True Distributed Processing in Modular Traffic Signal Systems—San Antonio Downtown System

RICHARD W. DENNEY, JR., AND MICHAEL J. CHASE

In San Antonio, limited funds precluded the traditional approach to a consultant-designed Urban Traffic Control System (UTCS) for traffic signal control in the downtown area. By applying principles now common in other parts of the computer industry, San Antonio engineers were able to formulate an alternative to the traditional approach that not only provides very substantial improvements in cost-effectiveness, maintainability, and reliability, but also provides the end user with complete access to the inner workings of the system. Emphasis on conforming to computer-industry standards of system design and commitment to open hardware and software architecture allowed full portability of software. Highly distributed processing greatly reduced the communications overhead while improving operation compared with other large-scale systems currently in operation in the United States. The San Antonio system has full-featured capabilities, including planned traffic-responsive operation, and uses one-tenth of the usual communications overhead without using a machine larger than a microcomputer. A comparison is made with a recent UTCS project.

The development of traffic signal systems has moved in two directions since the introduction of the Urban Traffic Control System (UTCS) in the early 1970s. The UTCS approach (*1*) is followed in most large system applications (more than 32 intersections); because of manufacturer specificity, approaches with more complete distribution of processing have evolved in order to provide a more cost-effective product for smaller systems.

Despite the obsolescence of the original UTCS approach to processing and communications, it is a boon to public-sector traffic engineers who must rely on consultants for design and who need a product allowing open competition for local controllers. Public employees could go to their city council or commission and present the design of the system as a consultant project (therefore not requiring low-bid purchasing in most areas) without being locked in to a specific manufacturer for future maintenance supplies. Unfortunately, UTCS is highly oriented to central processing and, even in its more recent manifestations, requires a large central computer and a dense communications network. Later implementations reduce the need for central computing and communications, but require customized programming, usually by the consultant designing the system. Therefore many of the significant modules of the UTCS software end up looking not at all like the UTCS original, thus undermining the purpose of public-domain software.

R. W. Denney, Jr., City of San Antonio, P.O. Box 839966, San Antonio, Tex., 78283-3966. M. J. Chase, Boulder Software Group, P.O. Box 14200, Boulder, Colo. 80308-4200.

More up-to-date systems are available from a number of manufacturers. Many of these products boast excellent performance, but all require a commitment to a particular manufacturer for maintenance and expansion materials. Also, many of the manufacturers are justifiably reluctant to expose their communications methods and protocols, preventing a city from improving or enhancing the system without going back to the manufacturer. Confidentiality of the protocols also makes it impossible for cities willing to experiment with different control technologies to get their ideas on the street without significant cost. Many signal system experts in the public sector are presented with a choice between the high cost of trying new methods with the manufacturer making the modification, or the even higher cost of developing an in-house system so that the public agency will own the program code. The latter choice is available only to very large organizations, for example, the Texas Department of Highways and Public Transportation's development of the Flexible Advanced Computer Traffic Signal System (*2*).

Most of the manufacturer-specific systems are designed for smaller applications. They are often referred to as closed-loop systems because they "close the loop" with the traffic engineer by providing remote access to the system. Larger cities find it easier to purchase these systems because they can still purchase other systems from other vendors for other parts of the city.

A common thread among many traffic engineers is the consideration of these systems primarily as traffic signal systems rather than as computer systems. A natural result of this thinking is that the traffic control features are mixed in with computer system features, and often traffic control considerations overshadow computer system necessities. Also, traffic engineers find themselves in the uncomfortable position of trying to toughen up (supposedly) competitive specifications in order to get the features they need.

This paper presents an example of a different approach to the conception of a large system. In San Antonio's case, this system was brought about by necessity. The bond program approved for the downtown signal system did not provide enough funds for a UTCS approach, nor were there adequate funds to pay for an outside design. In 1984, voters in San Antonio approved $1.5 million in general obligation bonds for a new traffic signal system in the downtown area. Because of lack of engineering resources, the project was not begun until August 1987. The central business district in San Antonio includes about 150 traffic signals, which were part of a PR system originally installed in 1957. San Antonio is the ninth largest city in the United States, with a population of over 1 million and approximately 1,150 traffic signals.

