DIJKSTRA VE BELLMAN-FORD EN KISA YOL ALGORİTMALARININ KARŞILAŞTIRILMASI

Özmen Emre DEMİRKOL, Aşkın DEMİRKOL

Özet **– Bu çalışmada bilgisayar ağlarında kullanılan en temel iki algoritmanın, kullanım yöntemleri ve farkları araştırılmıştır. Bu iki algoritmanın, kullanıldığı yerler ve çalışma prensipleri incelenmiştir. Matematiksel çözümler üzerinde örnek uygulamalar ve çözümleri anlatılmıştır. Çalışmamın temel amacı bilgisayar ağları üzerinde uzak noktalar arasındaki iletişimlerde en kısa yolun hesaplanması ve bu hesapların güvenilirliğini ölçmektir. Bu çerçevede tespit edilmiştir ki, Bellman-Ford algoritmasın, özellikle geniş ağlardaki performansının büyük ölçüde tahmine dayalı olması nedeniyle, Dijkstra algoritması daha iyi sonuç vermektedir.**

Anahtar Kelimeler **– En Kısa Yolun Bulunması, Dijkstra Algoritması, Bellman-Ford Algoritması, RIP, OSPF**

Abstract **– In this study, the usage methods and differences of two most basic algorithm in computer networks are explored. The factors while choosing these two algorithms, their usage places and running principals are examined. Sample applications and their solves on mathematical solves is explained. The basic goal of my study is calculating the most shortest way while communicating between far points in computer networks and to measure the reliability of this calculations. In particular, as this handicap of Bellman-Ford on large the nets doesn't appear in Dijkstra's Algorithm, the latter algorithm is an important and has advantages against Bellman-Ford's.**

Keywords **– Finding of The Shortest Path, Dijkstra Algorithm, Bellman-Ford Algorithm, RIP, OSPF**

Ö.E.DEMİRKOL; Sakarya Üniversitesi Bilgi İşlem Daire Başkanlığı
ozmend@sakarya.edu.tr
A.DEMİRKOL; Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü
askind@sakarya.edu.tr

I. GİRİŞ

İnternet dünyası gelişmeye başladığı günden beri en temel problem veriyi en hızlı yoldan hedef noktaya ulaştırmaktır. Temelde küçük bir ağ yapısı içerisinde veri transferi sorun olmaz. Mesafeler ve kaybolan verinin tazelenmesi çok kısa zamanlarda gerçekleştiğinden veri iletim sorunu hissedilmemektedir.

Asıl problem ağ yapısı büyüdükçe ve akan veri miktarı arttıkça ortaya çıkmaktadır[1].

Günümüz ağları oldukça yüksek kapasitede bant genişliğine sahip olmakla beraber, veriyi kaynaktan hedefe taşıma konusunda sıkıntılar yaşamakta ve bunların çözümü için yeni yöntemler geliştirmeye çalışmaktadır.

Bu yöntemlerin temel amacı, istemci ile kaynak nokta arasında bir yol belirlemek ve bu belirlenen yolun zaman ve bant genişliği açısından en uygun olanını tespit etmektir. Önemli olan bu yolu belirlerken işlem sürecini uzatmamak ve yönlendirici cihaz içerisinde veri işlemeyi uzun tutmamaktır. Eğer işlem süreci uzar verilerin tutulduğu alanlar genişlerse, yönlendirici noktada işlemci gücü, disk alanı ve zaman kaybı gibi sorunlar çıkar[1,2,4].

Bu tip kriterler göz önünde tutularak bilgisayar ağlarında birçok kısa yol hesaplama yöntemleri, yönlendiriciler üzerinde koşan algoritmalar geliştirilmiştir.

İnternet altyapısı düşünülürse oldukça karmaşık birçok yol seçeneklerine sahip sonuçlar ortaya çıkmaktadır. Bu karmaşık yapıda bir yol seçmek oldukça zor olduğundan algoritmalar aşırı detaya girmeden çözüme ulaşmak isterler. Çünkü çok fazla parametre olması, hem işlem gücü gerektirir hem de seçim şansını zorlaştırır. Aşırı derecede karşılaştırma işlemi karmaşaya da sebep olacaktır.

