On the Push&Pull Protocol for Rumour Spreading

[Extended Abstract] *

Hüseyin Acan
School of Mathematical Sciences
Monash University
Clayton, Australia
huseyin.acan@monash.edu

Andrea Collevecchio
School of Mathematics
Monash University
Clayton, Australia
and Ca’ Foscari University
Venice, Italy
andrea.collevecchio@monash.edu

Abbas Mehrabian
Department of Combinatorics and Optimization
University of Waterloo
Waterloo, Canada
amehrabi@uwaterloo.ca

Nick Wormald
School of Mathematical Sciences
Monash University
Clayton, Australia
nick.wormald@monash.edu

ABSTRACT

The asynchronous push&pull protocol, a randomized distributed algorithm for spreading a rumour in a graph G, is defined as follows. Independent exponential clocks of rate 1 are associated with the vertices of G, one to each vertex. Initially, one vertex of G knows the rumour. Whenever the clock of a vertex x rings, it calls a random neighbour y: if x knows the rumour and y does not, then x tells y the rumour (a push operation), and if x does not know the rumour and y knows it, y tells x the rumour (a pull operation). The average spread time of G is the expected time it takes for all vertices to know the rumour, and the guaranteed spread time of G is the smallest time t such that with probability at least 1 − 1/n, after time t all vertices know the rumour. The synchronous variant of this protocol, in which each clock rings precisely at times 1, 2, . . ., has been studied extensively.

We prove the following results for any n-vertex graph: in either version, the average spread time is at most linear even if only the pull operation is used, and the guaranteed spread time is within a logarithmic factor of the average spread time, so it is O(n log n). In the asynchronous version, both the average and guaranteed spread times are Ω(log n). We give examples of graphs illustrating that these bounds are best possible up to constant factors.

We also prove the first theoretical relationships between the guaranteed spread times in the two versions. Firstly, in all graphs the guaranteed spread time in the asynchronous version is within an O(log n) factor of that in the synchronous version, and this is tight. Next, we find examples of graphs whose asynchronous spread times are logarithmic, but the synchronous versions are polynomially large. Finally, we show for any graph that the ratio of the synchronous spread time to the asynchronous spread time is O(n^{2/3}).

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—data communications; F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—computations on discrete structures; G.2.2 [Discrete Mathematics]: Graph Theory—network problems

General Terms

Algorithms, Performance, Theory

Keywords

randomized rumour spreading; push&pull protocol; asynchronous time model

1. INTRODUCTION

Randomized rumour spreading is an important primitive for information dissemination in networks and has numerous applications in network science, ranging from spreading information in the WWW and Twitter to spreading viruses and diffusion of ideas in human communities. A well studied rumour spreading protocol is the (synchronous) push&pull protocol, introduced by Demers et al. [6] and popularized by Karp et al. [20]. Suppose that one node in a network is aware of a piece of information, the ‘rumour’, and wants to spread it to all nodes quickly. The protocol proceeds in rounds. In each round, every informed node contacts a random neighbour and sends the rumour to it (‘pushes’ the rumour), and every uninformed nodes contacts a random neighbour and gets the rumour if the neighbour knows it (‘pulls’ the rumour).
A point to point communication network can be modelled as an undirected graph: the nodes represent the processors and the links represent communication channels between them. Studying rumour spreading has several applications to distributed computing in such networks, of which we mention just two. The first is in broadcasting algorithms: a single processor wants to broadcast a piece of information to all other processors in the network (see [17] for a survey). There are at least four advantages to the push&pull protocol: it puts much less load on the edges than naive flooding, it is simple (each node makes a simple local decision in each round; no knowledge of the global topology is needed; no state is maintained), scalable (the protocol is independent of the size of network: it does not grow more complex as the network grows) and robust (the protocol tolerates random node/link failures without the use of error recovery mechanisms, see [11]). A second application comes from the maintenance of databases replicated at many sites, e.g., yellow pages, name servers, or server directories. There are updates injected at various nodes, and these updates must propagate to all nodes in the network. In each round, a processor communicates with a random neighbour and they share any new information, so that eventually all copies of the database converge to the same contents. See [6] for details. Other than the aforementioned applications, rumour spreading protocols have successfully been applied in various contexts such as resource discovery [16], distributed averaging [5], data aggregation [21], and the spread of computer viruses [3].

In this paper we only consider simple, undirected and connected graphs. Given a graph and a starting vertex, the spread time of a certain protocol is the time it takes for the rumour to spread in the whole graph, i.e. the time difference between the moment the protocol is initiated and the moment when everyone learns the rumour. For the asynchronous push&pull protocol, it turned out that the spread time is closely related to the expansion profile of the graph. Let \( \Phi(G) \) and \( o(G) \) denote the conductance and the vertex expansion of a graph \( G \), respectively. After a series of results by various scholars, Giakkoupis [14, 15] showed the spread time is \( O(\min\{\Phi(G)^{-1} \log n, o(G)^{-1} \log^2 n\}) \). This protocol has recently been used to model news propagation in social networks. Doerr et al. [7] proved an upper bound of \( O(\log n) \) for the spread time on Barabási-Albert graphs, and Fountoulakis et al. [13] proved the same upper bound (up to constant factors) for the spread time on Chung-Lu random graphs.

