Escaping Babel: The Flow Must Go On

Evolving the switch API without disrupting the network

Jonathan DiLorenzo, Waqar Mohsin, **Victor Rios**, Steffen Smolka

What is this talk about?

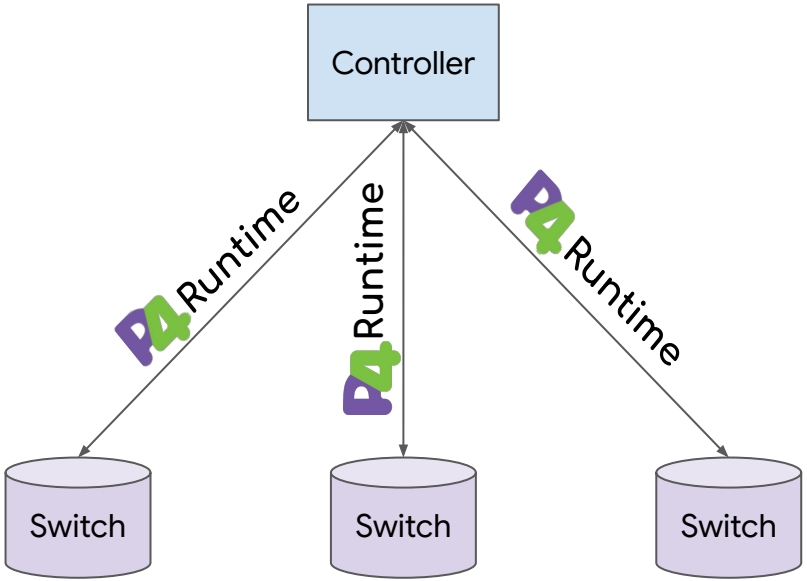
Google uses SDN

A controller programs switches through a switch API (e.g. P4Runtime).

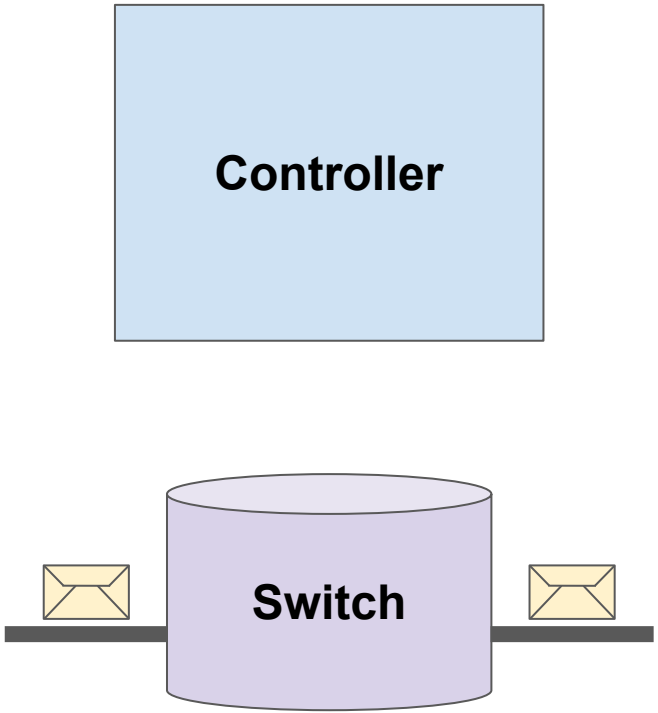
Problem: How to **rollout P4 program** and controller changes impacting the API?

- SDN = distributed system

Our Solution: Babel, a **framework for evolving the switch API** and its usage with minimal network disruption.

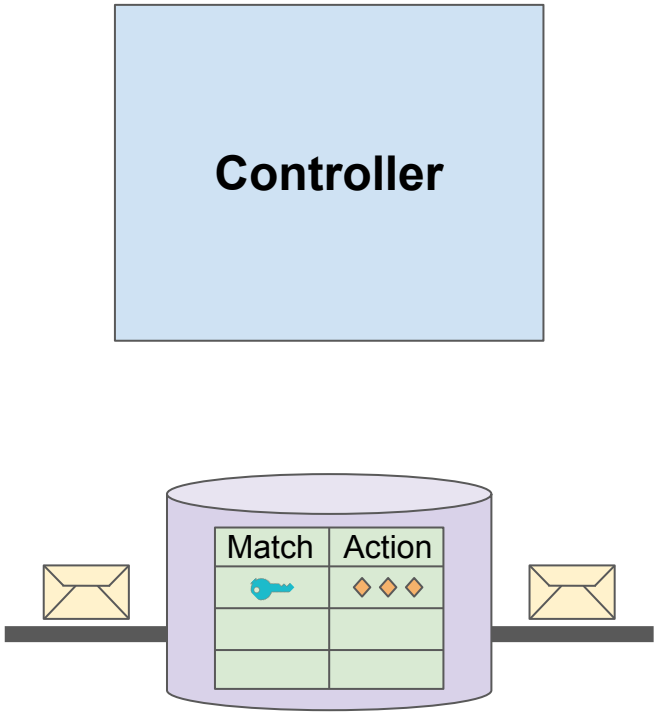


A Simple(ish) Model



Model:

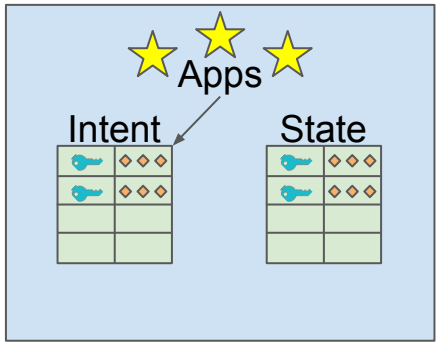
A Simple(ish) Model

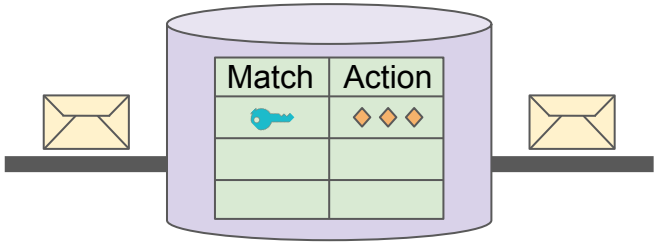


Model:

Switch = Relational Database

A Simple(ish) Model



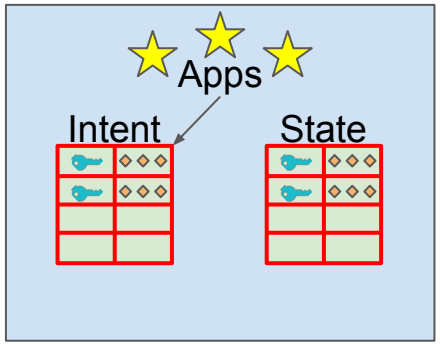


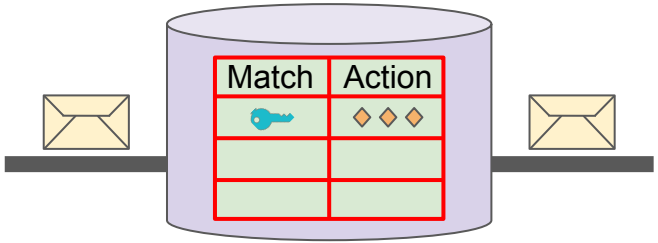
Model:

Switch = Relational Database

Controller = Apps and 2 Databases

A Simple(ish) Model





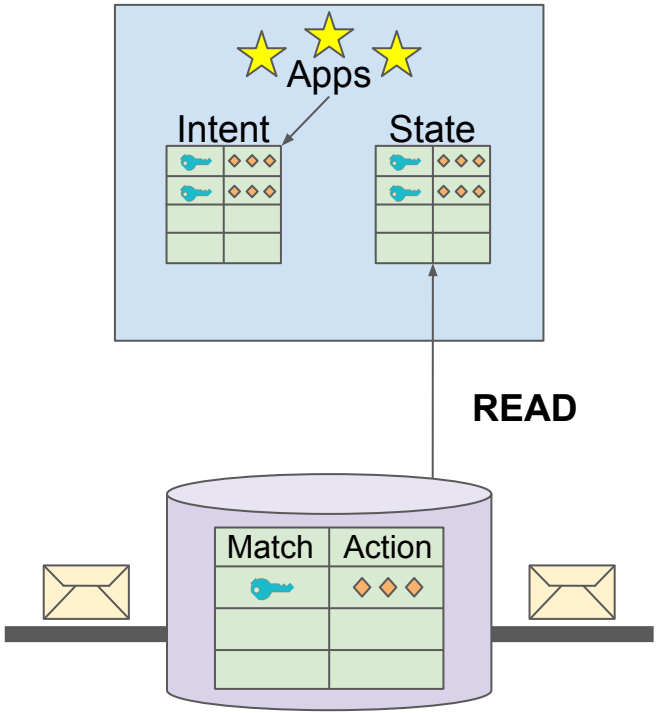
Model:

Switch = Relational Database

Controller = Apps and 2 Databases

Controller Schema = Switch Schema

A Simple(ish) Model



Model:

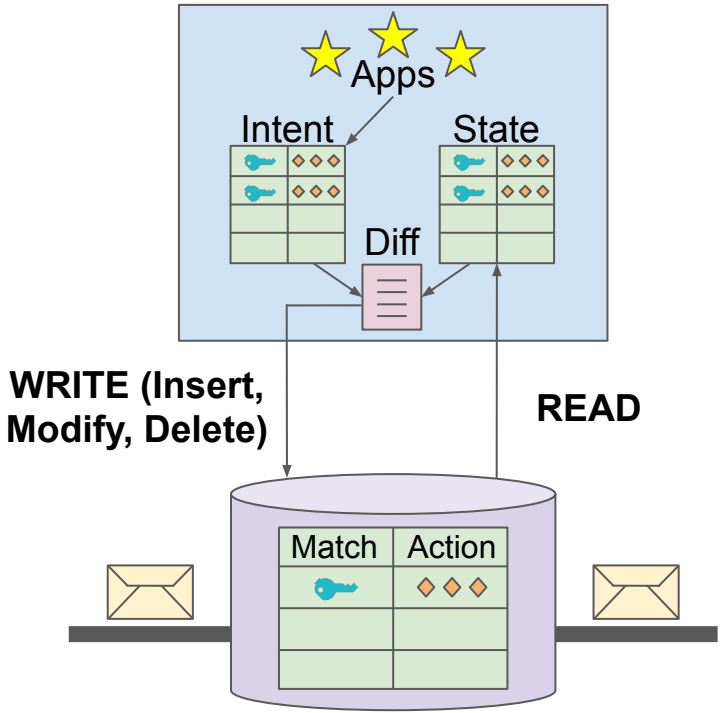
Switch = Relational Database

Controller = Apps and 2 Databases

Controller Schema = Switch Schema

Controller must READ from Switch

A Simple(ish) Model



Model:

Switch = Relational Database

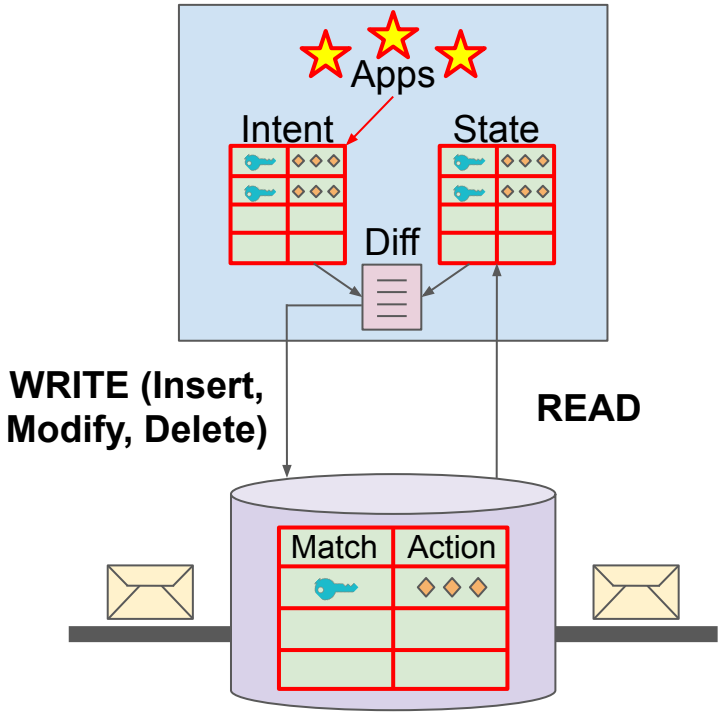
Controller = Apps and 2 Databases

Controller Schema = Switch Schema

Controller must READ from Switch

Controller must WRITE to Switch

A Simple(ish) Model



Model:

Switch = Relational Database

Controller = Apps and 2 Databases

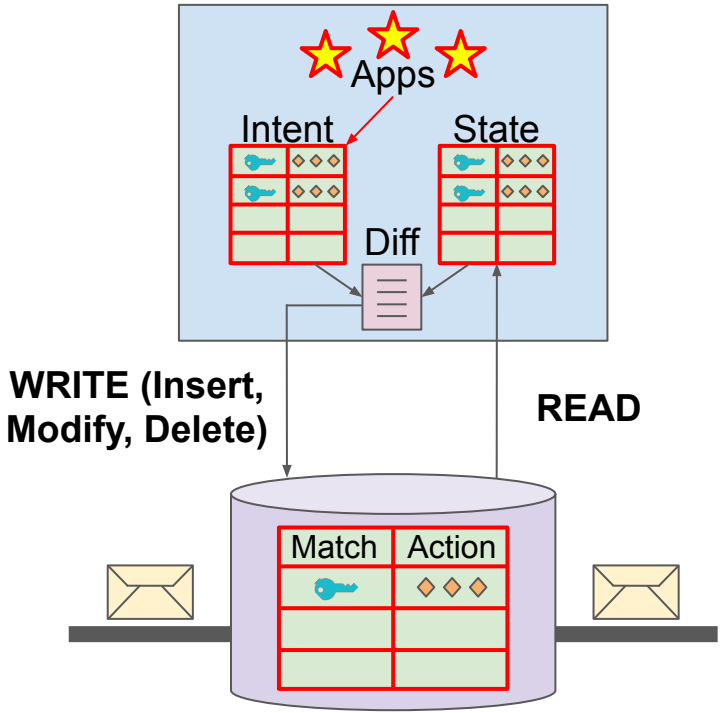
Controller Schema = Switch Schema

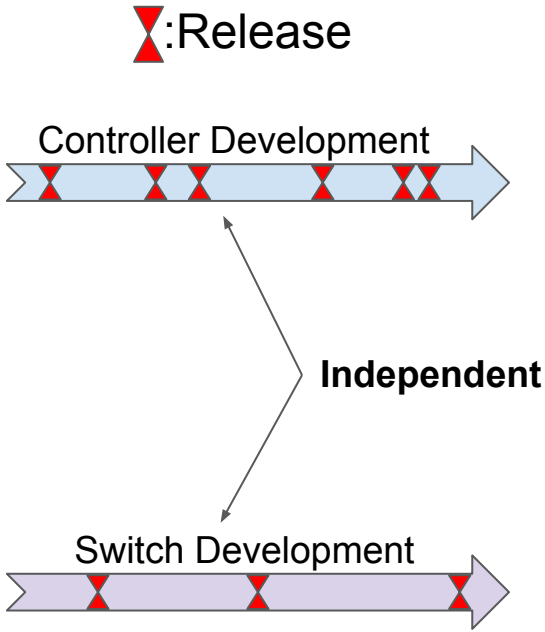
Controller must READ from Switch

Controller must WRITE to Switch

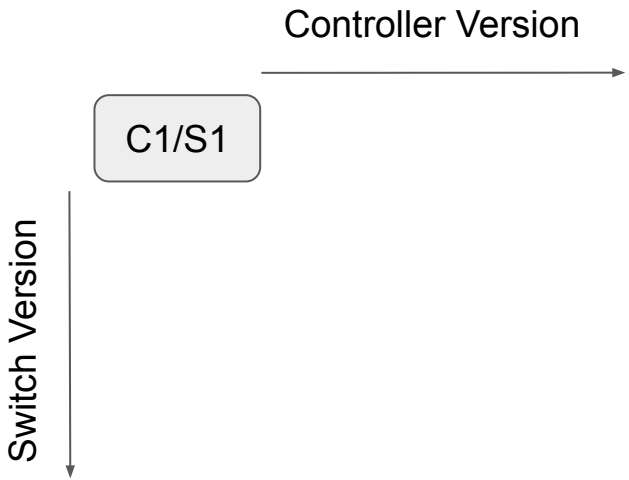
Network Evolution = Changing the schema or how it is used

How does the Network Evolve?



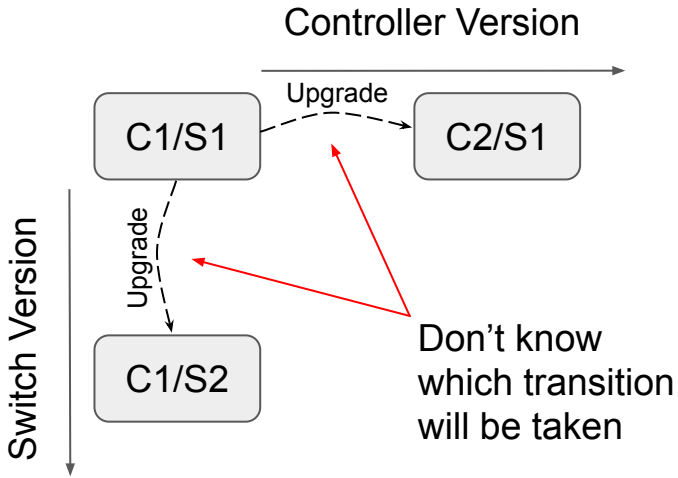


How does the Network Evolve?



Assumptions:

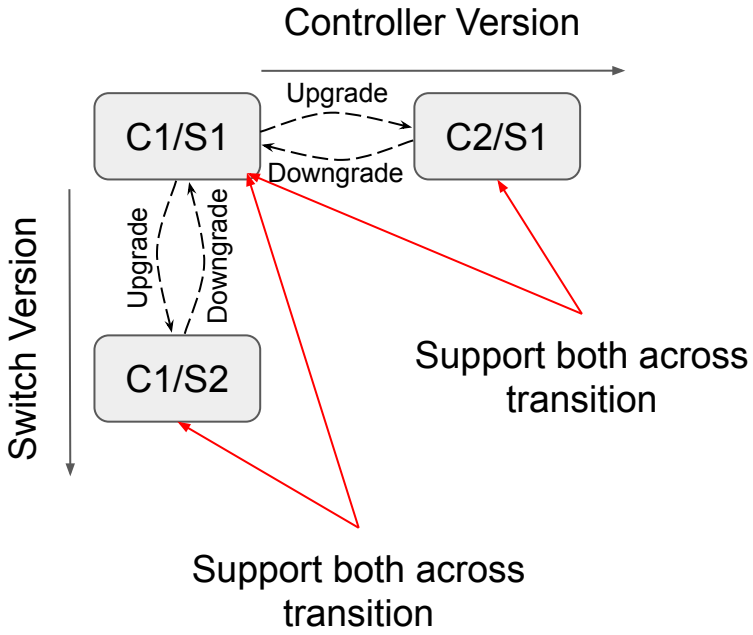
How does the Network Evolve?



Assumptions:

Order of upgrades is not fixed during development

How does the Network Evolve?

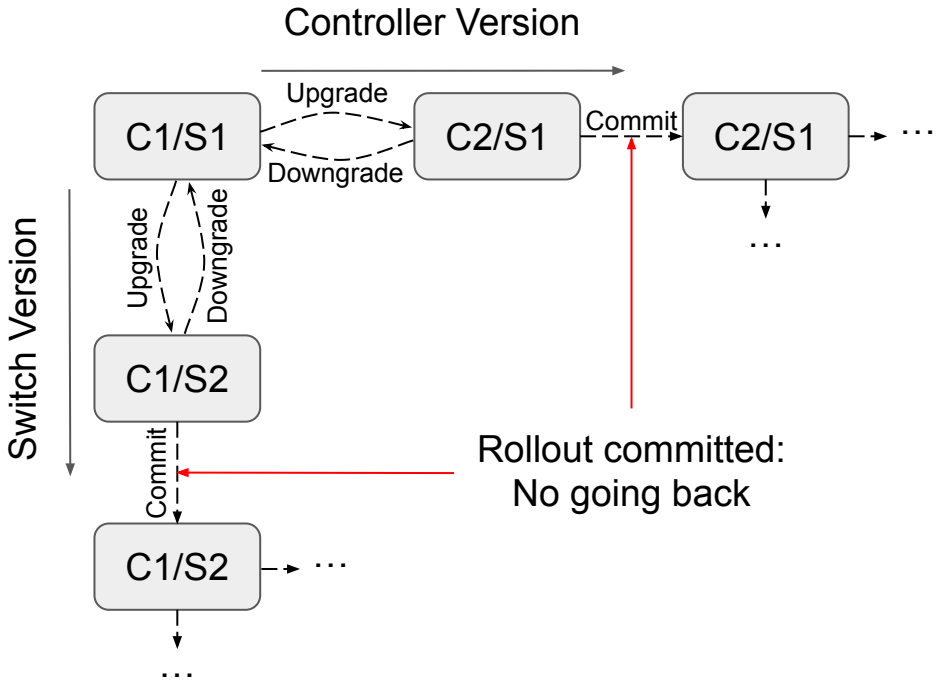


Assumptions:

Order of upgrades is not fixed beforehand

Must support downgrades

How does the Network Evolve?



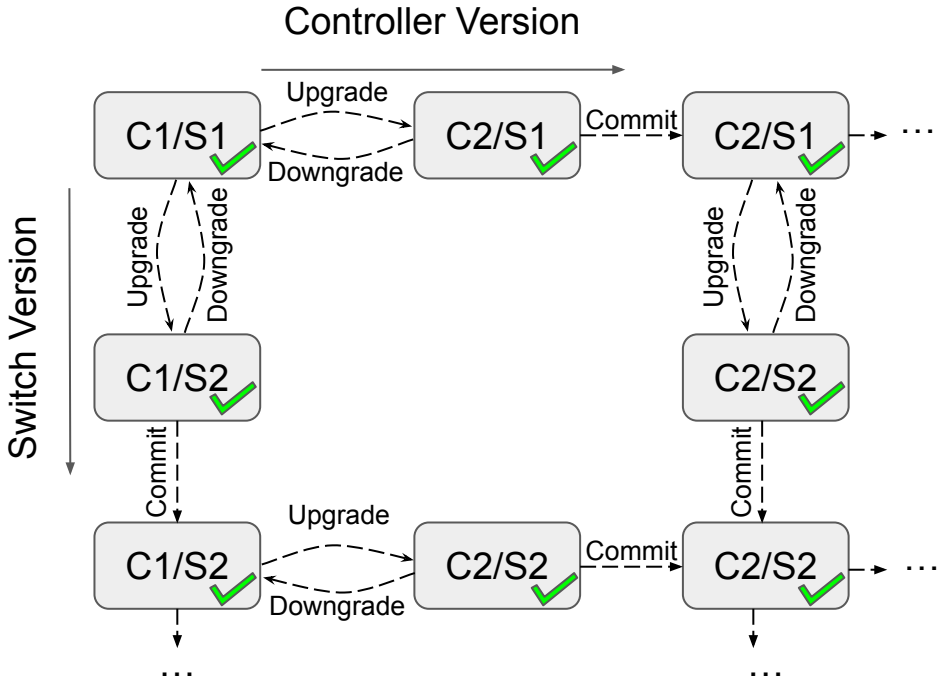
Assumptions:

Order of upgrades is not fixed during development

Must support downgrades

No double downgrades

How does the Network Evolve?



