Speccast

Leon Poutievski
Dr. Ken Calvert
Dr. Jim Griffioen
{leon, calvert, griff}@netlab.uky.edu

University of Kentucky
Laboratory for Advanced Networking

ActiveCast Project—Sponsored by DARPA

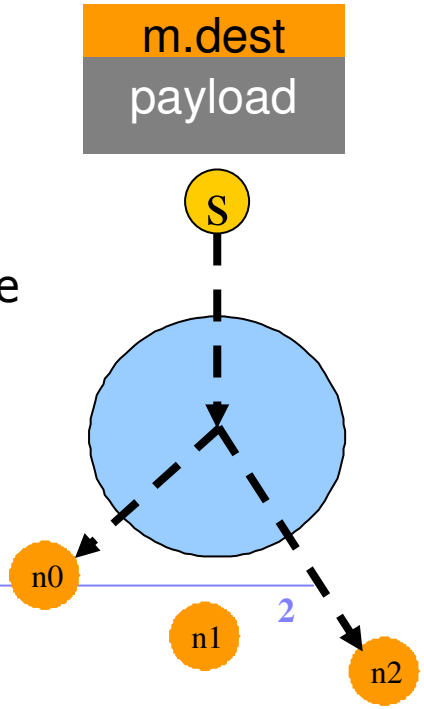
Speccast: General Problem

- Given
  - Network topology, *N* - set of nodes
  - *P* – set of predicates on nodes, i.e. functions: $N \rightarrow$ {true, false}
  - Each packet *m* carries a *destination predicate* m.dest $\in$ *P*
  - *m.dest* specifies a set of nodes to which *m* is delivered, i.e.
- Goal

  deliver *m* to all nodes *n*, s.t. *m.dest(n)* = true

  while minimizing overhead

  - Speccast service subsumes
    - Unicast: provided for each node, there is a point predicate satisfied by only that node
    - Multicast: provided for each multicast group, there is a predicate satisfied by only members of that group
  - The choice of predicate language is important



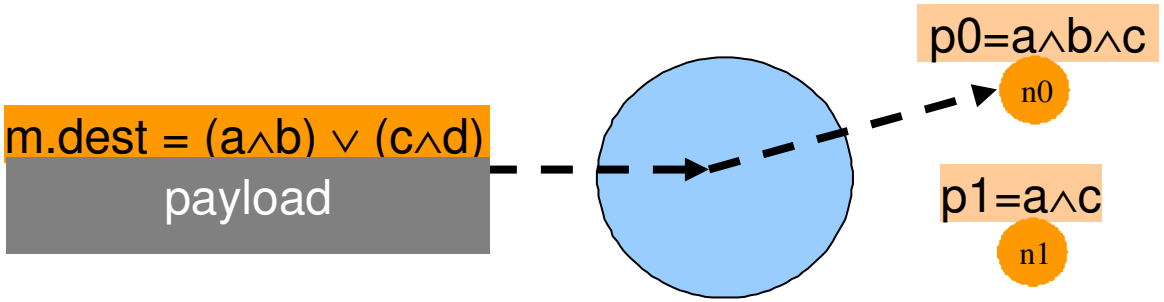
Speccast Routing/Forwarding

- Difference from previous services
  - No underlying unicast layer
    - Network-level service; runs on bare (uncontrolled) topology
  - No predefined address assignment
  - No resolution/discovery before forwarding
- Contributions of this paper
  - Define the new problem
  - Propose a solution for a simple predicate language
  - Investigate the effects of
    - Topology
    - Structure of predicate set
    - Assignment of predicates to nodes (locality)

    on efficiency of our solution
  - Result preview: general solution with modest overhead

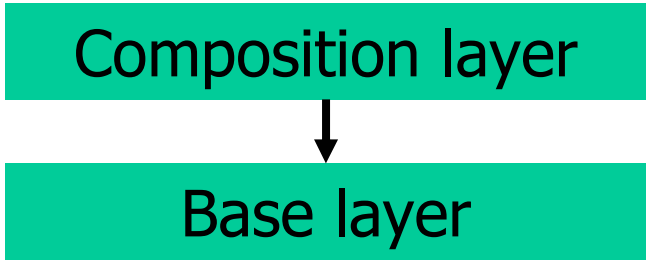
Assumptions for This Paper

- Simple predicate language
  - Disjunctions of conjunctions of positive atomic propositions



- Static topology

Layered Solution



- Base layer
  - deliver to all nodes satisfying a single atomic proposition
- Composition layer
  - deliver the packet using the base layer

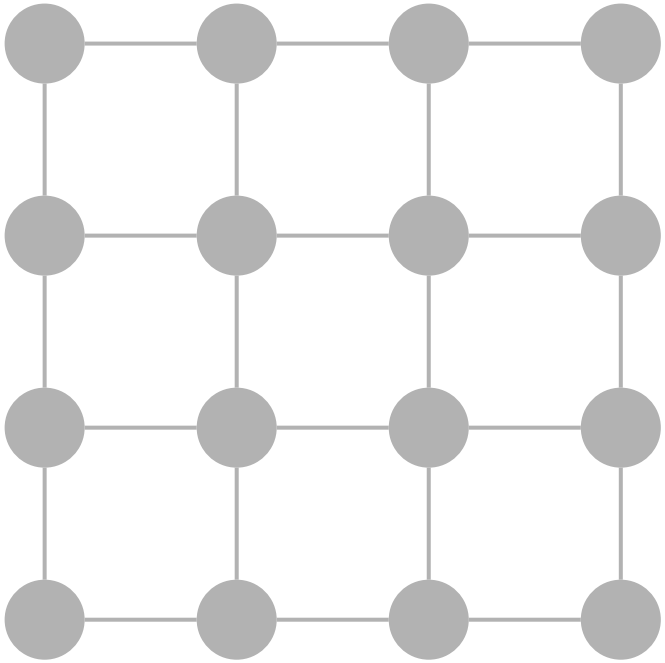
Layered Solution: Base Layer

- Problem:
  Deliver to all nodes satisfying an atomic proposition

- Solution:
  Build a tree spanning all nodes satisfying the atomic proposition
  - Nodes on the tree keep list of tree interfaces
  - Nodes not on the tree know next hop toward the tree

