Understanding the Effects of Network Topologies on Blockchains Performance and Efficiency

NiRvAna Annual Workshop
June 6-8, 2024, Udine, Italy

1506 UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO

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

1506 UNIVERSITÀ DEGLI STUDI DI URBINO CARLO BO

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

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

unine Université de Neuchâtel

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

Featured by NiRvAna project

CONTEXT #1

CONTEXT #1

Blockchain scenario

CONTEXT #1

Blockchain scenario

- Blockchain system as a distributed ledger between (mutually untrusted) nodes.

CONTEXT #1

Blockchain scenario

- Blockchain system as a distributed ledger between (mutually untrusted) nodes.
- Widespread adoption and proliferation of blockchain technology (**e.g., Algorand, Ethereum, Solana, Quorum**).

CONTEXT #1

Blockchain scenario

- Blockchain system as a distributed ledger between (mutually untrusted) nodes.
- Widespread adoption and proliferation of blockchain technology **(e.g., Algorand, Ethereum, Solana, Quorum)**.
- Flexibility and adaptability of blockchain solutions to diverse industries **(e.g., Finance, Supply Chain, Healtcare)**.

CONTEXT #1

Blockchain scenario

- Blockchain system as a distributed ledger between (mutually untrusted) nodes.
- Widespread adoption and proliferation of blockchain technology **(e.g., Algorand, Ethereum, Solana, 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 system as a distributed ledger between (mutually untrusted) nodes.
- Widespread adoption and proliferation of blockchain technology **(e.g., Algorand, Ethereum, Solana, 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 5. Comparison of several blockchain benchmarking tools. Legend: ◌ = not found; ○ = no-match; ● = match; ◐ = partial-no match; ◑ = partial-match; SC = Smart Contract; N = Native.

Tool	Year	Repo.	Last Commit	Language	Network settings	Distributed	Arch.	Repeatability	Versatility	Observability	Portability	Ease of presentation	Realism	Reproducibility
Shadow-Bitcoin [92]	2015	[118]	2020	.py	◌	○	Sim.	◌	○	○	○(Bitcoin)	○	○	◌
HIVE [18]	2016	[18]	2024	Go	◌	●	Sim.	◌	○	○	○(Ethereum)	○	○	◌
Simcoin [115]	2016	[115]	2018	.py	◌	○	Sim.	◌	○	●	○(Bitcoin)	○	○	◌
Bitcoin-Simulator [69]	2016	[1]	2016	C++	◑	○	Sim.	◌	○	○	●(Bitcoin-like)	○	○	◌
BlockBench [52]	2017	[103]	2022	C++,Go	◌	◌	◌	◌	◑	○	●(Permissioned)	○	●(SC)	◌
CIDDS [82]	2018	[4]	2021	.py	◑	◌	Sim.	◌	○	○	●(DAG-based)	○	◐	◌
Minichain [128]	2019	[129]	2019	.py	◑	○	Emu.	◌	○	○	●	○	○	◌
BlockLite [125]	2019	[76]	2019	Java	◌	○	Emu.	◌	○	○	●(PoW-public)	○	◐	◌
BlockSim-f [59]	2019	[34]	2020	.py	◑	○	Sim.	◌	○	○	●	○	○	◌
SimBlock [29]	2019	[54]	2021	Java	◑	◌	Sim.	◌	○	○	●	○	●(N)	◌
BlockSim-m [25]	2020	[89]	2022	.py	◑	○	Sim.	◌	○	○	●	○	●(N,◌)	◌
Core-Bit-Netw-Sim. [26]	2020	[101]	2020	.py	◑	◌	Sim.	◌	○	○	●	○	○	◌
BBB [104]	2020	[75]	2021	.py	●	◐	Emu.	◌	○	○	●(Permissioned)	○	◐	◌
SIMBA [60]	2020	[102]	2020	.py	◑	◌	Sim.	◌	○	○	●	○	◐	◌
BCTMark [113]	2020	[112]	2021	.py	◑	●	Emu.	◌	○	●	●	○	●(◌)	◌
BlockPerf [106]	2021	[49]	2021	C/C++,.py	◐	●	Both	◌	○	○	●(Bitcoin)	○	◌	◌
DLPS [62]	2021	[53]	2021	.py	◑	◌	◌	◌	○	●	●	○	●(◌)	◌
Gromit [98]	2022	[74]	2022	.py	◑	●	◌	◌	○	●	●	○	●(◌,SC)	◌
Diablo v2 [73]	2022	[83]	2022	Go,.py,Perl	○	●	Emu.	●	●	●	●	●	●(N,SC)	●
JABS [130]	2023	[77]	2023	Java	◑	◌	Sim.	◌	○	○	●	○	●(N,◌)	◌
TangleSim [88]	2023	[79]	2023	Go,.py	◌	◌	Sim.	◌	●	○	●(DAG-based)	○	○	◌
COCONUT [70]	2023	[61]	2024	Jinja,Java	◌	◌	Emu.	●	◑	○	●	◑	●(N,SC)	●
Lilith	2024	[58]	2024	.sh,.py	●	●	Emu.	●	●	●	●	●	●(N,SC)	●

CONTEXT #2

Blockchain benchmark tools comparison

Table 5. Comparison of several blockchain benchmarking tools. Legend: ◌ = not found; ○ = no-match; ● = match; ◐ = partial-no match; ◑ = partial-match; SC = Smart Contract; N = Native.

Tool	Year	Repo.	Last Commit	Language	Network settings	Distributed	Arch.	Repeatability	Versatility	Observability	Portability	Ease of presentation	Realism	Reproducibility
Shadow-Bitcoin [92]	2015	[118]	2020	.py	◌	○	Sim.	◌	○	○	○(Bitcoin)	○	○	◌
HIVE [18]	2016	[18]	2024	Go	◌	●	Sim.	◌	○	○	○(Ethereum)	○	○	◌
Simcoin [115]	2016	[115]	2018	.py	◌	○	Sim.	◌	○	●	○(Bitcoin)	○	○	◌
Bitcoin-Simulator [69]	2016	[1]	2016	C++	◑	○	Sim.	◌	○	○	●(Bitcoin-like)	○	○	◌
BlockBench [52]	2017	[103]	2022	C++,Go	◌	◌	◌	◌	◑	○	●(Permissioned)	○	●(SC)	◌
CIDDS [82]	2018	[4]	2021	.py	◑	◌	Sim.	◌	○	○	●(DAG-based)	○	◐	◌
Minichain [128]	2019	[129]	2019	.py	◑	○	Emu.	◌	○	○	●	○	○	◌
BlockLite [125]	2019	[76]	2019	Java	◌	○	Emu.	◌	○	○	●(PoW-public)	○	◐	◌
BlockSim-f [59]	2019	[34]	2020	.py	◑	○	Sim.	◌	○	○	●	○	○	◌
SimBlock [29]	2019	[54]	2021	Java	◑	◌	Sim.	◌	○	○	●	○	●(N)	◌
BlockSim-m [25]	2020	[89]	2022	.py	◑	○	Sim.	◌	○	○	●	○	●(N,◌)	◌
Core-Bit-Netw-Sim. [26]	2020	[101]	2020	.py	◑	◌	Sim.	◌	○	○	●	○	○	◌
BBB [104]	2020	[75]	2021	.py	●	◐	Emu.	◌	○	○	●(Permissioned)	○	◐	◌
SIMBA [60]	2020	[102]	2020	.py	◑	◌	Sim.	◌	○	○	●	○	◐	◌
BCTMark [113]	2020	[112]	2021	.py	◑	●	Emu.	◌	○	●	●	○	●(◌)	◌
BlockPerf [106]	2021	[49]	2021	C/C++,.py	◐	●	Both	◌	○	○	●(Bitcoin)	○	◌	◌
DLPS [62]	2021	[53]	2021	.py	◑	◌	◌	◌	○	●	●	○	●(◌)	◌
Gromit [98]	2022	[74]	2022	.py	◑	●	◌	◌	○	●	●	○	●(◌,SC)	◌
Diablo v2 [73]	2022	[83]	2022	Go,.py,Perl	○	●	Emu.	●	●	●	●	●	●(N,SC)	●
JABS [130]	2023	[77]	2023	Java	◑	◌	Sim.	◌	○	○	●	○	●(N,◌)	◌
TangleSim [88]	2023	[79]	2023	Go,.py	◌	◌	Sim.	◌	●	○	●(DAG-based)	○	○	◌
COCONUT [70]	2023	[61]	2024	Jinja,Java	◌	◌	Emu.	●	◑	○	●	◑	●(N,SC)	●
Lilith	2024	[58]	2024	.sh,.py	●	●	Emu.	●	●	●	●	●	●(N,SC)	●

- Diablo 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

- ***(C1)*** *Framework-specific hurdles.*

PROBLEMS

Blockchain benchmark challenges

- ***(C1)*** *Framework-specific hurdles.*
- ***(C2)*** *Lack of standardized benchmarks.*

PROBLEMS

Blockchain benchmark challenges

- ***(C1)*** *Framework-specific hurdles.*
- ***(C2)*** *Lack of standardized benchmarks.*
- ***(C3)*** *Infrastructure setup complexity.*

PROBLEMS

Blockchain benchmark challenges

- ***(C1)*** *Framework-specific hurdles.*
- ***(C2)*** *Lack of standardized benchmarks.*
- ***(C3)*** *Infrastructure setup complexity.*
- ***(C4)*** *Lack of realistic workloads.*

PROBLEMS

Blockchain benchmark challenges

- ***(C1)*** *Framework-specific hurdles.*
- ***(C2)*** *Lack of standardized benchmarks.*
- ***(C3)*** *Infrastructure setup complexity.*
- ***(C4)*** *Lack of realistic workloads.*
- ***(C5)*** *Resource management and transaction handling.*

PROBLEMS

Blockchain benchmark challenges

- ***(C1)*** *Framework-specific hurdles.*
- ***(C2)*** *Lack of standardized benchmarks.*
- ***(C3)*** *Infrastructure setup complexity.*
- ***(C4)*** *Lack of realistic workloads.*
- ***(C5)*** *Resource management and transaction handling.*
- ***(C6)*** *Steep learning curve.*

CLOUD LIMITATIONS #1

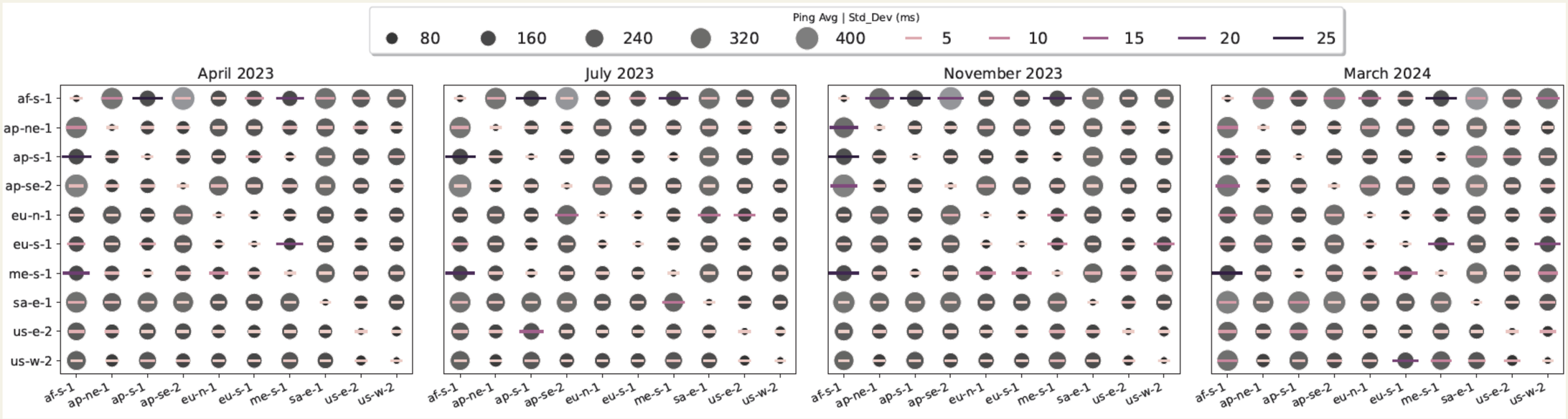
CLOUD LIMITATIONS #1

Cloud-based vs cluster benchmark

CLOUD LIMITATIONS #1

Cloud-based vs cluster benchmark

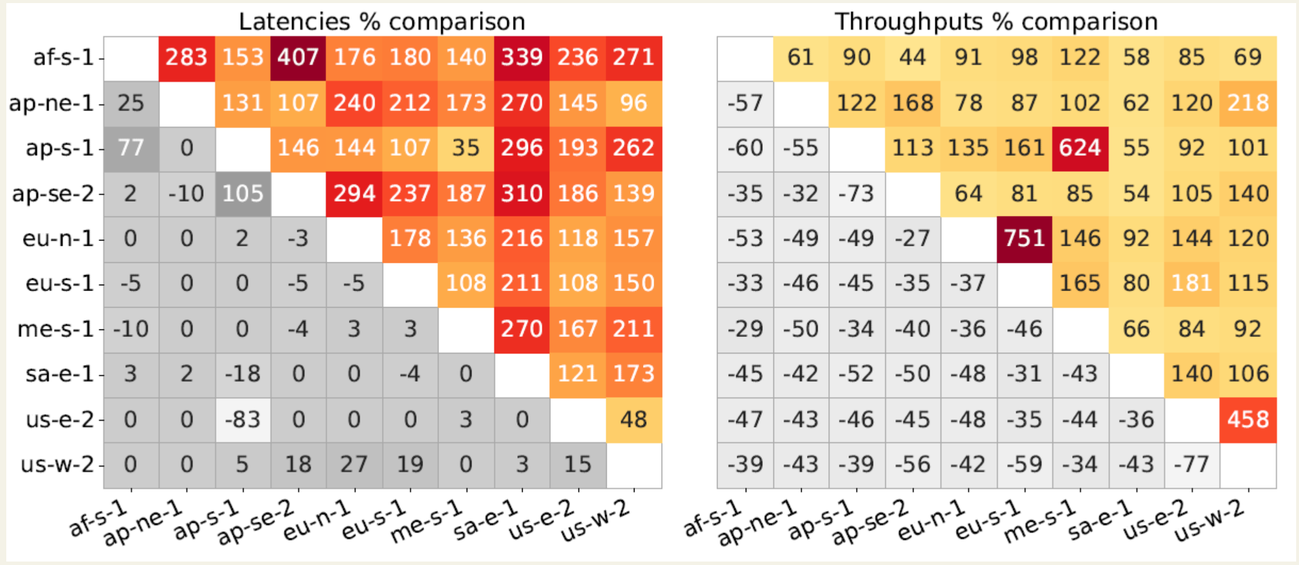
- Network links between the data centers changes throughout the day (e.g., infrastructure updates, contention on links, etc.).

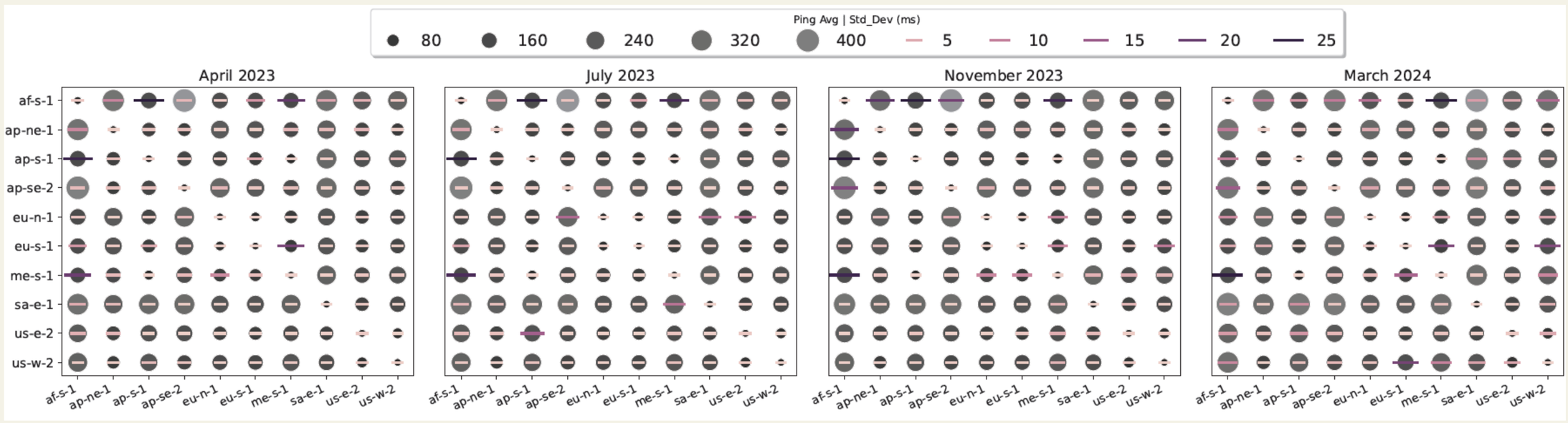


CLOUD LIMITATIONS #1

Cloud-based vs cluster benchmark

- 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).



