Router Primitives for Programmable Active Measurement

Joel Sommers  
Colgate University  
jsommers@colgate.edu

Paul Barford  
University of Wisconsin & Nemean Networks  
pb@cs.wisc.edu

Mark Crovella  
Boston University  
crovella@cs.bu.edu
Motivation

• Effective network measurement is critical
  • Assess SLA compliance, understand network properties, evaluate network performance

• Path-based assessment requires active probe-based measurements

• Many challenges associated with active measurements
  • Logistical: deploying & controlling measurement hosts
  • Technical: emitting and collecting probe streams with sufficient accuracy and precision, including application of accurate timestamps

• Our position: routers are in a unique position to provide programmatic support for active network measurement
Opportunities and Challenges

• Basic idea: programmatic support for probe generation, reception, and processing in routers

• Potential Benefits
  • No need to deploy additional measurement infrastructure
  • Opportunity to virtually eliminate impact of probes on customer traffic
  • Flexible active measurement capability built into all routers could yield great insight into network behavior and performance

• Key Challenges
  • Defining a set of primitives
  • Router resource management
  • Security and access control
Example: a low-impact record route

Premise: code is installed in routers along a path
Arrival of measurement packet triggers execution of code

Add a timestamp and input interface address to incoming measurement packet

Hold measurement packet until outgoing link has capacity

Add output interface address and timestamp to outgoing measurement packet

input-timestamp  
input-address

forward next-hop when
outputqueue == 0

output-address
output-timestamp
System Goals

• Flexibility in specifying probe emission and processing
  • Assembly-like primitives based on events and actions
• Improve accuracy of active measurement
  • Provide direct support in routers for gathering information along a path
• Ability to limit (or measure) impact of probing on customer traffic
  • E.g., avoid congestion when desired
• Provide secure access for multiple simultaneous users
  • Users obtain capabilities specifying what they’re able to do for a given router
• Support resource usage limits; low impact on router
  • Provide and enforce limits on memory and processor usage
Primitives: Events

- When should code segments be executed?
  - Programmable events trigger code execution

- Types of events
  - Packet arrival
    - E.g., annotate a measurement packet with additional information as it is forwarded along a path
  - Timer expiry
    - E.g., emit probes when timers expire
  - Subsystem state change
    - E.g., when a queue becomes empty, continue code execution (and forward measurement packet to next hop)
# Primitives: Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>Explanation</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set a timer</td>
<td>Schedule a future timer expiration event</td>
<td><code>after time label</code></td>
</tr>
<tr>
<td>Forward a packet</td>
<td>Allow measurement paths to coincide with data point forwarding path, or not</td>
<td><code>forward &lt;address,next-hop&gt; [conditional expr]</code></td>
</tr>
<tr>
<td>Create and send a new packet</td>
<td>Initiate a new probe</td>
<td><code>probe destination [probe spec]</code></td>
</tr>
<tr>
<td>Append a timestamp</td>
<td>Insert timestamp into packet payload (e.g., using IPMP path records)</td>
<td><code>&lt;input,output&gt;-timestamp</code></td>
</tr>
<tr>
<td>Append an interface address</td>
<td>Insert interface address into packet payload</td>
<td><code>&lt;input,output&gt;-address</code></td>
</tr>
<tr>
<td>Append SNMP MIB data</td>
<td>More generally, can consider various passive measurement data</td>
<td><code>&lt;input,output&gt;-mib &lt;mib&gt;</code></td>
</tr>
<tr>
<td>Store a packet for subsequent retrieval</td>
<td>Temporary storage at receiving endpoint to collect measurements</td>
<td><code>store &lt;label&gt; [conditional expr]</code></td>
</tr>
</tbody>
</table>

- **Conditionals**
  - `if cond [action]`
  - `when cond [action]`
- **Definite loops**
  - `repeat var in range`

- **Variables**
  - Variable state saved between invocations of actions
Further Issues

- Resource requirements
  - Code segments can be statically analyzed for CPU and memory resource demands
  - Memory needed for `when` clause processing, packet storage should be modest
    - What if memory fills? (Error propagation mechanisms yet to be determined)

- Access Control
  - Users obtain capabilities
    - Static capability set specifies what language features can be used
    - Dynamic capability set specifies user resource constraints
    - Capabilities may need to be revoked when resource constraints are violated
  - Capability set presented to router upon request to install code
  - Fine-grained capabilities suggest possibility for allowing restricted measurement capabilities to “outsiders”
Example:
standard end-to-end probing methods

Some initialization

Send the probe (consisting of three back-to-back packets)

If this is an even-numbered probe, send a probe in the next time slot
Otherwise, send the next probe at a geometrically distributed interval

Schedule the next probe (use a 5 millisecond discrete interval)

set seq 0
set slot 0
nextprobe:
  repeat i in 3:
    probe 10.0.0.1 udp dport 3000
    payload (slot/4B seq/4B i/4B)
  endrepeat
  if seq % 2 == 0:
    set next 1
  else:
    set next geom-rv
  slot += next
  seq += 1
after next * 0.005 nextprobe

Badabing loss probing

Send the probe (consisting of three back-to-back packets)

Schedule the next probe (use a 5 millisecond discrete interval)
Example: 
“drive-by” passive measurement collection 

Probe could be sent along a path to collect a set of related data
Simple, accurate available bandwidth measurement

Add timestamp and octet count to measurement packet on ingress

input-timestamp
input-mib
1.3.6.1.2.1.31.1.1.1.6

Add timestamp and octet count to measurement packet on egress

output-timestamp
output-mib
1.3.6.1.2.1.31.1.1.1.6
Conclusions and Future Work

• We propose flexible and secure router support for programmable active measurement
  • Event and action assembly-like primitives installed in routers
  • Proposed system would revive and significantly expand measurement capabilities that existed in the original IMPs

• What are the right primitives for service creation and measurement?
  • API-based extensibility mechanisms useful for adding functionalities that do not need to change frequently
  • On-the-fly programmability could be tremendously useful for network measurement

• Currently working on a Click-based implementation in order to develop and better understand aspects of the system