considered to be part and aggregate is whole . Aggregation relationships can be represented by aggregation hierarchies . An aggregation hierarchy is rooted at the aggregate object and proceeds downward to show its first order components . There may need of more than one aggregation hierarchy to describe the complete model of a single network.

Sometimes confusion occurs between inheritance hierarchy and aggregate hierarchy. The inheritance hierarchy describes the specializes from association between classes . Its goal is to describe taxonomies so that we can categorize objects . The categorisation of an object does not affect the operation of the object itself. But aggregation hierarchy defines the compositions so that we can assemble the objects to propagate the effects of operations on one object to another. It actually affects the operations of the objects. The same object class may appear twice in the aggregation hierarchy but every object class can occur exactly once in the inheritance hierarchy . These two hierarchies are complementary to each other . None of them is sufficient . They are both required together . An aggregation hierarchy is like an assembly blueprint which tells us how to put them together .The inheritance hierarchy helps us in finding the parts we need .The definition of aggregation is not cncerned with individual properties of the components.

There are two types of aggregation exclusion aggregation and inclusive aggregation . In exclusion aggregation each specific component instance can belong to only one aggregation instance while in inclusive aggregation some component may belong to more than one aggregate instance e.g. when we say that a hubcard component is a part of a wiringhub aggregate we imply that no other wiringhub object may possess the same hubcard component . This is an example of an exclusive aggregation . On the other hand suppose that a digitalvideoclip component object is a part of multimediadocument aggregate object . This does not prevent the same digitalvideoclip object from also being a component of a different multimediadocument object . This is an example of inclusive aggregation . There exists some confusion aggregation and capsules . It should be kept in mind that capsules are never instantiated whereas component objects are . Further a class may contain more than one instance of the same component whereas it can contain each capsule only once.

In order to describe how many times a component is required to form an aggregate , the concept of component multiplicity is introduced . It allows us to specify the number of components which may be contained in the aggregate . Component multiplicity is shown in the form of an interval [N,M] where N and M are non-negative integers . This interval infers that there are at least N and at most M instances of the component to form the aggregate . Component multiplicity may also be specified by a discrete set of enumerated integers . It is also possible to define component multiplicity in terms of attribute-defined expressions . In fact attribute-defined expression is actually the most general and most satisfactory way of specifying the component multiplicity . If the aggregate class has been specified correctly , it is usually possible to select one its capacity related attributes to specify component multiplicity . In every multiplicity specification where the network model seems to require a non-trivial integer constant for a limit it is better to look closely for any capacity related attribute of the aggregate class and examine it for possible use in an attribute defined expression for the component multiplicity.

An aggregate class and its component classes both have a place in the inheritance hierarchy . Therefore each can specialise into a number of descendants classes . Further the descendants of the component object classes have inherited its composition relationship with the aggregate object class and the descendants of the aggregate object class have inherited its decomposition relationship with the component object class . There are certain aggregation principles .

Aggregation principle of composition inheritance :- A descendant of a component class inherits the composition relationship of an ancestral with.the aggregate class .

Aggregation principle of decomposition inheritance :- A descendant of an aggregate class inherits the decomposition relationship of an ancestral class with the component class .

Aggregation principle of monotonic aggregation inheritance :- A descendant class may not cancel any composition or decomposition relationship defined for an ancestral class .

Aggregation principle of component multiplicity inheritance :- A descendant of an aggregate class inherits by default the component multiplicity of each decomposition relationship inherited from an ancestral class that it may modify in permissible ways .

Specialisation principle of component addition :- A descendant class may add a decomposition relationship with a new component class .

Specialisation principle of aggregate addition :- a descendant class may add a composition relationship with a new aggregate class .

Specialisation principle of component multiplicity restriction :- A descendant class may restrict the multiplicity domain of a decomposition relationship inherited from an ancestral class .

Aggregation principle of component multiplicity inheritance :- A descendant of an aggregate class inherits by default the component multiplicity of each decomposition relationship inherited from an ancestral class, which it may restrict .

Specialisation principle of component specialisation :- A descendant of an aggregate class may restrict an inherited decomposition relationship to selected descendants of the component class .

Specialisation principle of aggregate specialisation :- A descendant of a component class may restrict an inherited composition relationship to selected descendants of the aggregate class .

Aggregation principle of specalized component inheritance :- A descendant of an aggregate class has-as-a part at least one descendant of a mandatory component class , if any specialisation of the component class is complete .

Aggregation principle of specialised aggregate inheritance :- A descendant of a component class is-a-part-of at least one descendant of a mandatory aggregate class if any specialisation of the aggregate class is complete .

CHAPTER 3

TOKEN RING SPECIFICATION & DESIGN

3.1 Definitions

Abort sequence:- A sequence that terminates the transmission of a frame prematurely.

fill. A bit sequence which may be either 0 bits, 1 bits, or any combination thereof.

frame. A transmission unit that carries a protocol data unit (PDU) on the ring.

logical link control (LLC). That part of the data link layer that supports media independent data link functions, and uses the services of the medium access control sublayer to provide services to the network layer.

medium. The material on which the data may be represented. Twisted pairs, coaxial cables, and optical fibers are examples of media.

medium access control (MAC). The portion of the IEEE 802 data station that controls and mediates the access to the ring.

medium interface connector (MIC). The connector between the station and trunk coupling unit (TCU) at which all transmitted and received signals are specified.

monitor. The monitor is that function that recovers from various error situations. It is contained in each ring station; however, only the monitor in one of the stations on a ring is the active monitor at any point in time. The monitor function in all other stations on the ring is in standby mode.

multiple frame transmission. A transmission where more than one frame is transmitted when a token is captured.

network management (NMT). The conceptual control element of a station which interfaces with all of the layers of the station and is responsible for the setting and resetting of control parameters, obtaining reports of error conditions and determining if the station should be connected to or disconnected from the medium.

repeater. A device used to extend the length, topology, or interconnectivity of the transmission medium beyond that imposed by a single transmission segment.

ring latency. In a token ring medium access control system, the time required for a signal to propagate once around the ring. The ring latency time includes the signal propagation delay through the ring medium plus the sum of the propagation delays through each station connected to the token ring.

service data unit (SDU). Information delivered as a unit between adjacent entities which may also contain a PDU of the upper layer.

Station (or data station). A physical device that may be attached to a shared medium local area network for the purpose of transmitting and receiving information on that shared medium. A data station is identified by a destination address.

Token. The symbol of authority that is passed between stations using a token access method to indicate which station is currently in control of the medium.

Transmit. The action of a station generating a frame, token, abort sequence, or fill and placing it on the medium to the next station. In use, this term contrasts with repeat.

Trunk cable. The transmission cable that interconnects two trunk coupling units.

trunk coupling unit (TCU). A physical device that enables a station to connect to a trunk cable. The trunk coupling unit contains the means for inserting the station into the ring or. Conversely bypassing the station.

3.2 General Description

A token ring consists of a set of stations serially connected by a transmission medium. Information is transferred sequentially, bit by bit, from one active station to the next. Each station generally regenerates and repeats each bit and serves as the means for attaching one or more devices (terminals, work-stations) to the ring for the purpose of communicating with other devices on the network. A given station (the one that has access to the medium) transfers information onto the ring, where the information circulates from one station to the next. The addressed destination station(s) copies the information as it passes. Finally, the station that transmitted the information effectively removes the information from the ring. A station gains the right to transmit its information onto the medium when it detects a token passing on the medium. The token is a control signal comprised of a unique signaling sequence that circulates on the medium following each information transfer. Any station, upon detection of an appropriate token, mar capture the token by modifying it to a start-of-frame sequence and appending appropriate control and status fields, address fields, information field, frame-check sequence and the end-of-frame sequence. At the completion of its information transfer and after appropriate checking for proper operation, the station initiates a new token, which provides other stations the opportunity to gain access to the ring. A token holding timer controls the maximum period of time a station shall use (occupy) the medium before passing the token. Multiple levels of priority are available for independent and dynamic assignment depending upon the relative class of service required for any given message. for example, synchronous real-time voice), asynchronous (interactive), immediate (network recovery). The allocation of priorities shall be by mutual agreement among users of the network. Error detection and recovery mechanisms are provided to restore network operation in the event that transmission errors or medium transients (for example, those resulting from station insertion or removal) cause the access method to deviate from normal operation. Detection and recovery for these cases utilize a network monitoring function that is performed in a specific station with back-up capability in all other stations that are attached to the ring.

3. 3 Formats and Facilities

Formats. There are two basic formats used in token rings: tokens and frames. In the following discussion, the figures depict the formats of the fields in the sequence they are transmitted on the medium, with the left-most bit or symbol transmitted first.

Token Format

SD	AC	ED

SD = Starting Delimiter (1 octet)

AC = Access Control (1 octet)

ED = Ending Delimiter (1 octet)

Frame Format

SD	AC	FC	DA	SA	INFO	FCS	ED	FS

SD : Starting Delimiter (1 octet)

AC : Access Control (1 octet)

FC : Frame Control (1 octet)

DA : Destination Address (2 or 6 octets)

INFO Information (0 or more octets)

FCS :Frame Check Sequence (4 octets)

EFS End-of-Frame Sequence

ED : Ending Delimiter (1 octet)

SA = Source Address (2 or 6 octets)

FS : Frame Status (1 octet)

The frame format shall be used for transmitting both medium access control (MAC) and logical link control (LLC) messages to the destination station. It may or may not have an information INFO field.

ABORT –SEQUENCE

SD	ED

This sequence shall be used for the purpose of terminating the transmission of a frame prematurely. The abort sequence may occur any-where in the bit stream; that is, receiving stations shall be able to detect an abort sequence even if it does not occur on octet boundaries.

Fill. When a station is transmitting (as opposed to repeating), it shall transmit fill preceding or following frames, tokens, or abort sequences to avoid what would otherwise be an inactive or indeterminate transmitter state.

Fill may be either 0 or 1 bits or any combination thereof and may be *any number* of bits in length, within the constraints of the token holding timer.

Field Descriptions. The following is a detailed description of the individual fields in the tokens and frames.

Starting Delimiter (SD)

J	K	0	J	K	0	0	0

J = non-data-J

K = non-data-K

0 = binary zero

Access Control (AC)

P	P	P	T	M	R	R	R

PPP = priority bits T = token hit

M = monitor bit RRR = reservation bits

Priority Bits. The priority bits shall indicate the priority of a token and, therefore, which stations are allowed to use the token In a multiple-priority system, stations use different priorities depending on the priority of the PDU to be transmitted. The eight levels of priority increase from the lowest (000) to the highest (111) priority. For purposes of comparing priority values, the priority shall be transmitted most significant bit first; for example, 110 has higher priority than 011 (left-most bit transmitted first).

Token Bit. The token bit is a 0 in a token and a 1 in a frame. When a station with a PDU to transmit detects a token which has a priority equal to or less than the PDU to be transmitted, it may change the token to a start-of-frame sequence and transmit the PDU.

Monitor Bit. The monitor bit is used to prevent a token whose priority is greater than 0 or any frame from continuously circulating on the ring. If an active monitor detects a frame or a high priority token with the monitor bit equal to 1, the frame or token is aborted. This bit shall be transmitted as 0 in all frames and tokens. The active monitor inspects and modifies this bit. All other stations shall repeat this bit as received.

Reservation Bits. The reservation bits allow stations with high priority PDUs to request (in frames or tokens as they are repeated) that the next token be issued at the requested priority. The eight levels of reservation increase from 000 to 111. For purposes of comparing reservation values, the reservation shall be transmitted most significant bit first; for example, 110 has higher priority than 011 (left-most bit transmitted first).

Frame Control (FC)

F	F	Z	Z	Z	Z	Z	Z

The FC field defines the type of the frame and certain MAC and information frame functions.

Frame-Type Bits. The frame-type hits shall indicate the type of the frame as follows:
00 = MAC frame (contains an MAC PDU)
01 = LLC frame (contains an LLC PDU)
lx= undefined format (reserved for future use)

Medium Access Control (MAC) Frames. If the frame-type bits indicate a MAC frame, all stations on the ring shall interpret and, based on the finite state of the station, act on the ZZZZZZ control bits.

Logical Link Control (LLC) Frames. If the frame-type bits indicate an LLC frame, the ZZZZZZ bits are designated as rrrYYY. The rrr bits are reserved and shall be transmitted as 0's in all transmitted frames and ignored upon reception. The YYY bits may be used to carry the priority (Pm) of the PDU from the source LLC entity to the target LLC entity or entities. Note that P (the priority in the access control [AC] field of a frame) is less than or equal to Pm when the frame is transmitted onto the ring.

Destination and Source Address (DA and SA) Fields- Each frame shall contain two address fields: the destination (station) address and the source (station) address, in that order. Addresses may be either 2 or 6 octets in length; however, all stations of a specific LAN shall have addresses of equal length.

Information (INFO) Field. The information field contains 0. 1, or more octets that are intended for MAC, NMT, or LLC. Although there is no maximum length specified for the information field, the time required to transmit a frame may be no greater than the token holding period that has been established for the station.

The format of the information field is indicated in the frame-type bits of the FC field. The frame types defined are MAC frame and LLC frame.

Frame-Check Sequence (FCS). The FCS shall be a 32-bit sequence .The FCS shall be transmitted commencing with the coefficient of the highest term

Ending Delimiter (ED)

J	K	1	J	K	1	I	E

J = non-data-J K = non-data-K

1 = binary one I = intermediate frame bit

E = error-detected bit

The transmitting station shall transmit the delimiter as shown. Receiving stations shall consider the ending delimiter (ED) valid if the first six symbols J K 1 J K 1 are received correctly.

Intermediate Frame Bit (I Bit). To indicate that this is an intermediate (or first) frame of a multiple frame transmission, the I bit shall be transmitted as 1. An I bit of 0 indicates the last or only frame of the transmission.

Error-Detected Bit (E Bit). The error-detected bit (E) shall be transmitted as 0 by the station that originates the token, abort sequence, or frame. All stations on the ring check tokens and frames for errors (for example, FCS error, non-data symbols: see 4.2.1). The E bit of tokens and frames that are repeated shall be set to 1 when a frame with error is detected; otherwise the E bit is repeated as received.

Frame Status (FS).

A	C	r	r	A	C	r	r

A = address-recognized bits C = frame-copied bits

r = reserved bits

reserved bits. These bits are reserved for future standardization. They shall be transmitted as 0's; however, their value shall be ignored by the receivers.

