ACM SIGACT News Distributed Computing Column 7

Sergio Rajsbaum*

May 1, 2002

Abstract

The Distributed Computing Column covers the theory of systems that are composed of a number of interacting computing elements. These include problems of communication and networking, databases, distributed shared memory, multiprocessor architectures, operating systems, verification, internet, and the web.

This issue consists of some PODC announcements followed by two papers:

- Announcements about PODC 2002, 2003, 2004 and 2005 by Gil Neiger
- "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services" by Seth Gilbert and Nancy Lynch
- "Topology Control and Routing in Ad hoc Networks: a Survey" by Rajmohan Rajaraman

Many thanks to Gil, Rajmohan, Seth and Nancy for contributing to this issue.

Request for Collaborations: Please send me any suggestions for material I should be including in this column, including news and communications, open problems, and authors willing to write a guest column or to review an event related to theory of distributed computing.

ACM Symposium on Principles of Distributed Computing Announcements

PODC 2002

PODC 2002 will be held July 21–24, 2002 at the Monterey Doubletree in Monterey, California. The hotel's web page can be found at

`http://www.doubletreemonterey.com/`

The Conference Committee for PODC 2002 comprises Aleta Ricciardi (General Chair); Keith Marzullo (Program Chair); Rajeev Joshi (Local Arrangements Chair); Elizabeth Borowsky (Treasurer); Hyunyoung Lee (Publicity Chair); and Gil Neiger (Webmaster).

PODC 2002 will begin with a tutorial on Sunday morning, July 21. It is entitled "On the Cost of Fault-Tolerant Consensus When There Are No Faults" and will be presented by Idit Keidar and Sergio Rajsbaum.

The main technical program will begin on Sunday afternoon, and the conference will run through noon on Wednesday, July 24.

General information about the conference can be found at the following URL:

*Compaq Cambridge Research Laboratory, One Cambridge Center, Cambridge, MA 02142-1612. `Sergio.Rajsbaum@compaq.com, rajsbaum@math.unam.mx`. On leave from Instituto de Matemáticas, UNAM.

http://www.podc.org/podc2002/

That page will contain links to online registration information. Regular registration fees are $115 for students; $280 for ACM members; and $360 for others. Rates will go up after 17 June 2002.

The following regular papers have been accepted for presentation at PODC 2002:

- Nonatomic Mutual Exclusion with Local Spinning; James H. Anderson and Yong-Jik Kim **Winner of Best Student-Paper Award**
- A BGP-Based Mechanism for Lowest-Cost Routing; Joan Feigenbaum, Christos Papadimitriou, Rahul Sami, Scott Shenker
- Searching for a Black Hole in Arbitrary Networks: Optimal Mobile Agent Protocols; Stefan Dobrev, Paola Flocchini, Giuseppe Prencipe, Nicola Santoro
- On Perfectly Secure Communication over Arbitrary Networks; M V N Ashwin Kumar, Pranava Raja Goundan, K Srinathan, C Pandu Rangan
- Nearly Optimal FIFO Buffer Management for DiffServ; Zvi Lotker and Boaz Patt-Shamir
- Optimal Plans for Aggregation; Andrei Broder and Michael Mitzenmacher
- Sequential Composition of Protocols without Simultaneous Termination; Yehuda Lindell Anna Lysyanskaya Tal Rabin
- Fault-tolerant Routing in Peer-to-peer Systems; James Aspnes, Zoe Diamadi, Gauri Shah
- Active Disk Paxos with infinitely many processes; Gregory Chockler, Dahlia Malkhi
- Space-Optimal Multi-Writer Snapshot Objects Are Slow; Panagiota Fatourou, Faith Fich and Eric Ruppert
- Establishing Wireless Conference Calls Under Delay Constraints; Amotz Bar-Noy, Grzegorz Malewicz
- The Inherent Price of Indulgence; Partha Dutta and Rachid Guerraoui
- Efficient algorithms for leader election in radio networks; Tomasz Jurdzinski, Miroslaw Kutylowski, Jan Zatopianski
- Dynamic and Self-Stabilizing Distributed Matching; Subhendu Chattopadhyay, Lisa Higham, Karen Seyffarth
- Evaluating the Running Time of a Communication Round over the Internet; Omar Bakr and Idit Keidar
- Byzantine Agreement Secure Against Faulty Majorities From Scratch; Matthias Fitzi and Daniel Gottesman and Martin Hirt and Thomas Holenstein and Adam Smith
- Viceroy: A Scalable and Dynamic Emulation of the Butterfly; Dahlia Malkhi, Moni Naor and David Ratajczak
- Safe Memory Reclamation for Dynamic Lock-Free Objects Using Atomic Reads and Writes; Maged M. Michael
- Minimal Replication Cost for Availability; Haifeng Yu and Amin Vahdat
- Non-Blocking Timeout in Scalable Queue-Based Spin Locks; Michael L. Scott
- Non-Blocking Steal-half Work Queues; Danny Hendler and Nir Shavit
- Building secure file systems out of Byzantine storage; David Mazières and Dennis Shasha
- General Perfectly Periodic Scheduling; Zvika Brakerski and Aviv Nisgav and Boaz Patt-Shamir
- Optimal Lexicographic Smoothing for Broadband Traffic Multiplexing; Stergios Anastasiadis Peter Varman Jeffrey Scott Vitter Ke Yi

- GS3: Scalable Self-configuration and Self-healing in Wireless Networks; Hongwei Zhang Anish Arora
- Analysis of the Evolution of Peer-to-Peer Systems; Hari Balakrishnan and David Karger and David Liben-Nowell
- Tree Theorem and its Applications; Prasad Jayanti
- Two-Handed Emulation: How to build non-blocking implementations of complex data-structures using DCAS; Michael B. Greenwald
- Squirrel: A decentralized peer-to-peer web cache; Sitaram Iyer and Antony Rowstron and Peter Druschel

The following will be brief presentations at PODC 2002:

- Understanding Perfect Failure Detectors; Mikel Larrea
- Exploiting Precision vs. Efficiency Tradeoffs in Symmetric Replication Environments; Ugur Cetintemel, Peter Keleher
- The Lord of the Rings: Efficient Maintenance of Views at Data Warehouses; D. Agrawal, A. El Abbadi, A. Mostefaoui, M. Raynal, M. Roy
- A highly concurrent group mutual l-exclusion algorithm; K. Vidyasankar
- Dynamic-Sized Lock-Free Data Structures; Maurice Herlihy, Victor Luchangco, Paul Martin, Mark Moir
- Optimally Work-Competitive Scheduling for Cooperative Computing with Merging Groups; Chryssis Georgiou, Alexander Russell and Alex A. Shvartsman
- Assignment-based Partitioning in a Condition Monitoring System; Yongqiang Huang and Hector Garcia-Molina
- Asynchronous Interactive Consistency and its Relation with Error-Correcting Codes; Achour Mostefaoui, Sergio Rajsbaum, Michel Raynal
- Selective Anti-Entropy; Weihbin Zhao and Henning Schulzrinne
- Compact Routing for Average-Case Networks; Kazuo Iwama and Masaki Okita
- How to wait when you are not going to block; Michael Greenwald
- Building Responsive TMR-Based Servers in Presence of Timing Constraints; Paul Ezhilchelvan, Jean-Michel Helary and Michel Raynal
- Early Stopping in Global Data Computation; C. Delporte-Gallet, H. Fauconnier, J.-M. Helary, and M. Raynal
- Random Walk for Self-Stabilizing Group Communication in Ad-Hoc Networks; Shlomi Dolev, Elad Schiller, and Jennifer Welch

This list of accepted papers appears at the following URL:

`http://www.podc.org/podc2002/accepted.html`

PODC 2003

Sergio Rajsbaum of the Compaq Cambridge Research Laboratory will be the program chair for PODC 2003! Other members of the Conference Committee for PODC 2003 are Elizabeth Borowsky (General Chair); Mark Moir (Local Arrangements Chair); and Soma Chaudhuri (Treasurer). PODC 2003 will be held July 13–16, 2003 at the Sheraton Needham in Needham, Massachusetts. The hotel's web page can be found at

`http://www.sheratonneedham.com/`

PODC 2004

The PODC Steering Committee is pleased to announce that PODC 2004 will be held in St. John's, Newfoundland. This will be the thirteenth PODC to be held in Canada but the first to be held there since 1995. Many thanks to **Krishnamurthy Vidyasankar**, Professor at the Memorial University of Newfoundland, who will serve as Local Arrangements Chair. The General Chair will Soma Chaudhuri.

PODC 2005

There has been a tentative offer to host PODC 2005 in Europe. Given that PODC has always been sited in North America, the possibility of holding the conference elsewhere will be discussed at the Business Meeting of PODC 2002.

Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services

Seth Gilbert and Nancy Lynch
Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
`sethg@mit.edu,lynch@theory.lcs.mit.edu`

Abstract

When designing distributed web services, there are three properties that are commonly desired: consistency, availability, and partition tolerance. It is impossible to achieve all three. In this note, we prove this conjecture in the asynchronous network model, and then discuss solutions to this dilemma in the partially synchronous model.

