PROTOCOLARCHITECTURE

Author

A.Swetha

Email: andukuryswetha@gmail.com

Abstract: In the current generation we use protocol architectures such as TCP/IP and ISO suite. This seems to meet the demands of today's networks. However a number of new requirements have been proposed for the networks of today and some innovations in the networking are necessary. In this paper we discuss about the protocol its basic functions and the application for the protocol architecture the current generation architecture that are used in network as TCP/IP, and also an application about the architecture for wireless mono sensor and two new design protocols for this protocol architecture.

1. INTRODUCTION

The basic function of protocol architecture is in the concept of internetworking. An Internetwork is a interconnection of networks. An internetwork is sometimes referred to as a 'network of networks' because it is made up of a large number of small networks. The nodes that interconnect networks are called as 'routers'. The services and protocols are two different concepts. A SERVICE is a set of primitives (operations) that a layer provides to a layer above it. The service defines what operations the layer is prepared to perform on behalf of its users, but it says nothing at all about how these operations are implemented. A PROTOCOL, in contrast, is a set of rules governing the format and meaning of the packets, or messages that are exchanged by the peer entities within a layer. Entities use protocols to implement their service definitions. They are free to change their protocols at will, provided they do not change the service visible to their users.

2. PROTOCOL FUNCTIONS

The core function of protocols is to transfer application information among machines. Thus, an obvious way to review the function of protocols is to focus on the data transfer phase of protocol operation and to observe and catalog the operations that occur there. This leaves for separate consideration those operations such as session initiation, service location, and so on, which, while very important; do not occur at the same time as data transfer.

A distinction which seems to have great relevance is the separation of data transfer functions into two groups: those which actually read or modify the data, which we will call data manipulation and those functions that regulate the transfer, which we will call, transfer control. Despite the great complexity of many protocol suites, there are in fact not that many manipulation and transfer control functions that actually occur at the time of data transfer. Based on existing practice, the most important and universal of these functions are the following.

2.1 Data Manipulation:

There are six manipulation functions which are generally found in protocols. Note that while we list them as separate, several of them may be accomplished in the same operation.

Moving to/from the net: The most obvious and unavoidable manipulation function is the actual transfer of the data in or out of the network itself, which usually involves some sort of serial-to-parallel transformation. This function is usually performed in custom hardware which decouple the timing of the network from that of the host.

Error detection: Data is usually protected by some sort of checksum or related function, which is computed by reading the data. When an error is detected, there may be an error correction stage that also involves a data manipulation.

Buffering for retransmission: At the sending end of a transfer, many protocols reserve a copy of the data, so that if it is lost in transit it can be resent.

Encryption: either for privacy or integrity, data is encrypted. This function, if implemented, can sometimes also provide error detection.

Moving to/from application address space: most commonly, data is not moved directly between user address space and network interface, but is moved to some intermediate buffer in system address space. This may be a fundamental requirement on input, as discussed below, but is often dictated by the details of the system I/O structure.

Presentation formatting: most data transfers require that the data be reformatted into some common or external data representation.

Presentation formatting, usually involves moving the data into a new area, since the result of formatting may, in general, be of a different size than the original.

2.2 Transfer Control:

There are a number of operations that are directly related to the detailed control of the data transfer phase. We will focus on these “transfer control” operations and set aside the remaining control functions, such as connection initiation.

The most common transfer controls are the following:

Flow/Congestion control: To protect both the network and the receiver, the sender must be regulated to send no faster than the data can be accommodated. The minimal in band control function involves the pacing of the data at the transmitter and the monitoring of arrivals at the receiver. The actual computation and negotiation of the transfer rate can be performed on an out-of-band basis.

Detecting network transmission problems: Networks, especially packet switched networks, have specific failure modes. Data may be lost due to congestion overflow, and it may be reordered or duplicated as a part of processing. It is sometimes convenient to think of lost packets as a different problem from reordered packets, but we will have more to say on this later.

Acknowledgement: A common control function is positive acknowledgement of data receipt. This control is sometimes thought of as universal. In truth, it is but one of many methods for dealing with network errors. However, it is a step with considerable complexity and thus deserves separate mention.

Multiplexing: Several hosts share a network, and several processes share one host. Thus several data streams may interleave entering or leaving a host. These must be delivered properly, both to insure basic function, and to prevent security problems arising from mis-delivery.

