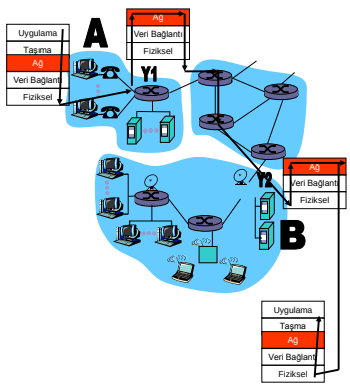
AĞ KATMANI
YÖNLENDİRME (ROUTING)

AĞ KATMANI

- TCP/IP başvuru modelinin üçüncü katmanı
- Bir üstteki Taşıma Katmanına hizmet sunar

Uygulama Katmanı (Application Layer)
Taşıma Katmanı (Transport Layer)
Ağ Katmanı (Network Layer)
Veri Bağlantı Katmanı (Data Link Layer)
Fiziksel Katman (Physical Layer)

AĞ KATMANI



- Ağ katmanı uçtan uca haberleşmenin en alt katmanıdır.
- Ağ katmanı paketlerin kaynaktan alınarak tüm yol boyunca varışa kadar ki durumu ile ilgilenir

AĞ KATMANI - Yönlendirme

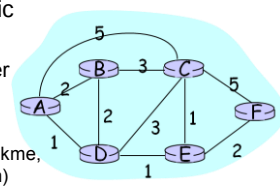
- Ağ katmanının temel fonksiyonlarında biri olan:
  - YÖNLENDİRME (Routing) : Yönlendirme paketlerin kaynaktan varışa kadar olan yolunun tüm ağ boyunca tanımlanması işidir.
- Ağ katmanının bir diğer fonksiyonu:
  - İLETİM (forwarding) İletim ise yönlendiricilerin görevi olup bir paketin yönlendiriciye girdi bağlantı ara yüzünden ilgili çıktı bağlantı ara yüzüne aktarılması işidir.

AĞ KATMANI - Yönlendirme

- Yönlendirme Prensibi:
  - Bir paket için takip etmesi gereken yolu bulmak
  - Bunu ağ katmanının yönlendirme protokolleri sağlar
- Yönlendirme Algoritmaları:
  - Bir paket için takip etmesi gereken "iyi" yolu (good path) bulmak
- İyi yol:
  - Maliyeti düşük (Least-cost) olan yoldur

AĞ KATMANI - Yönlendirme

- Grafik Soyutlama (Graphic Abstraction)
  - Düğümler = Yönlendiriciler
  - Bağlantılar= Fiziksel bağlantılar
    - Bağlantı maliyetleri = gecikme, fiat, tıkanıklık (congestion) seviyesi



AĞ KATMANI - Yönlendirme

- **Yönlendirme Algoritmalarının Sınıflandırılması - 1**
  - Küresel (global) veya Dağıtık (merkezi olmayan) (decentralized)
    - Küresel: Tüm yönlendiriciler ağın tüm topoloji ve bağlantı maliyetleri bilgisine sahiptir
    - Dağıtık: Yönlendiriciler sadece kendilerine direk bağlı olan komşuları ve onlarla aralarındaki bağlantı maliyetleri bilgisine sahiptirler.
      - Komşular arasında bilgi paylaşımı ve tekrarlanan hesaplama süreci gerçekleşir

AĞ KATMANI - Yönlendirme

- **Yönlendirme Algoritmalarının Sınıflandırılması - 2**
  - Statik (static) veya Dinamik (dynamic)
    - Statik: Yönlendirme güncellemeleri zaman içerisinde çok yavaş değişir, genellikle de ağ yöneticileri tarafından elle yapılır
    - Dinamik: Yönlendirme güncellemeleri, yönlendiriciler tarafından otomatik olarak yapılır

AĞ KATMANI - Yönlendirme

- İki temel yönlendirme algoritması
  - Bağlantı Durum Yönlendirme Algoritması (Link State Routing Algorithm)
  - Uzaklık Vektörü Yönlendirme Algoritması (Distance Vector Routing Algorithm)

