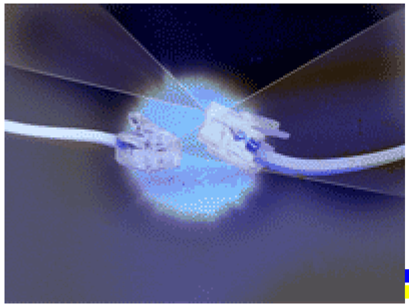
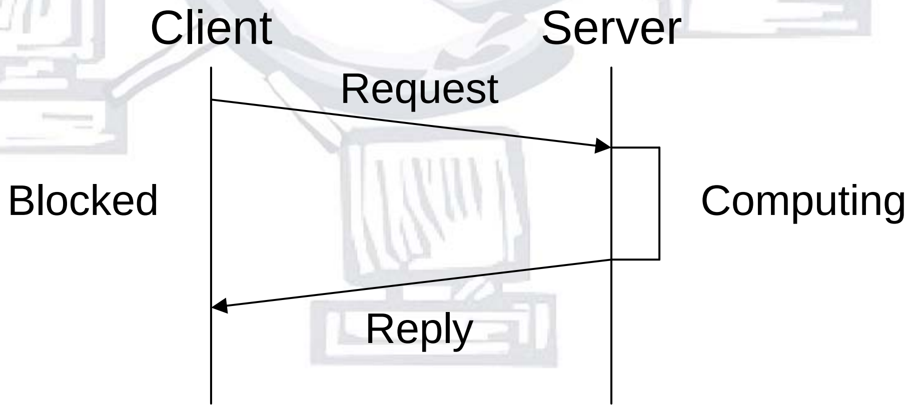
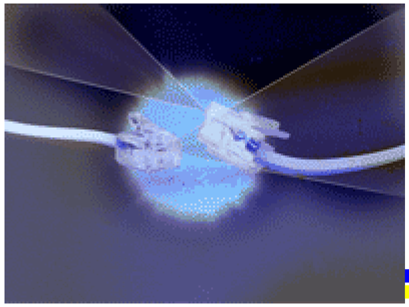


Remote Procedure Calls (RPC)

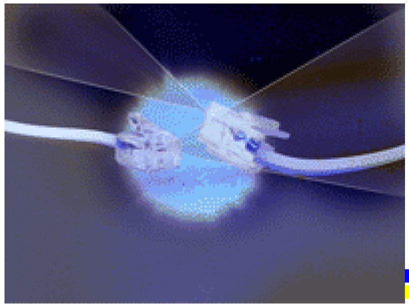
- Technique allowing an application to invoke a procedure whose code actually executes on another host
- Takes the form of a request / reply message exchange





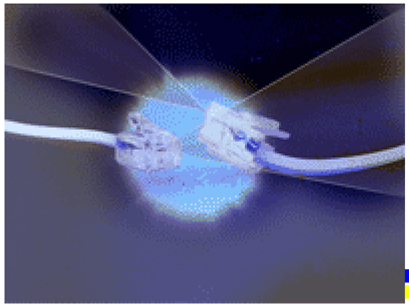
RPC (cont.)

- Goal is to make process of calling a remote procedure indistinguishable from calling a local procedure to the client app
- Difficulties
  - Network interaction introduces plenty of new complications that local call doesn't have
  - Client and server hosts might have different architectures and different representations of data (i.e. big vs. little endian, 32-bit vs. 64-bit)



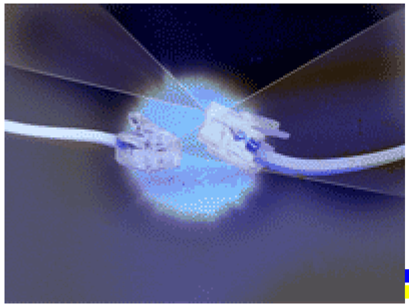
RPC (cont.)

