BGP#: A System for Dynamic Route Control In Data Centers

Chao-Chih Chen

*UC Davis**

Lihua Yuan Albert Greenberg

Randy Kern Tao Zhang

Parantap Lahiri John Arnold Kevin Grady

Microsoft

**Also a Microsoft Intern*

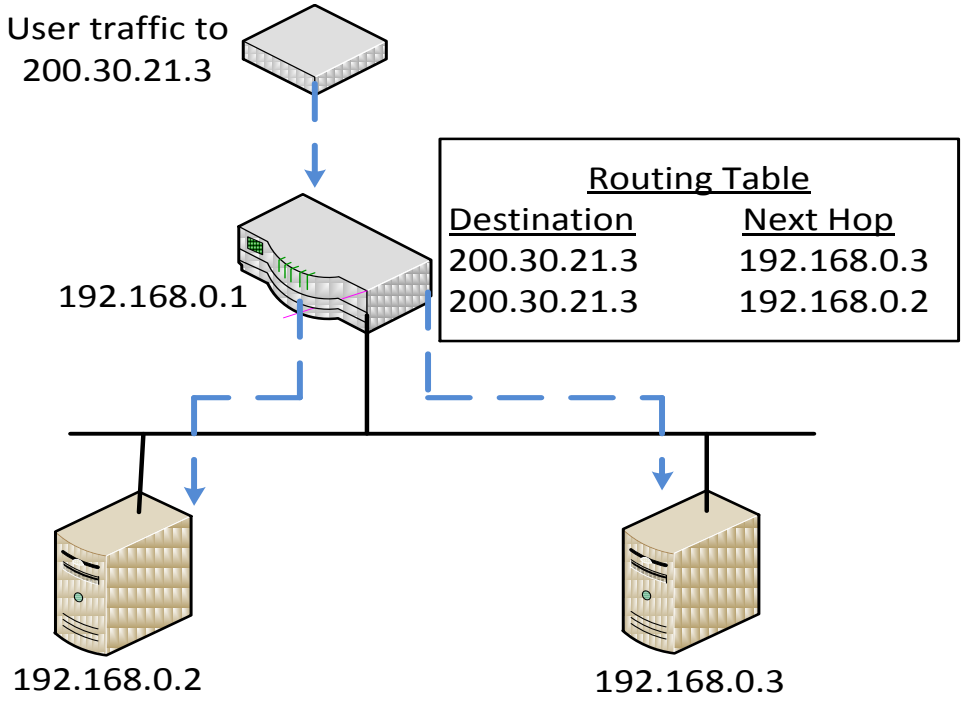
Data Centers – Tenants & Landlord

- One landlord
  - Owner and manager of the data centers
- Many tenants
  - Internal users
    - Search, email, online gaming, online office suites, etc..
  - External users
    - Utility computing customers, etc..
- Many challenges, this talk focuses on **empowering tenants with route control ability**

Routing Tensions

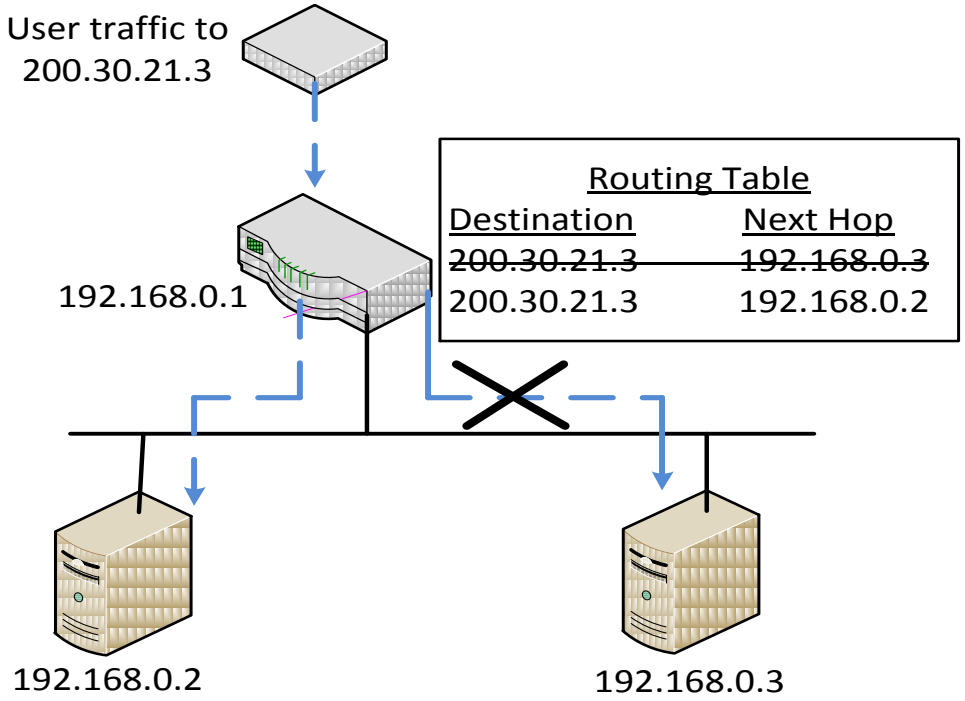
- Tenants have different goals
- But, tenants want to control their internal/external routes dynamically and on-demand
- Landlord manages shared infrastructure
  - Needs to empower users
  - Needs to control bad behavior
  - Needs to be scalable

Tenant Goal: Spread Traffic



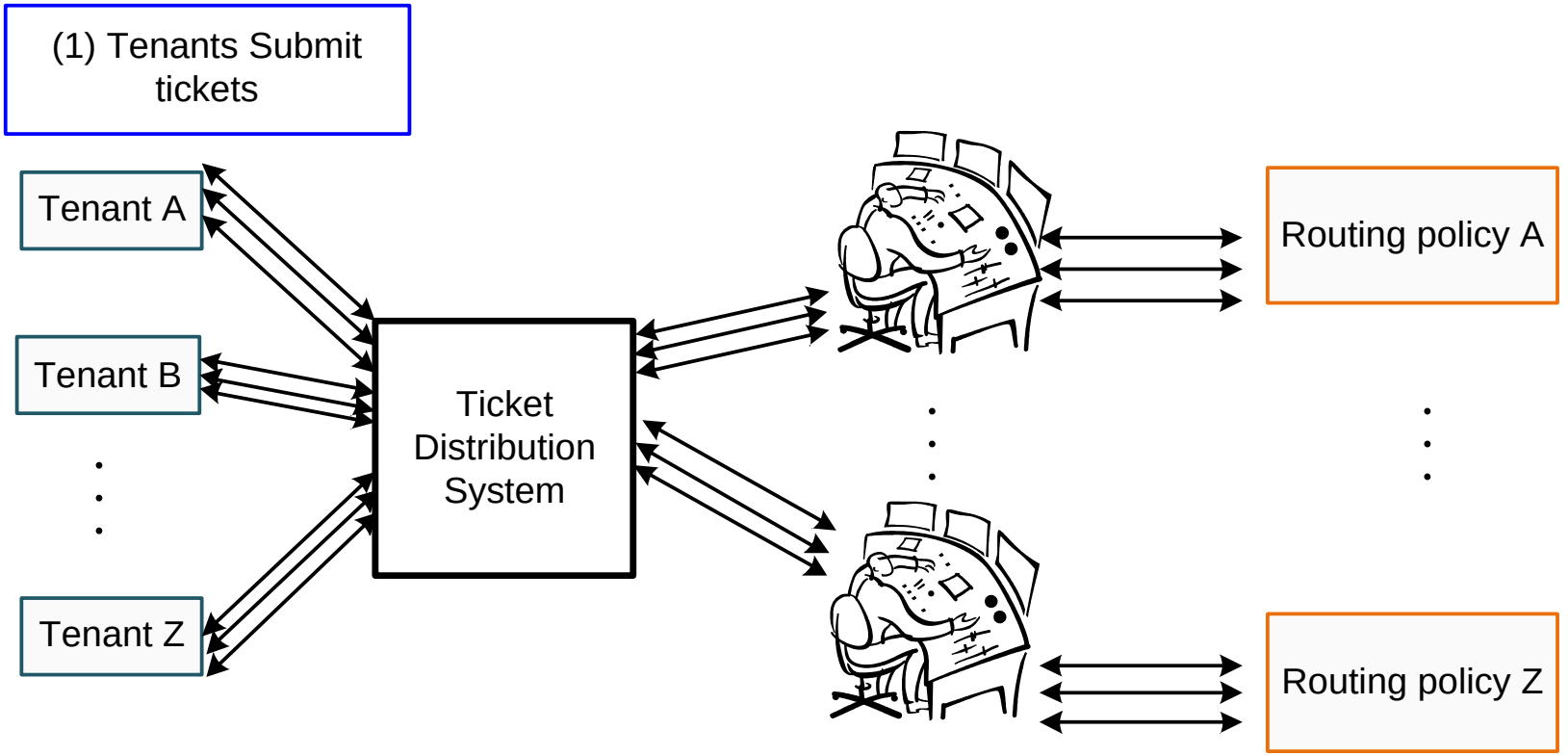
- Divide traffic between 192.168.0.2 and 192.168.0.3

