Firebreak: A DDoS Guard Deployment Architecture

Paul Francis

Dec. 2004

DDoS Goals

- DDoS is a serious and growing problem
- DDoS solutions require two components: detect and defend
- Detect the attack
  - Best done near the target (for the most part)
  - Why? Target sees all the traffic, and knows what is "normal"
- Defend: Identify and drop bad packets
  - Best done near the sources
  - Why? Bandwidth available to absorb the attack grows exponentially with distance from the target

We focus on defense deployment architecture

- Not because detection isn't important or hard
- Its just that we have ideas on how to deploy a defensive system
- Having said that: Some observations on detection
  - Increasingly sophisticated attacks look more and more like legitimate traffic
  - Detection will increasingly require application specific knowledge
  - Ultimately attack traffic may only be detected by its volume, possibly over long time scales

The holy grail of DDoS defense

- Every *edge router* has the ability to identify and drop bad packets,
- and any target can determine where packets are coming from (*traceback*), and direct individual edge routers to activate their defenses for packets to the target (*control*)

- Problem is, there is no immediate economic motivation to deploy DDoS defenses on this scale
- Our question: *how can we work towards this scale of defense*?

Two basic commercial approaches today: "CDN" and "ISP"

- CDN approach (e.g. Akamai)
  - Defenses are deployed at ***many*** ISPs to protect a ***small fraction*** of targets at ***many*** ISPs
  - Use DNS to steer packets to its defense boxes
  - Sold to content providers
- ISP approach (e.g. Riverhead/Cisco)
  - Defenses are deployed at ***a single*** ISP to protect ***some or all*** targets in ***a single*** ISP
  - IP/MPLS routing is used to steer packets to defense boxes
  - Sold to ISPs

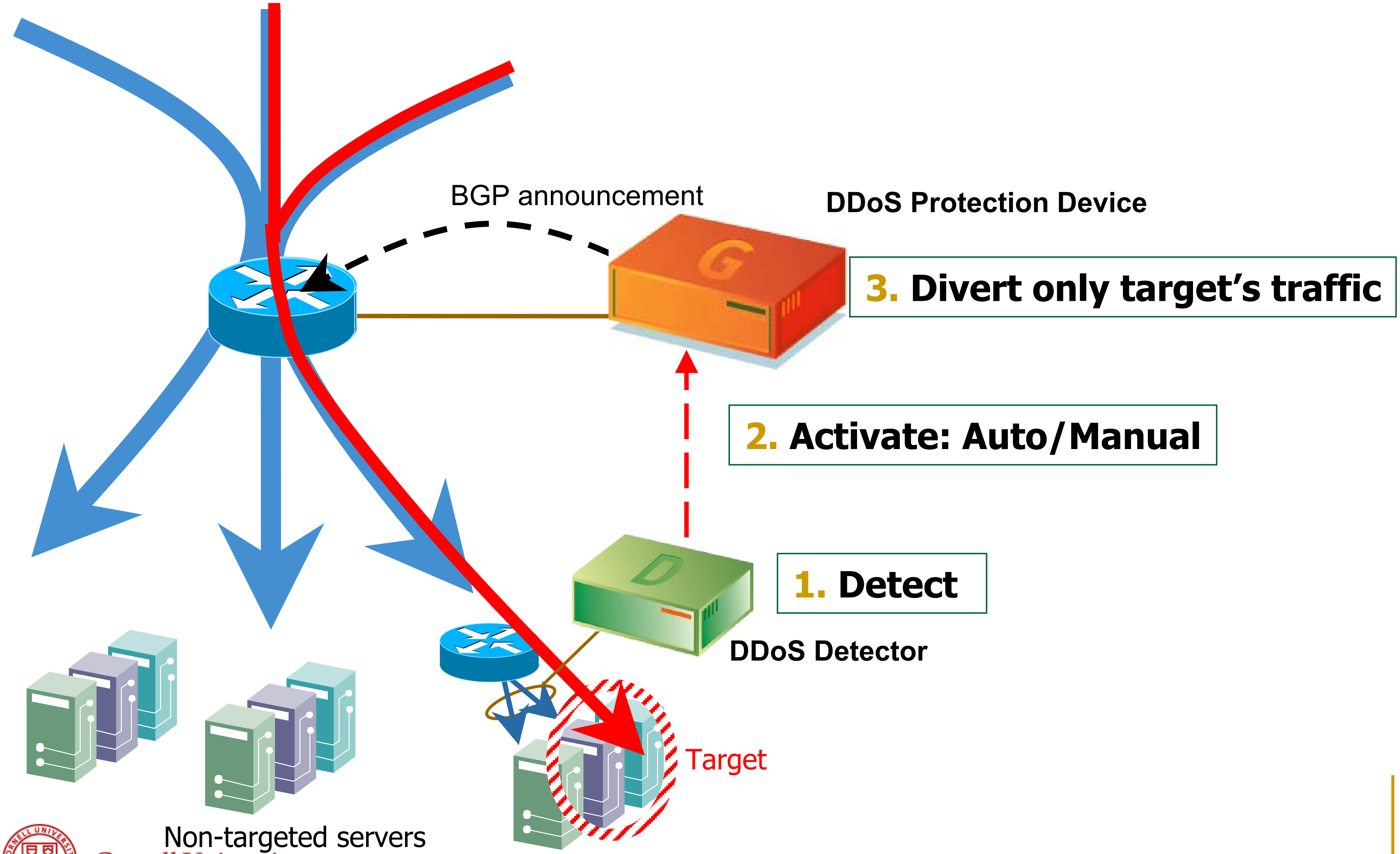
Akamai approach (as I understand it)

- 1000s of web proxies deployed in POPs around the world
- DNS servers steer clients to the proxies
- The proxies protect the origin servers
  - Both through their normal proxy job, and with additional mechanisms (I don't know details)
- DNS deployed in two tiers
- Dozens (?) of top level servers, long TTLs
- 1000s (?) of low level servers, short TTLs

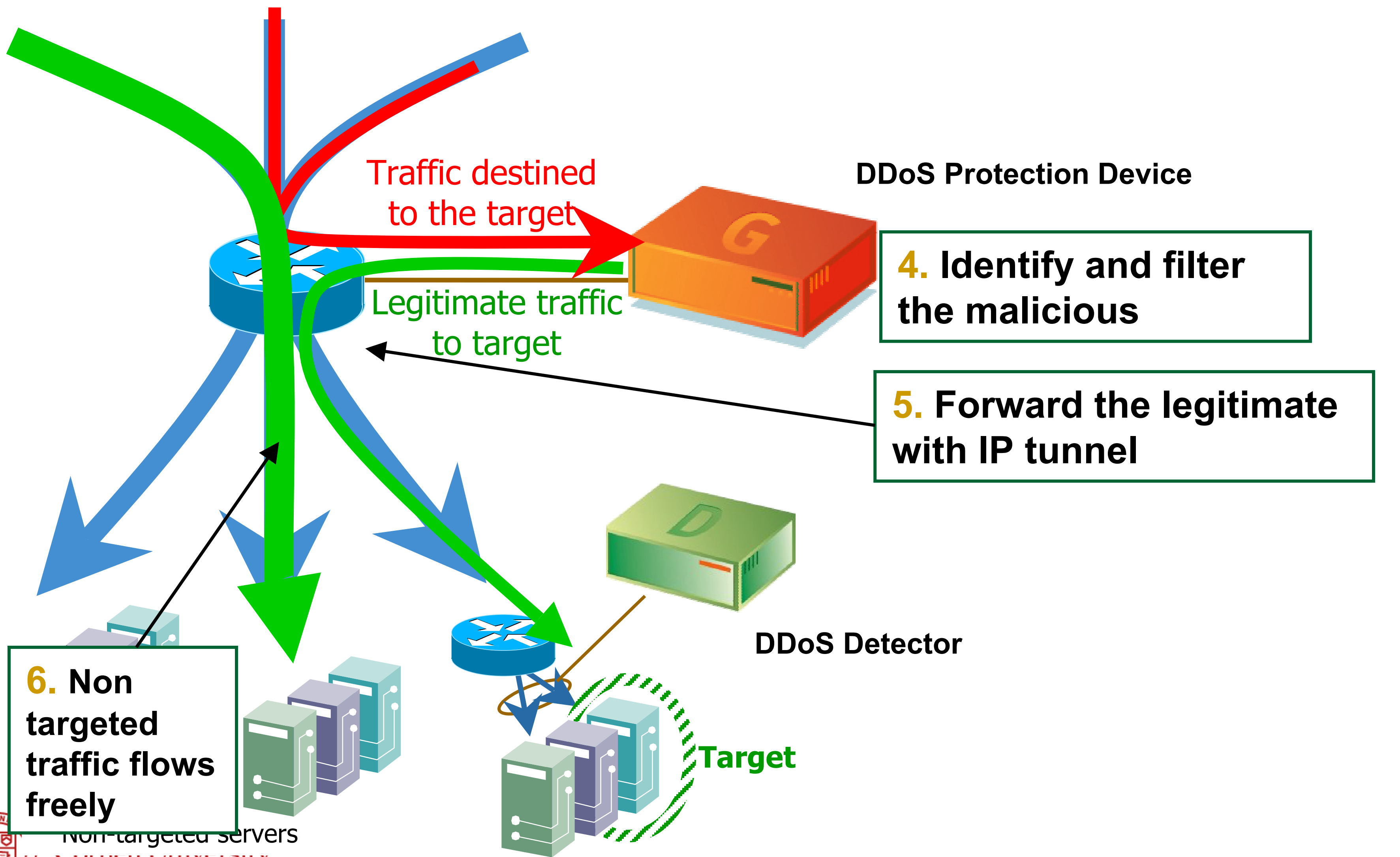
Shortcoming of Akamai's approach

- *The origin server IP address must be kept secret!*
  - Else attacker can bypass the DNS proxies
- Top tier of DNS is attackable
  - Indeed this has happened, with limited effect
  - Though Akamai can always beef up its DNS
    - Anycast, similar to some root servers
- Limited to DNS-based applications
  - Granted there are many of these, but still…

