MARCIN COPIK <MARCIN.COPIK@INF.ETHZ.CH>

DPHPC, Network models

Recitation session, 12.12.2019

Admin

- **Presentations starting next Monday!**

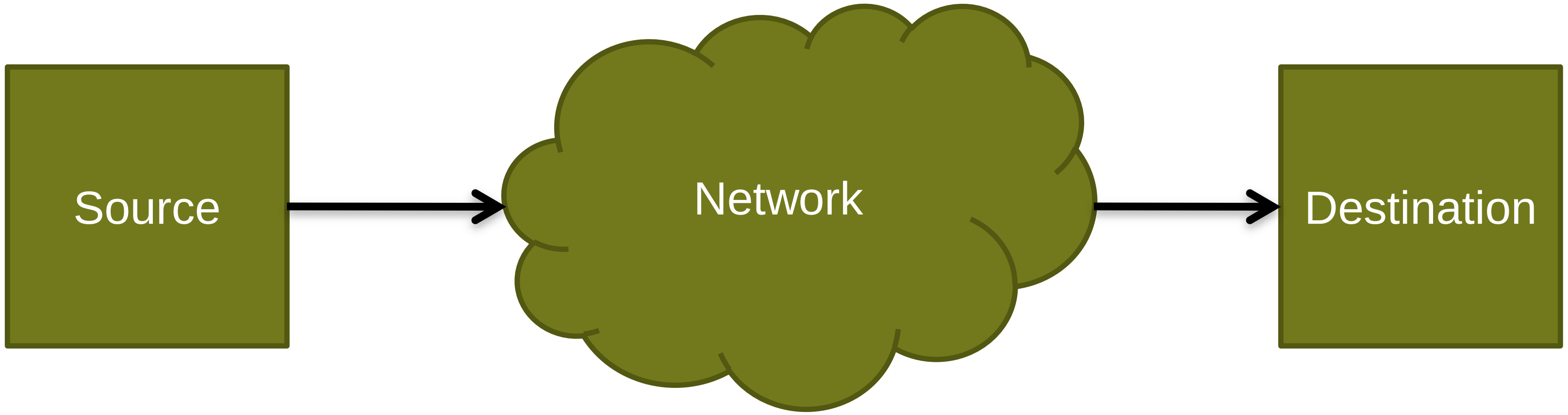
Admin

- **Presentations starting next Monday!**

- **How to give good presentations?**
  - Make multiple dry-runs!
  - Short and clear introduction & motivation
  - Plots should be self-explanatory.
  - Avoid long bullet-point lists and walls of texts.
  - PowerPoint/Impress make it quite easy and fast to create diagrams.

HPC Networking Basics

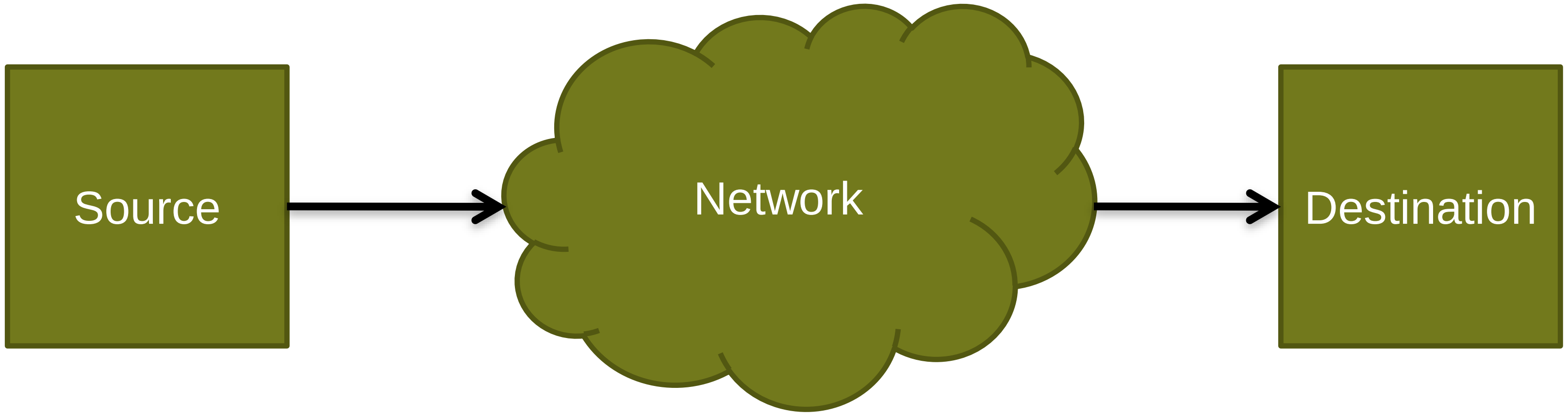
- **Familiar (non-HPC) network: Internet TCP/IP**
  - Common model:



- **Class Question: What parameters are needed to model the performance (including pipelining)?**

HPC Networking Basics

- **Familiar (non-HPC) network: Internet TCP/IP**
  - Common model:



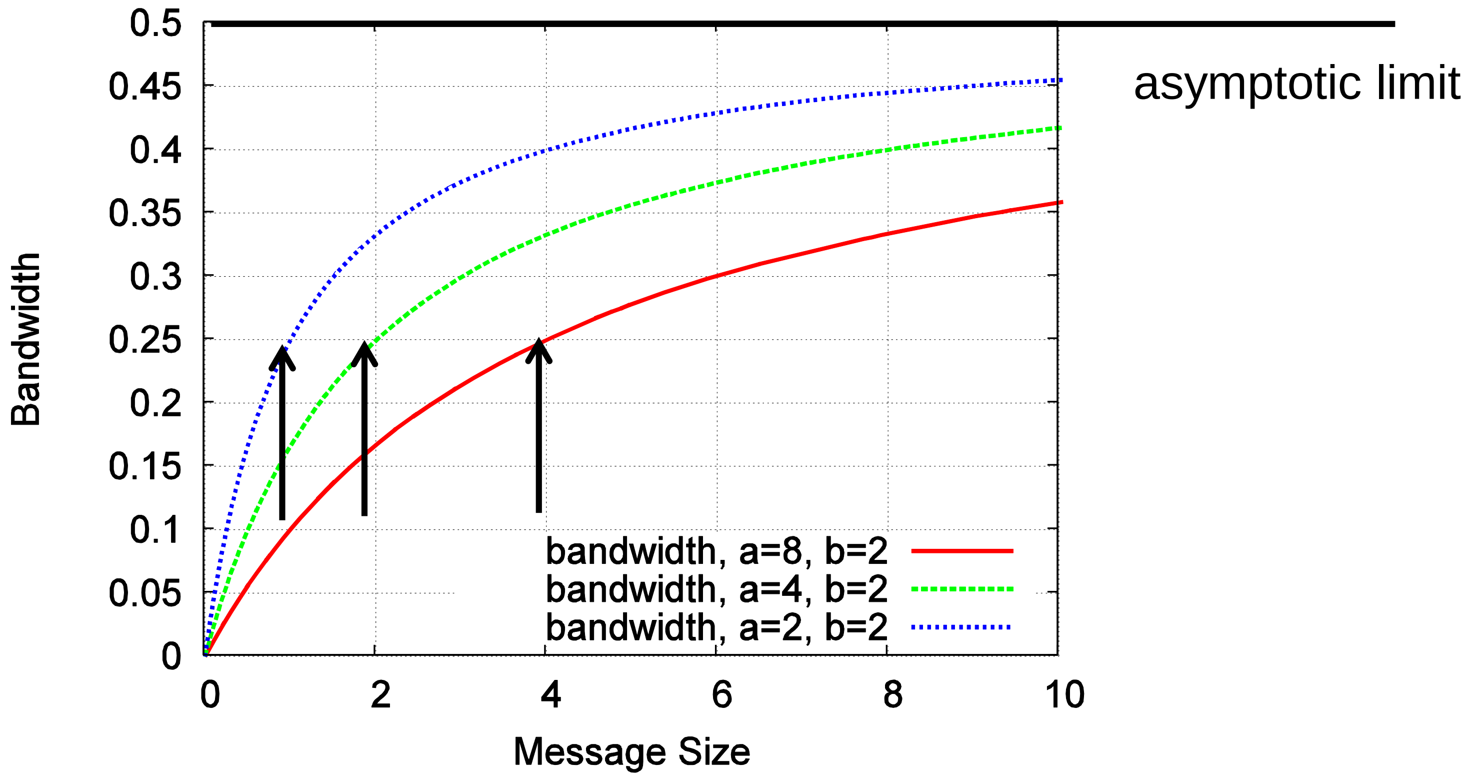
- **Class Question: What parameters are needed to model the performance (including pipelining)?**
  - Latency, Bandwidth, Injection Rate, Host Overhead

