Content-distribution networks

Portland State
Computer Science

Strategies

- Divide and conquer
  - Partition
  - Replicate
  - Distribute
  - Load balance

Outline

1. Server partitioning
2. DNS load balancing
3. Virtual servers
4. Case studies

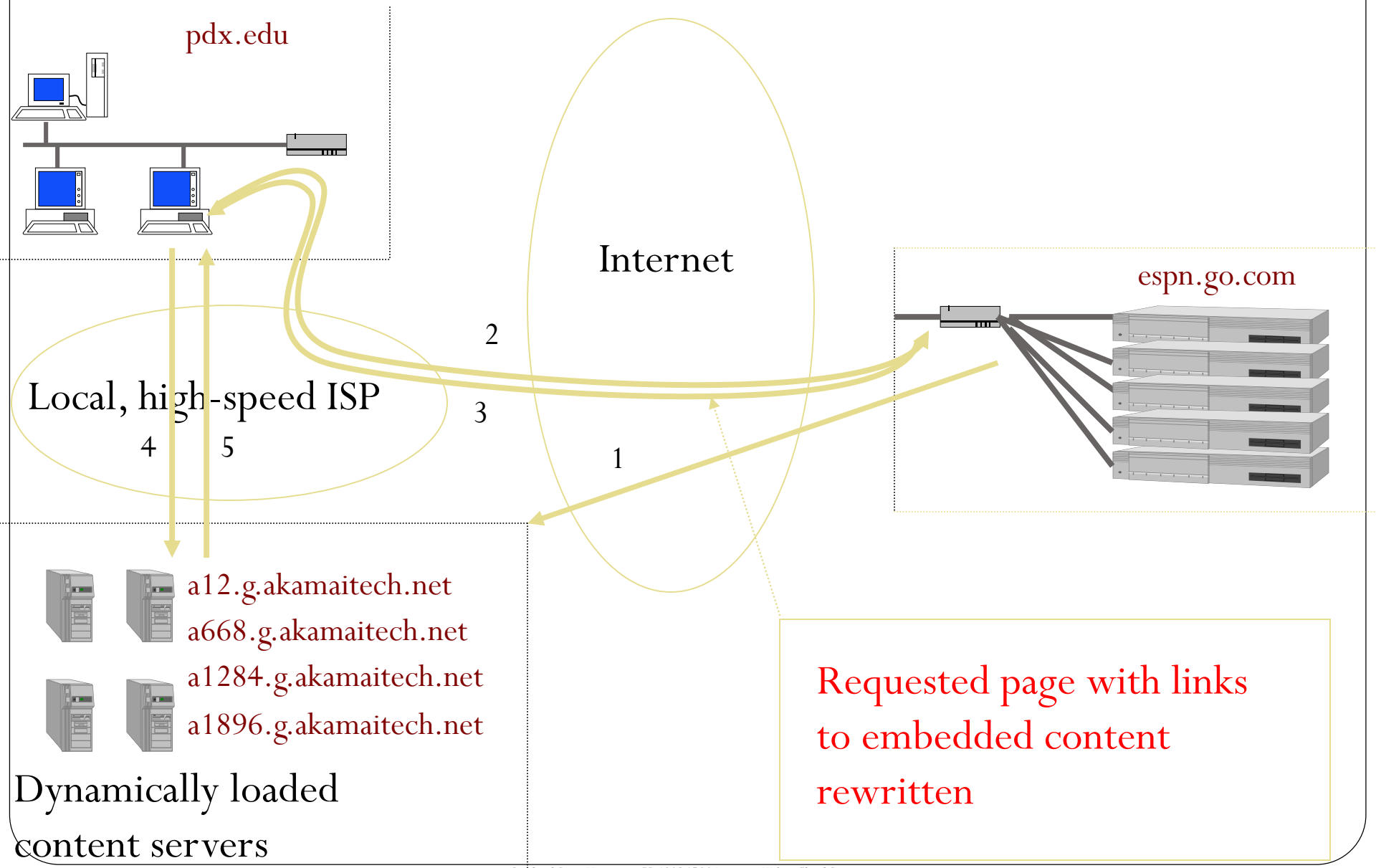
1. Server partitioning (static)

- Run a new server per resource/service
  - e.g. `www.blah.com, mail.blah.com, images.blah.com, shopping.blah.com`
- Advantages
  - Disk utilization (no need to replicate all content)
  - Cache performance
  - Better suited for DevOps, CI/CD
    - Distributed independent development/deployment etc. of "microservices"
    - Isolation of cookie policy, Content Security Policy amongst sub-properties
- Disadvantages
  - Without cloud provider support, you get…
    - Lower peak capacity if access to sites imbalanced
    - Coarse load balancing across sites, not adaptive to spikes
  - Management costs of multiple sites

1. Server partitioning (dynamic)

- Seamless, active, "forward deployment" of content to explicitly named servers near client
- Redirect requests from origin servers via dynamic URL rewriting of embedded content
- Application-level multicast based on geographic location of client
  - Example: Akamai, AWS Cloud Front, GCP Cloud CDN

