Randomized decentralized broadcasting algorithms

Laurent Massoulie*, Andy Twigg*, Christos Gkantsidis†, and Pablo Rodriguez‡
*Thomson Research, Paris. laurent.massoulie@thomson.net, andrew.twigg@cl.cam.ac.uk
†Microsoft Research, Cambridge, UK. chrisgk@microsoft.com
‡Telefonica, Barcelona. pablorr@tid.es

***Abstract*— We consider the problem of broadcasting a live stream of data in an unstructured network. The broadcasting problem has been studied extensively for edge-capacitated networks. We give the first proof that whenever demand $\lambda + \varepsilon$ is feasible for $\varepsilon > 0$, a simple local-control algorithm is stable under demand λ, and as a corollary a famous theorem of Edmonds. We then study the node-capacitated case and show a similar optimality result for the complete graph. We study through simulation the delay that users must wait in order to playback a video stream with a small number of skipped packets, and discuss the suitability of our algorithms for live video streaming.**

I. Introduction

We consider the problem of broadcasting a live stream of data, such as a movie, to all nodes in an unstructured network. When edges have capacities, this problem has been well-studied since the 1970s – Edmonds, Lovasz and Gabow and others have given centralized schemes based on packing spanning trees [1]–[5].

The broadcast problem is at the core of every content distribution system; in particular, current live streaming distribution systems such as CoolStreaming [6], PPLive [7], SplitStream [8]. These systems either construct the overlay topology in such a way to easy packet scheduling, and in doing so reduce the network efficiency by not optimally using all the available resources, or use heuristic algorithms for packet distribution with unknown performance properties.

We present a completely distributed and suprisingly simple algorithm for broadcasting in arbitrary networks, which does not require coding yet provably achieves the optimal broadcast rate. This is the first known result of this kind. Our analysis is based on fluid models and Lyapunov functions applied to a novel powerset representation of the network. As a corollary we retrieve a famous theorem of Edmonds [1].

In the second part of the paper, we introduce a new model for broadcasting a live stream of data in a peer-to-peer network based on the *node-capacitated* broadcast problem – each node has a specified upload capacity (we assume that download capacity is infinite), modeling the user's connection to the network, and the user must choose how this capacity is allocated among peers. This introduces an *allocation* problem in addition to the problem of *scheduling* packet transmissions. We present a completely distributed algorithm for the node-capacitated broadcast problem, and show that it achieves the optimal rate in certain classes of network.

This work was done while all authors were at Microsoft Research, Cambridge, UK.

We study the startup delay for streaming live data, and show analytical and experimental results that suggest that our algorithms are well suited to this application – more precisely, the startup delay decreases exponentially as we increase the maximum number of skipped frames that we can tolerate. We stress that our approach is theoretical because we are interested in finding heuristics with provable good performance. However, we believe that our results can be used to construct practical systems for live streaming.

The rest of this paper is organized as follows. In Section II we study the case of edge capacitated networks, we propose a simple algorithm for forwarding packets, and prove that it is optimal. In Section III we study the case of node capacitated networks; we propose an extension of the random forwarding algorithm and prove optimality for the case of complete graphs. Finally, we study its behaviour on other graphs through simulation.

II. Edge-capacitated networks

We model the network as a directed graph $G = (V, E)$ with n nodes and m edges, and with capacities $\{c_{uv}\}$ assigned to edges. For a partition $(S, \bar{S})$ of nodes of G, we use $C(S, \bar{S})$ to denote the sum of edge capacities crossing the cut, i.e. $C(S, \bar{S}) = \sum_{(u,v)\in E(G):u\in S, v\notin S} c_{uv}$. For a network G, the value of a minimum s, t-cut is

$$\mathrm{mincut}(s,t) = \min_{S\subset V(G), u\in S, v\notin S} C(S, \bar{S}).$$

The minimum s-cut is denoted $\mathrm{mincut}(s)$ and is equal to $\min_{t\in V(G)} \mathrm{mincut}(s,t)$.

In the broadcast problem, there is a source s that wishes to send the same data to all nodes in the network. For a network G, we denote the optimal broadcast rate from a source $s \in V(G)$ by $\lambda^*(s)$ (G shall be obvious from the context). In 1969, Edmonds gave a precise characterisation of $\lambda^*(s)$ in terms of the minimum s-cut.

Theorem 1 (Edmonds (1969)): For a directed graph G with source s, the optimal broadcast rate is $\lambda^*(s) = \mathrm{mincut}(s)$.

A directed spanning tree rooted at s is called an s-arborescence. Edmonds initially gave a constructive proof of his theorem by showing that the optimal rate can be achieved by packing edge-disjoint s-arborescences. The running time of his algorithm was exponential in n and was later improved to a small polynomial by Lovasz, Gabow and others. However, these algorithms are inherently centralized and so are unsuitable for use in a peer-to-peer network.

A. Random useful packet forwarding

We now describe our completely distributed algorithm for edge-capacitated broadcasting, which we call "random useful packet forwarding." For a node u, we use $P(u)$ to denote the set of packets that u has received. The algorithm is as follows: for each edge (u, v), at rate c_{uv} the edge (u, v) sends random packets in $P(u) \setminus P(v)$ from u to v.

We assume an injection model of packets at the source – for a specified *injection rate* $\lambda \geq 0$, the source receives new packets at rate λ, injected in order $1, 2, 3, \ldots$. This models the situation where not all the packets are initially available, for example in a live video stream. Clearly, this model is stronger than the model where the source is given all the packets initially. We can prove the following result for static networks, i.e. networks with no node arrivals or departures and no connectivity changes.

Theorem 2: For every edge-capacitated network G and source s, if the injection rate is $\lambda < \lambda^*(s)$ then random useful packet forwarding from s achieves broadcast rate λ.

Therefore, by injecting at a rate $\lambda^*(s) - \epsilon$ for some $\epsilon > 0$, we can broadcast at a rate arbitrarily close to the optimal rate.

