International Journal of Information Technology and Knowledge Management
January-June 2011, Volume 4, No. 1, pp. 21-25

OVERALL PERFORMANCE IMPROVEMENT IN WIRELESS SENSOR NETWORK SYSTEM USING END-END ARGUMENTS

Preeti Khera[1], Ashok Kumar[2] & Vibhuti PN Jaiswal[3]

This argument appeals to application requirements, and provides a rationale for moving function upward in a layered system, closer to the application that uses that function. However, low level mechanisms to support these functions are justified only as performance enhancements. This paper reviews the end-to-end arguments, discusses additional arguments for and against end-to-end implementations and provides an algorithm for transfer of packets together with the discussion of some examples like delivery guarantees, security, duplicate message suppression and queuing.

Keywords: End-to-End Arguments, Performance Parameters, Sensor Networks, Simulation, Transmission

1. Introduction

The concept of end-to-end was mentioned briefly, in the course of the consultations on the establishment of a Working Group on Internet Governance (WGIG) held at Geneva from 20-21 September 2004 [1]. It was advisable to add the following words: "or integrated with", after the phrase "high level services layered on". Since the layering approach has many advantages and should be retained along with more integrated system architectures. The paper "End-to-end arguments in system design" [14] (henceforth called "The Paper") has had a profound impact since it was published in 1984. At the same time, the recent success of peer-to-peer networking exemplifies the benefits of end-to-end implementations. The authors of the Paper have themselves revisited the original principle in a modern context, evaluating active networking in terms of end-to-end arguments [9], one author (Reed) has written on how end-to-end arguments remain pertinent today [5], and another (Clark) has written about their role in the context of the changing requirements of the Internet [4]. This paper first describes the primary end-to-end argument and the "careful file transfer" case study, as presented in the Paper. It then considers the performance implications of end- to-end implementations, and describes additional end-to-end arguments such as correct delivery guarantees, secure data transmission, Duplicate message suppression and Guaranteeing FIFO message delivery.

Finally, it considers how to transfer the packets between end users. So, in this paper, system model has been introduced which purposes an overall improvement in the performance of wireless sensor networks using end to end arguments.

[1 2, 3]Department of Electronics and Communication Engineering, ACE, Devasthali, Ambala, India

Email: [1]kherapreeti33@gmail.com, [2]ashokcalicut1993@gmail.com, [3]vibhuti.jaiswal85@gmail.com

2. End-to-End Arguments Parameters

The principle, called the end-to-end argument, suggests that functions placed at low levels of a system may be redundant or of little value when compared with the cost of providing them at that low level. The function in question can completely and correctly be implemented only with the knowledge and help of the application standing at the end points of the communication systems [14]. But the question arises where to implement these functions either at both end or within the network. Though, certain functions such as routing, security, error control etc are implemented in both end systems and the network. While the paper presents multiple end-to-end arguments but it emphasis on basic algorithm highligtening important steps for delivery of "careful File Transfer" [13] and "providing efficient & reliable communication in wireless sensor networks"- a challenging problem [2]. The data communication system, range includes encryption, duplicate message detection and suppression, message sequencing, guaranteed message delivery, detecting host crashes, and delivery receipts. In a broader context the argument seems to apply to many other functions of a computer operating system, including its file system [14].

3. Careful File Transfer

Consider the problem of "careful file transfer." A file is stored by a file system, in the disk storage of computer A. Computer A is linked by a data communication network with computer B, which also has a file system and a disk store. The object is to move the file from computer A's storage to computer B's storage without damage, in the face of knowledge that failures can occur at various points along the way. The application program in this case is the file transfer program, part of which runs at host A and part at host B. In order to discuss the possible threats to the file's integrity in this transaction, let us assume that the following specific steps are involved [14].

1. At host A the file transfer program calls upon the file system to read the file from the disk, where it resides on several tracks, and the file system passes it to the file transfer program in fixed-size blocks chosen to be disk-format independent.
2. Also at host A the file transfer program asks the data communication system to transmit the file using some communication protocol that involves splitting the data into packets. The packet size is typically different from the file block size and the disk track size.
3. The data communication network moves the packets from computer A to computer B.
4. At host B a data communication program removes the packets from the data communication protocol and hands the contained data on to a second part of the file transfer application, the part that operates within host B.
5. At host B, the file transfer program asks the file system to write the received data on the disk of host B.