Address-Recognized (A) Bits and Frame-Copied (C) Bits. The A and C bits shall be transmitted as 0 by the station originating the frame. If another station recognizes the destination address as its own address or relevant group address, it shall set the A bits to 1. If it copies the frame (into its receive buffer), it shall also set the C bits to 1. This allows the originating station to differentiate among three conditions:

1. Station non-existent/non-active on this ring
2. Station exists but frame not copied
3. Frame copied

Claim Token MAC Frame (CL__TK). When a station determines that there is no active monitor operating on the ring, it shall send claim token frames and inspect the source address of the claim token MAC frames it receives. If the SA matches its own (MA) address it has claimed the token and shall enter active monitor mode and generate a new token.

Duplicate Address Test MAC Frame (DAT). This frame is transmitted with DA = MA as part of the initialization process. If the frame returns with the A bits set to 1, it indicates that there is another station on the ring with the same address. If such an event occurs, the station's network manager is notified and the station returns to bypass state. A station that copies a DAT frame will ignore it.

Active Monitor Present MAC Frame (AMP). This frame is transmitted by the active monitor. It shall be queued for transmission following the successful purging of the ring or following the expiration of the TAM. Any station that receives this frame shall reset its TSM whose A and C bits equal 0, the TQP is reset. When timer TQP expires, an SMP PDU shall be queued for transmission.

Beacon MAC Frame (BCN). This frame shall be sent as a result of serious ring failure (for example, broken cable, jabbering station, etc). It is useful in localizing the fault.).The immediate upstream station is part of the failure domain about which the beacon is reporting. Therefore the address of the upstream station that was previously recorded is included in the MAC INFO field.

Purge MAC Frame (PRG). This frame is transmitted by the active monitor. It shall be transmitted following claiming the token or to perform reinitialization of the ring following the detection of an M bit set to I or the expiration of timer TVX.

Timer, Return to Repeat (TRR). Each station shall have a timer TRR to ensure that the station shall return to Repeat State. TRR shall have a value greater than the maximum ring latency. The maximum ring latency consists of the signal propagation delay around a maximum-length ring plus the sum of all station latencies.

Timer, Holding Token (THT). Each station shall have a timer THT to control the maximum period of time the station may transmit frames after capturing a token. A station may initiate transmission of a frame if such transmission can be completed before timer THT expires.

Timer, Queue PDU(TQP):- Each station shall have a timer TQP for the purpose of timing the enqueueing of an SMP PDU after reception of an AMP or SMP frame in which the A and C bits were equal to 0. The default time-out value of TQP is 10 ins.

Timer, Valid Transmission (TVX). Each station shall have a timer TVX ,which is used by the active monitor to detect the absence of valid transmissions. The time-out value of TVX shall be the sum of the time-out value of THT plus the time-out value of TRR.

Timer, No Token (TNT). Each station shall have a timer TNT to recover from various token-related error situations. TNT shall have a time-out value equal to TRR plus n times THT (where n is the maximum number of stations on the ring).

Timer, Active Monitor (TAM). Each station shall have a timer TAM which is used by the active monitor to stimulate the enqueuing of an AMP PDU for transmission. The default time-out value of timer TAM shall be 3 s.

Timer, Standby Monitor (TSM). Each station shall have a timer TSM which is used by the stand-by monitor(s) to assure that there is an active monitor on the ring and to detect a continuous stream of tokens. The default time-out value of timer TSM shall be 7 s.

Flags. Flags are used to *remember* the occurrence of a particular event. They shall be set when the event occurs. The flags used are:

I Flag: A flag which is set upon receiving an ED with the I bit equal to 0.

SFS FLAG :A flag which is set upon receiving an SFS sequence.

MA Flag: A flag which is set upon receiving an SA which is equal to the station's address.

Latency Buffer. The latency buffer serves two purposes. The first is to ensure that there are at least 24 bits of latency in the ring. The second is to provide phase jitter compensation. The token management is structured so that only one latency buffer shall be active in a normally functioning ring and is provided by the active monitor in the ring.

3.4 Token Ring Protocols

Frame Transmission. Access to the physical medium (the ring) is controlled by passing a token around the ring. The token gives the downstream (receiving) station (relative to the station passing the token) the opportunity to transmit a frame or a sequence of frames. Upon request for transmission of an LLC PDU or NMT PDU, MAC prefixes the PDU with the appropriate FC, DA, and SA fields and enqueues it to await the reception of a token that may be used for transmission.

Such a token has a priority less than or equal to the priority of the PDU(s) that is to be sent. Upon queuing the PDU for transmission and prior to receiving a usable token, if a frame or an unusable token is repeated on the ring, the station requests a token of appropriate priority in the RRR bits of the repeated AC field. Upon receipt of a usable token, it is changed to a start-of-frame sequence by setting the token bit. At this time, the station stops repeating the incoming signal and begins transmitting a frame. During transmission, the FCS for the frame is accumulated and appended to the end of the information field.

Token Transmission. After transmission of the frames has been completed, the station checks to see if the station's address has returned in the SA field, as indicated by the MA__FLAG. If it has not been seen, the station transmits fill until the MA_ FLAG is set, at which time the station transmits a token.

Stripping. After transmission of the token, the station will remain in transmit state until all of the frames that the station originated are removed from the ring. This is done to avoid unnecessary recovery action that would be caused if a frame were allowed to continuously circulate on the ring.

Frame Reception. Stations, while repeating the incoming signal stream, check it for frames they should copy or act upon. If the frame-type bits indicate a MAC frame, the control bits are interpreted by all stations on the ring. In addition, if the frame's DA field matches the station's individual address, relevant group address, or broadcast address, the FC, DA. SA, INFO, and FS fields are copied into a receive buffer and subsequently forwarded to the appropriate sublayer.

Priority Operation. The priority bits (PPP) and the reservation bits (RRR) contained in the access control (AC) field work together in an attempt to match the service priority of the ring to the highest priority PDU that is ready for transmission on the ring. These values are stored in registers as Pr and Rr. The current ring service priority is indicated by the priority bits in the AC field, which is circulated on the ring. The priority mechanism operates in such a way that fairness (equal access to the ring) is maintained for all stations within a priority level. This is accomplished by having the same station that raised the service priority level of the ring (the stacking station) return the ring to the original service priority. The priority operation is explained as follows: When a station has a priority (a value greater than zero) PDU (or EDU's) ready to transmit, it requests a priority token. This is done by changing the reservation bits (RRR) as the station repeats the AC field. If the priority level (Pm) of the PDU that is ready for transmission is greater than the RRR bits, the station increases the value of RRR field to the value Pm. If the value of the ERR bits is equal to or greater than Pm, the reservation bits (RRR) are repeated unchanged. After a station has claimed the token, the station transmits PDUs that are at or above the present ring service priority level until it has completed transmission of those PDUs or until the transmission of another frame could not be completed before timer TNT expires). The priority of all of the PDUs that are transmitted should be at the present ring service priority value. The station will then generate a new token for transmission on the ring. If the station does not have additional PDUs to transmit that have a priority (Pm) or does not have a reservation request (as contained in register Rr) neither of which is greater than the present ring service priority (as contained in register Pr), the token is transmitted with its priority at the present ring service priority and the reservation bits RRR) at the greater of Rr or Pm and no further action taken. However, if the station has a PDI ready for transmission or a reservation request (Er), either of which is greater than the present ring service priority, the token is generated with its priority at the greater of Pm or Rr and its reservation bits (RRR) as 0. Since the station has raised the service priority level of the ring, the station becomes a stacking station and, as such, stores the value of the old ring service priority as Sr and the new ring service priority as Sx. (These values will be used later to lower the service priority of the ring when there are no PDUs ready to transmit on the ring whose Pm is equal to or greater than the stacked Sx.)Having become a stacking station, the station claims every token that it receives that has a priority (PPP) equal to its highest stacked transmitted priority (Sx) in order to examine the RRR bits of the AC field for the purpose of raising, maintaining, or lowering the service priority of the ring. The new token is transmitted with its PPP bits equal to the value of the reservation bits (RRR) but no lower than the value of the highest stacked received priority (Sr), which was the original ring priority service level. If the value of the new ring service priority (PPP equal to Rr) is greater than Sr the HER bits are transmitted as 0, the old ring service priority contained in Sx is replaced with a new value Sx equal to Rr , and the station continues its role as a stacking station. However, if the Rr value is equal to or less than the value of the highest stacked received priority (Sr) the new token is transmitted at a priority value of the Sr, both Sx and Sr are removed *(popped)* from the stack, and if no other values of Sx and Sr are stacked, the station discontinues its role as a stacking station. The frames that are transmitted to initialize the ring have a PPP field that is equal to 0. The receipt of a PPP field whose value is less than a stacked Sx will cause any Sx or Sr values that may be stacked to be cleared in all stations on the ring.

Beaconing and Neighbor Notification. When a hard failure is detected in a token ring, its cause must be isolated to the proper failure domain so that recovery actions can take place. The failure domain consists of

1. the station reporting the failure (the beaconing station)
2. the station upstream of the beaconing station
3. the ring medium between them

To do accurate problem determination, all elements of the failure domain must be known at the time that the failure is detected. This implies that at any given time, each station should know the identity of its upstream neighbor station. A process for obtaining this identity known as Neighbor Notification is described below. Neighbor Notification has its basis in the address-recognized and frame-copied bits(the A and C bits) of the FS field. These bits are transmitted as O's . If a station recognizes the destination address of the frame as one of its own , the station sets the A bits to 1 in the passing frame. If a station also copies the *frame,* then the C bits are also set to a 1.When a frame is broadcast to all stations on a ring, the first station downstream of the broadcaster will see that the A and C bits are all 0's. Since a broadcast frame will have its destination address recognized by all of the stations on the ring, the first station downstream will, in particular, set the A bits to 1. All stations further downstream will, therefore. not see the A and C bits as all 0's. This process continues in a circular, daisy-chained fashion to let eves' station know the identity of its upstream neighbor The monitor begins Neighbor Notification by broadcasting the active monitor present (AMP) MAC frame. The station immediately downstream from it takes the following actions:

1.resets its timer TSM, based on seeing the AMP value in the FC field;

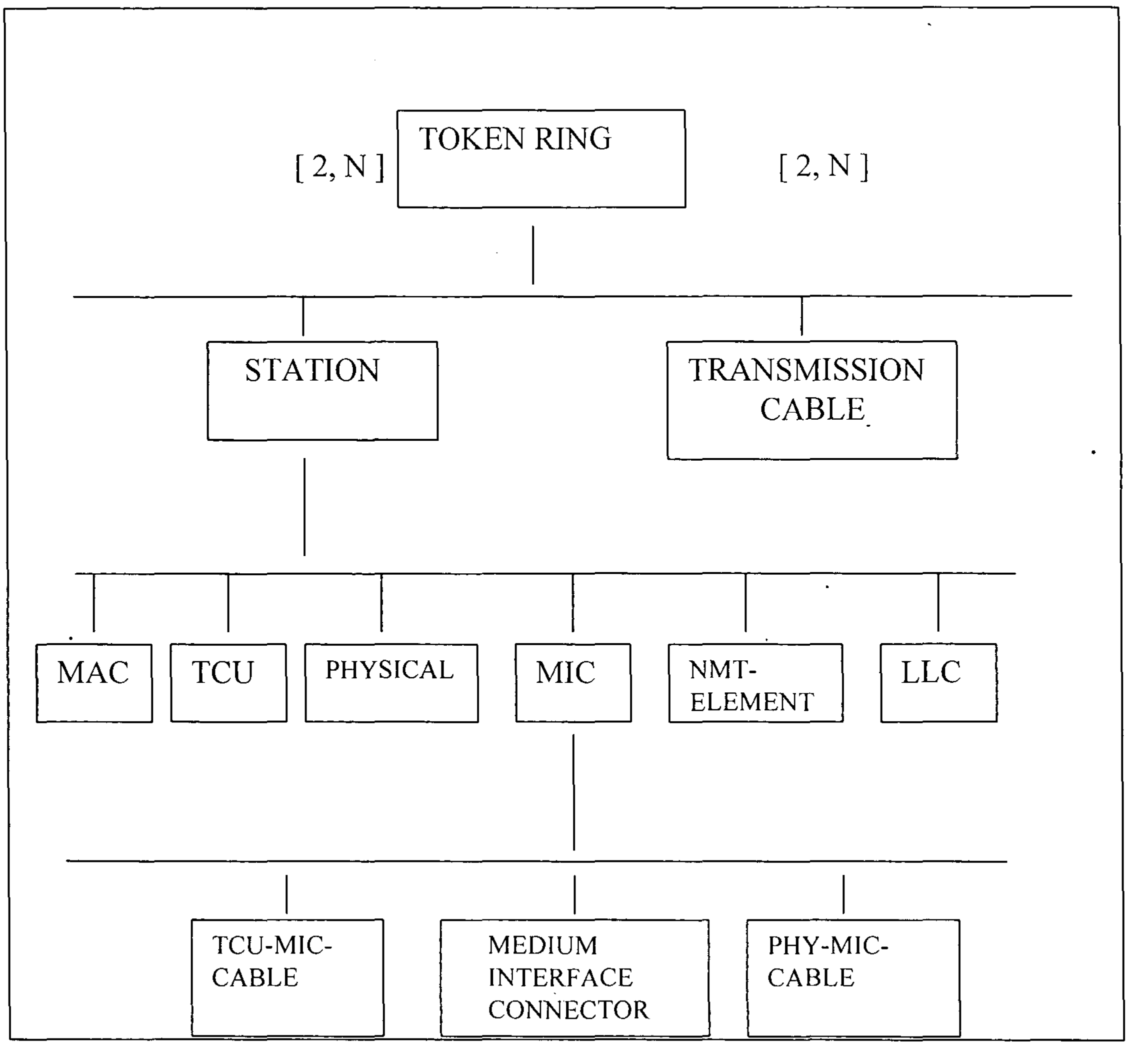
2. if possible, copies the broadcast AMP MAC frame and stores the up streamstation's identity in an upstream neighbor's address (UNA) memory location;
3. sets the A bits (and C bits if the frame was copied) of the passing frame to l's
4. at a suitable transmit opportunity, broadcasts a similar standby monitor present (SMP) MAC frame.