The different approach to the San Antonio system can be summed in one phrase: Design a computer system as a computer system, and then get the software it needs to run traffic signals. We believe that this thinking has resulted in the most cost-effective system in the United States, comparing favorably with recent independent developments in Taiwan and elsewhere, without sacrificing any capabilities that are important to the task of traffic control, especially access to the operation of the system by the owner to allow experimentation with various control strategies.

This paper first describes the general design of the San Antonio system, and then compares the resulting "computer-system" features to a recent implementation of UTCS. To emphasize the treatment of signal systems as computer systems, we make frequent comparisons to a familiar part of the computer industry—the standard desktop microcomputer.

DISTRIBUTED PROCESSING IN SAN ANTONIO

When setting out to purchase a microcomputer, the first-time buyer usually thinks only generally of the intended use, and concentrates primarily on processing power, user access, and other computing features. Most first-time users have too little experience with the available software packages to make an informed decision about which hardware to use. Generally, people using computers for artistic purposes, such as visual or musical art, purchase a Macintosh, and those needing more business-oriented applications use an IBM-type system. The distinction between the suitability of these different environments for various purposes has, however, blurred significantly over the years, and now most any kind of work can be done on either machine. The point is that many people select a hardware platform based more on suppositions of intended use rather than on actual experience with the software, and then create a demand for the software to run, say, Macintosh-like applications on their IBM, and vice versa. Very few first-time buyers can predict the applications to which they will put their computers, and most applications are discovered after the machine is available.

Most people, therefore, in some way violate the traditional approach to buying computers: Define specific uses, and then select the software and hardware to provide those capabilities. Rather, they buy the most powerful hardware they can afford, and then obtain software as the need arises. This approach may seem less "systematic," but it makes more sense in the long run. Using traditional thinking, users run the risk of having machines that cannot grow with their improved understanding, and they limit themselves to a premature concept. For example, many owners of off-brand microcomputers now look longingly at the IBM machines their competitors are using. The off-brand system was the most effective provider of their original concept, but now cannot do the other things their users now realize they need. In many cases, the off-brand computer costs as much or more than the more widely used IBM-based counterpart.

Purchasers of traffic signal systems have tended to make the mistake that most microcomputer purchasers have avoided (even if by accident). No specifications are more tightly written in the traffic engineering industry than those written to purchase signal systems. However, little detailed thinking is done about the computer hardware itself.

In San Antonio, we first committed to the hardware and then wrote a specification for software. This approach seems simplistic, and we emphasize that detailed thinking went into the hardware decision. When considering the hardware, we established a basic application of distributed processing by determining the level at which each function in the system would be performed. The basic rule of distributed processing is to place the processing power as close as possible to the processing need. The closer it is, the simpler and more economical the communications task becomes. In San Antonio, this approach led us to the following conclusions:

- All local intersection timing should be done by the local controller. Cycle length, offsets, splits, and even time-of-day and pattern scheduling should be handled at the point where it is used. As any operator of a time-based coordinated system knows, most controllers on the market already have these capabilities.
- All decisions made at the control-group level should be made by the zone master. Most systems on the market today use zone masters to control groups of local intersections. If local controllers can make all decisions about local timing, then the zone master must only handle traffic-responsive pattern selection and broker communications between the central computer and the local intersection. From a computer system standpoint, the latter feature is critical. If the central computer is to talk to individual intersections directly, as in the traditional UTCS system, the communications task is enormous, requiring large-scale multiplexing to allow messages for hundreds of locals to come into relatively few communications channels, or requiring a very expensive central computer to allow a channel for each local. Large systems no longer require the latter method.

From a traffic control perspective, the use of control groups and zone masters may, at first, seem to get in the way of the more subtle requirements of traffic control systems. For example, one significant feature of the UTCS software is the ability to redefine control-group boundaries by time of day. This feature is useful, though not implemented often, but it is an outgrowth of the basic UTCS operation. The original UTCS system used the central computer to advance the interval of each controller on the system, and all cycle length and split decisions were directed at the central point. The software was designed to break up the system into subsystems; within each subsystem the pattern selected would be the same. In a system where all signal timing is done at the local level, the zone master does not even have to know what the cycle length is, and control-group boundaries can strictly be a function of the way the patterns are designed in the local controller.

