Homework assignment
Ipv6 e protocolli di routing

Q1. In uno sistema autonomo si usano sia RIP che OSPF per il routing interno al sistema autonomo. I cammini ottenuti sono sempre gli stessi per i due protocolli, oppure no? Si supponga che i due protocolli operino sugli stessi dati e che non avvenga nessun cambiamento nel sistema autonomo durante l'esecuzione dei protocolli.
no perché i due protocolli possono usare metriche diverse: RIP usa sempre hop count, OSPF non è detto.

Q2. L'host M invia un datagram D IPv6 al server S (che usa IPv6) lungo il percorso mostrato sotto, in cui R_1, R_2, R_3, R_4 sono router IPv6 e tra R_2 ed R_3 il percorso attraversa una parte in cui sono presenti solamente router IPv4, e pertanto utilizza la tecnica del tunneling in tale parte. Nel tunnel, D subisce una frammentazione. Indicare – giustificando la risposta – chi, tra M, S, R_1, R_2, R_3, R_4 effettua il riassemblaggio di D.

All'inizio del tunnel (cioè in R_2), viene premessa a D una intestazione IPv4. All'atto della frammentazione, i dati per poter riassemblare D vengono inseriti in tale intestazione. Questa intestazione viene tolta da R_3 alla fine del tunnel, e nel resto del percorso, D viaggia con la sola intestazione IPv6. Quindi, S non potrebbe riassemblare D perché gli mancherebbero i dati per tale azione. Pertanto, il riassemblaggio deve avvenire in R_3, che è la destinazione del tunnel, e cioè all'uscita del tunnel.

Q3. Un nodo R che utilizza RIP come protocollo di routing, dopo aver calcolato che la sua distanza per la destinazione Z è n+2, riceve da un suo vicino V un advertisement A che pubblicizza una distanza n per Z. Indicare –giustificando la risposta– in quali casi R **non** aggiorna la sua distanza per Z dopo aver ricevuto A.
R aggiorna sempre a n+1 la sua distanza per Z, dato che DV(Z)+1=n+1<n+2.

Q4. Sia R un sistema autonomo connesso a due reti di provider A e B. Supponiamo che un gateway di R riceva da un router di B un advertisement BGP contenente una rotta per un prefisso P. Indicare, giustificando la risposta, in quali casi e come R propagherà l'informazione ricevuta, nonché i principali attributi della rotta stessa.
Un gateway G di R che riceve da un gateway di B un advertisement contenente una rotta per P accetta o meno tale rotta in base alle proprie politiche di importazione. Se la accetta, G propaga la rotta a tutti gli altri router di R con cui ha una connessione iBGP. I due attributi più importanti associati alla rotta sono AS-PATH, che indica la sequenza degli AS attraversati dall'advertisement, e NEXT-HOP, che indica l'interfaccia del gateway dell'ultimo AS in AS-PATH.

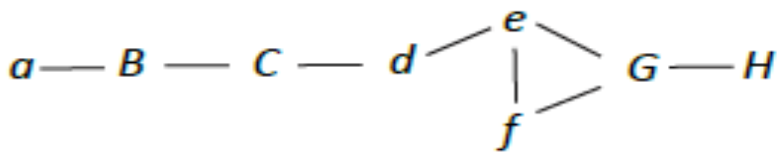
Q5. Un sistema autonomo S utilizza OSPF come protocollo di instradamento interno ad S. Sia R una sottorete esterna ad S, raggiungibile da S mediante il gateway G1 o il gateway G2, e sia A un router (non gateway) di S, nella cui tabella di inoltro è presente una riga relativa a R e contenente G1 come gateway verso R, e n+2 come costo dell'AS-PATH relativo. Supponiamo che A riceva da G2 l'annuncio di una rotta per R il cui AS-PATH contiene n sistemi autonomi. Indicare, giustificando al risposta, sotto quali ipotesi A aggiornerà la sua tabella sostituendo l'informazione già contenuta con quella ricevuta da G2.