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 1**
  - Ağ topolojisi ve tüm bağlantı maliyetleri tüm düğümler tarafından bilinmektedir
    - Bu bilgiler "bağlantı durum yayını" (link state broadcast) ile sağlanır
    - Tüm düğümlerde aynı bilgi bulunur
  - Bir düğümden (kaynaktan) diğer tüm düğümlere olan en düşük maliyetli yolu hesaplar
    - O düğüm için "iletim tablosunu" (forwarding table) oluşturur
  - Tekrarlanandır: K kadar tekrarlama sonrasında K tane düğüme kadar olan en düşük maliyetli yol bilgisini hesaplar

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 2**
- Dijkstra Algoritmasının Gösterimi
  - c(x,y): Düğüm x ile y arasındaki bağlantı maliyeti; düğümler birbirlerine direk bağlı değillerse = ∞
  - D(v): Kaynaktan hedef v düğümüne kadar olan yolun güncel maliyeti
  - p(v): Kaynaktan v düğümüne olan en az maliyetli yoldaki bir önceki düğüm
  - N': En az maliyetli yolu kesinlikle bilinen düğümler kümesi

AĞ KATMANI - Yönlendirme

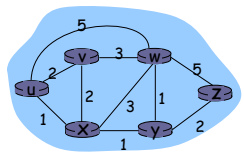
- **Bağlantı Durum Yönlendirme Algoritması – 2**
- Dijkstra Algoritması

```
1  Initialization:
2   N' = {u}
3   for all nodes v
4     if v adjacent to u
5         then D(v) = c(u,v)
6     else D(v) = ∞
7
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12      D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 3**
- Dijkstra Algoritması - Örnek

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1						
2						
3						
4						
5						

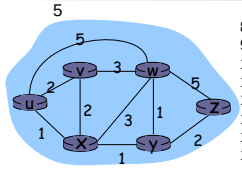



```
1  Initialization:
2    N' = {u}
3    for all nodes v
4      if v adjacent to u
5          then D(v) = c(u,v)
6      else D(v) = ∞
```

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 4**
- Dijkstra Algoritması - Örnek

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2						
3						
4						
5						

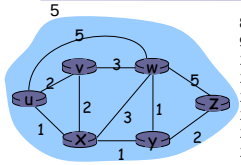



```
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 5**
- Dijkstra Algoritması - Örnek

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3						
4						
5						

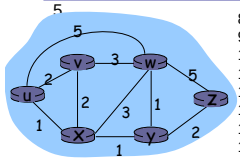



```
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 6**
- Dijkstra Algoritması - Örnek

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4						
5						

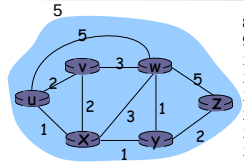



```
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 7**
- Dijkstra Algoritması - Örnek

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5						

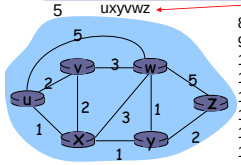



```
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum Yönlendirme Algoritması – 8**
- Dijkstra Algoritması - Örnek

Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					




```
8  Loop
9    find w not in N' such that D(w) is a minimum
10   add w to N'
11   update D(v) for all v adjacent to w and not in N' :
12     D(v) = min( D(v), D(w) + c(w,v) )
13   /* new cost to v is either old cost to v or known
14    shortest path cost to w plus cost from w to v */
15 until all nodes in N'
```

AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 1**
  - Her düğüm periyodik olarak komşularına uzaklık vektörü tahminlerini gönderir
  - Her bir düğüm sadece kendisine direk olarak bağlı komşusundan bilgi alır
  - Herhangi bir düğüm x komşundan yeni bir uzaklık vektörü tahmini aldığında, kendi uzaklık vektörü bilgisini günceller
    - Bellman-Ford Denklemi
  - Daha sonra bu bilgiyi yine komşularına gönderir.
  - Bu süreç tekrarlanan bir şekilde komşular arasında paylaşılacak yeni bilgi kalmayana dek devam eder.

AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 2**
- Bellman-Ford Denklemi

$$d_x(y) = \min \{c(x,v) + d_v(y)\}$$

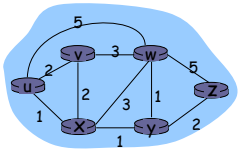
$d_x(y)$ x den y ye en düşük maliyetli yolun maliyeti

$\min \{c(x,v) + d_v(y)\}$ x in komşusuna ve komşunun da y ye olan uzaklıkları toplamlarının minimumunun alınır

AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 3**
- Bellman-Ford Denklemi

$$d_x(y) = \min \{c(x,v) + d_v(y)\}$$



$d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

$$d_u(z) = \min \{ c(u,v) + d_v(z), \; c(u,x) + d_x(z), \; c(u,w) + d_w(z) \}$$

$$= \min \{2 + 5, 1 + 3, 5 + 3\} = 4$$

AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 4**

