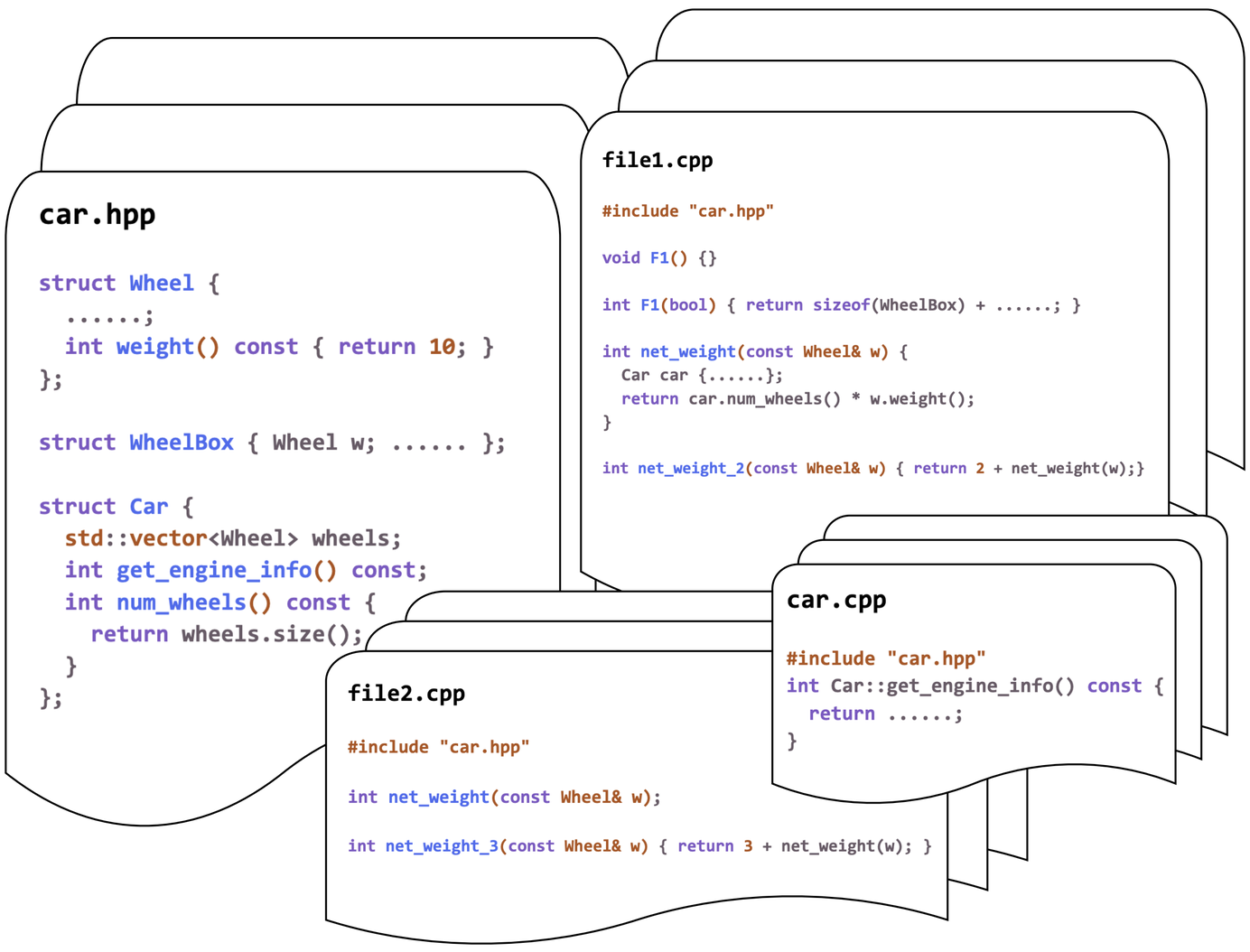
# CTwik : General Purpose Hot Patcher For C++

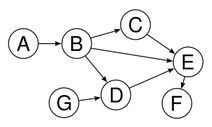
A tool for hot patching changed C++ code on a running process without restart




CTwik-Client:
Source Code Repo Watcher

Git diff

```
 struct Car {
-  std::vector<Wheel> wheels;
+  std::unordered_map<int, Wheel> wheels;
   int get_engine_info() const;
   int num_wheels() const {
```




Using Dependency Graph Among Global Entities Per TU

Minimum Impacted Area to Recompile

- ✗ F1 : Recompilation **Not** Required
- ✓ net_weight : Recompilation **Required**
- ✓ net_weight_2 : Recompilation **Required**
- ✗ net_weight_3 : Recompilation **Not** Required
- ✓ Car::get_engine_info: Recompilation **Required**

