AFRL-IF-RS-TR-2003-141
Final Technical Report
June 2003

INFORMATION DYNAMICS AND AGENT INFRASTRUCTURE

University of Maryland

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. K552, K359

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-2003-141 has been reviewed and is approved for publication.

APPROVED: John J. Lemmer

JOHN J. LEMMER
Project Engineer

FOR THE DIRECTOR: [illegible]

JAMES W. CUSACK, Chief
Information Systems Division
Information Directorate

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** JUNE 2003	**3. REPORT TYPE AND DATES COVERED** Final , Jun 2000 - Dec 2002

4. TITLE AND SUBTITLE INFORMATION DYNAMICS AND AGENT INFRASTRUCTURE	**5. FUNDING NUMBERS** C - F30602-00-2-0578 PE - 63760E PR - TASK TA - 00 WU - 15
6. AUTHOR(S) Ashok K. Agrawala, Udaya Shankar, and Ronald Larsen	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Maryland Department of Computer Science A. V. Williams Building College Park Maryland 20742	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Projects Agency AFRL/IFSF 3701 North Fairfax Drive 525 Brooks Road Arlington Virginia 22203-1714 Rome New York 13441-4505	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TR-2003-141

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: John J. Lemmer/IFSF/(315) 330-3657/ John.Lemmer@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

Acquisition, organization, management, retrieval, and distribution of information are fundamental purposes of digital libraries and their supporting infrastructures. Interoperable digital libraries pose particularly difficult system design issues. Interoperability research has focused largely on syntactic and semantic interoperability. In this paper, a third form of interoperability, analytic interoperability is proposed, with a framework in which to consider it. Since information is the essential commodity of interest, a comprehensive interoperability design should take into account the fundamental properties of information, including representation, composition, relationships, and dynamics. Information Dynamics considers how the nature of information can be used to achieve analytic interoperability.

14. SUBJECT TERMS Agent-Based Systems, Digital Library, Information Dynamics, Integration, Interoperability, System Design	**15. NUMBER OF PAGES** 152
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 **Standard Form 298 (Rev. 2-89)**
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

1. Project Administrative Information 1
2. Objective and Approach 1
3. Accomplishments.. 2
4. Technology Transition 3
5. Publications and Presentations.. 3
Appendix A Information Dynamics: An Information-centric Approach to System Design 5
Appendix B Information Dynamics and Its Applications...11

Appendix C Information Dynamics REF 49
Appendix D Information Dynamics: An Information-centric Approach to Digital Library Interoperability...59
Appendix E Information Dynamics and Interoperability...67
Appendix F Information Dynamics: An Information-centric Approach to Digital Library Interoperability...87.
Appendix G REF Example Problem: ARM Routing in Ad-hoc Networks.................. 95

Appendix H REF Example Problem: Attacker-Defender-Target Game 125

Appendix I REF Example Problem: Information Dynamics and Link-State Routing 130

1. Project Administrative Information

Project Title: Information Dynamics and Agent Infrastructure
Organization: University of Maryland, College Park
AO Number: K552
Contract Number: F30602-00-2-0578
Start Date: 30 Jun 2000
End Date: 31 Dec 2001
Principal Investigator: Dr. Ashok K. Agrawala
Project URL: http://www.cs.umd.edu/~shankar/InfoDyn/

2. Objective and Approach

The objective of the Information Dynamics Project at the University of Maryland, College Park, is to develop a framework for agent-based systems that gives a central position to the role of information, time, and the value of information, and to apply this framework to develop some agent-based systems. The expectation is that this emphasis will lead to better design and understanding of agent-based systems.

The Information Dynamics project was an approach to complex system design and implementation based on the philosophy that a greater understanding of the nature and characteristics of information is the key to improvements in system performance, maintenance, reliability, and extensibility.

Acquisition, distribution, management, and analysis of information are the fundamental purposes behind most complex constructed systems and infrastructures, and yet the design and implementation of such systems is fundamentally driven by process-centric approaches. Since information is the essential commodity in these endeavors, we believe that an effective design should take into account the fundamental properties of information, that is, its characteristics, representation, value, temporal dynamics, fusion, distillation, etc. Information Dynamics is an attempt to bring a degree of rigor to the understanding of the nature of information itself and how it is used in the pursuit of system objectives.

True innovation in system design and implementation requires a greater understanding of both the implicit and dynamic characteristics of information in order to ensure that information flows efficiently to the right place at the right time, and in a manner that guarantees its correct interpretation. Our approach to improving the understanding of the inherent properties of information is to explicitly consider:

- Properties and dynamics of implicit information
- Time dependent aspects of information
- Value (usefulness) of information
- Consequences of particular representation of information

- Dynamics of information requirements

Our efforts focus on application to both synthetic problems and inter-networking problems.

3. Accomplishments

- ARM: Adapting to Route-demand and Mobility:
  We have applied the information dynamics framework to the problem of efficient routing in ad-hoc networks. Specifically, we have developed a control mechanism called ARM (Adapting to Route-demand and Mobility) that allows any proactive routing protocol to dynamically adapt in a totally distributed manner to changes in node mobility and workload route-demands. The ARM agent on each node independently maintains mobility metric indicating how fast the neighborhood is currently changing, and a route-demand metric indicating which destinations are currently involved in data forwarding. Control functions use these metrics to dynamically adjust the period and the content of routing updates. Thus these control functions reflect the value of the information on mobility and route demand.

  We have applied ARM to the DSDV protocol, coming up with ARM-DSDV, and implemented a simulator. For various mobility and workload scenarios, ARM-DSDV typically achieves the same data delivery as DSDV with update period optimized for the scenario, while saving up to 60% in routing cost. Lower cost gives data traffic more available bandwidth.

- Application of Information dynamics to Link State Routing:
  Link State Routing uses the information about the state of individual links gathered at regular intervals. This information is sent to every node through flooding. However, the state of the network and its links changes very rapidly. As a consequence the value of the state information from the past decreases rapidly. The information dynamics formulation of this problem explicitly keeps track of the value of the state information and uses it to control the flooding and the route determination. We tested this approach using simulation models and results indicate that we can reduce the routing overhead (control) traffic by about three orders of magnitude.

- Attacker-Defender Intelligence Game:
  We have defined an Attacker-Defender Intelligence Game to study in detail the basic notions of Information Dynamics and the emergent behaviors that may arise as independent agents playing the roles of attackers and defenders interact according to defined strategies and constraints. Attackers and defenders move around in a space, scan for adversaries, and exchange information with fellow agents. The attackers' goal is to destroy certain targets (by surrounding them with many more attackers than defenders), and

the defenders' goal is to prevent this. The game is designed in an information-centric manner such that the information available, used and exchanged by each agent is explicitly controlled. The first version of this game has been implemented.

- Information Dynamics REF Formulation:
  The Information Dynamics REF was formulated on the Information Dynamics research effort being undertaken on this project. The idea has been to demonstrate the applicability of the information dynamics notion such as the value of information, to practical problem domains. During the first year we provided the REF leadership and formulated the initial framework for the REF that was based on an e-commerce application domain. During the last PI Meeting the primary direction of the problem domain evolved to be more directly related to the problems originating in the intelligence community.

4. Technology Transition

The results of this research publicized and made widely available through:
- Reports and papers available on the internet
- Presentations in national and international forums

5. Publications and Presentations

These reports detailing the accomplishments above are attached to this report. These reports, and further details, are available on the project web site (http://www.cs.umd.edu/~shankar/InfoDyn);

A. (Jan 2000) Information Dynamics: An Information-centric approach to system design(doc). Proceedings of the International Conference on Virtual Worlds and Simulation, January 2000, San Diego, CA.

B. (April 2001) Information Dynamics and Its Applications (ppt). TASK PIs Meeting, Santa Fe, April 2001.

C. (April 2001) Information Dynamics REF (ppt). TASK PIs Meeting, Santa Fe, April 2001.

D. (May 2001) Information Dynamics: An Information-centric Approach to Digital Library Interoperability (doc). NIT 2001 Global Digital Library Development in the New Millennium Beijing, China, May 2001.

E. (May/June 2001) Information Dynamics and Interoperability (ppt). NIT 2001 Global Digital Library Development in the New Millennium Beijing, china, May 2001, and DELOS Digital Library Brainstorming Meeting San Cassiano, Italy, June 2001.

F. (October 2001) Information Dynamics: An Information-centric Approach to Digital Library Interoperability (pdf). Global Digital Library Development in the New Millennium, Ching-chih-Chen, ed., Tsinghua University Press, Beijing, China, 2001, pp. 137-144.

G. (October 2001) REF Example Problem: ARM Routing in Ad-hoc Networks (pdf).

H. (October 2001) REF Example Problem: Attacker-Defender-Target Game (pdf).

I. (October 2001) REF Example Problem: Information Dynamics and Link-State Routing (pdf).

INFORMATION DYNAMICS: AN INFORMATION-CENTRIC APPROACH TO SYSTEM DESIGN[*]

Ashok K. Agrawala
Department of Computer Science
University of Maryland
College Park, Maryland 20742
agrawala@cs.umd.edu

Ronald L. Larsen
Maryland Applied Information Technology Initiative
University of Maryland
College Park, Maryland 20742
rlarsen@deans.umd.edu

Douglas Szajda
Institute for Advanced Computer Studies
University of Maryland
College Park, Maryland 20742
szajda@umiacs.umd.edu

KEYWORDS: system design, integration, analysis, information

ABSTRACT

Acquisition, distribution, management, and analysis of information are the fundamental purposes behind most complex constructed systems and infrastructures, and yet a process-centric approach is fundamental to the design and implementation of such systems. Since *information* is the essential commodity in these endeavors, we believe that an effective design should take into account the fundamental properties of information: its characteristics, its representation, its value, its temporal dynamics, its fusion, its distillation, etc. Information Dynamics is an attempt to bring a degree of rigor to the understanding of the nature of information itself and how it is used in the pursuit of system objectives.

INTRODUCTION

The most significant and far reaching technological development of the 2nd half of the last century is the development of information technology (IT). Today, IT touches nearly every aspect of our lives. And the rapid growth of new applications and new technologies assures that the trends of the 90's will accelerate in the current decade. The hardware technology exists to create ever more complex systems. However, our ability to design, implement, operate, maintain and support complex systems lags. We believe that one reason is the paradigm used in system design, namely that of the *process-centric* view of system design.

We propose Information Dynamics as an alternate paradigm in which we take an *information-centric* view. In this approach we explicitly consider the role information plays in a system, and design the system taking into account what information is needed and when, who has the information, and what happens to the information as it moves from one place to another. In an Information Dynamics framework, information is treated as a dynamic entity and its dynamics (e.g., location, currency, value) are explicitly considered. Any processing of information is referred to as "action" that is carried out under the control of "choice". Any action that carries out a transformation or related processing of information consumes resources, therefore requiring resources for some period of time, i.e., occupying some subspace in resource/time space. All actions take time and, therefore, have an impact on the dynamics of information that has to be considered in the design of a system. "Choice" defines the control mechanisms for actions in this framework. Further, we associate a "value" (sometimes called "utility") for the information in a particular "Context" and recognize that the value of information typically changes with time within a given context.

The use of information for effecting control and other decision processes is, of course, not new. Physical systems respond adaptively to linear-quadratic-gaussian (LQG) controllers, for example, when the physics is well understood and controllers have rigidly compartmentalized responsibilities. But decision making in network-based, distributed systems for which there is no nice physics poses a different class of problem. Early investigations into distributed decision making lead to a wealth of research in game theory and later team theory (e.g. Bascar

[*] This work is supported partly by DARPA / Rome Labs, Department of the Air Force, under contract F306020020578 to the Department of Computer Science at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force, DARPA, DOD, or the U.S. Government.

and Olsder 1982; Greenwald 1998; Owen 1968), but none of this work explicitly considered the temporal effects on the value of information, and how that affected system performance. Recent literature on autonomous agents (e.g. Stone and Veloso 1998; Washington 1998) reports work in distributed, multiagent decision making (using, for example, state space approaches such as Markov Decision Processes) but still fails to consider the temporal issues underlying the usage and value of information. So, while information, as an entity, is used in many disciplines, and differently in most of them, we are not aware of any approaches that explicitly approach the problem with a temporal perspective. We believe that Information Dynamics is not an incremental extension to any existing work, but, informed by this work, takes an orthogonal look at the problem.

In the following sections, we discuss aspects of the nature of information, its dynamic properties, and some of the implications for complex system design. We follow with an example of the theory applied to a networking problem.

WHAT IS INFORMATION?

We all have an intuitive notion of what information is, but making this precise is hardly intuitive. Information is a property, characteristic, or description of something physical, logical, virtual, or conceptual. That "something" may be other information. It may be a group, an action, a choice. Or it may be a relationship between any of these things.

Information cannot exist in isolation and has no value without a context. It refers to some entity that can be logical or physical, can be another piece of information, can be information about information, etc. Relationships between information may be direct or indirect, and exist whether they are enumerated or not. Relationships may be static or dynamic.

The causality principle applies. Information in the present can only affect the future; it cannot change the past. It may change the interpretation of the past, but it cannot change the past, itself. Further, delays involved in the movement of information assure that our knowledge of remote entity is necessarily delayed; the "present" may (and typically will) actually reflect a system's state at some past time instant.

As any system consists of a number of components, considering all possible pieces of relevant information and their relationships yields an arbitrarily large amount of information. In a typical system design only a small amount of relevant information is collected and used.

Note that when two pieces of information are related, their relationship may be considered a higher level of information. Processing establishes the validity of a relationship, or it may be used to derive one piece of information from the other. When the *relationship* is explicit, one piece of *explicit* information yields the second piece *implicitly.* In this regard, we consider information that can be derived from other information based on known relationships as implicit information. Note that deriving such implicit information, i.e. making it explicit, requires some processing for carrying out the interrelationship calculations.

Information Has Value

Every piece of information can have value *within a given context*. The value of information depends on its use and/or purpose. This is the role of context. Clearly the value of information changes with time and depends on the context and the frame of reference. The value can, and often will, depend on its relationships to other pieces of information. Within a context, information value can typically be quantified.

Value Of Information Is Time Dependent

The value of information typically decreases with time within a context. If the information is static, its value may also be static (but will change with context). The value of a piece of information may increase or decrease due to the discovery or instantiation of new relationships, or due to the transformation of some information from implicit to explicit.

Information Variable

We use the term "information variable" to refer to a piece of information and its associated metadata. An information variable consists (at least) of the following:

- Descriptor: Qualitatively defines the information variable and supplies the rules for interpreting its "magnitude".
- Type: "primitive" or "composite". Here primitive refers to an information variable as a basic element of information whereas composite refers to an interrelationship.
- Magnitude: a bit string associated with the information variable (traditionally referred to as its value, but that term is used in another way here). Not all information is quantifiable. For such information, the bit string may represent a text string, or any other symbolic form. The Descriptor provides the rules for interpreting the string.
- Provenance: a time indicator, showing the time when this information was collected or acquired or the time it refers to. It may also include a location indicator showing the location the information refers to or the location from which it was acquired.

- Confidence Indicator: defines the confidence measure in the magnitude. This measure is an indicator of the quality of the magnitude without any regard for the use that may be made of this information.
- Context vector: defines the context within which the information is relevant, enabling a relevance computation (e.g., cosine or dot product) to establish the importance of this information variable to the task at hand. The importance indicator may be only a subjective measure of the importance of the information.

The confidence indicator and context vector are both required in order to compute the value of information for this variable. The confidence indicator does not change with time but the context vector may need to accommodate the age of information or the passage of time.

For an information variable we can define two additional functions. The first is a time function that indicates how the context vector changes over time. This may be a probabilistic function. The second is a fusion function, which defines how two pieces of information can be combined, or "fused."

Note that while all of these components are required to fully define an information variable, many times only some of these components may be known or available. Others are either left unspecified or undefined and, hence, unused.

REPRESENTATION OF INFORMATION

We have used the term information to refer to abstract, or ideal, information (e.g., the concept behind an integer two). In order to carry out any manipulations on information it is essential to have a representation for it (e.g., a decimal symbol "2" or binary symbol "10"). Such representations are essential to store, move, or process the information. We note that some representations may use implicit information. For example when we use a data structure, it not only defines the representation of some quantities but also some relationships.

An interesting question arises regarding the representation of implicit information. As the implicit information is expressed through the interrelationships that can be processed to make the implicit information explicit, we may consider the known relationships as the implicit information. However, for the implicit information to convey the same meaning to the sender and the receiver, the two sites have to have the same understanding of the relationships and have a common understanding of how to extract any implicit information (make it explicit) from some given explicit information. They must agree on a common *context.*

Capture Of Information

We recognize two processes for capturing information: direct observation and inference. Direct observation may be through sensing/monitoring/measurement, etc. Such observations may be made directly or indirectly. Inference typically involves the use of interrelationships to make implicit information explicit. In this process we may use induction as well as deduction to capture new information.

It is interesting to note that in this context we may treat mathematics as a framework of interrelationships with a precise description of context and applicability. A framework of deduction and induction is also specified and we use these relationships in a variety of contexts invoking the implicit information. Analytical results are nothing but specified interrelationships with a description of their applicability.

Storage Of Information

In order to store information we need a *representation* for it. To use it, we need to retrieve it. A representation suitable for storage may not contain all the components or all the interrelationships. On retrieving we may be able to calculate some of them while others may be lost forever. In particular, information relating to time can be lost unless time stamping is explicitly done. Note that storing information is an action; therefore it requires resources and takes time. And all actions generate additional information that may or may not be captured.

Movement Of information

As with the storage of information, only explicit information can be moved from one location to the other. Further, the representation used for moving the information has to be such that the receiver can interpret it correctly. This requires a common understanding between the sender and the receiver which, in turn, may require conventions, protocols etc.

Moving information is an action; it consumes resources and time. Systems use networks to move information. Of course, a number of issues have to be resolved in moving information. Consider an infrastructure that can move information from location x to location y. First we have to decide who initiates the movement of information and why. How does Y know that X has the information it needs and how does X know that Y needs that information? The knowledge about who has what information is a crucial part of the design of a distributed system. This phenomenon is quite evident on the Web today.

Let us consider some immediate implications of information movement. Moving information from location x to location y takes t_{xy} time. As a direct consequence of this, at y we can only get information from x that is at least t_{xy} old. Therefore, in a distributed system it is impossible to capture the current state of the entire system at any one location!

Value Of information: Confidence Indicator And Context Vector

We may associate a value attribute to any information using a confidence indicator and a context vector. Clearly the value of information depends on its use or purpose (what we call context). The value of information changes with time, typically decreasing with time. When the underlying system is static, the value may, likewise, remain static. Under some circumstances the value may increase with time, as later information makes it more valuable. In this regard the value of information may also be associated with the interrelationships.

The confidence indicator may be represented by uncertainty models. For example, we may be interested in the waiting time at a router. When we measure it at time t, the measured value may be very precise. However, without having any additional measurements, the estimate of the waiting time will change, and in particular its variance will increase, moving towards the steady state value and the steady state variance.

Information Fusion

In the natural world, the amount of available information monotonically increases (barring catastrophic events such as the burning of the Alexandria library). Its effective management requires techniques for reducing, or aggregating, it while maintaining its quality. One way of distilling the captured information is by retaining higher-level information reflected through relationships.

Fusion functions, one of the key elements of information dynamics, are another method for distilling information. The fusion function permits us to define methods by which multiple views of a single information variable are combined. Aspects of fusion include specifying the effects on magnitudes, values, and confidence indicators. We envision several classes of fusion functions, corresponding to the characteristics of the information under consideration. For example, if we have multiple measurements of a single variable, we may fuse these measurements using minimum variance estimates to determine a single magnitude. If, on the other hand, we have single measurements of multiple variables, we may call upon known relationships among the variables to establish a statistically valid fusion.

Using Information

The use of information requires *action*. Action can create or capture information, store it, move it, transform it, or destroy it. Information can be processed to make implicit information explicit, to initiate another *action*, i.e. defining a "choice", or to activate a physical operation as an output.

As we use it here, the term *action* refers to processing that consumes resources and takes time. As resources reside at specific locations, actions are carried out at *locations*. It typically uses information as input, and starts under the control of choice. The outcome of an action may be additional information, choice, storage of information, movement of information, or some physical results in the form of commands to actuators, etc.

Choice

We use the term choice to define the control function. Choice defines what action has to be carried out where, at what time, under what conditions, and using what resources. It is based on the information and its location/time or value.

Note that choice is generated by processing information. Therefore the relationship between *information* and *choice* must be part of the implicit information. The interrelationships between information and choice may be fixed, leading to a hardwired design as a design-time choice. When the relationships are dynamic the choice has to reflect it.

Distributed Systems

Consider the implications of information dynamics on a distributed system. Such a system has a collection of entities (processing resources) capable of carrying out certain operations. A specific distributed system, designed to carry out a specific mission, uses physical resources to carry out actions and to store and move information. When such a system is interacting with an external physical system it also has sensors and actuators.

A distributed system may be considered a collection of nodes with a defined network infrastructure for communication among them. Let us examine the participation of a node in processing. The node maintains its view of the universe in the form of "perceived reality" which is based on:

- Prior model of the Universe,
- Explicit information received and processed

The explicit information is processed to integrate it with the perceived reality and is based on the model of the universe. Depending on the model, which is nothing but a collection of interrelationships, it may permit the new information to change the model.

At any node in a distributed system, all actions are initiated using the knowledge of its perceived reality that is not always explicitly defined or represented. The explicit representations may only have been used at design time, and the final system may only contain those parts that are considered essential for operation, retaining only such relationships that may be activated at runtime.

A far-reaching consequence of the movement of information is that **the perceived reality at any node**

CANNOT be assured to be the same as the actual reality at any remote node. Transmission delays assure that information received from any remote node is, by definition, historic. Further, it is not sufficient to receive messages; they must be interpreted and processed to integrate them with the local perceived reality. While the perceived reality of a node cannot be assured to be the same as the actual reality of a remote node, it can be consistent with models of remote reality. (Lamport (Lamport 1978) considered similar issues in the context of message ordering in distributed systems. He reached a "parallel" conclusion---that when transmission delays are not negligible compared to the time between events, it is sometimes impossible to say that one event occurred before another.)

Receiving Information

Message transfer is the primary means of communication in a distributed system. Upon receiving a message, a node processes it syntactically to interpret its structure. Then it processes it semantically to interpret its content, converting it into explicit information. The explicit information is then integrated with perceived reality, potentially altering the current state and modifying the choices made for processing of implicit information.

KEYS TO INFORMATION DYNAMICS

In Information Dynamics we take an information-centric view of a system and explicitly take into account the time dependent aspect of information, the value of information, and the role of implicit information. Within this framework we organize and design a system based on the desired dynamics of information, taking into account the constraints of the dynamics of information.

We believe that Information Dynamics provides a new and vital framework for the design and implementation of complex systems. It is also useful for planning and decision making. Effective decisions must be based on the appropriate perceived reality, taking into account the model of the universe and its dynamics, along with the sources of information and their role in defining dynamic choices. Clearly the value of information in terms of confidence indicators and context vectors, along with the changes in the value of information with time, play a key role in the planning process--they reflect the dynamics of information.

EXAMPLE: ROUTING

As a concrete example of the impact of information dynamics on a practical problem, consider the link state routing in a computer network. In this method, routes are chosen to be the shortest path from the source to the destination as determined according to the current knowledge of the state of links. The common practice is for the links to periodically measure their state, determine the waiting time, and broadcast this information so that all nodes have it and can use it in route determination.

Let us now consider the basic characteristics of the performance of a link. In a typical network the link is continuously transferring packets, handling the load as it is presented to the link. If we consider this link as a server, we can characterize its steady state behavior in terms of the mean, w, and the variance, v, of the waiting time. Let w(t) be the waiting time at this particular link as measured at the link at time t. Assuming that the measurement is done correctly, the variance of this measurement, v(t), is zero. Given no additional information about the state of the link, our estimate of the waiting time $w(t_1)$ at some later time t_1 will have to be based on w(t) and our knowledge of w. This estimate will have a variance, $v(t_1)$, which will not be zero. In fact, the variance will be an increasing function of the difference t_1-t, tending towards the steady state value v. Given w(t) and v(t), the actual values of $w(t_1)$ and $v(t_1)$ can be estimated with knowledge of the stochastic behavior of the link.

In this example, the basic information variable is the waiting time estimate for the link and the variance estimate is its confidence indicator. Recognizing that the transfer of the magnitude of w(t) to any other nodes in the network takes time, we require that any new estimates be made taking into account the information dynamics of the situation. Depending on the characteristics of the link, the estimates $w(t_1)$ and $v(t_1)$ may come so close to the steady state values w and v that the new measurements will have no significant impact on the link information retained by another node. As a consequence, we can significantly reduce the communication required for supporting link-state routing while at least maintaining the quality of routing decisions, if not improve them, by taking into account the variance of the delay estimates. Each node does need the steady-state information about the links. Note that if the steady state conditions change regularly, that knowledge can also be given to individual nodes. The key impact, though, is that by taking the value of information into account, the information dynamics approach allows us to improve the design of the routing scheme.

It is important to understand that although the above example uses statistical measures, the information dynamics framework is not limited to handling quantitative information. We believe that it can be equally effective in using other forms of information including qualitative information and its value expressions.

CONCLUSIONS

The management of information is at the core of many complex systems and yet we employ a process-centric approach to the design and implementation of such systems. Since information is the essential commodity in these endeavors an efficient design should take into account the fundamental properties of information. Information Dynamics is an attempt to bring a degree of rigor to the understanding of the nature of information itself.

The first principle in understanding the generic nature of information is recognizing the distinction between information and its representation. Computer systems are only capable of manipulating representations and it is through the processing of representations that we attempt to carry out the processing of information. These representations are limited in that they capture only a very limited portion of the generic information. Moreover, the various processing steps change the nature of information in ways that are not necessarily intended or anticipated. We claim that *implicit information* must be understood and elucidated.

The second fundamental principle of Information Dynamics is that information has value. We claim that information only has value in *context*; in addition to context, value may also depend on usage, and may reflect other aspects as well. We need to better understand the role and properties of this value as we use information directly in our systems. Specifically, how does processing affect the value of information? How do movement, representation, and storage, affect its value? And what are the ramifications of this for system design?

The third fundamental principle of Information Dynamics is that the value of information changes with time. Typically it decreases but it may alternatively remain the same or even increase. Understanding the role time plays in the value of information has a clear impact on the applicability of information. Movement of information takes time. When the delay caused by movement becomes large, the impact on the value of information may be significant enough that further movement may be not only unnecessary, but may, in fact, be detrimental.

The principles of Information Dynamics presented here represent an attempt to capture a sufficient understanding of the fundamental characteristics of information to allow us to better design and implement complex systems. Although such an understanding has proven elusive, our efforts to date indicate that the proposed framework has the potential for bringing about a significant advancement in the way information is handled in systems.

REFERENCES

Bacsar, T. and Olsder, G. J., *Dynamic Noncooperative Game Theory*, Academic Press, New York, Mathematics in Science and Engineering, v.160, 1982.

Greenwald, A., "Modern Game Theory: Deduction vs. Induction," TR 1998-756, New York University, 2/24/98.

Greenwald, A., "Learning to Play Network Games," TR1998-758, New York University, 2/24/98.

Lamport, L., "Time, Clocks, and the Ordering of Events in a Distributed System," *Communications of the ACM*, Vol. 21, No. 7, pp. 558-565, July 1978.

Owen, G., *Game Theory*, Saunders, Philadelphia, PA, 1968.

Stone, P. and Veloso, M., "Using Decision Tree Confidence Factors for Multi-Agent Control," *Proceedings of the 2nd International Conference on Autonomous Agents*, ACM Press, NY, 1998, pp. 86-91.

Washington, R., "Markov Tracking for Agent Coordination," *Proceedings of the 2nd International Conference on Autonomous Agents*, ACM Press, NY, 1998, pp. 70-77.

Information Dynamics and Its Applications

Ashok Agrawala
Ron Larsen
Udaya Shankar

University of Maryland

Information Dynamics

- Information-Centric View of the World
  - What information is needed and when
  - Where the information is
  - What happens to the information as it moves from one place to another
- Consider information as a dynamic entity and explicitly consider its dynamics

Information Dynamics Principles

1. Recognize the distinction between information and its representation
2. Information has value in context
3. Value of information changes with time

Information Dynamics

- Actions
- Choices
- Perceived Reality
- Goals
- Information – implicit vs. explicit
  - Value of information – in a context

Information Dynamics REF

Objective

- A framework for agent-based systems that gives a central position to the role of information, time, and the value of information.
- The expectation is that this emphasis will lead to better design and understanding of agent-based systems.

Problem Characteristics

- Time-sensitive value of information assessments
- Non-uniformity across agents with regards to their access to and evaluation of information
- The need for and consequences of coordinated behavior
- Examples
  1. multi-agent spidering for searching and cataloguing the web
  2. dynamic information services involving collections of agents using shared portals and intranets
  3. trading agents such as shopbots and travelbots
  4. client/server agents that implement secure, fault tolerant, and adaptive communication, caching and storage on the web

Framework Components

- World
- Agents
- Perceived Reality
- Activity
- Information Value Function
- Utility Function

Current Studies

- Holiday Travel Management
- Information Dynamics of Routing in Networks

Holiday travel world

Agents of World

- Travelers
- Hotels
- Airlines
- Search engines

Resources of World

- Customer resources / constraints
  - Money
  - Schedule
  - Preferences
- Allocatable commodities
  - hotel rooms
  - staff - in hotel, airline, travel agency
  - food in hotel kitchen
  - airplane seats – owned by airline, accessible by travel agent
- Decision-making resources / constraints
  - computing time
  - network access
  - hard drive space
  - processing power (CPU usage)

Information Dynamics of Routing

Routing in dynamic networks

- Network of nodes and links used for end-to-end connections
  - Subject to time-varying up/down status and traffic intensity
  - Objective is to route packets to optimize performance
- From an information perspective, routing algorithms involve the generation and dissemination of two kinds of information:
  - Local information at a node: information about some aspect of current state of node or outgoing link.
  - Remote information at a node: the perceived reality or view (e.g., routing table) that the node maintains about the rest of the network.
- When a node receives (local or remote) information, it integrates this information into its perceived reality and sends out some aspect of its new perceived reality.
  - Applicable to all routing algorithms (e.g., link state, distance vector, multi-path, multi-copy)

Routing in dynamic networks (2)

- For a node to choose a route or next hop for a data packet, *ideally* the node should know the state of each link traversed *when* the packet gets there.
- Thus the purpose of the routing algorithm is to provide information so as to allow the node to *predict* this future state with high accuracy and low cost.