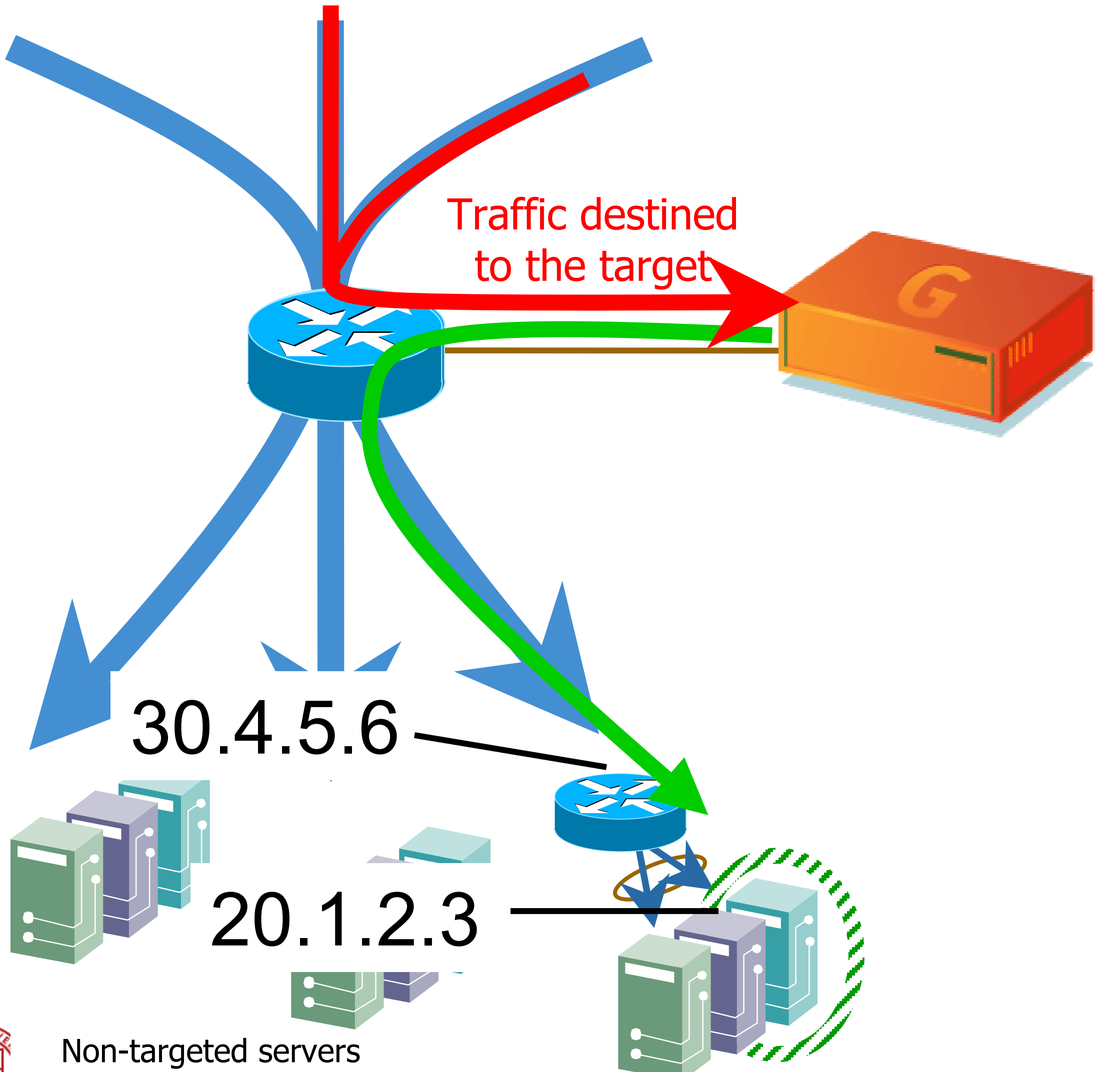
Riverhead's Diversion Approach



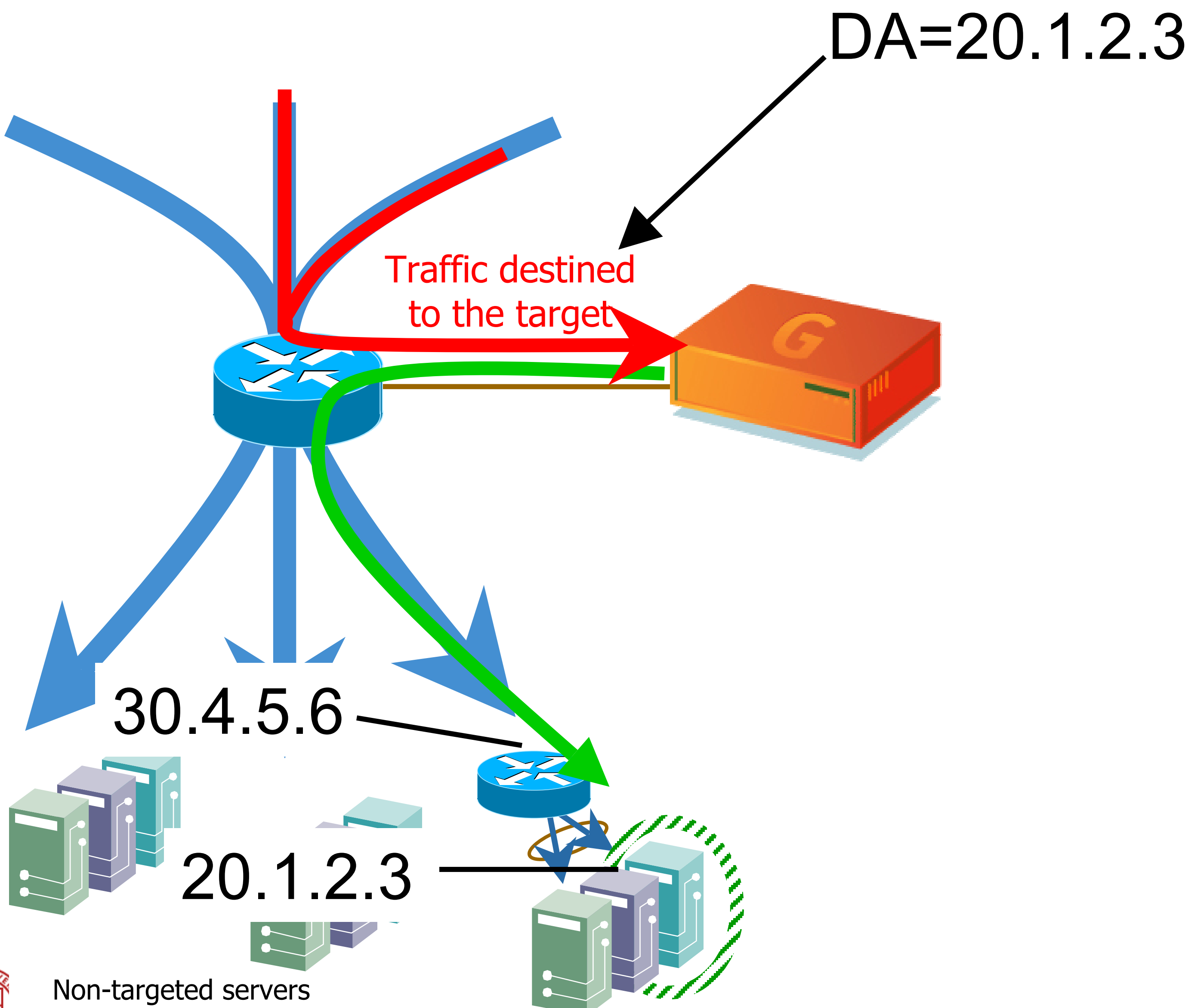
Riverhead’s Diversion Approach



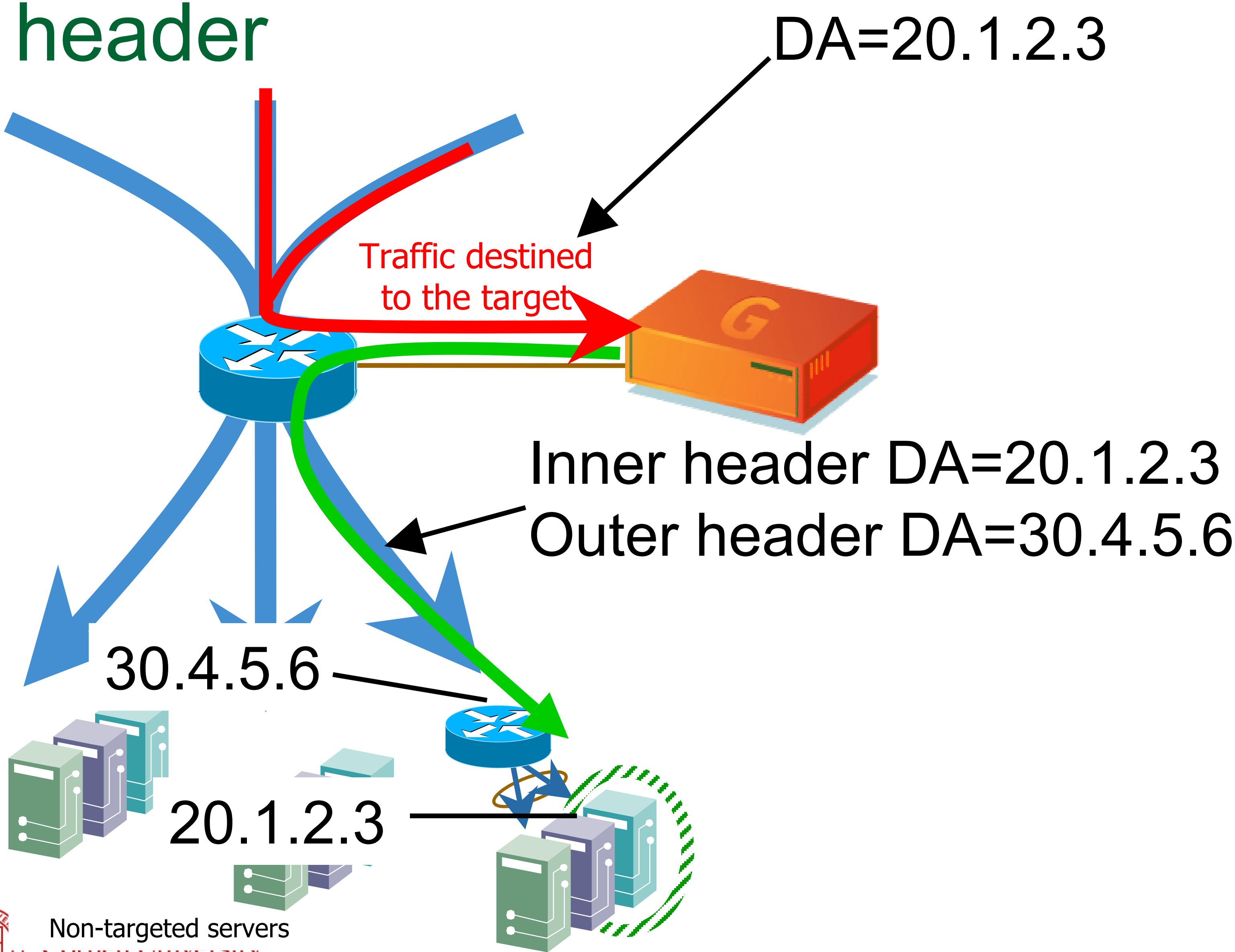
Tunnel details



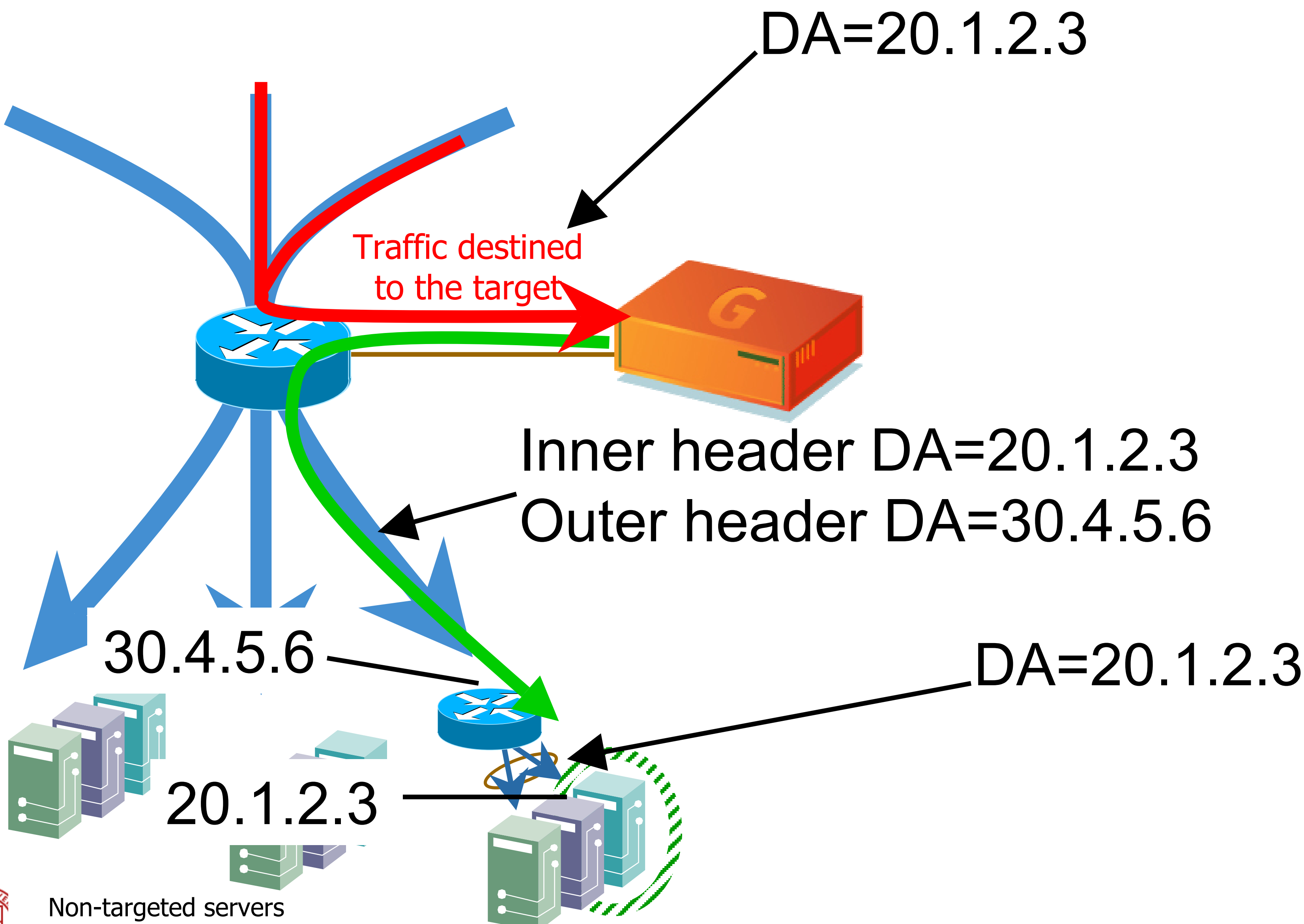
Untunneled traffic diverted to guard



Guard wraps packet in another IP header



Edge router unwraps outer IP header

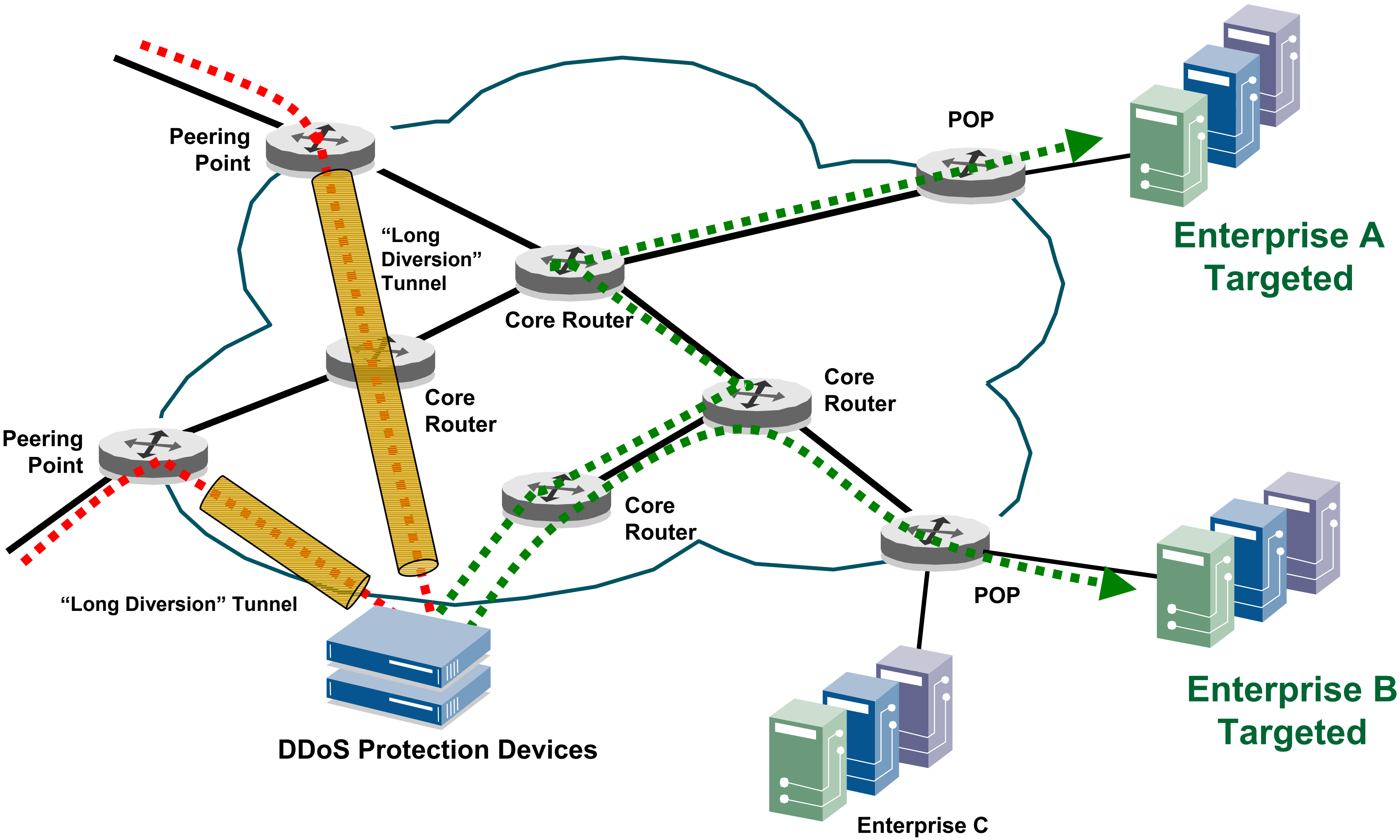


Cornell University

Diversion approach deployments

- Put guards near protected hosting center
  - Problem: large enough attack will simply overwhelm the bandwidth at the hosting center
- Distribute guards at all POPs (or peering points, more-or-less same thing) in ISP
  - Problem: this is a bigger commitment than an ISP may wish to make
- So Riverhead offers a third deployment alternative: “Long Diversion Tunnel”

Long Diversion Tunnel



Long Diversion Tunnel

- Selected destinations are tunneled to "centrally" located stacks of guards
- Tunnel can be MPLS, GRE, L2TP, etc.
- Enable long diversion tunnels when attack is detected

