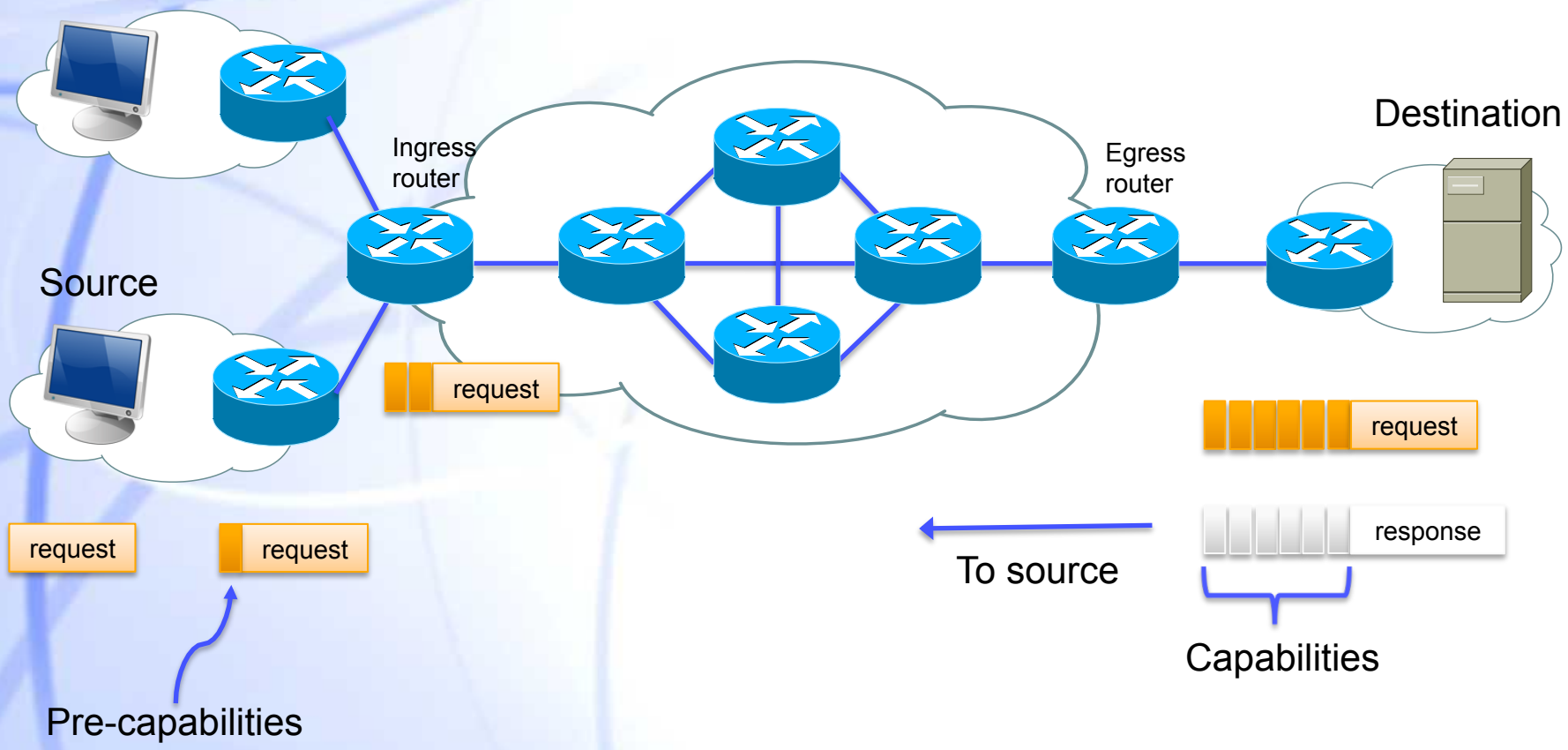
NETWORK CAPABILITIES

TVA

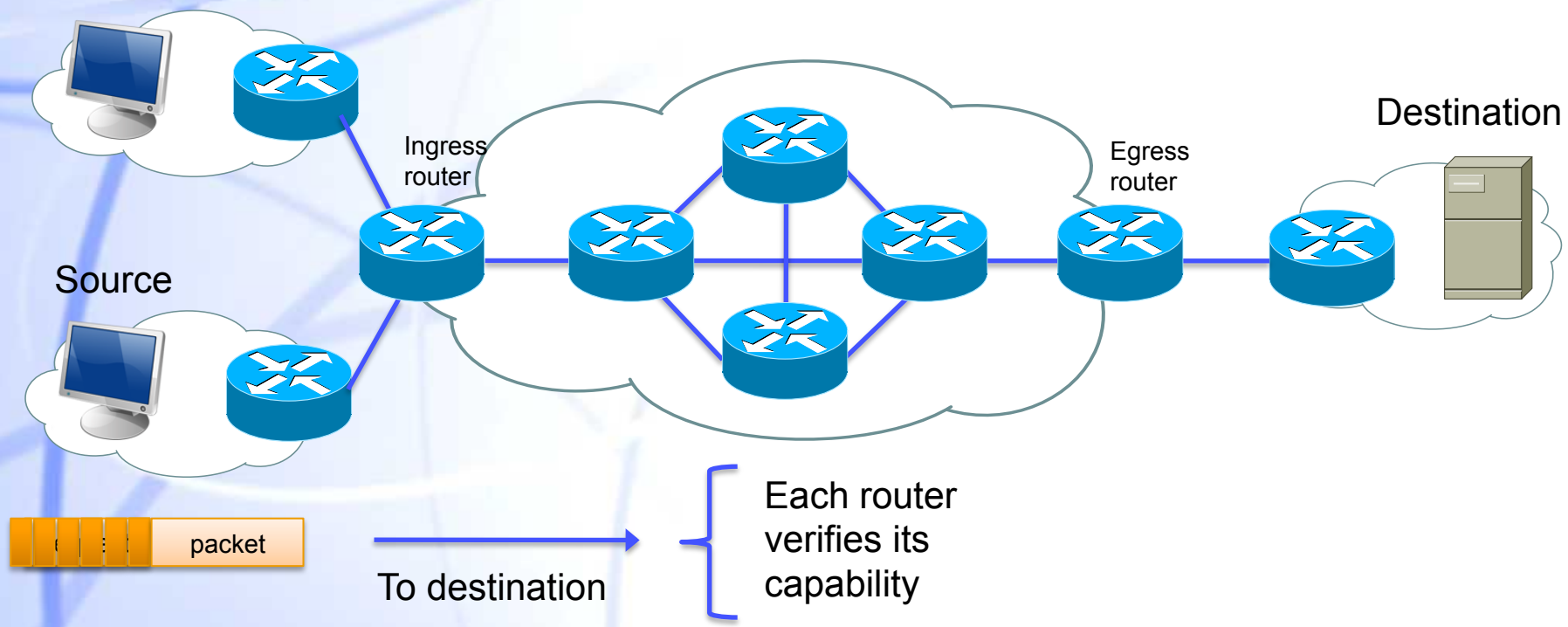
Introduction

- Described in:
  - X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *Proceedings of ACM SIGCOMM, August 2005.*
- TVA stands for Traffic Validation Architecture (inspired on Tennessee Valley Authority)
- Carefully designs and evaluates a more complete capability-based network architecture
- TVA counters broader set of attacks:
  - Flooding of the setup channel, router state exhaustion, network bandwidth consumption, etc.

