Icarus: a Caching Simulator for Information Centric Networking (ICN)

Lorenzo Saino, Ioannis Psaras and George Pavlou

Communications and Information Systems Group
Department of Electronic and Electrical Engineering
University College London

`http://icarus-sim.github.io`

Outline

- Background and motivation
  - Information Centric Networking (ICN)
  - Evaluating caching performance
- Icarus simulator
  - Architecture and design
  - Modelling tools
  - Performance evaluation
- Summary and conclusions

Information Centric Networking (ICN)

Information Centric Networking (ICN)

ICN is a recently proposed networking paradigm proposing a shift of the main network abstraction from node identifiers to location-agnostic content identifiers.

Information Centric Networking (ICN)

ICN is a recently proposed networking paradigm proposing a shift of the main network abstraction from node identifiers to location-agnostic content identifiers.

Several implementations proposed so far: CCN/NDN, NetInf, PSIRP/PURSUIT, COMET, MobilityFirst

Information Centric Networking (ICN)

ICN is a recently proposed networking paradigm proposing a shift of the main network abstraction from node identifiers to location-agnostic content identifiers.

Several implementations proposed so far: CCN/NDN, NetInf, PSIRP/PURSUIT, COMET, MobilityFirst

Main principles:

- Request-response model
- Location-agnostic content addressing
- Secure the content, not the channel
- In-network caching

Overlay vs. In-Network Caching

Important to understand:

"What are the differences between overlay and in-network caching?"

Overlay vs. In-Network Caching

Important to understand:

"What are the differences between overlay and in-network caching?"

- Caching at the chunk-level **not** at the file-level (probably **not** at the packet level either)
  - As contents pass through router-caches they replace existing "old" contents
  - Caching can happen transparently into the network at random or predefined (rendezvous) points

Overlay vs. In-Network Caching

Important to understand:

"What are the differences between overlay and in-network caching?"

- Caching at the chunk-level **not** at the file-level (probably **not** at the packet level either)
  - As contents pass through router-caches they replace existing "old" contents
  - Caching can happen transparently into the network at random or predefined (rendezvous) points
- Replacement happens at line-speed – what does this imply?
  - Overlay caching depends on centralised (control-plane) co-ordination and management of caches (or de-centralised among very few nodes) – In-network caching does not.

Overlay vs. In-Network Caching

Important to understand:

"What are the differences between overlay and in-network caching?"

- Caching at the chunk-level **not** at the file-level (probably **not** at the packet level either)
  - As contents pass through router-caches they replace existing "old" contents
  - Caching can happen transparently into the network at random or predefined (rendezvous) points
- Replacement happens at line-speed – what does this imply?
  - Overlay caching depends on centralised (control-plane) co-ordination and management of caches (or de-centralised among very few nodes) – In-network caching does not.
- Hence: no book-keeping possible
  - Impossible to co-ordinate with other caches, or the control plane – the exact location of contents cannot be known
  - Caching operations happen transparently inside the network
  - Decentralized distribution and replacement of contents in caches

Evaluating Caching Performance

Evaluating Caching Performance

Requirements:

Evaluating Caching Performance

Requirements:

- Large realistic topologies

Evaluating Caching Performance

Requirements:

- Large realistic topologies
- Many content requests to allow caches to reach steady-state

Evaluating Caching Performance

Requirements:

- Large realistic topologies
- Many content requests to allow caches to reach steady-state
- Trace-driven simulations if possible

Evaluating Caching Performance

Requirements:

- Large realistic topologies
- Many content requests to allow caches to reach steady-state
- Trace-driven simulations if possible

Many simulators and prototypes are available today for evaluating ICN designs but none are suitable for caching:

Evaluating Caching Performance

Requirements:

- Large realistic topologies
- Many content requests to allow caches to reach steady-state
- Trace-driven simulations if possible

Many simulators and prototypes are available today for evaluating ICN designs but none are suitable for caching:

- Bound to a specific architecture

Evaluating Caching Performance

Requirements:

- Large realistic topologies
- Many content requests to allow caches to reach steady-state
- Trace-driven simulations if possible

Many simulators and prototypes are available today for evaluating ICN designs but none are suitable for caching:

- Bound to a specific architecture
- Poor scalability

Evaluating Caching Performance

Requirements:

- Large realistic topologies
- Many content requests to allow caches to reach steady-state
- Trace-driven simulations if possible

Many simulators and prototypes are available today for evaluating ICN designs but none are suitable for caching:

- Bound to a specific architecture
- Poor scalability
- Inability to run trace-driven simulations

Evaluating Caching Performance

Requirements:

- Large realistic topologies
- Many content requests to allow caches to reach steady-state
- Trace-driven simulations if possible

Many simulators and prototypes are available today for evaluating ICN designs but none are suitable for caching:

- Bound to a specific architecture
- Poor scalability
- Inability to run trace-driven simulations

Scarce availability of open-source implementations of modelling tools for network caching research.

Icarus simulator

Icarus simulator

Python-based discrete-event simulator designed for evaluating the performance of:

- Caching and routing strategies
- Cache replacement policies

Icarus simulator

Python-based discrete-event simulator designed for evaluating the performance of:

- Caching and routing strategies
- Cache replacement policies

Non-functional requirements:

- Extensibility
- Scalability

Achieving extensibility

Achieving extensibility

- Plug-in registration system and extensive use of bridge pattern to provide loose-coupling

```
@register_cache_policy('FOO')
class FooCache(Cache)

    def get(self, k):
        ...

    def put(self, k):
        ...
```

```
# config
    .
    .
POLICIES = [‘LRU‘, 'FOO‘]
    .
    .
    .
```

Achieving extensibility

- Plug-in registration system and extensive use of bridge pattern to provide loose-coupling
- Support for `fnss` and `networkx` tools

```
@register_cache_policy('FOO')
class FooCache(Cache)

    def get(self, k):
        ...

    def put(self, k):
        ...
```

```
# config
    .
    .
POLICIES = ['LRU', 'FOO']
    .
    .
    .
```

Achieving scalability

Achieving scalability

- Flow-level abstraction

Achieving scalability

- Flow-level abstraction
- Parallel execution of experiments

Achieving scalability

- Flow-level abstraction
- Parallel execution of experiments
- Minimized disk access during experiment execution

Architecture and design

Architecture and design

Code organized in four loosely-coupled subsystems:

Architecture and design

Code organized in four loosely-coupled subsystems:

- Orchestration

Architecture and design

Code organized in four loosely-coupled subsystems:

- Orchestration
- Scenario generation

Architecture and design

Code organized in four loosely-coupled subsystems:

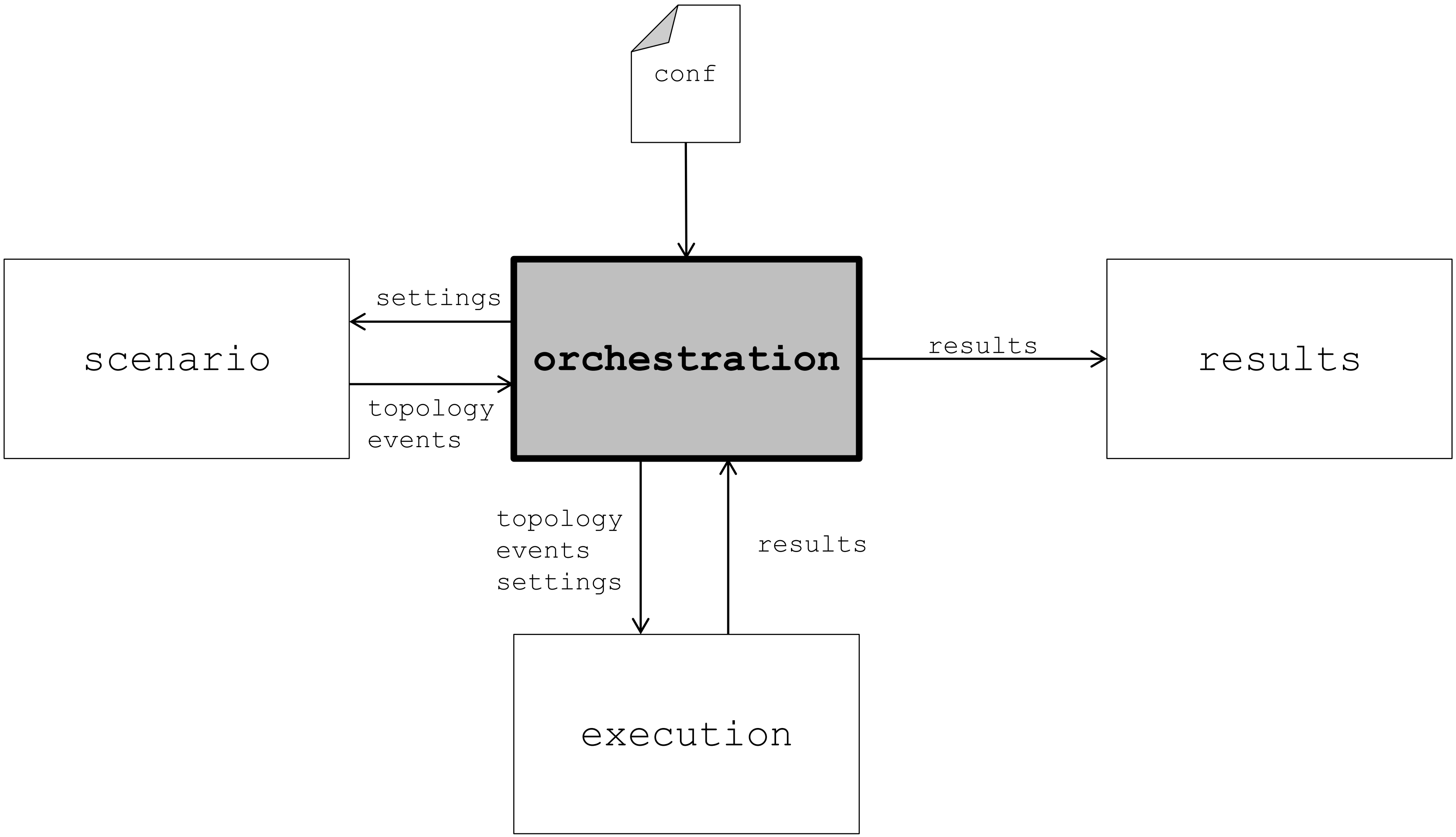
- Orchestration
- Scenario generation
- Execution