1 Introduction

At PODC 2000, Brewer[1], in an invited talk [2], made the following conjecture: it is impossible for a web service to provide the following three guarantees:

- Consistency
- Availability
- Partition-tolerance

All three of these properties are desirable – and expected – from real-world web services. In this note, we will first discuss what Brewer meant by the conjecture; next we will formalize these concepts and prove the conjecture; finally, we will describe and attempt to formalize some real-world solutions to this practical difficulty.

[1]Eric Brewer is a professor at the University of California, Berkeley, and the co-founder and Chief Scientist of Inktomi.

Most web services today attempt to provide strongly consistent data. There has been significant research designing ACID[2] databases, and most of the new frameworks for building distributed web services depend on these databases. Interactions with web services are expected to behave in a transactional manner: operations commit or fail in their entirety (atomic), transactions never observe or result in inconsistent data (consistent), uncommitted transactions are isolated from each other (isolated), and once a transaction is committed it is permanent (durable). It is clearly important, for example, that billing information and commercial transaction records be handled with this type of strong consistency.

Web services are similarly expected to be highly available. Every request should succeed and receive a response. When a service goes down, it may well create significant real-world problems; the classic example of this is the potential legal difficulties should the E-Trade web site go down. This problem is exacerbated by the fact that a web-site is most likely to be unavailable when it is most needed. The goal of most web services today is to be as available as the network on which they run: if any service on the network is available, then the web service should be accessible.

Finally, on a highly distributed network, it is desirable to provide some amount of fault-tolerance. When some nodes crash or some communication links fail, it is important that the service still perform as expected. One desirable fault tolerance property is the ability to survive a network partitioning into multiple components. In this note we will not consider stopping failures, though in some cases a stopping failure can be modeled as a node existing in its own unique component of a partition.

2 Formal Model

In this section, we will formally define what is meant by the terms *consistent*, *available*, and *partition tolerant*.

2.1 Atomic Data Objects

The most natural way of formalizing the idea of a consistent service is as an atomic data object. Atomic [4], or linearizable [3], consistency is the condition expected by most web services today.[3] Under this consistency guarantee, there must exist a total order on all operations such that each operation looks as if it were completed at a single instant. This is equivalent to requiring requests of the distributed shared memory to act as if they were executing on a single node, responding to operations one at a time. This is the consistency guarantee that generally provides the easiest model for users to understand, and is most convenient for those attempting to design a client application that uses the distributed service. See Chapter 13 of [5] for a more complete definition of atomic consistency.

[2] **A**tomic, **C**onsistent, **I**solated, **D**urable

[3] Discussing atomic consistency is somewhat different than talking about an ACID database, as database consistency refers to transactions, while atomic consistency refers only to a property of a single request/response operation sequence. And it has a different meaning than the **A**tomic in ACID, as it subsumes the database notions of both **A**tomic and **C**onsistent.

2.2 Available Data Objects

For a distributed system to be continuously available, every request received by a non-failing node in the system must result in a response.[4] That is, any algorithm used by the service must eventually terminate. In some ways this is a weak definition of availability: it puts no bound on how long the algorithm may run before terminating, and therefore allows unbounded computation. On the other hand, when qualified by the need for partition tolerance, this can be seen as a strong definition of availability: even when severe network failures occur, every request must terminate.

2.3 Partition Tolerance

The above definitions of availability and atomicity are qualified by the need to tolerate partitions. In order to model partition tolerance, the network will be allowed to lose arbitrarily many messages sent from one node to another. When a network is partitioned, all messages sent from nodes in one component of the partition to nodes in another component are lost. (And any pattern of message loss can be modeled as a temporary partition separating the communicating nodes at the exact instant the message is lost.) The atomicity requirement (§2.1) therefore implies that every response will be atomic, even though arbitrary messages sent as part of the algorithm might not be delivered. The availability requirement (§2.2) implies that every node receiving a request from a client must respond, even though arbitrary messages that are sent may be lost. Note that this is similar to wait-free termination in a pure shared-memory system: even if every other node in the network fails (i.e. the node is in its own unique component of the partition), a valid (atomic) response must be generated. No set of failures less than total network failure is allowed to cause the system to respond incorrectly.[5]

3 Asynchronous Networks

3.1 Impossibility Result

In proving this conjecture, we will use the asynchronous network model, as formalized by Lynch in Chapter 8 of [5]. In the asynchronous model, there is no clock, and nodes must make decisions based only on the messages received and local computation.

Theorem 1 *It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:*

- *Availability*
- *Atomic consistency*

in all fair executions (including those in which messages are lost).

Proof: We prove this by contradiction. Assume an algorithm A exists that meets the three criteria: atomicity, availability, and partition tolerance. We construct an execution of A in which there exists a request that returns an inconsistent response. The methodology is similar to proofs in Attiya

[4]Brewer originally only required *almost all* requests to receive a response. As allowing probabilistic availability does not change the result when arbitrary failures occur, for simplicity we are requiring 100% availability.

[5]Brewer pointed out in the talk that partitions of one node are irrelevant: they are equivalent to that node failing. However restricting our attention to partitions containing only components of size greater than one does not change any of the results in this note.

et al. [1] and Lynch [5] (Theorem 17.6). Assume that the network consists of at least two nodes. Thus it can be divided into two disjoint, non-empty sets: $\{G_1, G_2\}$. The basic idea of the proof is to assume that all messages between G_1 and G_2 are lost. If a *write* occurs in G_1, and later a *read* occurs in G_2, then the *read* operation cannot return the results of the earlier *write* operation.

More formally, let v_0 be the initial value of the atomic object. Let α_1 be the prefix of an execution of A in which a single *write* of a value not equal to v_0 occurs in G_1, ending with the termination of the *write* operation. Assume that no other client requests occur in either G_1 or G_2. Further, assume that no messages from G_1 are received in G_2, and no messages from G_2 are received in G_1. We know that this *write* completes, by the availability requirement. Similarly, let α_2 be the prefix of an execution in which a single *read* occurs in G_2, and no other client requests occur, ending with the termination of the *read* operation. During α_2 no messages from G_2 are received in G_1, and no messages from G_1 are received in G_2. Again we know that the *read* returns a value by the availability requirement. The value returned by this execution must be v_0, as no *write* operation has occurred in α_2.

Let α be an execution beginning with α_1 and continuing with α_2. To the nodes in G_2, α is indistinguishable from α_2, as all the messages from G_1 to G_2 are lost (in both α_1 and α_2, which together make up α), and α_1 does not include any client requests to nodes in G_2. Therefore in the α execution, the *read* request (from α_2) must still return v_0. However the *read* request does not begin until after the *write* request (from α_1) has completed. This therefore contradicts the atomicity property, proving that no such algorithm exists. ■

Corollary 1.1 *It is impossible in the asynchronous network model to implement a read/write data object that guarantees the following properties:*

- *Availability, in all fair executions,*
- *Atomic consistency, in fair executions in which* no *messages are lost.*

Proof: The main idea is that in the asynchronous model an algorithm has no way of determining whether a message has been lost, or has been arbitrarily delayed in the transmission channel. Therefore if there existed an algorithm that guaranteed atomic consistency in executions in which no messages were lost, then there would exist an algorithm that guaranteed atomic consistency in all executions. This would violate Theorem 1.

More formally, assume for the sake of contradiction that there exists an algorithm A that always terminates, and guarantees atomic consistency in fair executions in which all messages are delivered. Further, Theorem 1 implies that A does not guarantee atomic consistency in all fair executions, so there exists some fair execution α of A in which some response is not atomic.

At some finite point in execution α, the algorithm A returns a response that is not atomic. Let α' be the prefix of α ending with the invalid response. Next, extend α' to a fair execution α'', in which all messages are delivered. The execution α'' is now a fair execution in which all messages are delivered. However this execution is not atomic. Therefore no such algorithm A exists. ■

3.2 Solutions in the Asynchronous Model

While it is impossible to provide all three properties: atomicity, availability, and partition tolerance, any two of these three properties can be achieved.

3.2.1 Atomic, Partition Tolerant

If availability is not required, then it is easy to achieve atomic data and partition tolerance. The trivial system that ignores all requests meets these requirements. However we can provide a stronger liveness criterion: if all the messages in an execution are delivered, the system is available and all operations terminate. A simple centralized algorithm meets these requirements: a single designated node maintains the value of an object. A node receiving a request forwards the request to the designated node, which sends a response. When an acknowledgment is received, the node sends a response to the client.

Many distributed databases provide this type of guarantee, especially algorithms based on distributed locking or quorums: if certain failure patterns occur, then the liveness condition is weakened and the service no longer returns responses. If there are no failures, then liveness is guaranteed.

3.2.2 Atomic, Available

If there are no partitions, it is clearly possible to provide atomic, available data. In fact, the centralized algorithm described in Section 3.2.1 meets these requirements. Systems that run on intranets and LANs are an example of these types of algorithms.

