AN ISSUE IN COMPUTER SCIENCE
A CHALLENGE FROM INFORMATION TECHNOLOGY

Y.C. Tay
Department of Mathematics and Department of Computer Science
National University of Singapore
tay@acm.org http://www.math.nus.edu.sg/~mattyc

Invited Talk
Symposium on Challenges in Mathematical Sciences for the New Millennium
IIT Kharagpur, December 2000

1 Introduction

I was asked to talk on the challenges which are likely to be faced by the scientific community in the field of Computing Science and Information Technology in the years to come. Given the nature of this field, it is a daunting task for anyone, certainly impossible for me. My best shot would be to focus on two problems that motivate my own research, but which may be of general and fundamental interest.

The first problem is an issue in the theory of distributed computing [1]. This is an area of computer science that studies how entities — called processes — that are distributed across space collaborate on a computation by executing a protocol for exchanging information. The theory revolves around the correctness of these protocols in achieving their objectives, and their complexity.

A common approach in formulating the correctness proofs is to start with a global state of the processes at some instantaneous point in global time, or else assume there is a total ordering of all the events among the processes. This use of simultaneity across space and Newtonian time violates relativity and weakens the theory. As the centenary of Einstein's discovery draws near, and with plans afoot to build an interplanetary Internet, I believe the theory should be overhauled to remove this weakness.

The second problem is a challenge from engineering the Internet — that network of computers supporting the World Wide Web. This and similar networks have already suffered several serious disruptions caused by hardware failures, misbehaving routers and virus invasions [2]. There are worries that, with its frenzied growth, the Internet may suffer a congestion collapse [3].

There is no mathematical model for the behavior of the Internet: no model to show how the

number of hosts, the network topology, the protocol specifications, the traffic interactions, etc., affect its behavior; no model to guide the engineering community in designing its hardware and software architecture; no model to warn us how it might become unstable. The Internet will surely be a part of our lives for many years to come in this new century, so I believe work must begin to formulate a model that can help us analyze, understand and control its behavior.

2 An Issue

In distributed computing, a set of spatially distributed processes $P_1, \ldots, P_n$ collaborate in a computation to achieve some joint objective, using a protocol to exchange information. For example, in leadership election, the objective is to designate some P_i as the leader; in distributed consensus, each P_i may have a preferred value (say, 0 or 1) for a variable x, and the objective is for $P_1, \ldots, P_n$ to agree on a value for x; in deadlock detection, the objective is to determine if some of the P_i's are collectively waiting for messages from each other.

In the theory for distributed computing, one may wish to prove that a particular protocol is correct (e.g. a deadlock will always be detected), or that a protocol for a particular objective must have certain properties (e.g. the number of messages exchanged is exponential in n), or that no protocol is able to achieve a given objective (e.g. consensus despite a single process failure).

A common approach in the theory is to model the computation with a sequence

$$(s_{10}, s_{20}, \ldots, s_{n0}) \xrightarrow{e_1} (s_{11}, s_{21}, \ldots, s_{n1}) \xrightarrow{e_2} \cdots$$

where each s_{ik} is a state of process P_i, and each e_k is an event that occurs at some process. For the model to make sense, s_{ik} must be related to s_{jk} in some way. Lamport and Lynch, for example, define the vector as describing "the complete instantaneous state of the system" [1]; i.e. $(s_{1k}, s_{2k}, \ldots, s_{nk}) = (s_{1k}(t), s_{2k}(t), \ldots, s_{nk}(t))$ where $s_{ik}(t)$ is the state of P_i at time t. The issue is: What is t?

Some papers refer to t as "real time", meaning a global time, unobservable by any process, that measures progress for all processes, i.e. Newton's universal time. It is well-known, since the discovery of relativity, that such a t does not exist.

Alternatively, t could be fixed as time in the inertial frame of some P_i, say. In that case, however, one must consider any differences between the objective and how it is manifested in that

chosen inertial frame (e.g. whether a deadlock can always be observed in that frame). As far as I know, no paper in the literature does that. In any case, this is not doable in the context of general relativity.

