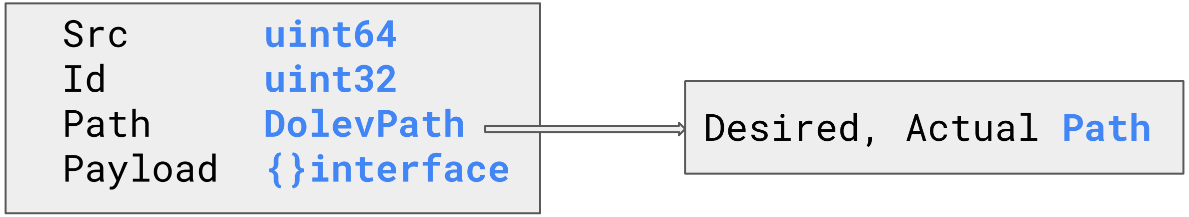
Status update wk10

Tim Anema

Previous week

- Evaluation
- Paper
- Peer review
- Waiting for feedback

Optimizations - DolevKnown message



```
Src      uint64
Id       uint32
Path     DolevPath
Payload  {}interface
```

```
Desired, Actual Path
```

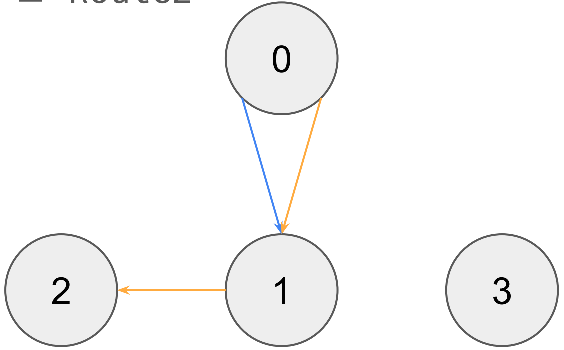
Close to normal Dolev message, with the exception of the added ‘desired path’

ORD.1 - Filtering subpaths

```
Route1 = [0,1] ✗
```

```
Route2 = [0,1,2]
```

Route1 ⊆ Route2



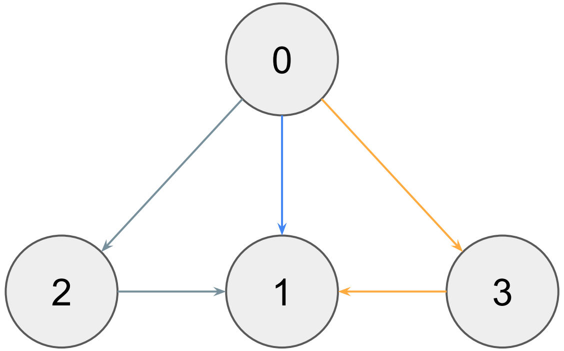
Subpaths are omitted from the routing table

ORD.2 - Single hop neighbour

```
Route1 = [0,1]
```

```
Route2 = [0,3,1] ✗
```

```
Route3 = [0,2,1] ✗
```

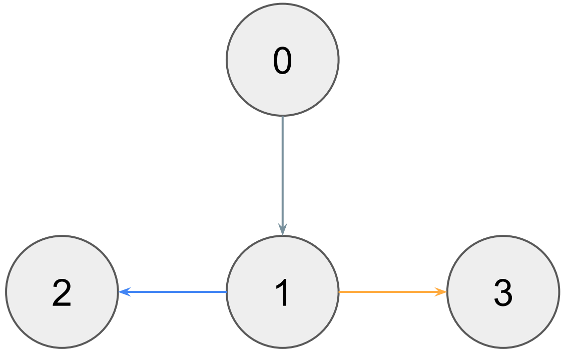


Neighbours only need a single route, which can be the direct hop

ORD.3 - Combine next hop when broadcasting

```
Route1 = [0,1,2]
```

```
Route2 = [0,1,3]
```



Messages with the same next hop(s) can be transmitted in a single message

Message is modified to include multiple paths

```
Src       uint64
Id        uint32
Path      []DolevPath
Payload   {}interface
```

