

# Design and Analysis of Dynamic Processes: A Stochastic Approach (Invited Paper)

Eli Upfal

Computer Science Department, Brown University  
Box 1910, Providence, RI 02912.  
E-mail: [eli@cs.brown.edu](mailto:eli@cs.brown.edu)  
<http://www.cs.brown.edu/people/eli>

**Abstract.** Past research in theoretical computer science has focused mainly on static computation problems, where the input is known before the start of the computation and the goal is to minimize the number of steps till termination with a correct output. Many important processes in today's computing are dynamic processes, whereby input is continuously injected to the system, and the algorithm is measured by its long term, steady state, performance. Examples of dynamic processes include communication protocols, memory management tools, and time sharing policies. Our goal is to develop new tools for the design and analyzing the performance of dynamic processes, in particular through modeling the dynamic process as an infinite stochastic processes.

## 1 Introduction

Rigorous analysis of the dynamic performance of computer processes is one of the most challenging current goals in theory of computation. Past research in theoretical computer science has focused mainly on static computation problems. In static computation the input is known at the start of the computation and the goal is to minimize the number of steps till the process terminates with a correct output. Many important processes in today's computing are dynamic processes, whereby input is continuously injected to the system, and the algorithm (which is not supposed to terminate at all) is measured by its long term (steady state) performance. Examples of dynamic computer processes include:

- Contention resolution protocols.
- Routing and communication algorithms.
- Memory management tools such as caching and paging.
- Time sharing and load balancing protocols.

### 1.1 Performance measures for dynamic processes

In evaluating the performance of dynamic algorithms one needs to distinguish between algorithms that satisfy all incoming requests and algorithms that may

drop requests. In the first case the primary performance measure of interest is the algorithm's stability conditions. Roughly speaking, a system is stable if in the long run the number of new arriving requests is no larger than the number of requests processed by the system. The goal of the analysis here is to characterize the (most general) input conditions (deterministic or stochastic) under which the system is stable. The corresponding measure for algorithms that may drop requests is the number of requests (as a function of the incoming stream of requests) that the algorithm successfully satisfied (or equivalently, the number of requests that the algorithm needs to reject in order to keep the system stable). A second criteria, which is important for both type of algorithms is speed, measured by the (maximum or expected) time to satisfy a request.

An algorithm is usually analyze in terms of its (worst case or average) performance with respect to a given class of inputs. However, in many dynamic algorithms it is useful to study instead the *competitive ratio* performance of the algorithm. Competitive ratio analysis compares the performance of the dynamic (on-line) algorithm to the performance of an "off-line" algorithm that "knows" in advance the whole sequence of requests. The quality of an on-line algorithm is measured by the maximum, over all possible input sequences, of the ratio between its performance and the performance of an off-line on the same input sequence. Thus, competitive analysis measures the quality of the algorithm with respect to the optimal one, rather than measuring the actual performance of the system. This measure is particular useful in cases when no algorithm can perform well on all instances of the set of inputs.

## 1.2 Stochastic analysis

As in the case of static algorithms, randomness is introduced into dynamic computation through either the algorithm, the input or both. Many interesting dynamic protocols are random, a well known example is the Aloha contention resolution protocol [32] which is used in the Ethernet and other similar applications. An execution of a dynamic random algorithm, even on a fixed input sequence, defines an infinite stochastic process in which a state at a given step depends on the history of the process. Analysis of such a process requires different approach and tools from the ones used for analyzing the finite execution of a randomized static computation.

Worst case analysis rarely gives an interesting insight on the actual performance of a dynamic algorithm. A worst case adversary can generate extremely hard sequences of requests, and the performance of the algorithm on these "pathological" cases does not accurately represent the efficiency of the algorithm. To offset the affect of rare cases it is useful to analyze the performance of dynamic systems under some stochastic assumptions on the stream of inputs. Such assumptions are more realistic in the dynamic setting, in particular when requests are originated by a number of independent processors, than in the case of static analysis. The stochastic process that control the stream of requests might be stationary, periodic, or even bursty. The goal is to obtain results that are valid under the weakest set of assumptions. The advantage and practicality

of this approach has been well demonstrated by the achievements of queueing theory [26]. Our goal is to applied similar techniques to dynamic computer processes that do not fit the queueing theory settings. Competitive ratio analysis is another alternative to standard worst case analysis. However, even in the case of competitive analysis, stochastic input may lead to better evaluation of the algorithm’s performance. One example is greedy load balancing. This procedure is very efficient in practice, while its worst case competitive ratio is high. On the other hand it can be shown that under some stochastic assumptions the average competitive ratio of greedy load balancing is very low [5], thus giving one possible explanation for the “real-life” performance of this procedure.