Bootstrapping capabilities



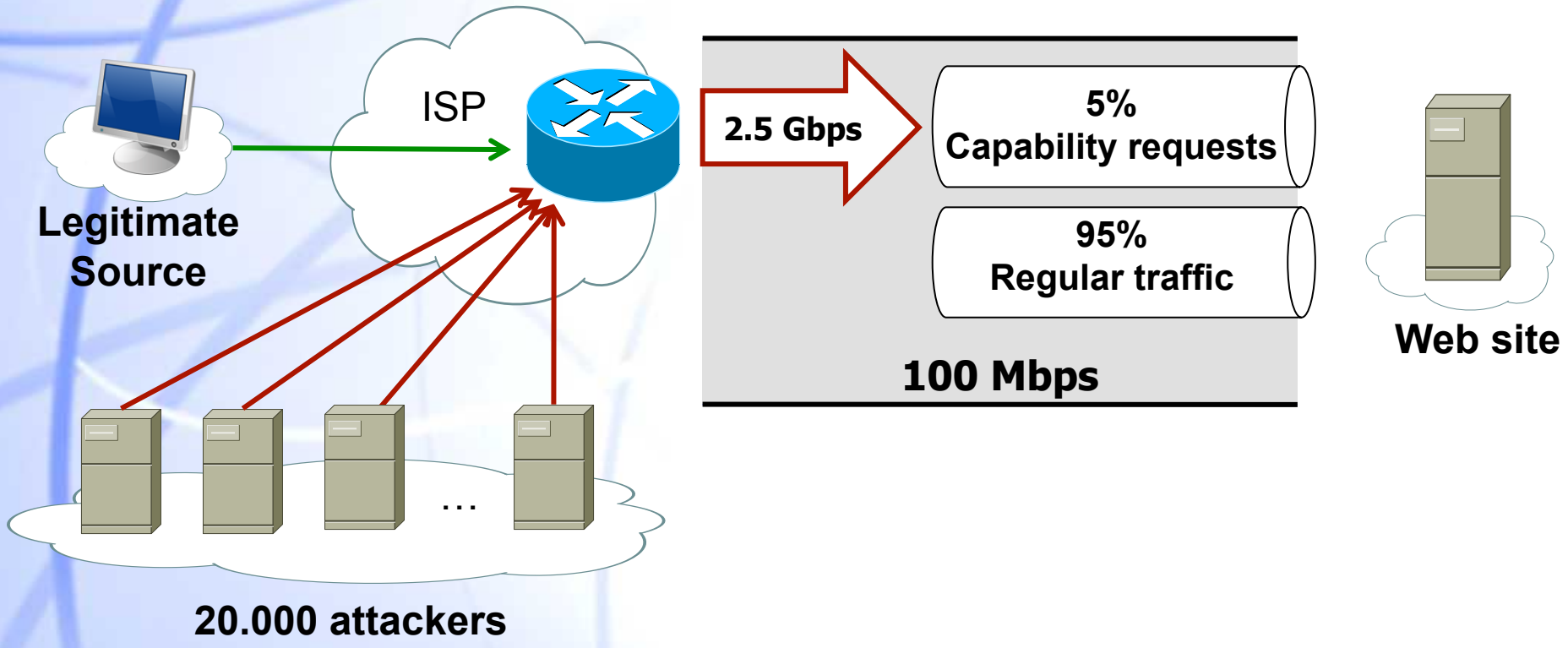
Bootstrapping capabilities



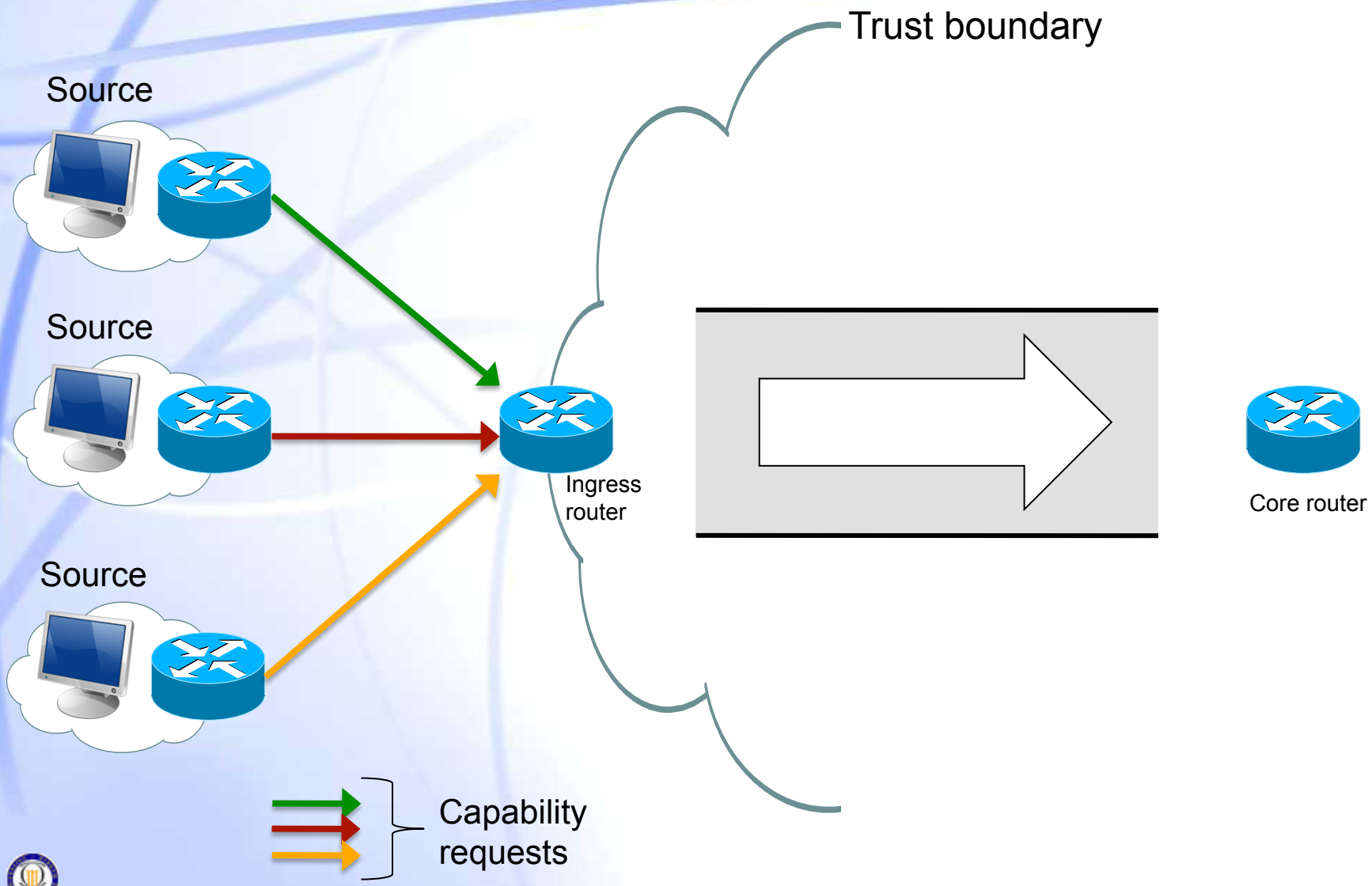
Bootstrapping capabilities (II)

