

Bölüm 7: **ÇOĞAGÖNDERİM YÖNLENDİRMESİ,** **GEZGİN DÜĞÜMLER İÇİN YÖNLENDİRME, ve** **YAPISIZ AĞLARDA YÖNLENDİRME**

Türkçe (İngilizce) karşılıklar

Çoğagönderim (multicast)
Yayın, Tümeğönderim (broadcast)
Kapsayan ağaç (spanning tree)
Merkez temelli ağaç (core-based tree)
Seyyar (portable)
Gezgin (mobile)
Sabit (stationary)
Yapısız ağ (ad-hoc network)

Bölüm Hedefi

Bu bölümü bitirdiğinizde

- Kapsayan ağaç oluşturmayı,
- Çoğagönderim yönlendirmeyi,
- Gezgin düğümler için yönlendirmeyi,
- Plansız ağlar, ve
- Plansız ağlarda yönlendirmeyi

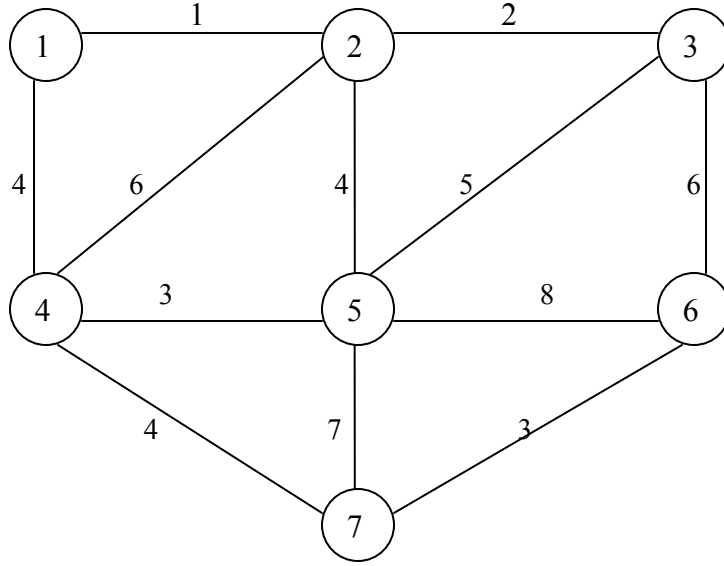
öğrenmiş olacaksınız.

Bu bölümün hedefi Bölüm 6'da incelenen yönlendirme algoritmalarının ötesinde, özel durumlarda kullanılacak yönlendirme teknikleri/algoritmaları üzerinde durmaktır. İncelenecek özel durumlar: çoğagönderim, gezgin düğümler ve yapısız (ad-hoc) ağlar için yönlendirme olacak.

7.1 Kapsayan Ağaç Oluşturma

Kapsayan ağaç, bir çizge (ağın topolojik yapısı olarak düşünebiliriz) üzerindeki tüm düğümleri içeren ve *düğüm sayısı-1* kenardan oluşan altçizgedir. Tüm düğümleri içermesi nedeniyle kapsayan ağaç olarak adlandırılır. Kenarlar çif yönlü bağlantıları gösterdiği ve yapı halka içermediği için bu ağaçlarda herhangi iki düğüm arasında sadece tek bir yol bulunur.

Bu bölümde Kruskal tarafından geliştirilen *en küçük kapsayan ağaç* algoritması açıklanacaktır. Kapsayan ağacın *en küçük* olarak adlandırılmasının nedeni kenarlar seçilirken ağırlığı en kısa kenara öncelik verilmesindedir. Eğer kenar uzunluklarının bir önemi yoksa, kenar uzunluklarını gösteren L matrisinde tüm kenarların uzunlukları 1 olarak gösterilebilir. Kruskal algoritması aşağıdaki gibidir:

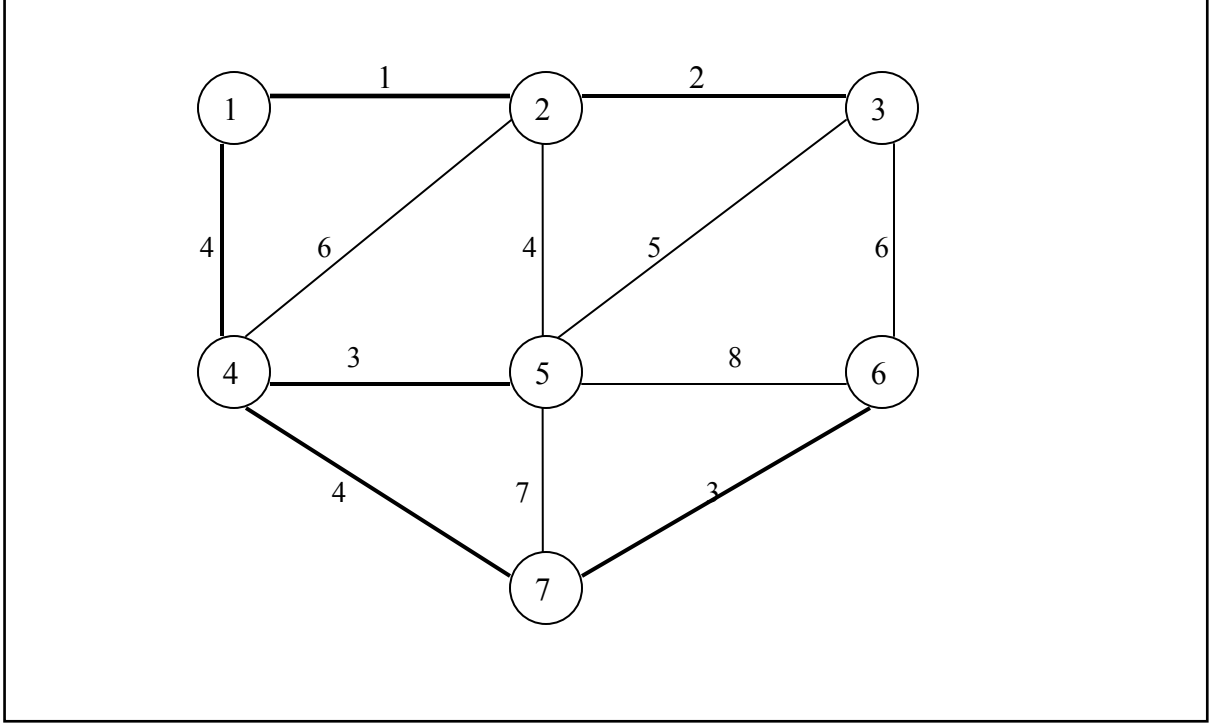


Şekil 7.1 En küçük kapsayan ağaç algoritması için topoloji örneği

Şekil 7.1’de verilen topolojiye Kruskal algoritması uygulandığında aşağıdaki adımlar gerçekleşir:

1. adım **(1, 2)** kenarı seçilir
 - 2.adım **(2,3)** kenarı seçilir
 3. adım (4,5) ya da (6,7) kenarları eşit uzunlukta, biri rastgele seçilir. Diyelim ki **(4,5)** kenarı seçilir
 4. adım **(6,7)** kenarı seçilir
 5. adım (1,4) ya da (4,7) ya da (2,5) kenarı eşit uzunlukta biri rastgele seçilir. Diyelim ki, **(1,4)** kenarı seçilir
 6. adım (4,7) ve (2,5) kenarları aynı uzunlukta, biri rastgele seçilir. Diyelim ki, **(2,5)** kenarı seçilir ancak iki düğüm de aynı kümede olduğu için bu kenar *kabul edilmez.*
 - 7.adım **(4,7)** kenarı seçilir
- 6 kenar seçildiği için işlem tamamlanır.**

Seçilen kenarlar sonrasında oluşan kapsayan ağaç Şekil 7.2’de kalın kenarlarla gösterilmiştir.

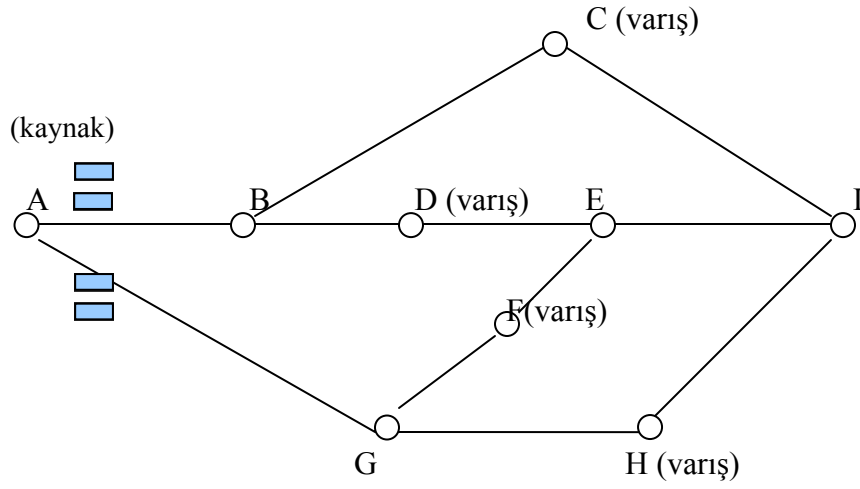


Şekil 7.2 Örnek topoloji üzerindeki ek küçük kapsayan ağaç

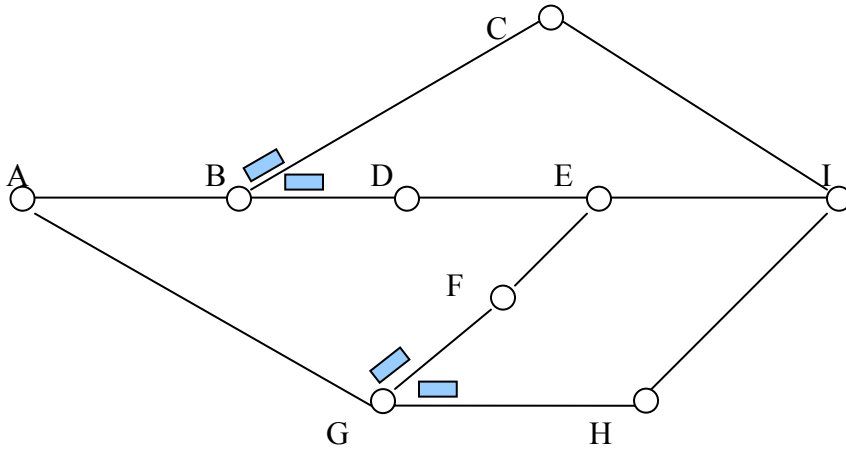
7.2 Çoğagönderim Yönlendirme

Tümeğönderim ve *çoğagönderim* bilgisayar haberleşmesinde sıkça karşılaşılan durumlardır. *Tümeğönderim* de hedeflenen bir paketin/verinin bir ağdaki/altağdaki *tüm* bilgisayarlara iletilmesidir. *Çoğagönderim* ise bir paketin/verinin *belli bir grup* bilgisayara iletilmesidir. *Tümeğönderim* tüm bilgisayarları hedef aldığı için gerçekleşmesi daha kolaydır.

