2 REVIEW OF LITERATURE

We will study review in two parts. First part introduces about traditional protocols and routing method. Second part will focus on modern network.

Routing is the act of moving information across an inter-network from a source to a destination. Along the way, at least one intermediate node typically is encountered. Routing is often contrasted with bridging, which might seem to accomplish precisely the same thing to the casual observer. The primary difference between the two is that bridging occurs at Layer 2 (the link layer) of the OSI reference model, whereas routing occurs at Layer 3 (the network layer). This distinction provides routing and bridging with different information to use in the process of moving information from source to destination, so the two functions accomplish their tasks in different ways.

Figure 6: OSI Reference Network Model

Routing achieved commercial popularity as late as the mid-1980s. The primary reason for this time lag is that networks in the 1970s were simple and work in homogeneous environments. Recently only large-scale internetworking has become popular.

Routing involves two basic activities: determining optimal routing paths and transporting information groups (typically called packets) through an internet work. In the context of the routing process, the later of these is referred to as packet switching.
2.1 ROUTING COMPONENTS

Routing a packet through network path, packet need to be determined via routers and for this switching is done. So basically two things: Path determination and Switching is required for packet routing.

2.1.1 Path determination

Routing protocols use metrics to evaluate what path will be the best for a packet to travel. A metric is a standard of measurement, such as path bandwidth, that is used by routing algorithms to determine the optimal path to a destination. To aid the process of path determination, routing algorithms initialize and maintain the routing tables, which contain route information. Route information varies depending on the routing algorithm used.

Routing algorithms fill routing tables with a variety of information. Destination/next hop associations tell a router that a particular destination can be reached optimally by sending the packet to a particular router representing the "next hop" on the way to the final destination. When a router receives an incoming packet, it checks the destination address and attempts to associate this address with a next hop. Figure 7 depicts a sample destination/next hop routing table.

![Diagram of routing components](image)

Figure 7: Destination/Next Hop Associations Determine the Data's Optimal Path

Routing tables also can contain other information, such as data about the desirability of a path. Routers compare metrics to determine optimal routes, and these metrics
differ depending on the design of the routing algorithm used. A variety of common metrics will be introduced and described later in this chapter.

Routers communicate with one another and maintain their routing tables through the transmission of a variety of messages. The routing update message is one such message that generally consists of all or a portion of a routing table. By analyzing routing updates from all other routers, a router can build a detailed picture of network topology. A link-state advertisement, another example of a message sent between routers, informs other routers of the state of the sender's links. Link information also can be used to build a complete picture of network topology to enable routers to determine optimal routes to network destinations.

### 2.1.2 Switching

Switching algorithms is relatively simple; it is the same for most of the routing protocols. In most cases, a host determines that it must send a packet to another host. Having acquired a router's address by some means, the source host sends a packet addressed specifically to a router's physical (Media Access Control [MAC]-layer) address. As it examines the packet's destination protocol address, the router determines that it either knows or does not know how to forward the packet to the next hop. If the router does not know how to forward the packet, it typically drops the packet. If the router knows how to forward the packet, however, it changes the destination physical address to that of the next hop and transmits the packet.

The next hop may be the ultimate destination host. If not, the next hop is usually another router, which executes the same switching decision process. As the packet moves through the internet work, its physical address changes, but its protocol address remains constant.

The earlier discussion describes switching between a source and a destination end system. The International Organization for Standardization (ISO) has developed a hierarchical terminology that is useful in describing this process. Using this terminology, network devices without the capability to forward packets between sub-networks are called *end systems (ESs)*, whereas network devices with these capabilities are called *intermediate systems (ISs)*. ISs are further divided into those that can communicate within routing domains (*intra-domain ISs*) and between
routing domains (*inter-domain ISs*). A routing domain generally is considered a portion of an internet-work under common administrative authority that is regulated by a particular set of administrative guidelines. Routing domains are also called autonomous systems. With certain protocols, routing domains can be divided into routing areas, but intra-domain routing protocols are still used switching within and between areas.

![Diagram of routing process](image)

Figure 8: Numerous Routers May Come into Play during the Switching Process

### 2.2 DESIGN GOALS
Routing algorithms have following design goals:
- Optimality
- Simplicity and low overhead
- Robustness and stability
- Rapid convergence
- Flexibility
**Optimality** refers to the capability of the routing algorithm to select the best route, which depends on the metrics and metric weightings used to make the calculation. For example, one routing algorithm may use a number of hops and delays, but it may weigh delay more heavily in the calculation. Naturally, routing protocols must define their metric calculation algorithms strictly.

Routing algorithms also are designed to be as *simple* as possible. In other words, the routing algorithm must offer its functionality efficiently, with a minimum of software and utilization *overhead*. Efficiency is particularly important when the software implementing the routing algorithm must run on a computer with limited physical resources.

Routing algorithms must be *robust*, which means that they should perform correctly in the face of unusual or unforeseen circumstances, such as hardware failures, high load conditions, and incorrect implementations. Because routers are located at network junction points, they can cause considerable problems when they fail. The best routing algorithms are often those that have withstood the test of time and that have proven *stable* under a variety of network conditions.

In addition, routing algorithms must converge rapidly. *Convergence* is the process of agreement, by all routers, on optimal routes. When a network event causes routes to either go down or become available, routers distribute routing update messages that permeate networks, stimulating recalculation of optimal routes and eventually causing all routers to agree on these routes. Routing algorithms that converge slowly can cause routing loops or network outages.

### 2.3 ROUTING METRICS

Routing tables contain information used by switching software to select the best route. For selecting the best route we require, routing tables and specific nature of the information and routing algorithms which determine which route is preferable to others.

Routing algorithms have used many different metrics to determine the best route. Sophisticated routing algorithms can base route selection on multiple metrics, combining them in a single (hybrid) metric. All the following metrics have been used:

- **Path length** is the most common routing metric. Some routing protocols allow network administrators to assign arbitrary costs to each network link. In this case,
path length is the sum of the costs associated with each link traversed. Other routing protocols define hop count, a metric that specifies the number of passes through internetworking products, such as routers, that a packet must take en route from a source to a destination.

- **Reliability**, in the context of routing algorithms, refers to the dependability (usually described in terms of the bit-error rate) of each network link. Some network links might go down more often than others. After a network fails, certain network links might be repaired more easily or more quickly than other links. Any reliability factors can be taken into account in the assignment of the reliability ratings, which are arbitrary numeric values usually assigned to network links by network administrators.

- **Routing delay** refers to the length of time required to move a packet from source to destination through the internet-work. Delay depends on many factors, including the bandwidth of intermediate network links, the port queues at each router along the way, network congestion on all intermediate network links, and the physical distance to be traveled. Because delay is a conglomeration of several important variables, it is a common and useful metric.

- **Bandwidth** refers to the available traffic capacity of a link. All other things being equal, a 10-Mbps Ethernet link would be preferable to a 64-kbps leased line. Although bandwidth is a rating of the maximum attainable throughput on a link, routes through links with greater bandwidth do not necessarily provide better routes than routes through slower links. For example, if a faster link is busier, the actual time required to send a packet to the destination could be greater.

- **Load** refers to the degree to which a network resource, such as a router, is busy. Load can be calculated in a variety of ways, including CPU utilization and packets processed per second. Monitoring these parameters on a continual basis can be resource-intensive itself.

- **Communication cost** is another important metric, especially because some companies may not care about performance as much as they care about operating expenditures. Although line delay may be longer, they will send packets over their own lines rather than through the public lines that cost money for usage time.
2.4 ROUTING PROTOCOLS FOR ROUTING ALGORITHMS

The routing algorithm is stored in the router's memory. The routing algorithm is a major factor in the performance of our routing environment. The purpose of the routing algorithm is to make decisions for the router concerning the best paths for data. The router uses the routing algorithm to compute the path that would best serve to transport the data from the source to the destination. Note that you do not directly choose the algorithm that your router uses. Routing protocols have been created in response to the demand for dynamic routing table. A routing protocol is a combination of rules and procedures that let routers in the internet inform each other of changes. It allows routers to share whatever they know about the internet or their neighborhoods. The routing protocols also include procedures for combining information received from other routers.