1. Server partitioning (dynamic)



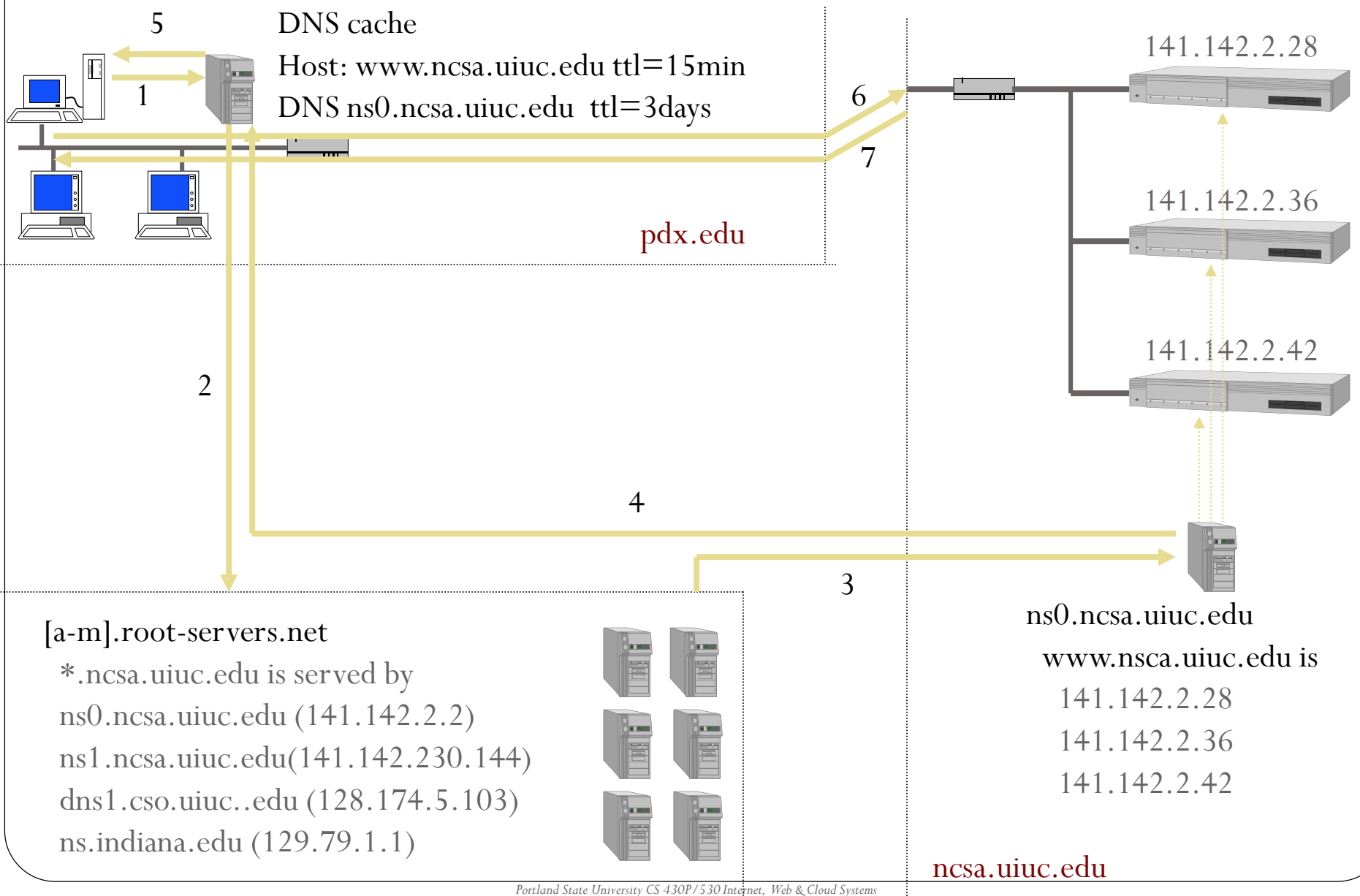
1. Server partitioning (dynamic)

- Advantages
  - Improved network utilization
  - Cost savings
    - Assuming $ network bandwidth >> $ storage
  - Better load distribution if replicas based on popularity
- Disadvantages
  - Distributed management costs
  - Complexity and vendor lock-in with integration to a CDN provider

2. DNS load balancing

- Popularized by NCSA circa 1993
- Fully replicated server farm
- IP address per node
- Adaptively resolve server name (round-robin, load-based, or geographic-based)
- The reason why multiple DNS addresses are returned on some responses