- Supervision of zone masters and provision of user access should be done at the central computer. The central computer has the capability to store information about the whole system, and should maintain a copy of the timing parameters, which are contained in each local and zone master to minimize the need for communications to provide the user with day-to-day information. In San Antonio, access to the system is needed from different locations, including the maintenance facility, the control center, the traffic engineering office, and the homes of staff members required to monitor the system after hours (to troubleshoot a problem, for instance). This need requires a multiuser access capability, such as that provided by minicomputers and microcomputers running a multiuser operating system, such as UNIX.

- Graphics displays and the user interface should be run on the user terminal. An operating system such as UNIX provides many of the capabilities for graphic display and user access to the system data base; however the substantial recurring graphics and screen display information would move very slowly over a modem or other long-distance serial link, and competition with other users would decrease some response. Following the rule of distributed processing, one realizes that the *pictures* are always the same, only the *data* presented in those pictures need be communicated from the central computer. By using terminals that are standard microcomputers, the user interface can be programmed on the terminal, thus freeing the central computer of the task of making and communicating pictures. Thus, remote terminals accessing the system via dial-up telephone links will be able to enjoy the same graphics and user interface as terminals with a direct link, with no significant loss in response time.

San Antonio selected the Type 170 controller for use as the local controller. The Type 170 is a microcomputer (albeit not a very powerful one), and its open architecture allows separate purchase of the hardware and software (*3*). The development of the software for the Type 170 controllers was included along with the development of the central computer software, and the hardware was purchased with the annual supply contract for controllers. This allowed us to specify the desired features from the software developer in a negotiated consultant contract (which is not based on low bid) and purchase the hardware separately and competitively. This arrangement would not have been possible with controllers following the National Electrical Manufacturers Association (NEMA) specification (*4*), where hardware and software are linked in a closed architecture. The result is that software features may be sacrificed in order to allow a competitive hardware bid.

The advantage of separately purchased software cannot be overemphasized. While in the NEMA sphere, features must be common to several manufacturers before a competitive specification can be formulated. The ability to purchase software separately allows features that are not even available to be specified, because the features are in the software and are not competitively bid. Therefore the tools of innovation are back in the hands of the system buyers, rather than hardware manufacturers who at times must have divergent concerns. Again, this is analogous to the personal computer industry. When IBM introduced the personal computer (PC), it opened the books on its architecture, bus, and peripheral design. Consequently, a foundation now exists for providing applications not dreamed of by the original designers. Innovative software, however, is only developed in a competitive environment where software developers are working from a common and open hardware standard. Such a standard is provided by the Type 170 specification, but not by the NEMA specification.

Based on these design guidelines, which were detailed in a 13-page preliminary design report, the city of San Antonion awarded a $163,000 consultant contract to develop the software and obtain the computer equipment.