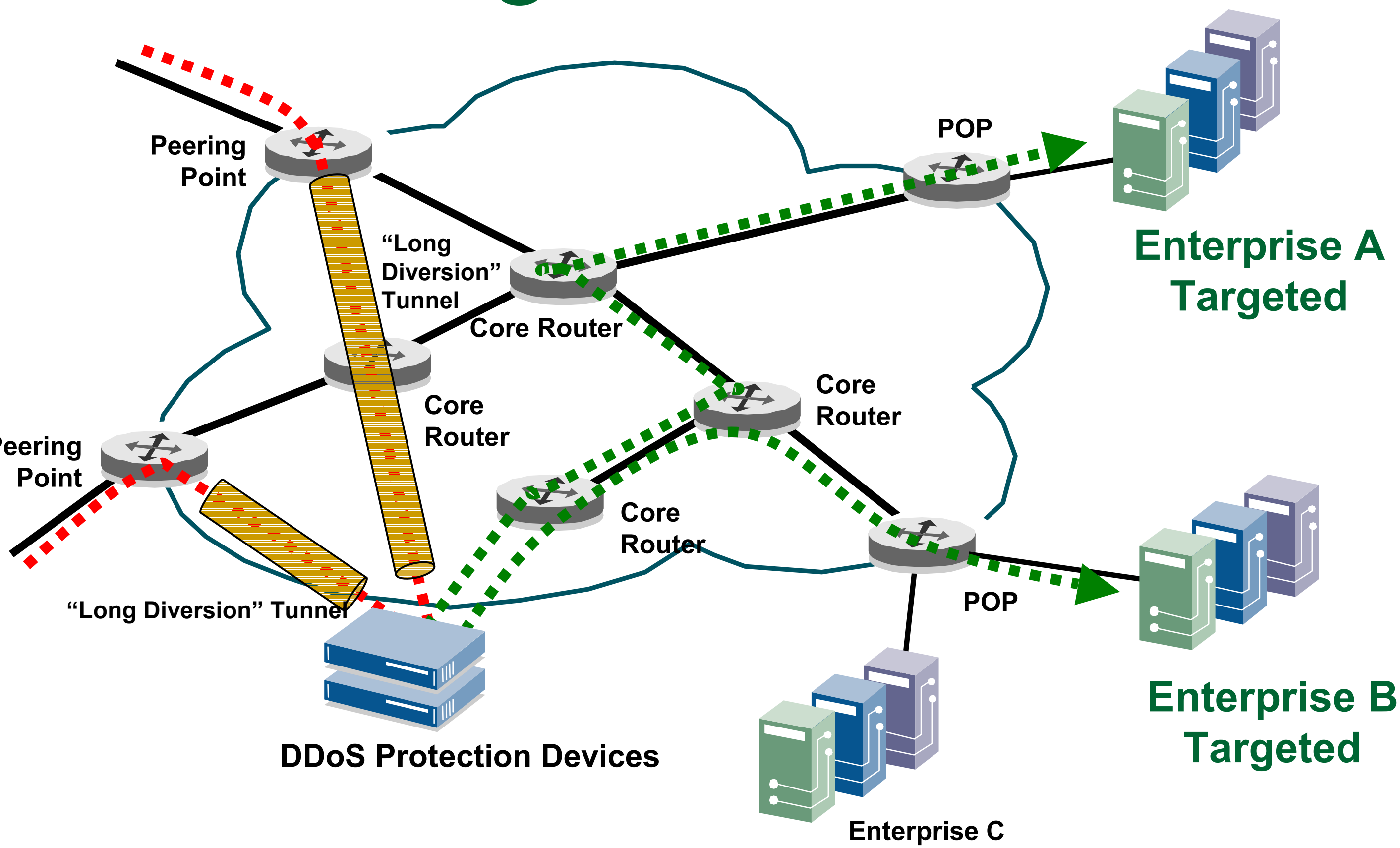
Riverhead deployment model shortcomings

- Ultimately all physical paths towards the target must be “modified”
  - A guard box or tunneling capabilities
- This is ok for a single ISP, but . . .
- *Each ISP must protect itself separately!!!*
  - Each ISP must individually scale up to protect against the largest attack
  - Duplication of effort over hundreds of ISPs

Riverhead across ISPs?

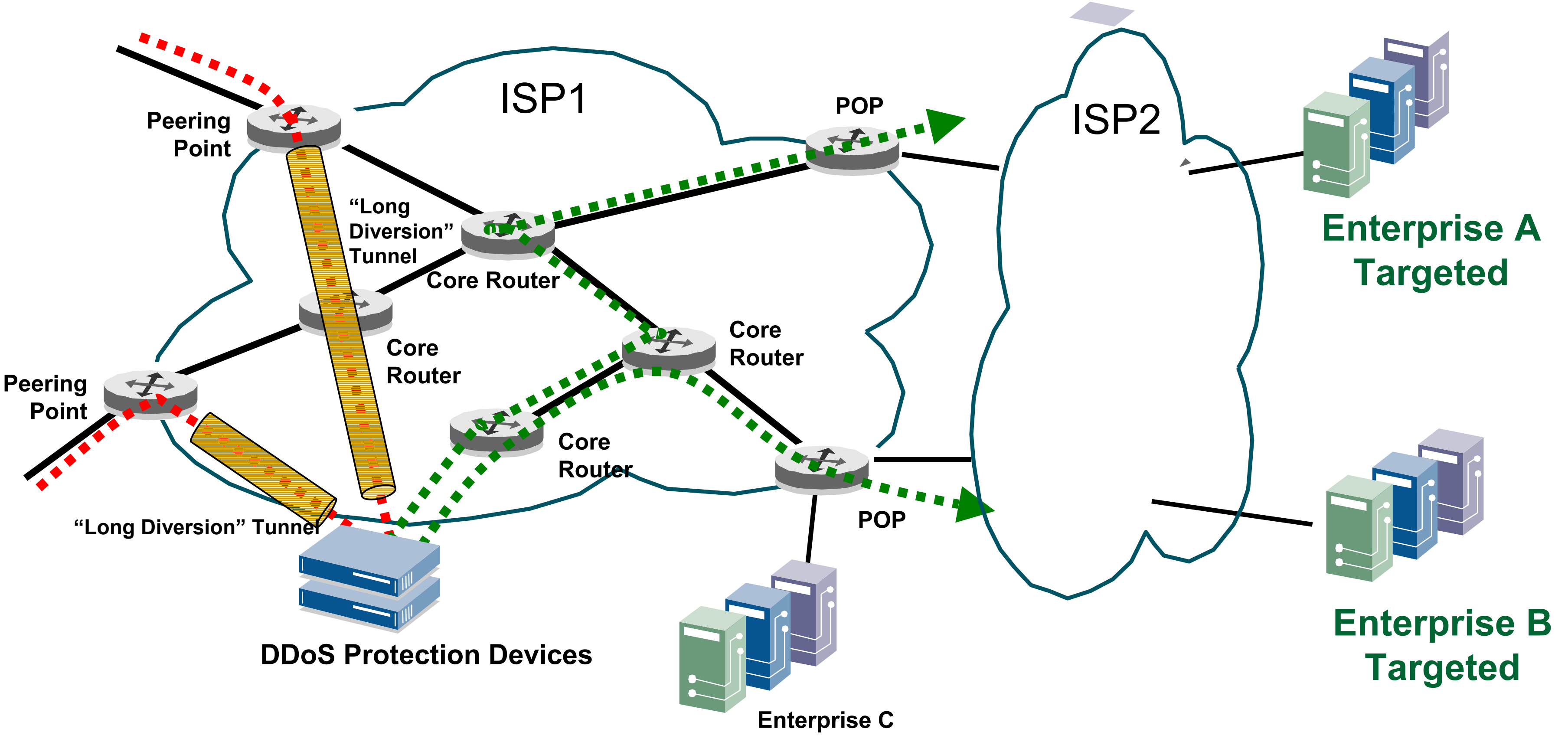
- Can the Riverhead approach work across multiple ISPs?
- Several issues:
  - How to deploy guards across ISPs
    - Diversion: Packets go through multiple guards?
    - Long Diversion: How to configure tunnels across ISPs
  - How to secure commands from the Detectors to the Guards across ISPs
  - How do Detectors know which Guards to activate?

Long Diversion Again



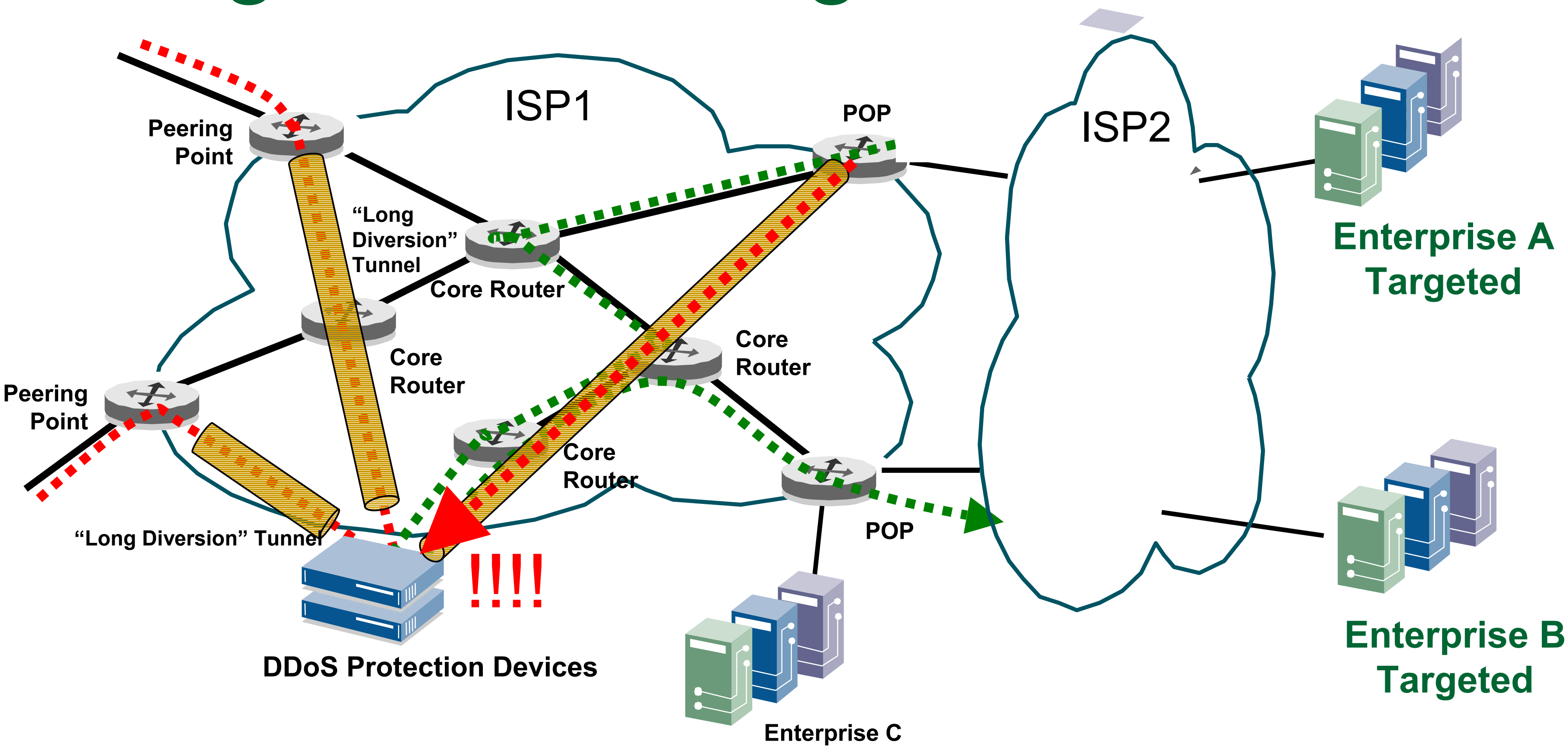
- Once a packet leaves a Guard, it should not encounter another tunneling router
  - At best inefficient, at worst loops may form

Long Diversion Again



- What if there are two ISPs?

