DIMACS Technical Report 2008-06
November 2008

The Impact of Communication Models on Routing-Algorithm Convergence

(Version of 2008–11–24)

by

Aaron D. Jaggard[1,2]
DIMACS
Rutgers University
`adj@dimacs.rutgers.edu`

Vijay Ramachandran[3,4]
Department of Computer Science
Colgate University
`vijayr@cs.colgate.edu`

Rebecca N. Wright[5,6]
DIMACS and Department of Computer Science
Rutgers University
`rebecca.wright@rutgers.edu`

[1]DIMACS Assistant Research Professor
[2]Partially supported by NSF awards CNS–0751674, CNS–0753492, and DMS–0239996, and by ONR award N00014–05–10818. Work done in part while at the Dept. of Mathematics, Tulane University.
[3]Work done in part while visiting DIMACS for the Special Focus on Algorithmic Foundations of the Internet and in part while a DIMACS postdoc at the Stevens Institute of Technology.
[4]Partially supported by NSF award CNS–0753061 and the Stevens Technogenesis Program.
[5]DIMACS Deputy Director
[6]Partially supported by NSF award CNS–0753061.

ABSTRACT

Distributed autonomous routing algorithms such as BGP, AODV, and others are intended to reach a consistent, global solution after nodes iteratively and independently collect, process, and share information. However, the important role of the mechanism used to share information has generally been overlooked in previous analyses of these algorithms. In this paper, we explicitly study how the network-communication model affects algorithm convergence. To do this, we consider a variety of factors, including channel reliability, how much information is processed from channels, and how many channels are processed simultaneously. Using these factors, we formally define a taxonomy of communication models and identify particular models of interest, including those used in previous theoretical work, those that most closely model real-world implementations of BGP, and those of potential interest for the design of future routing algorithms. We then analyze how the algorithm convergence properties of different models in our taxonomy are related.

We show that algorithm convergence depends on the communication model in nontrivial ways. For some pairs of models, any execution of the routing algorithm in one model can be realized as an execution of the same algorithm in the other models. Conversely, we show by example that some of the possible executions in some models cannot be realized in other models; importantly, there are network instances for which the algorithm always converges in one model but need not converge in another model. Here we give an extensive description of these types of relationships among the models in our taxonomy. These results are important for studying the convergence of distributed autonomous routing protocols because certain communication models are best for proving model-independent conditions that guarantee convergence, while other models are best for proving model-independent conditions that might permit nonconvergence.

Contents

1 Introduction

Distributed autonomous routing algorithms (*e.g.*, BGP [16], RIP [14], AODV [15], and PVEX [11]) are used to establish network connectivity when coordination among network entities is low. Much work has been done to analyze the behavior of such protocols, including giving examples of possible protocol divergence and proving sufficient conditions for protocol convergence. This work has used a variety of models for the communication between protocol participants; however, the possible impact of the communication model on the theoretical results has generally been neglected, so it was previously unknown even whether this is a significant factor. Here we address this issue comprehensively by defining a taxonomy of communication models for routing protocols and showing how the different models can affect convergence. In addition to showing that theoretical convergence results *do* depend on the choice of communication model, we consider the possible pairs of communication models from our taxonomy and prove an extensive set of results that show when protocol convergence or divergence results proved in one of the models necessarily hold in the other. In the process, we thoroughly explore the design space defined by our taxonomy; different points in this space correspond to different models used in previous research and also to different configuration settings in real-world networks. Our work identifies previously unstudied models that are important because they appear to better capture the flexibility of actual protocols and because of their strong ability to simulate algorithm behavior in other models.

Distributed autonomous routing algorithms iteratively compute path assignments; they promote a consistent view of the network without overwhelming it with unnecessary information. The essential steps in such an algorithm may be abstractly captured as the following actions, repeatedly performed by each node:

1. Collect updates of path information describing feasible paths to a destination node d;
2. Choose the most preferred path (according to local policy) from the paths known at the node; and
3. Announce any changes in path choice to neighboring nodes.

However, these actions do not determine aspects of the communication model that can have an impact on an algorithm's outcome.

Network operators and protocol designers hope that when a distributed autonomous routing protocol is run, it will eventually converge to a stable and consistent path assignment. However, previous work ([18], followed by a line of work analyzing BGP convergence, including [4, 6, 8, 9, 17]) has shown that divergence, which necessarily involves routing oscillations, is possible. Different work in this area has made different assumptions about how nodes execute actions (1)–(3) above. These differences can be realized in practice (*e.g.*, the BGP specification [16] allows flexibility in how actions (1) and (3) are carried out). As we discuss below, we may view these differences as assuming different properties of the communication channels between nodes.

Here we consider these differences as well as reliability of message delivery, which is important because routing algorithms may be run on networks that do not guarantee reliable message delivery (*e.g.*, in many wireless networks, or if TCP is not used); in such cases, communications between nodes running the protocol can be arbitrarily delayed or even lost. We define and explore the space of communication models using three dimensions, crucial to Internet routing, that correspond to answers to the following questions:

- Are updates ever lost?
- From how many neighbors are updates collected in action (1) above?
- How many announcements from each neighbor are collected in action (1) above? (As described below, given the answer to the first two questions, a communication channel between two nodes may contain more than one update.)

The space defined by these dimensions provides a taxonomy that includes points that systematically capture various combinations assumptions about synchrony, atomicity, and network delays; these map to realistic differences in today's network infrastructure. The space also includes both reliable and unreliable channels. To our knowledge, this is the first work to consider the algorithmic properties of interdomain routing over unreliable channels.

Considering these various communication models, we are particularly interested in whether a routing algorithm that is guaranteed to converge in a network (perhaps satisfying certain conditions) using one model is also guaranteed to converge if that network uses a different communication model. To this end, we formally define different notions of realizing an execution of one model in another and prove an extensive set of realization relationships between different pairs of models. We note that our focus is on the *communication* of updates and not on assumptions about the *computation* models assumed within nodes; in particular, the choice of communication model only affects actions (1) and (3) above. We show that for many pairs of the models in our taxonomy, all executions (in particular, all divergent executions) in one model can be realized in some form in the other; however, we also show that there are some pairs of models for which this is not true. We are also able to identify aspects of the communication models that do and do not affect convergence. This work enriches the important line of work understanding the convergence behavior of BGP and other networking protocols, *e.g.*, [3–6, 8–10, 17].

Our work here guides the future analysis of routing protocols by demonstrating which models are appropriate for which types of analysis results. Our results imply that there are "weak" models that are useful for exhibiting oscillations (so that divergence is then possible in all other models as well) and "strong" models that are useful for proving guarantees of convergence. We demonstrate that previous analyses of BGP in particular, while not explicitly considering the effects of communication models, satisfy our guidelines to ensure applicability across communication models. Our results also contribute to protocol analysis by demonstrating that some types of divergence are not possible in some models and by identifying important models in our taxonomy that are realistic and powerful but that have not been considered before. Finally, our results imply that, in this framework, reliable channels offer little benefit over unreliable channels for the purpose of guaranteeing convergence. However, always having access to the current network state (instead of sequentially processing update messages from a queue) can help guarantee convergence.

To summarize, our main contributions are as follows:

- We define a taxonomy of communication models (Sec. 2.2);
- We identify particular points of interest in our taxonomy, including "polling," "message-passing," and "queueing" models (Sec. 2.3); and
- We prove an extensive set of relationships between the different models in our taxonomy (Sec. 3).

Of particular interest is the fact that convergence *does* depend on the choice of communication model. Before presenting our contributions, we provide some background and technical preliminaries in Sec. 2.1. We discuss related work, including the relationship of our work to classical results about modeling in the distributed computing literature, in Sec. 4.

2 Communication Models

We start in Sec. 2.1 with a description of the routing problem and the general distributed autonomous routing algorithm. We introduce the dimensions of the model space in Sec. 2.2 and highlight some specific models of interest in Sec. 2.3.

2.1 Problem basics and algorithm

We focus on using distributed autonomous routing algorithms to solve the Stable Paths Problem (SPP) [9], an abstract representation of the interdomain-routing problem. An instance of SPP contains an undirected graph $G = (V, E)$ with a distinguished *destination node* d and, for each node $v \in V$, a set of *permitted paths* $\mathcal{P}_v$, which is a subset of all simple paths from v to d, and a ranking function $\lambda_v : \mathcal{P}_v \to \mathbb{N}$ indicating v's preference for each permitted path. (Assume that, like cost, paths with lower rank are more preferred.) Ties in ranking are not permitted except when two paths go through the same neighbor. The problem is to find a *path assignment* $\pi = \{\pi_v\}_{v \in V}$ that, for each $v \neq d$, is (1) *consistent*—assuming that the next hop along π_v is u, we have that $\pi_v = v\pi_u$ (if v extends a path from u to d, then that path from u to d is assigned to u)—and (2) *stable*—for all neighbors $w \neq u$ of v, $\lambda_v(v\pi_u) < \lambda_v(v\pi_w)$, *i.e.*, π_v is more preferred than any other $v\pi_w$. We assume that $\pi_d = d$.

