Flow Signatures of Popular Applications

Vladislav Perelman, Nikolay Melnikov, Jürgen Schönwälder

Jacobs University Bremen

IM-2011
Dublin, Ireland
24 May 2011

Overview

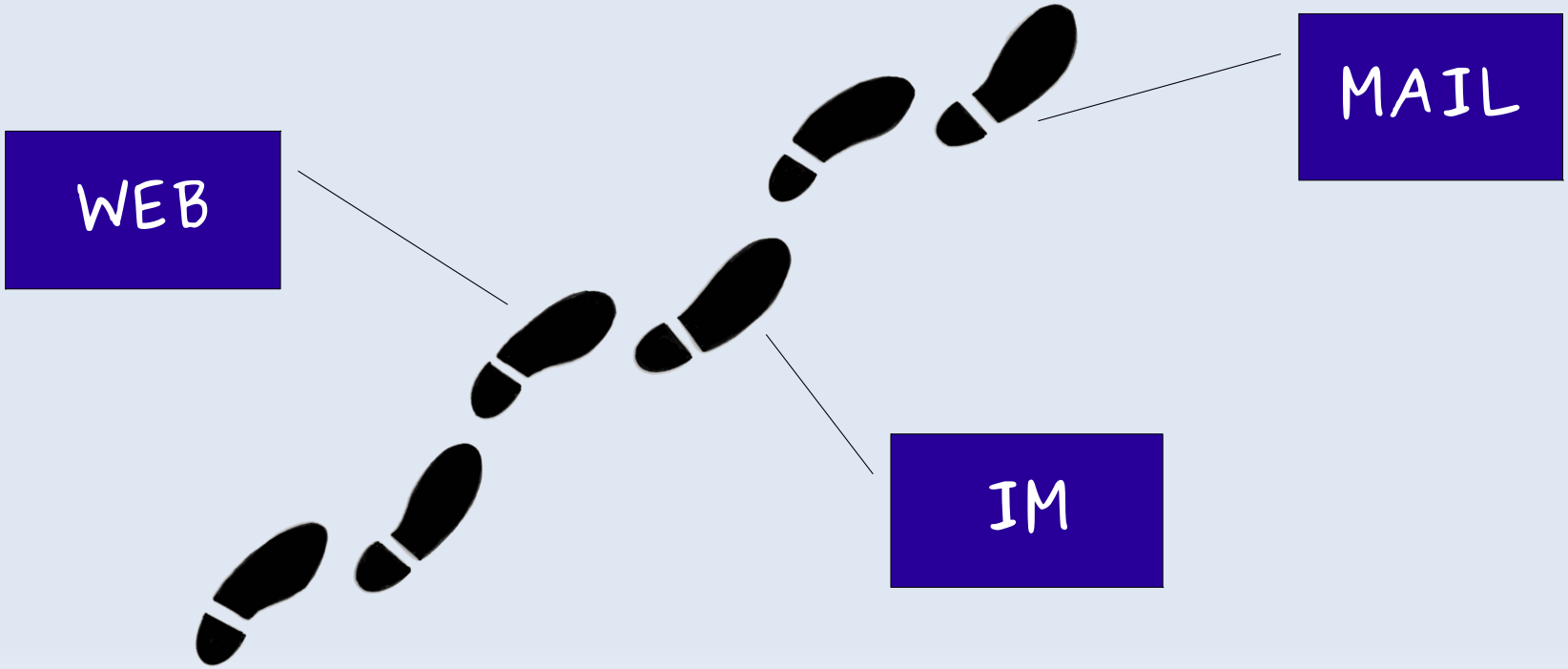
- Motivation
- Setup
- Glance at the Findings
- Flow Signatures
- Evaluation
- Conclusion

Network Flows

- Unidirectional sequence of packets with some common properties that pass through a network device
- *Flow Records* summarize the network traffic seen at an observation point:
  - Src/Dst IP addresses
  - Src/Dst ports
  - Protocol
  - ToS
  - ...