One by one, each station receives an SMP frame with the A and C bits set to 0's, stores its UNA, and continues the process by broadcasting such a frame itself. Since the AMP frame must pass each station on a regular basis (the active monitor present MAC frame sent by the monitor), the continuous transmission of tokens onto a ring can be detected. In addition to the timer TAM in the active monitor, each standby station has a timer TSM that is reset each time an AMP MAC frame passes. If timer TSM expires, that standby monitor station begins transmitting claim token frames. From the received symbols MAC detects various types of input data, such as tokens, MAC frames, and LLC information frames. In turn, MAC stores values, sets flags, and performs certain internal actions as well as generating tokens, frames, or fill, or flipping bits and delivering them to the PHY layer in the form of a serial stream of the 0, 1, J. and K symbols. For the purpose of accumulating the FCS and storing the contents of a frame, J and K symbols that are not part of the SD or ED shall be interpreted as 1 and 0 bits, respectively.

3.5 Carving out the design of token ring network

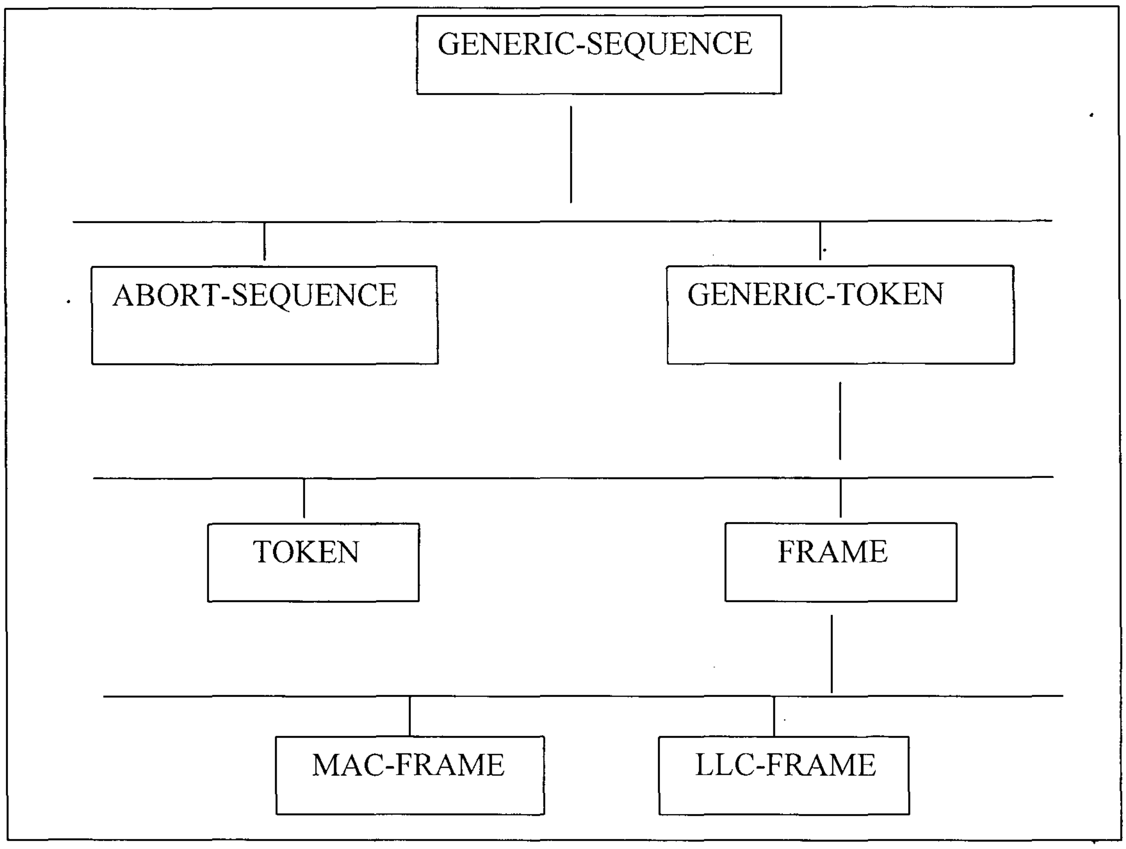
The token ring Network consists of the following distinguished components. Token Ring as a whole, stations, transmission cable, transmission control unit , medium interface cable , medium interface connector , network management and network administrator .

AGGREGATION:- Considering Token Ring Network as a whole , the stations are its main components . For a network the number of stations may be at least two or more but less than a fixed number . Corresponding to a station there is a medium interface cable and TCU (Transmission control unit). Therefore they will be shown as part of station class. The medium interface cable consists of medium interface connector, tcu-mic-cable and physical-layer-mic-cable. The working of station is layered in physical-layer, macSublayer and nmtElement. The aggregation hierarchy can be shown as follows

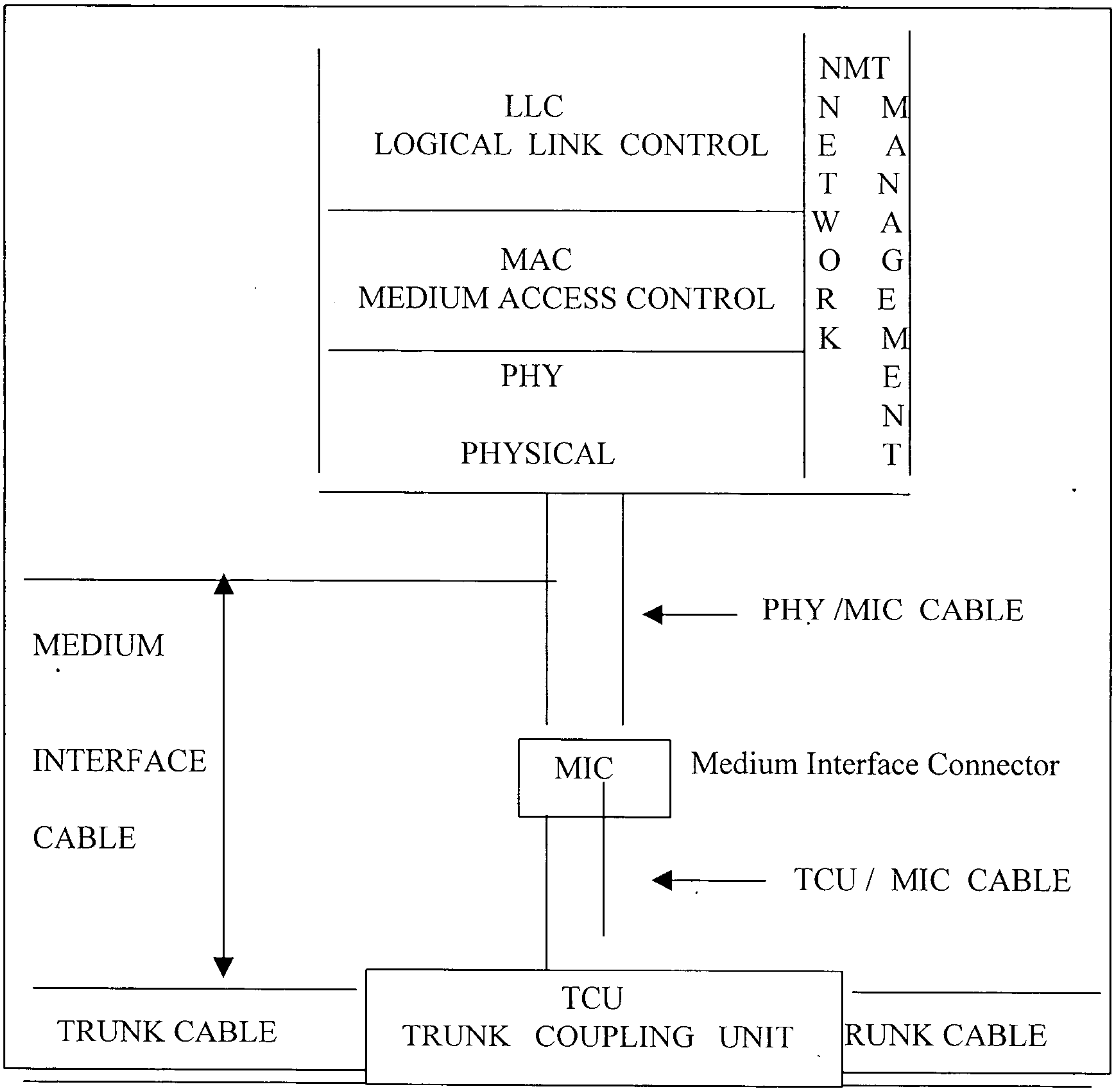


AGGREGATION HIERARCHY OF TOKEN RING

INHERITANCE :- Token and frame are two important components which are transmitted over the network. In the access-control if the token-bit is zero it means it is token otherwise it is a frame. So the token bit is chosen as a basis of specialization. A generic-token is the abstract class from which token and frame classes are derived using token bit as a basis of specialization . There is another sequence termed as abort-sequence consisting of start-delimiter and ending-delimiter only . An abstract class called as generic-sequence is created for the purpose of specification reuse. There are different types of frames known as mac-frame and llc-frame These are differentiated on the basis of frame-type-bits. So it is used as a basis of specialization. The specialization principle of attribute domain restriction is applied here .The inheritance tree created for the token and frames is as shown below.



Identifying the attributes and behavior:- The abort-sequence , token and frame all contain the start-delimiter and ending-delimiter . The start-delimiter and ending-delimiter are sequence of bits . They have no actual instance and they are not independently instantiated . So they are treated as capsules . Further access-control is also not instantiated independently so it is also treated as capsule . Generic-sequence is developed as an abstract class for the purpose of specification reuse. This class will have the functionality of checking start and ending-delimiter Another abstract class called generic-token class is required to be created as the capsule access-control is added providing the access and set functions .The frame class needs to have source and destination address, frame-check-sequence and info part. In addition it will have frame-control and frame-status. The class abort-sequence is meant to stop the transmission so it will have a function abort-transmission.



Layering:- The functioning of the station class is decomposed into layers viz. Physical-layer, mac-sublayer, llc-sublayer and nmtElement. Access and transmission is controlled by the mac-sublayer.These layers are arranged on the basis of layer principles i.e. lower layers provide services to upper layers through service access points . The arrangement of layers is as shown in the figure .

The logical link control layer accesses services from medium access control sublayer in order to services to higher layers which are not shown in the figure . In this dissertation layering is kept limited upto the llc sublayer . Medium access control sublayer accesses services provided by physical layer and interacts with nmtElement class for proper network management . It provides all information like status changes to nmtElement . Physical layer also interacts with nmtElement so that functions like insertion or removal of a station can be controlled automatically by network management without human intervention.

SERVICES :- The following are the interactions between logical link control layer and medium access control sublayer

MA-DATA-request.
MA-DATA-indication
MA-DATA-confirmation

MA-DATA-request:- This primitive is used by llc-sublayer to request macSublayer to send a service data unit to another llc-sublayer.

MA-DATA-indication :-It will be used by macSublayer to send a macSublayer frame to llc-sublayer to indicate the arrival of an llc frame at the local macSublayer entity .

MA-DATA-confirmation :- This primitive is used to provide an appropriate response to the llc-sublayer signifying the success or failure of the request

PHY to MAC Service. The services provided by the PHY layer allow the local MAC sublayer entity to exchange MAC data units with peer MAC sub-layer entities.
Interactions. The following primitives are defined for the MAC sub-layer to request service from the PHY layer:
PH-DATA-request
PH-DATA-indication
PH-DATA-confirmation

PH-DATA-request. This primitive defines the transfer of data from a local MAC sublayer entity to the station's PHY layer. The MAC sublayer shall send the PHY layer a PH-DATA-request every time the MAC sublayer has a symbol to output.

PH-DATA-indication. This primitive defines the transfer of data from the PHY layer to the MAC sublayer entity. The PHY layer shall use this primitive to send the MAC sublayer a PH-DATA-indication every time the PHY layer decodes a symbol.

PH-DATA.confirmation. This primitive will be used by physical layer to provide an appropriate response to the MAC sublayer PH-DATA-request signifying the acceptance of a symbol specified by the PH-DATA-request and willingness to accept another symbol.

MAC-NMT Services:- This interface is used by NMT to monitor and control operations of the MAC sublayer. The following primitives are defined for the NMT to request service from the MACsublayer.

MA-INITIALIZE-PROTOCOL-request
MA-INITIALIZE-PROTOCOL-confirmation
MA-CONTROL-request
MA-STATUS-indication
MA-NMT-DATA-request
MA-NMT-DATA-indication
MA-NMT-DATA-confirmation.

MA-INITIALIZE-PROTOCOL-request:-This primitive is used by NMT to reset the MAC sublayer and optionally to change the operational parameters of the MAC sublayer.

MA-INITIALIZE-PROTOCOL-confirmation:-This primitive is used by the MAC sublayer to inform NMT that the MA-INITIALIZE-PROTOCOL-request primitive is complete.

MA-STATUS-indication:-This primitive is used by the MAC sublayer to inform NMT of errors and significant status changes.

MA-NMT-DATA-request:-This primitive shall be generated by the NMT entity whenever data must be transferred to one or more NMT entities.

MA-NMT-DATA-indication:-This primitive defines the transfer of data from the MAC sublayer entity.The MA-NMT-DATA-indication primitive shall be generated by the MAC sublayer entity to the NMT entity to indicate the arrival of a MAC frame at the local MAC sublayer entity.

MA-NMT-DATA-confirmation:-This primitive shall provide an appropriate response to the NMT's MA-NMT-DATA-request primitive signifying the success or failure of the request.

PHY-NMT SERVICES :-The following primitives are defined for the NMT to request services from the physical layer.