Dijksta ve Bellman-Ford algoritmaları kendilerine belirledikleri bazı kriterler ile kısa yolların hesaplanması

konusunda çözümler üretmişler ve bu çözümler ağ cihazları üreticileri tarafından temel alınıp, internet ve bilgisayar ağları dünyasında kabul görmüştür[3,4].

Dijkstra algoritması OSPF (Open Shortest Path First) için bir temel oluşturmuş ve bu algoritmanın tüm kurallarına uyarak yönlendirici cihazlara uygulanmıştır[3,4,5,6,7].

Bellman-Ford algoritması ise RIP (Routing Information Protocol) için temel oluşturmuş ve çoğu özelliği korunarak adapte edilmiştir[8,9,10].

Bilgisayar ağlarında kullanılan en temel iki algoritmanın, kullanım yöntemleri ve farkları araştırıldığı çalışmamız, beş bölümden oluşmuştur.

Temel kavramların ele alındığı Giriş bölümünün yanısıra ikinci bölümde Dijkstra üçüncü bölümde Bellman-Ford algoritmaları karşılaştırmalı olarak ele alınmıştır. Dördüncü bölümde ise elde edilen sonuçlar değerlendirilirken, beşinci bölümde referanslara yer verilmiştir.

II. DIJKSTRA ALGORİTMASI İLE EN KISA YOLUN BULUNMASI

Dijkstra Algoritması kısa yol hesaplarında en çok kullanılan yöntemlerdendir. Sadece bilgisayar ağları değil, karayollarında taşımacılık, posta gibi hizmetlerde de en çabuk şekilde müşteriye veya depoya ulaşmak içinde kullanılmıştır [3,4,5,6,7].

Temel prensip olarak Dijkstra algoritması Link-State (Bağlantı-Durumu)mantığına dayanır. Link-State birçok parametre içerebilir. Örnek olarak length(uzaklık), capacity(kapasite veya bant genişliği), propagation(yayılma), delay(gecikme) gibi temel ölçüler verilebilir. Dijksta bunlardan bant-genişliğini temel alarak algoritmasını geliştirmiştir. Bant genişliği yüksek olan hatlarda özel bir durum sonucu kayıplar olmadıkça düşük kapasiteli hatlardan daha verimlidir. Sonuç olarak aynı miktar veri ele alındığında kullanılan yolun daha geniş olması verinin daha az kayıpla daha kısa sürede hedefine ulaşmasını sağlar.

Dijkstra yönteminde çıkan sonuç yolun Graf'ını verir. Elektrik sistemlerinde de kullanılan Graf yöntemi, bilgisayar ağlarında da kullanılan önemli bir çöztim tekniğidir. Burada ki Graf ağaç mantığındadır.[3,4,5,6,7]

Link-State "Cost[(1)]" olarak isimlendirilir ve matematiksel bir formül ile belirlenir. Cost asla (-) değer almaz.

Öncelikle bu algoritmada kullanacağımız öğeleri tanımlayalım.

$T =$ Ağdaki düğüm(yönlendirici noktaları)lerin oluşturduğu gurup

$S =$ Kaynak düğüm

$n =$ Hedef düğüm

$N =$ Ağda kaynağın her adımda bir sonraki düğüm ile birleşerek oluşturduğu yeni kaynağa verilen ad

$W =$ Hedef nokta olan "n" e gelmeden önce sıra ile üzerinden geçilen düğümlerle S noktası arasında oluşan yolların bileşkesi olan düğümler kümesidir.

$C(i,j) =$ Düğüm i ile düğüm j arasındaki bağlantı değeri. Eğer bir düğüm kaynak düğüm direkt olarak bağlı değilse ilk aşamada bağlantı değeri ∞ olarak kabul edilir.

