



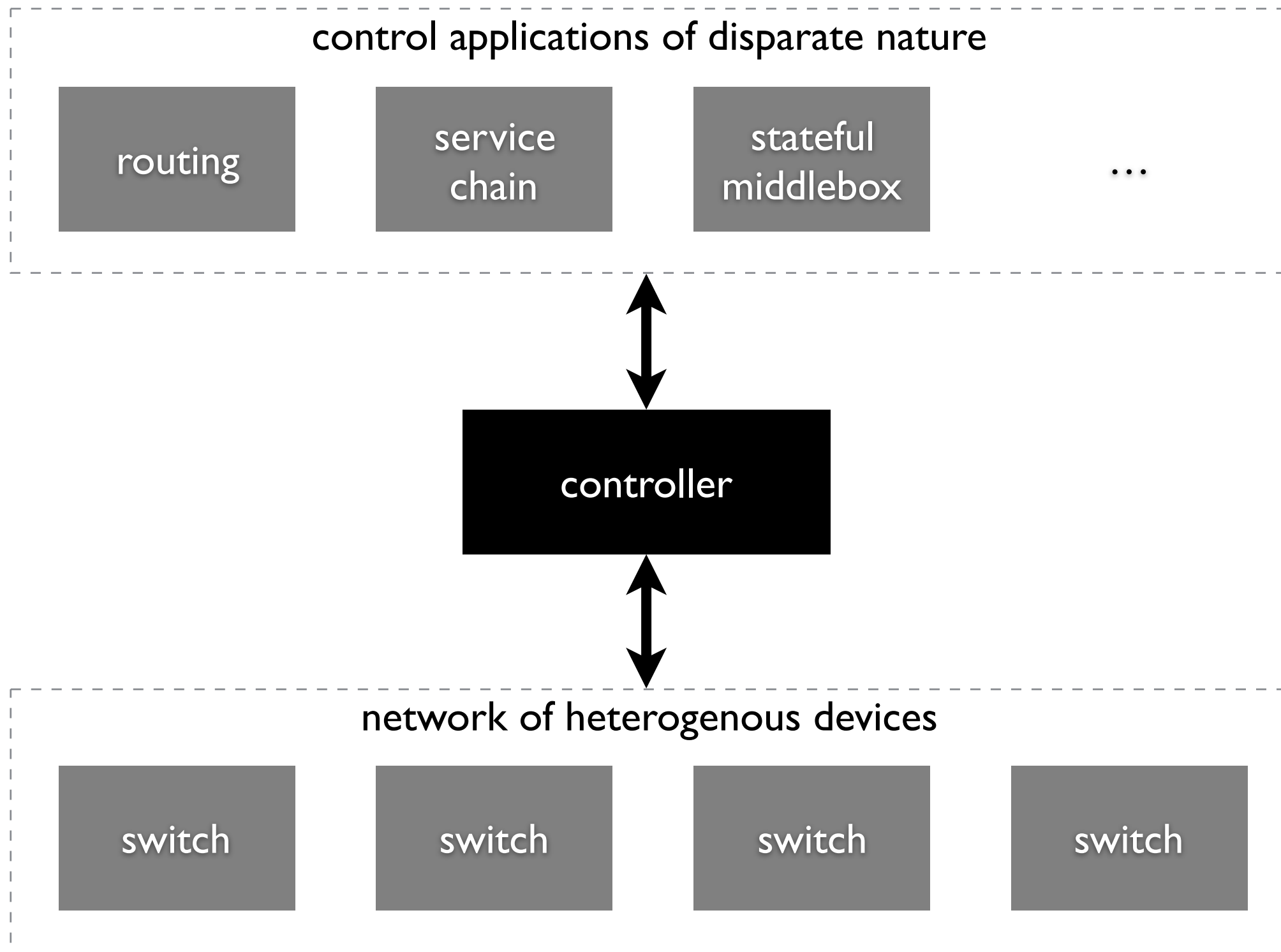
Ravel: a database-defined network

Anduo Wang* Xueyuan Mei† Jason Croft†
Matthew Caesar† Brighten Godfrey†

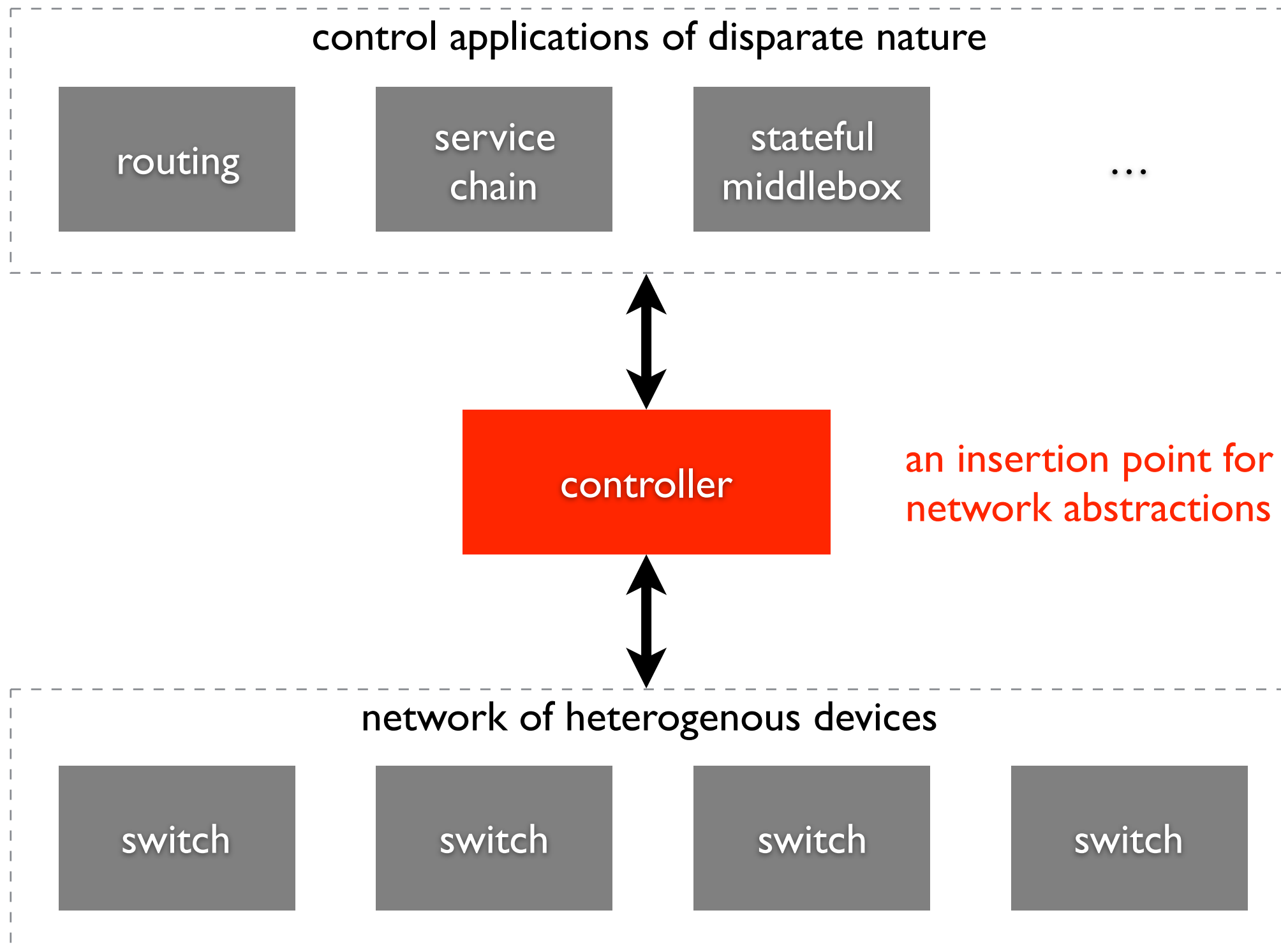
**Temple University*

†University of Illinois Urbana-Champaign

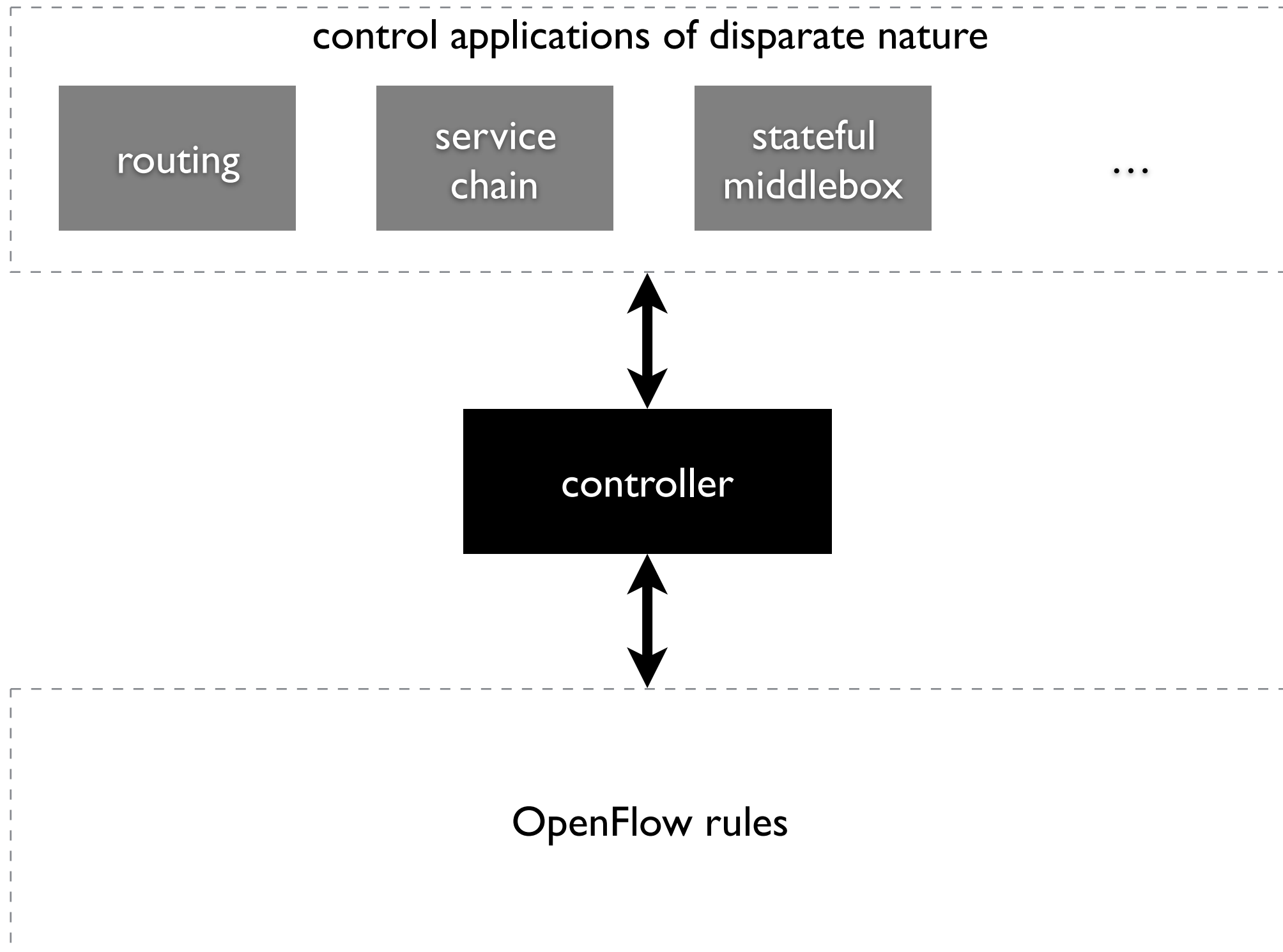
software-defined network



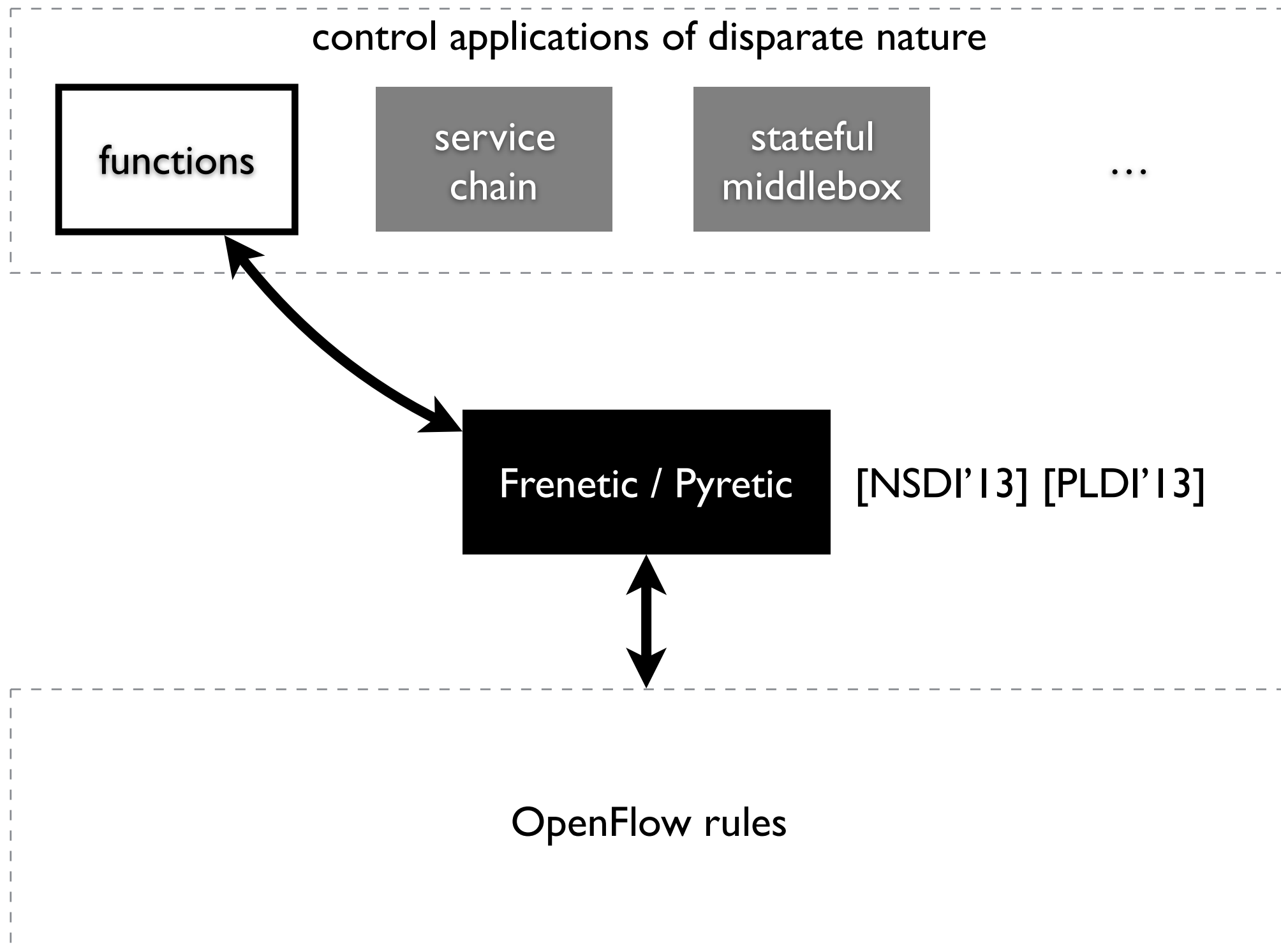
software-defined network



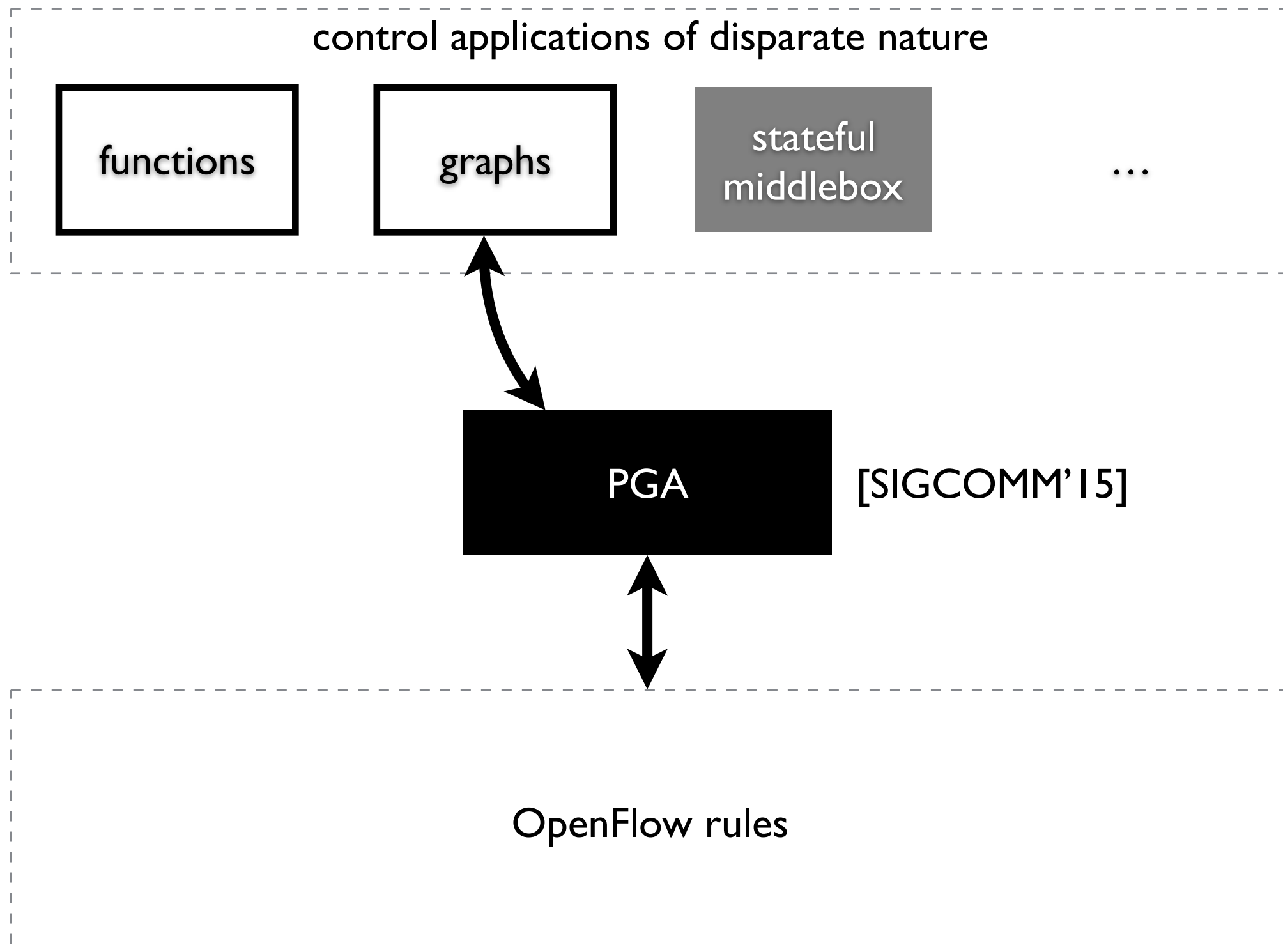
abstractions



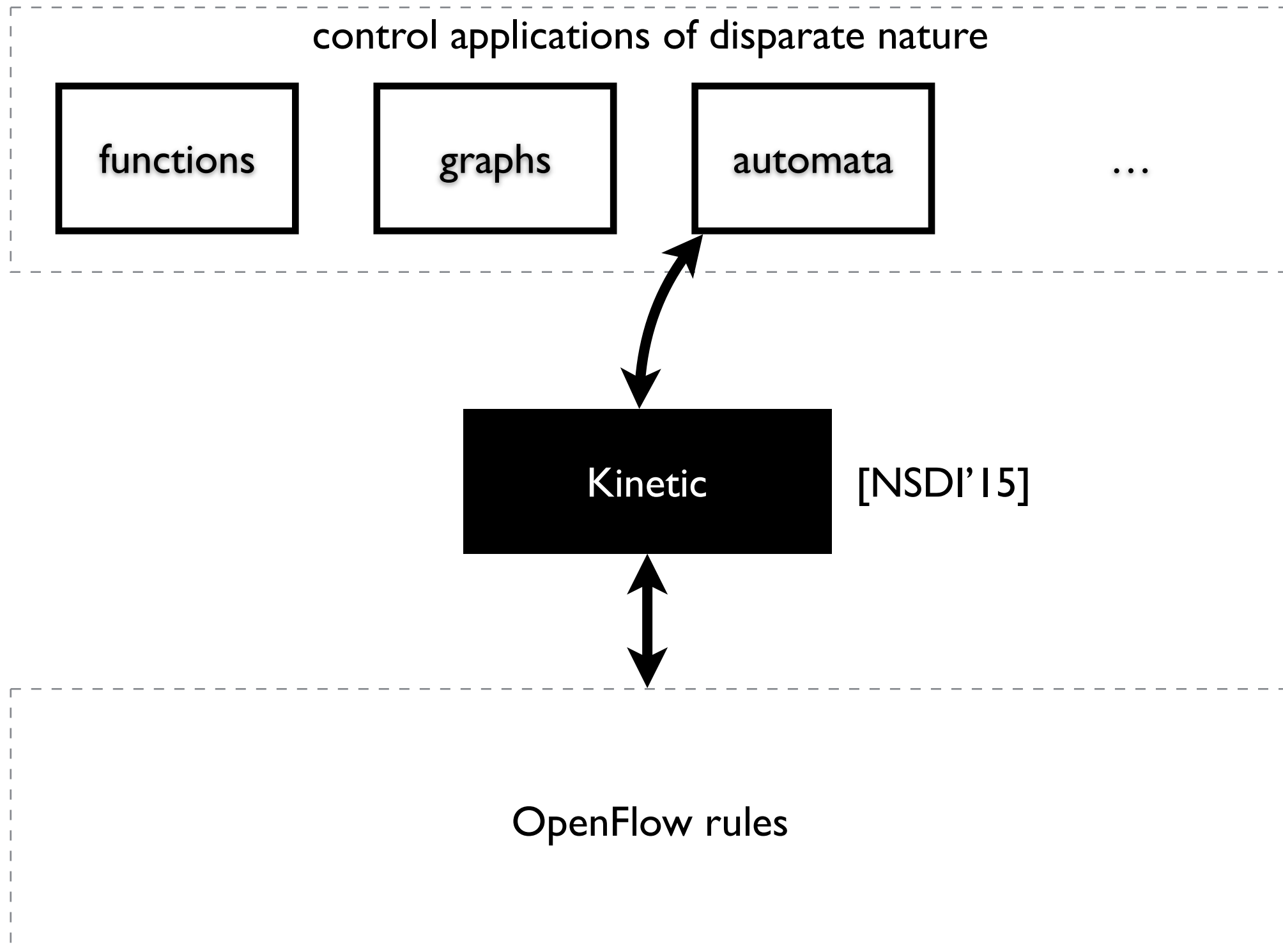
abstractions



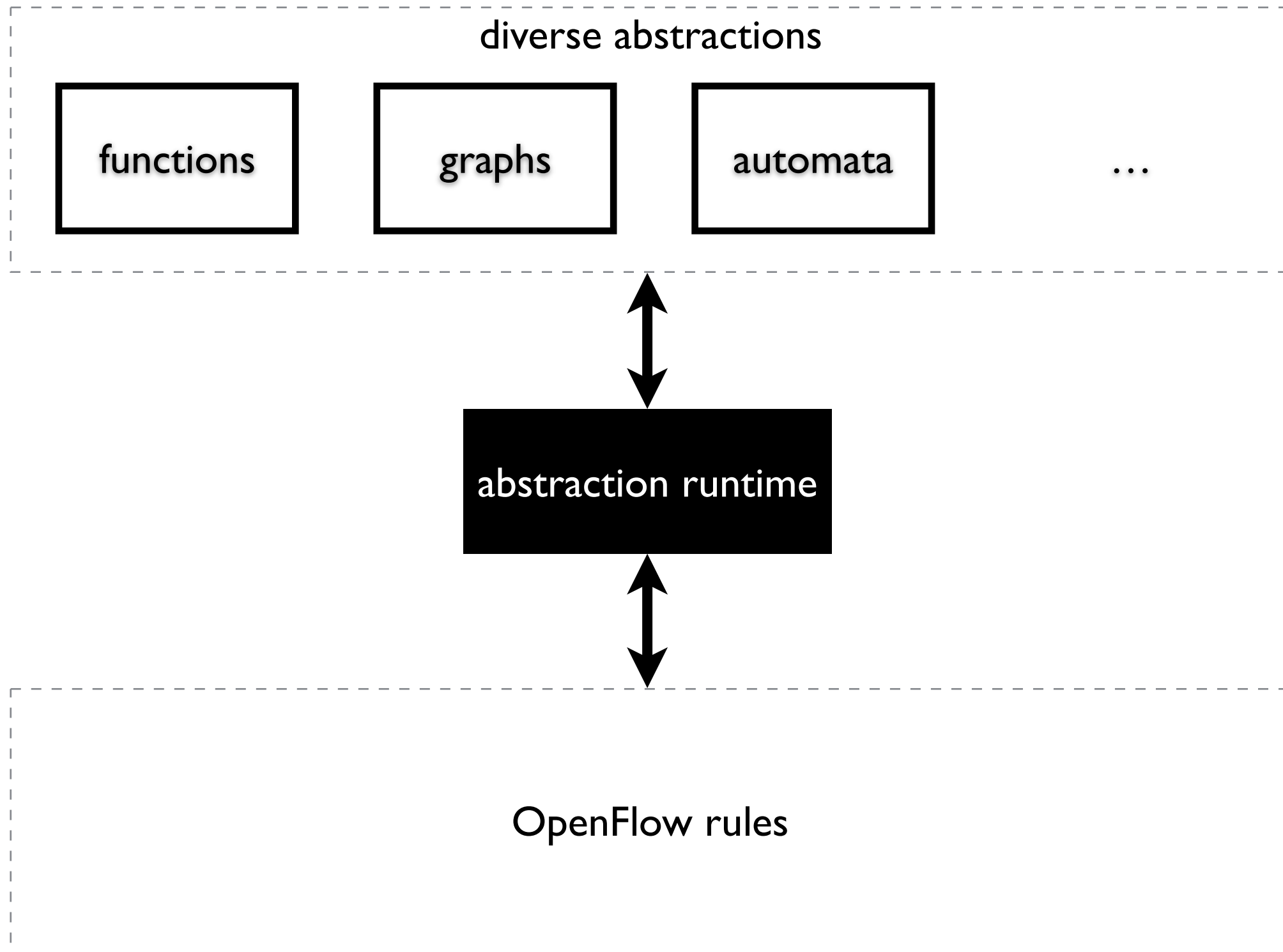
abstractions



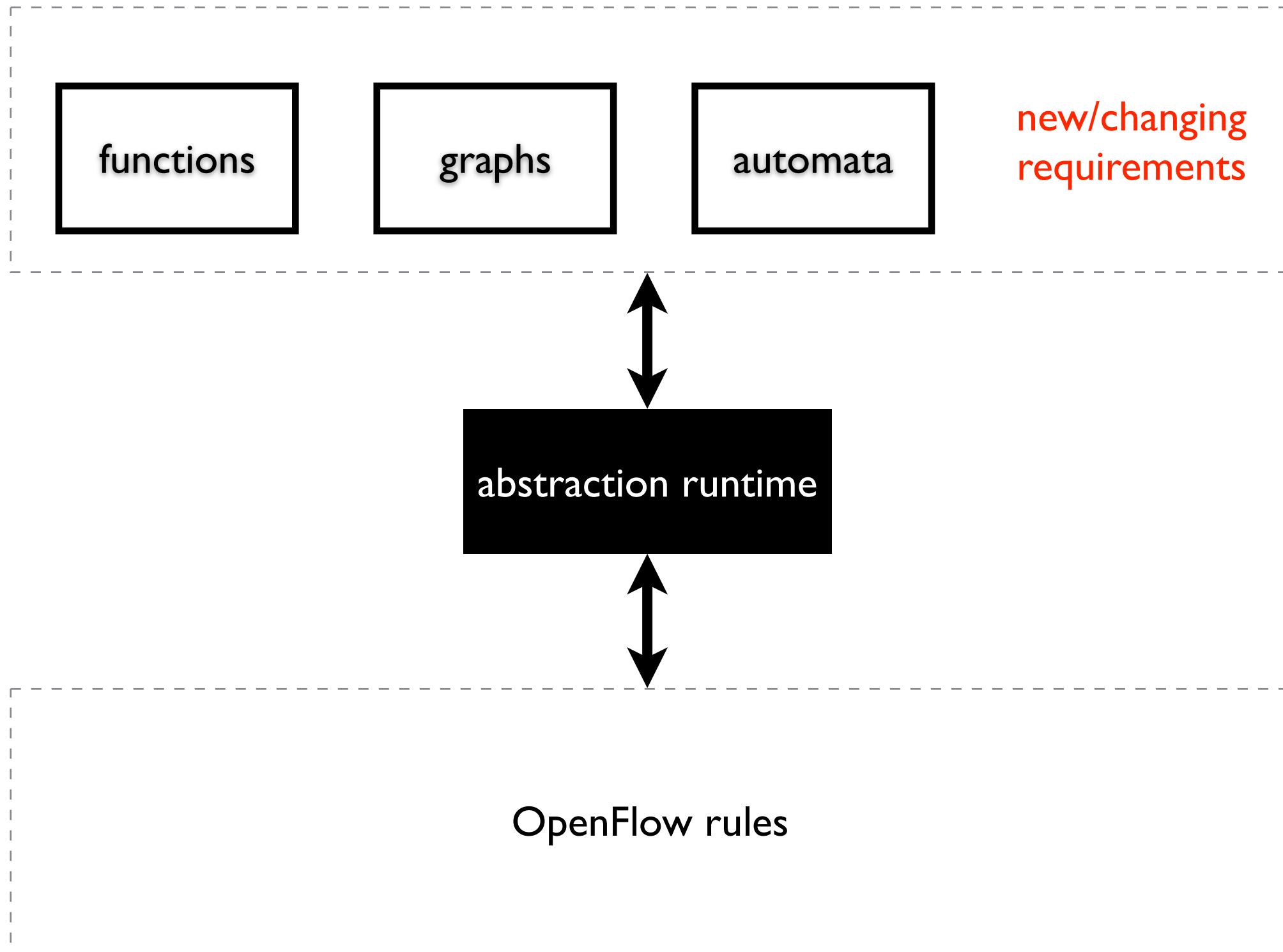
