Lock-free Signal/Slots

yupeng.zhang@autodesk.com

https://git.autodesk.com/media-and-entertainment/evaluation-manager

Signal

```
vector<function<void(Args...)>>
```

Slot

```
vector::iterator
```

Interface

```
template <typename Args…>
class Signal {
  Slot connect(std::function<void(Args…)>);
  void emit(Args&&…);
};

class Slot {
    ~Slot();
    void disconnect();
};
```

Example

```
Slot slot = signal.connect([](Args&&...){...});

signal.emit(args...);

slot.disconnect();
```

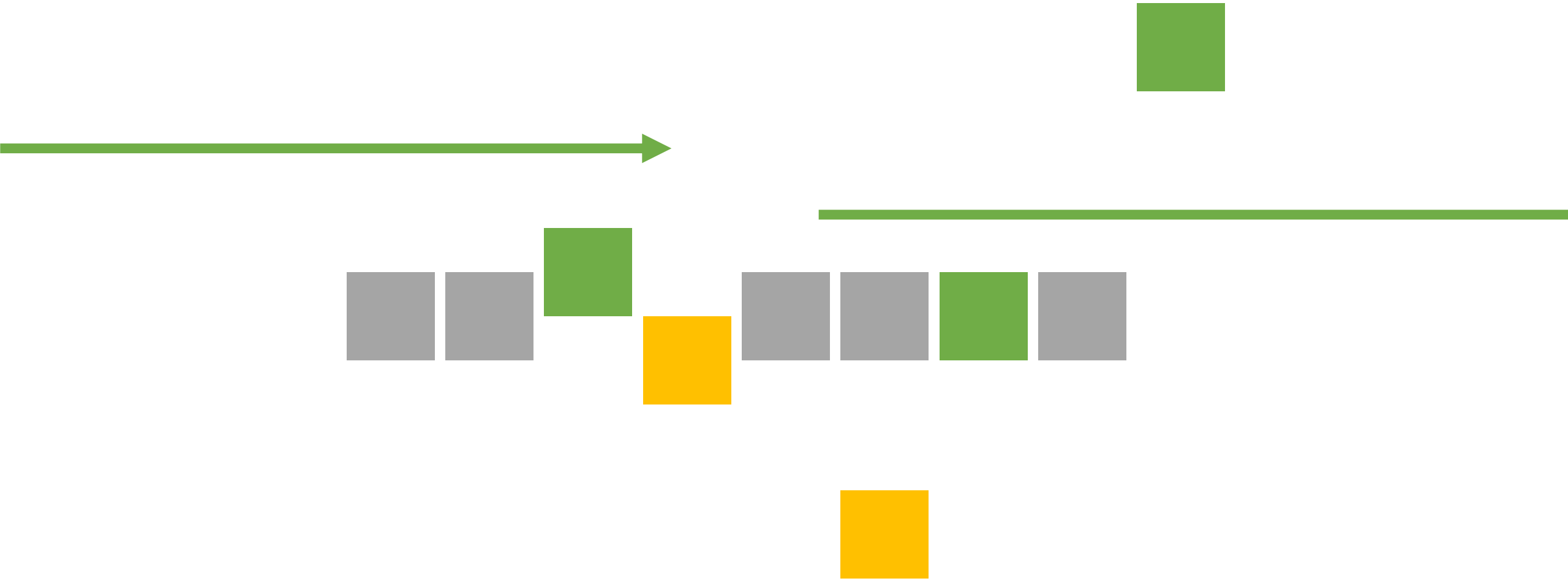
Recursive

```
Slot slot = signal.connect([&](Args&&...){
    ...
    Slot slot2 = signal.connect([](Args&&...){...});
    slot.disconnect();
    signal.emit();
});

signal.emit();
```

