

# On-path vs Off-path Traffic Steering, That Is The Question

---

Karima Saif Khandaker, Dirk Trossen, Jinze Yang, Zoran Despotovic, Georg Carle

# 1 **Problem of Traffic Steering**

---

Consider execution of services in distributed (e.g., virtualized) service environments:

- A **service**, realized through a **service instance**, available in one or possibly more network locations
- **Service transaction** requires affinity to a service instance after the initial service request due to possible ephemeral state created
- **Client** initiates a service transaction, utilizing one of the possibly many service instances available

**Problem:** At which layer of the networked system shall we make the decision as to WHICH instance available we shall utilize?

**Our Contribution:** Outline systems-at-test for Layer 3/3.5 vs Layer 7 decisions and compare performance against number of aspects for the same, given, decision process

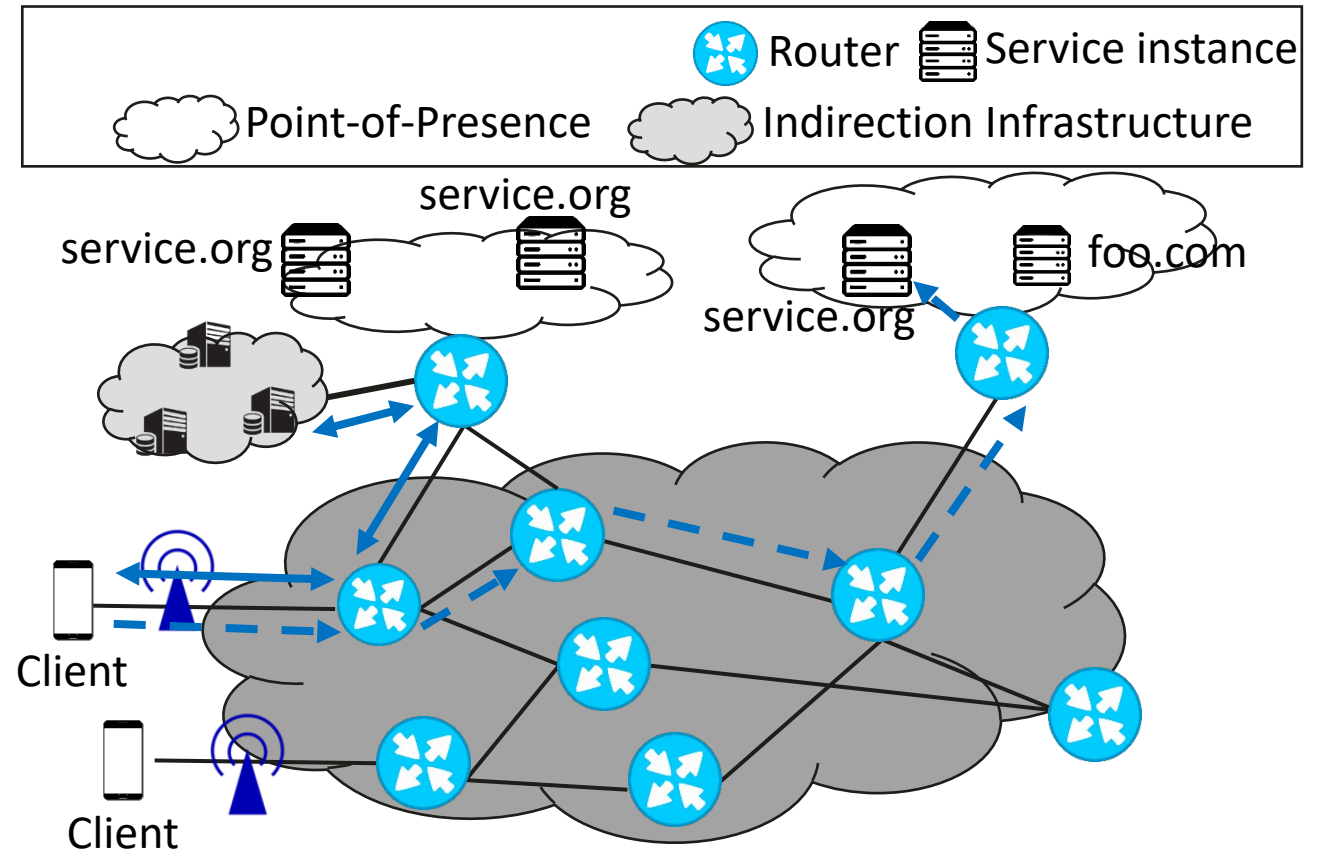
## System-At-Test for *Off-path* Traffic Steering

