**EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ**

**(DOKTORA TEZİ)**

# KABLOSUZ DUYARGA AĞLARINDA ÖLÇEKLENEBİLİR GÜVENLİ GRUP HABERLEŞMESİ

**Özgür SAĞLAM**

**Tez Danışmanı : Prof. Dr. Mehmet E. DALKILIÇ**

**Uluslararası Bilgisayar Anabilim Dalı**

**Bilim Dalı Kodu : 619.03.03**
**Sunuş Tarihi : 10.09.2009**

**Bornova-İZMİR**
**2009**

Özgür SAĞLAM tarafından doktora tezi olarak sunulan “Kablosuz Duyarga Ağlarında Ölçeklenebilir Güvenli Grup Haberleşmesi” başlıklı bu çalışma E.Ü. Lisansüstü Eğitim ve Öğretim Yönetmeliği ile E.Ü. Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi’nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 10.09.2009 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

**Jüri Üyeleri:** **İmza**

| | | |
|---|---|---|
| **Jüri Başkanı** | **: .....** | **.....** |
| **Raportör Üye** | **: .....** | **.....** |
| **Üye** | **: .....** | **.....** |
| **Üye** | **: .....** | **.....** |
| **Üye** | **: .....** | **.....** |

# ÖZET

## KABLOSUZ DUYARGA AĞLARINDA ÖLÇEKLENEBİLİR GÜVENLİ GRUP HABERLEŞMESİ

SAĞLAM, Özgür

Doktora Tezi, Uluslararası Bilgisayar Enstitüsü
Tez Yöneticisi: Prof. Dr. Mehmet Emin DALKILIÇ
Eylül 2009, 125 sayfa

Bu tezde, eş özellikli duyarga düğümlerinden oluşan geniş ölçekli Kablosuz Duyarga Ağlarında, kendi kendine organize olacak şekilde ağı denk kümelere ayıran ve düşük düğüm derecelerinde etkin performans gösteren çok zıplamalı bir kümeleme protokolü –Tekrarlı Kümeleme (TK)– geliştirilmiş; bu protokol ile yapılandırılan bir ağda, duyarga düğümleri arasındaki güvenli haberleşme için kullanılacak ikili ortak anahtarların yerel olarak belirlenmesinde kullanılabilecek anahtar belirleme protokollerinin TK protokolü ile çapraz seviye ilişkileri ortaya konmuştur.

Bu amaçla, geliştirilen TK protokolü Krishnan ve Starobinski (2006) tarafından önerilen yapılanma protokolü ile benzetim yolu ile karşılaştırılmış; artan ağ eleman sayısına göre daha hızlı ve ölçeklenebilir bir yapılanma zaman performansına sahip olduğu belirlenmiş ve düşük düğüm derecesine (7-8) sahip ağlarda yüksek kümeleme performansı gösterdiği ortaya konmuştur. TK protokolü ile güvenli haberleşme altyapısının kurulmasında ise anahtar belirleme maliyetlerinin, yapılanma performansını etkilemeden, TK protokolü ile olan çapraz ilişki kullanılarak azaltılabileceği gösterilmiş ve asimetrik anahtar belirlemenin kabul edilebilir bir bedel karşılığında simetrik anahtar ön dağıtımına göre tercih edilebileceği gösterilmiştir. Yapılanma tamamlandıktan sonra ağa yeni eleman eklenmesi veya ağdan eleman çıkarılması işlemlerinde kümeleme altyapısı sayesinde güvenlik ölçeklenebilir şekilde güncellenebilmektedir.

**Anahtar sözcükler:** Geniş ölçekli kablosuz duyarga ağları, kendi kendine organizasyon, tekrarlı kümeleme, çok zıplamalı yapılanma, eş duyarga düğümleri, ikili ortak anahtar belirleme, çapraz seviye ilişki, simetrik anahtar ön dağıtımı, asimetrik anahtar belirleme, ölçeklenebilir yapılanma ve güvenlik.

# ABSTRACT

## SCALABLE SECURE GROUP COMMUNICATION IN WIRELESS SENSOR NETWORKS

SAĞLAM, Özgür

Ph.D. in International Computer Institute
Supervisor: Prof. Dr. Mehmet Emin DALKILIÇ
September 2009, 125 pages

In this thesis, a multi-hop clustering protocol, namely Iterative Clustering (IC), which performs an efficient, self organizing balanced clustering in low node degrees where the sensor nodes are equally likely, has been developed and analyzed for large scale Wireless Sensor Networks. Then, the cross layer relations of the key establishment protocols, which are needed for establishing pairwise shared keys locally among sensor nodes for secure communication, and the IC protocol has been provided.

For this purpose, simulation results of the developed IC protocol performance has been compared with that of network organization protocol proposed in Krishnan and Starobinski (2006). The comparison results depict that IC protocol has a faster and scalable configuration time with the increasing network size. In addition, IC protocol provides an efficient clustering performance for the networks having low node degrees (7-8). From the security point of view, it has been experimentally shown that without affecting the configuration performance of the IC protocol, cost of the key establishment can be reduced by using the cross layer relations and asymmetric key establishment can be preferred instead of symmetric key pre-distribution in exchange of an acceptable cost. After the configuration, security update for membership operations like member addition and deletion are handled in corresponding clusters which makes these operations scalable.

**Key words:** Large scale Wireless Sensor Networks, self organization, iterative clustering, multihop configuration, equal sensor nodes, pairwise key establishment, cross layer relation, symmetric key pre-distribution, asymmetric key establishment, scalable configuration and security.

## TEŞEKKÜR

Doktora eğitimim ve tez çalışmam sırasında güvenlerini ve desteklerini sürekli olarak hissettiğim değerli öğretim üyeleri Sayın Prof. Dr. Turhan Tunalı ve Sayın Prof. Dr. Kayhan Erciyeş'e, tez çalışmasına değerli fikirleri ve yönlendirmeleri ile çok önemli katkı sağlayan Sayın Doç. Dr. Albert Levi'ye, yüksek lisans ve doktora eğitimim boyunca danışmanlığımı yapan, yakından ve uzaktan her türlü desteğini hiçbir zaman esirgemeyen ve her zaman güvenen çok değerli hocam Sayın Prof. Dr. Mehmet Emin Dalkılıç'a teşekkürlerimi bir borç bilirim.

Bu çalışmayı, üzerinde çalıştığım süre boyunca bana sabır gösteren ve her zaman yanımda olan çok sevgili eşime ve manevi desteklerini hiçbir zaman esirgemeyen anneme ve babama ithaf ediyorum.

# İÇİNDEKİLER

Sayfa

# İÇİNDEKİLER (devam)

Sayfa

# İÇİNDEKİLER (devam)

Sayfa

# ŞEKİLLER DİZİNİ

Şekil Sayfa

# ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

# ŞEKİLLER DİZİNİ (devam)

Şekil Sayfa

# ÇİZELGELER DİZİNİ

Çizelge Sayfa

# SİMGELER VE KISALTMALAR DİZİNİ

| Simgeler | Açıklama |
|---|---|
| $c$ | Küme |
| $d$ | Beklenen fiziksel komşu sayısı |
| $d'$ | Yerel bağlantı sayısı |
| $f$ | *floating* komşu sayısı |
| $f(\ )$ | Tek yönlü karıştırma (*hash*) fonksiyonu |
| $h$ | Bağlantı ağaç yüksekliği |
| $idx$ | Anahtar havuzundaki anahtar sıra numarası |
| $k$ | Düğümlere yüklenen anahtar listesi |
| $m$ | Radyo iletişim menzili |
| $n$ | Toplam ağ eleman sayısı |
| $p$ | Komşuların ortalama RSSI değerleri (Bölüm 2) |
| $p$ | En az bir ortak anahtara sahip olma olasılığı (Bölüm 3) |
| $r$ | Üstel rastsal sayı |
| $\tau_c$ | Kümeleme zamanı |
| $\tau_l$ | Hat gecikme zamanı |
| $\tau_n$ | Komşu keşif zamanı |

## SİMGELER VE KISALTMALAR DİZİNİ (devam)

| Simgeler | Açıklama |
|---|---|
| $u, v, w$ | Duyarga düğümleri |
| $C$ | Enerji maliyeti |
| E [ ] | Beklenen değer |
| $K_{pk}$ | Şifrelenmiş patika anahtarları |
| $K_i$ | $i$ düğümünün patika anahtarı |
| $K_{a,b}$ | $a$ ve $b$ düğümlerinin paylaştığı ikili oturum anahtarı |
| $K'_{a,b}$ | $a$ ve $b$ düğümleri arasındaki kimlikleme anahtarı |
| $K_{cluster}$ | Küme ortak anahtarı |
| $L$ | Mesaj uzunluğu |
| $M$ | Mesajlaşma maliyeti |
| $P$ | Temel Şema anahtar havuzu büyüklüğü |
| $P_c$ | Genel ağ bağlantı oranı |
| $Pu_i$ | $i$ düğümünün açık anahtarı |
| $P_i$ | $i$ düğümünün özel anahtarı |
| $\beta$ | Küme eleman sayısı sınırı |
| $\lambda$ | Üstel rastsal sayı üreteç değeri |

# SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar

| | |
|---|---|
| AES | Gelişmiş Şifreleme Standardı |
| BFS | *Breadth First Seach* |
| CA | Sertifika Otoritesi |
| ECC | *Elliptic Curve Cryptography* |
| ECDH | *Elliptic Curve Diffie-Hellman* |
| ECDSA | *Elliptic Curve Digital Signature Algorithm* |
| GKA-1 | Ana *Gateway* Karar Algoritması |
| GKA-2 | Alternatif *Gateway* Karar Algoritması |
| GPS | Küresel Konumlama Sistemi |
| ID | Özgün kimlik numarası |
| IEEE | Elektrik ve Elektronik Mühendisleri Enstitüsü |
| LEAP | Yerel Şifreleme ve Kimlikleme Protokolü |
| MAC | Mesaj Kimlikleme Kodu |
| Mitm | Ortadaki Adam (*man in the middle*) |
| MST | En Az Yayılan Ağaç |
| J | Joule |

## SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar

| | |
|---|---|
| RSA | Rivest, Shamir, Adleman |
| RSSI | Alınan Radyo Sinyal Gücü |
| Rx/Tx | Alıcı/Verici |
| SHA-1 | Güvenli Karıştırma Algoritması |
| SPINS | Duyarga Ağları için Güvenlik Protokolleri |
| TK | Tekrarlı Kümeleme |
| TŞ | Temel Şema |
| WSN | Kablosuz Duyarga Ağları |
| Ws | Watt.saniye |
| ZTK | Zamanlayıcı Tabanlı Kümeleme |

# 1 GİRİŞ

Kablosuz Duyarga Ağları (WSN-Wireless Sensor Networks), belirli bir alana yayılmış çok sayıdaki kablosuz duyarga düğümünün o alanda vuku bulan bir olay hakkında duyargaları ile elde ettikleri veriyi ağı sorgulayabilen bir ana istasyona kablosuz hat üzerinden etkin şekilde ulaştırmasını hedefler. Ağı oluşturan duyarga düğümleri, son dönem teknolojik gelişmelerin paralelinde küçük boyutlarda üretilebilen, işlemci, hafıza ve enerji bakımından kısıtlı kaynaklara sahip ve kablosuz bağlantı kurabilen elektronik aygıtlardır. Bu aygıtlar sivil veya askeri amaçlı olarak bulundukları alandaki nicelikleri duyargaları ile algılar ve bu değerleri iletilebilir verilere dönüştürürler. Pil ile çalıştıkları için açık alanlarda tek başına çalışmak üzere bırakılabilirler ve bu özellikleri ile insan erişiminin tehlikeli olduğu veya mümkün olmadığı alanlarda kullanımları mümkündür. Isı, nem, hareket, basınç, gürültü, cisim varlığı ve hareketi denetleme gibi nicelikleri izleyebilen bu düğümlerin kullanılabileceği bazı uygulamalar aşağıdaki gibidir:

- Askeri uygulamalar: Dost kuvvetleri ve ekipmanı izleme, savaş alanı kontrolü, hedefleme, savaş zayiatı belirleme, nükleer, biyolojik veya kimyasal saldırı algılama.

- Çevre uygulamaları: Orman yangını algılama, biyolojik harita çıkarılması, akıntı algılama/izleme.

- Sağlık uygulamaları: Hastanedeki hasta ve doktorların izlenmesi, hastanelerdeki ilaç yönetimi.

- Ev uygulamaları: Ev otomasyonu.

- Ticari uygulamalar: Ofis binalarında çevre kontrolü, etkileşimli müze, araba hırsızı yakalama/görüntüleme, araç izleme ve algılama.

Duyarga düğümlerinin, büyük ölçekli ağ oluşumu için, genellikle uçak veya helikopter gibi araçlardan rast gele atılarak hedef alana, çoğunlukla toprağın

üstüne, yerleştirildiği kabul edilir. Bu şartlar altında elde edilen verilerin karar verici mercilere ulaştırılması için gerekli olan altyapının oluşturulması ve iletilen verinin gizlilik derecesine göre iletimde güvenliğin sağlanması literatürde incelenen önemli konulardandır. Kısıtlı kaynakların varlığı kablosuz duyarga ağlarını diğer kablosuz ağlardan ayıran en temel özelliktir. Bu nedenle kablosuz ağlar için geliştirilen güvenli ağ yönetimi yöntemlerinin etkin şekilde kablosuz duyarga ağlarında da uygulanması her zaman mümkün değildir (Akyıldız et al., 2002).

Geniş ölçekli duyarga ağlarında iletişim altyapısının oluşturulması için ağın alt gruplara ayrılması ve bu gruplar ile iletişimin güvenli yapılması ile ilgili güvenli grup haberleşmesi adı altında literatürde çeşitli çalışmalar yapılmıştır. Bu çalışmalar genel olarak ağı katmanlı kümelere/bölümlere ayırma, oluşan kümelerde ve tüm ağda güvenli iletişim için kullanılacak oturum anahtarlarını belirleme ve bu güvenlik altyapısını kümelerdeki eleman üyelik işlemlerine göre güncelleme üzerinedir. Bu çözümlerin bazıları kümeleri/grupları oluşturan düğümlerin hesaplama ve haberleşme menzili açısından diğer düğümlere göre daha üstün olduğunu varsaymış (Thepvilojanapong et al., 2004; Ghosh et al., 2006; Sun et al., 2007), bazıları da ağda düğümler arasında ayrım varsaymamış fakat kümeler arası haberleşmeyi sağlayan küme düğümlerinin birbirleri ile doğrudan haberleşerek ana istasyona doğru çok zıplamalı bir hat oluşturabildiğini varsaymıştır (Wadaa et al., 2004). Bu çalışmalarda genel olarak güvenli iletişim için kullanılacak ortak anahtarlar kümeleri/grupları oluşturan düğümler tarafından veya ana istasyonun da dahil olduğu üst seviye hatlar üzerinden ve hatta ana istasyonun kendisi tarafından doğrudan belirlenip dağıtılmaktadır. Fakat bu tür yapılarda anahtar dağıtımı yapan özel düğümlerin varlığı sistemi tek nokta saldırılarına karşı zayıf duruma düşürmektedir. Ayrıca tüm ağ elemanları arasında paylaşılan ortak bir anahtar varsayıldığında ağda ele geçirilen bir düğüm nedeni ile tüm ağ güvenliğinin açığa çıkmasına neden olunabilir. Ana istasyonla paylaşıldığı varsayılan bire bir simetrik özel anahtarlar üzerinden kimlikleme yapılarak ağın yapılanmasını varsayan çözümlerde (Thepvilojanapong et al., 2004) ise yerleşimden sonra ağa her düğüm ekleme işlemi sırasında ana istasyona bilgi verilmesi ve bu düğümün geçerli bir düğüm olduğuna dair onay alınması gerekmektedir. Bu da yapılanma sırasında ağdaki iletişim maliyetlerinde ciddi bir

artışa neden olur. Özellikle geçersiz düğümler ile ağa bağlanma isteği gönderen saldırganlar sadece tek zıplamada iletişim kurdukları düğümün enerjisini değil iletim hattındaki tüm düğümlerin enerjisini de tüketirler. Bu nedenlerden dolayı güvenli grup haberleşme sistemi oluşumunun ilk aşaması olan iletişim altyapısının kurulmasında öncelikli olarak özel duyarga düğümlerinin varsayılmaması ve güvenli iletim için kullanılacak ortak anahtarların dağıtık şekilde yerel olarak belirlenmesi gereklidir. Bu şekilde eş duyarga düğümlerinden oluşan ağda düğümlerin alana rast gele yerleştirilmesi nedeni ile yerleşimden sonra ağın kendi kendine organize olması gerekmektedir. Diğer taraftan bu organizasyonun nasıl yapılacağı belirlenirken düğümler arası iletişimde çevre koşullarının etkisiyle oluşabilecek kısıtlamalar da önemlidir. Örneğin, duyarga düğümlerinin yerleştirildiği alandaki yüzey şekilleri ve hatta yeryüzünün kendisi radyo iletim mesafesinde ciddi değişikliklere sebep olabilir. Bu nedenle ağ organizasyonunu hedefleyen bir protokolün duyarga düğüm kısıtlarının yanında çevresel faktörleri de dikkate alması gerekir.

Bu bağlamda büyük ölçekli kablosuz duyarga ağlarında güvenli grup haberleşme altyapısının oluşturulması için öncelikli olarak ağı oluşturan düğümler arası bağlantıların tanımlanması ve ağın yapılandırılması gerekmektedir. Bu yapının oluşturulmasında ölçeklenebilirlik ve performans için katmanlı kümeleme yapısının gerekliliği daha önceki çalışmalarda ifade edilmiştir (Younis et al., 2006; Abbasi and Younis, 2007). Duyarga ağlarında kümeleme işlemi ağı oluşturan düğümlerin belirli kurallara göre birbirlerine bağlanması ile gerçekleştirilir. Her kümenin bir küme başı vardır ve küme üyeleri algıladıkları veriyi bu küme başına yönlendirir. Küme başları arasında ayrıca bir üst seviye bağlantı hattı vardır ve küme başlarına ulaşan veri bu hatlar kullanılarak ana istasyona ulaştırılır. Katmanlı kümelemenin altyapıya sağladığı ana katkı yapılanmada ölçeklenebilirliğin sağlanması ve duyarga düğümleri arası veri iletiminde kullanılan yönlendirme tablolarının azaltılmasıdır. Ayrıca küme içi veri yönetimi sayesinde algılanan veriler kümeye ait düğümler tarafından bütünleştirilerek küme başına iletilir ve ağın kalanına sadece bu veri yönlendirilir. Bu sayede ağdaki gereksiz ve tekrarlı veri iletimi dolayısı ile bant genişliği kullanımı azaltılır (Abbasi and Younis, 2007). Literarürde kablosuz duyarga ağları için geliştirilen kümeleme protokolleri genel olarak ağı kümelere ayırırken küme

üyeleri ile küme başı arasıda çok zıplamalı veya tek zıplamalı iletim hatları oluşturmaktadır. Daha sonra küme başları arasında doğrudan iletim hattı kurulabildiği varsayılarak ana istasyona doğru ikinci seviye bir bağlantı hattı oluşturulur. Oysa ki Bölüm 2.1'de detaylı olarak belirtildiği gibi kısıtlı kaynakları olan duyarga düğümlerinin iletim mesafeleri sınırlıdır ve bu nedenle küme başları arasında oluşturulacağı varsayılan üst seviye haberleşme hattının bu tür sınırlı özellikte duyarga düğümleri içeren bir ağda uygulanabilirliği tartışmalıdır.

Güvenli grup haberleşme altyapısının oluşturulmasında ikinci olarak iletilecek mesajların şifrelenmesi dolayısı ile haberleşen düğümlerin mesajları gizli şekilde iletebilmesi için bir ortak anahtar belirlemesi gerekmektedir. Kısıtlı işlemci ve enerji kaynakları nedeni ile duyarga ağlarında düğümler arasında paylaşılacak anahtarların yerel olarak belirlenebilmesi için önerilen protokoller çoğunlukla hesaplama maliyeti en düşük olan ön dağıtım tabanlı simetrik anahtar belirleme protokolleridir (Xiao et al., 2007). Bunun yanında açık anahtar belirleme yöntemleri de son dönemde incelenmiş ve hesaplama maliyeti fazla olmasına rağmen kablosuz duyarga ağlarında uygulanabilirliği üzerine çalışmalar yapılmıştır (Wander et al., 2005; Piotrowski et al., 2006; Liu and Ning, 2008). Bu çalışmalarda anahtar belirleme protokolleri duyarga ağının yapılanmasından bağımsız olarak analiz edilmiş ve buna göre bir duyarga düğümünün alana yerleştikten sonra bağlantı kurabileceği tüm düğümler ile dağıtık şekilde ortak anahtar belirleyebilmesine çalışılmıştır. Oysa, ağın yapılanma gereksinimleri ile birlikte düşünüldüğünde bu anahtar belirleme protokollerinin analizi için yapılan varsayımların ağın çalışma yapısına uygulanabilirliği tartışmalıdır. Örneğin, anahtar dağıtımı protokolleri tarafından varsayılan yüksek (>8) düğüm derecesi (*tek zıplama mesafesinde bağlantı kurulabilen komşu sayısı- ağ yoğunluğu*) belli bir alandaki kablosuz ortam paylaşımını doğrudan etkileyeceği için veri iletiminde etkinliğin azalmasına sebep olacaktır (Hwang and Kim, 2004). Ayrıca yüksek ağ yoğunluğu alandaki duyarga düğüm maliyetlerini de artıracaktır ki bu da bazı geniş ölçekli alan kapsama gereksinimi olan uygulamalar için kabul edilebilir olmayabilir. Tersi durumda da sınırlı kaynaklar nedeni ile ağ yapılanması sırasında ortaya çıkan kısıtlar sistemin güvenlik seviyesini etkileyebilir ve bu nedenle anahtar belirleme protokolünde yapılanma değişikliği yapılması gerekebilir. Anahtar belirleme protokolünde yapılan bu tür bir değişiklik duyarga

düğümlerinde daha fazla kaynak kullanımına sebep olabilir ve bu nedenle sistemin toplam maliyetini artırabilir. Dahası bu maliyet artışı sistemin ihtiyaç duyduğu güvenlik seviyesini de karşılamayabilir. Diğer taraftan anahtar belirleme ve ağ yapılanması arasındaki karşılıklı etkileşim toplam sistem maliyetini olumlu yönde de etkileyebilir. Örneğin, ağ yapılanması sırasında duyarga düğümleri arası kurulması gereken bağlantı sayısı anahtar belirleme protokol analizlerinde varsayılan bağlantı sayısından daha az olabilir ve bu da toplam iletim ve hesaplama maliyetlerinde indirim sağlayabilir. Dolayısıyla kablosuz duyarga ağlarında güvenli veri iletimi için seçilecek anahtar belirleme protokollerinin ağ yapılanması ile olan ilişkisinin ortaya konması ve analizlerinin bu şekilde yapılması gerekmektedir.

Bu tez çalışmasında, düşük ağ yoğunluklarında etkin çalışmak üzere, yerleşimden güvenli grup haberleşme altyapısının oluşturulmasına kadar geçen süreçteki kablosuz duyarga ağ organizasyonu ve bu organizasyonda güvenli iletim için anahtar belirleme işlemlerini tanımlayan ölçeklenebilir bir protokol tasarımı hedeflenmiştir. Bu protokol ağdaki tüm duyarga düğümlerinin eş özellikli, hareketsiz ve kendilerine özgü bir kimlik numarasına (*ID*) sahip olduğunu varsayar. Düğümler birbirlerine olan yakınlıklarını mesajlaşmalardaki RSSI (*Received Radio Signal Strength*) değerlerini kontrol ederek belirlerler.

Bu kapsamda ilk olarak eş özellikli düğümlerden oluşan geniş ölçekli duyarga ağlarının çok zıplamalı şekilde kendi kendine organize olmasını sağlayan Tekrarlı Kümeleme (TK) protokolü geliştirilmiştir. Protokolde kümeleme işlemi ilk olarak ana istasyon tarafından başlatılır ve tetiklenen yeni kümeler ile yayılan ağaç formuna uygun şekilde tekrarlı olarak ağ çevresine yayılır. Ana istasyon tarafından tetiklenen ilk küme dışındaki tüm yeni kümeler kendilerinden önce oluşturulan kümelerdeki ağ geçidi adı verilen seçilmiş duyarga düğümleri tarafından tetiklenir. Denk küme oluşumunda küme eleman sayısını sınırlandırmak için Krishnan ve Starobinski (2006) tarafından geliştirilen çok zıplamalı *persistent* kümeleme protokolü temel alınmıştır. Kümeleme işlemleri sırasındaki mesajlaşma maliyetlerini düşürmek için tez çalışması kapsamında *persistent* protokolüne bir de durum kontrol mekanizması eklenmiştir. Ağ geçidi olan düğümler tarafından yeni küme oluşumunun başlatılması sırasında kurulan

kümeler arası bağlantı hattı ile aynı zamanda küme başları arasında ana istasyona doğru yönlenmiş çok zıplamalı bir iletim hattı da oluşturulur.

Geliştirilen TK protokolünün performansı Krishnan ve Starobinski (2006) tarafından geliştirilen zamanlayıcı tabanlı kümeleme (ZTK) protokolü performansı ile benzetim yolu ile karşılaştırılmış ve avantajları ve dezavantajları ortaya konmuştur. Benzetim sonuçlarına göre özellikle 7 ve 14 gibi düşük duyarga düğüm derecesine sahip ağ yapılanmasında, oluşturulan küme adedine göre TK protokolü daha iyi performans göstermiştir. Zaman karmaşıklığı incelendiğinde, ağı oluşturan toplam eleman sayısı arttıkça TK ve ZTK protokollerinde tüm ağda kümeleme işlemlerinin tamamlanması için geçen sürenin logaritmik bir artış gösterdiği gözlemlenmiştir. Burada TK protokolü ZTK protokolüne göre daha hızlıdır ve daha çabuk yakınsar. Toplam mesajlaşma karmaşıklığı ise TK protokolünde bir miktar daha fazladır fakat bu fark ZTK protokolünde bulunmayan kümeler arası bağlantı hatlarının oluşumunu sağlayan ağ geçidi belirleme ve yeni küme başlatma işlemlerinden dolayı oluşmaktadır.

Tez çalışmasında ikinci olarak takrarlı kümeleme protokolünde küme içi ve kümeler arası güvenli mesaj iletimi için ikili ortak anahtarların belirlenmesinde kulanılabilecek olası anahtar belirleme protokolleri incelenmiştir. Bu protokoller ağın yapılanması sırasında TK protokolü ile birlikte çalışacak şekilde tasarlanmış ve karşılıkı performans ilişkileri ortaya konmuştur. İncelenen anahtar belirleme protokolleri simetrik anahtar yapısı kullanan anahtar ön dağıtımı ve asimerik anahtar yapısı kullanan açık anahtar belirleme tabanlıdır. Bu protokollerde ağ yapılanması sırasında ikili ortak anahtarların belirlenmesi için herhangi bir merkezi anahtar dağıtıcısına ihtiyaç yoktur ve anahtarlar yerel olarak belirlenir. Seçilen anahtar ön dağıtımı protokolü Eschenauer ve Gligor (2002) tarafından önerilmiştir ve Temel Şema olarak adlandırılır. Açık anahtar tabanlı anahtar belirleme analizi için ECDH (*Elliptic Curve Diffie-Hellman*) protokolü seçilmiştir. Çünkü ECDH kablosuz duyarga ağları için en uygun açık anahtar tabanlı protokoldür (Piotrowski et al. 2006). Bu analizlerde kümeleme protokolünün durum kontrol mekanizması kullanılarak ağ konfigürasyonuna bağlı *düz*, *reaktif* ve *proaktif* olmak üzere üç ayrı anahtar belirleme stratejisi geliştirilmiştir. Geliştirilen bu yöntemler kullanılarak anahtar belirleme

protokollerinin ağ yapılanması ile birlikte uygulanması sonucu elde edilen ağ bağlantı oranı, kümeleme performansı, toplam enerji maliyeti ve erişilen güvenlik seviyesi benzetim yolu ile analiz edilmiştir. Benzetim sonuçlarına göre, anahtar belirleme protokol maliyetlerinin ağ yapılanma protokolünün gereksinimlerine göre anahtar belirlenmesi gereken hat sayısının yeniden düzenlenmesi ile azaltılabileceği gösterilmiştir. Başka bir deyişle ağ yapılanmasının istenen seviyede gerçekleşebilmesi için tüm duyarga düğümlerinin fiziksel olarak komşu olduğu tüm duyarga düğümleri ile güvenli hat oluşturma zorunluluğu bulunmamaktadır. Sonuçta benzetim ile elde edilen toplam hesaplama ve haberleşme maliyetleri karşılaştırıldığında ECDH anahtar belirleme protokolünün TK protokolü ile uygulandığında daha iyi bir maliyet karakteristiğine sahip olduğu ve güvenlik seviyesinin de Temel Şema'ya göre yaklaşık üç kat daha yüksek olduğu belirlenmiştir. Ayrıca duyarga düğümlerinin ECDH ile anahtar belirleme işlemlerini yaparken birbirlerini kimlikleyebileceği RSA (*Rivest, Shamir, Adleman*) (Rivest et al, 1978) ve ECDSA (*Elliptic Curve Digital Signature Algorithm*) (Johnson et al., 2001) imza yöntemleri de maliyet olarak karşılaştırılmış, ECDH ve RSA hibrid kullanımının enerji etkin bir kimliklemeli güvenli iletişim altyapısı oluşturmak için en uygun seçenek olduğu ortaya konmuştur. Bu bölümde ayrıca kümelere ayrılmış ve paylaşılan ortak anahtarları belirlenmiş bir ağda eleman ekleme ve çıkarma durumlarında yapılacak anahtar güncelleme işlemleri de tanımlanmıştır ve maliyet analizleri ortaya konmuştur.

Tez çalışmasının ikinci bölümünde kablosuz duyarga ağları için önerilen kümeleme protokollerinin incelemesi ve TK protokolü ile ortaya konan çözümler, TK protokolünün yapılanma zaman analizi ve yapılanma performansının ortaya konması için ZTK protokolü ile karşılaştırmalı benzetim sonuçları verilmiştir. Üçüncü bölümde, seçilen anahtar belirleme protokollerinin TK protokolü ile birlikte uygulanması ve bu uygulamanın yapılanma performansı ve güvenliği üzerine incelemeleri bulunmaktadır. Bu bölümde tez çalışmasında incelenen anahtar belirleme protokollerinin TK protokolü ile çapraz seviye tasarım detayları ve performans analizleri ile birlikte benzetim sonuçları verilmiştir. Son bölümde ise yorumlar, katkılar ve yeni araştırma önerileri sunulmuştur.

# 2 TEKRARLI KÜMELEME (TK) PROTOKOLÜ

## 2.1 Motivasyon ve İlgili Çalışmalar

Kablosuz duyarga ağlarında duyarga düğümlerinin alana dağılımı ve ağdaki düğüm derecesi üzerinde yapılan varsayımlar geliştirilen yapılanma protokollerinin çalışma şeklini tanımlamaktadır. Bu çalışma şekli ağı kümelere ayıran yapılanma protokollerinde kümelerin oluşturulma yöntemini de belirler. Örneğin, duyarga düğümlerinin haberleşme yeteneği ile ilgili yapılan varsayımlar yapılanma sırasında oluşturulacak kümelerde hangi düğümlerin kümeye dahil edilebileceğini ve küme içi ve kümeler arası haberleşmede bağlantıların nasıl oluşturulabileceğini belirler. Bu konuda yapılan bazı çalışmalar kümeyi oluşturan ve yöneten küme başının küme üyelerine tek zıplama ile ulaşabildiğini varsaymıştır (Heinzelman et al., 2002; Younis and Fahmy, 2004). Diğer bazı çalışmalar da iletimdeki enerji sarfiyatını azaltmak ve bu sayede ağın ömrünü artırmak için küme içi çok zıplamalı haberleşmeye olanak vermiştir (Gupta and Younis, 2003; Chan and Perrig, 2004; Lindsey and Raghavendra, 2002; Xu and Gerla, 2002; Banerjee and Khuller, 2001). Kümeler arası haberleşme için çoğu kümeleme protokolü[*] küme başları arasında ikinci seviye çok zıplamalı iletişim hatları varsaymıştır. Bu seviyede küme başları birbirleri ile doğrudan haberleşebilir ve ana istasyona veri iletimi bu hat üzerinden yapılır. Bu protokollerin hiçbiri oluşturulan kümelerin sahip olabileceği eleman sayısı hakkında bir sınırlama getirmemiş ve haberleşme menzilindeki tüm duyarga düğümlerini küme içine dahil etmiştir. Younis ve Fahmy (2004), ağda küme başları arası bağlantının tamamlanabilmesi için küme başları arası iletim menzilinin en az iki küme çapına eşit olması gerektiğini belirtmiştir. Bu varsayım ağda küme başları arası doğrudan iletim varsayan bu tür protokoller için de geçerli olmalıdır.

Diğer taraftan, ağın işleyişi hakkında varsayımlar yapılırken rast gele yerleşimden sonra alandaki gerçek koşulların da dikkate alınması gerekmektedir. Aksi takdirde belirli uygulamalar için kümeleme algoritmalarının uygulanabilirliği

[*] Heinzelman et al., 2002; Gupta and Younis, 2003; Younis and Fahmy, 2004; Chan and Perrig 2004; Lindsey and Raghavendra, 2002; Xu and Gerla, 2002; Banerjee ve Khuller, 2001.

tartışmalı duruma düşebilir. Örneğin, ağ yapılanması için kümeleme ve haberleşme menzil gereksinimleri belirlenirken toprağın üzerine yerleşimin etkileri ve verimli radyo iletimi için gereken tek zıplama ile erişilebilecek komşu sayısı (düğüm derecesi, ağ yoğunluğu) parametrelerine dikkat edilmesi gerekmektedir. Duyarga düğümleri küçük cihazlar olduğundan toprak üstüne yerleştirildiklerinde kablosuz haberleşme kısımları da toprağa yakın olur. Holland ve ark. (2006), Tmote Sky (Sentilla Corporation, 2009) duyarga düğümlerinin toprağa yakın yerleştirilmeleri durumunda radyo haberleşme menzillerinin 13,716 metreye kadar düştüğünü deneysel olarak göstermiştir (Tmote Sky duyarga düğümlerinin özellik dokümanında belirtilen haberleşme menzili dış ortam için 125 metredir). Bu duyarga düğümünde bulunan kablosuz haberleşme ünitesi Chipcon (Texas Instruments, 2009) kablosuz alıcı/verici Mica2Dot (Crossbow Technologies, 2009) gibi yaygın kullanılan diğer duyarga düğümleri tarafından da kullanılmaktadır. Bu nedenle diğer duyarga tipleri için de haberleşme menzilinin bu seviyelerde olması beklenmelidir. Diğer taraftan, fiziksel duyarga düğüm derecesi dikkate alındığında radyo haberleşmesinin verimli olabilmesi ve ağ kapasitesinin en iyi kullanılabilmesi için teorik olarak önerilen en uygun değer 5 ile 8 arasıdır (Hwang and Kim, 2004). Bu nedenle bir duyarga düğümünün tek zıplama haberleşme menzilinde bulunan diğer düğümlerin sayısı bu civarda olmalıdır.

Bahsedilen haberleşme menzili ve yoğunluk kısıtları nedeni ile iki küme başı arasında doğrudan iletim hattı bulunduğunu varsayan protokoller büyük ölçekli ağlar için pratik değildirler. Örneğin, Younis ve Fahmy (2004) tarafından geliştirilen kümeleme protokolünde iki küme başı arasındaki haberleşme menzili en fazla 14 metre olabilir ve bu mesafenin en az iki küme çapına eşit olması gerekmektedir. Bu da küme çapının en fazla 7 metre olabileceği anlamına gelir ki bu çaptaki bir kümede tüm düğümler küme başı ile zaten doğrudan haberleşebilir. Fakat kablosuz ortam kullanımının etkin olabilmesi için bu çapın içindeki toplam düğüm derecesinin teorik olarak en fazla 8 olması gerekmektedir. Bu da demektir ki bir küme ancak 3,5 metre yarıçapındaki bir daire içinde yaklaşık 9 veya daha az adet düğümden oluşmalıdır. Diğer taraftan bu tür bir kablosuz iletişim menzili sınırlamasına karşın ağda özel düğümlerin varlığı varsayılabilir fakat bu düğümler ağı tek nokta hatasına açık hale getirir ve fiziksel olarak diğerlerinden ayırt

edilebileceklerinden ağın yapısını bozmak isteyen bir saldırgan için çekici hedef haline gelirler. Ayrıca haberleşme menzilinde çok fazla (>>8) düğüm olabileceğinden kablosuz ortam kullanımındaki etkinlik de azalır.

Sonuç olarak ağda eş duyarga düğümleri varsayıldığında büyük ölçekli ağlar için küme eleman sayısını ve kapsama alanını kısıtlamayacak çok zıplamalı bir haberleşme altyapısına ihtiyaç vardır. Bu altyapı haberleşme menzili ve yoğunluk kısıtlarına göre kümeler arası çok zıplamalı haberleşme hatlarının kurulumunu da tanımlamalıdır. Çoklu zıplama ile bir kümenin kapsadığı alan haberleşme kısıtlarına rağmen tek zıplamaya göre daha büyük olacaktır. Literatürdeki çoklu zıplamalı küme oluşumu hedefleyen bu tür protokollerden Banerjee ve Khuller (2001), ağ yapılandırmasını BFS (*Breadth First Seach*) ağacı oluşturma üzerine geliştirmiştir. Bu çalışmada temel olarak ağ yapılanması bir düğüm tarafından başlatılır ve bu yapılanma içindeki herhangi bir düğüm kök düğüme en yakın olan komşusunu ağaçtaki üst bağlantısı olarak belirler ve bu düğüme bağlanmaya çalışır. Küme boyutunu sınırlandırmak için yapılan işlemler ağaç oluşturulduktan sonra yapıldığı için kümelemedeki mesajlaşma karmaşıklığı yüksektir. Krishnan ve Starobinski (2006), sınırlı eleman sayısına sahip küme oluşumu için üç adet kümeleme protokolü tanımlamıştır; *expanding ring*, *rapid* ve *persistent*. *Rapid* ve *persistent* protokolleri küme boyutunu sınırlandırmak için sonradan işlem gerektirmediğinden mesajlaşma maliyetleri *expanding ring* protokolünden daha iyidir. Bu protokollerde küme oluşumu ağ içerisinde seçilen bir düğüm tarafından başlatılır ve belirlenen küme eleman sayı sınırını geçmeyecek şekilde yeni düğümler kümeye eklenir. Bu kümeleme protokolleri içerisinde *persistent*, mesajlaşma karmaşıklığı en az olan protokoldür. Krishnan ve Starobinski (2006), tüm ağın kendi kendine organizasyonu için ayrıca zamanlayıcı tabanlı bir kümeleme protokolü önermiştir. Bu protokolde *persistent* algoritması ile oluşturulan kümelerin küme başlatıcıları rastsal zamanlayıcılara göre belirlenir ve kümeler uzaysal ve zamansal olarak ayrık şekilde oluşturulmaya çalışılır. Bu protokole göre her bir duyarga düğümü yerleştirilmeden önce başlatıcı zamanlayıcısını çalıştırmaya başlar. Bu zamanlayıcının sınırı üstel rastsal sayı yöntemine göre belirlenir ve bu rastsal sayı üretimi için seçilen $\lambda$ değeri hedeflenen küme eleman sayısı sınırı $\beta$'ya ve $\tau_c$ küme oluşum zamanına bağlı olarak $1/\beta\tau_c$ değerine eşit veya küçüktür. Fakat bu protokolde kümeler

oluşturulurken kümeler arası bağlantı hatları belirlenmez ve bunu için yapılanma sonrasında ayrıca işlemler yapılması gerekmektedir.

Tez kapsamında geliştirilen TK protokolünde ise sınırlı kaynaklara sahip eş düğümler üzerinden sadece komşuluk bilgileri kullanarak sınırlandırılmış eleman sayısına sahip denk kümelerin oluşumu sağlanmaktadır ve yapılanma tamamlandığında kümeler arası bağlantı hattı da oluşturulmuş olmaktadır. Tasarlanan yapılanma protokolünün detayları ilerleyen bölümlerde verilmiştir.

## 2.2 TK Protokol Tasarımı

Tez kapsamında geliştirilen TK protokolü yayılan ağaç temellidir. Protokolde tek küme oluşumu Çizelge 2-1'de detayı verilen geliştirilmiş *persistent* protokolü ile gerçekleştirilir. Ağın yapılanması sırasında kümelerin oluşumunu başlatan düğümlerin seçimi yerel karar yöntemleri ile yapılır. Protokolde temel olarak seçilen bir başlatıcı düğüm kendi komşu düğümlerini kümesine ekleyerek genişlemeyi başlatır ve eklenen düğümler de uygun durumdaki kendi komşularını ekleyerek küme genişlemesini devam ettirir. Burada komşuların belirlenmesi de protokolün bir parçasıdır. Bir küme oluşumu tamamlandığında veya oluşturulurken, belirli bir algoritmaya göre seçilen yaprak düğümler (ağ geçidi - *gateway*) diğer kümeleri oluşturacak düğümleri (başlatıcı - *initiator*) seçerler. Dolayısı ile kümeleme hızı yeni kümeler eklendikçe artar çünkü her yeni küme ile birden fazla başlatıcı oluşturulabilir. Buradaki önemli nokta ağ geçidi seçiminin aynı alanda birden fazla kümenin oluşumuna izin vermeyecek şekilde yapılmasıdır.

Temel protokol işleyişi duyarga düğümlerinin alana yerleşimi tamamlandıktan sonra başlar. Ağ yapılandırılırken düğümlerde ve düğümler arası haberleşmede hata oluşmadığı varsayılmıştır. Yapılandırma işlemi ana istasyonun ilk kümeyi oluşturacak başlatıcı düğümü seçmesi ile başlar. Bu seçim için ana istasyon tarafından ağdaki en yakın duyarga düğümlerine bir yoklama mesajı bütüneyayılır. Bu mesaja cevap veren düğümlerin içinden RSSI değeri en yüksek olan ilk düğüm küme başlatıcısı olarak seçilir. Seçilen bu düğüm ana istasyondan yapılanma mesajı alır ve başlatıcı olur. Yeni başlatıcı hemen küme içi genişlemeyi

başlatır. Bu genişleme Çizelge 2-1'de detayı verilen geliştirilmiş *persistent* algoritmasına göre yapılır ve genişleme işlemi genişleyecek düğüm kalmayana veya küme eleman sayı sınırına ulaşılıncaya kadar devam eder.

