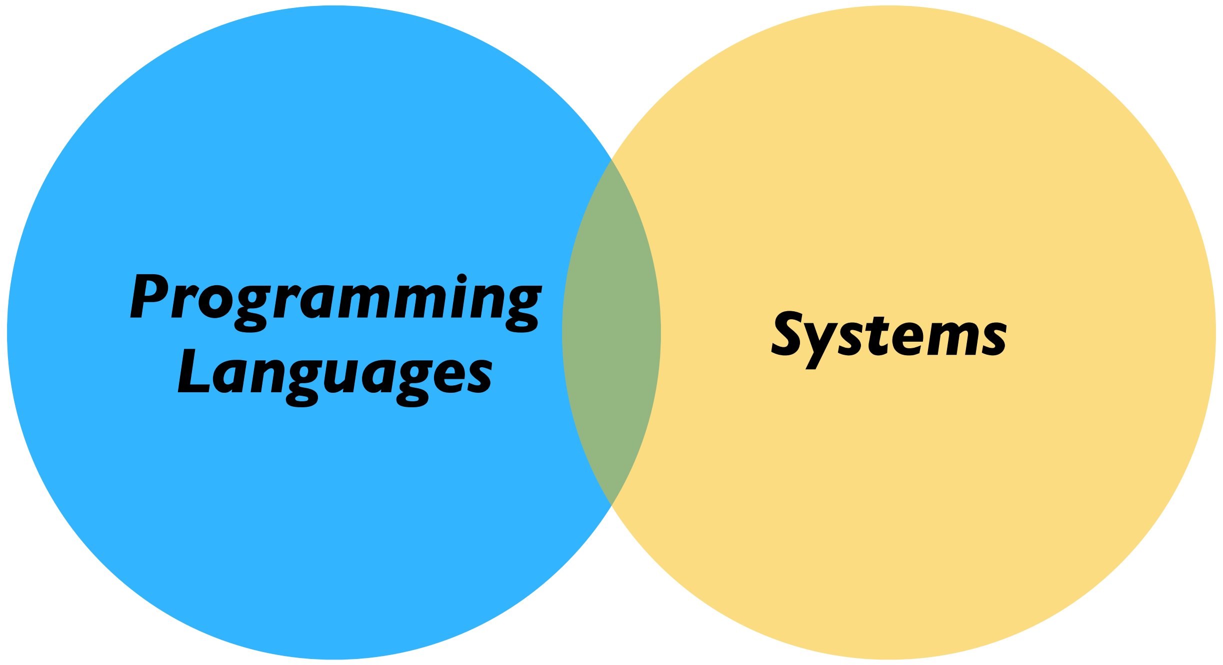
Multicore Support for Tezos Blockchain

KC Sivaramakrishnan
Computer Science and Engineering

My research

- PL has *central place* in solving computing problems

My research



Concurrent

Parallel

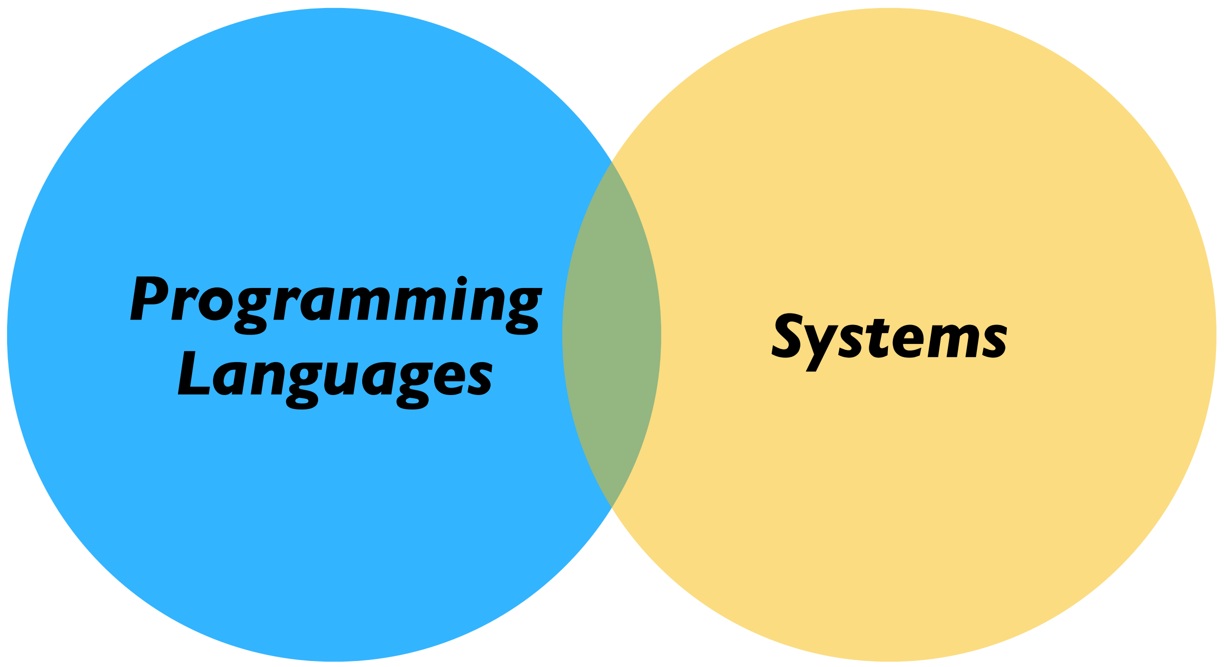
Distributed

Operating

...