Tenant Goal: Migrate Traffic

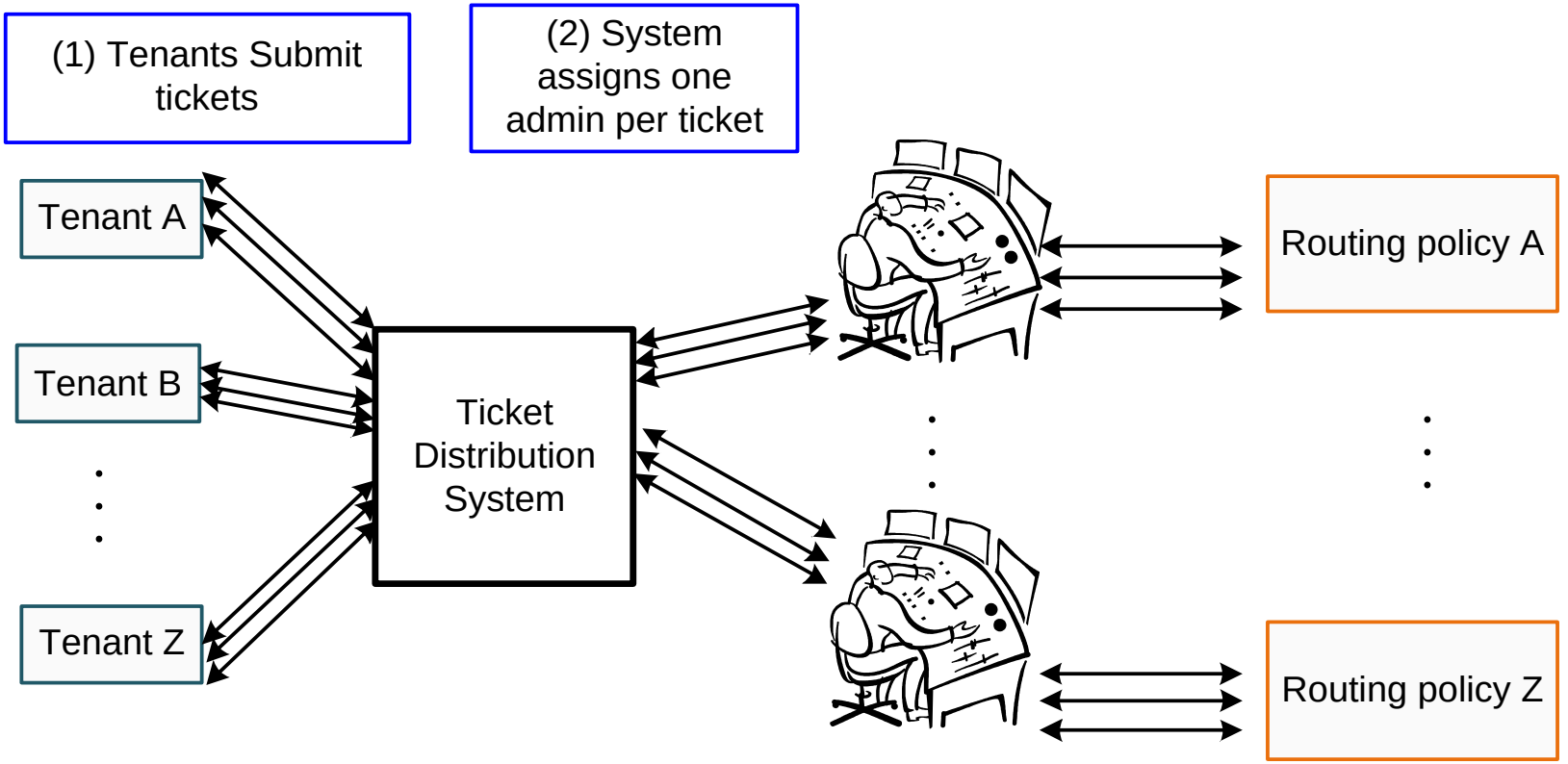


- Move traffic from 192.168.0.3 to 192.168.0.2

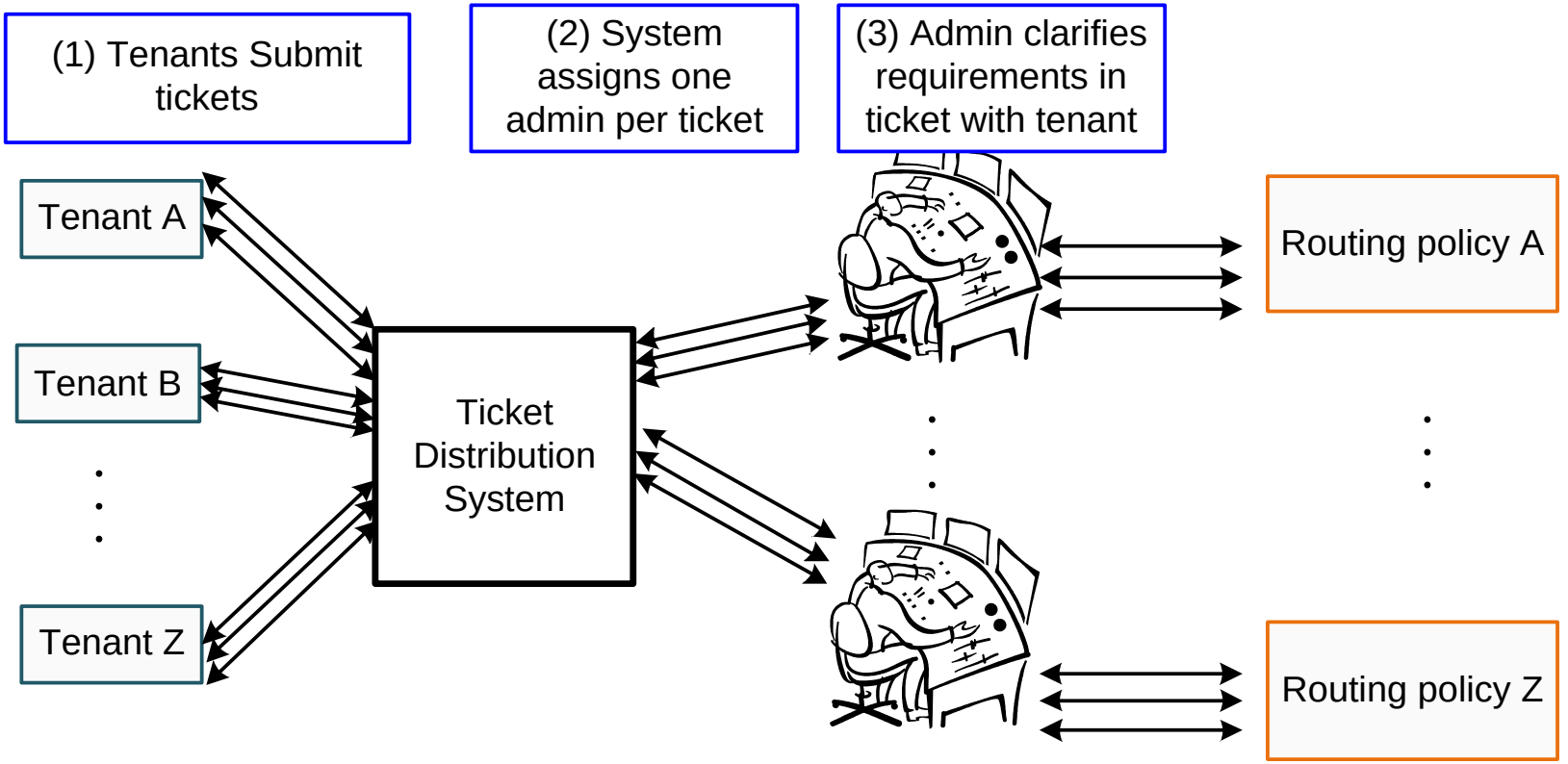
Route Control In Current Framework



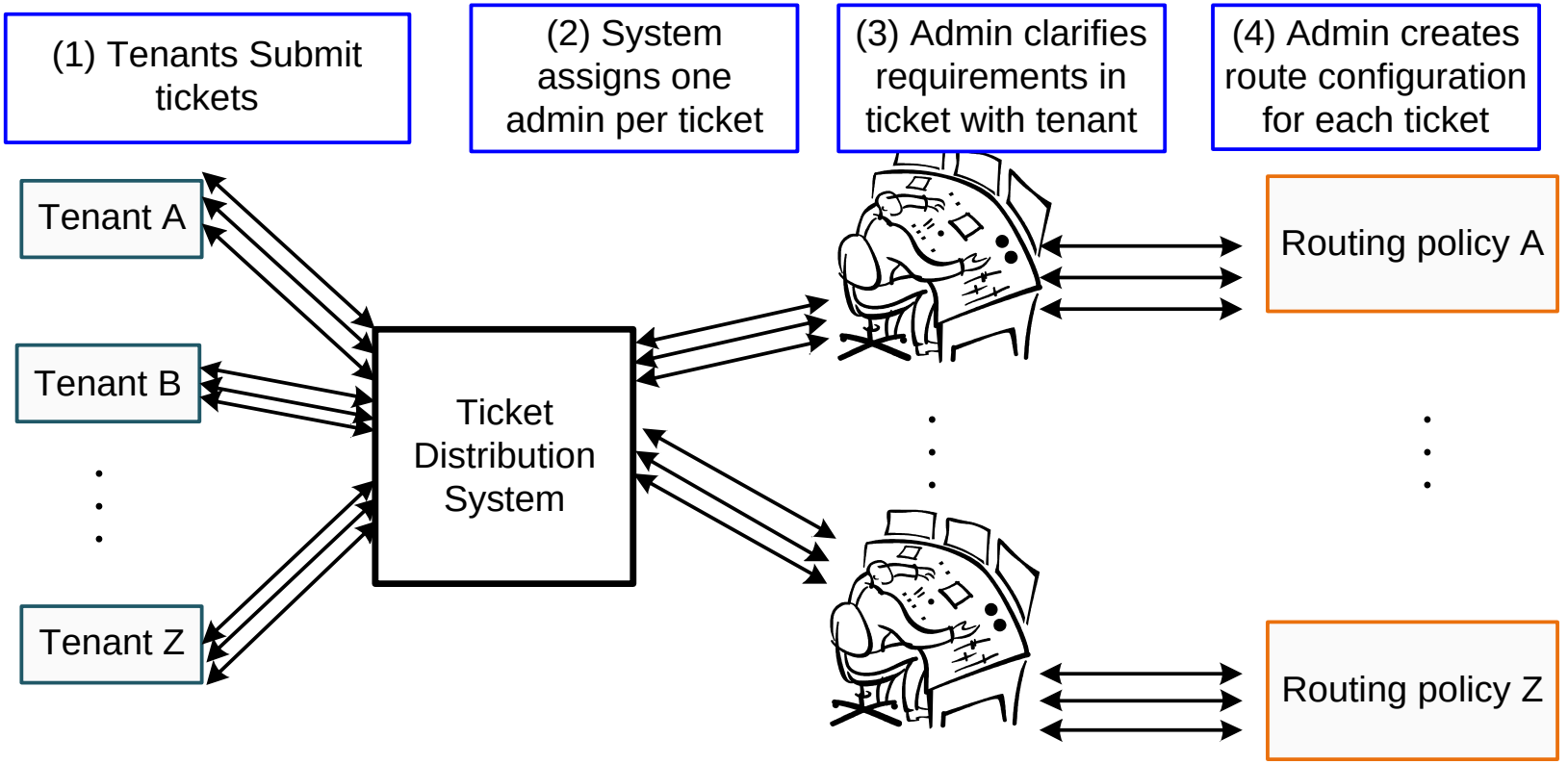
Route Control In Current Framework



