General Inter-ORB Protocol (GIOP) and derivatives

Andrew Watson
andrew@omg.org
Background - CORBA

- Open, non-proprietary distributed object infrastructure spec.
  - Developed and maintained by Object Management Group
  - Open, non-profit member-controlled industrial consortium
  - 40+ commercial and OSS implementations
  - Long-established interop with proprietary designs (DCOM)

- Minimising core functions allows small implementations
  - e.g. client-only CORBA + simple app on Palm Pilot in 45K
    (See: http://www.mico.org/pilot/index.html)

- Additional functions (e.g. transactional RPC, confidentiality, non-repudiation) provided by optional Object Services
Application Portability

- Supports writing clients & servers in C, C++, Smalltalk, Java, Ada, COBOL, Common Lisp, Python
  - Not (yet?) standard bindings for Eiffel, Perl, PL/1, Dylan ...
- Interfaces specified in Interface Definition Language (IDL)
  - IDL compiler generates custom API in selected language
  - Fixed API, and IDL generation rules set by CORBA spec
  - IDL provides language independence & interoperability
  - Simple to learn & use (IDL-generated stubs do marshalling)
Interoperability

- Layered specification

- Hinges on Interoperable Object Reference (IOR) information model holding addressing information for multiple protocols
  - Type ID + one or more opaque address profiles
  - Client may invoke object via any supplied address profile
General Inter-ORB Protocol (GIOP)

- Designed to be simple, scalable and easy to implement
  - Every ORB must support GIOP mapped onto local transport
- Can use almost any connection-oriented bytestream transport
  - Makes TCP/IP-like assumptions about transport’s properties
- Based on seven simple messages
- Common Data Representation (CDR) encoding of data types
  - Primitives (Ints, Floats, Octets, Booleans, Characters)
  - Constructed types (Struct, Union, Array, Sequence, String, Enum, Typecode, Any, Principal, Context, Exception, IOR)
Internet Inter-ORB Protocol (IIOP)

- Mapping of GIOP onto TCP/IP
  - i.e. constructing IOR profile for TCP addressing
  - Mandatory for all ORBs on IP networks

- Simple to implement, works well over WANs
  - See Pilot client example - 45K, calls server in Berkeley

- Optimal for high bandwidth/CPU-speed ratio
  - All quantities aligned on own-size-multiple boundaries

- Many firewalls block IIOP assigned port by default
  - Social, not technical, issue
Evaluation: GIOP-over-port-80

- How to get round firewall problem? Hijack a well-known port!
  - Port 80 the obvious choice
  - Unprincipled but effective

- Implementation choices seem to be:
  - Run IIOP directly over port 80 - what many ORBs do already
  - Encapsulate IIOP in HTTP POSTs - fool more firewalls
  - Replace GIOP’s CDR encoding with XML, run over HTTP - more verbose, more CPU overhead, but ASCII debugging

- CORBA’s multi-protocol support would allow all three approaches to be used in same ORB (so client can choose)
Conclusion

• IIOP provides scalable interoperability within many enterprises

• GIOP over HTTP transport a way through almost any firewall
  - Transparent to existing services (transactions, security etc)
  - Ensures application portability via current APIs & IDL
  - Easy to bridge to IIOP and other GIOP-based protocols (including IIOP-based Java RMI)
  - Can use existing bridge to proprietary platforms (e.g. DCOM)
  - Minimises new code (re-uses most of layered ORB interop)

• Using XML encoding inefficient & verbose, but gets attention
  - So OMG may run an RFP to adopt technology in this area
These slides are available at: