Séminaire STORE

Parsimonious Active Monitoring and Overlay Routing

Olivier Brun

LAAS–CNRS, Toulouse, France

February 23^{rd}, 2017

Outline

Introduction

- Classic measurement papers have shown that Internet routing results in paths that are sub-optimal with respect to a number of metrics
- Experiment with 20 nodes of the NLNOG Ring

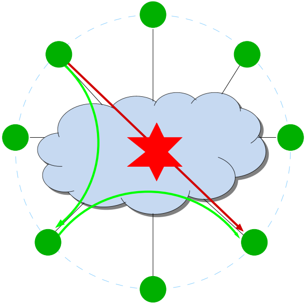
✔ The IP route is optimal only in 50% of cases

✔ Average gap to min latency is 31%

	IP	**OPT**
Moscow/Dublin	180	81
Singapore/Paris	322	153

Introduction

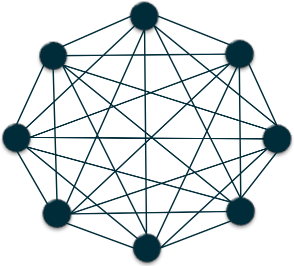
- **Routing overlays** were proposed as a method for improving performance, without the need to re-engineer the underlying network.
- Overlay nodes monitor the quality of the IP routes between themselves and cooperate to route messages.



Self-healing and self-optimizing application-layer virtual network.

Introduction

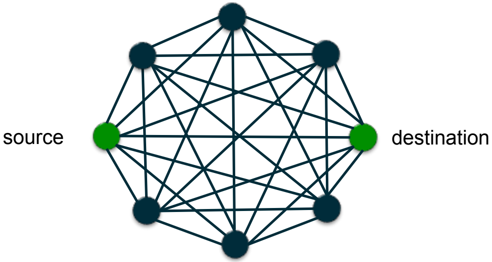
- **All-pairs probing**
  - ✔ In an overlay of n nodes, there are $O(n^2)$ links to monitor.
  - ✔ Monitoring the quality of all overlay links is excessively costly, and impairs scalability.



- **Problem:** how to design parsimonious monitoring strategies enabling to achieve near-optimal routing?

Introduction

- We consider a single origin/destination pair.



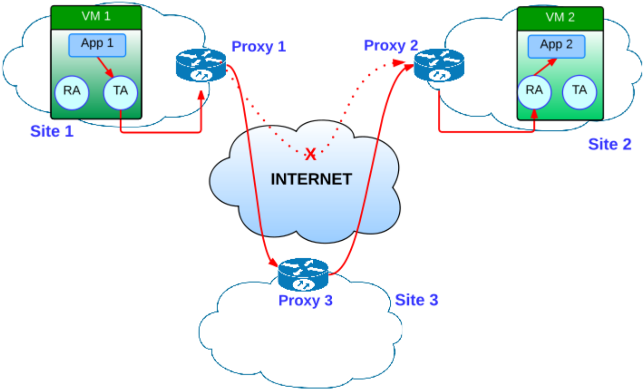
- How to discover an optimal route by probing only a small subset of possible paths?
  - ✔ Shortest path discovery problem
  - ✔ Learning-based routing in an adversarial environment
  - ✔ Learning-based routing as a POMDP

SMART

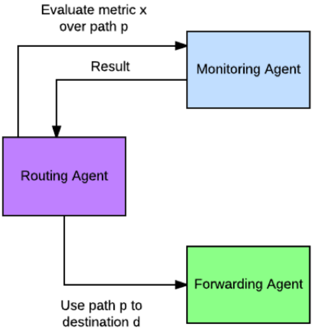
An open-source software for overlay routing

SMART – Self-MAnaging RouTing overlay

- Open source software for deploying self-healing and self-optimizing overlays over a sizable population of nodes