**Çizelge 2-1**: Düzenlenmiş *persistent* algoritması.

**Assumptions:** Network is $G = (V;E)$, there is no node failure during configuration

```
 1: for all nodes v ∈ V do
 2: BEGIN:
 3:   while RECEIVE cluster extension message from x do
 4:     A ← received_budget
 5:     if state is floating and A-1 equals to NULL then        /* end node so is gateway candidate*/
 6:       parent ← x
 7:       state ← gateway_c
 8:       SEND total subtree size ← 1 to parent
 9:       BROADCAST poll_gateway for neighbor discovery and start neighbor_discovery_timer
10:       RECEIVE poll_gateway_reply and update neighbor states and RSSI levels
11:       WAIT neighbor_discovery_timer out
12:       RUN gateway decision algorithm
13:       if state is gateway then
14:         goto END
15:       else
16:         goto BEGIN
17:       endif
18:     else if state is floating and A equals to max_cluster_ bound then
19:       parent ← x
20:       state ← initiator                    /*starts new clustering with maximum budget */
21:       BROADCAST poll_neigh for neighbor discovery and start neighbor_discovery_timer
22:       RECEIVE poll_neigh_reply and update neighbor states and RSSI levels
23:       WAIT neighbory_discovery_timer out
24:       if there is another initiator in neighbor list having bigger ID number then
25:         state ← node
26:         goto END
27:       else
28:         β ← A−1, δ ← neighbors(v)
29:       end if
30:     else if sate is floating and A is less than max_cluster_ bound then
31:       state ← node          /*intermediate node continues clustering with associated budget*/
32:       BROADCAST poll_neigh for neighbor discovery and start neighbor_discovery_timer
33:       RECEIVE poll_neigh_reply and update neighbor states and RSSI levels
34:       WAIT neighbor_discovery_timer out
35:       parent ← x, β ← A−1, δ ← neighbors (v)
36:     else if sate is node and message is received from parent then
37:       /*reextension is requested for an already clustered node with extra budget*/
38:       parent ← x, β ← A, δ ← neighbors (v)
39:     else if sate is node and message is not received from parent then
40:       goto BEGIN
41:     end if
42:     while β > 0 and δ > 0 do
43:       γ ← ⌊β / δ⌋
44:       SEND: γ +1 budget to β mod δ selected neighbors and γ budget to other available
                neighbors whose states are floating or node (same cluster)
45:       RECEIVE total subtree size from the neighbors
46:       β ← β − total subtree size
47:       δ ← neighbors(v)
48:     end while
49:     if state is initiator then
50:       cluster size ← total subtree size
51:     else
52:       SEND total subtree size to parent
53:     end if
54:   end while
55: END:
56: end for
```

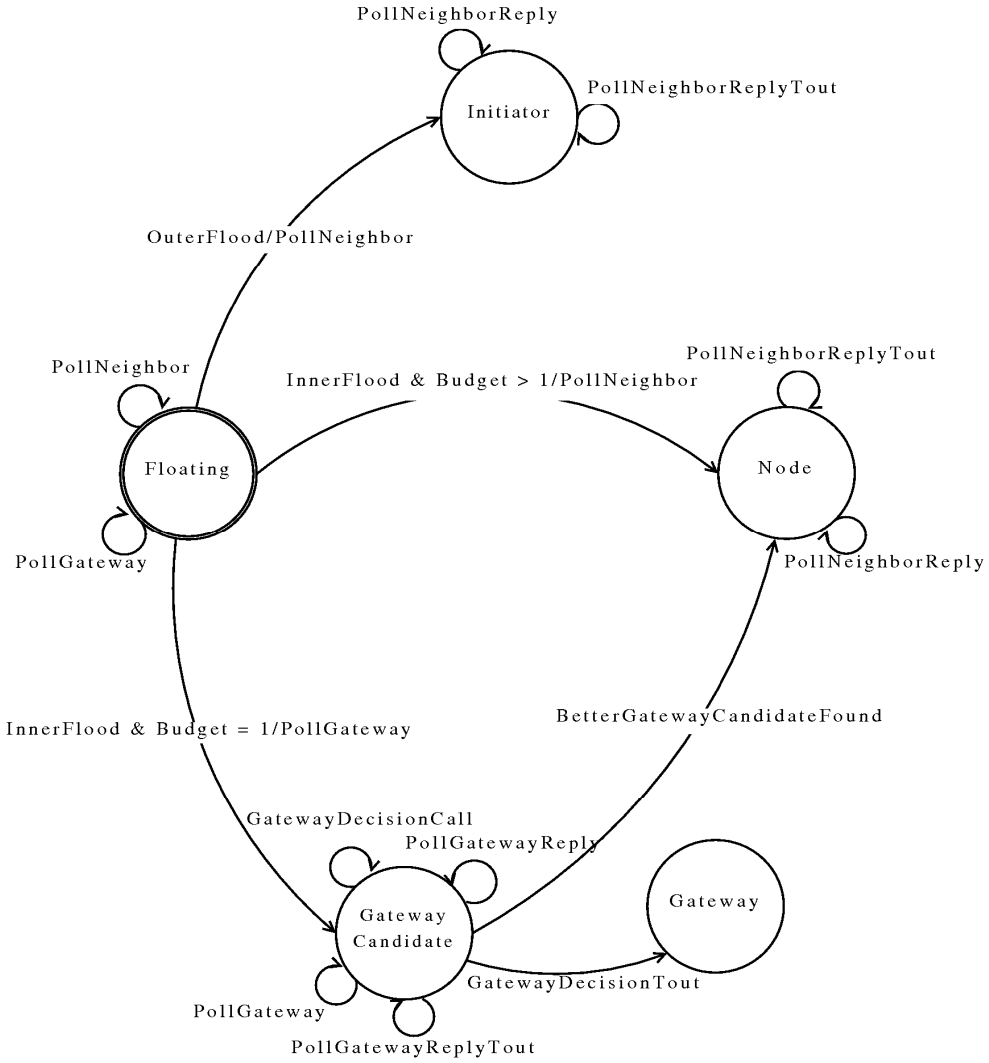
Küme oluşturulurken yayılan ağacın uç kısımlarında bulunan düğümler yeni kümeleri oluşturacak başlatıcıları seçecek adaylar olarak işaretlenirler. Bu adayların içinden hangisinin ağ geçidi olacağına karar verebilmek için iki ayrı algoritma geliştirilmiştir. Bu algoritmaların ilkinde karar komşuların sayısı ve RSSI ortalaması değerlerine göre verilir, ikincisinde ise bütüneyayım maliyeti daha az olan zamanlayıcı tabanlı bir çözüm sunulmuştur.

TK protokolüne göre ağda bir duyarga düğümü için beş durum söz konusudur. Bu durumlar duyarga düğümlerinin ağ yapılanmasındaki tiplerini de belirler. Başlangıçta tüm düğümler yapılanmayı beklemek üzere *floating* durumundadır. Yapılanma sırasında bir düğüm *initiator* (başlatıcı), *node* veya *gateway* (ağ geçidi) durumlarından birine sahip olabilir. Ağ geçidi olmaya aday duyarga düğümleri için ara durum olarak *gateway_c* tanımlanmıştır. Eğer durumu *gateway_c* olan bir düğüm seçim sonrasında ağ geçidi olamazsa durumunu *node* olarak değiştirir.

### 2.2.1 Protokol sonlu durum makinesi işleyişi

Alana yerleştirildikten sonra ağdaki tüm duyarga düğümleri Şekil 2-1'de basitleştirilmiş detayı verilen protokol sonlu durum makinesini çalıştırırlar. Buna göre başlangıçta tüm düğümler *floating* durumundadır ve yapılanma sırasında yerine getirmeleri gereken işlemi tanımlayacak mesajı beklemektedirler. *Floating* durumda olan herhangi bir duyarga düğümü komşu belirleme veya yapılanma için *PollNeigh, PollGateway, OuterFlood, InnerFlood* mesajlarından birini alabilir. *PollNeigh* mesajı alan düğüm gelen mesajın RSSI değeri belli bir limitin üzerinde ise kendi durum bilgisi ile yanıt verir. Bu RSSI sınır değeri yerleşim alanında iletişime olanak veren en zayıf radyo seviyesi olarak belirlenebilir. *PollGateway* mesajı yapısal olarak *PollNeigh* ile aynıdır fakat gönderen düğüm bir ağ geçidi adayıdır. *PollNeigh* mesajı ise *node, initiator* veya *gateway* düğümlerinden gelebilir. Alınan *PollGateway* mesajının RSSI değeri limitin üzerinde ise mesajı alan düğüm yoklamaya yanıt verir. Bu yoklama mesajları komşuları belirleme amacı ile yapıldığı için alıcı düğümlerde durum değişikliğine sebep olmazlar.

Eğer gelen mesaj *OuterFlood* ise *floating* düğümü yeni küme oluşumu için seçilmiş demektir ve durumunu *initiator* olarak değiştirerek küme yapılandırmasına başlar (Çizelge 2-1, Satır 18-29). *InnerFlood* mesajı yapısal olarak *OuterFlood* mesajı ile aynıdır fakat bu mesaj sadece küme içi genişleme için kullanılır. Bu mesajla alınan bütçe değerine göre mesajı alan düğüm durumunu *node* veya *gateway_c* olarak değiştirir (Çizelge 2-1, Satır 5-17; 30-35).




**Şekil 2-1:** TK protokolü basitleştirilmiş sonlu durum makinesi.

*InnerFlood* mesajı alan *floating* düğümler eğer alınan bütçe değeri 1'den büyük ise durumlarını *node* olarak değiştirirler. Alınan *InnerFlood* mesajı küme içi genişleme yapılması gerektiği anlamına gelmektedir. Mesajı alan düğüm kendisini mesajı yollayan düğümün kümesine dahil eder ve elindeki bütçeye göre genişleme amacı ile komşularını belirlemek için *PollNeigh* mesajını bütüneyayar.

Bu mesajdan sonra çevresindeki komşulardan *PollNeighReply* mesajını komşu keşif zamanı dolana kadar bekler. *Node*, alınan her *PollNeighReply* mesajındaki bilgi ile komşu listesini günceller. Zaman dolduğunda da komşu listesindeki durumu *floating* olan düğümlere ilgili genişleme bütçe değerleri *InnerFlood* mesajı ile yollanır (Çizelge 2-1, Satır 30-35).

Eğer alınan bütçe değeri 1 ise *InnerFlood* mesajları alan *floating* durumundaki düğümler durumlarını *gateway_c* olarak değiştirirler. Bu düğümler önce kendilerini mesajı yollayan düğümün kümesine dahil ederler ve sonra *PollGateway* mesajını bütüneyayarak komşularını bulmaya çalışırlar. *PollGateway* mesajı sadece komşuları belirlemek için değil aynı zamanda yakındaki diğer ağ geçidi adaylarının varlığını kontrol etmek için de bütüneyayılır. Komşu keşif zamanı dolana kadar ağ geçidi adayı komşularından *PollGatewayReply* mesajı bekler. Alınan her *PollGatewayReply* mesajı ile komşu listesi güncellenir. Zaman dolduğunda ağ geçidi adayı durumunu *gateway* veya *node* yapacak kararı verme işlemine başlar (Çizelge 2-1, Satır 5-17). Bu karar işlemi protokolde küme yapılanma performansını etkileyen en önemli kısımdır. Eğer birbirine yakın iki adet ağ geçidi oluşturulursa bu birden çok küme başlatıcısının aynı bölge içinde küme oluşturmak için yarışmasına sebep olacaktır. Bu da toplamda üretilen küme sayısını artıracak ve ortalama küme büyüklüğünü düşürecektir. Bu nedenle de kümeleme performansı düşecektir. Eğer seçilen ağ geçidinin bulunduğu bölgede genişlemeye uygun komşular bulunmuyor ise kümeleme operasyonu devam edemez ve ağda ayrık düğümlerin artmasına neden olunur. Benzetimlerde kullanılan detaylı protokol durum makinesinin işleyişi EK-A'da verilmiştir.

İlerleyen alt bölümlerde *gateway* kararı için tez kapsamında geliştirilen iki algoritmanın detayı verilmiştir.

#### 2.2.1.1 Ana *gateway* karar algoritması (GKA-1)

Çizelge 2-1'de 12. satırda başlayan *gateway* karar işlemi, ağ yapılanmasında protokol kümeleme performansını etkileyen en önemli işlemlerden biridir. Çünkü yapılanma sırasında eğer birbirine yakın en az iki ağ geçidi oluşturulur ise aynı

düğüm alanı içerisinde birden fazla başlatıcı küme oluşturmak için birbirleri ile yarışır. Bu da oluşturulan toplam küme sayısını artırır ve ortalama küme büyüklüğünü düşürür. Dolayısıyla da kümeleme performansı düşer. Diğer taraftan eğer seçilen ağ geçidinin bulunduğu bölgede küme genişlemesi için ihtiyaç duyulan yeterli sayıda *floating* komşu yok ise kümeleme devam edemez ve bu da bazı yerlerde ayrık düğümlerin kalmasına sebep olur.

Ağ geçidi belirlemede en uygun kararı verebilmek için ilk olarak *floating* komşu sayısı (*f*) ve bu komşuların ortalama RSSI değerlerinin (*p*) kontrolü üzerine dayalı *greedy* bir algoritma geliştirilmiştir (Bkz. Çizelge 2-2). Bu algoritmada *gateway* kararı aday düğümlerin kendi *f* ve *p* parametrelerini diğer adaylardan aldıkları parametreler ile karşılaştırması ile yerel şekilde alınır. Algoritmaya göre, bir düğüm durumunu *gateway_c* olarak değiştirip komşularını belirledikten sonra (Çizelge 2-1, Satır 5-11) haberleşme menzili içindeki diğer aday düğümler ile parametrelerini karşılaştırabilmek için ikinci bir yoklama mesajı yollar ve karar zamanlayıcısını çalıştırır. Bu yoklama mesajı *gateway_decision_call* olarak adlandırılır ve gönderen düğümün *f* ve *p* parametrelerini içerir (Çizelge 2-2, Satır 3). Bu mesajı alan herhangi bir ağ geçidi adayı kendi *f* ve *p* parametreleri ile mesajdakileri Çizelge 2-2, 8. satırda verilen koşula göre karşılaştırarak durumunu *node* olarak değiştirip değiştirmeyeceğine karar verir. Eğer parametreleri daha iyi ise durumunu değiştirmez ve mesajı yollayan düğüme *gateway* belirleme işlemini durdurması için kendi parametrelerini içeren *gateway_decision_reply* mesajını yollar (Çizelge 2-2, Satır 11). Durumunu *node* olarak değiştiren herhangi bir ağ geçidi aday düğümü karar işlemini de durdurur ve hiçbir işlem yapmaz (Çizelge 2-2, Satır 9). Ağ geçidi adayı, karar sürecini kontrol eden *gateway* karar zamanlayıcısı sıfırlanana kadar başka bir ağ geçidi adayından daha iyi parametreli bir *gateway_decision_call* veya *reply* mesajı almadığı durumda kendisini *gateway* olarak ilan eder ve yeni küme oluşumu için başlatıcı seçme çalışmalarını başlatır (Çizelge 2-2, Satır 14-17). Bu aşamada *gateway* düğümü *floating* durumda olan komşularından birini yeni kümenin başlatıcısı olarak belirler ve bu düğüme *OuterFlood* mesajını yollar.

Özetle *gateway* seçimi için belirlenen temel karar mekanizması ilk aşamada *gateway* adaylarının komşularını dolayısı ile *f* ve *p* parametrelerini belirlemesini

gerektirir. Sonrasında ağ geçidi adayları yoklanarak sahip oldukları *f* ve *p* parametrelerini paylaşır ve karşılaştırma yolu ile ağ geçidi olup olmama kararlarını verirler. Karardan sonra *floating* komşu sayısı en fazla olan en iyi ağ geçidi adayı haberleşme mesafesinde olan tüm diğer ağ geçidi adayları arasından seçilir ve *gateway* olarak belirlenir. Bu algoritma aynı kümede bulunan iki ağ geçidi arasındaki muhtemel uzaklığı artırır. Çünkü haberleşme menzilinde bulunan adayların içinden sadece bir tanesinin ağ geçidi olmasına izin verilir. Diğer taraftan bu işlem ağ organizasyonuna bir miktar ek mesajlaşma maliyeti getirmektedir. Fakat karşılığında ekstra hiçbir maliyet gerektirmeden ağın tümünde ana istasyona doğru döngü içermeyen bir yönlendirme hattı oluşturulmuş olur. Bu ekstra mesajlaşma maliyeti Bölüm 2.3'de benzetim yolu ile incelenmiştir.

**Çizelge 2-2:** *Gateway* karar algoritması (GKA-1).

**Assumptions:** Network is $G = (V;E)$, cluster members of node $i$ are $C_i \subset V$, there is no node failure during the configuration

```
 1: for all gateway candidates v∈ V do
 2: BEGIN:
 3:     BROADCAST gateway_decision_call and START gateway_decision_timer
 4:     while neighbor gateway candidates u∈ C_v of v and gateway_decision_timer is running do
 5:         RECEIVE gateway_decision_call or reply from u and update neighbor list
 6:         GET floating neighbor count f_u of u
 7:         GET average RSSI level of floating neighbors p_u of u
 8:         if f_u > f_v or (f_u = f_v and p_u > p_v) then
 9:             state ← node                    /*There is another candidate having better attributes*/
10:         else                                /*This node has better attributes than the neighbor candidate*/
11:             SEND gateway_decision_reply message to u
12:         end if
13:     end while
14:     if state does not equal to node then
15:         state ← gateway
16:         START new clustering by assigning new initiator
17:     end if
18: END:
19: end for
```

### 2.2.1.2 Alternatif *gateway* karar algoritması (GKA-2)

Çizelge 2-2'de verilen GKA-1 *gateway* karar algoritması belirli bir alan içinde yeni küme başlatmak için en uygun ağ geçidi adayının yerel ve etkileşimli olarak belirlenmesini sağlamaktadır ve haberleşme menzilindeki birimlerin aynı anda küme oluşturmasını engellemektedir. Fakat bu algoritmada ağ geçidi adayı olan her birimin diğer ağ geçidi adaylarının komşuluk bilgilerini alabilmek için fazladan bir adet bütüneyayım yapması gerekmektedir. Yeni kümeleri başlatacak sınır düğümlerin seçimindeki bu maliyeti azaltmak için tez çalışmasında ana algoritmaya alternatif olarak GKA-2 adlı zamanlayıcı tabanlı ikinci bir algoritma

geliştirilmiştir. Çizelge 2-3'de detayı verilen algoritmanın ilk aşamasında, komşularını belirleyen ağ geçidi adaylarından *floating* komşu sayısı (*f*) beklenen toplam komşu sayısından (*d*) fazla olanlar doğrudan başlatıcı seçim işlemlerine başlarlar (Çizelge 2-3, Satır 3-7). Eğer $1 \leq f \leq d$ ise bu düğümler bir üstel rastsal sayı *r* üretir ve bu sayıya bağlı olarak *gateway_decision_timer* zamanlayıcılarını çalıştırır (Çizelge 2-3, Satır 9-10). Bu algoritmada bir kümeye ait birbirine yakın ağ geçidi birimlerinin yakın zamanlarda başlatıcı seçme işlemlerine başlamaması gerekmektedir. Aksi takdirde GKA-1 algoritmasında bahsedildiği gibi aynı bölgede birden fazla kümenin oluşturulmaya çalışılması nedeni ile TK protokolünün kümeleme performansında düşüş yaşanacaktır. Burada rastsal belirlenen zamanlayıcı sınırı her bir ağ geçidi adayının farklı zamanlarda başlatıcı seçimi işlemine başlamasını sağlamak içindir. Bu zaman içerisinde *gateway_decision* mesajını alan ağ geçidi adayları durumlarını *node* olarak değiştirir ve başka bir işlem yapmaz (Çizelge 2-3, Satır 11). Eğer herhangi bir ağ geçidi adayı düğüm zamanlayıcısı sıfırlanana kadar kendi kümesine ait bir başka aday düğümden böyle bir mesaj almamış ise durumunu *gateway* olarak değiştirir ve yeni küme oluşturma işlemlerini başlatır (Çizelge 2-3, Satır 14-15).

**Çizelge 2-3:** Alternatif *gateway* karar algoritması (GKA-2).

**Assumptions:** Network is $G = (V;E)$, there is no node failure during the configuration

```
 1: for all gateway candidates v∈ V do
 2: BEGIN:
 3:     if floating neighbor count f_u > expected node degree d
 4:         state ← gateway
 5:         BROADCAST gateway_decision to neighbors
 6:         START new clustering by assigning new initiator
 7:         goto END
 8:     else
 9:         r ← exponential random number where λ = t_c
10:         START gateway_decision_timer set to r
11:         if gateway_decision message is received before gateway_decision_timer outs then
12:             state ← node
13:         else
14:             state ← gateway
15:             START new clustering by assigning new initiator
16:         end if
17:     end if
18: END:
19: end for
```

*Gateway* aday zamanlayıcılarının farklı zamanlarda sıfırlanması için üretilecek rastsal sayılar Krishnan ve Starobinski (2006) tarafından önerilen yöntemle belirlenir. Bu yöntem aslında tüm ağı kümelere ayırmak için küme başlatıcılarını zamanda ve uzayda ayrık şekilde belirlemek için geliştirilmiştir.

Algoritma ağdaki herhangi bir küme genişleme alanında yalnızca bir başlatıcının aktif olmasını sağlamaya çalışmaktadır. Bunu için ağdaki her düğüm bir üstel rastsal sayı belirleyerek bu değer sınır olmak üzere bir zamanlayıcı çalıştırır. Zamanlayıcı sıfırlandığında ise kendini başlatıcı ilan ederek küme oluşumunu başlatır. Krishnan ve Starobinski (2006), zamanlayıcıyı ayarlayan $\lambda$ parametresini toplam kümeleme zamanı $\tau_c$ ve küme eleman sayı sınırı $\beta$ olmak üzere aşağıdaki gibi tanımlamıştır:

$$\tau_c \leq \frac{1}{\lambda.\beta} \tag{2.1}$$

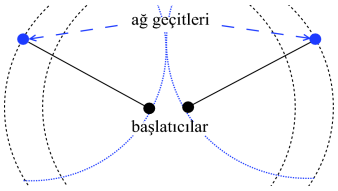
Aynı eşitsizlik TK protokolünde başlatıcıları seçecek ağ geçidi birimlerinin ayrık olarak oluşturulabilmesi için de kullanılabilir. Bu sayede GKA-2 algoritması ile yapılandırılan ağda GKA-1 algoritmasında ihtiyaç duyulan bütüneyayım miktarının ciddi oranda azaltılması beklenmektedir. Diğer taraftan birbirlerinin komşuluğundaki ağ geçidi adaylarının başlatıcı seçmelerini engelleyen kontrol mekanizmasının ortadan kalkması nedeni ile 1 elemanlı veya az sayıda elemana sahip küme oluşumunda bir artış oluşması da muhtemeldir. Ayrıca ağ geçidi seçiminde yeterli ayrıklık sağlanmış olsa bile aynı bölgede zamanlayıcısı dolan diğer ağ geçidi adayları daha önce *floating* olarak kaydettikleri komşu düğümler ile yeni küme oluşturmaya çalışacaktır. Fakat bu düğümler zaten yapılandırılmış olduğundan bunlarla yeni küme başlatma isteği başarılı olmayacaktır. Listesindeki tüm *floating* birimler ile küme oluşturmaya çalışan zamanlayıcısı sıfırlanmış ağ geçidi düğümleri de ağda yapılan toplam bireyayım miktarına artış getirecektir. Diğer taraftan ağ geçidi adaylarının çalıştırdığı zamanlayıcının sınır değerine bağlı olarak toplam ağ yapılanma zamanları da değişecektir. Bu bağlamda GKA-2 algoritması ile yapılanan TK protokolünün performans sonuçları avantaj ve dezavantajları ile benzetim çalışmaları bölümünde incelenmiştir.

### 2.2.2 Başlatıcı seçimi

TK protokolünün daha iyi ve hızlı kümeleme yapabilmesi için uzaysal olarak birbirinden ayrık başlatıcıların farklı alanlarda paralel küme oluşturmaları gerekmektedir. Ağ geçidi seçimi ile bu ayrıştırma ilk aşamada sağlanmaktadır

fakat bu başlatıcıların da ayrık olacakları anlamına gelmemektedir. Şekil 2-2'de gösterildiği gibi ilgili ağ geçidi düğümleri ayrık olsa bile en az iki başlatıcı birbirlerinin haberleşme menzilleri içerisinde olabilirler. Böyle durumlarda yalnızca bir başlatıcının genişlemesine izin verilmeli ve diğerlerinin genişlemesi durdurulmalıdır. Genişlemeyen başlatıcılar durumlarını *node* olarak değiştirir ve tek elemanlı küme olarak yapılanmadaki yerlerini alırlar. (Çizelge 2-1, Satır 24-25).

Burada hangi başlatıcının genişleyeceğine karar verebilmek için tüm başlatıcılar ağ geçidi belirleme ile aynı algoritmayı çalıştırabilirler. Fakat protokolde işlem karmaşıklığını azaltmak için daha basit bir yöntem kullanılmıştır. Buna göre birbirlerinin komşuluğunda olup henüz küme oluşumunu başlatmamış düğümler *ID* numaralarının büyüklüğüne göre küme genişlemesine devam edip etmeyeceklerinin kararını verirler. Bunun için komşu keşfi tamamlandıktan sonra komşuluğunda en az bir adet başlatıcı bulunan düğüm bu başlatıcılar ile *ID* numarasını karşılaştırır. Eğer kendi *ID* numarası daha büyük ise küme oluşturmaya devam eder ve aksi takdirde bir işlem yapmaz. Benzetim sonuçlarında bu kararın küçük küme oluşumunu azalttığı ve ortalama küme boyutunu artırdığı ayrıca bağlantı oranını yükselttiği gözlemlenmiştir.




**Şekil 2-2:** Muhtemel başlatıcı yakınlığı örneği.

Diğer taraftan paralel küme oluşumu için verilen kararlar neticesinde ağda herhangi bir kümeye bağlanmamış düğümler kalabilir. Bu düğümlerin bağlı olan düğümlere oranı ile ağın genel bağlantı oranı ölçülebilir. Örneğin 400 duyarga düğümü olan bir ağda düğüm derecesi 8 iken %98'lik bir ağ bağlantı oranı ile ayrık kalan düğümlerin sayısı sadece 8 olmaktadır. Gerçekte bu düğümlerin menzilde komşuları bulunabilir fakat genişleme için seçilmediklerinden ayrık kalmış olabilirler. Bu durumu çözmek için yoklama mesajını alan her *floating* durumundaki düğüm bir zamanlayıcı çalıştırır. Belirli bir zaman içerisinde bu

düğüm yapılandırılmamış ise kendisine yoklama yapan düğümlerden en yakın olana bağlanma isteğinde bulunur ve varsa kendi etrafındaki *floating* düğümler ile küme oluşturur. Bu zamanlayıcının değeri ağın yapılanma zamanından küçük olmalıdır. Bu algoritmada aynı anda birden fazla zamanlayıcı dolmasını engellemek için düğümler rastsal zamanlayıcı değerleri belirleyebilir. Tez çalışmasında bu tür ayrık birimler tek elemanlı küme olarak sayılmış ve kümeleme performansı bu şekilde değerlendirilmiştir.

Zamanlayıcı tabanlı ağ konfigürasyonu yapan Krishnan ve Starobinski'nin (2006) çalışmasından farklı olarak bu çalışmada geliştirilen tekrarlı kümeleme ile ağda birden fazla ana istasyon bulunması durumunda toplam ağ yapılanma süresi daha da iyileştirilebilir. Birbirlerinden uzaysal olarak ayrık olan ana istasyonlar ağda yapılanmayı aynı anda başlatabilir ve tek ana istasyonlu yapılanmaya göre daha kısa zamanda ağ yapılanması tamamlanabilir.

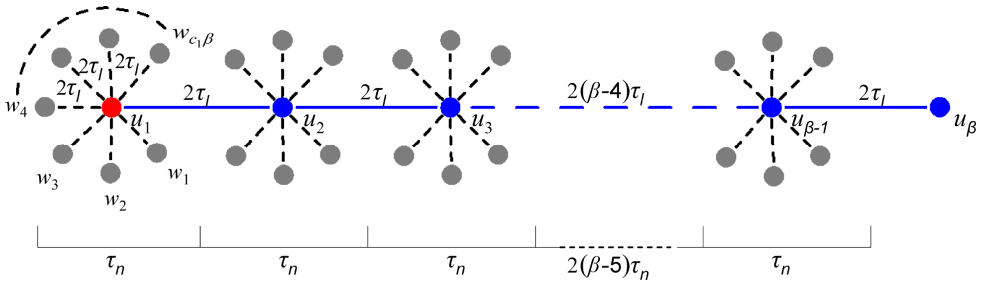
### 2.2.3 Zaman karmaşıklığı analizi

**Teorem I:** Değiştirilmiş *persistent* algoritmasının tek küme oluşumu için en kötü zaman karmaşıklığı toplam küme bütçesi $\beta$ olmak üzere $O(\beta^2)$'dir.

**İspat:** Başlatıcıdan itibaren küme yapılanırken her $u_i$ için genişleyecek sadece bir adet *floating* komşu $u_{i+1}$ varsa, son komşuya kadar toplamda $\beta$-1 hat oluşturulur. Verilen bütçe $\beta$ için, $\beta$-1 kümenin en uzun derinliğine eşittir. Bu derinlik kümenin paralellik olmadan sıralı yapılanma ile oluştuğu anlamına gelir. Krishnan ve Starobinski (2006) orijinal *persistent* algoritmasının mesajlaşma karmaşıklığını $O(\beta^2)$ olarak hesaplamıştır. Burada düğüm zaman $\tau_l$ hat gecikmesine eşittir. TK protokolünde ise $\tau_l$ hat gecikmesinin yanında $\tau_n$ komşu keşif zamanı da bulunmaktadır. Bu protokole göre $u_i$ (i=1,2,3,… $\beta$ -1), en kötü durumda, her $w_1 \ldots w_{c\beta}$ komşusuna en fazla $c_i\beta$ kez bütçe gönderir. Bu bütçe en kötü durumda mesaj yollanan son komşusu tarafından genişlemek için kullanılır. Duyarga düğümlerinde harcanan zaman düşünülmediğinde toplam kümeleme zamanı $\tau_c$ aşağıdaki gibidir:

$$\tau_c = \sum_{i=1}^{\beta-1}(2c_i\beta\tau_l + \tau_n) = c\tau_l\beta(\beta-1) + \tau_n(\beta-1) = (\beta-1)(c\tau_l\beta + \tau_n) \qquad (2.2)$$

Eğer $\tau_n = c_1\beta$ ise, $\tau_c = (\beta-1)(c\tau_l\beta + c_1\beta) = (c\tau_l + c_1)(\beta^2 - \beta)$ olur ve sonuçta en kötü zaman karmaşıklığı $O(\beta^2)$'dir. Bu en kötü küme yapısı Şekil 2-3'de gösterilmiştir. □



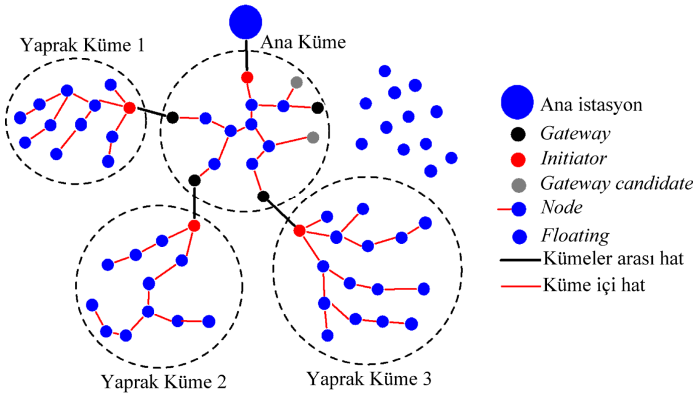

**Şekil 2-3:** En kötü küme içi yapılanması.

**Teorem II:** En kötü ağ yapılanması ağdaki herhangi bir kümenin en kötü şekilde yapılanması ile oluşur. Sonuçta oluşan kümeler arası bağlantı ağacı ya sıralıdır ya da rastsal *m-ary* şeklindedir.

**İspat:** Teorem I, kümelerin en kötü yapılanma zamanının küme içi genişleme sırasında her bir düğümün sadece bir adet genişleyecek komşusu bulunduğu zaman oluştuğunu belirtmişti. Kümeler düğüm ve kümeler arası bağlantılar kenar olmak üzere tüm ağı kümeler arası bağlantı ağacı olarak düşündüğümüzde (Şekil 2-4), aynı teorem ağın en kötü yapılanma zamanını hesaplamak için de kullanılabilir. Fakat kümeler arası yapılanma zamanı hesabında küme içi yapılanmaya göre iki farklılık bulunmaktadır.

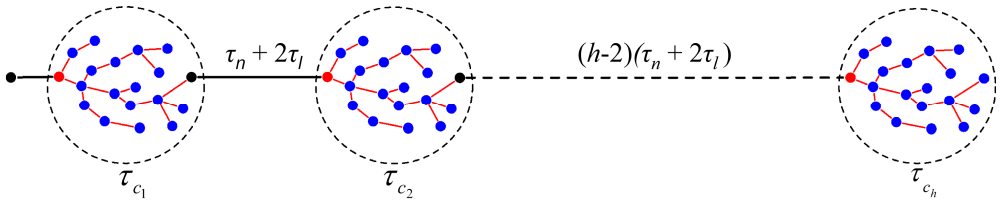
I. Küme içi genişlemede her bir düğüm $\beta$ bütçesini uygun komşularına dağıtmak zorundadır fakat kümeler arası bağlantıda komşu keşfinden sonra başlatıcının oluşturulabilmesi için sadece bir tane hat oluşturulur. Bu durumda hat oluşum maliyeti $\tau_n + \tau_l$ ile hesaplanabilir.

II. Teorem I'de her düğüm tarafından harcanan zamanın önemsiz olduğu belirtilmişti. Fakat kümeler arası yapılanmada bu süre $\tau_c$ küme içi yapılanma zamanına eşittir ($\tau_c >> \tau_n + \tau_l$.).




**Şekil 2-4:** Kümeler arası yapılanma ağacı örneği.

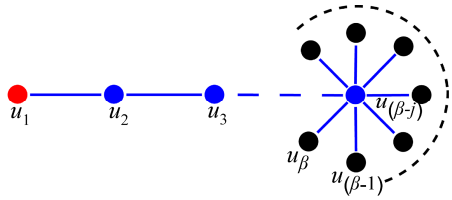
Sonuçta oluşan ağaç, zaman farkı dışında yapı olarak Teorem I'de oluşturulan ağacın aynısıdır (Şekil 2-5). Bu şekilde ağın yapılanması tamamen sıralıdır. Bu yapıda genel anlamı bozmadan kümeler arası hat oluşma zamanı $\tau_c + \tau_n + \tau_l = O(\tau_c)$ olarak belirtilebilir.



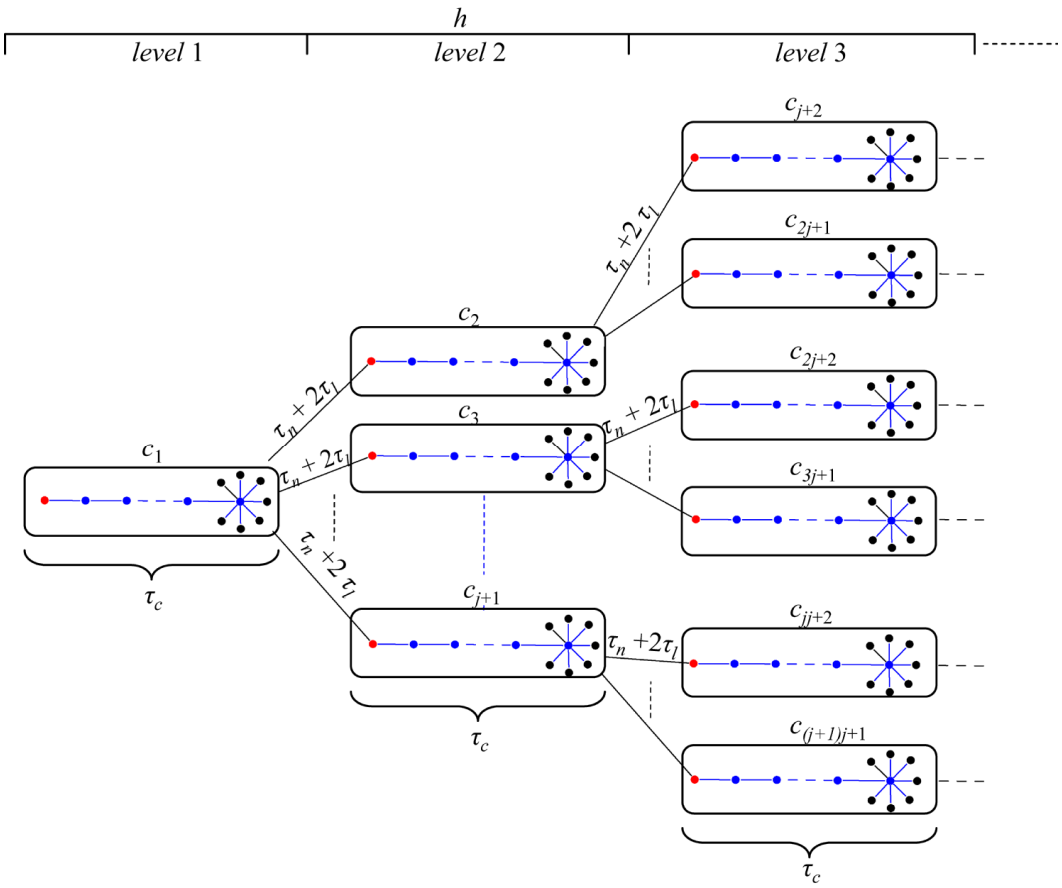

**Şekil 2-5:** En kötü kümeler arası yapılanma.

Ağda yeteri kadar düğüm olduğu varsayıldığında, her küme $c$ için $p$ olasılığında $j$ ($j \geq 0$) adet ağ geçidi bulunduğunu düşünelim. Bu durumda ağ yapılanma zamanı en kötü küme içi yapılanma zamanı $\tau_c$ ve kümeler arası ağaçta aynı seviyede başlatılan kümelerin sayısına bağlıdır. Burada ikinci madde paralel yapılanacak kümelerin sayısını belirler. Ne kadar paralel kümeleme yapılırsa toplam yapılanma zamanı o kadar az olur. Eğer her küme en kötü şekilde yapılanmış $\beta$-$j$ adet üyeden oluşup $j$ adet ağ geçidi düğümü Şekil 2-6'da gösterildiği gibi kümenin en son seviyesinde seçilirse bu durumda kümeler arası bağlantı ağacındaki tüm hatlar için en kötü yapılanma zamanı oluşur. Sonuçtaki

ağaç rast gele oluşan bir *j*-ary ağaçtır (Şekil 2-7). Bu ağacın her seviyesi en kötü $\tau_c + \tau_n + \tau_l$ zamanında yapılanır. Burada toplam ağ yapılanma zamanı $h(\tau_c + \tau_n + \tau_l)$ olarak hesaplanabilir. □




**Şekil 2-6:** En kötü *gateway* seçimi için tek küme yapılanması.




**Şekil 2-7:** Muhtemel paralellik ile kümeler arası yapılanma.

**Teorem III:** Kümeler arası yapılanmanın beklenen zamanı, ağdaki küme sayısı *n* olmak üzere $O(\lg n)$'dir.

**İspat:** Varsayalım ki Şekil 2-6'da gösterildiği gibi yapılanmış her küme için rast gele şekilde en fazla $j = 2$ adet ağ geçidi oluşturulabilsin. Bu varsayım ağ

yapılanmasındaki en düşük paralelleşmeyi verir. Sonuçta da çocukları rastsal şekilde oluşturulan bir ikili ağaç elde edilir. Bu durumda beklenen yapılanma zamanı problemi rastsal düğüm eklenerek oluşan ikili bağlantı ağacının beklenen yüksekliğinin belirlenmesi ile hesaplanabilir.

Literatürde ikili ağaçların beklenen yüksekliğini hesaplayan pek çok çalışma bulunmaktadır. Bu çalışmaların içinden rastsal oluşturulan ikili arama ağaçlarının beklenen yüksekliği analizi buradaki ispat için bir referans olabilir. Bunun için ilk olarak kümeler arası ağaç oluşturma ile rastsal ikili arama ağacı oluşturma işlemleri arasında ilişki kurulması gerekmektedir. Tanım olarak bir rastsal ikili arama ağacı, $i$ adet farklı anahtarın, $i!$ permütasyonun her birinin oluşma olasılığı eşit olmak üzere, rast gele bir sırada ağaca eklenmesi ile oluşturulur (Cormen et al., 2001). Eğer bu $i$ adet farklı anahtar, $n$ toplam ağ eleman sayısı ve $\beta$ küme eleman sayısı sınırı olmak üzere, $i = n/\beta$ adet küme numarası ile eşleştirilirse, yapılanma sonrası oluşacak herhangi bir kümeler arası bağlantı ağacından çekilen küme listesi, $i!$ anahtar permütasyon sırasından birine denk gelecektir. Bu durumda Cormen ve ark. (2001) tarafından verilen ispat, oluşan kümeler arası bağlantı ağacının beklenen yükseklik değerini hesaplamak için de kullanılabilir. Bu beklenen değer $O(\lg n/\beta)$'dır. Bu durumda kümeler arası bağlantı ağacının oluşumu (diğer deyişle ağın toplam yapılanma zamanı) için geçen zaman $O((\tau_c + \tau_n + \tau_l)\lg n/\beta) = O(\lg n)$ olarak hesaplanır (İspat için bkz. EK-B). □

## 2.3 Benzetim Çalışması

Kümeleme algoritması Omnet++ üzerinde geliştirilmiş olay tabanlı duyarga ağı benzetim aracı üzerinde uygulanmıştır. Duyarga düğümlerinin ağdaki yerleşim dağılımı 2D *uniform*'dur. Benzetim aracında kablosuz ortam kullanım seviyesinde IEEE 802.11b standart protokolü kullanılmaktadır. Yakınlık hesabı radyo gücünün iki düğüm arasındaki mesafenin karesine bölünmesi ile yapılmıştır. Sonuçlar farklı yerleşimlere sahip 10 değişik ağ üzerinde yapılan benzetimlerin ortalaması alınarak oluşturulmuştur. Protokol performansı benzetimler sonucu elde edilen kümeleme performansı, yapılanma zamanı ve bunların ağdaki duyarga düğüm derecesi ile ilişkisi ortaya konularak değerlendirilmiştir.

### 2.3.1 Benzetimler için protokol zamanlayıcı parametreleri

Benzetimlerde seçilecek zamanlayıcı parametreleri tez kapsamında geliştirilen TK protokolleri TK (GKA-1), TK (GKA-2) ve karşılaştırma yapılan ZTK protokolünün toplam yapılanma zamanlarını ve yapılanma performanslarını doğrudan etkilemektedir. Dolayısıyla bu bölümün alt başlıklarında tez kapsamında yapılan benzetimler için seçilen zamanlayıcı parametreleri hakkında bilgi verilmiştir.

#### 2.3.1.1 TK (GKA-1) protokolü için zamanlayıcı parametreleri

TK (GKA-1) protokolü için benzetimde kullanılan zamanlayıcı parametrelerinden özellikle toplam yapılanma zamanını etkileyen en önemli faktör komşu keşfi sırasında harcanan zamandır ($\tau_n$). Bu zaman ne kadar fazla olursa tek küme oluşumu için ihtiyaç duyulan $\tau_c$ zamanı da o kadar fazla olmaktadır. Artan $\tau_c$ zamanı ile de toplam yapılanma zamanı artmaktadır. Bu artışın komşu keşfine bağlı olarak en düşük düzeyde tutulabilmesi için Çizelge 2-1'de bir düğümün komşuluğundaki diğer düğümleri keşfedebileceği süre zamanlayıcısı olarak belirtilen *neighbor_discovery_timer*'ın belirlenmesi gerekmektedir. Özellikle yüksek yoğunluklu ağ yapılanmaları için alınan ön benzetim sonuçlarına göre bir düğümün komşularını belirlemek için yaklaşık 100 – 200 milisaniye zamana ihtiyacı olduğu belirlenmiştir. Dolayısıyla benzetimler için bu süreye güvenli bir fark konularak $\tau_n$ komşu belirleme zamanı 500 milisaniye olarak sabitlenmiştir.

TK (GKA-1) protokolünde toplam yapılanma zamanını etkileyen diğer parametre ise "*gateway_decision_timer*" değeridir. Bu zamanlayıcı, oluşturulacak yeni kümelerin başlatıcılarını belirleyecek ağ geçidi düğümlerinin belirlenmesi sırasında tüm ağ geçidi adayı düğümleri tarafından çalıştırılır. Ağ geçidi adayı düğümleri bu operasyondan önce komşu keşfi de yaptığından bu kümede ağ geçidi birimleri üzerinden diğer küme oluşumunun başlatılması için en az komşu keşfi ve *gateway* karar zamanı kadar beklenmesi gerekmektedir. *Gateway* kararının verildiği algoritmada haberleşme menzilindeki tüm ağ geçidi adayı düğümlerin *floating* komşu sayısını belirlemiş olması önem taşıdığından

benzetimlerde *gateway* karar zamanlayıcısı *gateway_decision_timer* komşu keşfi zamanlayıcısı *neighbor_discovery_timer*'dan büyük seçilmiştir ve 1000 milisaniye ($2\tau_n$) olarak sabitlenmiştir.

Benzetim ortamında ölçülen hat gecikme zamanı $\tau_l$ ise yaklaşık 1 milisaniyedir ve TK protokolünde küme oluşum zamanına bir etkisi yoktur.

#### 2.3.1.2 ZTK protokolü için zamanlayıcı parametreleri

Krishnan ve Starobinski (2006), zamanlayıcıların birim zamanını düğümler arası iletim gecikmesi olarak almış ve $\tau_c$ değerini en kötü kümeleme zamanı olan $\beta^2$ olarak belirlemiştir. Bu çalışmada komşu keşfi aşaması varsayılmadığından düğümler arası iletimde komşu keşfi ile ilgili bir zaman parametresi belirtilmemiştir. Dolayısıyla kümeleme zamanı sadece hat gecikmesi cinsinden belirlenmiştir. Bu durumda zaman açısından iyimser yapılanma için Eşitlik 2.1'de verilen formüle göre $\lambda$ değerinin $1/\beta^3$ değerinden küçük veya bu değere eşit seçilmesinin bu yöntemde en iyi kümeleme ve zaman performansını verdiği belirtilmiştir.

Tez çalışmasında yapılan benzetimlerde ise ağ yapılanması sırasında *persistent* kümeleme protokolünü çalıştırabilmek için düğümlerin komşularını da belirlemesi gerekmektedir. Bu durumda en kötü küme oluşumunda toplam kümeleme zamanı hesaplanırken komşu keşif zamanının da dikkate alınması gerekmektedir. Komşu keşif zamanı $\tau_n$ ve hat gecikme zamanı $\tau_l$ parametrelerine bağlı olarak *persistent* kümeleme algoritmasının en kötü zamanı Eşitlik 2.2'de verilmişti. Eğer yapılanma gereksinimlerine göre bu eşitlikte $\tau_n >> \tau_l$ olur ise toplam kümeleme zaman maliyeti $\tau_c$, komşu keşif zamanı $\tau_n$ biriminden belirlenmelidir. Benzetim altyapısında düğümler arası hat gecikmesi $\tau_l \approx 1$ milisaniyedir ve komşu belirleme zamanlayıcı sınırı ise $\tau_n = 500$ milisaniye olarak belirlenmiştir. Dolayısıyla küme oluşumu için birim zaman 500 milisaniye olarak referans alındığında en kötü kümeleme zamanı $\tau_c = 500\beta$ milisaniye olarak hesaplanır. Sonuç olarak komşu keşfi de dikkate alındığında $\lambda \leq 1/500\beta^2$ olmalıdır.