$D_s(n) = C(S,n)$ kaynak nodu "S" ile hedef düğüm "n" arasındaki *en kısa yol* değeri. $n \in \{T\}$

II.1 Link-State Cost Hesabı

Bu değerin hesaplanması oldukça basit olmakla beraber direkt olarak bant genişliği ile doğru orantılıdır. Link-State şöyle hesaplanır[8];

[(1)] Cost = 100 000 000 / Bant Genişliği bps(bit per second-saniyede akan, bit cinsinden veri miktarı) olarak

Örnek olarak, 10 000 000Mbps lık bir hat kapasitesine sahip kurum $10^9 / 10^8 = 10$ değerindedir. Veya $10^9 / 1544000 = 64$ şeklinde Cost hesaplanabilir.

Anlaşılacağı gibi bölme işleminde virgülden sonraki kalan kısım gözardı edilir. İkinci dikkat edilecek kısım ise değerin düşük olmasıdır. Ne kadar düşük sonuç verirse o kadar kaliteli ve yüksek bant-genişlikli bir hat olduğu anlaşılır[4,6].

Aşağıda örnek bir ağ yapısı üzerinde algoritmanın çalışması incelenmiştir.

II.2 Dijkstra Algoritması

En kısa yolun bulunmasına temel oluşturan Dijkstra Algoritması aşağıdaki adımlardan oluşmaktadır ;

Adım-1 : Grup $N = \{S\}$. Her düğüm için $n \in \{T\}$

$$\text{Grup } D_s(n) = C(S,n)$$

Adım-2 : $W \in \{T - N\}$. D_s(W) algoritma işlenirken hesaplanan kısa yolların toplan değeridir ve daha sonra bu değerin belirlendiği son düğüm olan W, yeni kaynak olarak atanır.

Bu aşamadan sonra yeni kısa yol değeri şu şekilde bulunur.

$$D_s(n) = Min\{D_s(n), D_s(W) + C(W,n)\}$$

If

$$D_s(W) + C(W,n) < D_s(n)$$

Then

$$D_s(n) = D_s(W) + C(W,n)$$

Else

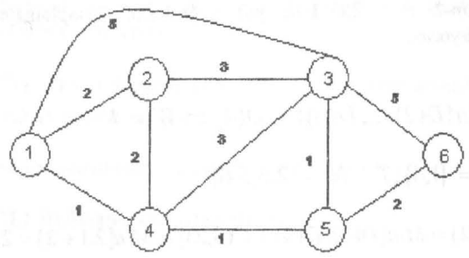
$$D_s(n) = D_s(n)$$

Yani hedef nokta ile kaynak arasında daha önceden bulunan kısa yol değeri kaynak ile yeni bir düğüm arasındaki kısayol değeri, artı ,bu yeni nokta ile hedef arsındaki Cost toplamı, büyük ise yeni kaynak-hedef kısayol değeri, yeni yol üzerinden hesaplanan olarak atanır. Tam tersi bir durum olursa ve 1. durum 2.den küçük olursa eski değer korunur. Yani yol değiştirilmez. Bu işlem her $n \in \{T - N\}$ için uygulanır.

Adım-3 : Adım 2, $N = T$ olana kadar devam ettirilir.

II.3 Örnek Ağ Uygulaması

Bu örnek ağ yapısı bizim algoritmalarımda temel aldığımız örnek bir ağ stili olacaktır. Yapıda 6 düğüm bulunmakta ve birbirlerine değişik yollardan link(bağlantı) kurmaktadırlar. Bu bağlantılarda verilen Link-State Cost değerleri her iki yöne doğru da eşit olarak kabul edilmiş ve tek bir hat çizgisi ile gösterilmiştir. Hatlardaki geliş ve gidiş bant genişliklerinin farklı olması çalışma prensibini etkilemez. Çünkü mantık hedefe doğru sürekli olarak ileri gitme şeklindedir.



Şekil 1. Örnek Ağ Modeli

Şekilde düğümlere birer numara verilmiş ve aralarındaki Cost belirtilmiştir.