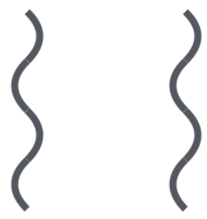
Impacted Area as C++ Code

**patch_001.cpp**

```
struct Wheel { ..... };

struct Car { ..... };

int net_weight(const Wheel& w) {
  Car car {.....};
  return car.num_wheels() * w.weight();
}

int net_weight_2(const Wheel& w) {
  return 2 + net_weight(w);
}

int Car::get_engine_info() const {
  return .....;
}
```

g++ -shared -fPIC

File: "**patch_001.so**"

| Main Application |
|---|
| Machine Code ; Data |
| Other Threads  **CTwik Thread**<br>- Listen to CTwik-client for patch requests<br>- **dlopen** "patch_001.so"<br>- **Change the machine code** of **this process (self )**<br>- Specially, override the old function's machine code's first 16 bytes to new instruction - "***long jump** to a new function-pointer*".  |

## Hot Patching Mechanism:

**Main Executable ( Possibly Statically Linked )**

```
0000000001433 <net_weight(Wheel const&)>:
 1433:       55                      push   %rbp
 1434:       48 89 e5                mov    %rsp,%rbp
 1437:       53                      push   %rbx
 1438:       48 83 ec 38             sub    $0x38,%rsp
 143c:       48 89 7d c8             mov    %rdi,-0x38(%rbp)
 1440:       48 8d 45 d0             lea    -0x30(%rbp),%rax
 .....

00000000011c1 <G()>:
 11c1:       55                      push   %rbp
 11c2:       48 89 e5                mov    %rsp,%rbp
 11c5:       48 83 ec 10             sub    $0x10,%rsp
 11c9:       48 8d 45 ff             lea    -0x1(%rbp),%rax
 11cd:       48 89 c7                mov    %rax,%rdi
 11d0:       e8 5e 02 00 00          callq  1433 <net_weight(Wheel const&)>
 11d5:       69 c0 c8 00 00 00       imul   $0xc8,%eax,%eax
 11db:       83 c0 04                add    $0x4,%eax
 11de:       c9                      leaveq
 11df:       c3                      retq
```

Long Jump to 0x01600

**patch_001.so**

```
0000000001600 <net_weight(Wheel const&)>:
 1600:       55                      push   %rbp
 1601:       48 89 e5                mov    %rsp,%rbp
 1604:       53                      push   %rbx
 1605:       48 83 ec 58             sub    $0x58,%rsp
 1609:       48 89 7d a8             mov    %rdi,-0x58(%rbp)
 160d:       48 8d 45 b0             lea    -0x50(%rbp),%rax
 .....
 .....

 1644:       48 8b 5d f8             mov    -0x8(%rbp),%rbx
 1648:       c9                      leaveq
 1649:       c3                      retq
```

**patch_002.so**

.....
.....