3.2.3 Available, Partition Tolerant

It is possible to provide high availability and partition tolerance, if atomic consistency is not required. If there are no consistency requirements, the service can trivially return v_0, the initial value, in response to every request. However it is possible to provide weakened consistency in an available, partition tolerant setting. Web caches are one example of a weakly consistent network. In Section 4.4 we consider one of the possible weaker consistency conditions.

4 Partially Synchronous Networks

4.1 Partially Synchronous Model

The most obvious way to try to circumvent the impossibility result of Theorem 1 is to realize that in the real world, most networks are not purely asynchronous. If you allow each node in the network to have a clock, it is possible to build a more powerful service.

For the rest of this paper, we will assume a partially synchronous model in which every node has a clock, and all clocks increase at the same rate. However, the clocks themselves are not synchronized, in that they may display different values at the same real time. In effect, the clocks act as timers: local state variables that the processes can observe to measure how much time has passed. A local timer can be used to schedule an action to occur a certain interval of time after some other event. Furthermore, assume that every message is either delivered within a given, known time: t_{msg}, or it is lost. Also, every node processes a received message within a given, known time: t_{local}, and local processing takes zero time. This can be formalized as a special case of the General Timed Automata model described by Lynch in Chapter 23 of [5].

4.2 Impossibility Result

It is still impossible to have an always available, atomic data object when arbitrary messages may be lost, even in the partially synchronous model. That is, the following analogue of Theorem 1 holds:

Theorem 2 *It is impossible in the partially synchronous network model to implement a read/write data object that guarantees the following properties:*

- *Availability*
- *Atomic consistency*

in all executions (even those in which messages are lost).

Proof: This proof is rather similar to the proof of Theorem 1. We will follow the same methodology: divide the network into two components, $\{G_1, G_2\}$, and construct an admissable execution in which a *write* happens in one component, followed by a *read* operation in the other component. This *read* operation can be shown to return inconsistent data.

More formally, construct execution α_1 as before in Theorem 1: a single *write* request and acknowledgment occur in G_1, and all messages between the two components, $\{G_1, G_2\}$, are lost. We will construct the second execution, α'_2, slightly differently. Let α'_2 be an execution that begins with a long interval of time during which no client requests occur. This interval must be at least as long as the entire duration of α_1. Then append to α'_2 the events of α_2, as defined above in Theorem 1: a single *read* request and response in G_2, again assuming all messages between the two components are lost. Finally, construct α by superimposing the two executions α_1 and α'_2. The long interval of time in α_2 ensures that the *write* request completes before the *read* request begins. However, as in Theorem 1, the *read* request returns the initial value, rather than the new value written by the *write* request, violating atomic consistency. ■

4.3 Solutions in the Partially Synchronous Model

In the partially synchronous model, however, the analogue of Corollary 1.1 does not hold. The proof of this corollary does in fact depend on nodes being unaware of when a message is lost. There are partially synchronous algorithms that will return atomic data when all messages in an execution are delivered (i.e., there are no partitions), and will only return inconsistent (and, in particular, stale) data when messages are lost. One example of such an algorithm is the centralized protocol described in Section 3.2.1, modified to time-out lost messages. On a *read* (or *write*) request, a message is sent to the central node. If a response from the central node is received, then the node delivers the requested data (or an acknowledgment). If no response is received within $2 * t_{msg} + t_{local}$, then the node concludes that the message was lost. The client is then sent a response: either the best known value of the local node (for a *read* operation), or an acknowledgment (for a *write* operation). In this case, atomic consistency may be violated.

4.4 Weaker Consistency Conditions

While it is useful to guarantee that atomic data will be returned in executions in which all messages are delivered (within some time bound), it is equally important to specify what happens in executions in which some of the messages are lost. In this section, we will discuss one possible weaker consistency condition that allows stale data to be returned when there are partitions, yet still place formal requirements on the quality of the stale data returned. This consistency guarantee will require availability and atomic consistency in executions in which no messages are lost, and is therefore impossible to guarantee in the asynchronous model as a result of Corollary 1.1.

In the partially synchronous model it often makes sense to base guarantees on how long an algorithm has had to rectify a situation. This consistency model ensures that if messages are delivered, then eventually some notion of atomicity is restored.

In an atomic execution, we would define a partial order of the *read* and *write* operations, and then require that if one operation begins after another one ends, the former does not precede the latter in the partial order. We will define a weaker guarantee, t-Connected Consistency, which defines a partial order in a similar manner, but only requires that one operation not precede another if there is an interval between the operations in which all messages are delivered.

Definition 3 *A timed execution, α, of a read-write object is* t-Connected Consistent *if two criteria hold. First, in executions in which no messages are lost, the execution is atomic. Second, in executions in which messages are lost, there exists a partial order P on the operations in α such that:*

1. *P orders all* write *operations, and orders all* read *operations with respect to the* write *operations.*

2. *The value returned by every* read *operation is exactly the one written by the previous* write *operation in P, or the initial value, if there is no such previous* write *in P.*

3. *The order in P is consistent with the order of* read *and* write *requests submitted at each node.*

4. *Assume there exists an interval of time longer than t in which no messages are lost. Further, assume an operation, θ, completes before the interval begins, and another operation, ϕ, begins after the interval ends. Then ϕ does not precede θ in the partial order P.*

This guarantee allows for some stale data when messages are lost, but provides a time limit on how long it takes for consistency to return, once the partition heals. This definition can of course be generalized to provide consistency guarantees when only some of the nodes are connected, and when connections are available only some of the time. These generalizations will be further examined in future work.

A variant of the centralized algorithm described in Section 4.3 is t-Connected Consistent. Assume node C is the centralized node. The algorithm behaves as follows:

- *read* at node A:
  A sends a request to C for the most recent value. If A receives a response from C within time $2 \cdot t_{msg} + t_{local}$, it saves the value and returns it to the client. Otherwise, A concludes that a message was lost and it returns the value with the highest sequence number that has ever been received from C, or the initial value if no value has yet been received from C. (When a client *read* request occurs at C, it acts like any other node, sending messages to itself.)

- *write* at A:
  A sends a message to C with the new value. A waits $2 \cdot t_{msg} + t_{local}$, or until it receives an acknowledgment from C, and then sends an acknowledgment to the client. At this point, either C has learned of the new value, or a message was lost, or both events occurred. If A concludes that a message was lost, it periodically retransmits the value to C (along with all values lost during earlier *write* operations) until it receives an acknowledgment from C. (As in the case of *read* operations, when a client *write* request occurs at C, it acts like any other node, sending messages to itself.)

- New value is received at C:
  C serializes the *write* requests that it hears about by assigning them consecutive integer tags. Periodically C broadcasts the latest value and sequence number to all other nodes.

Theorem 4 *The modified centralized algorithm is t-Connected consistent.*

Proof: First, it is clear that in executions in which no messages are lost, the operations are atomic. An execution is atomic if every operation acts as if it is executed at a single instant; in this case, that single instant occurs when C processes the operation. C serializes the operations, ensuring atomic consistency in executions in which all messages are delivered.

Next, we examine executions in which messages are lost. The partial order, P is constructed as follows. *Write* operations are ordered by the sequence number assigned by the central node. Each *read* operation is sequenced after the *write* operation whose value it returns. It is clear by the construction that the partial order P satisfies criteria 1 and 2 of the definition of t-Connected consistency. As the algorithm handles requests in the order received, criterion 3 is also clearly true.

In showing that the partial order respects criterion 4, there are four cases: *write* followed by *read*, *write* followed by *write*, *read* followed by *read*, and *read* followed by *write*. Let time t be long enough for a *write* operation to complete (and for C to assign a sequence number to the new value), and for one of the periodic broadcasts from C to occur.

1. *write* followed by *read*
   Assume a *write* occurs at A_w, after which an interval of time longer than t passes in which all messages are delivered. After this, a *read* is requested at some node. By the end of the interval, two things have happened. First, A_w has notified the central node of the new value, and the *write* operation has been assigned a sequence number. Second, the central node has rebroadcast that value (or a later value in the partial order) to all other nodes during one of the periodic broadcasts. As a result, the *read* operation does not return an earlier value, and therefore it must come after the *write* in the partial order P.

2. *write* followed by *write*
   Assume a *write* occurs at A_w, after which an interval of time longer than t passes in which all messages are delivered. After this, a *write* is requested at some node. As in the previous case, by the end of the interval in which messages are delivered, the central node has assigned a sequence number to the *write* operation at A_w. As a result, the later *write* operation is sequenced by the central node after the first *write* operation. Therefore the second *write* comes after the first *write* in the partial order P.

3. *read* followed by *read*
   Assume a *read* operation occurs at B_r, after which an interval of time longer than t passes in which all messages are delivered. After this, a *read* is requested at some node. Let ψ be the *write* operation whose value the first *read* operation at B_r returns. By the end of the interval in which messages are delivered, the central node has assigned a sequence number to ψ, and has broadcast the value of ψ (or a later value in the partial order) to all other nodes. As a result, the second *read* operation does not return a value earlier in the partial order than ψ. Therefore the second *read* operation does not precede the first in the partial order P.