Architecture and design

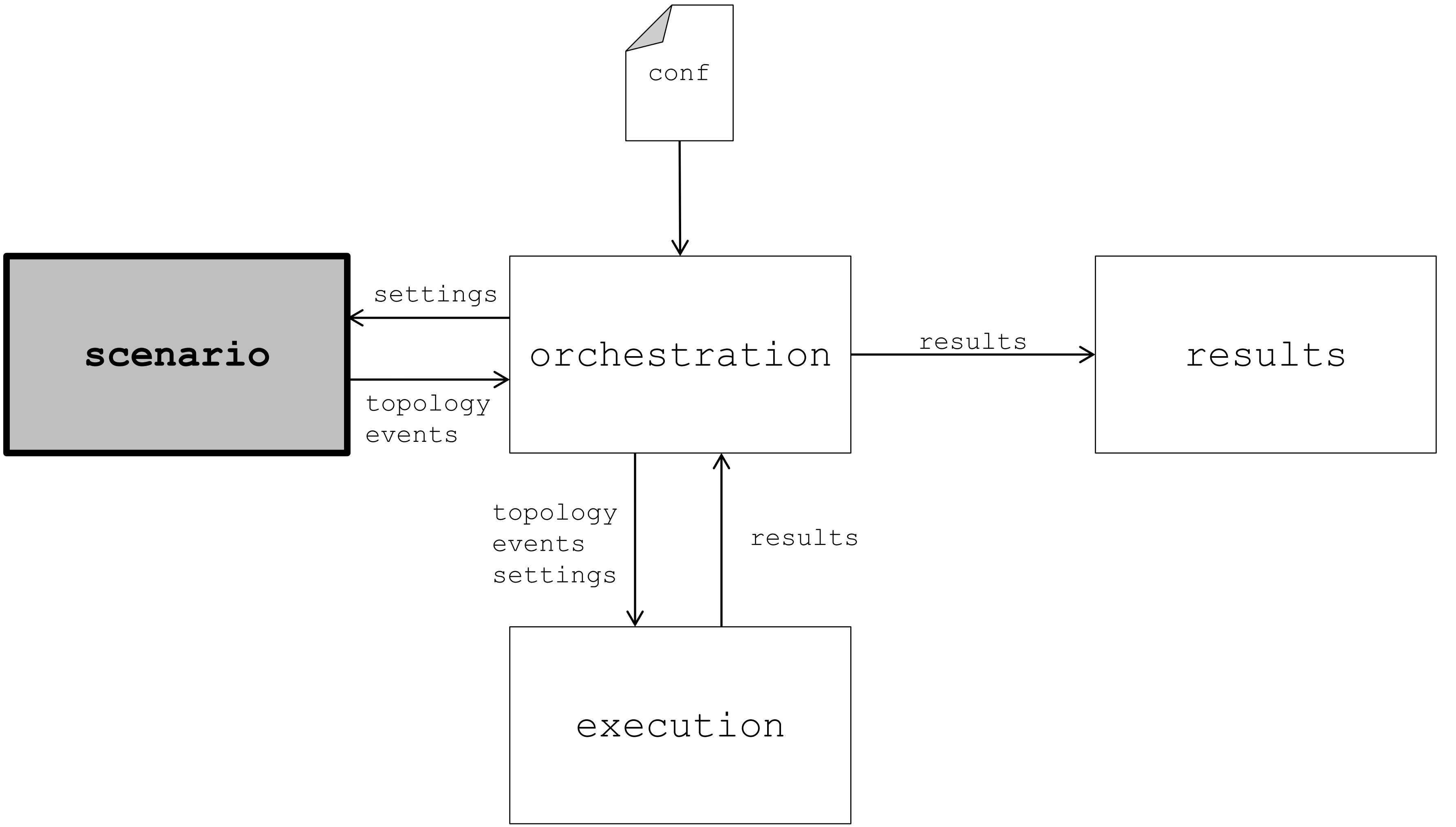
Code organized in four loosely-coupled subsystems:

- Orchestration
- Scenario generation
- Execution
- Results collection and analysis

