Building Extensible Networks with Rule-Based Forwarding

Lucian Popa  Norbert Egi  Sylvia Ratnasamy  Ion Stoica
UC Berkeley/ICSI  Lancaster Univ.  Intel Labs Berkeley  UC Berkeley
Making Internet forwarding flexible

- A network’s core functionality is to forward packets
  - “Power” of a network $\Leftrightarrow$ flexibility of its forwarding plane

- A long-held goal: \textit{flexible} forwarding
Making Internet forwarding flexible

- A long-held goal: *flexible* forwarding

- Example: Middlebox-aware forwarding
Making Internet forwarding flexible

- A long-held goal: *flexible* forwarding

- Example: Middlebox-aware forwarding

![Diagram showing a network flow with a firewall and an IDS. The diagram shows a source (S) with Port != 80, a firewall, and a destination (D) with Port = 80. An IDS is placed between S and D.]

Intrusion Detection System (IDS)
Making Internet forwarding flexible

- A long-held goal: *flexible* forwarding

- Example: Middlebox-aware forwarding

- Many such examples: source routes, multiple paths, anycast, mobility, multicast, active networks, etc.
Making Internet forwarding flexible

- A long-held goal: *flexible* forwarding

- Example: Middlebox-aware forwarding

- Many such examples: source routes, multiple paths, anycast, mobility, multicast, active networks, etc.

- Using general *forwarding directives* – instructions to the network on how to forward packets
Thesis

- Flexibility alone is not enough
  - Can compromise network security
  - *E.g.*, source routing, active networks
Thesis

- Flexibility alone is not enough
  - Can compromise network security
  - E.g., source routing, active networks

- Flexibility must be balanced by policy support
  - Every forwarding directive can be constrained by policies
Thesis

- Flexibility alone is not enough
  - Can compromise network security
  - *E.g.*, source routing, active networks

- *Flexibility must be balanced by policy support*
  - *Every forwarding directive can be constrained by policies*

- “Real world” example:
  1. A car can be driven only by its owner
Thesis

- Flexibility alone is not enough
  - Can compromise network security
  - *E.g.*, source routing, active networks

- **Flexibility must be balanced by policy support**
  - *Every forwarding directive can be constrained by policies*

- “Real world” example:
  1. A car can be driven only by its owner
  2. Anyone can drive *any* car
Thesis

- Flexibility alone is not enough
  - Can compromise network security
  - *E.g.*, source routing, active networks

- Flexibility must be **balanced** by **policy** support
  - *Every forwarding directive can be constrained by policies*

- “Real world” example:
  1. A car can be driven only by its owner
  2. Anyone can drive *any* car
  3. Can only drive a car with the *approval* of its owner
Thesis

- Flexibility alone is not enough
  - Can compromise network security
  - *E.g.*, source routing, active networks

- Flexibility must be **balanced** by **policy** support
  - *Every forwarding directive can be constrained by policies*

- Every entity that explicitly appears in a forwarding directive can refuse that directive
Constrain forwarding directives

- Every entity that explicitly appears in a forwarding directive can refuse that directive
Constrain forwarding directives

Every entity that explicitly appears in a forwarding directive can refuse that directive

Example: apply thesis to current Internet
- Forwarding directive = Send to destination D
- Policy of D = No packets from S
- Not respected in the current Internet
Constrain forwarding directives

- Every entity that explicitly appears in a forwarding directive can refuse that directive

- Example: IP source routing
  - Option available with current IP spec
  - Not supported by ISPs since there is no way to constrain it
  - Desirable: ISPs get to approve source routes
Constrain forwarding directives

- Every entity that explicitly appears in a forwarding directive can refuse that directive

Example: Middlebox-aware forwarding
- Allows use of in-network processing
- Policy of $M$: only process $S$-$D$ traffic
Constrain forwarding directives

- Every entity that explicitly appears in a forwarding directive can refuse that directive

Policy-compliance
Goal

- **Flexible and Policy-Compliant** architecture
- Flexible: path control, use in-network functionality and state
- Policy-compliant: all stakeholders’ policies are respected
Idea: Packet Rule