4. *read* followed by *write*
   Assume a *read* operation occurs at B_r, after which an interval of time longer than t passes in which all messages are delivered. After this, a *write* is requested at some node. Let ψ be the

write operation whose value the first *read* operation at B_r returns. By the end of the interval in which messages are delivered, the central node has assigned a sequence number to ψ, and as a result all *write* operations beginning after the interval are serialized after ψ. Therefore the *write* operation does not precede the *read* operation in the partial order P.

Therefore, P satisfies criterion 4 of the definition, and this algorithm is t-Connected Consistent. ■

5 Conclusion

In this note, we have shown that it is impossible to reliably provide atomic, consistent data when there are partitions in the network. It is feasible, however, to achieve any two of the three properties: consistency, availability, and partition tolerance. In an asynchronous model, when no clocks are available, the impossibility result is fairly strong: it is impossible to provide consistent data, even allowing stale data to be returned when messages are lost. However in partially synchronous models it is possible to achieve a practical compromise between consistency and availability. In particular, most real-world systems today are forced to settle with returning "most of the data, most of the time." Formalizing this idea and studying algorithms for achieving it is an interesting subject for future theoretical research.

Acknowledgments

We thank Eric Brewer for his interesting PODC talk, for providing us with his talk slides and notes, and for encouraging us in writing this note. We also thank Charles Leiserson for suggesting this problem and for interesting and helpful discussions.

References

[1] Hagit Attiya, Amotz Bar-Noy, Danny Dolev, Daphne Koller, David Peleg, and Rüdiger Reischuk. Achievable cases in an asynchronous environment. In *28th Annual Symposium on Foundations of Computer Science*, pages 337–346, Los Angeles, California, October 1987.

[2] Eric A. Brewer. Towards robust distributed systems. (Invited Talk) *Principles of Distributed Computing*, Portland, Oregon, July 2000.

[3] Maurice P. Herlihy and Jeannette M. Wing. Linearizability: A correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12(3):463–492, July 1990.

[4] Leslie Lamport. On interprocess communication – parts I and II. *Distributed Computing*, 1(2):77–101, April 1986.

[5] Nancy Lynch. *Distributed Algorithms.* Morgan Kaufman, 1996.

Topology Control and Routing in Ad hoc Networks: A Survey [6]

Rajmohan Rajaraman
College of Computer Science
Northeastern University
Boston, MA 02115, USA
rraj@ccs.neu.edu

1 Introduction

An ad hoc wireless network, or simply an *ad hoc network*, consists of a collection of geographically distributed nodes that communicate with one other over a wireless medium. An ad hoc network differs from cellular networks in that there is no wired infrastructure and the communication capabilities of the network are limited by the battery power of the network nodes. One of the original motivations for ad hoc networks is found in military applications. A classic example of ad hoc networking is network of war fighters and their mobile platforms in battlefields. Indeed, a wealth of early research in the area involved the development of packet-radio networks (PRNs) and survivable radio networks [16]. While military applications still dominate the research needs in ad hoc networking, the recent rapid advent of mobile telephony and plethora of personal digital assistants has brought to the fore a number of potential commercial applications of ad hoc networks. Examples are disaster relief, conferencing, home networking, sensor networks, personal area networks, and embedded computing applications [37].

The lack of a fixed infrastructure in ad hoc networks implies that any computation on the network needs to be carried out in a decentralized manner. Thus, many of the important problems in ad hoc networking can be formulated as problems in distributed computing. However, there are certain characteristics of ad hoc networks that makes this study somewhat different than traditional work in distributed computing. In this article, we review some of the characteristic features of ad hoc networks, formulate problems and survey research work done in the area. We focus on two basic problem domains: *topology control*, the problem of computing and maintaining a connected topology among the network nodes, and *routing*. This article is not intended to be a comprehensive survey on ad hoc networking. The choice of the problems discussed in this article are somewhat biased by the research interests of the author.

The remainder of this article is organized as follows. In Section 2, we describe various aspects relevant to modeling ad hoc networks. In Section 3, we discuss topology control. Since the nodes of an ad hoc network are often associated with points in 2-dimensional space, topology control is closely tied to computational geometry; we will briefly review this relationship and extant work in the area. In Section 4, we discuss routing protocols for ad hoc networks. After a brief overview of the many protocols that have been proposed, we discuss alternative approaches based on the adversarial network model.

2 Modeling ad hoc networks

One can model an ad hoc network as a collection of points in 2-dimensional (or 3-dimensional) Euclidean space, where each point represents a network node. Each node can be characterized by its computational and communication power. The computational power of a node determines the

[6]Supported by NSF CAREER award NSF CCR–9983901

level of coding and encryption that the node can perform, two key issues in wireless communication. The communication characteristics of the network are governed by the propagation characteristics of the radio channel and the environment, and the battery power and power control capabilities of the individual nodes. We now elaborate on these issues.

2.1 Radio propagation and interference

Modeling the wireless radio channel is a complex task; the wireless medium is susceptible to path loss, noise, interference and blockages due to physical obstructions. *Path loss* is the ratio of the received power to the transmitted power. It affects the quality of the received signal and is a function of the propagation distance. If P_R is the received signal power and P_T is the transmitted power, then in free-space (clear, unobstructed line-of-sight path), we have

$$P_R = O\left(\frac{P_t}{d^\alpha}\right). \tag{1}$$

The hidden constant in the big-Oh notation in Equation 1 depends on antenna gains and the carrier frequency, and $\alpha = 2$ [41, 50]. We also note that the particular values of the hidden constants also depend on the units used for expressing the different parameters (such as P_R, P_t and d). Realistic environments are not free-space, however; they include reflections, scattering, and diffraction by buildings, terrain, and other obstructions. Frequently, the same simple exponential propagation model, with α ranging from 2 to 4, is used for modeling such environments as well. For more information on path loss models, we refer the reader to [41].

In addition to path loss, the bit-error rate of a transmission, and hence the quality of the reception at any node, depends on the noise power and the transmission powers and locations of other transmitting nodes in the vicinity of u (assuming all of the nodes are transmitting on the same frequency band, which is what we assume in this article). We discuss here three simplified models that describe when a transmission is received successfully by its intended recipient. Let $\{X_k, k \in \mathcal{T}\}$ denote a set of nodes simultaneously transmitting at some time instant. Let P_k be the power level chosen by node X_k, for $k \in \mathcal{T}$. Then, under the *physical model* [21, 31], the transmission by a node X_i is successfully received by a node Y if

$$\frac{\frac{P_i}{d(X_i,Y)^\alpha}}{N + \sum_{k \in \mathcal{T}, k \neq i} \frac{P_k}{d(X_k,Y)^\alpha}} \geq \beta \tag{2}$$

where β specifies a threshold for the signal to interference ratio for successful receptions, and N is the noise power level (noise is usually modeled as a white Gaussian signal), normalized to appropriate units. The parameter β typically varies between 0.1 and 10, depending on the underlying wireless transmission technology. It is also often expressed in decibels; in this case, β is a lower bound on one tenth of the common logarithm of the left-hand side of Equation 2. The value of β depends on the modulation and coding schemes used [31]. Equation 2 represents a somewhat pessimistic transmission model; it assumes that the signals from all of the sources in $\{X_k : k \neq i\}$ interfere destructively with the signal from X_i. In practice, however, interfering signals tend to cancel each other, and their interference effect may be small, even when compared with the noise signal. A more optimistic high-level model that considers only pairwise interferences is the following. Under the *protocol model*, the transmission by a node X_i is received by a node Y if

$$\frac{P_i}{d(X_i,Y)^\alpha} \geq (1+\Delta)\frac{P_k}{d(X_k,Y)^\alpha} \tag{3}$$

for every other node X_k simultaneously transmitting over the same bandwidth, where $\Delta > 0$ models a protocol specified guard zone to prevent transmission interference. If we assume that the transmission powers of all the nodes are identical and fixed, then Equation 3 can be rewritten as a lower bound requirement on the ratio of $d(X_k, Y)$ and $d(X_i, Y)$ [21, 31]. In another variant of Equation 3, it is assumed that the transmission by a node X_i with power P_i blocks all nodes that are reachable from X_i with power $(1 + \Delta)P_i$ [1].

2.2 Modeling at higher layers

The radio propagation and interference models of Section 2.1 can be used to derive meaningful bounds on the capacity of ad hoc networks, given node locations and transmission power constraints [20, 21]. Such a model based on physical layer parameters, however, is cumbersome to use for designing and analyzing higher layer protocols. A simpler model that abstracts away the physical layer details is to represent an ad hoc network as a graph $G = (V, E)$ in Euclidean space. The set V is the set of all nodes. The set E contains an edge from node u to v if u can directly transmit to v; this can be determined by the path loss equation (Equation 1) and a basic signal-to-noise ratio formula (Equation 2 or 3) of Section 2.1. We refer to G as the *transmission graph*. Interference can be modeled to a limited extent by the following assumption: a transmission from u to v is successful only if there is no other node w that has an edge to v and is simultaneously transmitting. This is essentially the model that has been used to study packet radio networks (PRNs).

