ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Routing Queries in Unstructured P2P Networks

B.Siva
M.Tech Student
Gokul Institute of Technology & Sciences
Piridi, Vizianagaram, Bobbili, Andhra Pradesh.

G Bhagya Lakshmi
Assistant Professor
Gokul Institute of Technology & Sciences
Piridi, Vizianagaram, Bobbili, Andhra Pradesh.

ABSTRACT:

Finding a document or resource in an unstructured peer-to-peer network can be an exceedingly difficult problem. In this paper we propose a query routing approach that accounts for arbitrary overlay topologies, nodes with heterogeneous processing capacity, e.g., reflecting their degree of altruism, and heterogenous class-based likelihoods of query resolution at nodes which may reflect query loads and the manner in which files/resources are distributed across the network. The approach is shown to be stabilize the query load subject to a grade of service constraint, i.e., a guarantee that queries' routes meet pre-specified class-based bounds on their associated a priori probability of query resolution. An explicit characterization of the capacity region for such systems is given and numerically compared to that associated with random walk based searches. Simulation results further show the performance benefits, in terms of mean delay, of the proposed approach. Additional aspects associated with reducing complexity, estimating parameters, and adaptation to class-based query resolution probabilities and traffic loads are studied.

EXISTING SYSTEM:

- In a purely unstructured P2P network, a node only knows its overlay neighbors. With such limited information, search techniques for unstructured networks have mostly been based on limited-scope flooding, simulated random walks, and their variants.
- Much research in this area has focused on evaluating these search techniques based on the contact time, i.e., number of hops required to find the target, using the spectral theory of Markov chains on (random) graphs, see e.g., Unfortunately in heterogenous settings where service capacity or resolution likelihoods vary across peers, such search techniques perform poorly under high query loads.
- The inefficiencies of purely unstructured networks can be partially addressed by hybrid P2P systems, e.g., FastTrack and Gnutella2.

DISADVANTAGES OF EXISTING SYSTEM:

- In structured networks the difficulty of search/discovery is shifted to that of maintaining the structural invariants required to achieve efficient
- In query resolution particularly in dynamic settings with peer/content churn or when reactive load balancing is required.
- Standard backpressure-based routing our policies suffer from a major drawback: each node needs to share the state of its potentially large number of non-empty queues with its neighbors.
- Complexity problem will be also raised.

PROPOSED SYSTEM:

- Given a hybrid P2P topology and query classification, we propose a novel query resolution mechanism which stabilizes the system for all query loads within a 'capacity region', i.e., the set of loads for which stability is feasible.

- ❖ Essentially, our policy is a biased random walk where forwarding decision for each query is based on instantaneous query loads at super-peers.
- ❖ To balance the load across heterogeneous super-peers, the policy aims at reducing the differential backlog at neighboring super-peers, while taking into account the class and history information to improve the query's resolvability.
- ❖ Our policy draws upon standard backpressure routing algorithm, which is used to achieve stability in packet switching networks,
- ❖ We propose a query forwarding mechanism for unstructured (hybrid) P2P networks with the following properties.
- ❖ It dynamically accounts for heterogeneity in super-peer's 'service rate,' reflecting their altruism, and query loads across the network. To the best of our knowledge, this is the first work to rigorously account for such heterogeneity in devising a search mechanism for P2P networks.
- ❖ It is based on classifying queries into classes. This classification serves as a type of name aggregation, which enables nodes to infer the likelihoods of resolving class queries, which, in turn, are used in learning how to forward queries.
- ❖ Our approach is fully distributed in that it involves information sharing only amongst neighbors, and achieves stability subject to a Grade of Service (GoS) constraint on query resolution. The GoS constraint corresponds to guaranteeing that each query class follows a route over which it has a reasonable 'chance' of being resolved.
- ❖ We provide and evaluate several interesting variations on our stable mechanism that help significantly improve the delay performance, and further reduce the complexity making it amenable to implementation.

