

# NFQL: The Swiss-Army Knife of Efficient Flow-Record Processing


Vaibhav Bajpai, Johannes Schauer, Jürgen Schönwälder  
School of Electrical and Computer Science  
Campus Ring 1, Jacobs University Bremen  
{v.bajpai, j.schauer, j.schoenwaelder}@jacobs-university.de

**Abstract**—Cisco’s NetFlow protocol and IETF’s IPFIX open standard have contributed heavily in pushing IP flow export as the de-facto technique for collecting aggregate network traffic statistics. These flow records have the potential to be used for billing and mediation, bandwidth provisioning, detecting malicious attacks and network performance evaluation. However, understanding certain traffic patterns requires sophisticated flow analysis tools that can mine flow records for such a usage. We recently proposed a flow query language that can cap such flow-records. In this paper, we introduce Network Flow Query Language (NFQL). an efficient implementation of the query language. NFQL can process flow records, aggregate them into groups, apply absolute (or relative) filters, invoke Allen interval algebra rules, and merge group records. The implementation has been evaluated by suite of benchmarks against contemporary flow-processing tools.

## I. INTRODUCTION



## II. RELATED WORK

 `flow-tools` [2] is a suite of programs for capturing and processing NetFlow v5 flow records. It consists of 24 separate tools that work together by connecting them via UNIX pipes.

`nfdump` [3] works similar to `flow-tools` but uses a different storage format. Flow records are captured using `nfcapd` and then processed by `nfdump` which can filter as well as display the sorted and filtered result. The power of its filtering rules is similar to that of `flow-tools` and as such is mostly limited to absolute comparisons of flow attributes.

`tcpdump` and `wireshark` are the most popular tools used for packet capture and analysis. `tcpdump` [4] is a premier command-line utility that uses the `libpcap` [5] library for packet capture. The power of `tcpdump` comes from the richness of its expressions, the ability to combine them using logical connectives and extract specific portions of a packet using filters. `wireshark` [6] is a GUI application, aimed at both journeymen and packet analysis experts. It supports a large number of protocols, has a straightforward layout, excellent documentation, and can run on all major operating systems.

## III. FLOW QUERY LANGUAGE

The pipeline consists of a number of independent processing elements that are connected to one another using UNIX-based pipes. Each element receives the content from the previous pipe, performs an operation and pushes it to the next element in the pipeline. Fig. 1 shows an overview of the processing pipeline. A complete description on the semantics of each element in the pipeline can be found in [1]

The splitter takes the flow-records data as input in the `flow-tools` compatible format. It is responsible to duplicate the input data out to several branches without any processing whatsoever. This allows each of the branches to have an identical copy of the flow data to process it independently.

The filter performs *absolute* filtering on the input flow-records data. The flow-records that pass the filtering criterion are forwarded to the grouper, the rest of the flow-records are dropped. The filter compares separate fields of a flow-record against either a constant value or a value on a different field of the *same* flow-record. The filter cannot *relatively* compare two different incoming flow-records

The grouper performs aggregation of the input flow-records data. It consists of a number of rule modules that correspond to a specific subgroup. A flow-record in order to be a part of the group should be a part of at-least one subgroup. A flow-record can be a part of multiple subgroups within a group. A flow-record cannot be part of multiple groups. The grouping rules can be either absolute or relative. The newly formed groups which are passed on to the group filter can also contain meta-information about the flow-records contained within the group using the aggregate clause defined as part of the grouper query.

The group-filter performs *absolute* filtering on the input group-records data. The group-records that pass the filtering

criterion are forwarded to the merger, the rest of the group-records are dropped. The group-filter compares separate fields (or aggregated fields) of a flow-record against either a constant value or a value on a different field of the *same* flow-record. The group-filter cannot *relatively* compare two different incoming group-records

The merger performs relative filtering on the N-tuples of groups formed from the N stream of groups passed on from the group-filter as input. The merger rule module consists of a number of submodules, where the output of the merger is the set difference of the output of the first submodule with the union of the output of the rest of the submodules. The relative filtering on the groups are applied to express timing and concurrency constraints using Allen interval algebra [7]

The ungrouper unwraps the tuples of group-records into individual flow-records, ordered by their timestamps. The duplicate flow-records appearing from several group-records are eliminated and are sent as output only once.

