Authors: Yifei Yuan, Dong Lin, Ankit Mishra, Sajal Marwaha, Rajeev Alur, Boon Thau Loo

Quantitative Network Monitoring with NetQRE

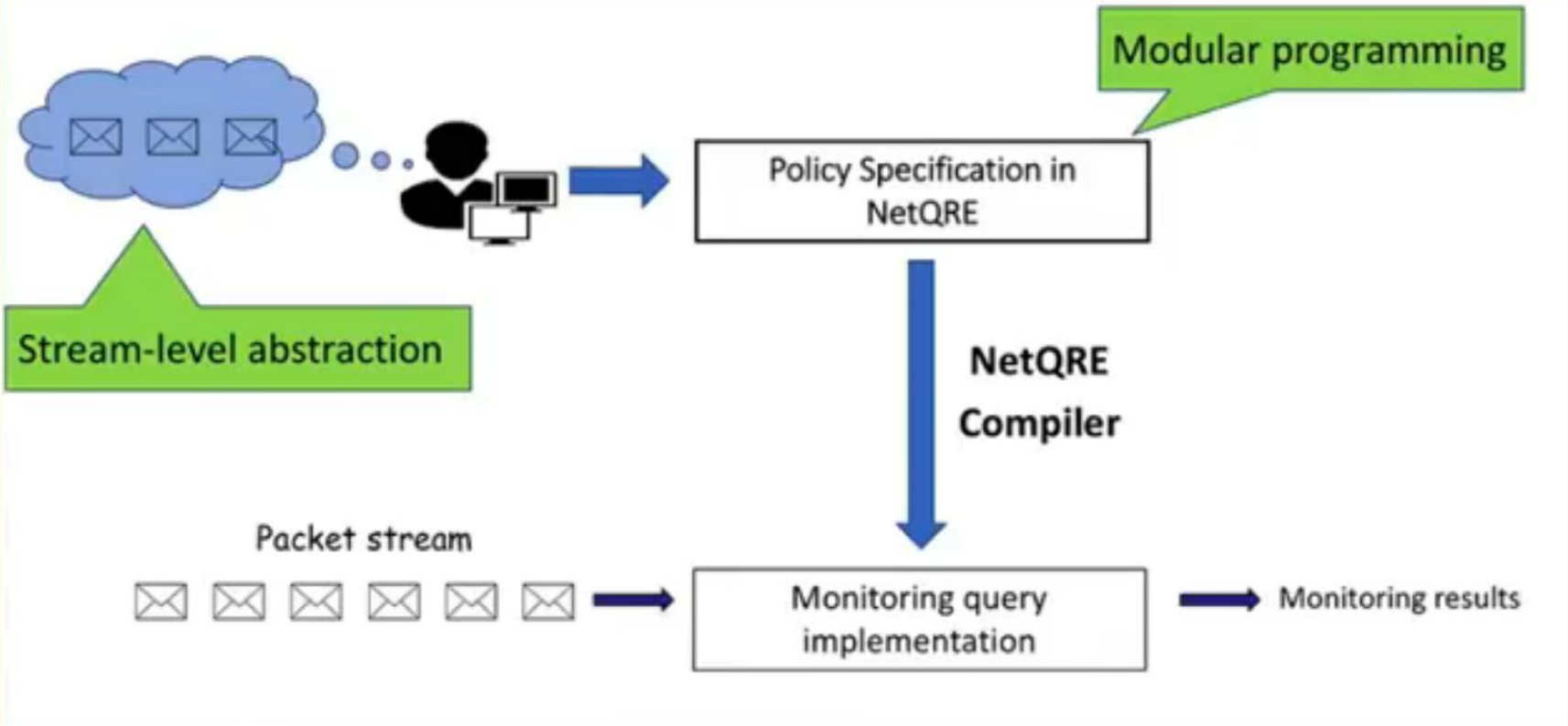
—

Dimitri Wessels
March 17, 2018

Motivation

- Network monitoring is important
- Other solutions focus on different programming abstraction.
- Not possible to do with existing tools, since they are typically focused on flow-level measurements, that do not capture application-level or session-level semantics
- Specify patterns to capture application level semantics

NetQRE Overview




WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

The NetQRE Language

- Input is always a stream of packets.
- Detects patterns of the input packet stream using the extended version of RE proposed by the QRE paper.
- Instead of matching patterns in strings we match patterns in packet headers.
- Some predefined predicates and operators for aggregating and composing streams.

Pattern matching

- Regular expressions
- Predicates: [srcip=’127.0.0.1’]
- Parameters: [srcip=x]
- Conditionals: /[srcip=’1.0.0.1’]/?1:0
- Split and aggregate: split(f, g, aggop)
- Iteration: iter(f, aggop)
- Stream composition: >>

Use Cases

- Flow-level traffic measurements
- Syn flood attack detection
- Application level analysis

Compilation

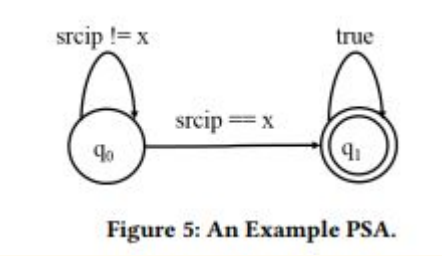


Figure 5: An Example PSA.