abstractions



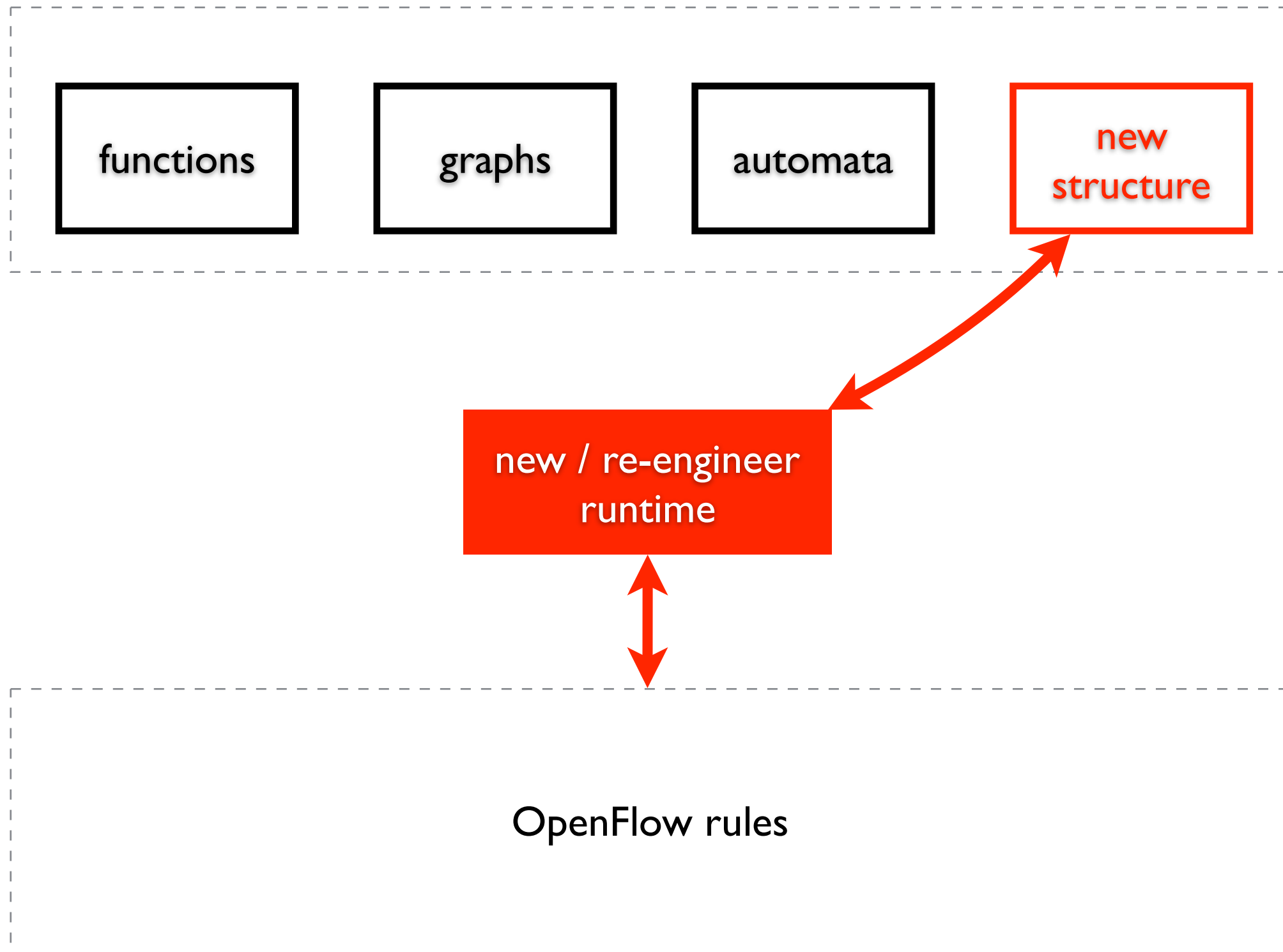
abstractions



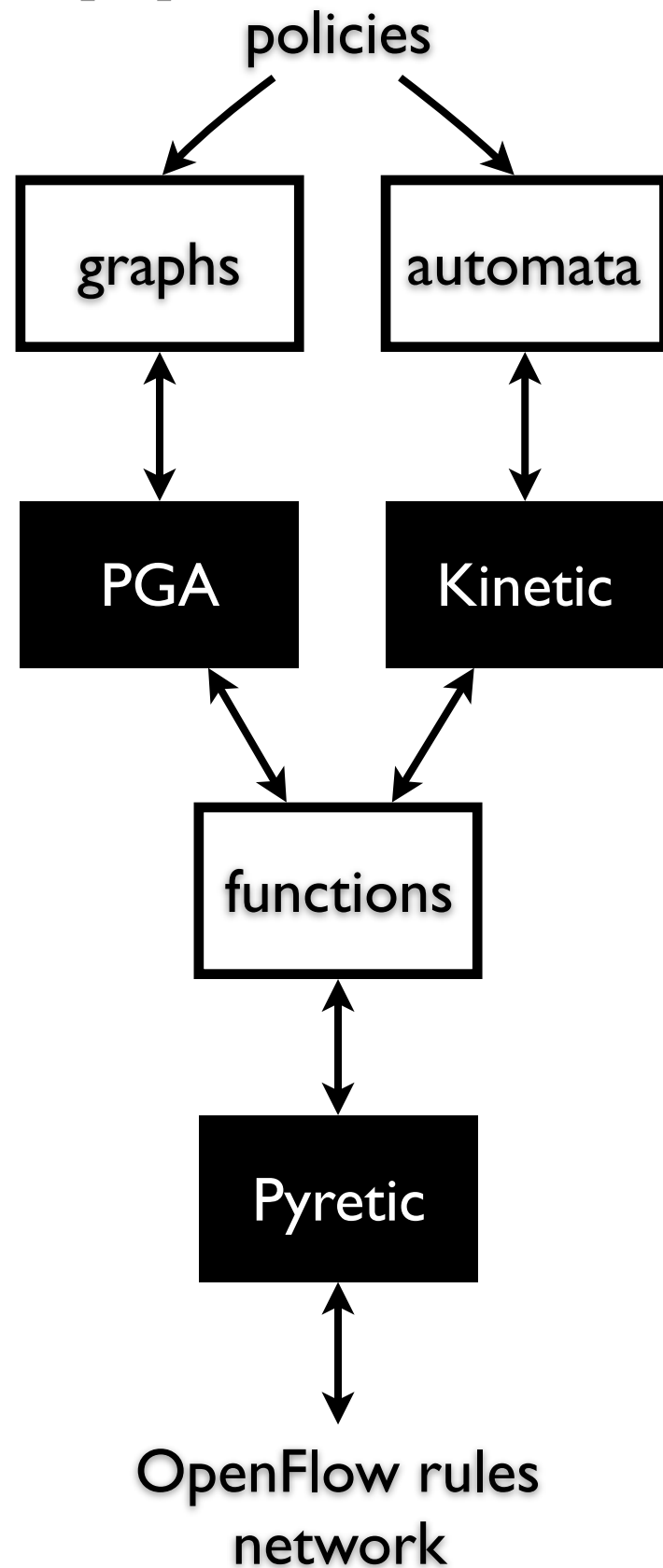
but network keeps evolving



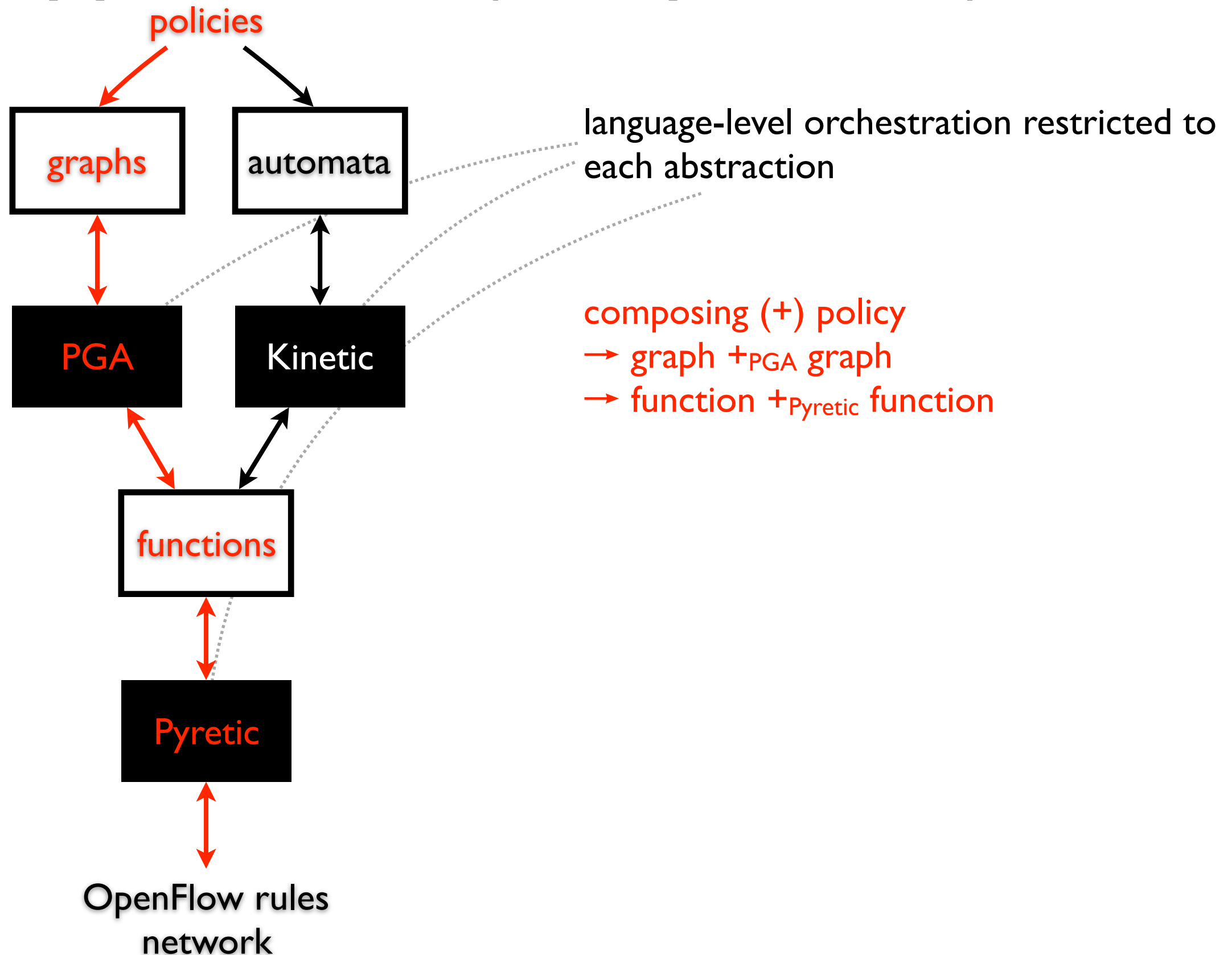
but network keeps evolving



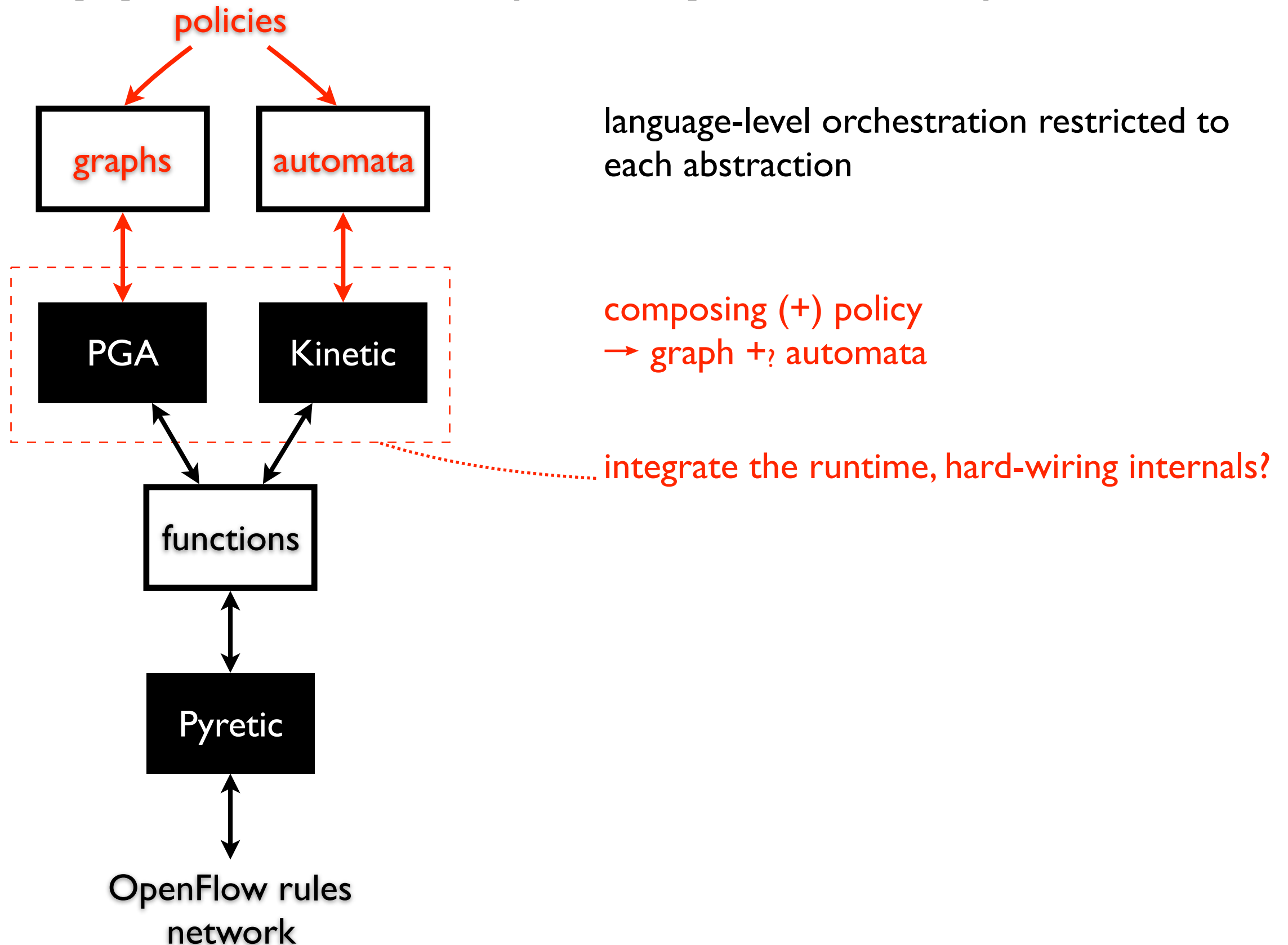
and applications (components) interact



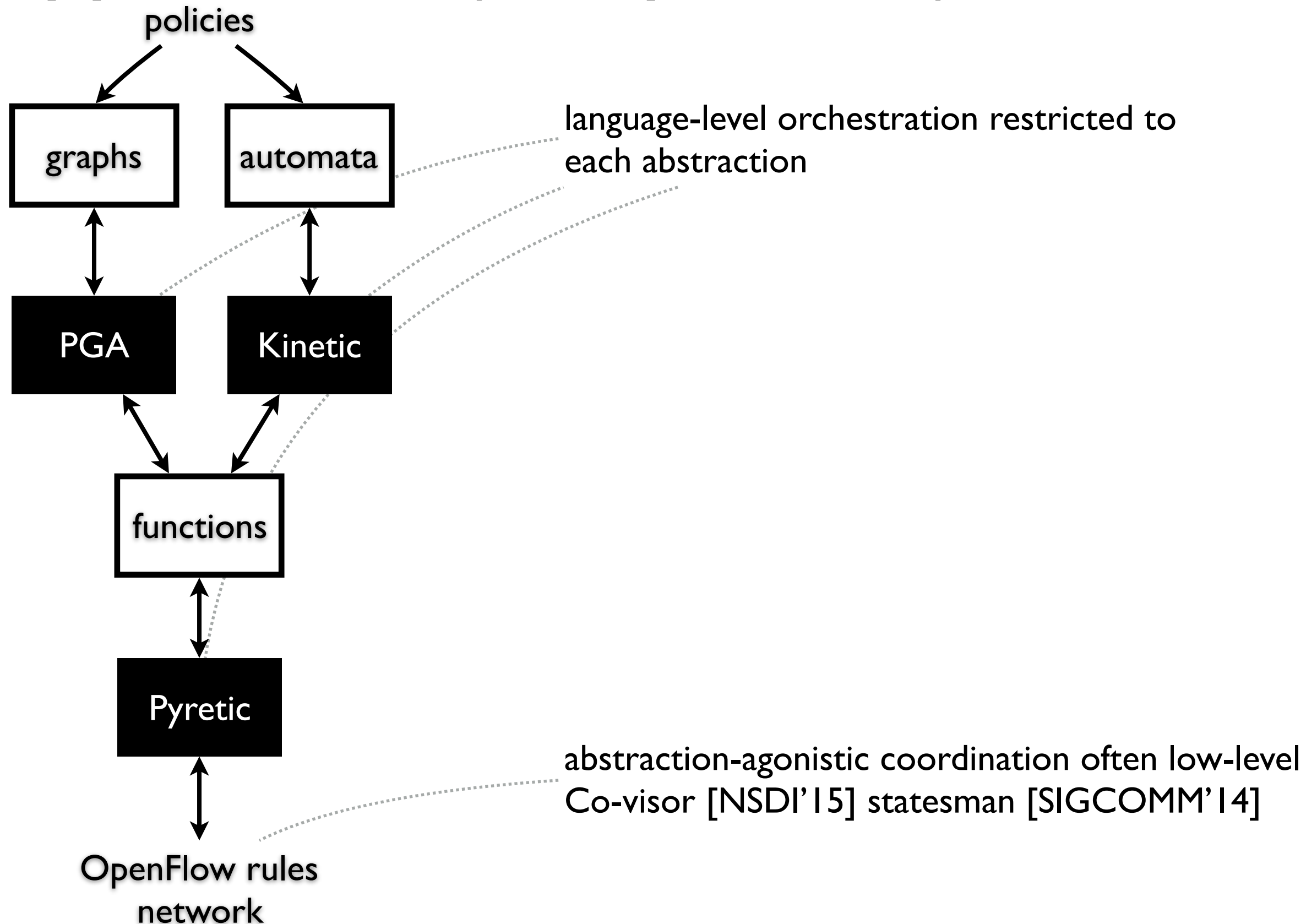
and applications (components) interact



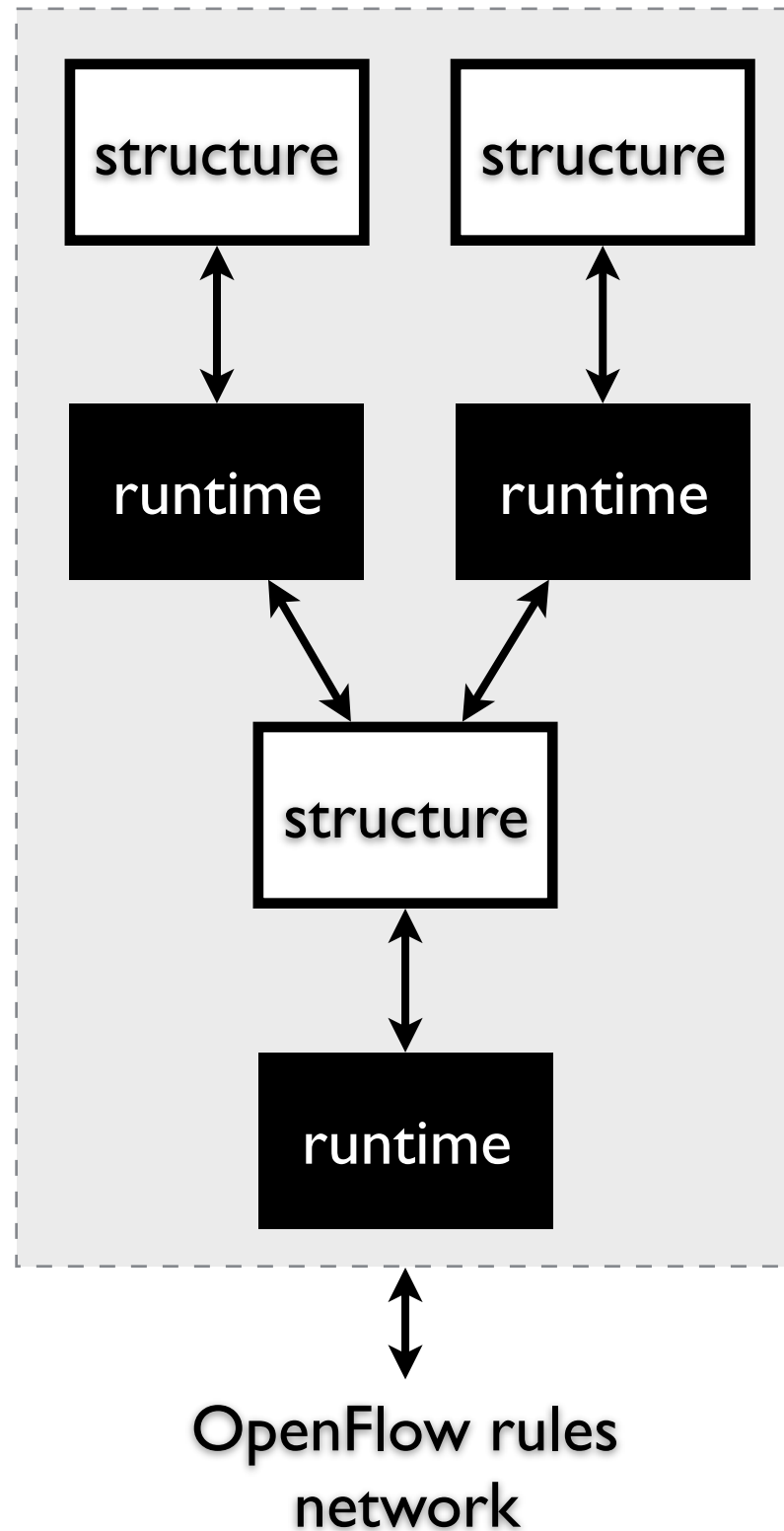
and applications (components) interact



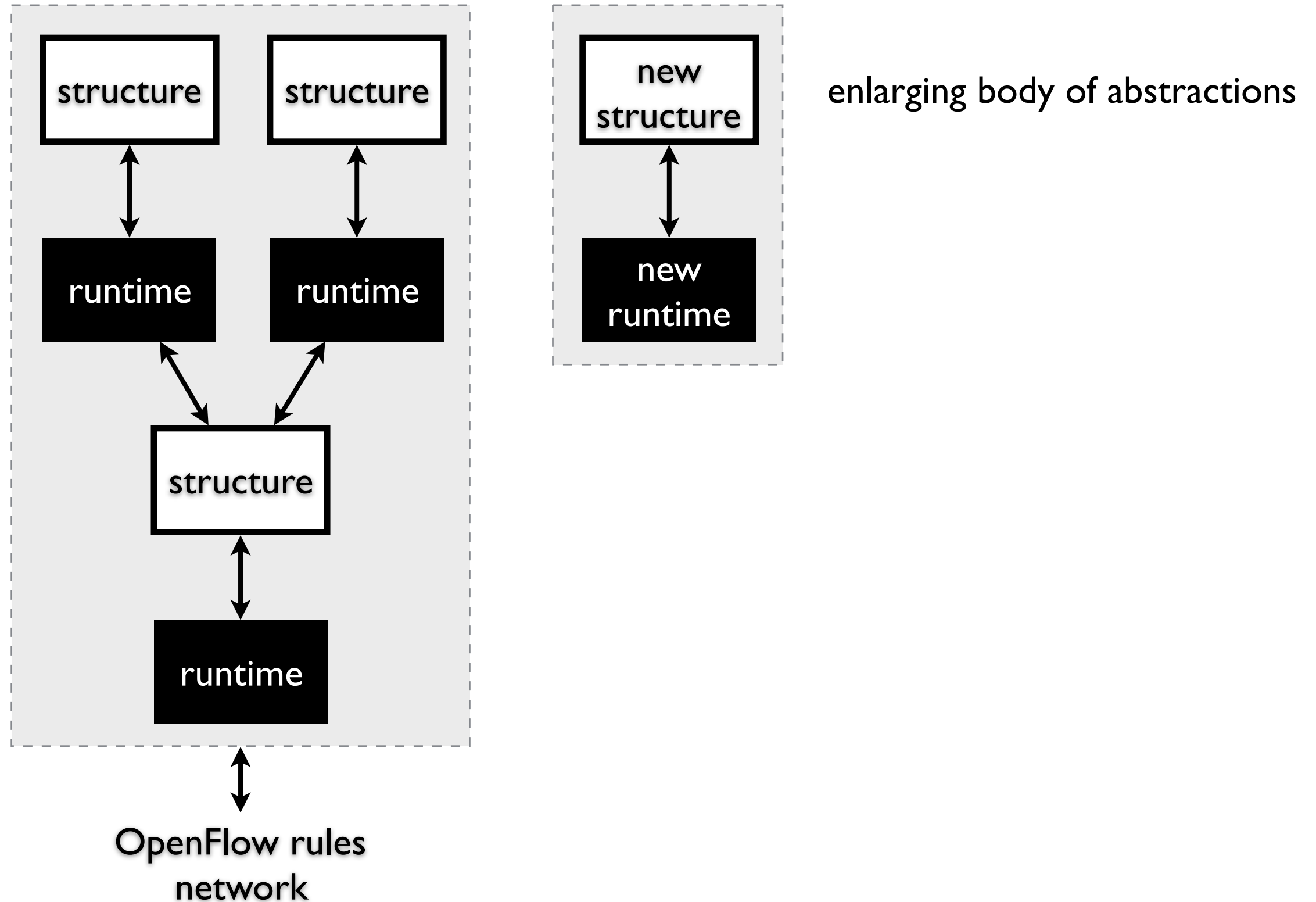
and applications (components) interact



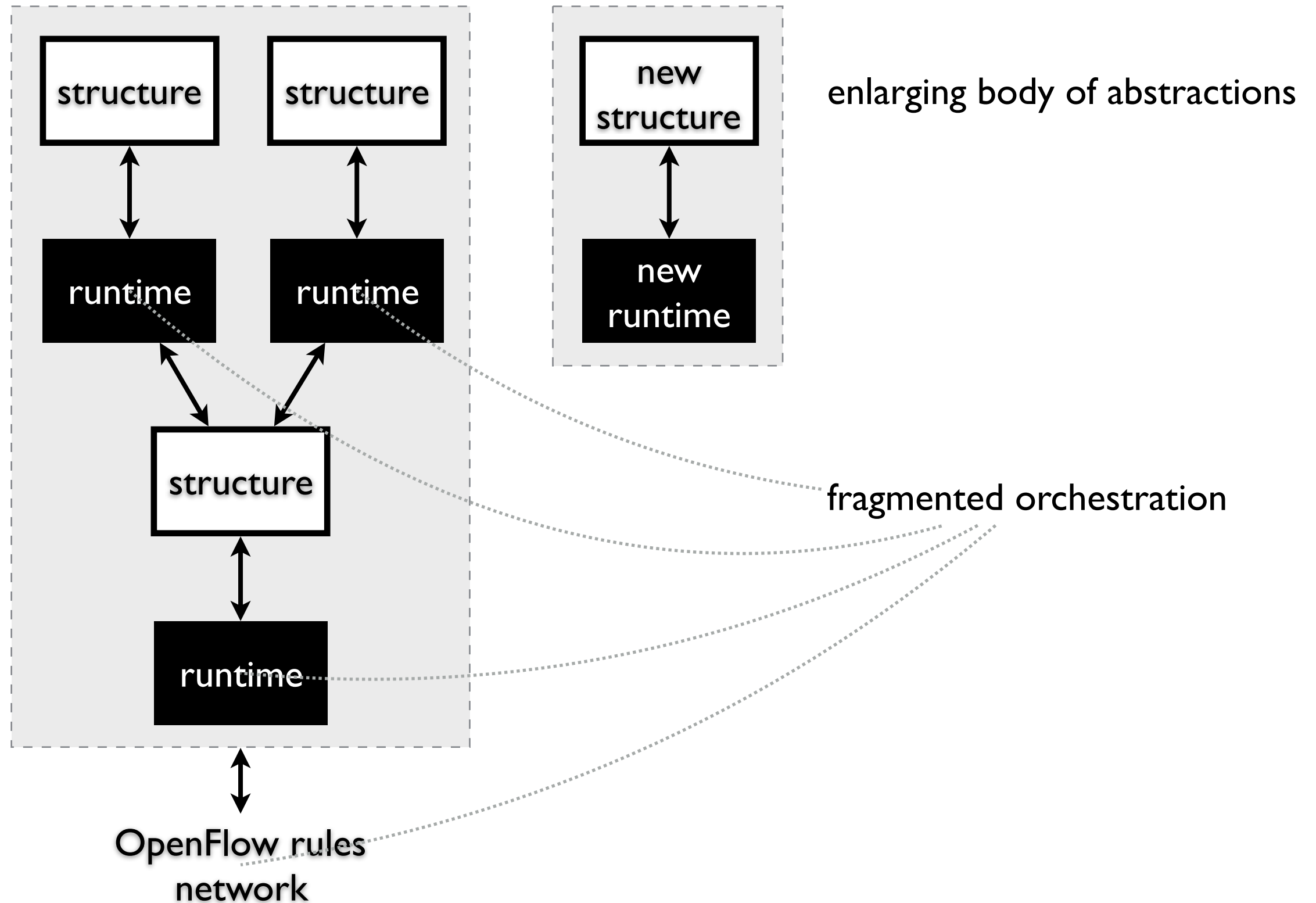
