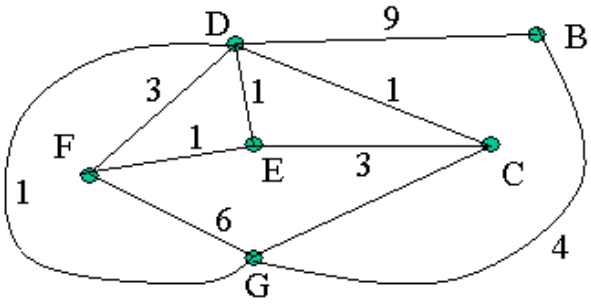
Problem 4:



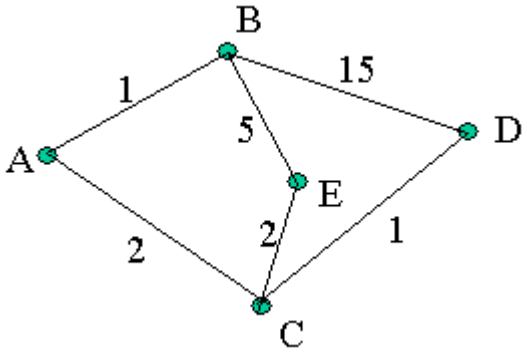
(from Mandy)

Step	*Start N*	*D(D), p(D)*	*D(E), p(E)*	*D(G),D(G)*	*D(B), p(B)*	*D(C),p(c*
0	***F***	***3,F***	***1,F***	***6,F***	***infinity***	***infinity***
1	***FE***	***2,E***	***--***	***6,F***	***infinity***	***4,E***
2	***FED***	***--***	***--***	***3,D***	***11,D***	***3,D***
3	***FEDC***	***--***	***--***	***3,D***	***11,D***	***--***
4	***FEDCG***	***--***	***--***	***--***	***7,G***	***--***

(from Jacob)

Step	Start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(G),p(G)
0	F	Inf	Inf	3,F	1,F	6,F
1	FE	Inf	4,E	2,E		6,F
2	FED	11,D	3,D			3,d
3	FEDG	7,G	3,D			
4	FEDGB		3,D			
5	FEDGBC					

(b)



ABCDE send data ABCDE send data BD send data

D^A	B	C
B	1	inf
C	inf	2
D	inf	inf
E	inf	inf

D^A	B	C
B	1	inf
C	inf	2
D	16	3
E	6	4

D^A	B	C
B	1	5
C	4	2
D	16	3
E	6	4

D^A	B	C
B	1	5
C	4	2
D	5	3
E	6	4

D^B	A	D	E
A	1	inf	inf
C	inf	inf	inf
D	inf	15	inf
E	inf	inf	5

D^B	A	D	E
A	1	inf	inf
C	3	16	7
D	inf	15	l5
E	inf	25	5

D^B	A	D	E
A	1	18	9
C	3	16	7
D	4	15	8
E	5	18	5

unchanged

D^C	A	D	E
A	2	inf	inf
B	inf	inf	inf
D	inf	1	inf
E	inf	inf	2

D^C	A	D	E
A	2	inf	inf
B	3	16	7
D	inf	1	12
E	inf	11	2

D^C	A	D	E
A	2	4	6
B	3	16	7
D	5	1	5
E	6	4	2

D^C	A	D	E
A	2	4	6
B	3	5	7
D	5	1	5
E	6	4	2

D^D	B	C	E
A	inf	inf	inf
B	15	inf	inf
C	inf	1	inf
E	inf	inf	10

D^D	B	C	E
A	16	3	inf
B	15	inf	15
C	inf	1	12
E	20	3	10

D^D	B	C	E
A	16	3	14
B	15	4	15
C	18	1	12
E	20	3	10

unchanged

D^E	B	C	D
A	inf	inf	inf
B	5	inf	inf
C	inf	2	inf
D	inf	inf	10

D^E	B	C	D
A	6	4	inf
B	5	inf	25
C	inf	2	11
D	20	3	10

D^E	B	C	D
A	6	4	13
B	5	5	25
C	8	2	11
D	20	3	10

D^E	B	C	D
A	6	4	13
B	5	5	14 (converge)
C	8	2	11
D	9	3	10

(From Justin)

