BGP FlowSpec IPv6
(draft-ietf-idr-flow-spec-v6-03)

Andy Karch

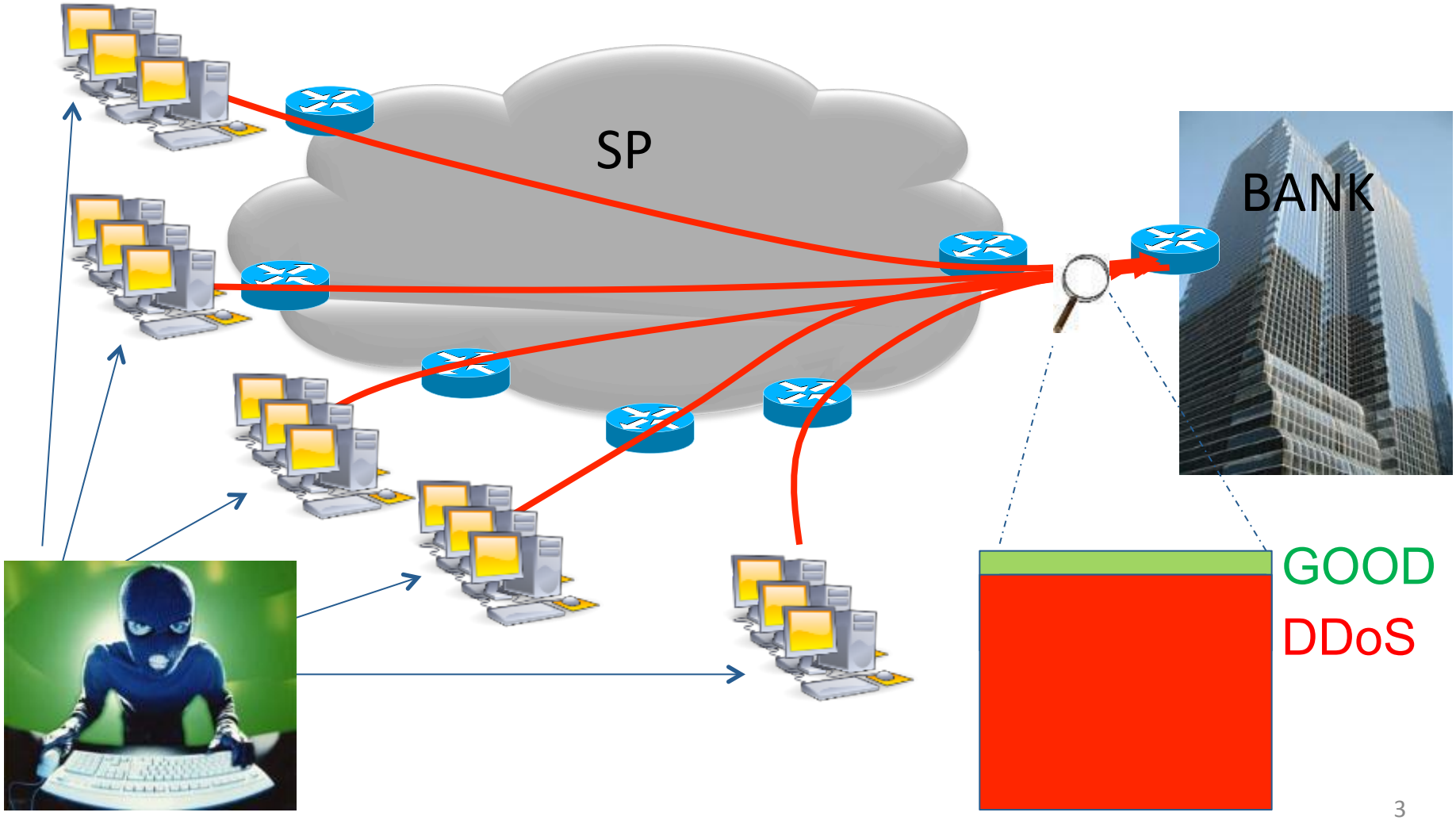
Robert Raszuk

Keyur Patel

What is FlowSpec?