Information Dynamics View

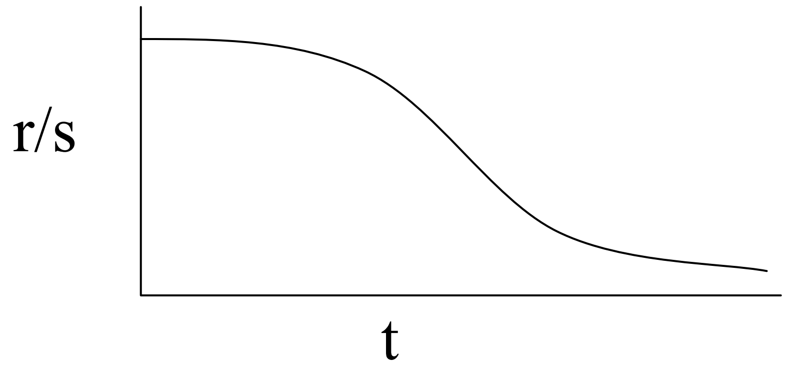
- Information elements of dynamic network
  - up/down status of node K at time t
  - traffic intensity of link J at time t
  - inter-packet time of connection L at time t
  - inter-failure time distribution of node K
  - etc
- What information best characterizes the network state and its evolution?
- How does the value of this information change with time?
- How much does it cost to disseminate this information to the nodes that need it?
- How to adapt to environmental situations to maximize value and minimize cost?

Utility and Value of Information

- Utility functions relevant to routing:
  - fraction of packets lost (minimize)
  - end-to-end delay of received packets (minimize)
  - class-based QoS (optimize)
- Value of an information element (e.g., link cost update) with respect to a Utility function is defined as the difference between
  - utility achieved by the system with the information
  - utility achieved without the information
- Because it is difficult to compute this for many utility functions, one often considers simplified utility functions, for example:
  - difference between a node's routing table and actual network state
  - link cost being indicated by queue length
  - etc

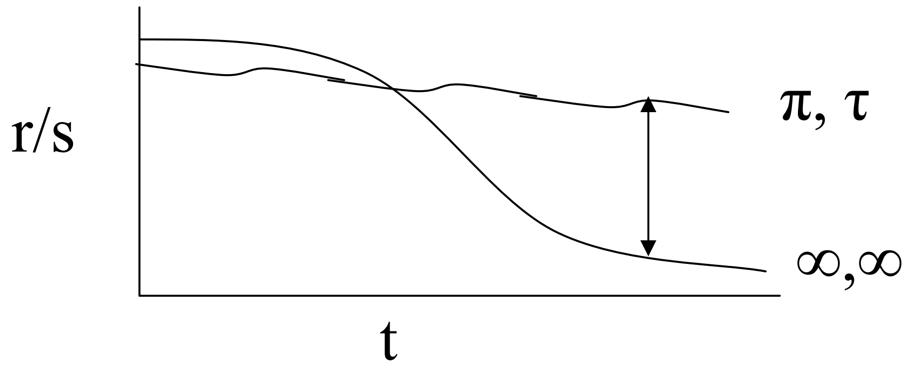
Information Value

- $U(\infty,\infty) \Rightarrow$ Routing tables are initialized but never updated



- Information value = marginal change in utility

  e.g., $U(\pi, \tau) - U(\infty,\infty)$ might look something like this:



Traditional routing approaches

- Assume that the state of a link or node in the near future is closely approximated by that in the “near past”.
- Hence routing algorithms maintain only the most recent information about a node or link, discarding all earlier history.
- Primary design trade-off is:

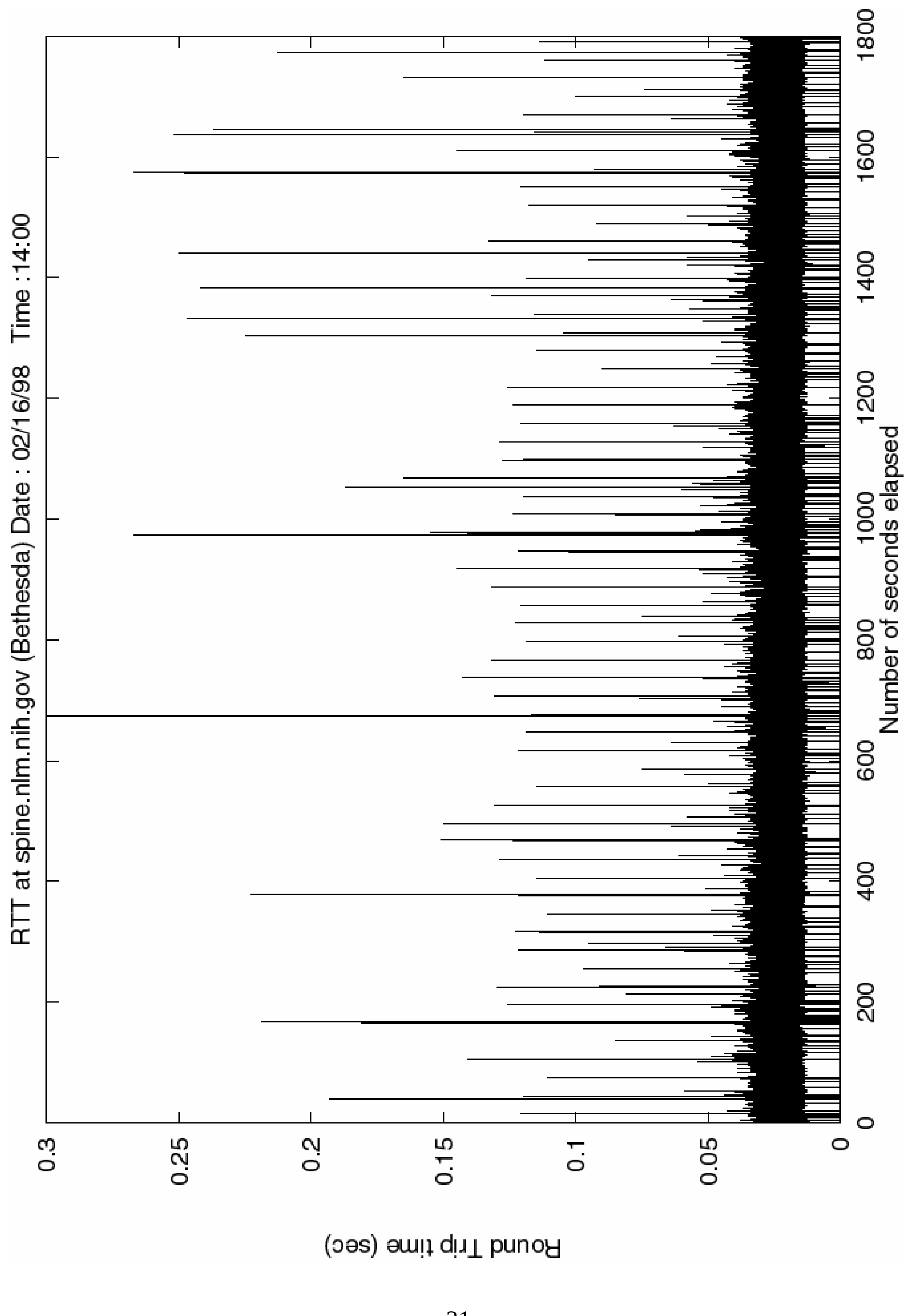
How up-to-date can the perceived realities of nodes be?

versus

How much does it cost to disseminate this information?

Information dynamics inspired questions

- **How true is it that the “near future” is similar to “near past”?**
- NetDyn experimental results demonstrate that it is not very valid.
- Thus the perceived reality should store quantities that can help predict the future state of links and nodes more accurately, for example:
  - past time evolution of statistics of links and nodes
  - steady-state statistics
  - periodicity in statistics
  - auto-correlation functions
  - cross-correlation functions across different links and nodes


RTT at spine.nlm.nih.gov (Bethesda) Date : 02/16/98 Time :14:00
Round Trip time (sec)
Number of seconds elapsed
0
0.05
0.1
0.15
0.2
0.25
0.3
200
400
600
800
1000
1200
1400
1600
1800

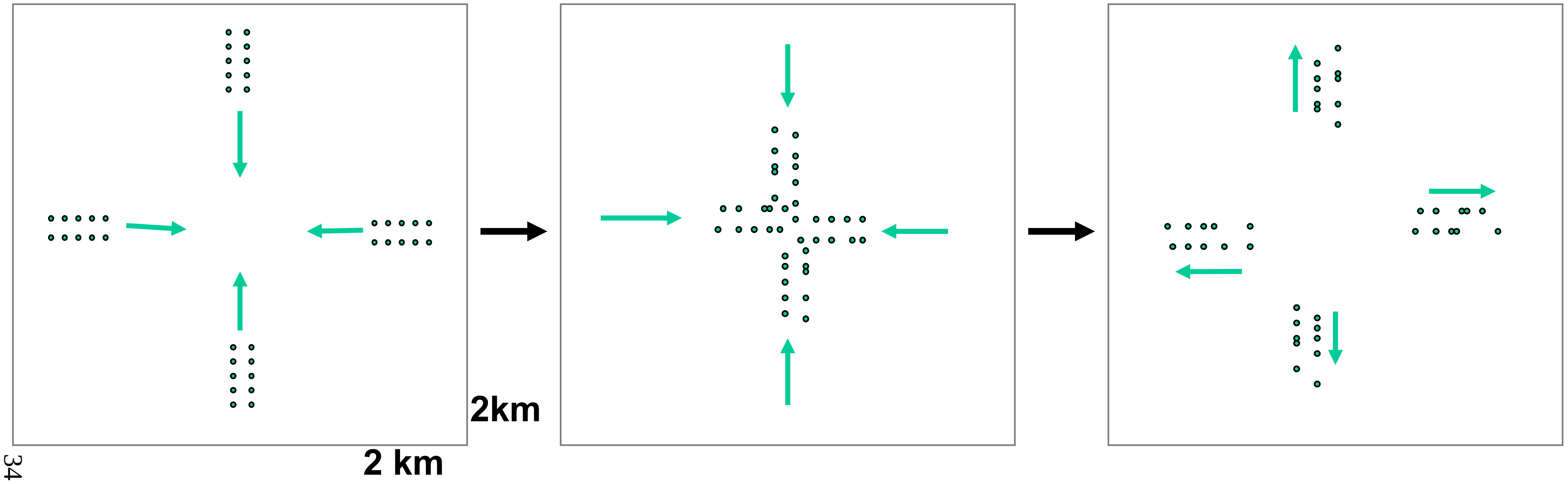
Routing with an ID Perspective

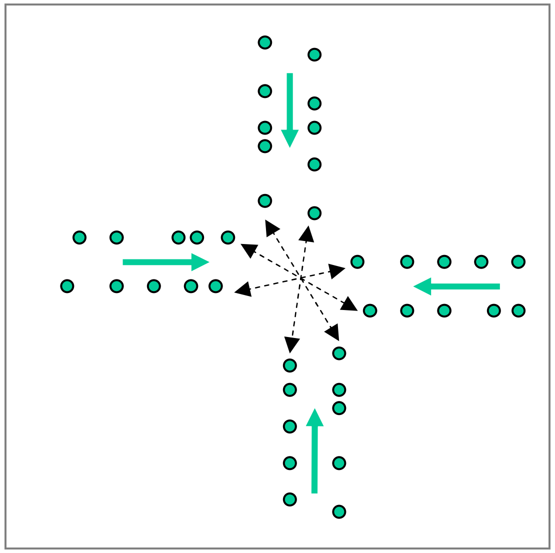
- **Adaptation**: Ad-hoc network routing protocols that Adapt to Route-demand and Mobility patterns (ARM-DSDV) *- current*
- **Optimization**: Link-state routing with optimized dissemination of information *- current*
- **Exploitation:** Dynamics of link costs *- future*
  - detect a fast moving node in an ad-hoc network.

ARM Applied to DSDV

- Each node tracks dynamic conditions (perceived reality)
  - Link status (mobility metric)
  - Traffic intensity (route-demand)
- Independently adapts to dynamics
  - Update period (based on mobility)
  - Update content (based on route-demand)
- Decentralized, well-suited to true mobility
- Simulation analysis of communication among vehicles passing through an intersection

Mobility Pattern – Highway Interchange



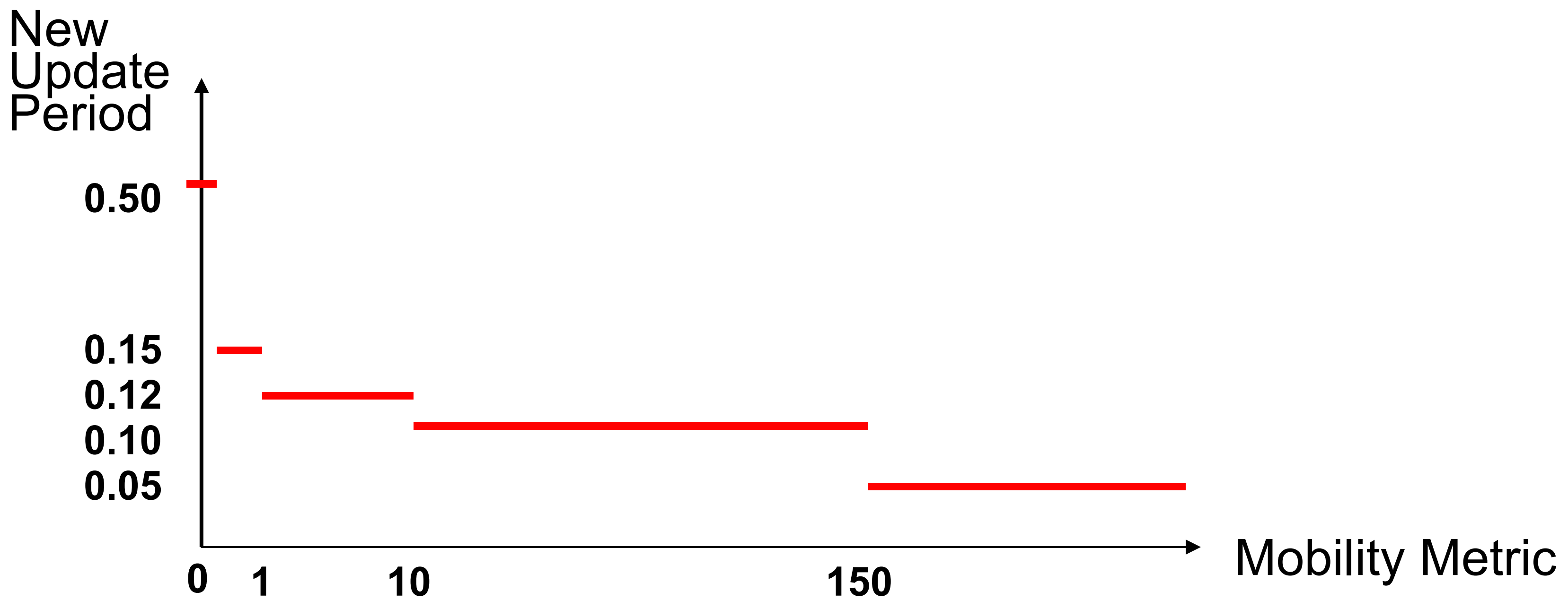


Communication peers

- 4 groups of 10 vehicles each
- 8 connections
- group speeds:
  5, 8, 9, 10 m/s

Node Performance Parameters

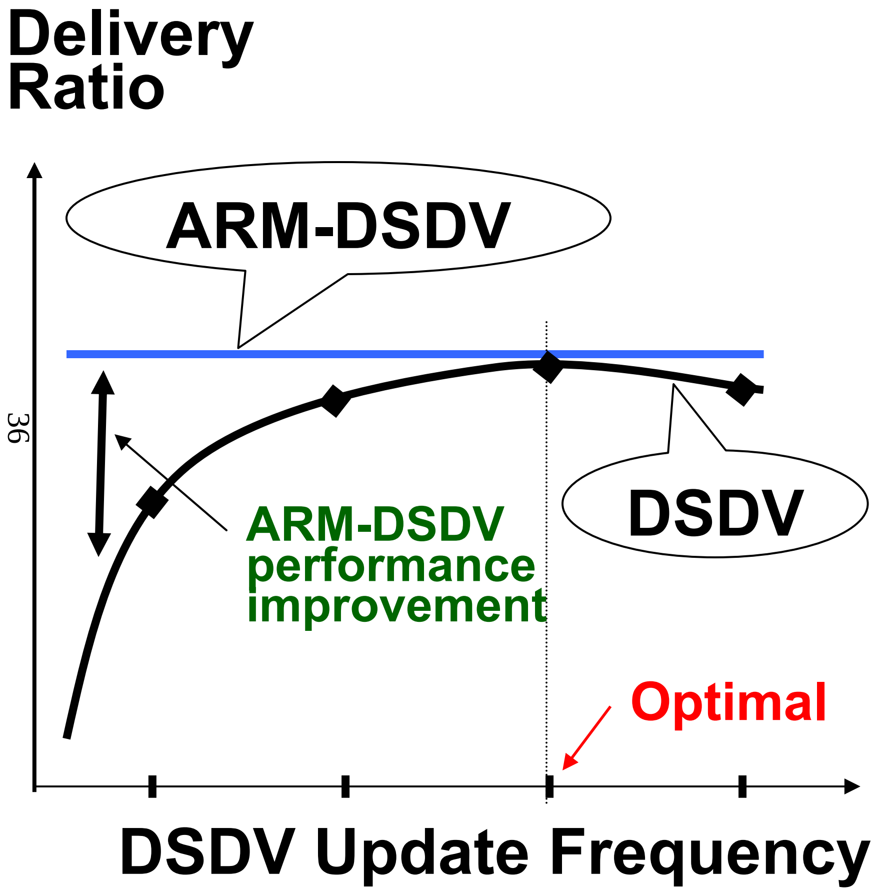
- Radio: transmission range = 100 m, bandwidth = 2 Mbps
- Connection: duration = 5 sec, 1 pkt/sec, 100 octets/pkt
- Update-period control function:

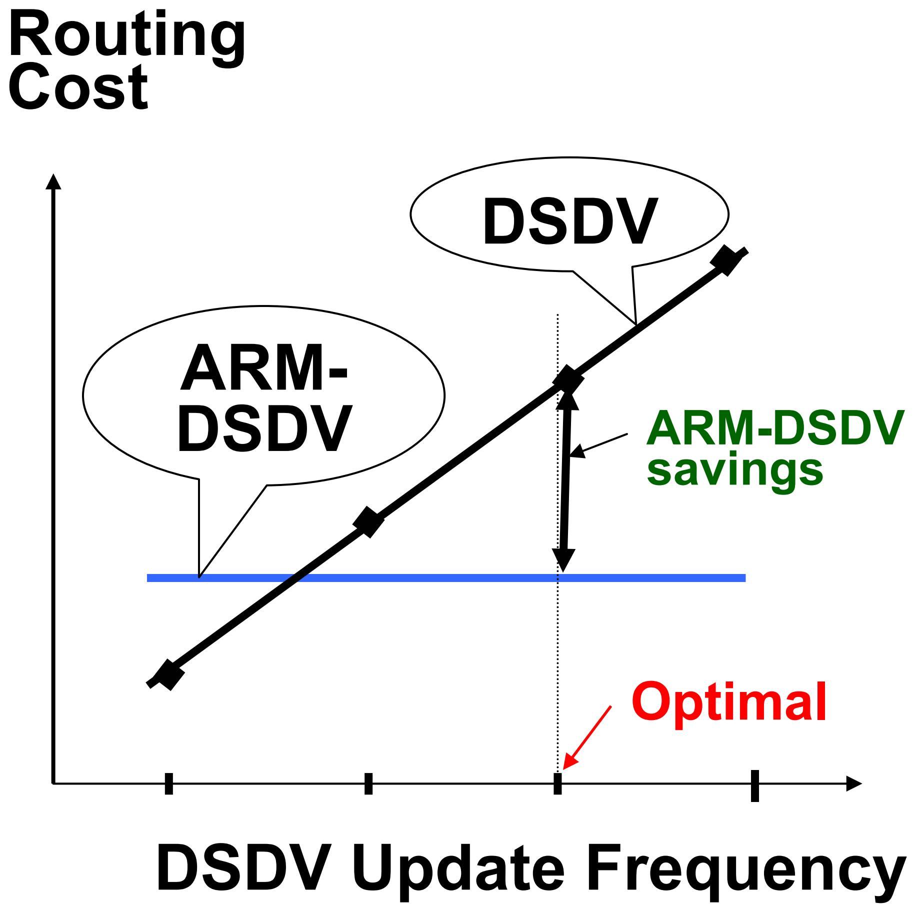


- Update-content control function:

  cutoff_recent = 3 sec
  skip every other update

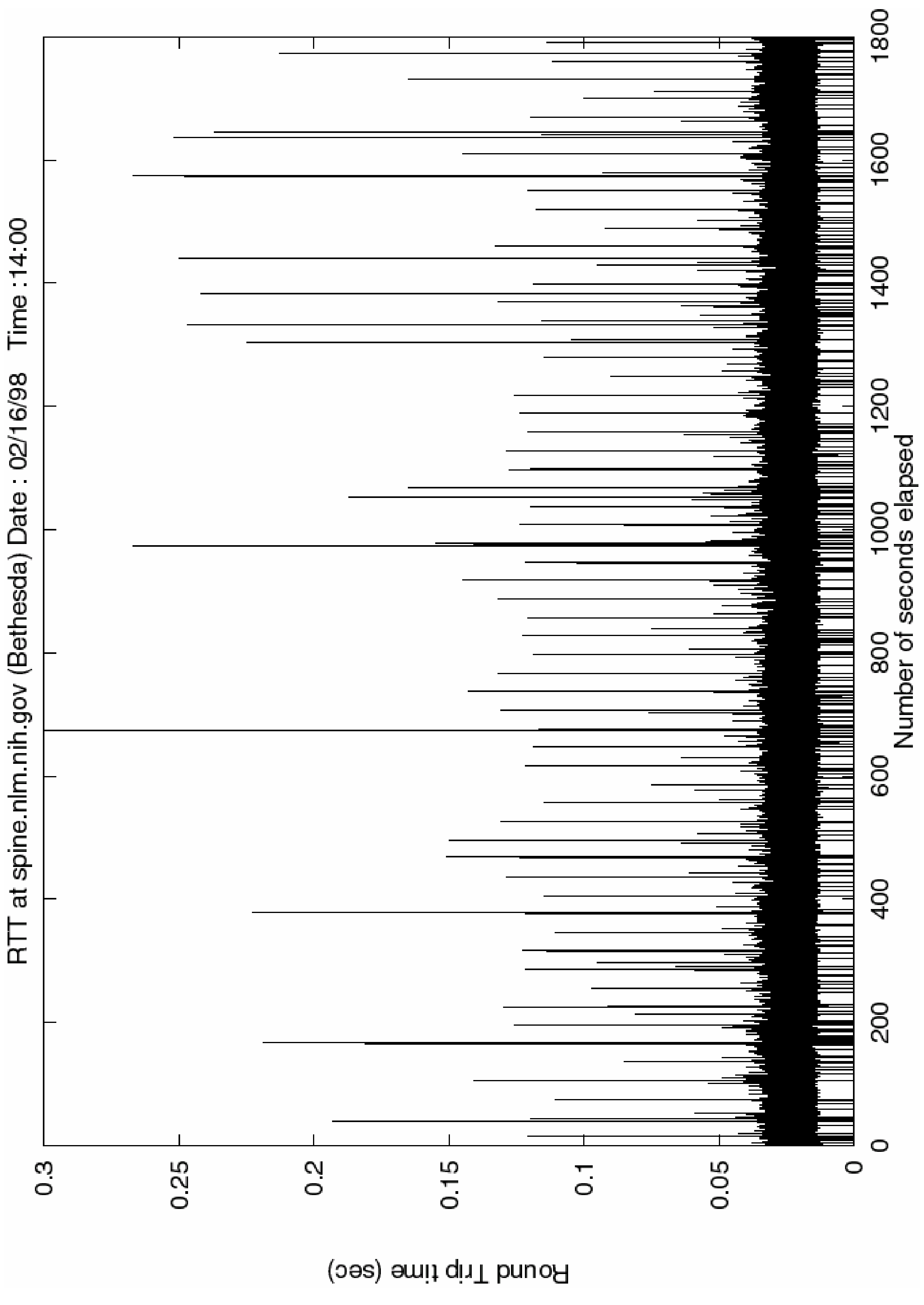
Generic Results for ARM-DSDV





Now Consider Link State Routing…

- Should we consider longer term history?
- What about that highly dynamic RTT behavior?


RTT at spine.nlm.nih.gov (Bethesda) Date : 02/16/98 Time :14:00
Round Trip time (sec)
Number of seconds elapsed
0
0.05
0.1
0.15
0.2
0.25
0.3
200
400
600
800
1000
1200
1400
1600
1800

Considering History

- Delay has a well defined periodic pattern
- Take that pattern into account in decision making
  - Routing decisions
  - Sending decisions
- Work in progress to evaluate the benefits
  - Initial results indicate that a significant improvement in performance can be obtained in the delay characteristics of packet transmission.

Short-term Link Characteristics

- Delay
  - Stochastic process
  - Make point measurements
    - Integrate over time (window)
  - How much benefit is there in sending information about delay on a link to everybody?

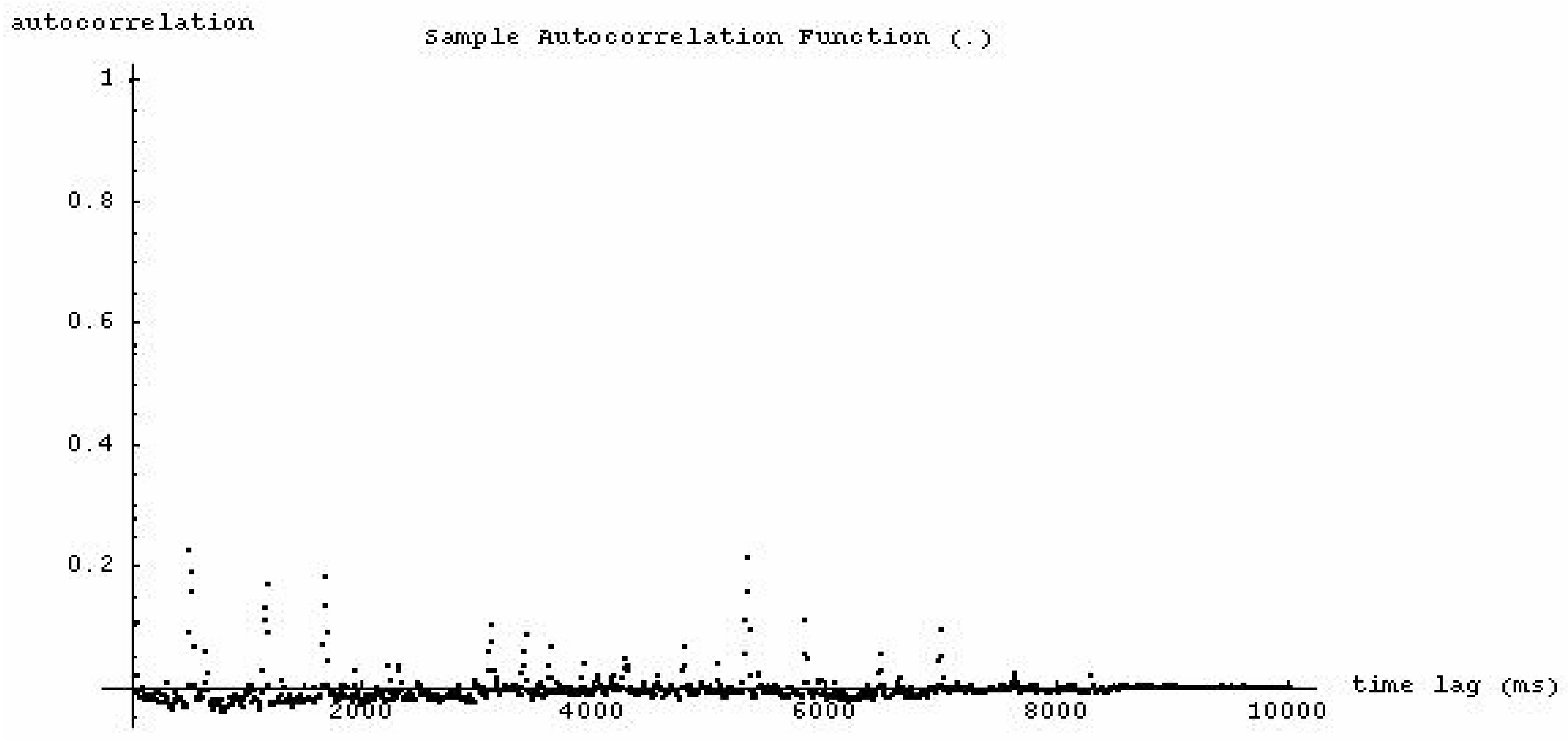
Link State Routing

- World view of a node (Perceived Reality)
  - Complete Topology
    - Information from past
  - Updates
    - Local information
      - Collected periodically
      - Flooded to all nodes
- Value of information
  - Variance

Value of Information

- Mean
  - Computed in a window
    - Fixed vs. moving
- Variance
- Estimate of mean and variance
  - Tend towards steady state values
    - How rapidly?
      - Within a *few milliseconds* close to the steady state values!!