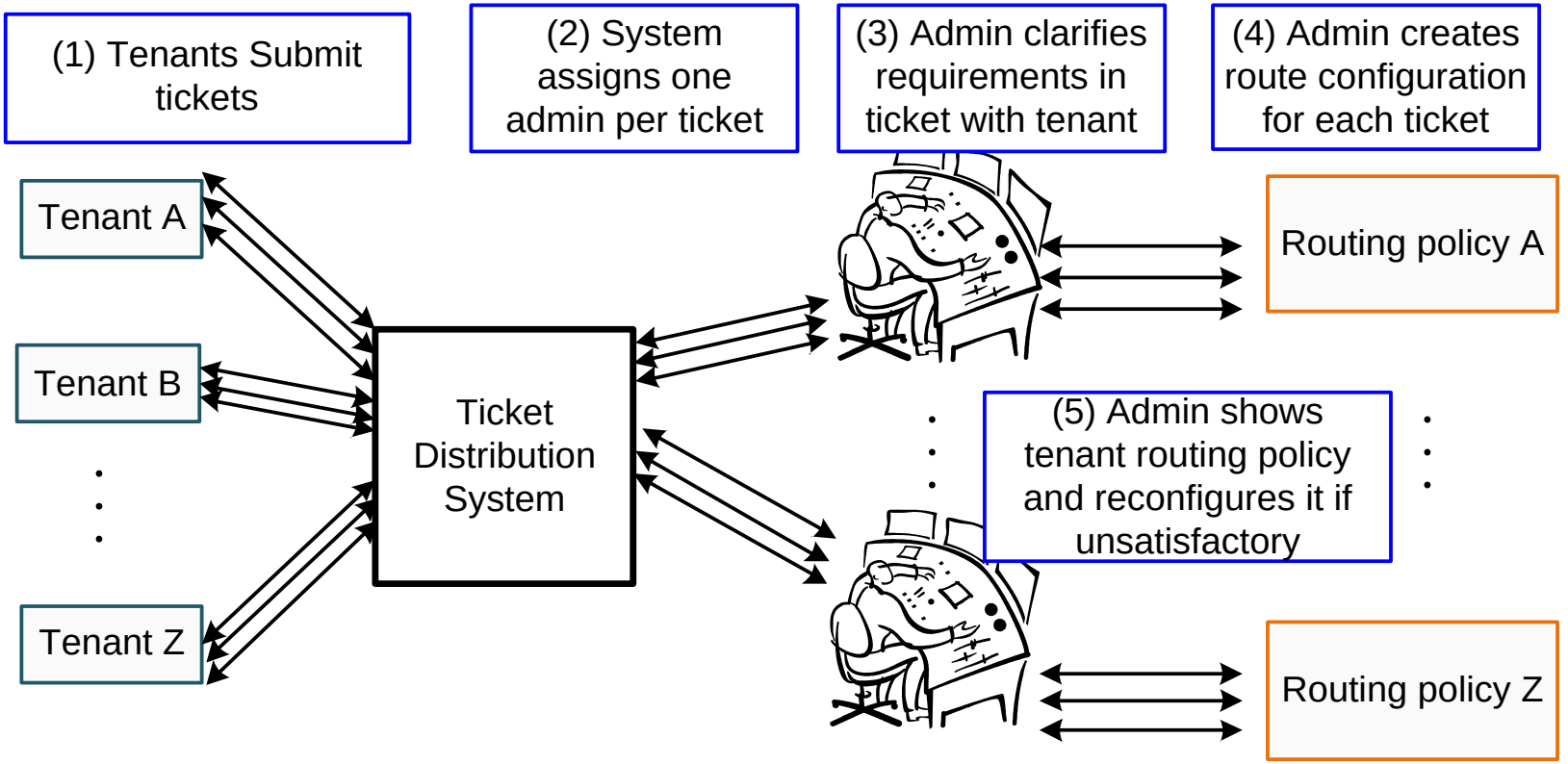
Route Control In Current Framework



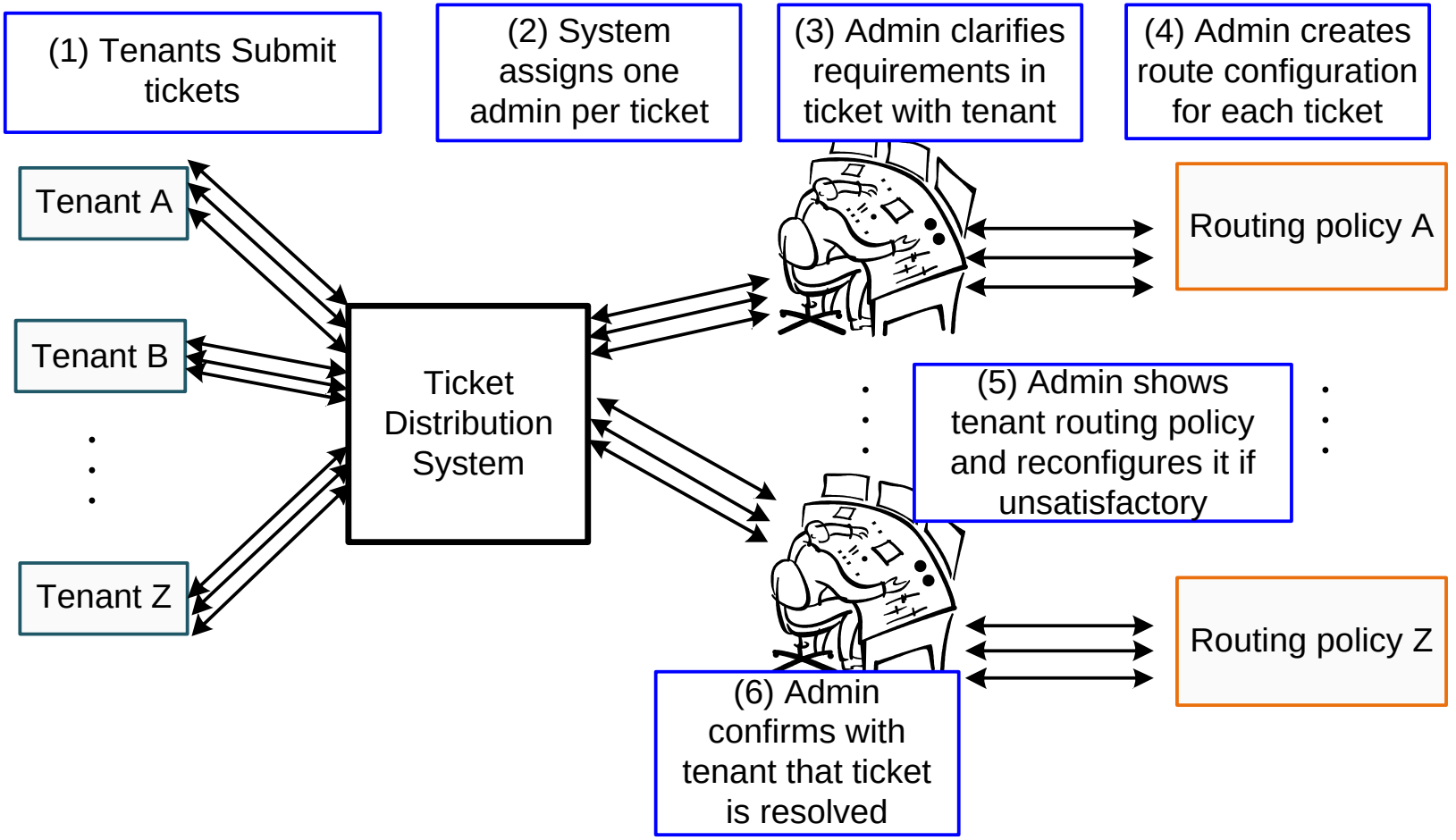
Route Control In Current Framework



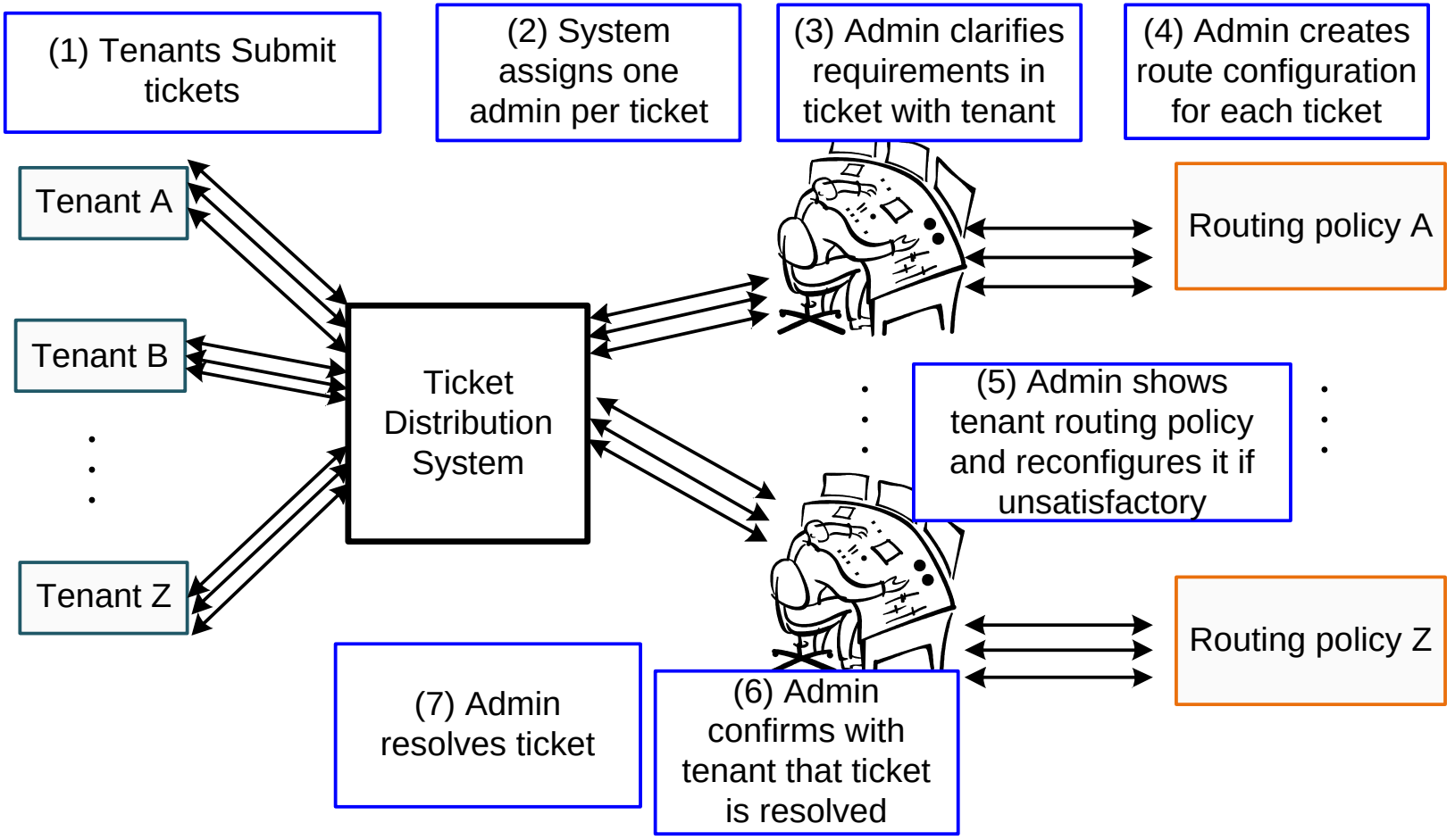
Route Control In Current Framework



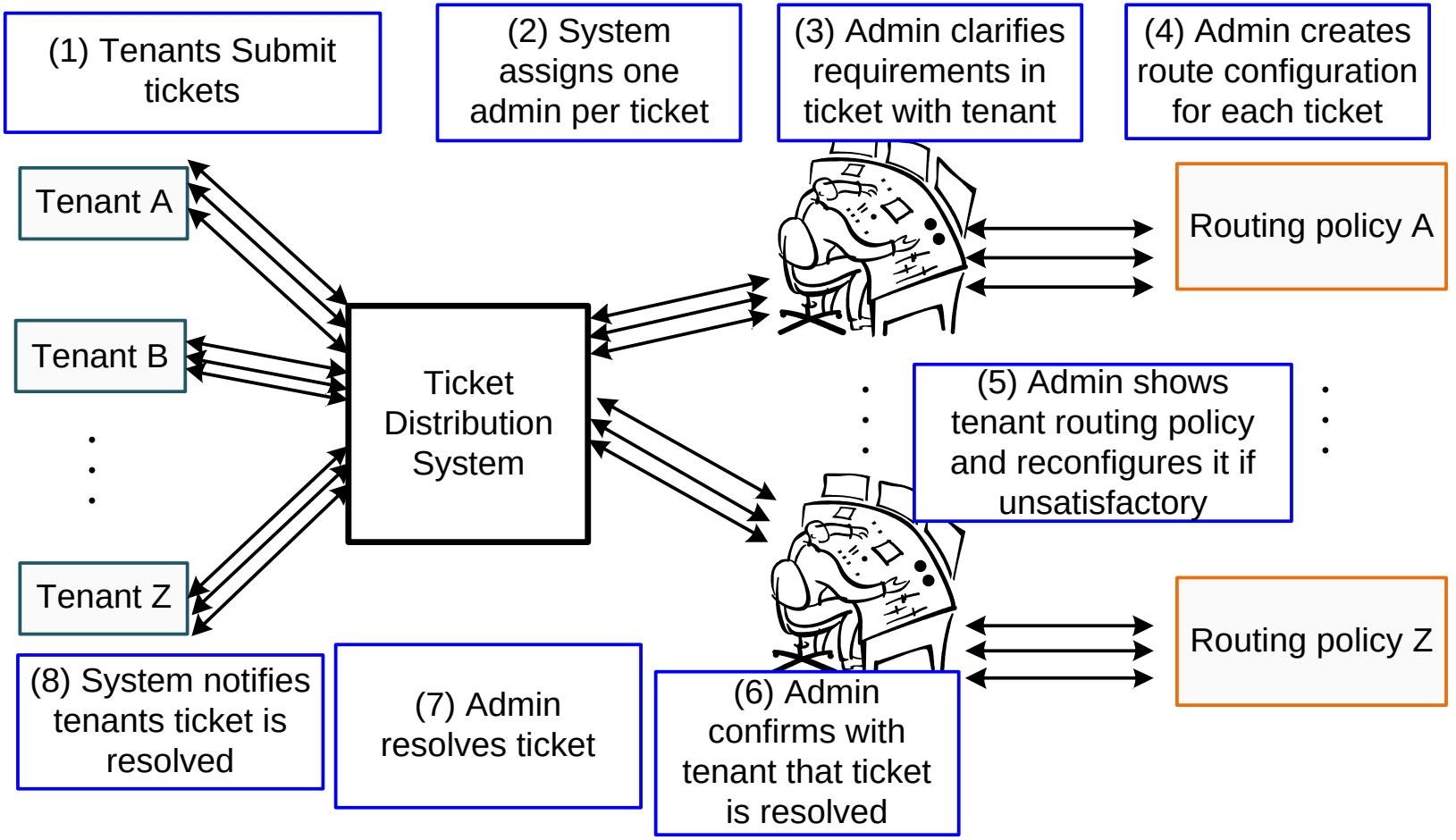
Route Control In Current Framework