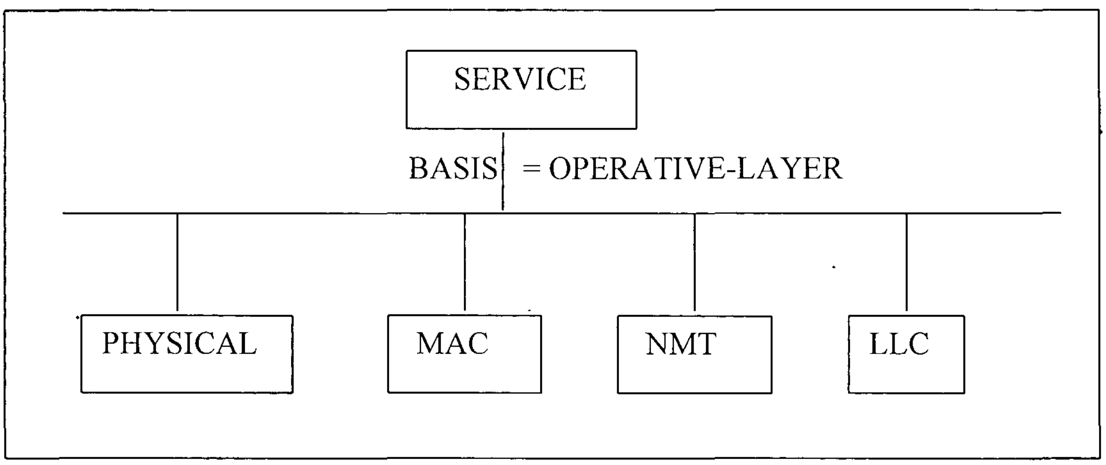
PH-CONTROL-request
PH-STATUS-indication

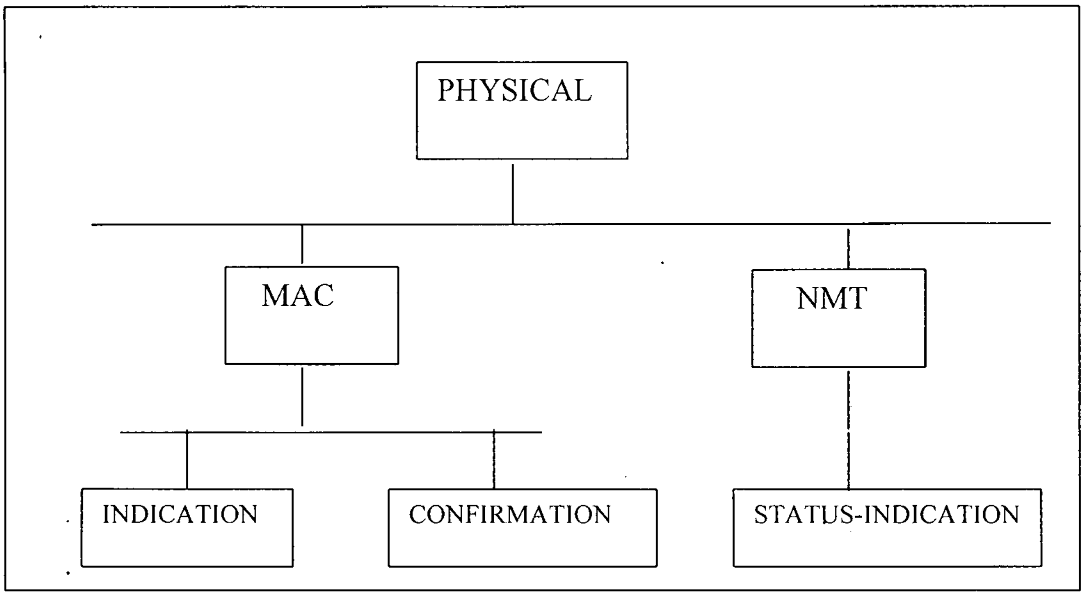
PH-CONTROL-request :- This primitive shall be generated by NMT to request the physical layer to insert or remove itself to/from the ring.

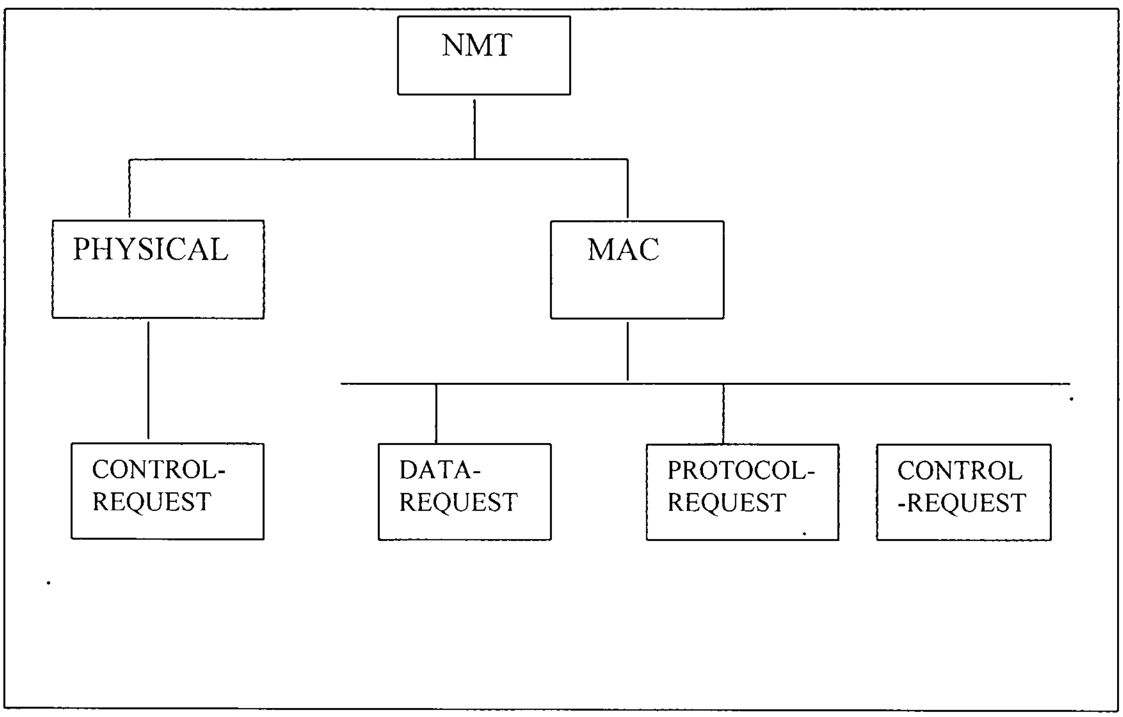
PH-STATUS-indication:-This primitive is used by the physical layer to inform NMT of errors and significant status changes.

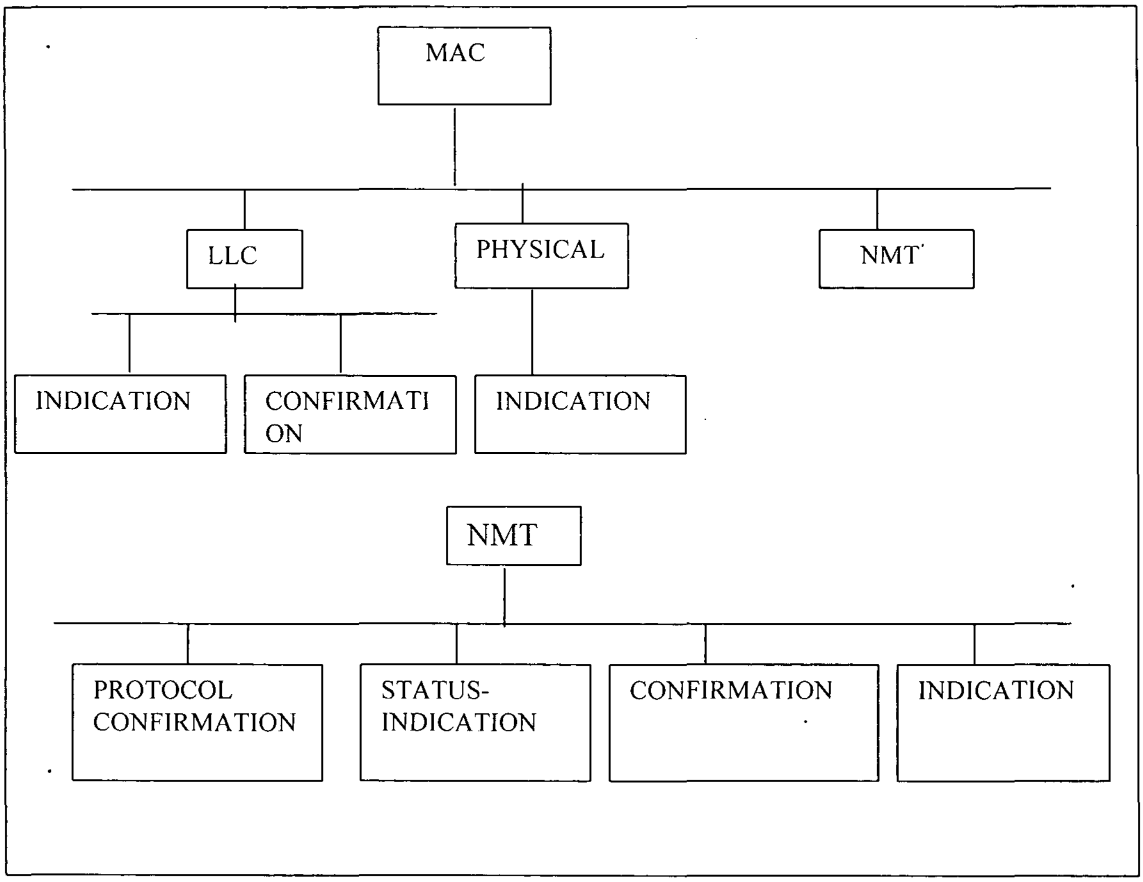
TRUNK-CABLE:-The function of the trunk cable medium is to transport data signals between successive stations of a baseband ring local area network. This communication medium consists of a set of TCU's interconnected sequentially by trunk cable links. Each TCU is connected to a TCU/MIC cable to which a station may be connected . The relationship between these embodiments and the LAN model is as shown in the figure. Repeaters may be used where required to extend the length of a trunk link beyond limits imposed by normal signal degradation due to link impairments. These repeaters serve to restore the amplitude,shape and timing of signals passing through them. The repeater's regenerative functions have the same characteristics as a repeating station on the ring and must be included in the count of the number of stations supported by the ring.

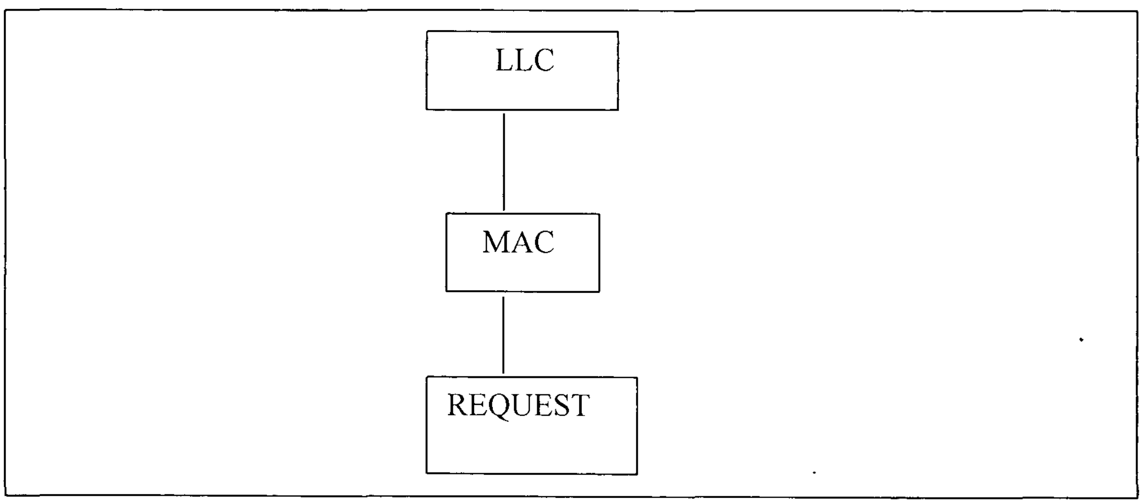
Ring Access Control:- Station insertion into the ring is controlled by the station. The mechanism for effecting the insertion or bypass of the station resides in the TCU. The staion exercises control of the mechanism via the media interface cable .




PHYSICAL
MAC
NMT
INDICATION
CONFIRMATION
STATUS-INDICATION


NMT
PHYSICAL
MAC
CONTROL-REQUEST
DATA-REQUEST
PROTOCOL-REQUEST
CONTROL-REQUEST





The services are classified on the basis of operative layer , receiving layer and the type of services . The inheritance tree of service class is shown. A service abstract class is created which has the common attributes and functions of all services . Choosing operative layer as the basis of specialisation. It is derived into four subclasses viz. Physical ,mac,nmtElement and llcSublayer . Specialisation is done using the specialisation principle of attribute domain restriction . as in case of each of these sublayers domain of the attribute operative layer has been restricted . These service classes are further derived into other services based on the value of the attribute receiving layer . This attribute specifies the layer using that service . In the inheritance tree downward derivation is again done on the basis of service type .

Identifying attributes and functions of the layers :- The mac sublayer is responsible for controlling access to the medium . so it needs certain timer by virtue of which it may control the access. These timers are timer-holding-token ,timer-return-to-repeat, timer-valid-transmission, timer-no-token, timer-queue-PDU, timer-standby-monitor and timer-active-monitor. These attributes have certain fixed values when the timers are reset. The mac-sublayer informs the nmtElement success or failure of the requests made by it so it needs to have an attribute status .The layer has to check frame condition and its report has to be sent to nmt so another attribute required is status-report. This layer needs the functions to check frame condition , reset timers TNT and TSM , enqueue the SMP-PDU, check for presence of active monitor , duplicate address , transmit sdu, append various fields to a frame , hold and regenerate the token on the ring . The physical layer carries out its operation with the aid of trunk-coupling–unit and medium-interface-cable . The trunk-coupling-unit has two state repeat and transmit so it has an attribute state which can have the values repeat and transmit and corresponding functions. The medium-interface-cable consists of physical-mic-cable, tcu-mic-cable and medium interface connector. The nmtElement class has to take actions in error conditions so it has parameters for different error conditions like beacon-state, active-monitor-not-present ,duplication-of-address ,insertion and removal of station from the ring and corresponding functions .