- The initial request channel should not open an avenue for DoS attacks, by
  - Flooding a destination
  - Denial of Capability
- Solution to first issue:
  - Request packets should comprise a small fraction of bandwidth
  - Requests are rate-limited at every network location (5% of the link capacity)

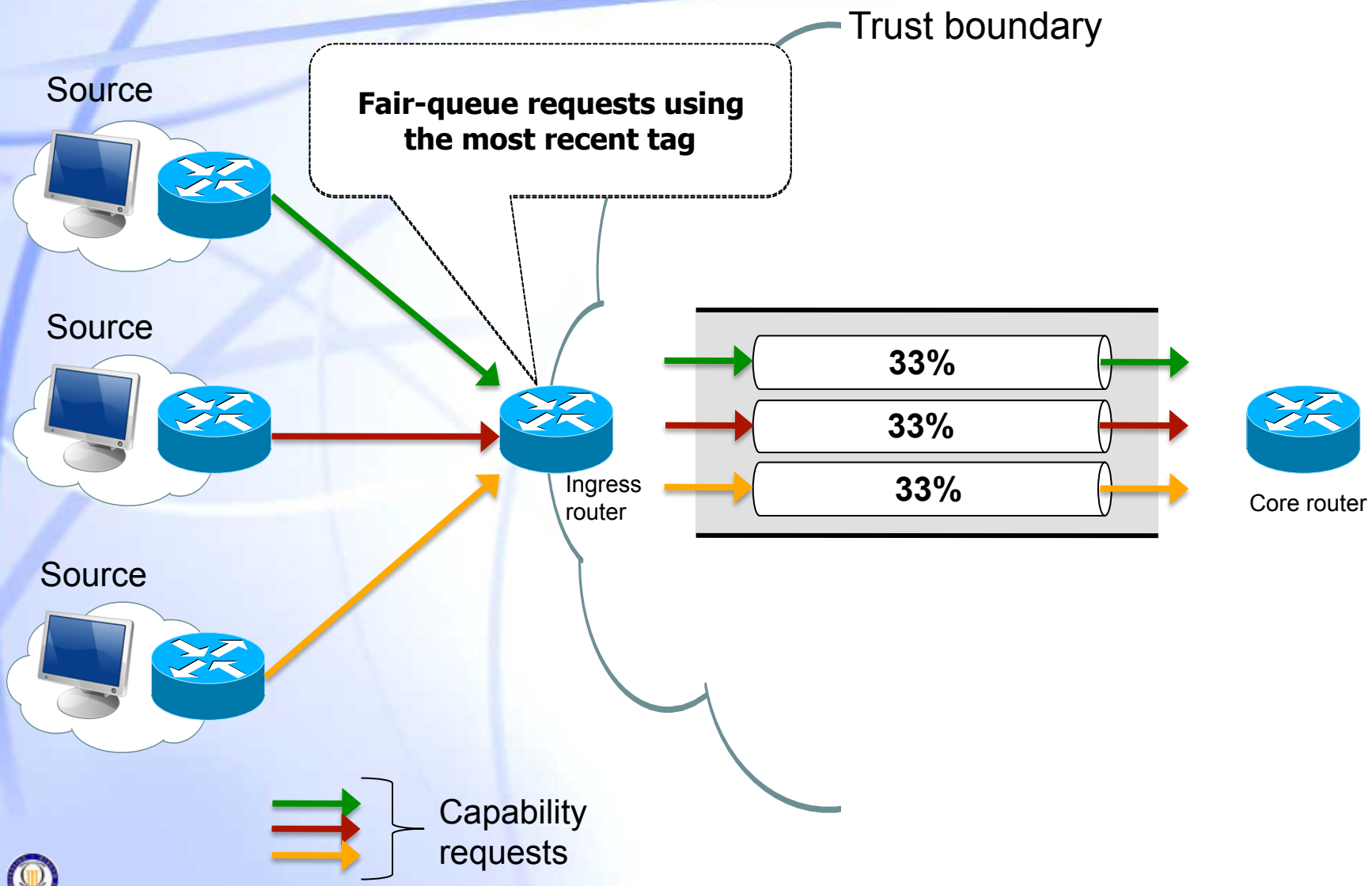
Denial of Capability: review



Bootstrapping capabilities (III)



Bootstrapping capabilities (III)



Unforgeable capabilities

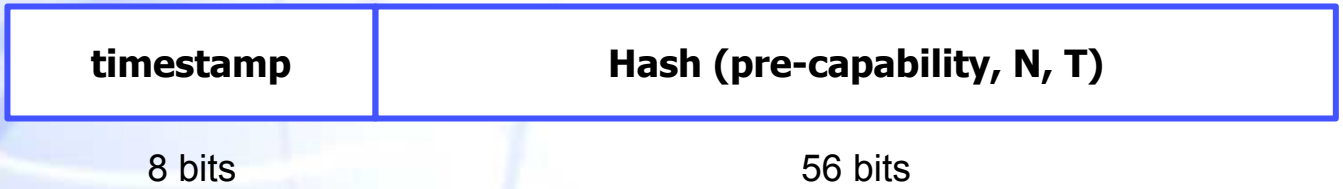
- **Attackers should not:**
  - Forge capabilities
  - Make use of a capability stolen or transferred from other parties
- **Solution:**
  - Each router that forwards a request packet attaches a pre-capability

Unforgeable capabilities (II)