In order to evaluate the performance of our algorithm, we shall first examine the lifetime of a typical packet. Once injected at the source, a packet p can be in a number of different states: (a) It can be replicated at all nodes in the system, hence successfully broadcasted. (b) It can be idle, that is not actively transferred, and be replicated at nodes u in some set $S \subset V$. The subset S cannot be arbitrary; it much contain a spanning tree rooted at s. We shall denote by $\mathcal{S}$ the collection of strict subsets of V that contain the source node s. (c) It can be replicated at some nodes $u \in S$, for some subset $S \in \mathcal{S}$, but also actively transferred along some edges $e \in F$, for some subset $F \in E$.

We shall describe the state of the system as follows: (a) For all $S \in \mathcal{S}$, X_S denotes the number of idle packets, that are replicated exactly at the nodes $u \in S$. (b) $A = \{G_1 = (W_1, F_1), \ldots, G_m = (W_m, F_m)\}$ is an unordered list of subgraphs which describes the active packets. W_i denotes the set of nodes at which the i-th active packet is currently replicated; F_i is the set of edges along which the i-th active packet is currently transferred.

We shall assume that at any given time, at most one packet is transferred along a given edge, hence, the total number of active packets is (at most) $|E|$. We shall further assume that the same active packet cannot be received from multiple incoming edges. We also enforce an *activity condition* which states that if there is no transfer along some edge (u, v), then necessarily there is no packet that could be transferred along this edge.

We now describe the transitions between the states of the system. When a new packet arrives at the source, then the number of idle packets at the source s increases by 1:

$$X_{\{s\}} \leftarrow X_{\{s\}} + 1$$

When an active packet, which can be described by $G = (W, F)$, finishes transmission along an edge $e = (u, v)$, then the following updates take place:

$$\begin{aligned} W &\leftarrow W \cup \{v\} \\ F &\leftarrow F \setminus e \end{aligned}$$

If $F = \emptyset$ as a result of the update, then the packet is now idle, so

$$X_W \leftarrow X_W - 1$$

After the finish of a transmission along edge $e = (u, v)$, we need to ensure that a new transmission starts, if there is a packet that can be transmitted from u to v; in other words, we need to ensure the activity condition. For all $S \in \mathcal{S}$ such that $u \in S, v \notin S$ the following state update will take place:

$$\begin{aligned} X_S &\leftarrow X_S - 1 \\ A &\leftarrow A \cup (S, (u, v)) \end{aligned}$$

with probability:

$$\frac{X_S}{X_{+u-v} + X^{\alpha}_{+u-v}}$$

where X_{+u-v} and X^{α}_{+u-v} is the number of idle and active packets respectively that exist in u but not in v. Observe that the state update above will take place if an idle packet is chosen for transmission. In the case of an active packet chosen for transmission the state update is as follows:

$$A \leftarrow A \setminus (W, F) \cup (W, F \cup (u, v))$$

with probability $1/\left(X_{+u-v} + X^{\alpha}_{+u-v}\right)$. Observe that in the state transitions described above each candidate packet for transmission from u to v is chosen with the same probability.

The process described above by the states $\left((X_S)_{S \in \mathcal{S}}, A\right)$ and the state transitions allows us to express the dynamics of packet distribution and to analyze the performance of broadcasting packets from a server to all nodes of a network. Using this framework, we will prove in Section II-B that random packet forwarding is optimal. For presentation clarity we shall assume that the time intervals between fresh packet arrivals at the source, and packet transfer times follow exponential distributions. In particular, we shall assume that the mean inter-packet arrival at the source is λ^{-1}, and that the mean packet transfer time along edge (u, v) is c_{uv}^{-1}. Of particular interest is to assume that the time intervals between fresh packet arrivals at the source and the packet transfer times along each edge are i.i.d. random variables, and, hence, the model is Markovian. In that case, we would have to augment the state space to keep track of the residual times until (a) the arrival of the next fresh packet at the source, (b) the completion of a transmission. In particular, we would be very interested to study the case of deterministic distributions for the arrival of packets at the source and the transmission along every edge; the deterministic case can be though to better model sources with a constant arrival of fresh packets and networks with constant capacity at each link. We defer the general case to future work, and prove optimality in the special case of exponential distributions.

B. Random packet forwarding is optimal

We shall now prove Theorem 2. We will show that the Markov process $((X_S)_{S\in\mathcal{S}}, A)$ is ergodic under the condition:

$$\lambda < \min_{S\in\mathcal{S}} \sum_{u\in S}\sum_{v\notin S} c_{uv}$$

This result will be established by using the so-called "fluid limits" approach, introduced and popularized by [9] and [10]. Informally, the approach consists in first establishing that trajectories of the original Markov process, after join rescaling of both time and space, evolve according to some simpler "fluid" dynamics, and then to prove that the trajectories of the fluid dynamics converge to zero in finite time. Due to space constraints we will outline the proof in this paper; a more detailed version can be found in [11].

The main intuitive difficulty is that, although the algorithm may waste bandwidth by unnecessarily replicating packets (as a result of nodes only having knowledge of their neighbours' collections of packets), the random nature of the protocol creates sufficient diversity that the fraction of wasted bandwidth goes to zero in the rescaled fluit limits. Note that a naive strategy of sending packets in FIFO order will clearly fail in the case when there are three nodes (one source, two receivers) connected in a triangle with unit-capacity edges – the optimal rate is 2 but the FIFO strategy will never make use of the edge connecting the receivers, giving a rate of 1.

Let us introduce the following definition.