A utilizza BGP per il routing tra AS, quindi seleziona la rotta con preferenza locale massima, e a parità di preferenza locale, quella con AS-PATH più piccolo. Quindi A aggiornerà la sua tabella se la rotta per R pubblicizzata da G2 ha preferenza locale maggiore od uguale di quella pubblicizzata da G1

Q6. Si consideri un AS formato da 13 router, 17 reti locali e 94 host. Quante connessioni TCP vengono instaurate per generare le sessioni iBGP interne all'AS necessarie al funzionamento di iBGP? Giustificare la risposta.

Ogni sessione iBGP utilizza una connessione TCP sulla porta 179. Per far funzionare iBGP si deve instaurare una sessione per ogni coppia di router dell' AS, quindi avremo 13x12/2=13x6=78 connessioni TCP.

Q7. Consideriamo il frammento di sistema autonomo illustrato sotto, i cui router utilizzano RIP come protocollo intra-dominio e in cui B, C, G, H sono router IPv6 mentre a, d, e, f sono router IPv4. Supponiamo che G riceva da H un datagram P IPv6 contenente FE80::0202:B3FF:FE1E:8329 come indirizzo sorgente e l'indirizzo IPv6 dell'interfaccia di B sul collegamento BC come indirizzo destinazione. Supponendo che G utilizzi il tunneling per inoltrare tale datagram, indicare il contenuto dei campi indirizzo sorgente e indirizzo destinazione del datagram trasmesso da G per inoltrare P.



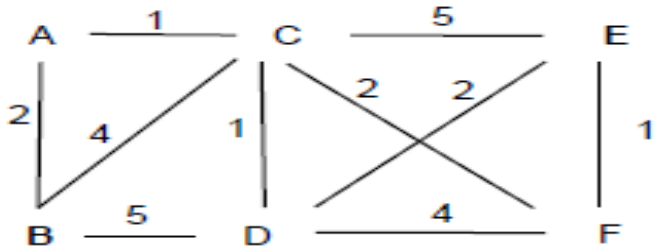
Il datagram trasmesso da G conterrà nel campo indirizzo sorgente l'indirizzo IPv4 dell'interfaccia di G sul collegamento eG e nel campo indirizzo destinazione l'indirizzo IPv4 dell'interfaccia di C sul collegamento Cd

Q8. Consideriamo una sottorete di Internet che utilizza IPv6. Indicare –giustificando la risposta– se è possibile o meno che in tale sottorete **non** sia necessario il processo di traduzione tra indirizzi IP e indirizzi di livello link.
Si, ciò è possibile poiché gli indirizzi di livello link (se di lunghezza non superiore a 64 bit) possono essere incorporati negli indirizzi IPv6.

Q9. Un sistema autonomo A utilizza RIP come protocollo di instradamento interno ad A. Z è una sottorete esterna ad A, raggiungibile da A mediante i gateway G1 e G2. Un router R interno ad A, dopo aver ricevuto l'annuncio di G1 di una rotta per Z con lunghezza di AS-PATH pari a n, riceve da G2 l'annuncio di una nuova rotta per Z anch'essa con lunghezza di AS-PATH pari a n. Indicare –giustificando la risposta– sotto quali ipotesi R aggiornerà la sua tabella di inoltro.

Dato che A utilizza BGP come protocollo di instradamento tra sistemi autonomi, R aggiornerà la sua tabella di inoltro se la rotta per Z pubblicizzata da G2 ha valore di preferenza locale maggiore di quella pubblicizzata da G1, oppure –se le due rotte hanno lo stesso valore di preferenza locale- se la rotta pubblicizzata da G2 ha NEXT-HOP a minor costo di quella pubblicizzata da G1.