Correlation function graph



Improving Link State Routing

- Forward information only if it is “far” from the steady state values
- Typically one to two hops
- Benefits (in terms of number of control messages sent)
  - O(N*d) to O(d*d)
    - N=100, d=3 300 vs. 9
- Verifying through simulations

Information Dynamics in Routing
Summary

Vast Solution Space

- Possible classes of routing solutions
  - flooding, limited flooding
  - random routing, hot potato routing
  - cost-based routing (assign cost to links and paths, and route on paths of minimum cost)
  - multi-path routing (having multiple paths to a destination)
  - multi-copy routing (sending multiple copies of a packet on different routes)
- Questions:
  - When to use which approach?
  - How to dynamically switch between them?
- Answer using Information Dynamics

Potential Payoff

- Clear understanding of routing algorithm tradeoffs
  - random versus deterministic routing
  - multi-path versus single-path routing
  - multi-copy versus single-copy routing
  - adaptive versus static control
- Phase-change boundaries for control settings, for example:
  - slow dynamics, light load: use single-copy, single-path
  - slow dynamics, high load: use single-copy, multi-path
  - rapid dynamics, light load: use multi-copy, multi-path, flooding
  - rapid dynamics, high load: use single-copy, multi-path, random
- Discovery of better routing protocols
  - mechanisms to monitor environment and adapt
  - make implicit information explicit and thereby increase effectiveness

Concluding Remarks

APPENDIX C

Information Dynamics REF

April 19, 2001

REF Characteristics

- Challenging
  - Framework for agent-based systems emphasizing:
    - Time-sensitive information value
    - Non-uniform information access
    - Coordinated behavior
- Relevant
  - A test-bed environment with clear association to problems of interest to the DoD
- Tractable
  - Readily available information sources and related infrastructure

Framework Components

- Networked Environment – *World (e.g., Web)*
- Actors – (*mobile) Agents*
- Actions - *What agents do*
- Information - *Perceived reality*
- Information Value - *how they choose actions*
- Utility function - *why they choose actions*

Problem

- *Application Domain*: Intelligence gathering & analysis from the Web
  - Collection, correlation, extraction
  - Confidence, reliability assessment
  - Time sensitivity
- *Scientific Domain*: Interaction theory for agent-based systems
  - Structural relationships
  - Interaction models
  - Emergent behavior
- *Anticipated Outcome*: Sound principles for developing scalable agent-based systems
  - Agent interactions at multiple scales
  - Improved group performance of agents

Driving Needs

- Warning generation
  - Crisis intervention
  - Terrorist activities
  - Unusual movement
- Situation monitoring
  - Watch-desk support
  - Persistent queries
  - Anomaly detection

Driving Questions

- The newspaper reporter questions
  - Who
  - What
  - Where
  - When
  - Why
- Where and when to look for information
- What sources to trust

REEF Partitions

- Agent Design
  - Collaborative search with multiple agents (Brown)
  - Temporal extensions and commitments (Michigan)
  - Belief, reputation, and trust establishment (Texas)
- Behavioral Analysis
  - Observations of agent interaction (SFI)
  - Explanatory and predictive models (ASU)
  - Emergent behaviors (UNM)
  - Analysis tools for extracting structure (Massachusetts)
- Instrumented Infrastructure
  - Test collections and measurement software (Veridian)
  - Information Dynamics framework (Maryland)

REEF Approach

- Conduct precise evaluation through exploration and evaluation in synthetic environment (e.g., artificial or extracted portion of Web)
  - Controlled environment provides "ground truth"
- Empirically identify real world influences on (actual) Web
  - Domain dependencies may have skewed experimental results
  - Precision and recall achievable in bounded collection may not scale to actual Web
- Enable rapid iteration between experimental environment and real world application

REF Team Collaboration

- Email list & archive
- Periodic collaboration
  - Telephone, teleconference
  - Netmeeting, Groove, Jiffy, …
- Selective face-to-face meetings, as required

Next Steps

- Application domain definition
- Corpus identification
- Infrastructure specification
- Instrumentation requirements
- Functional decomposition
- Interfaces discussion
- Minimal implementation constraints

Information Dynamics - An Information-centric Approach to Digital Library Interoperability[*]

Ronald L. Larsen
University of Maryland
College Park, Maryland 20742, USA
rlarsen@deans.umd.edu

KEYWORDS: digital library, interoperability, system design, integration, analysis

ABSTRACT

Acquisition, organization, management, retrieval, and distribution of information are fundamental purposes of digital libraries and their supporting infrastructures. Interoperable digital libraries pose particularly difficult system design issues. Interoperability research has focused largely on syntactic and semantic interoperability. In this paper, a third form of interoperability, *analytic interoperability* is proposed, with a framework in which to consider it. Since *information* is the essential commodity of interest, a comprehensive interoperability design should take into account the fundamental properties of information, including representation, composition, relationships, and dynamics. *Information Dynamics* considers how the nature of information can be used to achieve analytic interoperability.

INTRODUCTION

The growth of networked information resources, largely through the Internet and the World Wide Web, is both a result and a source of the growing interest in digital libraries. Digital libraries have emerged as the vehicle for organizing collections of digital information, much as traditional libraries have done for print and related media. They are becoming a major component of the global information infrastructure. But little standardization exists among digital libraries, and it can be argued that the international standards process is poorly suited to the rapid pace of technology development that has become familiar on the Web. Alternatively, developers of digital libraries focus more on *interoperability* among heterogeneous, or federated, systems. Andreas Paepcke [i] describes these as "cooperating systems where individual components are designed or operated autonomously." He suggests that "the ultimate goal for such a system is to have components evolve independently, yet to allow all components to call on each other efficiently and conveniently."

The rapid advancement of digital libraries throughout the 90's and the near-daily announcement of new technologies and systems all but assure that these trends will accelerate in the current decade. The hardware technology exists to create ever more complex networks of systems. However, our ability to design, implement, operate, maintain and support these increasingly complex systems lags. One reason for this is the paradigm used in system design remains largely *process-centric,* founded on economic principles inherited from many generations past of Moore's Law. Digital libraries provide the need and the opportunity to design around *information-centric* principles.

[*] This work is supported partly by DARPA / Rome Labs, Department of the Air Force, under contract F306020020578 to the Department of Computer Science at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force, DARPA, DOD, or the U.S. Government.

Information Dynamics[ii] is an alternate design paradigm that takes such an information-centric perspective. In this approach the role information plays is central; system design considers what information is needed and when, where the information is, and what happens to the information as it moves from one place to another. In an Information Dynamics framework, information is treated as a dynamic entity and its dynamics (e.g., location, timeliness, value) are explicitly considered. Information processing is performed through *actions* carried out as a result of explicit *choices*. Any action that carries out a transformation or related processing of information consumes resources, requiring these resources over some period of time. One can think of this in terms of actions that occupy a subspace in resource/time space. All actions take time and, therefore, have an impact on the dynamics of information. Further, information is considered to have value within a *context* and with respect to the achievement of a *goal*. To this end, the value of information typically changes over time within a given context.

The use of information for effecting time-dependent control and related decision processes is not new. Physical systems respond adaptively to linear-quadratic-Gaussian (LQG) controllers, for example, when the physics is well understood and controllers have rigidly bounded responsibilities. But decision making in network-based, distributed systems for which there is no nice physics poses a different class of problem. Early investigations into distributed decision-making led to a wealth of research in game theory and later team theory, [iii,iv,v,vi] but none of this work explicitly considered the temporal effects on the value of information, and how that affected system performance. Recent literature on autonomous agents[vii,viii] reports work in distributed, multi-agent decision-making (using, for example, state space approaches such as Markov Decision Processes) but still avoids consideration of the temporal issues underlying the usage and value of information. Interoperability research has distinguished between *syntactic* and *semantic* interoperability. While the boundary between them is blurry, syntactic interoperability typically refers to agreement on structural relationships within communication, while semantic interoperability addresses common interpretations of term usage and meaning. But this speaks to *how* information is shared among heterogeneous components. While information is used throughout these approaches, and differently in most of them, techniques that explicitly consider the role of the time/value of information (the *when/what*) and the reason for communicating (the *why*) are lacking. Analytic interoperability addresses these topics by considering intentions, or understanding what the purpose of the interaction is and using this to optimize actions.

In this paper, the dynamic nature of information is considered with regard to decision processes and the resulting implications on the design of interoperable federated digital libraries. For illustrative purposes, information dynamics is first applied to a networking problem.

WHAT IS INFORMATION?

Information is a property, characteristic, or description of something physical, logical, virtual, or conceptual, including other information. It may be a group, an action, a choice, or a relationship between any of these things. Information has value *within a context*. Relationships between information may be direct or indirect, and exist whether they are enumerated or not. Relationships may be static or dynamic. The causality principle applies. Information in the present can only affect the future; it cannot change the past. It may change the interpretation of the past, but it cannot change the past, itself. Further, delays involved in the movement of information assure that knowledge of a remote entity's state is necessarily delayed; the "present" system state as understood by any system entity will, therefore, actually reflect the collective state of system components at past time instant(s).

Every digital library consists of a large number of functional components and a much larger number of digital objects (the contents). Considering all possible pieces of relevant information and their relationships yields an arbitrarily large amount of information. In a typical system design only a small amount of relevant information is collected and used.

When multiple pieces of information are related, their relationship may be considered a higher level of information. Action is required to establish the validity of a relationship. When the *relationship* is explicit, the related information can be derived *implicitly.* In this regard, information that can be derived from other information based on known relationships is implicit information.

Information Has Value

Information has value (or *utility*) within a given context. Context can be considered the domain of the utility function relevant to the task at hand. The value of information may change with time within a given context, and its value may differ in different contexts at the same time instant. Information value may also depend on relationships with other information. Information is represented in *information variables* that include the item of information and associated metadata. A context vector defines the domain within which a relevance computation (e.g., cosine or dot product) can be made in order to establish the importance of the variable to the task at hand. Metadata associated with a variable may also include a measure of confidence (e.g., a probability distribution function) by which the significance of the variable can be assessed. The value, or utility, of the variable is a function of both context and confidence, both of which may also be a function of time. Note that while both context and confidence are formally required to understand the role of an information variable in a system, they are rarely considered explicitly in the design of systems.

Explicit information may be acquired through direct observation, communication, or inference. Only explicit information can be communicated. Implicit information inferred from known relationships by different observers or by communicating entities may still differ, unless they also share a common context and a common understanding of relationships.

Communication requires agreement on representation and protocol (the *how*), but that is only the beginning. More fundamental is the determination of *what* is to be communicated, *when*, and *why;* for every action, including communication, consumes resources and time. Knowledge of what information is required, and where and when it is needed, is a crucial part of federated system design. But for a federated digital library system, it is also the very reason for its existence.

Federated Systems

Consider the implications of information dynamics on a federated system. Such a system has a collection of entities (processing resources) capable of carrying out certain operations. A specific distributed system, designed to carry out a specific mission, uses physical resources to carry out actions and to store and move information. When such a system is interacting with an external physical system it may also have sensors and actuators.

A federated system is a collection of autonomous nodes with an interconnecting infrastructure for communication. Each node maintains a *perceived reality,* based on its prior model of the operating environment and explicit information it receives. Explicit information can be integrated into the model to update the perceived reality. Perceived reality at no node can ever be assured to be identical to *actual reality*, particularly with respect to non-local events. Transmission delays assure that information received from any remote node is, by definition, historic. Further, it is not sufficient to receive messages; they must be interpreted and integrated into the local perceived reality.

Information Dynamics incorporates an information-centric system view that explicitly considers temporal aspects of information, the value of information, and the role of implicit information. Within this framework, a system can capitalize on dynamic system behaviors that would otherwise be liabilities. Decisions are explicitly based on perceived reality, taking into account the environmental model, its dynamics, information sources, and their interactions. The value of information, conditioned on confidence levels and context vectors, plays a key role in system operation.

EXAMPLE: ROUTING

As a concrete example of the application of information dynamics to a practical problem, consider link state routing in a computer network. Shortest path routes from a source to a destination are determined according to the current known state of links. Routing algorithms typically measure their state (e.g., queue length) periodically, estimate the waiting time, and broadcast this information to neighboring nodes, which use it for best-route determination.

Consider the basic characteristics of link performance. In a typical network each link continuously transfers packets, as presented, up to the capacity of the link. Considering a link as a server, its steady state behavior can be characterized in terms of the mean, w, and the variance, v, of the waiting time. Let w(t) be the waiting time at a particular link as measured locally at time t. Assuming that the measurement is done correctly, the variance of this measurement, v(t), is zero. Given no additional information about the state of the link, the estimated waiting time $w(t_1)$ at some later time t_1 will necessarily be based on w(t) and our knowledge of w. This estimate will have a variance, $v(t_1)$, which will be nonzero. In fact, the variance will be an increasing function of the difference t_1-t, tending towards the steady state value v. Given w(t) and v(t), the actual values of $w(t_1)$ and $v(t_1)$ can be estimated with knowledge of the stochastic behavior of the link.

In this example, the basic information variable is the waiting time estimate for the link and the variance estimate is its confidence indicator. Recognizing that communicating w(t) to another node in the network takes time, any new estimates should take into account the dynamics of the situation. Depending on the characteristics of the link, the estimates $w(t_1)$ and $v(t_1)$ may come so close to the steady state values w and v that the new measurements will have negligible impact on the link information retained by another node. As a consequence, communication can be significantly reduced for link-state routing without decreasing the quality of routing decisions by considering the variance in delay estimates. Each node does need steady-state information about the links. Note that if the steady state conditions change regularly, that knowledge can also be shared. By explicitly considering the value of information in this simple algorithm, information dynamics improves the design of the routing scheme.[ix] Early results suggest a savings of at least 25% in routing control information is achievable using information dynamics approaches to link state routing.

It is important to understand that although the above example uses statistical measures, the information dynamics framework is not limited to handling quantitative information. It can be equally effective in using fuzzy or purely qualitative information.

DIGITAL LIBRARY IMPLICATIONS

A digital library has been defined as “the collection of services and information objects that support users in dealing with information objects and their organization and presentation, available directly or indirectly via electronic means.”[x] This is a sufficient working definition; digital libraries allow individuals and organizations to efficiently and effectively identify, assemble, correlate, manipulate, and disseminate information resources, regardless of the medium in which the information exists. Digital libraries provide tools to navigate and manipulate information in a multimedia, multilingual, multidisciplinary world. Task context, user values, and information provenance are critical elements in the information seeking process, but have yet to become part of the digital library infrastructure.

But how might information dynamics concepts be introduced into digital library design? Consider the information retrieval functions of digital libraries. A user formulates a query from a client entity, which sends the query to a search engine operating over some set of repositories. The repositories use the query terms to suggest materials in the local collection that may be responsive to the query, and a ranked list of responses is constructed. Either in middleware or in the client, itself, the responses from the multiple repositories are merged into a ranked list that is presented to the user, who is then responsible for perusing the list in the hope of finding relevant materials. Much effort has gone into developing high performance search engines, typically measured by *precision* and *recall* over a test corpus. But while precision and recall are used to evaluate performance in carefully constructed test scenarios, the results of these evaluations have not typically been used to *control* search engine performance. Consider, for example, the results of the TREC6 Conference shown in Figure 1, which is a typical display of state-of-the-art performance for information retrieval engines. While curves with higher precision and recall are generally superior, when these systems are placed into operation, the user has no control over where on these curves the system will perform for a particular search. Control parameters are set within the system implementation and are totally opaque to the user.

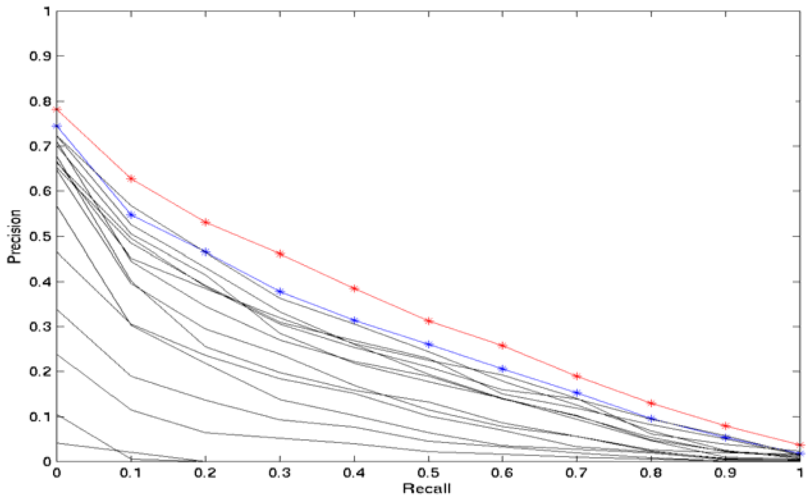


Figure 1. Precision & Recall curves from the TREC6 Conference [xi]

Information dynamics brings to the fore this kind of trade-off. It explicitly recognizes that each operation is unique, but that there is information available to tailor system performance to the specific character of the operation. For example, consider providing in the user’s query a parameter that alludes to the purpose (or goal) of the query. It may be that the user is only casually familiar with the field and needs introductory material. The search would be more effective, perhaps, by applying substantially more weight to precision than to recall. As the nature of the query moves from casual interest through tutorial (instructional), fact-finding (known-item search), research (focused inquiry), to survey (comprehensive exploration), the search engine could deliver more relevant responses by progressively shifting its weighting more toward recall and away from precision.

In like manner, information dynamics suggests strategies to incorporate *context* into queries. Query term ambiguity results in false returns from search engines. If the user incorporates multiple terms into a query, or is encouraged to do so through iterative refinement of a query by responding to irrelevant returns from poorly framed queries, the detrimental effects of term ambiguity can be reduced. But information dynamics suggests an alternate approach, capitalizing on the term disambiguation refinement work of ontologies and *entry vocabularies*.[xii] If the search engine is given sufficient information to limit the domain of search, there is reason to believe that the precision of returned results would improve.

Information retrieval operations in digital libraries are largely state-less transactions, particularly across multiple patrons. In stark contrast to the way reference librarians learn and improve their performance, later users of search engines rarely benefit from earlier searches conducted by those engines. But they could. Just as caching improves communications performance by capitalizing on temporal aspects of information demand, information retrieval can benefit from prior searches of like nature with a similar context. This represents the very early stages of analytic interoperability.

Digital library research has touched on analytic interoperability, without using the term. Several research projects have examined implicit and explicit collaborative techniques for improving an individual's success in information retrieval, for example, by capitalizing on prior search activities conducted by other individuals.[xiii] [xiv] Query languages and tools identify digital library materials across federated collections that are similar to the characteristics expressed in the query. These characteristics focus on the information artifact and are only beginning to consider non-bibliographic attributes to improve the search. Examples include identifying the types of individuals who have been reading specific material, the value they associated with it, and the paths they traversed to find it. These approaches require instrumented digital libraries, in order to build the perceived reality and set the context that will enable improved performance at the user-level. This goes beyond issues of functionally compatibility among federated systems, to a mission-oriented control structure designed to improve both the qualitative and quantitative performance *as perceived directly by the end-user*.

Digital libraries face significant technology challenges. These include real time ingest (capturing, interpreting, cataloging, and indexing high rate multimedia data flows in real time), federating distributed repositories (organizing heterogeneous distributed information sources into comprehensive discipline-oriented, user-accessible repositories), and cross-lingual interaction (automatically accessing and using information across multiple natural languages).[xv] Information dynamics holds the potential to raise the level of interoperability among users and digital libraries from a high dependence on syntax, structure, and word choice to greater exploitation of semantics, context, and concepts, thereby extending information search and filtering beyond purely bibliographic criteria to include contextual criteria related to the task, to the user, and to the time constraints of the user.

Scalability and interoperability are well-known, fundamental requirements for digital libraries. Scalable repository technology must support the federation of thousands of repositories, present to the user a coherent collection of millions of related items, and do this rigorously across many disciplines. Information dynamics holds the potential of addressing these issues with more than brute force bandwidth and capacity. As the size and complexity of information objects increases, so does the bandwidth required to use these objects. Time-critical applications requiring real-time interactivity push the bandwidth requirements even higher. *Broadband interoperability* refers to the dramatic changes in the user's work style that become feasible when the user's inputs are no longer constrained to a few keystrokes or mouse clicks. Information-centric design founded on context, utility (or information value), and temporal relationships offer the potential for real-time adaptation of scalable network and repository services

CONCLUSIONS

Whereas *information* management is the mission of complex systems, *process* management remains the dominant design and implementation approach. Since information is the essential commodity, effective design strategies should explicitly address the fundamental properties of information. The first principle is to recognize the distinction between information and its representation. Computer systems are only capable of manipulating representations and it is through the processing of representations that we attempt to carry out the processing of information. These representations are limited in that they capture only a limited portion of the generic information. Moreover, processing changes the nature of information in ways that are not necessarily intended or anticipated. *Implicit information* must also be understood and elucidated.

The second fundamental principle of Information Dynamics is that information has value in *context*. Processing affects the value of information. Movement, representation, and storage also affect information value. But the ramifications on system design are rarely considered. The third fundamental principle of Information Dynamics is that the value of information changes with time. Understanding the role time plays in the value of information impacts the applicability of information. Communication of information takes time. When the delay caused by communication becomes large, the value of the information may be reduced sufficiently that its communication may not only have been unnecessary, but may, in fact, be detrimental.

The principles of Information Dynamics presented here represent ongoing work to understand the fundamental characteristics of information within a federated system context. The objective is to develop information-centric models of system design and operation. The framework has shown the potential for bringing about a significant advancement in the way information is handled in systems.

---

[i] Paepcke, A., S. Chang, et al., "Interoperability for Digital Libraries Worldwide," Communications of the ACM, Volume 41, 4, April 1998.

[ii] Larsen, R., A. K. Agrawala, and D. Szajda, "Information Dynamics: An Information-Centric Approach to System Design," International Conference on Virtual Worlds and Simulation, San Diego, CA, January 2000.

[iii] Bacsar, T. and Olsder, G. J., *Dynamic Noncooperative Game Theory*, Academic Press, New York, Mathematics in Science and Engineering, v.160, 1982.

[iv] Greenwald, A., "Modern Game Theory: Deduction vs. Induction," TR 1998-756, New York University, 2/24/98.

[v] Greenwald, A., "Learning to Play Network Games," TR1998-758, New York University, 2/24/98.

[vi] Owen, G., *Game Theory*, Saunders, Philadelphia, PA, 1968.

[vii] Stone, P. and Veloso, M., "Using Decision Tree Confidence Factors for Multi-Agent Control," *Proceedings of the 2nd International Conference on Autonomous Agents*, ACM Press, NY, 1998, pp. 86-91.

[viii] Washington, R., "Markov Tracking for Agent Coordination," *Proceedings of the 2nd International Conference on Autonomous Agents*, ACM Press, NY, 1998, pp. 70-77.

[ix] Ahn, Sungjoon and A. Udaya Shankar, "An Application of the Information Dynamics Framework: Adapting to Route-demand and Mobility (ARM) in Ad hoc Network Routing," Computer Science Department, University of Maryland, March 2001.

[x] Leiner, Barry M., "The Scope of the Digital Library," Draft prepared for the DLib Working Group on Digital Library Metrics, January 16, 1998, Revised October 15, 1998, http://www.dlib.org/metrics/public/papers/dig-lib-scope.html

[xi] See the proceedings of the Sixth Text Retrieval Conference (TREC) at http://trec.nist.gov/

[xii] Buckland, M., Chen, A., Chen, H., Gey, F., Kim, Y., Lam, B., Larson, R., Norgard, B., and Purat, Y. "Mapping Entry Vocabulary to Unfamiliar Metadata Vocabularies," D-Lib Magazine, Vol.5 No.1, January 1999.

[xiii] Kantor, Paul B., Endre Boros, Benjamin Melamed, Vladimir Meñkov, "The Information Quest: A Dynamic Model of User's Information Needs," Rutgers University, http://aplab.rutgers.edu/ant/

[xiv] See http://www.packhunter.com

[xv] Larsen, Ronald L., and Jean Scholtz, "Digital Libraries and Scholarly Communication," to be published.

Information Dynamics & Interoperability

Presented at:
NIT 2001
Global Digital Library Development in the New Millennium
Beijing, China, May 2001, and
DELOS Digital Library Brainstorming Meeting
San Cassiano, Italy, June 2001

Ronald L. Larsen
University of Maryland

The MIND Lab
The Maryland Information and Network Dynamics Lab

Digital Library

- "the collection of services and information objects
- that support users in dealing with information objects and their organization and presentation,
- available directly or indirectly via electronic means."

Barry M. Leiner, "The Scope of the Digital Library," DLib Working Group on Digital Library Metrics

The MIND Lab
The Maryland Information and Network Dynamics Lab

Heterogeneous, Federated Systems

- "… *cooperating* systems where individual components are designed or operated autonomously."
- "… components evolve independently"
- "… all components call on each other efficiently and conveniently."