One widely used variation of the model is to start with the events, rather than the states: Babaoğlu and Marzullo, for example, assume that "in an actual execution, all events, including those at different processes, occur in some total order" [4]. This just says we can interleave the events $e_1, e_2, \ldots$ in a sequence, and it is implicit in that assumption that this ordering is induced by some time t, and the same issue arises.

It may seem far-fetched that one should worry about relativity when proving the correctness, complexity or existence of an algorithm. This may be the case previously, but not so for the new century. The increasingly popular Global Positioning System (GPS) uses satellites whose velocities relative to users are large enough that relativistic effects are observable [5]. Furthermore, an interplanetary Internet is under design [6] and, once deployed, some computers in such a network may experience intense gravitational fields.

In mathematics, an elegant theorem about a property is meaningless if no mathematical object can have that property. Similarly, it is vacuous to develop a theory from an assumption that is physically false. Besides, distributed protocols do not use global time in their logic, so one should be able to reason about them without using such an artifact. Thus, mathematically and aesthetically, the global time assumption is inappropriate.

Nonetheless, it may be that the theory with or without the assumption is the same, so the best way to clinch this argument would be to produce an example that differentiates the two. Such examples were in fact given by A.K. Singh, Abraham and Ben-David [7, 8] in the context of an extensive discussion in the theory community on the semantics of a parallel execution [9]. There, the issue was whether one may assume all events can be interleaved in a total order. The objections centered on event atomicity and nondeterminism, and Pratt added a relativistic argument when he suggested using an alternative partial order semantics [10].

That discussion and those counterexamples had no noticeable impact on the theory for distributed computing. Certainly, the theory for partial order semantics — or interleaving semantics itself — is never used to prove any protocol correct. This is not strange: Group Theory may prove

that a general quintic is not solvable by radicals, but the theory is irrelevant when one is considering a particular quintic, like $(x-1)(x^4-1)=0$.

My view

I see no point in revisiting an old controversy that had no impact. Instead, I want to focus on removing simultaneous states and global ordering from proofs about distributed protocols. As alternative, I propose using concurrent states $(s_1(t_1), s_2(t_2), \ldots, s_n(t_n))$, where t_i is a local time at process P_i, and $t_i \not\prec t_j$ for any i and j.

The partial order $\preceq$ is induced by the information exchange among the processes. Lamport first suggested such an ordering, and later justified the formalism on relativistic grounds [11, 12].

I believe concurrent states are the appropriate generalization of simultaneous states for defining concepts, proving properties and thus constructing the theory for distributed computing. This view is developed in a companion paper [13].

3 A Challenge

The challenge to formulate a mathematical model for Internet behavior can be questioned in several ways:

Q1 What is the point? One basic objective in performance modeling is quantitative prediction, and that is surely impossible for the Internet.

Q2 The Internet is an enormous moving target, growing in size everyday and constantly changing in topology [14]. Even if we have a faithful model of the Internet now, it is a snapshot that will be outdated by the time we are done analyzing it. Can the analysis remain relevant?

Q3 The hardware that is the Internet spans an enormous range in capacity and characteristics, from palmtops to portals, from dial-up lines to optical fibers — some of these links may be noisy (e.g. cellular wireless) and have asymmetric bandwidths (e.g. cable modems) or changing latency (e.g. satellite connections). How can one model incorporate such heterogeneity?

Q4 The software that is the Internet is a large suite of communication protocols, a rich alphabet soup of WAP, BGP, TCP, UDP, GRE, FTP, HTTP, IP, etc.; the Transmission Control Protocol TCP alone has multiple variants. How can one model these many protocols and their complex

interactions?

Q5 The many protocols produce a variety of traffic flows: small packets (remote login), large packets (file transfer), bursty spurts (telephony), inelastic flows (video), etc. An obvious candidate for a model is the queueing network, but the characteristics of the queues and jobs are way beyond the reach of techniques for analyzing such networks: every router has a finite buffer and drops packets (according to some rule) when this is full; packets may be classified and prioritized; packet trains can appear [15]; traffic is not Poisson distributed [16]; variances may be infinite [17]; source adaptation to network dynamics introduces feedback [18]; engineering decisions introduce correlation (e.g. between round-trip time and window size, as when ISPs buffer modem input [19]); etc. Can any model (based on queues or otherwise) avoid being a hopeless simplification?