Let $\mathcal{C}$ be the set of communication channels that might be used; for each edge $\{u, v\}$ in the undirected instance graph, $\mathcal{C}$ contains directed channels (u, v) and (v, u). We assume that each channel is FIFO, so that messages that are written by u (and no other party) to the channel (u, v) and are not dropped by the channel are processed by v in the same order in which they were written. We also assume that a single communication model is used throughout the network.

Definition 2.1 (Components of network state)**.** We keep track of various components of the network state, although for relationships between models we focus on the path assignments in particular. Each aspect of state depends on the step t of the algorithm's execution. The components we consider are:

Path assignments. $\pi_v(t)$ is the path to the destination that v chooses at the end of step t; we let $\pi(t) = \{\pi_v(t)\}_{v \in V}$ be the collection of all path assignments. Note that $\pi_v(t+1) = \pi_v(t)$ unless v runs the update algorithm in step $t+1$. We let $\pi_v(0) = \epsilon$ for $v \neq d$ and $\pi_d(0) = d$.
Known routes. After the execution of the first part of the algorithm in step t, $\rho_v(c;t)$ contains the contents of the last update that v successfully processed from the channel c. $\rho_v(c;t+1) = \rho_v(c;t)$ unless v updates from channel c in step $t+1$. We let $\rho_v(c;0) = \epsilon$ for every $v \in V$ and $c \in \mathcal{C}$.
Channel contents. For a channel $c = (u,v)$, we let $c(t)$ be the contents of c at the beginning of step t; let $m_c(t)$ be the number of messages in c at the beginning of step t. We will use $c_i(t)$ to denote the i^{th} message in c at the beginning of step t, where the first message is the oldest. We let $c(0) = \emptyset$ for every $c \in \mathcal{C}$.

In each iterative step of the algorithm, nodes (1) collect information from channels, (2) choose route objects based on their ranking function and permitted paths, and (3) share information by writing route objects to channels. The communication models we study here affect only the first of these three actions. We use an *activation sequence* to specify the nodes involved in each round and how the first action is executed; an activation sequence determines the algorithm's execution. We now define the most general notion of activation sequence; the different models we consider can be viewed as different restricted classes of activation sequences.

Definition 2.2. An activation sequence α is a function on the non-negative integers that assigns to each $t \in \mathbb{Z}_{\geq 0}$ a quadruple (U, X, f, g) such that:

- $U \subseteq V$ is the set of vertices that will update in step t;
- $X \subseteq \mathcal{C}$ is the set of channels that will be updated in step t. For each $c = (u,v) \in \mathcal{C}$, we require that $v \in U$, *i.e.*, the receiving end of each channel is one of the nodes that is updating in step t;
- $f : X \to \mathbb{Z}_{\geq 0} \cup \{\infty\}$ indicates how many messages from each channel should be processed. For $c = (u,v) \in X$, v will process $f(c)$ messages from c (if $f(c) = \infty$, then v will process all messages in the channel); and
- $g : X \to \mathcal{P}(\mathbb{Z}_{>0})$ indicates which, if any, messages will be dropped from each channel; the elements of $g(c)$ are the indices of the messages that will be dropped so if, *e.g.*, $2 \in g(c)$, then the second message in c will be dropped. We require that if $f(c) = 0$, then $g(c) = \emptyset$, and if $0 < f(c) < \infty$, then $g(c) \subseteq \{1, 2, \ldots, f(c)\}$.

Definition 2.3 (Algorithm execution using a general activation sequence)**.** Given a network instance and an activation sequence α, the iterative routing algorithm executes as follows, starting with $t = 0$.

1. Let $(U, X, f, g) = \alpha(t)$.
2. For each $v \in U$ and $u \in \mathcal{N}(v)$ such that $(u, v) \in X$
   (a) Let $c = (u, v)$
   (b) If $f(c) = \infty$, let $i = m_c(t)$; if $f(c) < \infty$, let $i = \max\{f(c), m_c(t)\}$.
   (c) If $\{1, 2, \ldots, i\} \setminus g(c) \neq \emptyset$, let j be the largest element of this set, and let $\rho_v(c; t)$ be the route in the j^{th} message in c. If $\{1, 2, \ldots, i\} \setminus g(c) = \emptyset$, let $\rho_v(c; t) = \rho_v(c; t-1)$.
   (d) Delete the first i messages from c, and set $m_c(t+1) = m_c(t) - i$.
3. For each $v \in U$, set $\pi_v(t)$ to be the most preferred path from the set $\{\rho_v((u, v); t) \cap \mathcal{P}_v \mid u \in \mathcal{N}(v)\}$ if $v \neq d$, and let $\pi_v(t) = d$ otherwise.
4. For each $v \in U$ and $u \in \mathcal{N}(v)$, if $\pi_v(t) \neq \pi_v(t-1)$ and if prescribed by export policy, write the path $\pi_v(t)$ to the channel (v, u) and increase $m_{(v,u)}(t+1)$ by one.
5. Increment t and repeat from Step 1.

Definition 2.4 (Fair activation sequence)**.** We say that an activation sequence α is *fair* if every node tries to read each of its channels infinitely often and, if a message is dropped from channel c (a possibility when unreliable channels are used), then there is a later message on c that is not dropped.

We will restrict our attention to fair activation sequences.

Definition 2.5 (Convergence of an activation sequence)**.** An activation sequence α *converges to the path assignment* π if, for the induced sequence of path assignments $\pi(t)$, there exists some t^* such that for any $t' > t^*$, $\pi(t') = \pi$. We write $\lim_{t\to\infty} \pi(t) = \pi$, and say α *converges* if the limit exists.

Various work—*e.g.*, [6, 9]—has studied restrictions that can be placed on networks to ensure that *every* fair activation sequence converges. As we relate different models below, we will focus on sequences of path assignments $\{\pi(t)\}_t$, which are present in every model, instead of other aspects of network state.

2.2 Dimensions of the model space

The quadruples that appear in a general activation sequence suggest four dimensions that might be considered in studying different models of communication: the number of nodes updating per step, the number of neighbors processed at each step, the number of messages processed per channel, and channel reliability.

Definition 2.6 (Dimensions of the model space)**.** The four dimensions we study are:

Number of nodes updating An activation sequence must specify the set of nodes that update at each step (the *activated* nodes). *In the remainder of this paper, we require exactly one node to update at each step.*[1] However, various possible values are:

Every Every node updates at every step, *i.e.*, $U = V$;

Unrestricted There are no restrictions on which nodes do or do not update, although to avoid trivial steps we assume $U \neq \emptyset$; and

One Exactly one node updates at each step, *i.e.*, $|U| = 1$.

Number of neighbors processed Each model specifies how many channels a node should process when it updates. The possible values are:

E **(Every)** Whenever a node updates, it processes messages from every one of its neighbors, *i.e.*, $X_v = \mathcal{N}(v)$ for every $v \in U$.

M **(Multiple)** Whenever a node updates, it processes messages from some subset of its neighbors (potentially multiple neighbors), including the possibility of processing *no* channels and that of processing *all* channels; this imposes no additional restriction on X_v.

1 Whenever a node updates, it processes messages from exactly one of its neighbors, *i.e.*, $|X_v| = 1$ for every $v \in U$.

Number of messages processed per channel Each model specifies how many messages a node should read from a channel when it processes that channel.

A **(All)** Whenever a node v updates and is assigned to process messages from a neighbor u, v processes all of the messages in this channel, *i.e.*, $f_v \equiv \infty$.

S **(Some)** There are no restrictions on the number of messages that a node processes from each of its neighbors when it updates.

F **(Forced)** Each node is forced to process at least one message from each of the neighbors from which it updates (although this set may omit some neighbors), but it may process multiple messages from each neighbor.

O **(One)** Whenever a node v updates and is assigned to process messages from a neighbor u, it processes exactly one message from this channel, *i.e.*, $f_v \equiv 1$. If unreliable channels are used, this message may be lost.

Channel reliability Channels are either reliable or unreliable in the following senses.

R **(Reliable)** Every message placed in a channel (u, v) by u is always read by v, *i.e.*, the functions g_v in the fourth component of an activation sequence entry are always identically equal to $\emptyset$.

[1]However, see Ex. A.6 in App. A for some initial results about multiple-node updates.

U **(Unreliable)** Some messages placed in channels are not read, *i.e.*, the functions g_v need not be identically equal to $\emptyset$.

The symbols for each option in the last three dimensions (E, M, *etc.*) are used to abbreviate the combinations; thus RMA is the model that uses reliable channels in which the node that updates in a given step processes an arbitrary set of its incoming channels and processes all messages from each of these channels.

2.3 Specific models

We now review some models that are of particular interest because of their previous use in research, their fidelity to the current BGP specification [16], or their potential for guiding the design of new routing protocols. (We note that although the BGP specification [16] assumes that BGP runs on TCP, thus ensuring reliable delivery, protocols like BGP are being designed for various purposes [10]. Our work permits analysis of convergence even when only unreliable channels are available.) As noted above, we restrict our attention to models in which exactly one node updates in every step, *i.e.*, $|U| = 1$ in every activation sequence quadruple.