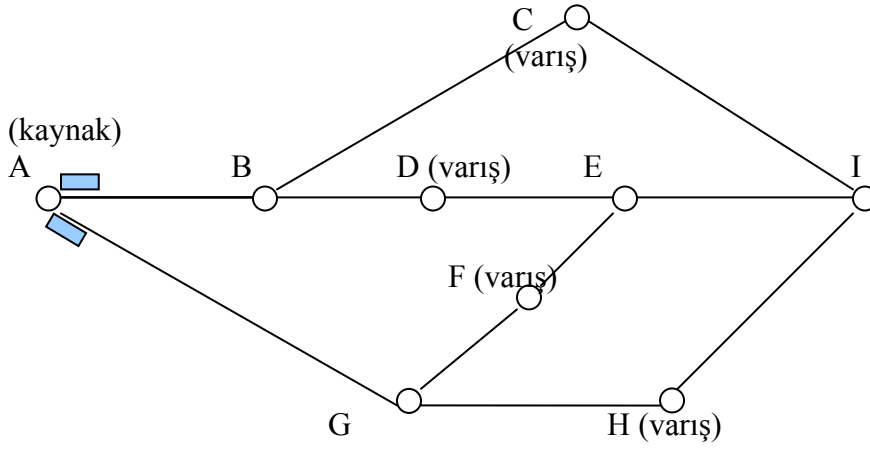
Verinin birden fazla bilgisayara iletilmesi durumunda kaçınılması gereken en önemli nokta aynı verinin aynı hatlar üzerinden tekrar tekrar aktarılmasıdır. Bu durum ağ kaynaklarının verimsiz kullanımına yol açar. Yapılması gereken: kaynak ve varış noktaları arasında ortak yollar oldukça, bu yollar üzerinde aynı verinin tek bir kez aktarılmasını ve gerek duyuldukça verinin kopyalanmasını sağlamaktır. Bu durum Şekil 7.3'de açıklanmıştır.



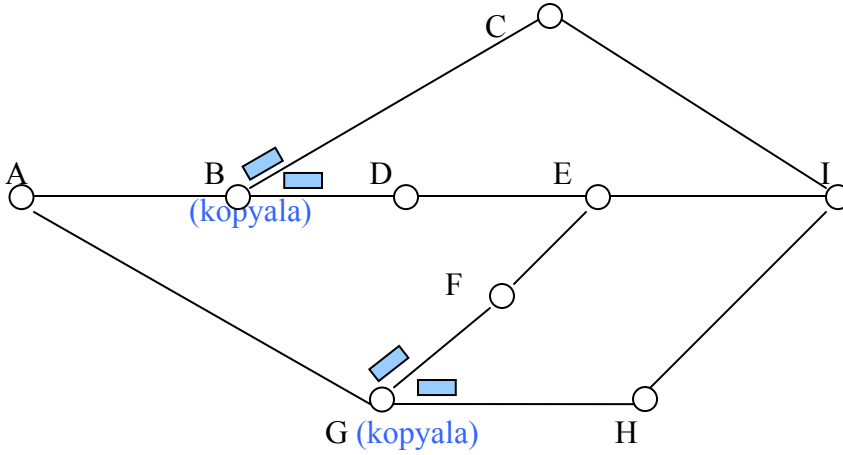
(a) Grubun her elemanı için bir paket yaratma durumu (1. sekme)



(b) Grubun her elemanı için bir paket yaratma durumu (2. sekme)



(c) Çoğagönderim ağacı üzerinde en az paket yaratma durumu (1. sekme)



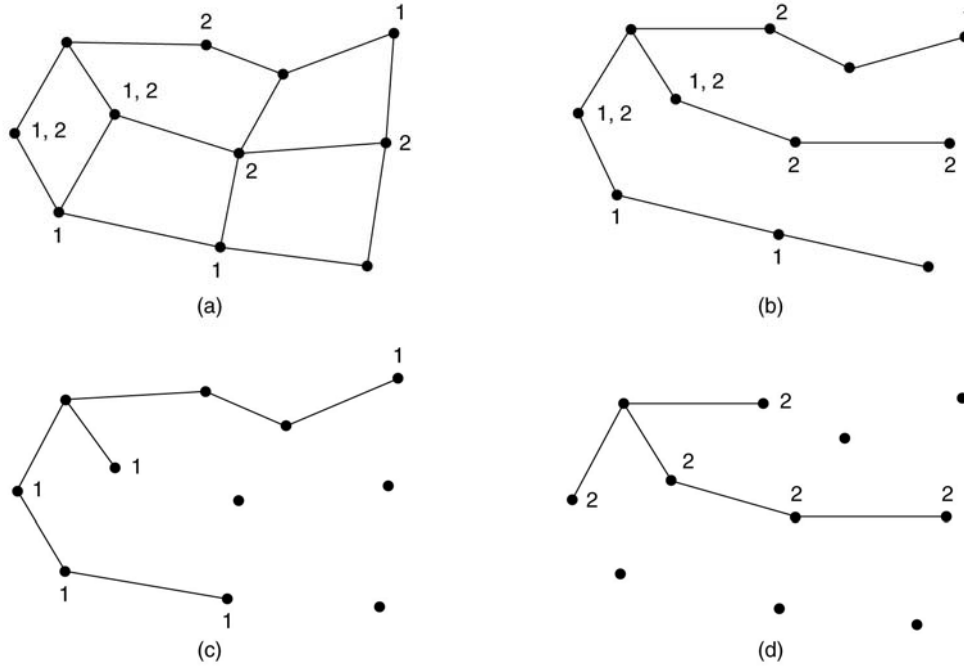
(d) Çoğagönderim ağacı üzerinde en az paket yaratma durumu (kopyalama ve 2. sekme)