A Simple Model for Communication

- **Transfer time $T(s) = \alpha+\beta s$**
  - α = startup time (latency)
  - β = cost per byte (bandwidth=$1/\beta$)
- **As s increases, bandwidth approaches $1/\beta$ asymptotically**
  - Convergence rate depends on α
  - $s_{1/2} = \alpha/\beta$
- **Assuming no pipelining (new messages can only be issued from a process after all arrived)**

Bandwidth vs. Latency

- **$s_{1/2}$ = α/β often used to distinguish bandwidth- and latency-bound messages**
  - $s_{1/2}$ is in the order of kilobytes on real systems



Broadcast

- **Simplest linear broadcast**
  - One process has a data item to be distributed to all processes
- **Broadcasting s bytes among P processes:**
  - T(s) = $(P - 1) \times (\alpha + \beta s) = \mathcal{O}(\mathrm{P})$

- **Class question: Do we know a faster method to accomplish the same?**

k-ary tree broadcast

- Origin process is the root of the tree, passes messages to k neighbors which pass them on.

- What is the broadcast time in the simple latency/bandwidth model?

k-ary tree broadcast

- **Origin process is the root of the tree, passes messages to k neighbors which pass them on.**

- **What is the broadcast time in the simple latency/bandwidth model?**
  - $T(s) \approx \lceil log_k(P) \rceil \cdot k \cdot (\alpha + \beta \cdot s) = \mathcal{O}(log(P))$ (for fixed k)

    # of messages on each level

    # tree levels

- **What is the optimal k?**

k-ary tree broadcast

- **Origin process is the root of the tree, passes messages to k neighbors which pass them on.**

- **What is the broadcast time in the simple latency/bandwidth model?**
  - $T(s) \approx \lceil log_k(P) \rceil \cdot k \cdot (\alpha + \beta \cdot s) = \mathcal{O}(log(P))$ (for fixed k)

# of messages on each level

tree levels

- **What is the optimal k?**
  - $0 = \frac{ln(P) \cdot k}{ln(k)} \frac{d}{dk} = \frac{ln(P)ln(k) - ln(P)}{ln^2(k)} \rightarrow k = e = 2.71...$
  - Independent of P, α, β, s!

Better tree broadcast

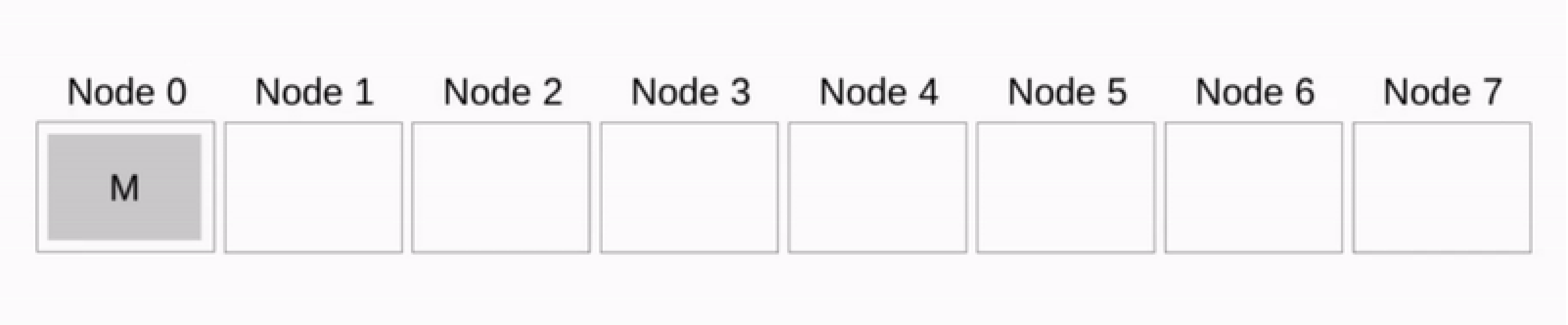
- **Class Question: Can we broadcast faster than in a ternary tree?**

Better tree broadcast

- **Class Question: Can we broadcast faster than in a ternary tree?**
  - Yes because each respective root is idle after sending three messages!
  - Those roots could keep sending!
  - Result is a k-nomial tree. For k=2, it’s a binomial tree

Better tree broadcast

- **Class Question: Can we broadcast faster than in a ternary tree?**
  - Yes because each respective root is idle after sending three messages!
  - Those roots could keep sending!
  - Result is a k-nomial tree. For k=2, it's a binomial tree



Source: Wikipedia, "Broadcast (parallel pattern)"

Better tree broadcast

- **Class Question: Can we broadcast faster than in a ternary tree?**
  - Yes because each respective root is idle after sending three messages!
  - Those roots could keep sending!
  - Result is a k-nomial tree. For k=2, it’s a binomial tree

- **Class Question: What about the runtime?**
  - $T(s) = \lceil log_k(P) \rceil \cdot (k-1) \cdot (\alpha + \beta \cdot s) = \mathcal{O}(log(P))$

Source: Wikipedia, “Broadcast (parallel pattern)”

Better tree broadcast

- **Class Question: Can we broadcast faster than in a ternary tree?**
  - Yes because each respective root is idle after sending three messages!
  - Those roots could keep sending!
  - Result is a k-nomial tree. For k=2, it's a binomial tree

- **Class Question: What about the runtime?**
  - $T(s) = \lceil log_k(P) \rceil \cdot (k-1) \cdot (\alpha + \beta \cdot s) = \mathcal{O}(log(P))$

- **Class Question: What is the optimal k here?**
  - T(s) d/dk is monotonically increasing for k>1, thus $k_{opt}=2$

Source: Wikipedia, "Broadcast (parallel pattern)"

Better tree broadcast

- **Class Question: Can we broadcast faster than in a ternary tree?**
  - Yes because each respective root is idle after sending three messages!
  - Those roots could keep sending!
  - Result is a k-nomial tree. For k=2, it's a binomial tree

- **Class Question: What about the runtime?**
  - $T(s) = \lceil log_k(P) \rceil \cdot (k-1) \cdot (\alpha + \beta \cdot s) = \mathcal{O}(log(P))$

- **Class Question: What is the optimal k here?**
  - T(s) d/dk is monotonically increasing for k>1, thus $k_{opt}=2$

- **Class Question: Can we broadcast faster than in a k-nomial tree?**
  - $\mathcal{O}(log(P))$ is asymptotically optimal for s=1!
  - But what about large s?

