AT&T

THE UNIVERSITY OF UTAH

DEPO: A Platform for Safe DEployment of POlicy in a Software Defined Infrastructure

Aisha Syed (University of Utah)
Vijay Gopalakrishnan (AT&T Research)
Bilal Anwer (AT&T Research)
Jacobus Van der Merwe (University of Utah)

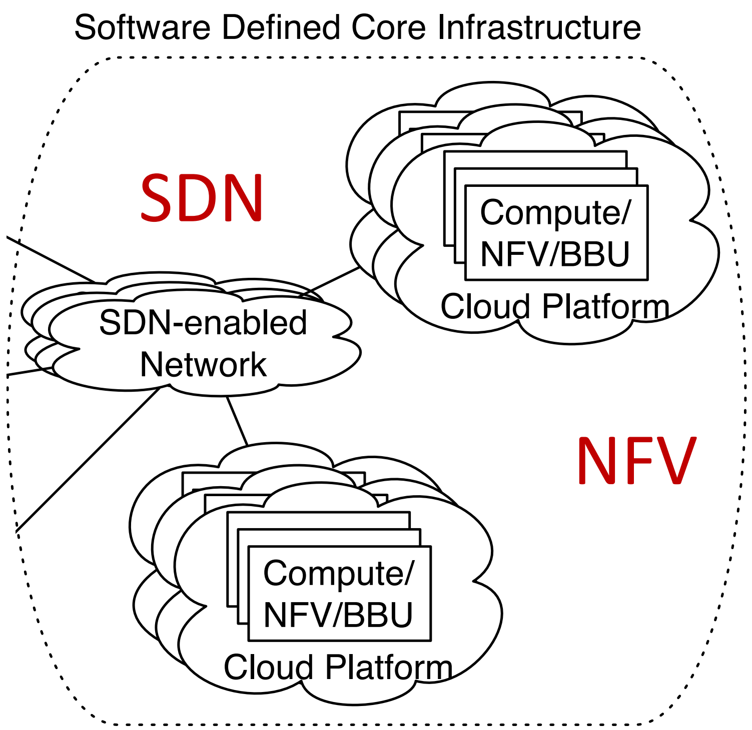
Presented at ACM Symposium on SDN Research (SOSR), April 3-4, 2019, San Jose, CA, USA

Motivation

- With SDN and in-network clouds enabling NFV, we see trend towards a **(mobile) software-defined infrastructure (SDI)**
    - being embraced by telcos, network service providers, equipment vendors

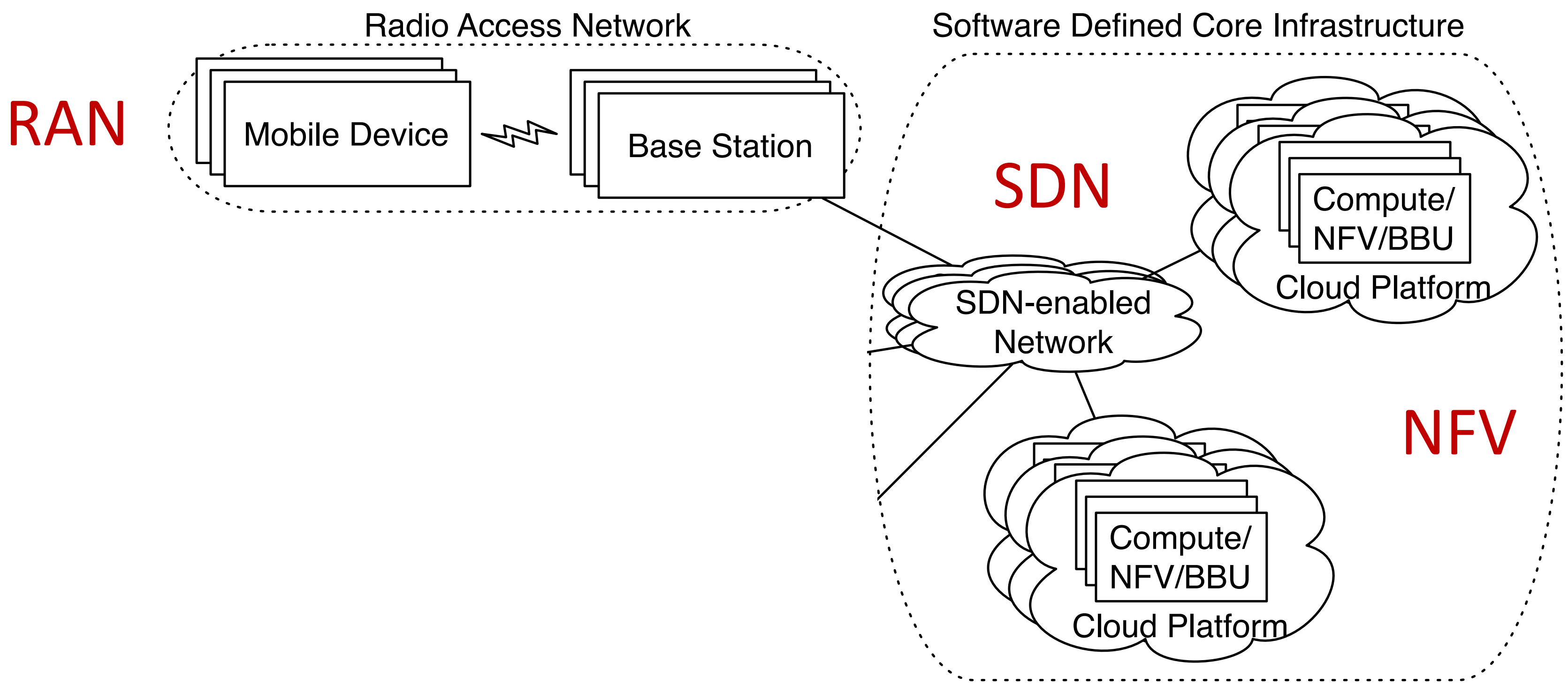
Motivation

- With SDN and in-network clouds enabling NFV, we see trend towards a **(mobile) software-defined infrastructure (SDI)**
  - being embraced by telcos, network service providers, equipment vendors



Motivation

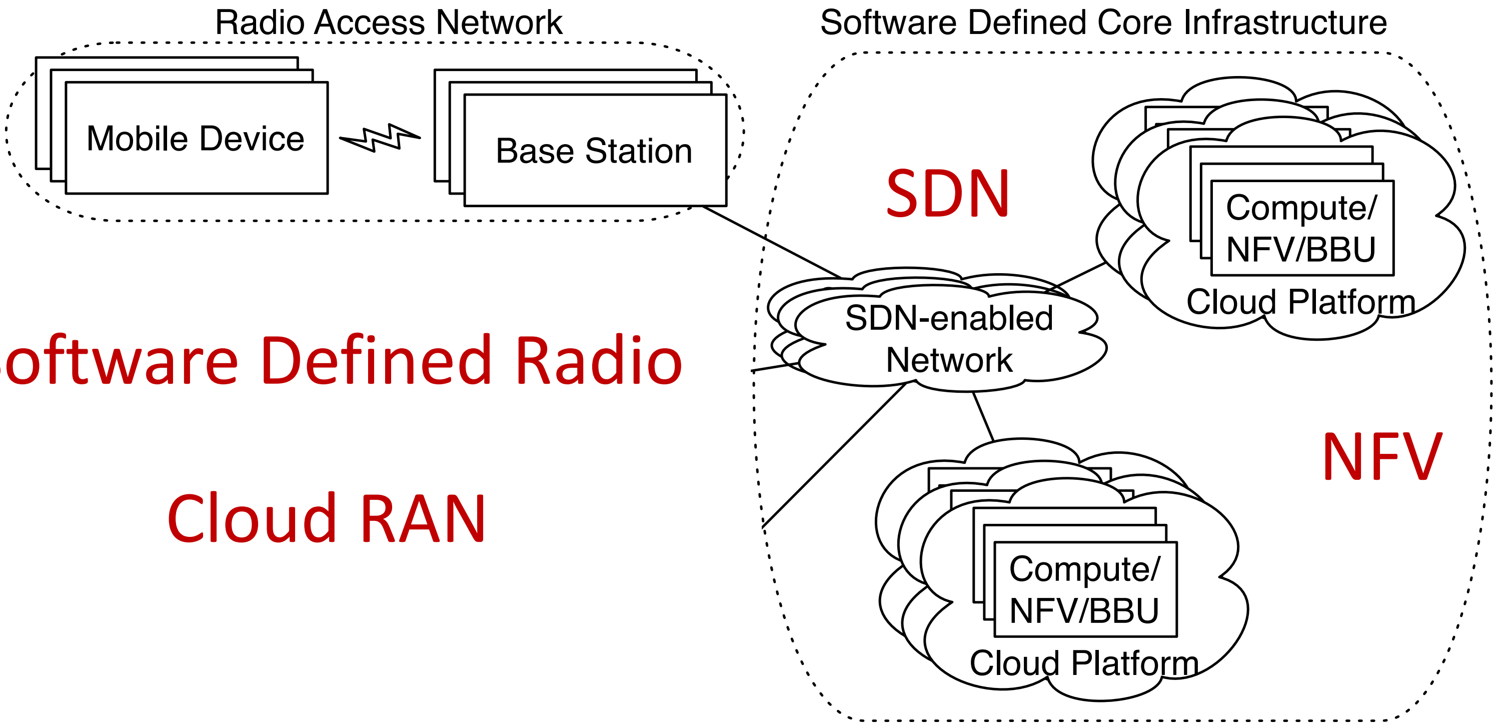
- With SDN and in-network clouds enabling NFV, we see trend towards a **(mobile) software-defined infrastructure (SDI)**
  - being embraced by telcos, network service providers, equipment vendors



Motivation

- With SDN and in-network clouds enabling NFV, we see trend towards a **(mobile) software-defined infrastructure (SDI)**
  - being embraced by telcos, network service providers, equipment vendors

RAN

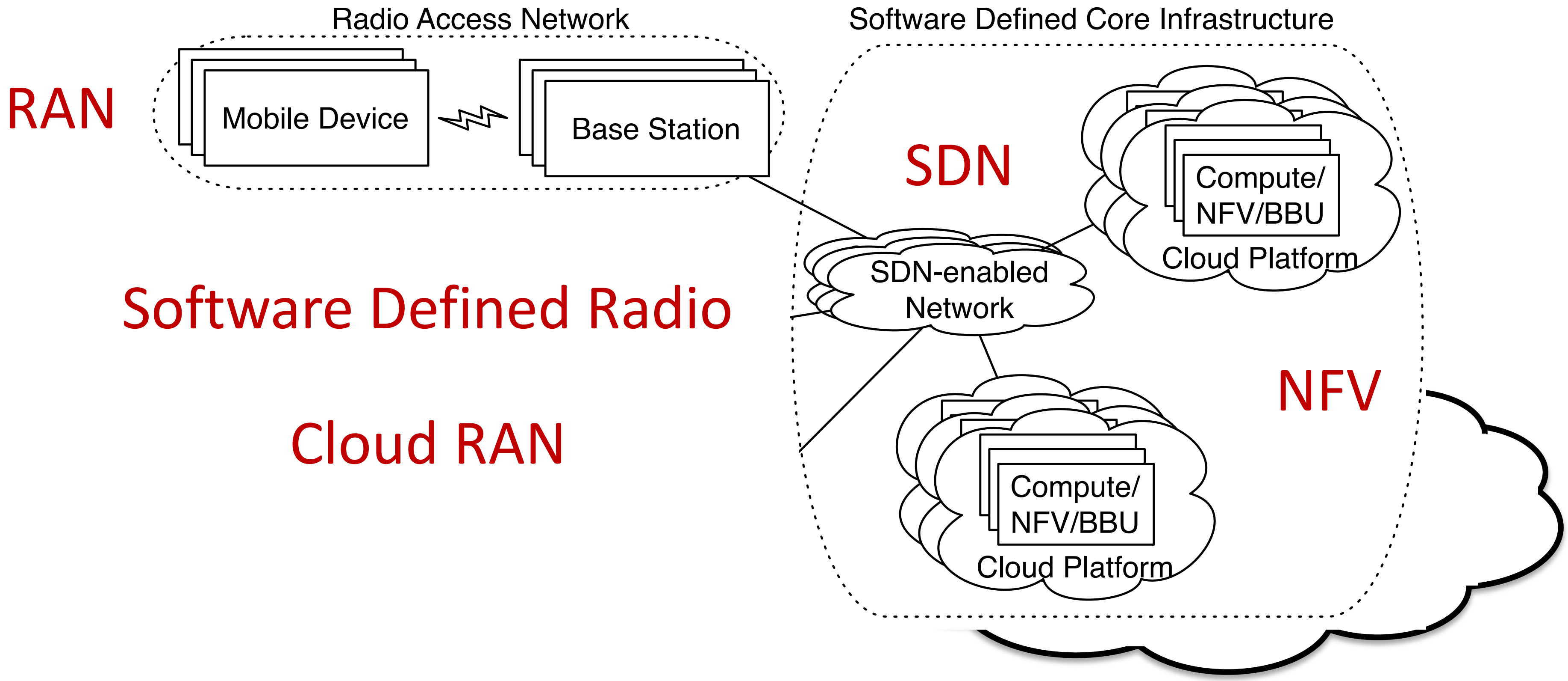


Software Defined Radio

Cloud RAN

Motivation

- Push in industry and academia towards a **(mobile) software-defined infrastructure (SDI)**
  - being embraced by telcos, network service providers, equipment vendors
- With SDN and network cloud enabling SDN/NFV control platforms
  - ONAP, OpenNFV, CORD, etc.
  - Automate network service instantiation and management

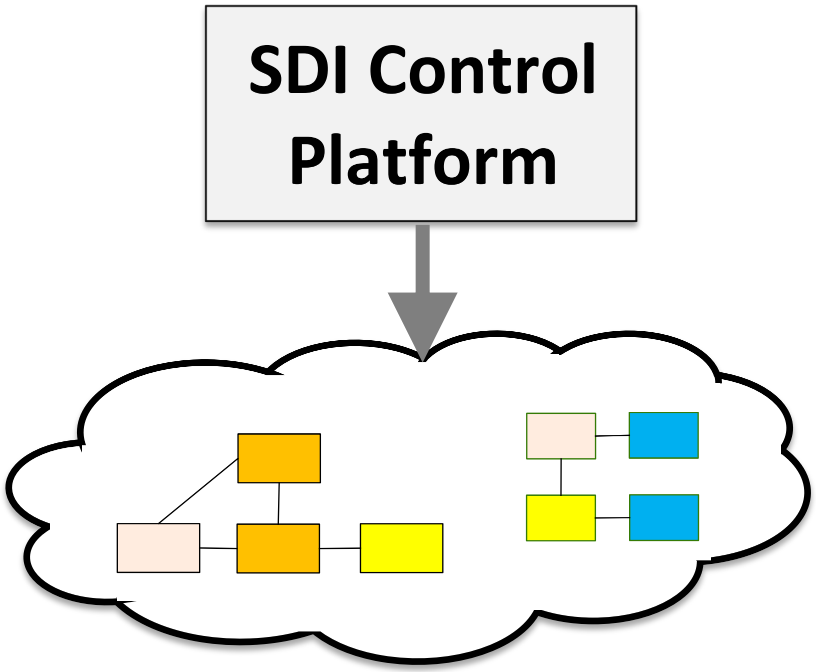


Softv

Motivation

- Push in industry and academia towards SDI control platforms
  - ONAP, OpenNFV, CORD, etc.
  - Automate network service instantiation and management

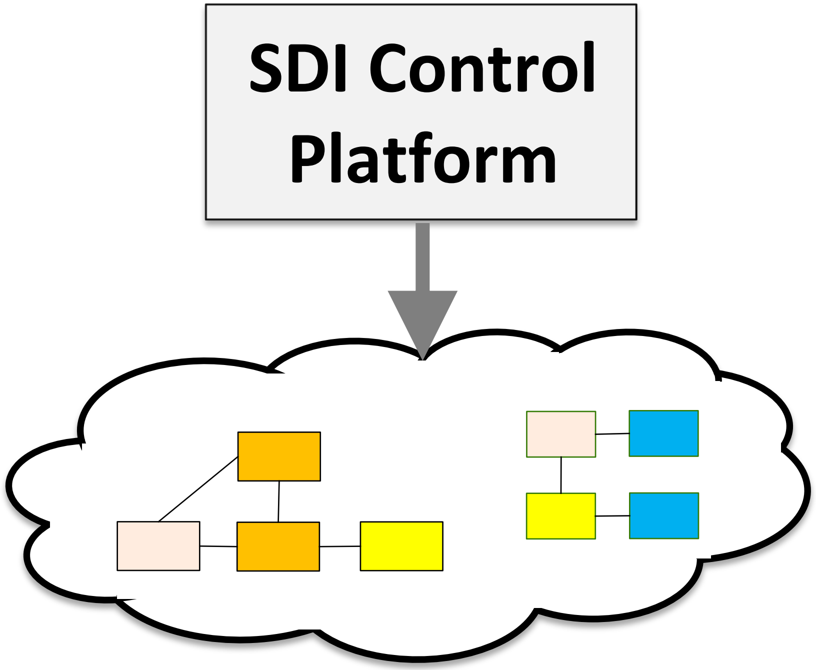
Software Defined Infrastructure (SDI)



Motivation

- Push in industry and academia towards SDI control platforms
  - ONAP, OpenNFV, CORD, etc.
  - Automate network service instantiation and management

- Automation expected to be done through **policies** *(condition → action)*
  - Unintended consequences
  - Performance degradation, SLA violations



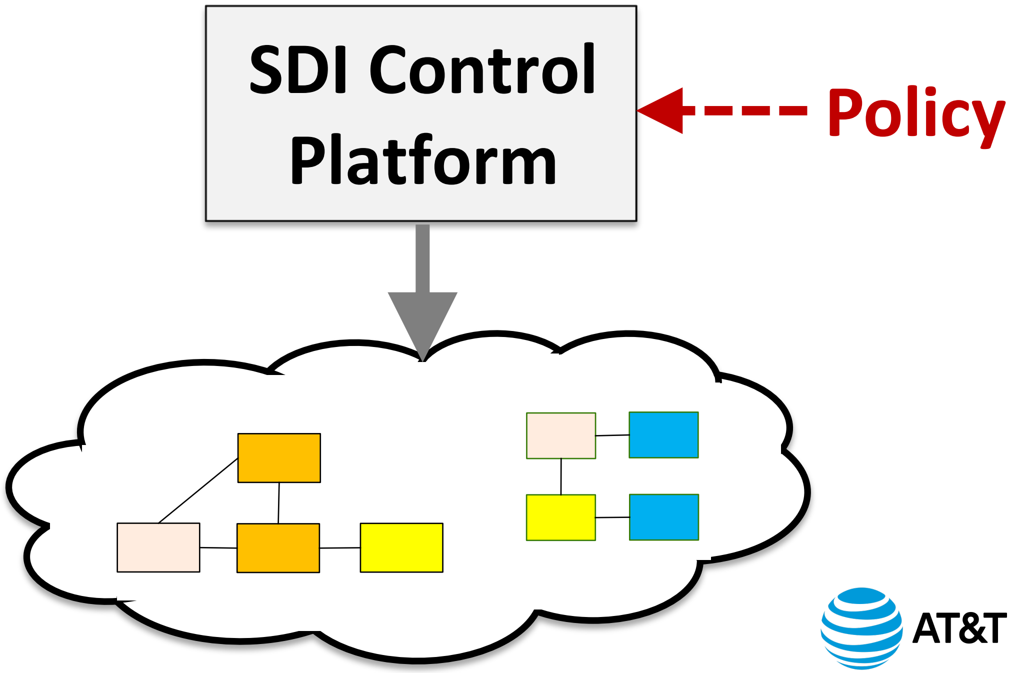
Software Defined Infrastructure (SDI)

Motivation

- Push in industry and academia towards SDI control platforms
  - ONAP, OpenNFV, CORD, etc.
  - Automate network service instantiation and management

- Automation expected to be done through **policies** *(condition → action)*
  - Unintended consequences
  - Performance degradation, SLA violations

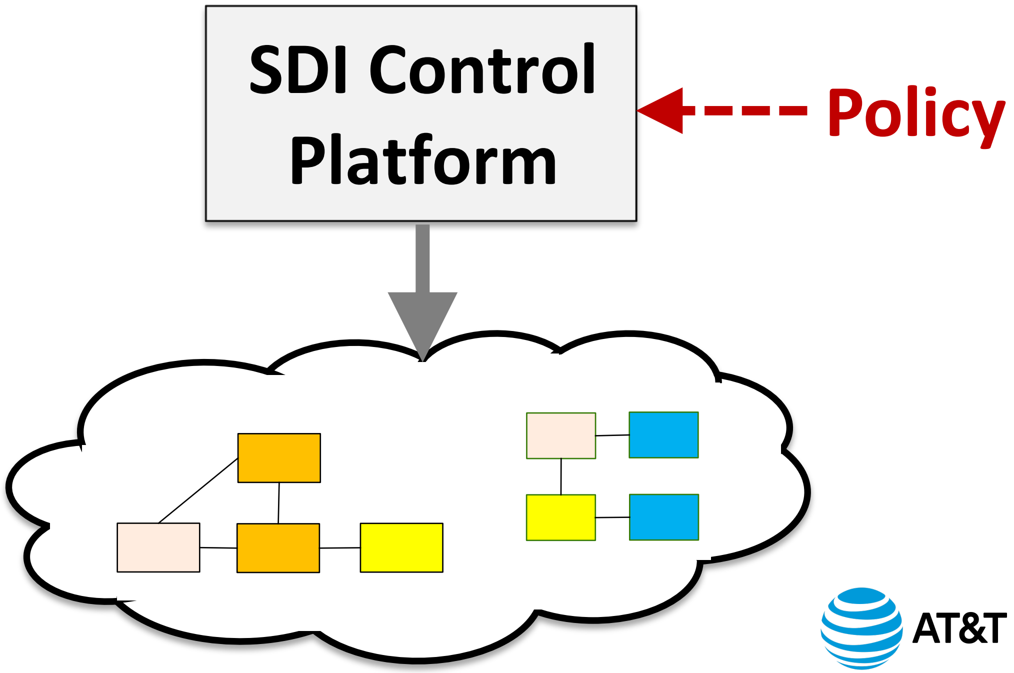
Software Defined Infrastructure (SDI)