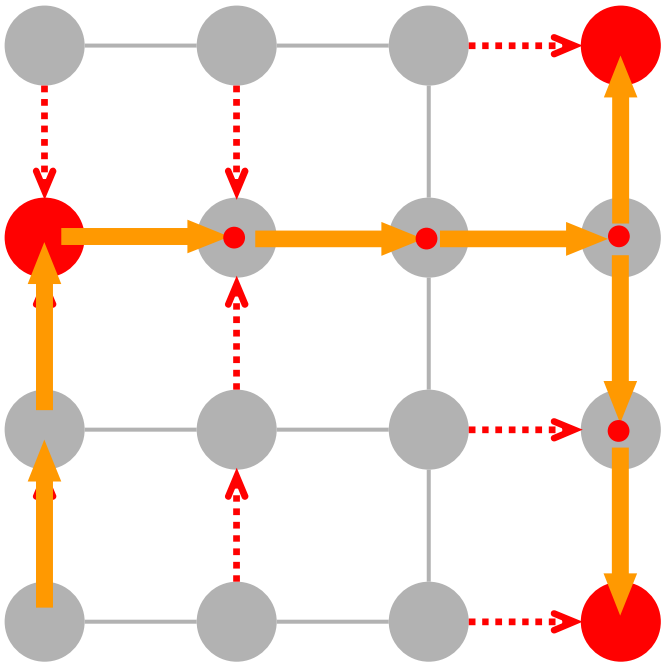
- Approach:
  Distance-vector algorithm
  - Exchange tuples <attribute, tree ID, distance, tree size>

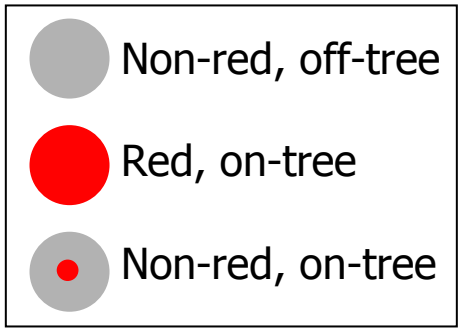
Layered Solution: Base Layer



Layered Solution: Base Layer

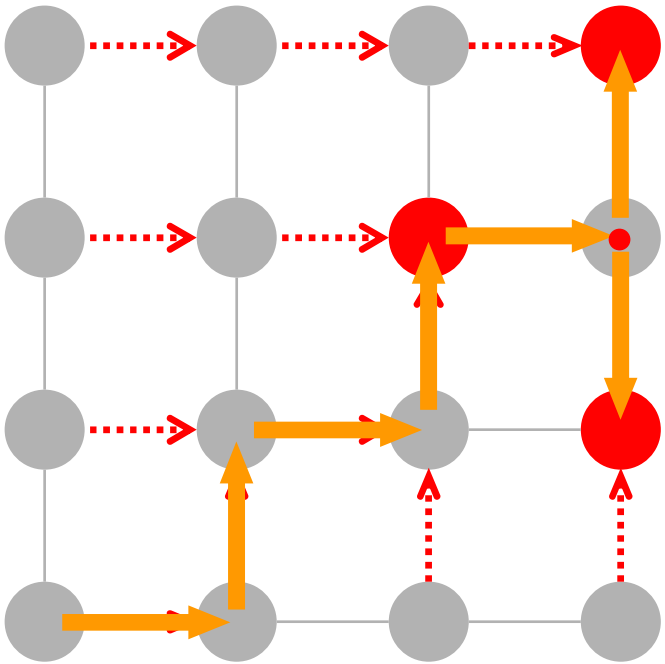
m.dest = red

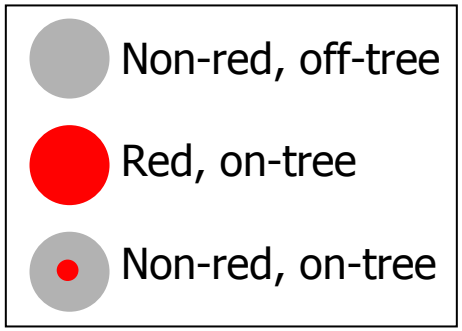




Layered Solution: Base Layer

m.dest = red





- *Locality* – correlation between node's location and set of predicates it satisfies

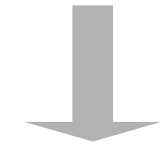
Layered Solution: Composition Layer

- Problem: Deal with more complicated predicates
  - state for arbitrary predicates grows exponentially
- Solution:
  - Disjunction: send to a union of trees
    - For a∨b∨c, send to a,b and c trees
  - Conjunction: send along a single tree
    - For a∧b∧c, send along one of a,b or c trees
- Approach:
  - Select a *single* atomic proposition from each conjunction
  - Forward to *all* trees corresponding to selected atomic propositions using the base layer
- Result:
  - Trade reduced state for overdelivery

