Duyarga Ağları için bir Eşuyumcusu Tasarım ve Uygulaması

Design and Implementation of a Synchronizer for Sensor Networks

Deniz Özsoyeller[1], Kayhan Erciyeş[2] , Orhan Dağdeviren[3]

1. Bilgisayar Mühendisliği Bölümü
İzmir Ekonomi Üniversitesi
deniz.ozsoyeller@ieu.edu.tr

2. Uluslararası Bilgisayar Enstitüsü
Ege Üniversitesi
kayhan.erciyes@ege.edu.tr

3. Bilgisayar Mühendisliği Bölümü
İzmir Yüksek Teknoloji Enstitüsü
orhandagdeviren@iyte.edu.tr

Özetçe

Zaman eşuyumlu algoritmaların dağıtık sistemlerde uygulanması, zaman eşuyumsuz algoritmaların uygulanmasına göre genelde daha sorunsuzdur. Eşuyumcular (Ing. Synchronizers) , eşuyumsuz bir algoritmanın dağıtık sistemlerde eşuyumlu olarak çalışmasını sağlarlar. Bu çalışmada, Telsiz Duyarga Ağları (TDA)'nda eşuyumsuz algoritma ve protokolleri eşuyumlu hale getirmek için bir eşuyumcusu öneriyoruz. Bu eşuyumcu, başlıca α ve β eşuyumcularından oluşmaktadır. Çalışmamızda, TDA önce kapsayan ağaçlardan oluşan kümelere ayrılmakta, sonra bu kümeler bir halka ağ protokolü ile birbirine bağlanmaktadır. Kümelerin içinde β eşuyumcusu, kümeler arasında ise α eşuyumcusu kullanılarak eşuyumcusu gerçeklenmektedir. TDA kümeleme ve halka oluşturma protokolleri açıklanarak alınan sonuçlar bildirilmiştir.

Abstract

Implementation of synchronous algorithms in distributed systems in general is less troublesome than the implementation of asynchronous algorithms. Synchronizers provide synchronous execution of an asynchronous algorithm in distributed systems. In this study, we propose a synchronizer for Wireless Sensor Networks (WSNs). This synchronizer consists of α and β synchronizers. In our work, the WSN is first divided into clusters and then these clusters are connected using a ring protocol. Synchronization is provided using the β synchronizer in the cluster and α synchronizer among the clusters. We describe the clustering algorithm and the ring formation algorithm for the WSNs and give the results obtained so far.

1. Giriş

Zaman eşuyumlu algoritmaların dağıtık sistemlerde uygulanması zaman eşuyumsuz algoritmaların uygulanmasına göre genelde daha sorunsuzdur. Eşuyumcular, eşuyumsuz bir algoritmanın dağıtık sistemlerde eşuyumlu olarak çalışmasını sağlarlar. TDA'larda kapsayan ağaç oluşturma; mesaj yönlendirmesi, kümeleme gibi çeşitli amaçlar için kullanılmaktadır. Gallagher, Humblet ve Spira'nın [1] algoritması, yönsüz bir çizge üzerinde dağıtık olarak en küçük kapsayan ağacı(EKKA) bulan öncü bir algoritmadır. Algoritmanın amacı küçük parçaları birleştirip daha büyük parçaları dış kenarlar üzerinden birleştirmektir. Parça denilen yapı EKKAya ait olan bir alt ağaçtır. Dış kenar, parçacıkları birbirlerine birleştiren en küçük kenarlardır. Parçaların birleşmesi bazı kurallar çerçevesinde gerçekleşmektedir. En kötü durumda toplam mesaj sayısı, N düğüm sayısı, E kenar sayısı olmak üzere $5Nlog_2N + 2E$ olacaktır. Algoritmanın zaman karmaşıklığı $O(E+Nlog2N)$ olarak verilebilir. Awerbuch[2], Garay[3] ve Peleg[4]'ın tarafından da dağıtık en küçük kapsayan ağaç algoritmaları önerilmiştir.