With this model of the steps involved, the following are some of the threats to the transaction that a careful designer might be concerned about [14]:

1. The file, though originally written correctly onto the disk at host A, if read now may contain incorrect data, perhaps because of hardware faults in the disk storage system.
2. The software of the file system, the file transfer program, or the data communication system might make a mistake in buffering and copying the data of the file, either at host A or host B.
3. The hardware processor or its local memory might have a transient error while doing the buffering and copying, either at host A or host B.
4. The communication system might drop or change the bits in a packet, or lose a packet or deliver a packet more than once.
5. Either of the hosts may crash part way through the transaction after performing an unknown amount (perhaps all) of the transaction.

The file may become corrupted at various points on the end-to-end path, e.g. on the communication channel (1), in intermediaries such as the router (2), or during disk access (3). For examples of the causes of errors, for discussion of link error control text such as Lin and Costello [15], and to Stone and Partridge [11] for a discussion of router and end-system errors. The authors argue that only a check made at the endpoints (i.e. from information stored on the disks) can "completely and correctly" ensure that no error has been introduced

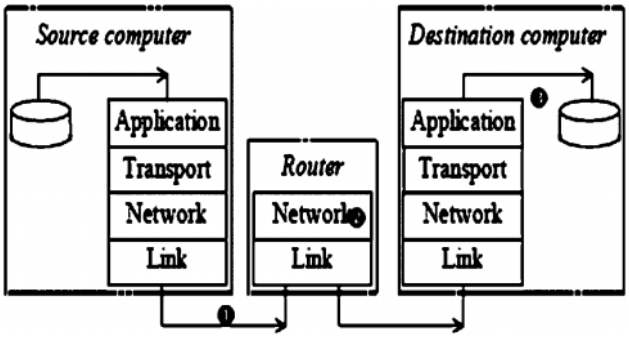

Fig. 1: Errors can Occur at Different Points (1, 2, and 3) on the End-to-end Path [13]

How would a careful file transfer application then cope with this list of threats? One approach might be to reinforce each of the steps along the way using duplicate copies, timeout and retry, carefully located redundancy for error detection, crash recovery, etc. The goal would be to reduce the probability of each of the individual threats to an acceptably small value. The alternate approach might be called "end-to-end check and retry". Suppose that as an aid to coping with threat number one, stored with each file is a checksum that has sufficient redundancy to reduce the chance of an undetected error in the file to an acceptably negligible value [14].

4. PERFORMANCE

The simple strategy outlined above, transmitting the file and then checking to see that the file arrived correctly, would perform more poorly as the length of the file increases. The probability that all packets of a file arrive correctly decreases exponentially with the file length, and thus the expected time to transmit the file grows exponentially with file length. Clearly, some effort at the lower levels to improve network reliability can have a significant effect on application performance. But the key idea here is that the lower levels need not provide "perfect" reliability. The "proper" tradeoff requires careful thought; for example one might start by designing the communication system to provide just the reliability that comes with little cost and engineering effort, and then evaluate the residual error level to insure that it is consistent with an acceptable retry frequency at the file transfer level. Performing a function at a low level may be more efficient, if the function can be performed with a minimum perturbation of the machinery already included in the low-level subsystem, but just the opposite situation can occur – that is, performing the function at the lower level may cost more for two reasons [14]. First, since the lower level subsystem is common to many applications, those applications that do not need the function will pay for

it anyway. Second, the low-level subsystem may not have as much information as the higher levels, so it cannot do the job as efficiently. The end-to-end argument does not tell us where to put the early checks, since either layer can do this performance-enhancement job.