Q6 Internet traffic has weekly and hourly cycles (driven by users) and sub-minute periodicities (driven by timers in protocols and applications), and router implementations may cause route flapping [20]. How can a model deal with such lack of a steady state?

Q7 What would the input parameters to the model be? The buffer sizes, link bandwidths, queueing disciplines, connection matrix, routing tables, traffic distributions, ... there is no end to specifying the diversity in *Q2*–*Q5*.

Q8 What performance measures would the model focus on? The measures that interest engineers most are not simple averages, but metrics that are hard to estimate accurately (e.g. probability of packet loss and delay distribution) or even quantify satisfactorily (e.g. fairness in bandwidth utilization).

My view

One can argue that a question like *Q2* is a reason for having a model, rather than an objection, since we want to know if the moving target is heading in a wrong direction. Similarly, the Internet engineering community has redesigned the Internet Protocol IP — that crucial software at the waist of the hourglass-shaped protocol architecture — but no one knows how an upgrade from the current IP version 4 to version 6 will affect the Internet's performance. The community is also considering whether the Internet should switch from best-effort packet delivery to service reservation and admission control [21], and the discussion would be better-informed if there is a

model to analyze the consequences of a switch.

In any case, the questions cast doubt on the feasibility of an Internet model, but then a challenge wouldn't be one if it were obviously tractable. However, I am optimistic. To explain why, I will need to refer to my own work on other performance models, since my optimism is based on that experience.

One way to construct an Internet model would be bottom-up, by first modeling the components. For example, there is a large body of work in modeling and analyzing TCP. However, the gap between a TCP connection model and an Internet model is huge: many studies focus on simulating TCP over a single bottleneck link, traffic behavior under aggregation is not obvious [22], and there are many other issues (e.g. *Q3*, *Q4*, *Q5*) to deal with.

This situation is similar to that for modeling memory usage in a personal computer. There, the approach — since Denning's working-set model thirty years ago — has been to use trace simulations of a single program, whereas a program usually executes in a multiprogramming mix, and a trace depends on the instruction set, compiler version and operating system. Rather than construct a memory model from smaller models for program references, memory management, multiprogramming interaction, etc., I adopted a top-down approach [23] by using Little's Law [24] to derive some basic equations relating various performance measures, and completing the set of equations with approximate relationships specific to the problem (in this case, page replacement). The idea is that, however chaotic the dynamics and intractable the distributions, we can avoid modeling them by exploiting the robustness of Little's Law. Such a topdown approach can drastically reduce the number of parameters needed to specify the model (*Q7*).

A key result in this memory model says that, if M is the size of main memory and n is the average number of times a page is read by a workload, then

$$\frac{M}{1-(1-\frac{1}{n})^2} = M^*, \qquad (E1)$$

where M^* is a constant with respect to M and n. This was derived without modeling the page replacement in detail. I believe it is possible that we can similarly cut through the fluctuations in the routing tables (*Q6*), the fractal nature of the traffic (*Q5*), the feedback mechanism in TCP variants (*Q4*) etc. to derive some analogous relationships.

The equation ($E1$) was validated with a Winstone99 benchmark that ran Netscape Navigator,

Microsoft Access, Excel, PowerPoint and Word in a multiprogramming mix on Windows 2000. How can a successful validation be possible when ($E1$) is derived without modeling the hardware, the memory manager, the applications, etc.? The effects of these are in fact represented by M^*, whose value is determined by the configuration and the workload. A similar encapsulation of the details and diversity (*Q3, Q4, Q5*) in the Internet may be possible.

The significance of ($E1$) is that it specifies how changing the memory size affects the number of times pages are read. One can interpret this as the effect of memory allocation on disk traffic, and thus derive rules for fairly reclaiming pages from processes when memory is tight, and choosing which process to swap out in a crunch. If we can characterize different traffic flows (*Q5*) in a similar way, a router may be able to use that characterization to fairly allocate bandwidth (*Q8*) and squelch misbehaving traffic when signs of a congestion collapse are detected.

