Kafka Java Project

Yusuf Özcan KARAMAN

1. Proje Gereksinimleri

Bu proje kapsamında, iki kule ve bir hareketli hedefin bulunduğu bir senaryo simüle edilmektedir. İlk kulede bir radar, ikinci kulede ise bir kamera bulunmaktadır. Radar, hareketli hedefin konum bilgilerini saniyede bir kez raporlayacak ve bu bilgileri Kafka topiğine yazacaktır. Kamera, bu konum bilgilerini alarak hedefe yönelmek için açısını ayarlayacaktır.

1.1.İstenilen Uygulamalar

world_simulator: Hedefin hareketini simüle eden uygulama.

radar_control: Radar tarafından hedefin konum bilgilerini işleyen uygulama.

camera_control: Kamera tarafından hedefin konum bilgilerini işleyen uygulama.

world_gui: Hedef ve kulelerin durumlarını görsel olarak gösteren JavaFX tabanlı arayüz uygulaması.

Geliştirme kapsamında world_gui ve world_simulator uygulamaları tek bir uygulama üzerinden gerçeklenmiştir. Hazırlanan yapı kapsamında incelemeler bu yaklaşımla incelenmelidir. Kafka ile haberleşme için Helper sınıflar olan KafkaProducerHelper ve KafkaConsumerHelper sınıfları hazırlanmıştır. Yazılan kodlar üzerinde önemli açıklamalar kod okuma kalitesini artırmak amacıyla kod bloğu üzerinden yorum satırlarıyla belirtilmiştir.

1.2.Kullanılan Teknolojiler

Proje içerisinde kod geliştirmesi Java ile yapılmıştır. Apache Kafka ile uygulamalar arası veri aktarımı sağlanmıştır. Docker aracılığıyla harici uygulamalar olan Kafka, Zookeeper ve AKHQ araçlarının çalıştırılması sağlanmıştır. Kullanıcı arayüzü için ise JavaFX kullanılmıştır.

2. SİSTEM MİMARİSİ

Sistem mimarisi, proje gereksinimlerinde verilen yapılara göre tasarlanmıştır. Tüm systemin haberleşme Kafka topikleri üzerinden gerçekleştirilmektedir. Figure 2-1 ile istenilen yapıya dair görsel aşağıda gösterilmiştir.

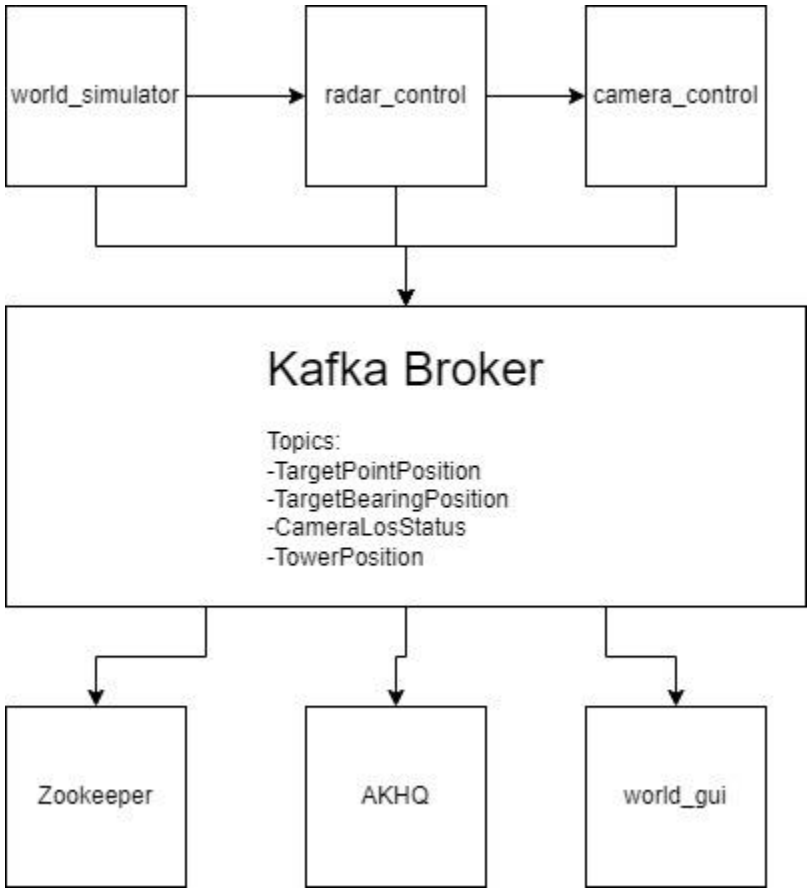

Figure 2-1 Sistem Mimarisi

Kodlama aşamasında yukarıda Figure2-1 ile gösterilen yapı içerisinde isterlerde onay verildiği üzere world_simulator ve world_gui uygulamaları birleştirilmiştir.