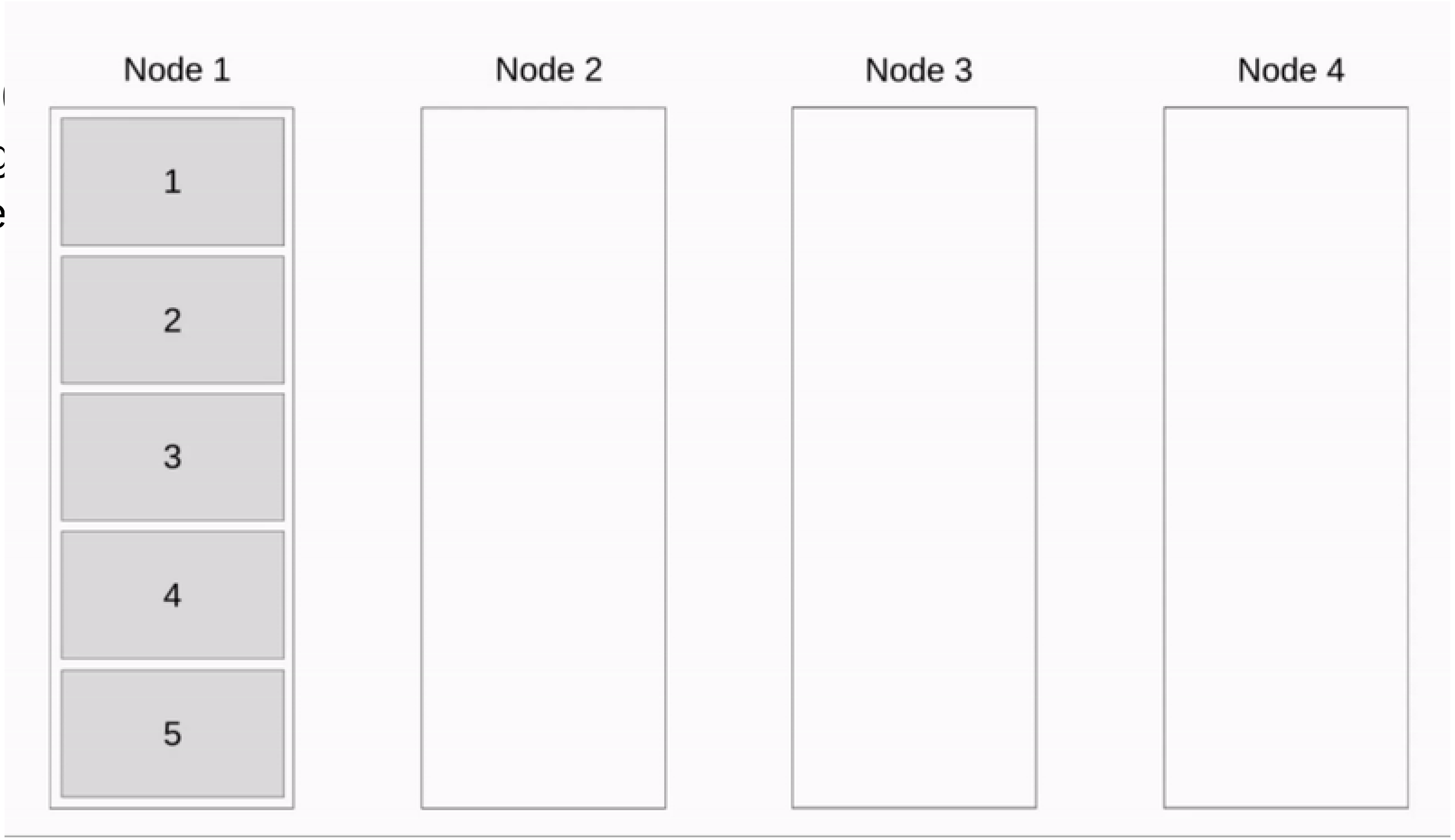
Source: Wikipedia, "Broadcast (parallel pattern)"

Very Large Message Broadcast

- **Extreme case (P small, s large): simple pipeline**
  - Split message into segments of size z
  - Send segments from PE i to PE i+1

Very Large Message Broadcast

- **Extreme case (**
  - Split messag
  - Send segme



Source: Wikipedia, “Broadcast (parallel pattern)”

Very Large Message Broadcast

- **Extreme case (P small, s large): simple pipeline**
  - Split message into segments of size z
  - Send segments from PE i to PE i+1
- **Class Question: What is the runtime?**
  - $T(s) = (P - 2 + \frac{s}{z})(\alpha + \beta z)$

Source: Wikipedia, “Broadcast (parallel pattern)”

Very Large Message Broadcast

- **Extreme case (P small, s large): simple pipeline**
  - Split message into segments of size z
  - Send segments from PE i to PE i+1
- **Class Question: What is the runtime?**
  - $T(s) = (P - 2 + \frac{s}{z})(\alpha + \beta z)$
- **Compare 2-nomial tree with simple pipeline for α=10, β=1, P=4, s=10^6, and z=10^5**
  - 2,000,020 vs. 1,200,120

Source: Wikipedia, “Broadcast (parallel pattern)”

Very Large Message Broadcast

- **Extreme case (P small, s large): simple pipeline**
  - Split message into segments of size z
  - Send segments from PE i to PE i+1
- **Class Question: What is the runtime?**
  - $T(s) = (P - 2 + \frac{s}{z})(\alpha + \beta z)$
- **Compare 2-nomial tree with simple pipeline for α=10, β=1, P=4, s=10^6, and z=10^5**
  - 2,000,020 vs. 1,200,120
- **Class Question: Can we do better for given α, β, P, s?**
  - Derive optimal z
- **What is the time for simple pipeline for α=10, β=1, P=4, s=10^6, z_{opt}?**
  - 1,008,964

Source: Wikipedia, “Broadcast (parallel pattern)”

Lower Bounds

- Class Question: What is a simple lower bound on the broadcast time?
  -

Lower Bounds

- **Class Question: What is a simple lower bound on the broadcast time?**
  - $T_{BC} \geq \min\{\lceil \log_2(P) \rceil \alpha, s\beta\}$

Lower Bounds

- **Class Question: What is a simple lower bound on the broadcast time?**
  - $T_{BC} \geq \min\{\lceil \log_2(P) \rceil \alpha, s\beta\}$
- **How close are the binomial tree for small messages and the pipeline for large messages (approximately)?**
  - Bin. tree is a factor of $\log_2(P)$ slower in bandwidth
  - Pipeline is a factor of $P/\log_2(P)$ slower in latency
- **Class Question: What can we do for intermediate message sizes?**

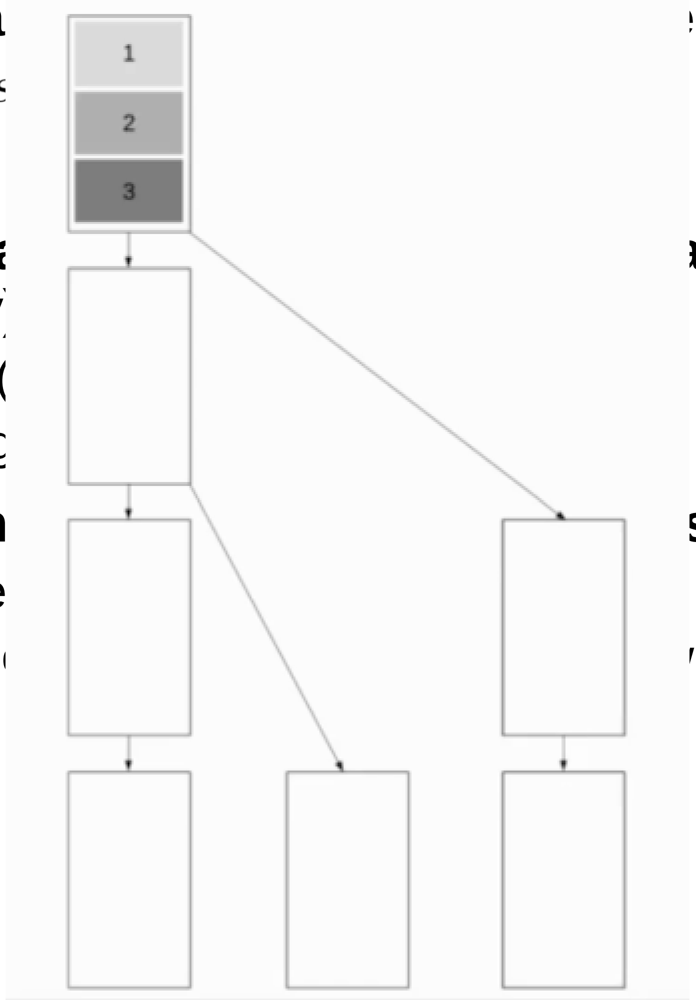
Lower Bounds

- **Class Question: What is a simple lower bound on the broadcast time?**
  - $T_{BC} \geq \min\{\lceil \log_2(P) \rceil \alpha, s\beta\}$

- **How close are the binomial tree for small messages and the pipeline for large messages (approximately)?**
  - Bin. tree is a factor of $\log_2(P)$ slower in bandwidth
  - Pipeline is a factor of $P/\log_2(P)$ slower in latency
- **Class Question: What can we do for intermediate message sizes?**
  - Combine pipeline and tree → pipelined tree
  - Achieve low latency for short messages and decent bandwidth for large ones.

Lower Bounds

- **Class Question: What is a** **broadcast time?**
  - $T_{BC} \geq \min\{\lceil \log_2(P) \rceil \alpha, s$
- **How close are the binomi** **and the pipeline for large messages (approximately)**
  - Bin. tree is a factor of $\log_2($
  - Pipeline is a factor of P/log
- **Class Question: What can** **ssage sizes?**
  - Combine pipeline and tree
  - Achieve low latency for sh idth for large ones.