Diğer taraftan, yapılan ön benzetim kontrollerinde teorik olarak belirlenen $\lambda$ değerinin ($\lambda$ = 1/500.32.32 = 1/512.000) ağ yapılanma zamanını uzun tuttuğu belirlenmiştir. Yapılan denemelerde ise $\lambda$ değerinin ortalama küme oluşum zamanı ile ortalama küme büyüklükleri kullanılarak belirlendiğinde yapılanma zamanı ve kümeleme performansı açısından en uygun sonucu verdiği belirlenmiştir. Buna göre küme eleman sayısı 32 ile sınırlandırılarak test edilen {7, 21, 35} ağ yoğunluklarında elde edilen ortalama küme büyüklüğü yaklaşık 20'dir. Bu kümelerin sahip olduğu ortalama derinlik ise yaklaşık 2'dir. Ortalama derinlik değeri aynı zamanda kümelerin ortalama oluşturulma zamanını da vermektedir. Çünkü küme oluşturulurken en fazla bu derinlik sayısı kadar sıralı komşu belirleme işlemi yapılmaktadır. Aynı kümedeki diğer komşu belirleme işlemleri ise paralel gerçekleştirilir. Dolayısıyla benzetim sonucu elde edilen ortalama küme oluşum zamanı 2x500 = 1000 milisaniye olarak hesaplanır. Bu durumda 1/$\lambda$ = 20x1000 olarak seçildiğinde ZTK benzetimleri sonucunda en uygun yapılanma performansının elde edilebilmesi beklenmektedir.

Çizelge 2-4'de 400 düğümlü bir ağda {7, 21, 35} ağ yoğunluklarında küme eleman sayısı 32 ile sınırlandırılmak üzere 1/$\lambda$ = 10.000, 20.000 ve 30.000 değerleri için ZTK ile elde edilen toplam yapılanma zamanı ve oluşturulan küme sayılarının karşılaştırmaları verilmiştir. Bu sonuçlara göre 1/$\lambda$ = 20.000 olduğunda toplam zamanının 1/$\lambda$ = 30.000 için olandan %40 daha az ve 1/$\lambda$ = 10.000 için olandan %9 daha fazla olduğu görülmektedir. Diğer taraftan, toplam küme adetleri karşılaştırıldığında 1/$\lambda$ = 20.000 iken oluşan küme sayısı 1/$\lambda$ = 10.000 için olandan yaklaşık %7 daha azdır. Burada düşük küme sayısı yapılanma değerlendirmesi açısından olumlu bir performans göstergesidir. 1/$\lambda$ = 30.000 olduğu durumda ise elde edilen küme sayısı 1/$\lambda$ = 20.000'deki küme sayısına neredeyse eşit durumdadır. $\lambda$ değerinin teorik değere eşitlenmesi durumunda ise 7 ve 21 gibi ağ yoğunluklarında oluşturulan toplam küme sayısı sadece %1 civarında iyileşirken toplam yapılanma zamanı yaklaşık 10 kat artmaktadır. Bu sonuçlarla kümeleme performansı öncelikli olarak değerlendirildiğinde, 1/$\lambda$ = 10.000 ile arasında ciddi bir yapılanma zaman farkı olmaması nedeni ile kümeleme performansı daha iyi olan 1/$\lambda$ = 20.000 tercih edilebilir durumdadır. Dolayısıyla devam eden bölümdeki benzetimlerde ZTK protokolü bu değer üzerinden çalıştırılmıştır.

**Çizelge 2-4:** Farklı $\lambda$ değerleri için ZTK protokolünde a) Yapılanma zamanı b) Toplam küme sayısı değişimi.

(a)

| *Ortalama Zaman (sn)* | | *Ağ yoğunluğu* | | |
|---|---|---|---|---|
| | | 7 | 21 | 35 |
| $1/\lambda$ | 10.000 | 98,11 | 74,58 | 71,37 |
| | 20.000 | 103,77 | 93,43 | 77,46 |
| | 30.000 | 146,84 | 95,57 | 110,16 |
| | 512.000 | 1427,33 | 986,29 | 877,32 |

(b)

| *Ortalama Küme Sayısı* | | *Ağ yoğunluğu* | | |
|---|---|---|---|---|
| | | 7 | 21 | 35 |
| $1/\lambda$ | 10.000 | 34,1 | 19,2 | 34,1 |
| | 20.000 | 32,7 | 19,1 | 32,7 |
| | 30.000 | 32,4 | 19,1 | 32,4 |
| | 512.000 | 32 | 18,8 | 18,6 |

### 2.3.1.3 TK (GKA-2) protokolü için zamanlayıcı parametreleri

Yapılan ön benzetim denemelerinde GKA-2 algoritmasında *gateway* karar aşamasında kullanılan üstel rastsal zamanlayıcılar için seçilecek $\lambda$ değerinin ZTK protokolü için teorik olarak hesaplanan eşitsizlik üzerinden $\lambda \leq 1/500\beta^2$ olarak belirlenmesi durumunda toplam yapılanma zamanının TK (GKA-1) sonuçları ile karşılaştırılamayacak derecede yüksek olduğu belirlenmiştir. Bunun yerine Bölüm 2.3.1.2'de belirlendiği gibi ortalama küme büyüklüğü ve yapılanma zamanı dikkate alınarak $1/\lambda$ = 20.000 olarak seçildiğinde ise kümeleme performansı değişmeden toplam yapılanma zamanının ciddi olarak iyileştiği fakat yine de sonucun TK (GKA-1) protokolünden daha fazla olduğu görülmüştür. Bu değer ağdaki tüm düğümlerin zamanda ve düzlemde ayrık olarak oluşturulabilmesi için gereklidir fakat GKA-2 algoritmasında sadece ağ geçidi adayı düğümler zamanlayıcı çalıştırmaktadır ve bu düğümlerden birinin zamanlayıcısı sıfırlandığında komşuluğundaki diğer ağ geçidi adayı düğümlerin zamanlayıcılarının ortalama küme oluşum zamanı kadar beklemeleri o bölgede küme oluşumunun düzenli oluşturulması için yeterlidir. Bu durumda $\lambda$ değerinin $t_c$ kümeleme zamanından büyük seçilmesinin de performans açısından yeterli sonuç vermesi beklenebilir.

Çizelge 2-5'de 400 düğümlü bir ağda {7, 21, 35} ağ yoğunluklarında küme eleman sayısı 32 ile sınırlandırılmak üzere $1/\lambda$ = 5000, 10.000, 20.000 değerleri için TK (GKA-2) protokolü ile elde edilen toplam yapılanma zamanı ve oluşturulan küme sayılarının karşılaştırmaları verilmiştir (Aynı ağ parametrelerinde $t_c$ kümeleme zamanının 1000 milisaniye olduğu Bölüm 2.3.1.2'de belirtilmişti; $1/\lambda$ değerleri buna göre seçilmiştir). Bu sonuçlara göre 5000 için elde edilen performans, küme adedi ve yapılanma zamanı açısından diğer seçeneklere göre öne çıkmaktadır.

**Çizelge 2-5:** Farklı $\lambda$ değerleri için TK (GKA-2) protokolünde a) Yapılanma zamanı b) Toplam küme sayısı değişimi.

(a)

| *Ortalama Zaman (sn)* | | *Ağ yoğunluğu* | | |
|---|---|---|---|---|
| | | 7 | 21 | 35 |
| $1/\lambda$ | 5000 | 28,58 | 31,55 | 47,57 |
| | 10.000 | 74,26 | 64,37 | 80,7 |
| | 20.000 | 102,57 | 140,3 | 119,3 |

(b)

| *Ortalama Küme Sayısı* | | *Ağ yoğunluğu* | | |
|---|---|---|---|---|
| | | 7 | 21 | 35 |
| $1/\lambda$ | 5000 | 34 | 24 | 38 |
| | 10.000 | 28 | 21 | 28 |
| | 20.000 | 28 | 19 | 26 |

Çizelge 2-5-(a)'da verilen sonuçlara göre $1/\lambda$ = 10.000 için elde edilen yapılanma zamanı 5000 ile yapılanmaya göre, her üç yoğunluktaki farkların ortalaması alındığında, yaklaşık %100 daha fazladır. Buna karşın Çizelge 2-5-(b)'de verilen sonuçlara göre kümeleme performansı sadece %10-%20 daha iyidir. 20.000 olduğunda ise kümeleme performansı yaklaşık %1-%10 oranında daha iyileşmektedir fakat burada toplam yapılanma zamanı 10.000'e göre yaklaşık %50-%100 oranında yükselmektedir. Dolayısıyla elde edilen bu sonuçlar toplu olarak değerlendirildiğinde yapılanma zamanındaki avantajı nedeni ile 5000 değeri benzetim çalışmasında referans olarak alınmıştır. Diğer taraftan aynı ağ yapısında kümeleme performansı yapılanma zamanından daha öncelikli olarak tercih edilirse 10.000 değeri de seçilebilir durumdadır. Ortalama küme eleman sayısı ve küme oluşturma zamanına göre belirlenen 20.000 değerinin ise nispeten

daha iyi kümeleme performansı olmasına rağmen yapılanma zamanında ciddi artışı bulunmaktadır.

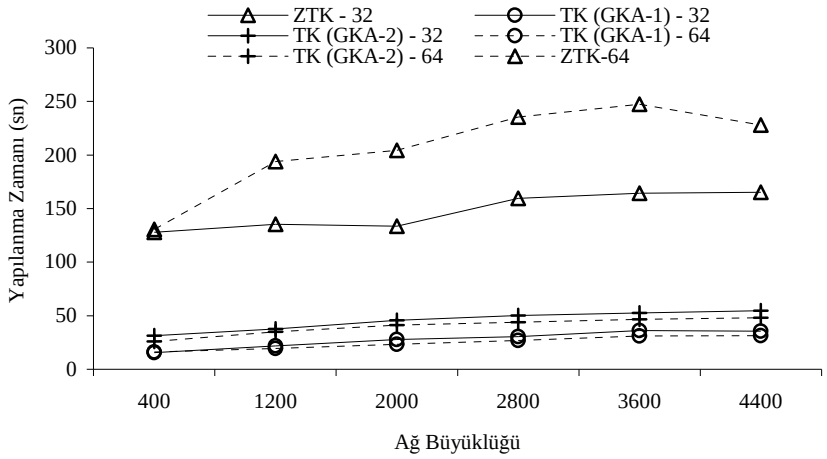
### 2.3.2 Karşılaştırmalı benzetim sonuçları

Bu bölümde ilk olarak Şekil 2-8'den Şekil 2-13'e kadar sabit duyarga düğüm derecesinde, artan ağ eleman sayısına bağlı olarak ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinde oluşan yapılanma performans değişimlerinin benzetim sonuçları verilmiştir. Bu benzetimlerde aksi belirtilmediği sürece bir kümeye ait olabilecek toplam eleman sayısının üst sınırı $\beta$ = 32 olarak alınmıştır. Ağın dağılım parametreleri farklı ağ eleman sayıları için ağdaki beklenen duyarga düğüm derecesi yaklaşık 8 olacak şekilde $d = (n-1)\pi m^2$ eşitliğine göre ayarlanmıştır. Burada *n* toplam ağ eleman sayısı ve *m* düğüm radyo iletişim menzilidir (Schaeffer et al., 2004). Toplam yapılanma zamanı ilk yapılanma mesajının yollanmasından son mesaj alınana kadar geçen süreye eşittir.

Şekil 2-8'de toplam yapılanma zamanının artan ağ eleman sayısı *n*'ye bağlı değişimi verilmiştir. Buradan anlaşılacağı üzere *n* arttıkça yapılanma zamanı her üç protokolde de logaritmik olarak ($O(\lg n)$) artmaktadır. Bu karakteristik Bölüm 2.2.3'de TK için yapılan analiz sonucunu ve Krishnan ve Starobinski (2006) tarafından yapılan ZTK analizlerini doğrulamaktadır. Bu zaman karmaşıklığı ZTK ve TK protokollerini ölçeklenebilir ve geniş ölçekli ağlar için uygulanabilir yapmaktadır. Fakat yapılanma sonuçları karşılaştırıldığında TK protokolleri ZTK protokolüne göre aynı ağ yerleşimi için yaklaşık 4-5 kat daha hızlı tamamlanmaktadır ve daha çabuk yakınsamaktadır. Diğer taraftan TK (GKA-1) ile yapılanma TK (GKA-2) ile yapılanmadan yaklaşık 18 saniyelik bir ofset ile daha az zamanda tamamlanmaktadır. İki protokol arasındaki bu fark GKA-2 algoritmasında ağ geçidi adayı düğümlerin çalıştırdıkları zamanlayıcıların ağ yapılanmasını belirli bir süre geciktirmesinden dolayı oluşmaktadır. Şekil 2-8'de verilen benzetim sonuçları bu farkın ağ eleman sayısı artsa bile sabit kaldığını göstermektedir.

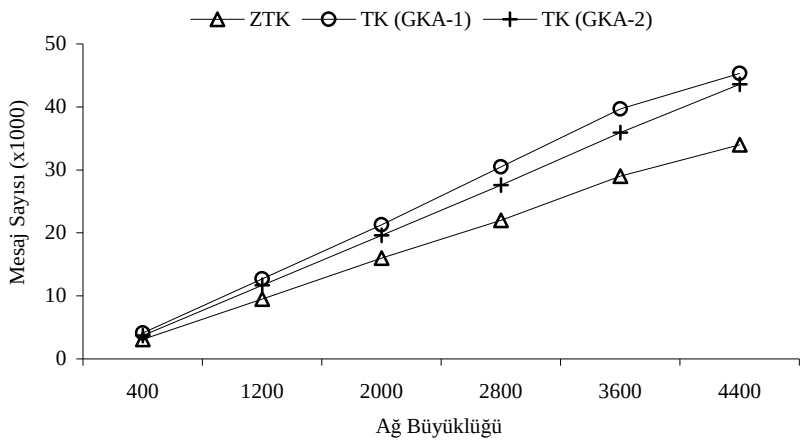
Ek olarak, Şekil 2-8'de verilen benzetim sonuçlarından elde edilen verilere göre küme eleman sayısı $\beta$ = 32'den $\beta$ = 64'e yükseltildiğinde TK (GKA-1) ve

TK (GKA-2) protokollerinde toplam yapılanma zamanının azaldığı belirlenmiştir. Bunun sebebi küme büyüklüğü arttıkça daha az sayıda küme oluşturulması ve dolayısıyla azalan başlatıcı sayısı ile daha az ağ geçidi seçimi işlemi yapılmasıdır ($\beta$ = 64 için TK (GKA-2) protokolünde $1/\lambda$ = 5000 alınmıştır. Çünkü ortalama küme oluşum zamanı 5000 milisaniyeden küçüktür). ZTK protokolünde ise ortalama küme büyüklüğü ve küme oluşma zamanı yükseldiği için zamanlayıcı parametresi $1/\lambda$ değeri yükselmektedir (ortalama küme yüksekliği $h$ = 3,5; $t_c$ = 3,5 x 500 = 1750 milisaniye ve ortalama küme eleman sayısı = 23 => $1/\lambda$ = 40.000). Bu da toplam yapılanma zamanını artırmaktadır.



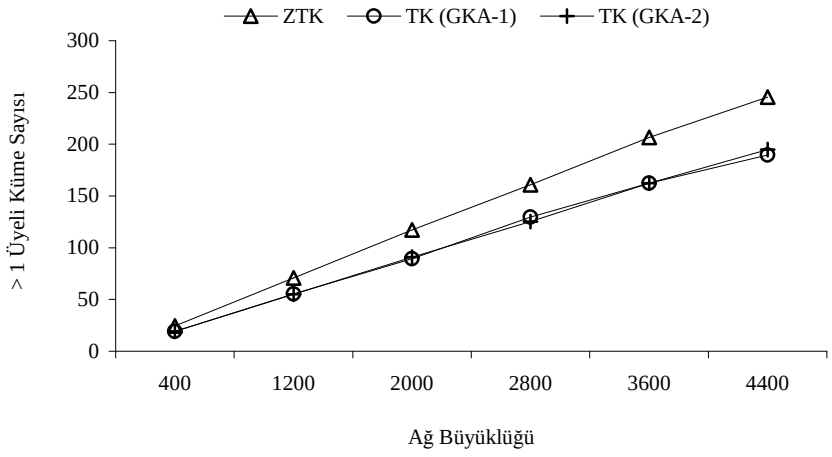

**Şekil 2-8:** Artan ağ eleman sayısına bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinin toplam yapılanma zaman değişimi.

Şekil 2-9'da verilen mesaj karmaşıklıkları incelendiğinde her üç protokolde de ağ eleman sayısı arttıkça ağ yapılanması için ihtiyaç duyulan mesajlaşma miktarı doğrusal şekilde artmaktadır. TK (GKA-1 ve TK (GKA-2) protokolleri yaklaşık aynı maliyette olmakla birlikte daha az bütüneyayım nedeni ile TK (GKA-2) protokolünün mesajlaşma maliyeti yaklaşık %10 daha azdır. Bunun yanında ZTK protokolündeki mesajlaşma maliyeti genel olarak TK protokollerine göre yaklaşık %25 daha azdır. Çünkü TK protokollerinde ağ geçidi seçimi ve yeni başlatıcı belirleme için ekstra mesajlaşmalar yapılır. Bu seçim sonunda ilgili ağ geçidi ve başlatıcı düğümleri arasında kurulan ağ iletim hattı aynı zamanda bu düğümlerin bağlı olduğu kümeler arasındaki bağlantıyı da oluşturur. Fakat ZTK protokolünde kümeler arası böyle bir bağlantı hattı tanımlanmamıştır ve yapılanma tamamlandıktan sonra kümeler ayrık şekilde oluşturulur.



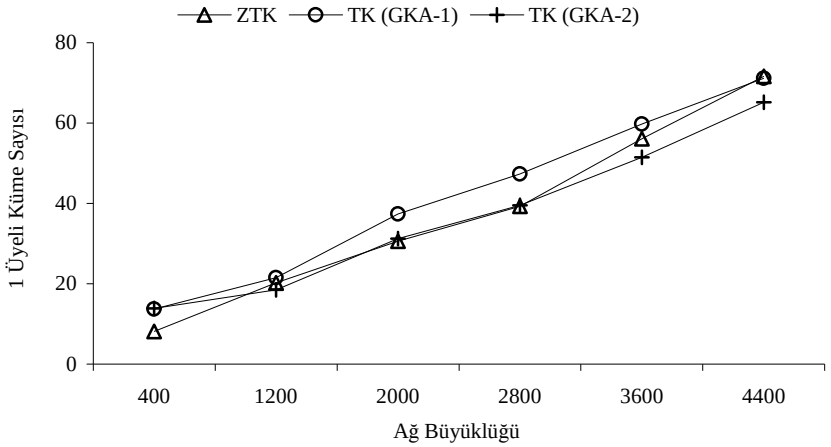

**Şekil 2-9:** Artan ağ eleman sayısına bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinin toplam mesajlaşma maliyetleri.

Artan ağ eleman sayısına göre toplam küme sayısındaki değişimin benzetim sonuçları Şekil 2-10'da verilmiştir. Bu sonuçlara göre her iki TK protokolünde eleman sayısı 1'den büyük kümelerin toplam sayısı birbirinin aynı karakteristiğe sahiptir ve ZTK protokolüne göre yaklaşık %20 daha azdır. Oluşturulan toplam küme sayısının belirlenen küme eleman sayı sınırına bağlı olarak daha az olması bir kümeleme protokolü için istenen bir özelliktir. Çünkü küme sayısının düşük olması protokol sonunda daha denk kümelerin oluşturulduğu anlamına gelmektedir. Fakat kümeleme performansının değerlendirilebilmesi için bu sonuç tek başına yeterli değildir. Tam değerlendirme yapabilmek için kümeleme sonunda ağda tek başına kalan veya tek elemanlı küme oluşturan toplam düğüm sayılarının da belirlenmesi gereklidir.




**Şekil 2-10:** Artan ağ eleman sayısına bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokolleri ile oluşturulan, üye eleman sayısı 1'den büyük toplam küme sayısı.

Bu amaçla kümeleme işlemi tamamlandığında ağda tek başına kalan düğümlerin toplam sayısındaki değişim Şekil 2-11'de verilmiştir. Bu sonuçlara göre her üç protokolde de tek kalan düğümlerin toplam sayıları arasında cüzi bir fark bulunmaktadır. Bu durumda Şekil 2-10'da sonuçları verilen toplam küme adetleri her üç protokolde de ağın eş oranda kapsanması sonucu elde edilen değerlerdir ve kümeleme performansını doğru olarak yansıtmaktadır.

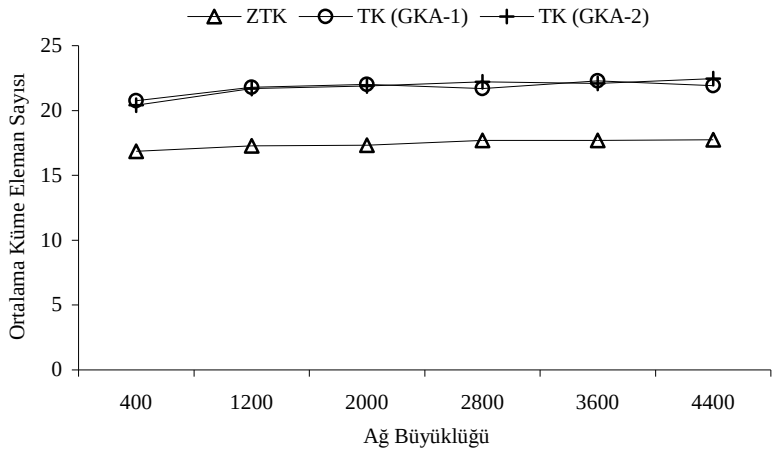



**Şekil 2-11:** Artan ağ eleman sayısına bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinde ayrık kalan düğüm sayısının değişimi.

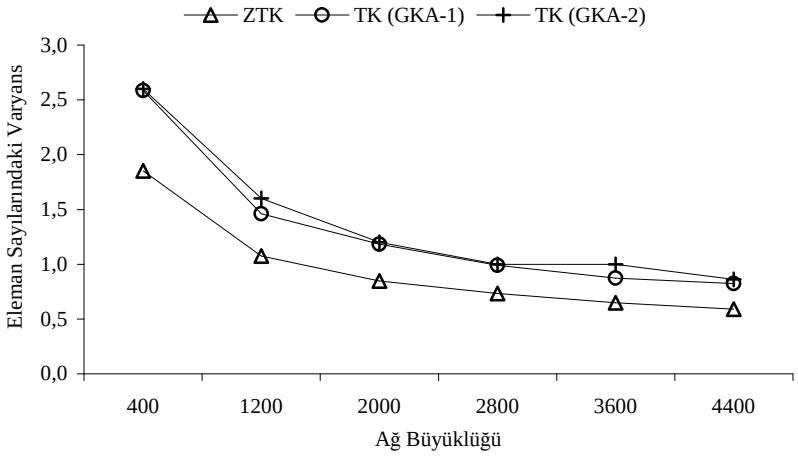
ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinde oluşturulan kümelerin ortalama eleman sayısının artan ağ büyüklüğüne bağlı olarak nasıl bir karakteristik gösterdiği Şekil 2-12 ve Şekil 2-13'de verilmiştir. Şekil 2-12'deki sonuçlara göre sabit yoğunluktaki bir ağda eleman sayısı artsa bile her üç protokolde de erişilen ortalama küme eleman sayısı sabit kalmaktadır. Bunun yanında, ZTK protokolü ile karşılaştırıldığında, TK protokolleri ile yapılandırılmış, eleman sayısı 1'den fazla olan kümelerin ortalama eleman sayısı yaklaşık %30 daha fazladır ve dolayısıyla sınır değere daha yakındır.

Şekil 2-13'de verilen sonuçlar ise eleman sayısı 1'den büyük kümelerin büyüklükleri arasındaki değişimi gösteren varyansı vermektedir. Elde edilen sonuçlara göre her üç protokol uygulamasında da oluşturulan kümelerin büyüklükleri arasındaki varyans değerleri ağ eleman sayısı arttıkça birbirine yakınlaşmaktadır. Burada, ortalama küme eleman sayısının düşük çıkması nedeni ile ZTK protokolü ile oluşturulan toplam küme adedinin daha fazla olması beklenmelidir ki, Şekil 2-10 bu beklentiyi doğrulamaktadır. TK protokollerinde

ise ortalama eleman sayısı sınır değere daha yakındır ve dolayısıyla ağ eleman sayısı arttıkça daha az sayıda kümeler oluşturulabilmektedir. Şekil 2-13'de ayrıca eleman sayısı arttıkça varyansın azaldığı görülmektedir. Bunun sebebi artan ağ büyüklüğü ile üst sınıra yakın daha fazla kümenin oluşturulmuş olmasıdır. Sonuç olarak TK protokolleri ile ZTK protokolünün küme büyüklük varyanslarının eleman sayısı arttıkça birbirine yakınlaşması ve ortalama küme eleman sayısının TK protokollerinde yaklaşık %30 daha fazla değerde sabit kalması nedeni ile TK (GKA-1) ve TK (GKA-2) protokolleri daha iyi kümeleme performansı göstermektedir.

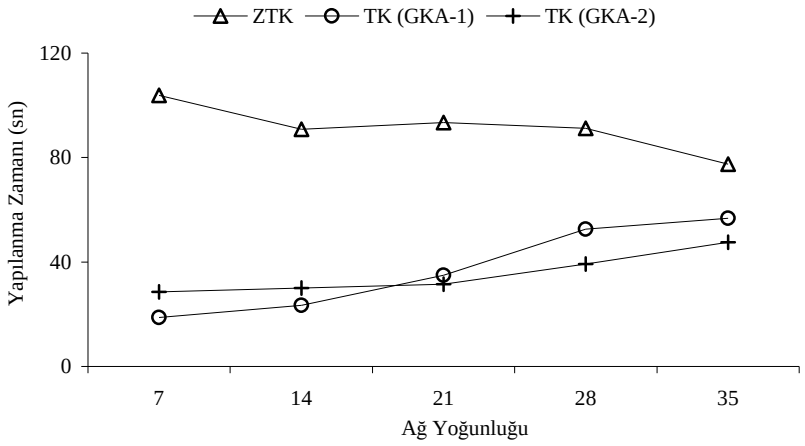



**Şekil 2-12:** Artan ağ eleman sayısına bağlı olarak ZTK, TK (GKA-1) ve TK (GKA-2) protokolleri ile oluşturulan kümelerin sahip olduğu ortalama eleman sayısı.




**Şekil 2-13:** Artan ağ eleman sayısına bağlı olarak ZTK, TK (GKA-1) ve TK (GKA-2) protokolleri ile oluşturulan 1'den büyük eleman sayısına sahip kümelerin varyansı.

Devam eden kısımda Şekil 2-14'den Şekil 2-19'a kadar sabit ağ eleman sayısına sahip bir ağda artan düğüm derecesine bağlı olarak ZTK ve TK (GKA-1) ve TK (GKA-2) protokollerinde oluşan yapılanma performans değişimlerinin benzetim sonuçları sunulmuştur. Bu benzetimlerde bir kümeye ait olabilecek toplam eleman sayısının üst sınırı $\beta$ = 32 ve ağ eleman sayısı $n$ = 400 olarak alınmıştır. İncelenen ağ yoğunlukları {7, 14, 21, 28, 35}'dir.



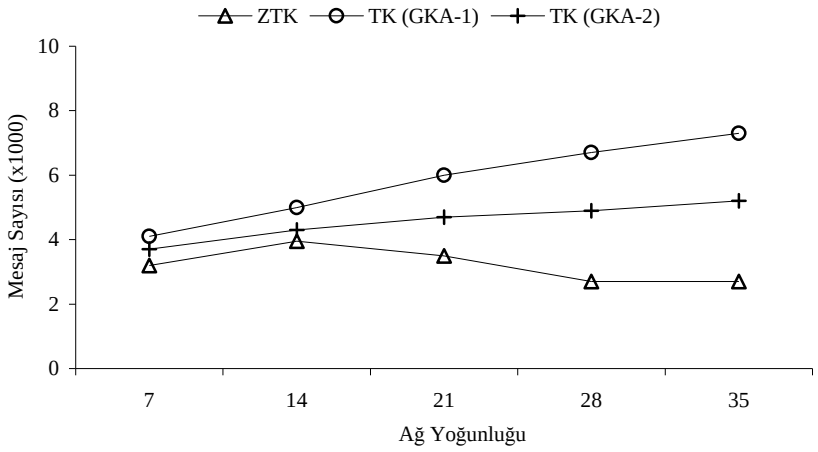

**Şekil 2-14:** Artan ağ yoğunluğuna bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinin toplam yapılanma zamanı.

Şekil 2-14'de her iki kümeleme protokolü için toplam yapılanma zamanının artan ağ yoğunluğuna bağlı değişimi verilmiştir. Sonuçlara göre ZTK protokolünde yoğunluk arttıkça yapılanma zamanı azalmaktadır. Bu karakteristik TK protokollerinde tam tersidir fakat bu protokollerin yapılanma zamanı ZTK protokolünden her zaman daha azdır. Sonuç olarak TK protokolü ZTK protokolüne göre özellikle düşük ağ yoğunluklarında daha az zamanda yapılanmayı tamamlamaktadır. Diğer taraftan, TK (GKA-2) protokolü düşük ağ yoğunlukları 7 ve 14 için TK (GKA-1) protokolüne göre daha yavaş tamamlanmasına rağmen artan ağ yoğunlukları 21-35 için daha hızlı tamamlanmaktadır. Bunun sebebi GKA-1 algoritmasında yapılan bütüneyayımların artan ağ yoğunluğunda daha fazla ağ geçidi adayı düğüm tarafından yapılması ve bu sırada ağın yüklenmesi nedeni ile yeni küme başlatıcısı belirleme işlemlerinin daha fazla zaman almasıdır. TK GKA-2 protokolünde de yoğunluk arttıkça ağ geçidi aday düğüm sayısı artmaktadır fakat bütüneyayım miktarı sıfıra yakın olduğundan yapılanma zamanını etkilememektedir. TK (GKA-2) protokolündeki göreli artış ise yoğunluk arttıkça daha fazla ağ geçidi

adayı birimin oluşmasından ve bu birimlerin zamanlayıcı dolduğunda başlatıcı belirlemek için daha fazla mesajlaşma yapmalarından dolayıdır. Düşük ağ yoğunluklarında ise TK (GKA-2) protokolünde ağ geçidi belirleme için kullanılan zamanlayıcıların toplam zamana etkisi daha baskın olmaktadır.

Şekil 2-15'de kümeleme protokollerinin mesajlaşma maliyetleri sunulmuştur. Bu sonuçlara göre ağ yoğunluğu arttıkça mesajlaşma miktarı ZTK protokolünde cüzi bir artış göstermektedir. Bu artış yoğunluk 14 olana kadar devam etmekte ve sonrasında azalmaya başlamaktadır. Krishnan ve Starobinski (2006) ise geliştirdikleri kümeleme protokolünün mesajlaşma maliyetinin ağ yoğunluğu arttıkça daima az miktarda arttığını belirtmiştir. Aslında ağ yoğunluğunun artmasından dolayı erişilebilecek daha fazla komşu olacağından bütçe dağıtımında daha az mesajlaşma yapılması beklenir fakat Krishnan ve Starobinski (2006), bütçe dağıtımı yaparken komşu düğümlerin durumunu kontrol etmediği için uygulamalarında mesajlaşma maliyeti yoğunluk arttıkça artmaktadır. Çünkü bütçe yollanacak herhangi bir düğüm daha önce zaten yapılandırılmış ise bu düğüme bütçe yollamak anlamlı değildir. Bu durumda yollanan bütçe aynen geri alınır ve bütçeyi tamamen harcayacak uygun düğümler bulunana kadar mesajlaşma devam eder. Yoğunluk arttıkça da önceden yapılandırılmış düğümlere ulaşma olasılığı artar ve bu nedenle daha fazla gereksiz bütçe dağılımı yapılır. Tez çalışmasında geliştirilen yapıda ise gereksiz bütçe dağıtımının engellenebilmesi için komşular belirlenirken aynı zamanda durumları da kaydedilir ve bütçe dağıtılırken sadece *floating* durumda kaydedilmiş komşulara paylaştırılır. Daha önce *floating* olarak kaydedilen bir komşu düğümün bütçe dağıtımı yapılmadan önce başka bir düğüm tarafından yapılandırılmış olma olasılığı tabi ki bulunmaktadır fakat her durumda durumu *floating* olarak kaydedilmiş düğümlere bütçenin yollanması komşu belirleme sırasında başka bir düğüm tarafından zaten yapılandırılmış olan komşu düğümlere gereksiz bütçe yollanmasını engellemektedir. ZTK protokolünde ağ yoğunluğunun 14'den büyük olması durumunda azalmaya başlayan mesajlaşma maliyeti karakteristiği de bu kontrol mekanizmasından ileri gelmektedir. TK (GKA-1) ve (GKA-2) protokollerinde ise ağ yoğunluğu 7 ve 14 iken toplam mesajlaşma maliyetleri ZTK protokolünden cüzi bir miktar fazladır fakat yoğunluk arttıkça bu maliyet farkı artış göstermektedir. Bu artışın sebebi TK (GKA-1) protokolünde bütüneyayım yapan

ağ geçidi adayı birimlerin sayısının artması ve TK (GKA-2) protokolünde ise artan sayıdaki ağ geçidi adayı düğümlerinin, kayıt listelerinde *floating* olarak görünen komşu düğümlerinden uygun olan biri ile yeni kümeleme işlemi yapmaya çalışmalarından ötürü toplamda yapılan bireyayım adetlerinin artmasından dolayıdır. Her iki protokol karşılaştırıldığında ise yoğunluk arttıkça TK (GKA-2) protokolünün TK (GKA-1) protokolüne göre yaklaşık %25 gibi bir oranda daha az mesajlaşma maliyetine sahip olduğu gözlenmiştir. Sonuç olarak TK protokollerinin genel olarak ZTK protokolüne göre daha fazla mesajlaşma maliyetine sahip olması ağ geçidi belirleme için yapılan ekstra mesajlaşmalardan ileri gelmektedir.

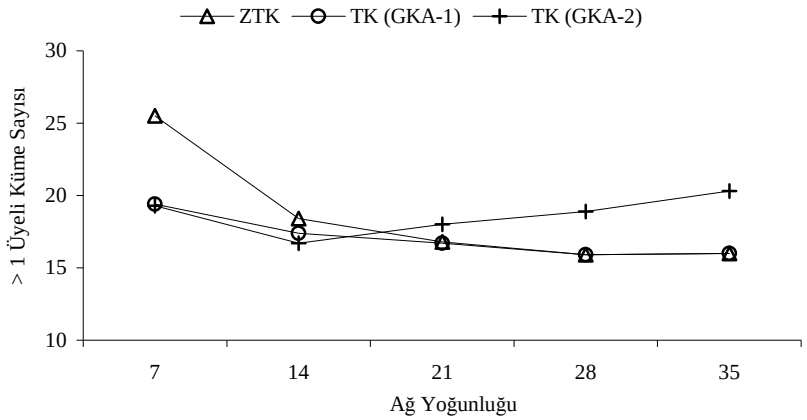



**Şekil 2-15:** Artan ağ yoğunluğuna bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinin toplam mesajlaşma maliyetleri.

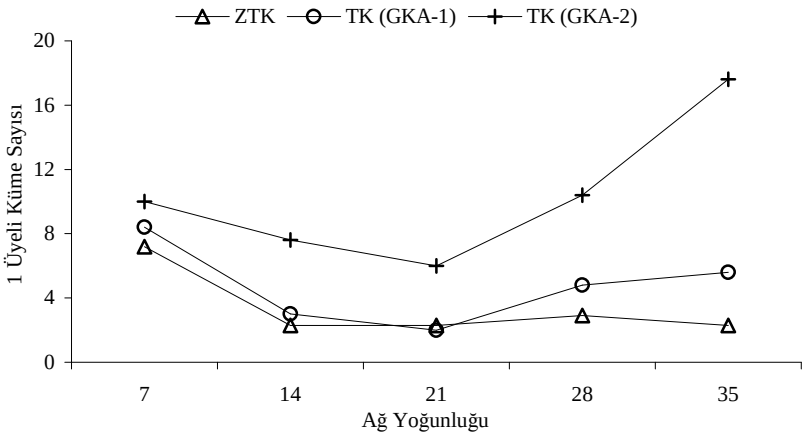
Devam eden kısımda, Şekil 2-16'dan Şekil 2-19'a kadar protokollerin ağ yoğunluğuna bağlı kümeleme performanslarını gösteren benzetim sonuçları verilmiştir.

Şekil 2-16'da verilen sonuçlara göre TK protokollerinde ağ yoğunluğu 7 iken oluşturulan eleman sayısı 1'den büyük toplam küme sayısı ZTK protokolüne göre %20 daha azdır. Fakat yoğunluk arttıkça ZTK ve TK (GKA-1) protokolleri aynı karakteristiği izleyerek azalan miktarda küme oluştururken TK (GKA-2) protokolü ile oluşan küme sayısı artmaktadır. Diğer taraftan ağ yoğunluğu 7 iken tek elemanlı kümelerin sayısı her üç protokolde de yaklaşık aynıdır fakat TK (GKA-2) protokolünde yoğunluk arttıkça diğer protokollere göre tek elemanlı küme sayısında ciddi bir artış gözlenmektedir (Şekil 2-17). Bunun sebebi artan

yoğunlukta ağ geçidi adaylarının sayısının da artması ve bu adayların her birinin potansiyel bir tek elemanlı küme oluşturucusu olmalarıdır. Bu sonuçlar birlikte değerlendirildiğinde artan ağ yoğunluklarında TK (GKA-2) protokolünün kümeleme performansının düşük olduğu, TK (GKA-1) ve ZTK protokol performanslarının ise aynı olduğu görülmektedir. En düşük ağ yoğunluğunda ise TK (GKA-1) ve TK (GKA-2) protokolleri birbirine yakın ve ZTK protokolünden daha iyi performans göstermektedir.



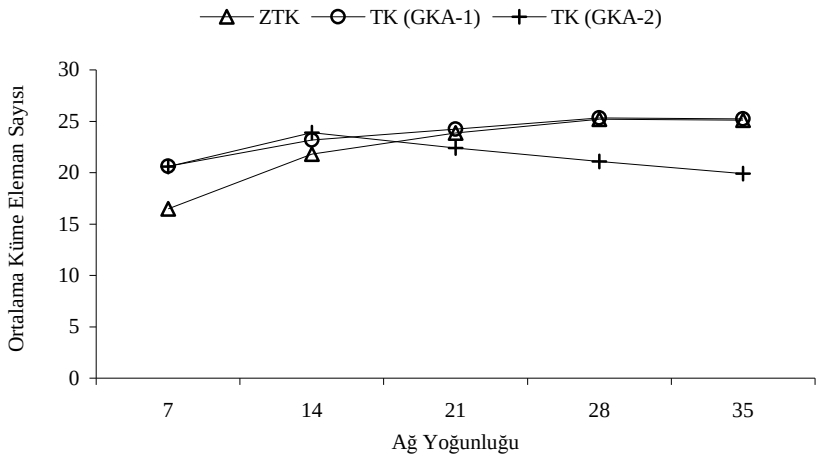

**Şekil 2-16:** Artan ağ yoğunluğuna bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokolleri ile oluşturulan üye eleman sayısı 1'den büyük toplam küme sayısı.




**Şekil 2-17:** Artan ağ yoğunluğuna bağlı ZTK, TK (GKA-1) ve TK (GKA-2) protokollerinde ayrık kalan düğüm sayısı değişimi.

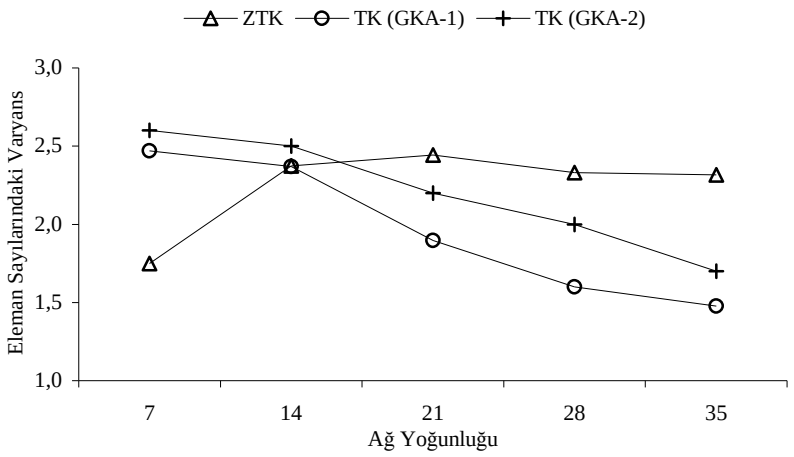
Şekil 2-18'de üç protokol için oluşturulan kümelerin ortalama büyüklüklerinin değişimi verilmiştir. Buna göre ağ yoğunluğu 7 olduğunda TK protokolleri ile oluşturulan kümelerin ortalama eleman sayısı ZTK protokolüne göre yaklaşık %30 daha yüksektir. Yoğunluk 14 ve daha büyük olduğunda ise

ortalama büyüklük yaklaşık olarak eşit düzeydedir. TK (GKA-2)'de ise büyüklük azalmaktadır.



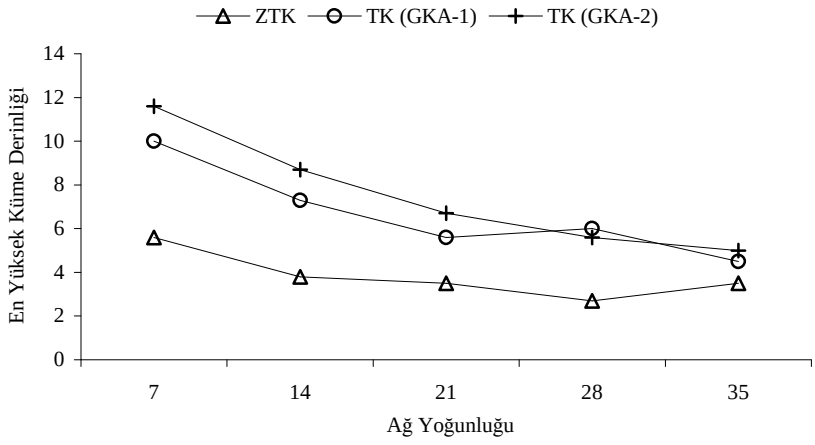

**Şekil 2-18:** Artan ağ yoğunluğuna bağlı olarak ZTK, TK (GKA-1) ve TK (GKA-2) protokolleri ile oluşturulan kümelerin sahip olduğu ortalama eleman sayısı.

Şekil 2-19'da verilen sonuçlar incelendiğinde ise yoğunluk 7 iken TK protokollerinde küme eleman sayıları arasındaki varyans daha yüksektir. Bu sonuçlara göre yoğunluk 7 olduğunda 1'den fazla elemana sahip kümelerin toplamının %20 daha az olması, 1 elemanlı kümelerin sayısının yaklaşık eşit olması ve ortalama küme eleman sayısının %30 daha yüksek olması nedeni ile TK protokollerinde performans daha yüksektir. Yoğunluk yükseldikçe ZTK ile TK (GKA-1) protokollerinin kümeleme performansları birbirine yaklaşmaktadır ve yoğunluk 14'den yüksek olduğunda ZTK ile daha az sayıda tek elemanlı küme oluşturulmaktadır. Burada, ZTK ve TK (GKA-1) protokollerinde 1'den fazla elemanlı küme adedinin ve ortalama küme büyüklüklerinin eşit olduğu düşünüldüğünde, ağ yoğunluğu arttıkça varyansın artması ZTK protokolünde küme denkliğinin azalması anlamına gelmektedir. TK (GKA-1) ve TK (GKA-2) protokollerinde ise yoğunluk arttıkça varyans azalmaktadır ve dolayısıyla küme eleman sayıları daha denktir. Fakat TK (GKA-2) ile oluşturulan kümelerin ortalama eleman sayısı yoğunluk arttıkça azalmaktadır. Dolayısıyla artan yoğunluklarda sınır değere yakın daha fazla küme oluşturmasından dolayı TK (GKA-1) protokolü diğerlerinden daha iyi performans göstermektedir. Düşük ağ yoğunluklarında ise TK (GKA-1) ve (GKA-2) protokolleri eşit ve ZTK protokolüne göre daha iyi performans göstermektedir.




**Şekil 2-19:** Artan ağ yoğunluğuna bağlı olarak ZTK, TK (GKA-1) ve TK (GKA-2) protokolleri ile oluşturulan 1'den büyük eleman sayısına sahip kümelerin varyansı.

Son olarak kümeleme protokolleri ile yapılanan ağda oluşan kümelerin sahip olduğu en fazla derinlik Şekil 2-20'de verilmiştir. Burada düşük küme derinliği daha iyi kümeleme performansı demektir çünkü küme oluşumu kümeyi oluşturmaya başlayan başlatıcı düğümün etrafında daha dengeli oluşmaktadır. Benzetim sonuçlarına göre ZTK protokolünde küme derinliğindeki değişim TK protokollerine göre daha iyi durumdadır. Bunun sebebi ZTK protokolünde başlatıcıların uzaysal olarak ayrık bölgelerde aktif olması ve küme genişlemesinin daha eş yönlü yapılmasıdır. TK protokollerinde ise kümeler çoğunlukla belirli bir yöne doğru genişleme durumundadır.



