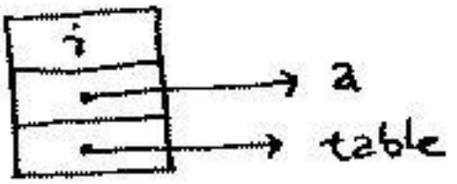
Marshalling & De-Marshalling

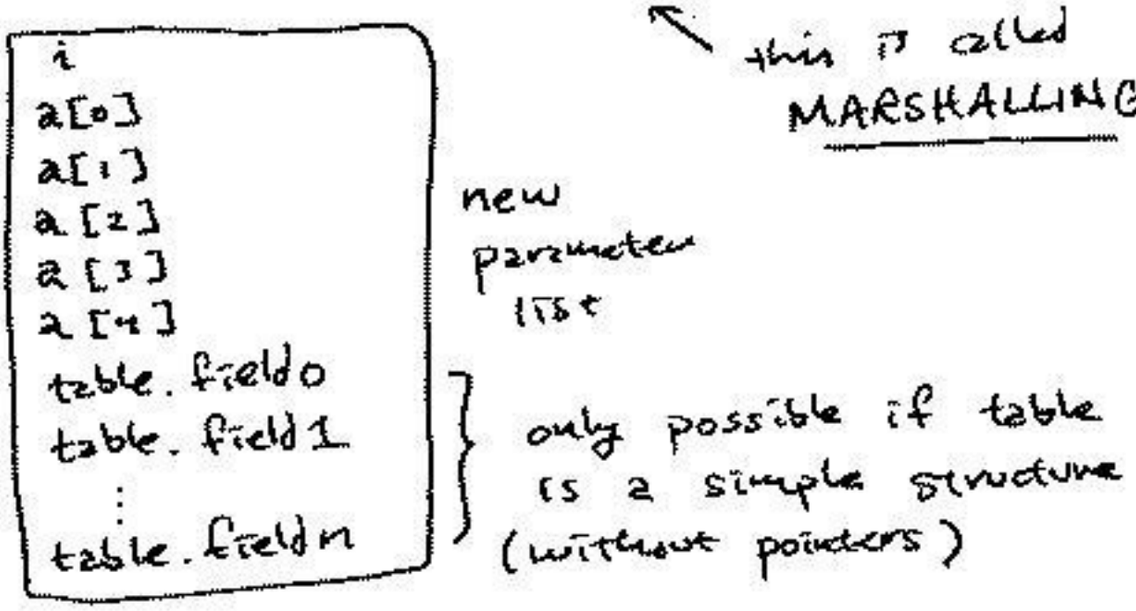
① On Host A, take given parameter list:

func (i, a, p)



convert to a list without pointers

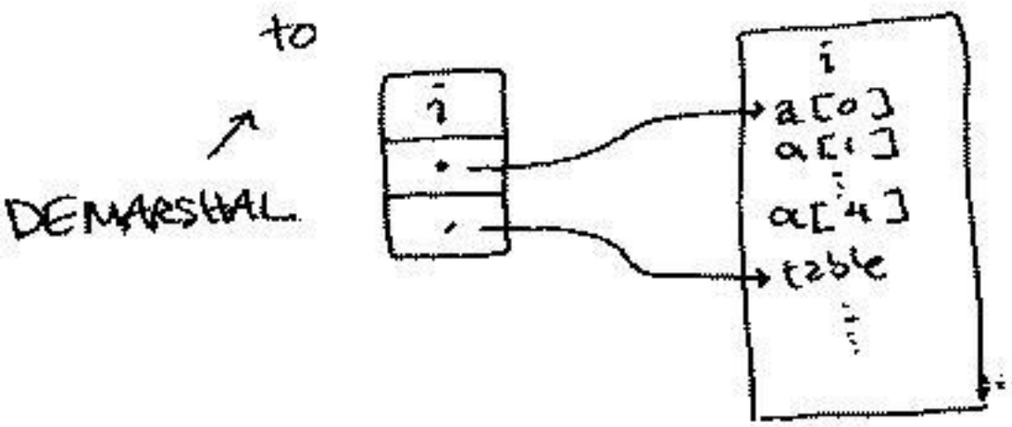
↖ this is called MARSHALLING



② Send converted parameter list along with name of function to be called (func 1) to Host B, using TCP or UDP

③ On Host B, convert received parameter list (actually, we construct a new one) to

↗ DEMARSHAL



④ call func(i, a, p) on Host B

Note: this call might change array a and table p

⑤ "Marshall" the (possibly modified) parameter list — not much to do here (may need only to store return value of func() call)

Put all of this into message and send to Host A

⑥ At Host A, copy values from returned parameter list back to array a and p → table

↗ DEMARSHALL

Some remaining problems:

what if Host A and Host B have different representations of integer, char, float ?

— may need conversion

what if a function parameter is an object reference, and the object is complex (contains pointers, other objects)

— we can't deal with this

NOT ALLOWED !

How to setup all of that conversion software ?