2. DNS load balancing



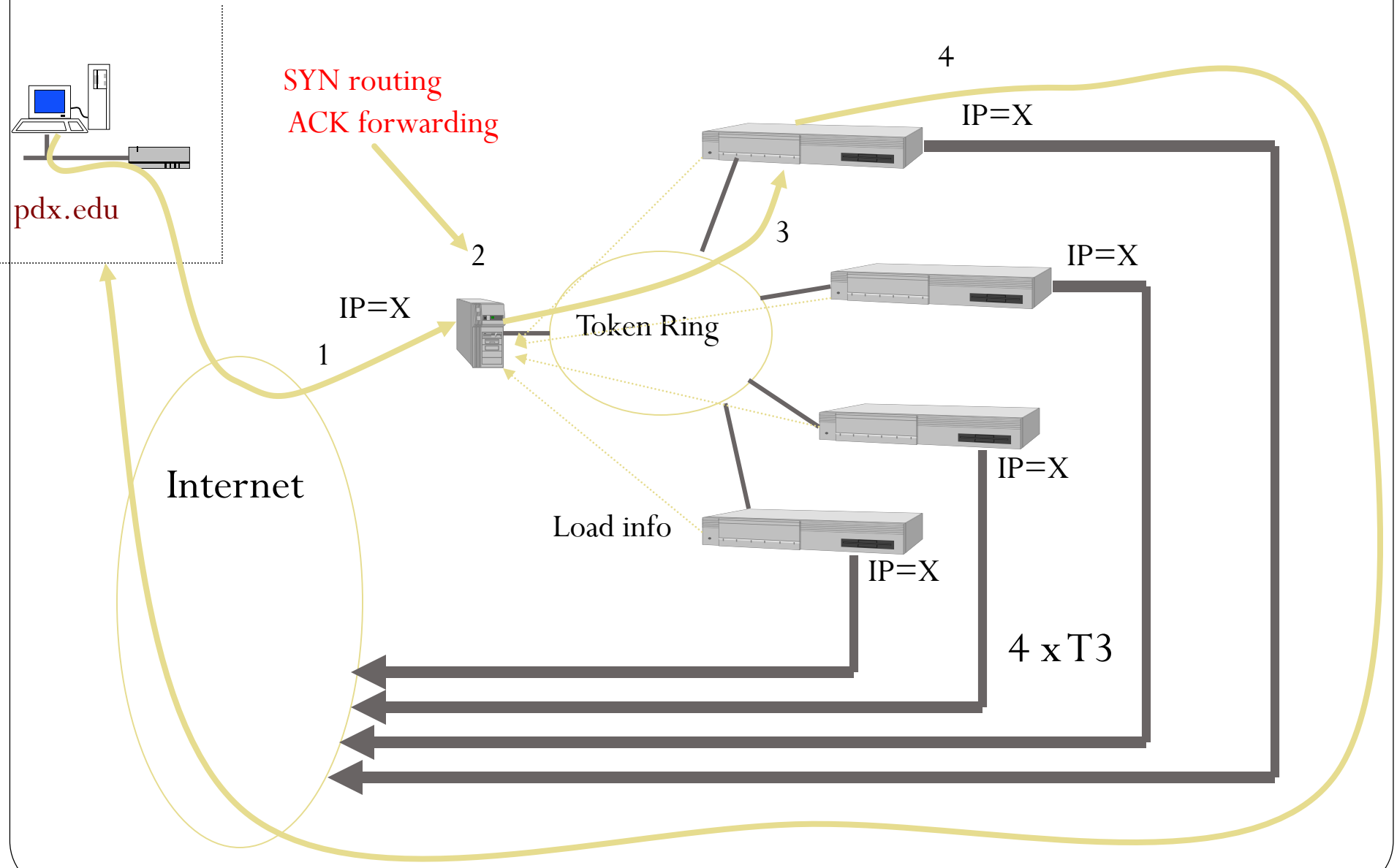
2. DNS load balancing

- Advantages
  - Simple to implement
  - Uses existing DNS infrastructure
- Disadvantages
  - Coarse load balancing over time
  - DNS caching at local name servers affects performance
  - Requires full server replication versus partitioning

3. Virtual servers

- Large server farm appearing as a single virtual server
  - Single front-end for connection routing

Olympic web server (1996)



Olympic web server (1996)

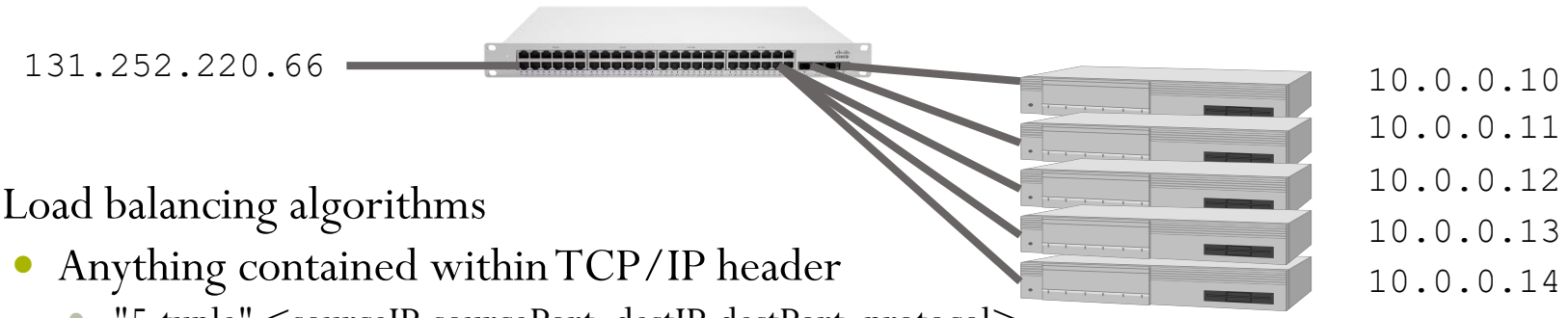
- Front-end implements a "reverse NAT"
- Front-end node
  - TCP SYN
    - Route to particular server based on policy
    - Store decision (connID, realServer)
  - TCP ACK
    - Rewrite packets and forward based on stored decision
  - TCP FIN or a pre-defined timeout
    - Remove entry
- Servers
  - IP address of outgoing interface = IP address of front-end's incoming interface
  - Treats front-end, token-ring, and cluster as one virtual server

