Opacity

Concurrent Algorithms 2011

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

What is TM?

What is TM?

What does TM guarantee?

Serializability

Serializability

```
int a = acc_a;
acc_a = a - 20;
int b = acc_b;
acc_b = b + 20;
```

Serializability

```
atomic { // t1
   int a = acc_a;
   acc_a = a - 20;
   int b = acc_b;
   acc_b = b + 20;
}
```

Serializability

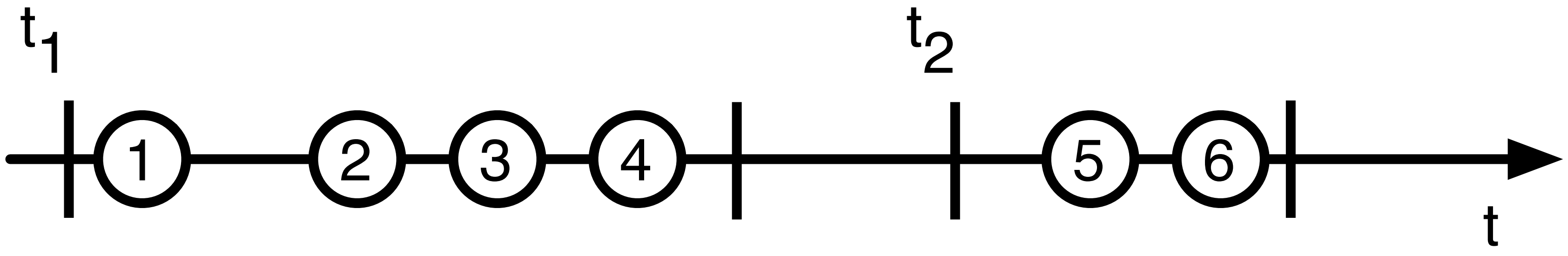
```
atomic { // t1
1: int a = acc_a;
2: acc_a = a - 20;
3: int b = acc_b;
4: acc_b = b + 20;
}
```

```
atomic { // t2
5: int a = acc_a;
6: acc_a = a + 10;
}
```

Serializability

```
atomic { // t1
1: int a = acc_a;
2: acc_a = a - 20;
3: int b = acc_b;
4: acc_b = b + 20;
}
```

```
atomic { // t2
5: int a = acc_a;
6: acc_a = a + 10;
}
```

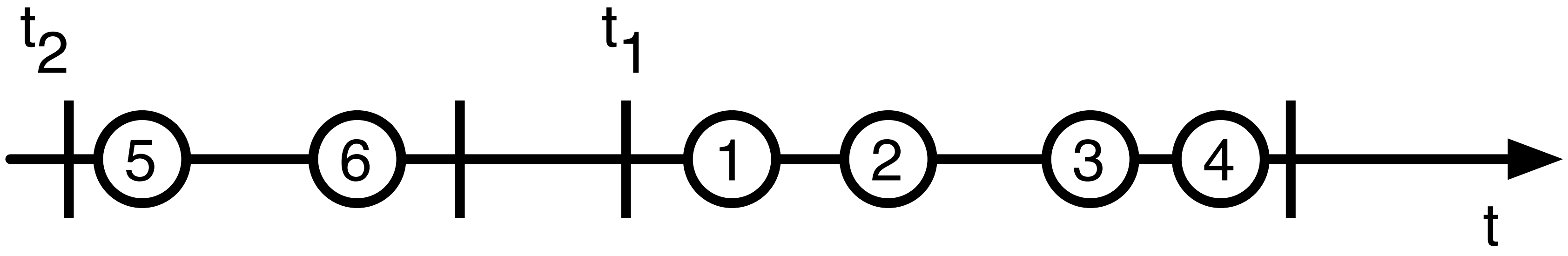


correct

Serializability

```
atomic { // t1
1: int a = acc_a;
2: acc_a = a - 20;
3: int b = acc_b;
4: acc_b = b + 20;
}
```

```
atomic { // t2
5: int a = acc_a;
6: acc_a = a + 10;
}
```

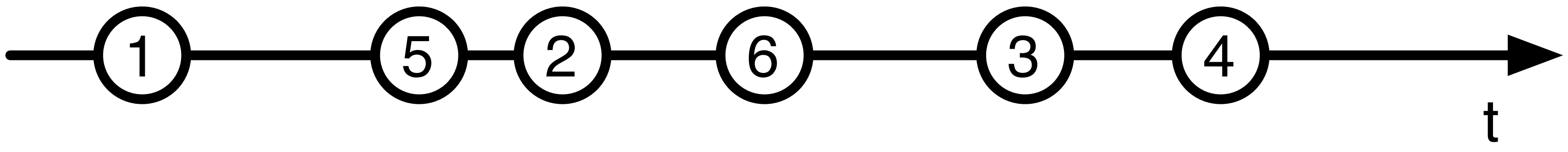


correct

Serializability

```
atomic { // t1
1: int a = acc_a;
2: acc_a = a - 20;
3: int b = acc_b;
4: acc_b = b + 20;
}
```

```
atomic { // t2
5: int a = acc_a;
6: acc_a = a + 10;
}
```

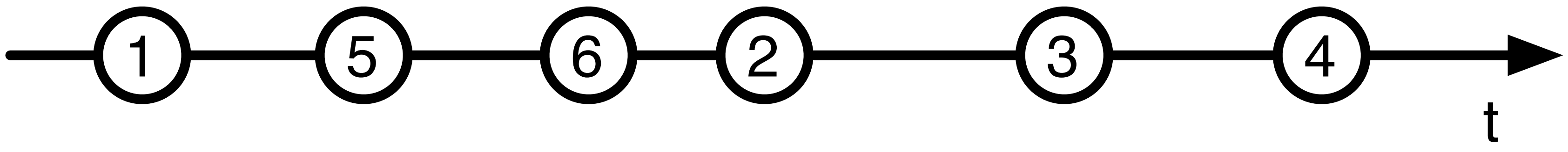


client ☺

Serializability

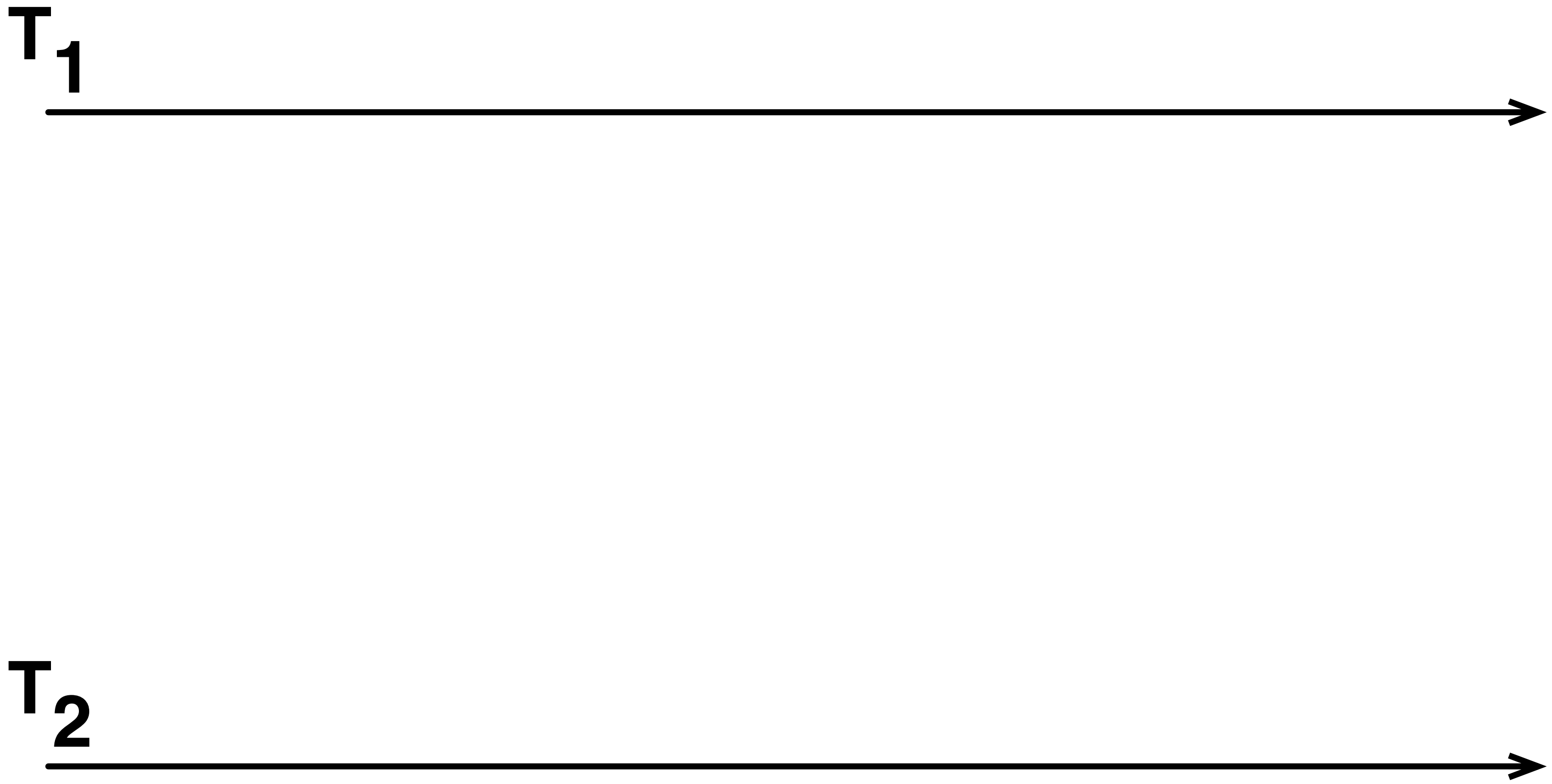
```
atomic { // t1
1: int a = acc_a;
2: acc_a = a - 20;
3: int b = acc_b;
4: acc_b = b + 20;
}
```

```
atomic { // t2
5: int a = acc_a;
6: acc_a = a + 10;
}
```

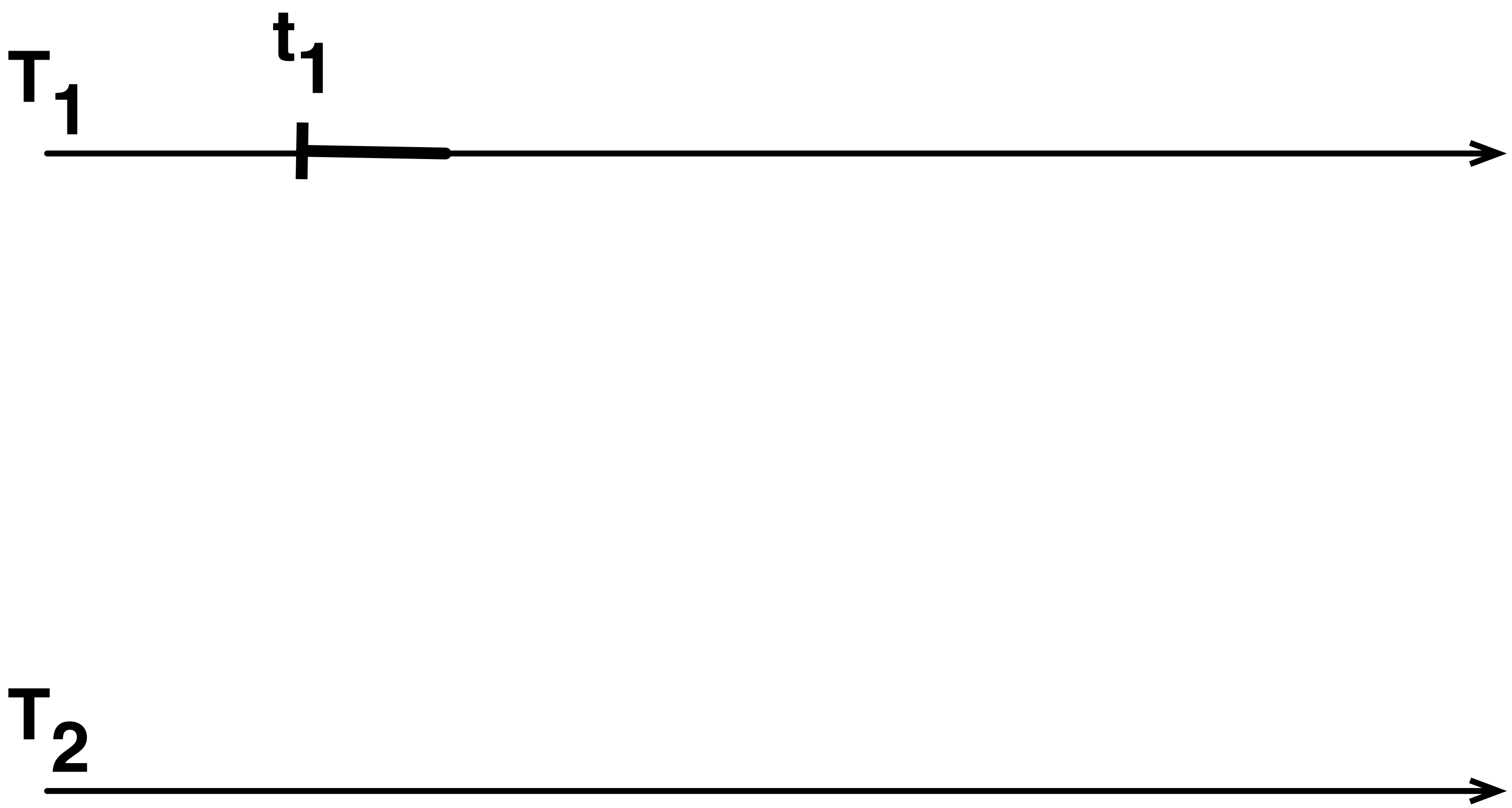


bank ☺

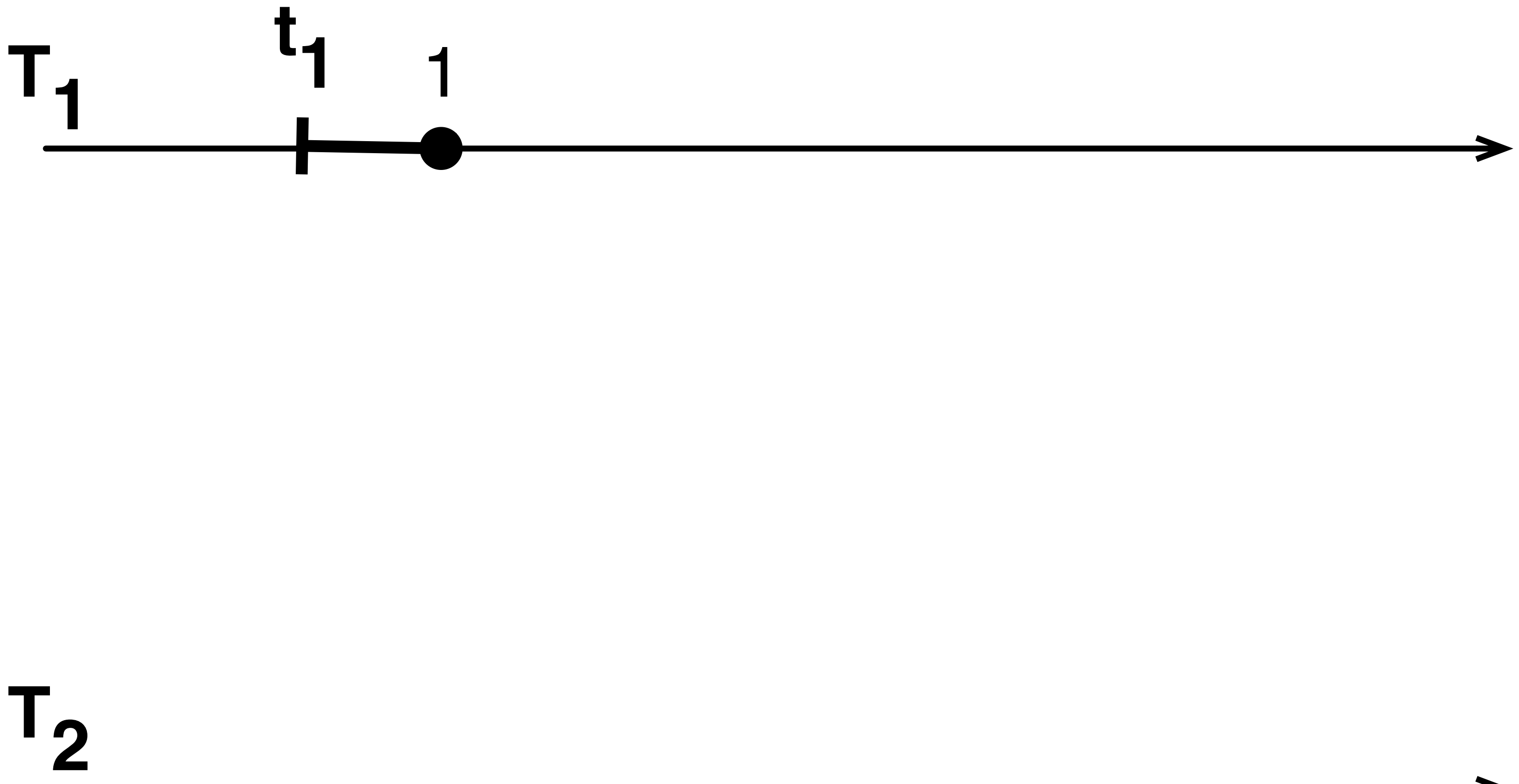
How is this achieved?