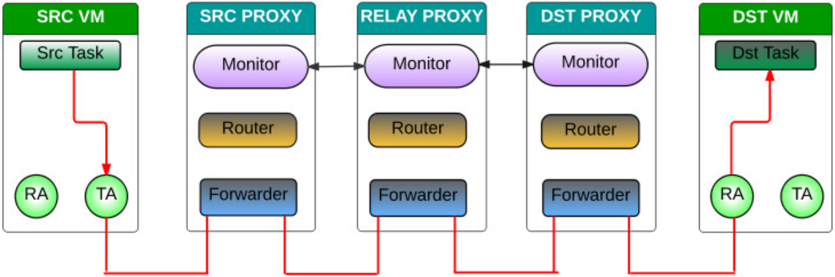


Proxy



- **Monitoring Agent:** it monitors the quality of the Internet paths between the local cloud and the other clouds (latency, bandwidth, loss rate).
- **Routing Agent:** It controls the monitoring agent so as to discover an optimal path with a minimum monitoring effort.
- **Forwarding Agent:** It forwards each incoming packet to its destination using source routing.

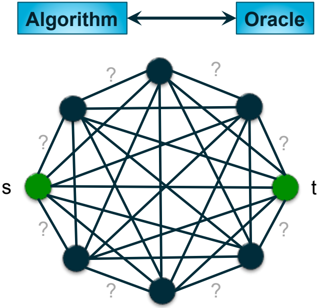
Packet Routing/Forwarding



Shortest Path Discovery Problem

Shortest Path Discovery Problem

- **Input:** a complete graph of n nodes whose edge lengths are unknown but can be discovered by querying an oracle.
- **Goal:** discover a shortest path from s to t by querying the minimum number of edges.
- **Online algorithm** with an approximation ratio of 2



- **Negative results:**
  - ✔ Any algorithm needs to query at least $n - 1$ edges
  - ✔ For any algorithm, there exists a bad instance for which the number of queries is $O(n^2)$

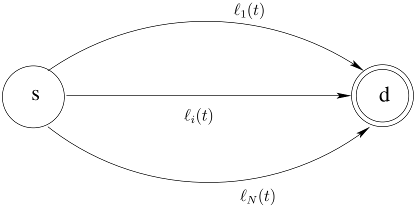
Learning-based Routing in an Adversarial Environment

Adversarial setting

- A decision algorithm $\mathcal{A}$ is given as input N paths from the origin to the destination, indexed from 1 to N.

- For example, the paths of at most k hops

$$N = \sum_{j=0}^{k-1} \frac{(n-2)!}{(n-2-j)!}$$



For each round $t = 1, 2, \ldots$

(1) a cost $\ell_i(t) \in [0,1]$ is assigned to each path i, but it is not revealed to the algorithm.

(2) then, the algorithm chooses a subset $Q(t) \subset \{1, 2, \ldots, N\}$ of K paths, observe their costs, and sends a message over a path $i^*(t) \in argmin_{i \in Q(t)} \ell_i(t)$.

Adversarial setting

- Cumulative cost of the algorithm over T rounds is defined as

$$L_T(\mathcal{A}) = \sum_{t=1}^{T} \min_{i \in Q(t)} \ell_i(t), \tag{1}$$

  whereas the cumulative cost of path i is $L_T(i) = \sum_{t=1}^{T} \ell_i(t)$.

- The Normalized regret of the algorithm $\mathcal{A}$ w.r.t. the best path is

$$R_T(\mathcal{A}) = \frac{1}{T}\left(L_T(\mathcal{A}) - \min_{i=1,\ldots,N} L_T(i) \right). \tag{2}$$

- **Goal**: design an algorithm $\mathcal{A}$ such that $R_T(\mathcal{A}) \to 0$ as $T \to \infty$.

Learning-based Routing in an Adversarial Environment

- Similar to a multi-armed bandits problem
  - ✔ Objective of the gambler is to maximize his reward (regret minimization)
  - ✔ The gambler does not know the expected reward of each arm but can learn by successively choosing different arms
- Well-studied problem in different contexts
  - ✔ Several applications: clinical trials, ad placement on webpages, etc.
  - ✔ Several variants: stochastic environment, adversarial environment, game, etc.

Learning-based Routing in an Adversarial Environment

- Randomized algorithm based on EXP3
  - ✔ Mixture of the uniform distribution and a distribution which assigns to each path a probability mass exponential in the estimated cumulative gain for that path:

$$p_i(t) = (1-\gamma)\,\frac{w_i(t)}{\sum_{j=1}^{N} w_j(t)} + \gamma\,\frac{1}{N}$$

  - ✔ Asymptotically the same average (per round) end-to-end performance as the best path.

$$R_T(EXP3) \leq \frac{11}{2}\sqrt{\frac{N\log(N/\delta)}{T}} + \frac{\log(N)}{2T}.$$

with probability at least $1-\delta$, for any $0<\delta<1$.

