Content Distribution Networks

Outline

Implementation Techniques
Hashing Schemes
Redirection Strategies

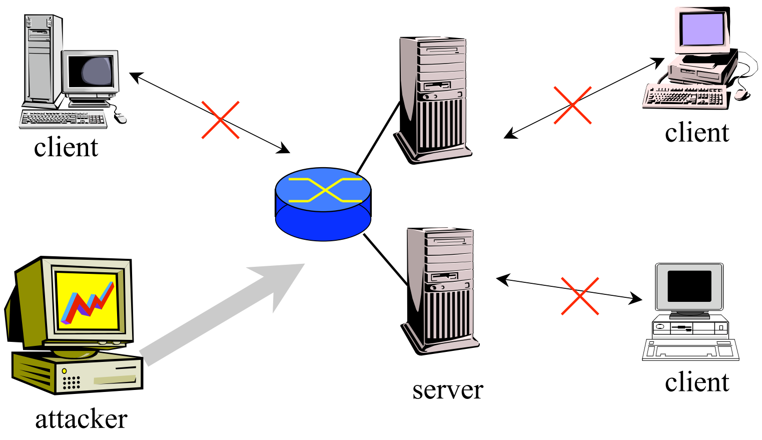
Design Space

- Caching
  - explicit
  - transparent (hijacking connections)
- Replication
  - server farms
  - geographically dispersed (CDN)

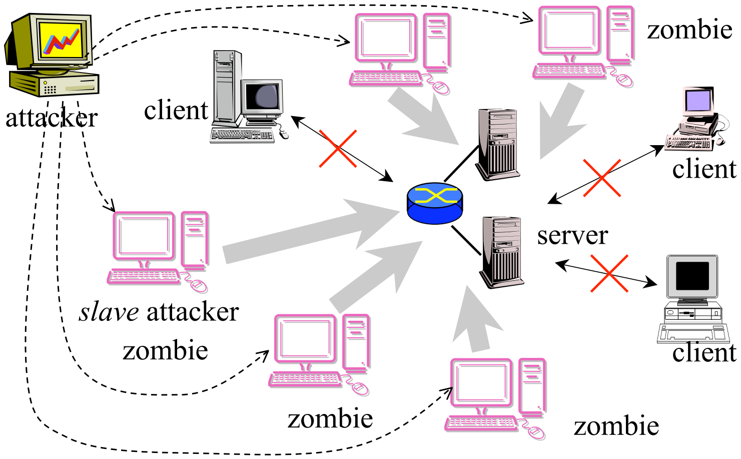
Story for CDNs

- Traditional: *Performance (Response Time)*
  - move content closer to the clients
  - avoid server bottlenecks
- New: *Flash Crowds & DDoS (System Throughput)*
  - distribute load over massive resources
  - multiplicatively raise level of resources needed to attack

Denial of Service Attacks (DoS)

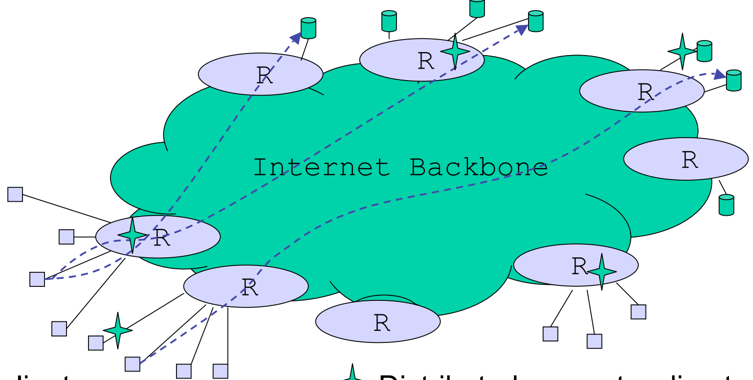


Distributed DoS (DDoS)