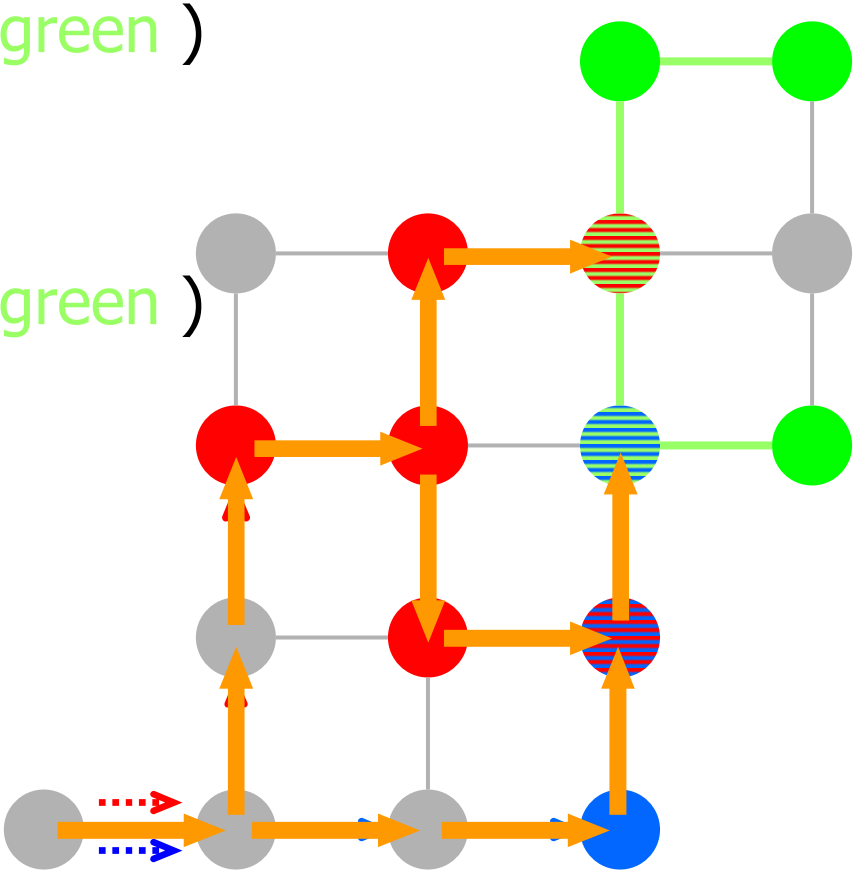
m.dest = red ∨ (blue ∧ green)

Layered Solution: Composition Layer

m.dest = red ∨ (blue ∧ green)



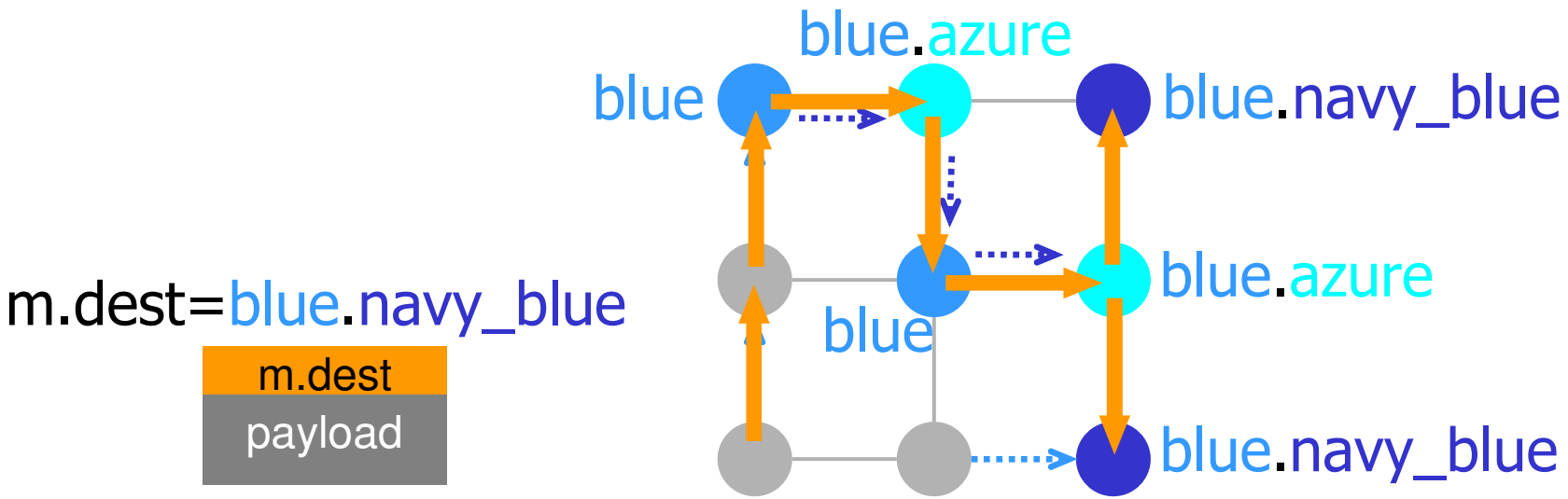
m.dest = red ∨ (blue ∧ green)



Dependent Attributes

- Conventional routing: use hierarchy for scalability
  - E.g. represent set of destinations with a common address prefix
- Idea: impose relationships among predicates
  - Abstraction predicates: $(p_0 \vee \ldots \vee p_k) \Rightarrow P$
- Solution: *hierarchical* atomic propositions

  Example: animal.mammal.human.student.graduate_student



Evaluation Metrics

- *Delay*
  - For each destination node *d*, number of edges on the path to *d*
  - *Stretch* – sum over all destinations over the sum of shortests distances to all destinations
- *Network load*
  - Number of links over which a message is forwarded
  - *Ratio* – ratio to the network load over the SPT
- *Network state*
  - Amount of information stored at all nodes combined
- *Forwarding-time computation*
  - Complexity of forwarding time at each node for each packet arrival