All the above results assumed a synchronized model, i.e., all nodes take action simultaneously at discrete time steps. In many applications and certainly in real-world social networks, this assumption is not very plausible. Boyd et al. [5] proposed an asynchronous time model with a continuous time line. Each node has its own independent clock that rings at the times of a rate 1 Poisson process. (Since the times between rings is an exponential random variable, we shall call this an exponential clock.) The protocol now specifies for every node what to do when its own clock rings. The rumour spreading problem in the asynchronous time model has so far received less attention. Rumour spreading protocols in this model turn out to be closely related to Richardson’s model for the spread of a disease [10] and to first-passage percolation [18] with edges having i.i.d. exponential weights. The main difference is that in rumour spreading protocols each vertex contacts one neighbour at a time. So, for instance in the ‘push only’ protocol, the net communication rate outwards from a vertex is fixed, and hence the rate that the vertex passes the rumour to any one given neighbour is inversely proportional to its degree (the push&pull protocol is a bit more complicated). Hence, the degrees of vertices play a crucial role not seen in Richardson’s model or first-passage percolation. However, on regular graphs, the asynchronous push&pull protocol, Richardson’s model, and first-passage percolation are essentially the same process, assuming appropriate parameters are chosen. In this sense, Fill and Pemantle [12] and Bollobás and Kohayakawa [4] showed that a.a.s. the spread time of the asynchronous push&pull protocol is \( \Theta(\log n) \) on the hypercube graph. Janson [19] and Amini et al. [2] showed the same results (up to constant factors) for the complete graph and for random regular graphs, respectively. These bounds match the same order of magnitude as in the synchronized case. Doerr et al. [9] experimentally compared the spread time in the two models. They state that ‘Our experiments show that the asynchronous model is faster on all graph classes [considered here].’ However, a general relationship between the spread times of the two variants has not been proved theoretically.

Fountoulakis et al. [13] investigated the asynchronous push&pull protocol on Chung-Lu random graphs with exponent between 2 and 3. For these graphs, they showed that a.a.s. after some constant time, \( n - o(n) \) nodes are informed. Doerr et al. [8] showed that for the preferential attachment graph (the non-tree case), a.a.s. all but \( o(n) \) vertices receive the rumour in time \( O(\sqrt{\log n}) \), but to inform all vertices a.a.s., \( O(\log n) \) time is necessary and sufficient. Panagiotou and Speidel [22] studied this protocol on Erdős-Rényi random graphs and proved that if the average degree is \( (1 + \Theta(1))\log n \), a.a.s. the spread time is \( (1 + o(1))\log n \).

1.1 Our contribution

In this paper we answer a fundamental question about the asynchronous push&pull protocol: what are the minimum and maximum spread times on an \( n \)-vertex graph? Our proof techniques yield new results on the well studied synchronous version as well. We also compare the performances of the two protocols on the same graph, and prove the first theoretical relationships between their spread times.

We now formally define the protocols. In this paper \( G \) denotes the ground graph which is simple and connected, and \( n \) counts its vertices, and is assumed to be sufficiently large.

Definition 1. (Asynchronous push&pull protocol) Suppose that an independent exponential clock of rate 1 is associated with each vertex of \( G \). Suppose that initially, some vertex \( v \) of \( G \) knows a piece of information, the so-called rumour. The rumour spreads in \( G \) as follows. Whenever the clock of a vertex \( x \) rings, this vertex performs an ‘action’: it calls a random neighbour \( y \); if \( x \) knows the rumour and \( y \) does not, then \( x \) tells \( y \) the rumour (a push operation), and if \( x \) does not know the rumour and \( y \) knows it, \( y \) tells \( x \) the rumour (a pull operation). Note that if both \( x \) and \( y \) know the rumour or neither of them knows it, then this action is useless. Also, vertices have no memory, hence \( x \) may call the same neighbour several consecutive times. The spread time of \( G \) starting from \( v \), written \( ST_\alpha(G, v) \), is the first time that all vertices of \( G \) know the rumour. Note that
this is a continuous random variable, with two sources of randomness: the Poisson processes associated with the vertices, and random neighbour-selection of the vertices. The guaranteed spread time of \(G\), written \(wast_4(G)\), is the smallest deterministic number \(t\) such that for every \(v \in V(G)\) we have \(P[ST_4(G, v) > t] \leq 1/n\). The worst average spread time of \(G\), written \(wast_4(G)\), is the smallest deterministic number \(t\) such that for every \(v \in V(G)\) we have \(E[ST_4(G, v)] \leq t\).

**Definition 2.** (Synchronous push\&pull protocol) Initially some vertex \(v\) of \(G\) knows the rumour, which spreads in \(G\) in a round-robin manner: in each round \(1, 2, \ldots\), all vertices perform actions simultaneously. That is, each vertex \(x\) calls a random neighbour \(y\); if \(x\) knows the rumour and \(y\) does not, then \(x\) tells \(y\) the rumour (a push operation), and if \(x\) does not know the rumour and \(y\) knows it, \(y\) tells \(x\) the rumour (a pull operation). Note that this is a synchronous protocol, e.g. a vertex that receives a rumour in a certain round cannot send it on in the same round. The spread time of \(G\) starting from \(v\), \(ST_4(G, v)\), is the first time that all vertices of \(G\) know the rumour. Note that this is a discrete random variable, with one source of randomness: the random neighbour-selection of the vertices. The guaranteed spread time of \(G\), written \(gst_4(G)\), and the worst average spread time of \(G\), written \(wast_4(G)\), are defined in an analogous way to the asynchronous case.