Redirection Overlay

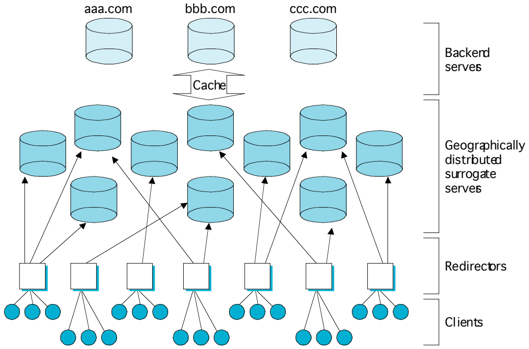
Geographically distributed server clusters



clients

Distributed request-redirectors

CDN Components



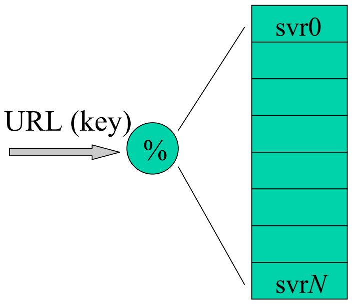
Techniques

- DNS
  - one name maps onto many addresses
  - works for both servers and reverse proxies
- HTTP
  - requires an extra round trip
- Router
  - one address, select a server (reverse proxy)
  - content-based routing (near client)
- URL Rewriting
  - embedded links

Redirection: Which Replica?

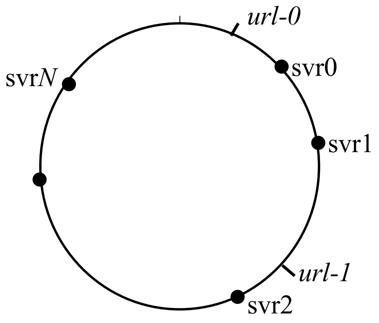
- Balance Load
- Cache Locality
- Network Delay

Hashing Schemes: Modulo



- Easy to compute
- Evenly distributed
- Good for fixed number of servers
- Many mapping changes after a single server change

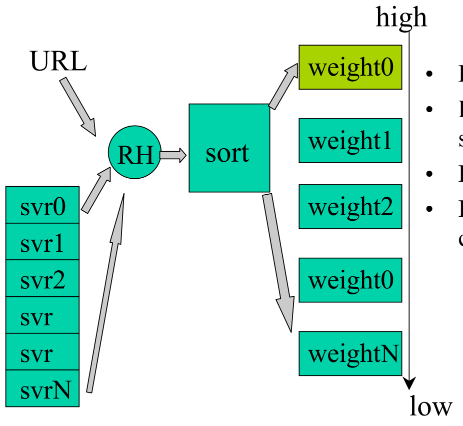
Consistent Hashing (CHash)



Unit circle

- Hash server, then URL
- Closest match
- Only local mapping changes after adding or removing servers
- Used by State-of-the-art CDNs

Highest Random Weight (HRW)



- Hash(url, svrAddr)
- Deterministic order of access set of servers
- Different order for different URLs
- Load evenly distributed after server changes

Redirection Strategies

- Random (Rand)
  - Requests randomly sent to cooperating servers
  - Baseline case, no pathological behavior
- Replicated Consistent Hashing (R-CHash)
  - Each URL hashed to a fixed # of server replicas
  - For each request, randomly select one replica
- Replicated Highest Random Weight (R-HRW)
  - Similar to R-CHash, but use HRW hashing
  - Less likely two URLs have same set of replicas

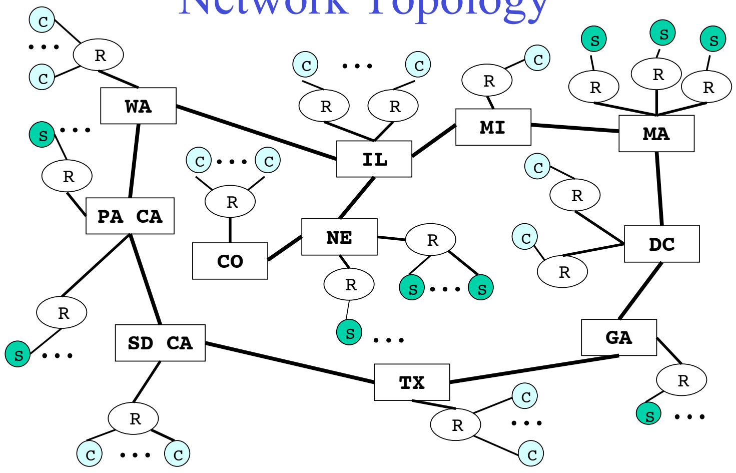
Redirection Strategies (cont)