Assumptions:

Order of upgrades is not fixed during development

Must support downgrades

No double downgrades

Must maintain READ/WRITE support throughout entire version matrix

Roadmap

1. Model & Requirements ✓
2. **Example: Why is network evolution hard?**
3. Solution: Babel
4. How general is Babel?
5. How do you use Babel?

A seemingly trivial real-world example

Queue IDs are a hardware-specific detail

Queue Names are hardware-agnostic

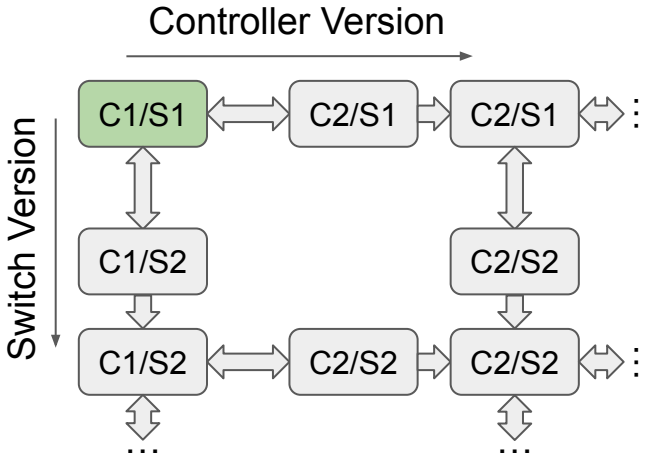
Mapping exists between IDs and names

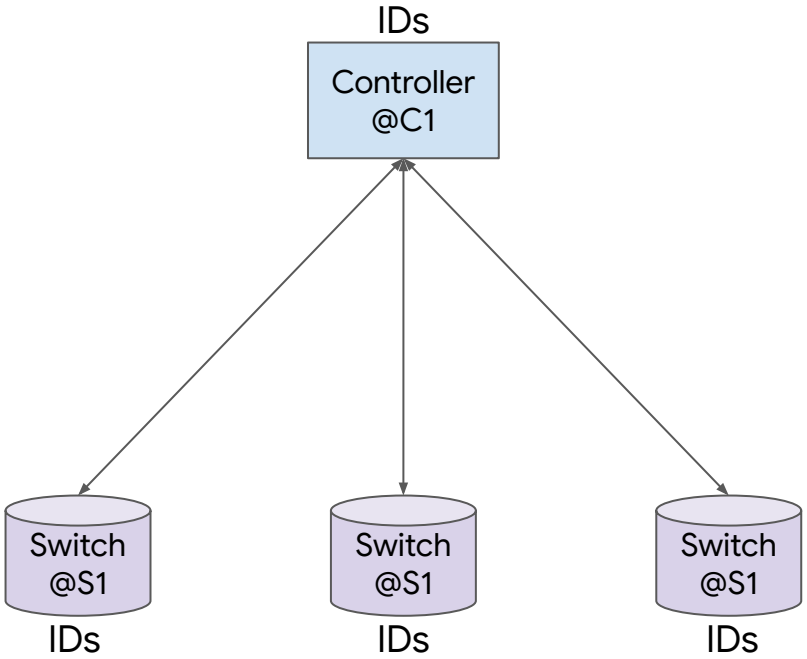
Change: Use queue names instead of IDs

Motivation: Portability, controller support for a variety of switch hardware

```
acl_ingress_table_entry {
  match {
    ether_type {
      value: "0x0806"
      mask: "0xffff"
    }
  }
  action {
    acl_trap {
-     qos_queue: "0x6"
+     qos_queue: "priority-6"
    }
  }
}
```

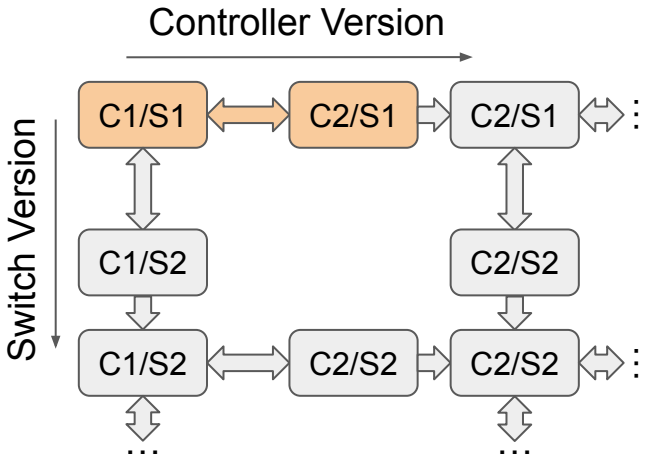
What goes wrong without Babel?

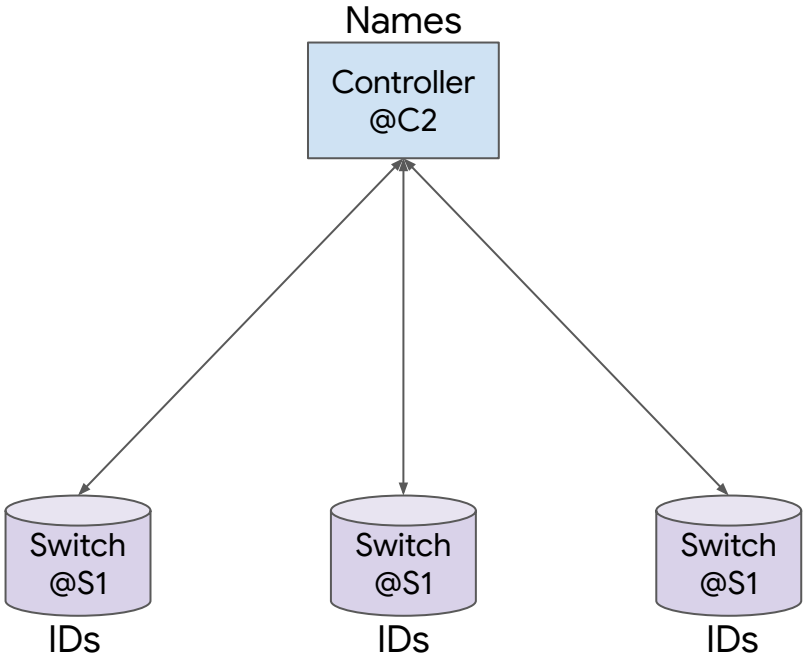




What goes wrong without Babel?

Bad Approach: Instantly upgrade the controller to use Names

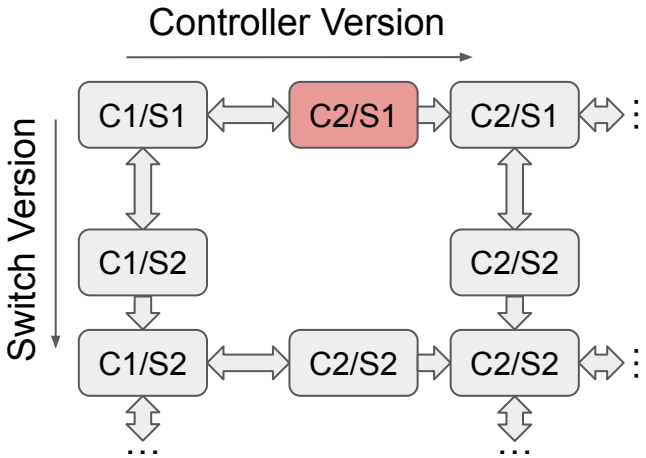


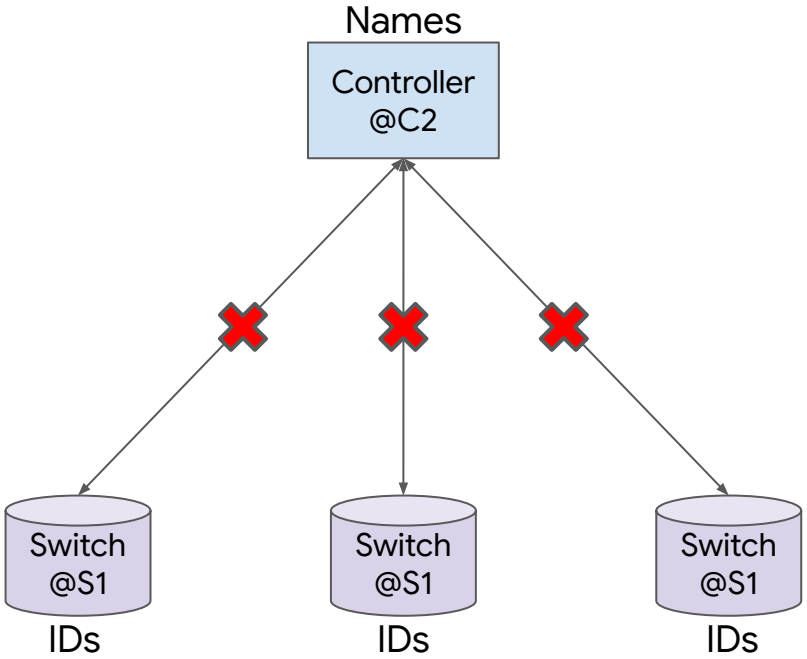


What goes wrong without Babel?

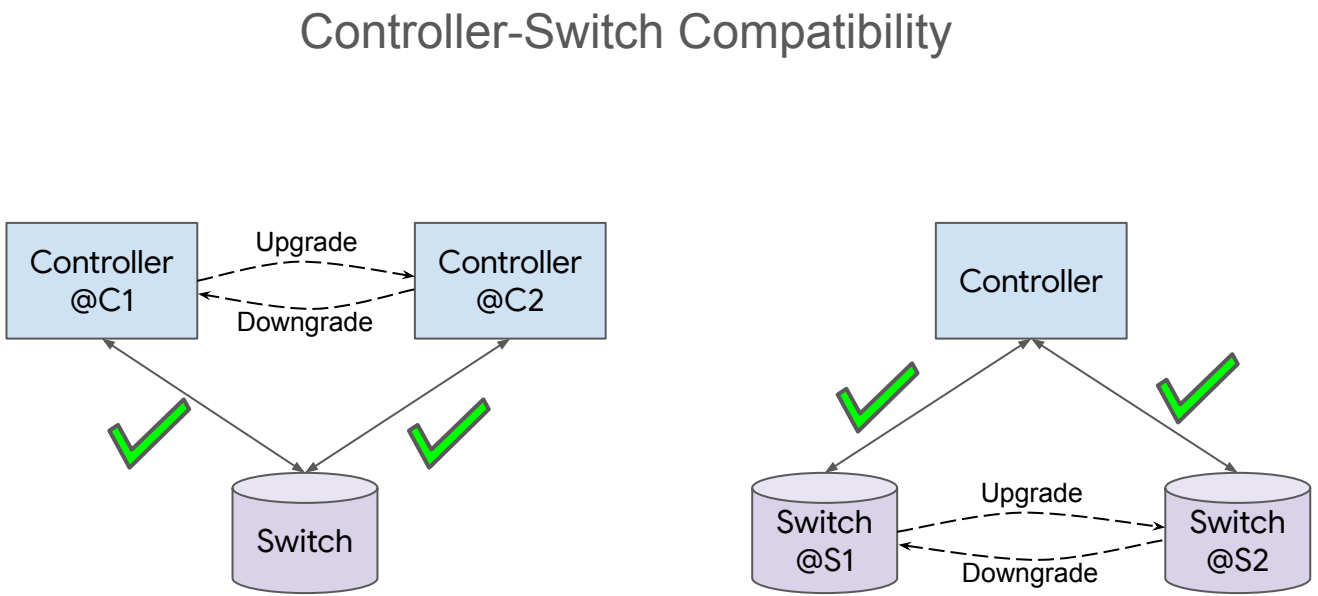
Bad Approach: Instantly upgrade the controller to use Names

Error: Read/Write not supported





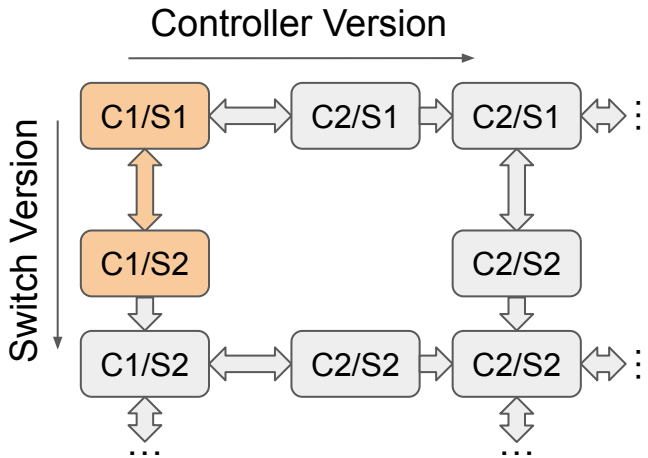
What is the root of the problem?

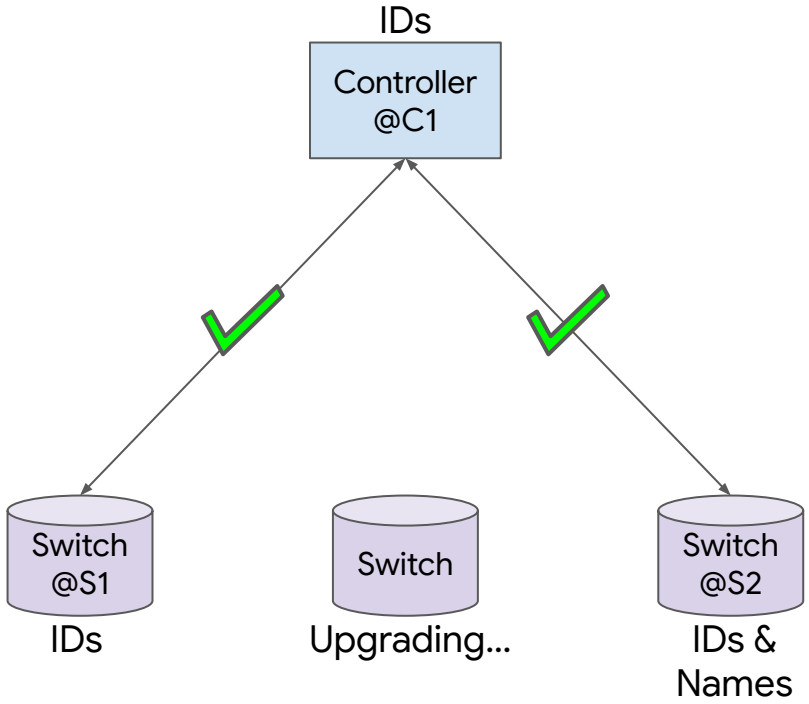


What goes wrong without Babel?

Better approach: First upgrade switch to support both IDs and Names

Non-breaking change to the schema

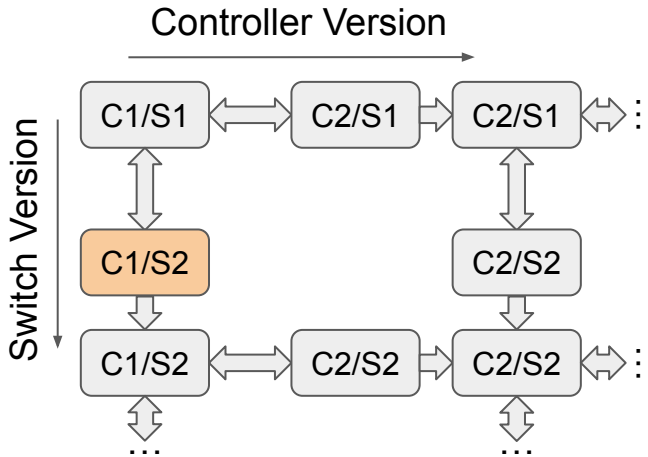


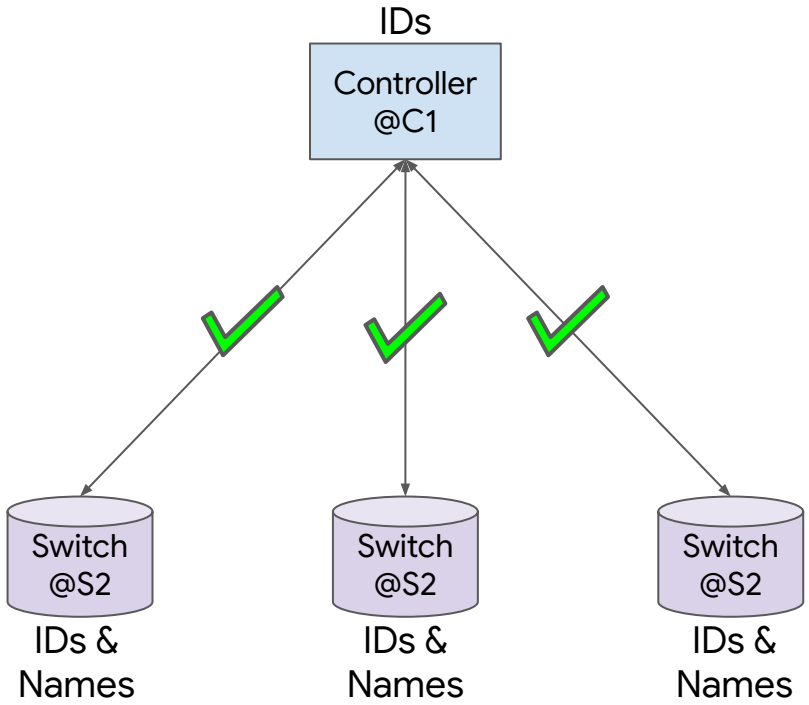


What goes wrong without Babel?

Better approach: First upgrade switch to support both IDs and Names

Non-breaking change to the schema

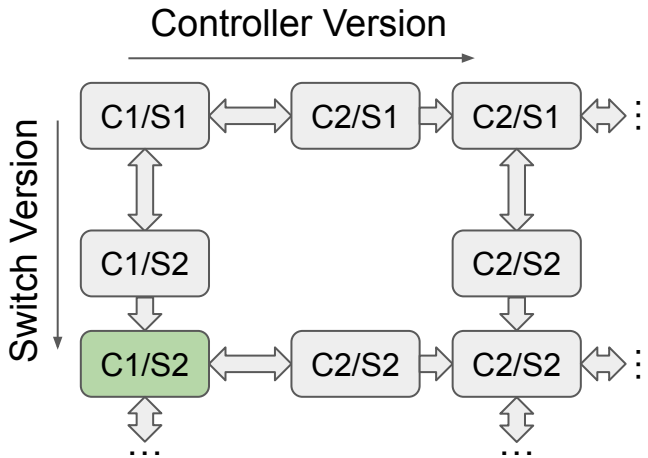


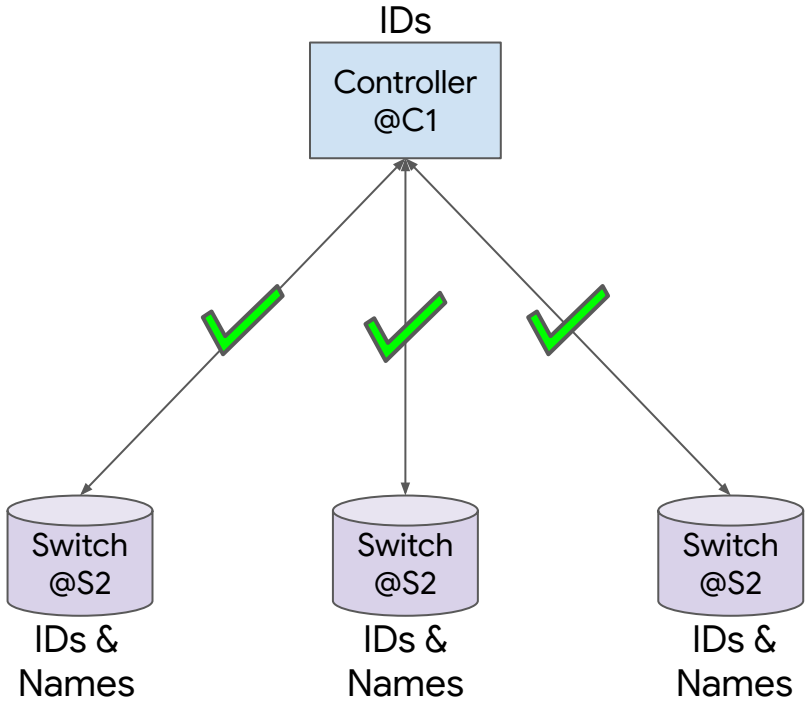


What goes wrong without Babel?

Better approach: First upgrade switch to support both IDs and Names

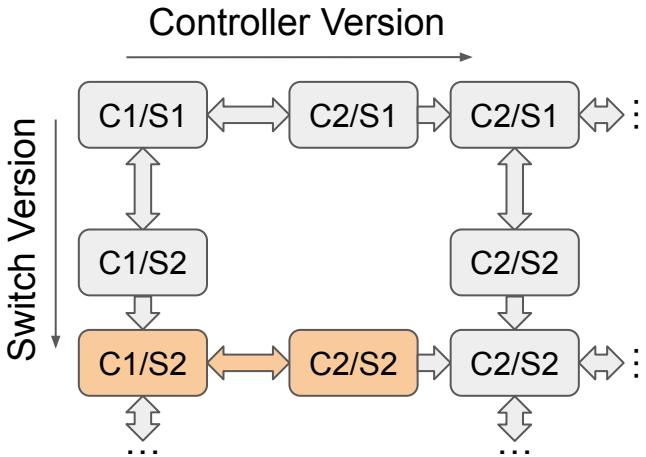
Non-breaking change to the schema

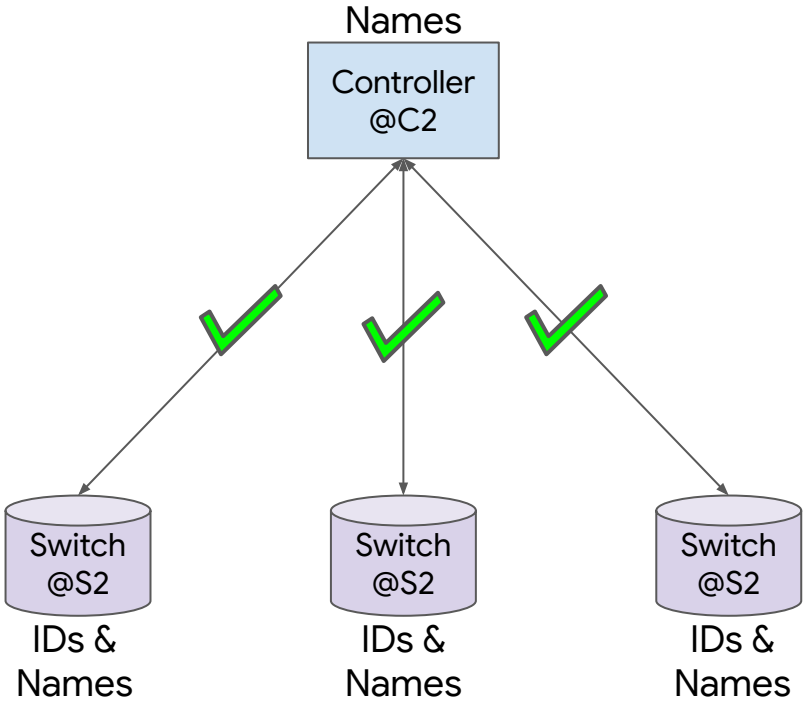




What goes wrong without Babel?

