Slate ShareDB

Crash course Slate

Slate's underlying data model: "Immutable" object

Value({

document: Document, ←--- this is the important part to focus. Selection is automatically updated by ops.

selection: Selection,

data: Data,

decorations: List<Decoration>,

})

Operations are exposed in onChange as an array.

```
onChange = ({value, operations}) => {
        console.log(operations); // operations = [{type: ‘insert_text’, }]
        this.setState({value: value});
}
```

important to note that the value passed here has already the operations applied.

Operations in slate are one of 13 types which can be applied and inverted

```
//   insert_text: ['path', 'offset', 'text', 'marks', 'data'],
//   remove_text: ['path', 'offset', 'text', 'marks', 'data'],
//   add_mark: ['path', 'offset', 'length', 'mark', 'data'],
//   remove_mark: ['path', 'offset', 'length', 'mark', 'data'],
//   set_mark: ['path', 'offset', 'length', 'properties', 'newProperties', 'data'],
//   insert_node: ['path', 'node', 'data'],
//   merge_node: ['path', 'position', 'properties', 'target', 'data'],
//   move_node: ['path', 'newPath', 'data'],
//   remove_node: ['path', 'node', 'data'],
//   set_node: ['path', 'properties', 'newProperties', 'data'],
//   split_node: ['path', 'position', 'properties', 'target', 'data'],
//   set_selection: ['properties', 'newProperties', 'data'],
//   set_value: ['properties', 'newProperties', 'data'],
```

Why OT?