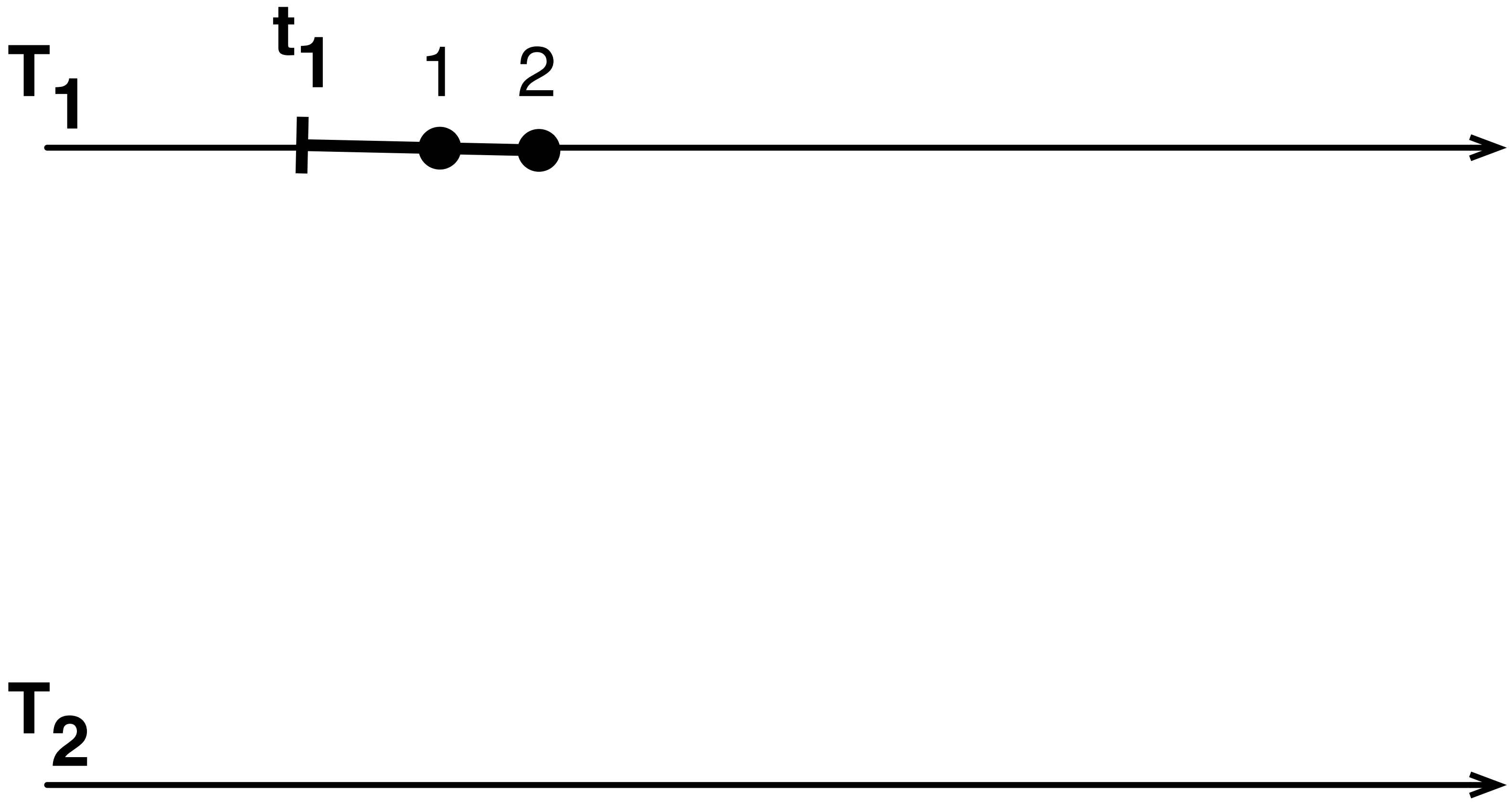
How is this achieved?



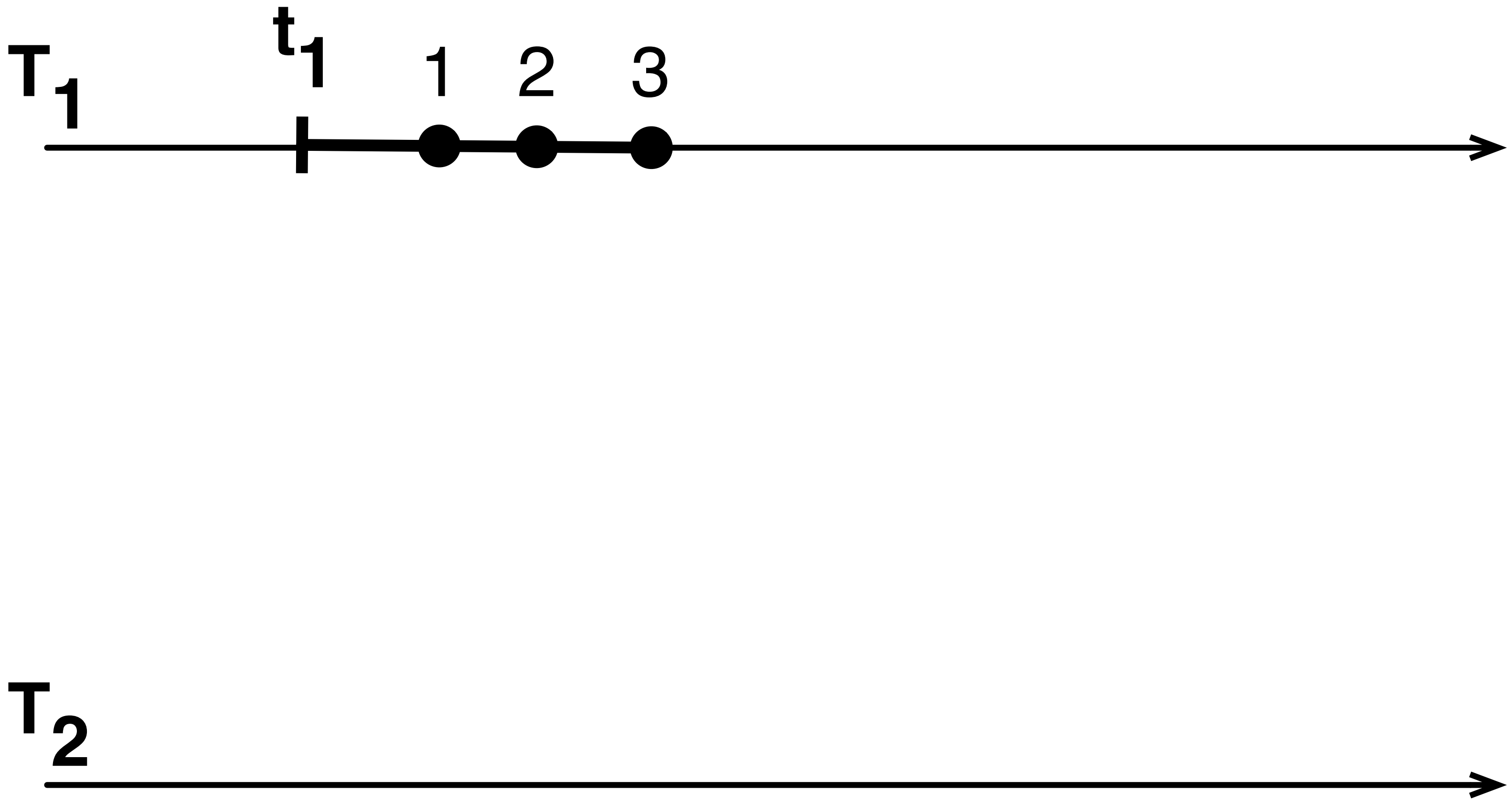
How is this achieved?



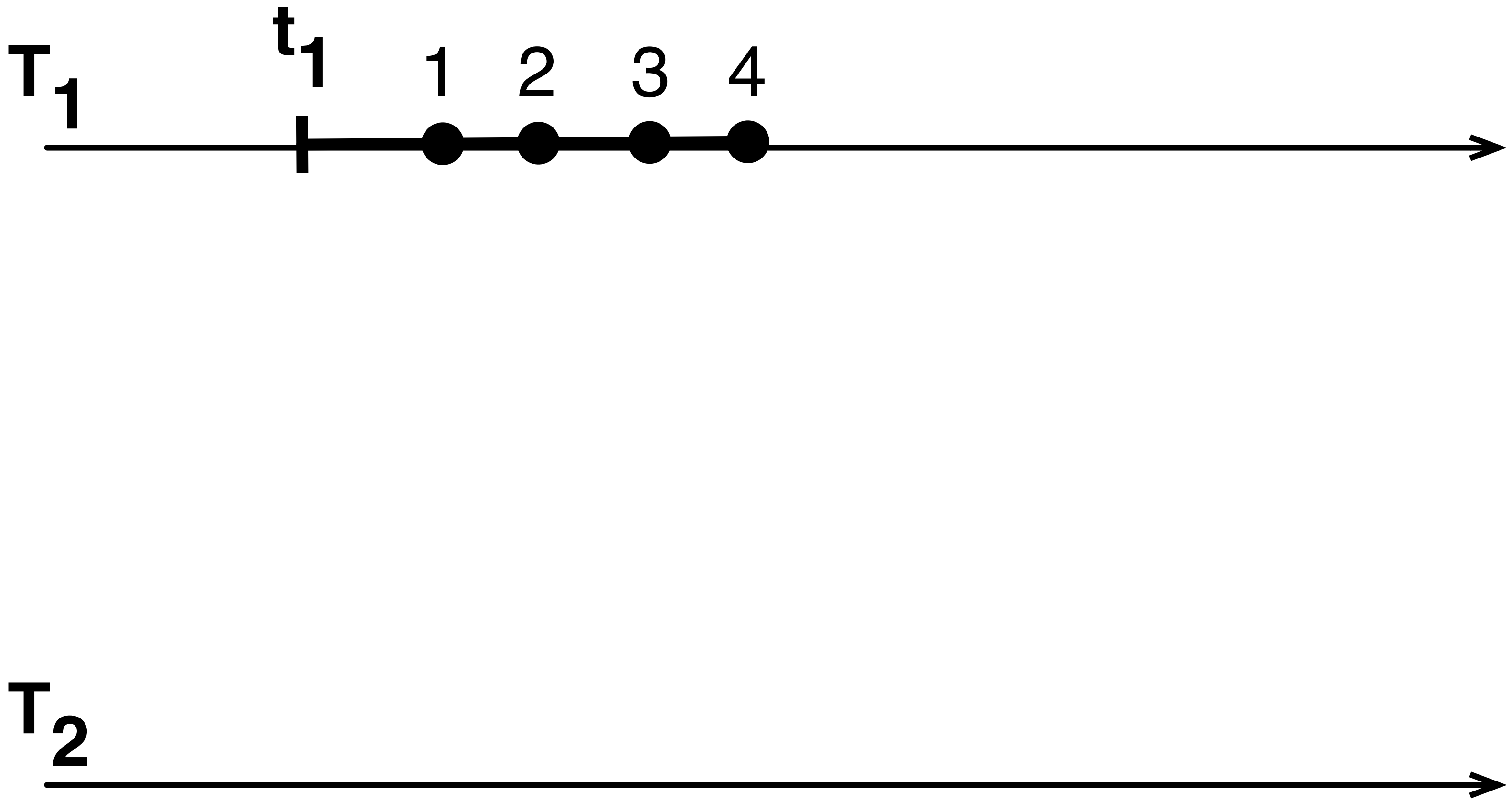
How is this achieved?



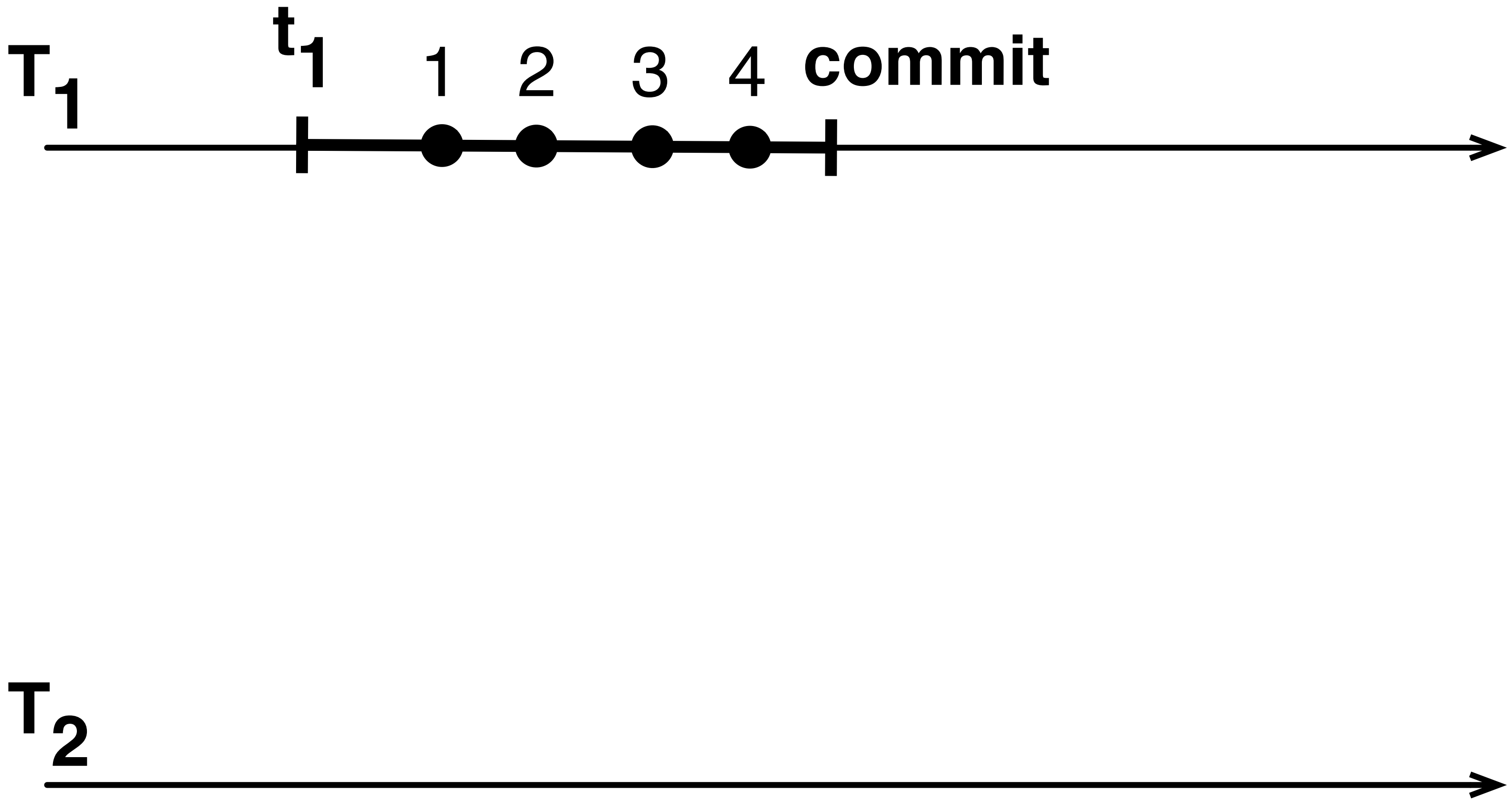
How is this achieved?



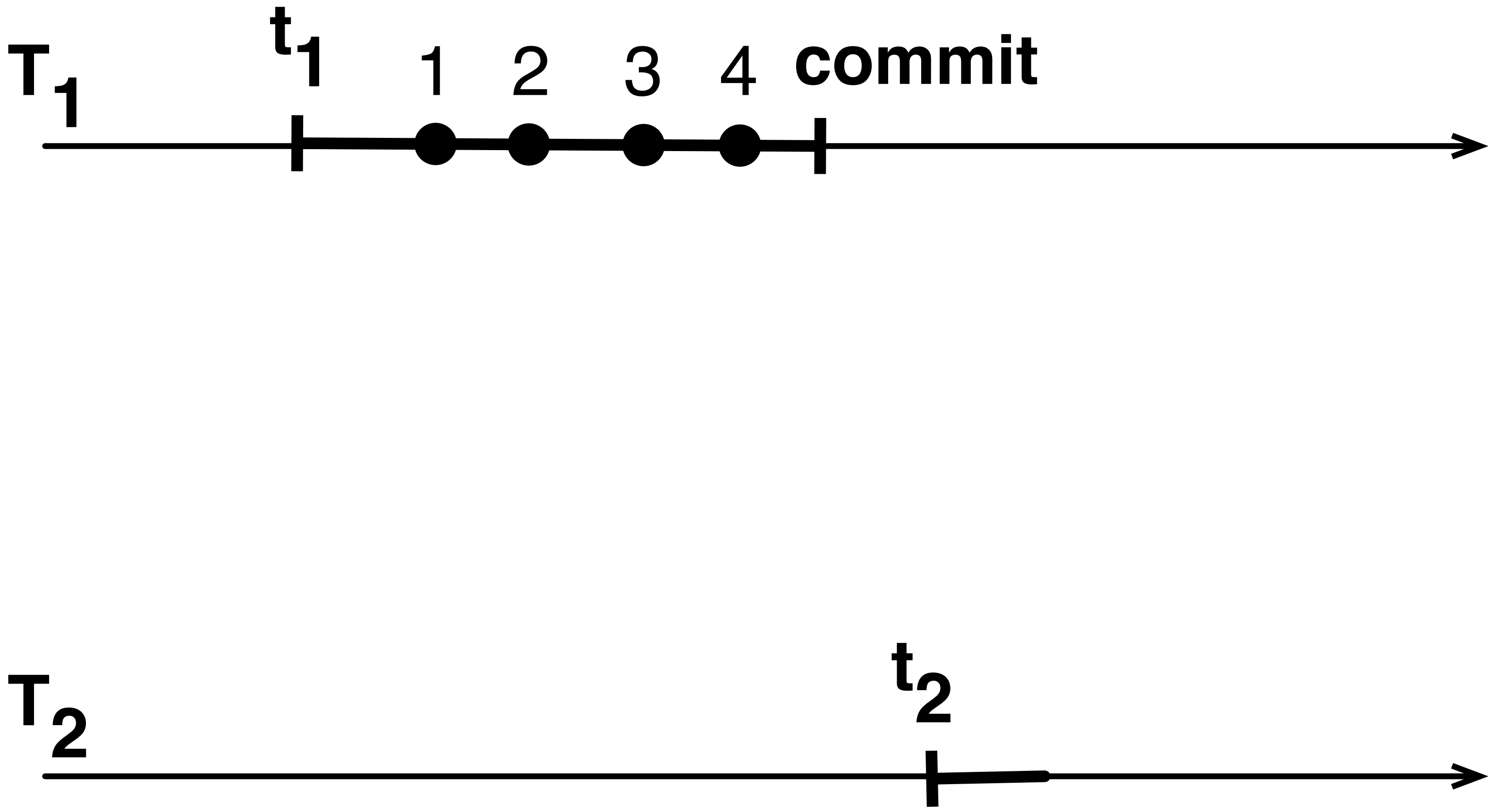
How is this achieved?



