Libra, a Multi-hop Radio Network Bandwidth Market

Lars Rasmusson, Sverker Janson
Lars.Rasmusson@sics.se, sverker@sics.se
SICS, Box 1263, SE-16428 Kista, Sweden

2007-04-18

Abstract

Libra is a two-level market which assigns fractional shares of time to the transmitting nodes in local regions of a multi-hop network. In Libra, users are assigned budgets by management and users assign funding to services within their budget limits. The purpose is to prioritize users and also optimize network utilization by preventing source nodes from injecting too much traffic into the network and thereby causing downstream packet loss. All transmitting nodes sell capacity in the region surrounding them, and buy capacity from their neighbors in order to be able to transmit. Streams buy capacity from each of the nodes on their paths, thus streams that cross the same region compete directly for the bandwidth in that region. Prices are adjusted incrementally on both levels.

1 Introduction

We present a market for the scarce local radio channel capacity in multi-hop ad hoc radio networks. We call this market Libra, because it dynamically "balances" network capacity allocation between competing users. Libra trades channel capacity on many levels, both on the local, "regional", level surrounding each transmitting node, as well as on the global level involving multiple regions on the same time. The latter is necessary to use the channel capacity efficiently in the presence of multi-hop traffic.

Libra is a two-level market which, at its fundamental level, assigns fractional shares of time to the transmitting nodes in local regions of the multi-hop network.

The nodes' shares are computed in a decentralized fashion, and in a way that enables the stream owners, or users, to acquire more or less capacity for their streams simply by increasing or decreasing the money rate at which they pay for the stream.

In brief the market operates in an iterative fashion, as follows. Money is sent from the users to the relaying nodes. A relaying node uses the incoming money to buy transmission time shares from the other relaying nodes in its local neighborhood. The acquired transmission time share is split between the streams. Users are informed of the bit rate they have acquired for their streams, and limit the stream sources transmission rates accordingly. The users update their spending rates if necessary, and the process starts over.

1.1 The Problem

In radio networks there are two main types of data loss, loss caused by the environment (noise), and loss caused by interfering transmissions. The first type is adressed by advanced coding techniques, and the second by coordinating the transmitters. The topic of this paper is a coordination mechanism to prevent the second type of loss.

The more transmitting nodes that use a shared, finite medium, such as a radio frequency band, the higher risk for collisions. Collisions result in packet-loss and low throughput. To avoid collisions, nodes in a region must be prevented from sending too much. For instance, the sum of the fraction of time each node transmits, must not exceed one, or there will absolutely be collisions. I.e. if we have four nodes in the same region, each sending at thirty percent of the channel capacity, then we will experience packet loss.

In a single-hop network, collisions can be avoided with centralized coordination, as is done in GSM networks, where the base-station assigns transmission time slots to the phones.

The situation is more complex in multi-hop networks, because a data packet must be sent on a path through many regions in order to reach its destination. A packet may be sent through many regions, but eventually reach a congested region where it gets dropped, or worse, collides with other packets. This is called "downstream packet loss", and is highly undesirable, because all the consumed resources along its path are now completely wasted. In situations of extreme downstream loss, the network may be very busy, without achieving any useful work.

Downstream packet loss can by some amount be reduced with store/forward. It means that the intermediary nodes store packets in a buffer, and resend the packet until it has arrived to the next node on its path. Store/forward is undesirable in resource-constrained, or low-latency networks, because without any control on incoming traffic, buffers will eventually fill up, and with large buffers, packets will spend a lot of time waiting in buffer queues in the network.

A better approach is to limit traffic at its source, i.e. prevent nodes from injecting too much new traffic into the network. But how much should they be allowed to inject? This is what Libra calculates.

2 Libra Model

The Libra model consists of six entities: a radio network, transmitting nodes, markets, streams, stream controllers, and users.

Here we present the entities in a bottom-up order, which is suitable for a technical understanding, but it may be advisable to read about them in a top-down order to get an understanding of how the system appears to its users.