![Figure 9: Classification of Routing Protocols](image)

Rather, the routing protocol we choose for our network determines which algorithm you will use. For example, whereas the routing protocol Routing Information Protocol (RIP) may use one type of routing algorithm to help the router move data, the routing protocol Open Shortest Path First (OSPF) uses another. The routing algorithm cannot be changed. The only way to change it is to change routing protocols. The overall performance of our network depends mainly on the routing algorithm, so you should research the algorithms each protocol uses before deciding which to implement on your network. There are two major categories of routing algorithms - distance vector
or link-state. Every routing protocol named "distance vector" uses the distance vector algorithm, and every link-state protocol uses the link-state algorithm.

One of the jobs of the routing protocol is to provide the information needed by the routing algorithm to compute its decisions. This is the point where many protocols differ. The information provided to the algorithm can be different from protocol to protocol. The routing protocol gathers information about networks and routers from the surrounding environment and stores the information within a routing table in the router's memory. The routing algorithm is run using the information within this table to calculate the best path from one network to another. Calculating the new values within the formula then generates a sum. The result of this calculation is used then to determine where to send information. For example, the table below illustrates a sample routing table for a fictitious routing environment. The information that is passed to the routing algorithm within the routing table is gathered by the routing protocol through a process known as a routing update. Through a series of updates, each router will tell the other what information it has. Eventually, an entire routing table will be built.

Table 1: A Simple Routing Table with Associated Metrics

<table>
<thead>
<tr>
<th>S. No</th>
<th>Router Link</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Router A to Router B</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Router B to Router C</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Router A to Router C</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>Router C to Router D</td>
<td>5</td>
</tr>
</tbody>
</table>

The sample routing algorithm states that the best path to any destination is the one that has the lowest metric value. A metric is a number that is used as a standard of measurement for the links of a network. Each link is assigned a metric to represent anything from monetary cost to use the line, to the amount of available bandwidth. When Router A is presented with a packet bound from Router C, the routing table shows two possible paths to choose from. The first choice is to send the packet from Router A directly over the link to Router C. The second option is to send the packet
from Router A to Router B and then on to Router C. The routing algorithm is used to
determine which option is best. Some routing protocols might only provide one metric
to the routing algorithm, whereas others might provide up to ten. On the other hand,
whereas two protocols might both send only one metric to the algorithm, the origin of
that metric might differ from protocol to protocol. One routing protocol might give an
algorithm the single metric of cost, but that cost could represent something different
than another protocol using the same metric. The algorithm in our example states that
the best path is the one with the lowest metric value. Therefore, by adding the metric
numbers associated with each possible link, we see that the route from Router A to
Router B to Router C has a metric value of 5, while the direct link to Router C has a
value of 6. The algorithm selects the A-B-C path and sends the information along.
A distance vector algorithm uses metrics known as costs in order to help determine
the best path to a destination. The path with the lowest total cost is chosen as the best
path. When a router utilizes a distance vector algorithm, different costs are gathered
by each router. These costs can be completely arbitrary numbers. Costs can also be
dynamically gathered values, such as the amount of delay experienced by routers
when sending packets over one link as opposed to another. All the costs are compiled
and placed within the router's routing table and then they are used by the algorithm to
calculate a best path for any given network scenario. Although there are many
resources that will offer complex mathematical representations of what distance
vector algorithms are and how they compute their decisions, the core concept remains
the same - by adding the metrics for every optional path on a network, we will come
up with at least one best path.

- **Distance Vector Algorithms**
The formula for this is as follows:

Best Path between two networks

\[ M(i,k) = \min[M(i,t) + M(t,k)] \]

This formula states that the best path between two networks (M(i,k)) can be found by finding the lowest (min) value of paths between all network points. Let's look again at the routing information in the table above. Plugging this information into the formula, we see that the route from A to B to C is still the best path:

\[ 5(A,C) = \min[2(A,B) + 3(B,C)] \]

Whereas the formula for the direct route A to C looks like this:

\[ 6(A,C) = \min[6(A,C)] \]

This example shows how distance vector algorithms use the information passed to them to make informed routing decisions. The algorithms used by routers and routing protocols are not configurable, nor can they be modified. Another major difference between distance vector algorithms and link state protocols is that when distance vector routing protocols update each other, all or part of the routing table (depending on the type of update) is sent from one router to another. By this process, each router is exposed to the information contained within the other router's tables, thus giving
each router a more complete view of the networking environment and enabling them to make better routing decisions. Examples of distance vector algorithms include RIP and BGP, two of the more popular protocols in use today. Other popular protocols such as OSPF are examples of protocols which use the link state routing algorithm. Distance vector algorithms are also known as Bellman-Ford routing algorithms and Ford-Fulkerson routing algorithms. In these algorithms, each router has a routing table which shows it the best route for any destination. A typical graph and routing table for router J is shown below.

![Graph for distance vector routing](image)

**Figure 11: Graph for distance vector routing**

**Table 2: Routing Table for Router J**

<table>
<thead>
<tr>
<th>Destination</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weight</strong></td>
<td>8</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>17</td>
<td>30</td>
<td>18</td>
<td>12</td>
<td>10</td>
<td>0</td>
<td>6</td>
<td>15</td>
</tr>
<tr>
<td><strong>Line</strong></td>
<td>A</td>
<td>A</td>
<td>I</td>
<td>H</td>
<td>I</td>
<td>I</td>
<td>H</td>
<td>H</td>
<td>I</td>
<td>N/A</td>
<td>K</td>
<td>K</td>
</tr>
</tbody>
</table>

The table shows that if router J wants to get packets to router D, it should send them to router H first. When the packets arrive at router H, the current router checks its own table and makes a decision how to send the packets to D. In distance vector algorithms, each router has to follow the following steps:

1. It counts the weight of the links directly connected to it and saves the information to its table.
2. In a particular period of time, the router sends its table to its neighbor routers (not to all routers) and receives the routing table of each of its neighbors.

3. Based on the information the router receives from its neighbors' routing tables, it updates its own.

Let's consider one more example (the figure represented below).

![Figure 12: An Simplified Diagram for the Distance Vector Routing](image)

The cost of each link is set to 1. Thus, the least cost path is simply the path with the fewer hops. The table below represents distance of each node from other nodes.

Table 3: An initial table for the figure 11

<table>
<thead>
<tr>
<th>Information stored at node</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$\infty$</td>
<td>1</td>
<td>1</td>
<td>$\infty$</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>D</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>1</td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
<td>$\infty$</td>
<td>$\infty$</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>1</td>
<td>$\infty$</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Initially, each node sets a cost of 1 to its directly connected neighbors and infinity to all the other nodes. Below is shown the initial routing table at node A.
Table 4: Initial routing table at node A

<table>
<thead>
<tr>
<th>Destination</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>1</td>
<td>1</td>
<td>∞</td>
<td>1</td>
<td>1</td>
<td>∞</td>
</tr>
<tr>
<td>Next Hop</td>
<td>B</td>
<td>C</td>
<td>-</td>
<td>E</td>
<td>F</td>
<td>-</td>
</tr>
</tbody>
</table>

During the next step, every node sends a message to its directly connected neighbors. That message contains the node's personal list of distances. Node F, for example, tells node A that it can reach node G at cost of 1; node A also knows that it can reach F at a cost of 1, so it adds these costs to get the cost of reaching G by means of F. Because 2 is less than the current cost of infinity, node A records that it can reach G at a cost of 2 by going through F. Node A learns from C that node B can be reached from C at a cost of 1, so it concludes that the cost of reaching B via C is 2. Because this is worse than the current cost of reaching B, which is 1, the new information is ignored. The final routing table at node A is shown below:

Table 5: Final routing table of node A

<table>
<thead>
<tr>
<th>Destination</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Next Hop</td>
<td>B</td>
<td>C</td>
<td>C</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

The process of getting consistent routing information to all the nodes is called convergence. The final set of costs from each node to all other nodes is shown in the table below:
Table 6: Final Set of Costs From Each Node to All Other Nodes

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

set to 1. Thus, the least cost path is simply the path with the fewer hops. One of the problems with distance vector algorithms is called "count to infinity." Let's examine the following problem with an example:

Consider a network with a graph as shown below. There is only one link between D and the other parts of the network.