Birleştiren Kümeleme Algoritması(BKA)[5] Gezgin Tasarsız Ağ(GTA) üzerinde kümeleri birleştirip istenilen aralıkta kümeleri bulmayı hedefleyen bir algoritma olup Gallagher'in parçacık fikrini temel almıştır. Fakat Gallagher'in algoritmasındaki gibi EKKA bulunmamış yerine kümelemeye odaklanılmıştır. Bu yaklaşım mesaj kompleksliğini $O(NlogN)$'den $O(N)$'e düşürmüştür. TDA üzerinde yapılmış birçok kapsayan ağaç (KA) ve kümeleme çalışması bulunmaktadır. HEED[6], algoritması TDA üzerinde dağıtık olarak kümeleme yapabilen bir algoritmadır. HEED içindeki küme liderleri sezgisel ve olasılık içeren bir yaklaşım kullanılarak seçilip düğümün ve çevresindeki komşu düğümlerin enerjisine dikkat eder. HEED homojen bir ağ olduğunu varsayar. LEACH[7], küme liderlerini dönerli bir şekilde kullanan ve rasgele seçen bir kümeleme algoritmasıdır. PEAS[8] algoritmasında bir düğüm çevresinde bir yönlendirme işlemi fark ettiği zaman uykuya geçer. GAF[9], algoritmasında duyarga ağ, kare ızgaralar şeklinde parçalanmıştır ve haberleşme çıkış düğümüne doğru yöneltilmiştir. GEAR[10] ve TTDD[11] protokolleri Telsiz Duyarga Ağı(TDA) üzerindeki diğer kümeleme örnekleridir. BCDCP[12] protokolü, LEACH protokolünü genişletip, kapsayan ağaç kurallarını eklemiştir. FTEEDA[13] protokolü, hata toleransı kavramı üzerinde durarak, kapsayan ağaç oluşturmayı hedeflemiştir.

Rubin ve arkadaşları[14] düğümleri yüksek kapasiteli ve düşük kapasiteli olarak sınıflandırmışlar ve güç durumuna

göre omurgayı oluşturmayı düşünmüşlerdir. Yüksek kapasiteli düğümlerin enerji olanakları diğer düğümlerden daha fazla olduğu için omurga üzerine almışlardır. YaFeng [15] optimum sanal çok gönderimli yönlendirmeyi en küçük yapacak bir omurga oluşturmaya çalışmışlardır. Haito ve Gupta [16] SBC algoritmasını enerji etkili bir çözüm oluşturmak için GTA üzerinde tasarlamışlardır. SBC algoritması iki basamakta çalışır, birinci basamak bir veya daha çok omurga çekirdek düğümü bulmaya çalışır. İkinci aşamada da bu omurgaya en yakın olan düğümler omurgaya dahil edilir. Min'in algoritması [17] olan RVBSM her düğümün kendi bölgesel bilgileri belirli periyotlarda kaydettiğini varsayar. Bölgesel bilgilere bakılarak omurga oluşturma işlemi yapılır. Ju [18] EBS algoritması ile iki tane zamanlayıcı koyarak bu zamanlayıcılar yardımı ile omurga oluşturma işlemini yapar. Omurga Oluşturma Algoritması(OOA) [19], GTA üzerinde dağıtık omurga oluşturan bir algoritmadır. Benzerlerinden farklı olarak OOA omurgayı, yönlendirilmiş bir halka şeklinde oluşturmuş ve bu topolojinin avantajı ile beraber üst katmanlara daha iyi bir servis kalitesi sunulmuştur [20-21]. TDA üzerinde sanal omurga oluşturmak için çeşitli çalışmalar sunulmuştur. EVBT algoritması [22] TDA üzerinde az mesaj kullanarak kapsayan ağaç omurgası oluşturmayı hedeflemiştir. Pil ömürlerini göz önüne alan BACDS [23] algoritması, TDA üzerinde enerji etkili bir çözüm sunmuştur. Ayrıca, benzer bir çalışma ViBES [24] algoritması ile sunulmuştur. Bu çalışmada, TDA için eşuyumsuz algoritma ve protokolleri eşuyumlu hale getirmek için bir eşuyumcusu öneriyoruz. Bu eşuyumcu, başlıca α ve β eşuyumcularından oluşmaktadır. Çalışmamızda, TDA önce kapsayan ağaçlardan oluşan kümelere ayrılmakta, sonra bu kümeler bir halka ağ protokolü ile birbirine bağlanmaktadır. Kümelerin içinde α eşuyumcusu, kümeler arasında ise β eşuyumcusu kullanılarak eşuyumcusu gerçeklenmektedir. Bilgilerimiz dahilinde, literatürde TDA için bir eşuyumcu tasarımı bildirilmemiştir. Ayrıca, kümeleme ve halka oluşturma için kullandığımız algoritmalar da daha önce tasarlanmamış özgün algoritmalardır. 2. Bölüm'de kümeleme ve kapsayan ağaç oluşturma açıklanmakta, 3. Bölüm'de ise halka protokolü TDA kümeleme ve halka oluşturma algoritması detaylandırılmaktadır. 4. Bölüm'de ise γ eşuyumcusunun çalışma yöntemi ve sonuçlar belirtilmektedir.