- Forwarding directives carried *in packet*

- Current Internet – packet sent to destination
- Rule Based Forwarding – packet sent to rule
Idea: Packet Rule

- Forwarding directives carried *in packet*

Rule:

```java
if (packet.dest_port == 80)
   sendto D
else
   sendto M
```

```
S  M  D
Port != 80
Port = 80
```
Idea: Packet Rule

- Forwarding directives carried in packet

Rules tell network *how* to forward packets
Idea: Packet Rule

- Forwarding directives carried *in packet*

- All rule participants authorize (sign) the rule
Idea: Packet Rule

- Forwarding directives carried *in packet*

- All rule participants authorize (sign) the rule
Idea: Packet Rule

- Forwarding directives carried \textit{in packet}

- All rule participants authorize (sign) the rule
- All packets carry rules
Idea: Packet Rule

- Forwarding directives carried \textit{in packet}
- All rule participants authorize (sign) the rule
- All packets carry rules

Rules tell network \textit{which} packets can be forwarded
Idea: Packet Rule

- Forwarding directives carried in packet

Rules tell network how to forward packets

Rules tell network which packets can be forwarded
Idea: Packet Rule

- Rules naturally tie in flexibility and policy-compliance
Idea: Packet Rule

- *Rules naturally tie in flexibility and policy-compliance*

  Specifies flexible directives

  Rule
Idea: Packet Rule

- Rules naturally tie in flexibility and policy-compliance

Specifies flexible directives

Rule

Policies approve/disapprove rule
Idea: Packet Rule

- Rules naturally tie in flexibility and policy-compliance

- Specifies flexible directives

- Encapsulates proof of policy-compliance

- Policies approve/disapprove rule
Idea: Packet Rule

- Rules naturally tie in flexibility and policy-compliance

Specifications:
- Specifies flexible directives
- Encapsulates proof of policy-compliance

Policy Evaluation:
- Policies approve/disapprove rule

Router Functions:
- Routers only need information in packet (rule) to:
  - Forward the packet
  - Verify that it complies with policies of all parties
Outline

- Motivation & Solution Overview
- Rule-Based Forwarding Architecture – Overview
- Rule Forwarding Mechanism & Examples
- Evaluation
Rule-Based Forwarding (RBF) Architecture

Senders

Routers

Destinations

Destinations own rules
For policy-compliance, rules are *certified* by trusted entities – Rule Certification Entities (RCEs)
All entities named in the rule (destination, middleboxes, routers) must authorize the rule.
Rule-Based Forwarding (RBF) Architecture

Senders  Routers  Destinations

RCE

R_D

D
Rule-Based Forwarding (RBF) Architecture

Extended DNS: returns rule instead of address

Insert rules into DNS

Senders

Routers

Destinations

DNS

RCE

R_D

D
Rule-Based Forwarding (RBF) Architecture

Sources obtain rules

Senders -> R_D -> DNS

RCE

Senders

Routers

Destinations
Rule-Based Forwarding (RBF) Architecture

Insert them *in packets*
Rule-Based Forwarding (RBF) Architecture

- **Senders (S)**
- **Routers**
  - Verify rule signature
  - Follow rule directives
- **Destinations (D)**
- **DNS**
- **Payload**
- **RCE**
Rule-Based Forwarding (RBF) Architecture

Packets may contain a *return* rule

<table>
<thead>
<tr>
<th>Sender</th>
<th>Router</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>R_S, R_D</td>
<td>D</td>
</tr>
<tr>
<td>R_D, R_S</td>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>

DNS

RCE
Rule-Based Forwarding (RBF) Architecture

Control Plane

- DNS
  - Distribution

Certification

Data Plane

- Senders
- Routers
- Destinations
RBF Assumptions

- Anti-spoofing mechanism
  - Ingress filtering

- Existence of Rule Certifying Entities and distribution of RCE keys to routers
  - RCEs few large Verisign-like entities or AS based

- Rule distribution (DNS) well provisioned against DDoS attacks
Outline

- Motivation & Solution Overview
- Rule-Based Forwarding Architecture – Overview
  - Rule Forwarding Mechanism & Examples
  - Evaluation
RBF Mechanism – Specification