Düğümlere verilen numaraların herhangi bir sırası bulunmamakla beraber kaynağa direkt bağlı olanlara numara önceliği verilmiştir.
Yapılacak olan işlem 1 numaralı düğümden diğer bütün düğümlere olan en kısa yolu Dijkstra Algoritması ile bulmaktır.

Şekil 1. deki Dijkstra Algoritmasının çözümü aşağıda verilmiştir.

$C(i,j) =$ i düğümü ile j düğümü arasındaki Cost(bant genişliği)

$D_1(n) = D(n) = C(1,n)$ eşitliği sağlanmaktadır.

Burada kaynak noktası $S = 1$ düğümüdür.

Adım-1 :

$N = \{1\}$

$D(2) = C(1,2) = 2$;
$D(3) = C(1,3) = 5$; $D(4) = C(1,4) = 1$;
$D(5) = C(1,5) = \infty$; $D(6) = C(1,6) = \infty$

Burada görüldüğü gibi başlangıçta N kümesi sadece kaynak düğüm olan 1 i kapsamaktadır. Daha sonra bu küme genişleyecektir. 1 numaralı düğümün diğer düğümlere olan direkt bağlantılarındaki Cost'lar hesaplanmış, 5 ve 6 nolu düğümler direkt bağlantılı

olmadığı için açık devre şeklinde, yani ∞ olarak kabul edilmiştir.

Adım-2 : İlk kısa yol değerlerini hesaplamaya başlıyoruz.

$$Min\{D(2),...,D(6)\} = D(4) \Rightarrow W = 4$$

$$N = \{1,4\}; T - N = \{2,3,5,6\}$$

$$D(2) = Min[D(2), D(4) + C(4,2)] = Min[2,1 + 2] = 2$$

$$D(3) = Min[D(3), D(4) + C(4,3)] = Min[5,1 + 3] = 4$$

$$D(5) = Min[D(5), D(4) + C(4,5)] = Min[\infty,1 + 1] = 2$$

$$D(6) = Min[D(6), D(4) + C(4,6)] = Min[\infty,1 + \infty] = \infty$$

Hesaplamanın ilk adımı T-N sonucu kalan düğümler arasında en küçük Cost değeri vereni bir sonraki kaynak düğüm olarak seçmek ve N kümesine dahil etmektir. T kümesi sabit kalmakta fakat N kümesi sürekli T'nin bir elemanını kendine dahil ederek T ye yaklaşmaktadır.

Daha sonra yeni küme 1,4 olarak belirlenmekte. 4 düğümünün değeri 4, yeni kaynak W olarak atanmaktadır.

Ardından kümenin diğer elemanları için en kısa yol hesabı yapılmaktadır. Bu bir karşılaştırma mantığıdır ve 1 düğümü yokmuş ya da 1 ile 4 birleşip yeni bir düğüm oluşturmuş gibi düşünülerek, 4 üzerinden diğerlerine olan değerlere bakılır. Bu değerler bir önceki aşamada bulunan ve 1 düğümü ile aralarındaki Cost'u belirten değerlerle karşılaştırılır.

Örnek olması için D(2) nin hesap satırını açıklayalım.

D(2) = Karşılaştır ve en küçük değeri seç [D(2) nin şu andaki değeri, D(4) şu andaki değeri + 4 ve 2 düğümleri arasındaki Cost] = karşılaştırma sonucundaki en küçük değer.

Diğer satırlar da aynı şekilde işlemektedir.