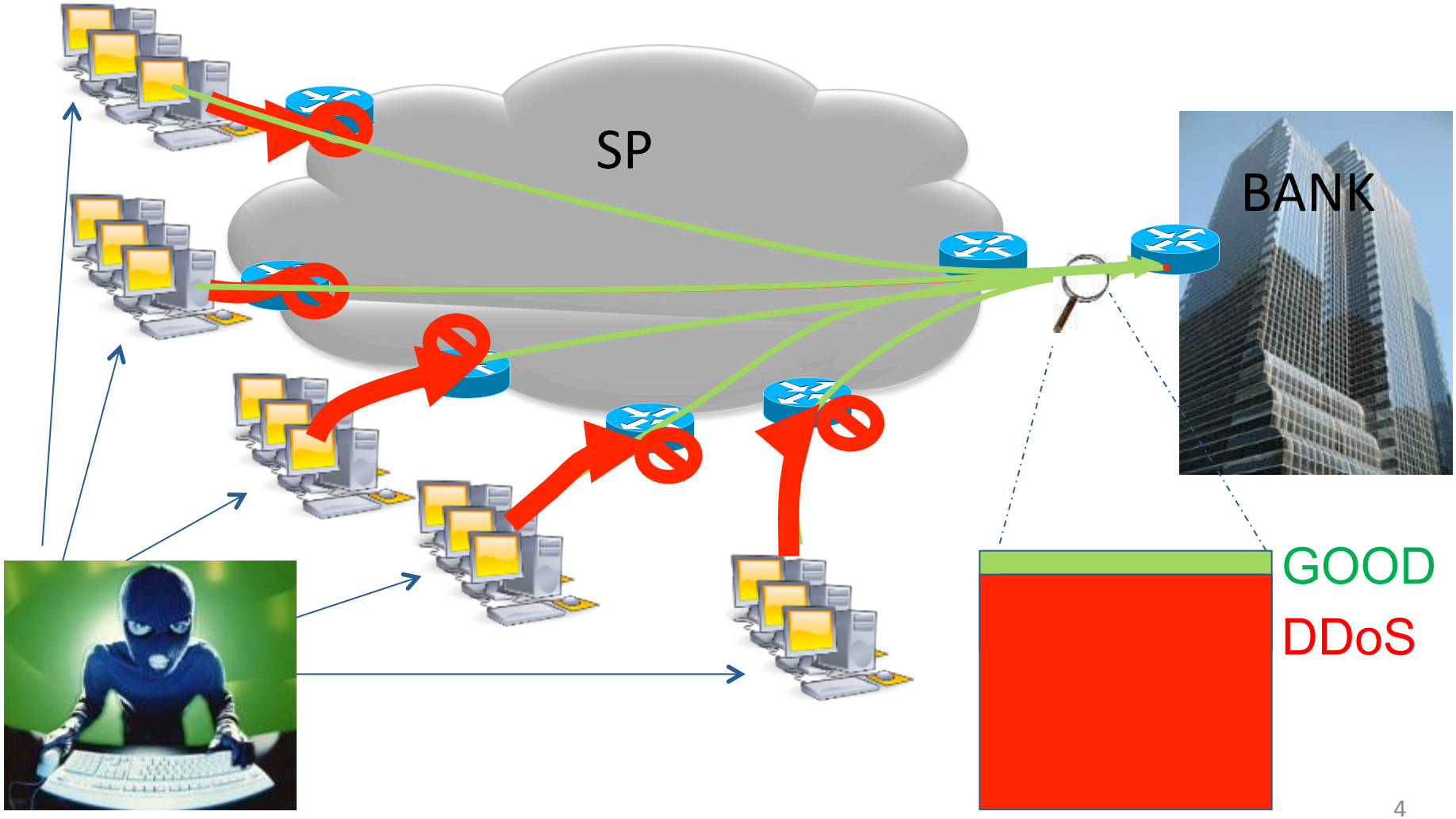
- RFC5575 - Dissemination of Flow Specification Rules
  - IPv4-Only
- BGP NLRI
  - SAFI 133 – IP
  - SAFI 134 – VPN
- Match and action
  - Match on all IP header fields, not just IP destination
  - Actions – rate-limit, drop, redirect, mark, sample
  - Similar to access-lists, policies, and filters
- Use-cases
  - DDoS Mitigation
  - Traffic Filtering