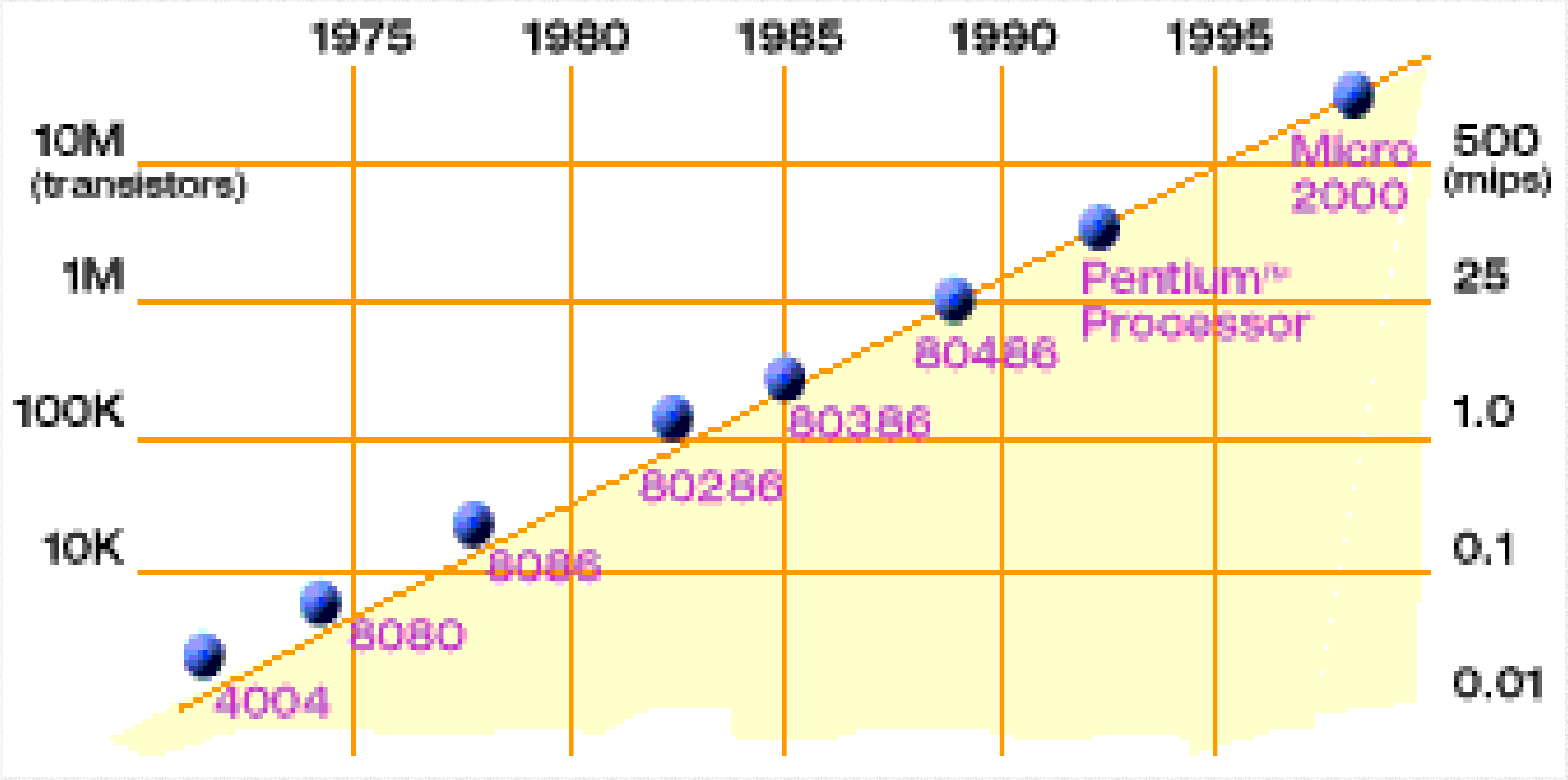
Andreas Paepcke, et. al., "Interoperability for Digital Libraries Worldwide," CACM 41,4

The Maryland Information and Network Dynamics Lab

Things we all know…

- Processor power
- Disk storage
- The Web
- Digital libraries

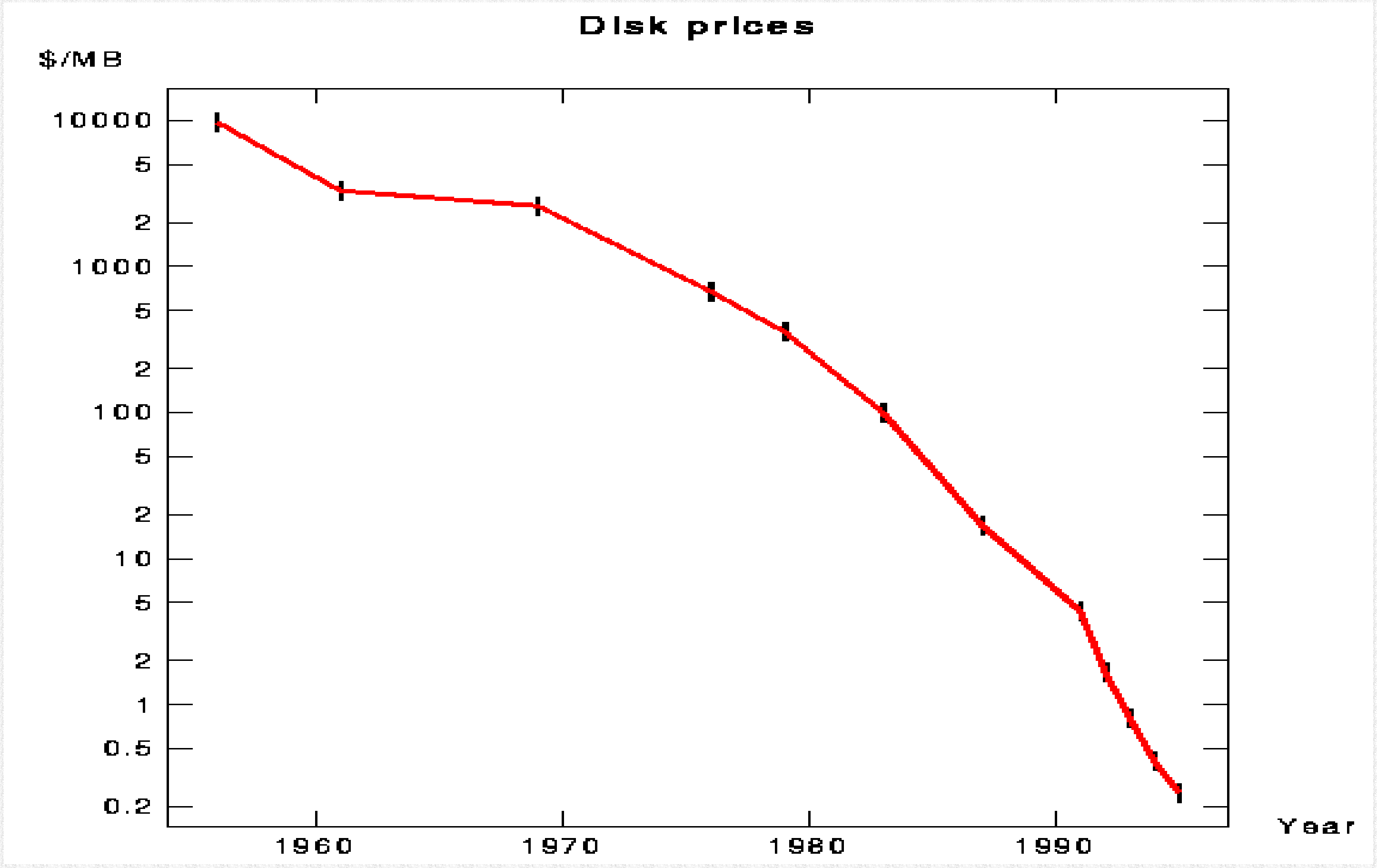
Processor power…
once precious, now ubiquitous



http://www.intel.com/intel/museum/25anniv/hof/moore.htm

The Maryland Information and Network Dynamics Lab

Disk storage… once costly, now nearly free

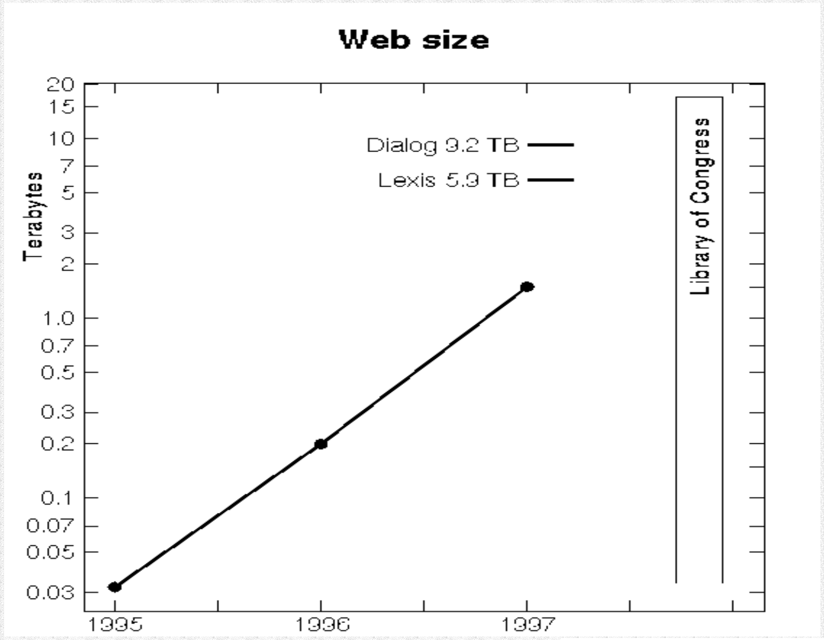


http://www.lesk.com/mlesk/diskprice.gif

The Maryland Information and Network Dynamics Lab

The Web… once a novelty, now a necessity

- World Wide Web contains ~20 terabytes of information in ~ 1 billion documents
- "Deep web" contains ~8,000 terabytes in ~550 billion documents



http://www.completeplanet.com/tutorials/deepweb/summary03.asp
http://www.lesk.com/mlesk/ksg97/ksg.html

The Maryland Information and Network Dynamics Lab

Digital Libraries…
once technical, now societal

- Valued as cultural resources
- Focus for large-scale digitization
- Opportunities for museums, libraries, professional associations, …
- Still largely stand-alone entities
- Need for interoperable services

The Maryland Information and Network Dynamics Lab

Things we’re learning…

- Interoperability is …

Necessary

Hard

Complex

Interoperability
Digital Library Components & Services

- Definition
  - Functional and logical interchangeability
  - Well-defined, publicly known interfaces
- Principles
  - Common abstractions
  - Open interfaces to services
  - Extensibility

http://www.dlib.org/dlib/may99/payette/05payette.html

The Maryland Information and Network Dynamics Lab

Interoperability

- **Syntactic (structural relationships within data)**
  - Communication, transport, storage and representation
  - Z39.50, ISO-ILL, XML, …
- **Semantic (interpretation of term usage and meaning)**
  - Different terms to describe similar concepts
    - e.g., 'Author', 'Creator', and 'Composer'
  - Identical terms to describe different concepts
    - term ambiguity

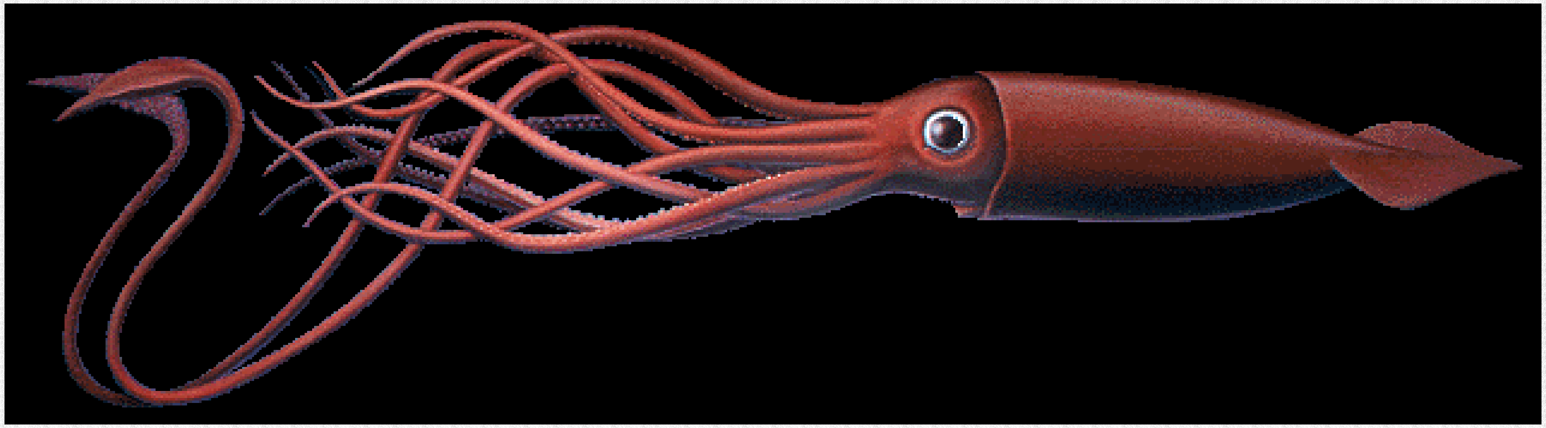
Adapted from http://www.ukoln.ac.uk/interop-focus/about/

The Maryland Information and Network Dynamics Lab

Term ambiguity

- “not all ***squids*** are **s**uperconducting **qu**antum **i**nterference **d**evice**s**.”

  - Jim Hendler, DARPA



Interoperability

- **Syntactic (structural relationships within data)**
  - Communication, transport, storage and representation
  - Z39.50, ISO-ILL, XML, …
- **Semantic (interpretation of term usage and meaning)**
  - Different terms to describe similar concepts
  - Identical terms to describe different concepts
- **Organizational (implications on support structures)**
  - Resource ownership and control
  - Staff (changing skill needs and user communities)
- **Inter-community (supporting new relationships)**
  - Multi-disciplinary research
  - Cross-sector operations (e.g., libraries, museums and archives)
- **International (bridging diversity)**
  - Differences in culture, law, and practice
  - Differences in language
- **Analytic (understanding intent)**
  - Context & task dependencies
  - Temporal & spatial relationships

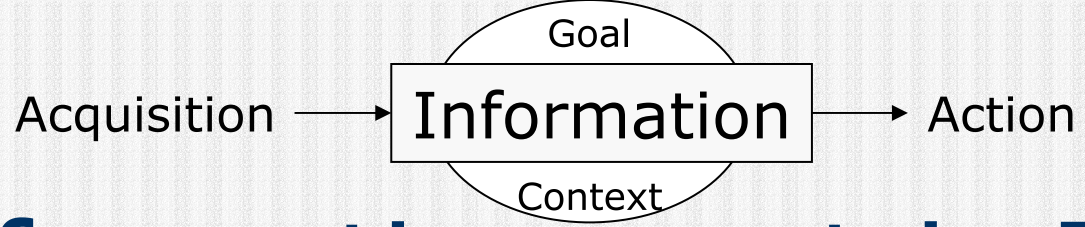
Adapted from http://www.ukoln.ac.uk/interop-focus/about/

The Maryland Information and Network Dynamics Lab

$$\begin{tikzcd}\text{Input} \arrow[r] & \boxed{\text{Process}} \arrow[r] & \text{Output}\end{tikzcd}$$

Process-centric Design

- What are the functions?
- How does the data flow?
- How do processes interact?
  - To communicate data
  - To synchronize operations
- What resources are required?
- How are resources allocated?

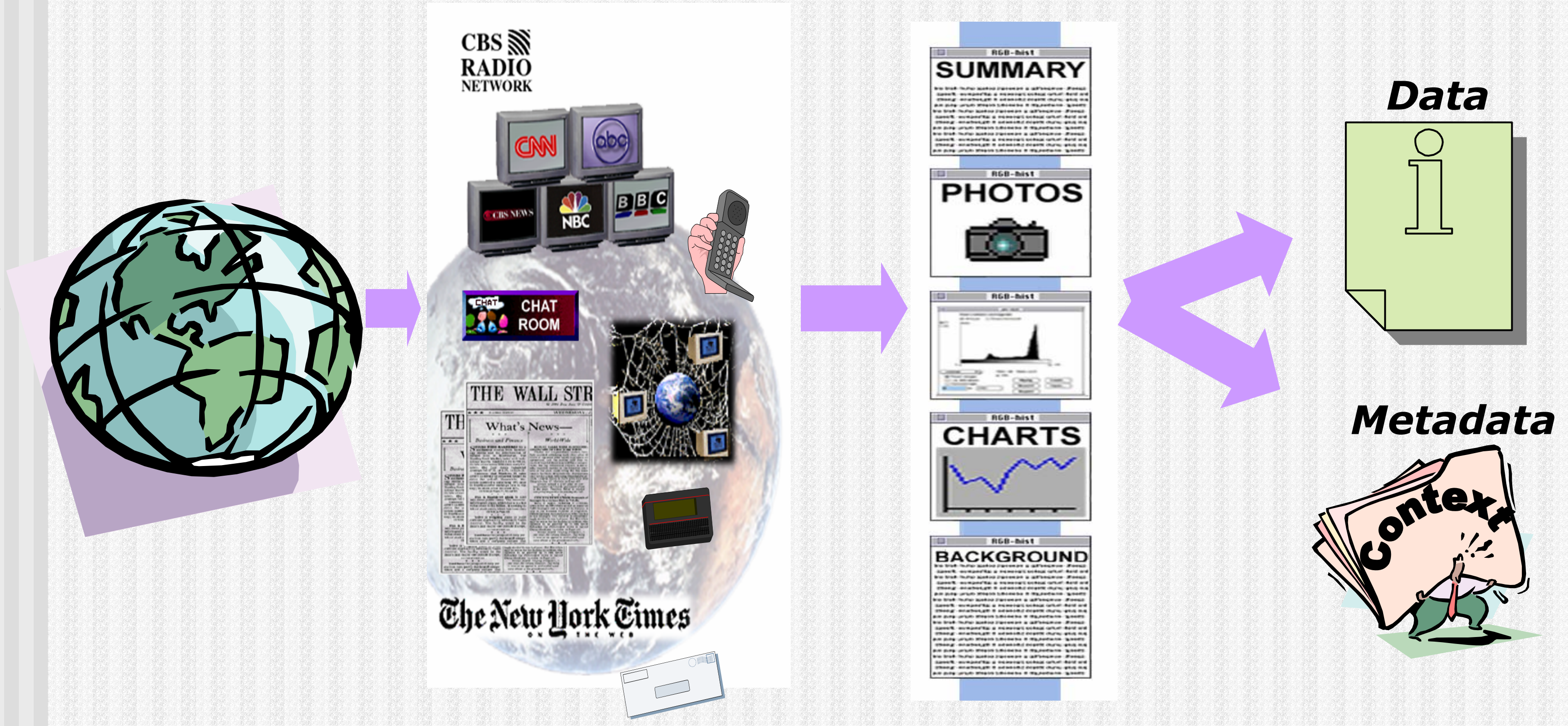


Information-centric Design

- What information is required?
- When is it needed?
- Where does it originate?
- Where will it be used?
- How can it get there?
- What happens to it on the way?

Information vs. Representation

Observing Capturing Encoding Representing



One Event Multiple Observers Edited Copy Recorded History

Information Dynamics

- **Information**
  - Explicit := perceived
  - Implicit := inferred
  - Represented := encoded
- **Dynamics**
  - Location := availability
  - Timeliness := relevance
  - Value := confidence
- ***Objective***
  - *Goal := represented in utility function*
  - *Context := domain of utility function*
  - *Action := maximize utility over domain*

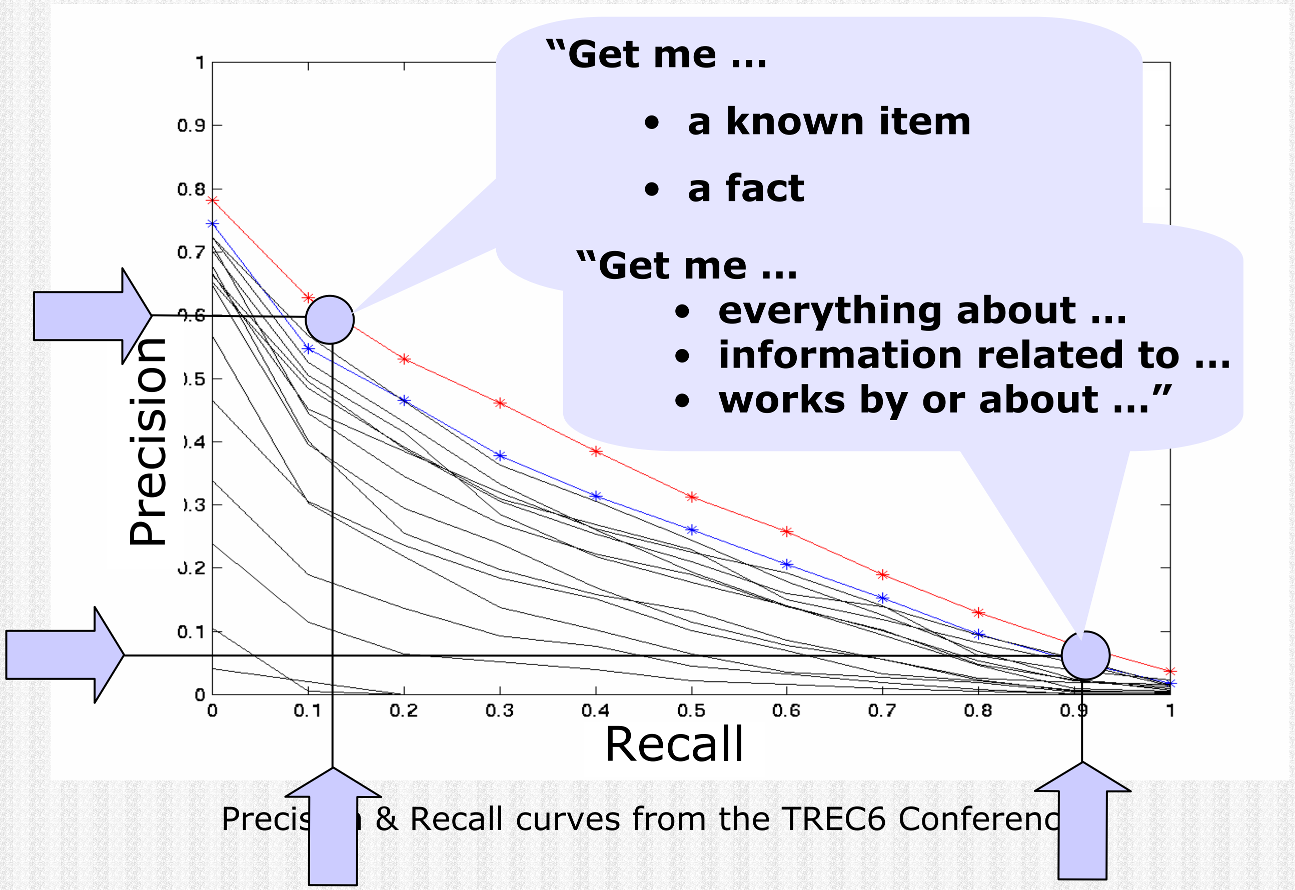
What is needed?

Where is it?

Will it help?

The Maryland Information and Network Dynamics Lab

Controlling Retrieval



"Get me …

- a known item
- a fact

"Get me …

- everything about …
- information related to …
- works by or about …"

Precis n & Recall curves from the TREC6 Conferenc

http://trec.nist.gov/

The Maryland Information and Network Dynamics Lab

Steps toward analytic interoperability

- Instrumented test-beds
- Better metrics
- Interoperable information substrate
  - Semantic web
    - XML
    - RDF
    - Ontologies
  - Time & location-aware agents
  - Context-aware clients

The Maryland Information and Network Dynamics Lab

Information Dynamics in Digital Libraries

- Framework for analytic interoperability
- Consider information value *in context*
- Recognize temporal dependencies
- Adapt user services to user needs

Understand (and exploit) the role time and location play in the value of information.

The Maryland Information and Network Dynamics Lab

APPENDIX F

Information Dynamics - An Information-centric Approach to Digital Library Interoperability[*]

Ronald L. Larsen
University of Maryland
College Park, Maryland 20742, USA
rlarsen@deans.umd.edu

KEYWORDS: digital library, interoperability**,** system design, integration, analysis

ABSTRACT

Acquisition, organization, management, retrieval, and distribution of information are fundamental purposes of digital libraries and their supporting infrastructures. Interoperable digital libraries pose particularly difficult system design issues. Interoperability research has focused largely on syntactic and semantic interoperability. In this paper, a third form of interoperability, *analytic interoperability* is proposed, with a framework in which to consider it. Since *information* is the essential commodity of interest, a comprehensive interoperability design should take into account the fundamental properties of information, including representation, composition, relationships, and dynamics. *Information Dynamics* considers how the nature of information can be used to achieve analytic interoperability.

INTRODUCTION

The growth of networked information resources, largely through the Internet and the World Wide Web, is both a result and a source of the growing interest in digital libraries. Digital libraries have emerged as the vehicle for organizing collections of digital information, much as traditional libraries have done for print and related media. They are becoming a major component of the global information infrastructure. But little standardization exists among digital libraries, and it can be argued that the international standards process is poorly suited to the rapid pace of technology development that has become familiar on the Web. Alternatively, developers of digital libraries focus more on *interoperability* among heterogeneous, or federated, systems. Andreas Paepcke [i] describes these as "cooperating systems where individual components are designed or operated autonomously." He suggests that "the ultimate goal for such a system is to have components evolve independently, yet to allow all components to call on each other efficiently and conveniently."

The rapid advancement of digital libraries throughout the 90's and the near-daily announcement of new technologies and systems all but assure that these trends will accelerate in the current decade. The hardware technology exists to create ever more complex networks of systems. However, our ability to design, implement, operate, maintain and support these increasingly complex systems lags. One reason for this is the paradigm used in system design remains largely *process-centric*, founded on economic principles inherited from many generations past of Moore's Law. Digital libraries provide the need and the opportunity to design around *information-centric* principles.

[*] This work is supported partly by DARPA / Rome Labs, Department of the Air Force, under contract F306020020578 to the Department of Computer Science at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Department of the Air Force, DARPA, DOD, or the U.S. Government.

Information Dynamics[ii] is an alternate design paradigm that takes such an information-centric perspective. In this approach the role information plays is central; system design considers what information is needed and when, where the information is, and what happens to the information as it moves from one place to another. In an Information Dynamics framework, information is treated as a dynamic entity and its dynamics (e.g., location, timeliness, value) are explicitly considered. Information processing is performed through *actions* carried out as a result of explicit *choices.* Any action that carries out a transformation or related processing of information consumes resources, requiring these resources over some period of time. One can think of this in terms of actions that occupy a subspace in resource/time space. All actions take time and, therefore, have an impact on the dynamics of information. Further, information is considered to have value within a *context* and with respect to the achievement of a *goal*. To this end, the value of information typically changes over time within a given context.

The use of information for effecting time-dependent control and related decision processes is not new. Physical systems respond adaptively to linear-quadratic-Gaussian (LQG) controllers, for example, when the physics is well understood and controllers have rigidly bounded responsibilities. But decision making in network-based, distributed systems for which there is no nice physics poses a different class of problem. Early investigations into distributed decision-making led to a wealth of research in game theory and later team theory, [iii,iv,v,vi] but none of this work explicitly considered the temporal effects on the value of information, and how that affected system performance. Recent literature on autonomous agents[vii,viii] reports work in distributed, multi-agent decision-making (using, for example, state space approaches such as Markov Decision Processes) but still avoids consideration of the temporal issues underlying the usage and value of information. Interoperability research has distinguished between *syntactic* and *semantic* interoperability. While the boundary between them is blurry, syntactic interoperability typically refers to agreement on structural relationships within communication, while semantic interoperability addresses common interpretations of term usage and meaning. But this speaks to *how* information is shared among heterogeneous components. While information is used throughout these approaches, and differently in most of them, techniques that explicitly consider the role of the time/value of information (the *when/what*) and the reason for communicating (the *why*) are lacking. Analytic interoperability addresses these topics by considering intentions, or understanding what the purpose of the interaction is and using this to optimize actions.

In this paper, the dynamic nature of information is considered with regard to decision processes and the resulting implications on the design of interoperable federated digital libraries. For illustrative purposes, information dynamics is first applied to a networking problem.

WHAT IS INFORMATION?

Information is a property, characteristic, or description of something physical, logical, virtual, or conceptual, including other information. It may be a group, an action, a choice, or a relationship between any of these things. Information has value *within a context.* Relationships between information may be direct or indirect, and exist whether they are enumerated or not. Relationships may be static or dynamic. The causality principle applies. Information in the present can only affect the future; it cannot change the past. It may change the interpretation of the past, but it cannot change the past, itself. Further, delays involved in the movement of information assure that knowledge of a remote entity's state is necessarily delayed; the "present" system state as understood by any system entity will, therefore, actually reflect the collective state of system components at past time instant(s).

Every digital library consists of a large number of functional components and a much larger number of digital objects (the contents). Considering all possible pieces of relevant information and their relationships yields an arbitrarily large amount of information. In a typical system design only a small amount of relevant information is collected and used.

When multiple pieces of information are related, their relationship may be considered a higher level of information. Action is required to establish the validity of a relationship. When the *relationship* is explicit, the related information can be derived *implicitly*. In this regard, information that can be derived from other information based on known relationships is implicit information.

Information Has Value

Information has value (or *utility*) within a given context. Context can be considered the domain of the utility function relevant to the task at hand. The value of information may change with time within a given context, and its value may differ in different contexts at the same time instant. Information value may also depend on relationships with other information. Information is represented in *information variables* that include the item of information and associated metadata. A context vector defines the domain within which a relevance computation (e.g., cosine or dot product) can be made in order to establish the importance of the variable to the task at hand. Metadata associated with a variable may also include a measure of confidence (e.g., a probability distribution function) by which the significance of the variable can be assessed. The value, or utility, of the variable is a function of both context and confidence, both of which may also be a function of time. Note that while both context and confidence are formally required to understand the role of an information variable in a system, they are rarely considered explicitly in the design of systems.

Explicit information may be acquired through direct observation, communication, or inference. Only explicit information can be communicated. Implicit information inferred from known relationships by different observers or by communicating entities may still differ, unless they also share a common context and a common understanding of relationships.

Communication requires agreement on representation and protocol (the *how*), but that is only the beginning. More fundamental is the determination of *what* is to be communicated, *when*, and *why;* for every action, including communication, consumes resources and time. Knowledge of what information is required, and where and when it is needed, is a crucial part of federated system design. But for a federated digital library system, it is also the very reason for its existence.

Federated Systems

Consider the implications of information dynamics on a federated system. Such a system has a collection of entities (processing resources) capable of carrying out certain operations. A specific distributed system, designed to carry out a specific mission, uses physical resources to carry out actions and to store and move information. When such a system is interacting with an external physical system it may also have sensors and actuators.

A federated system is a collection of autonomous nodes with an interconnecting infrastructure for communication. Each node maintains a *perceived reality,* based on its prior model of the operating environment and explicit information it receives. Explicit information can be integrated into the model to update the perceived reality. Perceived reality at no node can ever be assured to be identical to *actual reality*, particularly with respect to non-local events. Transmission delays assure that information received from any remote node is, by definition, historic. Further, it is not sufficient to receive messages; they must be interpreted and integrated into the local perceived reality.

Information Dynamics incorporates an information-centric system view that explicitly considers temporal aspects of information, the value of information, and the role of implicit information. Within this framework, a system can capitalize on dynamic system behaviors that would otherwise be liabilities. Decisions are explicitly based on perceived reality, taking into account the environmental model, its dynamics, information sources, and their interactions. The value of information, conditioned on confidence levels and context vectors, plays a key role in system operation.

EXAMPLE: ROUTING

As a concrete example of the application of information dynamics to a practical problem, consider link state routing in a computer network. Shortest path routes from a source to a destination are determined according to the current known state of links. Routing algorithms typically measure their state (e.g., queue length) periodically, estimate the waiting time, and broadcast this information to neighboring nodes, which use it for best-route determination.

Consider the basic characteristics of link performance. In a typical network each link continuously transfers packets, as presented, up to the capacity of the link. Considering a link as a server, its steady state behavior can be characterized in terms of the mean, w, and the variance, v, of the waiting time. Let w(t) be the waiting time at a particular link as measured locally at time t. Assuming that the measurement is done correctly, the variance of this measurement, v(t), is zero. Given no additional information about the state of the link, the estimated waiting time $w(t_1)$ at some later time t_1 will necessarily be based on w(t) and our knowledge of w. This estimate will have a variance, $v(t_1)$, which will be nonzero. In fact, the variance will be an increasing function of the difference t_1-t, tending towards the steady state value v. Given w(t) and v(t), the actual values of $w(t_1)$ and $v(t_1)$ can be estimated with knowledge of the stochastic behavior of the link.

In this example, the basic information variable is the waiting time estimate for the link and the variance estimate is its confidence indicator. Recognizing that communicating w(t) to another node in the network takes time, any new estimates should take into account the dynamics of the situation. Depending on the characteristics of the link, the estimates $w(t_1)$ and $v(t_1)$ may come so close to the steady state values w and v that the new measurements will have negligible impact on the link information retained by another node. As a consequence, communication can be significantly reduced for link-state routing without decreasing the quality of routing decisions by considering the variance in delay estimates. Each node does need steady-state information about the links. Note that if the steady state conditions change regularly, that knowledge can also be shared. By explicitly considering the value of information in this simple algorithm, information dynamics improves the design of the routing scheme.[ix] Early results suggest a savings of at least 25% in routing control information is achievable using information dynamics approaches to link state routing.

It is important to understand that although the above example uses statistical measures, the information dynamics framework is not limited to handling quantitative information. It can be equally effective in using fuzzy or purely qualitative information.

