

Re-architecting Security and Troubleshooting Functions in Routers

Z. Morley Mao

University of Michigan

Networks today lack security and reliability guarantees

- Incrementally deployable solutions attempted at the edge and application/management layer
 - e.g., routing anomaly detection to identify routing attacks; overlay routing to overcome transient failures.
- Limited router support to achieve effective solutions (no ground truths)
 - ping, traceroute probes
 - BGP feeds from a few network locations
 - inferred IP-to-AS mappings, topologies, policies...

Router support?

- Take advantage of programmable router platforms
- Add desired functions inside routers
 - e.g., self-diagnosis
- Identify router primitives whose compositions enable key operational tasks
- Study necessary support residing inside routers vs. at the management layer

Example config command sequence

```
no ip prefix-list PL123  
ip prefix-list PL123 permit  
1.2.3.0/24
```

```
clear ip bgp 1.2.3.4 soft  
in
```

```
interface serial:1_0  
no ip access-group 123 in
```

```
no access-list 123  
access-list 123 permit 1.2.3.4  
access-list 123 permit 1.2.3.5  
access-list 123 deny all
```

```
interface serial:1_0  
ip access-group 123 in
```

```
interface serial:1_0  
no ip access-group 123 in
```

```
no access-list 123  
access-list 123 permit 1.2.3.4  
access-list 123 permit 1.2.3.5  
access-list 123 deny all
```

```
no ip prefix-list PL123  
ip prefix-list PL123 permit  
1.2.3.0/24
```

```
interface serial:1_0  
ip access-group 123 in
```

```
clear ip bgp 1.2.3.4 soft  
in
```

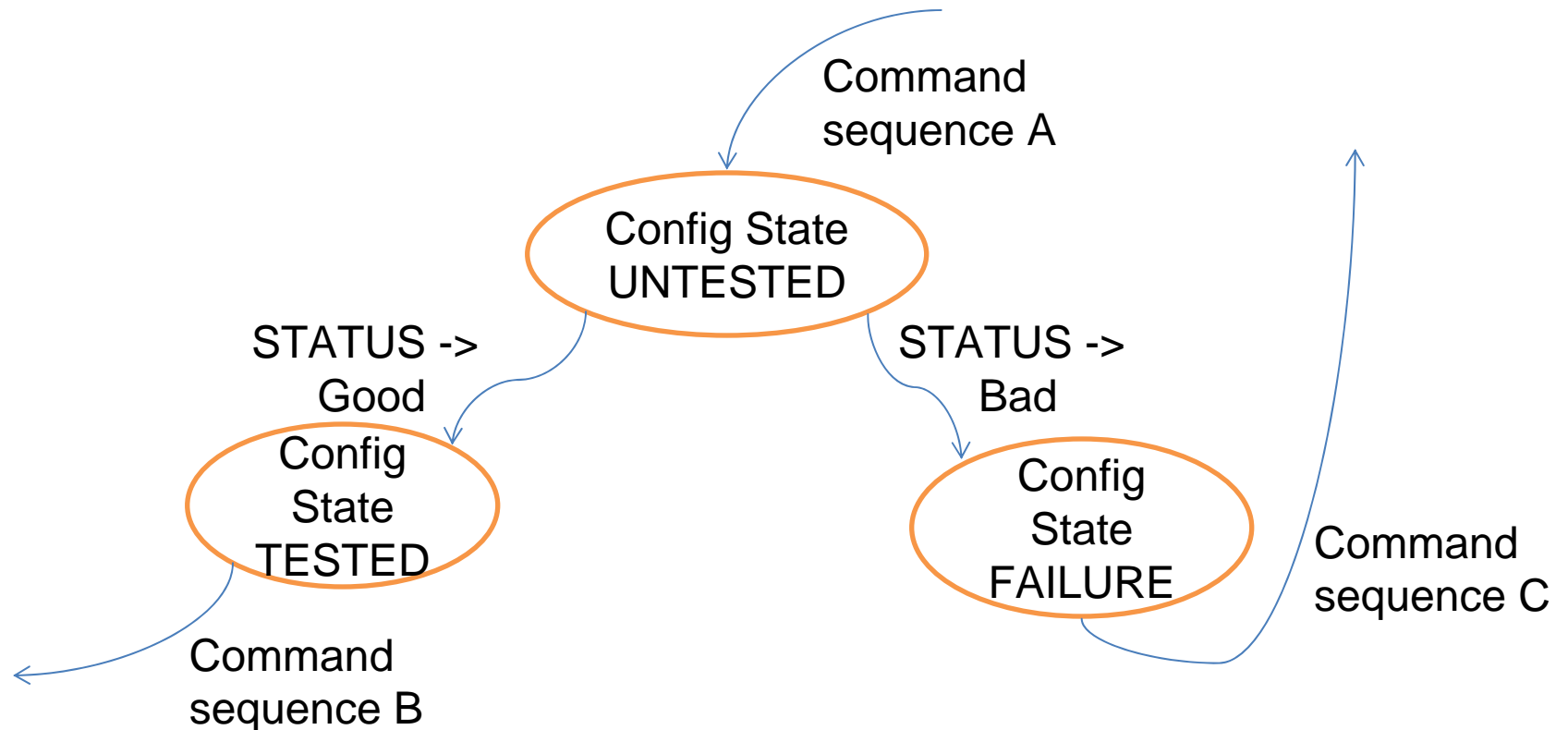
Example useful router support:

“Undo” the effects of configuration changes

- Configuration operations may not always succeed
 - require modifications of multiple routers
 - sequential dependency of consecutive steps
- “Speculative execution” like support
 - identify operational sequence with minimal disruption, easily reversible
 - transaction like semantics
- Example: ACL modification
 - buffer packets before commit

Modeling network state

- The operations performed on states with different status are different



Example useful router support:

Unique identifying information

- IP prefix hijacking can be detected using data-plane information
- Router fingerprints enable conflict detection
 - Attack indication: two traffic sources reach distinct networks

Proposed router support

- Sharing relevant information in response to queries
- Built-in, primitive programmable functions
- Cross-layer correlation, layer-independent management
- Intelligent resolution of management conflicts

Sharing relevant information in response to queries

- Static properties
 - router's geo location
 - router's AS number
 - router design, e.g., scheduling policies
- Dynamic properties
 - link utilization, packet loss statistics, failure logs
 - changes of router state
- Restricted access
 - rate-limit responses
 - enable access based on traffic sources
 - enable access to aggregated information

Built-in programmable functions

- Make use of virtualization and programmable platforms
- Detect deviations from “normal” behavior
 - failure and attack symptoms
- programmable packet filtering and scheduling
 - based on dynamically adjustable packet properties
- Routers self-discover their “roles”
 - propagate specific functionalities to neighboring routers (e.g., pushback)
 - control plane signaling across routers
 - intelligent distribution of functions across routers
 - e.g., placement of route filters

Cross-layer correlation, layer-independent management

- Information across protocol layers
 - physical, link, network, application layers
- Use idle processors for data correlation
 - e.g., detect forwarding path integrity violations
 - e.g., detect local impact of failures
- Layer-independent management
 - reduce dependencies
 - resolve conflicts automatically

Intelligent resolution of management conflicts

- Identify subtle dependencies to avoid and resolve conflicts across management tasks
 - tasks may undo each other's effects (e.g., DDoS mitigation and traffic engineering)
 - systematic conflict resolution
 - by modeling detailed steps of each task
 - conflict detection by checking invariants and assertions
 - e.g., link utilization below some threshold