Lower Bounds

- **Class Question: What is a simple lower bound on the broadcast time?**
  - $T_{BC} \geq \min\{\lceil \log_2(P) \rceil \alpha, s\beta\}$
- **How close are the binomial tree for small messages and the pipeline for large messages (approximately)?**
  - Bin. tree is a factor of $\log_2(P)$ slower in bandwidth
  - Pipeline is a factor of $P/\log_2(P)$ slower in latency
- **Class Question: What can we do for intermediate message sizes?**
  - Combine pipeline and tree → pipelined tree
  - Achieve low latency for short messages and decent bandwidth for large ones.
- **Class Question: What is the runtime of the pipelined binary tree algorithm?**

Lower Bounds

- **Class Question: What is a simple lower bound on the broadcast time?**
  - $T_{BC} \geq \min\{\lceil \log_2(P) \rceil \alpha, s\beta\}$

- **How close are the binomial tree for small messages and the pipeline for large messages (approximately)?**
  - Bin. tree is a factor of $\log_2(P)$ slower in bandwidth
  - Pipeline is a factor of $P/\log_2(P)$ slower in latency
- **Class Question: What can we do for intermediate message sizes?**
  - Combine pipeline and tree → pipelined tree
  - Achieve low latency for short messages and decent bandwidth for large ones.
- **Class Question: What is the runtime of the pipelined binary tree algorithm?**
  - $T \approx \left(\frac{s}{z} + \lceil \log_2 P \rceil - 2\right) \cdot 2 \cdot (\alpha + z\beta)$

Lower Bounds

- **Class Question: What is a simple lower bound on the broadcast time?**
  - $T_{BC} \geq \min\{\lceil \log_2(P) \rceil \alpha, s\beta\}$
- **How close are the binomial tree for small messages and the pipeline for large messages (approximately)?**
  - Bin. tree is a factor of $\log_2(P)$ slower in bandwidth
  - Pipeline is a factor of $P/\log_2(P)$ slower in latency
- **Class Question: What can we do for intermediate message sizes?**
  - Combine pipeline and tree → pipelined tree
  - Achieve low latency for short messages and decent bandwidth for large ones.
- **Class Question: What is the runtime of the pipelined binary tree algorithm?**
  - $T \approx \left(\frac{s}{z} + \lceil \log_2 P \rceil - 2\right) \cdot 2 \cdot (\alpha + z\beta)$
- **Class Question: What is the optimal z?**
  - $z_{opt} = \sqrt{\frac{\alpha s}{\beta(\lceil \log_2 P \rceil - 2)}}$

Towards an Optimal Algorithm

- **What is the complexity of the pipelined tree with z_{opt} for small s, large P and for large s, constant P?**
  - Small messages, large P: s=1; z=1 (s≤z), will give O(log P)
  - Large messages, constant P: assume α, β, P constant, will give asymptotically O(sβ)

    Asymptotically optimal for large P and s but bandwidth is off by a factor of 2! Why?

Towards an Optimal Algorithm

- **What is the complexity of the pipelined tree with z_{opt} for small s, large P and for large s, constant P?**
  - Small messages, large P: s=1; z=1 (s≤z), will give O(log P)
  - Large messages, constant P: assume α, β, P constant, will give asymptotically O(sβ)

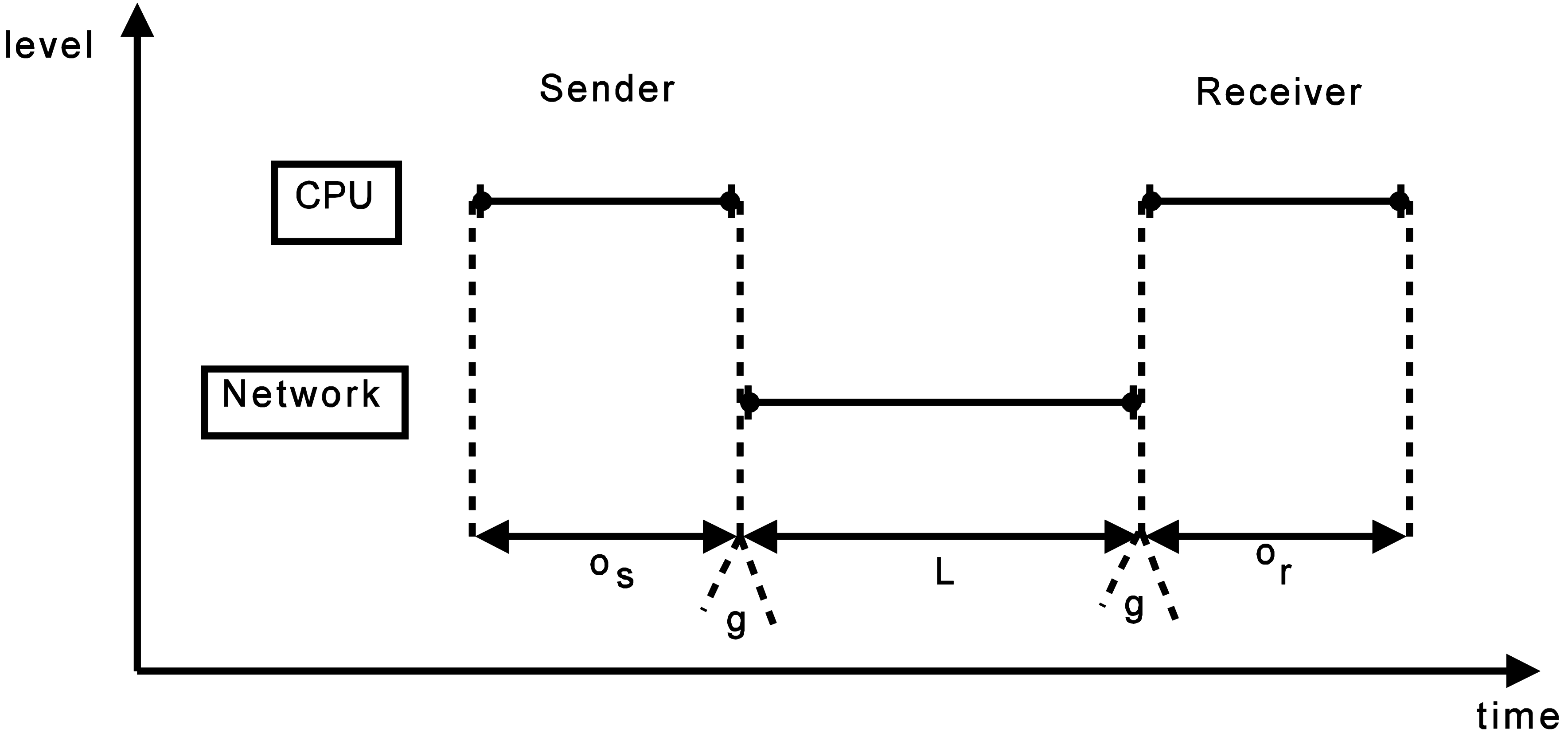
    Asymptotically optimal for large P and s but bandwidth is off by a factor of 2! Why?