Upgrade controller to use Names

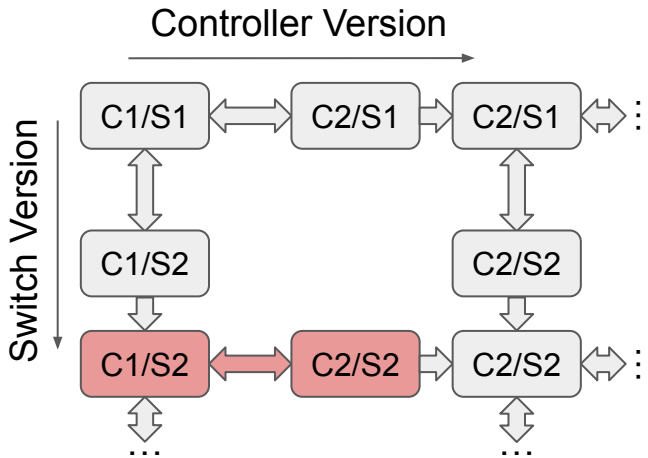


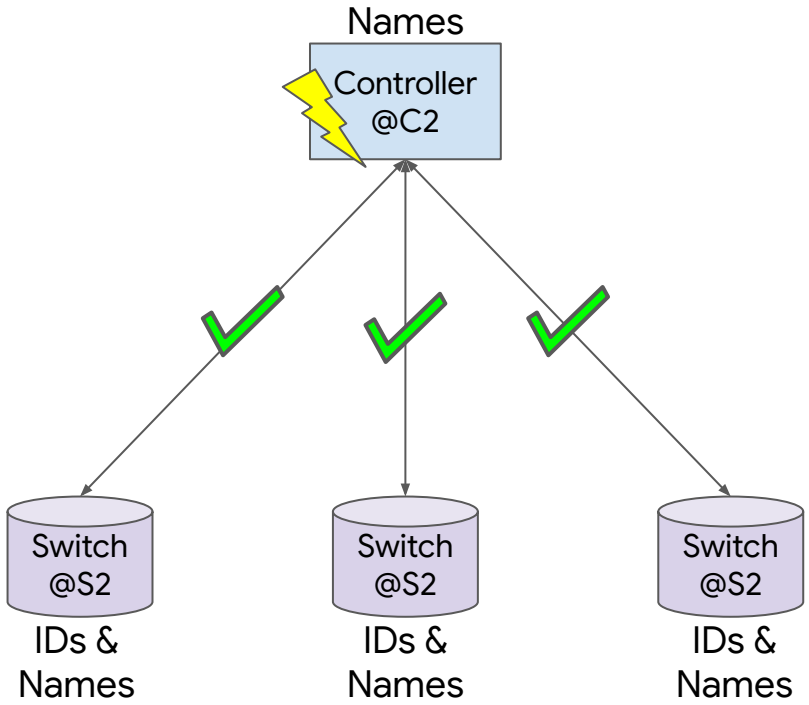


What goes wrong without Babel?

Upgrade controller to use Names

The new controller reported a fatal error… how did this happen?!

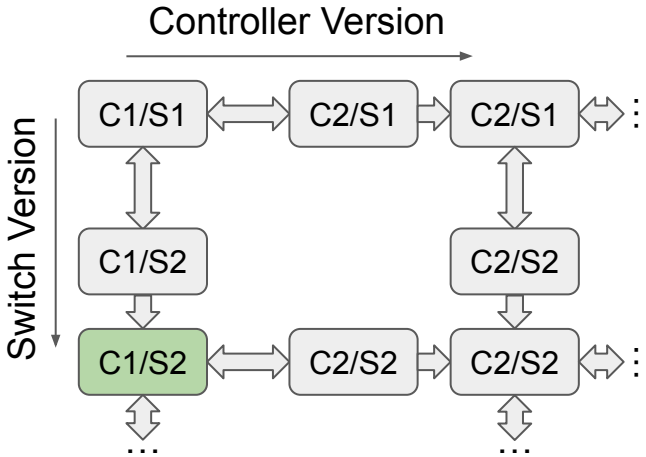


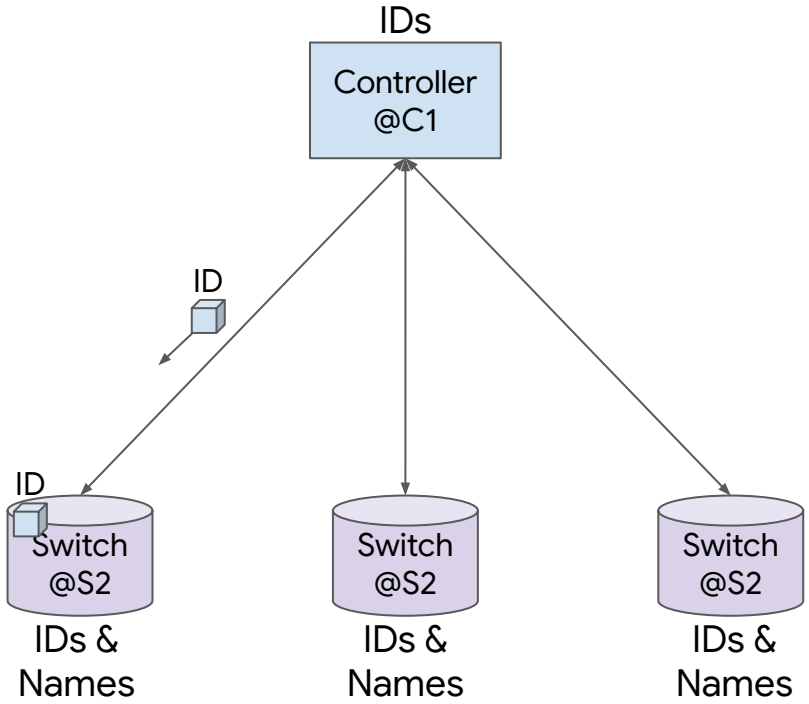


What goes wrong without Babel?

Rewind: Completed switch upgrade

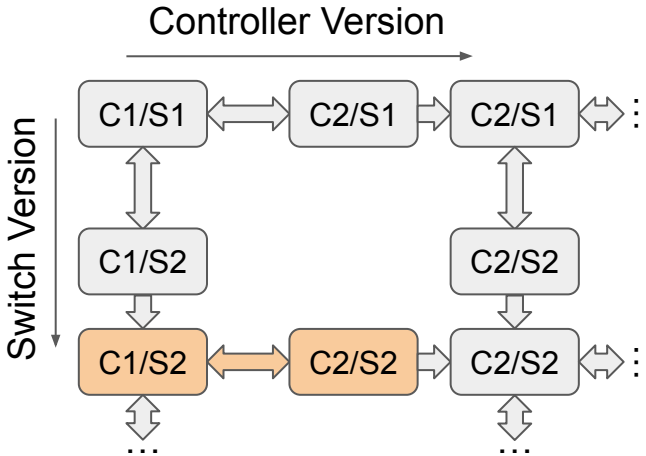
Old controller has written some entries

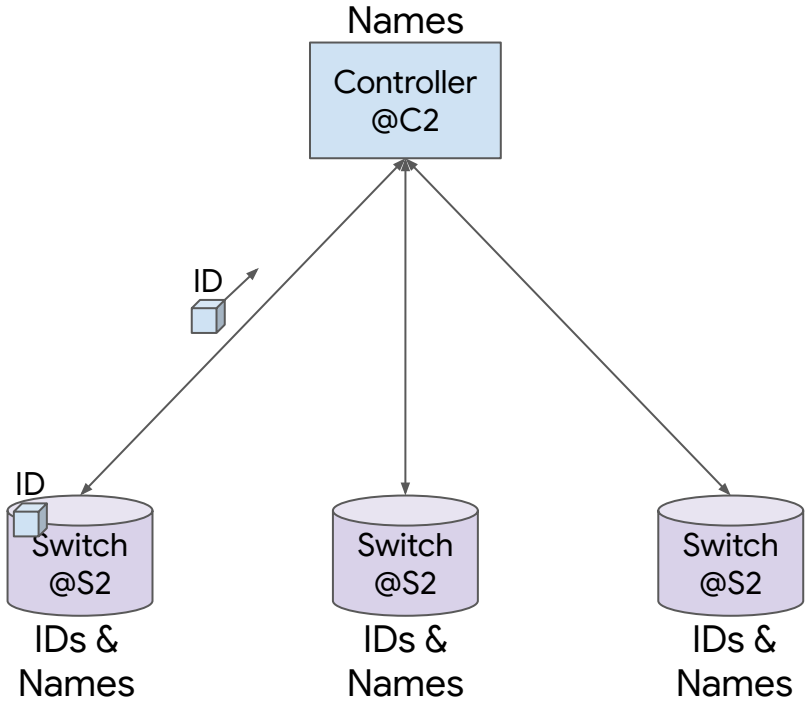




What goes wrong without Babel?

New controller boots up and reads entries on the switch and…

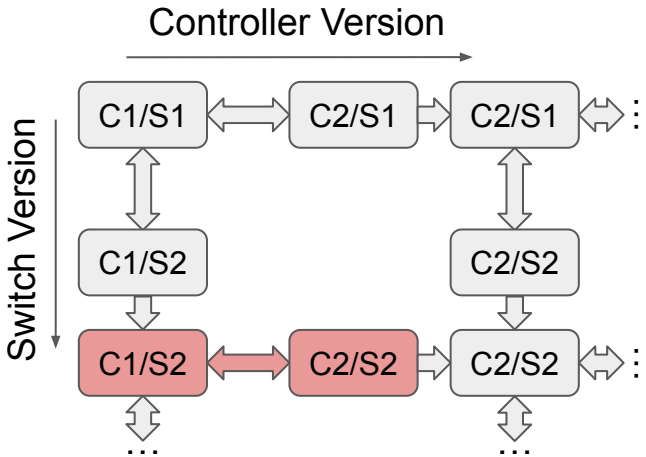


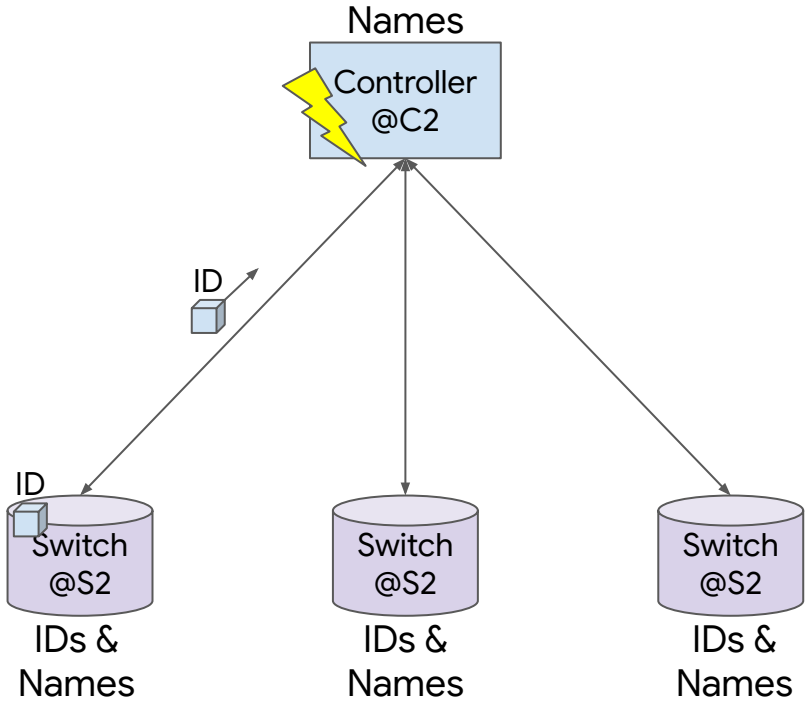


What goes wrong without Babel?

New controller boots up and reads entries on the switch and…Fatal error!

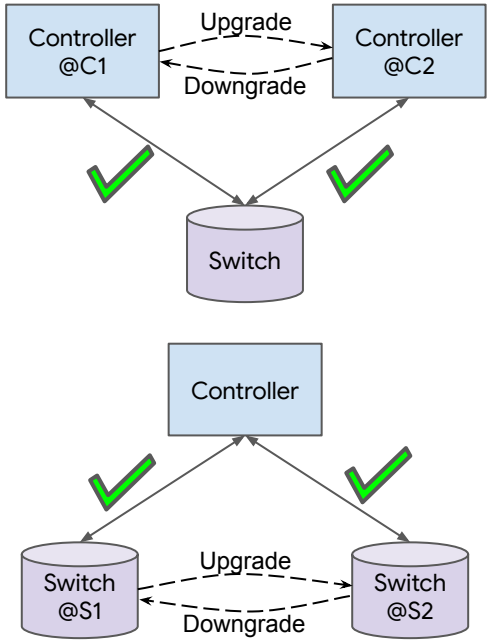
Can’t read old entries



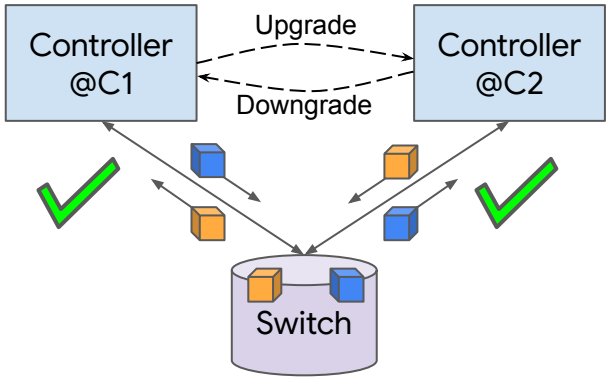


What is the root of the problem?

Controller-Switch Compatibility



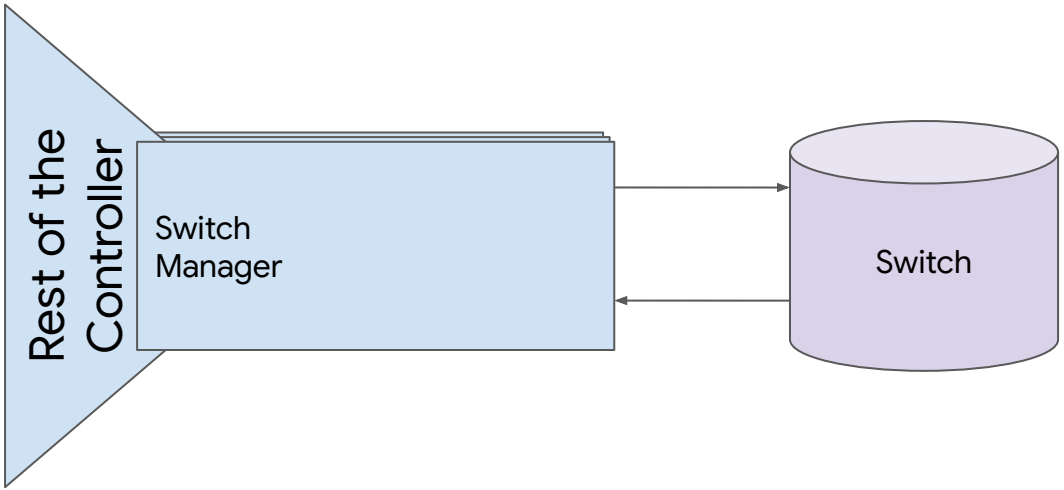
Controller-Controller Compatibility



Roadmap

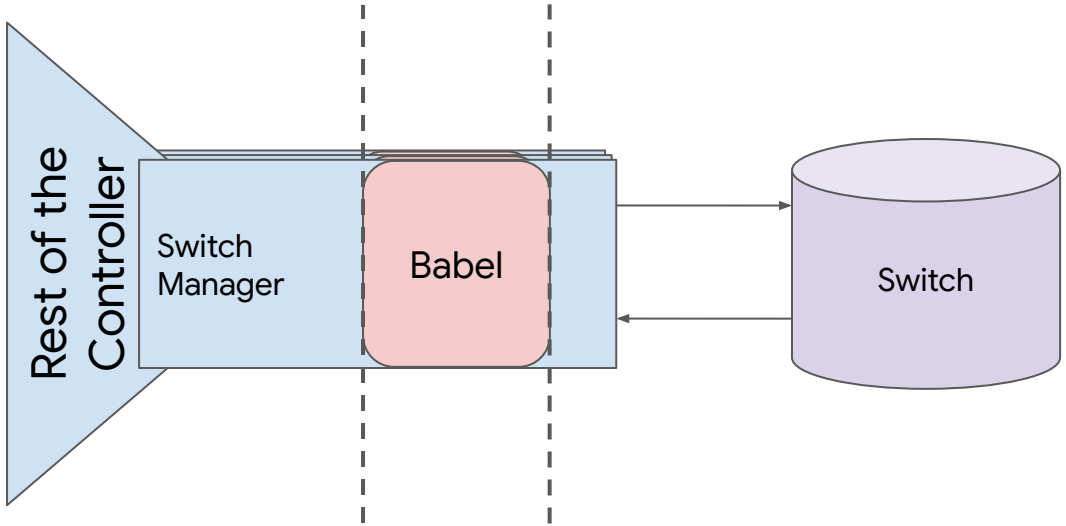
1. Model & Requirements ✓
2. Example: Why is network evolution hard? ✓
3. **Solution: Babel**
4. How general is Babel?
5. How do you use Babel?

What is Babel?



What is Babel?

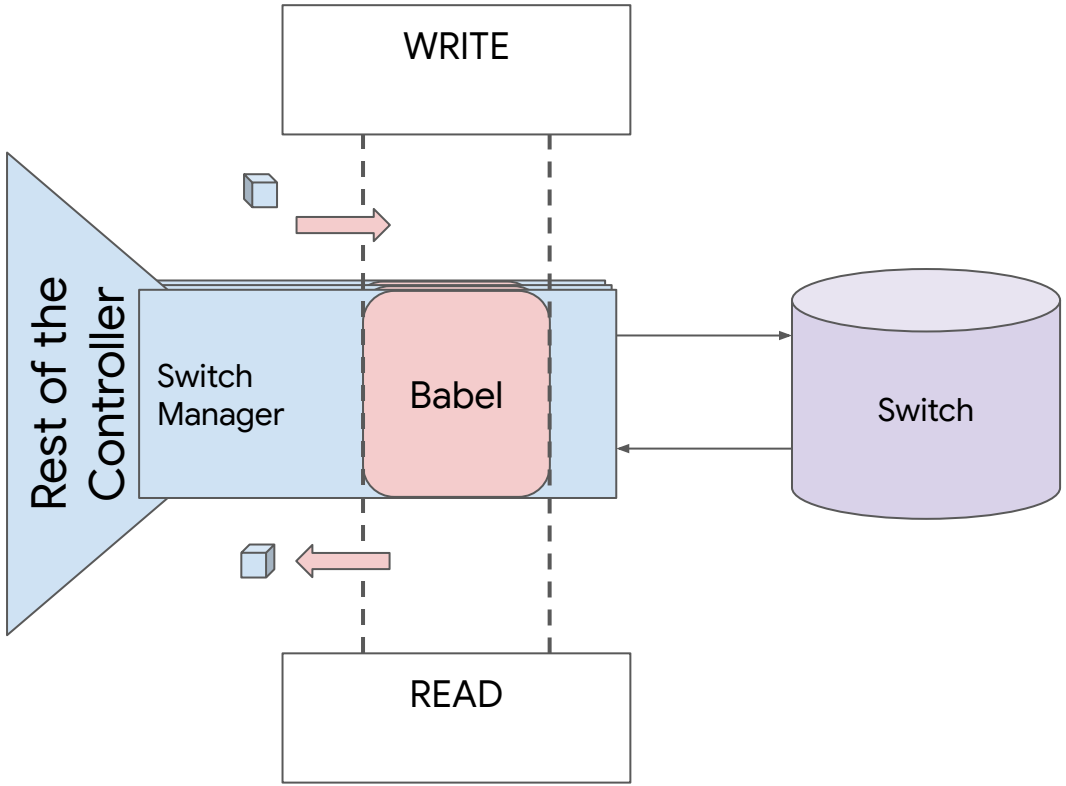
Babel: Transparent layer between the controller and switches



What is Babel?

Babel: Transparent layer between the controller and switches

Provides controller illusion of a **fixed schema version**

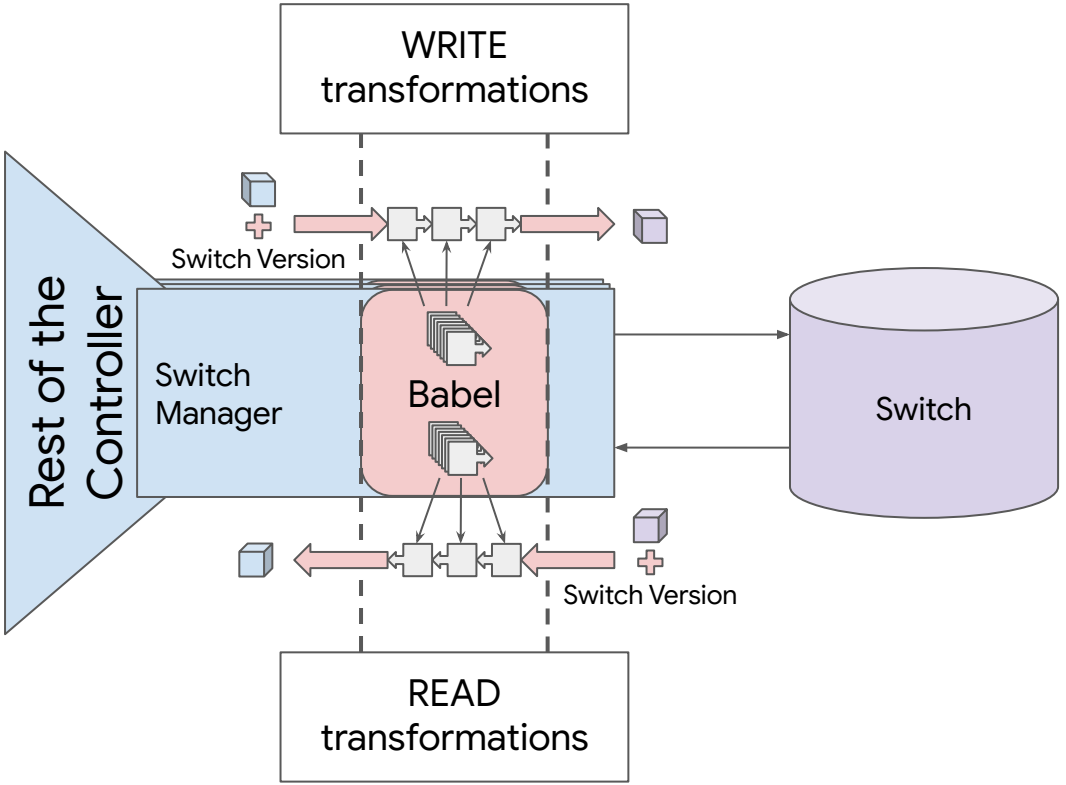


What is Babel?