Motivation

- Push in industry and academia towards SDI control platforms
  - ONAP, OpenNFV, CORD, etc.
  - Automate network service instantiation and management

- Automation expected to be done through **policies** *(condition → action)*
  - Unintended consequences
  - Performance degradation, SLA violations



Software Defined Infrastructure (SDI)

Motivation

- Need to ensure safe policy deployment by **determining impact**
  - Earlier efforts focus on low-level ACL or routing policies (e.g., BGP, SDN rules)
  - SDI enables orchestration and service level policies
    - Dynamic scaling, load balancing, orchestration and placement, migrations, edge cloud offloading, etc.
  - Static offline checks – not enough
  - Simulation – model may be incorrect or incomplete

Motivation

- Thus, there is a need for
  - an automated tool coupled with SDI control platforms
  - so policies can be tested for runtime impact before being deployed in production

- Required properties:
  - Able to test SDI policies for *runtime* impact in *varying environments*
    - E.g., varying traffic profiles, resource and service configurations, etc.
  - Coupled with SDI control platform
  - Automated

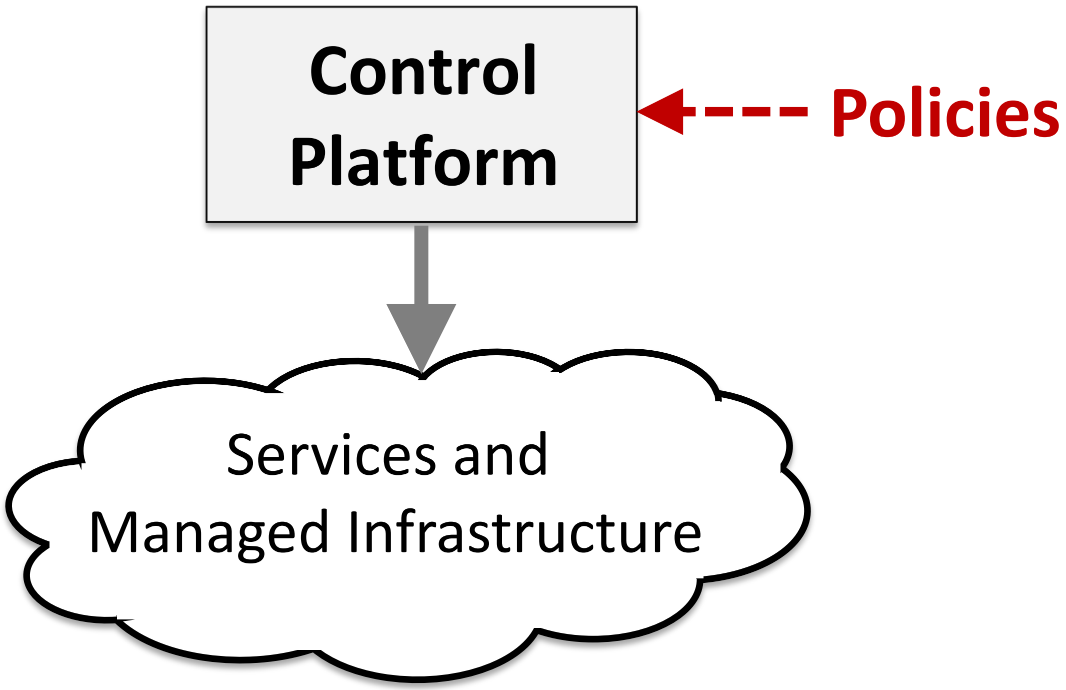
DEPO: A Platform for Safe DEployment of POlicy

- **Captures runtime impact of SDI policies using an iterative emulations based approach**

- **Required properties** and corresponding DEPO design principles
  - **Ability to test SDI policies for *runtime* impact in *varying environments***
    - **emulations** – realism with control
    - **continuous impact learning** – improves over time
  - **Coupled with SDI control platform**
    - **part of policy deployment workflow** – comes after static testing, before production deployment
  - **Automated**
    - **knowledge based modeling**
      - automates reasoning for test environment generation
      - automates ML model creation / statistical data analysis for learning impact

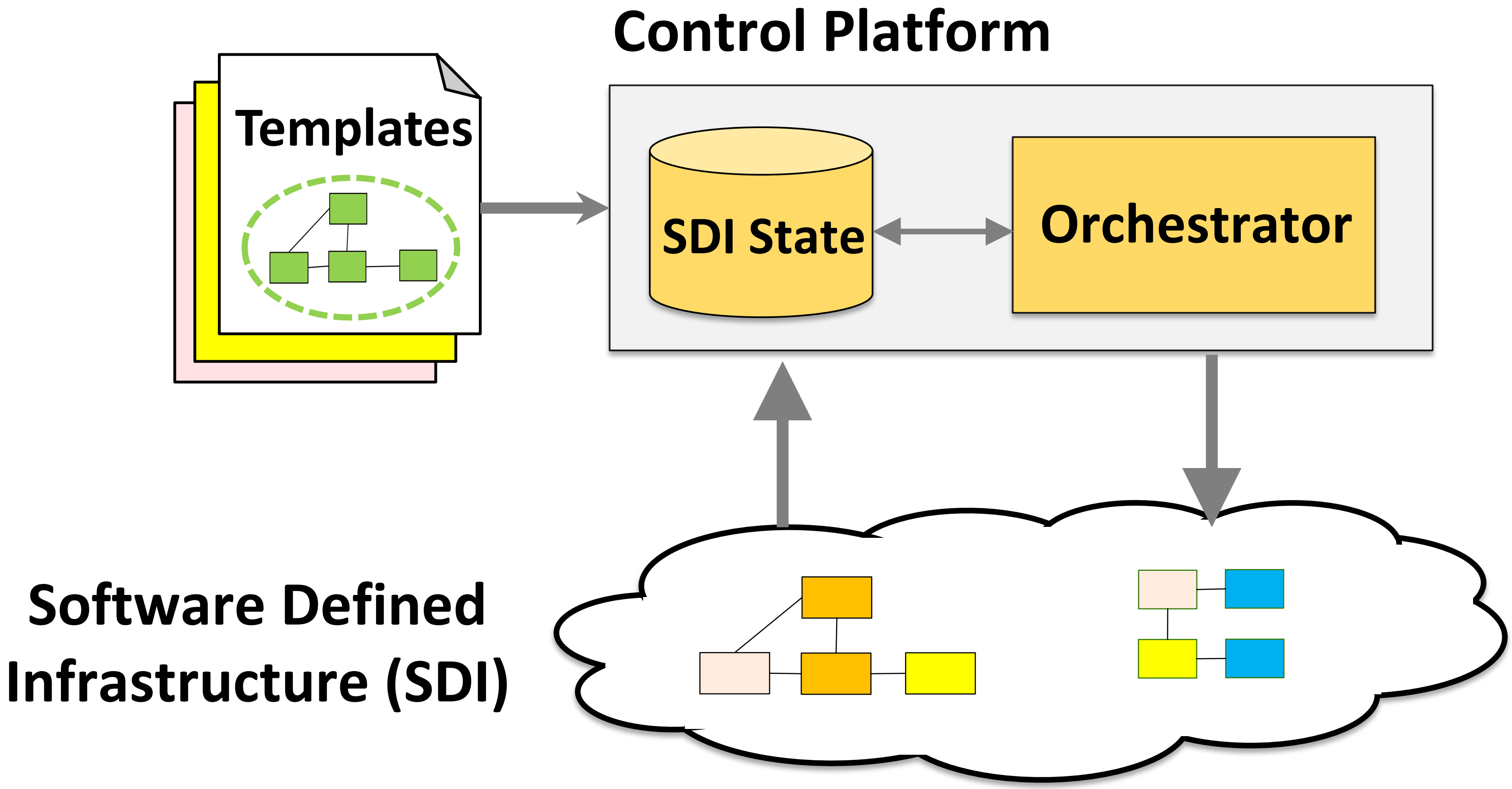
AT&T

DEPO Context

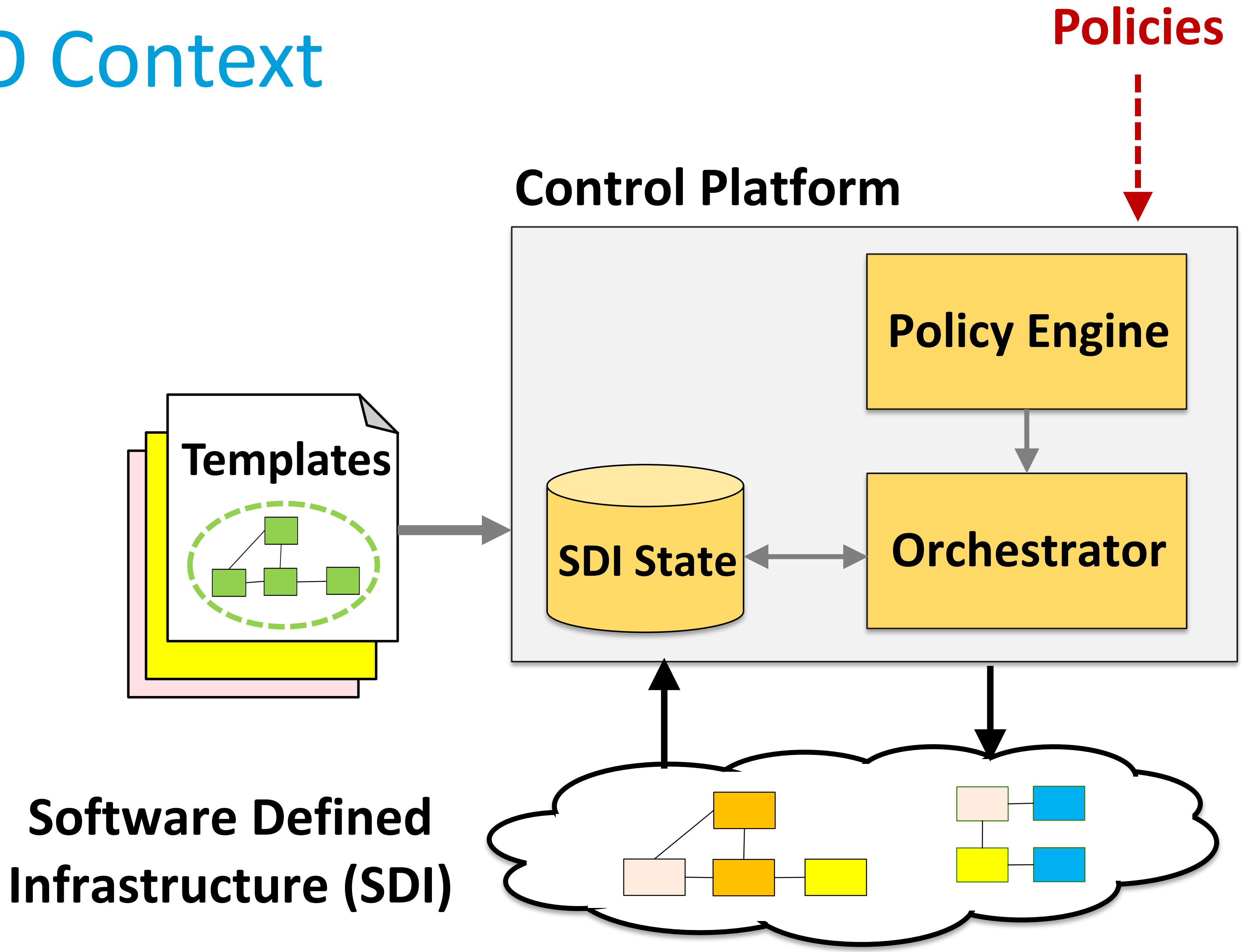


Software Defined Infrastructure (SDI)

