Online Broadcast Scheduling
New Perspectives and Results

Chandra Chekuri
University of Illinois
Online Broadcast Scheduling
New Perspectives and Results

Chandra Chekuri
University of Illinois

Sungjin Im

Ben Moseley
Goals of Talk

- Make you aware of/interested in broadcast sched
- Highlight known results, key open questions, some recent results in online case
- Interesting algorithmic idea(s) that could be of general interest

Impressionistic proofs
Pull-based Broadcast
Motivation

- Wireless and multicast where broadcast is natural (several applications)
- Batched scheduling (batch size infinity/large) (models studied in queuing theory)
- Other models: push vs pull, stochastic vs worst-case
- [Bartal-Muthukrishnan’00] [Kalyanasundaram-Pruhs-Velaithupillai’00] initiated work on worst-case online and offline algorithmic analysis in pull-based model
- Sustained interest due to simplicity and algorithmic interest/difficulty
Formal Model

♦ Server has $n$ pages of information
♦ Each clients request a specific page
♦ When server broadcasts a page $p$, all outstanding requests for page $p$ are simultaneously satisfied

Uniform page sizes: all pages have same size (1 wlog)

Non-uniform page sizes: ignored for most of talk
Formal Model contd

- Requests arrive at beginning of slot
- Transmission of page takes one time slot

\[ J_{p,i} : i_{th} \text{ request of page } p \]
- \( a_{p,i} : \text{arrival time} \)
- \( f_{p,i} : \text{finish time in some schedule} \)
- \( F_{p,i} = f_{p,i} - a_{p,i} : \text{flowtime/response time/waiting time} \)
Unicast Scheduling, unit sized jobs

To contrast with broadcast scheduling

Unicast job scheduling: all jobs unit-sized

- $J_i$: job $i$
  - $a_i$: arrival time, assume integer
  - $f_i$: finish time in some schedule
  - $w_i$: non-negative weight
What to optimize?

Flowtime $F_{p,i} : f_{p,i} - a_{p,i}$

Standard metrics:

- minimize average/total flowtime: $\Sigma_{p,i} F_{p,i}$
- minimize maximum flowtime: $\max_{p,i} F_{p,i}$
- minimize $L_k$ norms of flowtime: $\left( \Sigma_{p,i} (F_{p,i})^k \right)^{1/k}$
- weighted versions
- Maximize throughput (requests have deadlines)

New metric(s): delay factor
Worst-case Framework & Resource Augmentation

Input is worst-case (adversarial)

**Offline:** exact poly-time algo or approximation ratio

**Online:** competitive ratio

Resource augmentation [Kalyanasundaram-Pruhs]

Algorithm given \(s\)-speed server while adversary given 1-speed server for some \(s \geq 1\)
What is known?

**Offline** results for average flowtime

- $O(1)$-speed $O(1)$ approx [Kalyanasundaram-Pruhs-Velauthapillai’00]
- NP-Hard [Erlebach-Hall’02], simpler proof [Chang-Erlebach-Gailis-Khuller’08]
- $(1+\varepsilon)$-speed $O(1/\varepsilon)$ approx for any $\varepsilon > 0$ [Bansal-Charikar-Khanna-Naor’05], also $O(n^{1/2})$ approx
- $O(\log^2 n/\log \log n)$ approx [Bansal-Coppersmith-Svir’06]

All approx algorithms based on LP relaxation
What is known?

**Online** for average flowtime

- $\Omega(n)$ lower bound for any algorithm [K-P-V’00]
- “Reduction” to non-clairvoyant parallel scheduling problem [Edmonds-Pruhs’02]. Via reduction
  - BEQUI-EDF is $(4+\varepsilon)$-speed $O(1)$-competitive [EP’02]
  - LAPS is $(2+\varepsilon)$-speed $O(1)$-competitive [EP’09]
- Longest-Wait-First (LWF) is 6-speed $O(1)$-competitive. Not $O(1)$-competitive with $< 1.618$ speed [Edmonds-Pruhs’04]
What is known?

Max Flowtime:

♦ FIFO is 2-competitive? [Bartal-Muthukrishnan’00]
♦ First published proof [Chang etal’08]
♦ NP-Hard [Chang etal’08]
Why is it difficult?

