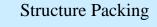


Endianness

- in memory, how are 32-bit words laid out?
- given 0x12345678 0x9abcdef0
- big-endian: (SPARC, 68k, PPC, Internet protocols)
- 0x12 0x34 0x56 0x78 0x9a 0xbc 0xde 0xf0
 little-endian: (x86)
- 0x78 0x56 0x34 0x12 0xf0 0xde 0xbc 0x9a
- when you're on the same computer, it doesn't matter
- when you send data between computers, it does

8b_marshalling: CSci551 SP2006 © John Heidemann



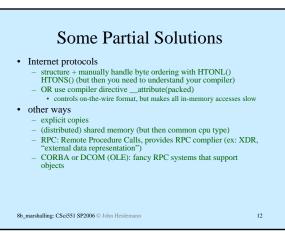
- how big is the structure:
 - struct s { char a; int b; char c; };
- answers:
 - 6 bytes: 1 + 4 + 1 (typical); 1 + 2 + 1?
 - maybe 12 (assuming 4-byte alignment) or maybe 9 = 1+3+4+1, or maybe 6
- why?
 - depends on processor and the compiler and maybe compiler options
 - most processors require alignment

8b_marshalling: CSci551 SP2006 © John Heidemann

Why care?

- what happens?
 - struct s; write(socket_fd, &s, sizeof(s)); (could output 9 or 12 bytes)
- it means you can't assume a structure is the same as a packet format
 - must understand the host computer
 - and the particular compiler!
- alternative to just writing structures?
 - to guarantee layout on the wire, maybe
 - explicitly copy each field into a memory buffer

8b_marshalling: CSci551 SP2006 © John Heidemann



Other Data Transmission Issues

- (endianness, structure packing)
- RPC systems might have procedures and even objects
- arrays? lists? pointers? trees?
 think about how you would marshal these things
 - ex: for a tree: maybe put the data in an array, and then make the pointers be array indicies

15

8b_marshalling: CSci551 SP2006 © John Heidemann

Where have you seen these issues?

17

- project a
- homework 2
- headers of protocols like TCP

8b_marshalling: CSci551 SP2006 © John Heidemann