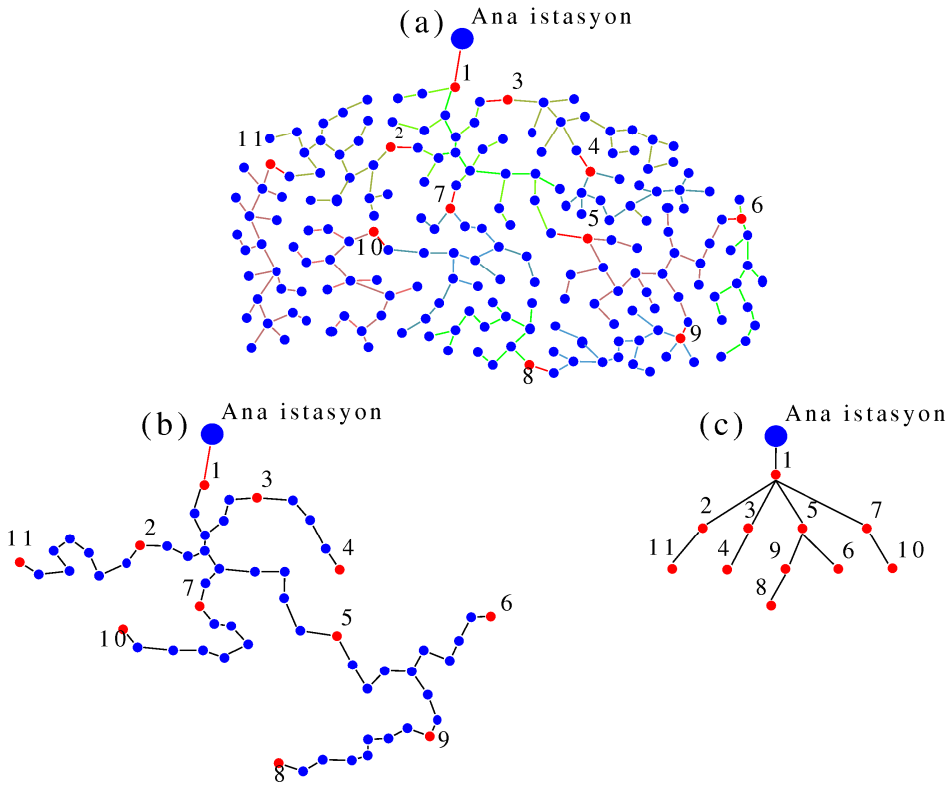

**Şekil 2-20:** Artan ağ yoğunluğunda sabit ağ eleman sayısına bağlı olarak ZTK TK (GKA-1) ve TK (GKA-2) protokollerinde küme derinliği değişimi.

## 2.4 Küme Başları Arası Bağlantı Ağacı Oluşumu

Kümeler yapılandıktan sonra kümeler arası bağlantı hattının ana istasyon tarafından oluşturulabilmesi için tekrarlı olarak oluşturulan kümelerden oluşumunu tamamlamış ve yeni bir küme başlatmamış olan kümelerin başları kendilerini seçen üst seviye kümelerin ağ geçidi düğümlerine küme kimlik numaralarını bireyayar. Bu kimlik numarası aslında küme başının kimlik numarasıdır ve bu numara ağda sadece bu düğüme dolayısıyla bu kümeye aittir. En az bir küme başlatmış kümelerin başları ise bu kümeleri başlatan ağ geçidi düğümlerinden alt seviyedeki kümelerin ağaç haline getirilmiş bağlantı bilgilerini bekler. Tüm ağ geçidi düğümlerinden bilgi alan küme başı, bu bilgileri kendisini başlatan kümenin ağ geçidi düğümüne iletir ve bu iterasyon ana istasyona kadar bu şekilde devam ettirilir. Sonucunda kümeler arası bağlantı bilgisini veren nihai ağaç ana istasyonda oluşturulmuş olur. Bu tür bir bağlantı ağacının örnek gösterimi Şekil 2-21'de verilmiştir.

**Teorem IV:** Kümeler arası bağlantı hattı ana istasyona doğru döngüsüz bir yol izler.

**İspat:** TK protokolünde Çizelge 2-1'de verilen düzenlenmiş *persistent* algoritmasına göre yapılandırılmış ve bir kümeye dahil olmuş bir düğüm bağlı olduğu kümeyi genişletecek düğümleri veya yeni küme başlatıcısını seçmek için sadece ve sadece henüz hiçbir kümeye bağlanmamış ve dolayısıyla *floating* durumunu değiştirmemiş komşu düğümlerini kullanabilir. Bu şekilde ana istasyon kök olmak üzere ağda yapılandırılan herhangi bir düğümün kendisini yapılandıran ebeveyn düğümün bağlı olduğu ağaç dalının haricinde ana istasyona ulaşmak için kullanabileceği alternatif bir başka patikası yoktur. Dolayısıyla küme başlarının da ana istasyona doğru iletim için kullanabileceği yalnızca bir patika bulunmaktadır. Bu da ana istasyona doğru kümeler arası bağlantıyı döngüden bağımsız yapmaktadır. □




**Şekil 2-21:** Küme başları arası bağlantı ağacı oluşumu örneği: (a) Örnek duyarga ağı (b) Küme başları arası bağlantı hatları (c) Yapılanma sonucu kümeler arası bağlantı ağacı.

**Teorem V:** TK protokolü ile yapılanma tamamlandığında oluşan kümeler arası bağlantının ana istasyon tarafından kümeler arası bağlantı ağacına dönüştürülebilmesi için yapılan toplam mesajlaşma maliyeti, $n$ toplam ağ eleman sayısı, $\beta$ küme eleman sayı sınırı olmak üzere $O(n\lg\beta)$'dır.

**İspat:** TK protokolünde yapılanmasını tamamlamış kümelerin içinden yeni bir küme oluşumu yapmış herhangi bir ağ geçidi düğümü bulunmayan kümelerin başları kendilerini seçen kümelerin ağ geçidi düğümlerine küme kimlik numaralarını bireyayar. Bu mesajları alan ağ geçidi düğümleri mesajı kendi küme başlarına doğru küme içi bağlantı ağacında çok zıplamalı bireyayım ile ulaştırırlar. Bu işlem sırasında her bir ağ geçidi düğümü için en fazla küme içi bağlantı ağacının yüksekliği kadar mesaj iletimi yapılabilir. Bu yüksekliğin beklenen değerinin ise eleman sayısı $n$ olmak üzere $O$ ($\lg n$) olduğu Teorem III'de belirtilmişti. Küme içi bağlantı ağacı için ise küme eleman sayısı en fazla $\beta$ olduğunda bu yükseklik $O$ ($\lg\beta$)'dır. Bir kümenin sahip olabileceği ağ geçidi sayısı ise en fazla o kümede oluşturulan uç düğümlerin sayısı kadardır. Bu

düğümler en fazla, kümeyi oluşturan düğüm hariç, diğer düğümlerin sayısı kadardır ($\beta$ - 1). Bu sonuçla her bir ağ geçidi düğümü için en fazla küme içi bağlantı ağacının beklenen yüksekliği kadar bireyayım yapıldığında yapılan toplam küme içi bireyayım mesajlaşma karmaşıklığı $O(\beta\text{-}1).O(\lg\beta) = O(\beta\lg\beta)$'dır. Bu işlem kümeler arası bağlantı ağacının yaprak kümeleri hariç tüm kümeler tarafından gerçekleştirilir. Ağ yapılanırken oluşturulabilecek kümelerin sayısı tüm kümelerin $\beta$ elemandan oluştuğu varsayıldığında $n/\beta$'dır ve küme eleman sayısı sabit kalmak üzere artan ağ eleman sayısı $n$ ile doğrusal bir bağlantı içindedir ($O(n)$). Bu doğrusal bağlantı Şekil 2-10'da benzetim yolu ile de gösterilmiştir. Sonuç olarak yapılanan ağda küme bağlantı ağacının ana istasyonda oluşturulabilmesi için yapılan toplam mesajlaşma karmaşıklığı aşağıdaki gibi hesaplanır;

$$O(n/\beta).O(\beta \lg\beta) = O(n\lg\beta) \square$$

Yapılanmadan sonra kümeler arası bağlantı ağacında yapılabilecek değişiklikler için bu ağacın tüm küme başlarının belleğinde tutulması gerekmektedir. Dolayısıyla bu ağacın bellekte kaplayacağı yer ağda oluşturulan küme sayısına ve dolayısıyla ortalama küme eleman sayısına bağlıdır. Bu nedenle küme eleman sayısı sınırı ağ büyüklüğüne ve duyarga düğümlerinin bellek kısıtlarına göre belirlenmelidir.

## 2.5 Özet ve Yorumlar

Çalışmanın bu bölümünde tez kapsamında geliştirilen ve Tekrarlı Kümeleme (TK) adı verilen geniş ölçekli kablosuz ağlar için ölçeklenebilir yapılanma zamanına sahip, düşük ağ yoğunluklarında etkin şekilde kendi kendine organize olabilen çok zıplamalı bir kümeleme protokolünün tasarımı, analizi ve benzetime dayalı performans sonuçları verilmiştir. Bu protokolde, oluşturulan kümelerin eleman sayısını sınırlandırmak için orijinali Krishnan ve Starobinski (2006) tarafından önerilen ve tez kapsamında durum kontrolü mekanizması ile geliştirilen *persistent* algoritması kullanılmıştır. Eklenen bu durum kontrolü ile orijinal *persistent* algoritmasındaki, özellikle yüksek yoğunluklu (>14) ağlarda

belirgin olan, gereksiz bütçe dağıtımı miktarı azaltılmış ve bu şekilde toplam mesajlaşma maliyetleri azaltılmışltır.

TK protokolü ile yapılanan ağda ilk küme hariç tüm kümelerin oluşumu, yapılanması tamamlanmış diğer küme elemanları tarafından başlatılır. Yeni kümeleri başlatma işi yapılanan kümelerdeki uç düğümler tarafından gerçekleştirilir. Bu uç düğümlerin ağdaki kümeleme performansını en uygun düzeyde tutabilecek şekilde belirlenebilmesi için tez çalışmasında GKA-1 ve GKA-2 algoritmaları geliştirilmiş ve bu algoritmaları kullanan TK protokollerinin benzetim yolu ile performans analizleri yapılmıştır. Bu analizler Krishnan ve Starobinski (2006) tarafından önerilen zamanlayıcı tabanlı kümeleme (ZTK) protokolünün benzetim sonuçları ile de karşılaştırılmış, bu protokollerin avantaj ve dezavantajları ortaya konmuştur.

Benzetim sonuçları özetlendiğinde özellikle yapılanma zaman maliyetleri açısından TK protokolleri ZTK protokolüne göre ciddi anlamda avantajlı durumdadır. Çünkü farklı ağ büyüklüğü ve yoğunluklarında TK protokolleri ZTK protokolünden 4-5 kat daha hızlı şekilde yapılanmayı tamamlamaktadır. Kümeleme performansları karşılaştırıldığında ise farklı ağ yapılanma parametrelerinde her protokol için farklı performans sonuçları alınmıştır. Bu sonuçlara göre özellikle düşük yoğunluklu (7-8) ağların yapılanmasında TK protokolleri ZTK protokolüne göre %20 daha az sayıda küme oluşturmakta ve oluşturulan kümelerin eleman sayı ortalaması %30 daha yüksek olmaktadır. Dolayısıyla TK kümeleme performansı ZTK'den daha iyi bir karakteristik göstermektedir. Ağ yoğunluğunun artması durumunda ise ZTK ile TK (GKA-1) protokolleri kümeleme performansları açısından benzer karakteristik göstermektedir fakat TK (GKA-2) kümeleme performansı düşmektedir. Mesajlaşma maliyetleri karşılaştırıldığında ise kümeleri başlatmak için yapılan mesajlaşmalar nedeni ile TK protokollerinin toplam mesajlaşma maliyeti ZTK protokolüne göre %30 daha fazladır. Bu mesajlaşma farkı ilk aşamada TK protokolleri için negatif gözükmektedir fakat TK protokolleri ile yapılanma tamamlandığında oluşan ağ bağlantı yapısında ana istasyona iletimi sağlayan doğal bir yönlendirme yolu da oluşturulmuş olmaktadır. ZTK protokolü ile

yapılanmada ise böyle bir bağlantı hattı yoktur ve bunun için yapılanma tamamlandıktan sonra ayrıca mesajlaşmalar yapılması gerekmektedir.

TK protokolleri kendi içinde karşılaştırıldığında ise TK (GKA-1) protokolünün mesajlaşma maliyetleri yapılan bütüneyayımlar nedeni ile özellikle ağ yoğunluğu arttıkça belirgin şekilde artmaktadır. TK (GKA-2) protokolünde ise bu maliyet %10 civarında düşürülmüştür fakat artan ağ yoğunluklarında kümeleme performansı TK (GKA-1) protokolüne göre daha kötüdür. Zaman maliyeti ele alındığında ise düşük yoğunluklu ağlarda TK (GKA-2) protokolü TK (GKA-1) protokolüne göre belli bir ofset değerinde daha geç tamamlamaktadır fakat bu performans bile ZTK protokolünden daha iyidir. Bu ofset değeri benzetim yapılanmasında 18 saniye olarak ölçülmüştür.

Sonuç olarak TK protokolleri düşük yoğunluklu ağlarda yapılanma zamanı ve oluşturulan kümeler açısından ZTK kümeleme protokolüne göre daha iyi performans sergilemektedir. Mesajlaşma maliyetleri açısından ise ZTK protokolü daha iyi gibi görünmesine karşın TK protokolündeki ekstra mesajlaşma maliyeti ana istasyon ile kümeler arasında döngüden bağımsız bir kümeler arası bağlantı hattı oluşumunu da sağlamaktadır. Fiziksel olarak oluşturulan bu hattın ana istasyon tarafından kümeler arası bağlantı ağacına dönüştürülebilmesi için ihtiyaç duyulan mesajlaşma karmaşıklığı $O$ ($n\lg\beta$) olarak hesaplanmıştır.

Son olarak, düşük yoğunluklu ağlar için zaman ve mesajlaşma maliyetlerinin önem derecesi değerlendirilerek TK (GKA-1) ve TK (GKA-2) protokolleri arasında ayrıca bir tercih yapılabilir. Uygulamaya göre yapılanma zamanı daha önemli ise TK (GKA-1) protokolünün kullanımı daha uygundur. Mesajlaşma maliyetinin daha öncelikli olması durumunda ise TK (GKA-2) protokolü daha iyi bir performans sergilemektedir.

# 3 AĞ SEVİYESİ ANAHTAR BELİRLEME

Bölüm 2'de detayları verilen Tekrarlı Kümeleme (TK) protokolünde kendi kendine organize olan duyarga düğümlerinin sadece haberleşme menzili içindeki komşuları ile haberleştiği bir ortamda, güvenli haberleşme için düğümler arasında paylaşılan ikili oturum anahtarlarının belirlenmesi ve yapılanma sonucu oluşturulan kümelerde ortak anahtar belirleme işlemlerinin tanımlanması bu bölümün temel konusunu oluşturmaktadır.

Alana yerleştirildikten sonra yapılandırılmaya başlanan duyarga düğümlerinin güvenli haberleşme için kullanacağı ortak anahtarların belirlenmesinde ağı yapılandıran protokol gereksinimlerinin ve ağ ile ilgili yapılan varsayımların dikkate alınması gereklidir. Çünkü bunlar aynı zamanda tanımlanacak anahtar belirleme protokolünün de gereksinimleri ve varsayımlarıdır. Bu doğrultuda, Bölüm 2.1'de bahsedildiği gibi, tipik bir duyarga düğümü için haberleşme menzilinin kısıtlı olması ve kablosuz ağlarda fiziksel ortam kullanımındaki kısıtlar nedeni ile TK protokolünde etkin yapılanma için ağda mümkün olan en düşük duyarga düğüm derecesinin varsayımı söz konusudur. Bu durumda ağ seviyesinde çalışacak anahtar belirleme protokolünün de düşük ağ yoğunluğunda etkin olarak çalışması gerekmektedir. Ayrıca TK protokolünde duyarga düğümlerinin sınırlı kaynaklara sahip eş birimlerden oluşması ve yapılanma sırasında sadece tek zıplama mesafesindeki komşular üzerinden kendi kendine yapılanması nedeni ile anahtar belirleme işlemlerinde özel birimlerin ve merkezi anahtar dağıtıcılarının varsayılması söz konusu değildir. Bu nedenle ağ seviyesinde TK protokolü ile çalışacak anahtar belirleme protokolünün aynı şekilde sadece tek zıplamadaki komşular üzerinden yapılan mesajlaşmalar ile paylaşılacak ikili oturum anahtarlarını belirleyebiliyor olması gerekir.

Kablosuz duyarga ağ literatüründe, ağı oluşturan düğümler arasında güvenli haberleşme için kullanılacak ikili ortak anahtarların belirlenmesi için hesaplama maliyetinin azlığı nedeni ile çoğunlukla simetrik anahtar dağıtımı üzerinde çalışılmıştır. Bunların içinden sadece haberleşme menzilindeki komşu düğümler ile mesajlaşarak paylaşılan ikili oturum anahtarlarını belirleyen ve bu şekilde

merkezi bir anahtar dağıtıcısına ihtiyaç duymayan ilk çalışma rastsal anahtar ön dağıtımı yöntemini öneren Temel Şema'dır (Eschenauer and Gligor, 2002). Sonraki çalışmalar temelde aynı yöntemi kullanmak ile birlikte farklı yaklaşımlar ile anahtar belirleme protokolünün dayanıklılığını artırmaya ve kaynak kullanımını azaltmaya çalışmışlardır. Chan ve ark. (2003), haberleşme maliyetlerinden ödün vererek Temel Şema'nın direncini artırmaya çalışmış ve sınırlı büyüklükteki ağlar için rastsal ikili anahtar dağıtımı yaparak kimliklemenin yapılabileceğini göstermiştir. Zhu ve ark. (2003b) ikili anahtar belirlemede anahtarı oluşturan bileşenleri alternatif patikalar üzerinden ileterek, Pietro ve ark. (2003) komşu duyarga düğümlerinin katkıları ile dayanıklılığı artırmaya çalışmıştır. Liu ve Ning (2003a) anahtar dağıtımını kriptografik özellikli polinom fonksiyonlar üzerinden yaparak Temel Şema'nın dayanıklılığını artırmaya çalışmıştır. Du ve ark. (2004) ise konuşlanma bilgisine dayanarak ağı alt bölgelere ayırmış ve komşu bölgelerdeki duyarga düğümlerinin ortak anahtar paylaşım olasılığını artırıp bellek kullanımını azaltmış ve bu sayede dayanıklılığı da artırmıştır. Bunların yanında GPS gibi yöntemlerle anahtar belirleme etkinliğini artıran yöntemler de geliştirilmiştir (Chan and Perrig, 2005).

Diğer taraftan, duyarga ağlarında ikili oturum anahtarlarının belirlenmesi için daha maliyetli olan açık anahtar şifreleme yöntemleri de önerilmiştir fakat bu çalışmalar daha çok kaynak ve uygulanabilirlik analizi seviyesindedir. Malan ve ark. (2004), duyarga ağlarında ECC (*Elliptic Curve Cryptography*) (Hankerson et al., 2004) tabanlı ilk açık anahtar şifreleme uygulamasını yapmıştır. Du ve ark. (2005), açık anahtarların kimliklenmesi için simetrik yöntemlerin kullanıldığı bir çalışma yapmıştır. Wander ve ark. (2005), RSA ve ECC açık anahtar şifreleme algoritmalarının duyarga düğümlerinin üzerindeki enerji analizini yapmış ve Piotrowski ve ark. (2006) bunların düğüm ömrüne etkilerini incelemiştir. Bu çalışmalarda genel olarak simetrik yöntemlerin enerji maliyetlerinin açık anahtar belirleme yöntemlerinden çok daha düşük olduğu kabul edilmiş fakat ECC tabanlı açık anahtar şifreleme işlemlerinin belirli aralıklar ile yapılmasının duyarga ağları için uygulanabilir olduğu sonucuna varılmıştır. Bunun yanında, hesaplama maliyetleri daha fazla gibi görünse de açık anahtar şifreleme tabanlı yöntemlerin saldırılara karşı sunduğu dayanıklılığın yüksek güvenlik gerektiren sistemler için tercih edilebilir olduğu da unutulmamalıdır.

Bu sonuçlarla, bölümün devam eden kısımlarında ilk olarak alana konuşlanmadan sonra ağ yapılanırken düğümler arasında paylaşılacak ikili oturum anahtarlarının belirlenmesi için kullanılacak anahtar belirleme protokol gereksinimleri belirlenmiştir. Bu amaçla seçilen dağıtım tabanlı ve asimetrik anahtar şifreleme yöntemlerinin TK protokolüne uyumluluğu ve gereksinimleri karşılama maliyetleri incelenmiş ve benzetim yolu ile elde edilen karşılaştırmalı sonuçlar verilmiştir. Bölümün sonunda ikili oturum anahtarlarını belirlemiş olan ağda eleman eklenmesi veya çıkarılması durumunda yapılacak anahtar güncelleme işlemleri tanımlanmış ve mesajlaşma maliyet analizleri verilmiştir.

## 3.1 Anahtar Belirleme Protokol Gereksinimleri

Bölüm 2'de verilen TK protokolü ile yapılanma zamanında oturum anahtarlarının da belirlenmesi ve bu sayede ağ iletişiminin güvenli şekilde yapılabilmesini sağlayacak altyapının oluşturulması için anahtar belirleme protokolünün yapılanma protokolü ile uyumlu bir şekilde çalışması gerekmektedir. Buna göre anahtar belirleme protokolü:

*i.* Duyarga düğümlerinin eşit kaynaklara sahip olduğunu varsaymalıdır.

*ii.* Ağ yapılanma protokolünün (TK) işleyişini ve performansını etkilememelidir.

*iii.* Düşük ağ yoğunluklarında ($\cong 8$) etkin çalışabilmelidir.

Bu gereksinimlerin sağlanmasında altyapı olarak LEAP (Zhu et al., 2003a) çalışmasında belirlenen katmanlı güvenlik yapısına benzer bir yapı kullanılabilir Fakat anahtar belirleme ve kümeleme gereksinimleri nedeni ile bahsedilen çalışmada yapılan bazı kabuller bu tez çalışması için geçerli değildir. Bu kabullerde yapılan değişiklikler aşağıdaki gibidir:

- Ağda sürekli kullanılan ve ana istasyon ile düğümler arasında doğrudan bir anahtar paylaşımı bulunamaz. Çünkü TK protokolünde ağdaki düğümlerin ana istasyon ile doğrudan iletişimi varsayılmamıştır.

- Ağda sürekli kullanılan ve tüm düğümler tarafından bilinen bir grup anahtarı yoktur. Çünkü ağı oluşturan düğümlerden herhangi birini ele geçiren bir saldırgan bu anahtarı da ele geçirip ağ güvenliğini devre dışı bırakabilir.

Bunlar dışında Zhu ve ark. (2003a) tarafından tanımlanan ikili oturum anahtarları ve küme anahtarları tez çalışmasında da tanımlanmıştır. Bu tanımla TK protokolü ile yapılanmada ikili oturum anahtarlarının belirlenmesi anahtar dağıtımı veya açık anahtar şifreleme yöntemlerinden biri ile yapılabilir fakat öncesinde bu seçeneklerden hangisinin TK protokolü ile daha etkin çalışabileceğinin belirlenmesi gereklidir. Burada etkin çalışmadan kasıt anahtar belirleme işleminin TK protokolünün yapılanma performansına etkisinin ve sisteme getirdiği yükün mümkün olduğu kadar az olması buna karşın sağladığı güvenlik seviyesinin yüksek olmasıdır. İkili ortak anahtarlar belirlendikten sonra da ağa yeni eleman katılması veya ayrılma işlemlerinde sistemin güvenliğini tehlikeye düşürmeyecek şekilde anahtar güncelleme işlemlerinin yapılması gerekmektedir.

Tez çalışmasının bu bölümünde anahtar dağıtımı ve açık anahtar şifreleme olmak üzere iki ana başlıkta incelenen anahtar belirleme protokollerinin TK protokolü ile ağ seviyesinde uygulanması ve en uygun performansı yakalayan yapılanma parametrelerinin belirlenmesi amaçlanmıştır. Daha sonra, yapılanmış ağa yeni eleman katılması ve ağdan ayrılma durumlarında haberleşmede kullanılan ortak anahtarların güncellenmesi ile ilgili anahtar dağıtım yöntemleri tanımlanmıştır.

Anahtar belirleme protokolünün TK protokolünün evrelerine bağlı olarak konuşlanmadan önce ve alana yerleştikten sonra yapılanma olmak üzere iki evresi vardır.

### 3.1.1 Konuşlanmadan önce yapılanma

Tüm duyarga düğümleri alana yerleştirilmeden önce kimlik numarası (*ID*) ve ikili oturum anahtarlarını oluşturabilecekleri parametreler ile yüklenirler.

Güvenlik için yüklenecek parametreler anahtar dağıtımı ve açık anahtar şifreleme tabanlı anahtar belirleme protokolleri için farklıdır. Amaç yapılanma zamanında bu parametreleri kullanarak haberleşilecek komşu düğümler ile paylaşılacak ikili oturum anahtarlarını belirlemek ve bu şekilde tüm ağ haberleşmesini güvenli kılmaktır.

### 3.1.2 Konuşlanma sonrası yapılanma

Duyarga düğümleri alana rast gele yerleştirildikten sonra ana istasyonun en yakın duyarga düğümüne yapılanma mesajı göndermesi ile ağ yapılanması başlar. Bu işlemin detayı Bölüm 2.2'de verilmiştir. Bu sırada herhangi bir duyarga düğümü ilgili yapılanma mesajını aldıktan sonra öncelikle tüm komşuları ile ikili ortak anahtar belirlemeye çalışır. Bu düğümün fiziksel komşuluğunda olup ortak anahtar belirleyemeyen komşu düğümler ise kümeleme işlemlerine dahil edilmezler. Anahtar belirleme protokolü ve kümeleme protokolünün karşılıklı performanslarına olan etkileri işte buradadır. Eğer yeteri kadar komşu ile anahtar belirlenemez ise bu durumda herhangi bir duyarga düğümünün komşuları ile gerçekte oluşturabildiği yerel bağlantı sayısı fiziksel olarak tek zıplama mesafesinde bulunan komşularının sayısından daha az olacaktır ve bu nedenle ağın kümeleme performansı düşecektir. Diğer taraftan istenen yerel bağlantı oranına ulaşmak için güvenlik protokolünün yapılanma parametrelerinde yapılacak değişiklikler duyarga düğümlerindeki kaynakların daha fazla kullanılmasına sebep olacaktır.

## 3.2 Anahtar Belirleme Protokolleri

Bu bölümde, TK protokolü ile birlikte performans analizi yapılan anahtar dağıtımı ve açık anahtar şifreleme tabanlı anahtar belirleme protokolleri hakkında ön bilgi verilmiştir. Bu amaçla ilk olarak anahtar dağıtımı tabanlı protokollerin ilk örneği olan Temel Şema ve sonra kablosuz duyarga ağları için önerilen açık anahtar belirleme protokolü ECDH protokolleri özetlenmiştir.

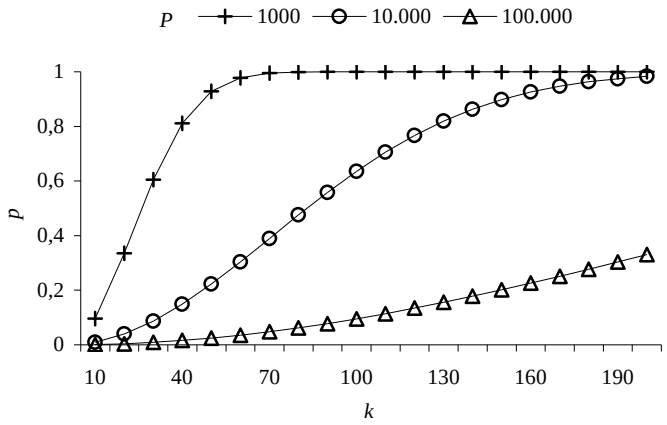
### 3.2.1 Dağıtım tabanlı anahtar belirleme

Duyarga ağını oluşturacak düğümler konuşlanmadan önce tek bir noktada yapılandırıldığı için bu aşamada düğümlere yüklenecek anahtarlar ile konuşlanmadan sonra ikili oturum anahtarlarının belirlenmesi mümkündür. Anahtar dağıtımının güvenlik açısından en üst seviyede olabilmesi için her düğüme olası her komşusu ile ortak paylaştığı bir anahtarın önceden yüklenmiş olması gerekmektedir. Bu yöntemde, $n$ ağdaki toplam düğüm sayısı olmak üzere, toplamda $n - 1$ adet farklı anahtarın ekstra olarak her düğüme yüklenmesi gerekir ki, sınırlı belleği olan binlerce duyarga düğümü içeren ağlar için bu uygulanabilir değildir. Eschenauer ve Gligor (2002), bu problemi gidermek için rastsal anahtar ön dağıtımı yöntemini geliştirerek duyarga ağlarında anahtar dağıtımını basit, uygulanabilir ve ölçeklenebilir hale getirmiştir. Bu yöntemde alana konuşlanmadan önce her düğüme önceden tanımlanmış bir anahtar havuzu $P$ içerisinden rast gele seçilmiş $k$ adetlik bir anahtar zinciri yüklenir (*anahtar ön dağıtımı*). Alana bırakılan düğümler ilk olarak sahip oldukları anahtarlar üzerinden paylaşılan anahtarları bulmak için keşif yaparlar. Anahtar listelerinde ortak anahtar bulunan herhangi iki düğüm bu anahtarı kullanarak birbirleri ile güvenli haberleşme hattı oluşturur (*ortak anahtar keşfi*). Son aşamada güvenli haberleşmek isteyen herhangi iki düğümün anahtar listelerinde ortak bir anahtar yok ise bu iki düğüme komşu olan ve her ikisi ile ortak anahtarı bulunan üçüncü bir düğüm üzerinden patika anahtarı belirlenmeye çalışılır (*patika anahtarı keşfi*). Eschenauer ve Gligor (2002), ağdaki herhangi iki düğümün en az bir ortak anahtara sahip olma olasılığı $p$ 'yi aşağıdaki gibi hesaplamıştır:

$$p = 1 - \frac{(1 - k/P)^{2(P-k+1/2)}}{(1 - 2k/P)^{(P-2k+1/2)}} \tag{3.1}$$

Bu eşitlikteki $p$ olasılığı ağın fiziksel yoğunluğu $d$'ye bağlı olarak herhangi bir düğümün anahtar keşfi aşamasında güvenli olarak haberleşebileceği beklenen komşu sayısını belirler. Buna göre bir düğümün fiziksel olarak tek zıplamalık haberleşme menzilinde bulunan komşu düğümlerinin beklenen sayısı $d$ olmak üzere ikili oturum anahtarı belirlendikten sonra bu düğüm için erişilebilecek gerçek yerel bağlantı adedi $d' = dp$ formülü ile hesaplanır. Burada $p$ olasılığı $P$

anahtar havuzu büyüklüğüne ve her bir düğüme yüklenecek anahtar adedi *k* değerine bağlıdır (Bkz. Şekil 3-1). Diğer taraftan *d* ve *d′* yoğunluk değerleri ağdaki radyo iletiminin etkinliğini ve ağın genel bağlantı oranını etkilemektedir. Artan *d* değeri ile radyo kanallarının kullanımındaki yoğunluk artacak ve bu nedenle mesaj iletimleri sırasında mesaj kayıpları oluşacaktır. Diğer taraftan *d* değerinin düşük olması *d′* değerinin daha da düşük olması anlamına gelmektedir ki bu durumda ağın genel bağlantı oranının azalması gibi istenmeyen sonuçlar ortaya çıkacaktır. Bu nedenle anahtar dağıtımı parametrelerinin ağ yapılanmasının gereksinimlerine göre seçilmesi gerekmektedir.




**Şekil 3-1:** *P* = 1000, 10.000, 100.000 anahtar havuzu büyüklüklerinde farklı *k* anahtar liste uzunlukları için iki düğüm arasında güvenli hat oluşturma olasılığı *p*'nin değişimi.

Anahtar dağıtımı parametrelerinin ağ yoğunluğu ve ağın genel bağlantı oranı ile ilişkisi ilk olarak Hwang ve Kim (2004) tarafından incelenmiştir. Bu çalışmaya göre, duyarga ağı bir Erdös *Renyi graph* (Erdös and Renyi, 1960) olarak düşünüldüğünde, ağı oluşturan düğümlerin tümünün bağlanma olasılığı $P_c$'nin 1'e yaklaşabilmesi için ihtiyaç duyulan *d′* yoğunluğu, ağdaki toplam duyarga düğüm sayısı *n* ve *c* herhangi bir gerçek sayı olmak üzere, aşağıdaki formül ile hesaplanır:

$$P_c = e^{-e^{-c}} \text{ ve } p = \frac{\ln(n)}{n} + \frac{c}{n} \text{ olmak üzere, } d' = p(n-1) \qquad (3.2)$$

Diğer taraftan Hwang ve Kim (2004), $P_c$'yi 1'e yaklaştırmaya çalışmak yerine, ağda birbirine bağlanan en büyük grubun (*giant component*) ağı oluşturan

düğümlerin toplam sayısı $n$'ye olan oranı $\beta$'yı belirli bir seviyeye kadar çıkarmanın, örneğin %98, ihtiyaç duyulan duyarga düğüm derecesi miktarını ciddi şekilde azaltacağını ($d' = 6$) belirtmiştir. Hwang ve Kim (2004), bu saptamayı yaptıktan sonra $d' > 0$ olmak üzere $\beta$ ile $d'$ ilişkisini aşağıdaki şekilde formülize etmiştir:

$$\beta + e^{-\beta d'} = 1 \tag{3.3}$$

Her iki durumda da $d'$ yerel bağlantı sayısı ile erişilebilen yerel bağlantı olasılığı $p_{required}$ aşağıdaki formül ile hesaplanır:

$$p_{required} = \frac{d'}{d} \tag{3.4}$$

Bu iki yaklaşım arasındaki fark bir örnek ile daha açık şekilde ortaya konabilir. Varsayalım ki $n$ = 10.000 olan bir ağ için fiziksel ağ yoğunluk miktarı $d$ = 50 olsun. Bu durumda $P_c$ = 0,9999 ve $\beta$ = 0,98 için sağlanması gereken $d'$ ve $p_{required}$ değerlerini hesaplayalım.

$P_c$ = 0,9999 için $c$ değeri yaklaşık 9,2103 olarak hesaplanır ve yerine konulduğunda $p$ = 0,001842 olarak bulunur. Buradan $d'$ = 0,001842(10000-1) ≅ 18,419 olarak hesaplanır. Bu değer için $p_{required}$ aşağıdaki gibidir:

$$p_{required_1} = 18{,}419 / 50 \cong 0{,}368$$

$\beta$ = 0,98 olması için sağlanması gereken $d'$ yoğunluğu ise aşağıdaki formül ile hesaplanır:

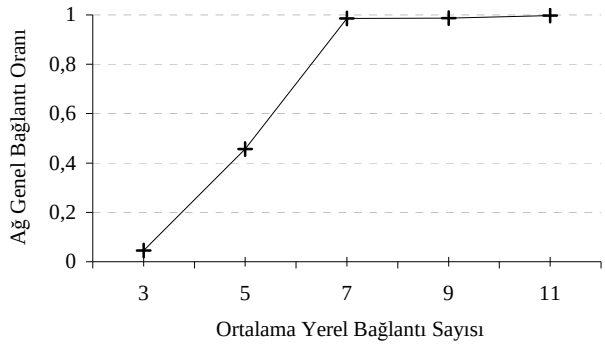
$$d' = -\frac{\ln(1-\beta)}{\beta} \approx 3{,}992$$

Bu değer için $p_{required_2}$ aşağıdaki gibidir:

$$p_{required_2} = 3{,}992 / 50 \approx 0{,}0798$$

Bu $p_{required}$ değerlerinin sağlanabilmesi için her bir düğüme yüklenmesi gereken anahtar listesi uzunluğu $k$, anahtar havuz büyüklüğü $P = 100.000$ olmak üzere $k_1 = 215$ ve $k_2 = 92$ olarak hesaplanır.

Bölüm 2.1'de duyarga düğümlerinin sahip oldukları radyo kapasiteleri ve çevresel faktörler nedeni ile ağın yoğunluğunun sınırlandırılması gerektiğinden bahsedilmişti. Bu sınırlandırma aynı zamanda ağın toplam maliyeti açısından da önemlidir. Çünkü kısıtlı radyo menzili nedeni ile bir duyarga düğümü tarafından taranabilecek alan büyüklüğü de sınırlıdır ve bu alanda yoğun şekilde bulundurulan düğümler gereksiz veri akışına neden olabilirler. İlgi altındaki alana gereğinden fazla duyarga düğümünün konuşlandırılması da ağın maliyetini artırır. Düğümlerin herhangi bir güvenlik mekanizması kullanılmadan TK protokolü ile organize olduğu durumda ağdaki duyarga düğüm derecesine bağlı olarak sahip olduğu genel bağlantı oranının değişiminin benzetim sonucu Şekil 3-2'de verilmiştir. Buna göre ortalama yerel bağlantı sayısı 7 olduğunda en büyük bağlı komponentin tüm düğüm sayısına oranı 0,98'in üstüne çıkmaktadır.




**Şekil 3-2:** TK (GKA-1) protokolünde yerel bağlantı sayısına bağlı olarak genel ağ bağlantı oranının değişimi.

### 3.2.2 Açık anahtar tabanlı anahtar belirleme

Bölüm 3.2.1'de bahsedilen ön dağıtım tabanlı anahtar belirleme protokolü hesaplama maliyetleri açısından oldukça tutumludur. Fakat ağın bağlantı oranını dengelemek için yapılan rastlantısal anahtar ön dağıtımları nedeni ile her duyarga düğümünün üzerinde belli bir miktar anahtar bulundurma zorunluluğunu getirmektedir. Bu ise anahtarların ele geçirilmesine yönelik saldırılarda ağın güvenli haberleşmeye devam edebilmesi özelliğini zayıflatmaktadır. Duyarga ağ

güvenliğinde devamlılığın en yüksek oranda sağlanabilmesi için dayanıklılığın en yüksek seviyede sağlanması gerekmektedir. Bunu yapabilmenin ise iki yolu vardır. İlk yol, duyarga ağı konuşlanmadan önce düğümlere tüm diğer düğümler ile paylaşacakları özgün ikili anahtarların yüklenmesidir. Bu işlem yüksek adetli duyarga ağlarında belki de duyarga düğümünün belleğinden daha fazla anahtarın yüklenmesini gerektirdiğinden tercih edilmez ve kullanım alanı yoktur. Diğer seçenek ise hesaplama maliyeti yüksek olan açık anahtar tabanlı asimetrik anahtar belirleme yöntemlerinin kullanılmasıdır. Literatürde bu yöntemlerin duyarga ağlarında kesin bir şekilde uygulanamaz olduğuna dair bir sonuca varılmamış olduğu görülmektedir (Walters et al., 2005; Wang et al., 2006; Xiao et al., 2007). Buna göre kullanılan duyarga düğümlerinin yapısı ve ağdan alınması planlanan hizmetin süresi de göz önünde bulundurularak ihtiyaç duyulan güvenlik seviyesi ve duyarga düğüm ömrü arasında bir ilişki kurularak bu yöntemin genel olarak uygulanabilirliği değerlendirilmiştir. Piotrowski ve ark. (2006), ECC açık anahtar şifreleme yönteminin bu kategorideki diğer yöntemlere göre duyarga düğümleri için daha uygun olduğunu ve MICA2 tip bir duyarga düğümünün 10 dakikada bir yapılan ECC 160 bit çarpım işlemini standart pil ömrü ile yaklaşık 5 yıl sürdürebilecek kapasitede olduğunu belirtmiştir. Bu durumda eğer ECC işlemlerinin sıklığı kontrol edilebilirse duyarga ağında kaynak kullanımının kabul edilebilir boyutlarda olacağı düşünülebilir. Duyarga düğümlerinin ele geçirilmesine karşı ağın dayanıklılığını simetrik yöntemlere göre ciddi oranda artırması nedeni ile de bu yöntemin kabul edilebilir bir bedel karşılığında duyarga ağlarında kullanılması mümkündür. Dolayısıyla asimetrik anahtar belirleme yöntemleri, sahip olduğu özellikler nedeni ile güvenliğin üst düzeyde önemli olduğu sistemlerde tercih edilebilir durumdadır.

## 3.3 TK Protokolü ile Anahtar Belirleme

Bu bölümde, TK protokolünün yapılanma performansını düşürmeden gerçekleştirilmesi gereken anahtar belirleme işlemlerinin sisteme getirdiği maliyetler incelenmiştir. Bu kapsamda, kümeleme algoritması ile tümleşik olarak çalışacak Temel Şema ve ECDH anahtar belirleme işlemlerinin bellek ve mesajlaşma maliyetleri analiz edilmiştir.

### 3.3.1 Temel Şema ile anahtar belirleme

Temel Şema'nın TK protokolü ile uygulanmasında duyarga düğümler alana yerleştirilmeden önce Temel Şema'da tanımlanmış olan anahtar ön dağıtımı işlemleri tüm düğümlere yapılmış durumdadır. TK protokolünün işleyişi sırasında da yapılanan düğümler tarafından komşuları belirlemek için yapılan yoklama mesajları ile anahtar belirleme protokolü devreye girmektedir.

Anahtar keşfi ve komşu belirleme aşamasında fiziksel olarak birbirleri ile haberleşme mesafesinde olan düğümlerin sayısı yapılacak mesajlaşma miktarını doğru orantılı şekilde etkilemektedir. Bu nedenle ağdaki fiziksel duyarga düğüm derecesinin mümkün olan en düşük seviyede tutulması toplam iletişim maliyetlerinin azaltılması için önemlidir. Kümeleme algoritması açısından bakıldığında ağın genel bağlantı oranının %99'dan büyük olmasını sağlayacak en düşük düğüm derecesinin sağlanması performans açısından yeterlidir. Çünkü benzetim sonuçlarına göre yoğunluk arttıkça oluşan küme sayısı düşük oranda azalmaktadır ve genel ağ bağlantı seviyesi sadece %0,1-0,3 oranında iyileşmektedir. Düşük düğüm derecesine sahip ağlarda anahtar dağıtımının kümeleme performansını etkilememesi için güvenli hat kurabilen düğümlerin ihtiyaç duyulan yerel bağlantı oranını sağlaması gerekmektedir. Bu da iki düğüm arasında güvenli hat oluşturma olasılığının yüksek olması demektir. Çünkü düşük olasılıkta elde edilecek yerel ağ bağlantı oranı genel bağlantı oranının yeterli düzeyde olmasına yetmeyecektir. Diğer taraftan, radyo ağlarında etkin iletişim için önerilen teorik üst sınır 8'dir (Hwang and Kim, 2004). Bu durumda ortalama komşu düğüm derecesi en fazla 7 olan bir ağda TK protokolü ile %99'dan büyük bir genel ağ bağlantı oranına sahip olabilmek için her düğümün sahip olduğu tüm komşuları ile ortak anahtar belirleyebiliyor olması gerekir. Bu da iki düğüm arasında sağlanması gereken güvenli hat bağlantı olasılığının 1'e yakın olması demektir ( $p_{required} \cong 1$). Bu olasılığın sağlanabilmesi için her duyarga düğümü tarafından bellekte taşınması gereken en az anahtar sayısı anahtar havuz boyutu $P$ = 10.000 için yaklaşık $k$ = 190'dir. Bunun yanında patika anahtar keşfi aşamasında kurulması olası güvenli hatların toplam güvenli hat bağlantı olasılığına etkisi incelendiğinde ekstra haberleşme maliyeti karşılığında aynı bağlantı olasılığı için daha az bellek kullanımın yeterli olduğu görülmektedir.

Hwang ve Kim (2004), rastsal anahtar ön dağıtımında ortak anahtar keşfinden sonra patika anahtarının belirlenmesinin iki şekilde yapılabileceğini belirtmiştir. Bunlardan *cascade-off counting* ile sadece ortak anahtar keşfi sırasında kurulan güvenli hatlar üzerinden patika anahtarı belirleme, *cascade-on counting* ile her yeni güvenli hat eklendiğinde patika anahtarını bu yeni hatları da kullanarak belirleme tanımlanmıştır. TK protokolünün çalışma yapısı dikkate alındığında ilk yoklama mesajına gelen yanıtlardaki anahtar listeleri ile yoklamayı yapan düğümün anahtar listesi arasında ortak anahtar kontrolü yapılarak ortak anahtar keşfi tamamlanır. Anahtar listesinin uzunluğuna göre bu aşamada güvenli hat kurulabilecek komşu sayısı belirlidir. Daha sonra eklenecek komşular için patika anahtarı belirlenmeye çalışılması gerekir. Patika anahtarının belirlenmesi için Liu ve Ning (2003a) tarafından önerilen yöntemde, yoklamayı yapan düğüm kendi listesini komşularına bütüneyayar ve ortak anahtar bulan komşusu kimlik numarası ile yanıt verir. Ortak anahtar bulamayan komşu ise kendi anahtar listesini yoklama yapan düğüme yollar. Yoklama yapan düğüm bu aldığı listeyi ortak anahtar paylaştığı düğümlere yollamak üzere bütüneyayar. Bu listeleri alan düğümler içinde listelerden en az biri ile ortak anahtara sahip olan her düğüm rastsal bir anahtar belirleyerek bunu yoklama yapan düğüm ve ortak anahtar paylaşılan diğer düğümün anahtarları ile ayrı ayrı şifreleyerek yoklama yapan düğüme bireyayar. Kaynak düğüm hedef düğümün anahtarı ile şifrelenmiş olan parçayı bu düğüme yollar ve bu şekilde ortak anahtar belirlenmiş olur. Bu yöntemde sadece tek zıplama komşuluğundaki düğümler üzerinden patika anahtarı belirlenmeye çalışılmaktadır.