Evaluation

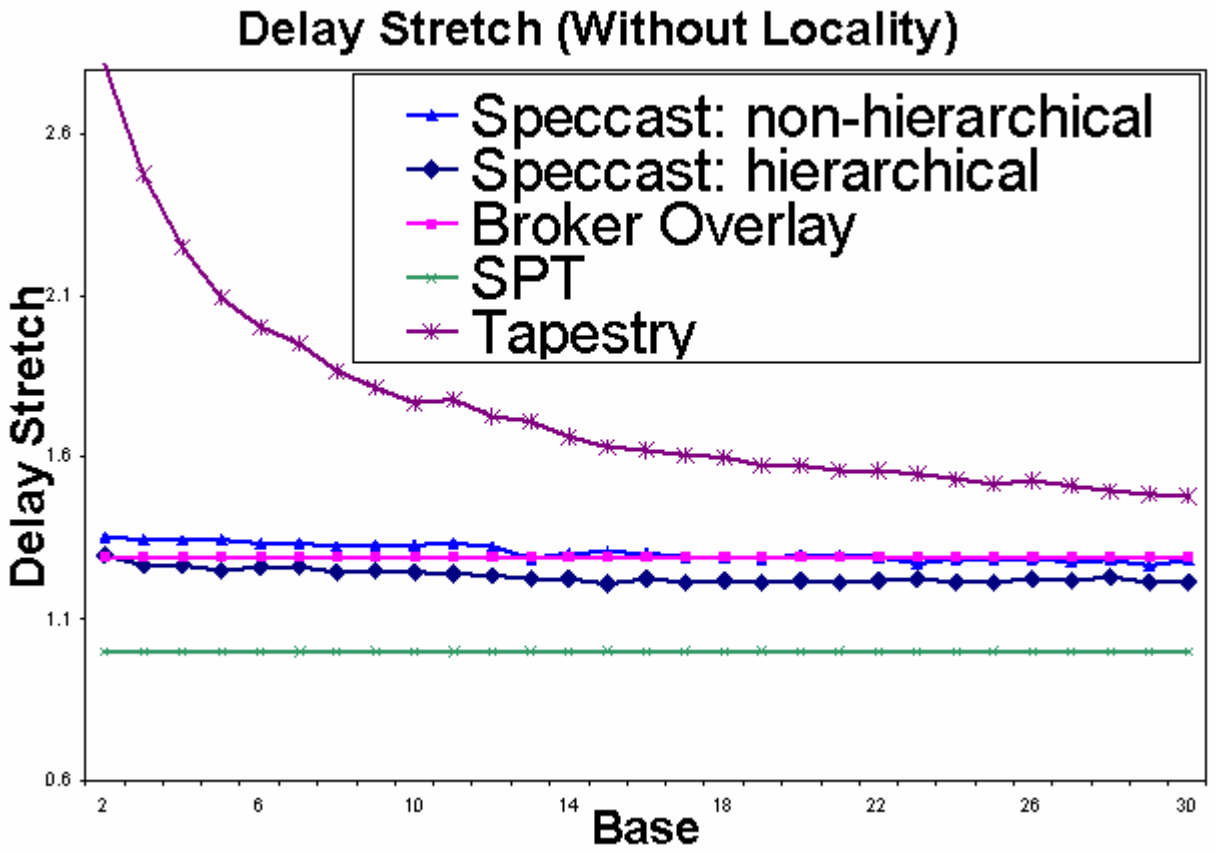
- Solutions Compared
  - Speccast
    - Non-hierarchical (without hierarchical predicate structure)
    - Hierarchical (with hierarchical predicate structure)
  - Shortest-path tree (SPF)
  - Controlled-topology overlay: Tapestry
  - Publish-subscribe overlay: Idealized Broker Overlay
- Usage Scenarios
  - Unicast (with high and low locality)
  - Multi-unicast (with high and low locality)
  - Multicast
  - Random predicates
- Topologies
  - GT-ITM, transit-stub graphs, 3000 nodes