DIGITAL LIBRARY IMPLICATIONS

A digital library has been defined as "the collection of services and information objects that support users in dealing with information objects and their organization and presentation, available directly or indirectly via electronic means."[x] This is a sufficient working definition; digital libraries allow individuals and organizations to efficiently and effectively identify, assemble, correlate, manipulate, and disseminate information resources, regardless of the medium in which the information exists. Digital libraries provide tools to navigate and manipulate information in a multimedia, multilingual, multidisciplinary world. Task context, user values, and information provenance are critical elements in the information seeking process, but have yet to become part of the digital library infrastructure.

But how might information dynamics concepts be introduced into digital library design? Consider the information retrieval functions of digital libraries. A user formulates a query from a client entity, which sends the query to a search engine operating over some set of repositories. The repositories use the query terms to suggest materials in the local collection that may be responsive to the query, and a ranked list of responses is constructed. Either in middleware or in the client, itself, the responses from the multiple repositories are merged into a ranked list that is presented to the user, who is then responsible for perusing the list in the hope of finding relevant materials. Much effort has gone into developing high performance search engines, typically measured by *precision* and *recall* over a test corpus. But while precision and recall are used to evaluate performance in carefully constructed test scenarios, the results of these evaluations have not typically been used to *control* search engine performance. Consider, for example, the results of the TREC6 Conference shown in Figure 1, which is a typical display of state-of-the-art performance for information retrieval engines. While curves with higher precision and recall are generally superior, when these systems are placed into operation, the user has no control over where on these curves the system will perform for a particular search. Control parameters are set within the system implementation and are totally opaque to the user.

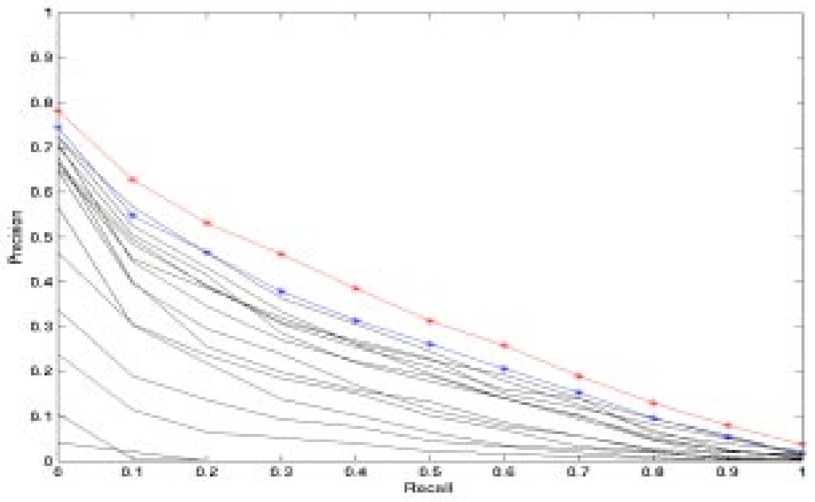


Figure 1. Precision & Recall curves from the TREC6 Conference [xi]

Information dynamics brings to the fore this kind of trade-off. It explicitly recognizes that each operation is unique, but that there is information available to tailor system performance to the specific character of the operation. For example, consider providing in the user's query a parameter that alludes to the purpose (or goal) of the query. It may be that the user is only casually familiar with the field and needs introductory material. The search would be more effective, perhaps, by applying substantially more weight to precision than to recall. As the nature of the query moves from casual interest through tutorial (instructional), fact-finding (known-item search), research (focused inquiry), to survey (comprehensive exploration), the search engine could deliver more relevant responses by progressively shifting its weighting more toward recall and away from precision.

In like manner, information dynamics suggests strategies to incorporate *context* into queries. Query term ambiguity results in false returns from search engines. If the user incorporates multiple terms into a query, or is encouraged to do so through iterative refinement of a query by responding to irrelevant returns from poorly framed queries, the detrimental effects of term ambiguity can be reduced. But information dynamics suggests an alternate approach, capitalizing on the term disambiguation refinement work of ontologies and *entry vocabularies*.[xii] If the search engine is given sufficient information to limit the domain of search, there is reason to believe that the precision of returned results would improve.

Information retrieval operations in digital libraries are largely state-less transactions, particularly across multiple patrons. In stark contrast to the way reference librarians learn and improve their performance, later users of search engines rarely benefit from earlier searches conducted by those engines. But they could. Just as caching improves communications performance by capitalizing on temporal aspects of information demand, information retrieval can benefit from prior searches of like nature with a similar context. This represents the very early stages of analytic interoperability.

Digital library research has touched on analytic interoperability, without using the term. Several research projects have examined implicit and explicit collaborative techniques for improving an individual's success in information retrieval, for example, by capitalizing on prior search activities conducted by other individuals.[xiii] [xiv] Query languages and tools identify digital library materials across federated collections that are similar to the characteristics expressed in the query. These characteristics focus on the information artifact and are only beginning to consider non-bibliographic attributes to improve the search. Examples include identifying the types of individuals who have been reading specific material, the value they associated with it, and the paths they traversed to find it. These approaches require instrumented digital libraries, in order to build the perceived reality and set the context that will enable improved performance at the user-level. This goes beyond issues of functionally compatibility among federated systems, to a mission-oriented control structure designed to improve both the qualitative and quantitative performance *as perceived directly by the end-user*.

Digital libraries face significant technology challenges. These include real time ingest (capturing, interpreting, cataloging, and indexing high rate multimedia data flows in real time), federating distributed repositories (organizing heterogeneous distributed information sources into comprehensive discipline-oriented, user-accessible repositories), and cross-lingual interaction (automatically accessing and using information across multiple natural languages).[xv] Information dynamics holds the potential to raise the level of interoperability among users and digital libraries from a high dependence on syntax, structure, and word choice to greater exploitation of semantics, context, and concepts, thereby extending information search and filtering beyond purely bibliographic criteria to include contextual criteria related to the task, to the user, and to the time constraints of the user.

Scalability and interoperability are well-known, fundamental requirements for digital libraries. Scalable repository technology must support the federation of thousands of repositories, present to the user a coherent collection of millions of related items, and do this rigorously across many disciplines. Information dynamics holds the potential of addressing these issues with more than brute force bandwidth and capacity. As the size and complexity of information objects increases, so does the bandwidth required to use these objects. Time-critical applications requiring real-time interactivity push the bandwidth requirements even higher. *Broadband interoperability* refers to the dramatic changes in the user's work style that become feasible when the user's inputs are no longer constrained to a few keystrokes or mouse clicks. Information-centric design founded on context, utility (or information value), and temporal relationships offer the potential for real-time adaptation of scalable network and repository services

CONCLUSIONS

Whereas *information* management is the mission of complex systems, *process* management remains the dominant design and implementation approach. Since information is the essential commodity, effective design strategies should explicitly address the fundamental properties of information. The first principle is to recognize the distinction between information and its representation. Computer systems are only capable of manipulating representations and it is through the processing of representations that we attempt to carry out the processing of information. These representations are limited in that they capture only a limited portion of the generic information. Moreover, processing changes the nature of information in ways that are not necessarily intended or anticipated. *Implicit information* must also be understood and elucidated.

The second fundamental principle of Information Dynamics is that information has value in *context*. Processing affects the value of information. Movement, representation, and storage also affect information value. But the ramifications on system design are rarely considered. The third fundamental principle of Information Dynamics is that the value of information changes with time. Understanding the role time plays in the value of information impacts the applicability of information. Communication of information takes time. When the delay caused by communication becomes large, the value of the information may be reduced sufficiently that its communication may not only have been unnecessary, but may, in fact, be detrimental.

The principles of Information Dynamics presented here represent ongoing work to understand the fundamental characteristics of information within a federated system context. The objective is to develop information-centric models of system design and operation. The framework has shown the potential for bringing about a significant advancement in the way information is handled in systems.

[i] Paepcke, A., S. Chang, et al., "Interoperability for Digital Libraries Worldwide," Communications of the ACM, Volume 41, 4, April 1998.

[ii] Larsen, R., A. K. Agrawala, and D. Szajda, "Information Dynamics: An Information-Centric Approach to System Design," International Conference on Virtual Worlds and Simulation, San Diego, CA, January 2000.

[iii] Bacsar, T. and Olsder, G. J., *Dynamic Noncooperative Game Theory*, Academic Press, New York, Mathematics in Science and Engineering, v.160, 1982.

[iv] Greenwald, A., "Modern Game Theory: Deduction vs. Induction," TR 1998-756, New York University, 2/24/98.

[v] Greenwald, A., "Learning to Play Network Games," TR1998-758, New York University, 2/24/98.

[vi] Owen, G., *Game Theory*, Saunders, Philadelphia, PA, 1968.

[vii] Stone, P. and Veloso, M., "Using Decision Tree Confidence Factors for Multi-Agent Control," *Proceedings of the 2nd International Conference on Autonomous Agents*, ACM Press, NY, 1998, pp. 86-91.

[viii] Washington, R., "Markov Tracking for Agent Coordination," *Proceedings of the 2nd International Conference on Autonomous Agents*, ACM Press, NY, 1998, pp. 70-77.

[ix] Ahn, Sungjoon and A. Udaya Shankar, "An Application of the Information Dynamics Framework: Adapting to Route-demand and Mobility (ARM) in Ad hoc Network Routing," Computer Science Department, University of Maryland, March 2001.

[x] Leiner, Barry M., "The Scope of the Digital Library," Draft prepared for the DLib Working Group on Digital Library Metrics, January 16, 1998, Revised October 15, 1998, http://www.dlib.org/metrics/public/papers/dig-lib-scope.html

[xi] See the proceedings of the Sixth Text Retrieval Conference (TREC) at http://trec.nist.gov/

[xii] Buckland, M., Chen, A., Chen, H., Gey, F., Kim, Y., Lam, B., Larson, R., Norgard, B., and Purat, Y. "Mapping Entry Vocabulary to Unfamiliar Metadata Vocabularies," D-Lib Magazine, Vol.5 No.1, January 1999.

[xiii] Kantor, Paul B., Endre Boros, Benjamin Melamed, Vladimir Meñkov, "The Information Quest: A Dynamic Model of User's Information Needs," Rutgers University, http://aplab.rutgers.edu/ant/

[xiv] See http://www.packhunter.com

[xv] Larsen, Ronald L., and Jean Scholtz, "Digital Libraries and Scholarly Communication," to be published.

APPENDIX G

Adapting to Route-demand and Mobility (ARM) in Ad hoc Network Routing*

Sungjoon Ahn and A. Udaya Shankar
({sjahn, shankar}@cs.umd.edu)
Computer Science Department
University of Maryland
College Park, MD 20742

CS-TR-#4214

October 26, 2001

Abstract

We present ARM (Adapting to Route-demand and Mobility), a control mechanism that allows any proactive routing protocol to dynamically adapt in a totally distributed manner to changes in node mobility and workload route-demands. Each node independently maintains a *mobility metric* indicating how fast its neighborhood is currently changing, and a *route-demand metric* indicating which destinations are currently involved in data forwarding. Control functions use these metrics to dynamically adjust the period and the content of routing updates.

We apply ARM to the DSDV protocol, coming up with ARM-DSDV. Simulations for various mobility and workload scenarios show that ARM-DSDV typically achieves better data delivery, while keeping the routing cost at reasonable levels, when compared to DSDV with update period optimized for the scenario. In designing and implementing ARM, the Information Dynamics framework provides a useful reference.

*This work is supported partly by DARPA/ Rome Labs, Department of the Air Force, under contract F306020020578 to the Department of Computer Science at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Air Force, DARPA, DOD, or the U.S. Government.

ARM is a result of applying the Information Dynamics framework to ad-hoc network routing. Information Dynamics emphasizes the role of information and its temporal dynamics in system design. In the context of ad-hoc nework routing, the routing state maintained by a node is its perceived reality, and the mobility and route-demand metrics are indicators of its current value.

1 Introduction

Ad hoc networks, also called MANETs (Mobile Ad hoc NETworks), are wireless data networks that do not require any communication infrastructure, unlike cellular networks and access point based wireless LANs. Ad hoc networks are suited for combat situations, search and rescue operations, and instant conferencing in infrastructure-absent geographic areas.

Ad hoc networks have characteristics that routing protocols for conventional networks need not deal with, among them dynamic topology, low bandwidth, short host battery life, and unreliable links. New routing protocols are needed for ad hoc networks and a number have been proposed, for example [1, 2, 3, 4, 5]. These routing protocols are usually classified as *proactive* or *reactive*. Proactive routing protocols [1, 2] periodically exchange routing updates to continuously maintain routes between all mobile host pairs, as in conventional wire routing protocols. Reactive protocols [3, 4, 5] send routing updates only when the data traffic demands routes and these updates are only for those routes.

However, in general no single routing protocol performs well over a wide range of mobility and route-demand patterns. In fact, few of the proposed routing algorithms adapt their behavior to changes in mobility and route-demand patterns. For example, most proactive protocols use a fixed update frequency even if the mobility patterns are changing.

This paper presents ARM (Adapting to Route-demand and Mobility), a control mechanism that allows any proactive routing protocol to adapt in a totally distributed manner to changes in node mobility and changes in data traffic demand for routes. Each node independently maintains two metrics. One is *mobility metric* indicating how fast its neighborhood is currently changing, thereby reflecting the current rate of mobility. The other is *route-demand metric* indicating which destinations are currently involved in data forwarding, thereby reflecting the current demand for routes. These metrics are used by two control functions, called *update-period control function* and *update-content control function*, to dynamically adjust the period and the content of routing updates.

The ARM approach can be readily applied to any proactive routing protocol. In this paper, we apply it to DSDV (Destination Sequence Distance Vector) protocol [1], coming up with ARM-DSDV, and evaluate its performance relative to DSDV for several simple control rules using simulations. The simulator has a physical layer modeled by transmission range and bandwidth, a link layer based upon IEEE 802.11 (including CSMA/CA and RTS/CTS), and a workload layer of end-to-end connections. The performance metrics are data delivery ratio (fraction of data packets delivered to destinations) and routing cost (number of routing update octets transmitted). For various mobility and workload scenarios, ARM-DSDV typically achieves better data delivery ratio than scenario-optimized DSDV while keeping the routing cost at reasonable levels. Naturally, ARM-DSDV achieves higher data delivery ratio than non-optimized DSDV.

Information dynamics of ad-hoc network routing

The Information Dynamics framework [14] plays an important role in designing the ARM control mechanism. The framework emphasizes the role of information and its temporal dynamics in agent-based system design. When applied to ad-hoc network routing, the agents correspond to the routing entities of the mobile ndoes. The network state maintained by a routing entity corresponds to its perceived reality. In traditional ad-hoc routing protocols, the state consists of the distances (or link costs) to destinations along various paths.

Information Dynamics tells us that in a mobile network, the value of a path's distance depends on the mobility of the nodes on the path and the demand for that path by data packets. Hence in an ARM agent, the network state consists of the usual information augmented by the mobility and route-demand metrics. These metrics reflect the value of information associated with a path, where value essentially decreases with time at a rate that depends on how rapidly the path is changing or how much demand there is for the path. That is, the value of the routing information indicates how beneficial the routing information is to the data traffic. Unpopular routing entry information has low value no matter how recent or accurate it is.

Mobility and route-demand motivate the two controls of ARM. The update-period control increases the value of the routing information according to the mobility. It uses the mobility metric, an additional piece of information, to estimate the mobility. The update-content control increases the value of the routing information according to the route-demand. It uses the route-demand metric, another piece of information, to estimate the route-demand. Like the routing information, the values of both metrics decrease over time.

Organization of the paper

This paper is organized as follows. Section 2 addresses related work. Section 3 explains the ARM mechanism. Section 4 details the ARM-DSDV protocol. Sections 5, 6, and 7 describe respectively the performance evaluation model, the performance metrics, and the simulation results. Section 8 concludes.

2 Related Work

Proactive routing protocols [1, 2] periodically exchange routing updates to continuously maintain routes between all mobile host pairs, as in conventional wire routing protocols. Their advantage is that a data packet can be sent out immediately without any routing delay if a route exists. On the other hand, a proactive protocol wastes a large portion of available bandwidth when most of the routes it maintains are not used. Some optimizations are suggested to mitigate the routing overhead in [1]. One is *incremental dump* where each node advertises only the difference from the last update, reducing update message traffic. *Delayed update* defers broadcasting route update messages in hopes of better routes arriving shortly. Despite these optimizations, the routing update traffic of proactive routing is rather large. This is the main reason why proactive protocols perform inefficiently in certain network conditions [6], especially those characterized by skewed workload.

Reactive protocols [3, 4, 5] aim to eliminate the excessive overhead of proactive protocols by sending routing packets only when the data traffic demands routes. One disadvantage is the unavoidable initial delay in forwarding the first packet of a connection. Also, even though the goal is to reduce routing traffic, reactive protocols can suffer from high routing traffic overhead because they flood the network when discovering a new route [7]. The effect of flooding can be disastrous when the network is large and demand for new routes is high. Modifications can be introduced to alleviate the cost of flooding. For instance, geographic location of each node is exploited in LAR (Location Aided Routing) [8] to limit flooding area. However this assumes that all nodes are equipped with special devices like GPS. Another approach [9] makes use of prior routing histories to localize route request queries.

Hybrid protocols try to combine the strengths of proactive and reactive protocols. ZRP (Zone Routing Protocol) [7] is a hierarchical routing protocol that combines proactive and reactive protocols. A network is divided into zones inside which routing is done proactively. Interzone routing is performed reactively and only among zone leaders, which are elected nodes responsible for providing interzone routes to their zone members. By allowing proactive routing only within a zone, bandwidth is not wasted in advertising route entries of other zones. Flooding is limited by their route query control schemes. The zone radius is a configurable parameter. With zone radius of one, ZRP becomes purely reactive. With infinite zone radius, it becomes purely proactive. Through simulation the authors show that the optimal zone radius value depends on the call-to-mobility ratio (the ratio of number of calls or connections to node speeds). A high ratio (i.e., large number of connections or low mobility) favors large radius or more proactivity, while a low ratio favors small radius or more reactivity. ZRP assumes that the network condition (call-to-mobility ratio) is relatively static and known a priori for setting the optimal zone radius. Dynamically changing the zone radius involves reconfiguring zones and electing zone leaders, which can be pretty expensive and may require data forwarding to be halted during the transition.

In a survey paper of ad hoc routing protocols [10], the authors suggest switching between different protocols according to current network conditions. But no detailed work has been reported about intelligently switching between routing protocols based on dynamic network conditions. The cost of switching would be high in general because it has to occur globally throughout the network.

To summarize, few routing protocols adjust their operational parameters dynamically during their execution. ARM, proposed in this paper, allows any proactive routing protocol to adapt dynamically to changes in mobility and route-demand patterns. Furthermore it is completely decentralized. Adaptations do not require network-wide synchronization. Each node adapts independently based on local observation and decision, spending a small amount of additional overhead.

3 ARM Mechanism

Recall that ARM has two controls. The update-period control maintains the mobility metric and dynamically adjusts the routing update period. The update-content control maintains the route-demand metric and dynamically adjusts the content of routing updates.

ARM can be applied to any routing protocol in which each node *peri-*

odically sends routing updates. Specifically, in each period a node sends an routing update message constructed from its current routing information and builds new routing information based upon routing update messages received. At the end of the period, the new routing information becomes the current routing information, and the cycle repeats. Data packets are forwarded based on the current routing information.

Most proactive routing protocols behave this way. Also, these protocols broadcast routing update messages and hence can lose them. ARM is robust to such loss.

3.1 Update-period control

The intuition behind update-period control is that a node should update more frequently in high mobility, so as to accurately reflect the current network topology, and less frequently in low mobility, when frequent updates do not bring additional accuracy but consume bandwidth.

A node measures mobility by measuring the rate of change in its *neighborhood*, i.e., the set of nodes within radio range. The node maintains two neighbor tables: *current_neighbor_table*, based on updates received by the start of the current update period, and *new_neighbor_table*, based on updates received during the current update period. Both tables have entries of the type {*neighbor_id*, *t_expiration*, *mobility_metric*}. Member *t_expiration* represents the time when the next update from the neighbor is expected to arrive. An entry is regarded as expired if *t_expiration* is less than current clock. This expiry time information is required because different nodes can have different update periods and update information from a node with a longer period should survive longer than that from a node with a shorter period. Member *mobility_metric* indicates the mobility metric of the neighbor, and is used in computing the update period as described below. To maintain these neighbor tables, routing update messages need to contain the sender's update period and the sender's mobility metric in addition to the sender's id.

Figure 1 describes the processing involved in the update-period control. At the start of a new period, an "instantaneous" mobility metric is obtained by dividing the number of nodes that became neighbors or stopped being neighbors by the previous update period. Smoothing the instantaneous metric over time interval *TW_SMOOTH* yields the mobility metric. An "aggregate" mobility metric is obtained by averaging the mobility metric of this node and the mobility metrics of the neighbors (available in new neighbor

```
At start of an update period:
 // Mobility metric computation
 Remove expired entries from new_neighbor_table;
 Let nbd_change be the number of nodes e such that
  e is expired in current_neighbor_table and not in new_neighbor_table
  or e is in new_neighbor_table and not in current_neighbor_table;
 Let inst_nbd_rate_change be nbd_change divided
  by current update period;
 Let mobility_metric be the inst_nbd_rate_change smoothed
  over TW_SMOOTH;

 // New update period computation
 Let aggregate_mobility_metric be the average of mobility
  metrics of this node and neighbors (from neighbor table)
 New update period is update-period control function value
  at aggregate_mobility_metric;

 // Neighbor tables update
 Remove expired entries from current_neighbor_table;
 For each entry n in new_neighbor_table
  if match m is found in current_neighbor_table then
   assign n.t_expiration to m.t_expiration;
   assign n.mobility_metric to m.mobility_metric;
  otherwise insert n into current_neighbor_table;
 Empty new_neighbor_table;

At reception of routing update message:
 Let expiration_time be sender's update period
  plus clock plus slack;
 If sender's id has match in new_neighbor_table then
  update t_expiration and mobility_metric in the table;
 otherwise insert the sender's id, expiration time,
  and mobility metric into the table;

At reception/overhearing of data packet:
 Let expiration_time be minimum update period
  plus clock plus slack;
 Let mobility_metric be a negative value;
 If sender's id has match in new_neighbor_table then
  update t_expiration and mobility_metric in the table;
 otherwise insert the sender's id, expiration time,
  and mobility metric into the table;
```

Figure 1: ARM update-period control

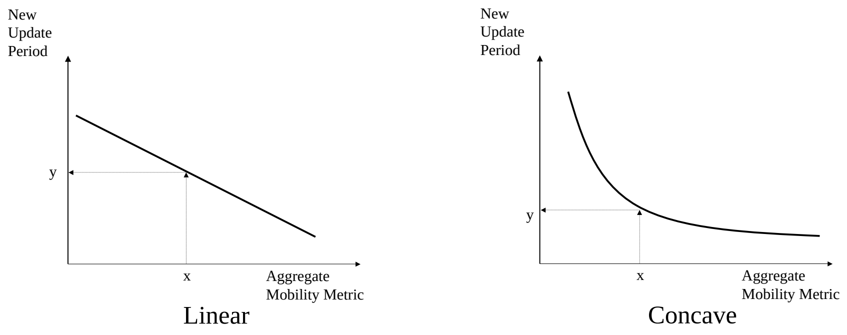


Figure 2: Simple update-period control functions

table). The update-period control function maps the neighborhood mobility metric to the new update period. The new neighbor table is merged into the current neighbor table and expired entries are deleted.

Data packets received or overheard by the node are also used to build the neighbor tables. Because data packets do not carry the sender's update period and sender's mobility metric, which are required for neighbor tables, the following is done: the sender's update period is set to the minimum update period of the update-period control function, and the sender's mobility metric is set to a negative value (to indicate that it should not be used in computing the aggregate mobility metric).

There are several points to note. In computing the mobility metric, an alternative to smoothing is to use weighted time averaging, as in TCP's RTT estimation [11]. The expiry times of entries in the neighbor tables is reasonably well-approximated by the sum of the current time and the sender's update period; some slack can be added for a more conservative estimate. When ARM is instantiated in a routing protocol, the routing table entries are subject to the same expiry time constraints as neighbor table entries. Regarding update-period control function, it turns out that even simple update-period control functions, such as shown in Figure 2, outperform non-adaptive update periods.

In obtaining the update-period control function, we rely on simple and intuitive procedures instead of a thorough analysis. According to simulations we have done with different control functions, the range of the new update periods is more important than the shape of the function. The range of aggregate mobility metric is obtained empirically and the maximum value between 20 and 50 is reasonable.

The maximum value for the new update period is roughly determined by considering the maximum end-to-end delay within which routing information is desired to be propagated from one end of the network to another. For example, 10 hops of network diameter and 5 seconds of desired maximum end-to-end delay yields 0.5 seconds of maximum new update period. The minimum value for the new update period is roughly determined by considering the allowed bandwidth share of routing update messages within a neighborhood together with average number of neighbors, average update message size, bandwidth, and link access delay. For example, 10 neighbors, 150 bytes of update message, 2 Mbps of bandwidth, 0 seconds of link access delay, and 5% routing bandwidth share yields 0.13 seconds of the minimum update period.

The update-period control computation is completely local to the node. The overhead consists of the extra fields in routing update messages and the computation/storage of mobility metric, new update period, and neighbor tables.

3.2 Update-content control

The intuition behind update-content control is that a node does not have to send a piece of routing information in every routing update if that information is not being used by other nodes. In ARM, each node keeps track of which destinations it has forwarded packets to recently, specifically within a time window *TW_RECENT*. When constructing a routing update message, routing information for a destination is included only if it satisfies a *update-content control function.*

To implement this, the node maintains an additional field, *t_forwarded_last*, for each entry in the current routing table to indicate when the last forwarding of a packet occurred to that destination. The recent route demands recorded in this additional field represent the *route-demand metric.* Figure 3 describes the update-content control operation.

A *filter function* slows down the rate at which routing information of an unpopular destination is advertised. It can be implemented in several ways, for example, including the routing information only in every Kth routing message, or only randomly with some probability, etc. We point out that even if the update-content control decides not to include any routing information, the node still transmits a routing update (depending on update-period control) in order to advertise its own presence and help the update-

```
At forwarding of a data packet:
 // Route-demand metric computation
 If the data packet's destination is in
  the current_routing_table then
   update the entry's t_last_forwarded to clock value;
  otherwise drop the packet;

At construction of a routing update message:
 For every entry in the current_routing_table
  // Update-content control function
  if the last forwarding has age less than TW_RECENT
   or if the entry passes the filter function
  then the entry is included in the message;
```

Figure 3: ARM update-content control

period control of its neighbors.

Determining the update-content control function involves choosing a proper value for *TW_RECENT* and the filter function. *TW_RECENT* should be in the order of interarrival times of connections and much larger than packet interarrival times. A few seconds should be reasonable.

If the filter function is aggressive, savings in routing traffic would be high. However it would result in longer delay when opening a connection to a new destination. Thus skipping once in every few updates should be reasonable. For instance, skipping every other update would double the delay in the worst case, but reduce the number of routing entries whose destinations are not currently being used by 50%.

The above description of update-content control assumes an underlying routing protocol which uses destination-based information, such as distance vector. However update-content control can be applied to other types of proactive protocols. For example, in a link state protocol the update-content control can decide whether or not to disseminate an outgoing link cost change based on how recently the link was used or whether the link is the current next hop to a recently used destination.

As in update-period control, update-content control is carried out locally and introduces minimal overhead, namely the computation/storage for the extra routing table entry field, *t_forwarded_last*, and the filter function.

In most proactive protocols, every available entry (in routing table or next hop table) is included when update messages are constructed. But not sending all the information each time would not seriously affect the operation of the network. Thus a network in which only some nodes have ARM would still work.

```
CONSTANTS: UPDATE_PERIOD
VARIABLES: dest_seq_no, t_update
           current_routing_table, new_routing_table
            each routing table entry has members
             {dest_id, next_hop_id, num_hops, dest_seq_no}
```

Figure 4: Variables of DSDV protocol

```
MESSAGE FIELDS: sender_id, sender_dest_seq_no,
                routing_update_vector
                 each entry in the vector has members
                  { dest_id, num_hops, dest_seq_no }
```

Figure 5: Routing update message of DSDV protocol

4 DSDV and ARM-DSDV Protocol

Details of DSDV is presented first. How ARM is applied to DSDV producing ARM-DSDV is presented next.

4.1 DSDV protocol

DSDV is a distance vector routing protocol that achieves the freedom of loop through the use of destination sequence numbers [1]. The original DSDV uses both periodic and triggered updates, but we consider only periodic updates in this paper.

Each node maintains variables as shown in Figure 4. Constant *UPDATE_PERIOD* denotes the globally fixed update period for all nodes. Variables *dest_seq_no* and *t_update* denote respectively the destination sequence number and the start of the next update period for the node. Routing entries to all possible destinations are maintained in *current_routing_table*, for data packet forwarding. *New_routing_table* stores routing entries based on the information in routing update messages received in the current period. Each routing entry has members indicating id of the destination, id of the next hop, number of hops to the destination, and destination sequence number of the destination.

As shown in Figure 5, a routing update message contains id of the sender, destination sequence number of the sender, and routing update vector indicating the sender's *current_routing_table* except for the *next_hop_id* members.