2.1 Radio Network

Libra relies on an already existing link layer radio network. The radio network uses a coding technique such that data loss in a region occurs only when the transmitters together send more than the channel capacity. As long as they send less, the data gets through. This behavior can be achieved for instance with a frequency hoppning or ultra-wideband radio link, and error correcting codes.

The above implies that the channel capacity in a region can be divided into fractional shares, which can be assigned to the senders in the region, and that if the senders do not send more than their share, there will not be packet loss caused by collision. Also, because Libra delegates the local coordination to the physical radio network layer, Libra essentially trades shares of the transmission time, without specifying when the sender should transmit.

Libra further assumes that the network is able to route packets on multi-hop paths from a source to a destination node, either using routing tables or via an explicitly requested path, so called source routing. The routing paths should be fairly stable, or the routing layer should be able to alert Libra about upcoming re-routing events (similar to hand-over events in GSM).

In this paper, we consider the network to use the same transmission rate on all links, because it gives a conceptually simpler meaning to capacity shares, but it is also possible to use heterogeneous nodes with different channel capacities.

2.2 Transmitting Nodes

The radio network used by Libra is built out of relaying nodes that both transmit and receive packets. A node has a set of neighboring nodes that it can hear, and a set of neighbors it can disturb, by which it is ment that a node m can disturb a node n if n can hear m. These sets are not necessarily identical, because of differences in transmitting power, signal propagation, etc.

The term "region" around a node, which we used informally above, is the set of nodes that it may disturb.

Each node hosts a market that trades capacity shares in its region. A node that want to transmit, buys capacity shares on each of the markets in its region. To prevent packet loss, the node may not transmit more than what it is allowed by the smallest share it gets.

The node gets its money from the streams (see below) that traverse the node. It uses that money in the node markets to get as much capacity as possible. It is continuously adjusting how much it buys in the different markets so as to maximize the smallest share it gets.

2.3 Streams

A stream is an established path through the network, with a budget and desired bit rate. Streams have unique identifiers, and packets belonging to the stream have the stream id written into the packet headers. A stream starts in a source node, goes via a set of relay nodes, and ends at a destination node.

Each stream has a local money account and a local spending rate in each node that transmits its packets. The stream's spending rate on a node determines the stream's share of the node's total transmission time.

A node gets its money by taking money from the local accounts, at the spending rate of those accounts. The total incoming money is used by the node to acquire capacity from the neighboring nodes it disturbs. The resulting capacity is then divided between the streams proportionally to their local spending rates. Thus each stream has a specific capacity at each node. An iterative process initiated by the stream's controller works towards getting the same capacity in all its nodes, thus maximizing throughput.

2.4 Libra Markets

The markets in Libra are implemented implicitly by the price and bid announcements made by the participants. There are markets at two levels, the node level, and the stream level.

On the node level, each node has a market at which it sells "the right to disturb it". This implicitly coordinates the transmitters in its region so that they will not cause packet collisions in its region.

The node level market is implemented as a broadcast message called $price()$, which each node sends out once every second to the nodes in its region. This message contains the bids, expressed as a spending rate, that it sends to its neighbors, and its price, which is the sum of the bids coming in to the node itself. The node keeps track of all bids coming in to it, and updates them when it hears a broadcast from one of its neighbors.

The share a node gets from another node equals its proportional or "fair share", i.e. its bid divided by the price on that node. To decide its bid, it computes the bid that maximizes the smallest share it gets on any of its neighbors. The optimal bid can be solved with the Lagrange multiplier method, and it turns out that the optimal bidding strategy for a node is to

distribute its money proportionally to how much the other nodes spend there, i.e. proportionally to the node's price minus the node's previous bid.

On the stream level, each node hosts a market where it resells the capacity it acquired on the node level market. A stream gets a share of a node's share that is proportional to the stream's spending rate on that node. Every second, $fund()$ messages are sent out along the streams, and they contain a desired bit rate. Thus, a stream that requests a bit rate r should bid enough that its bid divided by the sum of all bids gives it its desired rate. This will naturally change the allocation to the other streams, but their shares will be re-adjusted by the same process.

The node deducts money from the $fund()$ message and puts them in the stream's money account, so that it's balance is enough to cover two funding intervals, to avoid out-of-money problems at the end of the funding period.

