Programming in Java
Jacobs University Bremen
Dr. Jürgen Schönwälder

Course: 320341
Date: 2015-09-29
Due: 2015-10-06

Problem Sheet #2

Problem 2.1: *processing JSON encoded network flow records* (2+2+3+3 = 10 points)

Network flow records record basic properties of a network flow. A network flow is usually keyed by a 5-tuple consisting of a protocol number (`proto`), a source IP address (`sip`), a source port number (`sp`), a destination IP address (`dip`), and a destination port number (`dp`). A flow usually has a start time and an end time plus the number of packets (`pkt`) and octets (`oct`) carried from the source endpoint to destination endpoint and vice versa (`rpkt` and `roct`)) if it is a bidirectional flow. If the flow is a TCP flow, the flags set in the initial TCP segment are recorded in `iflags` for the forward direction and in `riflags` for the reverse direction. The flags set in all subsequent TCP segments are recorded in the `uflags` for the forward direction and in `ruflags` for the reverse direction.

For the purpose of this assignment, flow records are represented in JSON as illustrated by the following example. The example shows a TCP flow (`proto` has the value 6) that was initiated by the endpoint at the interface with the IPv6 address `2001:0708:0040:3001:6670:02ff:fefc:137f` using the port number 39367. The flow has been established to the interface with the IPv6 address `2001:0638:0709:3000::0019` using port number 443. The TCP port 443 over TCP usually carries HTTP over TLS traffic. The flow did carry 2552 octets in 11 packets from the client to the HTTP server and it did carry 2576 octets in 9 packets from the HTTP server to the client. (Note that this includes all TCP handshake and teardown segments as well as TLS handshake and teardown messages.) The flow did start on 2015-09-22 at 23:33:42.077 and it did last for 0.554 seconds.

```
{
  "start": "2015-09-22 23:33:42.077",
  "end": "2015-09-22 23:33:42.631",
  "duration": 0.554,
  "rtt": 0,
  "proto": 6,
  "sip": "2001:0708:0040:3001:6670:02ff:fefc:137f",
  "sp": 39367,
  "dip": "2001:0638:0709:3000::0019",
  "dp": 443,
  "iflags": "S",
  "uflags": "APF",
  "riflags": "AS",
  "ruflags": "APF",
  "isn": 767467070,
  "risn": 783117522,
  "tag": 0,
  "rtag": 0,
  "pkt": 11,
  "oct": 2552,
  "rpkt": 9,
  "roct": 2576
}
```

a) Write a `Flow` class that can represent network flows as described above. Provide functions that can serialize flows into the JSON representation and that can deserialize flows from a JSON representation. Write a `FlowTest` class providing unit test cases for the `Flow` class. You can write the JSON serializer/deserializer yourself or you can use a JSON library such as Gson[1].

b) Write an `App` class that receives a list of file-names as command line arguments and sequentially processes each file. Each files is expected to contain a JSON list of flow records.

[1] https://github.com/google/gson

c) Define an interface `FlowFilter` that defines the signature of a boolean `test()` method for a flow. Use this interface in the `App` class to filter flows by flow properties such as the protocol number. Use anonymous inner classes implementing the `FlowFilter` interface. The goal here is to make the processing data driven; it should be possible to add additional filters by adding data structures instead of modifying program logic.

d) Write a `FlowBin` class that is used to aggregate flow statistics (packet and octet counters). Extend the `App` class such that flow records in each file are filtered into TCP, UDP, ICMPv2, and any other protocol flows and use different `FlowBin` objects to provide summary statistics (including the number of flows aggregated in a bin) for each flow category. The summary statistics should be printed as a JSON object for each input file.

A sample execution of the resulting program might produce output as follows:

```
yaf-20150924233343-03878.json:
[
  {
    "descr": "UDP flows",
    "pkts": 724780,
    "octs": 97717837,
    "rpkts": 1248328,
    "rocts": 1648401896,
    "flows": 25226
  },
  {
    "descr": "ICMPv6 flows",
    "pkts": 116266,
    "octs": 12199481,
    "rpkts": 0,
    "rocts": 0,
    "flows": 8463
  },
  {
    "descr": "OTHER flows",
    "pkts": 121,
    "octs": 13794,
    "rpkts": 0,
    "rocts": 0,
    "flows": 2
  },
  {
    "descr": "TCP flows",
    "pkts": 3792079,
    "octs": 962555418,
    "rpkts": 8106730,
    "rocts": 9873160864,
    "flows": 56296
  },
  {
    "descr": "HTTP/HTTPS (TCP port 80/443) flows",
    "pkts": 3749199,
    "octs": 947092208,
    "rpkts": 8064779,
    "rocts": 9847598133,
    "flows": 52741
  },
  {
    "descr": "QUIC/QUICS (UDP port 80/443) flows",
    "pkts": 659613,
    "octs": 90255733,
    "rpkts": 1244447,
    "rocts": 1647725990,
    "flows": 5071
  }
]
```