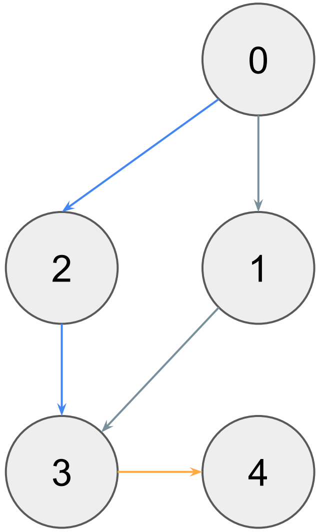
```
Desired, Actual Path
```

Possible future improvement to reduce bandwidth usage:
The actual path is identical until a split, only include it once

ORD.4 - Reuse paths

```
Route1 = [0,2,3]
```

```
Route2 = [0,2,3,4]
```



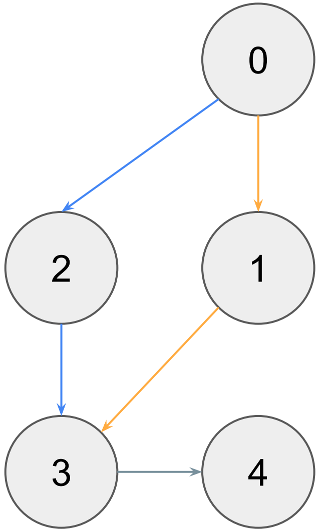
When ORD.3 is active, it is beneficial to reuse the same paths as long as possible.

During the creation of routes, priority is given to routes which reuse edges, by increasing the weight of unused edges (heuristic, optimal solution also exists)

ORD.5 - Delayed relay and merge

```
Route1 = [0,2,3,4]
```

```
Route2 = [0,1,3,4]
```



Messages are not relayed until delivered, gives opportunity to merge messages with same next hop. Some messages are marked as a priority message, which means they will always be relayed to ensure overall delivery.

Message is modified to include multiple paths. Paths include priority mark

```
Src        uint64
Id         uint32
Path       []DolevPath
Payload    {}interface
```

$\Longrightarrow$

```
Desired, Actual Path
Priority        bool
```

ORD.6 - Merge messages with identical payload

```
Src     = 0
Id      = 0
Payload = [79,65,65,74]
Route   = [0,2,3,4]
```

```
Src     = 1
Id      = 0
Payload = [79,65,65,74]
Route   = [1,5,3,4]
```

```
Src     = 3
Id      = 0
Payload = Wrapper{...}
Route   = [3,4]
```

Extension of ORD.5: Messages sharing the same payload can also be merged when possible.

Message re-transmitted with wrapper as payload. Can be used to reconstruct original messages:

```
Msgs    []Wrapper
Payload {}interface
```

$\Longrightarrow$

```
Src   uint64
Id    uint32
Paths []DolevPath
```

ORD.7 - Implicit desired paths

```
Src     = 0
Id      = 0
Payload = [79,65,65,74]
Actual  = [0,2]
```

```
Src = 0
Routes = {
    [0,2],[0,2,3],[0,2,4],
    ...
}
```

```
Implicit next: [],[3],[4]
```

Depending on other optimizations (ORD.3), one or all implicit valid next paths may be chosen

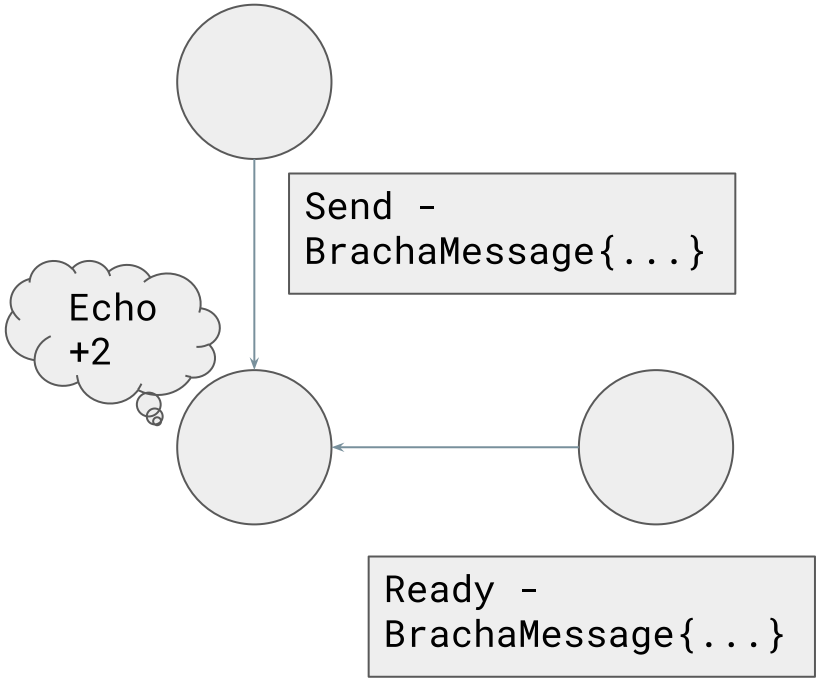
Desired path is removed from the message structure, and instead deduced from global routing table