Stochastic analysis of dynamic processes builds on the rich theory of stochastic processes, in particular queueing theory, and the theory of stationary processes. However, in many cases one needs to develop new tools to address the specific problems pose by computer related processes, which are discrete and involved complicated dependency conditions.

## 2 Dynamic Balanced Allocations

In [5] we studied a surprising combinatorial phenomena which we term *balanced allocations*. In the dynamic version of balanced allocation we have  $n$  boxes and  $n$  balls. In each step one ball, chosen uniformly at random, is removed from the system, and a new ball appears. The new ball comes with  $d$  possible destinations, chosen independently at random, and it is placed into the least full box, at the time of the placement, among the  $d$  possible destinations.

The case  $d = 1$  corresponds to the classic occupancy problem. It is easy to show that in the stationary distribution, with high probability the fullest box has  $\Theta(\log n / \log \log n)$  balls. The analysis of the case  $d \geq 2$  is significantly harder, since the locations of the current  $n$  balls might depend on the locations of balls that are no longer in the system. A surprising result which we proved in [5] shows that for  $d \geq 2$ , in the stationary distribution, with high probability no box contains more than  $\ln \ln n / \ln d + O(1)$  balls. Thus, an apparent minor change in the random allocation process results in an exponential decrease in the maximum occupancy per location. Several recent works have built on our original result. Mitzenmacher [34] studied a continuous time variant of our model in which balls arrive according to a Poisson process and are removed with an exponential distribution. Czumaj and Stemann [14] showed that a more efficient variant of our placement procedure results in similar maximum load.

The dynamic balanced allocation process has a number of algorithmic applications:

**Dynamic dictionary.** The efficiency of a hashing technique is measured by two parameters: the expected and the maximum access time. Our approach suggests a simple hashing technique, similar to hashing with chaining. We call it *2-way chaining*. It has  $O(1)$  expected, and  $O(\log \log n)$  maximum access time. We use two random hash functions. The two hash functions define two possible entries in the table for each key. The key is inserted to the least full location, at

the time of the insertion. Keys in each entry of the table are stored in a linked list. The expected insertion and look-up time is  $O(1)$ , and with high probability the maximum access time is  $\ln n \ln n / \ln 2 + O(1)$ , versus the  $\Theta(\log n / \log \log n)$  time when one random hash function is used.

**Dynamic Resource Allocation.** Consider a scenario in which a user or a process has to choose on-line between a number of identical resources (choosing a server to use among the servers in a network; choosing a disk to store a directory; etc.). To find the least loaded resource, users may check the load on all resources before placing their requests. This process is expensive, since it requires sending an interrupt to each of the resources. A second approach is to send the task to a random resource. This approach has minimum overhead, but if all users follow it, the difference in load between different servers will vary by up to a logarithmic factor. The balanced allocation process suggests a more efficient solution. If each user samples the load of two resources and sends his request to the least loaded, the total overhead is small, and the load on the  $n$  resources varies by only a  $O(\log \log n)$  factor, an exponential improvement over the pure random allocation.

### 3 Dynamic Flow Control for Packet Routing With Bounded Buffers

Most theoretical work on routing algorithms has focused on static routing: A set of packets is injected into the system at time 0, and the routing algorithm is measured by the time it takes to deliver all the packets to their destinations, assuming that no new packets are injected in the meantime (see Leighton [29] for an extensive survey). In practice however, networks are rarely used in this “batch” mode. Most real-life networks operate in a *dynamic* mode whereby new packets are continuously injected into the system. Each processor usually controls only the rate at which it injects its own packets and has only a limited knowledge of the global state. This situation is better modeled by a stochastic paradigm whereby packets are continuously injected according to some inter-arrival distribution, and the routing algorithm is evaluated according to its long term behavior. goal is to develop algorithms that perform close to optimal for a variety of inter-arrival distributions.

