

# Forwarding in a Content-Based Network

In Proceedings of SIGCOMM 2003

Reviewed By Michael Attig

<http://www.acm.org/sigcomm/sigcomm2003/papers/p163-carzaniga.pdf>  
Authors: Antonio Carzaniga and Alexander Wolf

Author's Webpage: <http://www.cs.colorado.edu/~carzanig/>

## Introduction

- Content-based communication
  - Instead of explicit address
- Receivers use *selection predicates* to say what interested in
- Senders just spit out info
- Model
  - Attribute/value pairs
  - Selection predicate logical disjunction of conjunctions
    - [alert-type="intrusion"  $\wedge$  severity>2  $\vee$  class="alert"  $\wedge$  device-type="web-server"]

## Applications

- Publish/subscribe event notifications
- System monitoring / management
- Network intrusion detection
- Data sharing
- And most importantly, distributed games!

## What is it?

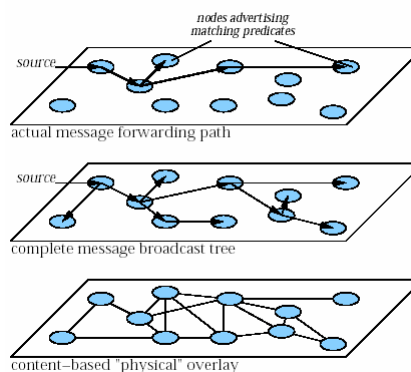
- Overlay network
- Routers perform specialized routing/forwarding
- Routing needs to make paths:
  - Topology → current router
  - selection of predicates → what they describe
- Apply topological constraints first, then determine which match content of message against set of predicates
- Focus of paper – quick forwarding

## Model

- Still need addresses
  - associate predicates with their issuers
  - Maintain topological routing info
  - Manage direct communication between nodes
- Predicate advertised by node is implicitly an address
- Message set of typed attributes

## Routing

- Use advertised predicates to prune branches of broadcast distribution trees
- Router uses 2 protocols
  - Broadcast routing
  - Content-based routing
    - “push” based on recv ads
    - “pull” based on snd requests
- Avoids sending out ads that have already been set up
- Use sender requests to collect routing info



## Algorithm Overview

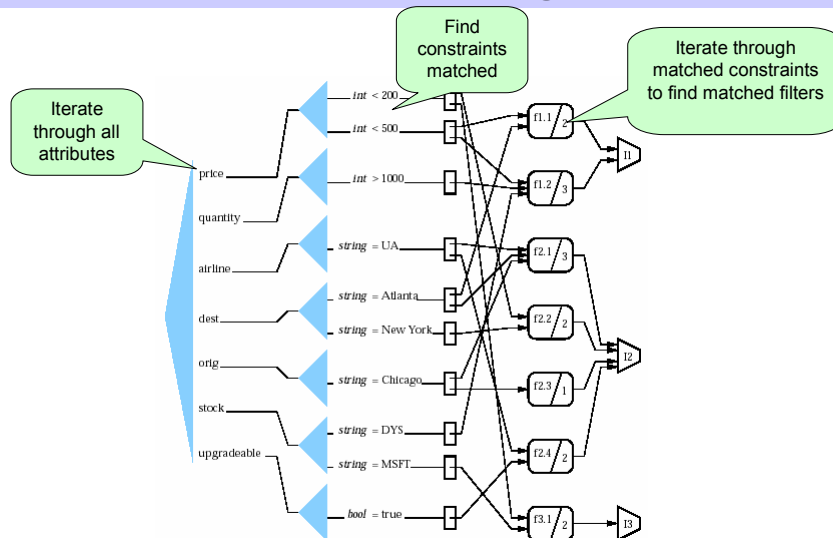
- Forwarding table one-to-one association of predicates to interfaces
- $I_s$  is an interface
- $f_{s,t}$  filter in disjunction of filters for particular  $I$
- Constraints in 3<sup>rd</sup> column
- Use *counting algorithm* to determine which interface predicate matches