### Explicitly resolving service name into network locator

- Done through explicit indirection infrastructure
- Client issues resolution request first
- Client then utilizes resolved network locator for the actual request

### Example Technologies

- DNS
- Global Server Load Balancing (GSLB) for CDN-based load balancing – see later
- ALTO (application layer traffic optimizations)
- HTTP redirect
- QUIC redirect for load balancing



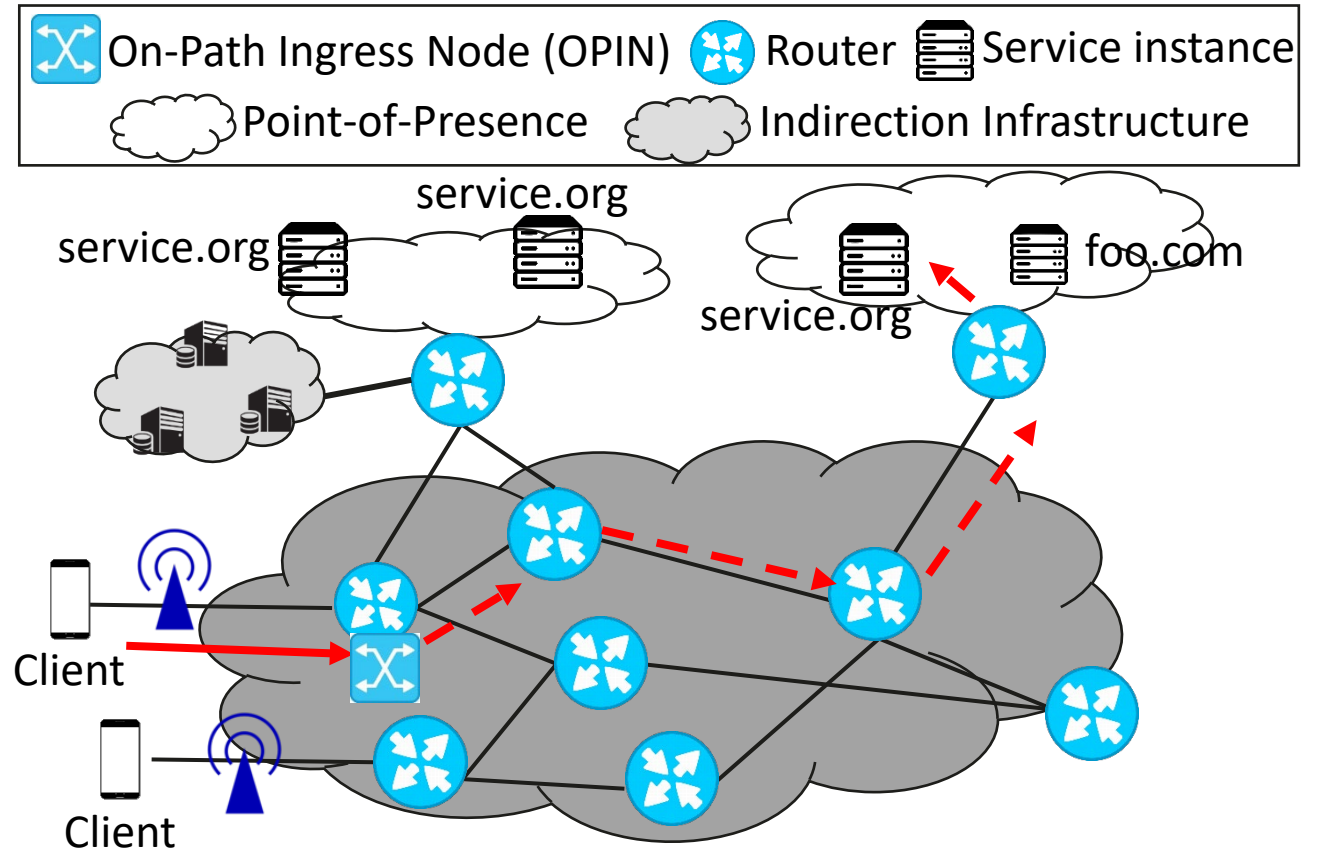
## 2 System-At-Test for *On-path* Traffic Steering

### On-path resolution of service name onto network locator

- Utilize on-path ingress node (OPIN) to take role of indirection infrastructure
- Client issues service request to OPIN, which includes service payload!
- OPIN maps service name onto network locator, forwarding packet to chosen instance
- Realized either at L3 (direct path) or L3.5 as shim overlay (overlay path)