Several recent articles have addressed the dynamic routing problem, in the context of packet routing on arrays [28,33,27,8], on the hypercube and the butterfly [38] and general networks [37]. Except for [8], the analyzes in these works assumes a Poisson arrival distribution and requires unbounded queues in the routing switches (though some works give a high probability bound on the size of the queue used [28,27]). Unbounded queues allow the application of some tools from queuing theory (see [20,21]) and help reduce the correlation between events in the system, thus simplifying the analysis at the cost of a less realistic model.

In [9] we focused on the design and analysis of efficient flow control mechanism for dynamic packet routing in networks with bounded buffers at the switching nodes, a setting that more accurately models real networks. Our goal was to

build on the vast amount of work that has been done for static routing in order to obtain results for the dynamic setting. In [9] we developed a general technique in which a static algorithm for a network with bounded buffers is augmented with a simple flow control mechanism to obtain a provably efficient dynamic routing algorithm on a similar network with bounded buffer. The crucial step in [9] is a general theorem showing that any communication scheme (a routing algorithm and a network) that satisfies a given set of conditions, defined only with respect to a *finite history* is stable up to a certain inter-arrival rate. The theorem also bounds the expected routing time under these conditions.

This technique was applied in [9] to a number of dynamic routing scenarios on the butterfly network. In particular we gave the first provable routing protocol for the butterfly network with bounded buffer that is stable for injection rate that is within a constant factor of the hardware bandwidth. Similar results are obtained in [11] for dynamic routing on the mesh. Another result in [9] shows that our general technique can be applied to the *adversarial* input model of [7]. In that model, instead of probabilistic assumptions on the input there is an absolute bound on the number of packets generated in any time interval and must traverse any particular edge. Our technique gives a dynamic randomized routing algorithm for a butterfly with bounded buffer that is optimal (up to constant factors) in that model [9].

## 4 Dynamic Virtual Circuit Allocation

Communication protocols for high-speed high bandwidth networks (such as the ATM protocol) are based on *virtual circuit switching*. The speed of the network does not allow for on-line routing of individual packets. Instead, upon establishing a connection, bandwidth is allocated along a path connecting the two endpoints for the duration of the connection. These “virtual circuits” are set up on a per-call basis and are disconnected when the call is terminated. Efficient utilization of the network depends on the allocation of virtual circuits between pairs of nodes so that no link is overloaded beyond its capacity.

As in other routing problems we distinguish between a static and a dynamic version. In the static version all the requests are given at once and must be simultaneously satisfied. In practice networks are rarely used in the “batch mode” modeled by the static problem. Real-life network performance is better modeled by a dynamic process whereby requests for connection are continuously arriving at the nodes of the network. A connection has a duration time, and once the communication has terminated its bandwidth can be used for another connection. A dynamic circuit switching problem is thus characterized by a number of parameters: the network topology, the channels physical bandwidth, the injection rate distribution of new requests and the distribution of the duration of connections.

Using a random walk approach we develop in [10] a simple and fully distributed protocol for dynamic path selection on bounded degree expander graphs. For the analysis we adopt the stochastic model assumed in the design of most

long-distance telephone networks [24]. Requests arrives according to a Poisson process, and the duration of a connection is exponentially distributed. Our solution achieves an almost optimal utilization of the edges under this stochastic model.

Our approach differs from the work on admission control [3,23,4] in that we do not reject requests. All requests are eventually satisfied in our model, but not immediately. In contrast, in the admission control model a request is either immediately satisfied or it is rejected. Our approach better models most types of computer communication, while the admission control approach is a better model for human (telephone) communication.

## 5 Stochastic Contention Resolution with Short Delays

One of the few processes that has been studied in the past in dynamic settings are contention resolution protocols and in particular the backoff based solutions.

The contention resolution problem is best defined in terms of multiple access channels. There are  $n$  senders and  $m$  receivers. at each of a series of time steps, one or more senders may generate a *packet*. A packet when generated has a *destination* — a unique receiver to which it must be delivered. Any sender may attempt to send a packet to any receiver at any step, but a receiver may only receive one packet in a step. If a receiver is sent more than one packet in a step (a *collision*), all packets sent to that receiver are lost and the senders notified of the loss. The senders must then try to send these packets again at a future step. There is no explicit communication between the senders for coordination the transmissions; the only information that senders have is the packet(s) they have waiting for transmission, and the history of losses. A packet can only be transmitted directly from its sender to its receiver; intermediate hops are disallowed. The case  $m = 1$  is a classical instance of sharing a common resource such as a bus or an Ethernet channel (the shared bus is modeled by the single “receiver”). The case with  $m = n$  has been studied recently in the static setting under the name *Optically Connected Parallel Computer*, or OCPC [31].