- Rule: **sequence of actions** conditioned by **if-then-else** statements

  ```
  if(<CONDITION>) ACTION1
  else ACTION2
  ```

- Conditions: *comparison operations* on packet header & router state (attributes)
RBF Mechanism – Actions

Rule actions are:

1. Modify packet header (attributes)
2. Drop packet
3. Forward packet (destination / next waypoint)
4. Invoke upper layer functionality (if available)
Rule Forwarding Mechanism

Current IP routers
Rule Forwarding Mechanism

RBF routers

New forwarding layer

Rule Forwarding Mechanism

- Rule Forwarding
- Router Attributes
- IP Forwarding
- FIB

RBF

IP
Rule Forwarding Mechanism

RBF routers

IDS  Multicast  Caching  ...

New forwarding layer

Rule Forwarding  Router Attributes

IP Forwarding  FIB

Specialized forwarding functions (optional)

RBF  IP
Rule Forwarding Mechanism

- **Rule Forwarding**
  - Controlled by ISPs
  - and middlebox owners

- **Router Attributes**
  - IDS
  - Multicast
  - Caching
  - ...
Rule Forwarding Mechanism

Examples:
- router’s address
- queue size
- availability of specialized function
Rule Forwarding Mechanism

- IDS
- Multicast
- Caching
- Rule Forwarding
- Router Attributes
- IP Forwarding
- FIB

Rules cannot modify router attributes

Examples:
- router’s address
- queue size
- availability of specialized function
Rule Forwarding Mechanism

- Rule Forwarding Mechanism
- Rule Forwarding
- Router Attributes
- IP Forwarding
- FIB

- 5 tuple
- Arbitrary semantics (e.g., middlebox was visited)
Rule Forwarding Mechanism

Rule Forwarding

Router Attributes

IP Forwarding

FIB

Rules can modify packet attributes

- 5 tuple
- Arbitrary semantics (e.g., middlebox was visited)
Rule Forwarding Mechanism
Rule Forwarding Mechanism

Rule can:
1. Modify packet attributes
Rule Forwarding Mechanism

Rule can:
1. Modify packet attributes

Example:
```
if(router.congestion > pkt.max_congestion)
    pkt.max_congestion = router.congestion
    sendto D
```
Rule Forwarding Mechanism

Rule can:
1. Modify packet attributes
2. Drop packet
Rule Forwarding Mechanism

Rule can:
1. Modify packet attributes
2. Drop packet

Example:
if(pkt.source != S) drop
sendto D
Rule Forwarding Mechanism

1. Modify packet attributes
2. Drop packet
3. Forward
Rule Forwarding Mechanism

Rule can:
1. Modify packet attributes
2. Drop packet
3. Forward

Example: sendto D
Rule Forwarding Mechanism

Rule can:
1. Modify packet attributes
2. Drop packet
3. Forward
4. Invoke
Rule Forwarding Mechanism

Rule can:
1. Modify packet attributes
2. Drop packet
3. Forward
4. Invoke

Example:
```python
if(router.has_caching == TRUE):
    invoke CachingFunc
sendto D
```
RBF Mechanism – Rule Lease

- Each rule has an associated lease period
- Routers drop expired rules
Examples – Waypoint

R_D:
“Go to R1 before reaching D”
Examples – Waypoint

R_D:
  if(packet.been_to_R1 == 0)
    if(router.address != R1)
      sendto R1
    else packet.been_to_R1 = 1
  if(packet.been_to_R1 == 1)
    sendto D
Examples – Waypoint

R_D:
if(packet.been_to_R1 == 0)
  if(router.address != R1)
    sendto R1
  else packet.been_to_R1 = 1
if(packet.been_to_R1 == 1)
  sendto D

Packet attribute indicating whether packet has visited R1
Examples – Waypoint

R_D:
if(packet.been_to_R1 == 0)
  if(router.address != R1)
    sendto R1
  else packet.been_to_R1 = 1
if(packet.been_to_R1 == 1)
  sendto D
Examples – Waypoint