- Different schedules can do different amounts of work – should one wait to broadcast in the hope of accumulating more requests or broadcast it now?
- In online case, a standard analysis technique from unicast scheduling does not apply [KPV’00]. No online algorithm even with speed is “locally” competitive in terms of queue size with respect to “all” schedules.
Key Open Questions

**Offline:** approximability of basic questions.
- Average flowtime: is there an \( O(1) \) approx or a PTAS?
- Maximum flowtime: is there a \( c \)-approx for \( c < 2 \)?
- \( L_k \) norms of flowtime?

**Online:**
- Is there a “scalable” algorithm for average flowtime?
  A \((1+\varepsilon)\)-speed \( O(f(1/\varepsilon)) \)-competitive for every \( \varepsilon > 0 \)?
- Competitive algorithms for \( L_k \) norm, \( k > 1 \)?
- Max weighted response time (the \( \infty \) norm)
New Online Results

Summary:

♦ Simpler/improved analysis of LWF

♦ New algorithms: LF, LF-W, LF-W+LF

♦ Scalable algorithm for max weighted flowtime

♦ [Im-Mosely’09] Scalable algorithm for average flowtime (*Scalable algorithm for $L_k$ norms?)

♦ Results extend to delay factor scheduling
New Online Results

- Simpler/improved analysis of LWF. Improved lower bound of $2^{-\varepsilon}$ on speed required for $O(1)$-competitiveness
- New algorithms: LF, LF-W, LF-W+LF
- $O(k)$-speed $O(k)$-competitive algorithm for $L_k$ norms
- Scalable algorithm for max weighted response time
- FIFO is 2-comp. for max flowtime with varying page sizes
- **[Im-Mosely’09]** Scalable algorithm for average flowtime (*Scalable algorithm for $L_k$ norms?)
- Above Results extend to delay factor scheduling
- Throughput scheduling as submodular function maximization, related results
Rest of Talk

- LWF and similar algorithms
- Simplified analysis of LWF
- Scalable algorithm for max weighted response time
- Concluding thoughts
Weighted case

Each request $J_{p,i}$ has weight $w_{p,i}$

- **Observation:** For average flowtime weights don’t matter in broadcast scheduling. Why? Also for $L_k$ norms for fixed $k$

- Weights make a big difference for $k = \infty$ (max weighted flowtime)

- Weighted case related to *delay factor*

- Helped understand/develop new algorithms
Understanding Broadcast

MRF: most requested first

Observation: MRF if not $O(1)$-competitive for any fixed speed $s$ [K-P-V’oo]

$\begin{align*}
p_1 \\
p_2 \\
p_3 \\
\vdots \\
p_n \\
\end{align*}$

$\begin{align*}
MRF & \quad p \quad p \quad p \quad p \\
OPT & \quad p \quad p_1 \quad p \quad p_2 \\
\end{align*}$
Understanding Broadcast

**MRF**: most requested first

**Observation**: MRF if not \(O(1)\)-competitive for any fixed speed \(s\) \([K-P-V’00]\)

Broadcast scheduling tradeoff:

- wait & merge requests for same page to save work
- accumulate flowtime

Difficulty exemplified by lack of good offline algos
Longest Wait First (LWF)

- $A(t)$: requests alive at time $t$
- For page $p$: $W(p,t) = \sum_{(p,i) \in A(t)} w_{p,i} (t - a_{p,i})$
- Schedule $q = \text{argmax}_p W(p,t)$

A natural and greedy algorithm/rule

Seems to work well in practice

First worst case analysis [Edmonds-Pruhs’04]
Longest First (LF)

Generalize LWF to cost metrics/objectives (example $L_k$ norm of flowtime for $k > 1$)

“Schedule page that has largest accumulated cost”

$LF_k$: LF for minimizing $L_k$ norms of flowtime

LWF is same as $LF_1$

FIFO is same as $LF_\infty$ (for unweighted)
How good is LWF/LF?

LWF requires 1.618 speed to be $O(1)$-comp. [EP’04]

*LWF requires $2-\varepsilon$ speed to be $O(1)$-comp. even for unicast scheduling

* $L_{F_k}$ requires $(k+1-\varepsilon)$ speed to be $O(1)$ competitive for $L_k$ norms.