We remark that the notion of ‘guaranteed spread time’ was first defined by Feige et al. [11] under the name ‘almost sure rumor coverage time’ for the ‘push only’ protocol. (In this protocol, which was studied prior to push\&pull, the informed nodes push the rumour, but the uninformed ones do nothing. The ‘pull only’ protocol is defined conversely.)

It turns out that changing the starting vertex affects the spread time by at most a multiplicative factor of 2. Specifically, in [1] we prove that for any two vertices \(u\) and \(v\),

\[
ST_4(G, u) \leq 2ST_4(G, v) \quad \text{and} \quad ST_4(G, u) \leq 2ST_4(G, v).
\]

(For random variables \(X\) and \(Y\), \(X \leq Y\) means \(X\) is stochastically dominated by \(Y\), that is, for any \(t\) we have \(P[X \leq t] \leq P[Y \geq t]\).) These imply that \(wast_4(G) \leq 2E[ST_4(G, v)]\) and \(wast_4(G) \leq 2E[ST_4(G, v)]\) for any vertex \(v\).

Our first main result is the following theorem.

**Theorem 1.** The following hold for any \(n\)-vertex graph \(G\).

\[
(1 - 1/n) wast_4(G) \leq gst_4(G) \leq e wast_4(G) \log n, \quad (2)
\]

\[
wast_4(G) = \Omega(\log n) \quad \text{and} \quad wast_4(G) = O(n), \quad (3)
\]

\[
gst_4(G) = \Omega(\log n) \quad \text{and} \quad gst_4(G) = O(n \log n). \quad (4)
\]

Moreover, these bounds are asymptotically best possible, up to the constant factors.

Our proof of the right-hand bound in (3) is based on the pull operation only, so this bound applies equally well to the ‘pull only’ protocol.

The arguments for (2) and the right-hand bounds in (3) and (4) can easily be extended to the synchronous variant, giving the following theorem. The bound (7) below also follows from [11, Theorem 2.1], but here we also show its tightness.

**Theorem 2.** The following hold for any \(n\)-vertex graph \(G\).

\[
(1 - 1/n) wast_4(G) \leq gst_4(G) \leq e wast_4(G) \log n, \quad (5)
\]

\[
wast_4(G) = \Omega(n) \quad \text{and} \quad wast_4(G) = O(n), \quad (6)
\]

\[
gst_4(G) = \Omega(n \log n). \quad (7)
\]

Moreover, these bounds are asymptotically best possible, up to the constant factors.

**Open problem 1.** Find the best possible constants factors in Theorems 1 and 2.

We next turn to studying the relationship between the asynchronous and synchronous variants on the same graph.

**Theorem 3.** For any \(G\) we have

\[
gst_4(G) = O(gst_4(G) \log n),
\]

and this bound is best possible, up to the constant factor.

For all graphs we examined a stronger result holds, which suggests the following conjecture.

**Conjecture 1.** For any \(n\)-vertex graph \(G\) we have \(gst_4(G) \leq gסט(G) + O(\log n)\).

Our last main result is the following theorem, whose proof is somewhat technical, and uses couplings with the sequential rumour spreading protocol.

**Theorem 4.** For any \(\alpha \in (0, 1)\) we have

\[
gst_4(G) \leq n^{1-\alpha} + O(gst_4(G)n^{(1+\alpha)/2}).
\]

**Corollary 1.** We have

\[
gst_4(G) = \Omega(1/\log n) \quad \text{and} \quad gست_4(G) = O(n^{2/3}),
\]

and the left-hand bound is asymptotically best possible, up to the constant factor. Moreover, there exist infinitely many graphs for which this ratio is \(\Omega\left(n^{1/3}(\log n)^{-1/3}\right)\).

**Open problem 2.** What is the maximum possible value of the ratio \(gست_4(G)/gст_4(G)\) for an \(n\)-vertex graph \(G\)?

The parameters \(wast_4(G)\) and \(wast_4(G)\) can be approximated easily using the Monte Carlo method: simulate the protocols several times, measuring the spread time of each simulation, and output the average. Another open problem is to design a deterministic approximation algorithm for any one of \(wast_4(G)\), \(gst_4(G)\), \(wast_4(G)\) or \(gst_4(G)\).

For the proofs we use standard graph theoretic arguments and well known properties of the exponential distribution and Poisson processes, in particular the memoryless, and the fact that the union of two Poisson processes is another Poisson process. For proving Theorem 4 we define a careful coupling between the synchronous and asynchronous protocols.

Previous work on the asynchronous push\&pull protocol has focused on special graphs. This paper is the first systematic study of this protocol on all graphs. We believe this protocol is fascinating and is quite different from its synchronous variant, in the sense that different techniques are required for analyzing it, and the spread times of the two
2. PRELIMINARIES AND EXAMPLES

Let $\text{Geo}(p)$ denote a geometric random variable with parameter $p$, namely for any $k \in \{0, 1, \ldots\}$, $\mathbb{P}(\text{Geo}(p) = k) = (1 - p)^k p$. Let $\text{Exp}(\lambda)$ denote an exponential random variable with parameter $\lambda$ and mean $1/\lambda$. For random variables $X$ and $Y$, $X \leqslant Y$ means $X$ and $Y$ have the same distribution. All logarithms are natural. We say an event happens approaches 1 as if the probability that it happens approaches 1 as $n$ grows. Whenever a new vertex is informed, by $v$'s clock rings, it calls $u$ with probability $1/\deg(v)$. Hence, for each vertex $v$, we can replace $v$'s clock by one exponential clock for each incident edge, these clocks being independent of all other clocks and having rate $1/\deg(v)$.