Unicast Predicate Assignment

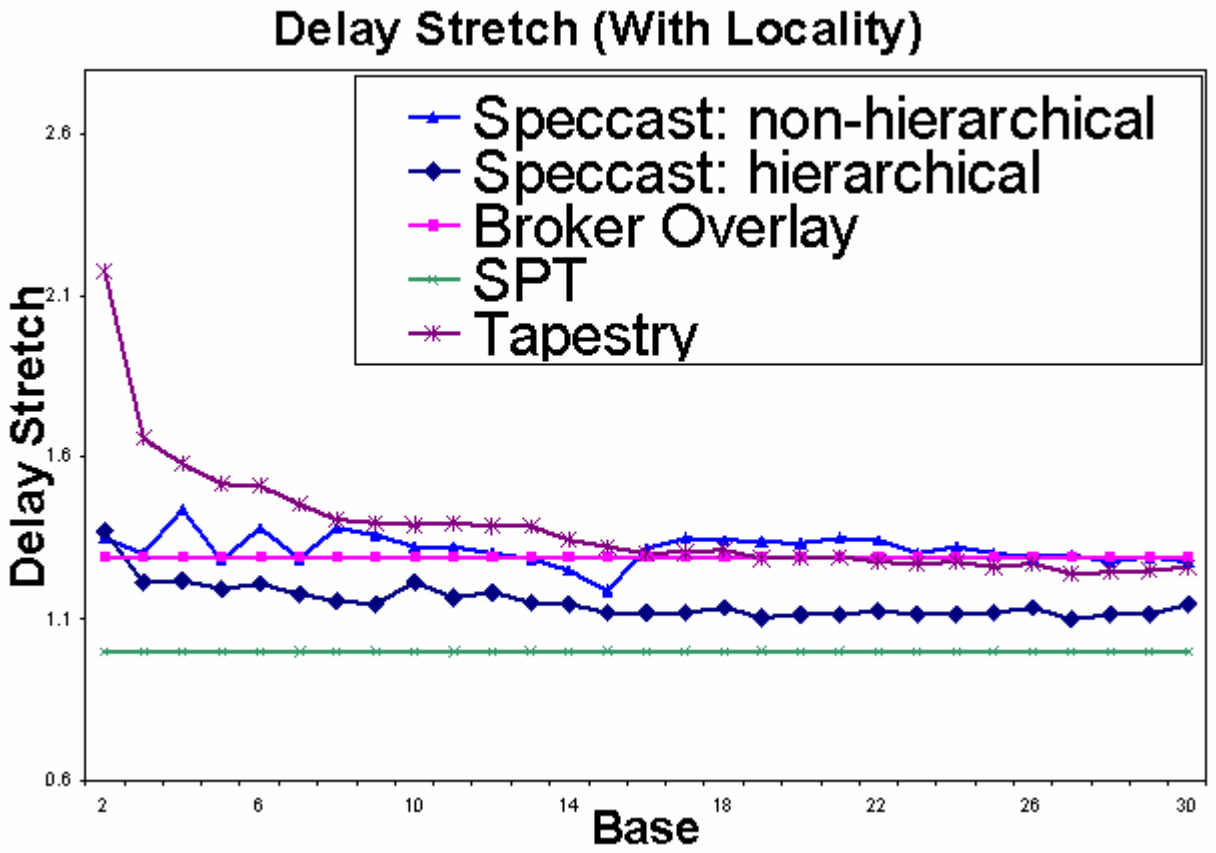
- Each node has a "pseudo-unicast-address":
  - Unique *B*-ary number with $log_B N$ digits, N = number of nodes
  - Node satisfies predicate digit_p_x,
    iff the digit in position *p* of node's number is *x*
  - Set of propositions is partitioned into independent subsets:
    nodes satisfying digit_1_5 will not satisfy digit_1_6

- Example: number 567, base 10
  - Flat: digit_0_7 $\wedge$ digit_1_6 $\wedge$ digit_2_5
    digit[0]=7 & digit[1]=6 & digit[2]=5
  - Hierarchical: digit.5.6.7

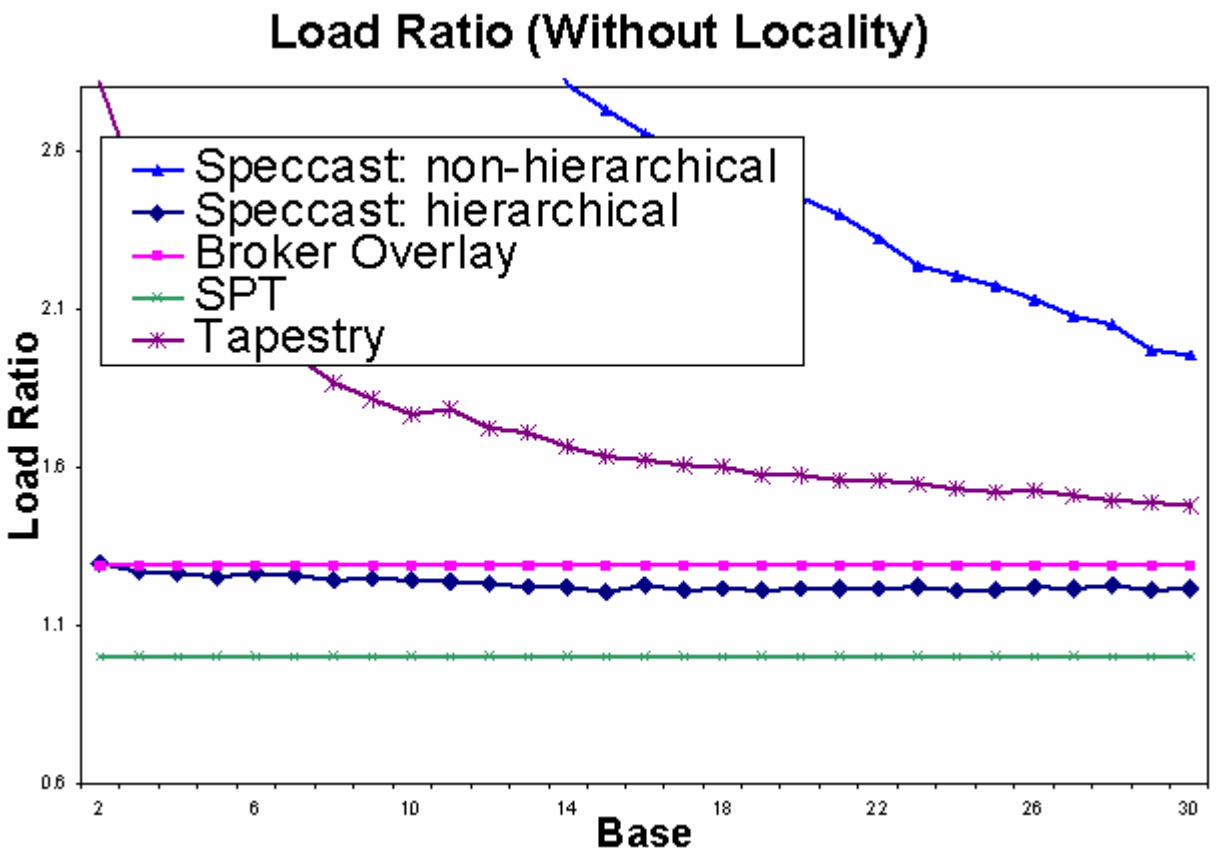
Results


Delay Stretch (Without Locality)
Speccast: non-hierarchical
Speccast: hierarchical
Broker Overlay
SPT
Tapestry
Delay Stretch
2.6
2.1
1.6
1.1
0.6
2
6
10
14
18
22
26
30
Base