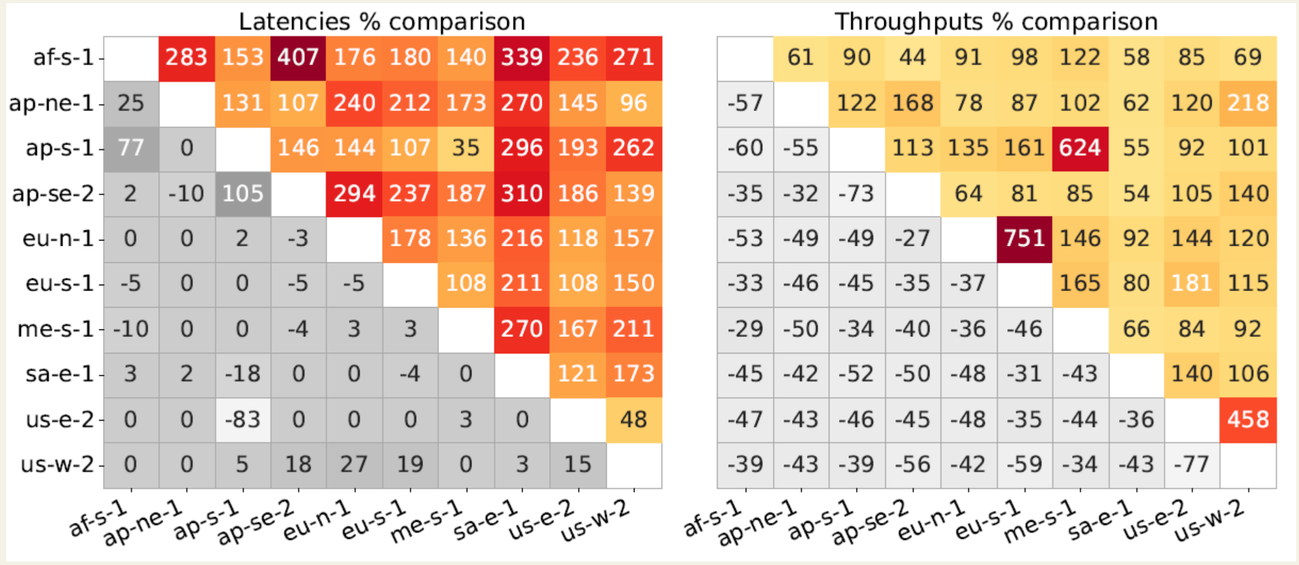


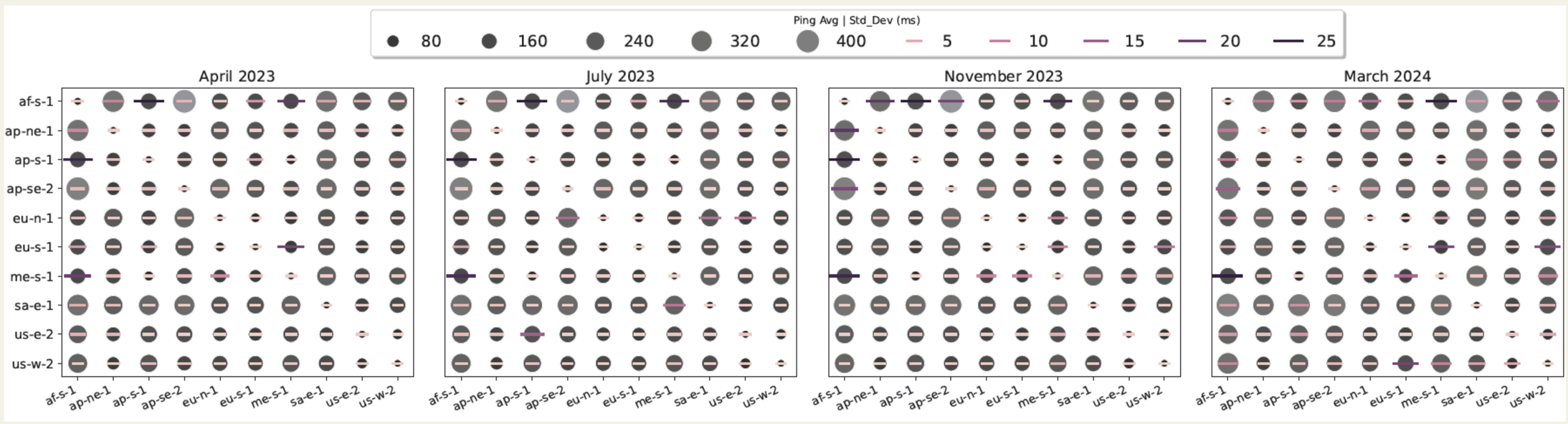
CLOUD LIMITATIONS #1

Cloud-based vs cluster benchmark

- 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.

	Latency	Costs	Network flexibility
Cloud	Physical	High	Not allowed
Cluster	None	(almost) None	Allowed





CLOUD LIMITATIONS #2

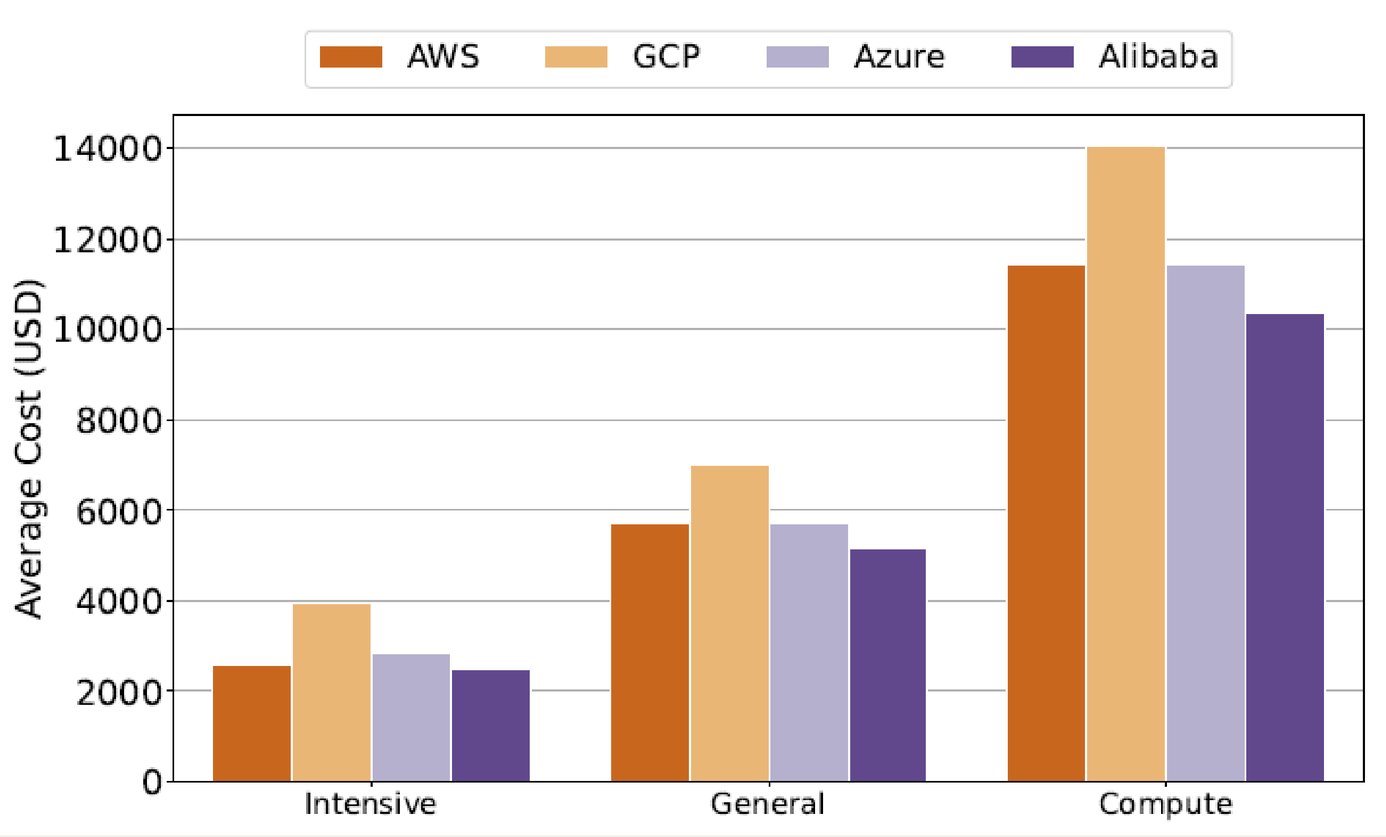
CLOUD LIMITATIONS #2

Cloud challenges

CLOUD LIMITATIONS #2

Cloud challenges

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

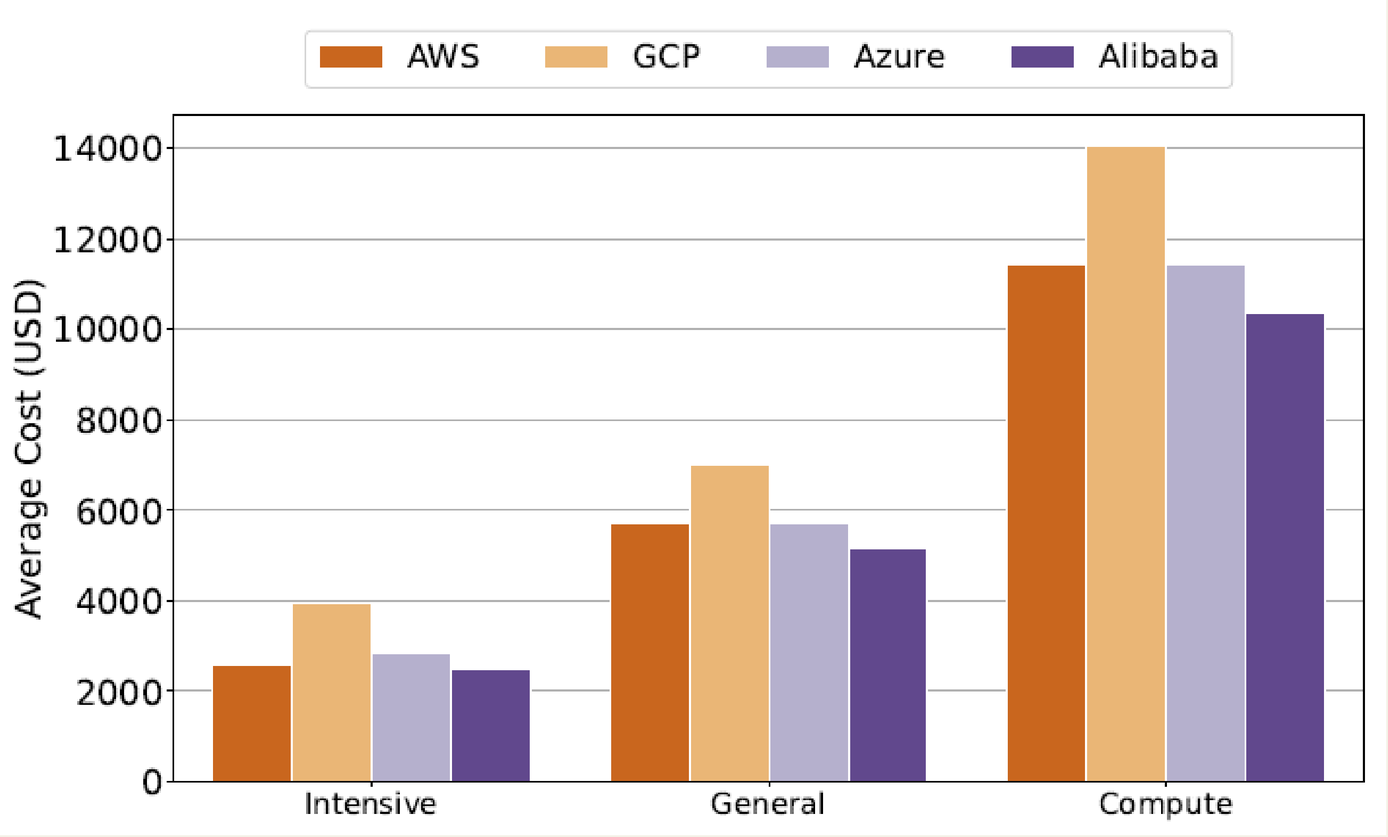


Intensive, 30/60 vCPU, 64/120 GiB RAM, 10 nodes
General, 4 vCPU, 8 GiB RAM, 200 nodes
Compute, 8 vCPU, 16 GiB RAM, 200 nodes

CLOUD LIMITATIONS #2

Cloud challenges

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

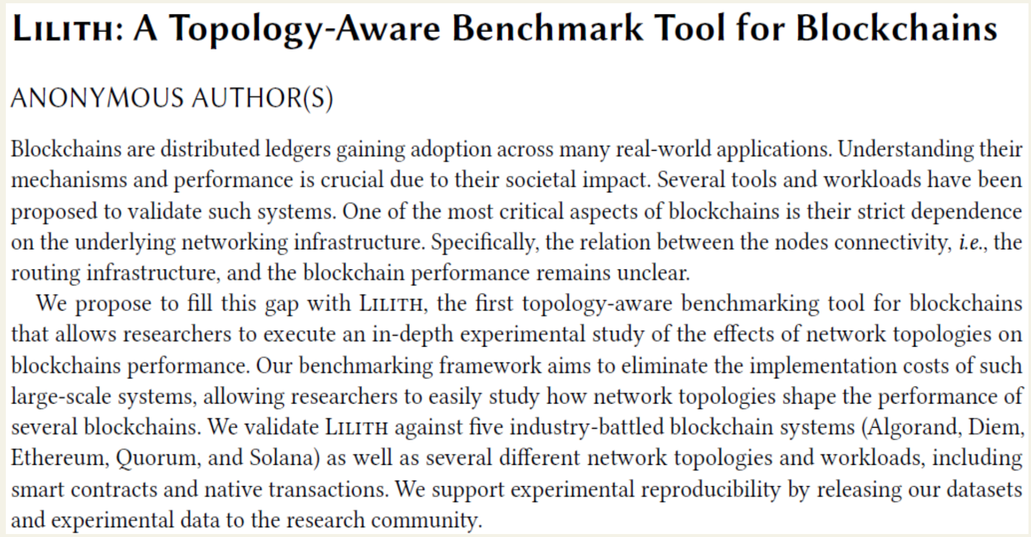


Intensive, 30/60 vCPU, 64/120 GiB RAM, 10 nodes
General, 4 vCPU, 8 GiB RAM, 200 nodes
Compute, 8 vCPU, 16 GiB RAM, 200 nodes

CONTRIBUTION

CONTRIBUTION

LILITH: A Topology-Aware Benchmark Tool for Blockchains



Lilith: A Topology-Aware Benchmark Tool for Blockchains

ANONYMOUS AUTHOR(S)

Blockchains are distributed ledgers gaining adoption across many real-world applications. Understanding their mechanisms and performance is crucial due to their societal impact. Several tools and workloads have been proposed to validate such systems. One of the most critical aspects of blockchains is their strict dependence on the underlying networking infrastructure. Specifically, the relation between the nodes connectivity, *i.e.*, the routing infrastructure, and the blockchain performance remains unclear.

We propose to fill this gap with Lilith, the first topology-aware benchmarking tool for blockchains that allows researchers to execute an in-depth experimental study of the effects of network topologies on blockchains performance. Our benchmarking framework aims to eliminate the implementation costs of such large-scale systems, allowing researchers to easily study how network topologies shape the performance of several blockchains. We validate Lilith against five industry-battled blockchain systems (Algorand, Diem, Ethereum, Quorum, and Solana) as well as several different network topologies and workloads, including smart contracts and native transactions. We support experimental reproducibility by releasing our datasets and experimental data to the research community.

