Complexity Management: A Major Issue for Telecommunications

by
David G. Messerschmitt

Department of Electrical Engineering and Computer Sciences
University of California at Berkeley

Department of Electrical Engineering and Computer Sciences
University of California
Berkeley, California 94720

Proc. International Conference on Communications, Computing, Control, and Signal Processing,
Stanford University, Palo Alto, CA, June 22-26, 1995.

To appear in *Kailath Festschrift*,
A. Paulraj, V. Roychowdhury, C. Schaper, Editors,
Kluwer Academic Publishers , 1996.

Acrobat version (best for printing).

1.0 ABSTRACT

As a result of years of geometrical advances in underlying electronics and photonics technology, traditional efficiency and performance considerations (which have been dominant activities in telecommunications research) will play a somewhat diminished role in the future. Simultaneously we are accumulating multiple standards, protocols, and transmission media, proliferating a variety of user-oriented applications, and seeing cost-effective software implementations and hardware systems with enormous complexity. These trends imply that an increasing barrier to progress in telecommunications is not cost or efficiency, but managing the tremendous complexity of heterogeneous networks, media, terminals, and applications in a multi-vendor environment. More generally, while complexity management has been a traditional issue in software engineering, and later in integrated circuit design, for the future it will be an increasingly important issue in large-

scale system design. Our hypothesis is that complexity management will be an increasing factor in telecommunications research and development. This does not imply that interesting issues in signal processing and communications theory disappear; to the contrary, complexity management considerations raise a number of new issues and will doubtless revitalize these fields. We briefly describe complexity management methodologies that have arisen in the software domain, and speculate on the nature of complexity management in large system design. Is it largely an issue in the management of the development process, or is it amenable to systematic and rigorous approaches? To be more concrete, we give examples from the telecommunications realm drawing on our own work.

2.0 INTRODUCTION

Telecommunications has traditionally been driven by advances in underlying electronics and photonics technologies, and had focused on a small set of universal applications -- namely voice telephony, video conferencing, and data transmission -- with most of the effort devoted to lowering costs and increasing efficiency. The dominant research themes have been largely efficiency-driven, such as the classical problems of increasing the bit rate derived from a transmission medium and minimizing the bit rate required to represent a given source. Our thesis is that this situation is rapidly changing, with complexity management considerations replacing efficiency as the dominant challenge in telecommunications research. We outline arguments in favor of this thesis, describe some general techniques available for complexity management, and give examples from telecommunications.

3.0 THE CHANGING ENVIRONMENT FOR RESEARCH

3.1 Advances in the underlying technology

Advances in electronics technology, roughly doubling in cost-performance every couple years, have greatly contributed to advances in communications efficiency by making sophisticated compression and coding techniques both feasible and affordable. However, in many transmission media, we appear to be approaching fundamental limits, even in the context of affordable implementations, as a result in the accumulated research in compression and coding. Simultaneously, photonics and storage technology have been advancing at the same rate as electronics, or even faster, making efficiency less crucial in backbone networks and storage environments. Indeed, traditional communication theory has made little headway in affecting the practice of fiber optics transmission.

In considering appropriate research themes, we should differentiate between bottlenecks that will disappear with advances in the underlying processing and transmission technologies, as contrasted with more fundamental problems that are not subject to mere technological solution.

Premier examples of the latter include traffic capacity of wireless multiaccess channels, and achieving adequate interactive response in the face of latency (which is lower bounded by propagation delay, already significant in global networks). In contrast, continuing advances in fiber optics, optical networking, electronic switching, and magnetic and optical storage media are providing ample (and increasing) bandwidth resources in backbone networks and storage capacity. There is of course continuing interest in better utilizing existing facilities like wirepair and the voiceband telephone channel -- where efficiency is critical -- but here technology is approaching fundamental limits. Further, we seem poised to finally provide widespread broadband access through some combination of new fiber and existing coaxial facilities.

Advances in electronics technology not only provide ample performance, but they also have two other important impacts:

- The feasible number of devices per chip is increasing, from millions today to hundreds of millions within the next decade. A dominant design problem for such chips is architecture and complexity management; indeed the functional design of such chips is conceptually similar to developing a large software system.
- Software implementation of most functions becomes feasible. Today, audio-frequency functions are typically implemented in software (often on a specialized, but increasingly on a general purpose, processor). In the future, video signal processing will fall in the same category, and customized hardware design will be relegated to radio and microwave frequencies.