Adım-3:

$$Min\{D(2), D(3), D(5), D(6)\} =$$
$$D(2) yada D(5) \Rightarrow W = 2$$

Burada D(2) ve D(5) sonuçları aynı olduğu için ilk önce bulunan algoritma yürütülmeye devam edilir.

$$N = \{1,2,4\}; T - N = \{3,5,6\}$$

$$D(3) = Min[D(3), D(2) + C(2,3)] = Min[4,3 + 2] = 4$$

$$D(5) = Min[D(5), D(2) + C(2,5)] = Min[2,2 + \infty] = 4$$

$$D(6) = Min[D(6), D(2) + C(2,6)] = Min[\infty,2 + \infty] = \infty$$

Adım-4 :

$$Min\{D(3), D(5), D(6)\} = D(5) \Rightarrow W = 5$$

$$N = \{1,2,4,5\}; T - N = \{3,6\}$$

$$D(3) = Min[D(3), D(5) + C(5,3)] = Min[4,2 + 1] = 3$$

$$D(6) = Min[D(6), D(5) + C(5,6)] = Min[\infty,2 + 2] = 4$$

Adım-5 :

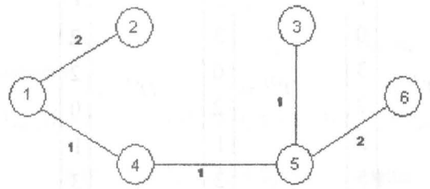
$$Min\{D(3), D(6)\} = D(3) \Rightarrow W = 3$$

$$N = \{1,2,3,4,5\}; T - N = \{6\}$$

$$D(6) = Min[D(6), D(3) + C(3,6)] = Min[4,3 + 5] = 4$$

Bu şekilde algoritmamız tamamlanmış olmaktadır.
6. düğüm için analiz yapmaya gerek yoktur. Çünkü son düğüm için hesap yapılsa bile daha önce hesaplananlardan daha küçük bir değer çıkmayacaktır.

Son durumdaki ağımızın son hali şu şekilde olmaktadır.



Şekil 2. Ağın Algoritma Sonucu Oluşan Graf'ı

Tablo olarak bakılırsa su şekilde olacaktır.

Tablo 1. Ağın İz Tablosu

Düğüm	1	2	3	4	5	6
D(n)	0	2	3	1	2	4
W	1	1	5	1	4	5

Tabloda ilk satır düğümlerin adlarını sıra ile göstermektedir.
İkinci satırda 1. düğüm ile diğer düğümler arasında hesaplanan en kısa Cost'u belirtir.

Üçüncü satır ise düğümlere varmadan bir önce hangi düğüm üzerinden geçileceğini göstermektedir.

Bu tabloya bakılarak hedeften geriye doğru gelinir ve kaynaktan ilk olarak hangi noktaya bilginin iletileceğine bakılır. Son olarak paket gerekli bilgilerle birlikte bir sonraki düğüme yollanır.

Bu algoritma daha öncede belirtildiği gibi OSPF'nin temelini oluşturmaktadır. OSPF paket bilgilerini göndermeden oluşturulan link-state durumu bilgisini kontrol eder ve bu özel bir bağlantı ile sağlanır. Yönlendiriciler arası akan bu özel bilgi içerisinde paket tanımlamaları, yetki verme mekanizması, bağlantı durum güncellemesi, veri kontrol bilgisi gibi bilgiler bulunur.

III. BELLMAN-FORD ALGORİTMASI İLE EN KISA YOLUN BULUNMASI

Bu algoritmanın temel prensibi ise, bir tek kaynak nokta yerine tüm noktaları birer kaynak gibi düşünerek bunlar üzerinden bağlantıları hesaplamaya dayanır.

$$D_{(i)yeni} = \min[C_{ij} + D_{(ij)alc}]$$

$D_{(ij)alc}$ = Komşu düğümlerden alınan uzaklık hesabı

Buradan da anlaşılacağı gibi Bellman-Ford Uzaklık-Vektör mantığı ile çalışır.

$C(i, j)$ = i düğümü ile komşusu olan j nodu arasındaki uzaklık hesabı.

Algoritmanın çalışması aşağıda verilmiştir.

III.1 Bellman-Ford Algoritması

Her nokta ile komşuları arasındaki Cost belirlenir ve birer matris olarak yazılır. Daha sonra bu matrisler arasında yapılan karşılaştırma sonucunda en kısa yollar hesaplanır.