As shown in Figure 6, a node handles two routing events: expiration of *t_update*, and reception of a routing update message. On expiration of

```
EVENT: expiration of t_update
 dest_seq_no++;
 update current_routing_table using new_routing_table;
 empty new_routing_table;
 build routing update message from current_routing_table
  and send it;
 t_update += UPDATE_PERIOD;

EVENT: reception of a routing update message
 for x ranging over sender and vector entries
  if ( new_routing_table has no entry for x )
   // if x is sender, x.dest_id is sender_id of the message
  then
   insert x into new_routing_table;
  else // let m be the matched entry in new_routing_table
   if ( (x.dest_seq_no > m.dest_seq_no) or
      ((x.dest_seq_no == m.dest_seq_no) and
       (x.num_hops + 1 < m.num_hops)       )
    // if x is sender, x.dest_seq_no is
    //  sender_dest_seq_no of the message
    // if x is sender, x.num_hops is zero
   then
    m.next_hop_id := sender_id of the message;
    update other members of m using members of x;
 endfor
```

Figure 6: Routing events of DSDV protocol

t_update, *dest_seq_no* of the node is incremented to favor the to-be-broadcast routing update message of the current period. Contents of *new_routing_table* is copied into *current_routing_table*. A routing update message is constructed from entries in *current_routing_table* and then broadcast to the neighbors. Finally, *t_update* is increased by *UPDATE_PERIOD* for the next update.

On reception of a routing update message, if an entry for the sender does not exist in *new_routing_table*, the node creates an entry for the sender and stores *sender_dest_seq_no* along with it. Otherwise, the destination sequence number of the existing entry is updated. Then the node processes the vector entries. A vector entry is inserted into *new_routing_table* if there is no entry with the same *dest_id*. Otherwise, the entry with the same *dest_id* is updated in the table only when the vector entry is favored, i.e., it either has the larger destination sequence number or has smaller number of hops if sequence numbers are identical. Because sequence numbers are monotonically increasing over time, this guarantees new routes are selected over old routes.

4.2 ARM-DSDV protocol

```
VARIABLES: dest_seq_no, t_update
           update_period                    // new for ARM-DSDV
           current_neighbor_table,          // new for ARM-DSDV
           new_neighbor_table               // new for ARM-DSDV
            each entry  has members
             {neighbor_id, t_expiration, mobility_metric}
           current_routing_table
            each entry has members
             {dest_id, next_hop_id, num_hops, dest_seq_no,
              t_expiration, t_forwarded_last}
             // t_expiration, t_forwarded_last: new for ARM-DSDV
           new_routing_table
            each entry has members
             {dest_id, next_hop_id, num_hops,
              dest_seq_no, t_expiration}
             // t_expiration: new for ARM-DSDV
```

Figure 7: Variables in ARM-DSDV protocol

```
MESSAGE FIELDS: sender_id, sender_dest_seq_no
                sender_update_period   // new for ARM-DSDV
                sender_mobility_metric // new for ARM-DSDV
                routing_update_vector
                 each entry in the vector has members
                  {dest_id, num_hops, dest_seq_no}
```

Figure 8: Routing update message of ARM-DSDV protocol

Figure 7 shows variables used in ARM-DSDV. Now *update_period* is varied to control update periods. The two neighbor tables are for update-period control, and the *t_forwarded_last* field of the current routing table is for update-content control as described in section 3. Each entry in the routing tables now has a new member, *t_expiration*, indicating its expiration time.

As shown in Figure 8, ARM-DSDV's routing update messages have two additional fields. *Sender_update_period* is used to compute *t_expiration* of table entries. *Sender_mobility_metric* is used to compute the neighborhood mobility metric.

On expiration of *t_update*, in addition to what is done in DSDV, are computation of next update period and update of *current_neighbor_table* as shown in Figure 9. *Current_routing_table* is updated using *new_routing_table* as follows. First, expired entries in both tables are thrown away. Second, for each entry n in *new_routing_table*, insert n into *current_neighbor_table* if no match is found. Otherwise, favored route of the two is selected and updated in *current_neighbor_table* if necessary. Entries in *current_neighbor_table* are used for forwarding in the current period regardless of their *t_expiration*.

```
EVENT: expiration of t_update
 dest_seq_no ++;
 update_period := value computed as in Fig. 1;
 update current_routing_table using new_routing_table;
 update current_neighbor_table using new_neighbor_table
  as in Fig. 1;
 empty new_neighbor_table and new_routing_table;
 build routing update message from current_routing_table
  and send it;
 t_update += update_period;

EVENT: reception of a routing update message
 tmp_t_expiration := clock +
              sender_update_period in the message + slack;
 insert sender's id into new_neighbor_table as in Fig. 1;
 for x ranging over sender and vector entries
  if ( new_routing_table has no entry for x )
   // if x is sender, x.dest_id is sender_id of the message
   // entry with same id but expired t_expiration
   //  is thrown away and processed here
   insert x into new_routing_table;
  else // let m be the matched entry in new_routing_table
   if ( (x.dest_seq_no > m.dest_seq_no) or
      ((x.dest_seq_no == m.dest_seq_no) and
       (x.num_hops + 1 < m.num_hops)        )
   // if x is sender, x.dest_seq_no is
   //  sender_dest_seq_no of the message
   // if x is sender, x.num_hops is zero
   then
    m.new next_hop_id := sender_id of the message;
    update other members in m using members in x;
 endfor

EVENT: reception/overhearing of a data packet
 tmp_t_expiration := clock +
              minimum_update_period + slack;
 insert sender's id into new_neighbor_table as in Fig. 1;
```

Figure 9: ARM-DSDV update-period control

```
EVENT: forwarding of a data packet
 if ( dest_id of data packet exists in current_routing_table )
 then
  update t_forwarded_last to clock value;
 else
  drop the packet;

EVENT: expiration of t_update
 // when building routing update messages
 for e ranging over current_routing_table entries
  if (( less than TW_RECENT has passed since e.t_last_forwarded )
      or ( filter function allows copy )                        )
  then
   copy e into the routing update message;
 endfor
```

Figure 10: ARM-DSDV update-content control

Routes updated by neighbors on shorter periods may be used beyond their *t_expiration*. An alternative would be stop using those entries for forwarding. But there is a possibility that those entries are being updated in *new_routing_table*. Another alternative would be to keep a single routing table and update entries in place whenever routing update messages are received. This prevents expired entries from being used for forwarding, but then the semantics of DSDV is somewhat compromised. *T_forwarded_last* field remains unchanged if there is a match in *new_routing_table*. Otherwise, it is set to indicate that no forwarding has occurred.

On reception of a routing update message, local variable *tmp_t_expiration* is calculated to be stored as *t_expiration* in neighbor and routing tables. Entries are updated in *new_neighbor_table* as explained in section 3.1. *New_routing_table* is updated like in DSDV except that expired entries in the table are thrown away and never considered as matches.

On reception or overhearing a data packet, sender's information is updated in *new_neighbor_table* as explained in section 3.1. *Minimum_update_period* is the minimum value that can be returned from the update-period control function.

Figure 10 illustrates update-content control of ARM-DSDV. For ARM-DSDV, destination information is recorded on packet forwarding and compared with routing table entries when constructing routing update messages. Other than this, no additional processing specific to DSDV is required for update-content control.

End-to-end workload:
CONNECTION_START_TIME
CONNECTION_DURATION
DPKT_LENGTH, DPKT_RATE

Routing layer:
DSDV : *UPDATE_PERIOD*
ARM-DSDV : *TW_SMOOTH* (for update-period control)
TW_RECENT (for update-content control)

Link layer:
RTS_LEN, CTS_LEN, ACK_LEN
PKT_HEADER, SLOT_TIME, QUEUE_LEN

Physical layer:
TX_RANGE, BANDWIDTH, TA_TIME

Mobility

Figure 11: Layer model and parameters

5 Performance Evaluation Model

To compare the performance of DSDV and ARM-DSDV, we have a simulator (written in CSIM [12]) with the layered structure shown in Figure 11. DSDV and ARM-DSDV share the common lower layers as well as the upper workload layer. The figure also summarizes parameters of each layer. The link layer and physical layer sections can be skipped by readers familiar with IEEE 802.11.

5.1 Mobility

The mobility of a node is modeled by a series of pauses and motions. A pause is defined by a time duration during which the node does not move. A motion is defined by direction, speed, and time duration. A motion can be followed by a pause or another motion. By juxtaposing motions, variable speed can be modeled.

5.2 Physical layer

The physical layer is characterized by three parameters: *TX_RANGE*, *BANDWIDTH*, *TA_TIME*. Two nodes are able to transmit packets to each other only when they are in transmission range, specified by parameter *TX_RANGE*. The transmission time is determined by dividing the packet length by *BANDWIDTH*. Parameter *TA_TIME* specifies the time interval required to turn the antenna from receiving mode to sending mode and vice versa.

Transmission delay is treated as zero because of the relatively short transmission distances. For a successful transmission of a packet from one node to another, the two nodes must stay within transmission range during the entire packet transmission time, and the receiver's antenna must be in receiving mode and should not receive (part or whole of) another packet during the same period of time.

We do not see any substantial reason to sacrifice simplicity and fast simulation by modeling physical layer details like multi-path fading, attenuation of signal strength, and noise. Also the model does not depend on any particular spread spectrum implementation (FHSS, DSSS, etc).

5.3 Link layer

The link layer model replicates IEEE 802.11 standard [13] MAC layer operations as described below.

The MAC layer uses CSMA/CA (Carrier Sense Multiple Access with Collision Avoidance) to share the channel among all nodes. In CSMA/CA, a node starts a transmission only if the channel is free and this is determined by both physical and virtual carrier sensing. Physical carrier sensing is done by analyzing signal strength at the air interface. Virtual carrier sensing is done by looking up the NAV (Network Allocation Vector), a data structure indicating if the channel is currently free of ongoing RTS/CTS handshakes. If the node finds the channel busy, the node performs binary back-off, waiting for its back-off counter to reach zero. The counter is initialized to a random integer that doubles on average at every consecutive failed transmission attempt. The counter decrements when the channel appears free for a specified period of time called *slot time* (typically set to 20 μ seconds).

The RTS/CTS handshake is as follows. A node with a data packet to send first broadcasts an RTS frame, advertising its intention to send a data frame. The intended destination station responds with CTS, advertising that it is ready to receive. The sender transmits the data packet. On valid reception, the destination sends back an ACK. On receiving or overhearing a RTS, CTS or data frame, a node sets its NAV busy for the time duration inferred from data packet length in the frame header. Since RTS/CTS frames are much shorter than data packets, recovery from a collision is much cheaper. RTS/CTS handshakes are not mandatory. They are not used for broadcast packets.

In the model, fragmentation and reassembly are avoided by having packets be smaller than fragmentation threshold. Every node has a send queue that holds both routing and data packets. The maximum size of the queue is given by *QUEUE_LEN* in number of packets. Routing packets have precedence over data packets. FIFO is used as the queuing discipline within data packets and LIFO is used within routing packets. When a routing packet arrives from the upper layer to a full send queue, the data packet most recently enqueued is dropped; if the queue has only routing packets, the oldest routing packet gets dropped.

PKT_HEADER octets are added to upper layer packets in order to accommodate link layer protocol header information. Parameter *SLOT_TIME* specifies the unit in which the back-off counter decrements.

Unlike the physical layer, it is important to have a detailed model of the link layer. Bandwidth consumption by link layer control frames and retransmissions have significant impact on instantaneous residual bandwidth for upper layers.

5.4 Routing layer

For both DSDV and ARM-DSDV, all the operations described previously are modeled into the simulator. We do not model the incremental dump and delayed update features of the original DSDV.

5.5 End-to-end workload

The end-to-end workload is modeled in terms of connections and packets. A connection is defined by source node, destination node, start time (*CONNECTION_START_TIME*), duration (*CONNECTION_DURATION*), data packet length (*DPKT_LENGTH*), and data packet rate (*DPKT_RATE*). The packet interarrival time is constant.

6 Performance Metrics

A simulation scenario is characterized by mobility pattern, end-to-end workload, average node speed, simulation duration, and routing protocols—ARM-DSDV with specified control functions and DSDV with specific update period. For a simulation scenario, we have the following performance metrics:

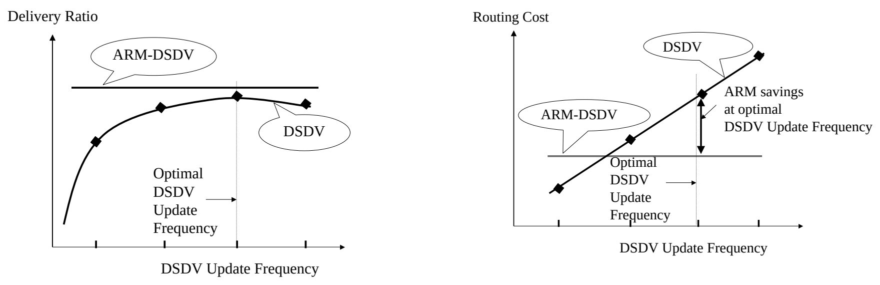


Figure 12: Generic result

- **Delivery ratio**: number of data packets that arrived at their destinations divided by the number of data packets sent out from source nodes.
- **Routing cost**: total number of octets in all the routing update messages sent.
- **Relative performance**: delivery ratio of ARM-DSDV divided by delivery ratio of DSDV.
- **Relative cost**: routing cost of ARM-DSDV divided by routing cost of DSDV.

For a mobility pattern, workload, node speed, and simulation time, we run DSDV several times spanning different update periods and ARM-DSDV once. Over a simulation run, ARM-DSDV keeps adapting its period and contents of routing updates. We expect ARM-DSDV and DSDV performance to be as illustrated in Figure 12. The solid lines connects the results for DSDV over different update frequencies. The shaded line indicates the result for ARM-DSDV. The delivery ratio of DSDV increases rapidly as the update frequency approaches the optimum. After that point, it either plateaus or decreases due to the contention introduced by excessively frequent updates. We expect the delivery ratio of ARM-DSDV to be similar to what DSDV achieves at optimal update frequency. The cost is compared between the cost at an update frequency of DSDV and the cost of ARM-DSDV. The routing cost of DSDV grows with update frequency. We expect the routing cost of ARM-DSDV to be much lower. Even if the routing cost is similar,

Physical layer:
$TX\_RANGE = 100$ m, $BANDWIDTH = 2$ Mbps
$TA\_TIME = 10\ \mu$ sec
Link layer:
$RTS\_LEN = 40$ octets, $CTS\_LEN = 40$ octets
$ACK\_LEN = 34$ octets, $PKT\_HEADER = 58$ octets
$SLOT\_TIME = 20\ \mu$ sec, $QUEUE\_LEN = 100$ packets
Routing layer:
DSDV :
$UPDATE\_PERIOD = 2, 1, 0.5, 0.2, 0.1, 0.05$ sec
ARM-DSDV :
$TW\_SMOOTH = 5$ sec, $TW\_RECENT = 5$ sec
update-period control function =

$$\begin{cases} -0.02 \times \text{ag\_mob\_metric} + 0.5 & \text{if ag\_mob\_metric is in } (0,20] \\ 0.1 & \text{otherwise} \end{cases}$$

filter function = skips one in every two advertisement opportunities

Figure 13: Common parameter values

ARM-DSDV still has an advantage over DSDV in that it does not require to manually configure optimal update frequency.

7 Simulation Results

We present simulation results for two mobility patterns. Figure 13 lists common parameter values that we use for all our simulations. The update-period control function of ARM-DSDV is linear and the maximum and the minimum update periods are 0.5 seconds and 0.1 seconds, respectively. *Ag_mob_metric* in the figure denotes the aggregate mobility metric.

7.1 Mobility pattern 1

Mobility pattern 1 models wireless nodes on vehicles crossing each other at a highway interchange as shown in Figure 14.

There are four groups of vehicles in a 1.5km $\times$ 1.5km geographical area. Each group has two rows of five vehicles. The rows are separated by 40m. Groups moving in opposite direction on adjacent lanes are separated by 20m. All groups have same group speed but individual vehicle speed varies within $\pm 5\%$ of the group speed. Adjacent nodes in each group start with 40m separation but the separation varies due to varying individual speeds.

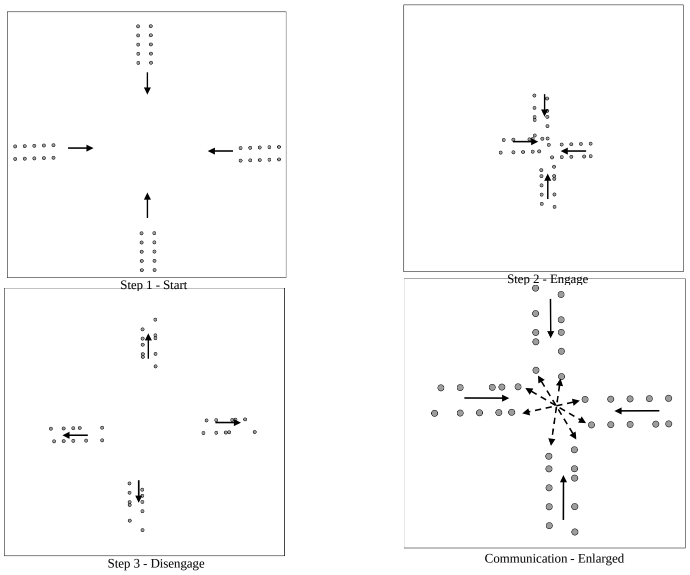


Figure 14: Mobility pattern 1

We have different group speeds of 15 m/sec and 20 m/sec (or 54 km/hr and 72 km/hr, respectively).

As the figure shows, each group starts from the fringe and moves towards the middle. The connections between the nodes are organized into two phases. The first phase of connections start when the simulation begins, and it consists of intra-group connections only. The second phase of connections start when all nodes obtain paths to each other. It consists of inter-group connections only, and these connections are closed before the vehicles pass

End-to-end workload:
CONNECTION_START_TIME =
1st time: when simulation begins
2nd time: when nodes start to have paths to each other
CONNECTION_DURATION = 5 sec
DPKT_LENGTH = 100 octets
DPKT_RATE = (10,3) and (15,5) packets/sec
number of connections = 40
Values of (node speed, simulation time, number of runs):
(speed = 15 m/sec, 70 sec, 10 runs)
(speed = 20 m/sec, 50 sec, 10 runs)

Figure 15: Scenario parameters of mobility pattern 1

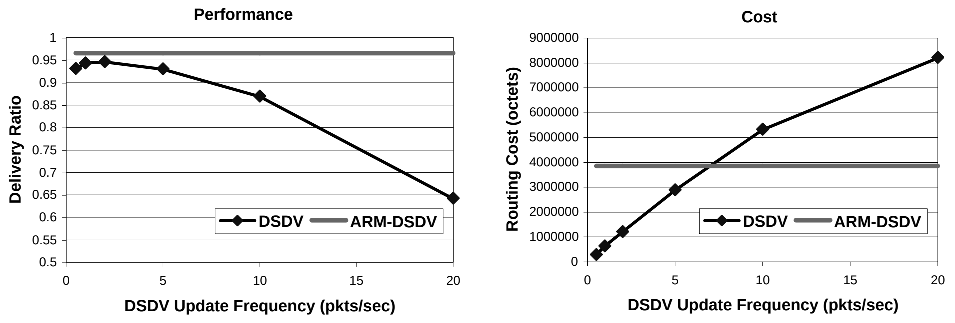


Figure 16: Performance/cost of mobility pattern 1

by each other completely and the paths break down. The last picture of Figure 14 shows which vehicles become connected for the second phase.

We have a low load condition where packet rate is 10 packets/sec for the first communication and 3 packets/sec for the second. The high load condition has packet rates of 15 packets/sec and 5 packets/sec, respectively.

Figure 15 gives scenario parameters. The simulation time depends on the speed of vehicles; the slower the speed, the longer the simulation time in order for vehicles to complete their trips. For each node speed, all the possible combinations of network connections and packet rates are simulated.

We present graphs in Figure 16 for 40 total connections, node speed of 15 m/sec, and high load as an example. Due to the space limitation, results from other scenarios are presented in Table 1. The left graph of Figure 16 presents delivery ratio achieved by DSDV and ARM-DSDV. Curve for DSDV clearly shows that it performs well only around optimal update frequency. ARM-DSDV achieves better delivery ratio at 96.6%. Best delivery ratio of DSDV is 94.7% at 2 updates/sec. Apparently, ARM-DSDV finds appropriate values of update frequency. For DSDV protocol, the next optimal (but less frequent) update frequency often yields similar delivery ratio to that of the optimal update frequency. The suboptimal frequency is 1 update/sec, which gives 94.5% of delivery ratio.

The right graph of Figure 16 illustrates the routing cost of both protocols. As expected, DSDV results in linear increase in the routing. One can observe that ARM-DSDV spends more in routing traffic compared to DSDV

Packet rate (pkts/s)	Number of connections	Speed (m/s)	Optimal and suboptimal dsdv update frequency (msgs/s)	Relative performance (%)	Relative cost (%)
10 then 3	40	15	5 2	100.1 102.0	133.9 316.6
		20	5 2	100.6 105.5	153.2 361.2
15 then 5	40	15	2 1	102.0 102.3	318.6 604.6
		20	5 2	101.3 105.3	150.5 357.1

Table 1: Relative metrics of mobility pattern 1

at optimal or suboptimal update frequencies. However, the routing traffic is kept at reasonable levels to leave bandwidth for the data traffic, indicated by ARM-DSDV's better delivery ratio.

Table 1 presents relative metrics for all the simulation scenarios of mobility pattern 1. The second numbers in the update frequency column are DSDV update frequencies one step below the optimal update frequencies. The relative performance and cost at these frequencies are presented as the second numbers in the relative metrics columns.

When compared at the optimal DSDV update frequencies, ARM-DSDV achieves better delivery ratios. This indicates that ARM-DSDV operates with proper value of update frequencies. The routing costs of ARM-DSDV are reasonable for most of the scenarios.

7.2 Mobility pattern 2

Mobility pattern 2 models a search and rescue operation. Nodes are located relatively close in the beginning. After halting for individually different time periods (less than 5 seconds), all nodes repeat 5 seconds of moving and 5 seconds of pausing until the end of simulation. Initially all nodes move with 2 m/sec (7.2 km/hr) and then, after 200 seconds, they move at the increased speed of 20 m/sec (72 km/hr) until the end of simulation. We have another case of mobility where node speeds change from 0 m/sec to 30 m/sec (108 km/hr).

Figure 17 shows how nodes move over the time; at time 100 seconds, the nodes are spread over 0.5km $\times$ 1km. There are forty nodes. Each node has identical number of connections to randomly selected peers.

Figure 18 shows the scenario parameters. For each node speed, all the

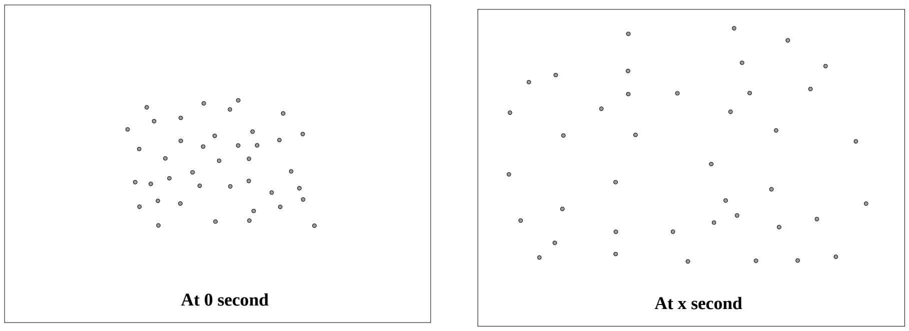


Figure 17: Mobility pattern 2

End-to-end workload:
  CONNECTION_START_TIME = uniformly distributed along simulation time
  CONNECTION_DURATION = 5 sec
  DPKT_LENGTH = 100 octets
  DPKT_RATE = 1, 5 packets/sec
  number of connections = 40, 80
Values of (node speed, simulation time, number of runs):
  (speed = 2 m/sec then 20 m/sec, 400 sec, 10 runs)
  (speed = 0 m/sec then 30 m/sec, 400 sec, 10 runs)

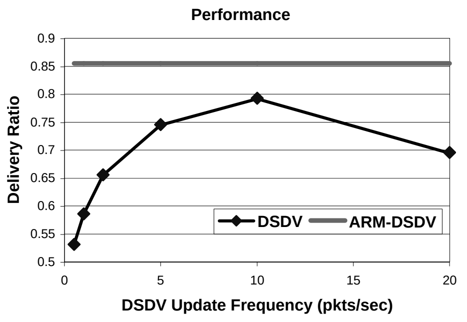
Figure 18: Scenario parameters of mobility pattern 2

possible combinations of network connections and packet rates are simulated also.

Figure 19 shows delivery ratios of ARM-DSDV and DSDV for 80 connections with 5 packets/sec packet rate. The node speeds are 0 m/sec then 30 m/sec. ARM-DSDV achieves 85.6% while the best delivery ratio of DSDV is 79.2% at 10 update/sec.

The routing cost is shown in the right graph of Figure 19. The relative cost at optimal DSDV update frequency is 73.1%. ARM-DSDV achieves better delivery ratio while spending much less in routing traffic.

Table 2 presents relative metrics for all the simulation scenarios of mobility pattern 2. Compared to mobility pattern 1, the savings in routing cost are much greater. Relative delivery ratios are better for ARM-DSDV in all the cases, indicating that ARM-DSDV adapts well to various mobility



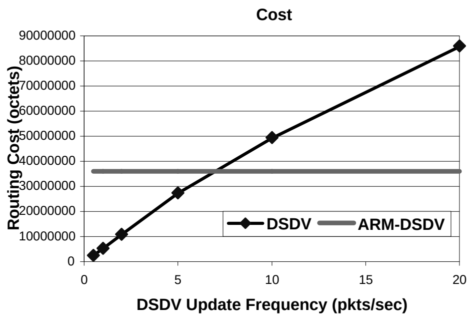


Figure 19: Performance/cost of mobility pattern 2

and route-demand patterns.

Packet rate (pkts/s)	Number of connections	Speed (m/s)	Optimal and suboptimal dsdv update frequency (msgs/s)		Relative performance (%)		Relative cost (%)	
1	40	2 then 20	10	5	102.5	108.4	69.2	132.4
		0 then 30	10	5	105.3	110.8	70.0	130.0
	80	2 then 20	10	5	101.0	104.2	69.2	132.4
		0 then 30	10	5	108.0	114.8	73.1	132.6
5	40	2 then 20	10	5	100.5	103.6	69.3	132.4
		0 then 30	10	5	106.3	111.8	73.8	134.2
	80	2 then 15	10	5	100.1	101.7	70.7	134.0
		0 then 30	10	5	109.7	115.6	74.6	134.3

Table 2: Relative metrics of mobility pattern 2

7.3 Mobility pattern 3

There are two types of nodes in mobility pattern 3, as shown in Figure 20. The first type of nodes are mobile vehicles that are organized into two rows. The rows run in opposite directions on the highway. Each row has twenty nodes. The second type of nodes are static and located at both sides of the highway. Each static group has twenty nodes. Vehicle speeds 5 m/sec (18 km/sec) in the low mobility case and 10 m/sec (36 km/sec) in the high

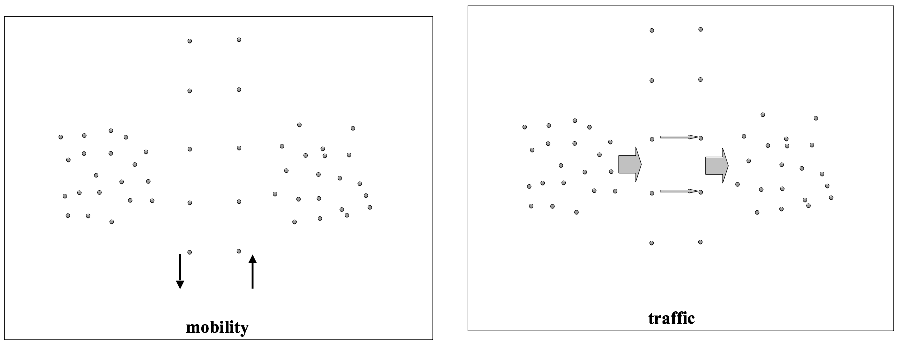


Figure 20: Mobility pattern 3

End-to-end workload:
CONNECTION_START_TIME = uniformly distributed along simulation time
CONNECTION_DURATION = 5 sec
DPKT_LENGTH = 100 octets
DPKT_RATE = 1, 10 packets/sec
number of connections = 20, 40
Values of (node speed, simulation time, number of runs):
(speed = 5 m/sec, 200 sec, 10 runs)
(speed = 10 m/sec, 200 sec, 10 runs)

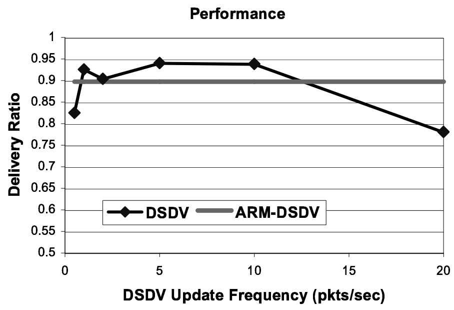
Figure 21: Scenario parameters of mobility pattern 3

mobility case.

Each group of static nodes open the same number of connections to nodes in the other static group. The right figure in Figure 20 shows the movement of the data traffic from the left group to the right group. The mobility pattern is such that all connections need to go through two mobile vehicles moving in opposite directions. Data traffic in the other direction is similar. The low packet rate is 1 packet/sec and the high packet rate is 10 packets/sec.

Figure 21 shows the scenario parameters. All the combinations of number of network connections, packet rates, and node speeds are simulated.