Both of these factors imply an increasing convergence between the technical problems faced in the telecommunications and applications software industries. This has long been the case in the signalling, control, and switching aspects of telecommunications as well, where the dominance of software is legendary.

3.2 Advanced applications

Most of the telecommunications applications available today are functionally simple applications with universal interest, like voice telephony and point-to-point video teleconferencing. Indeed, the telecommunications industry perceives itself as selling and provisioning these universal applications, rather than underlying services like audio, video, and data transport[1]. However, as desktop and untethered programmable platforms become standard foundations for networked applications, the cost-effective software realization of functionally complex applications becomes feasible. In addition, we can expect dynamic deployment of software-defined applications over the network to dramatically speed up the rate of innovation in commercially available telecommunications applications; that is, the telecommunications marketplace will begin to look more and more like desktop computing [1]. Most new applications will be functionally complex, often involving multipoint participants and integrated multimedia like audio, video, and graphics and shared data.

1. We distinguish between *applications*, which provide functionality to the end user, and *services*, like audio, video, and data transport, which are available to be incorporated in those applications.

3.3 Heterogeneity

In the distant past, telecommunications was characterized by a relatively straightforward homogeneous network providing a single application: voice telephony. As time has passed, this simple environment has evolved in different directions:

- There has been a proliferation of new applications (such as voiceband data, videophone, and facsimile) utilizing the telephony infrastructure.
- New standards have proliferated for any given application, such as voice or video encoding or voiceband data.
- New telecommunications media, such as coaxial cable, fiber optics, microwave radio, and recently mobile radio and wireless infrared have appeared.
- To date, most applications have been realized by dedicated special-purpose and non-programmable terminals, like telephones. This is rapidly changing, as programmable platforms such as the desktop computer, notebook computer, and personal digital assistant are becoming increasingly viewed as *communications* devices.
- An impact of global telecommunications deregulation is a proliferation of service providers, often with more than one service provider involved in provisioning a single application, along with the involvement of new equipment vendors (like the computer companies).

From a technical perspective, these developments imply a rapidly increasing level of *heterogeneity* in applications, *heterogeneity* in transport systems, and *heterogeneity* in terminals. Meanwhile, application developers, and especially end users, would like to be isolated from the myriad technologies and enterprises involved in their provisioning. They want applications to operate seamlessly across the telecommunications infrastructure, with both applications and networks appropriately scaling and configuring to whatever detailed technological components are involved. It is not acceptable to users to have their telecommunications applications restricted to only a portion of the network (and hence to a subset of the other users), or to equipment from a particular vendor. All parties involved -- users, application providers, content providers, and equipment vendors -- want a flexible and dynamic network that can scale and evolve to meet whatever demands are placed on it, and accommodate new advances without major dislocations or disinvestment. Users don’t want a closed, proprietary network that limits the available applications to those envisioned by a limited set of vendors and service providers, but rather an environment in which a variety of competing vendors and service providers can flourish, and innovation can reign supreme. These properties that have led to the recent success of the Internet, for example.

These factors imply a telecommunications infrastructure in the future that is vastly more complex than any existing software system. (In fact, the infrastructure will incorporate *many* existing and new large software systems, among other elements, like unreliable physical channels.) Unlike typical large software systems, it will be designed not by a single organization, but literally hundreds of equipment and software vendors, and tens of standards bodies and provider organizations. If experience with the development of software systems is a guide -- and it *should* be -- success will hinge on the containment of the complexity inherent in this large heterogeneous system.

4.0 SHORT TUTORIAL ON COMPLEXITY MANAGEMENT

We have asserted that management of complexity is a central issue in the future of telecommunications. What does this mean? How do we accomplish it? Can it be systematized, or is it largely an organizational management issue?

While large telecommunications systems are different from software systems, experience with the management of complexity in the latter domain is relevant, and can serve as a useful starting point in our thinking. Thus, we briefly summarize some techniques for complexity management from that domain.

Three basic components of complexity management are shown in Figure 1:

- *Architecture* is the prior plan of the system that partitions functionality among a set of interacting modules[1]. Modules are planned in a way that deliberately constrains their functionality to a limited set of mutually predictable behaviors, and which avoids interaction among their internal design details.
- *Theory* exploits the constrained behavior of architectural modules to establish certain predictable properties, or perhaps the absence of certain undesired behaviors (such as deadlock, instability, etc.)[2].
- *Tools* are software systems that keep track of the large design databases typical of complex systems, systematically synthesize more routine parts of the system, etc[3].

Architecture is by far the most critical element, and will now be discussed in greater detail, including some related elements like interfaces, abstraction, configurability and scalability, and reuse.

