Kablosuz İletişim Ağları Dersi

2022-2023 Bahar Dönemi

**PROJE RAPORU**

Teslim Tarihi: 25.05.2025

Öğrenci Adı: Mustafa Alp YANIKOĞLU

Öğrenci Numarası: 222802005

1. Giriş

Ödevin amacını, ağ topolojisindeki veri iletimini minimum maaliyet ile sonuçlandırmaktır.

Ağ Topolojisi Nedir?

Ağ topolojisi, bir bilgisayar ağındaki cihazlar ve bağlantılar arasındaki fiziksel veya mantıksal düzenlemeyi ifade eder. Bir ağ topolojisi, ağdaki cihazların nasıl bağlandığını ve veri iletişimini nasıl gerçekleştirdiğini belirler.

Ağ topolojisi, ağdaki cihazların fiziksel yerleşimini, bağlantıların nasıl yapılandırıldığını ve veri iletişiminin nasıl gerçekleştiğini tanımlar. Örneğin, bazı yaygın ağ topolojileri şunlardır:

Yıldız Topolojisi: Bu topolojide, tüm cihazlar merkezi bir noktaya bağlanır. Bu merkezi nokta genellikle bir anahtar veya bir yönlendirici olabilir. Her cihaz, merkezi noktaya tek bir bağlantı kullanarak diğer cihazlarla iletişim kurar. Yıldız topolojisi, bir cihazın arızalanması durumunda ağı etkileme riskini azaltır, çünkü diğer cihazlar etkilenmez.

Halka Topolojisi: Bu topolojide, her cihaz komşu cihazlarla bir halka oluşturacak şekilde bağlanır. Her cihaz, veriyi alır, kendi hedefine yönlendirir ve ardından halkanın bir sonraki cihazına iletir. Halka topolojisindeki bir cihazın arızalanması durumunda tüm ağ etkilenebilir.

Ağaç Topolojisi: Bu topolojide, cihazlar hiyerarşik bir yapıda düzenlenir. Bir kök düğüm (anahtar veya yönlendirici), alt düğümlerle bağlantıya geçer. Alt düğümler de diğer alt düğümler veya son düğümlerle bağlantı kurabilir. Ağaç topolojisi, büyük ağları yönetmek için etkili olabilir.

Mesh Topolojisi: Bu topolojide, her cihazın diğer tüm cihazlarla doğrudan bağlantısı vardır. Mesh topolojisi, yüksek ölçüde bağlantı ve yedeklilik sağlar, ancak çok sayıda bağlantı gerektirdiği için maliyetli olabilir.

Ağ topolojisi, veri iletişimini ve ağ performansını etkileyebilir. Doğru bir topoloji seçimi, ağın taleplerini karşılamak, verimlilik, ölçeklenebilirlik ve güvenilirlik gibi faktörleri göz önünde bulundurmak için önemlidir. Topoloji, ağdaki cihazlar arasındaki iletişimi düzenleyerek veri transferini optimize eder ve ağın genel performansını artırır.

ACOAlgorithm.cs’de kullanılan veriler:

List<int> FindShortestPath(): En kısa yolu bulmak için çalıştırılan ilk metotdur.

Void InitializePheromene(): Her bir hücreye atanacak başlangıç feromon değerini belirler. Bu değer, toplam düğüm sayısının tersine eşit olarak hesaplanır. Bu şekilde, başlangıçta her yol parçasının eşit bir feromon değerine sahip olması sağlanır.

Void CalculateDistance(): İki düğüm arasındaki mesafeyi hesaplar.

Void EvaporatePheromones(): Feromonların buharlaşmasını uygular. Feromon matrisindeki her bir hücrenin değerini belirli bir oranda azaltarak feromonların zamanla azalmasını sağlar.

List<int> ConstructAndPath(): Karıncanın tüm düğümleri gezerek bir yol oluşturmasını sağlar.

İnt ChooseNextNode(int currentNode, bool[] visited): Feromon miktarı ve mesafe değerlerine dayalı olarak her düğüm için bir olasılık hesaplar ve bu olasılıklara dayanarak bir sonraki düğümü seçer.

Double CalculatePathDistance(List<int> path): Yol boyunca her iki düğüm arasındaki mesafeleri toplar.

Void UpdatePheromone(List<List<int>> antPaths, double[] antDistances): Güncellenen feromon matrisi hesaplanır ve eski değeri ile ağırlıklı olarak karşılaştırılır.

Edge.cs’de kullanılan veriler:

Ağ yapısında iki düğüm arasındaki bir kenarı temsil eden sınıfını tanımlar.

Source: Kenarın başlangıç düğümünü temsil eden bir tamsayı değeri.