M^* is similar in nature to a calibration factor f that I used in a load-sharing model for multimedia-on-demand [25]. The issue was whether and how a cluster of cheap low-end servers connected by a switch can equal one expensive high-end server in performance (specifically, waiting time) by sharing the workload. There, f was used to hide details of the application interface, protocol implementation, server hardware, bandwidth reservation, etc. Although these details were not explicitly modeled, the equations led to the following necessary condition on the cluster size N_{server}:

$$N_{server} \leq \max\left(\frac{B_{switch}}{B_{link}}, \frac{1}{2} + \sqrt{\frac{1}{4} + \frac{B_{switch}}{(1 - \frac{N_{term}}{N_{stream}}\frac{T_{active}}{T_{sleep}+T_{active}})N_{stream}B_{stream}}}\right), \qquad (E2)$$

where B_{switch}, B_{link} and B_{stream} are bandwidths, N_{term} and N_{stream} are cardinalities, T_{active} and T_{sleep} are durations, and all these are input parameters. In other words, one can omit many details from a model, and still get results that are useful (in this case, for capacity planning).

There were three difficulties in formulating that model: the queues do not form a separable network [24], so the many results known for such networks were not useful; load sharing hinges on temporary imbalances in workload, thus making the application of steady state modeling techniques nontrivial; and load sharing is meaningful only when queue servers have high utilization, for which waiting times have high variances. The success of that model makes *Q5*, *Q6* and *Q8* less intimidating.

The fact that the Internet does not form a separable queueing network (*Q5*) is irrelevant, since

separable networks are notoriously hard to analyze — even proving obvious properties like throughput concavity is nontrivial. One way of tackling this analytic intractability is to use approximations of the performance measures that are easier to analyze, then interpret the results as approximate relationships of the exact performance measures [26]. For example, in a closed multiclass separable network,

$$\frac{S_{mr}}{X_k}\frac{\partial X_k}{\partial S_{mr}} = -\delta_{rk}T_{mrk} - d_m Q_{mk} T_{mrk} - \sum_{c \neq k} \frac{S_{mr}}{X_c}\frac{\partial X_c}{\partial S_{mr}} \sum_{i=1}^{M} d_i Q_{ik} T_{ick}, \qquad (E3)$$

$$\text{where } T_{mrk} = \frac{B_{mr} Q_{mr}}{\sum_{i=1}^{M} B_{ik} Q_{ik}(1 + d_i Q_{ik})}, \quad B_{mr} = \frac{1}{1 + d_m \frac{U_{mr}}{N_r}},$$

k, r and c are class indices, m and i are queue indices, X is throughput, S is service time, Q is queue length, U is queue utilization, d indicates whether a queue has a delay server, N_r is the number of jobs in class r, and $\delta_{rk} = 1$ if $r = k$ and 0 otherwise. The equation (E3) thus decomposes the effect of service time S_{mr} on throughput X_k into a first-order effect $-\delta_{rk}T_{mrk}$ that vanishes unless class k visits queue m, a second-order effect $-d_m Q_{mk} T_{mrk}$ which affects every class that visits the queue (even if service time S_{mk} for that class is unchanged), and a third-order feedback effect from all other classes at all queues. The factor B is surely an artifact of the approximation, a price paid for such a neat breakdown of the effect into service, queueing and feedback. Perhaps a similar decomposition is possible for how router characteristics can affect throughput (*Q5*).

For the separable network, the approximation provided expressions that could be used to analyze an anomaly — speeding up a server in a multiclass network can reduce the throughput of some classes. This is the role I see for an Internet model — helping us analyze and understand its behavior — rather than predicting performance (*Q1*). To see the significance of such a model, consider: every parent wants to be able to analyze and understand (and control!) a child's behavior, without hoping that it can be predicted.

The behavior of Internet traffic is partly determined by contention for link bandwidth and a router's buffer space and processor time. The behavioral analysis would be easier if we could decouple the forces to study their interaction.