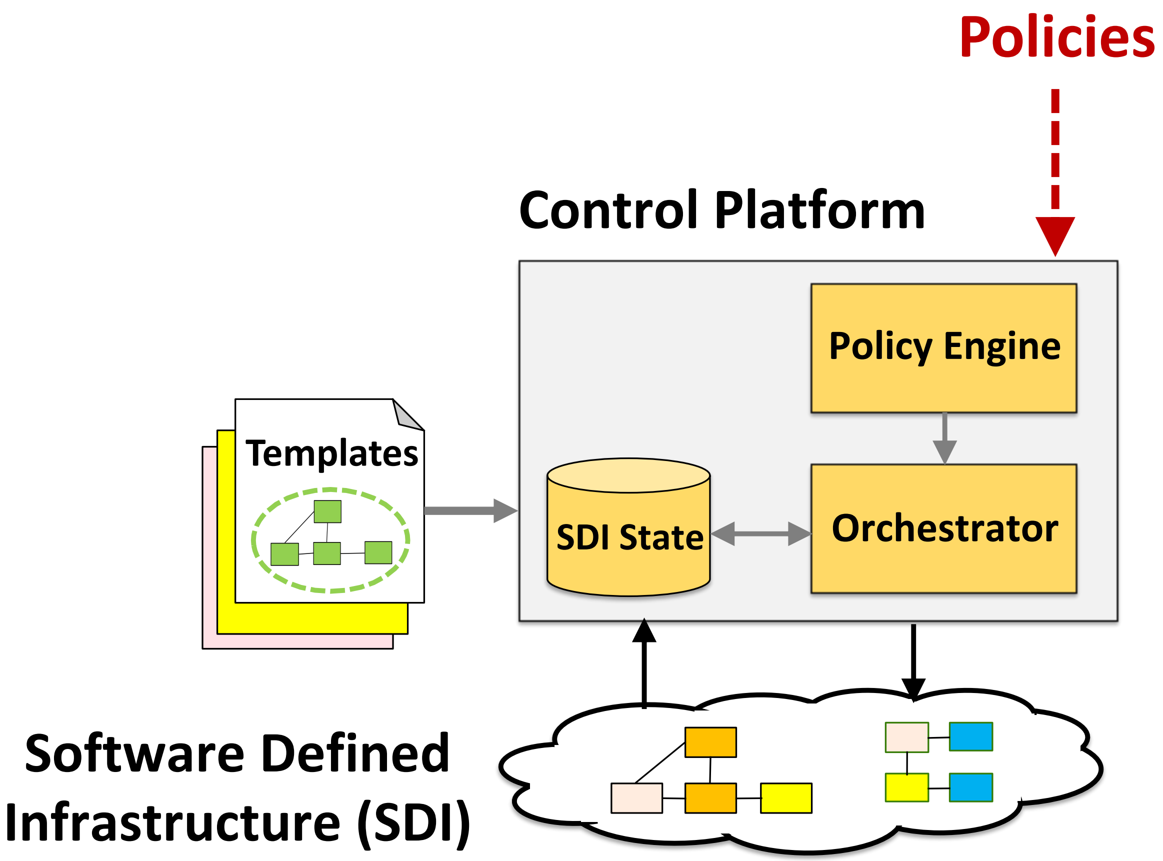
DEPO Context



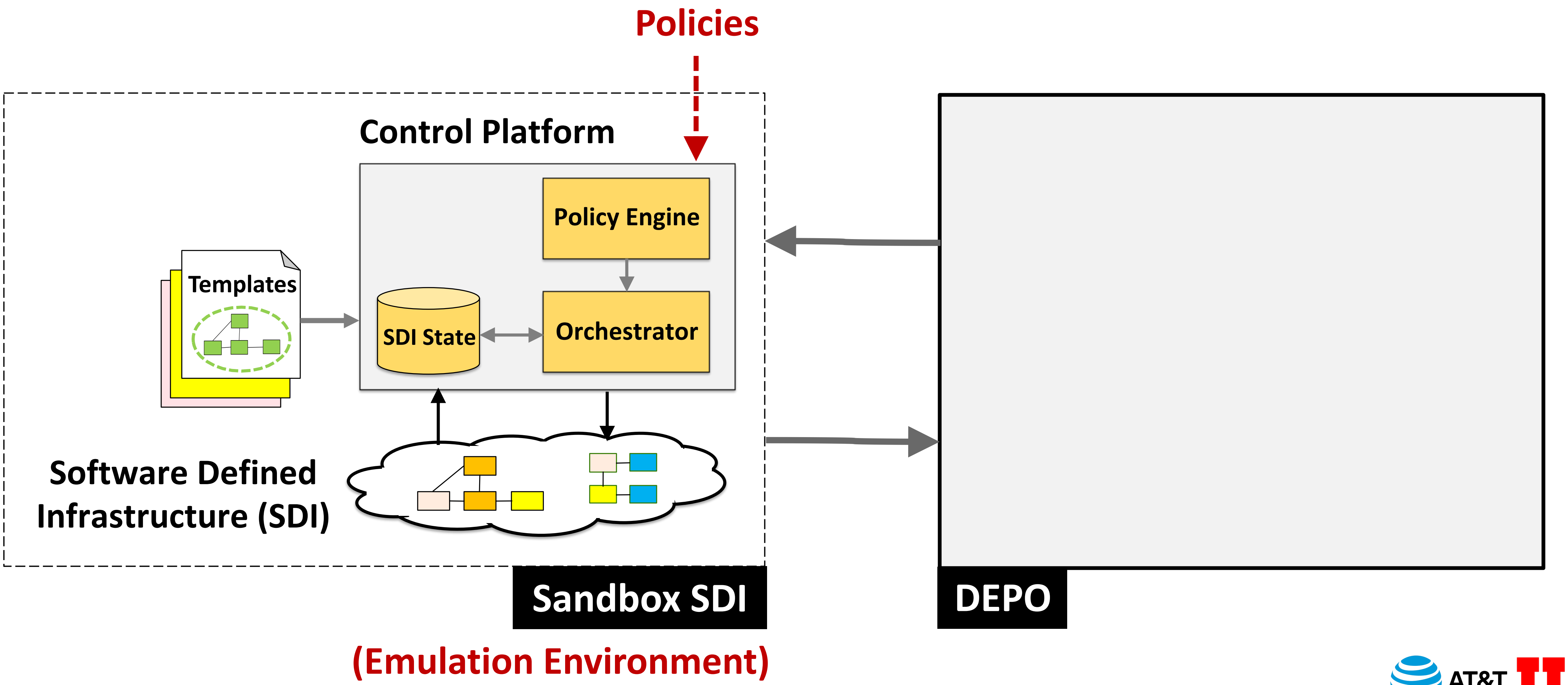
DEPO Context



DEPO Context

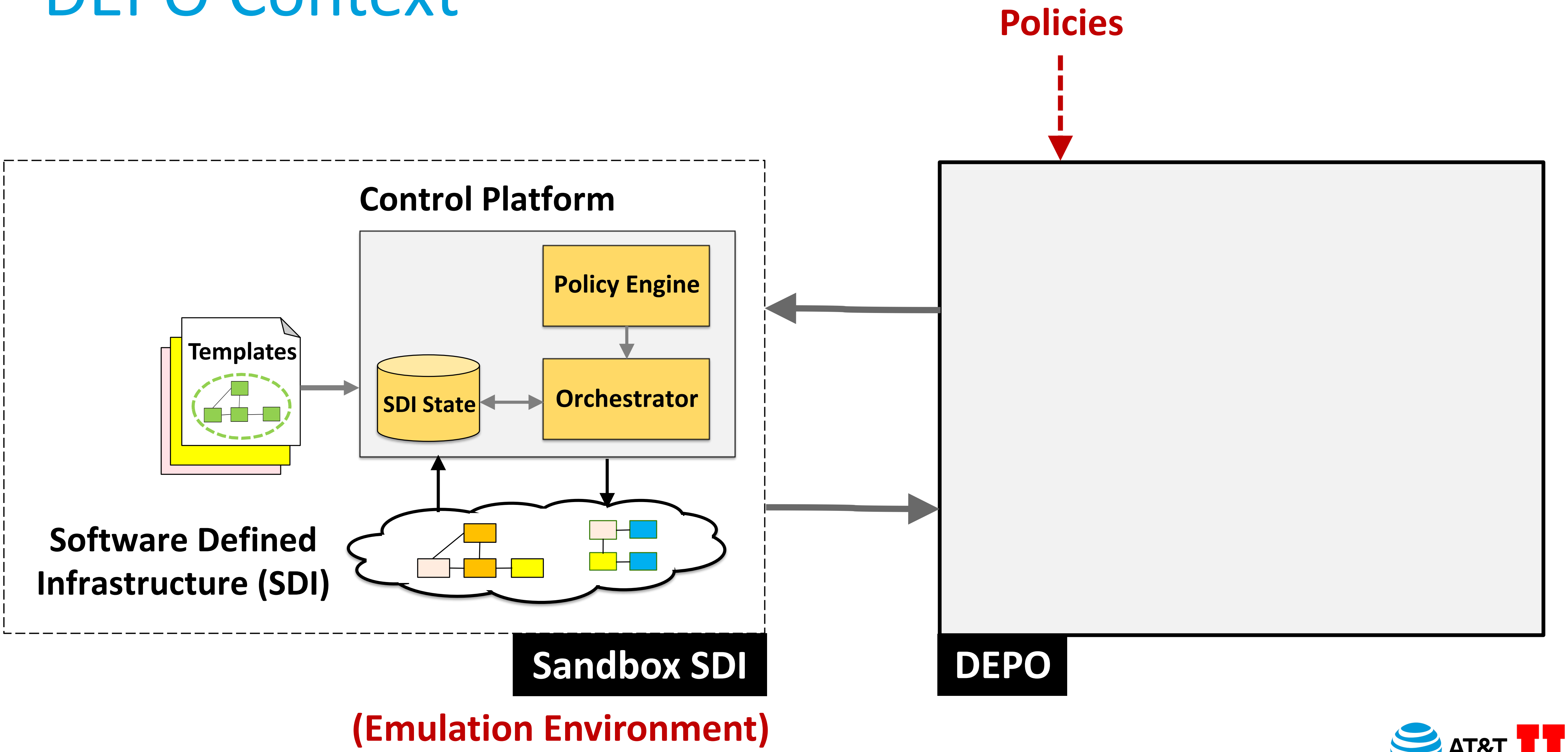


DEPO Context

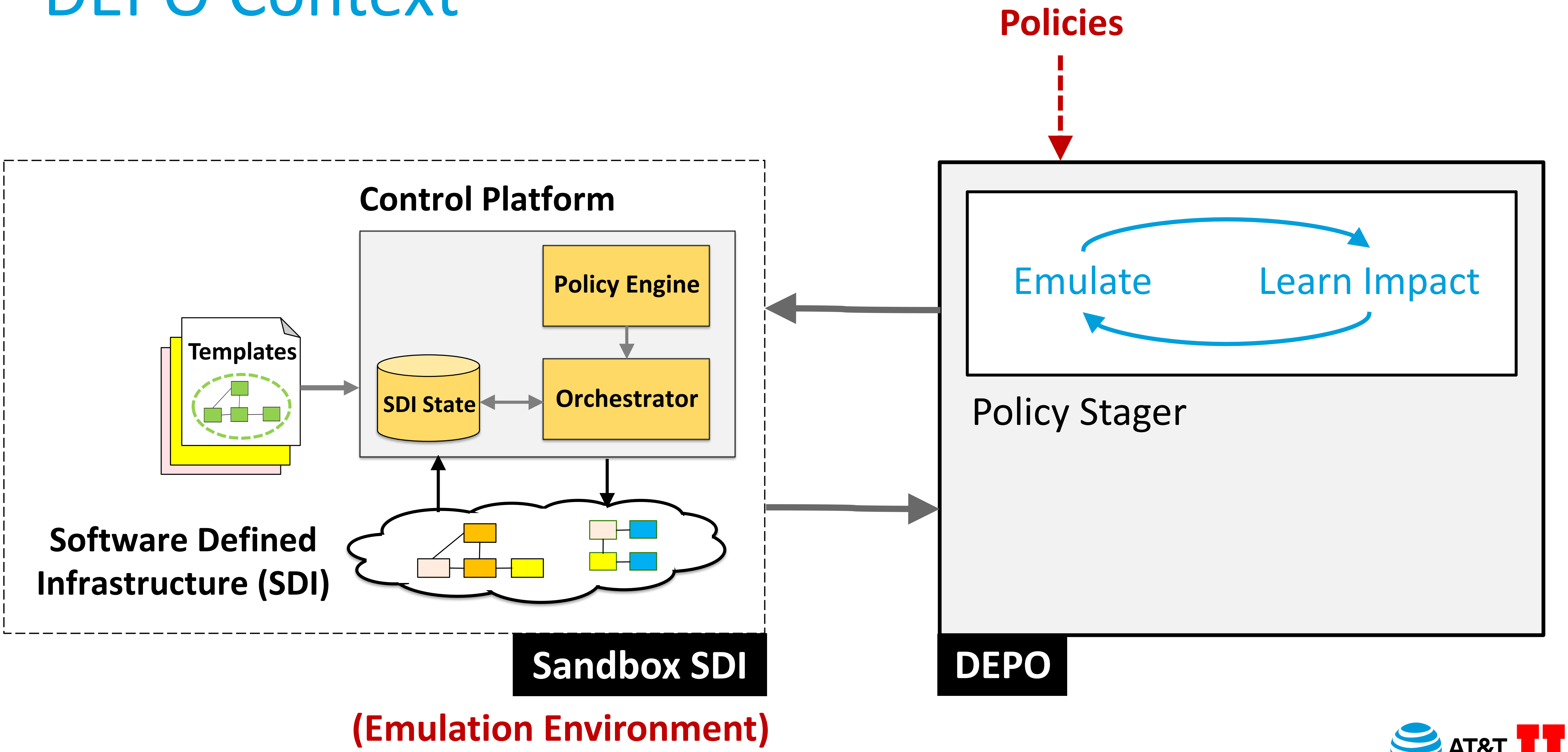


AT&T

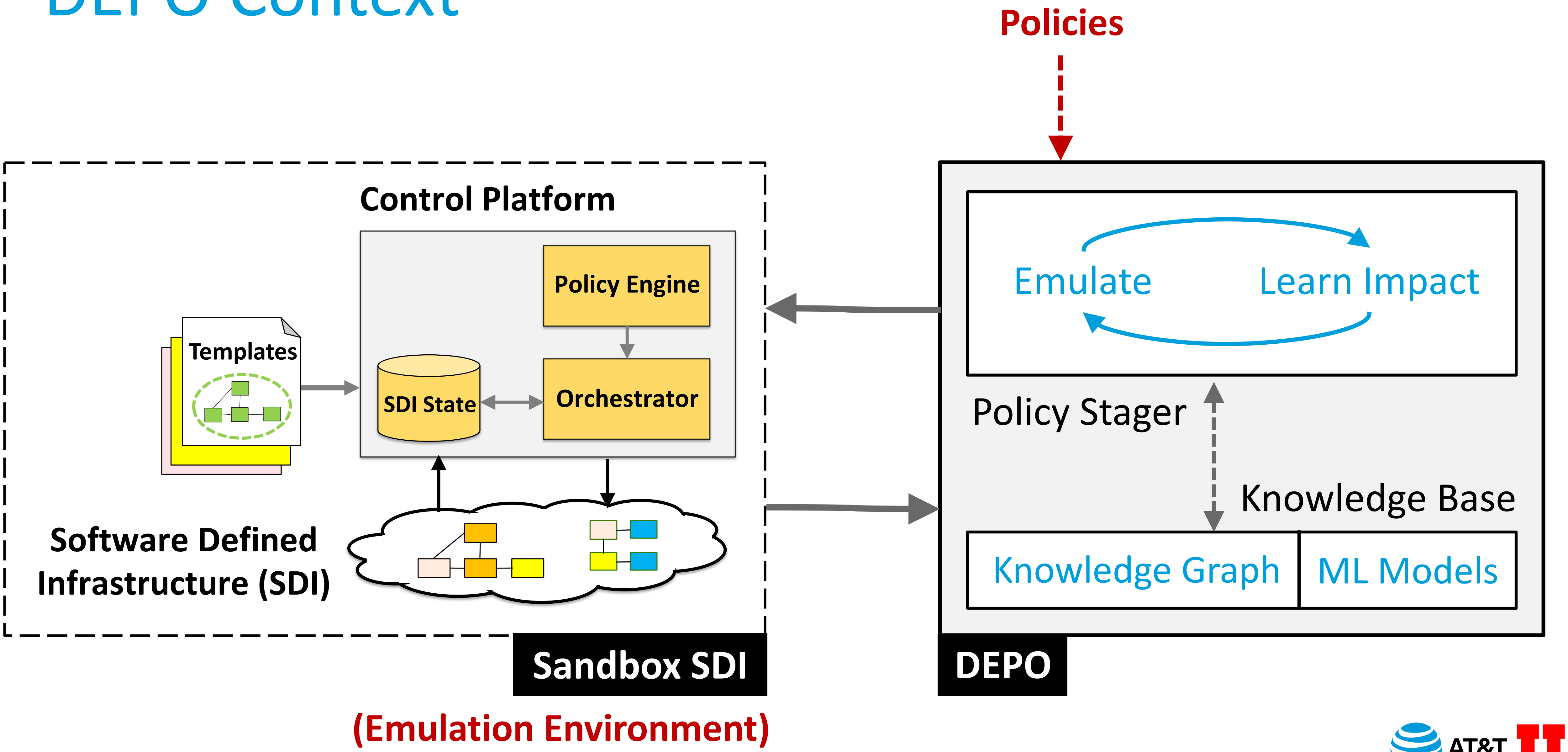
DEPO Context



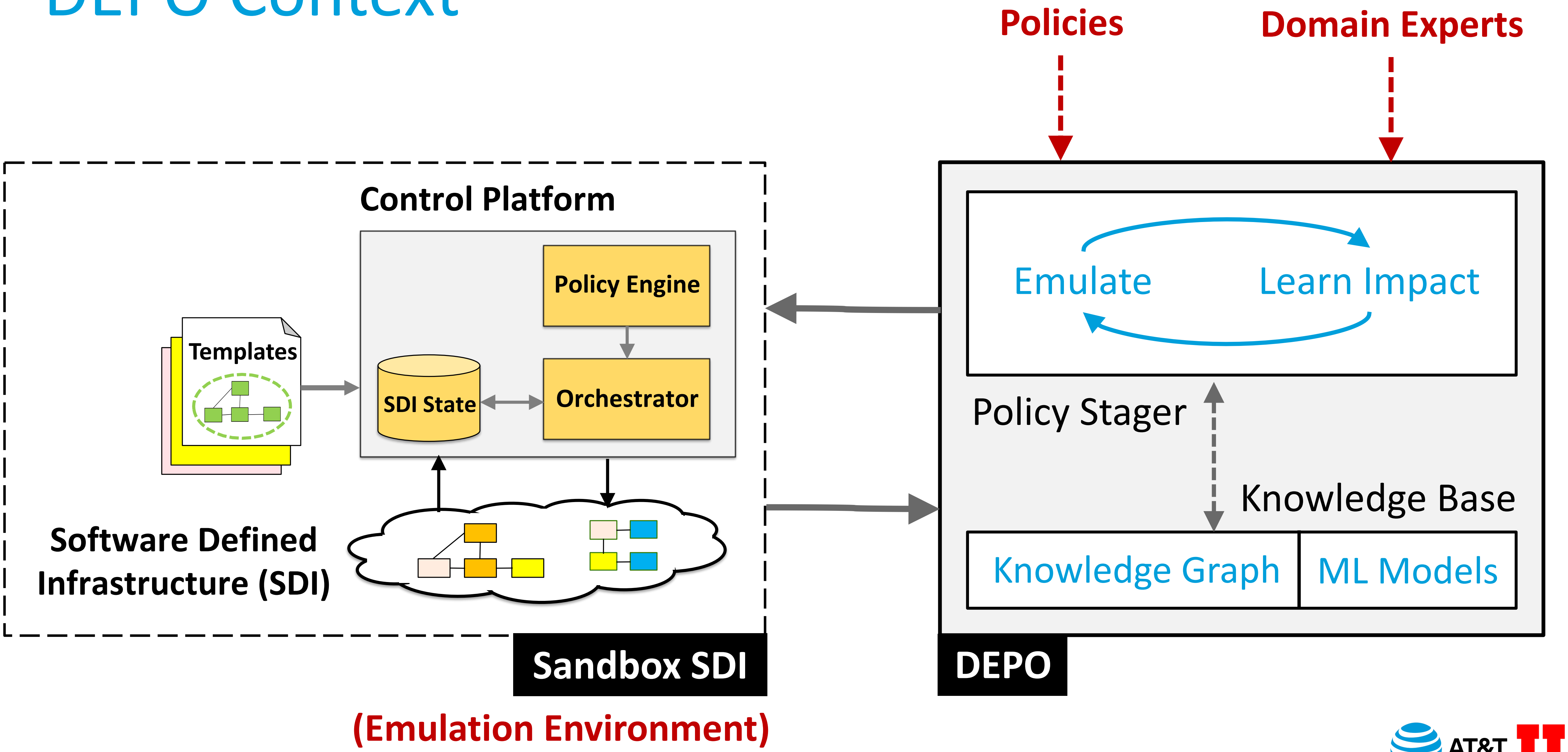
DEPO Context



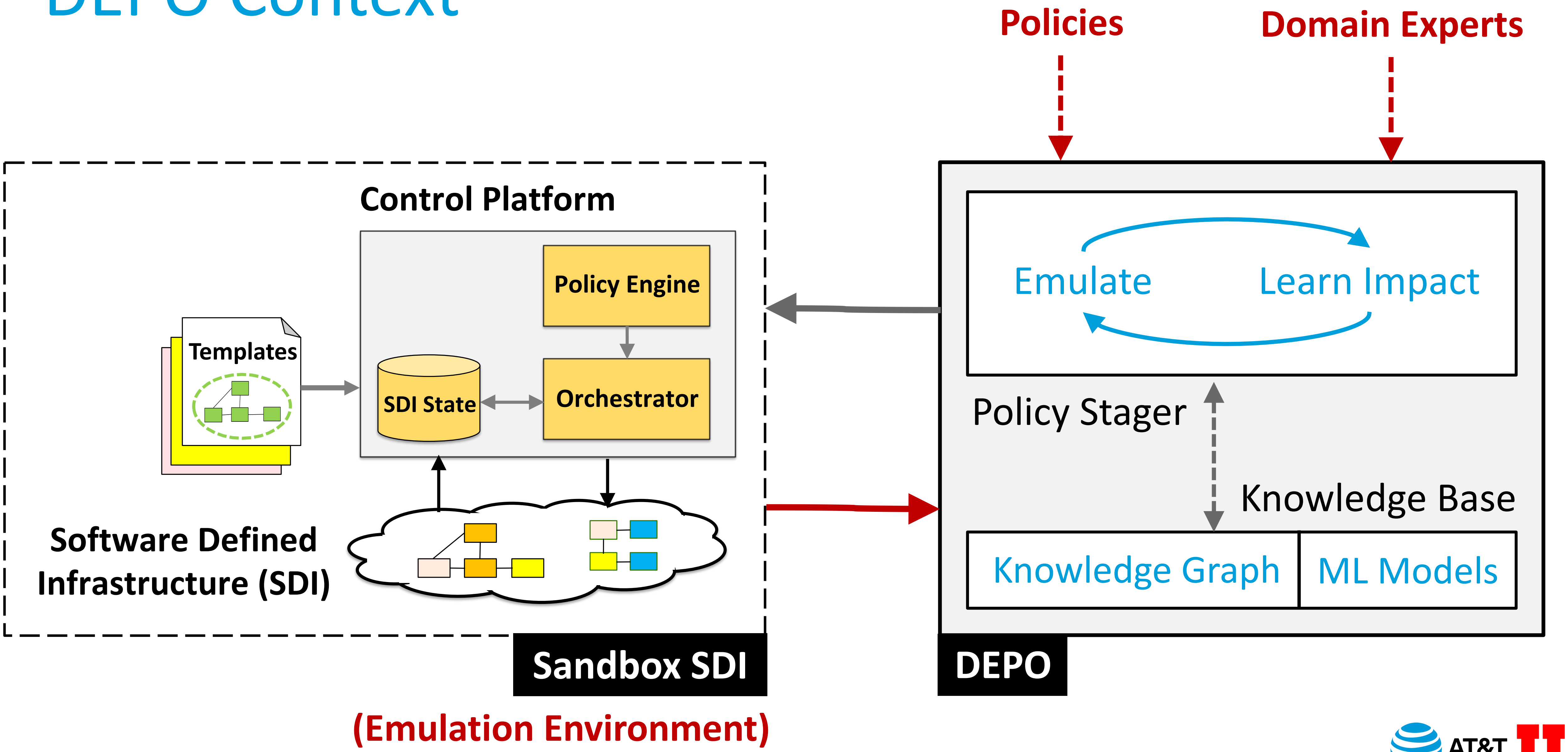
DEPO Context



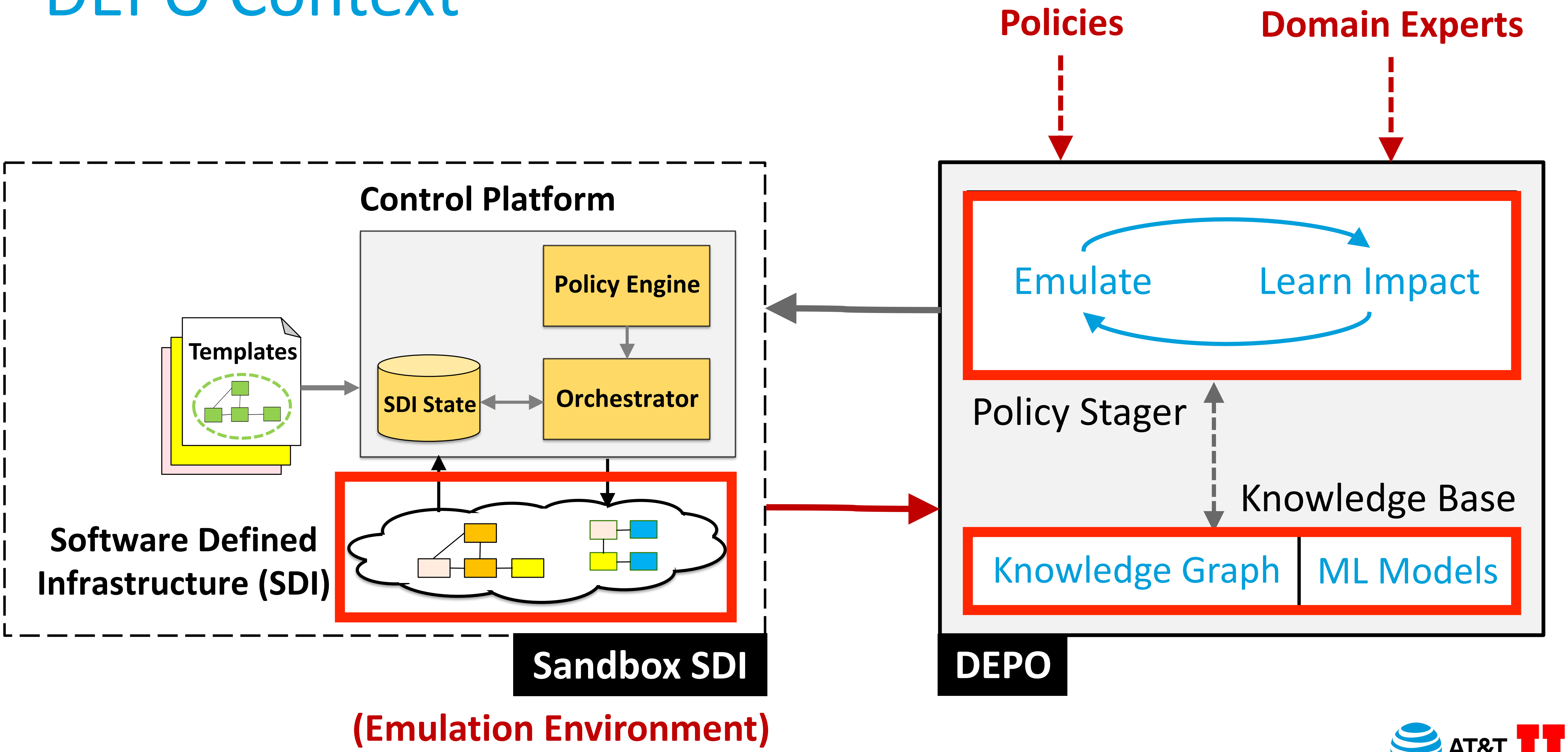
DEPO Context



DEPO Context

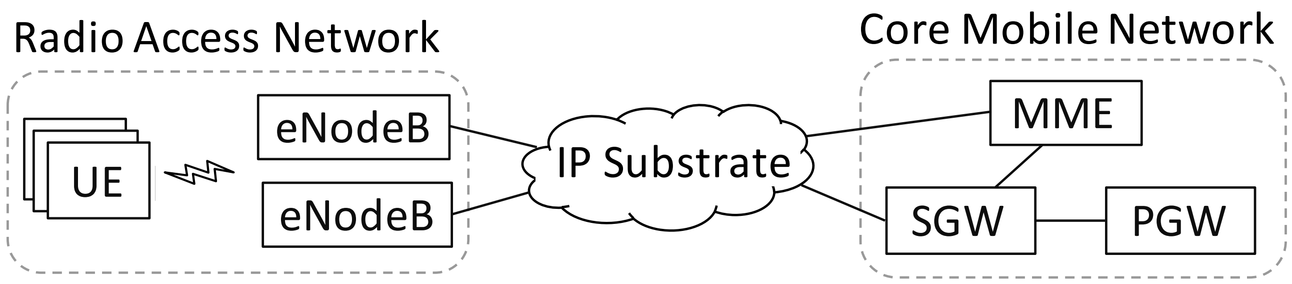


DEPO Context



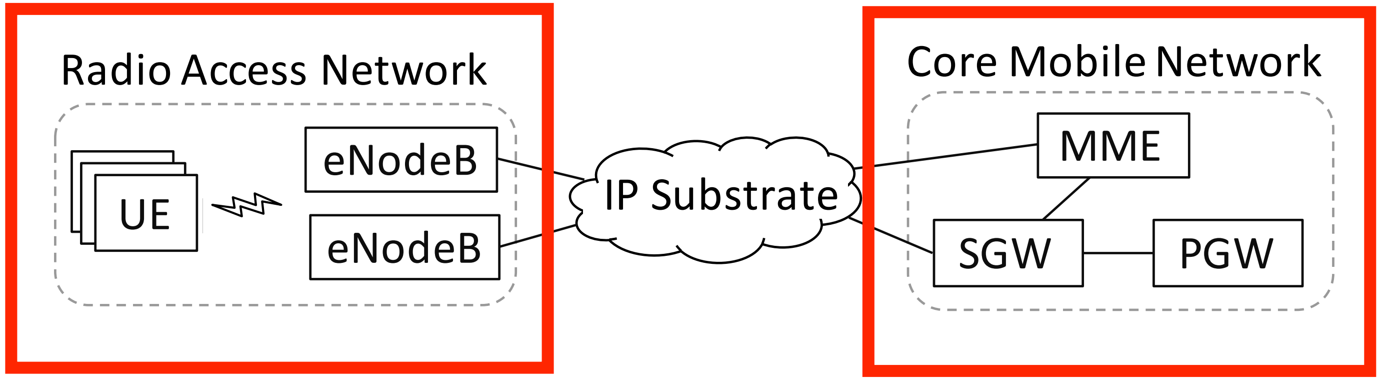
SDI Use-Case Services

- Use-case 1: Standard 4G LTE/EPC broadband service



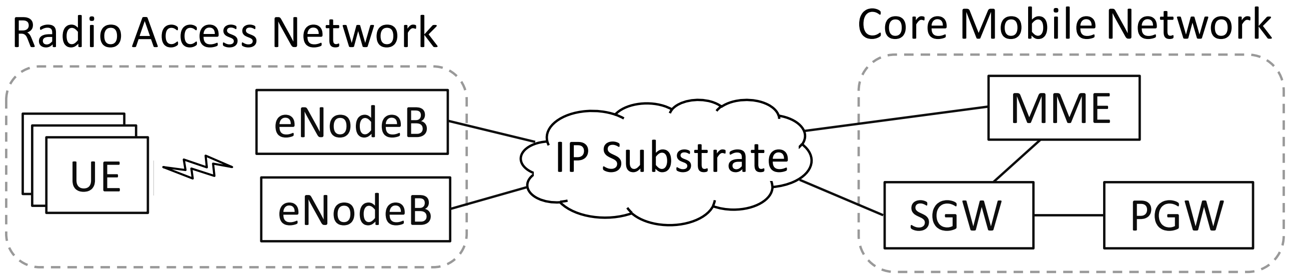
SDI Use-Case Services

- Use-case 1: Standard 4G LTE/EPC broadband service

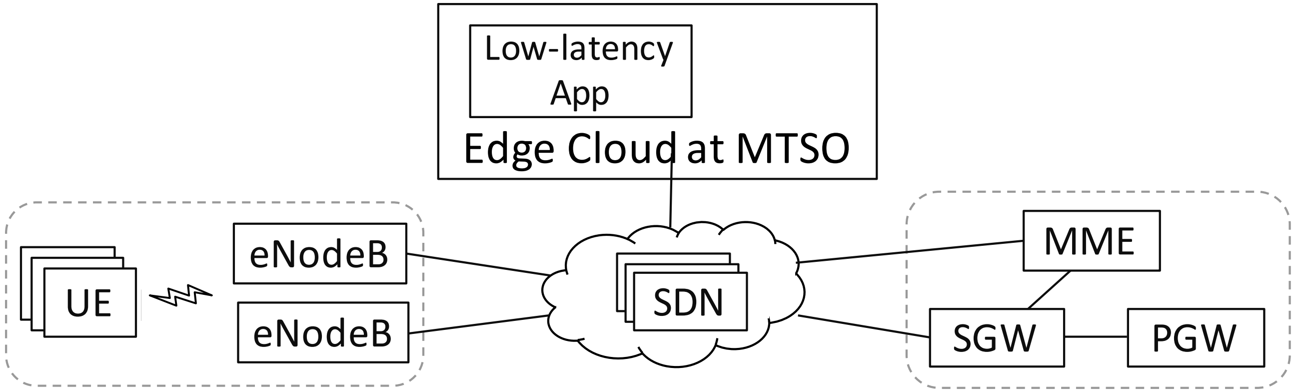


SDI Use-Case Services

- Use-case 1: Standard 4G LTE/EPC broadband service

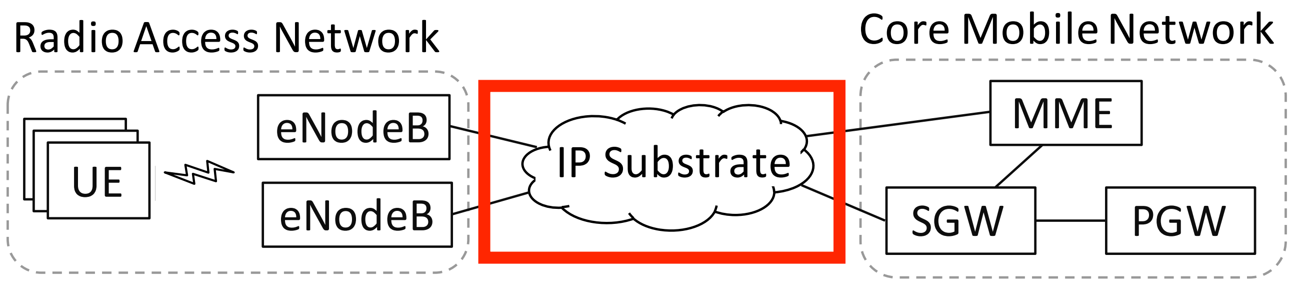


- Use-case 2: 4G LTE/EPC service **with edge cloud offloading** (*SMORE by Cho et al. ATC'14*)

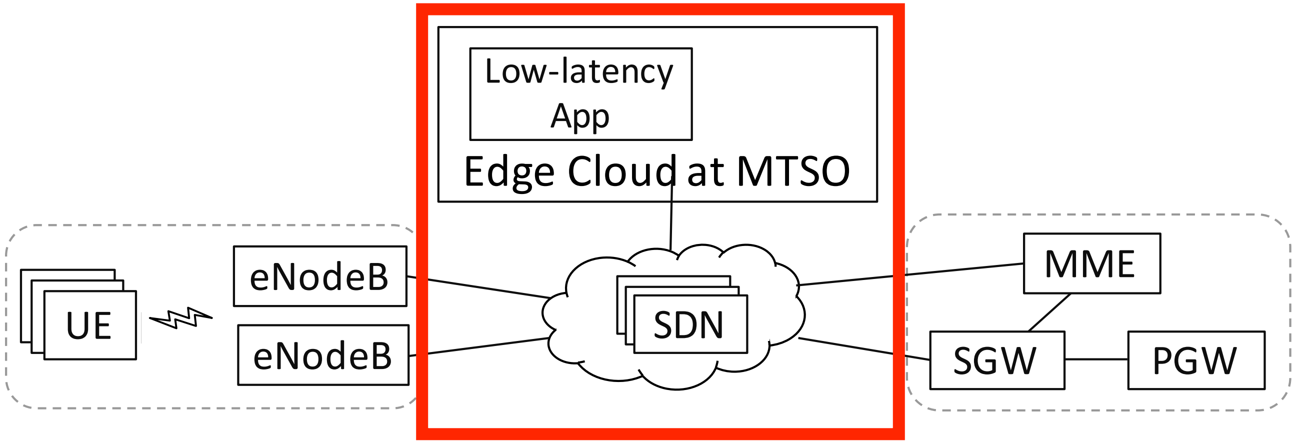


SDI Use-Case Services

- Use-case 1: Standard 4G LTE/EPC broadband service

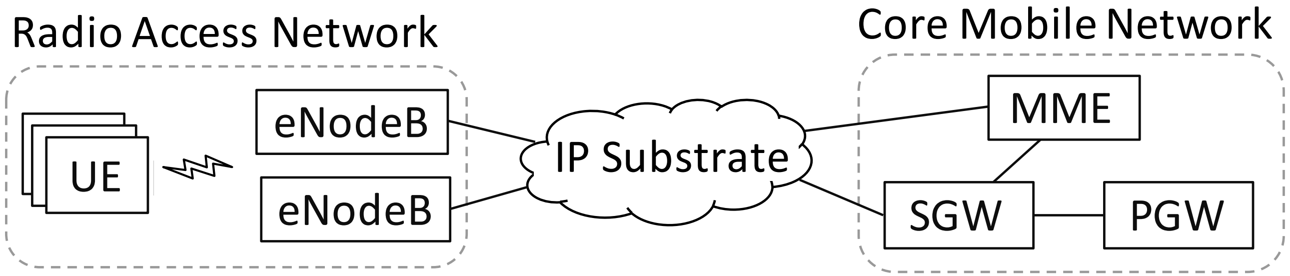


- Use-case 2: 4G LTE/EPC service **with edge cloud offloading** (*SMORE by Cho et al. ATC'14*)

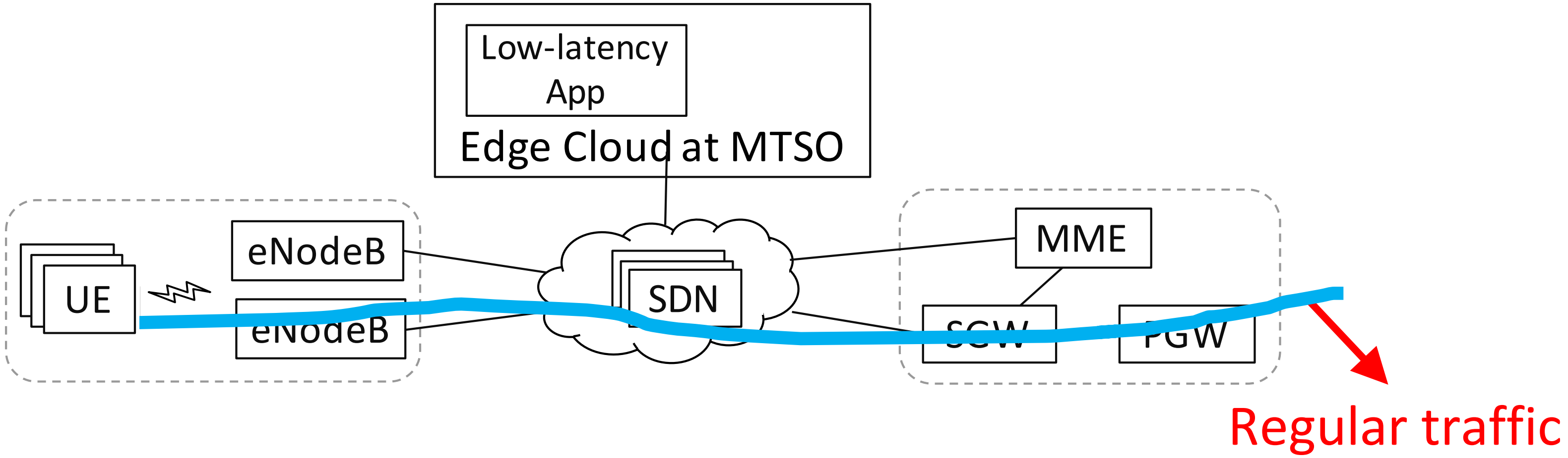


SDI Use-Case Services

- Use-case 1: Standard 4G LTE/EPC broadband service

