Impact of Network Topologies on Blockchains Performance

NiRvAna Annual Workshop
May 29-31, 2025, Urbino, Italy

Vincenzo Di Perna
(PhD Student - UniCam/Urb)
vincenzo.diperna@unicam.it

Valerio Schiavoni
(Senior lecturer - UniNe)
valerio.schiavoni@unine.ch

Miguel Matos
(Senior lecturer - IST Lisbon)
miguel.marques.matos@tecnico.ulisboa.pt

Sebastiao Amaro
(PhD Student - INESC-ID)
sebastiao.amaro@tecnico.ulisboa.pt

Francesco Fabris
(Professor - UniTs)
ffabris@units.it

Marco Bernardo
(Professor - UniUrb)
marco.bernardo@uniurb.it

Featured by NiRvAna project

CONTEXT #1

CONTEXT #1

Blockchain scenario

CONTEXT #1

Blockchain scenario

- Blockchain operates as a distributed ledger shared among nodes that do not inherently trust each other.

CONTEXT #1

Blockchain scenario

- Blockchain operates as a distributed ledger shared among nodes that do not inherently trust each other.

- The technology has seen broad adoption across various platforms, including **Algorand, Ethereum, Solana, and Quorum**.

CONTEXT #1

Blockchain scenario

- Blockchain operates as a distributed ledger shared among nodes that do not inherently trust each other.

- The technology has seen broad adoption across various platforms, including **Algorand, Ethereum, Solana, and Quorum**.

- Flexibility and adaptability of blockchain solutions to diverse industries **(e.g., Finance, Supply Chain, Healtcare)**.

CONTEXT #1

Blockchain scenario

- Blockchain operates as a distributed ledger shared among nodes that do not inherently trust each other.

- The technology has seen broad adoption across various platforms, including **Algorand, Ethereum, Solana, and Quorum**.

- Flexibility and adaptability of blockchain solutions to diverse industries **(e.g., Finance, Supply Chain, Healtcare)**.

- Assessing and comparing the performance of different blockchain systems.

CONTEXT #1

Blockchain scenario

- Blockchain operates as a distributed ledger shared among nodes that do not inherently trust each other.

- The technology has seen broad adoption across various platforms, including **Algorand, Ethereum, Solana, and Quorum**.

- Flexibility and adaptability of blockchain solutions to diverse industries **(e.g., Finance, Supply Chain, Healtcare)**.

- Assessing and comparing the performance of different blockchain systems.

- Presence of several tool and suite for evaluating blockchain performance.

CONTEXT #2

CONTEXT #2

Blockchain benchmark tools comparison

CONTEXT #2

Blockchain benchmark tools comparison

Table 1: Comparison of several blockchain benchmark tools (◌ = not found; ○ = no match; ● = match; ◑ = partial match; Centr. = Centralized; Distr. = Distributed; Simul. = Simulation; Emul. = Emulation; SC = Smart Contract; TT = Transfer Transactions).

Tool	Year	Network settings	Orchestration	Mode	Reproducibility	Versatility	Observability	Portability	Ease to run	Workload
Shadow-Bitcoin [54]	2015	◌	Centr.	Simul.	○	○	○	○– Bitcoin	○	TT
HIVE [14]	2016	◌	Distr.	Simul.	○	○	○	○– Ethereum	○	◌
Simcoin [3]	2016	◌	Centr.	Simul.	○	○	●	○– Bitcoin	○	TT
Bitcoin-Simulator [39]	2016	◑	Centr.	Simul.	○	○	○	●– Bitcoin-like	○	TT
BlockBench [30]	2017	◌	◌	Simul.	○	●	○	●– Private	○	SC
Caliper [45]	2018	◑	Distr.	Emul.	○	○	●	●– Ethereum,Hyperledger	○	SC
Minichain [75]	2019	◑	Centr.	Emul.	○	○	○	●	○	◌
BlockLite [73]	2019	◌	Centr.	Emul.	○	○	○	●– PoW-public	○	◌
BlockSim-f [34]	2019	◑	Centr.	Simul.	○	○	○	●	○	◌
SimBlock [20]	2019	◑	Centr.	Simul.	○	○	○	●	○	TT
BlockSim-m [16]	2020	◑	Centr.	Simul.	○	○	○	●	○	TT
Core-Bit-Netw-Simul. [17]	2020	◑	Centr.	Simul.	○	○	○	●	○	◌
BBB [60]	2020	◑	Centr.	Emul.	○	○	○	●– Private	○	◌
SIMBA [35]	2020	◑	Centr.	Simul.	○	○	○	●	○	◌
BCTMark [65]	2020	◑	Distr.	Emul.	○	○	●	●	○	TT
BlockPerf [62]	2021	○	Distr.	Both	○	○	○	○– Bitcoin	○	TT
DLPS [36]	2021	◑	Distr.	Emul.	○	○	●	●	○	TT
Gromit [56]	2022	◑	Distr.	Emul.	○	○	●	●	○	SC
Diablo v2 [42]	2022	○	Distr.	Emul.	●	●	●	●	●	TT,SC
JABS [77]	2023	◑	Centr.	Simul.	○	○	○	●	○	TT
COCONUT [40]	2023	◌	Distr.	Emul.	●	●	○	●	●	TT,SC
GFBE [50]	2024	○	Distr.	Emul.	●	●	●	●	●	SC
STABL [43]	2024	◑	Distr.	Emul.	○	●	●	●	●	TT,SC
Lilith	2024	●	Distr.	Emul.	●	●	●	●	●	TT,SC

CONTEXT #2

Blockchain benchmark tools comparison

Table 1: Comparison of several blockchain benchmark tools (◌ = not found; ○ = no match; ● = match; ◑ = partial match; Centr. = Centralized; Distr. = Distributed; Simul. = Simulation; Emul. = Emulation; SC = Smart Contract; TT = Transfer Transactions).

Tool	Year	Network settings	Orchestration	Mode	Reproducibility	Versatility	Observability	Portability	Ease to run	Workload
Shadow-Bitcoin [54]	2015	◌	Centr.	Simul.	○	○	○	○– Bitcoin	○	TT
HIVE [14]	2016	◌	Distr.	Simul.	○	○	○	○– Ethereum	○	◌
Simcoin [3]	2016	◌	Centr.	Simul.	○	○	●	○– Bitcoin	○	TT
Bitcoin-Simulator [39]	2016	◑	Centr.	Simul.	○	○	○	●– Bitcoin-like	○	TT
BlockBench [30]	2017	◌	◌	Simul.	○	●	○	●– Private	○	SC
Caliper [45]	2018	◑	Distr.	Emul.	○	○	●	●– Ethereum,Hyperledger	○	SC
Minichain [75]	2019	◑	Centr.	Emul.	○	○	○	●	○	◌
BlockLite [73]	2019	◌	Centr.	Emul.	○	○	○	●– PoW-public	○	◌
BlockSim-f [34]	2019	◑	Centr.	Simul.	○	○	○	●	○	◌
SimBlock [20]	2019	◑	Centr.	Simul.	○	○	○	●	○	TT
BlockSim-m [16]	2020	◑	Centr.	Simul.	○	○	○	●	○	TT
Core-Bit-Netw-Simul. [17]	2020	◑	Centr.	Simul.	○	○	○	●	○	◌
BBB [60]	2020	◑	Centr.	Emul.	○	○	○	●– Private	○	◌
SIMBA [35]	2020	◑	Centr.	Simul.	○	○	○	●	○	◌
BCTMark [65]	2020	◑	Distr.	Emul.	○	○	●	●	○	TT
BlockPerf [62]	2021	○	Distr.	Both	○	○	○	○– Bitcoin	○	TT
DLPS [36]	2021	◑	Distr.	Emul.	○	○	●	●	○	TT
Gromit [56]	2022	◑	Distr.	Emul.	○	○	●	●	○	SC
Diablo v2 [42]	2022	○	Distr.	Emul.	●	●	●	●	●	TT,SC
JABS [77]	2023	◑	Centr.	Simul.	○	○	○	●	○	TT
COCONUT [40]	2023	◌	Distr.	Emul.	●	●	○	●	●	TT,SC
GFBE [50]	2024	○	Distr.	Emul.	●	●	●	●	●	SC
STABL [43]	2024	◑	Distr.	Emul.	○	●	●	●	●	TT,SC
Lilith	2024	●	Distr.	Emul.	●	●	●	●	●	TT,SC

- Lilith stands out as the tool that serve all the features and respect the criteria for blockchain benchmark.

PROBLEMS

PROBLEMS

Blockchain benchmark challenges

PROBLEMS

Blockchain benchmark challenges

- Framework-specific hurdles.

PROBLEMS

Blockchain benchmark challenges

- Framework-specific hurdles.
- Lack of standardized benchmarks.

PROBLEMS

Blockchain benchmark challenges

- Framework-specific hurdles.
- Lack of standardized benchmarks.
- Infrastructure setup complexity.

PROBLEMS

Blockchain benchmark challenges

- Framework-specific hurdles.
- Lack of standardized benchmarks.
- Infrastructure setup complexity.
- Lack of realistic workloads.

PROBLEMS

Blockchain benchmark challenges

- Framework-specific hurdles.
- Lack of standardized benchmarks.
- Infrastructure setup complexity.
- Lack of realistic workloads.
- Resource management and transaction handling.

PROBLEMS

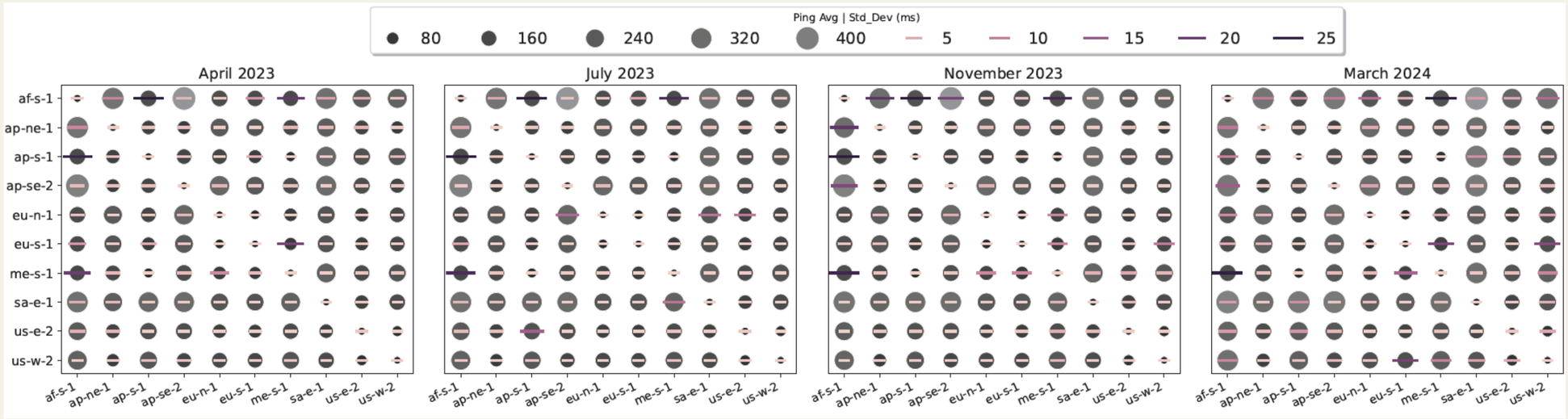
Blockchain benchmark challenges

- Framework-specific hurdles.
- Lack of standardized benchmarks.
- Infrastructure setup complexity.
- Lack of realistic workloads.
- Resource management and transaction handling.
- Steep learning curve.

CLOUD LIMITATIONS #1

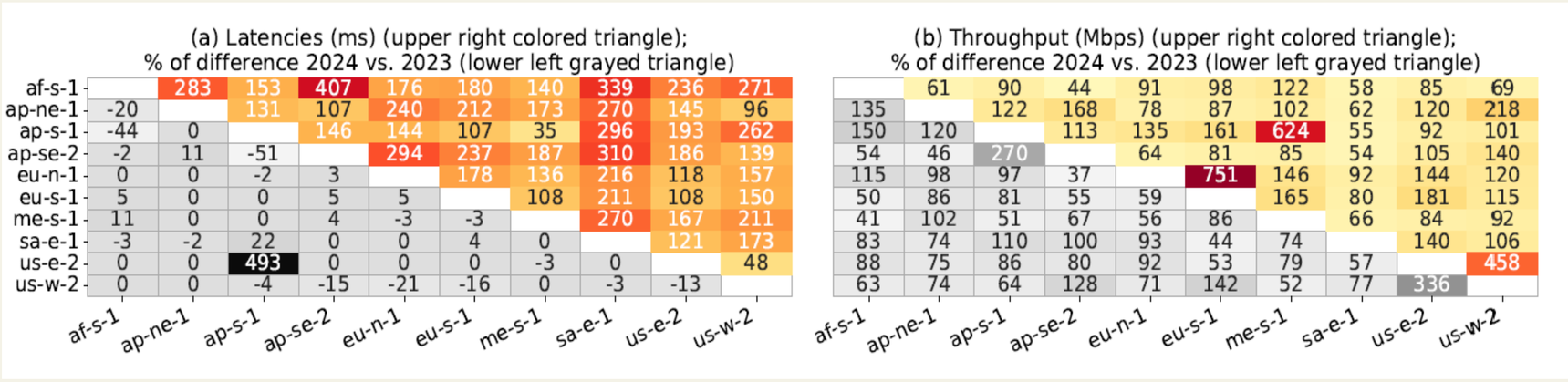
CLOUD LIMITATIONS #1

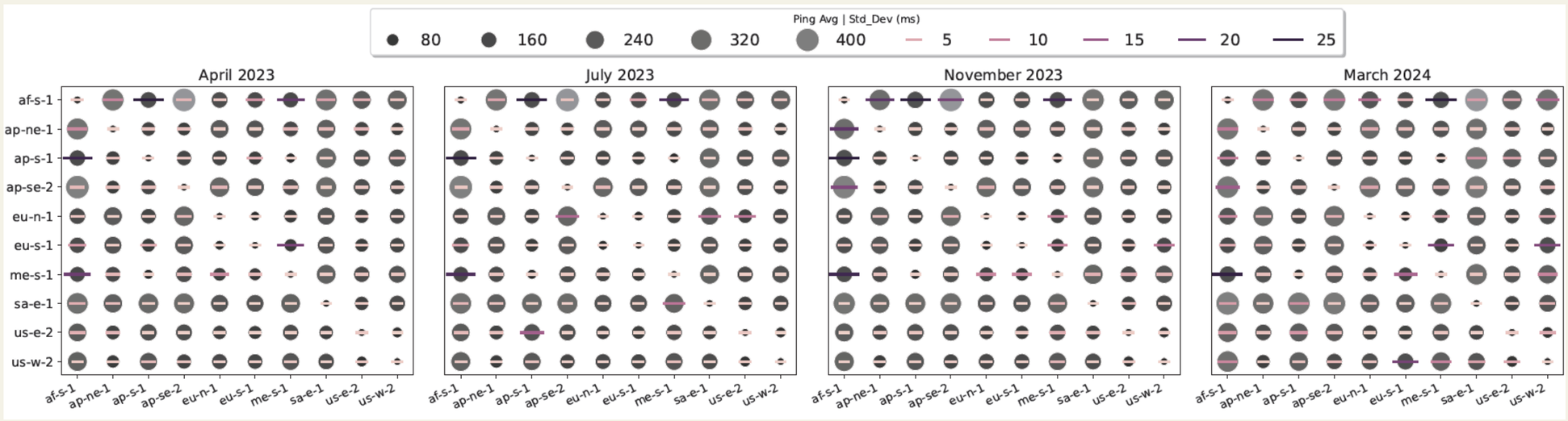
- Network links between the data centers changes throughout the day (e.g., infrastructure updates, contention on links, etc.).



CLOUD LIMITATIONS #1

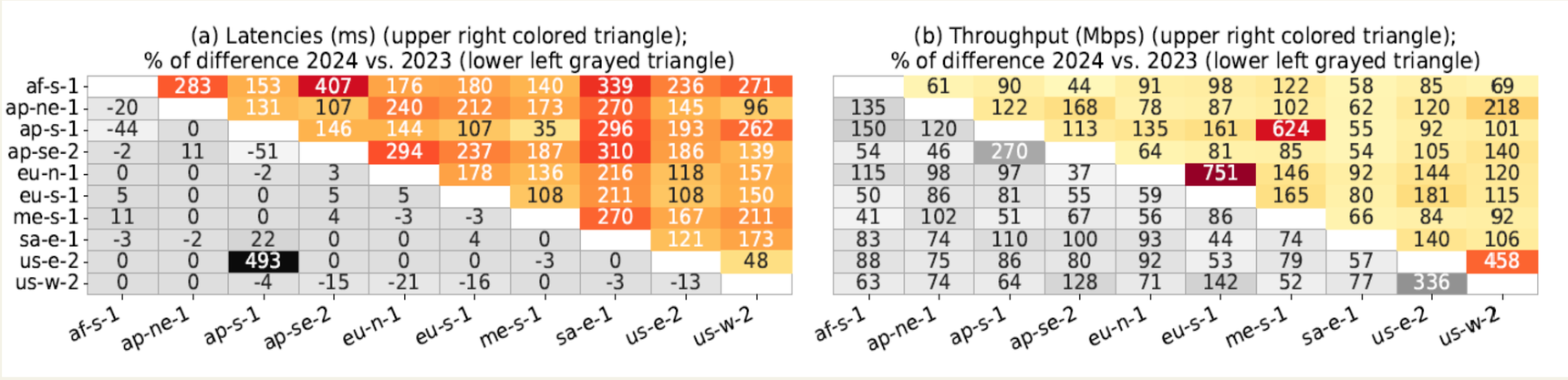
- Network links between the data centers changes throughout the day (e.g., infrastructure updates, contention on links, etc.).
- Infrastructure evolution (increased network throughput capacity and decreased latency).