1. Allow concurrent edits to users.
2. Assure convergence of their documents.

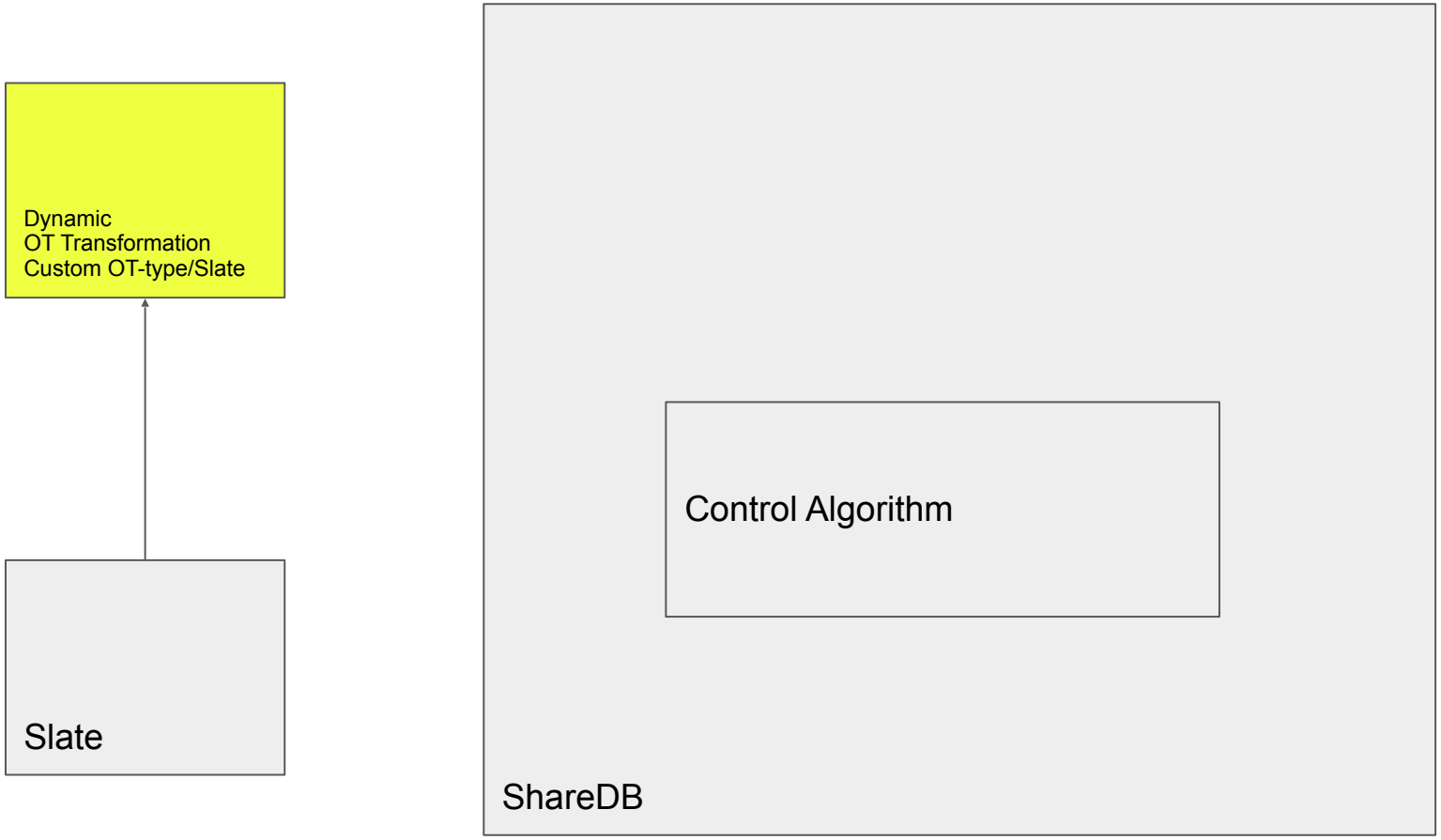
Refresher OT

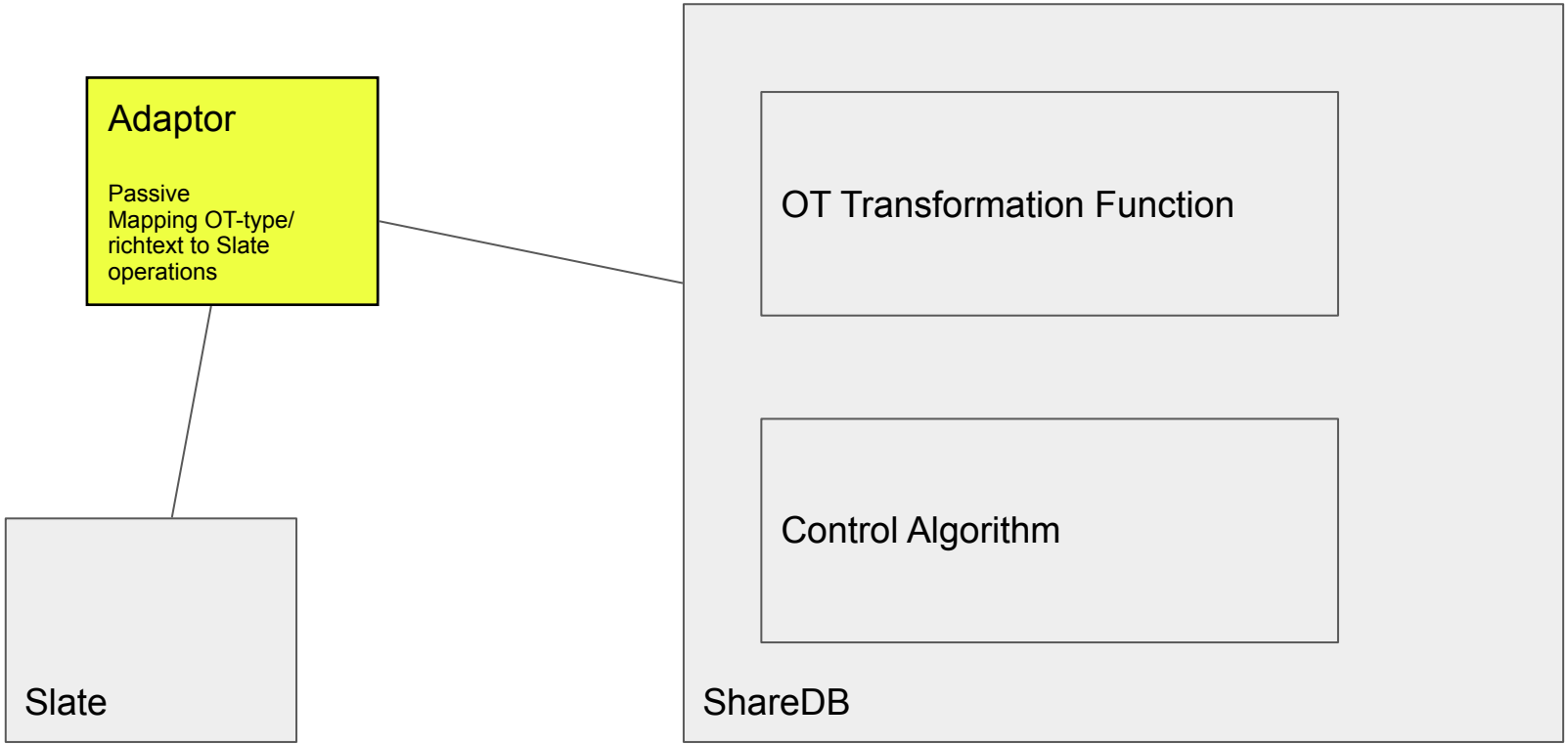
1. Every change represented as an operation.
2. Transformation functions that produce new operations that preserve the first operation's intended change.
3. Many settings for OT, centralised/distributed. Client-server approach reduces the workload on the server, and simplifies the thinking to only two actors (ShareDB).