E1. Consideriamo il sistema autonomo AS0 sotto riportato, i cui unici gateway sono B ed E, e i cui nodi utilizzano come preferenza locale per BGP la distanza minima calcolata da distance vector con poisoned reverse. Supponiamo che al tempo t, sia B che E comunichino agli altri router di AS0 la raggiungibilità di tre sistemi autonomi esterni: AS1, AS2 ed AS3. In particolare, B trasmette un advertisement di AS1 con |AS-PATH|=8, uno di AS2 con |AS-PATH|=7 e uno di AS3 con |AS-PATH|=3 , mentre E trasmette un advertisement di AS1 con |ASPATH|=10, uno di AS2 con |AS-PATH|=5 e uno di AS3 con |AS-PATH|=3. Determinare –giustificando la risposta– su quali ***collegamenti*** A, C, D e F inoltreranno i pacchetti destinati ad AS1, AS2 e AS3 dopo avere ricevuto tutti gli advertisement sopra menzionati.



- Poiché DA(B)=2 e DA(E)=4, A inoltrerà i pacchetti destinati ad AS1, AS2 e AS3 sul collegamento AB.
- Poiché DC(B)=3=DC(E), C inoltrerà i pacchetti destinati ad AS1 sul collegamento CA, quelli destinati ad AS2 sul collegamento CD o sul collegamento CF e quelli destinati ad AS3 sul collegamento CA o sul collegamento CD o sul collegamento CF.
- Poiché DD(B)=4 e DD(E)=2, D inoltrerà i pacchetti destinati ad AS1, AS2 e AS3 sul collegamento DE.
- Infine poiché DF(B)=5 e DF(E)=1, F inoltrerà i pacchetti destinati ad AS1, AS2 e AS3 sul collegamento FE.

E2. Consideriamo una versione del protocollo OSPF in cui:

Fase 1 - Quando un nodo viene attivato invia un messaggio OSPF di tipo "hello" a tutti i suoi vicini e quindi attende, per al più tx secondi, di ricevere una risposta di tipo "database description" per inizializzare il proprio LSDB. Se dopo tx secondi non ha ricevuto alcuna risposta di tipo "database description" il nodo invia di nuovo il messaggio "hello".

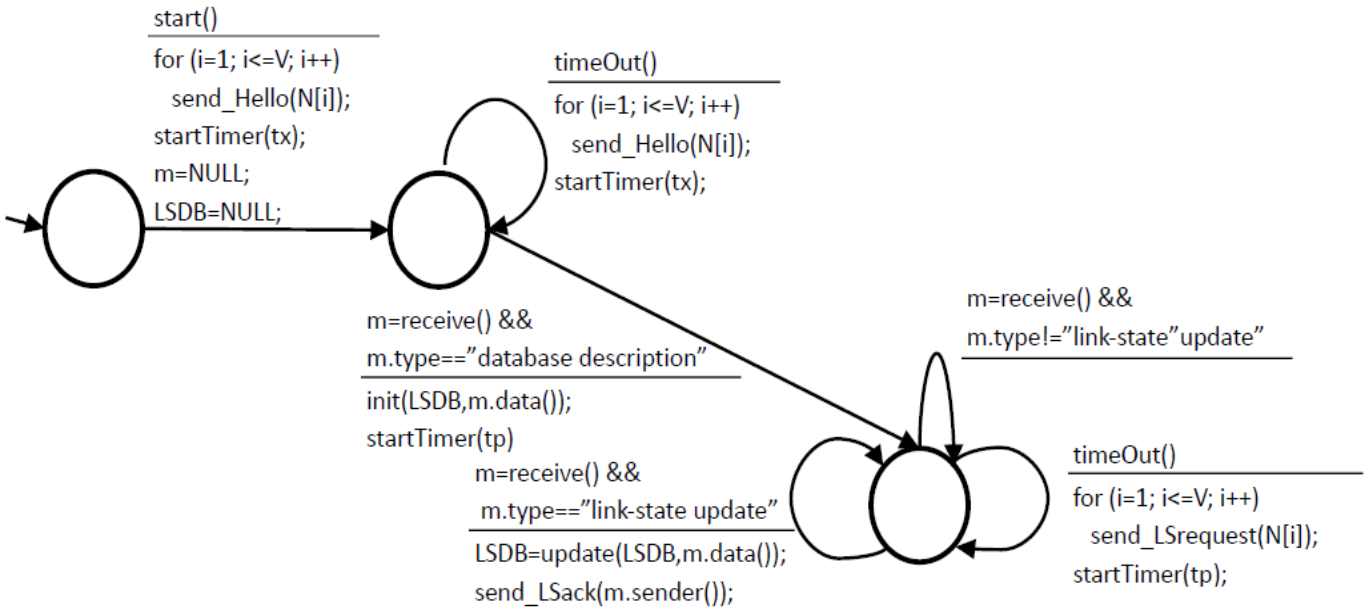
Fase 2 – Non appena riceve una risposta di tipo "database description", il nodo passa alla fase 2, in cui invia ogni tp secondi un messaggio OSPF di tipo "link-state request" a ciascun suo vicino per continuare ad aggiornare il proprio LSDB nel tempo.