Her i düğümü iki vektörün izini tutar.

$$D^{(i)} = [D_1(i)....D_N(i)] \; ve \; S^{(i)} = [S_1(i)....S_N(i)]$$

$D^{(i)}$ = Düğüm i nin ağdaki diğer düğümler ile arasındaki minimum gecikmenin hesabı

$S^{(i)}$ = Her hedef düğüm için en iyi bağlantı yaptığı düğüm

N = Ağdaki düğümlerin numaraları

k = Kaynağa direkt bağlı veya en fazla 1 atlamayla ulaşılabilecek düğümlerin kümesi

Uzaklık-Vektör bilgisi her düğüm için yaklaşık 2/3 saniyede bir komşu yönlendiricilerden alınan bilgi ile yenilenir.

Düğüm i nin Uzaklık-Vektörü aşağıdaki formül kullanılarak düzenlenir.

$$D_j(i) = \min_{k \in N(i)}[D_k(i) + D_j(k)]$$

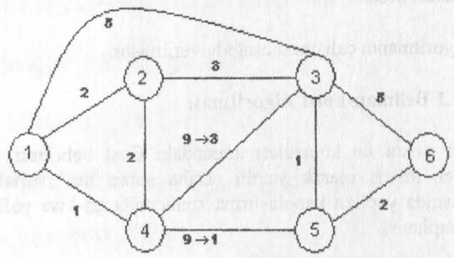
$N(i)$ = Düğüm i nin komşularının oluşturduğu gurup

$D_j(i)$ = Düğüm ile j arasındaki gecikmenin kesin hesabı

III.2 Örnek Ağ Uygulaması

Daha önceki ağı, karşılaştırmanın daha kolay olabilmesi için tekrar kullanıyoruz.

Burada belirlenen Cost'lar Dijkstra Algoritması'ndakilerle aynı prensiple hesaplanmış bu nedenle de aynı sonucu vermiştir. Bazı ufak farklar bulunmakla beraber nedeni yapıya zenginlik kazandırmaktır.



Şekil 3. Örnek Ağ Yapısı

Burada görüldüğü üzere 4 - 3 ve 4 - 5 düğümleri arasındaki değerler henüz yenilenmiş ve 9 dan 1 e ve 9 dan 3 e düşmüştür.

1. düğümün yönlendirme tablosu ilk olarak aşağıdaki gibi oluşmuştur.

Tablo 2. Ağın İlk İz Tablosu

Hedef	Gecikme ($D^{(0)}$)	Sonraki Düğüm ($S^{(0)}$)
1	0	-
2	2	2
3	5	3
4	1	4
5	6	3
6	8	3

Bu aşamadan sonra 1. düğümün komşusu olan her düğüm kendisine kendi bağlantı durumu ile ilgili bilgileri gönderir.

1. düğüm bu bilgileri alarak kendisindeki bağlantı bilgilerini güncelleştirir.

$$D^{(2)} = \begin{bmatrix} 2 \\ 0 \\ 3 \\ 2 \\ 3 \\ 5 \end{bmatrix} \quad D^{(3)} = \begin{bmatrix} 3 \\ 3 \\ 0 \\ 2 \\ 1 \\ 3 \end{bmatrix} \quad D^{(4)} = \begin{bmatrix} 1 \\ 2 \\ 2 \\ 0 \\ 1 \\ 3 \end{bmatrix}$$

Görüldüğü gibi burada düğüm 5 ve düğüm 6 ile ilgili güncelleme bilgileri bulunmamaktadır.

Bellman-Ford algoritmasının temeli hop count yani atlama sayısı mantığında yatmaktadır. Bellman-Ford algoritmasında eğer mümkünse hedefe direkt varılmak istenir. Buraya kadar Dijkstra ile aynı yapıda olup buradan sonra ayrılırlar. Bellman-Ford eğer hedefe direkt ulaşamıyorsa kendisine güncelleme yapan yönlendiriciler üzerinden(bunlar da kaynak düğüme direkt bağlıdır) 1 atlama yaparak hedefe ulaşmaya çalışır. Bu nedenle diğer düğümlerdeki bilgilere gerek duymaz.