Placing the early retry protocol in the file transfer application simplifies the communication system, but may increase overall cost, since the communication system is shared by other applications and each application must now provide its own reliability enhancement. Placing the early retry protocol in the communication system may be more efficient, since it may be performed inside the network on a hop-by-hop basis, reducing the delay involved in correcting a failure. The principle performance benefit of end-to-end implemen-tations is that such implementations tend to reduce the amount of processing required in the network, allowing the network to operate at higher speed when processing is the bottleneck (as is currently common with optical transmission technology). Second performance benefits of simple networks (which follow end-to-end arguments) are that they are easier to design and change, and this short design turnaround time allows them to track improvements in implementation technologies. Finally, end-to-end functions need only be encountered once (at the endpoints), whereas localized functions may be encountered multiple times, e.g. once for each hop that the traffic takes through the network. This repeated processing can also degrade performance.

A. Correct Delivery Guarantees

Data communication network can easily return an acknowledgement to the sender for every message delivered to a recipient. Although this acknowledgement may be useful within the network as a form of congestion control (originally the ARPANET [14] refused to accept another message to the same target until the previous RFNM had returned) it was never found to be very helpful to applications using the ARPANET. Another strategy for obtaining immediate acknowledgements is to make the target host sophisticated enough that when it accepts delivery of a message it also accepts responsibility for guaranteeing that the message is acted upon by the target application. This approach can eliminate the need for an end-to-end acknowledgement in some, but not all applications. An end-to-end acknowledgement is still required for applications in which the action requested of the target host should be done only if similar actions requested of other hosts are successful. This kind of application requires a two-phase commit protocol [5,10,15] which is a sophisticated end-to-end acknowledgement. According to the end-to-end arguments, applications, not transport layers, should check integrity [13].

B. Secure Data Transmission

Another area in which an end-to-end argument can be applied is that of data encryption. The argument here is threefold [14]. First, if the data transmission system performs encryption and decryption, it must be trusted to manage securely the required encryption keys. Second, the data will be in the clear and thus vulnerable as it passes into the target node and is fanned out to the target application. Third, the authenticity of the message must still be checked by the application. This network-level encryption can be quite unsophisticated – the same key can be used by all hosts, with frequent changes of the key. No per-user keys complicate the key management problem. The use of encryption for application level authentication and protection is complementary. Neither mechanism can satisfy both requirements completely.

C. Duplicate Message Suppression

A more sophisticated argument can be applied to duplicate message suppression. A property of some communication network designs is that a message or a part of a message may be delivered twice, typically as a result of time-out-triggered failure detection and retry mechanisms [14] operating within the network. The network can provide the function of watching for and suppressing any such duplicate messages or it can simply deliver them. One might expect that an application would find it very troublesome to cope with a network that may deliver the same message twice; indeed it is troublesome. Unfortunately, even if the network suppresses duplicates, the application itself may accidentally originate duplicate requests, in its own failure/retry procedures. These application level duplications look like different messages to the communication system, so it cannot suppress them; suppression must be accomplished by the application itself with knowledge of how to detect its own duplicates. If the application level has to have a duplicate-suppressing mechanism anyway, that mechanism can also suppress any duplicates generated inside the communication network, so the function can be omitted from that lower level. The same basic reasoning applies to completely omitted messages as well as to duplicated ones. Accessing data at a repository is done by sending it a message specifying the object to be accessed, the version, and type of access (read/write), plus a value to be written if the access is a write. The underlying message communication system does not suppress duplicate messages, since a) the object identifier plus the version information suffices to detect duplicate writes, and b) the effect of a duplicate read request message is only to generate a duplicate response, which is easily discarded by the originator. Consequently, the low-level message communication protocol is significantly simplified.

D. Guaranteeing FIFO Message Delivery

Ensuring that messages arrive at the receiver in the same order they are sent is another function usually assigned to the communication subsystem. The mechanism usually used to achieve such first-in, first-out (FIFO) behavior guarantees FIFO ordering among messages sent on the same virtual circuit. Messages sent along independent virtual circuits, or through intermediate processes outside the communication subsystem may arrive in an order different from the order sent. A distributed application in which one node can originate requests that initiate actions at several sites cannot take advantage of the FIFO ordering property to guarantee that the actions requested occur in the correct order. Instead, an independent mechanism at a higher level than the communication subsystem must control the ordering of actions [14].