CHAPTER 4

DESIGN IN ASN.1

```
-- The following is the definition of the aggregation modeling construct

AGGREGATION                 ::=      CLASS
{
        &aggregateClass              OBJECT-CLASS
        &componentClass              OBJECT-CLASS
        &componentMultiplicity       Multiplicity
        &aggregationLabel            OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX
{
        AGGREGATE                    &aggreagteClass
[       WITH MULTIPLICITY            &aggreagteMultiplicity        ]
        HAS-AS-A-PART
        COMPONENT                    &componentClass
[       WITH MULTIPLICITY            &componentMultiplicity        ]
        IDENTIFIED BY                &aggreagtionLabel
}

stationPhysical_layer                AGGREGATION :: =
{
        AGGREGATE                    station
        HAS-AS-A-PART
        COMPONENT                    physical_layer
        IDENTIFIED BY                stationPhysical_layerLbl
}
-- Physical Layer is a component of station class

stationLlc_layer                     AGGREGATION :: =
{
        AGGREGATE                    station
        HAS-AS-A-PART
        COMPONENT                    llc_layer
        IDENTIFIED BY                stationLlc_layerLbl
}
-- Llc is a component of station class

stationMac_sublayer                  AGGREGATION ::=
{
        AGGREGATE                    station
        HAS-AS-A-PART
```

```
        COMPONENT                   mac_sublayer
        IDENTIFIED BY               stationMac_sublayerLbl
}
-- Mac Layer is a component class of station class

stationNmt_element                  AGGREGATION ::=
{
        AGGREGATE                   station
        HAS-AS-A-PART
        COMPONENT                   nmt_element
        IDENTIFIED BY               stationNmt_elementLbl
}
-- Nmt_element  is the network manager and controls the overall operation

stationTcu                          AGGREGATION ::=
{
        AGGREGATE                   station
        HAS-AS-A-PART
        COMPONENT                   tcu
        IDENTIFIED BY               stationTcuLbl
}
-- Tcu stands for Transmission control unit

stationMic                          AGGREGATION ::=
{
        AGGREGATE                   station
        HAS-AS-A-PART
        COMPONENT                   mic
        IDENTIFIED BY               stationMicLbl
}
--  Mic stands for medium interface cable

tokenringStation                    AGGREGATION ::=
{
        AGGREGATE                   tokenring
        HAS-AS-A-PART
        COMPONENT                   station
        WITH MULTIPLICITY                 {
                                          theRangeWith :
                                          { lowerBound { constant : 2};
                                            upperBound {ade : {
                                          numberOfStationsLbl} }
                                          }
                                    }
        IDENTIFIED BY               tokenringStationLbl
}
-- Token Ring contains at least two stations but the  maximum   number  of  stations
-- depends on the attribute  numberofStations.
```

```
micMediumInterfaceConnector          AGGREAGTION ::=
{
        AGGREGATE                    mic
        HAS-AS-A-PART
        COMPONENT                    mediumInterfaceConnector
        IDENTIFIED BY                micMedsiumInterfaceConnectorLbl
}

micPhy/micCable                      AGGREGATION ::=
{
        AGGREGATE                    mic
        HAS-AS-A-PART
        COMPONENT                    phy/micCable
        IDENTIFIED BY                micPhy/micCableLbl
}

micTcu/micCable                      AGGREGATION ::=
{
        AGGREGATE                    mic
        HAS-AS-A-PART
        COMPONENT                    tcu/micCable
        IDENTIFIED BY                micTcu/micCableLbl
}
-- The following is the definition of the capsule modeling construct

CAPSULE     :: = CLASS
{
        &Attributes                  ATTRIBUTE  OPTIONAL
        &Functions                   FUNCTION   OPTIONAL
        &Capsules                    CAPSULE    OPTIONAL

        &capsuleLabel                     OBJECT IDENTIFIER UNIQUE
}

WITH SYNTAX
{
        [ ATTRIBUTES                 &Attributes           ]
        [ FUNCTIONS                  &Functions            ]
        [ CAPSULES                   &Capsules             ]
        IDENTIFIED BY                &capsuleLabel
}

start-delimiter                      CAPSULE ::=
{
        ATTRIBUTES
                        {
                                     symbol_J_one,
```

```
                                        symbol_K_one,
                                        symbol_zero_one,
                                        symbol_J_two,
                                        symbol_K_two,
                                        symbol_zero_two,
                                        symbol_zero_three,
                                        symbol_zero_four
                        }
        IDENTIFIED BY                   start-delimiterLbl
}

access-control                          CAPSULE ::=
{
        ATTRIBUTES
                        {
                                        priority,
                                        token-bit,
                                        monitor-bit,
                                        reserved-priority
                        }
        IDENTIFIED BY                   access-controlLbl
}

ending-delimiter                        CAPSULE ::=
{
        ATTRIBUTES
                        {
                                        symbol-J-one,
                                        symbol-K-one,
                                        symbol-one-one,
                                        symbol-J-two,
                                        symbol-K-two,
                                symbol-one-two,
                                symbol-I,
                                symbol-E
                        }
        IDENTIFIED BY           ending-delimiterLbl
}
-- generic sequence is the abstract class from which abort-sequence class is derived

generic-sequence                        OBJECT-CLASS ::=
{
        MANDATORY  CAPSULES
                                        {
                                                start-delimiter,
                                                ending-delimiter,
                                        }
        FUNCTIONS               {
```

```
                              check-start-delimiter,-- checks starting delimiter
                              check-ending-delimiter,
                    }
      IDENTIFIED BY           generic-sequenceLbl
}

abort-sequence                OBJECT-CLASS ::=
{
      SPECIALIZES-FROM
                    {{
                              superclass          generic-sequenceLbl,
                              basisOfSpecialization
                              {simplePredicate :
                                    {functionAdded :    abort-
                                          transmissionLbl }
                              }
                     }}
      IDENTIFIED BY     abort-sequenceLbl }

-- abort-sequence is the  class used for aborting the transmission

generic-token           OBJECT-CLASS ::=
{
      SPECIALIZES-FROM
                    {{
                              superclass          generic-sequenceLbl,
                              basisOfSpecialization
                              { simplePredicate :
                                                  {capsuleAdded :
                                                  access-controlLbl}
                              }
                    }}
      FUNCTIONS
                {
                        get-priority,
                        get-token-bit,
                        get-monitor-bit,
                        get-reserved-priority,
                        set-priority,
                        set-token-bit,
                        set-monitor-bit,
                        set-reserved-priority
                }

      IDENTIFIED BY     generic-tokenLbl
}
-- generic token is also an abstract class
-- get function returns the value of that attribute
```

```
-- set function modifies the value of that attribute

token                  OBJECT-CLASS
{
      SPECIALIZES-FROM
                         {{
                               superclass      generic-tokenLbl,
                               basisOfSpecialization
                               { simplePredicate :
                               {attributeDomainRestricted :
                                     {
                                          newValueOf :
                                          {attribute      token-bit,
                                            is             equalTo,
                                            value                0
                                          }
                                     }
                               }
                               }
                         }}—token is derived from the generic-token class on
the
                   }—token-bit attribute with 0 value
      IDENTIFIED BY      tokenLbl
}

frame                         OBJECT-CLASS ::=
{
      SPECIALIZES-FROM
                         {{
                               superclass      generic-tokenLbl,
                               basisOfSpecialization
                               {simplePredicate :
                               {attributeDomainRestricted :
                                     { newValueOf:
                                          { attribute      token-bit,
                                            is             equalTo,
                                            value                1
                                          }
                                     }
                               }}
                         }}
      ATTRIBUTES
                   {
                         destination-address,
                         source-address,
                         frame-check-sequence,
                         info                    OPTIONAL
                   }
```

```
        CAPSULES
                {
                        frame-control,
                        frame-status
                }
        IDENTIFIED BY           frameLbl
}
-- frame class is also derived from generic-token class with token-bit =1

frameLbl ::=frame.&objectClassLabel

frame-control                   CAPSULE ::=
{
        ATTRIBUTES
                {
                        frame-type-bits,
                        control-type-bits,
                }
        IDENTIFIED BY   frame-controlLbl
}

macframe                        OBJECT-CLASS ::=
{
        SPECIALIZES-FROM
                        {{
                                superclass      frameLbl,
                                basisOfSpecialization
                                {simplePredicate :
                                {attributeDomainRestricted:
                                {newValueOf :
                                {attribute      frame-type-bits,
                                  is            equalTo,
                                   value                0
                                }
                                }}}
                        }}
}
-- macframe is derived from the frame class with frame-type-bit =0

llcframe                        OBJECT-CLASS ::=
{
        SPECIALIZES-FROM
                        {{
                                superclass      frameLbl,
                                basisOfSpecialization
                                {simplePredicate :
                                {attributeDomainRestricted:
                                {newValueOf :
```

```
                                        {attribute      frame-type-bits,
                                          is            equalTo,
                                          value              1
                                        }
                                        }}}
                                   }}
}
-- llcframe is derived from the frame class with frame-type-bit = 1

OBJECT-CLASS                       ::=            CLASS
{
        &SpecializesFrom     Specialization OPTIONAL,
        &Attributes          ATTRIBUTE  OPTIONAL,
        &Functions           FUNCTION   OPTIONAL,
        &MandatoryCapsules   CAPSULE    OPTIONAL,
        &OptionalCapsules    CAPSULE    OPTIONAL,
        &ObjectClassLabel    OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX
{
        [SPECIALIZES-FROM               &Specialization            ]
        [ATTRIBUTES                     &Attributes                ]
        [FUNCTIONS                &Functions              ]
        [MANDATORY CAPSULES             &MandatoryCapsules         ]
        [OPTIONAL CAPSULES              &OptionalCapsules          ]
        IDENTIFIED BY                   &objectClassLabel
}
-- The following is the definition of the attribute modelling construct

ATTRIBUTE                          ::=     CLASS
{
        &AttributeType,
        &attributeDomain                Domain{&AttributeType}OPTIONAL,
        &attributeLabel                 OBJECT IDENTIFIER UNIQUE
}
WITH SYNTAX
{
        ATTRIBUTE-TYPE                  &AttributeType
        [ATTRIBUTE-DOMAIN               &attributeDomain     ]
        IDENTIFIED BY                   &attributeLabel
}
-- The following is the definition of the function modelling construct

FUNCTION                           ::=     CLASS
{
        &Arguments                      ARGUMENT        OPTIONAL,
        &Results                        RESULT          OPTIONAL,
        &Exceptions                     EXCEPTION       OPTIONAL,
```

```
        &Specification                    SET OF SEQUENCE OF
                                          formalSpecification   OPTIONAL,
        &functionLabel                    OBJECT      IDENTIFIER  UNIQUE
}
WITH SYNTAX
{
        [ARGUMENTS                        &Arguments          ]
        [RESULTS                          &Results            ]
        [EXCEPTIONS                       &Exceptions         ]
        [SPECIFICATION                    &Specification              ]
        IDENTIFIED BY                     &functionLabel
}

-- The following is the definition of the argument modelling construct

ARGUMENT                        ::=       CLASS
{
        &ArgumentType,
        &argumentDomain                   Domain{&ArgumentType}
        &argumentLabel                    OBJECT      IDENTIFIER  UNIQUE

}

WITH SYNTAX
{
        ARGUMENT-TYPE                     &ArgumentType
        [ARGUMENT-DOMAIN                  &argumentDomain    ]
        IDENTIFIED BY                     argumentLabel
}

-- The following is the definition of the result modelling construct

RESULT                          ::=       CLASS
{
        &ResultType,
        &ResultDomain                     Domain{&ResultType}
        &resultLabel                      OBJECT      IDENTIFIER  UNIQUE
}
WITH          SYNTAX
{
        RESULT-TYPE                       &ResultType
        [RESULT-DOMAIN                    &ResultDomain               ]
        IDENTIFIED BY                     &resultLabel
}

-- The following is the definition of the exception modelling construct
EXCEPTION                       ::=       CLASS
{
```

```
        &ExceptionParamType                 OPTIONAL
        &exceptionParamDomain               Domain{&ExceptionParamType}
        &exceptionLabel                     OBJECT      IDENTIFIER UNIQUE
}
WITH        SYNTAX
{
[EXCEPTION-PARAMETER-TYPE           &ExceptionParamType
[EXCEPTION-PARAMTETER-DOMAIN  &exceptionParamDomain        ]
IDENTIFIED BY                       &exceptionLabel.
} .

Bit             ::=     INTEGER(0..1)

symbol-J-one                    ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
        IDENTIFIED BY           symbol-J-oneLbl
}

symbol-J-oneLbl       ::= symbol-J-one.&objectClassLabel

symbol-K-one                    ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
        IDENTIFIED BY           symbol-K-oneLbl
}

symbol-K-oneLbl ::=symbol-K-one.&objectClassLabel

symbol-zero-one                 ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
        IDENTIFIED BY           symbol-zero-oneLbl
}

symbol-zero-oneLbl  ::= symbol-zero-one.&objectClassLabel

symbol-J-two                    ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
        IDENTIFIED BY           symbol-J-twoLbl
}

symbol-J-twoLbl  ::= symbol-J-two.&objectClassLabel

symbol-K-two                    ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
```

```
	IDENTIFIED BY		symbol-K-twoLbl
}

symbol-K-twoLbl ::= symbol-K-two.&objectClassLabel


symbol-zero-two			ATTRIBUTE ::=
{
	ATTRIBUTE-TYPE		Bit,
	IDENTIFIED BY		symbol-zero-twoLbl
}

symbol-zero-twoLbl ::= symbol-zero-two.&objectClassLabel

symbol-zero-three		ATTRIBUTE ::=
{
	ATTRIBUTE-TYPE		Bit,
	IDENTIFIED BY		symbol-zero-threeLbl
}

symbol-zero-threeLbl ::= symbol-zero-three.&objectClassLabel

symbol-zero-four		ATTRIBUTE ::=
{
	ATTRIBUTE-TYPE		Bit,
	IDENTIFIED BY		symbol-zero-fourLbl
}

symbol-zero-fourLbl ::= symbol-zero-four.&objectClassLabel

Priority		::=	ENUMERATED{0,1,2,3,4,5,6,7}

priority				ATTRIBUTE ::=
{
	ATTRIBUTE-TYPE		Priority,
	IDENTIFIED BY		priorityLbl
}
priorityLbl	::=priority.&objectClassLabel

token-bit			ATTRIBUTE ::=
{
	ATTRIBUTE-TYPE		Bit,
	IDENTIFIED BY		token-bitLbl
}token-bitLbl ::=	token-bit.&objectClassLabel

monitor-bit			ATTRIBUTE ::=
{
	ATTRIBUTE-TYPE		Bit,
```

```
        IDENTIFIED BY               monitor-bitLbl
}
monitor-bitLbl::=       monitor-bit.&objectClassLabel

reserved-priority                   ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Priority,
        IDENTIFIED BY               reserved-priorityLbl
}
reserved-priorityLbl   ::=reserved-priority.&objectClassLabel

symbol-one-one                      ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Bit,
        IDENTIFIED BY               symbol-one-oneLbl
}
symbol-one-oneLbl  ::= symbol-one-one.&objectClassLabel

symbol-one-two                      ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Bit,
        IDENTIFIED BY               symbol-one-twoLbl
}
symbol-one-twoLbl  ::= symbol-one-two.&objectClassLabel

symbol-I                            ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Bit,
        IDENTIFIED BY               symbol-ILbl
}
symbol-ILbl  ::= symbol-I.&objectClassLabel

symbol-E                            ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Bit,
        IDENTIFIED BY               symbol-ELbl
}
symbol-ELbl  ::= symbol-E.&objectClassLabel

source-address                      ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Address,
        IDENTIFIED BY               source-addressLbl

}
source-addressLbl              ::=     source-address.&objectClassLabel

destination-address                 ATTRIBUTE ::=
```

```
{
        ATTRIBUTE-TYPE              Address,
        IDENTIFIED BY               destination-addressLbl

}
destination-addressLbl              ::=     destination-address.&objectClassLabel

info                                ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              INFO,
        IDENTIFIED BY               infoLbl
}
infoLbl                             ::=     info.&objectClassLabel

INFO                        ::=     SEQUENCE OF OCTET
STRING(SIZE(NUMBER))
NUMBER                      ::=     ENUMERATED {2,6}

Frame-check-sequence                ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              INTEGER32,
        IDENTIFIED BY               frame-check-sequenceLbl
}
frame-check-sequenceLbl     ::=     frame-check-sequence.&objectClassLabel

frame-type-bits                     ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Bit,
        IDENTIFIED BY               frame-type-bitsLbl
}
frame-type-bitsLbl          ::=     frame-type-bits.&objectClassLabel

control-type-bits                   ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Bit,
        IDENTIFIED BY               control-type-bitsLbl
}
control-type-bitsLbl        ::=     control-type-bits.&objectClassLabel

check-start-delimiter               FUNCTION  ::=
{
        ARGUMENTS                   start-delimiter
        RESULTS                     result
        IDENTIFIED BY               check-start-delimiterLbl
}
check-start-delimiterLbl    ::=     check-start-delimiter.&objectClassLabel

start-delimiter                     ARGUMENT ::=
```

```
{
        ARGUMENT-TYPE              CAPSULE,
        IDENTIFIED BY              start-delimiterLbl
}
start-delimiterLbl          ::=    start-delimiter.&objectClassLabel

result                             RESULT     ::=
{
        RESULT-TYPE                Bit,
        IDENTIFIED BY              resultLbl
}
resultLbl                   ::=    result.&objectClassLabel

check-ending-delimiter             FUNCTION  ::=
{
        ARGUMENTS                  ending-delimiter
        RESULTS                    result
        IDENTIFIED BY              check-ending-delimiterLbl
}
check-ending-delimiterLbl   ::=    check-ending-delimiter.&objectClassLabel

ending-delimiter                   ARGUMENT ::=
{
        ARGUMENT-TYPE              CAPSULE,
        IDENTIFIED BY              ending-delimiterLbl
}
ending-delimiterLbl         ::=    ending-delimiter.&objectClassLabel

get-priority                       FUNCTION  ::=
{
        RESULTS                    priority,
        IDENTIFIED BY              get-priorityLbl
}
get-priorityLbl             ::=    getpriority.&objectClassLabel

get-token-bit                      FUNCTION  ::=
{
        RESULTS                    token-bit,
        IDENTIFIED BY              get-token-bitLbl
}
get-token-bitLbl            ::=    get-token-bit.&objectClassLabel

token-bit                          RESULT     ::=
{
        RESULT-TYPE                Bit,
        IDENTIFIED BY              token-bitLbl
}
token-bitLbl                ::=    token-bit.&objectClassLabel
```

```
get-monitor-bit                              FUNCTION  ::=
{
        RESULTS                              monitor-bit,
        IDENTIFIED BY                        get-monitor-bitLbl
}
get-monitor-bitLbl               ::=         get-monitor-bit.&objectClassLabel

monitor-bit                                  RESULT     ::=
{
        RESULT-TYPE                          Bit,
        IDENTIFIED BY                        monitor-bitLbl
}
monitor-bitLbl                   ::=         monitor-bit.&objectClassLabel

get-reserved-priority                        FUNCTION  ::=
{
        RESULTS                              reserved-priority,
        IDENTIFIED BY                        get-reserved-priorityLbl
}
get-reserved-priorityLbl         ::=         get-reserved-priority.&objectClassLabel

reserved-priority                            RESULT     ::=
{
        RESULT-TYPE                          Priority,
        IDENTIFIED BY                        reserved-priorityLbl
}
reserved-priorityLbl             ::=         reserved-priority.&objectClassLabel

set-priority                                 FUNCTION  ::=
{