ITERATION #1

D-A	B	C
B	1	INF
C	INF	2

D-B	A	E	D
A	1	INF	INF
E	INF	5	INF
D	INF	INF	15

D-C	E	A	D
A	INF	2	INF
E	2	INF	INF
D	INF	INF	1

D-D	B	C
B	15	INF
C	INF	1

D-E	B	C
B	5	INF
C	INF	2

ITERATION #2

D-A	B	C
B	1	INF
C	INF	2
D	16	3
E	6	INF

D-B	A	E	D
A	1	INF	INF
C	3	7	16
D	INF	INF	15
E	INF	5	INF

D-C	E	A	D
A	INF	2	INF
B	7	3	16
D	INF	INF	1
E	2	INF	INF

D-D	B	C
A	16	3
B	15	INF
C	INF	1
E	20	3

D-E	B	C
A	6	4
B	5	INF
C	INF	2
D	20	3

ITERATION #3

D-A	B	C
B	1	5
C	4	2
D	16	3
E	6	4

D-B	A	E	D
A	1	9	18
C	3	7	16
D	4	8	15
E	7	5	18

D-C	E	A	D
A	6	2	4
B	7	3	16
D	5	5	1
E	2	8	4

D-D	B	C
A	16	3
B	15	4
C	18	1
E	20	3

D-E	B	C
A	6	4
B	5	5
C	8	2
D	20	3

ITERATION #4

D-A	B	C
B	1	5
C	4	2
D	5	3
E	6	4

D-B	A	E	D
A	1	9	18
C	3	7	16
D	4	8	15
E	5	5	18

D-C	E	A	D
A	6	2	4
B	7	3	5
D	5	5	1
E	2	6	4

D-D	B	C
A	16	3
B	15	4
C	18	1
E	20	3

D-E	B	C
A	6	4
B	5	5
C	8	2
D	13	3

Finished, All of the Distance Vector tables have now converged.

(From Nick)

Step One

	Cost to destination via																
Destination	$D^A()$	B	C	$D^B()$	A	D	E	$D^C()$	A	D	E	$D^D()$	B	C	$D^E()$	B	C
	B	**1**	Inf	A	**1**	Inf	Inf	A	**2**	Inf	Inf	A	Inf	Inf	A	Inf	Inf
	C	Inf	**2**	C	Inf	Inf	Inf	B	Inf	Inf	Inf	B	**15**	Inf	B	**5**	Inf
	D	Inf	Inf	D	Inf	**15**	Inf	D	Inf	**1**	Inf	C	Inf	**1**	C	Inf	**2**
	E	inf	Inf	E	inf	Inf	**5**	E	inf	Inf	**2**	E	inf	Inf	D	inf	Inf

Step Two

	Cost to destination via																
Destination	$D^A()$	B	C	$D^B()$	A	D	E	$D^C()$	A	D	E	$D^D()$	B	C	$D^E()$	B	C
	B	**1**	Inf	A	**1**	Inf	Inf	A	**2**	Inf	Inf	A	16	**3**	A	6	**4**
	C	Inf	**2**	C	**3**	16	7	B	**3**	16	7	B	**15**	Inf	B	**5**	Inf
	D	Inf	**3**	D	Inf	**15**	Inf	D	Inf	**1**	Inf	C	Inf	**1**	C	Inf	**2**
	E	6	**4**	E	inf	Inf	**5**	E	inf	Inf	**2**	E	20	**3**	D	20	**3**

Step Three

	Cost to destination via																
Destination	$D^A()$	B	C	$D^B()$	A	D	E	$D^C()$	A	D	E	$D^D()$	B	C	$D^E()$	B	C
	B	**1**	5	A	**1**	18	9	A	**2**	4	6	A	16	**3**	A	6	**4**
	C	4	**2**	C	**3**	16	7	B	**3**	16	7	B	15	**4**	B	**5**	5
	D	16	**3**	D	**4**	15	8	D	5	**1**	5	C	18	**1**	C	8	**2**
	E	6	**4**	E	5	18	**5**	E	6	4	**2**	E	20	**3**	D	20	**3**

Step Four

	Cost to destination via																
Destination	$D^A()$	B	C	$D^B()$	A	D	E	$D^C()$	A	D	E	$D^D()$	B	C	$D^E()$	B	C
	B	**1**	5	A	**1**	18	9	A	**2**	4	6	A	16	**3**	A	6	**4**
	C	4	**2**	C	**3**	16	7	B	**3**	5	7	B	15	**4**	B	**5**	5
	D	5	**3**	D	**4**	15	8	D	5	**1**	5	C	18	**1**	C	8	**2**
	E	6	**4**	E	5	18	**5**	E	6	4	**2**	E	20	**3**	D	9	**3**