Descrivere, utilizzando un automa a stati finiti, il comportamento sopra descritto di un nodo R, assumendo di disporre dei comandi

```
receive()                   // per ricevere un messaggio,
send_Hello(recipient)       // per inviare un messaggio di tipo "hello",
send_LSrequest(recipient)   // per inviare un messaggio di tipo "link-state request",
startTimer(duration)        // per avviare un timer
```

e assumendo che R memorizzi gli indirizzi dei suoi vicini in un array N[1]...N[V].

Per semplicità supporre che R non riceva messaggi di tipo "hello" né di tipo "link-state request".



E3. Scrivere il frammento di (pseudo)codice con cui un router R che utilizza IPv6 gestisce l'inoltro di un datagram ricevuto. Supponiamo per semplicità che il datagram ricevuto sia rappresentato da una struttura D, dove D.x è il contenuto del campo x dell'header del datagram. Supponiamo, sempre per semplicità, che le destinazioni siano identificate da numeri naturali, che i router direttamente collegati a R siano identificati dai primi N naturali e che R utilizzi i vettori:

- *nextHop,* dove nextHop[i]=j indica che il nextHop di R per la destinazione i è j,
- *typeIP,* dove typeIP[i]=k indica che k è la versione IP utilizzata da i, e
- *nextMTU,* dove nextMTU[i]=m indica che m è l'MTU accettato da i.

Supponiamo infine che R disponga (tra le altre) delle procedure:

- *ICMPv6(dest,type)*, per inviare un messaggio ICMP di tipo type al router dest,
- *TUNNEL(next,D)*, per inviare D in tunneling al vicino next,
- *IPv6Send(next,D)*, per inviare D al vicino next.

****

```
if D.hopLimit==0
        {ICMPv6(D.source, "03")}
    else {
        next=nextHop[D.dest];
        if typeIP[next]=="IPv4"
            {TUNNEL(next,D)}
          else if nextMTU[next]<40+D.payload
                {ICMPv6(D.source, "02")}
            else {
                D.hopLimit--;
                IPv6send(next,D)
              }
    }
```

E4. Consideriamo una variante del protocollo RIP che avvelena le distanze tenendo conto sia del *nextHop* (il vicino) che del *nextHop* del *nextHop* (il vicino del vicino). Descrivere – utilizzando uno pseudo-codice – il comportamento di un router R che utilizza tale protocollo quando riceve un advertisement da un vicino. Per semplicità, assumere che i nodi della rete siano identificati dai primi N naturali e che i primi M siano i vicini di R (ovviamente, M<N).