Definition 1: For all $S \in \mathcal{S}$, all $u \in S, v \notin S$, there exist non-negative functions $t \to \phi_{S,(uv)}(t)$ that represent the number of packets previously replicated at nodes in S that have been sent over edge (uv), up to time t. The real-valued non-negative functions $t \to y_S(t)$, $S \in \mathcal{S}$, are called fluid trajectories of the above Markov process if they satisfy the following conditions:

$$\begin{array}{rcl} y_{\{s\}}(t) & = & y_s(0) + \lambda t - \sum_{v\in V\setminus\{s\}} \phi_{\{s\},(sv)}(t) \\ S \neq \{s\}: \; y_S(t) & = & y_S(0) + \\ & & \sum_{u\in S}\sum_{v\in S\setminus\{u\}} \phi_{S\setminus\{v\},(uv)}(t) \\ & & - \sum_{u\in S}\sum_{v\notin S} \phi_{S,(uv)}(t), \end{array} \tag{1}$$

and that are non-decreasing, Lipschitz continuous with Lipschitz constants c_{uv}. In addition, for all $(u,v) \in E$, it holds that:

$$\sum_{S\in\mathcal{S}:u\in S, v\notin S} \phi_{S,(uv)} \text{ is } c_{uv}\text{-Lipschitz.}$$

Moreover at almost every point t, the function $\phi_{S,(uv)}$ is differentiable, and the following holds:

$$y_{+u-v}(t) > 0 \Rightarrow \frac{d}{dt}\phi_{S,(uv)}(t) = c_{uv}\frac{y_S(t)}{y_{+u-v}(t)}, \tag{2}$$

where we have used the notation

$$y_{+u-v}(t) := \sum_{S'\in\mathcal{S}:u\in S', v\notin S'} y_{S'}(t). \tag{3}$$

$\diamond$

The following result shows in what sense such fluid trajectories describe the dynamics of the original Markov process after spatial and temporal rescaling:

Theorem 3: Consider a sequence of initial conditions $(X^N(0), A^N(0))$, $N > 0$, such that

$$\lim_{N\to\infty} \frac{1}{N} X_S^N(0) = x_S(0), \; S \in \mathcal{S}.$$

Introduce the rescaled process

$$Y_S^N(t) := \frac{1}{N} X_S^N(Nt),$$

where $X_S^N(t)$ represents the state of the Markov process with initial conditions $(X^N(0), A^N(0))$ at time t. Then for any subsequence of indices N, there exists a further subsequence, denoted N', such that, for some fluid trajectory (y_S) as per Definition 1, with initial conditions $(x_S(0))$, the following uniform convergence takes place:

$$\lim_{N'\to\infty} \sup_{t\in[0,T]} |Y_S^{N'}(t) - y_S(t)| = 0, \quad S \in \mathcal{S}, \; T \in \mathbb{R}_+. \tag{4}$$

Proof: See [11]. ■

The following theorem establishes that any fluid trajectories as per Definition 1 satisfy a suitable stability property:

Theorem 4: Assume $\lambda < \min_{S\in\mathcal{S}} \sum_{u\in S}\sum_{v\notin S} c_{uv}$. Let $(y_S)_{S\in\mathcal{S}}$ denote fluid trajectories as per Definition 1. For all $S \subset V$, define:

$$y_{\subseteq S} = \sum_{S'\in\mathcal{S}, S'\subseteq S} y_{S'}.$$

Then there exist positive parameters $\beta_1, \ldots, \beta_{|V|-1}$, and $\epsilon > 0$ such that the function

$$L(\{y_S\}_{S\in\mathcal{S}}) := \sup_{S\subset V} \beta_{|S|} y_{\subseteq S}$$

verifies:

$$L(y(t)) \leq \max\left(0, L(y(0)) - \epsilon t\right). \tag{5}$$

Proof: See [11]. ■

The proof of Theorem 2 will require to combine Theorems 3, 4 and the following ergodicity criterion, which is a direct consequence of Theorem 8.13, p.224 in Robert [12]:

Theorem 5: Let $Z(t)$ be a Markov jump process on a countable state space $\mathcal{Z}$. Assume there exists a function $L : \mathcal{Z} \to \mathbb{R}_+$ and constants $M, \epsilon, \tau > 0$ such that for all $z \in \mathcal{Z}$:

$$L(z) > M \Rightarrow \frac{1}{L(z)} \mathbf{E_z L}\left(\mathbf{Z}(\mathbf{L}(\mathbf{z})\tau)\right) \leq \mathbf{1} - \epsilon. \tag{6}$$

If in addition the set $\{z : L(z) \leq M\}$ is finite, and $\mathbf{E_z L}(\mathbf{Z}(\mathbf{1})) < +\infty$ for all $z \in \mathcal{Z}$, then the process $Z(t)$ is ergodic.

The detailed proof of Theorem 2 can be found in [11].

Theorem 2 proves that random packet forwarding can achieve optimal broadcast rate for static networks and under the assumption of exponential arrivals of fresh packets at the source and exponential packet transmission along each edge. We conjecture that the main result can be generalized to arbitrary distributions, and in particular with deterministic

distributions. Indeed, we have performed extensive simulation studies of deterministic distributions and show that the main conclusion of Theorem 2 applies. The case of dynamic networks, with node arrivals and departures, and connectivity changes, is an interesting open problem.

This result has several implications:

- It shows that we can achieve the optimal broadcast rate using only local information in a distributed network;
- It gives a constructive proof of Edmonds' theorem that does not rely on packing edge-disjoint arborescences.

III. Node-capacitated networks

Traditionally, network flow problems assume capacities assigned to edges. This a natural model when the edge bandwidths are indeed the bottlenecks. However, in peer-to-peer networks, the capacity bottlenecks are not at the edges (which are often high capacity links) but are dictated by the users' connections to the network, for example cable, dsl, etc. Often the user has a specified total upload capacity and must choose how to allocate this among its peers. We shall assume for simplicity that the download capacity is infinite, but our algorithms can be easily extended to handle both bounded download and upload capacities.

Given a graph G with node capacities $C(u)$, an edge capacity assignment $\{c_{uv}\}$ is called *feasible* if $\sum_v c_{uv} \leq C(u)$. By Edmonds' theorem, every feasible edge capacity assignment has an associated λ^* that can be achieved by packing edge-disjoint arborescences, as shown in Figure 1. We can then study the problem of constructing a feasible edge capacity assignment that maximizes the broadcast rate λ^*.

There is no known explicit characterisation of λ^* for the node-capacitated case. A fairly weak upper bound is given by the sum of upload capacities, divided by the total number of receivers:

$$\lambda^* \leq \frac{\sum_u C(u)}{n-1} \tag{7}$$

The maxflow-mincut theorem is often defined with edge capacities, but it is well-known that it can easily be defined with node capacities using a simple transformation: given a node-capacitated network G, replace each node u of G by two nodes u^- and u^+ linked by an edge of capacity $C(u)$. Point all the incoming edges of u to u^- and make all the outgoing edges of u leave from u^+ (all these edges have capacity ∞). Call this new edge-capacitated graph G'. It is easy to see that any feasible edge capacity assignment in G corresponds to a feasible flow in G'. Therefore the value of the minimum s,t-cut in the node-capacitated G is equal to the minimum s^-,t^--cut in the edge-capacitated G'. This solves the node-capacitated case for the case of a single receiver (the unicast problem).