Such a decoupling can be done in the case of real-time transactions [27]. One can identify three forms of contention: resource contention, which determines waiting time for compute cycles and disk transfers; data contention, which determines waiting time for the read/write locks that

must be acquired before accessing data; and time contention, which determines the probability of missing a deadline. For a particular workload, one can measure these forms of contention with three metrics W_{RC}, W_{DC} and W_{TC} respectively. Specifically, W_{RC} is the average number of transactions that are not waiting for data locks (and therefore free to contend for resources), and

$$W_{DC} = \frac{L^2 M}{G} \ , \tag{E4}$$

where L is the number of locks requested by each transaction, M is the multiprogramming level, and G is the total number of locks in the database. The interaction between resource and data contention, for example, is given by

$$W_{RC} = \frac{2}{3}\left(\frac{(L+1)(6+4(L-1))}{4(L+1)+W_{DC}} - (L-1)\right)M \ . \tag{E5}$$

There is no contradiction in the fact that two forces are not independent but can be decoupled: In a Jacksonian queueing network, for example, the queues are clearly not independent, but their joint density is just the product of their marginal densities [24]. Can the forces shaping Internet traffic be decoupled analytically?

A metric like W_{DC} combines the input parameters into a workload that measures the stress on a system. As W_{DC} increases, the waiting time for locks escalates (transactions wait for locks that are held by transactions who are also waiting for locks) and

$$\text{transaction throughput drops when} \quad W_{DC} > 2. \tag{E6}$$

It does not make sense to drive the system to such a state, so $(E4)$ and $(E6)$ identify the part of parameter space that the model should concentrate on. The first cut of an Internet model should focus on the pre-congestion region, and it would help if the combination of input parameters $(Q7)$ that defines this region can be similarly specified by a workload.

The time contention workload W_{TC} is defined through considering the probability p_{miss} that a transaction misses a deadline, and is given by

$$W_{TC} = \frac{f(0)L^3 M g(1)}{GS} \ , \tag{E7}$$

where f is a density function for the deadline (slack), S the mean for f, and $g(1)$ measures the execution time of a transaction. The appropriateness of this definition was validated when measurements of W_{TC} and p_{miss} from a simulator showed that they were linearly related. I call this

analytic validation: a relationship derived with approximations is confirmed by measurements. In contrast, many performance models are validated by comparing measurements to performance predictions calculated with the models.

Performance models for computer systems are invariably too complicated for exact analysis. (Some stochastic models, even after simplification and approximations, take longer to solve than running the simulation.) Since approximations are necessary, one must check that the properties derived with the model are not those of the approximations, but properties of the system being modeled — hence the need for analytic validation.

The cubic relationship in $(E7)$ is striking, and argues strongly that a real-time transaction should set only a minimal number of data locks. It is known that the probability of dropping a packet at a router is nonlinear with respect to the buffer size and the number of active connections; what is the form of this relationship?

When multiple transaction classes — with different deadline distributions, data requirements and resource usage — run together, they affect each other's probability of missing their deadlines. Extending $(E7)$ to such a workload gives

$$W_{TC(j)} = \frac{f_j(0)L_j}{S_j} \frac{\sum_{k=1}^{C} L_k^2 M_k g_k(1)}{G} \ , \qquad (E8)$$

where j and k are class indices. From the linear relationship between $W_{TC(j)}$ and $p_{miss(j)}$, we then get

$$\frac{p_{miss(j)}S_j}{f_j(0)L_j} = \frac{p_{miss(k)}S_k}{f_k(0)L_k} \quad \text{for all } j \text{ and } k, \qquad (E9)$$

which is a possible definition of fairness among transaction classes. (Note the absence of M_k and g_k.) Can fairness among traffic flows be similarly defined $(Q8)$?

Incidentally, a system administrator may find a miss probability of $p_{miss} \geq 0.3$, say, unacceptable. A requirement like $p_{miss} < B$ can be interpreted as a statement about the value of a transaction. I believe it is possible to similarly interpret a bound on the probability of a packet loss in terms of some utility function for an application [28].

Like the constraint on the parameter space that follows from $(E6)$, a practical requirement such as $p_{miss} < B$ narrows the scope of a model, thus helping to make it tractable. This is the case with a model for the IEEE 802.11 medium access protocol [29]. This protocol specifies how multiple

transmitters in a wireless local area network can share the bandwidth, and the crucial performance measure is the probability $p_{collide}$ that a transmission collides with another transmission. While it may be an interesting intellectual exercise to formulate a model that is accurate for large values of $p_{collide}$ (> 0.5, say), such a model is of no engineering significance because the protocol is not designed for such high bandwidth contention.