Observation 1. Consider the asynchronous variant. Let $uv$ be an edge. Whenever $v$'s clock rings, it calls $u$ with probability $1/\deg(v)$. Hence, for each vertex $v$, we can replace $v$'s clock by one exponential clock for each incident edge, these clocks being independent of all other clocks and having rate $1/\deg(v)$.

Observation 2. Whenever a new vertex is informed, by memorylessness of the exponential random variable, we may imagine that all clocks are restarted.

The following definition will be used throughout.

Definition 3. (Communication time) For an edge $e = uv$, the communication time via edge $e$, written $T(e)$, is defined as follows. Suppose $\tau$ is the first time that one of $u$ and $v$ learns the rumour, and $\rho$ is the first time after $\tau$ that one of $u$ and $v$ calls the other one. Then $T(e) = \rho - \tau$, which is nonnegative. Note that after time $\rho$, both $u$ and $v$ know the rumour.

Observation 3. Let $uv \in E(G)$. In the synchronous version,

$$T(uv) \overset{d}{=} 1 + \min\{\text{Geo}(1/\deg(u)), \text{Geo}(1/\deg(v))\}.$$  

Using Observations 1 and 2 we obtain a nicer formula for the asynchronous version.

Proposition 1. Let $uv \in E(G)$. In the asynchronous version,

$$T(uv) \overset{d}{=} \text{Exp}(1/\deg(u)) + 1/\deg(v).$$  

Moreover, the random variables $\{T(e)\}_{e \in E(G)}$ are mutually independent.

We next study some important graphs and bound their spread times, partly for showing tightness of some of the bounds obtained, and partly to serve as an introduction to the behaviour of the protocols.

2.1 The complete graph

For the complete graph, $K_n$, we have $\text{wast}_s(K_n) = \log n + O(1)$ (see [1]). In fact, Janson [19] showed that a.a.s. we have $\text{ST}_s(K_n, v) \sim \log n$. Moreover, it is implicit in his proof that $\text{gst}_s(K_n) \sim (3/2) \log n$. For the synchronous version, Karp et al. [20] showed that a.a.s. $\text{ST}_s(K_n, v) \sim \log n$. It follows that $\text{wast}_s(K_n) \sim \log n$. It is implicit in their proof that $\text{gst}_s(K_n) = O(\log n)$.

2.2 The star

The star $G^*_n$ with $n$ vertices has $n - 1$ leaves and a central vertex that is adjacent to every other vertex. It is clear that $\text{ST}_s(G^*_n, v) = 1$ if $v$ is the central vertex and $\text{ST}_s(G^*_n, v) = 2$ otherwise. So we have $\text{wast}_s(G^*_n) = \text{gst}_s(G^*_n) = 2$. In the asynchronous case, we have that $\text{wast}_s(G^*_n) \sim \log n$ and $\text{gst}_s(G^*_n) \sim 2 \log n$ (see [1]). This graph gives that the left-hand bounds in (2), (3), (4), and Corollary 1, and Theorem 3, are tight, up to constant factors.

2.3 The path

For the path graph $P_n$, we have $\text{wast}_s(P_n) \sim n$, which shows that the right-hand bound in (3) is tight, up to the constant factor. Moreover, $\text{gst}_s(P_n) \sim n$. For the synchronous protocol, we have $\text{wast}_s(P_n) = (4/3)n - 2$, which shows that the right-hand bound in (6) is tight, up to the constant factor. Finally, we have $\text{gst}_s(P_n) \sim (4/3)n$. See [1] for details.

2.4 The double star

Consider the tree $DS_n$ consisting of two adjacent vertices of degree $n/2$ and $n - 2$ leaves, see Figure 1(Top). In [1] we show that $\text{gst}_s(DS_n)$ and $\text{gst}_s(DS_n)$ are both $\Theta(n \log n)$, while the average times $\text{wast}_s(DS_n)$ and $\text{wast}_s(DS_n)$ are $\Theta(n)$. This example hence shows tightness of the right-hand bounds in (2), (4), (5) and (7) up to constant factors. The main delay in spreading the rumour in this graph comes from the edge joining the two centres. The idea is that it takes $O(n)$ units of time on average for this edge to pass the rumour, but to be sure that this has happened with probability $1 - 1/n$, we need to wait $O(n \log n)$ units of time.

2.5 The necklace graph

Let $m$ and $k \geq 2$ be positive integers, and let $G$ be the necklace graph given in Figure 1(Bottom), where there are $m$ diamonds, each consisting of $k$ edge-disjoint paths of length 2 with the same end vertices, which we call hubs. The number of vertices is $n = km + m + 1$. Let us analyze the average spread time.