Such a transformation does not work for the broadcast problem. Consider the network in Figure 2 – the node-capacitated network on the left can clearly only support a broadcast rate of one, but the min-mincut of the network on the right is two (and by Edmonds' theorem has broadcast rate two). The problem arises because the transformation 'creates' capacity when there are multiple receivers sharing the same flow paths.

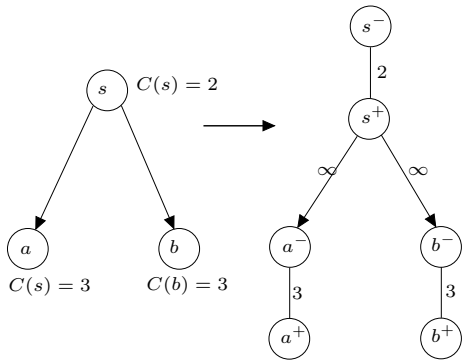

Fig. 2. An example of when the simple node-capacity transformation used in the unicast case fails in the broadcast case

A. LP formulation

We can formulate the maximum rate node-capacitated multicast problem as a linear program, as shown below. For a source s and receivers $T_1, \ldots, T_k$, we add directed edges $(T_i S)$ for each receiver T_i, as in [13]. This makes the LP more concise.

$$\begin{array}{l} \text{maximize } \chi \\ \text{subject to} \\ \\ \begin{array}{ll} \chi \leq f_i(T_i S) & \forall i \\ f_i(\overrightarrow{uv}) \leq c(\overrightarrow{uv}) & \forall i, \forall \overrightarrow{uv} \neq \overrightarrow{T_i S} \\ \sum_v (f_i(\overrightarrow{uv}) - f_i(\overrightarrow{vu})) = 0 & \forall i, \forall u \\ \sum_v c(\overrightarrow{uv}) \leq C(u) & \forall u \\ c(\overrightarrow{uv}), f_i(\overrightarrow{uv}), \chi \geq 0 & \forall i, \forall \overrightarrow{uv} \end{array} \end{array} \tag{8}$$

The LP has a number of variables polynomial in n, so the problem is solvable in polynomial time. However, the linear program is both very slow in practice (we have been unable to solve it for more than 40 nodes) and is inherently centralized, making it unsuited for use in a dynamically-changing peer-to-peer network where nodes only have local information.

Subgradient methods offer a general method of solving optimization problems in a distributed way. By working on the dual of the LP in (8), we can write the following subgradient formulation:

SUBGRADIENT-UPDATE(u)
1 Find the node k minimizing $\text{mincut}(s,k)$ with $\{c_{uv}\}$
2 Set $c_{uv} = c_{uv} + \delta(t), \forall (uv)$ in the minimum s,k-cut
3 Normalize $\{c_{uv}\}$ so that $\sum_v c_{uv} = C(u)$ for all u

The algorithm is initialised by choosing any feasible edge capacity assignment $\{c_{uv}\}$; we choose the *balanced* assignment, where each node shares its capacity equally among its neighbours. Each node u then runs SUBGRADIENT-UPDATE(u) until the maximum rate achieved (given by the min-mincut with the current edge capacity assignments), with the parameter t denoting the iteration number $1, 2, \ldots$. For the algorithm to converge, we require

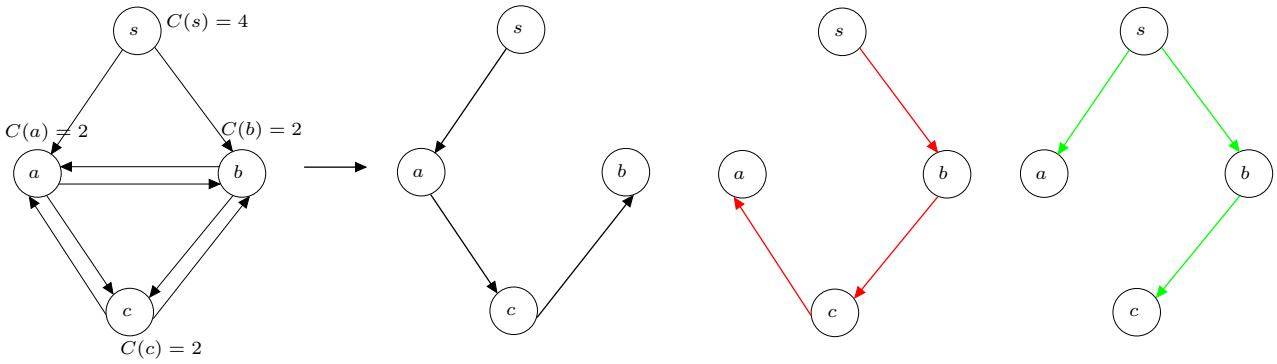

Fig. 1. A node-capacitated network with $\lambda^* = 3$, and a packing of edge-disjoint arborescences achieving it (each aborescence contributes a rate of 1). The corresponding edge capacity assignments are $c_{sa} = 2, c_{sb} = 2, c_{ac} = 1, c_{bc} = 2, c_{ca} = 1, c_{cb} = 1$ and all other capacities are 0.

that the step function $\delta(t)$ be nonnegative ($\forall t, \delta(t) \geq 0$), converge to zero ($\lim_{t\to\infty} \delta(t) = 0$) and its sum divergent ($\sum_{t=1}^{\infty} \delta(t) = \infty$). The family of functions $\delta(t) = \frac{a}{bt+c}$ for $a, b, c > 0$ satisfies these requirements.

The subgradient algorithm solves the problem of constructing a feasible edge capacity assignment having maximum rate, but does not tell us how to actually do the broadcasting. For this, we can use the edge-capacitated graph corresponding to the computed assignments as input to any edge-capacitated broadcast algorithm. In particular, if we feed the assignment into the random useful packet forwarding scheme for edge-capacitated broadcasting at each iteration, we get the following result.