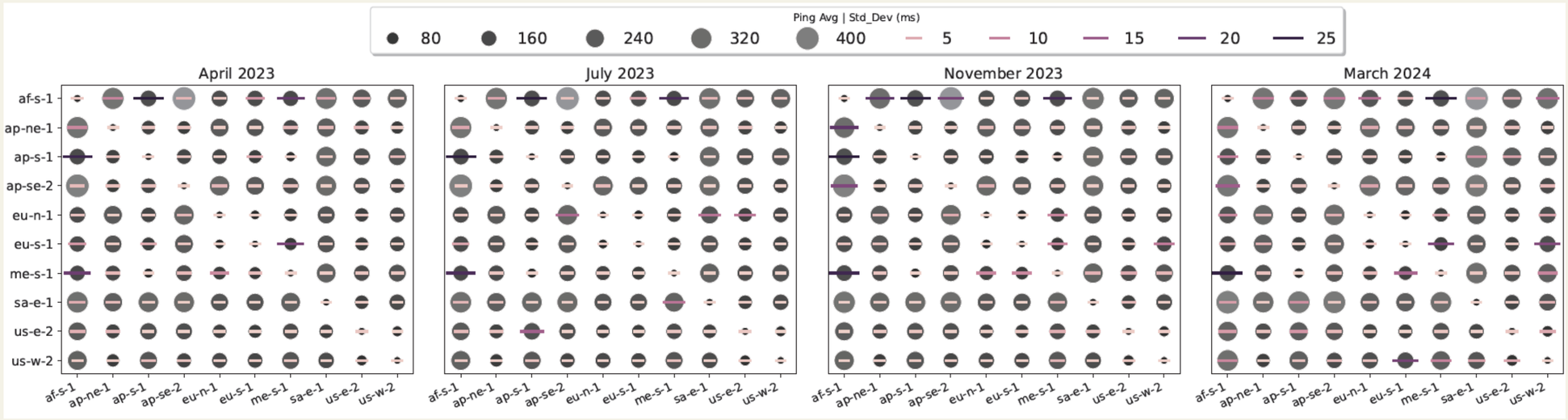




CLOUD LIMITATIONS #1

- Network links between the data centers changes throughout the day (e.g., infrastructure updates, contention on links, etc.).
- Infrastructure evolution (increased network throughput capacity and decreased latency).
- Limited opportunities to adjust network properties.

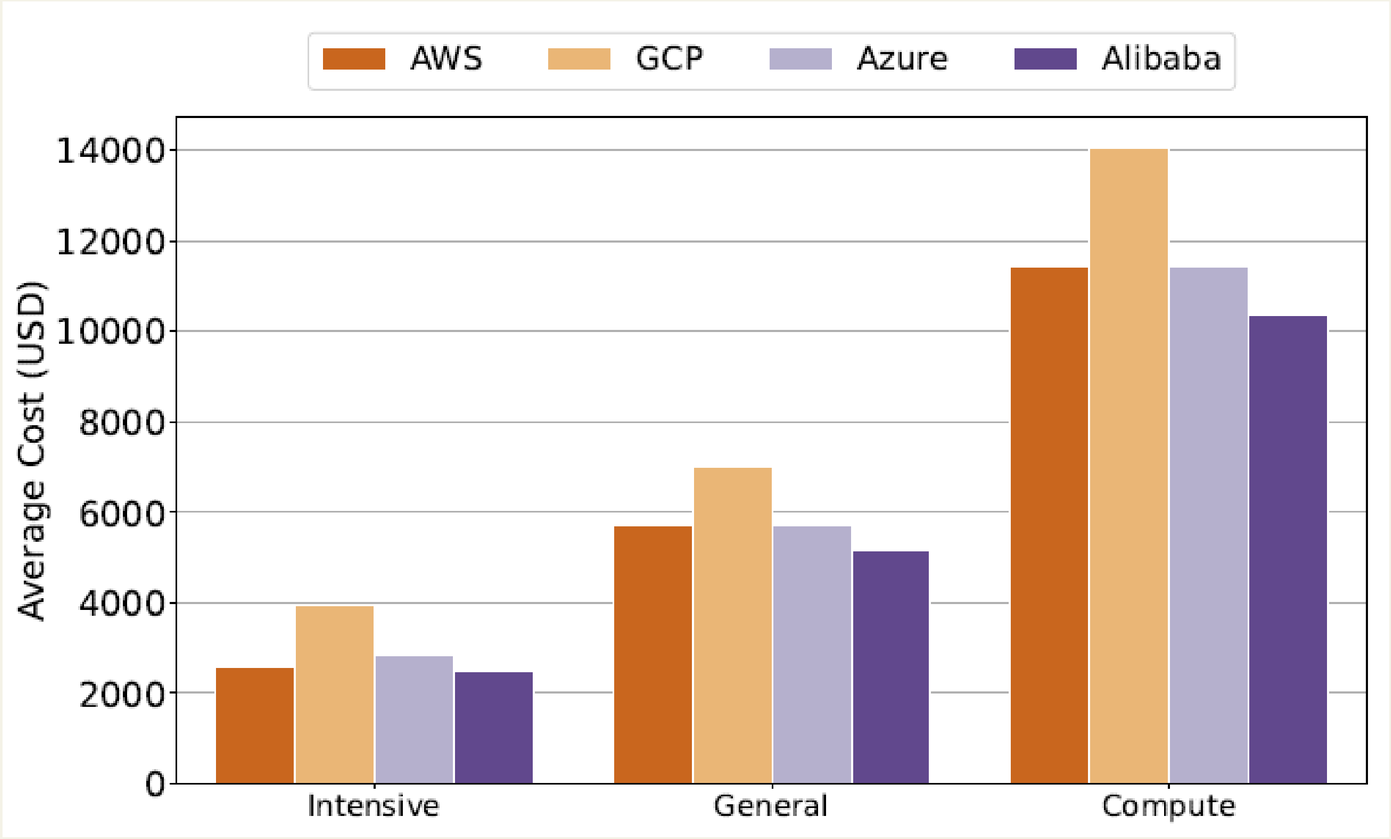




CLOUD LIMITATIONS #2

CLOUD LIMITATIONS #2

- Implementation and infrastructure costs (e.g., Amazon Web Service).



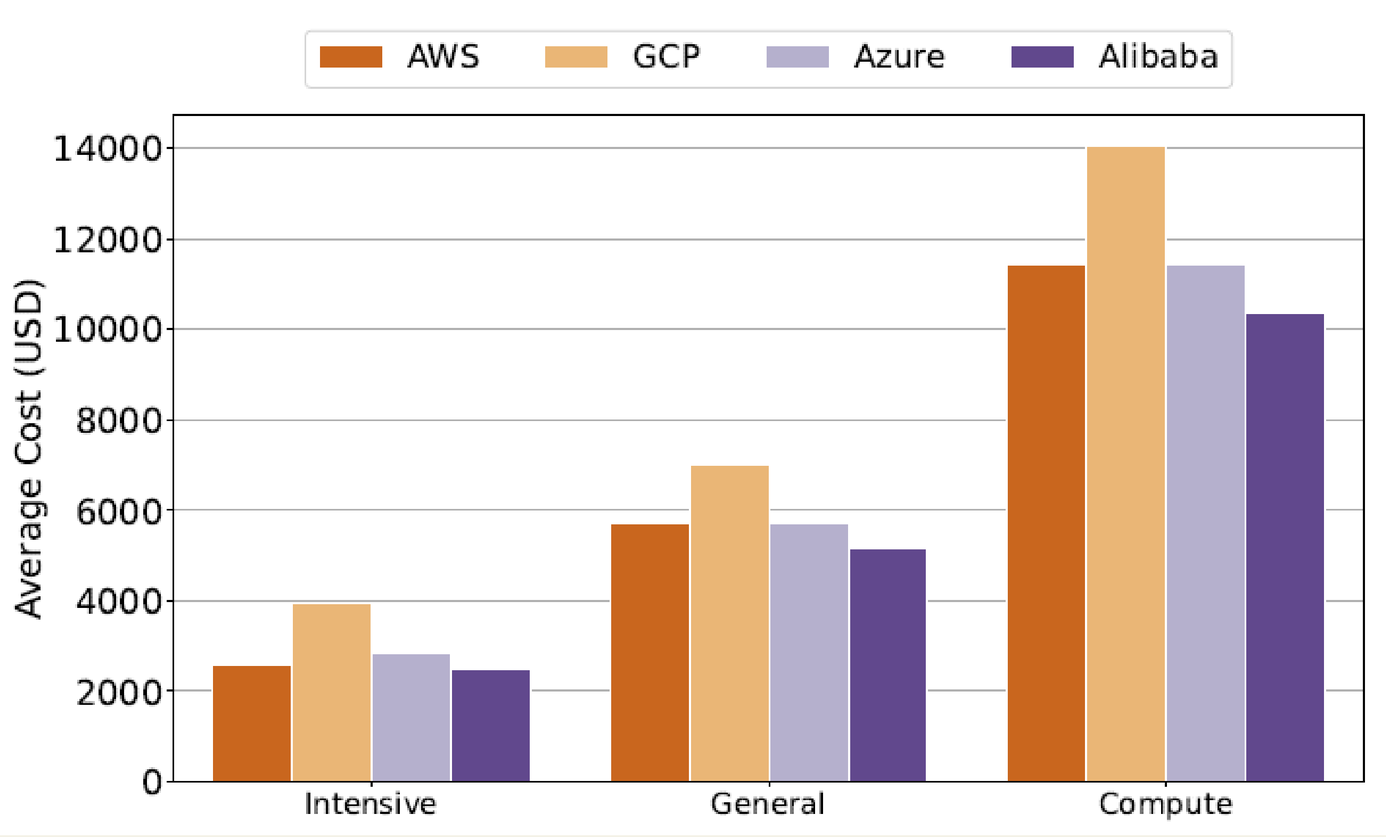
Intensive, 30/60 vCPU, 64/120 GiB RAM, 10 nodes

General, 4 vCPU, 8 GiB RAM, 200 nodes

Compute, 8 vCPU, 16 GiB RAM, 200 nodes

CLOUD LIMITATIONS #2

- Implementation and infrastructure costs (e.g., Amazon Web Service).
- Network modeling/malleability.



Intensive, 30/60 vCPU, 64/120 GiB RAM, 10 nodes
General, 4 vCPU, 8 GiB RAM, 200 nodes
Compute, 8 vCPU, 16 GiB RAM, 200 nodes

CONTRIBUTION

CONTRIBUTION

***LILITH*: A Topology-Aware Benchmark Tool for Blockchains**

CONTRIBUTION

LILITH: A Topology-Aware Benchmark Tool for Blockchains

1. Integrates Diablo benchmark suite and the distributed network topology emulator Kollaps.

CONTRIBUTION

LILITH: A Topology-Aware Benchmark Tool for Blockchains

1. Integrates Diablo benchmark suite and the distributed network topology emulator Kollaps.
2. Observe the performance of several blockchains **(Algorand, Diem, Ethereum PoA, Solana, Quorum-IBFT)** under different topologies.

CONTRIBUTION

***LILITH*: A Topology-Aware Benchmark Tool for Blockchains**

1. Integrates Diablo benchmark suite and the distributed network topology emulator Kollaps.

2. Observe the performance of several blockchains **(Algorand, Diem, Ethereum PoA, Solana, Quorum-IBFT)** under different topologies.

3. Serving a controlled environment.

CONTRIBUTION

LILITH: A Topology-Aware Benchmark Tool for Blockchains

1. Integrates Diablo benchmark suite and the distributed network topology emulator Kollaps.
2. Observe the performance of several blockchains **(Algorand, Diem, Ethereum PoA, Solana, Quorum-IBFT)** under different topologies.
3. Serving a controlled environment.
4. Assessing the feasibility of achieving comparable performance in a cost-effective cluster setup.

EVALUATIONS

EVALUATIONS

Research Questions

- **RQ1 -** How do topologies affect blockchain performance, and which is optimal for each?

- **RQ2 -** What is the impact on performance of network perturbations, such as packet loss, congestion, node failures, and increased latency, and which blockchain is most affected?

EVALUATIONS

Research Questions

- **RQ1 -** How do topologies affect blockchain performance, and which is optimal for each?

- **RQ2 -** What is the impact on performance of network perturbations, such as packet loss, congestion, node failures, and increased latency, and which blockchain is most affected?

EVALUATIONS

Research Questions

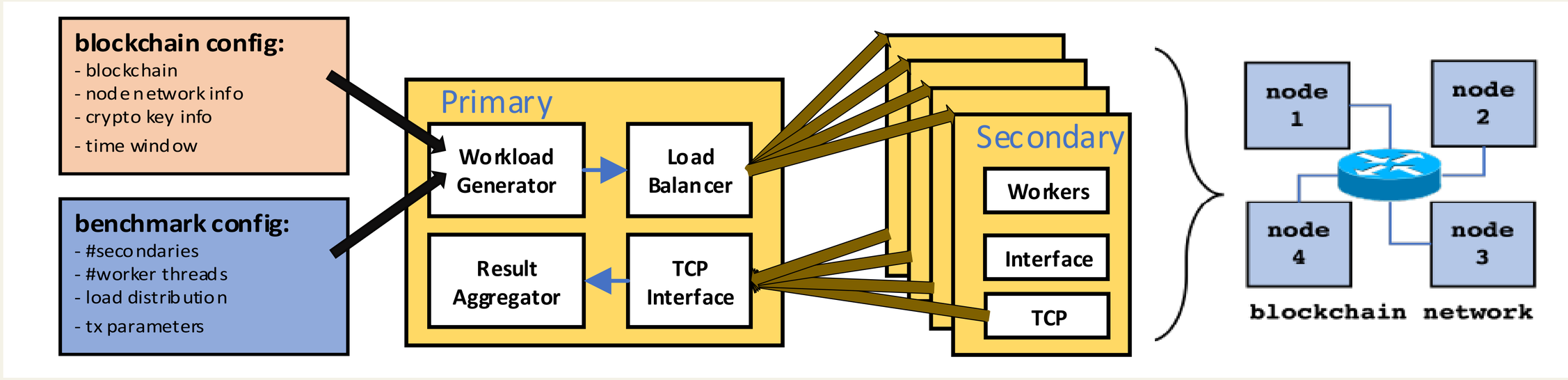
- **RQ1 -** How do topologies affect blockchain performance, and which is optimal for each?

- **RQ2 -** What is the impact on performance of network perturbations, such as packet loss, congestion, node failures, and increased latency, and which blockchain is most affected?

DIABLO BENCHMARK #1

DIABLO BENCHMARK #1

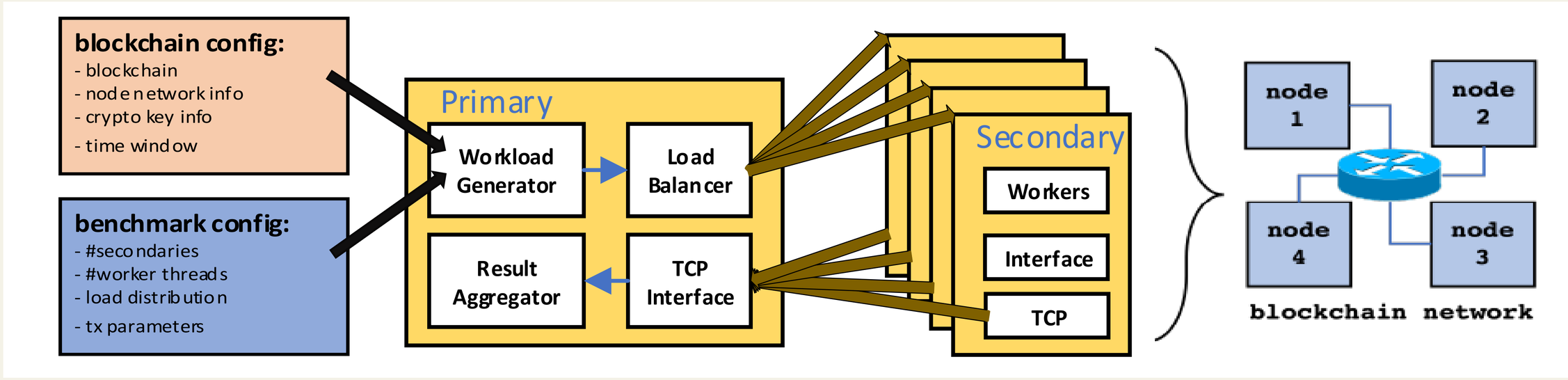
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **Primary** element transmits a description of the transactions to Secondaries elements, waits for all of them to be ready and informs when to start the benchmark.
- Each **Secondary** sends its results to the Primary and an aggregator collects them indicating the timestamps that can be used to generate time series, analyze latencies, etc.

DIABLO BENCHMARK #1

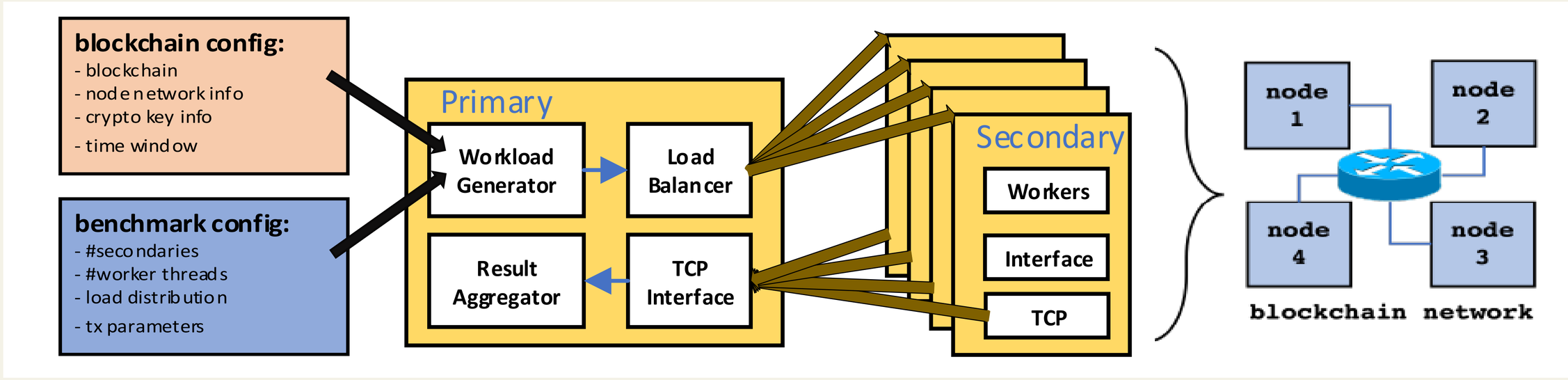
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **Primary** element transmits a description of the transactions to Secondaries elements, waits for all of them to be ready and informs when to start the benchmark.
- Each **Secondary** sends its results to the Primary and an aggregator collects them indicating the timestamps that can be used to generate time series, analyze latencies, etc.