### Example Technologies

- Dyncast (work in IETF) to on-path mapping of anycast IP onto IP unicast
- Routing on service addresses (ROSA) as described in IFIP Networking 2022 [1]



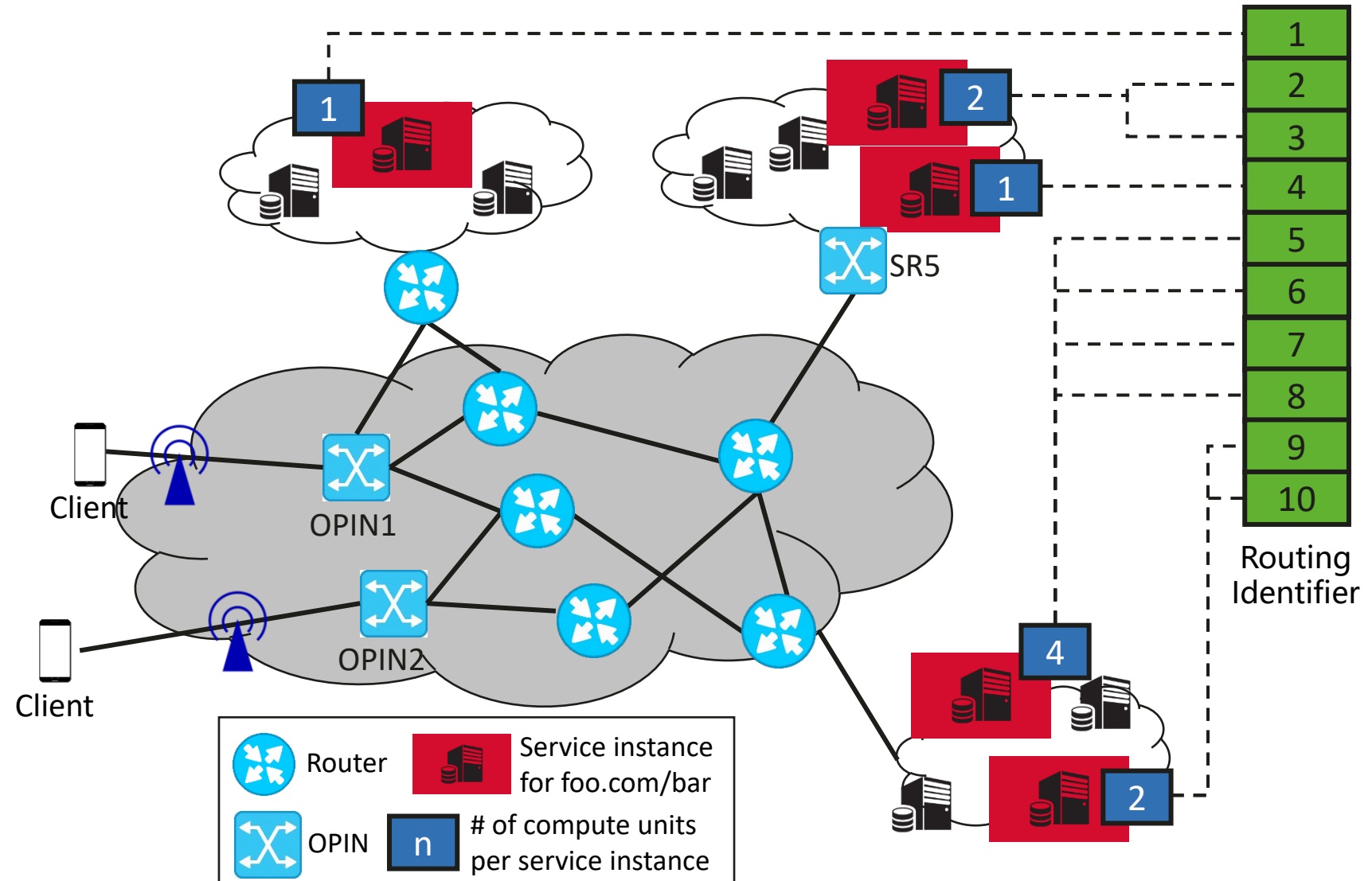
### 3 Traffic Steering Algorithm: Compute-Aware Distributed Scheduling (CArDS)

#### Key Criteria:

- Compute-aware
- Little/no signaling overhead
- Implementable at link speed

#### CArDS [1]:

- Each service instance assigned a normalized **compute (resource) unit**
- Compute units distributed to all OPINs as an identifier-specific **routing identifier**
- Distributed scheduling as **round robin** over the identifier for each incoming service request
- Implementable, e.g., in P4





