Bölüm 6 :
YÖNLENDİRME TEKNİKLERİ ve ALGORİTMALARI

Türkçe (İngilizce) karşılıklar
Statik (static)
Topoloji (topology)
Ölçüt (metric)
Kaynakta yönlendirme (source routing)
Sekerek yönlendirme (hop by hop routing)
Yönlendirme tabloları (routing tables)
Taşkın (flooding)
Uzaklık vektörü yönlendirmesi (distance vector routing)
Hat durumu yönlendirmesi (link state routing)

Bölüm Hedefi
Bu bölümü bitirdiğinizde
- Bilgisayar ağlarında yönlendirme nedir?
- En kısa yol bulma (Dijkstra) algoritmasını
- Taşkın tekniğini
- Uzaklık vektörü yönlendirmesini
- Hat durumu yönlendirmesini

öğrenmiş olacaksınız.

Bu bölümde bir haberleşme ağı üzerinde yönlendirme tablolarını oluştururken kullanılan algoritmalar/teknikler üzerinde durulacaktır. Yönlendirme tabloları temelde *statik* ve *dinamik* olmak üzere iki yaklaşım ile oluşturulur. Statik yönlendirme tabloları belli bir algoritmaya dayanarak önceden oluşturulur ve bir daha değiştirilmez. Bu durumda, bir düğümden diğer düğümlere ulaşmak için kullanılacak yollar önceden bellidir ve ağdaki trafiğin değişiminden etkilenmez. Dinamik yönlendirme tablolarında ise tabloların zaman içinde, ağ trafiğinde ya da bağlantılarda meydana gelen değişimlerle, güncellenmesi hedeflenir.

Her iki tip yönlendirme tablosu kullanımının da olumlu ve olumsuz yönleri vardır. Statik yönlendirme basittir. Güncelleme gerektirmez ancak zaman zaman belli noktalarda oluşan tıkanıklıklar, trafiği farklı yollara yönlendirme imkanı olmadığı için, başarımın düşmesine neden olur. Ya da bazı hatların kopması sonunda, önceden atanmış yolları değiştirmek mümkün olmadığı için bazı düğümler arasında bağlantı kurulamayabilir. Dinamik yönlendirmede ise düğümler üzerindeki trafik yükünün artması, tıkanmalar ya da bağlantıların kopması sonucunda alternatif yollar oluşturulur. Ancak, dinamik yönlendirme algoritmaları statik algoritmalardan daha karmaşıktır. Hataya dayanıklı olması beklenen ağlarda yönlendirme tablolarının dinamik teknikler kullanılarak oluşturulması gerekir.

Paketlerin yönlendirilmesi (geçilecek düğümlerin belirlenmesi) iki şekilde gerçeklenir : *kaynakta yönlendirme(source routing)* ve *sekerek yönlendirme (hop by hop routing)*.
Kaynakta yönlendirmede kaynak düğüm, paketin sırası ile geçeceği düğümleri (yönlendiricileri) belirler ve bu bilgi pakete eklenir. Yönlendiriciler bu bilginin bulunduğu özel alana bakarak paketi sırası ile geçmesi gereken düğümlere aktarırlar. Bu yöntemde, yönlendirme işlemi kaynak düğümde yapılır. Diğer düğümler paketin belirlenen yol üzerinden geçmesini sağlar. Sekerek yönlendirmede ise paketin üzerindeki varış düğümü adresine

bakarak paketin gönderileceği bir sonraki düğümün adresi belirlenir ve paket o düğüme aktarılır. Yol üzerindeki her sekmede (düğümde) bu işlem yapılır ve paket varış düğümüne kadar ulaştırılır.

6.1 Yönlendirme Teknikleri :

Literatürde pek çok paket yönlendirme tekniğine rastlayabilirsiniz. Bu bölümde en temel yönlendirme teknikleri üzerinde durulacaktır.