Düğüm 1 aşağıdaki ifadeyi kullanarak kısa yol hesabı yapar.

$$D_j(1) = \min_{k \in N(i)} [D_k(1) + D_j(k)]$$

Bu ifadenin anlamı şudur;

1 düğümü ile j düğümü arasındaki en kısa değer; $k = 1$ den başlamak üzere k ile 1. düğüm arasındaki Cost, artı, j düğümü ile k düğümü arasındaki Cost.

Bu toplama işlemi k = n olana kadar devam eder ve sonuçta birde fazla toplam değeri ortaya çıkar. Bu aşamadan sonra yapılması gereken, bulunan değerler arasında en küçük olanı yeni değer olarak atamaktır.

1 düğüm için bu değer her zaman 0 olur.

$$D_{(1)yeni}(1) = 0$$

$$D_{(2)yeni}(1) = \min[D_2(1) + D_2(2), D_3(1) + D_2(3), D_4(1) + D_2(4)]$$

$$= \min[2+0,5+3,1+2] = 2, \quad \text{düğüm} \quad 2$$

üzerinden

$$D_{(3)yeni}(1) = \min[D_2(1) + D_3(2), D_3(1) + D_3(3), D_4(1) + D_3(4)]$$

$$= \min[2+3,5+0,1+2] = 3, \quad \text{düğüm} \quad 4$$

üzerinden

$$D_{(4)yeni}(1) = \min[D_2(1) + D_4(2), D_3(1) + D_4(3), D_4(1) + D_4(4)]$$

$$= \min[2+3,5+2,1+0] = 2, \quad \text{düğüm} \quad 4$$

üzerinden

$$D_{(5)yeni}(1) = \min[D_2(1) + D_5(2), D_3(1) + D_5(3), D_4(1) + D_5(4)]$$

$$= \min[2+3,5+1,1+1] = 2, \quad \text{düğüm} \quad 4$$

üzerinden

Aynı yöntem kullanılarak, düğüm 4 üzerinden yapılan yeni hesaplarla, düğüm 6'ya olan en kısa yol hesaplanabilir.

$D_{(6)yeni}$ = 4, düğüm 4 üzerinden

Düğüm 1 in yeni yönlendirme tablosu şu şekilde oluşur.

Tablo 3. Ağın Son İz Tablosu

Hedef	Gecikme ($D^{(0)}$)	Sonraki Düğüm ($S^{(0)}$)
1	0	-
2	2	2
3	3	4
4	1	4
5	2	4
6	4	4

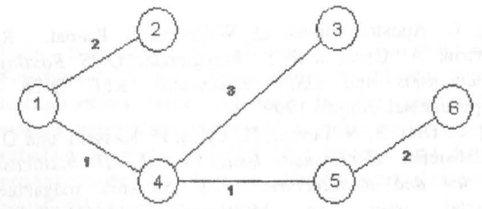
OSPF de olduğu gibi sistemin işleyişi öncesi birçok bilgi iletişimi yapılarak ortam hazırlanır.
Bir yönlendirici hem OSPF hem RIP çalışabilir. Bunun için birden fazla çıkış ve bu çıkışlara özel ayar yapılmalıdır. Bir başka tespit ise, port bazında yapılan ayarlamalar sonucunda bir kısım yönlendirmelerin OSPF, bir kısım yönlendirmeler ise RIP şeklinde olduğudur.

IV. SONUÇ

Çalışmamızda, Bellman-Ford algoritmasından elde edilen sonuçlar, Disjkstra da elde edilenler ile yaklaşık(küçük farklar dışında) aynı değerleri vermektedir.

Bilgisayar ağlarında yoğun bilgi, fazla işlem ve karmaşaya sebep olmaktadır. Bu da, Bellman-Ford'u anlaşılabilirliği ve basitliği açısından daha ön plana çıkarmaktadır.

Buna göre oluşan yeni Graf aşağıdaki gibi olacaktır.