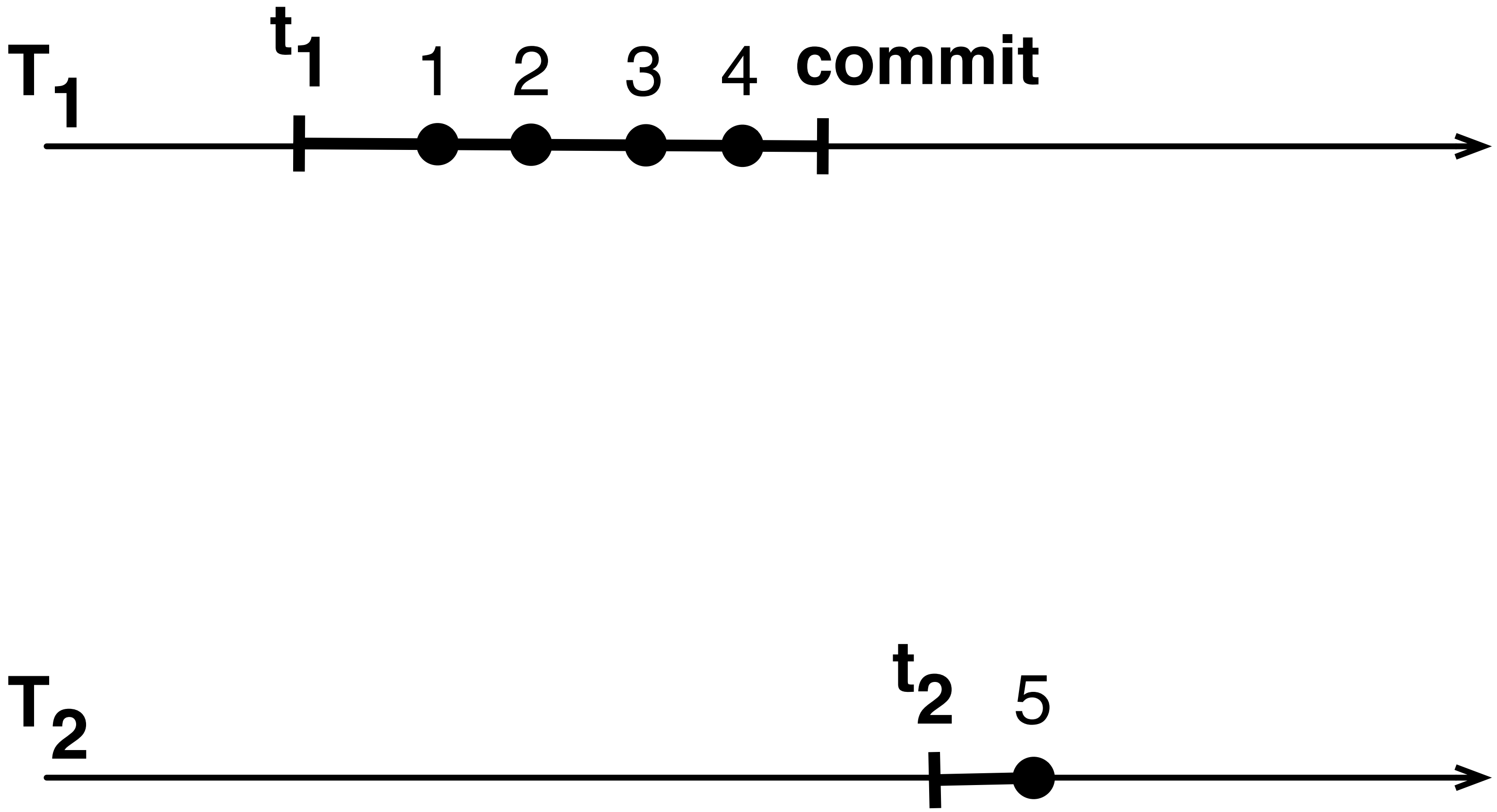
How is this achieved?



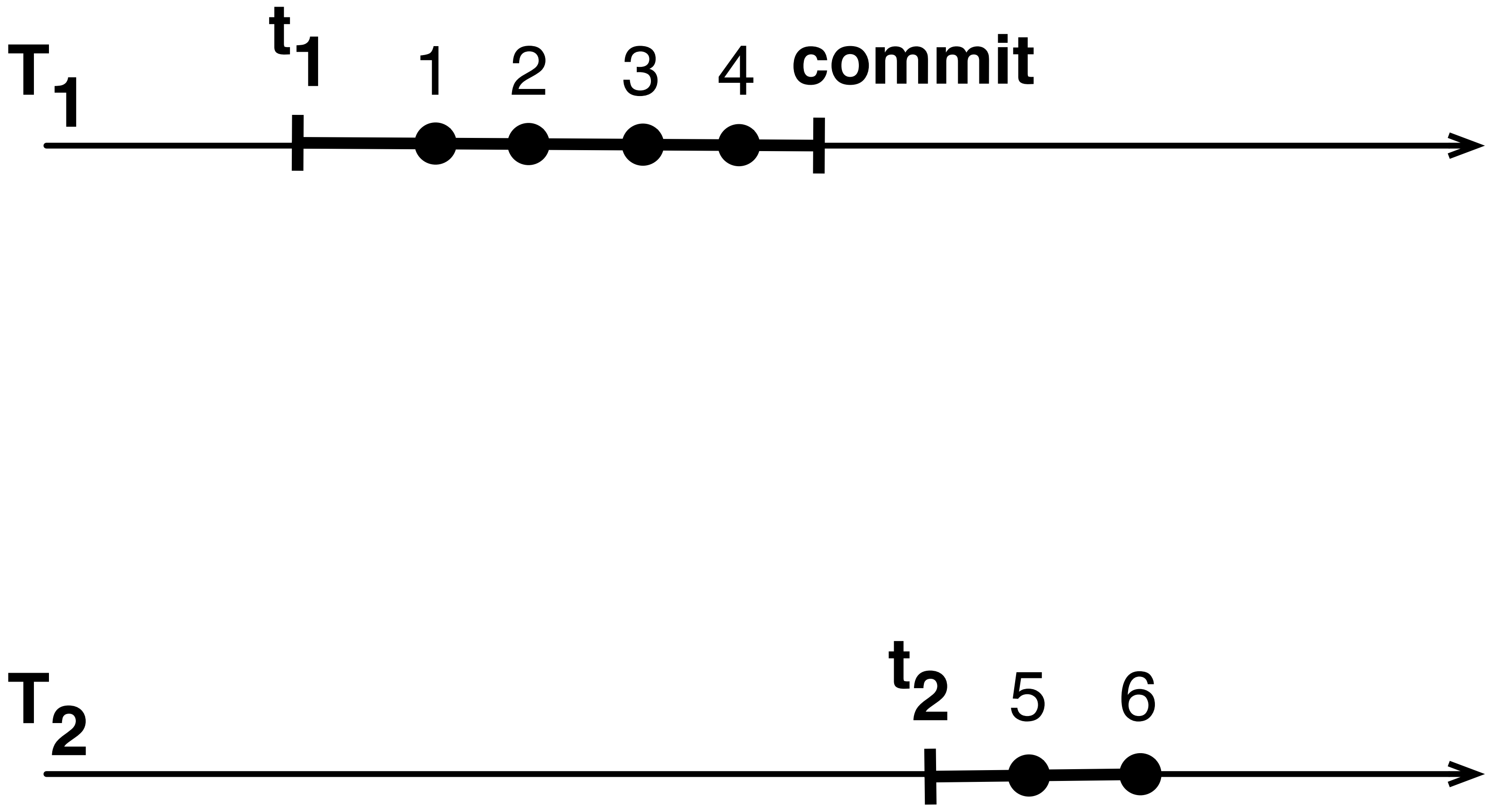
How is this achieved?



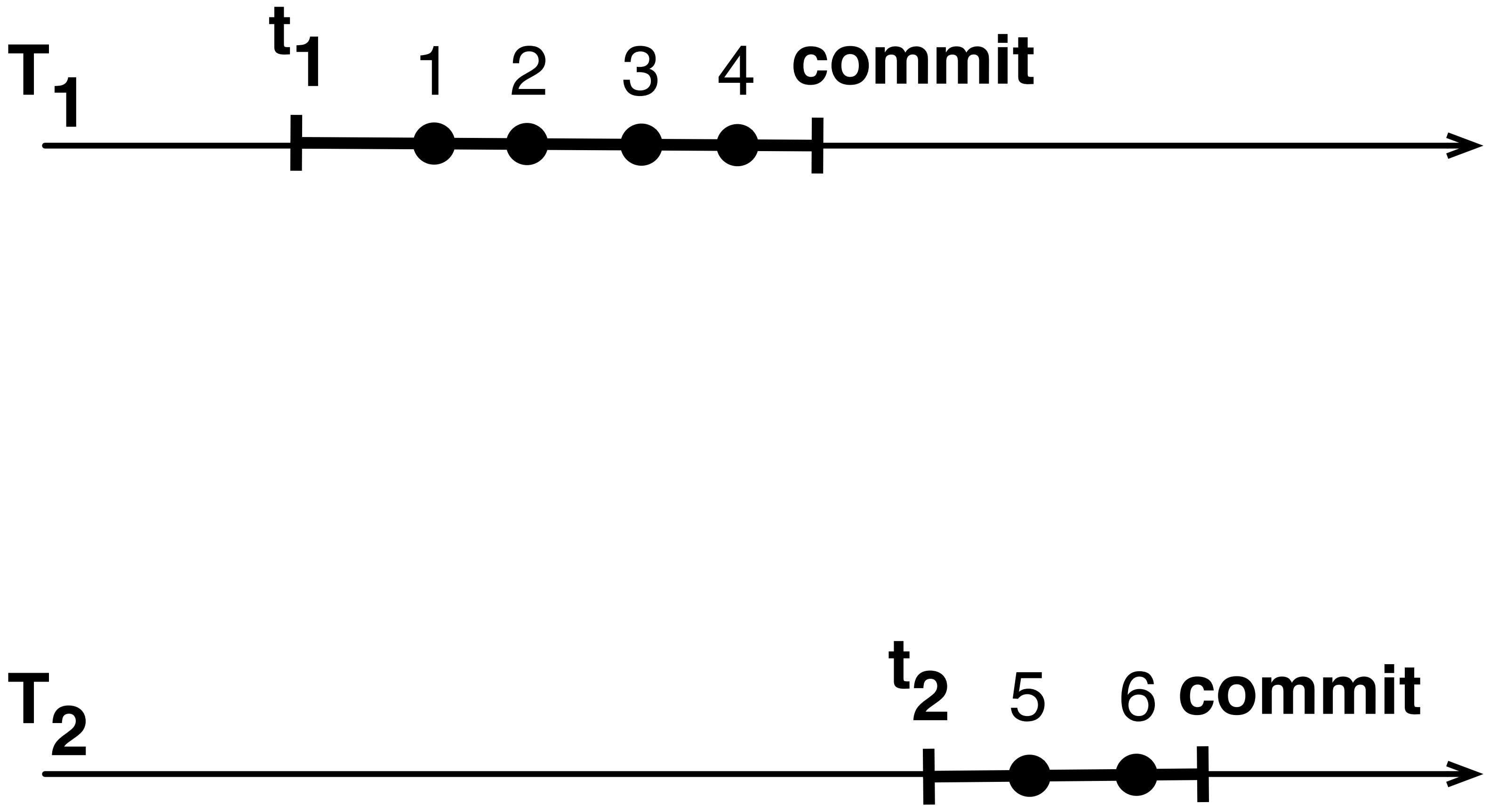
How is this achieved?



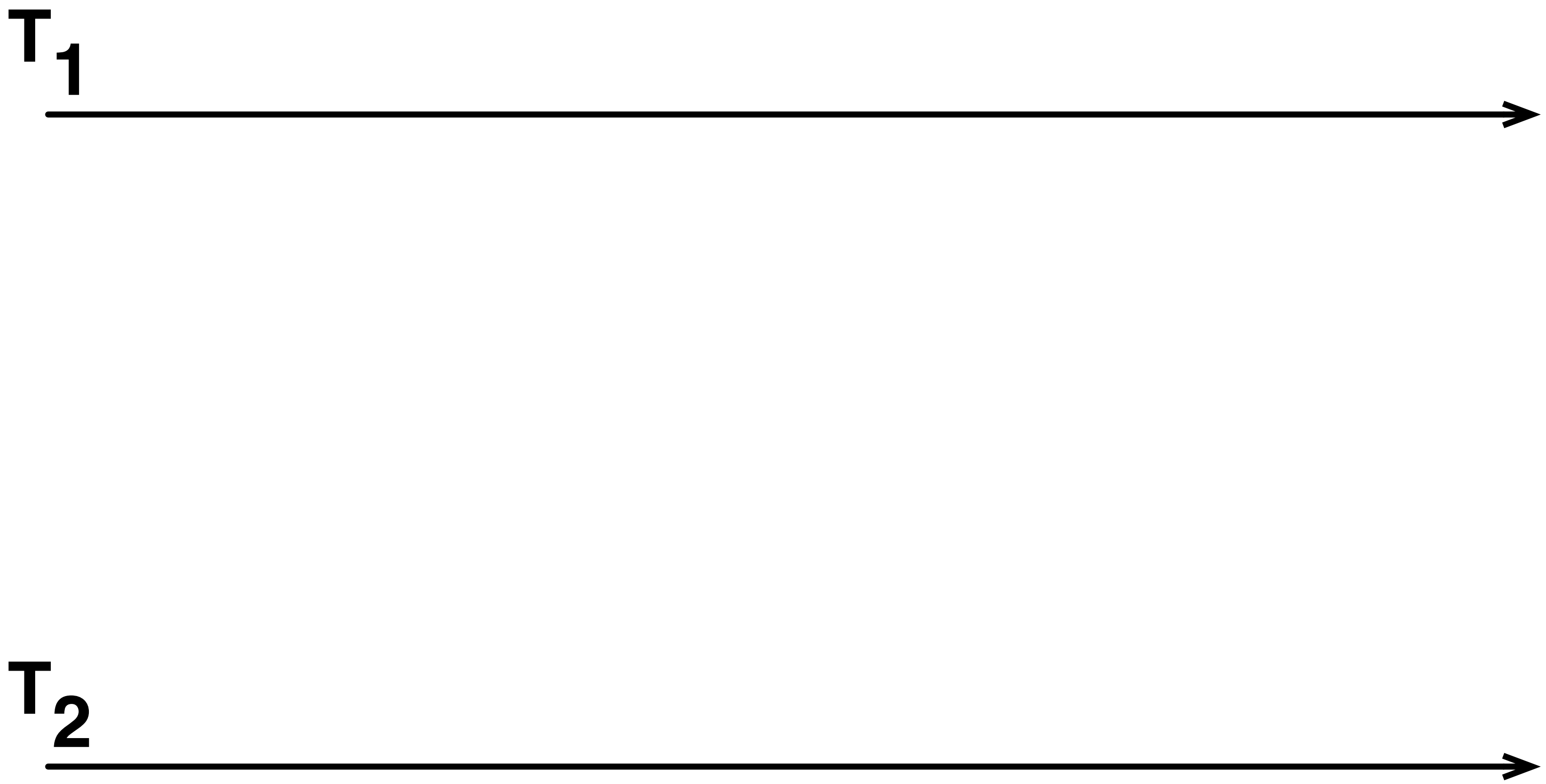
How is this achieved?



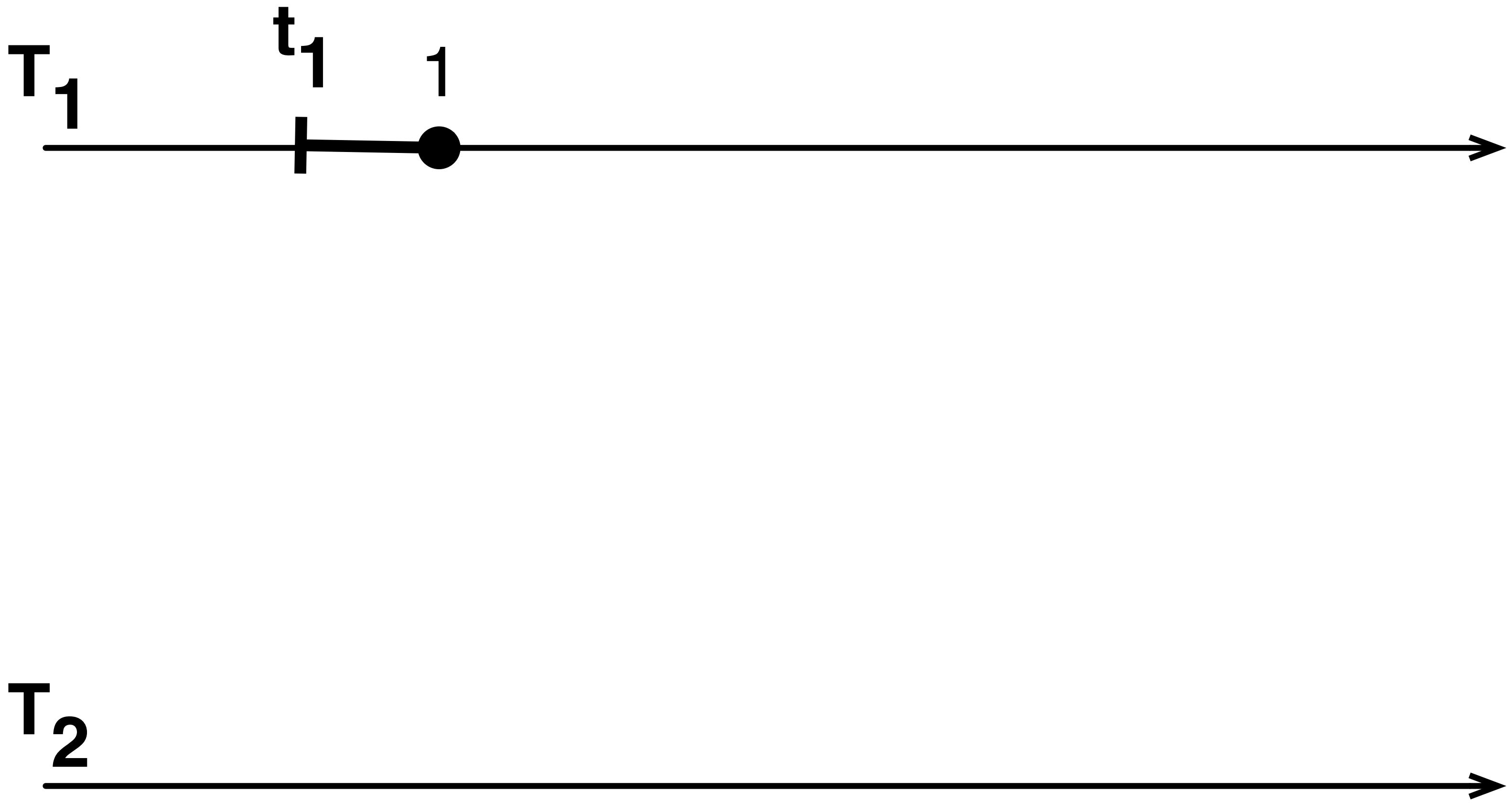
How is this achieved?