Şekil 7.3 Çoğagönderim paketlerinin iletimi (iki durum için)

Çoğagönderim, grup yönetimi gerektirir. Verinin iletileceği bilgisayarlar/düğüm/ kullanıcılar bir grup oluşturur. Gruba yeni elemanlar katılması ya da varolan bir elemanın ayrılması durumunda gerekli güncellemelerin yapılması gerekir. Yönlendiriciler de kendilerine bağlı düğümlerin hangi gruplarda yer aldıklarını bilmelidirler. Bu işlem düğümlerin yönlendiricileri bilgilendirmesi ya da yönlendiricilerin periyodik olarak

sorgulaması ile gerçekleşir. Yönlendiriciler tarafından elde edilen bilgiler komşu yönlendiricilerle paylaşılır. Böylece gruplara ait bilgiler ağ içinde yayılır.

Çoğagönderim yönlendirmesi yapmak için her yönlendirici diğer yönlendiricileri içeren bir kapsayan ağaç oluşturur. Daha sonra belli bir gruba bu kapsayan ağaç üzerinden erişmek için gerekli olmayan tüm yollar budanır ve kullanılacak yollar bulunur. Örneğin Şekil 7.4'te gösterilen ağ üzerinde iki çoğagönderim grubu olsun (grup 1 ve grup 2). Şekildeki ağ üzerinde ilgili yönlendiriciler 1 ve 2 ile belirtilmiştir (Şekil 7.4(a)). Kaynak düğümün en üst sol köşedeki düğüm olduğunu düşünürsek, diğer düğümlerin altısı birinci çoğagönderim grubu ile, beşi ikinci çoğagönderim grubu ile ilgilidir. Bu durumda kaynak düğümünden her iki grubun yönlendiricilerini içeren kapsayan ağaç Şekil 7.4(b)'de, sadece birinci grubun yönlendiricilerini içeren kapsayan ağaç Şekil 7.4(c)'de, sadece ikinci grubun yönlendiricilerini kapsayan ağaç ise Şekil 7.4(d)'de verilmiştir.



Şekil 7.4 İki çoğagönderim grubuna sahip ağ. [1]'den alınmıştır.

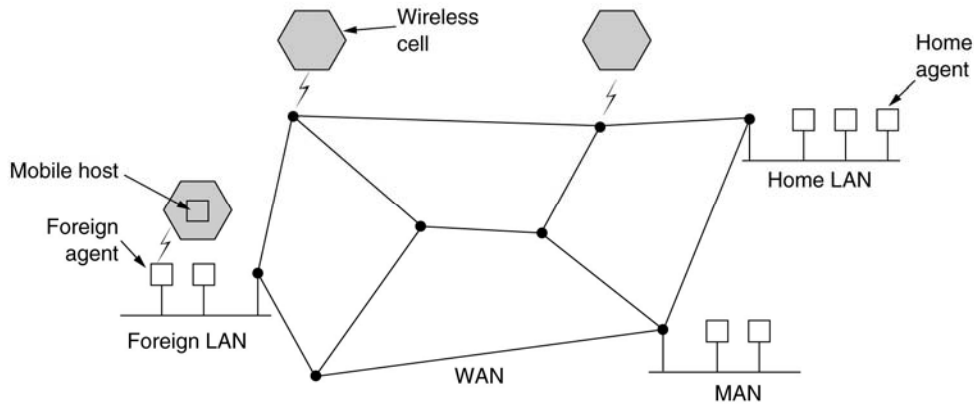
Kapsayan ağaçlar tüm düğümleri içerir oysa ki biz, her bir çoğagönderim grubu için bir kapsayan ağaç oluşturmak ve bu ağacında grubun üyelerini içermesini istiyoruz. Bu durumda gereksiz kenarların kesilerek (bu işleme *budama* denir) ağacın küçültülmesi anlamlıdır. Kapsayan ağacın *budanması* (*pruning*) için pek çok yöntem kullanılabilir. Bunların en basiti hat durumu yönlendirmesinde görülür. Çünkü bu teknik her düğümde tüm topolojinin yaratılmasını sağlar. Budama işlemi kaynağa uzak düğümlerden kaynağa doğru yapılır. Dikkat edilmesi gereken bir diğer nokta ise bazı düğümlerin çoğagönderim grupları içinde yer almamasına karşın grubun uzaktaki elemanlarına ulaşmak için kapsayan ağaca dahil edilmesidir. Şekil 7.4(c)'de görüldüğü gibi.

Uzaklık vektörü yönlendirmesinde ise kapsayan ağacı budamak için dağıtılmış bir teknik kullanılır. Bu teknikte herhangi bir yönlendirici kendisinin ya da komşularının yer almadığı bir çoğagönderim grubuna ait bir mesaj aldığı anda o mesajı gönderen yönlendiriciye *budama* mesajı gönderir. Budama mesajını alan yönlendirici o mesajı gönderen yönlendiriciye ait hattı budar.

