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: \(N \rightarrow \{\text{true, false}\}\)
  - Each packet \(m\) carries a **destination predicate** \(m.\text{dest} \in P\)
  - \(m.\text{dest}\) specifies a set of nodes to which \(m\) is delivered, i.e.

- **Goal**
  - **deliver \(m\) to all nodes \(n\), s.t. \(m.\text{dest}(n) = \text{true}\)** 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)
  - Result preview: general solution with modest overhead
Assumptions for This Paper

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

- Static topology

\[
m.\text{dest} = (a \land b) \lor (c \land d)
\]

\[
p0 = a \land b \land c
\]

\[
p1 = a \land c
\]

\[
payload
\]
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

\[ m.\text{dest} = \text{red} \]
Layered Solution: Base Layer

- **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 $a \lor b \lor c$, send to $a$, $b$ and $c$ trees
  - Conjunction: send along a single tree
    - For $a \land b \land c$, 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
  
  $$m.\text{dest} = \text{red} \lor (\text{blue} \land \text{green})$$

---

**Speccast**
Layered Solution: Composition Layer

\[ m.\text{dest} = \text{red} \lor (\text{blue} \land \text{green} ) \]

\[ m.\text{dest} = \text{red} \lor (\text{blue} \land \text{green} ) \]
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: \((p_0 \lor \ldots \lor p_k) \Rightarrow P\)
- Solution: *hierarchical* atomic propositions
  Example: \texttt{animal.mammal.human.student.graduate\_student}

\[
\begin{align*}
\text{m.dest} &= \texttt{blue.navy\_blue} \\
\text{m.dest} &= \texttt{blue.navy\_blue} \\
\text{m.dest} &= \texttt{blue.navy\_blue}
\end{align*}
\]
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 shortest 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
    o Non-hierarchical (without hierarchical predicate structure)
    o 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 $\text{digit}_p_x$,  
    iff the digit in position $p$ of node’s number is $x$  
  - Set of propositions is partitioned into independent subsets:  
    nodes satisfying $\text{digit}_1_5$ will not satisfy $\text{digit}_1_6$

- Example: number 567, base 10  
  - Flat: $\text{digit}_0_7 \land \text{digit}_1_6 \land \text{digit}_2_5$  
    $\text{digit}[0]=7$ & $\text{digit}[1]=6$ & $\text{digit}[2]=5$  
  - Hierarchical: $\text{digit}.5.6.7$
Results

Delay Stretch (Without Locality)

- Speccast: non-hierarchical
- Speccast: hierarchical
- Broker Overlay
- SPT
- Tapestry
Results

Delay Stretch (With Locality)

- Speccast: non-hierarchical
- Speccast: hierarchical
- Broker Overlay
- SPT
- Tapestry

Delay Stretch vs Base
Results

Load Ratio (Without Locality)

- Speccast non-hierarchical
- Speccast hierarchical
- Broker Overlay
- SPT
- Tapestry
Results

Load Ratio (With Locality)

- Speccast: non-hierarchical
- Speccast: hierarchical
- Broker Overlay
- SPT
- Tapestry

Load Ratio vs Base
Optimizations

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

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

• Default routes
  o Trade-off state for network load in tree like parts of the network
  o Do not store state in nodes with only one or two interfaces

• Compress repeated entries for hierarchical attributes
m.dest = \textcolor{red}{red} \land \textcolor{blue}{blue}

m.dest = \textcolor{red}{red} \land \boxed{\textcolor{blue}{blue}}

\textbf{Filter:} “no red on this branch”
Results

Total State (Without Locality)

- 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

Base vs. Total State chart showing the performance of different systems.
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