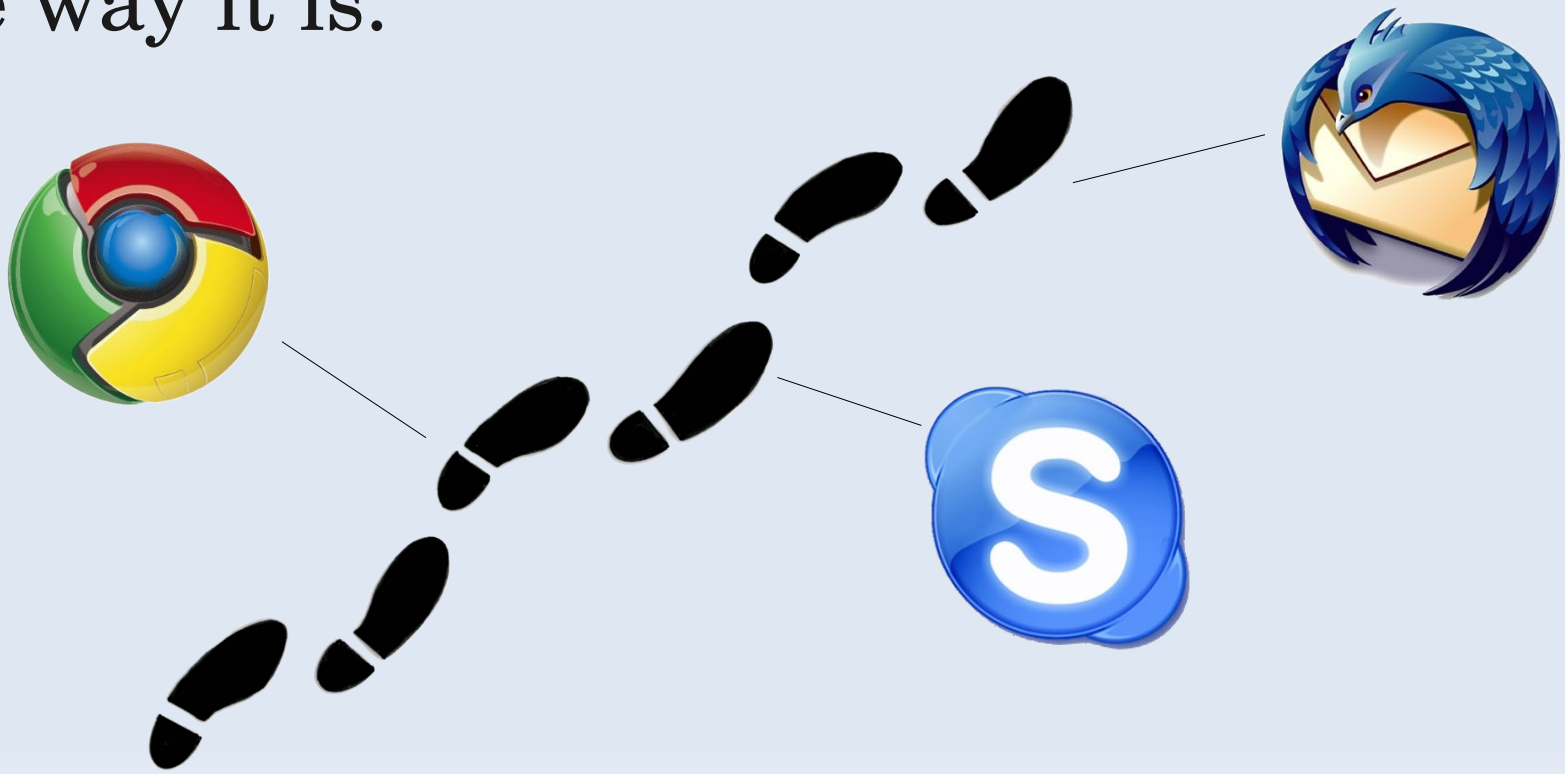
What can we get from it?

- Application identification based on the network activity.
- Has been done before, however only on the level of application protocols.