Her yönlendiricinin, her grup için bir kapsayan ağaç oluşturması durumunda, ağdaki düğüm ve grup sayısına bağlı olarak gerekli bellek kapasitesi ciddi bir şekilde artabilir. Bu sorunu çözmek için, *merkez temelli ağaç (core-based tree)* tekniği önerilmiştir. Bu teknik ile her çoğagönderim grubu için tek bir ağaç oluşturulur. Bu ağacın *kökü (merkezi)* olarak, gruptaki tüm yönlendiricilere eşit uzaklıkta bir yönlendirici seçilir. Her zaman bu kriterin tam olarak sağlanması mümkün olmayabilir, ancak önemli olan merkez yönlendirici ile diğer yönlendiriciler arasında dengeli bir uzaklığın olmasıdır. Gruba mesaj göndermek isteyen yönlendirici, mesajını merkez yönlendiriciye gönderir ve mesaj merkez yönlendiriciden diğer yönlendiricilere yayılır. Bu durumda, ağ üzerindeki her bir çoğagönderim grubu için tek bir ağaç oluşturulur.

7.3 Gezgin (Mobile) Düğümler için Yönlendirme

Gelişen teknoloji ile pek çok insan seyyar (portable) bilgisayarları ile dolaşmakta ve gittikleri yerlerden kendilerine gelen e-posta mesajlarını okumak, kendi dosyalarına erişmek istemektedir. Günümüzde bu, bir sunucuya bağlanmak şeklinde gerçekleşmektedir. Üzerinde çalışılan konulardan biri de benzeri durumlarda mesajların doğrudan ilgili bilgisayara iletilmesi içindir. Cep telefonlarında olduğu gibi. Bu durum için örnek bir ortam Şekil 7.5'te verilmiştir.



Şekil 7.5 Yerel alan ağlarının, telsiz ağların bağlı olduğu bir geniş alan ağı. [1]'den alınmıştır.

Hiç hareket etmeyen bilgisayarları *sabit (stationary)* olarak adlandırıyoruz. Gezgin bilgisayarları ise iki grupta toplayabiliriz. Bunların biri *sistem değiştiren (migratory)*

bilgisayarlar grubu, diğeri ise *dolaşan (roaming)* bilgisayarlar grubudur. Sistem değiştiren gezgin bilgisayarlar, gidilen yerdeki bilgisayar ağını kullanırlar ve kullanım sırasında hareketli değildirler. Dolaşan gezgin bilgisayarlar ise, devamlı hareket halindedir ve bu sırada haberleşme ağına erişmek ihtiyacındadırlar.

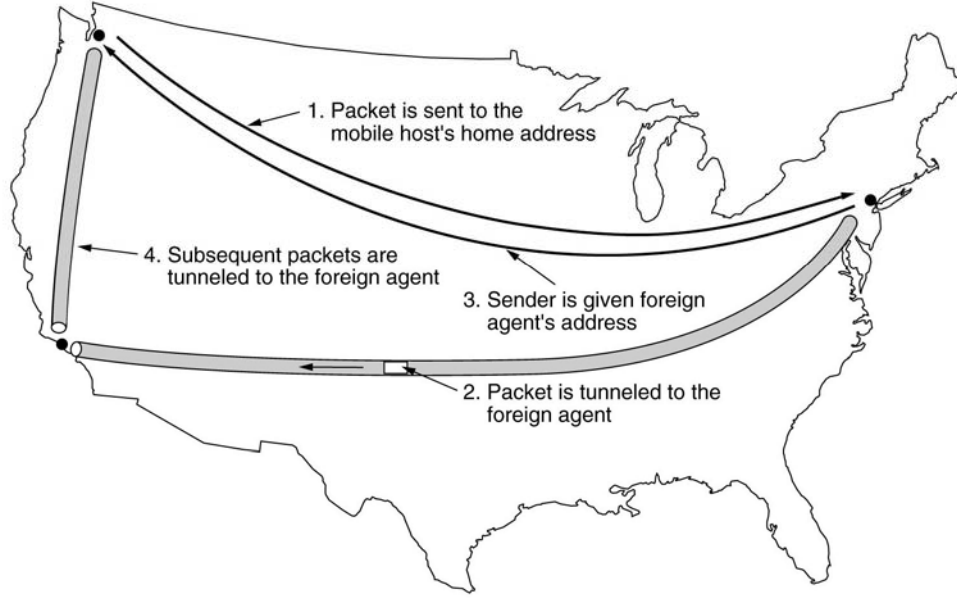
Bu yaklaşımda, tüm bilgisayarların değişmez bir adresi, *ev yeri (home location)* olduğu kabul edilmektedir. Telefon numaralarımız ev yeri için iyi bir örnek oluşturur.

Şekil 7.5'teki gibi dünyanın coğrafi olarak küçük birimlere (alanlara) bölündüğünü düşünebiliriz. Bu birimler yerel alan ağları, telsiz ağların kapsadığı alanlar, ... olabilir. Bu alanlara kayıtlı, yani orayı ev yeri olarak kullanan bilgisayarlar (düğümler) olduğu gibi, bu alanlarda dolaşan *ziyaretçi (visiting)* bilgisayarlar da olabilir. Bu alanlarda ziyaretçi bilgisayarlarla ilgili bir ya da birden fazla *yabancı ajan (foreign agent)* bulunur. Bu ajanların görevi ziyaretçi bilgisayarları algılamak ve onlarla ilgili işlemleri yapmaktır. Benzer şekilde, bir alana ait bilgisayarların diğer alanlardaki varlıkları ile ilgili kayıtları tutan *ev ajanları (home agents)* da vardır.