- First problem solved by creating messaging protocol to mask network issues
- Second problem solved by creating support to package arguments into requests and unpackage return values from responses
  - Machine specific
  - Maybe language specific
  - Called *marshalling / demarshalling*
  - Done by *stub compiler*



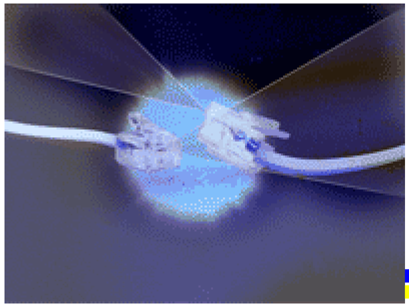
RPC Network Protocol

- Could run on top of TCP or UDP
  - TCP connection setup/teardown fairly wasteful to exchange a request and reply
  - UDP leaves several problems that protocol must address – mostly reliability
- Might consider this as an alternative transport protocol, since it is working at the app-to-app level



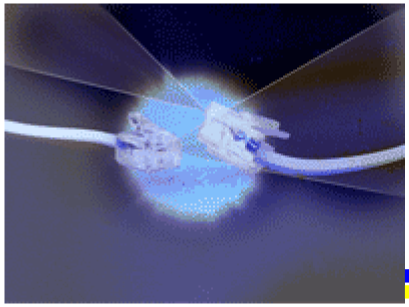
RPC Network Protocol (cont.)

- Book introduces three-level *microprotocol* stack to accomplish different tasks
  - – BLAST – handle message fragmentation
    - More efficient that IP
  - – CHAN – synchronize messages
  - – SELECT – dispatch requests to processes
- <u>Not</u> standard protocols



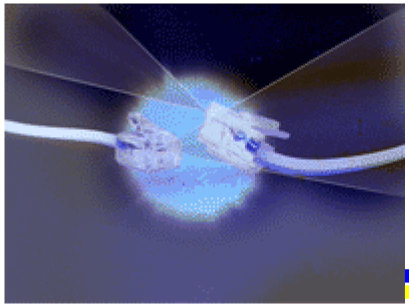
BLAST

- Function similarly to IP fragmentation
- Allows sender to acknowledge multiple fragments
  - Everything after a hole doesn’t need to be retransmitted
- More aggressive in guaranteeing that all fragments are delivered
- Not completely reliable



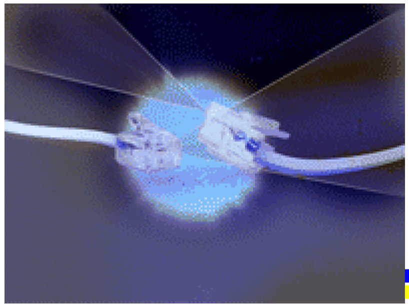
CHAN

- Adds reliability
  - – Guarantees message delivery
  - – Ensures only one copy of message delivered
  - – Allows synchronization of client and server
- Implements *at-most-once* semantics
  - – Message might not get through at all
  - – If it does, it won't be delivered more than once
  - – Essential for many remote procedures



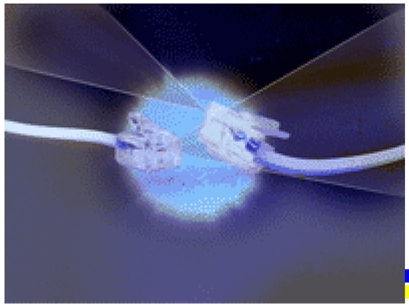
CHAN (cont.)

- Synchronous protocol – client blocks while server is working, so only one call outstanding
- Provides multiple channels
  - Possible to work around previous restriction by using parallel channels from one client to one server

