Competitive Analysis of Organization Networks Or Multicast Acknowledgement: How much to wait?

Carlos Brito[1], Elias Koutsoupias[1,2⋆], and Shailesh Vaya[1]

[1] University of California Los Angeles
fisch@cs.ucla.edu, vaya@cs.ucla.edu
[2] University of Athens
elias@di.uoa.gr

Abstract. We study, from the competitive analysis perspective, the trade off between communication cost and delay cost (or simply the send-or-wait dilemma) on a hierarchy (rooted tree). The problem is an abstraction of the message aggregation problem on communication networks and the organizational problem in network hierarchies. We consider the most natural variant of the problem, the distributed asynchronous regime, and give tight (within an additive constant) upper and lower bounds of the competitive ratio.
We also consider the centralized version of the problem, where we combine the natural rent-to-buy strategy with prediction techniques to achieve the first constant competitive ratio algorithm for any non-trivial class of network topologies.

1 Introduction

In [2], Papadimitriou and Schreiber introduce a model for communication in organization networks. An organization network is modeled as a rooted tree in which the nodes represent employees of the organization. The root node represents the leader who is also the decision-maker of the organization. Messages arrive from external sources to the nodes, containing information about the world outside the organization. This information must be send to the root, so that the leader has an accurate view of the world to make decisions. Each employee can only communicate with its immediate supervisor in the hierarchy. There is a cost involved in the communication, which is *independent of the amount of information* being conveyed. For simplicity the communication is assumed to be instantaneous, but employees may decide to hold messages, waiting for more information to arrive, in order to reduce the communication cost. However, a delay cost is incurred for each piece of information that is being held by an employee.

⋆ Research supported in part by NSF and the IST Program.

This cost models the penalty imposed to the organization, as a result of the decision-maker not being up-to-date with the world view. The delay cost may depend on many factors, but a reasonable simple model is to assume that *the delay cost is proportional to the number of messages and the elapsed time*. The apparent trade off between communication cost and delay cost is a fundamental problem in decision-making: How much should each node wait before delivering a message, such that, the total cost incurred by the organization is minimized?

A similar trade off appears in the Multicast Acknowledgment Aggregation problem [1]. A multicasting protocol sends data from a sender to a group of receivers in a communication network. For efficiency, instead of sending a separate message to each individual receiver, a single message is sent along the multicast tree that spans the entire group of receivers. If the underlying network is unreliable, packets may be dropped. A possible solution to improve the reliability under these circumstances, is to have the receivers acknowledge (ACK) each packet correctly received [9]. However, this mechanism can introduce a significant communication overhead. Since the acknowledgment packets are small, a reasonable approach is to delay their transmission and aggregate several ACKs in the same message. This delay, however, may impact the overall performance of the protocol. The question again becomes: How much should each node in the network wait before delivering the ACKs such that the cost incurred by the network is minimized?

1.1 Problem Formulation.

The two optimization problems discussed above share the following common abstraction:

Let T be a directed rooted weighted tree, such that all the edges are directed towards the root node of the tree. Consider an arbitrary arrival of sequence of packets at the (leaf or internal) nodes of the tree. All the packets must be sent to the root of T. Two costs are incurred in this process. The delay cost: when a packet is delayed at some node of the tree a delay cost proportional to the waiting time incurred. The communication cost: when a set of packets is delivered along an edge a delivery cost equal to the weight of the edge (and independent of the number of packets) is incurred. The total cost is the sum of the delay cost for each packet plus the communication cost of all deliveries.

The associated online problem then is to try to minimize the total cost for an unknown sequence of packet arrivals. Papadimitriou and Schreiber in [2], consider the following hierarchy of availaibility of information to the nodes of the network:

- **The Asynchronous Regime:** Each node acts in isolation of other nodes i.e., there is no way for the nodes to coordinate amongst themselves.
- **The Synchronous Regime:** A global clock is provided to all the nodes allowing them to synchronize their actions. It is easily noted that this information is useful only if the packets can only be introduced in the network at the beginning of a time slot.

– **The Full Information Regime:** Each node has complete knowledge of the arrival of packets at all the other nodes in the tree.

Clearly, for the Acknowledgment Aggregation Problem the most appropriate regime is the first one. However, the other two regimes are reasonable abstractions for send-or-wait situations. The hardest to defend is clearly the Full Information regime which essentially assumes a centralized online algorithm: How can we assume a central entity when the communication cost is part of the problem? However, there are two distinct types of information here; the centralized algorithm doesn't need to know the nature of the information at the various nodes, only the size of it. For example, it is one thing to interrupt your supervisor to deliver a piece of information, and a completely different thing to let someone else (the centralized online algorithm or her secretary) know that you have something valuable to discuss with her.

1.2 Related Work.

The special case when the network has only two nodes was analyzed by Dooly, Goldman and Scott, [3]. The authors show that a simple algorithm based on a rent-to-buy strategy [7] has an optimal competitive ratio 2.

For shallow trees of depth 2 and Poisson arrival process on packets upper bounds are proved for all three regimes in [2]. Also, simulation experiment results are provided to compare the efficiency of the algorithms in the three regimes.

