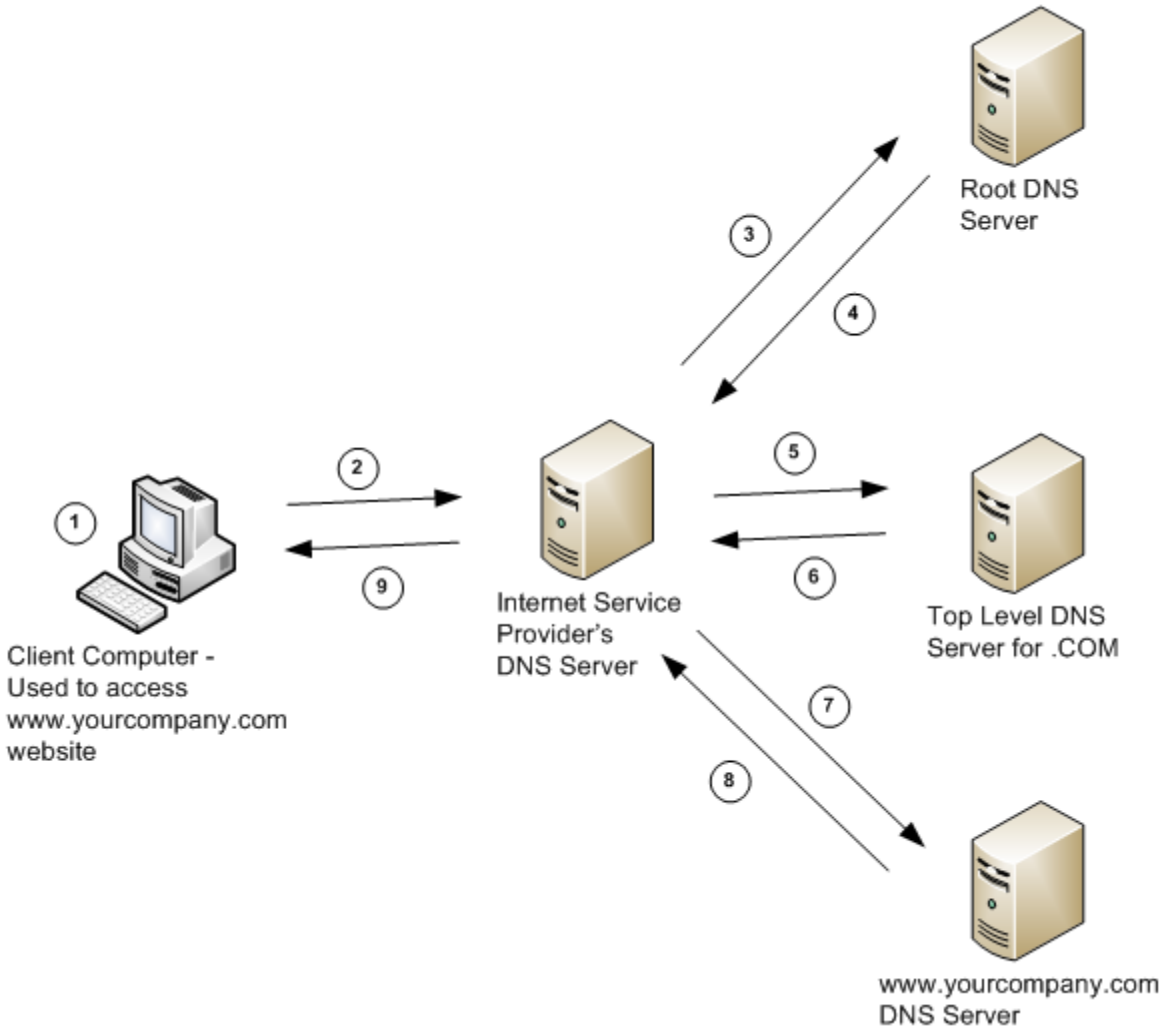
DNS

Short for ***D****omain* ***N****ame* ***S****ystem* (or ***S****ervice* or ***S****erver*), anInternet service that translates *domain names* into IP addresses. Because domain names are alphabetic, they're easier to remember. The Internet however, is really based on IP addresses. Every time you use a domain name, therefore, a DNS service must translate the name into the corresponding IP address. For example, the domain name *www.example.com* might translate to*198.105.232.4*.

The DNS system is, in fact, its own network. If one DNS server doesn't know how to translate a particular domain name, it asks another one, and so on, until the correct IP address is returned.



Load – Balancing

In this, the processes are distributed among nodes to equalize the load among all nodes. The scheduling algorithms that use this approach are known as Load Balancing or Load Leveling Algorithms. These algorithms are based on the intuition that for better resource utilization, it is desirable for the load in a distributed system to be balanced evenly. This a load balancing algorithm tries to balance the total system load by transparently transferring the workload from heavily loaded nodes to lightly loaded nodes in an attempt to ensure good overall performance relative to some specific metric of system performance.

We can have the following categories of load balancing algorithms:

- Static: Ignore the current state of the system. E.g. if a node is heavily loaded, it picks up a task randomly and transfers it to a random node. These algorithms are simpler to implement but performance may not be good.

- Dynamic: Use the current state information for load balancing. There is an overhead involved in collecting state information periodically; they perform better than static algorithms.

- Deterministic: Algorithms in this class use the processor and process characteristics to allocate processes to nodes.

- Probabilistic: Algorithms in this class use information regarding static attributes of the system such as number of nodes, processing capability, etc.

- Centralized: System state information is collected by a single node. This node makes all scheduling decisions.

- Distributed: Most desired approach. Each node is equally responsible for making scheduling decisions based on the local state and the state information received from other sites.

- Cooperative: A distributed dynamic scheduling algorithm. In these algorithms, the distributed entities cooperate with each other to make scheduling decisions. Therefore they are more complex and involve larger overhead than non-cooperative ones. But the stability of a cooperative algorithm is better than of a non-cooperative one.

- Non-Cooperative: A distributed dynamic scheduling algorithm. In these algorithms, individual entities act as autonomous entities and make scheduling decisions independently of the action of other entities.

C) Load – Sharing Approach

Several researchers believe that load balancing, with its implication of attempting to equalize workload on all the nodes of the system, is not an appropriate objective. This is because the overhead involved in gathering the state information to achieve this objective is normally very large, especially in distributed systems having a large number of nodes. In fact, for the proper utilization of resources of a distributed system, it is not required to balance the load on all the nodes. It is necessary and sufficient to prevent the nodes from being idle while some other nodes have more than two processes. This rectification is called the Dynamic Load Sharing instead of Dynamic Load Balancing.

The design of a load sharing algorithms require that proper decisions be made regarding load estimation policy, process transfer policy, state information exchange policy, priority assignment policy, and migration limiting policy. It is simpler to decide about most of these policies in case of load sharing, because load sharing algorithms do not attempt to balance the average workload of all the nodes of the system. Rather, they only attempt to ensure that no node is idle when a node is heavily loaded. The priority assignments policies and the migration limiting policies for load-sharing algorithms are the same as that of load-balancing algorithms.