Consider the asynchronous case first. Proposition 1 gives that for each edge $e$,

$$T(e) \overset{d}{=} \text{Exp}(1/2 + 1/k) \overset{\leq}{=} \text{Exp}(1/2),$$

and that the $\{T(e)\}_{e \in E}$ are independent. Between any two consecutive hubs there are $k$ disjoint paths of length 2, so the communication time between them is stochastically dominated by $Z = \min\{Z_1, \ldots, Z_k\}$, where the $Z_i$ are independent random variables equal in distribution to the sum of two independent $\text{Exp}(1/2)$ random variables.

408
Lemma 1. We have $\mathbb{E}[Z] = O(1/\sqrt{k})$.

Proof. For any $t \geq 0$ we have

$$
P[Z > t] = \prod_i P[Z_i > t] = P[Z_1 > t]^k 
\leq \left(1 - P[\text{Exp}(1/2) \leq t/2]^2\right)^k 
= \left(2e^{-t/4} - e^{-t/2}\right)^k.
$$

Thus, using the inequality $2e^{-t/4} - e^{-t/2} \leq e^{-t/64}$ valid for $t \in [0, 4]$, we find

$$
\mathbb{E}[Z] = \int_0^\infty P[Z > t] \, dt 
\leq \int_0^4 e^{-kt/64} \, dt + \int_4^\infty (2e^{-t/4})^k \, dt 
\leq 8\sqrt{\pi/k} + \frac{2^k}{ke^k} = O(1/\sqrt{k}). \quad \square
$$

By Lemma 1, the expected time for all the hubs to learn the rumour is $O(mk^{-1/2})$. Once all the hubs learn the rumour, a degree 2 vertex pulls the rumour in $\text{Exp}(1)$ time. The expected value of the maximum of all $\text{Exp}(1)$ variables is $O(\log km)$. So by linearity of expectation, $\text{wast}_s(G) = O(\log n + mk^{-1/2})$.

In the synchronous case, for any $G$ we have $\text{wast}_s(G) \geq \text{diam}(G)$. For this graph, we get $\text{wast}_s(G) \geq 2m$. Choosing $k = \Theta((n/\log n)^{2/3})$ and $m = \Theta(n^{1/3}(\log n)^{2/3})$ gives

$$
\text{wast}_s(G) = O(\log n) \quad \text{and} \quad \text{wast}_s(G) = \Omega(n^{1/3}(\log n)^{2/3}).
$$

This graph has $\text{wast}_s(G)/\text{wast}_s(G) = \Omega((n/\log n)^{1/3})$ and is the example promised by Corollary 1.

3. EXTREME SPREAD TIMES

In this section we prove Theorems 1 and 2.

3.1 Proof of (2) and its tightness

For a given $t \geq 0$, consider the protocol which is the same as push\&pull except that, if the rumour has not spread to all vertices by time $t$, then the new process reinitializes. Coupling the new process with push\&pull, we obtain for any $k \in \{0, 1, 2, \ldots\}$ that

$$
P[\text{ST}_s(G, v) > kt] \leq P[\text{ST}_s(G, v) > t]^k.
$$

and

$$
P[\text{ST}_s(G, v) > kt] \leq P[\text{ST}_s(G, v) > t]^k.
$$

Combining (10) with

$$
P[\text{ST}_s(G, v) > e \mathbb{E}[\text{ST}_s(G, v)]] < 1/e,
$$

which comes directly from Markov’s inequality, we obtain

$$
P[\text{ST}_s(G, v) > e \log n \mathbb{E}[\text{ST}_s(G, v)] < 1/n.
$$

Since $\mathbb{E}[\text{ST}_s(G, v)] \leq \text{wast}_s(G)$ for all $v$, this gives the right-hand inequality in (2) directly from the definition of $\text{gst}_s$. This inequality is tight up to the constant factor, as the double star has $\text{wast}_s(DS_n) = O(n)$ and $\text{gst}_s(DS_n) = \Theta(n \log n)$ (see Section 2.4).

To prove the left-hand inequality, let $\tau = \text{gst}_s(G)$ and let $v$ be a vertex such that $\mathbb{E}[\text{ST}_s(G, v)] = \text{wast}_s(G)$. Then

$$
\text{wast}_s(G) = \int_0^\infty P[\text{ST}_s(G, v) > t] \, dt 
= \sum_{i=0}^{\infty} \int_{t^i}^{(i+1)t} P[\text{ST}_s(G, v) > t] \, dt 
\leq \sum_{i=0}^{\infty} \tau/n^i
$$

by (10) with $t = \tau$. Hence $\text{wast}_s(G) \leq \tau/(1 - 1/n)$. This inequality is tight up to a constant factor, as the star has $\text{wast}_s(G) = \Theta(\text{gst}_s(G)) = \Theta(\log n)$ (see Section 2.2).

3.2 Proof of the right-hand bound in (3) and its tightness

We will actually prove this using pull operations only. Indeed we will show $\text{wast}_s^{\text{pull}}(G) \leq 4n$, where the superscript pull means the ‘pull only’ protocol. Since the path has $\text{wast}_s^{\text{pull}}(P_n) \geq \text{wast}_s(P_n) = O(n)$ (see Section 2.3), this bound would be tight up to the constant factor.