Target: Kenarın hedef düğümünü temsil eden bir tamsayı değeri.

LinkLabel: Kenarın ilişkilendirilmiş bir etiketi veya açıklaması olan bir dize değeri.

Node.cs’de kullanılan veriler:

Ağ yapısındaki düğümleri temsil etmek için kullanılır.

Label: Düğümün adını veya etiketini temsil eden bir dize değeri.

Country: Düğümün ilişkilendirildiği ülkeyi temsil eden bir dize değeri.

Longitude: Düğümün boylam koordinatını temsil eden bir ondalık sayı değeri.

Latitude: Düğümün enlem koordinatını temsil eden bir ondalık sayı değeri.

1. Geliştirme ve Çalıştırma Ortamları

Algoritmayı geliştirdiğiniz geliştirme ortamı(Visual Studio vs.) ve programlama dilini belirtiniz.

1. Algoritma Akışı

Bu kod, Karınca Kolonisi Optimizasyonu (Ant Colony Optimization - ACO) algoritmasını kullanarak bir ağ topolojisindeki verileri graf üzerinde en kısa yolu bulmayı amaçlar. Aşağıda kodun akışı hakkında bir açıklama bulunmaktadır:

ACOAlgorithm sınıfı tanımlanır ve gerekli sabit değerler ve değişkenler tanımlanır. ACOAlgorithm sınıfı, nodes ve edges adında iki liste parametresi alarak başlatılır. Bu parametreler, grafın düğümlerini ve kenarlarını içerir. FindShortestPath metodu, en kısa yolu bulmak için çağrıldığında çalışır. İlk olarak, feromon matrisi ve mesafe matrisi başlangıç değerleriyle başlatılır. Ardından, belirli bir sayıda iterasyon yapılır. Her iterasyonda, ConstructAntPath metodu kullanılarak her bir karınca için bir yol oluşturulur. CalculatePathDistance metodu kullanılarak oluşturulan yolların uzunlukları hesaplanır. En kısa yol ve uzunluğu güncellenir ve kaydedilir. UpdatePheromone metoduyla feromon matrisi güncellenir. EvaporatePheromones metoduyla feromonlar buharlaşır. En kısa yol ve uzunluğu içeren bestPath ve bestDistance değerleri döndürülür. Diğer yardımcı metotlar arasında, feromon matrisini başlangıç değerleriyle başlatan InitializePheromone, mesafe matrisini hesaplayan CalculateDistance, bir sonraki düğümü seçen ChooseNextNode, yolu hesaplayan CalculatePathDistance ve yolun feromonlarını güncelleyen UpdatePheromone bulunur.

1. Sonuç

Ağ topolojisinde paket iletim yolunu optimize etmek için ACO algoritmasını tercih ettik. Tercih sebeplerinden biri eğer paket aynı noktadan başlayıp aynı noktada bitecek ise karıncalar yolu ezbere bir şekilde hızla erişebilir. Tekrar hesaplanmasına gerek yok.

ACO algoritmasının bize sağladığı faydalar;

Ağ topolojisini optimize etmek için esnek bir çözüm sunar. Karıncaların feromon izlerini kullanarak yeni rotalar keşfetmesi ve en iyi rotayı belirlemesi sağlanır. Bu, ağ topolojisinin dinamik olarak değiştiği durumlarda bile etkili bir şekilde çalışabilmesini sağlar.

Paralel işlem yapabilmemizi sağlaması optimizasyon işlemini hızlandırır ve daha büyük ağlarda daha iyi performans elde etmeyi sağlayabilir.

İterasyonlar aracılığıyla feromon izleri güncellenmesi ağ topolojisini otomatik olarak optimize eder ve sürekli olarak iyileştirme sağlar. Uzun vadede, ağ performansının artması ve veri iletiminin daha verimli hale gelmesi beklenir.

Feromon buharlaşmasının kullanılması, ağ topolojisindeki değişikliklere hızlı tepki verilmesini ve yeni durumlara adapte olması sağlanır.

diyagram, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

çizgi, diyagram, öykü gelişim çizgisi; kumpas; grafiğini çıkarma, eğim, bayır içeren bir resim

Açıklama otomatik olarak oluşturuldu

Yukarıdaki şekillerde örnek birer sonuç verilmiştir Washington CDC’ den yola çıkan bir paketin Denver’a en az maaliyet ile gideceği yol gösterilmiştir. Her siyah nokta farklı bir koordinatı belirtmektedir bu koordinatlara aşağıdaki bağlantıdan ulaşabilirsiniz (<http://topology-zoo.org/files/Abvt.gml>).

Bağlantıda örnek bir ağ topolojisi verilmiştir. Bu veriler programa manuel olarak eklenerek test edilmiştir.