SDN abstraction and security: a database perspective

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

*Temple University  †University of Illinois Urbana-Champaign
software-defined network

control applications of disparate nature

forwarding  service chain  stateful middlebox ...

controller

OpenFlow network

switch  switch  switch  switch
software-defined network

control applications of disparate nature

forwarding  service chain  stateful middlebox ...

controller (abstraction runtime)

an insertion point for network abstractions

OpenFlow network

switch  switch  switch  switch
what is the right abstraction?

- forwarding
- service chain
- stateful middlebox

abstraction runtime

OpenFlow rules
What is the right abstraction?

- Functions
- Service chain
- Stateful middlebox

Frenetic / Pyretic

OpenFlow rules

[NSDI’13] [PLDI’13]
abstractions

what is the right abstraction?

functions

graphs

stateful middlebox

PGA

[SIGCOMM’15]

OpenFlow rules
What is the right abstraction?

functions  graphs  automata  ...

Kinetic

[NSDI'15]

OpenFlow rules
abstractions

diverse abstractions

functions graphs automata ...

abstraction runtime

OpenFlow rules
but network keeps evolving

functions  graphs  automata

new/changing requirements

abstraction runtime

OpenFlow rules
but network keeps evolving

functions  graphs  automata  new structure

add / re-engineer runtime

OpenFlow rules
and applications (components) interact
and applications (components) interact
and applications (components) interact

language-level orchestration restricted to each abstraction

composing (+) policy
→ graph +? automata

how to integrate the runtime?
hard-wire internals?
and applications (components) interact

- **graphs**
- **automata**

- **PGA**
- **Kinetic**

- **functions**

- **Pyretic**

- **OpenFlow rules**
  - network

- **policies**

- language-level orchestration restricted to each abstraction

- abstraction-agonistic coordination often low-level
  - Co-visor [NSDI’15]
  - statesman [SIGCOMM’14]
current state of abstraction research

OpenFlow rules
network
current state of abstraction research

enlarging body of abstractions

OpenFlow rules
network
current state of abstraction research

- Structure
- Runtime

- Structure
- Runtime

- New structure
- New runtime

- Enlarging body of abstractions

- Fragmented orchestration

- OpenFlow rules
- Network
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

operator and/or application

view

new view

view

table

table

OpenFlow rules network

high-level app views

collection

low-level inventory tables
a database-defined network

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

operator and/or application

view
new view

view

view

table

OpenFlow rules network

high-level app views

low-level inventory tables
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
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
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
Ravel: a realization with SQL database

- Ad-hoc programmable abstraction via views
- Orchestration across abstractions via view mechanism
- Orchestration across applications via data mediation
- Network control via SQL
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
Ravel: a realization with SQL database

attractive features

- abstraction
- orchestration
- SQL

Ravel: a realization with SQL database

attractive features

- abstraction
- orchestration
- SQL
abstraction: network tables

**reachability matrix**

<table>
<thead>
<tr>
<th>fid</th>
<th>src</th>
<th>dst</th>
<th>vol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>h₁</td>
<td>h₄</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>h₂</td>
<td>h₃</td>
<td>9</td>
</tr>
</tbody>
</table>

**topology**

<table>
<thead>
<tr>
<th>sid</th>
<th>nid</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁</td>
<td>S₂</td>
</tr>
<tr>
<td>S₁</td>
<td>S₃</td>
</tr>
<tr>
<td>S₁</td>
<td>h₁</td>
</tr>
</tbody>
</table>

**configuration**

<table>
<thead>
<tr>
<th>fid</th>
<th>sid</th>
<th>nid</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S₁</td>
<td>S₄</td>
</tr>
<tr>
<td>1</td>
<td>S₄</td>
<td>h₄</td>
</tr>
</tbody>
</table>

**Flow 1**

- h₁
- S₁
- h₄

**Flow 2**

- h₂
- S₂
- h₃
CREATE TABLE acl (
  end1 integer, end2 integer, allow integer
);