2.3.1 Polling models

Informally, in "polling" models, nodes learn the current state of (some or all of) their neighbors in the first algorithm action when activated. The REA polling model was used in [3, 4]; as we show below, this allows the hardness results in that work to apply regardless of model used. The REA polling model has also been used in the application of mechanism design to routing (*e.g.*, [7, 12]), although the primary focus of that work is not algorithm convergence.

Because all messages in a channel are processed before the active node chooses its best route, the active node essentially ignores any intermediate messages and only uses each neighbors' most recent announcement. (We note that in adversarial or unfaithful settings, nodes may use the additional information from intermediate messages to modify their actions; otherwise, only the most recent information is used in route choice.)

Because of the constraints on activation sequences that define polling models, we may simply abbreviate the elements of the activation sequence as the updating node and the neighbors from which it updates. If REA is being used, we may further abbreviate this to just the node itself (writing, *e.g.*, $\alpha(t) = v$), because all channels are read.

Informally, we call R1A "poll one," RMA "poll some," and REA "poll all."

2.3.2 Message-passing models

"Message-passing" models were used in the original definition and proof of NP-completeness of SPP [9]. In these models, when a node is assigned to update, it reads (or possibly drops if unreliable channels are used) one message from each channel being read. As we show below, any nonconvergent algorithm execution in any of the other models we consider can also occur in the message-passing models R1O and RMO (but not necessarily in the model REO, in which a nodes essentially act on the first message in every nonempty channel simultaneously). The message-passing model R1O can be thought of as an "event-driven" model, in which nodes respond individually to each incoming update, or in which node activation is triggered by an announcement.

2.3.3 Queueing models

The "queueing" models RMS and UMS have not, to our knowledge, been used in earlier work on path-vector protocols. We believe these are of particular interest because the flexibility of configuration parameters in the BGP specification [16] suggest that they most naturally correspond to correct operation of BGP on the Internet. Furthermore, as we prove below, they are very strong in their ability to realize the other models in our taxonomy.

There is one queueing model for each option of channel reliability; each model then allows any number of channels to be processed in a single step, and any number of messages to be processed from each channel (so that X and f are unrestricted and either $g \equiv \emptyset$ or g is also unrestricted).

3 Realization Relationships between Communication Models

In this section, we analyze the algorithm executions that can occur in the various models generated by the dimension values outlined in the previous section. We begin by defining types of relationships between models and then go on to prove how an algorithm execution in one model might be seen as an "equivalent" execution in another.

One main result of this section is that most of the queueing and message-passing models capture at least as many protocol executions as any other model in the space. We also show that the models REF and REO cannot capture all the executions of other models, but they both capture all the executions possible in REF, REO, R1A, RMA, and REA. Finally, we show that the polling models capture even fewer executions than REF and REO.

3.1 Notions of realization

In studying the effects of communication models, we are essentially interested in the following relationship between models.

Definition 3.1 (Oscillation preservation)**.** We say that model B *preserves the oscillations of* another model A if the existence of a nonconvergent activation sequence α in A for some network instance I implies that there exists an activation sequence α' in B for network instance I that does not converge. We then write $A \trianglelefteq B$.

This definition formally allows a node to have a limiting path assignment in the first model but not in the second model, so long as some nodes oscillate.

While the oscillation-preserving relationship is weaker than most of the relationships that we actually prove, it captures the essential impact of model choice on convergence. Most of our results involve the following realization relations, which imply the preservation of oscillations. Saying that model A realizes model B (in some sense) might be thought of as asserting that for every execution of a system under model B, there is an execution of the system under model A such that a specific type of simulation relation may be defined from the first execution to the second.

Definition 3.2 (Execution realization)**.** We say that model B *realizes the executions of* model A (*exactly*, *exactly with repetition*, or *as subsequences*) if, for every network instance using B and activation sequence α (in B), there exists an activation sequence α' in the A such that, if $\{\pi(t)\}_t$ and $\{\pi'(t)\}_t$ are the path-assignment sequences induces by α and α', we have

Exact realization $\forall t$, $\pi'(t) = \pi(t)$, *i.e.*, the sequences are the same. We then write $A \leq B$.

Exact realization with repetition $\exists f : \mathbb{N} \to \mathbb{N}$ such that $\forall i, j$, $i < j \Rightarrow f(i) < f(j)$ and $f(t) \leq k < f(t+1) \Rightarrow \pi'(k) = \pi(t)$, *i.e.*, $\{\pi'(t)\}_t$ is obtained from $\{\pi(t)\}_t$ by replacing each $\pi(t)$ with one or more occurrences of $\pi(t)$. We then write $A \lesssim B$.

Realization as a subsequence $\exists f : \mathbb{N} \to \mathbb{N}$ such that $\forall i, j$, $i < j \Rightarrow f(i) < f(j)$ and $\forall t$, $\pi'(f(t)) = \pi(t)$, *i.e.*, $\{\pi(t)\}_t$ is a subsequence of $\{\pi'(t)\}_t$. We then write $A \preceq B$.

By examining these definitions, we see that exact realization implies exact realization with repetition, which implies realization as a subsequence. If one model realizes the executions of another, in any of these senses, it immediately follows that the first model preserves the oscillations of the second model.

3.2 Foundational positive results

Here we present our positive results, which show that certain models may be realized in various senses by other models. We build on these results by combining them with each other, our negative results in Sec. 3.3, and additional arguments to obtain a broad range of relationships as described in Sec. 3.5. We start with some general exact realizations.

Proposition 3.3. *For every* $w \in \{\mathsf{R}, \mathsf{U}\}$*,* $x \in \{\mathsf{1}, \mathsf{M}, \mathsf{E}\}$*, and* $y \in \{\mathsf{O}, \mathsf{S}, \mathsf{F}, \mathsf{A}\}$*:*

1. $\mathsf{U}xy$ *exactly realizes* $\mathsf{R}xy$*;*
2. $wx\mathsf{S}$ *exactly realizes* $wx\mathsf{F}$*;*
3. $wx\mathsf{F}$ *exactly realizes* $wx\mathsf{O}$ *and* $wx\mathsf{A}$*; and*
4. $w\mathsf{M}y$ *exactly realizes* $w\mathsf{1}y$ *and* $w\mathsf{E}y$*.*

Proof. This follows from the following straightforward observations:

1. Any activation sequence in $\mathsf{R}xy$ is also an activation sequence in $\mathsf{U}xy$.
2. Any activation sequence in $wx\mathsf{S}$ is also an activation sequence in $wx\mathsf{F}$.
3. Any activation sequence in $wx\mathsf{O}$ is also an activation sequence in $wx\mathsf{F}$. Any activation sequence in $wx\mathsf{A}$ is also an activation sequence in $wx\mathsf{F}$.
4. Any activation sequence in $w\mathsf{1}y$ is also an activation sequence in $w\mathsf{M}y$. Any activation sequence in $w\mathsf{E}y$ is also an activation sequence in $w\mathsf{M}y$.

$\square$

We may also prove the following exact realization.

Proposition 3.4. *For every w, $w\mathsf{ES}$ exactly realizes $w\mathsf{MS}$.*

Proof. Let α be a fair activation sequence in $w\mathsf{MS}$. For each $t \geq 0$, if $\alpha(t) = (\{v\}, X, f, g)$, we define $\alpha'(t) = (\{v\}, X', f', g')$ where $X' = \{(u,v) | u \in \mathcal{N}(v)\}$, $f'(c) = f(c)$ if $c \in X$ and $f'(c) = 0$ if $c \notin X$, and $g'(c) = g(c)$ if $c \in X$ and $g'(c) = \emptyset$ if $c \notin X$. Then α' is a fair activation sequence in $w\mathsf{ES}$ and the sequence of path assignments induced by α' equals that induced by α. $\square$

We have the following general result that involves realization with repetition; in light of Prop. 3.10, it cannot be strengthened, at this level of generality, to use exact realization.

Theorem 3.5. *For every $w \in \{\mathsf{R}, \mathsf{U}\}$, $y \in \{\mathsf{O}, \mathsf{S}, \mathsf{F}, \mathsf{A}\}$, $w\mathsf{1}y$ realizes $w\mathsf{M}y$ with repetition.*

Proof. For fixed $w \in \{\mathsf{R}, \mathsf{U}\}$ and $y \in \{\mathsf{O}, \mathsf{S}, \mathsf{F}, \mathsf{A}\}$, let α be a fair activation sequence in $w\mathsf{M}y$. For each element $(\{v\}, X, f, g)$ in $\{\alpha(t)\}$ we define a sequence of quadruples that are legal entries in activation sequences (for this network) in $w\mathsf{M}y$; replacing each element of $\{\alpha(t)\}$ by its corresponding sequence will produce a valid activation sequence in $w\mathsf{1}y$ that meets the claimed conditions. We assume that all channels in the $w\mathsf{1}y$ system contain exactly the messages, and in the same order, that appear in the $w\mathsf{M}y$ system; this is true before the system starts, and each sequence of steps in the $w\mathsf{1}y$ system (corresponding to a single step in the $w\mathsf{M}y$ system) will preserve this.