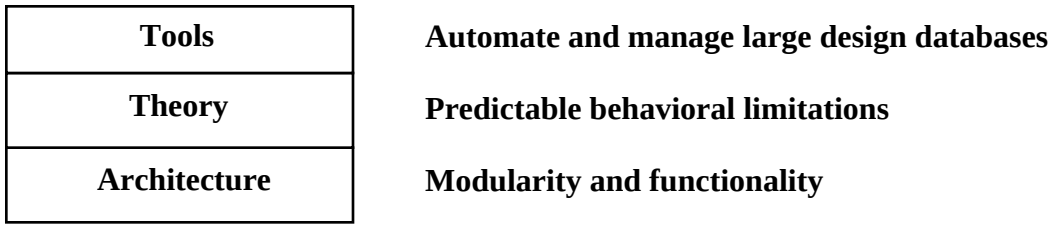

Figure 1. Three basic elements of complexity management.

1. It is interesting to compare with the more general dictionary definition of these terms. *Architecture* is defined as "the manner in which the components of a computer or computer system are organized and integrated" [2].
2. *Theory* is defined as "the analysis of a set of facts in their relation to one another" [2].
3. A *tool* is defined as "something (as an instrument or apparatus) used in performing an operation or necessary in the practice of a vocation or profession" [2].

4.1 Architecture

The architecture of the system is the basic plan that insures it performs the functions for which it is intended. The most important aspect of the architecture is the basic *modularity*, or partitioning of functionality into mutually interacting elements[1]. The interacting modules, which display limited and predictable behaviors, should be as *independent* from one another as possible. Further, they are designed with carefully constructed *interfaces*, with the internal implementation beyond what is displayed at those interfaces carefully hidden. By independence of modules, we mean that the functionality of one module should depend on the functionality of other modules only the extent appropriate and necessary, and not on incidental implementation details. This independence makes the system more open to change and evolution, since changes to the implementation of one module should not affect other modules.

A familiar example of an architecture is a computing system, shown in Figure 2. This oversimplified architecture divides a computer system into basic modules of arithmetic and logic unit (ALU), cache memory, main memory, secondary storage, and a bus that serves to connect the other modules. Each module has a clearly defined and limited function. The bus is key to insuring independence of the other modules, since it forces them to communicate in a standardized (module-independent) way. By being asynchronous (using handshaking techniques), valid operation can even be insured independent of speed.

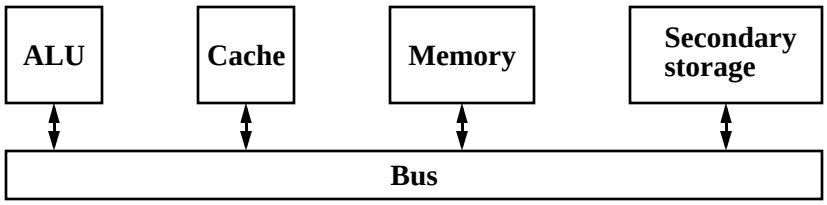

Figure 2. A simplified and familiar example of an architecture.

4.2 Abstraction

Abstraction is a key concept underlying modularity, as well as other aspects of the design process[2]. Abstraction refers to the conscious hiding of unnecessary implementation or functional details while making visible behavioral properties that are essential and important to other modules. Abstraction helps insure the independence of the architectural modules, as one module cannot be dependent on the hidden properties of another module. For example, in Figure 2, the visible properties of the memory may be sequenced read or write requests, information stored in fixed sizes units, an address associated with each unit, and random access. Deliberately hidden at the interface are the technology (be it CMOS, bipolar, bubble memory, etc.), the speed of memory access (variations are accommodated by handshaking), and internal organization. The internal organization may be quite different than suggested by the interface abstraction, for example

1. A *module* is defined as “an independently-operable unit that is a part of the total structure” [2].
2. *Abstract* is defined as “disassociated from any specific instance” [2].

turning streamed access into an random access by internally performing multiple accesses (as would be the case in magnetic storage).

Abstraction can be used in other ways that are important to containing the complexity of a design. One is illustrated in Figure 3, where a layered logical (as opposed to physical or functional) representation of the same computer system is shown. The electronic *device* can be considered an abstraction of the underlying semiconductor physics, hiding unnecessary details like holes and electronics while preserving terminal properties like the voltage-current curve. *Logic* defines modules (consisting internally of transistors) with limited behaviors like “nor gates” and “inverters” that hide device properties and circuit details (like CMOS vs. bipolar technology). The logic layer also defines arguably one of the most important abstractions of modern technology, the “bit”, which simplifies the logic outputs to have only two states (the reality is more complicated). *Registers* define higher-level modules like “shift registers” and “two’s complement adders” which form specific functions while hiding their internal logic implementation details (and indeed admitting many possible implementations). The register-transfer level defines the basic modules used in the architecture of the ALU, which externally hides such details through the definition of an instruction set. The *instruction set* presents a powerful abstraction to the software, since it separates the latter from all the internal details of the hardware. Similarly, the *operating system* layer defines a number of abstractions that separate the user-level process from the details of memory, storage, and communications resource management.

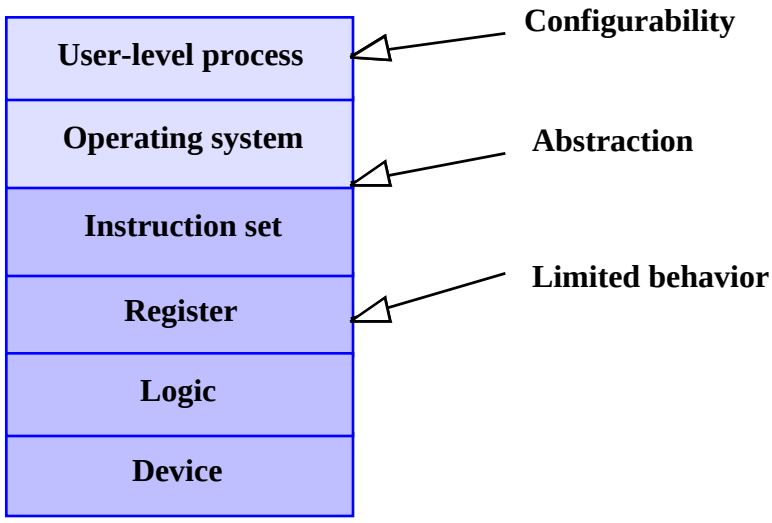

Figure 3. A familiar example of the use of abstraction in a computing system ALU.

4.3 Reusability: configurability, scalability, and adaptability

One basic approach of complexity management is reusability, which has several aspects:

- *Modularity* encourages reusability, since it defines a grouping of functionality that is carefully defined and documented. *Abstraction* at the interface enhances reusability since usage is separated from implementation specifics.

- *Configurability* insures that modules are not designed for a very specific environment or use, but rather internal parameters can be modified to accommodate new unanticipated uses.
- Closely associated with configurability is *scalability*, a property of the architecture that maximizes the flexibility to configure to different levels of performance (processing power, bandwidth, number of ports, etc.), as unconstrained as possible by technology. A scalable architecture is more broadly configurable.
- *Adaptability* requires configurability, and adds the capability to base the configuration on active observation of the module environment.

The simplest implication of reusability is the saving of design costs. In an operational system, many of the same characteristics that allow reusability can lead to a system that is self-organizing. Configurable and adaptable modules that adjust to their observed environment allow a distributed-control architecture that is in itself a powerful form of complexity management.

5.0 A TELECOMMUNICATIONS EXAMPLE

In addition to the generic issues of heterogeneity, future networks supporting multimedia datatypes will have a number of objectives, many of which interact in complicated ways. These include:

- Differing connection *topologies*, including point-to-point, multicast, and multisource. (For example, video conferencing requires both multicast and multisource topologies.)
- *Untethered* (no wires), *nomadic* (accessible from different locations), and *mobile* (accessible while moving) multimedia applications will be important for some users.
- *Privacy* by *end-to-end encryption* will be demanded by a subset of the users, as they gain familiarity with privacy in their data applications.
- *Traffic efficiency*, which is particularly an issue on wireless access links. Wireless access traffic is interference-limited within a given volume of space and bandwidth, and will become an increasingly serious bottleneck as backbone networks become faster.
- High *subjective quality*, which is indirectly impacted by compression algorithms and by loss and corruption in the transport.
- *Low latency* is critical for some interactive applications. Since the propagation delay in terrestrial global networks is significant (hundreds of milliseconds), there is little headroom to increase latency through signal processing and queueing for these interactive applications.

Simultaneously satisfying these requirements requires a carefully crafted architecture. In part this is because these diverse requirements create dependencies among system elements that must be carefully managed for good modularity. We will now give illustrative examples from our own work.