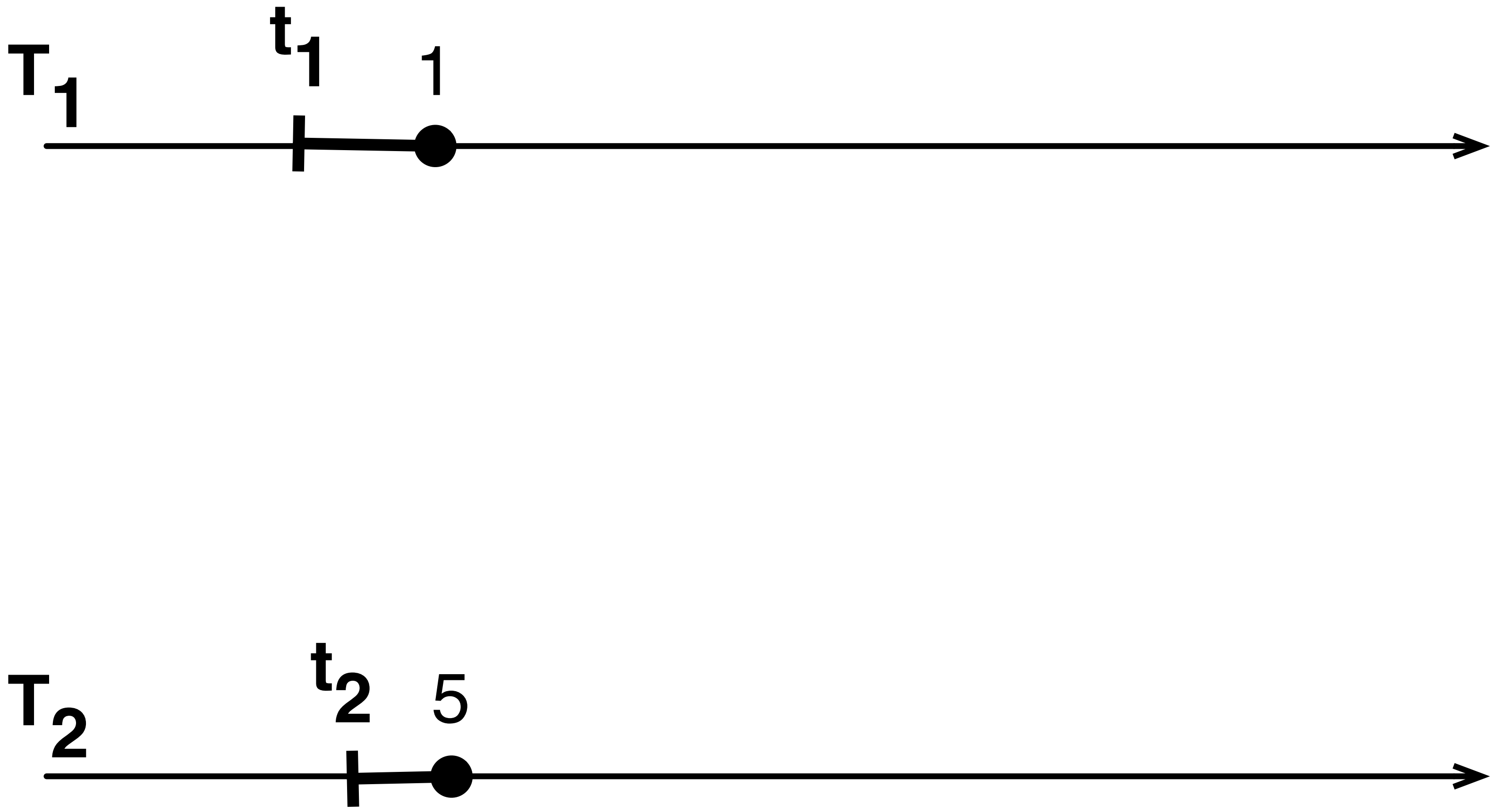
How is this achieved?



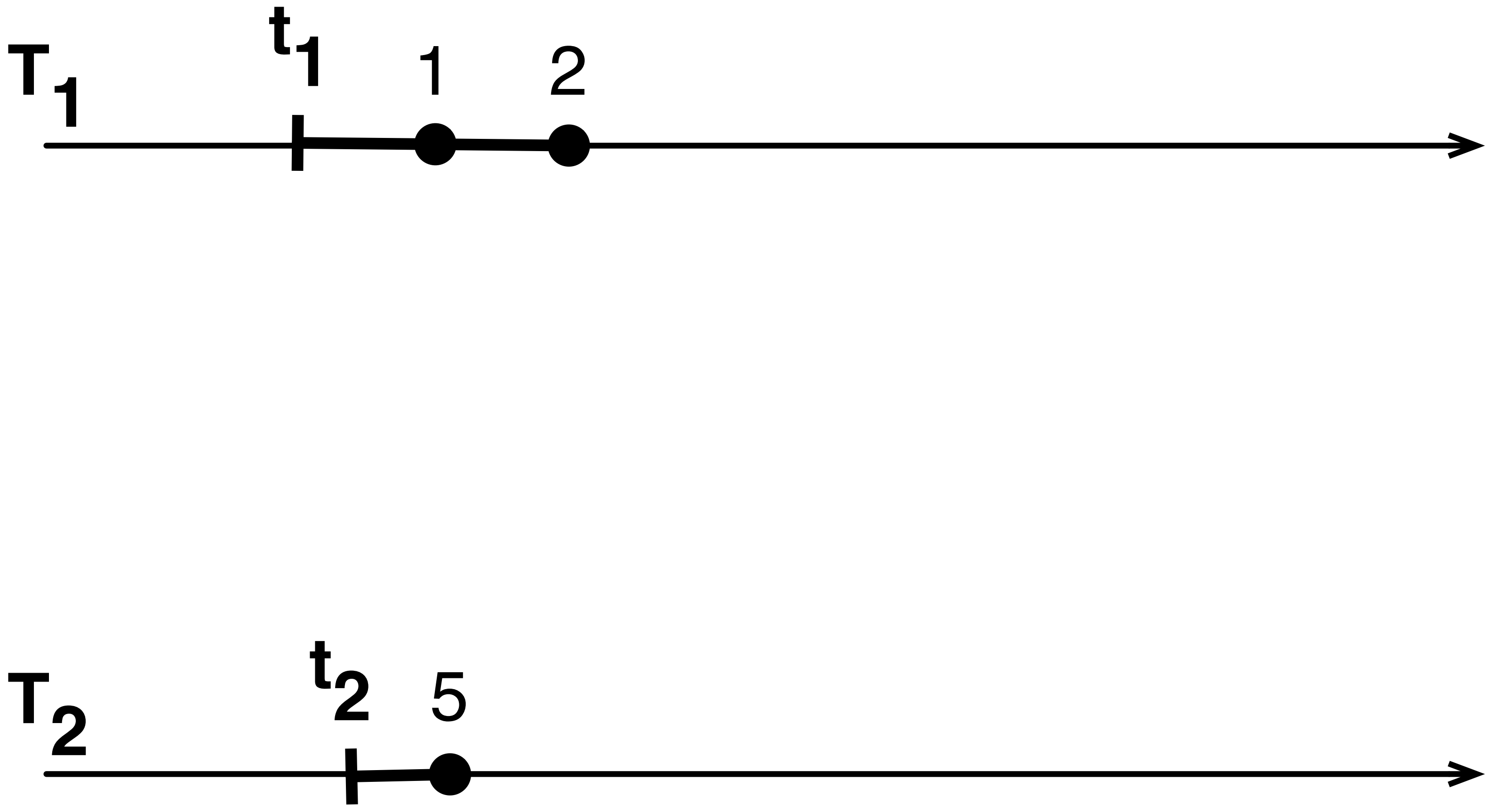
How is this achieved?



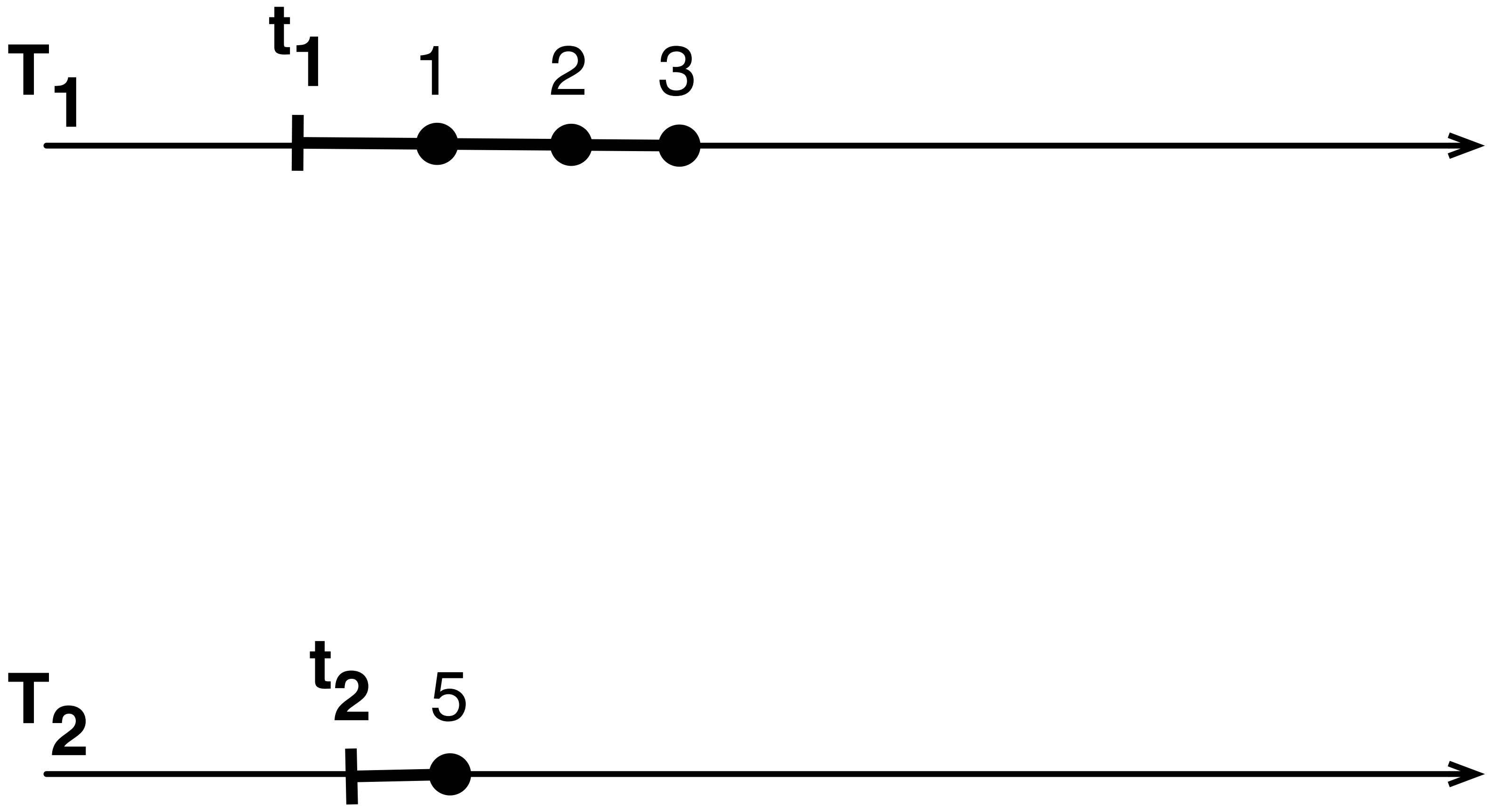
How is this achieved?



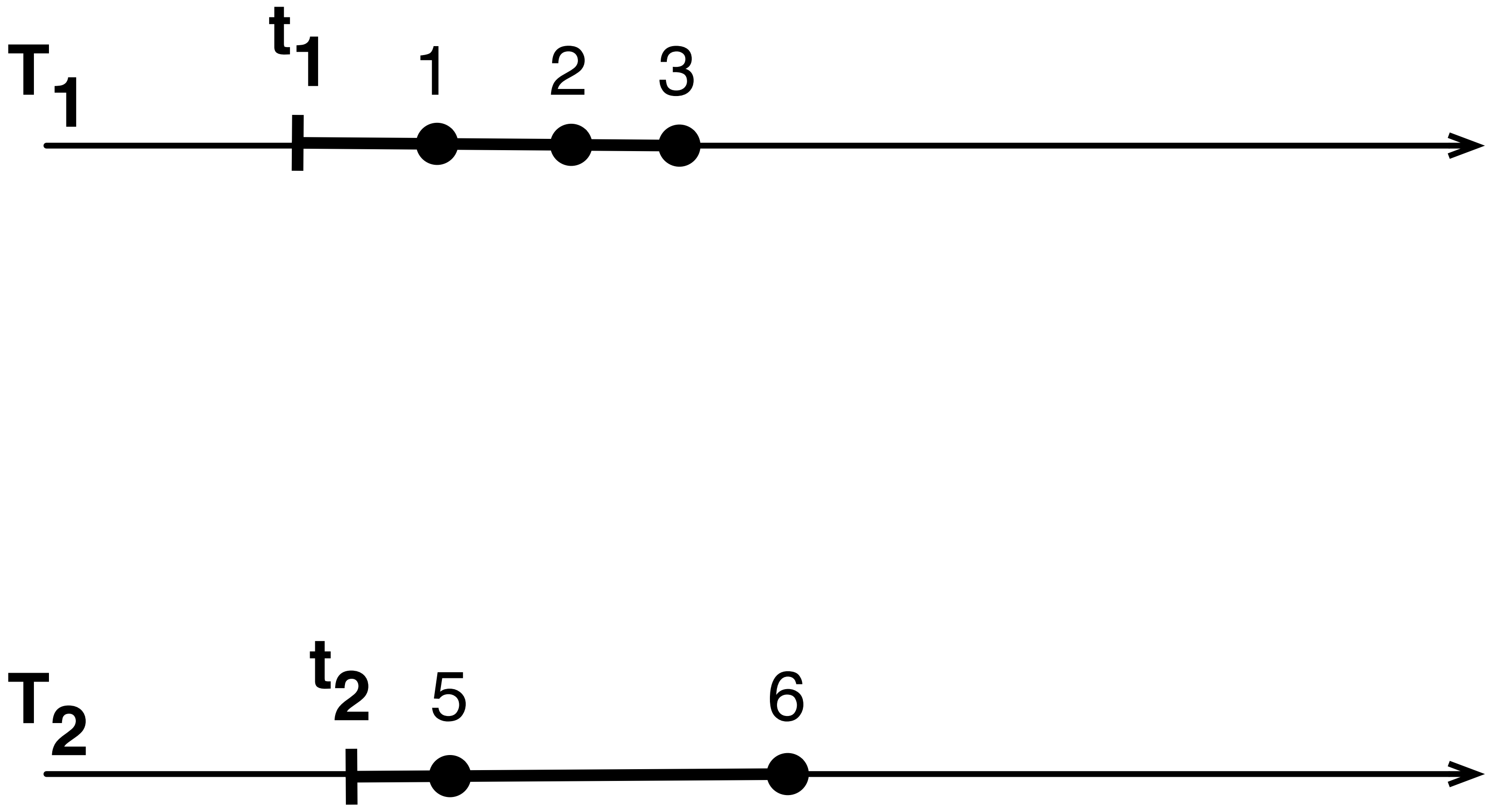
How is this achieved?



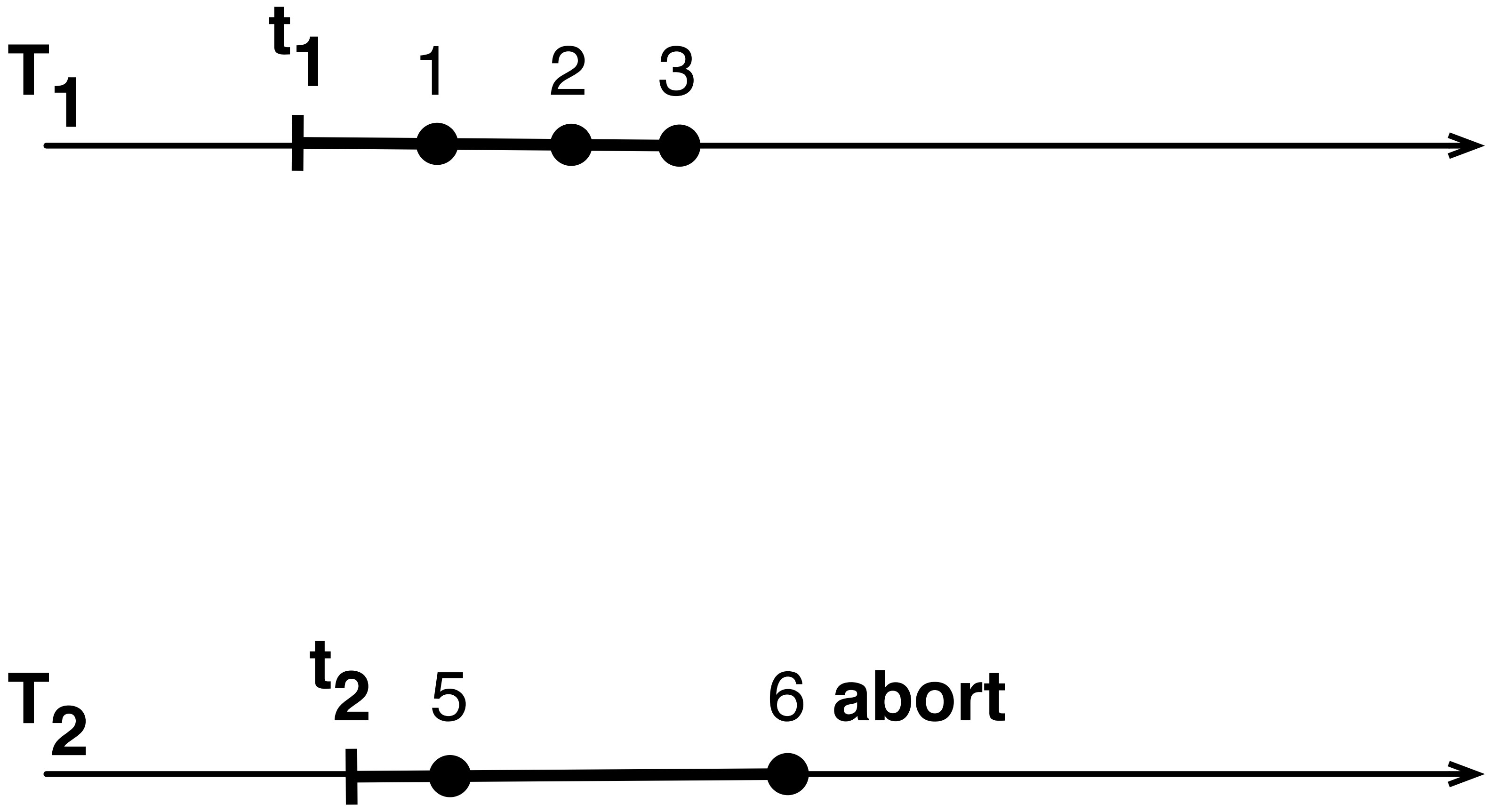
How is this achieved?