- Regular expressions can be translated to DFA
- NetQRE has parameters and iter/split/stream composition which are special cases normal regular expressions don't have to worry about
- The problem of parameters is solved using lazy instantiation
- Special case for split/iter

Evaluation: Expressiveness

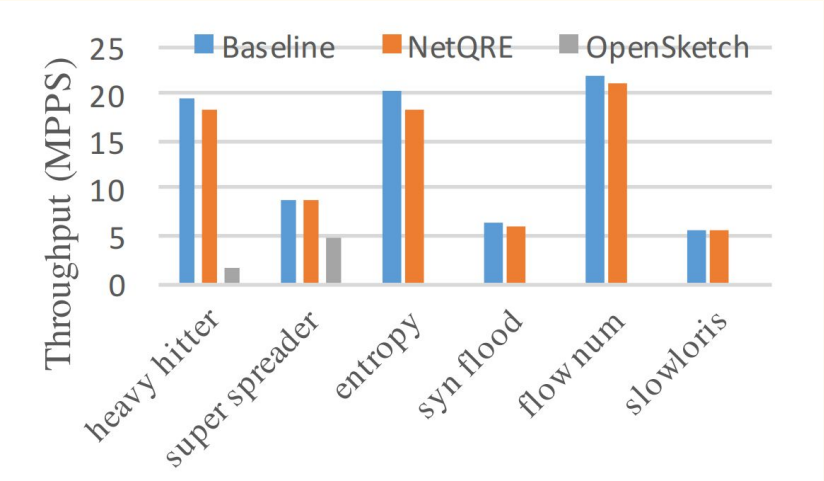
- Expressiveness: Impressive claims, but no code
- "As an interesting comparison we encoded the VoIP call use case in Bro, a well-known intrusion detection system. Bro required 51 lines of code, as compared to only 7 in NetQRE."

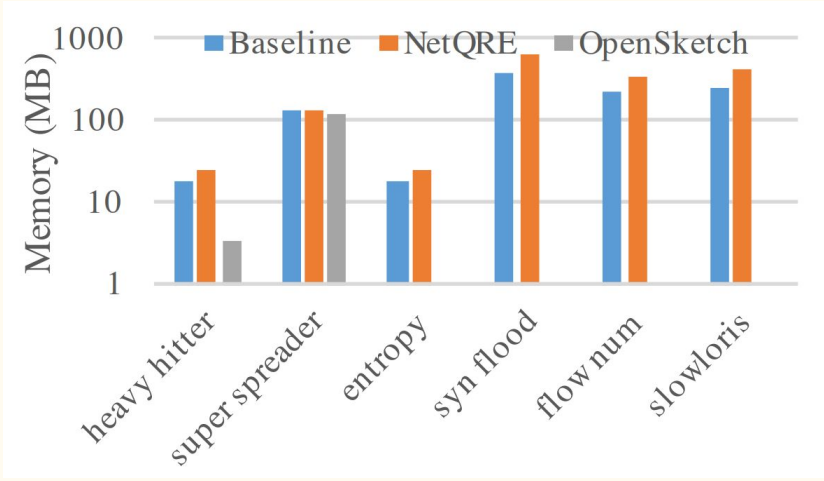
	LoC
Heavy Hitter (§4.1)	6
Super Spreader (§4.1)	2
Entropy Estimation [40]	6
Flow size dist. [18]	8
Traffic change detection [35]	10
Count traffic [40]	2
Completed flows (§4.2)	6
SYN flood detection (§4.2)	9
Slowloris detection (§4.2)	12
Lifetime of connection	8
Newly opened connection recently	11
# duplicated ACKs	5
# VoIP call	7
VoIP usage (§4.3)	18
Key word counting in emails	11
DNS tunnel detection [12]	4
DNS amplification [20]	4

Table 1: Example monitoring applications NetQRE supports.

Evaluation: Performance

- Claim to have built a prototype, but code is not available
- Compare against “carefully hand-crafted C++ implementation, that we spent days optimizing” (code not available)
- Test traffic trace in memory, packets do not contain payloads





Observations

- Very specific and detailed discussion about the NetQRE language
- Code for compiler and evaluation not available - especially odd considering that they show of netQRE code.
- For application layer pattern matching stream of packet needs to be put together. Is that a problem? Memory?
- Language is described with many examples. Sometimes it's not very clear: For example split is introduced as split(f1, f2, aggop) but used in examples as split(f1, f2, f3, aggop)
- iter/split performance?

Potential problems

“Note that this programming model is a conceptual one, and the actual execution is optimized by our compiler. For example, it will incur unacceptable storage and performance overhead if the runtime system has to log all incoming packets and present them as a program input. We have implemented a NetQRE compiler which compiles a NetQRE program into an optimized imperative program“

“We have developed a NetQRE prototype...”

- Didn’t find the implementation of the compiler -> Not possible to repeat results
- How is it so fast

Uh?

RE: Regular expression

PSRE: Parameterized symbolic regular expressions

QRE: Quantitative regular expressions

PQRE: Parameterized quantitative regular expressions

PSA: Parameterized symbolic automaton