

# BİLGİSAYAR AĞLARI

## YMK-945

---

MEHMET ARDA HAKBİLEN

YUNUS EMRE ASLAN

ÖZGÜL YAREN ARSLAN

ZEYNEP TUANA ZENGİN

# SENTELFS

DÖNEM PROJESİ PROJE SUNUMU

NEO

SENTELFS **NEO**

DISTRIBUTED SYNC  
ENGINE

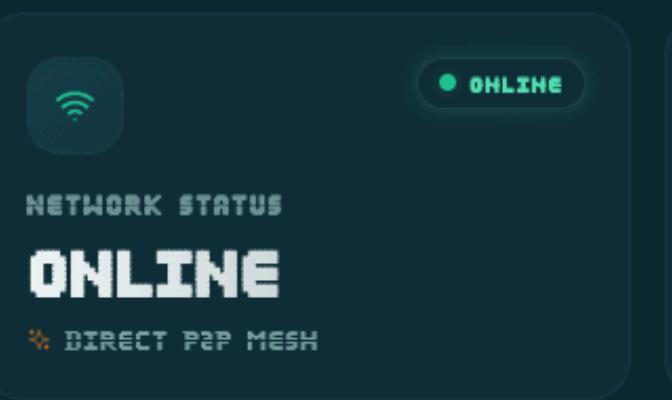
OVERVIEW

STATISTICS

LOGS

SETTINGS

X DASHBOARD



# Defining the Problem

## Temel Sorun

Güncel dosya senkronizasyon çözümlerinin çoğu:

- Tam dosya kopyalama yapıyor → büyük dosyalarda gereksiz bant genişliği ve disk I/O.
- Merkezi mimariye bağımlı → tek nokta hatası, zor ölçeklenme, kapalı kutu tasarım.
- Çoğu ürün, gömülebilir bir “sync engine kütüphanesi” değil; uçtan uca bir uygulama.

SentinelFS-Neo’nun çözmeye çalıştığı temel problem, geliştiricilerin projelerine kolayca entegre edebileceği, hafif, P2P ve delta-bazlı bir senkronizasyon motorunun eksikliği.

# Defining the Problem

## Mevcut Durum

Piyasadaki pek çok sync aracı:

- Monolitik: Uygulamaya gömülü, modüler bir C++ kütüphanesi olarak kullanmak zor.
- Yüksek bağımlılıklı: Büyük runtime'lar, karmaşık konfigürasyonlar ve güçlü makineler gerektirebiliyor.
- Şeffaf olmayan mimari: Delta hesaplama, metadata yönetimi, ağ topolojisi gibi katmanlar genelde içerde saklı.

SentinelFS bu tabloya,

- modüler daemon + plugin mimarisi,
- SQLite tabanlı atomik metadata,
- Adler32/SHA-256 delta sync,
- UDP discovery + TCP fallback ve auto-remesh
- gibi bileşenlerle, geliştirici dostu ve üretim seviyesinde bir alternatif getirmeyi hedefliyor.

# Defining the Problem

## Zorluklar

Bu tür bir engine tasarlarken öne çıkan zorluklar:

- Deterministik durum yönetimi: Plugin'lerin event bus üzerinden konuşurken tutarlı state üretmesi, her adımda explicit status code döndürmesi.
- Ağ koşulları: RTT, jitter ve packet-loss sürekli değişirken, P2P mesh topolojisini sağlıklı tutmak (auto-remesh, yeniden deneleme).
- Performans: Adler32 + SHA-256 kullanan delta engine'i CPU-bound olduğu için thread pool ile verimli çalıştmak.
- Platform çeşitliliği: Linux/macOS/Windows'ta farklı file watcher API'lerini tek bir event modeli altında birleştirmek.
- Güvenlik: AES-256 şifreleme ve session code ile özel mesh'leri korurken bant genişliği ve gecikme maliyetini düşük tutmak.

# Vision and Mission

## Vizyonumuz

SentinelFS-Neo'nun vizyonu, dağıtık uygulamalar için varsayılan sync motoru olmak.

Geliştiricilerin kendi sistemlerine gömebilecekleri,

- plugin tabanlı (DaemonCore, PluginLoader, INetworkAPI, IFileAPI, IStorageAPI),
- event-driven (EventBus, EventHandlers),
- delta-bazlı (DeltaEngine, DeltaSyncProtocolHandler)

hafif bir C++ senkronizasyon katmanı sunmak.

# Vision and Mission

## Misyonumuz

üretim ortamlarında çalışabilecek şekilde:

- Dosya değişikliklerini file watcher + EventHandlers ile anında yakalayan,
- FileSyncHandler ve DeltaEngine ile sadece değişen blokları senkronize eden,
- SQLite tabanlı storage plugin ile tüm metadata'yi tutarlı ve atomik yöneten,
- Ağ tarafında P2P peer discovery, auto-remesh ve güvenli veri aktarımını sağlayan

modüler bir daemon inşa etmek ve bunu geliştiriciler için kolay entegre edilebilir bir sync engine olarak sunmak.

