Packet Language for Active Networks

PLAN (Packet Language for Active Networks)
SNAP (Safe and Nimble Active Packets)

Presented by Songlin Cai
Outline

• PLAN
  – Motivation: why Packet Language?
  – Requirement
  – Several Functions
  – Two Examples: Ping, TraceRoute
  – Language characterization

• SNAP
  – Safety
  – Flexibility
  – Efficiency
Motivation

• IP header ----primitive program
• New functionality means change in packet format and its semantics
• Active Network
  – Getting programmability into the network
  – Evolution of network at the pace of technology
Programming Interface

• Two levels
  – PLAN (Packet Language for Active Networks)
  – Node-resident service

<table>
<thead>
<tr>
<th>Language</th>
<th>Packet level</th>
<th>Service level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code location</td>
<td>PLAN</td>
<td>flexible</td>
</tr>
<tr>
<td>Expressibility</td>
<td>in packet</td>
<td>on node</td>
</tr>
<tr>
<td>Authentication?</td>
<td>limited</td>
<td>general purpose</td>
</tr>
<tr>
<td></td>
<td>no</td>
<td>when needed</td>
</tr>
</tbody>
</table>

Table 1: Comparison of the Packet and Service levels
Requirement I

• Flexibility
  – No need to be completely general
    • “little” programs for network configuration and diagnostics,
    • provide distributed communication
  – What Should PLAN Provide?
    • Embody a model of distributed computing
    • Have some simple, transmissible datatypes
    • Cope with and recover from errors
    • Move from fixed set of operations to node-resident services
Requirement II

• Safety and Security
  – Reducing the risk of mistakes or unintended behavior
  – Protecting privacy, integrity, availability under malicious attack
  – What PLAN has?
    • Functional, stateless, strongly-type language: pointer-safe, programs execute independently
    • Authentication of every packet?
      – Possible but not required!
Requirement III

• Performance
  – Offer new functionality without comprise performance offered by current network.
    • Impossible if do packet authentication.
  – PLAN
    • Lightweight
Requirement IV

• Usability
  – Priori assurances about a program’s behavior
  – PLAN
    • Statically typeable and guaranteed to terminate.
    • Based on typed lambda calculus
    • Basic error handling facilities
## PLAN packet

<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunk</td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>top-level functions and values</td>
</tr>
<tr>
<td>entry point</td>
<td>first function to execute</td>
</tr>
<tr>
<td>bindings</td>
<td>arguments for entry function</td>
</tr>
<tr>
<td>evalDest</td>
<td>node on which to evaluate</td>
</tr>
<tr>
<td>RB</td>
<td>global resource bound</td>
</tr>
<tr>
<td>routFun</td>
<td>routing function name</td>
</tr>
<tr>
<td>source</td>
<td>source node of initial packet</td>
</tr>
<tr>
<td>handler</td>
<td>function for error-handling</td>
</tr>
</tbody>
</table>

Table 2: The PLAN packet
PLAN function: onRemote

\[ \text{OnRemote} \left( E, H, Rb, Routing \right) \]

E: an expression of type \textit{chunk}

H: type host

Rb: resource bound

Routing: type host-> host x dev

\[ \text{OnNeighbor} \left( E, H, Rb, D \right) \]
PLAN func: foldr

- \textit{foldr} : (\alpha \times \beta \rightarrow \beta) \times \alpha \ list \times \beta \rightarrow \beta
  
  As for \textit{foldl}, the function argument to \textit{foldr} must be a function name. The meaning of
  
  \hspace{1cm} \text{foldr}(f, [a_1; \ldots; a_n], b)
  
  is
  
  \hspace{1cm} f(a_1, f(a_2, \ldots f(a_n, b) \ldots))

- \textit{foldl} : (\alpha \times \beta \rightarrow \alpha) \times \alpha \times \beta \ list \rightarrow \alpha
fun ping (src:host, dest:host) : unit =
    if (not thisHostIs(dest)) then
        OnRemote(!ping!(src,dest), dest, getRB(),
                    defaultRoute)
    else
        OnRemote(!ack!(), src, getRB(),
                defaultRoute)

fun ack() : unit = print("Success")
Example Two: traceroute

Figure 2: Evaluation of the traceroute program
TracerRoute

fun print_host(h:host, count:int) : int =
    (print(h); print(" : ");
    print(count); print(" ");count+1)

fun ack(l:host list) : unit =
    (foldr(print_host,l,1); print("--\n"))

fun traceroute (src:host, dst:host,
    l: host list, count:int) : unit =
let val this:host = hd(thisHost())
  in
    (OnRemote(|ack|this::l), src, count, defaultRoute);
    if (not(thisHostIs(dst)) then
      let val p:(host * dev) = defaultRoute(dst)
        in
          OnNeighbor(|traceroute|src, dst, this::l, count+1),
          fst p, getRB (), snd p)
  end
else ()
end
Language Characterization

• Flow of control
  – Statement sequence, conditional execution, iteration over lists with fold and exceptions

• The Type System
  – Statically typeable, dynamically checked

• Scoping

• Mutable State
Two More Characterizations

• Error handling
  – Abort service
  – For exception
    • Provide a mechanism for error handling through a special field in the packet header.

• Encapsulation
  – Chunks can be included in the bindings lists of other chunks.
Resource Bound

Figure 5: Resource Cube
Exponential function

```plaintext
fun f1():unit = ()
fun f2():unit = (f1(); f1())
fun f3():unit = (f2(); f2())
fun f4():unit = (f3(); f3())

fun exponential():unit = (f4(); f4())
```
SNAP

• Safety
  – Node integrity
  – Packet isolation
  – Resource safety

• Flexibility

• Efficiency
Resource Safety

• CPU safety
  – All branches in SNAP go forward.
  – All SNAP instructions execute in constant time

• Memory safety
  – For all of the SNAP instructions except one, zero or more arguments are consumed from the stack…

• Bandwidth safety
  – SNAP packet resource bound is decremented upon reception.

• Services and Safety
Flexibility

• PLAN’s flexibility is well-documented in the literature.

• If most of PLAN’s program could be compiled into SNAP, then SNAP’s flexibility is close to PLAN.
Efficiency: SNAP vs IP

Fig. 4. Ping latencies
Efficiency: SNAP vs IP

<table>
<thead>
<tr>
<th></th>
<th>IP</th>
<th>SNAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Per-packet (μs)</td>
<td>71</td>
<td>95</td>
</tr>
<tr>
<td>Per-byte (μs)</td>
<td>0.134</td>
<td>0.135</td>
</tr>
</tbody>
</table>

TABLE II
Per-node switching costs
Efficiency: SNAP vs IP

Fig. 5. Throughput measurements.
Efficiency: SNAP vs PLAN

Fig. 6. Comparing SNAP ping to PLAN ping.