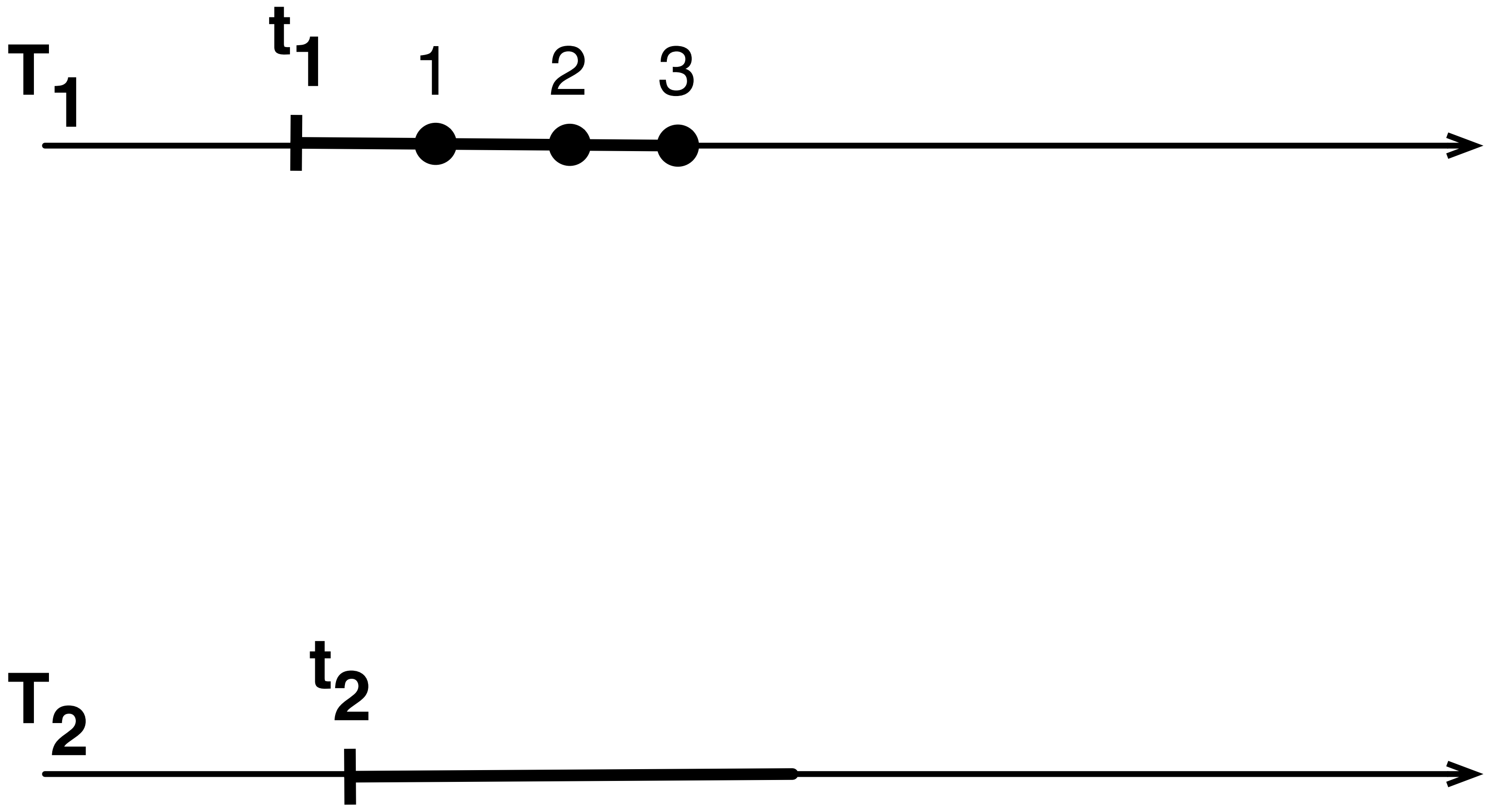
How is this achieved?



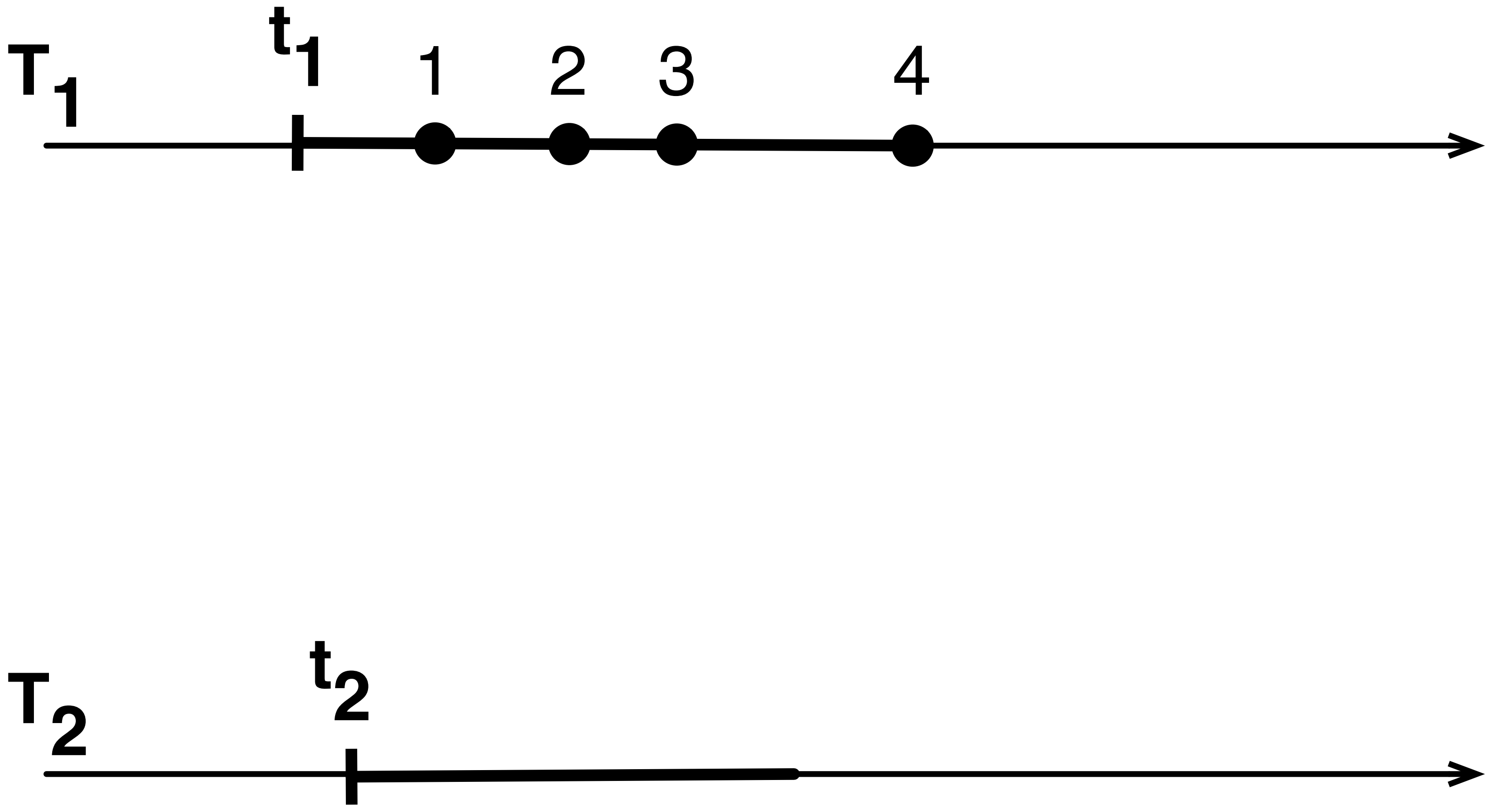
How is this achieved?



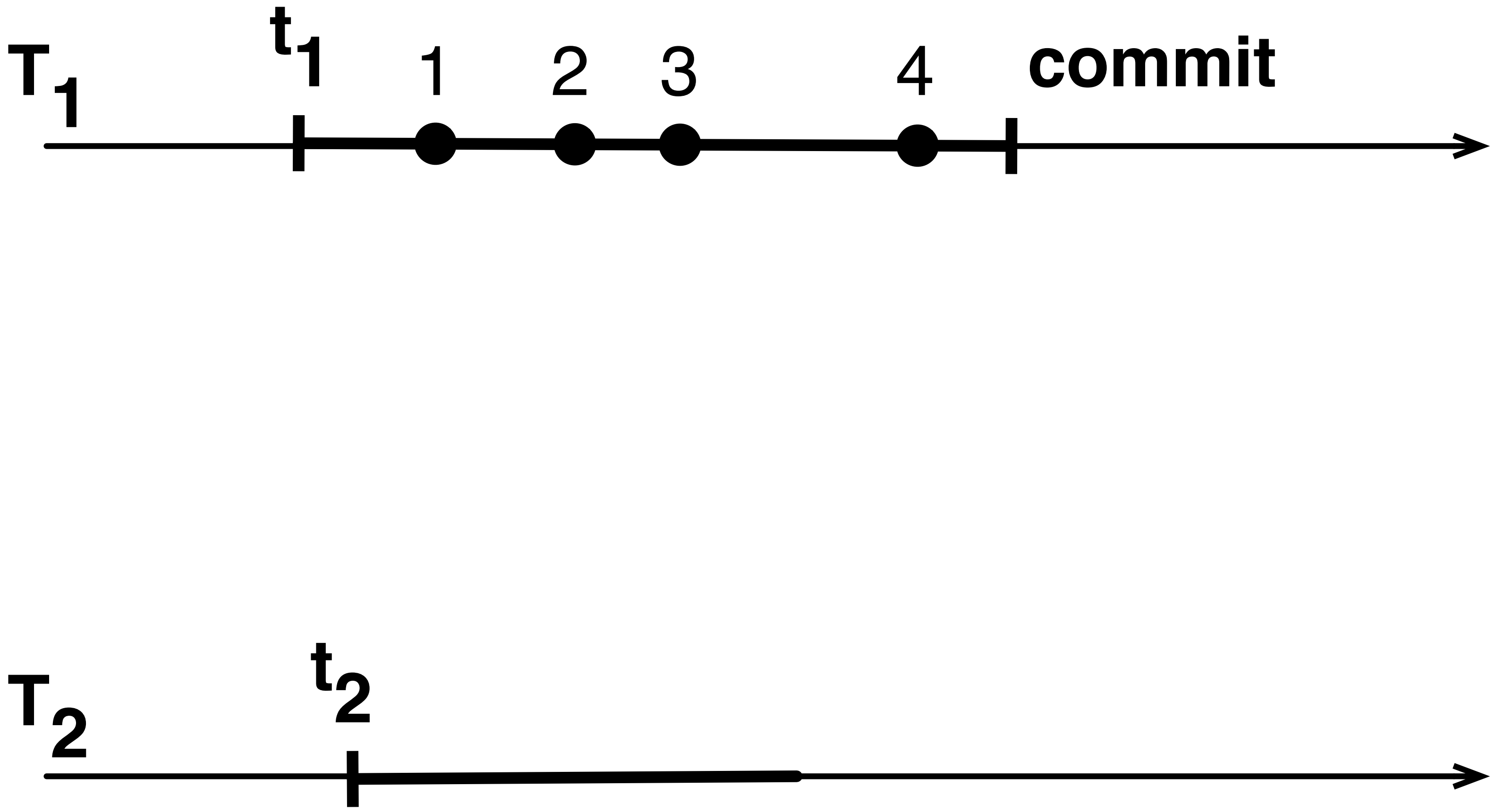
How is this achieved?



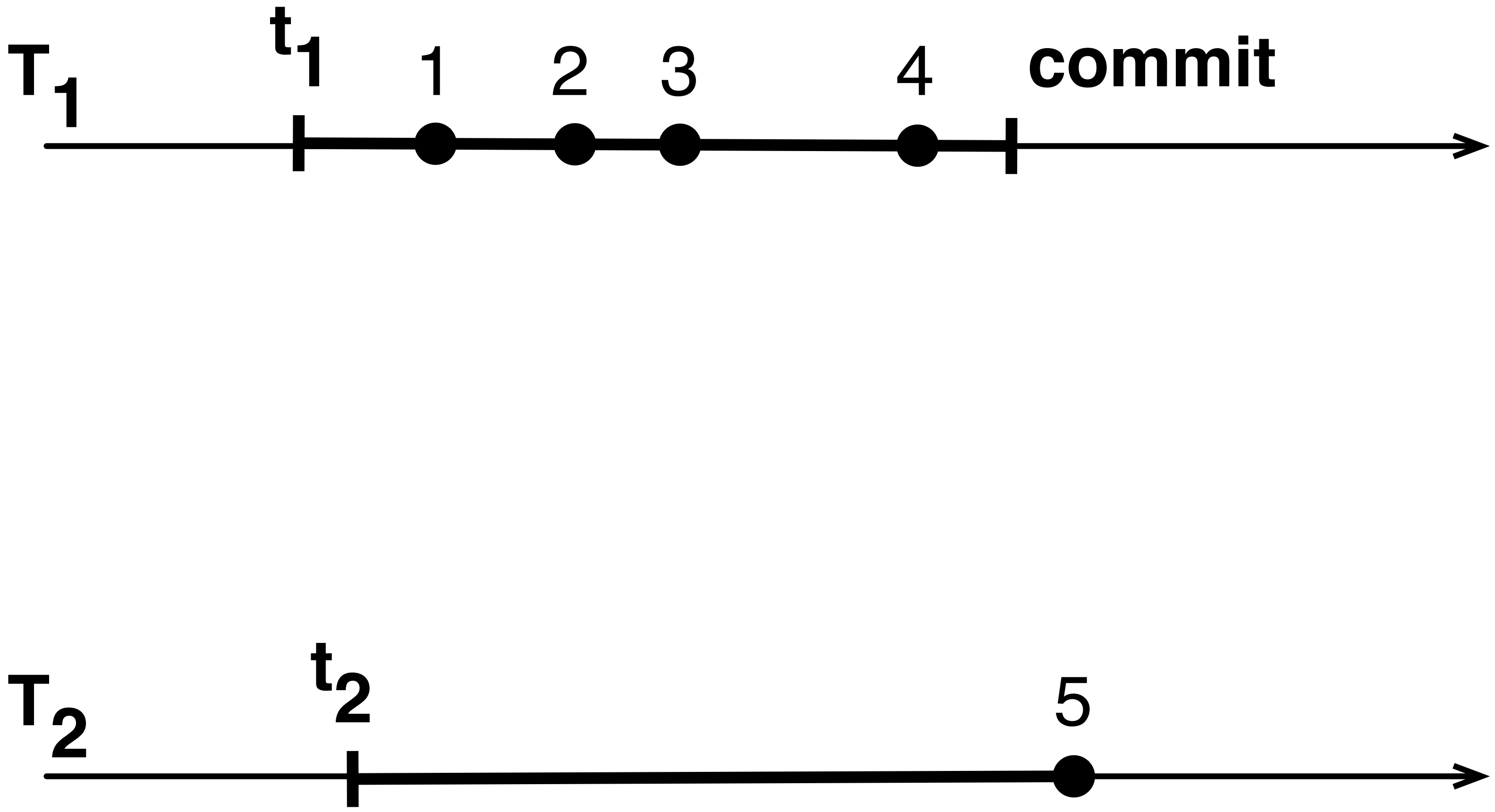
How is this achieved?



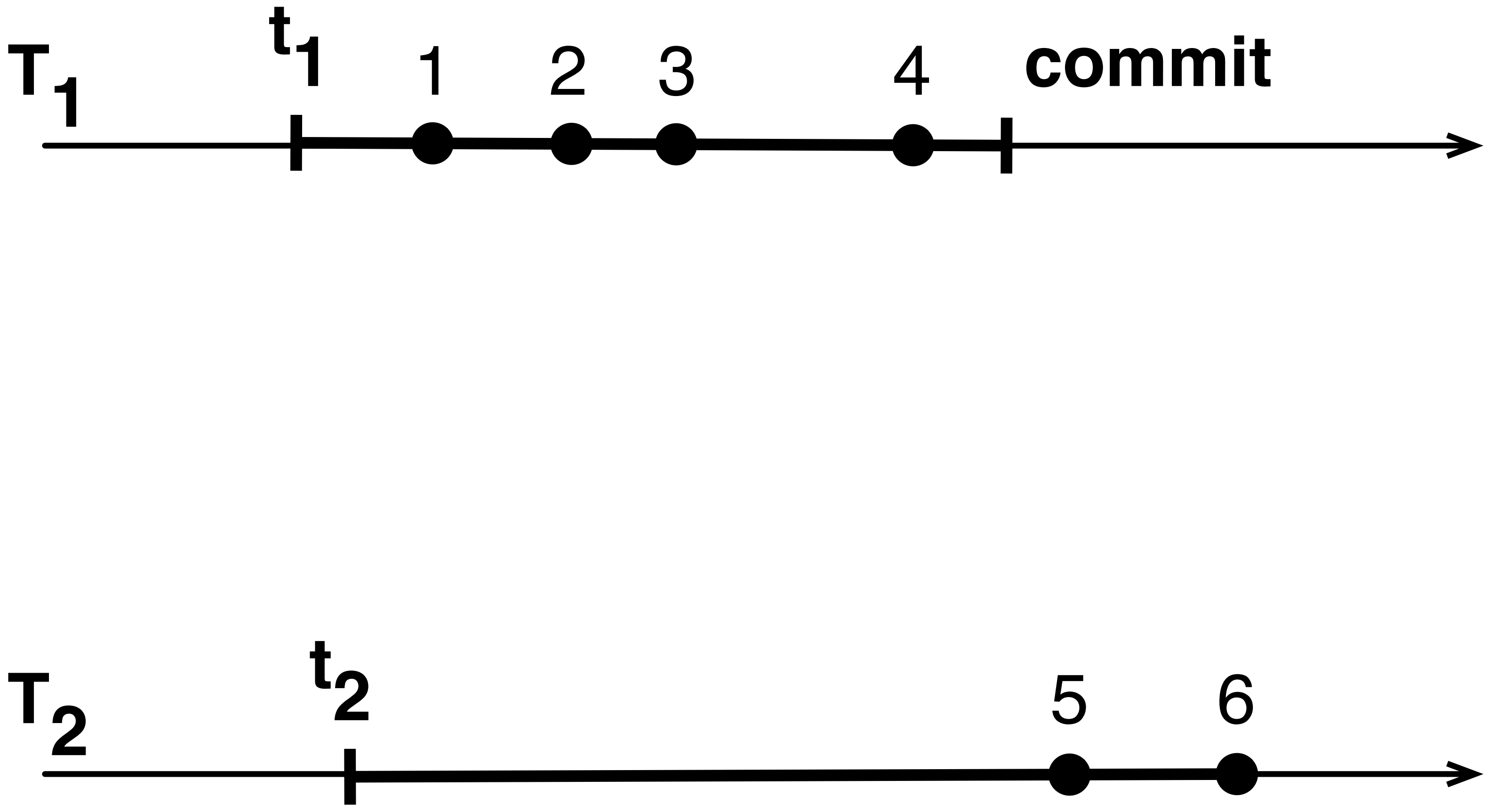
How is this achieved?



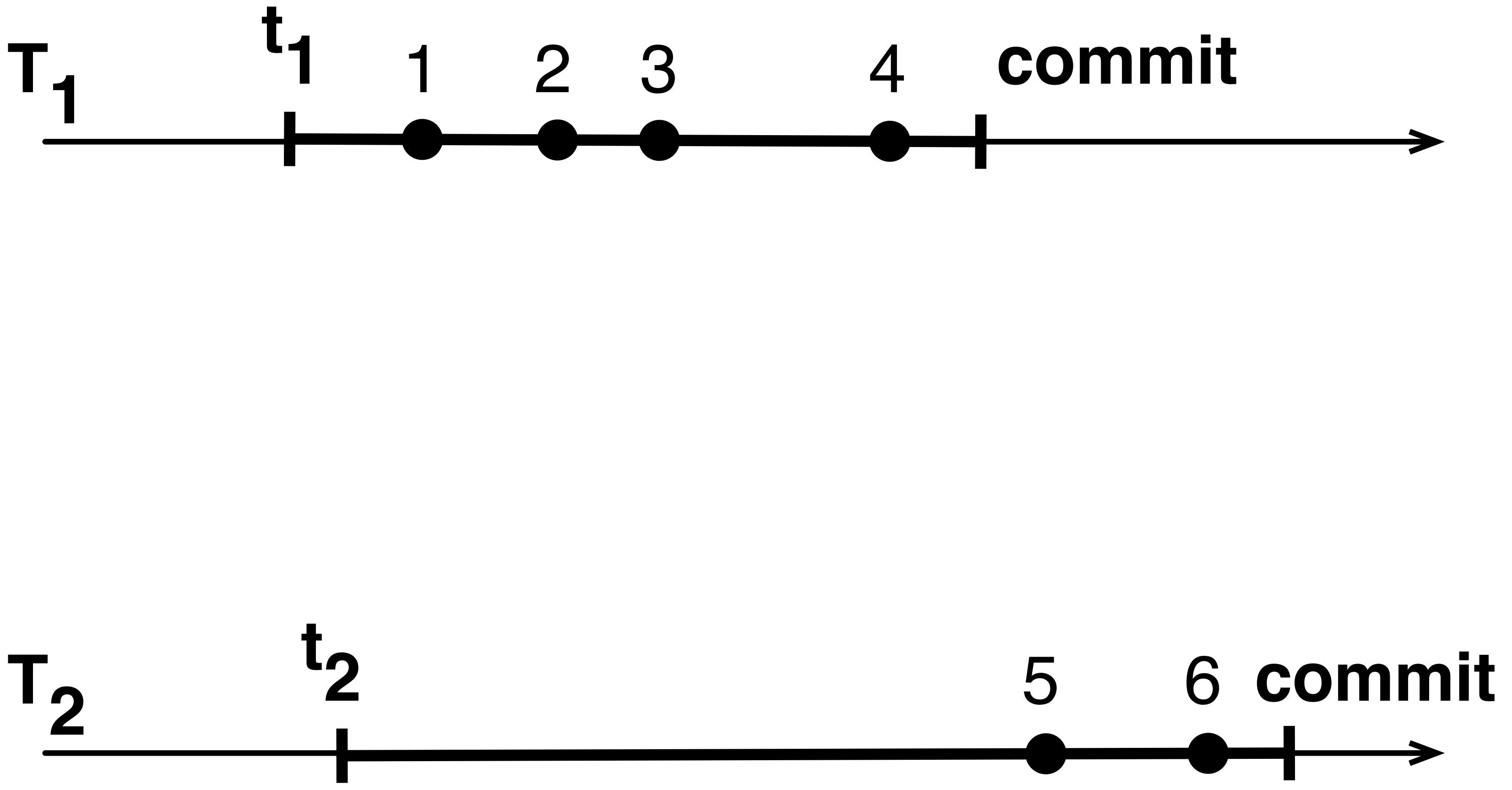
How is this achieved?



How is this achieved?



How is this achieved?



How is this achieved?

- TM monitors accesses to objects
- When it detects *conflicting* access
  - one transaction is *aborted*
  - its actions are *rolled back*
  - it is *restarted*
- When all actions are not conflicting
  - transaction *commits*

Is serializability enough?

Is serializability enough?

```
Variables:
    int x=0, y=1;
Invariant:
    x < y
```

Is serializability enough?

```
Variables:
    int x=0, y=1;
Invariant:
    x < y
```

```
atomic {
   int xl = x;
   int yl = y;
   x = yl;
   y = yl * 2;
}
```

Is serializability enough?

```
Variables:
    int x=0, y=1;
Invariant:
    x < y
```

```
atomic {
int xl = x;
int yl = y;
x = yl;
y = yl * 2;
}
```

```
atomic {
int xl = x;
int yl = y;
int zl = 1/(yl-xl);
z = zl;
}
```

Consistent view

- All transactions must observe consistent views of memory at all times
  - even the aborted ones

Opacity

- Serializability
  - there exists an equivalent serial (one thread) execution
- Consistent memory view
  - no transaction can e.g. divide by zero because of non-consistent reads

TM semantics

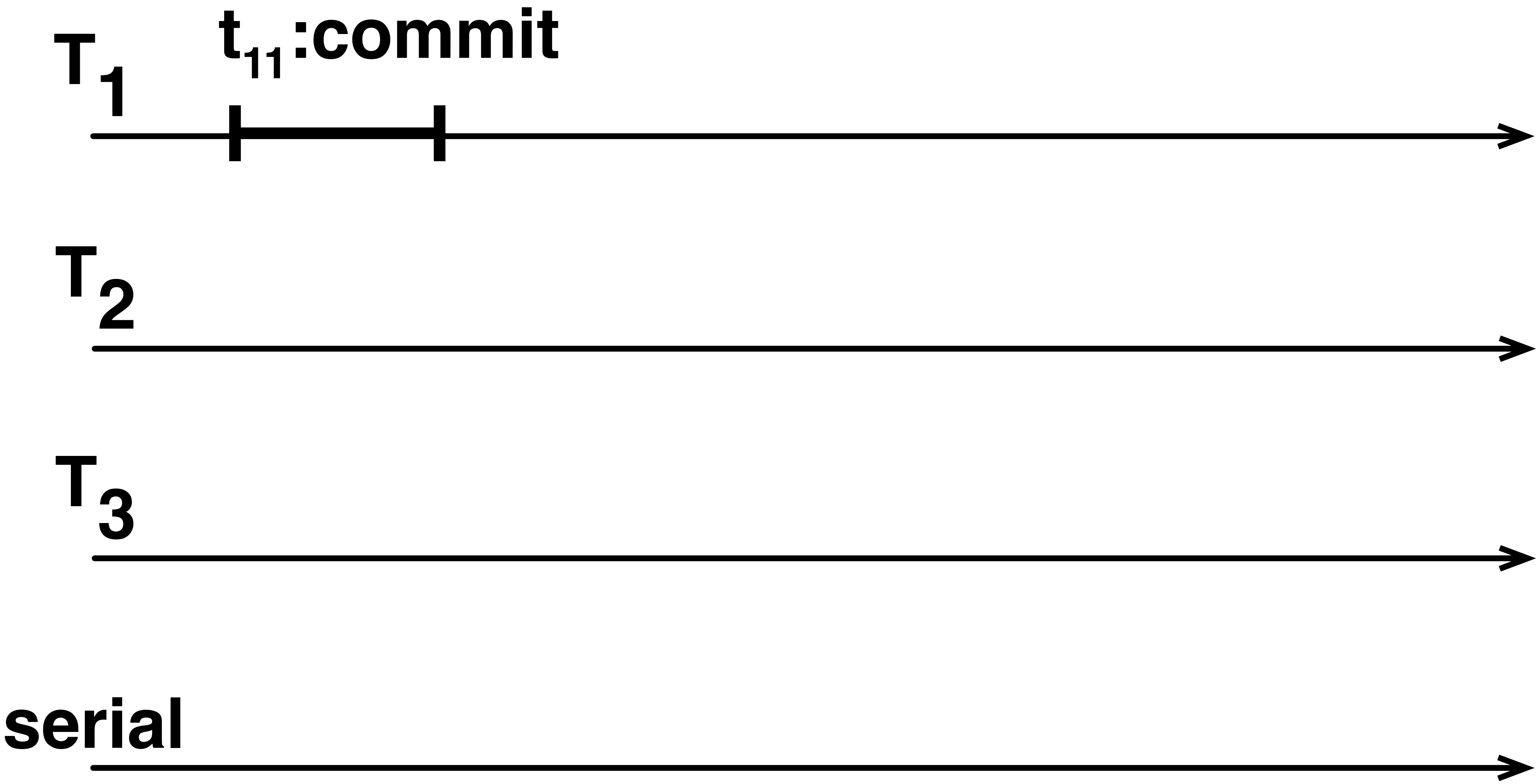
- Committed: instantaneous
- Aborted: never visible
- All: observe consistent state