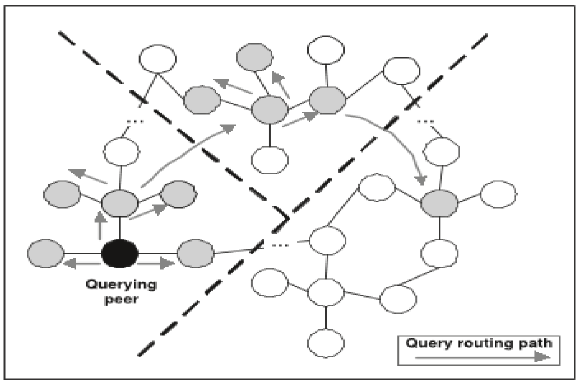
ADVANTAGES OF PROPOSED SYSTEM:

- ❖ Reducing complexity
- ❖ Estimating parameters, and adaptation to class-based query resolution probabilities and traffic loads are studied.
- ❖ Stable Policies
- ❖ Estimating Query Resolution Probabilities
- ❖ Alternate Grades of Service Strategies
- ❖ It is based on classifying queries into classes
- ❖ The GoS constraint corresponds to guaranteeing that each query class follows a route over which
- ❖ It has a reasonable 'chance' of being resolved
- ❖ This provides a basis for substantially reducing complexity by approximations

ALGORITHM

- ❖ *Basic Backpressure Algorithm*
- ❖ The weights used in above algorithm for each link are different from those used in traditional multi-commodity backpressure algorithm

SYSTEM ARCHITECTURE:



MODULES:

- ❖ Constructing System Model
- ❖ Stable Query Forwarding Policy
- ❖ Estimating Query Resolution Probabilities
- ❖ Reducing Complexity

MODULES DESCSRIPTION:

Constructing System Model

- ❖ In the initial module, we develop the system with the entities required to show the proposed model with the evaluation proof of your novel contribution. So we develop the module with nodes. We assume that time is slotted, and each super-peer has an associated service rate, corresponding to positive integer number of queries it is willing to resolve/forward in each slot.

- We assume that super peers keep a record of files/resources available at subordinate peer. This information is communicated to super peers when a subordinate peer joins a super peer. Subordinate peers may initiate a query request at a super peer, but do not participate in forwarding or query resolution
- If a class query at node cannot be resolved it may be forwarded to one of its neighbors. The likelihood a node can resolve such a query depends not only on its class but also its*history*, i.e., the set of nodes it visited in the past. Note that the history is not ordered. For example, suppose 3 nodes in a network partition files/resources associated with class. If two of these nodes attempted and failed in resolving a given class query then it will for sure be resolved at the third node. In other contexts, if a search for a particular media file failed at many nodes, it is more likely that the file is rare, and the conditional likelihood that it is resolved at the next node might lower.

Stable Query Forwarding Policy

- In this module, we will propose a query scheduling and forwarding policy that ensures the GoS for each class, is distributed, easy to implement, and is stable. We begin by defining the stability for such networks and the associated capacity region. The module is develop such that the following aspects arising in P2P search systems: (a) history dependent probability of query resolution at each node, (b) updates in 'types' of queries as they get forwarded to different nodes, (c) computing the quality of service received by query via its history and designing an appropriate exit strategy upon receiving enough service.
- While the routing decisions are to be based on instantaneous queue loads at the neighbors, the decisions themselves affect the type/queue to which a query belongs. In this module, we develop a distributed dynamic algorithm where each node makes decisions based on its queue states and that of its neighbors and only needs to know

Estimating Query Resolution Probabilities

- In this module, we develop the estimation of query resolution probabilities. So far we have assumed that resolution probabilities for queries of different types are known. In practice they can be easily estimated. In order to ensure unbiased estimates can be obtained at each node, suppose a small fraction of all queries is marked 'RW', forwarded via the random walk policy with a large TTL, and given scheduling priority over other queries.
- With a sufficiently large TTL this ensures that each node will see a random sample of all query and types it could see and thus allow for unbiased estimates. All queries which are not marked 'RW' are treated according to our backpressure policy based on the estimated query resolution probabilities. A node receives 'RW' marked samples in time. Thus the error is small for large enough. If the contents are static, one may discontinue the estimation process after large enough time, in which case the time-averaged performance of the policy remains unchanged.
- Alternatively, to allow persistent tracking of changes in resolution probabilities, we may estimate the query resolution probabilities via samples provided from a control algorithm, without using a separate unbiased random walk. The convergence of estimation and stability of the system can be jointly obtained via stochastic approximation framework under time scale

separation between content dynamics and search dynamics.