- The destination receives a ordered list of pre-capabilities:
  - Bounded to a network path, source IP address and destination IP address
- If the destination authorizes the request, it returns back to the sender an ordered list of capabilities
  - Capabilities allow the sender to send packets towards the destination, through the network path

Fine-grained capabilities

- Capabilities grant the right to send up to N bytes within the next T seconds
  - E.g. 100 KB in 10 seconds
- Destination converts pre-capabilities to capabilities



- {Capabilities, N, T} are returned to authorize the sender

Capability validation

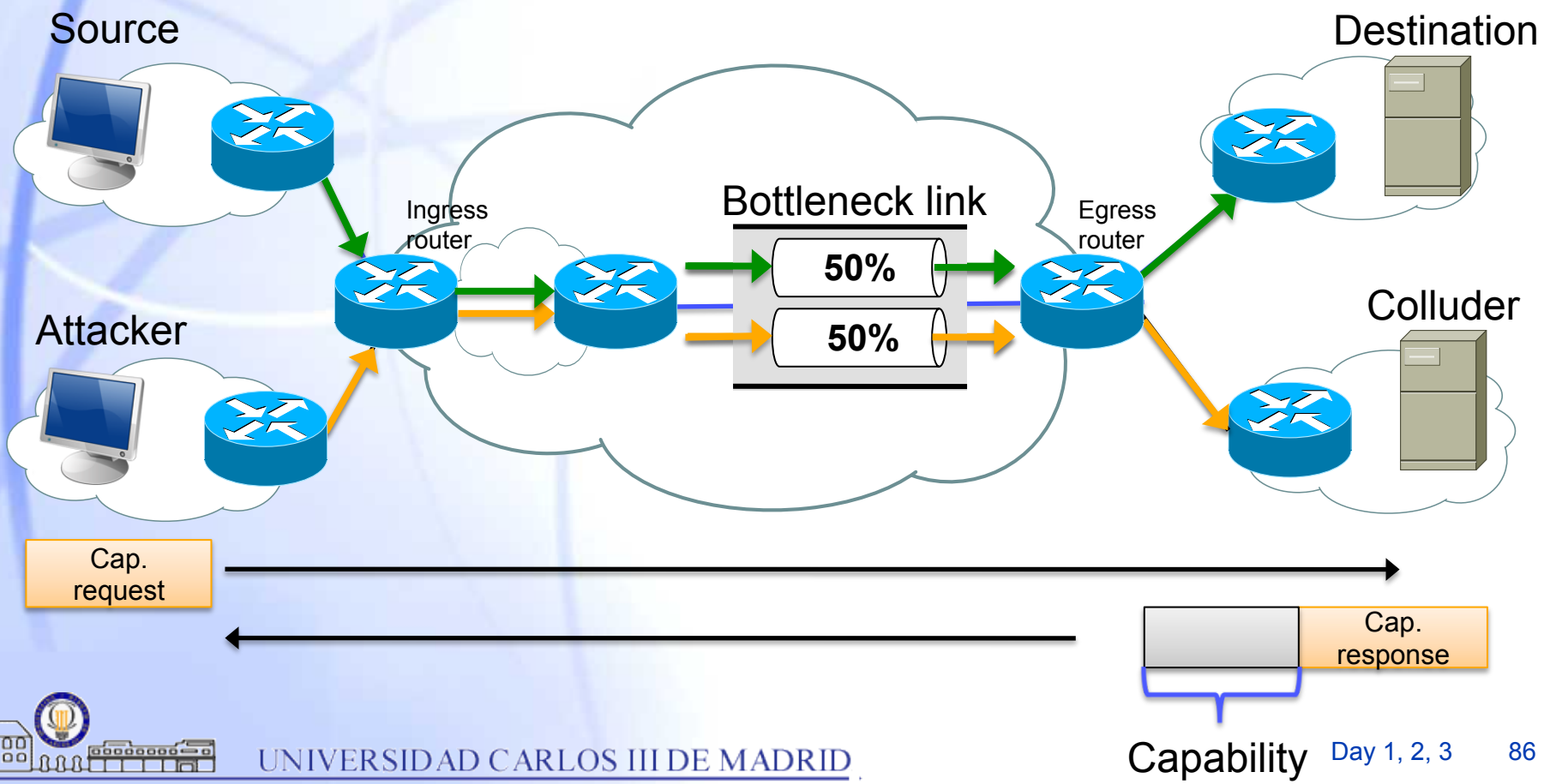
- Source includes the list of capabilities, N and T within each packet
- A router on the path:
  - Uses its secret to recompute its pre-capability:
    - ✓ Source and destination IP addresses are obtained from the packet
    - ✓ The timestamp is obtained from the capability
  - Uses the pre-capability to recompute the capability:
    - ✓ N and T are included in the packet
    - ✓ Checks if the result matches the capability value
  - Checks if the capability has expired:
    - ✓ From N and T