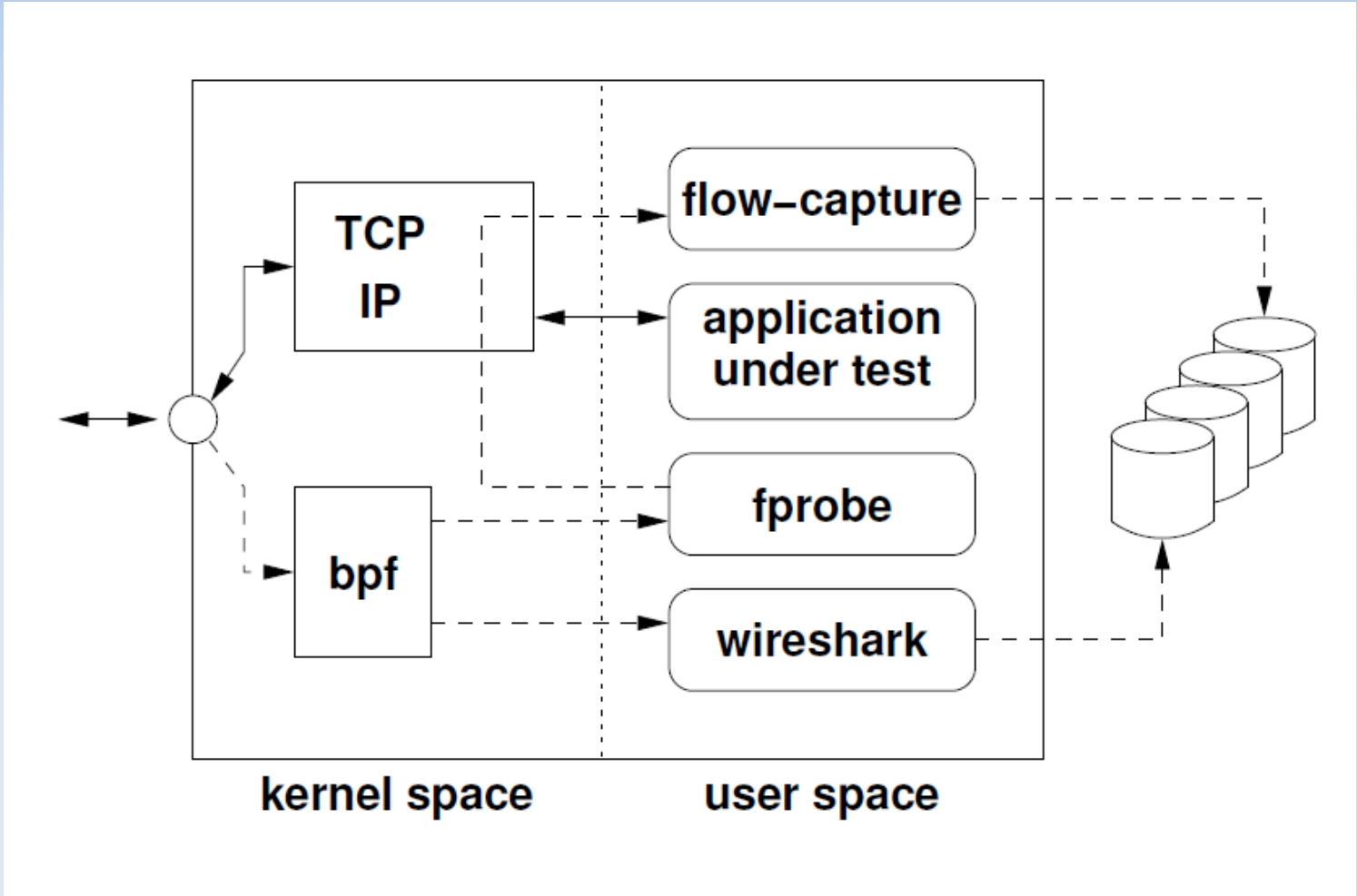


Why not take it a step further?

- Do specific applications generate their own signatures which could be traced?
- Focus on what is observed, not on why it is the way it is.



Experimental Setup



Applications analyzed

A small study was conducted among 60 students of Jacobs University in order to select most popular applications among web browsers, IM clients, mail clients and media players

Applications analyzed

Web Browsers:



IM Clients:




Mail Clients:



Media Players:



Web Browsers

- Round 1 – startup with about:blank
- Round 2 – simple page (www.google.com)
- Round 3 – visiting several web pages
- Round 4 – using RSS and plugins

Browsers do a lot more than a mere user would expect in order to increase their performance and security.