$L_{F\infty}$ is not $O(1)$ comp. with any const speed for weighted

Why are LWF/LF not (as) good? They don’t distinguish between pages of same cost. Better to give preference to higher weight/more recent pages
How good are LWF/LF?

- LWF is 6-speed $O(1)$ comp. Needs 1.618 speed \[\text{EP'04}\]
- LWF is 3.44-speed $O(1)$-comp. *Needs 2-$\epsilon$ speed even for weighted unicast scheduling
- $L^k$ is $O(k)$-speed $O(k)$-comp for $L^k$ norms. *$L^k$ needs $(k+1-\epsilon)$ speed.
- $L^\infty$ is not $O(1)$ comp. with any const speed for max weighted flowtime

$L^k$ performance deteriorates with $k$. Why?
Weakness of LWF/LF

- They do not distinguish between pages of same cost.
- Can give preference to low weight pages that have waited very long instead of high weight pages that arrived more recently.
- Damage worse for large k

Fix?
New Algorithm: LF-W

LF-W(c) with parameter $c \geq 1$

$F_{\text{max}}(t)$: maximum page at $t$

$Q(t)$: all pages alive at $t$ with cost $\geq F_{\text{max}}(t)/c$

Among pages of $Q(t)$, schedule one with max weight/max number of requests
New Algorithm: LF-W

LF-W(c) with parameter \( c \geq 1 \)

Fmax(t): maximum page at \( t \)

Q(t): all pages alive at \( t \) with cost \( \geq \frac{Fmax(t)}{c} \)

Among pages of \( Q(t) \), schedule one with \textit{max weight} / \textit{max number of requests}

\textbf{Conjecture:} LF-W(1/2) is \( O(1) \)-speed \( O(1) \)-comp for all \( k \)

True for \( k=1 \) and \( k=\infty \)
Hybrid: LF-W+LF

LF-W(c)+LF

- For 9 of 10 time slots use LF-W(c)
- Use LF for the 10th time slot
Hybrid: LF-W+LF

LF-W(\(c\))+LF

* For 9 of 10 time slots use LF-W(\(c\))
* Use LF for the 10\(^{th}\) time slot

Easier to analyze than LF-W(\(c\)) and provably good!

*LF-W(1/2)+LF is \(O(1)\)-comp with \(O(1)\)-speed for all \(k\)?

[Im-Mosely’09] \((1+\varepsilon)\)-speed \(O(1/\varepsilon^{11})\) competitive algorithm for average flowtime (variant of above)
Remaining time?

♦ Sketch of LWF analysis
♦ Sketch of LF-W analysis for max weighted flowtime

The above two ingredients are key for all our results
Analysis of LWF

Several nice/original ideas in [Edmonds-Pruhs’04] but difficult to read/understand

We present a simpler view while borrowing the key ideas from [EP’04]. Allowed several subsequent improvements
Analysis of LWF

Assume LWF is given 5 speed

- Partition requests into $S$ and $N$
- $S$: self-chargeable $F_{p,i} \leq c F_{p,i}^*$
- $N$: non-self-chargeable $F_{p,i} > c F_{p,i}^*$
Analysis of LWF

♦ Partition requests into $S$ and $N$

♦ $S$: self-chargeable $F_{p,i} \leq c F^*_{p,i}$

♦ $N$: non-self-chargeable $F_{p,i} \leq c F^*_{p,i}$

From definition: $F(S) \leq c \text{ OPT}$
Analysis of LWF

- Partition requests into $S$ and $N$
  - $S$: self-chargeable $F_{p,i} \leq c F^*_p,i$
  - $N$: non-self-chargeable $F_{p,i} \leq c F^*_p,i$

From definition: $F(S) \leq c \text{OPT}$

Key idea: show $F(N) \leq \delta (F(S) + F(N)) = \delta \text{LWF}$ for $\delta < 1$

Charge part of LWF to itself!
Analysis of LWF

- Partition requests into $S$ and $N$
- $S$: self-chargeable $F_{p,i} \leq c F^*_{p,i}$
- $N$: non-self-chargeable $F_{p,i} \leq c F^*_{p,i}$

From definition: $F(S) \leq c \text{ OPT}$

Key idea: show $F(N) \leq \delta (F(S) + F(N)) = \delta \text{ LWF}$ for $\delta < 1$