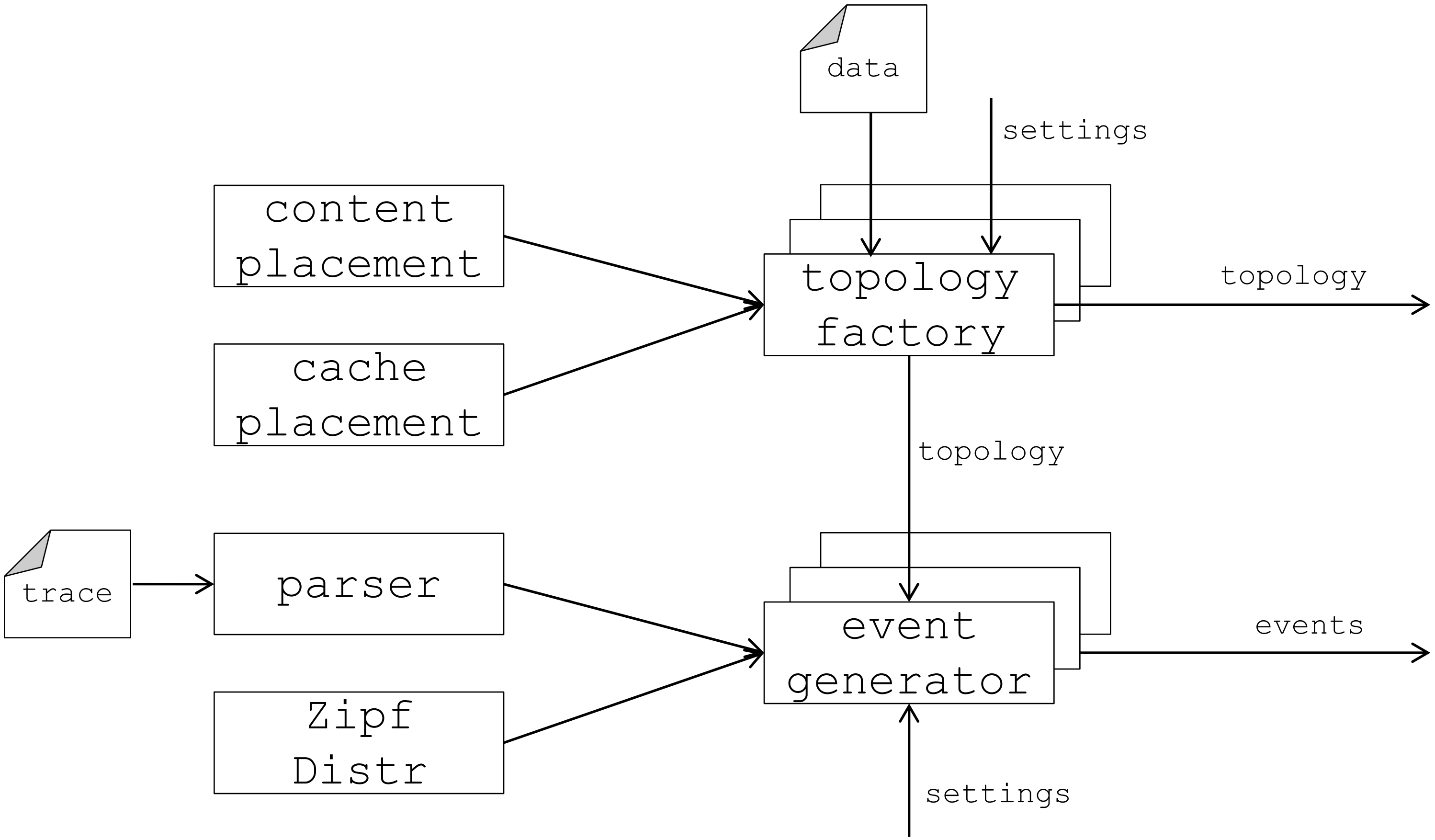
Orchestration



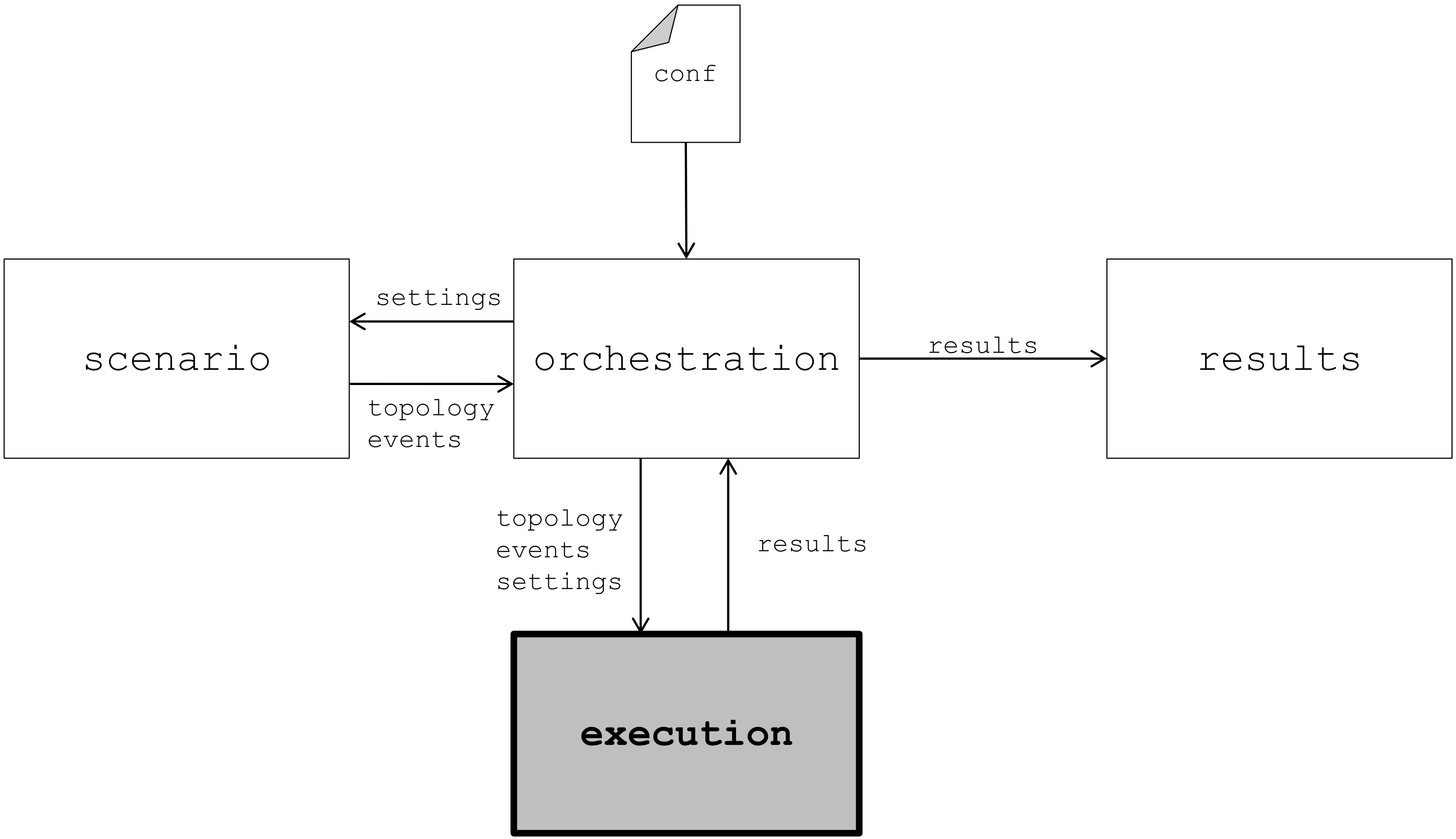
Scenario generation



Scenario generation

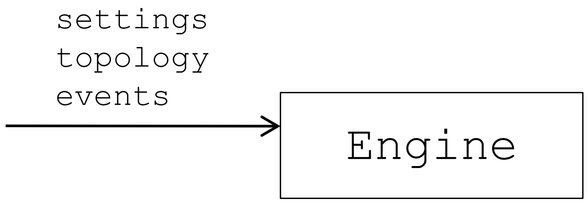


Execution

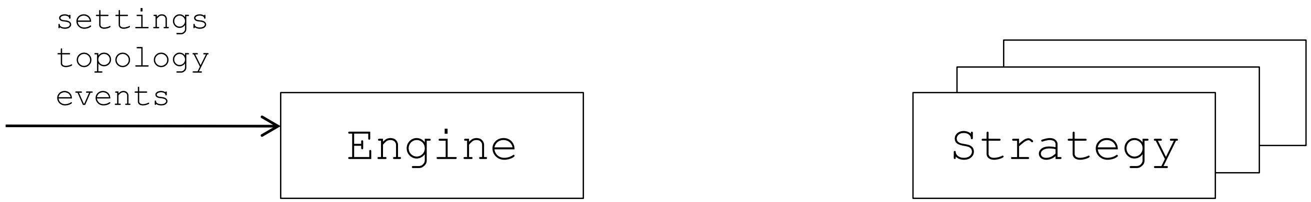

conf
scenario
settings
topology
events
orchestration
results
results
topology
events
settings
results
execution