Reducing Complexity

- ❖ In this module, we develop the Reducing the complexities in the system. Not unlike standard backpressure-based routing our policies suffer from a major drawback: each node needs to share the state of its potentially large number of non-empty queues with its neighbors. For backpressure-based routing the number of queues per node corresponds to the number of flows (commodities) in the network. In our context, the number of queues per node corresponds to number of query types it could see worst case.
- ❖ In this module we propose simple modification and approximations that considerably reduce the overheads, albeit with some penalty in the performance. The key idea is to define equivalence classes of query types that share a ‘similar’ history, in the sense that they have similar conditional probabilities of resolution, and have them share a queue. For example, all query types of class which have visited the same number of nodes might be grouped together, reducing the number of queues to or better. Alternatively we will show one can further reduce overheads by approximately grouping similar query types based on their classes and the cumulative number of class files/resources they have seen in nodes, reducing the number of queues to where is a set of quantization levels. Intuitively such queries have seen similar opportunities if files/resources are randomely made available in the network.

IMPLEMENTATION MODULES:

- ❖ Constructing System Model
- ❖ Stable Query Forwarding Policy
- ❖ Estimating Query Resolution Probabilities
- ❖ Reducing Complexity

MODULES DESCSRIPTION:

Constructing System Model

- ❖ In the initial module, we develop the system with the entities required to show the proposed model with the evaluation proof of your novel contribution. So we develop the module with nodes. We assume that time is slotted, and each super-peer has an associated service rate, corresponding to positive integer number of queries it is willing to resolve/forward in each slot.
- ❖ We assume that super peers keep a record of files/resources available at subordinate peer. This information is communicated to super peers when a subordinate peer joins a super peer. Subordinate peers may initiate a query request at a super peer, but do not participate in forwarding or query resolution
- ❖ If a class query at node cannot be resolved it may be forwarded to one of its neighbors. The likelihood a node can resolve such a query depends not only on its class but also its*history*, i.e., the set of nodes it visited in the past. Note that the history is not ordered. For example, suppose 3 nodes in a network partition files/resources associated with class. If two of these nodes attempted and failed in resolving a given class query then it will for sure be resolved at the third node. In other contexts, if a search for a particular media file failed at many nodes, it is more likely that the file is rare, and the conditional likelihood that it is resolved at the next node might lower.

Stable Query Forwarding Policy

- ❖ In this module, we will propose a query scheduling and forwarding policy that ensures the GoS for each class, is distributed, easy to implement, and is stable. We begin by defining the stability for such networks and the associated capacity region. The module is develop such that the following aspects arising in P2P search systems: (a) history dependent probability of query resolution at each node, (b) updates in 'types' of queries as they get forwarded to different nodes, (c) computing the quality of service received by query via its history and designing an appropriate exit strategy upon receiving enough service.

❖ While the routing decisions are to be based on instantaneous queue loads at the neighbors, the decisions themselves affect the type/queue to which a query belongs. In this module, we develop a distributed dynamic algorithm where each node makes decisions based on its queue states and that of its neighbors and only needs to know

Estimating Query Resolution Probabilities

❖ In this module, we develop the estimation of query resolution probabilities. So far we have assumed that resolution probabilities for queries of different types are known. In practice they can be easily estimated. In order to ensure unbiased estimates can be obtained at each node, suppose a small fraction of all queries is marked 'RW', forwarded via the random walk policy with a large TTL, and given scheduling priority over other queries.