DDoS Impact



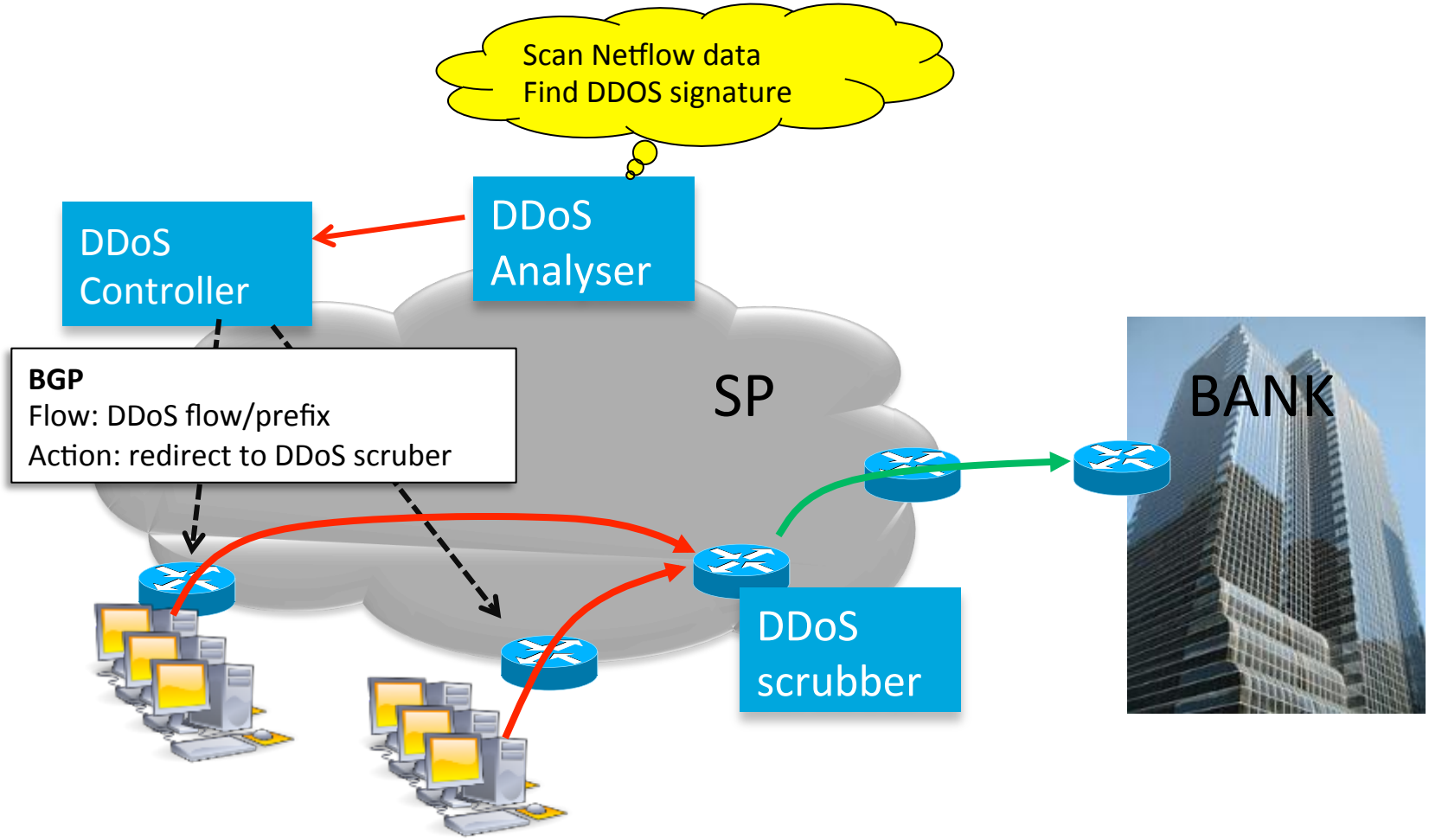
DDoS Mitigation

Drop at network ingress



DDoS Mitigation

Redirect traffic to DDoS scrubber



Changes for IPv6 Match Criteria

- Added
  - IPv6 Flow Label
- Modified
  - Source Address (prefix offset)
  - Destination Address (prefix offset)
  - IP Protocol -> Next Header
  - DSCP -> Traffic Class
- Removed
  - Fragment

Points for Discussion

1) Prefix Offset

2) Ordering of Traffic Filtering Rules

3) Fragmentation

Prefix Offset

- Don't care bits
- Field inside IPv6 prefix components
  - Destination IPv6 Prefix
  - Source IPv6 Prefix
- New for IPv6
- Allows flexible match on part of the IPv6 address
  - Match on end or interior of address.
- Prefix Encoding
  - Based on MP_REACH_NLRI in BGP UPDATE

Encoding: <type (1 octet), prefix length (1 octet), **prefix offset (1 octet)**, prefix>

Prefix Offset
Problems

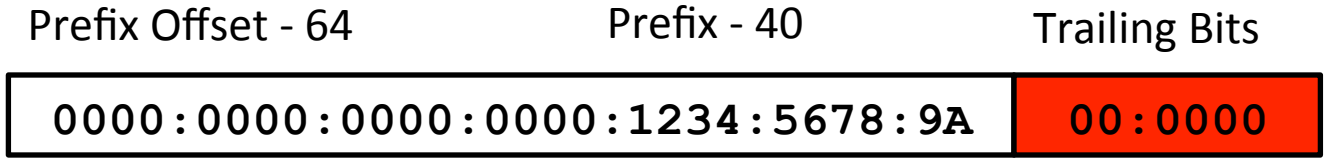
- Encoding
  - Do we include the offset bits?

- Order of Traffic Filtering Rules
  - How does the offset affect ordering?

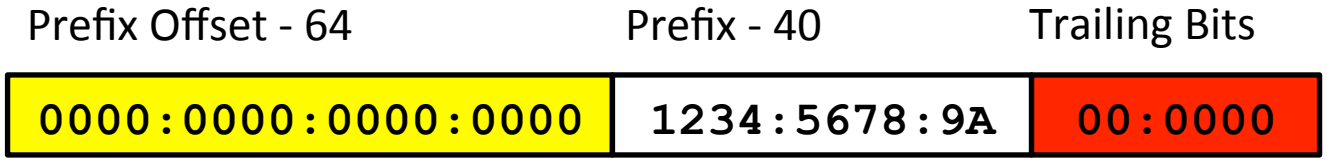
Prefix Offset Encoding