In [1], Khanna, Naor, Raz consider the problem for general trees for both Full Information Regime and Synchronous Regime. For the first case, they analyze a natural algorithm based on rent-to-buy strategy. Then, the algorithm is shown to be $O(log\alpha)$-competitive where α is the sum of costs assigned to all the edges in the network. For the analysis of the Synchronous Regime, it is assumed that all the events in the problem, i.e., packet arrivals and deliveries, must wait at least one unit of time upon arrival before it can be delivered. These restrictions are made for both the online and offline algorithms. An oblivious algorithm for this model is provided, where nodes do not have knowledge about the topology, or their particular location in the network and is shown to be $O(h \cdot log\alpha)$-competitive, where h is the height of the tree.

1.3 Overview of our Results.

In this paper we study the problem for both the Asynchronous and Full Information Regimes.

For Asynchronous Regime we express our bounds in terms of a quantity $\chi^*(T)$ which precisely captures the intricacies of the problem for an distributed asynchronous system. As we show in Appendix A, when all edge costs are 1, $\chi^*(T)$ is Θ(height of T), but in the general case, it can take any value between 1 and the cost of the most expensive path in T. Using $\chi^*(T)$ we give almost tight upper and lower bounds. More precisely we show that the competitive ratio is between $\chi^*(T)$ and $\chi^*(T) + 1$. We can further tighten the result for the case of *memoryless* distributive algorithm: in this case the competitive ratio is between $\chi^*(T) + \frac{\chi^*(T)}{\chi^*(T)+1}$ and $\chi^*(T)+1$. Previously, the lower bounds in this regime were known only for the trivial case of two nodes, by [3].

Using Interleaving lemma we can prove a lower bound of $X^*(\tau) + 1$ on the competitive ratio of an arbitrary distributed asycnchronous algorithm. Depending on the topology of the tree, this ratio can be far from a constant. For example, for a serial network, with h nodes in a series, with weight one assigned to each edge of the network, there exists a lower bound of $\Omega(h)$ on the competitive ratio of the distributed algorithm. We wonder whether the dependence of the competitive ratio on the topology of the network is because of the lack of coordination amongst the nodes in the network i.e., distributed nature of the problem or because of lack of temporal information about the arrival of packets in the network i.e., the online nature of the problem. Suppose, there is a centralized agency that receives information about the arrival of packets in the entire network. Further, assume that this agency can also make decisions and coordinate the delivery of packets in the network. A study of competitive ratio of the optimization problem under consideration with an assumption of a centralized control i.e., full information regime, may provide an insight into where the trade-off really lies. Infact, both [2], [1] studied the centralized as well as the distributed version of the optimization problem. However, the results in [2], [1] are not tight enough to understand the trade-off.

We consider a serial network with n nodes. For this network we show a constant competitive ratio algorithm indeed exists if there is a centralized coordination mechanism available to the nodes in the network. Considering this result in consonance with the linear (in terms of number of nodes in the network) lower bound on the competitive ratio of the distributed version of the problem we can see that the important trade off is because of the lack of coordination between the nodes in the network i.e., the distributed nature of the problem rather then the lack of availability of future arrival times of the packets.

Almost, all the results on this problem [2], [1], including ours for the distributed version of the problem have been based on the classical rent-to-buy strategy, [7]. We also consider the rent-to-buy strategy for the full information regime, however an interleaving sequence is easily constructed for which this strategy does not obtain any significant advantage even if the centralized coordination mechanism is available. However, if the rent-to-buy strategy is combined with prediction techniques then a constant competitive ratio algorithm indeed exists for the serial network.

The constant competitive algorithm for the full information regime for shallow networks requires a sophisticated prediction technique. Due to shortage of space we just state the algorithm on the shallow network topology and refer the reader to full version of the paper for detailed proof of competitiveness.

2 Basic Definitions and Notation

We are given a directed rooted tree T with root r. Each edge e in T has an associated cost c_e. If $\mathbf{p_i}$ is the path between node n_i and the root of T, then let $\mathbf{weight}(\mathbf{p_i})$ denote the sum of the costs associated to each edge in p_i.

Let $\mathbf{\Psi} = \{p_1, \ldots, p_n\}$ be a set of paths and $\mathbf{\Psi_t}$ be the subtree formed by the edges that appear in one of the paths p_i, $p_i \in \Psi$.

Similarly, let τ [1] be a subtree of T. Define $\tau_\mathbf{p}$ be the set of paths from the nodes in τ to the root of the tree T. Now, extend the definition of the function *cost* so that $\mathbf{weight}(\tau)$ denotes the sum of the weights of the edges in the subtree τ. Let $|\tau|$ denote the number of nodes in subtree τ.

A *local online algorithm* Θ_i controls the deliveries performed by a node n_i based only on the internal state of n_i. Let $\rho = \langle \alpha_\mathbf{1}, \ldots, \alpha_\mathbf{l} \rangle$ be an arrival sequence to node n_i, where $\alpha_\mathbf{i}$ gives the number of packets in the i^{th} message of ρ. Then, the total cost incurred by $\mathbf{\Theta_i}$ to handle the packets in ρ is given by:

$$k \cdot c_i + \sum_{j=1}^{l} d_j \cdot \alpha_j$$

where k is the number of distinct deliveries performed by Θ_i, c_i is the cost associated with the outgoing edge of n_i, and d_j is the amount of delay incurred by the α_j packets before they are delivered by Θ_i.

We say that a local online algorithm Θ_i is *memoryless* if it resets itself after each transmission (whenever its memory is empty).