****

```
// R riceve da V advertisement A e aggiorna la propria tabella
for (i=0;i<N;i++)
        if ( D[i].next==empty || 1+A[i].cost<D[i].cost || V==D[i].next )
                { D[i].cost = 1+A[i].cost;
                  D[i].next = V;
                  D[i].nextnext = A[i].next;
                 }
// R invia la sua tabella (avvelenata) ai vicini
for (j=0;j<M;j++)
        {
          for (i=0;i<N;i++)
                {
                        if ( D[i].next==j || D[i].nextnext==j )
                                { toSend[i].cost = 16; }
                        else {
                                toSend[i].cost = D[i].cost;
                                 toSend[i].next = D[i].next;
                             }
        send(j,toSend);
        }
```

E5. Un sistema autonomo S consiste di n nodi, e suddivide tutti gli altri sistemi autonomi di internet in tre categorie: i nemici (memorizzati in una *black list*), gli incerti (memorizzati in una *grey list*), e gli amici (che non sono memorizzati in alcuna lista perché si possono dedurre per esclusione dalle due liste precedenti). Tali liste sono note a tutti i nodi di S. La politica di preferenza locale usata da eBGP prevede di scartare comunque tutti gli aggiornamenti che contengono sistemi autonomi della black list. Un messaggio di aggiornamento X ha preferenza locale superiore a quella di un messaggio di aggiornamento Y se entrambi non contengono nodi nemici, e X contiene un minor numero di sistemi autonomi che sono nella grey list rispetto a Y. Descrivere con uno pseudocodice il comportamento di un router di confine R per decidere se aggiornare o meno la sua tabella di inoltro (eventualmente propagando nel sistema autonomo tale aggiornamento) quando riceve un messaggio di aggiornamento A che pubblicizza la raggiungibilità di un sistema autonomo remoto T. Si supponga per semplicità che i vicini di R siano rappresentati dai primi m numeri interi, e si supponga di avere a disposizione *receive(A)* per ricevere l'aggiornamento A, *send(A,i)* per inviare A al vicino i, la funzione *isin(L,AS)* che restituisce true se e solo se AS è nella lista L. Inoltre, si supponga che la tabella di inoltro di R sia memorizzata in *tabinoltro*, il cui generico elemento h-esimo, associato al percorso per raggiungere il sistema autonomo Y, ha (tra gli altri) i campi *grey(h)*, intero che contiene il numero di sistemi autonomi dell'AS path appartenenti alla grey list nel percorso attualmente utilizzato per raggiungere Y, *pathlength(h)* che contiene la lunghezza di tale AS path, e che sia inizializzato a NULL quando il sistema viene attivato. Inoltre, si supponga che l'AS path contenuto in A sia nel vettore *A.path(.)*, la cui lunghezza è in *A.pathlength*.

****

La soluzione seguente prevede che path vector sia già stato utilizzato e quindi non si debba controllare se il sistema autonomo S sia presente o meno nell'AS path. Se si vuol controllare anche questo, basta (ad esempio) modificare la condizione del primo if in questo modo:

if (*isin*(blacklist,A.path(i))||*(A.path(i)==S)*)

****

```
receive(A);
black=false;
grey=0;
iter=A.pathlength;
i=1;
while ((i<iter)&&(!black))
      { if isin(blacklist,A.path(i))
              {black=true}
         else  if  isin(greylist,A.path(i))
                    {grey++};
          i++;}
if          ((!black)  &&  ((tabinoltro(h)!==NULL)  ||  (grey<tabinoltro(h).grey)  ||
(  (grey==tabinoltro(h).grey) && (A.pathlength<tabinoltro(h).pathlength)))
     {
         tabinoltro(h)=A;
         tabinoltro(h).grey=grey;
          for i=1 to m
              {send(A,i)};
      }
```