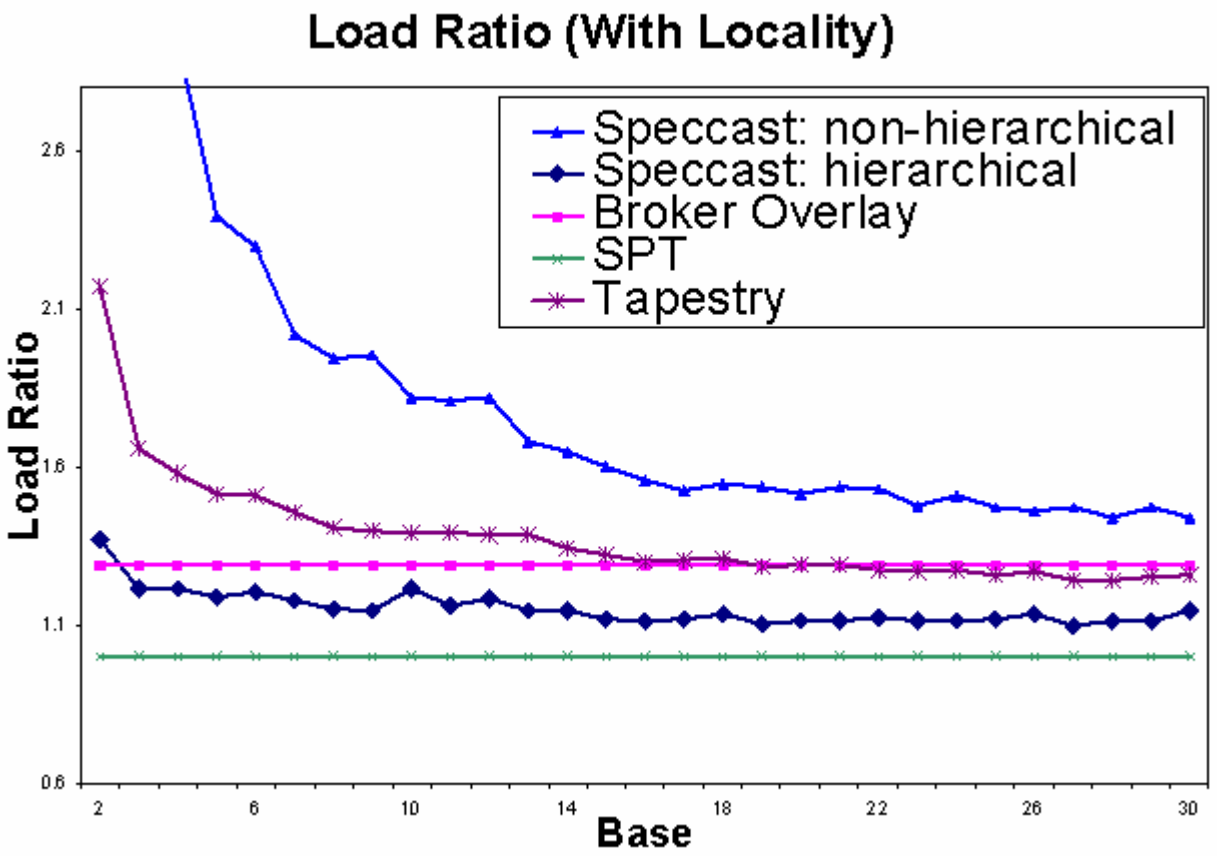
Results


Delay Stretch (With Locality)
Speccast: non-hierarchical
Speccast: hierarchical
Broker Overlay
SPT
Tapestry
Delay Stretch
2.6
2.1
1.6
1.1
0.6
2
6
10
14
18
22
26
30
Base

Results


Load Ratio (Without Locality)
Speccast: non-hierarchical
Speccast: hierarchical
Broker Overlay
SPT
Tapestry
Load Ratio
2.6
2.1
1.6
1.1
0.6
2
6
10
14
18
22
26
30
Base

Results


Load Ratio (With Locality)
Speccast: non-hierarchical
Speccast: hierarchical
Broker Overlay
SPT
Tapestry
Load Ratio
2.6
2.1
1.6
1.1
0.6
2
6
10
14
18
22
26
30
Base

Optimizations

- Filtering (reduce overdeliveries due to flat conjunctions)
  - Additional state to reduce load
  - Shared learning among packets
  - Store knowledge about tree intersections