5. RELIABLE DATA TRANSMISSION

The Massachusetts Institute of Technology (MIT) µ- AMPS Project has developed application specific architecture with the following assumptions:

a) The Base stations are far from nodes.

b) All nodes are energy constrained.

c) Data correlation as shown in the Fig.2

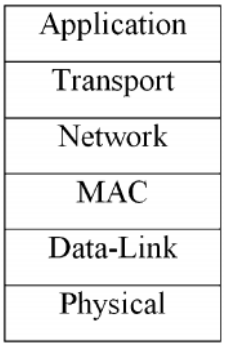

Fig. 2: Application Specific Architecture [3]

which emphasis on application layer to design protocols for high quality, energy efficiency and spectrum efficiency.

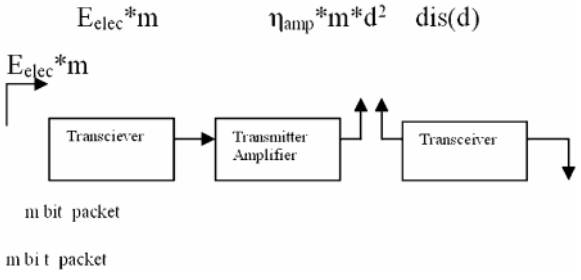

Fig.3: Framework for Simulation

Algorithm

Design for m clusters per round

For each packet do

Repeat

{Sender- Side algorithm}

Receive a packet from application layer

Translate next hop node address of this packet to the next cluster address

Send this packet with read request

{Receiver – Side algorithm

running on multiple nodes}

Receive a packet

Send this packet to the application layer to the selected receiver only.

if packet received correctly

ACK

else

NAK

until

The packet reaches destination

End for

6. CONCLUSION

The End-to-End arguments are responsible for the determination of overall performance of the system. In order to determine if the end-to-end arguments are applicable to a certain service, it is important to consider what entity is responsible for ensuring that service. So, paper concludes "By providing the simulation framework and an algorithm for reliable transfer of packets between end users which can be further modeled using Network Simulation tool".

REFERENCES

[1] Patrice A. Lyons: "The End-End Principle and Definition of Internet", November 10, 2004.

[2] Qing Cao. Tarek Abdelzaher, Tian He and Robin Kravets: "Cluster based Forwarding for Reliable End-to-End Delivery in Wireless Sensor Networks.

[3] W. Heinzeelman, A.Chandrakasan and H.Balakrishnan "An Application Specific Protocol Architecture for Wireless Microsensor Networks", MIT, IEEE Transaction on Wireless Communication,1, No.4, October 2002.

[4] D. Clark and M. S. Blumenthal: "Rethinking the Design of the Internet: The End to End Arguments vs. the Brave New World"; Workshop on The Policy Implications of End-to-End, Dec. 2000.

[5] D. Reed, "The End of the End-to-end Argument", http://www.reed.com/Papers/endofendtoend.html, 2000.

[6] J. Stone and C. Partridge "When the CRC and TCP Checksum Disagree", Proc. SIGCOMM, 2000.

[7] S. Liebowitz and S. Margolis, "Network Externality",The New Palgraves Dictionary of Economics and the Law, MacMillan, 1998.

[8] N. Negroponte: "The Future of Phone Companies", Wired, 4(9), Sep. 1996.

[9] D. Reed, J. Saltzer and D. Clark, "Active Networking and End-to-end Arguments", IEEE Net. Mag., 12(3), 69-71, May/Jun. 1998.

[10] S. Bhattacharjee, K. Calvert and E. Zegura, "Active Networking and the End-to-end Argument", Proc. Int'l Conf. on Network Protocols, 1997.

[11] L. Peterson and B. Davie, Computer Networks: A Systems Approach. Morgan Kaufmann, 1996.

[12] B. Carpenter: "Architectural Principles of the Internet", IETF, RFC 1958, Jun. 1996.

[13] Tim Moors: "A Critical Review of End-to-End Arguments in System Design", Polytechnic University, Brooklyn, NY 11201, USA.

[14] J. Saltzer, D. Reed and D. Clark, "End-to-End Arguments in System Design", ACM Trans. Comp. Sys., 2(4), 277-88, Nov. 1984.

[15] S. Lin and D. Costello, Jr., Error Control Coding: Fundamentals and Applications, Prentice-Hall, 1983.