# Key Objectives

- Modüler daemon ve plugin yapısı kurmak
  - DaemonCore + plugin loader üzerinden network, filesystem, storage, ML Anomali Tespiti gibi bileşenleri ayrı .so'lar olarak yükleyebilmek.
  - Arayüzler (INetworkAPI, IFileAPI, IStorageAPI) sayesinde her bileşeni bağımsız geliştirebilmek.
- Event-driven ve deterministik bir sync boru hattı sağlamak
  - EventBus + EventHandlers ile FILE\_MODIFIED, DATA\_RECEIVED, PEER\_DISCOVERED vb. olayları gevşek bağlı hale getirmek.
  - Tüm akışı log ve metriklerle (Logger, MetricsCollector) gözlemlenebilir kılmak.

# Key Objectives

- Delta bazlı, verimli dosya senkronizasyonu yapmak
  - FileSyncHandler ile değişen dosyaları tespit edip veritabanına işlemek.
  - DeltaEngine + DeltaSyncProtocolHandler ile sadece değişen blokları (Adler32 + SHA-256) hesaplayıp chunk bazlı taşımak.
- Ağ ve bağlantı sorunlarına dayanıklı olmak
  - Offline/peer yokken gelen değişiklikleri OfflineQueue ve pendingChanges\_ ile kuyrukta tutmak, peer'ler döndüğünde tekrar denemek.
  - Peer bağlantılarını event'ler (PEER\_CONNECTED, PEER\_DISCONNECTED) üzerinden yöneterek mesh'i sağlıklı tutmak.

# Key Objectives

- Konfigürasyon ve durumu güvenli şekilde yönetmek
  - Config gibi thread-safe yapıların üzerinde, daemon'ın çalışma parametrelerini güvenli biçimde saklamak.
  - Metadata'yı (dosya listesi, sync durumu vb.) veritabanı katmanında tutarlı ve atomik güncellemek.

# Market Research Findings

## 1. Geliştirilebilir “sync engine” neredeyse yok

- Kod tabanındaki DaemonCore, plugin arayüzleri (INetworkAPI, IFileAPI, IStorageAPI) ve bağımsız plugin yükleme mantığı, pazar analizinde gördüğümüz bir boşluğa yanıt veriyor:
- Çoğu çözüm uçtan uca ürün halinde geliyor; gömülebilir, C++17 tabanlı, plugin’lenebilir bir daemon olarak kullanabileceğin hafif “sync engine” neredeyse yok. Geliştiriciler, var olan sistemlerine küçük bir daemon eklemek yerine, koca bir ürünü adapte etmek zorunda kalıyor.



# Market Research Findings

## 2. Delta-sync var, ama şeffaf ve özelleştirilebilir değil

- DeltaEngine, DeltaSyncProtocolHandler ve açık metin protokol mesajları (UPDATE\_AVAILABLE|..., DELTA\_DATA|...) şunu gösteriyor:
- Piyasadaki pek çok araç delta-sync yapıyor olsa da, nasıl yaptıkları şeffaf değil ve geliştiricinin bu süreci inceleyip uyarlaması zor. Block boyutu, checksum stratejisi (Adler32 + SHA-256), chunk transfer şekli gibi kararlar çoğu zaman kapalı kutu. SentinelFS, bu süreci kod seviyesinde görünürlük ve değiştirilebilirliği hedefliyor.



# Market Research Findings

## 3. Dağıtık senaryolarda esneklik ve gözlemlenebilirlik eksik

EventBus, EventHandlers, OfflineQueue, MetricsCollector gibi bileşenler, pazar tarafında gördüğümüz iki eksikliğe cevap veriyor:

- Esneklik: Çoğu ürün offline senaryoları, geç bağlanan peer'leri veya mesh topolojisinin dinamik değişimini iyi yönetemiyor. Kodda ise OfflineQueue, pending değişiklik kuyrukları ve peer event'leriyle bu durum doğrudan adreslenmiş.
- Gözlemlenebilirlik: Birçok aracın sync sürecinin içini metrik ve log seviyesinde takip etmek zor. Burada ise metrik toplayıcı, detaylı log'lar ve IPC ile dışarıya açılan bir kontrol yüzeyi tasarlanmıştır.



# SCOPE DEFINITION

## 01 Included Elements

- P2P delta sync engine
  - DeltaEngine, DeltaSyncProtocolHandler, FileSyncHandler, EventHandlers
  - Adler32 + SHA-256 tabanlı blok seviyesinde delta hesaplama ve uygulama
- Plugin tabanlı daemon mimarisi
  - DaemonCore + plugin loader
  - Network, filesystem, storage, ML modüllerinin ayrı plugin'ler olarak yüklenmesi
- Event-driven işleyiş
  - EventBus üzerinden FILE\_MODIFIED, DATA\_RECEIVED, PEER\_DISCOVERED vb. event'lerin yönetilmesi
- Metadata ve konfigürasyon yönetimi
  - SQLite üzerinde dosya, peer, sync kuyruğu, log tabloları
  - Config ile thread-safe key/value konfigürasyon deposu

# SCOPE DEFINITION

## 02 Excluded Elements

- Tam entegre “cloud storage” ürünü
  - Sunucu taraflı çok-kiralıklı (multi-tenant) bulut dosya sistemi bu projenin kapsamı değil.
- Kapsamlı kullanıcı/rol yönetimi
  - Kullanıcı hesap sistemi, yetki hiyerarşisi, billing vb. yok; odak sync engine.
- İleri seviye çatışma çözüm UI’ları
  - GUI’de detaylı merge/konflikt arayüzleri yerine, core’da basit conflict handling mantığı.
- Mobil ve tarayıcı tarafı client’lar
  - Şu anda hedef masaüstü/daemon senaryoları; mobil/web SDK kapsam dışı.

