From Protocol Stack to Protocol Heap

-- Role-Based Architecture (RBA)

Bob Braden, Ted Faber
USC Information Sciences Institute

Mark Handley
ICSI Center for Internet Research

HotNets I
Princeton University
October 28, 2002
Outline

• Motivation
• RBA overview
• Using RBA
• Related Work
• Conclusions
Motivation

- The IETF has become an architectural pretzel factory.
  - Layer violations
  - Sub-layer proliferation
    - E.g., MPLS at 2.5, IPsec at 3.5, and TLS at 4.5.
  - Feature interactions
    - Cross-product complexity
  - Erosion of E2E model -- middleboxes
    - Firewalls, NATs, proxies, caches, ...
- A paradise for lovers of complexity
- Can we somehow reduce the complexity and increase the architectural flexibility?
Motivation ...

- Suggestion 1: Replace the traditional protocol layering paradigm with a more general model.
  - Many of these problems seem to be related to traditional layering.

- Suggestion 2: Provide a protocol mechanism to allow “signaling” in-band with the data flow.
  - Image: attach color-code “stickies” to packets in the network.

- These suggestions led to the concepts of Role-Based Architecture (RBA)
  - Note: Giving up layering has profound consequences for how we think about protocols.
What Does Non-Layered *Mean*?

- **Traditional layered architecture**
  - Modularity
    - functional unit for each protocol layer.
  - Packet header format:
    - Sub-header for each layer, forming a logical stack.
  - Header processing rules:
    - Order: Headers processed in order by layer (LOFO)
    - Access: A functional module can read/write only its own sub-header
• **Non-Layered architecture**
  
  – Modularity:
    • Functional building blocks called *roles*.
  
  – Packet header format:
    • An arbitrary collection of sub-headers -- “role data”.
    • These are Role-Specific Headers (*RSHs*).
    • Header data structure is a logical *heap* of RSHs.
  
  – Processing rules: need new rules for order, access.
RSH Processing in a Node

Network Node

Role A

Role B

Role C

Payload

Packet

Heap

RSH 1

RSH 2

RSH 3

read

write

Role-Based Architecture -- Braden@isi.edu
Objectives of RBA (1)

- **Clarity:**
  - Replace “layer violations” with architected role interactions
- **Flexibility**
  - Flexible relationships among roles
- **Extensibility**
  - Roles are modular and hopefully orthogonal. No layer restrictions.
- **Inband Signaling**
  - RSHs can act as “stickies” -- e.g., to control middle boxes.
- **Auditability**
  - Can leave RSHs after they have been “consumed”, to signal to downstream nodes that a function has been performed.
Objectives of RBA (2)

- **Portability**
  - Allow roles to be sited arbitrarily on nodes.
    - *For extra credit: mobile* roles that migrate among nodes

- **Re-Modularization**
  - Current protocol layers are large and complex; with RBA could re-modularize into smaller functional units
    - This is not a new idea
    - It is unclear how far one should go towards micro-roles
    - But RBA gives us freedom of choice on functional granularity

- **Security**
  - Hide particular role data (*Don’t muck with my meta-data!*)
  - RSH might be unit for encryption of role data
Role-Based Architecture

Outline
- Role Data
- Role Definition
- Naming and Addressing
- Processing Rules
- Trivial Example
- Implementation: Packet Layout
More About Role Data

- RSHs can be added, modified, or deleted as packet is forwarded.
- RSHs subdivide the header information (meta-data) along role boundaries.
- Granularity of RSHs is an important design parameter
  - Trade off processing overhead against reusability
- RSHs generally carries metadata, but some may modify processing just by their presence.
Defining Roles

• Roles communicate with each other only via RSHs
  – (for mobility)

• Role will often have some local API to node software.

• Role specification:
  – Its internal state, its algorithms, its APIs, and the RSHs it will send and receive.