$LWF = F(S) + F(N) \leq c \text{ OPT} + \delta \text{ LWF}$

therefore $LWF \leq c \text{ OPT}/(1-\delta)$
Analysis for LWF

Key idea: show \( F(N) \leq \delta (F(S) + F(N)) = \delta \text{ LWF for } \delta < 1 \)

Analyze \( N \) for each \( p \)

LWF’s x’th and (x+1)st transmission of \( p \)

OPT’s last \( p \) in \( I_{p,x} \)
Analysis for LWF

non-self chargeable requests

$N_{p,x}$ for $p$ in $l_{p,x}$

$F_{p,x}$: their total flowtime
Analysis for LWF

**Observation:** By time $t^*$, reqs in $N_{p,x}$ have accumulated flowtime $\geq \frac{1}{2} F_{p,x}$

- **OPT**
  - $p$
  - $p$
  - $t^*$

- **LWF**
  - non-self chargeable requests $N_{p,x}$ for $p$ in $I_{p,x}$
  - $F_{p,x}$: their total flowtime
Analysis for LWF

Observation: By time $t^*$, reqs in $N_{p,x}$ have accumulated flowtime $\geq \frac{1}{2} F_{p,x}$. Why did LWF do $p_i$ and not $p$ at $t$? Implies flowtime for $p_i$ at $t$ is $\geq \frac{1}{2} F_{p,x}$.
Analysis for LWF

Charge $F_{p,x}$ to flowtime of $p_1$ to $p_5$: $5 \frac{F_{p,x}}{2}$ available at $t$
Analysis for LWF

Charging scheme:

♦ Can charge $F_{p,x}$ to any $t$ in $[t^*, \text{end of } I_{p,x}]$, $\frac{5F_{p,x}}{2}$ available at $t$

♦ However, only half a time slot available; to avoid overcharging by other pages

♦ Thus $\frac{5F_{p,x}}{4}$ available to charge $F_{p,x}$

♦ Thus overall, $F(N) \leq \frac{4}{5}$ LWF
Analysis for LWF

Why can’t we charge $F_{p,x}$ to any $t$ in interval? Multiple pages may want to charge to same $t$!
Analysis for LWF

Charging scheme: why is a unique half-slot available? Use a matching argument similar to [E-P’04]

Intuition: OPT has a unique broadcast for each (p,x) in N and we use only half the interval to charge
LF for $L_k$ norm of flowtime

Easy modifications of our LWF analysis shows

LF is $O(k)$-speed $O(k)$-competitive for $L_k$ norm of flowtime for any $k \geq 1$

Also holds for $L_k$ norm of delay factor

More technical and difficult analysis shows LWF is $O(1)$-competitive with 3.44-speed

**Conjecture:** LWF is $2$-speed $O(1)$-competitive matching lower bound
Minimizing Weighted Max Flowtime

\[ \min \max_{p,i} w_{p,i} F_{p,i} \]

- **Unweighted**: FIFO is 2-competitive \cite{Changetal'08}
- **Weighted**
  - \( \Omega(W^{0.4}) \) lower bound where \( W \) is max weight even for unicast scheduling \cite{C-Moseley'09} (related to lower bound for minimizing maximizing stretch \cite{Bender-Chakrabarti-Muthukrishnan'98})
  - Need resource augmentation
Algorithm

LF-W(c):
• $F_{\text{max}}(t)$: max weighted flowtime of alive reqs at time $t$
• $Q(t) = \{ \text{alive request with } F_{p,i}(t) > F_{\text{max}}(t)/c \}$
• Schedule page $p$ with largest weight in $Q(t)$
Algorithm

LF-W(c):
• \( F_{\text{max}}(t) \): max weighted flowtime of alive reqs at time \( t \)
• \( Q(t) = \{ \text{alive request with } F_{p,i}(t) > F_{\text{max}}(t)/c \} \)
• Schedule page \( p \) with largest weight in \( Q(t) \)

Theorem: If \( c > (1+\frac{2}{\varepsilon}) \), LF-W(c) is \( c^2 \)-competitive with a \((1+\varepsilon)\)-speed server.