- PL has *central place* in solving computing problems

My research



Concurrent

Parallel

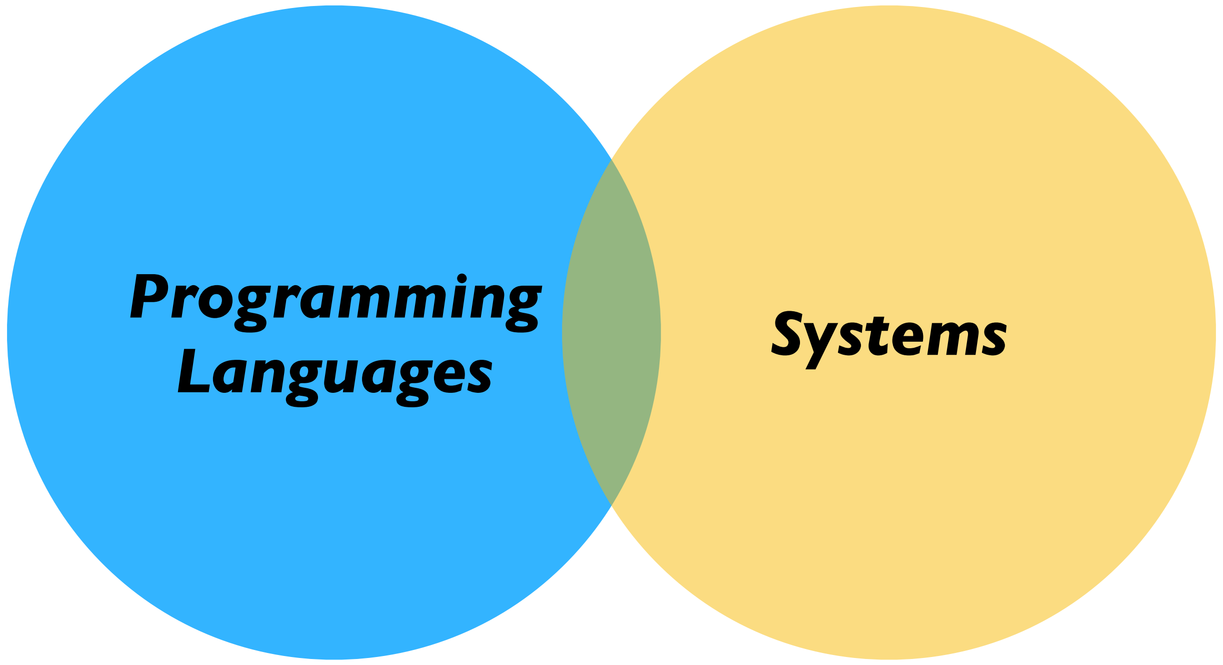
Distributed

Operating

…

- PL has *central place* in solving computing problems
- PL as a tool to *formally* reason about complex systems
    - ✦ Develop *abstractions* for simplifying systems

My research



Concurrent

Parallel

Distributed

Operating

...

- PL has *central place* in solving computing problems
- PL as a tool to *formally* reason about complex systems
  - ✦ Develop *abstractions* for simplifying systems
- **Interests:** programming language runtimes, distributed databases, concurrency, secure systems engineering

Tezos Blockchain

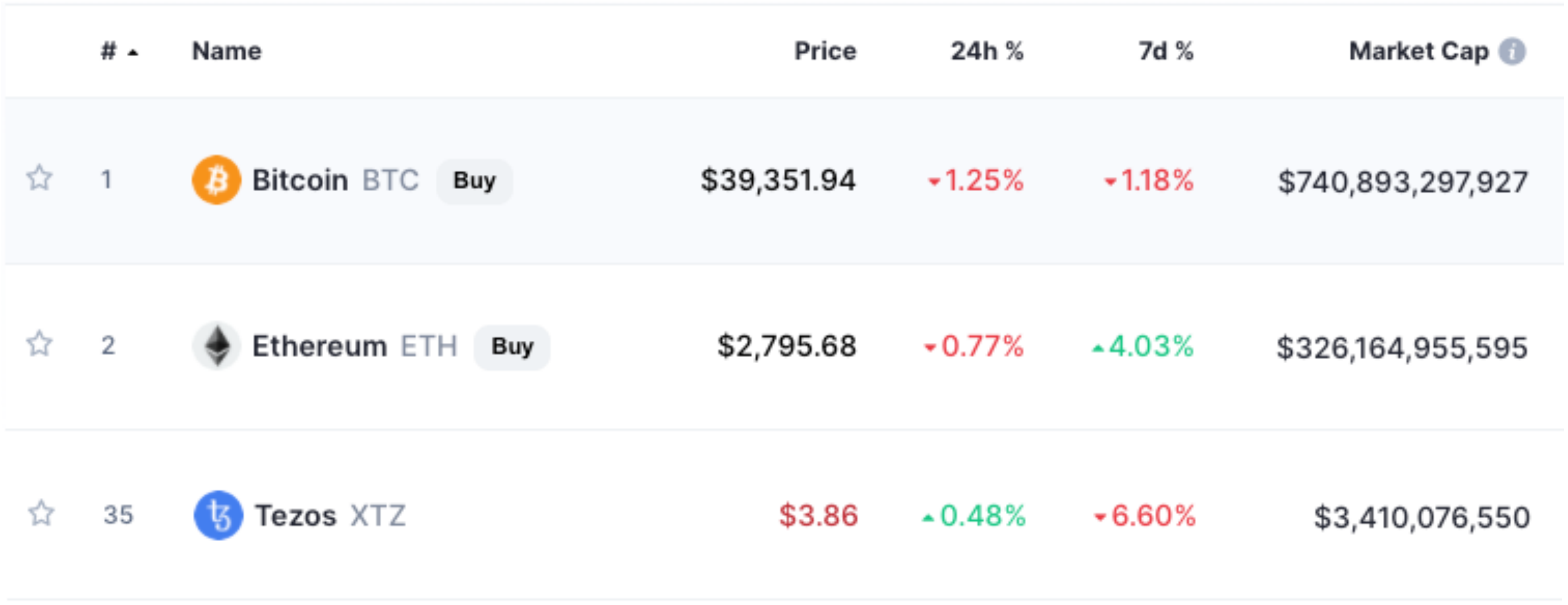
- Public, Permission-less, Proof-of-Stake blockchain capable of running smart contracts