Example – Opera 10.10

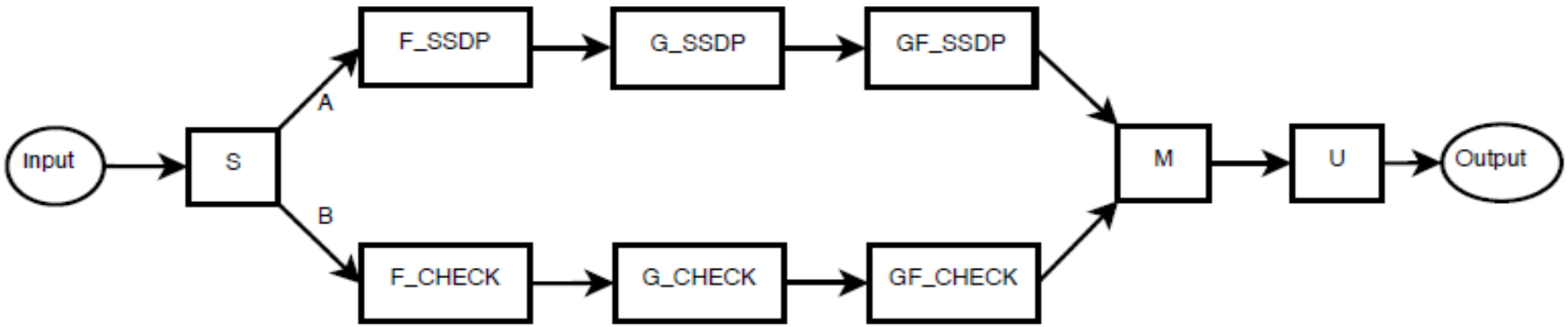
- On start-up – discovery of local Opera Unite users
  - Simple Service Discovery Protocol
    - UDP multicast to 239.255.255.250
- Security check for every website
  - `sitecheck2.opera.com`
    - TCP connection to 91.203.99.45

Flow Signatures

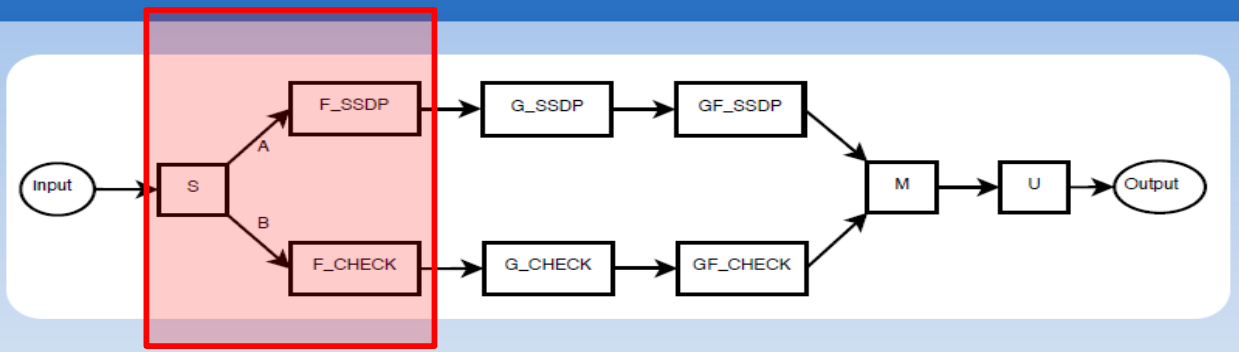
- Formalize flow signatures as queries of the stream-based flow query language *flowy* (developed at Jacobs University Bremen)
- Each query consists of several elements that are connected with each other via pipes

Example – Opera 10.10

```
52   "input.h5"-> S
53   S branch A-> F_SSDP -> G_SSDP -> GF_SSDP -> M
54   S branch B-> F_CHECK-> G_CHECK-> GF_CHECK-> M
55   M -> U -> "output.h5"
```



Example – Opera 10.10




```
 1  splitter S {}
 2
 3  filter F_SSDP {
 4     dstport = 1900
 5     prot = protocol("UDP")
 6     dstip = 239.255.255.250
 7  }
 8
 9  filter F_CHECK {
10     dstport = 80
11     prot = protocol("TCP")
12     dstip = 91.203.99.45
13  }
```