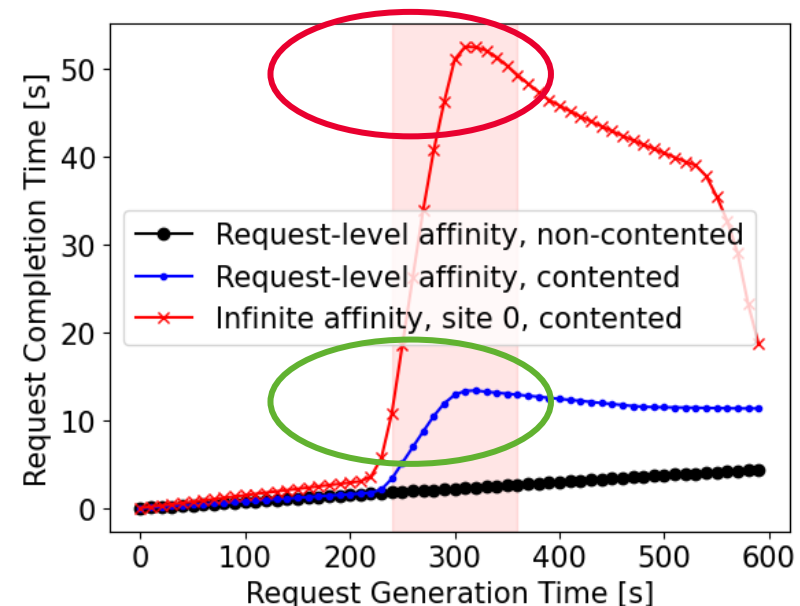
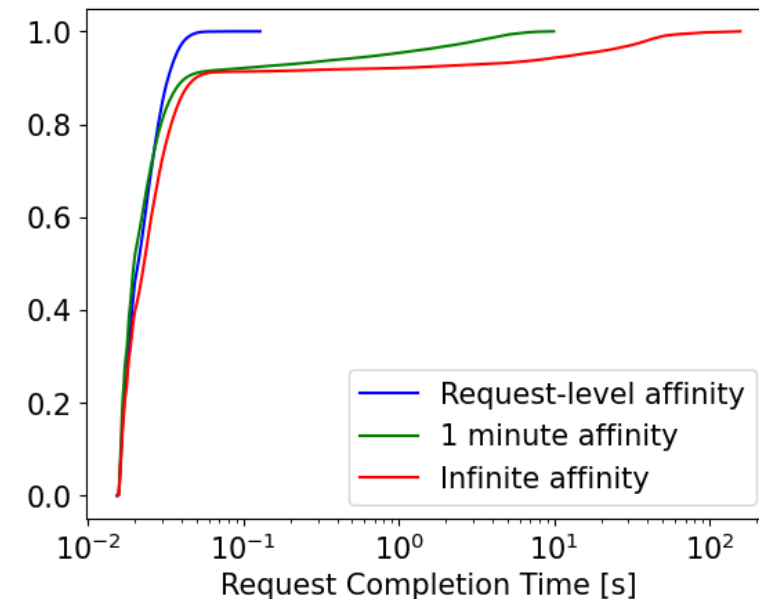
## Key Insights (2)

### Supporting *small affinities*

- Use case of AR/VR with 500ms max chunk retrieval (100ms average)
- Studied impact on request latency variance
  - > **average latencies** hardly affected
  - > **significant reduction of latency variance**, leading to smoother user experience & less buffering costs

### Impacting use case performance in *resilience situations*

- Studied impact of failure of one service instance on overall latencies
- Long affinities equally impact affect ALL clients in affected group
- Small affinities lowered this impact by about 4.5 fold
  - > **shared fate** for small affinity realizations
- **Important insight:** Problematic to apply small affinities in off-path systems to gain variance and resilience benefits
  - Would **lead to ~6.2k DNS requests per second** from this use case alone
    - > most DNS systems are rate limited and would not allow for such rates
  - Would **exceed latency budget** for use cases like AR/VR due to resolution latency



# Conclusion

---

- The answer to L3 vs L7 is largely driven by the needs for **dynamicity of relations** between client and the (compute) resources implementing the service instances
  - Hence, **new scenarios** like AR/VR, edge computing, distributed learning, ..., may be the driver here!
- **Latency variance** may be the biggest gain for having the ability to send to any of the possibly many service instances (without the latency penalties current L7 systems introduce)
- **Fate sharing** through virtualization can lead to better resilience across all clients
  - When combined with **fast server failure reporting** to OPINs, this may even improve on recovery times (not studied in our work)

Thank you.