![Figure 13: Example Based on the Same algorithm for Describing the Problem](image)

Information stored at node

\[
d[A][A] = 0 \quad d[A][B] = 1 \quad d[A][C] = 2 \quad d[A][D] = 3
\]

Now the C to D link crashes so cost \([C][D] = \infty\). C used to forward any packets with address D directly on the CD link, but now link is down, so C has to recomputed its distance vector (and make a new choice of how to forward packets to D) - similarly D has to update its vector. After updating their vectors at C and D, we have
Table 7: Initial Table for figure 13

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 8: After Updating Vectors at C and D

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

C views B as the best route to D, with cost 1 + 2, so C sends new vector to B. B learns that its former choice for sending to D via C now has higher cost, so B should recomputed its vector.

Table 9: B's Recomputed Table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

View of B is that routing to D can either go via A or C with equal cost - B sends updated vector. Both A and C get updated vector from B and learn that their preferred route to D now has higher cost, so they recomputed their own vectors.
Table 10: Updated Vector for A and C

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

Then A and C send their vectors, B has to update its vector again, sending another round to A and C, obtaining.

Table 11: B’s Upgraded Table

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>C</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>0</td>
</tr>
</tbody>
</table>

Notice that the routing table is very slowly converging to the fact that

\[ d[X \rightarrow D] = \infty \quad \text{where} \quad X = A \quad X = B \quad X = C \]

This process loops until all nodes find out that the weight of link to D is infinity. In this way, experts say that distance vector algorithms have a slow convergence rate. In conclusion, distance vector algorithm is not robust. One way to solve this problem is for routers to send information only to the neighbors that are not exclusive links to the destination. For example, in this case, B should not send any information to C about D, because C is the only way to D.

- Link-State Algorithms

Distance vector algorithms and link-state algorithms both favor the path with the lowest cost. However, link-state protocols work in more localized manner. Whereas a router running a distance vector algorithm will compute the end-to-end path for any given packet, a link-state protocol will compute that path as it relates to the most immediate link. That is, where a distance vector algorithm will compute the lowest
metric between Network A and Network C, a link-state protocol will compute it as two distinct paths, A to B and B to C. This process is very efficient for larger environments. Link-state algorithms enable routers to focus on their own links and interfaces. Any one router on a network will only have direct knowledge of the routers and networks that are directly connected to it (or, the state of its own links). In larger environments, this means that the router will use less processing power to compute complicated paths. The router simply needs to know which one of its direct interfaces will get the information where it needs to go the quickest. The next router in line will repeat the process until the information reaches its destination. Another advantage to such localized routing processes is that protocols can maintain smaller routing tables. Because a link-state protocol only maintains routing information for its direct interfaces, the routing table contains much less information than that of a distance vector protocol that might have information for multiple routers. Like distance vector protocols, link-state protocols require updates to share information with each other. These routing updates, known as Link State Advertisements (LSAs), occur when the state of a router's links changes. When a particular link becomes unavailable (changes state), the router sends an update through the environment alerting all the routers with which it is directly linked.

In Link-State Algorithms, every router has to follow these steps:

1. Identify the routers that are physically connected to them and get their IP addresses
   When a router starts working, it first sends a "HELLO" packet over network. Each router that receives this packet replies with a message that contains its IP address.
2. Routers measure the delay time (or any other important parameters of the network, such as average traffic) for neighbor routers. In order to do that, routers send echo packets over the network. Every router that receives these packets replies with an echo reply packet. By dividing round trip time by 2, routers can count the delay time. The delay time includes both transmission and processing times - the time it takes the packets to reach the destination and the time it takes the receiver to process it and reply.
3. Broadcast its information over the network for other routers and receive the other routers' information. In this step, all routers share their knowledge and broadcast their information to each other. In this way, every router can know the structure and status of the network.

4. Routers use an appropriate algorithm to identify the best route between two nodes of the network. In this step, routers choose the best route to every node. They do this using an algorithm, such as the Dijkstra shortest path algorithm. In this algorithm, a router, based on information that has been collected from other routers, builds a graph of the network. This graph shows the location of routers in the network and their links to each other. Every link is labeled with a number called the weight or cost. This number is a function of delay time, average traffic, and sometimes simply the number of hops between nodes. For example, if there are two links between a node and a destination, the router chooses the link with the lowest weight.

**Dijkstra algorithm**

The Dijkstra algorithm goes through the following steps:

1. The router builds a graph of the network. Then it identifies source and destination nodes, for example R1 and R2. The router builds then a matrix, called the "adjacency matrix." In the adjacent matrix, a coordinate indicates weight. [i, j], for example, is the weight of a link between nodes Ri and Rj. If there is no direct link between Ri and Rj, this weight is identified as "infinity."

2. The router then builds a status record for each node on the network. The record contains the following fields:
- Predecessor field - shows the previous node.
- Length field - shows the sum of the weights from the source to that node.
- Label field - shows the status of node; each node have one status mode: "permanent" or "tentative."

3. In the next step, the router initializes the parameters of the status record (for all nodes) and sets their label to "tentative" and their length to "infinity".
4. During this step, the router sets a T-node. If R1 is to be the source T-node, for example, the router changes R1's label to "permanent." Once a label is changed to "permanent," it never changes again.
5. The router updates the status record for all tentative nodes that are directly linked to the source T-node.
6. The router goes over all of the tentative nodes and chooses the one whose weight to R1 is lowest. That node is then the destination T-node.
7. If the new T-node is not R2 (the intended destination), the router goes back to step (5).
8. If this node is R2, the router extracts its previous node from the status record and does this until it arrives at R1. This list of nodes shows the best route from R1 to R2.

**Dijkstra algorithm example**

Let's find the best route between routers A and E. There are six possible routes between them (ABE, ACE, ABDE, ACDE, ABDCE, ACDBE), and it's obvious that ABDE is the best route because its weight is the lowest. But life is not always so easy, and there are some complicated cases in which we have to use algorithms to find the best route.

1. The source node (A) has been chosen as T-node, and so its label is permanent (permanent nodes are showed with filled circles and T-nodes with the -> symbol).
2. In this step, the status record of tentative nodes directly linked to T-node (B, C) has been changed. Also, because B has less weight, it has been chosen as T-node and its label has changed to permanent.

![Graph showing the network with nodes A, B, C, D, E and their weights](image)

3. Like in step 2, the status records of tentative nodes that have a direct link to T-node (D, E), have been changed. Because router D has less weight, it has been chosen as T-node and its label has changed to permanent.

![Graph showing the network with updated weights](image)

4. Because we do not have any tentative nodes, we just identify the next T-node. Because node E has the least weight, it has been chosen as T-node.

Now we have to identify the route. The previous node of E is node D, and the previous node of D is node B, and B's previous node is node A. So, we determine that the best route is ABDE. In this case, the total weight is 4 (1+2+1). This algorithm works well, but it is so complicated that it may take a long time for routers to process it. That would cause the efficiency of the network to fail. Another note we should make here is that if a router gives the wrong information to other routers, all routing decisions will be ineffective.

