

Using Distributed Object Technologies for Network Management

George Pavlou
Centre for Communication Systems Research
University of Surrey, UK

<http://www.ee.surrey.ac.uk/CCSR/Networks/>

- After the emergence of CORBA, a lot of research has taken place in the mid- to late-1990's on using **Distributed Object Technologies (DOTs)** for management
 - DOTs are naturally suited to **service management** which has a lot to do with flow-through automation and process re-engineering
 - **Network management** is more about information retrieval and manipulation in real-time, more difficult to readily deploy DOTs
- The X/Open-TMF Joint Inter-Domain Management (JIDM) Task Force produced generic mappings from SNMP SMI and OSI-SM GDMO to CORBA IDL and proposed interaction mappings
 - The approach was never deployed per se in a large scale but variations of it have and are being used in telecom environments
 - Mostly through semantic rather than generic/syntactic mapping of existing information models to CORBA IDL
- Based on these efforts and lessons learned, a **simple semantic approach** is proposed to be possibly used for Internet management

- GDMO objects have attributes and actions (methods)
- At the object boundary, possible retrieval operations are:
 - `get(attrNameList)`, `get(allAttrs)`, `get(noAttrs)`
- At the Agent/Protocol level (CMIS/P), operations for information retrieval are:
 - `Get(objName, scope, filter, attrNameList|allAttrs|noAttrs)`
 - Scope works on tree-like MIB structure
 - 1 linked reply per selected object + an empty series terminator
- Tables are modelled as multi-instance objects “hanging” from a container object
 - Access: `Get(containerObjName, scope=1stLevel, noFilter, allAttrs)`

- **SNMP objects are simple scalar entities (of integer, string, OID type) that can be read and written**
 - **Similar to attributes in GDMO and DOT information models**
 - **No explicit association of objects modelling an entity, e.g. a protocol machine, a table entry, etc.**
- **At the Agent/Protocol level, operations for information retrieval are:**
 - **Get(objNameList)**
 - **GetNext(oidList)**
 - **GetBulk(oidList, repetitions) – simplified**
 - **GetNext and GetBulk work based on the lexicographically linear MIB structure, used mostly for table retrieval**
- **(2-dimensional) Tables are modelled as rows of objects**

- **Primarily distributed software frameworks, with simple request-response (RPC-like) protocol**
 - **Formal specification of object interfaces e.g. CORBA IDL**
 - **Simple to use API with stub objects in local address space**
 - **Strict-typing against loose SNMP-CMIS/P typing**
- **Advantages: method support, simple to use API, multiple language bindings, services, not management-specific**
- **Disadvantages: resource-expensive for large object populations, sub-optimal information retrieval**
 - **Default: 1 method per object attribute**

Which Features Are Really Necessary?

- **First of all, a reminder that we are talking mainly about information retrieval**
- **CMIS/P scoped Get and SNMP GetNext/GetBulk are mostly used for table retrieval**
 - Plain Get is adequate for 99% of all other cases
- **Modelling table entries as separate dynamic objects can be very resource expensive**
 - For example, in CORBA this is prohibitive for $O(10^5)$ object populations even with the Portable Object Adapter (POA)
- **The key problems to address in DOTs are:**
 - Multiple attribute retrieval per interface in one go
 - Table modelling and retrieval

Methods for Multiple Attribute Retrieval (Strong Typing)

- **Proposal for non-table objects: *semantic* attribute grouping per interface with access methods**
 - Typically a method for static attributes (i.e. properties)
 - Methods grouping dynamic counters (including also time)

- **Example: TCP protocol machine**

```

TcpStaticAttrs    getStaticAttrs();    // RtoAlg, RtoMin, RtoMax
TcpConnCounters   getConnCounters();    // Time, ActiveOpens,
                                           // PassiveOpens, ...
TcpSegmCounters    getSegmCounters();    // Time, InSegs, OutSegs,
                                           // RetransSegs
TcpCounters        getAllCounters();    // Connection + Segment
    
```

- **Semantic grouping based on usage requirements**
- **Also possibly available individual attribute access methods and a `getAllAttrs()` method**
- **Strong typing approach with method signatures**

- **Every interface inherits from a generic MO interface (similar to the *top* GDMO class)**
 - **Keeps names/values of derived class attributes (values as Any type)**
 - **Supports get(attrNameList) method for arbitrary combination of attributes (like in CMIS/P and SNMP)**
- **This approach is more complex and arguably against the strong-typing nature of DOTs**

- One possibility is to model table entries as separate interfaces, with the containing object returning interface references to them
 - TCP example: `IntRefList getConnRefs();`
 - Too many fine-grain distributed objects => does not scale
- Proposal: model tables through a “list of records” structure accessible through a containing object method
 - Methods to retrieve the number of entries, retrieve the whole table and add/remove an entry
 - Similar to GDMO set- or sequence-valued attributes
 - TCP example: `long getConnNo(); TcpConnList getConnInfo();`
 - Retrieval method may return a large amount of information, we rely on CO reliable transport protocol e.g. CORBA IIOP, WS SOAP/HTTP/TCP

- **Simple approach that supports bulk retrieval for tables**
- **Only static objects have interfaces advertised through the naming service**
 - Relatively few objects per node => scalability
- **Attribute grouping for retrieval based on usage requirements**
- **Similar grouping required for configuration settings**