Babel: Transparent layer between the controller and switches

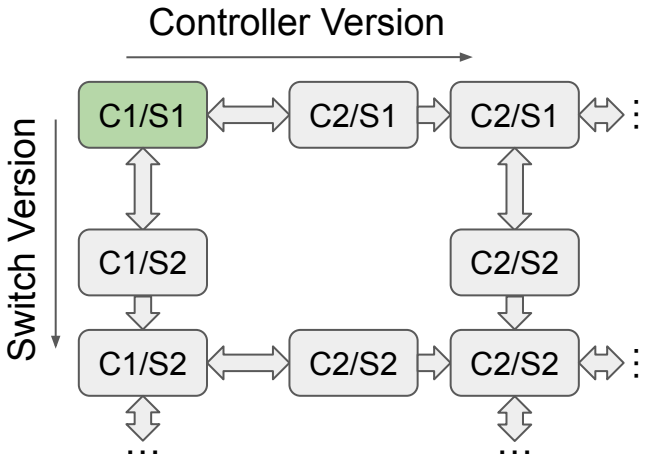
Provides controller illusion of a **fixed schema version**

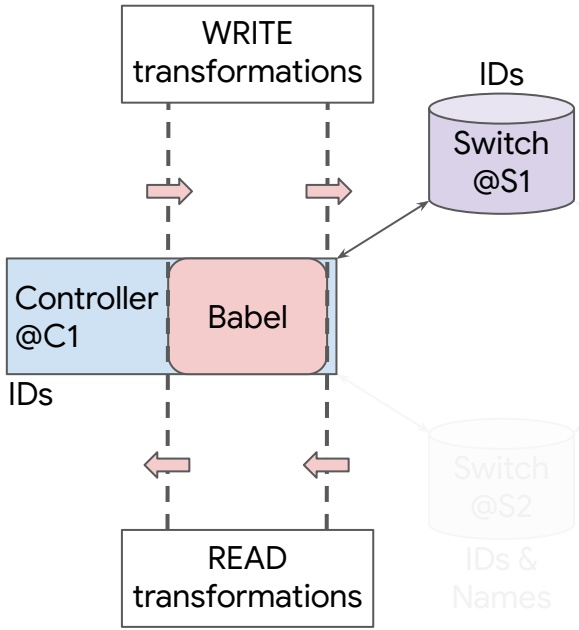
Applies user-provided transformations based on switch version



Can Babel solve our problem?

Initially: Controller and switch both use queue IDs

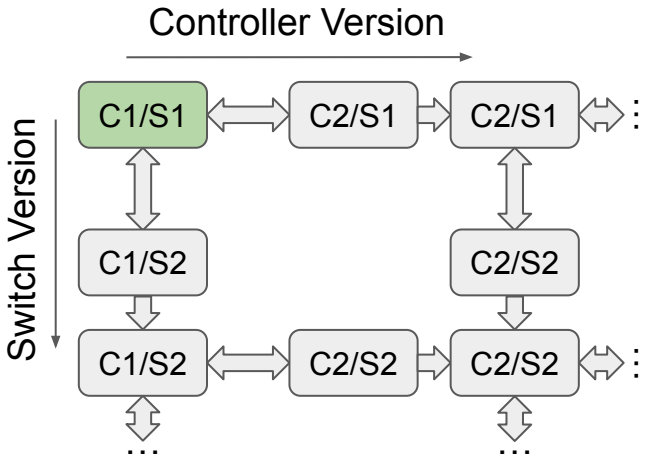


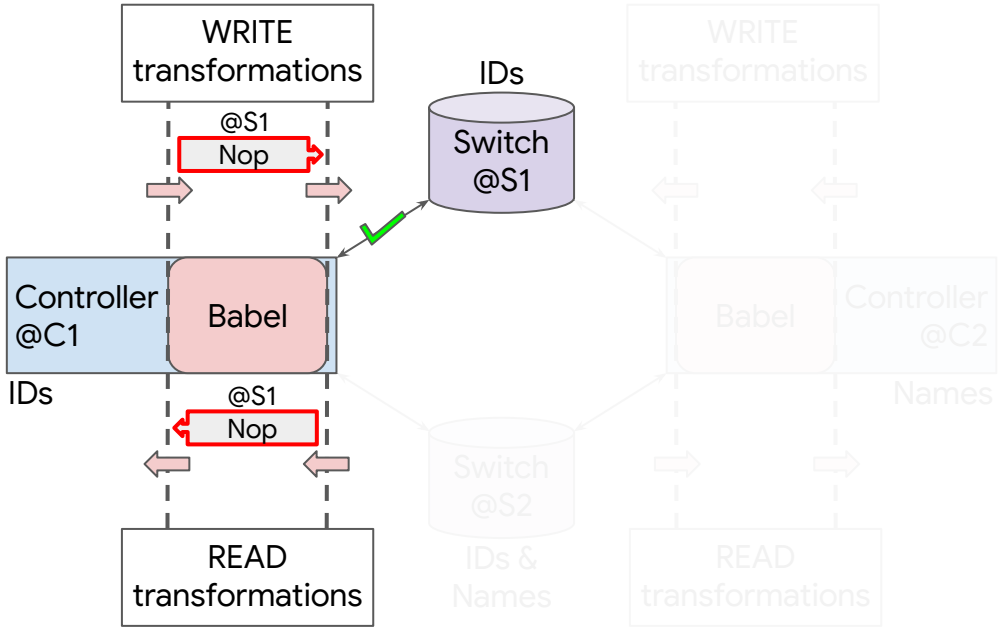


Can Babel solve our problem?

No compatibility issues

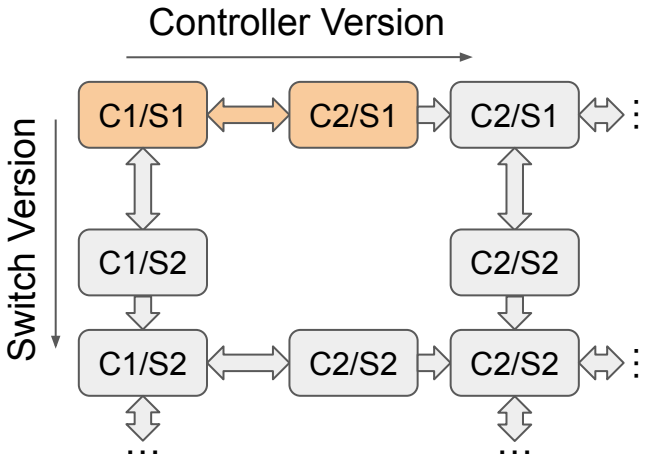
Babel doesn't need to do anything

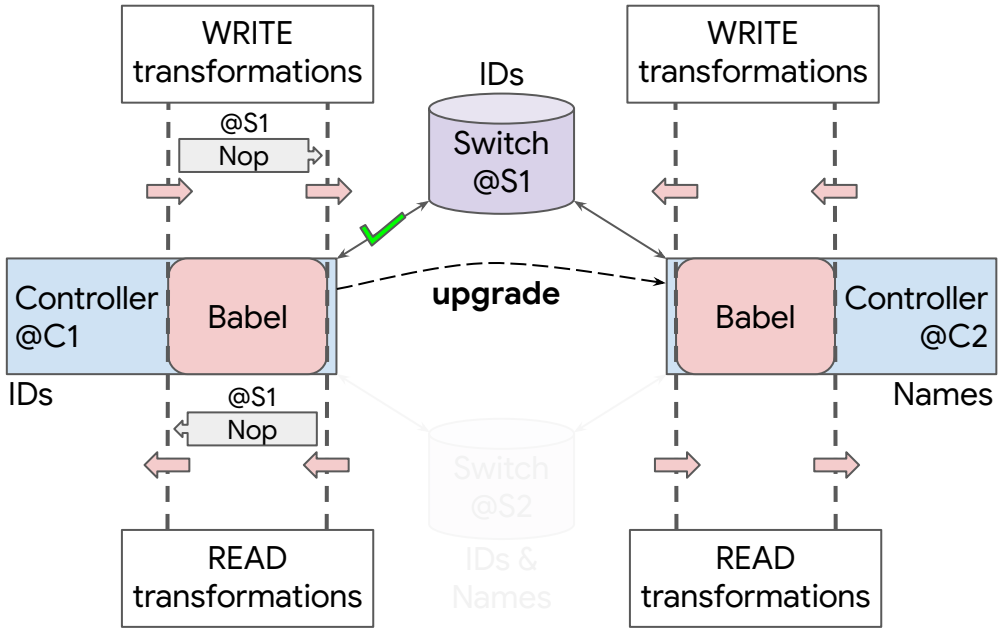




Can Babel solve our problem?

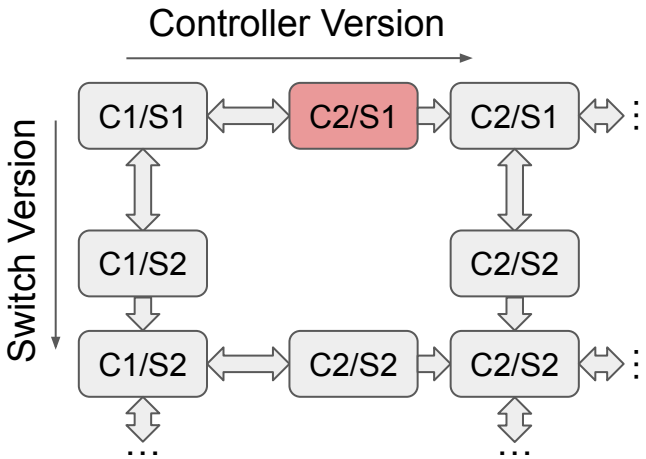
Possible roll-out path 1: Upgrade controller first

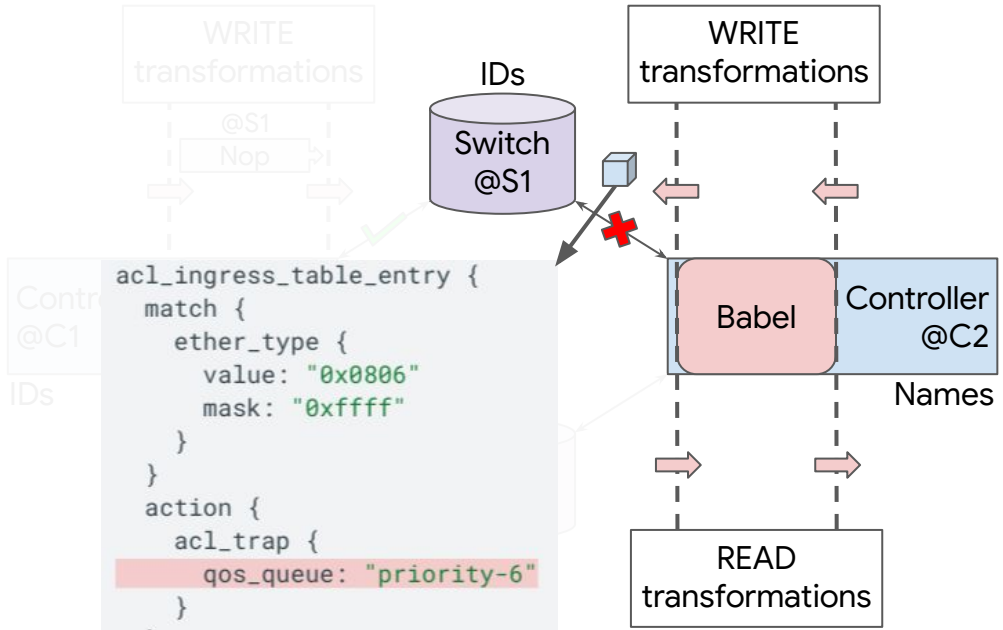




Can Babel solve our problem?

Controller-Switch Compatibility Issue

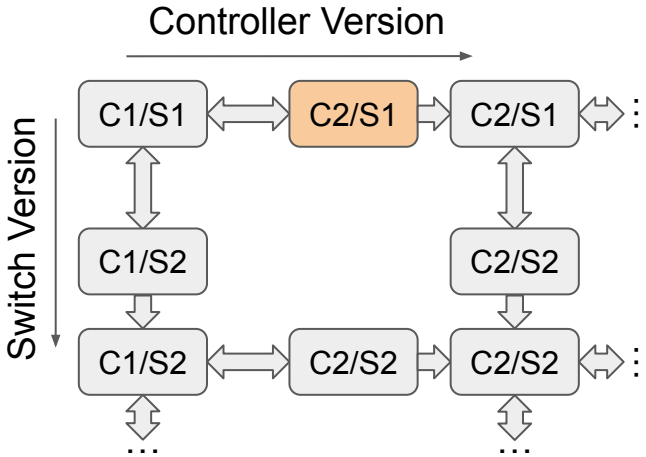


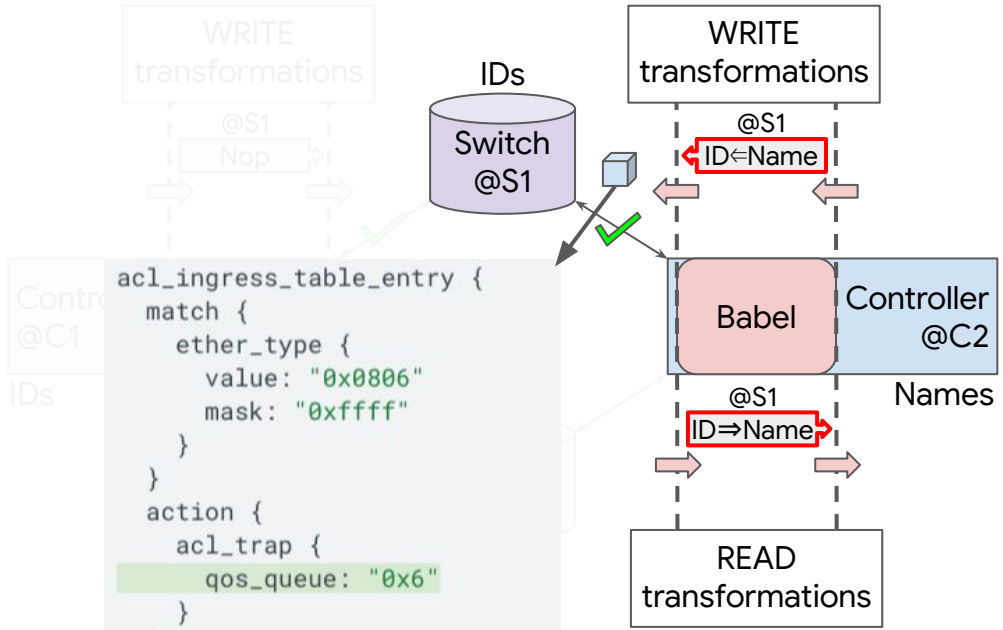


Can Babel solve our problem?

Controller-Switch Compatibility Issue

ID⇔Name Transformations



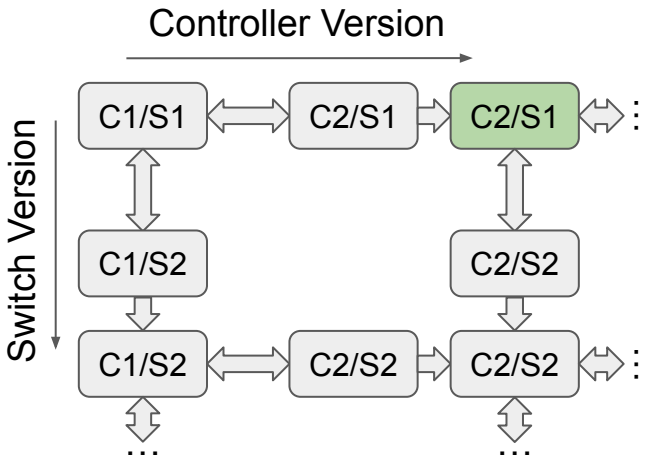


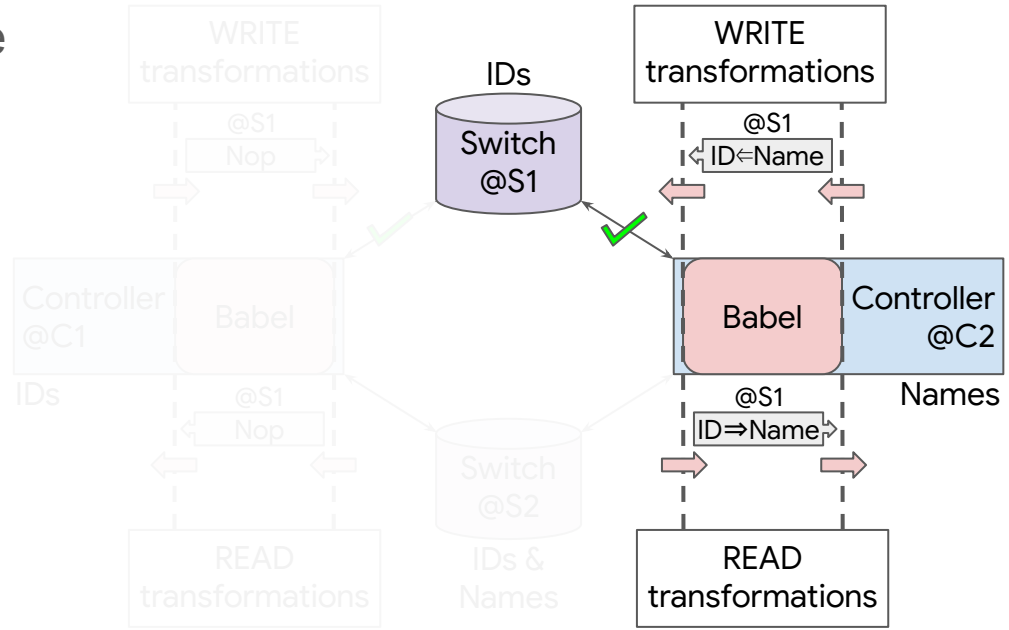

```
acl_ingress_table_entry {
  match {
    ether_type {
      value: "0x0806"
      mask: "0xffff"
    }
  }
  action {
    acl_trap {
      qos_queue: "0x6"
    }
  }
}
```

Can Babel solve our problem?

Controller-Switch Compatibility Issue

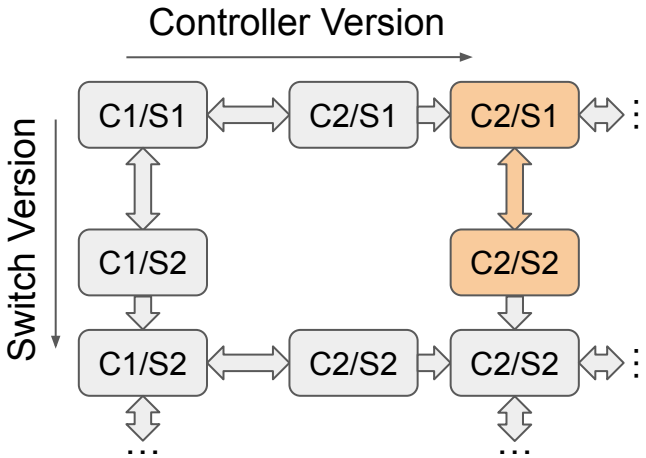
ID⇔Name Transformations

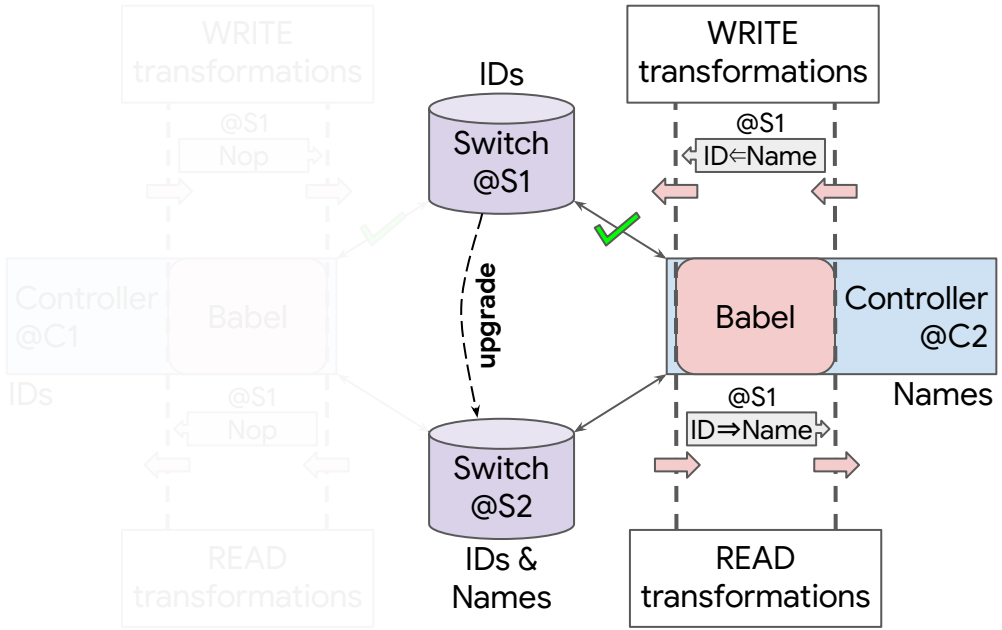




Can Babel solve our problem?

