

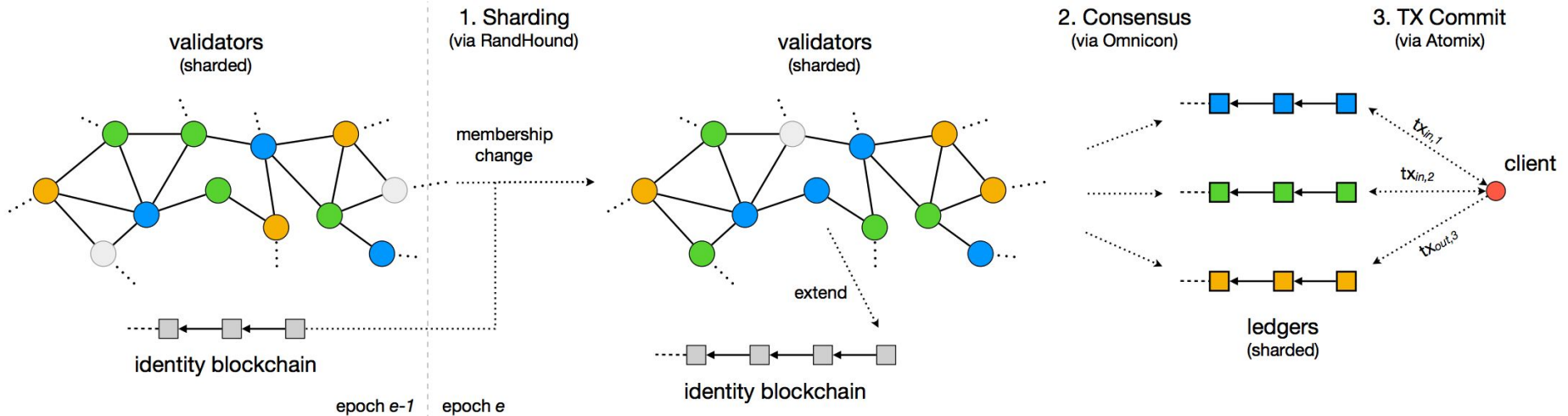


Open consensus for 10B People

Scaling Smart Contracts via Systems and Language Design

Stephen Tse
harmony.one/lang

Harmony: Production Network of 100k Nodes



A high-performance blockchain demands **10x innovations** in transport networks, consensus protocols & systems tools.

Backtesting? Adversarial simulation, parameter optimizations?

We bring **proven innovations** to *large-scale* production [OmniLedger*, RapidChain, Chainspace, Avalanche].

Scaling Trust to 10B people & 100B devices

Can we *agree* now but foresee **inconsistency** in practice?

Can we *rely* on **intuition** when expressing terms & conditions?

Can we *iterate* without lawyers, regional laws, or **penalties**?

Can smart contracts be **safe, easy, fast**?

Trusting Contracts: As **SAFE** as Java

Java Virtual Machine makes web applications so *boringly* reliable for teams of 100+ developers

Trusting Contracts: As **SAFE** as Java

Run-time checks maintain **global constraints** (termination, resource consumption, balance flow)

Trusting Contracts: As **SAFE** as Java

Formal verification via **dependent types** (Twelf, ProVerif, Coq)
guarantees *hacker-proof* before deploy

Trusting Contracts: As **SAFE** as Java

Java Virtual Machine makes web applications so *boringly* reliable for teams of 100+ developers

Run-time checks maintain **global constraints** (termination, resource consumption, balance flow)

Formal verification via **dependent types** (Twelf, ProVerif, Coq) guarantees *hacker-proof* before deploy

Tse/TOPLAS: Verified interoperable implementations of security protocols

Tse/IEEE-SP: Run-time principals in information-flow type systems

Writing Contracts: As **EASY** as Python

Minimal syntax (no distraction!), human has limited cognitive focus for abstractions such as *invariants* & *isomorphism*

Writing Contracts: As **EASY** as Python

Functional and *process calculi* to avoid managing states, **type inference** as theorem proving to tame structures & complexity

Writing Contracts: As **EASY** as Python

Understand theories behind dependencies (information flow) & **parametricity** (higher-order polymorphism) vs “fat languages”

Writing Contracts: As **EASY** as Python

Minimal syntax (no distraction!), human has limited cognitive focus for abstractions such as *invariants & isomorphism*

Functional and *process calculi* to avoid managing states, **type inference** as theorem proving to tame structures & complexity

Understand theories behind dependencies (information flow) & **parametricity** (higher-order polymorphism) vs “fat languages”

Tse/Penn: Concise concrete syntax (generalized LR parser), min-lang.com

Tse/ICFP: Translating dependency into parametricity

Running Contracts: As **FAST** as OCaml

Compile to native code & run tests in 200 ms

Tezos, Zilliqa's Scilla, Coda's SNARK, **Coq** are written in OCaml

Declarative style leaves freedom for memory management, parallelism strategies, graph executions, [sharding algorithms](#)

Beautiful and Secure Code

```
black_scholes
```

```
  s : ℝ # stock price
```

```
  x : ℝ # strike price
```

```
  t : ℝ # expiration time in years
```

```
  r : ℝ # risk-free interest rate
```

```
  σ : ℝ # volatility
```

```
  : ℝ
```

```
= s φ(d1) - x e(-r t)φ(d2) @
```

```
φ = Normal.cdf
```

```
d0 = log s/x + (r + σ2/2)t
```

```
d1 = d0 / σ√t
```

```
d2 = d1 - σ√t
```

harmony.one/type-checks

Join **Harmony** team! Bring Empathy, Passion, Excellence



Stephen: security protocols PhD

Nicolas: VR startup founder

Alok: **Apple Siri** ML

Rongjian: Google search

Minh: **Google infrastructure**

Nick: Stanford AI masters

Sahil: Harvard MBA

Eugene: Amazon networking

Leo: **Amazon Phone OS**

Hakwan: Rhode Scholars

Kayuet: Oxford PhD

Rust engineers, protocol researchers,
compiler writers to **s@harmony.one**

See harmony.one/law, /sharding and /tgi