Note, however, that restricting the model to small $p_{collide}$ does not mean that the workload is low, since a wireless channel can saturate — i.e. maximum possible throughput under the protocol is reached — even with a small $p_{collide}$. This is so because, aside from collisions, throughput is also determined by backoff, which is the delay in retransmission required by the protocol when there is a collision. The backoff is controlled by a window size W. For TCP, Padhye et al. have shown that throughput X_{TCP} is similarly determined by a maximum window W_{max} and the probability p for a packet loss through

$$X_{TCP} = \min\left(\frac{W_{max}}{T_{RTT}}, \frac{1}{T_{RTT}\sqrt{\frac{2bp}{3}} + T_0 \min\left(1, 3\sqrt{\frac{3bp}{8}}\right)p(1+32p^2)}\right) ,$$

where T_{RTT} is the round-trip time, T_0 is a timeout, and b is a parameter for increasing the window size [19].

In the 802.11 model, it is possible to go one step further and show that the optimum window size is

$$W_{opt} \approx N_{trans}\sqrt{L_{packet}} , \qquad (E10)$$

where N_{trans} is the number of transmitters and L_{packet} is the size of a packet. This relationship ($E10$) can be used to help 802.11 adapt its window size. A major concern in the design and analysis of TCP — used extensively by other protocols on the Internet — is how its window size should be adjusted. If one can obtain the analog of ($E10$), relating TCP's optimum window size to packet lengths, buffer sizes, active connections, etc., that may point the way to a window control robust enough to prevent a congestion collapse.

The situation for 802.11 is much simpler, and one can explicitly identify the onset of congestion, when throughput saturates, as at

$$N_{trans} \approx \frac{1}{\lambda L_{packet}}\left(1 - \frac{1}{3 + W\lambda L_{packet}}\right), \qquad (E11)$$

where each transmitter sends packets at rate λ. As with the real-time transaction model, ($E11$) was validated by identifying the congestion point with a simulator.

But the simulator is needed not at the end of the modeling exercise (to validate the results), but at the beginning. While modeling, one is often faced with a choice of approximations — e.g. between one that is simple and another that is possibly more accurate, but also likely to make the model less tractable — and the simulator can help us decide if the loss in accuracy is too much for the gain in simplicity.

In this regard, recent progress in modeling Internet topology [30, 31] and traffic [32] is encouraging. These models can be used to generate Internet-like topologies and run traffic over them in a multi-protocol simulator (e.g. VINT [14]). Unlike the memory model, which uses Little's Law to brush aside all sorts of details but can still be validated with a real machine running an actual workload, an Internet model must rely heavily on a simulator for validation.

Acknowledgement

Microsoft and Windows 2000 are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

References

1. L. Lamport and N.A. Lynch. Distributed computing: models and methods. In *Handbook of Theoretical Computer Science*, 1159–1199, van Leeuwen (ed.), MIT Press, Cambridge (1990).
2. P.G. Neumann. *Computer-Related Risks*. Addison-Wesley, New York (1995).
3. S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the Internet. IEEE/ACM Trans. Networking 7, 4(Aug. 1999), 458–472.
4. Ö. Babaoğlu and K. Marzullo. Consistent global states of distributed systems. In *Distributed Systems*, 55–96, S. Mullender (ed.), Addison-Wesley, New York (1993).
5. H.M. Beisner, J.G. Rudd and R.H. Benner. Real-time APL prototype of a GPS system. Proc. Int. Conf. on APL, Lancaster, UK (July 1996), 31–39.

6. Interplanetary Internet Research Group Charter, `http://www.irtf.org/charters/ipnrg.html`.

7. A.K. Singh. On the validity of the global time assumption. Proc. Int. Conf. Distributed Computing Systems, Arlington, USA (May 1991), 282–289.

8. U. Abraham and S. Ben-David. On the limitation of the global time assumption in distributed systems. Proc. Int. Workshop Distributed Algorithms, Delphi, Greece (Oct. 1991), 1–8.