        ARGUMENTS                            priority,
        RESULTS                              priority,
        IDENTIFIED BY                        set-priorityLbl
}
set-priorityLbl                  ::=         setpriority.&objectClassLabel

priority                                     ARGUMENT ::=
{
        ARGUMENT-TYPE                        Priority,
        IDENTIFIED BY                        priorityLbl
}
priorityLbl                      ::=         priority.&objectClassLabel

priority                                     RESULT     ::=
{
        RESULT-TYPE                          Priority,
        IDENTIFIED BY                        priorityLbl
```

```
}
priorityLbl                 ::=     priority.&objectClassLabel

set-monitor-bit                         FUNCTION   ::=
{
        ARGUMENTS                   monitor-bit,
        RESULTS                     monitor-bit,
        IDENTIFIED BY               set-monitor-bitLbl
}
set-monitor-bitLbl          ::=     set-monitor-bit.&objectClassLabel

monitor-bit                         ARGUMENT ::=
{
        ARGUMENT-TYPE               Bit,
        IDENTIFIED BY               monitor-bitLbl
}
monitor-bitLbl              ::=     monitor-bit.&objectClassLabel

set-reserved-priority               FUNCTION   ::=
{
        ARGUMENT                    reserved-priority,
        RESULTS                     reserved-priority,
        IDENTIFIED BY               set-reserved-priorityLbl
}
set-reserved-priorityLbl    ::=     set-reserved-priority.&objectClassLabel

reserved-priority                   ARGUMENT ::=
{
        ARGUMENT-TYPE               Priority,
        IDENTIFIED BY               reserved-priorityLbl
}
reserved-priorityLbl        ::=     reserved-priority.&objectClassLabel

frame-status                        CAPSULE    ::=
{
        ATTRIBUTES
                        {
                                    first-A-bit,
                                    first-C-bit,
                                    first-r-bit,
                                    second-r-bit,
                                    second-A-bit,
                                    second-C-bit,
                                    third-r-bit,
                                    fourth-r-bit
                        }
        IDENTIFIED BY               frane-statusLbl
}
```

```
frame-statusLbl          ::=      frame-status.&objectClassLabel

first-A-bit                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             first-A-bitLbl
}
first-A-bitLbl           ::=      first-A-bit.&objectClassLabel

first-C-bit                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             first-C-bitLbl
}
first-C-bitLbl           ::=      first-C-bit.&objectClassLabel

first-r-bit                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             first-r-bitLbl
}
first-r-bitLbl           ::=      first-r-bit.&objectClassLabel



second-A-bit                      ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             second-A-bitLbl
}
second-A-bitLbl          ::=      second-A-bit.&objectClassLabel

second-C-bit                      ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             second-C-bitLbl
}
second-C-bitLbl          ::=      second-C-bit.&objectClassLabel

second-r-bit                      ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             second-r-bitLbl
}
second-r-bitLbl          ::=      second-r-bit.&objectClassLabel

third-r-bit                       ATTRIBUTE ::=
```

```
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             third-r-bitLbl
}
third-r-bitLbl            ::=     third-r-bit.&objectClassLabel

forth-r-bit                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE            Bit,
        IDENTIFIED BY             forth-r-bitLbl
}
fourth-r-bitLbl           ::=     fourth-r-bit.&objectClassLabel

Specialization      ::=           SEQUENCE
{
        superclass                ObjectClassLabel,
        basisOfSpecialization     LogicalPredicate
}

abort-transmission                FUNCTION  ::=
{
        RESULTS                   status,
        IDENTIFIED BY             abort-tarnsmissionLbl
}
abort-transmissionLbl ::=         abort-transmission.&objectClassLabel

status                            RESULT     ::=
{
        RESULT-TYPE               Bit,
        IDENTIFIED BY             statusLbl
}
statusLbl           ::=           status.&objectClassLabel

service                           OBJECT-CLASS          ::=
{
        ATTRIBUTES
                {
                                  serviceName,
                                  serviceType,
                                  serviceEntity,
                                  operativeLayer,
                                  receivingLayer
                }
        IDENTIFIED BY             serviceLbl
}
serviceLbl          ::=           service.&objectClassLabel

physical-LayerService             OBJECT-CLASS          ::=
```

```
{
        SPECIALIZES-FROM {{
                                        Superclass      serviceLbl,
                                        BasisOfSpecialization
                                        {simplePredicate:
                                        {attributeDomainRestricted:
                                        {newValueOf:
                                        {attribute      operativeLayerLbl,
                                         is             equalTo,
                                         value          physical-Layer,
                                        }}}}
}}

ph-data-indication              OBJECT-CLASS    ::=
{
        SPECIALIZES-FROM                {{
                                superclass      physiacal-LayerLbl,
                                basisOfSpecialization {compoundPredicate:
                                 {connectBy and,
                                        predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              serviceType,
                                 is                     equalTo,
                                 value                  indication
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              receivingLayer,
                                 is                     equalTo,
                                 value                  macSublayer
                                        }}}}

                                        }}
        IDENTIFIED BY           ph-data-indicationLbl
}
ph-data-indicationLbl ::=       ph-data-indication.&objectClassLabel


ph-data-confirmation            OBJECT-CLASS    ::=
{
        SPECIALIZES-FROM                {{
                                superclass      physiacal-LayerLbl,
                                basisOfSpecialization {compoundPredicate:
                                      {connectBy and,
                                predicates
```

```
                                        {simplePredicate:
                                        {attributeDomainRestricted:
                                        {newValueOf:
                                        {attribute          serviceType,
                                         is                 equalTo,
                                         value              confirmation
                                        }}}}
                                        {simplePredicate:
                                        {attributeDomainRestricted:
                                        {newValueOf:
                                        {attribute          receivingLayer,
                                         is                 equalTo,
                                         value              macSublayer
                                             }}}}

                                        }}
        IDENTIFIED BY                   ph-data-confirmationLbl
}
ph-data-confirmationLbl        ::=      ph-data-confirmation.&objectClassLabel

ph-status-indication                    OBJECT-CLASS     ::=
{
        SPECIALIZES-FROM                     {{
        superclass                      macSublayerLbl,
                                        basisOfSpecialization {compoundPredicate:
                                            {connectBy and,
                                              predicates
                                        {simplePredicate:
                                        {attributeDomainRestricted:
                                        {newValueOf:
                                        {attribute          serviceType,
                                         is                 equalTo,
                                         value              status-indication
                                        }}}}
                                        {simplePredicate:
                                        {attributeDomainRestricted:
                                        {newValueOf:
                                        {attribute          receivingLayer,
                                         is                 equalTo,
                                         value              nmtElement
                                        }}}}
                                        }}
        IDENTIFIED BY                   ph- status-indication Lbl
}
ph- status-indication Lbl      ::=      ph- status-indication .&objectClassLabel

ma-status-indication                    OBJECT-CLASS     ::=
{
```

```
SPECIALIZES-FROM                {{
                        superclass      macSublayerLbl,
                        basisOfSpecialization {compoundPredicate:
                            {connectBy and,
                        predicates
                        {simplePredicate:
                        {attributeDomainRestricted:
                        {newValueOf:
                        {attribute              serviceType,
                         is                     equalTo,
                    value                  status-indication
                    }}}}
                    {simplePredicate:
                    {attributeDomainRestricted:
                    {newValueOf:
                    {attribute             receivingLayer,
                     is                    equalTo,
                     value                 nmtElement
                    }}}}

                    }}
    IDENTIFIED  BY  ma- status-indication Lbl
}
ma- status-indication Lbl     ::=     ma- status-indication .&objectClassLabel

macSublayerService          OBJECT-CLASS            ::=
{
    SPECIALIZES-FROM {{
                    Superclass      serviceLbl,
                    BasisOfSpecialization
                    {simplePredicate:
                    {attributeDomainRestricted:
                    {newValueOf:
                    {attribute      operativeLayerLbl,
                     is             equalTo,
                     value          macSublayer,
                    }}}}
                    }}
}

ph-data-request                 OBJECT-CLASS      ::=
{
    SPECIALIZES-FROM                {{
                    superclass      macSublayerLbl,
                    basisOfSpecialization {compoundPredicate:
                     {connectBy and,
                    predicates
                    {simplePredicate:
```

```
                        {attributeDomainRestricted:
                        {newValueOf:
                        {attribute          serviceType,
                         is                 equalTo,
                         value              request
                            }}}}
                            {simplePredicate:
                            {attributeDomainRestricted:
                            {newValueOf:
                            {attribute          receivingLayer,
                             is                 equalTo,
                             value              physical-Layer
                            }}}}

ma-nmt-data-indication          OBJECT-CLASS     ::=
{
    SPECIALIZES-FROM                {{
                            superclass      macSublayerLbl,
                            basisOfSpecialization {compoundPredicate:
                                {connectBy and,
                            predicates
                            {simplePredicate:
                            {attributeDomainRestricted:
                            {newValueOf:
                            {attribute          serviceType,
                             is                 equalTo,
                             value              indication
                            }}}}
                            {simplePredicate:
                            {attributeDomainRestricted:
                            {newValueOf:
                            {attribute          receivingLayer,
                             is                 equalTo,
                             value              nmtElement
                            }}}}

ma-data-indication              OBJECT-CLASS     ::=
{
    SPECIALIZES-FROM                {{
                            superclass      macSublayerLbl,
                            basisOfSpecialization {compoundPredicate:
                                {connectBy and,
                            predicates
                            {simplePredicate:
                            {attributeDomainRestricted:
                            {newValueOf:
                            {attribute          serviceType,
                             is                 equalTo,
```

```
                                value                   indication
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              receivingLayer,
                                 is                     equalTo,
                                 value                  llcSublayer
                                }}}}

ma-nmt-data-confirmation        OBJECT-CLASS    ::=
{
    SPECIALIZES-FROM                {{
                                superclass      macSublayerLbl,
                                basisOfSpecialization {compoundPredicate:
                                    {connectBy and,
                                predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              serviceType,
                                 is                     equalTo,
                                 value                  confirmation
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              receivingLayer,
                                 is                     equalTo,
                                 value                  nmtElement
                                }}}}

}}
    IDENTIFIED BY               ma-nmt-data-confirmationLbl
}
ma-nmt-data-confirmationLbl::=  ma-nmt-data-confirmation.&objectClassLabel

ma-data-confirmation            OBJECT-CLASS    ::=
{
    SPECIALIZES-FROM                {{
                                superclass      macSublayerLbl,
                                basisOfSpecialization {compoundPredicate:
                                    {connectBy and,
                                predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              serviceType,
```

```
                                is                      equalTo,
                                value                   confirmation
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              receivingLayer,
                                is                      equalTo,
                                value                   llcSublayer
                                }}}}

}}
    IDENTIFIED BY               ma-data-confirmationLbl
}
ma-data-confirmationLbl   ::=   ma-data-confirmation.&objectClassLabel

ma-initialize-protocol-confirmation  OBJECT-CLASS   ::=
{   SPECIALIZES-FROM                {{
                                superclass   macSublayerLbl,
                                basisOfSpecialization {compoundPredicate:
                                    {connectBy and,
                                predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              serviceType,
                                is                      equalTo,
                                value                   procol-confirmation
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute              receivingLayer,
                                is                      equalTo,
                                value                   nmtElement
                                }}}}
}}
    IDENTIFIED BY               ma-initialize-protocol-confirmationLbl
}
ma-initialize-protocol-confirmationLbl    ::=    ma-initialize-protocol-
confirmation.&objectClassLabel

ma-data-request                 OBJECT-CLASS   ::=
{
    SPECIALIZES-FROM                {{
                                superclass   llcSublayerLbl,
                                basisOfSpecialization {compoundPredicate:
                                    {connectBy and,
```

```
                                predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute            serviceType,
                                 is                   equalTo,
                                 value                request
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute            receivingLayer,
                                 is                   equalTo,
                                 value                macSublayer
                                }}}}

ma-nmt-data-request             OBJECT-CLASS    ::=
{
    SPECIALIZES-FROM                {{
                                superclass    nmtElementLbl,
                                basisOfSpecialization {compoundPredicate:
                                    {connectBy and,
                                predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute            serviceType,
                                 is                   equalTo,
                                 value                request
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute            receivingLayer,
                                 is                   equalTo,
                                 value                macSublayer
                                }}}}


ma-initialize-protocol-request  OBJECT-CLASS    ::=
{
    SPECIALIZES-FROM                {{
                                superclass    nmtElementLbl,
                                basisOfSpecialization {compoundPredicate:
                                    {connectBy and,
                                predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
```

```
                                {newValueOf:
                                {attribute           serviceType,
                                 is                  equalTo,
                                 value               protocol-request
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute           receivingLayer,
                                 is                  equalTo,
                                 value               macSublayer
                                }}}}
}

ma-control-request              OBJECT-CLASS    ::=
{
        SPECIALIZES-FROM            {{
                                superclass      nmtElementLbl,
                                basisOfSpecialization {compoundPredicate:
                                     {connectBy and,
                                predicates
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute           serviceType,
                                 is                  equalTo,
                                 value               contol-request
                                }}}}
                                {simplePredicate:
                                {attributeDomainRestricted:
                                {newValueOf:
                                {attribute           receivingLayer,
                                 is                  equalTo,
                                        value                macSublayer
                                        }}}}
}

PROTOCOL-LAYERING               ::=     CLASS
{
        &upperProtocol                  ProtocolLabel,
        &lowerProtocol                  ProtocolLabel,
        &demultiplexPoint               INTEGER     OPTIONAL,
        &layeringLabel                  OBJECT IDENTIFIER UNIQUE
}

WITH SYNTAX
{
        PROTOCOL                        &upperProtocol
```

```
        LAYERS-ABOVE
        PROTOCOL                      &lowerProtocol
        [DEMULTIPLEX POINT            &demultiplexPoint    ]
        IDENTIFIED BY                 &layerLabel
}

llcSublayer-macSublayer               PROTOCOL-LAYERING   :      ::=
{
        PROTOCOL                      llcSublayer
        LAYERS-ABOVE
        PROTOCOL                      macSublayer
        IDENTIFIED BY                 llcSublayer-macSublayerLbl
}
llcSublayer-macSublayerLbl ::=        llcSublayer-
macSublayer.&objectClassLabel


macSublayer-physicalLayer             PROTOCOL-LAYERING   ::=
{
        PROTOCOL                      macSublayer
        LAYERS-ABOVE
        PROTOCOL                      physicalLayer
        IDENTIFIED BY                 macSublayer-physicalLayerLbl
}
macSublayer-physicalLayerLbl     ::=  macSublayer-
physicalLayer.&objectClassLabel



nmtElement-physicalLayer         PROTOCOL-LAYERING   ::=
{
        PROTOCOL                 nmtElement
        LAYERS-ABOVE
        PROTOCOL                 physicalLayer
        IDENTIFIED BY            nmtElement-physicalLayerLbl
}
nmtElement-physicalLayerLbl      ::=    nmtElement-
physicalLayer.&objectClassLabel

nmtElement-macSublayer           PROTOCOL-LAYERING   :      ::=
{
        PROTOCOL                 nmtElement
        LAYERS-ABOVE
        PROTOCOL                 macSublayer
        IDENTIFIED BY            nmtElement-macSublayerLbl
}
nmtElement-macSublayerLbl ::=    nmtElement-macSublayer.&objectClassLabel
```

```
serviceName                     ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          PrintableString
        IDENTIFIED BY           serviceNameLbl
}
serviceNameLbl          ::=     serviceName.&objectClassLabel

PrintableString                 ::=     SEQUENCE OF OCTET STRING

ServiceType                     ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          PrintableString,
        IDENTIFIED BY           serviceTypeLbl
}
serviceTypeLbl          ::=     serviceType.&objectClassLabel

operativeLayer                          ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Layer
        IDENTIFIED BY           operativeLayerLbl
}
operativeLayerLbl       ::=     operativeLayer.&objectClassLabel

LAYER                   ::=     ENUMERATED
{
        physicalLayer(0),
        macSublayer(1),
        llcSublayer(2),
        nmtElement(3)
}

receivingLayer                          ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Layer
        IDENTIFIED BY           receivingLayerLbl
}
receivingLayerLbl       ::=     receivingLayer.&objectClassLabel

macSublayer                     OBJECT-CLASS            ::=
{
        ATTRIBUTES
                {
                                individual-mac-address,
                                group-mac-address,