```
1  Initialization:
2  for all adjacent nodes v:
3     D  (*,v) = infty       /* the * operator means "for all rows" */
4     D  (v,v) = c(X,v)
5  for all destinations, y
6     send min  D  (y,w) to each neighbor  /* w over all X's neighbors */
```

AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 5**

```
8  loop
9   wait (until I see a link cost change to neighbor V
10        or until I receive update from neighbor V)
11
12  if (c(X,V) changes by d)
13    /* change cost to all dest's via neighbor v by d */
14    /* note: d could be positive or negative */
15    for all destinations y:  D  (y,V) =  D  (y,V) + d
16
17  else if (update received from V wrt destination Y)
18    /* shortest path from V to some Y has changed  */
19    /* V has sent a new value for its  min  D^V(Y,w) */
20    /* call this received new value is "newval"    */
21    for the single destination y: D^X(Y,V) = c(X,V) + newval
22
23  if we have a new min  D  (Y,w)for any destination Y
24    send new value of min  D  (Y,w) to all neighbors
25
26  forever
```

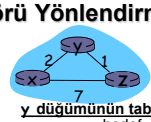
Bekle komşudan bağlantı maliyeti değişikliği bilgisi

Uzaklık tablosunu **hesapla**

Bir hedefe olan en düşük maliyetli yol bilgisinde değişiklik olursa komşulara **haber ver**

AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 6**



zaman

x düğümünün tablosu

kaynak \ hedef	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

y düğümünün tablosu

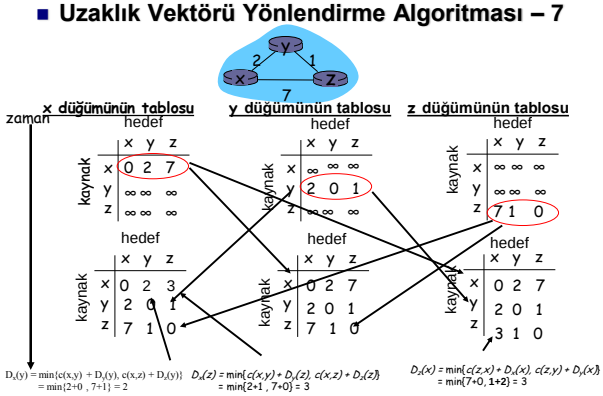
kaynak \ hedef	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

z düğümünün tablosu

kaynak \ hedef	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

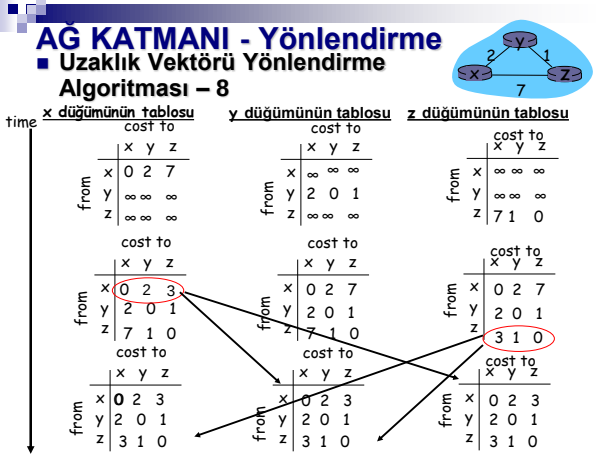
AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 7**



AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 8**



AĞ KATMANI - Yönlendirme

- **Uzaklık Vektörü Yönlendirme Algoritması – 9**
- Bağlantı maliyeti değişiklikleri
  - İyi haber hızlı yayılır
  - Kötü haber yavaş yayılır
    - Bunu engellemek için poisoned reverse tekniği kullanılır (Bağlantı maliyeti artan düğüm komşusuna bağlantı maliyeti olarak ∞ bilgisini gönderir.)

AĞ KATMANI - Yönlendirme