Figure 22 shows delivery ratios of ARM-DSDV and DSDV for the case of 20 connections, 10 packets/sec load, and the vehicle speed of 5 m/sec. Here, ARM-DSDV achieves 89.7% while the best delivery ratio of DSDV



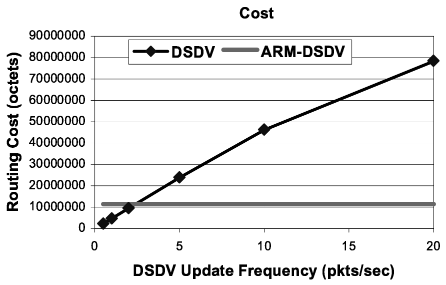


Figure 22: Performance/cost of mobility pattern 3

is 94.0% at 5 update/sec. The routing cost is shown in the right graph of Figure 22. The relative cost at optimal DSDV update frequency is 48.0%.

Table 3 presents relative metrics for all the simulation scenarios. Unlike in mobility pattern 1 and 2, in this mobility pattern 3, ARM-DSDV achieves worse data delivery while spending less in routing traffic. Mobility pattern 3 differs from mobility pattern 1 and 2 in that some of the nodes see high mobility while others see low mobility. In mobility pattern 1 and 2, all nodes see a similar level of mobility. For scenarios with local variations in mobility, each node could advertise routing entries with different update periods. For example, routing entries, which are originated from or propagated through a high mobility region, should be updated with higher frequency. This requires significant changes to the current ARM-DSDV controls. The trade-offs between benefits and costs of having such controls should be analyzed, too. However, even with current version of controls, ARM-DSDV still performs better than some of DSDVs with non-optimized update periods in such mobility patterns as pattern 3.

8 Conclusion

The ARM control mechanism presented here allows a proactive routing protocol to dynamically adjust the period and content of its routing updates in order to adapt to the mobility and route-demand pattern. Furthermore, ARM is completely decentralized, allowing each node to adapt independently, and its overhead is low.

Packet rate (pkts/s)	Number of connections	Speed (m/s)	Optimal and suboptimal dsdv update frequency (msgs/s)		Relative performance (%)		Relative cost (%)	
1	20	5	10	5	89.8	94.6	24.0	46.9
		10	10	5	92.6	94.6	28.5	55.0
	40	5	10	5	89.3	90.7	37.8	73.3
		10	10	5	80.9	81.8	49.9	96.5
10	20	5	5	2	95.5	99.3	48.0	120.5
		10	10	5	85.2	87.1	28.8	55.1
	40	5	10	5	81.6	82.7	41.2	79.3
		10	10	5	85.2	85.4	52.6	100.2

Table 3: Relative metrics of mobility pattern 3

We applied ARM to the DSDV protocol, coming up with ARM-DSDV. We showed that for various mobility and workload scenarios, ARM-DSDV typically achieves better delivery ratio than DSDV with update period optimized for the mobility and workload scenario, while spending reasonable amount in routing cost. Naturally, ARM-DSDV achieves higher data delivery ratio than non-optimized DSDV.

Changes in neighborhood appear to be a good approximation to the actual extent of mobility. Keeping track of forwarded packets appears to be a useful yet inexpensive way of assessing the route-demand patterns.

The Information Dynamics framework provided a useful reference in designing the ARM control mechanism, thereby demonstrating the effectiveness of the framework in constructing real-world agent-based distributed systems. Regarding future work, Information Dynamics can inform us of alternative ways to obtain mobility and route-demand metrics. Sophisticated controls may outperform the simple ones used here. One possible example can be the control schemes suggested in the section presenting mobility pattern 3.

Another area of future work is to make reactive protocols adapt to mobility pattern. This appears to be conceptually harder than ARMing proactive protocols. A proactive protocol can be made adaptive by slowing down its proactivity, reducing routing information being exchanged among nodes. But for a reactive protocol, one would need to add new mechanisms of information gathering.

References

[1] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance vector routing for mobile computers. In *Proceedings of ACM SIGCOMM*, August 1994.

[2] S. Murphy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Applications Journal*, October 1996.

[3] C. Perkins and E. Royer. Ad hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, February 1999.

[4] D. Johnson and D. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, Kluwer Academic, 1996.

[5] V. Park and S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proceedings of IEEE INFOCOM*, April 1997.

[6] D.A. Maltz, J. Broch, and D. Johnson. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proceedings of ACM MOBICOM*, October 1998.

[7] Z. Haas and M. Pearlman. Performance of query control schemes for the zone routing protocol. In *Proceedings of ACM SIGCOMM*, August 1998.

[8] Y. Ko and N. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. In *Proceedings of ACM MOBICOM*, October 1998.

[9] R. Castaneda and S. Das. Query localization techniques for on-demand routing protocols in ad hoc networks. In *Proceedings of ACM MOBICOM*, August 1999.

[10] E. Royer and C-K. Toh. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communications, 6(2):46–55*, 1999.

[11] V. Jacobson. Congestion avoidance and control. In *Proceedings of ACM SIGCOMM*, August 1998.

[12] H. Schwetman. CSIM user's guide. *Microelectronics and Computer Technology Corporation*, 1992.

[13] B. Crow, I. Widjaja, J. Kim, and P. Sakai. IEEE 802.11 wireless local area networks. *IEEE Communications Magazine*, September 1997.

[14] A. Agrawala, R. Larsen, and D. Szajda. Information dynamics: an information-centric approach to system design. In *Proceedings of the International Conference on Virtual Worlds and Simulation.* January 2000.

APPENDIX H

ADT (Attacker-Defender-Target) Game *

Andrzej Kochut[1], Ashok K. Agrawala[1], Ronald L. Larsen[2], A. Udaya Shankar[1]

[1]Department of Computer Science
University of Maryland
College Park, Maryland 20742
{kochut, agrawala, shankar}@cs.umd.edu

[2]Maryland Applied Information Technology Initiative
University of Maryland
College Park, Maryland 20742
rlarsen@deans.umd.edu

CS-TR-4265
UMIACS-TR-2001-45

October 26, 2001

1 Introduction

Information Dynamics is a framework for agent-based systems that gives a central position to the role of information, time, and the value of information. We illustrate system design in the Information Dynamics Framework by developing an intelligence game called ADT involving attackers, defenders and targets operating in some space of locations. The goal of the attackers is to destroy all targets. Target destruction takes place when the number of attackers in the target's neighborhood exceeds the number of defenders in this neighborhood by a value WINNING_DIFFERENCE. The goal of defenders is to prevent attackers from achieving their goal.

The model that we present has attributes that, when appropriately used, can generate either statically or dynamically features such as:

- hierarchy of agents (e.g., information collectors and controllers)
- information fusion
- restricted communication models

*This work is supported partly by DARPA/ Rome Labs, Department of the Air Force, under contract F306020020578 to the Department of Computer Science at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Air Force, DARPA, DOD, or the U.S. Government.

2 Model

2.1 Space

The game is played on a rectangular board of size XMAX by YMAX. Each location L is identified by its (x,y) coordinate, $0 \leq x <$ XMAX, $0 \leq y <$ YMAX, and can be in one of two states, accessible or inaccessible. Accessible locations can be occupied by one or more entities involved in the game. Inaccessible locations can not be occupied by any entity. For a given location L we define neighborhood of L as:

neighborhood(L) = {all accessible locations X for which $|X.x - L.x| \leq 1$ and $|X.y - L.y| \leq 1$ and not $(X.x = L.x$ and $X.y = L.y)$}

2.2 Entities involved in the game

The board is populated with following entities:

- attacker agents
- defender agents
- targets

Each agent has a unique identifier. The parameters ENTITY_IDS and INITIAL_POSITIONS describe, respectively, the identifiers and initial positions of the entities on the board. At each instant an entity occupies one accessible location of the board. An accessible location can have one of the following configurations of entities at any instant:

- empty
- one or more attacker
- one or more defender
- a target

2.3 State of the agent

The state maintained at each agent contains a set of variables encoding the information that the agent gathered, for example, the time-stamped history of scan results, received messages, and results of move operations. Moreover the agent's state may contain any information that agent derived from its state.

2.4 Allowed actions for attackers and defenders

An agent at location L can perform following actions:

- Move(loc): to a location *loc* in its neighborhood. The operation returns boolean value of true when the move succeeded and false when the operation failed (due to the location being inaccessible, occupied by a target, or occupied by agents of different type).

- Scan(loc): a neighboring location. The result of the scan operation consists of following data:
  - occupancy type - specifies the type of entities at the location. Possible values are:
    - 0 - empty location
    - 1 - location occupied by the agents of the same type
    - 2 - location occupied by the agents of the opposite type
    - 3 - location occupied by a target
    - 4 - inaccessible location
  - expected number of entities at the location (used for occupancy type 2)
  - maximal deviation from the expected value (used for occupancy type 2)

  There is a function SCAN_RESULT_FUNCTION that defines distribution of values.
- Chat(agent_id(s)): communicate with other agents. An agent can send one message to CHAT_DEGREE other agents of its kind. The message may contain part or all of the perceived world information of the agent. The message is sent using the destination agent's identifier as the receiver address. An agent can receive multiple messages from other agents of its kind.

2.5 Strategy of the agent

Each agent defines a STRATEGY_FUNCTION that implements its logic of execution. The function may make use of the agent's state and perform actions allowed for the agent.

2.6 Allowed target actions

At each instant a target is located at a board location. The target can move to a neighboring empty location. Each target has a distribution function TARGET_MOVE_DIST defining the probability of target to move to one of neighboring locations or to stay at the current location.

Notes: The TARGET_MOVE_DIST function may depend on location or its neighborhood. For example, the function may define equal probabilities for 8 neighboring locations and currently occupied location. On the other hand we can have the distribution preferring moves in a given direction.

2.7 Goal of the attackers

The goal of the attackers is to destroy all targets. The destruction of a target takes place when the number of attackers in the target's neighborhood exceeds the number of defenders in this neighborhood by WINNING_DIFFERENCE.

2.8 Goal of the defenders

The goal of the defenders is to prevent attackers from destroying targets.

3 Sample strategies

3.1 Attacker's strategy

Each of attacker agents executes following strategy:

- move randomly searching for a target
- once the target is sensed decide whether to chase. To do so agent performs random selection between two choices: follow the target or ignore the target.
- if the attacker decides to follow the target it starts moving in direction of board position where the target was last sensed. Moreover the attacker sends messages to other attackers giving them coordinate of the current target's location.
- each agent that is not currently chasing a target and receives message from other agent starts moving towards the direction specified in the message
- the chase ends once either the chased target is destroyed or a specified timeout expires

3.2 Defender's strategy

Each of defender agents executes following strategy:

- move randomly searching for a target
- once the target is sensed and the number of other defenders in the neighborhood is smaller than specified value the defender remains near the target
- if the defender finds out that the number of attackers exceeds the number of defenders in the target's neighborhood, it sends messages to other defenders which includes endangered target's location
- each defender that is not currently protecting any targets and receives the message starts moving towards the location specified in the message
- the defense stops once the number of attackers in the target's neighborhood becomes smaller than the number of defenders in this area

4 The simulator

We developed a software package that may be used to perform simulation of agents' behavior in the environment described above. The user specifies strategy functions for each entity taking part in the game and the system performs discrete simulation. Figure 1 illustrates the board situation as seen in the user interface at some intermediate points of a ADT game evolution.

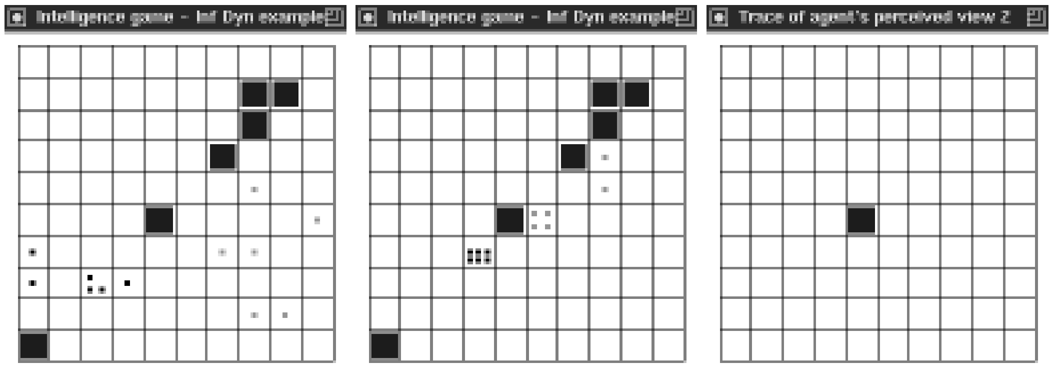

Intelligence game – Inf Dyn example
Intelligence game – Inf Dyn example
Trace of agent's perceived view 2

Figure 1: Example intermediate points in an ADT game evolution. The left figure shows attackers and defenders searching for targets. The middle figure depicts agents converging near the chosen target. The right figure shows the perceived view of the world seen by one attacker.

INFORMATION DYNAMICS APPLIED TO LINK-STATE ROUTING[1]

Hyeonsang Eom, Ashok K. Agrawala, Sam H. Noh, A. Udaya Shankar
({hseom, agrawala, noh, shankar}@cs.umd.edu)
Computer Science Department
University of Maryland
College Park, 20740

CS-TR-#4297
UMIACS-TR-#2001-75

October 25, 2001

Abstract

Information Dynamics [Agrawala, 2000] is an information-centric framework that provides a sufficient understanding of the characteristics of information used in systems for better system design and implementation. In this paper, we describe how to improve link-state routing based on this framework. Link-state routing protocols such as OSPF (Open Shortest Path First) [Moy, 1991] are currently used in many networks. In link-state routing, routes are determined based on link-delay estimates, which are periodically flooded throughout the network. This flooding of link-delay estimates is done without considering the relevance of these estimates to routing quality, i.e. without taking into account the usefulness of the link-delay information. We have developed a new approach that improves link-state routing by estimating future link delays and flooding these estimates only to the extent that they are relevant. This means that we consider the dynamics of the link-delay information and its usefulness. Simulation studies suggest that our approach can lead to significant reductions in routing traffic with noticeable improvements of routing quality in high-load conditions, demonstrating the effectiveness of the framework. We plan to further investigate the conditions where our information-dynamics approach is better than the standard approach.

[1] This work is supported partly by DARPA/Rome Labs, Department of the Air Force, under contract F306020020578 to the Department of Computer Science at the University of Maryland. The views, opinions, and/or findings contained in this report are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the Department of Air Force, DARPA, DOD, or the U.S. Government.

1. Introduction

Information plays a major role in the operation of systems. In general, such information used in or generated by systems is also dynamic in nature. The Information-Dynamics framework [Agrawala, 2000] provides a new perspective for systems with a focus on information, information usefulness (or "value"), and the changes of information and its usefulness over time. Hence, with the framework, we can better understand the interactions between different components of a system that uses information. Such better understanding provides a basis for better system design and implementation. In this paper, we apply the information-dynamics framework to network systems. In particular, we focus on link-state routing where we show that the dynamic nature of link-delay information plays a key role in determining the dissemination of this information, and that the understanding of this role eventually leads to more efficient routing.

In link-state routing, each node in the network maintains a view of the current state of the network. The view is essentially a graph with vertices corresponding to the network nodes, edges corresponding to network links, and for each link, a cost representing an estimate of the current delay on the link. Each node makes (periodic and/or event-driven) measurements of the state of each of its outgoing links. It periodically constructs an estimate for the current delay on the link from these state measurements, and floods these link-delay estimates to all other nodes in the network [Peterson, 1996; Rosen, 1980] so that other nodes can update their views. Each node periodically uses its view to compute least-cost paths to all other nodes, where the cost of a path is the sum of the costs of all the links in the path. When a node receives a workload packet, it forwards the packet to the neighbor that is the next node in the least-cost path to the destination node of the packet.

The information-dynamics framework defines the interactions of entities (the basic building blocks of a system) in terms of information, thereby providing guidance on how a link-state routing system can be improved. In the framework, agents are defined as active entities that have capabilities to autonomously perform operations or actions, and that can also initiate actions. The nodes in a network are agents because they initiate routing activities (series of actions), i.e. periodic view and route updates.

Each agent has its perceived reality, i.e. its view on the *world*. Each node as an agent maintains its perceived reality that includes its network-state view, routes, and route costs. A context of an agent is a relevant part (to given information) of the agent's perceived reality that includes a goal and the cost involved in achieving the goal. In link-state routing, given link-delay information, each node has a context. The goal of each node in a link-state routing system is to route workload packets toward minimizing the end-to-end delays. Each node takes actions with the information, such as using and broadcasting information, in order to accomplish its goal. The main cost involved is the overhead of broadcasting link-delay estimates. The information-dynamics framework allows us to consider this context of each node in improving link-state routing.

The key concept in this whole framework is the notion of information dynamics, that is, the fact that the usefulness of information as well as information itself may change over time. This notion is a basis for improving link-state routing because in a network, the delays of links in a network are dynamic and the confidence level of link-delay information propagated to other nodes in link-state routing decreases over time. To help make use of such dynamic information, particularly in link-state routing the usefulness of information, the framework associates the notion of information utility to an agent as the benefit that the agent can receive by using the information. The utility function of the agent quantifies the benefit. Thus, the concept of information utility provided by the framework helps understand how the nodes in a link-sate routing system can evaluate link-delay information that they exchange. This understanding allows us to improve link-state routing by considering the utility.

With its utility function, an agent is bound to take actions toward maximizing the utility. The context of the agent for information is the domain of the utility function. Since the purpose of link-state routing is to provide information for accurate estimation of the current link delays at low cost, the utility of to-be-sent or received link-delay information may be determined by the closeness of the information to the current delay and the cost of the broadcast.

The usefulness of information to an agent in the context is the difference between the utility achieved with the information and the utility without it. Based on the usefulness of information, an agent decides whether or not to request, send, receive, store, or use the information. The information-dynamics framework allows us to understand that the usefulness of link-delay information to a node in the context of link-state routing is decided by considering its utility, i.e. based on its contribution to the accurate estimation of the current link delays and its overhead.

This research is motivated by the fact that in link-state routing, each node floods its link-cost estimates without regard to whether the estimates will lead to less costly paths. From an information-dynamics perspective, the usefulness of the link-cost information is not considered. This could result in significant unnecessary routing traffic. We propose a new approach that allows each node to disseminate link-cost information only when necessary (for estimating the current link delays i.e., when the information is useful), thereby leading to routing-traffic reduction. This reduction is the primary benefit of our approach.

Ideally, a workload packet should be routed based on the delays it will encounter at each link of the path at the time the packet gets to the link. That is, for each link along a potential route, the node doing the routing needs an estimate of the link delay at the (future) time when the workload packet would arrive at the link. We refer to this future delay as **encountered delay**. In the standard link-state routing, the encountered delay of a link is estimated by the exponential average of past link-state measurements. In our approach, each node estimates the encountered delay of a link based on a model of the dynamic change of the expected link delay given an instantaneous link-delay measurement. This estimation technique allows us to consider the dynamics of not only the link-delay information but also its usefulness. We expect that this estimation technique can improve the workload performance (e.g., delay, throughput). This improvement is an additional benefit of our approach.

The remainder of the paper is as follows: in Section 2, we give a more formal description of the problem that is addressed. We then present the approach that we take in Section 3. The experiments are presented in Section 4. In this section, we describe the network configuration and scenarios for our simulation studies, and present the preliminary comparison results obtained from these studies. Section 5 briefly surveys major related works. Finally, Section 6 concludes our work and summarizes our future work.

2. Problem Formulation

For a link, we treat the delay x at time t as a stationary stochastic process $\{x(t)\}$. Thus, the mean and variance of $x(t)$ are constant (independent of time *t*). Let m and s^2 denote the mean and variance, respectively. Also, the autocorrelation function $E[\{x(t)-m\}\{x(t+t)-m\}]/s^2$ depends only on the lag t and not on time t. Let $r(t)$ denote this autocorrelation function.

Consider the instantaneous conditional mean and variance, respectively, of the delay given a measurement x_0 at time t_0:

$$E\{x(t) \mid x(t_0) = x_0\}\text{, where } t_0 < t$$

$$Var\{x(t) \mid x(t_0) = x_0\}\text{, where } t_0 < t$$

If no other measurement is available, we expect the instantaneous conditional mean to change from x_0 towards m over time. Similarly, we expect that the instantaneous conditional variance to change from zero to s^2 over time. When the measurement is made, the conditional variance is zero because the measurement is valid at that time.

Our approach is to develop estimates for the functions $E\{x(t) \mid x(t_0) = x_0\}$ and $Var\{x(t) \mid x(t_0) = x_0\}$. Then we will use these estimates to do selective broadcasts and determine least encountered-cost paths.

3. Approach

We assume that the conditional mean decays exponentially over time to its steady-state value. Based on this assumption, we use

$$\hat{m}(x_0, t_0, t) = x_0 + (m - x_0)(1 - e^{-a(t-t_0)}) \quad (t \geq t_0)$$

as an estimate of $E\{x(t) \mid x(t_0) = x_0\}$, where a is a non-negative constant to be determined. This is illustrated in Figure 1. Similarly, we use an exponential-decaying estimate $\hat{s}^2(x_0, t_0, t)$ for $Var\{x(t) \mid x(t_0) = x_0\}$, as illustrated in Figure 2.

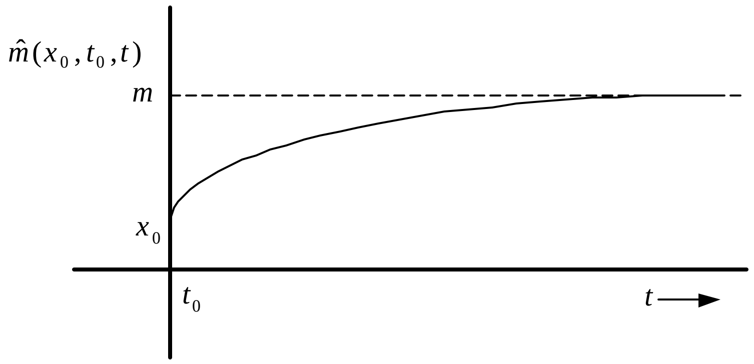


Figure 1 Evolution of the instantaneous conditional delay-mean estimate

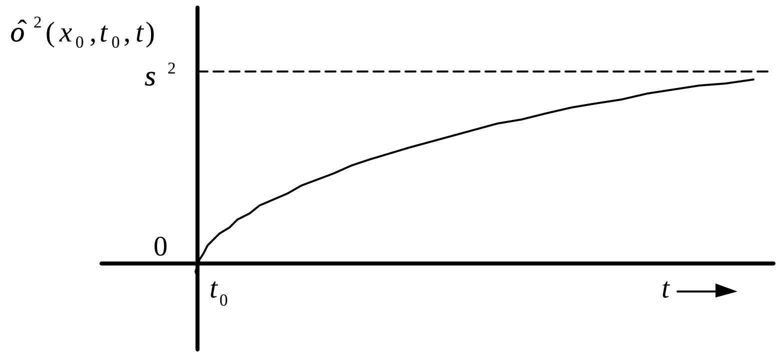


Figure 2 Evolution of the instantaneous conditional delay-variance estimate

Given these estimation functions, a node computes the encountered delay of a packet on a path as follows. Let the path have links $l_1, l_2, \ldots, l_n$, and let the node send the packet into the path at time t_0. Let $\hat{m}^{l_i}(t)$ be the function estimating the encountered delay on link l_i at time t. The estimated encountered delay for the packet on link l_1 is $\hat{m}^{l_1}(t_0)$. The estimated encountered delay for the packet on link l_2 is $\hat{m}^{l_2}(t_0 + \hat{m}^{l_1}(t_0))$. And so on. So the estimated encountered delay for the packet on the path is given by:

$$\hat{m}^{l_1}(t_0) + \hat{m}^{l_2}(t_0 + \hat{m}^{l_1}(t_0)) + \hat{m}^{l_3}(t_0 + \hat{m}^{l_1}(t_0) + \hat{m}^{l_2}(t_0 + \hat{m}^{l_1}(t_0))) + \ldots + \hat{m}^{l_n}(\ldots)$$

Computing path costs in this way, the node would route the packet on the path with the least encountered-delay estimate. Each node determines the least-cost paths using the standard shortest-path algorithm [Dijkstra, 1959] as in link-state routing. Thus routing is as in standard link-state routing except for the path-cost computation. To do this computation, each node maintains a view as in link-state routing except that a measured delay and measurement time are kept for each link. The node updates its view of a local link (i.e., a link outgoing from itself) whenever a workload packet is sent on that link. The node updates its view of a remote link whenever it receives a measurement update for the link. View updates are not periodic.

At each view update, each node broadcasts the updated delay information to its neighbors only if the estimated encountered delay on the corresponding link at that time is significantly different from the steady-state mean. This is how the node determines the dynamic usefulness of the updated delay information with respect to routing-quality improvement, and broadcasts only useful information. We assume that every node knows the steady-state value of each link. Hence, no propagation of link-delay measurements is required beyond some point; if a node does not receive any measurement for a link, it will use the steady-state value.

Each node maintains a routing table that indicates the next hop for each destination. With its view for all links, each node updates its routing table by computing the least-cost paths to all the other nodes just before it decides which of its outgoing links to send the packet onto when it receives a workload packet. This update technique is called the "just-in-time route-update" method. This method allows each node to determine the least-cost paths for the most recent time using the most recent delay information for each link. If the periodic-update scheme of link-state routing were used, the link delays estimated using our approach would be used without any change until the next route-update time. The problem with this is that these estimates could be different from the steady-state values. Thus, the periodic scheme is not suitable for our approach.

4. Simulation

To show the overall applicability of this approach to link-state routing, we compared via simulation a routing scheme using our approach with SPF (Shortest Path First), a link-state routing technique. We call our routing scheme the InfoDyn (Information-Dynamics) scheme.

For simulation studies, we used MaRS, the Maryland Routing Simulator [Alaettinoglu, 1994; Shankar, 1992]. We tried SPF with two kinds of link -cost functions, a delay cost function and a hop-normalized-delay function [Khanna, 1989].

4.1 Network configuration and scenarios

We conducted studies for the NSFNET-T1-backbone topology. In this configuration, there are 14 nodes connected via 21 links. Each link represents two one-way channels. Each node can process a data packet of 544 bytes in 1 ms, and each link channel has 183 KB (1.4 Mbps) bandwidth. We initially assume that there is no propagation delay for each link.

In this network, a workload is generated by FTP source and sink pairs. These sources and sinks are connected to nodes. FTP is regulated by a flow-control mechanism and an acknowledgement mechanism with retransmission. The flow-control mechanism is a static window-based scheme implemented in MaRS. This scheme consists of two windows: produce and send windows. We set the produce-window size to infinity, and the send-window size to eight. Also, we initially use 120 seconds as the total simulated time.

There are two kinds of FTP flows: regular and on-off flows. In each regular flow, the source starts transmitting packets at time 0, and sends as many packets as possible with an inter-packet production delay of 1 ms. For each on-off flow, there are alternating constant-length on and off intervals. Each on-off flow starts at a different time (from 0 to 24 seconds), and has a different length (from 20 to 120 seconds). Also, a certain number of packets are produced at once at the beginning of on intervals while no packets are produced during off intervals. The number of packets for each on interval is determined so that the packets of that number would be successively transmitted during the on interval without any flow-control mechanism and without any other flow. Specifically, the number is the length of an on interval divided by the transmission time of a data packet, where the transmission time is the packet size divided by the link bandwidth.

We initially consider five scenarios in this network configuration: N0 – N4. The level of queuing delay of these scenarios is high: in the best cases (lowest-average-delay cases after parameter tuning) of using SPF with 1 second route-update intervals, the average queuing-delay portions of the average round-trip delay per packet are around 94 %. Also, the utilizations (the average fractions of the time when packet queue size > 1) range from 0.73 to 0.74. There are 121 FTP flows in Scenario N0 to N3, and 131 flows in Scenario N4. Table 1 shows the differences between the scenarios. In particular, Scenario N4 has two hot spots (each of which receives packets from every other node).

Table 1 Differences in the FTP-flow characteristics between scenarios

Scenario	*Number of Regular Flows*	*Number of On-Off Flows*	*Length of On-Off Intervals (Seconds)*
N0	60	61	5
N1	60	61	10
N2	60	61	15
N3	0	121	5
N4	55	76	5

4.2 Preliminary results

For the InfoDyn scheme, we used an exponential-change-rate (a) value and a threshold value for the selective broadcast of routing packets, during each simulation run for each scenario. We tried seven threshold values. Also, we tried eight a values across the full value range in each of these different-threshold-value cases. As the steady-state value of each link in each simulation run using the InfoDyn scheme, we used the sample delay mean of the corresponding link computed in a simulation run using SPF with 1 second route-update intervals for the same scenario.

The use of the InfoDyn scheme without any routing-packet broadcast (thereby with only local-link view update) is called the InfoDyn Short-Term Steady-State (STSS) case. Hereafter, "best" means leading to the lowest average round-trip delay per packet.

4.2.1 InfoDyn Short-Term Steady-State (STSS) case

The InfoDyn STSS case with the best a of the exponential model results in 3 to 8 % reduction in the Average (Avg) Round-Trip (RT) delay per packet and 4 to 22 % reduction in the standard deviation (STD) in all scenarios compared with the best cases of using SPF with 1, 10, and 30 second route-update intervals - we obtained the best result of using SPF for each combination of a route-update-interval length and a scenario by tuning several cost-function parameters. Figure 3 shows these reductions. Note that there are no routing packets sent out in this InfoDyn case while 75,642, 7,602, and 2,562 routing packets are sent out with 1, 10, and

30 second route-update intervals, respectively, in the SPF cases. These results imply that when every node knows the "long-term" steady-state delay-mean values of all links and uses our routing approach, flooding requirements can be significantly reduced with noticeable reductions in the average delay and the variance, compared with the standard link-state routing approach where each node periodically broadcasts "short-term" steady-state values (exponential averages) for link delays.