2. Eşuyumcusu

E uyumcusu, bilgisayar a ının kümelenmi oldu unu ve kümeler içinde β e uyumcusunun çalı tı ını, kümeler arasında ise α e uyumcusunun çalı tı ını varsayar. Bir β eşuyumcusu, kapsayan ağaç üzerinde çalışır ve her zaman diliminde, kök düğüm çocuklarına, çocuklar da çocuklarına o zaman diliminde çalışma mesajını iletirler. İşlem bittiğinde ise ağacın çocuk düğümleri ebeveynlerine, ebeveynler de ebeveynlerine ve köke kadar işlemin bittiği mesajı ulaştırılır. Bu mesaj kök düğüme geldiğinde, kök düğüm bir sonraki zaman dilimini başlatabilir. α eşuyumcusu ise her adımdan sonraki adımı gerçekleyebilmek için tüm düğümlerden yanıt bekler ve yanıtlar geldiğinde tüm düğümlere bir sonraki adımı başlatma mesajı gönderir.

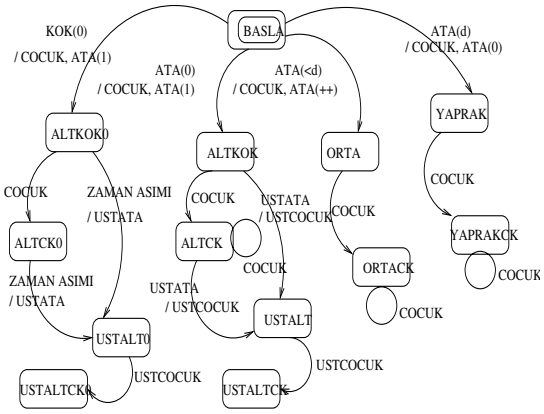
Böyle bir eşuyumcuyu uygulamak için başlıca kümeleme ve ağaç yapısı oluşturma ve bu ağaçların kök düğümlerini haberleştirme algoritma ve protokollerine ihtiyaç vardır. 3. Bölüm'de TDA'da kümeleme ve ağaç yapısı oluşturmak için iki ayrı algoritma önermekteyiz. Kümelenmiş TDA'da küme içinde β eşuyumcu çalışabilecektir. Her TDA kümesinin ağaç yapısının kök düğümleri arasında 4. Bölüm'de açıklanan bir halka oluşturma algoritması yoluyla α eşuyumcu işlevi, TDA kümeleri arasında gerçeklenecektir.

3. Kümeleme ve Kapsayan Ağaç Oluşturma

Bu bölümde kapsayan ağaç tabanlı kümemle algoritmasının genel çalışmasını, bir örnek işlemini ve kısaca teorik analizini sunacağız.

3.1. Genel Özellikler

Kümeleme, duyarga ağlarının yönetimde önemli bir metot olup, kümeleme sayesinde mesaj transferlerindeki enerji kullanımı düşünülebilinir. Bir diğer önemli metot da kapsayan ağaç oluşturmaktır. Kök düğümün başlangıç noktası olduğu bir kapsayan ağacın oluşturulması sayesinde duyarga ağı üzerindeki veriler daha kolay bir şekilde toparlanır. Bu iki yöntemi bir arada kullanan duyarga ağ üzerinde çalışıp, kapsayan ağaç tabanlı kümeleme yapan bir algoritma öneriyoruz. Algoritmamız iki fazdan oluşmaktadır. Birinci fazda, düğümler ağaç üzerindeki seviyelerine göre *ALTKÖK*, *ORTA* ve *YAPRAK* düğüm olarak ayrılırlar. İkinci fazda ise *ALTKÖK* düğümler birbirleri arasında ikinci seviyede kapsayan ağaç yapısı oluştururlar. İkinci seviyedeki kapsayan ağaç yapısı aynı zamanda omurga görevi görmektedir. Bu çalışmada omurgayı halka şeklinde oluşturacağımız için ikinci faz üzerinde durmayacağız. Algoritmanın sonlu durumlu makinesi Şekil 1'de verilmiştir. Kök düğüm tarafından düzenli olarak gönderilen *ATA* mesajı ile algoritma çalışmaya başlar. *ATA* mesajını alan düğüm bu mesajın içindeki zıplama sayısı(Ing. hop count) alanına göre durumuna karar verir ve gerekli mesajı komşularına gönderir.

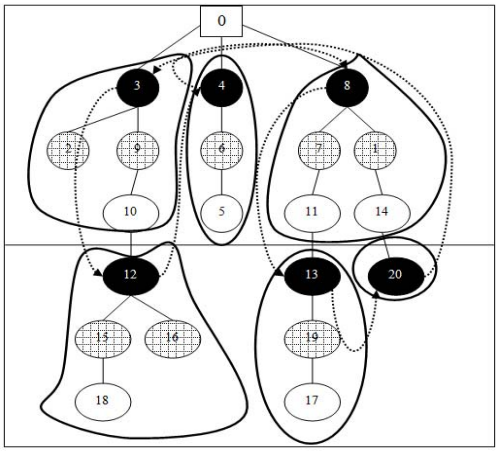


Şekil 1*: Küme Tabanlı Kapsayan Ağaç Algoritmasının Sonlu Durumlu Makinesi*

Ata düğüm tarafından çocuklara *ATA* mesajı gönderilip çocuklardan *ÇOCUK* mesajı toparlanır. *ÜSTATA* ve *ÜSTÇOCUK* mesajları ikinci seviye için gereklidir. Zaman aşımına uğrayan düğümler gerekli mesajları tekrardan gönderirler. Her mesaj alıcı, tip ve zıplama sayısı alanlarını tutar. Zıplama sayısı 0'a eşit olduğu zaman düğüm durumunu *ALTKÖK* olarak değiştirir. Zıplama sayısı derinlik parametresine eşit olduğu zaman düğüm durumunu *YAPRAK* olarak değiştirir. *YAPRAK* ve *ALTKÖK* durumunda olmayan düğüm *ORTA* durumundadır. Çocuğu olan *ALTKÖK* durumundaki bir düğüm *ALTCK* durumuna, *ORTA* durumunda olan bir düğüm ise ORTACK durumuna geçer. *ALTKÖK0* kök düğümden *ATA* mesajını alan *ALTKÖK*'e denir.