Herhangi iki duyarga düğümünün anahtar listelerinde en az bir ortak anahtar bulunması olasılığı $p$ olarak tanımlandığında, fiziksel duyarga düğüm derecesi $d$ olan bir ağda herhangi bir kaynak düğümün ortak anahtar keşfi ile güvenli hat kurabileceği komşu sayısı $v_{shared\_key}$'in beklenen değeri aşağıdaki gibidir:

$$\mathrm{E}[v_{shared\_key}] = \sum_{i=1}^{d} p = dp \text{'dir.} \qquad (3.5)$$

$p$ olasılığı $P$ ve $k$ değerlerine bağlıdır ve bu ilişkinin denklemi Eşitlik 3.1'de verilmiştir. Kaynak düğüm *A*, anahtar listesinde ortak anahtar bulunmayan

komşu düğümleri ile patika anahtarı belirleme aşamasında güvenli hat kurmaya çalışır. Bu aşamada güvenli hat kurulan komşu sayısı $v_{path\_key}$'in beklenen değeri $E[v_{path\_key}]$ aynı zamanda patika anahtarı belirlemek için yapılacak mesajlaşma sayısı $M_{path\_key}$'in de beklenen değerini verir. Çünkü kaynak düğüm, kurulacak her yeni güvenli hat için ayrı bir patika anahtarı belirleme işlemi gerçekleştirir. $E[v_{path\_key}]$ değerini bulabilmek için patika anahtarı belirleme aşamasında, *A*'nın komşuluğundaki her düğümün güvenli link kurma olasılığı $p_{path\_key}$'in bulunması gereklidir. Herhangi bir düğümün, ortak anahtar keşfi aşamasında kaynak düğüme bağlanmama olasılığı (1-*p*) ve patika anahtarı belirleme aşamasında aynı düğümün kaynakla ortak anahtar paylaştığı en az bir komşu düğüm bulma olasılığı $1-(1-p^2)^d$ olduğunda[*];

$$p_{path\_key} = (1-p)(1-(1-p^2)^d) \tag{3.6}$$

Eşitlik 3.6 kullanılarak $E[v_{path\_key}]$ değeri aşağıdaki gibi hesaplanır;

$$E[v_{path\_key}] = \sum_{i=1}^{d} p_{path\_key} = d(1-p)(1-(1-p^2)^d) \tag{3.7}$$

$v_{path\_key}$, tüm anahtar kontrol operasyonları sonunda güvenli haberleşme hattı kurulan toplam komşu sayısı $v_{total}$'den $v_{shared\_key}$ sayısı çıkartılarak da bulunabilir. Bu durumda $v_{path\_key}$'in beklenen değeri aşağıdaki gibi hesaplanır;

$$E[v_{path\_key}] = E[v_{total} - v_{shared\_key}] = E[v_{total}] - E[v_{shared\_key}]$$

Bu denklemde $E[v_{total}]$, ortak anahtar belirleme ve patika anahtarı belirleme operasyonları sonunda bağlantı kurulan toplam komşu adedinin beklenen değeridir. Herhangi bir düğümün, ortak anahtar keşfi aşamasında kaynak düğüme bağlanmama olasılığı (1-*p*) ise bu düğümün komşuluğundaki hiçbir düğüme bağlanmama olasılığı $(1-p)(1-p^2)^d$ olur[*]. Buradan herhangi bir komşunun ortak anahtar keşfi ve patika anahtar keşfi sonunda kaynak düğüm ile güvenli link

[*] $(1-p^2)^d$, patika anahtarı belirleme aşamasında herhangi bir düğümün kaynak düğüm ile ortak anahtar paylaştığı en az bir komşu düğüm bulmama olasılığıdır.

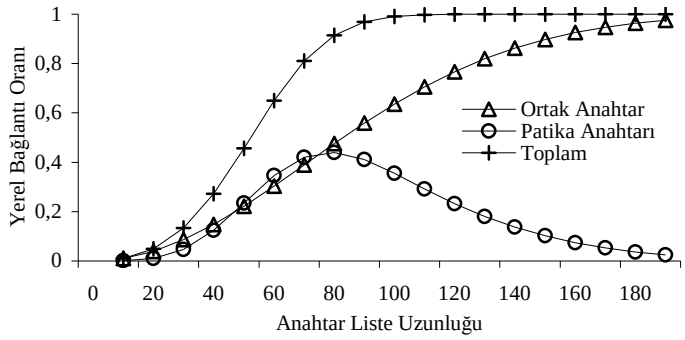
oluşturma olasılığı $p' = 1-(1-p)(1-p^2)^d$ olarak bulunur. Bu sonuca göre toplam komşu sayısının beklenen değeri aşağıdaki gibi hesaplanır;

$$\mathrm{E}[v_{total}] = \sum_{i=1}^{d} p' = d(1-(1-p)(1-p^2)^d) \tag{3.8}$$

Eşitlik 3.5 ve 3.8 kullanılarak $\mathrm{E}[v_{path\_key}]$ değeri aşağıdaki şekilde hesaplanır:

$$\mathrm{E}[v_{path\_key}] = \mathrm{E}[v_{total}] - \mathrm{E}[v_{shared\_key}] = d(1-p)(1-(1-p^2)^d) \tag{3.9}$$

Şekil 3-3'de $d$ = 8 ve $P$ = 10.000 olmak üzere farklı $k$ değerlerine göre elde edilen $p$ hat bağlantı olasılığına göre ortak anahtar (Eşitlik 3.5) ve patika anahtarı (Eşitlik 3.7) belirleme aşamalarında erişilebilecek yerel bağlantı oranları ve toplam hat oluşumu (Eşitlik 3.8) grafik olarak gösterilmiştir. Grafikten anlaşılacağı üzere $k = 90$ civarlarında patika anahtarı belirleme aşamasında güvenli hat oluşturulan komşu sayısı en yüksek seviyeye ulaşmıştır. Bu noktada düğümlerin yaklaşık yarısı ortak anahtar keşfi aşamasında diğer yarısı patika anahtarı belirleme aşamasında komşuluğa eklenmiştir. Diğer taraftan bir duyarga düğümünün sadece ortak anahtar keşfi ile tüm fiziksel komşularına güvenli olarak bağlanabilmesi için en az $k = 190$ uzunluğundaki bir anahtar listesinin her duyarga düğümünde saklanması gerekmektedir. Oysa, patika anahtarı keşfi de yapıldığında $k$ yaklaşık 90 iken güvenli hat kurulan düğümlerin fiziksel komşulara oranının 1'e çok yaklaştığı görülmektedir.

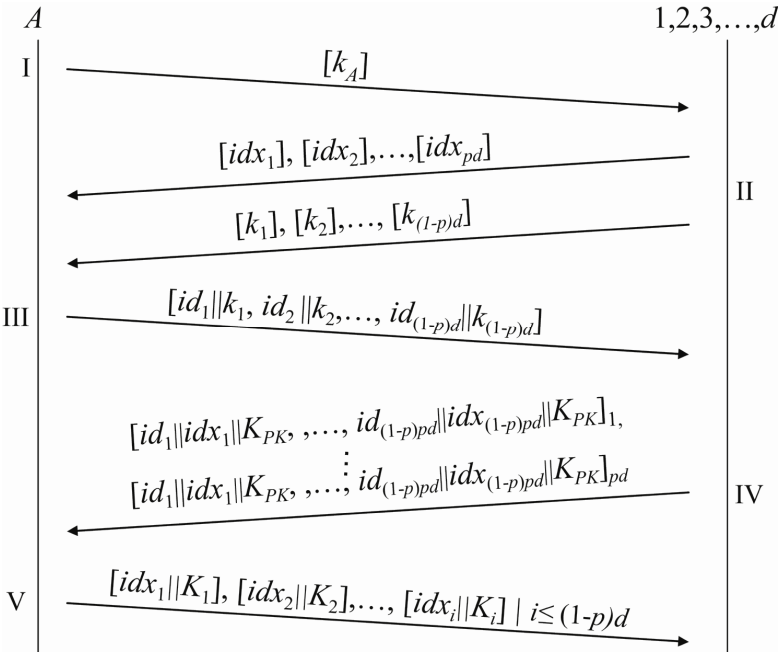



**Şekil 3-3:** Anahtar dağıtımında ortak anahtar keşfi ve patika anahtar keşfi sonucu erişilen yerel bağlantı oranı.

Duyarga düğümleri konuşlandıktan sonra ilk haberleşme ana istasyon ile en yakın duyarga düğümü arasında yapılır. Burada ana istasyon tüm anahtarlara sahip olduğu için hat anahtarı belirlenmeme olasılığı yoktur. Devam eden kurulumda Temel Şema'nın TK protokolü üzerine uygulanışı komşu keşfi aşamasında olmaktadır. Bu aşamada yapılan mesajlaşmalar Şekil 3-4'de aşama aşama ifade edilmiştir. Buna göre ağın yapılanması sırasında yapılanan herhangi bir duyarga düğümü *A* ilk olarak komşu keşfi için çevresine yoklama mesajı bütüneyayar. Bu mesaj *A*'nın anahtar listesinde bulunan anahtarların sıra numaralarını içeren liste $k_A$ 'yı da içerir (Şekil 3-4-I). Mesajı alan komşu düğümler $n_i$ ($i$ = 1,2,...,$d$), içeriğindeki $k_A$ listesi ile kendi anahtar listeleri $k_i$'yi ortak anahtar paylaşımının kontrolü amacı ile karşılaştırır. Eğer en az bir anahtar paylaşımı var ise ilgili komşu düğüm $n_i$ bu anahtarın sıra numarası $idx_i$'yi *A*'ya cevap olarak yollar (Şekil 3-4-II). Bu mesaj *A* ile *i*. duyarga düğümü arasındaki hat anahtarının belirlenmesini sağlar. Ortak anahtar keşfi aşamasında *A* ile hat anahtarı belirlenebilen komşu düğümlerin toplam sayısı *pd*'dir. Dolayısıyla *A*, (1-*p*)*d* adet fiziksel komşusu ile bu aşamada ortak anahtar belirleyememektedir. Bu durumda (1-*p*)*d* adet komşu düğüm, anahtar listesi $k_i$'yi ($i$ = 1,2,…,(1-*p*)*d*) patika anahtarı belirleme aşamasında kullanılmak üzere *A*'ya cevap olarak yollar (Şekil 3-4-II). Bütün cevapları alan *A*, anahtar paylaşma durumlarına göre tüm komşularının anahtar listelerini kayıt altında tutar. Bu anahtar listeleri, örneğin *idx* 2 byte olmak üzere, bellekte en fazla 2*k*(1-*p*)*d* byte yer tutar. *p* olasılığına bağlı olarak ilk aşamada ortak anahtar belirlenememiş düğüm var ise *A*, *pd* adet güvenli komşusu üzerinden patika anahtar belirleme işlemine başlar. *A* ilk olarak ortak anahtar keşfi aşamasında aldığı $k_i$ ($i$ = 1,2,…,(1-*p*)*d*) anahtar listelerini ilgili düğüm kimlik numaraları $id_i$ ile ilişkilendirerek bütüneyayar (Şekil 3-4-III). Bu mesajları alan *pd* adet güvenli komşu düğüm mesajdaki anahtar listeleri ile ortak anahtar kontrolü yapar ve ortak anahtara sahip olduğu her liste için patika anahtarı belirler. Belirlenen patika anahtarı *A* ve patika anahtarı bulunan liste ile paylaşılan anahtarlar ile şifrelenerek iki kopya şeklinde ($K_{pk}$) *A*'ya geri yollanır (Şekil 3-4-IV). Bu cevap mesajlarında her bir $n_i$ ($i$ = 1,2,…,(1-*p*)*d*) için ortak anahtar paylaşılan *pd* adet komşu tarafından belirlenmiş birden fazla $K_{pk}$ patika anahtarı bulunabilir. Bu durumda *A* bunlardan birini seçer ve ilgili patika anahtarı $K_i$ 'yi ilgili komşusuna liste sıra numarası $idx_i$ ile birlikte yollar ($i \leq$ (1-*p*)*d*) (Şekil 3-4-V). Bu aşamadan sonra eğer hala patika anahtarı belirlenememiş düğüm kalmış

ise *A* patika anahtarı belirleme işlemine devam eder. Bu fazda en son eklenmiş komşu(lar) da ortak anahtar paylaşılan komşu olarak değerlendirilir. Bu işlem patika anahtarı belirleme sırasında eklenecek hiçbir komşu kalmayana kadar devam eder.

Bu bölümün devamında Temel Şema ile ortak anahtar belirleme ve patika anahtarı belirleme işlemlerinin mesajlaşma ve hesaplama maliyet analizi yapılmıştır. Analizi basitleştirmek için patika anahtarlarının sadece ortak anahtar keşfi aşamasında güvenli hat kurulan düğümler üzerinden belirlendiği varsayılmıştır (*cascade-off counting*).




**Şekil 3-4:** Temel Şema'nın TK protokolüne uygulanışı.

### 3.3.1.1 Mesajlaşma ilişkisi

**Lemma 1:** İki düğüm arasında güvenli hat oluşturma olasılığı *p* olan ve toplamda *n* adet duyarga düğümü içeren bir ağda, ortak anahtar belirleme aşamasında her düğüm $v_i$ ($i$ = 1,2,3,…,$n$), bir adet bütüneyayım ve beklenen fiziksel komşu adedi, $E_{neighbor}$, kadar bireyayım yapar.

**İspat:** Başlangıçta ağı oluşturacak tüm duyarga düğümleri *floating* durumundadır ve protokol durum makinesine göre bir düğüm ancak yapılanma

mesajı aldığında durumunu değiştirerek komşu belirleme işlemine başlar. Diğer durumlarda yapılanma mesajı geçerli değildir ve herhangi bir işlem yapılmaz. Dolayısı ile tüm ağ düğümlerinin bağlandığı varsayıldığında herhangi bir düğüm $v_i$ yalnızca bir kez geçerli bir yapılanma mesajı alır ve yalnızca bir kez komşu belirleme için bütüneyayım yapar. Bunun ile bağlantılı olarak herhangi bir düğüm $v_i$, menzil olarak komşuluğunda olan her düğümden ($u_1, u_2, u_3, ..., u_{E_{neighbor}}$) komşu belirleme aşamasında yalnızca bir kez yoklama mesajı alır ve bu mesaja o anki durum bilgisi ve gerekirse anahtar sıra listesi ile yanıt verir. Bu yanıtın sayısı radyo menziline düşen komşuluktaki duyarga düğümü sayısı kadardır. Bu sayının beklenen değeri $E_{neighbor}$, duyarga düğülerinin yerleşiminde gösterdiği dağılım, toplam düğüm adedi, radyo menzili ve alanın boyutlarına bağlıdır. Buna göre komşu belirleme için düğüm başına gönderilen toplam mesaj adedi aşağıdaki gibidir.

$$M_{neighbor_{v_i}} = 1 + E_{neighbor} \quad \square$$

**Lemma 2:** İki düğüm arasında güvenli hat oluşturma olasılığı $p$ olan ve $n$ adet düğüm içeren bir ağda, ortak anahtar belirleme aşamasında her düğüm $v_i$ ($i$ = 1,2,3,…,$n$), $E_{neighbor}$ adet bütüneyayım ve $E_{neighbor}$ adet bireyayım mesajı alır.

**İspat:** Lemma 1'de belirtildiği gibi komşu belirleme aşamasında her düğüm $v_i$ bir adet bütüneyayım yapar. Bu mesajı alan komşuluk menzilindeki $E_{neighbor}$ adet düğüm ($u_1, u_2, u_3, ..., u_{E_{neighbor}}$) komşuluk bilgilerini bireyayım yolu ile yoklayan düğüme iletirler. Tüm mesajların eksiksiz alındığı varsayıldığında toplam $E_{neighbor}$ adet bireyayım mesajı $v_i$ tarafından alınmış olur. Diğer taraftan $v_i$, komşuları $u_1, u_2, u_3, ..., u_{E_{neighbor}}$ tarafından kendi komşularını belirlemek için yalnızca bir kez yapılan bütüneyayım mesajlarını da alır. Yine tüm mesajların eksiksiz alındığı varsayıldığında toplam $E_{neighbor}$ adet bütüneyayım mesajı $v_i$ tarafından alınmış olur. □

**Lemma 3:** İki düğüm arasında güvenli hat oluşturma olasılığı $p$ olan ve $n$ adet düğüm içeren bir ağda, komşu belirleme aşamasında her düğüm $v_i$ ($i$ =

1,2,3,…,*n*) tarafından yapılan toplam mesaj gönderim maliyeti en fazla aşağıdaki gibidir:

$$C_{\mathrm{M}_{tx}} = C_{tx}(L_{poll\_sk} + L_{poll\_pk} + \mathrm{E}_{neighbor}(pL_{poll\_reply\_s} + (1-p)(L_{poll\_reply} + p^2\mathrm{E}_{neighbor}L_{ask\_key} + L_{ask\_key}/2))) \quad (3.10)$$

($C_{tx}$ bit başına radyo gönderim maliyeti, $L_{poll\_sk}$ ortak anahtar keşfi aşamasında bütüneyayım, $L_{poll\_pk}$ patika anahtarı belirleme aşamasında bütüneyayım, $L_{poll\_reply\_s}$ ortak anahtar bulunduğuna dair cevap, $L_{poll\_reply}$ ortak anahtar bulunmadığına dair cevap ve $L_{ask\_key}$ patika anahtarı belirlemek için yapılan bireyayım mesaj uzunluklarını ifade etmektedir)

**İspat:** Lemma 1'de belirtildiği gibi her düğüm $v_i$ ortak anahtar belirleme aşamasında anahtar listesini içeren bir adet bütüneyayım yapmaktadır. Patika anahtarı belirleme aşamasında da bağlantı kurulamamış düğümlerin anahtar listelerinin yayınlandığı başka bir bütüneyayım daha yapılır. Bu bütüneyayım mesajlarının bit uzunlukları sırasıyla $L_{poll\_sk}$ ve $L_{poll\_pk}$'dir. Her $v_i$ düğümü komşuluğundaki $\mathrm{E}_{neighbor}$ adet düğümden aldığı bu bütüneyayım mesajlarına anahtar listelerinde ortak anahtar bulunma olasılığına göre yanıt verir. Buna göre $p\mathrm{E}_{neighbor}$ adet düğüme sadece ortak anahtar sıra numarasını, $(1-p)\mathrm{E}_{neighbor}$ adet düğüme de $k$ adetlik anahtar sıra listesini gönderir. Bu mesajların uzunlukları sırayla $L_{poll\_reply\_s}$ ve $L_{poll\_reply}$'dir. Ek olarak her $v_i$, patika anahtarı belirleme için kaynak düğümlerden aldığı bütüneyayımlardan en fazla $p\mathrm{E}_{neighbor}$ tanesine en fazla $p(1-p)\mathrm{E}_{neighbor}$ adet patika anahtarını bireyayım ile yollar. Her bir patika anahtar mesaj uzunluğu $L_{ask\_key}$ kadardır. En son patika anahtarı belirlenen komşulara şifreli anahtar mesajını yollama maliyeti $(1-p)\mathrm{E}_{neighbor}$ komşu için $L_{ask\_key}/2$ kadardır. Sonuç olarak toplam bütüneyayım mesaj uzunluğu $L_{poll\_sk} + L_{poll\_pk}$ ve toplam bireyayım mesaj uzunluğu aşağıdaki gibi olmak üzere;

$$= p\mathrm{E}_{neighbor} L_{poll\_reply\_s} + (1-p)\mathrm{E}_{neighbor} L_{poll\_reply} +$$
$$p^2(1-p)\mathrm{E}^2_{neighbor} L_{ask\_key} + (1-p)\mathrm{E}_{neighbor} L_{ask\_key} / 2$$
$$= \mathrm{E}_{neighbor}(pL_{poll\_reply\_s} + (1-p)(L_{poll\_reply} +$$
$$p^2\mathrm{E}_{neighbor} L_{ask\_key} + L_{ask\_key} / 2))$$

Bu eşitlik kullanılarak $C_{tx}$ bit başına radyo gönderim maliyeti olduğunda yapılan toplam mesaj gönderim maliyeti aşağıdaki gibi hesaplanır;

$$C_{\mathrm{M}_{tx}} = C_{tx}(L_{poll\_sk} + L_{poll\_pk} + \mathrm{E}_{neighbor}(pL_{poll\_reply\_s} + (1-p)(L_{poll\_reply} + p^2\mathrm{E}_{neighbor} L_{ask\_key} + L_{ask\_key} / 2))) \square$$

**Lemma 4:** İki düğüm arasında güvenli hat oluşturma olasılığı $p$ olan ve $n$ adet duyarga düğümü içeren bir ağda, komşu belirleme aşamasında her düğüm $v_i$ (i = 1,2,3,…,$n$) tarafından yapılan toplam mesaj alım maliyeti $C_{rx}$ bit başına radyo gönderim maliyeti olmak üzere en fazla aşağıdaki gibidir:

$$C_{\mathrm{M}_{rx}} = C_{rx}\mathrm{E}_{neighbor}((L_{poll\_sk} + L_{poll\_pk}) + pL_{poll\_reply\_s} + (1-p)(L_{poll\_reply} + p^2\mathrm{E}_{neighbor} L_{ask\_key} + L_{ask\_key} / 2)) \quad (3.11)$$

**İspat**: Ağdaki herhangi bir düğüm $v_i$ tüm komşularından Lemma 3'de belirtilen bütüneyayım mesajlarını alır. Ayrıca kendisinin yaptığı bütüneyayım mesajlarına karşılık olarak kendi komşularından da yanıt mesajları alır. İlk olarak ortak anahtar keşfi için yapılan bütüneyayım mesajına karşın komşularının $p\mathrm{E}_{neighbor}$ adedinden sadece ortak anahtar sıra numarasını ($L_{poll\_reply\_s}$), $(1-p)\mathrm{E}_{neighbor}$ adedinden de $k$ adetlik anahtar listesini ($L_{poll\_reply}$) alır. Patika anahtarı belirleme aşamasında $p\mathrm{E}_{neighbor}$ adet komşusundan en fazla $p(1-p)\mathrm{E}_{neighbor}$ adet komşusu için patika anahtarı ($L_{ask\_key}$) mesajı alabilir. Son olarak patika anahtarı ($L_{ask\_key} / 2$) kaynak düğümden alınır ve bu mesaj en fazla $(1-p)\mathrm{E}_{neighbor}$ adet komşusundan alınabilir. Sonuç olarak alınan toplam bütüneyayım mesaj uzunluğu $\mathrm{E}_{neighbor}(L_{poll\_sk} + L_{poll\_pk})$ ve toplam bireyayım mesaj uzunluğu aşağıdaki gibi olduğunda;

$$= \mathrm{E}_{neighbor} pL_{poll\_reply\_s} + \mathrm{E}_{neighbor}(1-p)L_{poll\_reply} +$$
$$p^2\,\mathrm{E}^2_{neighbor}(1-p)L_{ask\_key} + \mathrm{E}_{neighbor}(1-p)L_{ask\_key}/2$$
$$= \mathrm{E}_{neighbor}(pL_{poll\_reply\_s} +$$
$$(1-p)(L_{poll\_reply} + p^2\mathrm{E}_{neighbor}L_{ask\_key} + L_{ask\_key}/2))$$

$C_{rx}$, bit başına radyo alım maliyeti olmak üzere toplam alım maliyeti aşağıdaki gibi hesaplanır;

$$C_{\mathrm{M}_{rx}} = C_{rx}\mathrm{E}_{neighbor}((L_{poll\_sk} + L_{poll\_pk}) + pL_{poll\_reply\_s} +$$
$$(1-p)(L_{poll\_reply} + p^2\mathrm{E}_{neighbor}L_{ask\_key} + L_{ask\_key}/2)) \quad \square$$

Sonuç olarak herhangi bir düğüm $v_i$ için komşuları ile güvenli link oluşturma işleminin toplam mesajlaşma maliyeti aşağıdaki gibidir:

$$C_{\mathrm{M}_{neighbor}} = C_{M_{tx}} + C_{M_{rx}} \tag{3.12}$$

### 3.3.1.2 Hesaplama ilişkisi

Hesaplama maliyetleri düşünüldüğünde, herhangi bir düğüm $v_i$ patika anahtarı keşfi sırasında en fazla (1-$p$)$d$ adet patika anahtarı belirler. Bunun için kaynak kendisi olmak üzere yokladığı komşularından aldığı patika anahtarını elde edebilmek için (1-$p$)$d$ anahtar deşifre işlemi gerçekleştirir. Aynı düğüm bu sefer kendi komşuları kaynak olmak üzere patika anahtarı işlemi yaptığında $d$ adet komşusundan aldığı (1-$p$)$d$ adet anahtar listesinin her biri için ortak anahtar bulma olasılığı $p$ ve aynı zamanda kaynak düğümle de ortak anahtar paylaşma olasılığı $p$'ye göre $2p^2(1-p)d^2$ kez şifreleme işlemi gerçekleştirir. Sonuç hesaplama maliyeti aşağıdaki gibidir:

$$C_{computation} = (2p^2(1-p)d^2)_{enc} + ((1-p)d)_{dec} \tag{3.13}$$

### 3.3.1.3 Toplam enerji maliyeti

Temel Şema'nın enerji maliyetinin analizinde kullanılan ağ seviyesi mesaj yapısı aşağıdaki gibidir:

$$A \rightarrow B : ID_A, ID_B, M, MAC\{ID_A, ID_B, M\}$$

Mesaj bloğundaki *ID* kısmı duyarga düğümlerinin ağdaki kimlik numaralarıdır ve her bir düğüm için özgündür. Bu numaralar büyük ölçekli ağlar için (> 512) 16 bit olarak alınabilir. Mesaj bloğundaki diğer kısım *M* ise Temel Şema'nın gerektirdiği anahtar liste ve kimlik numaralarını içerir ve uzunluğu anahtar belirleme protokolünün aşamalarına göre değişkenlik göstermektedir. Burada MAC (*Message Authentication Code*) SHA-1 tabanlıdır ve mesaj bloğu içerisinde 160 bitlik yer kaplar. Referans alınan simetrik şifreleme ve deşifreleme protokolü ise AES 128'dir. AES 128'de kullanılan anahtar boyutu 128 bittir. Analizde anahtar havuz büyüklüğü $P$ = 10.000 olarak alınmıştır. Bu büyüklükteki bir havuz için anahtarların sahip olduğu sıra numaraları 2 byte ile ifade edilebilir. Değişken olarak düğümlere yüklenen anahtar listesi uzunluğu *k* seçilmiş ve analiz bu değişkene göre ağda yapılan toplam mesajlaşma ve hesaplama maliyetlerinin nasıl değiştiğini göstermeye çalışmıştır.

Bölüm 3.3.1.1'de mesajlaşma detayı verilen Temel Şema uygulamasında kaynak düğüm *A* tarafından ortak anahtar keşfi aşamasında komşularına yolladığı $M = L_{poll\_sk}$ mesajının içeriğinde *A*'da bulunan anahtarların sıra numaralarını içeren 2*k* byte'lık anahtar listesi bulunmaktadır. Cevap olarak bu listeyle ortak anahtar bulamayan düğüm kendi anahtar listesini içeren $M = L_{poll\_reply}$ mesajı gönderir ve bu mesajda da 2*k* byte bulunmaktadır. Eğer *A* ile paylaşılan ortak anahtar bulunuyorsa $M = L_{poll\_reply\_s}$ mesajı ile sadece paylaşılan ortak anahtarın sıra numarası gönderilir ki bu da sadece 2 byte'dır. Patika anahtarı belirleme aşamasındaki $M = L_{poll\_pk}$ yoklama mesajında ise *A* düğümünün ortak anahtar keşfi sırasında güvenli link kuramadığı düğümlerin *ID* numaraları (2 byte) ve anahtar listeleri ( $L_{poll\_reply} = L_{poll\_sk}$ ) bulunur. Bu düğümlerin sayısı ortak anahtar

keşfi aşamasında iki düğüm arasında ortak anahtar bulunma olasılığı *p*'ye ve ağ yoğunluğu *d*'ye bağlıdır.

$$M = L_{poll\_pk} = d(1-p)(L_{poll\_sk} + 2) \tag{3.14}$$

Patika anahtarını belirleyen bir komşu bu anahtarı kaynak düğümle paylaştığı anahtar ve bağlantı kurulacak diğer düğümle paylaştığı anahtar ile şifreleyerek kaynak düğüme yollar. Bu mesaj $M = L_{ask\_key}$ olup iki adet 128 bitlik (16 byte) anahtar, bağlantı kurulacak düğümün *ID* numarası (2 byte) ve anahtar sıra numarasını (2 byte) içerir. Patika anahtarını alan kaynak düğüm bu anahtarı paylaşmak üzere ilgili düğüme deşifreleme için kullanacağı anahtarın sıra numarası ile birlikte bir adet 128 bitlik anahtar yollar. Bu sonuçlara göre mesaj uzunlukları Çizelge 3-1'de gösterilmiştir.

**Çizelge 3-1:** Temel Şema mesaj uzunlukları.

| *Mesaj tipi* | *Mesaj uzunluğu (bit)* |
|---|---|
| $L_{poll\_sk}$ | $16k$ |
| $L_{poll\_reply}$ | $16k$ |
| $L_{poll\_reply\_s}$ | 16 |
| $L_{poll\_pk}$ | $d(1\text{-}p)\ (L_{poll\_sk} + 16)$ |
| $L_{ask\_key}$ | 2(16+128) |

Her bir mesaj gönderici ve alıcının 16 bitlik kimlik numaralarını ve 160 bitlik SHA-1 çıktısını da içerir. Buna göre Eşitlik 3.12'yi oluşturan Eşitlik 3.10 ve 3.11 Çizelge 3-1'deki değerler ile yeniden yazıldığında aşağıdaki sonuçlara ulaşılır:

$$\begin{aligned} C_{\mathrm{M}_{tx}} = C_{tx}(16k + 192 + d(1-p)(16k + 16) + 192 + d(p208 \\ + (1-p)(16k + 192 + p^2 d480 + 336))) \end{aligned} \tag{3.15}$$

$$\begin{aligned} C_{\mathrm{M}_{rx}} = C_{rx} d(16k + 192 + d(1-p)(16k + 16) + 192 + p208 + \\ (1-p)(16k + 192 + p^2 d480 + 336)) \end{aligned} \tag{3.16}$$

Piotrowski ve ark. (2006) çalışmasında Mica2Dot duyarga düğümü için verilen Rx/Tx radyo iletimi enerji maliyetleri Çizelge 3-2'de verilmiştir. Bu

çizelgedeki bit başına harcanan enerji (Ws = J (joule)) $C_{tx}$ ve $C_{rx}$ değerlerini tanımlamaktadır. Anahtar belirleme operasyonları sırasında yapılan hesaplama maliyetleri de Piotrowski ve ark. (2006) çalışmasında Mica2Dot düğümü için verilmiştir. Bu maliyetler Çizelge 3-3'de bulunmaktadır.

**Çizelge 3-2:** Mica2Dot için 868 MHz radyo iletim maliyetleri.

| *Radyo* | *Enerji* |
|---|---|
| Rx | 0.750 μJ/bit |
| Tx 5 dBm | 1.984 μJ/bit |

**Çizelge 3-3:** Mica2Dot için simetrik hesaplama maliyetleri.

| *Hesaplama türü* | *Enerji* |
|---|---|
| $AES128_{enc}$ | 1,62 μJ/byte |
| $AES128_{dec}$ | 2,49 μJ/byte |
| SHA-1 | 5,9 μJ/byte |

SHA-1 MAC hesaplamalarına tabi olan mesaj uzunlukları haberleşme maliyetlerinin hesaplandığı mesajlaşmalardan (Bkz. Eşitlik (3.15); Eşitlik (3.16)) 160 bitlik MAC uzunluğu çıkarılarak elde edilir. Buna göre SHA-1 hesaplama yapılan toplam mesaj uzunluğu düğüm başına aşağıdaki gibidir:

$$\begin{aligned} C_{M_{SHA-1}} = {} & C_{SHA-1}(16k + d(1-p)(16k+16) + d(p16 \\ & + (1-p)(16k+192 + p^2 d\,288 + 144))) \\ & + d(16k + d(1-p)(16k+16) + p16 \\ & + (1-p)(16k+192 + p^2 d\,288 + 144)) \end{aligned} \tag{3.17}$$

Düğüm başına AES-128 anahtar hesaplama işlemi ise patika anahtarı belirleme sırasında yapılan mesajlaşmalar sırasında olur. Buna göre bir düğüm $pd$ komşusundan $(1-p)d$ adet patika anahtarı talebi alır ve her bir komşusu için $p(1-p)d$ adet patika anahtarı belirleyerek 2 adet şifreleme işlemi gerçekleştirir. Yine her düğüm $(1-p)d$ adet komşusu için aldığı patika anahtarlarını deşifreler ve aynı patika anahtarını aynı zamanda komşusu da deşifreler. Bu işlemlerden her bir şifreleme deşifreleme 128 bitlik AES anahtarları için yapılır dolayısıyla toplam işlem sayısı 16 ile çarpılır (128 bit = 16 byte). Bu sonuçlar Eşitlik 3.13 üzerine uygulandığında toplam şifreleme ve deşifreleme maliyeti aşağıdaki gibi olur:

$$C_{computation} = (2p^2(1-p)d^2)16_{enc} + (2(1-p)d)16_{dec} \quad (3.18)$$

Toplam yapılanma enerji maliyeti ise mesaj iletimleri ve MAC/AES 128 hesapları sırasında harcanan enerjinin toplamına eşittir:

$$C_{configuration} = C_{\mathrm{M}_{neighbor}} + C_{M_{SHA-1}} + C_{computation} \quad (3.19)$$

### 3.3.2 ECDH ile anahtar belirleme

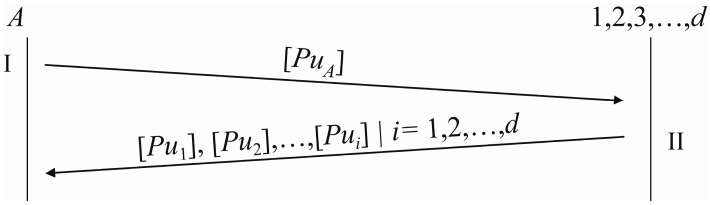
ECDH tabanlı asimetrik anahtar belirleme yönteminin kümeleme protokolünün ağ seviyesinde uygulanışı oldukça basittir ve kümeleme protokolüne ekstra bir mesajlaşma maliyeti getirmemektedir. Öncelikle alana bırakılmadan önce duyarga düğümlerine güvenlik için 160 bit ECDH protokolünde kullanılmak üzere belirlenen özel anahtar ve bu anahtar kullanılarak hesaplanmış açık anahtarlar yüklenir. Özel anahtar belirleme ve açık anahtar hesaplama işlemi duyarga düğümleri dışında daha güçlü hesaplama gücü ve enerji kaynağı olan güvenilir bir düğüm tarafından yapılmaktadır. Böylece ECDH için ihtiyaç duyulan hesaplamaların bir kısmı yeteneği, enerji ve bellek bakımından çok daha üstün olan bir sistem tarafından gerçekleştirilmiş olur ve duyarga düğümleri tarafından anahtar belirleme için sadece bir adet nokta çarpım işlemi yapılır.

Açık anahtar belirleme yöntemi ile ikili ortak anahtar belirlenmesinde herhangi bir düğüm *A*, ilk aşamada çevresindeki düğümleri, $u_1, u_2, u_3, ..., u_d$, bulmak için radyo menzilindeki tüm düğümlere açık anahtarını içeren bir yoklama mesajı atar (Şekil 3-5-I). Anahtar belirleme için yapılan mesajlaşmalarda *d* beklenen komşu sayısı, *P* özel anahtar ve *Pu* açık anahtar olmak üzere bu mesaj aşağıdaki gibidir.

$$A \rightarrow u_i : \{Pu_A\} \mid i = 1,2,3,...,d$$

Mesajı alan düğümler kendi açık anahtarlarını bireyayım yaparak yanıt verir (Şekil 3-5-II).

$$u_i \rightarrow A:\{Pu_i\},\ |\ i=1,2,3,...,d$$




**Şekil 3-5:** ECDH yönteminin TK protokolüne uygulanışı.

*A* yoklama cevabını aldıktan sonra komşuları ile paylaşacağı oturum anahtarını ECDH yöntemine göre hesaplar. Böylece bu düğümler arasındaki iletişimin şifrelenmesinde kullanılacak ortak ikili hat anahtarlar $K_{link_i}$ aşağıdaki gibi hesaplanır;

$$K_{link_i} = P_A Pu_i \quad | \quad i=1,2,3,...,d$$
$$K_{link_i} = P_i Pu_A \quad | \quad i=1,2,3,...,d$$

ECDH ile ortak anahtar belirlenirken düğüm kimliklemesi yapılmadığı durumlarda yetkisi olmayan herhangi bir saldırgan düğümün ECC açık parametrelerini kullanarak ağa bağlantı yapması ve iki düğüm arasındaki güvenli iletişimin arasına girerek *mitm* (*man in the middle*) saldırısı yapması olasılığı bulunmaktadır. Saldırgan açısından anlamlı olabilmesi için bu saldırının ortak anahtarların belirlendiği yapılanma zamanında gerçekleştirilmesi gerekmektedir. Çünkü yapılanma tamamlandığında duyarga düğümleri arasında ikili oturum anahtarı belirleme işlemleri tamamlanmıştır ve bu düğümler birbirleri ile paylaştıkları ortak anahtarlar üzerinden güvenli haberleşmektedirler. Bu durumda *mitm* saldırısı düğüm ele geçirme saldırısı yapılarak ikili ortak anahtar ele geçirilmeden gerçekleştirilemez.

RSA ve ECDSA imza protokolleri açık anahtar belirleme için tanımlanmış temel kimlikleme protokolleridir. Bu protokollerde duyarga düğümlerinin sahip olduğu açık anahtarlar güvenilir bir üçüncü yapının, CA (*Certificate Authority*), özel anahtarı ile imzalanır ve imza kopyası yapılanmadan önce ilgili düğümlere yüklenir. Burada imza hazırlanması işleminin duyarga düğümleri tarafından değil

hesaplama kabiliyeti yüksek ve enerji kısıdı bulunmayan güvenilir bir düğüm tarafından gerçekleştirilmesi nedeni ile bu aşamadaki enerji maliyetleri yapılanma enerji maliyetlerine dahil değildir. Yapılanma sırasındaki anahtar belirleme aşamasında ise açık anahtarlarını karşılıklı paylaşan düğümler aynı zamanda imzayı da paylaşır ve düğümlerde bulunan CA açık anahtarı kullanılarak bu imzanın doğruluğu kontrol edilir.

Çizelge 3-4'de Mica2Dot duyarga düğümü üzerinde yapılan asimetrik anahtar işlemlerinin maliyetleri verilmiştir (Piotrowski et al., 2006). Bu maliyetler incelendiğinde RSA imza oluşturma işleminin ECDSA yöntemine göre 15 kat daha yüksek olduğu buna karşın onay maliyetlerinde 1/4 oranında daha düşük olduğu görülmektedir. İmza oluşturma işlemlerinin yapılanmadan önce kaynak problemi olmayan güvenli bir düğüm tarafından gerçekleştirildiği düşünüldüğünde, yapılanmadan sonra anahtarlar belirlenirken RSA onay yöntemi ile kimlikleme yapılmasının duyarga düğümleri için en uygun yöntem olduğu sonucuna varılabilir. Asimetrik anahtar belirlemede ise RSA sunucu maliyetleri ECDH sunucu maliyetlerine göre 15 kat daha fazladır. Bu durumda duyarga ağlarında ikili anahtarların belirlenmesi için ECDH protokolü enerji ve bellek kullanımı açısından en doğru seçenektir. Çünkü ağ yapılanma zamanında ikili ortak anahtarlar belirlenirken istemci ve sunucu işlemlerinin her ikisi de duyarga düğümleri tarafından yapılmaktadır.

**Çizelge 3-4:** Mica2Dot için asimetrik işlem maliyetleri.

| *İmza* | *Oluşturma* | *Onaylama* |
|---|---|---|
| RSA 1024 | 304mJ | 11,9mJ |
| ECDSA 160 | 22,82mJ | 45,09mJ |
| *Anahtar Belirleme* | *İstemci* | *Sunucu* |
| RSA 1024 | 15,04mJ | 304mJ |
| ECDH 160 | 22,3mJ | 22,3mJ |

Bu sonuçlara göre ECDH ve RSA kullanılarak yapılanma zamanında kimliklemeli ortak anahtar belirlemenin hesaplama enerjisi kullanımı açısından en uygun seçim olduğu görülmektedir. Bu yöntemlerin duyarga düğümlerinde birlikte kullanımının getirdiği maliyetler bellekte tutulan yer miktarı, iletimde kullanılan mesaj uzunluğu ve hesaplamalarda kullanılan enerji miktarıdır. Yapılanma sırasında sadece imza onay işlemleri yapıldığı için bu işlemlerde CA

açık anahtarı ve imzalı mesaj (ortak anahtar belirlenecek düğümün imzalı açık anahtarı) kullanılır. Bu durumda RSA imza yönteminin sadece açık anahtar işlemlerinin duyarga düğümlerinde uygulanmış olması yeterlidir. Gura ve ark. (2004), RSA açık anahtar işlemlerinin 8 bitlik bir mikro kontrolcünün kod belleğinde 1 Kbyte yer kapladığını belirtmiştir. Aynı çalışmada belirtildiği üzere 160 bitlik ECC anahtar belirleme işlemlerini gerçekleştiren kodun boyutu aynı platform için 3,68 Kbyte'dır. Piotrowski ve ark. (2006), RSA imzasının mesaj uzunluğunun en fazla anahtar uzunluğu kadar olabileceğini belirtmiştir. Dolayısıyla 1024 bit'lik RSA işleminde imza uzunluğu 1024 bit olarak alınabilir. ECDSA yönteminde ise imza uzunluğu anahtar uzunluğunun iki katıdır. Yani 160 bit ECDH için imza uzunluğu 320 bit'dir. Bu çıkarımlara göre imza yöntemi ile kimliklemenin duyarga ağı yapılanmasına olan maliyeti karşılaştırmalı olarak benzetim çalışmaları bölümünde incelenmiştir.

#### 3.3.2.1 Mesajlaşma ilişkisi

Asimetrik anahtar belirleme aşamasında yapılan mesajlaşma adedi simetrik anahtar belirlemenin ortak anahtar keşfi aşamasındaki kadardır. Çünkü asimetrik yöntemde $p$ = 1 olasılığında iki düğüm ortak anahtar belirleyebilir. Yapılan bütün işlem, düğümlerin komşularını belirlemesi ve açık anahtarların paylaşılmasıdır. Bu durumda Lemma1 ve Lemma 2'de belirtilen mesajlaşma karmaşıklığı burada da geçerlidir. Buna göre her bir düğüm $v_i$, anahtar belirleme aşamasında en fazla bir adet bütüneyayım ($L_{poll}$) ve $\mathrm{E}_{neighbor}$ adet bireyayım ($L_{poll\_reply}$) mesajı gönderir. Aynı şekilde komşularından $\mathrm{E}_{neighbor}$ adet bütüneyayım ($L_{poll}$) ve $\mathrm{E}_{neighbor}$ adet bireyayım ($L_{poll\_reply}$) mesajı alır. Mesaj paketlerindeki güvenlik için kullanılan kısmın uzunluğu bütüneyayım ve bireyayım mesajları için aynıdır. Dolayısı ile toplam mesajlaşma maliyeti aşağıdaki gibi hesaplanır;

$$C_{\mathrm{M}_{neighbor}} = C_{tx}(L_{poll} + \mathrm{E}_{neighbor} L_{poll\_reply}) + C_{rx}\mathrm{E}_{neighbor}(L_{poll} + L_{poll\_reply}) \tag{3.20}$$

Bu denklemde $L_{poll} = L_{poll\_reply}$ olup ECDH anahtar uzunluğuna (160 bit) eşittir. Dolayısı ile mesajlaşma karmaşıklığı komşu sayısı $\mathrm{E}_{neighbor}$'a bağlı olarak

değişir. ECDH yöntemi kullanıldığında komşu olan herhangi iki düğüm arasında güvenli link oluşma olasılığı $p$ = 1 olduğu için ağın yapılanma performansı hiç güvenlik yapılmadığı durumdaki gibidir.

### 3.3.2.2 Hesaplama ilişkisi

ECDH ile ikili anahtarları belirleme için her düğüm $v_i$ tarafından yapılan çarpma işlem sayısı o düğümün komşularının adedi $E_{neighbor}$ kadardır. $E_{neighbor}$ değerinin artırılması duyarga düğümü başına düşen komşu sayısını artıracağı için yapılan hesaplama miktarını da artıracaktır. Bu değerin azaltılması ise ağda küme oluşumunu olumsuz yönde etkileyip dengesiz küme oluşumunu tetikleyecektir. Bu nedenle $E_{neighbor}$ değerinin duyarga düğümlerinin kaynaklarına ve ağın yapılanma gereksinimlerine uygun seçilmesi gerekmektedir.