The next example shows how to find the best routes among all the nodes in a network. The example uses the Shortest Path Dijkstra algorithm. Consider the network shown below:
Let's use the Dijkstra's algorithm to find the routes that A will use to transmit to any of the notes on the network. The Dijkstra's routing algorithm is represented in the following table:

Table 12 : A Table for Finding Best Route Using Dijkstra Algorithm

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>A</td>
<td>2-A</td>
<td>3-A</td>
<td>5-A</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>Step 2</td>
<td>AB</td>
<td></td>
<td>3-A</td>
<td>5-A</td>
<td>7-B</td>
<td>9-B</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>Step 3</td>
<td>ABC</td>
<td></td>
<td>4-C</td>
<td>9-B</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
<td>∞</td>
</tr>
<tr>
<td>Step 4</td>
<td>ABCD</td>
<td>4-C</td>
<td>9-B</td>
<td>∞</td>
<td>11-D</td>
<td>∞</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>ABCDE</td>
<td>8-E</td>
<td>12-E</td>
<td>7-E</td>
<td>∞</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>ABCDEH</td>
<td>8-E</td>
<td>12-E</td>
<td></td>
<td>11-H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 7</td>
<td>ABCDEHFK</td>
<td>10-F</td>
<td>11-H</td>
<td></td>
<td>11-H</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 8</td>
<td>ABCDEHFG</td>
<td></td>
<td></td>
<td>11-H</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Step 9</td>
<td>ABCDEHFGI</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
This how the network looks after all the updates, showing the shortest route among the nodes:

![Diagram of network showing shortest route](image)

Figure 16: A Graph for Best Route Using Dijkstra Algorithm

### 2.4.1 Interior routing

Packet routing in the Internet is divided into two general groups: interior and exterior routing. Interior routing happens inside or interior to an independent network system. In TCP/IP terminology, these independent network systems are called autonomous systems. Within an autonomous system (AS), routing information is exchanged using an interior routing protocol chosen by the autonomous system's administration. The exterior routing protocols, on the other hand are used between the autonomous systems. Interior routing protocols determine the "best" route to each destination, and they distribute routing information among the systems on a network. There are several interior protocols:

- **Routing Information Protocol (RIP)** is the interior protocol most commonly used on UNIX systems. RIP uses distance vector algorithm that selects the route with the lowest "hop count" (metric) as the best route. The RIP hop count represents the number of gateways through which data must pass to reach its destination. RIP assumes that the best route is the one that uses the fewest gateways.

- **Hello** is a protocol that uses delay as the deciding factor when choosing the best route. Delay is the length of time it takes a datagram to make the round trip between its source and destination.
• **Intermediate System to Intermediate System (IS-IS)** is an interior routing protocol from the OSI protocol suite. It is a link-state protocol. It was the interior routing protocol used on the T1 NSFNET backbone.

• **Open Shortest Path First (OSPF)** is another link-state protocol developed for TCP/IP. It is suitable for very large networks and provides several advantages over RIP.

2.4.2 Exterior routing

Exterior routing occurs between autonomous systems, and is of concern to service providers and other large or complex networks.

• **Border Gateway Protocol (BGP)** is an inter autonomous system routing protocol. An autonomous system is a network or group of networks under a common administration and with common routing policies. BGP is used to exchange routing information for the Internet and is the protocol used between Internet service providers (ISP). Customer networks, such as universities and corporations, usually employ an Interior Gateway Protocol (IGP) such as RIP or OSPF for the exchange of routing information within their networks. Customers connect to ISPs, and ISPs use BGP to exchange customer and ISP routes.

• **External BGP** When BGP is used between autonomous systems (AS), the protocol is referred to as External BGP (EBGP).

• **Interior BGP** If a service provider is using BGP to exchange routes within an AS, then the protocol is referred to as Interior BGP (IBGP). Figure 17 illustrates this distinction.
BGP is a very robust and scalable routing protocol, as evidenced by the fact that BGP is the routing protocol employed on the Internet. At the time of this writing, the Internet BGP routing tables number more than 90,000 routes. To achieve scalability at this level, BGP uses many route parameters, called attributes, to define routing policies and maintain a stable routing environment. Routing policy is the main criterion for BGP. It tells what path should be chosen. Routing inside an AS is called intra-domain routing and routing between AS is called inter-domain routing.

2.5 PROBLEMS WITH RIP AND OSPF

RIP and OSPF are not good for Inter Domain Routing:

**RIP disabilities:**
1. Sometimes a route with the smallest hop count is not preferred route.
2. Due to instability, routers announce only the number of hop counts to the destination without actually defining the path that leads to that destination.
3. Advertising packets may be fooled if shortest path is actually calculated through the receiving router itself.

**OSPF disabilities:**
The internet is usually too big and so required to maintain a huge link state database. It would also take a long time for each router to calculate its routing table using the Dijkstra algorithm. The following table shows major differences between standard protocols.

Table 13: Comparison between RIP, OSPF, BGP

<table>
<thead>
<tr>
<th></th>
<th>RIP</th>
<th>OSPF</th>
<th>BGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource Consumption</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Version</td>
<td>V2</td>
<td>V2</td>
<td>V4</td>
</tr>
<tr>
<td>Request for Comments(RFC)</td>
<td>2453</td>
<td>2328</td>
<td>1771</td>
</tr>
<tr>
<td>IGP-EGP</td>
<td>IGP</td>
<td>IGP</td>
<td>EGP</td>
</tr>
<tr>
<td>Type</td>
<td>Distance vector</td>
<td>Link State</td>
<td>Path Vector</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Bellmen Ford</td>
<td>Dijkstra</td>
<td>Best Path Selectin</td>
</tr>
<tr>
<td>Convergence</td>
<td>Medium</td>
<td>Fast</td>
<td>Medium</td>
</tr>
<tr>
<td>OSI Level/Port</td>
<td>UDP/520</td>
<td>IP/89</td>
<td>TCP/179</td>
</tr>
<tr>
<td>Configuration Complexity</td>
<td>Easy</td>
<td>Medium</td>
<td>Hard</td>
</tr>
<tr>
<td>Scalable For</td>
<td>&lt;100 Routers</td>
<td>&lt;100 Routers</td>
<td>&gt;100 Routers</td>
</tr>
</tbody>
</table>

2.6 PROBLEMS WITH EXISTING ROUTING ALGORITHMS

- **Off-line routing**: with this routing, resulting routing solutions heavily dependent upon the accuracy of the traffic matrix and the calculations for large networks are generally very complex. Here highly dynamic networks poses problems like solution produces offline might be very quickly out of date if the network stage diverges significantly from traffic predictions.

- **Adaptive/dynamic routing** (fall between on and off line routing algorithm) here route calculations triggered directly by changes in network states such as:
link failure, new hosts being added and routing tables is updated automatically. Such types of algorithms do not take into account congestion.

- **Online routing:** Here route calculations need to be made as demand arises, so incurring delay in call set up, deciding on each route may itself be an NP-complete task depending upon the metrics used. Also, route calculations depend on extensive network state information which may be rapidly changing and needs to be accessible from throughput the network. For large network, information updates becomes very expensive to send and suffer from non-negligible delays.

- **Distributed routing:** (hop to hop routing) It is hard to apply complex heuristic in a distributed manner and that global properties are hard to maintain like (prevention of loops and adherence to end to end constraints such as delay).

- **Source routing:** First drawback is lacking of scalability. As hardware becomes larger, it becomes impractical to store and manipulate sufficient network state information in each node to obtain good routes. Second drawback is: source routing nodes will always be working with outdated information due to the propagation time of information updates in the network.

- **Hierarchical routing:** It make able to useful routing decision based on aggregation. So give imprecise network state information.

- **Multicast routing:** Due to need to manage the multicast group much of the difficulty in routing for multicast is with updating existing multicast trees when user join and leave rather than generating complete tree from scratch.

- **Multi authority routing:** Problem with this routing is to find a way of making use of restricted information to make adequate routing decisions when passing through domains controlled by several different authorities.

- **Quality of service routing:** There are number of difficulties need to be dealt with. QoS connections often depend on many different parameters. The number of parameters along with large numbers of different applications to very heterogeneous traffic types, making traffic prediction difficult. QoS connections are very sensitive to the network states; this sensitivity reduces
the usefulness of pre-determined routes. As the network state changes, the best route to take changes rapidly with it.

- **Making future reservation:** Problem occurs when call times holding times are not known in advance. If the holding time becomes known on connection time, the problem is considerably simplified since this information can be used to predict a clash with service reserved at some time in the future.