Of primary interest, in the dynamic setting, is whether a protocol can sustain packet arrivals at some rate without *instability*. For stable protocols, two quantities are of interest: (1) the *maximum arrival rate*  $\lambda$  that can be sustained stably (measured as a fraction of the hardware capacity), and (2) the *delay*, defined to be the maximum over all senders of the expected number of steps from the generation of a packet to its delivery, in the steady state. Delay is of particular importance in high-speed communications applications such as video and ATN networks.

Most previous analysis focused on backoff protocols. The binary exponential backoff protocol used in the Ethernet was proposed by Metcalfe and Boggs [32]. Aldous [1] showed that for any positive constant  $\lambda$ , binary exponential backoff is unstable if the number of senders is infinite. Kelly [25] showed that any polynomial backoff protocol is unstable for infinitely many senders. Håstad, Leighton and Rogoff [19] studied systems with a finite number of senders. they showed that

binary exponential backoff is unstable for  $\lambda$  slightly larger than 0.567 even for a system with a finite number of senders, and that polynomial backoff protocol is stable for any  $\lambda < 1$  and for any finite number of senders.

A major drawback of all the backoff protocols is the long expected delay in delivering packets (at least linear in the number of senders) as was shown in [19]. In [36] we present the first contention resolution protocol with  $o(n)$  expected delay, our protocol is stable for a constant injection rate and the expected delay is logarithmic in the number of senders. Two recent works [35,16] have build on our results to obtain constant expected delay.

## References

1. D. Aldous. Ultimate instability of exponential back-off protocol for acknowledgment based transmission control of random access communication channels. *IEEE Transactions on Information Theory*, Vol. IT-33, 1987, pp. 219–223.
2. M. Andrew, B. Awerbuch, A Fernandez, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. *Proceedings of the 37th Annual Symp. on Foundations of Computer Science*, pages 380–389, 1997.
3. B. Awerbuch, Y. Azar, and S. Plotkin. Throughput competitive online routing. In *Proceedings of the 34th IEEE Conference on Foundations of Computer Science*, pages 32–40, 1993.
4. B. Awerbuch, R. Gawlick, T. Leighton, and Y. Rabani. On-line admission control and circuit routing for high-performance computing and communication. *Proceedings of the 35th Annual Symp. on Foundations of Computer Science*, October 1994, pp 412–423.
5. Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 593–602, 1994.
6. Y. Azar, J. Naor, and R. Rom. The competitiveness of on-line assignments. In *Proceedings of the 3rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 203–210, 1992.
7. A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan and D. P. Williamson Adversarial queuing theory. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 376–385, 1996.
8. A. Z. Broder and E. Upfal. Dynamic Deflection Routing in Arrays. *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 348–355, 1996.
9. A.Z. Broder, A.M. Frieze, and E. Upfal. “A general approach to dynamic packet routing with bounded buffers.” *Proceedings of the 37th IEEE Symp. on Foundations of Computer Science*. Burlington, 1996, pp. 390–399.
10. A.Z. Broder, A.M. Frieze, and E. Upfal. “Static and dynamic path selection on expander graphs: a random walk approach”. *Proceedings of the 29th ACM Symp. on Theory of Computing*. El Paso, 1997.
11. A. Broder, A. Frieze, and E. Upfal. “Dynamic packet routing on arrays with bounded buffers”. *Third Latin American Symposium on Theoretical Informatics - LATIN '98* Campinas, Brazil. April 1998. In *Springer-Verlag Lecture Notes in Computer Science 1380*, pp 273–281, 1998.
12. R.L. Cruz. A calculus for network delay, part I: Network elements in isolation. *IEEE Trans. on Information Theory*, Vol. 37, pages 114–131, 1991