We define a *distributed online strategy* Σ to be a collection of local online algorithms assigned to the nodes in the network. For a given arrival process ρ, the cost incurred by Σ to handle the packets in ρ is given by the sum of the costs incurred by each of the local online algorithms executed by the nodes of the network. A distributed online strategy Σ is *memoryless* if all the local online algorithms assigned to the nodes are memoryless.

A *centralized online algorithm* $\mathcal{A}$ controls the delivery performed by every node in the network based on the knowledge of the internal state of each node.

Definition: Let Ψ be a set of paths and Ψ_t be the tree formed by the edges included in this path. Define, the function

$$\chi(\mathbf{\Psi_t}) = \frac{\sum_{\mathbf{p_i} \in \Psi} \mathbf{weight}(\mathbf{p_i})}{\mathbf{weight}(\mathbf{\Psi_t})}$$

For χ^* we consider all possible subtrees of the tree.

Definition: Let $\mathcal{S_T}$ be the set of all subtrees of T, then the function χ^* is defined for T as:

$$\chi^*(\mathbf{T}) = \max_{\tau \in \mathcal{S_T}} \chi(\tau)$$

[1] Whenever not mentioned explicitly, the root of a subtree τ of T is the same as the root of T

3 A cruel adversary for the Asynchronous Regime

In this section, we prove an important lemma, called the Interleaving Lemma which is the main ingredient for proving the lower bound on the competitive ratio of any deterministic algorithm for asynchronous regime.

The *Interleaving Lemma* basically states that, for any reasonable distributed online strategy, there exists a *cruel adversary* which can always devise a sequence of packet arrivals that forces every intermediate node in the network to make a separate delivery for each message that it gets from each of its children. However, an offline algorithm is able to handle the same arrivals with minimal delivery cost (one message per link of the network) and negligible delay cost. This implies that delaying cannot be used, in an effective way, to merge messages from distinct nodes, and so the deliveries end up being interleaved on the network.

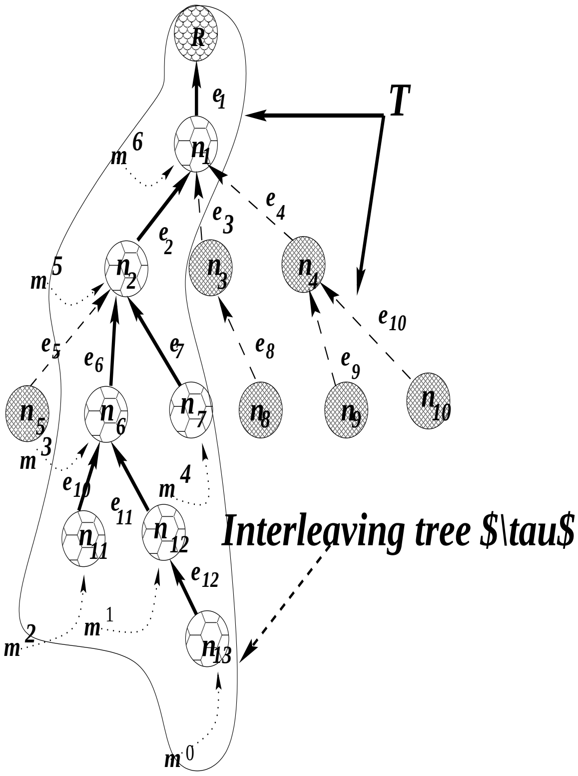

Fig. 1. Interleaving on a subtree

The intuition behind the interleaving lemma is appropriately exposed by the figure 4. The subtree τ, formed by the nodes $\{n_1, n_2, n_6, n_7, n_{11}, n_{12}, n_{13}\}$, is used for interleaving. The adversary provides $m^6, m^5, m^3, m^4, m^2, m^1, m^0$ packets to the nodes $n_1, n_2, n_6, n_7, n_{11}, n_{12}, n_{13}$ respectively. m is a large integer. Since, we are considering distributed paradigm, no node has any knowledge about the status of other nodes. If the nodes are implementing a memoryless strategy it can be seen that the 1) a separate delivery is made for the packets waiting at each of the nodes 2) the order in which the nodes make the deliveries are $n_1, n_2, n_6, n_7, n_{11}, n_{12}, n_{13}$. The total cost paid by any online algorithm is atleast $weight(\tau_p)$. The adversary though is cruel and delivers all the messages at once, thus incurring a much smaller cost of $weight(\tau)$. His advantage over the online algorithm becomes $\frac{weight(\tau_p)}{weight(\tau)} = \chi(\tau)$.

3.1 The Interleaving Lemma.

Now, we shall develop the intuition described above into a rigorous claim, the Interleaving Lemma. First, we shall capture the behavior of an arbitrary deterministic algorithm formally.

A deterministic online algorithm Θ, keeps an internal state s based on which it makes decisions. The following two functions capture the behavior of a deterministic algorithm Θ_i, assigned to node n_i, at some specific instants of time, and are sufficient to prove our results.

- $T_i(\alpha, s)$ gives the amount of time Θ_i delays the next delivery if α packets arrive when Θ_i is in state s, and no further arrival occurs;
- $S_i(\alpha, s)$ gives the state of Θ_i after the next delivery, if α packets arrive when Θ_i is in state s, and no further arrival occurs.