**Summary:**
Routing algorithms, while different in nature, all have the same basic objectives. While some algorithms are better than others. All routing protocols have their advantages and disadvantages. There are many companies that produce routers: *Cisco, Juniper, Bay, Nortel, 3Com, Cabletron*, etc. Each company's product is different in how it is configured, but most will interoperate so long as they share common physical and data link layer protocols. The following table is a summary of the essential characteristics of the major routing protocols and can be used as part of your evaluation criteria. It has the most common protocols and a number of points of evaluation.

Table 14: Routing Protocol Evaluation Criteria

<table>
<thead>
<tr>
<th>Protocol</th>
<th>RIPv1</th>
<th>RIPv2</th>
<th>OSPF</th>
<th>EIGRP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Distance-vector</td>
<td>Distance-vector</td>
<td>Link-state</td>
<td>Distance-vector</td>
</tr>
<tr>
<td>Convergence Time</td>
<td>Slow</td>
<td>Slow</td>
<td>Fast</td>
<td>Fast</td>
</tr>
<tr>
<td>VLSM</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Bandwidth Consumption</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Resource Consumption</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>Multi-path</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
- **Convergence time**: Convergence time is the point at which all routers on your network know about all current routes for the network. When a router is added or removed from a network, a certain amount of time — convergence time — must pass before this change is propagated to all routers on the network.

- **Variable length subnet masks (VLSM)**: This term refers to whether all routers on the network are required to use the same subnet mask. This requirement reduces your flexibility in assigning IP address network IDs to the network segments on your network.

- **Bandwidth consumption**: This term refers to the amount of necessary network bandwidth to maintain and distribute routing table information on the network. To share and distribute routing table information, all routing protocols need to send an amount of data over the network, and some send more than others.

- **Resource consumption**: In calculating and maintain routing table information on a router, a certain amount of processing power and memory is used.

- **Multi-path support**: When routes are discovered on the network that have loops in their paths, some segments have two possible routes, which represent multiple paths. Some routing protocols have support for multiple paths, by storing alternative paths in their routing information.

- **Scales well**: Some routing protocols operate well on small networks, but as the number of routers increases on the network, the routing protocol does not function as well. Routing protocols that can be used on small to very large networks scale well in size.

- **Proprietary**: The routing protocol based on open standards or a proprietary protocol owned by one company can affect the level of support and the speed of changes.
2.7 MODERN NETWORKS

This part focused on Content Delivery Networks which are modern networks. Modern networks have evolved to overcome the inherent limitations of the Internet in terms of user perceived Quality of Service (QoS) when accessing Web content. A modern network replicates content from the origin server to cache servers, scattered over the globe, in order to deliver content to end-users in a reliable and timely manner from nearby optimal surrogates. Content distribution on the Internet has received considerable research attention. It combines development of high-end computing technologies with high-performance networking infrastructure and distributed replica management techniques. Therefore, my aim is here to explore the uniqueness, weaknesses, opportunities, and future directions in this field. Generally we study the traditional networks in terms of their infrastructure, request-routing mechanisms, content replication techniques, load balancing, and cache management. Here we will be more concerned for request-routing techniques.

2.7.1 Introduction

With the proliferation of the Internet, popular Web services often suffer congestion and bottlenecks due to large demands made on their services. Such a scenario may cause unmanageable levels of traffic flow, resulting in many requests being lost. Replicating the same content or services over several mirrored Web servers strategically placed at various locations is a method commonly used by service providers to improve performance and scalability. The user is redirected to the nearest server and this approach helps to reduce network impact on the response time of the user requests.

Modern Networks provide services that improve network performance by maximizing bandwidth, improving accessibility and maintaining correctness through content replication. They offer fast and reliable applications and services by distributing content to cache or edge servers located close to users. A modern network has some combination of content-delivery, request-routing, distribution and accounting infrastructure. The content-delivery infrastructure consists of a set of edge servers (also called surrogates) that deliver copies of content to end-users. The request-routing infrastructure is responsible to directing client request to
appropriate edge servers. It also interacts with the distribution infrastructure to keep an up-to-date view of the content stored in the network caches. The distribution infrastructure moves content from the origin server to the edge servers and ensures consistency of content in the caches. The accounting infrastructure maintains logs of client accesses and records the usage of the modern network servers. This information is used for traffic reporting and usage-based billing. In practice, modern networks typically host static content including images, video, media clips, advertisements, and other embedded objects for dynamic Web content. Typical customers of a modern network are media and Internet advertisement companies, data centers, Internet Service Providers (ISPs), online music retailers, mobile operators, consumer electronics manufacturers, and other carrier companies. Each of these customers wants to publish and deliver their content to the end-users on the Internet in a reliable and timely manner. A modern network focuses on building its network infrastructure to provide the following services and functionalities: storage and management of content; distribution of content among surrogates; cache management; delivery of static, dynamic and streaming content; backup and disaster recovery solutions; and monitoring, performance measurement and reporting.

2.7.2 Overview

Collaboration among distributed modern networks components can occur over nodes in both homogeneous and heterogeneous environments. Modern networks can take various forms and structures. They can be centralized, hierarchical infrastructure under certain administrative control, or completely decentralized systems. There can also be various forms of internetworking and control sharing among different modern networks entities. The typical functionality of a modern network includes:

- **Request redirection and content delivery services** to direct a request to the closest suitable surrogate server using mechanisms to bypass congestion, thus overcoming flash crowds or Slashdot effects.

- **Content outsourcing and distribution services** to replicate and/or cache content to distributed surrogate servers on behalf of the origin server.

- **Content negotiation services** to meet specific needs of each individual user
(or group of users).

- **Management services** to manage the network components, to handle accounting, and to monitor and report on content usage.

A modern network provides better performance through caching or replicating content over some mirrored Web servers (i.e. surrogate servers) strategically placed at various locations in order to deal with the sudden spike in Web content requests, which is often termed as **flash crowd** or **Slashdot effect**. The users are redirected to the surrogate server nearest to them. This approach helps to reduce network impact on the response time of user requests. In the context of modern networks, **content** refers to any digital data resources and it consists of two main parts: the **encoded media** and **metadata**. The encoded media includes static, dynamic and continuous media data (e.g. audio, video, documents, images and Web pages). Metadata is the content description that allows identification, discovery, and management of multimedia data, and also facilitates the interpretation of multimedia data. Content can be pre-recorded or retrieved from live sources; it can be persistent or transient data within the system.

The three key components of a modern network architecture are – content provider, modern network provider and end-users. A **content provider or customer** is one who delegates the URI name space of the Web objects to be distributed. The origin server of the content provider holds those objects. A **modern network provider** is a proprietary organization or company that provides infrastructure facilities to content providers in order to deliver content in a timely and reliable manner. **End-users or clients** are the entities who access content from the content provider’s website.

### 2.8 EVOLUTION OF MODERN NETWORK'S

Over the last decades, users have witnessed the growth and maturity of the Internet. As a consequence, there has been an enormous growth in network traffic, driven by rapid acceptance of broadband access, along with increases in system complexity and content richness. The over-evolving nature of the Internet brings new challenges in managing and delivering content to users. A sudden spike in Web content requests may cause heavy workload on particular Web server(s), and as a result a **hotspot** can be generated. Coping with such unexpected demand
causes significant strain on a Web server. Eventually the Web servers are totally overwhelmed with the sudden increase in traffic, and the Web site holding the content becomes temporarily unavailable.

Content providers view the Web as a vehicle to bring rich content to their users. A decrease in service quality, along with high access delays mainly caused by long download times, leaves the users in frustration. Companies earn significant financial incentives from Web-based e-business. Hence, they are concerned to improve the service quality experienced by the users while accessing their Web sites. As such, the past few years have seen an evolution of technologies that aim to improve content delivery and service provisioning over the Web. When used together, the infrastructures supporting these technologies form a new type of network, which is often referred to as content network.