Time stamping: Some real-time protocols rely on packet timestamps to support the regeneration of inter-packet timing.
Framing: Encapsulation-based protocols require that frame boundaries be conveyed between sending and receiving entities.

3. TCP/IP Protocol Architecture:

TCP/IP comprises a large collection of protocols that are Internet standard. There are mainly 5 layers in the TCP/IP model. This is the model this is designed after ISO-OSI (International Standard Organization Open System Interface) reference model. This consists of Application layer, Transport, Internet, Network access, Physical layers.

Application Layer: provides access to the TCP/IP environment for users and also provides distributed information services to the users. This layer uses protocol services such as SMTP, FTP, SSH, HTTP.

Transport Layer: Transfer of data between end points. Many provide error control, flow control, congestion control, reliable delivery. This layer uses protocols such as TCP, UDP.

Internet Layer: Shield higher layers of network from details of physical layer configuration. Provides routing, Many provide QOS, and congestion control. This layer uses protocols as IPv4, IPv6 (ICMP, OSPF, RSVP).

Network Access Layer: this is also known as data link layer. This provides logical interface to network hardware. May be stream of packet oriented. May be it is reliable. The protocols it uses are Ethernet, WI-Fi, Frame relay, ATM.

Physical Layer: Transmission of Bit-stream specifies medium, signal encoding, data rate, bandwidth, and physical connector. Transport layer mainly use Twisted fiber cable, optical cable, Satellite, Microwave

4. Application: Protocol Architecture for Wireless Micro sensor Networks:

Networking together hundreds or thousands of micro sensor nodes allows users to accurately monitor a remote environment by intelligently combining the data from the individual nodes. These networks require robust wireless communication protocols that are energy efficient and provide low latency. We develop and analyze low-energy adaptive clustering hierarchy (LEACH), a protocol architecture for micro sensor networks that combines the ideas of energy-efficient cluster-based routing and media access together with application-specific data aggregation to achieve good performance in terms of system lifetime, latency, and application-perceived quality.
Advances in sensor technology, low-power electronics, and low-power radio frequency (RF) design have enabled the development of small, relatively inexpensive and low-power sensors, called *micro sensors* that can be connected via a wireless network. These wireless micro sensor networks represent a new paradigm for extracting data from the environment and enable the reliable monitoring of a variety of environments for applications that include surveillance, machine failure diagnosis, and chemical/biological detection. An important challenge in the design of these networks is that two key resources—communication bandwidth and energy—are significantly more limited than in a tethered network

environment. These constraints require innovative design techniques to use the available bandwidth and energy efficiently. In order to design good protocols for wireless micro sensor networks, it is important to understand the parameters that are relevant to the sensor applications. While there are many ways in which the properties of a sensor network protocol can be evaluated, we use the following metrics.

A. Ease of Deployment

Sensor networks may contain hundreds or thousands of nodes, and they may need to be deployed in remote or dangerous environments, allowing users to extract information in ways that would not have been possible otherwise. This requires that nodes be able to communicate with each other even in the absence of an established network infrastructure and predefined node locations.

B. System Lifetime

These networks should function for as long as possible. It may be inconvenient or impossible to recharge node batteries. Therefore, all aspects of the node, from the hardware to the protocols, must be designed to be extremely energy efficient.

C. Latency

Data from sensor networks are typically time sensitive, so it is important to receive the data in a timely manner.

D. Quality

The notion of “quality” in a micro sensor network is very different than in traditional wireless data networks. For sensor networks, the end user does not require all the data in the network because 1) the data from neighboring nodes are highly correlated; making the data redundant and 2) the end user cares about a higher-level description of events occurring in the environment being monitored. The quality of the network is, therefore, based on the quality of the aggregate data set, so protocols should be designed to optimize for the unique, application- specific quality of a sensor network.

LEACH employs the following techniques to achieve the design goals stated:

1) Randomized, adaptive, self-configuring cluster formation;

2) Localized control for data transfers;

3) Low-energy media access control (MAC); and

4) Application-specific data processing, such as data aggregation or compression.