current states of abstraction



current states of abstraction



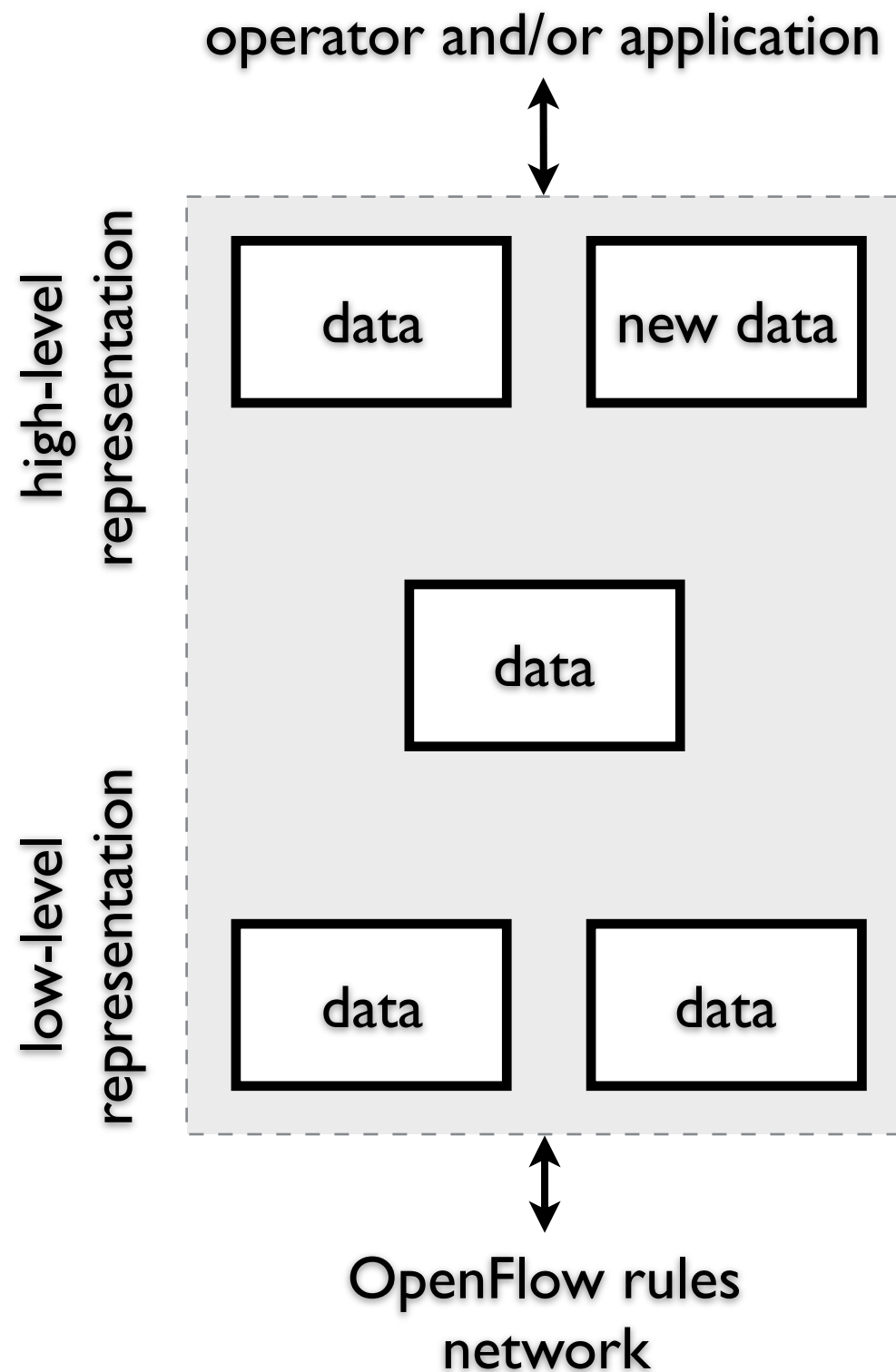
current states of abstraction



our perspective

SDN control revolves around data representation

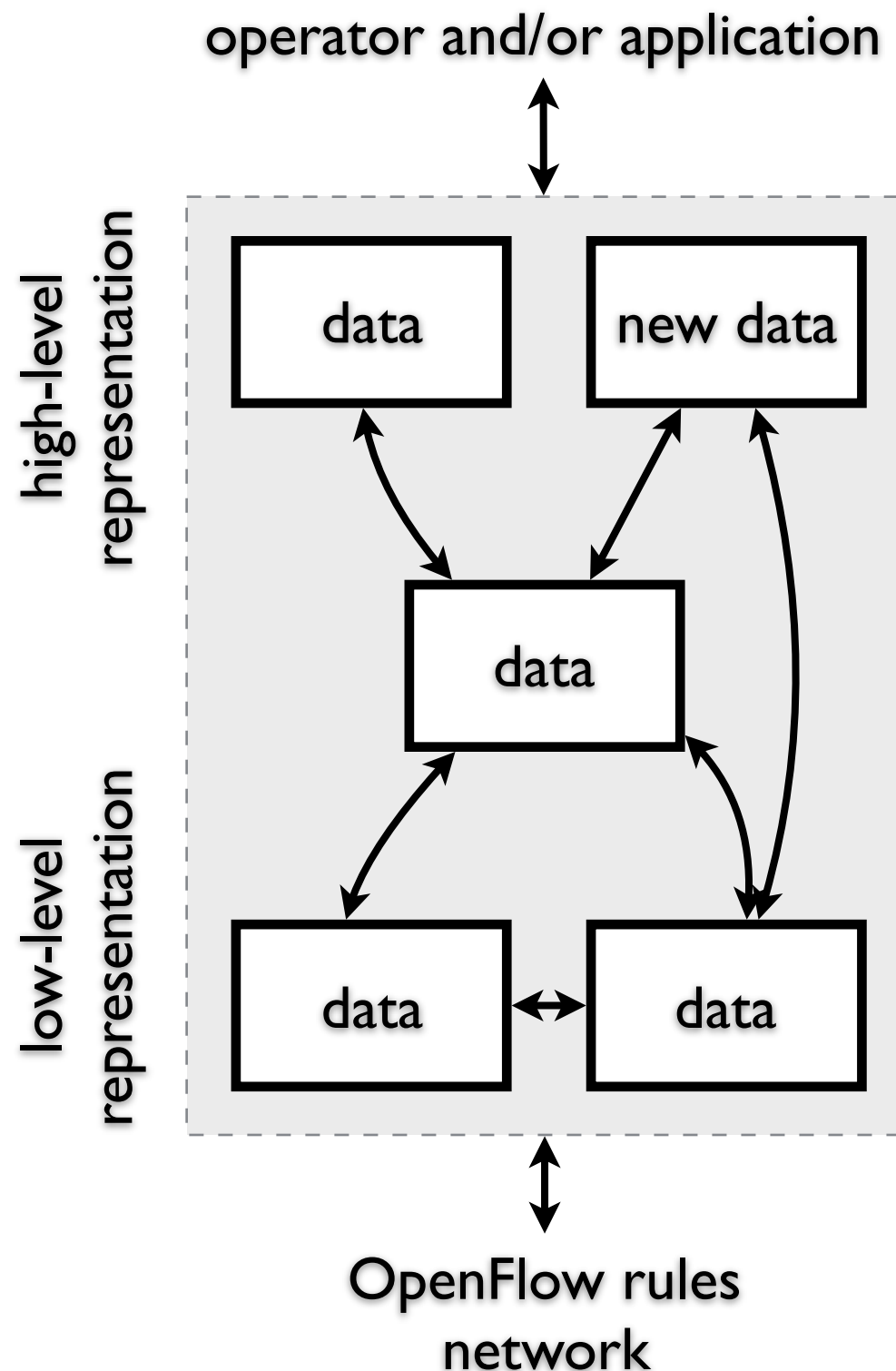
- discard specialized, pre-compiled, fixed structures
- adopt a *plain data representation*



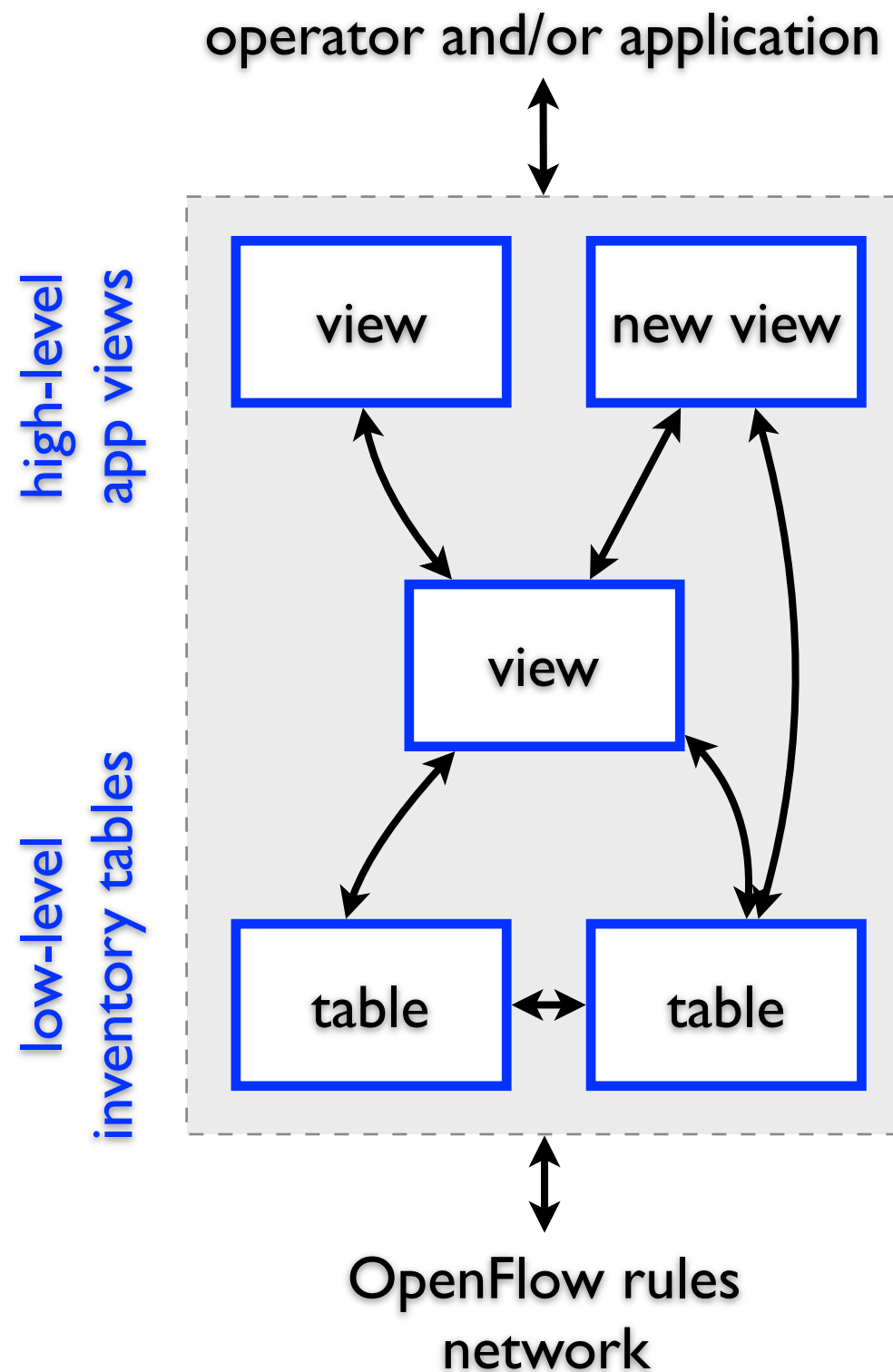
our perspective

SDN control revolves around data representation

- discard specialized, pre-compiled, fixed structures
- adopt a *plain data representation*
- use a *universal data language*