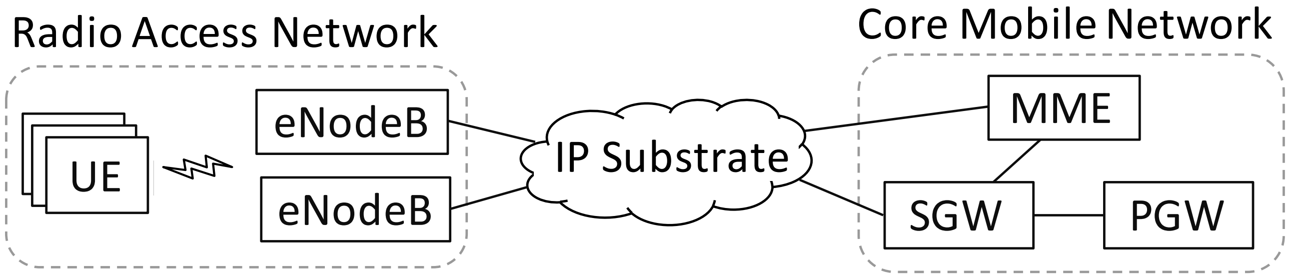


- Use-case 2: 4G LTE/EPC service **with edge cloud offloading** (*SMORE by Cho et al. ATC'14*)

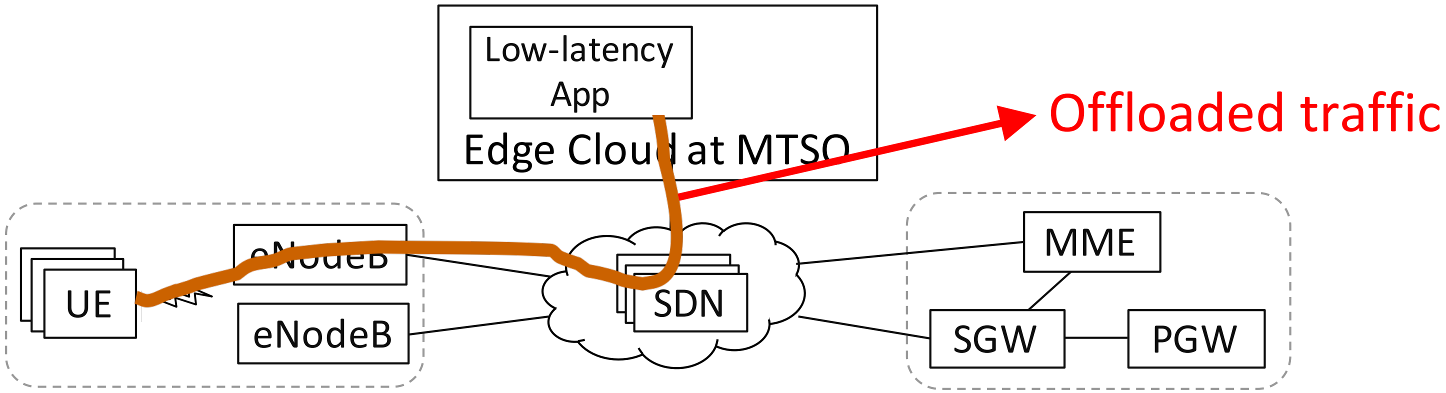


SDI Use-Case Services

- Use-case 1: Standard 4G LTE/EPC broadband service



- Use-case 2: 4G LTE/EPC service **with edge cloud offloading** (*SMORE by Cho et al. ATC'14*)

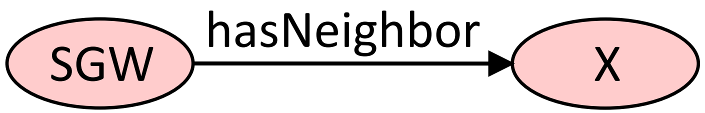


DEPO Component: Knowledge Graph (KG)

- Unit of data storage is a ***fact***
  - 3-tuple of the form: **node1 relationship node2**


```
EPC isA Service
Service hasComponent NF
EPC hasComponent NF
EPC1 hasType EPC
EPC2 hasType EPC
VNF isA NF
MME isA VNF
MME1 hasType MME
EPC1 hasComponent MME1
Server isA ComputeNode
Server hosts VNF
Server1 isA Server
Server2 hosts MME1
MME.usage isA UsageVariable
MME1.usage hasValue 90% [timestamp=123]
```

Example KG



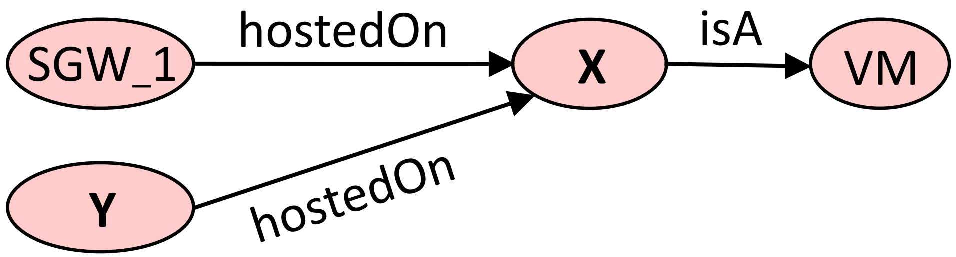
Example Query

DEPO Component: Knowledge Graph (KG)

- Unit of data storage is a ***fact***
  - 3-tuple of the form: **node1 relationship node2**

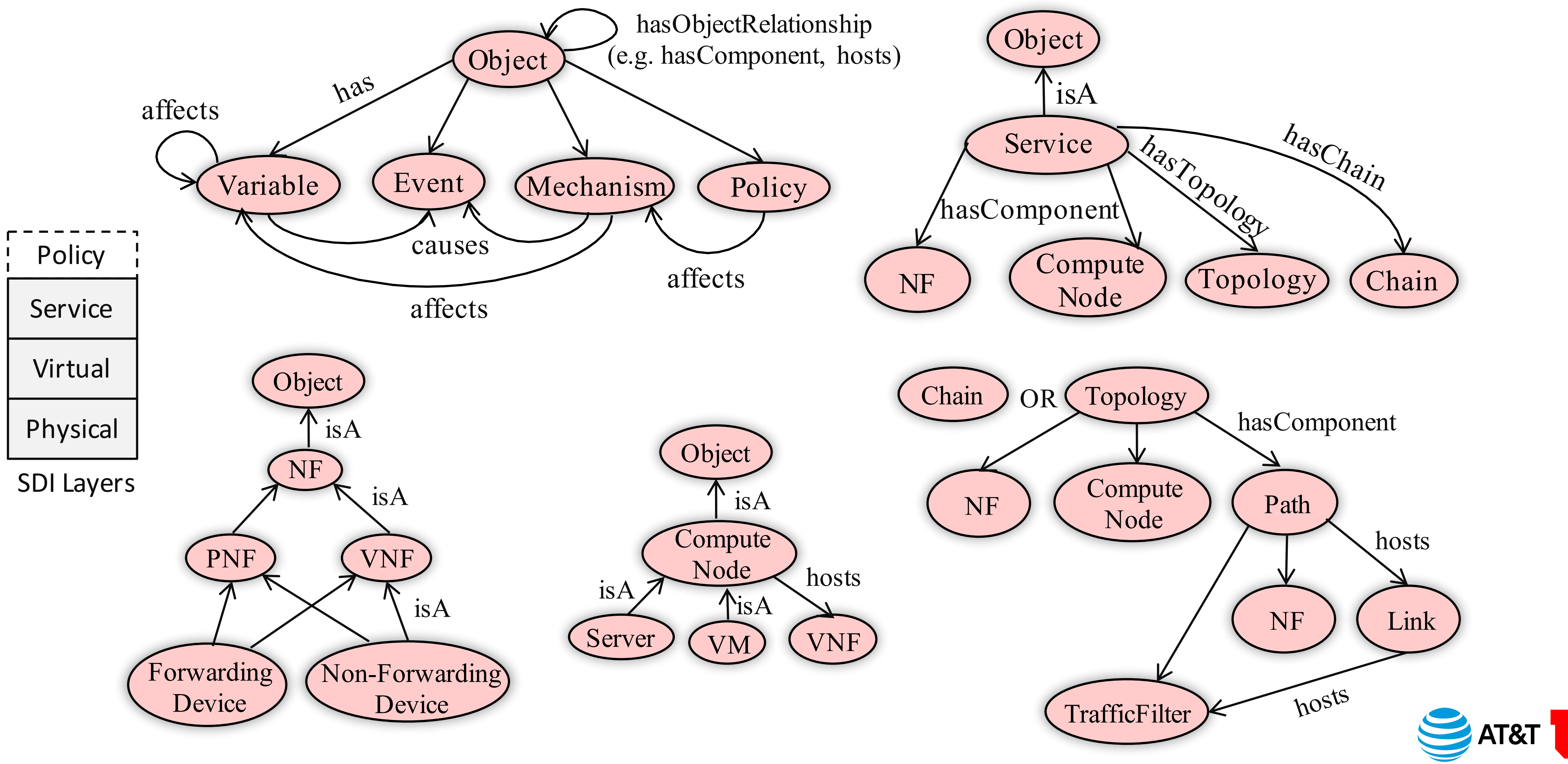

```
EPC isA Service
Service hasComponent NF
EPC hasComponent NF
EPC1 hasType EPC
EPC2 hasType EPC
VNF isA NF
MME isA VNF
MME1 hasType MME
EPC1 hasComponent MME1
Server isA ComputeNode
Server hosts VNF
Server1 isA Server
Server2 hosts MME1
MME.usage isA UsageVariable
MME1.usage hasValue 90% [timestamp=123]
```