❖ With a sufficiently large TTL this ensures that each node will see a random sample of all query and types it could see and thus allow for unbiased estimates. All queries which are not marked 'RW' are treated according to our backpressure policy based on the estimated query resolution probabilities. A node receives 'RW' marked samples in time. Thus the error is small for large enough. If the contents are static, one may discontinue the estimation process after large enough time, in which case the time-averaged performance of the policy remains unchanged.

❖ Alternatively, to allow persistent tracking of changes in resolution probabilities, we may estimate the query resolution probabilities via samples provided from a control algorithm, without using a separate unbiased random walk. The convergence of estimation and stability of the system can be jointly obtained via stochastic approximation framework under time scale separation between content dynamics and search dynamics.

Reducing Complexity

❖ In this module, we develop the Reducing the complexities in the system. Not unlike standard backpressure-based routing our policies suffer from a major drawback: each node needs to share the state of its potentially large number of non-empty queues with its neighbors. For backpressure-based routing the number of queues per node corresponds to the number of flows (commodities) in the network. In our context, the number of queues per node corresponds to number of query types it could see worst case.

❖ In this module we propose simple modification and approximations that considerably reduce the overheads, albeit with some penalty in the performance. The key idea is to define equivalence classes of query types that share a 'similar' history, in the sense that they have similar conditional probabilities of resolution, and have them share a queue. For example, all query types of class which have visited the same number of nodes might be grouped together, reducing the number of queues to or better. Alternatively we will show one can further reduce overheads by approximately grouping similar query types based on their classes and the cumulative number of class files/resources they have seen in nodes, reducing the number of queues to where is a set of quantization levels. Intuitively such queries have seen similar opportunities if files/resources are randomely made available in the network.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

INPUT DESIGN AND OUTPUT DESIGN

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2.It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

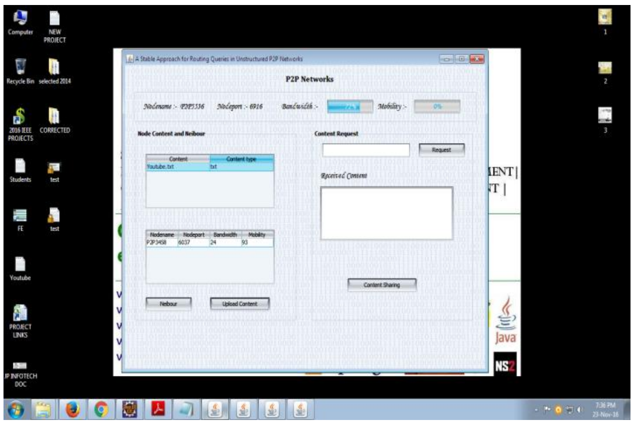
2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