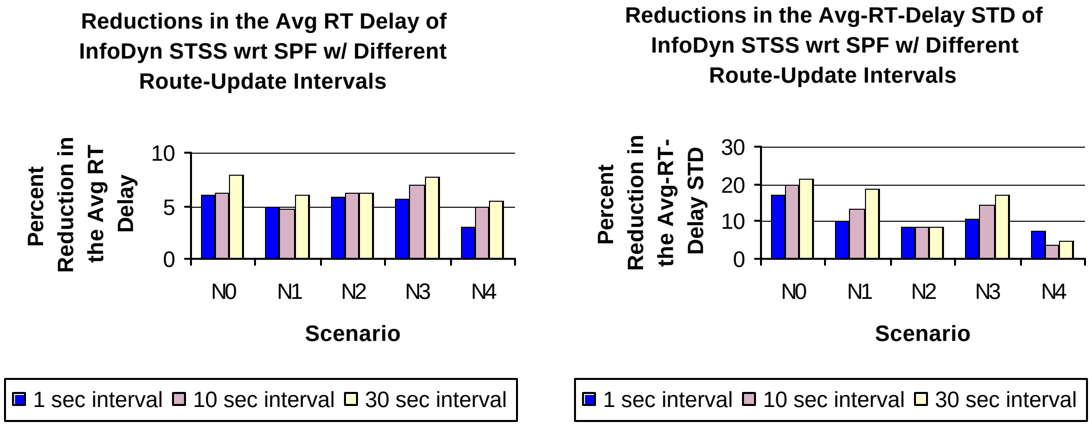


Figure 3 Reductions in the average round-trip delay and STD of InfoDyn STSS

4.2.2 Impact of routing-packet broadcast

For each scenario with a fixed a value, the average delay and STD are almost the same across simulation runs with different threshold values except for those runs where a very large number of routing packets are broadcast. For example, Figure 4 shows the impact of varying the threshold value in Scenario N3 (the all-on-off case) when the best a is used. There are three charts. The left-most and middle charts indicate the changes in the average delay and STD, respectively, depending on the threshold value used. The right-most chart shows the numbers of routing packets used for different threshold values. The smaller the threshold value, the more routing packets are sent out. When about 40,000 routing packets are used (the 100 ms threshold-value case), there are 0.53 ms and 0.66 ms increases in the average delay and STD, respectively, compared with 152.03 ms average delay and 92.24 ms STD of the best case (the 130 ms threshold-value case). Similar impacts of routing-packet broadcast are observed for the other scenarios. Figure A1 (Pages 16 and 17) in APPENDIX shows the same three charts in each row for each of the other scenarios. As in the figure, for the threshold values that correspond less than 100,000 routing packets in each scenario, the variation of the average delays is within 1 ms and that of STDs is within 5 ms. Note that these numb ers are the scale units in the delay and STD charts, respectively.

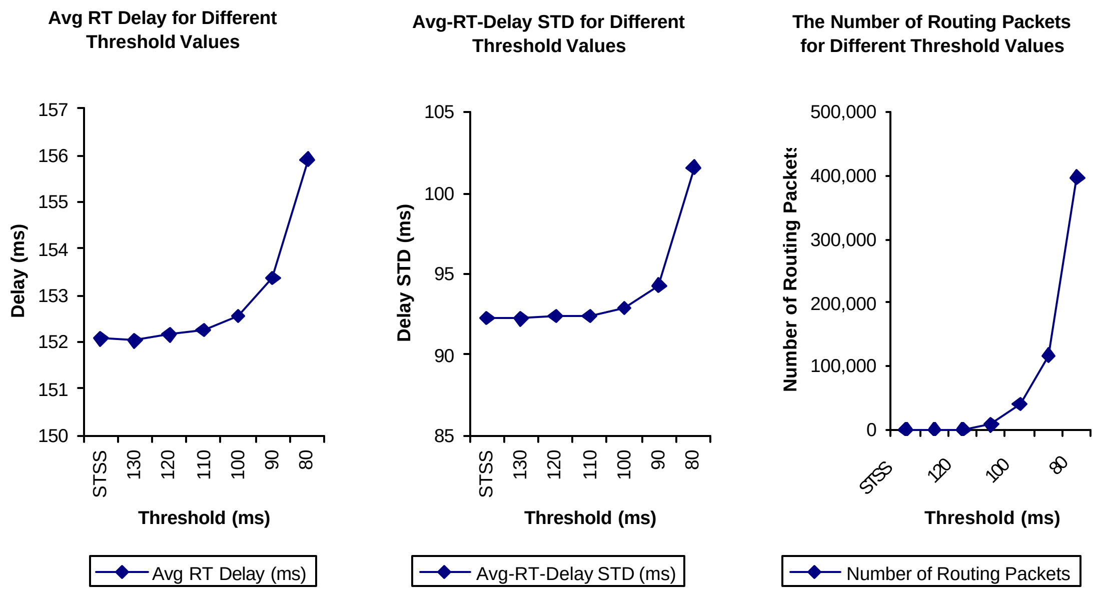


Figure 4 Impact of varying the threshold in Scenario N3 (when using the InfoDyn scheme w/ the best a)

The average delay and STD range from 152.0 to 161.9 ms and from 89.7 to 109.1 ms, respectively, in the best cases of using the InfoDyn scheme (with the best a) in all scenarios when routing packets are broadcast. Compared with this best case for each scenario, the InfoDyn STSS case with the same best a leads to increases in the average delay and STD by up to 0.1 ms and 0.3 ms, respectively. The reason why these increases are small is that the impact of a routing packet on routing quality is transient: the encountered link delay estimated by the receiving node using the delay measurement contained in the packet soon becomes the steady-state value. These results indicate that each node may not need to broadcast link-delay measurements when using the InfoDyn scheme.

4.2.3 Impact of varying the a value

There are two possible sources for the routing-quality improvement: use of the long-term steady-state link-delay means and link-delay estimation with the exponential delay-mean change. To see the influence of each of these factors, we first set a to infinity. Then, the link-delay means are used without any change in route determination. In the InfoDyn STSS case, this Static-Routing case leads to up to 5 % and 18 % increases in the average delay and STD, respectively, in four scenarios and 2 % and 9 % decreases, respectively, in one scenario compared with the best cases of using SPF. These results mean that the use of the link-delay means is not a source of routing-quality improvement in most cases. However, the use of the best a results in 4 to 11 % and 7 to 22 % decreases in the average delay and STD, respectively, in all scenarios compared with these Static-Routing cases. These results indicate that the selection of the a value is crucial for routing-quality improvement.

The best routing quality is achieved with the same a across all scenarios in the case of using the same threshold value or in the InfoDyn STSS case. For example, Figure 5 shows the effects of using different a values in Scenario N3 in the InfoDyn STSS case. The left and right charts indicate the changes in the average delay and STD, respectively, depending on the a value. As in the figure, the average delay and STD increase as the a value used becomes more and more different from the value (1000) for the best result. Similar trends are observed for the other scenarios. Figure A2 (Pages 18 and 19) in APPENDIX shows the same two charts in each row for each of the other scenarios. Therefore, if we can find the best or a near-best setting in one case, we may reduce the average delay and STD by using the same setting in other cases. In fact, the range of the a value for routing-quality improvement is wide. Table 2 shows the a ranges of the InfoDyn STSS cases in all scenarios that lead to decreases in the average delay with respect to the best SPF cases. Therefore, the parameter tuning is not required.

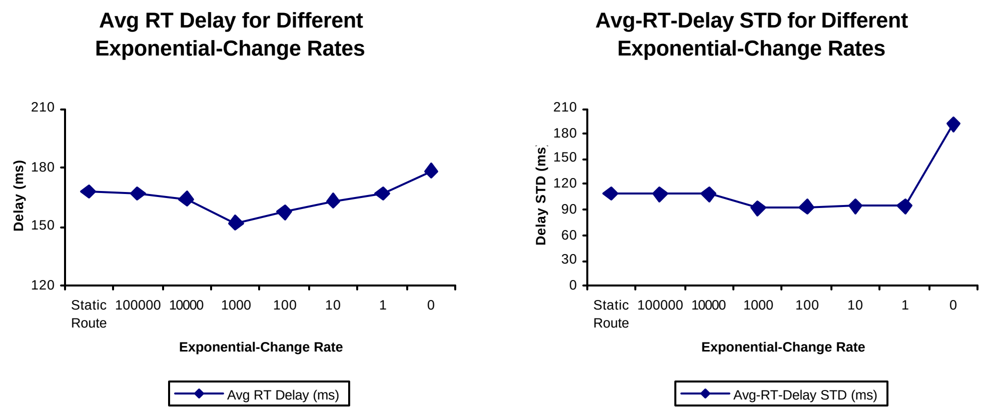


Figure 5 Impact of varying the a value in Scenario N3 (in the InfoDyn STSS case)

Table 2 a Ranges of the InfoDyn STSS cases leading to decreases in the average RT delay wrt the best SPF cases

Scenario	*Rate (Circled if Routing Quality is Improved)*							
	Static Route	*100,000*	*10,000*	*1,000*	*100*	*10*	*1*	*0*
N0	O	O	O	O	O	O		
N1		O	O	O	O	O		
N2				O	O	O		
N3				O	O			
N4				O	O			

5. Related Work

Typically delays vary and change rapidly in a network. For example, at a fine-grained level, the characteristics of the Internet are highly dynamic [Agrawala, 1998]. Such dynamics in networks make it difficult to estimate encountered link delays. Many researchers have investigated the dynamic behavior of networks such as the dynamics of end-to-end Internet packet delays. [Agrawala, 1998; Labowitz, 1998; Paxson, 1999; Pointek, 1997; Sanghi, 1993].

For statistical uncertainty modeling concerning information estimation, there are two basic approaches: modeling based on past observations followed by extrapolation, and modeling via the analysis of factors that determine the information at the target estimation time. An example of the first modeling approach is a time-series model such as an AutoRegressive Integrated Moving Average (ARIMA) model [Box, 1994; Chatfield, 1984]. An example of the second is a regression model for factor(s)-and-effect information pairs (or tuples). The parameters of both modeling approaches can be estimated using least-squares fitting [Trivedi, 1982].

6. Conclusion and Future Work

Our preliminary results indicate that our approach is promising. When we compared our routing scheme based on a new link-delay-estimation technique with SPF via simulation for various FTP-workload scenarios with the NSF-T1-backbone network topology, we found that our routing scheme could achieve 100 % reductions in routing traffic with 3 to 8 % decreases of the average round-trip delay per packet in high-load conditions.

These routing-traffic reduction and routing-quality improvement resulted from the estimation of future (encountered) link delays based on the dynamics of the expected link delay given an instantaneous link-delay measurement, and from the consideration of the dynamic usefulness of the link-delay measurement via this estimation. Hence these benefits demonstrate the effectiveness of the information-dynamics framework.

We plan to characterize the situations where we can improve link-state routing by using our information-dynamics approach. For this research, we plan to further investigate the effectiveness of our routing scheme via extensive simulation studies with different patterns of dynamic workload and/or with different parameter settings for the network. We will try random scenarios created by enabling random parameters such as the average number of packets per FTP connection and the average delay between connections. In addition to FTP workload, we will try other types of workload. Also, we will create and try scenarios with different levels of load condition to investigate the relationship between load level and the benefit of using our scheme. In addition, we will try higher and/or different link-bandwidth and propagation-delay values. Based on the results of these studies, we will determine the characteristics of the situations that lead to significant routing-traffic reductions with routing-quality enhancements in the case of using our approach, compared with standard link-state routing.

Each node needs to estimate the steady-state value of each link in order for our approach to be practical. Our preliminary results indicate that each node may compute the sample mean of the delay of each local link using the standard approach for a long period of time, and flood the sample mean periodically (but, at a lower frequency) so that all other nodes can use it as the steady-state value. We will also study different ways to compute the sample mean and provide a guideline for the computation.

7. Reference

[Agrawala, 2000] A.K. Agrawala, R. Larsen, and D. Szajda "Information Dynamics: An Information-Centric Approach to System Design," *Proceedings of the International Conference on Virtual Worlds and Simulation*, January 2000.

[Agrawala, 1998] A.K. Agrawala, The NetCalliper Project, http://www.cs.umd.edu/projects/sdag/netcalliper, 1998.

[Alaet tinoglu, 1994] C. Alaettinoglu, A.U. Shankar, K. Dussa-Zieger, and I. Matta, "Design and Implementation of MaRS: A Routing Testbed," *Journal of Internetworking: Research & Experience*, 5 (1):17-41, 1994.

[Box, 1994] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel, *Time Series Analysis: Forecasting and Control,* Prentice Hall, 1994.

[Chatfield, 1984] C. Chatfield, *The Analysis of Time Series: an Introduction*, Chapman and Hall, London & New York, 1984.

[Dijkstra, 1959] E.W. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik*, 1, pp. 269-271, 1959.

[Khanna, 1989] A. Khanna and J. Zinky, "The Revised ARPANET Routing Metric," *Proceedings of ACM SIGCOMM*, pp. 45-56, 1989.

[Labowitz, 1998] C. Labowitz, G.Malan, and F. Jahanian, "Internet Routing Instability," *IEEE/ACM Trans. Networking*, 6 (5):515-528, October 1998.

[Moy, 1991] J. Moy, *Open Shortest Path First v.2*, RFC 1247, 1991.

[Paxson, 1999] V. Paxson, "End-to-End Internet Packet Dynamics," *IEEE/ACM Trans. Networking*, 7 (3):277-292, June 1999.

[Peterson, 1996] LL.Peterson and B.S. Davie, *Computer Network: A Systems Approach*, Reprographic Services, 1996.

[Pointek, 1997] J. Pointek, F. Shull, R. Tesoriero and A.K. Agrawala, "NetDyn Revisited: A Replicated Study of Network Dynamics", *Computer Networks and ISDN Systems*, 29(7):831-840, August 1997.

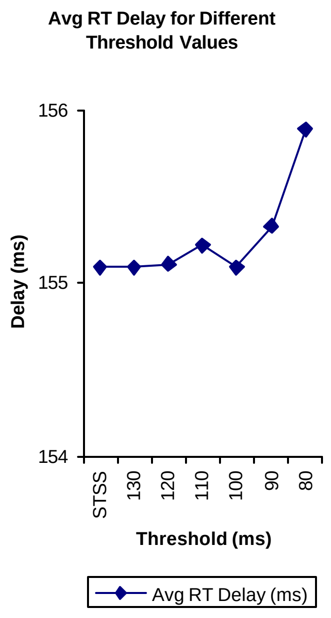
[Rosen, 1980] E.C. Rosen, "The Updating Protocol of ARPANET's New Routing Algorithm," *Computer Networks*, 4(1), pp. 11-19, 1980.

[Sanghi, 1993] D. Sanghi, O. Gudmundsson, and A.K. Agrawala, "A Study of Network Dynamics," *Computer Networks and ISDN Systems*, 26(3), pp. 371-378, November 1993.

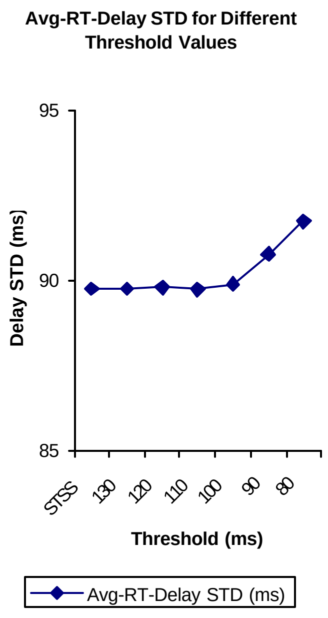
[Shankar, 1992] A.U. Shankar, C. Alaettinoglu, I. Matta, and K. Dussa-Zieger, "Performance Comparison of Routing Protocols using MaRS: Distance-Vector versus Link-State," *Proceedings of ACM SIGMETRICS,* pp. 181-192, 1992.

[Trivedi, 1982] K.S. Trivedi, *Probability & Statistics with Probability, Queuing and Computer Science Applications*, Prentice Hall, 1982.

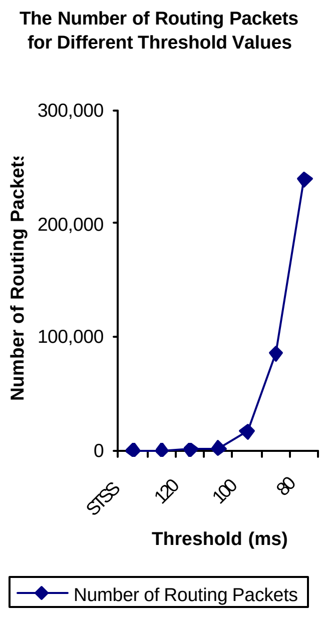
APPENDIX Supplementary Figures



[a]

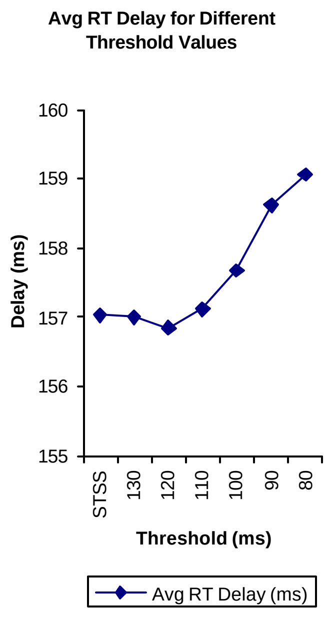


[b]

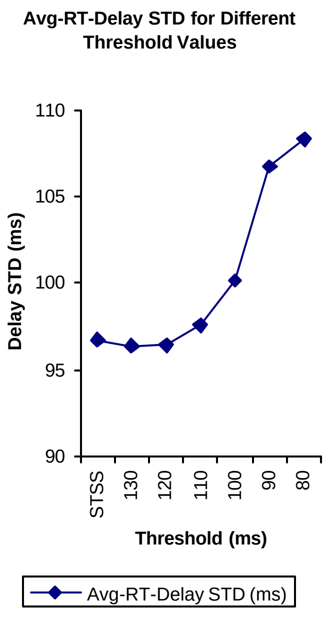


[c]

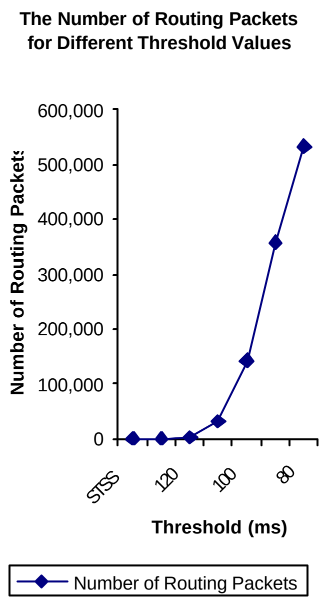
([a], [b], and [c]: Scenario N0)



[d]

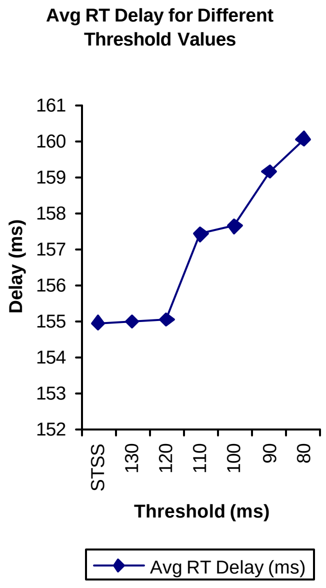


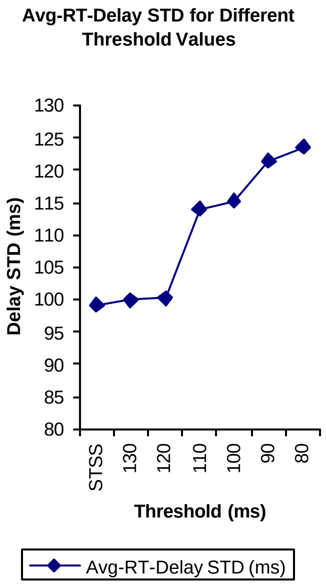
[e]

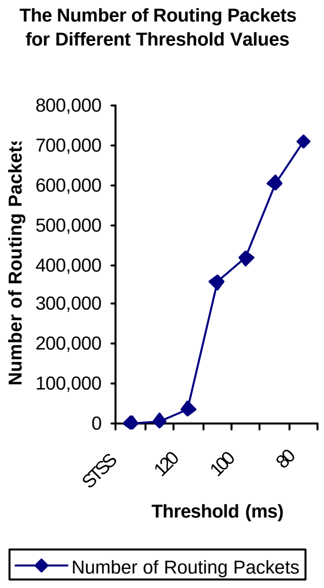


[f]

([d], [e], and [f]: Scenario N1)

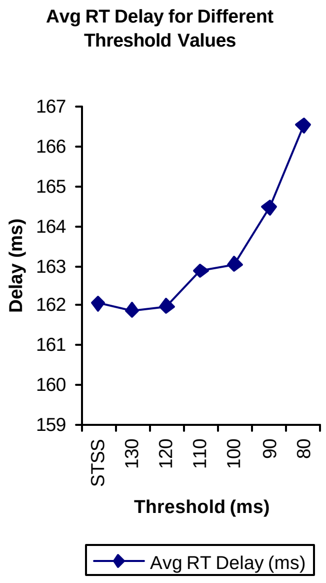


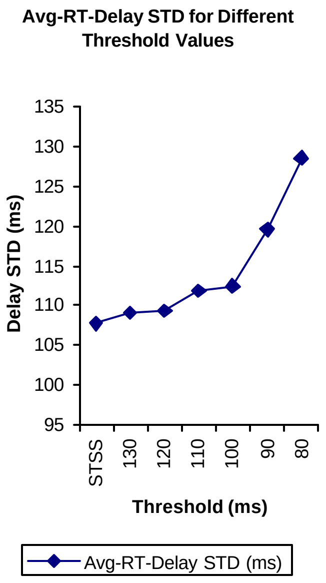


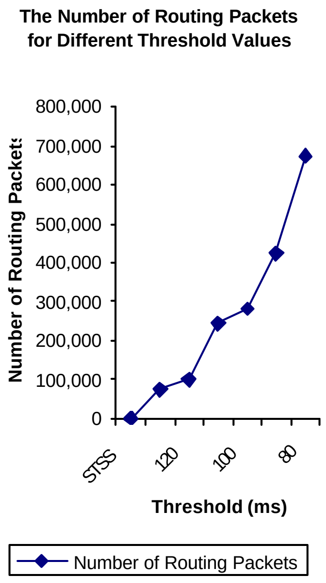


[g] [h] [i]

([g], [h], and [i]: Scenario N2)



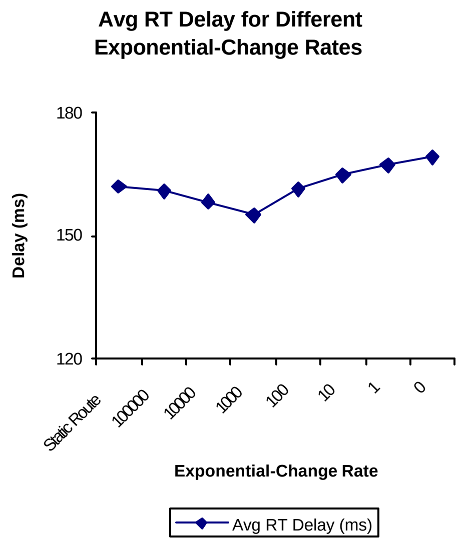




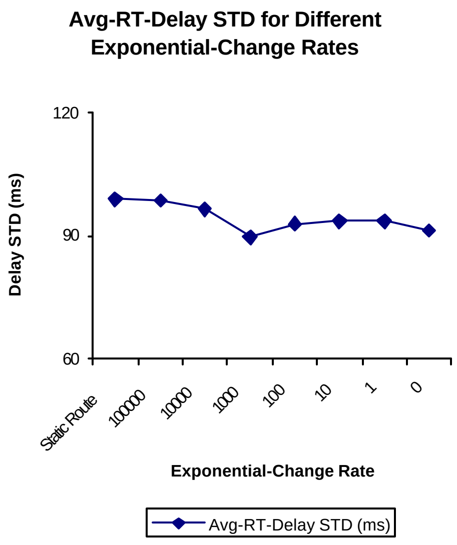
[j] [k] [l]

([j], [k], and [l]: Scenario N4)

Figure A1 Impact of varying the threshold in Scenarios N0, N1, N2, and N4 (when using the InfoDyn scheme w/ the best a)

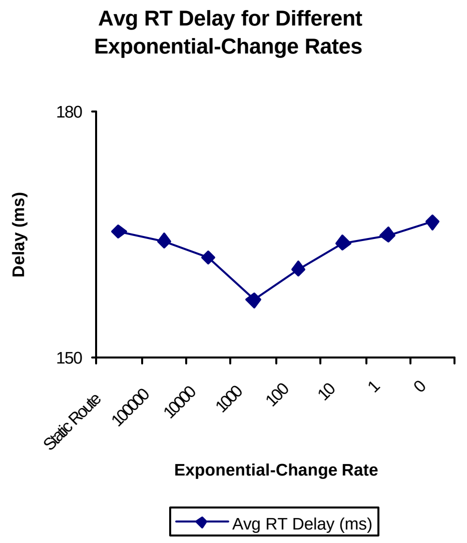


[a]

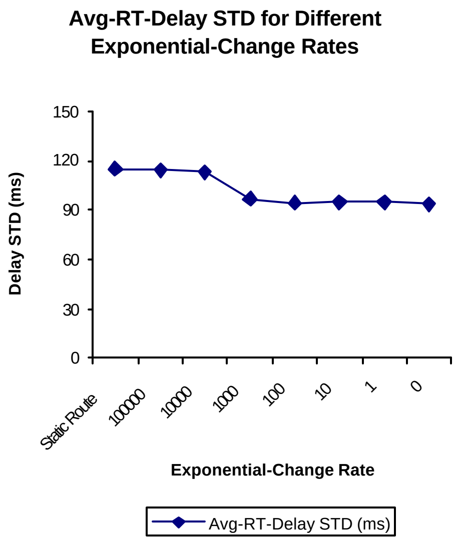


[b]

([a] and [b]: Scenario N0)

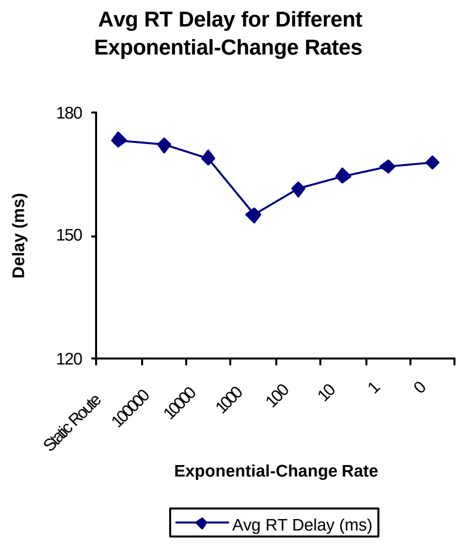


[c]

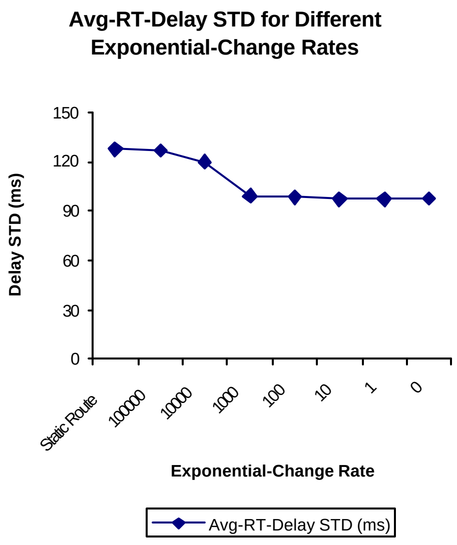


[d]

([c] and [d]: Scenario N1)

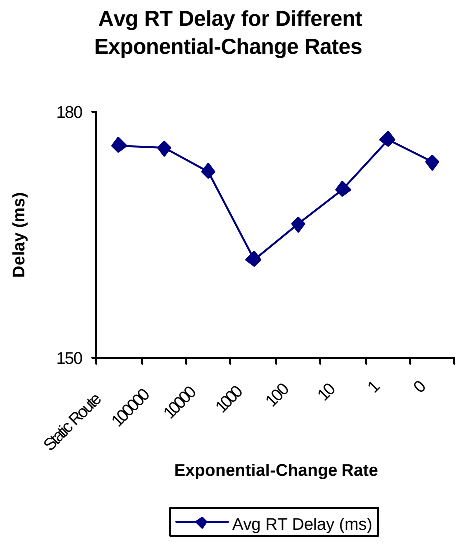


[e]

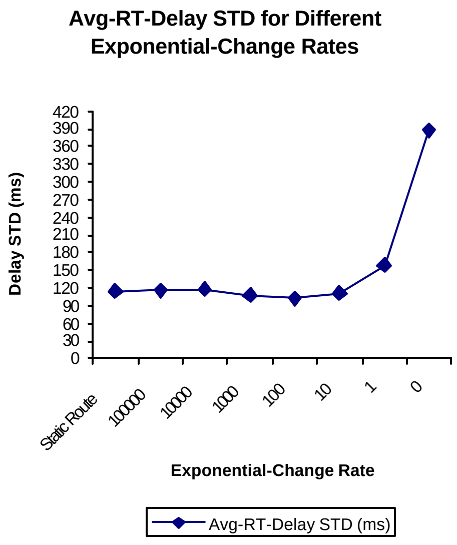


[f]

([e] and [f]: Scenario N2)



[g]



[h]

([g] and [h]: Scenario N4)

Figure A2 Impact of varying the a value in Scenarios N0, N1, N2, and N4 (In the InfoDyn STSS cases)