TM semantics

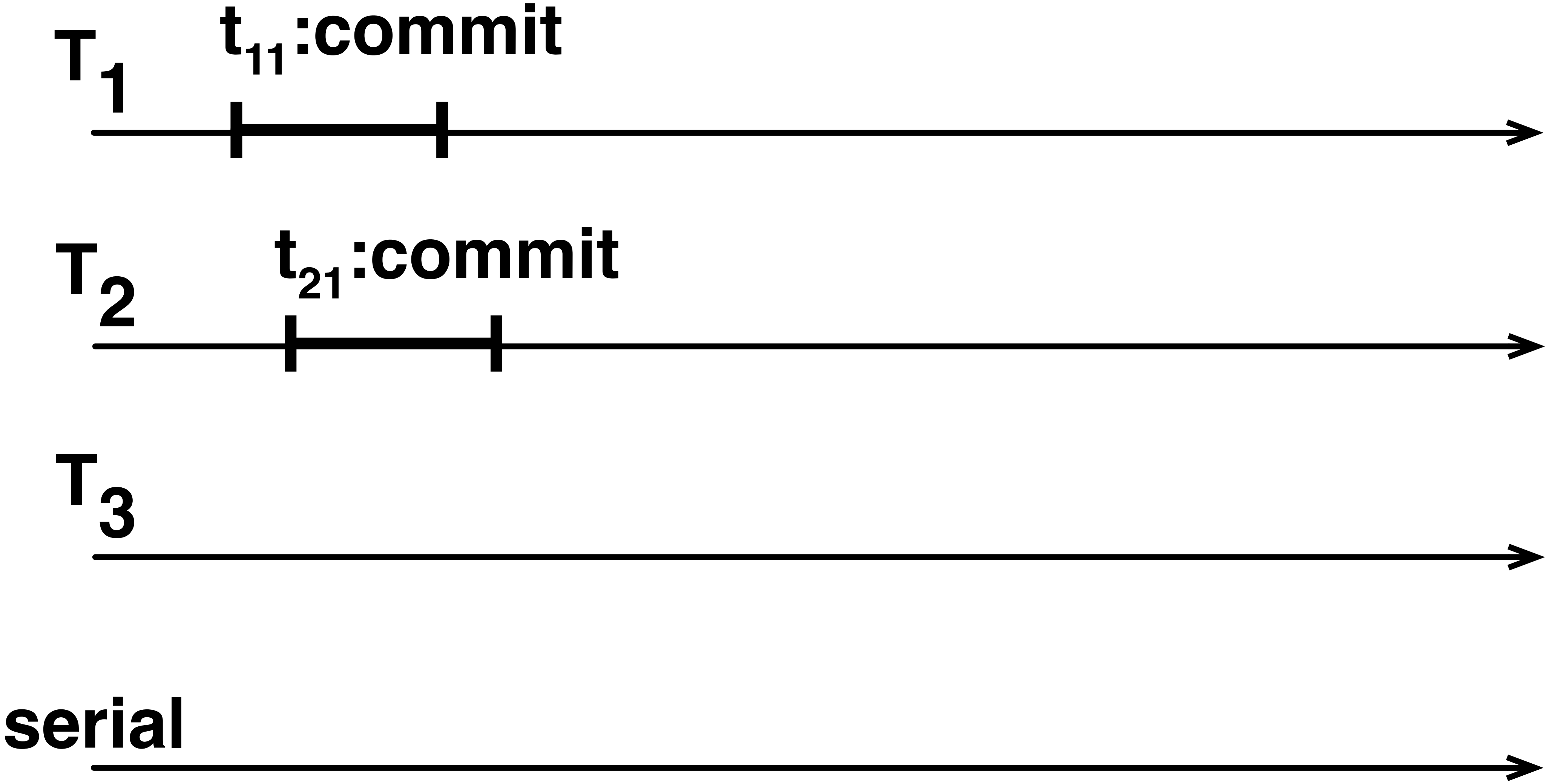
TM semantics

T_1
T_2
T_3
serial

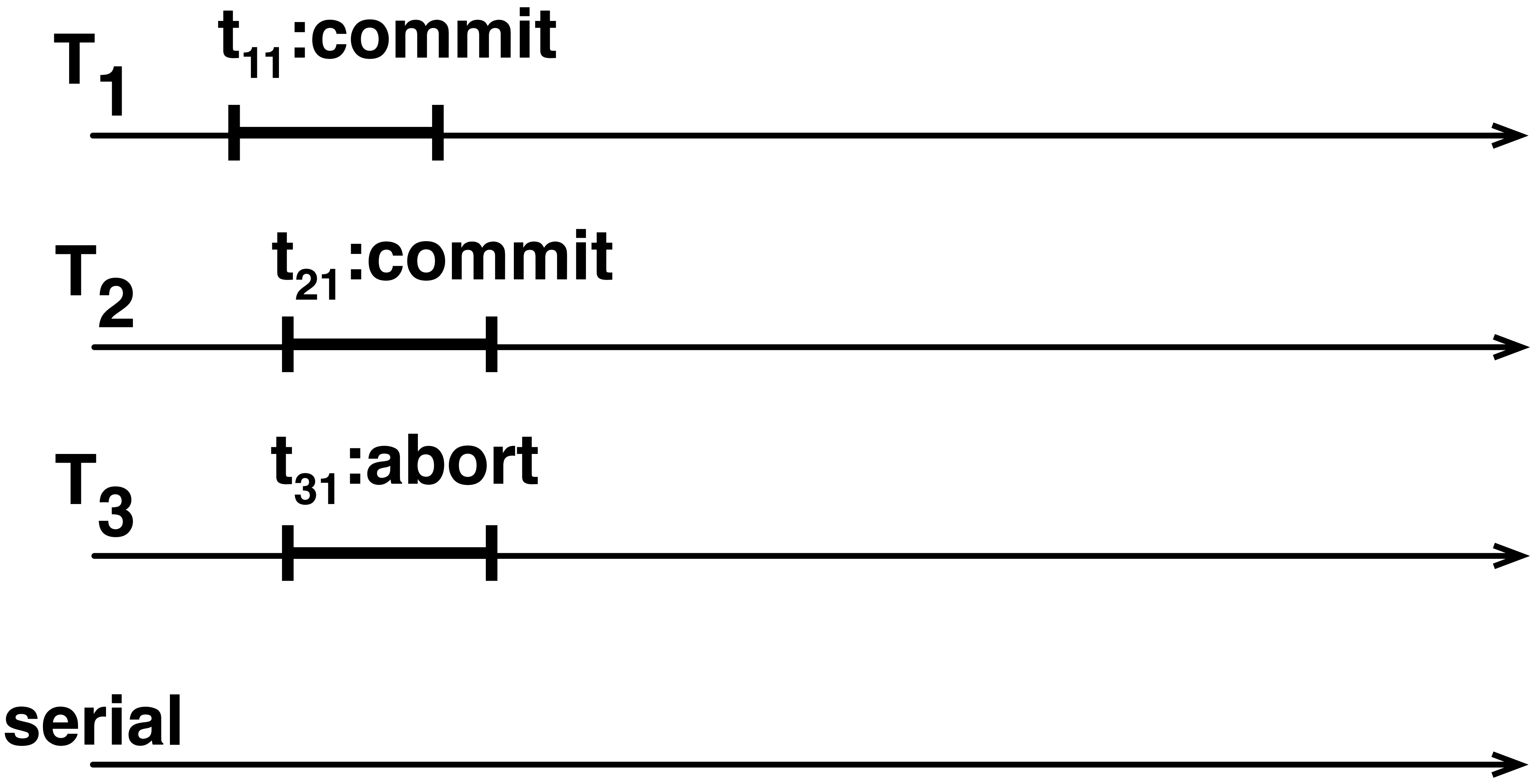
TM semantics



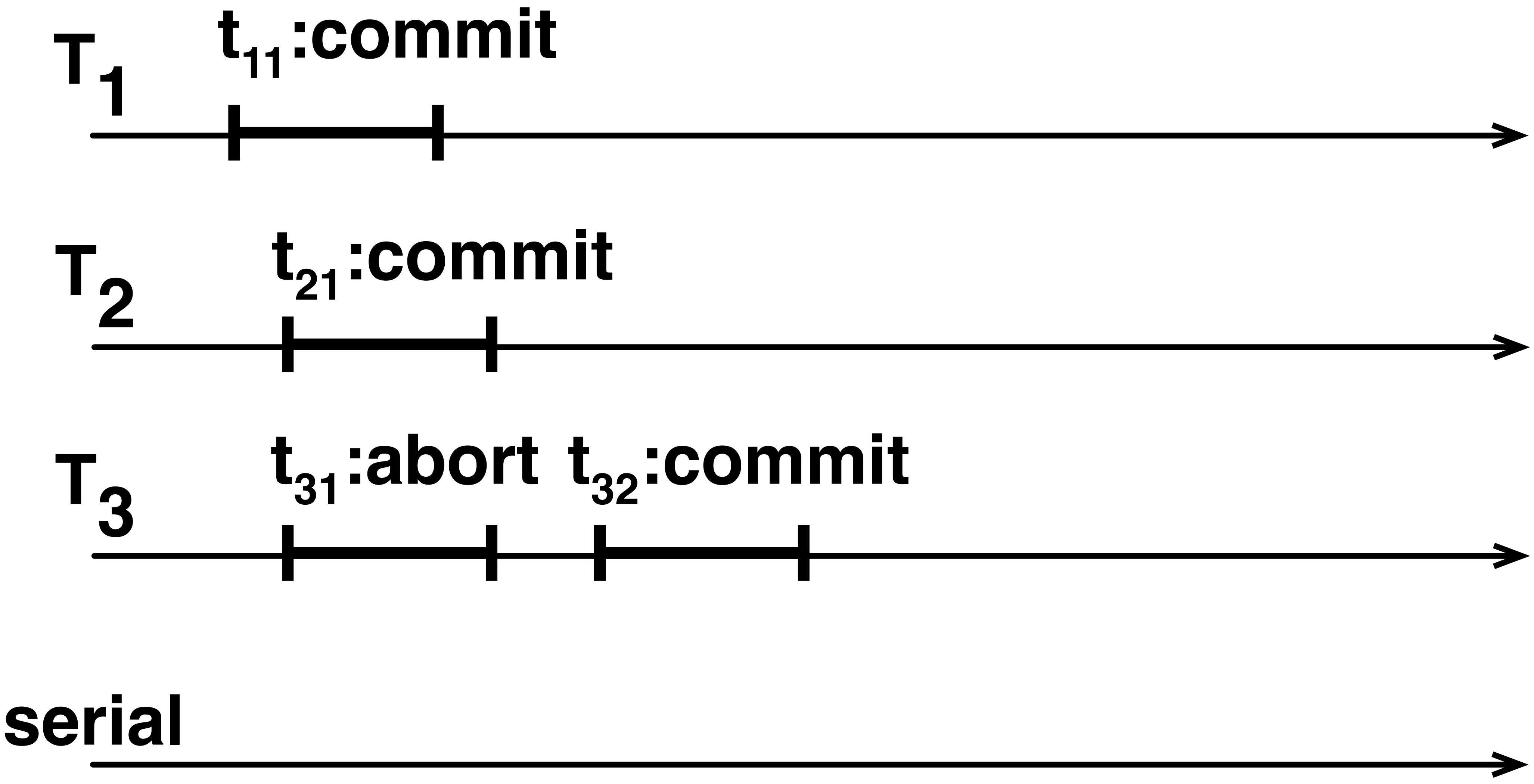
TM semantics



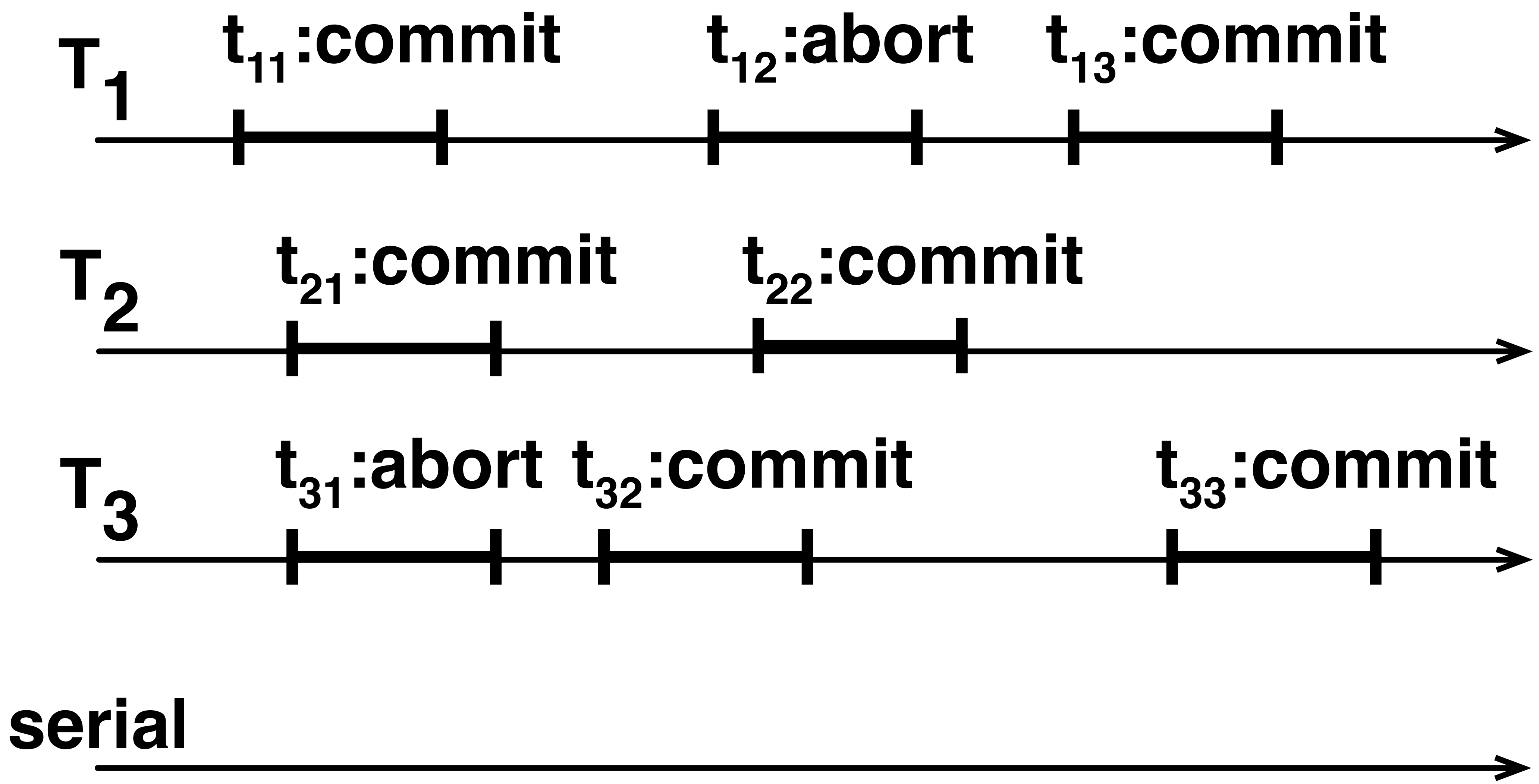
TM semantics



TM semantics



TM semantics



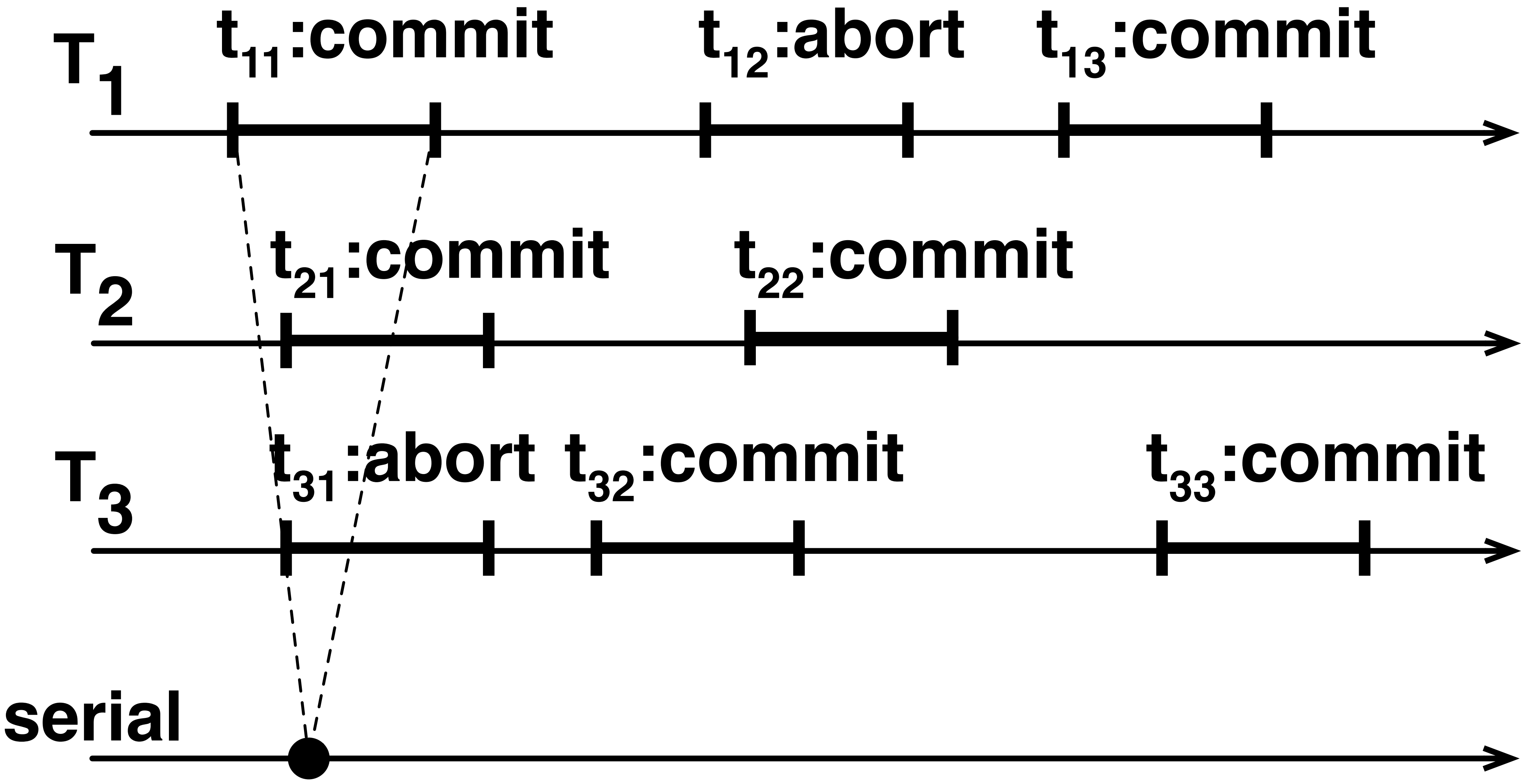
T_1 t_{11}:commit t_{12}:abort t_{13}:commit

T_2 t_{21}:commit t_{22}:commit

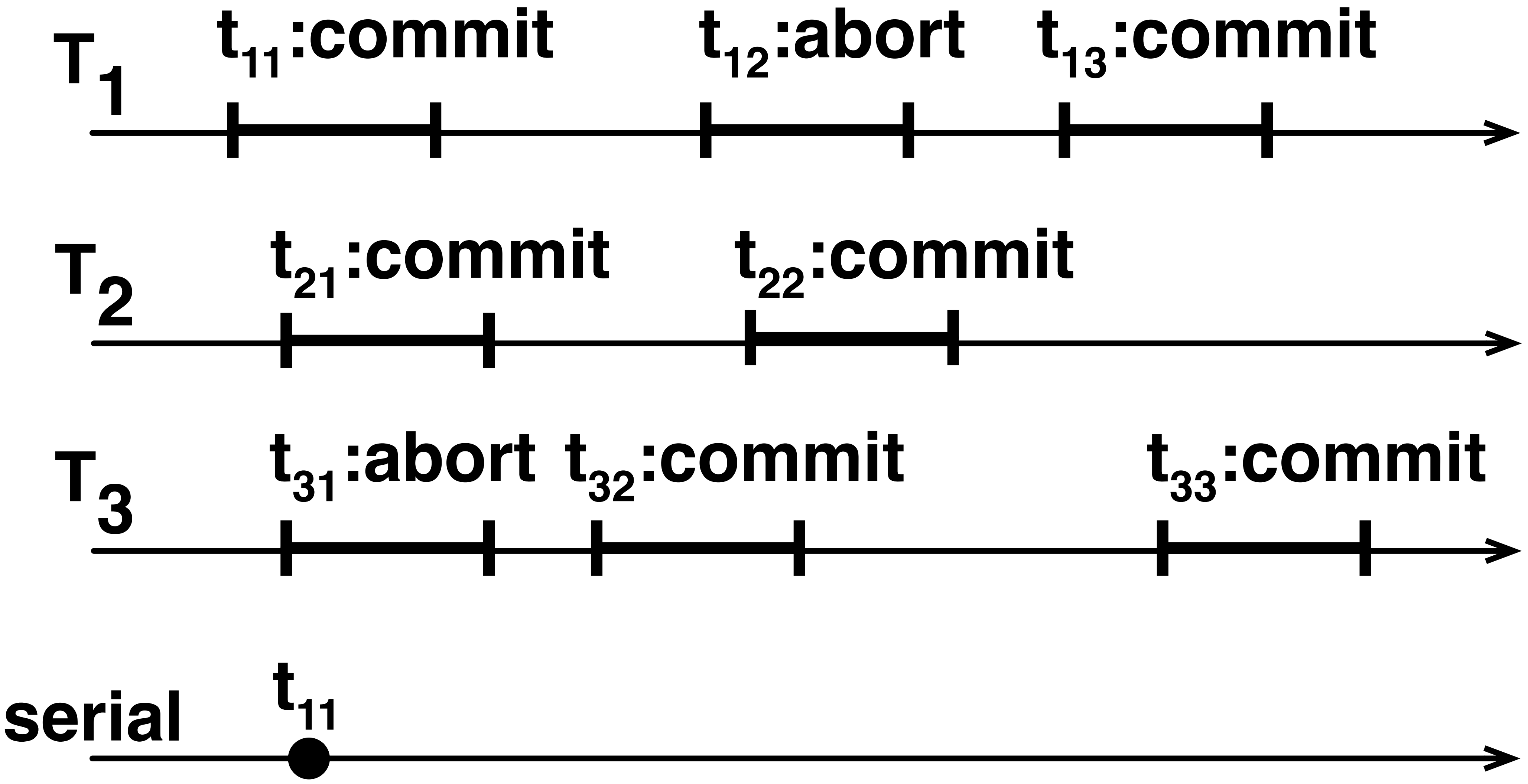
T_3 t_{31}:abort t_{32}:commit t_{33}:commit