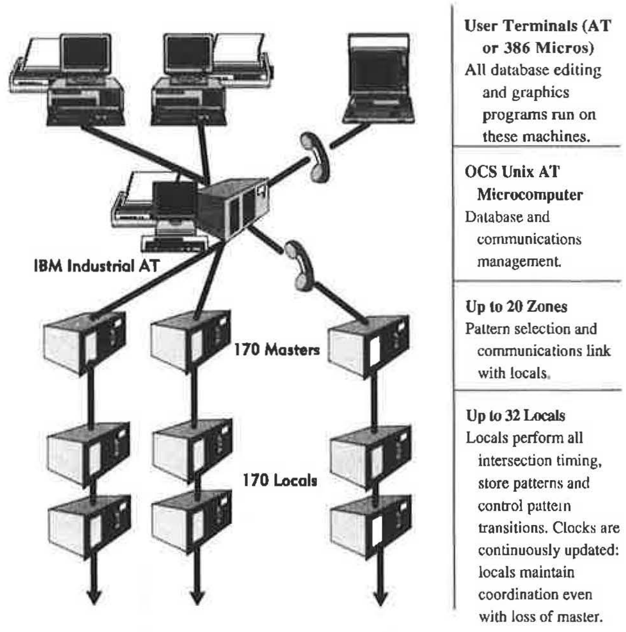
SYSTEM DESCRIPTION

The multilevel hierarchy of the system architecture allows independent development of the various components of the system. The mechanism that directs the interaction of these various processes is called a kernel; in this case the kernel is part of the central control program (CCP) software running on the central computer under UNIX. The other components of the software communicate with each other by sending a series of messages to the kernel to be passed on to other processes. By defining these messages first, the basic function of each component of the software can be defined before the program is actually written, and all program modules are written with a firm understanding of how they fit into the whole. This system description begins with a discussion of the kernel and the central computer, and then describes some of the features of the local controllers, zone masters, and terminals.

Kernel and Central Hardware

The CCP kernel is written in the C programming language under the UNIX operating system, following the AT&T System V Interface Definition and ANSI X3.159 programming conventions. As such, it will run without modification when compiled on any computer running AT&T standard System V UNIX. Therefore, the size of the central computer then becomes strictly a question of needed hardware horsepower. Because the central computer is not required to direct individual intersection operation or generate presentation graphics, the task of simply brokering communications of the various elements of the system is not large, even on a large system. Processing power is not as critical as communications capability. Using intelligent, multiple serial port add-on boards made by one of several manufacturers, a standard desktop PC can have up to 24 serial ports. Each serial port is used to talk to a zone master, a terminal, or a modem. In San Antonio's design (see Figure 1), two directly connected terminals use two of these ports. Other users access the system via a dial-up modem, which uses another two ports (to allow two dial-up users at one time). The remaining 20 ports in a PC environment can be used to talk with 20 zone masters. As shown later, each zone master can operate up to 32 locals, allowing a theoretical maximum of 640 intersections on a PC-based system. In actual practice, running the system at capacity would inhibit the flexibility to configure some zone masters with less than 32 intersections, which would prohibit future in-filling.

The advantage of using C and UNIX becomes apparent when the system outgrows the 24 serial ports allowed on standard microcomputers. The central computer can then be replaced with a small and inexpensive minicomputer, which can easily accommodate up to 64 serial connections. No modifi-



User Terminals (AT or 386 Micros)
All database editing and graphics programs run on these machines.

OCS Unix AT Microcomputer
Database and communications management.

Up to 20 Zones
Pattern selection and communications link with locals.

Up to 32 Locals
Locals perform all intersection timing, store patterns and control pattern transitions. Clocks are continuously updated: locals maintain coordination even with loss of master.

FIGURE 1 Organization of San Antonio CBD signal system.

cation of the software will be required, only recompiling the C programs on the new machine (C compilers are readily available in all UNIX implementations). Because UNIX provides complete access to communications hardware in the machine without machine-dependent instructions, the software is completely portable across all machines running UNIX.

The downtown system in San Antonio contains only 150 intersections; therefore, the PC environment was chosen. The hardware itself is an IBM Industrial AT, which is a rack-mounted version of the standard desktop AT. The rack-mounted case offers better control and filtration of cooling air, providing a cleaner environment for the hard disk. Otherwise, the components of the Industrial AT are no different from the components in a standard desktop PC-AT. The AT included a 30-megabyte hard disk, monochrome monitor, 4 megabytes of random access memory, and three plug-in cards with eight serial ports each. Also purchased was a 750-watt uninterruptable power supply, which also includes line conditioning and filtration. All these components can be easily and cheaply bought in any computer store, and no component of the system costs more than a few hundred dollars.

Because UNIX is a multiuser, multitasking operating system, many programs can run simultaneously. Each program running under UNIX is called a process. A process is invoked for each serial port, whether it is a terminal or a zone master; in addition, processes are run to maintain information about the current status of the system and to provide real-time supervision of the zone masters.