Route Control In Current Framework

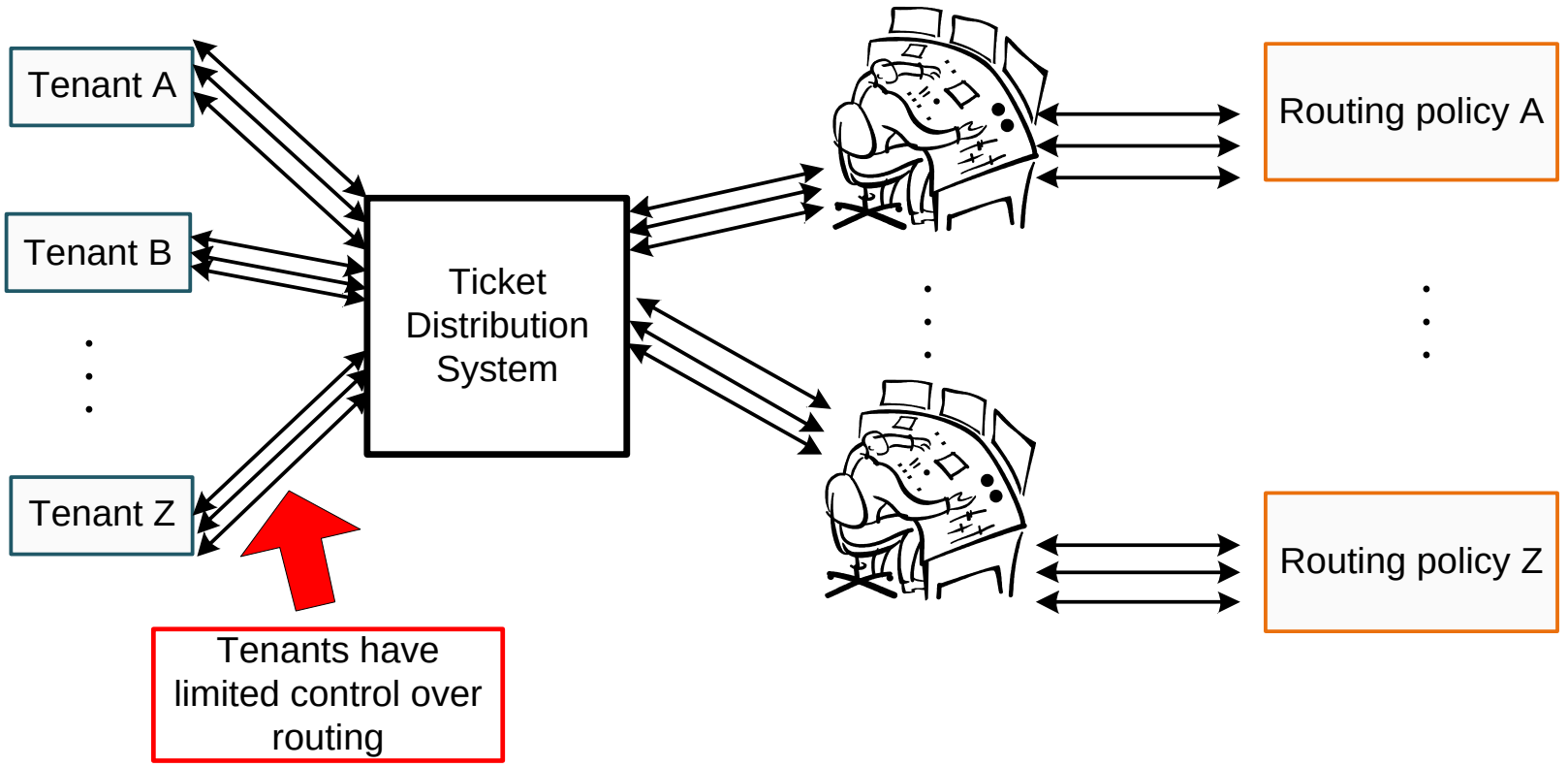


Route Control In Current Framework

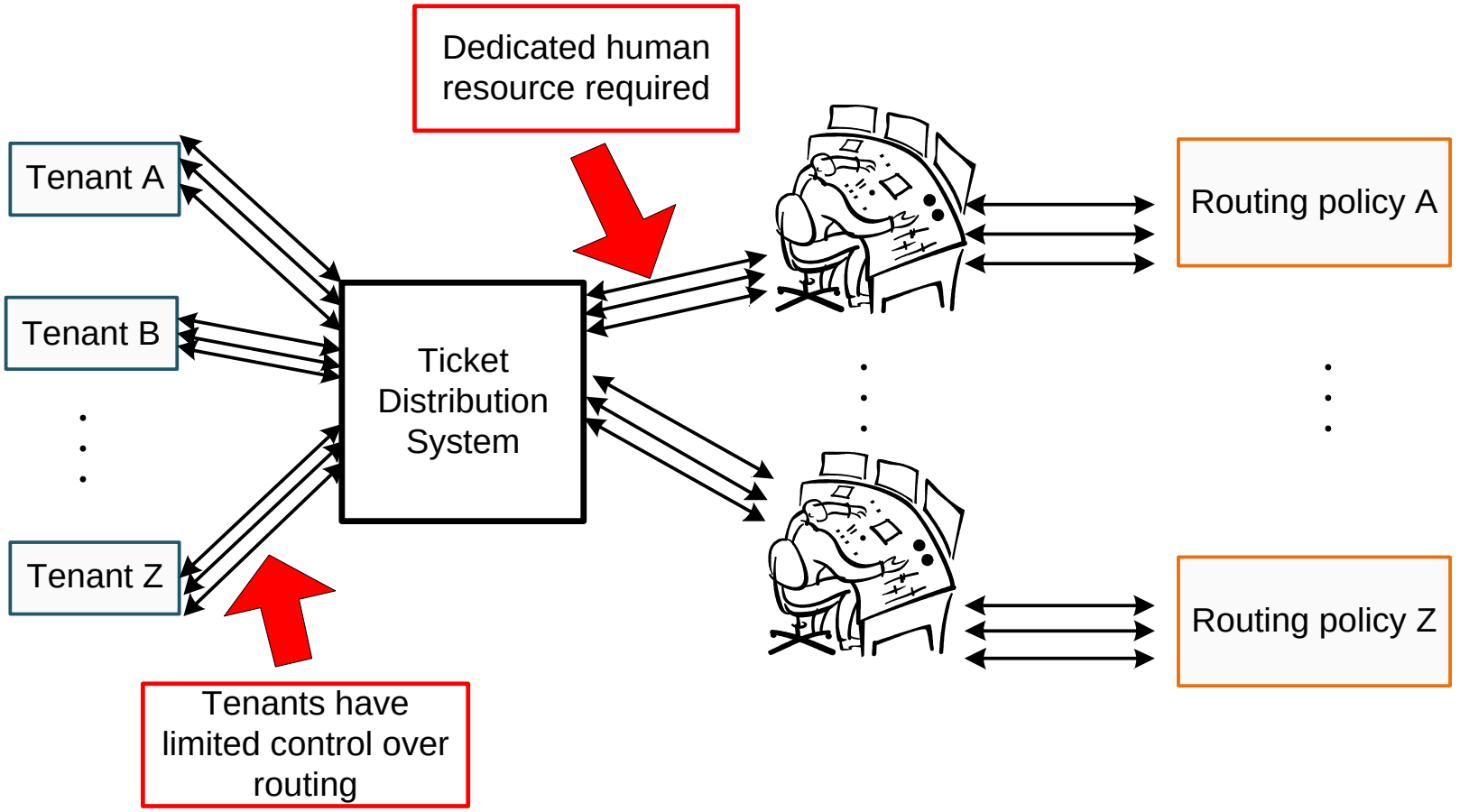


- Above processes are simplified

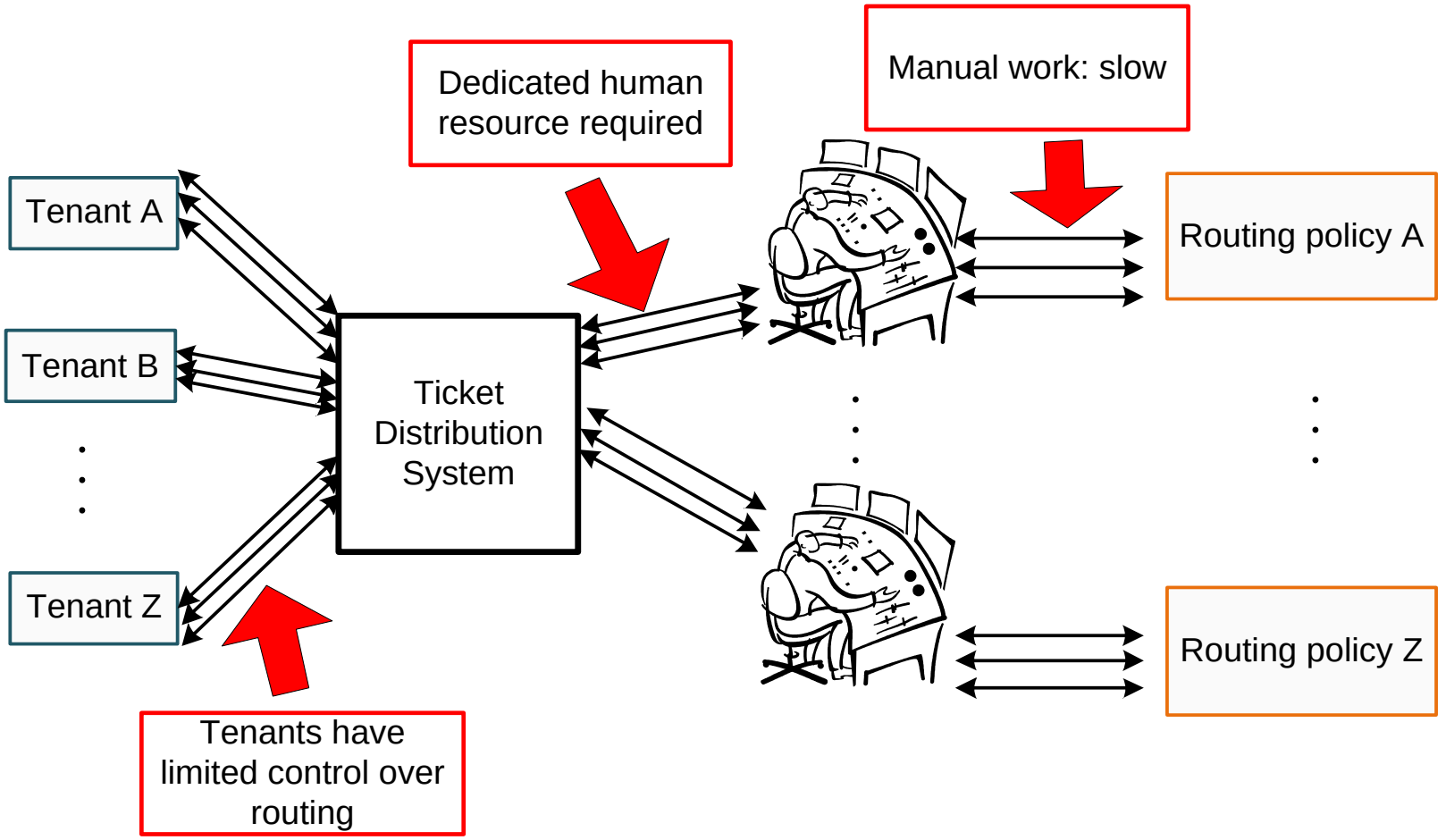
Problems in Today's Data Center Framework