The proof is by induction: we prove that when there are precisely $m$ uninformed vertices, just $b$ of which have informed neighbours (we call these $b$ vertices the boundary vertices), the expected remaining time for the rumour to reach all vertices is at most $4m - 2b$. The inductive step is proved as follows. Let $I$ denote the set of informed vertices, $B$ the set of boundary vertices, and $R$ the set of the remaining vertices. Let $|B| = b$ and $|B| + |R| = m$. Let $d(v)$ denote the degree of $v$ in $G$ and, for a set $S$ of vertices, let $d_S(v)$ count the number of neighbours of $v$ in $S$. We consider two cases.

Firstly, suppose that there exists a boundary vertex $v$ with $d_R(v) \geq d_B(v)$ (see Figure 2(Left)). We can for the next step ignore all calls from vertices other than $v$, so the process is forced to wait until $v$ is informed before any other vertices. This clearly gives an upper bound on the spread time. The expected time taken for $v$ to pull the rumour from vertices in $I$ is

$$
\frac{d(v)}{d_I(v)} = \frac{d_I(v) + d_R(v) + d_B(v)}{d_I(v)} 
\leq 1 + \frac{2d_R(v)}{d_I(v)} \leq 1 + 2d_R(v).
$$
Once \( v \) is informed, the number of uninform vertices decreases by 1, and the number of boundary vertices increases by \( d_R(v) - 1 \). The inductive hypothesis concludes this case since

\[
1 + 2d_R(v) + 4(m - 1) - 2(b + d_R(v) - 1) < 4m - 2b.
\]

Otherwise, if there is no such \( v \) (as in Figure 2(Right)), then any boundary vertex has a 'pulling rate' of

\[
\frac{d_I(v)}{d_I(v) + d_R(v) + d_B(v)} \geq \frac{1}{1 + d_R(v) + d_B(v)} \geq \frac{1}{2d_B(v)} \geq \frac{1}{2b}.
\]

Since there are \( b \) boundary vertices, together they have a pulling rate of at least 1/2, so the expected time until a boundary vertex is informed is at most 2. Once this happens, \( m \) decreases by 1 and \( b \) either does not decrease or decreases by 1, and the inductive hypothesis concludes the proof.

### 3.3 Proof of the left-hand bound in (3) and its tightness

In this section we show for any vertex \( v_0 \) of a graph \( G \) we have \( \mathbb{E}[\text{ST}_*(G, v_0)] = \Omega(\log n) \). This is tight as the star has \( \text{wast}_*(G^*) = O(\log n) \) (see Section 2.2). We give an argument for an equivalent protocol, defined below.

**Definition 4.** (Two-clock-per-edge protocol) On each edge place two exponential clocks, one near each end. All clocks are independent. On an edge joining vertices \( u \) and \( v \), the clocks both have rate \( \deg(u)^{-1} + \deg(v)^{-1} \). Note that this is the rate of calls along that edge, combined, from \( u \) and \( v \) (see Proposition 1). At any time that the clock near \( v \) on an edge \( uv \) rings, and \( v \) knows the rumour but \( u \) does not, the rumour is passed to \( u \).

**Lemma 2.** The two-clock-per-edge protocol is equivalent to the asynchronous push\&pull protocol.

**Proof.** Consider an arbitrary moment during the execution of the two-clock-per-edge protocol. Let \( I \) denote the set of informed vertices. For any edge \( uv \) with \( u \in I \) and \( v \notin I \), the rate of calls along \( uv \) is \( \deg(u)^{-1} + \deg(v)^{-1} \). Moreover, the edges act independently. So, the behaviour of the protocol at this moment is exactly the same as that of the asynchronous push\&pull protocol. Hence, the two protocols are equivalent.

In view of Lemma 2, we may work with the two-clock-per-edge protocol instead. Let \( X_c \) be the time taken for the first clock located near \( v \) to ring. Then \( X_c \) is distributed as \( \text{Exp}(f(v)) \) where \( f(v) = 1 + \sum \deg(u)^{-1} \), the sum being over all neighbours \( u \) of \( v \). Hence, \( \sum f(v) = 2n \).

On the other hand, for a vertex \( v \neq v_0 \) to learn the rumour, at least one of the clocks located near \( v \) must ring. Thus

\[
\max\{X_v : v \in V(G) \setminus \{v_0\}\} \leq \text{ST}_*(G, v_0).
\]

Let \( X = \max\{X_v : v \in V(G) \setminus \{v_0\}\} \). Hence to prove \( \mathbb{E}[\text{ST}_*(G, v_0)] = \Omega(\log n) \) it suffices to show \( \mathbb{E}[X] = \Omega(\log n) \).

Let \( \tau = \log(n - 1)/3 \) and \( A = V(G) \setminus \{v_0\} \). Then we have

\[
\mathbb{P}[X < \tau] = \prod_{u \in A} (X_u < \tau) = \prod_{u \in A} \left(1 - e^{-\tau f(u)}\right)
\]

\[
\leq \exp \left(-\sum_{u \in A} e^{-\tau f(u)}\right)
\]

\[
\leq \exp \left(-(n - 1)e^{-\tau} \sum_{v} f(v)/(n - 1)\right)
\]

\[
\leq \exp \left(-(n - 1)e^{-3\tau}\right) = e^{-1}.
\]

Here the first inequality follows from \( 1 - x \leq e^{-x^2} \), the second from the arithmetic-geometric mean inequality, and the last one from \( 2n = \sum f(v) \leq 3(n - 1) \) which holds for any \( n \geq 3 \). Consequently,

\[
\mathbb{E}[X] \geq \mathbb{P}[X \geq \tau] \geq (1 - e^{-1}) \log(n - 1)/3 = \Omega(\log n),
\]

as required.