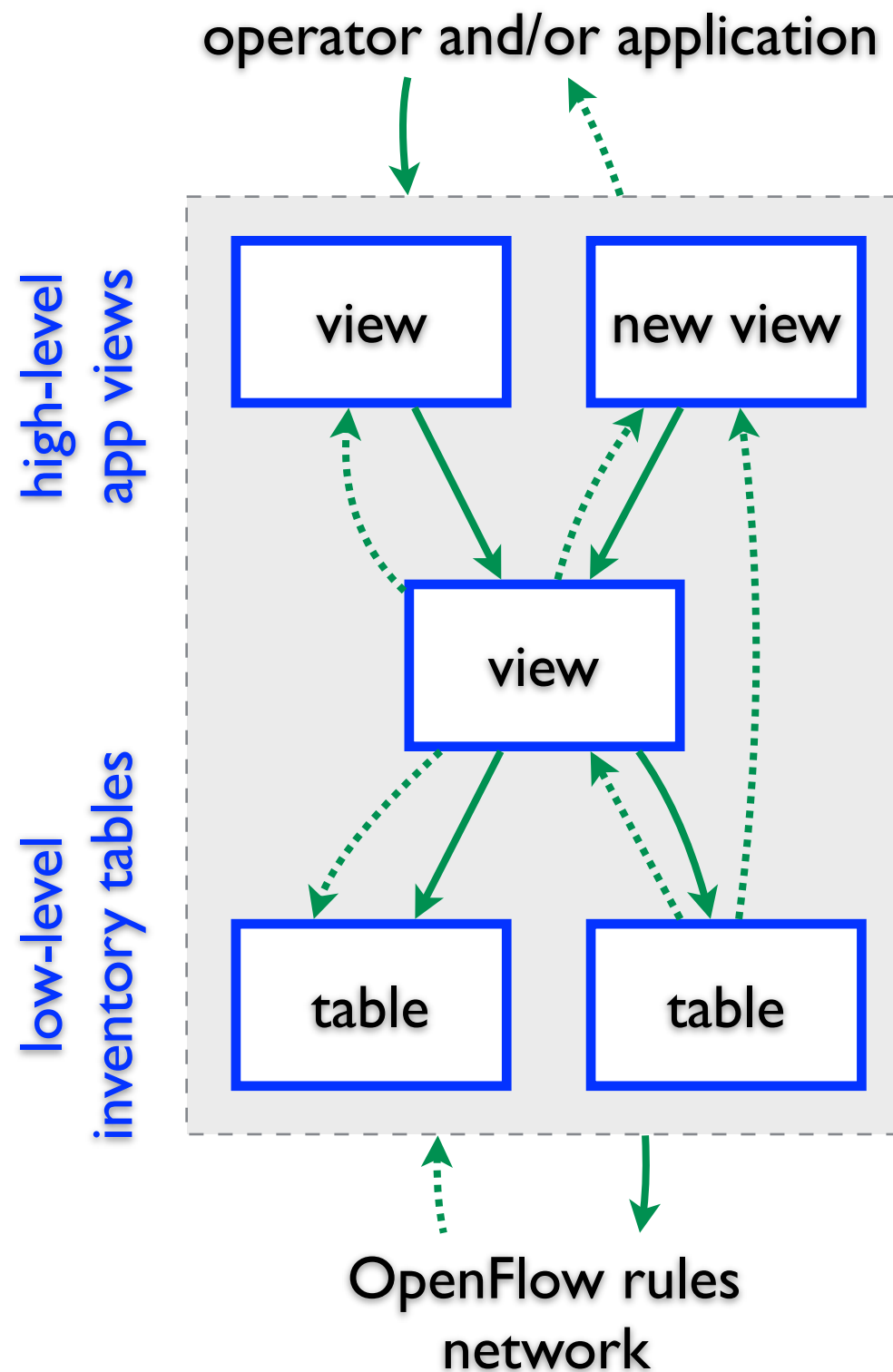


a database-defined network



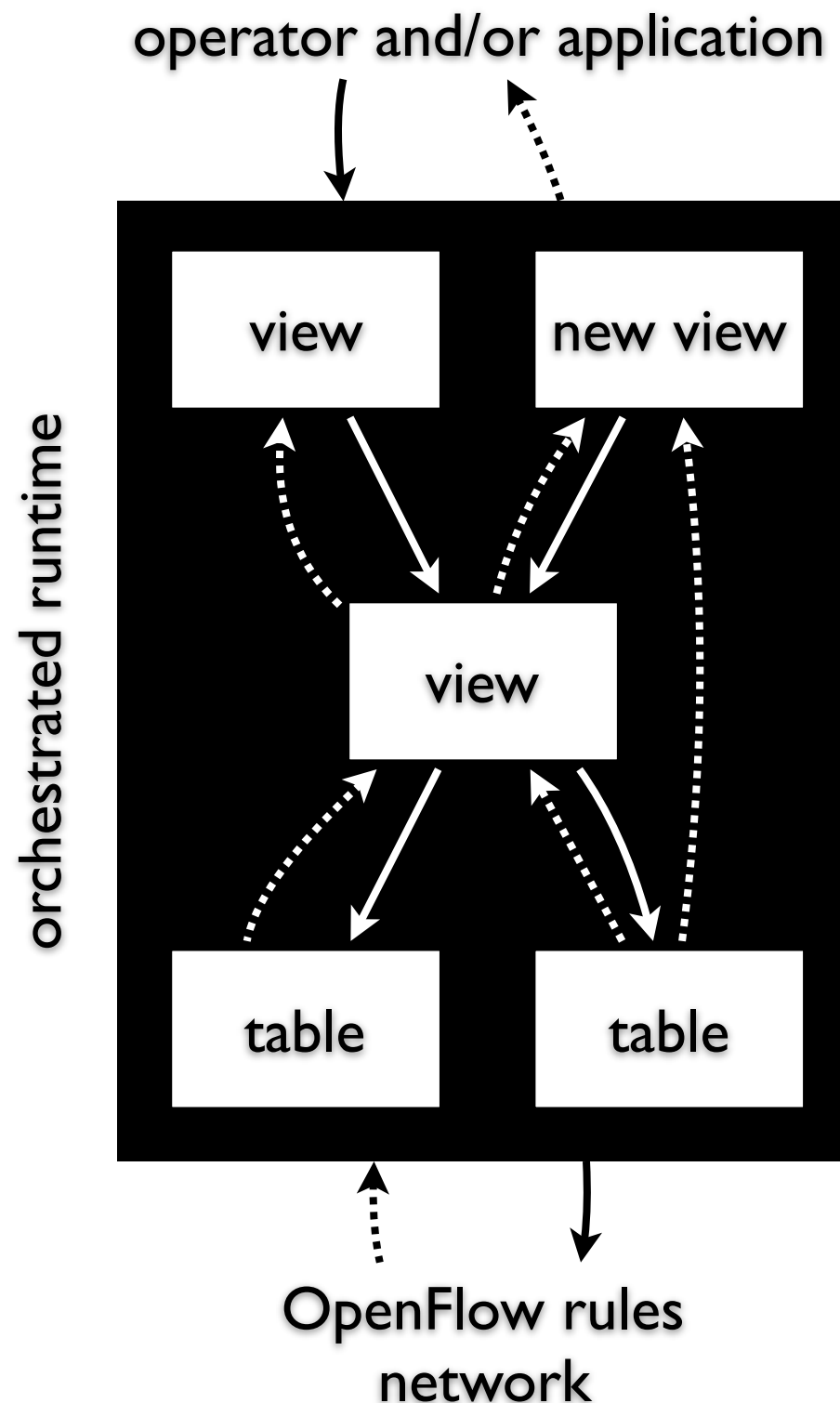
- **relation** — the plain data representation
- table — stored relation
- view — virtual relation

a database-defined network



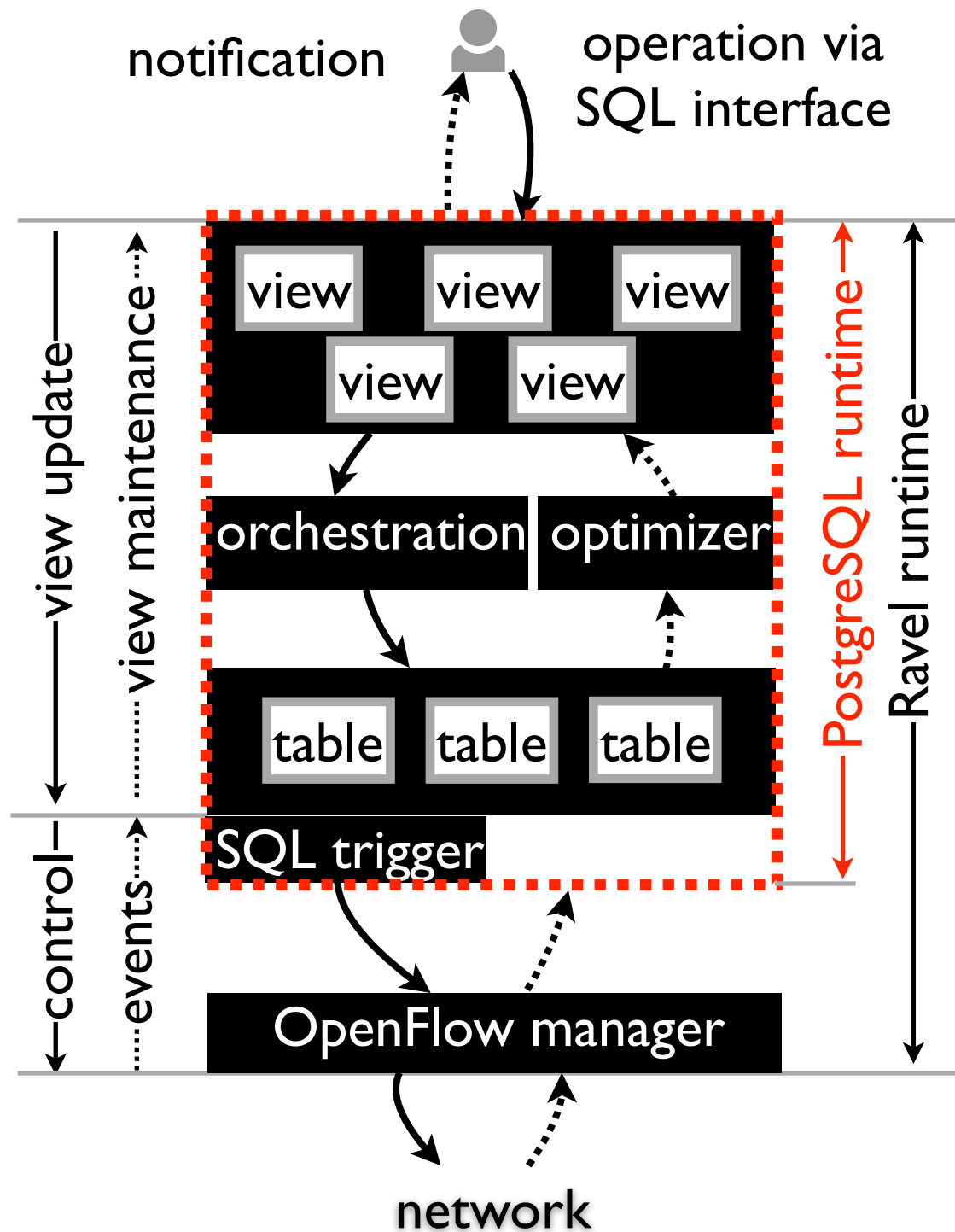
- ─ **relation** — the plain data representation
 - ─ table — stored relation
 - ─ view — virtual relation
- ─ **SQL** — the universal data language
 - ─ query, update, trigger, rule

a database-defined network



- relation — the plain data representation
- table — stored relation
- view — virtual relation
- SQL — the universal data language
- query, update, trigger, rule
- SQL database — the high-performance runtime
- orchestration challenge: refine runtime behavior by data mediation

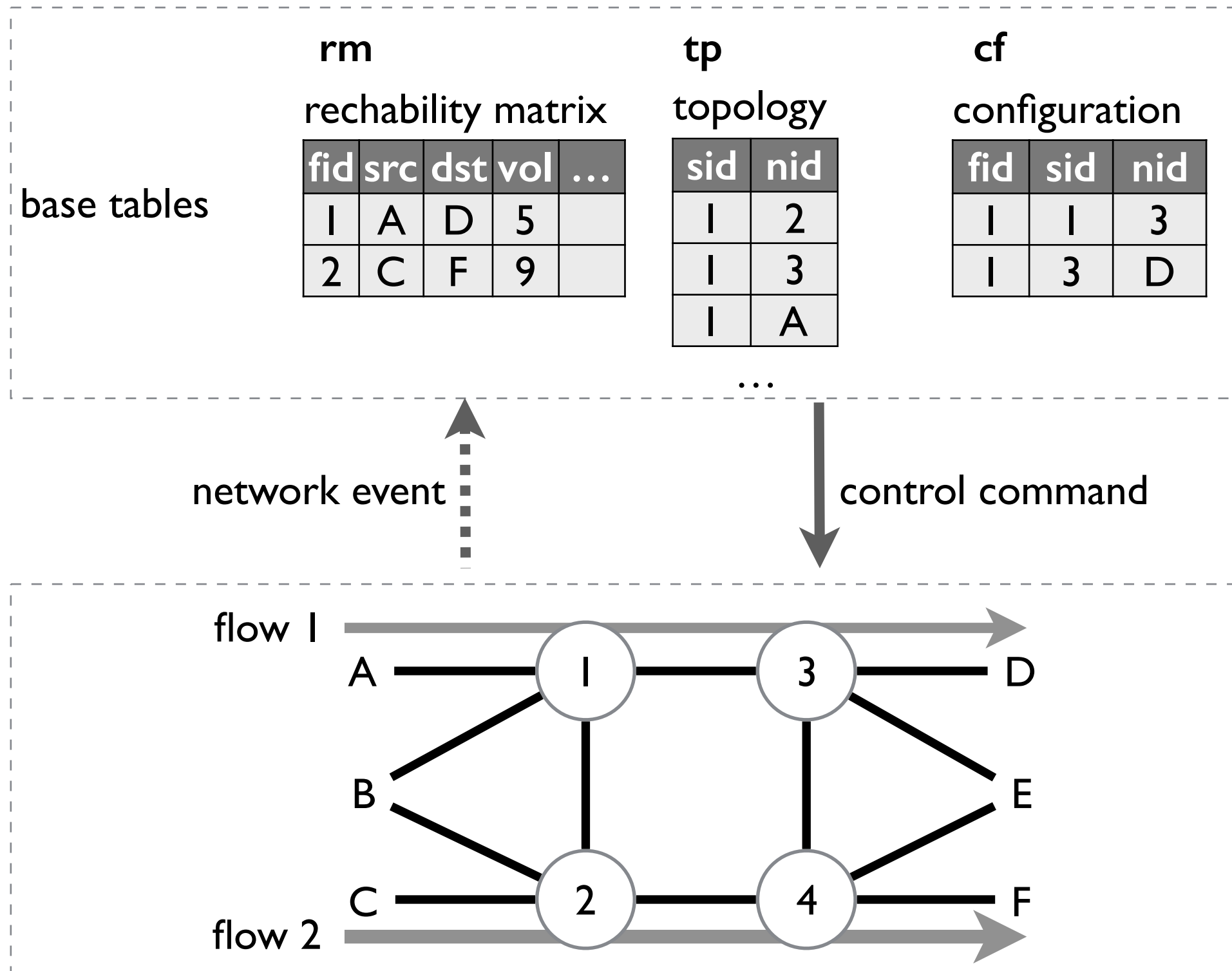
Ravel: a realization with SQL database



attractive features

- ad-hoc programmable abstraction via views
- orchestration across abstractions via view mechanism
- orchestration across applications via data mediation
- network control via SQL

abstraction: network tables



abstraction: application view

firewall view

```
CREATE VIEW acl_violation AS (  
  SELECT fid  
  FROM rm  
  WHERE FW = 1 AND  
    (src, dst) NOT IN  
    (SELECT end1, end2 FROM acl  
      WHERE allow = 1)  
);
```

```
CREATE TABLE acl (  
  end1 integer, end2 integer, allow integer  
);
```

abstraction: application view

firewall view

```
CREATE VIEW acl_violation AS (  
  SELECT fid  
  FROM rm  
  WHERE FW = 1 AND  
    (src, dst) NOT IN  
    (SELECT end1, end2 FROM acl  
      WHERE allow = 1)  
);
```

```
CREATE TABLE acl (  
  end1 integer, end2 integer, allow integer  
);
```

control loop: monitoring firewall view and repairing violation

```
CREATE RULE acl_repair AS  
  ON DELETE TO acl_violation  
  DO INSTEAD  
    DELETE FROM rm WHERE fid = OLD.fid;
```

abstraction: application view

firewall view

```
CREATE VIEW acl_violation AS (  
  SELECT fid  
  FROM rm  
  WHERE FW = 1 AND  
    (src, dst) NOT IN  
    (SELECT end1, end2 FROM acl  
      WHERE allow = 1)  
);
```

```
CREATE TABLE acl (  
  end1 integer, end2 integer, allow integer  
);
```

control loop: monitoring firewall view and repairing violation

```
CREATE RULE acl_repair AS  
  ON DELETE TO acl_violation  
  DO INSTEAD  
    DELETE FROM rm WHERE fid = OLD.fid;
```

- many more
 - routing, stateful firewall, service chain policy between subdomains ...

abstraction: application view

firewall view

```
CREATE VIEW acl_violation AS (  
  SELECT fid  
  FROM rm  
  WHERE FW = 1 AND  
    (src, dst) NOT IN  
    (SELECT end1, end2 FROM acl  
      WHERE allow = 1)  
);
```

```
CREATE TABLE acl (  
  end1 integer, end2 integer, allow integer  
);
```

control loop: monitoring firewall view and repairing violation

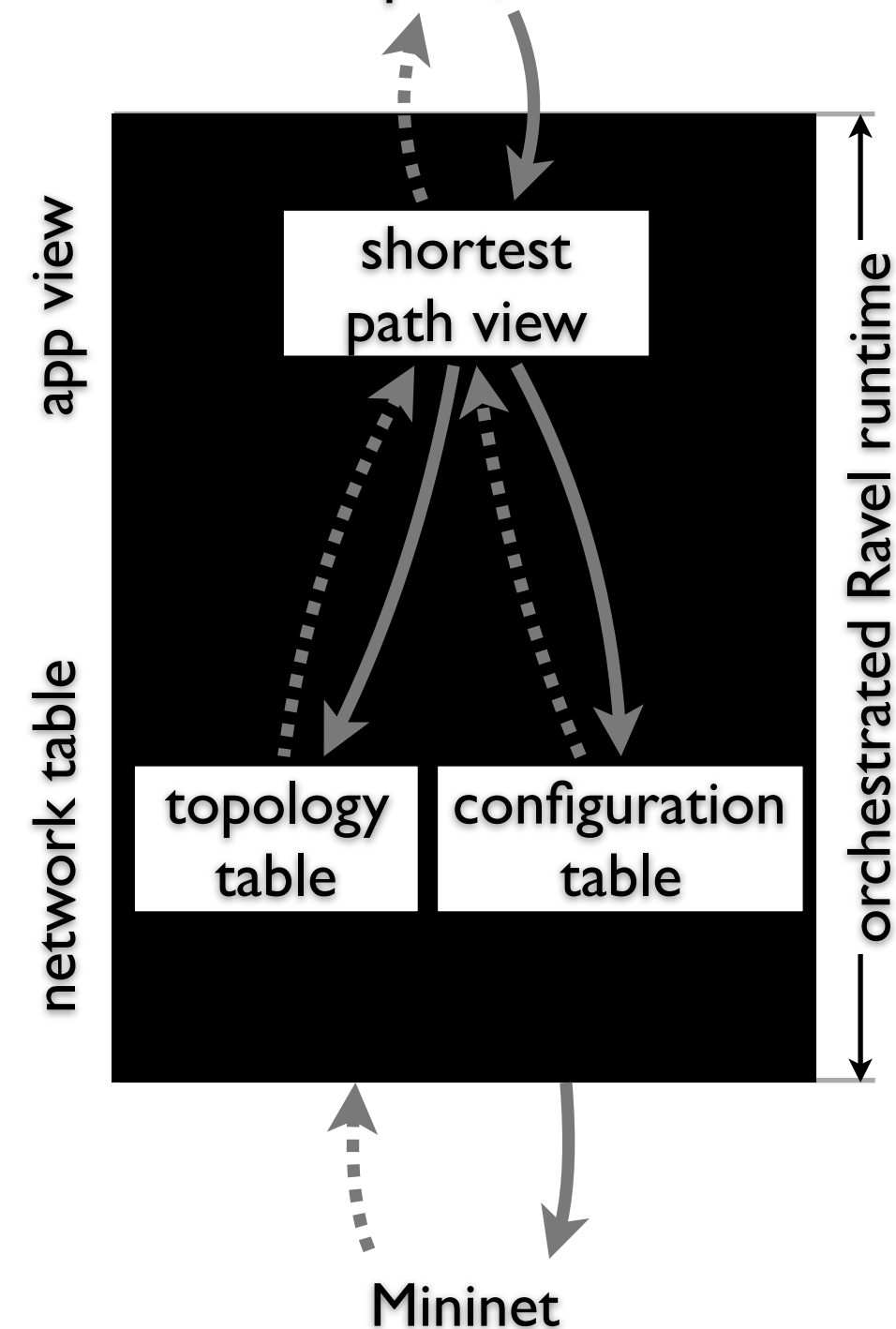
```
CREATE RULE acl_repair AS  
  ON DELETE TO acl_violation  
  DO INSTEAD  
    DELETE FROM rm WHERE fid = OLD.fid;
```

- many more
 - routing, stateful firewall, service chain policy between subdomains ...
- optimizing application by materializing views
 - (one order of magnitude) faster access with small maintenance overhead (.01~10ms)