2.4.1 Market-Based Resource Allocation

With Libra, prices that change all the time, which causes resource allocations to fluctuate over time. It may be seen as an inconvenience, but the reason is that dynamic prices leads to a more efficient resource allocation. Even though it is convenient for some purposes, fixed prices and static reservations prevent a system from adapting to changing demand.

With dynamic prices, scarce resources cost more, and plentiful resources cost less. This clearly conveys the fact that buying/allocating scarce resources will prevent others from doing useful work. A user that strives to minimize its cost will also automatically minimize its negative impact on other users.

Market-based resource allocation is useful not only in non-cooperative systems, but also in cooperative systems, such as within an organization. It simplifies delegation of control, and encourages decentralized decision making. Higher level management assigns resource budgets to its lower levels, and the lower levels make local resource decisions.

2.5 Stream Controllers

Stream controllers are control loops that are run at the source of a stream. The controllers try to estimate the maximum bit rate they can buy for a specific stream, at a certain spending rate. The stream controller starts out with a guessed bit rate. It sends out money to the nodes along the stream with $fund()$ messages, requesting the guessed bit rate. When the message reaches the destination node, it is returned to the stream controller. If the stream controller gets back more money than expected, it adjusts its guessed bit rate upwards, and if it gets back less money than expected, the guess is adjusted downwards.

The exact details of how big the adjustments should be are a topic for our future research. To large adjustments create oscillations, while too small adjustments cause slow convergence. However, our very simple controllers show that convergence happens even for quite dynamic scenarios.

2.6 Users

Users are the entities that create streams. Each user is located in one of the transmitting nodes, which serves as the source (or destination) of its streams.

Users run applications that require data transfers through the network. A user has a budget which is set by upper level management, and the user should use those money to acquire bandwidth for its streams in order to maximize the perceived quality of service.

The user informs the stream controller of its desired quality of service by setting a spending rate (currency per second), and not by specifying of a bit rate, which is the traditional approach. The spending rate buys the user some actual bit rate, and the user has to decide whether the current bit rate provides sufficient quality, or whether to pay more, by reallocate funds from other streams to the current stream.

We prefer the idea of allocating money to streams to that of setting hard quality of service constraints, even though they are essentially functionally equivalent. A simple focus on the bit rate obscures the fact that capacity is allocated at someone else's expense, while the use of spending rates highlights the fact that there is a cost involved for someone else.

2.7 Management

In Libra, money is a tool for specifying priorities between entities on the same level. Entities on the same level compete for resources, but their relative priorities are determined by the funding they get from higher levels.

One particular advantage of market-based resource allocation is that it allows higher level management to delegate detailed decisions to lower levels, while keeping control on the overall priorities between different tasks.

Libra users get their money from some higher level, which is not part of Libra, or of the radio network, but the higher levels are part of the market that Libra operates on. For instance, in Libra a user may be a staff person, or even a department, that gets funded by higher levels of management.

If an entity at one level gets problems which cannot be solved by reallocating its resources, it will need more resources, and the problem is treated at the next upper level as a budget problem. On that level, the entity manager likewise has to choose whether to reallocate money between its lower entities, or whether to ask for more money from the next higher level.

The use of money as a tool for prioritization has an unexpected result. Money should be spent optimally, but entities should not try to save money. Instead,

they should spend all the money. If they do not, then higher levels are prevented from using money to control priorities, for instance by increasing money allocation to some important user or stream.

2.8 Implementation

Libra uses two kinds of messages, $price()$ and $fund()$ messages, and a small amount of state in the nodes. To avoid cluttering the formulas, we assume all funding intervals to be one second, as extending them is straight forward.

Each node n keeps track of the spending rate $r_{s,n}$, balance $b_{s,n}$, and last fund time $t_{s,n}$ for stream s on n. For each node m in n's region N, n keeps track of m's bid $x_{m,n}$ on n, of n's bid $x_{n,m}$ on m, and the price p_m of m.