- **Bandwidth-optimal algorithms exist, e.g., Sanders et al. *"Full Bandwidth Broadcast, Reduction and Scan with Only Two Trees"*. 2007**
  - Intuition: in binomial tree, all leaves (P/2) only receive data and never send → wasted bandwidth
  - Send along two simultaneous binary trees where the leafs of one tree are inner nodes of the other
  - Construction needs to avoid endpoint congestion (makes it complex)

The LogP Model

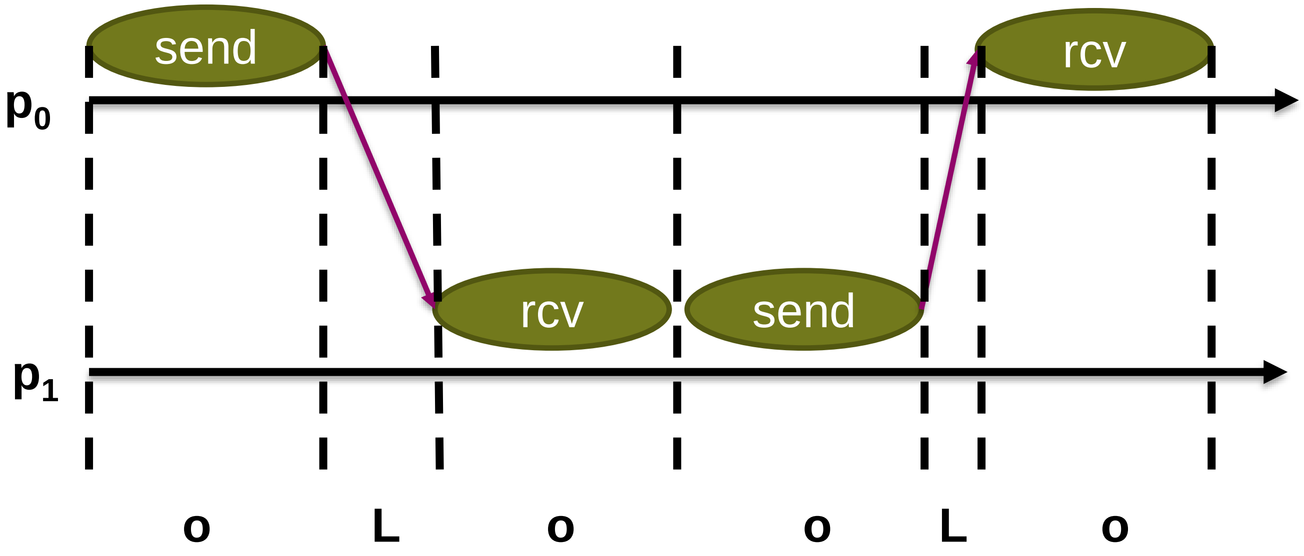
- **Defined by four parameters:**
  - L: an upper bound on the latency, or delay, incurred in communicating a message containing a word (or small number of words) from its source module to its target module.
  - o: the overhead, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations.
  - g: the gap, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a processor. The reciprocal of g corresponds to the available per-processor communication bandwidth.
  - P: the number of processor/memory modules. We assume unit time for local operations and call it a cycle.

The LogP Model

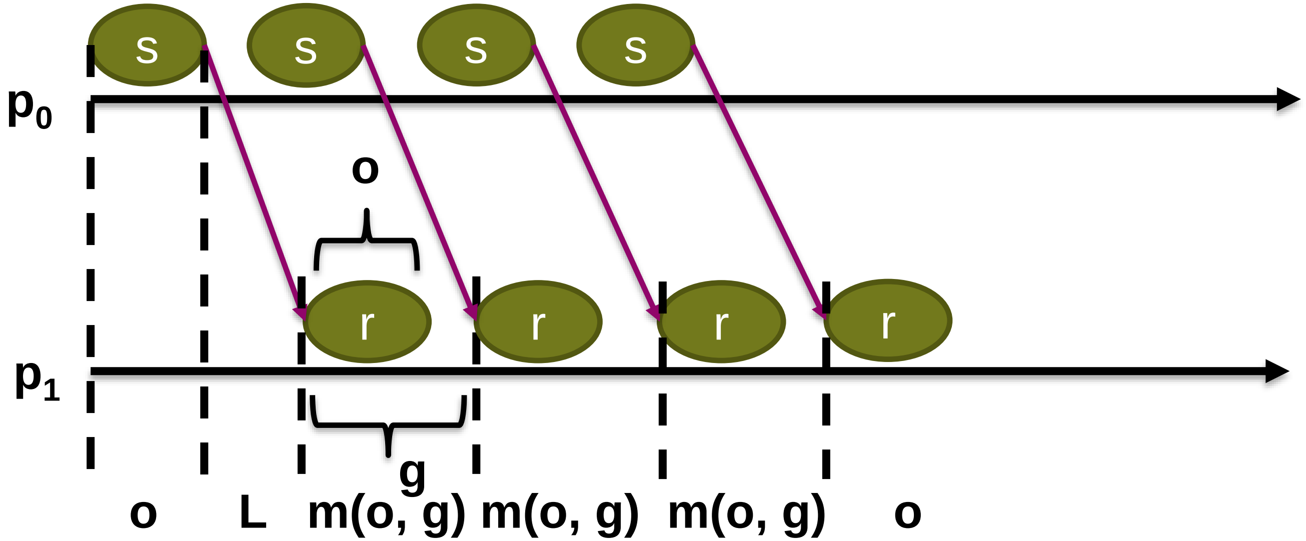


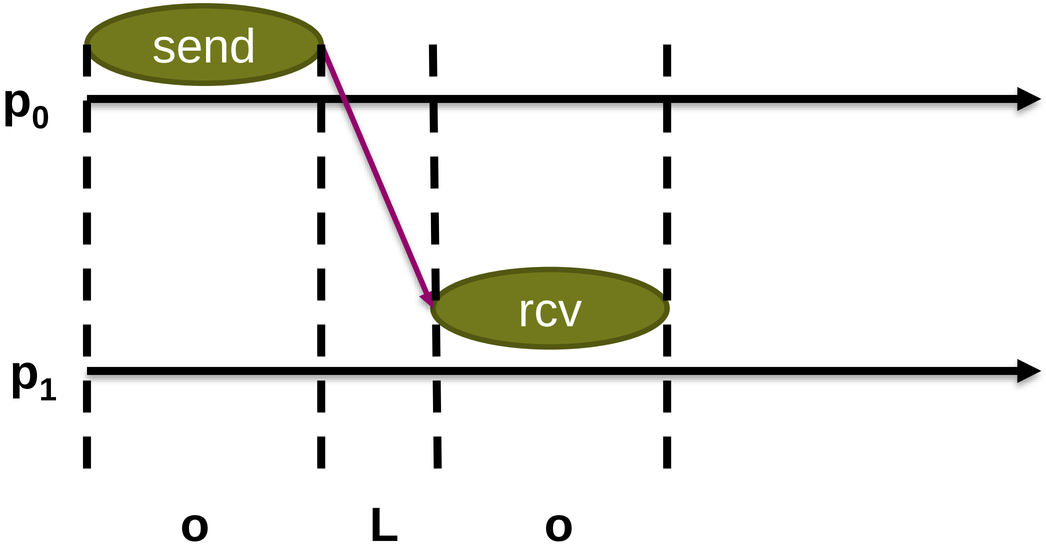
Simple Examples

- **Sending a single message**
  - $T = 2o + L$



- **Ping-Pong Round-Trip**
  - $T = 4o + 2L$
- **Transmitting n messages**
  - $T = L + (n - 1) * \max(g, o) + 2o$





Simplifications

- **o is bigger than g on some machines**
  - g can be ignored (eliminates max() terms)
  - be careful with multicore!
- **Offloading networks might have very low o**
  - Can be ignored (not yet but hopefully soon)
- **L might be ignored for long message streams**
  - If they are pipelined
- **Account g also for the first message**
  - Eliminates “-1”