Submitted…

CONTRIBUTION

LILITH: A Topology-Aware Benchmark Tool for Blockchains

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

Lilith: A Topology-Aware Benchmark Tool for Blockchains

ANONYMOUS AUTHOR(S)

Blockchains are distributed ledgers gaining adoption across many real-world applications. Understanding their mechanisms and performance is crucial due to their societal impact. Several tools and workloads have been proposed to validate such systems. One of the most critical aspects of blockchains is their strict dependence on the underlying networking infrastructure. Specifically, the relation between the nodes connectivity, *i.e.*, the routing infrastructure, and the blockchain performance remains unclear.

We propose to fill this gap with Lilith, the first topology-aware benchmarking tool for blockchains that allows researchers to execute an in-depth experimental study of the effects of network topologies on blockchains performance. Our benchmarking framework aims to eliminate the implementation costs of such large-scale systems, allowing researchers to easily study how network topologies shape the performance of several blockchains. We validate Lilith against five industry-battled blockchain systems (Algorand, Diem, Ethereum, Quorum, and Solana) as well as several different network topologies and workloads, including smart contracts and native transactions. We support experimental reproducibility by releasing our datasets and experimental data to the research community.

Submitted… 

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.

Lilith: A Topology-Aware Benchmark Tool for Blockchains

ANONYMOUS AUTHOR(S)

Blockchains are distributed ledgers gaining adoption across many real-world applications. Understanding their mechanisms and performance is crucial due to their societal impact. Several tools and workloads have been proposed to validate such systems. One of the most critical aspects of blockchains is their strict dependence on the underlying networking infrastructure. Specifically, the relation between the nodes connectivity, *i.e.*, the routing infrastructure, and the blockchain performance remains unclear.

We propose to fill this gap with Lilith, the first topology-aware benchmarking tool for blockchains that allows researchers to execute an in-depth experimental study of the effects of network topologies on blockchains performance. Our benchmarking framework aims to eliminate the implementation costs of such large-scale systems, allowing researchers to easily study how network topologies shape the performance of several blockchains. We validate Lilith against five industry-battled blockchain systems (Algorand, Diem, Ethereum, Quorum, and Solana) as well as several different network topologies and workloads, including smart contracts and native transactions. We support experimental reproducibility by releasing our datasets and experimental data to the research community.

Submitted…

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.

Lilith: A Topology-Aware Benchmark Tool for Blockchains

ANONYMOUS AUTHOR(S)

Blockchains are distributed ledgers gaining adoption across many real-world applications. Understanding their mechanisms and performance is crucial due to their societal impact. Several tools and workloads have been proposed to validate such systems. One of the most critical aspects of blockchains is their strict dependence on the underlying networking infrastructure. Specifically, the relation between the nodes connectivity, *i.e.*, the routing infrastructure, and the blockchain performance remains unclear.

We propose to fill this gap with Lilith, the first topology-aware benchmarking tool for blockchains that allows researchers to execute an in-depth experimental study of the effects of network topologies on blockchains performance. Our benchmarking framework aims to eliminate the implementation costs of such large-scale systems, allowing researchers to easily study how network topologies shape the performance of several blockchains. We validate Lilith against five industry-battled blockchain systems (Algorand, Diem, Ethereum, Quorum, and Solana) as well as several different network topologies and workloads, including smart contracts and native transactions. We support experimental reproducibility by releasing our datasets and experimental data to the research community.

Submitted…

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.

Lilith: A Topology-Aware Benchmark Tool for Blockchains

ANONYMOUS AUTHOR(S)

Blockchains are distributed ledgers gaining adoption across many real-world applications. Understanding their mechanisms and performance is crucial due to their societal impact. Several tools and workloads have been proposed to validate such systems. One of the most critical aspects of blockchains is their strict dependence on the underlying networking infrastructure. Specifically, the relation between the nodes connectivity, *i.e.*, the routing infrastructure, and the blockchain performance remains unclear.

We propose to fill this gap with Lilith, the first topology-aware benchmarking tool for blockchains that allows researchers to execute an in-depth experimental study of the effects of network topologies on blockchains performance. Our benchmarking framework aims to eliminate the implementation costs of such large-scale systems, allowing researchers to easily study how network topologies shape the performance of several blockchains. We validate Lilith against five industry-battled blockchain systems (Algorand, Diem, Ethereum, Quorum, and Solana) as well as several different network topologies and workloads, including smart contracts and native transactions. We support experimental reproducibility by releasing our datasets and experimental data to the research community.

Submitted…

EVALUATIONS

EVALUATIONS

Research Questions

EVALUATIONS

Research Questions

- **RQ1 -** Is it possible to reproduce results comparable to those achieved on AWS using on-premises infrastructures?

EVALUATIONS

Research Questions

- **RQ1 -** Is it possible to reproduce results comparable to those achieved on AWS using on-premises infrastructures?
- **RQ2 -** Which blockchain is more performant (e.g., TPS) when evaluated with the given topologies?

EVALUATIONS

Research Questions

- **RQ1** - Is it possible to reproduce results comparable to those achieved on AWS using on-premises infrastructures?
- **RQ2** - Which blockchain is more performant (e.g., TPS) when evaluated with the given topologies?
- **RQ3** - How do the characteristics of different topologies influence blockchain performance, and which topology demonstrates greater efficiency in terms of performance when compared across various blockchains, and why?

EVALUATIONS

Research Questions

- **RQ1 -** Is it possible to reproduce results comparable to those achieved on AWS using on-premises infrastructures?
- **RQ2 -** Which blockchain is more performant (e.g., TPS) when evaluated with the given topologies?
- **RQ3 -** How do the characteristics of different topologies influence blockchain performance, and which topology demonstrates greater efficiency in terms of performance when compared across various blockchains, and why?
- **RQ4 -** As the number of nodes in the network increases, how do the performance characteristics of blockchains change?

EVALUATIONS

Research Questions

- **RQ1 -** Is it possible to reproduce results comparable to those achieved on AWS using on-premises infrastructures?
- **RQ2 -** Which blockchain is more performant (e.g., TPS) when evaluated with the given topologies?
- **RQ3 -** How do the characteristics of different topologies influence blockchain performance, and which topology demonstrates greater efficiency in terms of performance when compared across various blockchains, and why?
- **RQ4 -** As the number of nodes in the network increases, how do the performance characteristics of blockchains change?
- **RQ5 -** By using the 2023 and 2024 network datasets to represent two different moments in the cloud, can we assert that the execution of a blockchain benchmark varies depending on these datasets?

EVALUATIONS

Research Questions

- **RQ1** - Is it possible to reproduce results comparable to those achieved on AWS using on-premises infrastructures?
- **RQ2** - Which blockchain is more performant (e.g., TPS) when evaluated with the given topologies?
- **RQ3** - How do the characteristics of different topologies influence blockchain performance, and which topology demonstrates greater efficiency in terms of performance when compared across various blockchains, and why?
- **RQ4** - As the number of nodes in the network increases, how do the performance characteristics of blockchains change?
- **RQ5** - By using the 2023 and 2024 network datasets to represent two different moments in the cloud, can we assert that the execution of a blockchain benchmark varies depending on these datasets?
- **RQ6** - Which blockchain suffers the most when different packet drop rates are applied?

EVALUATIONS

Research Questions

- **RQ1 -** Is it possible to reproduce results comparable to those achieved on AWS using on-premises infrastructures?
- **RQ2 -** Which blockchain is more performant (e.g., TPS) when evaluated with the given topologies?
- **RQ3 -** How do the characteristics of different topologies influence blockchain performance, and which topology demonstrates greater efficiency in terms of performance when compared across various blockchains, and why?
- **RQ4 -** As the number of nodes in the network increases, how do the performance characteristics of blockchains change?
- **RQ5 -** By using the 2023 and 2024 network datasets to represent two different moments in the cloud, can we assert that the execution of a blockchain benchmark varies depending on these datasets?
- **RQ6 -** Which blockchain suffers the most when different packet drop rates are applied?
- **RQ7 -** Regarding the results of the experiments executed using topology construction method 2, what can be inferred about the network configuration used (i.e., Scale-free, switches, links)?

DIABLO BENCHMARK #1

DIABLO BENCHMARK #1

- Versatile blockchain benchmark framework.

DIABLO BENCHMARK #1

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

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.

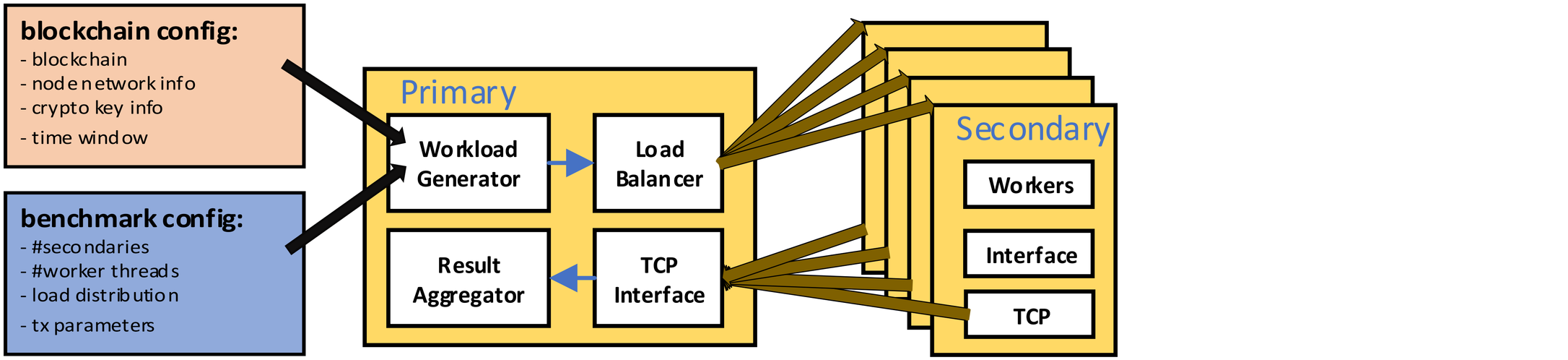
DIABLO BENCHMARK #1

- 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.

DIABLO BENCHMARK #1

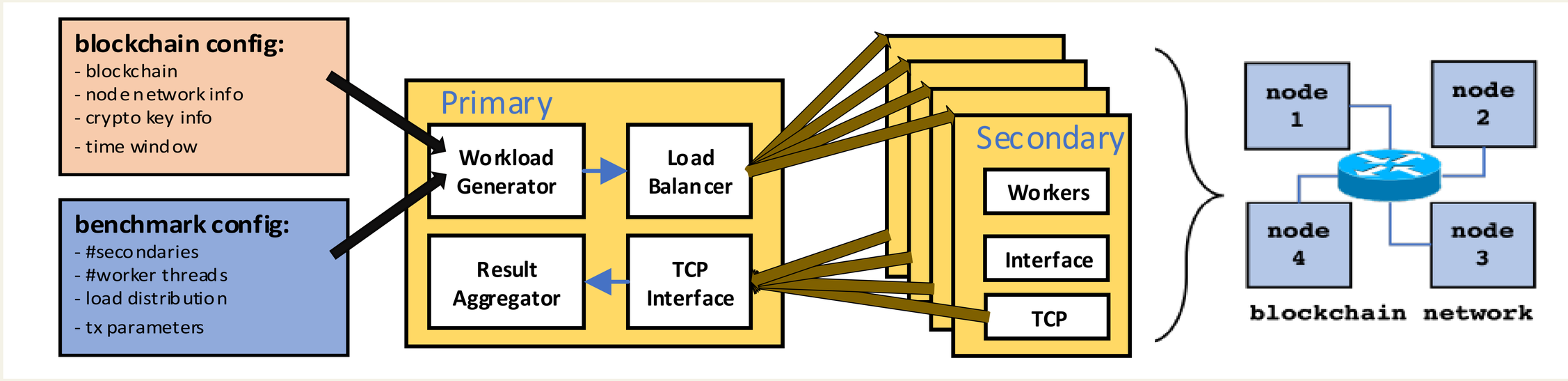
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **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.

DIABLO BENCHMARK #1

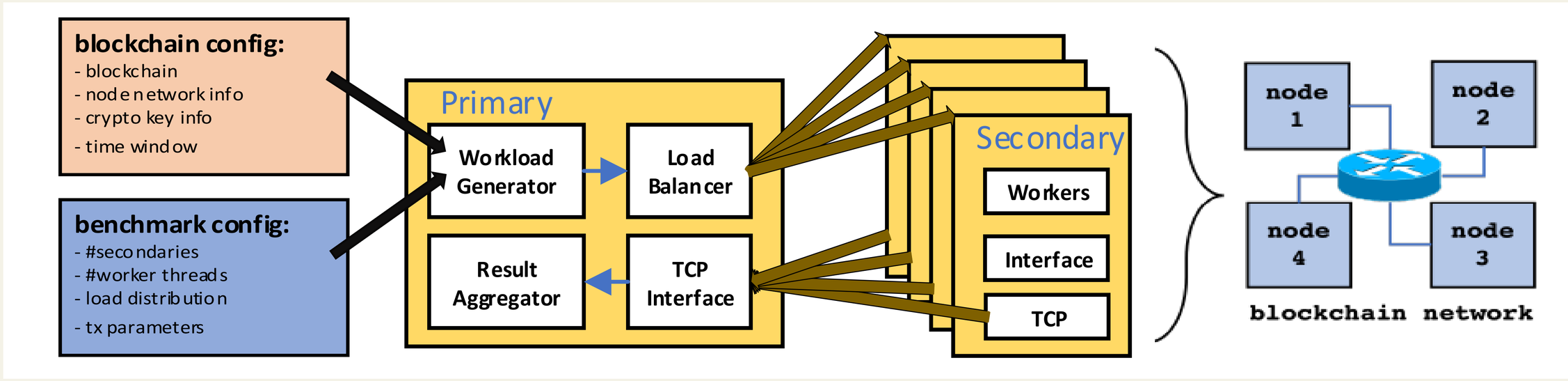
- Versatile blockchain benchmark framework.
- Define, evaluate and compare different blockchains under realistic and tunable workloads/**Dapps (e.g., Exchange, Web Service)**.
- Enhanced distributed workload generation.



- **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.

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

DIABLO BENCHMARK #2

Benchmark definition

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

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;

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

DIABLO BENCHMARK #3

Paper Experimental Settings

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

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 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)**.

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
Katharà	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.

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).

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.

KOLLAPS #1

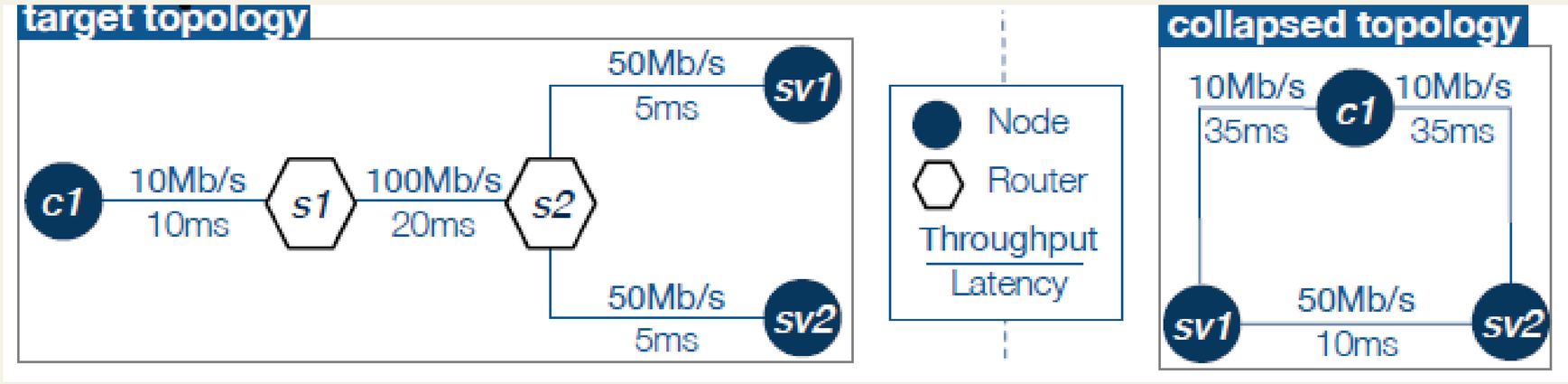
- Decentralized network emulator for large-scale applications.
- Observable end-to-end properties (latency, jitter, bandwidth and packet loss) without emulating the full-state of network elements (routers, switches).
- Fully-distributed emulation model allowing to scale without sacrificing accuracy.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

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

KOLLAPS #2

Components

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

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.