Every second, n updates its bids and price

$$x_{n,m} := \frac{p_m - x_{n,m}}{\sum_k p_k - x_{n,k}} \sum_s r_{s,n}, p_n := \sum_m x_{m,n}$$

and broadcasts

$$price(n, p_n, \{(m, x_{n,m}) | m \in N\})$$

n may now send at $c_n = C x_{n,m}/p_m$, where C is the max link capacity. When a node receives a price message $price(m, p_m, \{..., (n, x_{m,n}, ...)\})$, it updates p_m and $x_{m,n}$.

When node n, at time t, receives a $fund(s, a, g_s)$ message containing amount a and requesting bit rate g_s, and for which it is not the final destination, it first deducts money $b_{s,n} := b_{s,n} - (t - t_{s,n}) r_{s,n}$. Then it computes the necessary spending rate $r_{s,n} := g_s \sum_u r_{u,n}/c_n$, and updates $a := a - (2r_{s,n} - b_{s,n}), b_{s,n} := 2r_s$ and $t_{s,n} = t$. If the node is the final destination, it forwards the fund message back to the originating stream controller.

The stream controller for stream s keeps track of its spending rate r_s and its bit rate guess g_s. Every second the stream controller issues $fund(s, 2r_s, g_s)$ messages addressed to the destination node of stream s.

When the controller gets back a $fund(., a, .)$ message, it increases g_s if $a > r_s$ and decreases it otherwise. In our test implementation we multiply (or divide) b_s with 1.01, but a more robust controller, like a PID regulator should be used.

3 Results

We have tested the Libra model briefly in a simulated environment, and can here report some qualitative results. We used a random network topology, created by adding nodes on a 2d square until the network was connected.

Figure 1 shows the behavior of the node level market, as nodes iteratively update their bids. To maximize their minimal share they need to move their money onto nodes where they have a small share, thus eventually all money are on the few congested nodes. There is no point in buying capacity upstream if the streams are saturated downstream. When funding levels change, the new price level is reached smoothly.

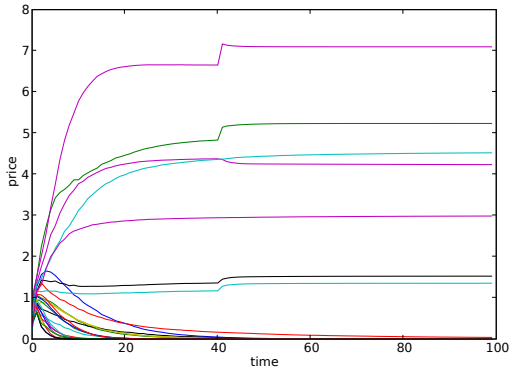

Figure 1: All nodes have budget=1. A single stream flows through the network. Note that only a few nodes get congested, most nodes have price zero. Prices stabilize quickly after one of the nodes got increased funding at t=40.

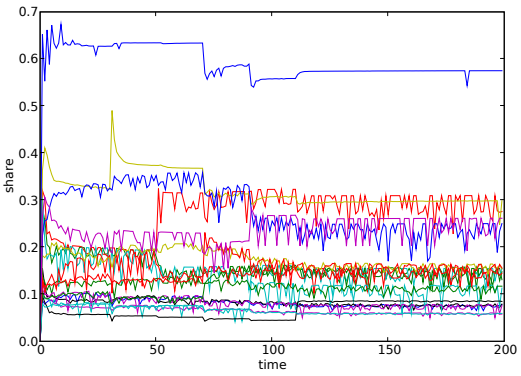

Figure 2: Stream funding changed every 20 s. On a short time scale the streams' shares fluctuate a lot, because of other streams and nodes adjust their bids.

Figure 2 shows the shares obtained by the nodes. In this scenario, streams get adjusted funding rate every 20s. Node bid updates causes other nodes' shares to fluctuate. The fluctuations appear to belong to different regimes, determined by the price level. At some price levels, fluctuations can be large, while at others, the fluctuations are smaller.

Even though the shares fluctuate quite a lot on the short time scale of the figure, the shares are quite stable on just a little longer time scales, which means that it is sufficient with quite small end-point buffers to negate the fluctuations.