DIABLO BENCHMARK #1

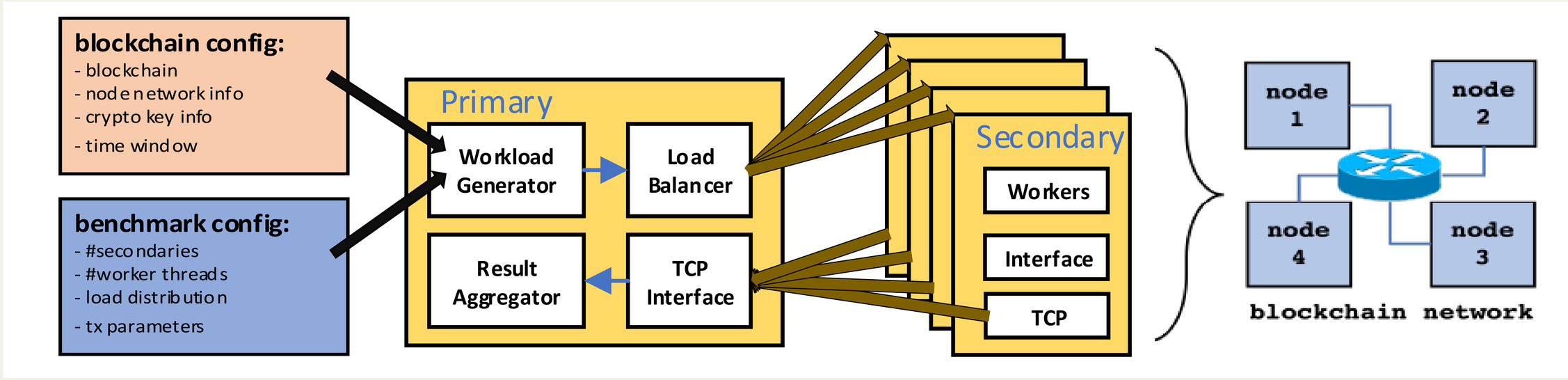
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **Primary** element transmits a description of the transactions to Secondaries elements, waits for all of them to be ready and informs when to start the benchmark.
- Each **Secondary** sends its results to the Primary and an aggregator collects them indicating the timestamps that can be used to generate time series, analyze latencies, etc.

DIABLO BENCHMARK #1

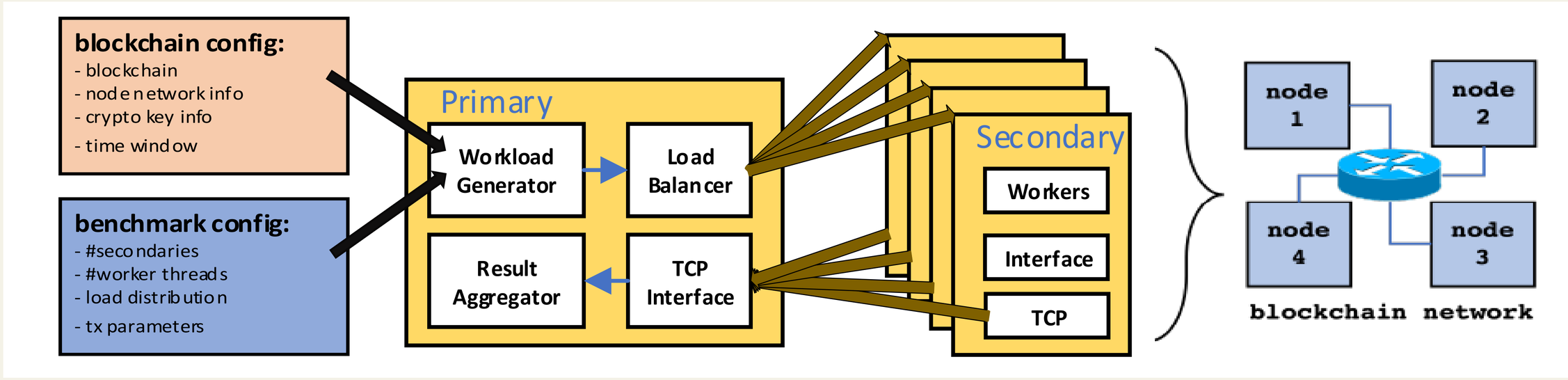
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **Primary** element transmits a description of the transactions to Secondaries elements, waits for all of them to be ready and informs when to start the benchmark.
- Each **Secondary** sends its results to the Primary and an aggregator collects them indicating the timestamps that can be used to generate time series, analyze latencies, etc.

DIABLO BENCHMARK #1

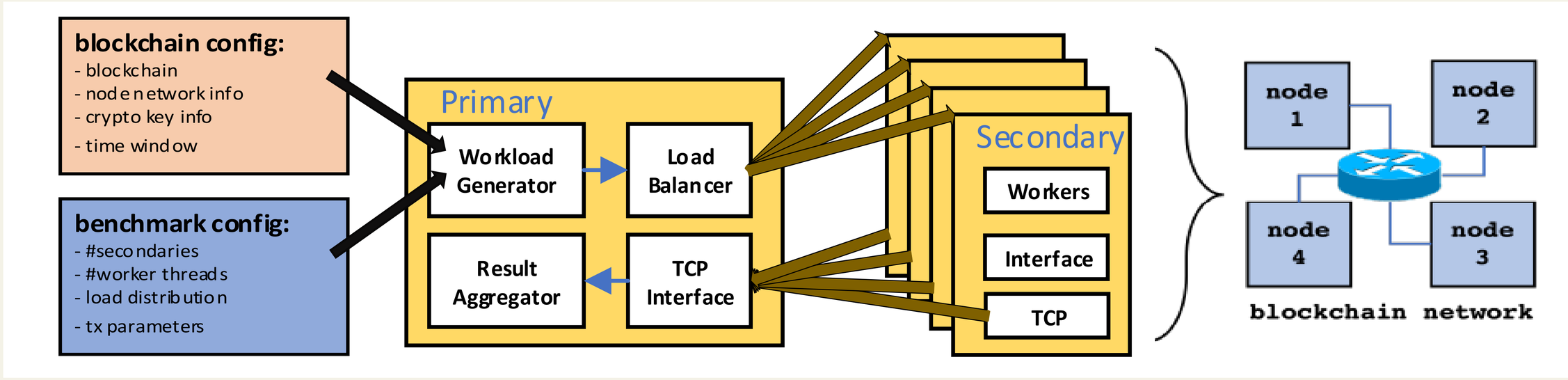
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **Primary** element transmits a description of the transactions to Secondaries elements, waits for all of them to be ready and informs when to start the benchmark.
- Each **Secondary** sends its results to the Primary and an aggregator collects them indicating the timestamps that can be used to generate time series, analyze latencies, etc.

DIABLO BENCHMARK #1

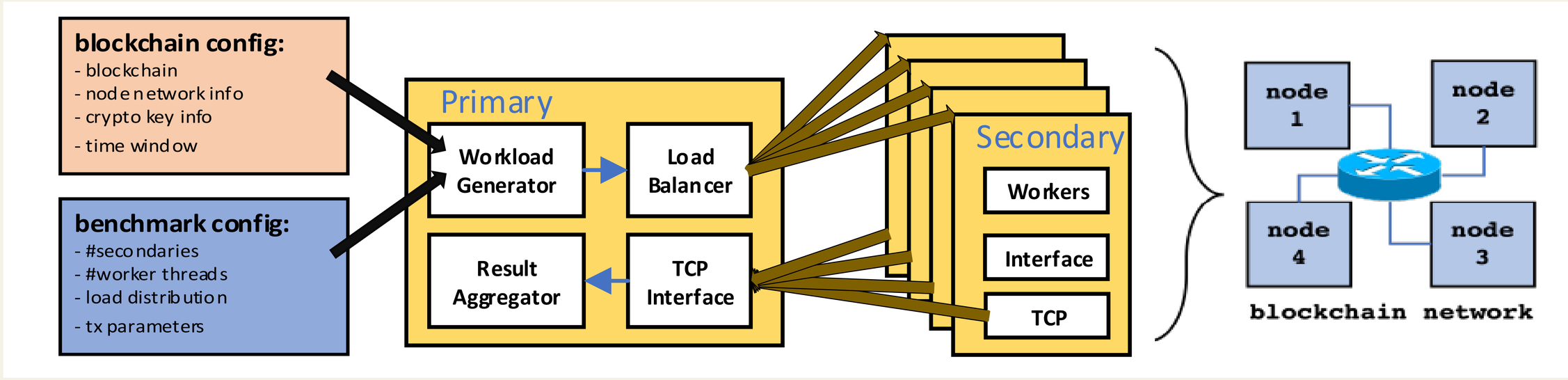
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **Primary** element transmits a description of the transactions to Secondaries elements, waits for all of them to be ready and informs when to start the benchmark.
- Each **Secondary** sends its results to the Primary and an aggregator collects them indicating the timestamps that can be used to generate time series, analyze latencies, etc.

DIABLO BENCHMARK #1

- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **Primary** element transmits a description of the transactions to Secondaries elements, waits for all of them to be ready and informs when to start the benchmark.
- Each **Secondary** sends its results to the Primary and an aggregator collects them indicating the timestamps that can be used to generate time series, analyze latencies, etc.

DIABLO BENCHMARK #2

DIABLO BENCHMARK #2

Benchmark definition

- Prior to starting the benchmark, the workload generator parses 2 configuration files:

- **benchmark** configuration file, defining the distribution of DIABLO Secondaries, transaction information, function calls, data types and variants of what data is passed into the transaction;

- **blockchain** configuration file, defining the experimental setup, denoting the configuration of machines to make the network of the blockchain.

DIABLO BENCHMARK #2

Benchmark definition

- Prior to starting the benchmark, the workload generator parses 2 configuration files:

- **benchmark** configuration file, defining the distribution of DIABLO Secondaries, transaction information, function calls, data types and variants of what data is passed into the transaction;

- **blockchain** configuration file, defining the experimental setup, denoting the configuration of machines to make the network of the blockchain.

DIABLO BENCHMARK #2

Benchmark definition

- Prior to starting the benchmark, the workload generator parses 2 configuration files:

- **benchmark** configuration file, defining the distribution of DIABLO Secondaries, transaction information, function calls, data types and variants of what data is passed into the transaction;

- **blockchain** configuration file, defining the experimental setup, denoting the configuration of machines to make the network of the blockchain.

DIABLO BENCHMARK #2

Benchmark definition

- Prior to starting the benchmark, the workload generator parses 2 configuration files:

- **benchmark** configuration file, defining the distribution of DIABLO Secondaries, transaction information, function calls, data types and variants of what data is passed into the transaction;

- **blockchain** configuration file, defining the experimental setup, denoting the configuration of machines to make the network of the blockchain.

DIABLO BENCHMARK #3

DIABLO BENCHMARK #3

Paper Experimental Settings

- 6 blockchains: *Algorand*, *Avalanche*, *Diem*, *Quorum*, *Ethereum*, *Solana*.

- 5 configurations on up to 200 machines distributed in 10 countries around the world.

- Diablo emulate clients sending either native or smart-contract transactions to the blockchain during the duration of the workload, e.g., 120 seconds at a constant rate of 1000 TPS **(order of magnitude as the average load of the VISA system)**.

- If the measured throughput is close to 1000 TPS, the blockchain handles the simple payment use case for the configuration.

DIABLO BENCHMARK #3

Paper Experimental Settings

- 6 blockchains: *Algorand*, *Avalanche*, *Diem*, *Quorum*, *Ethereum*, *Solana*.

- 5 configurations on up to 200 machines distributed in 10 countries around the world.

- Diablo emulate clients sending either native or smart-contract transactions to the blockchain during the duration of the workload, e.g., 120 seconds at a constant rate of 1000 TPS **(order of magnitude as the average load of the VISA system)**.

- If the measured throughput is close to 1000 TPS, the blockchain handles the simple payment use case for the configuration.

DIABLO BENCHMARK #3

Paper Experimental Settings

- 6 blockchains: *Algorand*, *Avalanche*, *Diem*, *Quorum*, *Ethereum*, *Solana*.

- 5 configurations on up to 200 machines distributed in 10 countries around the world.

- Diablo emulate clients sending either native or smart-contract transactions to the blockchain during the duration of the workload, e.g., 120 seconds at a constant rate of 1000 TPS **(order of magnitude as the average load of the VISA system)**.

- If the measured throughput is close to 1000 TPS, the blockchain handles the simple payment use case for the configuration.

DIABLO BENCHMARK #3

Paper Experimental Settings

- 6 blockchains: *Algorand*, *Avalanche*, *Diem*, *Quorum*, *Ethereum*, *Solana*.

- 5 configurations on up to 200 machines distributed in 10 countries around the world.

- Diablo emulate clients sending either native or smart-contract transactions to the blockchain during the duration of the workload, e.g., 120 seconds at a constant rate of 1000 TPS **(order of magnitude as the average load of the VISA system)**.

- If the measured throughput is close to 1000 TPS, the blockchain handles the simple payment use case for the configuration.

DIABLO BENCHMARK #3

Paper Experimental Settings

- 6 blockchains: *Algorand*, *Avalanche*, *Diem*, *Quorum*, *Ethereum*, *Solana*.

- 5 configurations on up to 200 machines distributed in 10 countries around the world.

- Diablo emulate clients sending either native or smart-contract transactions to the blockchain during the duration of the workload, e.g., 120 seconds at a constant rate of 1000 TPS **(order of magnitude as the average load of the VISA system)**.

- If the measured throughput is close to 1000 TPS, the blockchain handles the simple payment use case for the configuration.

NETWORK EMULATION

Name	Year	Orchestration	Concurrent deployments	Path congestions	Topology dynamics	Depl. Unit
DelayLine	1994	Centralized	✗	✗	✗	P
ModelNet	2002	Centralized	✗	✓	✓	P
Nist NET	2003	Centralized	✗	✗	✗	P
NetEm	2005	*note1	*note1	*note1	✗	P
Trickle	2005	*note1	*note1	*note1	✗	P
EmuSocket	2006	*note1	*note1	*note1	✗	P
ACIM/FlexLab	2007	Centralized	✗	✓	✓	V
NCTUns	2007	Centralized	✗	✓	✗	P
Emulab	2008	Centralized	✗	✓	✓	V
IMUNES	2008	Centralized	✗	✗	✗	P
MyP2P-World	2008	Centralized	✗	✗	✗	P
P2PLab	2008	Centralized	✗	✗	✗	P
Netkit	2008	Centralized	✗	✓	✗	V
DFS	2009	Centralized	✓	✗	✓	P
Dummynet	2010	Centralized	✗	✗	✗	P
Mininet	2010	Centralized	✗	✓	✓	P
SliceTime	2011	Centralized	✗	✓	✓	V
Mininet-HiFi	2012	Centralized	✗	✗	✓	C
SPLAYNET	2013	Decentralized	✓	✓	✓	P
MaxiNet	2014	Centralized	✗	✓	✓	P
EvalBox	2015	Centralized	✗	✗	✓	P
ContainerNet	2016	Centralized	✗	✓	✓	VC
Kathará	2018	Centralized	✗	✓	✗	C
Dockemu 2.0	2019	Centralized	✗	✗	✗	C
NEeaaS	2020	Decentralized	✗	✗	✗	VC
DockSDN	2021	Decentralized	✗	✗	✗	VC
Testground	2022	Centralized	✓	✗	✓	PC
KOLLAPS	2023	Decentralized	✓	✓	✓	PVC

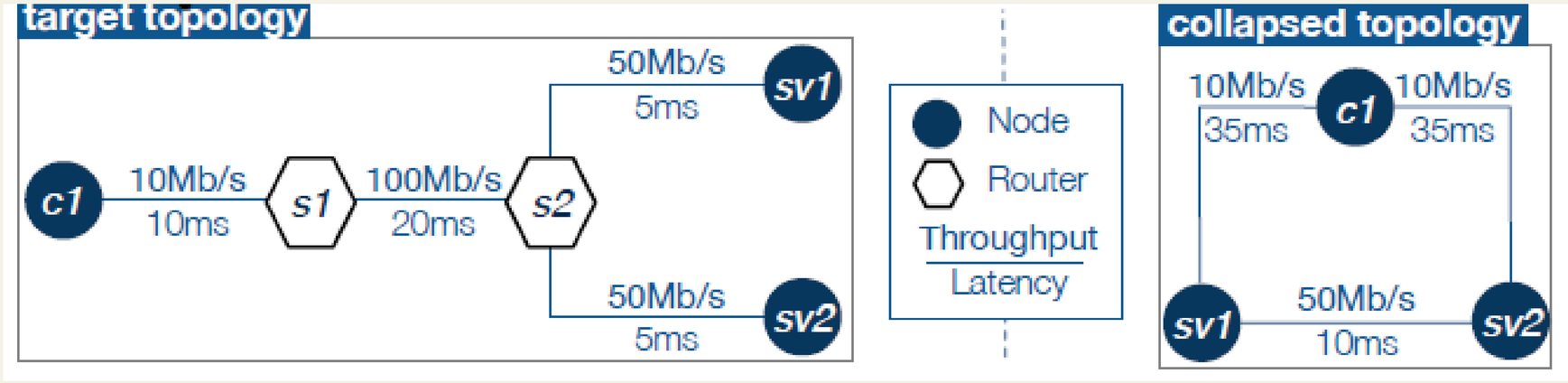
*P=process, V=virtual machine, C=container

*note1:(N/A: single link emulation only)

KOLLAPS #1

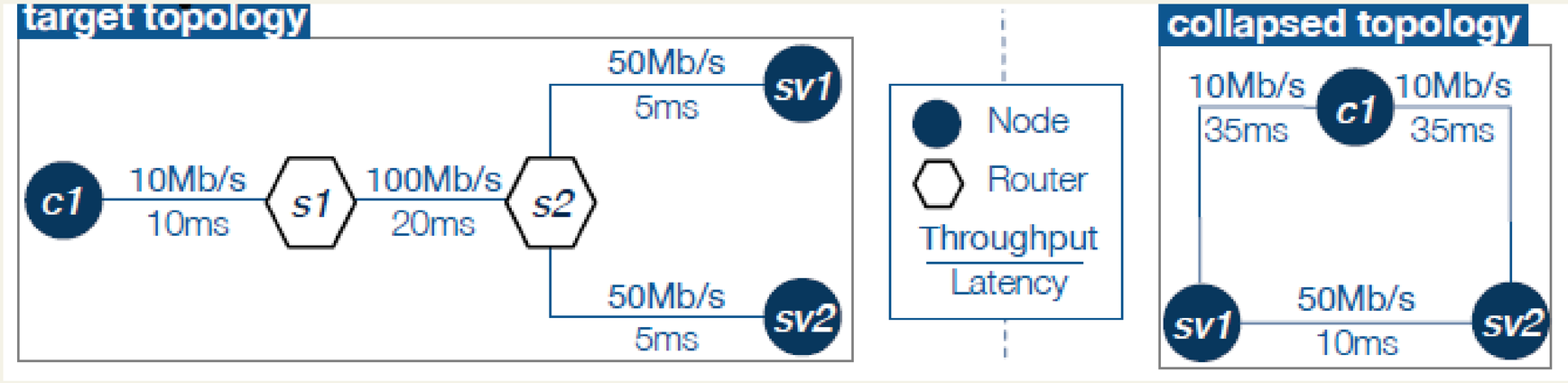
KOLLAPS #1

- Decentralized network emulator for large-scale applications.
- Observable end-to-end properties (latency, jitter, bandwidth and packet loss) without emulating the full-state of network elements (routers, switches).
- Fully-distributed emulation model allowing to scale without sacrificing accuracy.
- Quick changes to emulate dynamic events (e.g., link removals).
- Obtain a collapsed network topology.