All of the traditional information reporting the status and integrity of the computer system is provided as in any large-scale system, including alarm status of zone masters and locals, on-line status, local controller status (current plan, conflict flash, etc.), and so on. Also added to these features are the standard features found on all UNIX systems, such as multiple-level security, electronic mail, user maintenance, and non-CCP programs.

The final user of the system is encouraged to make modifications to enhance the system; the user can write programs for independent processes that communicate with the CCP via the messages originally defined. For example, the San Antonio system specification requires the ability for the city staff to experiment with different means of coordinating the effort of adjacent traffic-responsive zone masters. Several schemes for coordinating control groups have been formulated, but few have been experimented with because implementing the proposals traditionally requires costly modifications to proprietary software. The result is that most systems do not grow with the improved understanding of their operators, as described previously. The city of San Antonio explicitly avoided this drawback by requiring an open architecture at the outset. City staff will be able to write a small program implementing a proposed algorithm for subsystem coordination by merely asking for the appropriate messages from the CCP kernel, making the necessary calculations, and sending the appropriate messages back to the kernel for subsequent direction to the zone masters. This independent program can be developed in any computer language running under UNIX (e.g., C, Pascal, FORTRAN). This example illustrates the need for providing mechanisms for future enhancements without necessarily knowing what those enhancements might be. Indeed, the algorithm used in this example is not yet formulated, and no research has been undertaken to shed light on how it might work.

The ability of the end user to add or modify control algorithms is one of the fundamentally important benefits of standard programming practices and open software architecture. The following scenario illustrates how this would work using the above example of traffic-responsive zone master control. The user would write a program called, say, Zone_Control. Zone_Control would be a continuously running process under UNIX. At the beginning of every control period, say, every 5 min or so, Zone_Control would send a message to the kernel to get the choice made by Zone Master 6. If Zone Master 6 traffic-responsively selected Plan 5, for instance, Zone_Control would discover this by sending the message Get_The_Selected_Plan(Master_6). The kernel is programmed to respond to such a message by sending a message to the process communicating with the zone master, which then queries the master and returns the result back to the kernel. The kernel then sends the return message Selected_Plan(Master_6, Plan_5) back to Zone_Control. Zone_Control then makes the decision to confirm or deny this selection by looking at the adjacent zones, and perhaps specific detector data (which would be obtained by sending a message to the kernel, etc.). At the conclusion of this algorithm, Zone_Control would send the message Run_This_Plan(Master_6, Plan_4) if the algorithm determined that Plan 4 would better suit the situation. Zone_Control could be easily programmed in Pascal, for example, and would use the standard UNIX techniques for talking with other processes. The programmer of Zone_Control would not have to know anything about the mechanics of communicating with signal controllers on the street, nor about how the kernel works. Only the messages (Get_The_Selected_Plan) and the system constants (Master_6) must be known.

As the technology of traffic control proceeds, these programming techniques will become more important. A gap already exists between computer programmers and most traffic engineers; a message-based control system allows the engineer to reasonably map out the logic of traffic control features without understanding the details of programming the communications and data base protocals. Many practicing engineers will be able to go even farther by actually writing the simple programs involved.

Zone Masters

As previously indicated, the zone masters will control up to 32 intersections. Because the locals already maintain time of day, it is not necessary for the zone master to provide any time-critical information to the local, such as interval advance or synchronization pulse. The zone master is therefore free to poll the intersection continuously while the communications capacity is not otherwise used. For example, the zone master polls each of the 32 intersections once each 1 or 2 sec, unless the user wants to look at a graphic display of an intersection. The needs of a real-time graphic display are too large to allow polling every second, and the zone master, while maintaining the high flow of information required by the graphic display; may only be able to poll the other intersections once every 5 or 6 sec. This is not a problem, because nothing being communicated is time-critical. Consequently, the demand on the communications network is greatly reduced, and only two twisted pairs are required.

The zone master sends and receives a small packet of information from each local. The outgoing packet includes the address of the local being accessed, the selected pattern, and the time of day. The returning information includes the status of the intersection, the green return for the coordinated phases, and any system detector information for system detectors attached to that local controller.