Olympic web server (1996)

- Advantages
  - Minimal packet rewriting (e.g. Only ACK packets rewritten)
  - More reactive to load than DNS
- Disadvantages
  - Potential non-stickiness between requests
    - SSL sessions for a single client
    - Cache performance versus partitioned servers

Virtual server variations (L2-L4)

- Evolved into hardware switch implementations for performance

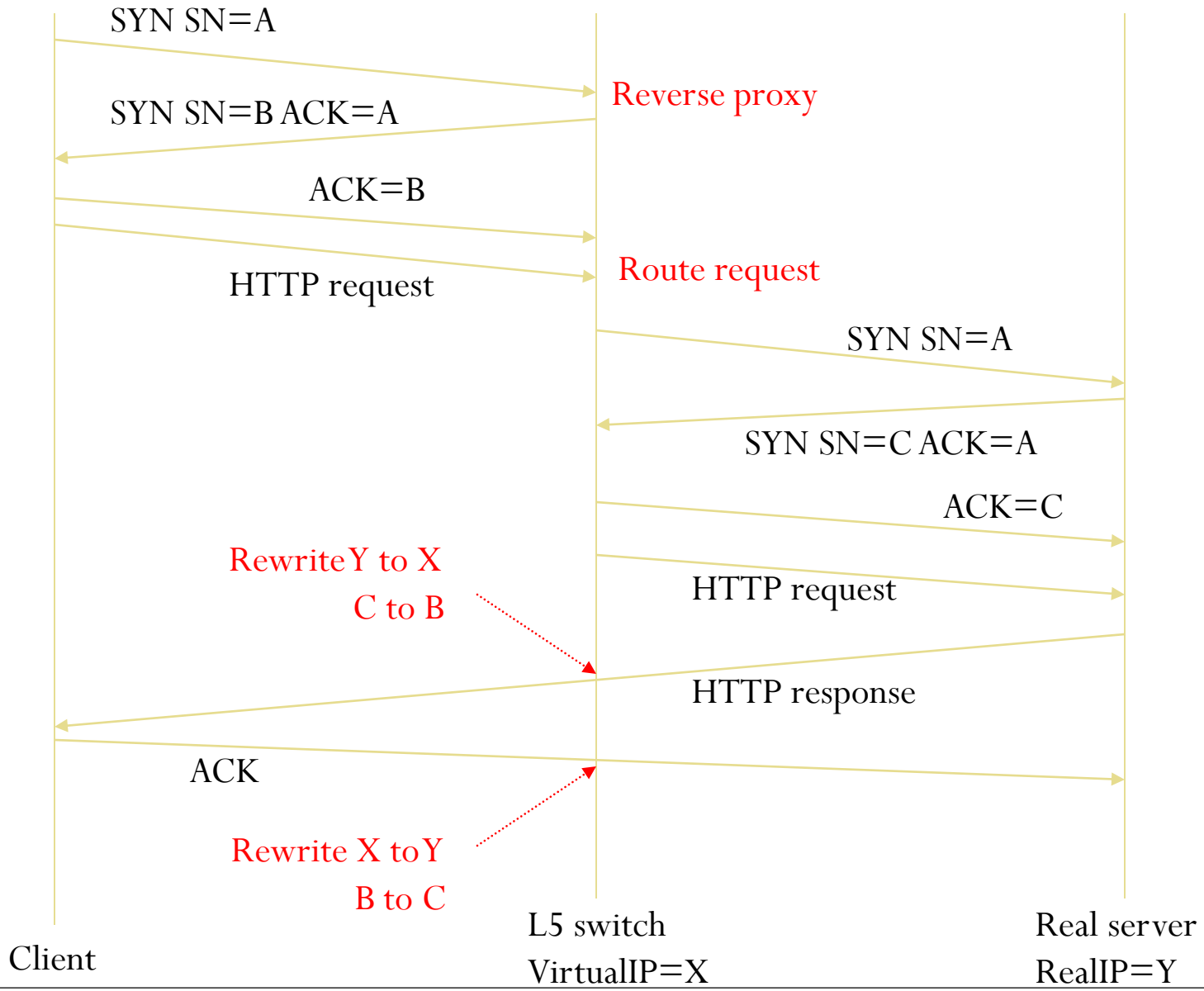


- Load balancing algorithms
  - Anything contained within TCP/IP header
    - "5-tuple" <sourceIP, sourcePort, destIP, destPort, protocol>
    - hash(source, dest, protocol)
  - Server characteristics
    - Least number of connections
    - Fastest response time
    - Server idle time
  - Other
    - Weighted round-robin based on server capabilities
    - Random