- **0000:0000:0000:0000:1234:5678:9A00:0000**
  - Length - 104
  - Don't care – 64
  - (40 match bits)

- Do we encode the entire <PrefixLength> bits?
  - The <PrefixOffset> bits are dead weight.



- Do we encode <PrefixLength> - <PrefixOffset> bits?
  - Most efficient, but perhaps <PrefixLength> is misleading.



Prefix Offset Encoding
Example - offset bits encoded

An example of a flow specification encoding for: "all packets to ::1234:5678:9A/80-104 from 192::/8 and port {range [137, 139] or 8080}".

destination	source	port
0x01 68 40 **00 00 00 00 00** **00 00 00 12 34 56 78 9A**	02 08 00 c0	04 03 89 45 8b 91 1f 90

- Destination prefix component length - 16
- NLRI Total Length - 28

Prefix Offset Encoding Example - offset bits omitted

An example of a flow specification encoding for: "all packets to ::1234:5678:9A/80-104 from 192::/8 and port {range [137, 139] or 8080}".

destination	source	port
0x01 68 40 **12 34 56 78 9A**	02 08 00 c0	04 03 89 45 8b 91 1f 90

- Destination prefix component length - 8
- NLRI Total Length – 20

Order of Traffic Filtering Rules

Problem:

- More than one rule may match a particular traffic flow.