Tezos Blockchain

- Public, Permission-less, Proof-of-Stake blockchain capable of running smart contracts
- Tezos Foundation (HQ: Zug, Switzerland) promotes Tezos

Tezos Blockchain

- Public, Permission-less, Proof-of-Stake blockchain capable of running smart contracts
- Tezos Foundation (HQ: Zug, Switzerland) promotes Tezos
- Had the biggest ICO $232 million of its time



#	Name	Price	24h %	7d %	Market Cap
1	Bitcoin BTC Buy	$39,351.94	▾1.25%	▾1.18%	$740,893,297,927
2	Ethereum ETH Buy	$2,795.68	▾0.77%	▴4.03%	$326,164,955,595
35	Tezos XTZ	$3.86	▴0.48%	▾6.60%	$3,410,076,550

What sets Tezos apart

What sets Tezos apart

- On-chain governance
    - Participants vote to bring in updates on the chain
    - Avoids *Hard Fork* problems — Bitcoin, Bitcoin Lite, Bitcoin Cash, Bitcoin SV…

What sets Tezos apart

- On-chain governance
  - Participants vote to bring in updates on the chain
  - Avoids *Hard Fork* problems — Bitcoin, Bitcoin Lite, Bitcoin Cash, Bitcoin SV…
- (Liquid) Proof-of-stake instead of Proof-of-work
  - Proof-of-work is *energy intensive* — Bitcoin *129 TWH* ~= Norway

What sets Tezos apart

- On-chain governance
  - ✦ Participants vote to bring in updates on the chain
  - ✦ Avoids *Hard Fork* problems — Bitcoin, Bitcoin Lite, Bitcoin Cash, Bitcoin SV…
- (Liquid) Proof-of-stake instead of Proof-of-work
  - ✦ Proof-of-work is *energy intensive* — Bitcoin *129 TWH* ~= Norway
- Tezos is amenable for *formal verification*
  - ✦ *Michelson*, low-level smart contract language is expressed as a OCaml GADT
    - ✤ Rules out large classes of errors by construction
  - ✦ Many efforts around *full-functional verification* of Tezos smart contracts — Mi-Cho-Coq, Albert

Performance

	₿	Ξ	PayPal	VISA	ꜩ
Transactions per second:	7	30	200	3000	40
Confirmation Latency:	1 hour	10 minutes	Few seconds	Few seconds	30 minutes

Performance

			PayPal	VISA	
Transactions per second:	7	30	200	3000	40
Confirmation Latency:	1 hour	10 minutes	Few seconds	Few seconds	30 minutes

- **Tezos Goal**
  - ✦ Increase throughput 100x to 1000x
  - ✦ Latency of ~1 min

Performance

	₿	Ξ	PayPal	VISA	ꜩ
Transactions per second:	7	30	200	3000	40
Confirmation Latency:	1 hour	10 minutes	Few seconds	Few seconds	30 minutes

- **Tezos Goal**
  - ✦ Increase throughput 100x to 1000x
  - ✦ Latency of ~1 min
- **Strategy:** *Exploit multicore parallelism*

Tezos: Implementation Language

Recently turned 25!

OCaml

Tezos: Implementation Language

Recently turned 25!

OCaml

Industry

ahrefs

Jane Street

docker

FACEBOOK

Tarides

Bloomberg

Projects

The Astrée Static Analyzer

flow

hack

COMPCERT

OCaml: Langauge of choice

Recently turned 25!

OCaml

Industry

ahrefs

Jane Street

docker

FACEBOOK

Tarides

Bloomberg

Projects

The Astrée Static Analyzer

flow

hack

COMPCERT

No multicore support!

Multicore OCaml

Multicore OCaml

- Adds native support for *concurrency* and *shared-memory parallelism* to OCaml

Multicore OCaml

- Adds native support for *concurrency* and *shared-memory parallelism* to OCaml

IIT MADRAS

UNIVERSITY OF CAMBRIDGE

OCaml Labs

Jane Street

Opsian

λ 0x1F42B Segfault Systems

Multicore OCaml

- Adds native support for *concurrency* and *shared-memory parallelism* to OCaml

IIT MADRAS   UNIVERSITY OF CAMBRIDGE   OCaml Labs   Jane Street   Opsian   λ 0x1F42B Segfault Systems

- Research
  - ✦ Concurrent and parallel garbage collector for OCaml [ICFP '20]
  - ✦ Novel concurrency substrate [PLDI '21]
  - ✦ Modular memory model [PLDI '18]

Multicore OCaml

- Adds native support for *concurrency* and *shared-memory parallelism* to OCaml

IIT MADRAS UNIVERSITY OF CAMBRIDGE OCaml Labs Jane Street Opsian λ 0x1F42B Segfault Systems

- Research
  - ✦ Concurrent and parallel garbage collector for OCaml [ICFP '20]
  - ✦ Novel concurrency substrate [PLDI '21]
  - ✦ Modular memory model [PLDI '18]
- Challenge
  - ✦ Not just a prototype, but *millions of lines of legacy code*
  - ✦ Fast and predictable performance

Multicore OCaml

- Adds native support for *concurrency* and *shared-memory parallelism* to OCaml

IIT MADRAS UNIVERSITY OF CAMBRIDGE OCaml Labs Jane Street Opsian λ 0x1F42B Segfault Systems