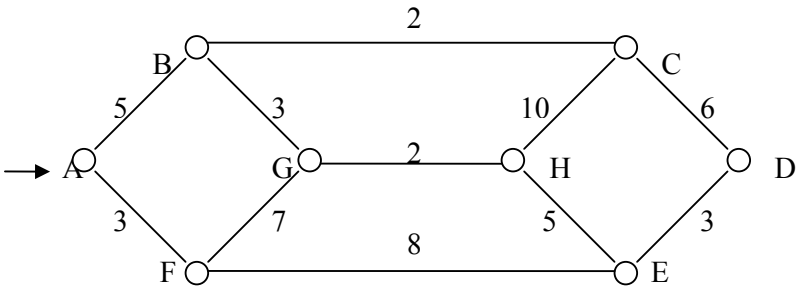
6.1.1 En Kısa Yolu Bulma Algoritması *(Dijkstra's Shortest Path Algorithm)*

Yönlendirme teknikleri üzerinde düşünmeye başladığınızda aklınıza gelebilecek ilk teknik iki nokta arasındaki en kısa yolu bulmak ve paketleri o yol üzerinden aktarmak olacaktır. Bilgisayar ağlarında iki nokta arasındaki en kısa yolu bulurken ölçüt olarak:
- bağlantı noktaları arasındaki coğrafi uzaklık,
- geçilen düğüm (sekme) sayısı, ya da
- hatlar üzerinde ortaya çıkan aktarım süreleri, düğümlerdeki kuyruklarda bekleme süreleri (gecikme değerleri)

kullanılabilir. Sonuçta amacımız, kullanılan ölçüte bağlı olarak kaynak noktasından varış noktasına en kısa yolun bulunmasıdır.

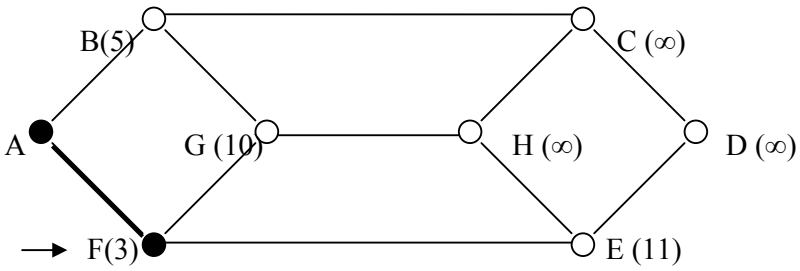
Bu bölümde açıklanacak en kısa yol algoritması Dijkstra tarafından geliştirilmiştir.

Algoritmamızı Şekil 6.1'de verilen ağ üzerinde daha iyi açıklayabiliriz. Şekilde kullanılan ölçütün gecikme değerleri olduğunu düşünelim. Amacımız A noktasından diğer düğümlere en kısa yolu bulmak olsun.