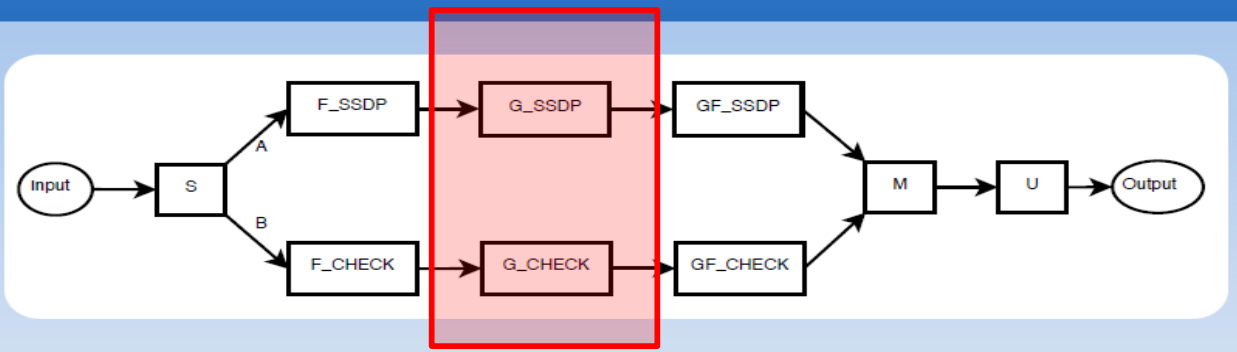
Splitter simply copies the input to all of the defined branches

Filters do absolute filtering on the input flow records

F_SSDP will filter out those flows that correspond to Opera's User Discovery mechanism

F_CHECK will filter out flows corresponding to Opera's security check

Example – Opera 10.10 (cont.)



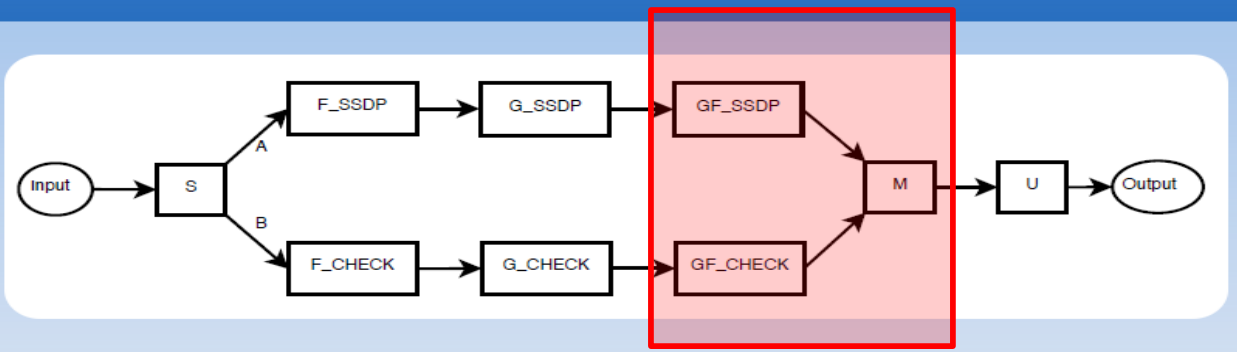

```
15  grouper G_SSDP {
16     module g1 {
17        srcip = srcip
18        dstip = dstip
19        srcport = srcport
20     }
21     aggregate srcip, sum(bytes) as bytes
22  }
23
24  grouper G_CHECK {
25     module g1 {
26        srcip = srcip
27        dstip = dstip
28        srcport = srcport
29     }
30     aggregate srcip, sum(bytes) as bytes
31  }
32
```

Groupers group together several flow records based on some criteria described in the *modules*

Each group contains aggregated information for the grouped flow records

G_SSDP and G_CHECK group together flow records arriving from and destined to the same IP addresses.

Example – Opera 10.10 (cont.)




```
33  groupfilter GF_SSDP {
34      bytes = 516
35  }
36
37  groupfilter GF_CHECK {
38      bytes > 1
39  }
40
41  merger M {
42      module m1 {
43          branches A, B
44          A.srcip = B.srcip
45          A o B delta 1ms
46      }
47      export m1
48  }
```

Groupfilters do the absolute filtering on groups of flow records based on the aggregated information collected by *Groupers*

GF_SSDP checks if the byte size of the group corresponds to three multicast SSDP packets