SCREEN SHOTS


A Stable Approach for Routing Queries in Unstructured P2P Networks

P2P Networks

Nodename :- P2P5536 Nodeport :- 6916 Bandwidth :- Mobility :- 0%

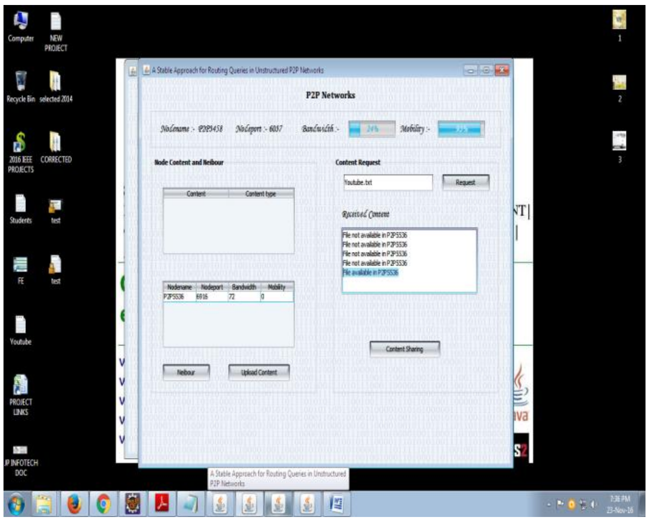
Node Content and Neibour

Content	Content type
Youtube.txt	txt

Nodename	Nodeport	Bandwidth	Mobility
P2P3458	6037	24	93

Content Request

Received Content



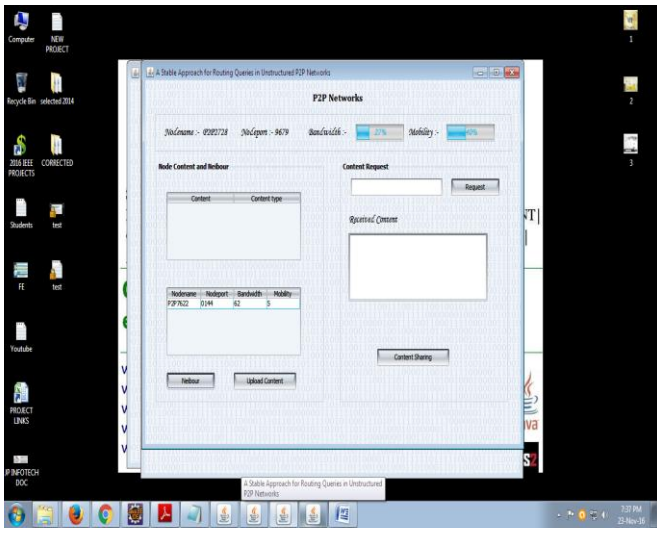
Nodename :- P2P5458 Nodeport :- 6057

Content Request: Youtube.txt

Content	Content type

Nodename	Nodeport	Bandwidth	Mobility
P2P5536	6916	72	0

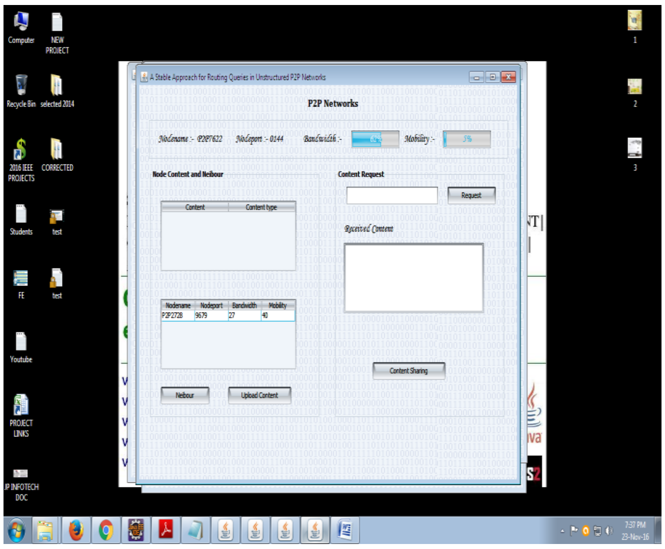
File not available in P2P5536
File not available in P2P5536
File not available in P2P5536
File not available in P2P5536
File available in P2P5536



Nodename :- P2P2728 Nodeport :- 9679

Content	Content type

Nodename	Nodeport	Bandwidth	Mobility
P2P7622	0144	62	5



Nodename :- P2P7622 Nodeport :- 0144

Content	Content type

Nodename	Nodeport	Bandwidth	Mobility
P2P2728	9679	27	40

CONCLUSION

To summarize, we provided a novel, distributed, and reliable search policy for unstructured peer-to-peer networks with super-peers. Our backpressure based policy can provide capacity gains of as large as 68% over traditional random walk techniques. We also provided modifications to the algorithm that make it amenable to implementation.

REFERENCES

[1] Wikipedia, "Peer-to-peer," 2011 [Online]. Available: http://en.wikipedia.org/wiki/Peer-to-peer

[2] I. Stoicaet al., "Chord: A scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Trans. Netw., vol. 11, no. 1, pp. 17–32, Feb. 2003.