Figure 3 shows the effect of the simple stream

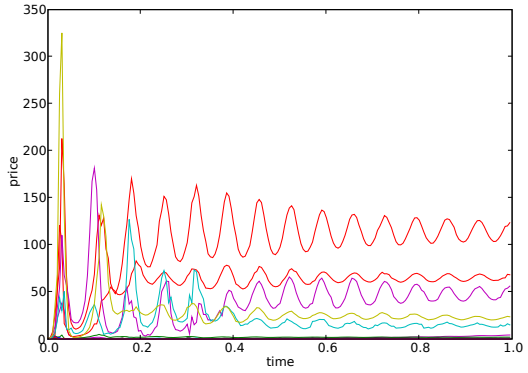

Figure 3: Active stream controller adapts bit rate guess with simple minded controller. Prices oscillate and converge slowly.

controller which changes the bit rate guess for the streams. Because of the system lag, the prices oscillate, but they seem well behaved enough that it should be feasible to get rid of them with a better control algorithm.

4 Related Work

The idea to use rate control at the edge of the network has roots in wireline network. Kelly, et al. [1] proposed an elegant and stable rate control scheme, but, unlike Libra, it assumes that users have individual utility functions, which are sufficiently well-behaved.

Qiu, et al. [2] present a price-based allocation scheme, in which users charge other users a proce for relaying their packets, using iterative price adaptation. Their model therefore models transmitting power scarcity, rather than link contention.

Rate control at the edge has also been proposed for ad hoc networks. Curescu et al. [3], and Xue, et al. [4] build on the work of Kelly, and applies the ideas to ad hoc networks. Thus they also rely on utility functions, and use shadow prices to create a distributed version of an optimization problem. Unlike Kelly, they use a two-tier market and compute internal prices in interior sub-cliques in the network.

5 Discussion

The market described in this paper is intended for a cooperative system with the purpose of providing a conceptually simple model that provides a fair resource utilization, and where users have a simple model of the consequences of their actions, i.e. pay more to get more.

The nodes that transmit have to be cooperative, and follow the protocols. This is the case for all radio protocols, as it is very easy to disturb a radio network.

It is easy to detect non-cooperative nodes in the network, since they transmit more than their fair share, which results in higher packet-losses in that area. Non-cooperative nodes have to be handled outside of the protocol (i.e. by sending someone out to shut down the node physically).

Since we assume the nodes are cooperative, they do not have to perform extensive book-keeping of where money goes. The purpose of the bandwidth market is to get a high and fair network throughput, which is in the interest of all the users in a cooperative system. It is sufficient that the users keep track of their spending, and don't over-spend.

The rules allow the streams' $fund()$ messages to have negative balances. This is not assumed to be a problem, again because of the cooperative nature of the system. Should it be a problem, the transit nodes could start shaping the traffic, to prevent someone from disturbing the system by trying to allocate much larger shares than can be afforded.

An interesting case for future work is to use the Libra market prices to guide the building of network routing tables or source routes. High prices are indicators of network congestion, and therefore the network capacity could be better used if routes were built to avoid expensive, and thus congested, parts of the network. One approach would thus be to use the Libra node prices as the "distance" in a weighted distance vector routing protocol.

The work on Libra is funded by SICS Center for Networked Systems.

References

[1] F. Kelly, A. Maulloo, and D. Tan. Rate control for communication networks: shadow prices, proportional fairness and stability. In *Journal of the Operational Research Society*, volume 49, pages 237–252, 1998.

[2] Y. Qiu and P. Marbach. Bandwidth Allocation in Ad Hoc Networks: A Price-Based Approach. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2, pages 797–807. IEEE, 2003.

[3] Calin Curescu and Simin Nadjm-Tehrani. Price/Utility-based Optimized Resource Allocation in Wireless Ad Hoc Networks. In *IEEE SECON 2005. Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, pages 85–95. IEEE, 2005.

[4] Yuan Xue, Baochun Li, and Klara Nahrstedt. Optimal Resource Allocation in Wireless Ad Hoc Networks: A Price-Based Approach. *IEEE Transactions on Mobile Computing*, 5(4):347–364, 2006.