Optimizations - BrachaKnown message

```
Src      uint64
Id       uint32
Payload  {}interface
```

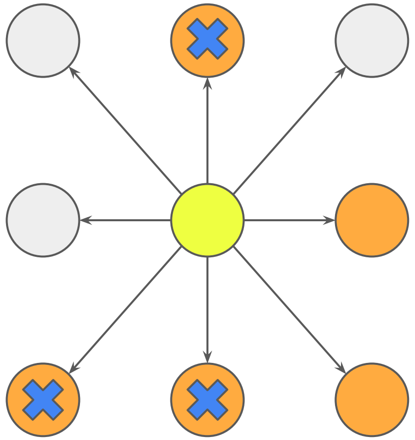
Identical to normal Bracha message

ORB.1 - Implicit echo



Send and `ready` messages will also double as an echo message

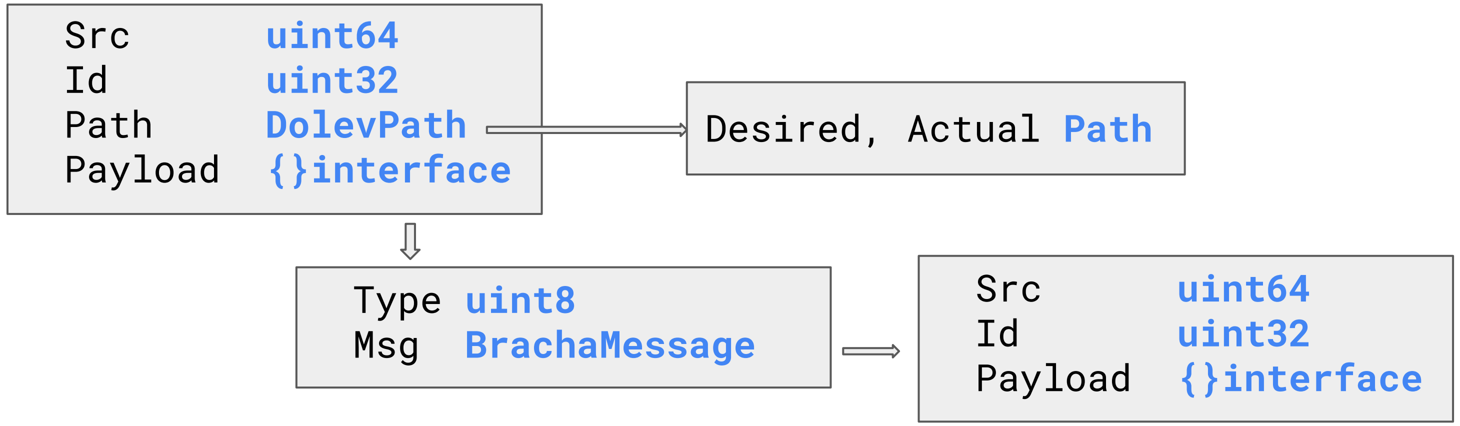
ORB.2 - Subset of neighbours



Pick subset of neighbours for `echo` and `ready` phase, based on *minimum sum of edges* (heuristic, optimal solution also exists)

Only `ready` messages are broadcasted to all

Optimizations - BrachaDolev message



```
Src      uint64
Id       uint32
Path     DolevPath
Payload  {}interface
```

```
Desired, Actual Path
```