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).

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.

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

KOLLAPS #3

Implementation and evaluation

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

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);

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);

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.

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.

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).

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.

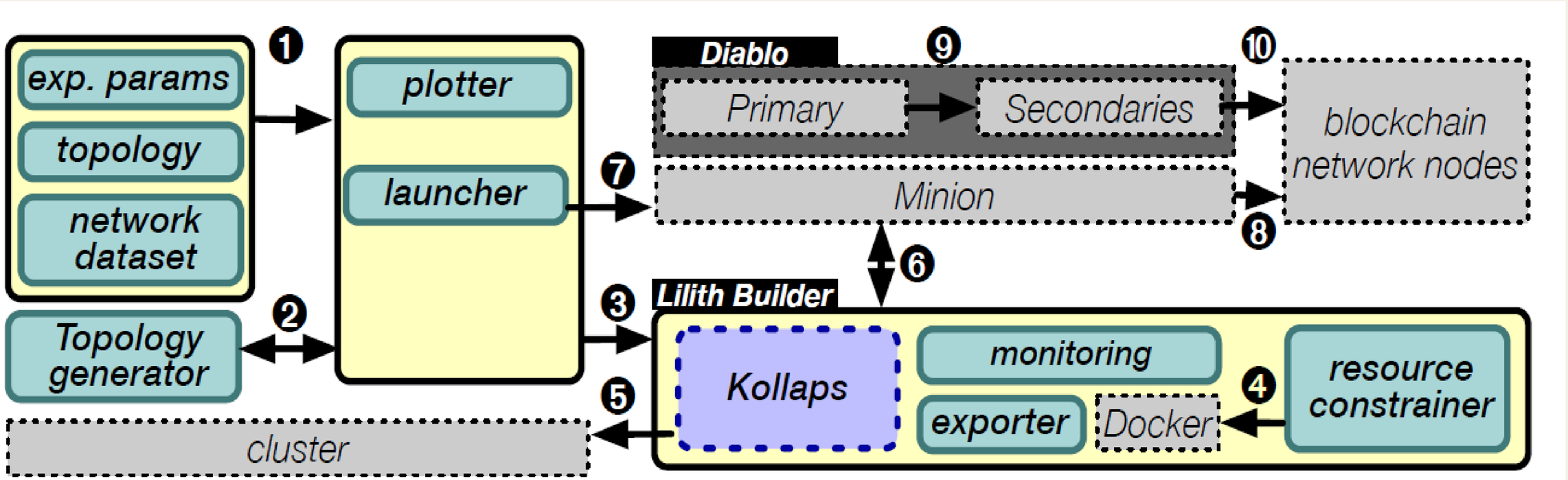
IMPLEMENTATION

IMPLEMENTATION

Benchmarking process

IMPLEMENTATION

Benchmarking process



TESTING SCENARIO

TESTING SCENARIO

Workflow Evolution

TESTING SCENARIO

Workflow Evolution

TESTING SCENARIO

Workflow Evolution

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.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

Challenges related

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

Challenges related

- ***(C1)*** Integration of multiple blockchain systems.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

Challenges related

- ***(C1)*** Integration of multiple blockchain systems.
- ***(C2)*** Improved result analytics alongside blockchain node log files.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

Challenges related

- ***(C1)*** Integration of multiple blockchain systems.
- ***(C2)*** Improved result analytics alongside blockchain node log files.
- ***(C3)*** Transitioning from cloud to cluster, resulting in cost reduction.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

Challenges related

- ***(C1)*** Integration of multiple blockchain systems.
- ***(C2)*** Improved result analytics alongside blockchain node log files.
- ***(C3)*** Transitioning from cloud to cluster, resulting in cost reduction.
- ***(C4)*** Facilitate real-world benchmarking scenarios.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

Challenges related

- ***(C1)*** Integration of multiple blockchain systems.
- ***(C2)*** Improved result analytics alongside blockchain node log files.
- ***(C3)*** Transitioning from cloud to cluster, resulting in cost reduction.
- ***(C4)*** Facilitate real-world benchmarking scenarios.
- ***(C5)*** Enabling network malleability, flexibility, and availability for configuration on a per-request basis.

GOALS

- Study and evaluate how blockchains perform using different network topologies.
- Obtain a cost-effective infrastructure to benchmark blockchain solutions.

Challenges related

- ***(C1)*** Integration of multiple blockchain systems.
- ***(C2)*** Improved result analytics alongside blockchain node log files.
- ***(C3)*** Transitioning from cloud to cluster, resulting in cost reduction.
- ***(C4)*** Facilitate real-world benchmarking scenarios.
- ***(C5)*** Enabling network malleability, flexibility, and availability for configuration on a per-request basis.
- ***(C6)*** Enrich already existing documentation by providing both network datasets and execution traces.

TOPOLOGIES #1

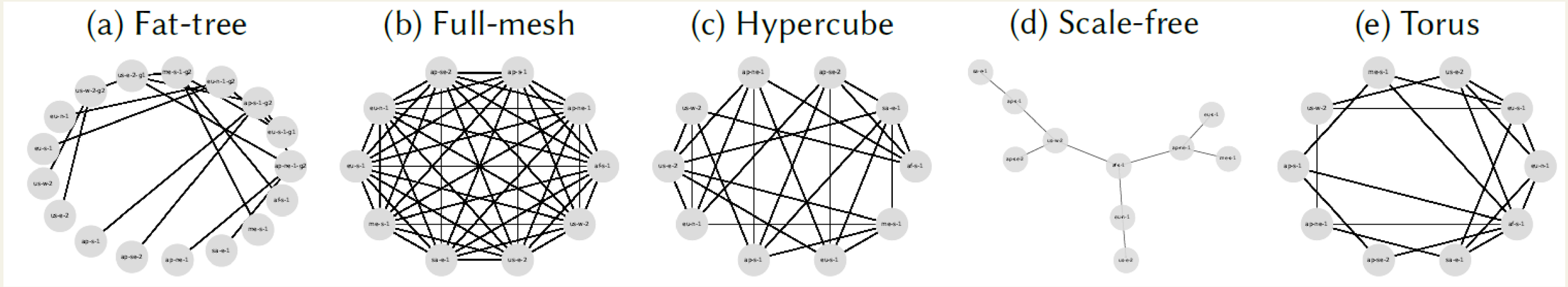
TOPOLOGIES #1

Real-world network topologies employed

TOPOLOGIES #1

Real-world network topologies employed

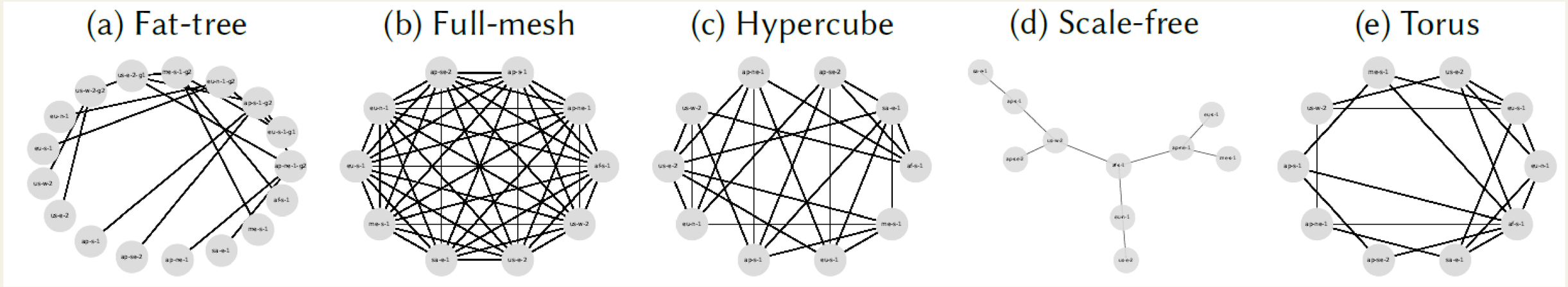
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.



TOPOLOGIES #1

Real-world network topologies employed

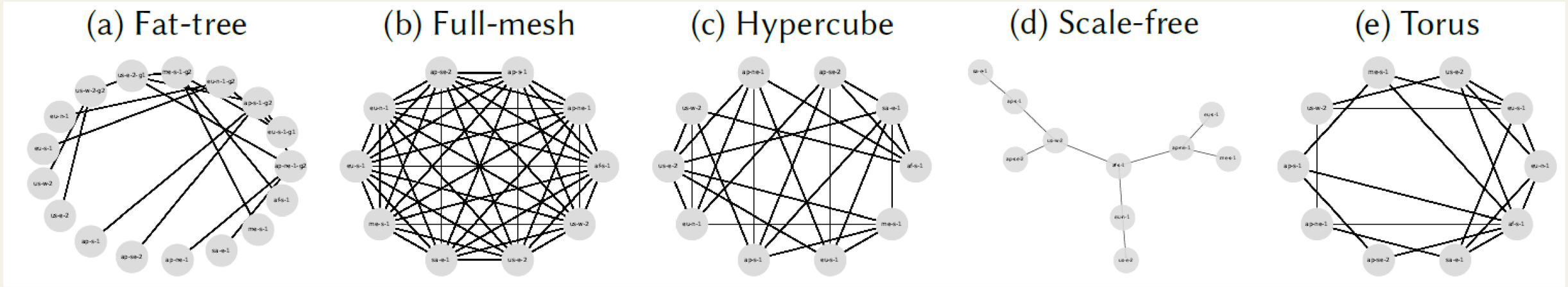
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full-mesh***. Each node is connected to every other node, ensuring maximum connectivity.



TOPOLOGIES #1

Real-world network topologies employed

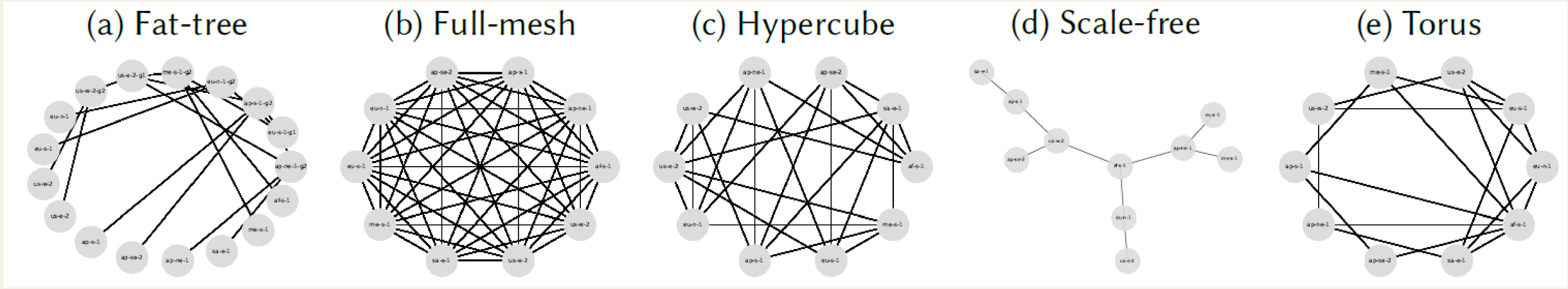
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full-mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional cube.



TOPOLOGIES #1

Real-world network topologies employed

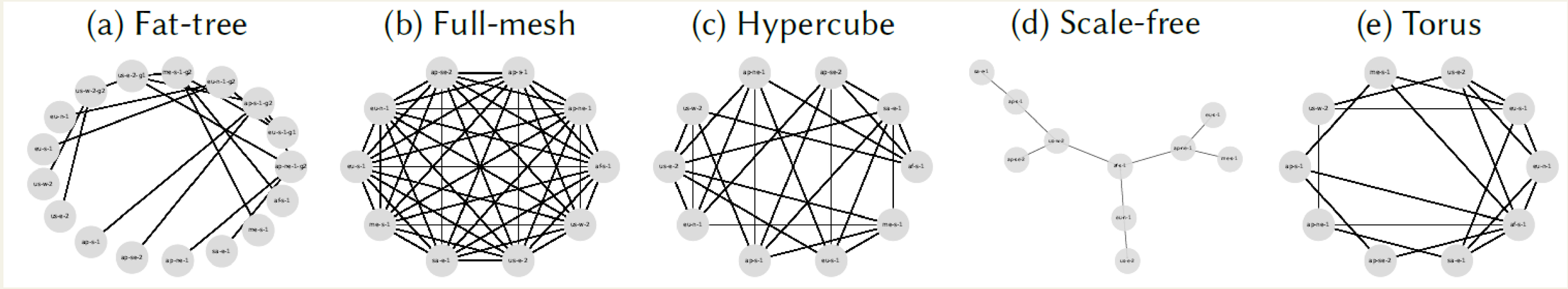
- ***Fat-tree***. A hierarchical topology consisting of core, aggregation, and edge layers.
- ***Full-mesh***. Each node is connected to every other node, ensuring maximum connectivity.
- ***Hypercube***. Each node is connected to its adjacent nodes in a binary-like fashion, forming a multidimensional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



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

TOPOLOGIES #2

Kollaps Network construction

Topology	Degree	Link	Average Latency	Building logic	Latency: %Mean (st.dev.)	TPut: %Mean (st.dev.)
Fat-tree (k ports=4, l level=2)	k	$\frac{k^l}{2}$	102	Latency-based	$133.8(\sigma = 171.2)$	$-65(\sigma = 36.7)$
Full-mesh (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	193.88	–	$2.3(\sigma = 1)$	$-17.7(\sigma = 20.2)$
Hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	207.64	–	$78(\sigma = 173.7)$	$-40.6(\sigma = 37.8)$
Scale-free (N nodes)	(Based on Power Law)		218.3	Latency-based	$380(\sigma = 540)$	$-79.7(\sigma = 28.5)$
Torus (N nodes per row=5, n dimensions=2)	$2n$	$n \cdot N^n$	196.80	Latency-based	$119(\sigma = 151.9)$	$-53.9(\sigma = 42.47)$
AWS-2024 (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	188.15	–	$6.5(\sigma = 3.5)$	$-55.4(\sigma = 10.9)$

TOPOLOGIES #2

Kollaps Network construction

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

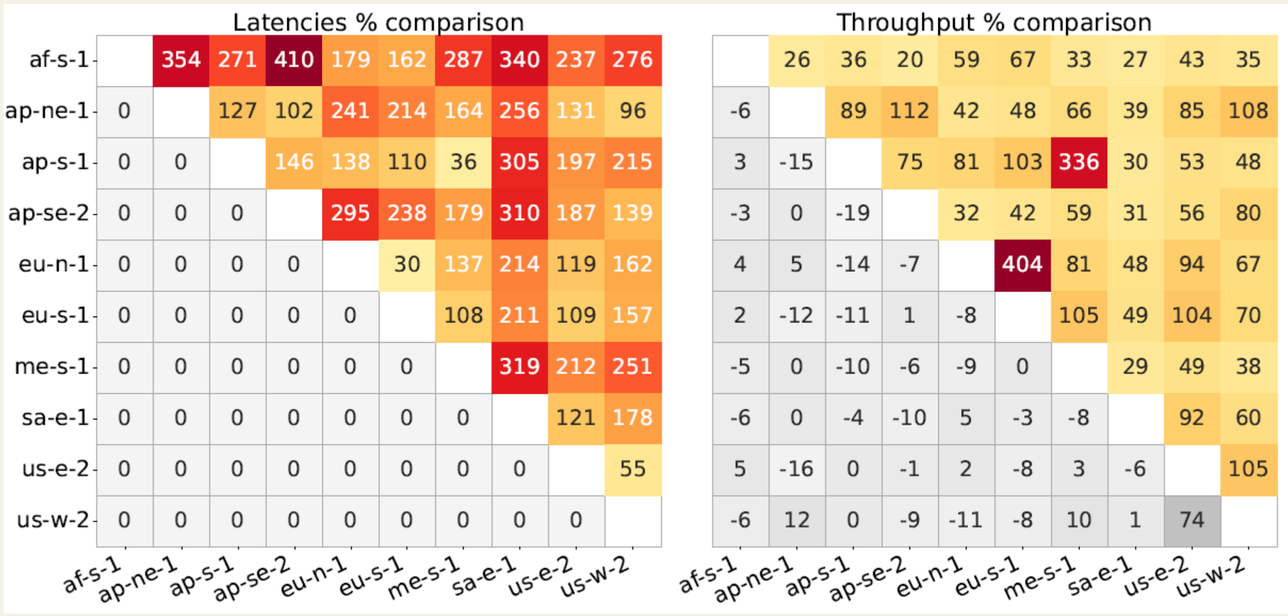
Topology	Degree	Link	Average Latency	Building logic	Latency: %Mean (st.dev.)	TPut: %Mean (st.dev.)
Fat-tree (k ports=4, l level=2)	k	$\frac{k^l}{2}$	102	Latency-based	$133.8(\sigma = 171.2)$	$-65(\sigma = 36.7)$
Full-mesh (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	193.88	–	$2.3(\sigma = 1)$	$-17.7(\sigma = 20.2)$
Hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	207.64	–	$78(\sigma = 173.7)$	$-40.6(\sigma = 37.8)$
Scale-free (N nodes)	(Based on Power Law)		218.3	Latency-based	$380(\sigma = 540)$	$-79.7(\sigma = 28.5)$
Torus (N nodes per row=5, n dimensions=2)	$2n$	$n \cdot N^n$	196.80	Latency-based	$119(\sigma = 151.9)$	$-53.9(\sigma = 42.47)$
AWS-2024 (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	188.15	–	$6.5(\sigma = 3.5)$	$-55.4(\sigma = 10.9)$

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).