The PRN model, as described above, assumes that each node of an ad hoc network always transmits at the same transmission power. Modern mobile wireless units have the ability of adjusting their transmission power according to the transmission needs, subject to a maximum limit. Such *power control* reduces interference, conserves battery power of the mobile units, and hence allows for better use of the channel bandwidth. For example, if we represent the network using the transmission graph G as described in the preceding paragraph, we can have a node u successfully transmitting to v, even if there is a node w adjacent to v that is transmitting at the same time; this may happen because the received power at v of w's transmission may be much less than that of the received power at v of u's transmission, owing to different levels at which u and w are transmitting at that time.

An interesting framework, the class of *local probabilistic control* MAC protocols (LPC), has been developed in [1] to incorporate power control and interference in the analysis of high-level routing protocols. Given an arbitrary collection of ad hoc network locations, an LPC protocol assigns a probability p_{ij} for each ordered pair (i, j) of nodes such that if each node i attempted to transmit to node j with probability p_{ij} in any step, then the probability that a particular transmission, when attempted, is blocked with probability less than $1/2$. This ensures that a routing scheme implemented on top of the LPC protocol does not need any additional coordination among the nodes for power control and addressing interference. In practice, power control and interference are usually addressed at the multiple-access layer of the OSI 7-layer hierarchy (e.g., the IEEE 802.11 MAC protocol and suggested enhancements [2]). Many routing protocols for ad hoc network separate the multiple-access control (MAC) and the network layer concerns. Thus, at the network layer, one can simply model the network as a directed graph and view each edge as an independent non-interfering channel of communication. The particular directed graph is typically a subgraph of the transmission graph and is determined so that it satisfies certain desirable properties with respect to size, power-efficiency, and throughput. The problem of determining an appropriate topology is the topic of Section 3.

2.3 Mobility

There are two approaches to modeling mobility in ad hoc networks. One approach, often used in simulations, is to model the motion of a node as a mobility vector, that gives the direction and speed of the node. Each node independently chooses a mobility vector that defines its motion for a period of time, after which a new random mobility vector is assigned (e.g., see [30, 33]). Models for group movement, whereby a group of nodes may move in the same general direction have also been recently studied [2, 30].

For a theoretical analysis, detailed models of mobility, as above, are difficult to work with. Instead, mobility can be represented by changes in the underlying transmission graph. For instance, we can analyze the robustness of an ad hoc network routing protocol by considering the amount of work needed to be done when an elementary change in the transmission graph occurs; that is, when an edge is removed or added or the neighborhood of a node changes [17, 47]. Another interesting model for capturing node mobility is the recently proposed adversarial network model [7], in which an adversary may alter the underlying graph in an unpredictable manner. Arbitrary node movements can be represented by adversarial changes in topology. We discuss this model in greater detail in Section 4.4.

3 Topology control

The absence of a central infrastructure implies that an ad hoc network does not have an associated fixed topology. Indeed, an important task of an ad hoc network consisting of geographically dispersed nodes is to determine an appropriate topology over which high-level routing protocols are implemented. In this section, we consider topology control, the problem of determining an appropriate topology in an ad hoc network. Let V denote the collection of nodes and let G denote the graph on V in which there is an edge from node u to node v if and only if u can directly reach v. Let T denote the topology returned by the topology control algorithm. The quality of the topology T can be evaluated according to several criteria including connectivity, energy-efficiency, throughput, and robustness to mobility. In the remainder of this section, we elaborate on these measures.

3.1 Connectivity and energy-efficiency

Perhaps, the most basic requirement of a topology is that it be connected. More precisely, we require that any two nodes that are connected in G are also connected in T. Since the topology T forms the underlying network for routing protocols, it is also desirable that there exist energy-efficient paths between potential source-destination pairs. One notion of energy-efficiency is the *energy stretch factor*, which we now define. As discussed in Section 2, the power required, and hence the energy consumed, for a transmission from u to v is a polynomial function of the distance between u and v. Define the energy used for delivering a packet along a path to be the sum of the energy used along the edges of the path. For two nodes u and v, let $E_G(u,v)$ (resp., $E_T(u,v)$) denote the energy of the minimum energy path between u and v in G (resp., T). We now define the *energy stretch factor* of T to be the maximum, over all u and v, of $E_G(u,v)/E_T(u,v)$. A notion of quality similar to the energy stretch factor is the *hop stretch factor* which measures the ratios of the hop-counts rather than that of the energy.

We would like to provide connectivity and energy-efficiency using a "simple" topology that is "easy" to maintain. While there is no single way to formalize "simplicity" and "maintainability", some objective measures that influence these subjective goals are the size of the topology in terms of the number of edges in T and the maximum degree of any node in T.

Connectivity, degree, and size are network design measures common to both wired and wireless settings. Analogous to the notion of energy stretch factor is that of *distance stretch factor* (or simply the stretch factor) in fixed-connection networks, where the distance between two nodes is the length of the shortest path between the two nodes. The problem of designing topologies with low stretch factors has been extensively studied by network designers. Of most relevance in this context is the notion of a spanner. Given a graph G on a set of nodes, a *spanner* is a subgraph H of G in which the distance between any two nodes is within a constant factor of the distance between the two nodes in G. Put differently, the distance stretch factor of a spanner is $O(1)$. If we adopt the model, as discussed in Section 2, that power attenuates as distance raised to an exponent greater than 1 (a typical assumption is an exponent between 2 and 4), then it can be shown that a topology with a $O(1)$ distance stretch factor also has a $O(1)$ energy stretch factor.

What distinguishes the topology control problem in the mobile ad hoc setting from traditional network design is that we need to determine the topology in a completely distributed environment. A number of distributed topology control algorithms have been proposed recently [32, 43, 52, 53]. These algorithms draw upon computational geometry techniques that define connected topologies on points in Euclidean space. The techniques, and the topologies obtained, vary in the degree of simplicity, the quality of the topology, and their suitability for distributed implementation. We now review some well-studied geometric structures and their associated topology control algorithms.

For nodes in Euclidean space, a number of *proximity graphs* have been proposed. These include relative neighborhood graphs and Gabriel graphs [49]. Let V be a collection of nodes in Euclidean space. The *relative neighborhood graph* (RNG) has an edge between two nodes u and v, if there is no node w such that $\max\{d(u,w), d(v,w)\} < d(u,v)$. The *Gabriel graph* (GG) has an edge between two nodes u and v if and only if there is no node w such that $d^2(u,w) + d^2(v,w) \leq d^2(u,v)$. Both RNG and GG are easy to compute using a local algorithm. While both RNG and GG are connected graphs, they have poor spanning ratios in the worst case. The worst-case spanning ratio of GG is $\Omega(\sqrt{n})$, while that of RNG is $\Omega(n)$. Even for random point sets, it has been shown that GG and RNG have $\omega(1)$ spanning ratios. With regard to energy-efficiency, GG has energy-stretch of 1 and hence is optimal, while RNG has polynomial energy-stretch. The worst-case degree of GG is $\Omega(n)$.

An elegant generalization of proximity graphs due to Yao [56] yields spanners for an arbitrary collection of points in finite-dimensional Euclidean space. Given a set of nodes in 2-dimensional space, suppose we partition the space around each node into *sectors* of a fixed angle and connect the node to the nearest neighbor in each sector. If each sector has an angle of $\Theta < \pi/3$, the resultant graph, commonly referred to as a Θ-graph, has been shown to be a connected graph with stretch $1/(1 - 2\sin(\Theta/2))$ [44]; thus, the Θ-graph is a spanner.

For ad hoc wireless networks, the Θ-graph can be easily constructed using a fast local algorithm in which each node queries nodes within its transmission radius, and selects the nearest nodes in each sector as its adjacent nodes. We note that the construction of the Θ-graph requires that the nodes know their own positions, either from a geographical positioning system (GPS) or through other means, such as inertial sensors and acoustic range-finding devices. One drawback of the Θ-graph is its large maximum degree in the worst-case. Consider the example of all nodes on the circumference of a circle, and a node at the center that is reachable from every node on the circumference of the circle. Then, the center is a neighbor for every node in the Θ-graph. One can obtain a bounded-degree subgraph of the Θ-graph that is also a spanner by processing the edges in order by length and adding an edge (u,v) to the subgraph if there is no other edge (u,w) or (v,w) already added and having an angle close to that of (u,v) [45] (a related idea is used in [5]). The topology control algorithm of [53] adopts a similar approach to convert a Θ-graph to a constant-degree spanner. The basic idea is to eliminate an edge between u and v if there exists a node w such that (u, w) and (w, v) are also edges, and $d(u, w)$ and $d(w, v)$ are both smaller than $d(u, v)$ (or some constant q times $d(u, v)$). Performing this elimination process in a completely uncoordinated manner may, however, adversely affect the spanner property. If the edges of the Θ-graph are processed in a particular order, then it is shown that the spanner property is maintained and each node ends up with a constant number of adjacent edges. The edge processing order, however, relies on a global ranking and it is not apparent how to implement the postprocessing of the Θ-graph edges in time independent of the diameter of the network. A different postprocessing of the Θ-graph has also been proposed in [55, 52]; the graph, referred to as the *Yao-Yao graph*, can be easily computed locally and also guarantees constant degree for each node. Whether the Yao-Yao graph satisfies the spanner property is unknown.