Example KG



Example Query

DEPO: Modeling Knowledge



DEPO: Impact Learning Approach

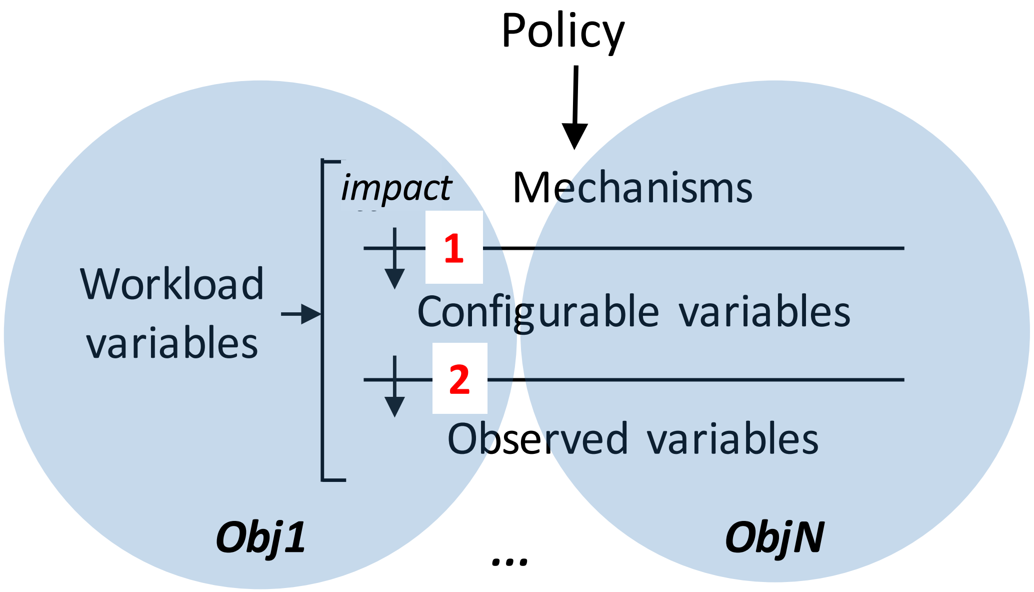
- Variables
  - **configurable**: num CPU, location, IP address, num of SGW, status
  - **observed**: usage (CPU, mem), throughput, response time
  - **workload**: rate of requests to SMORE VNF, UE attaches, traffic types
  - **emulation environment parameters**: subset of configurable/workload
- Mechanisms
  - potentially cause some change: start, stop, update, migrate
- Policies
  - higher level mechanism (if-then)
- Policy impact (statistically significant change – not categorized as good/bad)
  - Policy → mechanism → variable

Template

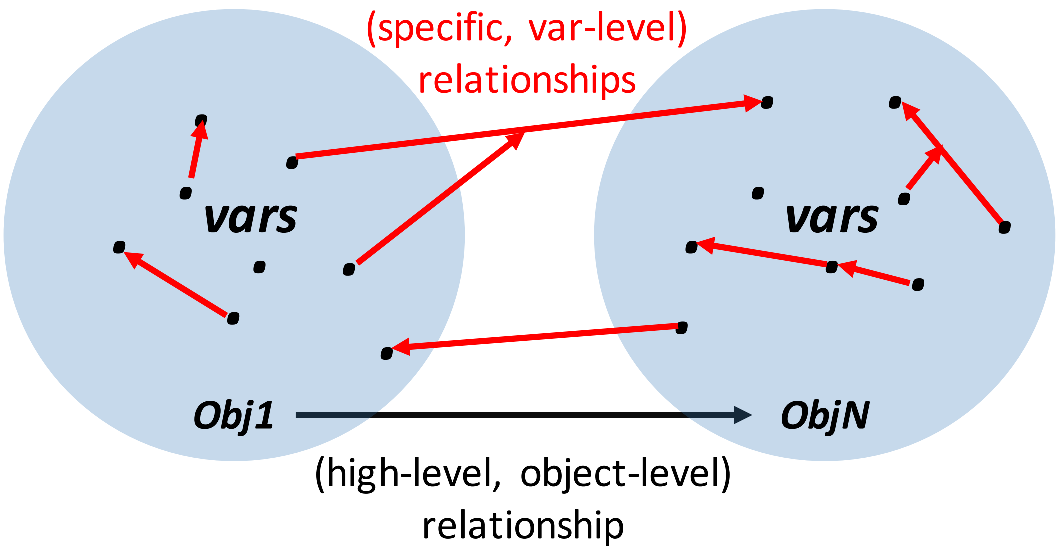
Variables
Mechanisms (lifecycle)
Policies
…
Topology spec
Orchestration recipes
Implementation artifacts
…

AT&T

DEPO: Impact Learning Approach



Defining impact systematically

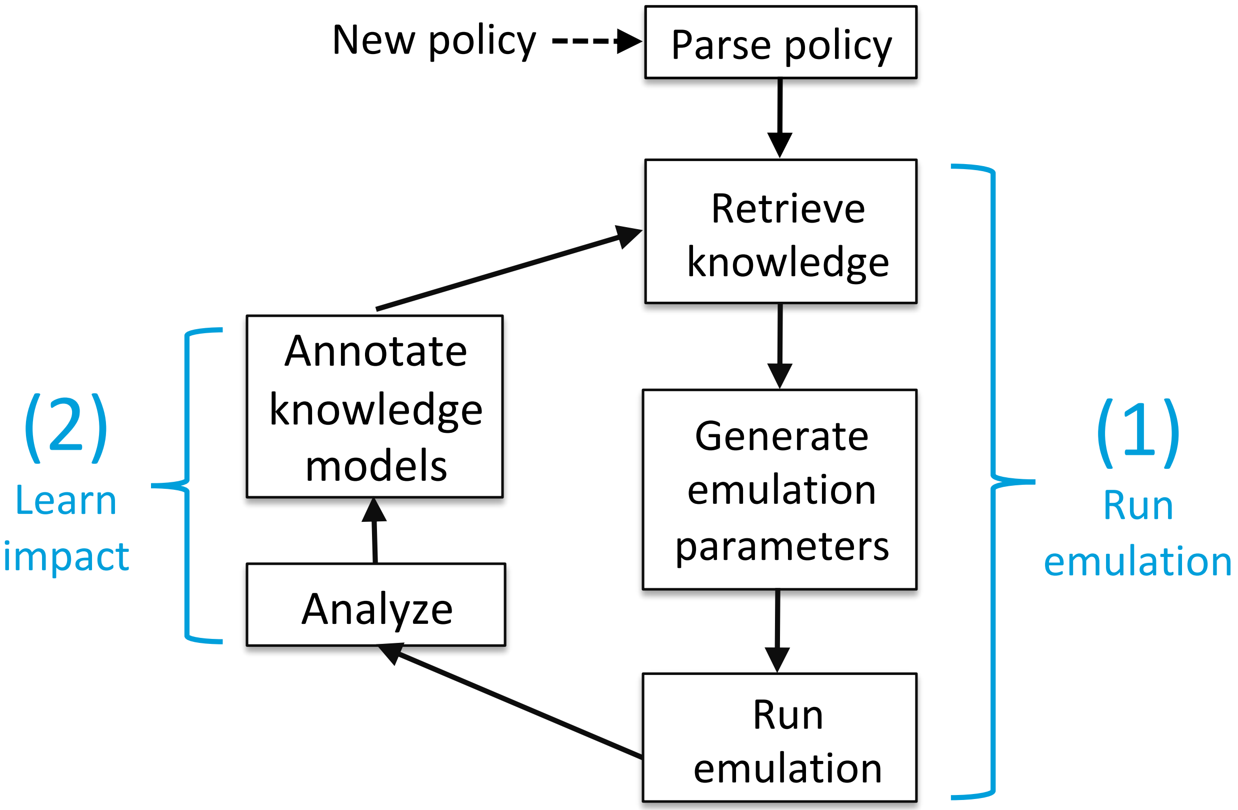


- Policy writers may already know high-level impact – object-level relationships
- But non trivial to quantify impact at finer granularities – variable-level relationships

DEPO: Impact Learning Approach

1) Run emulation
   - Generating emulation environment suitable for testing a policy in sandbox SDI
   - Running emulation by running traffic through it and collecting logs

2) Learning impact
   - Analyze logs and annotate knowledge models



DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

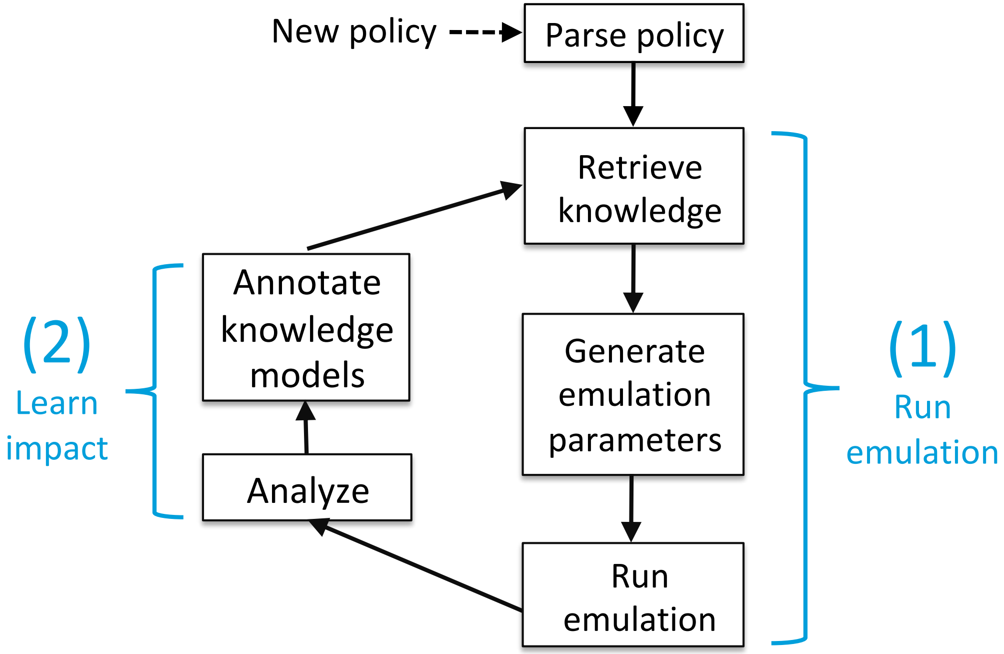
```
when:
     SGW.usage >= THRESH  //THRESH = 70%
then:
     SGW.scaleUp()
```

DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
      SGW.usage >= THRESH  //THRESH = 70%
then:
      SGW.scaleUp()
```

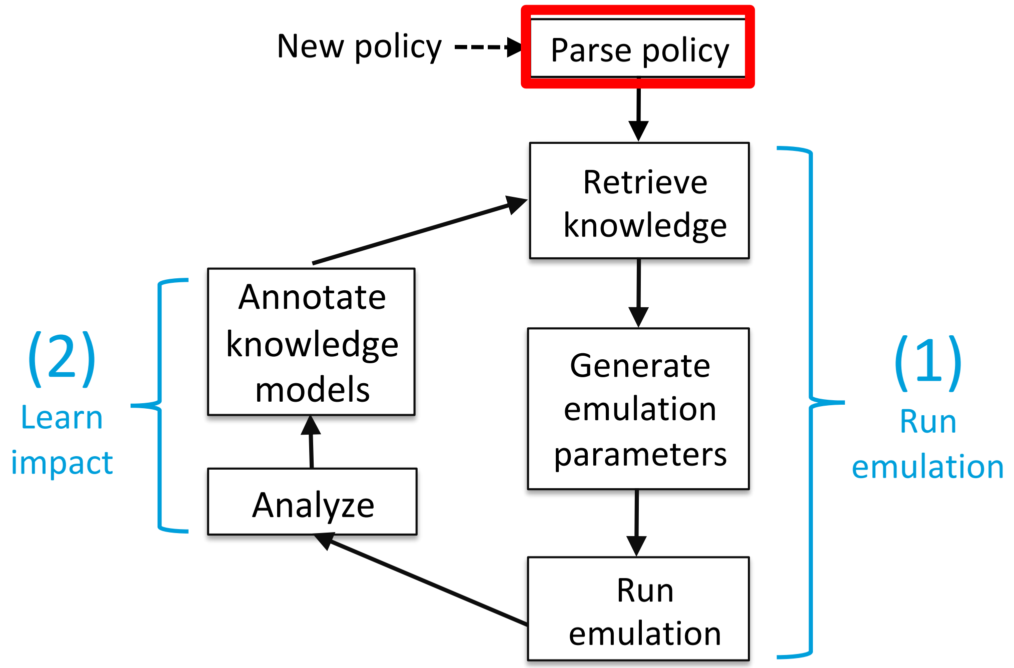


DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
      SGW.usage >= THRESH  //THRESH = 70%
then:
      SGW.scaleUp()
```



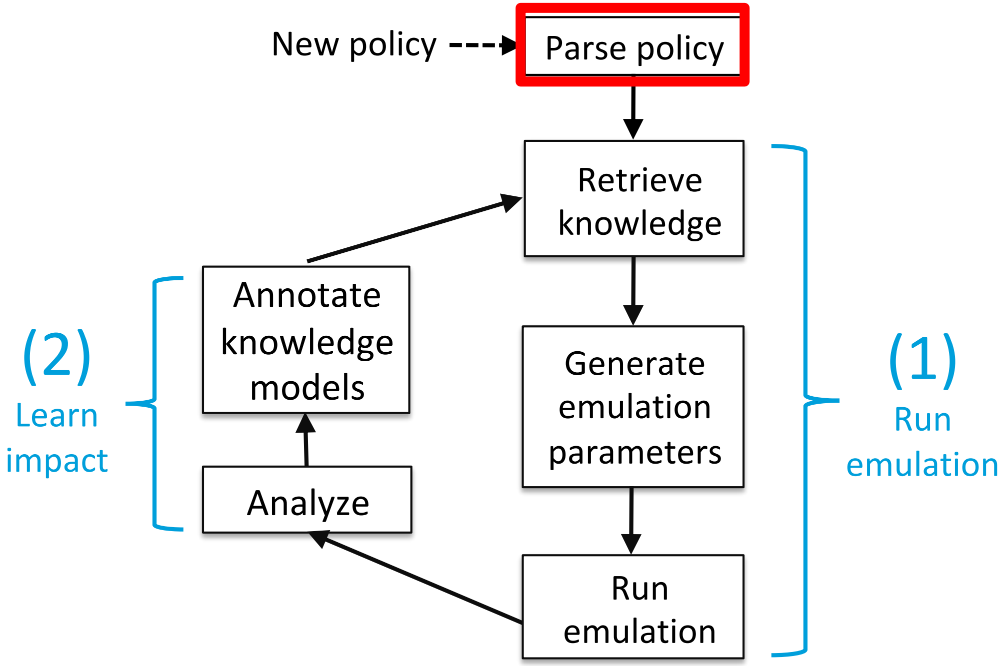
DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
      SGW.usage >= THRESH  //THRESH = 70%
then:
      SGW.scaleUp()
```

SGW

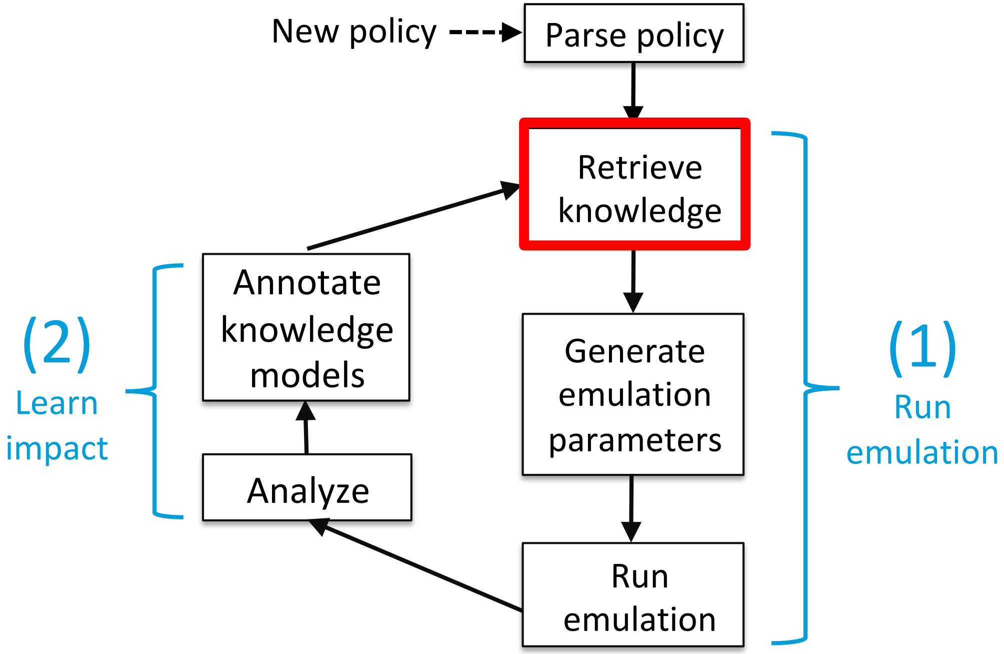


DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
      SGW.usage >= THRESH  //THRESH = 70%
then:
      SGW.scaleUp()
```



DEPO: Impact Learning Approach

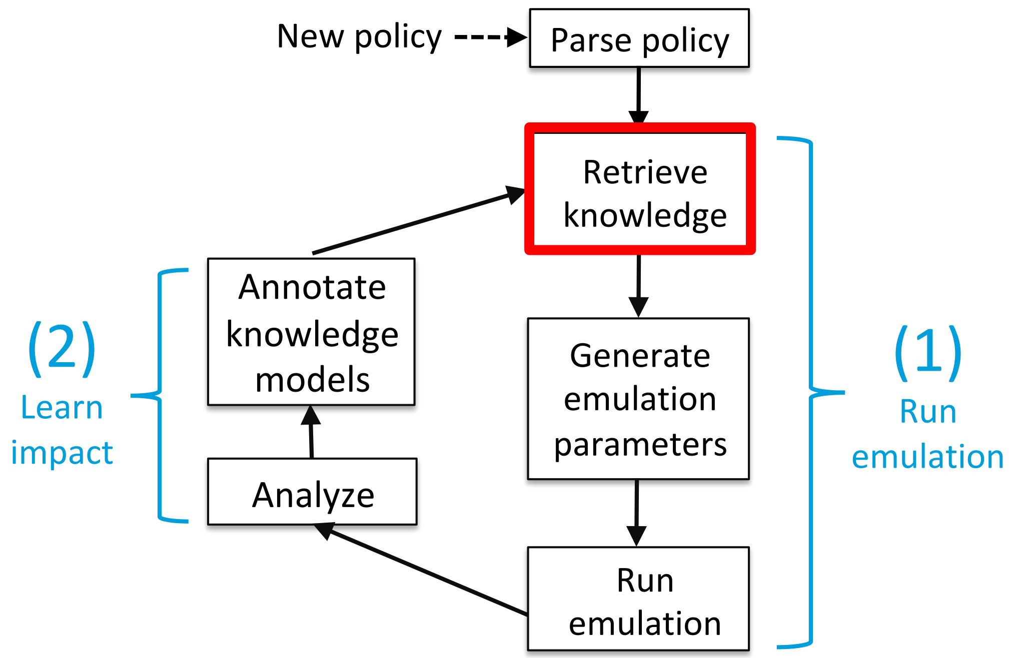
Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
     SGW.usage >= THRESH  //THRESH = 70%
then:
     SGW.scaleUp()
```

SGW

- Is this a component of a service?
- Is this hosted on compute



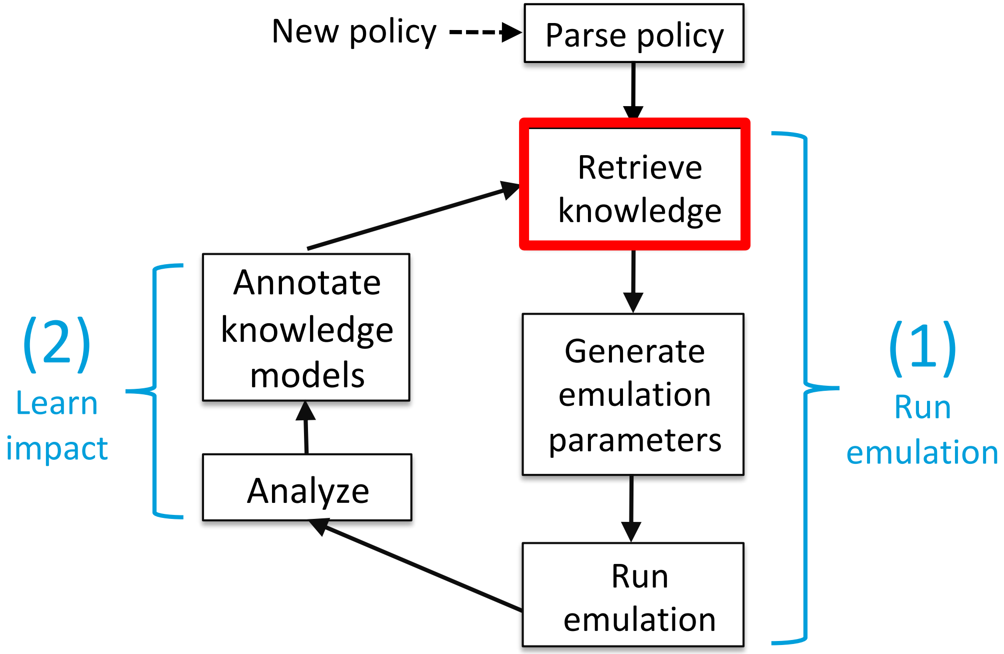
DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
     SGW.usage >= THRESH  //THRESH = 70%
then:
     SGW.scaleUp()
```