Simulation results show that LEACH is able to achieve the desired properties of sensor networks. When designing protocol architectures for wireless micro sensor networks, it is important to consider the function of the application, the need for ease of deployment, and the severe energy constraints of the nodes. These features led us to design LEACH, a protocol architecture where computation is performed locally to reduce the amount of transmitted data, network configuration and operation is done using local control, and media access control (MAC) and routing protocols enable low-energy networking. Results from our experiments show that LEACH provides the high performance needed under the tight constraints of the wireless channel.

5. NEW DESIGN ISSUES

In this paper we discuss two new design principles,

5.1. Application Level Framing and .

5.2.Integrated Layer Processing.

Additionally, it identifies the presentation layer as a key aspect of overall protocol performance.

Application Processing of Mis-ordered or lncorrplete Data:

The manner of coping with data loss is highly dependent on the needs of the application. The classic transport model is that the protocol will suspend delivery of data to the receiving client, and retransmit from a copy of the data saved at the sender. But this is not the only pattern. Another option is for the application to accept less than perfect delivery and continue unchecked. This will work for real-time delivery of video and voice. Another option is for the sending end to retransmit, but for the application rather than the transport protocol to provide the data. This permits the sending application to recompute the lost data values, rather than buffering them. Finally, in some real time situations, the application may not literally retransmit lost data, but it might send some new data which eventually "fixes" the consequences of the original loss. So without understanding the transformation that the presentation layer has done, it is impossible for the application layer to relate lost data at the transport to the equivalent application data.

5.1 Application Level Framing:

The way to avoid this problem is for the lower layers such as presentation and transport to deal with data in units that the application specifies. In other words, the application should break the data into suitable aggregates, and the lower levels should preserve these frame boundaries as they process the data. This proposal will call these aggregates

Application Data Units, or ADUs. ADUs will then take the place of the packet as the unit of manipulation. We call this design principle Application Level Framing. The – fundamental characteristic of our definition is that each ADU can be processed out of order with respect to other ADUs. This rule permits the ADU boundaries to take the place of the packet boundaries for purposes of manipulation functions such as end-to-end error detecting codes or moving into application address space. At the same time, the ADU now becomes the unit of error recovery. Since the ADU is defined to be the smallest unit which the application (or presentation conversion function) can deal with out of order, it follows that if even part of an ADU is lost in transmission, the application will, in general, be unable to deal with it. Since our application layer takes on the responsibility of recovering lost data, it will almost certainly need to assume the whole ADU is lost, even if parts exist. l' Unless the presentation layer can translate the identity of the lost data into terms the application understands, the application cannot understand which of its elements have actually been lost.

This suggests that ADU lengths should be reasonably bounded, so that when data is lost the application need do no more work than necessary. Indeed, since the loss of even one bit will trigger the loss of a whole ADU, excessively large ADUs might prevent useful progress at all, since the probability of any ADU having at least one uncorrected error would approach one. The final characterization of an ADU is thus the following.

1. The sending and receiving application must define what data goes in an ADU such that
2. the sender can compute a name for each ADU that permits the receiver to understand its place in the sequence of ADUs produced by the sender, and
3. The sender uses a transfer syntax that permits the ADU to be processed out of order.

5.2 Integrated Layer Processing:

Layered protocol suites provide isolation between the functional modules of distinct layers. A major architectural benefit of isolation is that it facilitates the implementation of Sub systems whose scope is restricted to a small subset of the suite's layers. For example, the implementation of a network layer relay is largely independent of the upper layer protocols used by its clients. However, we are frequently concerned with the implementation of complete end systems that coincidentally terminate most of the layers of the protocol stack. The naïve implementation of a layered suite involves the sequential processing of each unit of information, as it is passed down through the individual layer entities of the transmitter’s protocol stack, and as it is passed up through the peer entities of the receiver’s stack. Such a simple ordering of operations may well conflict with the efficient engineering of the end systems, and it is sometimes possible to arrange the operations in a more efficient order that achieves the same result. One example of inter-layer optimization is the provision for encryption processing in the system described in . The Auto net protocol suite has been carefully structured to facilitate the implementation of end system interfaces that entwine the session-specific encryption operations with data link level operations. This relaxation of the conventional ordering constraints does not interfere with layer isolation: intermediate nodes perform the traditional data link and network layer operations without regard to the session encryption operation.

