***LAN system design patterns :***

***After object-oriented programming had been used for some years, design patterns emerged and were documented . These patterns codified and regularized general approaches to solving particular common programming problems. This codification further improved the general state of the art in programming because it made it easier for less experienced programmers to produce well engineered code, and led to the development of reusable libraries***

***Ambassador pattern***

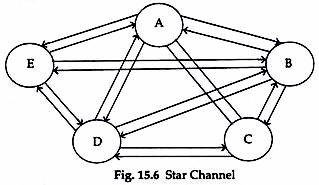
***The first pattern that we have observed is the ambassador pattern. Ambassador containers proxy communication to and from a main container. For example, a developer might pair an application that is speaking the memcache protocol with a twemproxy ambassador. The application believes that it is simply talking to a single memcache on localhost, but in reality twemproxy is sharding the requests across a distributed installation of multiple memcache nodes elsewhere in the cluster. This container pattern simplifies the programmer’s life in three ways: they only have to think and program in terms of their application connecting to a single server on localhost, they can test their application standalone by running a real memcache instance on their local machine instead of the ambassador, and they can reuse the twemproxy ambassador with other applications that might even be coded in different languages. Ambassadors are possible because containers on the same machine share the same localhost network interface.***

***Adapter pattern***

***The final single-node pattern we have observed is the adapter pattern. In contrast to the ambassador pattern, which presents an application with a simplified view of the outside world, adapters present the outside world with a simplified, homogenized view of an application. They do this by standardizing output and interfaces across multiple containers. A concrete example of the adapter pattern is adapters that ensure all containers in a system have the same monitoring interface. Applications today use a wide variety of methods to export their metrics (e.g. JMX, statsd, etc). But it is easier for a single monitoring tool to collect, aggregate, and present metrics from a heterogenous set of applications if all the applications present a consistent monitoring interface. Within Google, we have achieved this via code convention, but this is only possible if you build your software from scratch. The adapter pattern enables the heterogenous world of legacy and open-source applications to present a uniform interface without requiring modification of the original application. The main container can communicate with the adapter through localhost or a shared local volume. This is shown in Figure 3. Note that while some*** ***existing monitoring solutions are able to communicate with multiple types of back-ends, they use applicationspecific code in the monitoring system itself, which provides a less clean separation of concerns***

***Star Network:***

***Under star communication network all members of the group communicate with each other and exchange information. This network is a must for group communication or where teamwork is involved. This network channel of communication is open to all members of the group. The members communicate with each other without hesitation.***



***The effectiveness of the above networks of communication channels depend upon their users i.e. the managers at all levels, their subordinates and other members of the organization and above all the seriousness with which all these human resources make use of the facilities provided to them by the organization to accomplish its objectives***