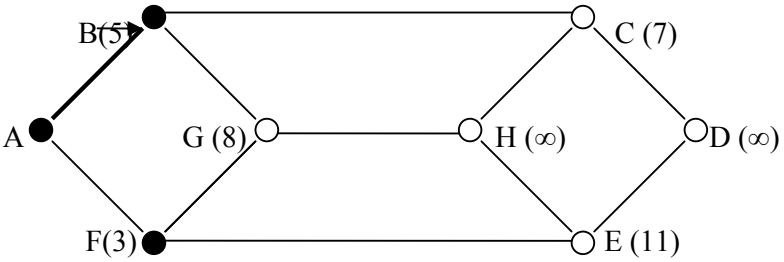


Şekil 6.1 Örnek ağ yapısı

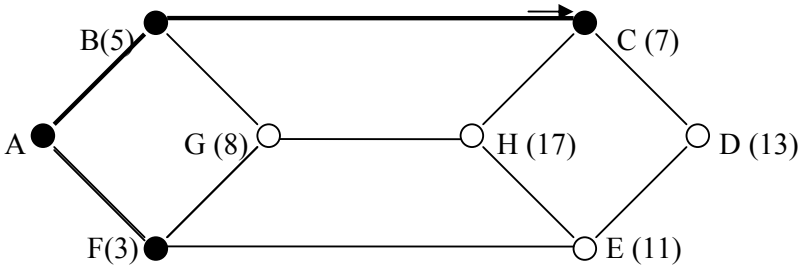
Bu durumda önce A düğümünden başlıyoruz ve ona komşu olan düğümleri inceleyip onlara *geçiçi uzaklık değerlerini* atıyoruz. Şekil 6.2(a)'da B ve F düğümleri için belirlenmiş geçiçi uzaklık değerleri bulabilirsiniz. Diğer düğümler henüz incelenmediği için onların geçici uzaklık değerleri sonsuz işareti ile gösterilmiştir.



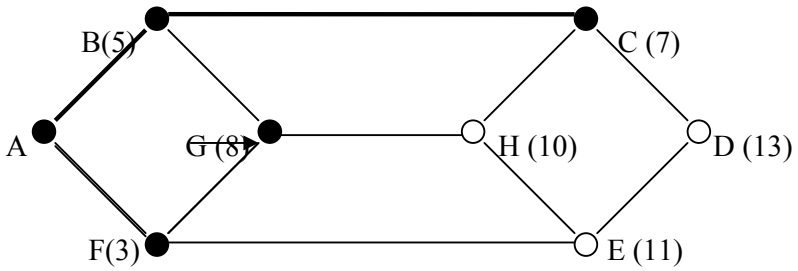
a)



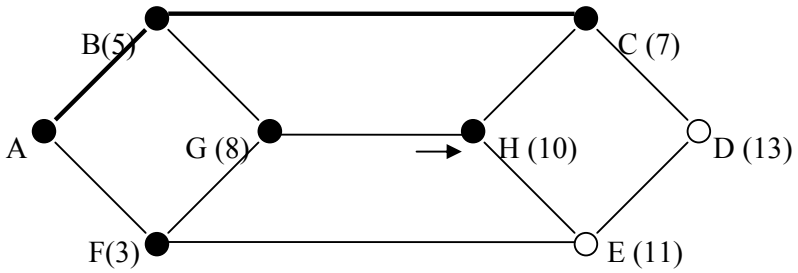
b)



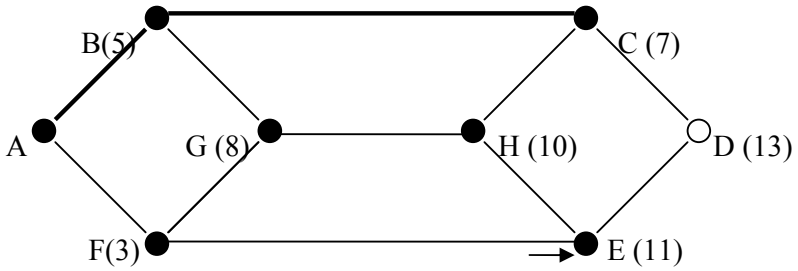
c)


B(5)
C (7)
A
G (8)
H (10)
D (13)
F(3)
E (11)

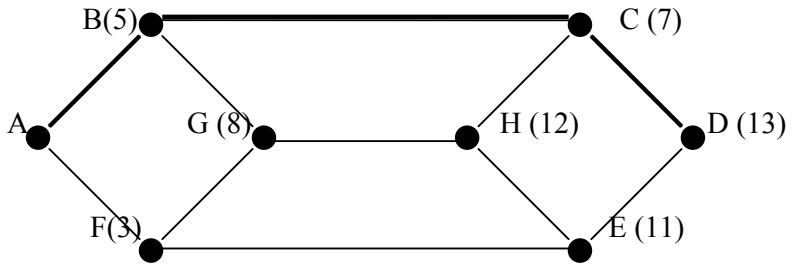
d)


B(5)
C (7)
A
G (8)
H (10)
D (13)
F(3)
E (11)

e)


B(5)
C (7)
A
G (8)
H (10)
D (13)
F(3)
E (11)

f)



g)

Şekil 6.2 Geçici uzaklık değerlerinin belirlenmesi

Daha sonra sırası ile A'dan ulaşılan en yakın düğümden (bu örnekte F) başlayarak uzaklık belirlemeye devam ediyoruz. F'nin seçiminden sonraki uzaklık değerleri şekil 6.2(b)'de verilmiştir. B, C, G, H ve E'nin seçiminden sonraki uzaklık değerleri 6.2(c-g)'de gösterilmiştir.