We say that a state s_0 of Θ_i is *vulnerable* if one of the following holds:

1. there exists a sequence $\rho = \langle \alpha_1, \ldots, \alpha_k \rangle$, such that, for $j = 1, \ldots, k$, $T_i(\alpha_j, s_{j-1}) = 0$, where $s_j = S_i(\alpha_j, s_{j-1})$, for arbitrarily large k;
2. $\exists \epsilon > 0$ such that $\forall \alpha_0 \; \exists \alpha > \alpha_0 : T_i(\alpha, s_0) \geq \epsilon$

Lemma 1. *Let Σ be an online strategy that assigns algorithm Θ_i to node n_i. If Θ_i has a vulnerable state s_0, which an adversary can force Θ_i to reach with finitely many packet arrivals, then the competitive ratio of Σ is unbounded.*

Proof. We construct a sequence ρ of packet arrivals to n_i consisting of two parts. The first part is a sequence that forces Θ_i into state s_0. Let C^* denote the cost incurred by Σ to handle the first part of ρ.

Now, assume that state s_0 satisfies condition (1) above. Then, the second part of ρ is just a sequence of messages $\langle \alpha_1, \ldots, \alpha_k \rangle$, for arbitrarily large k. The messages in ρ arrive arbitrarily close to each other, but the values α_i are chosen such that Θ_i performs k separate deliveries. For each of these deliveries, Σ must pay at least the cost of the outgoing link of n_i, say c. Then, Σ incurs cost at least $C^* + k \cdot c$ to handle the arrival sequence ρ.

Consider an offline algorithm that has the same behavior as Θ_i for the first part of ρ, but collects all packets in the second part and makes a single delivery.

The delay cost incurred with the second part of ρ is negligible, and the delivery cost is just $weight(p_i)$. Hence, the competitive ratio of Σ is at least

$$\frac{C^* + k \cdot c}{C^* + weight(p_i)}$$

which can be made arbitrarily large by appropriate choice of k.

Now, assume that s_0 satisfies condition (2) of a vulnerable state. Then, the second part of ρ is just a message with β packets, for arbitrarily large β. Since Θ_i delays the delivery of the β packets for a fixed time $\varepsilon > 0$, the total cost incurred by Σ with ρ is at least $C^* + \beta \cdot \varepsilon$. An offline algorithm that has the same behavior as Θ_i for the first part of ρ, and delivers packets β as soon as they arrive, witnesses that Σ has competitive ratio at least

$$\frac{C^* + \beta \cdot \varepsilon}{C^* + weight(p_i)}$$

which is unbounded, since β is arbitrary and ε fixed.

Lemma 2 (Interleaving Lemma). *Let Σ be a deterministic distributed online strategy. Then, one of the following must hold:*

1. *Σ has unbounded competitive ratio; or,*
2. *there exist arrival sequences $\langle \beta_1^{(i)}, \ldots, \beta_{l_i}^{(i)} \rangle$, one for each node n_i, such that all packets in all sequences arrive in an arbitrarily small interval; but, for $i \neq j$, packets from $\beta_{l_i}^{(i)}$ and $\beta_{l_j}^{(j)}$ are never merged into the same message.*

Proof. We may assume that the algorithms assigned by Σ to the nodes in the network do not have reachable vulnerable states, otherwise (1) holds and the lemma follows.

Let n_i and n_j be two nodes in the network such that n_i is not on the path from n_j to root. Let Θ_i and Θ_j be the algorithms assigned by Σ to n_i and n_j, respectively.

Since Θ_i has no vulnerable state, we can find a sequence of arrivals $\langle \beta_1^{(i)}, \ldots, \beta_{l_i}^{(i)} \rangle$, such that Θ_i delivers $\beta_1^{(i)}, \ldots, \beta_{l_i-1}^{(i)}$ immediately, but delays the delivery of $\beta_{l_i}^{(i)}$ by some positive interval Δt. Similarly, we can find a sequence $\langle \beta_1^{(j)}, \ldots, \beta_{l_j}^{(j)} \rangle$ that implies the same behavior for Θ_j.

Now, since no algorithm assigned to the nodes on the path from n_j to root has a vulnerable state, we can choose $\beta_{l_j}^{(j)}$ large enough so that the packets arrived at n_j reach the root in less than Δt units of time, that is, before Θ_i delivers packets $\beta_{l_i}^{(i)}$. Thus, packets from $\beta_{l_i}^{(i)}$ and $\beta_{l_j}^{(j)}$ are never delivered in the same message.

To obtain the set of arrival sequences described in (2), we just enumerate the nodes in a bottom-up way, and choose appropriate sequences of arrivals according to the method above. Thus, if n_i appears before n_j in the enumeration, all packets arrived at n_j reach the root before n_i delivers the packets $\beta_{l_i}^{(i)}$.

4 Lower Bounds for Asynchronous Regime

The next result follows directly from the Interleaving Lemma:

Theorem 1. *The competitive ratio of any distributed online strategy Σ on a tree T is at least $\chi^*(T)$.*

Proof. Let τ be the subtree of T such that $\chi(\tau) = \chi^*(T)$. Interleaving on this subtree using the interleaving sequence given by gives the desired bound.