- Coarse Dynamic Replication (CDR)
  - Using HRW hashing to generate ordered server list
  - Walk through server list to find a lightly loaded one
  - # of replicas for each URL dynamically adjusted
  - Coarse grained server load information
- Fine Dynamic Replication (FDR)
  - Bookkeeping min # of replicas of URL (popularity)
  - Let more popular URL use more replicas
  - Keep less popular URL from extra replication

Simulation

- Identifying bottlenecks
  - Server overload, network congestion…
- End-to-end network simulator prototype
  - Models network, application, and OS
  - Built on NS + LARD simulators
  - 100s of servers, 1000s of clients
  - >60,000 req/s using full-TCP transport
  - Measure capacity, latency, and scalability

Network Topology

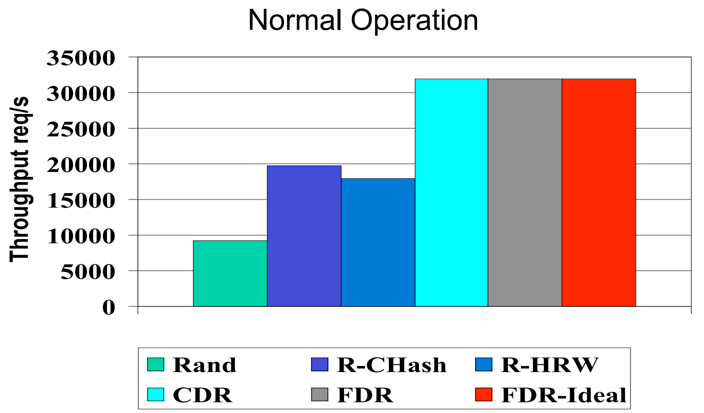


S – Server, C – Client, R - Router

Simulation Setup

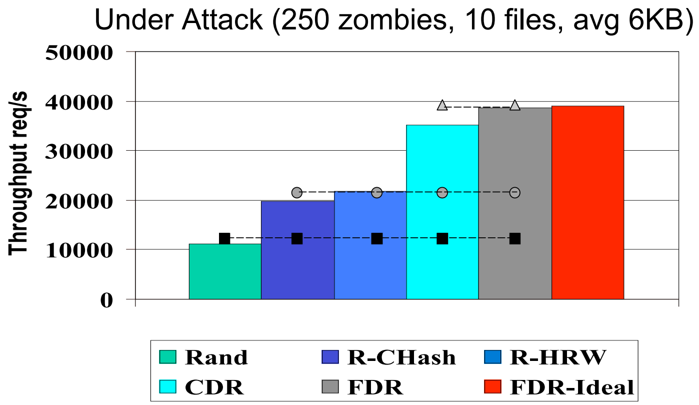
- Workload
  - Static documents from Web Server trace, available at each cooperative server
  - Attackers from random places, repeat requesting a subset of random files
- Simulation process
  - Gradually increase offered request load
  - End when servers very heavily overloaded

Capacity: 64 server case



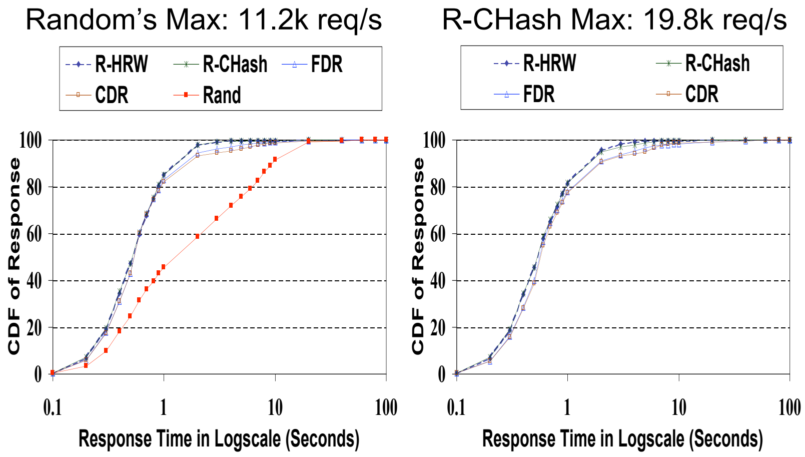
A single server can handle ~600 req/s in simulation