Örnekten de anlaşılacağı gibi bu yöntemde son düğüm dışında tüm düğümler bir kez işlenmiştir. Yukarıdaki örnekte A düğümüne olan en kısa yollar hesaplanmıştır. Yani kaynak düğüm A'dır. Aynı yöntem kullanılarak diğer kaynak düğümler için de hesaplama yapılır.

Dijkstra'nın en kısa yol algoritması Şekil 6.3'te verilmiştir. Bu algoritmada iki boyutlu L matrisi duğümler arasındaki uzaklıkları (ya da gecikme değerlerini) verir. *L[i,j], i* ve *j* düğümleri arasındaki uzaklığı (veya gecikmeyi) verir. Matriste sonsuz işareti ile gösterilmiş alanlar ilgili iki düğüm arasında bağlantı olmadığını gösterir. Yukarıdaki ağ topolojisindeki kenarlar çift yönlü bağlantıları göstermektedir. Yani hat üzerinde verilen değer hem *i*'den *j*'ye ulaşmak için hem de *j*'den *i*'ye ulaşmak için kullanılır. Bu nedenle *L[i,j]=L[j,i]* ve matris simetriktir. Bu özellik her ağ yapısında olmayabilir. Aşağıda, örneğimizdeki topoloji için L matrisi verilmiştir. Bu matris Şekil 6.1'e bakarak hazırlanır.

L=

A	*B*	*C*	*D*	*E*	*F*	*G*	*H*	
0	5	∝	∝	∝	3	∝	∝	*A*
5	0	2	∝	∝	∝	3	∝	*B*
∝	2	0	6	∝	∝	∝	10	*C*
∝	∝	6	0	3	∝	∝	∝	*D*
∝	∝	∝	3	0	8	∝	5	*E*
3	∝	∝	∝	8	0	7	∝	*F*
∝	3	∝	∝	∝	7	0	2	*G*
∝	∝	10	∝	5	∝	2	0	*H*

Aşağıdaki algoritmada tek boyutlu *D* matrisi kaynak düğümden diğer düğümlere olan en kısa yolların değerini hesaplarken (*geçiçi uzaklık değerlerini* saklamak için) kullanılmıştır.

Matrislerde indeks değeri olarak kullanılan 1, 2, 3, 4, 5, 6, 7, 8 …. sırası ile A, B, C, D, E, F, G, H düğümlerini ifade etmektedir. *n* ağdaki düğüm sayısını gösterir. Bu örnekte 8'dir.

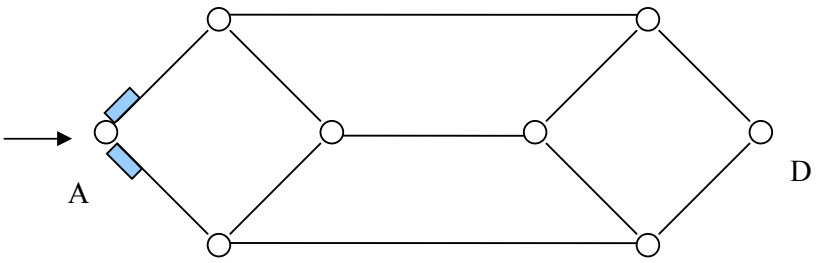
```
function Dijkstra(L[1..n, 1..n]) : array [2..n]
        array D[2..n]
        set C                           {incelenecek düğümleri saklar}

        C ← {2, 3, 4, 5, 6, …, n}
        for i ← 2 to n
                D[i] ← L[1,i]
        repeat  n - 2 times
                v ← C'den en küçük D[v] değerini sağlayan bir eleman
                v ← C \ {v}             {bu eleman C setinden çıkarılır}
                for each w ∈ C do
                        D[w] ← min(D[w], D[v] + L[v,w])
        return D
```