Problem 5: (from Kelly)
Udp_echo_server.c

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define BUFFER_SIZE 2048
```

```
int main(int argc, char *argv[])
{
 int port_number = 0;
 int pre_file_descriptor, file_descriptor;
 int client_length = 0;
 int bytes_read = 0;
 int m = 0, n = 0;
 char buffer[BUFFER_SIZE];
 struct sockaddr_in client_addr, server_addr;

 if (argc < 2)
 {
  printf("Usage: udp_echo_server <port number>\n");
  exit(-1);
 }

 port_number = atoi(argv[1]);

 bzero(buffer, BUFFER_SIZE);

 bzero( (char *)&server_addr, sizeof(server_addr));
 bzero( (char *)&client_addr, sizeof(client_addr));

 if( (pre_file_descriptor = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
 {
  printf("No socket descriptors available.\n");
  exit(1);
 }

 server_addr.sin_family = AF_INET;
 server_addr.sin_addr.s_addr = htonl(INADDR_ANY);
 server_addr.sin_port = htons(port_number);

 if(bind (pre_file_descriptor, (struct sockaddr *) &server_addr, sizeof(server_addr)) < 0)
 {
  printf("unable to bind\n");
  exit(-2);
 }

 printf("waiting on UDP port number %d\n", port_number);

 while(1)
 {
  client_length = sizeof(client_addr);
  bzero(buffer, BUFFER_SIZE);
```

```
   n = recvfrom(pre_file_descriptor, buffer, BUFFER_SIZE, 0, (struct sockaddr *)
&client_addr, &client_length);
   if(n < 0)
   {
    printf("error reading from UDP port %d\n", port_number);
   }

   m = sendto(pre_file_descriptor, buffer, BUFFER_SIZE, 0, (struct sockaddr *)
&client_addr, client_length);
   if(m)
   {
    printf("error writing to UDP port %d\n", port_number);
   }

   printf("%s", buffer);

  }
}
```

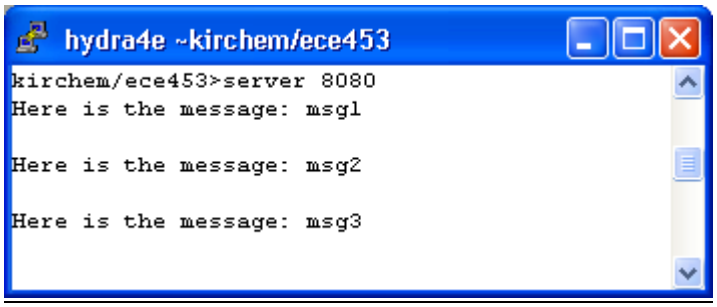
udp_echo_client.c

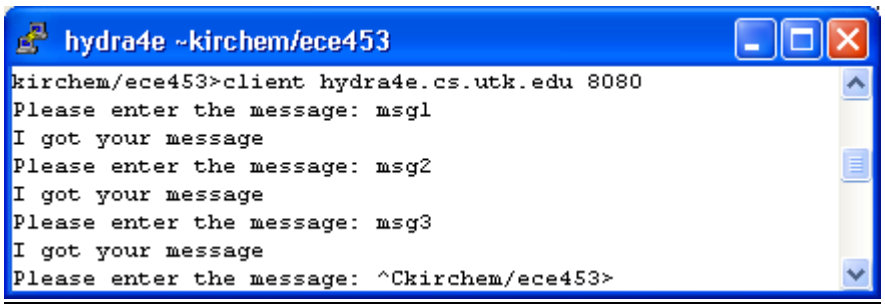
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#define BUFFER_SIZE 2048

int main(int argc, char *argv[])
{
 int port_number = 0;
 int pre_file_descriptor, file_descriptor;
 int client_length = 0;
 int bytes_read = 0;
 int m = 0, n = 0;
 char buffer[BUFFER_SIZE];
 struct sockaddr_in client_addr, server_addr;
 struct hostent *server;

 if (argc < 3)
 {
  printf("Usage: udp_echo_client <hostname> <port number>\n");
  exit(-1);
 }
```

```
  port_number = atoi(argv[2]);

  bzero(buffer, BUFFER_SIZE);

  bzero( (char *)&server_addr, sizeof(server_addr));
  bzero( (char *)&client_addr, sizeof(client_addr));

  if( (server = gethostbyname(argv[1])) == NULL)
  {
   printf("%s does not resolve\n", argv[1]);
  }

printf("server IP: %s\n", (char *)server->h_addr);


  server_addr.sin_family = AF_INET;
//  bcopy( (char *)server->h_addr, (char *)&server_addr.sin_addr.s_addr, server-
>h_length);
  server_addr.sin_port = htons(port_number);

  if( (pre_file_descriptor = socket(AF_INET, SOCK_DGRAM, 0)) < 0)
  {
   printf("No socket descriptors available.\n");
   exit(1);
  }

  printf("ready to send to %s on UDP port number %d\n", argv[1], port_number);

  while(1)
  {
   bzero(buffer, BUFFER_SIZE);
   fgets(buffer, BUFFER_SIZE, stdin);

printf("fget: %s\n", buffer);
   client_length = sizeof(client_addr);

   m = sendto(pre_file_descriptor, buffer, BUFFER_SIZE, 0, (struct sockaddr *)
&server_addr, server_length);
write(file_descriptor, buffer, BUFFER_SIZE);
   if(m)
   {
    printf("error writing to UDP port %d\n", port_number);
   }

   bzero(buffer, BUFFER_SIZE);
```

```
    n = read(pre_file_descriptor, buffer, BUFFER_SIZE);
    if(n < 0)
    {
      printf("error reading from UDP port %d\n", port_number);
    }

    printf("%s", buffer);

  }
}
```