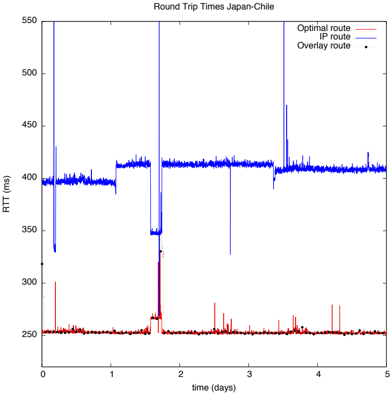
Latency Minimization: NLNog ring

- We restrict ourselves to paths with at most one intermediate overlay node
  - ✔ SMART probes only 5 overlay links per measurement epoch (among 342 links in total)
  - ✔ SMART uses the optimal 2-hop routes in 96% of the cases (gap to opt. latency is 0.39%)

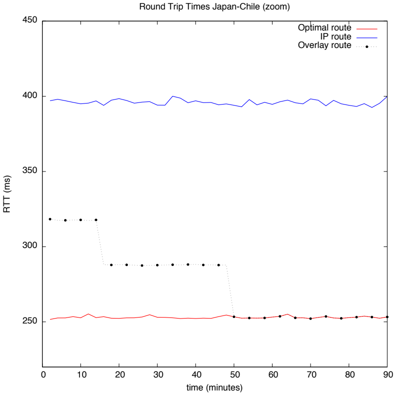
Average RTT (ms)

	IP route	SMART
Melbourne/Gibraltar	390.0	274.7
Narita/Santiago	406.7	254.5
Moscow/Dublin	179.9	81.9
Honk Kong/Calgary	267.1	131.8
Singapore/Paris	322.3	154.9
Tokyo/Haifa	322.6	180.8

Latency Minimization (2)



RTT (ms) Japan/Chile over 5 days



Zoom over the first 90 minutes

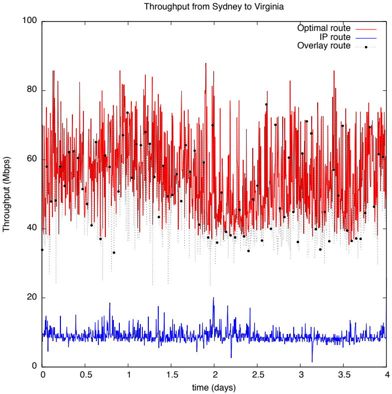
Throughput Maximization: Amazon EC2

- We restrict ourselves to paths with at most one intermediate overlay node
  - ✔ SMART probes only 5 overlay links per measurement epoch (among 72 links in total)
  - ✔ SMART uses the optimal 2-hop routes in 70% of the cases (gap to opt. latency is 6.6%)

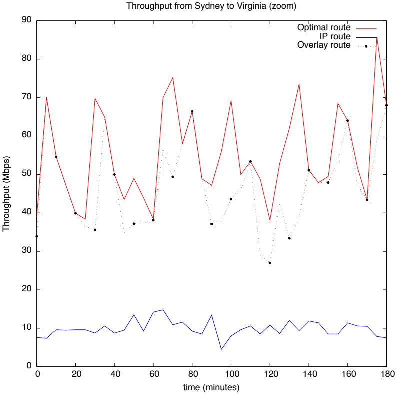
Average Throughput (Mbps)

	IP route	SMART
Dublin/Sydney	11.5	35.5
Singapore/Sao Paulo	12.8	39.5
Sydney/Virginia	8.5	50.7
Virginia/Singapore	7.4	31.2
Virginia/Sydney	6.9	32.2
Virginia/Tokyo	10.3	37.5

Throughput Maximization (2)



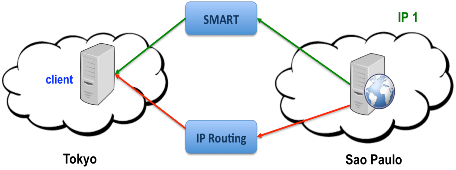
Throughput (Mbps) from Virginia to Sydney over 4 days