Capacity: 64 server case

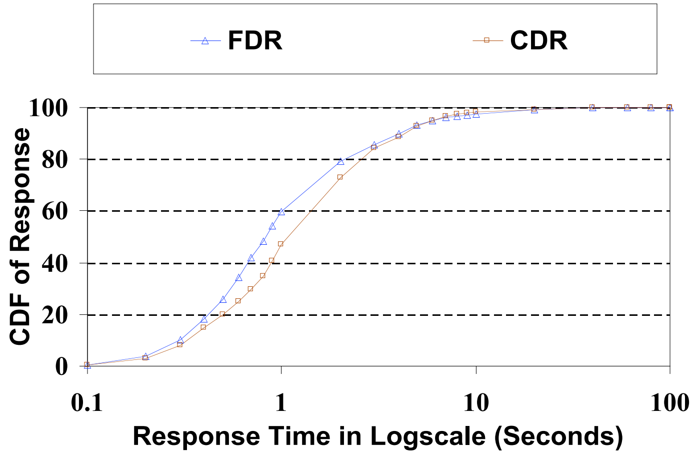


A single server can handle ~600 req/s in simulation

Latency: 64 Servers Under Attack



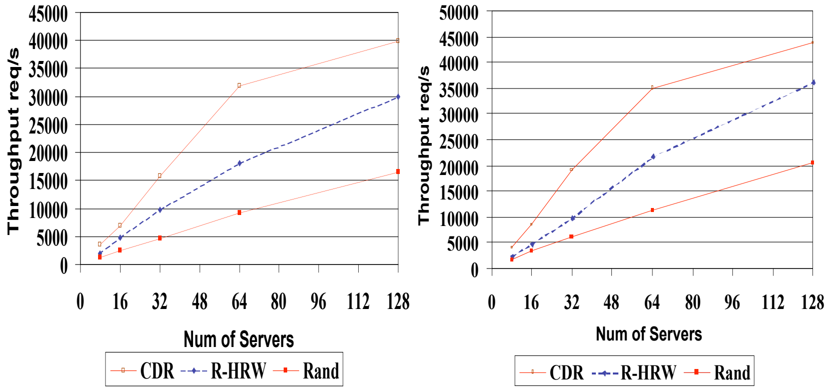
Latency At CDR's Max: 35.1k req/s



Capacity Scalability

Normal Operation

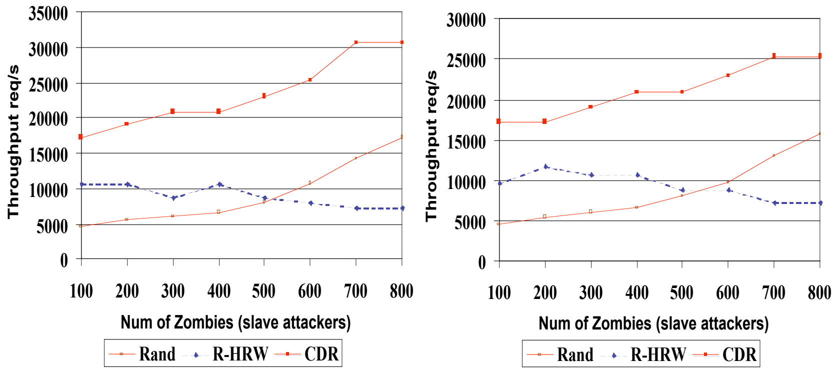
Under Attack (250 zombies, 10 files)



Various Attacks (32 servers)

1 victim file, 1 KB

10 victim files, avg 6KB



Deployment Issues

- Servers join DDoS protection overlay
  - Same story as Akamai
  - Get protection and performance
- Clients use DDoS protection service
  - Same story as proxy caching
  - Incrementally deployable
  - Get faster response and help others