If there is no presentation conversion, then state-of-the art implementations are already running into the performance limit imposed by the very simple data transformations of the transport and lower layers - data copying and checksums. In these cases, ILP may be a very practical next step to improve throughput, especially with RISC processors. This will, of course, not be true universally; ILP is just an engineering principle, to be applied only when useful. But the very simple performance experiments we reported above, as well as some of the advanced experiments performed by Van Jacobson on TCP , suggest the utility of this approach.

If there is a presentation conversion being performed, then the cost of this step may well dominate the other manipulation costs. In this case, the application code, which must participate in the presentation conversion, must be the focus of performance tuning. ILP can help here, because the processing steps can be pipelined in a flexible manner.

The key argument in favor of ILP is that an integrated processing loop is more efficient than several separate steps which read the data from memory, possibly convert it, and write it again. This point seems most obviously true for RISC chips, where a major performance limitation is memory cycles. We claim that layered engineering designs should not be thought of as fundamental, but only as one approach, which must be evaluated on the basis of overhead and simplicity against other designs. ILP designs may lead to greater efficiency through careful attention to the ordering of the processing steps and the delineation of the data units.

6.Conclusion

In this paper, we make several structural observations about protocols, and attempt to draw some architectural conclusions. First we have discussed about different architectures such as SMTP, SSH, HTTP etc.., all these protocols works well with TCP/IP reference architecture. We have discussed one of the applications for wireless micro sensor networks. We have discussed some of the important characteristics. After that we have discussed two new design issues in protocol architecture, and attempt to draw some architectural conclusions.

In summary, our reasoning has the following points.

- Data manipulation costs more than transfer control operations.
- Presentation can cost more than all other manipulations combined.
- To implement presentation efficiently, it is necessary to keep . the processing pipeline going, including the application process.
- Application data units are the natural pipelining units.
- They correspond to what applications want, not to the network technology of the day, which can and will change in the near future.
- The key architectural principle we have identified is Application Level Framing.
- The key engineering principle is Integrated Layer Processing, which allows applications to process their data incrementally, and permits efficient implementation of data manipulations on RISC processors.

In this approach to protocol architecture, the different functions are "next to" each other, not "on top of," to the extent possible. The traditional functions of transport and session are control functions, and as such should not have a strict relation to the manipulation functions such as presentation conversion. This is a more parallel organizational model, similar to the HOPS model of Haas.

This discussion of protocol structure should not be taken to mean that layering is unsuited as a tool for protocol design. The principle role of layering is the semantic isolation of functional modules. Although alternative organizational schemes exist, layered isolation has proven to be particularly effective in the network environment.

A potential drawback to the ILP approach is that it could lead to complex designs in which each protocol stack variant has its own fully customized implementation. Clearly this would complicate the maintainability and overall utility of the protocol software. However, we believe that there are approaches to protocol organization that can minimize the liabilities associated with "vertical integration."

In existing protocol suites the semantics of a functional module, i.e., a layer protocol, are closely tied to the syntax of the symbols that are exchanged. Usually there is a one-to-one correspondence between semantic information and specific fields within protocol data units. This arrangement gives rise to the present day scheme of hierarchical encapsulation in which each layer appends its own syntactical symbols to those generated by the layers above.

In many respects this approach corresponds to the "compilation" of the protocol suite, while the encapsulation approach corresponds to its "interpretation". We believe that the principle of Application Layer Framing permits this shared use of fields and structures across layers, and ILP is one speculative example of a "compiled" implementation of a protocol suite.

References:

E.A. Arnould et al. The Design of Nectar: A Network Backplane for Heterogeneous Multicomputers. In ASPLOS-III Proceedings, Third International Conference on Architectural Support for Programming Languages and Operating Systems, pages 205-216, IEEE/ACM.

G. Chesson, B. Eich, V. Schryver, A. Cherenson, and A. Whaley. XTP Protocol Definition. Technical Report Revision 3.0, Silicon Graphics, Inc.,.

D. Clark, V. Jacobson, J. Romkey, and H. Salwen. An Analysis of TCP Processing Overhead. IEEE Communications, 27(6):22-26.

Architectural Considerations for a New Generation of Protocols
David D. Clark and David L. Tennenhouse
Laboratory for Computer Science, M. I. T pages 200-208

An Application-Specific Protocol Architecture for Wireless Microsensor Networks
Wendi B. Heinzelman, *Member, IEEE*, Anantha P. Chandrakasan, *Senior Member, IEEE*, and
Hari Balakrishnan, *Member, IEEE* pages 660-669