### 3.4 Proof of (4) and its tightness

The bounds in (4) follow immediately from (2) and (3). The left-hand bound is tight as the star has \( \text{gst}_*(G^*) = \Theta(\log n) \) (see Section 2.2), and the right-hand bound is tight as the double star has \( \text{gst}_*(DS_n) = \Theta(n \log n) \) (see Section 2.4).

### 3.5 Proof of Theorem 2

The proof of (5) and its tightness are exactly the same as that for (2). The proof for (6) is similar to the one for the right-hand bound in (3) given in Section 3.2, and can be found in [1]. This gives \( \text{wast}_*(G) = O(n) \), which is tight up to the constant factor, as the path has diameter \( n - 1 \) and hence \( \text{wast}_*(P_n) \geq n - 1 \). Finally, the bound \( \text{gst}_*(G) = O(n \log n) \) is a direct consequence of (5) and (6), and it is tight as the double star has guaranteed spread time \( \Theta(n \log n) \) (see Section 2.4).
4. COMPARISON OF THE TWO PROTOCOLS

We first prove Corollary 1 assuming Theorems 3 and 4, and in the following subsections we prove these theorems. The left-hand bound in Corollary 1 follows from Theorem 3; it is tight, up to the constant factor, as the star graph has \( \text{gst}_4(G_n^*) = \Theta(n \log n) \) and \( \text{gst}_4(G_n^*) = 2 \) (see Section 2.2). The right-hand bound in Corollary 1 follows from Theorem 4 by choosing \( \alpha = 1/3 \). A graph \( G \) was given in Section 2.5 having \( \text{wast}_4(G) / \text{wast}_4(G) = \Omega \left( \left( n / \log n \right)^{1/3} \right) \). Using (2) and (5), we get \( \text{gst}_4(G) / \text{gst}_4(G) = \Omega \left( n^{1/3} (\log n)^{-4/3} \right) \) for this \( G \).

4.1 The lower bound

In this section we prove Theorem 3. Let \( G \) be an \( n \)-vertex graph and let \( s \) denote the vertex starting the rumour. We build a coupling between the two versions. Consider a ‘collection of calling lists for vertices’: for every vertex \( u \), we have an infinite list of vertices, each entry of which is a uniformly random neighbour of \( u \), chosen independently from other entries. The coupling is built by using the same collection of calling lists for the two versions of the push&pull protocol. Note that \( ST_4(G, s) \) is determined by this collection, but to determine \( ST_4(G, s) \) we also need to know the Poisson processes associated with the vertices.

Let \( B \) denote the event ‘\( ST_4(G, s) \leq 2 \text{gst}_4(G) \)’, which depends on the calling lists only. Inequality (11) gives \( \Pr[B] \leq 1/n^2 \). (Here, \( B^c \) denotes the complement of \( B \).) Partition the time interval \([0, 2 \text{gst}_4(G) \times 4 \log n] \) into subintervals \([0, 4 \log n], [4 \log n, 8 \log n], \ldots \). Consider a ‘decelerated’ variant of the asynchronous push&pull protocol in which each vertex makes a call the first time its clock rings in each subinterval (if it does), but ignores later clock rings in that subinterval (if any). The spread time in this protocol is stochastically larger than that in the asynchronous push&pull protocol, so without loss of generality we may and will work with the decelerated variant. Let \( A \) denote the event ‘during each of these 2 \( \text{gst}_4(G) \) subintervals, all clocks ring at least once.’ If \( A \) happens, then an inductive argument gives that for any \( 1 \leq k \leq 2 \text{gst}_4(G) \), the set of informed vertices in the decelerated variant at time \( 4k \log n \) contains the set of informed vertices after \( k \) rounds of the synchronous version. Hence, if both \( A \) and \( B \) happen, then we would have

\[
ST_4(G, s) \leq (4 \log n) ST_4(G, s) \leq (8 \log n) \text{gst}_4(G) \, .
\]

Hence to complete the proof, we need only show that \( \Pr[A^c] < 1/n - 1/n^2 \).

Let \( I \) denote a given subinterval of length \( 4 \log n \). In the asynchronous version, the clock of any given vertex rings with probability at least \( 1 - n^{-4} \) during \( I \). By the union bound, all clocks ring at least once during \( I \), with probability at least \( 1 - n^{-3} \). The number of subintervals in the definition of \( A \) is \( 2 \text{gst}_4(G) \), which is \( O(n \log n) \) by (7). By the union bound again, \( \Pr[A^c] = O(\log n / n^2) \), as required.

Theorem 3 is tight, up to the constant factor, as the star has \( \text{gst}_4(G_n^*) = \Theta(n \log n) \) and \( \text{gst}_4(G_n^*) = 2 \) (see Section 2.2).

4.2 The upper bound

The proof of Theorem 4 can be found in [1]. Here we sketch the proof. The main ingredients in the proof are a coupling between the two protocols, and sharp concentration bounds. Consider the asynchronous version. List the vertices in the order their clocks ring. The list ends once all the vertices are informed. Now consider the natural coupling between the two protocols, and the asynchronous actions follow the same ordering as in the list. We partition the list into blocks according to a certain rule in such a way that the blocks have the following property: the synchronous protocol in each round will inform a superset of the set of vertices informed by the asynchronous variant in any single block. For example, if we require that in each block each vertex communicates with the others at most once, then we would have this property. However, in order to get our bound, we need to use a more delicate rule for building the blocks. To conclude, we find an upper bound for the number of blocks, which coincides with the right-hand side of (8).