Let c be the channel from which v learns the path it selects after step t (in the $w\mathsf{M}y$ system) and let d be the channel from which it learns the path it selects after step $t-1$ (if $t > 0$). Order the channels in X as $c_1, \ldots, c_k$ so that, if $c \neq d$, $c_1 = c$ (if $c \in X$) and $c_k = d$ (if $d \in X$); if $c = d \in X$, then let $c_1 = c$ if the path chosen after step t is higher-ranked than the path chosen after step $t-1$, and let $c_k = d$ if the opposite is true. The requisite sequence of quadruples is $(\{v\}, \{c_1\}, f'_1, g'_1), \ldots, (\{v\}, \{c_k\}, f'_k, g'_k)$ where $f'_i(c_i) = f(c_i)$ and $g'_i(c_i) = g(c_i)$. Before proceeding through any of these steps, the $w\mathsf{1}y$ system is (by induction) using the path $P = \pi_v(t-1)$. After proceeding through these steps, the $w\mathsf{1}y$ system will be using the path $Q = \pi_v(t)$ because it will know the same set of routes that the $w\mathsf{M}y$ system does after step t. We claim that, if $P \neq Q$, then at some point in this sequence of steps, the new system switches from P to Q and that this is the only change in path assignment that it makes. If v did not already know Q before it starts this sequence of steps, it learns Q in the first step unless it prefers P to Q and it learns both P and Q from the same channel c. (In that case, all paths that v learns in the intermediate steps are less-preferred than Q, and thus P, which it knows about until the final step, when it learns—and switches to—P; this case thus satisfies the claim.) Any new paths that v learns in intermediate steps must be less-preferred than Q; if there are old paths that are more-preferred than Q, these are less-preferred than P and are thus never chosen (because P is then chosen until the last step). Thus the claim holds. □

We also have the following positive results. In view of Cor. 3.14, the one for reliable channels cannot be strengthened to realization with repetition.

Proposition 3.6. R1O *realizes* R1S *as a subsequence, and* U1O *realizes* U1S *with repetition.*

Proof. For the reliable channel case, we let α be a fair activation sequence in R1S. For each element $(\{v\}, \{c\}, f, g)$ in $\{\alpha(t)\}$ we define a sequence of quadruples that are legal entries in activation sequences (for this network) in R1O; replacing each element of $\{\alpha(t)\}$ by its corresponding sequence will produce a valid activation sequence in R1O that meets the claimed conditions. In the R1O system, we will informally "flag" some messages to indicate the ends of sequences of messages that are processed in a single update in the R1S system.

If $f(c) > 0$, let $k \geq 1$ be the number of messages in c, after the steps in the R1O system that correspond to steps $0, \ldots, t-1$ in the R1S system, up to and including the $f(c)^{\text{th}}$ "flagged" message. Replace $(\{v\}, \{c\}, f, g)$ with k copies of $(\{v\}, \{c\}, f', g')$, where $f'(c) = 1$ and $g'(c) = \emptyset$ as required by R1O. Finally, flag the update messages that v sends after processing the k^{th} of these updates. If $f(c) = 0$, then we delete $(\{v\}, \{c\}, f, g)$ and do not replace it with anything. The resulting sequence is a legal activation sequence in R1O. The R1O system may have various path assignments while processing the k update commands, but after processing the final one, the set of known paths at v (and at all other nodes, none of which have updated during this time) is the same as at v in the R1S system after step t, so the path assignments will be the same. As a result, we see that R1O realizes R1S as a subsequence.

For unreliable channels, given a fair activation sequence α in U1S, we replace $(\{v\}, \{c\}, f, g)$ in $\{\alpha(t)\}$ with nothing if $f(c) = 0$; but, if $f(c) = k > 0$, we replace $(\{v\}, \{c\}, f, g)$ with $(\{v\}, \{c\}, f'_1, g'_1), \ldots, (\{v\}, \{c\}, f'_k, g'_k)$, where $f'_i(c) = 1$ for every i and $g'_i(c) = \emptyset$ if i is the largest element of $\{1, \ldots, k\}$ that is not in $g(c)$ and $g'_i(c) = \{1\}$ otherwise (including if no such integer exists). This means that the only messages not dropped in the U1O system are exactly the ones that the U1S system actually uses (*i.e.*, that contribute known paths at the end of a step); thus the U1O system may repeat path assignments, but it does not transition through other path assignments. This shows that U1O realizes U1S with repetition. □

The following property is important in that it allows us to move from the unreliable case to the reliable case. The proof of this theorem relies on the fact that every dropped message is followed (eventually) by a non-dropped message in the definition of a fair activation sequence. Without this assumption, however, R1S does not even preserve the oscillations of U1O.

Theorem 3.7. R1S *exactly realizes* U1O.

Proof. Let α be a fair activation sequence in U1O; each element is of the form $(\{v\}, \{c\}, f, g)$, where $f(c) = 1$ and $g(c)$ is either $\emptyset$ or $\{1\}$. Construct α' by: replacing all elements $(\{v\}, \{c\}, f, g)$ for which $g(c) = \{1\}$ with $(\{v\}, \{c\}, f', g')$, where $f'(c) = 0$ and $g'(c) = \emptyset$, and replacing those elements for which $g(c) = \emptyset$ with $(\{v\}, \{c\}, f', g')$, where: $f'(c) \geq 1$ equals the number of elements $(\{v\}, \{c\}, f, g)$ in α (including the current one) since the previous such element with $g(c) = \emptyset$; and $g'(c) = \emptyset$. This is then a fair activation sequence in R1S, and the sequence of path assignments it induces is the same one induced by α, because in any channel there is always a non-dropped message after every dropped message. □

3.3 Foundational negative results

Our negative results show that certain models cannot be realized, in various senses, by other models. These are proved by exhibiting a network and an activation sequence from the first model that produces behavior that provably cannot be seen in the second model (either at all, to show that oscillations cannot be preserved, or without the addition of repeated or other states, to show that certain other realization relations do not hold). The examples referenced in these proofs are in App. A.

Theorem 3.8. *The oscillations of* R1O *are not preserved by* REO*,* REF*,* R1A*,* RMA*, and* REA*.*

Proof. Example A.1 presents a network that may oscillate in R1O but that cannot be made to oscillate in REO, REF, R1A, RMA, and REA. □

Theorem 3.9. *The oscillations of* REO *and* REF *are not preserved by* R1A*,* RMA*, and* REA*.*

Proof. Example A.2 presents a network that may oscillate in REO and REF but that cannot be made to oscillate in R1A, RMA, or REA. □

Proposition 3.10. REO *cannot be exactly realized in* R1O*.*

Proof. Example A.3 presents network and path assignment sequence induced by an activation sequence in REO that cannot be induced by any activation sequence in R1O. □

Proposition 3.11. REA *cannot be realized with repetition in* R1O*.*

Proof. Example A.4 presents a network and path assignment sequence induced by an activation sequence in REA such that no expansion of it obtained by replacing each assignment by one or more copies of the assignment is a path assignment sequence induced by an activation sequence in R1O. □

Proposition 3.12. REA *cannot be exactly realized by* R1S*.*

Proof. Example A.5 presents a network and path assignment sequence induced by an activation sequence in REA that cannot be induced by an activation sequence in R1S. □

Proposition 3.13. REO *cannot be exactly realized by* R1S.

Proof. The REA activation sequence in Example A.5 (used in Prop. 3.12) is also an REO activation sequence. □

3.4 Remarks on transitivity

As noted in Sec. 3.1, exact realization is stronger than realization with repetition, which in turn is stronger than realization as a subsequence, which in turn is stronger than the preservation of oscillations. If a model M_2 realizes a model M_1 in sense R_1, and M_3 realizes model M_2 in sense R_2, then we have that M_3 must also realize M_1 in the weaker of the senses R_1 and R_2 (although it might also realize M_1 in a stronger sense). This is illustrated in Fig. 1, which (along with Fig. 2) helps depict how to reason about realization relationships.

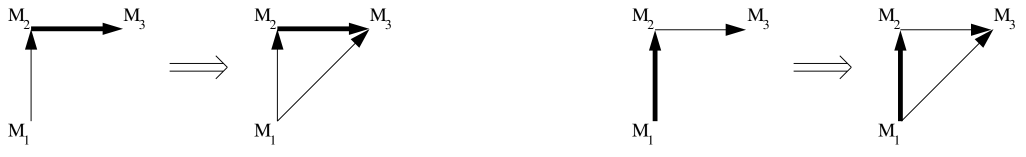

Figure 1: Graphical representation of positive transitivity rules. In both cases, model M_2 realized model M_1 and model M_3 realizes model M_2 (indicated to the left of the implication double arrow). As a result, we know that M_3 realizes M_1 the weaker of the previously known realization senses. Here, the thicker arrow suggests stronger a (weakly) stronger realization relationship. Informally, (left) we may "push" the head of a realization arrow (going from the realized model to the realizing model) forward along a weakly stronger realization arrow, and (right) we may "pull" the tail of a realization arrow backward along a weakly stronger realization arrow.