Main points:

1. Control algorithm (when to apply which ops) ---> handled by ShareDB

2. Transformation function (how to apply incoming ops) ---> custom by us.


Dynamic
OT Transformation
Custom OT-type/Slate
Slate
Control Algorithm
ShareDB


Adaptor
Passive
Mapping OT-type/
richtext to Slate
operations
OT Transformation Function
Control Algorithm
Slate
ShareDB

Custom type VS adaptor?
Not sure which is best.

Custom type atm.

Refresher OT

1. For a client-server acknowledgment model. Transformation has to satisfy CP1 property.

   i.e: op1 * transform(op2, op1) === op2 * transform(op1, op2),

   needs to be tested by generating a lot of random operations and using npm module fuzzer

2. For undo, operations must be invertible and stored in history. Done by slate.

Simplest transformation example of only insert on a string:

```
Tii(Ins[p1,c1], Ins[p2, c2]) {

  if (p1 < p2  or (p1 = p2 and u1 > u2))

     return Ins[p1, c1];

  else return Ins[p1+1, c1];
```

Refresher ShareDB

Client server model where client has to wait for server acknowledgements.

Server stores all operations ever accepted.

------> Common “history”. When operations are received from the server, transform operations that are in buffer with the incoming op. (Done automatically by ShareDB).

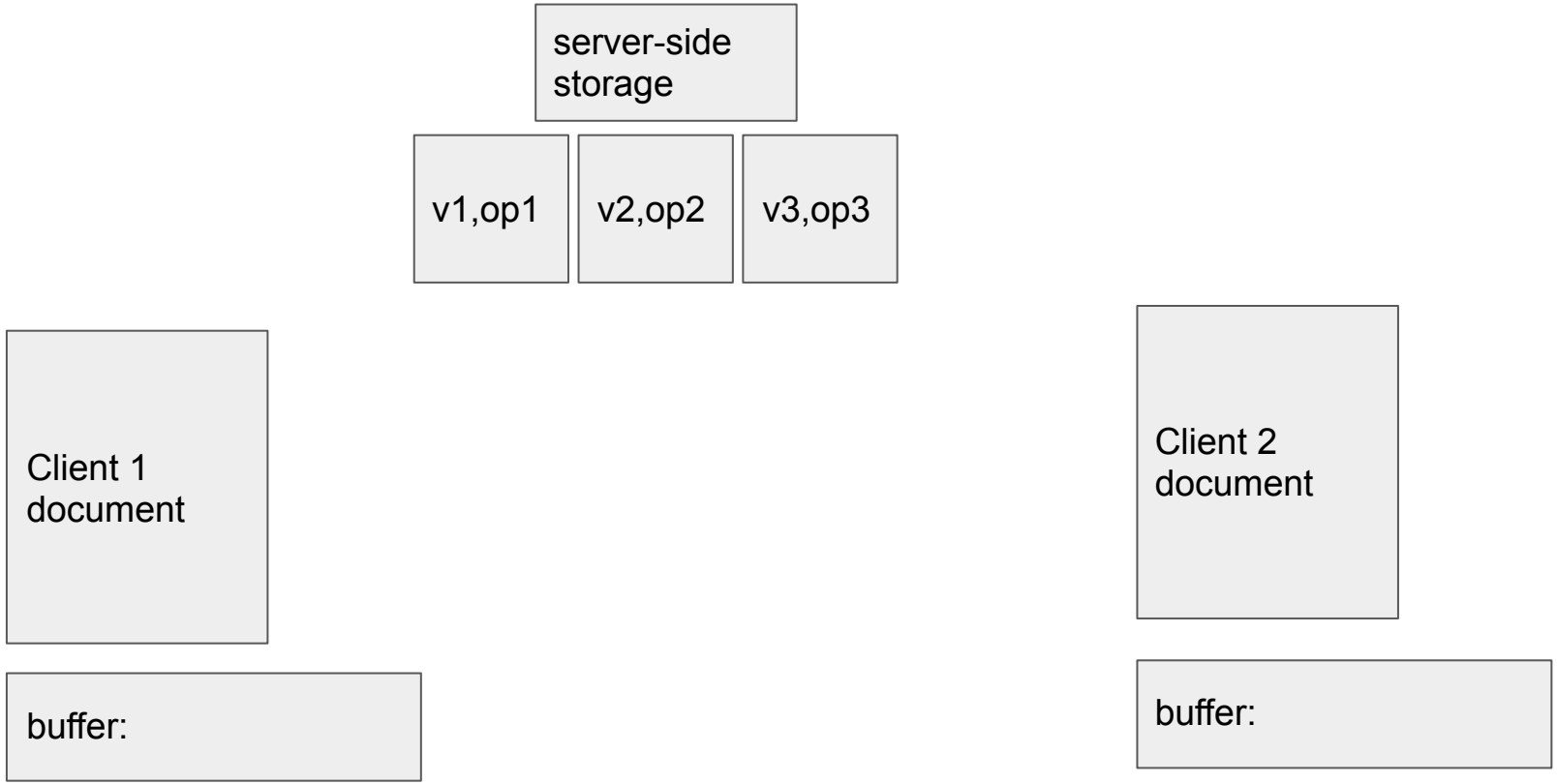
Refresher ShareDB

ShareDB allows custom operation types to be defined through:

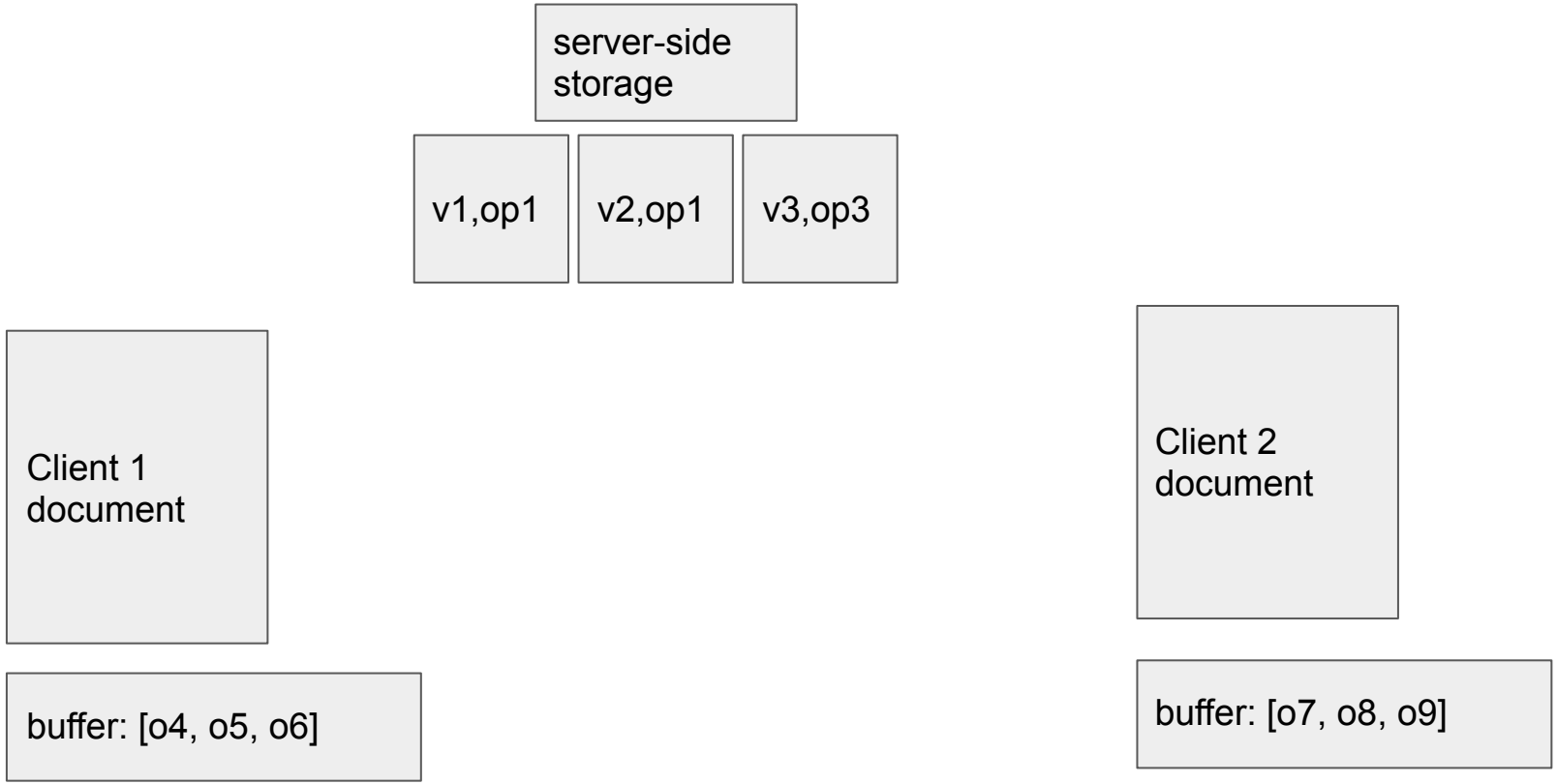
sharedb.register(type);

See code... (implemented skeleton).