Long Diversion Again



- Must be very careful to distinguish incoming from outgoing packets to prevent loops
  - Tricky, especially at high speed

Multi-ISP Long Diversion appears tricky

- In ISP1, packet is tunneled to Guard at a peering router (ingress)
- Guard forwards the packet on towards the destination
- Packet reaches an egress peering router in the same ISP
- How does that peering router know not to simply tunnel the packet back to the guard again?
  - Some relatively complex filtering rule?

Multi-ISP (regular) diversion

- Detector has to somehow tell Guards in other ISPs to start filtering
  - Perhaps using a VPN consisting of Guards and Detectors
  - Authentication here **cannot** fail. If compromised:
  - An attacker could disable the Guards, or
  - Activate Guards for many targets, thus overloading them (with legitimate traffic!)
- Multiple guards traversed on each packet
  - Attacker gets a multiplication effect…

Our Approach: Firebreak

- Naturally we want all the “pros” and none of the “cons” of current approaches

✓ All defenses deployed at many ISPs can be brought to bear on any given attack

  - 100’s of ISPs can leverage each other’s resources

✓ Operates at the IP level

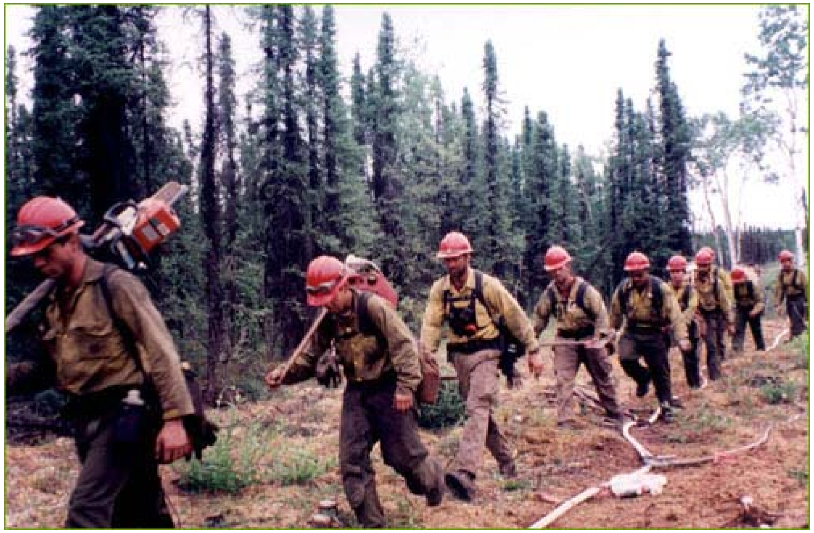
  - Target addresses do not need to be secret
  - Any application can be protected

✓ Incrementally deployable---don’t have to cover every access point across the Internet

✗ All Internet destinations can be protected

Firebreak:

A long swath of cleared vegetation used to contain wildfires



Firebreaks can be natural

Similar to Riverhead in several respects

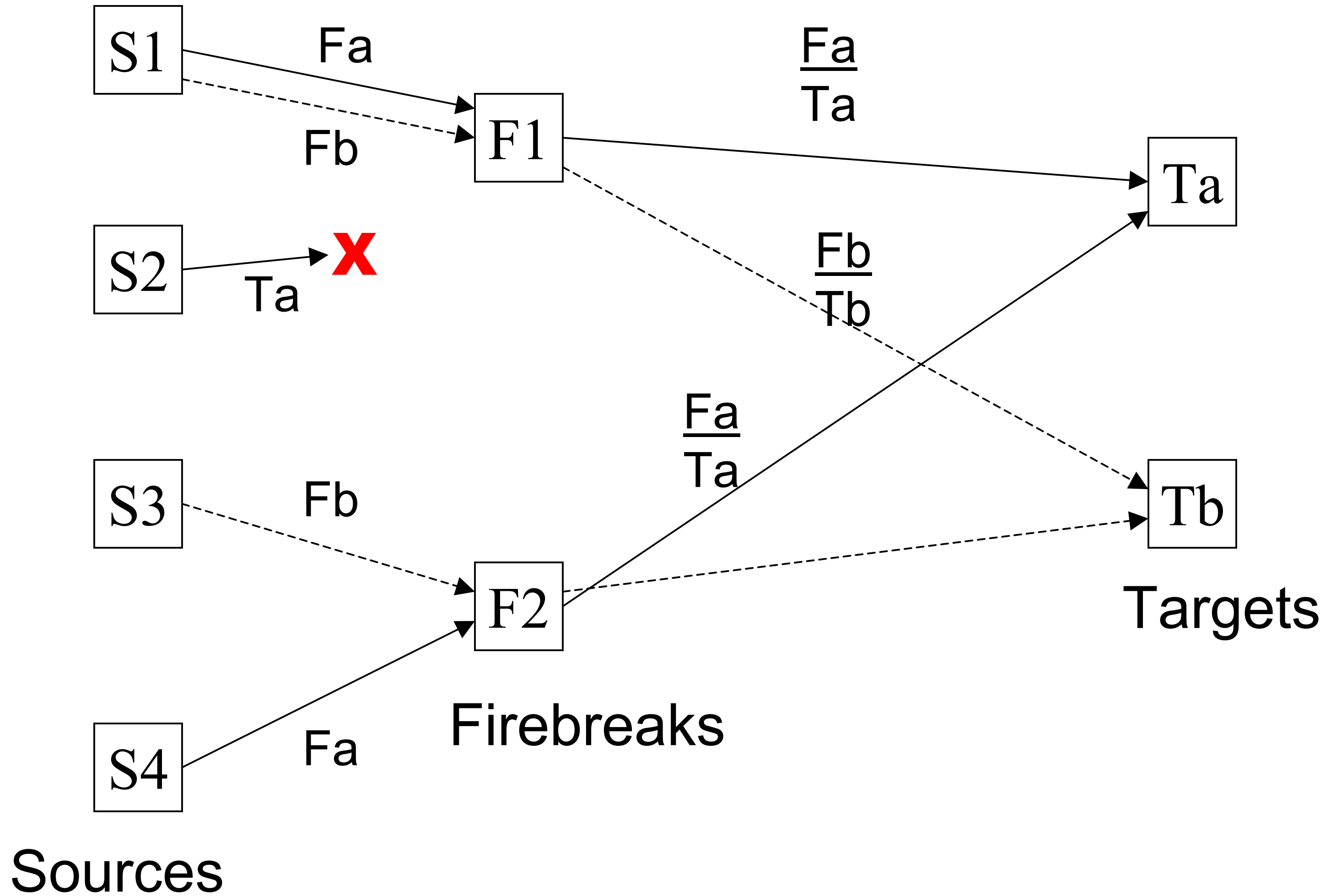
- Detector near the target detects attack
- Guards "in the network" capture and filter packets at IP level
- Indeed, Riverhead product could be used for these two components with not too much modification

- *The differences are in how firebreak does packet capture and guard control*

Basic Firebreak insight: *IP Indirection*

- In normal IP, the packet source of a packet uses the IP address of the packet target destination
- In Firebreak, the destination IP address used by a packet source routes packets to a nearby defense box, *not the target*!!!
  - Defense box is called a “firebreak”
- The firebreak maps this address into the true target address, and tunnels the packet to the target

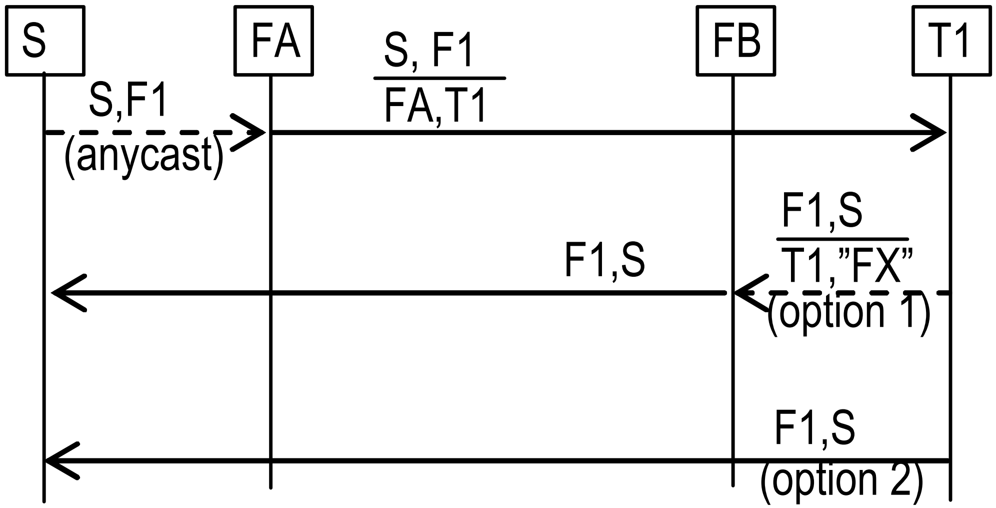
Firebreak concept



Firebreak requirements

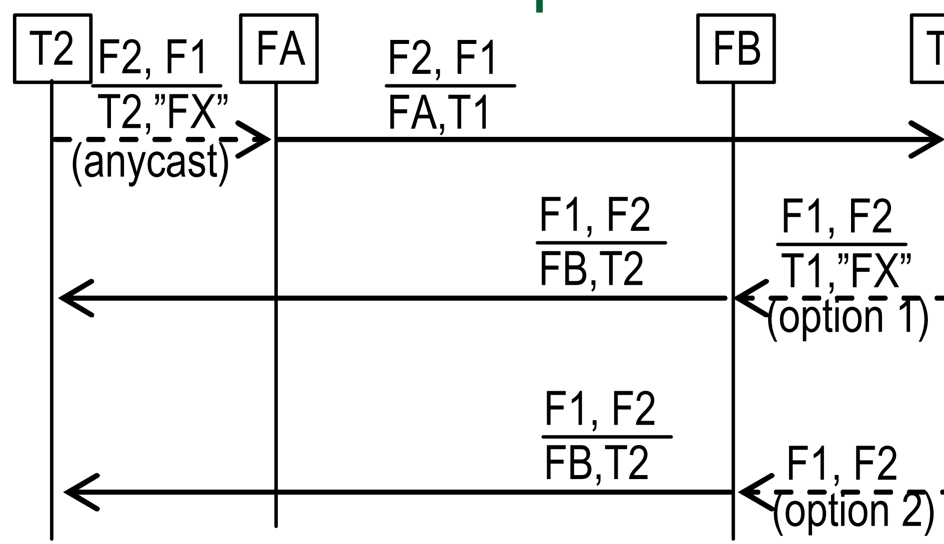
- There is one “firebreak address” for every “target address”
  - Every firebreak must know all such mappings
- Every firebreak must advertise all firebreak addresses into the routing infrastructure
  - This is what causes packets to be routed to the nearest one (and to be quickly rerouted should a firebreak fail)
  - To scale, target addresses must come from large blocks of addresses
- Target addresses must be IP reachable from firebreaks, but **not** from normal source hosts
  - Done by “scoping” routing updates