13. R.L. Cruz. A calculus for network delay, part II: Network analysis in isolation. *IEEE Trans. on Information Theory*, Vol. 37, pages 132–141, 1991
14. A. Czumaj and V. Stemann. “Randomized Allocation Processes”. Preprint, 1997.
15. M. Dietzfelbinger, A. R. Karlin, K. Mehlhorn, F. Meyer auf der Heide, H. Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. In *Proceedings of the 29th IEEE Conference on Foundations of Computer Science*, pages 524–531, 1988.
16. L.A. Goldberg and P.D. MacKenzie. Contention Resolution with Guaranteed Constant Expected Delay. Preprint, 1997.
17. J. Goodman, A.G. Greenberg, N. Madras, and P. March. Stability of Binary Exponential Backoff. *J. of the ACM*, Vol. 35 (1988) pages 579–602.
18. A.G. Greenberg, P. Flajolet, and R.E. Ladner. Estimating the Multiplicities of Conflicts to Speed Their Resolution in Multiple Access Channels. *J. of the ACM*, Vol. 34, 1987, pp. 289–325.
19. J. Håstad, T. Leighton, and B. Rogoff. Analysis of backoff protocols for multiple access channels. *Proceedings of the 19th ACM Symp. on Theory of Computing*, 1987, pp. 241–253.
20. M. Harcol-Balter and P. Black. Queuing analysis of oblivious packet routing networks. *Procs. of the 5th Annual ACM-SIAM Symp. on Discrete Algorithms*. Pages 583–592, 1994.
21. M. Harcol-Balter and D. Wolf. Bounding delays in packet-routing networks. *Procs. of the 27th Annual ACM Symp. on Theory of Computing*, 1995, pp. 248–257.
22. R. M. Karp, M. Luby, and F. Meyer auf der Heide. Efficient PRAM simulation on a distributed memory machine. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, pages 318–326, 1992.
23. A. Kamath, O. Palmon, and S. Plotkin. Routing and Admission Control in General Topology Networks with Poisson Arrivals. *SODA 96*.
24. F.P. Kelley. Blocking probabilities in large circuit-switching networks. *Adv. Appl. Prob.*, 18:573–505, 1986.
25. F.P. Kelly. Stochastic models of computer communication systems. *J. Royal Statistical Soc. B*, Vol. 47, 1985, pp. 379–395.
26. L. Kleinrock. *Queueing systems*. Wiley, New York, 1975.
27. N. Kahale and T. Leighton. Greedy dynamic routing on arrays. *Procs. of the 6th Annual ACM-SIAM Symp. on Discrete Algorithms*. Pages 558–566, 1995.
28. T. Leighton. Average case analysis of greedy routing algorithms on arrays. *Procs. of the Second Annual ACM Symp. on Parallel Algorithms and Architectures*. Pages 2–10, 1990.
29. F. T. Leighton. *Introduction to Parallel Algorithms and Architectures*. Morgan-Kaufmann, San Mateo, CA 1992.
30. F.T. Leighton and S. Rao. Circuit switching: a multi-commodity flow based approach. *Proc. of 9th International Parallel Processing Symposium*, 1995.
31. P.D. MacKenzie, C.G. Plaxton, and R. Rajaraman. On contention resolution protocols and associated probabilistic phenomena. *Proceedings of the 26th ACM Symposium on Theory of Computing*, 1994, pp. 153–162.
32. R. Metcalfe and D. Boggs. Ethernet: distributed packet switching for local computer networks. *Communication of the ACM*, Vol. 19, 1976, pp. 395–404.
33. M. Mitzenmacher. Bounds on the greedy algorithms for array networks. *Procs. of the 6th Annual ACM Symp. on Parallel Algorithms and Architectures*. Pages 346–353, 1994.

34. M. Mitzenmacher. Load balancing and density dependent jump Markov processes. *Procs. of the 37th IEEE Annual Symp. on Foundations of Computer Science*, pages 213–222, October 1996.
35. M. Paterson and A. Srinivasan. Contention resolution with bounded delay. *Proc. of the 34th Annual IEEE Symp. on Foundation of Computer Science*, pages 104–113, 1995.
36. P. Raghavan and E. Upfal. Stochastic contention resolution with short delays. *Proc. of 24th ACM Symp. on Theory of Computing*, pages 229–237, 1995.
37. C. Scheideler and B. Voecking. Universal continuous routing strategies. *Procs. of the 8th Annual ACM Symp. on Parallel Algorithms and Architectures*. 1996.
38. G. D. Stamoulis and J. N. Tsitsiklis. The efficiency of greedy routing in hypercubes and butterflies. *Procs. of the 6th Annual ACM Symp. on Parallel Algorithms and Architectures*. Pages 346–353, 1994.