4.1 Memoryless Algorithms.

For memoryless strategies we shall show a simple result called *Waiting Lemma* to obtain a slightly higher lower bound. The Waiting Lemma shows that some minimal amount of delay is still required, otherwise the competitive ratio of the distributed online strategy can become unbounded. Waiting Lemma applies only to memoryless strategies, while the Interleaving Lemma holds for every deterministic distributed online strategy.

Lemma 3 (Waiting Lemma). *Let Σ be a memoryless distributed online strategy. Let $\varepsilon > 0$. Assume that there exists a path p_j from node n_j to root such that:*

1. *$weight(p_j) = d$;*
2. *$\forall \alpha_0\ \exists \alpha > \alpha_0$ such that if α packets arrive at n_j and all nodes in p_j are empty, then the α packets reach the root before Σ accumulates delay cost ε.*

Then, the competitive ratio of Σ is at least d/ε.

Proof. Consider a sequence of packet arrivals $\rho = \langle \alpha_0, \dots, \alpha_k \rangle$ satisfying:

1. for $i = 1, \dots, k$, if α_i packets arrive at n_j, then they reach the root before Σ accumulates waiting cost ε;
2. $(\alpha_{i-1})^2 \leq \alpha_i$, $i = 2, \dots, k$;
3. the α_i packets arrive immediatelly after packets α_{i-1} reach the root.

The online strategy Σ performs a separate delivery for each arrival in ρ, at a cost of at least $k \cdot d$.

Now, consider an offline algorithm which waits until all packets in ρ arrive, and then makes a single delivery. The sequence ρ was defined such that the waiting cost incurred by this algorithm is approximately $k \cdot \varepsilon$. Thus, the competitive ratio of Σ is at least $\frac{kd}{k \cdot \varepsilon + d} = \frac{d}{\varepsilon + d/k}$. Since k is arbitrary, the result follows.

Theorem 2. *The competitive ratio of a* **memoryless** *distributed online strategy Σ on a tree T is at least $\chi^*(T) + \frac{\chi^*(T)}{\chi^*(T)+1}$.*

Proof. Let n_i be an arbitrary node of T, and let p_i denote the path from n_i to root. Then, we may assume that there exists $\alpha_i > 0$ such that, if $\alpha > \alpha_i$ packets arrive at n_i and all the nodes in p_i are empty, then Σ accumulates delay cost at least $\frac{weight(p_i)}{\chi^*(T)+1}$ before the packets reach the root. If this is not the case, then lemma 3 gives that the competitive ratio of Σ is at least $weight(p_i)/\left(\frac{weight(p_i)}{(\chi^*(T)+1)}\right) = \chi^*(T) + 1$, and the result follows.

Now, let $\alpha^* = max_i\{\alpha_i\}$. Let τ be a subtree of T such that $\chi^*(T) = \chi(\tau)$. Let ρ be a sequence of arrivals to the nodes of τ given by the Interleaving Lemma [2] , such that each arrival includes at least α^* packets. Then, Σ pays $\sum_{n_i \in \tau} weight(p_i)$ to deliver the packets, and incurs cost at least $\sum_{n_i \in \tau} \frac{weight(p_i)}{(\chi^*(T)+1)}$ by delaying the packets. On the other hand, an offline algorithm that delivers all the packets immediately as they arrive, incurs negligible delay cost plus $weight(\tau)$ for delivery. Hence, Σ has competitive ratio at least $\chi(T) + \frac{\chi(T)}{\chi^*(T)+1}$.

5 Upper Bounds for Asynchronous Regime

To obtain an upper bound for distributed online strategies, we first define the following generic online algorithm for a node of the tree:

Online Algorithm Schema k-wait: A node n_i implements algorithm **k-wait** if it delays the delivery of packets arriving from the external source until it accumulates k units of delay cost. At this point, n_i delivers all its packets. Packets arrived from other nodes in the network are delivered immediately upon arrival, incurring no delay cost.

Theorem 3. *The online distributed strategy Σ that assigns algorithm $\left(\frac{weight(p_i)}{\chi^*(T)}\right)$-* **wait** *to each node n_i of an arbitrary tree T, has competitive ratio at most $\chi^*(T) + 1$.*

Proof. Consider an arbitrary sequence of packet arrivals, and fix an optimal algorithm $\mathcal{O}$ for such arrival process. Let $D = \langle \delta_1, \delta_2, \ldots, \delta_l \rangle$ be the sequence of delivery events generated by $\mathcal{O}$ under these arrivals.

For each delivery δ_i, let τ_i denote the subtree consisting of the nodes included in δ_i.

Fix a delivery δ_i and a node $n_j \in \tau_i$. Let p_j be the path from n_j to root. Denote by α_{i_j} the set of packets delivered by δ_i that arrived at n_j.