Execution

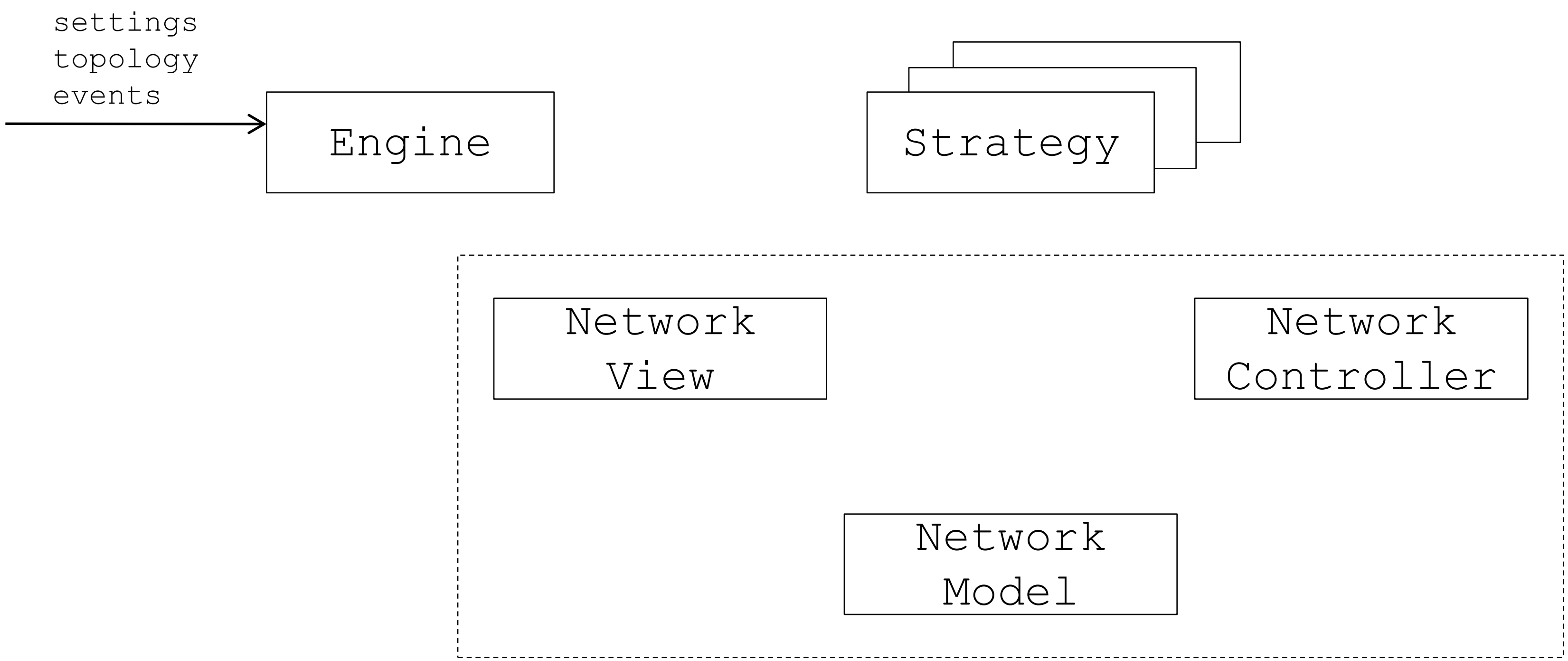
Execution



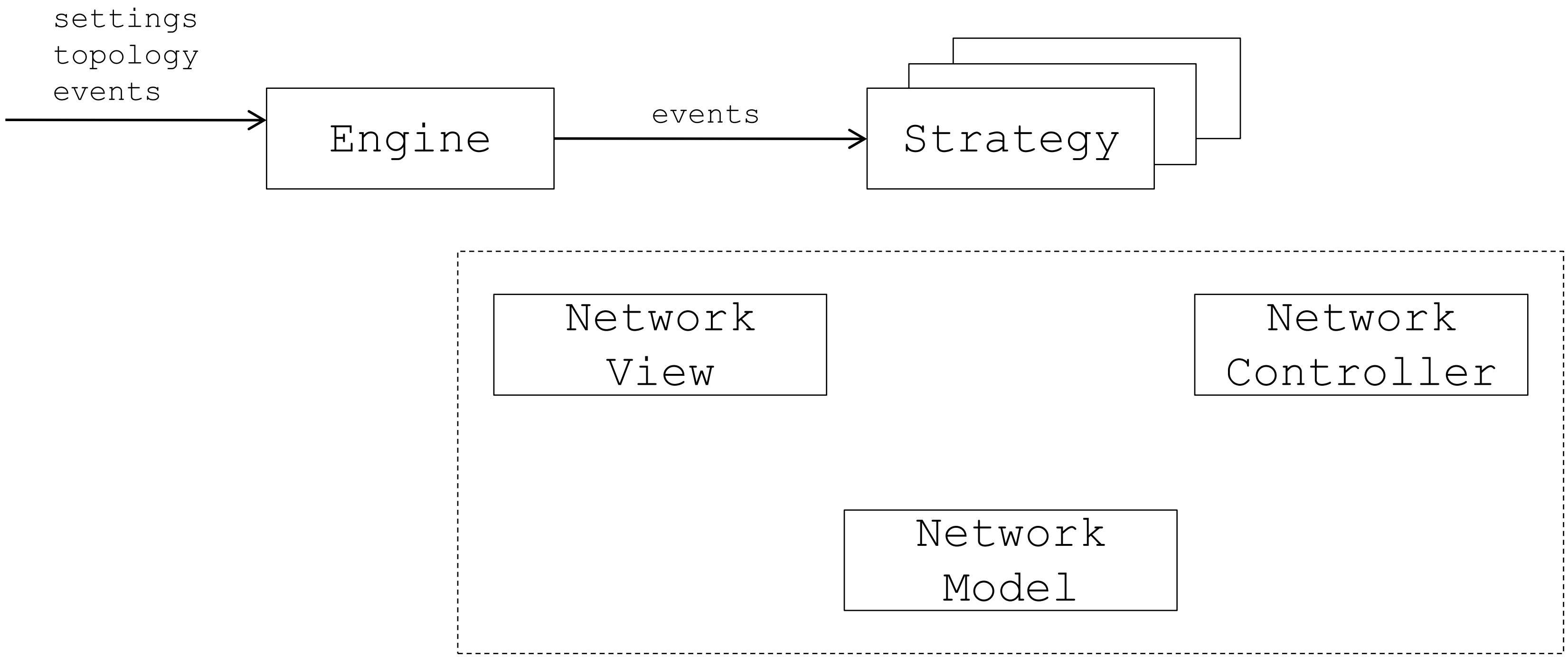
Execution



Execution

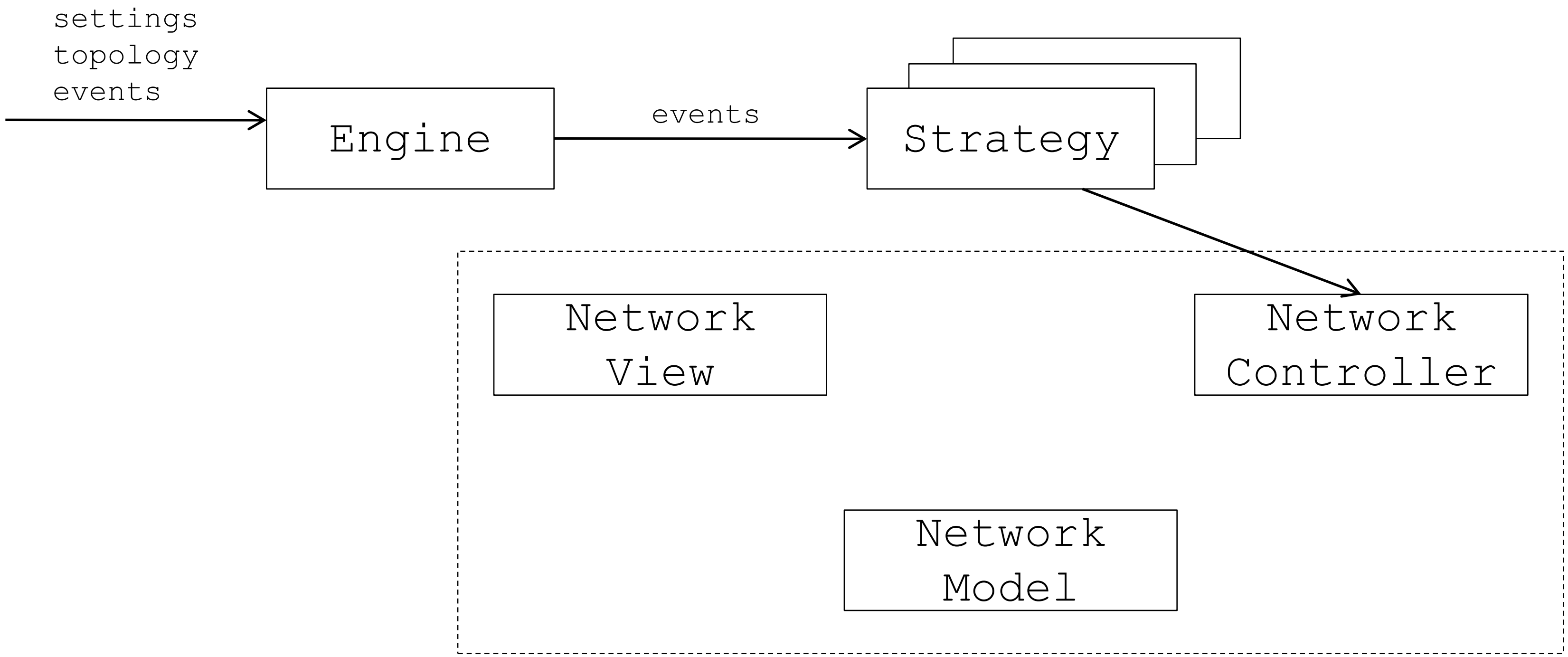

settings
topology
events
Engine
Strategy
Network View
Network Controller
Network Model

Execution



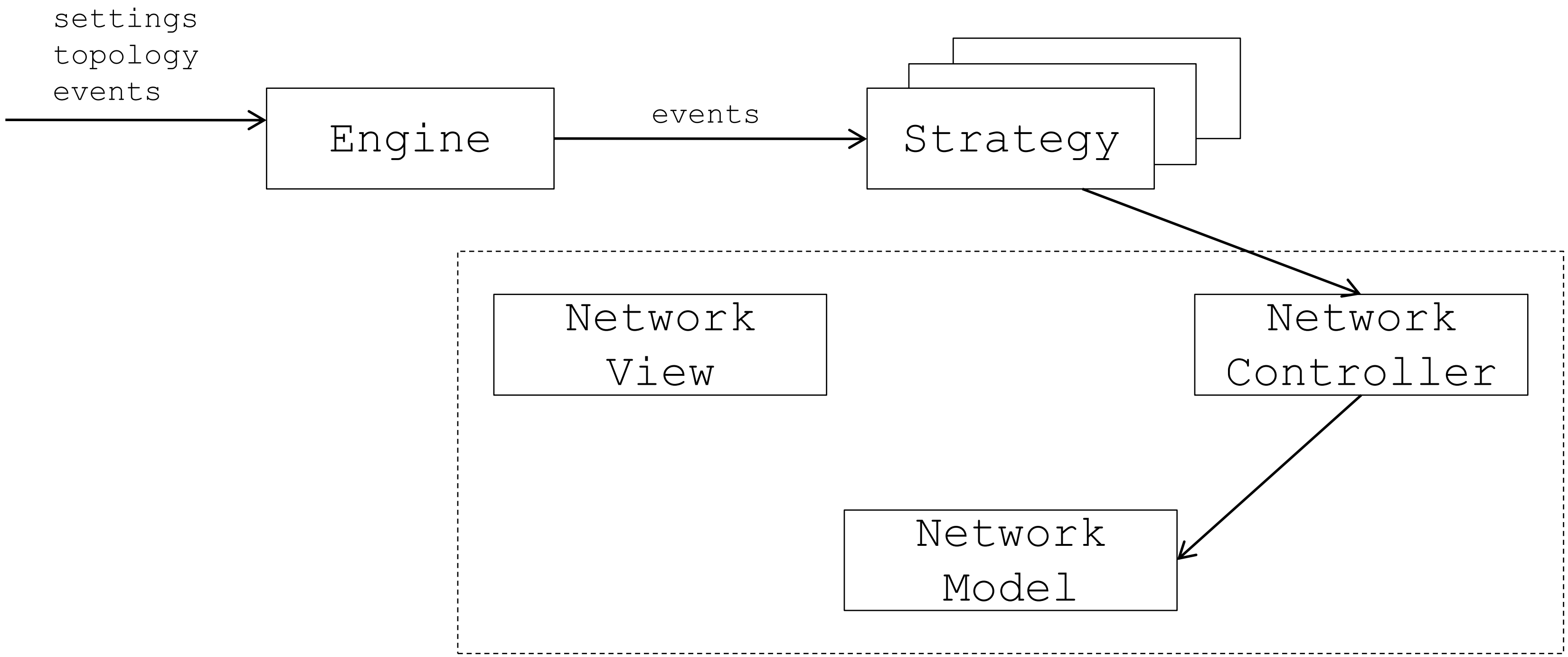
settings
topology
events
Engine
events
Strategy
Network View
Network Controller
Network Model

Execution

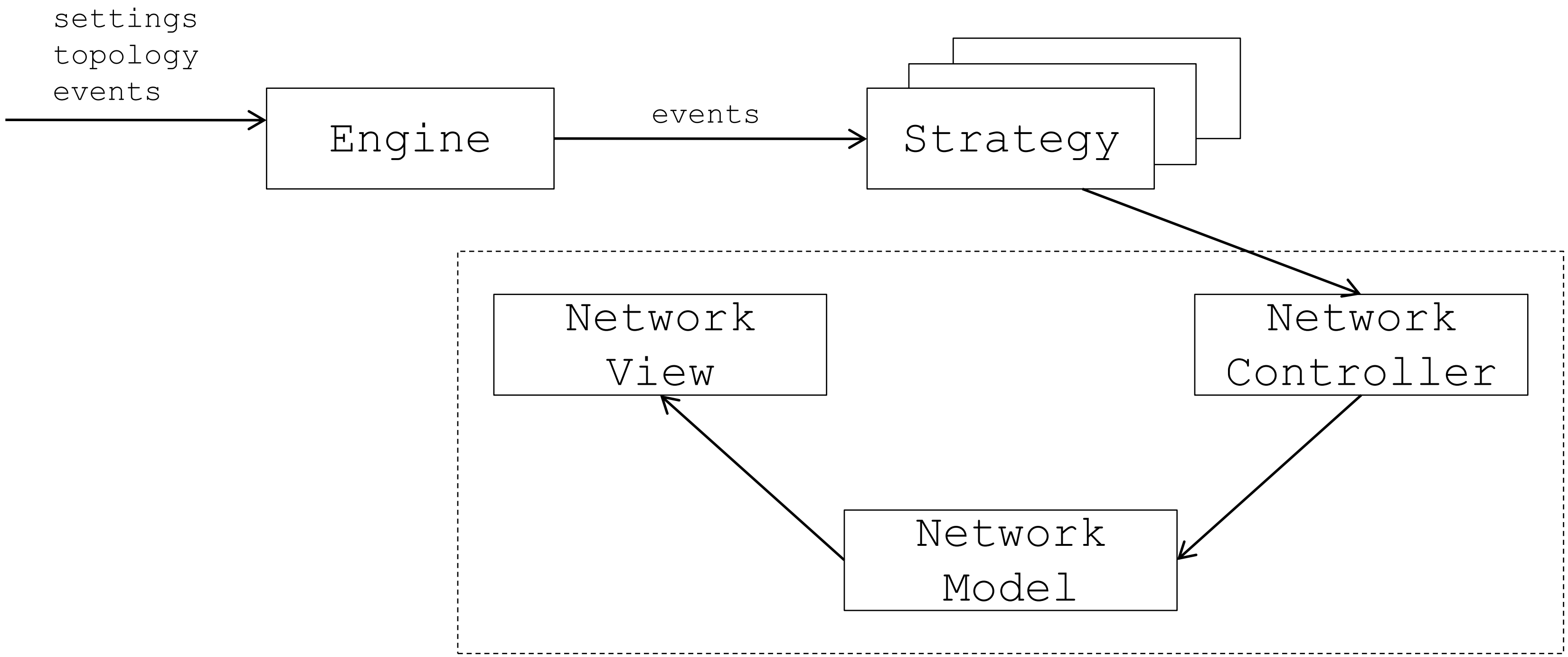


settings
topology
events
Engine
events
Strategy
Network View
Network Controller
Network Model