Şekil 4. Ağın Algoritma Sonucu Oluşan Graf'ı

Bellman-Ford Algoritması RIP'in temelini oluşturmaktadır. RIP'te en fazla atlama sayısı ilk değer olarak 15 şeklinde atanmıştır. Değiştirilebilir olmakla beraber dengesizlikler çıkarması mümkün olabilir. Ayrıca diğer yönlendiriciler de aynı şekilde yapılandırılmalıdır.

Bunun yanı sıra Dijkstra Algoritması Bellman-Ford'a göre daha doğru sonuçlar vermektedir. Bulunan değerler kesin ve değişmezdir. Bu değerler sadece hat kapasitelerinin değişimi sonucunda farklılık gösterir. Bellman-Ford da ise daha çok tahmini bir durum söz konusudur. Belirtilen atlama noktasından sonraki kısımlar için tahmini yönlendirme yapılmaktadır. Bu da dengesizlik yaratarak, bu yöntemin etkinliğini azaltmaktadır.

Şekil 2 ve 4 de görüldüğü gibi, Dijkstra için 1 numaralı kaynaktan 3 numaralı hedefe gitmek için sırası ile 1 - 4 - 5 - 3 numaralı düğümlerden geçilmelidir. Bu yol üzerindeki $Cost = 1+1+1 = 3$ olarak bulunmaktadır.

Fakat Bellman-Ford algoritması uygulandığında, izlenecek yok 1 - 4 - 3 sırasında oluşmuştur. Buna bağlı olarak $Cost = 1+3 = 4$ şeklinde çıkmıştır.

Bu verilerden, Dijkstra Algoritması'nın en kısa yol hesaplarında, kesinlikle daha iyi sonuç verdiği, ayrıca gelişen ve daha da güçlenen ağ cihazlarının günümüzde artması sonucu, Bellman-Ford Algoritması'nın Dijkstra'a kıyasla mevcut handikaplarıyla daha az tercih edilmesi gerektiği tespit edilmiştir. Bu yapısıyla Bellman-Ford'un ancak çok klasik ve kapasitesi düşük ağ cihazlarında tercih edilebileceği görülmüştür.

KAYNAKLAR

[1]. G. Apostolopoulos, D. Williams, S. Kamat, , R. Guerin, A. Orda, and T. Przygienda. *QOS Routing Mechanisms and OSPF Extensions*. RFC 2676 - Experimental, August 1999

[2]. C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. *Deployment issues for the IP multicast service and architecture*. IEEE Network magazine special issue on Multicasting, 14(1):78--88, January/February 2000

[3]. N. M. Malouch, Z. Liu, D. Rubenstein, and S. Sahu. *A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted*, End-System Multicast. In 12th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'02), May 2002. 143

[4]. G. Apostolopoulos, R. Guerin, and S. Kamat, "*Implementation and Performance Measurements of QoS Routing Extensions to OSPF*," in Proc. of IEEE Infocom, March 1999

[5]. Y. Breitbart, M. Garofalakis, A. Kumar and R. Rastogi, " *Optimal Configuration of OSPF Aggregates*" , In Proc. of IEEE INFOCOM2002

[6]. Moy, J.; "*The OSPF Specification*," Draft RFC, Oct. 89

[7]. Dirceu Cavendish and Mario Gerla. *Internet QoS Routing using the Bellman-Ford Algorithm*. In IFIP Conference on High Performance Networking, 1998

[8]. Xin Yuan, "*On the extended bellman-ford algorithm to solve twoconstrained quality of service routing problems*," in International Conference on Computer Communications and Networks(ICCN'99), Oct. 1999

[9]. Q. Ma, P. Steenkiste, "*Routing Traffic with Quality-of-Service Guarantees in Integrated Services Networks*", In 8th IEEE/ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'98), England, July 1998

[10]. Chowdhury A., Luse P., Frieder O., Wan P., "*Network Survivability Simulation of the Commercially Deployed Dynamic Routing System Protocol*", IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems, May 2000