Achieving high traffic capacity on wireless links requires joint source/channel coding (JSCC); that is, coordination of the source coding with resource costs in the transport. For example, the source coding will display a trade-off between bitrate and bit-error tolerance, while the transmission media associates a resource cost to the bitrate, delay, and reliability of each source

stream. Much past source coding research has emphasized the minimization of bitrate -- without regard for the reliability or delay requirements -- but wireless links attach a high cost to the stringent reliability requirements that are often associated with aggressive compression. Maximizing traffic capacity requires adjusting the source bitrate, reliability and delay trade-offs, taking into account the resource cost of these quality-of service (QOS) requirements in the transmission media. The source representation must be highly scalable to different bitrates and reliability, for example connections with and without wireless access. Further gains can be achieved by segmenting information from a single source into different QOS classes, and fine tuning the transmission resources (like power, coding redundancy, etc.) so that no source segment receives more transmission resources than required.

The concatenation of transmission media with quite different characteristics (for example a broadband backbone with wireless access will be common in the future) is a complication. It is easy in this context to seriously violate modularity, for example, by using transcoding from one source coding standard to another as shown in Figure 4a. Using a different compression standard on each link allows a customized per-link JSCC. (A past example of this is digital cellular telephony, which utilizes transcoding from 64 kb/s μ-255 speech to 13 kb/s VCELP.) Transcoding has poor modularity because the source coding and the transport are designed tightly as a unit, and one cannot easily change without the other. Further, it is difficult to introduce new and improved compression standards if existing standards are already widely deployed within the network. There are problems related to the other requirements as well. The transcoder introduces considerable delay (a serious disadvantage in a global network) and the accumulation of quantization impairment, and is incompatible with end-to-end encryption.

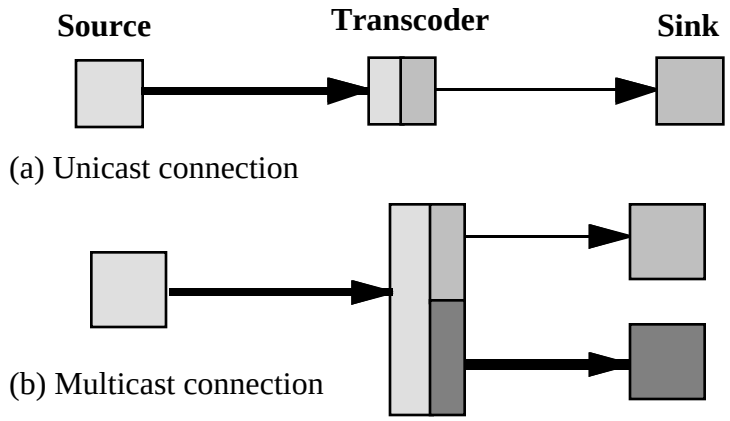

Figure 4. Transcoder architecture for JSCC in a heterogeneous transport environment.

Multicast connections (from a single source to two or more sinks) are a scalable solution to multiple sinks (as in group video conferencing for example), since the source generates a single stream irrespective of the number of sinks. However, multicast offers a more serious JSCC challenge, as illustrated in Figure 4b. Downstream from a multicast splitting point, there will in general be heterogeneous transport links that have to be accommodated *simultaneously.* Again, transcoding is a feasible solution, but one that has the same difficulties as in unicast.

Fortunately, JSCC in both unicast and multicast connections can be accommodated utilizing the alternative substream architecture [3] shown in Figure 5. The abstraction of the transport from the source perspective is a set of substreams, with different quality of service (QOS) requirements (delay and reliability) for each substream. Those QOS requirements can be negotiated between source and transport at setup. The source then configures itself for substreams with the expected QOS, trying to attain the highest subjective quality. To the transport, the source is abstracted as a set of substreams with specified QOS objectives. Internally, the transport disaggregates the negotiated QOS for each substream to configure the QOS of the individual links.

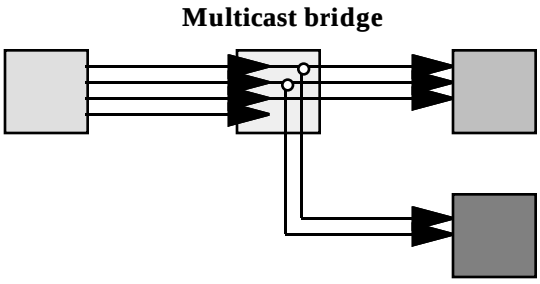

Figure 5. A substream architecture manages JSCC as well as heterogeneous transport and terminal environment in unicast and multicast connections.