Because the time of day is sent on each poll, the system clocks are always updated and synchronized. In the event of a failure, the locals are all synchronized, and automatic time-based coordination proceeds.

These features were already available in the zone master software running on Type 170 controllers at the time the project began. For this project, we added a further requirements that each zone master was to be capable of operating up to three completely independent control groups. This allows the user to define small control groups without a corresponding increase in the communications load. Each control group can traffic-responsively select its own pattern using the standard volume plus weighted occupancy information from system detectors, a capability used by most systems in the United States. Further subdividing of the system is possible, however, by merely defining the patterns in such a way as to be compatible with the patterns in an adjacent control group, whether or not it is part of the same zone master. For example, a control group may call for Plan 6, Offset B in a particular case. In some controllers, Plan 6 is a 50-sec cycle, while in others, Plan 6 is a 70-sec cycle. If an adjacent control group is running a 70-sec cycle, then its locals will be coordinated with the 70-sec cycle controllers in the first group. The zone master only sends the current pattern to use; the corresponding cycle length in the local controller is of no consequence to the zone master. This provides complete flexibility to the signal timing designer. At system capacity, each control group will average 10 or 11 locals, but patterns can be further arranged to be compatible across control-group boundaries as necessary. With fewer controllers on each zone master, as will usually be the case, the flexibility is even greater.

Local Controller Software

The city of San Antonio has standardized on the Type 170 controller since the early 1980s. Several times during that period, systems were designed and constructed using these controllers. Each time a new system was purchased, the hardware remained the same, but the demands on the software were increased. Each time new software was purchased, features not then available were included in the specification. The result has been that the software for Type 170 controllers has been upgraded based on the direct leadership of practitioners in the field working in a noncompetitive situation with the software developers. The downtown system is a further example of this process.

The local controller software being implemented on this system includes most of the operational features found in the latest proprietary NEMA systems and far exceeds the operational requirements of the NEMA specification. For example, the controller software includes built-in time-based coordination, several interconnect alternatives (including seven-wire, NEMA-coordinator, single-pair modem, and two-pair, two-way modem communications), two levels of railroad preempts, four levels of emergency vehicle preempts, a feature to allow the coordinated phase to gap early, and the ability to allow controlled accommodation of pedestrian timing when infrequently called. Also included is a pretimed mode, very useful in the downtown system, which allows the pedestrian walk intervals to expand to use all the time available. The software also includes features not usually found in NEMA-plus controllers, such as the ability to collect and store data from all local detectors and the ability to monitor the length of actuated phases.

In addition to these features, the specification for the downtown system has added an improved method of pattern transition. Each phase has a defined timing parameter known as the transition minimum. This interval is longer than the initial time for the phase, which is (and should be) a function of the type and layout of the approach detectors. During a transition, however, these transition minimums will not be violated. When the controller receives a directive to change patterns from the zone master, it will calculate the time necessary to hold the coordinated phase until it is in step with the new pattern. The controller will then add up the transition minimums of successive phases to see if, by timing these minimums, the controller can get in step in one cycle. If not, the controller adds one cycle length to the transition time, and prorates the force-offs (splits) over that transition time. The resulting transition cycle provides the same percentage split to each phase as before, and still gets in step within a single cycle. With such rapid transitions, traffic-responsive operation can reasonably and effectively use much shorter control periods than is practical in a UTCS system, say, 5 min instead of 15 min.

Terminals

The terminals used in the system are standard desktop IBM-type microcomputers. The specification required the software consultant to obtain these computers from local San Antonio vendors in order to maintain nearby service capabilities. A PC-AT (intended as a hardware spare for the rack-mounted AT), two AT portables, and two desktop machines using the 80386 microprocessors were purchased. Also included were a plotter, a laser printer, and various off-the-shelf software packages. All desktop units have large (71-megabyte) hard disks and very-high-resolution (VGA) graphics displays. As with the central computer, all components are standard items readily available from any local computer store.