5. ACKNOWLEDGEMENTS

Andrea Collevecchio was supported by ARC Discovery Project grant DP140100559 and STREP project MATH-EMACS; Abbas Mehrabian was supported by the Vanier Canada Graduate Scholarships program; most of this work was done while he was visiting Monash University. Nick Wormald was supported by Australian Laureate Fellowships grant FL120100125.

6. REFERENCES

[1] H. Acan, A. Collevecchio, A. Mehrabian, and N. Wormald, On the push&pull protocol for rumour spreading. arXiv, 1411.0948 [cs.DC], 2014.
[2] H. Amini, M. Draief, and M. Lelarge, Flooding in weighted sparse random graphs. SIAM J. Discrete Math., 27(1):1–26, 2013.
[3] N. Berger, C. Borgs, J. Chayes, and A. Saberi, On the spread of viruses on the Internet. In Proc. 16th Symp. Discrete Algorithms (SODA), pages 301–310, 2005.
[4] B. Bollobás and Y. Kohayakawa, On Richardson’s model on the hypercube. In Combinatorics, geometry and probability (Cambridge, 1993), pages 129–137. Cambridge Univ. Press, Cambridge, 1997.
[5] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, Randomized gossip algorithms. IEEE Transactions on Information Theory, 52(6):2508–2530, 2006.
[6] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, Epidemic algorithms for replicated database maintenance. In Proc. 6th Symp. Principles of Distributed Computing (PODC), pages 1–12, 1987.
[7] B. Doerr, M. Fouz, and T. Friedrich, Social networks spread rumors in sublogarithmic time. In Proc. 43th Symp. Theory of Computing (STOC), pages 21–30, 2011.
[8] B. Doerr, M. Fouz, and T. Friedrich, Asynchronous rumor spreading in preferential attachment graphs. In Proc. 13th Scandinavian Workshop Algorithm Theory (SWAT), pages 307–315, 2012.
[9] B. Doerr, M. Fouz, and T. Friedrich, Experimental analysis of rumor spreading in social networks. In Design and analysis of algorithms, volume 7659 of Lecture Notes in Comput. Sci., pages 159–173. Springer, Heidelberg, 2012.
[10] R. Durrett. Stochastic growth models: recent results and open problems. In *Mathematical approaches to problems in resource management and epidemiology (Ithaca, NY, 1987)*, volume 81 of *Lecture Notes in Biomath.*, pages 308–312. Springer, Berlin, 1989.

[11] U. Feige, D. Peleg, P. Raghavan, and E. Upfal. Randomized broadcast in networks. *Random Struct. Algorithms*, 1(4):447–460, 1990.

[12] J. A. Fill and R. Pemantle. Percolation, first-passage percolation and covering times for Richardson's model on the $n$-cube. *Ann. Appl. Probab.*, 3(2):593–629, 1993.

[13] N. Fountoulakis, K. Panagiotou, and T. Sauerwald. Ultra-fast rumor spreading in social networks. In *Proc. 23th Symp. Discrete Algorithms (SODA)*, pages 1642–1660, 2012.

[14] G. Giakkoupis. Tight bounds for rumor spreading in graphs of a given conductance. In *28th International Symposium on Theoretical Aspects of Computer Science (STACS 2011)*, volume 9, pages 57–68, 2011.

[15] G. Giakkoupis. Tight bounds for rumor spreading with vertex expansion. In *Proc. 25th Symp. Discrete Algorithms (SODA)*, pages 801–815, 2014.

[16] M. Harchol-Balter, F. T. Leighton, and D. Lewin. Resource discovery in distributed networks. In *Proc. 18th Symp. Principles of Distributed Computing (PODC)*, pages 229–237, 1999.

[17] S. M. Hedetniemi, S. T. Hedetniemi, and A. L. Liestman. A survey of gossiping and broadcasting in communication networks. *Networks*, 18(4):319–349, 1988.

[18] C. D. Howard. Models of first-passage percolation. In *Probability on discrete structures*, volume 110 of *Encyclopaedia Math. Sci.*, pages 125–173. Springer, Berlin, 2004.

[19] S. Janson. One, two and three times log $n/n$ for paths in a complete graph with random weights. *Combin. Probab. Comput.*, 8(4):347–361, 1999.

[20] R. Karp, C. Schindelhauer, S. Shenker, and B. Vöcking. Randomized Rumor Spreading. In *Proc. 41st Symp. Foundations of Computer Science (FOCS)*, pages 565–574, 2000.

[21] D. Kempe, A. Dobra, and J. Gehrke. Gossip-based computation of aggregate information. In *Proc. 44th Symp. Foundations of Computer Science (FOCS)*, pages 482–491, 2003.

[22] K. Panagiotou and L. Speidel. Asynchronous rumor spreading on random graphs. In L. Cai, S.-W. Cheng, and T.-W. Lam, editors, *Algorithms and Computation*, volume 8283 of *Lecture Notes in Computer Science*, pages 424–434. Springer Berlin Heidelberg, 2013.