Oblivious Routing: Worst-Case Routing is not Breaking the Internet's Legs

Thibaut Cuvelier

Oblivious Routing: Worst-Case Routing is not Breaking the Internet's Legs

Thibaut Cuvelier, Orange Labs (France)

Internet is highly dynamic

- Routing in Internet challenged by large throughput variations:
  - Day vs. night
  - Week days vs. week ends
  - Popular events (like sports or Black Friday)
- And still users expect excellent quality of service around the clock!
- Network operators must fine-tune their routing to guarantee sufficient bandwidth and low enough latency in all conditions

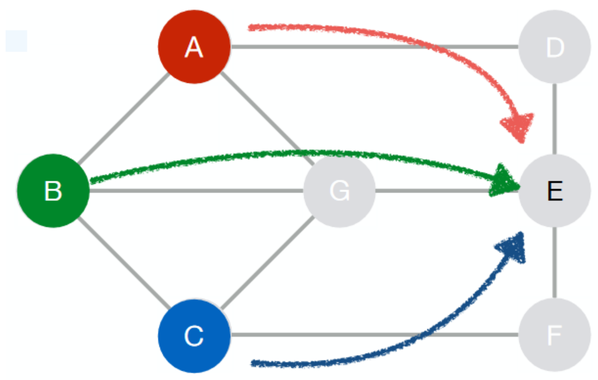
How to compute a routing?

- Very common approach: IGP protocols and shortest path
  - An administrator chooses the "distance" metric between two routers, according to their own criteria
  - Examples: OSPF, IS-IS, RIP
  - **Often not capacity-aware**: only dealing with connectivity, not traffic
- More centralised traffic engineering? A promise of SDN
  - Use optimisation tools (mostly linear programming — LP)
  - Exploit traffic-matrix measurements
- **What about uncertainty?** It lies in traffic and failures
  - Discarded with usual protocols! However, SDN proposes to change the situation

Multi-commodity flow (MCF) formulation

- Optimisation variables:
  - flow $f_k(e)$ in each link $e \in E$ for each origin-destination pair $k \in K$, expressed in MB/s
  - μ, a capacity reduction factor (reduce the capacity of the links by a factor μ)
- Constraints:
  - Link capacity C_e (scaled with μ): $\sum_{k \in K} f_k(e) \leq \mu C_e, \quad \forall e \in E$
  - Flow conservation: $\underbrace{\sum_{e \in \delta^+(v)} f_k(e)}_{\text{what flows in}} - \underbrace{\sum_{e \in \delta^-(v)} f_k(e)}_{\text{what flows out}} = \begin{cases} D_k & \text{if } v \text{ source of } k \\ -D_k & \text{if } v \text{ destination of } k, \\ 0 & \text{otherwise} \end{cases} \quad \forall v \in V, \forall k \in K$

  where D_k is the traffic for the origin-destination pair k (in MB/s)



- Objective: minimise μ, i.e. use the links as little as possible, be far below their capacity
  - Taking capacities as low as possible is equivalent to sending multiple times the demand, and to minimising the maximum congestion (defined as *load / capacity* for a link)

Oblivious routing

- Integrate **traffic uncertainty** into the MCF
- Optimise for all possible traffic matrices (that respect the capacity constraints)
  - The routing must minimise the worst congestion for all these matrices
- Algorithm? Iteratively add new constraints that limit the capacity of the edges
  - Use the same MCF to compute the demand that leads to the largest flow for a given edge e while respecting the capacity constraints, repeat for all edges
  - Pick the worst congestion among all the edges
  - Generate a new capacity constraint with C_e divided by the current value of μ
  - Start again until convergence
- Principles very similar to robust optimisation
- Important theoretical result [Räcke, 2008]: for any traffic matrix, the maximum congestion of oblivious routing is O(polylog # nodes) w.r.t. optimal routing

In practice

- Can be used at realistic scale (1,000 nodes, 1,000s edges, 10,000s demands)
  - Only a few minutes of computations!
  - Not useable for real- time applications, though
- Can generate both single- and multi-path routings
  - Single-path requires more computational power (as the flow cannot be split at a node)

Any industry interest?

- Facebook: semi-oblivious routing
  - Generate multiple paths (oblivious routing)
  - Balance the load between them

Extensions

- Optimising for all possible traffic matrices that respect the capacity constraints is probably too demanding
  - ⇨ Exploit traffic history to derive an uncertainty set
  - The traffic matrices must be a convex combination of previously seen matrices
  - The traffic matrices must be within a "distance" to an average matrix
  - Other techniques to derive an uncertainty set?
- Per se, no capacity uncertainty
  - Links may see their capacity reduced ⇨ Similar to current approach!
  - Links or nodes may fail ⇨ Completely different kind of uncertainty…
- Oblivious solutions are usually close to optimal, can we do better?
  - Idea: cluster traffic matrices to ensure the routing is oblivious only for a subset of matrices, switch routing when outside this subset
  - Problem: very large computation times, as clustering is performed based on routing similarity
- Maybe monitoring the whole traffic matrix provides a lot of useless information? Maybe only some links must be monitored?
  - This could improve runtimes, and thus allow more dynamic routing schemes

Räcke, H. (2008). Optimal Hierarchical Decompositions for Congestion Minimization in Networks. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing (pp. 255–264). New York, NY, USA: ACM.