- Does it have neighbors?
- Topological, protocol peers

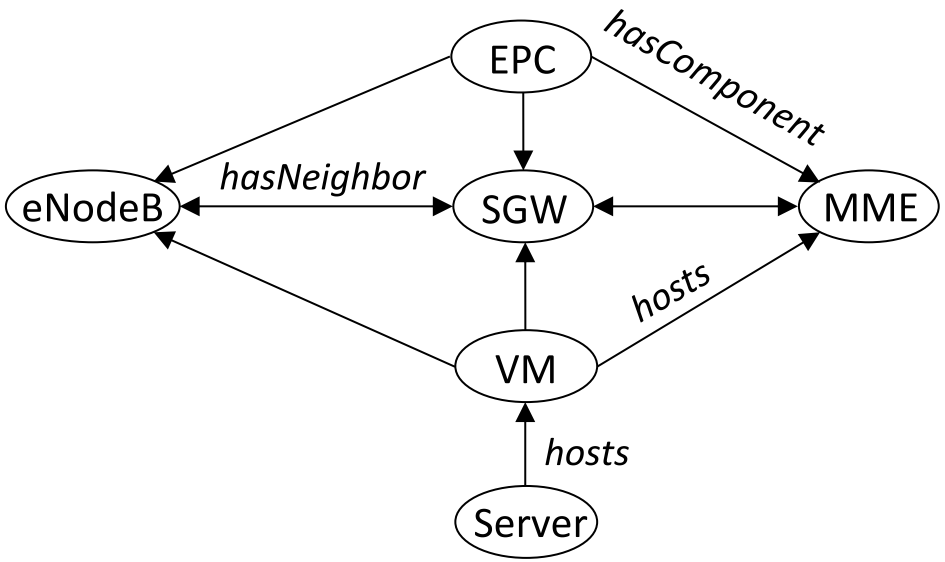


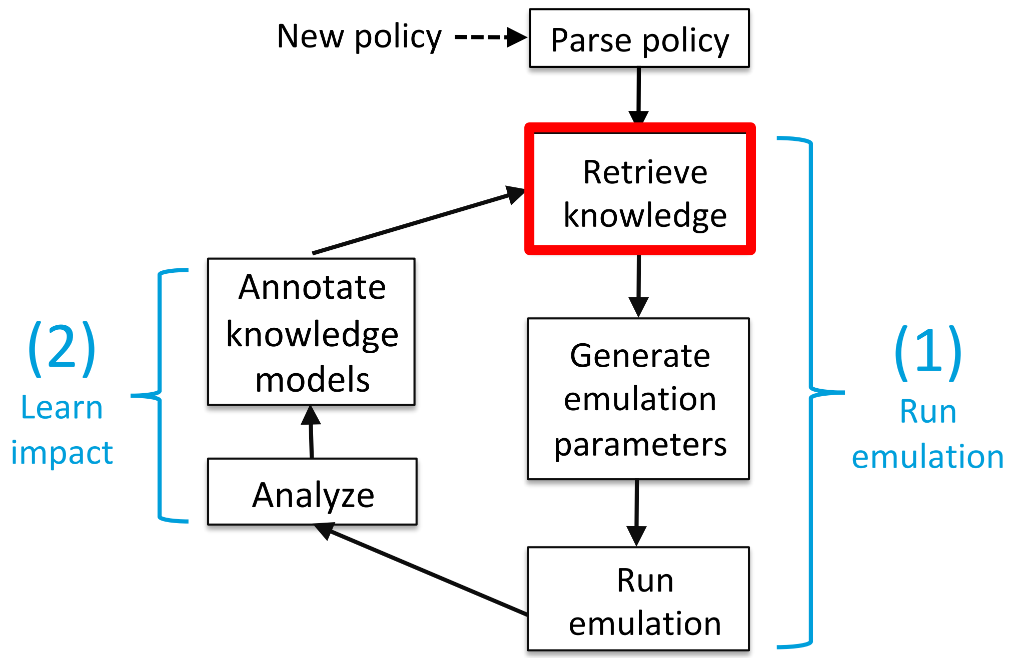
DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
     SGW.usage >= THRESH  //THRESH = 70%
then:
     SGW.scaleUp()
```



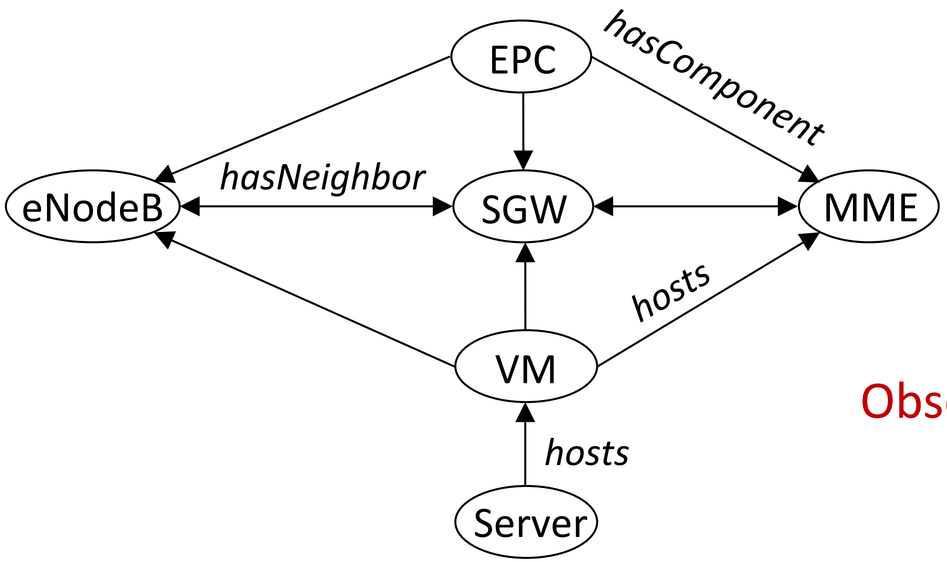


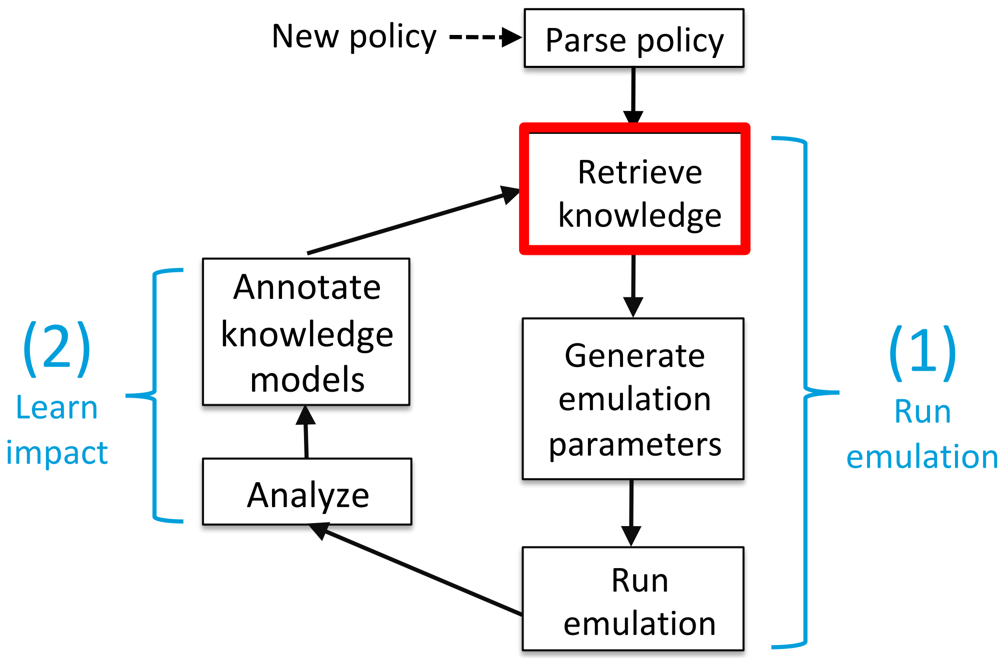
DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
     SGW.usage >= THRESH  //THRESH = 70%
then:
     SGW.scaleUp()
```





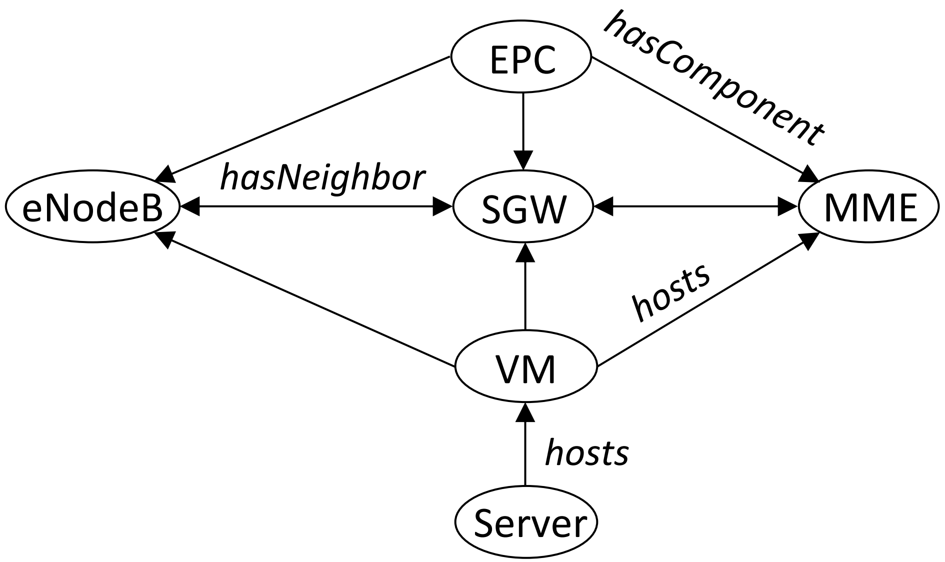
Observing impact results from emulation run later will refine this subgraph

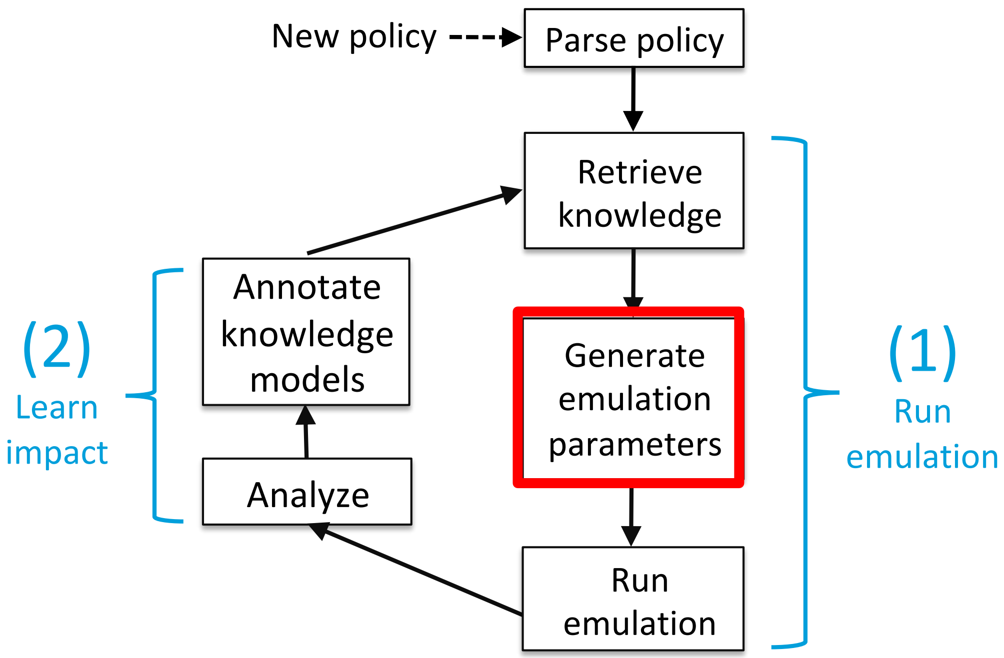
DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
     SGW.usage >= THRESH  //THRESH = 70%
then:
     SGW.scaleUp()
```





- **Variable examples from our prototype SDI**

Server	**VM**	**Switch/Link**	**VNF**	**Service**
location	location	location	location	topologyVars
status	status	status	status	numENB
rateStatusChange	rateStatusChange	ifaceVariables	type	numSGW
totalMem	totalMem	rateStatusChange	totalMem	numPGW
allocatedMem	numCPU	numFlaps	numCPU	numMME
numCPU	version	latency	version	numWebserver
numAllocatedCPU	type	totalMem	cpuUsage	latency, throughput
cpuOversubscription	cpuUsage	numCPU	memUsage	...
numAllocatedVM	memUsage	version	throughput	**Workload**
numRunningVM	migrationVars	type	latency	numUE
version, memUsage	numNeighborVM	cpuUsage	topologyVars	rateOfRequests
cpuUsage, type	osImage	memUsage	migrationVars	interarrivalTime
ifaceVars	ifaceVars	propagationDelay	cacheVars	num/rateOfMobility
...	...	...	...	concurrentVNF/Service

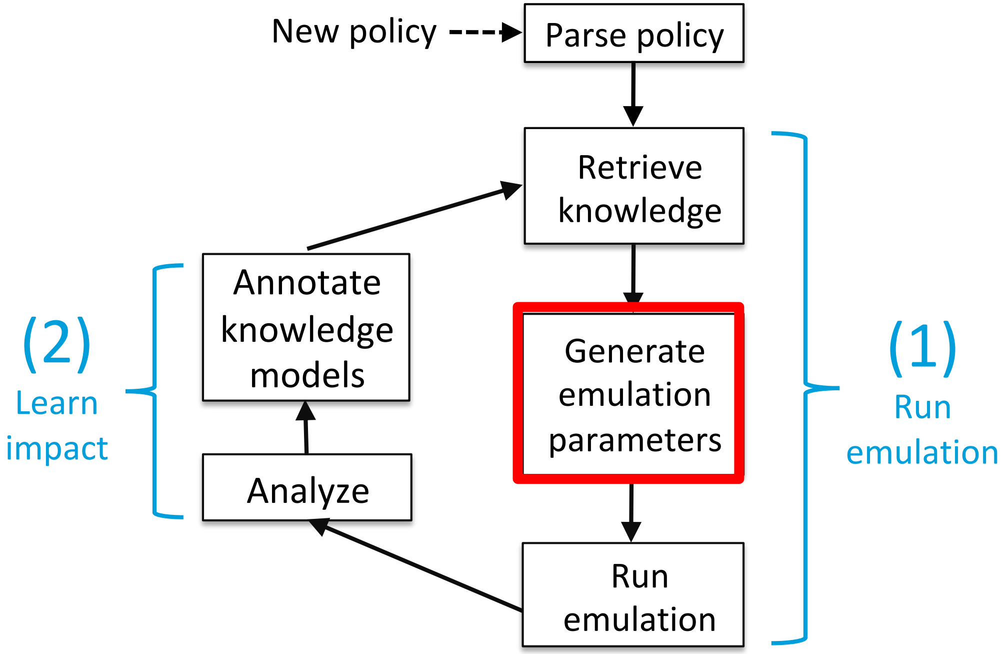
DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
     SGW.usage >= THRESH  //THRESH = 70%
then:
     SGW.scaleUp()
```