Corollary: \((1+\varepsilon)\)-speed \( O(1/\varepsilon^2) \)-competitive algorithm for max weighted flowtime

Note: algorithm’s parameter \( c \) depends on speed
Analysis

$t^*$: first time when some req $J_{q,k}$ has $w_{q,k} F_{q,k} > c^2 \text{OPT}$

**Key defn:** $t_1$ is smallest time such that in $I = [t_1, t^*]$ all requests $(p,i)$ done by algorithm satisfy

- $w_{p,i} F_{p,i} \geq \text{OPT}$ (flowtime worse than OPT) and
- $w_{p,i} \geq w_{q,k}$ (larger weight than $(q,k)$)
Analysis contd

\[ |I| \geq (c^2-c) x \]

\[ t_1 - 2cx \quad t_1 \quad t^* \]

\[ R = \text{requests picked to schedule} \text{ by LF-W during } I \]

\[ x = \frac{\text{OPT}}{w_{q,k}} \]

**Lemma 1:** Every request in \( R \) is satisfied by \( \text{OPT} \) by a separate broadcast (even if they are for same page).

**Lemma 2:** No request in \( R \) arrives before \( t_1 - 2cx \)

**Lemma 3:** \( I \) is long, that is \( |I| \geq (c^2-c) x \)
**Lemma 1:** Every request in $R$ is satisfied by $OPT$ by a separate broadcast (even if they are for same page).

**Lemma 2:** No request in $R$ arrives before $t_1 - 2cx$

**Lemma 3:** $I$ is long, that is $|I| \geq (c^2-c) \times t_1 - 2cx$

$|R| = (1+\epsilon) |I|$ since $LF-W$ has $(1+\epsilon)$-speed.

$OPT$ has to do all these requests in $[t_1-2cx, t^*]$ with 1 speed.

Contradiction by simple algebra if $c > (1+2/\epsilon)$
Lemma 1: Every request in $\mathbf{R}$ is satisfied by $\mathbf{OPT}$ by a separate broadcast (even if they are for same page).

Suppose $(p,i)$ and $(p,j)$ satisfied by $\mathbf{OPT}$ by same broadcast

Flowtime of $(p,i) \geq \mathbf{OPT}$ and $(p,j)$ arrives after $(p,i)$ is finished

Thus if $(p,i)$ and $(p,j)$ are merged by $\mathbf{OPT}$ then $F_{p,i}^* > \mathbf{OPT}$!
Lemma 2: No request in $R$ arrives before $t_1 - 2c x$

Suppose some request $(p,i)$ in $R$ arrived at $t < t_1 - 2c x$

Case analysis to contradict definition of $t_1$


**Analysis contd**

Lemma 3: \( I \) is long, that is \( |I| \geq (c^2-c)x \)

\[ t^* = a_{q,k} + c^2 x, \text{ define } t' = a_{q,k} + c x \]

By \( t' \), \( (q,k) \) has already accumulated \( c \) OPT flowtime,

- \( (q,k) \) is in \( Q(t) \) for all \( t \) in \([t', t^*)\) otherwise contradicts defn of \( t^* \)
- Implies \( t_1 \leq t' \) and hence \( |I| \geq (c^2-c)x \)
FIFO

- Can use LF-W analysis idea to show FIFO is 2-competitive for max flowtime even for varying sized pages
- Matches lower bound of 2 for deterministic algorithms even for unit-sized pages
- Proof is different from that of [Chang etal’08] who assume unit-sized pages and time-slot arrivals
Future Directions

✻ Offline: $O(1)$ approx for average flow-time? How bad/good is the LP relaxation?

✻ Online:
  ✷ Tight bounds for LWF. Conjecture: 2-speed $O(1)$-comp
  ✷ Simplify/improve the new scalable algorithms of [Im-Moseley’09]. Potential function based analysis?
  ✷ Prove conjecture on LF-W(1/2)
  ✷ “Understand” BEUIQ and LAPS algorithms [E-P]

✻ Empirical evaluation of recent algorithms

✻ Batch scheduling
Thanks!
Delay Factor

[Chang etal’08]

- Request $J_{p,i}$ has *deadline* $d_{p,i}$
- Slack $S_{p,i} = d_{p,i} - a_{p,i}$
- Delay factor $D_{p,i} = \max(1, \ F_{p,i} / S_{p,i})$
- 1 if job/request done before deadline, otherwise the relative delay when compared to slack
- syntactic similarity to $w_{p,i} = 1 / S_{p,i}$
- Most of our results carry over to delay factor sched