Upgrade switch second

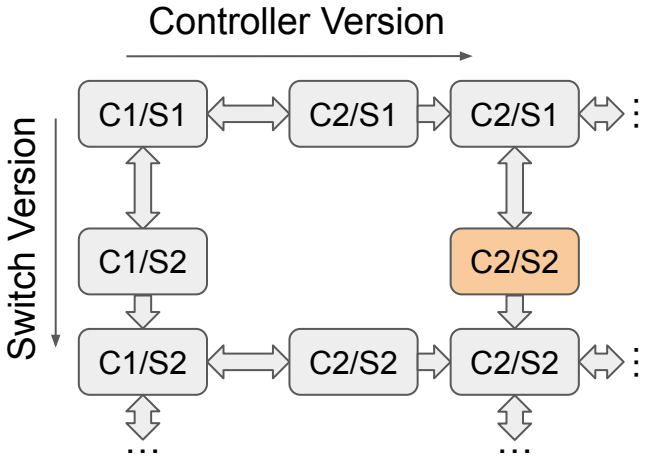


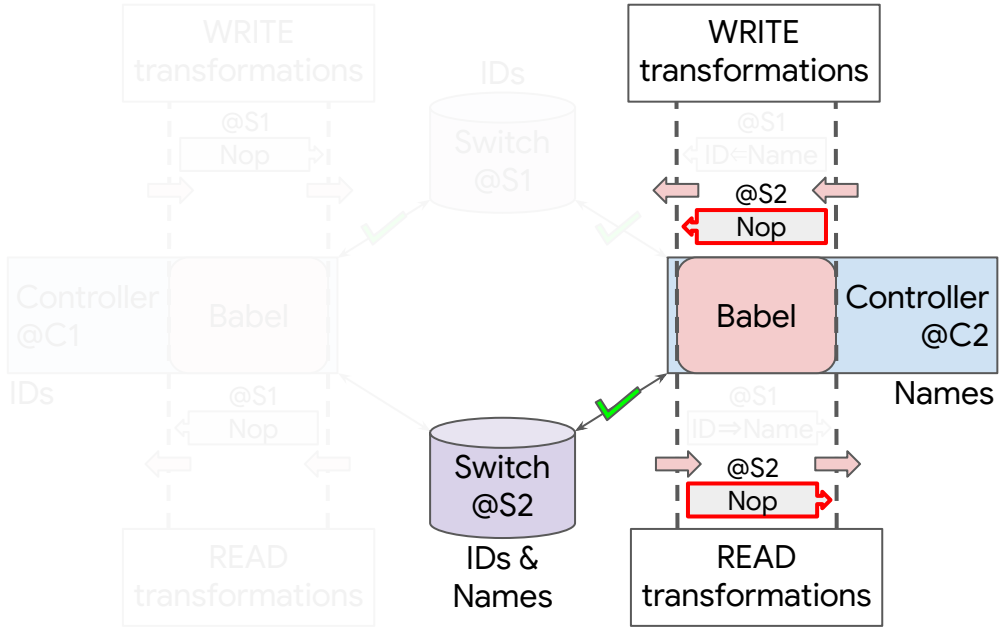


Can Babel solve our problem?

No compatibility issue

Babel doesn't need to do anything

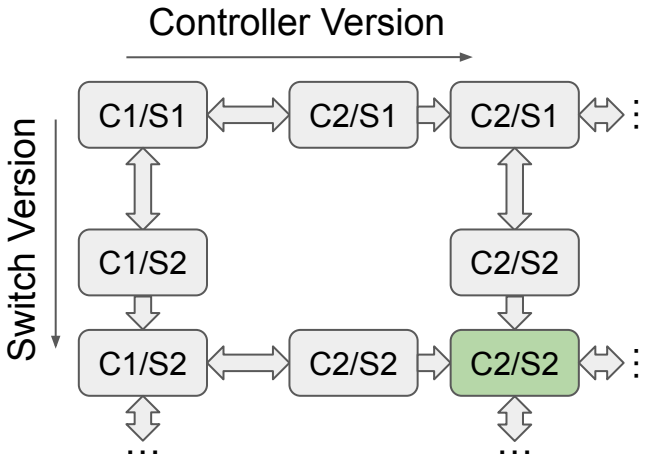


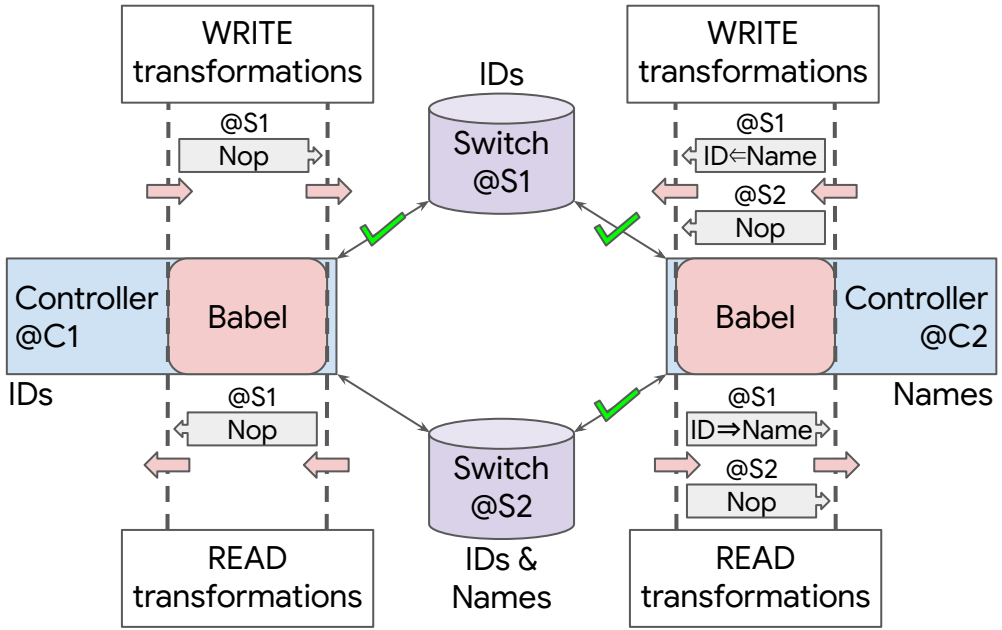


Can Babel solve our problem?

Migration complete

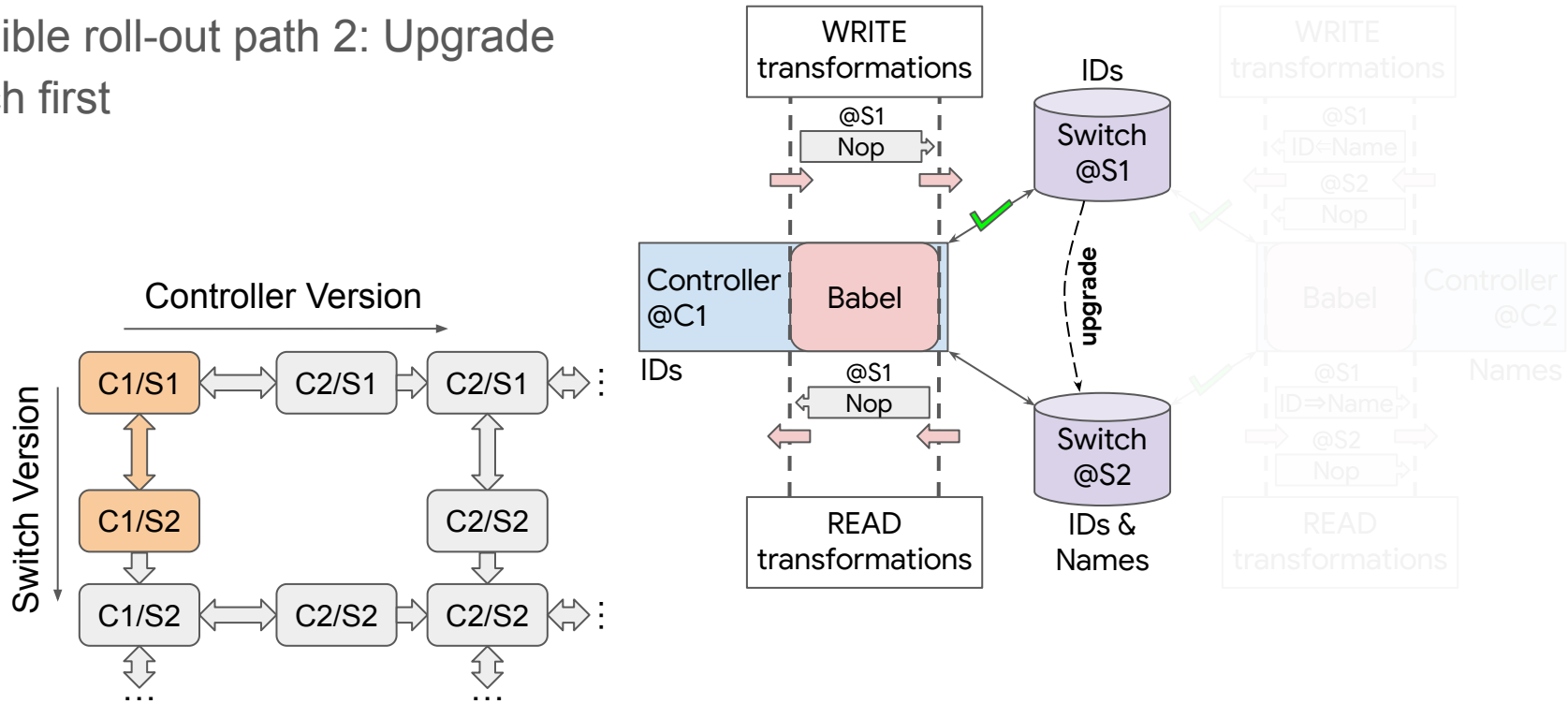
Another roll-out path is possible…





Can Babel solve our problem?

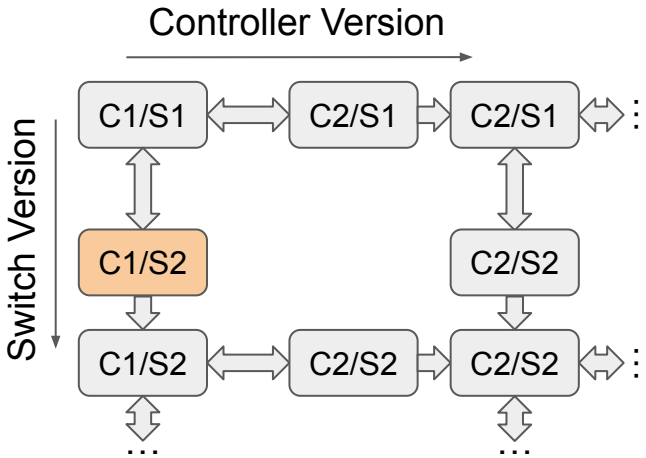
Possible roll-out path 2: Upgrade switch first

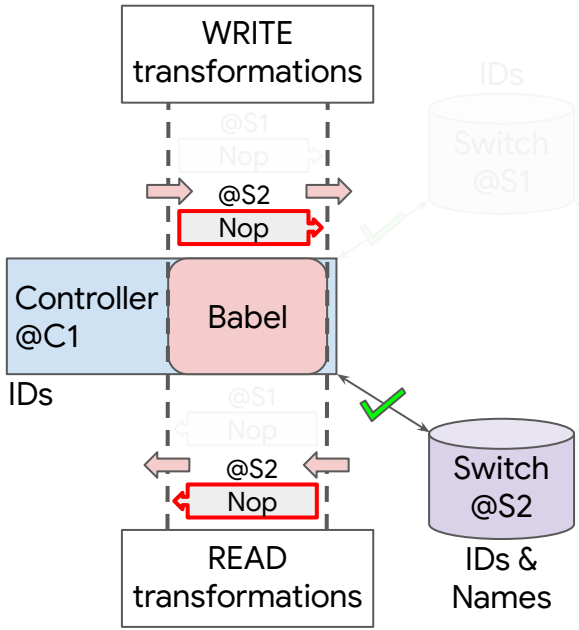


Can Babel solve our problem?

No compatibility issues

Babel doesn't need to do anything

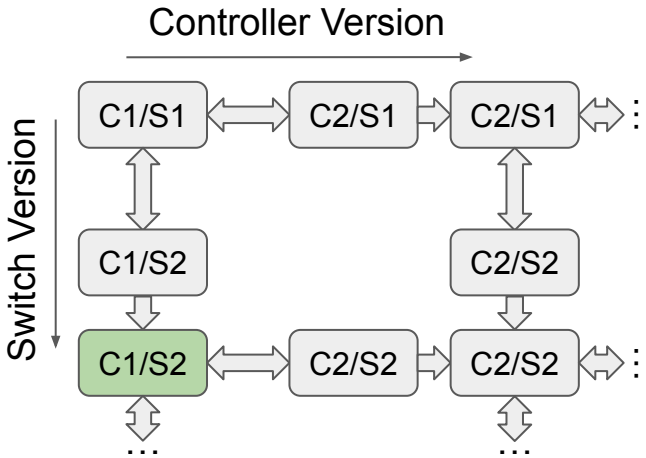


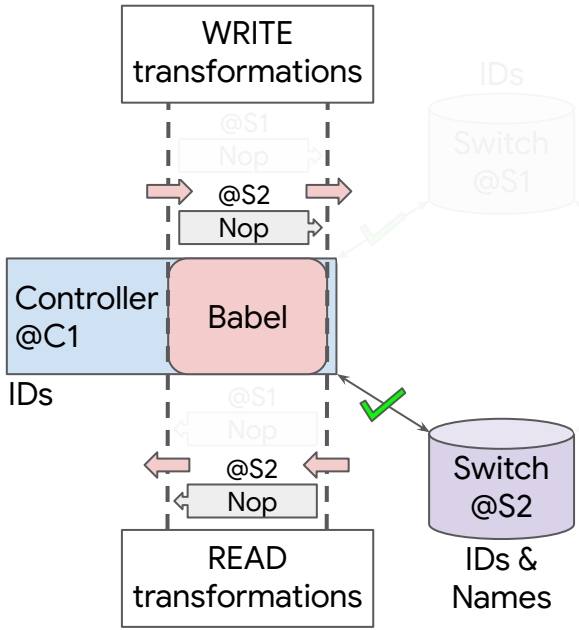


Can Babel solve our problem?

No compatibility issues

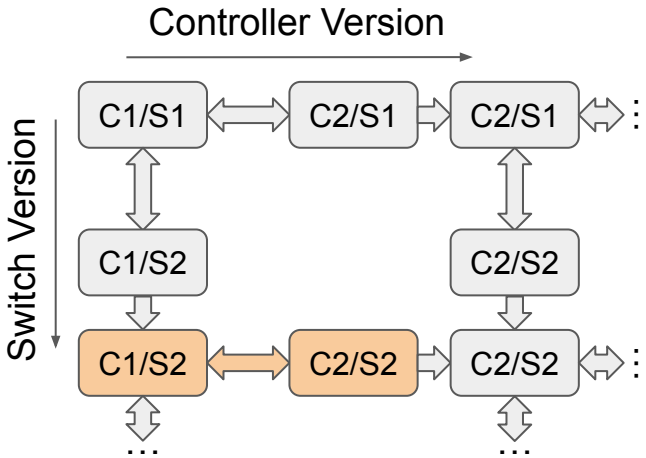
Babel doesn't need to do anything

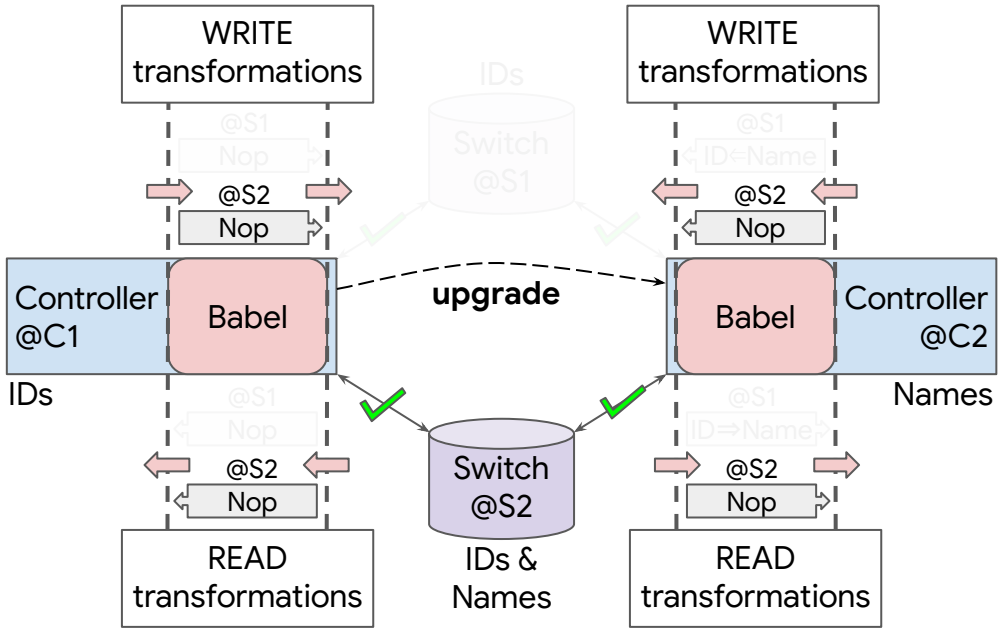




Can Babel solve our problem?

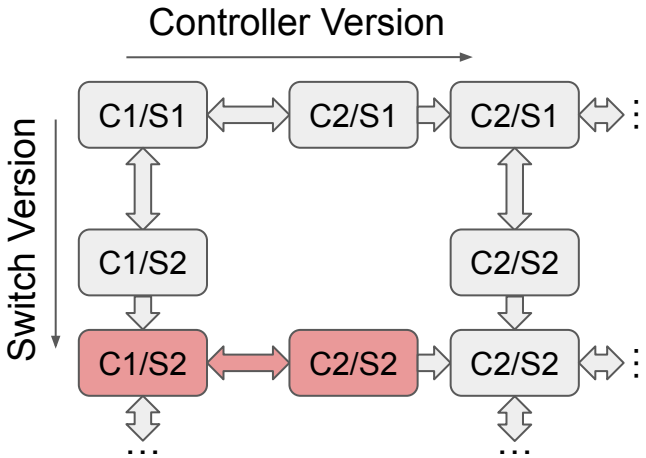
Upgrade controller second

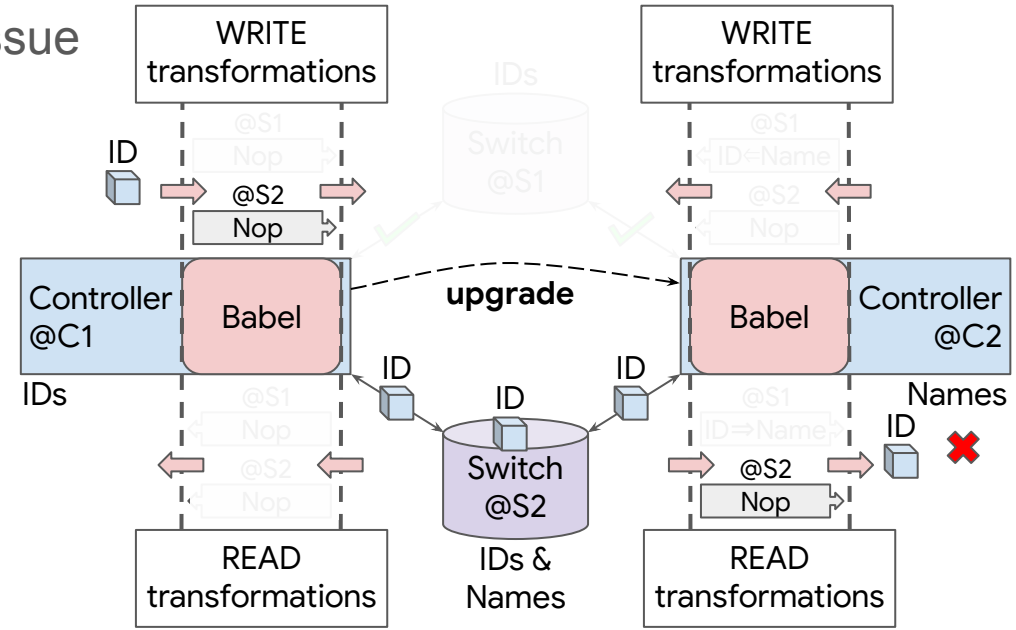




Can Babel solve our problem?

Controller-Controller Compatibility Issue



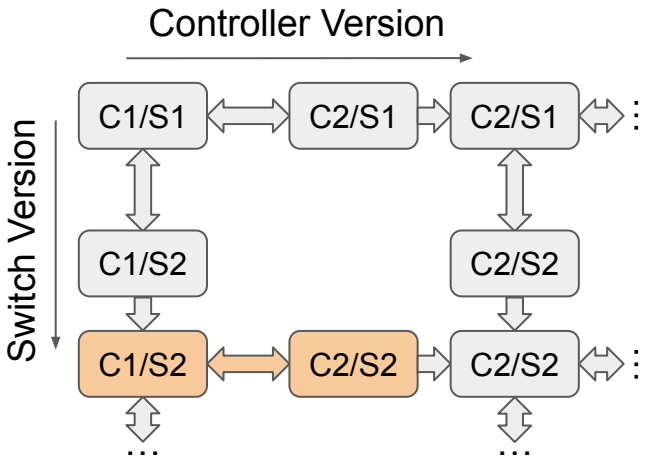


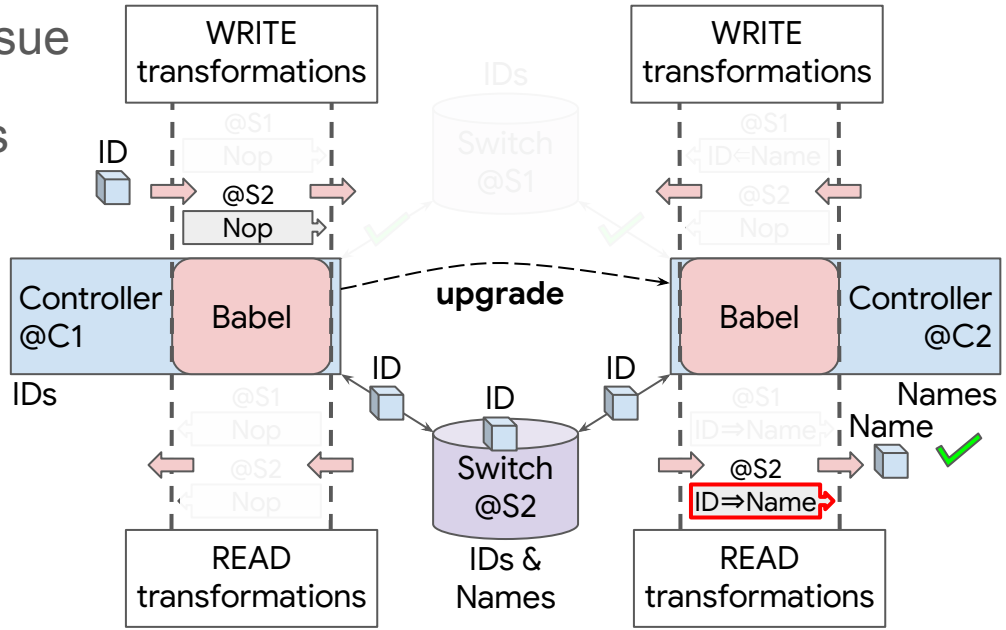
Can Babel solve our problem?