The terminal user interface (TUI) is designed to run under MS-DOS on a standalone PC workstation. It provides all the basic user editing and graphics display functions typical of the best closed-loop systems currently on the market. When logging into the system, the users enter the terminal mode of the TUI to allow direct communication with UNIX and the CCP kernel. From there, the users can perform any of the functions available directly from the CCP. Once logged in, they return to direct control of TUI, which then communicates with UNIX transparently to the users. At that time, they can upload, modify, download, and store any signal timing information in the system. They can also review the system status and observe an individual intersection using real-time graphics. The individual intersection display includes a graphics representation of the particular intersection, the current time of day, pattern, local and master cycle timers, offset, and real-time displays of each green, amber, "walk," "don't walk," and vehicle detection at the intersection. Each zone master can also be observed graphically, showing the intersection status and the green return for the coordinated phase.

DISTRIBUTED PROCESSING AND TRADITIONAL APPROACHES COMPARED

Three key results of process distribution at the level implemented in San Antonio are the reduced demand on the communications network, the reduced demand on any one component of the system, and dramatically increased reliability. In the traditional UTCS approach, the central computer is responsible for intersection timing. Even with powerful and expensive communications multiplexing, this real-time load is very demanding of central hardware, and requires a large minicomputer even for systems of moderate size. In one recent implementation of such a system, the cost for the central computers to allow a theoretical build-out of 800 intersections was $377,000, not including any ancillary computer equipment such as PCs and printers. The cost of equipment in San Antonio, for all of the PCs, including plotter, etc., and including 20 zone masters, is less than $100,000 for a theoretical capacity of 640 intersections. Software cannot be directly compared because the UTCS software, which had to be extensively (and expensively) customized, was part of the overall design contract which exceeded $500,000, and the software remains in escrow with access by the purchaser restricted. The portion of San Antonio's contract with the software developers that did not include the above hardware was less than $90,000, and includes full availability (under license) of source code, including the training necessary to know how to modify it. In another situation, the cost to update a UTCS computer to use a larger hard disk required over $10,000 in modifications to the software. In San Antonio's system, such a modification would require only the cost of the hard disk (a few hundred dollars) and the time necessary to move the system files (less than a day).

The major cost difference, however, lay in the required communications network. Because only critical data are actually transferred from one level of the system to another, and because the processing is performed at the location where it is used, the load on the network is very light. The central computer communicates with each zone master at 1,200 bits/sec on a direct serial link. This link may alternatively consist of only two wire pairs using the inexpensive short-haul serial modems now available, or dial-up modems (though real-time zone supervision and monitoring is not practical with dial-up links). In San Antonio, all Type 170 controllers running zone master software are located in the control center with the central computer, and their serial ports are directly connected to the serial connectors on the communications add-on boards.

Summarizing the cost of the system, we are paying approximately $8,000 per intersection, including the local and zone controllers, central equipment, control center remodeling, software development by consultant, and in-house engineering and construction. We plan to add as much as $1,000 to this cost for the future installation of system detectors to allow traffic-responsive operation, for a total projected cost of about $9,000 per intersection. This cost does not include communications cable, which was from the previous system. Fully expanded (640 intersections), the system would cost, excluding cable, about $6,760 per intersection, including about $6,000 for local controllers (installed).

Each zone master talks to up to 32 locals using two pairs. The standard Model 400 modems purchased with every controller are used.

Because of this very light demand on the communications plant, we were able to continue using the twisted-pair cables originally installed in 1957 for the PR system. The PR system cable includes eight circuits with nine usable pairs in each circuit for 150 intersections. We are currently using only about 40 percent of the pairs available in the cable plant. Even including the spares, the system has only 72 usable pairs entering the control center, which is sufficient to accommodate the maximum capacity of the system (40 pairs). By contrast, the UTCS system previously mentioned had over 400 pairs entering the control center, at very high cost, even considering that the system allowed local intersections to time themselves with only once-per-minute polling by the central computer.