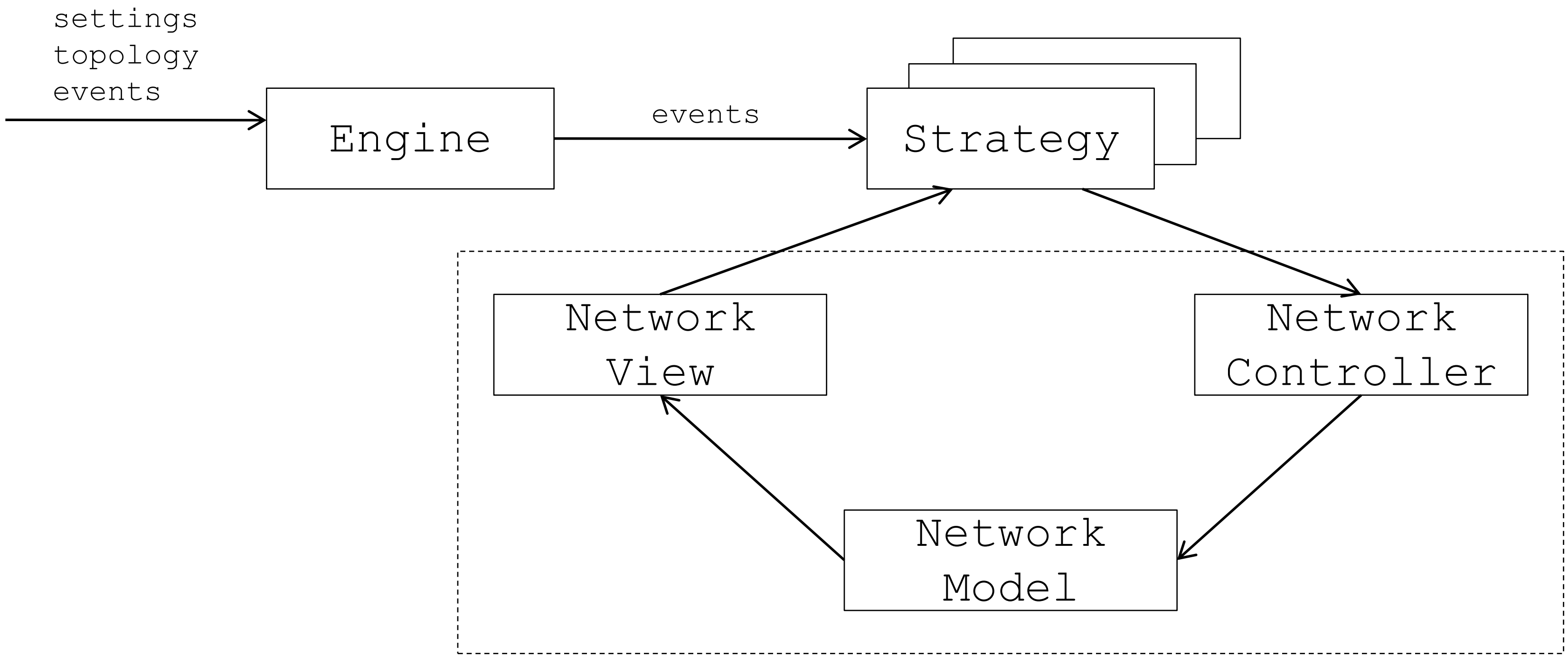
Execution


settings
topology
events
Engine
events
Strategy
Network
View
Network
Controller
Network
Model

Execution

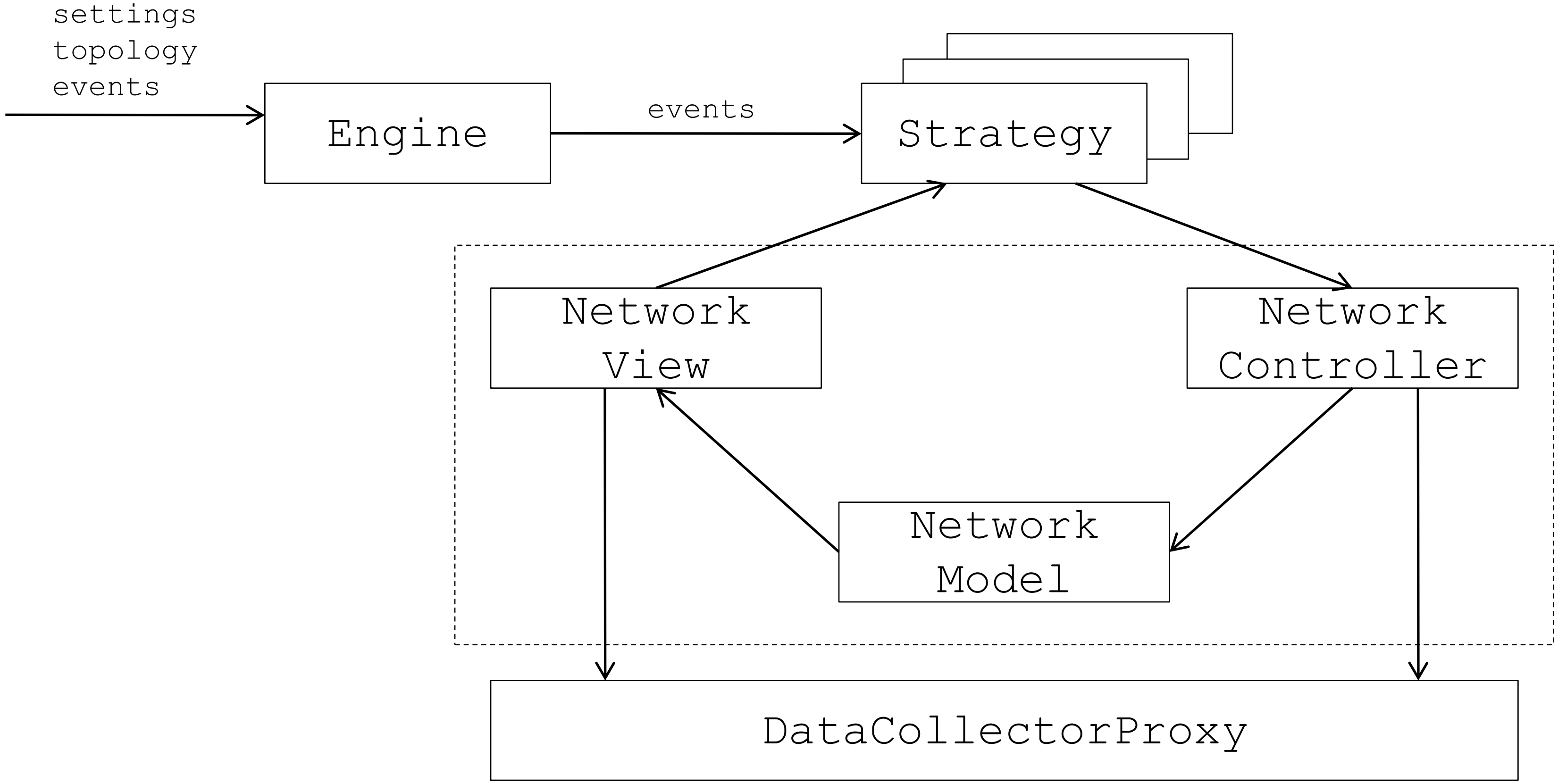

settings
topology
events
Engine
events
Strategy
Network
View
Network
Controller
Network
Model

Execution

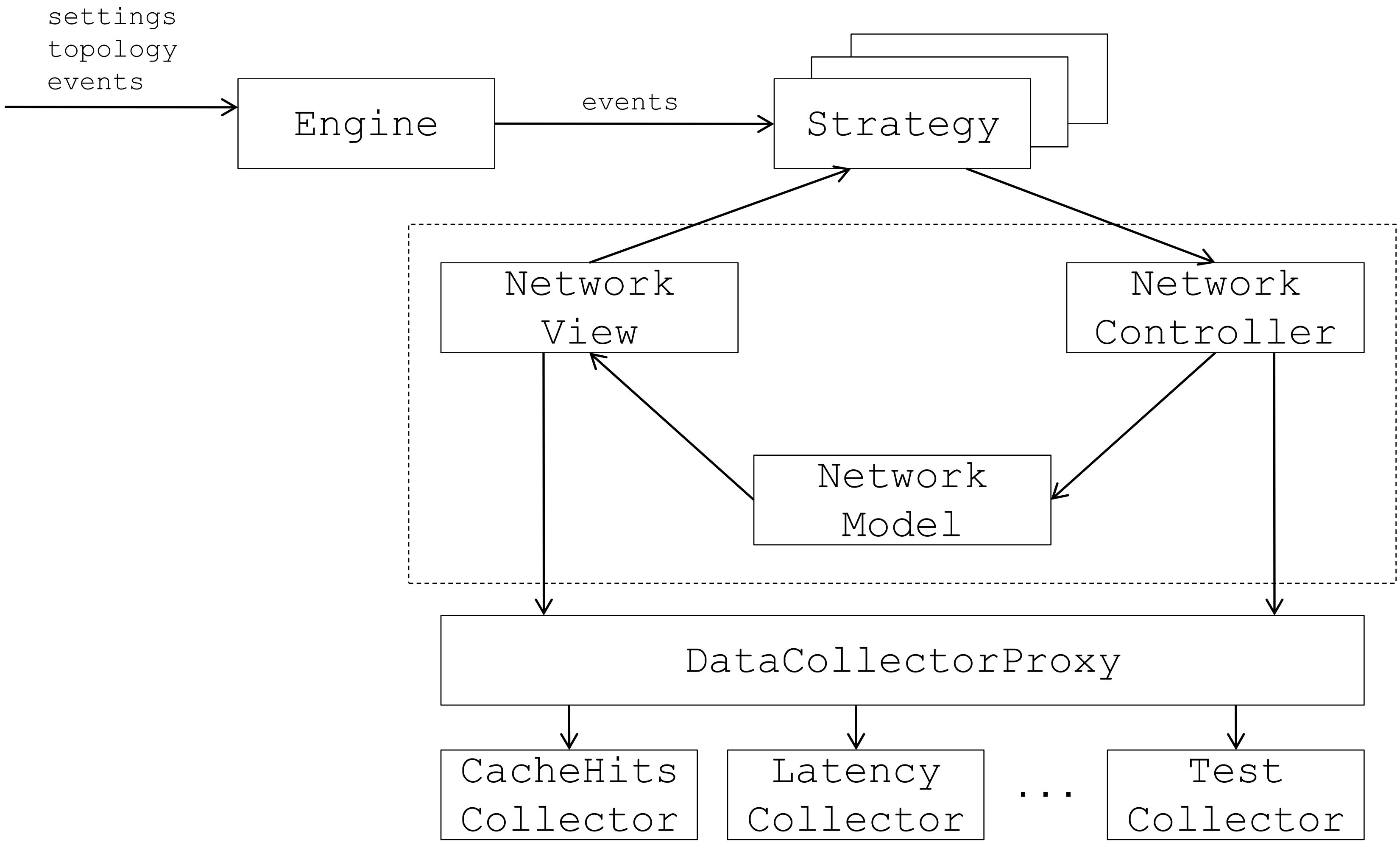


settings
topology
events
Engine
events
Strategy
Network View
Network Controller
Network Model