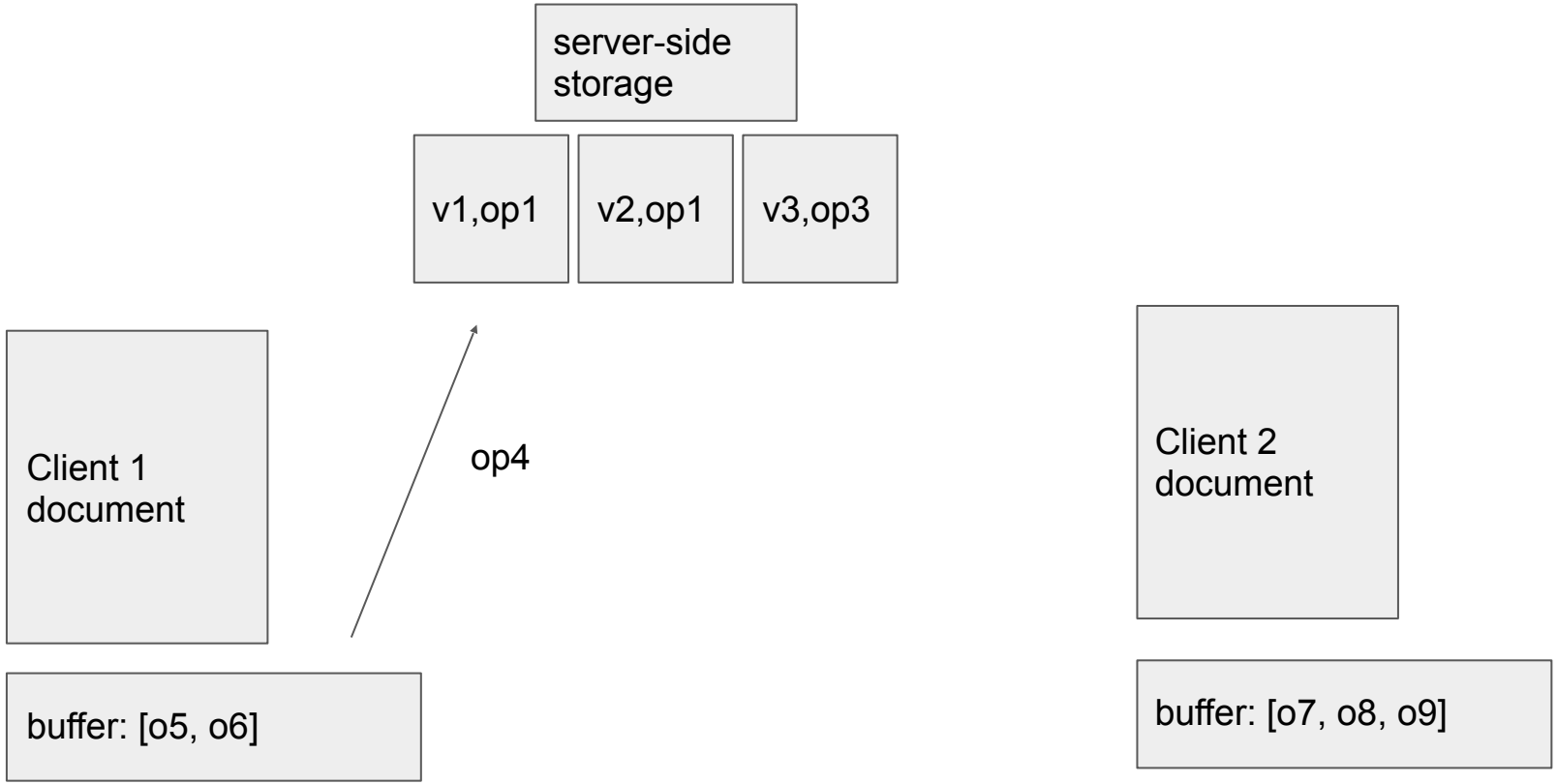
Workings of ShareDB



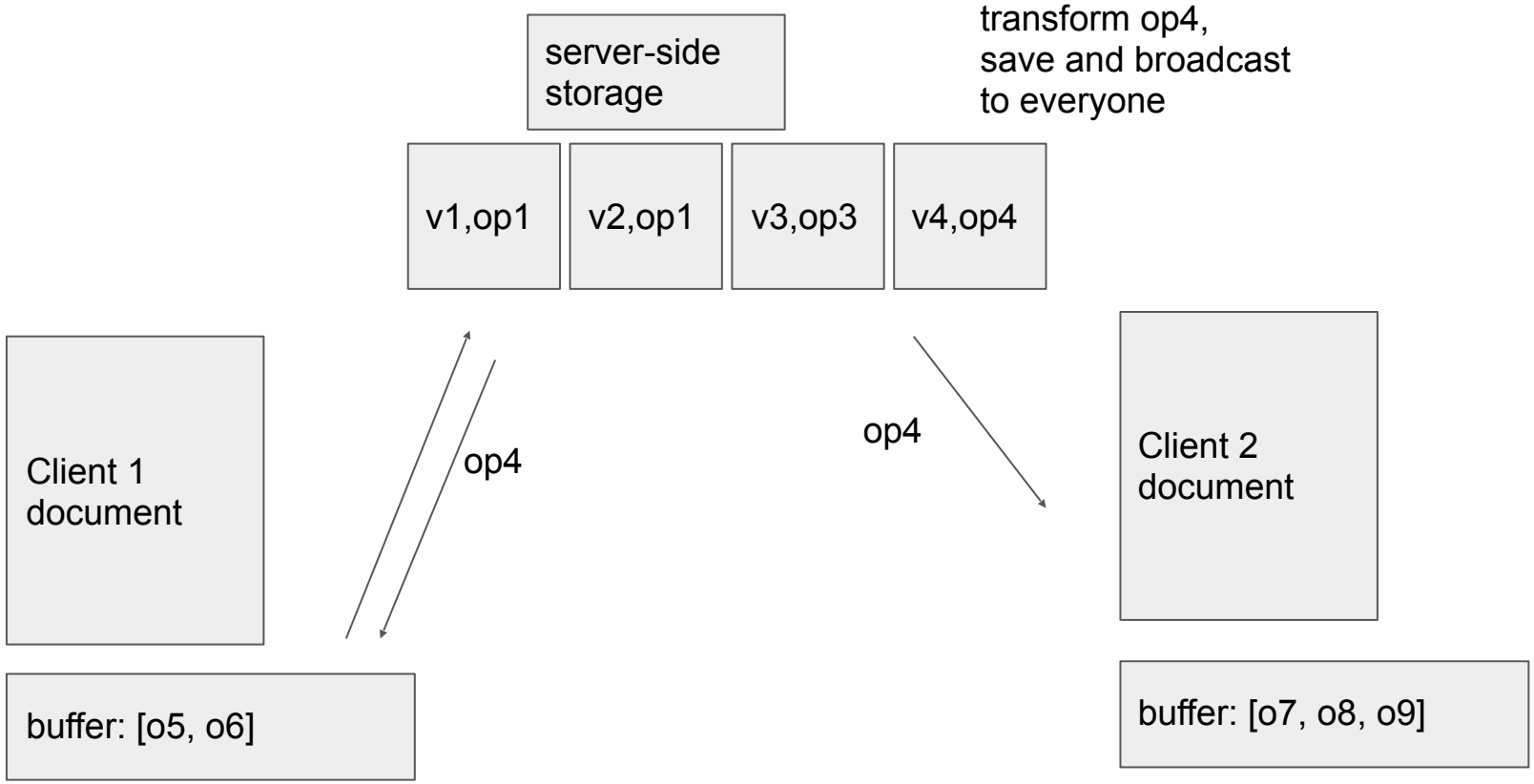