Let w_{i_j} denote the delay cost accumulated by $\mathcal{O}$ with packets α_{i_j}. Then, the total cost incurred by $\mathcal{O}$ with the packets in delivery δ_i is given by

$$weight(\tau_i) + \sum_j w_{i_j}$$

Now, we estimate the cost incurred by Σ to handle the same packets. Since node n_j only performs a delivery when the delay cost accumulated by its packets

[2] For memoryless strategies, an Interleaving sequence can be composed with just a single message per node

reach $weight(p_j)/\chi^*(T)$, the packets α_{i_j} can be distributed in at most $\left\lceil \frac{w_{i_j}}{weight(p_j)/\chi^*(T)} \right\rceil$ distinct deliveries. For each of these deliveries Σ pays

$$weight(p_j) + weight(p_j)/\chi^*(T) = weight(p_j)\left(\frac{\chi^*(T)+1}{\chi^*(T)}\right)$$

Thus, the total cost incurred by Σ with the packets in delivery δ_i is at most:

$$\sum_{n_j \in \tau_i} \left[\frac{w_{i_j}}{weight(p_j)/\chi^*(T)} + 1\right] \cdot \left[weight(p_j) \cdot \left(\frac{\chi^*(T)+1}{\chi^*(T)}\right)\right]$$

So, the ratio between the costs of Σ and $\mathcal{O}$ for the packets in d_i is at most:

$$\left(\frac{\chi^*(T)+1}{\chi^*(T)}\right) \cdot \frac{\sum_j \left[\frac{w_{i_j} \cdot \chi^*(T)}{weight(p_j)} + 1\right] \cdot weight(p_j)}{weight(\tau_i) + \sum_j w_{i_j}}$$

$$\leq \left(\frac{\chi^*(T)+1}{\chi^*(T)}\right) \cdot \left[\frac{\sum_j w_{i_j} \cdot \chi^*(T)}{\sum_j w_{i_j}} + \frac{\sum_j weight(p_j)}{weight(\tau_i)}\right]$$

But observing that the second term inside the brackets is $\chi(\tau_i) \leq \chi^*(T)$, we get that the entire expression is smaller than or equal to $\chi^*(T)+1$. Since δ_i is an arbitrary delivery, the result follows.

6 Full Information Regime

In this section we explore competitive algorithms for special topoligies when centralized coordination is available to the nodes of the network.

We first consider the serial network topology as demonstrated by figure 2. For this network, we show that a standard application of rent-to-buy technique doesn't give a constant competitive algorithm. Next, we show that by paying an extra cost by delivering packets for a larger segment of the network provides an insurance to the online algorithm that it can use in future to balance the offline algorithm. Another way to view this approach is predicting the behavior of the offline algorithm based on the current state of the network. For the serial network, the application of the prediction technique is straight forward and limited by the peculiarity of the network topology. Shallow networks, 3, requires a not so straightforward prediction of the behavior of the offline algorithm. For shallow networks, we compute deadlines uptill which the nodes in the network can wait on the packets.

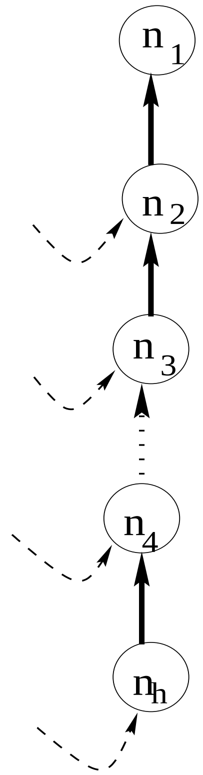

Fig. 2. A Serial Network

6.1 Rent-to-buy strategy is not constant competitive.

The rent-to-buy strategy applied to the serial network, 2 is as follows: *When the delay cost incurred for a set of packets in some segment of the network is equal to the delivery cost for delivering the packets of that segment, then deliver the packets.*

Consider the following sequence of arrivals of packets in the network: node 0 is given m^h packets, node 1 is given m^{h-1} packets, node h is given $m^0 = 1$ packets, where m is a very large number at time t. The online algorithm that applies the rent-to-buy strategy for this arrival sequence makes a separate delivery for each set of packets waiting at nodes $0, 1, 2, \ldots, h$, thereby incurring a delivery cost of $\frac{h(h+1)}{2}$. The adversary delivers all the packets at once incurring a total cost of h. This makes the online algorithm at least $\frac{h+1}{2}$ competitive.

6.2 Serial networks

We use look ahead along with rent-to-buy to design an effecient algorithm for this topology. The online algorithm still makes a delivery whenever the delay cost accumulated on a subtree τ reaches $weight(t)$. But now, the algorithm looks ahead and includes packets on some other nodes as well as part of the delivery. The extra cost paid by the online algorithm is just a multiple of the cost that would have been paid if the delivery was based only on the rent-to-buy strategy.

Let us introduce a few notations.

We extend the domain for which we have defined the function *weight* so far. Let S be some subset of nodes. Then, **weight(S)** denote the sum of the outgoing edges from these nodes.

Let w_j be the delay cost accumulated at some node n_j with α packets. Let c_j be the weight of the outgoing link from n_j. Define, **triggering-time(n_j)** to be the time in future when the delay cost accumulated by the packets currently waiting at n_j reaches c_j i.e.,

$$triggering - time(n_j) = t_0 + (c - w)/\alpha$$

Assume that at some time the delay accumulated in a subtree t reaches $weight(t)$ and a delivery is triggered. Let **triggering-tree(t)** be the subtree for which the delay accumulated in the subtree t is equal to the weight of the edges in the tree t. Let **delivery-tree(δ)** denote the subtree for which the packets waiting at the nodes in the subtree are delivered in delivery δ.

Consider the following algorithm,

Algorithm $\mathcal{A}$:

- Wait until the delay cost accumulated on a segment of the serial network, $triggering{-}tree(t)$, is equal to its delivery cost.
- Choose the segment for delivery, $delivery - tree(\delta_i)$, as the maximal weight segment that includes $triggering - tree(t)$, such that the total weight of the $delivery - tree(\delta_i)$ is atmost twice the weight of $triggering - tree(t)$.