(from Nick)





compiled with:

gcc –o client client.c –lsocket –lnsl

gcc –o server server.c –lsocket -lnsl

```
/* A simple server in the internet domain using UDP
   The port number is passed as an argument */
#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/uio.h>
#include <netinet/in.h>

void error(char *msg)
{
```

```
    perror(msg);
    exit(1);
}

int main(int argc, char *argv[])
{
     int sockfd, newsockfd, portno, clilen;
     char buffer[256];
     struct sockaddr_in serv_addr, cli_addr;
     int n, sz;

     // check cmd line arguments
     if (argc < 2) {
         fprintf(stderr,"ERROR, no port provided\n");
         exit(1);
     }

     // Create socket
     sockfd = socket(AF_INET, SOCK_DGRAM, 0);
     if (sockfd < 0)
        error("ERROR opening socket");

     // Bind Socket
     bzero((char *) &serv_addr,sizeof(serv_addr));
     portno = atoi(argv[1]);
     serv_addr.sin_family = AF_INET;
     serv_addr.sin_addr.s_addr = INADDR_ANY; /* server can have more
than 1 IP address */
     serv_addr.sin_port = htons(portno);
     if (bind(sockfd, (struct sockaddr *)                    &serv_addr,
              sizeof(serv_addr)) < 0)
              error("ERROR on binding");

     clilen = sizeof(cli_addr);
     bzero(buffer,256);

     // Receive any incoming datagrams
     while (1) {
        bzero((char *) &cli_addr,sizeof(cli_addr));
        sz = sizeof(cli_addr);
        n = recvfrom(sockfd, buffer, 255, 0, &cli_addr, &sz);
        if (n < 0)
           error("ERROR reading from socket");
        printf("Here is the message: %s\n",buffer);
        n = sendto(sockfd,"I got your
message",18,0,&cli_addr,sizeof(cli_addr));
        if (n < 0)
           error("ERROR writing to socket");
     }
     return 0;
}

// Client Daemon using UDP

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
```

```
#include <sys/uio.h>
#include <netinet/in.h>
#include <netdb.h>

void error(char *msg)
{
    perror(msg);
    exit(0);
}

int main(int argc, char *argv[])
{
    int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

    char buffer[256];
    if (argc < 3) {
       fprintf(stderr,"usage %s hostname port\n", argv[0]);
       exit(0);
    }
    portno = atoi(argv[2]);

    // Create socket
    sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd < 0)
        error("ERROR opening socket");
    server = gethostbyname(argv[1]);
    if (server == NULL) {
        fprintf(stderr,"ERROR, no such host\n");
        exit(0);
    }
    bzero((char *) &serv_addr, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    bcopy((char *)server->h_addr,
         (char *)&serv_addr.sin_addr.s_addr,
         server->h_length);
    serv_addr.sin_port = htons(portno);

    // Send messages until CTRL-C
    while (1)
    {
      // Get the message to send
      printf("Please enter the message: ");
      bzero(buffer,256);
      fgets(buffer,255,stdin);

      // Send datagram
      n =
sendto(sockfd,buffer,strlen(buffer),0,&serv_addr,sizeof(serv_addr));
      if (n < 0)
           error("ERROR writing to socket");
      bzero(buffer,256);
      n = recvfrom(sockfd,buffer,255, 0, NULL, NULL);
      if (n < 0)
           error("ERROR reading from socket");
      printf("%s\n",buffer);
```

```
    }

    return 0;
}
```