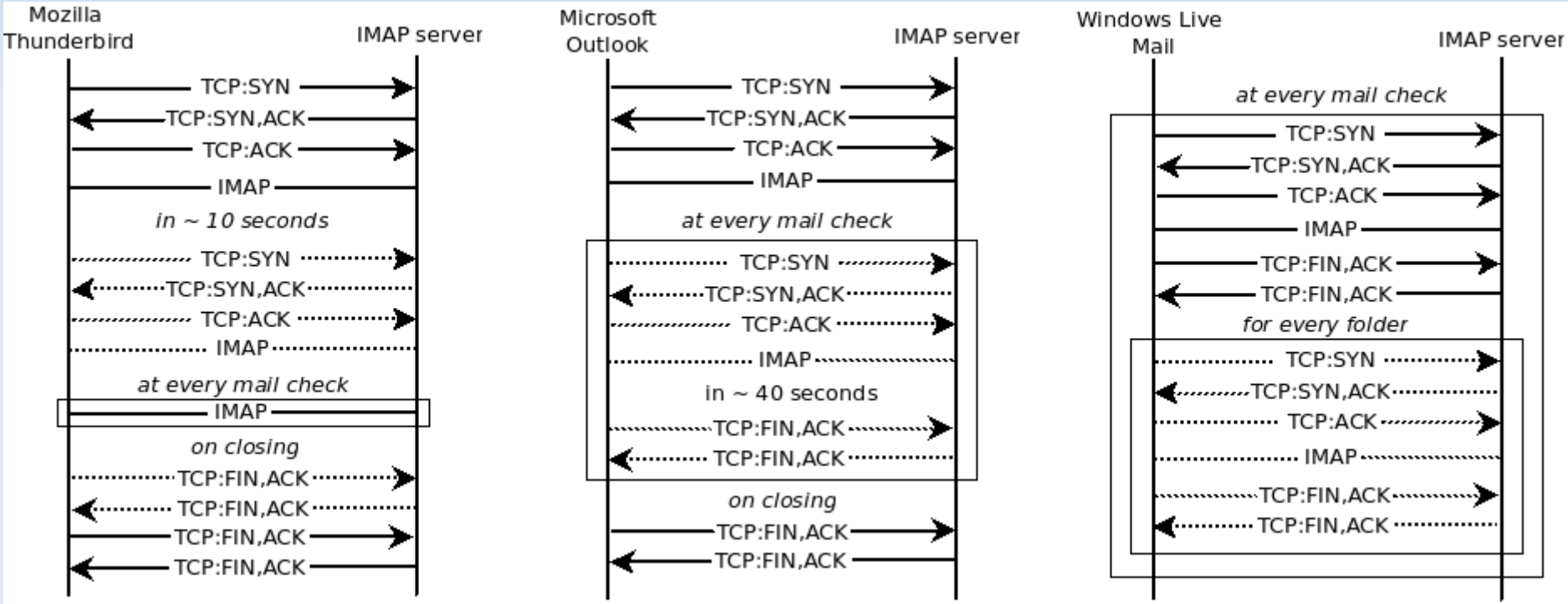
Merger merges groups from several branches based on the rules described in the modules

Further interesting findings

- Google Chrome
  - 3 random Type A and Type AAAA DNS queries
    - Check if ISP does redirecting
- iTunes
  - mDNS PTR queries for `_daap._tcp.local` to 224.0.0.251
    - Share music libraries over DAAP

Further interesting findings

Mozilla Thunderbird vs. Microsoft Outlook vs. Windows Live Mail
IMAP mail retrieval



Evaluation

- “Controlled” evaluation:
  - 10 users
  - 2 weeks of collection period
- Submitted flow traces were run against selected applications' flow signatures
- Results were compared to the expected outcome

Evaluation

User	Skype	Opera	Amarok	Chrome	Live
U0	⊙	○	⊡	○	○
U1	⊙	○	○	○	○
U2	○	○	○	○	○
U3	⊙	○	⊡	○	○
U4	○	○	○	○	○
U5	⊙	○	⊙	⊙	○
U6	○	○	○	○	○
U7	○	⊙	⊙	○	○
U8	○	○	○	○	○
U9	⊙	⊙	⊙	⊙	○

⊙ Correctly identified

○ Correctly not identified

⊡ False positive

Conclusion

- We found that it is possible to identify certain applications by the specific slow signatures they generate
- Formalized flow signatures of several applications using a stream-based flow record query language
- Can be used by network administrators for security and statistical reasons

Thank you!