[3] X. Li and J. Wu, "Searching techniques in peer-to-peer networks," in Handbook of Theoretical and Algorithmic Aspects of Ad Hoc, Sensor,Peer-to-Peer Networks. Boca Raton, FL, USA: CRC Press, 2004.

[4] C. Gkantsidis, M. Mihail, and A. Saberi, "Random walks in peer-topeer networks," in Proc. IEEE INFOCOM, 2004, pp. 120–130.

[5] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer to peer networks," in Proc. IEEE INFOCOM, 2005, pp. 1526–1537.

[6] S. Ioannidis and P. Marbach, "On the design of hybrid peer-to-peer systems," in Proc. ACM SIGMETRICS, Annapolis, MD, USA, Jun. 2008, pp. 157–168.

[7] P. Patankar, G. Nam, G. Kesidis, T. Konstantopoulos, and C. Das, "Peer-to-peer unstructured anycasting using correlated swarms," in Proc. ITC, Paris, France, Sep. 2009, pp. 1–8.

[8] R. Gupta and A. Somani, "An incentive driven lookup protocol for chord-based peer-to-peer (P2P)

networks," in Proc. Int. Conf. High Perform.Comput., Bangalore, India, Dec. 2004, pp. 8–18.

[9] D. Menasche, L. Massoulie, and D. Towsley, "Reciprocity and barter in peer-to-peer systems," in Proc. IEEE INFOCOM, 2010, pp. 1–9.

[10] B. Mitra, A. K. Dubey, S. Ghose, and N. Ganguly, "How do superpeer networks emerge?," in Proc. IEEE INFOCOM, 2010, pp. 1–9.

[11] D. Karger and M. Ruhl, "Simple efficient load balancing algorithms for peer-to-peer systems," in Proc. 16th ACMSPAA, 2004, pp. 36–43.

[12] B. Yang and H. Garcia-Molina, "Designing a super-peer network," in Proc. IEEE ICDE, 2003, pp. 49–60.

[13] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," IEEE Trans. Autom. Control, vol. 37, no. 12, pp. 1936–1948, Dec. 1992.

[14] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time varying wireless networks," in Proc. IEEE INFOCOM, 2003, pp. 745–755.

[15] L. Ying, S. Shakkottai, A. Reddy, and S. Liu, "On combining shortest-path and back-pressure routing over multihop wireless networks," IEEE/ACM Trans. Netw., vol. 19, no. 3, pp. 841–854, Jun. 2011.

[16] M. Alresaini, M. Sathiamoorthy, B. Krishnamachari, and M. Neely, "Backpressure with adaptive redundancy (BWAR)," in Proc. IEEE INFOCOM,
Mar. 2012, pp. 2300–2308.

[17] L. Bui, R. Srikant, and A. Stolyar, "A novel architecture for reduction of delay and queueing structure complexity in the back-pressure algorithm," IEEE/ACM Trans. Netw., vol. 19, no. 6, pp. 1597–1609, Dec. 2011.

[18] B. Ji, C. Joo, and N. Shroff, "Delay-based back-pressure scheduling in multihop wireless networks," IEEE/ACM Trans. Netw., vol. 21, no. 5, pp. 1539–1552, Oct. 2013.

[19] Y. Cui, E. Yeh, and R. Liu, "Enhancing the delay performance of dynamic backpressure algorithms," IEEE/ACM Trans. Netw., 2015, to be published.

[20] L. Georgiadis, M. J. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," Found. Trends Netw., vol. 1, no. 1, pp. 1–144, 2006.

[21] Y. Cui, V. Lau, R. Wang, H. Huang, and S. Zhang, "A survey on delayaware resource control for wireless systems—Large deviation theory, stochastic Lyapunov drift, distributed stochastic learning," IEEE Trans. Inf. Theory, vol. 58, no. 3, pp. 1677–1701, Mar. 2012.

[22] S. Asmussen, Applied Probability and Queues. NewYork,NY,USA: Springer, 1987.

[23] H. J. Kushner and G. Yin, Stochastic Approximation and Recursive Algorithms and Applications. New York, NY, USA: Springer, 2003.