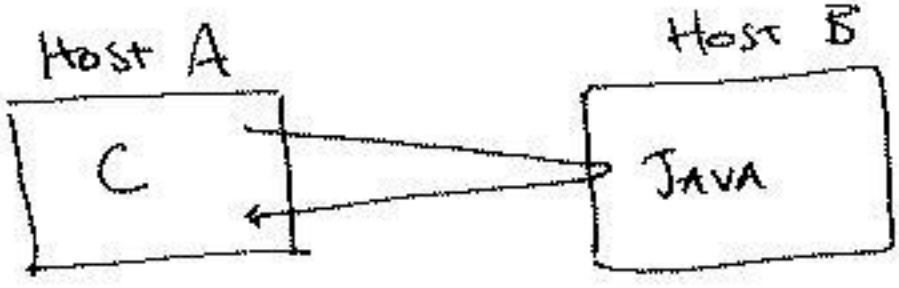
— IDL's, Stubs,

Stub generation

Interface Description Language (IDL)

A formalism to describe the calling parameters, return value, and RPC semantics of a function / method in a programming language-independent way.

GOAL: describe func(i, o, p) so it could be implemented in C, Java, Python, whatever.



IDL is <u>not</u> a full programming language

STUBs.

A stub is a substitute for a function / method. The stub does:

- Marshalling / Demarshalling
- Conversion of types
- Message / send - receive

For RPC, there is a stub for the caller (client) and another stub for the invoked routine (server)

IDL Compiler

Input: IDL description of func(i,a,p)

Output: stub for client,
stub for server.

Putting it all together.

User follows these steps

(1) write function func(i, a, p)

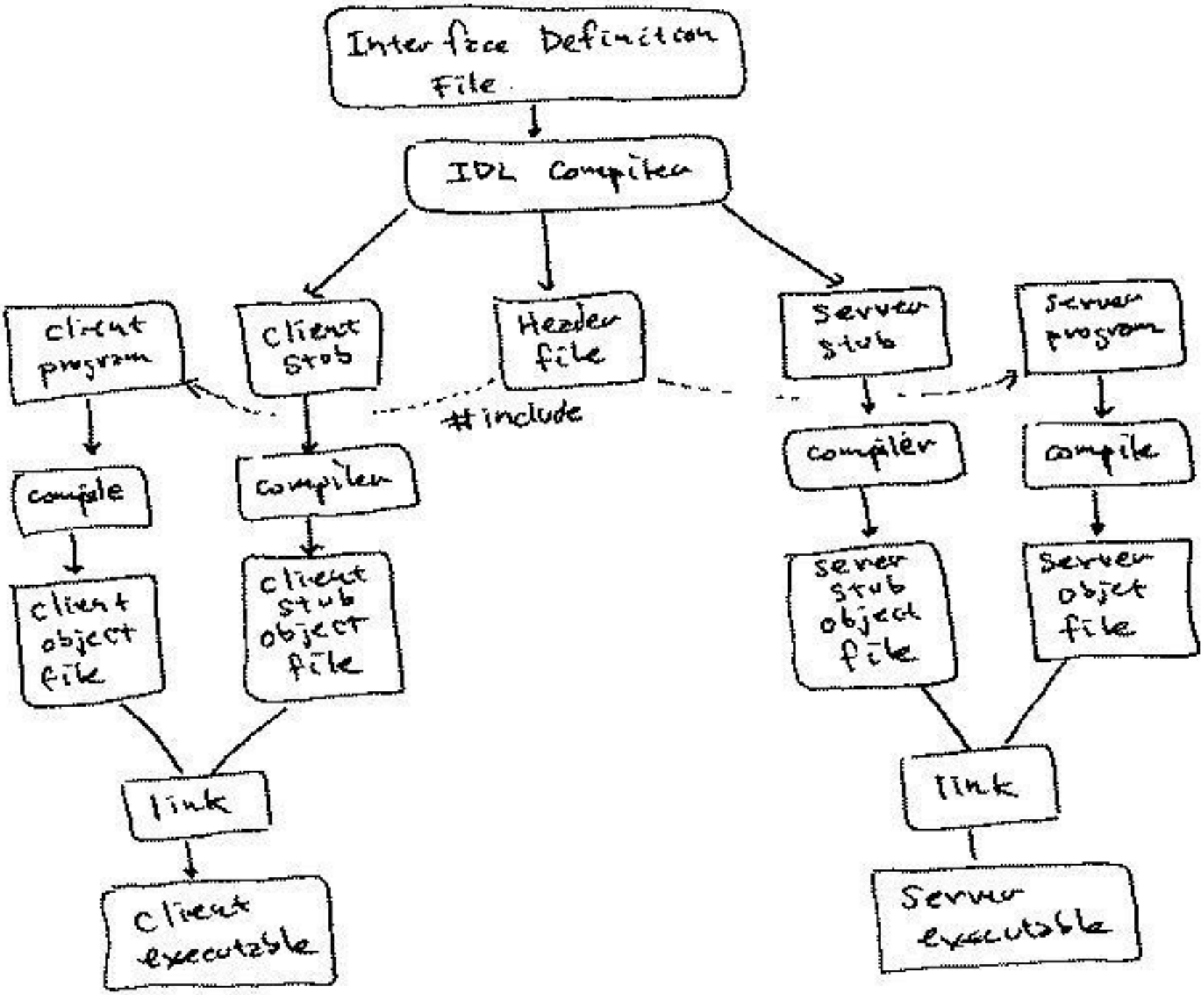
(2) write IDL for func(i, a, p)

(3) compile func, compile func IDL

(4) put stubs into libraries

(5) "register" the RPC if needed, with RPC directory

(6) link client and server programs



Not shown, all the support libraries for messaging, etc.

Beyond DCE RPC

- Force users to code all parameters in XML —

  avoids all the issues of conversion, objects, pointers

- Define "network object references" so that each object has a globally unique reference name

  this would allow the use of references in parameter lists

- Special, language-specific RPC implementations — example, Java's RMI.

  — uses synchronized and Serializable attributes