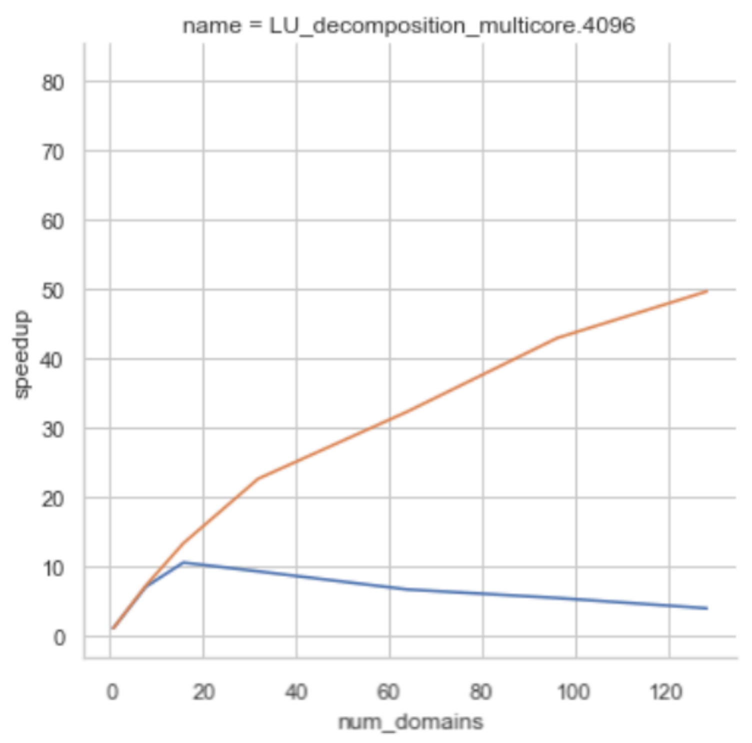
- Research
  - ✦ Concurrent and parallel garbage collector for OCaml [ICFP '20]
  - ✦ Novel concurrency substrate [PLDI '21]
  - ✦ Modular memory model [PLDI '18]

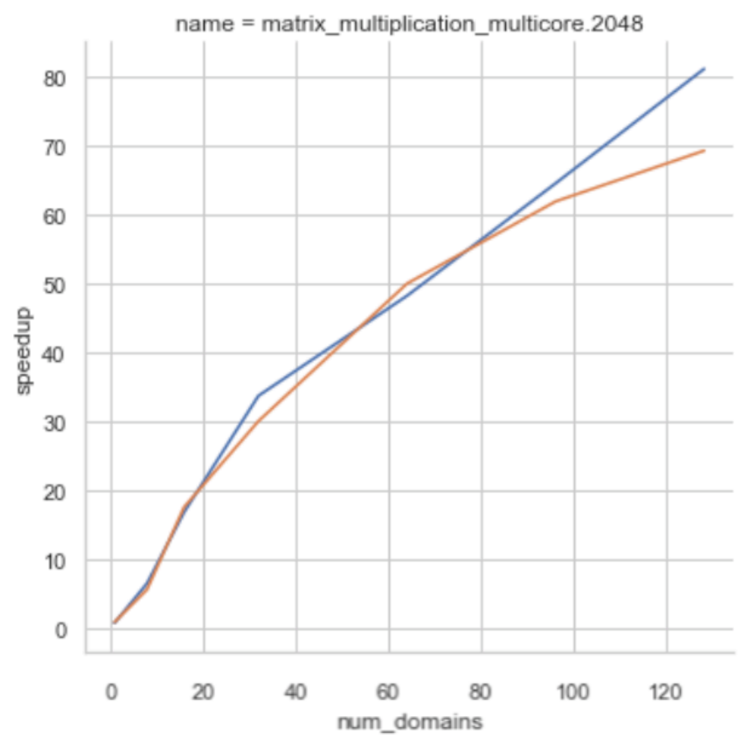
funded by Tezos Foundation!

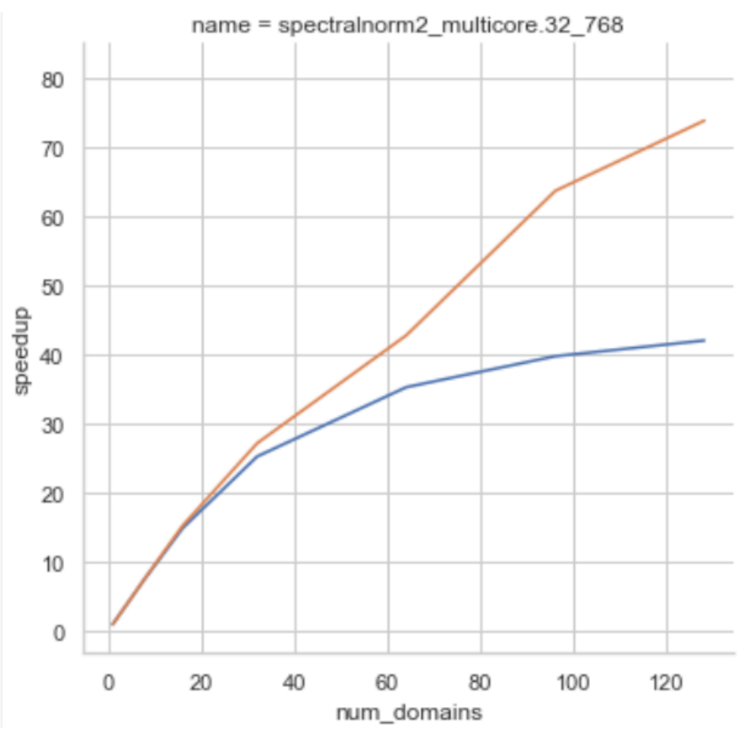
- Challenge
  - ✦ Not just a prototype, but *millions of lines of legacy code*
  - ✦ Fast and predictable performance

Parallel Scalability

Hot off the presses!