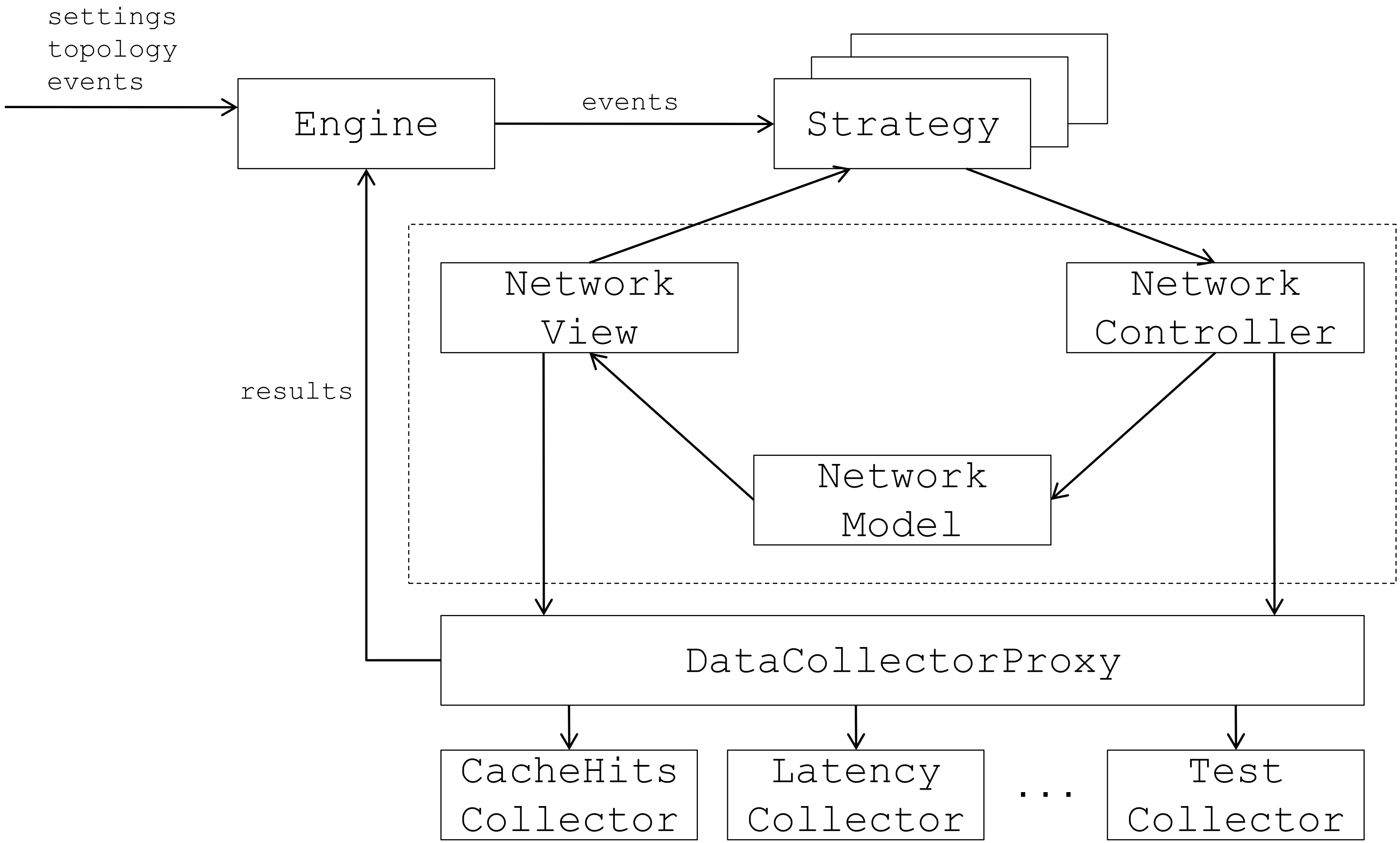
Execution


settings
topology
events
Engine
events
Strategy
Network View
Network Controller
Network Model
DataCollectorProxy

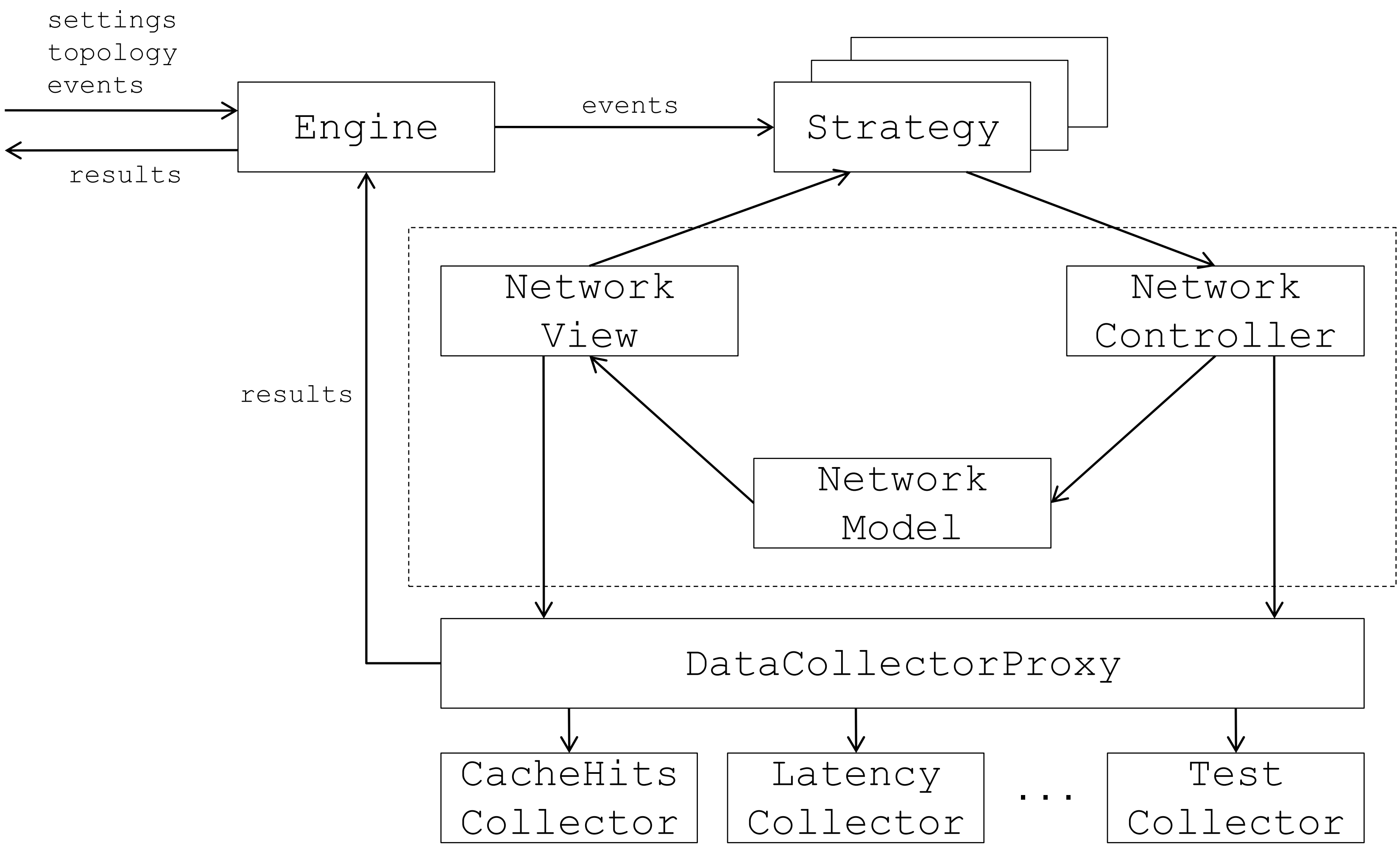
Execution


settings
topology
events
Engine
events
Strategy
Network View
Network Controller
Network Model
DataCollectorProxy
CacheHits Collector
Latency Collector
...
Test Collector

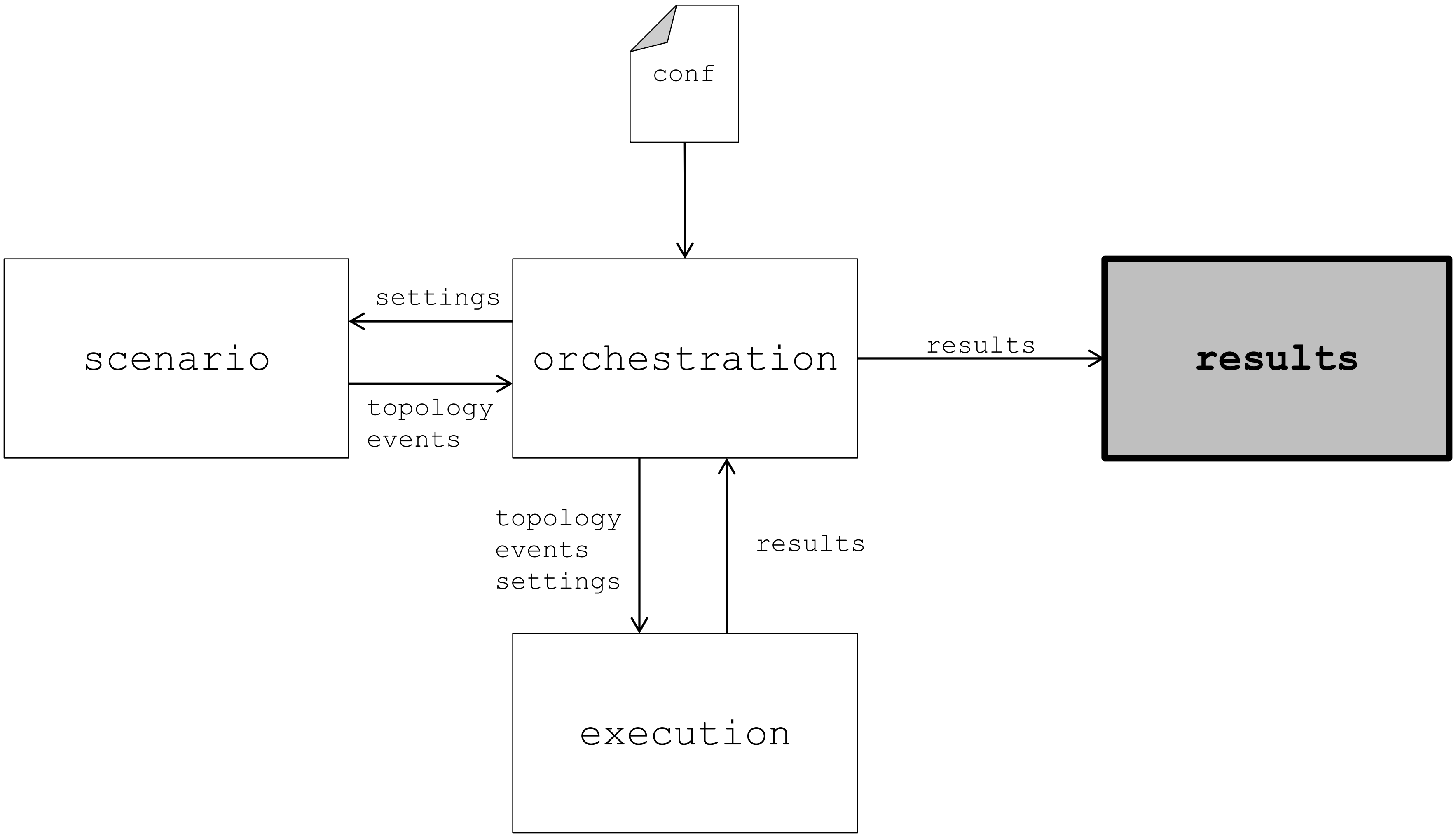
Execution


settings
topology
events
Engine
events
Strategy
results
Network View
Network Controller
Network Model
DataCollectorProxy
CacheHits Collector
Latency Collector
...
Test Collector