Reliability is a key advantage to a fully distributed system. Because processing power is distributed to the lowest level possible, the criticality of key components further up the line is substantially reduced. For example, only traffic-responsive operation and continuous time-clock updating are lost when the communications network or the zone master fails, and even then they are lost only for the zone suffering the problem. The controllers revert automatically to time-based coordination on a time-of-day basis, using the timings already contained in the local hardware. If the central computer fails, the zones continue to operate independently and traffic-responsively (if so programmed). All that is lost is system-level supervision. No mechanism in the system can cause a general failure of all intersections. Such failures are, however, not unknown in the UTCS world. One of the authors saw a recent report of a UTCS system which, because of a failure in the central computer, placed hundreds of intersections on the system on conflict flash, and each local controller had to be manually reset. These kinds of failures are not possible in a system where no one element has global control at the detailed level. Most failures in the San Antonio system are monitored and repaired before any detrimental operation is seen by drivers. So far, all of the failures have been either in the local controller (to the same extent as in any other microprocessor-based controller) or in the communications network caused by the extensive construction currently under way in downtown San Antonio.

Another benefit of a high distribution of processing is the decreased physical size of the components. For example, the San Antonio system requires, including all the zone masters, three full-height racks. The rack reserved for the computer is mostly empty, and would be large enough for the small minicomputer should expansion become necessary. The UTCS example used a large minicomputer requiring six racks for the computer alone.

We hasten to affirm that the UTCS example works very well, even though future modification will be expensive and therefore not readily available to the system operators. The point in citing this example is to illustrate the potential cost savings that result by spreading the workload over enough machines so that no machine need be larger than a microcomputer, and by not communicating data to remote processors.

WHAT DOES THE FUTURE HOLD?

Traditional research into traffic signal systems has concentrated solely on the algorithms of traffic control, and little attention has been paid to the architecture of systems from a computer standpoint. Of course, traffic operations are the reason signal systems are installed in the first place, and there the emphasis must be. More study, however, of how the computer mechanisms interrelate with traffic control needs will allow us to ask more and better questions about traffic control methods. An open architecture in which many independent software developers can provide *features* for standardized hardware would result in more powerful systems with more design input by the practitioners who must implement them. Once a completely open system, such as PC users enjoy, is generally available, practitioners will be able to work with researchers to experiment with different control strategies; then the real questions will be able to be asked.

The San Antonio system is not particularly innovative, from either a traffic control standpoint or a computer system standpoint. What is innovative, we believe, is the system design approach that emphasizes good computer system thinking within a solid foundation of traffic control experience. By synergistically taking the best from each technology, the San Antonio design has illustrated the huge cost savings and other advantages of fully distributed processing, while giving the end user unprecedented access to the mechanics of the system.

We are convinced, however, that this step is only the beginning. As local intersections increase in power, more and more thinking will be done at the local intersection level. Once practitioners and researchers can program the operation of individual controllers at the algorithm level, then basic calculations (e.g., volume/capacity or residual queuing) can be done locally, greatly reducing the task of real-time signal timing optimization. Thus the door will be opened for widespread research in parallel processing and neural network technologies now causing excitement in other parts of the computer business.

Researchers in particular need these capabilities as they move more deeply into adaptive control systems and intelligent vehicle-highway systems. With control systems based on open hardware and software, researchers will be able to develop, one piece at a time, the complex control algorithms that will be required of systems that learn.

The experience in San Antonio, despite the smallness of the step in that direction, leads to the conclusion that these new technologies will only flourish under the open architecture now so important in most of the computer industry.

REFERENCES

1. R. Wilshire, R. Black, R. Grochoske, and J. Higginbotham. *Traffic Control Systems Handbook, Revised Edition—1985*. Report FHWA-IP-85-11. FHWA, U.S. Department of Transportation, 1985, pp. 7.25–7.30.
2. H. E. Haenel. *Texas Stand-Alone Arterial Systems*. Texas Department of Highways and Public Transportation, 1981.
3. *Type 170 Traffic Signal Controller System—Hardware Specification*. Report FHWA-IP-78-16. FHWA, U.S. Department of Transportation, 1978.
4. *Traffic Control Systems*. Standards Publication TS1-1983. National Electrical Manufacturers Association, Washington, D.C., 1983.

Publication of this paper sponsored by Committee on Traffic Signal Systems.