Conversely, if M_2 realizes M_1 in sense R_1 but M_3 cannot realize M_1 in sense R_2, and if R_1 is at least as strong as R_2, then M_3 also cannot realize M_2 in sense R_2. (Otherwise, the composition of the realization of M_1 in M_2 and the realization of M_2 in M_3 would produce a realization of M_1 in M_3 in sense R_2.) Similarly, if M_3 realizes M_1 in sense R_1 but M_3 cannot realize M_2 in sense R_2, and if R_1 is at least as strong as R_2, then M_1 also cannot realize M_2 in sense R_2. (Again, transitivity would lead to a realization of M_2 in M_3 in sense R_2.) These rules are illustrated in Fig. 2.

We may apply these arguments to the results above to obtain relationships between the models presented here. Section 3.5 presents all of these corollaries in a condensed form. As a detailed example, we may start with Prop. 3.11 and obtain the following collection of results.

Corollary 3.14. *For every y, y', z with $z \neq \mathsf{O}$, $\mathsf{R}yz$ cannot be realized with repetition in $\mathsf{R}y'\mathsf{O}$.*

Proof. By Thm. 3.5, R1O realizes RMO (and thus, by Prop. 3.3, REO) with repetition. As this is at least as strong as the sense of non-realization in Prop. 3.11, for every $y' \in \{\mathsf{1}, \mathsf{M}, \mathsf{E}\}$,

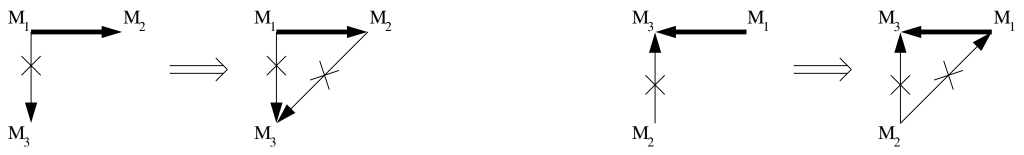

Figure 2: Graphical representation of negative transitivity rules as described in the text. The crossed-out arrows represent realization relationships that do not hold, and the thicker arrow a weakly stronger realization relationship that does hold. Informally, (left) we may "push" the tail of a non-realization arrow forward along a weakly stronger realization arrow, and (right) we may "pull" the head of a non-realization arrow backward along a weakly stronger realization arrow.

Ry'O cannot realize REA with repetition. Repeated applications of Prop. 3.3 show that for $y \in \{\mathsf{M}, \mathsf{E}\}$ and $z \in \{\mathsf{S}, \mathsf{F}, \mathsf{A}\}$, R$yz$ realizes REA exactly; by Thm. 3.5, we then have that R1z realizes REA with repetition for $z \in \{\mathsf{S}, \mathsf{F}, \mathsf{A}\}$. The arguments above and the first observation imply that for every $y, y' \in \{1, \mathsf{M}, \mathsf{E}\}$ and $z \in \{\mathsf{S}, \mathsf{F}, \mathsf{A}\}$, R$y'$O cannot realize R$yz$ with repetition. □

3.5 Comprehensive listing of realization results

Figures 3 and 4 summarize the results stated in Sec. 3.2–3.3 along with various corollaries obtained from these by the transitivity arguments above.

It is important to note how strong we have shown the queueing models to be. RMS is able to realize all reliable channel models exactly and all unreliable channel models either with repetition or exactly. UMS is able to exactly realize all models, reliable and unreliable. Furthermore, among the reliable channel models, R1O, RMO, R1S, RMS, RES, R1F, and RMF are all able to capture all of the oscillations of *all* other models in our taxonomy. In contrast, REO, REF, R1A, RMA, and REA are provably unable to capture some oscillations that may occur when using other models, so conditions that guarantee convergence in these models may not do so in general.

4 Related Work

Many general results in distributed computing [13] consider assumptions about network communication, including atomicity and synchrony. In contrast to those general results, we do not attempt to simulate an algorithm in one model by a modified algorithm in another; instead, we are interested in properties of the executions of the *same* algorithm in a variety of models. Rather than general simulation, we define the more specialized notion of a realization relationship (Sec. 3) and apply it to routing-algorithm convergence. Recently, Chambart and Schnoebelen [1] studied the impact of channel reliability on the ability of an algorithm execution to transition between two states; their results do not directly apply to our context because state transitions do not preclude the possibility of nonconvergence.

	R1O	RMO	REO	R1S	RMS	RES	R1F	RMF	REF	R1A	RMA	REA
R1O	–	4	-1	4	4	4	4	4	-1	-1	-1	-1
RMO	3	–	-1	3	4	4	3	4	-1	-1	-1	-1
REO	3	4	–	3	4	4	3	4	4	-1	-1	-1
R1S	2	2	-1	–	4	4	$\geq$2	$\geq$2	-1	-1	-1	-1
RMS	2	2	-1	3	–	4	2,3	$\geq$2	-1	-1	-1	-1
RES	2	2	-1	3	4	–	2,3	$\geq$2	-1	-1	-1	-1
R1F	2	2	-1	4	4	4	–	4	-1	-1	-1	-1
RMF	2	2	-1	3	4	4	3	–	-1	-1	-1	-1
REF	2	2	$\leq$2	3	4	4	3	4	–	-1	-1	-1
R1A	2	2	$\leq$2	4	4	4	4	4		–	4	
RMA	2	2	$\leq$2	3	4	4	3	4		3	–	
REA	2	2	$\leq$2	3	4	4	3	4	4	3	4	–
U1O	$\geq$2	$\geq$2	-1	4	4	4	$\geq$2	$\geq$2	-1	-1	-1	-1
UMO	2,3	$\geq$2	-1	3	$\geq$3	$\geq$3	2,3	$\geq$2	-1	-1	-1	-1
UEO	2,3	$\geq$2		3	$\geq$3	$\geq$3	2,3	$\geq$2		-1	-1	-1
U1S	2	2	-1	$\geq$3	$\geq$3	$\geq$3	$\geq$2	$\geq$2	-1	-1	-1	-1
UMS	2	2	-1	3	$\geq$3	$\geq$3	2,3	$\geq$2	-1	-1	-1	-1
UES	2	2	-1	3	$\geq$3	$\geq$3	2,3	$\geq$2	-1	-1	-1	-1
U1F	2	2	-1	$\geq$3	$\geq$3	$\geq$3	$\geq$2	$\geq$2	-1	-1	-1	-1
UMF	2	2	-1	3	$\geq$3	$\geq$3	2,3	$\geq$2	-1	-1	-1	-1
UEF	2	2	$\leq$2	3	$\geq$3	$\geq$3	2,3	$\geq$2		-1	-1	-1
U1A	2	2	$\leq$2	$\geq$3	$\geq$3	$\geq$3	$\geq$2	$\geq$2				
UMA	2	2	$\leq$2	3	$\geq$3	$\geq$3	2,3	$\geq$2		$\leq$3		
UEA	2	2	$\leq$2	3	$\geq$3	$\geq$3	2,3	$\geq$2		$\leq$3		

Figure 3: Summary of the ability of reliable-channel models to realize the various models we consider here. The entry whose row is labeled with model A and whose column is labeled with model B indicates what we have proved about B's ability to realize A. If the entry is 4, then B exactly realizes A; if the entry is 3, then B realizes A with repetition; if the entry is 2, then B realizes A as a subsequence. A $\geq$ symbol indicates that the value is a lower bound and the relationship could be stronger, while a $\leq$ symbol indicates an upper bound, so the relationship could be weaker. In cases where both upper and lower bounds are known, we list all of the possible relationships. If the entry is -1, then B does not preserve the oscillations of A. Blank entries indicate pairs for which the relationship is unknown.