serial

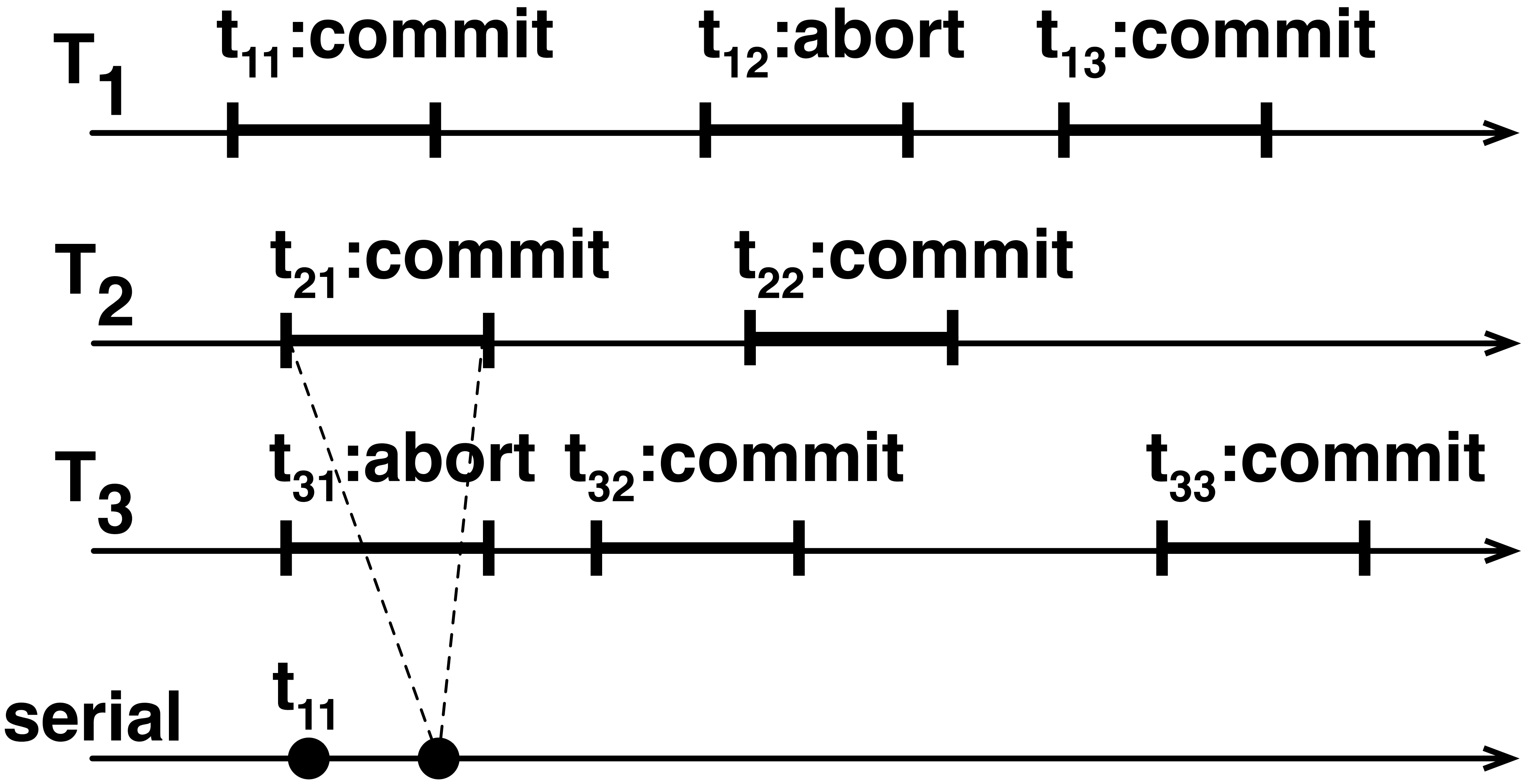
TM semantics



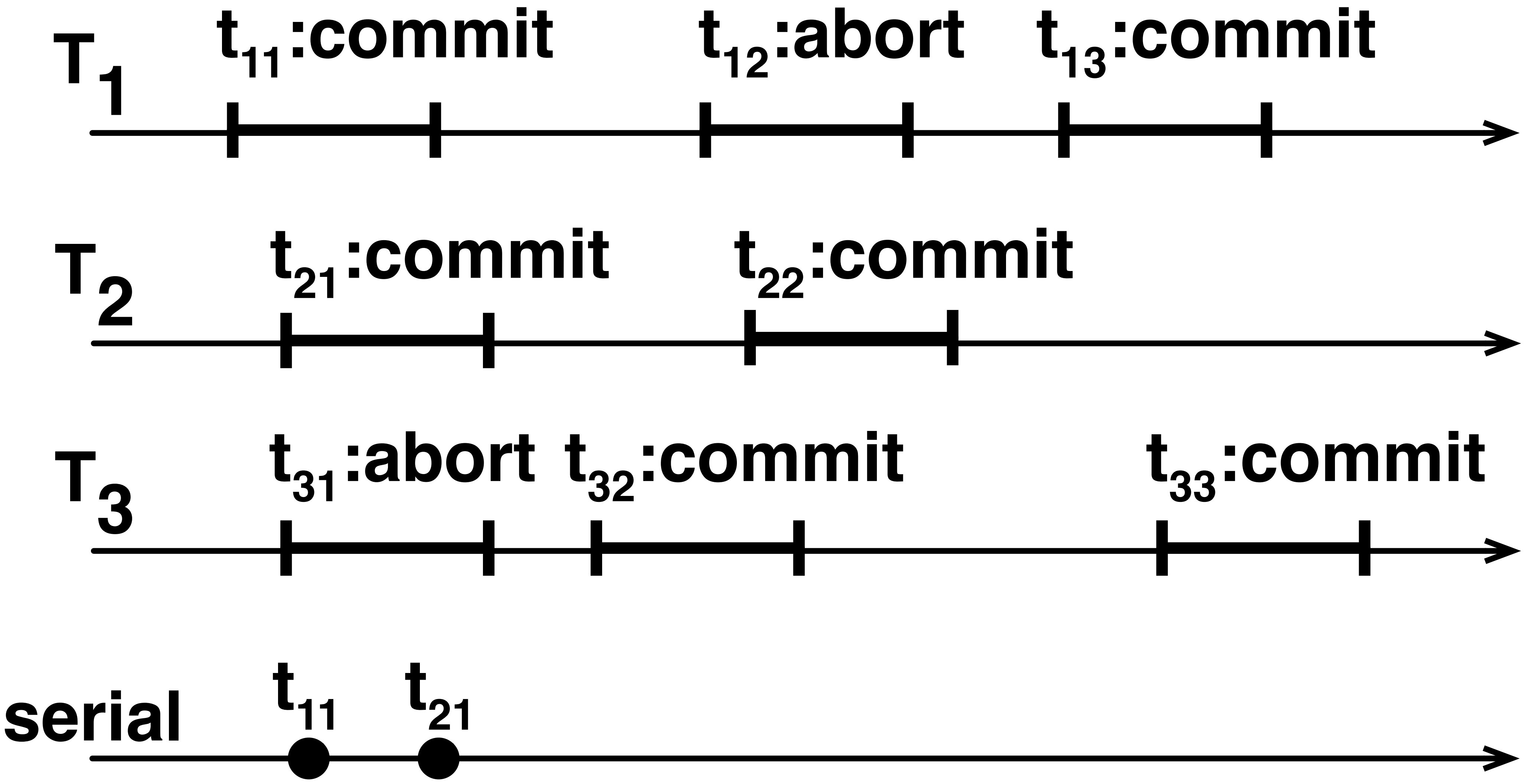
TM semantics



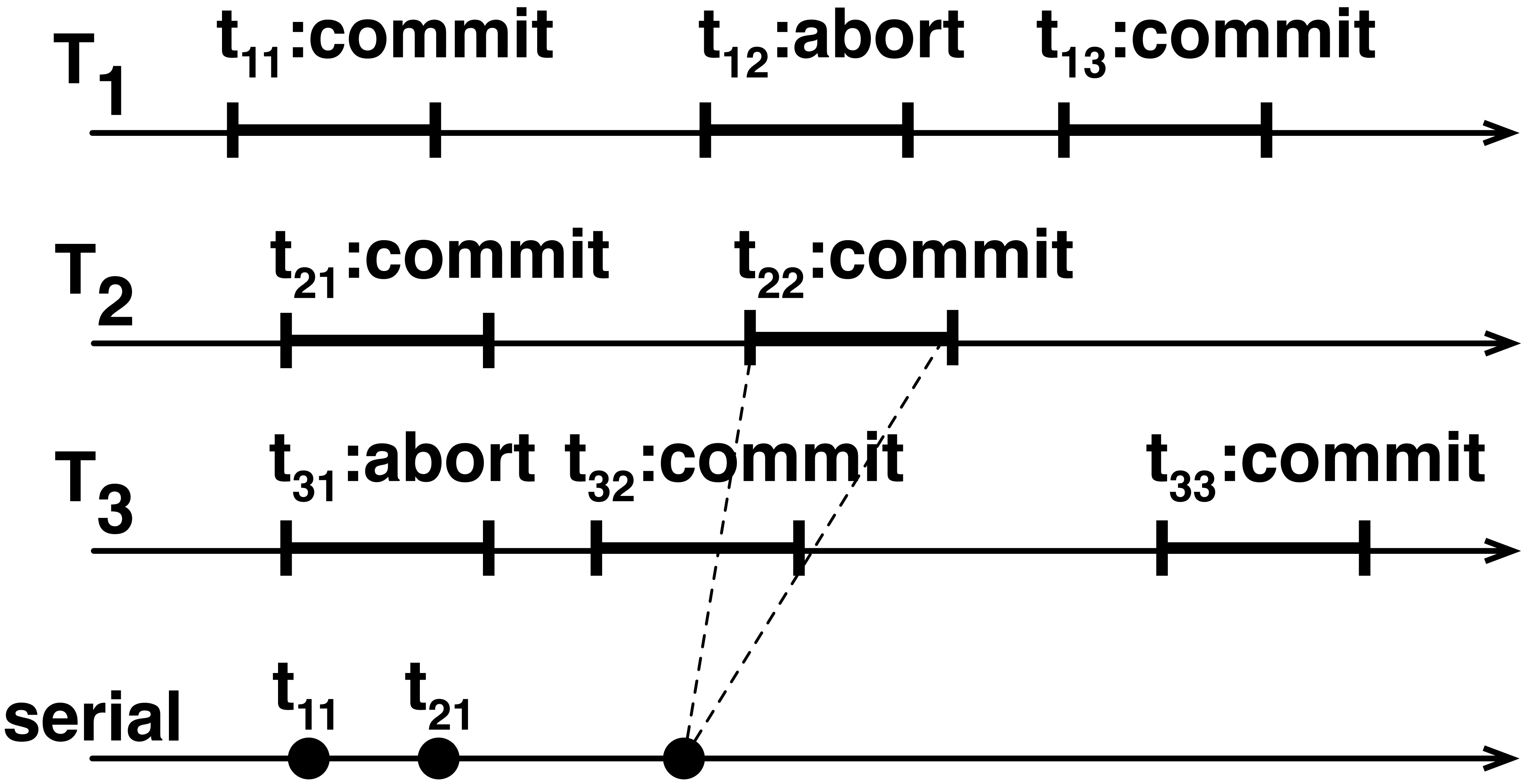
TM semantics



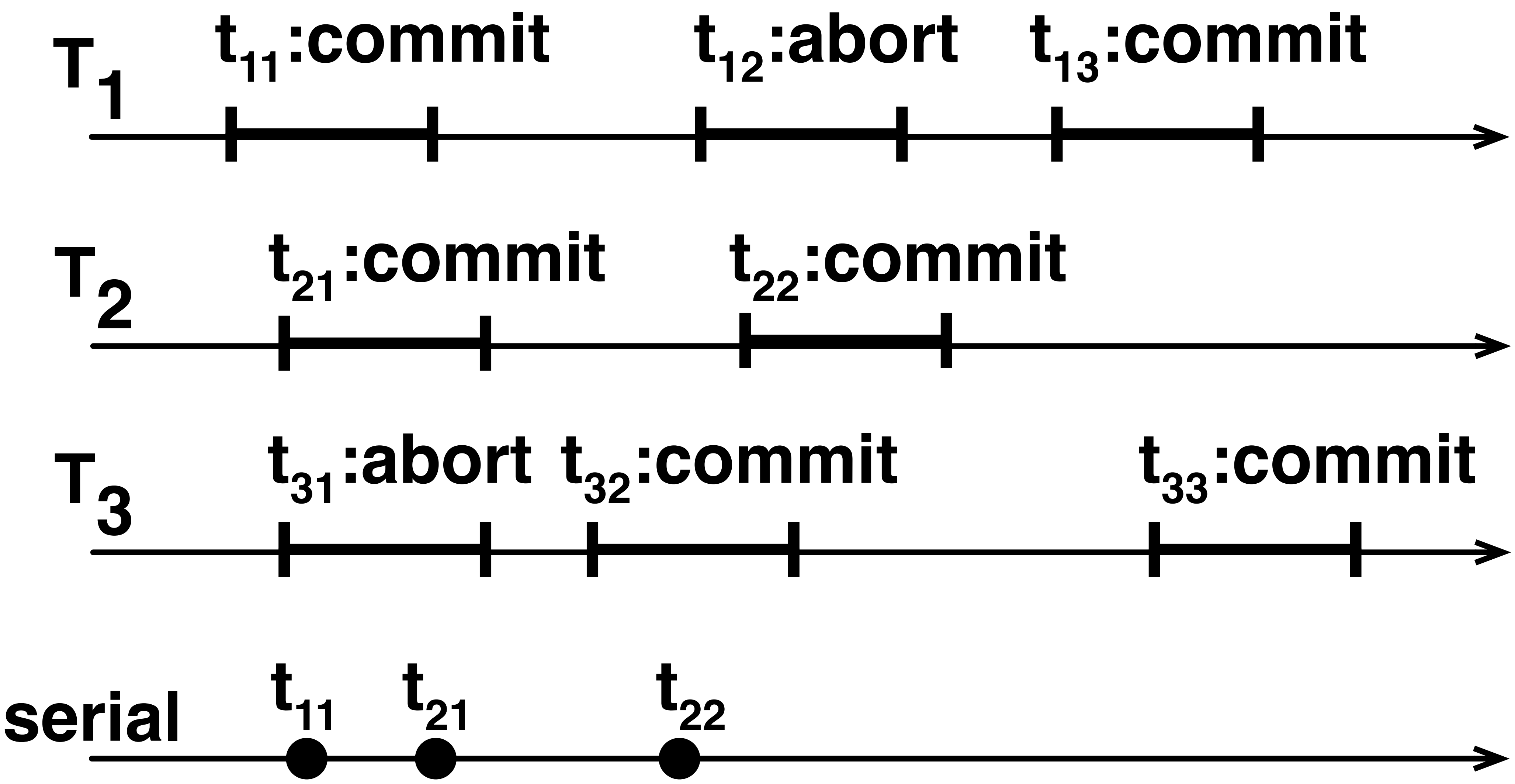
TM semantics



TM semantics



TM semantics



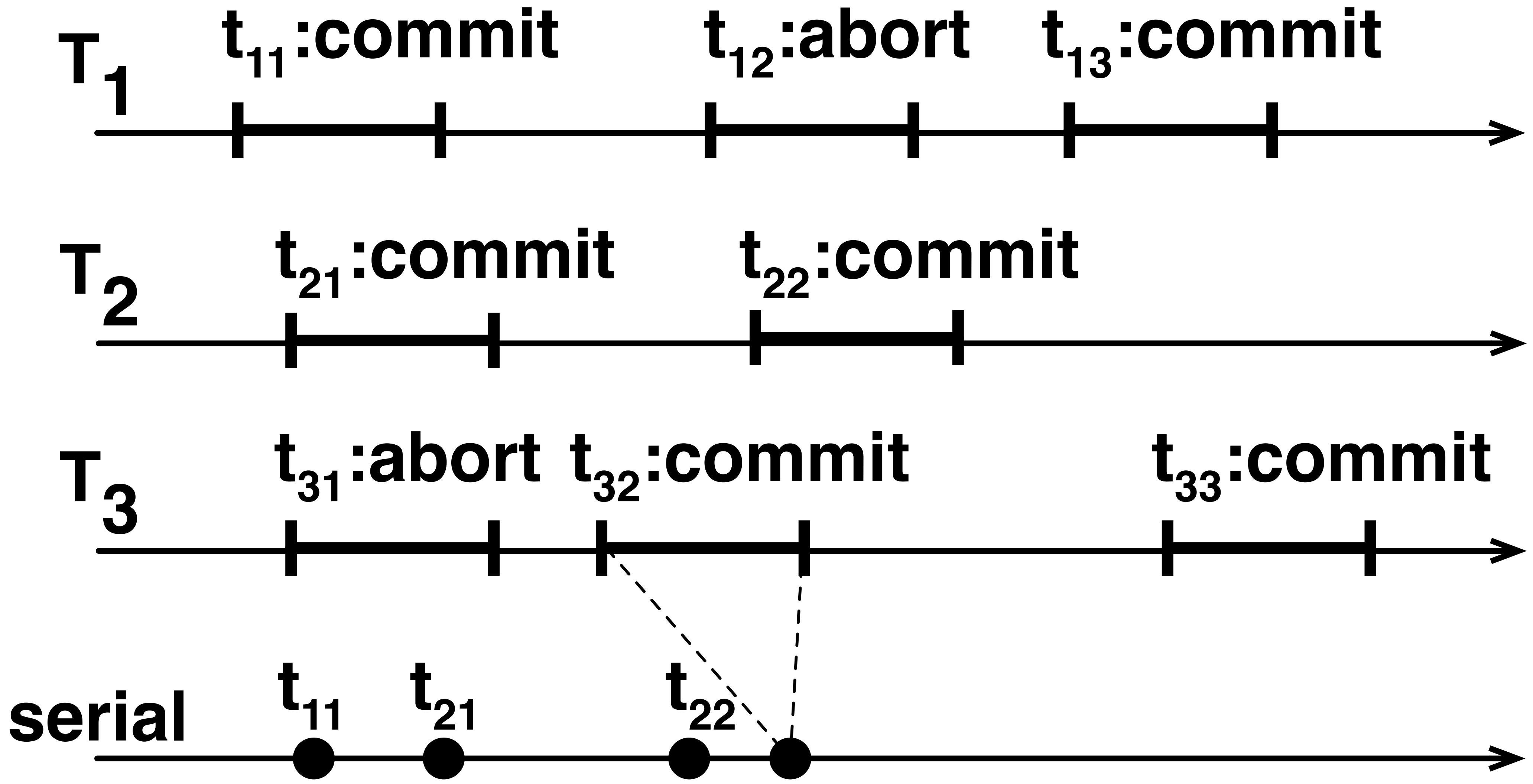
T_1 t_{11}:commit t_{12}:abort t_{13}:commit

T_2 t_{21}:commit t_{22}:commit

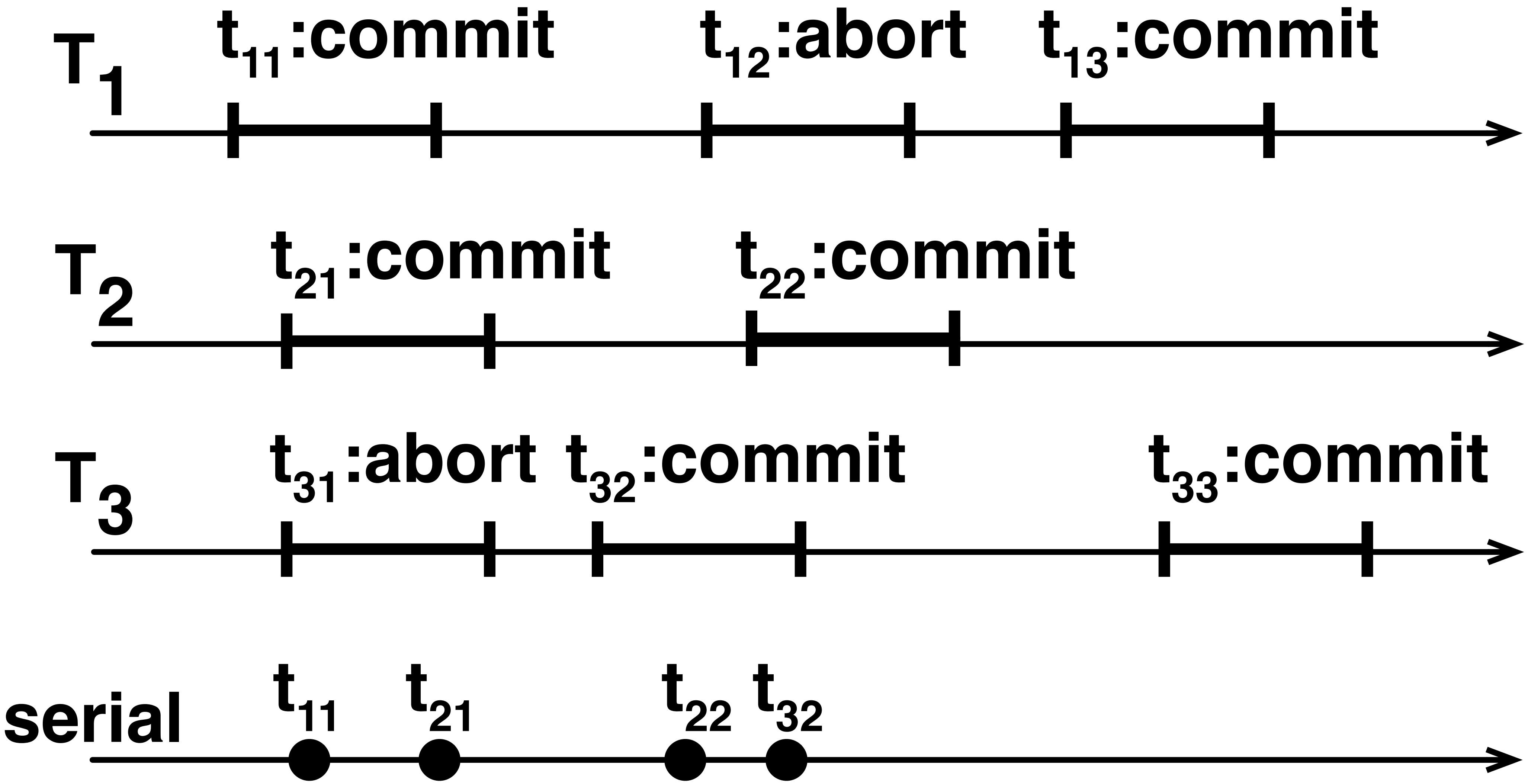
T_3 t_{31}:abort t_{32}:commit t_{33}:commit

serial t_{11} t_{21} t_{22}

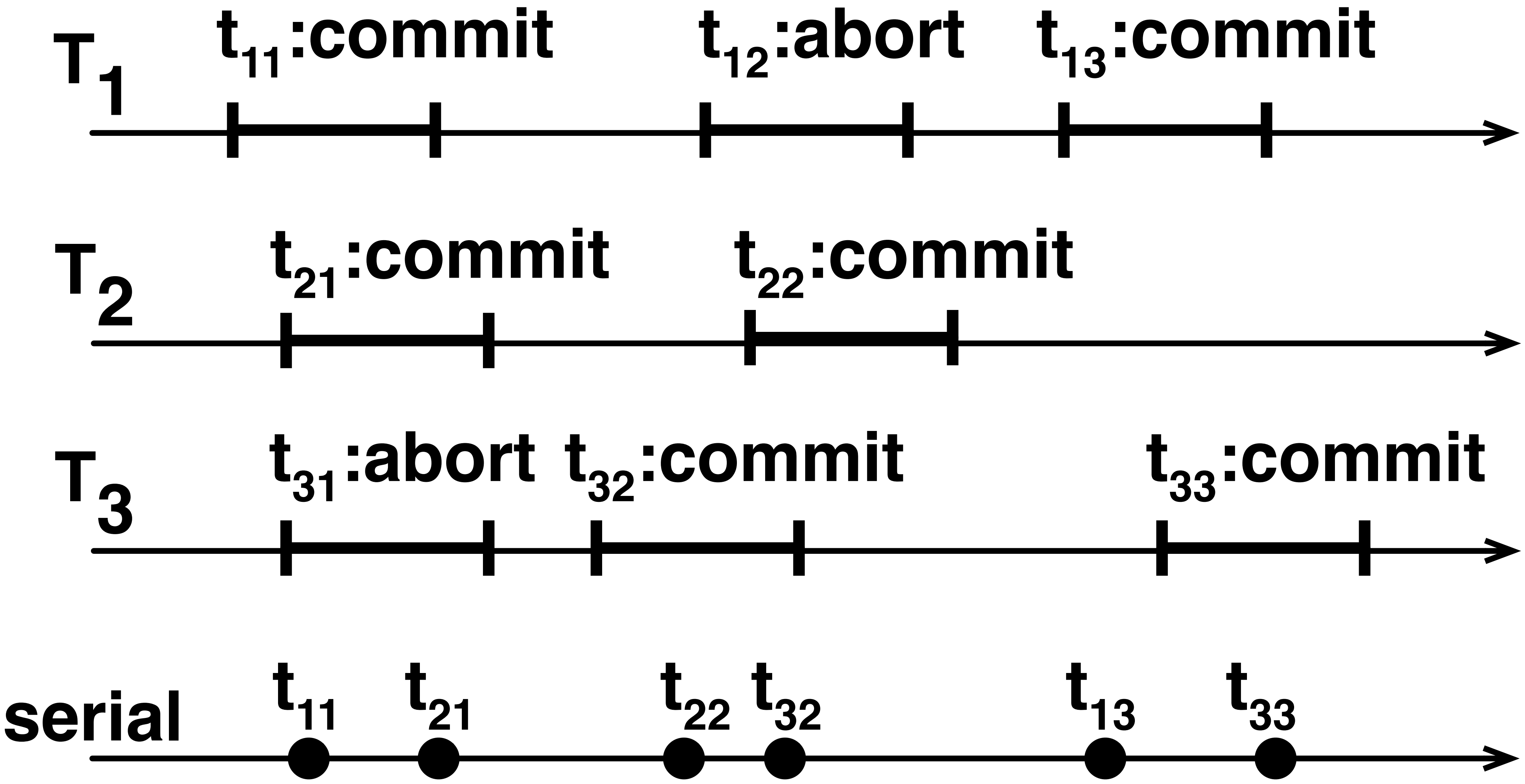
TM semantics



TM semantics



TM semantics


T_1
t_{11}:commit
t_{12}:abort
t_{13}:commit
T_2
t_{21}:commit
t_{22}:commit
T_3
t_{31}:abort
t_{32}:commit
t_{33}:commit
serial
t_{11}
t_{21}
t_{22}
t_{32}
t_{13}
t_{33}