Several content networks attempt to address the performance problem through using different mechanisms to improve the Quality of Service (QoS). One approach is to modify the traditional Web architecture by improving the Web server hardware adding a high-speed processor, more memory and disk space, or maybe even a multi-processor system. This approach is not flexible. Moreover, small enhancements are not possible and at some point, the complete server system might have to be replaced. Caching proxy deployment by an ISP can be beneficial for the narrow bandwidth users accessing the Internet. In order to improve performance and reduce bandwidth utilization, caching proxies are deployed close to the users.

A more scalable solution is the establishment of server farms. It is a type of content network that has been in widespread use for several years. A server farm is comprised of multiple Web servers, each of them sharing the burden of answering requests for the same Web site. It also makes use of a Layer 4-7 switch, Web switch or content switch that examines content request and dispatches them among the group of servers. A server farm can also be constructed with surrogates instead of a switch. This approach is more flexible and shows better scalability. Moreover, it provides the inherent benefit of fault tolerance. Deployment and growth of server farms progresses with the upgrade of network links that connects the Web sites to the Internet. Although server farms and caching through caching proxies are useful techniques
to address the Internet Web performance problem, they have limitations. In the first case, since servers are deployed near the origin server, they do little to improve the network performance due to network congestion. Caching proxies may be beneficial in this case. But they cache objects based on client demands. This may force the content providers with a popular content source to invest in large server farms, load balancing, and high bandwidth connections to keep up with the demand. To address these limitations, another type of content network has been deployed in late 1990s. This is termed as **Content Distribution Network or Content Delivery Network**, which is a system of computers networked together across the Internet to cooperate transparently for delivering content to end-users.

With the introduction of modern network, content providers started putting their Web sites on a modern network. Soon they realized its usefulness through receiving increased reliability and scalability without the need to maintain expensive infrastructure. Hence, several initiatives kicked off for developing infrastructure for modern networks. As a consequence, Akamai Technologies evolved out of an MIT research effort aimed at solving the flash crowd problem. Within a couple of years, several companies became specialists in providing fast and reliable delivery of content, and modern networks became a huge market for generating large revenues. The flash crowd events like the 9/11 incident in USA, resulted in serious caching problems for some site. This influenced the modern network providers to invest more in modern network infrastructure development, since modern networks provide desired level of protection to Web sites against flash crowds. First generation modern networks mostly focused on static or Dynamic Web documents. On the other hand, for second generation of modern networks the focus has shifted to Video-on-Demand (VoD), audio and video streaming which are still in research phase.

With the booming of the modern network business, several standardization activities also emerged since vendors started organizing themselves. The Internet Engineering Task Force (IETF) as a official body took several initiatives through releasing RFCs (Request For Comments)]. Other than IETF, several other organizations such as Broadband Services Forum (BSF), ICAP forum, Internet Streaming Media Alliance took initiatives to develop standards for delivering broadband content, streaming rich media content – video, audio,
and associated data – over the Internet. In the same breath, by 2002, large-scale ISPs started building their own modern network functionality, providing customized services. In 2004, more than 3000 companies were found to use modern networks, spending more than $20 million monthly. A market analysis shows that modern network providers have doubled their earnings from streaming media delivery in 2004 compared to 2003. In 2005, modern network revenue for both streaming video and Internet radio was estimated to grow at 40%. A recent marketing research shows that combined commercial market value for streaming audio, video, streaming audio and video advertising, download media and entertainment was estimated at between $385 million to $452 million in 2005. Considering this trend, the market was forecasted to reach $2 billion in four-year (2002-2006) total revenue in 2006, with music, sports, and entertainment subscription and download revenue for the leading content categories. However, the latest report from AccuStream iMedia Research reveals that since 2002, the modern network market has invested $1.65 billion to deliver streaming media (excluding storage, hosting, applications layering), and the commercial market value in 2006 would make up 36% of the $1.65 billion four-year total in media and entertainment, including content, streaming advertising, movie and music downloads and User Generated Video (UGV) distribution. A detailed report on modern network market opportunities, strategies, and forecasts for the period 2004-2009, in relation to streaming media delivery can be found in references.

2.9 INSIGHT INTO MODERN NETWORK’s

Figure 18 shows a typical content delivery environment where the replicated Web server clusters are located at the edge of the network to which the end-users are connected. A content provider (i.e. customer) can sign up with a modern network provider for service and have its content placed on the content servers. The content is replicated either on-demand when users request for it, or it can be replicated beforehand, by pushing the content to the surrogate servers. A user is served with the content from the nearby replicated Web server. Thus, the user ends up unknowingly communicating with a replicated modern network server.
close to it and retrieves files from that server. 
modern network providers ensure the fast delivery of any digital content. They host third-party content including static content (e.g. static HTML pages, images, documents, software patches), streaming media (e.g. audio, real time video), User Generated Videos (UGV), and varying content services (e.g. directory service, e-commerce service, file transfer service). The sources of content include large enterprises, Web service providers, media companies and news broadcasters. The end-users can interact with the modern network by specifying the content/service request through cell phone, smart phone/PDA, laptop and desktop. Figure 19 depicts the different content/services served by a modern network provider to end-users.

Figure 18: Architecture of Modern Network
Modern network providers charge their customers according to the content delivered (i.e. traffic) to the end-users by their surrogate servers. Modern networks support an accounting mechanism that collects and tracks client usage information related to request-routing, distribution and delivery. This mechanism gathers information in real time and collects it for each modern network component. This information can be used in modern networks for accounting, billing and maintenance purposes. The average cost of charging of modern network services is quite high, often out of reach for many small to medium enterprises (SME) or not-for-profit organizations. The most influencing factors affecting the price of modern network services include:

- bandwidth cost
- variation of traffic distribution
- size of content replicated over surrogate servers
- number of surrogate servers
Figure 20: Request–Routing in a modern network Environment

Figure 20 provides a high-level view of the request-routing in a modern network environment. The interaction flows are: (1) the client requests content from the content provider by specifying its URL in the Web browser. Client’s request is directed to its origin server; (2) when origin server receives a request, it makes a decision to provide only the basic content (e.g. index page of the Web site) that can be served from its origin server; (3) to serve the high bandwidth demanding and frequently asked content (e.g. embedded objects – fresh content, navigation bar, banner ads etc.), content provider’s origin server redirects client’s request to the modern network provider; (4) using the proprietary selection algorithm, the modern network provider selects the replica server which is ‘closest’ to the client, in order to serve the requested embedded objects; (5) selected replica server gets the embedded objects from the origin server, serves the client requests and caches it for subsequent request servicing.

2.10 REQUEST-ROUTING ALGORITHMS
The algorithms invoked by the request-routing mechanisms can be adaptive or non-adaptive. Adaptive algorithms consider the current system condition to select a cache server for content delivery. Current condition of the system is obtained by estimating some metrics like load on the replica servers or the congestion of selected network links. Non-adaptive request-routing algorithms use some heuristics for selecting a cache server rather than considering the current system
condition. A non-adaptive algorithm is easy to implement, while the former is more complex. Complexity of adaptive algorithms arises from their ability to change behavior to cope with an enduring situation. A non-adaptive algorithm works efficiently when the assumptions made by the heuristics are met. On the other hand, an adaptive algorithm demonstrates high system robustness in the face of events like flash crowds.

![Taxonomy of Request Routing Algorithm](image)

**Non-adaptive request-routing algorithms**: An example of the most common and simple non-adaptive request-routing algorithm is round-robin, which distributes all requests to the modern network cache servers and attempts to balance load among them. It is assumed that all the cache servers have similar processing capability and that any of them can serve any client request. Such simple algorithms are efficient for clusters, where all the replica servers are located at the same place. But the round-robin request-routing algorithm does not perform well for wide area distributed systems where the cache servers are located at distant places. In this case it does not consider the distance of the replica servers. Hence, client requests may be directed to more distant replica servers, which cause poor performance perceived by the users. Moreover, the aim of load balancing is not fully achieved since processing different request can involve significantly different computational costs.

In another non-adaptive request-routing algorithm, all replica servers are ranked according to the predicted load on them. Such prediction is done based on the number of requests each of the servers has served so far. This algorithm takes client-server distance into account and client requests are directed to the replica servers in such a way that load is balanced among them. The assumption here is that the replica server load and the client-server distance are the most influencing factors for the efficiency of request processing. Though it has been observed that deploying this
algorithm can perform well for request-routing, the client perceived performance may still be poor.