Thread safety

Thread safety Lock-free, Block-free

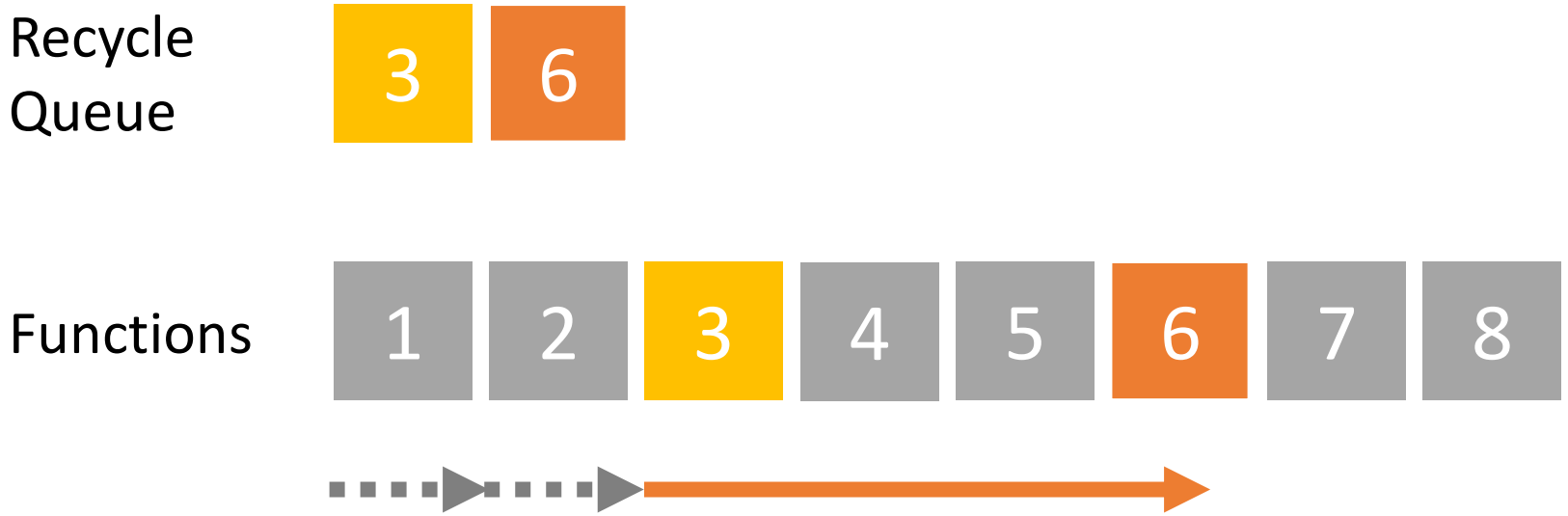


Emit

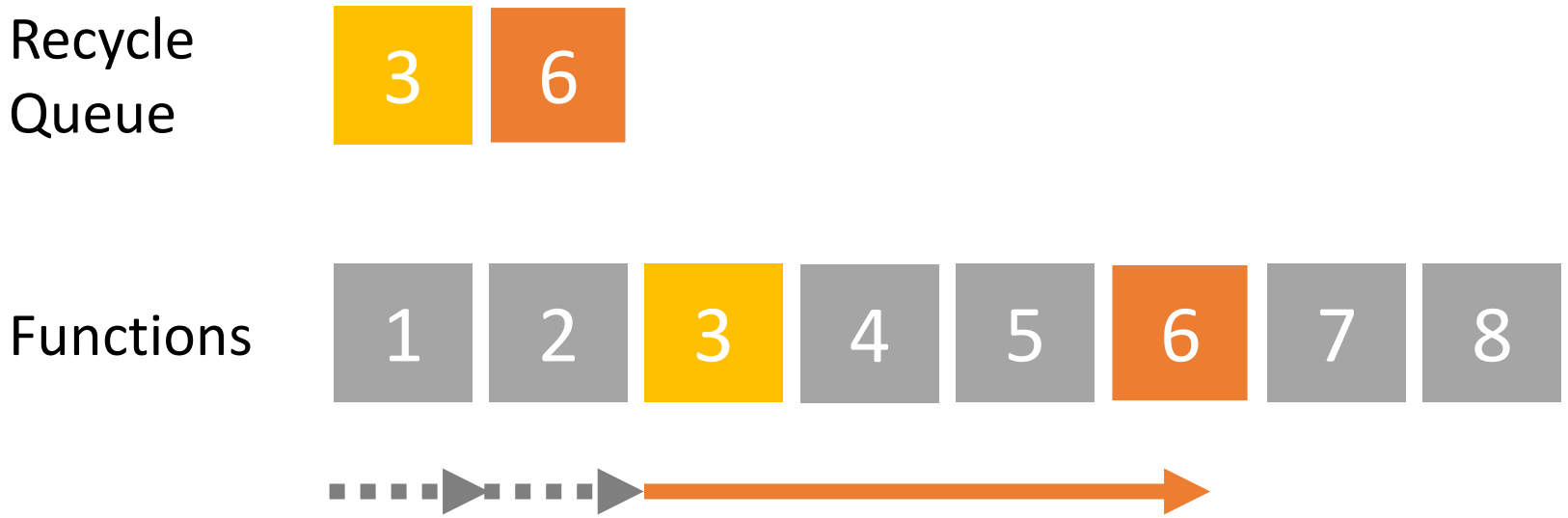
Disconnection

Disconnect

Disconnect



Connect : Reusing 3/6



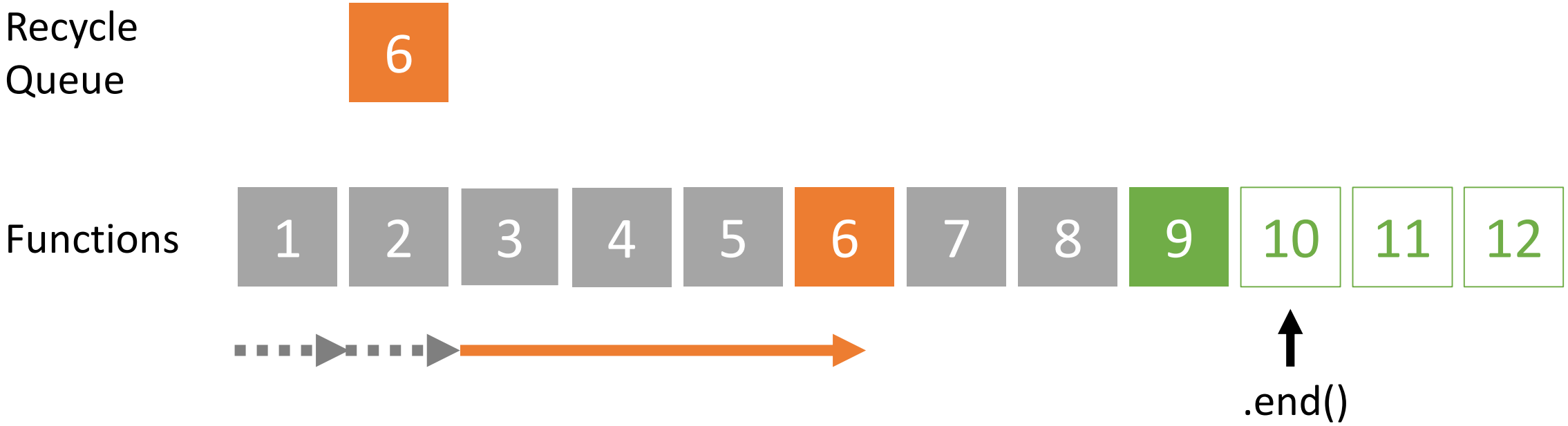
Connect : Reusing 3/6

Recycle Queue

Functions

Connect : Append

Connect : Append



Reference stability

```
std::vector<int> v;
auto itr = v.begin();
v.push_back(1);
something(itr);

/*----------------------------*/

std::deque<int> v;
auto itr = v.begin();
v.push_back(1);
something(itr);
```

Lock-free != no std::mutex

```
std::atomic<bool> lock;

bool expected;
do{
    // spin-locking on ‘lock’
    expected = false;
} while(!lock.compare_exchange_weak(expected, true));
```

Very short critical section

```
critical_section(sum) {
    sum += 1;
}

/*-----------------------------*/

std::atomic_fetch_add(&sum, 1)
```

Do large work in private

```
critical_section(sum) { global = new double[100]; }

/*-----------------------------------------------*/

double* local = new double[100];
double* expected = nullptr;
if (!global.compare_exachange_strong(expected, local))
{
    delete local;
    local = expected;
}
```

Memory order & OOO execution

```
ready.store(false, release);     // A
bool r1 = emit.load(acquire);    // B
if (r1 == 0)
    function = nullptr;          // $C
```

B -> A -> $C

Sequential cast?

```
ready.store(false, seq_cast);     // A
bool r1 = emit.load(seq_cast);    // B
if (r1 == 0)
    function = nullptr;           // $C
```

A -> B -> $C

Combine status

```
status == (ready << 31) | emit
```

```
bool r1 = status.fetch_and( ~ready_bit, acq_rel );
if (!(r1 & ~ready_bit))
    function = nullptr;
```

Lock-free Signal/Slots Questions?

yupeng.zhang@autodesk.com

https://git.autodesk.com/media-and-entertainment/evaluation-manager