### 3.3.2.3 Toplam enerji maliyeti

ECDH enerji maliyetinin analizinde de aşağıdaki mesaj yapısı kullanılmıştır.

$$A \rightarrow B : ID_A, ID_B, M, MAC\{ID_A, ID_B, M\}$$

ECDH ile anahtar belirlemede komşu belirleme için yapılan yoklama mesajı $L_{poll}$ içerisinde kaynak düğümün 160 bitlik açık anahtarı bulunmaktadır. Duyarga düğüm fiziksel yoğunluğu $d$ olduğunda yanıt veren $d$ adet komşu da mesajlarında $L_{poll\_reply}$ 160 bitlik açık anahtarlarını bulundururlar. ECDH için gönderilen ve alınan mesaj toplamı Bölüm 3.3.1.1'de verilen Lemma 1 ve Lemma 2 sonuçları ile aynıdır. Dolayısıyla mesajlaşma maliyeti sadece duyarga düğümü fiziksel yoğunluğu $d$'ye bağlı olarak değişir. Haberleşen düğümlerin kimlik numaraları ve 160 bitlik MAC değeri de eklendiğinde bir düğümün komşuları ile anahtar oluşturmak için yapacağı toplam mesajlaşma miktarı Eşitlik 3.20 yeniden yazıldığında aşağıdaki gibidir:

$$\begin{aligned} C_{\mathrm{M}_{neighbor}} &= C_{tx}(160+192)(1+d) + C_{rx}(160+192)(2d) \\ &= C_{tx}\,352(1+d) + C_{rx}\,704d \end{aligned} \tag{3.21}$$

Düğüm başına hesaplama maliyetleri düşünüldüğünde ECDH anahtar belirleme protokolünde özel anahtar ve açık anahtar çiftleri ana istasyon tarafından hesaplanmış ve duyarga düğümlerine yüklenmiştir. Dolayısı ile ikili ortak anahtarı belirlemek için duyarga düğümlerine sadece bir nokta çarpım işlemi yapmak kalır. Bu işlemin enerji maliyeti Mica2Dot için 160 bit ECC kullanıldığında alıcı ve verici tarafında düğüm başına 22,30 mJ'dür (Bkz. Çizelge 3-4). Toplam fiziksel komşu adedi *d* kadar bu hesabın yapılması gerektiğinden düğüm başına toplam anahtar hesaplama maliyeti aşağıdaki gibidir:

$$C_{computation} = d\text{x}22{,}30\text{mJ} \tag{3.22}$$

MAC hesabı yapılan toplam mesaj uzunluğu ise yine alınan ve verilen toplam mesaj uzunluğuna bağlı olarak hesaplanır. Buna göre SHA-1 işlemine tabi tutulan toplam mesaj uzunluğu aşağıdaki gibidir:

$$\begin{aligned} C_{M_{SHA-1}} &= C_{SHA-1}(160+32)(1+d)+(160+32)(2d) \\ &= C_{SHA-1}192+576d \end{aligned} \tag{3.23}$$

Toplam yapılanma enerji maliyeti ise mesaj iletimleri ve MAC/ECDH hesapları sırasında harcanan enerjinin toplamına eşittir. Bu eşitlik tüm düğümler ile ortak anahtar belirlenmeye çalışıldığı durum için (*proaktif* hat seçimi) geçerlidir:

$$C_{configuration} = C_{\mathrm{M}_{neighbor}} + C_{M_{SHA-1}} + C_{computation} \tag{3.24}$$

İmza yöntemi kullanıldığında ise iletimdeki mesaj yapısına 160 bit'lik açık anahtar yanında ECDSA için 320 ve RSA için 1024 bit'lik imza mesajı eklentisi yapılır. MAC hesabı da bu değişikliğe göre güncellenir. Hesaplama maliyetleri ise anahtar hesaplama ile imza onay maliyetlerinin toplamını içerir.

### 3.3.3 Çapraz seviye tasarımı

Anahtar belirleme protokolleri ile TK protokolünün karşılıklı etkileşimi ilk aşamada ağdaki fiziksel duyarga düğüm derecesi ve güvenli link oluşturma

olasılığına bağlı yerel bağlantı oranının değişimine göre ağın genel bağlantı oranının değişimi ile oluşmaktadır. Buna karşın TK protokolünde durum kontrolü ile çalışan yapılanma organizasyonunda bağlantı kurulması gereken düğümler kontrol edilebilir durumdadır. Yani herhangi bir düğüm yoklama sırasında kaydettiği komşuları içerisinden sadece *floating* olanları kullanarak yapılanmayı devam ettirmektedir. Diğer durumlardaki komşu duyarga düğümleri ya karar işlemleri için ya da sadece bilgi amaçlı kaydedilir. İlgili komşularla güvenli hat oluşturma için anahtar belirleme operasyonları dikkate alındığında ise yapılanma açısından ciddi bir katkısı olmayan düğümler ile güvenli hat oluşturulması sisteme fazladan maliyet getirmektedir. Bunu engelleyebilmek için TK protokolündeki durum kontrolü ile anahtar belirlenecek komşu düğümlerin kontrollü olarak seçilmesi mümkündür.

Tez çalışmasında, Temel Şema'nın TK protokolü ile uygulanmasında güvenli hat kurulacak komşu düğümlerinin seçimi için üç yöntem tanımlanmıştır. Bunlardan ilki olan *proaktif* yöntemde ortak anahtar keşfi ve patika anahtarı keşfi aşamaları işletilerek fiziksel komşuluktaki tüm duyarga düğümleri ile güvenli hat kurulmaya çalışılır. Bu yöntemde patika anahtarlarının belirlenmesi *cascade-on counting* ile yapılır. İkinci yöntem olan *reaktif* seçimde, ortak anahtar belirleme aşamasında sadece *floating* durumunda olan düğümler ile güvenli hat kurulmaya çalışılır. Bu işlem sonrasında hala ortak anahtar belirlenememiş *floating* düğümler kalmış ise komşuluğa eklenen düğümler üzerinden patika anahtarları belirlenmeye çalışılır (*cascade-off counting*). Son yöntem olan *düz* seçimde ise sadece ortak anahtar keşfi aşamasında güvenli hat kurulan düğümler ile yapılanma devam ettirilir. Anahtar belirleme işlemleri sırasında aynı zamanda komşu keşfi de yapıldığından Çizelge 2-1'de verilen TK algoritmasında komşu keşfinin yapıldığı 9-11, 21- 23, 32-34 numaralı satırlar yerine Çizelge 3-5'de verilen anahtar belirleme algoritması çalıştırılır.

Temel Şema'dan farklı olarak ECDH tabanlı anahtar belirlemenin güvenli hat oluşturma olasılığı 1 olduğu için TK protokolü ile uygulamasında *düz* yönteminin uygulanması söz konusu değildir. *Reaktif* ve *proaktif* yöntemlerinin ECDH için uygulanışı ise oldukça basittir. *Proaktif* yöntemde fiziksel komşulukta olup henüz ortak anahtar belirlenmemiş tüm duyarga düğümleri ile güvenli hat

kurulmaya çalışılır. Buna göre kaynak düğümün komşularını belirlemek için bütüneyaydığı yoklama mesajına fiziksel komşuluktaki tüm duyarga düğümleri açık anahtarları ile cevap verir. *Reaktif* yöntemde ise sadece *floating* durumunda olup henüz ortak anahtar belirlememiş düğümler yoklama mesajına cevap verir. Bu basit algoritma Çizelge 3-6'da verilmiştir.

**Çizelge 3-5:** Temel Şema için çapraz seviye anahtar belirleme algoritması.

**Assumptions:** Network is $G = (V;E)$, there is no node failure during the configuration

```
BEGIN:
    i ← 0; j ← 0; new_neighbor_added ← 0
    BROADCAST poll with k_v for neighbor discovery and start neighbor discovery timer
    RECEIVE poll_reply from u
    if shared key reply then
        ADD u∈ V to neighbor list as a shared key neighbor
        i ← i + 1
    else
        ADD key ring k_u and state of u∈ V to neighbor list
        j ← j + 1
    endif
    WAIT neighbor discovery timer out
    if straight_method then
       goto END
    else if reactive_method then
       if there is no floating neighbor in j neighbors then
          goto END
       else
          BROADCAST key rings k of only floating unsecured neighbors for path key discovery
    else if proactive_method then
       BROADCAST key rings k of all unsecured neighbors for path key discovery
    endif
PATH KEY DISCOVERY:
    RECEIVE path keys K_PK from each u_1,2,..,i
    for selected j neighbors u_j∈ V of v do
       if one K_j exists in K_PK then
          SEND K_j to u_j as the shared key
          ADD u_j to neighbor list as a shared key neighbor
          i ← i + 1
          j ← j - 1
          new_neighbor_added ← 1
       end if
    end for
    if proactive_method and j > 0 and new_neighbor_added then
       BROADCAST key rings k of all unsecured neighbors for path key discovery
       new_neighbor_added ← 0
       goto PATH KEY DISCOVERY
    end if
END:
```

**Çizelge 3-6:** ECDH için çapraz seviye anahtar belirleme algoritması.

**Assumptions:** Network is $G = (V;E)$, there is no node failure during the configuration

```
BEGIN:
    BROADCAST poll with Pu_v for neighbor discovery and start neighbor discovery timer
    /*proactive: all neighbors reply; reactive: only floating neighbors reply*/
    RECEIVE poll_reply and Pu_u from u
    K_uv ← Pu_u P_u
END:
```

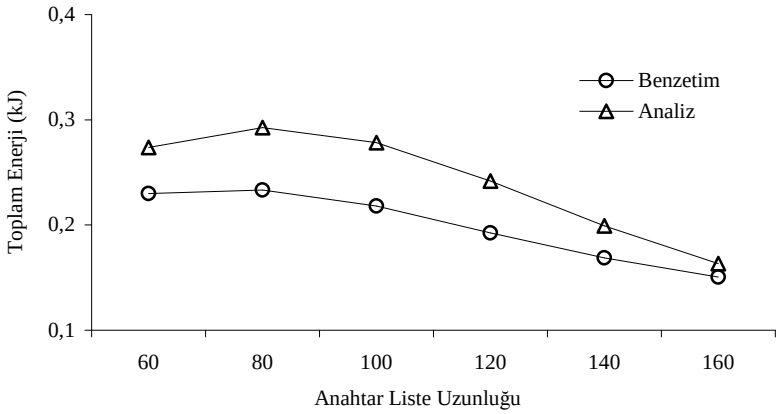
Bu yöntemlerin sistem organizasyonunu nasıl etkilediği benzetim yolu ile analiz edilmiş ve sonuçları yorumlanmıştır. Bir sonraki bölümde bu sonuçlar verilmiştir.

### 3.3.4 Anahtar belirleme benzetim sonuçları

Bu bölüm kapsamında yapılan benzetimlerin ana amacı ağın yapılanmaya başlamasından itibaren bitişine kadar anahtar belirileme işlemleri sırasında alınan ve gönderilen toplam bit sayısını ve yapılan toplam hesaplama miktarını ölçmektir. Benzetimler sonucu elde edilen sonuçlar Çizelge 3-2, Çizelge 3-3 ve Çizelge 3-4'de verilen düğüm enerji maliyetleri ile çarpılarak yapılanma sırasında mesajlaşma ve hesaplama için ihtiyaç duyulan toplam enerji maliyetleri hesaplanmıştır. Benzetim sonuçları alınırken analizlerde kullanılan mesaj yapısı aynen kullanılmıştır. Benzetimlerde TK (GKA-1) protokolü kullanılmış (Bu bölümde kısaca TK kısaltması ile isimlendirilmiştir) ve küme eleman sayısı sınırı 32 olarak belirlenmiştir. Dayanıklılık sonuçları belirlenirken a) Ağda ele geçirildiği varsayılan bir düğüm ile düğüme bağlı olan iletim hatları ve o düğümde kayıtlı olan anahtarları kullanan diğer hatlar ele geçirilmektedir b) Temel Şema'da, şifrelenmiş patika anahtarlarının kablosuz iletimi nedeni ile bu anahtarları şifreleyen ortak anahtarlardan birinin ele geçirilmesi durumunda ilgili patika anahtarı da ele geçirilmektedir. İlk olarak Şekil 3-6'dan Şekil 3-16'ya kadar Temel Şema'nın TK protokolü ile çapraz uygulanması sonucu ortaya çıkan performans sonuçları verilmiştir.

Temel Şema'nın TK protokolü ile Bölüm 3.3.1'de belirtildiği gibi uygulanması sonucu toplam enerji harcamasındaki değişimin farklı anahtar uzunluklarına göre analiz ve benzetim sonuçları Şekil 3-6'da verilmiştir. Bu sonuçlar için ağ yapılanma parametreleri sırasıyla ağ eleman sayısı $n$ = 2000, fiziksel duyarga düğüm derecesi $d$ = 8 ve anahtar havuzu büyüklüğü $P$ = 10.000 olarak alınmıştır. Bu yapılanma parametreleri ile "*idx*" anahtar sıra numaraları ve "*ID*" düğüm kimlik numaralarının mesaj içerisindeki uzunlukları 16 bit olarak alınmıştır. Bölüm 3.3.1'de yapılan toplam enerji harcama analizinde her bir düğümün komşuluğundaki tüm düğümler ile *cascade off counting* yöntemi ile anahtar belirleme işlemi yaptığı varsayıldığı için Şekil 3-6'daki analiz sonucu bu

şartlarda oluşabilecek en kötü durumu vermektedir. Benzetimde ise herhangi iki düğüm arasında sadece bir kere anahtar belirleme işlemi yapıldığı için enerji maliyetindeki değişim en kötü durum grafiğine benzer bir karakteristik izlemekle birlikte büyüklüğü daha az olmaktadır.

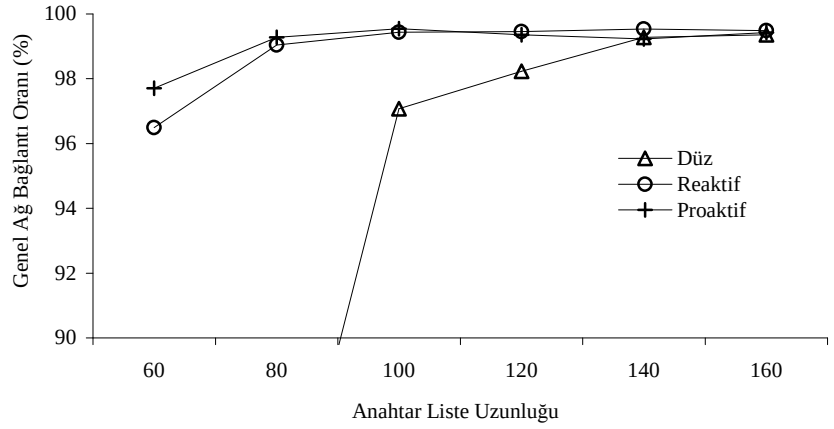



**Şekil 3-6:** Temel Şema enerji maliyetinin analiz ve benzetim sonuçları.

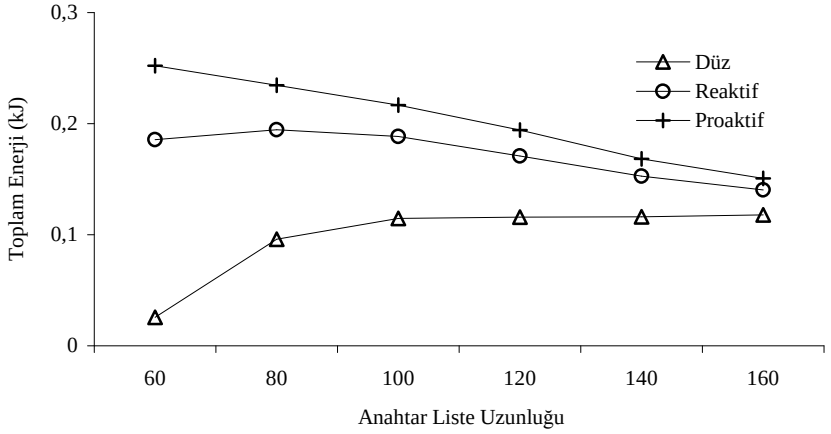
Şekil 3-7, Şekil 3-8, Şekil 3-9'da ise aynı ağ yapılanma parametreleri ile yapılanan TK protokolünde anahtar belirleme işlemleri için önerilen *düz*, *reaktif* ve *proaktif* yöntemlerinin performans sonuçları verilmiştir. Şekil 3-7'de verilen sonuçlara göre güvenli hat oluşturulacak düğümler *düz* yöntemi ile belirlendiğinde %99 genel ağ bağlantı oranının geçilebilmesi için her bir düğümde $k$ = 160 adet anahtar bulundurulması gerekmektedir. Diğer taraftan *proaktif* ve *reaktif* seçimde genel ağ bağlantı oranının %99'dan büyük olabilmesi için yaklaşık 100 adet anahtar içeren bir listenin duyarga düğümlerine yüklenmesi yeterlidir. Bu anahtar liste uzunluğu Şekil 3-3'de verilen analiz sonucunda yaklaşık 90 olarak hesaplanmıştı. Aradaki bu fark kablosuz iletimdeki kısıtlardan dolayı kümeleme sırasında kaybedilen mesajlardan dolayı kaybedilen linklerden kaynaklanmış olabilir.

Şekil 3-8'de görüldüğü üzere *düz* yöntemi ile anahtar belirlenen yapılanmada k = 160 iken harcanan toplam enerji miktarı *reaktif* ve *proaktif* yöntemlere göre en düşüktür. Çünkü patika anahtarı belirleme işlemi *düz* yönteminde bulunmamaktadır. Fakat bu durumda anahtar listesini saklayabilmek için her duyarga düğüm belleğinde 160x128 bit = 20.480 bit = 2,5 Kbyte yer ayrılması gerekmektedir. Ek olarak düğümlere yüklenen anahtar listelerinin uzun

olması ağın dayanıklılığını düşürmektedir. Çünkü ele geçirilen bir düğüm nedeni ile açığa çıkacak olan anahtar sayısı da fazla olur ve bu nedenle açığa çıkacak hatların sayısı artar. Aynı genel ağ bağlantı oranı için daha kısa bir anahtar liste uzunluğu gerektiren *reaktif* ve *proaktif* yöntemlerde ise patika anahtarı belirleme işleminden dolayı enerji maliyetleri daha fazladır. Bu iki yöntemde de yaklaşık aynı dayanıklılık ve genel ağ bağlantı oranını veren $k$ = 100 için *reaktif* yapılanma sonucu harcanan toplam enerji miktarı *proaktif* yönteminkine göre %13 daha azdır. $k$ = 100 olması durumunda anahtar belirleme operasyonları için kullanılan toplam bellek miktarı ise anahtar listesi kaydı için 100x128 bit = 1,6 Kbyte ve patika anahtarı belirleme aşamasında kaydedilecek anahtar sıra listelerinin kaydı için $2k(1-p)d$ = 100(1-0,63)x8x2 byte = 592 byte olmak üzere toplamda yaklaşık 2,2 Kbyte'dır.

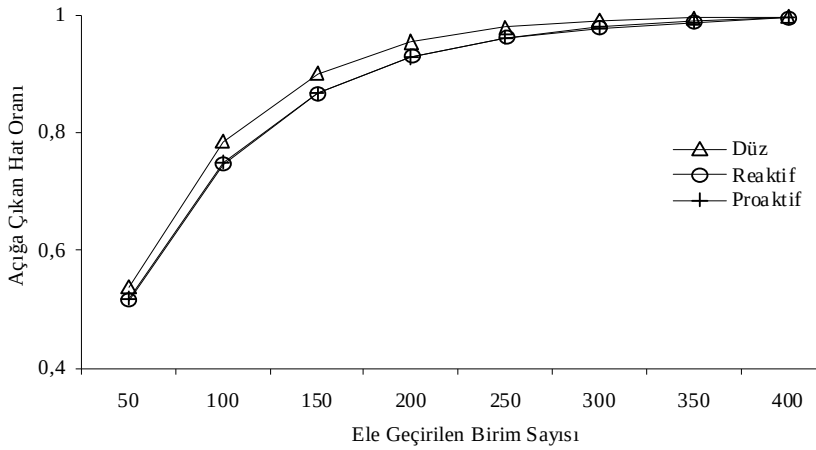



**Şekil 3-7:** $P$ = 10.000 için farklı anahtar liste uzunluklarında genel ağ bağlantı oranı değişimi.




**Şekil 3-8:** $P$ = 10.000 için farklı anahtar liste uzunluklarında ağ yapılanması için harcanan toplam enerji değişimi.

Şekil 3-9'da verilen dayanıklılık karakteristikleri genel olarak değerlendirildiğinde *reaktif* ve *proaktif* yöntemleri *düz* yöntemine göre daha iyi olmasına rağmen her üç yöntemde de dayanıklılığın çok düşük olduğu görülmektedir. Örneğin 100 adet duyarga düğümünün anahtarları ile birlikte ele geçirildiği durumda anahtarı açığa çıkan güvenli hatların ağda oluşturulmuş tüm güvenli hatlara oranı *reaktif* ve *proaktif* yapılanmalarda %75, *düz* yapılanmada %78,5 olmaktadır. Bu dayanıklılık sonuçlarının alındığı anahtar liste uzunluğu değerleri *reaktif* ve *proaktif* hat seçim yöntemleri için 100, *düz* yöntemi için 160'dır.

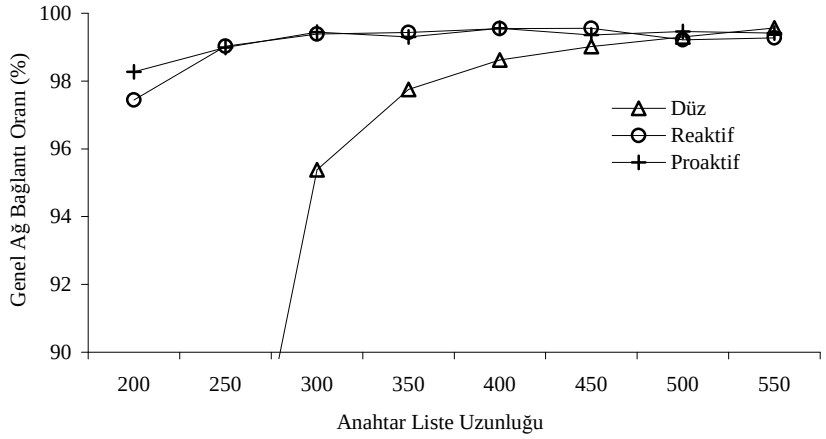



**Şekil 3-9:** $P$ = 10.000 için farklı anahtar liste uzunluklarında dayanıklılık değişimi.

$P$ = 10.000 iken Temel Şema ile erişilebilen dayanıklılık performansının artırılabilmesi için aynı ağ yapılanması bu kez $P$ = 100.000 için benzetime tabi tutulmuştur. Bu kısımda Şekil 3-10'dan Şekil 3-16'ya kadar verilen benzetim sonuçlarının alındığı ağ yapılanma parametreleri sırasıyla toplam ağ eleman sayısı $n$ = 2000, duyarga düğüm derecesi $d$ = 8 olarak alınmıştır. Bu yapılanma parametreleri ile *idx* anahtar sıra numaraları 17 bit, *ID* düğüm kimlik numaraları da 16 bit olarak alınmıştır.

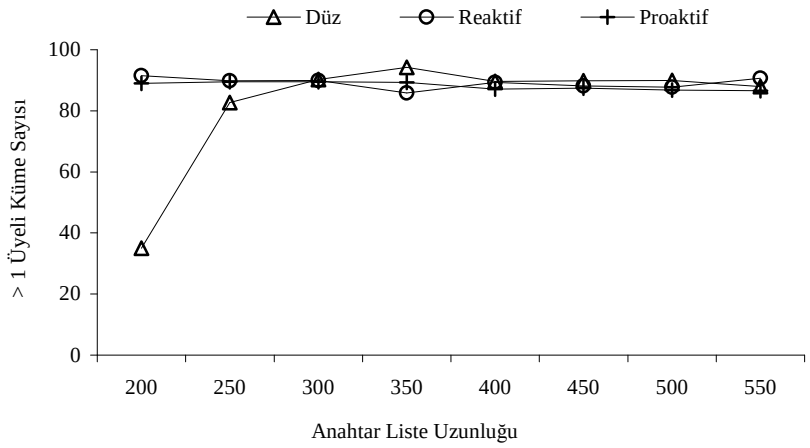
Şekil 3-10'da verilen sonuçlara göre güvenli hat oluşturulacak düğümler *düz* yöntemi ile belirlendiğinde %99 genel ağ bağlantı oranı için bu sefer her bir düğümde $k$ = 500 adet anahtar bulundurulması gerekmektedir. Fakat, *proaktif* ve *reaktif* seçimde genel ağ bağlantı oranının %99 olabilmesi için yaklaşık 300 adet anahtar içeren bir listenin duyarga düğümlerine yüklenmesi yeterlidir. Dolayısıyla

ağın genel bağlantı oranını sabit tutmak sureti ile anahtar havuz büyüklüğü *P*'yi artırmak her bir duyarga düğümüne yüklenmesi gereken anahtar sayısını da artırmaktadır. Bu artış özellikle patika anahtarı belirleme aşamasında yayınlanan anahtar listeleri nedeni ile toplam mesajlaşma maliyetini de artırmaktadır.



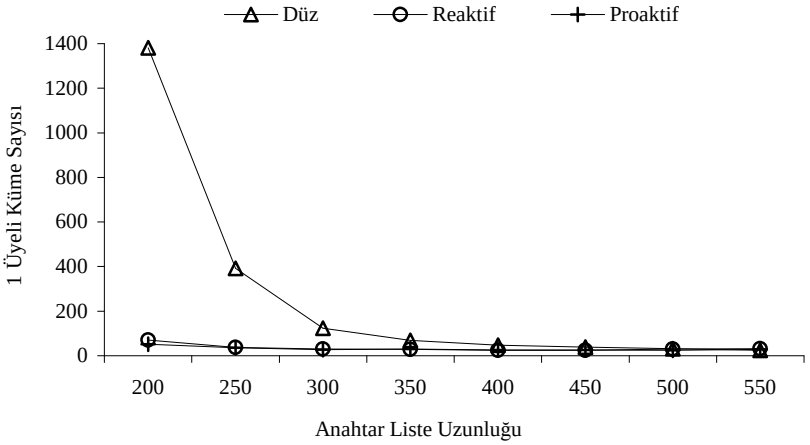

**Şekil 3-10:** *P* = 100.000 için farklı anahtar liste uzunluklarında genel ağ bağlantı oranı değişimi.

Şekil 3-11'den Şekil 3-14'e kadar her üç hat belirleme yöntemi için ağ yapılanması sonucu elde edilen kümeleme performanslarının benzetim sonuçları verilmiştir. Şekil 3-11'de verilen sonuçlara göre *reaktif* ve *proaktif* seçim ile yapılanan ağlarda eleman sayısı 1'den büyük toplam küme sayıları tüm anahtar uzunlukları için eşit düzeyde olmaktadır. *Düz* seçim sonucu ise ancak anahtar liste uzunluğu 300 olduğunda bu değerler ile aynı seviyede olmaktadır.



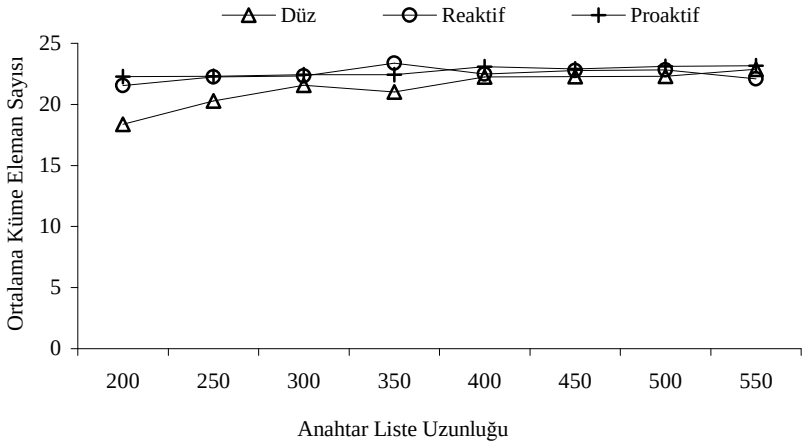

**Şekil 3-11:** *P* = 100.000 için eleman sayısı 1'den büyük toplam küme sayısındaki değişim.

Şekil 3-12'de *reaktif* ve *proaktif* hat seçimi sonucu oluşan tek elemanlı ayrık kümelerin toplam sayıları birbirine yakındır fakat *düz* seçim sonucu oluşan ayrık düğümlerin toplamı anahtar liste uzunluğu ancak 500 civarında olduğunda diğerlerinin seviyesine inebilmektedir. Dolayısıyla *düz* yöntem ile elde edilen genel bağlantı oranı ancak anahtar liste uzunluğu 500 civarında olduğunda *reaktif* ve *proaktif* hat seçim performansına ulaşmaktadır.



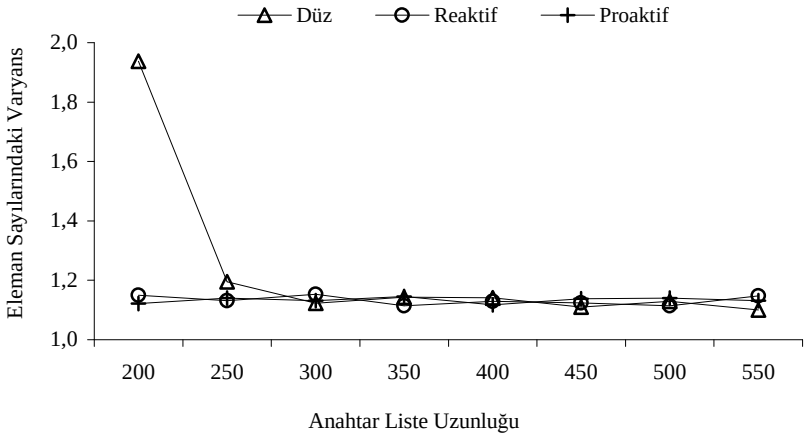

**Şekil 3-12:** *P* = 100.000 için eleman sayısı 1'e eşit toplam küme sayısındaki değişim.

Eleman sayısı 1'den büyük ortalama küme eleman sayısı Şekil 3-13'de verilmiştir. Bu sonuçlara göre *düz* hat seçimi yapıldığında anahtar listesi uzunluğu 400 iken ortalama küme eleman sayısı *reaktif* ve *proaktif* performansına yaklaşmaktadır. *Reaktif* ve *proaktif* hat seçim yöntemlerinde ise küme eleman sayısı farklı anahtar uzunluklarında benzer performans göstermektedir.




**Şekil 3-13:** *P* = 100.000 için eleman sayısı 1'den fazla kümelerin ortalama eleman sayısı.

Şekil 3-14, oluşturulan kümelerin büyüklüklerinin varyans değerlerini vermektedir. Bu sonuçlara göre *düz* hat seçimi yapıldığında anahtar listesi uzunluğu 350 küme eleman sayıları arasındaki varyans değerini *reaktif* ve *proaktif* performansına yaklaşmaktadır. *Reaktif* ve *proaktif* hat seçim yöntemlerinde ise varyans farklı anahtar uzunluklarında benzer performans göstermektedir.

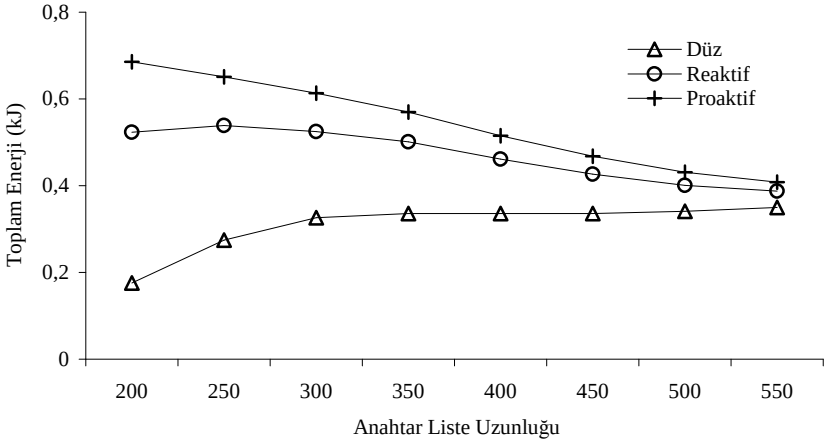



**Şekil 3-14:** *P* = 100.000 için eleman sayısı 1'den fazla kümelerin büyüklük varyansı.

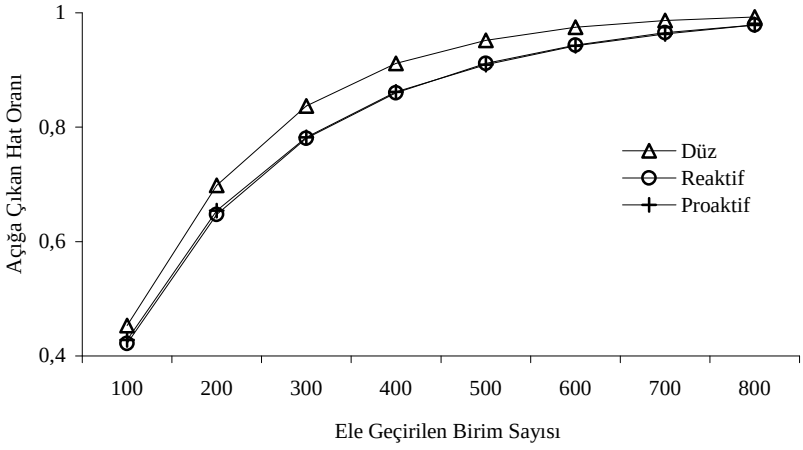
Şekil 3-15'de *P* = 100.000 için verilen toplam enerji maliyeti Şekil 3-8'de *P* = 10.000 için verilen toplam enerji maliyetlerini yaklaşık üçe katlamıştır. Şekil 3-15'de görüldüğü gibi *düz* yöntemi ile anahtar belirlenen yapılanmada %99 genel bağlantı oranını veren *k* = 500 için harcanan toplam enerji miktarı *reaktif* ve *proaktif* yöntemlere göre en düşüktür fakat anahtar listesini saklayabilmek için her duyarga düğüm belleğinde 500x128 bit = 64 Kbit = 8Kbyte yer ayrılması gerekmektedir. *Reaktif* ve *proaktif* yöntemlerde ise yaklaşık aynı dayanıklılık ve genel ağ bağlantı oranını veren *k* = 300 için *reaktif* yapılanma sonucu harcanan toplam enerji miktarı *proaktif* yönteminkine göre %15 daha azdır. *k* = 300 olması durumunda anahtar belirleme operasyonları için kullanılan toplam bellek miktarı ise anahtar listesi için 300x128 = 38,4 Kbit = 4,8 Kbyte ve patika anahtarı belirleme aşamasında kaydedilecek anahtar sıra listelerinin kaydı için $2k(1\text{-}p)d$ = 300x(1-0,59)x8x17 bit = 16,728 Kbit = 2,1 Kbyte olmak üzere toplamda yaklaşık 7 Kbyte'dır.

*P* = 100.000 için erişilen dayanıklılık seviyeleri ise Şekil 3-16'da verilmiştir. Bu sonuçların alındığı anahtar liste uzunluğu değerleri %99 genel ağ bağlantı oranını veren *reaktif* ve *proaktif* hat seçim yöntemleri için 300, *düz*

yöntemi için 500'dür. $P$ = 10.000 seçildiğinde *reaktif* ve *proaktif* hat seçim yöntemleri ile yapılanan ağda %75 oranında hattın açığa çıkması için 100 adet düğümün ele geçirilmesi yeterli iken $P$ = 100.000 olması durumunda *reaktif* ve *proaktif* yapılanmalarda %75 oranında hattın açığa çıkabilmesi için 300 adet duyarga düğümünün ele geçirilmesi gerekmektedir. Dolayısıyla $P$ = 100.000 olduğunda dayanıklılık yaklaşık üç kat artmıştır. *Düz* yapılanmada ise aynı adette duyarga düğümünün ele geçirilmesi durumunda %81 oranında hat açığa çıkmaktadır.




**Şekil 3-15:** $P$ = 100.000 için farklı anahtar liste uzunluklarında ağ yapılanması için harcanan toplam enerji değişimi.




**Şekil 3-16:** $P$ = 100.000 için farklı anahtar liste uzunluklarında dayanıklılık değişimi.

Analiz ve benzetim sonuçları değerlendirildiğinde Temel Şema'nın harcadığı toplam enerji çoğunlukla mesajlaşma ve iletilen mesajlar için yapılan SHA-1 hesapları tarafından kullanılmaktadır. Temel Şema'da dayanıklılığın

artırılması için anahtar havuzu *P*'nin büyüklüğünün daha da artırılması mümkündür fakat bu durumda arzu edilen genel ağ bağlantı oranına ulaşabilmek için daha fazla anahtarın düğüm belleğinde tutulması gerekmektedir. Bu da daha uzun anahtar listesi iletimi nedeni ile toplamdaki enerji maliyetlerini artıracaktır.

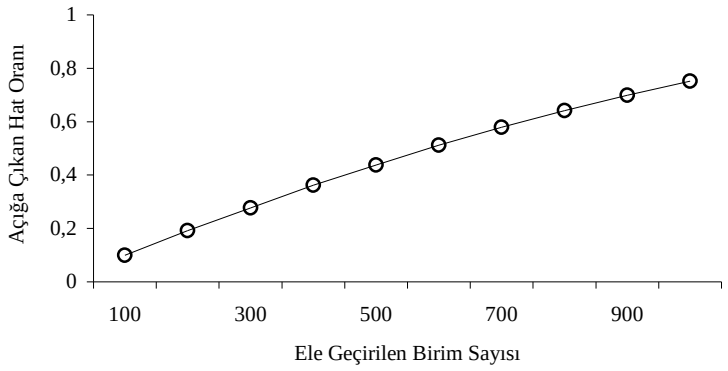
Çizelge 3-7 ve Şekil 3-17'de ECDH tabanlı anahtar belirlemenin kümeleme protokolü ile çapraz seviye uygulanması sonucu elde edilen benzetim sonuçları bulunmaktadır. Bu benzetimde ağ yapılanma parametreleri sırasıyla toplam ağ eleman sayısı $n$ = 2000, fiziksel duyarga düğüm derecesi $d$ = 8 ve *ID* numaraları 16 bit olarak alınmıştır. *Reaktif* ve *proaktif* yöntemlerinin kullanıldığı Çizelge 3-7'de verilen yapılanma benzetim sonuçlarına göre her iki durumda da elde edilen toplam ağ bağlantı oranı ve kümeleme performans sonuçları birbirine çok yakın olmak ile birlikte *reaktif* yapılanma sonucu harcanan toplam enerji miktarı *proaktif* yapılanmaya göre %5 oranında daha azdır. *Reaktif* yöntemdeki bu azalma genişlemek için seçilen komşu adedindeki düşüş ve buna bağlı olarak hesaplanan ECDH çarpım adetlerindeki düşüşten kaynaklanmaktadır.

**Çizelge 3-7:** ECDH ile anahtar belirleme benzetim sonuçları.

| *Parametreler* | *Proaktif* | *Reaktif* | *Fark (%)* |
|---|---|---|---|
| Genel bağlantı | % 99,34 | % 99,25 | 0,1 |
| Tek üyeli küme | 29,2 | 29,8 | -2,1 |
| Çok üyeli küme | 93,2 | 91,87 | 1,4 |
| Ortalama küme | 21,5 | 21,8 | -1,4 |
| Varyans | 1,167 | 1,173 | -0,5 |
| Enerji (Benzetim) | 0,42 kJ | 0,40 kJ | 5 |
| Enerji (Analiz) | 0,43 kJ | --- | --- |

Toplam enerji maliyetleri karşılaştırıldığında ise %99 genel ağ bağlantı oranı için ECDH ile yapılanma sırasında harcanan enerji miktarı Temel Şema'da aynı ağ yapılanma özelliklerini sağlayan ($P$ = 100.000 ve $k$ = 300) *reaktif* ve *proaktif* yapılanmaya göre %20–%30 daha azdır. Ayrıca Şekil 3-17'de sonuçları verilen ECDH dayanıklılık seviyesi Temel Şema'da elde edilenden yaklaşık üç kat daha iyidir. Şöyle ki ECDH ile yapılanan ağda %75 oranında güvenli hattın ele geçirilebilmesi için 1000 adet düğümün ele geçirilmesi gerekmektedir. Temel

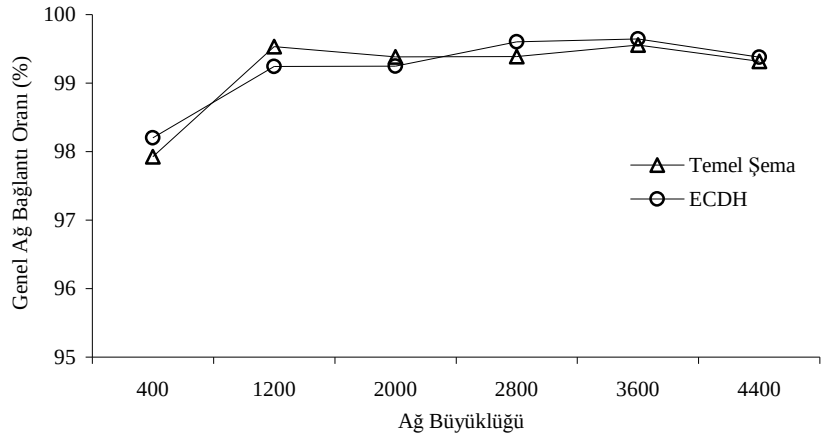
Şema'da ise $P$ = 100.000 ve $k$ = 300 olduğunda aynı oranda hattın ele geçirilebilmesi için 300 adet düğümün ele geçirilmesi yeterlidir.



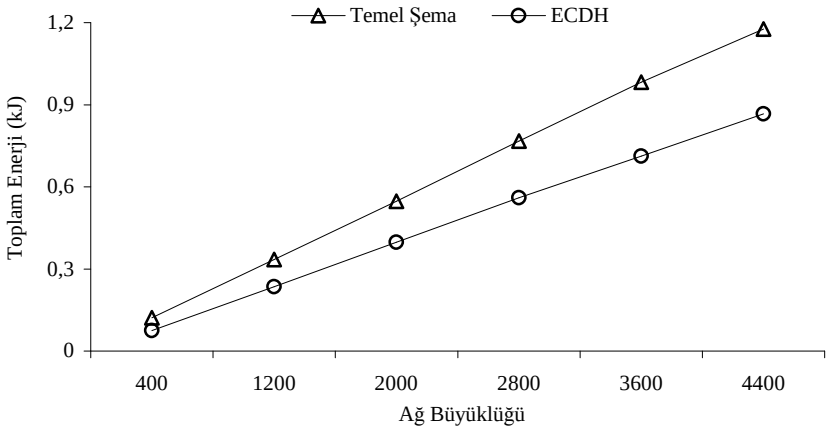

**Şekil 3-17:** *Reaktif* yapılanmada ECDH dayanıklılık performansı.

Şekil 3-18'de *reaktif* hat seçimi uygulayan Temel Şema ve ECDH anahtar belirleme uygulamalarında artan ağ eleman sayısına bağlı olarak genel ağ bağlantı oranındaki değişimin benzetim sonuçları verilmiştir. Bu benzetimde Temel Şema için ağ havuzu büyüklüğü $P$ = 100.000 ve anahtar liste uzunluğu $k$ = 300 olarak seçilmiştir. Ağ yoğunluğu önceki benzetimlerde seçilen ile aynıdır. Bu sonuçlara göre her iki anahtar belirleme protokolündeki değişim küçük farklar dışında benzeşmektedir. Eleman sayısı 400 iken genel ağ bağlantı oranının %99'un altına indiği görülmektedir. 1200 ve daha fazla eleman sayısına sahip bir ağ için bu oran daima %99'un üstündedir. 400 eleman sayılı bir ağda genel bağlantı oranının göreceli olarak düşük çıkmasının sebebi duyarga düğümlerinin yerleştiği alanın küçüklüğünden dolayı sınır koşulların daha belirleyici olmasındandır. Sonuç olarak artan ağ eleman sayısı ile anahtar belirleme protokol uygulamaları ile elde edilen genel ağ bağlantı oranının değişmediği söylenebilir.

Şekil 3-19'da artan ağ eleman sayısına bağlı olarak her iki protokol uygulamasındaki toplam enerji maliyetlerindeki değişimin benzetim sonuçları verilmiştir. Bu değişim her iki protokol için de doğrusal olarak artan bir karakteristik göstermektedir. Fakat Temel Şema'nın doğrusal artış hızı ECDH uygulamasındakinden %30 daha fazladır. Dolayısı ile ECDH çapraz seviye uygulaması Temel Şema'dan daha iyi performans göstermektedir.




**Şekil 3-18:** Anahtar belirleyerek yapılanan TK protokolünde artan ağ büyüklüğüne göre genel ağ bağlantı oranı değişimi.