Virtual servers with L5

- Can also load balance based on content (i.e. URL)
- Requires one to proxy server connection *until* URL sent, before routing to backend servers
- Front-end implements a "reverse proxy" (versus a reverse NAT)
  - Examples: `nginx`, Google's front-end (GFE), CloudFlare, many hardware switches
- Switch/proxy
  - Terminates TCP handshake
  - Rewrites sequence numbers going in both directions

L5 switches

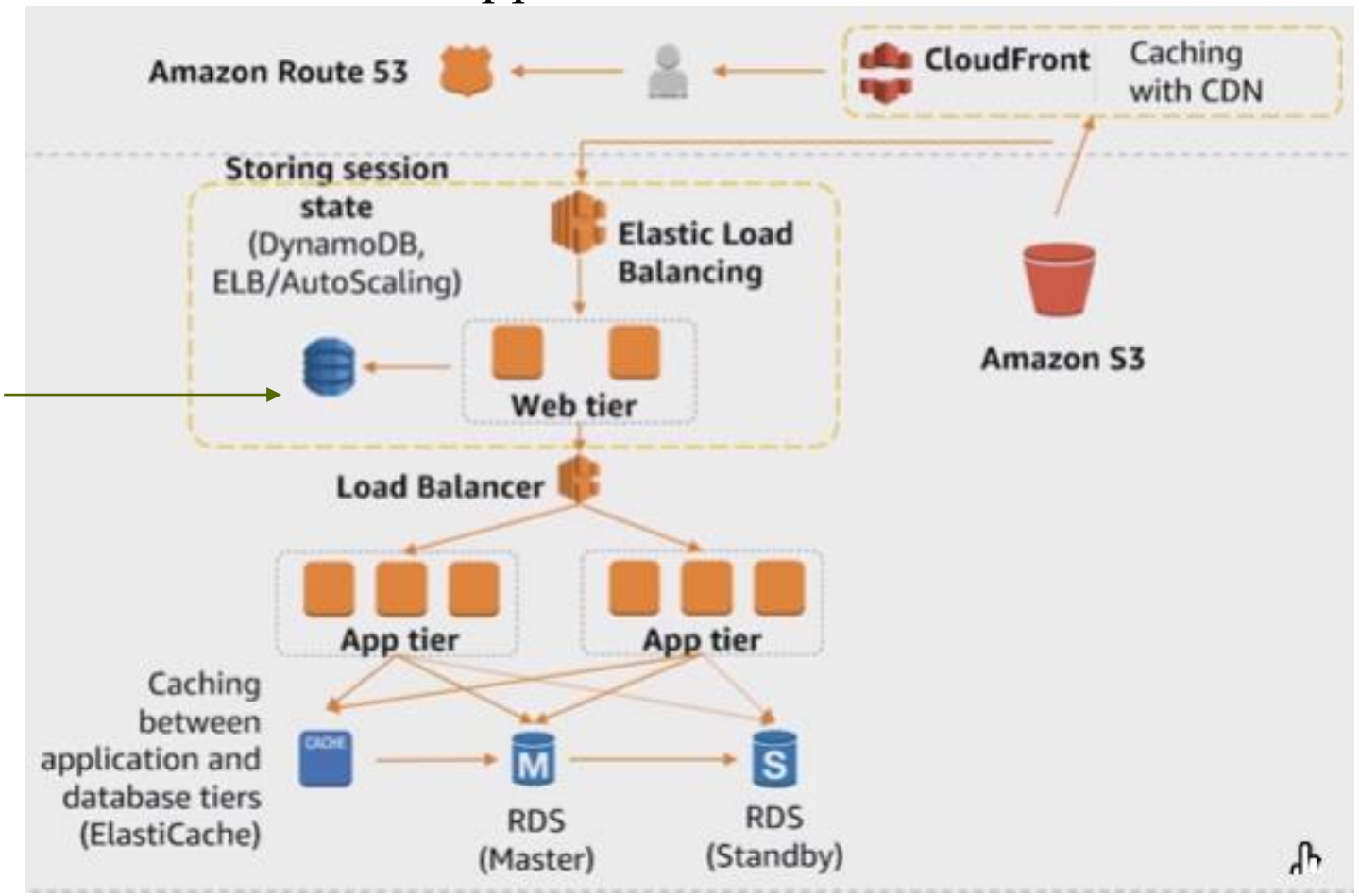

SYN SN=A
Reverse proxy
SYN SN=B ACK=A
ACK=B
HTTP request
Route request
SYN SN=A
SYN SN=C ACK=A
ACK=C
Rewrite Y to X
C to B
HTTP request
HTTP response
ACK
Rewrite X to Y
B to C
Client
L5 switch
VirtualIP=X
Real server
RealIP=Y