Theorem 6: Let G be a network with node capacities $C(u)$ and optimal broadcast rate $\lambda^*(G)$. For any injection rate $\lambda < \lambda^*(G)$, there is a distributed algorithm that converges to a broadcast rate λ.

The subgradient method has several problems, in particular the following:

- Although each node u has to maintain only the current values of c_{uv} for its neighbours v, the algorithm is not completely distributed; each iteration requires us to solve n mincut computations, one for each receiver. Although there are distributed algorithms for solving these mincut computations (such as the ε-relaxtion method of Tseng and Bertsekas [14]), these make the algorithm costly both in communication and time.
- Although the theory of subgradient optimization guarantees that with the correct choice of the function δ, the algorithm will converge geometrically to the optimal solution, there are no guidelines for how to choose δ. In practice, we have observed that it can have a large influence on the convergence time, and depends on graph parameters such as connectivity.

B. Random useful packet forwarding

In this section we prove the suprising result that the random useful packet forwarding idea of Section II can be extended to the node-capacitated case. Each node u does the following at rate $c(u)$: choose a neighbor v, then send v a packet chosen at random from $P(u) \setminus P(v)$. All we need is a way to pick the neighbor v. We shall study two neighbor selection strategies: *random* and *most deprived* neighbor. For the random neighbor selection, each node u chooses at random a neighbor v having $|P(u) \setminus P(v)| > 0$. For the most deprived neighbor strategy, each node u chooses a neighbor maximizing $|P(u) \setminus P(v)|$ (if there is more than one such neighbor, one is chosen them at random). This algorithm is given below.

RANDOM-USEFUL-PACKET-FORWARDING(u)
1 At rate $C(u)$
2 $v \leftarrow$ random neighbor maximizing $|P(u) \setminus P(v)|$
3 Send a random packet in $P(u) \setminus P(v)$ to v

It might seem surprising that this algorithm (with the most deprived neighbor strategy) should work well – Figure 3 shows a network where intuitively it might fail. The intuitive reason for this might be that the source will, in the long term, send half its packets to a and half to b. Since node a has capacity zero, it cannot forward any packets to b, and so node b receives at rate five instead of the optimal rate of nine. Indeed, this is what happens with the random neighbor selection strategy.

With the most deprived neighbor strategy, what happens is the following. Since b gives packets to a, the source observes that b becomes the more deprived node, so the source diverts its capacity towards b. Unfortunately, b cannot give packets to a at rate ten (the source capacity), so the source starts to observe that a sometimes becomes its most deprived neighbor and switches its capacity to a. Over the long term, this switching process results in s giving a a rate of one and b a rate of nine. Since b can give node a a rate of eight, in the long term, a, b both receive at rate nine, which is optimal.

Similarly to Section II-A the system is described by a a graph $G = (V, E)$. However, the capacities are now associated with nodes rather than with edges. We shall denote by c_u the capacity of node u, and assume that each node devotes its capacity to one of its "most deprived neighbors.". Using the same notation as in Section II-A, this reads:

$$Z_{+u-v} = X_{+u-v} + X^a_{+u-v}.$$

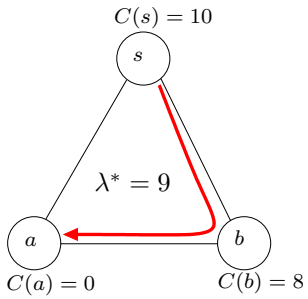

Fig. 3. An example of when the random forwarding most-deprived neighbor algorithm might intuitively fail, but does not. Intuitively, the zero-capacity node 'sucks' half the source capacity but contributes nothing.

It then elects one neighbor v for which the corresponding quantity Z_{+u-v} is maximal. Ties can be broken either at random, or in a systematic manner. Once the target neighbor v is chosen, then one of the Z_{+u-v} packets held by u and useful to v is chosen, and forwarded from u to v, at rate c_u.

Packet selection We now describe how packets are elected for transmission once a node's capacity becomes available. For non-source nodes u, who have chosen to transmit to some most deprived neighbor v, then the packet to be transmitted is selected at random among all the possible Z_{+u-v} possible choices.

For the source node s, having chosen to transmit to some most deprived neighbor v, the following strategy is used: if the source has a packet that it has not sent to anyone before (a *fresh* packet), that is if $X_{\{s\}} > 0$, then one such fresh packet is forwarded to node v; if no such fresh packets are available, then the packet to be forwarded is selected uniformly at random from the Z_{+s-v} possible choices.

As in the edge capacitated case, the state space consists in the collection of variables X_S, for all $S \in \mathcal{S}$, and the collection of active packet states $A = ((W_1, F_1), \ldots, (W_m, F_m))$. The constraints on these active packet states are different though: we now assume that each node forwards a packet to only one of its neighbors at a given time. Thus for each node u, there is at most one edge (u, w) appearing in the sets F_i, $i = 1, \ldots, m$. Otherwise the same constraints apply: for a given active packet (W, F), and each edge $(u, v) \in F$, necessarily, $u \in W$ and $v \notin W$; also, there is no other edge (u', v) pointing towards v in F.

We shall assume that packet transmissions are not pre-empted, even if a neighbor of some node u becomes more deprived than the neighbor v to which node u is currently transmitting.

As in the edge-capacitated case, in the present work we focus on the case where completion of a packet transmission by node u is an Exponential random variable with mean $1/c_u$, and where fresh packets arrive at the source node s at the instants of a Poisson process with rate λ.

Next, we shall see that the random useful packet forwarding performs well in complete graphs (Section III-C) and other topologies (Section IV-A)

C. Proof of optimality for complete graphs

We can prove that the random forwarding algorithm with the most deprived neighbor selection strategy achieves the optimal broadcast rate in complete networks.

We first define the candidate fluid trajectories for the system under consideration:

Definition 2: The real-valued, non-negative functions $(y_S)_{S \in \mathcal{S}}$ are called fluid trajectories of the node-capacitated system if the following properties hold.