R_D:
if(packet.been_to_R1 == 0)
  if(router.address != R1)
    sendto R1
  else packet.been_to_R1 = 1
if(packet.been_to_R1 == 1)
sendto D

At the waypoint

S

R1

D

R_D 

been_to_R1 = 1
Examples – Waypoint

R_D:
if(packet.been_to_R1 == 0)
  if(router.address != R1)
    sendto R1
  else packet.been_to_R1 = 1
else packet.been_to_R1 = 1
if(packet.been_to_R1 == 1)
  sendto D

After the waypoint
Examples – Middlebox

R_D:
if(packet.been_to_R1 == 0)
  if(router.address != R1)
    sendto R1
  else
    packet.been_to_R1 = 1
    invoke IDS_func
  
if(packet.been_to_R1 == 1)
  sendto D

Addition to the waypoint rule

R1 – IDS functionality
Examples - Secure Middlebox

R_D:
if(packet.been_to_R1 == 0)
  if(router.address != R1)
    sendto R1
  else
    packet.been_to_R1 = 1
    invoke IDS_func
if(packet.been_to_R1 == 1)
  sendto D

Malicious user could set the packet attributes so that packet appears to have visited the middlebox.
Examples – Secure Middlebox (1)

R_D:
    if(packet.been_to_R1 == 0)
        if(router.address != R1)
            sendto R1
        else
            packet.been_to_R1 = 1
            packet.source = R1
            invoke IDS_func
        if(packet.been_to_R1 == 1)
            if(packet.source == R1)
                sendto D

Allow only packets from R1 when state equals 1
Anti-spoofing does not allow spoofing the source attribute
Examples – Secure Middlebox (2)

R_D:
if(packet.been_to_R1 == 0)
  if(router.address != R1)
    sendto R1
  else
    packet.been_to_R1 = 1
    invoke Crypto_proof
if(packet.been_to_R1 == 1)
  packet.been_to_R1 = 2
  invoke IDS_func
if(packet.been_to_R1 == 2)
  if(router.address != D)
    sendto D
  else
    invoke Verify_and_Deliver

Invoke functionality to (cryptographically) prove packet visited middlebox

Invoke functionality to verify the middlebox proofs at D
Examples – Conditioned Middlebox

R_D:
if(packet.dest_port == 80)
sendto D
else
  //Middleware rule
...

Use the Middleware only for packets not destined to port 80
Examples – DoS protection

- Create “capability-like rules”, e.g., for a client with address S

```
R_S_D:
    if(packet.source != S)
        drop
    sendto D
```
Examples – DoS protection

- Create “capability-like rules”, *e.g.*, for a client with address S

```
R_S_D:
  if(packet.source != S)
    drop
  sendto D
```

- D can control number of simultaneous clients by controlling number of authorized rules (a rule for each client)
Examples – DoS protection

• Create “capability-like rules”, e.g., for a client with address S

```
R_S_D:
    if(packet.source != S)
        drop
        sendto D
```

• D can control number of simultaneous clients by controlling number of authorized rules (a rule for each client)

• Need to grant rules on demand
  • Dynamic (vs. static DNS)
  • Provision this service against DDoS (denial of rule)
  • DNS redirects to third parties providing this service
RBF Examples

- Filter ports/prefixes – only receive specific traffic
- Protect against DoS attacks
- Mobility
- Middleboxes
- Secure loose path forwarding – select provider, reliability
- Multiple paths
- Anycast
- Record path state – network probing, ECN, path identifier
- On-path redirection – Delay Tolerant Networks
- Use on-path router functions deployed by ISPs – Multicast, caching, WAN optimizers, content-routing, energy efficiency
- ...
Rule Properties

1. Flexible
Rule Properties

1. Flexible

- Rules enable endpoints to:
  a) Block unwanted packets in the network
  b) Control path selection using waypoints
  c) Use router state in forwarding decisions and record this state
  d) Use specialized functions at middleboxes and routers, if available
Rule Properties

