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
abstractions

what is the right abstraction?

functions

service chain

stateful middlebox

Frenetic / Pyretic

[NSDI’13] [PLDI’13]

OpenFlow rules
abstractions

what is the right abstraction?

functions

graphs

PGA

stateful middlebox

OpenFlow rules

[SIGCOMM'15]
abstractions

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

- Policies
  - Graphs
    - PGA
  - Automata
    - Kinetic
  - Functions
    - Pyretic
    - OpenFlow rules
    - Network
and applications (components) interact

language-level orchestration restricted to each abstraction

composing (+) policy
→ graph +PGA graph
→ function +Pyretic function
and applications (components) interact

language-level orchestration restricted to each abstraction

composing (+) policy

how to integrate the runtime?

hard-wire internals?
and applications (components) interact

- graphs
- automata

- PGA
- Kinetic

- functions

- Pyretic

- OpenFlow rules
- network

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

structure
runtime
structure
runtime
structure
runtime
structure
runtime
```
current state of abstraction research

- structure
- runtime

enlarging body of abstractions

- new structure
- new runtime

OpenFlow rules
network
current state of abstraction research

- Enlarging body of abstractions

- Fragmented orchestration

- OpenFlow rules network
SDN control revolves around data representation:
- discard specialized, pre-compiled, fixed structures
- adopt a *plain data representation*

Our perspective...

```
operator and/or application
```

```
high-level representation
data
new data
```

```
low-level representation
data
data
data
```

```
OpenFlow rules
network
```
our perspective

SDN control revolves around data representation

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

operator and/or application

data
new data

data
new data

data
old data

data

OpenFlow rules
network
a database-defined network

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

operator and/or application

view
new view

view

view

view

openFlow rules
network

high-level
app views

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

![Diagram](image)
**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

attractive features

![Diagram showing operator and/or application, view, new view, table, table, OpenFlow rules network, and database runtime connections.]
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

![Diagram](image-url)
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>

... flow 1

h₁ ← S₁ → S₄ → h₄

... flow 2

h₂ ← S₂ → S₃ → h₃

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>

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>
abstraction: application view

firewall view: monitoring unsafe flows violating acl policy

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