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

CREATE RULE acl_repair AS 
  ON DELETE TO acl_violation 
  DO INSTEAD 
    DELETE FROM rm WHERE fid = OLD.fid;
abstraction: application view

firewall view: monitoring unsafe flows violating acl policy

```
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 
);
```

firewall control: 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 …
orchestration across representations

routing app: check broken path, re-route

SQL rule: upon broken path, re-route

<table>
<thead>
<tr>
<th>shortest path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>topology</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
orchestration across representations

routing app: check broken path, re-route

Mininet

SQL rule: upon broken path, re-route

<table>
<thead>
<tr>
<th>topology</th>
<th>sid</th>
<th>nid</th>
<th>active</th>
</tr>
</thead>
<tbody>
<tr>
<td>-172 39</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Mininet link (172,39) down
orchestration across representations

routing app: check broken path, re-route

SQL rule: upon broken path, re-route

Mininet link (172,39) down

<table>
<thead>
<tr>
<th>topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>sid</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>-</td>
</tr>
<tr>
<td>+</td>
</tr>
</tbody>
</table>

Mininet link (172,39) down
orchestration across representations

Routing app: check broken path, re-route

Mininet

SQL rule: upon broken path, re-route

Topo table:

<table>
<thead>
<tr>
<th>sid</th>
<th>nid</th>
<th>active</th>
</tr>
</thead>
<tbody>
<tr>
<td>172</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>172</td>
<td>39</td>
<td>0</td>
</tr>
</tbody>
</table>

Mininet link (172,39) down
orchestration across representations

routing app: check broken path, re-route

SQL rule: upon broken path, re-route

Mininet link (172,39) down
routing app: check broken path, re-route

SQL rule: upon broken path, re-route

Mininet link (172,39) down

Shortest path update:
- {...,172,39,156,...}
+ {...,172,38,148,...}

Configuration update:
- 
  fid   sid   nid
  ...   172   39
  ...   39   156
+ 
  ...   172   38
  ...   38   148

Mininet link (172,39) down

orchestrated updates: re-route via (172, 38)
orchestration across applications

priority: low → high

balance → firewall → maintain

load balancer
access control
shortest path

tenant virtual net

load balancer
access control
shortest path

tenant virtual net

reachability matrix
configuration

Mininet
orchestration across applications

priority: low → high

load balancer
access control
shortest path

load balancer
access control
shortest path

network table
network table
Mininet

reachability matrix
configuration table

reachability matrix
configuration table

tenant virtual net

tenant request
tenant virtual net

tenant request host 1238 to server 1003

tenant virtual net

... host server
... 1238 1003
orchestration across applications

priority: low → high

app view

load balancer
access control
shortest path

load balancer
sid load
+ 1003 4
- 1003 3

access control

shortest path

load balancer
sid load
+ 1003 4
- 1003 3

tenant virtual net

tenant request host 1238 to server 1003

reachability matrix

tenant request

network table

tenant virtual net

host 1238 to server 1003

Mininet
orchestration across applications

priority: low → high

load balancer

access control

shortest path

reachability matrix

configuration table

Mininet

app

network table

tenant request

request host 1238 to server 1003

tenant virtual net

load balancer

access control

shortest path

reachability matrix

configuration table

tenant request host 1238 to server 1003

load balancer

access control

shortest path

reachability matrix

configuration table

tenant virtual net

load balancer

access control

shortest path

reachability matrix

configuration table

tenant virtual net
orchestration across applications

priority: low → high

load balancer
- sid  load
  - 1003  4
  - 1003  3
  - 1034  1
  + 1034  2

access control
- src  dst  allow
  - 12381034  1
  - 12381003  0

tenant virtual net
+ host  server
  + ...  1238  1003
  + 1238  1034

reachability matrix

configuration
orchestration across applications

priority: low → high

load balancer
<table>
<thead>
<tr>
<th>sid</th>
<th>load</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ 1003</td>
<td>4</td>
</tr>
<tr>
<td>- 1003</td>
<td>3</td>
</tr>
<tr>
<td>- 1034</td>
<td>1</td>
</tr>
<tr>
<td>+ 1034</td>
<td>2</td>
</tr>
</tbody>
</table>