Problems in Today's Data Center Framework



Problems in Today's Data Center Framework



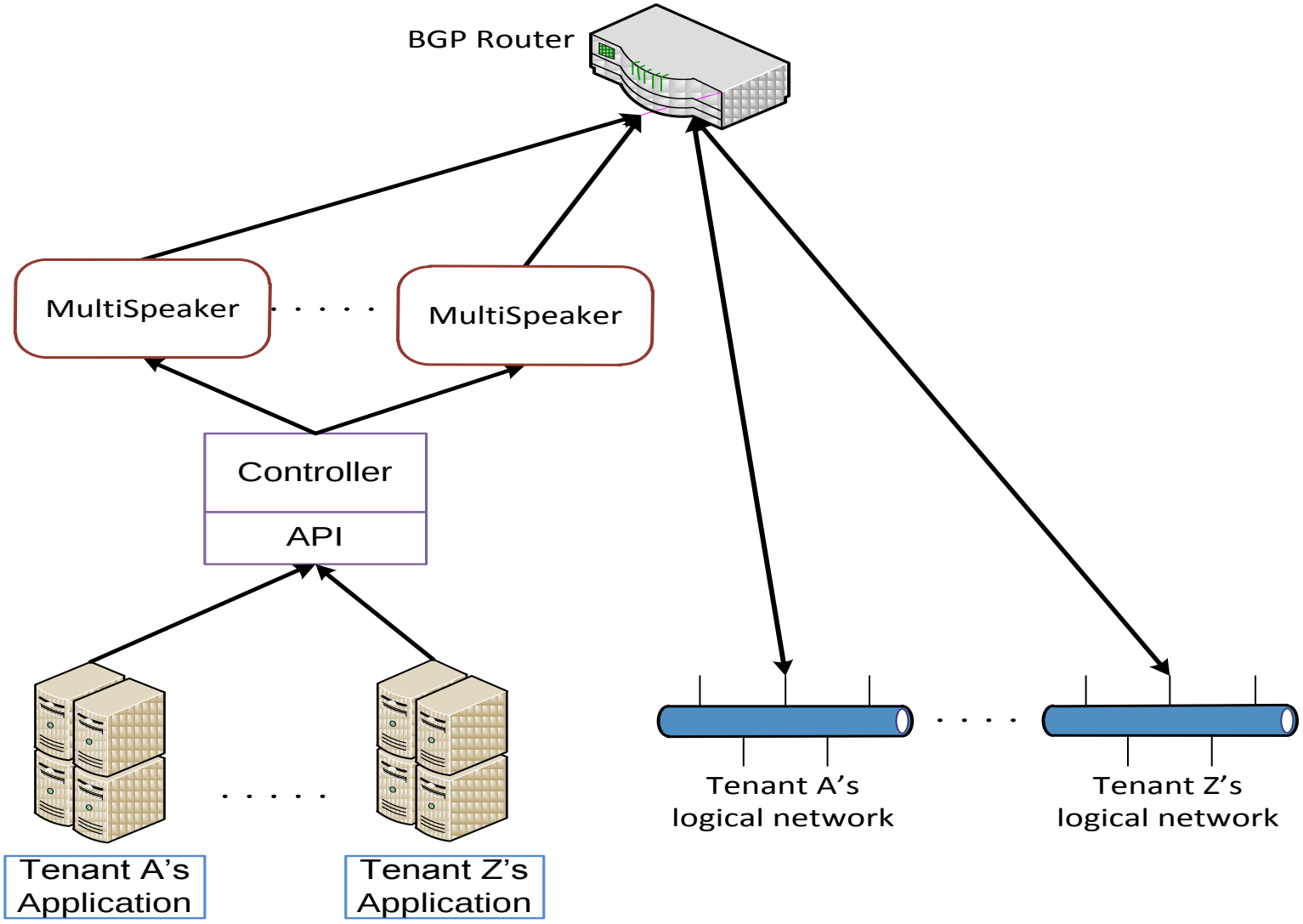
A Better System

- Allows for automated route control
  - Use application programming interfaces
- Allows tenants independent & safe route control
  - Support route validation
- Ensures better scalability
  - Factor out policy control for system scalability
  - Eliminate per-ticket manual intervention for human scalability
- Tolerates failures and planned maintenance
  - Deploy redundant components

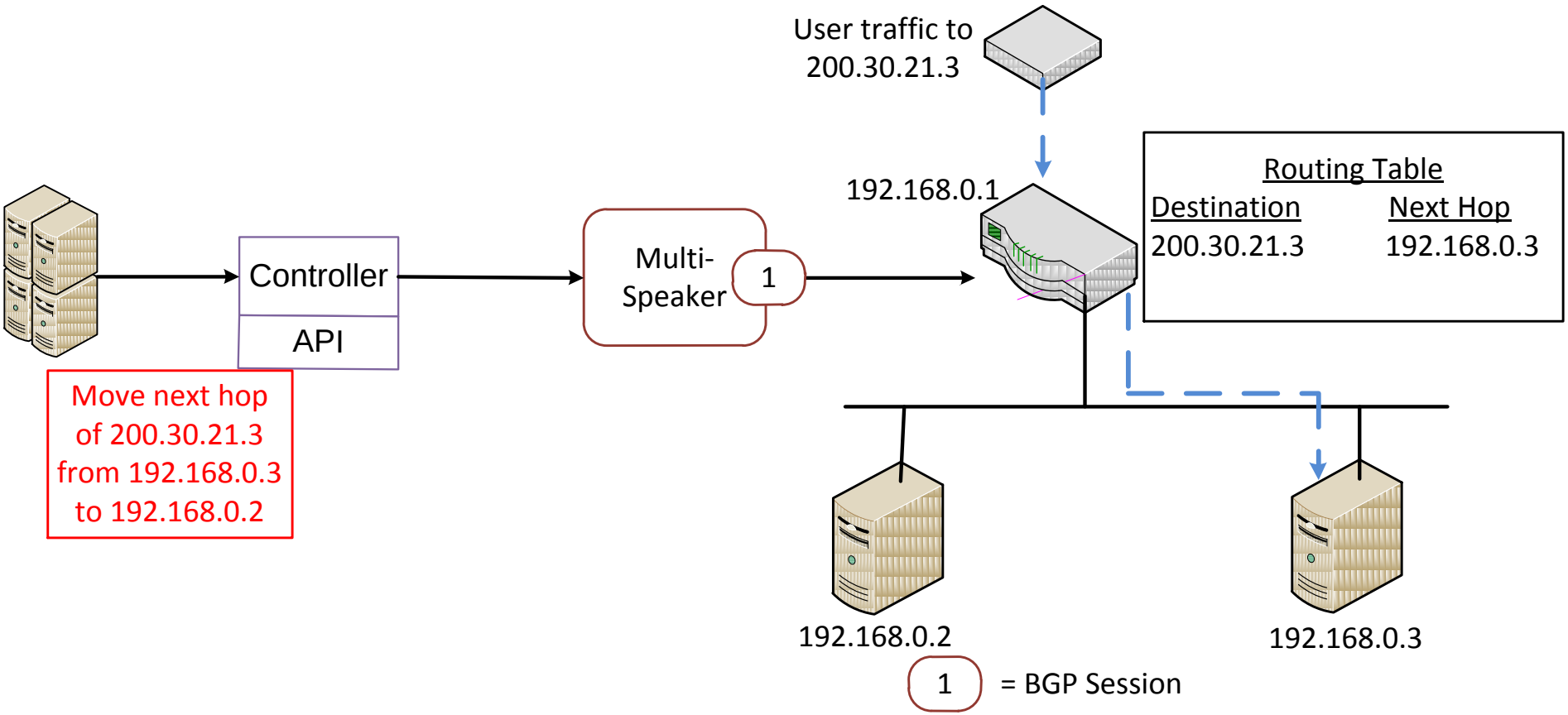
Solution: BGP#

- Simple speakers ("**MultiSpeaker**")
  - Peer with BGP routers
  - Send route announcements/withdrawals (ECMP-capable)
- Stateful controller ("**Controller**")
  - Controls and coordinates the speakers
  - Exposes API to tenants
- Custom client applications ("**Application**")
  - Discover services offered by controllers' API
  - Modify routing to tenants' network via controller's API

