30

```
class safe_thread
{
public:
    template<typename F, send... Args>
    safe_thread (F&& f, Args&&... args)
        : thread (std::forward<F> (f), std::forward<Args> (args)...)
    {
        // N.B. We can't constrain F to the concept due to recursion of is_move_constructable
        // So we have to statically assert it
        static_assert (send<F>);
    }

    safe_thread (safe_thread&& other)
        : thread (std::move (other.thread))
    {
    }

private:
    std::jthread thread;
};
```

```
class safe_thread
{
public:
    template<typename F, send... Args>
    safe_thread (F&& f, Args&&... args)
        : thread (std::forward<F> (f), std::forward<Args> (args)...)
    {
        // N.B. We can't constrain F to the concept due to recursion of is_move_constructable
        // So we have to statically assert it
        static_assert (send<F>);
    }

    safe_thread (safe_thread&& other)
        : thread (std::move (other.thread))
    {
    }

private:
    std::jthread thread;
};
```

Send in C++: *Moved between threads*

```
template<typename F, send... Args>
safe_thread (F&& f, Args&&... args)
    : thread (std::forward<F> (f), std::forward<Args> (args)...)
{
    static_assert (send<F>);
}
```