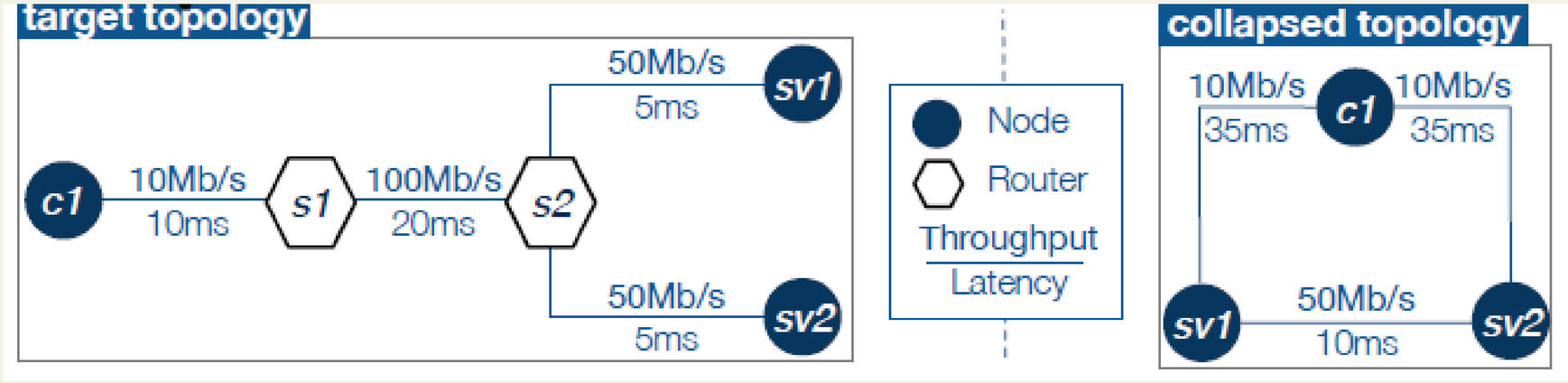
KOLLAPS #1

- Decentralized network emulator for large-scale applications.
- Observable end-to-end properties (latency, jitter, bandwidth and packet loss) without emulating the full-state of network elements (routers, switches).
- Fully-distributed emulation model allowing to scale without sacrificing accuracy.
- Quick changes to emulate dynamic events (e.g., link removals).
- Obtain a collapsed network topology.



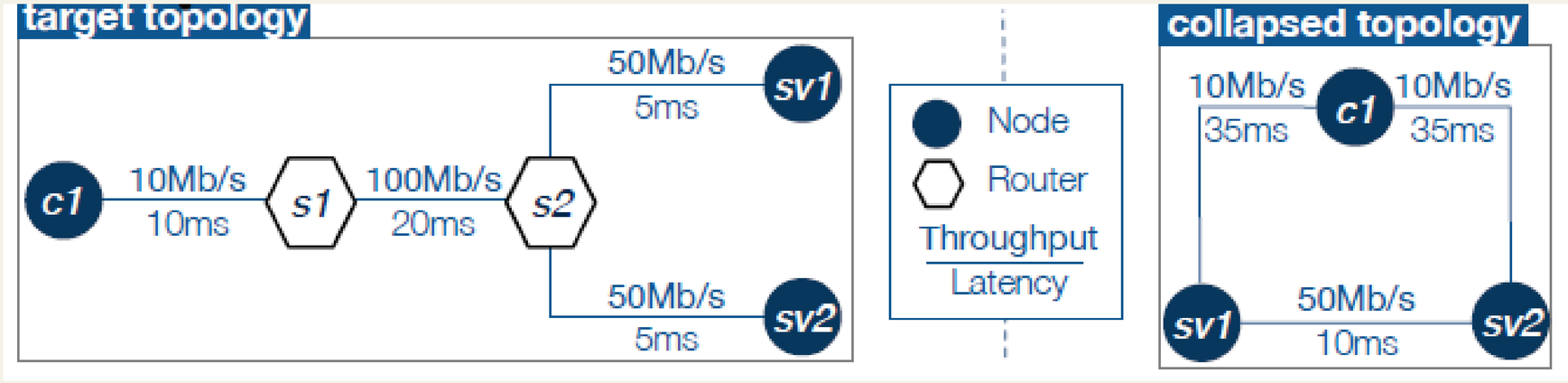
KOLLAPS #1

- Decentralized network emulator for large-scale applications.
- Observable end-to-end properties (latency, jitter, bandwidth and packet loss) without emulating the full-state of network elements (routers, switches).
- Fully-distributed emulation model allowing to scale without sacrificing accuracy.
- Quick changes to emulate dynamic events (e.g., link removals).
- Obtain a collapsed network topology.



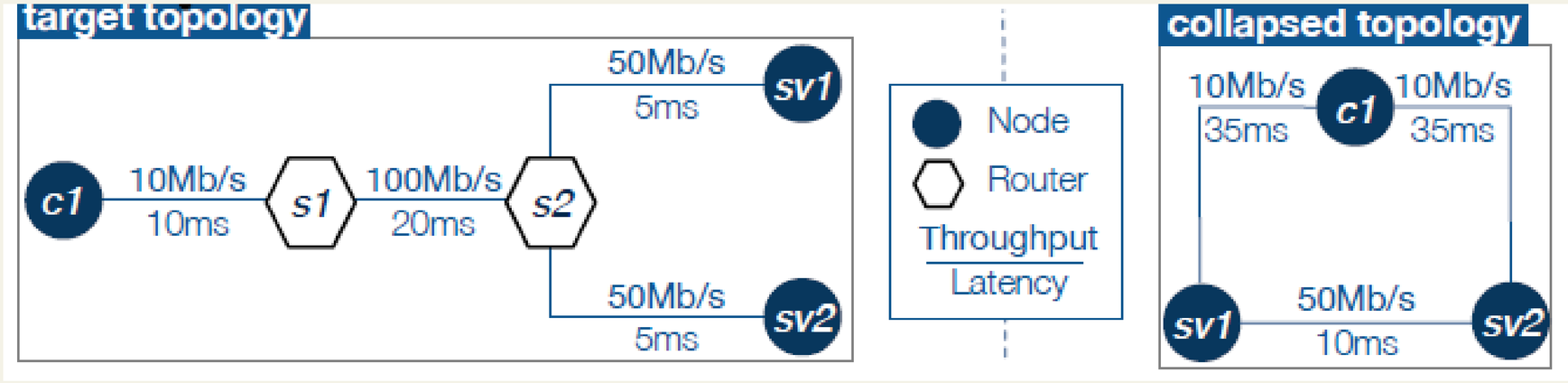
KOLLAPS #1

- Decentralized network emulator for large-scale applications.
- Observable end-to-end properties (latency, jitter, bandwidth and packet loss) without emulating the full-state of network elements (routers, switches).
- Fully-distributed emulation model allowing to scale without sacrificing accuracy.
- Quick changes to emulate dynamic events (e.g., link removals).
- Obtain a collapsed network topology.



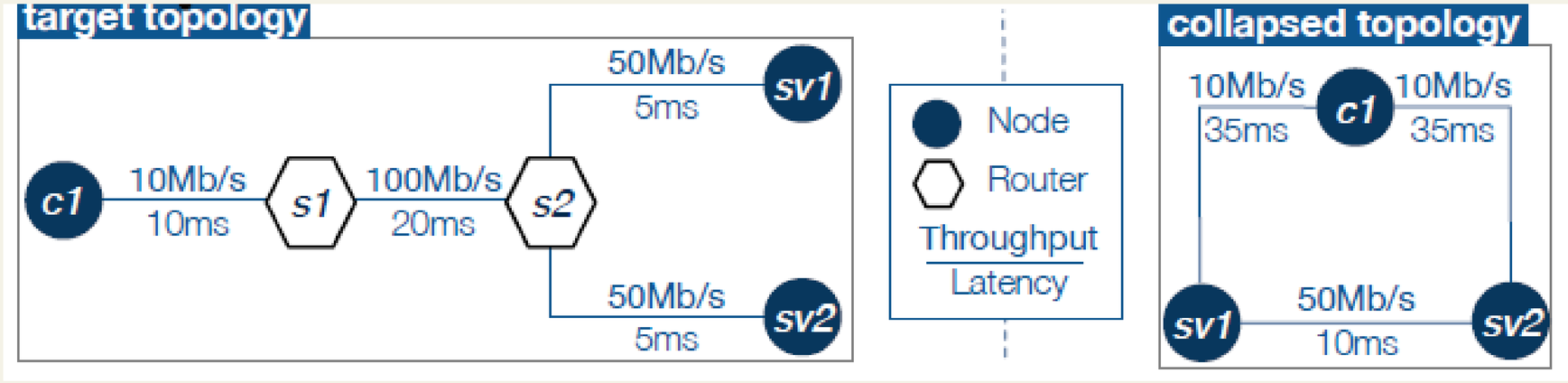
KOLLAPS #1

- Decentralized network emulator for large-scale applications.
- Observable end-to-end properties (latency, jitter, bandwidth and packet loss) without emulating the full-state of network elements (routers, switches).
- Fully-distributed emulation model allowing to scale without sacrificing accuracy.
- Quick changes to emulate dynamic events (e.g., link removals).
- Obtain a collapsed network topology.



KOLLAPS #1

- Decentralized network emulator for large-scale applications.
- Observable end-to-end properties (latency, jitter, bandwidth and packet loss) without emulating the full-state of network elements (routers, switches).
- Fully-distributed emulation model allowing to scale without sacrificing accuracy.
- Quick changes to emulate dynamic events (e.g., link removals).
- Obtain a collapsed network topology.



KOLLAPS #2

KOLLAPS #2

Components

- **Emulation Manager,** a single instance per physical machine responsible for maintaining the emulation model.

- **TC Abstraction Layer** (TCAL), once per application container, it retrieves and sets the link properties.

- **Bootstrapper,** once per physical machine, it initiates Kollaps in Docker Swarm deployments (not needed under Kubernetes).

- Web-based **Dashboard,** to monitor the experiments.

- **Deployment Generator** converts an experiment description into a deployment plan.

KOLLAPS #2

Components

- **Emulation Manager,** a single instance per physical machine responsible for maintaining the emulation model.

- **TC Abstraction Layer** (TCAL), once per application container, it retrieves and sets the link properties.

- **Bootstrapper,** once per physical machine, it initiates Kollaps in Docker Swarm deployments (not needed under Kubernetes).

- Web-based **Dashboard,** to monitor the experiments.

- **Deployment Generator** converts an experiment description into a deployment plan.

KOLLAPS #2

Components

- **Emulation Manager,** a single instance per physical machine responsible for maintaining the emulation model.

- **TC Abstraction Layer** (TCAL), once per application container, it retrieves and sets the link properties.

- **Bootstrapper,** once per physical machine, it initiates Kollaps in Docker Swarm deployments (not needed under Kubernetes).

- Web-based **Dashboard,** to monitor the experiments.

- **Deployment Generator** converts an experiment description into a deployment plan.

KOLLAPS #2

Components

- **Emulation Manager,** a single instance per physical machine responsible for maintaining the emulation model.

- **TC Abstraction Layer** (TCAL), once per application container, it retrieves and sets the link properties.

- **Bootstrapper,** once per physical machine, it initiates Kollaps in Docker Swarm deployments (not needed under Kubernetes).

- Web-based **Dashboard,** to monitor the experiments.

- **Deployment Generator** converts an experiment description into a deployment plan.

KOLLAPS #2

Components

- **Emulation Manager,** a single instance per physical machine responsible for maintaining the emulation model.

- **TC Abstraction Layer** (TCAL), once per application container, it retrieves and sets the link properties.

- **Bootstrapper,** once per physical machine, it initiates Kollaps in Docker Swarm deployments (not needed under Kubernetes).

- Web-based **Dashboard,** to monitor the experiments.

- **Deployment Generator** converts an experiment description into a deployment plan.

KOLLAPS #2

Components

- **Emulation Manager,** a single instance per physical machine responsible for maintaining the emulation model.

- **TC Abstraction Layer** (TCAL), once per application container, it retrieves and sets the link properties.

- **Bootstrapper,** once per physical machine, it initiates Kollaps in Docker Swarm deployments (not needed under Kubernetes).

- Web-based **Dashboard,** to monitor the experiments.

- **Deployment Generator** converts an experiment description into a deployment plan.

KOLLAPS #3

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

KOLLAPS #3

Implementation and evaluation

- Integration with container images and orchestration tools (e.g., Docker Swarm, Kubernetes) need to configure:

  1. orchestrator APIs (used at runtime for name resolution);
  2. topology descriptor file (read by each Em instance to setup the initial network state and compute the graph of the dynamic changes);
  3. setup of multiple virtual networks.

- Similar results by running applications with Kollaps in a cluster or in Amazon EC2, scaling and having constant cost regardless of the emulated bandwidth usage.

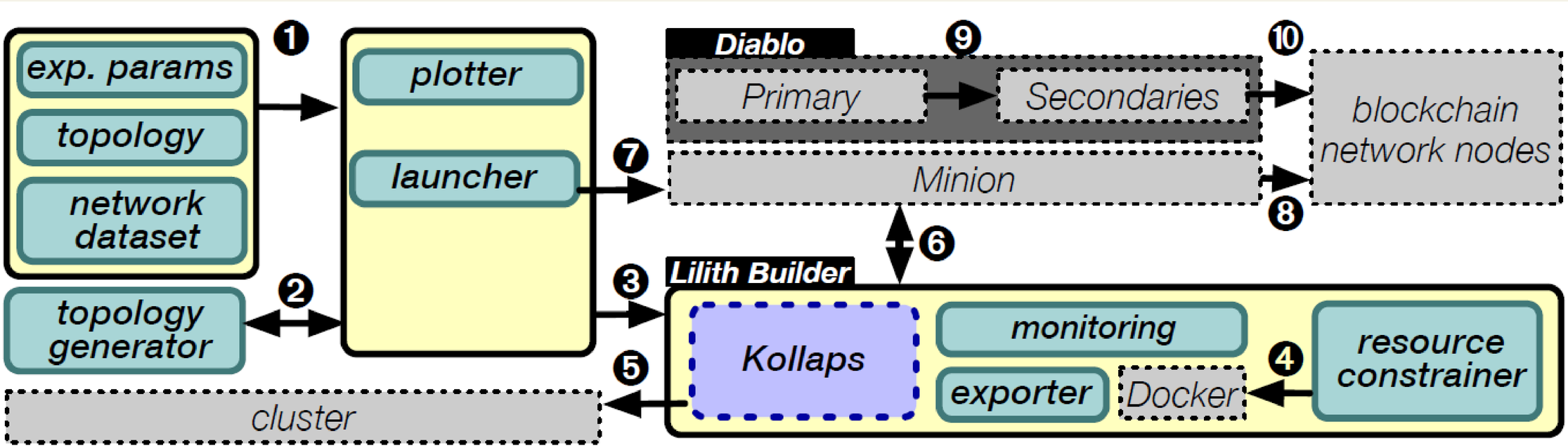
- Emulation accuracy comparable with full network state emulator (Mininet).

- Enables what-if scenario (e.g., moving Cassandra nodes in different countries) by changing the topology configuration file.