BGP# Architecture

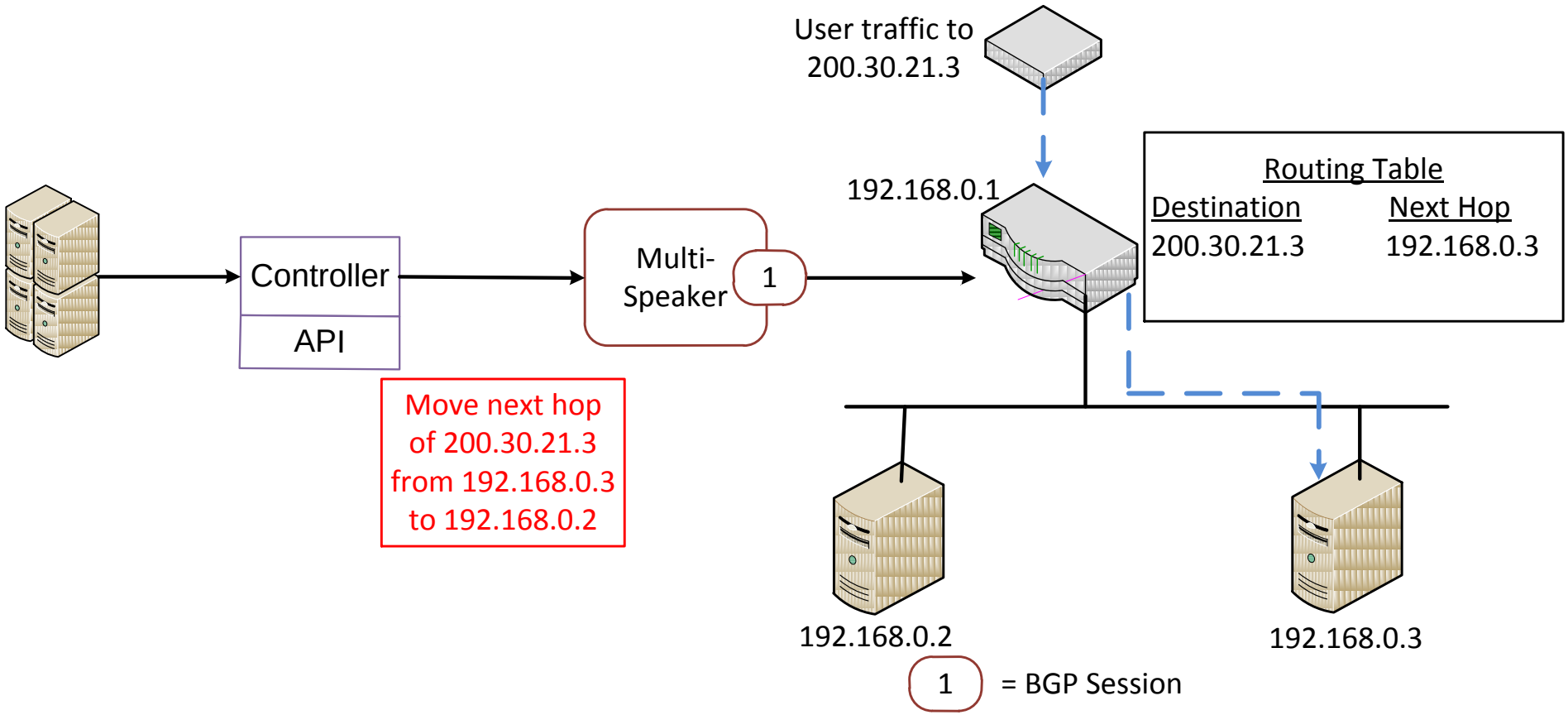


Using BGP# To Migrate Traffic



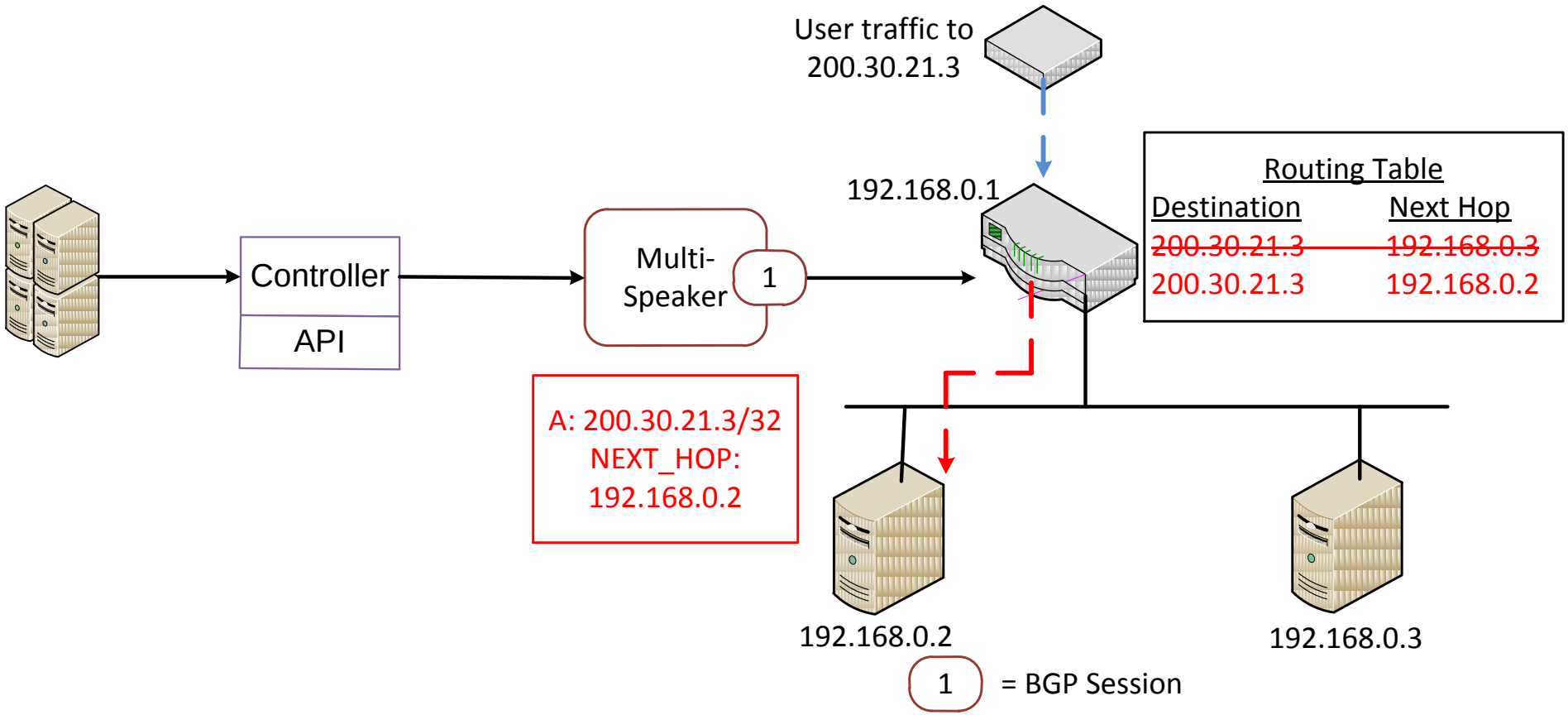
- Application issues route change request

Using BGP# To Migrate Traffic



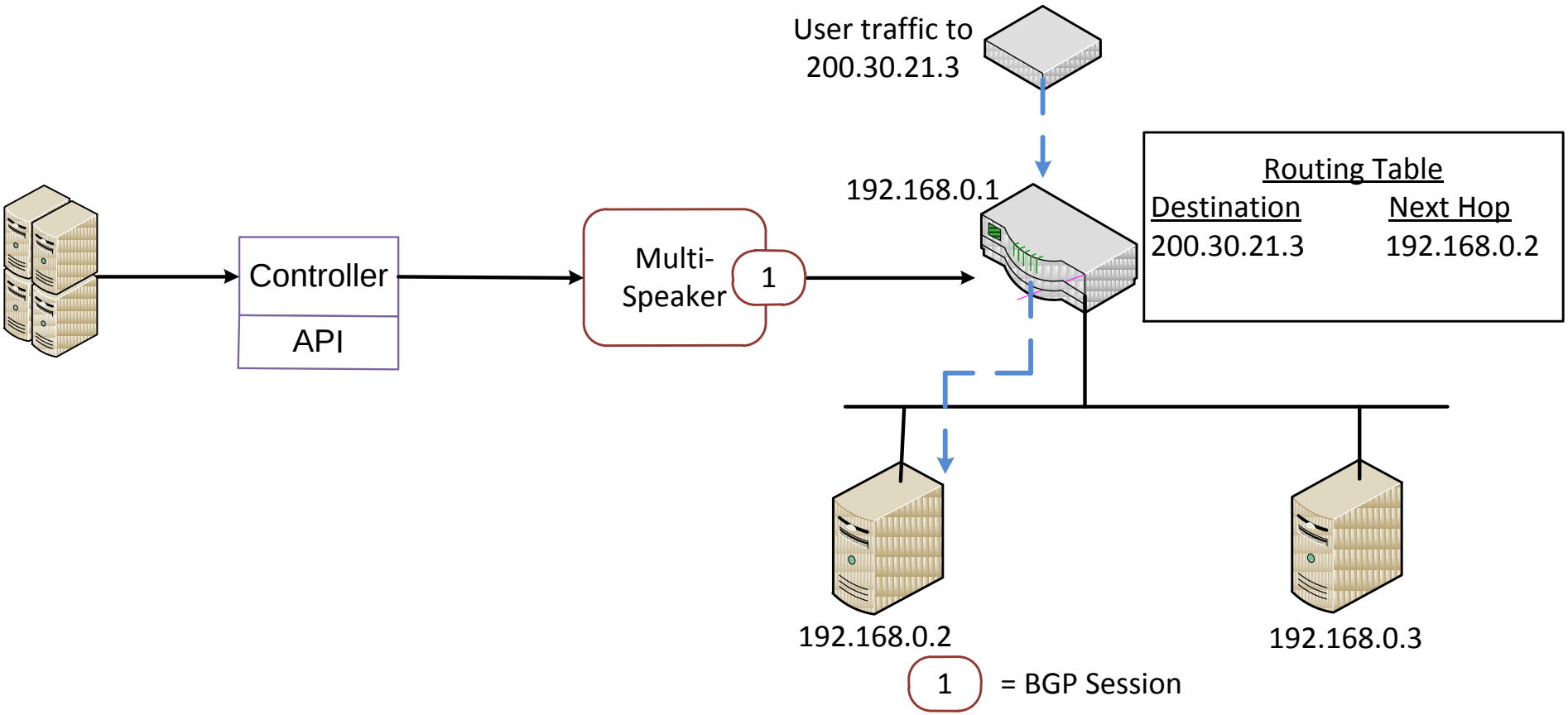
- Controller validates and forwards request

Using BGP# To Migrate Traffic



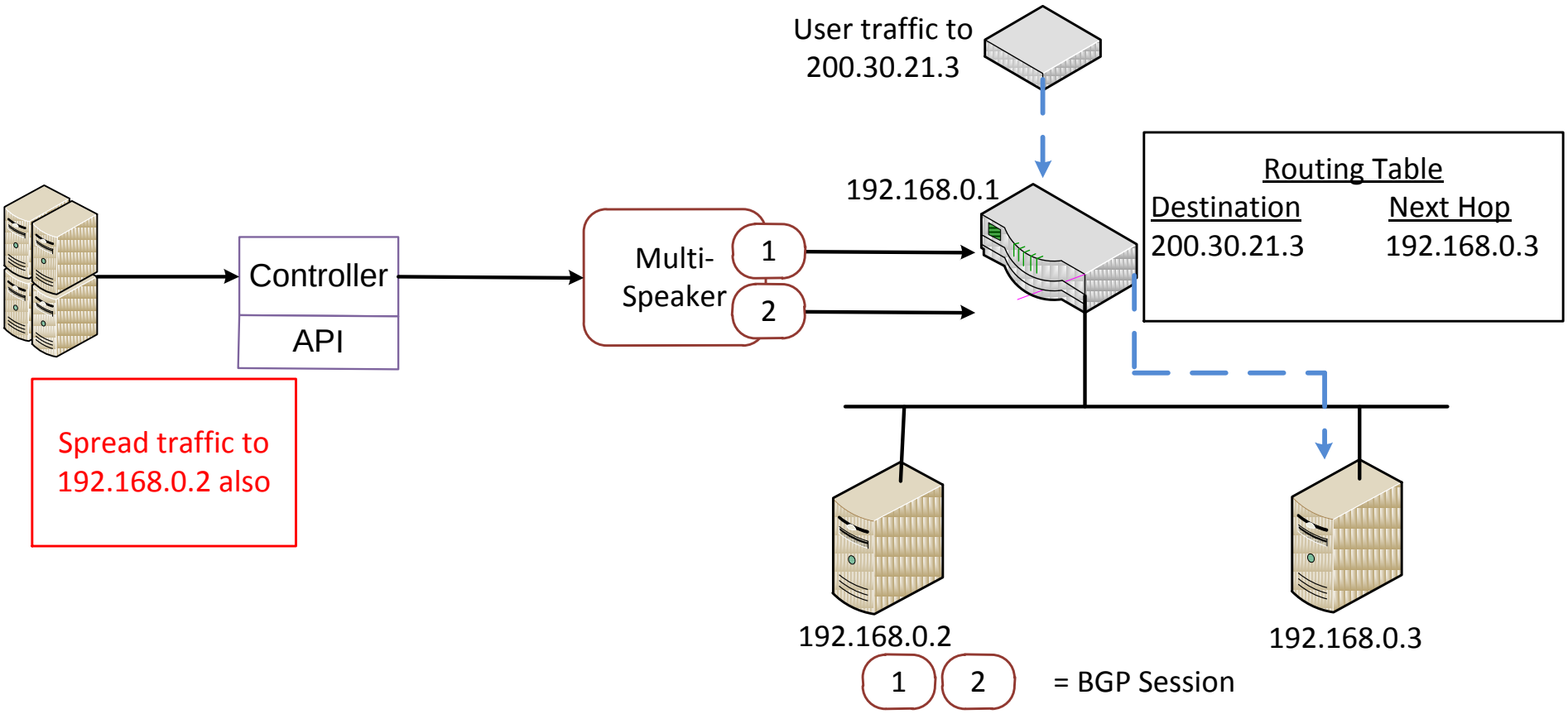
- MultiSpeaker transforms request into BGP message

Using BGP# To Migrate Traffic



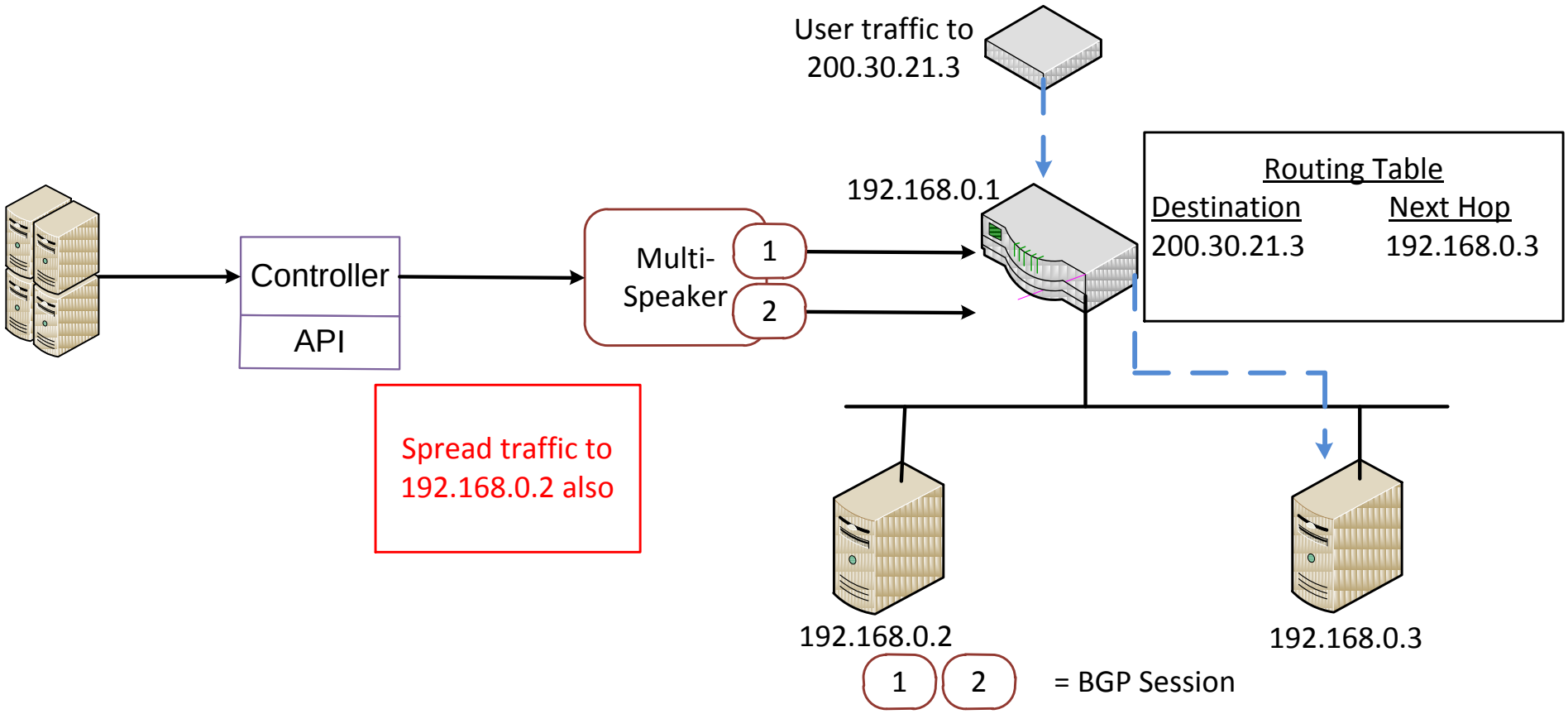
- MultiSpeaker needs to have announced the original route for the migration to succeed