access control
<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>allow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1238</td>
<td>1034</td>
<td>1</td>
</tr>
<tr>
<td>1238</td>
<td>1003</td>
<td>0</td>
</tr>
</tbody>
</table>

shortest path

tenant virtual net
<table>
<thead>
<tr>
<th>...</th>
<th>host</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>...</td>
<td>1238</td>
</tr>
<tr>
<td>+</td>
<td>1238</td>
<td>1003</td>
</tr>
<tr>
<td>+</td>
<td>1238</td>
<td>1034</td>
</tr>
</tbody>
</table>

reachability matrix
<table>
<thead>
<tr>
<th>fid</th>
<th>sid</th>
<th>nid</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>...</td>
<td>1238</td>
</tr>
<tr>
<td>+</td>
<td>1238</td>
<td>1034</td>
</tr>
</tbody>
</table>

Mininet
orchestration across applications

priority: low → high

load balancer

access control

shortest path

reachability matrix

configuration table

tenant virtual net

tenant request

host 1238 to server 1003

load balancer

access control

shortest path

reachability matrix

configuration
Mininet

orchestrated database runtime

network table

app view

priority: low → high

re-load check maintain path

load balancer access control shortest path

load balancer:

<table>
<thead>
<tr>
<th>sid</th>
<th>load</th>
</tr>
</thead>
<tbody>
<tr>
<td>1003</td>
<td>4</td>
</tr>
<tr>
<td>1003</td>
<td>3</td>
</tr>
<tr>
<td>1034</td>
<td>1</td>
</tr>
<tr>
<td>1034</td>
<td>2</td>
</tr>
</tbody>
</table>

access control:

<table>
<thead>
<tr>
<th>src</th>
<th>dst</th>
<th>allow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1238</td>
<td>1034</td>
<td>1</td>
</tr>
<tr>
<td>1238</td>
<td>1003</td>
<td>0</td>
</tr>
</tbody>
</table>

shortest path:

<table>
<thead>
<tr>
<th>+</th>
<th>...</th>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>...</td>
<td>{1238,...,1034}</td>
</tr>
</tbody>
</table>

tenant virtual net

tenant request host 1238 to server 1003

reachability matrix

<table>
<thead>
<tr>
<th>fid</th>
<th>sid</th>
<th>nid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>1238</td>
<td>1034</td>
</tr>
</tbody>
</table>

configuration

<table>
<thead>
<tr>
<th>fid</th>
<th>sid</th>
<th>nid</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>1034</td>
</tr>
</tbody>
</table>
**orchestration across applications**

priority: low → high

re-load  check  maintain

load balancer  access control  shortest path

load balancer
- 1003  3
- 1034  1
+ 1034  2

access control
- 1238 1003  0
+ 1238 1034  1

shortest path
... path
... {1238,...,1034}

tenant virtual net
+ 1238 1003
+ 1238 1034

tenant request
host 1238 to server 1003

orchestrated updates: install alternative route that is load-balanced and safe

reachability matrix
+ ... 1238 1034
+ ... ... 1034

configuration

Mininet
achieving *Ravel* advantages

attractive features

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

- challenge: inefficient user view
- solution: optimizer
  - materialize user view with fast maintenance algorithm
  - one order of magnitude faster access with small maintenance overhead — 0.01~10ms
orchestration across applications