Benefits

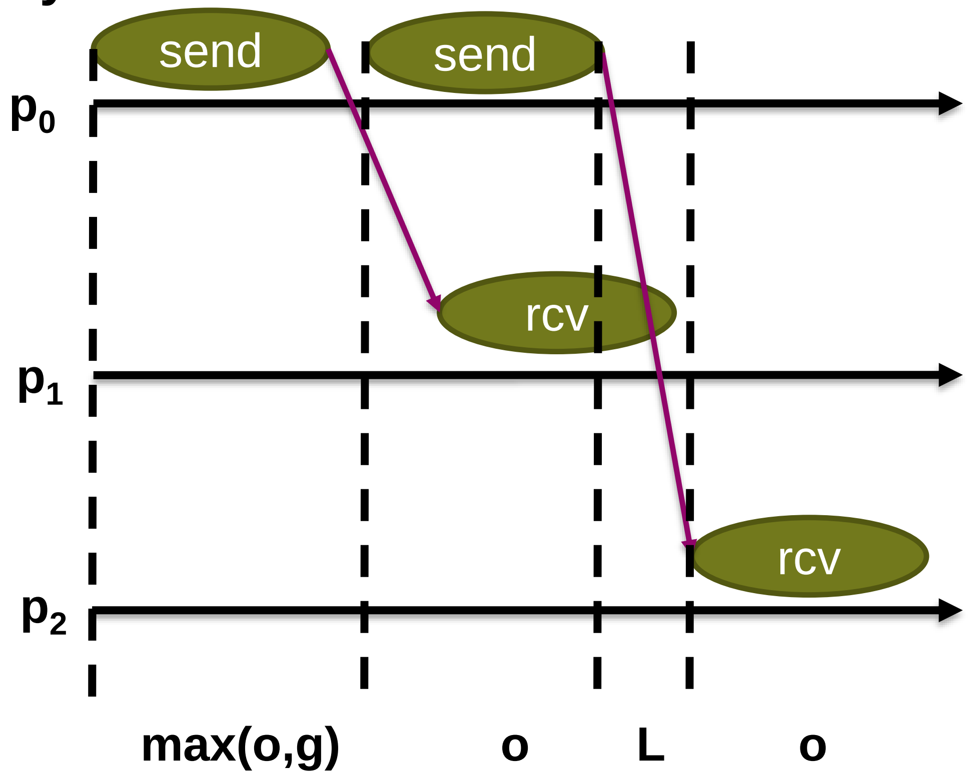
- **Models pipelining**
  - How to model N incoming messages in alfa-beta model?
  - Finite capacity of network L/g messages can be "in flight"
  - Captures state of the art (cf. TCP windows)
- **Models computation/communication overlap**
  - CPU and NIC operations are not necessarily serialized.
  - Asynchronous algorithms
- **Models endpoint congestion/overload**
  - Can the CPU/NIC process a sequence of incoming packages?
  - Benefits balanced algorithms

Example: Broadcast

- **Class Question: What is the LogP running time for a linear broadcast of a single packet?**
  - $T = L + (P - 2) * \max(o, g) + 2o$

- **Class Question: Approximate the LogP runtime for a binary-tree broadcast of a single packet?**
  - $T \leq \log_2 P * (L + \max(o, g) + 2o)$

- **Class Question: Approximate the LogP runtime for an k-ary-tree broadcast of a single packet?**
  - $T \leq \log_k P * (L + (k - 1) * \max(o, g) + 2o)$



Example: Broadcast

- **Class Question: Approximate the LogP runtime for a binomial tree broadcast of a single packet (assume L > g)?**
  - $T \leq \log_2 P * (L + 2o)$

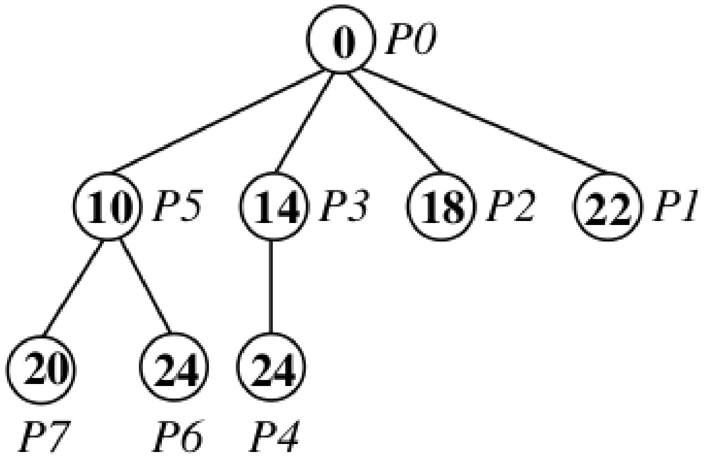
- **Class Question: Approximate the LogP runtime for a k-nomial tree broadcast of a single packet?**
  - $T \leq \log_k P * (L + (k-2) * \max(o, g) + 2o)$

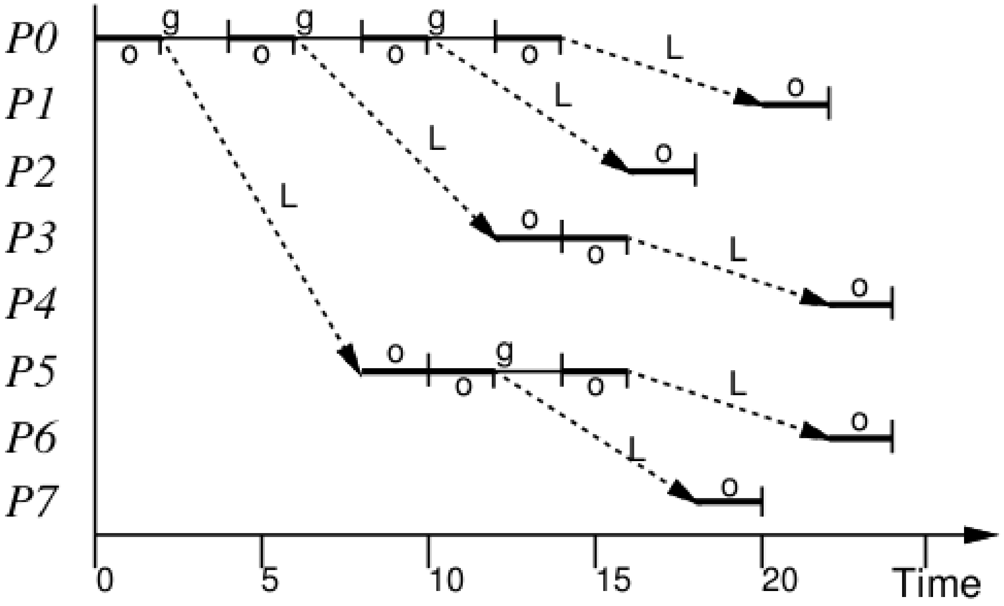
Broadcast: can we do better?

- **Can we do better than k_{opt}-ary binomial broadcast?**
  - Problem: fixed k in all stages might not be optimal
  - We can construct a schedule for the optimal broadcast in practical settings
  - First proposed by Karp et al. in “Optimal Broadcast and Summation in the LogP Model”

Optimal broadcast

- **Broadcast to P-1 processes**
  - Each process who received the value sends it on; each process receives exactly once





Source: Culler et al., "LogP: towards a realistic model of parallel computation."

Optimal broadcast runtime

- **This determines the maximum number of PEs (P(t)) that can be reached in time t**
- **P(t) can be computed with a generalized Fibonacci recurrence (assuming o> g):**

$$P(t) = \begin{cases} 1 : & t < 2o + L \\ P(t-o) + P(t-L-2o) : & \text{otherwise.} \end{cases} \qquad (1)$$

How many processors could receive message sent L + 2o time ago?

- **Which can be bounded by (see [1]):**

$$2^{\left\lfloor \frac{t}{L+2o} \right\rfloor} \leq P(t) \leq 2^{\left\lfloor \frac{t}{o} \right\rfloor}$$