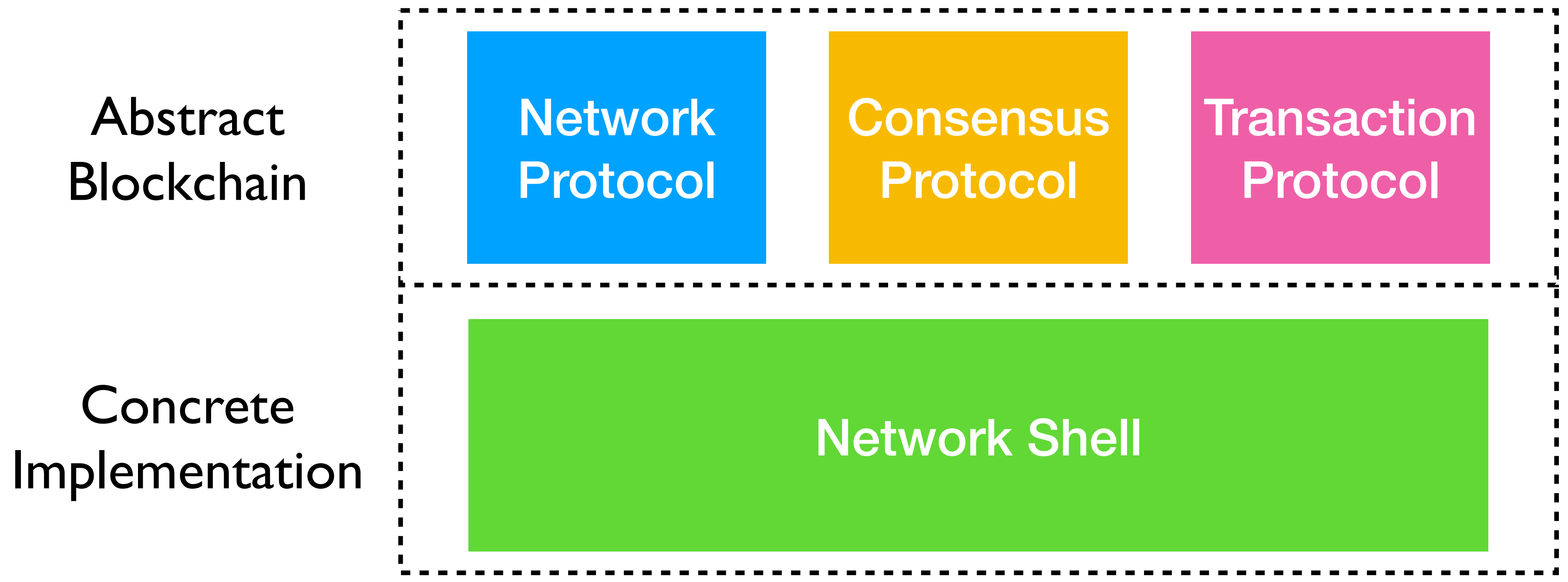




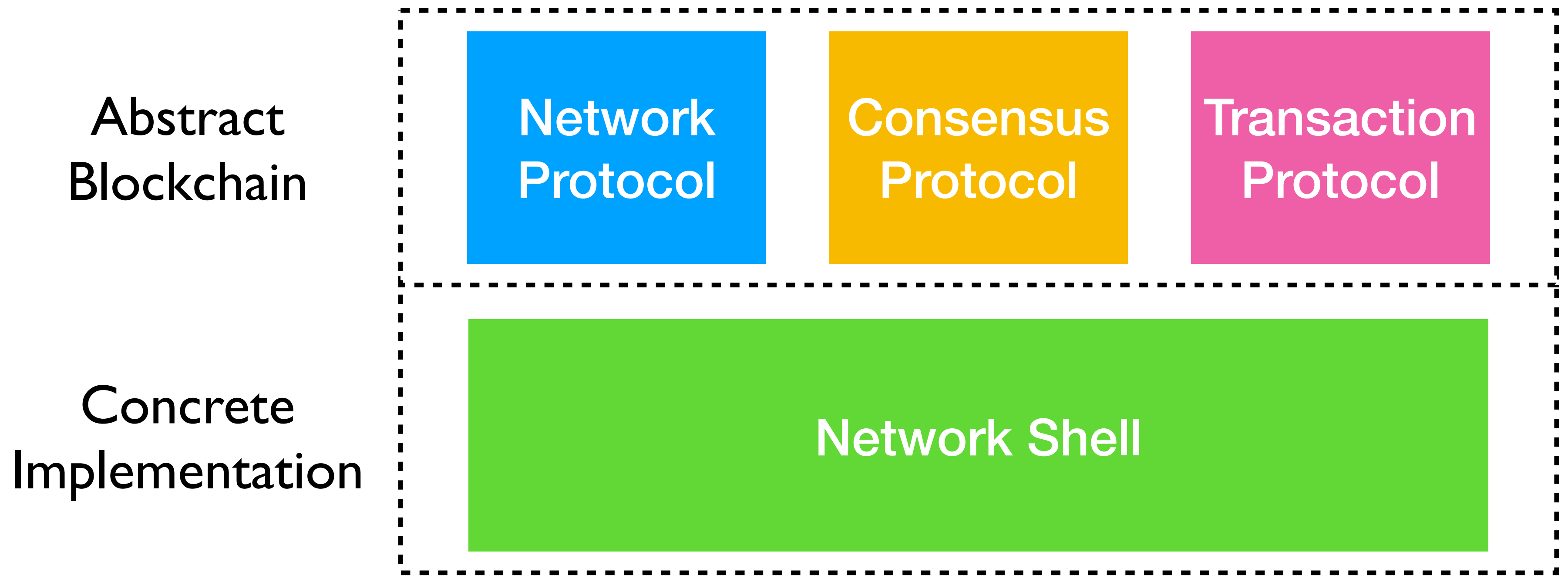


50x — 80x speedup on 128-core machine

Tezos Protocol