Workings of ShareDB



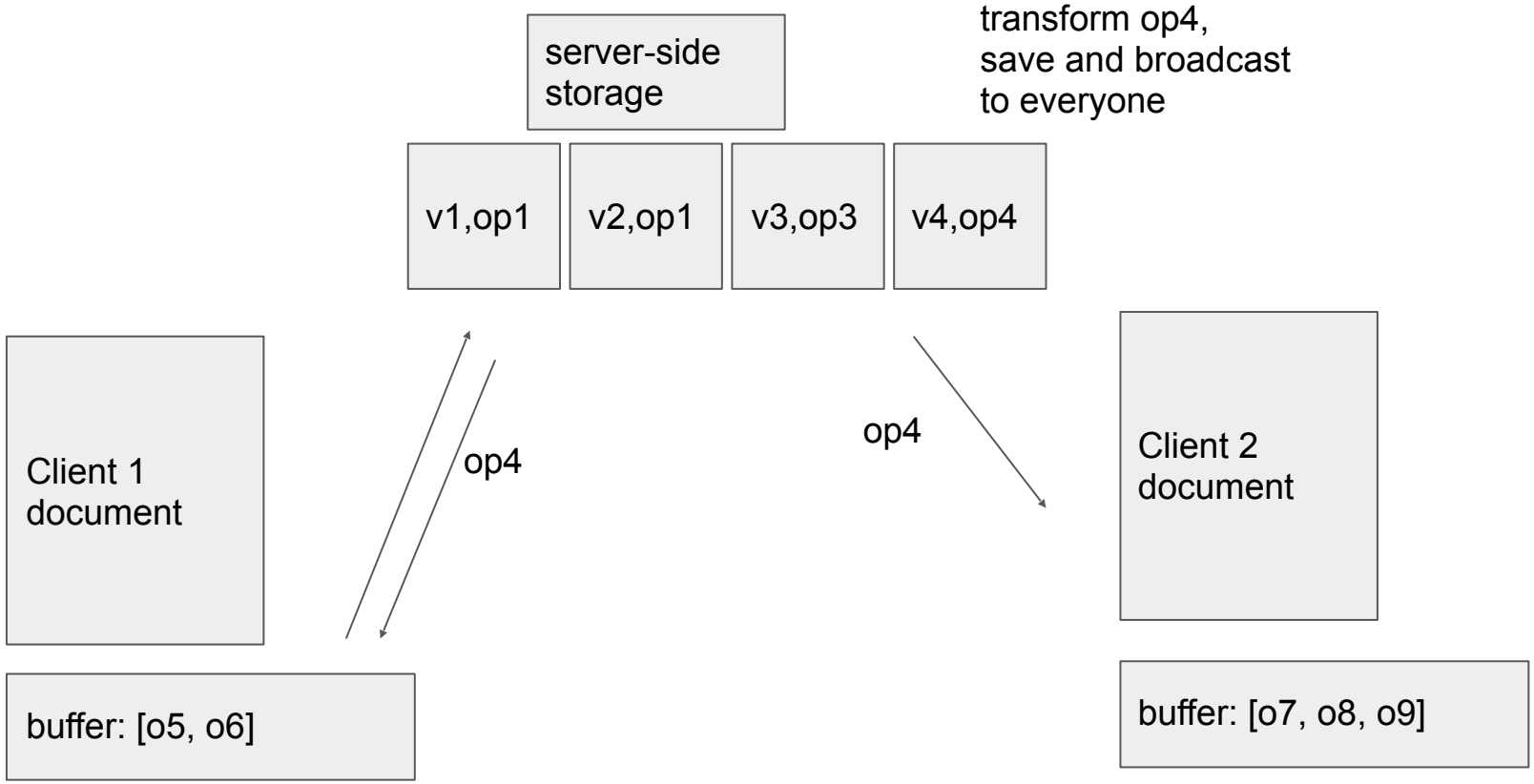
Workings of ShareDB



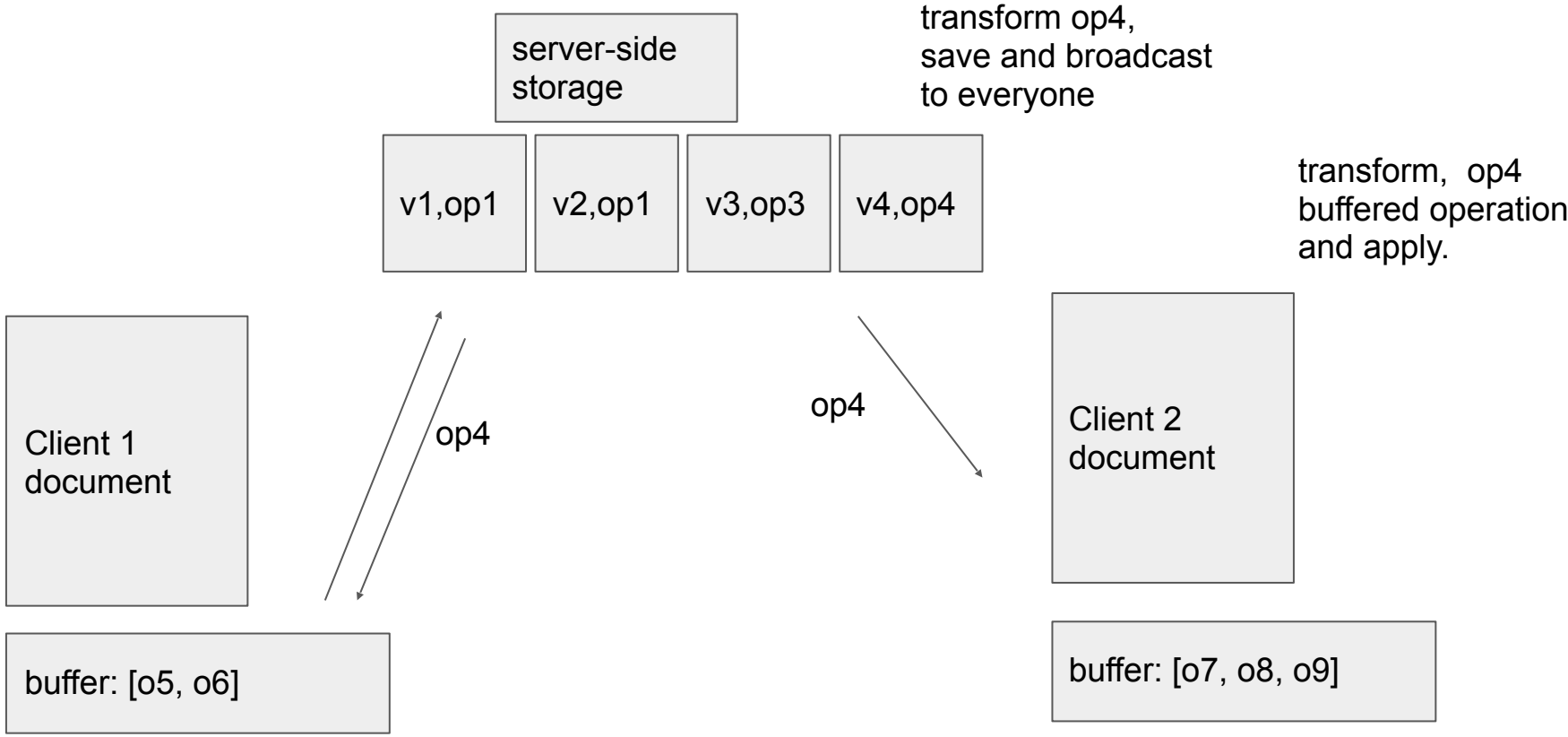