1. Flexible
2. Policy Compliant
Rule Properties

1. Flexible
2. Policy Compliant
   - Rules are certified by trusted entities – Rule Certifying Entities (RCEs)
   - Rules are above routing-controlled layer – IP
     - Route discovery and computation fully controlled by ISPs
Rule Properties

1. Flexible
2. Policy Compliant
3. Safe
Rule Properties

1. Flexible
2. Policy Compliant
3. Safe

- Bounded forwarding time
  - No loops, only comparison operations, cannot modify payload
- Cannot modify router state
- Cannot amplify traffic
  - No network loops (static analysis), cannot replicate packets
- Invoked functions are fully controlled by ISPs/Mbox owners
  - Resource isolation and access control to prevent attacks
  - Rules merely offer a (policy compliant) mechanism to use them
Related Work

1. Flexibility
2. Policy-Compliance
3. Some of each
Related Work

1. Flexibility

- Active Networks, ESP, Overlays (e.g., i3, DOA), Loose path forwarding, DTN, Mobility (e.g., Mobile IP, HIP), Multiple paths (e.g., MIRO), etc.

- Rules vs. Active Networks:
  - Forwarding directives vs. programs
  - Safe and statically analyzable
  - Policy-compliance for multiple-parties
  - Allow invoking ISP deployed functions for processing
Related Work

1. Flexibility

2. Policy-Compliance
   - In-network filters (PushBack, AITF, StopIt, PredicateRouting, Off-by-default), Network Capabilities (TVA, SIFF)
   - RBF:
     - Adds flexibility
     - Adds multi-party policy compliance
Related Work

1. Flexibility
2. Policy-Compliance
3. Some of each

- *E.g.* Platypus, NUTSS, ICING enable policy-compliant source routing
- RBF:
  - Generalizes flexibility
  - Enables richer policies based on *entire* forwarding behavior
Outline

- Motivation & Solution Overview
- Rule-Based Forwarding Architecture – Overview
- Rule Forwarding Mechanism & Examples

Evaluation
Evaluation – Questions

- Size overhead of rules

- Forwarding overhead
  - Fast path (no rule verification)
  - Slow path (involves rule verification)

- Performance isolation between invoked functions and forwarding

- Load on RCEs

- Security analysis
Evaluation – Questions

- Size overhead of rules

- Forwarding overhead
  - Fast path (no rule verification)
  - Slow path (involves rule verification)

- Performance isolation between invoked functions and forwarding

- Load on RCEs

- Security analysis
Evaluation – Rule Sizes

![Chart showing rule sizes for different rule types with categories for encoding, control, and signature.](image-url)
Evaluation – Rule Sizes

Overhead of one rule is ~60-140 bytes
Evaluation – Rule Sizes

Average 85 bytes: 13% average Internet packet (630B)
Evaluation – Rule Sizes

Average 85 bytes: 13% average Internet packet (630B)

27% if using RSA signatures
Evaluation – Prototype RBF Router

- Software router on top of RouteBricks [SOSP 2009]
- 8 core Nehalem server, 2 dual-port NICs
- Example router setup:
Evaluation – Forwarding Using Rules

- No signature verification, using all 8 cores

![Graph showing throughput rates for different traffic types and configurations.](image)
Rule forwarding incurs little overhead on RouteBricks
Evaluation – Forwarding Using Rules

No overhead for packets > 500B
Soft router RBF can forward 35Gbps real traffic.
Evaluation – Signature Verification

- Only at trust boundary routers (lower traffic than core)

- Result can be cached
  - Cache is small (e.g., 19 bytes/rule) and exact match lookup
  - Packets from new flows represent 1% of backbone link capacity on average, worst case 5% of packets
    - Doable with existing hardware (crypto processors, ASICs)
    - 10% slow down on prototype router with RSA signatures & real traffic

- Can be parallelized!
Summary

- **RBF** – flexible and policy compliant architecture
  - Packets carry rules

- **Rule** – contains forwarding directives
  - *Flexible*: if-then-else conditions on packet & router attributes, modify packet header and use in-network functions
  - *Policy-compliant*: signed by third parties – RCEs
  - *Safe*: cannot corrupt routers or amplify traffic