Using BGP# To Spread Traffic



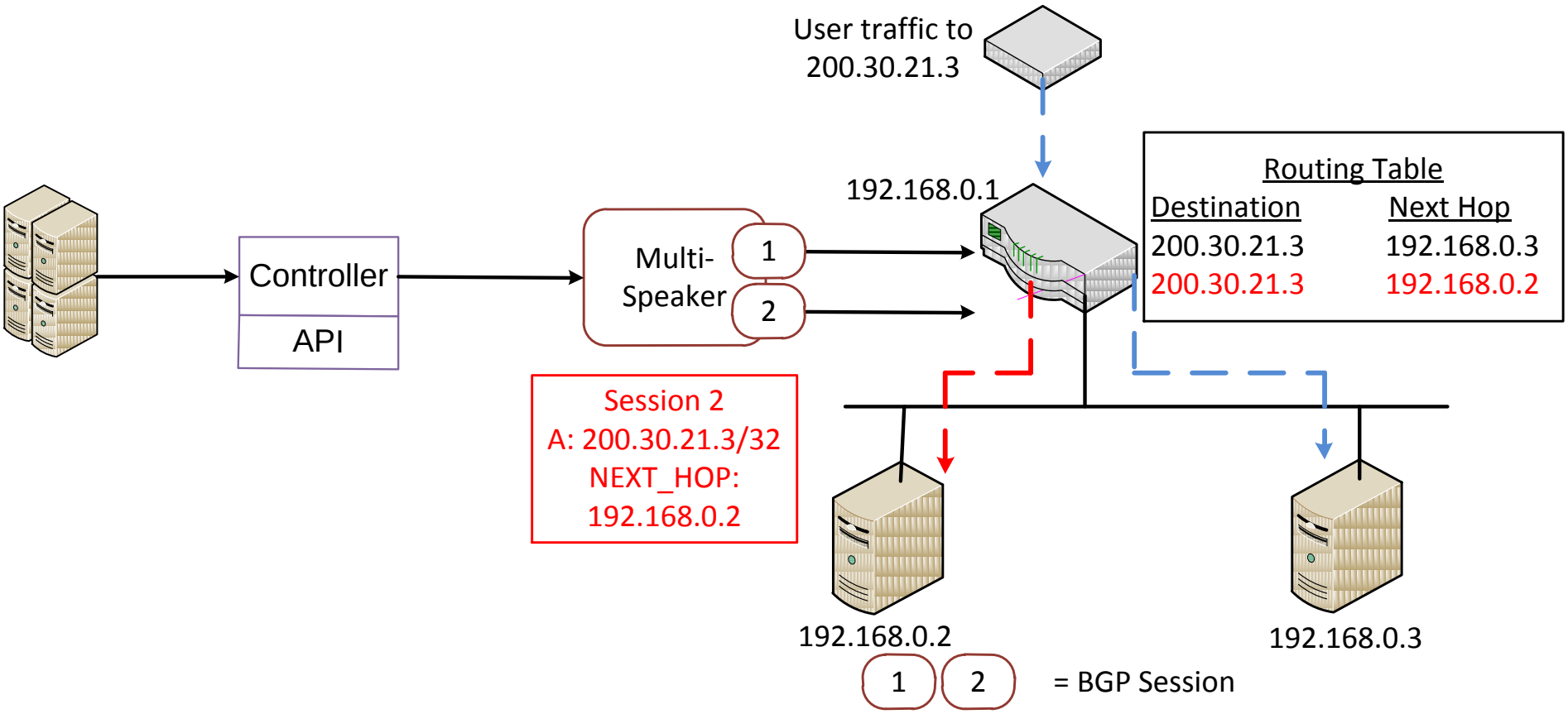
- Session 1 of MultiSpeaker announced existing route
- Router enabled ECMP

Using BGP# To Spread Traffic



- Controller validates and forwards request

Using BGP# To Spread Traffic



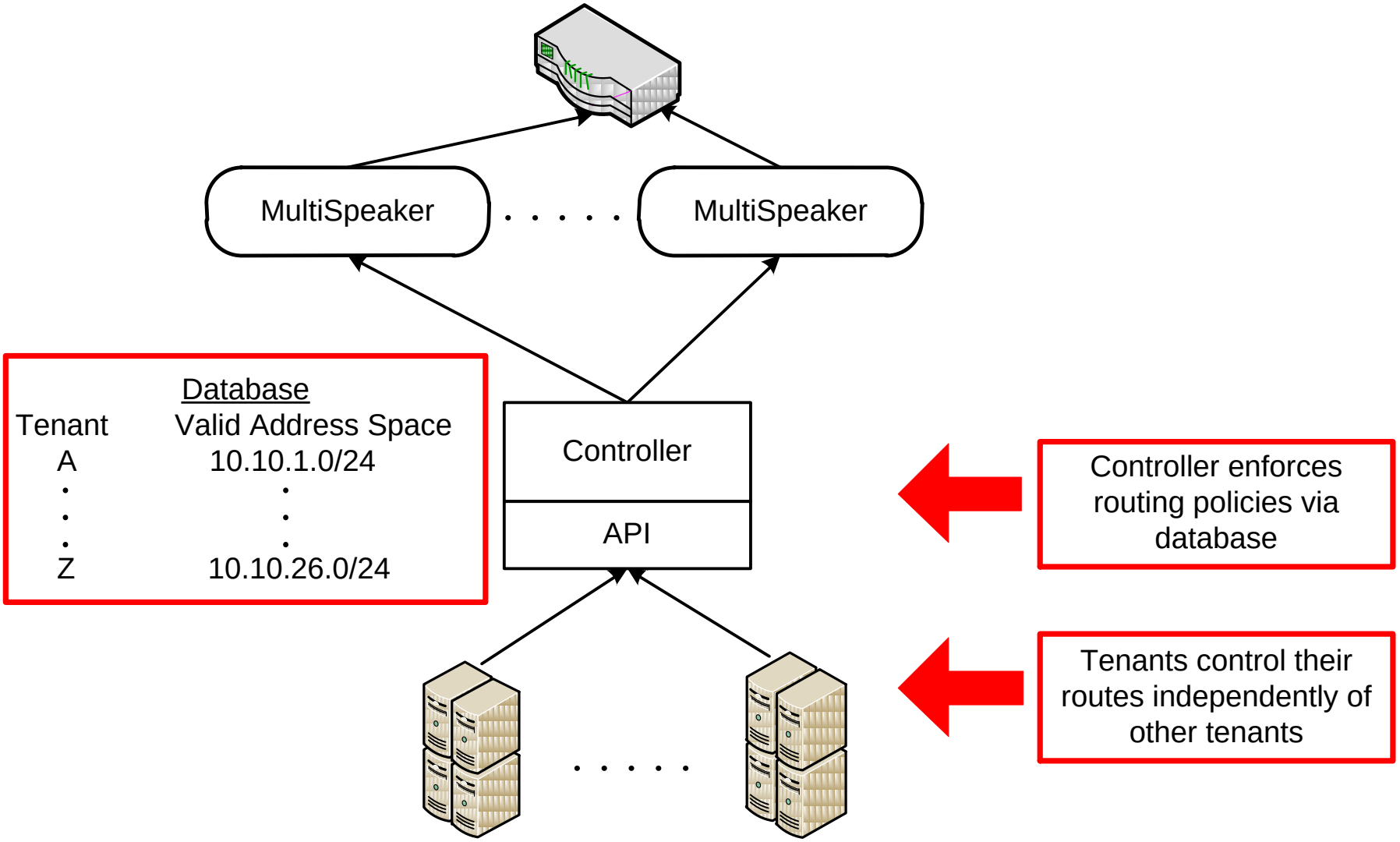
- MultiSpeaker transforms request into BGP message

Automated Route Control

- Controller API allows for custom applications
- Application can automatically manage routes to meet tenant's goals
  - Validated to manipulate only tenant's routes

Example	
Goal	Route Control Program Behavior
Fast server failover	Replace dead server IP with live server IP
High throughput	Replace IP of servers having heavy link utilization with IP of servers having light link utilization

Independent and Safe Route Control



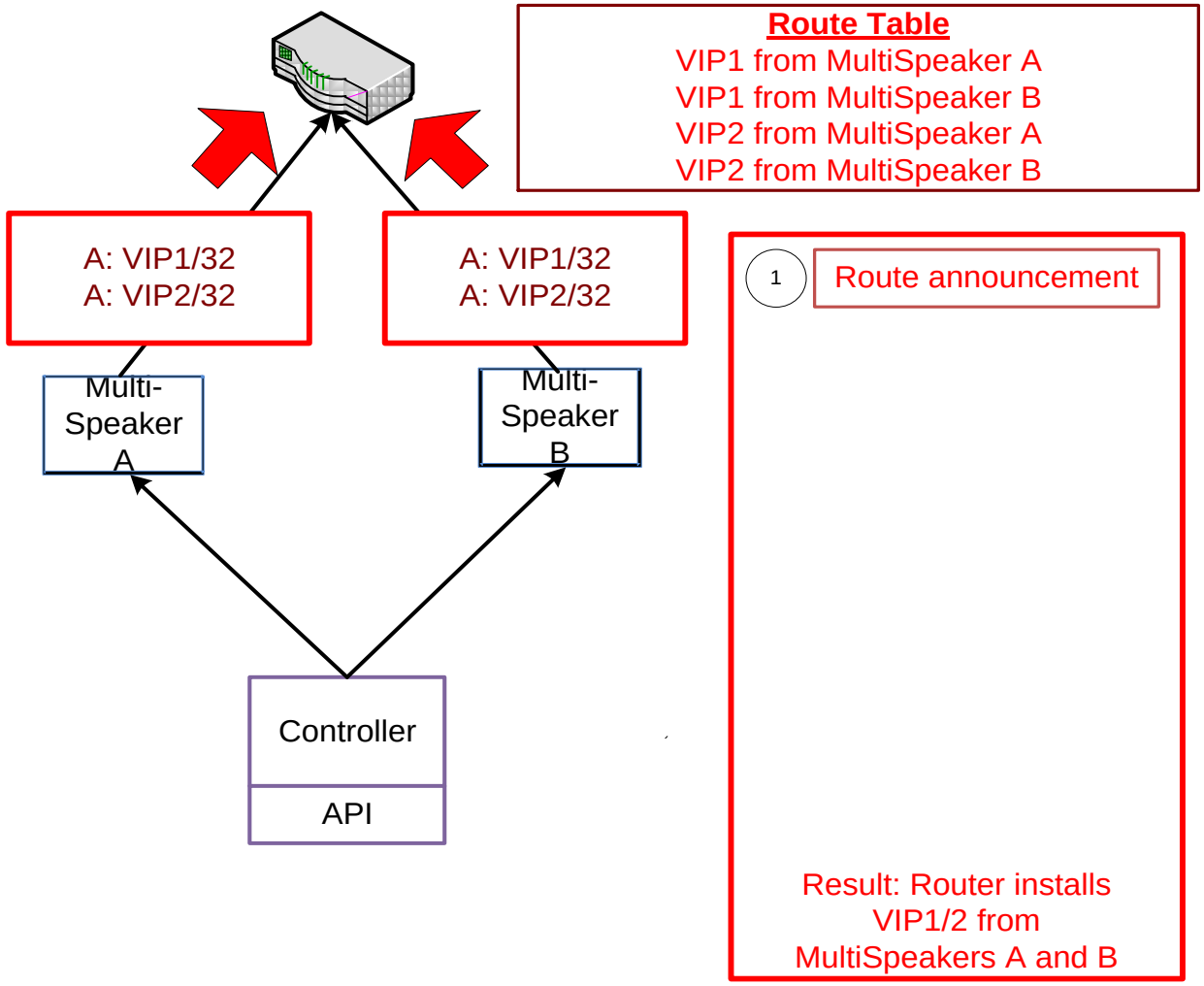
Scalability

- Factor out policy control
  - MultiSpeakers and Controller are not placed in machines handling user traffic
    - Eliminates need for one policy controller per machine
    - Reduces peering sessions to router
- Eliminate per-ticket manual intervention
  - Policy enforced at Controller
    - Guarantees tenant routing behaviors are isolated from others