Yeni bir bilgisayar (birim) ev alanından farklı bir alana girerse öncelikle o alana kaydolmalıdır. Kayıt işlemi aşağıdaki gibidir:

- 1) Periyodik olarak her yabancı ajani kendi alanına giren ziyaretçi bilgisayarları belirlemek için paket yayar. Bazı durumlarda, ortamda yabancı ajani olup olmadığı ziyaretçi bilgisayarlar tarafından da sorgulanabilir
- 2) Ziyaretçi bilgisayar kendini yabancı ajanına kaydettirir. Adresini, ortama erişim için gerekli verileri ve güvenlik bilgilerini aktarır.
- 3) Yabancı ajani, ziyaretçi bilgisayarın ev ajani ile bağlantı kurar ve onu ziyaretten haberdar eder. Bu bilgileri güvenilir kılmak için gerekli güvenlik kodlarını ekler.
- 4)İlgili ev ajani, gelen verileri inceler, uygun görürse yabancı ajanın işleme devam etmesine izin verir.
- 5) Yabancı ajani gerekli izni alınca ziyaretçi bilgisayarı kendi tablosuna kaydeder.

Gezgin bilgisayarlara gönderilmek istenen veri öncelikle onların ev adreslerine gider. Şekil 7.6'da bu konuyu açıklayan bir örnek verilmiştir.



Şekil 7.6 Gezgin bilgisayarlar için paket yönlendirmesi. [1]'den alınmıştır. Bu örnekte gezgin bilgisayarın ev yeri New York'tur ve Seattle'daki bir bilgisayardan ona veri göndermek istenmektedir. Ancak bu sırada gezgin bilgisayar Los Angeles'dadır. Paketler öncelikle gezgin bilgisayarın ev yerine gelir ve buradan gezgin bilgisayarın geçici adresi öğrenilir.

Bu işlem sırasında ev ajanı, gezgin bilgisayar için gelen paketleri kapsülleyerek Los Angeles'taki yabancı ajanına gönderir. Bu ajan gerekli kapsülleri açarak veriyi gezgin bilgisayara ulaştırır.

Bunun yanı sıra, ev ajanı, gönderen bilgisayara, paketleri kapsülleyerek Los Angeles'taki yabancı ajanına göndermesini de bildirir. Bu durumda paketler boş yere New York'a gönderilmezler, kapsülenererek doğrudan gezgin bilgisayarın geçici adresine gönderilirler.

7.4 Yapısız (Ad Hoc) Ağlarda Yönlendirme

Telsiz ortamlarda baz istasyonunun varsa tüm haberleşme baz istasyonu (bazen *erişim noktası*, *access point*, olarak da adlandırılır) üzerinden aktarılır. Diğer durumlarda ise bilgisayarlar çevrelerindeki bilgisayarlara doğrudan veri gönderilerek bir haberleşme ağı yaratabilirler. Bu durumda veri bir bilgisayardan diğerine geçerek ilerler. Bu presip üzerine kurulmuş ağlara *yapısız ağlar* (*ad hoc networks*) denir. Pek çok ortam yapısız ağların kullanımı için uygun olabilir. Bunları:

- hareketli araçların oluşturduğu bir savaş ortamı,
- denizdeki bir filo,
- deprem sonrasında ilkyardım hizmeti verilen bir ortam

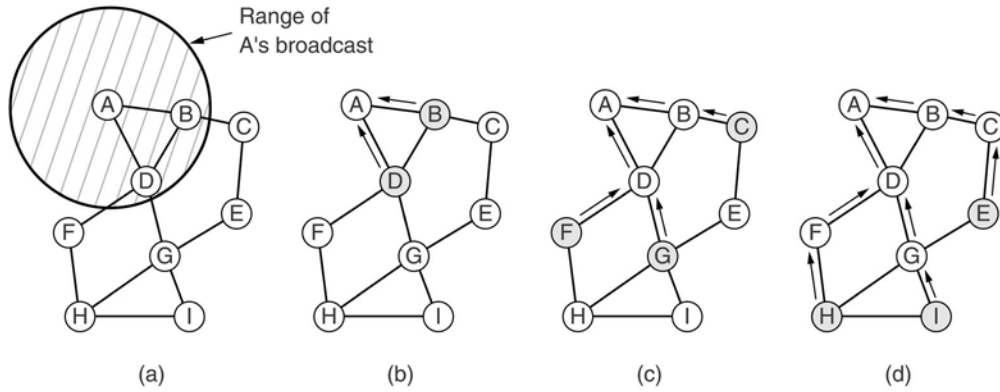
olarak sıralayabiliriz.

Bu durumda her bilgisayar hem yönlendirici hem de *konak (host)* görevini üstlenir. Bu ağlara *MANET'ler (Mobile Ad hoc NETWORKS)* de denir. Yapısız ağlarda hem kullanıcılar hem de yönlendiriciler hareketli olduğu için sabit bir topoloji yoktur. Zaman içinde düğümlerin komşuları değişir. Bu nedenle klasik telli ağlarda kullanılan teknikler yapısız ağlarda kullanılmaz.

Yapısız ağlar için önerilmiş yönlendirme algoritmalarından biri *AODV (Ad hoc On-demand Distance Vector)* yönlendirme algoritmasıdır. Bu teknik, ortamdaki düşük bant genişliğini ve kısa ömürlü pilleri göz önüne alır. Bu tekniğin bir diğer özelliği de gönderilecek bir paket olduğunda yol bulma çalışmalarının yapılmasıdır.