- If no prior knowledge about impact is available



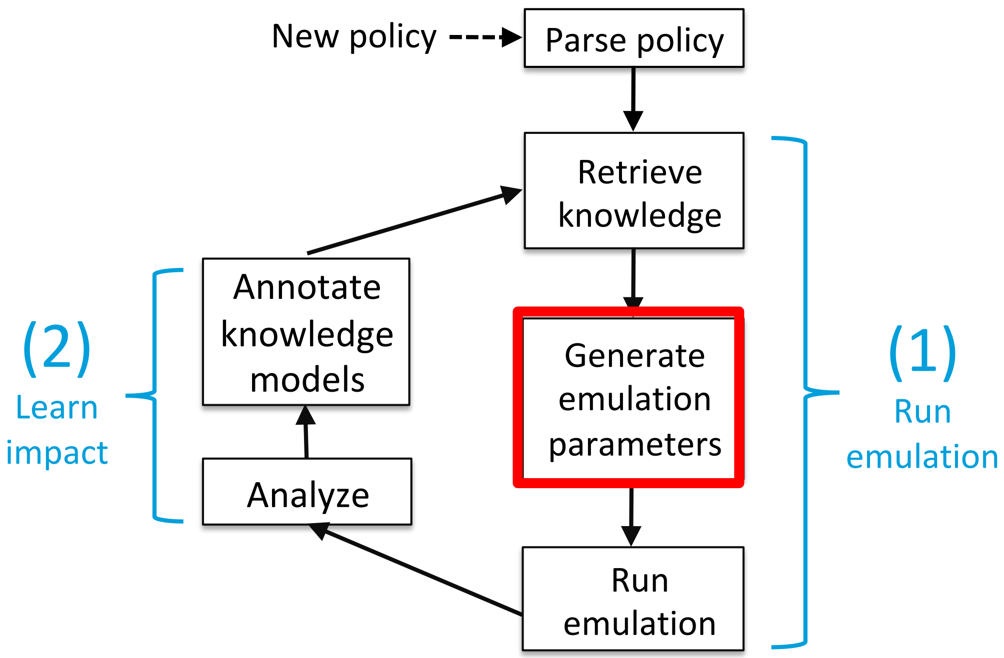
DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
      SGW.usage >= THRESH  //THRESH = 70%
then:
      SGW.scaleUp()
```

- If no prior knowledge about impact is available
  - Randomly pick a small set of emulation parameter values
- If prior knowledge about impact is available then greedily vary emulation parameters that cause the most (good/bad) impact for this policy

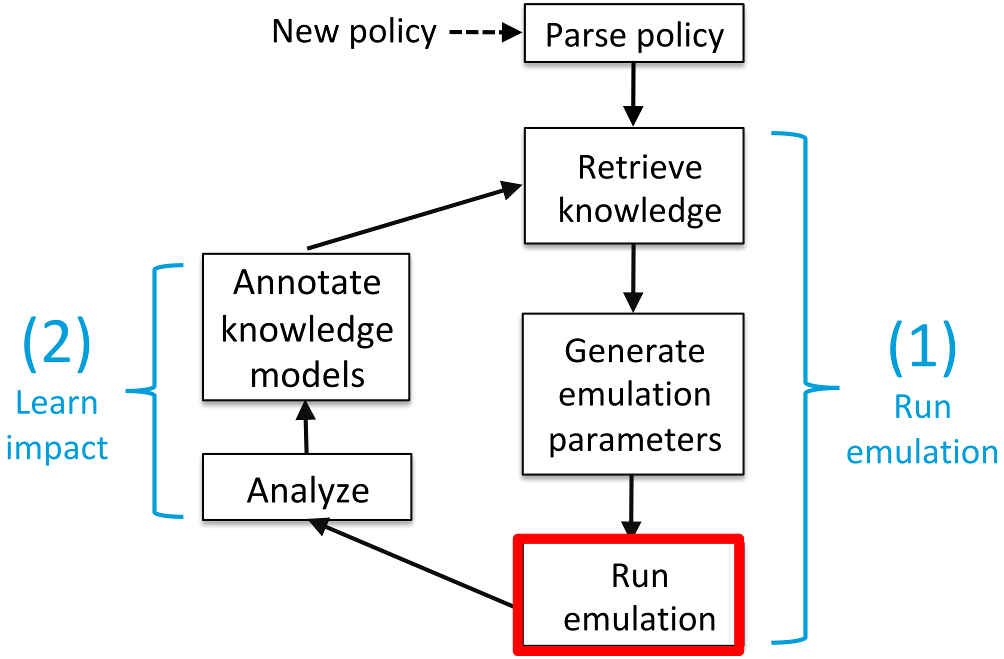


DEPO: Impact Learning Approach

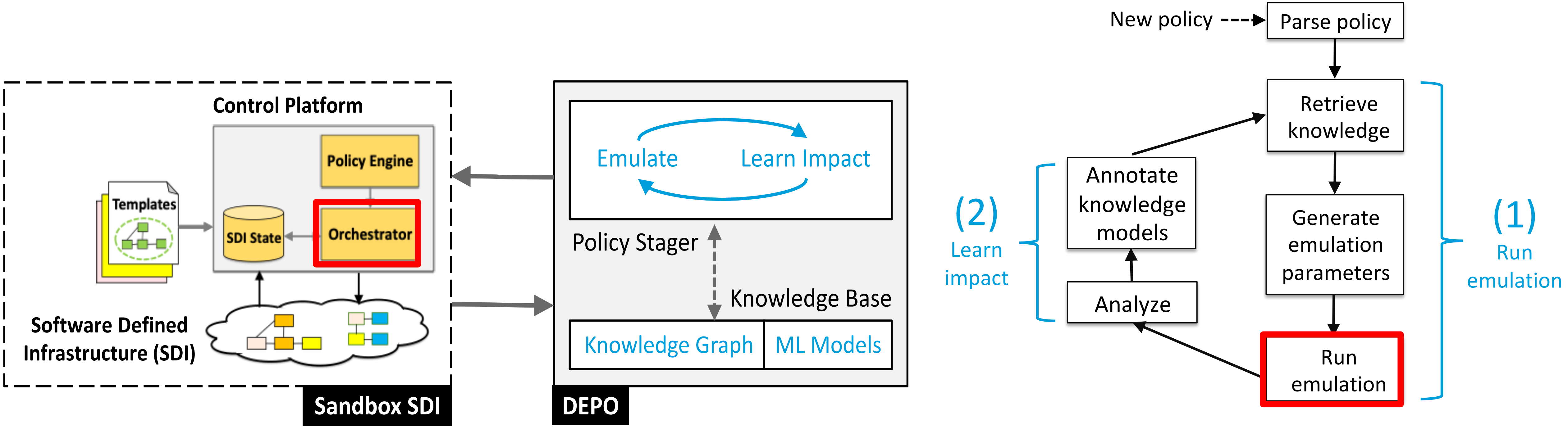
Learning impact of policies

- Affected variables learned for SGW scale up policy

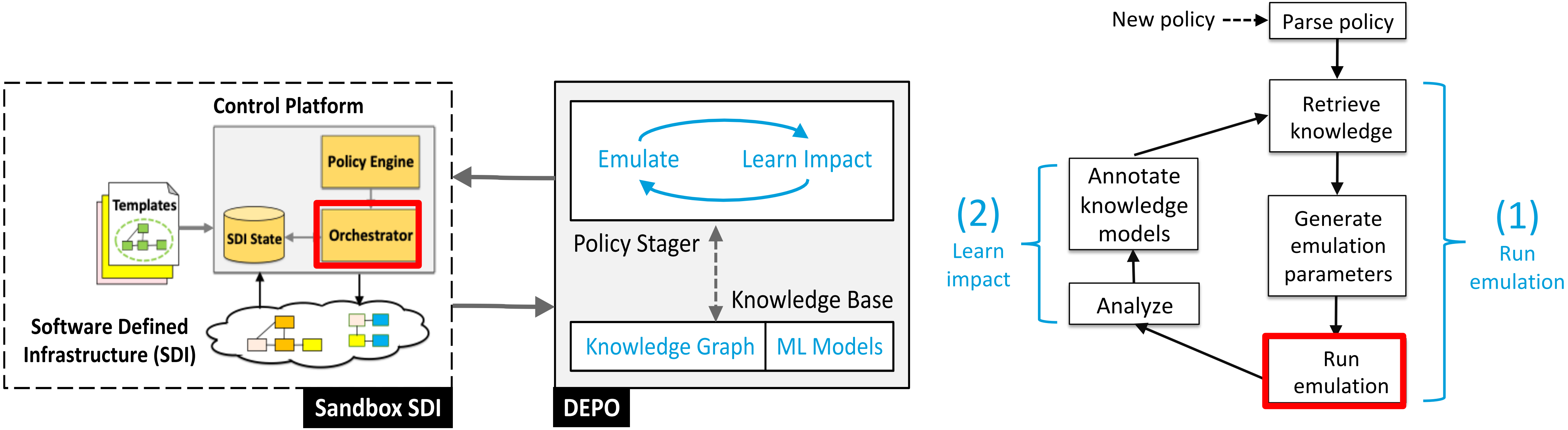
```
when:
      SGW.usage >= THRESH  //THRESH = 70%
then:
      SGW.scaleUp()
```



DEPO: Impact Learning Approach



DEPO: Impact Learning Approach



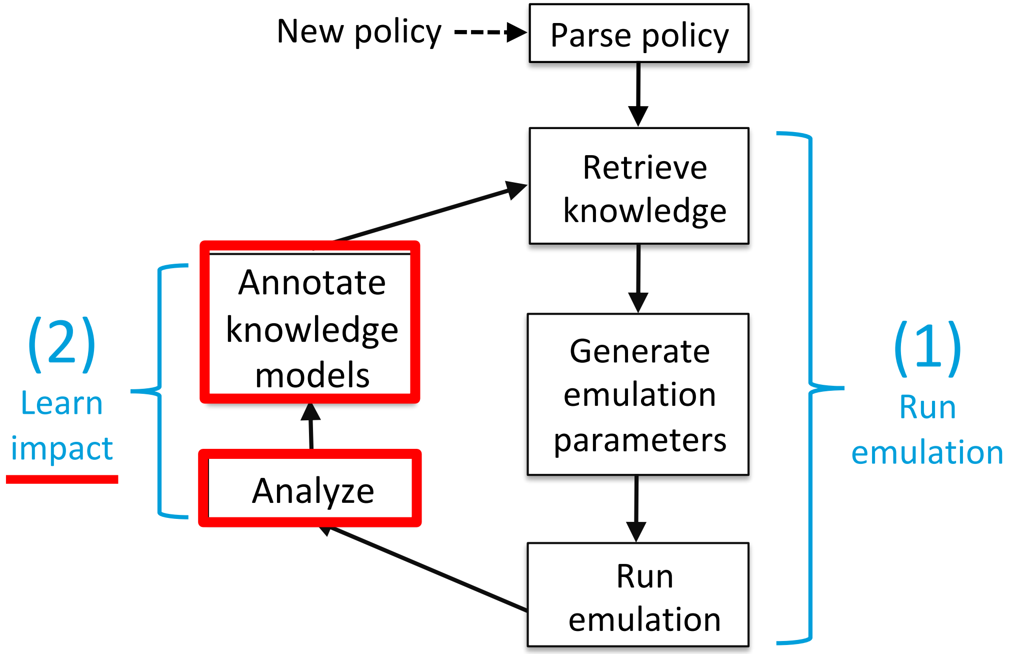
- Traffic generator parameters
  - Can be set based on past domain expertise
  - Or let DEPO consider these part of emulation parameters

DEPO: Impact Learning Approach

Learning impact of policies

- Affected variables learned for SGW scale up policy

```
when:
      SGW.usage >= THRESH  //THRESH = 70%
then:
      SGW.scaleUp()
```



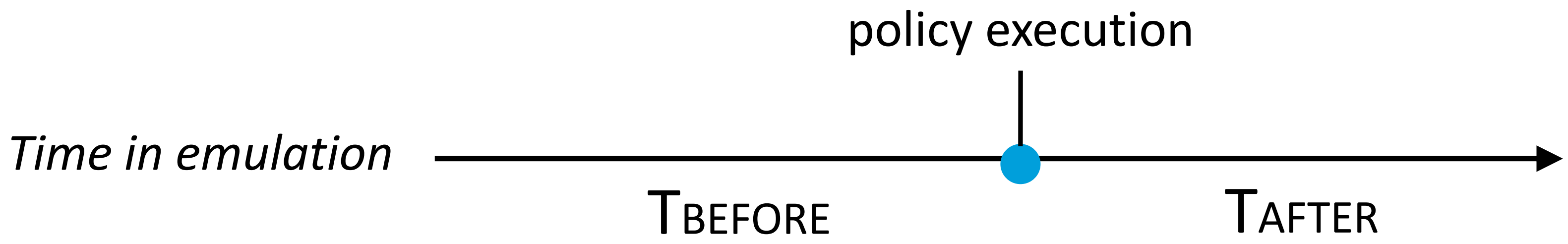
(3) Learn Impact

- Logs collected include
  - Traces of policy executions
  - Configurable and observed variable logs for object instances – state changes in time

- Learn impact using
  1. Generating ML models
  2. Change analysis

3) Learn Impact

1) **Change analysis**
   - Perform *before – after* change analysis for each variable to get course-grained impact info
   - Note statistically significant difference (e.g., 95% confidence) by comparing CDFs collected from before, and after policy triggerings



3) Learn Impact

1) **Change analysis**
   - Perform *before – after* change analysis for each variable to get course-grained impact info
   - Note statistically significant difference (e.g., 95% confidence) by comparing CDFs collected from before, and after policy triggerings

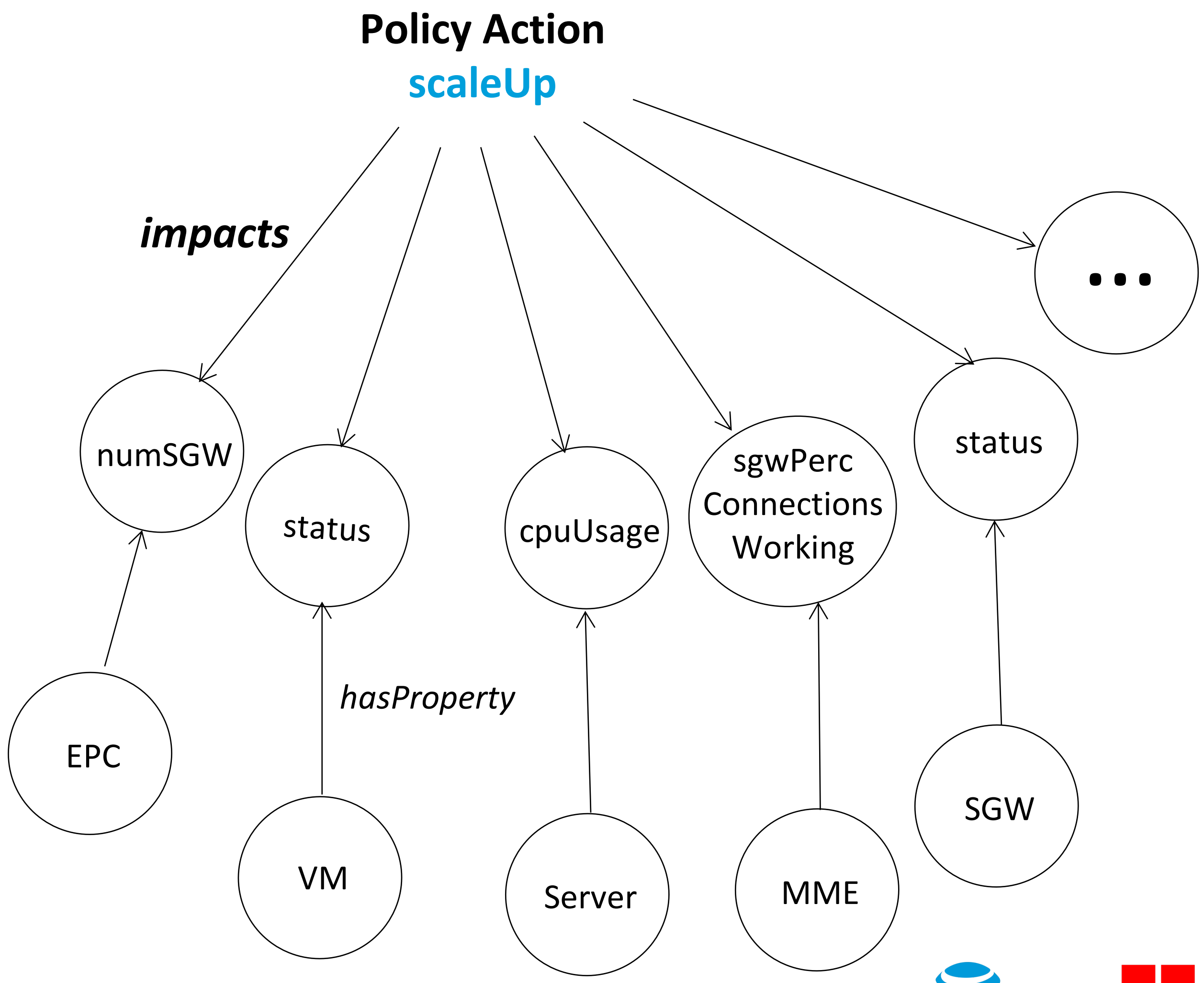
3) Learn Impact

1) **Change analysis**
   - Perform *before – after* change analysis for each variable to get course-grained impact info
   - Note statistically significant difference (e.g., 95% confidence) by comparing CDFs collected from before, and after policy triggerings
     - Kolmogorov-Smirnov 2-sample goodness of fit test
       - Generic test that works across variable types (makes no assumptions about distribution of data – worked well in our evaluations
       - Alternate specialized mechanisms can be plugged in here
     - Very parallelizable – large number of instance logs can be processed in parallel
   - Output the list of impacted object instances

3) Learn Impact

Impact knowledge graph facts → Alternate representation (partial for demo)

```
PolicyAction:scaleUpSGW
       Server:
              server_numVM
              server_cpuUsage
              server_memUsage
              server_allocated_mem
              server_num_allocated_cpu
       VM:
              vm_status
              vm_networkConfig
              vm_cpuUsage
              vm_memUsage
       EPC:
              epc_numSGW
       SGW:
              sgw_status
              sgw_networkConfig
              sgw_mmeNumConnections
              sgw_mmePercentOfConnectionsWorking
              sgw_cpuUsage
              sgw_memUsage
       MME:
              mme_numSGW
              mme_sgwNumConnections
              mme_sgwPercentOfConnectionsWorking
```



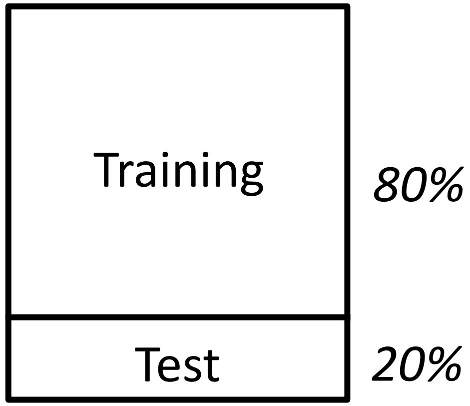
3) Learn Impact

2) Generating ML models

- Auto-generate ML models
  - Multiple linear regression
  - SVM
  - Random Forest

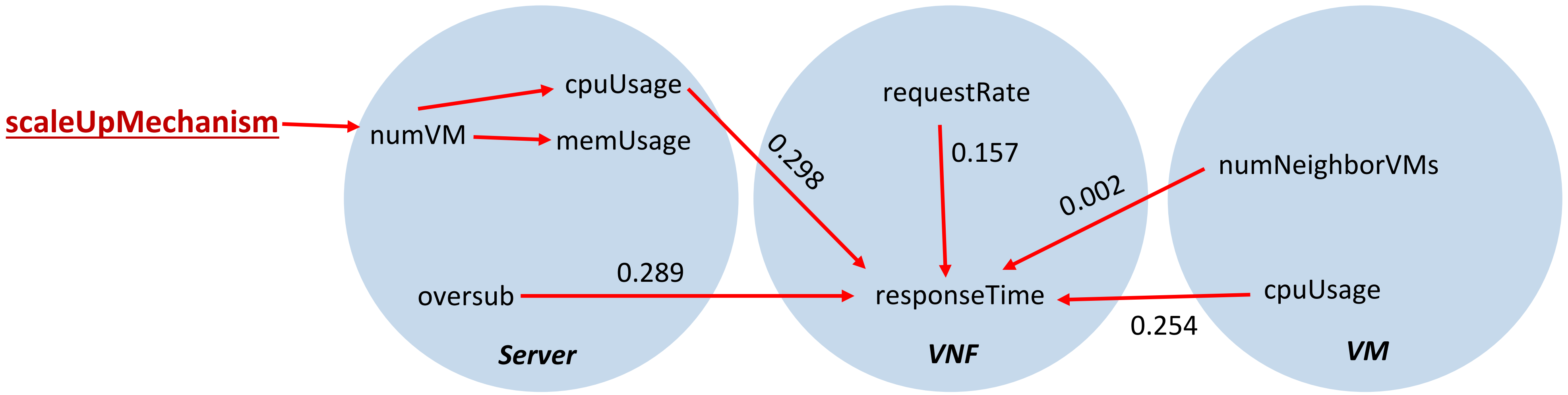
- Easier to auto generate as part of workflow
- Capture both linear and non-linear relationships
- Model for each impacted variable
  - Compute accuracy using training/test division of logs
  - Kfold cross validation for computing accuracy
  - Grid search for tuning on the model parameters

Emulation Logs



3) Learn Impact

2) Generating ML models

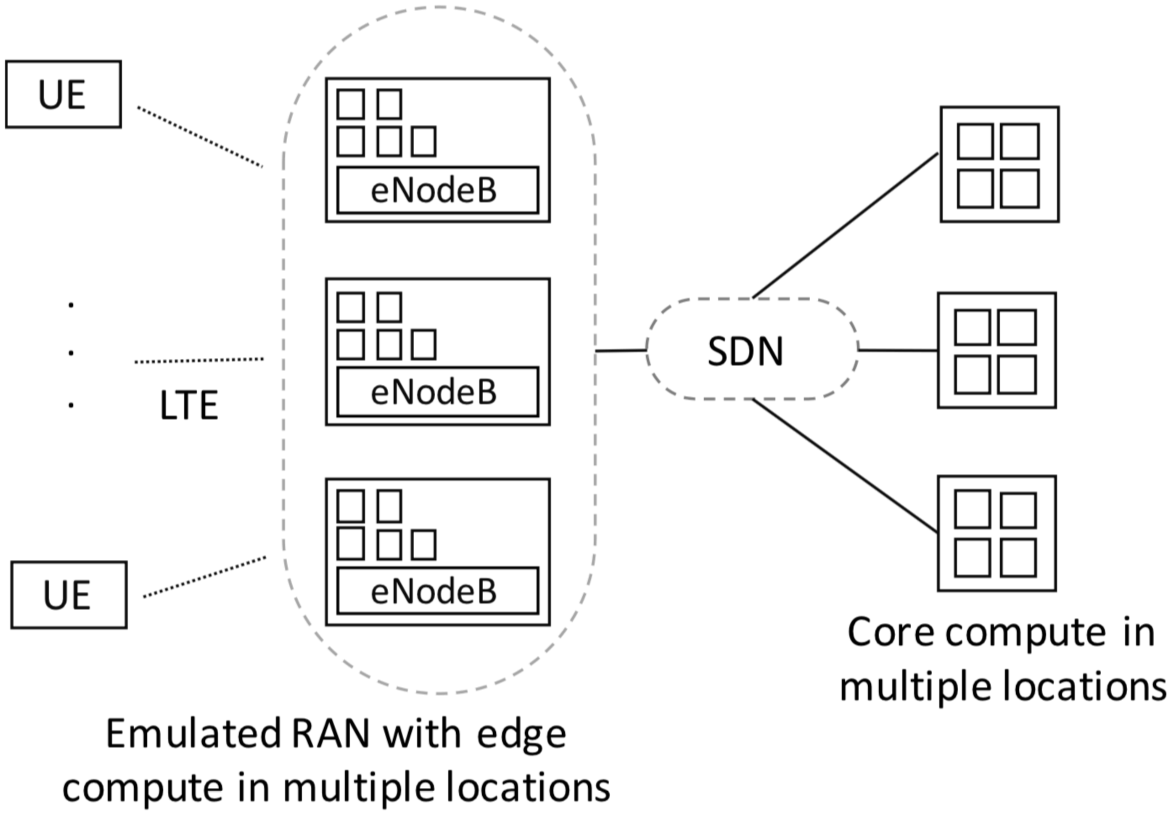


Quantified variable-level impact

- Continuous learning from emulations improves impact knowledge over time
  - Leads to knowledge correction over time

Evaluations

- Sandbox SDI created in PhantomNet mobility testbed
  - Emulated RAN and core with multiple locations
  - Different servers and VM configs
- Extended SDI orchestrator and templates available in the community
  - Exposed more variables/mechanisms for EPC and SMORE services
  - Created SDI policies (Drools)



AT&T

- **Example policies**

Listing 1: Update server

```
when:
   Server.updateAvailable == True AND
   Server.locatedAtEdge == True
then:
   Server.update()
```

Listing 2: Oversubscribe

```
when:
   Server.cpuUsage_10minAvg < THRESH1
   //THRESH1 = 50%
then:
   Server.setCPU_Oversub(oversubPerc =
          THRESH2) //THRESH2 = 50%
```

Listing 3: Scaling SGW

```
when:
   SGW.cpuUsage >= THRESH
   //THRESH = 70%
then:
   EPC.scaleUpSGW()
```

Listing 4: Scaling SMORE