2.1.Proje Bileşenleri ve Rolleri

Proje birleştirilmiş world_gui ve world_simulator uygulamalarıyla beraber 4 ana Java ve harici destekleyici uygulamalardan oluşmaktadır.

2.1.1. world_simulator Uygulaması

Uygulamanın amacı hedefin hareketini simüle etmesi ve konum bilgisini Kafka'ya göndermesidir. Hedef koordinatları X ve Y olmak üzere rastgele üretilmektedir. Ardından Kafka'ya TargetPointPosition topiğine gönderilmektedir. Ekstra olarak kulelerin sabit pozisyonlarına dair bilgiler TowerPosition topiğine gönderilmektedir.

Hedefin hareketi sırasında meydana gelebilecek hatalar loglanmakta ve simulasyon durumu GUI üzerinden control edilmektedir.

2.1.2. radar_control Uygulaması

Uygulama hedefin konum bilgilerini alır ve ardından açısal pozisyon ve mesafe bilgilerini hesaplama işlemlerine başlar. Elde edilen sonuçlar Kafka'ya gönderilir. Kafka'dan alınan hedef konum bilgileri işlenir, açı ve mesafe hesaplanır ve TargetBearingPosition topiğine gönderilir.

Uygulama içerisinde geçersiz koordinat bilgiler gibi hatalı veriler kontrol edilmektedir ve ardından loglama işlemleri yapılmaktadır. Uygulamanın kapanmasının sağlıklı olması için shutdown hook kullanılmıştır.

2.1.3. camera_control Uygulaması

Radardan gönderilen açısal pozisyon bilgiler alınır ve kamera açısı güncellenir. TargetBearingPosition topiğinden alınan açı bilgisi kullanılarak kamera açısı güncellenmektedir. Ardından veriler CameraLosStatus topiğine gönderilmektedir.

Uygulama içerisinde hatalı açı verileri kontrol edilmektedir. Kontrol sonrası loglama işlemleri yapılmaktadır. Uygulamanın kapatılmasının sağlıklı olması için shutdown hook kullanılmıştır.

2.1.4. world_gui Uygulaması

Hedefin ve kulenin durumlarını grafiksel olarak gösteren kullanıcı arayüzüdür. JavaFX tabanlı bir yapı kullanılmıştır. Uygulama içerisinde hedef ve kulelerin pozisyonları ekranda gösterilmektedir. Kullanıcı hedefin hareketini Play ve Stop butonlarıyla durdurabilmektedir. Uygulama içerisinde loglama işlemleri yapılmaktadır.

2.1.5. Harici Bileşenler

Java üzerinde geliştirilen kodların yanında uygulamalar arası haberleşme için Kafka kullanılmaktadır. Bunun yanında Zookeeper ve AKHQ uygulamaları da sistem içerisinde çalışmayı sürdürmektedir. Zookeeper, Kafka'nın çalışması için gerekli olan koordinasyon

hizmetlerini sağlar. Kafka Broker, Uygulamalar arasında veri aktarımını sağlayan mesaj kuyruğu hizmeti sunar.

Topikler: TargetPointPosition, TowerPosition, CameraLosStatus, TargetBearingPosition.

2.2.Veri Akışı

world_simulator uygulaması, hedefin konum bilgilerini üretir ve TargetPointPosition topiğine gönderir. radar_control uygulaması, TargetPointPosition topiğinden gelen verileri alır, açı ve mesafe bilgilerini hesaplar ve TargetBearingPosition topiğine gönderir. camera_control uygulaması, TargetBearingPosition topiğinden gelen açı bilgilerini alır ve kameranın açısını günceller, ardından CameraLosStatus topiğine gönderir. world_gui uygulaması, tüm bu bilgileri alır ve grafiksel arayüzde gösterir.

3. KURULUM ve BAŞLATMA

Proje içerisindeki yazılımlar debug ortamların IDE kullanılarak ayağa kaldırılmıştır. İstenildiği takdirde Docker üzerinden çalıştırmak da mümkündür. Harici uygulamalar ise direkt Docker üzerinde çalıştırılmıştır. Proje dosyası içerisinde gönderilen yml uzantılı dosya ile kolayca harici uygulamalara dair kurulumlar gerçekleştirilebilir.

3.1. Docker Ortamının Hazırlanması

Docker Compose dosyası kullanılarak gerekli hizmetler başlatılır. Terminal veya komut satır arayüzünde Docker'ın kurulu olduğu varsayılan bir bilgisayarda "docker-compose up" komutu çalıştırılarak harici hizmetler çalışır hale getirilebilir. Bu komut aracılığıyla Zookeeper, Kafka ve AKHQ hizmetleri başlatılmış olur.

3.2.Kafka Topiklerinin Oluşturulması

Kafka uygulamlar arasında veri aktarımını sağlamaktadır. Projede bulunan topikler aşağıda verilmiştir.

- TargetPointPosition
- TowerPosition
- CameraLosStatus
- TargetBearingPosition

Kullanıcıya iki adet Docker Compose sunulmuştur. Birinde tüm topikler otomatik olarak script üzerinden oluşturulurken diğerinde Kafka ve Zookeeper' a ait container oluşturulduktan sonra manuel olarak topiklerin oluşturulması gerekmektedir.

3.3.Java Tabanlı Uygulamalar

Java tabanlı uygulamalar IDE üzerinden Run edilerek çalışır hale getirilebilr. Bu noktada kullanıcıyı karşılayan arayüz üzerinde Play tuşuna basılarak hedefte hareket sağlanır. Stop butonuyla ile de hareket sonlandırılır. Uygulama içerisinde veri akışları ve hatalar loglar üzerinden takip edilebilir.

3.4.AKHQ ile Kafka Yönetimi

AKHQ, Kafka kümelerini izlemek ve yönetmek için grafiksel bir arayüz sunar. AKHQ'ya erişmek için tarayıcıda http://localhost:8080 adresine gidilir. Varsayılan kullanıcı adı ve şifre

admin/admin'dir. AKHQ ile Kafka topiklerini izleyebilir, mesajları görebilir ve Kafka kümelerinin durumunu kontrol edebilir.

Kullanıcıya iki adet yml sunulmuştur. Birinden tüm harici sistemler otomatik olarak ayağa kaldırılırken diğerinde farklı uygulamalar farklı yml dosyalarına bölünmüş ve topik oluşturma işlemleri manuelleştirilmiştir. Bunun nedeni Docker üzerinde karşılaşılan problemlerde hazırlanan iki adet çözüm yolu olduğundadır. Geliştirmeler esnasında Kafka, Zookeeper için bir docker compose dosyası hazırlanırken AKHQ için ayrıca bir docker compose dosyası oluşturulmuş ve topik oluşumları manuel olarak yapılmıştır.

Projenin bitiminde ise tüm bu sistemlerin otomatize edilmesi amaçlanmış ve tek bir yml dosyası üzerinde tutulmuştur. Burada bellek yetersizliğinden kaynaklanan çeşitli problemlerle karşılaşılmıştır ancak detaylı test etme adına süre olmadığından sorun çıkması durumunda docker loglarından bakılması gerekmektedir.

4. ÖNLEMLER ve ÖNEMLİ NOKTALAR

Hata İşleme ve Loglama adına tüm uygulamalarda olası hatalar kontrol edilmekte ve loglanmaktadır. Bu, hata ayıklama ve izleme süreçlerini kolaylaştırmaktadır. Thread Yönetimi adına uygulamalarda kullanılan thread'ler düzgün bir şekilde yönetilmektedir. Uygulamaların düzgün kapanması için shutdown hook kullanır. Kaynak Yönetimi için Kafka producer ve consumer nesneleri düzgün bir şekilde kapatılır. Bu, kaynakların doğru bir şekilde yönetilmesini sağlamaktadır. Kapsayıcı Sağlığı için Docker Compose dosyasında sağlık kontrolleri eklenmiştir. Bu, hizmetlerin sağlıklı çalışıp çalışmadığını kontrol edilmekte ve gerektiğinde yeniden başlatılmaktadır.

Bu proje, iki kule ve bir hareketli hedefin bulunduğu bir senaryoyu simüle eder. Hedefin konum bilgileri world_simulator uygulaması tarafından üretilir, radar_control uygulaması tarafından işlenir ve camera_control uygulaması tarafından kameranın açısı güncellenir. Tüm bu süreçler Kafka aracılığıyla gerçekleşmekte ve AKHQ ile izlenebilmektedir.