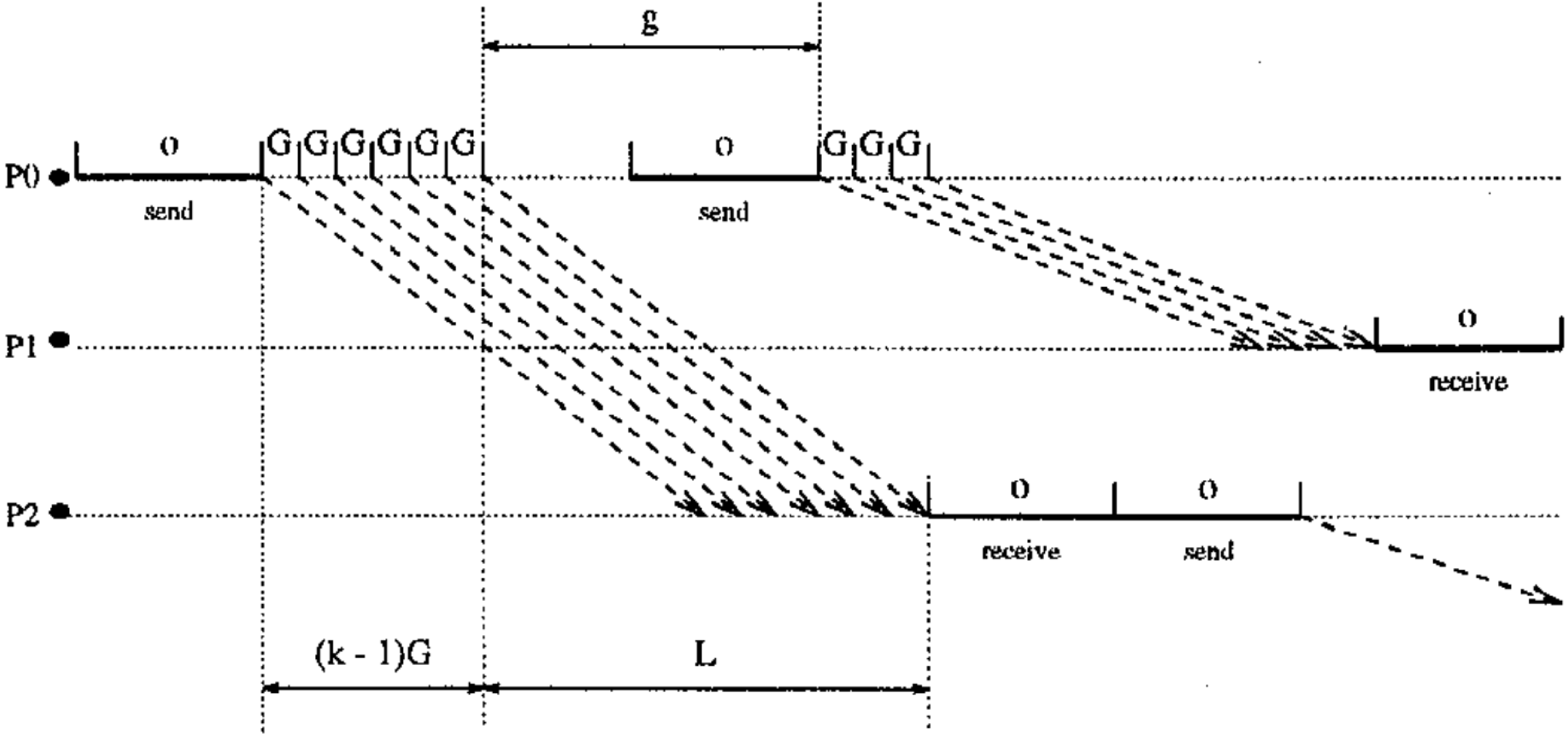
How many processors are processing receive from o time ago?

[1]: Hoefler et al.: “Scalable Communication Protocols for Dynamic Sparse Data Exchange” (Lemma 1)

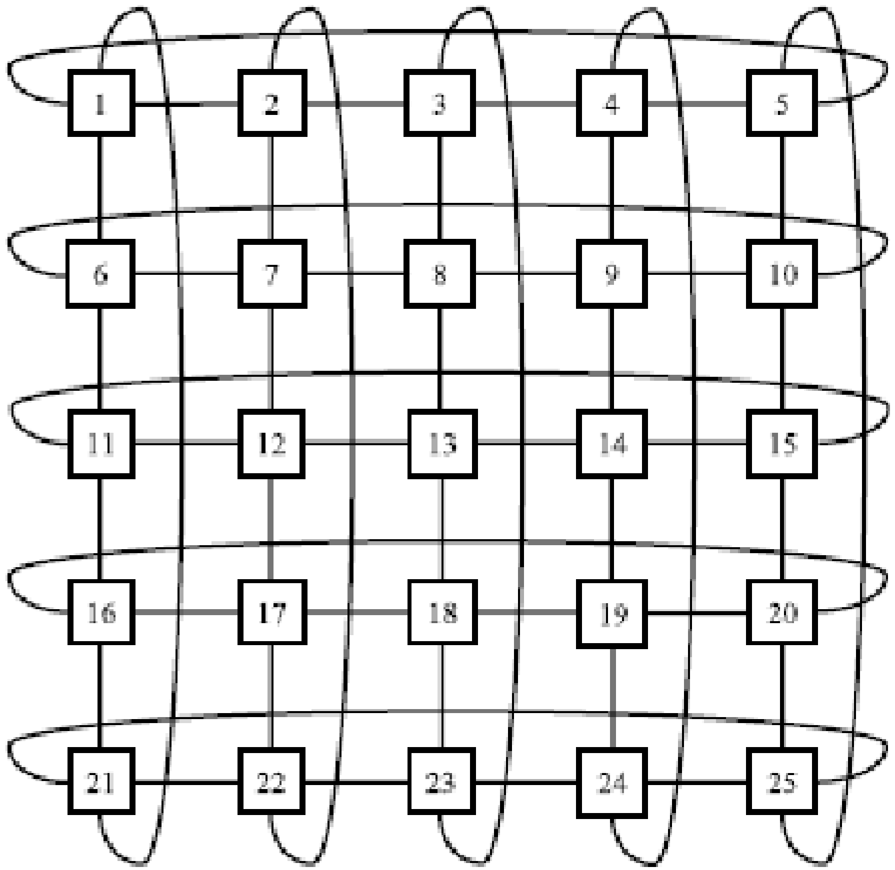
The LogGP Model

- **Defined by four parameters:**
  - L: an upper bound on the latency, or delay, incurred in communicating a message containing a word (or small number of words) from its source module to its target module.
  - o: the overhead, defined as the length of time that a processor is engaged in the transmission or reception of each message; during this time, the processor cannot perform other operations.
  - g: the gap, defined as the minimum time interval between consecutive message transmissions or consecutive message receptions at a processor. The reciprocal of g corresponds to the available per-processor communication bandwidth.
  - **G: the Gap per byte for long messages, defined as the time per byte for a long message. The reciprocal of G characterizes the available per processor communication bandwidth for long messages.**
  - P: the number of processor/memory modules. We assume unit time for local operations and call it a cycle.

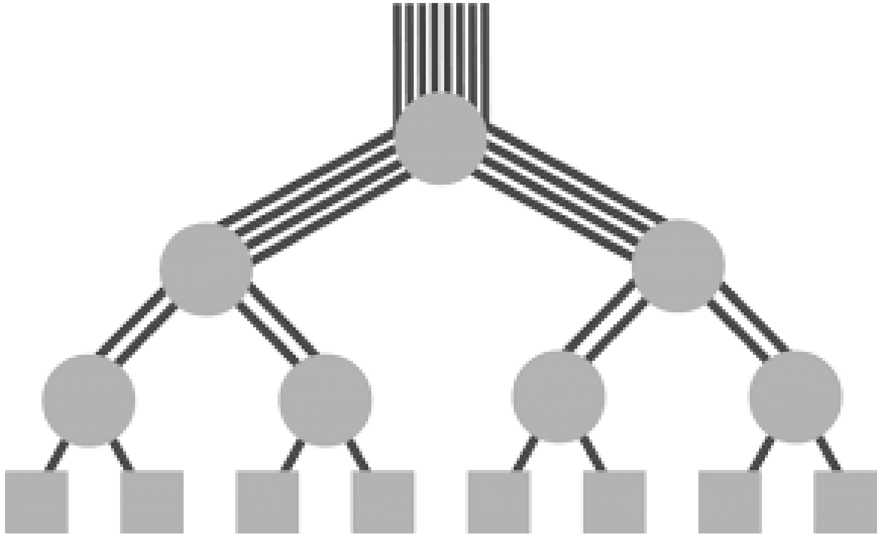
The LogGP Model



Network topologies



2D Torus



Fat tree

Graph Metrics

- **Diameter**
  - What is the maximum distance between any two nodes?

Topology	Symbol	Example System	Diameter
3-dimensional torus [3]	T3D	Cray Gemini [3]	$\lceil 3/2 \sqrt[3]{N_r} \rceil$
5-dimensional torus [9]	T5D	IBM BlueGene/Q [8]	$\lceil 5/2 \sqrt[5]{N_r} \rceil$
Hypercube [42]	HC	NASA Pleiades [42]	$\lceil \log_2 N_r \rceil$
3-level fat tree [30]	FT-3	Tianhe-2 [15]	4
3-level Flat. Butterfly [27]	FBF-3	-	3
Dragonfly topologies [28]	DF	IBM PERCS [4]	3
Random topologies [29]	DLN	-	3–10
Long Hop topologies [39]	LH-HC	Infinetics Systems [39]	4–6
Slim Fly MMS	**SF**	-	**2**

TABLE II: Topologies compared in the paper, their diameters (§ III-A), and example existing HPC systems that use respective topologies.