L5 switching

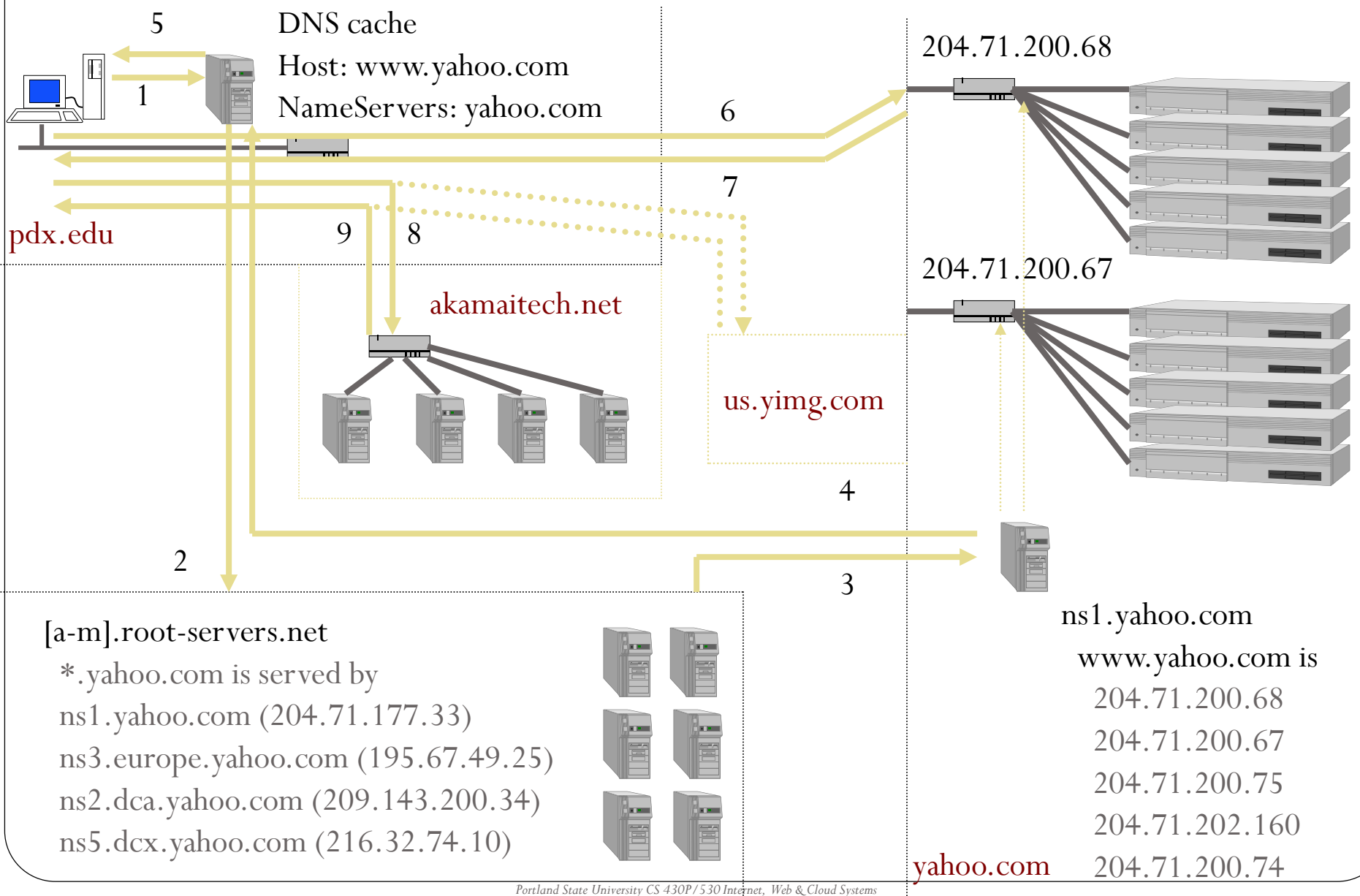
- Advantages
  - Increases effective cache/storage sizes (partition by URL)
  - Allows for session persistence (SSL,cookies)
  - Support for user-level service differentiation
    - Service levels based on cookies, user profile, User-Agent, URL
    - DDoS prevention based on request/user
- Disadvantages
  - Hot-spots
  - Overhead (custom ASICs needed to process at line-speed)

Alternatives to support session persistence

- Have all web frontends share one big memory cache in the cloud
  - Done via in-memory datastores (Redis, Memcached)
- Example: AWS ElastiCache applied to user session state on web tier



Putting it together: Yahoo!



Support in cloud platforms

- GCP Cloud DNS, AWS Route 53
  - Map DNS records to your instances
- GCP Cloud Load Balancer, AWS Elastic Load Balancer
  - Spread HTTP requests across machines
    - L4 connection load balancing
    - L5 content-based load balancing
    - Geographic and network latency based load balancing
- GCP Cloud CDN or AWS CloudFront
  - Forward deploy content via compute engine instances in load balancer to leverage edge caches in GCP
- See CDN lab

CDNs for DDoS protection

DDoS problem

Over nine million cameras and DVRs open to APTs, botnet herders, and voyeurs

Re-branded IP cameras and DVRs sold by over 100 companies can be easily hacked, researchers say.