```
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)
);
```

firewall control: 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: 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

SQL rule: upon broken path, re-route

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

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

<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
routing app: check broken path, re-route

SQL rule: upon broken path, re-route

Mininet link (172,39) down

shortest path view

configuration table

topology table

app view

app

network table

link down

Mininet

orchestration across representations
routing app: check broken path, re-route

SQL rule: upon broken path, re-route

Mininet link (172,39) down

app

app view

shortest path view

network table

topology table

configuration table

Mininet

shortest path

path

{...,172,39,156,...}

{...,172,38,148,...}

topology

<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>

configuration
orchestration across representations

Routing app: check broken path, re-route

Mininet link (172,39) down

SQL rule:
Upon broken path, re-route

**shortest path**

<table>
<thead>
<tr>
<th>path</th>
</tr>
</thead>
<tbody>
<tr>
<td>{...,172,39,156,...}</td>
</tr>
<tr>
<td>{...,172,38,148,...}</td>
</tr>
</tbody>
</table>

**configuration**

<table>
<thead>
<tr>
<th>fid</th>
<th>sid</th>
<th>nid</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>172</td>
<td>39</td>
</tr>
<tr>
<td>+</td>
<td>172</td>
<td>38</td>
</tr>
<tr>
<td>-</td>
<td>39</td>
<td>156</td>
</tr>
<tr>
<td>+</td>
<td>38</td>
<td>148</td>
</tr>
</tbody>
</table>

**topology**

<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>
<tr>
<td>172</td>
<td>39</td>
<td>1</td>
</tr>
<tr>
<td>172</td>
<td>38</td>
<td>1</td>
</tr>
</tbody>
</table>

Mininet (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

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

priority: low → high

balance | firewall | maintain

load balancer | access control | shortest path

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

tenant virtual net

reachability matrix
configuration table

Mininet
orchestration across applications

priority: low → high

load balancer

access control

shortest path

load balancer

sid | load
--- | ---
+ 1003 | 4
- 1003 | 3

access control

tenant virtual net

... | host | server
--- | --- | ---
... | 1238 | 1003

reachability matrix

classifier firewall

network state table

Mininet

network load balancer

app view

app

balance firewall maintain path

load

load balancer

access control

shortest path

tenant request

tenant virtual net

host 1238 to server 1003

reachability matrix

configuration

10
orchestration across applications

priority: low → high

load balancer

access control

shortest path

access control

tenant virtual net

reachability matrix

configuration table

network table

app view

app

tenant request

tenant request host 1238 to server 1003

reachability matrix

Mininet
orchestration across applications

priority: low → high

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

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

shortest path:

+ ... path
+ ... \{1238, ..., 1034\}

reachability matrix:

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

configuration:

tenant virtual net:

<table>
<thead>
<tr>
<th>host</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1238</td>
<td>1003</td>
</tr>
<tr>
<td>1238</td>
<td>1034</td>
</tr>
</tbody>
</table>

network table:

Mininet

app view:

load balancer
access control
shortest path
orchestration across applications

priority: low \(\rightarrow\) 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

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

tenant virtual net

<table>
<thead>
<tr>
<th>host</th>
<th>server</th>
</tr>
</thead>
<tbody>
<tr>
<td>1238</td>
<td>1003</td>
</tr>
<tr>
<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>1238</td>
<td>1034</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></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

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

... path

... \{1238, ..., 1034\}

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>
<tr>
<td>+</td>
<td>...</td>
<td>...</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>
<tr>
<td>+</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
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
runtime

notification

operation via SQL interface

view view view

view view

orchestration optimizer

table table table

network

- 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
We present a novel SDN design, Ravel, based on a standard SQL database. This allows rapid prototyping and deployment of new distributed protocols, making extensible and efficient routing infrastructure. This allows rapid prototyping and deployment of new distributed protocols, making extensible and efficient routing infrastructure. The database runtime, enhanced with view mechanisms and a data mediating protocol, allows multiple disparate applications — collaborative or competitive — to collectively drive the network in a user-defined meaningful way. A prototype built on the PostgreSQL database exhibits equivalent materialized tables that offer faster access with small overhead.

**Evaluation**

![Bar chart showing end-to-end delay for route insertion and deletion across different AS topologies.](chart.png)

**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>

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

- Compute path
- Lookup ports
- Write to table
- Trigger/rule
We present a novel SDN design, Ravel, based on a standard SQL database. Ravel optimizes application views by translating them into equivalent materialized tables that offer faster access with small overhead. (b,c) CDF of maintenance delay (ms).

Figure 5 (a) compares the performance (query delay) on a view and its materialized equivalent. (b,c) CDF of maintenance delay (ms).

In the pre-SDN era, declarative networking targeted usage — mediating applications with higher-level user supervision. Declarative networking — to collectively drive the network in a user-defined meaningful way. A prototype built on the PostgreSQL database exhibits consistency, network-wide views over distributed network elements, and multiple controller instance. Unlike Ravel, these systems use SDN programming APIs.

With the simple and familiar SQL query, constraints, and declarative languages, the database runtime, database. With the simple and familiar SQL query, constraints, and declarative — to collectively drive the network in a user-defined meaningful way. A prototype built on the PostgreSQL database exhibits consistency, network-wide views over distributed network elements, and multiple controller instance. Unlike Ravel, these systems use SDN programming APIs.

ACM, pp. 338–347.

Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems

Proceedings of the Twenty-fifth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems
evaluation

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

<table>
<thead>
<tr>
<th>Rocketfuel ISP topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS#</td>
</tr>
<tr>
<td>4755</td>
</tr>
<tr>
<td>3356</td>
</tr>
<tr>
<td>7018</td>
</tr>
</tbody>
</table>

compute path
lookup ports
write to table
trigger/rule
We present a novel SDN design, Ravel, based on a standard SQL database. Ravel optimizes application views by translating them into equivalent — to collectively drive the network in a user-defined meaningful way. A prototype built on the PostgreSQL database exhibits native means to create and adjust application-specific ad-hoc views. In Ravel, the database is almost identical to that of SQL views and contributing new data mediation techniques, with enhanced with view mechanisms and a data mediating protocol, allowing multiple disparate applications — collaborative or competitive — to use a database as a passive recipient that only executes queries and transactions. Furthermore, the network-wide views are often predefined by the system (e.g., Onix’s NIB APIs with fixed schemas for triggers, non-experts can rapidly launch, modify, and switch between applications views. In Ravel, the database centered database views: control applications and the dynamic orchestration delay (ms) of SQL views and contributing new data mediation techniques, with consistent network-wide views over distributed network elements are an order of magnitude faster (.1ms vs 1-2ms). As policy sizes (lb+acl+rt) are measured on three fat-tree topologies (k=16,32,64) and two scenarios: updates (deletion and insertion) to lb + acl. In the pre-SDN era, declarative networking scales well to large network size. View maintenance incurs small delay (single-digit ms) that in all cases, view maintenance incurs small delay (single-digit ms). As policy sizes (lb+acl+rt) are an order of magnitude faster (.1ms vs 1-2ms). As policy sizes (lb+acl+rt) are an order of magnitude faster (.1ms vs 1-2ms).
evaluation

<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>

Rocketfuel ISP topology

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

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

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

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 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

AS 4755  AS 3356  AS 7018

0.1  1   10  100  0.1  1   10  100  0.1  1   10  100  1000

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

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

- View
- View maintenance
- SQL trigger
- OpenFlow manager
- SQL interface
- PostgreSQL
- Ravel runtime
- Network

Notifications and operations via SQL interface.
ACL in Ravel

operation via SQL interface

SQL trigger

OpenFlow manager

network

PostgreSQL

Ravel runtime

view

view

view

view

view

orchestration

optimizer

table

table

table

notification

view maintenance

view update

control events
ACL in Ravel

alice -> ???
bob -> ???
carol -> ???

Ravel runtime

PostgreSQL

orchestration

optimizer

table

SQL trigger

OpenFlow manager

network
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>

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

<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 topology

<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>carol</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

ACL on configuration
ACL in Ravel

Ravel runtime
PostgreSQL

Authorize views

authenticate at database login

greater-level finer-grained authorization via SQL
ACL in Ravel

Ravel runtime

secure PostgreSQL

view view

view

views

authorization

orchestration optimizer

table table table

SQL trigger

OpenFlow manager

network
authorization views: a strawman solution

associate each table with an ACL
- \( <\text{principal}, \text{allowed operation}> \)

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)

cohesive, 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

app views

server  back-up

authorization views

view  view  view

view  view  view

view  view  view

cross-layer independent failure by multiple-view constraint

table  table  table  table  table  table

cohort network states processing traffic
looking forward

data integration as a networking service

app views

untrusted view

authorization views

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