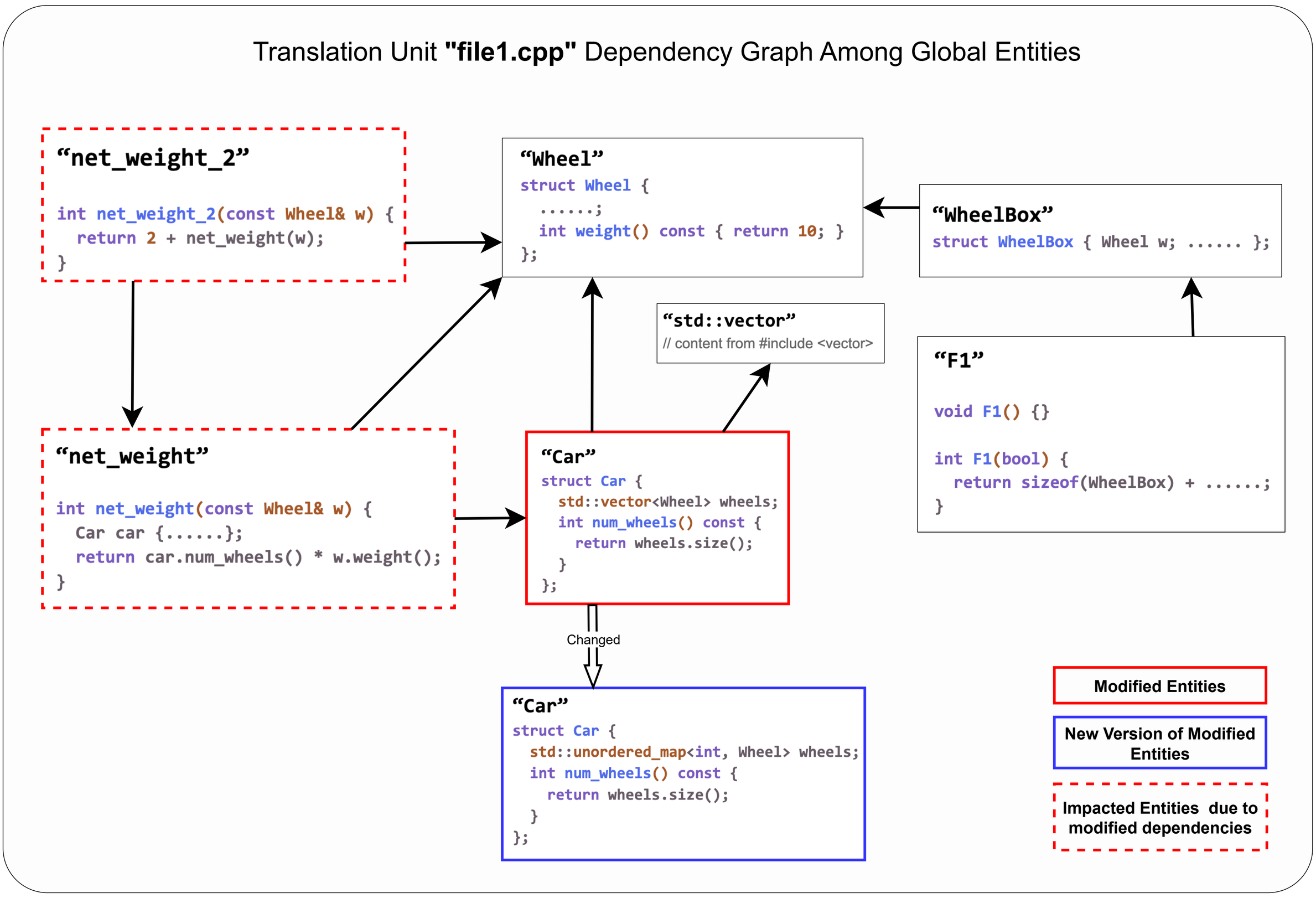
**Thread Safety**: CTwik-Server requires that none of the functions being hot patched are present in any thread's call stack. Failing this condition results in undefined behavior.

**Patching Protocol**: Hot patching must occur after write-acquiring a read-write-lock. This lock must be placed at the very top level entry points of the application. During the patching time, all new user API calls must be blocked until patching completes. Conversely, hot patching must wait for ongoing API calls to finish before proceeding.

**Limitation**: CTwik cannot be used to hot patch the "main" function or any top-level functions outside the mutex-protected section.

# CTwik : General Purpose Hot Patcher For C++

A tool for hot patching changed C++ code on a running process without restart




- **Impacted Entities** = **Inverse_Dependency_Cover** of Modified Entities.
- **Impacted Functions** = Set of Functions out of Impacted Entities.
- **Exportable Entities for Patching** = **Dependency_Cover** of Impacted Function.
  - This is required for compiling the impacted functions.
- **patch.cpp content** = Entities dumped in their original order ( Intentionally avoiding topological sorting ).

Here, **Dependency_Cover** is the **transitive closure** in the directed graph formed by dependency edges..

Similarly **Inverse_Dependency_Cover** is **transitive closure** in the directed graph formed by inverse-of-dependency-edges.

| | file1.cpp | file2.cpp | car.cpp |
|---|---|---|---|
| Modified Entities | • Car | • Car | • Car |
| Impacted Entities | • Car<br>• net_weight<br>• net_weight_2 | • Car | • Car<br>• Car::get_engine_info |
| Impacted Functions | • net_weight<br>• net_weight_2 | None | • Car::get_engine_info |
| Entities For Patch | • std::unordered_map<br>• Wheel<br>• Car<br>• net_weight<br>• net_weight_2 | None | • std::unordered_map<br>• Wheel<br>• Car<br>• Car::get_engine_info |

### Why CTwik

- **Faster Development**: Cuts down build times dramatically, turning hours or minutes into seconds, and often achieving up to **50x speedups**, especially in large C++ codebases.
- **Production Fixes Without Downtime**: Apply urgent patches on-the-fly.
- **Avoids in-memory state loss**: By skipping app restarts.
- **Accelerated Debugging**: Rapid iteration during development cycles.
- **CTwik in production**: We have been using it for years. It supports more than 95% of the code changes we encounter in our day-to-day development.

bit.ly/ctwik