3.2. Örnek bir Uygulama

Şekil 2 de 1. fazı örnek bir uygulaması gösterilmiştir. Bu örnekte derinlik parametresi 3 olarak verilmiştir. İlk olarak kök düğüm olan düğüm 0, *ATA* mesajını düğüm 3, düğüm 4 ve düğüm 8'e göndermiştir. Bu mesajı alan düğüm 3, düğüm 4 ve düğüm 8 *ALTKÖK0* durumuna geçip mesajdaki zıplama sayısını 1 arttırıp komşularına göndermişlerdir. Bu mesajları alan düğüm 2, düğüm 9, düğüm 6, düğüm 7 ve düğüm 1 *ORTA* durumuna geçmişlerdir. Zıplama sayısını 3 alan düğüm 10, düğüm 5, düğüm 11 ve düğüm 14 *YAPRAK* düğümler olmuşlardır. Algoritma sonlandığında koyu renkli boyanmış düğümler *ALTKÖK*, kare desenli boyanmış düğümler *ORTA*, içleri boş kalan düğümler ise *YAPRAK* durumunda kalır. Ayrıca Şekil 2'de koyu çizgiler ile küme sınırları da belirtilmiştir. Örneğin, Şekil 2'de görüldüğü üzere düğüm 3'ün liderliğinde, düğüm 2, düğüm 9 ve düğüm 10 aynı küme içinde yer almaktadır. Noktalı çizgiler ile gösterilen oklar omurgayı gösterdiğinden dolayı bir sonraki bölümde anlatılacaktır.



Şekil 2: *Kapsayan Ağaç Tabanlı Küme Algoritmasının Örnek bir Uygulaması*

3.3. Analiz

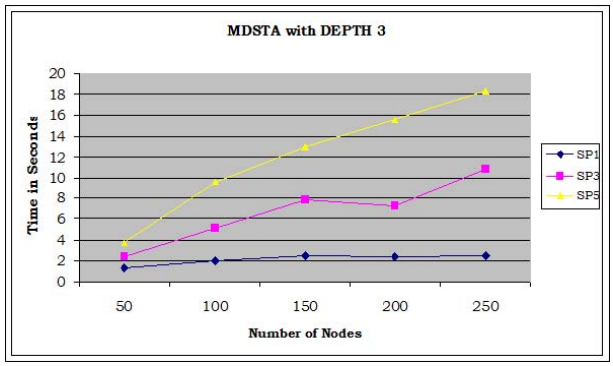
Dağıtık algoritmaların en önemli tasarım kriterlerinden biri algoritmanın kullandığı mesaj sayısıdır. Bu çalışmada kullandığımız algoritmanın birinci fazının mesaj karmaşıklığı Teorem 1'de verilmiştir.

Teorem 1: Maksimum *d* derinliğindeki yerel kapsayan ağaç ve *k* derinliğindeki ikinci seviyedeki ağacın oluşturulması için algoritmanın ilk fazı için *O(2kd)* mesaj gerekir.

İspat: Her yerel ağacın oluşması için *ATA/ÇOCUK* mesaj değişimleri ile *O(2d)* mesaj gerekir. *O(k)* adet yerel ağaç olduğu için toplam mesaj karmaşıklığı *O(2kd)*'dir.

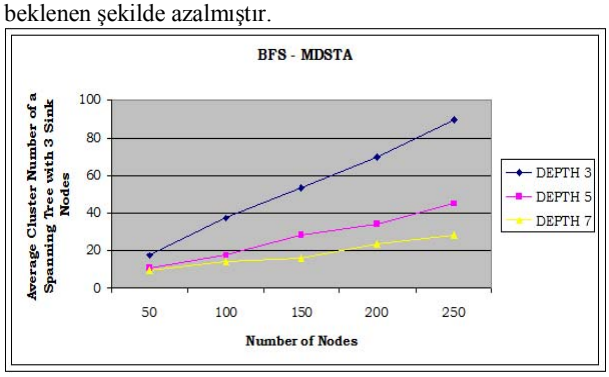
3.4. Sonuçlar

Kapsayan ağaç tabanlı kümeleme algoritması ns2 benzetim ortamı üzerinde uygulanmıştır. Şekil 3'de derinlik 3'e sabitlenerek kök düğüm ve toplam düğüm sayısını değiştirildi. Birden fazla kök düğüm olmasına rağmen çalışma zamanının doğrusal olarak arttığı Şekil 3'de görülebilinir.



Şekil 3: *Derinliğin 3 olduğu birden fazla kök düğümden elde edilen kapsayan ağaç algoritmasının çalışma zamanı*

Şekil 4'te ise 3 kök düğüm için derinlik parametresi ve toplam düğüm sayısı arttırılarak küme sayısı ölçülmüştür. Şekil 4'te görüldüğü üzere küme sayıları, düğüm sayıları ile doğrusal olarak artmış ve derinlik parametresinin artmasıyla da beklenen şekilde azalmıştır.