Şekil 6.3 Dijkstra'nın en kısa yolu bulma algoritması

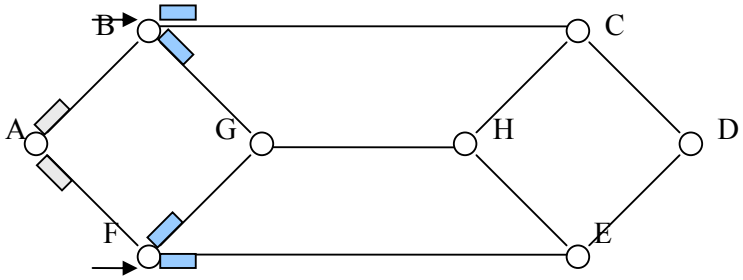
Dijkstra algoritması ağda oluşacak değişimlere duyarlı değildir. Bu nedenle bir bağlantının kopması ya da bir bağlantı üzerindeki trafik yükünün artması bazı düğümler arasındaki aktarımı imkansız ya da çok güç hale getirebilir.

6.1.2 Taşkın *(Flooding)*

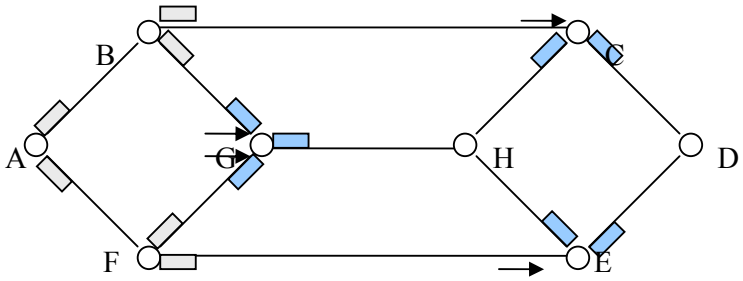
Bu yöntemde bir düğüme ulaşan paketin kopyaları çıkarılır ve bu kopyalar paketin geldiği bağlantı (hat) dışındaki tüm bağlantılardan gönderilir. Doğal olarak bu yöntem aynı paketin pek çok kopyasının yaratılmasına ve bu kopyaların ağdaki trafiği aşırı derecede yoğunlaştırmasına neden olacaktır. Bu dezavantaja karşın, taşkın yönteminde seçilecek hat için özel hesaplamalar yapılmasına gerek kalmaz. Paket, doğal olarak, ek kısa yol üzerinden varış noktasına erişir. Ancak bu sırada aynı paketin pek çok kopyası yaratılır. Hatta aynı kopyalar pek çok kez aynı düğümlere ulaşır. Şekil 6.4 (a-d), A düğümünden D düğümüne gönderilen bir paketin üç sekme içindeki çoğalmasını ve eriştiği düğümleri gösterir.



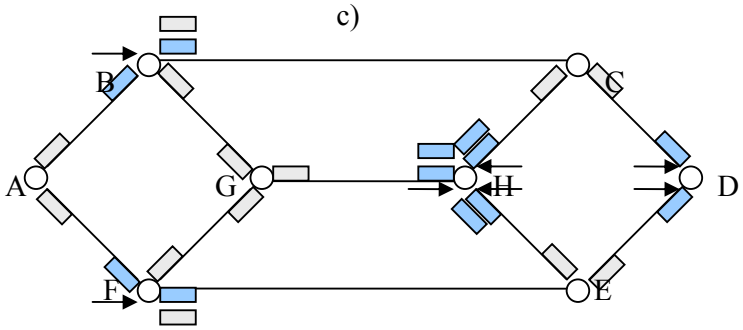
a)



b)



c)



d)