Tunneling strategies



Packets between a protected host and a non-protected host

Packets between two protected hosts



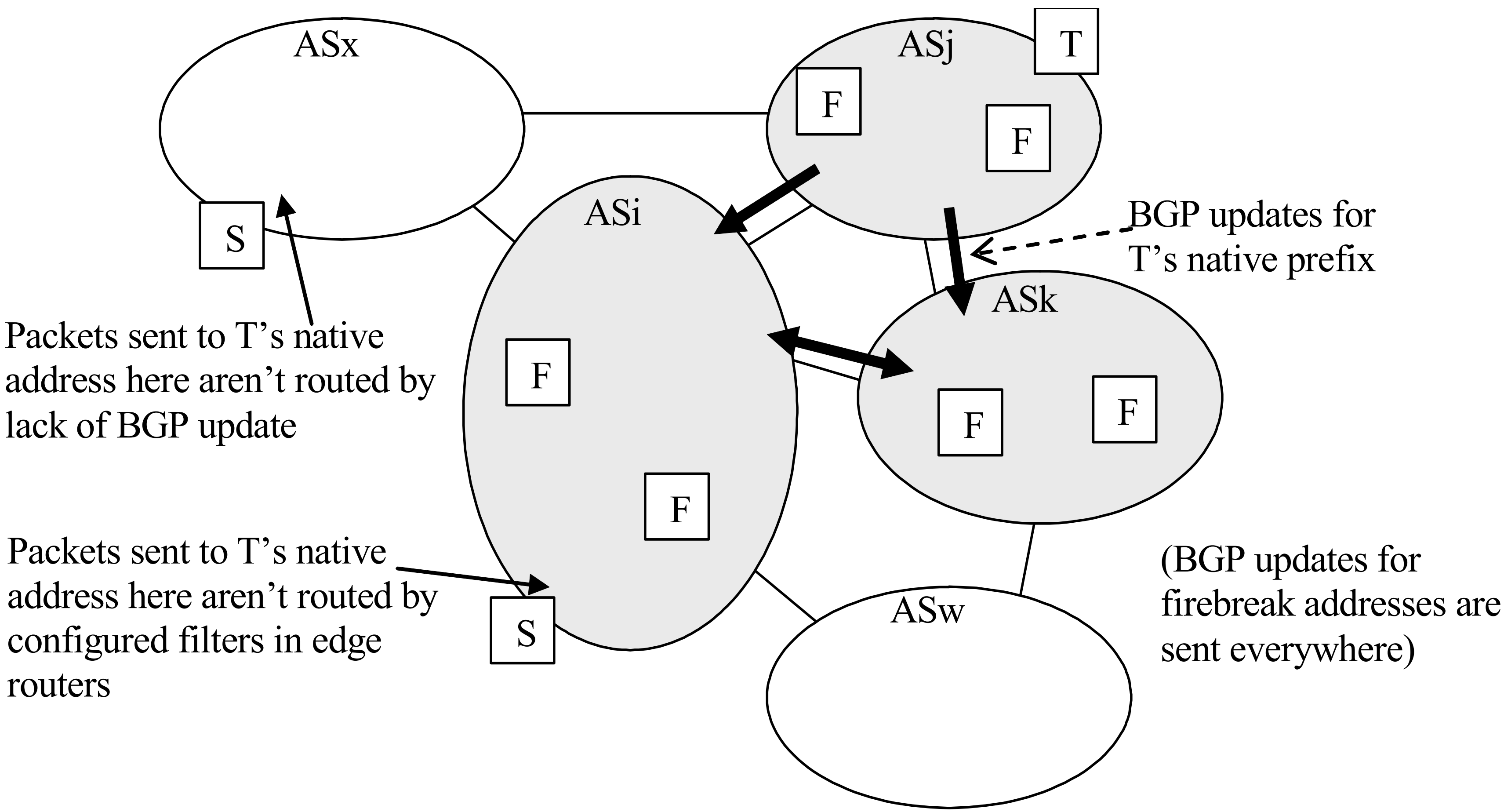
S = unprotected endhost addr
T1, T2 = protected endhost addrs
F1, F2 = firebreak addrs that map into endhost addrs
FA, FB = Individual firebreak addrs
FX = Generic (anycast) firebreak address (not mapped)

Scoped routing updates

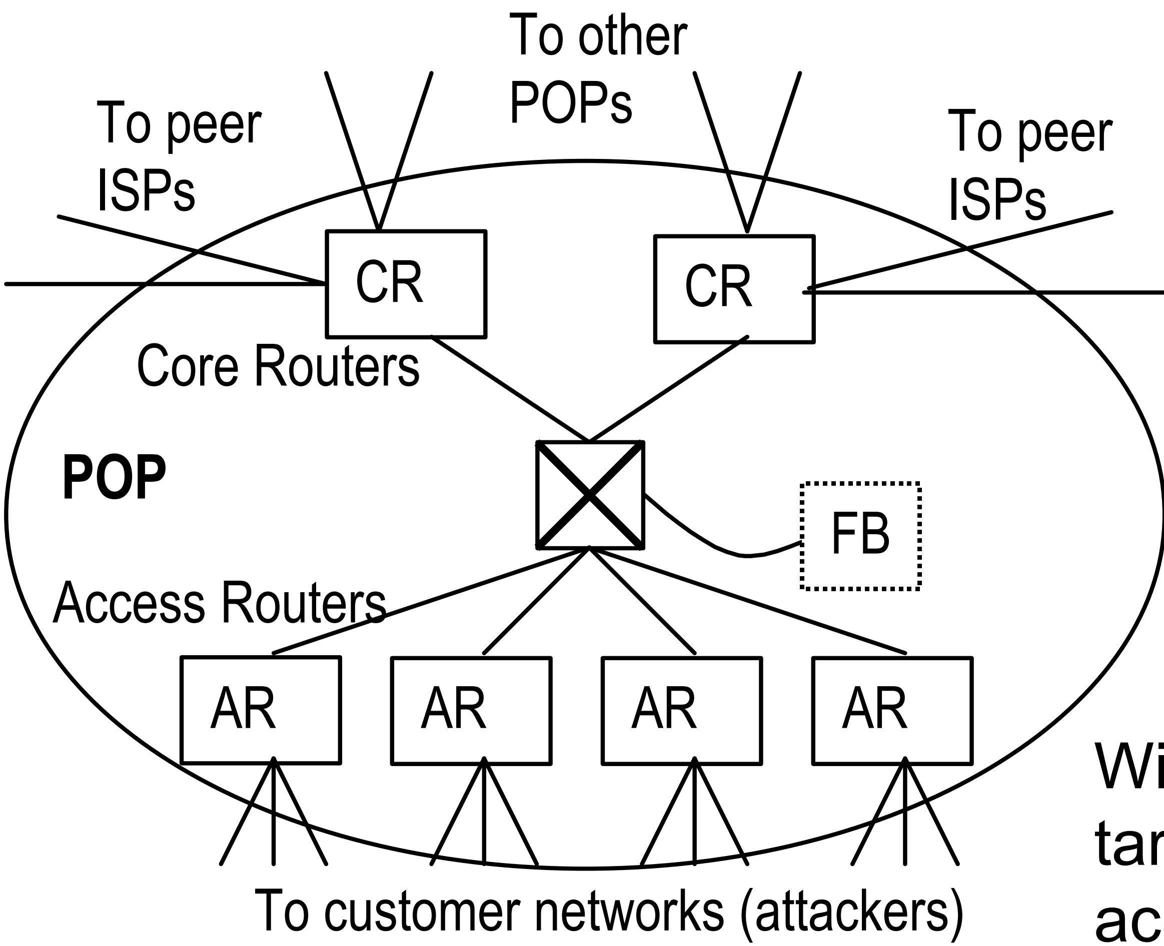
Two cases:

1. ISPs without installed firebreaks
   - Simply withhold all eBGP updates for target addresses
   - As a result, the entire ISP drops packets with target addresses
2. ISPs with installed firebreaks
   - Withhold iBGP updates for edge routers
   - Deploy firebreaks “behind” edge routers