Şekil 4: 3 Kök düğüm ve enine arama tekniğiyle kapsayan ağaç algoritmasının küme sayıları

4. Halka Oluşturma

Bu bölümde kümelenmiş bir duyarga ağı üzerinde halka oluşturan algoritmanın genel özelliklerini ve bir örnek uygulamasını göstereceğiz. Ayrıca bu algoritmanın toplam kullandığı mesaj sayısını inceleyeceğiz.

4.1. Genel Özellikler

OOA[19]'ya göre ağ kümelendikten sonra küme liderleri konum bilgilerini (X,Y) koordinatları biçiminde tüm ağa yayarlar. Her lider diğer liderlerin bilgilerini topladıktan sonra merkezi bir algoritma çalıştırarak bir sonraki küme liderini bulur ve bu şekilde halka yapısı oluşur. Duyarga ağı üzerindeki düğümler enerji kısıtlı oldukları için ve mesajlaşma en çok enerji harcatan etken olduğu için tüm ağa sel yöntemi ile mesaj yaymak maliyetli bir iştir. Bunun yerine önerimiz lider düğümlerin daha önceden oluşturulmuş kapsayan ağaç yapısı üzerinden kök düğüme halkaya katılma isteklerini göndermeleridir. Kök düğüm merkezi bir algoritma çalıştıracak ve her lider düğüme halka üzerindeki bir sonraki lider düğümünün adresini gönderecektir. Bu yapının üç önemli avantajı vardır. Birinci olarak liderler kök düğüme halkaya katılma isteklerini gönderirken daha önceden oluşturulan kapsayan ağaç yapısıyla birlikte az sayıda mesaj

kullanacaklardır. İkinci olarak merkezi algoritmayı sadece kök düğüm çalıştıracak böylece bellek ve işlemci kısıtlı diğer düğümler bu yükten kurtulacaklardır. Üçüncü olarak kök düğüm, küme lideri düğümlere halka bilgisini gönderirken daha önceden küme lideri düğümlerin kullandığı patikaları kullanabilecektir.

Önerdiğimiz algoritma küme liderleri üzerinde çalışan ve kök düğümde çalışan olmak üzere iki adet kod parçasından oluşmaktadır. Her küme lideri halkaya katılma isteğini kapsayan ağaç yapısı üzerinden kök düğüme gönderir ve kök düğümden bir halka üzerindeki bir sonraki lider bilgisinin gelmesi bekler:

1. Küme Lideri Halka Oluşturma Algoritması
2. Başla.
3. Oluşturulmuş kapsayan ağaç yapısı üstünden kök düğüme *HALKAYA_KATILMA_İSTEĞİ* mesaj gönder.
4. Kök düğümden *BİR_SONRAKİ_LİDER* mesajını al.
5. Sonlandır.

Kök düğüme halkaya katılma istekleri komşuları üzerinden yönlendirilerek gelir. Bu komşular aynı zamanda kapsayan ağaç tabanlı küme mimarisi üzerinde küme liderleridir. Kök düğüm her komşusu için bir grup oluşturup komşuları üzerinden gelen küme lideri mesajlarını bu gruplara yerleştirir. Kime liderlerinden tüm mesajlar geldikten sonra derinliğe arama(Ing. Depth-first Search) tekniğine benzer bir şekilde önce aynı gruptaki düğümleri birbirine bağlar. Daha sonra her gruptaki bağlanmamış lider düğümü bir sonraki gruptaki kök düğüme komşu olan düğüm ile bağlar. Gruplar arasındaki sıralama basit olarak adrese göre olabilir. Yapılan bütün bağlama işlemleri merkezi olarak yapılır. Kök düğüm son olarak lider düğümlerin her birine bağlandıkları düğümü *BİR_SONRAKİ_LİDER* mesajı ile gönderir. Kök düğüm üzerinde algoritma şu şekildedir:

1. Kök Düğüm Halka Oluşturma Algoritması
2. Başla.
3. Komşu sayısı kadar grup oluştur ve her komşuya farklı birer grup adresi ver.
4. Küme liderlerinden gelen *HALKAYA_KATILMA_İSTEĞİ* mesajlarını toparla ve en son üzerinden geçen komşumun adresine ait gruba yerleştir.
5. Gruplar içinde bulunan küme liderleri arasında derinliğine arama tekniğine benzer bir şekilde bağ oluştur.
6. Grubun bağlanmamış küme liderini bir sonraki grubun kök düğüme komşu olan küme liderine bağla ve son gruba ulaşana kadar bu işleme devam et.
7. En son grubun bağlanmamış küme liderini ilk grubun kök düğüme komşu olan liderine bağla ve halka oluşumunu tamamla.
8. Her grubun tüm liderlerine ulaşacak şekilde *BİR_SONRAKİ_LİDER* mesajını gönder.
9. Sonlandır.

4.2. Örnek bir Uygulama

Örnek bir uygulama Şekil 2 de verilmiştir. Daha önceden kapsayan ağaç tabanlı kümeleme algoritması kullanılarak ağ kümelere ayrılmış ve kapsayan ağaç kurulmuştur. Bu örnekte 0 kök düğüm, düğüm 3, düğüm 4, düğüm 8, düğüm 12, düğüm 13, düğüm 20 ise küme lideri düğümlerdir. Şekil 3 üzerinde halka bağlantıları noktalı oklarla gösterilmiştir. Bütün lider düğümler kök düğüme *HALKAYA_KATILMA_İSTEĞİ* mesajı gönderirler. Kök düğüm bu mesajları topladıktan sonra merkezi olarak algoritmayı çalıştırmaya başlayıp, düğümleri bağlamaya başlar. Düğüm 3 ve düğüm 12 aynı grup içindedirler ve düğüm 3, düğüm 12'e bağlanmıştır. Düğüm 4'ün grubunda başka eleman olmadığı için düğüm 8'e bağlanmıştır. Düğüm 8, düğüm 13'e, düğüm 13 ise düğüm 20'ye bağlanmıştır. En son gruba ait bağlanmamış düğüm olan düğüm 20 ise düğüm 3'e bağlanmış ve böylece halka yapısı tamamlanmıştır. Son olarak kök düğüm bağlantı bilgilerini *BİR_SONRAKİ_LİDER* mesajlarıyla lider düğümlere gönderir.

4.3. Analiz

Teorem 2'de halka oluşturma algoritmasının en kötü durumundaki mesaj sayısı verilmiştir.

Teorem 2: Maksimum *d* zıplama uzaklığındaki *k* adet lider düğümün bulunduğu bir duyarga ağı üzerinde halka algoritması en kötü durumda *2kd* mesaj ile çalışır.

İspat: k adet lider düğüm *d* zıplama uzaklıktaki kök düğüme toplamda *kd* adet *HALKAYA_KATIL* mesajı gönderir ve kök düğümden *kd* adet *BİR_SONRAKİ_LİDER* mesajını alır. Bundan dolayı genel toplamda *2kd* mesajlaşma olur.

5. Sonuçlar

Bu çalışmada, TDA için α ve β eşuyumcularından oluşan bir eşuyumcusunun tasarım ve uygulama detaylarını oluşturduk. Kümeleme algoritmaları ile ilgili sonuçları verdik. Çalışmamız devam etmektedir ve yakın bir gelecekte *lider seçimi* gibi bilinen bir zaman eşuyumsuz dağıtık sistem problemini tasarladığımız eşuyumcu yoluyla, zaman eşuyumlu olarak TDA'da denemeyi planlamaktayız.

6. Kaynakça

[1] R.G. Gallager, P.A. Humblet and P.M. Spira, "A Distributed Algorithm for Minimum-weight Spanning Trees," ACM TOPLAS, Vol.5, no. 1, pp 66-77, 1983.

[2] B. Awerbuch, "Optimal Distributed Algorithms for Minimum Weight Spanning Tree, Counting, Leader Election and Related Problems", ACM STOC, 1987.