- **Bağlantı Durum ve Uzaklık Vektörü Yönlendirme Algoritmalarının Karşılaştırılması**

	Bağlantı Durum	Uzaklık Vektörü
Mesaj zorluğu	Her düğümün ağdaki tüm bağlantı maliyetini bilmesini gerektirir	Sadece birbirine direk olarak bağlı komşular arasında mesaj değişimi gerektirir
Yakınsama hızı	Çok sayıda mesajlaşma gerektirir.	Yavaş olarak yakınsanır ve yönlendirme döngüleri meydana gelebilir
Sağlamlık	Her düğüm kendi tablosunu hesaplar. Bu da bir miktar sağlamlık sağlar	Her düğümün tablosu başka düğümler tarafından da kullanıldığından hatalar ağ boyunca çoğalır

AĞ KATMANI - Yönlendirme

- **Hiyerarşik Yönlendirme - 1**
- Şimdiye kadar incelediğimiz örnekler ideal durumları göstermekteydi
  - Bütün yönlendiricilerin birbirinin aynı
  - Ağın düz bir yapıda olduğunu varsaydık

Gerçekte büyüklük:
yaklaşık 200 milyon
hedef: Yönlendirme tablolarında bunların tümünün bilgisinin tutulması imkansız! Yönlendirme tablolarının iletilmesi bağlantıları batırır!

İdari özerklik
- internet = ağların ağı
- Her ağ yöneticisi kendi ağının içerisindeki yönlendirmeleri kendine göre kontrol etmek ister

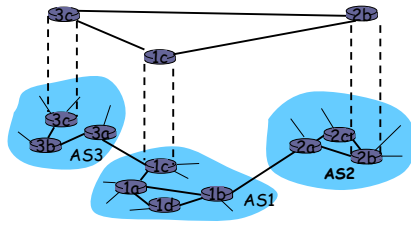
AĞ KATMANI - Yönlendirme

- **Hiyerarşik Yönlendirme – 2**
- Yönlendiriciler Otonom-Özerk Sistem (OS)(autonomous systems - AS) adı verilen bölgelerde kümelendirilirler
- Aynı OS içerisindeki yönlendiriciler aynı yönlendirme protokolünü kullanırlar
  - OS içi yönlendirme protokolü
- Farklı OS ler farklı yönlendirme protokolleri kullanabilirler

AĞ KATMANI - Yönlendirme

- **Hiyerarşik Yönlendirme - 3**

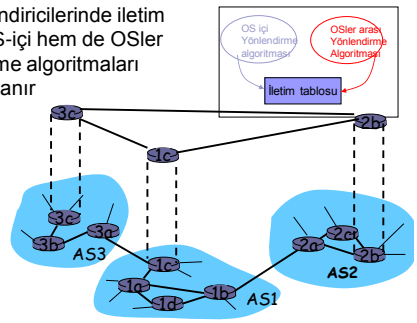
Ağ geçidi yönlendiricileri: Başka bir OS içerisindeki yönlendiricilere direk bağı olan yönlendirici



AĞ KATMANI - Yönlendirme

- **Hiyerarşik Yönlendirme - 3**

Ağ geçidi yönlendiricilerinde iletim tablosu hem OS-içi hem de OSler arası yönlendirme algoritmaları tarafından ayarlanır



AĞ KATMANI - Yönlendirme

- Ağ katmanı
- Yönlendirme
- Yönlendirme Algoritmaları
  - Bağlantı Durum Yönlendirme
  - Uzaklık Vektörü Yönlendirme
  - Hiyerarşik Yönlendirme

Reference: *Kurose, J & Ross, K. (2003). Computer Networking: A top down approach featuring the Internet. Boston, NY: Pearson Education Inc.*