I <sub>1</sub>	f <sub>1,1</sub>	string dest = int price < 500
	f <sub>1,2</sub>	string stock = DYS int quantity > 1000 int price < 500
I <sub>2</sub>	f <sub>2,1</sub>	string airline = UA string orig = Chicago string dest = Atlanta
	f <sub>2,2</sub>	string dest = New York int price < 200
	f <sub>2,3</sub>	string orig = Chicago
	f <sub>2,4</sub>	string airline = UA bool upgradeable = true
I <sub>3</sub>	f <sub>3,1</sub>	string stock = MSFT int price < 200

## Forwarding Table

- Organized in 2 parts (left-to-right)
- Left side – index of all individual constraints found in all predicates associated with all neighbors of router
- Connected to right side as Boolean inputs
- Right side implements network of logical connections
  - represents conjunctions of constraints into filters
  - Disjunction of filters into predicates of interfaces

## Representation of Forwarding Table



## More...

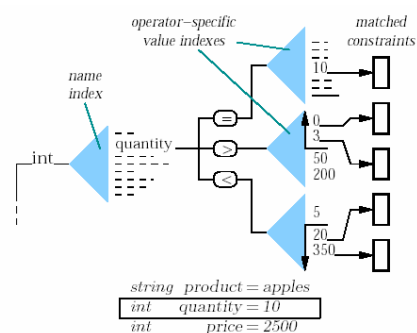
- Optimized to do lookups
- Modifications handled by rebuilding whole table – yikes!
  - Can buffer modification ops to do at manageable rate
- Counting Algorithm
  - Use 2 data structures
    - Table of counters for partially matched filters
    - Set containing interfaces to which messages to be forwarded
  - For each constraint found through constraint index, alg incs the counter of all filters linked from that constraint
  - When reach  $f$  (total # constraints linked), filter satisfied, add interface linked to  $f$  to set of matched interfaces

## Index based optimizations

- Index forming left side aids in speeding up process of finding constraints satisfied
- First stage in index should be based on constrained attribute's name and type
  - use TST for strings representing attribute names
  - Use sub indexes that exploit specific properties of each constraint operator

## Example

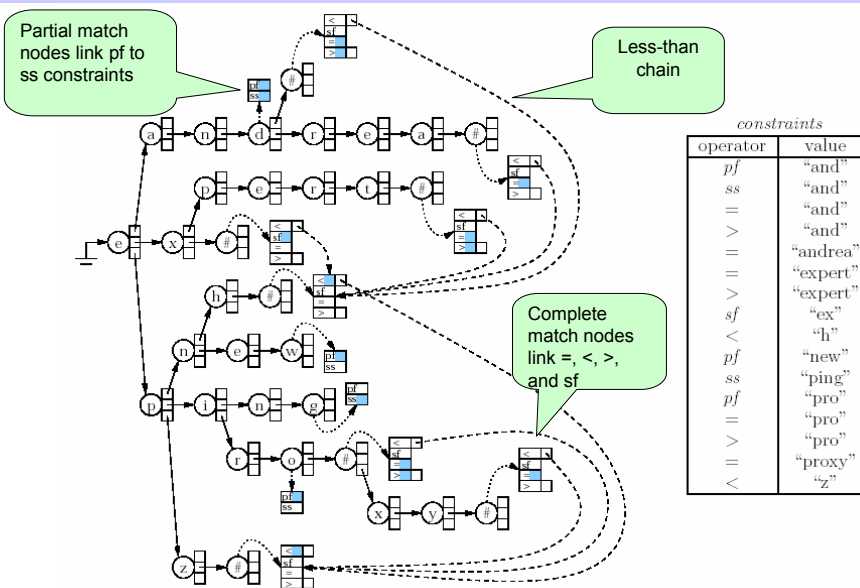
- Index for integer constraints processed against corresponding attribute in message



## Multi-Operator Index

- Supports =, <, >, prefix, suffix, substring operators all in single index
- Uses TST, but extended
  1. Can match partial strings
  2. Ability to link sequence of < and > constraints inserted as leaves in TST
  3. Added backtrack functions to move from partial match to closest complete match