Bu ağlarda iki bilgisayar birbirlerinin kapsama alanına giriyorsa komşu olarak görülmektedir. Yani aralarında bir hat (bağlantı) varmış gibi düşünülebilir. İki bilgisayardan biri diğerinden daha güçlü bir vericiye sahipse bu bağlantı tek yönlü de olabilir. Genelde çalışmalarda, kolaylık olması amacıyla, tüm vericilerin güçlerinin eşit olduğu varsayılır. Ancak bilgisayarların kapsama alanları içinde olmaları onların haberleşebileceği anlamına gelmez. Aralarında binalar, dağlar vb gibi haberleşmeyi engelleyici unsurlar olabilir.

Yönlendirme algoritmasının çalışmasını Şekil 7.7'yi takip ederek daha iyi anlayabiliriz. Şekildeki A düğümünün, I düğümüne bir paket göndermek istediğini düşünelim. AODV algoritması her düğümde bir tablo oluşturur. Bu tabloda hangi varış noktasına hangi komşu düğüm üzerinden erişileceği bilgisi vardır. A'nın tablosunda I'ya nasıl erişileceği hakkında veri bulunmadığını varsayalım. Bu durumda A, I'ya ulaşmak için bir yol bulmalıdır. Yol bulma girişimi, bir istek mesajı ile yapılır.



Şekil 7.7 Yapısız ağlarda yönlendirme. [1]'den alınmıştır.

I'nın yerini öğrenmek için, A, *ROUTE REQUEST* paketi oluşturur ve yayar (broadcast). Bu paket Şekil 7.7'de görüldüğü gibi B ve D'ye ulaşır.

ROUTE REQUEST paketinin yapısı Şekil 7.8’de verilmiştir. Bu paket kaynak ve varış adreslerinin (*Source address, Destination address*) yanı sıra *istek-numarası (Request ID)* olarak adlandırılan bir alana sahiptir. Bu yerel bir sayaçtır ve her düğümde saklanır. Bu sayaç her ROUTE REQUEST mesajı ile bir artırılarak gönderilir. Böylece kaynak-adresi ve istek-numarası alanlarındaki değerler ROUTE REQUEST paketlerini tanımlar ve birbirleri ile karışmamasını sağlar. Bu durumda, aynı paketten bir tane daha geldiğinde ikinci paket hemen silinir.

İstek-numarası sayacı dışında, her düğüm ikinci bir *sıralama-sayacı (sequence counter)* tutar. Bu sayaç ne zaman birinin isteğine cevap gönderilse bir artırılır. ROUTE REQUEST paketindeki dördüncü alan kaynağın sıralama-sayacını saklar. Beşinci alan ise A’nın I’nın sıralama sayacını en son gördüğü değerdir (hiç görmediyse sıfırdır). *Sekme-sayacı (hop count)* paketin kaç kez sektiğini gösterir. İlk yaratıldığında sıfırdır.

Source address	Request ID	Destination address	Source sequence #	Dest. sequence #	Hop count
----------------	------------	---------------------	-------------------	------------------	-----------

Şekil 7.8 (5.21) ROUTE REQUEST paketinin yapısı. [1]’den alınmıştır.

ROUTE REQUEST paketini alan bir düğüm aşağıdaki adımları tamamlar:

- 1) (*kaynak-adresi, istek-numarası*) yerel *geçmiş tablosunda (history table)* aranır (daha önce görülmüş ve cevaplanmış olup olmadığını anlamak için). Eğer tekrar gelmiş bir mesaj ise silinir. Aksi halde (*kaynak-adresi, istek-numarası*) geçmiş tablosuna yazılır ve işleme devam edilir.
- 2) Mesajı alan düğüm yönlendirme tablosundan varışa daha yeni bir yol bilip bilmediğini kontrol eder. Biliyorsa ROUTE REPLY paketi kaynağa bu bilgi gönderilir. Yolun yeni olup olmadığını anlamak için paketteki varış-sıra-numarası ile tablodaki varış-sıra-numarası karşılaştırılır. Daha yeni yol tablodaki varış-sıra-numarasının pakettekinden büyük ya da eşit olması ile anlaşılır. Eğer daha küçükse kaynağın yol bilgisinin daha güncel olduğu anlaşılır ve 3. adıma geçilir.
- 3) Paketi alan düğüm daha yeni bir yol bilmediği için sekme-sayacını bir artırır ve ROUTE REQUEST paketini tekrar yayar. Ayrıca paketteki verilerin bir kopyasını *dönüş-yönlendirme-tablosuna (reverse routing table)* yazar. Gelen cevabı kaynağa iletebilmek için. Şekil 7.7’deki oklar dönüş yönlendirmesi için kullanılırlar.

Örneğimize geri dönersek ne B ne de D, I’nın yerini bilmektedir. Her biri A’ya geri dönmeyi sağlayacak dönüş-yolu bilgisini saklar, sekme sayacını 1 yapar ve paketi tekrar yayarlar (broadcast). B’nin yaydığı paket C’ye ve D’ye ulaşır. D paketi ikinci kez aldığı için siler. Aynı şekilde D’nin yaydığı paketi de B siler. Ancak D’nin yaydığı paket F ve G tarafından kabul edilir ve saklanır (Şekil7.7(c)). E, F ve I’da yayılan ROUTE REQUEST mesajı varış noktasına erişmiş olur. Dikkat edileceği gibi farklı düğümlerin yaydığı mesajlar bir koordinasyon gerektirmemektedir.