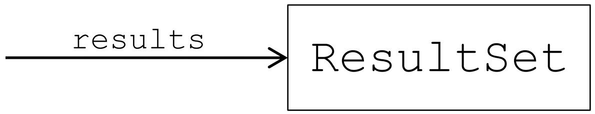
Execution


settings
topology
events
results
Engine
events
Strategy
Network View
Network Controller
results
Network Model
DataCollectorProxy
CacheHits Collector
Latency Collector
...
Test Collector

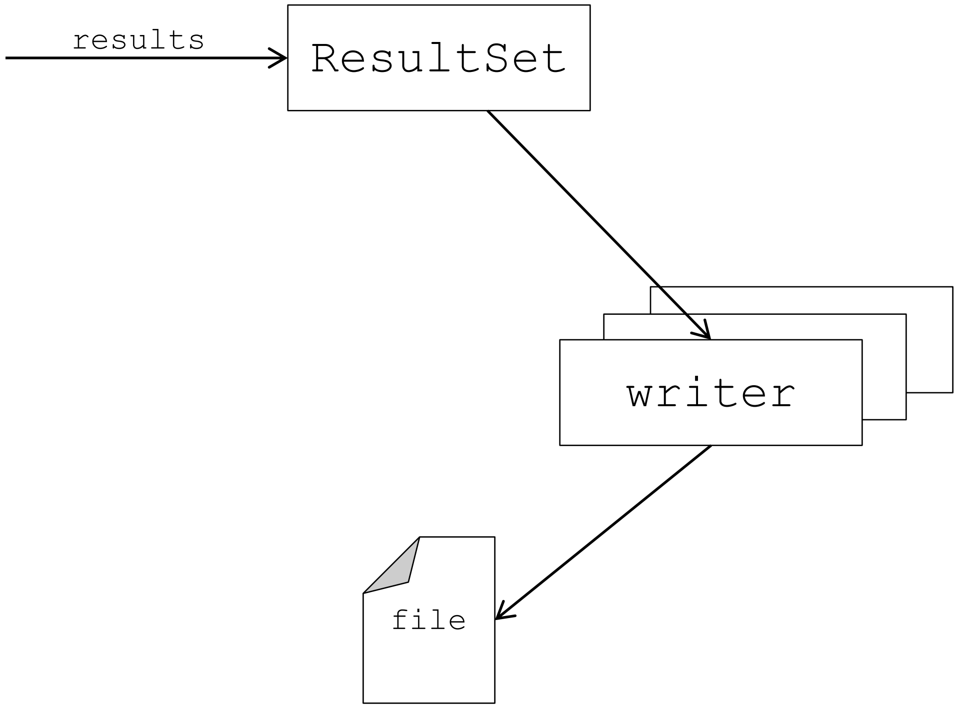
Results collection and analysis



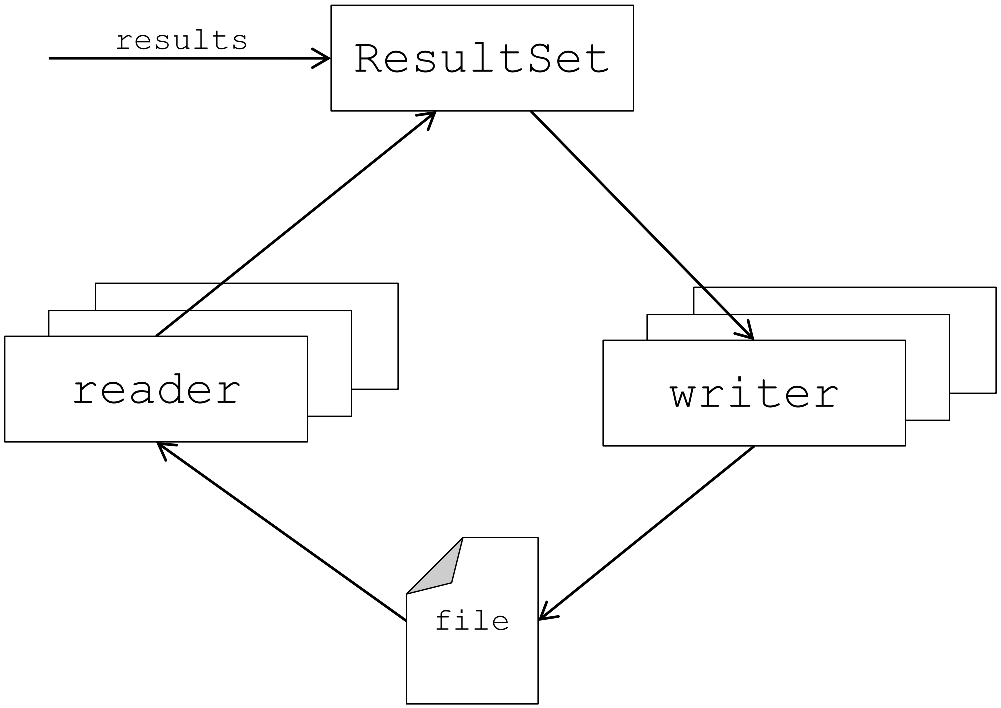
Results collection and analysis



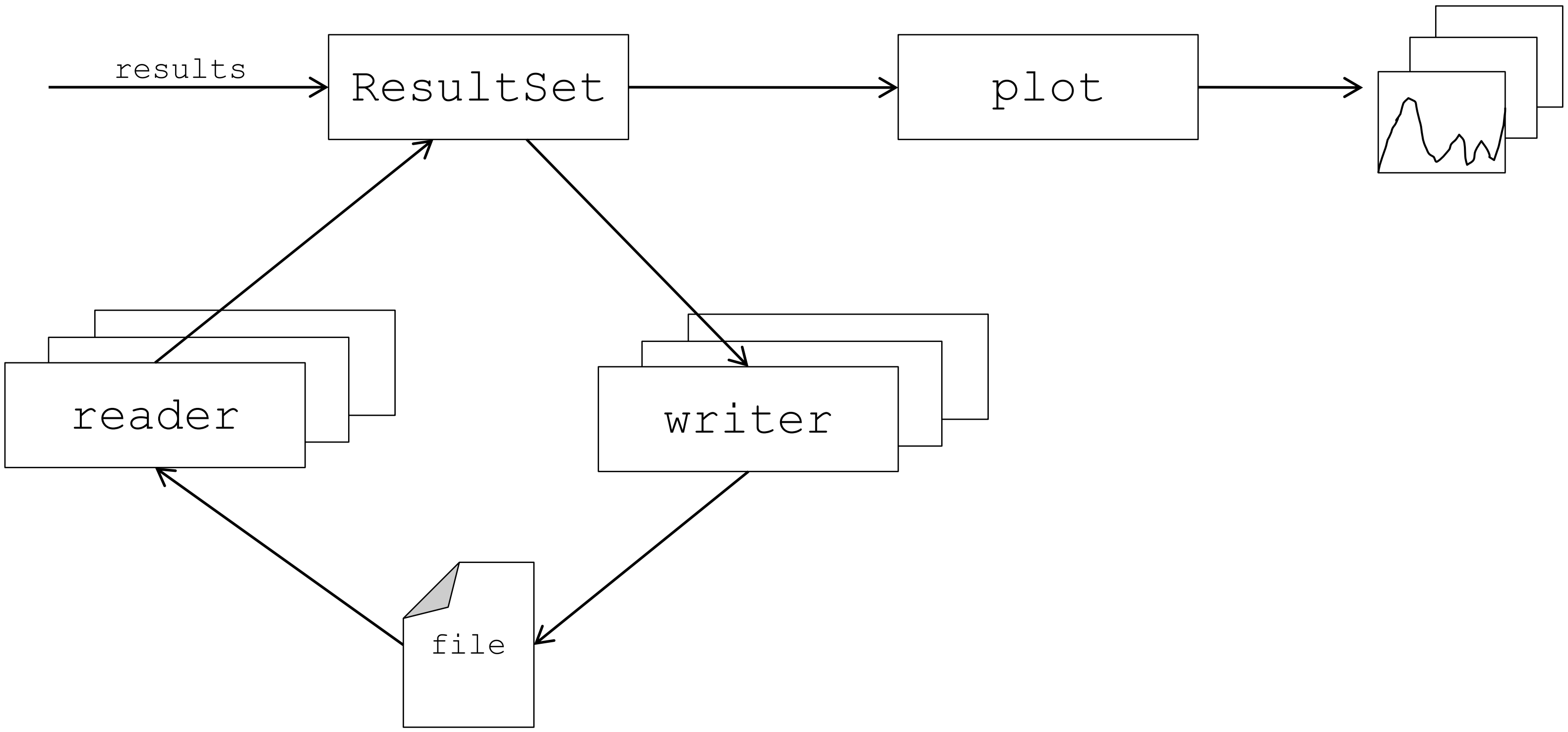
Results collection and analysis



Results collection and analysis



Results collection and analysis



Modelling tools

Cache performance

Workloads

Modelling tools

Cache performance

- Che’s approximation

Workloads

```
>>> import icarus as ics
>>> ics.che_cache_hit_ratio(
          ics.TruncatedZipfDist(alpha=0.8, n=1000).pdf,
          100)
0.36482948293429832
```

