160118737084
Priya chillaru

CN LAB Record

AIM: To capture and analyze IP packets by executing trace route

DESCRIPTION: The tracert command is a cmd prompt command that is used to show several details about the path that a packet takes from the computer or device you're on to whatever destination you specify.

Tracert command options

Item	Description
-d	This option prevents tracert from resolving IP address to hostname, often resulting in much faster results.
-h Maxhops	This tracert option specifies the maximum number of hops in the search target
-w Timeout	you can specify the time in milliseconds, to allow each reply before timeout using this tracert option.
-4	This option forces tracert to use IPv4 only
-6	This option forces tracert to use IPV6 only
Target	This is the destination, either an IP address or host
/?	use the help switch with the tracert command to show detailed help about the commands ser option

COMMANDS:

1) tracert 192.168.1.1

2) tracert www.google.com

3) tracert -d www.yahoo.com

4) tracert -h 3 lifewire.com > z:\traceroute

OUTPUT:

1) Tracing route to 192.168.1.1 over a maximum of 30 hops

1 <1 ms <1 ms <1ms 192.168.1.254

2 <1 ms <1 ms <1ms 192.168.1.1

Trace complete

2) Tracing route to www.l.google.com [209.85.225.104]

over a maximum of 30 hops

1 <1ms <1ms <1ms 10.1.0.1

2 35 ms 19 ms 29ms 98.245.140.1

3 11 ms 27ms 9ms ic-0-3.dnvr.comcast.net [68.85.105.201]

...

13 81 ms 76ms 75 ms 209.85.241.37

14 84 ms 91ms 87ms 209.85.248.102

15 76ms 112ms 76ms iy-in-f104.1e100.net [209.85.225.104]

Trace Complete

3) Tracing route to any-fp.wal.b.yahoo.com [209.191.122.70]

over a maximum of 30 hops

1 <1ms <1ms <1 10.1.0.1

2 29ms 23 ms 20ms 98.245.140.1

...

14 80 ms 88ms 85 ms 68.142.193.11

15 77ms 79ms 78ms 209.191.122.70

Trace complete

4. Tracing route to lifewire.com [157.101.66.114]

over a maximum of 3 hops:

1	<1ms	<1ms	<1ms	[illegible] [192.168.86.1]
2	1ms	1ms	<1ms	192.168.1.1
3	17ms	16ms	17ms	giant wals 64.71.222.1.giawarmm.net [64.71.222.1]