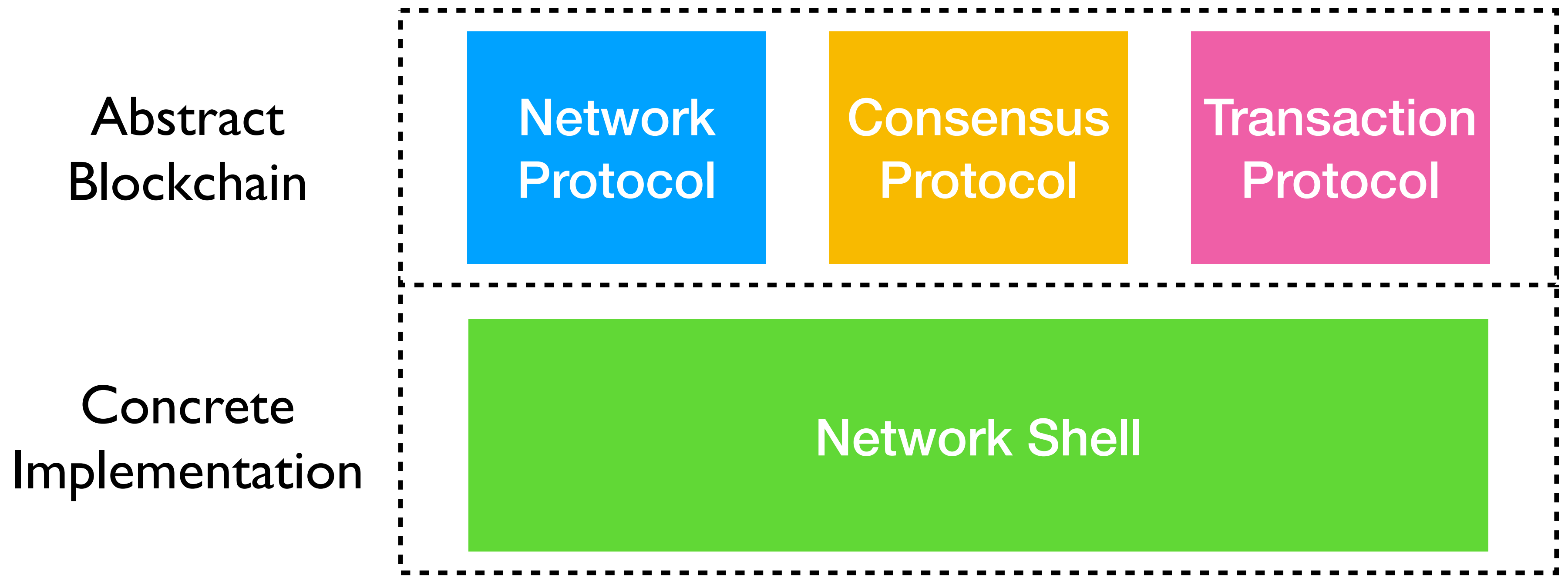


Tezos Protocol



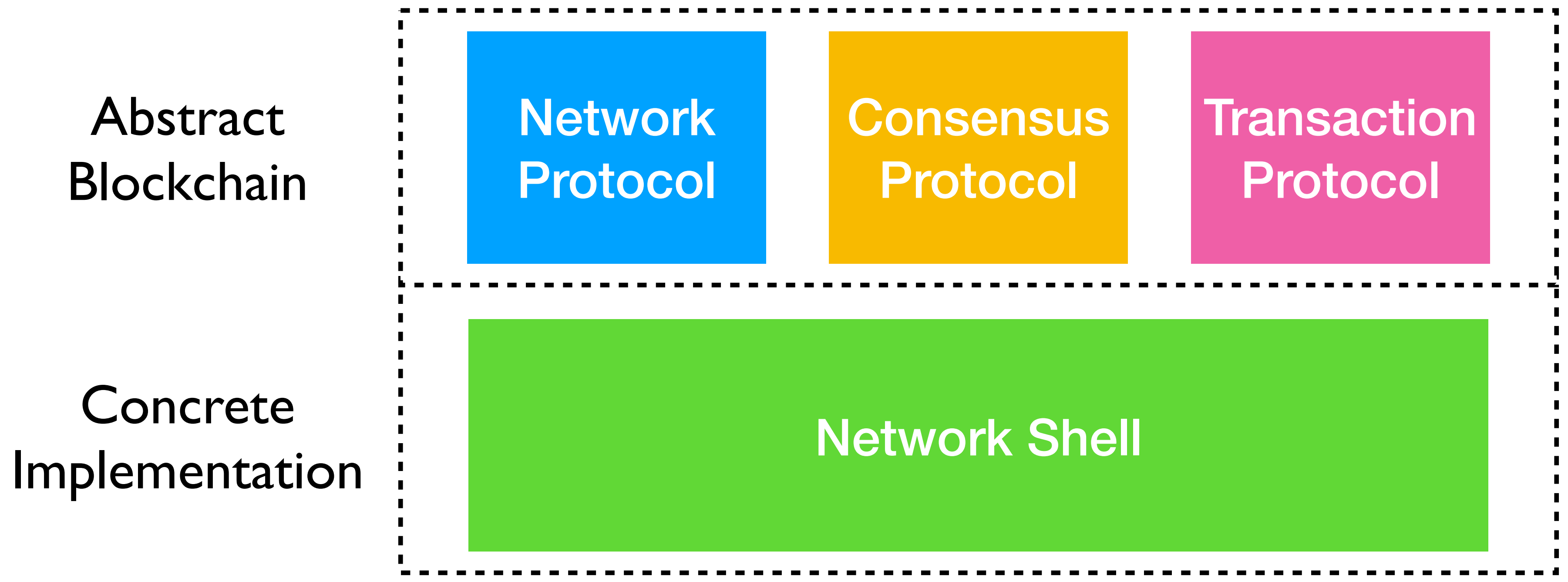
- Network protocol — Peer discovery & publishing blocks