Topology	Degree	Link	Average Latency	Building logic	Latency: %Mean (st.dev.)	TPut: %Mean (st.dev.)
Fat-tree (k ports=4, l level=2)	k	$\frac{k^l}{2}$	102	Latency-based	$133.8(\sigma = 171.2)$	$-65(\sigma = 36.7)$
Full-mesh (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	193.88	–	$2.3(\sigma = 1)$	$-17.7(\sigma = 20.2)$
Hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	207.64	–	$78(\sigma = 173.7)$	$-40.6(\sigma = 37.8)$
Scale-free (N nodes)	(Based on Power Law)		218.3	Latency-based	$380(\sigma = 540)$	$-79.7(\sigma = 28.5)$
Torus (N nodes per row=5, n dimensions=2)	$2n$	$n \cdot N^n$	196.80	Latency-based	$119(\sigma = 151.9)$	$-53.9(\sigma = 42.47)$
AWS-2024 (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	188.15	–	$6.5(\sigma = 3.5)$	$-55.4(\sigma = 10.9)$

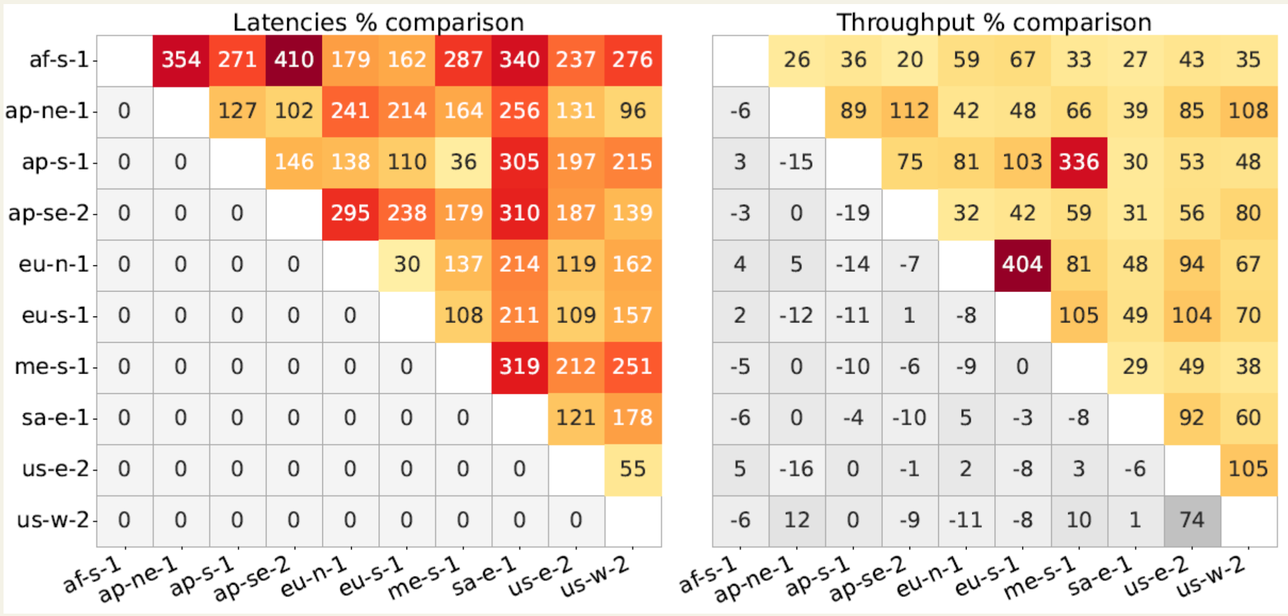
TOPOLOGIES #2

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	Link	Average Latency	Building logic	Latency: %Mean (st.dev.)	TPut: %Mean (st.dev.)
Fat-tree (k ports=4, l level=2)	k	$\frac{k^l}{2}$	102	Latency-based	$133.8(\sigma = 171.2)$	$-65(\sigma = 36.7)$
Full-mesh (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	193.88	–	$2.3(\sigma = 1)$	$-17.7(\sigma = 20.2)$
Hypercube (n dimensions=4)	n	$n \cdot 2^{(n-1)}$	207.64	–	$78(\sigma = 173.7)$	$-40.6(\sigma = 37.8)$
Scale-free (N nodes)	(Based on Power Law)		218.3	Latency-based	$380(\sigma = 540)$	$-79.7(\sigma = 28.5)$
Torus (N nodes per row=5, n dimensions=2)	$2n$	$n \cdot N^n$	196.80	Latency-based	$119(\sigma = 151.9)$	$-53.9(\sigma = 42.47)$
AWS-2024 (N nodes)	$N-1$	$\frac{N\times(N-1)}{2}$	188.15	–	$6.5(\sigma = 3.5)$	$-55.4(\sigma = 10.9)$

TOPOLOGIES #3

TOPOLOGIES #3

Kollaps Network construction method 2

TOPOLOGIES #3

Kollaps Network construction method 2

- To gain more insights from the characteristics of each topology (e.g., considering shared links).

TOPOLOGIES #3

Kollaps Network construction method 2

- To gain more insights from the characteristics of each topology (e.g., considering shared links).

- Given an arbitrary number of switches (independent from the cloud regions), where the switches of the AWS regions then connect randomly.

TOPOLOGIES #3

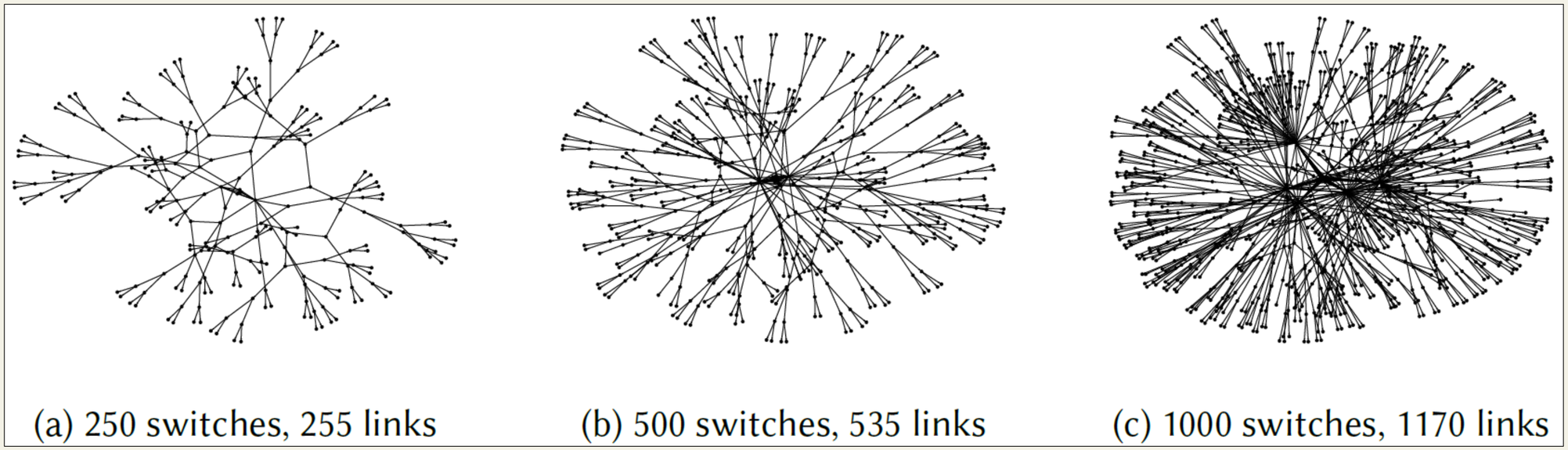
Kollaps Network construction method 2

- To gain more insights from the characteristics of each topology (e.g., considering shared links).
- Given an arbitrary number of switches (independent from the cloud regions), where the switches of the AWS regions then connect randomly.
- We select the Scale-free topology, as it provides a high variation in linkages and degree among nodes.

TOPOLOGIES #3

Kollaps Network construction method 2

- To gain more insights from the characteristics of each topology (e.g., considering shared links).
- Given an arbitrary number of switches (independent from the cloud regions), where the switches of the AWS regions then connect randomly.
- We select the Scale-free topology, as it provides a high variation in linkages and degree among nodes.
- We use an increasing number of switches (250, 500, and 1000) with an increasing latency value (25, 50, 100) on the links.



BLOCKCHAINS

BLOCKCHAINS

- ***Ethereum Clique (Proof-of-Authority).*** Validates transactions in a round-robin fashion through approved validators with a minimum block interval.

	Throughput assertion (TPS)	Latency Block Finality (s)	Consensus	VM	DApp language
Ethereum	10−15 [96]	10−20 [61]	Clique	geth	Solidity
Quorum	0.7K−2.5K [5]	2−15 [70]	IBFT	geth	Solidity
Solana	65K [16]	12 [17]	TowerBFT	eBPF	Solang
Algorand	7.5K [6]	3.3 [6]	BA⋆	AVM	PyTeal
Diem	60−1K [32, 114]	100 [85]	HotStuff	MoveVM	Move

BLOCKCHAINS

- ***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.

	Throughput assertion (TPS)	Latency Block Finality (s)	Consensus	VM	DApp language
Ethereum	10−15 [96]	10−20 [61]	Clique	geth	Solidity
Quorum	0.7K−2.5K [5]	2−15 [70]	IBFT	geth	Solidity
Solana	65K [16]	12 [17]	TowerBFT	eBPF	Solang
Algorand	7.5K [6]	3.3 [6]	BA⋆	AVM	PyTeal
Diem	60−1K [32, 114]	100 [85]	HotStuff	MoveVM	Move

BLOCKCHAINS

- ***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.

	Throughput assertion (TPS)	Latency Block Finality (s)	Consensus	VM	DApp language
Ethereum	10−15 [96]	10−20 [61]	Clique	geth	Solidity
Quorum	0.7K−2.5K [5]	2−15 [70]	IBFT	geth	Solidity
Solana	65K [16]	12 [17]	TowerBFT	eBPF	Solang
Algorand	7.5K [6]	3.3 [6]	BA⋆	AVM	PyTeal
Diem	60−1K [32, 114]	100 [85]	HotStuff	MoveVM	Move

BLOCKCHAINS

- ***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.
- ***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.

	Throughput assertion (TPS)	Latency Block Finality (s)	Consensus	VM	DApp language
Ethereum	10−15 [96]	10−20 [61]	Clique	geth	Solidity
Quorum	0.7K−2.5K [5]	2−15 [70]	IBFT	geth	Solidity
Solana	65K [16]	12 [17]	TowerBFT	eBPF	Solang
Algorand	7.5K [6]	3.3 [6]	BA⋆	AVM	PyTeal
Diem	60−1K [32, 114]	100 [85]	HotStuff	MoveVM	Move

BLOCKCHAINS

- ***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.
- ***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.

	Throughput assertion (TPS)	Latency Block Finality (s)	Consensus	VM	DApp language
Ethereum	10−15 [96]	10−20 [61]	Clique	geth	Solidity
Quorum	0.7K−2.5K [5]	2−15 [70]	IBFT	geth	Solidity
Solana	65K [16]	12 [17]	TowerBFT	eBPF	Solang
Algorand	7.5K [6]	3.3 [6]	BA⋆	AVM	PyTeal
Diem	60−1K [32, 114]	100 [85]	HotStuff	MoveVM	Move

WORKLOADS

WORKLOADS

- Based on both native transactions and smart contract requests.

WORKLOADS

- Based on both native transactions and smart contract requests.
- ***PayPal and VISA.*** To test their transactions capacity to a blockchain.

Workload	Type	Duration (s)	Scenario	TPS
PayPal*	Native Tx	300	Constant Rate	200
VISA*	Native Tx	300	Constant Rate	1.8 K
DDoS	Native Tx	120	Constant Rate	10 K
Exchange	Smart-contract	180	Burst	20 K down to 100
Gaming	Smart-contract	276	Intensive	13 K
FIFA*	Smart-contract	100	High sending rate	45 K

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***PayPal and VISA.*** To test their transactions capacity to a blockchain.
- ***DDoS.*** To test the robustness of the specific blockchain under high request scenarios.

Workload	Type	Duration (s)	Scenario	TPS
PayPal*	Native Tx	300	Constant Rate	200
VISA*	Native Tx	300	Constant Rate	1.8 K
DDoS	Native Tx	120	Constant Rate	10 K
Exchange	Smart-contract	180	Burst	20 K down to 100
Gaming	Smart-contract	276	Intensive	13 K
FIFA*	Smart-contract	100	High sending rate	45 K

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***PayPal and VISA.*** To test their transactions capacity to a blockchain.
- ***DDoS.*** To test the robustness of the specific blockchain under high request scenarios.
- ***Exchange/Nasdaq.*** 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.

Workload	Type	Duration (s)	Scenario	TPS
PayPal*	Native Tx	300	Constant Rate	200
VISA*	Native Tx	300	Constant Rate	1.8 K
DDoS	Native Tx	120	Constant Rate	10 K
Exchange	Smart-contract	180	Burst	20 K down to 100
Gaming	Smart-contract	276	Intensive	13 K
FIFA*	Smart-contract	100	High sending rate	45 K

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***PayPal and VISA.*** To test their transactions capacity to a blockchain.
- ***DDoS.*** To test the robustness of the specific blockchain under high request scenarios.
- ***Exchange/Nasdaq.*** 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/Dota2.*** Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.

Workload	Type	Duration (s)	Scenario	TPS
PayPal*	Native Tx	300	Constant Rate	200
VISA*	Native Tx	300	Constant Rate	1.8 K
DDoS	Native Tx	120	Constant Rate	10 K
Exchange	Smart-contract	180	Burst	20 K down to 100
Gaming	Smart-contract	276	Intensive	13 K
FIFA*	Smart-contract	100	High sending rate	45 K

WORKLOADS

- Based on both native transactions and smart contract requests.

- ***PayPal and VISA.*** To test their transactions capacity to a blockchain.
- ***DDoS.*** To test the robustness of the specific blockchain under high request scenarios.
- ***Exchange/Nasdaq.*** 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/Dota2.*** Executing the trace of an online battle arena video game comprising players who interact with each other and with the environment.
- ***Web service/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.