Another geometric structure that leads to a spanner is the Delaunay triangulation of the set of points, which is a collection of edges satisfying the property for each edge that there is a circle containing the edge endpoints but not containing any other points. Without additional restrictions, however, the Delaunay triangulation graph may include edges much longer than the transmission range of a node. It has been shown that restricted Delaunay graphs [17], in which we only include Delaunay edges with a limited fixed transmission radius, are also spanners. Furthermore, the number of edges in a restricted Delaunay graph is linear in the number of nodes; the maximum degree of a node may be $\Omega(n)$ in the worst case, however. For a comprehensive survey on geometric spanners and other structures in geometric network design, see [14].

The spanner property only ensures the existence of distance- and energy-efficient paths; how are these paths computed online when routing requests arise? While for some topologies such as the Θ-graph and certain variants, the quality paths are easily calculated using local control, for others, including the restricted Delaunay graph and the Yao-Yao graph, this is not the case [17, 52]. (Also see Section 4 for more discussion on online selection of paths, when we consider routing protocols.)

Our preceding discussion on energy-efficiency has focused on unicast energy usage. Also of interest is to identify energy-efficient structures for broadcast and multicast operations. A number of greedy heuristics for broadcast routing have been studied in [51, 54]. While it has been shown that a constant-factor approximation is achievable (in, fact, the minimum spanning tree is shown to be $O(1)$-approximate), the complexity of the optimization problem is still open. Also, the algorithms analyzed in [51] are all centralized; no efficient distributed algorithms are known.

3.2 Throughput

In addition to connectivity and energy-efficiency, we would like to have a topology with high capacity or throughput; that is, it must be feasible to route "about as much traffic" in the topology as any other topology, satisfying the desired constraints. Depending on the network characteristics that are being studied and the traffic patterns being considered, one can formalize the notion of throughput of an ad hoc network in different ways, and we review some definitions in the following.

In [21], Gupta and Kumar analyze the throughput of ad hoc networks under both the physical and protocol models of interference, as defined in Section 2.1. They define the throughput in terms of terms of a bit-distance product. Suppose we say that the network transports one *bit-meter* when one bit has been transported a distance of one meter. Then, the throughput of a network can be measured in terms of the number of bit-meters that are transported per second. It is shown in [21] that for n identical nodes randomly located in a disk of unit area, each node using a fixed transmission range, the throughput achievable for each source for a randomly selected destination, measured as a bit-distance product, is inversely proportional to $\sqrt{n \log n}$. It is also shown that if we allow the nodes as well as the source-destination pairs to be selected optimally, the maximum bit-distance product achievable by the network per unit time grows as $\Theta(\sqrt{n})$; thus, the average bit-distance product per node is inversely proportional to $\sqrt{n}$. The preceding results suggest that under certain assumptions about traffic pattern or node locations, the average throughput furnished to each user decreases as the number of users increases. In a follow up study [20], it is shown that this apparent "inscalability" can be alleviated for delay-tolerant applications under certain (optimistic) assumptions about node mobility and probabilistic assumptions about source-destination pairs and node locations.

The preceding definitions of throughput capture the capabilities of the network in the best-case and average-case (assuming a random traffic pattern) settings. Quite often, though, the traffic pattern does not fit into either of these moulds. A better model in these situations is to define throughput for a traffic pattern determined by arbitrary source-destination pairs exchanging data at arbitrary rates. One performance measure is then *throughput-competitiveness* which is the largest number $\phi \leq 1$ such that given any set $\{(s_i, t_i)\}$ of source destination pairs, if a flow of r_i can be routed in G among each source-destination pair, then a flow of $\phi \cdot r_i$ can be routed in T. Throughput-competitiveness captures the *worst-case* performance of the topology and is, thus, a hard performance measure to optimize. In scenarios where we have some knowledge of traffic patterns, we can restrict the class of source-destination pairs and measure throughput-competitiveness in a more meaningful manner.

The throughput-competitiveness of a topology depends on, among other factors, the level of interference inherent to the topology. Define the *interference number* of an edge e in T to be the maximum number of other edges in T that interfere with e, in the sense of Section 2 [6]. Define the interference number of the topology to be the maximum interference number of an edge in T. A plausible goal then is to seek a topology with a small interference number. The particular interference number achievable, however, depends on the relative positions of the ad hoc network nodes and their transmission radii. This leads to the following open problem in network design: Given a collection of ad hoc network nodes, design a connected topology that minimizes the interference number. It seems unlikely that the preceding optimization problem can be solved effectively by a local algorithm; nevertheless, a centralized algorithm for the problem may be of theoretical interest.

Can we provide simultaneous guarantees on throughput and energy-efficiency, given an arbitrary collection of nodes forming an ad hoc network? This is the focus of research in a recent work [6], where tradeoffs among congestion, energy-efficiency, and dilation (hopcount-efficiency) are studied. It is shown that there exists an ad hoc network and a set of source-destination pairs with associated flow rates such that any topology incurs a $\Omega(n^{1/3})$ factor overhead in either congestion or total energy consumed, where n is the number of nodes in the network. Bounds on the product of congestion and dilation and the product of dilation and energy have also been derived in [6].

3.3 Robustness to mobility

An additional challenge in the design of distributed topology control algorithms is to ensure some degree of robustness to the mobility of nodes. One measure of robustness of the topology is given by the maximum number of nodes that need to change their topology information as a result of a movement of a node. This number, which may be referred to as the *adaptability* of the topology control algorithm, depends on the size of the transmission neighborhood of the mobile node u, and the relative location of the nodes. The topology control algorithms based on proximity graphs all have low adaptability, since a change in a node location will only require the nodes in its neighborhood (both old and new) to recompute their edges in the topology. The topology of Gao et al [17] is more complex since it relies on a hierarchical clustering of the nodes. Under certain assumptions about the distribution of points on the plane, however, they have shown that the number of nodes that need to be updated due to a change in the underlying transmission graph is proportional to the number of nodes in the immediate neighborhood of the mobile node, the update time per node being a constant. Other than maintaining the topology, mobility also entails changes in the routing paths. The maintenance of routing paths in the presence of mobility is discussed in Section 4.

4 Routing protocols

In the previous section, we considered the design of topologies that have certain desirable properties in terms of connectivity, energy-efficiency, and throughput. We now consider the design of routing schemes that harness these properties. We note that while the presentation in this article follows the approach of separating the network design and routing scheme design components, the two components are closely intertwined. The choice of the particular topology control algorithm may have a strong impact on the choice of the routing scheme. Since the topology is constantly changing, the routing scheme has to be robust to changes in topology.

How do we analyze the efficiency of an ad hoc network routing protocol? One framework is to analyze the cost of individual routing requests using the measures laid out in Section 3, namely, stretch and power stretch. Also relevant are the measures of adaptability and the memory overhead. The memory overhead is simply the size in bits of all the data structures used by the routing protocol. In Sections 4.1, 4.2, and 4.3, we survey several routing protocols proposed for ad hoc networks and discuss the tradeoffs among the different performance measures. In Section 4.4, we discuss the adversarial model, an alternative framework for analysis.

4.1 Flat routing protocols

For an ad hoc network, given a topology, represented by an undirected graph $G = (V, E)$, a routing scheme has to select paths between source destination pairs in much the same manner as in wired networks. Two paradigms that underlie Internet routing protocols are Distance Vector (DV) and Link State (LS) algorithms. Both DV and LS algorithms require continual exchange of global routing information. This enables the individual nodes of the network to maintain a close approximation of the current network map at every instant. For ad hoc networks, *proactive* routing protocols follow the DV or LS paradigm and attempt to keep routing information for all the nodes up to date, e.g., OLSR, DSDV [38]. When the topology of an ad hoc network is under constant flux, however, LS generates large number of link state changes, while DV algorithms frequently suffer from out of date state. The size of the network and the mobility of the nodes are two hurdles in the design of scalable routing protocols. For example, the DSDV protocol has $O(1)$ stretch but requires each node to store an $O(n)$-size distance vector; one consequence of the latter memory overhead is that the adaptability of the network is high since all distance vectors may need to be updated when a node moves.

In contrast to proactive algorithms, *reactive* routing protocols cache topological information and update the cached information on-demand. Reactive protocols avoid the prohibitive cost of routing information maintenance of proactive protocols, and tend to work well in practice. While the idea of aggressive caching and occasional update results in good average performance, the worst-case latency could be high. Examples of reactive protocols are Dynamic Source Routing (DSR) [25], Ad-hoc On-Demand Distance Vector Routing (AODV) [36], and TORA [35]. For a comparison of certain proactive and reactive routing protocols, see [13].

Hybrids of proactive and reactive protocols, e.g., Zone Routing Protocol [22], have also been proposed, that maintain a clustering of the network and keep routing information up-to-date within a cluster while using a reactive paradigm for collecting information about distant nodes.