	U1O	UMO	UEO	U1S	UMS	UES	U1F	UMF	UEF	U1A	UMA	UEA
R1O	4	4		4	4	4	4	4				
RMO	3	4		≥3	4	4	≥3	4				
REO	3	4	4	≥3	4	4	≥3	4	4			
R1S	≥3	≥3		4	4	4	≥3	≥3				
RMS	3	≥3		≥3	4	4	≥3	≥3				
RES	3	≥3		≥3	4	4	≥3	≥3				
R1F	≥3	≥3		4	4	4	4	4				
RMF	3	≥3		≥3	4	4	≥3	4				
REF	3	≥3		≥3	4	4	≥3	4	4			
R1A	≥3	≥3		4	4	4	4	4		4	4	
RMA	3	≥3		≥3	4	4	≥3	4		≥3	4	
REA	3	≥3		≥3	4	4	≥3	4	4	≥3	4	4
U1O	–	4		4	4	4	4	4				
UMO	3	–		≥3	4	4	≥3	4				
UEO	3	4	–	≥3	4	4	≥3	4	4			
U1S	≥3	≥3		–	4	4	≥3	≥3				
UMS	3	≥3		≥3	–	4	≥3	≥3				
UES	3	≥3		≥3	4	–	≥3	≥3				
U1F	≥3	≥3		4	4	4	–	4				
UMF	3	≥3		≥3	4	4	≥3	–				
UEF	3	≥3		≥3	4	4	≥3	4	–			
U1A	≥3	≥3		4	4	4	4	4		–	4	
UMA	3	≥3		≥3	4	4	≥3	4		≥3	–	
UEA	3	≥3		≥3	4	4	≥3	4	4	≥3	4	–

Figure 4: Summary of the ability of unreliable-channel models to realize the various models we consider here. The entry whose row is labeled with model A and whose column is labeled with model B indicates what we have proved about B's ability to realize A. If the entry is 4, then B exactly realizes A; if the entry is 3, then B realizes A with repetition; if the entry is 2, then B realizes A as a subsequence. A $\geq$ symbol indicates that the value is a lower bound and the relationship could be stronger, while a $\leq$ symbol indicates an upper bound, so the relationship could be weaker. In cases where both upper and lower bounds are known, we list all of the possible relationships. If the entry is -1, then B does not preserve the oscillations of A. Blank entries indicate pairs for which the relationship is unknown.

The BGP specification [16] outlines technical details of interdomain route computation, but leaves some room for interpretation. As an example, the amount of time nodes wait before sending route announcements can vary. In some cases, longer wait times may slow BGP convergence because nodes' discovery of potential routes is delayed; in other cases, longer wait times may hasten convergence because nodes do not waste resources on spurious or transient announcements. We have studied these types of effects in this paper, and some of the identified models represent various interpretations of the BGP specification.

The *Route Refresh Capability* [2], which is an optional extension to BGP, can be used to learn, immediately and on demand, a node's current route choice. Although intended to prevent information loss after a policy change, nodes can use this capability to poll nodes for the most current update. This permits implementation of the polling models, and the BGP specification alludes to this possibility. We show that certain types of nonconvergence cannot be realized in these polling models.

Nodes' ranking functions and permitted paths are independently configured inputs, and previous work showed that BGP may not converge on certain inputs [18]. Attempting to characterize these inputs has given some necessary conditions [4] and some sufficient conditions [6, 8, 9, 17] on the inputs for convergence, but not a complete characterization. These results were proved in different communication models. For example, Feamster, Johari, and Balakrishnan [4] used a polling model. In contrast, other work [6, 8, 9, 17] assumed a queueing or message-passing model. In addition to these results about policy constraints, some hardness results are also known. Using a queueing model, Griffin, Shepherd, and Wilfong [9] showed the decision problem of whether an SPP instance has a stable, consistent path assignment is NP-complete. Using a polling model, Fabrikant and Papadimitriou [3] showed that determining whether BGP converges on a succinctly-represented problem instance is PSPACE-complete. Finally, previous work on game-theoretic models of BGP (*e.g.*, [7, 12]) have assumed polling models. Our results relating various models, including those used in previous work, suggest which models are best used for continuing this line of work. For example, here we show that although the sufficient condition in [9] was shown in a queueing model, the condition also guarantees convergence in a polling model.

5 Conclusions and Future Work

We have systematically defined a taxonomy of communication models that play an important, but generally overlooked, role in the convergence behavior of distributed autonomous routing algorithms; we have defined new realization relationships and given an extensive description of which ones hold between different pairs of these models. We have shown that the queueing (and some other) models are very strong, in that they can capture the algorithm behavior seen in all the other models. As such, they are useful for proving guarantees of convergence. We have also shown that the polling models are strictly weaker than other models in that they do not capture all the algorithm behavior that may be seen using other models (and in the real world); thus, they may be useful for providing examples of divergence but not for proving convergence guarantees.

There are important questions remaining in this line of work. We have not extensively studied models in which multiple nodes are activated simultaneously; polling in that case is strictly stronger than the polling models we consider here,[2] but we do not expect this to be true of other models. The question of algorithm behavior in the context of mixed channels may also be interesting. Although unreliable channels model reliable channels—so our results for unreliable channels do apply with a mixture of reliable and unreliable channels—we do not have results when, *e.g.*, some nodes poll and other act on messages. Finally, the question of behavior in the presence of adversarial or unfaithful nodes is important. We assume that nodes use the most recent information they have to execute the algorithm when activated; however, in adversarial settings, nodes may use the other messages read to act in different ways that may affect algorithm convergence.

References

[1] P. Chambart and P. Schnoebelen. Mixing lossy and perfect FIFO channels. In *CONCUR*, pages 340–355, 2008.

[2] E. Chen. Route refresh capability for BGP–4. RFC 2918, September 2000.

[3] A. Fabrikant and C. H. Papadimitriou. The complexity of game dynamics: BGP oscillations, sink equilibria, and beyond. In *Proc. SIAM-ACM Symp. Discrete Alg. (SODA)*, pages 844–853, January 2008.

[4] N. Feamster, R. Johari, and H. Balakrishnan. Implications of autonomy for the expressiveness of policy routing. *IEEE/ACM Trans. Networking*, 15(6):1266–1279, December 2007.

[5] L. Gao, T. Griffin, and J. Rexford. Inherently safe backup routing with bgp. In *INFOCOM*, pages 547–556, 2001.

[6] L. Gao and J. Rexford. Stable internet routing without global coordination. *IEEE/ACM Trans. Networking*, 9(6):681–692, December 2001.

[7] S. Goldberg, S. Halevi, A. D. Jaggard, V. Ramachandran, and R. N. Wright. Rationality and traffic attraction: incentives for honest path announcements in BGP. In *Proc. ACM SIGCOMM*, pages 267–278, 2008.

[8] T. G. Griffin, A. D. Jaggard, and V. Ramachandran. Design principles of policy languages for path vector protocols. In *Proc. ACM SIGCOMM'03*, pages 61–72, August 2003.

[9] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Trans. Networking*, 10(2):232–243, April 2002.

[2]See Ex. A.6 in App. A.

[10] T. G. Griffin and J. L. Sobrinho. Metarouting. In *Proc. ACM SIGCOMM'05*, pages 1–12, August 2005.

[11] B. Leong, S. Mitra, and B. Liskov. Path vector face routing: Geographic routing with local face information. In *Proc. 13th IEEE Int'l Conf. Network Protocols (ICNP)*, pages 147–158, November 2005.

[12] H. Levin, M. Schapira, and A. Zohar. Interdomain routing and games. In *Proc. 40th ACM Symp. Theory of Computing (STOC)*, pages 57–66, May 2008.

[13] N. A. Lynch. *Distributed Algorithms.* Morgan Kaufmann Publishers, 1996.

[14] G. S. Malkin. RIP version 2. RFC 2453, November 1998.

[15] C. E. Perkins, E. M. Belding-Royer, and S. R. Das. Ad hoc on-demand distance vector (AODV) routing. RFC 3561, July 2003.

[16] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (BGP–4). RFC 4271, January 2006.

[17] J. L. Sobrinho. An algebraic theory of dynamic network routing. *IEEE/ACM Trans. Networking*, 13(5):1160–1173, October 2005.

[18] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *Comp. Networks*, 32(1):1–16, January 2000.

A Some Examples

Example A.1. The instance DISAGREE from [9] is depicted in Fig. 5. In this network, node x prefers choosing the route through neighbor y over the direct route to d; likewise, node y prefers the route through x over the direct route to d. The network contains two stable solutions, $\pi_1 = (d, xyd, yd)$ and $\pi_2 = (d, xd, yxd)$. A result from [9] states that multiple stable solutions imply the existence of a policy structure called a *dispute wheel*; the absence of a dispute wheel is the broadest-known sufficient condition for convergence. However, the existence of a dispute wheel does not imply divergence; in particular, this example shows that divergence depends on the communication model.

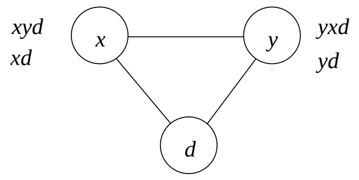

Figure 5: A network instance that cannot oscillate in REO, REF, R1A, RMA, or REA.

In the R1O model, let node d activate, announcing itself to nodes x and y. Then let node x activate, processing the message from d, and then let node y activate, processing the message from d. Both nodes now choose the direct route to d and have announced this choice in the channel to the other. Now, alternately activating nodes x and y *ad infinitum*, processing the channels between them, will lead to a fair oscillation (assuming that the channels to and from node d are processed infinitely often, which have no effect on the route choice). This is because the choice of more preferred route causes a message to be placed in the channel withdrawing the direct route; that withdrawal removes the preferred route on the next activation, causing the direct route to be chosen and advertised. This returns the network to a previous state—including the messages in the channels—indicating a cycle in the algorithm execution. Note that, because this oscillation is an R1O activation sequence, a similar oscillation can be produced in any model that realizes R1O as a subsequence.