Workload	Type	Duration (s)	Scenario	TPS
PayPal*	Native Tx	300	Constant Rate	200
VISA*	Native Tx	300	Constant Rate	1.8 K
DDoS	Native Tx	120	Constant Rate	10 K
Exchange	Smart-contract	180	Burst	20 K down to 100
Gaming	Smart-contract	276	Intensive	13 K
FIFA*	Smart-contract	100	High sending rate	45 K

FINDINGS

Answer to RQ1: *Although Diablo's team states it uses configurations with this network setup in [72], it doesn't display the corresponding results (except for native transactions workloads) or explain any deployment errors, if present. Therefore, we can only partially answer this question if we consider that, for instance, Quorum fails to commit under heavier workloads also in [72], or that Ethereum's performance is low in [72] as well.*

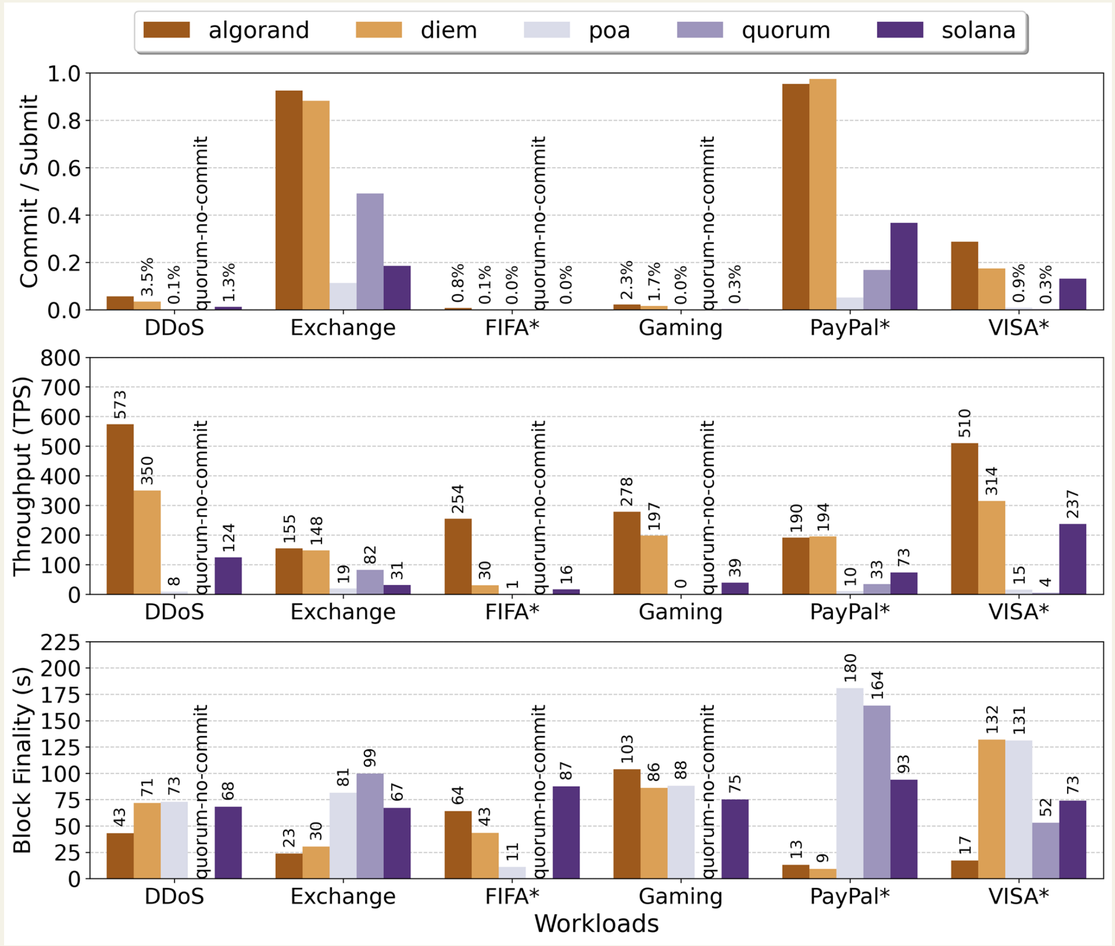
Answer to RQ2: *We report that Diem and Algorand exhibit more consistent results across different topologies, with Algorand performing better under more demanding workloads. Following them are Quorum and Solana. Finally, Ethereum, while showing consistent results, has the lowest performance among the blockchains tested.*

Answer to RQ3: *Considering the characteristics in Table 1 for each topology used in the experiments, we observe that Torus performs quite well with higher workloads, likely due to its high number of links and low number of switches, which ensure lower congestion. The topologies that perform best appear to be Full-mesh and Hypercube, due to their high degree and link capacity.*

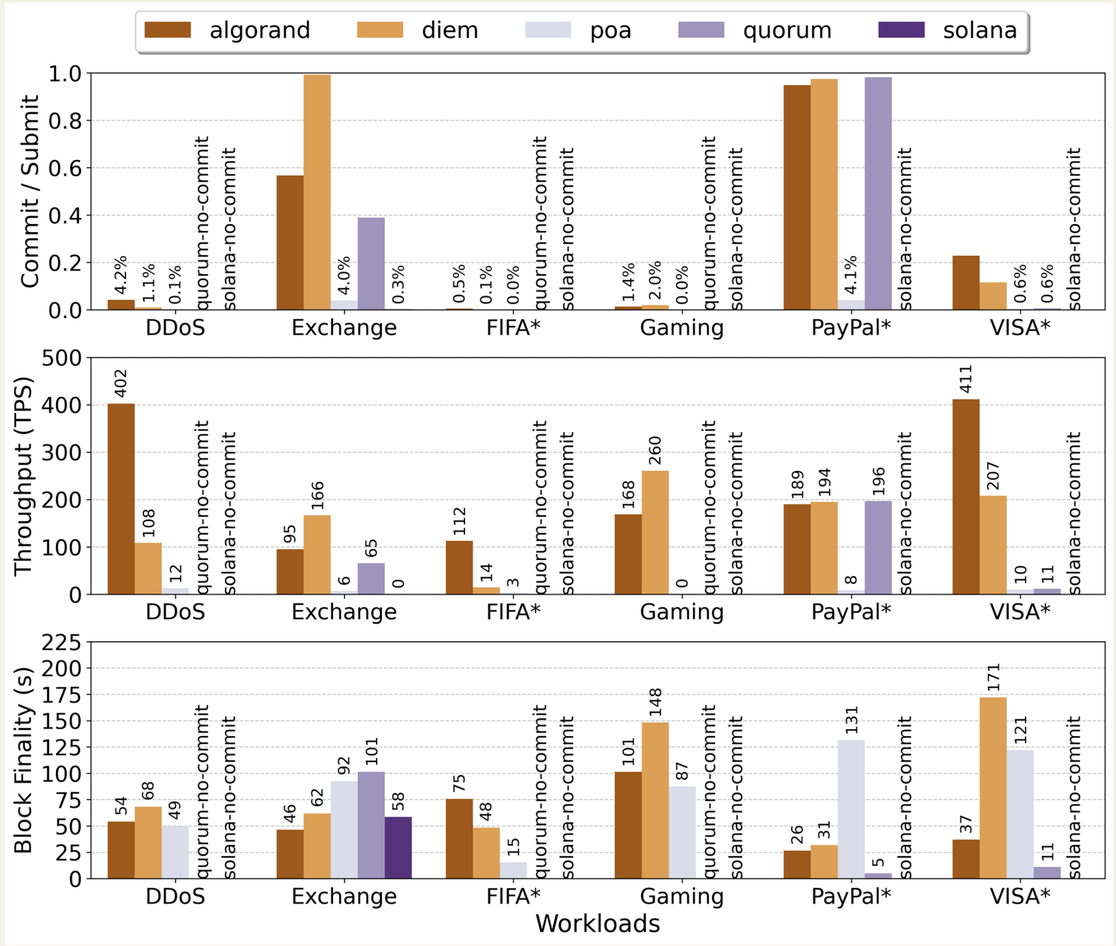
Answer to RQ4: *We observe that increasing the nodes per region the commit rate decreases and the average block latency increases.*