By Catalin Cimpanu for Zero Day | October 9, 2018 -- 15:35 GMT (08:35 PDT) | Topic: Security

21 KrebsOnSecurity Hit With Record DDoS

SEP 16

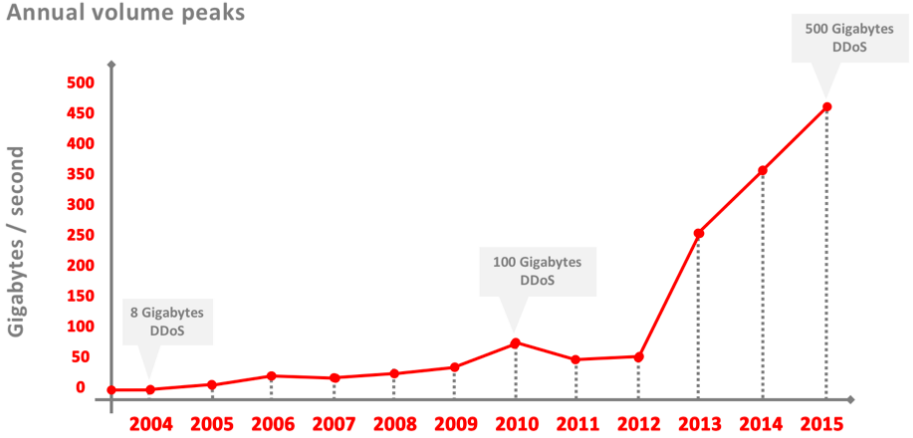
On Tuesday evening, KrebsOnSecurity.com was the target of an extremely large and unusual distributed denial-of-service (DDoS) attack designed to knock the site offline. The attack did not succeed thanks to the hard work of the engineers at **Akamai**, the company that protects