## Example TST



## Steps

- Begin at first character
- Use TST lookup subfunction
  - Recognizes partial matches also
- When partial-match node reached, function returns prefix constraint and/or substring constraint
  - Ptr to internal node
  - Ptr to position reached in input string
- If final node touched leaf, return corresponding =, <, > or sf
- If not leaf, backtrack to two closest leaf nodes
- $O(l(\log N + 1) + |result|)$ 
  - $l$  = length input string
  - $N$  = # strings in TST

## Attribute Selectivity

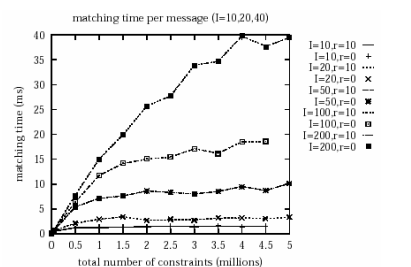
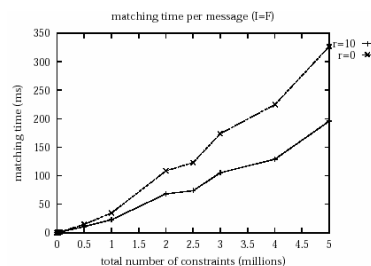
- Save time eliminating interfaces from consideration ASAP
- Call attribute name  $a$  the determinant attribute name for  $I$  if every filter of  $I$  contains at least one constraint on  $a$
- Selectivity Table – associates attribute names with interfaces for which attribute name is determinant
  - Compute intersection of attribute names of all filters for each interface
  - Sort in descending order by # elements in a set of excluded interfaces (largest found first)
- Check this table as pre-processing step to eliminate interfaces that will not match

## Evaluation Setup

- 100 messages (M)
  - 1 to 19 attributes
- 1000 attribute names ( $|D_a|$ )
  - Non-overlapping
  - Taken from dictionary
  - Or can be integer
    - Range of 100 values
- 50% strings, 50% ints

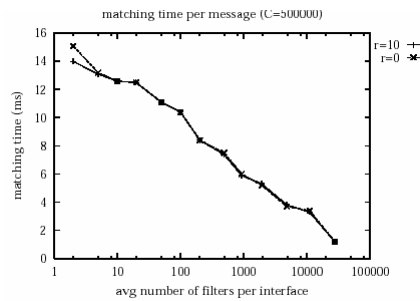
## Results

- Matching time (ms) per message vs. number of constraints
- Compares selectivity table pre-processing rounds
  - $r = 0$
  - $r = 10$
- Top figure models centralized router
- Other models network of content-based routers



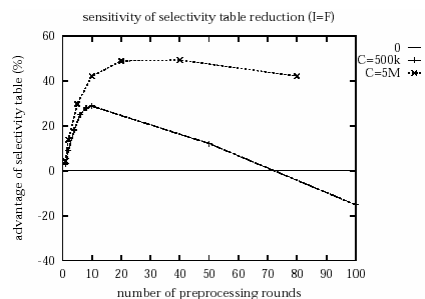
## More results...

- Parameter to determine behavior of algorithm is ratio of filters to interfaces, with fixed # constraints
- Forwarding algorithm performs best with large disjunctions



## Number of Pre-processing rounds

- Reduction of interfaces to look at depends on:
  - Level of selectivity of each name
  - Number pre-processing rounds
- Find point of diminishing returns for rounds



## Network Effect

- Network of these routers outperform single, centralized router?
- Look at end-to-end latency 2 scenarios
  - Single router – latency of 350 ms
  - Many routers – latency of 40 ms

## Conclusion

- Basic short-circuit evaluation of filters greatly reduces processing time
- Selectivity table improves ability to short circuit forwarding function
- Selectivity table has no measurable costs over basic algorithm
- Algorithm good solution for implementing forwarding function in content-based routers