Bounded router state

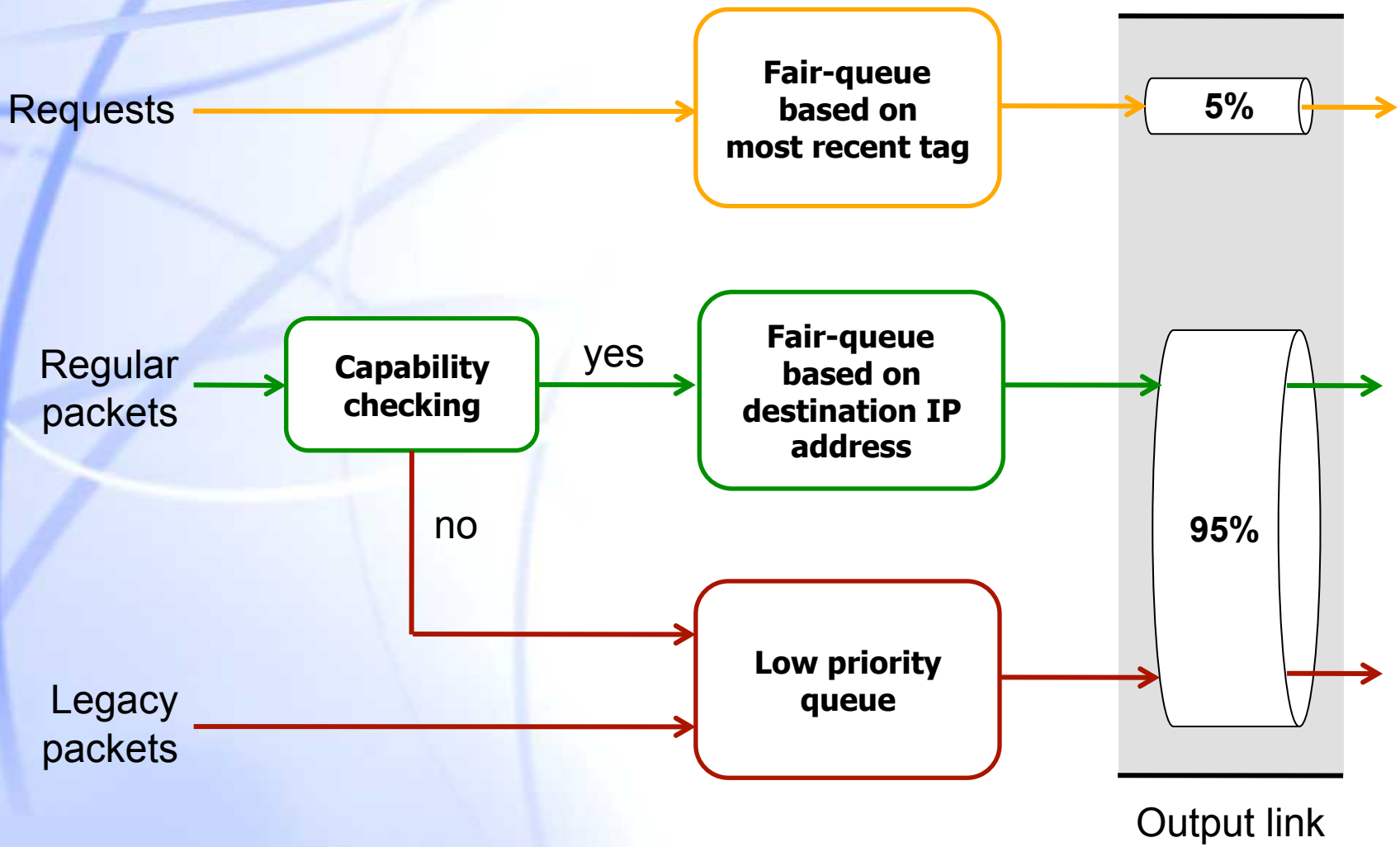
- Routers check that capabilities are not used for more than N bytes
  - Router state is required
  - Attackers should not exhaust router state
- An algorithm is designed that bounds the bytes sent using a capability:
  - It uses a fixed amount of router state
  - High-level idea: keep state only for flows with valid capabilities that send faster than N/T
  - In the worst case, a capability may be used to send 2N bytes

Balancing authorized traffic

- The proposal is vulnerable to floods of authorized traffic
- Solution: fair-queuing based on the destination IP address



Queue management at routers



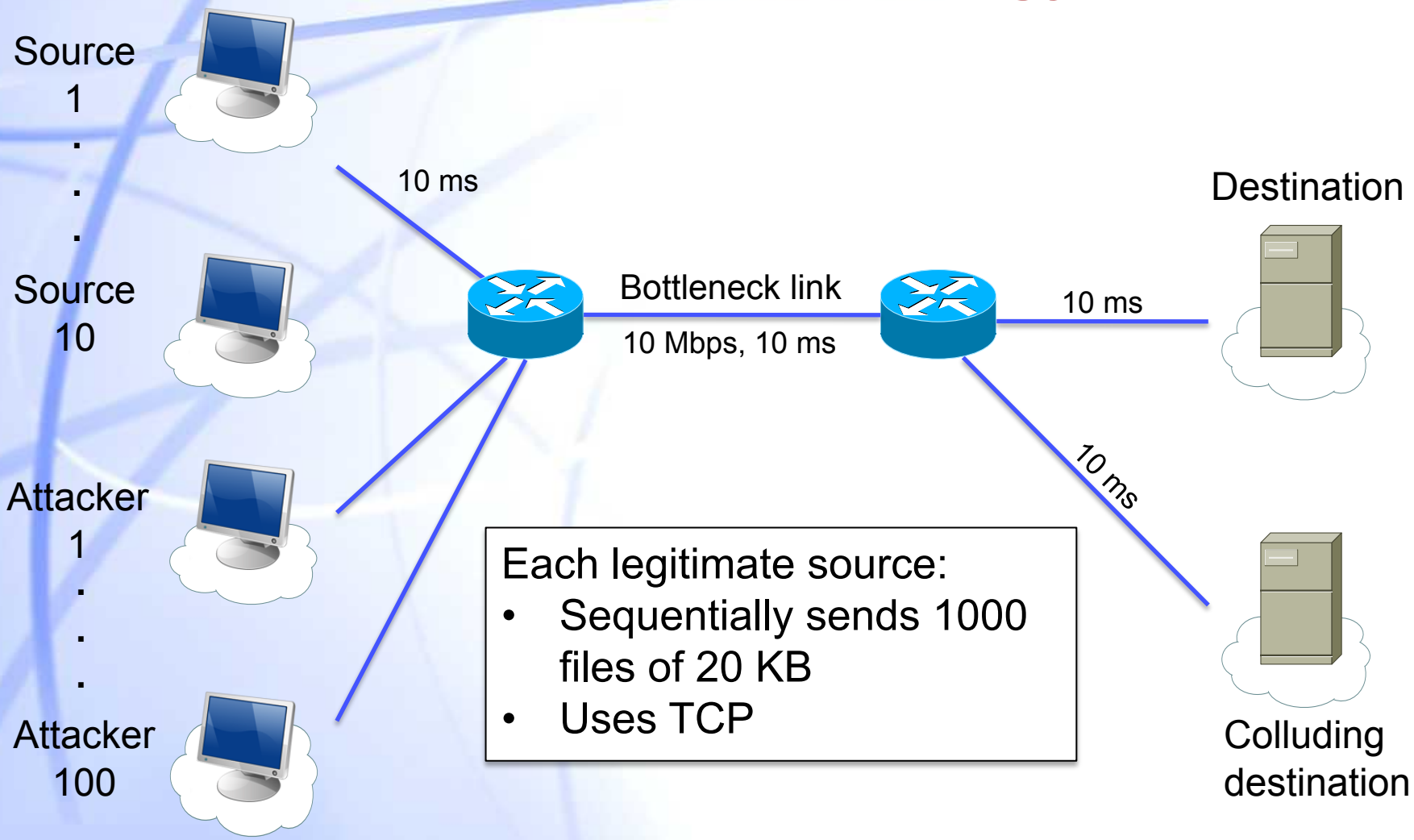
Efficient capabilities

- When a sender obtains capabilities, it generates a random flow nonce
  - The nonce is included in the packets
- A router caches the capability relevant information and the flow nonce
- Subsequent packets carry the flow nonce and omit the list of capabilities
- But cache can expire!
  - Senders model cache expiration at routers