IMPLEMENTATION

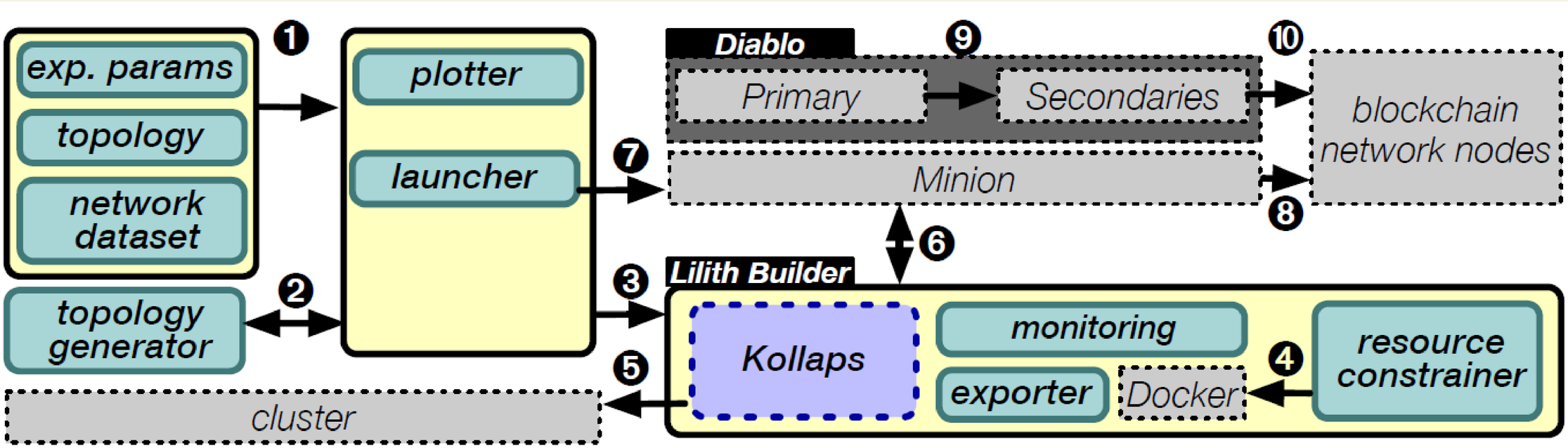
IMPLEMENTATION

Benchmarking process



IMPLEMENTATION

Benchmarking process



TESTING SCENARIO

TESTING SCENARIO

Workflow Evolution

amazon web services™ | EC2

```
<experiment boot="kollaps:2.0">
  <services>
    <service name="dashboard" image="kollaps/dashboard:1.0" supervisor="true"
    <service name="client1" image="kollaps/iperf3-client:1.0" command="['serve
    <service name="server" image="kollaps/iperf3-server:1.0" command="['server
  </services>

  <bridges>
    <bridge name='b1' />
  </bridges>

  <links>
    <link origin="client1" dest="server" latency="0.001" upload="22Mbps" downl
  </links>

  <dynamic>
    <schedule name="client1" time="0.0" action="join">
    <schedule name="server" time="0.0" action="join">
  </dynamic>
</experiment>
```

TESTING SCENARIO

Workflow Evolution

amazon web services™ | EC2

```
<experiment boot="kollaps:2.0">
  <services>
    <service name="dashboard" image="kollaps/dashboard:1.0" supervisor="true"
    <service name="client1" image="kollaps/iperf3-client:1.0" command="['serve
    <service name="server" image="kollaps/iperf3-server:1.0" command="['server
  </services>

  <bridges>
    <bridge name='b1' />
  </bridges>

  <links>
    <link origin="client1" dest="server" latency="0.001" upload="22Mbps" downl
  </links>

  <dynamic>
    <schedule name="client1" time="0.0" action="join">
    <schedule name="server" time="0.0" action="join">
  </dynamic>
</experiment>
```

TESTING SCENARIO

Workflow Evolution

amazon web services™ | EC2

```
<experiment boot="kollaps:2.0">
  <services>
    <service name="dashboard" image="kollaps/dashboard:1.0" supervisor="true"
    <service name="client1" image="kollaps/iperf3-client:1.0" command="['serve
    <service name="server" image="kollaps/iperf3-server:1.0" command="['server
  </services>

  <bridges>
    <bridge name='b1' />
  </bridges>

  <links>
    <link origin="client1" dest="server" latency="0.001" upload="22Mbps" downl
  </links>

  <dynamic>
    <schedule name="client1" time="0.0" action="join">
    <schedule name="server" time="0.0" action="join">
  </dynamic>
</experiment>
```

TESTING SCENARIO

Workflow Evolution

amazon web services™ | EC2

```
<experiment boot="kollaps:2.0">
  <services>
    <service name="dashboard" image="kollaps/dashboard:1.0" supervisor="true"
    <service name="client1" image="kollaps/iperf3-client:1.0" command="['serve
    <service name="server" image="kollaps/iperf3-server:1.0" command="['server
  </services>

  <bridges>
    <bridge name='b1' />
  </bridges>

  <links>
    <link origin="client1" dest="server" latency="0.001" upload="22Mbps" downl
  </links>

  <dynamic>
    <schedule name="client1" time="0.0" action="join">
    <schedule name="server" time="0.0" action="join">
  </dynamic>
</experiment>
```

GOALS

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

GOALS

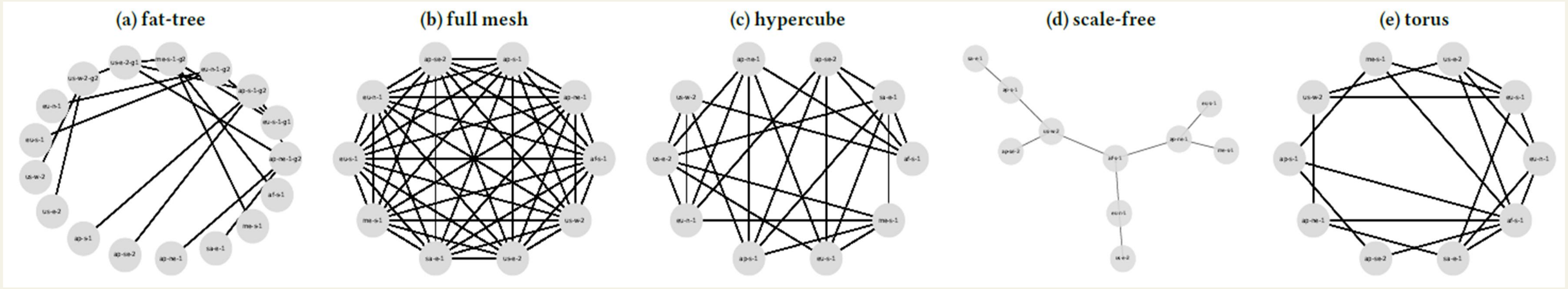
- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.
- Integration of multiple blockchain systems.
- Improved result analytics alongside blockchain node log files.
- Transitioning from cloud to cluster, resulting in cost reduction.
- Facilitate real-world benchmarking scenarios.
- Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- Enrich already existing documentation by providing both network datasets and execution traces.

TOPOLOGIES #1

TOPOLOGIES #1

Real-world network topologies employed

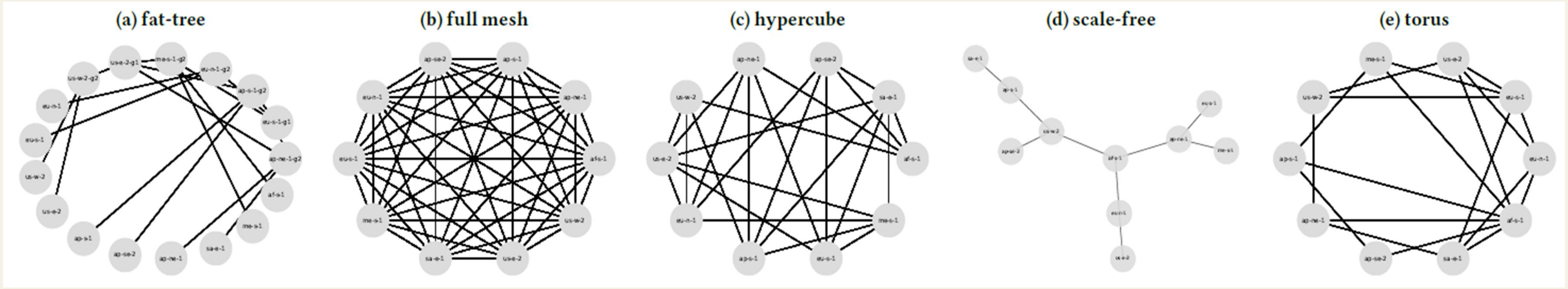
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional cube.
- ***Scale-free***. A few nodes have significantly more connections than others.
- ***Torus***. Resembling a grid where each node is connected to its adjacent nodes in a wrap-around fashion.



TOPOLOGIES #1

Real-world network topologies employed

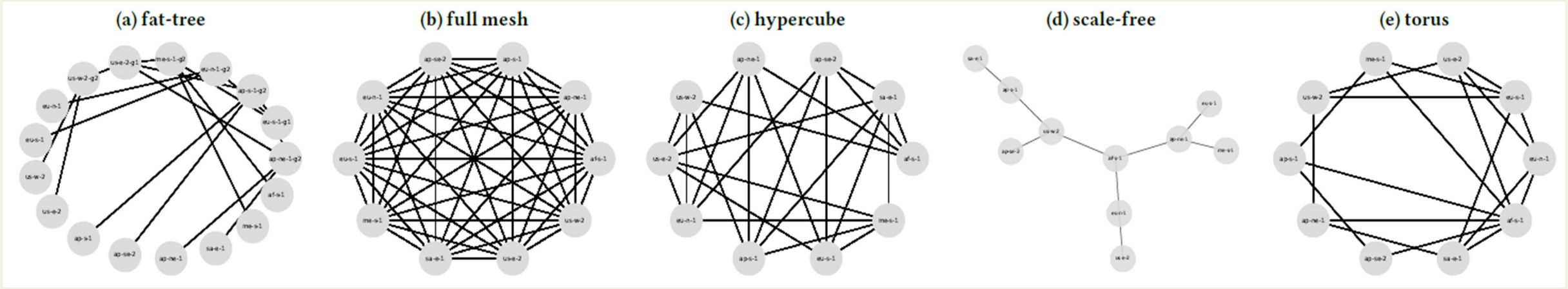
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional cube.
- ***Scale-free***. A few nodes have significantly more connections than others.
- ***Torus***. Resembling a grid where each node is connected to its adjacent nodes in a wrap-around fashion.



TOPOLOGIES #1

Real-world network topologies employed

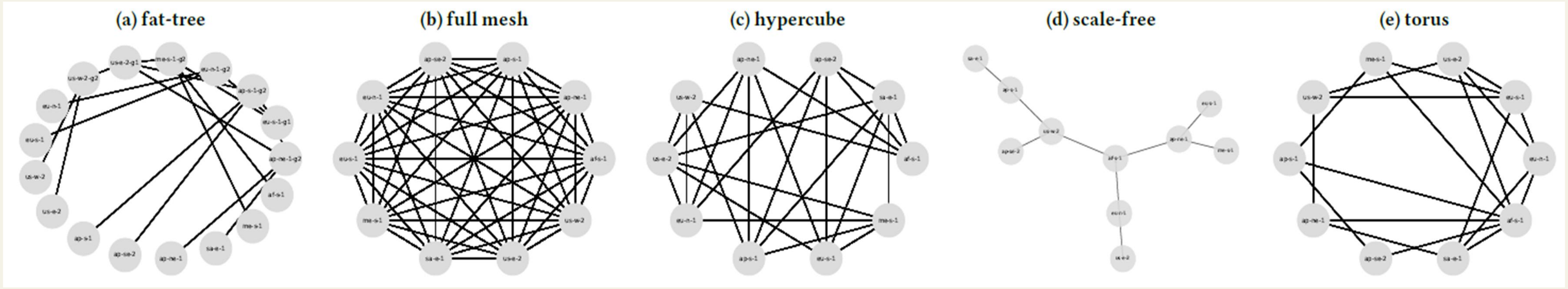
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional cube.
- ***Scale-free***. A few nodes have significantly more connections than others.
- ***Torus***. Resembling a grid where each node is connected to its adjacent nodes in a wrap-around fashion.



TOPOLOGIES #1

Real-world network topologies employed

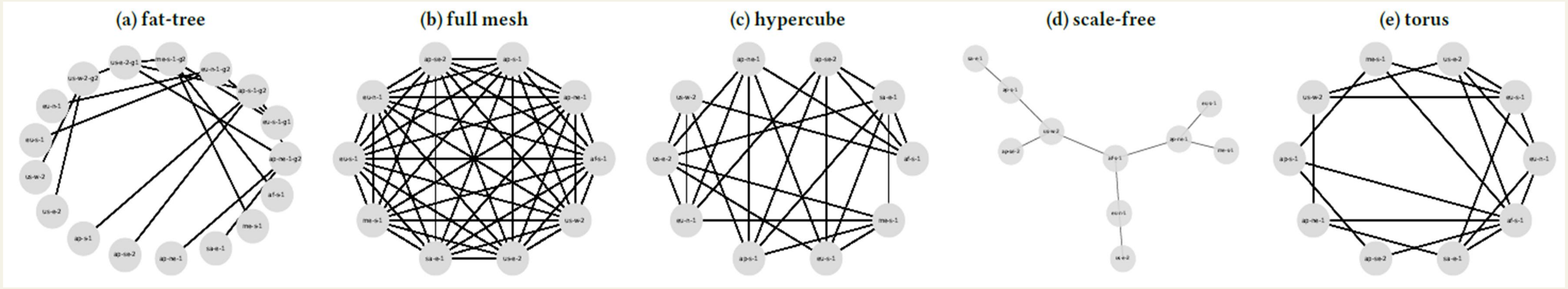
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional cube.
- ***Scale-free***. A few nodes have significantly more connections than others.
- ***Torus***. Resembling a grid where each node is connected to its adjacent nodes in a wrap-around fashion.



TOPOLOGIES #1

Real-world network topologies employed

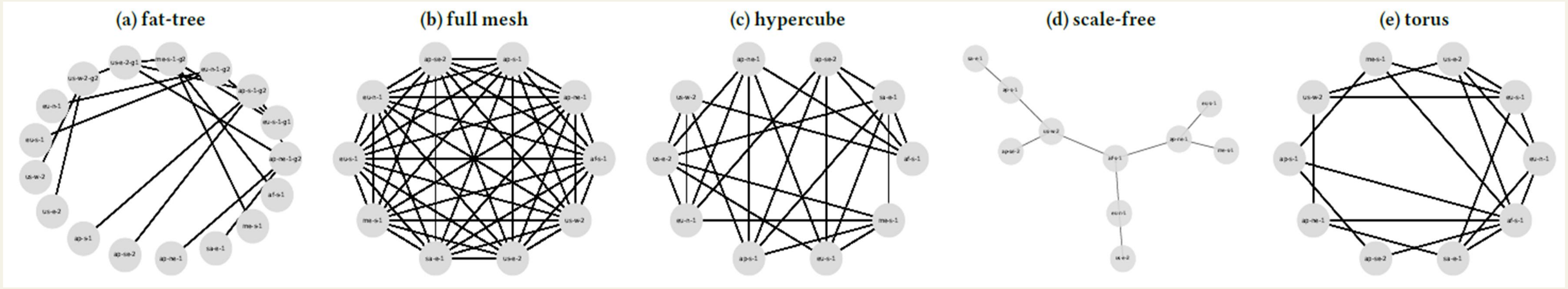
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional cube.
- ***Scale-free***. A few nodes have significantly more connections than others.
- ***Torus***. Resembling a grid where each node is connected to its adjacent nodes in a wrap-around fashion.



TOPOLOGIES #1

Real-world network topologies employed

- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional cube.
- ***Scale-free***. A few nodes have significantly more connections than others.
- ***Torus***. Resembling a grid where each node is connected to its adjacent nodes in a wrap-around fashion.



TOPOLOGIES #2

TOPOLOGIES #2

Kollaps Network construction

- Network measurements in the cloud don't provide much insight into the underlying infrastructure.

- Having node with degree >= 2 in Kollaps, it is not possible to obtain the same network performance (Kollaps activates a best-route selection mechanism).

- This confirms that measurements in the cloud only show a part of its network infrastructure design.

Topology	Degree	Links	Avg. Lat. (ms)	Avg. TPut (Mbps)
fat-tree (k ports=4, l level=2)	k	$k^l/2$	102	712
full mesh (N nodes)	$N-1$	$N \times (N-1)/2$	194	600
hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	208	560
scale-free (N nodes)	(Based on Power Law)		218	432
torus (N nodes/row=5, n dim.=2)	$2n$	$n \cdot N^n$	197	728

TOPOLOGIES #2

Kollaps Network construction

- Network measurements in the cloud don't provide much insight into the underlying infrastructure.

- Having node with degree >= 2 in Kollaps, it is not possible to obtain the same network performance (Kollaps activates a best-route selection mechanism).

- This confirms that measurements in the cloud only show a part of its network infrastructure design.

Topology	Degree	Links	Avg. Lat. (ms)	Avg. TPut (Mbps)
fat-tree (k ports=4, l level=2)	k	$k^l/2$	102	712
full mesh (N nodes)	$N-1$	$N \times (N-1)/2$	194	600
hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	208	560
scale-free (N nodes)	(Based on Power Law)		218	432
torus (N nodes/row=5, n dim.=2)	$2n$	$n \cdot N^n$	197	728

TOPOLOGIES #2

Kollaps Network construction

- Network measurements in the cloud don't provide much insight into the underlying infrastructure.

- Having node with degree >= 2 in Kollaps, it is not possible to obtain the same network performance (Kollaps activates a best-route selection mechanism).

- This confirms that measurements in the cloud only show a part of its network infrastructure design.

Topology	Degree	Links	Avg. Lat. (ms)	Avg. TPut (Mbps)
fat-tree (k ports=4, l level=2)	k	$k^l/2$	102	712
full mesh (N nodes)	$N-1$	$N \times (N-1)/2$	194	600
hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	208	560
scale-free (N nodes)	(Based on Power Law)		218	432
torus (N nodes/row=5, n dim.=2)	$2n$	$n \cdot N^n$	197	728

TOPOLOGIES #2

Kollaps Network construction

- Network measurements in the cloud don't provide much insight into the underlying infrastructure.

- Having node with degree >= 2 in Kollaps, it is not possible to obtain the same network performance (Kollaps activates a best-route selection mechanism).

- This confirms that measurements in the cloud only show a part of its network infrastructure design.

Topology	Degree	Links	Avg. Lat. (ms)	Avg. TPut (Mbps)
fat-tree (k ports=4, l level=2)	k	$k^l/2$	102	712
full mesh (N nodes)	$N-1$	$N \times (N-1)/2$	194	600
hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	208	560
scale-free (N nodes)	(Based on Power Law)		218	432
torus (N nodes/row=5, n dim.=2)	$2n$	$n \cdot N^n$	197	728

TOPOLOGIES #2

Kollaps Network construction

- Network measurements in the cloud don't provide much insight into the underlying infrastructure.