E6. Descrivere, con uno pseudocodice, le azioni svolte da un router R_g non di confine appartenente ad un sistema autonomo AS_z di transito, che utilizza RIP come protocollo di routing intra-AS, per inserire od aggiornare nella sua tabella di inoltro globale *tabin*, le informazioni relative alla raggiungibilità di un sistema autonomo AS_x , secondo quanto stabilito dal protocollo iBGP. Il messaggio *mess* che contiene tali informazioni, ha i campi: *mess.AS*, contenente il nome del sistema autonomo destinazione (AS_x in questo caso), *mess.reti*: lista delle reti appartenenti ad AS_x, *mess.routconf* identificativo del router di confine di AS_z a cui inoltrare i datagram destinati alle reti di AS_x, e *mess.ASpl*, che indica la lunghezza del cammino per raggiungere AS_x, secondo la metrica usata da BGP. La tabella *tabin* ha (tra le altre) le colonne *dest* (che contiene la rete destinazione), *nexthop* (che contiene il router a cui inoltrare i datagram destinati alla rete in dest), *ASpl* che contiene la lunghezza del cammino per raggiungere AS_x, e *costo*, che contiene il costo (secondo la metrica normalmente usata in iBGP) per raggiungere le reti destinazione. Inoltre, ogni router ha il vettore *distr* che contiene la distanza tra lui e tutti gli n router del suo sistema autonomo. Si supponga, per semplicità, che i router vicini di R_g siano rappresentati dai numeri interi compresi tra 1 ed m, che AS_x contenga una sola rete, che AS_z non utilizzi nessuna preferenza locale, e che, a parità di tutti gli altri attributi, si scelga l'ultimo percorso ricevuto. Si utilizzino le procedure *receive (mess, neigh)* per ricevere il messaggio mess, neigh è il vicino che l'ha mandato; e *lookup (tabin, rete, i)* che cerca "rete" in tabin. Se la trova, i è l'indice di tabin in cui si trova; altrimenti i è negativo. Infine, si supponga che tabin sia sovradimensionata, e che *lasttabin* sia l'indice dell'ultimo elemento di tabin che contiene informazioni valide (cioè, tabin ha informazioni valide comprese tra gli elementi 0 e lasttabin).

****

```
receive (mess, neigh); //per ricevere il messaggio mess; neigh è il vicino che l'ha mandato
lookup (tabin, mess.rete, i); // cerca "mess.rete" in tabin. Se lo trova, i è l'indice di tabin in cui si trova; else i è negativo
if ((i>=0)&&(mess.Aspl<=tabin[i].Aspl))
        { tabin[i].Aspl=mess.Aspl;
          tabin[i].nexthop=neigh;
          tabin[i].costo=distr(mess.routconf)+1;
         }
  else if (i<0)
             { lasttabin++;
               i=lasttabin;
               tabin[i].Aspl=mess.Aspl;
               tabin[i].nexthop=neigh;
               tabin[i].costo=distr[mess.routconf]+1;
               tabin[i].dest=mess.rete;

              }
```

E7. Descrivere, con uno pseudocodice, il comportamento del router demone, *routedaem*, di RIP di un nodo R. Per semplicità, si supponga che i vicini di R siano rappresentati dai primi M numeri interi, e si supponga di avere a disposizione:

- *UDP_receive(j,A)* per ricevere un aggiornamento A dal vicino j

- *UDP_send(i,A)* per inviare l'aggiornamento A al vicino i

- *tabinoltro* tabella di inoltro di n righe con i campi, tra gli altri, *costo* (che contiene il costo del percorso memorizzato in quella riga), e *nexthop* (che contiene il nome dell'entità a cui inviare i datagram se si usa quel percorso). Una riga che non contiene informazioni valide avrà valore *NULL*.

Per semplicità, si supponga che la tabella di inoltro contenga già tutte le *n* righe (alcune con contenuto valido ed altre che saranno necessarie in futuro, queste ultime inizializzate a NULL): quindi non bisogna inserire nuove righe.

****

Si suppone che la tabella abbia n righe. Si può supporre che se tab[i]==NULL, allora tab[i].costo è infinito.

```
UDP_receive(j,tab);
for i=1 to n, i++
        { if (tab[i]!=NULL)&&((tabinoltro[i]==NULL)||(tabinoltro[i].costo>tab[i].costo)||
(tabinoltro[i].nexthop==  tab[i].nexthop)))
                {tabinoltro[i]=tab[i]}
     }
for i=1 to n, i++
   { tabxvicini[i].costo=tabinoltro[i].costo++;
     tabxvicini[i].nexthop=R;
    }
for i=1 to M, i++
   {UDP_send(i, tabxvicini)}
```