For all $S \in \mathcal{S}$, $u \in S$, $v \notin S$ such that $(u, v) \in E$, there exist non-decreasing, Lipschitz-continuous functions $\phi_{S,(uv)}$ with Lipschitz constant c_u, such that Equations (1) hold. Furthermore, using notation

$$y_{+u-v} := \sum_{S \in \mathcal{S}: u \in S, v \notin S} y_S,$$

for all $S \in \mathcal{S}$, $u \in S$, the functions $\{\phi_{S,(uv)}\}_{v \notin S,(uv) \in E}$ are differentiable at almost every t, and if $\sum_{v:(u,v) \in E} y_{+u-v}(t) > 0$, their derivatives satisfy:

$$\frac{d}{dt}\phi_{S,(uv)}(t) = 0 \text{ if } y_{+u-v}(t) < \max_{v':(u,v') \in E} (y_{+u-v'}(t)), \quad (9)$$

$$\sum_{v:(uv) \in E} \sum_{S: u \in S, v \notin S} \frac{d}{dt}\phi_{S,(uv)}(t) = c_u. \quad (10)$$

If $u \neq s$, that is for a non-source node, one also has, for all v such that $(uv) \in E$ and assuming the condition

$$\sum_{S: u \in S, v \notin S} \frac{d}{dt}\phi_{S,(uv)}(t) > 0$$

holds, the following equation:

$$\forall S / u \in S, v \notin S :$$
$$\frac{d}{dt}\phi_{S,(uv)}(t) = \frac{y_S(t)}{\sum_{S': u \in S', v \notin S'} y_{S'}(t)} \cdot \sum_{S': u \in S', v \notin S'} \frac{d}{dt}\phi_{S',(uv)}(t)$$

For the source node s, one has the following:

$$y_{\{s\}} > 0 \Rightarrow \sum_{v \neq s} \frac{d}{dt}\phi_{\{s\},(sv)}(t) = c_s. \quad (11)$$

In the case where $y_{\{s\}} = 0$, one then has for all v such that $(sv) \in E$, assuming the condition

$$\sum_{S \in \mathcal{S}: S \neq \{s\}, v \notin S} \frac{d}{dt}\phi_{S,(sv)}(t) > 0$$

holds, the following:

$$\forall S \in \mathcal{S} / S \neq \{s\}, v \notin S :$$
$$\frac{d}{dt}\phi_{S,(sv)}(t) = \frac{y_S(t)}{\sum_{S' \in \mathcal{S}: S' \neq \{s\}, v \notin S'} y_{S'}(t)} \cdot \sum_{S' \in \mathcal{S}: S' \neq \{s\}, v \notin S'} \frac{d}{dt}\phi_{S',(sv)}(t)$$

◇

We now establish the following

Theorem 7: Consider a sequence of initial conditions $(X^N(0), A^N(0))$, $N > 0$, such that the limit

$$\lim_{N\to\infty} \frac{1}{N} X^N(0) = y(0) \in \mathbb{R}_+^{\mathcal{S}}$$

exists, with $y(0) \neq 0$. Then for any subsequence, there exists a further subsequence, that we denote by N', and a fluid trajectory with initial condition $y(0)$, such that for all $T > 0$, all $S \in \mathcal{S}$,

$$\lim_{N'\to\infty} \sup_{t\in[0,T]} \left| \frac{1}{N'} X_S^{N'}(Nt) - y_S(t) \right| = 0. \tag{12}$$

Proof: See [11]. ■

The main result we shall establish is in the case of the complete graph, that is all edges (u, v), $u \neq v$, are present in E. We then have the following

Theorem 8: Assume that the graph $G = (V, E)$ is complete, and that the injection rate λ verifies:

$$\lambda < \min\left(c_s, \frac{\sum_{u\in V} c_u}{K-1} \right), \tag{13}$$

where $K = |V|$. Then the Markov process keeping track of the system state under "random useful to most deprived neighbor" scheduling strategy is ergodic.

The proof of Theorem 8 parallels exactly that of Theorem 2, relying on a combination of Theorem 5 with Theorem 7 (taking the role played by Theorem 3 in the proof of Theorem 2) and of Theorem 9 below (taking the role played by Theorem 4 in the proof of Theorem 2).

Theorem 9: For any $y = (y_S)_{S\in\mathcal{S}} \in \mathbb{R}_+^{\mathcal{S}}$, define the *workload* function $w(y)$ as:

$$w(y) = \sum_{S\in\mathcal{S}} y_S \left(K - |S|\right), \tag{14}$$

where $K = |V|$. Under the assumption (13), when the graph G is complete, any fluid trajectory y as per Definition 2 is such that, for some $\epsilon > 0$,

$$w(y(t)) \leq \max(0, w(y(0)) - \epsilon t). \tag{15}$$

Proof: See [11]. ■

It is open to determine whether the scheme is optimal for general networks.

IV. Live video streaming

A motivating application for our work is the streaming of live video. Assume that we have a network that can support a broadcast rate $\lambda < \lambda*$, and that our algorithm can indeed broadcast at rate λ. We assume there is some coding process that is generating packets at a fixed rate λ, and we want to broadcast these packets. At time zero, the source begins broadcasting packets. At time D, node u will start playing the stream at rate λ. Each node must playback the packets in order $1, 2, 3, \ldots$, and any packets that are not present when they are needed for playback are said to be *skipped*. We are interested in the relationship between number of skipped packets and the playback delay required.

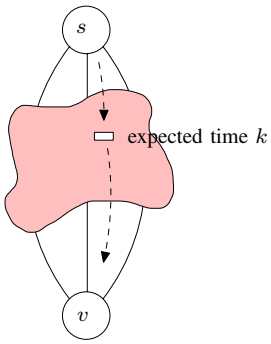

Fig. 4. A simple model of delay for the random packet forwarding scheme

Assume that the stream shall be played back until time T. Consider some node u. Since we can broadcast to u at rate λ, we shall approximate the network as follows. Let k be the expected time for a packet to travel from the source to u (k is bounded by the previous assumption). Construct a network with a source s connected to a single node v, as in Figure 4. Initially, s has k packets $1, \ldots, k$ and v has no packets. At rate λ, s gives random useful packets to u.