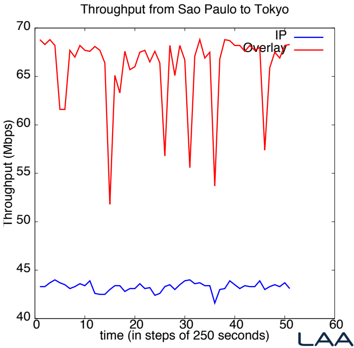


Throughput (Mbps) from Virginia to Sydney (zoom over the first 3 hours)

Live Experiment with SMART

- A client in Tokyo repeatedly downloads 100 MB files from a server in Sao Paulo
  - ✔ Every 4 mn, a file is downloaded via a Proxy in Oregon
  - ✔ A few seconds later, it is downloaded via the IP route



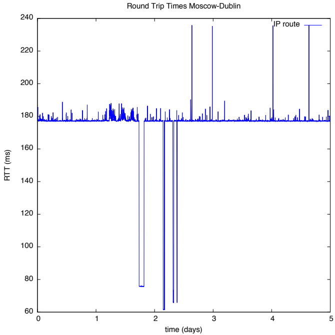


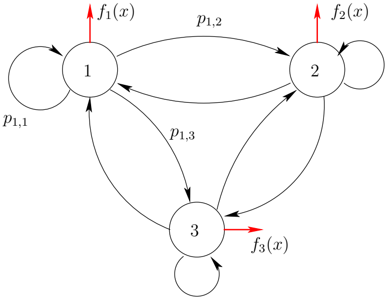
Throughput (Mbps)

Learning-based Routing as a POMDP

Stochastic setting

- Analysis of latency data collected over NLNog and RIPE Atlas shows that RTT time series can be modeled as Markov chains or Hidden Markov Models (HMM).





- Calibration phase for parameter estimation from measured RTT data

Stochastic setting

For each round $t = 1, 2, \dots$

(1) **State evolution:** the state $x_i(t)$ of each path i evolves according to to a DTMC over state space $\mathcal{S}_i$ with transition matrix $P_i = (p_i(u, v))$. The delay in state x is $\ell_i(x)$

(2) **Monitoring decision:** the player observes the states of a subset $Q(t)$ of paths, each at some cost c

(3) **Routing decision:** the player sends a message over path $r(t)$ and incurs cost $\ell_{r(t)}\left(x_{r(t)}(t)\right)$.

Stochastic setting

- **Goal:** minimise the expected sum of transmission delays and monitoring costs

$$\mathbb{E}\left\{\sum_{t=0}^{\infty} \beta^t \left(\ell_{r(t)}\left(x_{r(t)}(t)\right) + c\,|Q(t)|\right)\right\},$$

  where $\beta < 1$ is a given positive discount factor

- **Tradeoff** between the cost of monitoring paths, which brings more up-to-date state information, and the higher probability of experiencing high transmission delay

Partially Observable Markov Decision Process

- **Sufficient statistics:** all the information available at time t is summarized by the vector $\mathbf{s} = (s_1, s_2, \dots, s_N)$, where $s_i = (y_i, \tau_i)$
  - ✔ y_i is the last observed state of path i
  - ✔ τ_i is the age of this observation
- **Belief** on the state of path i: $\left[p_i^{(\tau_i)}(y_i, 1), p_i^{(\tau_i)}(y_i, 2), \dots\right]$
- **Transition probabilities:** given the set Q of monitored paths,

$$\pi_Q(\mathbf{s}, \mathbf{s}') = \prod_{i=1}^{N} \pi_Q^{(i)}(s_i, s_i')$$

where the information on link i evolves from

  - ✔ (y, τ) to $(y, \tau + 1)$ with prob. 1 if $i \notin Q$,
  - ✔ (y, τ) to $(w, 1)$ with prob. $p_i^{(\tau)}(y, w)$ otherwise.