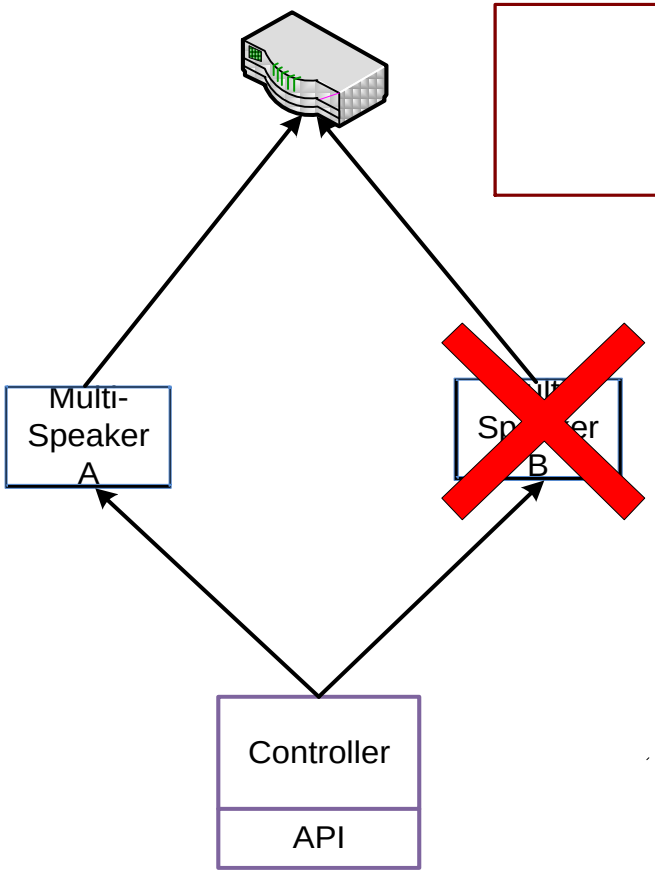
Resiliency

- **System resiliency**: Ensure system continues operating
  - Instantiate multiple MultiSpeakers
    - Single MultiSpeaker failure does not affect other MultiSpeakers' availability
  - Separate MultiSpeakers and Controller
    - Controller failure does not affect MultiSpeakers' availability
- **Prefix resiliency**: Ensure prefix stays available
  - Announce the same prefixes from multiple MultiSpeakers
    - Router retains prefix as long as one MultiSpeaker is alive
  - Separate MultiSpeakers and Controller

Example – Prefix Resiliency



Example – Prefix Resiliency



Route Table

VIP1 from MultiSpeaker A

~~VIP1 from MultiSpeaker B~~

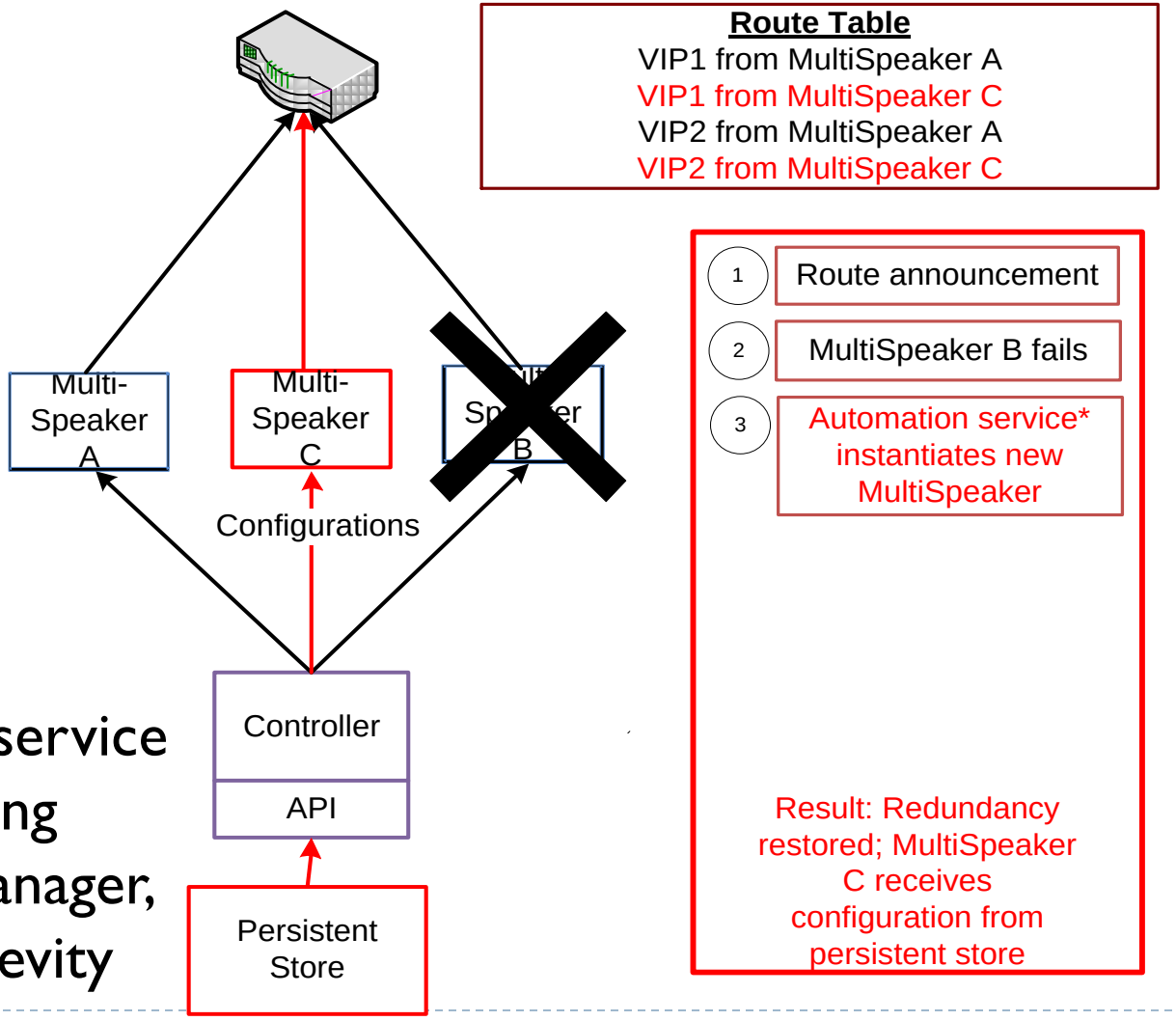
VIP2 from MultiSpeaker A

~~VIP2 from MultiSpeaker B~~

1. Route announcement
2. MultiSpeaker B fails

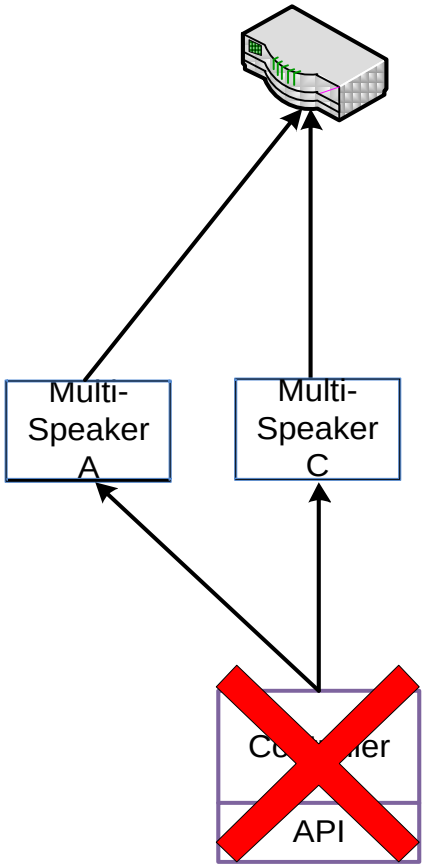
Result: Prefixes VIP1/2 remains available

Example – Prefix Resiliency



* Automation service is an overarching data center manager, omitted for brevity

Example – Prefix Resiliency

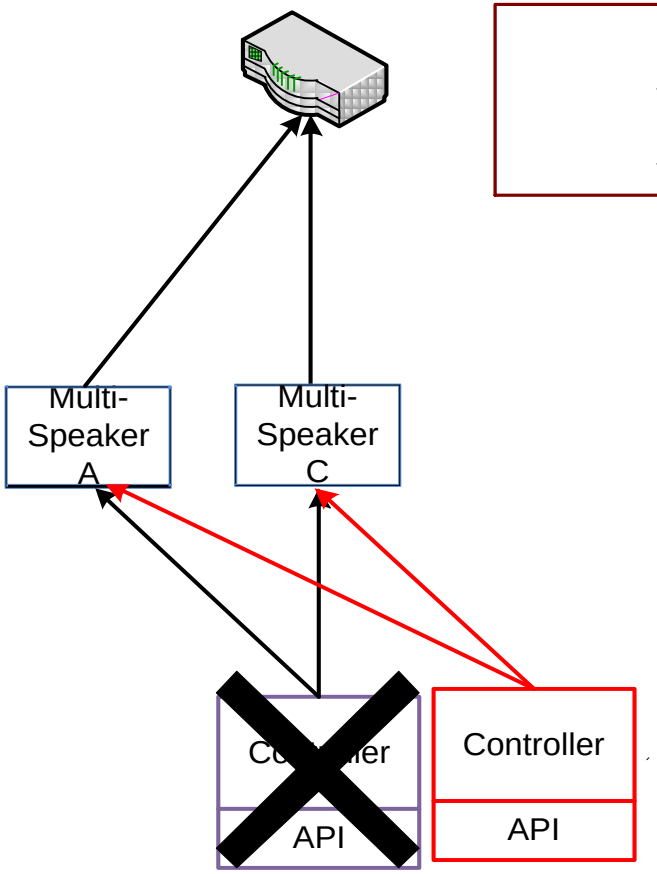


Route Table
VIP1 from MultiSpeaker A
VIP1 from MultiSpeaker C
VIP2 from MultiSpeaker A
VIP2 from MultiSpeaker C

1. Route announcement
2. MultiSpeaker B fails
3. Automation service instantiates new MultiSpeaker
4. Controller fails, MultiSpeaker A & C maintain peering

End Result: Prefixes VIP1/2 unaffected by controller failure

Example – Prefix Resiliency



Route Table

VIP1 from MultiSpeaker A
VIP1 from MultiSpeaker C
VIP2 from MultiSpeaker A
VIP2 from MultiSpeaker C

1. Route announcement
2. MultiSpeaker B fails
3. Automation service instantiates new MultiSpeaker
4. Controller fails, MultiSpeaker A & C maintain peering
5. Automation servce instantiates new controller

End Result: New controller establishes session with MultiSpeakers

No Inconsistency With Multiple MultiSpeakers

- Suppose some MultiSpeakers become unresponsive
  - BGP# listening tool detects the lack of router re-advertisement
- Suppose MultiSpeaker reboots and is in different state than other MultiSpeakers
  - Obtain current configuration file from persistent store

Alternate Approach?

- Each tenant sets up its own BGP instances
  - Tenants need to implement one BGP instance per machine
    - Ticket system dependency
    - Delayed BGP instance operation
  - Landlord needs to deal with many BGP peers
    - Manual configuration
    - Dedicated human resource
    - Increased complexity

Conclusions

- Tenants have more power
  - API makes it possible for tenants to perform automated route control
- Landlord retains responsibility of validation
  - Controller provides centralized control point
- System achieves scalability and resiliency
  - Distributed components ensure near zero-impact on single point of failure

- DEMO available after talk – find me if interested!