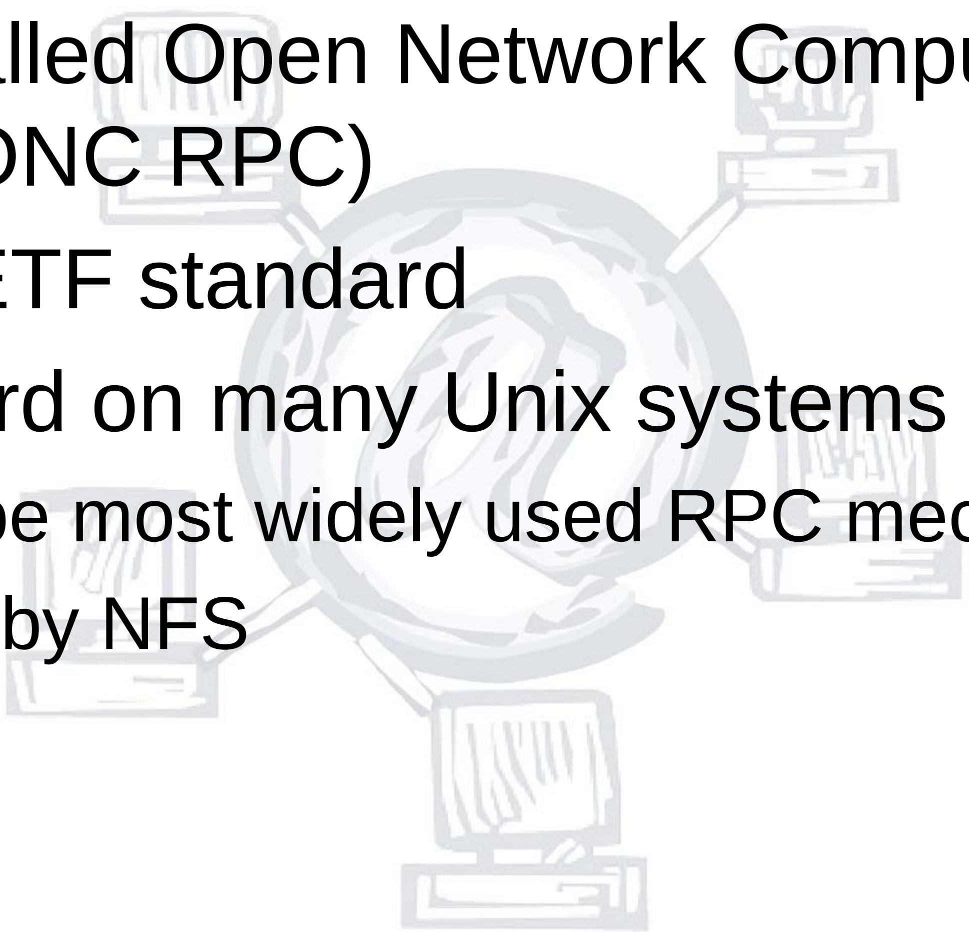


SELECT

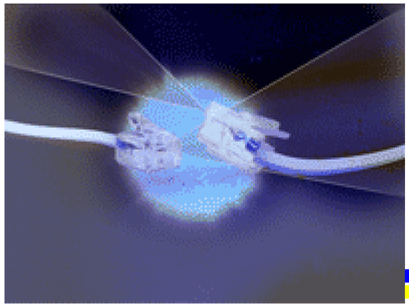
- Dispatcher that directs data from a message to correct procedure
- Provides mechanism to identify application and procedure to call in that application
- Manages multiple channels as necessary for parallel calls to server



Real Implementation - SunRPC

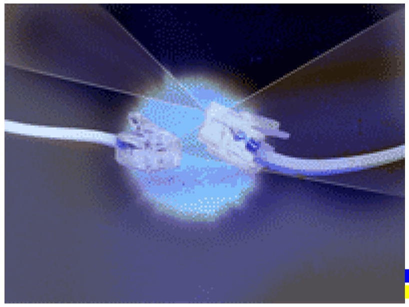


- Now called Open Network Computing RPC (ONC RPC)
- Draft IETF standard
- Standard on many Unix systems
  - May be most widely used RPC mechanism
  - Used by NFS



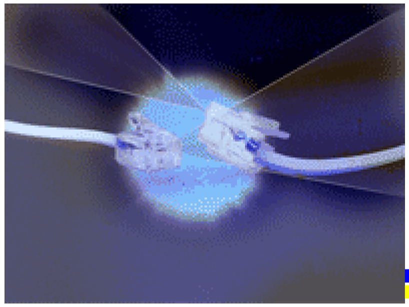
SunRPC (cont.)

- Implemented on top of UDP
  - Implements CHAN
  - IP used for BLAST (not as aggressive or efficient)
  - UDP provides dispatch to correct program, SunRPC selects correct procedure



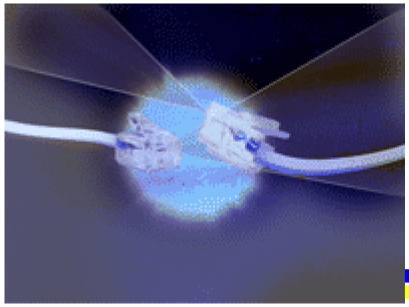
SunRPC (cont.)

- Identifies program and procedure using two 32-bit numbers (program & procedure)
- Uses **PortMapper** to map 32-bit program number to UDP port
  - Runs on well known UDP port (111)
- Client caches port number
  - Avoids calling PortMapper for every procedure invocation



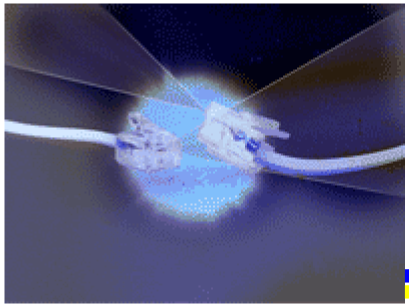
SunRPC (cont.)