- Having node with degree >= 2 in Kollaps, it is not possible to obtain the same network performance (Kollaps activates a best-route selection mechanism).

- This confirms that measurements in the cloud only show a part of its network infrastructure design.

Topology	Degree	Links	Avg. Lat. (ms)	Avg. TPut (Mbps)
fat-tree (k ports=4, l level=2)	k	$k^l/2$	102	712
full mesh (N nodes)	$N-1$	$N \times (N-1)/2$	194	600
hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	208	560
scale-free (N nodes)	(Based on Power Law)		218	432
torus (N nodes/row=5, n dim.=2)	$2n$	$n \cdot N^n$	197	728

BLOCKCHAINS

BLOCKCHAINS

- ***Algorand.*** Featuring Silvio Micali, employs a pure Proof of Stake (PoS) consensus algorithm, ensuring swift transaction finality and efficient scalability through node selection using sortition.
- ***Diem.*** Formerly known as Libra (Facebook), employs a modified HotStuff consensus protocol to ensure deterministic resolution of the consensus problem, reduce communication overhead, and imposes a memory pool limit of 100 transactions per signer, with a sequence number in each transaction, akin to Ethereum.
- ***Ethereum Clique (Proof-of-Authority).*** Validates transactions in a round-robin fashion through approved validators with a minimum block interval.
- ***Quorum IBFT.*** A permissioned Ethereum variant maintained by ConsenSys for enterprise use, provides a web socket streaming API for accessing real-time blockchain data with a 1-second block period.
- ***Solana.*** A high-performance blockchain utilizing Proof-of-History combined with Proof-of-Stake for scalability and throughput, requires 30 confirmations for transaction finality and appends blocks every 400 milliseconds by replacing both Merkle Patricia Trie and ECDSA.

Blockchain	Consensus	VM	DApp	Block Finality (s)	Claimed TPS
Algorand	BA [41]	AVM	PyTeal [8]	3.3 [9]	7.5K [9]
Diem	HotStuff [82]	MoveVM	Move	100 [61]	60–1K [83]
Ethereum	Clique [70]	geth	Solidity	10–20 [4]	10–15 [67]
Quorum	IBFT [66]	geth	Solidity	2–15 [52]	0.7K–2.5K [10]
Solana	TowerBFT [80]	Sealevel	Solang	12 [42]	65K [11]

BLOCKCHAINS

- ***Algorand.*** Featuring Silvio Micali, employs a pure Proof of Stake (PoS) consensus algorithm, ensuring swift transaction finality and efficient scalability through node selection using sortition.
- ***Diem.*** Formerly known as Libra (Facebook), employs a modified HotStuff consensus protocol to ensure deterministic resolution of the consensus problem, reduce communication overhead, and imposes a memory pool limit of 100 transactions per signer, with a sequence number in each transaction, akin to Ethereum.
- ***Ethereum Clique (Proof-of-Authority).*** Validates transactions in a round-robin fashion through approved validators with a minimum block interval.
- ***Quorum IBFT.*** A permissioned Ethereum variant maintained by ConsenSys for enterprise use, provides a web socket streaming API for accessing real-time blockchain data with a 1-second block period.
- ***Solana.*** A high-performance blockchain utilizing Proof-of-History combined with Proof-of-Stake for scalability and throughput, requires 30 confirmations for transaction finality and appends blocks every 400 milliseconds by replacing both Merkle Patricia Trie and ECDSA.

Blockchain	Consensus	VM	DApp	Block Finality (s)	Claimed TPS
Algorand	BA [41]	AVM	PyTeal [8]	3.3 [9]	7.5K [9]
Diem	HotStuff [82]	MoveVM	Move	100 [61]	60–1K [83]
Ethereum	Clique [70]	geth	Solidity	10–20 [4]	10–15 [67]
Quorum	IBFT [66]	geth	Solidity	2–15 [52]	0.7K–2.5K [10]
Solana	TowerBFT [80]	Sealevel	Solang	12 [42]	65K [11]

BLOCKCHAINS

- ***Algorand.*** Featuring Silvio Micali, employs a pure Proof of Stake (PoS) consensus algorithm, ensuring swift transaction finality and efficient scalability through node selection using sortition.
- ***Diem.*** Formerly known as Libra (Facebook), employs a modified HotStuff consensus protocol to ensure deterministic resolution of the consensus problem, reduce communication overhead, and imposes a memory pool limit of 100 transactions per signer, with a sequence number in each transaction, akin to Ethereum.
- ***Ethereum Clique (Proof-of-Authority).*** Validates transactions in a round-robin fashion through approved validators with a minimum block interval.
- ***Quorum IBFT.*** A permissioned Ethereum variant maintained by ConsenSys for enterprise use, provides a web socket streaming API for accessing real-time blockchain data with a 1-second block period.
- ***Solana.*** A high-performance blockchain utilizing Proof-of-History combined with Proof-of-Stake for scalability and throughput, requires 30 confirmations for transaction finality and appends blocks every 400 milliseconds by replacing both Merkle Patricia Trie and ECDSA.

Blockchain	Consensus	VM	DApp	Block Finality (s)	Claimed TPS
Algorand	BA [41]	AVM	PyTeal [8]	3.3 [9]	7.5K [9]
Diem	HotStuff [82]	MoveVM	Move	100 [61]	60–1K [83]
Ethereum	Clique [70]	geth	Solidity	10–20 [4]	10–15 [67]
Quorum	IBFT [66]	geth	Solidity	2–15 [52]	0.7K–2.5K [10]
Solana	TowerBFT [80]	Sealevel	Solang	12 [42]	65K [11]

BLOCKCHAINS

- ***Algorand.*** Featuring Silvio Micali, employs a pure Proof of Stake (PoS) consensus algorithm, ensuring swift transaction finality and efficient scalability through node selection using sortition.
- ***Diem.*** Formerly known as Libra (Facebook), employs a modified HotStuff consensus protocol to ensure deterministic resolution of the consensus problem, reduce communication overhead, and imposes a memory pool limit of 100 transactions per signer, with a sequence number in each transaction, akin to Ethereum.
- ***Ethereum Clique (Proof-of-Authority).*** Validates transactions in a round-robin fashion through approved validators with a minimum block interval.
- ***Quorum IBFT.*** A permissioned Ethereum variant maintained by ConsenSys for enterprise use, provides a web socket streaming API for accessing real-time blockchain data with a 1-second block period.
- ***Solana.*** A high-performance blockchain utilizing Proof-of-History combined with Proof-of-Stake for scalability and throughput, requires 30 confirmations for transaction finality and appends blocks every 400 milliseconds by replacing both Merkle Patricia Trie and ECDSA.

Blockchain	Consensus	VM	DApp	Block Finality (s)	Claimed TPS
Algorand	BA [41]	AVM	PyTeal [8]	3.3 [9]	7.5K [9]
Diem	HotStuff [82]	MoveVM	Move	100 [61]	60–1K [83]
Ethereum	Clique [70]	geth	Solidity	10–20 [4]	10–15 [67]
Quorum	IBFT [66]	geth	Solidity	2–15 [52]	0.7K–2.5K [10]
Solana	TowerBFT [80]	Sealevel	Solang	12 [42]	65K [11]

BLOCKCHAINS

- ***Algorand.*** Featuring Silvio Micali, employs a pure Proof of Stake (PoS) consensus algorithm, ensuring swift transaction finality and efficient scalability through node selection using sortition.
- ***Diem.*** Formerly known as Libra (Facebook), employs a modified HotStuff consensus protocol to ensure deterministic resolution of the consensus problem, reduce communication overhead, and imposes a memory pool limit of 100 transactions per signer, with a sequence number in each transaction, akin to Ethereum.
- ***Ethereum Clique (Proof-of-Authority).*** Validates transactions in a round-robin fashion through approved validators with a minimum block interval.
- ***Quorum IBFT.*** A permissioned Ethereum variant maintained by ConsenSys for enterprise use, provides a web socket streaming API for accessing real-time blockchain data with a 1-second block period.
- ***Solana.*** A high-performance blockchain utilizing Proof-of-History combined with Proof-of-Stake for scalability and throughput, requires 30 confirmations for transaction finality and appends blocks every 400 milliseconds by replacing both Merkle Patricia Trie and ECDSA.

Blockchain	Consensus	VM	DApp	Block Finality (s)	Claimed TPS
Algorand	BA [41]	AVM	PyTeal [8]	3.3 [9]	7.5K [9]
Diem	HotStuff [82]	MoveVM	Move	100 [61]	60–1K [83]
Ethereum	Clique [70]	geth	Solidity	10–20 [4]	10–15 [67]
Quorum	IBFT [66]	geth	Solidity	2–15 [52]	0.7K–2.5K [10]
Solana	TowerBFT [80]	Sealevel	Solang	12 [42]	65K [11]

WORKLOADS

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***DDoS***. To test the robustness of the specific blockchain under high request scenarios.
- ***FIFA***. A customization implementation of the FIFA website workload during the 1998 soccer world cup, and implemented as a simple (but contended) counter smart contract with an add function.
- ***GAFAM***. Implemented as a decentralized exchange smart contract with functions to buy and check the availability of the stocks for Google, Apple, Facebook, Amazon and Microsoft.
- ***Gaming***. Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.
- ***PayPal and VISA***. To test their transactions capacity to a blockchain.

Workload	Type	Scenario	Duration (s)	Injected TPS
DDoS	Transfer Tx	Constant rate	120	10,000
FIFA	Smart contract	High sending rate	100	45,000
GAFAM	Smart contract	Burst	180	20,000 down to 100
Gaming	Smart contract	Intensive	276	13,000
PayPal	Transfer Tx	Constant rate	300	200
VISA	Transfer Tx	Constant rate	300	1.800

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***DDoS***. To test the robustness of the specific blockchain under high request scenarios.
- ***FIFA***. A customization implementation of the FIFA website workload during the 1998 soccer world cup, and implemented as a simple (but contended) counter smart contract with an add function.
- ***GAFAM***. Implemented as a decentralized exchange smart contract with functions to buy and check the availability of the stocks for Google, Apple, Facebook, Amazon and Microsoft.
- ***Gaming***. Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.
- ***PayPal and VISA***. To test their transactions capacity to a blockchain.

Workload	Type	Scenario	Duration (s)	Injected TPS
DDoS	Transfer Tx	Constant rate	120	10,000
FIFA	Smart contract	High sending rate	100	45,000
GAFAM	Smart contract	Burst	180	20,000 down to 100
Gaming	Smart contract	Intensive	276	13,000
PayPal	Transfer Tx	Constant rate	300	200
VISA	Transfer Tx	Constant rate	300	1.800

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***DDoS***. To test the robustness of the specific blockchain under high request scenarios.
- ***FIFA***. A customization implementation of the FIFA website workload during the 1998 soccer world cup, and implemented as a simple (but contended) counter smart contract with an add function.
- ***GAFAM***. Implemented as a decentralized exchange smart contract with functions to buy and check the availability of the stocks for Google, Apple, Facebook, Amazon and Microsoft.
- ***Gaming***. Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.
- ***PayPal and VISA***. To test their transactions capacity to a blockchain.

Workload	Type	Scenario	Duration (s)	Injected TPS
DDoS	Transfer Tx	Constant rate	120	10,000
FIFA	Smart contract	High sending rate	100	45,000
GAFAM	Smart contract	Burst	180	20,000 down to 100
Gaming	Smart contract	Intensive	276	13,000
PayPal	Transfer Tx	Constant rate	300	200
VISA	Transfer Tx	Constant rate	300	1.800

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***DDoS***. To test the robustness of the specific blockchain under high request scenarios.
- ***FIFA***. A customization implementation of the FIFA website workload during the 1998 soccer world cup, and implemented as a simple (but contended) counter smart contract with an add function.
- ***GAFAM***. Implemented as a decentralized exchange smart contract with functions to buy and check the availability of the stocks for Google, Apple, Facebook, Amazon and Microsoft.
- ***Gaming***. Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.
- ***PayPal and VISA***. To test their transactions capacity to a blockchain.

Workload	Type	Scenario	Duration (s)	Injected TPS
DDoS	Transfer Tx	Constant rate	120	10,000
FIFA	Smart contract	High sending rate	100	45,000
GAFAM	Smart contract	Burst	180	20,000 down to 100
Gaming	Smart contract	Intensive	276	13,000
PayPal	Transfer Tx	Constant rate	300	200
VISA	Transfer Tx	Constant rate	300	1.800

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***DDoS***. To test the robustness of the specific blockchain under high request scenarios.
- ***FIFA***. A customization implementation of the FIFA website workload during the 1998 soccer world cup, and implemented as a simple (but contended) counter smart contract with an add function.
- ***GAFAM***. Implemented as a decentralized exchange smart contract with functions to buy and check the availability of the stocks for Google, Apple, Facebook, Amazon and Microsoft.
- ***Gaming***. Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.
- ***PayPal and VISA***. To test their transactions capacity to a blockchain.

Workload	Type	Scenario	Duration (s)	Injected TPS
DDoS	Transfer Tx	Constant rate	120	10,000
FIFA	Smart contract	High sending rate	100	45,000
GAFAM	Smart contract	Burst	180	20,000 down to 100
Gaming	Smart contract	Intensive	276	13,000
PayPal	Transfer Tx	Constant rate	300	200
VISA	Transfer Tx	Constant rate	300	1.800

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***DDoS***. To test the robustness of the specific blockchain under high request scenarios.
- ***FIFA***. A customization implementation of the FIFA website workload during the 1998 soccer world cup, and implemented as a simple (but contended) counter smart contract with an add function.
- ***GAFAM***. Implemented as a decentralized exchange smart contract with functions to buy and check the availability of the stocks for Google, Apple, Facebook, Amazon and Microsoft.
- ***Gaming***. Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.
- ***PayPal and VISA***. To test their transactions capacity to a blockchain.

Workload	Type	Scenario	Duration (s)	Injected TPS
DDoS	Transfer Tx	Constant rate	120	10,000
FIFA	Smart contract	High sending rate	100	45,000
GAFAM	Smart contract	Burst	180	20,000 down to 100
Gaming	Smart contract	Intensive	276	13,000
PayPal	Transfer Tx	Constant rate	300	200
VISA	Transfer Tx	Constant rate	300	1.800

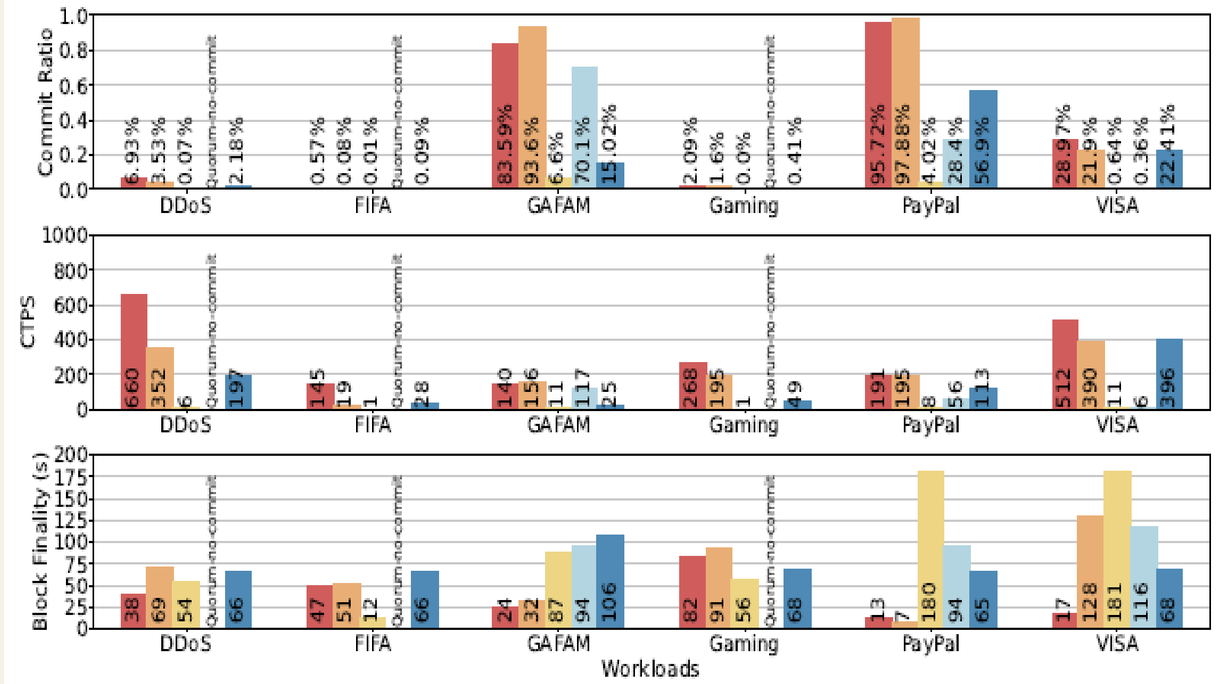
FINDINGS

RQ1 Torus allows generally for better results with more demanding workloads. We achieve the best overall results, across the five tested blockchains, atop full mesh and hypercube, due to their high degree and link capacity.