Lemma 1: In the simple model, for an expected number of skipped packets α, we require delay $D \geq \log \frac{T}{\alpha} + O(k)$.

Proof: For a packet i, we have

$$\Pr(i \notin P(A) \text{ at time } t) = \begin{cases} (1-1/k)^t & \text{for } i \leq k \\ (1-1/k)^{t-(i-k)} & \text{for } i > k \end{cases} \tag{16}$$

After delay D, the node v starts playing its packets in order. We can then write

$$\Pr(i \text{ not received by time } i+D) \leq \begin{cases} (1-1/k)^D & i \leq k \\ (1-1/k)^{D+k} & i > k \end{cases} \tag{17}$$

Let Y be the number of skipped packets when playing to time T after waiting delay D. Then

$$Y \sim \text{Binom}\left((1-1/k)^D, k\right) + \text{Binom}\left((1-1/k)^{D+k}, T\right).$$

The expected number of skipped packets is then

$$E[Y] = k(1-1/k)^D + T(1-1/k)^{D+k},$$

so for $E[Y] \leq \alpha$, taking logs of both sides gives

$$D \geq \frac{\log \frac{Tk}{\alpha} + k\log(1-1/k)}{2\log\frac{k}{k-1}}.$$

■

Setting $\alpha = pT$ for some fraction p gives a simple corollary – the expected fraction of skipped packets decreases exponentially with increasing playback delay. If this simple model is indeed a good model then random useful packet forwarding should be well-suited to streaming movies and other data where the source is not initially given all the packets. We now present some experimental results that support this claim.

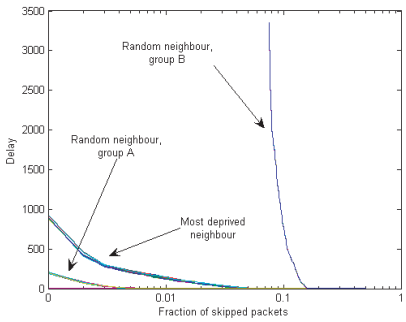

Fig. 5. Delay vs. packet loss fraction for the clustered topology of Figure 6. Choosing neighbours at random is clearly a bad strategy, as for some nodes the delay goes to infinity. For the most-deprived neighbour strategy, it can be seen that the delay falls exponentially with increasing tolerable packet loss fraction and the delays of all nodes are closely clustered together – a desirable property for a live video streaming application.

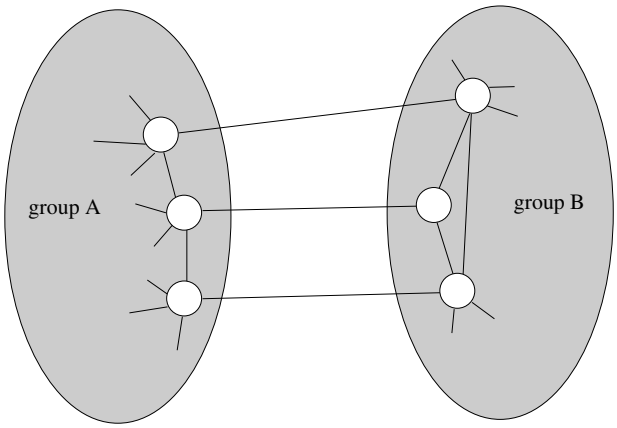

Fig. 6. Another example of when the random forwarding most-deprived neighbor algorithm might intuitively fail, but does not. Each group of nodes is well-connected, e.g. a clique or an expander. The algorithm must fully utilize the links crossing the two groups in order to achieve a good broadcast rate.

A. Simulation results

Since we have been unable to prove that results for the random-forwarding-to-most-deprived strategy in the case of general networks, we simulated it on various topologies in an attempt to gain insight into its performance. The simulations also allowed us to study the tradeoff between startup (or playback) delay and tolerable packet loss, both crucial parameters for a live video streaming application.

1) Topologies with clusters: A basic but interesting example is the case where the network contains two well-connected groups of nodes (such as cliques or expanders), which are connected together by only a few edges. Such a graph is shown in Figure 6. We simulated a network with two groups A,B; each group is a clique containing one hundred nodes, each with an upload capacity of one (the source is in group A). The optimal broadcast rate is one (construct a Hamiltonian cycle in the network), yet the random neighbor strategy gives a rate of one to nodes in group A, and a rate of approximately $1/100$ to nodes in group B . This is because it cannot realize that the nodes having edges between the groups must divert their capacity completely to their neighbors in the other group. With the most deprived neighbor strategy, all nodes receive at the optimal rate of one. This happens because the nodes in group B will remain more deprived than those in group A unless they are given packets from the nodes on the boundary, and this forces them to be selected for transmission by their neighbors in group A.

2) Heterogeneous capacities: In the previous examples we have assumed that all nodes have the same capacity. However, in many applications of practical interest is common to have few nodes with large capacities and many slower nodes. For example, the table in Figure 7 shows the distribution of node upload capacities in a Gnutella network, as observed by Bharambe et al. [15].

We constructed a random graph with node capacities following the distributions given in [15], and plot in Figure 8 the performance of the most-deprived random packet forwarding algorithm on this network. For each node, we plot the delay that each node must wait to play back the entire stream at the injection rate, as a function of the tolerable fraction of packet loss. As predicted by the simple model in Section IV, the startup (or playback) delay decreases exponentially as the tolerable fraction of skipped packets increases.

Upload capacity (Kb/s)	Fraction of nodes
128	0.2
384	0.4
1000	0.25
5000	0.15

Fig. 7. The distribution of node upload capacities used for the simulation. (Source: [15].)

