One-pager 1

Martin Vassor

1 Improving connectivity at the network layer

1.1 Ron goals

Ron claims three goals: (i) improving failure detection and recovery; (ii) tighter integration of routing and path selection with the application; (iii) expressive policy routing [1]. The points (ii) and (iii) break layer separation and aiming for those is probably a fallacy, and should be dealt with in upper layers. Hence, we only consider the point (i) to be put in the Network layer. Thus, the goal is to provide fast error detection and recovery at the network layer.

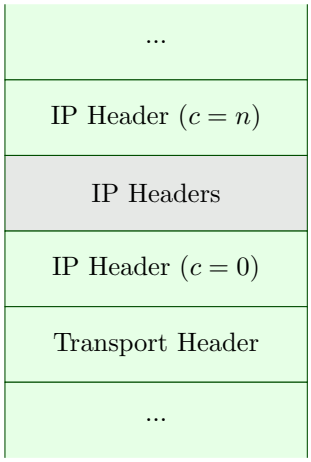

Figure 1: Multi IP header stack

1.2 Ron and network layer

To understand which ideas can be shared or not, let first synthesize the main differences between RON and the network layer. First, RON is a scattered network, it contains only a few nodes, while the Internet contains much more nodes. This allow RON routers to consider a more general view of the network topology. Second, RON routing algorithm has more control on the whole path, when network layer routing algorithm only control the next node.

With the multi-hop strategy, the challenge is to know of some distant router without knowing global properties of the network.

1.3 Multi-hop at Network layer

Multiple IP header encapsulation The idea to allow multi-hop at Network level is to encapsulate multiple time the Transport layer segment. This requires to change the IP header, to include a counter which indicates the number of IP headers (see Figure 1). When the packet reach its first header destination, the header is popped and the rest of the packet is either forwarded (if the counter is not 0) or passed to the Transport layer (otherwise). This allow the first router to have some control over intermediate *way points*[1].

***Way points* discovery** The next challenge is to discover distant routers. The source router can extract information from the headers of packets it has previously forwarded. With those informations it can build a table of existing potential *way points*[2]. When the source router gets a segment from the Transport layer, it chooses n *way points* in its table.

***Way points* routing** The last point to manage is the way to choose each router to select as *way point*. Two parameters can be easily scanned: inserting a *way point time stamp* in the (i-th) IP header can give an indication of the $i \to i+1$ latency to the $i+1$ router. Also, the bandwidth from a router can be estimated from the number of messages one router receive from it. Scanning the passing-by traffic should be quite efficient: if a router crashes, no packet should contains its address after the maximum time-to-live, and should quickly be erased from the tables.

Some details have been ignored (probabilistic

[1] We call *way point* a known distant router

[2] Not all have to be in the table (it is impossible, since each router is a potential *way point*).

choice, damping, etc.). Those details must be essential to guarantee some properties (connected component, connectivity, etc.). (word count: 500, including footnotes, titles, caption, excluding references)

2 Multi-path in *Service Access Layer*

2.1 Layer choice

SERVAL proposes a new architecture which separate *location* and *identity* to allow mobility of processes. NORDSTRÖM *et al.* claim in [2] that it is possible to implement a routing protocol which uses simultaneously multiples paths to improve the quality of the transfer[3]. The new layer they introduce (the Service Access Layer) serves the purpose of translating from an *identity* to a *location* (see Figure 2), and to provide this location to the underlying network layer, which is left unchanged.

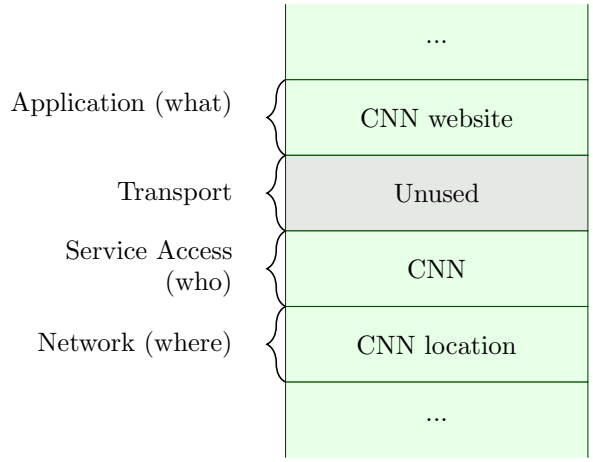

Figure 2: The new stack proposed by SERVAL

As explained in [3], "network-oriented protocols such as routing [...] will be grouped in this layer.". This statement shows that the multi-path routing should not be implemented in the Service Access Layer, as another layer already manage this point. Ignoring historical reasons, a fresh implementation should provide this feature in the Network Layer.

The choice made by NORDSTRÖM *et al.* is probably due to the practical impossibility to change the Network Layer nowadays.

2.2 Implementation challenges

Possible improvements and goals One can claim that using multi-paths to allow parallel message transmission can improve the bandwidth, by potentially adding parallel path to the bottleneck point, and then increasing the bandwidth (theoretically) up to the max-flow of the topology graph. However, the minimum latency being the minimum latency over all possible path, it should not be improved by using parallel path.

(Multi-)routing algorithm and path overloading To reach the max-flow of the graph, the routing algorithm must be completely redesign. The current *vector distance* algorithm works only for single paths. The main advantage of this algorithm is that it can be distributed over the routers.

Designing a routing algorithm for multi-path is much more complicated, as if an edge is shared by two paths, its capacity should not be counted twice, hence one can not simply adapt the current algorithm to return the n better paths. Moreover, the multi-routing algorithm should have a global knowledge of the topology graph and the currently studied paths. Thus, providing a distributed implementation[4] is more expensive, as all those informations have to be carried along each router. For instance, in Figure 3, the best path (of quality 9) has to be excluded from the best 2-path (of quality 10), but a local computation (say in R3) would return a best path of 9.

Other challenges Finding a scalable multi-routing seems to be the main issue of multi-routing. An implementation would leads to other problems

[3] latency, bandwidth, reliability, etc. The exact definition of quality is not relevant here.

[4] aiming to a situation where every router have a local routing algorithm as with *distance vector* nowadays.

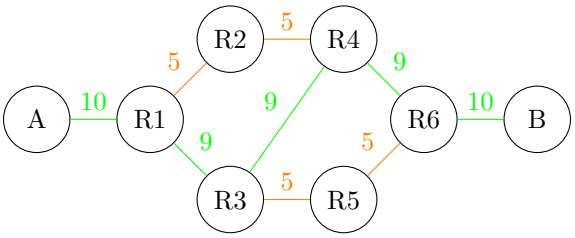

Figure 3: Example of graph with the best path is not in the best 2-path

such as more reordering as the latency can vary from one path to the other. Nevertheless, TCP is designed to handle these problem, hence adjusting parameters (window size computation, etc.) should work to solve these details. (word count: 500, including footnotes, titles, caption, excluding references)

References

[1] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient Overlay Networks. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, SOSP '01, pages 131–145, New York, NY, USA, 2001. ACM.

[2] Erik Nordström, David Shue, Prem Gopalan, Robert Kiefer, Matvey Arye, Steven Y Ko, Jennifer Rexford, and Michael J Freedman. Serval: An end-host stack for service-centric networking. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 7–7. USENIX Association, 2012.

[3] H. Zimmermann. OSI Reference Model — The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, COM-28:425 – 432, April 1980.