                                tht-value,
                                trr-value,
                                tvx-value,
                                tnt-value,
```

```
                                        tqp-value,
                                        tsm-value,
                                        tam-value,
                                        priority-Amp-Data-Unit,
                                        status,
                                        m-sdu,
                                        status-report,
                                        frame-control,
                                        reception-status,
                                        transmission-status,
                                        provided-service-class,
                                        symbol
                }
        FUNCTIONS
                {
                                        report-transmission-status,
                                        receive-frame-control,
                                        receive-m-sdu-identification,
                                        receive-requested-service-class,
                                        master-reset,
                                        insert,
                                        check-frame-condition,
                                        check-for-active-monitor,
                                        set-parameters,
                                        reset-TNT,
                                        reset-TSM,
                                        enqueue-SMP-PDU,
                                        set-flag,
                                        append-DA-to-msdu,
                                        transmit-msdu-macSublayer,
                }
        IDENTIFIED BY                   macSublayerLbl
}
macSublayerLbl                  ::=     macSublayer.&objectClassLabel

nmtElement                              OBJECT-CLASS            ::=
{
        ATTRIBUTES
                {
                                        control-action,
                                        frame-control,
                                        destination-address,
                                        m-sdu,
                                        requested-service-class,
                                        status,
                }
        FUNCTIONS
                {
```

```
                                        reporting-frame-condition,
                                        report-transmission-status,
                                        report-activemonitor-not-present,
                                        report-beacon-state,
                                        report-provided-service-class,
                                        report-frame-control,
                                        report-destination-address,
                                        report-source-address,
                                        report-m-sdu,
                                        report-reception-status,
          }
     IDENTIFIED BY                      nmtElementLbl
}
nmtElementLbl               ::=         nmtElement.&objectClassLabel

status-report                           ATTRIBUTE ::=
{
     ATTRIBUTE-TYPE                     PrintableString,
     IDENTIFIED BY                      status-reportLbl
}
status-reportLbl            ::=         status-report.&objectClassLabel

transmission-status                     ATTRIBUTE ::=
{
     ATTRIBUTE-TYPE                     Bit,
     IDENTIFIED BY                      transmission-statusLbl
}
transmission-statusLbl                  ::=     transmission-status.&objectClassLabel


get-symbol                              FUNCTION  ::=
{
     RESULTS                            symbol,
     IDENTIFIED BY                      get-symbolLbl
}
get-symbolLbl               ::=         get-symbol.&objectClassLabel

symbol                                  RESULT    ::=
{
     RESULT-TYPE                        Bit,
     IDENTIFIED BY                      symbolLbl
}
symbolLbl                   ::=         symbol.&objectClassLabel

symbol-decoded                          FUNCTION  ::=
{
     ARGUMENTS                          stream,
     RESULTS                            symbol,
```

```
        IDENTIFIED BY           symbol-decodedLbl
}
symbol-decodedLbl       ::=     symbol-decoded.&objectClassLabel

stream                          ARGUMENT ::=
{
        ARGUMENT-TYPE           Stream,
        IDENTIFIED BY           streamLbl
}
streamLbl               ::=     stream.&objectClassLabel

burst-correction-start          FUNCTION  ::=
{
        RESULTS                 symbol
        IDENTIFIED BY           burst-correction-startLbl
}
burst-correction-startLbl  ::=  burst-correction-start.&objectClassLabel


burst-correction-end            FUNCTION  ::=
{
        ARGUMENTS               transition,
        IDENTIFIED BY           burst-correction-endLbl
}
burst-correction-endLbl    ::=  burst-correction-end.&objectClassLabel


latency-buffer-overflow         FUNCTION  ::=
{
        RESULTS                 status-report,
        IDENTIFIED BY           latency-buffer-overflowLbl
}
latency-buffer-overflowLbl ::=  latency-buffer-overflow.&objectClassLabel

status-report                   RESULT    ::=
{
        RESULT-TYPE             PrintableString,
        IDENTIFIED BY           status-report.&objectClassLabel
}

latency-buffer-underflow        FUNCTION  ::=
{
        RESULTS                 status-report,
        IDENTIFIED BY           latency-buffer-underflowLbl
}
latency-buffer-underflowLbl ::= latency-buffer-underflow.&objectClassLabel

tht-value                       ATTRIBUTE ::=
```

```
{
        ATTRIBUTE-TYPE          TimeStamp,
        IDENTIFIED BY           tht-valueLbl
}
tht-valueLbl            ::=     tht-value.&objectClassLabel

trr-value                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          TimeStamp,
        IDENTIFIED BY           trr-valueLbl
}
trr-valueLbl            ::=     trr-value.&objectClassLabel

tvx-value                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          TimeStamp,
        IDENTIFIED BY           tvx-valueLbl
}
tvx-valueLbl            ::=     tvx-value.&objectClassLabel

tnt-value                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          TimeStamp,
        IDENTIFIED BY           tnt-valueLbl
}
tnt-valueLbl            ::=     tnt-value.&objectClassLabel

tqp-value                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          TimeStamp,
        IDENTIFIED BY           tqp-valueLbl
}
tqp-valueLbl            ::=     tqp-value.&objectClassLabel

tsm-value                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          TimeStamp,
        IDENTIFIED BY           tsm-valueLbl
}
tsm-valueLbl            ::=     tsm-value.&objectClassLabel

tam-value                       ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          TimeStamp,
        IDENTIFIED BY           tam-valueLbl
}
tam-valueLbl            ::=     tam-value.&objectClassLabel
```

```
individual-mac-address          ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Address,
        IDENTIFIED BY           individual-mac-addressLbl
}
individual-mac-addressLbl   ::= individual-mac-address.&objectClassLabel

group-mac-address               ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          AddressSet,
        IDENTIFIED BY           group-mac-addressLbl
}
group-mac-addressLbl            ::=     group-mac-address.&objectClassLabel

AddressSet                  ::= SET OF  Address

priority-Amp-Data-Unit          ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Priority
        IDENTIFIED BY           priority-Amp-Data-UnitLbl
}
priority-Amp-Data-UnitLbl   ::= priority-Amp-Data-Unit.&objectClassLabel

status                          ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
        IDENTIFIED BY           statusLbl
}
statusLbl                   ::= status.&objectClassLabel

reception-status                ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
        IDENTIFIED BY           reception-statusLbl
}
reception-statusLbl         ::= reception-status.&objectClassLabel

m-sdu                           ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          INFO,
        IDENTIFIED BY           m-sduLbl
}
m-sduLbl                    ::= m-sdu.&objectClassLabel

transmission-status             ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE          Bit,
        IDENTIFIED BY           transmission-statusLbl
```

```
}
transmission-statusLbl              ::=     transmission-status.&objectClassLabel

provided-service-class              ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE              Priority,
        IDENTIFIED BY               provide-service-classLbl
}
provided-service-classLbl     ::=   provided-service-class.&objectClassLabel

check-frame-conditio                FUNCTION  ::=
{
        ARGUMENT                    input-frame,
        RESULTS                     status-report,
        IDENTIFIED BY               check-frame-conditionLbl
}
check-frame-conditionLbl      ::=   check-frame-condition.&objectClassLabel

input-frame                         ARGUMENT ::=
{
        ARGUMENT-TYPE               Frame,
        IDENTIFIED BY               input-frameLbl
}
input-frameLbl                ::=   input-frame.&objectClassLabel

check-for-active-monitor            FUNCTION  ::=
{
        RESULTS                     value,
        IDENTIFIED BY               check-for-active-monitorLbl
}
check-for-active-monitorLbl   ::=   check-for-active-monitor.&objectClassLabel

value                               RESULT     ::=
{
        RESULT-TYPE                 Bit,
        IDENTIFIED BY               valueLbl,
}
valueLbl                      ::=   value.&objectClassLabel

set-parameters                      FUNCTION  ::=
{
        RESULTS                     result-status
        IDENTIFIED BY               set-parameterLbl
}
set-parameterLbl              ::=   set-parameter.&objectClassLabel

result-status                       RESULT     ::=
{
```

```
        RESULT-TYPE             Bit,
        IDENTIFIED BY           result-statusLbl
}
result-statusLbl            ::= result-status.&objectClassLabel

reset-TNT                       FUNCTION ::=
{
        ARGUMENTS               tnt-value,
        RESULTS                 tnt-value,
        IDENTIFIED BY           reset-TNTLbl
}
reset-TNTLbl                ::= reset-TNT.&objectClassLabel

tnt-value                       ARGUMENT ::=
{
        ARGUMENT-TYPE           TimeStamp,
        IDENTIFIED BY           tnt-valueLbl
}
tnt-valueLbl                ::= tnt-value.&objectClassLabel

tnt-value                       RESULT ::=
{
        RESULT-TYPE             TimeStamp,
        IDENTIFIED BY           tnt-valueLbl
}
tnt-valueLbl                ::= tnt-value.&objectClassLabel


reset-TSM                       FUNCTION ::=
{
        ARGUMENTS               tsm-value,
        RESULTS                 tsm-value,
        IDENTIFIED BY           reset-TSMLbl
}
reset-TSMLbl                ::= reset-TSM.&objectClassLabel

tsm-value                       ARGUMENT ::=
{
        ARGUMENT-TYPE           TimeStamp,
        IDENTIFIED BY           tsm-valueLbl
}
tsm-valueLbl                ::= tsm-value.&objectClassLabel

tsm-value                       RESULT ::=
{
        RESULT-TYPE             TimeStamp,
        IDENTIFIED BY           tsm-valueLbl
}
```

```
tsm-valueLbl                   ::=     tsm-value.&objectClassLabel

enqueue-SMP-PDU                        FUNCTION  ::=
{
        ARGUMENTS                      pdu,
        RESULTS                        position-in-queue,
        IDENTIFIED BY                  enqueue-SMP-PDULbl
}
enqueue-SMP-PDULbl             ::=     enqueue-SMP-PDU.&objectClassLabel

pdu                                    ARGUMENT ::=
{
        ARGUMENT-TYPE                  INFO,
        IDENTIFIED BY                  pduLbl,
}
pduLbl                         ::=     pdu.&objecClassLabel

position-in-queue                      RESULT    ::=
{
        RESULT-TYPE                    INTEGER,
        IDENTIFIED BY                  position-in-queueLbl,
}
position-in-queueLbl           ::=     position-in-queue.&objectClassLabel

set-flag                                       FUNCTION  ::=
{
        ARGUMENTS                      flag,
        RESULTS                        flag,
        IDENTIFIED BY                  set-flagLbl
}
set-flagLbl                    ::=     set-flag.&objectClassLabel

flag                                   ARGUMENT ::=
{
        ARGUMENT-TYPE                  Bit,
        IDENTIFIED BY                  flagLbl
}
flagLbl                        ::=     flag.&objectClassLabel

flag                                   RESULT    ::=
{
        RESULT-TYPE                    Bit,
        IDENTIFIED BY                  flagLbl
}
flagLbl                        ::=     flag.&objectClassLabel

append-source-address-to-m-sdu         FUNCTION  ::=
{
```

```
        ARGUMENTS               m-sdu,
        RESULTS                 m-sdu,
        IDENTIFIED BY           append-source-address-to-m-sduLbl
}
append-source-address-to-m-sduLbl ::=append-source-address-to-m-
sdu.&objectClassLabel

m-sdu                           ARGUMENT ::=
{
        ARGUMENT-TYPE           INFO,
        IDENTIFIED BY           m-sduLbl
}
m-sduLbl                ::=     m-sdu.&objectClassLabel

m-sdu                           RESULT    ::=
{
        RESULT-TYPE             INFO,
        IDENTIFIED BY           m-sduLbl
}

append-DA-to-m-sdu              FUNCTION  ::=
{
        ARGUMENTS               m-sdu,
        RESULTS                 m-sdu,
        IDENTIFIED BY           append-DA-to-m-sduLbl
}
append-DA-to-m-sduLbl   ::=     append-DA-to-m-sdu.&objectClassLabel

transmit-sdu-macSublayer        FUNCTION  ::=
{
        ARGUMENTS               frame,
        RESULTS                 transmission-status,
        IDENTIFIED BY           transmit-sdu-macSublayerLbl
}
transmit-sdu-macSublayerLbl::=  transmit-sdu-macSublayer.&objectClassLabel

transmission-status             RESULT    ::=
{
        RESULT-TYPE             Bit,
        IDENTIFIED BY           transmission-statusLbl
}
transmission-statusLbl          ::=    transmission-status.&objecClassLabel

report-transmission-status      FUNCTION  ::=
{
        RESULTS                 transmission-status
        IDENTIFIED BY           report-transmission-statusLbl
}
```

```
report-transmission-statusLbl ::=       report-transmission-status.&objecClassLabel

receive-frame-control                   FUNCTION   ::=
{
        ARGUMENTS                       frame-control,
        IDENTIFIED BY                   receive-frame-controlLbl
}
receive-frame-controlLbl        ::=     receive-frame-control.&objecClassLabel

frame-control                           ARGUMENT ::=
{
        ARGUMENT-TYPE                   Frame-control,
        IDENTIFIED BY                   frame-controlLbl
}
frame-controlLbl                ::=     frame-control.&objectClassLabel

receive-destination-address             FUNCTION   ::=
{
        ARGUMENTS                       destination-address,
        IDENTIFIED BY                   receive-destination-addressLbl
}
receive-destination-addressLbl          ::=     receive-destination-
address.&objectClassLabel

destination-address                     ARGUMENT ::=
{
        ARGUMENT-TYPE                   Address,
        IDENTIFIED BY                   destination-addressLbl
}
destination-addressLbl                  ::=     destination-address.&objectClassLabel

receive-m-sdu-identification            FUNCTION   ::=
{
        ARGUMENTS                       m-sdu,
        IDENTIFIED BY                   receive-m-sdu-identificationLbl
}
receive-m-sdu-identificationLbl         ::=     receive-m-sdu-
identification.&objectClassLabel

m-sdu                                   ARGUMENT ::=
{
        ARGUMENT-TYPE                   INFO,
        IDENTIFIED BY                   m-sduLbl
}

receive-requested-service-class         FUNCTION   ::=
{
        ARGUMENTS                       requested-service-class,
```

```
    IDENTIFIED BY               receive-requested-service-classLbl
}
receive-requested-service-classLbl    ::=    receive-requested-service-
class.&objectClassLabel

requested-service-class             ARGUMENT ::=
{
    ARGUMENT-TYPE               Priority,
    IDENTIFIED BY               requested-service-classLbl
}

master-reset                        FUNCTION ::=
{
    ARGUMENTS                   parameters,
    RESULTS                     parameters,
    IDENTIFIED BY               master-resetLbl
}
master-resetLbl             ::=     master-reset.&objectClassLabel

parameters                          ARGUMENT ::=
{
    ARGUMENT-TYPE               Parameters,
    IDENTIFIED BY               parametersLbl
}
parametersLbl               ::=     parameters.&objectClassLabel

parameters                          RESULT   ::=
{
    RESULT-TYPE                 Parameters,
    IDENTIFIED BY               parametersLbl
}

insert                              FUNCTION ::=
{
    RESULTS                     status,
    IDENTIFIED BY               insertLbl
}
insertLbl                   ::=     insert.&objectClassLabel

control-action                      ATTRIBUTE ::=
{
    ATTRIBUTE-TYPE              Bit,
    IDENTIFIED BY               control-actionLbl
}
control-actionLbl           ::=     control-action.&objectClassLabel

destination-address                 ATTRIBUTE ::=
{
```

```
        ATTRIBUTE-TYPE          Address,
        IDENTIFIED BY           destination-address Lbl
}
destination-addressLbl          ::=     destination-address.&objectClassLabel

reporting-frame-condition       FUNCTION   ::=
{
        ARGUMENTS               input-frame,
        RESULTS                 status-report,
        IDENTIFIED BY           reporting-frame-conditionLbl
}
reporting-frame-conditionLbl ::=    reporting-frame-condition.&objectClassLabel

report-active-monitor-not-present   FUNCTION   ::=
{
        RESULTS                 status,
        IDENTIFIED BY           report-active-monitor-not-presentLbl
}
report-active-monitor-not-presentLbl ::= report-active-monitor-not-
present.&objectClassLabel

report-beacon-state             FUNCTION   ::=
{
        ARGUMENTS               tnt-value,
        RESULTS                 result-status,
        IDENTIFIED BY           report-beacon-stateLbl
}
report-beacon-stateLbl          ::=     report-beacon-state.&objectClassLabel

report-provided-service-class   FUNCTION   ::=
{
        RESULTS                 provided-service-class,
        IDENTIFIED BY           report-provided-service-classLbl
}
report-provided-service-classLbl    ::=     report-provided-service-
class.&objectClassLabel

provided-service-class          RESULT     ::=
{
        RESULT-TYPE             Priority,
        IDENTIFIED BY           provided-service-classLbl,
}

report-frame-control            FUNCTION   ::=
{
        RESULTS                 frame-control,
        IDENTIFIED BY           report-frame-controlLbl,
}
```

```
report-frame-controlLbl         ::=     report-frame-control.&objectClassLabel

frame-control                           RESULT      ::=
{
        RESULT-TYPE                     Frame-control,
        IDENTIFIED BY                   frame-controlLbl
}

report-destination-address              FUNCTION    ::=
{
        ARGUMENTS                       frame,
        RESULTS                         destination-address,
        IDENTIFIED BY                   report-destination-addresssLbl
}
report-destination-addressLbl ::=       report-destination-addresss.&objectClassLabel

destination-address                     RESULT      ::=
{
        RESULT-TYPE                     Address,
        IDENTIFIED BY                   destination-addressLbl
}

report-source-address                   FUNCTION    ::=
{
        ARGUMENTS                       frame,
        RESULTS                         source-address,
        IDENTIFIED BY                   report-source-addressLbl
}
report-source-addressLbl        ::=     report-source-adress.&objectClassLabel

source-address                          RESULT      ::=
{
        RESULT-TYPE                     Address,
        IDENTIFIED BY                   source-addressLbl
}
report-m-sdu                            FUNCTION    ::=
{
        ARGUMENTS                       frame,
        RESULTS                         m-sdu,
        IDENTIFIED BY                   report-m-sduLbl
}
report-m-sduLbl                 ::=     report-m-sdu.&objectClassLabel

m-sdu                                   RESULT      ::=
{
        RESULT-TYPE                     INFO,
        IDENTIFIED BY                   m-sduLbl
}
```

```
report-reception-status              FUNCTION   ::=
{
        ARGUMENTS                    input-frame,
        RESULTS                      reception-status,
        IDENTIFIED BY                report-reception-statusLbl
}
report-reception-statusLbl     ::=   report-reception-status.&objectClassLabel

reception-status                     RESULT     ::=
{
        RESULT-TYPE                  Bit,
        IDENTIFIED BY                reception-statusLbl
}

tcu                                  OBJECT-CLASS     ::=
{
        ATTRIBUTES
                                     {       state,
                                     }
        FUNCTIONS
                                     {
                                             insert,
                                             repeat,
                                             transmit,
                                             loopbacktest,
                                     }
        IDENTIFIED BY                tcuLbl
}
tcuLbl                         ::=   tcu.&objectClassLabel

state                                ATTRIBUTE ::=
{
        ATTRIBUTE-TYPE               State,
        IDENTIFIED BY                stateLbl
}
stateLbl                       ::=   state.&objectClasslabel

State                          ::=   ENUMERATED
                                     {
                                             repeat(0),
                                             transmit(1)
                                     }

insert                               FUNCTION   ::=
{
        RESULTS                      output,
        IDENTIFIED BY                insertLbl
```

```
}
insertLbl                 ::=    insert.&objectClassLabel

output                           RESULT     ::=
{
        RESULT-TYPE              Bit,
        IDENTIFIED BY            outputLbl
}
outputLbl                 ::=    output.&objectClassLabel


remove                                FUNCTION   ::=
{
        RESULTS                  output,
        IDENTIFIED BY            removeLbl
}
removeLbl                 ::=    remove.&objectClassLabel

repeat                           FUNCTION   ::=
{
        ARGUMENTS                input-frame,
        RESULTS                  output,
        IDENTIFIED BY            repeatLbl
}
```