Multicast presents special scalability challenges. On the one hand, heterogeneous downstream links and terminals can be accommodated by choosing a subset of substreams for each downstream multicast subtree. On the other hand, sinks may be entering or leaving the multicast group at any time. It is not scalable to presume that the source can negotiate with an indeterminate (and even varying) number of sinks and transport links, or that all sinks should be required to reconfigure whenever a sink enters or leaves the multicast group. Thus, a serious challenge is to configure the source coding and substreams to *simultaneously* satisfy the differing needs of a generic and probably unknown set of downstream transport links and sinks.

The substream abstraction offers good modularity, since the source needs no knowledge of the details of how the transport achieves a given QOS. The only details of the transport visible to the source are the fundamental properties: delay and reliability. The transport has no knowledge of the service or compression standard being utilized, only the desired QOS (delay and reliability) and bitrate. Configurability allows reuse of the transport for new sources or compression standards in the future. Further, the architecture is consistent with end-to-end encryption, as long as each substream is independently encrypted.

Having defined an architecture, a number of interesting new research issues arise. Source coding must interface the variable-QOS substream transport abstraction, configuring to the available substream QOS while achieving the highest subjective quality [4]. In addition, source coding needs scalability to the varying bandwidth, processing, and display resolution of heterogeneous sinks. (In fact, if multicast connections are supported, the source representation has to embed these differing requirements in a common set of substreams.) Another interesting problem is JSCC in the delay dimension, which we have addressed for both video [5][6] and graphics [7]. Within the transport, exploiting the substream architecture for higher multiaccess

wireless traffic capacity is particularly interesting, as it leads to the new problem of variable QOS in multiaccess environments. We have addressed this in CDMA, utilizing power control to provision variable reliability QOS [8][9], as well as packet scheduling to provision variable delay QOS [6].

6.0 CONCLUSIONS

As we move from homogeneous networks provisioning simple, universal, largely non-configurable telecommunications applications, to an environment that is heterogeneous in applications, transport media, and terminals, complexity management becomes a critical element of success. Carefully crafted architectures are needed to meet all the functional requirements, achieve adequate levels of performance, and offer a seamless environment for the provisioning of applications. Complexity management, far from displacing traditional signal processing, communications theory, and queuing considerations, raises many interesting new questions and serious challenges in all the detailed constituent traditional disciplines, such as compression, encryption, error-control, modulation, protocols, etc. While efficiency remains an important consideration in multiaccess wireless systems, many other considerations relating to the new functional, configurability, and scalability requirements are brought to the fore. These traditional disciplines will doubtless be revitalized by these new issues.

7.0 REFERENCES

1. D.G. Messerschmitt, “The convergence of communications and computing: What are the implications today?”, submitted to *IEEE Proceedings*. (Also available at http://www.eecs.berkeley.edu/~messer/PAPERS/PRESS/Convergence.html)

2. *Merriam-Webster's Collegiate Dictionary*, Tenth Edition, Merriam-Webster, Incorporated, 1995.

3. P.Haskell and D.G. Messerschmitt, “In favor of an enhanced network interface for multimedia services”, to appear in *IEEE Multimedia Magazine*.

4. L.C. Yun and D.G. Messerschmitt, “Digital Video in a Fading Interference Wireless Environment”, *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing,* Atlanta, GA., May 1996.

5. Lao, A., Reason, J., and Messerschmitt, D.G., “Layered asynchronous video for wireless services”, *IEEE Workshop on Mobile Computing Systems and Applications,* Santa Cruz, CA., Dec. 1994.

6. J.M. Reason, L.C. Yun, A.Y. Lao, D.G. Messerschmitt, ``Asynchronous Video: Coordinated Video Coding and Transport for Heterogeneous Networks with Wireless Access'', *Mobile Computing*, H. F. Korth and T. Imielinski, Ed., Kluwer Academic Press, Boston, MA., 1995.

7. R. Han and D.G. Messerschmitt, ``Asymptotically Reliable Transport of Text/Graphics Over Wireless Channels", *Proc. Multimedia Computing and Networking*, San Jose, January 29-31, 1996.

8. L.C.Yun and D.G. Messerschmitt, "Power Control and Coding for Variable QOS on a CDMA Channel", *Proc. IEEE Military Communications Conference*, Oct. 1994.

9. L.C. Yun and D.G. Messerschmitt, "Variable quality of service in CDMA systems by statistical power control'", *Proc. IEEE International Conference on Communications*, June 18-21, 1995, Seattle, WA.