LILY HAY NEWMAN SECURITY 03.01.18 11:01 AM

GITHUB SURVIVED THE BIGGEST DDOS ATTACK EVER RECORDED

DDoS attack evolution

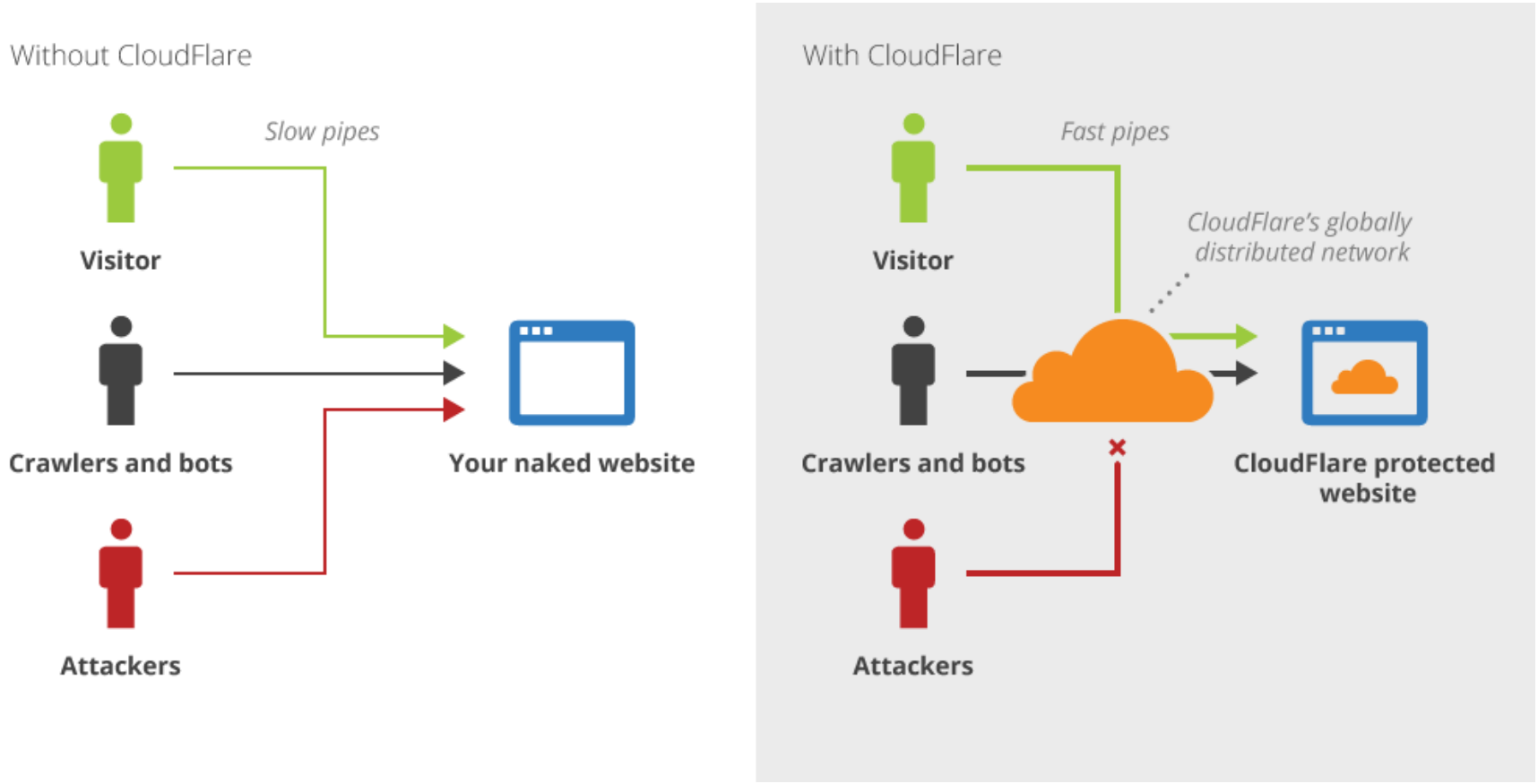
Annual volume peaks



CDNs to the rescue?

- Distributed denial-of-service mitigation
  - CDN manages your DNS to point to forward-deployed nodes
  - Performs a reverse proxy operation on nodes as previously
    - Terminates connections and examines request, before forwarding to content nodes
  - Drops sources of unwanted requests
    - Mirai traffic, GitHub attack traffic, Dyn DNS attack traffic (2016), etc.
  - Can also drop malicious requests after analysis by web-application firewall (WAF)
    - Common XSS payloads, known exploits
- Examples: CloudFlare, Akamai, Google, Microsoft
  - Google now protecting high-profile anti-hacking sites for free

General architecture



Azure DDoS Protection (4/18/18)

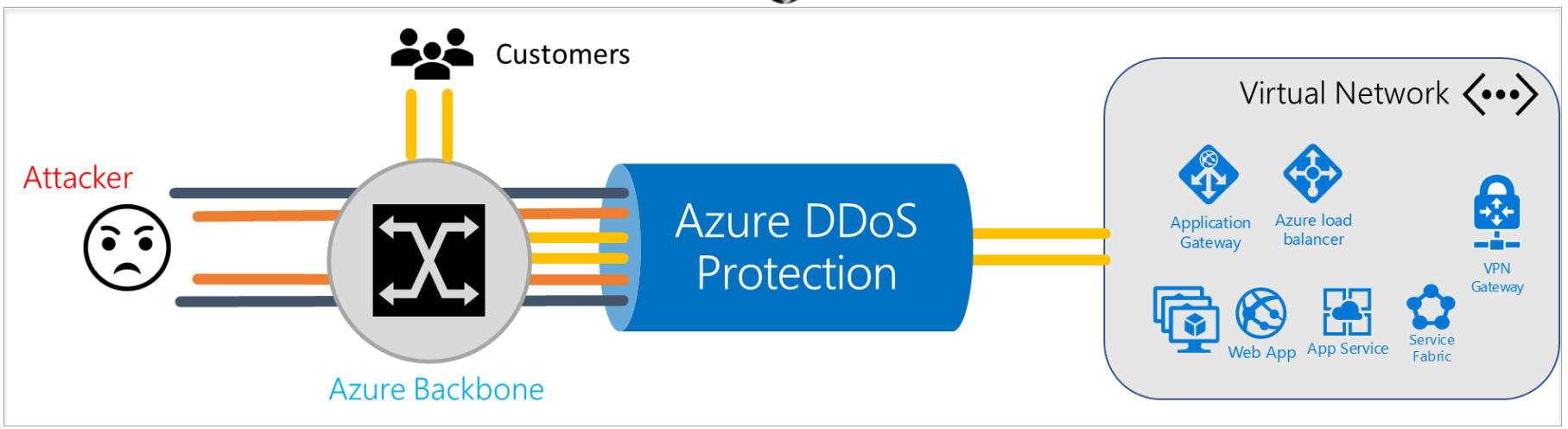
- Reverse-proxy at edge
- "L7" protection
  - WAF (SQLi, XSS filter)
  - Rate-limit per IP addr
  - Protocol attacks (floods)

Microsoft Azure

Azure DDoS Protection for virtual networks generally available

Posted on April 18, 2018

Anupam Vij, Senior Product Manager, Azure Networking



https://azure.microsoft.com/en-us/blog/azure-ddos-protection-for-virtual-networks-generally-available/

Issue: HTTPS proxying

- To proxy an https connection at edge, CDN must have
  - Certificate of site it's protecting
  - Private key associated with certificate to decrypt key from client
    - e.g. client encrypts random key with public key of site to establish symmetric encryption
    - Can only be decrypted by server's private key
- But, not all sites want to give up private key to CloudFlare (or other CDNs)
  - Breaks end-to-end security guarantees that TLS was intended to provide!
- Trade-off DDoS resilience for application security

Key server architecture

- Site must either give up private key to CDN or co-locate key server to the edge to implement part of TLS requiring private key