RESULTS #1

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



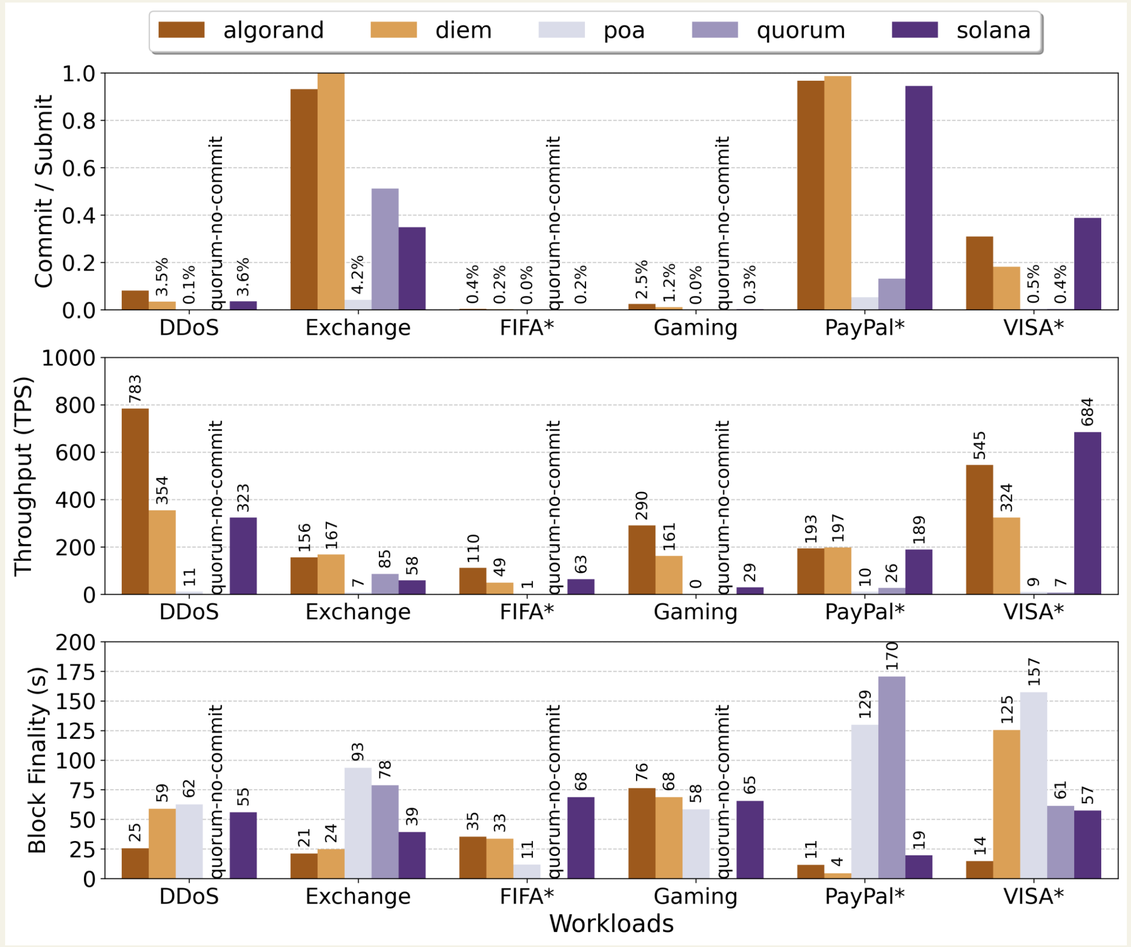
1-node per region



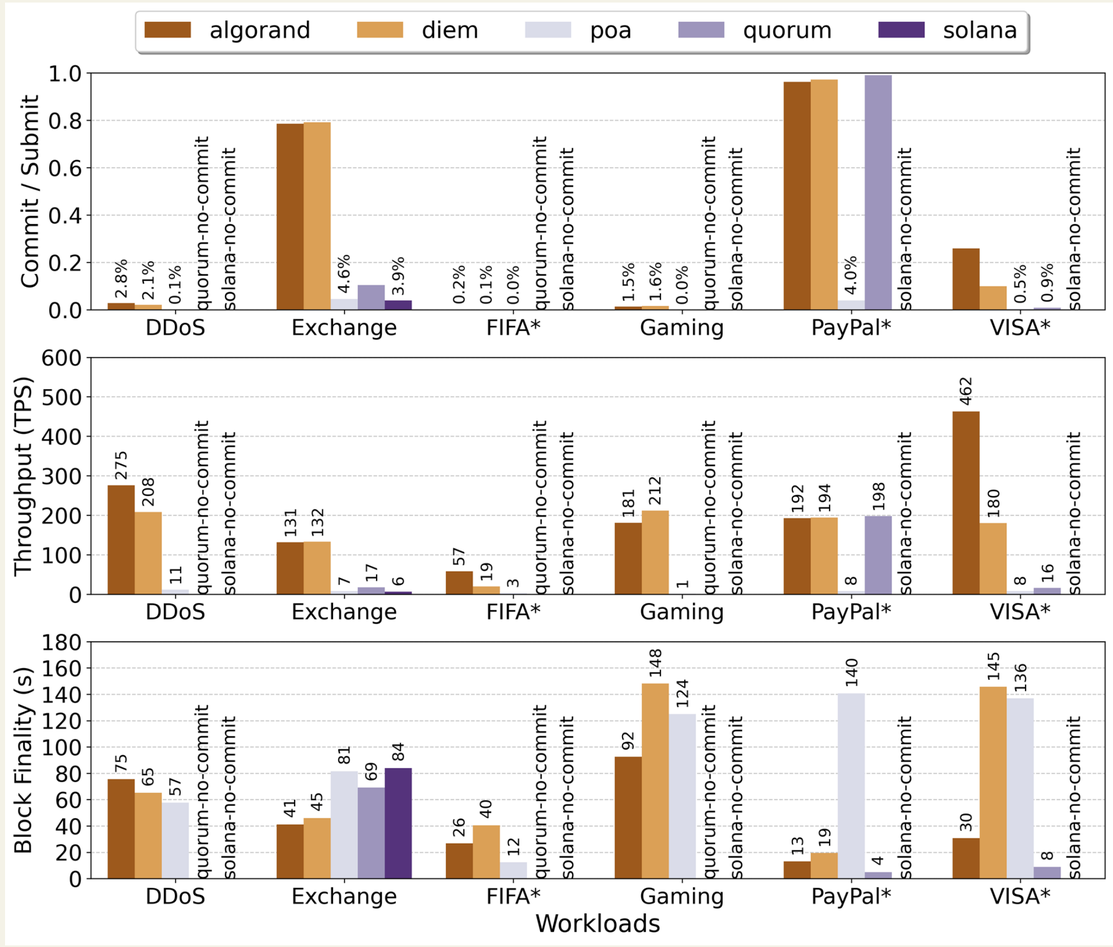
4-nodes per region

RESULTS #2

Benchmark on a Full-mesh topology with different blockchain network size



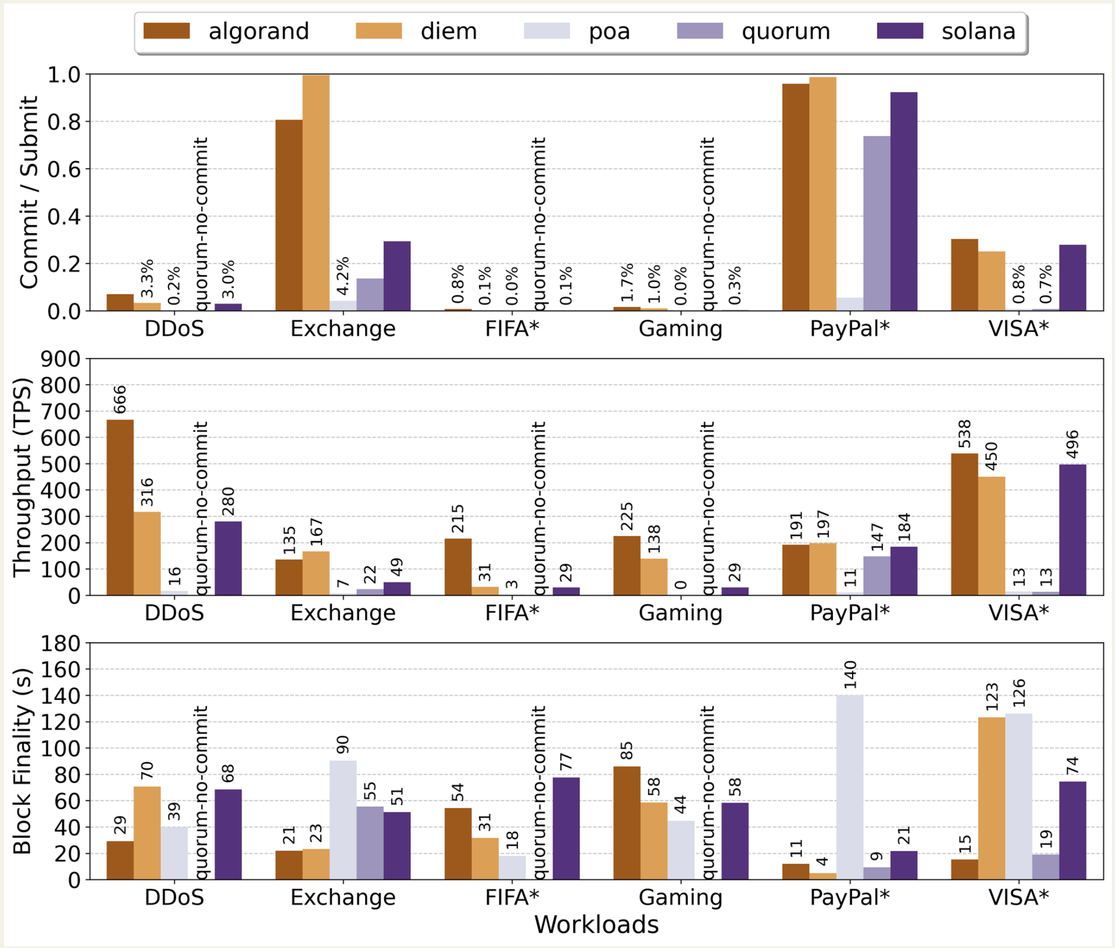
1-node per region



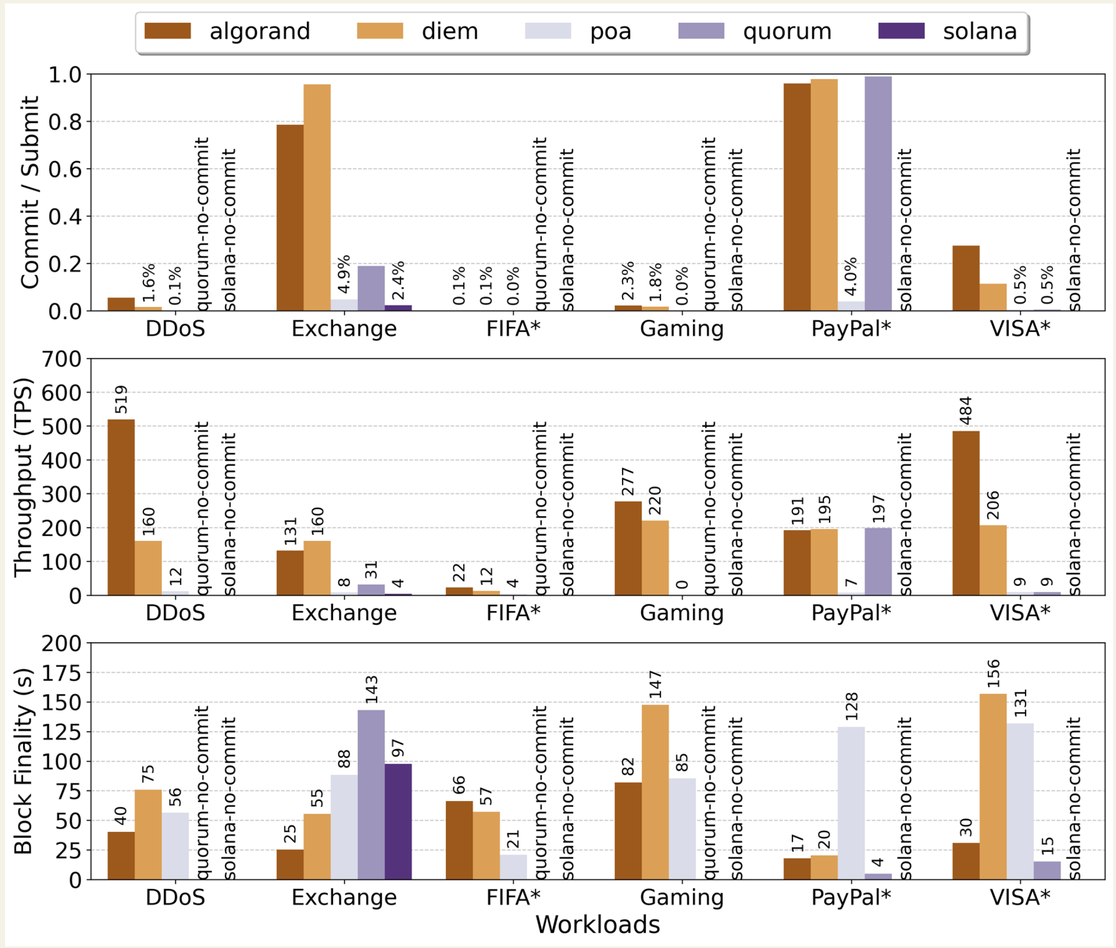
4-nodes per region

RESULTS #3

Benchmark on a Hypercube topology with different blockchain network size



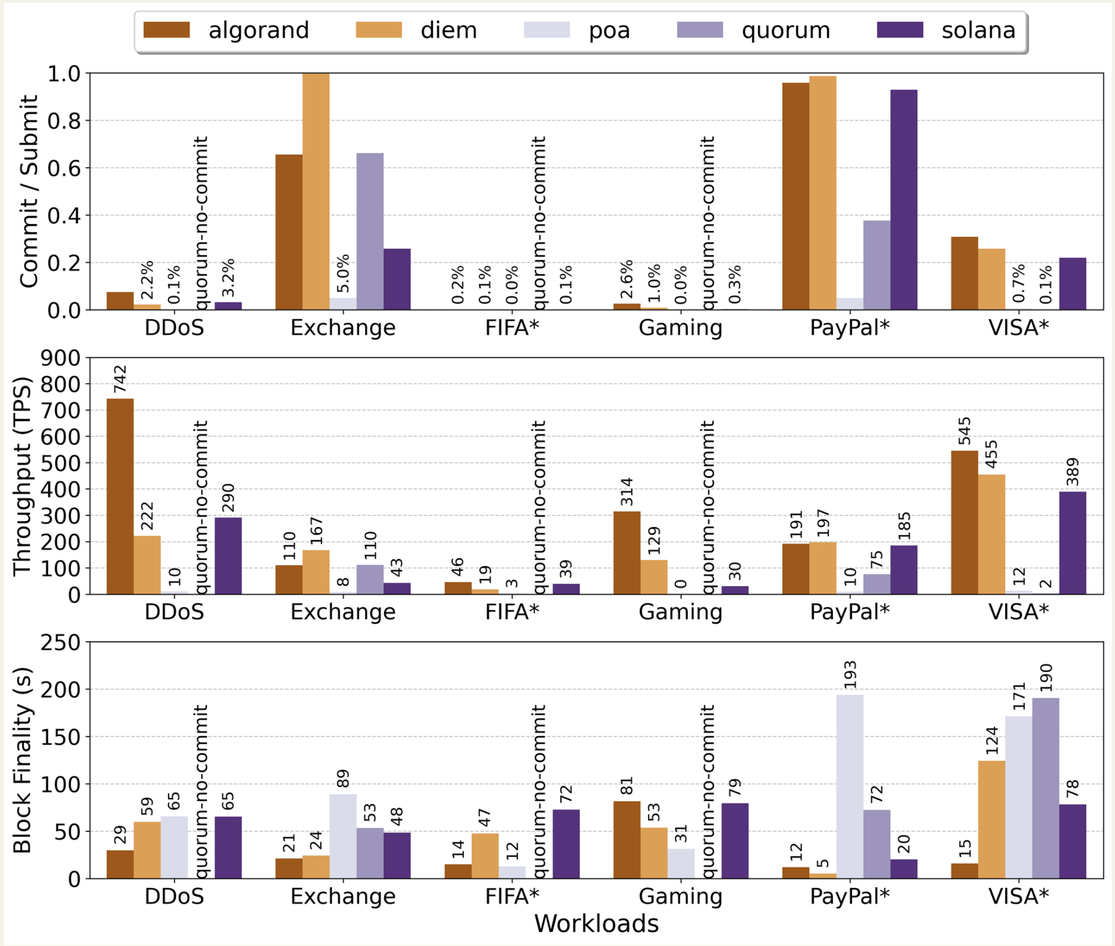
1-node per region



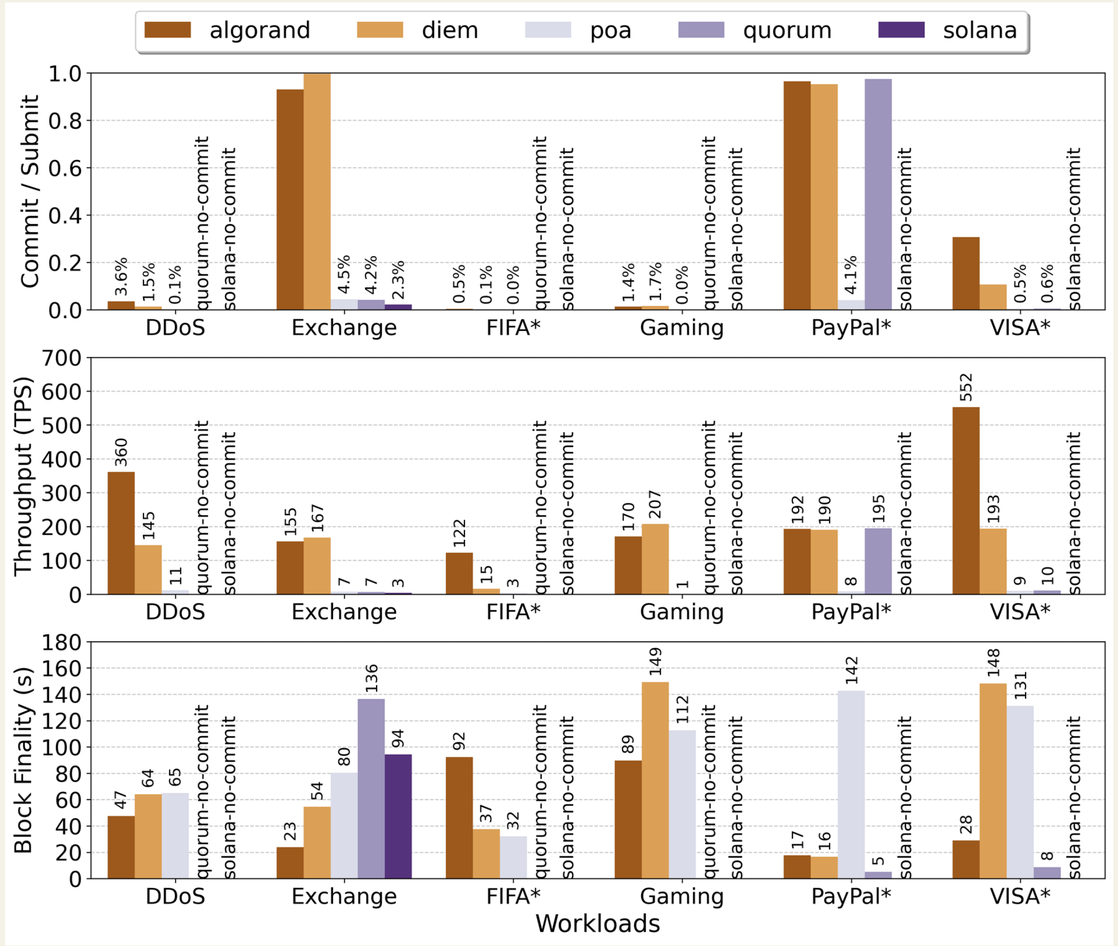
4-nodes per region

RESULTS #4

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



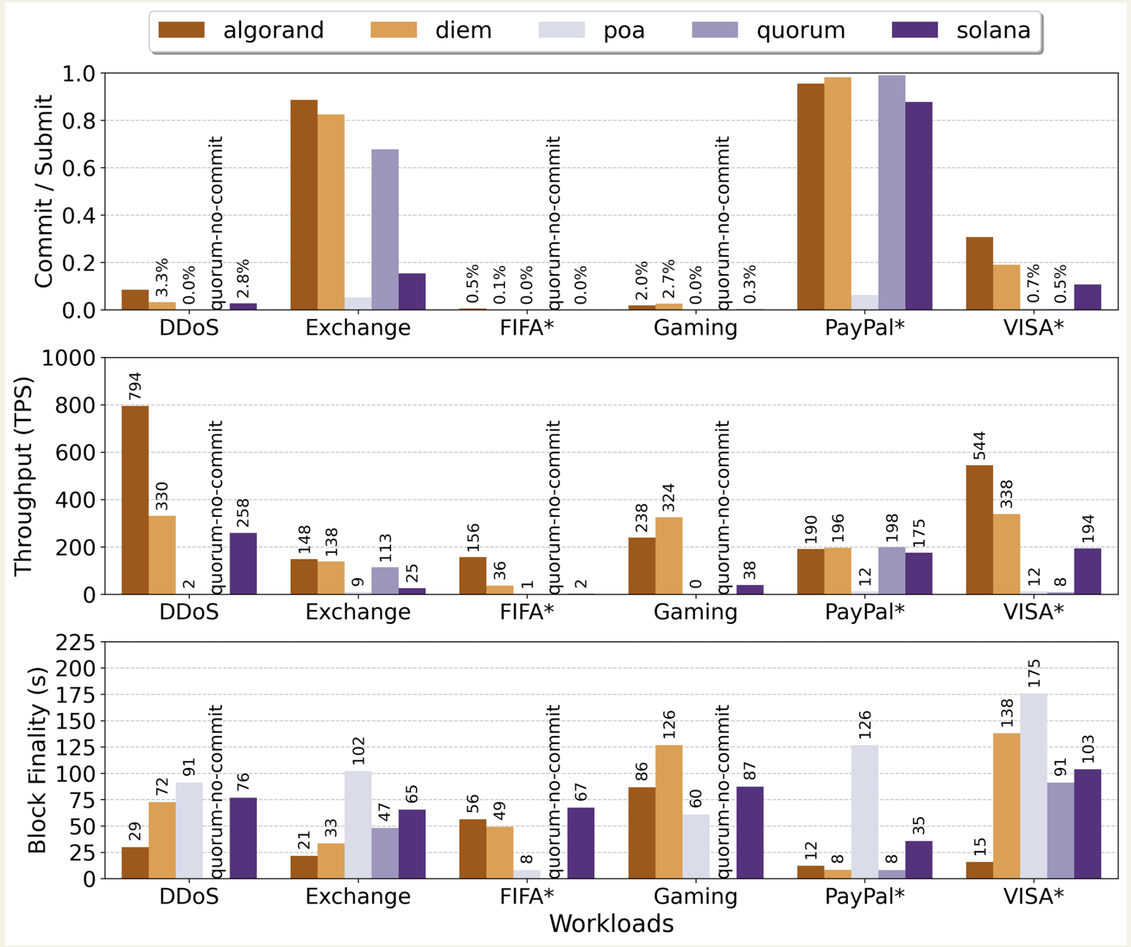
1-node per region



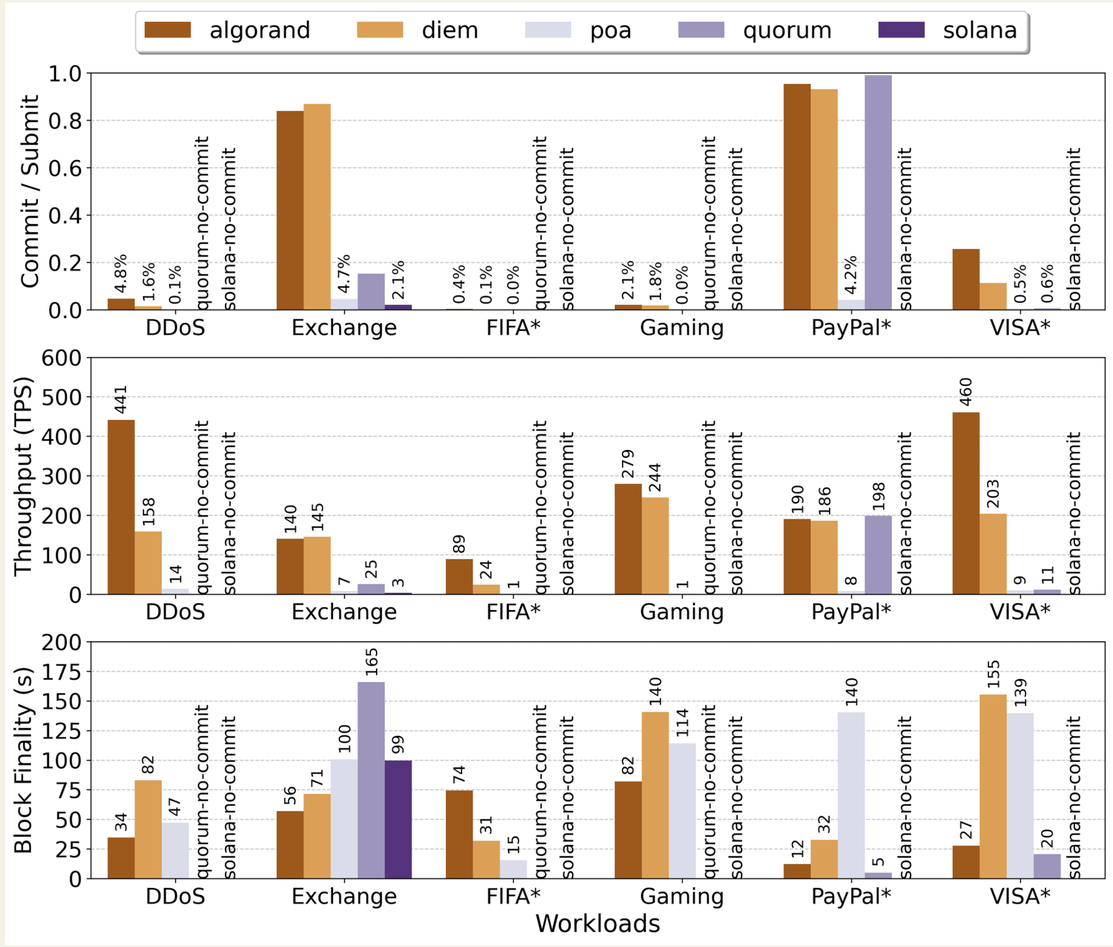
4-nodes per region

RESULTS #5

Benchmark on a Torus topology with different blockchain network size



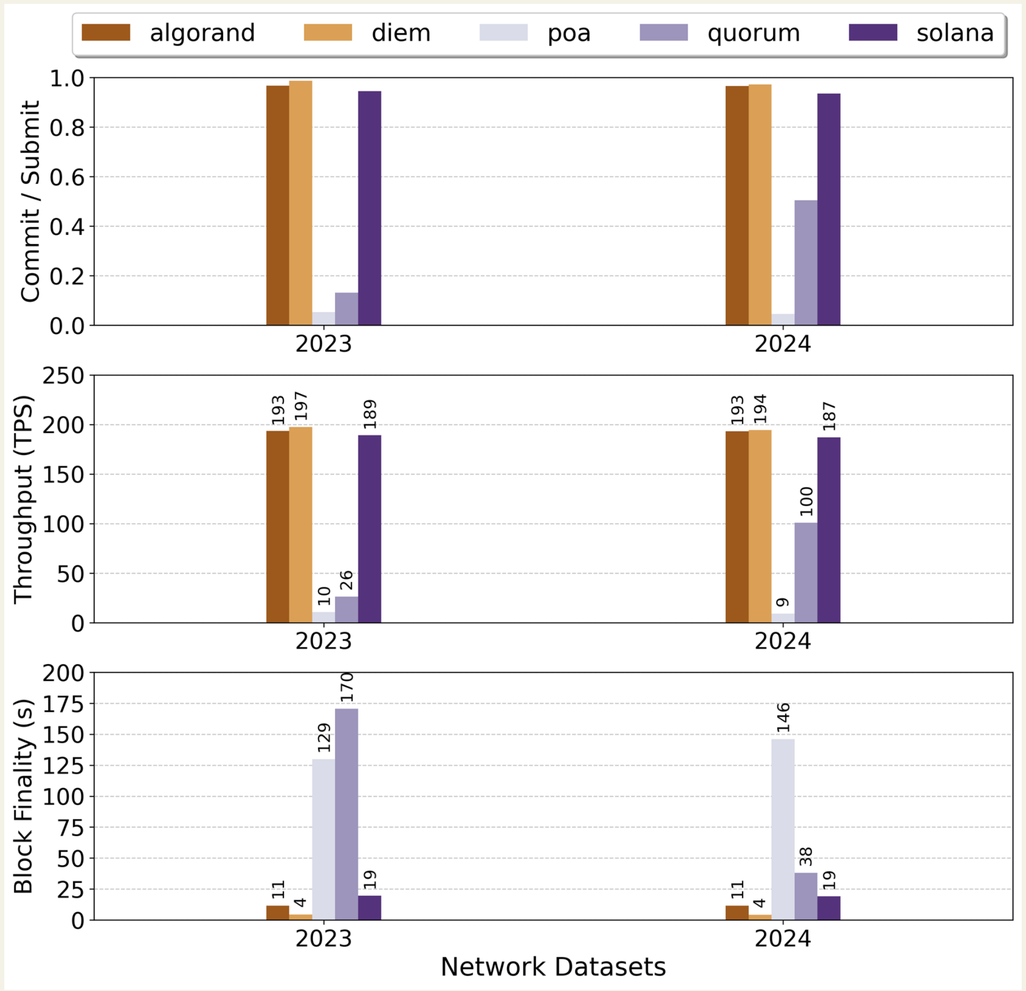
1-node per region



4-nodes per region

RESULTS #6

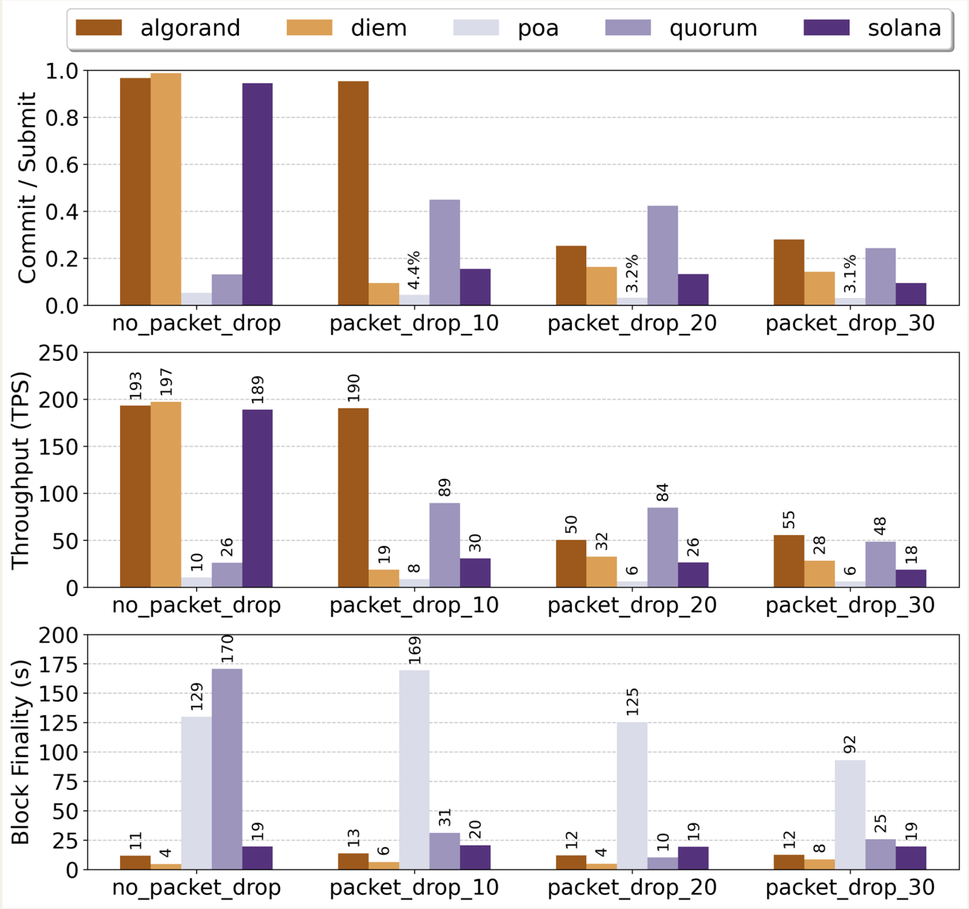
Benchmark on a Full-mesh topology with different network datasets



Answer to RQ5: *We can confirm that emulating two different moments taken from AWS in two separate measurements (2023 and 2024) shows different results for the benchmark. Referring specifically to Fig. 12, the most significant variation impacts Quorum.*

RESULTS #7

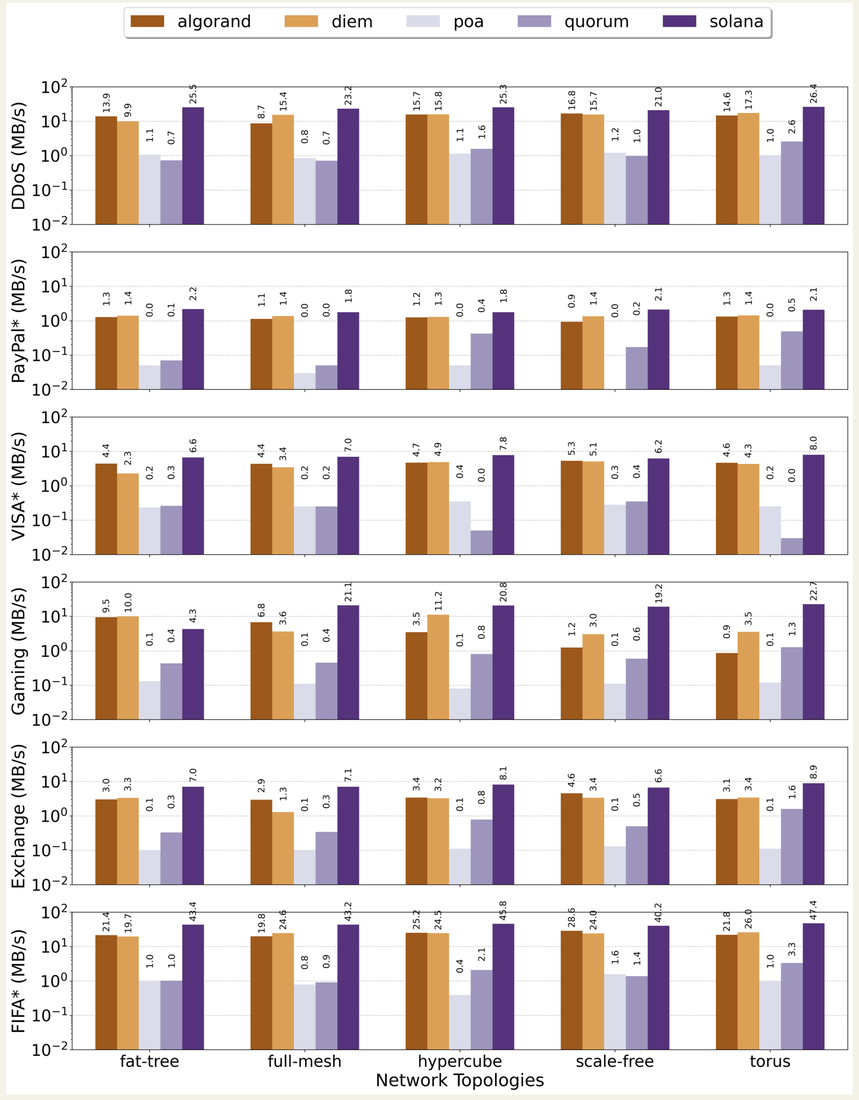
Benchmark on a Full-mesh topology with different packet drop percentage during the workload execution



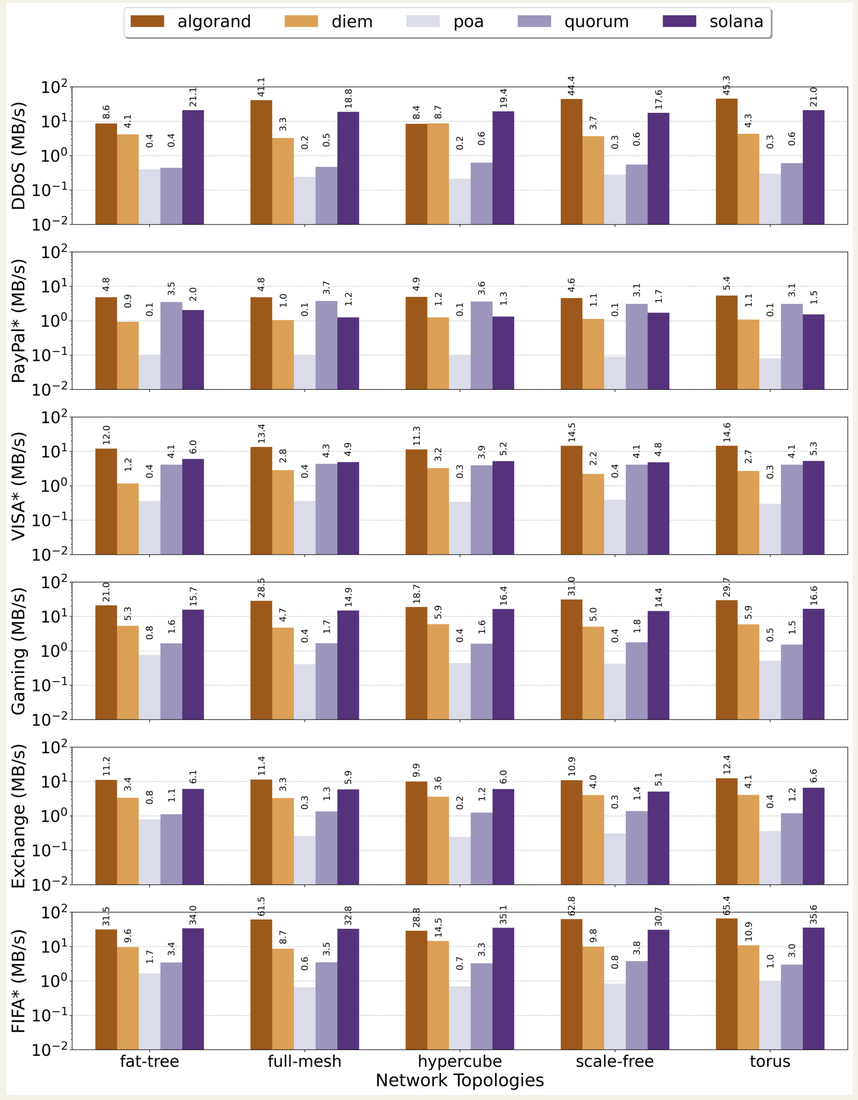
Answer to RQ6: *Regarding throughput, the blockchains that show the most performance degradation as the packet drop rate increases are Diem and Solana, followed by Quorum and Algorand. Although Ethereum consistently has a much lower TPS compared to the others, it is the least affected by the packet drop rate.*

RESULTS #8

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



1-node per region



4-nodes per region

RESULTS #9