Several other interesting non-adaptive request-routing algorithms are implemented in the Cisco Distributed Director. One of these algorithms considers the percentage of client requests that each replica server receives. A server receiving more requests is assumed to be more powerful. Hence, client requests are directed to the more powerful servers to achieve better resource utilization. Another algorithm defines preference of one server over another in order to delegate the former to serve client requests. The Distributed Director also supports random request distribution to replica servers. Furthermore, some other non-adaptive algorithms can be found which considers the client’s geographic location to redirect requests to the nearby replica. But this algorithm suffers from the fact that client requests may be assigned to overloaded replica servers, which may degrade client perceived performance.

**Adaptive request-routing algorithms:** Globule uses an adaptive request-routing algorithm that selects the replica server closest to the clients in terms of network proximity. The metric estimation in Globule is based on path length which is updated periodically. The metric estimation service used in globule is passive, which does not introduce any additional traffic to the network. However, results in show that the distance metric estimation procedure is not very accurate.

Andrews et al. and Ardiaz et al. have proposed adaptive request-routing algorithms based on client-server latency. In this approach, either client access logs or passive server-side latency measurements are taken into account, and the algorithms decide to which replica server the client requests are to be sent. Hence, they redirect a client request to a replica which has recently reported the minimal latency to the client. These algorithms are efficient since they consider latency measurements. However, they require the maintenance of central database of measurements, which limits the scalability of systems on which these algorithms are deployed.

Cisco Distribute Director has implemented an adaptive request-routing algorithm. The request-routing algorithm deployed in this system takes into account a weighted combination of three metrics, namely –inter-AS distance, intra-AS distance, and end-to-end latency. Though this algorithm is flexible since it makes use of three
metrics, the deployment of an agent in each replica server for metric measurement makes it complex and costly. Moreover, the active latency measurement techniques used by this algorithm introduce additional traffic to the Internet. Furthermore, the isolation of Distributed Director component from the replica server makes it unable to probe the servers to obtain their load information.

Akamai uses a complex adaptive request-routing algorithm. It takes into consideration a number of metrics such as replica server load, the reliability of loads between the client and each of the replica servers, and the bandwidth that is currently available to a replica server.

2.11 REQUEST ROUTING MECHANISM

Request-routing mechanisms inform the client about the selection of replica server, generated by the request-routing algorithms. Request-routing mechanisms can be classified according to several criteria. In this section we classify them according to the variety of request processing. As shown in Figure 15, they can be classified as: Global Server Load Balancing (GSLB), DNS-based request-routing, HTTP redirection, URL rewriting, anycasting, and modern network peering.

![Figure 22: Taxonomy of request-routing mechanisms](image-url)
These routing schemes are stated in the following:

**Global Server Load Balancing (GSLB):** In this approach, service nodes (which serve content to end-users) consisting of a GSLB-enabled Web switch and a number of real Web servers are distributed in several locations around the world. Two new capabilities of the service nodes allow them to support global server load balancing. The first is *global awareness* and the second is *smart authoritative DNS*. In local server load balancing each service node is aware of the health and performance information of the Web servers directly attached to it. In GSLB, one service node is aware of the information in other service nodes and includes their virtual IP address in its list of servers. Hence, the Web switches making up each service node are globally aware and each knows the addresses of all the other service nodes. They also exchange performance information among the Web switches in GSLB configuration. To make use of such global awareness, the GSLB switches act as a smart authoritative DNS for certain domains. The advantage of GSLB is that since the service nodes are aware of each other, each GSLB switch can select the best surrogate server for any request. Thus this approach facilitates choosing servers not only from the pool of locally connected real servers, but also the remote service nodes. Another significant advantage of GSLB is that the network administrator can add GSLB capability to the network without adding any additional networking devices. A disadvantage of GSLB is the manual configuration of the service nodes to enable them with GSLB capability.

**DNS-based request-routing:** In this approach, the content distribution services rely on the modified DNS servers to perform the mapping between a surrogate server’s symbolic name and its numerical IP address. It is used for full-site content selection and delivery. In DNS-based request-routing, a domain name has multiple IP addresses associated to it. When an end-user’s content request comes, the DNS server of the service provider returns the IP addresses of servers holding the replica of the requested object. The client’s DNS resolver chooses a server among these. To decide, the resolver may issue probes to the servers and choose based on response times to these probes. It may also collect historical information from the clients based on previous access to these servers. Both full and partial-site modern network providers use DNS redirection. The performance and effectiveness of DNS-based request-routing has been examined in a number of recent studies. The advantage of this
approach is the transparency as the services are referred to by means of their DNS names, and not their IP addresses. DNS-based approach is extremely popular because of its simplicity and independence from any actual replicated service. Since it is incorporated to the name resolution service it can be used by any Internet. In addition, the ubiquity of DNS as a directory service provides advantages during request-routing. The main disadvantage of DNS-based request-routing is that, it increases network latency because of the increase in DNS lookup times. modern network administrators typically resolve this problem by splitting modern network DNS into two levels (low-level DNS and high-level DNS) for load distribution. Another disadvantage is that it does not take into account the IP address of the clients. The knowledge of Internet location of the client DNS server limits the ability of the request-routing system to determine a client’s proximity to the surrogate. Another limitation of this approach is that it is not scalable and it has limited control on client identification since a DNS query does not carry the addresses of the querying client. Most significantly, DNS cannot be relied upon to control all incoming requests due to caching of DNS data at both the ISP and client level. Indeed, it can have control over as little as 5% of requests in many instances. Furthermore, since clients do not access the actual domain names that serve their requests, it leads to the absence of any alternate server to fulfill client requests in case of failure of the target surrogate server.

**HTTP redirection:** The approach propagates information about replica server sets in HTTP headers. HTTP protocols allow a Web server to respond to a client request with a special message that tells the client to re-submit its request to another server. HTTP redirection can be used for both full-site and partial-site content selection and delivery. This mechanism can be used to build a special Web server, which accepts client requests, chooses replica servers for them and redirects clients to those servers. It requires changes to both Web servers and clients to process extra headers. Clients must also be modified to implement replica server selection. The main advantage of this approach is flexibility and simplicity. Another advantage is that replication can be managed at fine granularity, since individual web pages are considered as a granule. The most significant disadvantage of HTTP redirection is the lack of transparency. Moreover, the overhead perceived through this approach is significant since it introduces extra message round-trip into request processing as
well as over HTTP.

**URL rewriting or Navigation hyperlink:** Though most modern network systems use a DNS based routing scheme, some systems use the URL rewriting. It is mainly used for partial-site content selection and delivery where embedded objects are sent as a response to client requests. In this approach, the origin server redirects the clients to different surrogate servers by rewriting the dynamically generated pages’ URL links. For example, with a Web page containing an HTML file and some embedded objects, the Web server would modify references to embedded objects so that the client could fetch them from the best surrogate server. To automate this process, modern networks provide special scripts that transparently parse Web page content and replace embedded URLs. URL rewriting can be proactive or reactive. In the pro-active URL rewriting, the URLs for embedded objects of the main HTML page are formulated before the content is loaded in the origin server. In reactive approach involves rewriting the embedded URLs of a HTML page when the client request reaches the origin server. The main advantage of URL rewriting is that the clients are not bound to a single surrogate server, because the rewritten URLs contain DNS names that point to a group of surrogate servers. Also finer level of granularity can be achieved through this approach since embedded objects can be considered as granule. The disadvantages through this approach are the delay for URL-parsing and the possible bottleneck introduced by an in-path element. Another disadvantage is that content with modified reference to nearby surrogate server rather than to the origin server is non-cacheable.

