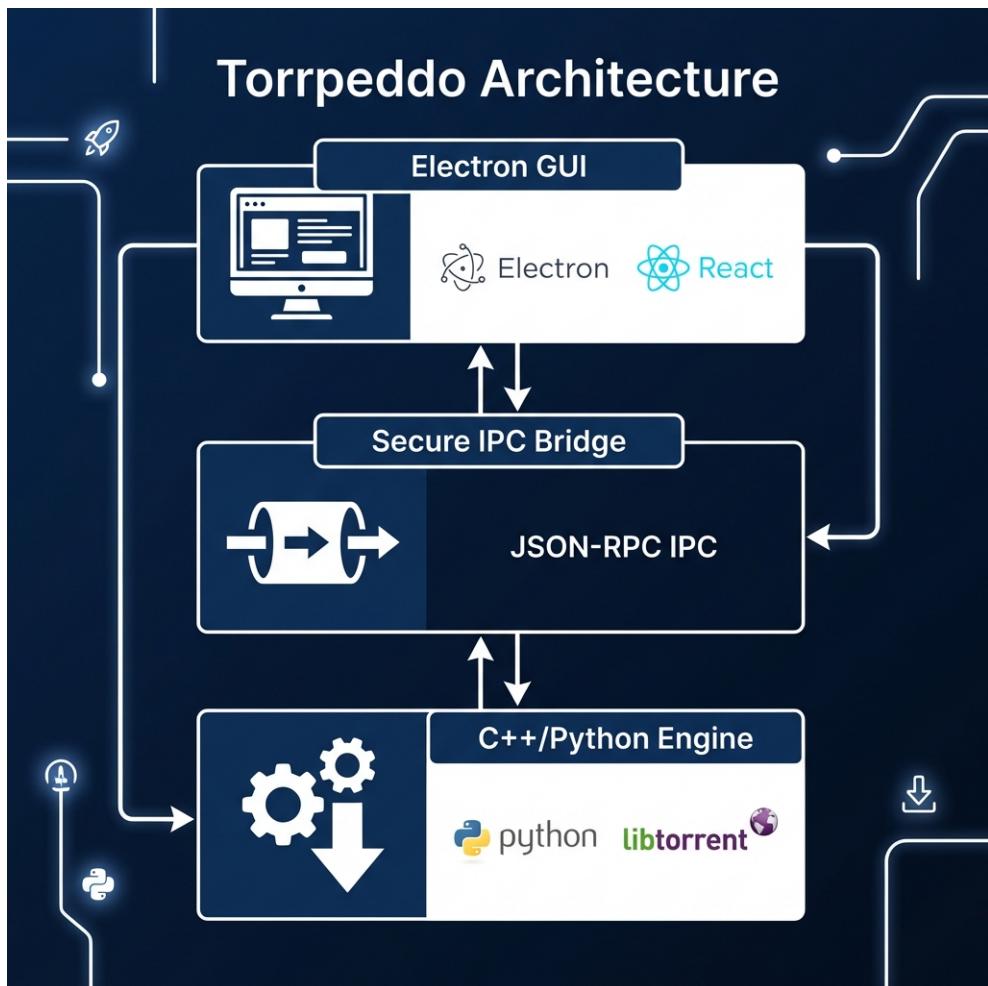


# TORRPEDDO PROJECT BOOK



## Executive Summary

## Architectural Deep Dive

Torrpedo follows a decoupled architectural pattern, separating the presentation layer from the core logic and network engine. This is achieved through three primary layers:

## **1. Frontend: Electron Framework**

### **Overview: What is Electron?**

#### **Benefits for Torpeddo**

## **2. The Bridge: IPC (Inter-Process Communication)**

### **Concept: What is IPC?**

IPC, or Inter-Process Communication, is a mechanism that allows different processes to share data and coordinate actions. In Torpeddo, we use a custom IPC bridge to connect the Electron frontend with the Python backend.

### **Implementation: Secure JSON-RPC**

Communication is handled via a JSON-RPC protocol over stdin/stdout channels. - The Electron process spawns a dedicated Python child process. - Commands (e.g., add\_magnet, get\_status) are serialized into JSON strings and sent to the Python process. - The Python process executes the logic and returns a structured JSON response.

### **Advantages of the IPC Bridge**

- Decoupling: The engine can be updated, debugged, or even replaced without touching the UI.
- Security: The backend is isolated from the frontend.

## **3. Backend Engine: Python & libtorrent**

### **The Core: libtorrent with Python Bindings**

At the heart of Torpeddo is libtorrent, a feature-complete BitTorrent implementation. While the underlying engine is implemented in high-performance C++, Torpeddo utilizes the official Python bindings for

rapid development and seamless integration with the bridge logic.

### **Multi-threaded Performance**

- Engine Level: The libtorrent 2.0+ engine utilizes an internal thread pool for disk I/O, network polling, and piece validation. This allows for parallel processing of multiple torrent fragments simultaneously.

- Manager Level: The manager handles non-blocking operations and stalls while metadata is being loaded.

---

## **Development Process & Methodology**

The Torrpedo project followed a "Platform-First" methodology:

### **Phase 1: Language Choice**

Python was selected for its extensive library support and ease of integration with libtorrent and IPC protocols.

### **Phase 2: Engine Validation**

Rigorous testing of libtorrent benchmarks to ensure maximum throughput on varied hardware.

### **Phase 3: Bridge Optimization**

Implementation of non-blocking I/O in the IPC bridge to prevent UI "micro-stutters".

### **Phase 4: Packaging & Distribution**

Integration of electron-builder and PyInstaller to create unified, single-binary distributors for end-users.

---