The most basic clustering that has been studied in the context of ad hoc networks is based on dominating sets. Given an undirected graph $G = (V, E)$, a *dominating set* D of G is a subset of V such that for every node $v \in V$, either $v \in D$ or there exists a node $u \in D$ such that $(u, v) \in E$. The set D of *dominators* identifies a set of clusters, each cluster consisting of a node in D and its adjacent nodes. (If we need non-overlapping clusters, then we can have each node v in $V - D$ associate with exactly one node in D that is adjacent to v.) The dominators act as clusterheads that store global routing information. To minimize the cost of updating the global routing database due to network changes, clusterings based on a small number of clusters are desirable. The problem of finding a dominating set of minimum size, however, is a classic NP-complete optimization problem [18] and it is known that unless NP has $n^{O(\log\log n)}$-time deterministic algorithms, the best approximation ratio achievable in polynomial time is $O(\log n)$ [15].

In the context of ad hoc networks, we are interested in distributed algorithms for finding a small dominating set. Kutten and Peleg describe a distributed dominating set algorithm which takes $O(\log^* n)$ time [29] on any network, assuming a synchronous model of computation, in which each node can exchange a message with each neighbor in each step. The primary emphasis in [29] is on time complexity, however, and there is no nontrivial asymptotic upper bound on the approximation ratio. In contrast, Haas and Liang [33] present a distributed implementation of the greedy algorithm that achieves the same approximation ratio as the greedy algorithm; however, there exist networks for which the distributed greedy algorithm takes $\Omega(n)$ time. Building upon NC algorithms for the set cover problem [11, 40], a randomized distributed implementation of the sequential algorithm is presented in [24] and is shown to achieve an $O(\log n)$ approximation in $O(\log^2 n)$ time with high probability. It is still open whether one can derive the same approximation in $O(\log n)$ time. Also, better approximations for nodes in Euclidean space merit further investigation.

4.2 Hierarchical routing protocols

The idea of one-level clustering, as discussed in Section 4.1, can be easily generalized to a multilevel hierarchical network decomposition. Indeed, this is an old concept in networking dating back to the 70s [26, 34]. While many of the hierarchical routing protocols were originally designed for fixed networks, they are applicable, with suitable modification, for ad hoc networks. The main idea of hierarchical routing is to organize the network as a hierarchy of nested clusters of nodes. Each node of the network is a level-0 cluster. The level-i clusters are grouped together into a certain number of level-$i+1$ clusters, for $i \geq 0$. In the most basic clustering, one assumes that all level-i clusters are disjoint; many routing protocols use overlapping clusters to provide fault-tolerance and make the protocol more adaptive to dynamic network changes.

A hierarchical control structure enforces a hierarchical addressing on the network nodes, which can form the basis of a routing scheme. In a typical routing scheme, each cluster elects certain leaders within the cluster, which obtain and represent network state information at multiple levels of granularity. Routing can be performed by forwarding the given packet to a level-i cluster which contains the destination node, successively for decreasing value of i, until the packet reaches a level-0 cluster containing the destination, which is the destination itself. Routing protocols differ in the precise mechanism by which network state information is gathered and the particular paths used in the routing process.

The hierarchical control structures on which different routing protocols are based differ in the number m of levels of the hierarchy, the size and diameter of the clusters at different levels of the hierarchy, and amount of overlap among the clusters. The different choices lead to a natural tradeoff between memory overhead and the stretch factor. Steenstrup [47] reviews several practical hierarchical clustering protocols proposed for ad hoc networks. Most of these protocols rely on heuristics and, as such, do not provide provable worst-case guarantees.

From a theoretical standpoint, the protocol presented in [10] provides a suitable tradeoff between stretch and memory overhead. The routing protocol, designed for fixed-connection networks, uses the elegant network decomposition technique of sparse neighborhood covers to achieve a stretch of $O(k^2)$ while using $O(kn^{1/k} \log^2 n \log D)$ local memory overhead, where D is the diameter of the network. (See [19] for a recent survey on tradeoffs in routing for fixed-connection networks.) Thus, the preceding protocol guarantees the near-optimal tradeoff of achieving $O(\text{polylog}(n))$ stretch while only using $O(\text{polylog}(n))$ local memory overhead. There are two major disadvantages of the above protocol in the context of ad hoc networks. First, responding to mobility of nodes may require significant overhead. Second, the decomposition technique is complex and would be impractical for scenarios in which the network nodes have limited power and computational capabilities.

4.3 Geographic routing protocols

A recent approach to designing simple protocols that keep overhead small is to exploit the underlying geometry (and geography) of the ad hoc network locations. In the Greedy Perimeter Stateless Routing (GPSR) protocol [27], each node only maintains information about their "neighborhood", which is the set of nodes that the node can directly reach. Using positioning information, the source node greedily passes a given packet to a neighbor that is closest to the destination; if greedy forwarding is impossible, then the packet is forwarded along a perimeter of the region to reach the destination. While GPSR guarantees connectivity, the best bound on the stretch of the protocol is $\Omega(n)$. GPSR is however, adaptive to node mobility since the nodes maintain neighborhood information only. As such, GPSR does not take into account energy efficiency.

An even simpler approach than GPSR is to use the Θ-graph, as discussed in Section 3. The Θ-graph not only defines a topology but also directly yields a simple routing protocol with $O(1)$ stretch and $O(1)$ memory overhead; the memory overhead is constant since each node needs to store the coordinates of the nearest node in each of a constant number of sectors. This approach is used in [23] for routing in the plane. The worst-case adaptability of the routing scheme is at least the maximum in-degree of a Θ-graph, which may be large; consequently, the movement of a single node may require updates in a large number of nearby nodes. One approach to alleviate this problem is to use the constant-degree variants of the Θ-graph, as discussed in Section 3. Unfortunately, while the topology control algorithms based on these variants guarantee the existence of energy-efficient paths, a constructive mechanism for calculating these paths in a distributed manner is not known.

4.4 Adversarial model

A second framework for analyzing ad hoc network routing algorithms is the adversarial model, first developed in [12] and subsequently enhanced in several recent studies [4, 8]. In the context of ad hoc networks, we can model mobility and traffic patterns using an adversary. Mobility can be modeled by allowing the adversary to activate/deactivate network edges; arbitrary traffic patterns can be modeled by allowing the adversary to determine the rate of packet arrival and the source-destination pairs for each packet. We describe here the most general adversarial model considered thus far [8]. In this model, the adversary is allowed to inject packets at arbitrary nodes at arbitrary times and can activate an arbitrary number of incoming or outgoing edges subject to a maximum degree bound Δ for each node. The destination for the each packet is also selected by the adversary. There is one more constraint, that the buffer size of each node is limited to a value B. If at any time, the number of packets in a buffer exceeds B then the excess packets have to be dropped.

The adversarial control of the network topology is intended to model the dynamic nature of an ad hoc network in which edges may appear and disappear over time. We assume that no packets are lost; furthermore the model does not cover malicious faults in the sense that it is implicitly assumed that all of the nodes faithfully execute a given routing protocol. The adversarial control of packet injection models the dynamic and unpredictable nature of network traffic.

The performance of a given routing algorithm can now be measured by means of a competitive analysis framework. For a given sequence σ of packet injections and edge activations, let $\mathrm{OPT}_B(\sigma)$ denote the maximum number of packets that can be delivered assuming the buffer of each node is of size at most B. For a given algorithm $\mathcal{A}$, let $\mathcal{A}_{B'}(\sigma)$ denote the number of packets delivered by $\mathcal{A}$ assuming the buffer of each node is of size at most B'. We call $\mathcal{A}$ (c, s)-competitive if for all σ and B, we have

$$\mathcal{A}_{sB}(\sigma) \geq c \cdot \mathrm{OPT}_B(\sigma) - r,$$

for some value $r \geq 0$ that is independent of $\mathrm{OPT}_B(\sigma)$ [8, 46].

The best result known for the above adversarial model is a simple local balancing algorithm that send packets from nodes with high load to nodes with low load; the load on a node is signified by the height of the buffer [8]. (The local balancing approach has also formed the basis of fast algorithms for load balancing [3] and multicommodity flow [9].) For a given $T \geq B + 2(\Delta - 1)$, the algorithm of [8] is $O(1 - \varepsilon, 1 + (1 + (T + \Delta)/B)L/\varepsilon)$-competitive, where L is the average path length used by successful packets in an optimal solution; in particular, if the degree Δ is always constant, then there exists a simple $(1 - \varepsilon, O(L/\varepsilon))$-competitive algorithm. An important open question is what is the smallest value of s that yields a (δ, s)-competitive algorithm for some constant $0 < \delta < 1$.

5 Concluding remarks

In Section 4, we noted that many routing protocols, proposed for ad hoc networks are adaptations of protocols originally devised for fixed-connection networks. We anticipate that due to the advent of peer-to-peer computing, the two problem domains will bear an even stronger relationship; peer-to-peer networks share many of the same concerns with ad-hoc networks, e.g., a need to quickly adapt to the frequent changes in the system and completely decentralized organization. An interesting direction for future research is to see whether resource location protocols designed for peer-to-peer networks [28, 39, 42, 48] can be adapted to yield effective routing protocols for ad hoc networks.