5. CONCLUSION

Object-oriented methodology described in chapter 2 is a powerful modelling technique to design communication networks . It is applied for designing Token Ring Network . It can be equally applied for designing other communication networks like Token Bus and FDDI. The work here included the design upto the Medium Access control Layer. It can be further enhanced by including the full design of Logical Link Control Sublayer.

6. REFERENCES

1. Chappel David, "Abstract Syntax Notation(ASN.1)",Journal of Data and Computer Communications, spring 1989.

2.Ungaro C.B. " The Local Network handbook edition II", McGraw Hill publication, Data Communication book series .

3.George C. Sachet, " IBM's Token Ring Networking handbook", MC-Graw Hill Publication, 1993.

4. Comer Douglas E., " Computer Networks and Internets", Prentice Hall Publication .

5. Tanenbaum Andrew S., " Computer Networks Third Edition", Prentice Hall India Publication.

6. Stephen Saunders, " Building a Better Token Ring Network ", Data Communication, May 1994.

7. Wolfgang Guenther and Gerd Wackerbarth, "Object-Oriented Design of ISDN, Call processing Software", IEEE Communication Magazine, April 1993.

8. Booch G., " Object Oriented Analysis & Design", Benjamin Cummings Publishinh 1994.

9. Coad P. and Yourdon F., " Object Oriented Analysis", Yourdon Press, 1991.

10. Rambaugh J.,Blaha M.,Premerlani W., Eddy F. and Lorensen W., " Object Oriented Modeling and Design", Prentice Hall of India Private Ltd 1997.