## IV. IMPLEMENTATION

### A. Splitter

### B. Filter

### C. Grouper

#### 1) Group Aggregations:

### D. Group Filter

### E. Merger

### F. Ungrouper

## V. PERFORMANCE EVALUATION

performance evaluation goes here ...

## VI. CONCLUSION

The NFQL conclusion goes here ...

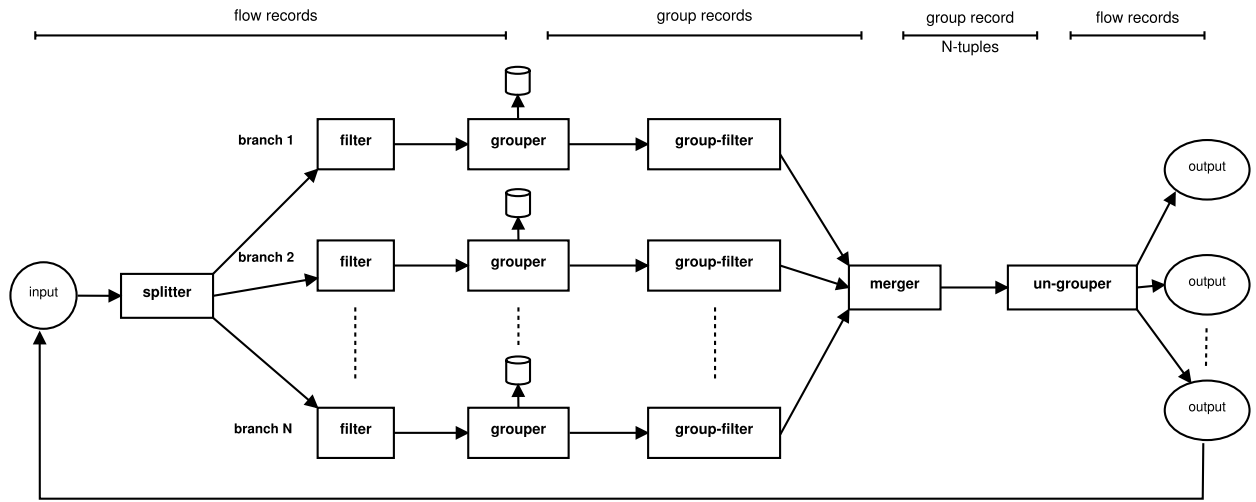


Fig. 1. NFQL Pipeline [1]

## REFERENCES

- [1] V. Marinov and J. Schönwälder, "Design of a Stream-Based IP Flow Record Query Language," in *Proceedings of the 20th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management: Integrated Management of Systems, Services, Processes and People in IT*, ser. DSOM '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 15–28. [Online]. Available: [http://dx.doi.org/10.1007/978-3-642-04989-7\\_2](http://dx.doi.org/10.1007/978-3-642-04989-7_2)
- [2] S. Romig, "The OSU Flow-tools Package and CISCO NetFlow Logs," in *Proceedings of the 14th USENIX conference on System administration*. Berkeley, CA, USA: USENIX Association, 2000, pp. 291–304. [Online]. Available: <http://dl.acm.org/citation.cfm?id=1045502.1045521>
- [3] P. Haag, "Netflow Tools NfSen and NFDUMP," in *Proceedings of the 18th Annual FIRST conference*, 2006.
- [4] V. Jacobson, C. Leres, and S. McCanne, *tcpdump - dump traffic on a network*, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA.
- [5] —, *pcap - Packet Capture library*, Lawrence Berkeley National Laboratory, University of California, Berkeley, CA.
- [6] G. Combs, *wireshark - Interactively dump and analyze network traffic*, University of Missouri, Kansas City.
- [7] J. F. Allen, "Maintaining knowledge about temporal intervals," *Communications of the ACM*, vol. 26, pp. 832–843, November 1983. [Online]. Available: <http://doi.acm.org/10.1145/182.358434>