orchestration across representations

routing app: check
broken path, re-route

SQL rule:
upon broken path, re-route

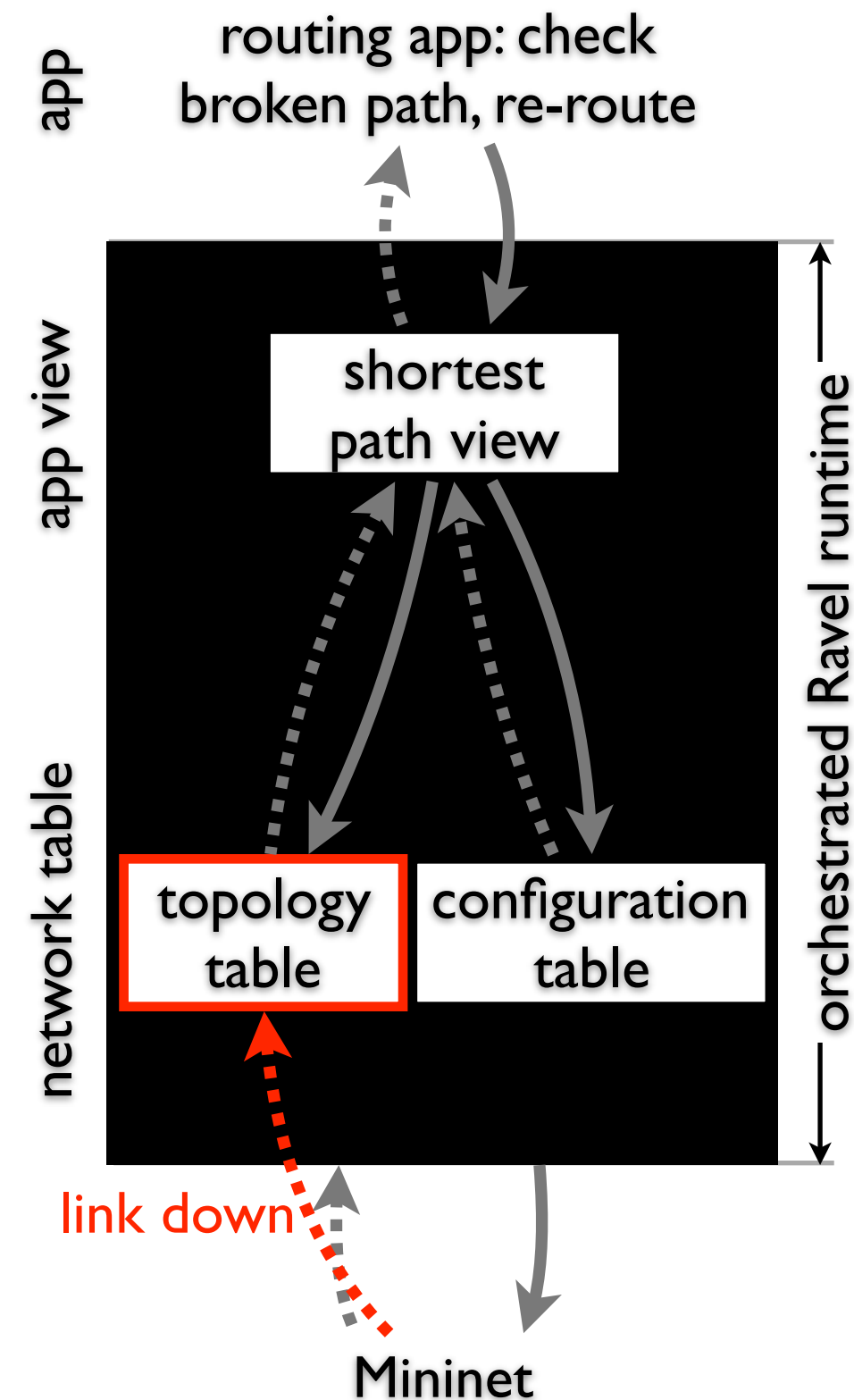


shortest path	

topology		

configuration		

orchestration across representations



SQL rule:
upon broken path, re-route

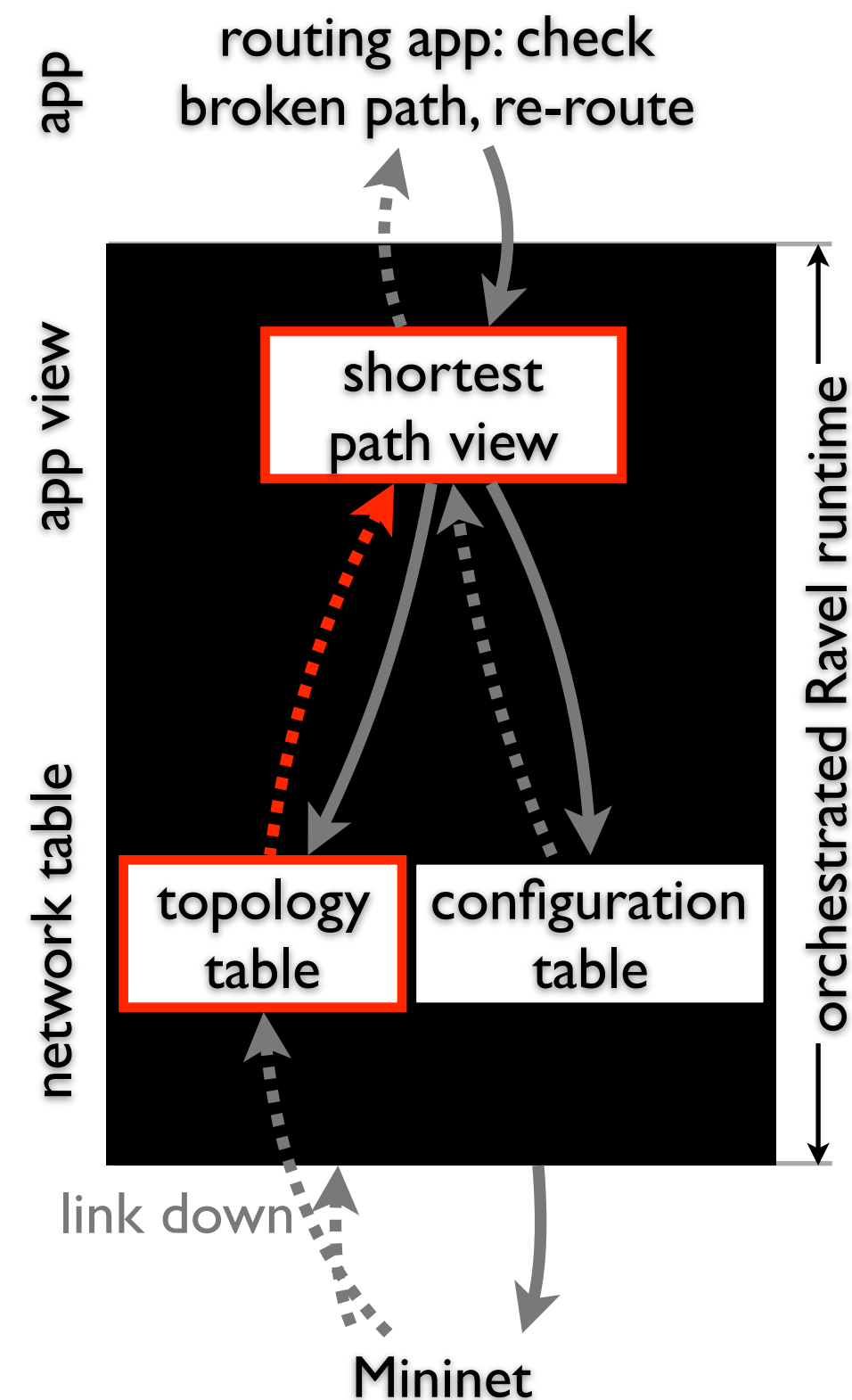
shortest path	

	topology		
	sid	nid	active
-	172	39	1
+	172	39	0

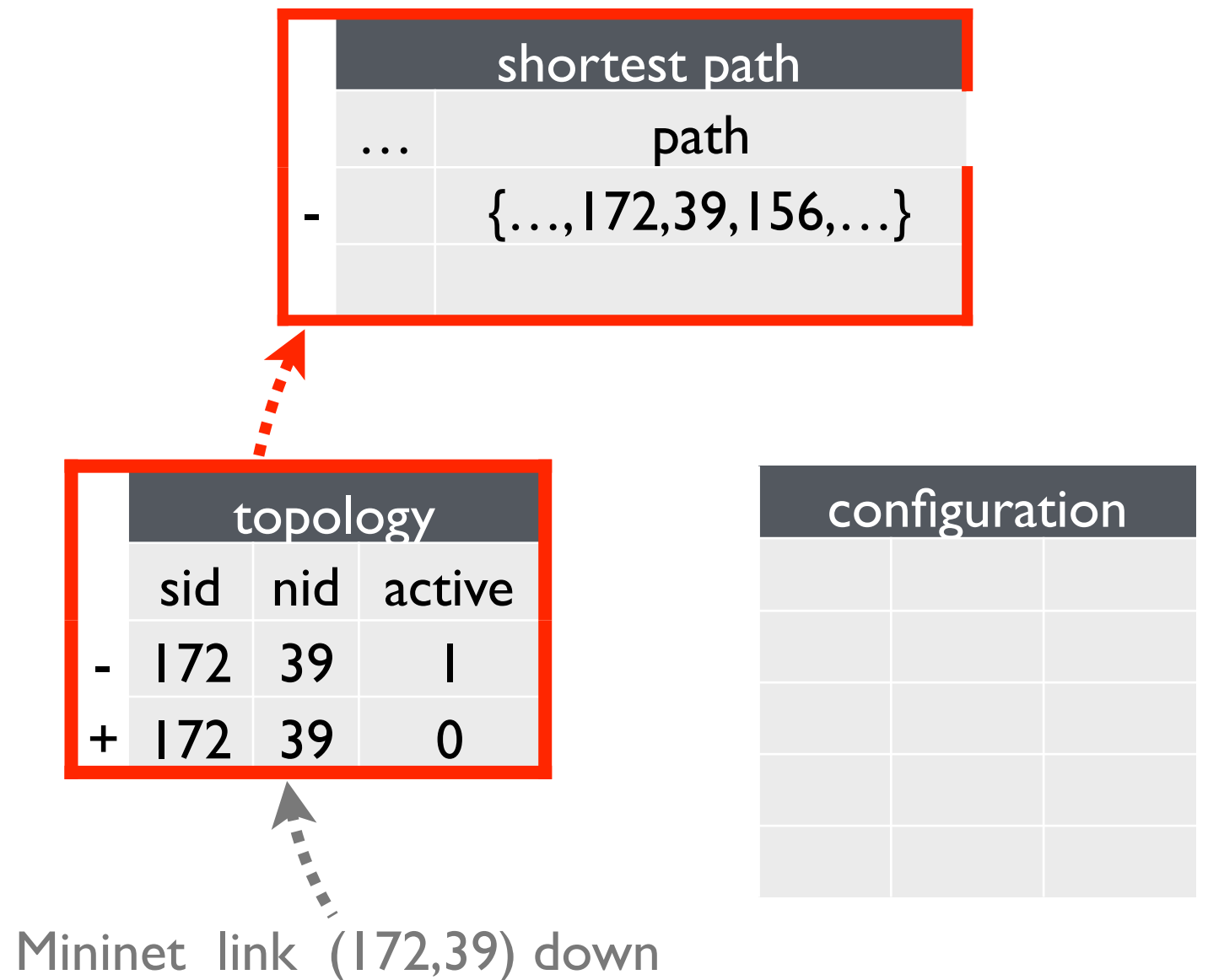
configuration		

Mininet link (172,39) down

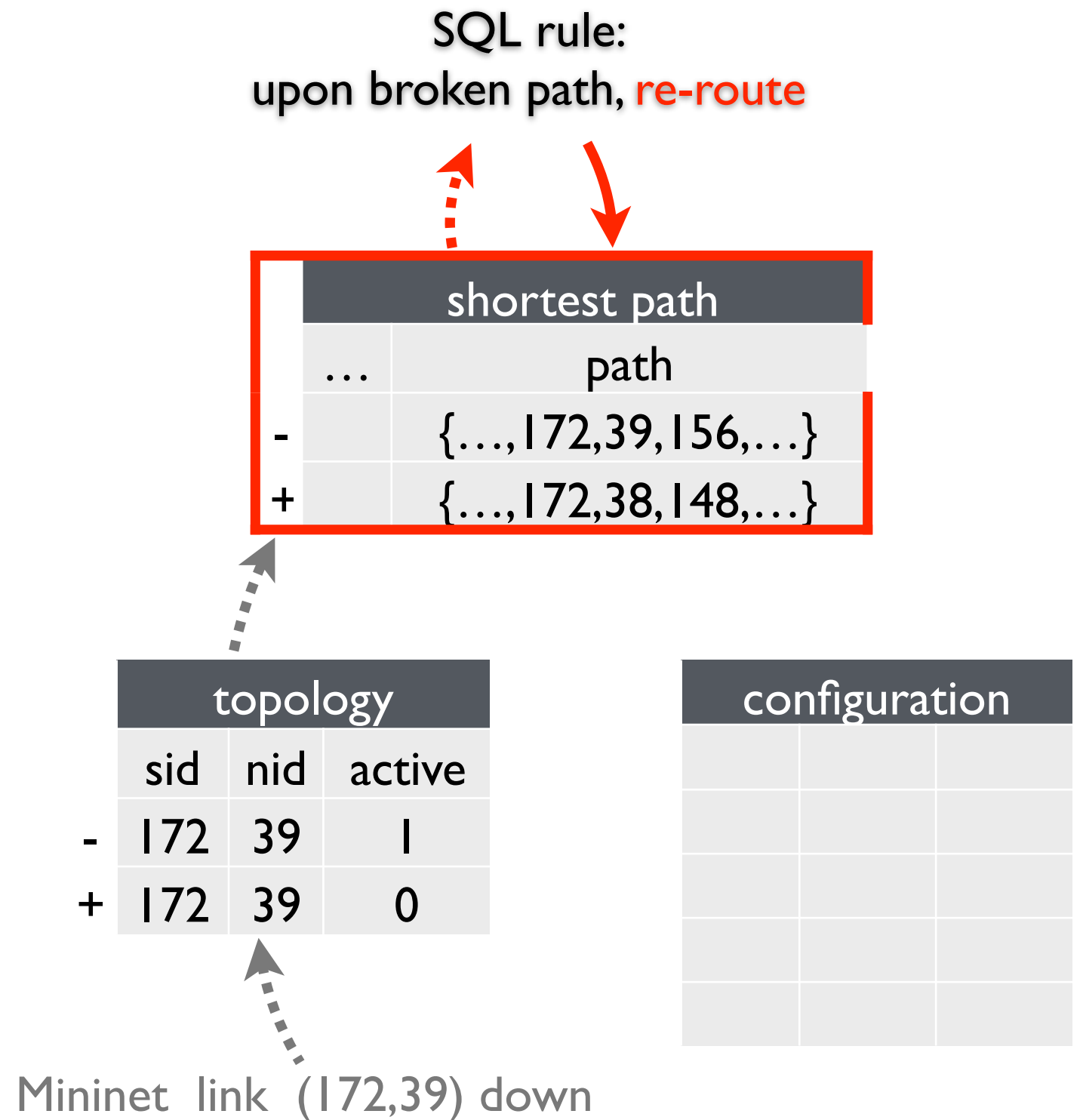
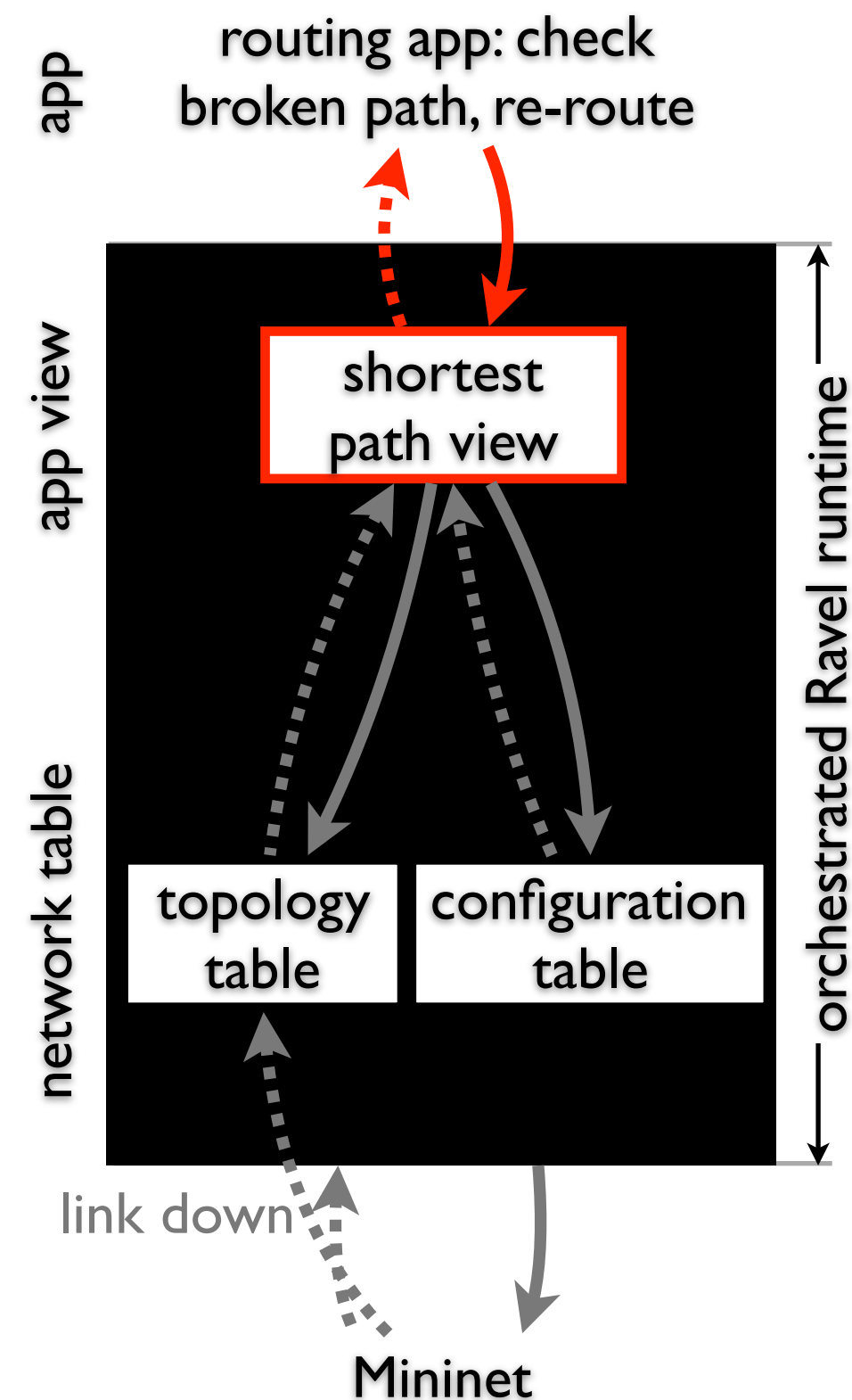
orchestration across representations



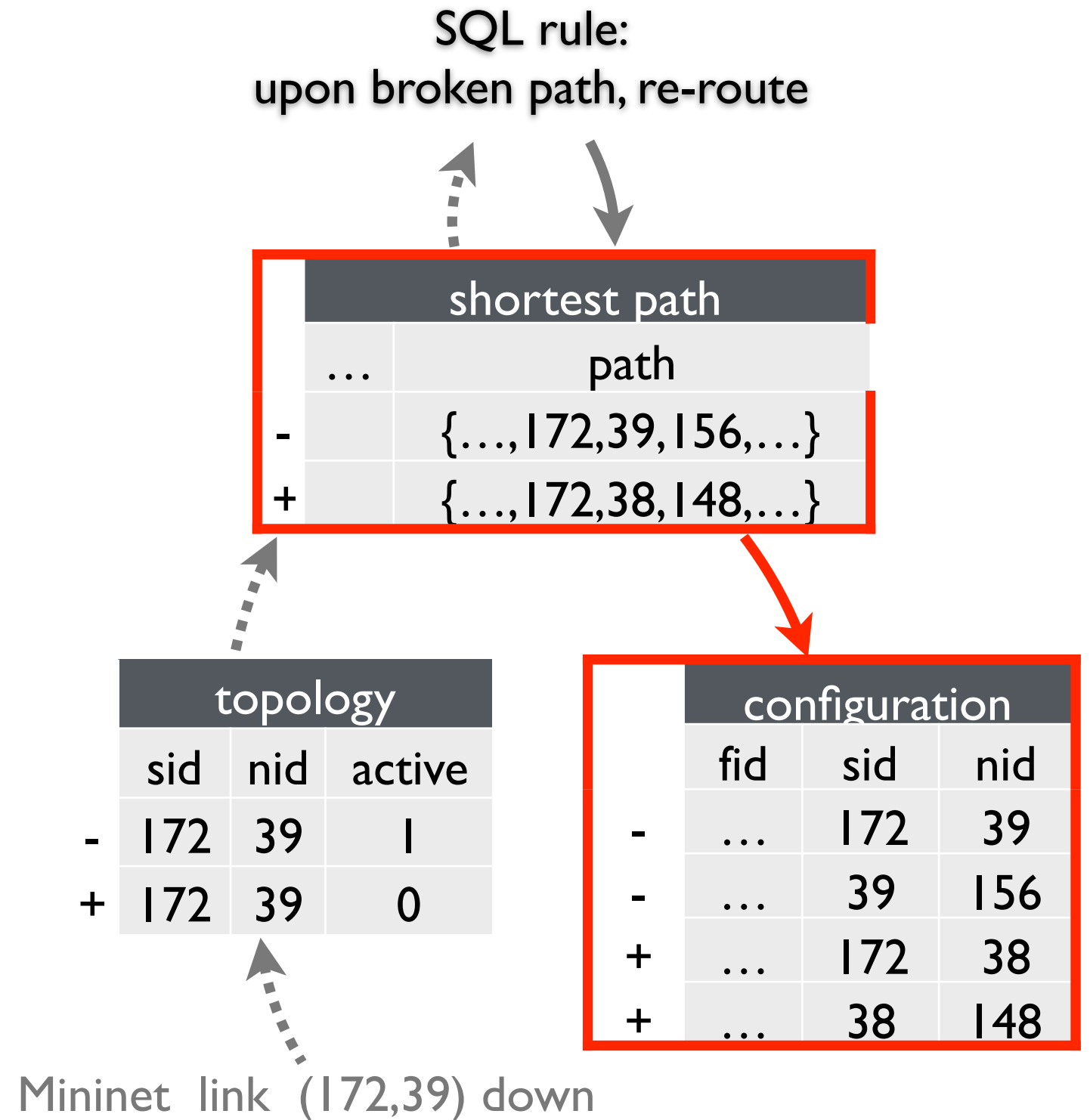
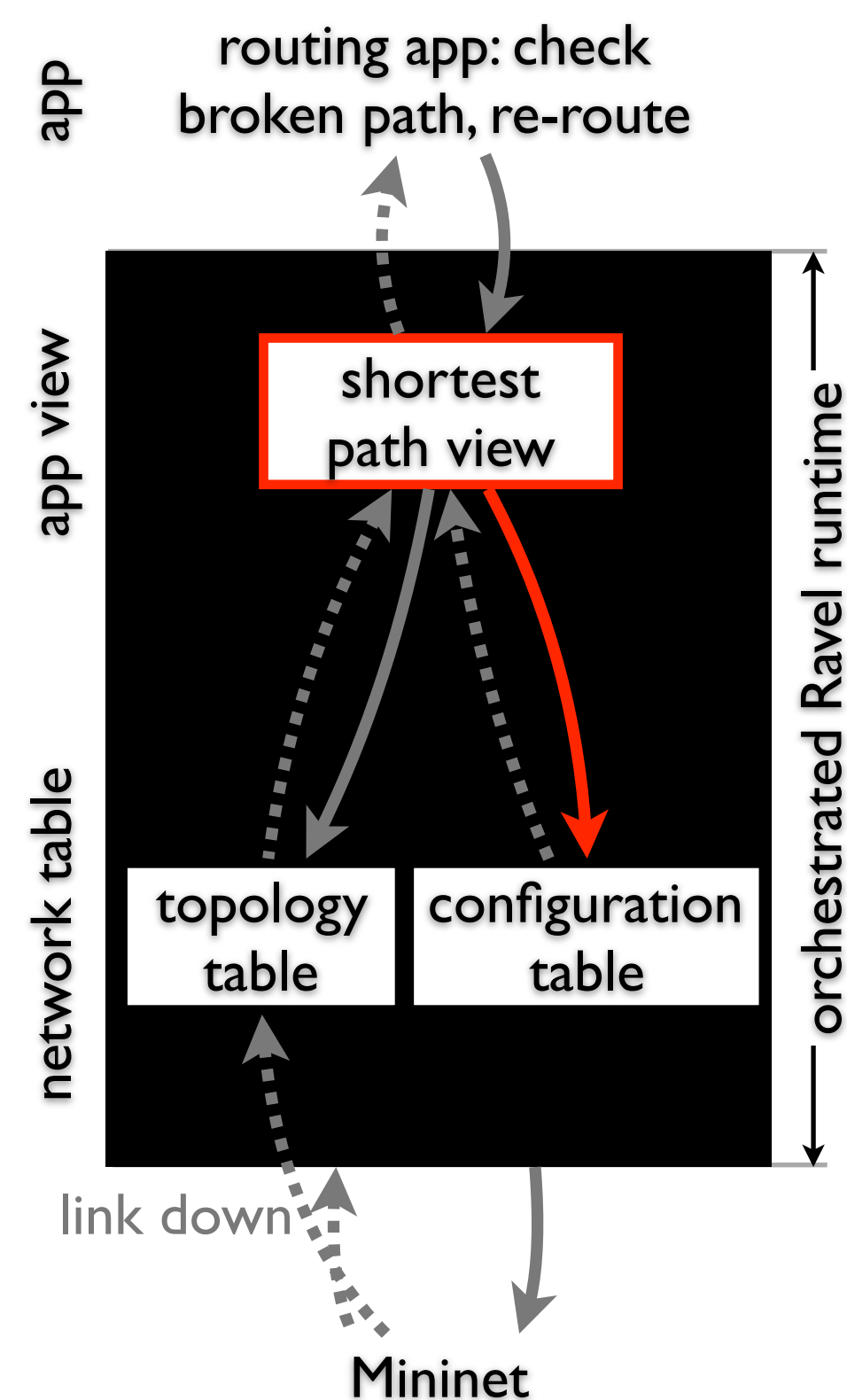
SQL rule:
upon broken path, re-route



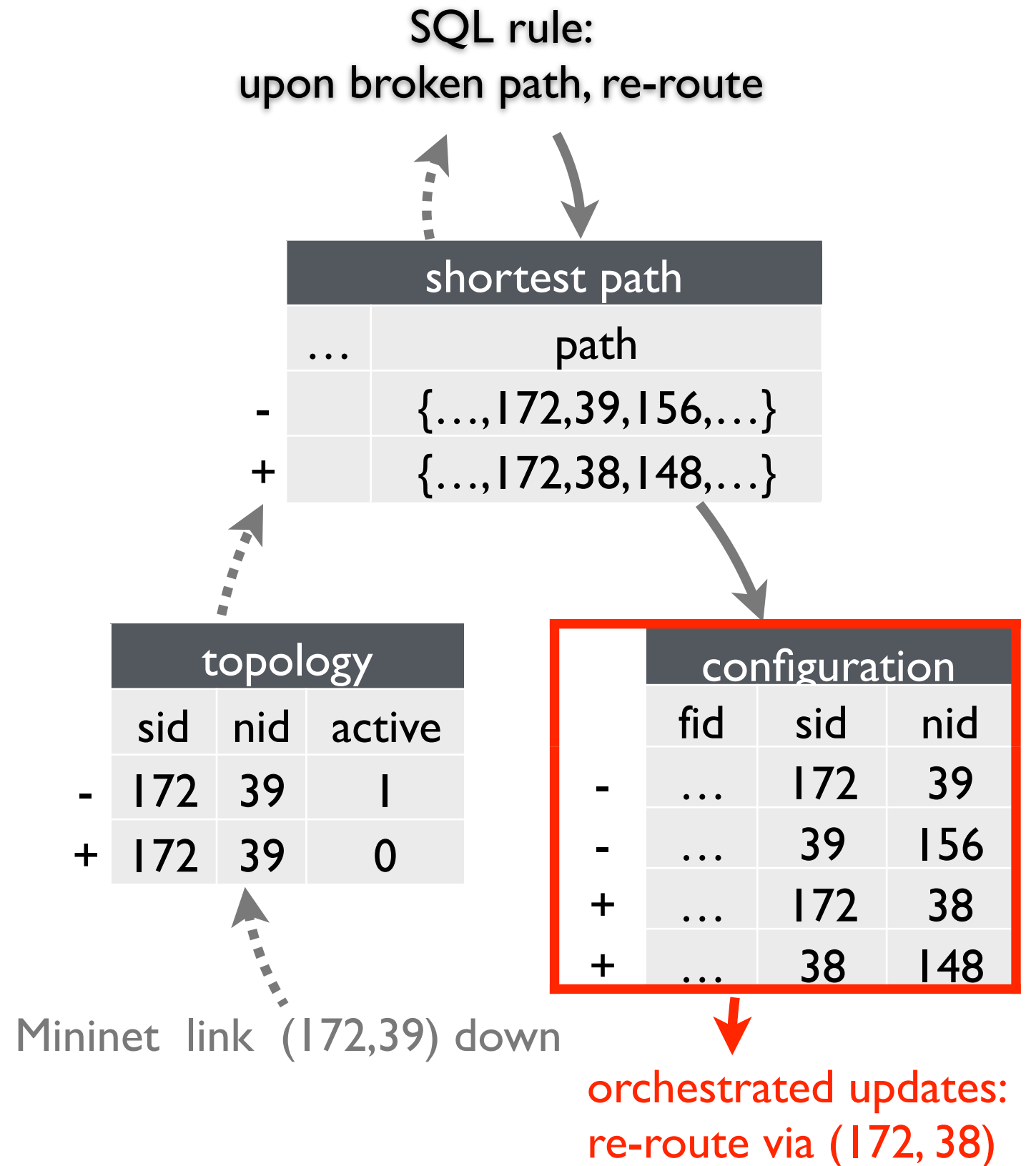
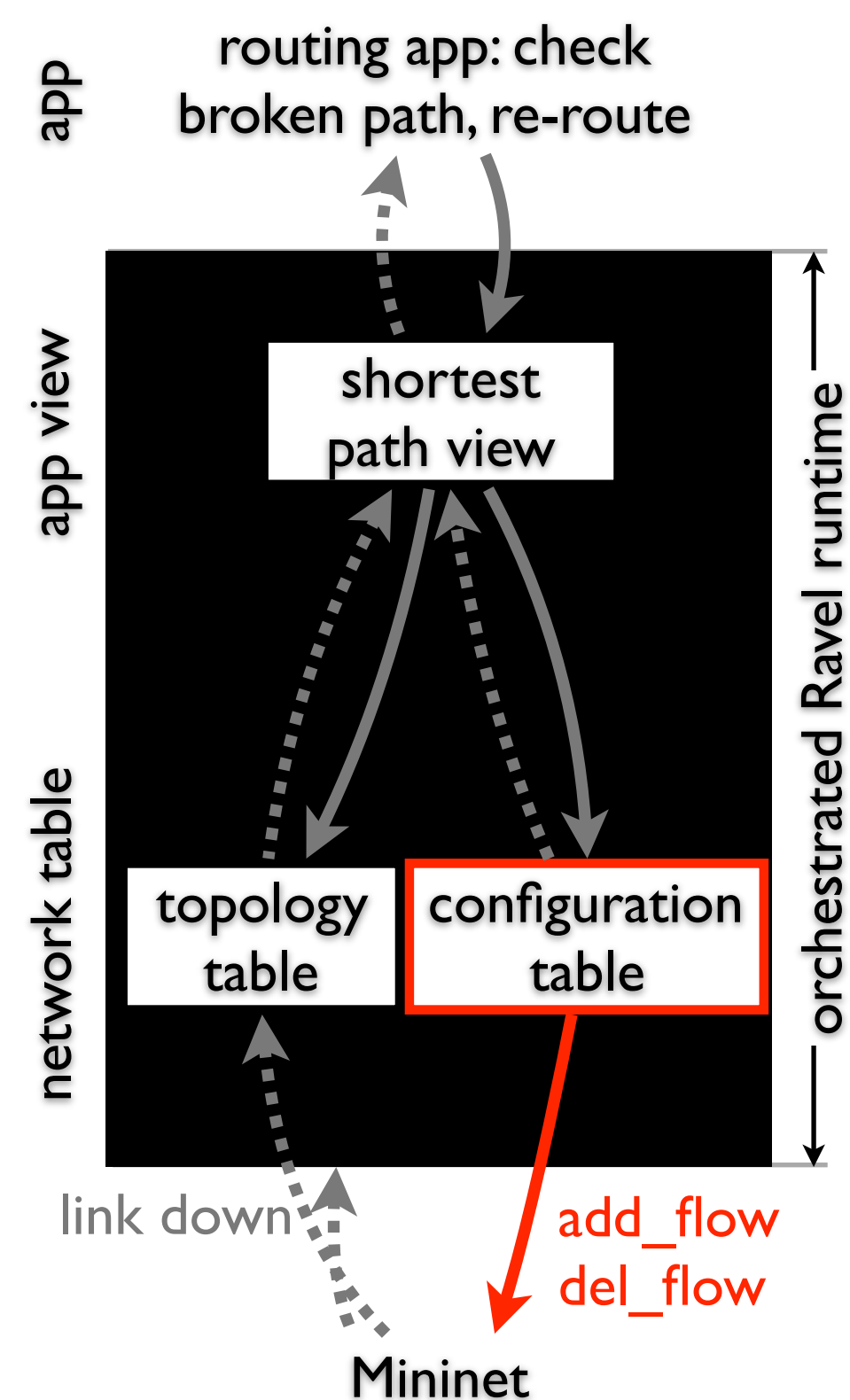
orchestration across representations