Inter-AS BGP router configuration



Scoped routing updates within a typical ISP POP



Provide routes to targets to core routers

Deploy firebreak "behind" access routers

Withhold routes to targets from access routers

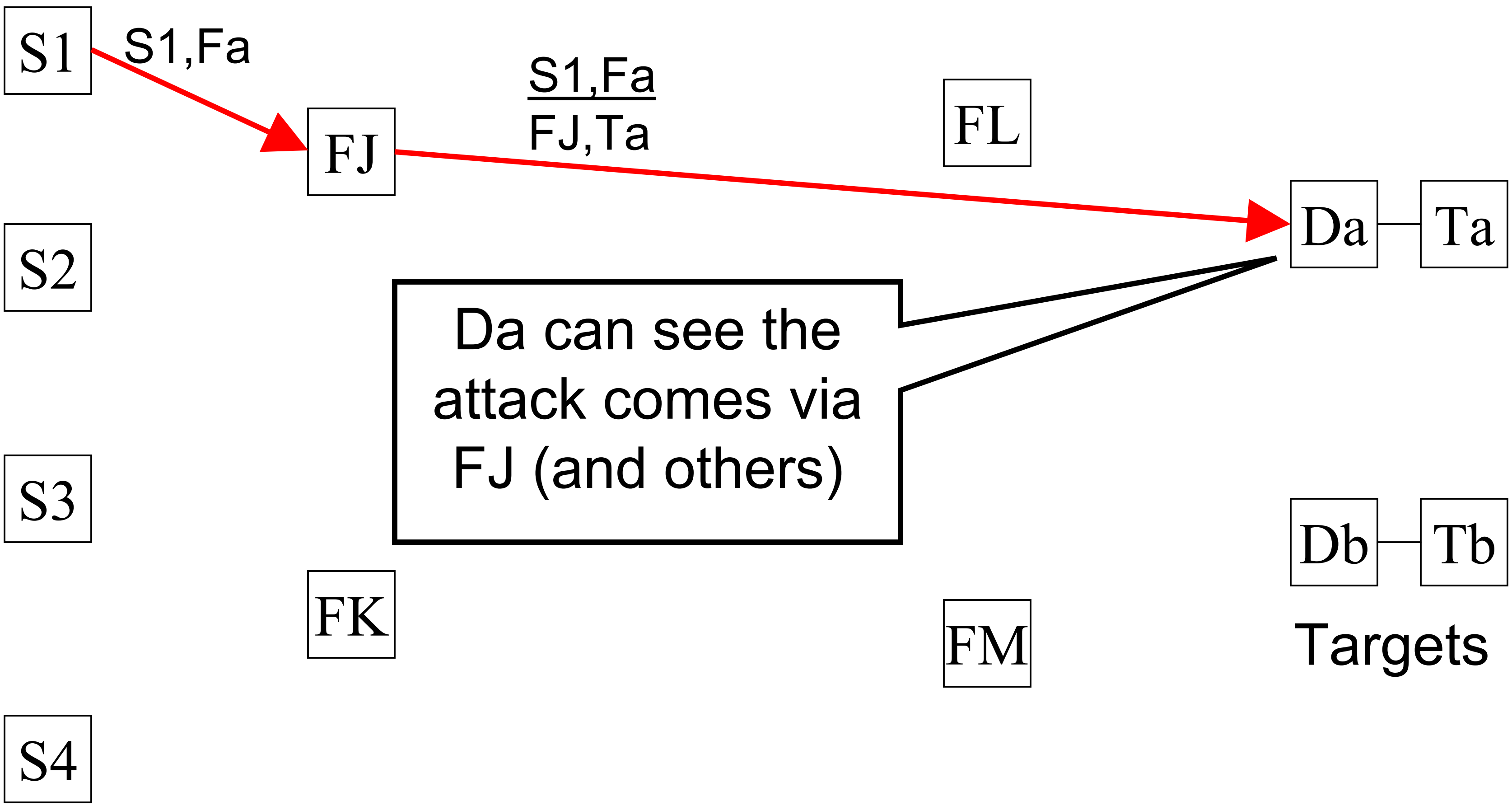
Handling an attack

- Detectors near targets detect the attack
- They can tell which firebreaks the attack is coming through, and the nature of the attack
- They instruct the corresponding firebreaks to execute defensive actions (for packets to the attacked target only)
  - Drop packets with spoofed source addresses
  - Fair queue packets to limit attack packets
    - Time scale varies depending on scope of attack!
  - Etc.

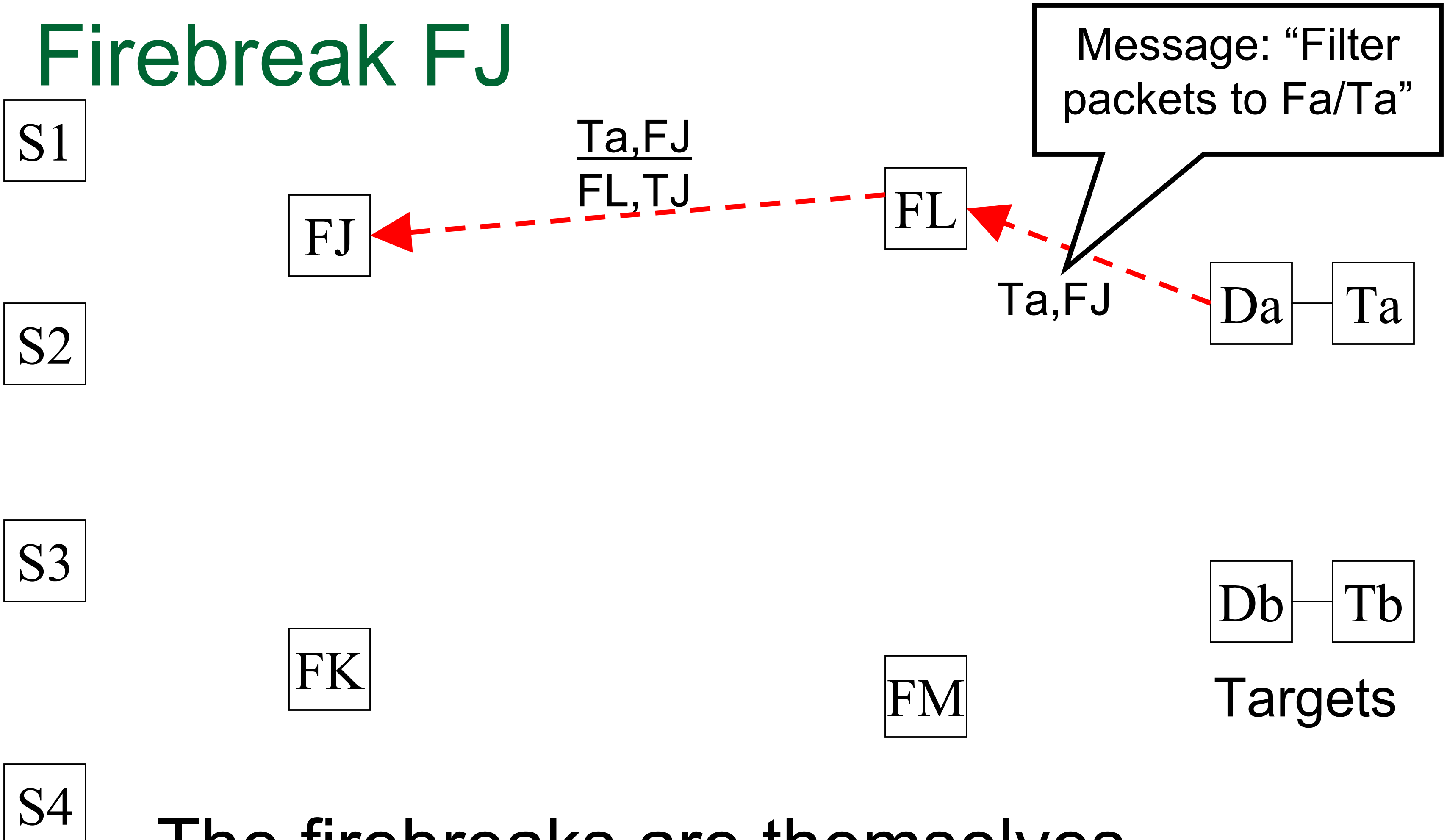
Authenticating control messages

- Uses concept of “return routability”---a Detector can only install filtering rules about addresses where it can be reached
  - Cheriton/Argyraki (Stanford) has proposed something similar
    - (In the context of an unwieldy router traceback architecture)
- This allows a simple, lightweight nonce challenge of Detectors by Guards
- Attacker must be in the physical paths between Guard and every Detector it wants to spoof

An attack from S1 (and others)