Route changes and failures

- The design accommodates route changes and failures:
  - A packet may arrive to a router that has no associated capability state:
    - ✓ The packet is demoted to the same priority as legacy traffic
    - ✓ The destination notifies the demotion to the sender
    - ✓ The sender re-acquires new capabilities

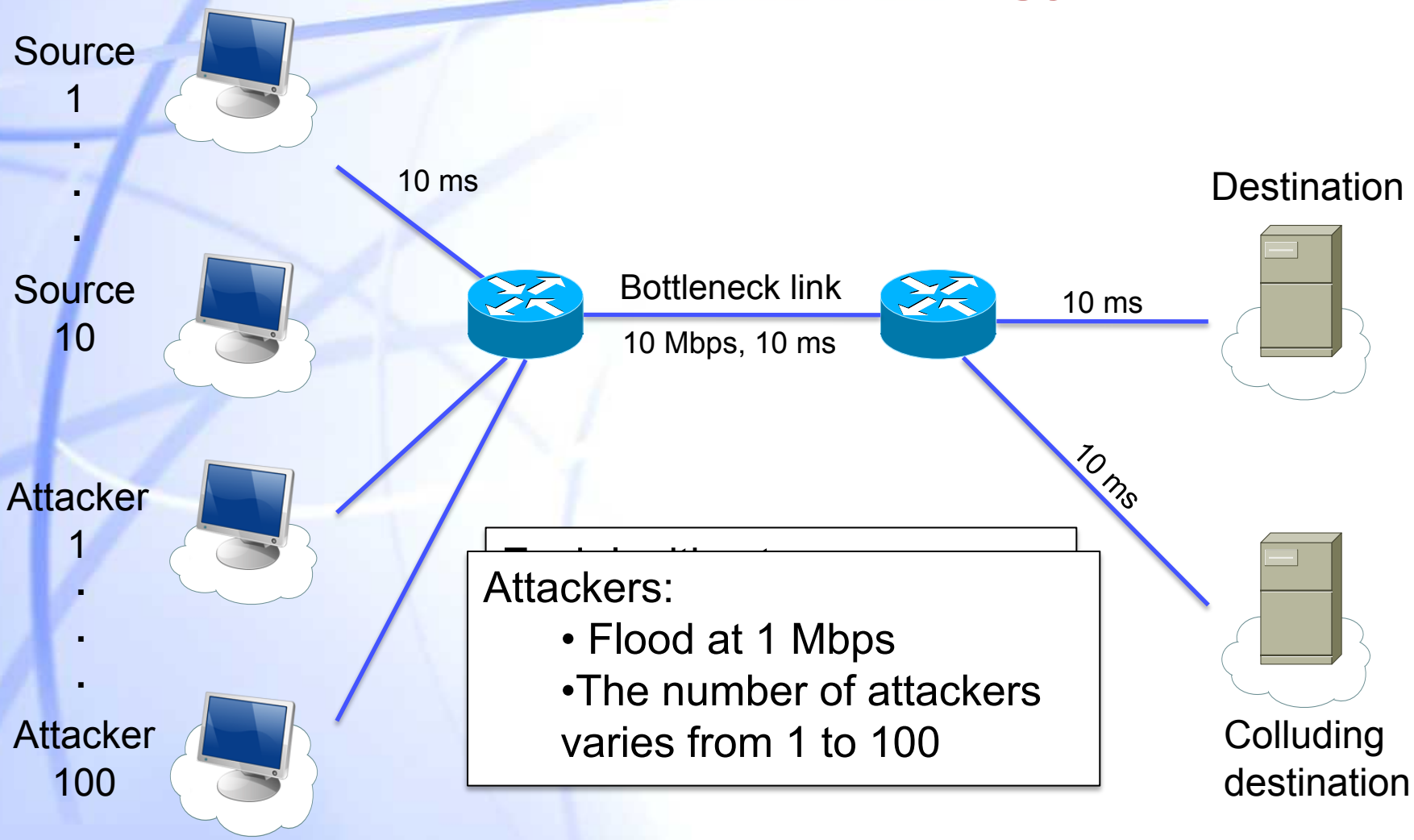
Evaluation: simulation topology



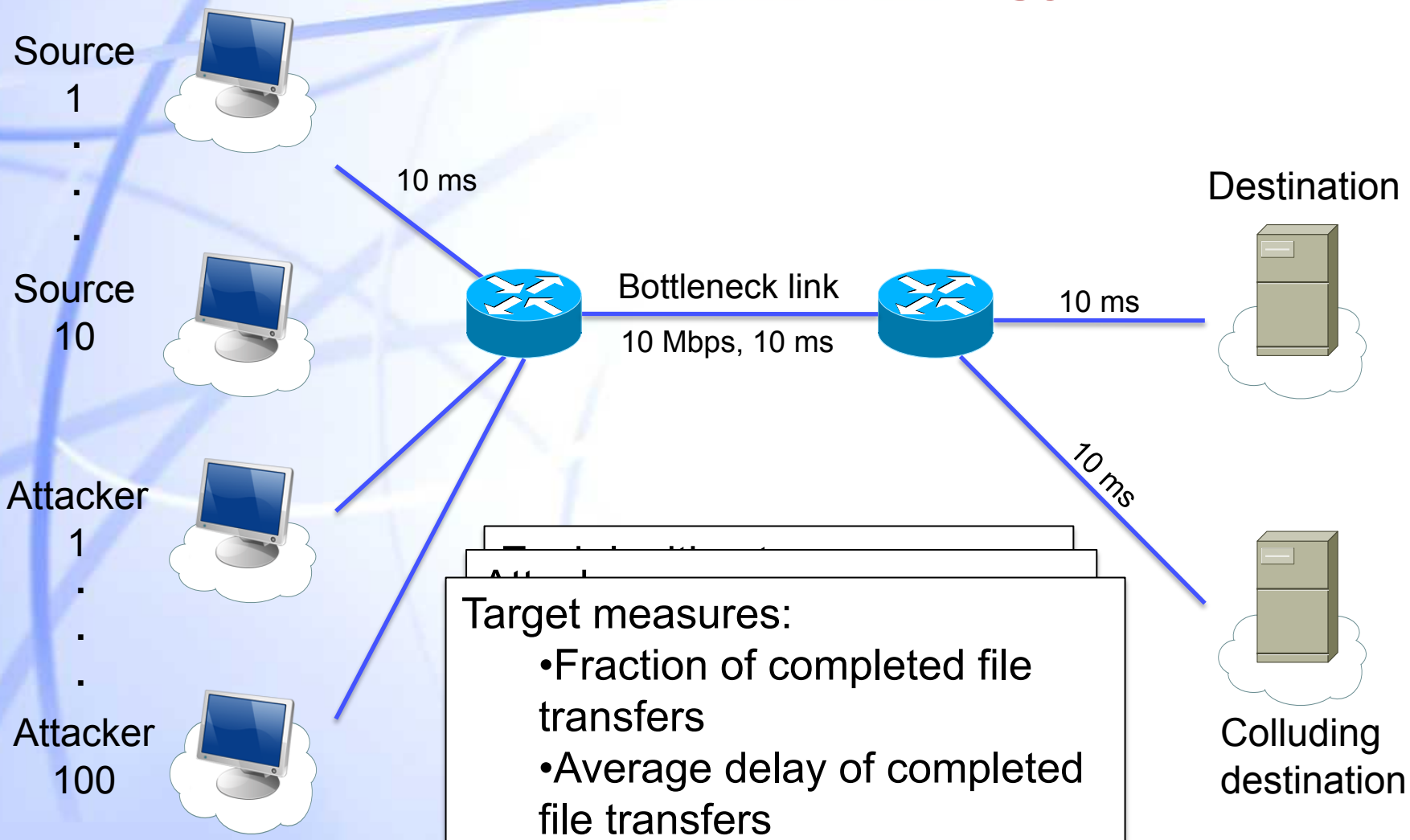
Each legitimate source:

- Sequentially sends 1000 files of 20 KB
- Uses TCP

Evaluation: simulation topology



Evaluation: simulation topology

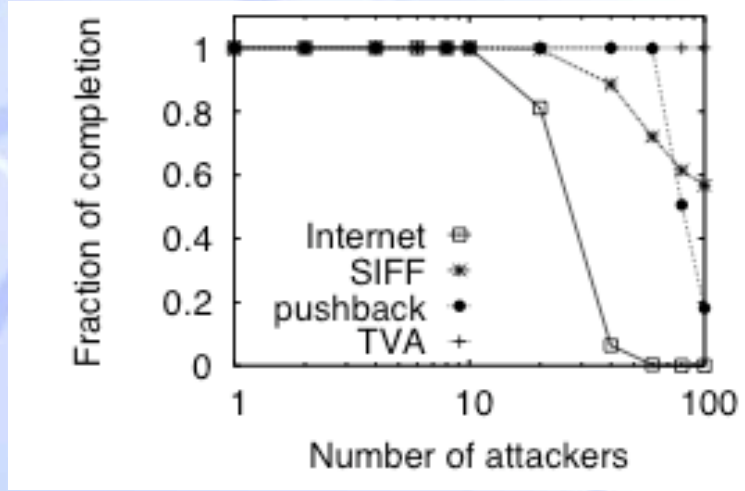


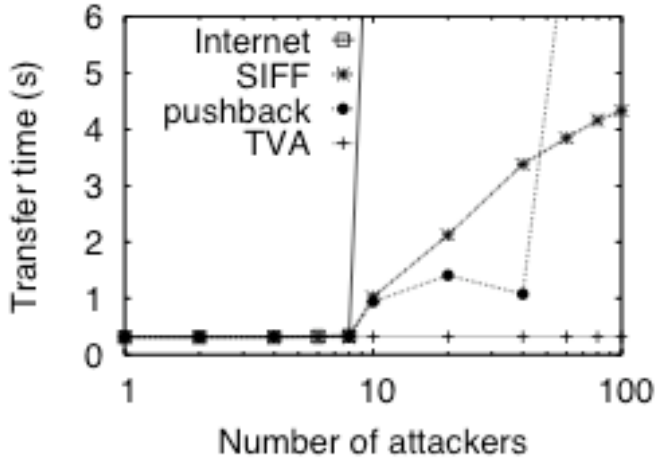
Target measures:

- Fraction of completed file transfers
- Average delay of completed file transfers

Evaluation: legacy packet floods

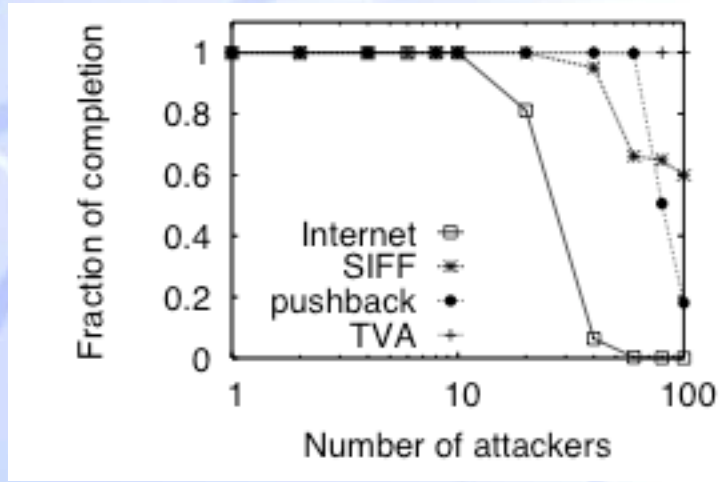
- Each attacker floods the destination with legacy traffic at 1 Mbps

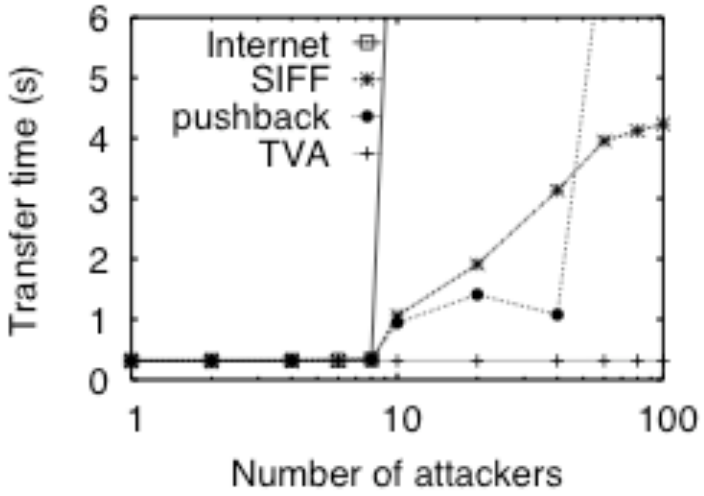




Evaluation: request packet floods

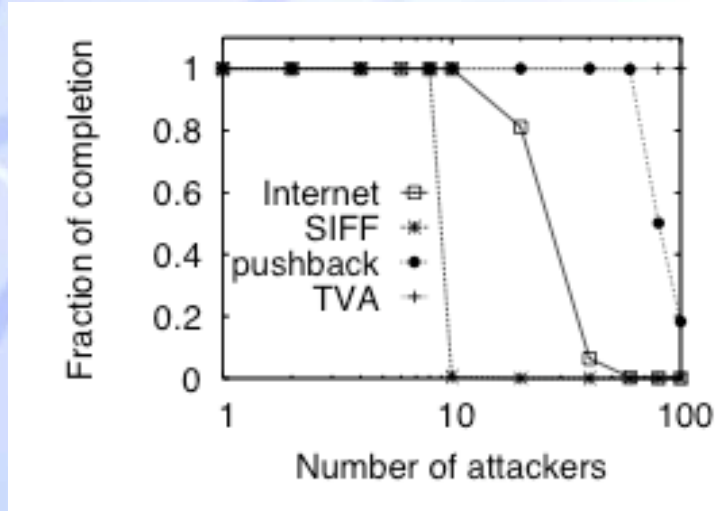
- Each attacker floods the destination with request packets at 1 Mbps