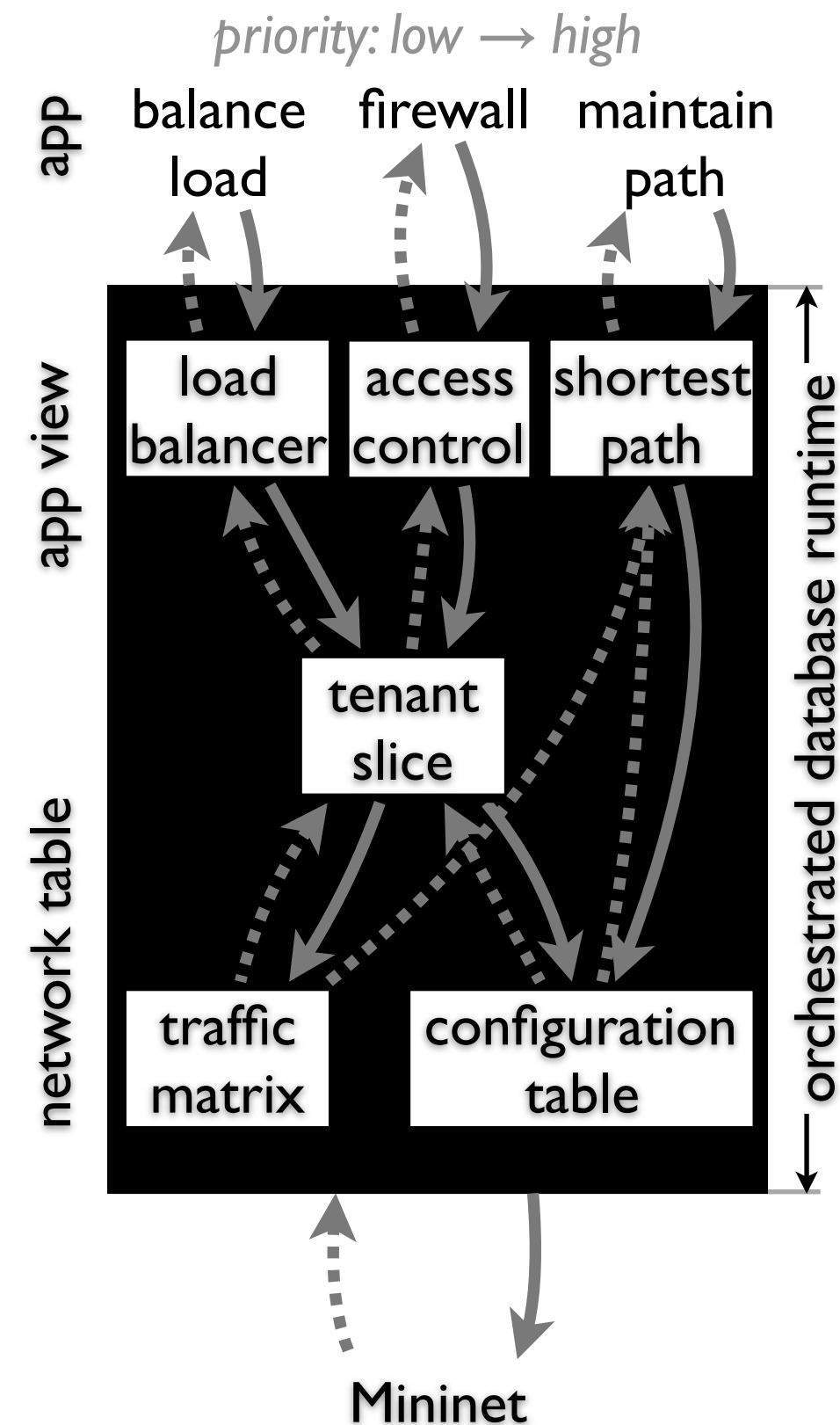
orchestration across representations



orchestration across representations



orchestration across applications



load balancer	

access control		

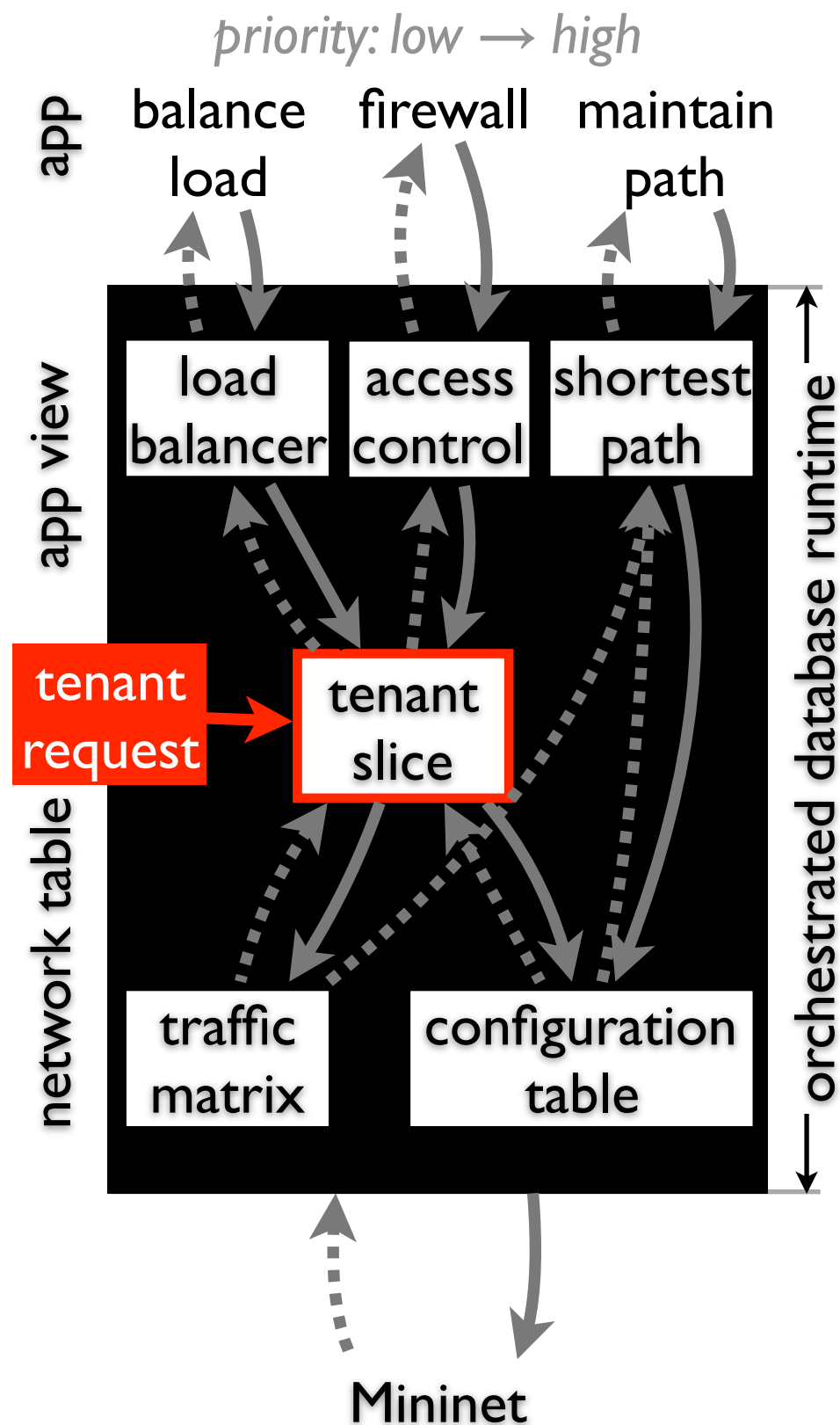
shortest path	

tenant policy		

traffic matrix		

configuration		

orchestration across applications



load balancer	

access control		

shortest path	

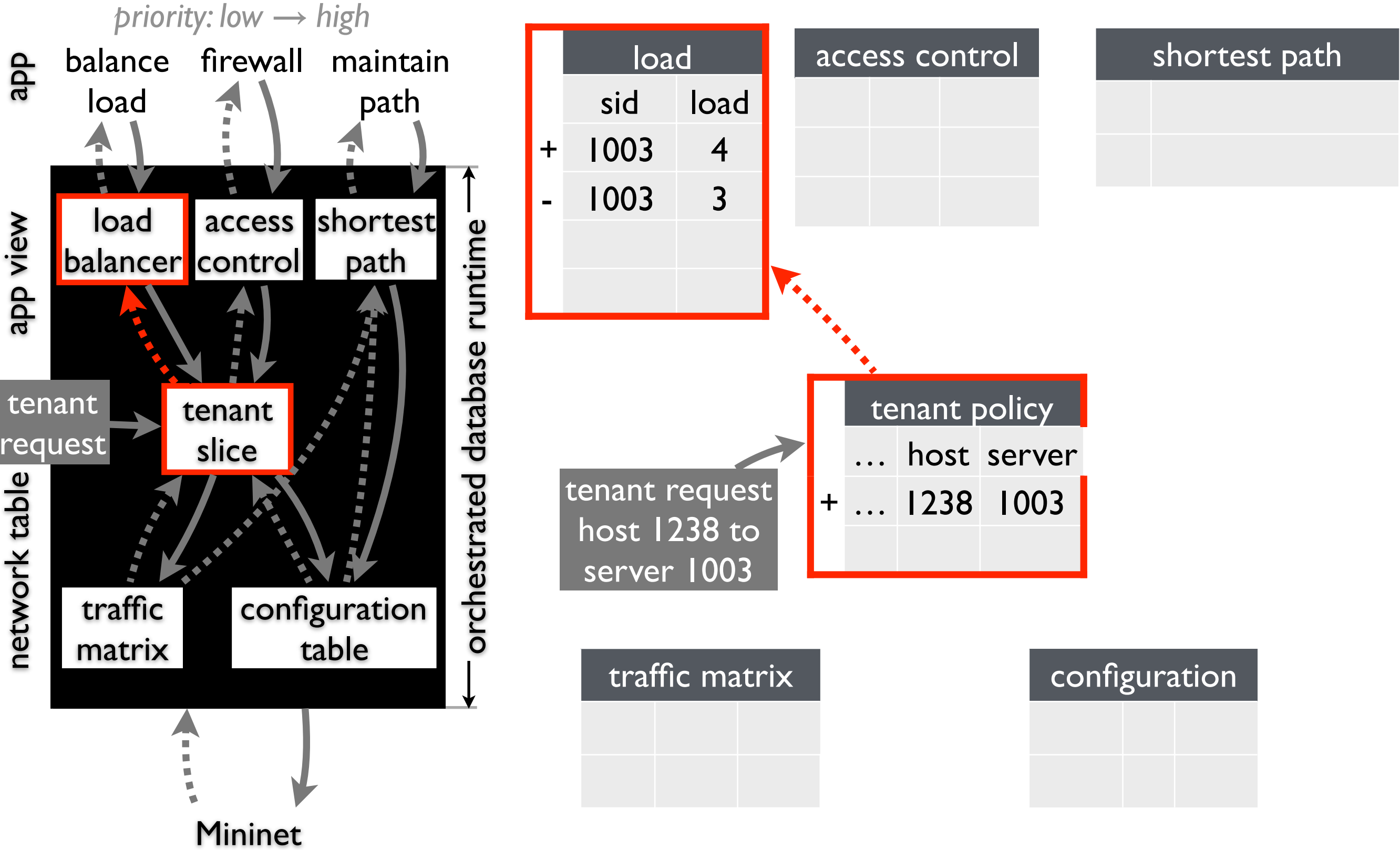
tenant policy		
...	host	server
+	...	1238 1003