Theorem 4. *Algorithm $\mathcal{A}$ has competitive ratio at most 8 on any serial network.*

Proof. Let $\delta_1, \delta_2, \delta_3, \ldots, \delta_{k-1}, \delta_k$ be the sequence of deliveries made by $\mathcal{A}$. We shall prove that at all instants of time t, the competitive ratio of $\mathcal{A}$ is maintained.

First, we bound the total cost incurred by $\mathcal{A}$ in delivery δ_i,

Lemma 4. *The total cost incurred by $\mathcal{A}$ on packets delivered in delivery δ_i^{th}, $cost(\mathcal{A}, \delta_i)$ is at most $4 * weight(triggering - tree(\delta_i))$.*

Next, we bound the corresponding cost paid by the offline algorithm for the packets delivered in δ_i,

Lemma 5. *The total cost incurred by $\mathcal{O}$ on packets delivered in delivery δ_i^{th}, $cost(\mathcal{O}, \delta_\rangle)$ is at least $\frac{triggering-tree(\delta_i)}{2}$.*

The proof of lemma shall appear in the full version of the paper.

Let $wait(\mathcal{A}, t)$ and $wait(\mathcal{O}, t)$ be the cost incurred by the online and offline algorithm, respectively, for packets not yet delivered. The ratio of the total cost paid by the online algorithm and the total cost paid by the offline algorithm at any time t $(\geq \delta_i)$ is
$\frac{wait(\mathcal{A},t)+\sum_i cost(\mathcal{A},\delta_i)}{wait(\mathcal{O},t)+\sum_i cost(\mathcal{O},\delta_i)}$

From 4 and 6.2, we have that $\frac{cost(\mathcal{A},\delta_i)}{cost(\mathcal{O},\delta_i)} \leq 8$. Also, it follows from easy observation that the total waiting cost paid by the online algorithm $wait(\mathcal{A}, t)$ for packets waiting in the network of $\mathcal{A}$ is $\leq 4 * wait(\mathcal{O}, t)$.

Thus, algorithm $\mathcal{A}$ is 8 competitive.

6.3 Shallow Networks.

Figure 3 describes the structure of shallow networks. Consider the following algorithm for shallow networks,

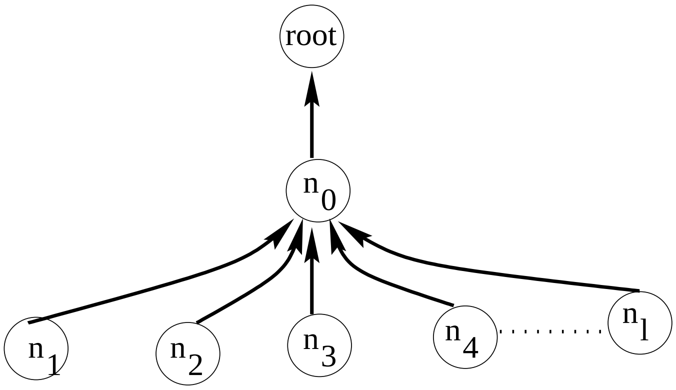

Fig. 3. A Shallow Network

Algorithm $\mathcal{B}$:

- Wait until the delay cost accumulated on a subtree $triggering - tree(\delta_i)$ is equal to its delivery cost;
- Choose the $delivery - tree(\delta_i)$ as follows: The smallest set of leaf nodes which are not part of $triggering - tree(t)$ are selected in increasing order of triggering times, such that the total cost of the delivery tree is at least equal to the weight of the triggering tree.

Algorithm $\mathcal{B}$ has a competitive ratio of at most 8. The proof of competitivity is very technical and shall appear in the full version of the paper.

7 Open Problems

Our (almost) tight lower bound for the Asynchronous Regime is based on the Interleaving Lemma. However, the proof of the lemma relies on the capacity of the adversary to create messages with unbounded number of packets, which in turn makes synchronization among the nodes infeasible. But for both the message aggregation problem and the organizational problem, it is a reasonable assumption that the adversary has limited such capacity —because of fixed queue lengths or bounded arrival rate. It would be an interesting extension of this work to tackle these problems. Randomization should also be explored towards the same goal.

Resolving the competitive ratio of the Full Information Regime on arbitrary topologies is an outstanding open problem. The conjecture is that there exists an algorithm with constant competitive ratio independent of the topology ([1]

give a competitive algorithm but its ratio depends on the topology). Our ideas of combining rent-to-buy strategies with prediction can be potentially useful for arbitrary topologies.

Acknowledgments

We would like to thank Joseph Naor and Christos Papadimitriou for helpful discussions.