Trace complete

```
C:\Users\srija>tracert -d www.yahoo.com

Tracing route to new-fp-shed.wg1.b.yahoo.com [2406:2000:e4:1605::9000]
over a maximum of 30 hops:

  1    90 ms     2 ms     2 ms  2409:4070:2e12:93b4::3e
  2     *        *        *     Request timed out.
  3    72 ms    34 ms    46 ms  2405:200:393:eeee:20::226
  4   243 ms   200 ms    56 ms  2405:200:801:700::c0e
  5    57 ms    38 ms    46 ms  2405:200:801:700::c0d
  6    42 ms    56 ms    55 ms  2405:200:801:900::ce4
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9     *        *        *     Request timed out.
 10   181 ms   100 ms    97 ms  2406:2000:f015:7::1
 11   101 ms    88 ms    87 ms  2406:2000:e4:fe01::1
 12   131 ms    92 ms    87 ms  2406:2000:e4:f814::1
 13   223 ms    99 ms    88 ms  2406:2000:e4:e208::1
 14   216 ms   202 ms   100 ms  2406:2000:e4:1605::9000
```

```
C:\Users\srija>tracert -h 6 www.meridianoutpost.com

Tracing route to www.meridianoutpost.com [2606:4700:8dec:ce38:80de:36:7965:4b14]
over a maximum of 6 hops:

  1    94 ms     5 ms     2 ms  2409:4070:2e12:93b4::3e
  2     *        *        *     Request timed out.
  3    65 ms    62 ms    43 ms  2405:200:393:eeee:20::226
  4    62 ms    38 ms    47 ms  2405:200:801:700::c0c
  5    71 ms    53 ms    49 ms  2405:200:801:700::c0f
  6    88 ms    62 ms    62 ms  2405:200:801:900::ce2

Trace complete.
```

```
C:\Users\srija>tracert www.google.com

Tracing route to www.google.com [2404:6800:4007:805::2004]
over a maximum of 30 hops:

  1   102 ms     2 ms     2 ms  2409:4070:2e12:93b4::3e
  2     *        *        *     Request timed out.
  3    74 ms   136 ms    35 ms  2405:200:393:eeee:20::226
  4    79 ms    93 ms    43 ms  2405:200:801:700::c0e
  5    52 ms    37 ms    48 ms  2405:200:801:700::c0d
  6    81 ms    67 ms    53 ms  2405:200:801:900::1d
  7     *        *        *     Request timed out.
  8    63 ms    47 ms    58 ms  2405:200:80c:760::5
  9     *        *        *     Request timed out.
 10    83 ms    65 ms    54 ms  2001:4860:1:1:0:da1c:0:16
 11    71 ms    57 ms     *     2001:4860:0:e00::1
 12   217 ms     *       87 ms  2001:4860:0:1::448f
 13    83 ms    52 ms    58 ms  maa05s13-in-x04.1e100.net [2404:6800:4007:805::2004]

Trace complete.
```

```
racing route to syd15s04-in-f3.1e100.net [216.58.196.131]
ver a maximum of 30 hops:

 1   111 ms     1 ms     1 ms  192.168.43.1
 2     *        *        *     Request timed out.
 3    51 ms    38 ms    49 ms  10.72.212.81
 4    73 ms    35 ms    43 ms  172.25.124.208
 5   152 ms    65 ms    48 ms  172.25.124.207
 6    49 ms    37 ms    29 ms  172.26.100.68
 7    75 ms    44 ms    63 ms  172.26.100.82
 8    77 ms    66 ms    46 ms  192.168.59.112
 9   100 ms    45 ms    33 ms  192.168.59.113
10    81 ms    69 ms    52 ms  172.31.2.67
11     *       60 ms    58 ms  72.14.217.252
12    75 ms    58 ms    62 ms  108.170.253.121
13   320 ms   203 ms    97 ms  74.125.251.157
14   222 ms   304 ms   304 ms  108.170.234.71
15   672 ms   611 ms   306 ms  142.250.62.191
16   380 ms   189 ms   219 ms  108.170.247.81
17   427 ms   307 ms   305 ms  209.85.142.137
18   506 ms   408 ms   305 ms  syd15s04-in-f3.1e100.net [216.58.196.131]
```

```
C:\Users\srija>tracert -R www.yahoo.com

Tracing route to new-fp-shed.wg1.b.yahoo.com [2406:2000:e4:1605::9001]
over a maximum of 30 hops:

  1    95 ms     2 ms     2 ms  2409:4070:2e12:93b4::3e
  2     *        *        *     Request timed out.
  3    55 ms    48 ms    48 ms  2405:200:393:eeee:20::226
  4    48 ms    59 ms    48 ms  2405:200:801:700::c0c
  5    59 ms    38 ms   106 ms  2405:200:801:700::c0f
  6    64 ms    60 ms    66 ms  2405:200:801:900::ce4
  7     *        *        *     Request timed out.
  8     *        *        *     Request timed out.
  9     *        *        *     Request timed out.
 10   175 ms   136 ms   188 ms  ae-4.msr1.sg3.yahoo.com [2406:2000:f015:7::1]
 11   139 ms   174 ms   401 ms  ae-2.clr2-a-gdc.sg3.yahoo.com [2406:2000:e4:fe01::1]
 12     *        *        *     Request timed out.
 13     *        *        *     Request timed out.
 14     *        *        *     Request timed out.
 15     *        *        *     Request timed out.
 16     *        *        *     Request timed out.
 17     *        *        *     Request timed out.
 18     *        *        *     Request timed out.
 19     *        *        *     Request timed out.
 20     *        *        *     Request timed out.
 21     *        *        *     Request timed out.
 22     *        *        *     Request timed out.
 23     *        *        *     Request timed out.
 24     *        *        *     Request timed out.
 25     *        *        *     Request timed out.
 26     *        *        *     Request timed out.
 27     *        *        *     Request timed out.
 28     *        *        *     Request timed out.
 29     *        *        *     Request timed out.
 30     *        *        *     Request timed out.

Trace complete.
```

AIM: To implement Dijkstra Algorithm

DESCRIPTION: Dijkstra's Algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent. It was conceived by computer scientist edsger w

PROGRAM:

Vertex.java

```
import java.util.*;
public class Vertex implements Comparable
{
    private boolean visited;
    private String name;
    private List list;
    private double dist = Double.MAXVALUE;
    private Vertex pr;
    public vertex( String name)
    {
        this.name = name;
        this.list = new ArrayList();
    }
    public List getList()
    {
        return list();
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public void setList(List list)
    {
        this.list = list;
    }
```

```
Public   addNeighbour(Edge edge)
{
    this.adjacent.add(edge);
}
Public   boolean visited()
{
    return visited;
}
Public   void setVisited(boolean visited)
{
    this.visited = visited;
}
Public   Vertex getPr()
{
    return pr;
}
Public   void setPr(Vertex pr)
{
    this.pr = pr;
}
Public   double getDist()
{   this.dist = dist;
}
public   void setDist(double dist)
{   this.dist = dist;
}
Public   String toString()
{   return this.name;
}
Public   int compareTo(Vertex otherV)
{
    return Double.compare(this.dist, otherV.getDist());
}
}
```

```
Public class Edge
{
    Private double weight;
    Private Vertex start;
    Private Vertex target;
    Public Edge( double weight, Vertex start, Vertex target)
    {
        this.weight = weight;
        this.start = start;
        this.target = target;
    }
    Public double getWeight()
    {
        return weight;
    }
    Public void setWeight( double weight)
    {
        this.weight = weight;
    }
    Public Vertex getStartVert()
    {
        return start;
    }
    Public void setStartVert( Vertex start)
    {
        this.start = start;
    }
    Public Vertex getTargetVert()
    {
        return target;
    }
    Public void setTargetVert(Vertex target)
    {
        this.target = target;
    }
}
```

PathFinder.java

```
import java.util.*;
Public class PathFinder
{
    Public void ShortestP (Vertex sourceV)
    {
        sourceV.setDist(0);
        PriorityQueue pq = new PriorityQueue();
        pq.add(sourceV);
        sourceV.setVisited(true);
        while (!pq.isEmpty())
        {
            Vertex av = pq.poll();
            for (Edge edge : av.getList())
            {
                Vertex v = edge.getTargetVert();
                if (!v.visited())
                {
                    double newDistance = newGetList() + edge.getWeight();
                    if (newDistance < v.getDist())
                    {
                        pq.remove(v);
                        v.setDist(newDistance);
                        v.setPr(actualVertex);
                        pq.add(v);
                    }
                }
            }
            actualVertex.setVisited(true);
        }
    }
    Public List getShortestPathTo(Vertex target)
    {
        hint path = new ArrayList();
```

```
for ( Vertex Vertex = target; vertex != null; vertex = vertex.getPrev())
  {
      path.add(vertex);
  }
  Collections.reverse(path);
     return path;
  }
}
```

PathFinder.java

```
public class Dijkara
{
   public static void main (String args[])
    {
        Vertex VA = new Vertex("A");
        Vertex VB = new Vertex("B");
        Vertex VC = new Vertex("C");
        Vertex VD = new Vertex("D");
        Vertex VE = new Vertex("E");

        VA.addNeighbour( New Edge(3, VA, VC));
        VA.addNeighbour( newEdge(5, VA, VB));
        VC.addNeighbour( newEdge(2, VC, VB));
        VC.addNeighbour( new Edge(6, VC, VD));
        VC.addNeighbour( new Edg(5, VC, VE));
        VB.addNeighbour( newEdg(4, VB, VC));
        VB.addNeighbour( newEdge(3, VB, VD));
        VB.addNeighbour( newEdge(4, VB, VE));
        PathFinder P1 = new PathFinder();
        P1.shortestP(VA);
        System.out.println("Minimum from A to B:" + VB.getDist());
```

```
System.out.println("Minimum from A to C:" + Vc.getDist());
System.out.println("Minimum from A to D:" + VD.getDist());
System.out.println("Minimum from A to E:" + Ve.getDist());
System.out.println();
System.out.println("Shortest path from A to B:" + pf.getshortest(VB));
System.out.println("Shortest paten from A to C:" + pf.getshortest(Vc));
System.out.println("Shortest path from A to D:" + pf.getshortest(VD));
System.out.println("Shortest path from A to E:" + pf.getshortest(Ve));
}
```

```
C:\Users\srija\Downloads>javac DijkstraShortestPath.java

C:\Users\srija\Downloads>javac DijkstraMain.java

C:\Users\srija\Downloads>java DijkstraMain
=======================================
Calculating minimum distance
=======================================
Minimum distance from A to B: 15.0
Minimum distance from A to C: 10.0
Minimum distance from A to D: 16.0
Minimum distance from A to E: 21.0
====================   ================
Calculating Paths
=======================================
Shortest Path from A to B: [A, C, B]
Shortest Path from A to C: [A, C]
Shortest Path from A to D: [A, C, B, D]
Shortest Path from A to E: [A, C, E]
```