- challenge: database lacking inter-view support
- solution: mediation protocol
  - translate app priority into view updates that dynamically merge into a coherent data plane
- SDN control via SQL
  - challenge: database lacks connection to network data plane
  - solution: SQL trigger + OF manager
a high-performance runtime
- PostgreSQL
- orchestration
- optimizer
- SQL trigger and OF manager
evaluation

profile end to end delay (normalized per-rule, 30 rounds) for route insertion and deletion

Rocketfuel ISP topology

<table>
<thead>
<tr>
<th>AS#</th>
<th>nodes</th>
<th>links</th>
</tr>
</thead>
<tbody>
<tr>
<td>4755</td>
<td>142</td>
<td>258</td>
</tr>
<tr>
<td>3356</td>
<td>1772</td>
<td>13640</td>
</tr>
<tr>
<td>7018</td>
<td>25382</td>
<td>11292</td>
</tr>
</tbody>
</table>

- compute path
- lookup ports
- write to table
- trigger/rule
evaluation

profile end to end delay (normalized per-rule, 30 rounds) for route insertion and deletion
evaluation

profile end to end delay (normalized per-rule, 30 rounds) for route insertion and deletion

Rocketfuel ISP topology

<table>
<thead>
<tr>
<th>AS#</th>
<th>nodes</th>
<th>links</th>
</tr>
</thead>
<tbody>
<tr>
<td>4755</td>
<td>142</td>
<td>258</td>
</tr>
<tr>
<td>3356</td>
<td>1772</td>
<td>13640</td>
</tr>
<tr>
<td>7018</td>
<td>25382</td>
<td>11292</td>
</tr>
</tbody>
</table>

AS 4755 | AS 3356 | AS 7018

<table>
<thead>
<tr>
<th>deletion (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>insertion (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

compute path
lookup ports
write to table
trigger/rule
evaluation

profile end to end delay (normalized per-rule, 30 rounds) for route insertion and deletion

Rocketfuel ISP topology

<table>
<thead>
<tr>
<th>AS#</th>
<th>nodes</th>
<th>links</th>
</tr>
</thead>
<tbody>
<tr>
<td>4755</td>
<td>142</td>
<td>258</td>
</tr>
<tr>
<td>3356</td>
<td>1772</td>
<td>13640</td>
</tr>
<tr>
<td>7018</td>
<td>25382</td>
<td>11292</td>
</tr>
</tbody>
</table>

compute path
lookup ports
write to table
trigger/rule
evaluation

profile end to end delay (normalized per-rule, 30 rounds) for route insertion and deletion

Rocketfuel ISP topology

<table>
<thead>
<tr>
<th>AS#</th>
<th>nodes</th>
<th>links</th>
</tr>
</thead>
<tbody>
<tr>
<td>4755</td>
<td>142</td>
<td>258</td>
</tr>
<tr>
<td>3356</td>
<td>1772</td>
<td>13640</td>
</tr>
<tr>
<td>7018</td>
<td>25382</td>
<td>11292</td>
</tr>
</tbody>
</table>

compute path
lookup ports
write to table
trigger/rule
evaluation

similar profile on fat-tree topology (fewer nodes, more links)
- total delay < 30ms for fat-tree with 5120 switches and 196608 links
evaluation

Orchestration delay (ms) normalized per-rule for 3 scenarios:
- access control and routing (acl+rt)
- load balancing and routing (lb+rt)
- access control, load balancing, and routing (acl+lb+rt)
evaluation

Figure 3: Sources of Ravel delay (ms) for route insertion and deletion.

Figure 4: CDF of orchestration delay: normalized per-rule for 3 scenarios: access control and routing (acl+rt), load balancing and routing (lb+rt), access control, load balancing, and routing (acl+lb+rt).

CDF of querying (ms) on a view and its materialized tables that offers faster access with equivalent materialized tables that offers faster access.

Figure 5: (a) CDF of querying (ms) on a view and its materialized tables that offers faster access. (b) As policy size grows (from 10 to 1000), the performance gain is more obvious. Figure 5 (b,c) shows (0.1ms vs 1-2ms). As policy size grows (from 10 to 1000), the performance gain is more obvious.