Controller-Controller Compatibility Issue

ID⇒Name Transformation On Reads

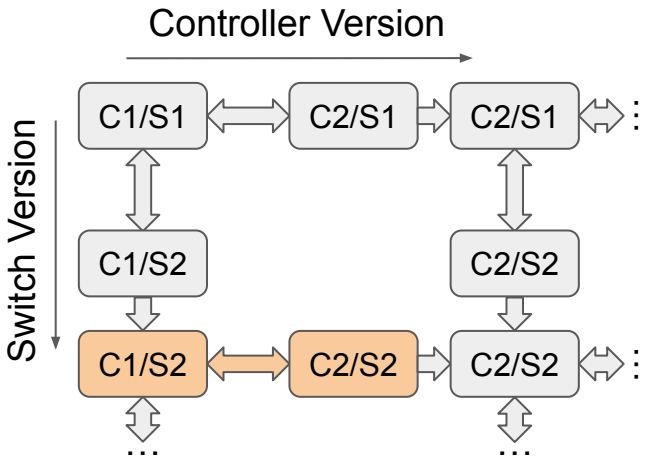
*ID⇒Name is NOP for Names

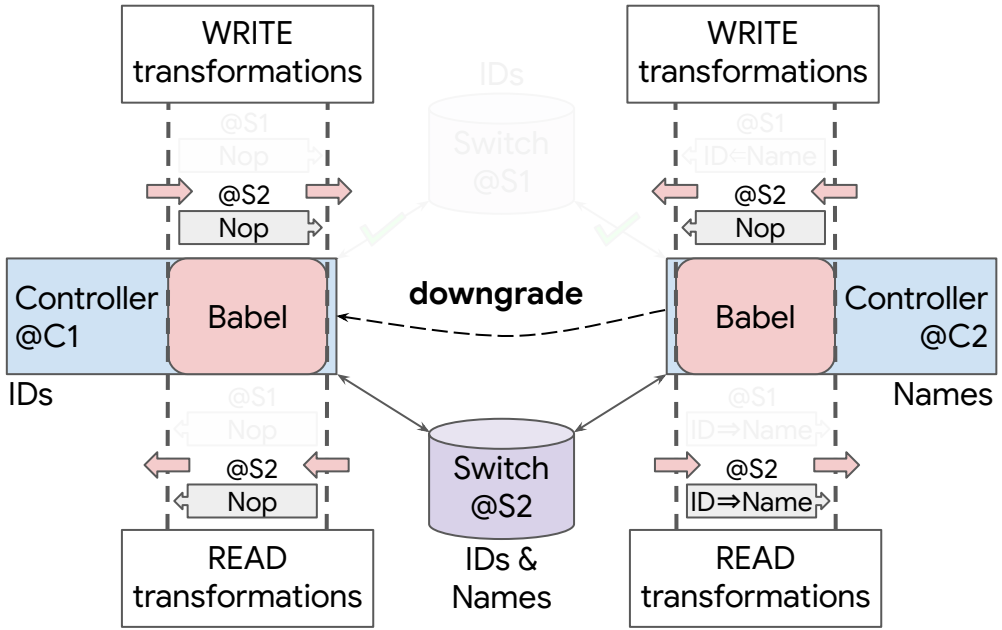




Can Babel solve our problem?

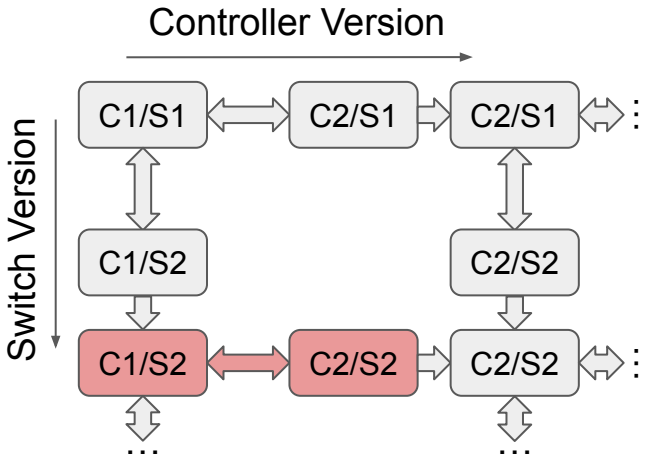
Downgrades could occur

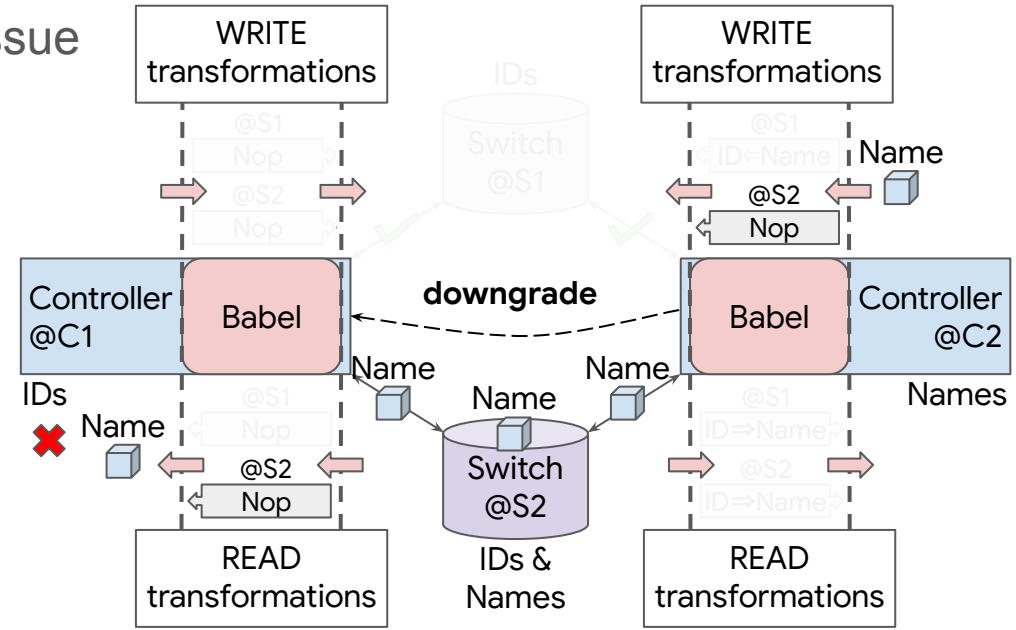




Can Babel solve our problem?

Controller-Controller Compatibility Issue



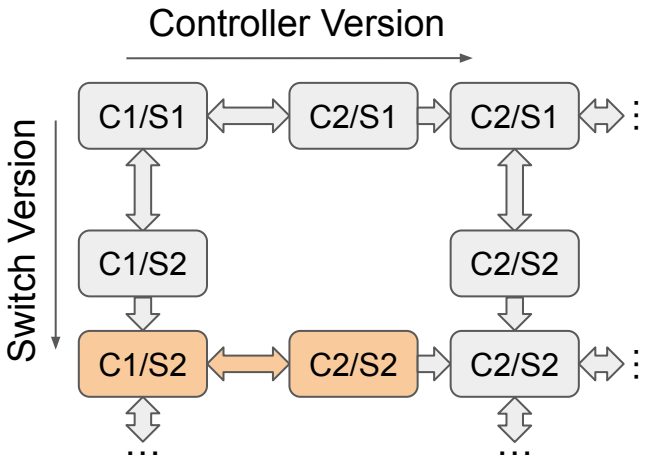


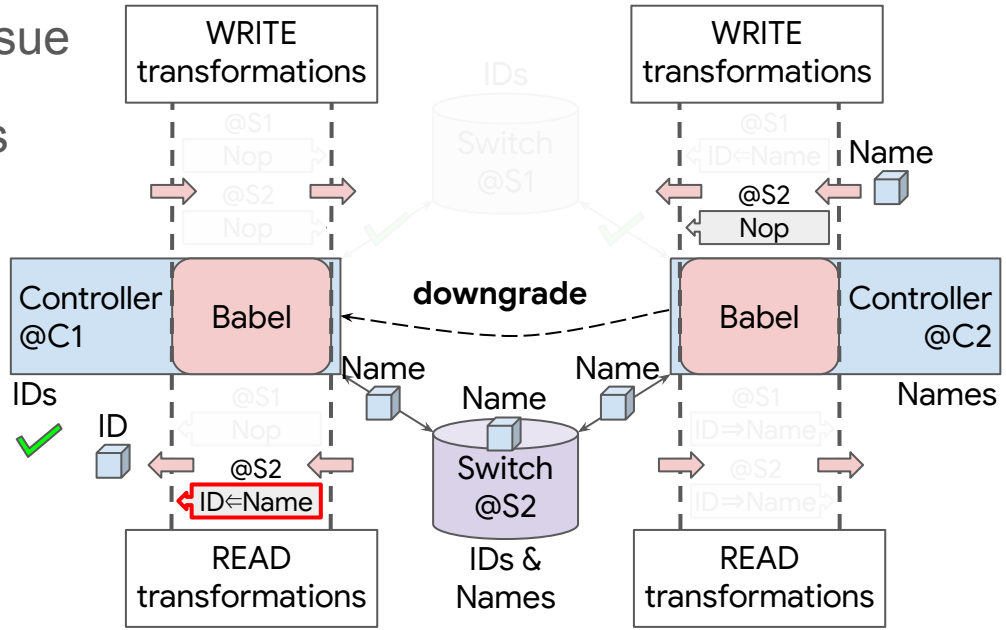
Can Babel solve our problem?

Controller-Controller Compatibility Issue

Name⇒ID Transformation On Reads

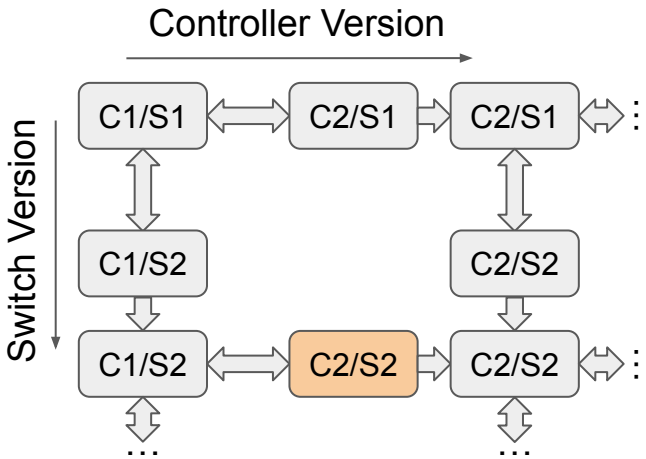
*Name⇒ID is NOP for IDs

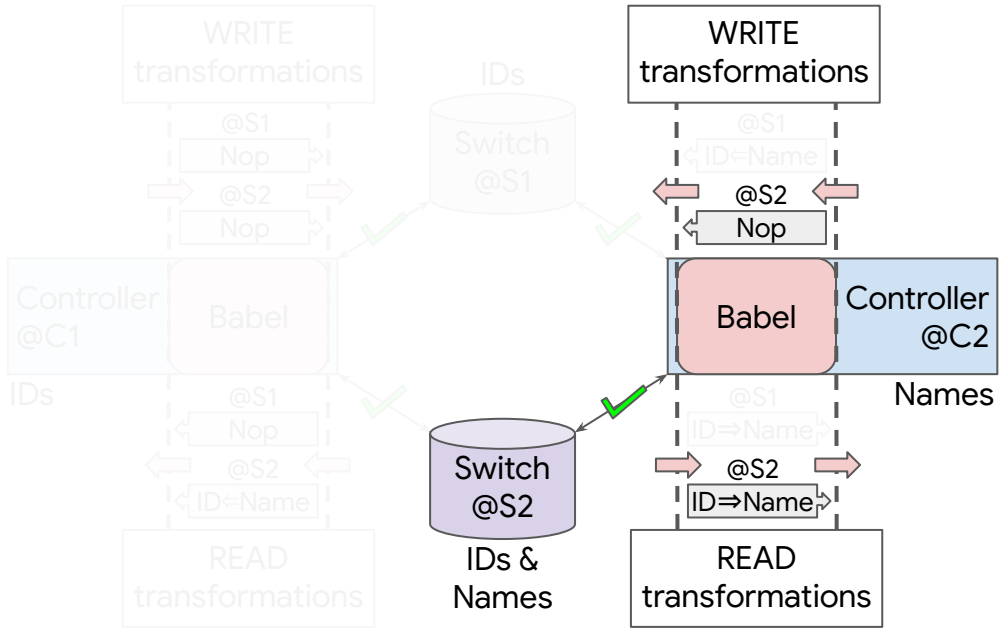




Can Babel solve our problem?

No compatibility issues

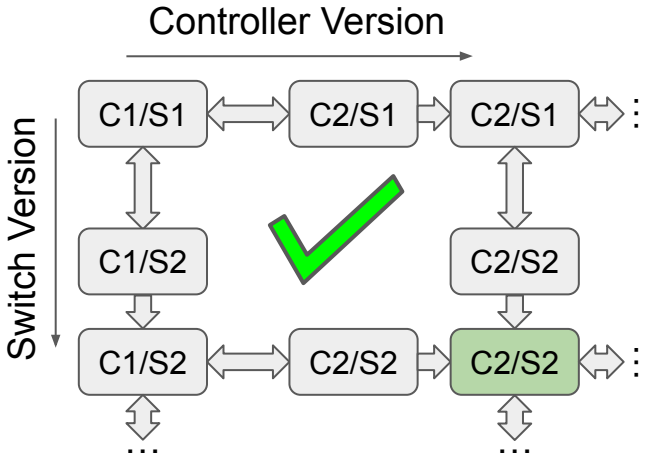


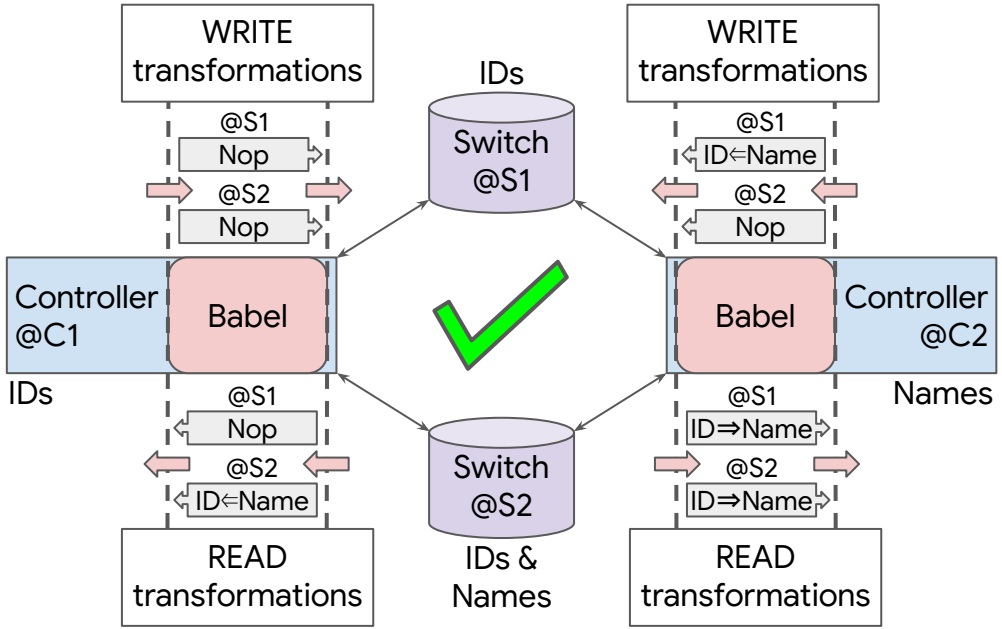


Can Babel solve our problem? Yes it can

Migration complete

Both rollout paths are supported





Roadmap

1. Model & Requirements ✓
2. Example: Why is network evolution hard? ✓
3. Solution: Babel ✓
4. **How general is Babel?**
5. How do you use Babel?

What about a breaking change?

Optionals contain a value

Ternaries contain a value and a mask

Optionals can be represented as ternaries (mask is all 1’s if present, all 0’s if omitted)

Change: Use type Ternary instead of Optional in match field (on existing value set)

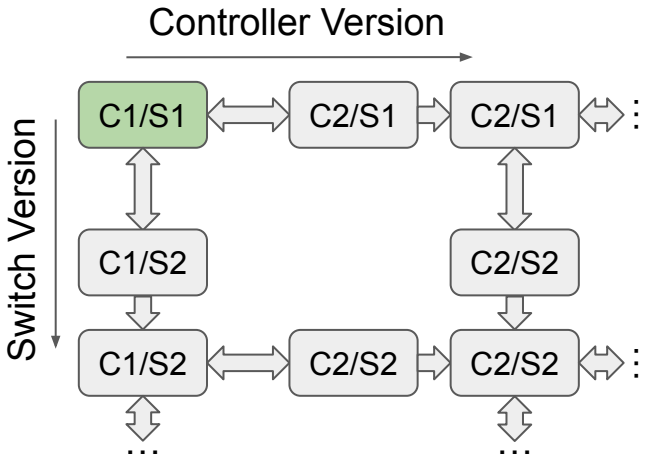
Motivation: Align with reality, Optionals are already treated as Ternaries on switch

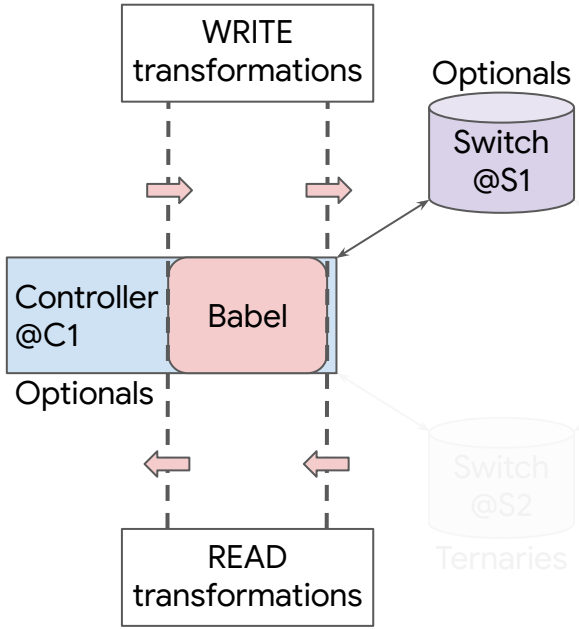
```
acl_ingress_table_entry {
  match {
    metadata {
      value: "0xbed7"
+     mask: "0xffff"
    }
  }
  action {
    acl_drop {
    }
  }
}
```

Ternary Mask

Can Babel handle a breaking change?

Initially: Controller and switch both use type Optional

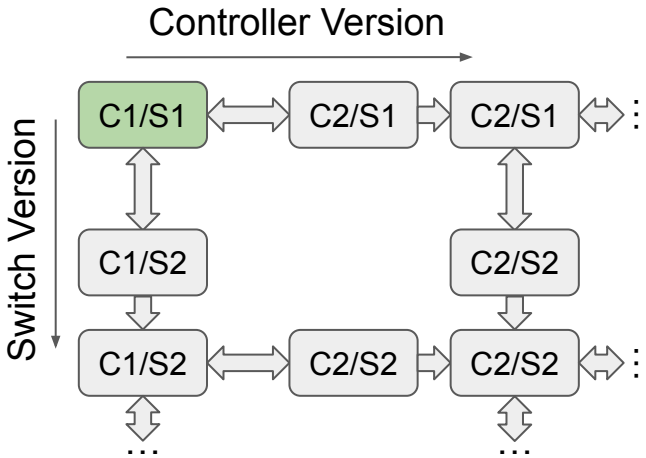


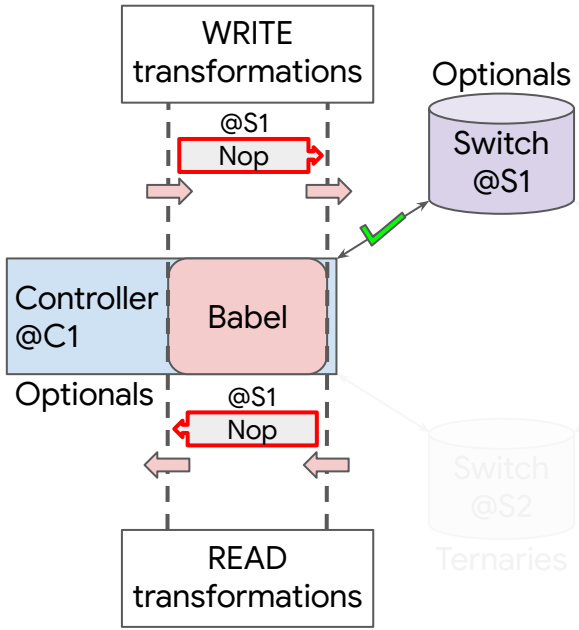


Can Babel handle a breaking change?

No compatibility issues

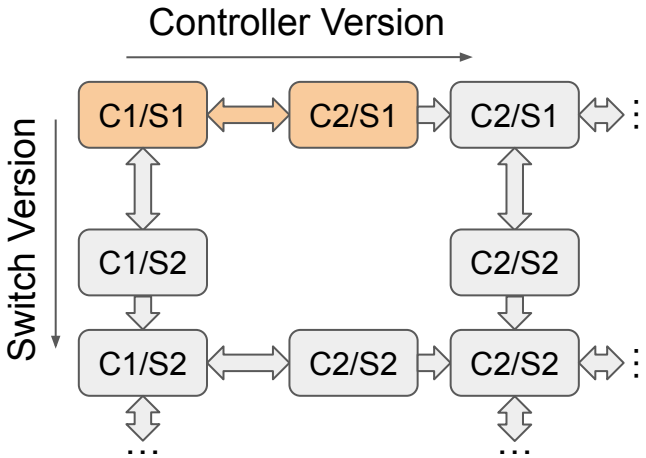
Babel doesn't need to do anything

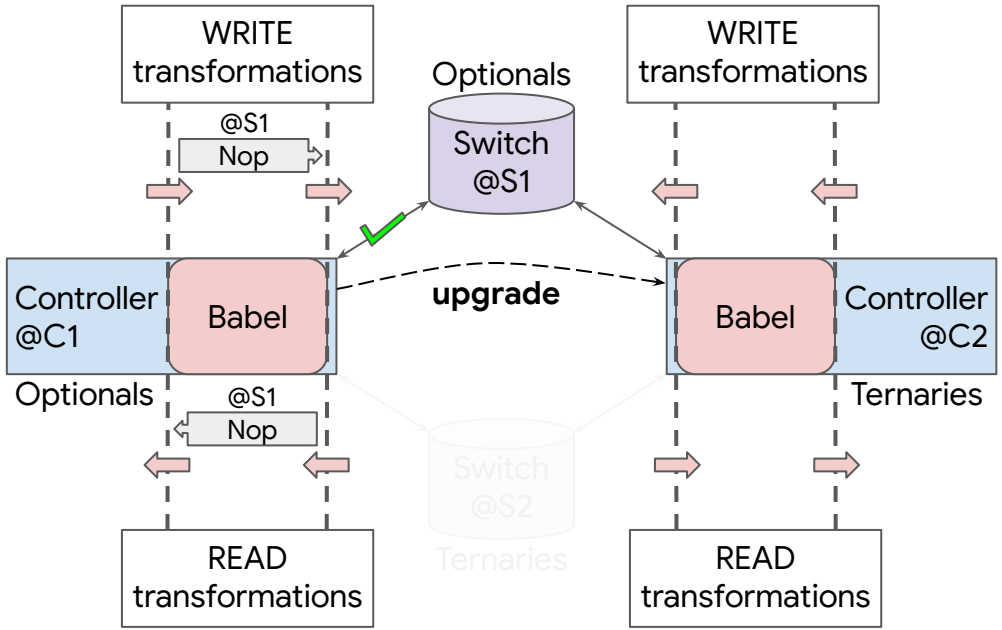




Can Babel handle a breaking change?