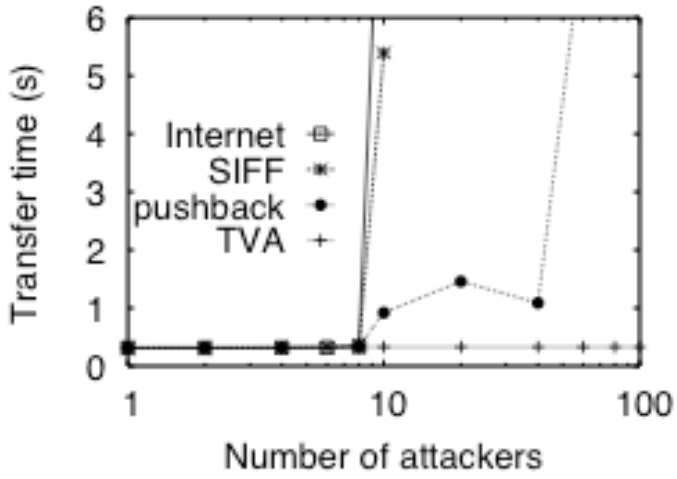




Evaluation: authorized packet floods

- Attackers cooperate with a colluding destination
- Colluder grants capabilities to attackers, allowing them to send authorized traffic at their maximum rate

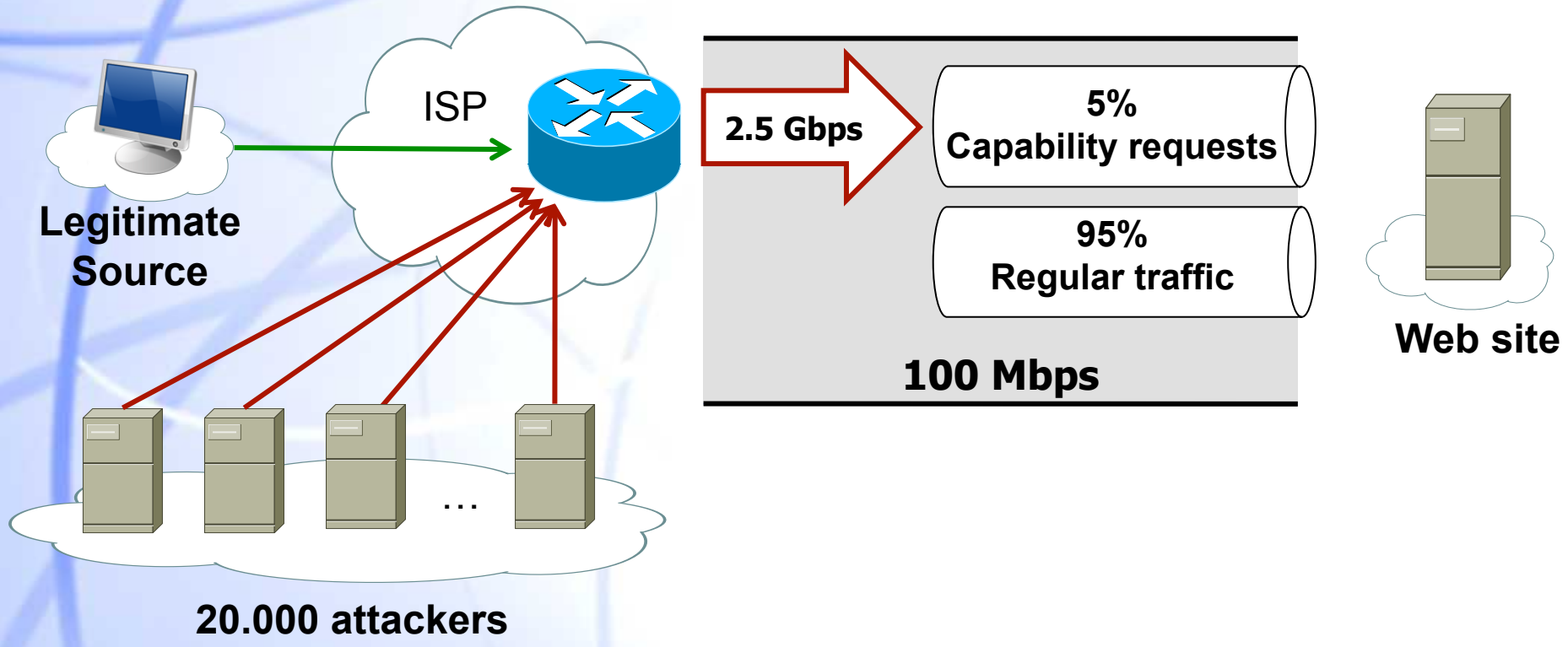




NETWORK CAPABILITIES

Portcullis: addressing the DoC attack

Reminder: Denial of Capability (DoC)



Introduction

- **Described in:**
  - **B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. "Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks". In *ACM SIGCOMM, 2007***
- **Portcullis augments the proposed capabilities based solutions with puzzle based protection against DoC**

Design overview

- **The sender:**
  - Generates a puzzle, using a puzzle generation algorithm
  - Computes the solution to the puzzle
  - The puzzle and the solution are included in the header of the request packet
- **The routers:**
  - Verify the authenticity of the puzzle and the solution
  - Give priority to requests that contain higher-level puzzles

Puzzle generation algorithm

- **Definition of the puzzle:**

  $P = H(x \;||\; r \;||\; h_i \;||\; dest\ IP \;||\; l)$

- **Where:**
  - h_i: seed
  - r: random 64-bit nonce
  - l: difficulty level of the puzzle
- **To solve the puzzle, the sender finds a 64-bit value x such that the last l bits of p are zero**
  - The sender must resort to a brute-force approach, by trying random values of x

Strategies

- **Legitimate sender strategy:**
  - Computes a solution to the lowest level puzzle and transmit a request
  - If the request fails, solve a puzzle that requires twice the computation
- **Attacker strategy:**
  - Send the highest priority puzzles possible,
  - while still saturating the victim‘s bottleneck