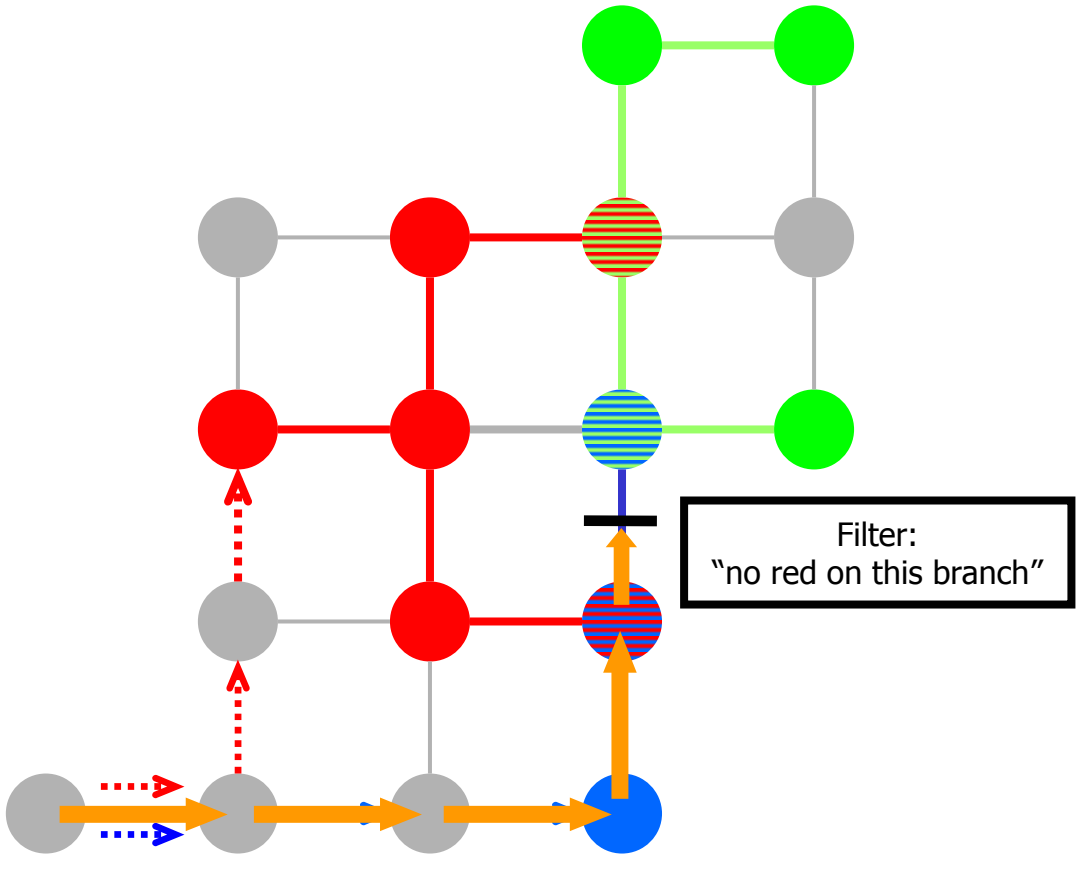
  - *Positive filters*
    - record existence of one tree along some branch of another tree
  - *Negative filters*
    - record absence of one tree along some branch of another tree

  - Default routes
    - Trade-off state for network load in tree like parts of the network
    - Do not store state in nodes with only one or two interfaces
- Compress repeated entries for hierarchical attributes