- Does not guarantee *at-most-once* semantics
  - Possible for request to be delivered to server twice for some rare network conditions
  - Not addressed because protocol originally designed for use on a LAN, not an internet

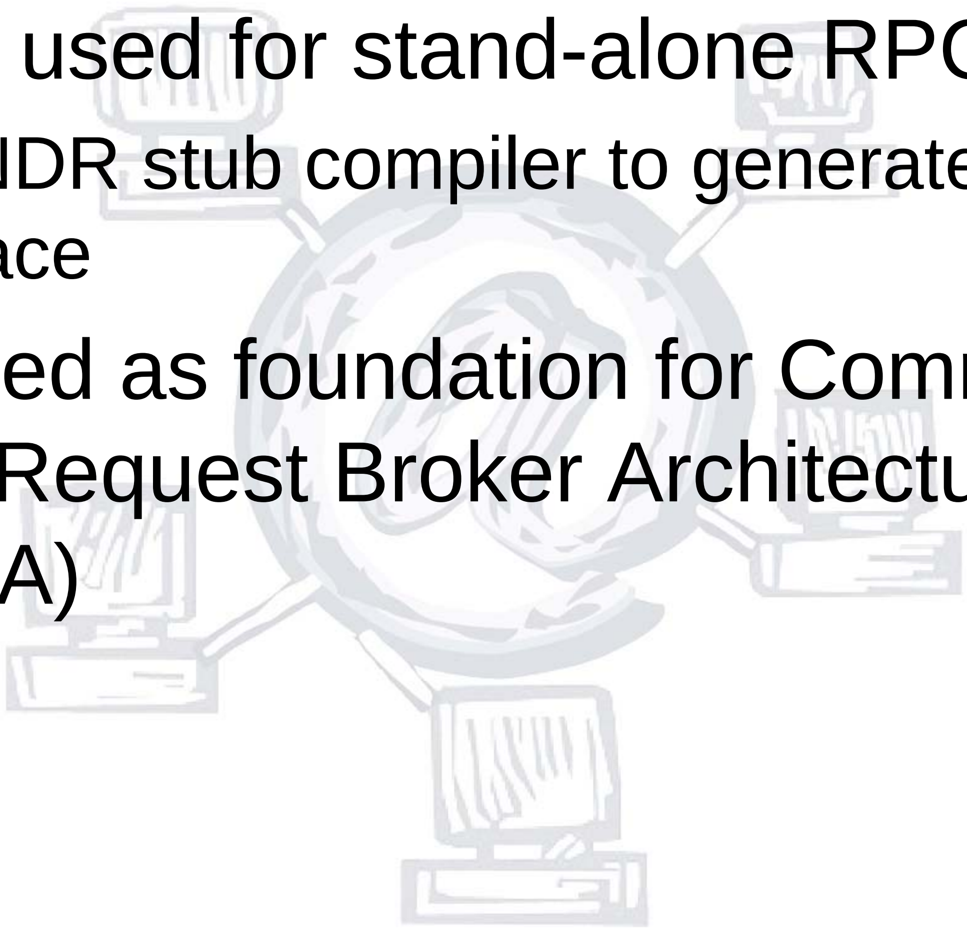


eXternal Data Representation (XDR)

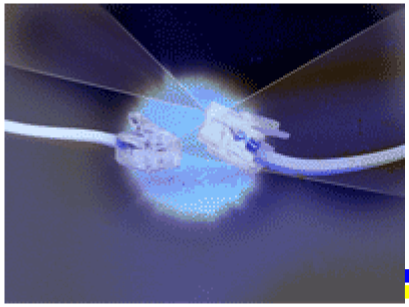
- Accompanying specification for mapping host data to message
  - – Host architecture independent
  - – Network independent
- Described in RFC 1014
- Presentation layer protocol
  - – Can be used independently of SunRPC



OSF Distributed Computing Environment (DCE)



- Can be used for stand-alone RPC
  - Use NDR stub compiler to generate language interface
- Also used as foundation for Common Object Request Broker Architecture (CORBA)



DCE (cont.)

- Also implemented on UDP
- Also uses "endpoint mapping service" to select correct UDP port
- Implements fragmentation (like BLAST)
- Implements *at-most-once* semantics
  - Can also support *zero-or-more* semantics