# SCOPE DEFINITION

## O3 Assumptions

- Ortak dosya sistemi semantiği
  - Linux/macOS/Windows native FS API'lerinin temel davranışları biliniyor ve benzer kabul ediliyor.
- Güvenli ağ ortamı + ek katmanlar
  - AES-256 + session key kullanılıyor, ancak firewall, VPN gibi ek güvenlik katmanları uygulama ortamına bırakılmış.
- Geliştirici odaklı kullanım
  - Kullanıcı kitlesi geliştiriciler; C++17, SQLite, OpenSSL gibi bağımlılıkları yönetebilecekleri varsayılıyor.
- Sürekli çalışan daemon modeli
  - Senkronizasyonun bir “service/daemon” olarak, uzun süre çalışan bir süreçte yönetildiği kabul ediliyor.

# SCOPE DEFINITION

## 04 Constraints

- Tek metadata kaynağı olarak SQLite
  - Bütün metadata güncellemleri transaction içinde, başka DB motoru desteklenmiyor.
- C++17 ve düşük bağımlılık
  - Kod C++17 ile derleniyor; ek kütüphaneler minimumda tutuluyor (OpenSSL, SQLite, ONNX opsiyonel).
- Event-driven, busy-loop yasak
  - Async işleyiş event odaklı; spin-wait veya aktif bekleme tasarım açısından kabul edilmıyor.
- Güvenlik ve ağ kuralları
  - P2P discovery: UDP + TCP fallback
  - Tüm network mesajları struct-bazlı ve AES-256 ile korunmak zorunda.

# Architecture Overview

## Genel Yapı

- **Daemon (sentinel\_daemon.cpp)**
  - CLI argümanlarını okur, DaemonConfig oluşturur.
  - DaemonCore, EventBus, EventHandlers, IPCHandler ve auto-remesh thread'lerini başlatır.
- **DaemonCore**
  - PluginLoader üzerinden dynamic .so plugin'lerini yükler.
  - Network, filesystem, storage ve ML plugin pointer'larını tutar.
  - Çalışma döngüsünü ve düzgün kapanışı (shutdown) yönetir.
- **Plugin Katmanı**
  - **Network Plugin** → INetworkAPI: peer discovery, bağlantılar, veri gönderme/alma.
  - **Filesystem Plugin** → IFileAPI: inotify/FSEvents/ReadDirectoryChangesW ile FS event'leri.
  - **Storage Plugin** → IStorageAPI: SQLite üzerinde dosyalar, peer'ler, kuyruklar, loglar.
  - **ML Plugin** → IPlugin: ONNX tabanlı anomaly detection.

# Architecture Overview

## Event-Driven Sync Boru Hattı

- **EventBus**
  - *PEER\_DISCOVERED, FILE\_MODIFIED, DATA RECEIVED, ANOMALY\_DETECTED* gibi event'ler için pub/sub.
  - Daemon bileşenleri arasındaki bağı gevsetir (loosely coupled).
- **EventHandlers**
  - EventBus'a abone olur; gelen event'leri doğru handler'a yönlendirir.
  - FileSyncHandler ve DeltaSyncProtocolHandler ile birlikte çalışır.
  - Sync enable/disable, offline queue, pending değişiklikler gibi durumları yönetir.
- **FileSyncHandler & DeltaSyncProtocolHandler**
  - FileSyncHandler: FS event'lerinden gelen dosyaları hash'ler, SQLite'a yazar, *UPDATE\_AVAILABLE* yayar.
  - DeltaSyncProtocolHandler: *UPDATE\_AVAILABLE / REQUEST\_DELTA / DELTA\_DATA / FILE\_DATA / DELETE\_FILE* mesajlarını işleyerek DeltaEngine üzerinden dosya delta'larını hesaplar ve uygular.

# Architecture Overview

## Konfigürasyon ve Durum Yönetimi

- **Config (core/utils/Config.h)**
  - Thread-safe key/value konfigürasyon deposu, daemon'ın çalışma parametrelerini tutar.
- **SQLite Tabanlı Metadata**
  - Dosya listesi, versiyon, sync kuyruğu, peer bilgisi, file access log gibi tablolar.
  - Tüm güncellemeler transaction içinde, tek gerçek kaynak (single source of truth).

# Project Timeline



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

## Planning Period

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

## Development Stage

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

## Testing Phase

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

# Project Team and Responsibilities

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

## Project Lead

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

## Core Team

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.