# Fast regular expression matching in network traffic

Jan Kaštil

Teoretická Informatika

10.12.2008

# Motivation

- Security applications
  - Snort, Bro, L7 filters
- Modern networks work at the 10 Gb/s speed
  - More than $10^9$ characters per second
- Not all data belongs to one stream
  - More searching units can work in parallel
- Streams can be interleaved
  - Need for storing information about previous search for each stream
  - Possible thousands or million streams

# Finite Automata for pattern matching

- Pattern could be described by regular expressions
- Searching for regular expression is done by Finite automata.
- Need for optimal implementation of Finite Automata.

# Nondeterministic Finite Automata

NFA is 5-tuple (Σ, Q, S, F, δ) :

- Σ is an alphabet
- Q is a set of states
- $S \in Q$ Is a starting state
- $F \subset Q$ Is a set of finite states
- $\delta \subseteq (Q \times (\Sigma \cup \{\epsilon\})) \times Q$ Is a state transition relation

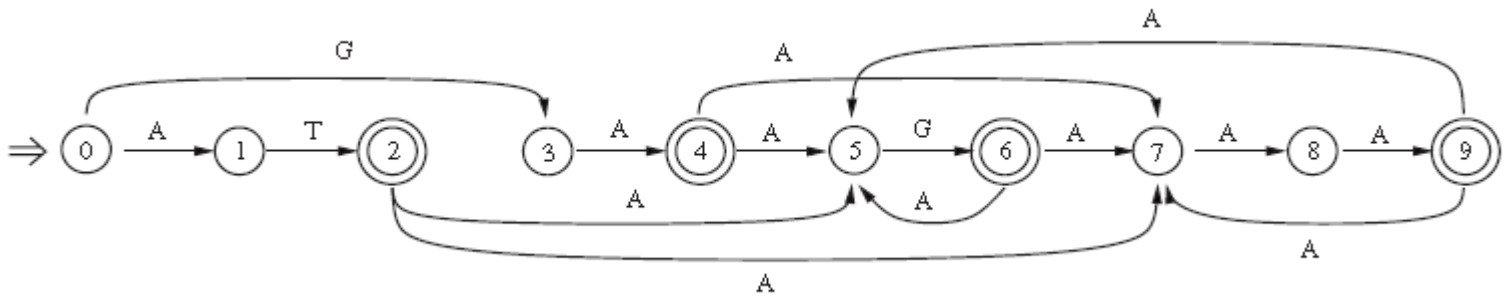
- More than linear time complexity or more than one active state
- Require big amount of data per stream
- Easier HW implementation
- Changing patterns is difficult

# Thompson's construction

- Create NFA for each part of the regular expression and connect them together by epsilon transitions.
- NFA can be easily transform into ε-free NFA and implement in HW
- Commonly used for construction of NFA

# Glushkov's construction

- Properties of Glushkov automaton
  - $\delta \subseteq (Q \times \Sigma) \times Q$
  - $\forall \langle q_1, a_1, q_2 \rangle, \langle q_3, a_2, q_4 \rangle \in \delta :$
    $q_2 = q_4 \Rightarrow a_1 = a_2$
- Example: (AT|GA)((AG|AAA)*)



# Bitparallel implementation

- Each state of NFA is represented by bit vector with only one active bit
  - Each set of active states can be represented by bit vector of the same length
- $(2^{|Q|} * |\Sigma|) * |Q|$ Bits for storing transition table without using any property of the automaton
- Reducing this memory consumption by using property of Glushkov construction
  - Each state has vector of next states
  - Each symbol has vector of states reachable by this symbol
  - Bitwise and of these tables is vector of new states
  - Leads to $(2^{|Q|} + |\Sigma|) * |Q|$ needed bits of memory

# Deterministic Finite Automata

- DFA is NFA with condition that:

$$\forall \langle q_{s1}, a_1, q_{k1} \rangle, \langle q_{s2}, a_2, q_{k2} \rangle \in \delta :$$
$$(q_{s1} = q_{s2} \wedge a_1 = a_2) \Rightarrow q_{k1} = q_{k2}$$

- Can be minimized, yet can have exponentially more states than NFA in worst case
- Only one active state
  - Small amount of memory per stream
  - Only one possible transition
- Small implementation logic and big memory table
  - Implementing DFA can be reduced to fast lookup in transition table

# Reducing Memory Requirements

- Bit-Split algorithm
  - Split input FA into several FA each of them accept part of the input symbol. If all FAs end in ending state, pattern is found.
- Special Memory structure
  - If patterns are only string, FA is a tree. Special structure such as treebitmap can implement transition table.
- Transition table with hash functions
  - Addition memory for collision resolution

# Faster matching

- More matching units work in parallel
  - Fails for faster streams
  - Or scan each packets separately
- Accepting more characters per step
  - Bigger alphabets
  - Bigger transition table
  - Sparse transition table

# Literature

Gonzalo Navarro and Mathieu Raffinot: New Techniques for Regular Expression Searching, Algorithmica (2005) 41: 89–116

Lin Tan, Brett Brotherton, Timothy Sherwoon: Bit-Split String Matching Engines for Intrussion Detection and Prevention

Nathan Tuck, Timothy Sherwood, Brad Calder and George Varghese: Deterministic memory-efficient string matching algorithms for intrusion detection. In IEEE Infocom, Hong Kong, pages 333–340, 2004.