Finally, we note that most of the results we have discussed in this article rely on MAC layer protocols for power control and contention resolution, and implicitly assume that the ad hoc network is deployed over open, flat terrain. Designing power control protocols to improve bandwidth and energy efficiency is an active area of research [2, 30]. Models for radio propagation in the presence of blockages that have been used in simulation studies tend to be complex; it would be useful to develop simpler models suitable for analyzing routing protocols.

Acknowledgments

The author wishes to thank Sergio Rajsbaum for several valuable comments on an earlier draft of this article.

References

[1] M. Adler and C. Scheideler. Efficient communication strategies for ad-hoc wireless networks. In *Proceedings of the 10th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 259–268, June 1998.

[2] S. Agarwal, S. Krishnamurthy, R. Katz, and S. Dao. Distributed power control in ad-hoc wireless networks. In *Proceedings of PIMRC*, 2001.

[3] W. Aiello, B. Awerbuch, B. Maggs, and S. Rao. Approximate load balancing on dynamic and asynchronous networks. In *Proceedings of the 25th Annual ACM Symposium on Theory of Computing*, pages 632–641, May 1993.

[4] M. Andrews, B. Awerbuch, A. Fernández, J. Kleinberg, T. Leighton, and Z. Liu. Universal stability results for greedy contention-resolution protocols. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science*, pages 380–389, October 1996.

[5] S. Arya and M. Smid. Efficient construction of a bounded degree spanner with low weight. In *Proceedings of 2nd European Symposium on Algorithms*, pages 48–59, 1994.

[6] F. Meyer auf der Heide, C. Schindelhauer, K. Volbert, and M. Grunewald. Energy, congestion and dilation in radio networks. In *Proceedings of the 14th Annual ACM Symposium on Parallel Algorithms and Architectures*, August 2002. To appear.

[7] B. Awerbuch, P. Berenbrink, A. Brinkmann, and C. Scheideler. Simple routing strategies for adversarial systems. In *Proceedings of 42nd Annual IEEE Symposium on Foundations of Computer Science*, October 2001.

[8] B. Awerbuch, A. Brinkmann, and C. Scheideler. Unicasting and multicasting in adversarial systems. Unpublished manuscript, November 2001.

[9] B. Awerbuch and F. T. Leighton. Improved approximation algorithms for the multi-commodity flow problem and local competitive routing in dynamic networks. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 487–496, May 1994.

[10] B. Awerbuch and D. Peleg. Routing with polynomial communication space tradeoff. *SIAM Journal on Discrete Mathematics*, 5:151–162, 1990.

[11] B. Berger, J. Rompel, and P. Shor. Efficient NC algorithms for set cover with applications to learning and geometry. *Journal of Computer and System Sciences*, 49:454–477, December 1994.

[12] A. Borodin, J. Kleinberg, P. Raghavan, M. Sudan, and D. Williamson. Adversarial queueing theory. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 376–385, May 1996.

[13] J. Broch, D. Maltz, D. Johnson, Y.-C. Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, August 1998.

[14] D. Eppstein. Spanning trees and spanners. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, pages 425–461. Elsevier, Amsterdam, 2000.

[15] U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pages 314–318, May 1996.

[16] J. Freebersyser and B. Leiner. A DoD perspective on mobile ad hoc networks. In C. Perkins, editor, *Ad Hoc Networking*, pages 29–51. Addison-Wesley, 2001.

[17] J. Gao, L. Guibas, J. Hershberger, L. Zhang, and A. Zhu. Geometric spanner for routing in mobile networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking & Computing*, October 2001.

[18] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[19] C. Gavoille. Routing in distributed networks: Overview and open problems. *ACM SIGACT News*, 32(1):36–52, March 2001. Distributed Computing Column.

[20] M. Grossglauser and D. Tse. Mobility increases the capacity of wireless adhoc networks. In *Proceedings of IEEE Infocom*, 2001. To appear in IEEE/ACM Transactions on Networking.

[21] P. Gupta and P. Kumar. Capacity of wireless networks. *IEEE Transactions on Information Theory*, IT-46:388–404, 2000.

[22] Z. Haas and M. Pearlman. The performance of query control schemes for the zone routing protocol. In *Proceedings of ACM SIGCOMM*, September 1998.

[23] Y. Hassin and D. Peleg. Sparse communication networks and efficient routing in the plane. In *Proceedings of the 19th Annual ACM Symposium on Principles of Distributed Computing*, pages 41–59, July 2000.

[24] L. Jia, R. Rajaraman, and T. Suel. An efficient distributed algorithm for constructing small dominating sets. In *Proceedings of the 20th Annual ACM Symposium on Principles of Distributed Computing*, pages 33–42, August 2001.

[25] D. Johnson and D. Maltz. Dynamic source routing in adhoc wireless networks. In T. Imielinski and H. Korth, editors, *Mobile Computing*, pages 153–181. Kluwer Academic Publishers, 1996.

[26] F. Kamoun and L. Kleinrock. Stochastic performance evaluation of hierarchical routing for large networks. *Computer Networks*, 3(5):337–353, 1979.

[27] B. Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *Proceedings of ACM Symposium on Mobile Computing and Networking*, pages 243–254, August 2000.

[28] J. Kubiatowicz, B. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and Zhao. B. Oceanstore: An architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, November 2000.

[29] S. Kutten and D. Peleg. Fast distributed construction of k-dominating sets and applications. *Journal of Algorithms*, 28:40–66, 1998.

[30] T. Kwon and M. Gerla. Clustering with power control. In *Proceedings of MILCOM*, volume 2, pages 1424–1428, 1999.

[31] C. Lau and C. Leung. Capture models for mobile packet radio networks. *IEEE Transactions on Communications*, 40:917–925, 1992.

[32] L. Li, J. Halpern, V. Bahl, Y.-M. Wang, and R. Wattenhofer. Analysis of a cone-based distributed topology control algorithms for wireless multi-hop networks. In *Proceedings of ACM Symposium on Principles of Distributed Computing*, pages 264–273, August 2001.

[33] B. Liang and Z. Haas. Virtual backbone generation and maintenance in ad hoc network mobility management. In *Proceedings of the 2000 IEEE INFOCOM*, March 2000.

[34] J. McQuillan. Adaptive routing algorithsm for distributed computer networks. Technical Report 2831, Bolt Beranek & Newman, May 1974.

[35] V. Park and S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE Infocom*, pages 1405–1413, April 1997.

[36] C. Perkins. Adhoc on demand distance vector (AODV) routing. Internet draft, draft-ietf-manet-aodv-04.txt, October 1999.

[37] C. Perkins. Ad hoc networking: An introduction. In C. Perkins, editor, *Ad Hoc Networking*, pages 1–28. Addison-Wesley, 2001.

[38] C. Perkins and P. Bhagwat. Highly-dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of ACM SIGCOMM*, September 1994.

[39] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32:241–280, 1999.

[40] S. Rajagopalan and V. Vazirani. Primal-dual RNC approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28:525–540, 1998.

[41] T. Rappaport. *Wireless Communications: Principles and Practices.* Prentice Hall, 1996.

[42] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In *Proceedings of ACM SIGCOMM*, August 2001.

[43] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. *IEEE Journal Selected Areas in Communications*, 17:1333–1344, August 1999.

[44] J. Ruppert and R. Seidel. Approximating the d-dimensional complete Euclidean graph. In *Proceedings of the 3rd Annual Canadian Conference on Computational Geometry*, pages 207–210, 1991.

[45] J. Salowe. Euclidean spanner graphs with degree four. *Discrete Applied Mathematics*, 54:55–66, 1994.

[46] C. Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of STACS*, September 2001.

[47] M. Steenstrup. Cluster-based networks. In C. Perkins, editor, *Ad Hoc Networking*, pages 75–138. Addison-Wesley, 2001.

[48] I. Stoica, R. Morris, F. Kaashoek, D. Karger, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*, August 2001.

[49] G. Toussaint. The relative neighborhood graph of a finite planar set. *Pattern Recognition*, 12:261–268, 1980.

[50] P. Varaiya and J. Walrand. *High-Performance Communication Networks.* Morgan Kaufmann, 2000. Chapter 7.

[51] P. Wan, G. Calinescu, X. Li, and O. Frieder. Minimum-energy broadcast routing in static ad hoc wireless networks. In *Proceedings of the IEEE INFOCOM Conference*, 2001.

[52] Y. Wang and X.-Y. Li. Distributed spanner with bounded degree for wireless ad hoc networks. In *Parallel and Distributed Computing Issues in Wireless networks and Mobile Computing*, April 2002.

[53] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proceedings of IEEE Infocom*, 2001.

[54] J. Wieselthier, G. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. In *Proceedings of IEEE Infocom*, 2000.

[55] Yu Wang Xiang-Yang Li, Peng-Jun Wan and Ophir Frieder. Sparse power efficient topology for wireless networks. *Journal of Parallel and Distributed Computing*, 2002.

[56] A. Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.