3) Grid network: We then ran simulations on a grid network, containing 1600 nodes (arranged in a 40x40 grid). The source is placed at the center of the grid and all nodes have equal upload capacity. Although the algorithm achieves the optimal broadcast rate (by using a Hamiltonian path, for example), we are interested in the delay required for playback. With a Hamiltonian path, the delay would be high for nodes at the end of the path. Figure 9 shows how for a fixed fraction (0.02) of skipped packets, the delay increases with position in the grid. As expected, the delay increases with distance from the source. We then added shortcut edges randomly (independently with probability 0.01 for each edge), and studied the delay for the same fraction of skipped packets. Figure 10 shows that this completely changes the picture; the delay is now almost equal for every node in the network, only the random fluctations from shortcuts placement are visible. Even though the shortcut edges do not increase the optimal broadcast rate, they allow us to reduce the startup delay. The simulations here show that the random forwarding algorithm can take advantage of these shortcut edges.

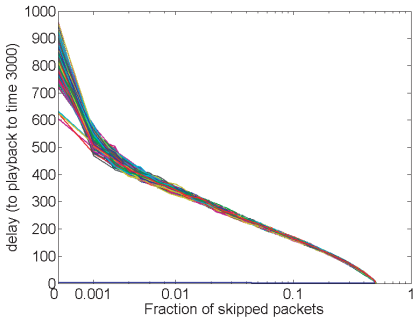

Fig. 8. Delay vs. tolerable packet loss fraction for random graph with gnutella-like distribution of node capacities. The delay decreases exponentially with increasing fraction of skipped packets. For example, for packet loss fraction 0.01, a startup delay of about 300 is required, in order to playback until time 3000.

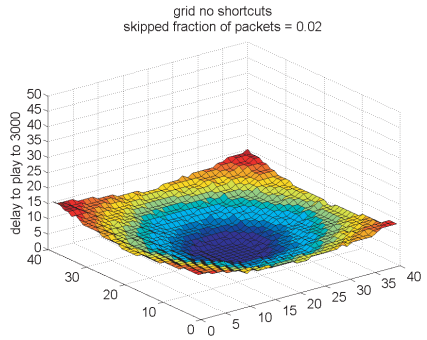

Fig. 9. Simulation on a grid network without shortcuts. The source is at the center, and the delay increases with distance from the source.

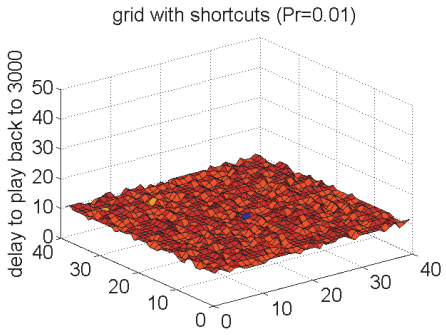

Fig. 10. Simulation on a grid network with shortcut edges, added randomly with Pr=0.01. The source is at the center, and the delay is almost equal over all the nodes. This shows that random forwarding algorithm takes advantage of these shortcut edges.

V. Remarks

A major problem with current peer-to-peer systems that use some distributed algorithm to broadcast a stream of data is the following: when the system starts running slowly, how do we know whether it is due to the algorithm making bad use of the available network resources, or that there is a problem in the network (but the algorithm is making full use of the available resources)?

We have presented the first proof that a simple local-control algorithm can route packets at a broadcast rate λ, whenever a rate of $\lambda + \varepsilon$ is feasible, for some $\varepsilon > 0$, and without resorting to network coding. The strength of this result is that it should inform inform and validate the design of peer-to-peer broadcast and live streaming protocols, by basing their design on a provably good foundation with known performance guarantees.

For other applications such as multicasting and handling multiple commodities, the situation is much more complex. Our proof techniques relied upon knowing a tight maxflow-mincut theorem for the broadcast problem (established by Edmonds in 1969). For multicasting, it is known that without network coding, there is no such tight result. For the multi-commodity case, it may be possible but there is no known exact maxflow-mincut theorem from which to start.

References

[1] J. Edmonds, "Edge-disjoint branchings," in *Combinatorial Algorithms*, R. Rustin, Ed., pp. 21–31. Algorithmics Press, 1972.

[2] Laszlo Lovasz, "On two minimax theorems in graph," *Journal of Combinatorial Thoery*, vol. 21, pp. 96–103, 1976.

[3] Po Tong and E.L. Lawler, "A faster algorithm for finding edge-disjoint branchings," *Information Processing Letters*, vol. 17, pp. 73–76, 1983.

[4] H.N. Gabow, "A matroid approach to finding edge connectivity and packing arborescences," *Journal of Computer and System Sciences*, vol. 50, no. 2, pp. 259–273, 1995.

[5] Harold Gabow and K.S. Manu, "Packing algorithms for arborescences (and spanning trees) in capacitated graphs," *Mathematical Programming*, vol. 82, pp. 83–109, 1998.

[6] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *IEEE Infocom*. 2005, IEEE Press.

[7] "Pplive," `http://www.pplive.com/`.

[8] "Splitstream: High-bandwidth content distribution," `http://research.microsoft.com/~antr/SplitStream/default.htm`, Aug 2003.

[9] H.Q. Ye, J.H. Ou, and X.M. Yuan, "Stability of data networks: Stationary and bursty models," *Operations Research*, vol. 53, no. 1, pp. 107–125, 2005.

[10] J.G. Dai, "On positive harris recurrence of multiclass queueing networks: a unified approach via fluid limit models," *Ann. of Applied Probability*, vol. 5, pp. 49–77, 1995.

[11] L. Massoulie, A. Twigg, C. Gkantsidis, and P. Rodriguez, "Provably optimal decentralized broadcast algorithms," Tech. Rep. MSR-TR-2006-105, Microsoft Research, Jul 2006.

[12] P. Robert, *Stochastic Networks and Queues*, Springer, 2003.

[13] Zongpeng Li, Baochun Li, Dan Jiang, and Lap Chi Lau, "On achieving optimal throughput with network coding," in *Proc. Infocom*, 2005.

[14] Paul Tseng and Dimitri P. Bertsekas, "A epsilon-relaxation method for generalized separable convex cost network flow problems," in *Proceedings of the 5th International IPCO Conference on Integer Programming and Combinatorial Optimization*, London, UK, 1996, pp. 85–93, Springer-Verlag.

[15] Ashwin Bharambe, Cormac Herley, and Venkata Padmanabhan, "Analyzing and improving bittorrent performance," Tech. Rep. MSR-TR-2005-03, Microsoft Research, feb 2005.