**Şekil 3-19:** Anahtar belirleyerek yapılanan TK protokolünde artan ağ büyüklüğüne göre toplam enerji değişimi.

Son olarak ECDH ile anahtar belirlemede RSA ve ECDSA imza yöntemlerinin kullanımı ile yapılanma sonucu enerji maliyetlerindeki değişim Çizelge 3-8'de verilmiştir. Bu sonuçlara göre ECDH ve RSA protokollerinin birlikte kullanımı ile yapılanma sonucunda harcanan toplam enerji miktarı ECDH ve ECDSA protokollerinin birlikte kullanım maliyetinden yaklaşık %45 oranında daha azdır. Bunun sebebi RSA imza protokolündeki asimetrik hesaplama maliyetlerinin istemci lehine, ECDSA protokolünde ise istemci aleyhine olmasıdır. Bu imza işlemlerinde sunucu tarafından yapılan hesaplamaların yapılanmadan önce enerji problemi olmayan güvenilir bir düğüm tarafından belirlenmesi nedeni ile sadece istemci tarafındaki onay işlemleri açısından RSA ile daha az enerji harcanmaktadır. Burada toplam enerji maliyetleri

karşılaştırıldığında, RSA imza yöntemi ile kimliklenen *reaktif* ECDH anahtar belirleme işlemi için harcanan toplam enerjinin, $P$ = 100.000 ve $k$ = 300 için *reaktif* hat seçimine göre anahtar belirleyen Temel Şema uygulamasının toplam enerji maliyetinden yaklaşık %27 daha fazla olduğu görülmektedir. ECDSA ile kimlikleme yapılması durumunda ise harcanan toplam enerji aynı yapılanmadaki Temel Şema maliyetinin yaklaşık iki katıdır.

**Çizelge 3-8:** ECDH ve imza onayı için harcanan toplam enerji maliyetleri.

| *Hat Seçimi* | *ECDH + $RSA_{1024}$* | *ECDH + $ECDSA_{160}$* |
|---|---|---|
| Proaktif | 0,71kJ | 1,25 kJ |
| Reaktif | 0,66kJ | 1,18 kJ |

### 3.3.5 Anahtar belirlendikten sonra güvenlik

TK protokolünde ortak anahtarlar belirlendikten sonra yapılacak ikili kimlikleme, gizlilik, bütünlük ve tazelik operasyonları gibi güvenlik işlemleri için SPINS (Perrig et al., 2002) protokolü referans alınmıştır. SPINS protokolündeki ikili haberleşmede sağlanan güvenlik operasyonları, iki düğüm arasında bir ortak anahtar varsaymaktadır. Bu ortak anahtarın nasıl oluşturulacağı SPINS protokolü kapsamında değildir. Perrig ve ark. (2002), haberleşen iki düğüm A ve B'nin, $\chi_{AB} = \chi_{BA}$ gizli anahtarını paylaştıklarını varsayar. Her iki düğüm bu ortak anahtarı *pseudo-random* bir fonksiyonda kullanarak ortak şifreleme anahtarı $K_{AB} = K_{BA}$'yı türetirler. Kimlikleme ve veri doğruluğunun elde edilebilmesi için iletilen mesajların içinde MAC bulunmaktadır. MAC için kullanılan anahtar $K'_{AB}$, $K_{AB}$'den türetilir. Mesajların tazeliği mesajlaşan düğümler arasında özel olan bir sayıcı (*counter*) değerinin kontrolü ile yapılır. Bu yöntemde her mesajlaşmada sayıcı değeri artırılarak içeriği aynı olan iki mesajın şifrelenmiş çıktılarının birbirinden farklı olmaları sağlanır.

$$A \rightarrow B : ID_A, ID_B, \{counter_A, M\}_{K_{AB}}, MAC\left(ID_A, ID_B, \{counter_A, M\}_{K_{AB}}, K'_{AB}\right)$$

TK protokolünde her küme oluşumu tamamlandığında ilgili küme başı kriptografik bir sayı, $K_{cluster}$, üretir ve bu anahtarı ikili ortak anahtarlar kullanılarak

küme içi bağlantı ağacını kullanarak güvenli şekilde dağıtır. Bu anahtar belirli periyotlarla güncellenir ve yine ikili ortak anahtarlar kullanılarak kümeye dağıtılır. Dağıtımdan sonra küme içi haberleşmede mesaj şifrelemesi için sadece $K_{cluster}$ kullanılır. İkili kimlikleme için $K'_{AB}$ kullanılmaya devam eder.

SPINS tümeyayım güvenliği için $\mu Tesla$ protokolünü kullanmaktadır (Perrig et al., 2002). Fakat yüksek adetli ağlarda $\mu Tesla$ protokol eş zamanlamasının tutturulması problemlidir (Liu and Ning 2003b). Ek olarak ana istasyonun ağdaki tüm duyarga düğümlerine ulaşabildiği varsayılmıştır. Fakat duyarga düğümlerinin bulunduğu fiziksel coğrafyanın özelliklerine göre bu mümkün olmayabilir. TK protokolünde ise bütüneyayım çoklu zıplama ile yapılmaktadır bu nedenle $\mu Tesla$ protokolü uygulanamaz.

#### 3.3.5.1 Kümeler arası haberleşme

Kümeler oluşturulduktan sonra kümeler arası haberleşme küme başları üzerinden yapılır. Bunun için küme başları arasında çok zıplamalı iletim hattı kurulur. Bu hat aynı zamanda ana istasyonu da içerir. Bu hattın oluşumu Bölüm 2.4'de tanımlanmıştır.

Küme başları arası haberleşmede bir küme başından diğerine yollanan mesaj o küme içinde $K_{cluster}$ ile şifreli olarak taşınır. İki küme arasındaki bağlantıyı sağlayan düğümler arasındaki iletimde kimlikleme ise ikili haberleşmede kullanılan anahtarlar ile yapılır.

Mesaj iletiminde aynı kümeden iki düğüm arasında yapılan mesajlaşma aşağıdaki gibidir:

$$A \rightarrow B : ID_A, ID_B, \{ID_{next\_cluster}, M\}_{K_{cluster}},$$
$$MAC\left(ID_A, ID_B, \{ID_{next\_cluster}, M\}_{K_{cluster}}, K'_{A,B}\right)$$

Farklı kümelere ait iki düğüm arasında yapılan mesajlaşma ise aşağıdaki gibidir:

$$A \rightarrow B : ID_A, ID_B, \left\{ ID_{next\_cluster}, M \right\}_{K_{A,B}},$$
$$MAC\left( ID_A, ID_B, \left\{ ID_{next\_cluster}, M \right\}_{K_{A,B}}, K'_{A,B} \right)$$

Küme içinde ara düğümler tarafından taşınan mesajda $K_{cluster}$ ile şifrelenen küme başları arası bağlantı mesajı küme içi haberleşen düğümler tarafından deşifre edilmez ve değiştirilmez. Bu mesaj sadece diğer kümeye iletecek uç düğüme ulaştığında $K_{cluster}$ ile deşifrelenir ve sonrasında diğer kümenin uç elemanı ile paylaşılan ikili ortak anahtarlar ile şifrelenerek iletilir. Mesaj, iletildiği kümeden ana istasyona doğru giden diğer küme sınırına kadar ara düğümler tarafından iletilir ve bu mesaj güvenliğinde artık yeni kümenin küme anahtarı kullanılır.

#### 3.3.5.2 Üye operasyonları

**Yeni eleman ekleme:** ECDH ile yapılanmada imza yöntemi ile kimlikleme yapılarak ağa yeni üye katılımı güvenli şekilde sağlanabilir. Bunun için katılacak düğüm açık anahtarını ve CA tarafından imzalanmış kopyası ile ağdaki en yakın kısmına katılma isteğinde bulunur. İsteği alan düğüm elindeki CA açık anahtarı ile mesajdaki sertifikayı doğrulamaya çalışır. Eğer sertifika doğru ise yeni düğüm ağa katılır. Bu yöntem, imza onay kısmına *ID* bilgisini de ekleyen TinyPk Watro ve ark. (2004) çalışmasında da aynen kullanılmıştır. Sonuç olarak yeni üye ağdaki bir kümeye eklenir ve ilgili küme başına bu bilgi küme içi çok zıplamalı iletim ile yollanır. Daha sonra küme başı yeni bir küme anahtarı, $K_{cluster}$, belirleyerek bunu çok zıplamalı olarak küme içindeki tüm düğümlere iletir. Yeni küme anahtarının dağıtımında küme içi bağlantı ağacı kullanılır ve iletimde güvenlik için her hatta ait ikili ortak anahtarlar kullanılır.

Watro ve ark. (2004), gerçek zamanlı CA erişimi olmadan özel anahtarı ele geçirilen bir düğümün doğrudan yakalanmasının mümkün olmadığını belirtmiştir.

**Teorem IV:** TK protokolünde yeni eleman ekleme işleminin mesajlaşma karmaşıklığı $\beta$ küme eleman sayısı ve $d$ beklenen komşu sayısı olmak üzere $O$ ($\beta$ + lg$\beta$ + $d$)'dir.

**İspat:** Ağa eklenecek birim öncelikle ağa katılma isteğini bütüneyayar. Bu isteğe haberleşme menzilindeki $d$ adet düğüm yanıt verir ve bu operasyon sırasında toplamda $1 + d$ adet mesajlaşma yapılmış olur. Kimlik doğrulama işlemleri tamamlandığında yeni düğümün bağlandığı ağ üyesi küme başına yeni düğüm kimlik bilgisini çok zıplamalı küme içi hat üzerinden iletir. Bu iletim küme anahtarı ile şifrelenerek yapılır. TK protokolünde kümeye ait herhangi bir birimin küme başına doğru mesaj iletiminde yapılan zıplama miktarı en fazla küme içi bağlantı ağacının yüksekliği kadardır. Teorem III'deki ispatta kümeler birer düğüm olarak düşünüldüğünde aynı ispat küme içi bağlantı ağacının beklenen yüksekliği için de geçerlidir. Dolayısıyla bu ağacın beklenen yüksekliği $\beta$ küme eleman sayısına bağlıdır ve logaritmik bir karmaşıklık gösterir ($O$ (lg$\beta$)). Sonuç olarak yeni üye bilgisi küme başına doğru $O$(lg$\beta$) adet mesaj ile iletilir. Küme başı yeni düğüm eklenme bilgisini aldıktan sonra $K_{cluster}$ küme anahtarını günceller ve küme üyelerine bireyayım yolu ile dağıtır. Bu dağıtımda ikili ortak anahtarlar kullanılır. Dağıtım tüm küme üyelerine yapıldığı için bu işlem için toplam $\beta$ adet bireyayım yapılır. Sonuç olarak mesajlaşma karmaşıklığı $O$ ($1 + d +$ lg$\beta + \beta$) = $O$ ($\beta$ + lg$\beta + d$)'dir. □

**Kümeden ayrılma:** Bir duyarga düğümünün ağdaki operasyonu devam ederken başka bir kümeye bağlanması gerekirse önce ayrılma isteğini küme başına iletir. Küme başı bu isteğe karşı kümenin ortak anahtarını değiştirerek bunu küme içi bağlantı ağacı üzerinden ikili ortak anahtarları kullanarak ayrılan düğüm hariç kümeye yeniden bildirir. Bu anahtar kümeden ayrılan düğüme iletilmez.

**Teorem V:** TK protokolünde bir kümeye ait düğümün ayrılma işleminin mesajlaşma karmaşıklığı $\beta$ küme eleman sayısı olmak üzere $O$ ($\beta$)'dir.

**İspat:** Kümeden ayrılacak düğüm ayrılma isteğini küme başına bireyayım ile iletir. Bu istek var olan $K_{cluster}$ ile gerçekleştirir. Bu işlem en fazla küme yüksekliği kadar mesajlaşma gerektirir ve bu da ağacın beklenen yüksekliği olan $O$ (lg$\beta$) adet bireyayım demektir. Küme başı çıkma isteğini aldıktan sonra $K_{cluster}$ küme anahtarını günceller ve küme üyelerine bireyayım yolu ile dağıtır. Bu dağıtımda ikili ortak anahtarlar kullanılır. Dağıtım çıkan birim hariç tüm küme

üyelerine yapıldığı için bu işlem için toplam $\beta$ - 2 bireyayım yapılır. Sonuç olarak yapılan tüm mesajlaşma karmaşıklığı $O$ (lg$\beta$ + $\beta$ - 2) = $O$ ($\beta$) adettir. □

### 3.3.6 Güvenlik maddeleri

**Veri gizliliği:** Gizlilik, duyarga ağında yapılan mesajlaşmaların belirlenen oturum anahtarları ile şifrelenmesi ile sağlanır. Önce ağ içerisinde birbiri ile haberleşen düğüm çiftleri *A*,*B* arasında ikili ortak anahtar ( $K_{A,B}$ ) belirlenir. Daha sonra küme başının kontrolünde küme anahtarı ( $K_{cluster}$ ) belirlenir ve bu anahtar düğümler arası ikili oturum anahtarları kullanılarak küme elemanlarına dağıtılır. Küme içerisinde iletilen mesajlar bu küme anahtarı ile şifrelenir. Bu sayede küme içi düğümler arasındaki haberleşmede şifre/deşifre işlemleri nedeni ile oluşacak enerji ve zaman kayıpları en aza indirgenmiş olur.

**Veri doğruluğu:** Bu özelliğin sağlanması için her mesajın sonuna SHA-1 MAC çıktısı konulur.

**Veri tazeliği:** Güvenli iletimde veri tazeliği için SPINS protokolünde tanımlanan sayıcı kontrollü çözüm aynen kullanılır.

**Kimlikleme:** Anahtar dağıtımı bazlı anahtar belirleme protokolünde düğümlere kaydedilen anahtar listeleri aynı zamanda bu düğümlerin aynı kaynak tarafından oluşturulduğunu belirler. ECC tabanlı anahtar belirleme protokolünde ise ilk kurulum sırasında ECDH ile ortak anahtar belirlenirken RSA veya ECDSA imza yöntemleri kullanılarak kimlikleme yapılır. İkili ortak anahtar $K_{A,B}$ belirlendikten sonra ise $f(K_{A,B}) = K'_{A,B}$ anahtarı ile kimlikleme yapılır.

**Direnç:** Güvenli haberleşme sırasında kullanılan sayıcı değerleri mesaj tazeliği oluşturduğundan bu mesajların tekrarlanarak kullanılması mümkün değildir. Bu nedenle güvenlik protokolü dirençlidir.

**Dayanıklılık:** Asimetrik anahtar belirleme için herhangi bir duyarga düğümünün ele geçirilmesi durumunda bu düğümün üye olduğu küme dışında ağın diğer bölümlerindeki güvenli haberleşmenin açığa çıkması mümkün değildir.

Çünkü her küme kendine has bir anahtar ile iç haberleşmesine devam eder. Diğer kümeler ile olan bağlantı ise ağ geçitleri arası ikili ortak anahtarlar ile gerçekleştirilir. Ayrıca küme içi anahtar belirleme ve dağıtımı ikili ortak anahtarlar üzerinden yapıldığı için yalnızca bu düğümün komşuları ile yapmış olduğu haberleşmedeki ikili oturum anahtarları açığa çıkar. Bu nedenle güvenlik protokolü asimetrik yöntemler kullanıldığında ele geçirme saldırılarına karşı dayanıklı hale gelir. Simetrik anahtar belirleme protokolünde ise dayanıklılık düşüktür.

## 3.4 Özet ve Yorumlar

Çalışmanın bu bölümünde ilk olarak tez kapsamında geliştirilen ve Tekrarlı Kümeleme (TK) protokolü ile ağ seviyesinde uygulanan anahtar belirleme protokollerinin analizi, çapraz seviye tasarım detayları ve benzetime dayalı performans sonuçları verilmiştir. Daha sonra ikili oturum anahtarlarının belirlenmesinden sonra ağda güvenli iletim için kullanılacak anahtarların belirlenmesi işlemleri tanımlanmış ve ağa katılma ve ağdan çıkarılma durumlarında yapılacak anahtar güncelleme işlemlerinin maliyetleri analiz edilmiştir. Bu kapsamda incelenen anahtar belirleme protokolleri, duyarga düğümlerinin yapılanırken sadece tek zıplama mesafesindeki komşuları ile paylaşacakları ikili ortak anahtarları belirleyebilmesini sağlayan anahtar ön dağıtımı tabanlı Temel Şema ve açık anahtar şifreleme tabanlı ECDH protokolleridir.

Bu protokollerin çapraz seviye uygulamalarının performans analizi, TK protokolünde genel ağ bağlantı seviyesini yüksek seviyede tutabilen mümkün olan en düşük fiziksel ağ yoğunluğunda yapılmıştır. Yapılan benzetimlerde, Temel Şema ve ECDH kullanılarak oluşturulan güvenli hatlar üzerinden yapılanan TK protokolünün *düz*, *reaktif* ve *proaktif* yapılanma seçenekleri için performans sonuçları alınmış ve bu sonuçlar karşılaştırmalı olarak değerlendirilmiştir.

Benzetim sonuçları özetlendiğinde Temel Şema ve ECDH anahtar belirleme protokollerinin TK protokolü ile çapraz seviye uygulamasında *reaktif* ve *proaktif* ECDH yapılanmasındaki enerji maliyetlerinin yüksek dayanıklılık ve yapılanma

performansı veren Temel Şema yapılanmasının enerji maliyetine göre yaklaşık %20–%30 daha düşük olduğu belirlenmiştir. Burada referans alınan Temel Şema yapılanma parametreleri, anahtar havuzu büyüklüğü $P$ = 100.000 ve anahtar liste uzunluğu $k$ = 300'dür. Buna rağmen Temel Şema ile sağlanan dayanıklılık seviyesi ECDH yapılanmasına göre üçte bir oranında daha düşüktür. Bu sonuçların alınmasında Temel Şema'nın asıl maliyet kalemi haberleşmede, ECDH protokolünün ise hesaplamadadır.

Temel Şema'da ECDH ile aynı düğüm derecesine sahip bir ağda karşılaştırılabilir bir genel ağ bağlantı oranı ve dayanıklılık elde edebilmek için duyarga düğümlerine belirli bir miktarda anahtarın önceden yüklenmesi gerekmektedir. Bu da Temel Şema'nın yapılanma sırasındaki iletim maliyetlerini artıran bir unsurdur. Temel Şema'nın asıl maliyet kalemini oluşturan bu haberleşmenin azaltılabilmesi için ağdaki duyarga düğüm derecesinin artırılması ve aynı genel ağ bağlantı oranı için gerekli olan yerel bağlantı oranını sağlayabilecek daha kısa anahtar listelerinin kullanılması mümkündür. Bu sayede iletilecek anahtar liste uzunlukları azaltılabilir ve mesajlaşma maliyeti düşürülebilir. Hatta, daha az kullanılan anahtar uzunlukları ile sistemin dayanıklılığı daha da artırılabilir. Fakat bu yöntem ile birim alana düşen duyarga düğüm sayısı artacağı için özellikle geniş ölçekli ağlarda ağın toplam maliyeti de artmaktadır. Ayrıca ortak anahtar keşfi için yapılan ilk bütüneyayım mesajı daha fazla düğüm tarafından alınacağı için iletim maliyetlerinde mutlak bir düşüşten de bahsedilemez.

ECDH ile yapılanma durumunda ise kablosuz ortamın izin verdiği ölçüde fiziksel komşulukta bulunan tüm duyarga düğümleri ile bağlantı kurulabildiği için duyarga düğüm derecesinin en düşük seviyede tutulması ile istenen genel ağ bağlantı oranına ulaşılabilmektedir. Bu da ECDH hesaplama maliyetlerinin kendi içinde en düşük seviyede tutulmasını sağlamaktadır. Burada ECDH protokolü ile yapılanmanın kimliklemeli olmadığı durumda *mitm* saldırılarına açık olması Temel Şema ile yapılanmaya göre bir dezavantaj olarak sıralanabilir. Bu dezavantajı giderebilmek için ise tez çalışmasında açık anahtar şifreleme protokolleri için önerilen imza yöntemleri ile kimliklemenin sisteme maliyeti incelenmiştir. Burada RSA ile imza yönteminin hesaplama maliyetlerindeki

asimetrisi sayesinde yüksek maliyet gerektiren imza işlemi kaynak açısından daha güçlü olan bir sistem tarafından (Örneğin ana istasyon) yerleşimden önce gerçekleştirilmekte ve yapılanma sırasında düğümlerin daha az maliyetli olan onay işlemlerini yaparak enerjiyi daha etkin kullanabilmelerine olanak sağlanmaktadır. Benzetim sonuçlarına göre RSA imza yöntemi ile gerçekleştirilen *reaktif* ECDH anahtar belirleme operasyonlarının sistemin enerji maliyetini %50 oranında artırdığı fakat toplam maliyetin Temel Şema'ya göre sadece %27 daha fazla olduğu belirlenmiştir. Sonuç olarak bu fark yüksek güvenlik gerektiren duyarga ağı uygulamaları için kabul edilebilir bir maliyet artışıdır çünkü bu enerji sadece ortak anahtarlar belirlenirken harcanacak olup karşılığında alınan güvenlik seviyesinde düğümlerin bire bir kimliklemesi yapılabilmekte ve ağın dayanıklılığı ciddi oranda artırılmaktadır.

Dolayısı ile TK protokolünde güvenli haberleşme için ikili ortak anahtarların belirlenmesinde kullanılacak anahtar belirleme protokolünün RSA imza yöntemi ile birlikte kullanılan ECDH protokolü olarak seçilmesi yüksek güvenlik hedefi için daha uygundur. Çapraz seviye tasarım ile elde edilen *reaktif* yapılanma yöntemi ise yapılanma performansını etkilemeden maliyeti daha da düşürmektedir. Diğer taraftan bu sonuçlar ağ yoğunluğu yaklaşık 8 iken yapılanabilen tüm duyarga ağ organizasyon protokollerine de genellenebilir durumdadır.

İkili ortak anahtarlar belirlendikten sonra ağda güvenli iletim için kullanılacak oturum anahtarları SPINS çalışmasında belirtildiği gibi belirlenir. Küme içi güvenli haberleşmede her zıplamada şifreleme deşifreleme işlemi yapılmasını engellemek için küme başı tarafından belirlenen ve ikili oturum anahtarları ile kümeye dağıtılan $K_{cluster}$ kullanılır. Kümeye eleman katılımı veya kümeden eleman çıkarılması söz konusu olduğunda bu anahtar küme başı tarafından güncellenir ve ikili oturum anahtarları kullanılarak kümeye yeniden dağıtılır. Kümelere elemen ekleme ve çıkarma için yapılan toplam mesajlaşma karmaşıklığı, ağ yoğunluğu sabit ve küme eleman sayısından küçük olmak üzere, küme eleman sayısına doğrusal olarak bağlıdır. Bu da güvenli şekilde yapılan üye işlemlerinin toplam ağ eleman sayısına göre ölçeklenebilir olmasını sağlamaktadır.

# 4 SONUÇ

Bu tez çalışmasında kablosuz duyarga ağlarında güvenli grup haberleşme altyapısının oluşturulmasında düşük düğüm derecesine sahip ağlarda yüksek kümeleme performansı ve bağlantı oranı sağlayan ölçeklenebilir bir yapılanma protokolü geliştirilmiş ve bu yapılanma sırasında duyarga düğümleri arasında güvenli iletim için kullanılacak ikili ortak anahtarların belirlenmesinde anahtar belirleme protokolleri ile yapılanma protokolü arasındaki çapraz ilişki ortaya konmaya çalışılmıştır.

Tez kapsamında geliştirilen Tekrarlı Kümeleme (TK) protokolü, geniş ölçekli kablosuz ağlar için ölçeklenebilir yapılanma zamanına sahip, düşük düğüm yoğunluklarında (7-8) etkin şekilde kendi kendine organize olabilen çok zıplamalı bir kümeleme protokolüdür. Protokolde, oluşturulan kümelerin eleman sayısını sınırlandırmak için orijinali Krishnan ve Starobinski (2006) tarafından önerilen ve bu tez kapsamında durum kontrolü mekanizması ile geliştirilen *persistent* algoritması kullanılmıştır. Orijinal algoritmaya eklenen bu durum kontrolü ile küme oluşumu sırasında yapılan toplam mesajlaşma maliyetlerini azaltılmıştır.

TK protokolü ile yapılanan ağda ilk küme hariç tüm kümelerin oluşumu, yapılanması tamamlanmış diğer küme elemanları tarafından başlatılır. Yeni kümeleri başlatma işi yapılanan kümelerdeki uç düğümler tarafından gerçekleştirilir. Bu uç düğümlerin belirlenebilmesi için tez çalışmasında GKA-1 ve GKA-2 algoritmaları geliştirilmiş ve bu algoritmaları kullanan TK protokollerinin benzetim yolu ile performans analizleri yapılmıştır. Bu analizler Krishnan ve Starobinski (2006) tarafından önerilen zamanlayıcı tabanlı kümeleme (ZTK) protokolünün benzetim sonuçları ile de karşılaştırılmış ve bu protokollerin avantaj ve dezavantajları ortaya konmuştur.

Benzetim sonuçlarına göre, yapılanma zaman maliyetleri açısından TK protokolleri ZTK protokolünden 4-5 kat daha hızlı şekilde yapılanmayı tamamlamaktadır. Kümeleme performansları karşılaştırıldığında, özellikle düşük yoğunluklu ağların yapılanmasında TK protokolleri ZTK protokolüne göre %20 daha az sayıda küme oluşturmakta ve oluşturulan kümelerin eleman sayı

ortalaması %30 daha yüksek olmaktadır. Bu performans TK protokolünün ZTK protokolünden daha iyi bir kümeleme performans karakteristiği gösterdiğini ortaya koymaktadır. Ağ yoğunluğunun artması durumunda ise ZTK ile TK (GKA-1) protokolleri benzer karakteristik göstermekle birlikte TK (GKA-2) kümeleme performansı düşmektedir. Mesajlaşma maliyetleri karşılaştırıldığında ise kümeleri başlatmak için yapılan mesajlaşmalar nedeni ile TK protokollerinin toplam mesajlaşma maliyeti ZTK protokolüne göre %30 daha fazladır. Bu fazla maliyet, kümelerin başlatılması sırasında kurulan kümeler arası hatların oluşumu içindir ve bu hatlar aynı zamanda ana istasyona iletimi sağlayan doğal bir yönlendirme yolu da oluşturur. ZTK protokolü ile yapılanmada ise böyle bir bağlantı hattı oluşturulmaz ve bunun için yapılanma tamamlandıktan sonra ayrıca mesajlaşmalar yapılması gerekmektedir. TK protokolündeki bu mesajlaşma maliyeti küme başlatıcılarının seçilmesi için yapılan bütüneyayım miktarının fazla olması nedeni ile TK (GKA-1) protokolünde TK (GKA-2) protokolüne göre %10 daha fazladır fakat TK (GKA-2) protokolünün yapılanma zamanı zamanlayıcı tabanlı başlatıcı seçimi nedeni ile TK (GKA-1) protokolüne göre belli bir ofset değeri daha yüksektir (Benzetimlerde bu değer 18 saniye ölçülmüştür). Bu nedenle düşük yoğunluklu ağlar için zaman ve mesajlaşma maliyetlerinin önem derecesi değerlendirilerek TK (GKA-1) ve TK (GKA-2) protokolleri arasında ayrıca bir tercih yapılabilir. Uygulamaya göre yapılanma zamanı daha önemli ise TK (GKA-1) protokolünün kullanımı daha uygundur. Mesajlaşma maliyetinin daha öncelikli olması durumunda ise TK (GKA-2) protokolü daha iyi performans sergilemektedir. Bu çalışmada, TK protokolü ile yapılanma sırasında fiziksel olarak oluşan bu kümeler arası bağlantının ana istasyonda kümeler arası bağlantı ağacına dönüştürülebilmesi için ihtiyaç duyulan mesajlaşma karmaşıklığın $O$ ($n$lg $\beta$) olduğu belirlenmiştir.

Geliştirilen TK protokolünde yapılanma sırasında ikili ortak anahtarların belirlenmesi için seçilen anahtar belirleme protokolleri, duyarga düğümlerinin yapılanırken sadece tek zıplama mesafesindeki komşuları ile paylaşacakları ikili ortak anahtarları belirleyebilmesini sağlayan anahtar ön dağıtımı tabanlı Temel Şema (Eschenauer and Gligor, 2002) ve açık anahtar şifreleme tabanlı ECDH protokolleridir. Tez kapsamında bu anahtar belirleme protokollerinin ağ seviyesinde uygulanması ve enerji analizi, çapraz seviye tasarım detayları ve

benzetime dayalı performans sonuçları verilmiştir. TK protokolü ile çapraz seviye ilişki, anahtar belirleme protokollerinin farklı yapılanma parametreleri ile uygulanması sonucu elde edilen genel bağlantı oranı, kümeleme performansı, toplam enerji maliyetleri ve dayanıklılık değişimlerinin benzetim yolu ile ortaya konmaya çalışılmıştır. Bu amaçla, yapılanma sırasında ikili ortak anahtar belirleyecek hatlar TK protokolündeki durum kontrolü kullanılarak *düz*, *reaktif* ve *proaktif* olmak üzere üç farklı şekilde seçilmiş ve bu seçenekler için ayrı performans sonuçları alınarak karşılaştırmalı olarak değerlendirilmiştir.

Benzetim sonuçları özetlendiğinde Temel Şema ve ECDH anahtar belirleme protokollerinin TK protokolü ile çapraz seviye uygulamasında *reaktif* ve *proaktif* ECDH yapılanmasındaki enerji maliyetlerinin Temel Şema'ya göre %20–%30 daha düşük olduğu belirlenmiştir. Ayrıca Temel Şema ile sağlanan dayanıklılık seviyesi ECDH yapılanmasına göre üçte bir oranında daha düşüktür. Bu sonuçların alınmasında Temel Şema'nın asıl maliyet kalemi haberleşmede, ECDH protokolünün ise hesaplamadadır. Burada ECDH protokolü ile yapılanmanın kimliklemeli olmadığı durumda *mitm* (*man in the middle*) saldırılarına açık olması Temel Şema ile yapılanmaya göre bir dezavantaj olarak sıralanabilir. Bu dezavantajı giderebilmek için ise tez çalışmasında açık anahtar şifreleme protokolleri için önerilen imza yöntemleri ile kimliklemenin sisteme maliyeti incelenmiştir. Benzetim sonuçlarına göre RSA imza yöntemi ile gerçekleştirilen *reaktif* ECDH anahtar belirleme operasyonlarının sistemin enerji maliyetini %50 oranında artırdığı fakat toplam maliyetin Temel Şema'ya göre sadece %27 daha fazla olduğu belirlenmiştir. Sonuç olarak bu fark yüksek güvenlik gerektiren duyarga ağı uygulamaları için kabul edilebilir bir maliyet artışıdır çünkü bu enerji sadece ortak anahtarlar belirlenirken harcanacak olup karşılığında alınan güvenlik seviyesinde düğümlerin bire bir kimliklemesi yapılabilmekte ve ağın dayanıklılığı ciddi oranda artırılmaktadır. Dolayısı ile TK protokolünde güvenli haberleşme için ikili ortak anahtarların belirlenmesinde kullanılacak anahtar belirleme protokolünün RSA imza yöntemi ile birlikte kullanılan ECDH protokolü olarak seçilmesi yüksek güvenlik hedefi için daha uygundur. Çapraz seviye tasarım ile elde edilen *reaktif* yapılanma yöntemi ise yapılanma performansını etkilemeden maliyeti daha da düşürmektedir. Diğer taraftan bu sonuçlar ağ yoğunluğu yaklaşık

8 iken yapılanabilen tüm duyarga ağ organizasyon protokollerine de genellenebilir durumdadır.

Yapılanma sırasında ikili oturum anahtarları belirlendikten ve yapılanma tamamlandıktan sonra ağda güvenli iletim için kullanılacak anahtarlar SPINS çalışmasında belirtildiği gibi belirlenir. Küme içi güvenli haberleşmede küme başı tarafından belirlenerek küme üyelerine dağıtılan $K_{cluster}$ kullanılır. Kümeye eleman katılımı veya kümeden eleman çıkarılması söz konusu olduğunda bu anahtar küme başı tarafından güncellenir ve ikili oturum anahtarları kullanılarak kümeye yeniden dağıtılır. Kümelere elemen ekleme ve çıkarma için yapılan toplam mesajlaşma karmaşıklığı sırasıyla $O(\beta + \lg\beta + d)$ ve $O(\beta)$ olarak belirlenmiştir. Küme eleman sayısı ile doğru orantılı olarak değişen bu karmaşıklıklar güvenli şekilde yapılan üye işlemlerinin toplam ağ eleman sayısına göre ölçeklenebilir olmasını sağlamaktadır.

Bölümün devamında tez çalışmasında ortaya konan katkılar hakkında özet bilgiler verilmiştir. Bu katkılar TK, çapraz seviye anahtar belirleme ve performans analizi ve grup haberleşme protokol yapısı ana başlıkları altında gruplanabilir. Bölüm sonunda geleceğe dönük yapılabilecek çalışmalar hakkında öneriler sunulmuştur.

## 4.1 Katkılar

### 4.1.1 TK protokolü

**Durum kontrollü kümeleme:** TK protokolünde her bir kümenin sahip olabileceği eleman sayısını sınırlamak için kullanılan algoritma Krishnan ve Starobinsky (2006) tarafından geliştirilen mesaj etkin *persistent* algoritmasından türetilmiştir. Orijinal *persistent* algoritmasında düğümlerin komşularını bildiği varsayılmıştır ve küme genişleme miktarını sınırlayan bütçe değeri bilinen komşulara eşit olarak paylaştırılır. Bu durumda yapılanma sırasında zaten bir kümeye eklenmiş olan duyarga düğümüne, kendisini kümeye ekleyen komşusu hariç diğer komşuları tarafından anlamlı olmayan bütçe dağıtımları yapılır. Tez çalışmasında ise fazladan yapılan bu mesajlaşma maliyetlerini indirmek için

orjinal *persistent* algoritması üzerine bütçe dağıtımı sırasında tüm komşu düğümlerin durum kontrolünün yapıldığı bir algoritma geliştirilmiştir. Bu algoritmanın ilk aşamasında bir kümeye eklenen duyarga düğümü bütçe dağıtacağı fiziksel komşularını keşfederken aynı zamanda bu düğümlerin durum bilgilerini de alır. İkinci aşamada ise elinde bulunan bütçeyi sadece durumları uygun olan (*floating*) komşularına paylaştırır. Dolayısı ile durumu *floating* olmayan düğümlerün önceden yapılandırılıdığı belli olduğundan bu düğümlere bütçe dağıtılmaz ve gereksiz mesajlaşma engellenmiş olur.

**TK ile kendi kendine organizasyon:** TK protokolünde, gelştirilmiş *persistent* algoritması ile ağ yapılanması sırasında yeni kümelerin oluşumunun başlatılması için geliştirilen yöntem yapılan diğer bir katkıdır. Bu yöntemde bir küme yapılanması başka bir kümeye ait bir uç düğüm tarafından başlatılır. Ağ yapılanmasında ilk küme oluşumunu başlatacak düğüm ana istasyon tarafından, diğer tüm kümelerinki ise kendilerinden önce oluşturulan kümelerdeki ağ geçidi (*gateway*) adı verilen seçilmiş uç düğümler tarafından belirlenir. Bu başlatıcıları belirleyecek uç düğümlerin belirlenmesi için iki adet ağ geçidi seçimi algoritması geliştirilmiştir. Bu algoritmalardan ilki olan TK (GKA-1) bir kümenin uç kısımlarında bulunan ve yeni küme başlatıcısı belirleyebilecek tüm aday düğümler arasından birbirlerine uzak ve aynı zamanda genişleme için en fazla komşuya sahip olan düğümleri belirler. İkinci algoritma olan TK (GKA-2) ise yeni küme başlatıcısını seçecek *ağ geçidi* düğümlerini üstel rastsal dağılımla programlanmış zamanlayıcılar ile belirler ve bu sayede yüksek ağ yoğunluklarındaki kümeleme perfromansından feragat edilerek TK (GKA-1) protokolündeki bütüneyayım maliyetleri ihmal edilebilir boyutlara indirgenir. Her iki durumda da kümelerin genişleme alanları mümkün olduğu kadar birbirinden ayrılır ve düşük eleman sayılı küme oluşumlarının önüne geçilmiş olur. Benzetim sonuçlarına göre Bu protokollerin ağ yapılanma zaman karmaşıklığı $O(\lg n)$'dir.

**Tekrarlı ve zamanlayıcı tabanlı ağ yapılanma performans karşılaştırması:** Tez çalışmasında *persistent* algoritması ile kümeleri oluşturan TK protokollerinin ağ yapılanma performansı Krishnan ve Starobinski (2006) tarafından önerilen ZTK protokolünün performansı ile benzetim yolu ile karşılaştırılmış ve avantaj ve dezavantajları ortaya konmuştur. Bu sonuçlara göre

düşük yoğunluklu (≅ 8) ağlarda TK protokolleri ile yaklaşık 4-5 kat daha hızlı yapılanma mümkündür. Ayrıca küme eleman sayısı sınırı artırıldığında TK protokollerinin yapılanma zamanlarının azalmasına karşın ZTK protokolünde yapılanma zamanı daha da artmaktadır. Oluşan küme sayıları karşılaştırıldığında ise düşük yoğunluklu ağlarda TK protokollerinin performansları ZTK protokolüne göre %20 daha iyidir. TK protokollerinin mesajlaşma maliyetleri ise ZTK protokolüne göre yaklaşık %20 daha fazladır fakat bu artışın kaynağı küme başlatıcılarının seçimi için yapılan karar mesajlaşmalarıdır. ZTK protokolünde ise kümeler arası böyle bir bağlantı oluşumu tanımlanmamıştır. Bunun için yapılanmadan sonra ayrıca işlem yapılıp kümeler arası bağlantıların oluşturulması gerekir. Bu da ZTK protokolünde ekstra mesajlaşmaların yapılması anlamına gelmektedir. Son olarak oluşturulan kümelerdeki en fazla derinlik değerleri karşılaştırıldığında ağ yoğunluğu 7 olduğunda ZTK protokol sonuçları TK protokollerinin %60'ı kadardır. Bu açıdan ZTK protokolü performansı daha iyidir çünkü seçilen küme başlatıcıları uzaysal olarak ayrıktır ve kümeye eklenen elemanlar da başlatıcıya daha yakındır.

**Yapılanma sonrası oluşturulan yönlendirme altyapısı:** TK protokolünde kümelerin uç düğümleri ve başlatıcıları arasında kurulan hatlar tüm ağda ana istasyon kök olmak üzere küme başları arası döngüden bağımsız bir bağlantı ağacı oluşturur. Bu bağlantı ağacı ile herhangi bir kümeye ait düğüm ana istasyona göndereceği veriyi önce kümenin başlatıcısı olan küme başına yollar. Bu veri küme başı tarafından kendisini seçen diğer küme uç düğümüne yönlendirilir ve bu şekilde veri ana istasyona kadar herhangi bir döngü kontrolü yapılmadan çok zıplamalı şekilde ulaştırılabilir.

### 4.1.2 Çapraz seviye anahtar belirleme

Geliştirilen TK protokolünde güvenli iletişim için anahtar belirleme protokollerinin çapraz seviye uygulaması ve bu protokollerin performans analizleri yapılmıştır. Uygulanan anahtar belirleme protokolleri anahtar ön dağıtımı ve açık anahtar belirleme tabanlıdır. Seçilen anahtar ön dağıtımı protokolü Eschenauer ve Gligor (2002) tarafından önerilmiştir ve Temel Şema olarak adlandırılır. Açık anahtar tabanlı anahtar belirleme analizi için ECDH

protokolü seçilmiştir. Bu analizlerde kümeleme protokolünün durum kontrol mekanizması kullanılarak ağ konfigürasyonuna bağlı *düz*, *reaktif* ve *proaktif* olmak üzere üç ayrı anahtar belirleme stratejisi geliştirilmiştir. Benzetimlere dayalı analiz sonuçlarına göre anahtar dağıtımı protokollerinin TK protokolü ile çapraz seviye uygulanması sonucu elde edilen güvenlik seviyesi ve buna karşın harcanan toplam enerji maliyetleri karşılaştırıldığında ECDH tabanlı anahtar belirleme performansının daha iyi olduğu ortaya konmuştur. Anahtar belirlenecek hatların yapılanma protokolüne bağlı olarak seçilmesi ile ağ yapılanma performansını etkilemeden anahtar belirleme maliyetlerinin azaltılmasının mümkün olduğu gösterilmiştir. Ayrıca ECDH ile RSA imza yöntemi kullanımı ile ECDSA kullanımına göre daha az enerji harcayarak kimliklemeli yapılanma gerçekleştirilebileceği ve kabul edilebilir bir bedel karşılığında yüksek güvenlik gerektiren kablosuz algılayıcı ağlar için tercih edilebilir olduğu gösterilmiştir.

## 4.2 Tezden Çıkan Yayınlar

Sağlam, Ö. and Dalkılıç, M. E. (2009) A Self Organizing Multihop Clustering Protocol for Wireless Sensor Networks. *To appear in Proceedings of 5th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN'09)*, Wu Yi Mountain, China, 14-16 December. IEEE, New York, NY.

Sağlam, Ö. and Dalkılıç, M. E. (2009) Cross Layer Implementation of Key Establishment Protocols in Wireless Sensor Networks. *In Proceedings of 24th International Symposium on Computer and Information Sciences (ISCIS'09)*, Güzelyurt, Northern Cyprus, 14-16 September, pp. 356-361. IEEE, New York, NY.

## 4.3 Gelecek Çalışmalar

Tez çalışmasında ortaya konan güvenli grup haberleşmesi altyapısı yapılacak yeni çalışmalar ile geliştirilebilir durumdadır. Bu bölümde tez ile ilgili gelecekte yapılabilecek çalışmalar hakkında genel bilgi verilmiştir.

**Enerji etkin veri iletimi:** Tez çalışmasında tanımlanan güvenli grup haberleşme altyapısı ile oluşturulan küme içi ve kümeler arası bağlantılar veri iletimini daha hızlı ve enerji etkin bir şekilde yapacak şekilde yeniden organize edilebilir.

*Kümeler arası bağlantı ağacının iyileştirilmesi:* Ağ-geçitleri üzerinden yapılan bağlantılar küme başları arasında ana istasyona ulaşım için fiziksel olarak tanımlanabilecek en yakın bağlantılar olmayabilir. Bu nedenle ilgili kümelere daha kısa bağlantı kurulumu için küme başı kontrolünde yeni bir yapılanma işlemi başlatılarak kümeler arası bağlantı ağacında iyileştirme yapılabilir. Örneğin Şekil 2-21-III'de örneği verilen kümeler arası bağlantı ağacında 8 numaralı küme başı ağaç üzerinde ana istasyona daha yakın olan 7 numaralı kümenin fiziksel komşuluğunda olduğunu anlayabilir ve ana istasyon kontrolünde küme başları arası bağlantı ağacını güncelleyerek daha kısa yoldan ana istasyona bağlanabilir.

*Küme içi bağlantı ağacının iyileştirilmesi:* Kümeler yapılandırıldıktan sonra küme içi bağlantı ağacı çok zıplamalı yapıya uygun bir MST (*Minimum Spanning Tree*) oluşturacak şekilde yeniden organize edilebilir ve bu sayede küme içi veri iletim masrafı en aza indirgenebilir. Başka bir çözüm olarak küme içinde haberleşmede menzil kısıdı gözetilerek örneğin Heinzelman ve ark. (2002), Gupta ve Younis (2003), Younis ve Fahmy (2004) tarafından önerilen kümeleme protokollerinden biri kullanılarak ikinci bir kümeleme yapılabilir ve küme içi bağlantısı daha enerji etkin şekilde tekrar oluşturulabilir. Ayrıca kümeler oluştururlurken bütçe kullanımı yanında derinlik parametresi de dikkate alınarak uzaysal olarak alana daha eşit yayılan kümelerin oluşumu da sağlanabilir.

**Küme başı değişimi:** Tez çalışmasında, yapılanma sonucu oluşturulan kümelerin başlatıcıları aynı zamanda o kümelerin küme başıdır. Ağın ilerleyen zamanlarında küme başı görevini yürüten duyarga düğümünün enerji durumuna göre daha uzun ömürlü iş görebilmesi için zamanı geldiğinde küme içindeki başka düğümlere görevini devredebilmelidir. Yeni çalışmalar ile protokol yapısına uygun küme başı belirleme yöntemlerinin geliştirilmesi hedeflenmektedir.

**Hata esnekliği:** Küme içinde enerjisi bittiği için veya çevresel faktörlerden dolayı işlev göremeyen duyarga düğümünün fark edilmesi ve küme içinde gerekli üye operasyonlarının başlatılması ağın devamlılığı için önemlidir. Duyarga ağında düğümler üzerinde oluşacak hatalar zaman boyutunda kalıcı, aralıklı ve geçici olmak üzere üç kısma ayrılmıştır (Gupta and Younis, 2003). Bu hatalar bazı çevresel faktörler nedeni ile haberleşmenin etkilenmesinden, haberleşme yoğunluğu nedeni ile oluşan mesaj çarpışmalarından, enerji bitmesi veya donanım bozulması nedenleri ile artık iletim yapamamaktan veya bir saldırgan tarafından ele geçirilerek etkisiz hale getirilmekten kaynaklanabilir. Uygulanacak hata toleransı mekanizmaları operasyon olarak mesaj iletim servisinin devamını sağlamanın yanında hatanın giderildiği yeni durumda sistemin ilk kurulumda sağladığı güvenlik seviyesini de korumalıdır.