Workings of ShareDB



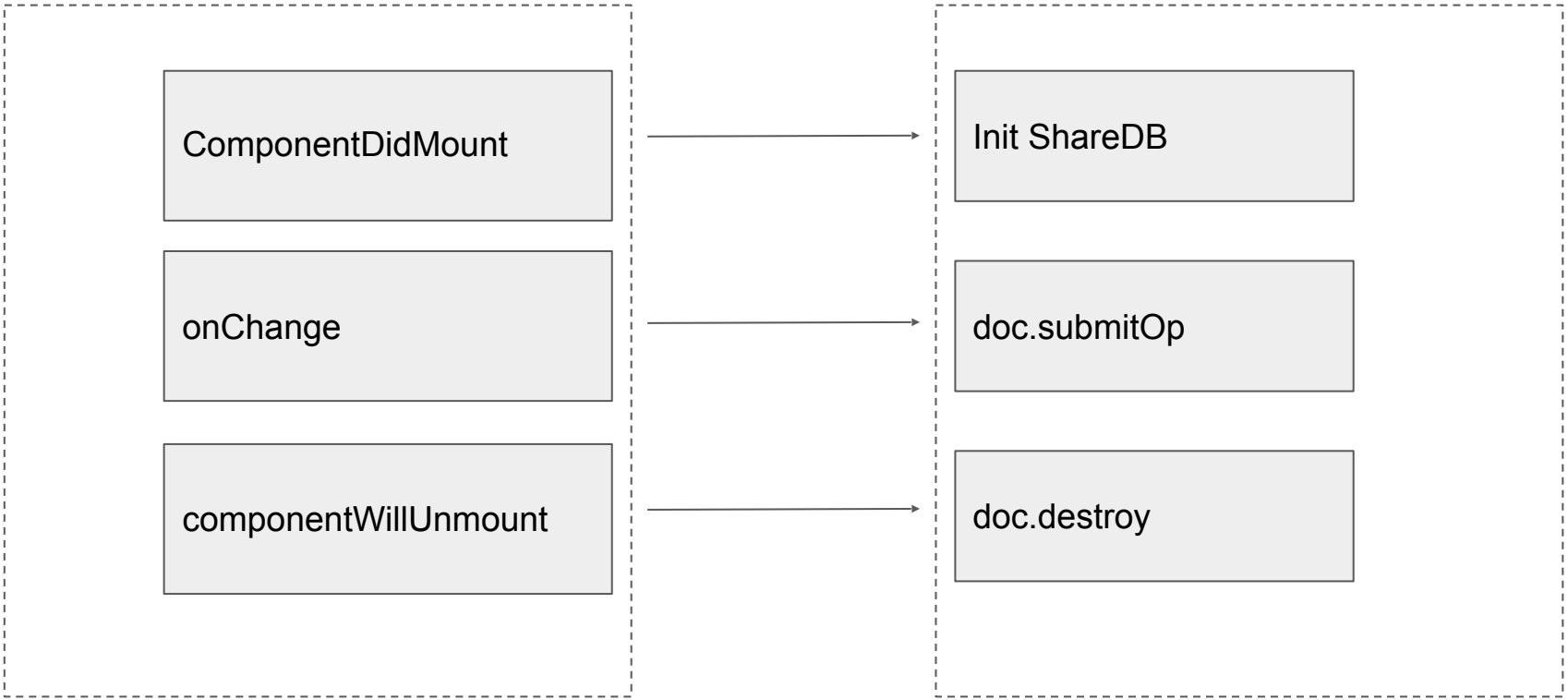
Workings of ShareDB



Workings of ShareDB



Interfaces ShareDB and Slate



Resources

Complete overview of OT: http://www3.ntu.edu.sg/home/czsun/projects/otfaq/