```
when:
   SMORE_Webserver.cpuUsage_5minAvg
             >= THRESH  //THRESH = 70%
then:
   SMORE.scaleUpWebserver()
```

Listing 5: SMORE caching

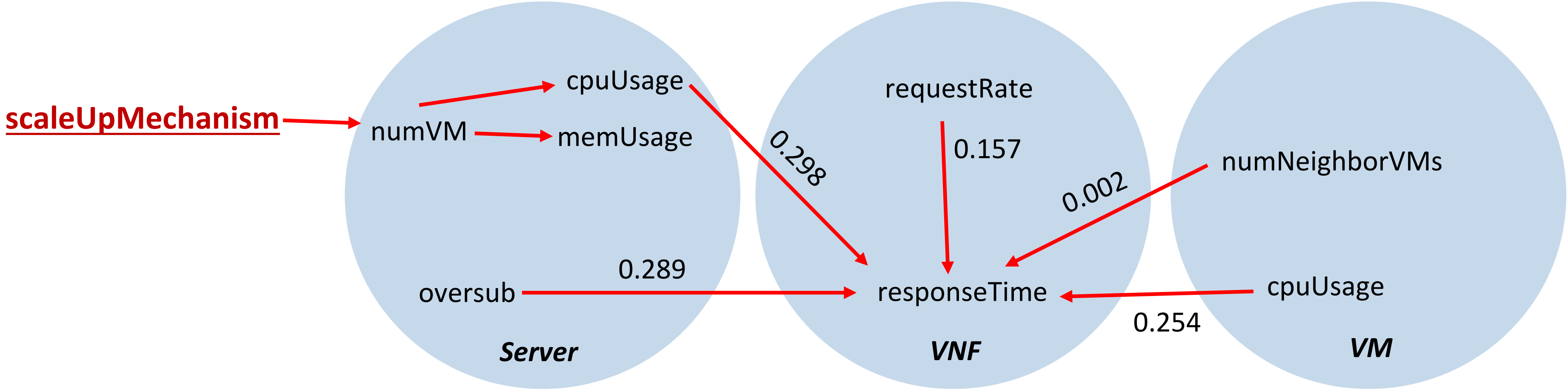
```
when:
   SMORE.subscriberLatency_5minAvg >=
          THRESH
   //THRESH = 30ms
then:
   SMORE.setCaching()
```

Listing 6: SMORE offload

```
when:
   EPC.subscriberLatency_10minAvg >=
          THRESH
   //THRESH = 30ms
then:
   EPC.augmentSMORE(subscriberList)
```

Evaluations

2) Generating ML models



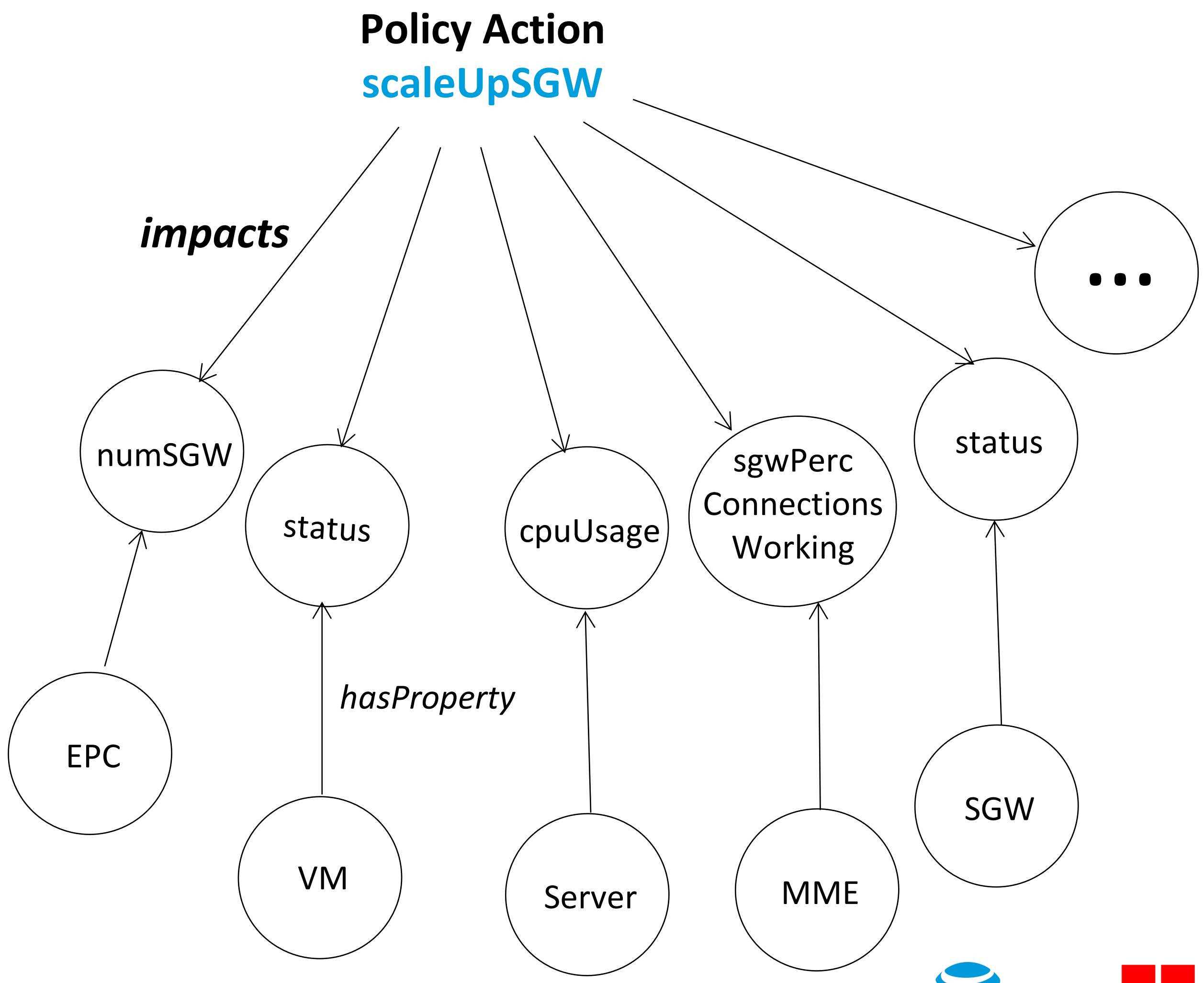
Quantified variable-level impact

- Continuous learning from emulations improves impact knowledge over time
  - Leads to knowledge correction over time

Evaluations

Impact knowledge graph facts → Alternate representation (partial for demo)

```
PolicyAction:scaleUpSGW
        Server:
                server_numVM
                server_cpuUsage
                server_memUsage
                server_allocated_mem
                server_num_allocated_cpu
        VM:
                vm_status
                vm_networkConfig
                vm_cpuUsage
                vm_memUsage
        EPC:
                epc_numSGW
        SGW:
                sgw_status
                sgw_networkConfig
                sgw_mmeNumConnections
                sgw_mmePercentOfConnectionsWorking
                sgw_cpuUsage
                sgw_memUsage
        MME:
                mme_numSGW
                mme_sgwNumConnections
                mme_sgwPercentOfConnectionsWorking
```



Evaluations

- **Example policies**

Listing 1: Update server

```
when:
   Server.updateAvailable == True AND
   Server.locatedAtEdge == True
then:
   Server.update()
```

Listing 2: Oversubscribe

```
when:
   Server.cpuUsage_10minAvg < THRESH1
   //THRESH1 = 50%
then:
   Server.setCPU_Oversub(oversubPerc =
          THRESH2) //THRESH2 = 50%
```

Listing 3: Scaling SGW

```
when:
   SGW.cpuUsage >= THRESH
   //THRESH = 70%
then:
   EPC.scaleUpSGW()
```

Listing 4: Scaling SMORE

```
when:
   SMORE_Webserver.cpuUsage_5minAvg
              >= THRESH  //THRESH = 70%
then:
   SMORE.scaleUpWebserver()
```

Listing 5: SMORE caching

```
when:
   SMORE.subscriberLatency_5minAvg >=
          THRESH
   //THRESH = 30ms
then:
   SMORE.setCaching()
```

Listing 6: SMORE offload

```
when:
   EPC.subscriberLatency_10minAvg >=
          THRESH
   //THRESH = 30ms
then:
   EPC.augmentSMORE(subscriberList)
```

Evaluations

Impact knowledge graph facts

```
PolicyAction:setCaching
        SMORE:
                smore_cachingStatus
                smore_latency
        SMORE_Loadbalancer:
                smore_loadbalancer_cachingStatus
                smore_loadbalancer_availableCacheSize
                smore_loadbalancer_usedCacheSize
                smore_loadbalancer_resourceFrequencyOfAccess
        SMORE_Webserver:
                smore_webserver_status
                smore_webserver_networkConfig
                smore_webserver_loadbalancerConnectionStatus
                smore_webserver_cpuUsage
                smore_webserver_memUsage
                smore_webserver_resourceFrequencyOfAccess
```

Evaluations

Impact knowledge graph facts

PolicyAction:update	Server.[status, version, numVM, memUsage, cpuUsage, percentFailedMigrations], VM.[status, memUsage, cpuUsage], VNF.[status, memUsage, cpuUsage], Service.[smoreLatency]
Server.migrateVM	Server.[numVM, memUsage, cpuUsage, percentFailedMigrations], VM.[status, memUsage, cpuUsage], VNF.[status, memUsage, cpuUsage], Service.[smoreLatency]
Server.sendVM	Server.[numVM, memUsage, cpuUsage], VM.[status, memUsage, cpuUsage], VNF.[status, memUsage, cpuUsage], Service.[smoreLatency]
Server.receiveVM	Server.[numVM, memUsage, cpuUsage, percentFailedMigrations], VM.[status, memUsage, cpuUsage], VNF.[status, memUsage, cpuUsage], Service.[smoreLatency]
VM.stop	Server.[memUsage, cpuUsage], VM.[status, memUsage, cpuUsage], VNF.[status, memUsage, cpuUsage], Service.[smoreLatency]
Server.installUpdate	Server.[status, version, memUsage, cpuUsage]
Server.reboot	Server.[status, memUsage, cpuUsage]
VM.start	Server.[memUsage, cpuUsage], VM.[status, memUsage, cpuUsage]
VNF.start	Server.[memUsage, cpuUsage], VM.[memUsage, cpuUsage], VNF.[status, memUsage, cpuUsage], Service.[smoreLatency]

Evaluations

Impact knowledge graph facts

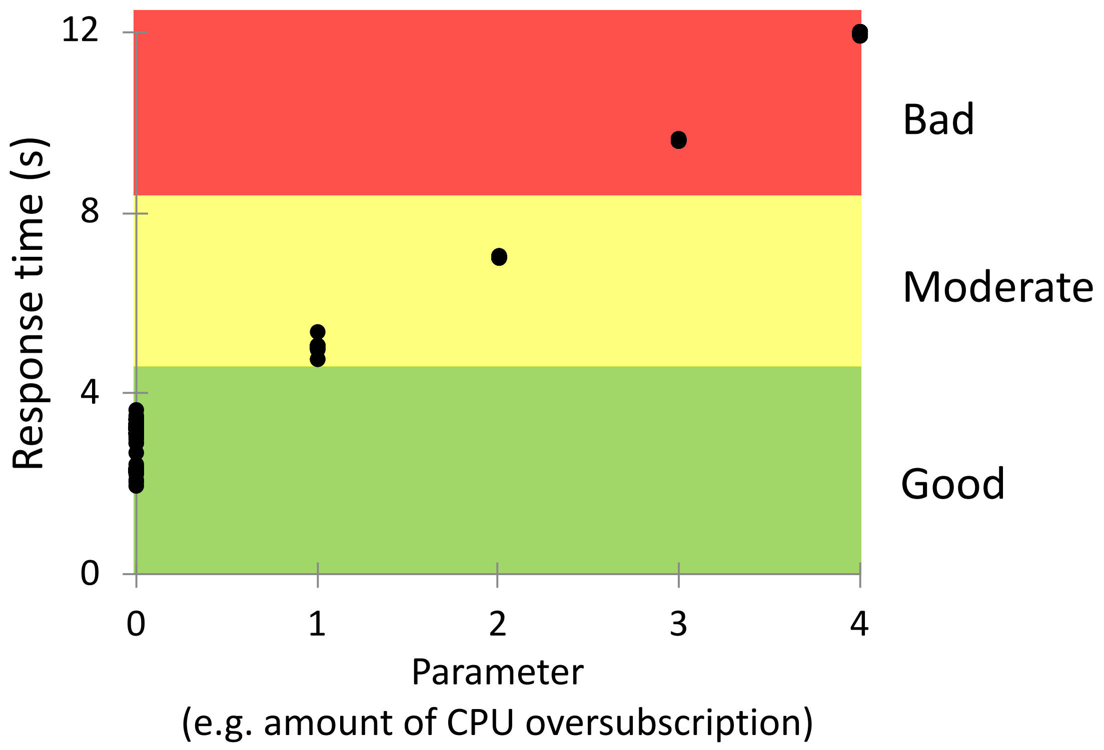
```
PolicyAction:augmentSMORE
        Server:
                server_numVM
                server_cpuUsage
                server_memUsage
                server_allocated_mem
                server_num_allocated_cpu
        VM:
                vm_status
                vm_networkConfig
                vm_cpuUsage
                vm_memUsage
        SMORE:
                smore_latency
        SMORE_Loadbalancer:
                smore_loadbalancer_webserverNumConnections
                smore_loadbalancer_webserverPercentOfConnectionsWorking
                smore_loadbalancer_resourceFrequencyOfAccess
        SMORE_Switch:
                smore_switch_status
                smore_switch_routeConfig
        SMORE_Webserver:
                smore_webserver_status
                smore_webserver_networkConfig
                smore_webserver_loadbalancerConnectionStatus
                smore_webserver_cpuUsage
                smore_webserver_memUsage
                smore_webserver_resourceFrequencyOfAccess
```

Evaluations

- **Variable examples from our prototype SDI**

Server	**VM**	**Switch/Link**	**VNF**	**Service**
location	location	location	location	topologyVars
status	status	status	status	numENB
rateStatusChange	rateStatusChange	ifaceVariables	type	numSGW
totalMem	totalMem	rateStatusChange	totalMem	numPGW
allocatedMem	numCPU	numFlaps	numCPU	numMME
numCPU	version	latency	version	numWebserver
numAllocatedCPU	type	totalMem	cpuUsage	latency, throughput
cpuOversubscription	cpuUsage	numCPU	memUsage	...
numAllocatedVM	memUsage	version	throughput	**Workload**
numRunningVM	migrationVars	type	latency	numUE
version, memUsage	numNeighborVM	cpuUsage	topologyVars	rateOfRequests
cpuUsage, type	osImage	memUsage	migrationVars	interarrivalTime
ifaceVars	ifaceVars	propagationDelay	cacheVars	num/rateOfMobility
...	...	...	...	concurrentVNF/Service

Evaluations



Evaluations

Determining environment variables that are major impact contributors

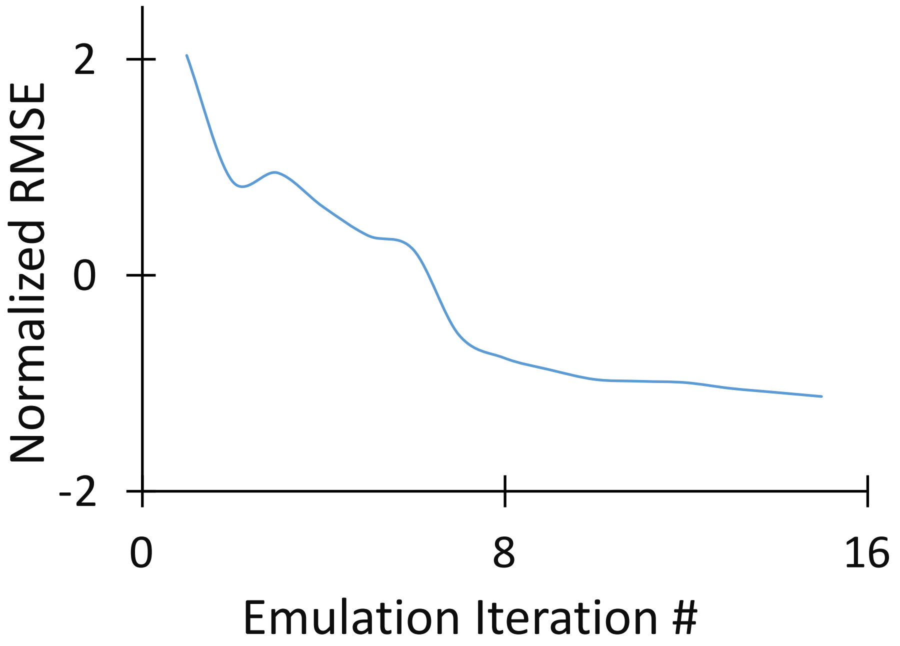
- Knowing this helps in greedily varying env variables that show policy's worst impact
- For impact of server CPU oversubscription policy on service response time:

Variables	Relative Importance
VM.cpuUsage	0.254
VM.numNeighborVMs	0.002
Server.cpuUsage(aggregate)	0.298
Server.cpuOvsersubscription	0.289
Service.requestRate	0.157

Evaluations

Improved learning over time from emulations using iterative approach

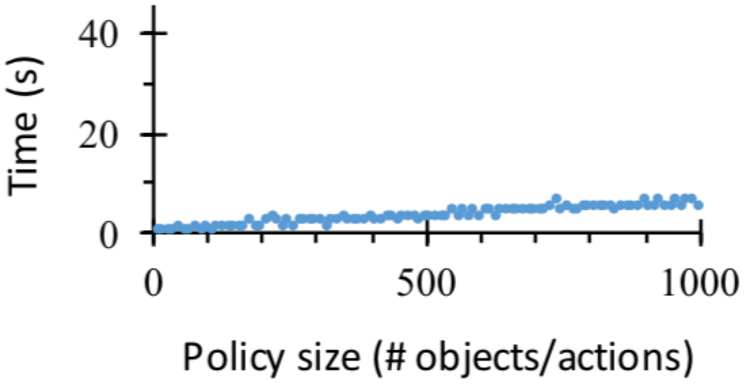
- Decreasing average RMSE over emulation iterations
    - Averaged from models for service response time for the 6 policy examples



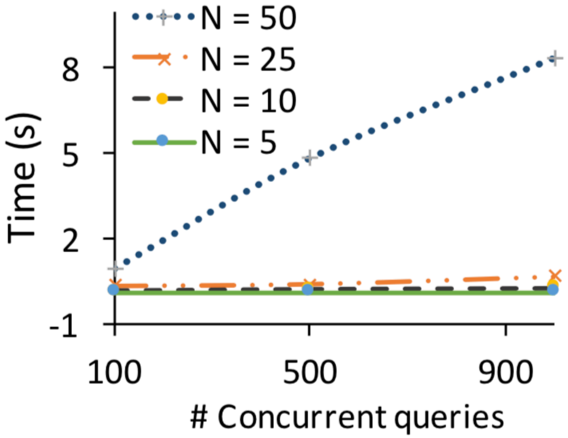
Evaluations

Performance and scalability

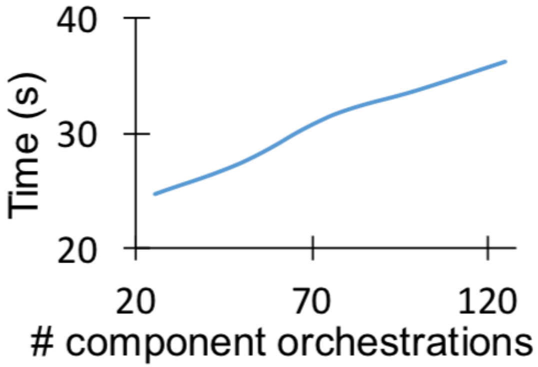
- Parallelizable and scalable DEPO process – reasonable bottleneck of emulation environment orchestration



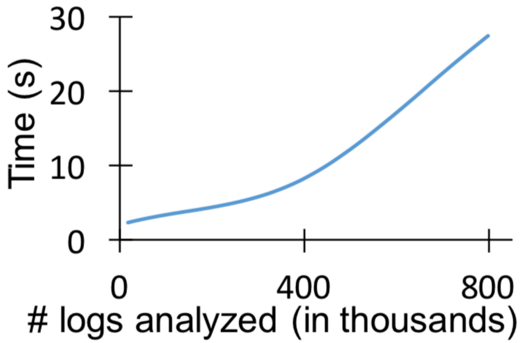
Policy parsing



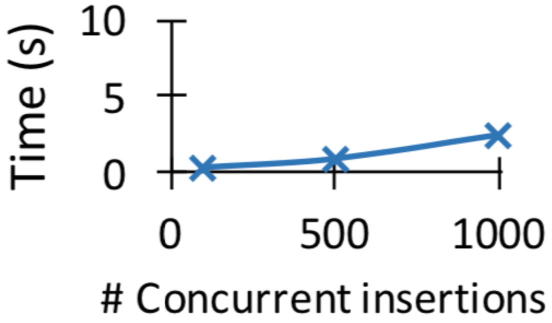
Emulation parameter generation



Emulation orchestration



Data analysis after emulation



KG insertion (updating knowledge)

Conclusion

- We presented DEPO
  - given SDI orchestration and service-level policies,
  - allows us to determine and quantify their impact on objects in an SDI.
- DEPO uses a systematic approach to modeling domain knowledge
  - Enables more informed policy writing and policy impact checking
  - Further annotate / verify initial model by incorporating knowledge learnt using emulations
- DEPO uses statistical analysis and machine learning to enable knowledge-based inference and reasoning
  - Describes how a given SDI policy affects the SDI objects
- We prototyped DEPO and evaluated it on a testbed with policies from realistic usecase services in a sandbox SDI

AT&T