Tezos Protocol



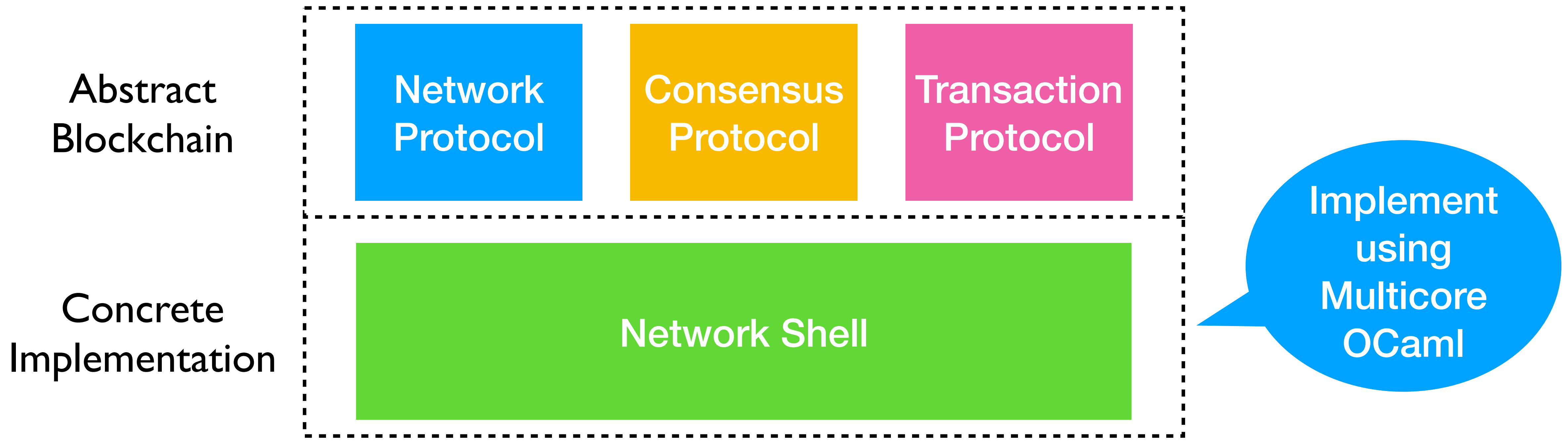
- Network protocol — Peer discovery & publishing blocks
- Consensus protocol — Block acceptance, miner reward schedules

Tezos Protocol



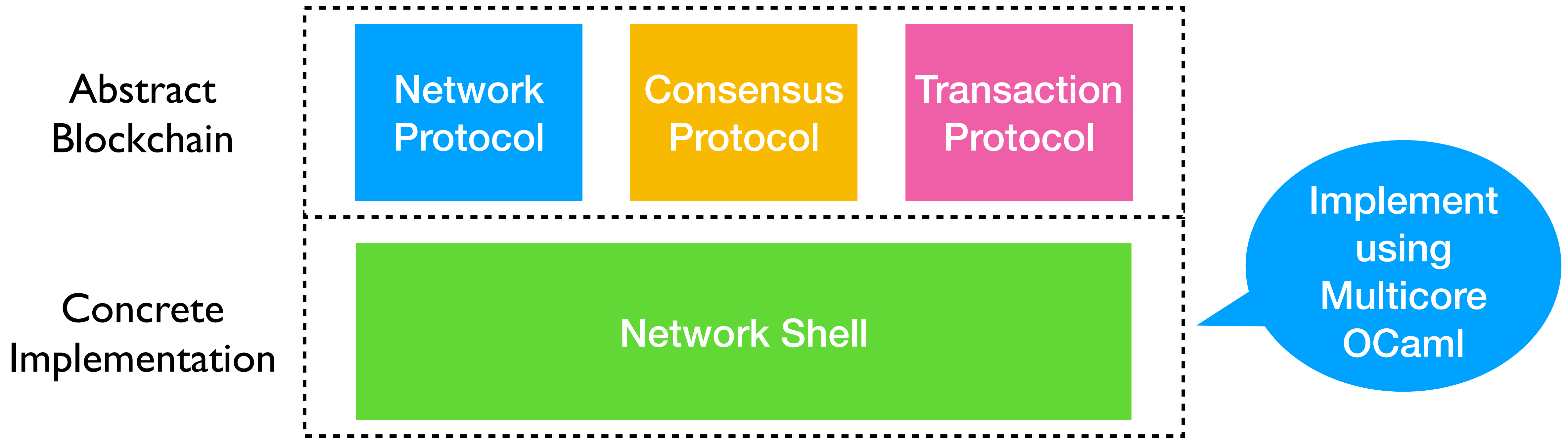
- Network protocol — Peer discovery & publishing blocks
- Consensus protocol — Block acceptance, miner reward schedules
- Transaction protocol — Validity of transaction, blocks

Tezos + Multicore OCaml



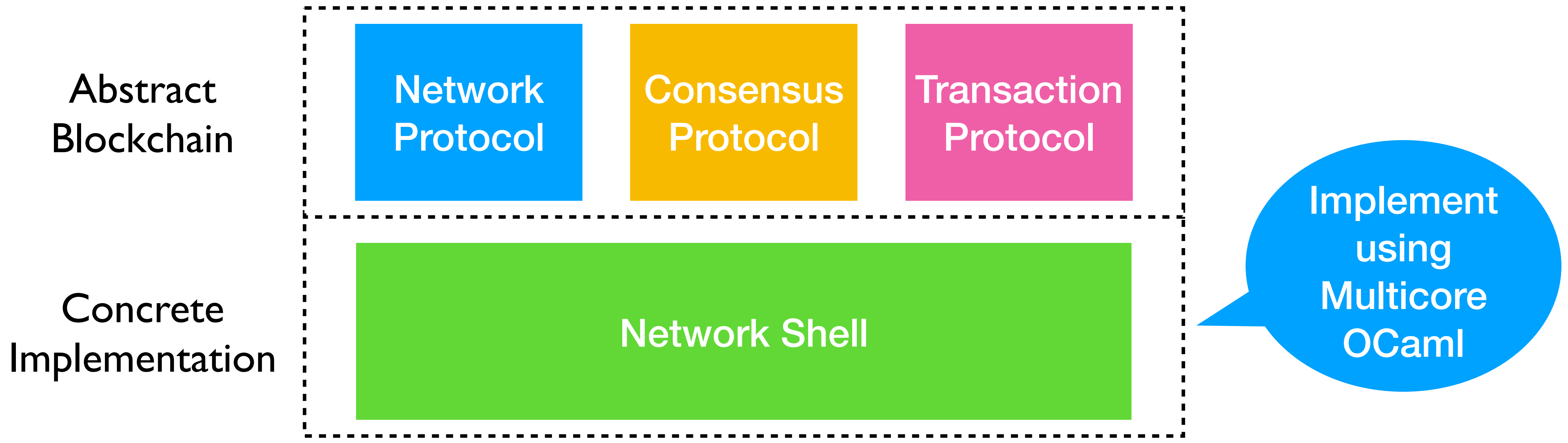
- Offload compute intensive tasks of transaction protocol (block validation, serialisation) to spare cores