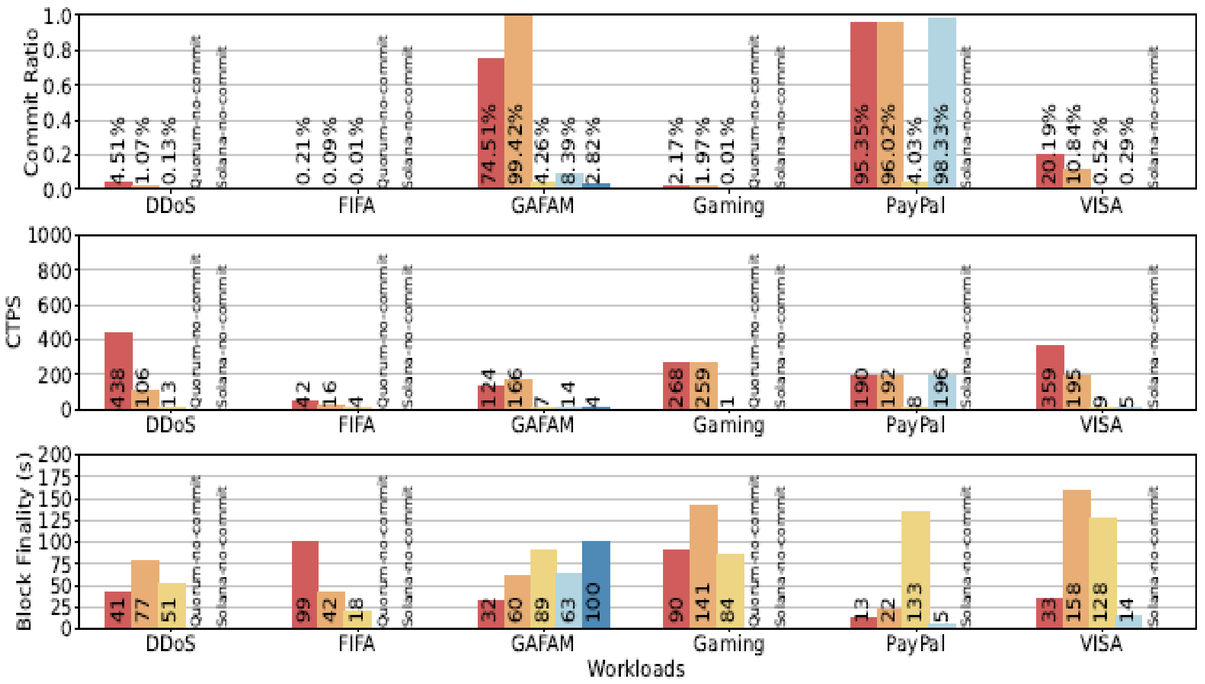
RQ2 Our results demonstrate that Diem and Solana are the most affected by packet loss, followed by Quorum and Algorand. Classical consensus mechanism, as in Quorum, are less reactive to increasing latencies. Despite consistently exhibiting lower CTPS compared to the others, Ethereum is the least affected by these dynamic network events.

RESULTS #1

Benchmark on a fat-tree topology with different blockchain network size



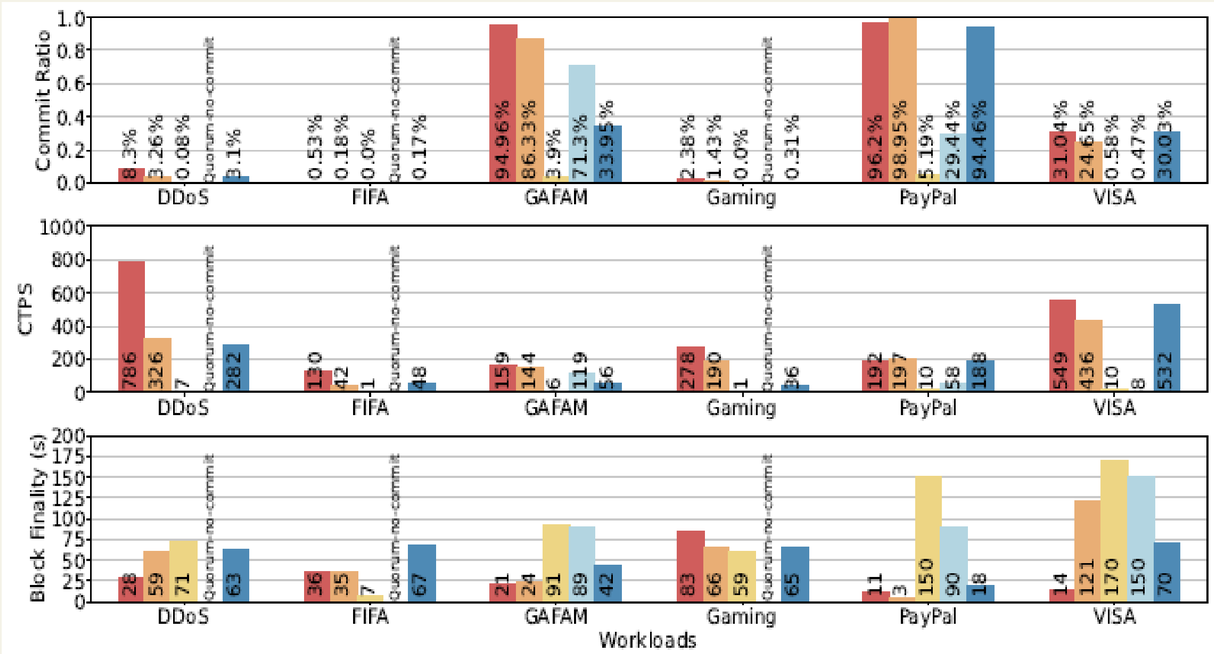
(a) Fat-tree, one node/region.



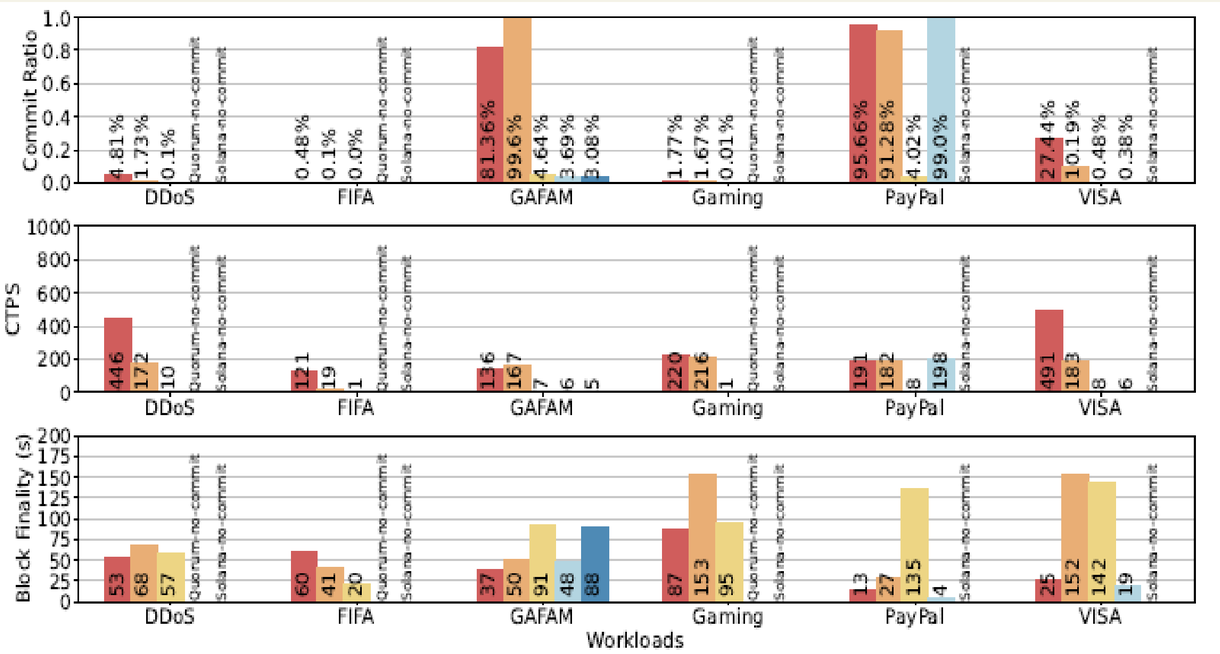
(b) Fat-tree, four nodes/region.

RESULTS #2

Benchmark on a full mesh topology with different blockchain network size



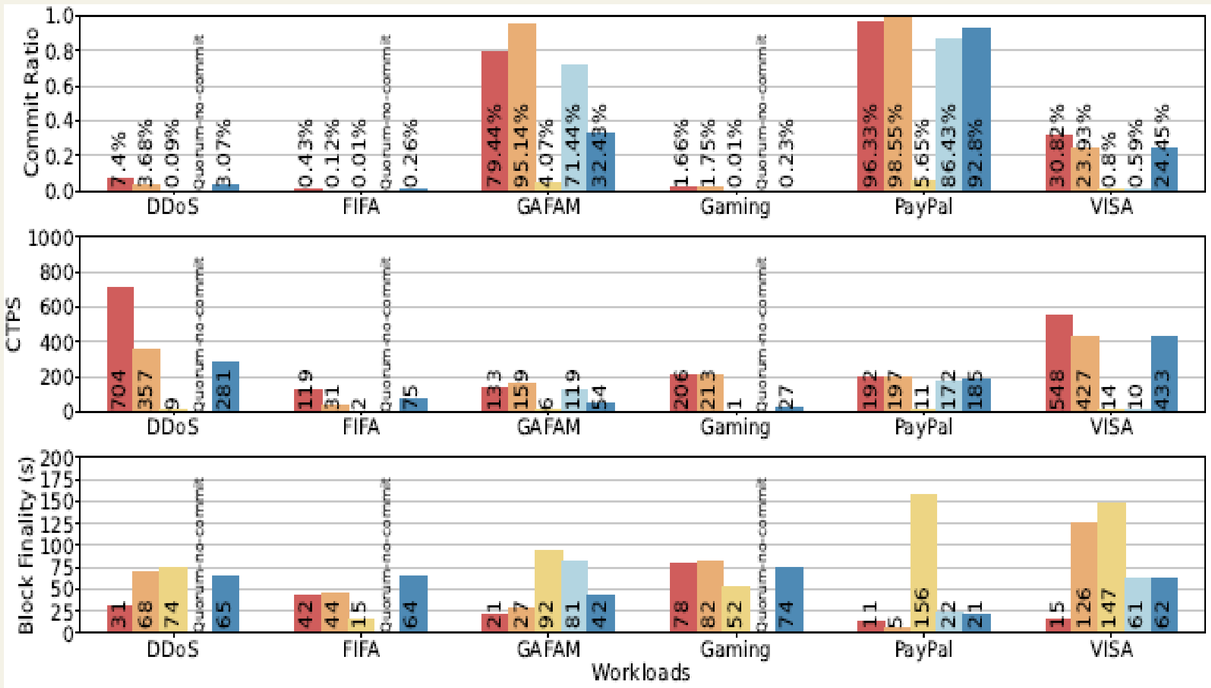
(c) Full mesh, one node/region.



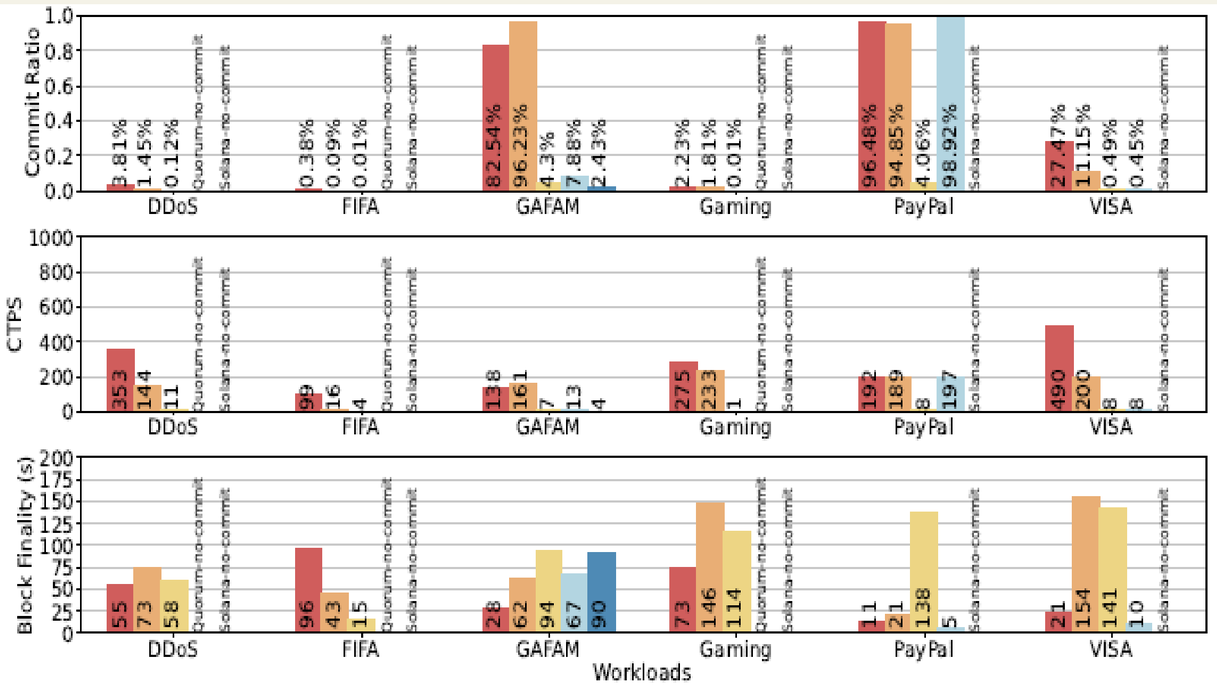
(d) Full mesh, four nodes/region.

RESULTS #3

Benchmark on a hypercube topology with different blockchain network size



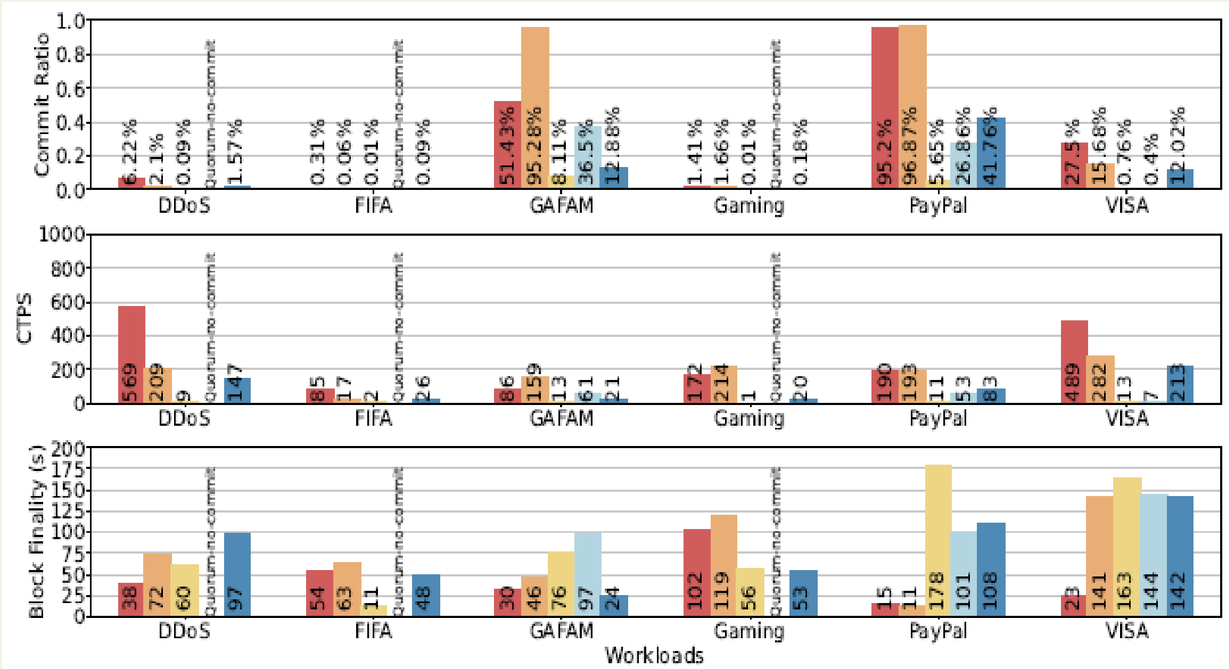
(e) Hypercube, one node/region.



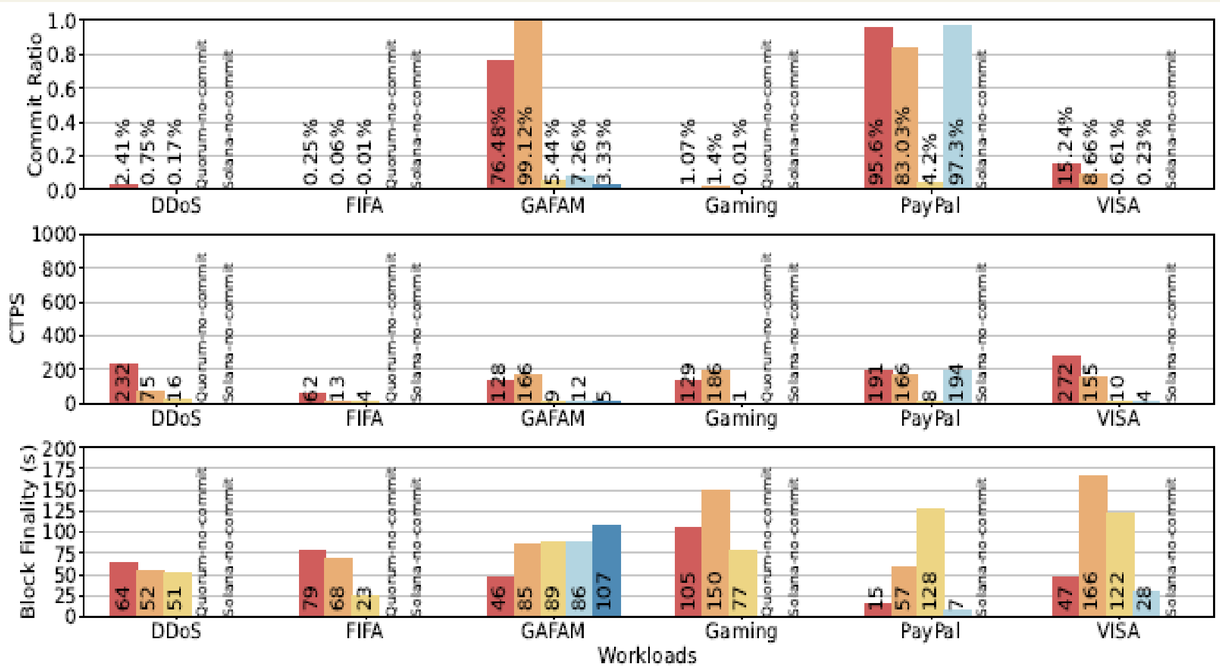
(f) Hypercube, four nodes/region.