Figure 5 (a) compares the performance on optimized view (blue shade) are one magnitude faster version (query delay) on a load balancer view (small overhead).

Optimizing application views by translating them into equivalent materialized tables that offers faster access with equivalent materialized tables that offers faster access.

We find that Ravel adds a small delay for orchestration, because of its semantics.

Delay is dominated by (¡1ms) since it only needs to read from its blacklist, configure the switches. In contrast, orchestration delay (ms) on various network sizes.

A prototype built on the Postgres database exhibits promising performance even for large scale networks.

Declarative networking.

7. RELATED WORK

Declarative design point that strikes a balance among its peers like Ravel, these systems use the database as a passive recipient that only executes queries. Unlike Ravel, these systems use the database as a reactive controller with user-centered database views: controlling applications and the dynamic orchestrations are moved into the database itself, while SQL offers a native means to create and adjust application-specific ad-hoc views.

We build on relational database research, making novel use of SQL views and contributing new data mediation techniques, with target usage — mediating applications with higher-level user support in a centralized setting — better infrastructure. This allows rapid implementation and deployment of new distributed protocols, making it an alternative to large network size.

8. CONCLUSION

The use of declarative networking [11, 17, 16, 19] — a combined effort of deduc-}

Recursive query engine as an extensible and efficient route optimizer (recursive datalog) and distributed system (distributed query optimization) research — uses a distributed database (recursive datalog) and distributed system (distributed query optimization).
evaluation

orchestration delay (ms) normalized per-rule for 3 scenarios: access control and routing (acl+rt), load balancing and routing (lb+rt), access control, load balancing, and routing (acl+lb+rt)

orchestration also scales gracefully on fat-tree
- < 30ms for fat-tree with 5120 switches and 196608 links
[ravel@ravelvm ravel]$
towards a secure Ravel

improper modification of data
- unauthorized modification
- one-directional information flow
towards a secure Ravel

expectation of data quality
improper medication of data
- unauthorized modification — access control (ACL)
- one-directional information flow
ACL in Ravel

- notification
- operation via SQL interface
- view
- view
- view
- view
- view
- view
- orchestration
- optimizer
- table
- table
- table
- SQL trigger
- OpenFlow manager
- network
- view maintenance
- view update
- view
- PostgreSQL
- Ravel runtime
ACL in Ravel

notification

operation via SQL interface

view
view
view
view
view

view maintenance

view update

network

control events

SQL trigger

OpenFlow manager

PostgreSQL

Ravel runtime

view
view
view

orchestration

optimizer

table
table
table
ACL in Ravel
example scenario

a SDN network and multiple tenants

- admin can see/modify all resources, see/modify the network
- tenants can only see the resources they pay
- tenants can only manage their portions of network under contract

SLA (service level agreement)

<table>
<thead>
<tr>
<th>tenant</th>
<th>switches</th>
<th>rate limit</th>
<th>connectivity</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice</td>
<td>{1,2,3,4}</td>
<td>20</td>
<td>{alice}</td>
</tr>
<tr>
<td>bob</td>
<td>{51,52,53,…}</td>
<td>50</td>
<td>{bob, alice}</td>
</tr>
<tr>
<td>carol</td>
<td>{100,101,…}</td>
<td>10</td>
<td>{carol, alice}</td>
</tr>
</tbody>
</table>
### explicit access control list (ACL)

**<principal, subject, operation>**

#### ACL on topology

<table>
<thead>
<tr>
<th>users</th>
<th>switches</th>
<th>privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice</td>
<td>1</td>
<td>read</td>
</tr>
<tr>
<td>alice</td>
<td>2</td>
<td>read</td>
</tr>
<tr>
<td>alice</td>
<td>...</td>
<td>read</td>
</tr>
<tr>
<td>bob</td>
<td>...</td>
<td>read</td>
</tr>
<tr>
<td>carol</td>
<td>...</td>
<td>read</td>
</tr>
<tr>
<td>admin</td>
<td>...</td>
<td>read,write</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

