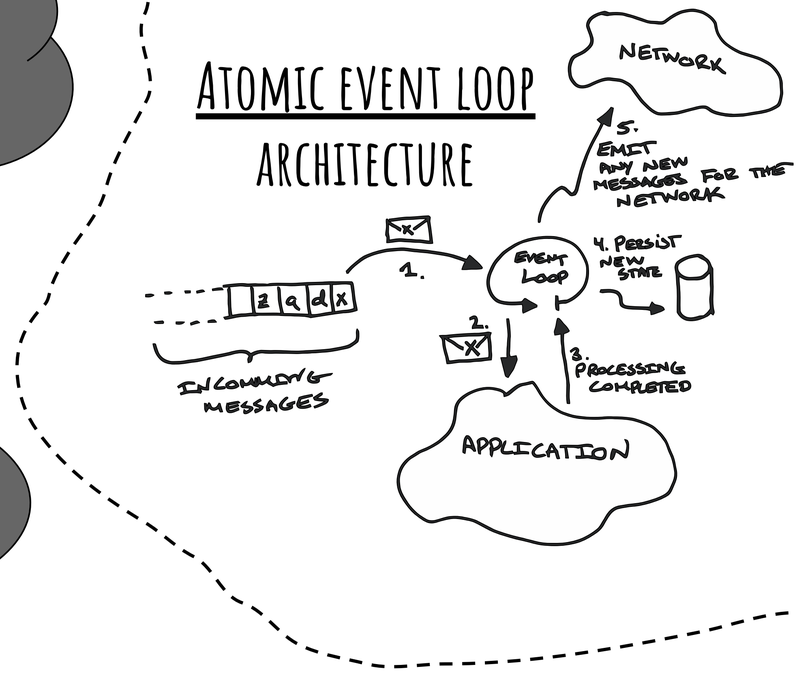
Corums.NET

Simplifying implementation of consensus algorithms

Future Work

- Benchmarks
- Efficient persistency
- Parallelizing event loop
- Reconfiguration

Atomic event loop architecture



What is Corums?

.NET Framework targeted at simplifying implementation of distributed algorithms & data structures in particular focusing on **consensus algorithms**

- the Observer pattern
- Higher-order Operators
- composability

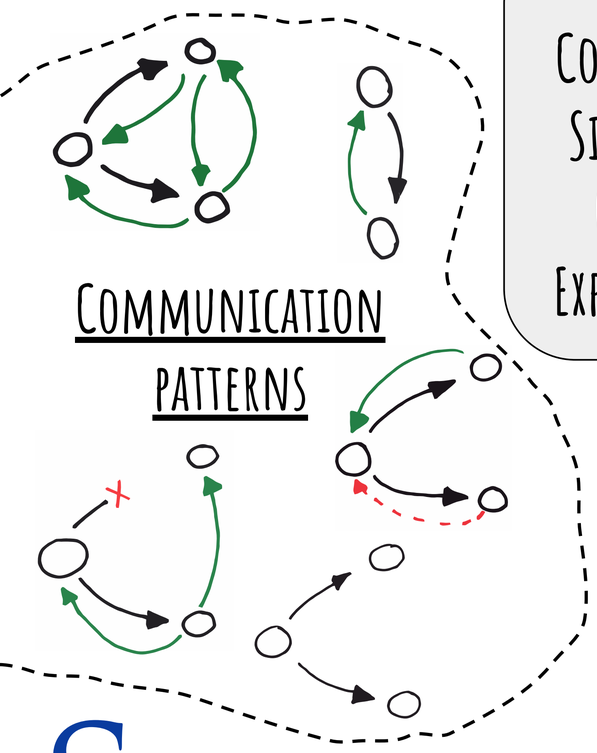
What is Reactive programming?

ReactiveX

The why

Concise & understandable code
Simple Single threaded model
Out-of-the-box persistency
Expressive & Useful abstractions

Communication patterns



Continue after a crash?

```
Task<bool?> CommitClientRequest(clientProposal, ballot)
  var promises = bus.Broadcast(Prepare(_ballot))
    .Take(_quorum)
    .ToList()

  if (promises.Any(r => r is Nack))
    return null

  var proposal = promises
    .Cast<Promise>()
    .MaxFor(p => p.ValueBallot).Value ?? clientProposal

  var learns = bus.Broadcast(new Accept(_ballot, proposal))
    .Take(_quorum)
    .ToList()

  if (learns.Any(r => r is Nack))
    return null

  return learns.First().Value == CLIENT_PROPOSAL
```

Single decree Paxos election and proposal workflow comparison

Corums	Event Handlers

`Task<bool?> CommitClientRequest(clientProposal, ballot)`
`  var promises = bus.Broadcast(Prepare(ballot))`
`  .Take(_quorum)`
`  .ToList()`
 	
`  if (promises.Any(r => r is Nack))`
`  return null`
 	
`  var proposal = promises`
`  .Cast<Promise>()`
`  .MaxFor(p => p.ValueBallot).Value ?? clientProposal`
 	
`  var learns = bus.Broadcast(Accept(ballot, proposal))`
`  .Take(_quorum)`
`  .ToList()`
 	
`  if (learns.Any(r => r is Nack))`
`  return null`
 	
`  return learns.First().Value == clientProposal`
	`Fields _ballot, _promises, _clientProposal,`
	`  _nackReceived, _callback, _learns`
	
	`Start(clientProposal, ballot, callback)`
	`  _ballot = ballot`
	`  _clientProposal = clientProposal`
	`  _promises.clear()`
	`  _learns.clear()`
	`  _nackReceived = false`
	`  _callback = callback`
	`  Broadcast Prepare`
	
	`Handle(Nack n)`
	`  If (n.Ballot < _ballot or _nackReceived) return`
	`  _nackReceived = true`
	`  callback(undecided)`
	
	`Handle(Promise p)`
	`  If (p.Ballot < _ballot or _nackReceived) return`
	`  _promises.add(p)`
	`  If (_promises.count < quorum) return`
	`  FindValue(_promises, _clientProposal)`
	`  Broadcast Accept`
	
	`Handle(Accepted a)`
	`  If (a.Ballot < _ballot or _nackReceived) return`
	`  _learns.add(a)`
	`  If (_learns.count < quorum) return`
	`  _callback(_clientProposal == _learns.First().value)`
	`  _learns.clear()`
	`  _nackReceived = true`

University of Stavanger

By: Thomas Stidsborg Sylvest - Supervisors: Hein Meling, Leander Jehl & Vero Estrada-Galinanes