Experiments with Scale-free and increasing number of switches-link

Table 4. Results for experiments executed using the topology construction method 2 on Scale-free shape with increasing number of switches-links (250-255, 500-535, 1000-1170) and latencies (25, 50, 100 ms). On the left 0.1 Gbps, while on the right side 1 Gbps.

		Algorand			Diem			PoA			Solana		
		250	500	1000	250	500	1000	250	500	1000	250	500	1000
Throughput (TPS)	25	31.7	46.0	62.0	15.7	19.6	46.1	4.2	3.0	2.9	43.4	54.3	72.4
	50	204.9	90.7	171.5	13.1	21.9	8.9	1.8	1.1	2.2	–	97.2	36.6
	100	40.7	116.8	128.1	40.3	14.4	15.9	1.1	2.1	4.2	–	–	–
Block Finality (s)	25	17.2	37.8	53.1	60.8	54.2	36.4	9.1	12.8	8.4	65.5	60.6	61.8
	50	58.6	58.9	68.0	35.6	51.7	38.3	6.6	3.4	8.2	–	56.8	66.9
	100	66.6	80.5	61.3	79.2	33.9	61.2	11.1	13.7	8.2	–	–	–

		Algorand			Diem			PoA			Solana		
		250	500	1000	250	500	1000	250	500	1000	250	500	1000
Throughput (TPS)	25	27.4	338.2	13.0	20.5	40.0	48.3	5.3	3.4	1.4	10.9	70.2	42.4
	50	25.8	104.3	72.0	10.6	59.4	13.9	0.9	3.7	2.1	–	66.2	49.9
	100	111.0	77.0	111.9	15.4	1.2	35.2	2.1	0.8	4.2	–	–	–
Block Finality (s)	25	20.3	58.4	13.0	41.3	58.5	31.9	16.2	7.1	11.0	75.1	59.9	68.3
	50	19.8	58.7	58.3	30.1	57.3	50.3	3.3	16.8	8.0	–	74.9	67.7
	100	79.5	67.1	57.5	66.7	74.1	36.1	2.4	10.5	7.1	–	–	–

Answer to RQ7: *We observe that: (i) increasing the available bandwidth does not always improve Block Finality; (ii) adding more switches amplifies the dispersive effect of messages in the network for some blockchains (e.g., Diem); (iii) with the same number of switches, increased latency does not always lead to performance degradation.*

FUTURE WORK & CONCLUSIONS

FUTURE WORK & CONCLUSIONS

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

FUTURE WORK & CONCLUSIONS

- Collect and utilize real blockchain data (from specific periods)
- Use temporal data to simulate dynamic network events

FUTURE WORK & CONCLUSIONS

- 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

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