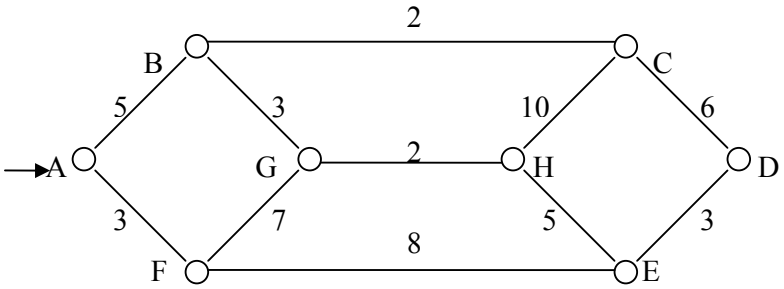
Şekil 6.4 Taşkın tekniğinde paketlerin ağ içinde yayılması

Tekniğin dezavantajlarını önlemek için paketlere *sekme sayacı* eklenmesi önerilmiştir. Sekme sayacına, paket yaratıldığında, kaynak düğüm tarafından kaynak ve varış düğümleri arasındaki sekme sayısını gösteren bir değer atanır. Sekme sayacının değeri geçilen her düğümde bir azaltılır. Sekme sayacı sıfır değerine ulaştığında, paket varış noktasına ulaşmamışsa yok edilir. Paketlerin aynı düğüm tarafından tekrar tekrar kopyalanmasını engellemek için önerilen bir diğer yöntem ise her düğümün yarattığı/kopyaladığı paketlerin kaydını tutmasıdır. Yeni gelen bir paketin kopyalarının yaratılmasından önce bu kayıtlar kontrol edilir ve paket daha önce çoğaltılmadıysa işleme devam edilir aksi halde paket yok edilir.

6.1.3 Uzaklık Vektörü Yönlendirmesi *(Distance Vector Routing)*

Uzaklık vektörü yönlendirmesi dinamik bir yönlendirme algoritmasıdır. Bu teknikte her yönlendirici bir yönlendirme tablosu tutar bu tabloda ağdaki her yönledirici için bir satır bulunur. Her satırda ilgili yönlendiricinin tablonun bulunduğu yönlendiriciye olan uzaklığı ve ilgili yönlendiriciye hangi çıkış hattı üzerinden ulaşılacağı bilgisi saklanır. Kullanılan ölçütün uzaklık olması gerekmez. Bu ölçüt *gecikme*, *sekme sayısı*, … olabilir. Gecikmenin ölçüt olarak kullanıldığı durumlarda, her yönlendirici kendisi ile komşu yönlendiriciler arasındaki gecikmeyi doğrudan bulabilir.

Uzaklık vektörü yönlendirmesinde, her yönlendirici (periyodik olarak) her *T* milisaniyede bir kendi tablosunda bulunan ölçüt değerlerini komşularına gönderir ve benzer bir tabloyu da komşusundan alır. Gelen tablolardaki verilere bakarak her yönlendirici *doğrudan bağlı olmadığı* yönlendiriciler ile arasındaki gecikme değerlerini ve o yönlendiricilere nasıl ulaşacağını bulabilir. Şekil 6.1'deki topoloji üzerindeki değerlerin bir anlık gecikme değerleri olduğunu ve ağın çok çok kısa bir süre önce aktif hale geldiğini varsayalım. Bu durumda her yönlendirici sadece komşusuna olan gecikmeyi bilsin ve diğer yönlendiricilere nasıl ulaşılacağı konusunda bilgi sahibi olmasın. Yönlendiricilerin tabloları Şekil 6.5'deki gibi olacaktır.



A Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	0	-
B	5	B
C	∞	-
D	∞	-
E	∞	-
F	3	F
G	∞	-
H	∞	-

B Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	5	A
B	0	-
C	2	C
D	∞	-
E	∞	-
F	∞	-
G	3	G
H	∞	-

C Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	∞	-
B	2	B
C	0	-
D	6	D
E	∞	-
F	∞	-
G	∞	-
H	10	H

D Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	∞	-
B	∞	-
C	6	C
D	0	-
E	3	E
F	∞	-
G	∞	-
H	∞	-

E Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	∞	-
B	∞	-
C	∞	-
D	3	D
E	0	-
F	8	F
G	∞	-
H	5	H

F Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	3	A
B	∞	-
C	∞	-
D	∞	-
E	8	E
F	0	-
G	7	G
H	∞	-

G Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	∞	-
B	3	B
C	∞	-
D	∞	-
E	∞	-
F	7	F
G	0	-
H	2	H

H Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	∞	-
B	∞	-
C	10	C
D	∞	-
E	5	E
F	∞	-
G	2	G
H	0	-

Şekil 6.5 Uzaklık vektörü yönlendirmesinde tabloların ilk durumu

Bu andan sonra B, F ve H yönlendiricilerinden gelen tablolara göre G yönlendiricisinin kendi yönlendirme tablosunu güncelleyişi Şekil 6.6'da verilmiştir. G yönlendiricisi B, F ve H'den kendisine ulaşan vektörlere göre yönlendirme tablosundaki değerleri günceller ve doğrudan bağlı olmadığı yönlendiricilere olan gecikmeyi hesaplar. Güncelleme sırasında amaç en kısa gecikmeye sahip çıkış hattını bulmaktır. Bir yönlendiriciye ulaşmak için alternatif yollar olabilir ancak bu yolların en kısası seçilmelidir. Örneğin, G yönlendiricisi, A'ya F (7+3) ya da B (3+5) üzerinden ulaşabilir. Ancak B üzerinden gidilen yol daha kısa olduğu için onu tercih eder.

Uzaklık vektörlerinin aktarılmasından sonra G'nin doğrudan bağlı olmadığı A, C ve E yönlendiricilerine olan gecikme değerlerine sahip olduğunu görmekteyiz.

B Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	5	A
B	0	-
C	2	C
D	∞	-
E	∞	-
F	∞	-
G	3	G
H	∞	-

F Yönlendiricisi

	gecikme	Bir-sonraki düğüm
A	3	A
B	∞	-
C	∞	-
D	∞	-
E	8	E
F	0	-
G	7	G
H	∞	-

H Yönlendiricisi

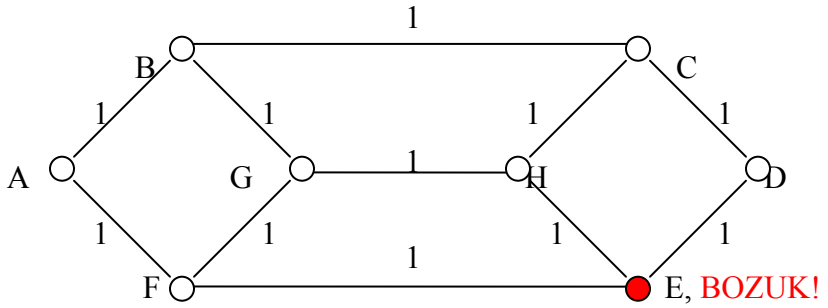
	gecikme	Bir-sonraki düğüm
A	∞	-
B	∞	-
C	10	C
D	∞	-
E	5	E
F	∞	-
G	2	G
H	0	-

G Yönlendiricisi

	gecikme	**Bir-sonraki düğüm**
A	**3+5**	**B**
B	**3**	**B**
C	**3+2**	**B**
D	**-**	**-**
E	**2+5**	**H**
F	**7**	**F**
G	**0**	**-**
H	**2**	**H**

Şekil 6.6 G yönlendiricisindeki tablonun güncellenmesi

Bu teknikte, uzaklık vektörleri *T* milisaniyede bir komşulara gönderildiği için ağ topolojisindeki bir değişikliğin (bir yönlendiricinin bozulması ya da bir hattın kopması) diğer yönlendiricilere ulaşması, bozukluğun olduğu noktanın diğer noktalara olan uzaklığı ile orantılı olarak değişim gösterir. Bilginin tüm düğümlere aynı anda ulaşma imkanı yoktur. Bu da *sonsuza sayma* dediğimiz problemi ortaya çıkarır. Varsayalım ki, ölçütümüz *sekme sayısı* ve ağımızdaki E yönlendiricisi bozuldu (Şekil 6.7).