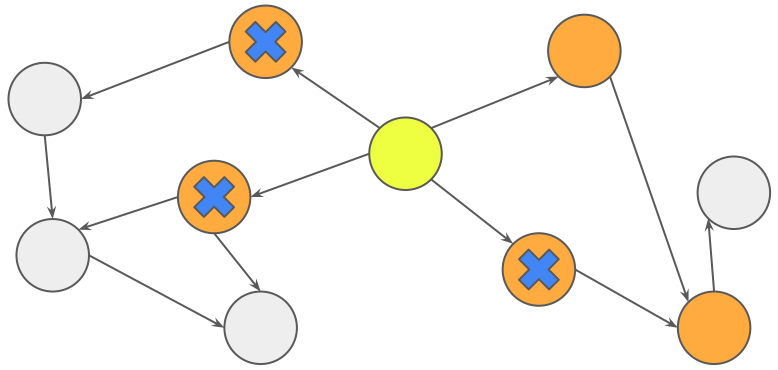
```
Type uint8
Msg  BrachaMessage
```

```
Src      uint64
Id       uint32
Payload  {}interface
```

Using DolevKnown and BrachaKnown, including a small wrapper to include message type (send, echo, ready)

Optimizations - Application of other optimizations

- Dolev optimizations (ORD.1-7) can be applied as is
- Bracha optimizations need fixing:
  - ORB.1 can be applied as is
  - ORB.2 selection changes from *minimum sum of weights* to *closest neighbours*

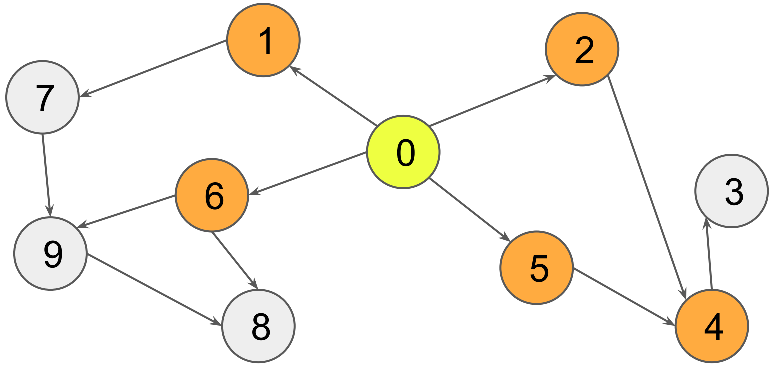


ORBD.1 - Partial broadcasts

```
Partial routing table:
1 = {...}
2 = {...}
4 = {...}
5 = {...}
6 = {...}
```

```
Normal routing table:
1 = {...}
2 = {...}
3 = {...}
4 = {...}
5 = {...}
6 = {...}
7 = {...}
8 = {...}
9 = {...}
```

Extension of ORB.2: Not all processes need to receive the `send/echo` message, omit these from routing table in those cases



ORBD.2 - Merge Bracha messages with same next hop

```
Src     = 0
Id      = 0
Payload = BrachaMessage{...}
Route   = [0,2,3,4]
```

```
Src     = 1
Id      = 0
Payload = BrachaMessage{...}
Route   = [1,5,3,4]
```

```
Src     = 3
Id      = 0
Payload = Wrapper{...}
Route   = [3,4]
```

Similar to ORD.5/6: Messages from the same original Bracha broadcast can be merged (`send/echo/ready`)

Message re-transmitted with wrapper as payload. Can be used to reconstruct original messages:

```
Msgs    []Wrapper
Payload {}interface
Src     uint64
Id      uint32
```

$\Longrightarrow$

```
Src   uint64
Id    uint32
Type  uint8
Paths []DolevPath
```

Additional notes

- Combining ORD.7 (implicit paths), ORBD.1 (partial broadcast), and ORBD.2 (merging bracha messages) can be tricky, as broadcast information such as the `partial` flag could get lost, which negates the effects of ORBD.1
- ORD.6 (similar payload merging) and ORBD.2 (merging bracha messages) are mutually exclusive, since they use the same buffer and one changes the payload. ORBD.2 > ORD.6 in the general case, ORD.6 >> ORBD.2 when multiple identical payload broadcasts.
- ORD.4 is not very effective. Likely cause is the heuristic, which also favors paths which use edges of multiple paths (should be avoided). In networks with maximum *f* almost all edges are used anyways.

This week

- Redo some evaluation
- Finalize paper
- Peer review group
- Ensure reproducibility