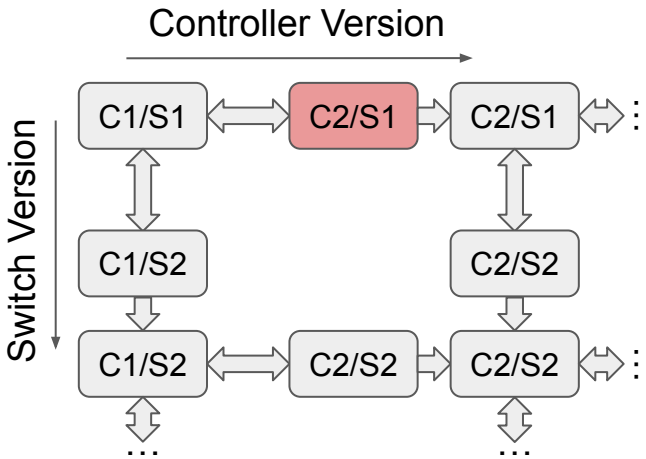
Possible roll-out path 1: Upgrade controller first

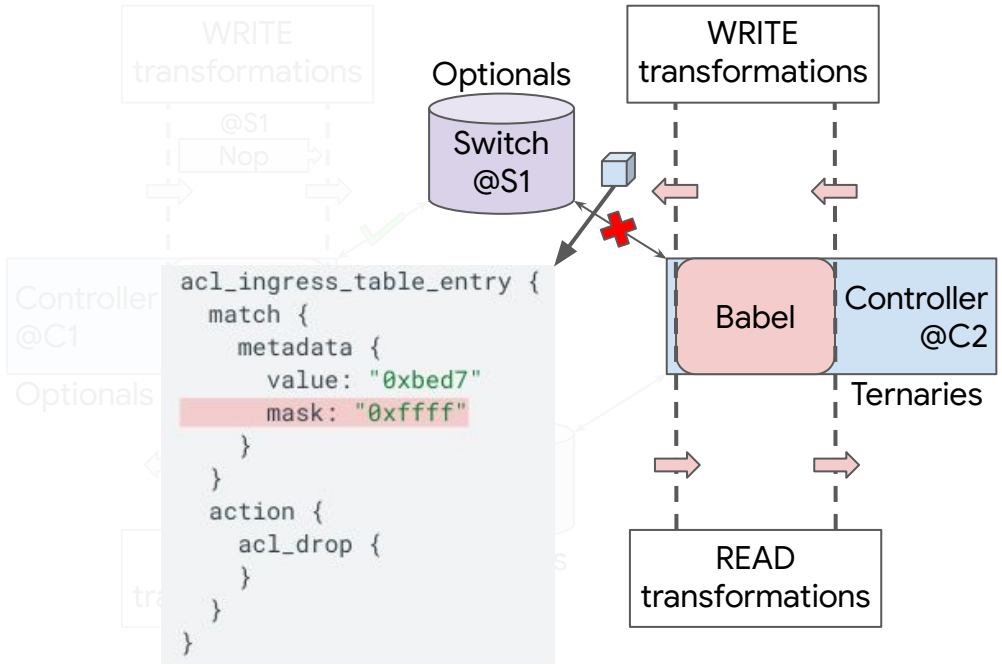




Can Babel handle a breaking change?

Controller-Switch Compatibility Issue

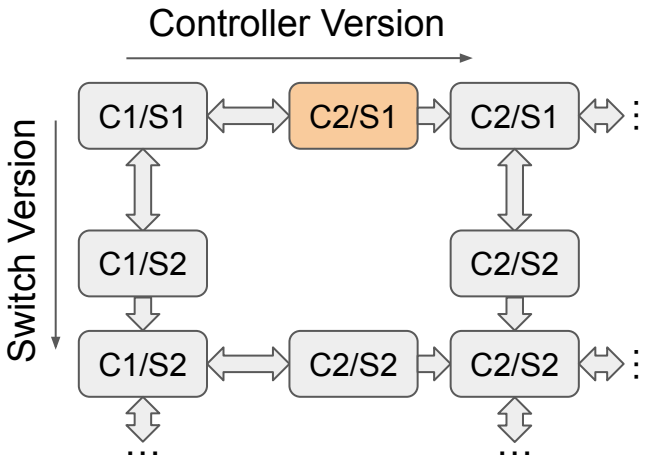


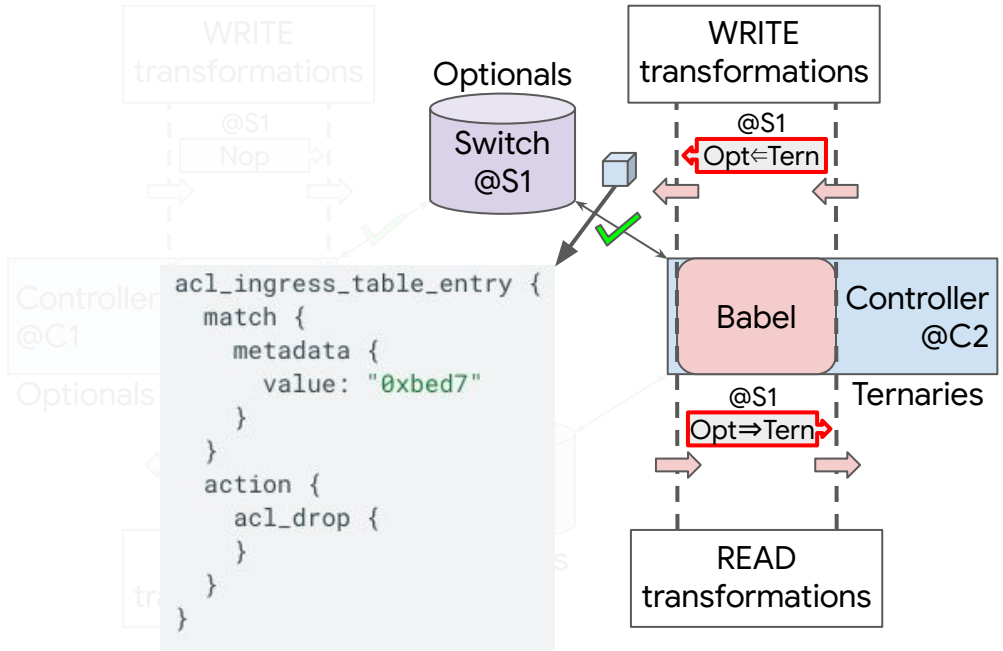


Can Babel handle a breaking change?

Controller-Switch Compatibility Issue

Optional⇔Ternary Transformations

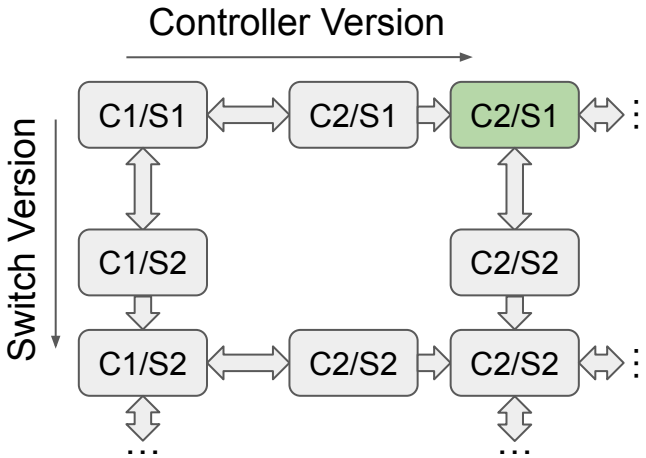


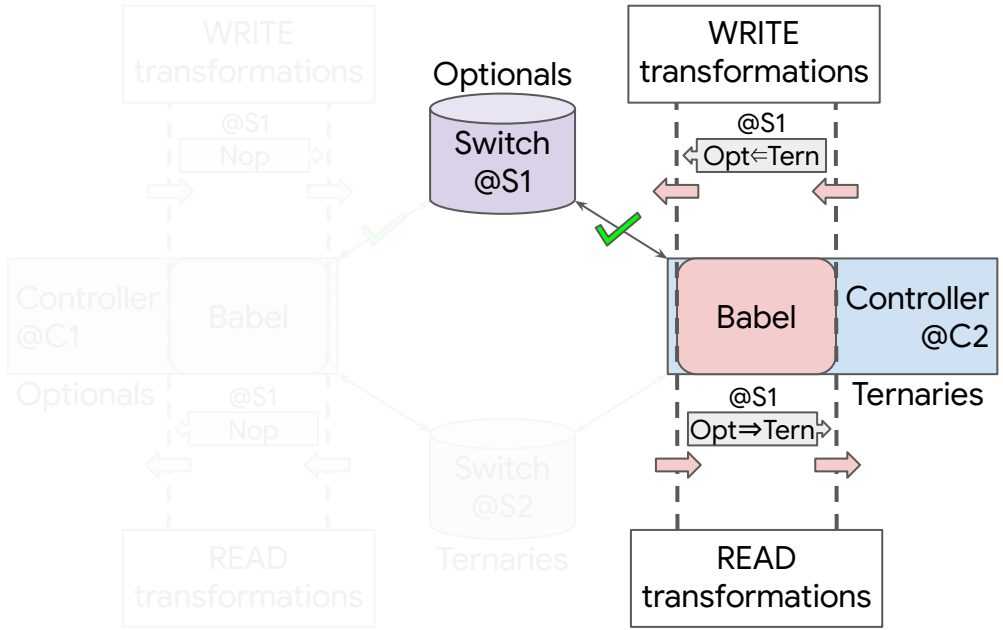


Can Babel handle a breaking change?

Controller-Switch Compatibility Issue

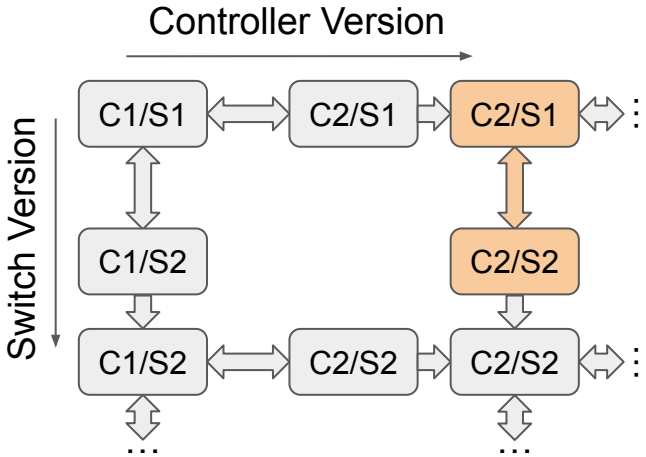
Optional⇔Ternary Transformations

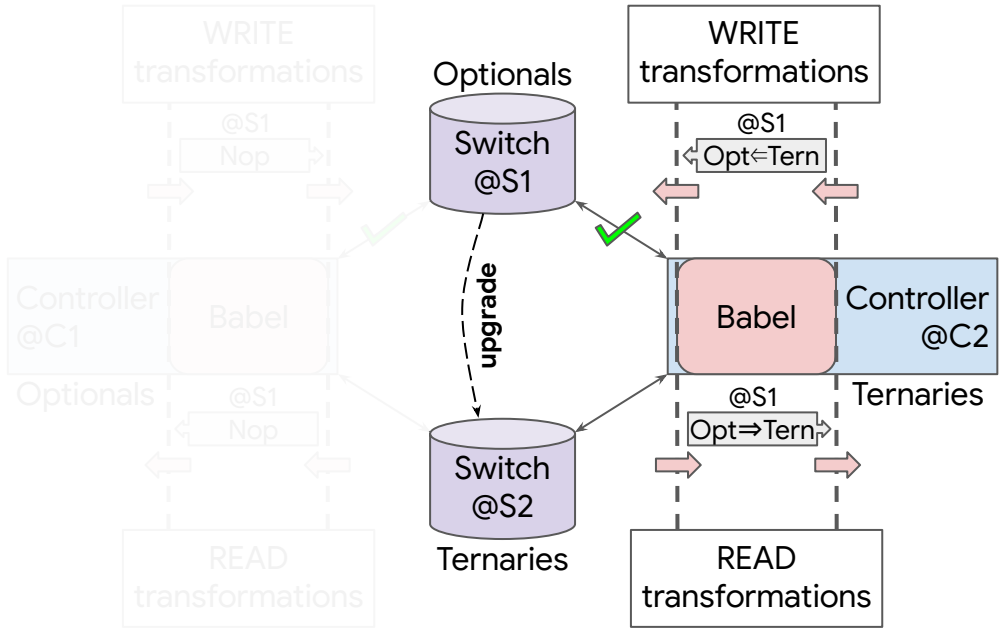




Can Babel handle a breaking change?

Upgrade switch second

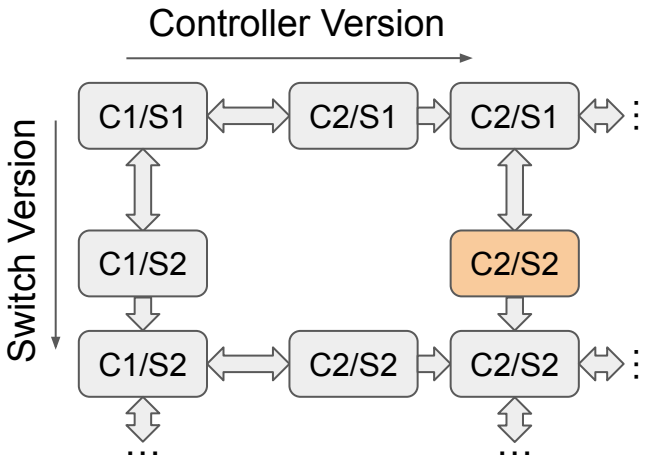


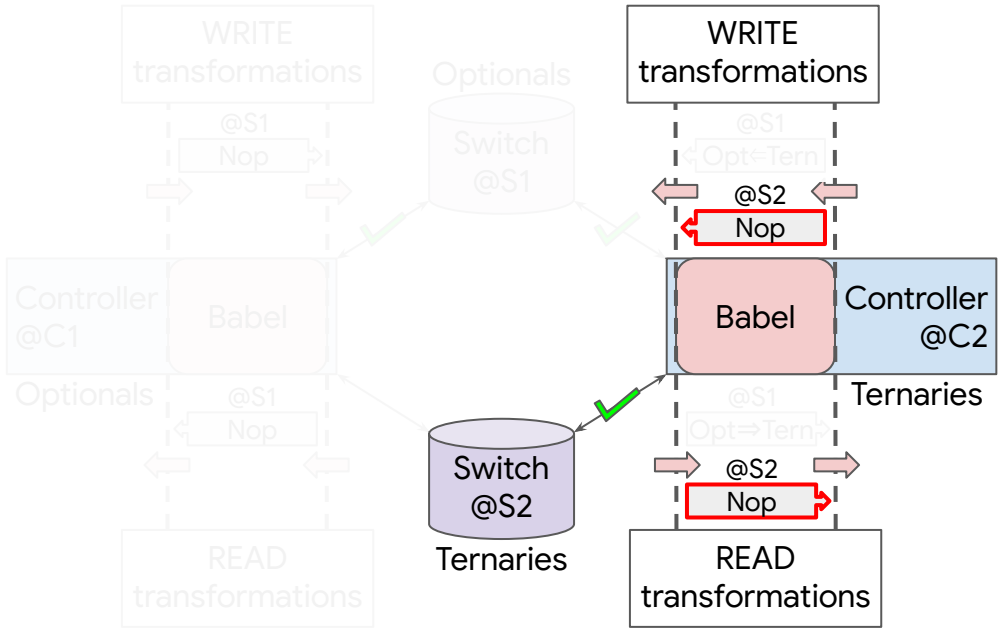


Can Babel handle a breaking change?

No compatibility issue

Babel doesn't need to do anything

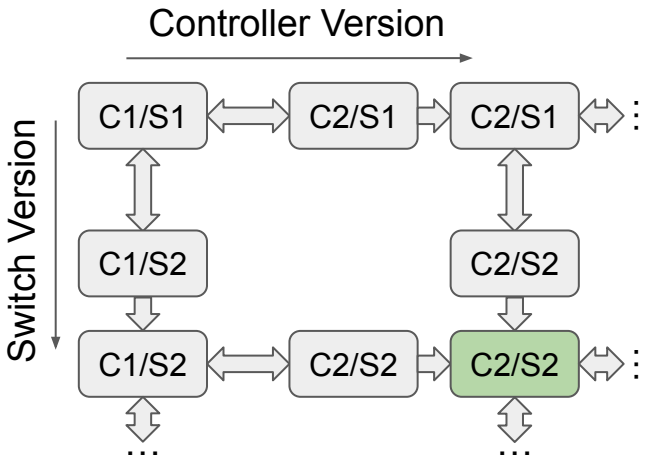


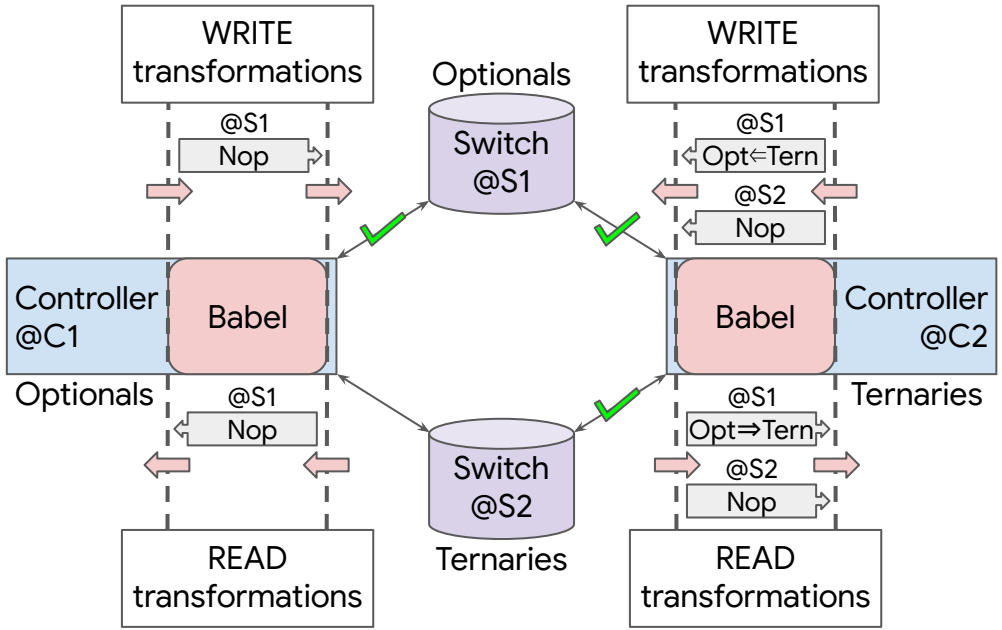


Can Babel handle a breaking change?

Migration complete

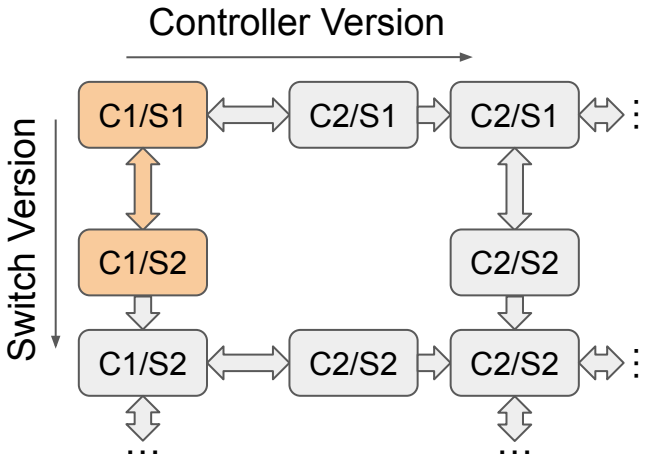
Another roll-out path is possible…

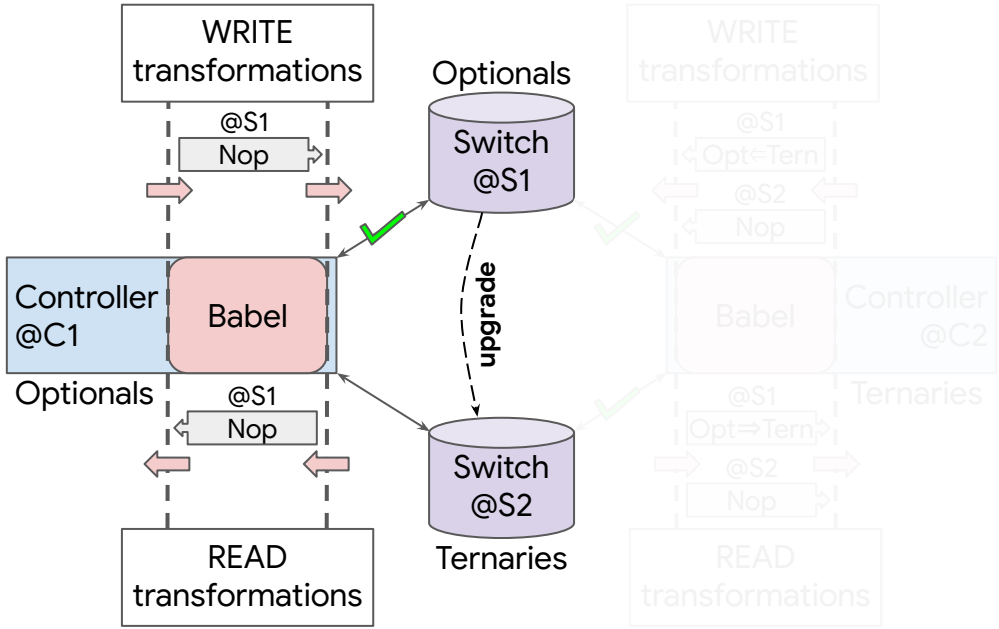




Can Babel handle a breaking change?