**Anycasting:** The any casting approach can be divided into two: IP anycasting and Application-level any casting. IP anycasting, proposed by Partridge et al., assumes that the same IP address is assigned to a set of hosts and each IP router holds a path in its routing table to the host that is closest to this router. Thus, different IP routers have paths to different hosts with the same IP address. IP anycasting can be suitable for request-routing and service location. It targets network-wide replication of the servers over potentially heterogeneous platforms. A disadvantage of IP any casting is some part of IP address space is to be allocated for any cast address. Fei et al. Proposed an application level anycasting mechanism where the service consists of a set of **anycast resolvers**, which performs the **anycast domain names** to IP address mapping. Clients interact with the anycast
resolvers by generating an anycast query. The resolver processes the query and replies with an anycast response. A Metric database, associated with each anycast resolver contains performance data about replica servers. The performance is estimated based on the load and the request processing capability of the servers. The overhead of the performance measurement is kept at a manageable level. The performance data can be used in the selection of a server from a group, based on user-specified performance criteria. An advantage of application level anycasting is that better flexibility can be achieved through this approach. One disadvantage of this approach is, deploying the anycasting mechanism for request-routing requires changes to the servers as well as to the clients. Hence, it may lead to increased cost considering possibly large number of servers and clients.

**Modern network peering:** Peer-to-peer content networks are formed by symmetrical connections between host computers. Peered modern networks deliver content on each other’s behalf. Thus, a modern network could expand its reach to a larger client population by using partnered modern network servers and their nearby forward proxies. A content provider usually has contracts with only one modern network and each modern network contacts other peer modern networks on the content provider’s behalf. Peering modern networks are more fault-tolerant as the necessary information retrieval network can be developed on the peering members themselves instead of relying on a dedicated infrastructure like traditional networks. To locate content in modern network peering, a centralized directory model, Distributed Hash Table (DHT), flooded request model or document routing model can be used. In a centralized directory model, peers contact a centralized directory where all the peers publish content that they want to share with others. When the directory receives a request it responses with the information of the peer that holds the requested content. When more than one peer matches the request, the best peer is selected based on metrics such as network proximity, highest bandwidth, least congestion and highest capacity. On receiving the response from the directory, the requesting peer contacts the peer that it has been referred to for content retrieval. The drawback of this approach is that, the centralized directory is subject to a single point of failure. Moreover, the scalability of a system based on a centralized directory is limited to the capacity of the directory. Archi, WAIS are the examples of centralized directory systems.
for retrieving FTP files located on various systems. In systems using DHTs, peers are indexed through hashing keys within a distributed system. Then a peer holding the desired content can be found through applying complex queries. The advantage of this approach is the ability to perform load balancing by offloading excess loads to the less-loaded peers. In the flooded request model, a request from a peer is broadcast to the peers directly connected to it. These peers in turn forwards the message to other peers directly connected to them. This process continues until the request is answered or some broadcast limit is reached. The drawback of this approach is that it generates unnecessary network traffic and hence, it requires enormous bandwidth. Thus, it suffers from scalability problem and it limits the size of the network. Gnutella is the example of a system using the flooded request model. In document routing model an authoritative peer is asked for referral to get the requested content. Each peer in the model is helpful, though they partially complete the referral information. Each referral moves to the requester closer to a peer that can satisfy the query. The main advantage of this approach is that it can complete a comprehensive search within a bounded number of steps. Moreover, it shows good performance and it is scalable enough to grow significantly large.

2.12 ADVANTAGES AND DISADVANTAGES OF USING A MODERN NETWORK

A growing trend in the Internet community is the usage of a modern network to provide data across nodes in the most effective manner possible. For those considering implementing such a system, there are many advantages to doing so. However, there are also several disadvantages that must be also be considered prior to implementation.

Advantages:
The advantages include:

- **Increase in concurrent users**: Strategically placing the servers in a modern network can result in high network backbone capacity, which equates to a significant increase in the number of users accessing the network at a given time. For example, where there is a 100 GB/s network backbone with 2 tb/s capacity, only 100 GB/s can be delivered. However, with a modern network, 10 servers will be available at 10 strategic locations and can then provide a total capacity of 10 x 100 GB/s.
• **Decrease server load:** As a result, the strategic placement can decrease the server load on interconnects backbones and public and private peers, which frees up overall capacity and decreases delivery costs. Essentially, the content is spread out across several servers, as opposed to offloading them onto one large server.

• **Faster content delivery:** Since modern networks place servers as close to a group of users as possible, latency and packet loss are minimized due to a shorter distance traveled. Theoretically, the closer the content is to the user, the faster the delivery. Therefore, users will experience less jitter when streaming, fewer network spikes, and an overall improved streaming quality. Due to the reliability, operators can deliver high quality content with a high level of service, low network server loads, and thus, lower costs. Additionally, many modern network providers offer TCP acceleration technology which boosts performance, thus improving user experience. Since modern networks decrease latency, the acceleration working in conjunction with an already high-performing network results in explosive content.

• **100 percent availability:** Due to the distribution of assets across many regions, modern networks have automatic server availability sensing mechanisms with instant user redirection. As a result, modern network websites experience 100 percent availability, even during massive power outages, hardware issues or network problems.

• **Control of asset delivery:** Another beneficial feature of modern network technology is that more control of asset delivery and network load is awarded. Operators have the ability to deliver real-time load statistics, optimize capacity per customer, display active regions, indicate which assets are popular, and report viewing details to their customers. These details are extremely important, since usage logs are deactivated once the server source has been added to the modern network.

**Disadvantages**

Unfortunately, there are several disadvantages to modern network’s, which include:

• **Impractical for many organizations:** Due to the inherent nature of the Internet being global, websites receives hits from across the world. Therefore,
it is impractical for most organizations to maintain duplicate servers around strategically positioned around the world.

- **Cost:** As a result, organizations must rely on support from third-party modern network vendors. Therefore, another one of the greatest limitations of a modern network are the fees associated with the service. Many of the larger modern network have high setup fees and other hidden fees. The high fee structure could potentially be to keep away smaller clients, focusing only on large business entities. In many instances, the pricing structure is hidden, not readily available or can be difficult to understand all the moving parts – one of which is the limitation of high cost per GB or storage and data transfer. Therefore, it is critical to understand every aspect of the terms and conditions prior to entering into a contract.

- **Support:** Since most organizations utilize third-party vendors to maintain the modern network, there is always the question of support availability. If a major issue arises, will the operator be able to fix it in a timely manner and prevent the same problem from occurring again?

- **Maintenance:** Similarly, the modern network operator must also effectively maintain each server with the proper updates and patches without disrupting the client’s content network. Placing a company’s entire corporate network into the hands of an operator is a major step. Therefore, all factors must be considered and backup plans implemented prior to setup and usage. This also includes timely maintenance and application of updates.

- **Verification of the best locations:** Additionally, organizations must research the location of the servers offered by each modern network and find those that best fit their customer’s locations. It is pointless to utilize a modern network that is a significant distance from users, which will result in potential service disruptions, jittering streaming of video, downtime, low latency and thus low performance. Clients must completely verify the exact locations of all servers and determine if the modern network will be beneficial to its services and client base. With the high number of products available on the Internet and the amount of money spent to purchase those items while checking information, reading, writing and conducting many other activities on the
Internet, it is critical to have a system that delivers performance and reliability. Modern networks maintain those tasks and do much more. Although there are several disadvantages to utilize a modern network, there are even more advantages, which makes it one of the best online computing options available.

- **Impractical for critical applications:** Second generation of modern networks the focus has shifted to Video-on-Demand (VoD), audio and video streaming. Modern network routing methods are not based on applications awareness. They are still in research phase and have not reached to the market yet.

**Summary:**
Recently, the modern network industry is getting consolidated as a result of acquisitions. Along with the proliferation, formation, and consolidation of the modern network landscape, new forms of Internet content and services are coming into the picture. Consequently, content distribution, caching and replication techniques are gaining more attention in order to meet up the new technical and infrastructure requirements of the next generation modern networks. This may lead to new issues in the design and architecture of modern networks. Present trends in content networking domain indicate that better understanding and interpretation of the essential concepts in this area is necessary. This part will not only serve as a tool to understand this complex area, but also help to map the future research efforts in content networking.