#### ACL on configuration

<table>
<thead>
<tr>
<th>users</th>
<th>flows (source, destination, rate)</th>
<th>privilege</th>
</tr>
</thead>
<tbody>
<tr>
<td>alice</td>
<td>(1,2,&lt;20)</td>
<td>read,write</td>
</tr>
<tr>
<td>alice</td>
<td>(2,3,&lt;20)</td>
<td>read,write</td>
</tr>
<tr>
<td>alice</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>bob</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- very low-level
- update ACL as tenant contract evolves
ACL in Ravel

authenticate at database login

higher-level finer-grained authorization via SQL

authorization views

alice bob carol

Ravel runtime

PostgreSQL

SQL trigger

OpenFlow manager

network

view view view

view view

orchestration optimizer

table table table

ACL in Ravel
ACL in Ravel
authorization views: a strawman solution

associate each table with an ACL
  - <principal, allowed operation>

create a separate view
  - if only a portion of a table is granted to a principal
  - benefit: dynamic, content-based
authorization views: a strawman solution

-- admin policy
GRANT SELECT, UPDATE, INSERT, DELETE ON topology TO admin;
GRANT SELECT, UPDATE, INSERT, DELETE ON configuration TO admin;

-- alice policy
CREATE OR REPLACE VIEW topology_alice AS (
    SELECT sid, nid FROM topology
    WHERE (topology.sid = 1 OR topology.sid = 2 OR ...);

CREATE OR REPLACE VIEW configuration_alice AS (
    SELECT fid, sid, nid FROM configuration
    WHERE ((topology.sid = 1 AND topology.nid = 2) OR
        (topology.sid = 1 AND topology.nid = 2) OR ...) AND
    rate < 20);

GRANT SELECT ON topology_alice TO alice;
GRANT SELECT, INSERT, DELETE, UPDATE ON configuration_alice TO alice;

-- bob policy, carol policy ...
limitations

many tenants
- for each tenant, create a separate view?

dynamic tenant membership
- add/remove views?

SLAs evolving
- update tenant views?

more examples:
- tenants can only access the resources they pay
- raise tenant rate limit to 100
finer-grained, higher-level ACL

capture the intent rather than extent
dynamic, context-based

SQL query over data in p and other parts of the network database

a network table of arity n
p(_,_,...,_)

access control view of n+1 arity
p_acl (principal, _, _, ..., _)
finer-grained, higher-level ACL

- a tenant can only access the leased network topology
- admin can access the whole topology

```
CREATE VIEW topology_acl AS
    ( -- admin policy
        (SELECT 'admin' as principal, sid, nid
         FROM topology)
    UNION
    -- tenant policy
        (SELECT tenant as principal, sid, nid
         FROM topology, SLA
         WHERE topology.sid IN SLA.switches
         AND topology.nid IN SLA.switches)));
```

```
CREATE VIEW topology_public AS
    (SELECT sid, nid FROM topology_acl
     WHERE principal = 'current_user')

GRANT SELECT ON topology_public TO public;
```
looking forward

data integration as a networking service

integrator (orchestration, optimization)

coherent network states processing traffic
looking forward

data integration as a networking service

conflict (cyclic update dependency) by formal analysis

coherent network states processing traffic
looking forward

data integration as a networking service

cross-layer independent failure by multiple-view constraint

coherent network states processing traffic
looking forward

data integration as a networking service

automatic security enforcement by query rewrite

coherent network states processing traffic
conclusion

this talk: via SQL
- orchestratable abstraction
- finer-grained access control

looking forward
- data integration as a networking service
playtime

download Ravel
  ravel-net.org/download

start playing: tutorials, add your own app
  ravel-net.org

explore more
  github.com/ravel-net