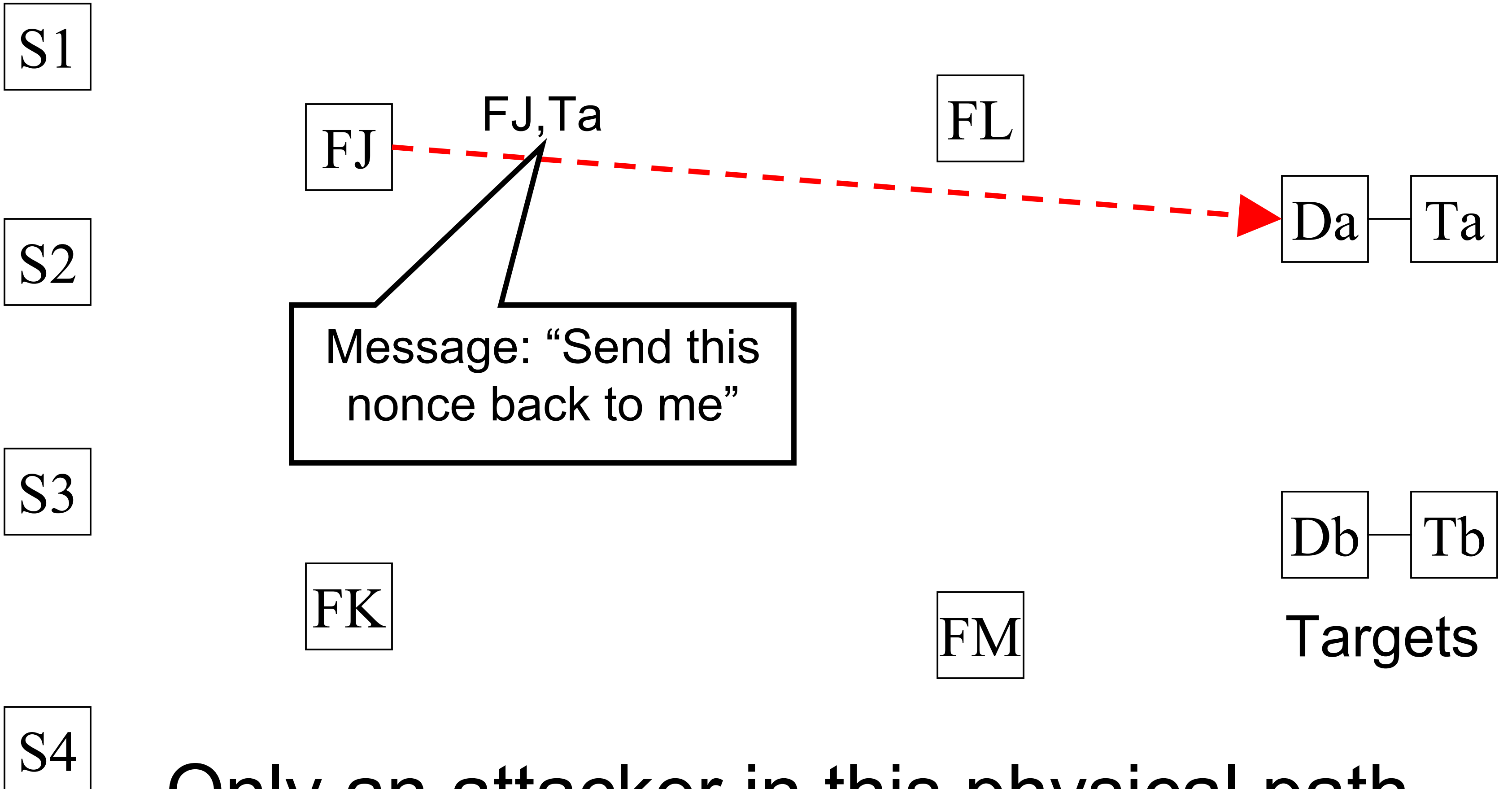


Detector sends a control message to Firebreak FJ



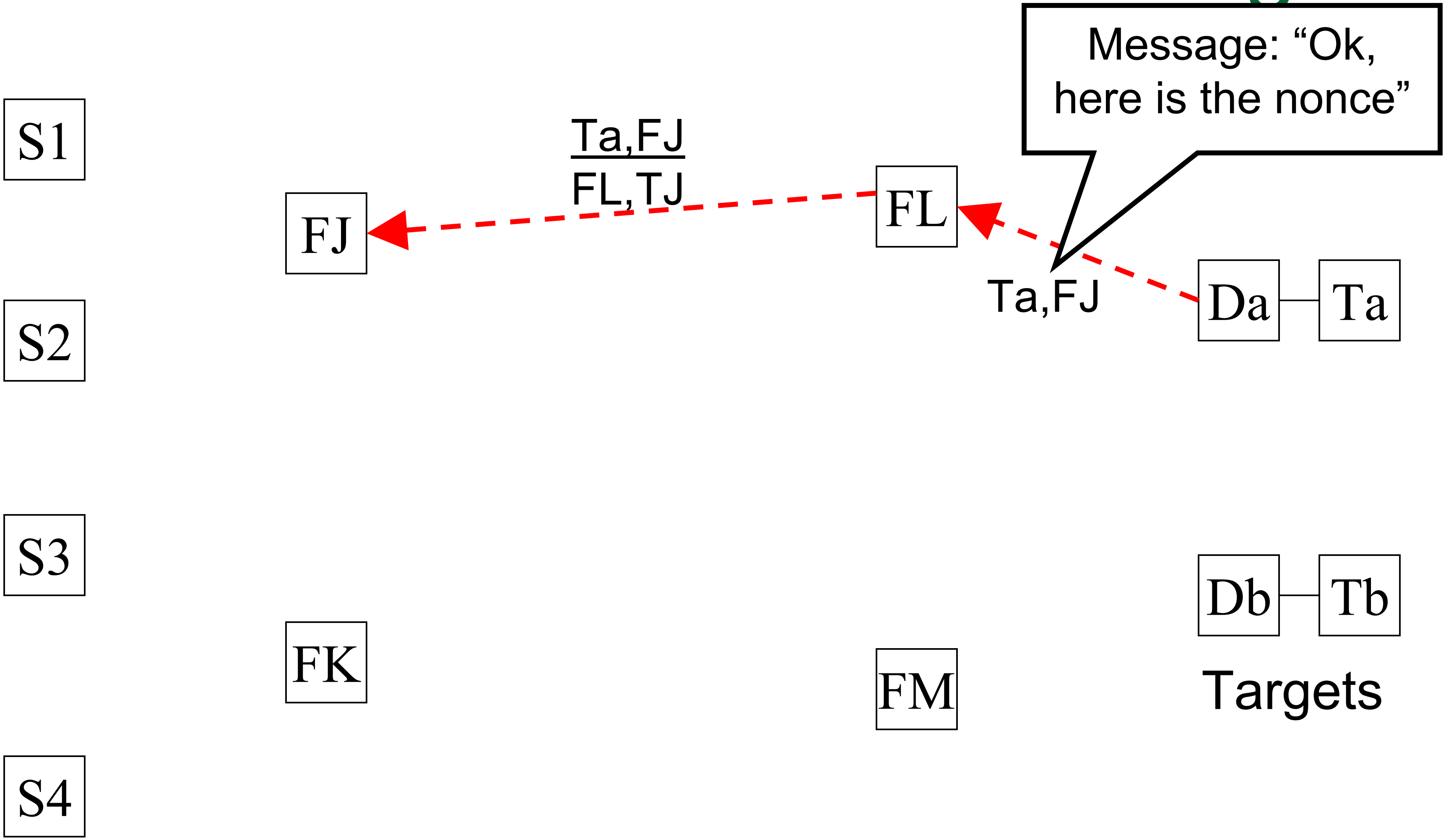
The firebreaks are themselves protected by the firebreak system!

Firebreak sends a random challenge to the detector

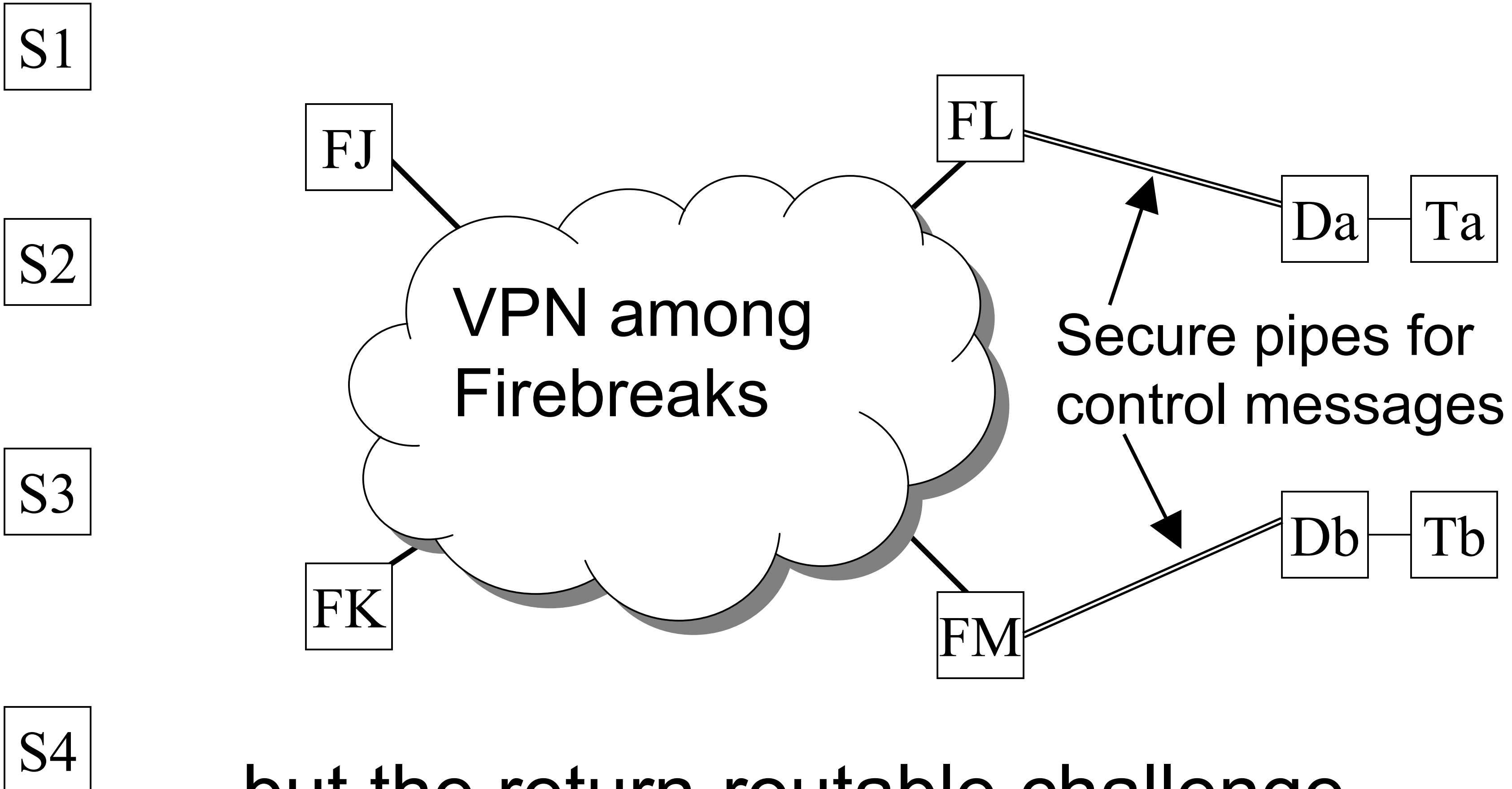


Only an attacker in this physical path can spoof this message's reply

Detector answers the challenge



Additional security is possible…



…but the return-routable challenge should still exist

Broadly, Riverhead versus Firebreak tradeoffs

- Pro Firebreak:
  - Firebreak does not require full “perimeter” coverage
    - In either diversion or long diversion form
  - Firebreak is more amenable to multi-ISP defense
    - Though the immediate business model is CDN, not ISP
- Pro Riverhead:
  - Riverhead does not change packet path in “peacetime”
    - Firebreak requires packets to go through firebreaks

A little of both…
This really is our punch line!

- Each ISP install enough Guard capacity to defend against a small-to-medium attack
  - Using Riverhead-style diversion at all egress points
  - Market this as differentiator…all customers get some DDoS protection for no extra charge
- ISPs combine in a *Firebreak Alliance*
  - Same guards ***also*** deployed Firebreak-style
  - Market to content providers as a paid service for protection against massive attacks

Issues: Addressing

- Firebreak requires two addresses for every protected host
  - Yep
  - (IPv6 would be great here…just divide the address space into half with one-to-one mapping between the target and firebreak portions)
- Firebreak addresses must come in large contiguous blocks
  - To avoid large edge router tables
  - Requires forethought and planning by provider

Issues: Deployment

- Requires detunneling at target
  - Suggests a for-pay protection model
  - Detunneling is not an expensive procedure
  - Many tunnels terminate at target, so require a lighter weight model than routers currently have
- Does large-scale anycast work well?
  - Can we control load at Firebreaks?
  - Are there any BGP dynamics issues?
  - Need experimentation

Issues: Scaling

- All packets must traverse firebreak
  - But, in peacetime, they only require tunneling
  - Quite lightweight
  - Normal methods deal with firebreak failure
- Control message load at attack time
  - Potentially thousands of firebreaks must be notified
  - Even so, this doesn't strike us as a problem…

Conclusions

- Firebreak is a promising, IP-level DDoS guard deployment strategy
- Allows for a multi-ISP deployment
- Does not require full perimeter coverage
- Business model:
  - CDN or "ISP Alliance"
  - Initially fits a target pays premium service model
  - But could become commodity as functionality is move into edge routers

THANKS!

- Questions / Comments???