RESULTS #4

Benchmark on a scale-free topology with different blockchain network size



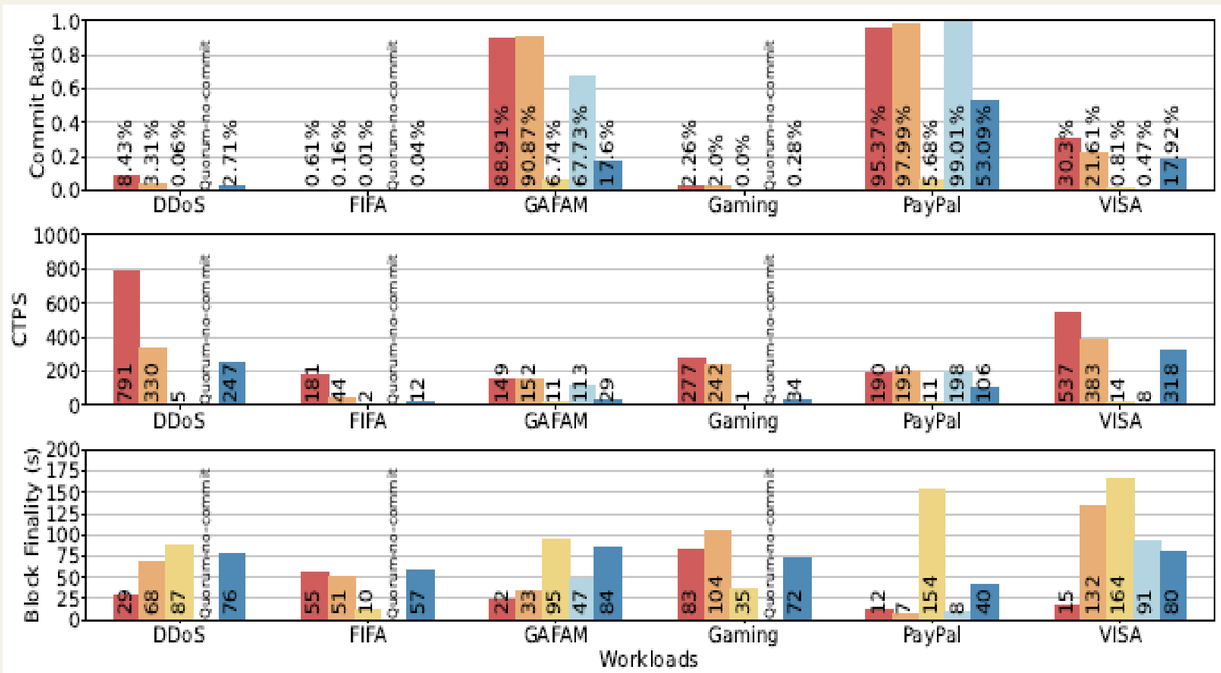
(g) Scale-free, one node/region.



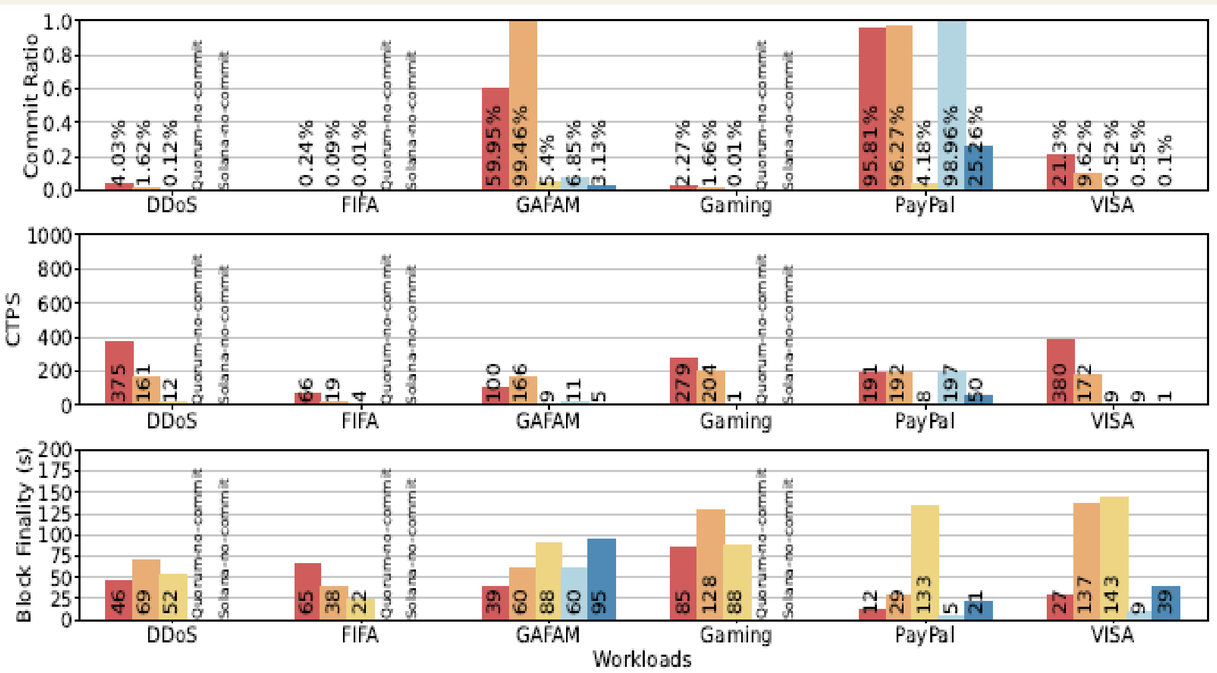
(h) Scale-free, four nodes/region.

RESULTS #5

Benchmark on a torus topology with different blockchain network size



(i) Torus, one node/region.



(j) Torus, four nodes/region.

RESULTS #6

Network load based on the different workload executed on the topologies set

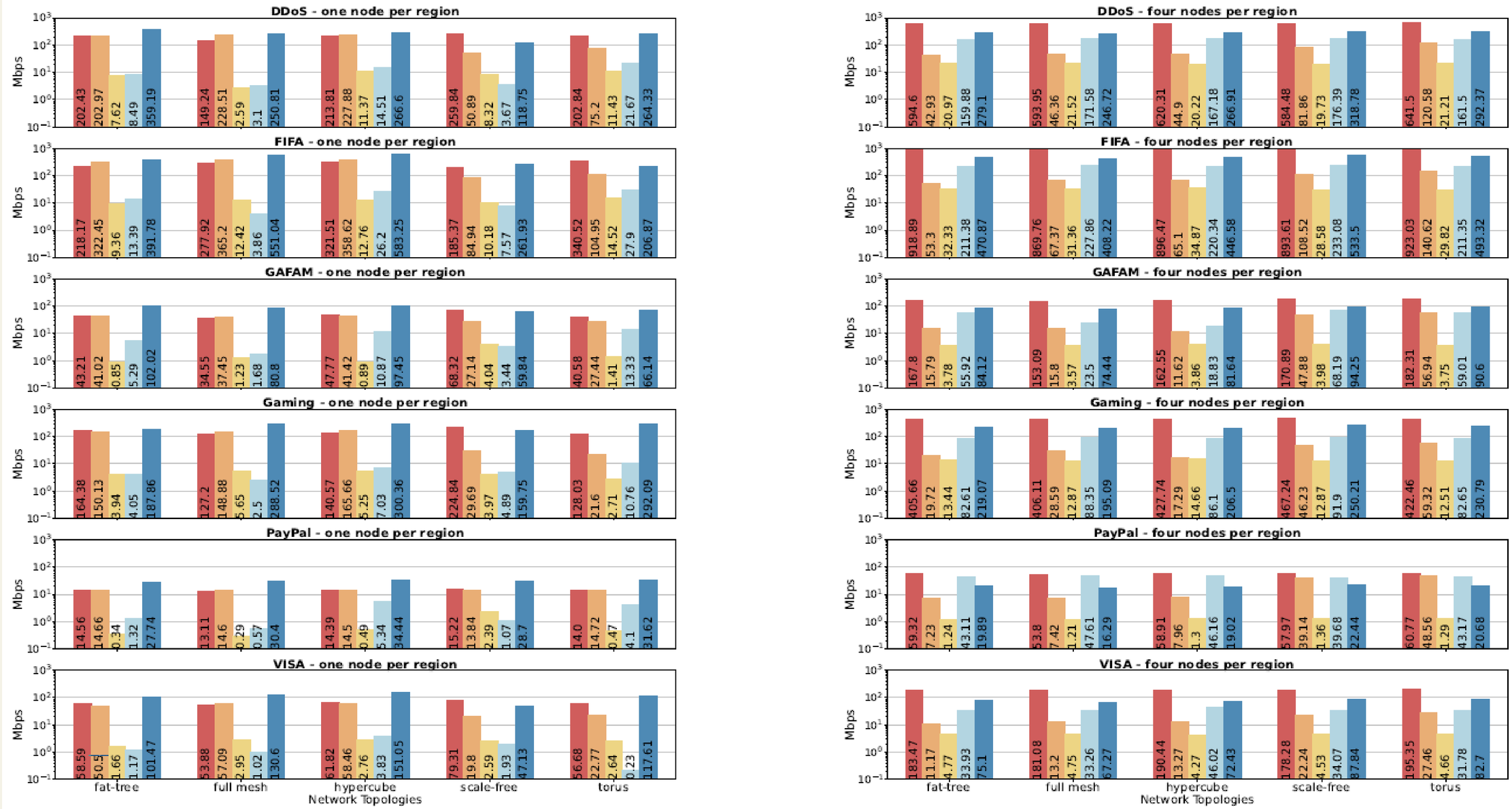
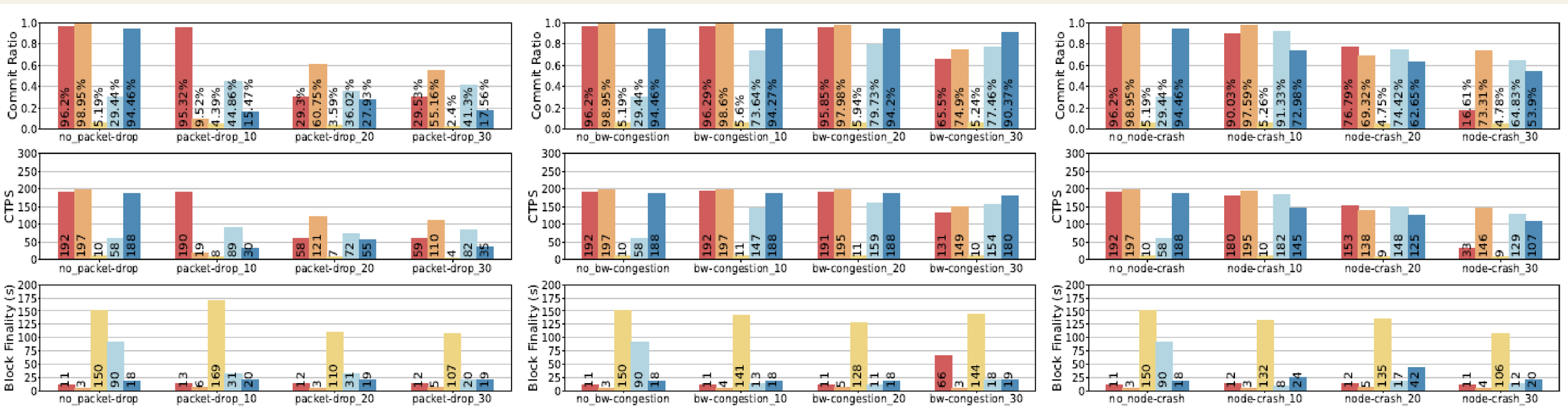


Figure 7: Network load (Mbps) during the workloads execution (Algorand, Diem, Ethereum, Quorum, Solana).

RESULTS #7

Benchmark on a full mesh topology with different packet drop percentage, link bandwidth congestion and node crash during the workload execution

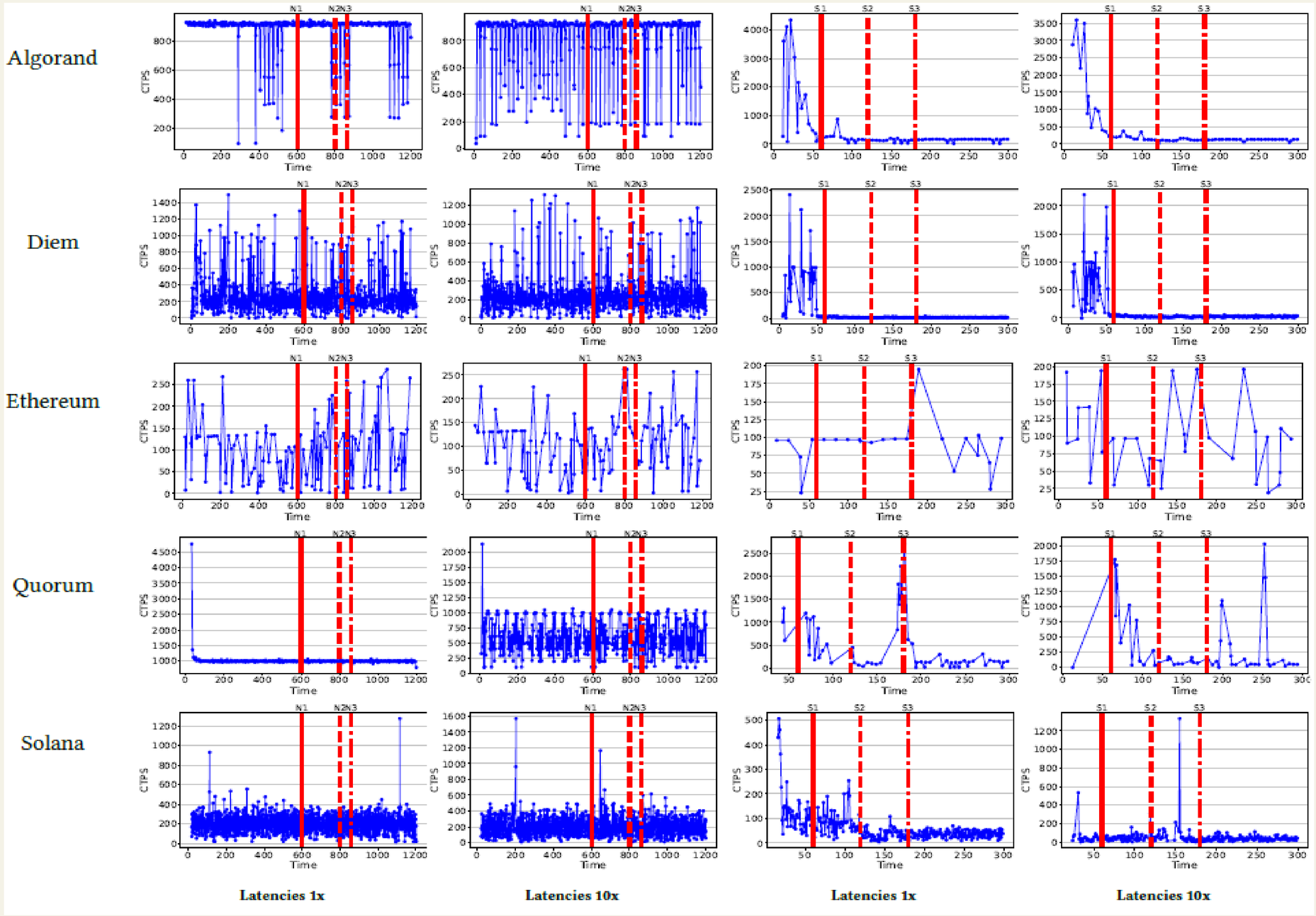


(a) Packet drop. (b) Link bandwidth congestion. (c) Node crash.

Figure 8: Network dynamics, one node per region, full mesh topology (Algorand, Diem, Ethereum, Quorum, Solana).

RESULTS #8

Benchmark on a full mesh topology with increasing latencies during the PayPal (left) and GAFAM (right) workloads execution



CONCLUSIONS & FUTURE WORK

CONCLUSIONS & FUTURE WORK

- Lilith enables reproducible evaluation of blockchain performance across realistic, geo-distributed network topologies.

- Collect and utilize real blockchain data (from specific periods)

- Use temporal data to simulate dynamic network events

- Implement and compare other (potentially new) blockchain protocol

CONCLUSIONS & FUTURE WORK

- Lilith enables reproducible evaluation of blockchain performance across realistic, geo-distributed network topologies.

- Collect and utilize real blockchain data (from specific periods)

- Use temporal data to simulate dynamic network events

- Implement and compare other (potentially new) blockchain protocol

CONCLUSIONS & FUTURE WORK

- Lilith enables reproducible evaluation of blockchain performance across realistic, geo-distributed network topologies.

- Collect and utilize real blockchain data (from specific periods)

- Use temporal data to simulate dynamic network events

- Implement and compare other (potentially new) blockchain protocol

CONCLUSIONS & FUTURE WORK

- Lilith enables reproducible evaluation of blockchain performance across realistic, geo-distributed network topologies.

- Collect and utilize real blockchain data (from specific periods)

- Use temporal data to simulate dynamic network events

- Implement and compare other (potentially new) blockchain protocol

Thanks for your attention

References

Vincent Gramoli, Rachid Guerraoui, Andrei Lebedev, Chris Natoli, and Gauthier Voron. 2023. *Diablo: A Benchmark Suite for Blockchains*. In *Proceedings of the Eighteenth European Conference on Computer Systems (EuroSys '23)*. Association for Computing Machinery, New York, NY, USA, 540–556. https://doi.org/10.1145/3552326.3567482

P. Gouveia, J. Neves, C. Segarra, L. Liechti, S. Issa, V. Schiavoni, and M. Matos. 2020. *Kollaps: Decentralized and Dynamic Topology Emulation. In Proceedings of the Fifteenth European Conference on Computer Systems (EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 23, 16 pages. https://doi.org/10.1145/3342195.3387540