**Hareketli ağ yapısı:** Tez kapsamında duyarga düğümlerinin sabit oldukları varsayılmıştır. Gelecekte bu çalışmanın hareketli duyarga düğümleri içeren bir ağ ortamına uyumlandırılması ve performans analizinin yapılması hedeflenen diğer bir çalışmadır.

**Üst seviye uygulamalar:** Ağ seviyesinde geliştirilen güvenli grup haberleşme altyapısı üzerinde çeşitli duyarga ağı uygulamaları tanımlanarak ağın bu uygulamalara karşı sunduğu performans incelenmesi ve daha genel bir protokol tasarımı yapılabilecek diğer bir çalışmadır.

**Güvenli grup haberleşmesi:** Son olarak küme üyelik işlemleri sırasında bu işlemlerin yapıldığı kümelerde yeniden anahtar belirlenmesi ve dağıtımı işlemlerinin daha enerji etkin şekilde yapılması ile ilgili yeni çalışmalar yapılabilir. Ayrıca kümeler arası bağlantı hattı kullanılarak birden fazla küme ile çoğayayım grupları oluşturan ve bu gruplara çoğayayım yapan mekanizmalar geliştirilerek performans analizleri yapılabilir.

# KAYNAKLAR

**Abbasi A. A., Younis M.**, A survey on clustering algorithms for wireless sensor networks, Computer Communications, vol.30 no.14-15, pp. 2826-2841, October, 2007 .

**Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E.**, "Wireless Sensor Networks: A Survey", Computer Networks (Elsevier) Journal, Vol. 38, No. 4, pp. 393-422, March 2002.

**Banerjee S., Khuller S.**, A clustering scheme for hierarchical control in multi-hop wireless networks, in: Proceedings of 20th Joint conference of the IEEE Computer and Communications Societies (INFOCOM' 01), Anchorage, AK, April 2001

**Chan, H., Perrig, A., and Song, D**., Random key predistribution schemes for sensor networks. In *IEEE Symposium on* Security *and Privacy*. Berkeley, California, 197–213, 2003.

**Chan, H. and Perrig, A.**, ACE: An Emergent Algorithm for Highly Uniform Cluster Formation. European Workshop on Sensor Networks. pp154-171, 2004.

**Chan, H. and Perrig, A.**, Pike: Peer intermediaries for key establishment in sensor networks. IEEE Infocom, 2005.

**Cormen T. H., Leiserson C. E., and Rivest R. L.**, Clifford Stein, *Introduction to Algorithms Second Edition*. MIT Press, Cambridge, MA, 2001, pages 265-267, 2001.

**Crossbow Technology**, Mica2Dot Data Sheet, http://www.xbow.com/products/Product_pdf_files/Wireless_pdf/MICA2DOT_Datasheet.pdf, (Erişim tarihi: 18 Ağustos 2009).

**Du, W., Deng, J., Han, Y., Chen, S., and Varshney, P.,** A key management scheme for wireless sensor networks using deployment knowledge. In IEEE Infocom'04, 2004.

**Du W., Wang R., and Ning P.,** "An efficient scheme for authenticating public keys in sensor networks," in Proceedings of 6th ACM International Symposium on Mobile Ad Hoc Network and Computing (MobiHoc), pp. 58–67, 2005.

**Erdos P. and Renyi A.,** "On the evolution of random graphs," Publications of the Mathematical Institute of the Hungarian Academy of Sciences, 5:17–61, 1960.

**Eschenauer, L. and Gligor, V. D.**, A key-management scheme for distributed sensor networks. In 9th ACM conference on Computer and Communications Security, 2002.

**Ghosh, S.K., Patro, R.K., Raina, M., Thejaswi, C., Ganapathy, V.**, Secure group communication in wireless sensor Networks", *Wireless Pervasive Computing, 2006 1st* International *Symposium on* , vol., no., pp. 5 pp.-, 16-18 Jan. 2006.

## KAYNAKLAR (devam)

**Gupta G., Younis M.**, "Fault-Tolerant Clustering of Wireless Sensor Networks," in the Proceedings of the IEEE Wireless Communication and Networks Conference (WCNC 2003), New Orleans, Louisiana, 2003.

**Gura N., Patel A., Wander A., Eberle H., Chang Shantz S.,** "*Comparing Elliptic Curve Cryptography and RSA* on *8-bit CPUs*", CHES, August 2004.

**Hankerson D., Menezes A., Vanstone S.,** "Guide to Elliptic Curve Cryptography", Springer-Verlag New York, Inc. 2004. ISBN 0-387-95273-X.

**Hwang J., and Kim Y.,** "Revisiting random key pre-distribution for sensor networks," In ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN'04), Oct. 2004, pp. 43-52.

**Heinzelman W.B., Chandrakasan A.P., Balakrishnan H.**, Application specific protocol architecture for wireless microsensor networks, IEEE Transactions on Wireless Networking, 2002.

**Holland M., Aures R. and Heinzelman W.,** Experimental Investigation of Radio Performance in Wireless Sensor Networks, Poster *Session*, IEEE SECON 2006.

**Johnson D., Menezes A. and Vanstone S.**, The elliptic curve digital signature algorithm (ECDSA). *International Journal of Information Security*, 1, 36-63, 2001.

**Krishnan R. and Starobinski D.**, Efficient clustering algorithms for self-organizing wireless sensor networks. Ad Hoc Networks 4(1), pages 36-59, 2006.

**Lindsey S. and Raghavendra C. S.**, "PEGASIS: Power Efficient GAthering in Sensor Information Systems," in the Proceedings of the IEEE Aerospace Conference, Big Sky, Montana, March 2002.

**Liu D. and Ning P.**, "Establishing pairwise keys in distributed sensor networks," Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03), pp. 52-61, 2003a.

**Liu, D and Ning, P.**, Efficient distribution of key chain commitments for broadcast authentication in distributed sensor networks. Proceedings of the 10th Annual Network and Distributed System Security Symposium, pp. 263–276, 2003b.

**Liu A., Ning P.**, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks," International Conference on Information Processing in Sensor Networks, pp.245-256, 2008.

**Malan, D., Welsh, M., and Smith, M.**, A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography. In First IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON04), 2004.

**Perrig A., Szewczyk R., Tygar J. D., Wen V. and culler D. E.**, Spins: security protocols for sensor networks. Wireless Networking, 8(5):521–534.

## KAYNAKLAR (devam)

**Pietro, R., Mancini, L., and Mei, A**., Random key assignment secure wireless sensor networks. In 1st ACM workshop on Security of Ad Hoc and Sensor Networks, 2003.

**Piotrowski K., Langendoerfer P., and Peter S.**, How public key cryptography influences wireless sensor node lifetime. In Proceedings of the Fourth ACM Workshop on Security of Ad Hoc and Sensor Networks. SASN '06. ACM, New York, NY, 169-176, 2006.

**Rivest R. L., Shamir A., and Adleman L.,** A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM J.*, 21, 120-126, 1978.

**Schaeffer S. E., Clemens J. C., Hamilton P.**, "Decision Making in a Distributed Sensor Network." In Proceedings of the Santa Fe Institute Complex Systems Summer School, Santa Fe, NM, USA, 2004.

**Sun, H. M., Chen, C. M., Chu, F, Y.**, An Efficient and Scalable Key Management Protocol for Secure Group Communications in Wireless Sensor Networks ISCC 2007: 495-500, 2007.

**Teksas Instruments**, CC1000 Data Sheet, http://focus.ti.com/lit/ds/ symlink/cc1000.pdf, (Erişim tarihi: 18 Ağustos 2009).

**Sentilla Corporation**, Tmote Sky Datasheet, http://www.sentilla.com/pdf/eol /tmote-sky-datasheet.pdf, (Erişim tarihi: 18 Ağustos 2009).

**Thepvilojanapong N., Tobe Y. and Sezaki K.**, Securing Group Communication in Wireless Sensor Networks, IEEE TENCON 2004, Thailand, Nov. 2004.

**Wander A., Gura N., Eberle H., Gupta V., and Shantz S.,** Energy analysis of public-key cryptography for wireless sensor networks. In PERCOM '05: Proceedings of the Third IEEE International Conference on Pervasive Computing and Communications, pages 324–328, Washington, DC, USA, 2005.

**Wang Y., Attebury G., Ramamurthy B.,** A survey of security issues in wireless sensor networks. Communications Surveys & Tutorials, IEEE 8 (2), 2-23, 2006.

**Watro R., Kong D., Cuti S. F., Gardiner C., Lynn C. and Kruus P.** 2004. Tinypk: securing sensor networks with public key technology. In SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks, pages 59–64, New York, NY, USA ACM Press.

**Xiao Y., Rayi V. K., Sun B., Du X., Hu F. and Galloway M.**, A survey of key management schemes in wireless sensor networks. Comput. Commun. 30, 11-12 , 2007

**Xu K., Gerla M.**, A heterogeneous routing protocol based on a new stable clustering scheme, in: Proceeding of IEEE Military Communications Conference (MILCOM 2002), Anaheim, CA, October 2002.

**Younis O., Krunz M., and Ramasubramanian S**., "Node Clustering in Wireless Sensor Networks: Recent Developments and Deployment Challenges," IEEE Network, vol. 20, issue 3, pp. 20-25, May 2006.

## KAYNAKLAR (devam)

**Younis O., Fahmy S.,** HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks, IEEE Transactions on Mobile Computing, v.3 n.4, p.366-379, October 2004.

**Zhu, S., Setia, S., and Jajodia, S.,** Leap: Efficient security mechanisms for large-scale distributed sensor networks. In 10th ACM Conference on Computer and Communications Security (CCS ’03), 2003a.

**Zhu S., Xu S., Setia S. and Jajodia S.,** Establishing pair-wise keys for secure communication in ad hoc networks: A probabilistic approach. In Proceedings of the 11th IEEE International Conference on Network Protocols (ICNP’03), pages 326–335, 2003b.

# TÜRKÇE-İNGİLİZCE SÖZLÜK

| Türkçe | İngilizce |
|---|---|
| Ağ | Network |
| Ağ geçidi | Gateway |
| Ağ geçidi adayı | Gateway candidate |
| Ağ yoğunluğu | Node degree |
| Duyarga düğümü | Sensor node |
| Alıcı/Verici | Transceiver |
| Ana | Parent |
| Ana istasyon | Base station |
| Anahtar belirleme | Key establishment |
| Anahtar dağıtımı | Key distribution |
| Anahtar ön dağıtımı | Key pre-distribution |
| Arama ağacı | Search tree |
| Başlatıcı | Initiator |
| Benzetim | Simulation |
| Bireyayım | Unicast |
| Düğüm derecesi | Node degree |
| Bütüneyayım | Broadcast |
| Çapraz seviye | Cross layer |
| Çoğayayım | Multicast |
| Çok zıplamalı | Multihop |
| Dayanıklık | Resilience |
| Direnç | Resistance |
| Durum | State |
| Duyarga | Sensor |
| Düz | Straight |
| En büyük bağlı komponent | Giant component |

# TÜRKÇE-İNGİLİZCE SÖZLÜK (devam)

| Türkçe | İngilizce |
|---|---|
| Eş zamanlılık | Synchronization |
| Açık anahtar | Public key |
| Genel bağlantı oranı | Global connectivity |
| Grup haberleşmesi | Group communication |
| Güvenlik | Security |
| Güvenli haberleşme | Secure communication |
| İletim/Haberleşme menzili | Communication range |
| Kapsama | Coverage |
| Katmanlı | Hierarchical |
| Kendi kendine organize olan | Self organizing |
| Kimlikleme | Authentication |
| Konuşlanma, Yerleşim | Deployment |
| Kök | Root |
| Küme başı | Cluster head |
| Küme içi | Intra cluster |
| Kümeleme | Clustering |
| Kümeler arası | Inter cluster |
| Olay bazlı | Event based |
| Ortak anahtar | Shared key |
| Ölçeklenebilir | Scalable |
| Patika | Path |
| Rastgele | Random |
| Serbest | Floating |
| Sonlu durum makinesi | Finite state machine |
| Tazelik | Freshness |
| Tek nokta hatası | Single point of failure |

# TÜRKÇE-İNGİLİZCE SÖZLÜK (devam)

| Türkçe | İngilizce |
|---|---|
| Tekrarlı | Iterative |
| Temel Şema | Basic scheme |
| Temsili kod | Pseudocode |
| Tetikleme | Trgiggering |
| Uzaysal | Spatial |
| Yapılandırılmış | Configured |
| Yapılanma | Configuration |
| Yayılan ağaç | Spanning tree |
| Yerel | Local |
| Yerel Bağlantı oran | Local connectivity |
| Yoklama | Poll |
| Yönlendirme | Routing |
| Zamanlayıcı | Timer |
| Zıplama | Hop |

# İNGİLİZCE-TÜRKÇE SÖZLÜK

| İngilizce | Türkçe |
|---|---|
| Authentication | Kimlikleme |
| Broadcast | Bütüneyayım |
| Budget | Bütçe |
| Cascade-off counting | Art arda sayım kapalı |
| Cascade-on counting | Art arda sayım açık |
| Clustering | Kümeleme |
| Configuration | Yapılanma |
| Counter | Sayıcı |
| Decision | Karar |
| Discovery | Keşif |
| Expandig ring | Genişleyen halka |
| Floating | Serbest |
| Gateway | Ağ geçidi |
| Gateway candidate | Ağ geçidi adayı |
| Giant Component | En büyük bağlı komponent |
| Initiator | Başlatıcı |
| Inter cluster | Kümeler arası |
| Intra cluster | Küme içi |
| Level | Seviye |
| Link | Hat |
| Network | Ağ |
| Node | Düğüm |
| Node degree | Ağ yoğunluğu/Düğüm derecesi |
| Pairwise | İkili |
| Parent | Ana/Ebeveyn |
| Path | Patika |

# İNGİLİZCE-TÜRKÇE SÖZLÜK (devam)

| İngilizce | Türkçe |
|---|---|
| Persistent | Israrcı |
| Poll | Yoklama |
| Pseudocode | Temsili kod |
| Random | Rastgele |
| Rapid | Hızlı |
| Sensor | Duyarga |
| Shared key | Ortak anahtar |
| State | Durum |
| Timer | Zamanlayıcı |

# EKLER

EK A TK (GKA-1) PROTOKOLÜ DURUM MAKİNESİ

EK B RASTSAL İKİLİ ARAMA AĞACI

# EK-A: TK (GKA-1) Protokolü Durum Makinesi

**Çizelge A-1:** *floating* durum tablosu

| **AKTİF DURUM: FLOATING** | | | | |
|---|---|---|---|---|
| | **GİRDİ** | | **ÇIKTI** | **SONRAKİ** |
| 1 | Poll_Neighbor | | Poll_Neighbor_Reply | FLOATING |
| 2 | Poll_Gateway | | Poll_Neighbor_Reply | FLOATING |
| 3 | Gateway_Decision_Call | | State_Reply (Opt.) | FLOATING |
| 4 | Outer_Flood | Budget = Max. | Poll_Neighbor | INITIATOR |
| 5 | Inner_Flood | Budget > 1 | Poll_Neighbor | NODE |
| | | Budget = 1 | Poll_Gateway | GATEWAY_C |

Duyarga düğümünün alana atılmadan önce durumu *floating* olarak belirlenir. Bu durumda olan düğümler henüz yapılanmamış ve organizasyona dahil edilmemiştir. Komşuları belirlemek için başka düğümler tarafından bütüneyayımı yapılan *Poll_Neighbor* (Çizelge A-1, Satır 1), *Poll_Gateway* (Çizelge A-1, Satır 2) mesajlarını alan bir *floating* düğüm durum bilgisi ve *ID* numarasını içeren *Poll_Neighbor_Reply* mesajı ile yanıt verir. Ağ geçidi adayı düğümler tarafından yollanan *Gateway_Decision_Call* (Çizelge A-1, Satır 3) mesajına da seçmeli olarak durum bilgisini içeren bir bilgi mesajı yollar. Bu cevap protokol için zorunlu değildir ve sadece yoklama yapan düğüme güncel durum bilgisinin iletilmesi için yollanır. Bu üç girdi mesajı da duyarga düğümünün *floating* durumunu değiştirmez. *Outer_Flood* (Çizelge A-1, Satır 4) mesajı sadece *floating* durumdaki düğümlere yollanır ve mesajdaki bütçe en yüksek değerdedir. Bu mesajı alan düğüm yeni küme başlatıcı olarak seçildiğini belirler ve durumunu *initiator* olarak değiştirerek komşularını belirlemek için *Poll_Neighbor* mesajını bütüneyayar. *Inner_Flood* (Çizelge A-1, Satır 5) mesajı küme içi genişleme için yollanır ve bu mesajı alan *floating* durumdaki düğüm içindeki bütçe değerine göre alacağı aksiyonu belirler. Eğer bütçe değeri 1'den büyük ise (değer en yüksek) mesajı yollayan düğümün kümesine dahil olarak durumunu *node* olarak değiştirir ve komşularını belirlemek için *Poll_Neighbor* mesajını bütüneyayar. Ardından *Poll_Neighbor_Reply_Tout* değerine eşitlenmiş zamanlayıcısını çalıştırır. Bütçe değeri 1'e eşit ise yine kendisini mesajı yollayan düğümün kümesine dahil eder fakat küme genişlemesi için elinde ekstra bütçe kalmadığı için küme genişlemesini durdurur. Kümenin uç düğümü olması nedeni ile de durumunu *gateway_c* olarak değiştirir ve komşularını belirlemek için *Poll_Gateway* mesajını

bütüneyayar. Ardından *Poll_Gateway_Reply_Tout* değerine eşitlenmiş zamanlayıcısını çalıştırır.

**Çizelge A-2:** *node* durum tablosu

| **AKTİF DURUM: NODE** | | | | |
|---|---|---|---|---|
| | **GİRDİ** | | **ÇIKTI** | **SONRAKİ** |
| 1 | Poll_Neighbor | | Poll_Neighbor_Reply | NODE |
| 2 | Poll_Neighbor_Reply | | NOP | NODE |
| 3 | Poll_Neighbor_Reply_Tout | | Inner_Flood | NODE |
| 4 | Poll_Gateway | | Poll_Gateway_Reply | NODE |
| 5 | Poll_Gateway_Reply | | NOP | NODE |
| 6 | Poll_Gateway_Reply_Tout | | NOP | NODE |
| 7 | Gateway_Decisicion_Call | | State_Reply (Opt.) | NODE |
| 8 | Gateway_Decisicion_Reply | | NOP | NODE |
| 9 | Gateway_Decisicion_Reply_Tout | | NOP | NODE |
| 10 | Outer_Flood | Budget = Max. | NOP | NODE |
| 11 | Inner_Flood | Parent valid | Inner_Flood | NODE |
| 12 | | Parent not valid | Reply_Budget | NODE |
| 13 | Config_Ack | Budget&Neighbor | Inner_Flood | NODE |
| 14 | | Else | Config_Ack | NODE |

*Node* durumunda olan bir duyarga düğümü girdi olarak aldığı *Poll_Neighbor* (Çizelge A-2, Satır 1) ve *Poll_Gateway* (Çizelge A-2, Satır 4) mesajlarına durum bilgisi ve *ID* numarasını içeren *Poll_Neighbor_Reply* ve *Poll_Gateway_Reply* mesajları ile yanıt verir. Duyarga düğümü bu girdiler sonucunda *node* olan durumunu değiştirmez. Protokole göre eğer bir düğüm daha önce *floating* durumundan *node* durumuna geçmiş ise bu aşamada ilk olarak komşularını belirlemek için *Poll_Neighbor* mesajını bütüneyayar ve komşu belirleme zamanlayıcısını çalıştırır. Bu mesaja haberleşme menzilindeki duyarga düğümleri *Poll_Neighbor_Reply* (Çizelge A-2, Satır 2) mesajları ile yanıt verirler. Bu yanıtlar yoklamayı yapan düğüm tarafından *node* durumunda iken alınır ve mesajda bulunan *ID* ve durum bilgileri komşu listesine kaydedilir. Durum değiştirilmez ve bir çıktı oluşturulmaz. Komşu belirleme zamanlayıcısı sıfırlandığında *Poll_Neighbor_Reply_Tout* (Çizelge A-2, Satır 3) mesajı alınır ve komşu listesine kaydedilen *floating* durumundaki duyarga düğümlerine bütçe taksimi yapılır (Çizelge 2-1, Satır 42-48). Burada bütçe tamamen kullanılıncaya veya bütçeyi kullanacak düğüm kalmayıncaya kadar bütçe dağıtımı yapılır ve en sonunda toplamda harcanan bütçe bilgisi ebeveyn duyarga düğümüne yollanır (Çizelge 2-1, Satır 52). Eğer duyarga düğümü *gateway_c* durumunda iken *gateway* karar işlemleri sırasında durumunu *node* olarak değiştirmek zorunda kalmış ise *gateway_c* durumunda iken alması gereken *Poll_Gateway_Reply* (Çizelge A-2, Satır 5) mesajını *node* durumunda iken alır. Bu girdi ile mesajda

bulunan *ID* ve durum bilgileri komşu listesine kaydedilir ve başka işlem yapılmaz, durum değiştirilmez. Aynı şekilde *gateway_c* durumunda iken başlatılan zamanlayıcı da *node* durumunda iken sıfırlanır. Bu şekilde, *node* durumunda iken *Poll_Gateway_Reply_Tout* (Çizelge A-2, Satır 6) mesajını alan düğüm aslında uç düğüm olduğu için hiçbir işlem yapmayarak durumunu sabit tutar ve bir sonraki girdiyi bekler. *Gateway_Decision_Call* (Çizelge A-2, Satır 7) mesajı *node* durumundaki düğüm tarafından alındığında seçmeli olarak durum bilgisi içeren bir bilgi mesajı yollanır. *Floating* durumda da olduğu gibi hazırlanan bu cevap protokol için zorunlu değildir ve sadece yoklama yapan düğüme güncel durum bilgisinin iletilmesi için yollanır. *Gateway_Decisicion_Reply* (Çizelge A-2, Satır 8) mesajı *gateway* durumunda iken karar işlemleri sırasında *node* durumuna geçmek zorunda kalan duyarga düğümleri tarafından alındığında sadece komşu durum bilgileri güncellenir ve başka bir işlem yapılmaz, durum değiştirilmez. Aynı şekilde *Gateway_Decisicion_Reply_Tout* (Çizelge A-2, Satır 9) girdisi alındığında hiçbir işlem yapılmaz. *Outer_Flood* (Çizelge A-2, Satır 10) mesajı durumu *node* olan düğümlere gönderilmez fakat daha önce *floating* olarak kaydedilmiş fakat daha sonra durumunu *node* olarak değiştirmiş düğüme durum güncellemesi yapılmamış olması nedeni ile yollanabilir. Bu durumda hiçbir işlem yapılmaz ve durum değiştirilmez. *Node* durumunda iken alınan *Inner_Flood* mesajı eğer *parent* tarafından yollanmış ise (Çizelge A-2, Satır 11) bu mesaj arta kalan bütçenin değerlendirilmek üzere taksim edilerek daha önce bütçe yollanmış düğümlere tekrar yollanmış olduğu anlamına gelir. Bu mesaj ile bütçe dağıtımı algoritması tekrar çalıştırılır ve genişlemeye müsait komşusu bulunan tüm komşu düğümlere taksim edilir. Eğer düğümün *parent*'i tarafından yollanmamış ise (Çizelge A-2, Satır 12) alınan bütçe aynen geri yollanır. *Config_Ack* mesajı küme genişlemesi sırasında bütçe yollanan komşu düğümlerden alınır ve içeriğinde komşuların kendilerine ayrılan bütçenin ne kadarını kullanıdıklarına dair bilgi bulunur. Bütçe dağıtılan tüm komşulardan yanıt alındığında, arta kalan bütçe varsa bu bütçe tekrar genişleme yapabilecek komşulara taksim edilir ve genişleme işlemi bütçe tamamlanana veya genişleyecek düğüm kalmayana kadar devam eder (Çizelge A-2, Satır 13). Sürecin sonunda düğüm elinde kalan bütçe bilgisini *Config_Ack* mesajı ile *parent*'ına yollar (Çizelge A-2, Satır 14).

**Çizelge A-3:** *gateway_c* durum tablosu

| **AKTİF DURUM: GATEWAY_C** | | | | |
|---|---|---|---|---|
| | **GİRDİ** | | **ÇIKTI** | **SONRAKİ** |
| 1 | Poll_Neighbor | | Poll_Neighbor_Reply | GATEWAY_C |
| 2 | Poll_Gateway | | Poll_Gateway_Reply | GATEWAY_C |
| 3 | Poll_Gateway_Reply | | NOP | GATEWAY_C |
| 4 | Poll_Gateway_Reply_Tout | | Gateway_Decision_Reply | NODE |
| 5 | | | Gateway_Decision_Reply | GATEWAY |
| 6 | Gateway_Decisicion_Call | | NOP | GATEWAY_C |
| 7 | Outer_Flood | Budget = Max. | NOP | GATEWAY_C |
| 8 | Inner_Flood | Parent valid | Inner_Flood | NODE |
| 9 | | Parent not valid | Reply_Budget | GATEWAY_C |

Protokole göre eğer bir düğüm *floating* durumundan *gateway_c* durumuna geçmiş ise bu aşamada ilk olarak komşularını belirlemek için *Poll_Gateway* mesajını bütüneyayar ve komşu belirleme zamanlayıcısını çalıştırır. *Gateway_c* durumunda olan bir duyarga düğümü girdi olarak aldığı *Poll_Neighbor* (Çizelge A-3, Satır 1) ve *Poll_Gateway* (Çizelge A-3, Satır 2) mesajlarına durum bilgisi ve *ID* numarasını içeren *Poll_Neighbor_Reply* ve *Poll_Gateway_Reply* mesajları ile yanıt verir. Duyarga düğümü bu girdiler sonucunda durumunu değiştirmez. *gateway_c* durumunda iken alınan *Poll_Gateway_Reply* (Çizelge A-3, Satır 3) mesajı ile mesajda bulunan *ID* ve durum bilgileri komşu listesine kaydedilir ve başka işlem yapılmaz, durum değiştirilmez. *gateway_c* durumunda iken komşu belirleme zamanlayıcısı sıfırlanması ile *Poll_Gateway_Reply_Tout* mesajını alan düğüm haberleşme menzilindeki komşularını belirlemiş durumdadır. Protokole göre bir duyarga düğümü durumunu *gateway_c* olarak değiştirmesinden komşu belirleme zamanlayıcısı sıfırlanana kadar geçen sürede diğer ağ geçidi adaylarından *Gateway_Decisicion_Call* (Çizelge A-3, Satır 6) mesajı ile aldığı *f* ve *p* parametrelerini kaydeder. Zamanlayıcı sıfırlandığında ise kendisine ait parametreleri belirler ve kaydettiği diğer *gateway_c* düğümlerine ait değerler ile karşılaştırır. Eğer daha iyi parametreye ait komşu bir *gateway_c* varsa durumunu *node* olarak değiştirir (Çizelge A-3, Satır 4). Aksi takdirde durumunu *gateway* olarak değiştirir (Çizelge A-3, Satır 5). Her iki durumda da listesinde kendisinden daha düşük parametretelere sahip kayıtlı tüm diğer *gateway_c* komşu düğümlerine *gateway* karar işlemini durdurmaları için *Gateway_Decisicion_Reply* mesajı yollar. *Outer_Flood* (Çizelge A-3, Satır 7) mesajı durumu *gateway_c* olan düğümlere gönderilmez fakat daha önce *floating* olarak kaydedilmiş fakat daha sonra durumunu *gateway_c* olarak değiştirmiş düğüme durum güncellemesi yapılmamış olması nedeni ile yollanabilir. Bu durumda hiçbir işlem yapılmaz ve

durum değiştirilmez. *Gateway_c* durumunda iken alınan *Inner_Flood* mesajı eğer *parent* tarafından yollanmış ise (Çizelge A-3, Satır 8) bu arta kalan bütçenin değerlendirilmek üzere taksim edilerek daha önce bütçe yollanmış düğümlere tekrar yollanması anlamına gelir. Bu mesaj ile *durum* node olarak değiştirilir, bütçe dağıtımı algoritması tekrar çalıştırılır ve genişlemeye müsait komşusu bulunan tüm komşu düğümlere taksim edilir. Eğer mesaj düğümün *parent*'i tarafından yollanmamış ise (Çizelge A-3, Satır 9) alınan bütçe aynen geri yollanır.

**Çizelge A-4:** *gateway* durum tablosu

| **AKTİF DURUM: GATEWAY** | | | |
|---|---|---|---|
| | **GİRDİ** | **ÇIKTI** | **SONRAKİ** |
| 1 | Poll_Neighbor | Poll_Neighbor_Reply | GATEWAY |
| 2 | Poll_Gateway | Poll_Gateway_Reply | GATEWAY |
| 3 | Gateway_Decisicion_Call | NOP | NODE |
| 4 | | Gateway_Decision_Reply | GATEWAY |
| 5 | Gateway_Decisicion_Reply | NOP | NODE |
| 6 | Gateway_Decision_Tout | Outer_Flood | GATEWAY |
| 7 | Gateway_Outerflood_Ack_Tout | Outer_Flood | GATEWAY |
| 8 | Config Ack | NOP | GATEWAY |
| 9 | Outer_Flood | NOP | GATEWAY |
| 10 | Inner_Flood | Reply_Budget | GATEWAY |

*Gateway* durumuna giren bir duyarga düğümü bir sonraki kümeyi başlatmadan önce kendi haberleşme menzilinde ağ geçidi olmaya en uygun olan düğümün kendisi olduğuna dair son kararı vermesi gerekmektedir. *Gateway_c* iken aldığı *Gateway_Decisicion_Call* mesajları ile bu işlemin bir kısmı gerçekleştirilir. Diğer taraftan *gateway_c* olan durumunu *gateway* olarak değiştiren herhangi bir adayın haberleşme menzilinde henüz komşu belirleme işlemlerini bitirmemiş dolayısı ile bu düğüme *Gateway_Decisicion_Call* mesajını henüz yollamamış başka ağ geçidi adayları da bulunabilir. Bu adayların daha iyi komşuluk parametrelerine sahip olma olasılıkları bulunduğu için mutlaka değerlendirilmeye alınması gerekmektedir. Bunun için *gateway_c* durumundan *gateway* durumuna geçen düğüm hemen *Gateway_Decisicion_Call* mesajını bütüneyayar ve yoklama zamanlayıcısını çalıştırır. Bu süre içerisinde girdi olarak aldığı *Poll_Neighbor* (Çizelge A-4, Satır 1) ve *Poll_Gateway* (Çizelge A-4, Satır 2) mesajlarına durum bilgisi ve *ID* numarasını içeren *Poll_Neighbor_Reply* ve *Poll_Gateway_Reply* mesajları ile yanıt verir. Duyarga düğümü bu girdiler sonucunda durumunu değiştirmez. *Gateway* durumundaki düğüm *Gateway_Decisicion_Call* mesajı aldığında mesajdaki $f$ ve $p$ parametreleri ile

kendisine ait parametreleri karşılaştırır. Eğer mesajdaki parametreler daha iyi ise durumunu *node* olarak değiştirir (Çizelge A-4, Satır 3). Aksi takdirde durumunu değiştirmez ve mesajı yollayan ağ geçidi adayı komşu düğüme karar işlemini durdurması için *Gateway_Decisicion_Reply* mesajı yollar (Çizelge A-4, Satır 4). *Gateway_Decisicion_Reply* (Çizelge A-4, Satır 5) mesajı alındığında *gateway* olan durum *node* olarak değiştirilir ve başka bir işlem yapılmaz. Yoklama zamanlayıcısı dolduğunda ağ geçidi adayı düğüm *Gateway_Decision_Tout* (Çizelge A-4, Satır 6) girdisini alır, karar işlemini sonuçlandırır ve yeni küme başlatıcısını komşu listesi içinden seçerek *Outer_Flood* mesajını yollar. Belli bir süre içerisinde bu mesaja onay alınamazsa zamanlayıcı sıfırlanır ve *Gateway_Outerflood_Ack_Tout* (Çizelge A-4, Satır 7) girdisi alınır. Bu durumda listedeki komşulardan başka biri seçilir ve bu düğüme *Outer_Flood* mesajı yollanır. *Config_Ack* (Çizelge A-4, Satır 8) mesajı alındığında işlem yapılmaz ve küme başlatıcısı seçme işlemi tamamlanır. *Outer_Flood* (Çizelge A-4, Satır 9) mesajı durumu *gateway* olan düğümlere gönderilmez fakat daha önce *floating* olarak kaydedilmiş fakat daha sonra durumunu *gateway* olarak değiştirmiş düğüme durum güncellemesi yapılmamış olması nedeni ile yollanabilir. Bu durumda hiçbir işlem yapılmaz ve durum değiştirilmez. *Gateway* durumunda iken alınan *Inner_Flood* (Çizelge A-4, Satır 10) mesajındaki bütçe değeri mesajı yollayan düğüme aynen geri yollanır ve başka işlem yapılmaz.

**Çizelge A-5:** *initiator* durum tablosu

| **AKTİF DURUM: INITIATOR** | | | |
|---|---|---|---|
| | **GİRDİ** | **ÇIKTI** | **SONRAKİ** |
| 1 | Poll_Neighbor | Poll_Neighbor_Reply | INITIATOR |
| 2 | Poll_Neighbor_Reply | NOP | INITIATOR |
| 3 | | Stop_Cluster_Extension | NODE |
| 4 | Poll_Neighbor_Reply_Tout | Inner_Flood | INITIATOR |
| 5 | Poll_Gateway | Poll_Gateway_Reply | INITIATOR |
| 6 | Config_Ack | Inner_Flood | INITIATOR |
| 7 | Gateway_Decisicion_Call | State_Reply (Opt.) | INITIATOR |
| 8 | Outer_Flood | NOP | INITIATOR |
| 9 | Inner_Flood | Reply_Budget | INITIATOR |

*Initiator* durumunda olan bir duyarga düğümü girdi olarak aldığı *Poll_Neighbor* (Çizelge A-5, Satır 1) ve *Poll_Gateway* (Çizelge A-5, Satır 5) mesajlarına durum bilgisi ve *ID* numarasını içeren *Poll_Neighbor_Reply* ve *Poll_Gateway_Reply* mesajları ile yanıt verir. Duyarga düğümü bu girdiler sonucunda durumunu değiştirmez. Protokole göre eğer bir düğüm daha önce

*floating* durumundan *initiator* durumuna geçmiş ise bu aşamada ilk olarak komşularını belirlemek için *Poll_Neighbor* mesajını bütüneyayar ve komşu belirleme zamanlayıcısını çalıştırır. Bu mesaja haberleşme menzilindeki duyarga düğümleri Poll_*Neighbor_Reply* mesajları ile yanıt verirler. Bu mesajlar yoklamayı yapan düğüm tarafından *initiator* durumunda iken alınır ve mesajda bulunan *ID* ve durum bilgileri komşu listesine kaydedilir, durum değiştirilmez ve bir çıktı oluşturulmaz (Çizelge A-5, Satır 2). Eğer cevap veren komşular içinde *initiator* durumunda olan bir düğüm var ise *ID* numaraları karşılaştırılarak küme genişlemesinin durdurulma veya devam etme kararı verilir. Gelen mesajdaki *ID* numarası büyük ise genişleme durdurulur ve durum *node* olarak değiştirilir (Çizelge A-5, Satır 3). Komşu belirleme zamanlayıcısı sıfırlandığında *Poll_Neighbor_Reply_Tout* (Çizelge A-5, Satır 4) mesajı alınır ve komşu listesine kaydedilen *floating* durumundaki duyarga düğümlerine bütçe taksimi yapılır. Burada bütçe tamamen kullanılıncaya veya bütçeyi kullanacak düğüm kalmayıncaya kadar bütçe dağıtımı yapılır. *Config_Ack* (Çizelge A-5, Satır 6) mesajı küme genişlemesi sırasında bütçe yollanan komşu düğümlerden alınır ve içeriğinde komşuların kendilerine ayrılan bütçenin ne kadarını kullanıdıklarına dair bilgi bulunur. Bütçe dağıtılan tüm komşulardan yanıt alındığında arta kalan bütçe varsa bu bütçe tekrar genişleme yapabilecek komşulara taksim edilir ve genişleme işlemi bütçe tamamlanana veya genişleyecek düğüm kalmayana kadar devam eder. *Gateway_Decision_Call* (Çizelge A-5, Satır 7) mesajı *initiator* durumundaki düğüm tarafından alındığında seçmeli olarak durum bilgisi içeren bir bilgi mesajı yollanır. Bu cevap protokol için zorunlu değildir ve sadece yoklama yapan düğüme güncel durum bilgisinin iletilmesi için yollanır. *Outer_Flood* (Çizelge A-5, Satır 8) mesajı durumu *initiator* olan düğümlere gönderilmez fakat daha önce *floating* olarak kaydedilmiş fakat daha sonra durumunu *initiator* olarak değiştirmiş düğüme durum güncellemesi yapılmamış olması nedeni ile yollanabilir. Bu durumda hiçbir işlem yapılmaz ve durum değiştirilmez. *Initiator* durumunda iken alınan *Inner_Flood* (Çizelge A-5, Satır 9) mesajındaki bütçe değeri mesajı yollayan düğüme aynen geri yollanır ve başka işlem yapılmaz.

# EK-B: Rastsal İkili Arama Ağacı

Aşağıdaki rastsal sayılar tanımlanmak üzere (Cormen et al., 2001);

- $X_n = n$ anahtar üzerinden rast gele oluşturulan ikili arama ağacının yüksekliği
- $Y_n = 2^{X_n} = Üstel\ yükseklik$
- $R_n =$ Ağaç kökünün *n* adet anahtar içerisindeki sırası
  - Eşit olasılıkla {1, 2, … , *n*}'den biri olabilir.
  - Eğer $R_n = i$ ise,
    - Sol ağaç parçası *i*-1 adet anahtardan rast gele oluşturulur.
    - Sağ ağaç parçası *n-i* adet anahtardan rast gele oluşturulur.

$\mathrm{E}[Y_n]$ ile $\mathrm{E}[X_n]$ ilişkisi Jensen eşitsizliği kullanılarak $\mathrm{E}[f(X)] \geq f(\mathrm{E}[X])$ olarak kurulur.

Eğer $R_n = i$ ise ağacın yüksekliği çocuklarının maksimum yüksekliğinin bir fazlasıdır.

Temel durumlar:

- $Y_1 = 1$ (1 düğümlü ağacın beklenen yüksekliği)
- $Y_0 = 0$

Gösterge rastsal değişkenler $\mathrm{Z}_{n,1}, \mathrm{Z}_{n,2}, ..., \mathrm{Z}_{n,n}$ aşağıdaki şekilde tanımlanırsa,

$\mathrm{Z}_{n,i} = \mathrm{I}\{R_n = i\}$ ve $R_n$ eşit olasılıkla {1, 2, … , *n*}'den biri olabilir;
$\Rightarrow \Pr\{R_n = i\} = 1/n \Rightarrow \mathrm{E}[Z_{n,i}] = 1/n$

Sadece bir tane $\mathrm{Z}_{n,i}$ 1'e eşit olabilir ve diğerleri 0'dır. Buna göre $Y_n$, aşağıdaki eşitlikle bulunabilir.

$$Y_n = \sum_{i=0}^{n} \mathrm{Z}_{n,i} (2 \cdot \max(Y_{i-1}, Y_{n-i})).$$

Burada $\mathrm{E}[Y_n]$'nin *n*'e üstel bağlı olduğunun gösterilmesi gerekmektedir.

$\mathrm{Z}_{n,i} = \mathrm{I}\{R_n = i\}$ gösterge rastsal değişkenleri $Y_{i-1}$ ve $Y_{n-i}$ değerlerinden bağımsızdır. $R_n = i$ seçili iken sol ağaç parçası sıra numaraları *i*'den küçük *i*-1

anahtar üzerine kurulur. Bu ağaç parçasının yapısı $R_n = i$'ye bağlı değildir çünkü $Y_{i-1}$ ve $Z_{n,i}$ bağımsız rastsal değişkenlerdir. Aynı şekilde $R_n = i$ seçili iken sağ ağaç parçası sıra numaraları *i*'den büyük *n-i* anahtar üzerine kurulur. Bu ağaç parçasının yapısı $R_n = i$'ye bağlı değildir çünkü $Y_{n-i}$ ve $Z_{n,i}$ bağımsız rastsal değişkenlerdir. Dolayısı ile,

$$\begin{aligned}
\mathrm{E}[Y_n] &= \mathrm{E}\left[\sum_{i=1}^{n} Z_{n,i}(2\cdot\max(Y_{i-1},Y_{n-i}))\right] \\
&= \sum_{i=1}^{n} \mathrm{E}[Z_{n,i}(2\cdot\max(Y_{i-1},Y_{n-i}))] \\
&= \sum_{i=1}^{n} \mathrm{E}[Z_{n,i}]\cdot\mathrm{E}[(2\cdot\max(Y_{i-1},Y_{n-i}))] \\
&= \sum_{i=1}^{n} \frac{1}{n}\cdot\mathrm{E}[(2\cdot\max(Y_{i-1},Y_{n-i}))] \\
&= \frac{2}{n}\sum_{i=1}^{n} \mathrm{E}[\max(Y_{i-1},Y_{n-i})] \\
&\le \frac{2}{n}\sum_{i=1}^{n} (\mathrm{E}[Y_{i-1}]+\mathrm{E}[Y_{n-i}])
\end{aligned}$$

Son basamaktaki toplama;

$$(\mathrm{E}[Y_0]+\mathrm{E}[Y_{n-1}])+(\mathrm{E}[Y_1]+\mathrm{E}[Y_{n-2}])+\ldots+(\mathrm{E}[Y_{n-1}]+\mathrm{E}[Y_0]) = 2\sum_{i=0}^{n-1}\mathrm{E}[Y_i]$$

Olup aşağıdaki sonucu verir;

$$\mathrm{E}[Y_n] \le \frac{4}{n}\sum_{i=0}^{n-1}\mathrm{E}[Y_i]$$

Buradan aşağıdaki eşitsizliğe ulaşılır.

$$\mathrm{E}[Y_n] \le \frac{1}{4}\binom{n+3}{3}, \quad n>0$$

Bu denklemde erişilen $Y_n$'in sınırı Jensen eşitsizliğinden $X_n$'in de sınırını verir.

$$2^{\mathrm{E}[X_n]} \le \mathrm{E}[2^{X_n}] = \mathrm{E}[Y_n]$$

$$2^{\mathrm{E}[X_n]} \le \frac{1}{4}\binom{n+3}{3} = \frac{n^3+6n^2+11n+6}{24}$$

İki tarafın da logaritması alındığında $X_n$ ağaç yüksekliğinin beklenen değerinin $\mathrm{E}[X_n] = O(\lg n)$ olduğu bulunur. □

# ÖZGEÇMİŞ

Ad Soyad: Özgür SAĞLAM
Doğum Yeri: SAMSUN
Doğum Tarihi: 11.11.1976
Uyruğu: TC

**Eğitim**

- Doktora: Ege Üniversitesi, Uluslararası Bilgisayar Enstitüsü, 2002 – 2009
- Yüksek Lisans: Ege Üniversitesi, Uluslararası Bilgisayar Enstitüsü, Ağ Teknolojileri Ana Bilim Dalı, 1998 – 2002
- Lisans: Dokuz Eylül Üniversitesi, Elektrik-Elektronik Mühendisliği Bölümü, 1993 – 1998

**İş Deneyimi**

- 2003-2009, Arçelik AŞ, Proje Lideri (Uzman Mühendis)
- 2002-2003, Sar Elektronik AŞ, Yazılım Geliştirme Mühendisi
- 2001–2002, Vestel Komünikasyon AŞ, Dijital-Arge, Proje Mühendisi
- 1998–2001, Aselsan AŞ, Mikrodalga ve Sistem Teknolojileri Grubu İzmir Araştırma Laboratuvarı, Elektrik-Elektronik Mühendisi
- 1997-1998, Aselsan AŞ, Mikrodalga ve Sistem Teknolojileri Grubu İzmir Araştırma Laboratuvarı, Geçici Teknik Eleman

**Ödüller**

- Dokuz Eylül Üniversitesi Mühendislik Fakültesi Elektrik Elektronik Mühendisliği Bölüm Üçüncülüğü Ödülü

**Yayınlar**

- Sağlam, Ö. and Dalkılıç, M. E., "A Self Organizing Multihop Clustering Protocol for Wireless Sensor Networks," *To appear in Proceedings of 5th International Conference on Mobile Ad-Hoc and Sensor Networks (MSN'09)*, Wu Yi Mountain, China, 14-16 December, 2009, IEEE, New York, NY.
- Sağlam, Ö. and Dalkılıç, M. E., "Cross Layer Implementation of Key Establishment Protocols in Wireless Sensor Networks," *In Proceedings of 24th International Symposium on Computer and Information Sciences (ISCIS'09)*, Güzelyurt, Northern Cyprus, 14-16 September, pp. 356-361, 2009, IEEE, New York, NY.
- Özgür Sağlam, Mehmet E. Dalkılıç and Kayhan Erciyeş, "Design and Implementation of a Secure Group Communication Protocol on a Fault Tolerant Ring", *Lecture Notes in Computer Science*, vol. 2869, pp. 802-810, 2003, Springer, Berlin.