Optimal Policy

- **Optimal routing decision:** send the message on the path with minimum expected delay given the available information

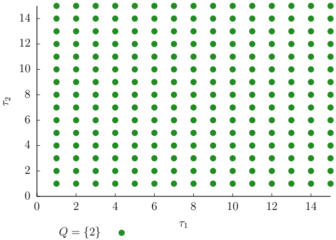
$$D(\mathbf{s}) = \min\left(\min_{i \notin Q(t)} \mathbb{E}\ell_i(x_i(t)), \min_{i \in Q(t)} \ell_i(y_i)\right)$$

- **Optimal monitoring decision** (Bellman equation)

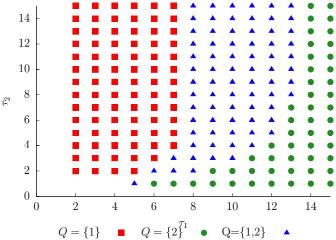
$$J^*(\mathbf{s}) = \min_A \left\{ c|Q| + \sum_{\mathbf{s}'} \pi_Q(\mathbf{s}, \mathbf{s}') \left(D(\mathbf{s}') + \beta J^*(\mathbf{s}')\right) \right\}$$

- **Value Iteration** Algorithm

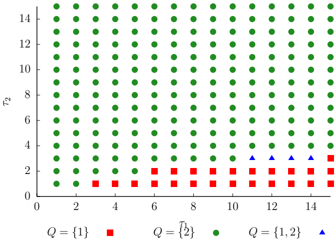
Example for two links



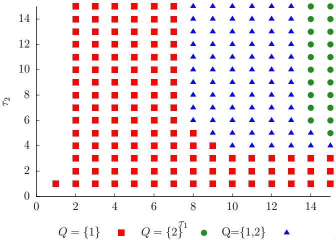
$y_1 = 0, \ y_2 = 0$



$y_1 = 1, \ y_2 = 0$



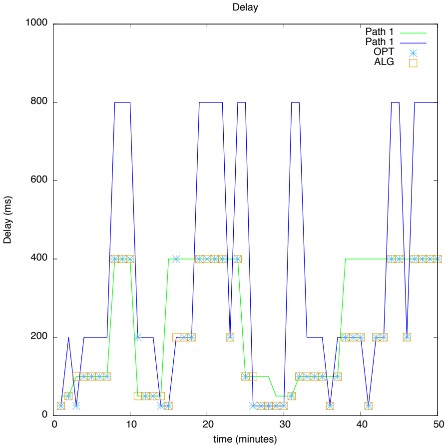
$y_1 = 0, \ y_2 = 1$



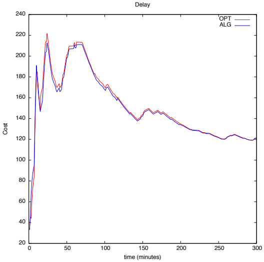
$y_1 = 1, \ y_2 = 1$

Threshold Policies

- Monitor a link iff its **success probability** (probability that it is the minimum delay link) is in between ϵ and Δ (e.g., $\epsilon = 0.3$, $\Delta = 0.8$).



Delay



Cost

Conclusion

Conclusion

- Internet routing works reasonably well most of the times, but routing overlay can yield spectacular improvements over native IP routing in some cases.
- A trade-off between the quality of the routes discovered and the monitoring effort to discover them is required.
- Probing does not cover all possible paths but only a few paths which have been observed to be of good quality, exploring also at random other paths whose quality might have improved recently.
- Future work will focus on the analysis of threshold policies in the stochastic setting.

Questions?

References

- C. Thraves-Caro, J. Doncel and O. Brun. **Optimal Path Discovery Problem with Homogeneous Knowledge**, LAAS report n°15181, 2015.
- O. Brun, H. Hassan and J. Vallet, Scalable, **Self-Healing and Self-Optimizing Routing Overlays**, IFIP Networking 2016, Vienna, Austria, 2016.
- O. Brun, L. Wang and E. Gelenbe, **Big Data for Autonomic Intercontinental Overlays**, IEEE Jour. Selected Areas in Communications, 2016.
- L. Wang, O. Brun and E. Gelenbe, **Adaptive Worload Distribution for Local and Remote Clouds**, 2016 IEEE International Conference on Systems, Man and Cybernetics, Budapest, Hungary, 2016.