• Successive refinement
  – Derive role specification from a generic functional definition.
  – For reasoning about protocols and for developing new roles.
More about Roles

- **Actor**: the program/process that implements a role in a particular node.
- **Roles are often coupled in conjugate pairs**
  - E.g., \{Encrypt, Decrypt\} \{Compress, Expand\} \{Fragment, Reassemble\}
  - ((Our concepts get mushy here... Is a conjugate pair one distributed role with two actors, or two interrelated roles?))
Naming and Addressing in RBA

• Role type is identified by unique name: RoleID
  – “Color-coded”

• We assume an address space for node interfaces {NodeID}

• RSHs are addressed to role(s)

  \[ <\text{RoleAddr}> ::= \text{RoleID}@\text{NodeID} \mid \text{RoleID}@* \]

  Wildcard NodeID: RSH will be processed by any instance of the RoleID that it encounters along the path.

  Symbolically, an RSH is:
  \[ \text{RSH}( \ <\text{RoleAddr}>, \ldots \ ; \text{RSHbody} ) \]
  (More accurately: \text{RSH}( \ <\text{RoleAddr}>:\langle\text{access bits}\rangle, \ldots \ ; \text{RSHbody} ))
Processing Rules

• A Role R on node X may access an RSH if:
  (1) The RSH is explicitly addressed to R
      RoleAddr = R@X or R@*,
  (2) or R is promiscuously listening for RoleID R’ that is addressed
      by the RSH.
      Either may be restricted by access control bits.

• Enforce Sequencing rules
  – Legal ordering of conjugate roles
    • compress -> expand, or encrypt -> decrypt
  – Proper nesting: compress -> encrypt -> decrypt -> expand
  – Use presence/absence of RSHs (between nodes) plus
    precedence rules for roles (within the same node).
Trivial Example Using RBA

- Trivial example: “UDP datagram”

```plaintext
{ RSH( LinkLayer@*; MAC-src, MAC-dest),

RSH( HBHforward@*; dest-NodeID ),
/* role in every router */

RSH( DestApp@dest-NodeID; AppID, payload ),
/* generic application role: deliver payload to AppID */

RSH( HBHsource@*; src-NodeID ),
}
```
Possible RBA Packet Layout

```
<table>
<thead>
<tr>
<th>RoleID</th>
<th>RoleID</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeID or zero</td>
<td>Flags Stack Chain Byte Offset Access Bits</td>
</tr>
</tbody>
</table>

Element of Index Vector

RSH format

Payload

Heap Area

RSH Body
```
Using RBA -- Possibilities

• Pure RBA architecture
  • All functions, from current link layer to applications, using roles.

• RBA only above the Link Layer
  • Probably want to treat the link layer as god-given.

• RBA only above IP layer
  • Retain forwarding efficiency of IP in routers.
  • RBA overhead then only in end systems and middleboxes

• RBA only in app layer
  • We need an application layer architecture; RBA could be a nifty framework for it. Would still help immensely with middleboxes.

• RBA only as abstraction for reasoning about protocols.
Related Work

• Hasn’t this all been done before? Not really...

• Modular construction of protocol stacks

• Protocol decomposition into micro-protocols
  – For re-usability & customization --
    O’Malley & Peterson 1992, Bhatti&Schlichting 1995,
    Kohler et al 2000 (Click), Kohler et al 1999 (Prolac).

• These all focused on protocol implementations, not on the
  protocols themselves.

• RBA is orthogonal concept; in fact, the earlier work may provide a
  basis for realizing RBA.
Conclusions

• This is a position paper.
  – We have not yet built an RBA prototype, although a USC grad student is working on it.
  – We have worked through some simple examples.
  – Some of the basic definitions are still subject to debate.

• I hope I have convinced you that a non-layered approach to protocols might not be totally crazy.
  – But we are so used to thinking in a layerist manner that using RBA does twist the head a bit.
Conclusions

• RBA might be:
  – The Next Great Thing in networking, or
  – only useful for re-organizing particular protocol layers, e.g., the application layer, or
  – only an abstraction for reasoning about protocols.

• RBA appears to have considerable richness and scope for further research.