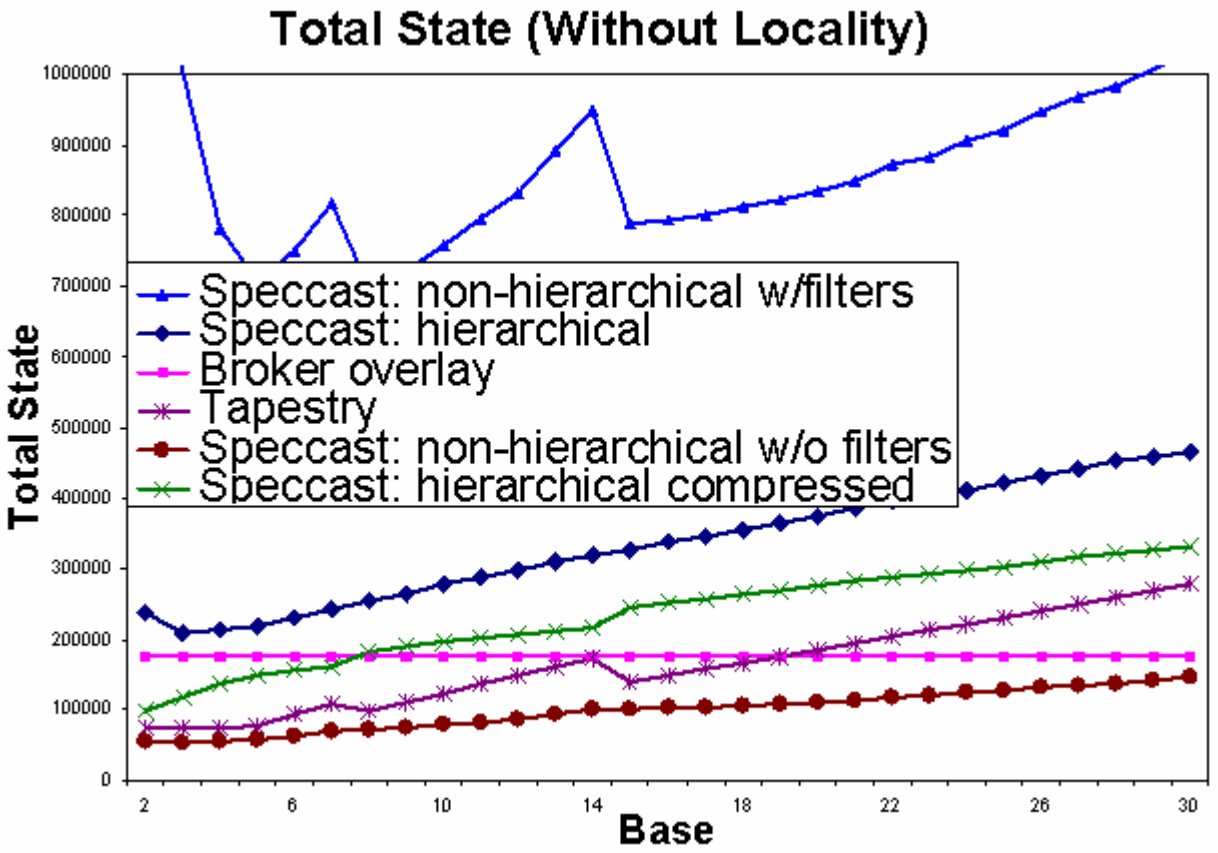
Optimization: Filters

m.dest = red ∧ blue

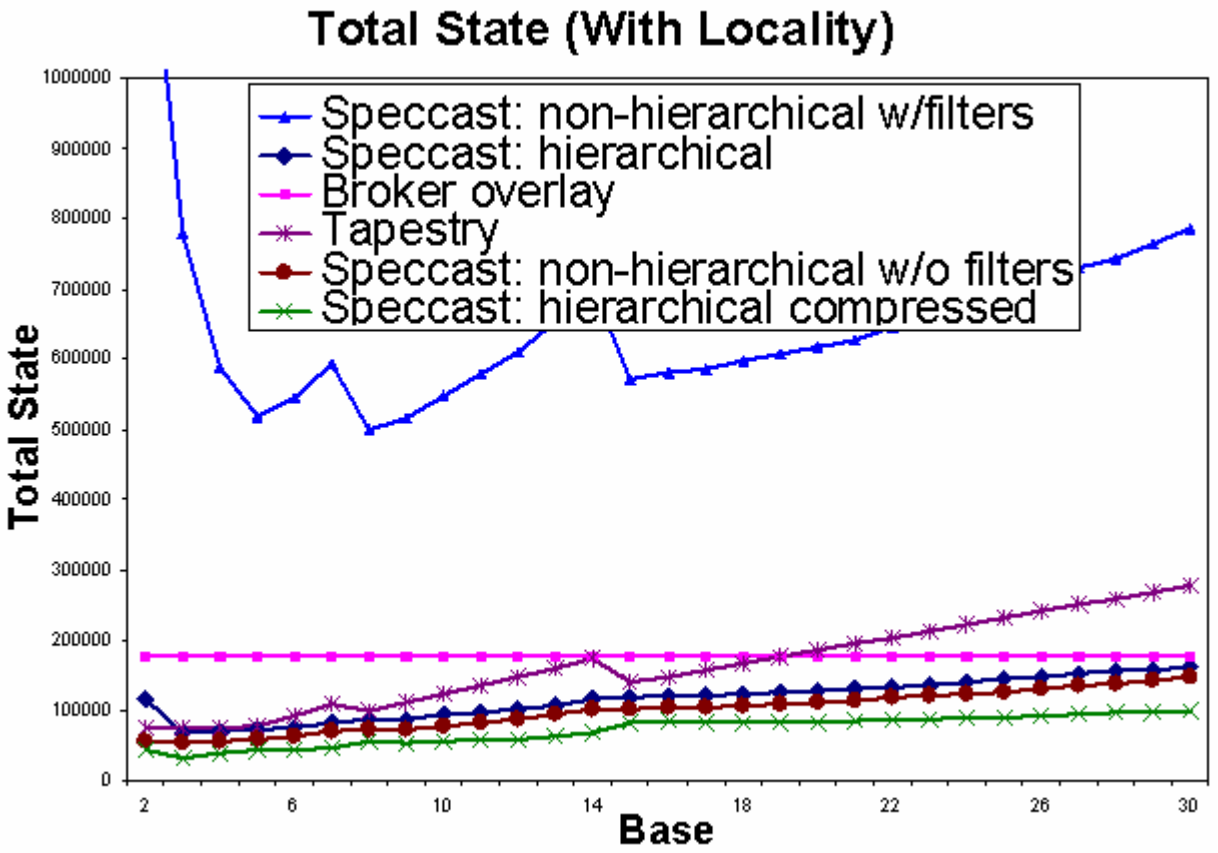
m.dest = red ∧ blue



Results


Total State (Without Locality)
Total State
Base
1000000
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
2
6
10
14
18
22
26
30
Speccast: non-hierarchical w/filters
Speccast: hierarchical
Broker overlay
Tapestry
Speccast: non-hierarchical w/o filters
Speccast: hierarchical compressed

Results


Total State (With Locality)
Speccast: non-hierarchical w/filters
Speccast: hierarchical
Broker overlay
Tapestry
Speccast: non-hierarchical w/o filters
Speccast: hierarchical compressed
Total State
1000000
900000
800000
700000
600000
500000
400000
300000
200000
100000
0
2
6
10
14
18
22
26
30
Base

Related Work

- Publish-subscribe (Siena)
  - Dual problem (receiver vs. sender control)
  - Unicast semantics (point predicates) is unclear
  - Overlay solution
- Overlays
  - DHT (Tapestry)
  - INS (shared tree overlay)
  - i3
- Sensor networks
  - Directed diffusion

Conclusions

- Locality improves all metrics for Speccast
- Hierarchical Speccast overhead is small
  - Delay stretch is the lowest:
    1.1 (with locality), 1.2 (without locality)
- Hierarchical Speccast out-performs
  - Flat Speccast (for unicast and multi-unicast)
  - All other solutions (for unicast and multi-unicast with locality)
  - State grows sublinearly

Questions?

Speccast

Leon Poutievski
University of Kentucky

leon@netlab.uky.edu
http://protocols.netlab.uky.edu/~leon