References

1. Sanjeev Khanna, Joseph(Seffi) Naor, and Dan Raz *Control Message Aggregation in Group Communication Protocols** International Colloquium on Automata Languages and Programming, 2002
2. C. H. Papadimitriou, Edouard Servan-Schreiber *Optimizing communication in organizations* Presented at workshop on Complexity in Economic Games in Aix-en-Provence, 1999 To appear in an edited book on the economics of information. http://www.cs.berkeley.edu/ christos/deadlines.ps
3. D. R. Dooly, S. A. Goldman and S. D. Scott *On-line analysis of the TCP acknowledgement delay problem* Journal of the ACM,48:243-273, 2001
4. E. Bortnikov and R. Cohen *Schemes for scheduling of control messages by heirarchical protocols* IEEE INFOCOM'98, March 1998
5. C. Papadimitriou *Computational aspects of organizational theory* In ESA'96
6. Sandy Irani, Sandeep Shukla, Rajesh Gupta *Online Strategies for Dynamic Power Management in Systems with Multiple Power Saving States* Accepted for presentation at the Design Automation and Test Conference (DATE 2002)
7. Anna R. Karlin, Claire Kenyon, Dana Randall *Dynamic TCP acknowledgement and other stories about e/(e-1)* Symposium on Theory of Computing, 2001
8. B. R. Badrinath and P. Sudame *Gathercast: The design and implementation of a programmable aggregation for the internet.* Submitted for publication, 1999.
9. Reliable multicast protocols. *http://www.tascnets.com/mist/doc/mcpCompare.html*
10. M. Hofmann *A generic concept for large-scale multicast* In B. Plattner, editor, International Zurich Seminar on Digital Communication, number 1044, pages 95-106. Springer Verlag, Feburary 1996.
11. Daniel D. Sleator and Robert E. Tarjan *Amortized efficiency of list update and paging rules* Communications of the ACM, 28(2):202-208, February 1985.

A The χ^* Number of a Tree

In this section we define and characterize a combinatorial quantity, $\chi^*(T)$, for a directed rooted tree T. Except the definition of function χ, the results in this section are not necessary to understand the rest of the paper and may be skipped for a first reading.

Definition: Let Ψ be a set of paths and Ψ_t be the tree formed by the edges included in this path. Define, the function

$$\chi(\mathbf{\Psi_t}) = \frac{\sum_{\mathbf{p_i}\in\mathbf{\Psi}} \mathbf{weight}(\mathbf{p_i})}{\mathbf{weight}(\mathbf{\Psi_t})}$$

For χ^* we consider all possible subtrees of the tree.

Definition: Let $\mathcal{S_T}$ be the set of all subtrees of T, then the function χ^* is defined for T as:

$$\chi^*(\mathbf{T}) = \max_{\tau\in\mathcal{S_T}} \chi(\tau)$$

It may be worth mentioning that for every rooted tree T, there exists a subtree τ of T such that $\chi^*(T) = \chi(\tau)$. Thus, the problem of computing $\chi^*(T)$ reduces to the one of finding a subtree τ with $\chi^*(T) = \chi(\tau)$. Next, we will characterize a subtree τ of T such that $\chi(\tau) = \chi^*(T)$.

We shall prove the following theorem.

Theorem 5. *Let T be an arbitrary tree, and let p be the path of T with maximum cost. Then, $\chi^*(T) \leq weight(p)$.*

First, we shall show the following result,

Lemma 6. *Let τ be a subtree of T. Let n_k be a node of τ and let $d = weight(p_k)$, where p_k is the path from n_k to the root of T. Let t be a subtree of T rooted at node n_k such that $\tau \cap t = \{n_k\}$. Then, the following are equivalent:*

1. $\chi(\tau) < \chi(t) + d \cdot \frac{(|t|-1)}{weight(t)}$
2. $\chi(\tau) < \chi(\tau \cup t)$

Proof. We can express $\chi(\tau \cup t)$ as

$$\chi(\tau \cup t) = \frac{\chi(\tau)\cdot weight(\tau)+\chi(t)\cdot weight(t)+d\cdot(|t|-1)}{weight(\tau)+weight(t)}$$

$$= \chi(\tau) + \frac{weight(t)}{weight(\tau)+weight(t)} \cdot \left[\left(\chi(t) + \frac{d\cdot(|t|-1)}{weight(t)}\right) - \chi(\tau)\right]$$

since $weight(\tau)$ and $weight(t)$ are positive, we get that (1) holds if and only if (2) holds.

Theorem 6. *Let τ be a subtree of T. Then, $\chi^*(T) = \chi(\tau)$ if and only if the following condition hold:*

If n_k is a node of τ, $d = weight(p_k)$, where p_k is the path from n_k to the root of T, and t is a subtree of T with root n_k, then

1. *if $\tau \cap t = \{n_k\}$, then $\chi(\tau) \geq \chi(t) + d \cdot \frac{(|t|-1)}{weight(t)}$*
2. *if $\tau \cap t = t$, then $\chi(\tau) \leq \chi(t) + d \cdot \frac{(|t|-1)}{weight(t)}$*

Proof. Follows from lemma 6.

Proof of Theorem 5 Let τ be a subtree such that $\chi^*(T) = \chi(\tau)$. Let p_k be a path from leaf node $n_k \in \tau$ to root. Let n_j be the parent of n_k. Let c be the cost of link $n_j \rightarrow n_j$. Let t be the subtree rooted ar n_j with n_k as only leaf node. Theorem 6 gives:

$$\begin{aligned}\chi(\tau) &\leq \chi(t) + \frac{(weight(p_k)-c)(|t|-1)}{c}\\ &= weight(p_k)/c\end{aligned}$$

since $weight(p_k) \leq weight(p)$ and $c \geq 1$, the result follows. □

The semantic of the above theorem can be translated to construct a simple polynomial time algorithm for computing the χ^* value of a tree. It has been omitted from this version of the paper due to lack of space.