Modelling tools

Cache performance

- Che's approximation
- Laoutaris' approximation

Workloads

```
>>> import icarus as ics
>>> ics.laoutaris_cache_hit_ratio(0.7, 1000, 100)
0.359348209359255
```

Modelling tools

Cache performance

- Che’s approximation
- Laoutaris’ approximation
- Optimal hit ratio

Workloads

```
>>> import icarus as ics
>>> ics.optimal_cache_hit_ratio(
          ics.TruncatedZipfDist(alpha=0.8, n=1000).pdf,
          100)
0.52582651157679017
```

Modelling tools

Cache performance

- Che's approximation
- Laoutaris' approximation
- Optimal hit ratio
- Numeric hit ratio

Workloads

```
>>> import icarus as ics
>>> ics.numeric_cache_hit_ratio(
          ics.TruncatedZipfDist(alpha=0.8, n=1000).pdf,
          ics.LruCache(100))
0.37861264056574684
```

Modelling tools

Cache performance

- Che’s approximation
- Laoutaris’ approximation
- Optimal hit ratio
- Numerical hit ratio

Workloads

- Zipf fit

```
>>> import icarus as ics
>>> ics.zipf_fit(ics.TruncatedZipfDist(alpha=0.8, n=1000).pdf)
(0.799999999571759, 1.0)
```

Modelling tools

Cache performance

- Che’s approximation
- Laoutaris’ approximation
- Optimal hit ratio
- Numerical hit ratio

Workloads

- Zipf fit
- Trace parsers

```
>>> import icarus as ics
>>> ics.parse_wikibench('wikibench.txt')
```

Evaluating scalability

Evaluating scalability

Scenario:

- Tree topology
- Zipf-distributed content popularity ($\alpha = 0.7$)
- Constant cache/catalogue ratio: 10%
- 500K requests per experiment

Evaluating scalability

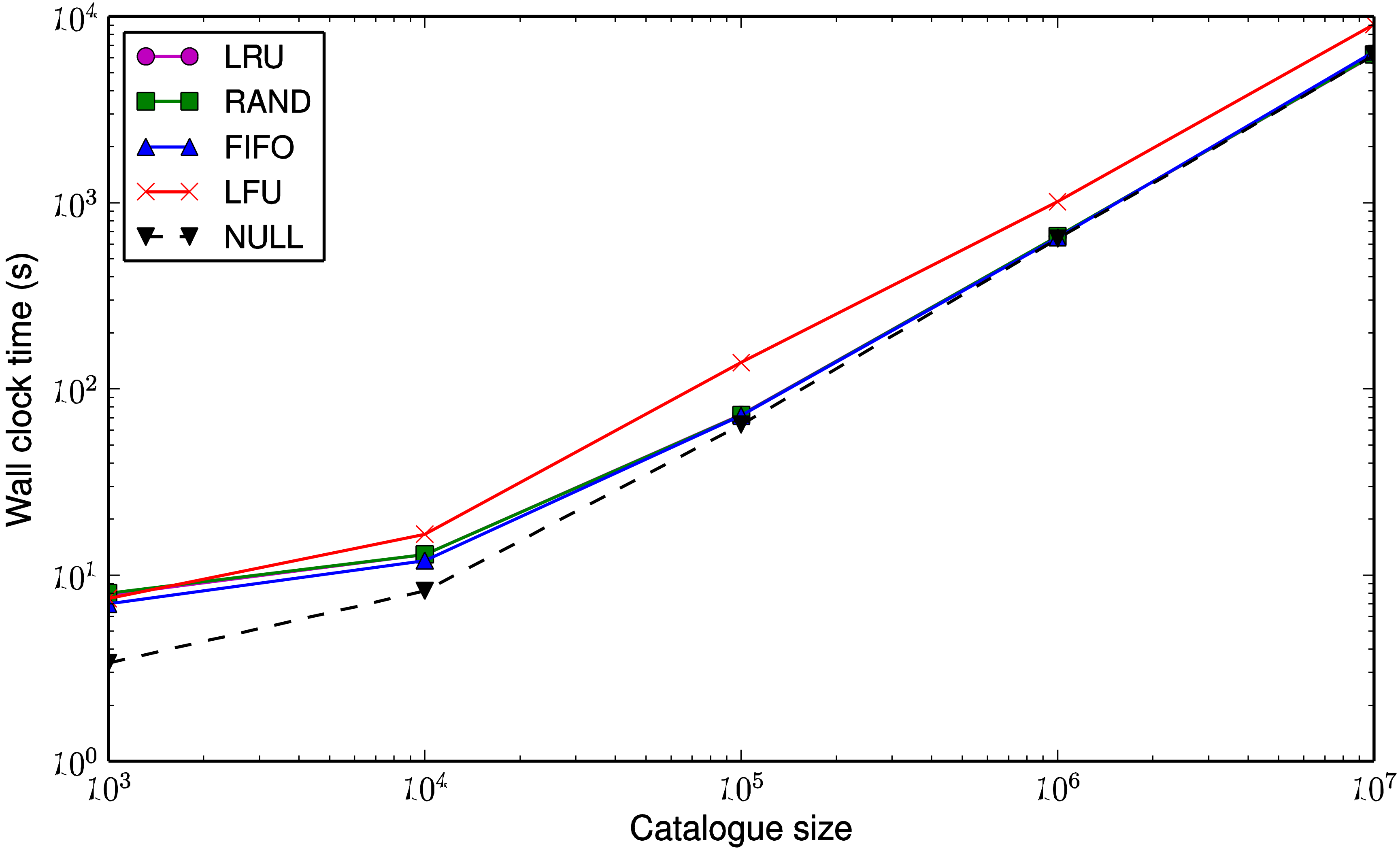
Scenario:

- Tree topology
- Zipf-distributed content popularity ($\alpha = 0.7$)
- Constant cache/catalogue ratio: 10%
- 500K requests per experiment

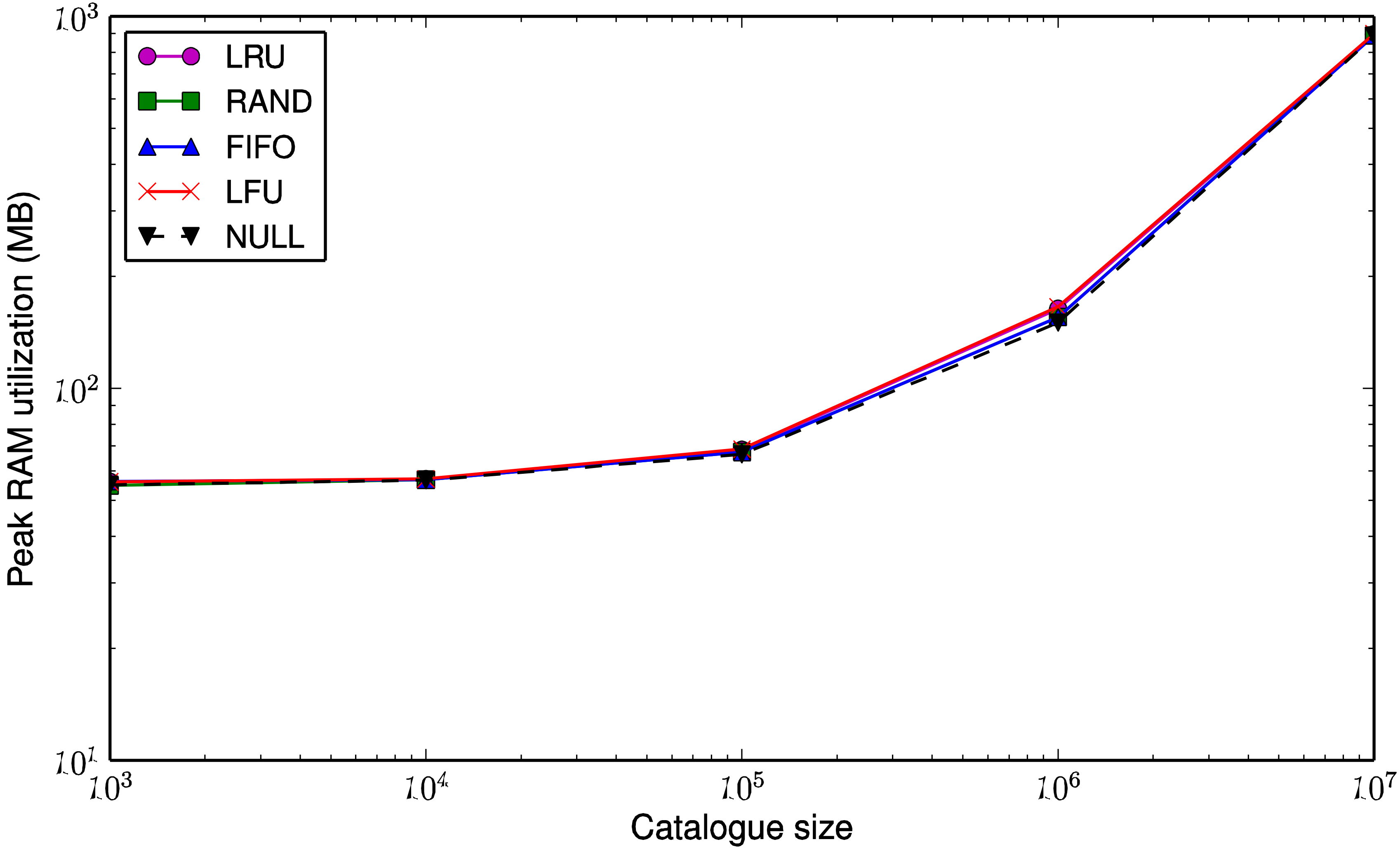
Metrics:

- CPU load and memory utilization vs. content catalogue size

Processing load vs content catalogue size



Memory utilization vs content catalogue size



Summary and conclusions

Summary and conclusions

- We presented Icarus, a caching simulator for Information Centric Networking (ICN)

Summary and conclusions

- We presented Icarus, a caching simulator for Information Centric Networking (ICN)
- Designed for extensibility and scalability

Summary and conclusions

- We presented Icarus, a caching simulator for Information Centric Networking (ICN)
- Designed for extensibility and scalability
- Comprises a set of modelling tools for cache performance and workloads analysis

http://icarus-sim.github.io