However, the models that do not realize R1O as a subsequence, namely REO, REF, R1A, RMA, and REA, are separated from the others through this example. We now show that DISAGREE cannot oscillate in these five models. We do this by showing impossibility of oscillation in the REF and RMA models; impossibility in the other three models is implied by our general realizability results.

First consider RMA. In this model, when a node is activated, it receives the most current state of any neighbor that is polled. Consider any fair activation sequence in this model. At some time, node d must be activated, after which point its existence will be known by x or y when the channel to d is polled. Without loss of generality, let x be the first node to poll d after d is activated; this causes x to choose the direct route xd. Any further activations of x or d have no effect. Because the sequence is fair, y must then activate, and either x alone, d alone, or both x and d will be polled at that time. If x is included in the set of neighbors polled, then y will choose the route yxd, and any further activations of any node will not change the network path assignment (convergence). If x is not included, then y will choose the direct route yd. After this point, because the sequence is fair, either x must activate polling y, or y must activate polling x (any other activation and polling will have no effect). The first node that does will choose the route through the other, entering a stable converged state. Therefore, any RMA activation sequence must converge.

Now consider REF. In this model, nodes pull one message off every channel when activated. Consider any fair activation sequence. Until node d activates, nodes have no route and will announce nothing. Once d activates, its announcement appears in the channels to x and y. Without loss of generality, let x activate first after this point; it will choose xd after processing the announcement from d and will place a corresponding announcement in the channel to y. The next time y activates, it must read this announcement, causing it to choose its most preferred route yxd. This results in the converged state (d, xd, yxd).

To summarize, there is no oscillatory activation sequence for DISAGREE in REO, REF, R1A, RMA, or REA; however, an oscillatory sequence can be found in any of the other models. This provides a separation result among models, because these five models cannot realize any of the others, even in the weakest way (REF $\ntrianglelefteq$ R1O, RMA $\ntrianglelefteq$ R1O, and anything implied by these).

Example A.2. Consider the instance depicted in Fig. 6. Each node's route preferences are listed next to that node from top to bottom in order of decreasing preference. We show that this instance cannot oscillate in the polling models (R1A, RMA, REA), but can oscillate in the other two models in which DISAGREE (Ex. A.1) cannot, providing a separation among the polling and other models.

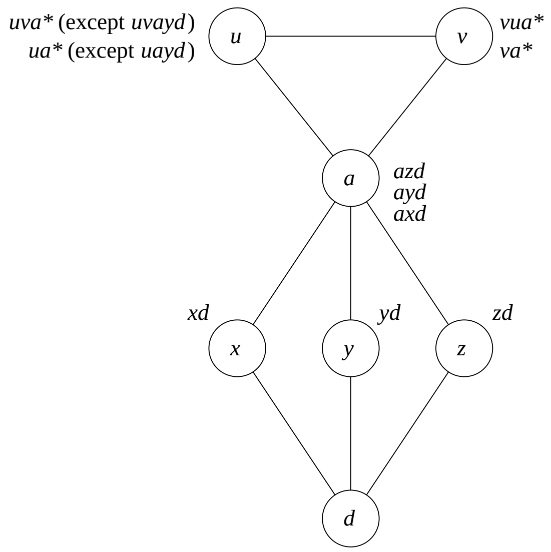

Figure 6: A network instance that can oscillate in REO and REF but not in R1A, RMA, or REA.

We show that an oscillation cannot occur in RMA; the other results are implied by our general realizability results. Consider any fair activation sequence in RMA; recall that, in such a sequence, nodes are activated to poll the current state of at least one of its neighbors. Before d is activated, the path assignment does not change. At some point after d is activated, node z must be activated; after that, node a must activate polling z at some point, when it will choose its most preferred route azd. After this point, node a's choice will not change, because all subpaths of azd are most preferred at their respective source nodes, and the choices of x and y will not change unless a is polled. Now consider the first activation of either u or v that polls a after the choice of azd. We have four cases:

1. Node u is activated, and it believes that node v is routing directly through a (either from a poll of v or from a previously learned, potentially stale state). Then u chooses the extension of the route through v. The next noteworthy activation (one in which the route choice of u or v changes) is that of node v. We have two cases:

   (a) If v polls u, then v must choose $vazd$. The next noteworthy activation is node u polling v, at which point u will choose $uvazd$. This can only result in convergence to the path assignment $(d, xd, yd, zd, azd, uvazd, vazd)$.

   (b) If v does not poll u, then v may choose $vazd$ (leading to convergence as describe above), or it may choose some stale route through u (which must be $vuayd$ or

$vuaxd$). The next noteworthy activation is when u polls v, falling into the next case below.

2. Node u is activated, and it believes that node v does not have a feasible route it can extend (either from a poll of v or from a previously learned, potentially stale state). Then u must choose $uazd$. Node u will not change unless node v chooses a direct route through a. There are two cases for node v's next activation:
   (a) Node v polls node u; if so, the instance converges to $(d, xd, yd, zd, azd, uazd, vuazd)$, because $vuazd$ is most preferred.
   (b) Node v does not poll node u; in this case, it chooses $vazd$. Now, both u and v have chosen to directly route through a. The next activation of either that polls the other will reach a converged state, and no other activation will change the choice of u and v. This must happen because of fairness.
3. Node v is activated, and it believes that node u is using $uaxd$. It then chooses $vuaxd$. The only noteworthy activation is when u polls a; node u would update its choice to $uazd$. The next time node v polls u, node v chooses $vuazd$, which leads to the converged state $(d, xd, yd, zd, azd, uazd, vuazd)$.
4. Node v is activated, and it believes that node u has no feasible route that it can extend (u has no route or is routing through it). Node v must then choose $vazd$. Activating v again changes nothing (unless it learns that u is using $uaxd$, which is the case covered above.) We then have two cases when u next activates:
   (a) Node u polls node v before or at the same time that it polls node a. In this case, node u chooses $uvazd$, leading to the converged state $(d, xd, yd, zd, azd, uvazd, vazd)$.
   (b) Node u polls node a before it polls node v. In this case, u chooses the route $uazd$, having had no route earlier. Now, both x and y are routing directly through a; the next activation of either that polls the other will lead to a converged state.

The above discussion shows that the network in Fig. 6 cannot oscillate in RMA (and consequently cannot in R1A or REA). We next show that the network can oscillate in REO (and consequently REF, because REO $\leq$ REF). An activation sequence in this model consists of activating a node, which pulls exactly one message off of all nonempty channels, processes those messages, and only then chooses a route and announces an update if necessary.

Consider the activation sequence and corresponding route choices below:

$$\begin{array}{rcccccccccccccc}
t & = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
U(t) & = & d & x & a & u & v & y & a & u & v & z & a & v & u \\
\pi_{U(t)}(t) & = & d & xd & axd & uaxd & vuaxd & yd & ayd & (u)\epsilon & vayd & zd & azd & vazd & uazd
\end{array}$$

$$\begin{array}{ccccccccc}
 & 14 & 15 & 16 & 17 & \dots & 2k & 2k+1 & \dots \\
\cdots & v & u & v & u & \dots & v & u & \dots \\
 & vuazd & uvazd & vuazd & uvazd & \dots & vuazd & uvazd & \dots
\end{array}$$

The first seven steps are straightforward; the activated nodes choose the best available routes, and at most one message is ever in a channel. In step 8, node u does not have a path, because it refuses paths containing y. It processes ayd and $vuaxd$ from its channels, and is left to choose ϵ because both of those are infeasible. This permits node v to choose $vayd$ in step 9, as it processes both the announcement of ayd from a and ϵ from u. Note that this injects an announcement of $vayd$ into the channel to u. When v is next activated in step 12, it chooses $vazd$ because the channel from u is empty; it subsequently injects the announcement $vazd$ into the channel to u. Therefore, although u does not have a path, there are two messages in the channel from v; the delay in receiving the current state (two activations are necessary in REO) is essential for the oscillation to work. When u finally activates in step 13, it chooses $uazd$ because the message from v contains an infeasible path ($vayd$). At this point, both nodes are routing directly to a, but simultaneously have in their channels announcements that will change their route choice upon next activation. The classic DISAGREE oscillation will work here, and as u and v alternately activate, they will oscillate between their direct and indirect routes.

This example thus gives a separation result REO $\not\lesssim$ RMA, showing that there are oscillations realizable in REO and REF that cannot be realized in the polling models.

Example A.3. Earlier, we showed that $x\mathsf{E}y \lesssim x\mathsf{1}y$ for any x, y we consider in this paper. The network instance depicted in Fig. 7 shows that this general result cannot be strengthened to an exact realization, *i.e.*, $x\mathsf{E}y \not\leq x\mathsf{1}y$.