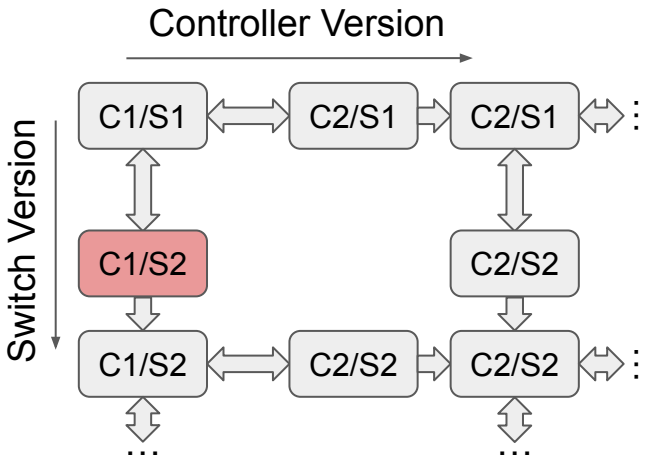
Possible roll-out path 2: Upgrade switch first

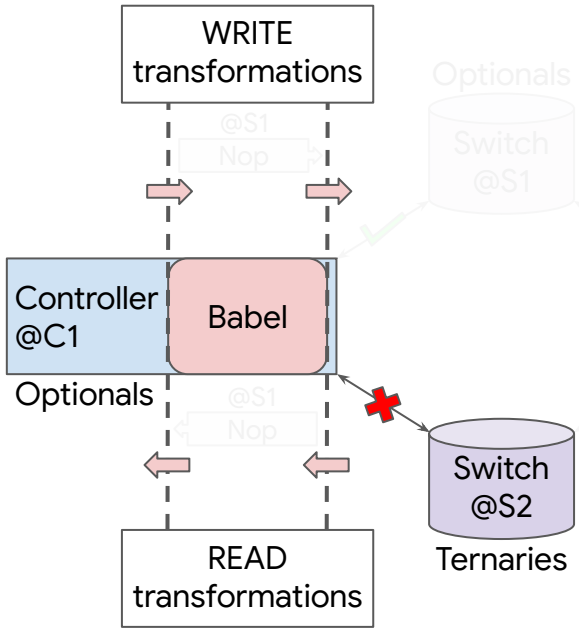




Can Babel handle a breaking change?

Controller-Switch compatibility issue

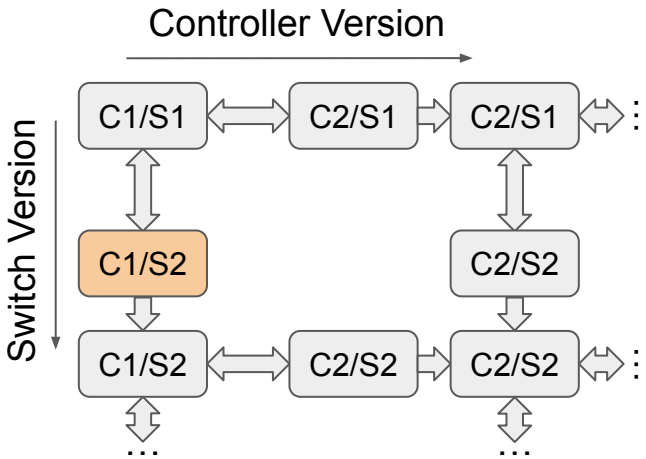


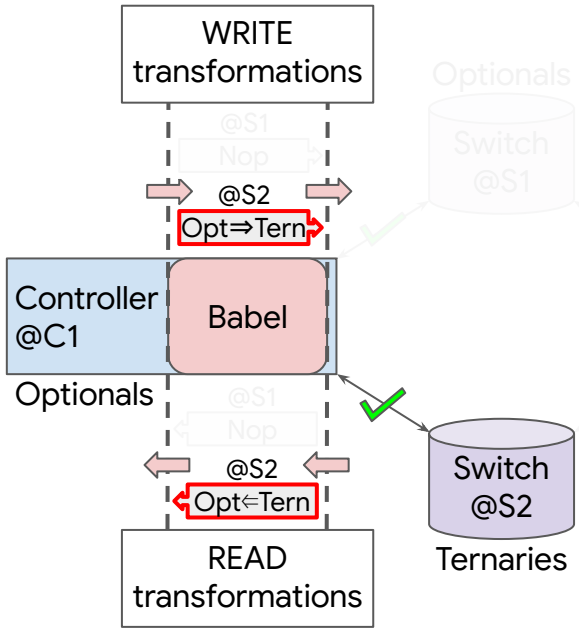


Can Babel handle a breaking change?

Controller-Switch compatibility issue

Optional⇔Ternary Transformations

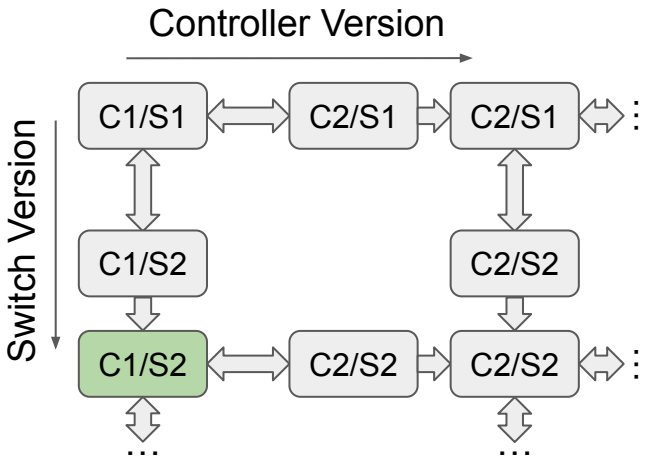


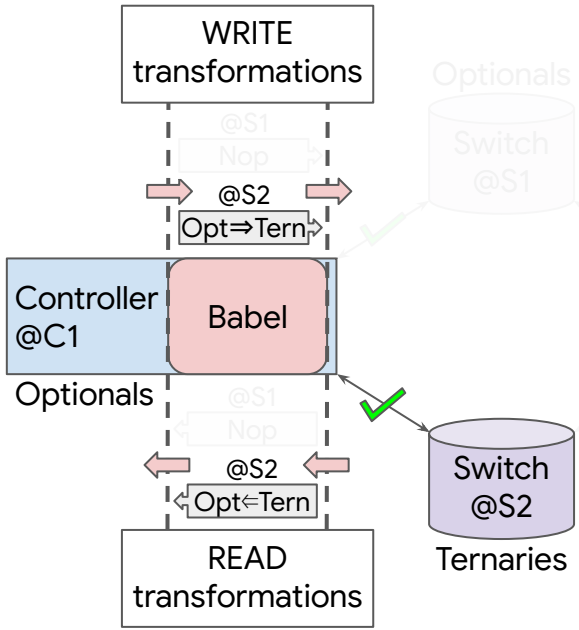


Can Babel handle a breaking change?

Controller-Switch compatibility issue

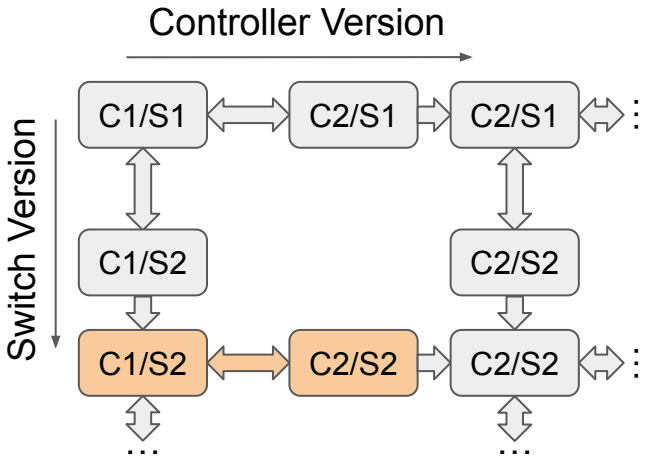
Optional⇔Ternary Transformations

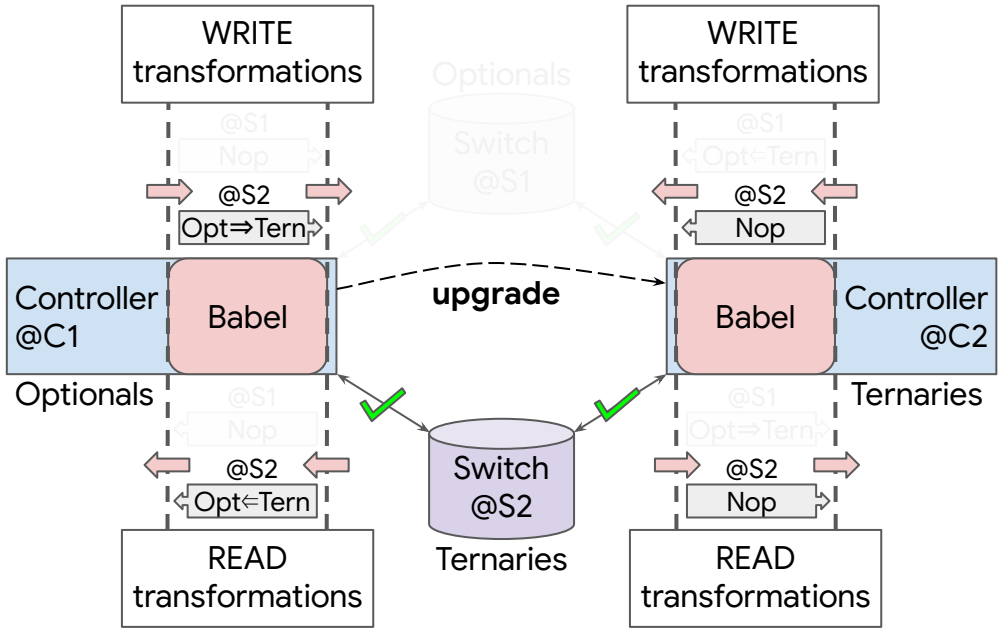




Can Babel handle a breaking change?

Upgrade controller second

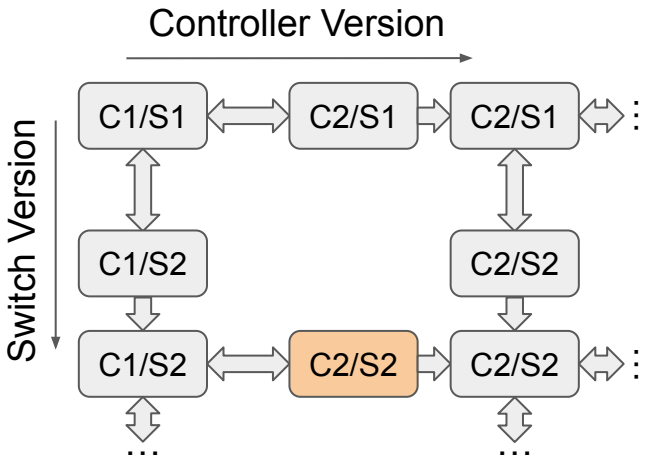


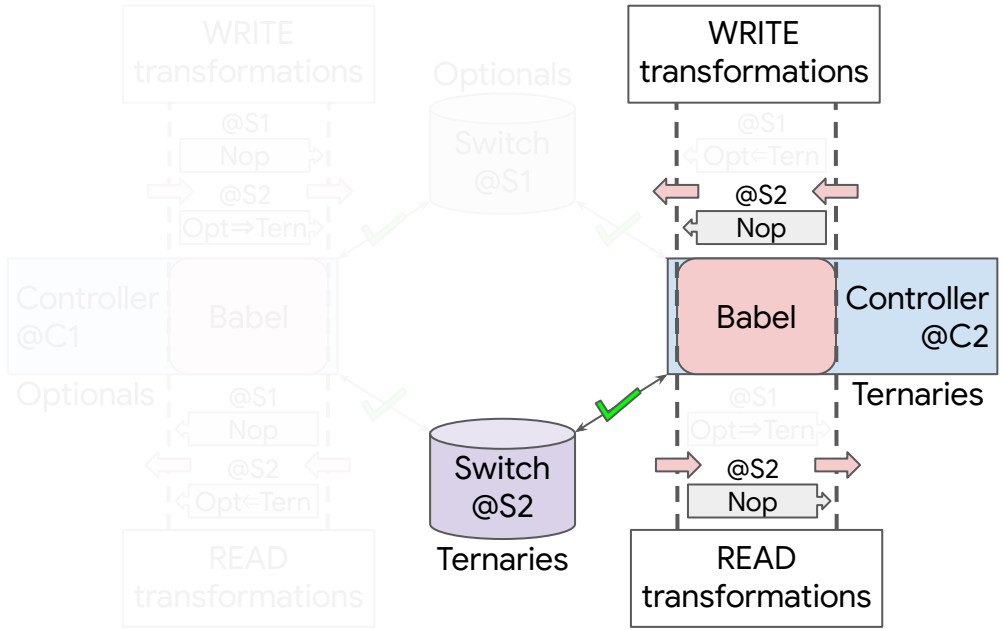


Can Babel handle a breaking change?

No compatibility issues

Babel doesn't need to do anything

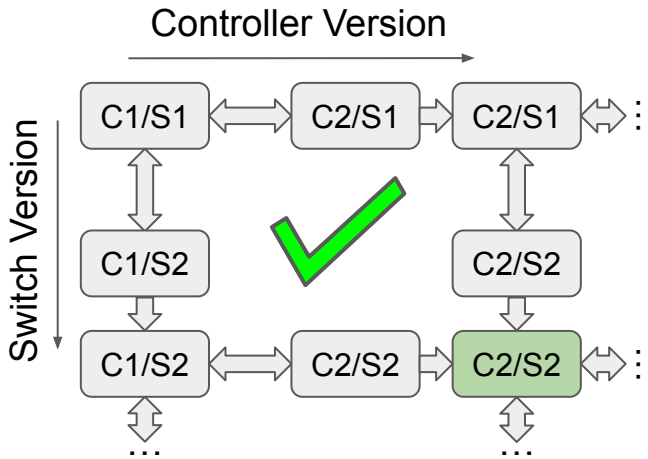


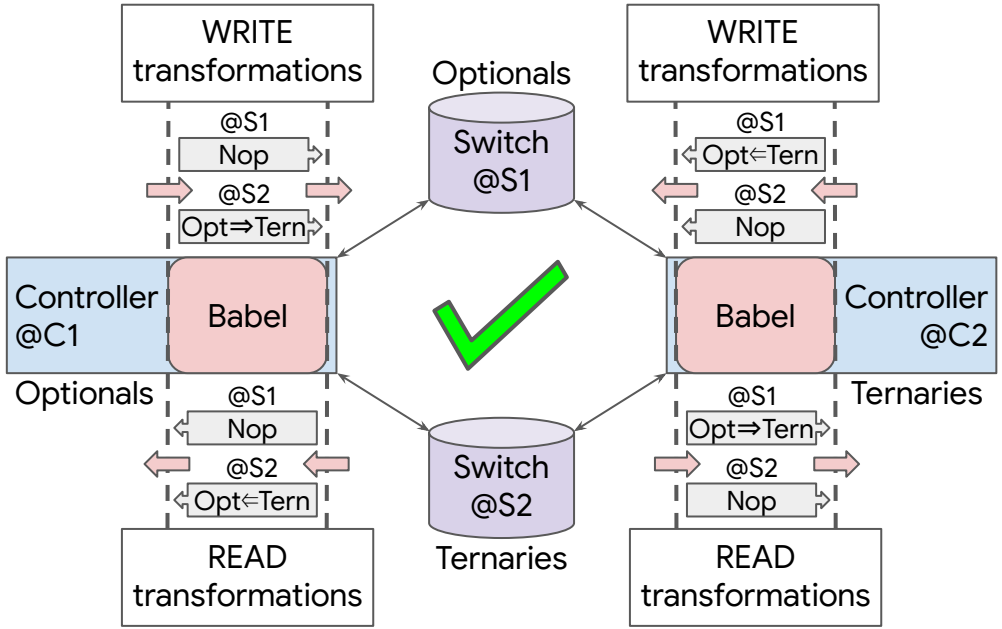


Can Babel handle a breaking change? Yes it can

Migration complete

Both rollout paths are supported





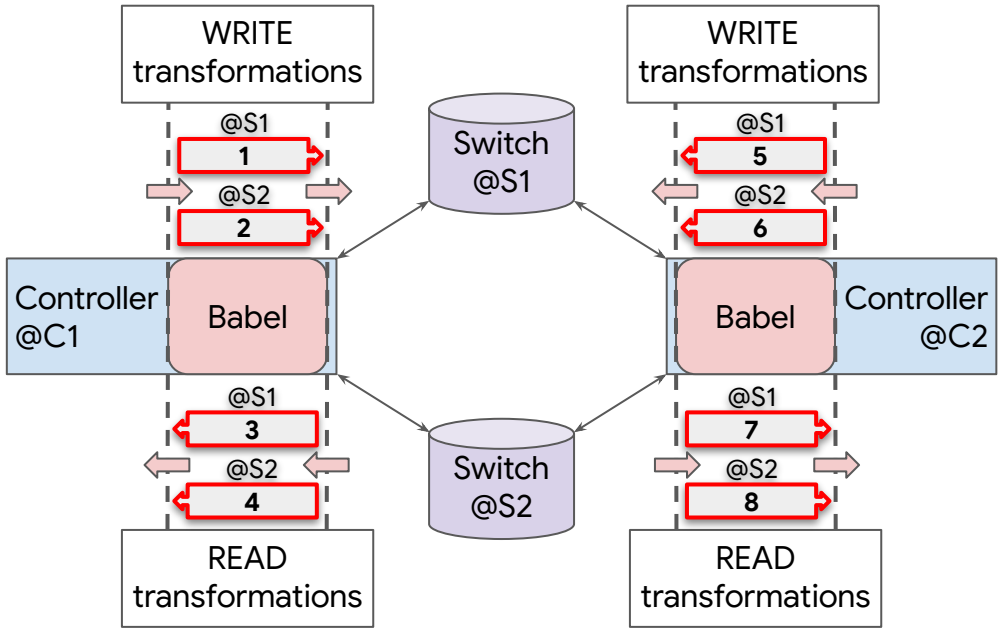
Roadmap

1. Model & Requirements ✓
2. Example: Why is network evolution hard? ✓
3. Solution: Babel ✓
4. How general is Babel? ✓
5. **How do you use Babel?**

How do you use Babel?

Babel provides a framework for smooth rollouts:

- Users specify 8 transformations
- Babel ensures safe rollout



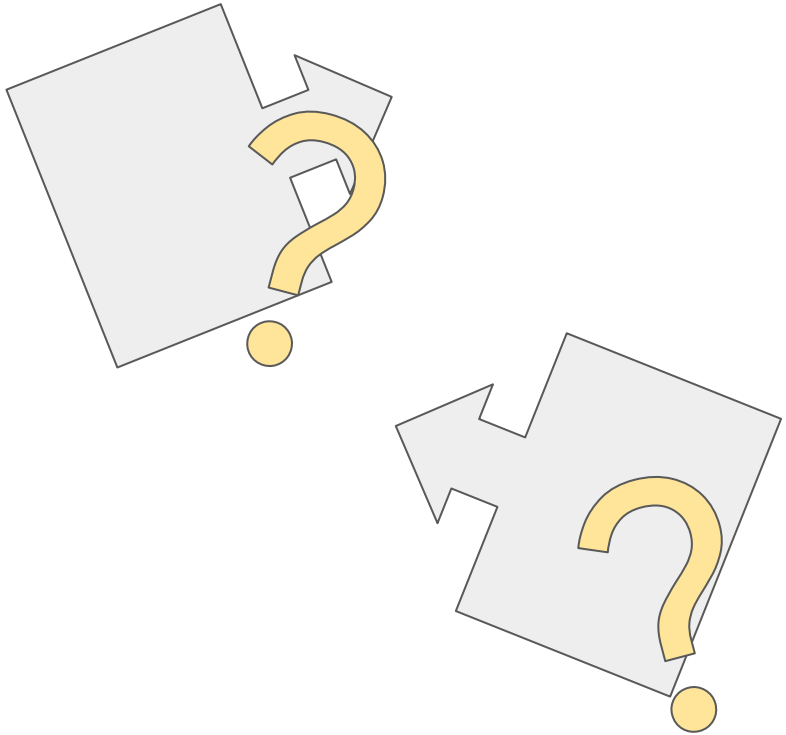
How does Babel ensure safe rollouts?

While transformation details depend on the network change...

...Babel guarantees that a given change can be rolled out safely...

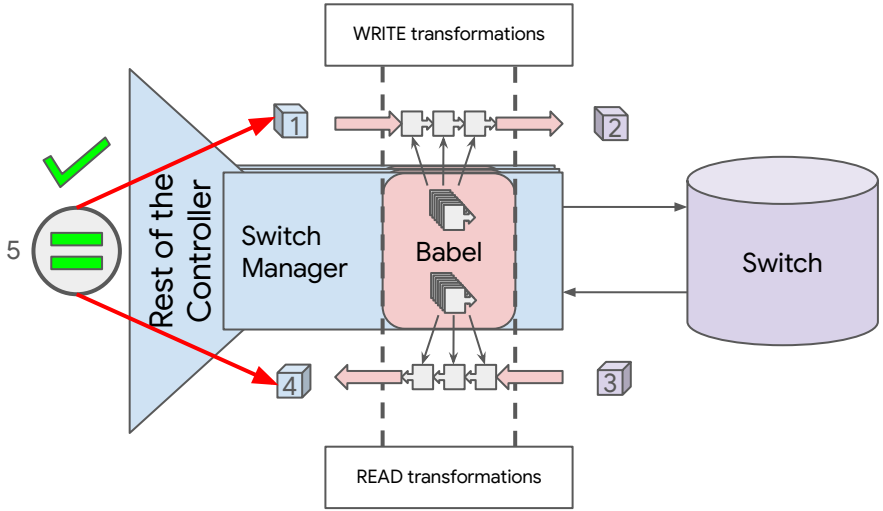
...by requiring that transformations satisfy 4 **sufficient conditions***

Babel enforces the conditions using property-based testing



***** Not yet formally proven

Example Condition: Controller Roundtrip Property



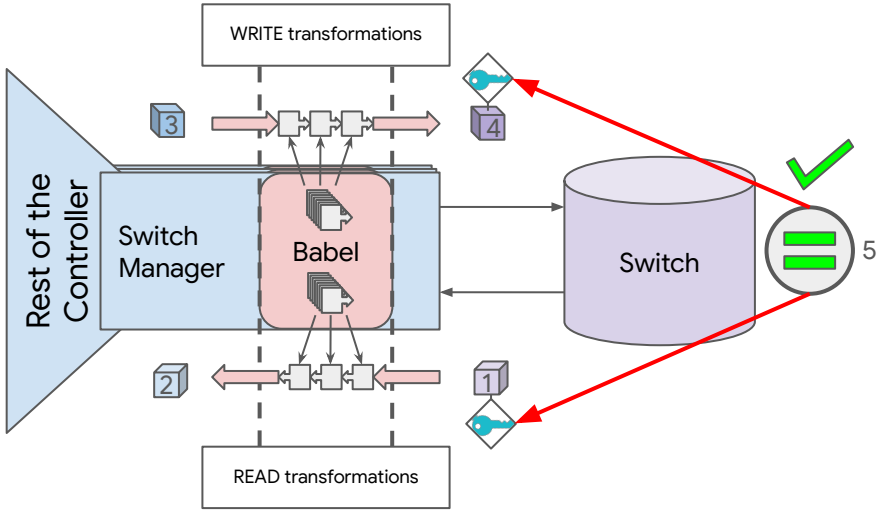
ReadTrans(WriteTrans(entry))
== entry

Property:

1. Entries written by the controller
2. Going to the switch (via Babel)
3. That are later read back
4. To the controller (via Babel)
5. Produce the **same entries** we started with

Preserves READ-WRITE symmetry provided by switch

Example Condition: Switch Roundtrip Property



Key(WriteTrans(ReadTrans(entry))
== Key(entry)

Property:

1. Entries read from the switch
2. Going to the controller (via Babel)
3. That are later written back
4. To the switch (via Babel)
5. Produce the **same keys** we started with

Ensures controller can MODIFY and DELETE entries on the switch

Thank You