Gelen pakete cevap olarak I, *ROUTE REPLY* paketi yaratır. Bu paketin yapısı Şekil 7.9'da verilmiştir. Kaynak-adresi, varış-adresi, sekme-sayacı gelen mesajdan kopyalanır. Varış-sıra-numarası I'nın *sıralama-sayacından* okunur. Sekme-sayacı, 0 yapılır. *Yaşam süresi (lifetime)* alanı yolun ne süredir geçerli olduğunu tutar. Yaratılan paket *ROUTE REQUEST* paketinin geldiği düğüme (G) gönderilir. Oradan dönüş yolu bilgisini kullanarak D'ye oradan da A'ya gönderilir. Her düğümden sekme sayacı bir arttırılır.

Source address	Destination address	Destination sequence #	Hop count	Lifetime
----------------	---------------------	------------------------	-----------	----------

Şekil 7.9 *ROUTE REPLY* paketinin yapısı. [1]'den alınmıştır.

Geri dönüş yolu üzerinde her bir düğüm gelen mesajı inceler ve bulunan yol yerel yönlendirme tablosuna I'ya ulaşılacak yol olarak girilir. Bunun gerçekleşmesi için aşağıdaki durumlardan birinin sağlanması gereklidir:

- 1) I'ya ulaşmak için bir yol bilinmiyorsa,
- 2) *ROUTE REPLY* paketi üzerindeki sıra numarası yerel yönlendirme tablosundakinden daha büyükse,
- 3) Sıra numaraları eşit ancak yeni yol daha kısa ise (Sekme-sayacına bakılarak anlaşılır).

Bu şekilde dönüş yolu üzerindeki tüm düğümler I'ya nasıl erişileceğini öğrenmiş ve tablolarını güncellemiş olurlar. *ROUTE REQUEST* paketini almış ancak dönüş yolu üzerinde olmayan düğümler (B, C, E, F, H) belli bir süre sonra dönüş-yönlendirme-tablolarındaki ilgili kayıtları silerler.

Bu teknikte pek çok yayın (broadcast) yapılır. Yayınların sayısını düşürmek için IP paketlerinde olduğu gibi *yaşam-süresi (Time to Live)* alanı kullanılır ve kaynak tarafından belli bir değer verilir. Bu değer her sekmede azaltılır sifıra erişince paket silinir. Böylece paketlerin sekme sayısı belirlenmiş bir alanın dışına yayılması engellenmiş olur.

7.4.1. Yolların güncellenmesi

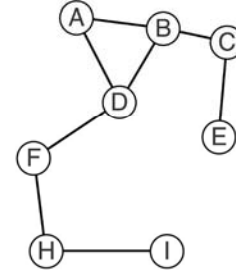
Yapısız ağlarda düğümler gezgin olduğu ve bazı düğümlerin zaman içinde enerjisi bittiği için topoloji anlık değişimler gösterebilir. Eğer Şekil 7.7'deki G düğümünün enerjisi biterse, A düğümü I'ya ulaşmak için kullanacağı yolun (ADGI) artık aktif olmadığını farkına varamaz. Bu gibi sorunları çözmek için, düğümler periyodik olarak *hello* mesajı yayarlar ve komşularının bu mesaja cevap vermesini beklerler. Gelen cevaplar komşu düğümlerin hangileri olduğunu gösterir. Daha önce komşu olan bir düğümden cevap gelmemesi o düğümün yerinin değiştiğini ya da enerjisinin bittiğini gösterir. Bu veriler doğrultusunda yönlendirme tabloları güncellenir.

Ağdaki her düğüm, her varış noktası için, ΔT süre içinde kendisine paket gönderen düğümleri kaydeder. Bu düğümler o varış noktasına ulaşmada düğümümüzün *aktif komşuları (active neighbours)* olarak bilinirler. Belli bir varış düğümüne erişmek için takip edilecek düğüm, sekme sayısı, o varışa ait aktif komşular, en yeni varış sıra

numarası bir tabloda varış adresine göre indekslenerek tutulur. D düğümü için olası bir yönlendirme tablosu Şekil 7.10(a)'da verilmiştir.

Dest.	Next hop	Distance	Active neighbors	Other fields
A	A	1	F, G	
B	B	1	F, G	
C	B	2	F	
E	G	2		
F	F	1	A, B	
G	G	1	A, B	
H	F	2	A, B	
I	G	2	A, B	

(a)



(b)

Şekil 7.10 a) D için olası yönlendirme tablosu, G varken, b) G düğümünden sonra topoloji. [1]'den alınmıştır.

Bir düğümün komşularından biri erişilmez hale gelince (yer değiştirme ya da pil bitmesine bağlı olarak *yok-olma*) o düğümün tablosunun güncellenmesi gerekir. Bu düğümün hangi yollar üzerinde olduğu incelenir ve bu yollara ait aktif komşulara bilgi verilir. Bu durumda yok-olan düğümüne ait verilerin tablolardan çıkarılması gerekir. Aktif komşular bu bilgiyi kendi aktif komşularına iletirler. Böylece haber ağ içinde yayılır.

Örnek olarak, G'nin enerjisinin bittiğini düşünelim. Değişen topoloji Şekil 7.10(b)'de verilmiştir. D, G'nin yok olduğunu anlayınca yönlendirme tablosunu kontrol eder ve E, G ve I'ya erişimin bu durumdan etkilendiğini fark eder. Bu durumdan etkilenen aktif komşular A ve B'dir. D onlara paket göndererek bilgilendirir ve E, G ve I'ya ait satırları tablosundan siler.

AODV'de tabloların düğümler arasında gönderilmesi gibi işlemler yoktur. Çünkü bu işlemlerin enerji gereksinimi fazladır ve düğümün enerjisinin bitmesine yol açarlar. Bu nedenle mümkün olduğunca kısa ve az mesaj gönderilmesi hedeflenir.