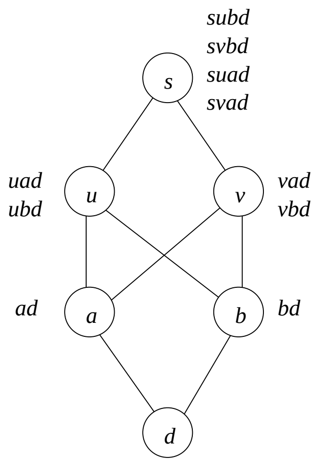

Figure 7: A network instance on which an execution in REO cannot be exactly realized by an execution in R1O.

To show this, we begin with an activation sequence in the REO model and show that it cannot be exactly realized in the R1O model. Consider the activation sequence below; the path sequence marked with a star ($\star$) is induced in the REO model (parts of the other assignment sequence may vary as explained below):

$$\begin{array}{rcccccccccc} t = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ U(t) = & d & b & u & v & a & u & v & s & s & s \\ \star\ (\mathsf{REO})\ \pi_{U(t)}(t) = & d & bd & ubd & vbd & ad & uad & vad & subd & suad & suad \\ (\mathsf{R1O})\ \pi_{U(t)}(t) = & d & bd & ubd & vbd & ad & uad & vad & subd & suad & svbd \end{array}$$

We now attempt to exactly realize this sequence in the R1O model. To achieve the same sequence of path assignments in steps 1–7 as $\star$, we must have the identical activation sequence in R1O. Also note that for each of these activations, only one incoming channel has a message, and that channel has only one message, so even though R1O offers a choice of channel, the messages processed are identical in both models' executions.

After step 7, the channel from u to s has the messages $\{ubd,\ uad\}$ in it, and the channel from v to s has the messages $\{vbd,\ vad\}$ in it. To achieve the assignment $\pi_s(8) = subd$, we must activate s and pull the message *ubd* from the channel from u. In the REO model, this means that the message *vbd* is also pulled from the channel from v, because in this model, every neighbor's channel is processed when activated. Because *subd* is preferred to *svbd*, the route *svbd* is never assigned to s in the REO path sequence (node v, already having its most preferred route *vad*, will never again announce *vbd* along the channel). However, in the R1O model, only one channel is processed at a time; therefore, the message *vbd* remains to be examined at a later step.

To achieve $\pi_s(9) = suad$, we must process the message *uad* from the channel from u; this makes *subd* infeasible. The only reason that s would not choose *suad* as its route is if it also knows the route *svbd*, which is ranked higher. In the REO model, when the channel is read to process *uad*, the channel from v (now containing *vad*) must also be read, and this removes the route *svbd* from consideration. In the R1O model, it must be that the channel from v is not read. After step 9, the REO execution converges to $(d, ad, bd, uad, vad, suad)$. In the R1O model, the outstanding messages must be processed; this causes $\pi_s(10) = svbd$ because *svbd* is preferred to *suad*. The choice of *svbd* would never happen in the REO model, but to exactly realize the path assignment sequence up to step 9, it is impossible for the message *vbd* to be processed earlier. Because this state in the R1O path-assignment sequence does not appear in the REO path-assignment sequence, we conclude that REO $\not\lesssim$ R1O, which means the general result $x\mathsf{E}y \lesssim x\mathsf{1}y$ cannot be strengthened.

Although other examples may show this, we remark that trivial network examples do not lead to the desired result. Informally, this is because reordering message processing can overcome the restriction of processing only one channel at a time.

Example A.4. Consider the instance depicted in Fig. 8; the permitted paths are *ad*, *bd*, *ubd*, *uad*, *suad*, and *subd*, with *ubd* preferred to *uad* and *suad* preferred to *subd*. In the REA model, activate the nodes in the sequence d, a, u, b, u, s; the path selected in step t by the active node $U(t)$ is shown in the third line ((REA) $\pi_{U(t)}(t)$) below:

$$\begin{array}{rcccccc} t = & 1 & 2 & 3 & 4 & 5 & 6 \\ U(t) = & d & a & u & b & u & s \\ (\mathsf{REA})\ \pi_{U(t)}(t) = & d & ad & uad & bd & ubd & subd \end{array}$$

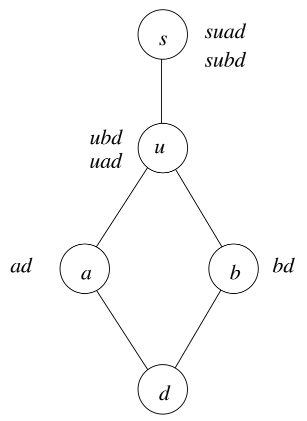

Figure 8: A network instance on which an execution in REA cannot be realized with repetition in R1O.

Before the last step ($t = 6$), the first message in the channel (u, s) is *uad* and the second message is *ubd*. Now consider any expansion of this sequence of path assignments obtained by replacing each path assignment by one or more copies of itself. This again has one or more instances of *uad* in the channel (u, s) before any instances of *ubd*. Any activation sequence in R1O that attempts to achieve such an expansion would need to process these instances of *uad* before assigning *subd*; however, *subd* must be assigned to s before *suad* ever is, so R1O cannot realize this path assignment sequence with repetition. (However, from other results, we know that it is indeed realizable as a subsequence; in this example, the R1O activation sequence that activates the channels $(d, a), (a, u), (d, b), (b, u), (u, s), (u, s)$ produces a path assignment sequence that differs from the original one only in the insertion of *suad* just before *subd*.) Thus we have that REA$\not\lesssim$R1O.

Example A.5. Consider the instance depicted in Fig. 9; the permitted paths are *ad*, *bd*, *xd*, *cad*, *cbd*, *scad*, *scbd*, and *sxd*, with *scbd* preferred to *sxd* preferred to *scad* at s and *cad* preferred to *cbd* at c. In the REA model, activate the nodes in the sequence d, b, c, x, s, a, c, s; the path selected in step t by the active node $U(t)$ is shown in the third line ((REA) $\pi_{U(t)}(t)$) below:

$$\begin{array}{rcccccccc} t = & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ U(t) = & d & b & c & x & s & a & c & s \\ (\mathsf{REA})\ \pi_{U(t)}(t) = & d & bd & cbd & xd & scbd & ad & cad & sxd \end{array}$$

We now attempt to induce this sequence in the R1S model. The sequence of channels activated must start with $(d, b), (b, c), (d, x), (c, s)$ because the changed paths are all being learned for the first time. Note in particular that the first activation of s in the REA model, at which point s chooses *scbd*, also allows s to learn the path *sxd* without a second activation because it is learning information from all of its neighbors. The sequence of activated channels must continue with $(d, a), (a, c)$, again because the changed routes are being learned for the first time. In the R1S model, at this point: s now knows only *scbd*, the channel (c, s) contains *cad*, and the channel (x, s) contains *xd*. In order to match the

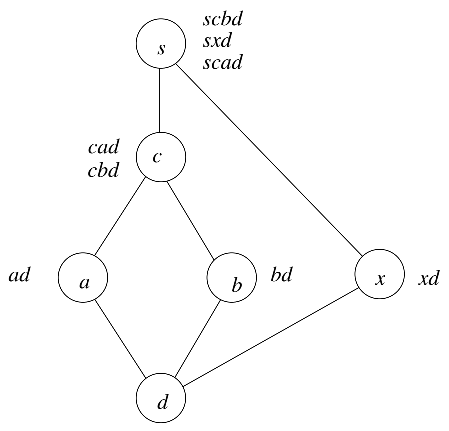

Figure 9: A network instance in which an execution in REA cannot be exactly realized in R1S.

original path assignment sequence, we must activate a single channel and, as a result, have s choose sxd. This channel must thus be (x, s), because s has not yet learned sxd; however, because s does not activate (c, s) until some point later in the execution, s will continue to use its most preferred path $scbd$. Thus, REA $\not\leq$ R1S.

Example A.6. Consider the instance DISAGREE from Ex. A.1 above, and assume a model like R1A except that multiple nodes may be activated in each step (with each node, as in R1A, processing all of the messages in exactly one of its incident channels). If the channels that are activated at each step are given by $X(t)$, then we can have a cycle as follows.

$t =$	1	2	3	4	5
$X(t) =$	$\{(d,d)\}$	$\{(d,x),(d,y)\}$	$\{(x,y),(y,x)\}$	$\{(d,x),(d,y)\}$	$\{(x,y),(y,x)\}$
$\pi_x(t) =$	$--$	xd	xyd	xyd	xd
$\pi_y(t) =$	$--$	yd	yxd	yxd	yd

$\cdots$	6	7	$\ldots$
$\cdots$	$\{(d,x),(d,y)\}$	$\{(x,y),(y,x)\}$	$\ldots$
$\cdots$	xd	xyd	$\ldots$
$\cdots$	yd	yxd	$\ldots$

Here x and y are always activated simultaneously, either both polling their channels to d or both polling their channels to the other. If we require that each of (x, y) and (y, x) is activated in infinitely many steps in which the other is *not* activated (perhaps as a modification of fairness for the case in which multiple nodes may be activated in a given step), then the arguments of Ex. A.1 apply and we cannot get an oscillation.