9. D.A. Peled, V.R. Pratt and G.J. Holzmann. *Partial Order Methods in Verification*. American Mathematical Society, Providence (1997).

10. V. Pratt. Modeling concurrency with partial orders. Int. J. Parallel Prog. 15, 1(1986), 33–71.

11. L. Lamport. Time, clocks, and the ordering of events in a distributed system. Comm. ACM 21, 7(Nov. 1978), 558–565.

12. L. Lamport. The mutual exclusion problem: Part I — A theory of interprocess communication. J. ACM 33, 2 (Apr. 1986), 313–326.

13. Y.C. Tay and X.B. Shen. Using concurrent states to rebuild the theory for distributed computing. Proc. Int. Conf. Recent Advances in Mathematical Sciences, Kharagpur, India (Dec. 2000).

14. V. Paxson and S. Floyd. Why we don't know how to simulate the Internet. Proc. Winter Simulation Conf., Atlanta, USA (Dec. 1997), 1037–1044.

15. R. Jain and S.A. Routhier. Packet trains — measurements and a new model for computer network traffic. IEEE J. Selected Areas in Communications 4 (Sept. 1986), 986–995.

16. V. Paxson and S. Floyd. Wide area traffic: The failure of Poisson modeling. IEEE/ACM Trans. Networking 3, 3(June 1995), 226–244.

17. M.E. Crovella and A. Bestavros. Self-similarity in World Wide Web traffic: Evidence and possible causes. Proc. ACM SIGMETRICS Conf., Philadelphia, USA (May 1996), 160–169.

18. S. Bajaj, L. Breslau and S. Shenker. Is service priority useful in networks? Proc. ACM SIGMETRICS Conf., Madison, USA (June 1998), 66–77.

19. J. Padhye, V. Firoiu, D. Towsley and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. Proc. ACM SIGCOMM Conf., Vancouver, Canada (Aug. 1998),

303–314.

20. C. Labovitz, G.R. Malan and F. Jahanian. Internet routing instability. Proc. ACM SIGCOMM Conf., Cannes, France (Sept. 1997), 115–126.

21. L. Breslau and S. Shenker. Best-effort versus reservations: A simple comparative analysis. Proc. ACM SIGCOMM Conf., Vancouver, Canada (Aug. 1998), 3–16.

22. R. Morris and D. Lin. Variance of aggregated Web traffic. Proc. IEEE INFOCOM, Tel Aviv, Israel (Mar. 2000), 360–366.

23. Y.C. Tay. How memory size and memory allocation affect disk reads. National Univ. of Singapore Math. Dept. Research Report No. 764 (Nov. 1999).

24. R. Jain. *The Art of Computer Systems Performance Analysis*. Wiley, New York (1991).

25. Y.C. Tay and HH Pang. Load sharing in distributed multimedia-on-demand systems. IEEE Trans. Knowledge and Data Engineering 12, 3 (May 2000), 410–428.

26. Y.C. Tay. An approach to analyzing the behavior of some queueing networks. Operations Research 40 (May 1992), S300-311.

27. Y.C. Tay. Some performance issues for transactions with firm deadlines. Proc. Real-Time Systems Symposium, Pisa, Italy (Dec. 1995), 322–331.

28. S. Shenker. Fundamental design issues for the future Internet. IEEE J. Selected Areas in Communications 13, 7 (Sept. 1995), 1176–1188.

29. Y.C. Tay and K.C. Chua, A capacity analysis for the IEEE 802.11 MAC protocol. ACM/Baltzer Wireless Networks, to appear.

30. K.L. Calvert, M.B. Doar and E.W. Zegura. Modeling Internet topology. IEEE Communications Magazine (June 1997), 160–163.

31. M. Faloutsos, P. Faloutsos and C. Faloutsos. On power-law relationships of the Internet topology. Proc. ACM SIGCOMM Conf., Cambridge, USA (Aug. 1999), 251–262.

32. A. Feldmann, A.C. Gilbert, P. Huang and W. Willinger. Dynamics of IP traffic: a study of the role of variability and the impact of control. Proc. ACM SIGCOMM Conf., Cambridge, USA (Aug. 1999), 301–313.