Şekil 6.7 Ağ üzerindeki E yönlendiricisinin bozulması ve sekme sayısının ölçüt olarak kullanılması durumu

- Bu bilgi ilk aşamada H, D ve F yönlendiricilerine ulaşacaktır. Bu yönlendiriciler tablolarında ilgili alanı sonsuz yaparken G yönlendiricisi H'ye 'ben E'ye 2 sekmede ulaşıyorum' bilgisini içeren vektörünü gönderecektir (doğrudur, ölçüt sekme sayısı olduğu için F üzerinden ulaşıyordur). H bu bilgiye güvenip tablosunu E'ye G üzerinden 3 sekmede ulaşacak şekilde değiştirir.
- Bir sonraki aktarımda F'den E'nin çöktüğü bilgisi G'ye ulaşır. Ancak H, G'nin gönderdiği eski vektöre dayanarak E'ye 3 sekmede ulaşabileceğini bildirecektir. Bu durumda G, H'den gelen vektöre güvenerek tablosunu E'ye 4 sekme ile ulaşacak şekilde günceller.
- Gecikme değeri hatalı olarak bu şekilde artmaya devam eder.

6.1.4 Hat Durumu Yönlendirmesi *(Link State Routing)*

Hat durumu yönlendirmesi de dinamik bir tekniktir. Amacı, toplojideki değişimlere kolayca adapte olmak ve trafikteki değişimlere göre gerektiğinde alternatif yollar bulmaktır. Hat durumu yönlendirmesini, en kısa yol bulma (Dijkstra) algoritmasının uygulaması olarak da düşünebiliriz. Burada, yönlendiriciler bağlı oldukları hatlar (komşuları ile aralarındaki) üzerindeki gecikmeleri gösteren verileri ağdaki tüm yönlendiricilere ulaşması için gönderirler. Bu verileri alan yönlendiriciler ağ topolojisini oluşturur ve diğer yönlendiricilere en kısa yoldan ulaşmak için bu topoloji üzerinde en kısa yol algoritmasını çalıştırırlar. Uygun yolları belirlerler. Hat durumunu gösteren paketler, en kolay taşkın yöntemi ile yayılabilir. Bu durumda paketlerin gereksiz yere ağ içinde dolaşmasını önlemek için hat durumu paketlerinin üzerine *yaş* (sekme sayacı) alanı koymak anlamlıdır.

Ağ üzerindeki trafik zamana bağımlı olarak değişim gösterebilir, yönlendiriciler bu değişimleri yansıtmak için yeni hat durumunu paketleri yaratır ve gönderirler. Bu

durumda yeni paketlerin eskilerinden ayırt edilmesini sağlamak için paketlere sıra numarası verilir. Bir yönlendiriciye ait daha güncel (daha büyük sıra numarasına sahip) bir paket geldiğinde eski paket yok edilir. Sıra numarası ile ilgili olarak karşılaşılacak bir problem sıra numarası bozulmasıdır. Örneğin iletimdeki bir hata sonucu sıra numarası bozulur ve 1 sıra numarası 65'e dönüşürse, aynı düğüm tarafından gönderilen, 2 ile 65 sıra numarası arasındaki tüm hat durumu paketleri geçersiz sayılır. Bu tekniğe ait bir diğer sorun ise: yönlendiricilerden bir bozulup tekrar düzeltildiğinde yarattığı hat durumu paketlerı 0 sıra numarası ile başlar. Ne yazık ki, bu hat durumu paketi daha önce aynı yönlendirici tarafından yaratılmış hat durumu paketlerinden daha düşük (veya eşit) sıra numarasına sahip olduğu için diğer yönlendiriciler tarafından yok edilecektir.