tenant request
host 1238 to
server 1003

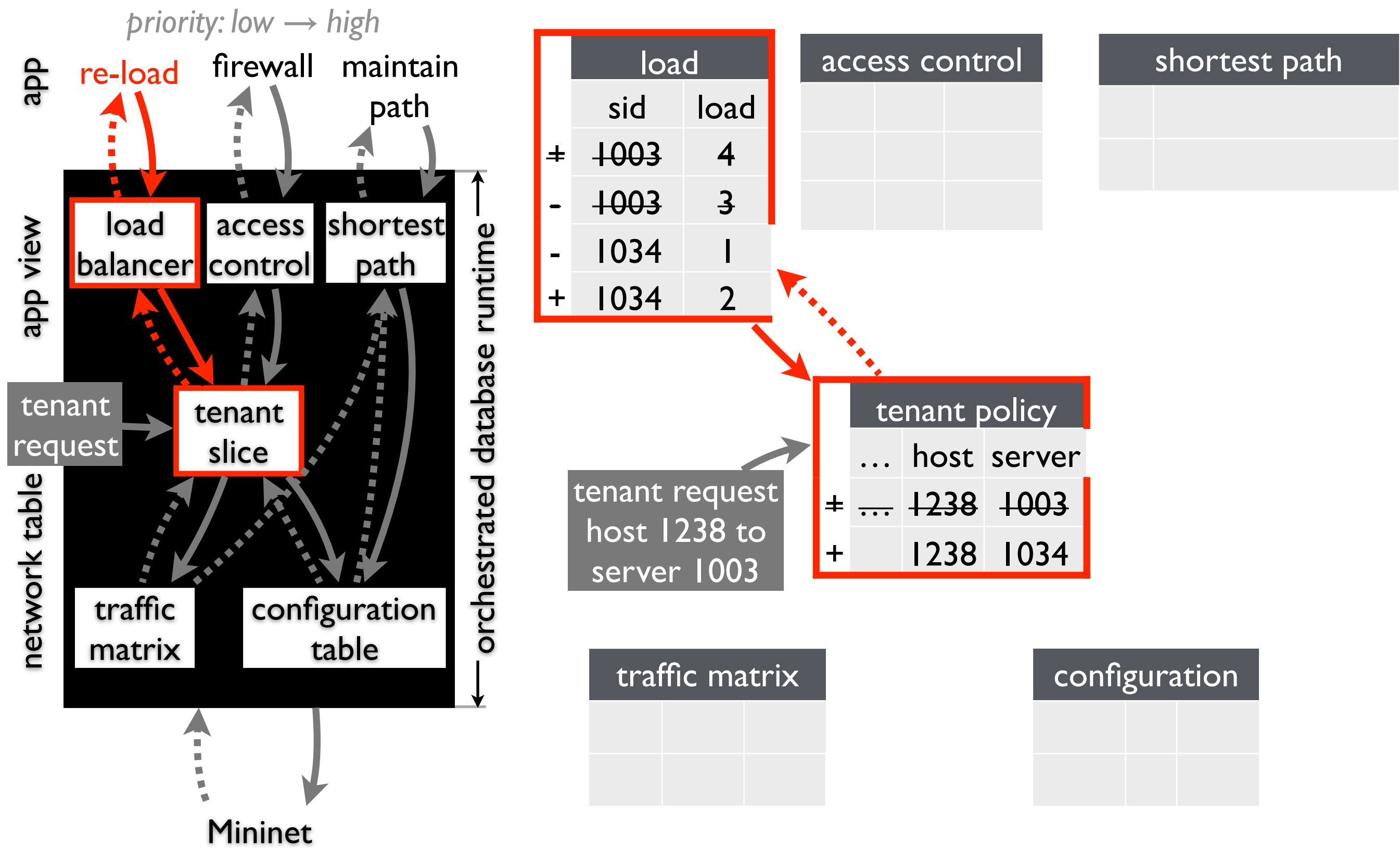
traffic matrix		

configuration		

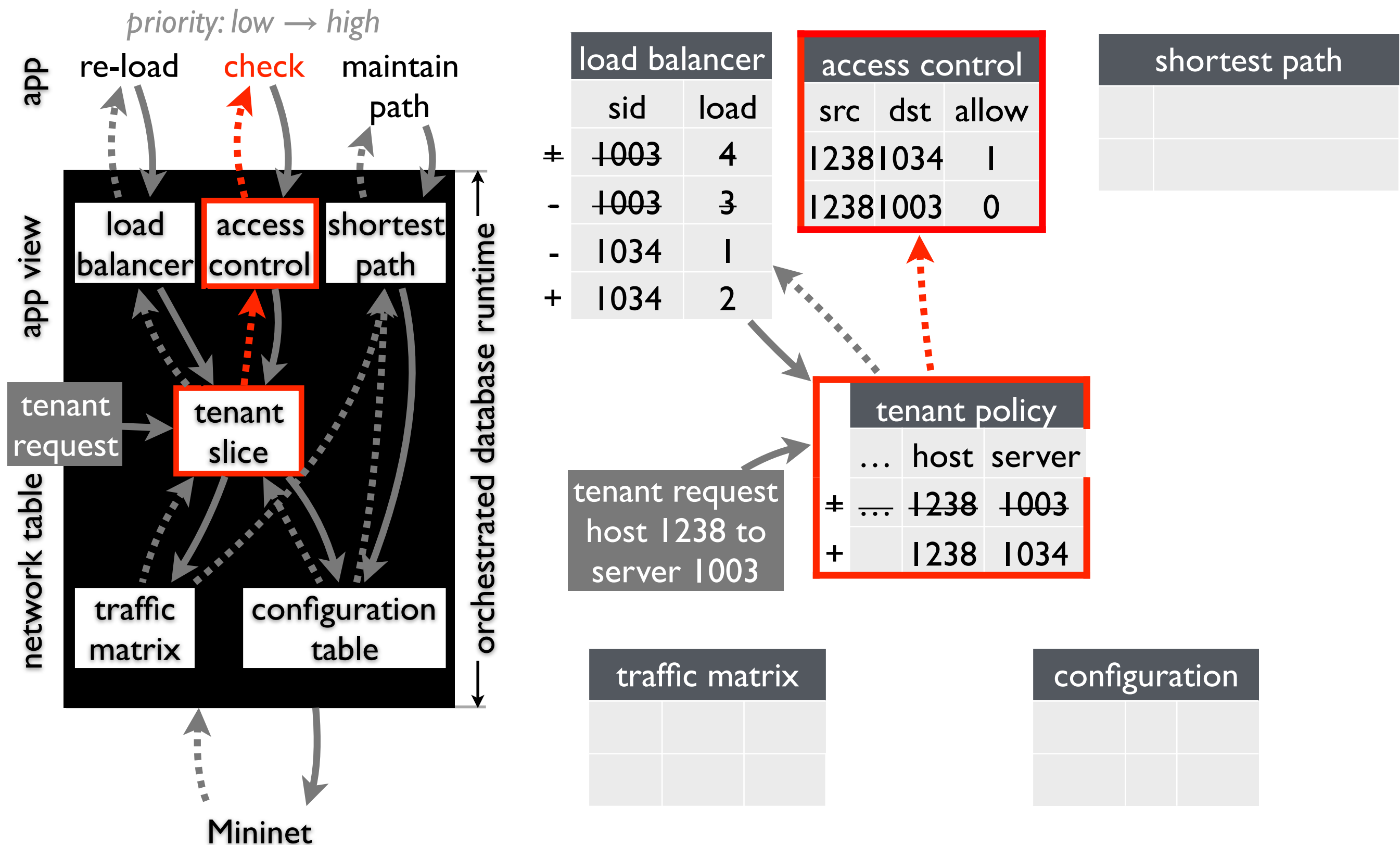
orchestration across applications



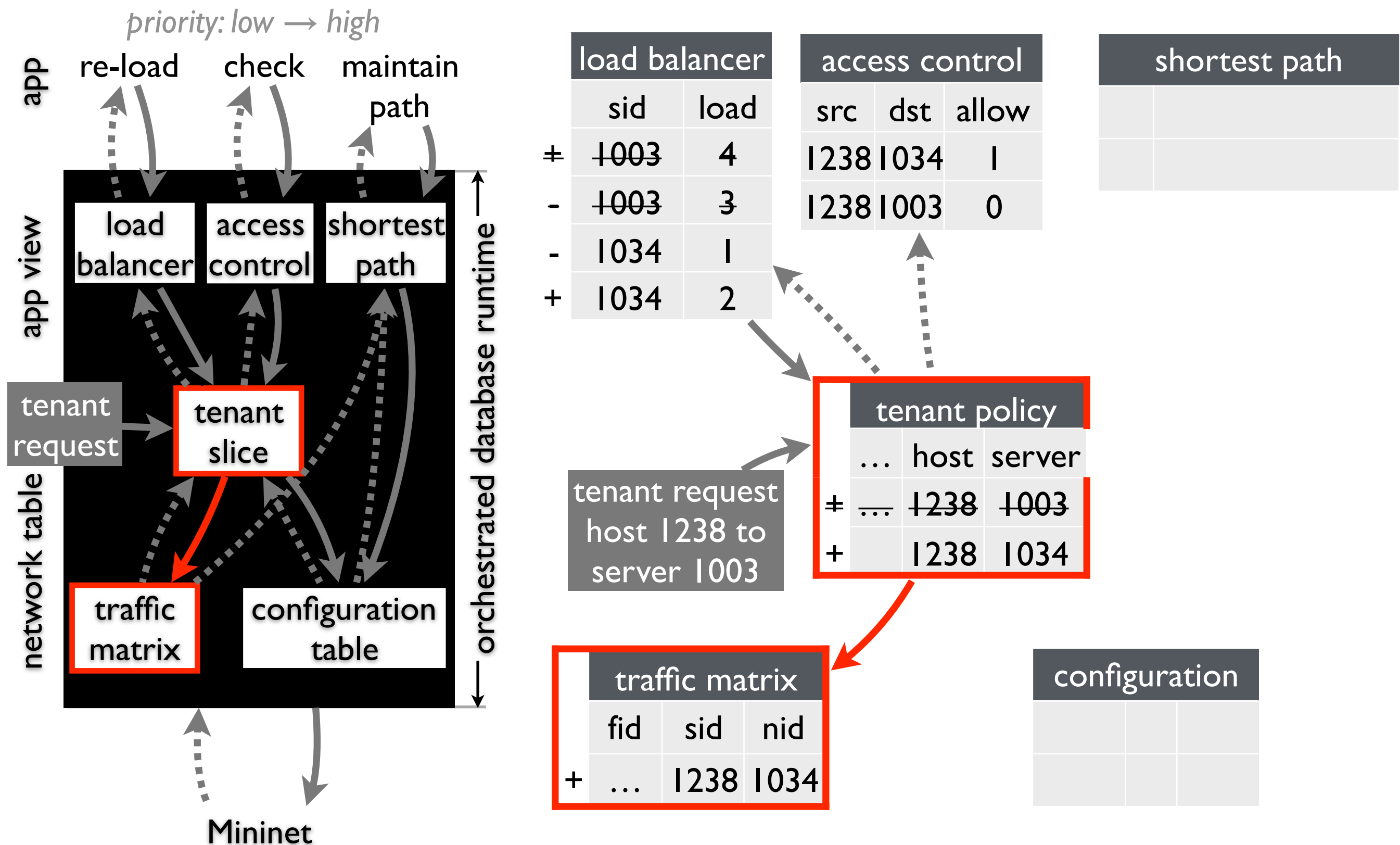
orchestration across applications



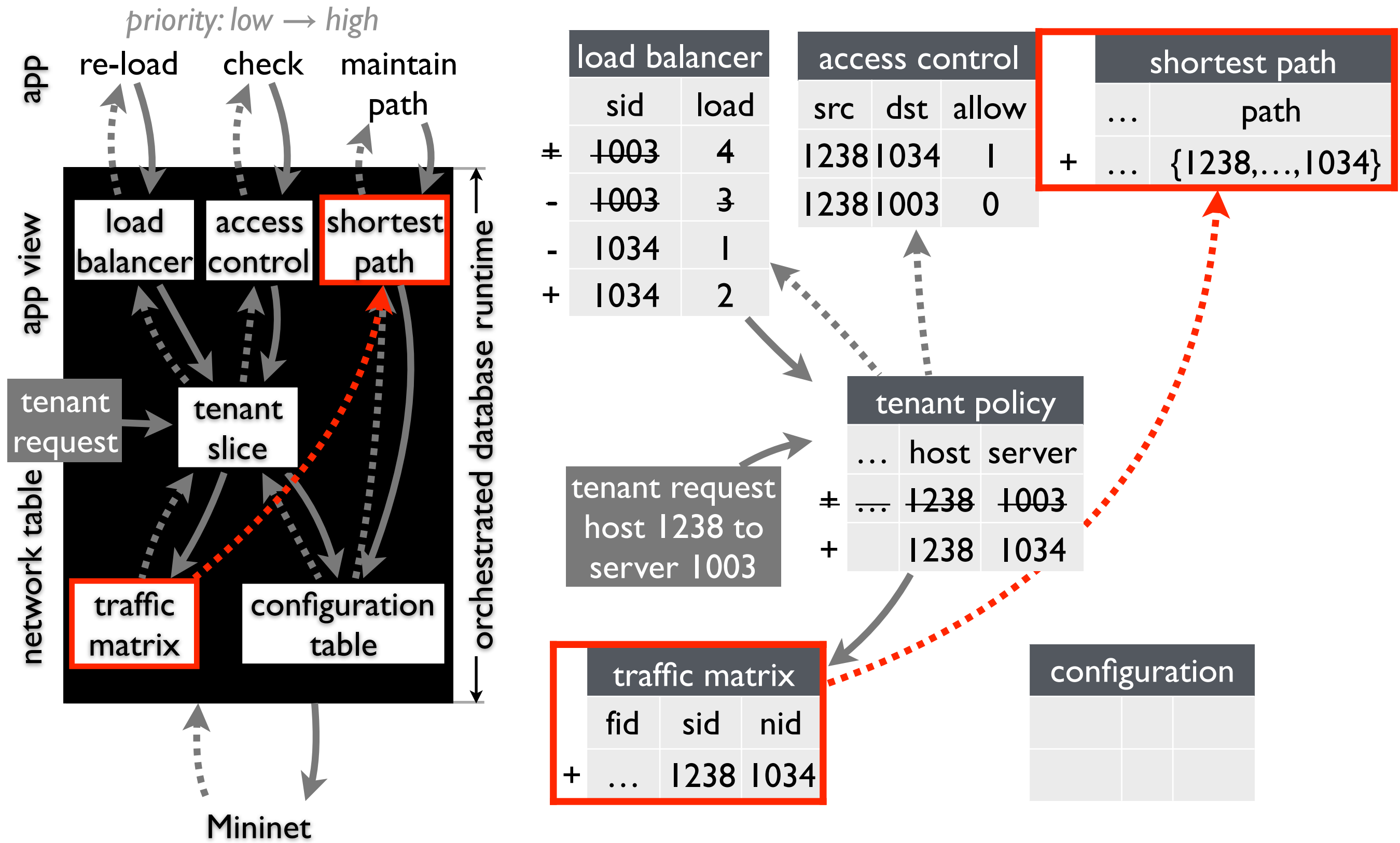
orchestration across applications



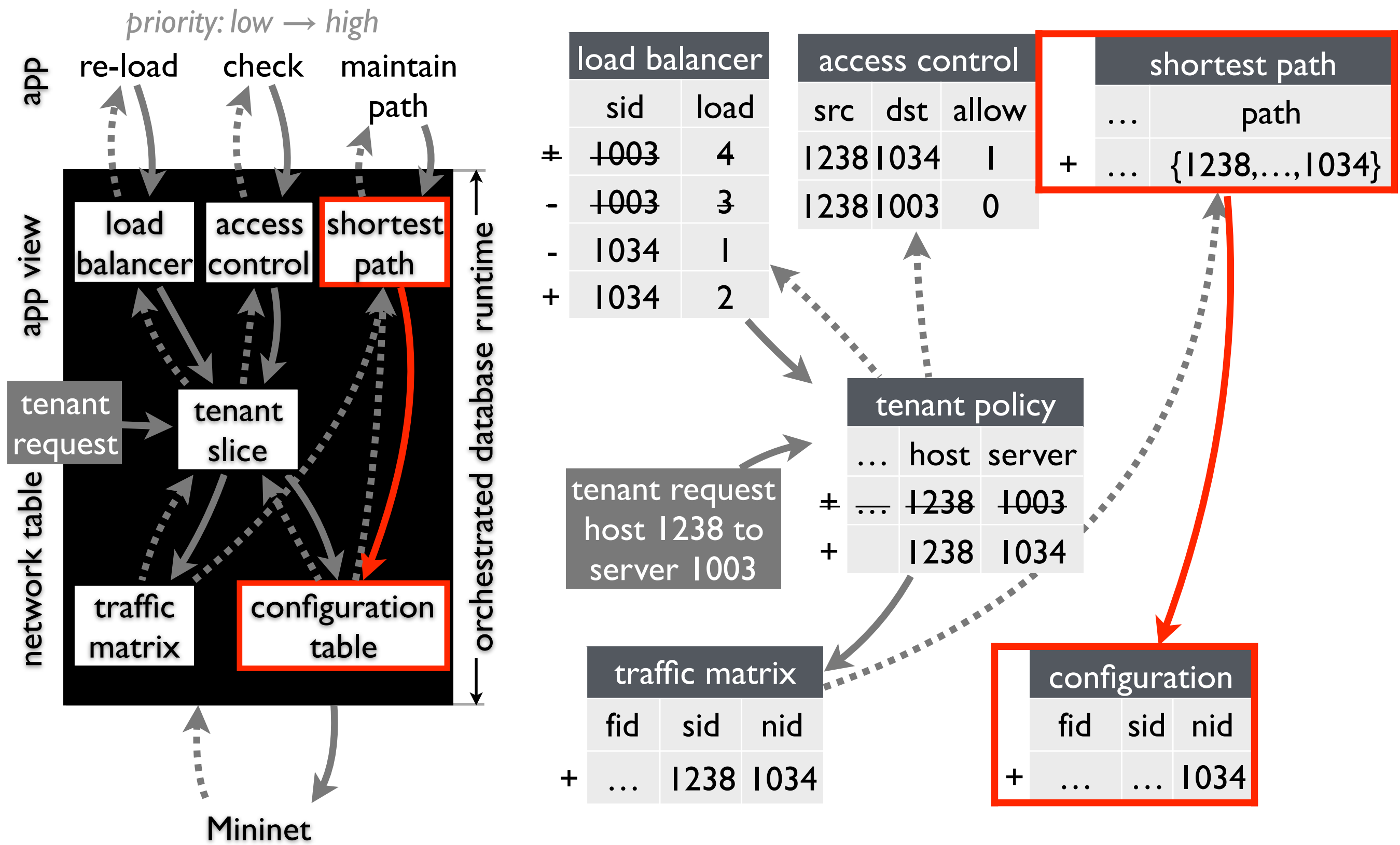
orchestration across applications



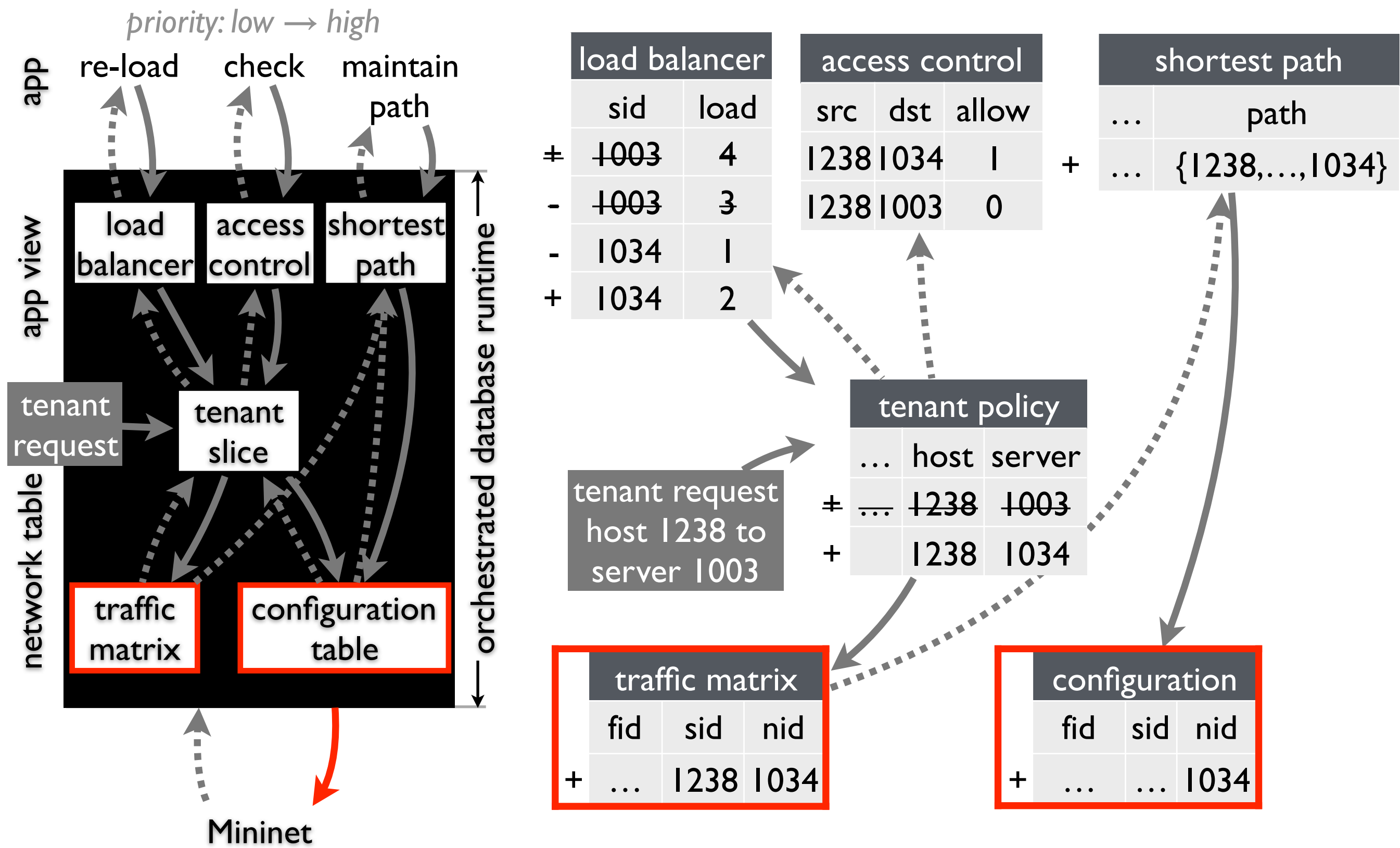
orchestration across applications



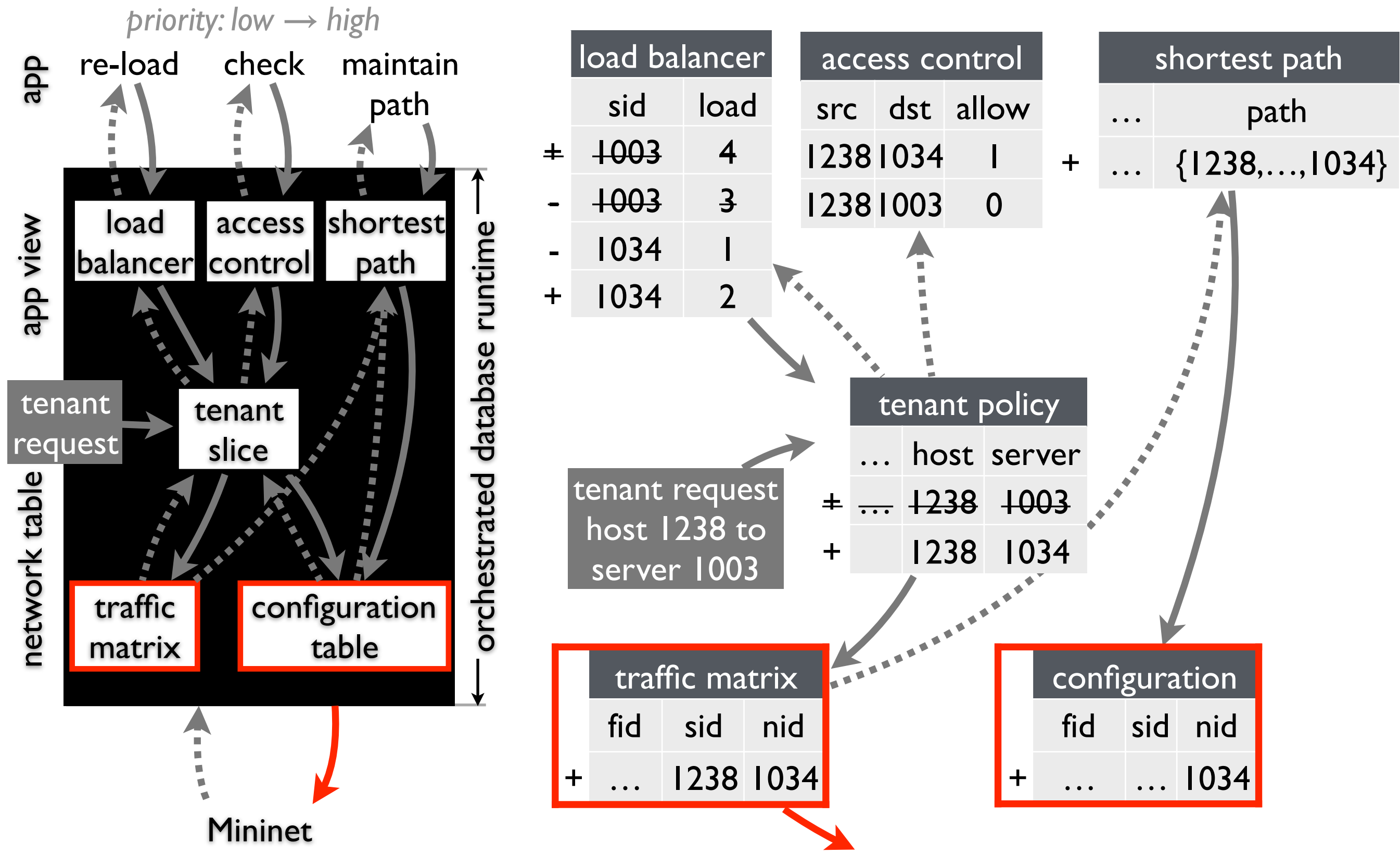
orchestration across applications



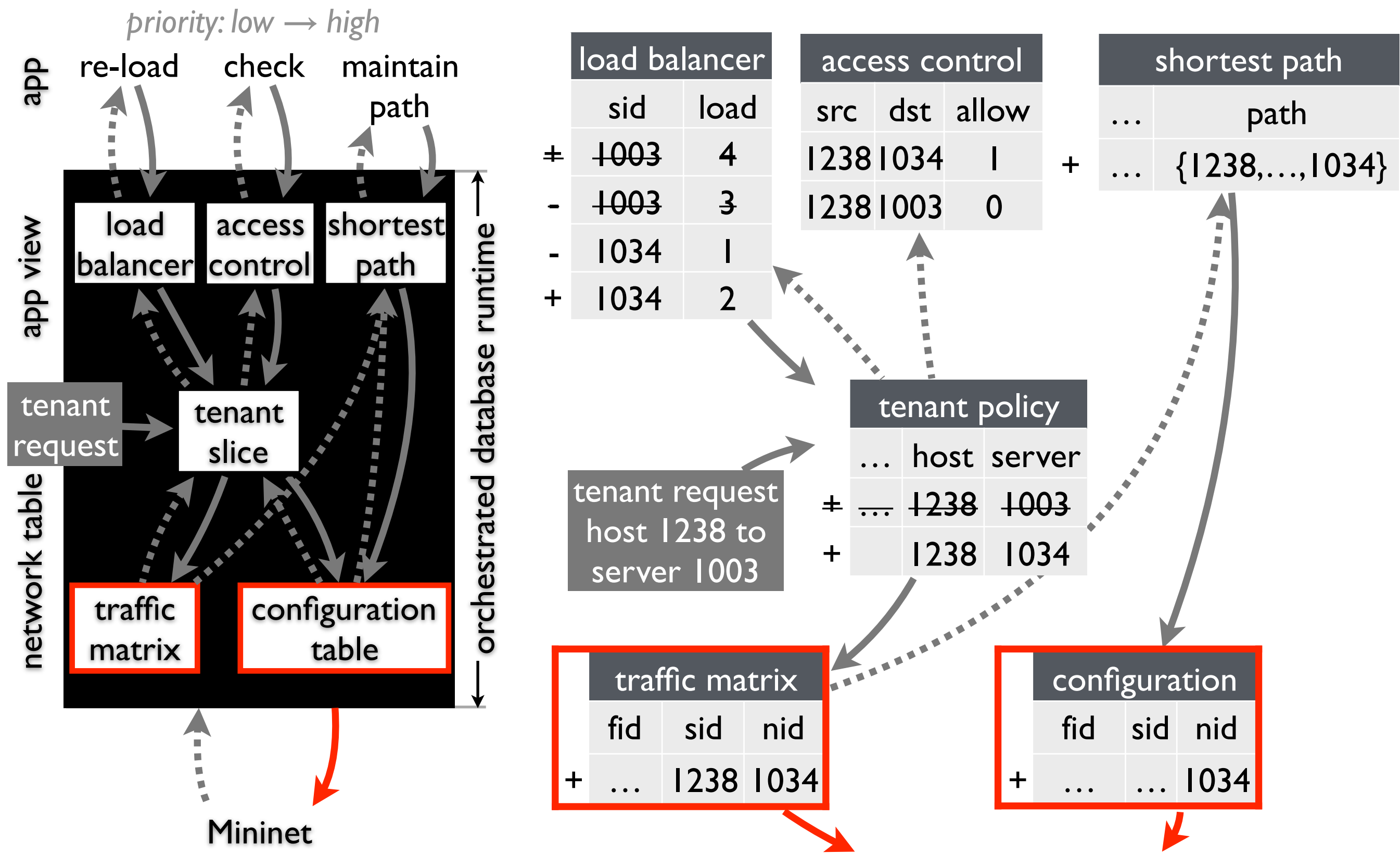
orchestration across applications



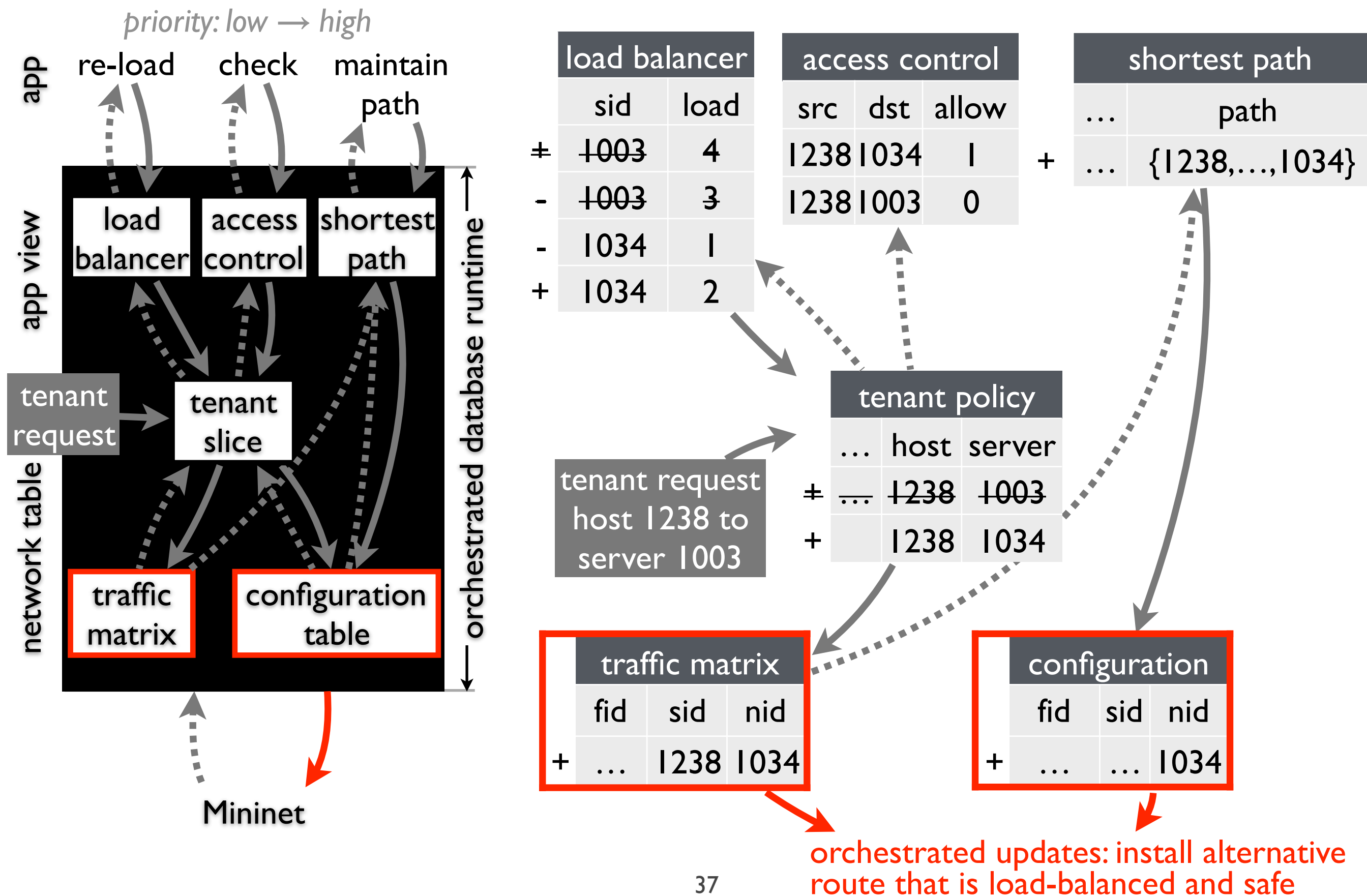
orchestration across applications



orchestration across applications

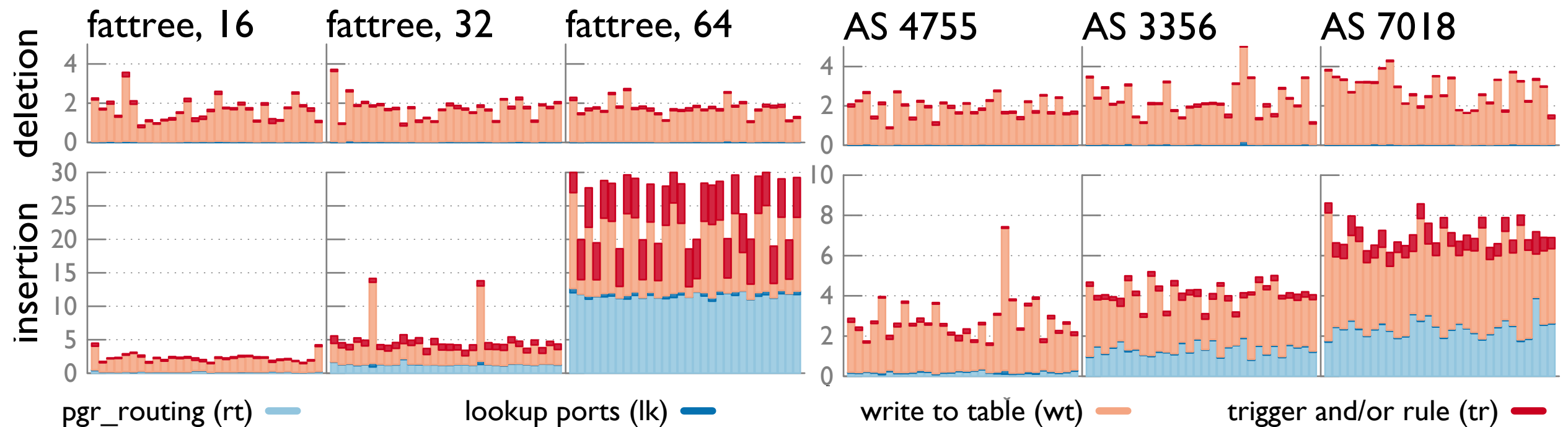


orchestration across applications



evaluation

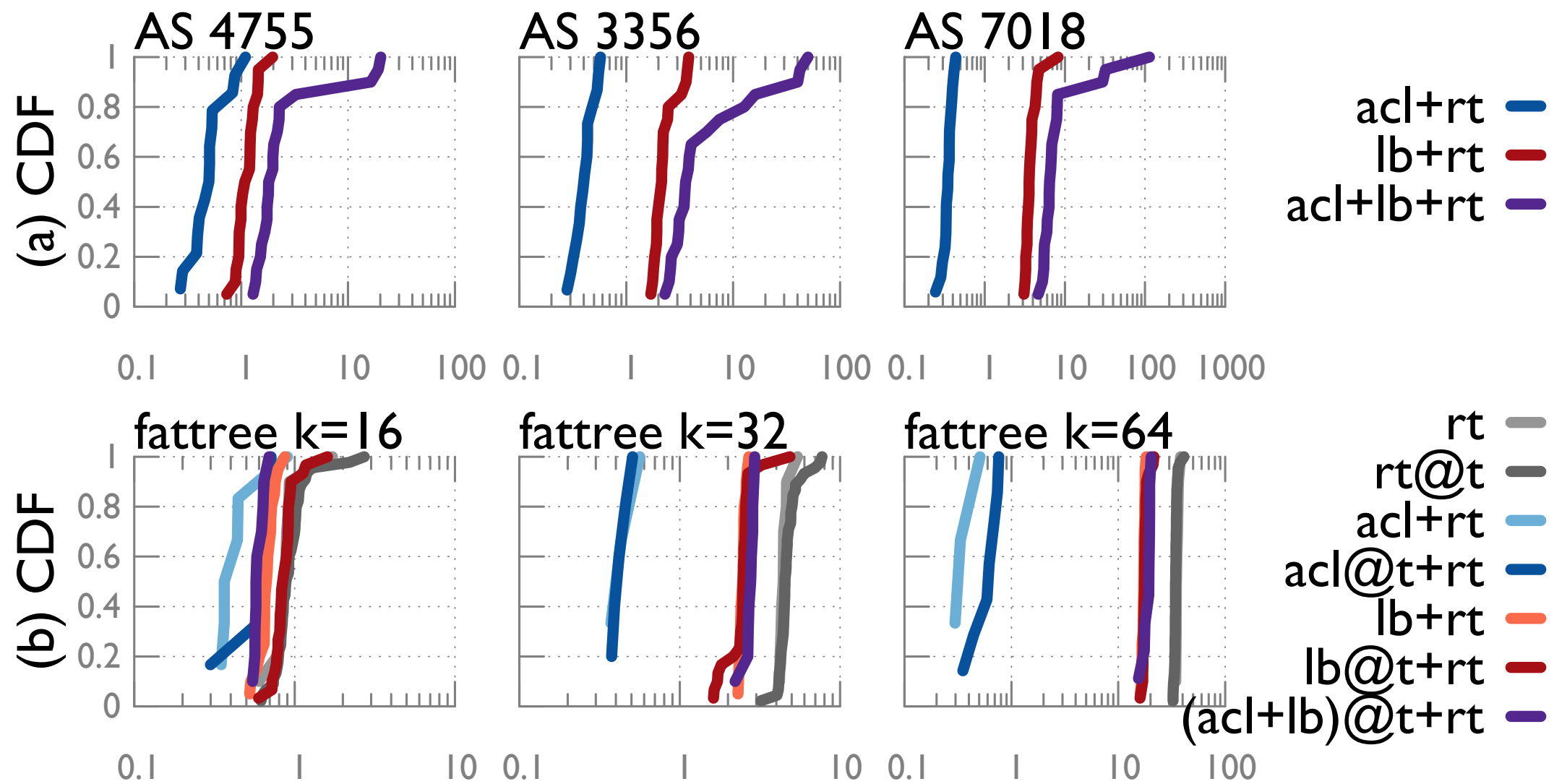
profiling database delay — route insertion/deletion



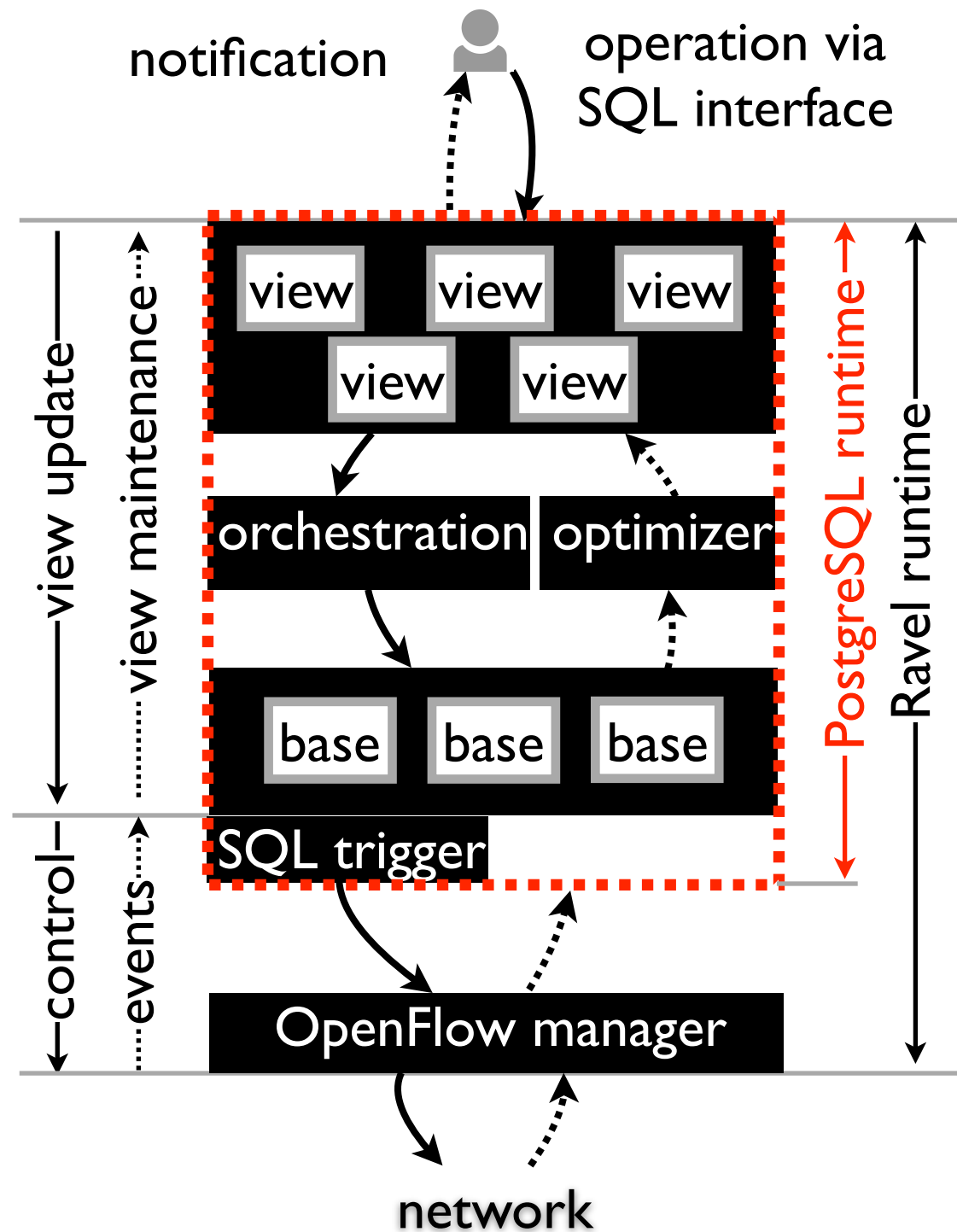
fat-tree			ISP		
k	switches	links	AS#	nodes	links
16	320	3072	4755	142	258