[3] [J. Garay, S. Kutten and D. Peleg, "A Sub-Linear Time Distributed Algorithm for Minimum-Weight Spanning Trees, IEEE FOCS, 1993.

[4] D. Peleg and V. Rubinovich, "A Near Tight Lower Bound on the Time Complexity of Distributed Minimum Spanning Tree Construction", IEEE FOCS, 1999.

[5] O. Dagdeviren, K. Erciyes and D. Cokuslu, "A Merging Clustering Algorithm for Mobile Ad-hoc Networks", ICCSA 2006, LNCS, pp 681- 690, 2006.

[6] [O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad-hoc Sensor Networks", IEEE Transactions on Mobile Computing, Vol.3, issue 4, pp. 366-379, 2004.

[7] [W.R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks", HICSS, 2000.

[8] F. Ye, G. Zhong, J. Cheng, S. Lu and L. Zhang, "PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks", ICDCS, 2003.

[9] Y. Xu, J. Heidemann, and D. Estrin, "Geography informed Energy Conservation for Ad-hoc Routing", MobiCom, 2001, pp 70-84.

[10] Y. Yu, R. Govindan and D. Estrin, "Geographical and Energy Aware Routing: A Recursive Data Dissemination

Protocol for Wireless Sensor Networks", UCLA Technical Report UCLA/CSD-TR-01-0023, 2001.
[11] F. Ye, H. Luo, J. Cheng, S. Lu and L. Zhang, "A Two-tier Data Dissemination Model for Large-scale Wireless Sensor Networks", MOBICOM, September 2002.
[12] S.D. Muruganathan, D.C.F. Ma, R.I. Bhasin and A.O. Fapojuwo, "A centralized energy-efficient routing protocol for wireless sensor networks", IEEE Communications Magazine, Vol. 43, pp 8-13, 2005.
[13] S. Coleri and P. Varaiya, "Fault tolerant and energy efficient routing for sensor networks", IEEE GLOBACOM, pp. 10-15, 2004.
[14] I. Rubin, A. Behzad, Z. Runhe, L. Huiyu, and E. Caballero. Tbone, "A Mobile-backbone Protocol for Ad-hoc Wireless Networks", IEEE International Conference on Aerospace Conference, Vol.6, pp 2727-2740, 2002.
[15] W. Ya-feng, X. Yin-long, C. Guo-liang and W. Kun, "On the Construction of Virtual Multicast Backbone for Wireless Ad- hoc Networks", MASS, pp 25-27, 2004.
[16] L. Haitao and R. Gupta, "Selective Backbone Construction for Topology Control in Ad-hoc Networks", MASS, pp 41-50, 2004.
[17] M. Min, F. Wang, D.-Z. Du and P. M. Pardalos, „Selective Backbone Construction for Topology Control in Ad-hoc Networks", MASS, 2005.
[18] J. Huejiun and I. Rubin, "Enhanced Backbone Net Synthesis for Mobile Wireless Ad-hoc Networks", GLOBECOM, Vol.5, pp 5-10, 2005.
[19] O. Dagdeviren, K. Erciyes, "A Distributed Backbone Formation Algorithm for Mobile Ad-hoc Networks", ISPA, Springer Verlag LNCS, 2006.
[20] K. Erciyes, "Distributed Mutual Exclusion Algorithms on a Ring of Clusters", ICCSA, LNCS, Vol.3045, pp 518-527, 2004.
[21] K. Erciyes, "Cluster-based Distributed Mutual Exclusion Algorithms for Mobile Networks", ICCS, 2005, LNCS,Vol.3149, pp 933-940, 2005.
[22] Z. Bosheng, A. Marshall and L. Tsung-Han, "An energy-aware virtual backbone tree for wireless sensor networks", GLOBECOM, Vol. 1, p. 6, 2005.
[23] C. Ma, Y. Yang and Z. Zhang, "Constructing battery-aware virtual backbones in sensor network, ICPP, pp. 203-210 , 2005.
[24] S. Basagni, M. Elia and R. Ghosh, "ViBES: virtual backbone for energy saving in wireless sensor networks", MILCOM, Vol. 3, pp. 1240-1246, 2004.