Solution Requirements:

- Order must be constant in the network.
- Order must not depend on the arrival order of the flow specification's rules

Analogous to Longest-Prefix-Match

Order of Traffic Filtering Rules RFC5575

- IP prefix values (IP destination and source prefix):
  1. Lowest IP value of the common prefix length;
  2. if the common prefix is equal, then the most specific prefix has precedence.
  3. **NO PREFIX OFFSET**

```
if (component_type(comp1) == IP_DESTINATION || IP_SOURCE) {
    common = MIN(prefix_length(comp1), prefix_length(comp2));
    cmp = prefix_compare(comp1, comp2, common);
    // not equal, lowest value has precedence
    // equal, longest match has precedence
} else {
```

Order of Traffic Filtering Rules
Problem

- Without considering prefix offset, multiple flows that differ only by offset may appear equal in priority!

- Example
  - <Length – 32>, <Offset – 1>, <Prefix - 0x01020304)>
  - <Length – 32>, <Offset – 0>, <Prefix - 0x01020304)>

Order of Traffic Filtering Rules IPv6 FlowSpec

- IP prefix values (IP destination and source prefix):
  1. Lowest offset has precendence
     1. RATIONALE: Lowest offset matches more bits
  2. If the offset is equal, lowest IP value of the common prefix length;
  3. if the common prefix is equal, then the most specific prefix has precedence.

```
if (component_type(comp1) == IPV6_DESTINATION || IPV6_SOURCE) {
    // offset not equal, lowest offset has precedence
    // offset equal ...
    common_len = MIN(prefix_length(comp1), prefix_length(comp2));
    cmp = prefix_compare(comp1, comp2, offset, common_len);
    // not equal, lowest value has precedence
    // equal, longest match has precedence
} else {
```

Fragmentation

- Defined in RFC5575
  - Don’t Fragment
  - Is a Fragment
  - First Fragment
  - Last Fragment
- Removed in IPv6 draft

Fragmentation Undetermined Transport

- Unknown EH
- Upper-layer protocol field not in first fragment.
  - Last EH next-header field
- Upper-layer header not in first fragment
  - TCP, UDP, SCTP, ICMP…

Goals

1) Gather feedback and comments

2) Update draft
   - Converge on encoding, ordering, fragmentation
   - Clarify language
   - Provide encoding examples

3) Identify 2 implementations

4) Inter-op in the next few months.

5) Move towards RFC status

Backup Slides

Prefix Offset Component (offset bits omitted)

Type 1 - Destination IPv6 Prefix

Encoding: <type (1 octet), prefix length (1 octet), prefix offset (1 octet), prefix>

Defines the destination prefix to match. Prefix offset has been defined to allow for flexible matching on part of the IPv6 address where we want to skip (don't care) of N first bits of the address. This can be especially useful where part of the IPv6 address consists of an embedded IPv4 address and matching needs to happen only on the embedded IPv4 address. The encoded prefix contains enough octets for the bits used in matching (length minus offset bits).

Prefix Offset Component (offset bits encoded)

Type 1 - Destination IPv6 Prefix

Encoding: <type (1 octet), prefix length (1 octet), prefix offset (1 octet), prefix>

Defines the destination prefix to match. Prefix offset has been defined to allow for flexible matching on part of the IPv6 address where we want to skip (don't care) of N first bits of the address. This can be especially useful where part of the IPv6 address consists of an embedded IPv4 address and matching needs to happen only on the embedded IPv4 address. The default value for prefix offset bits SHOULD be 0, where matching uses subsequent bits up to prefix length. Otherwise prefixes are encoded as in BGP UPDATE messages, prefix length in bits is followed by enough octets to contain the prefix information.