32	1280	24576	3356	1772	13640
64	5120	196608	7018	25382	11292
			2914	5939	16520

evaluation

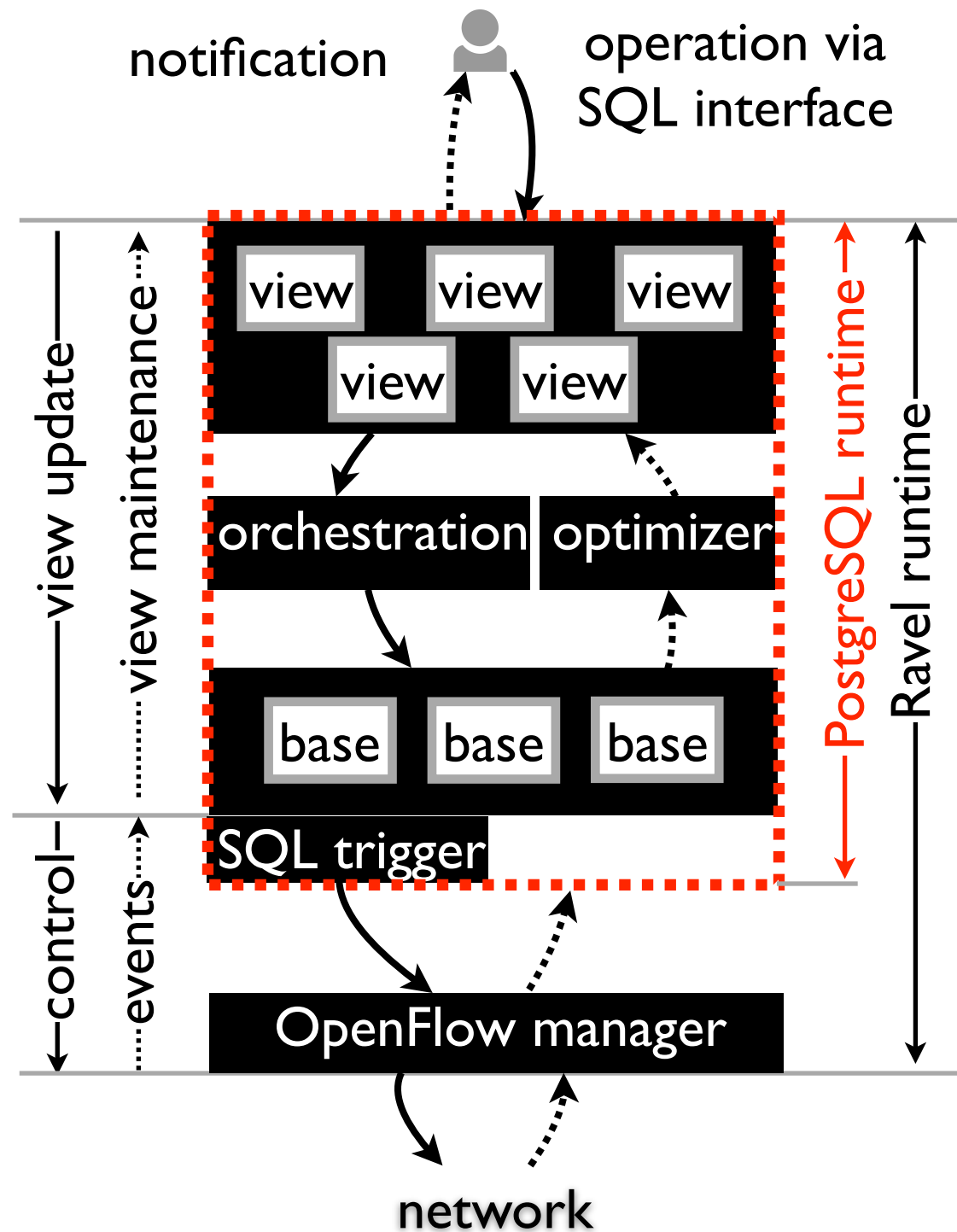
orchestrating access control(acl), load balancer(lb), and routing(rt): normalized per-rule delay (ms)



conclusion



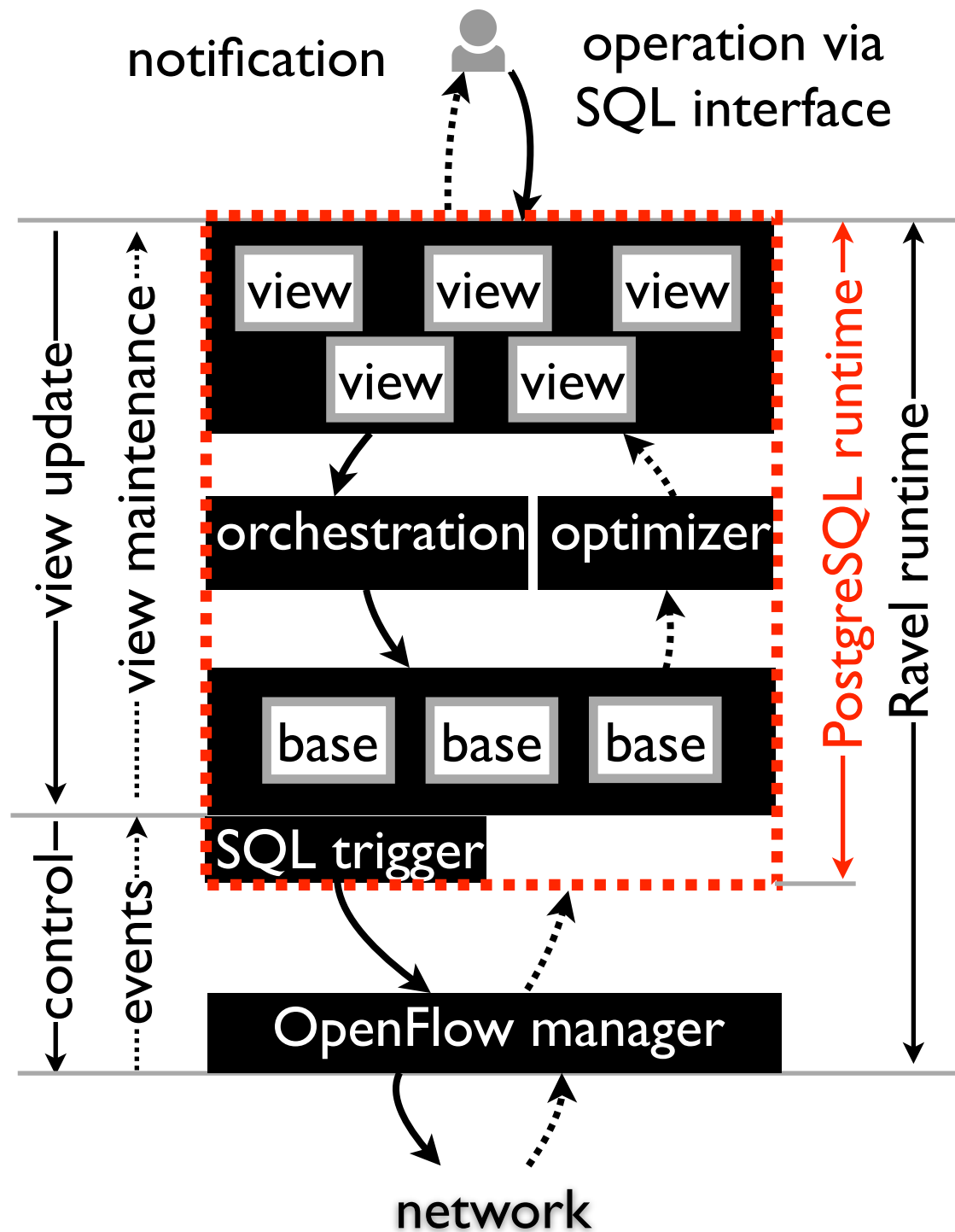
conclusion



flexible abstraction via SQL

- ad-hoc extensible, orchestratable
- promising performance

conclusion



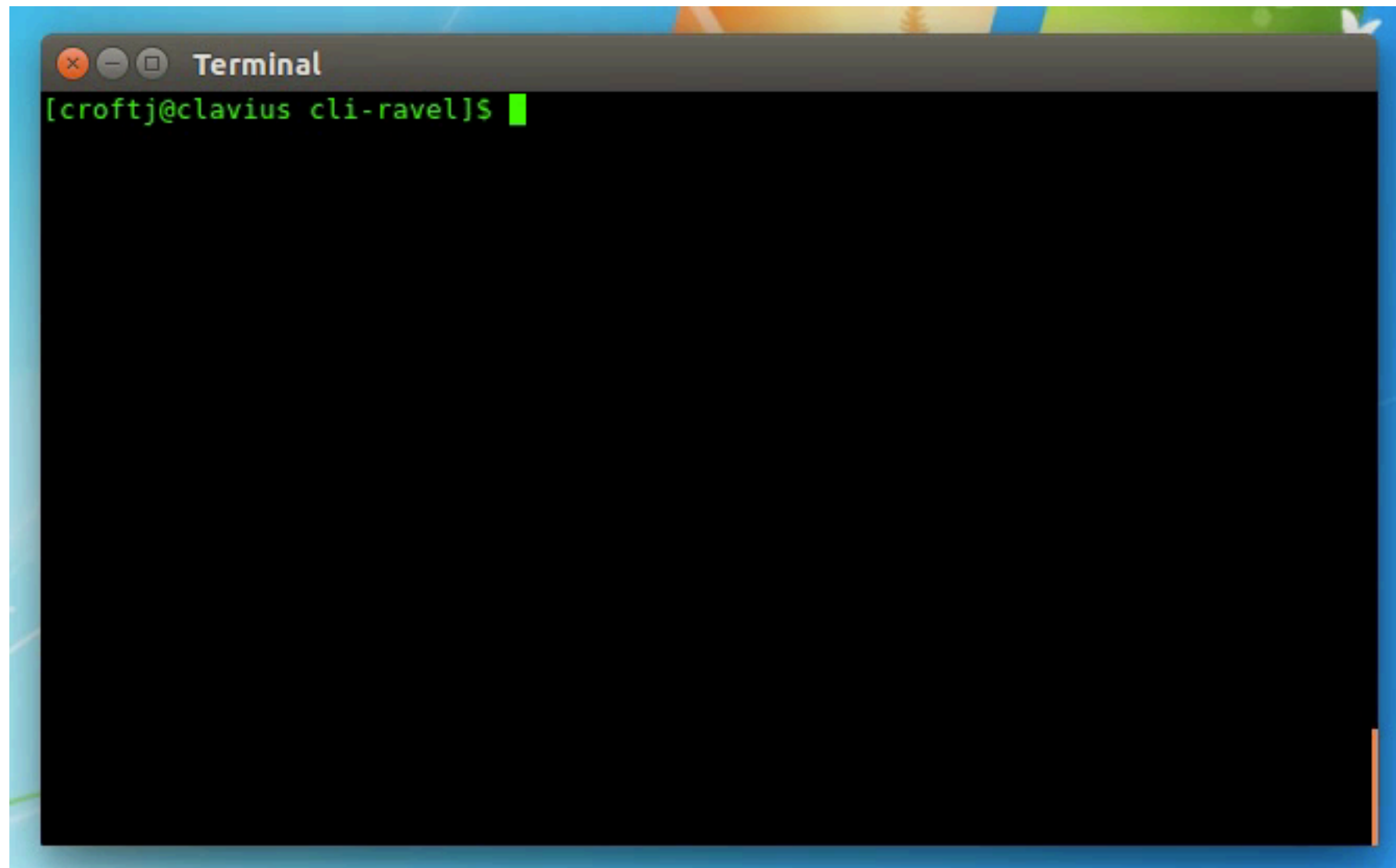
flexible abstraction via SQL

- ad-hoc extensible, orchestratable
- promising performance

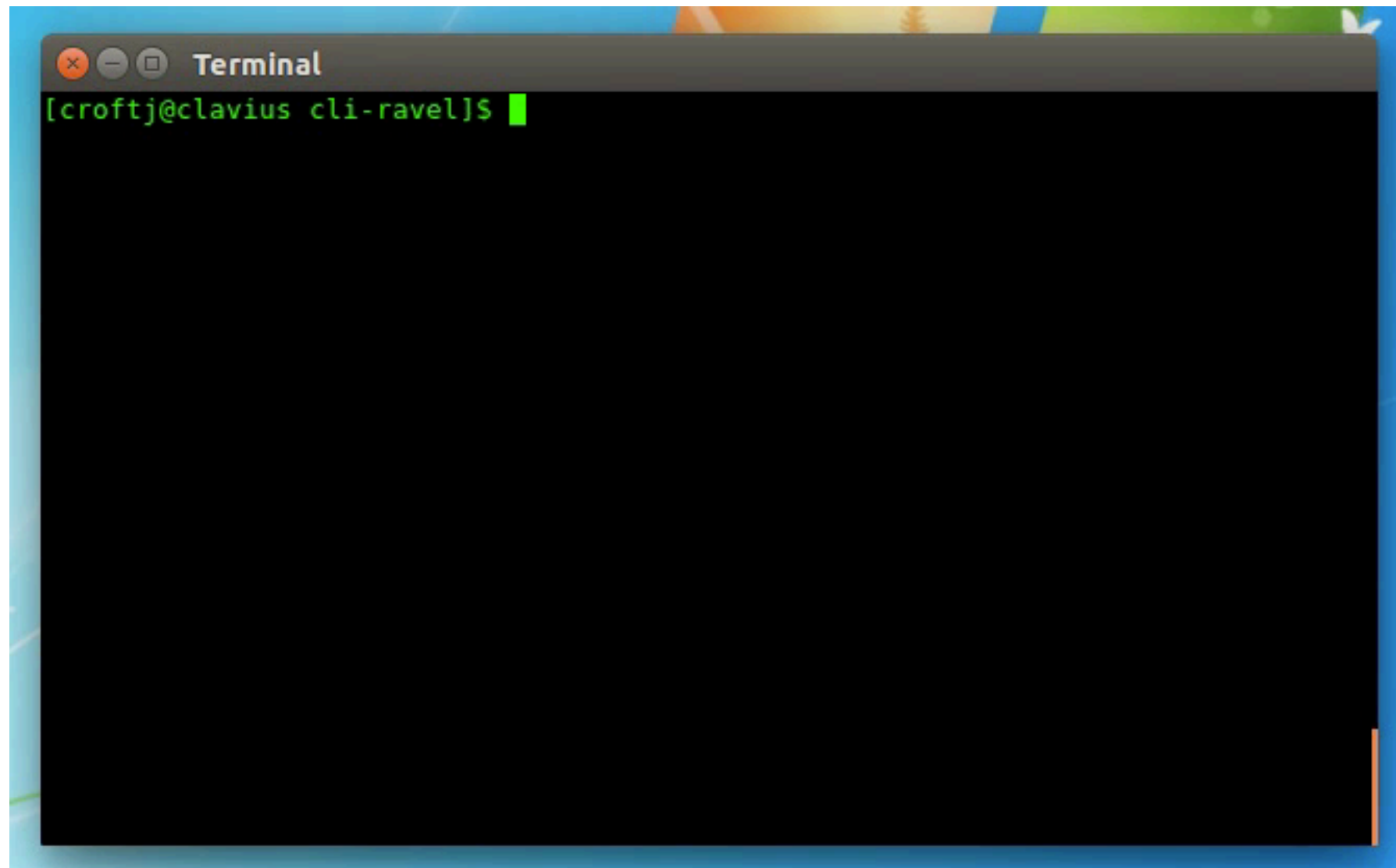
looking forward

- application of database features
 - network-wide transaction
 - bootstrapping legacy networks
- enhancing database
 - better runtime: orchestration
 - better control decision: view analysis
- interpretability
 - integrate foreign applications, plug-n-play 3rd party solvers

demo



demo





playtime

download *Ravel* and install

ravel-net.org/download

start playing: tutorials, add your own app

ravel-net.org

more to explore

github.com/ravel-net