Besta et al.: “Slim Fly: A Cost Effective Low-Diameter Network Topology”

Graph Metrics

- **Average distance**

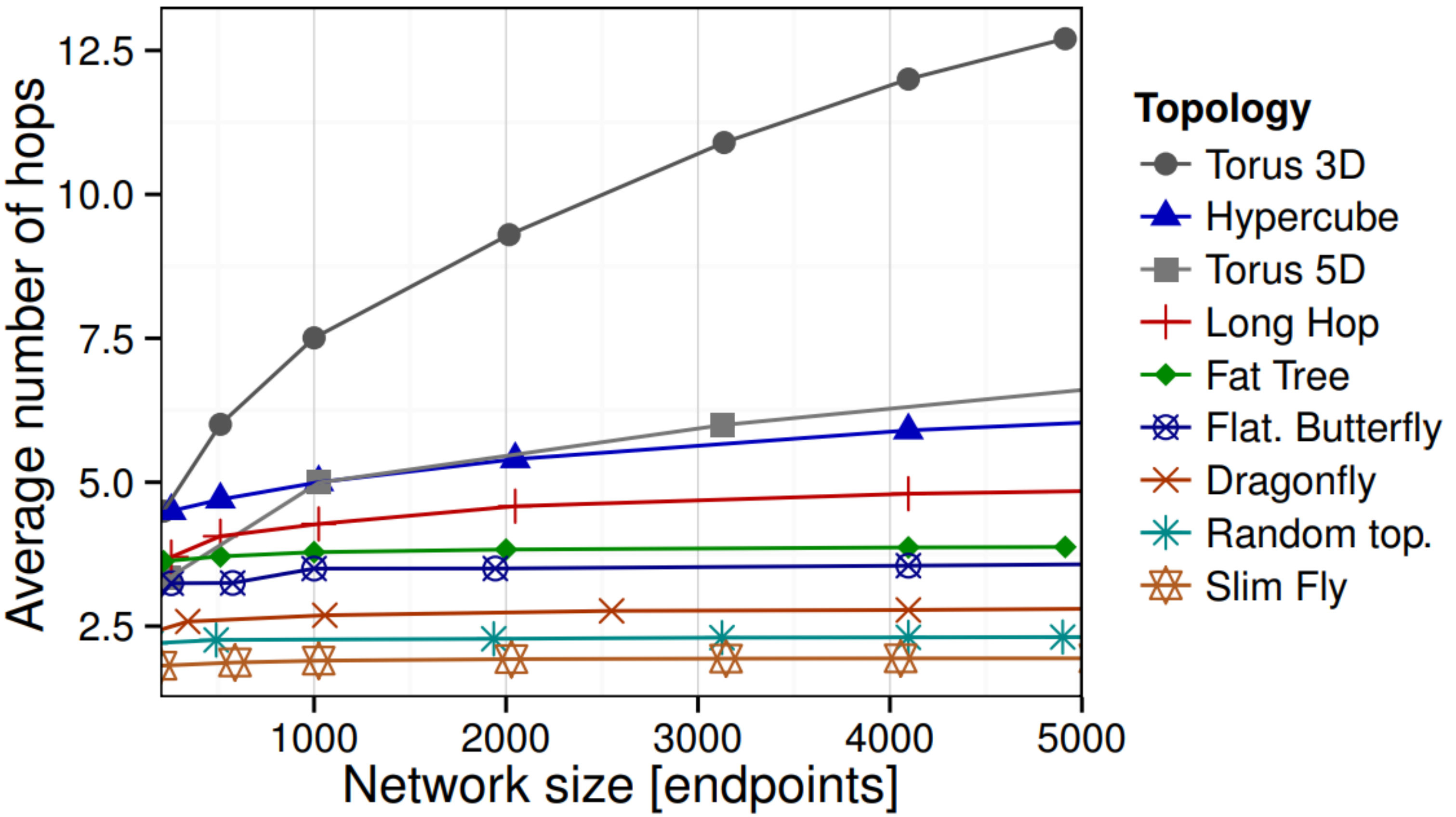
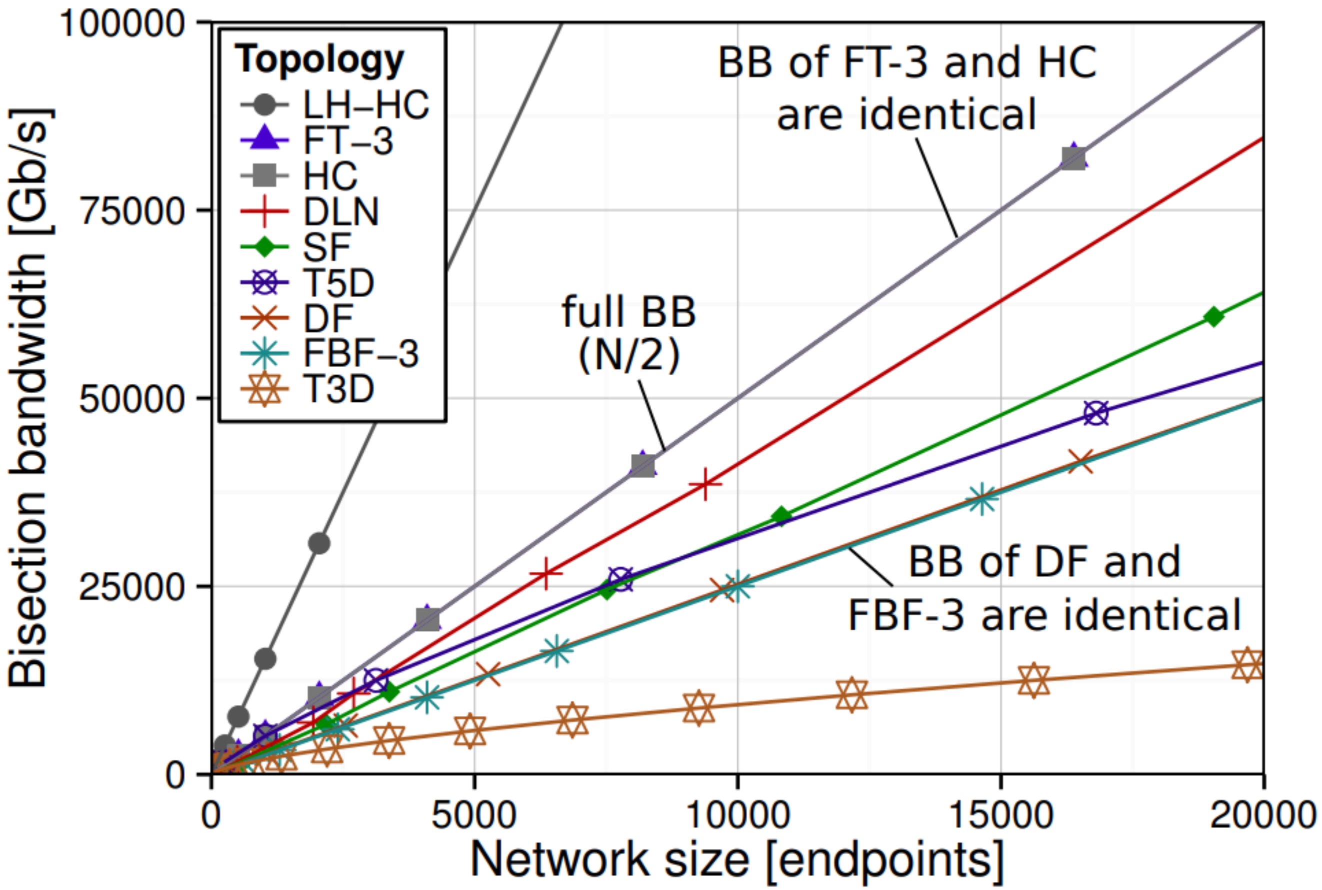


Fig. 1: Comparison of the average number of hops (uniform traffic) in Slim Fly and other networks. Topologies are in balanced or close to balanced configurations (explained in Section III), allowing for highest global bandwidth.[1]

Besta et al.: "Slim Fly: A Cost Effective Low-Diameter Network Topology"

Graph Metrics

- **Bisection bandwidth**
  - If we cut a graph into two partitions, what's the bandwidth between them? Find the minimum!
  - Reveals true bandwidth of the network – potential bottleneck.



(c) Bisection bandwidth (BB) comparison (§ III-C).

Besta et al.: "Slim Fly: A Cost Effective Low-Diameter Network Topology"

Good luck!

marcin.copik@inf.ethz.ch