Tezos + Multicore OCaml



- Offload compute intensive tasks of transaction protocol (block validation, serialisation) to spare cores
- Block reconciliation in mempool reminiscent of GC
  - ✦ Implement parallel GC for block reconciliation

Tezos + Multicore OCaml



- Offload compute intensive tasks of transaction protocol (block validation, serialisation) to spare cores
- Block reconciliation in mempool reminiscent of GC
  - ✦ Implement parallel GC for block reconciliation
- Exploit *deterministic parallelism* in inter-contract calls

Inter-contract call semantics

```
C1

storage = …

fun a () =
  call c2.b()
  call c3.c()
```

```
C2

storage = …

fun b () =
  call c4.d()
```

```
C3

storage = …

fun c () =
  call c4.d()
```

```
C4

storage = …

fun d () =
  //modify local storage
  return
```

Inter-contract call semantics

```
C1

storage = …

fun a () =
  call c2.b()
  call c3.c()
```

```
C2

storage = …

fun b () =
  call c4.d()
```

```
C3

storage = …

fun c () =
  call c4.d()
```

```
C4

storage = …

fun d () =
  //modify local storage
  return
```

Sequential Execution

Finished a : [b, c]

Finished b : [d1, c]

Finished d1: [c]

Finished c : [d2]

Finished d2 : [ ]

Inter-contract call semantics

```
C1

storage = …

fun a () =
  call c2.b()
  call c3.c()
```

```
C2

storage = …

fun b () =
  call c4.d()
```

```
C3

storage = …

fun c () =
  call c4.d()
```

```
C4

storage = …

fun d () =
  //modify local storage
  return
```

Sequential Execution

Finished a : [b, c]

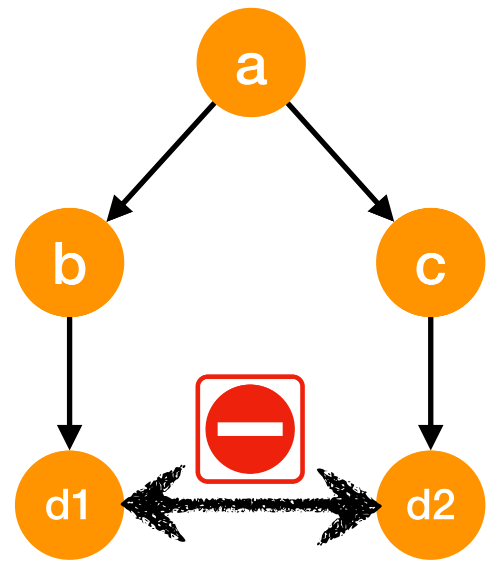
Finished b : [d1, c]

Finished d1: [c]

Finished c : [d2]

Finished d2 : [ ]

Parallel Execution



Inter-contract call semantics

```
C1

storage = …

fun a () =
  call c2.b()
  call c3.c()
```

```
C2

storage = …

fun b () =
  call c4.d()
```

```
C3

storage = …

fun c () =
  call c4.d()
```

```
C4

storage = …

fun d () =
  //modify local storage
  return
```

Sequential Execution

Finished a : [b, c]

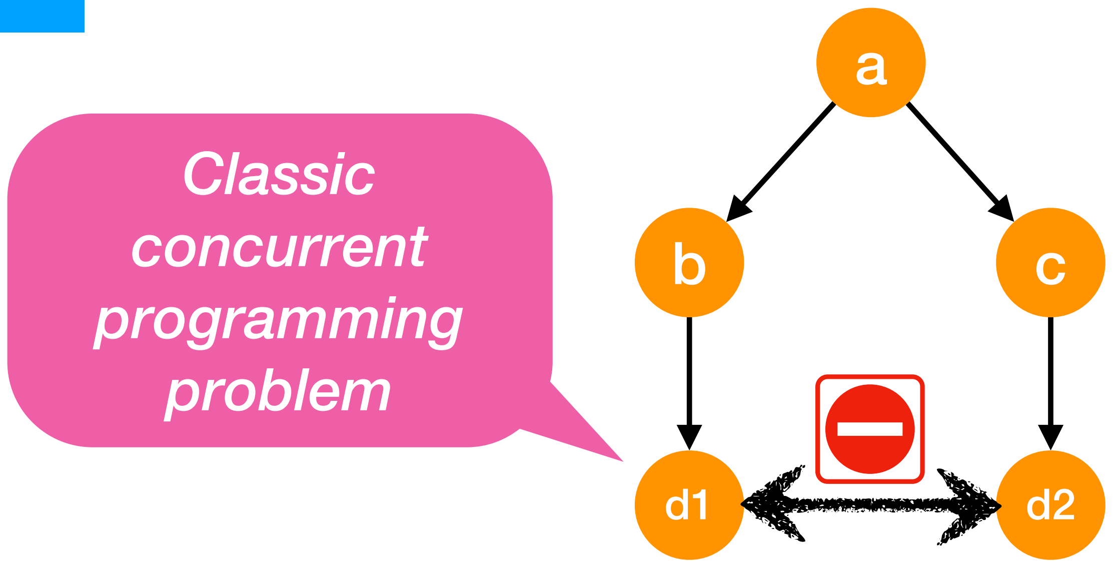
Finished b : [d1, c]

Finished d1: [c]

Finished c : [d2]

Finished d2 : [ ]

Parallel Execution



Thanks!

github.com/ocaml-multicore