CS 498
Hot Topics in High Performance Computing

Networks and Fault Tolerance

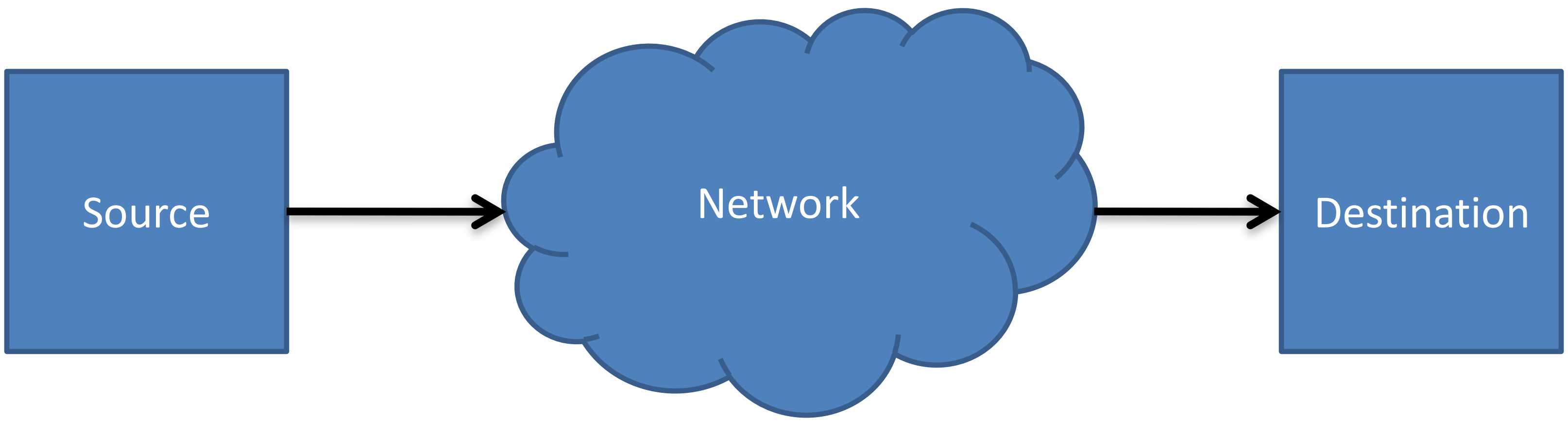
4. HPC Networking Basics

Intro

- What did we learn in the last lecture
  - Linear, k-ary tree, k-nomial tree, pipeline, pipelined tree algorithms and runtimes
  - Asymptotic optimality for broadcast
  - Deriving an asymptotically optimal algorithm
- What will we learn today
  - The LogP model and examples (more broadcasts)
  - Analyzing a parallel Fast Fourier Transform in LogP

Section III – HPC Networking Basics

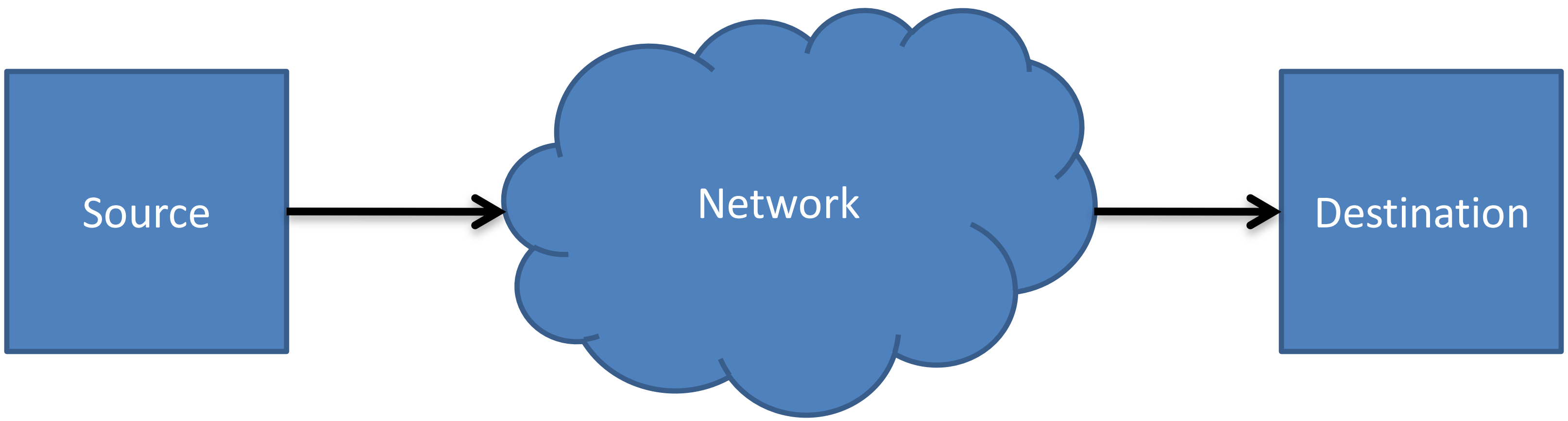
- Familiar (non-HPC) network: Internet TCP/IP
  - Common model:



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

Section III – HPC Networking Basics

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

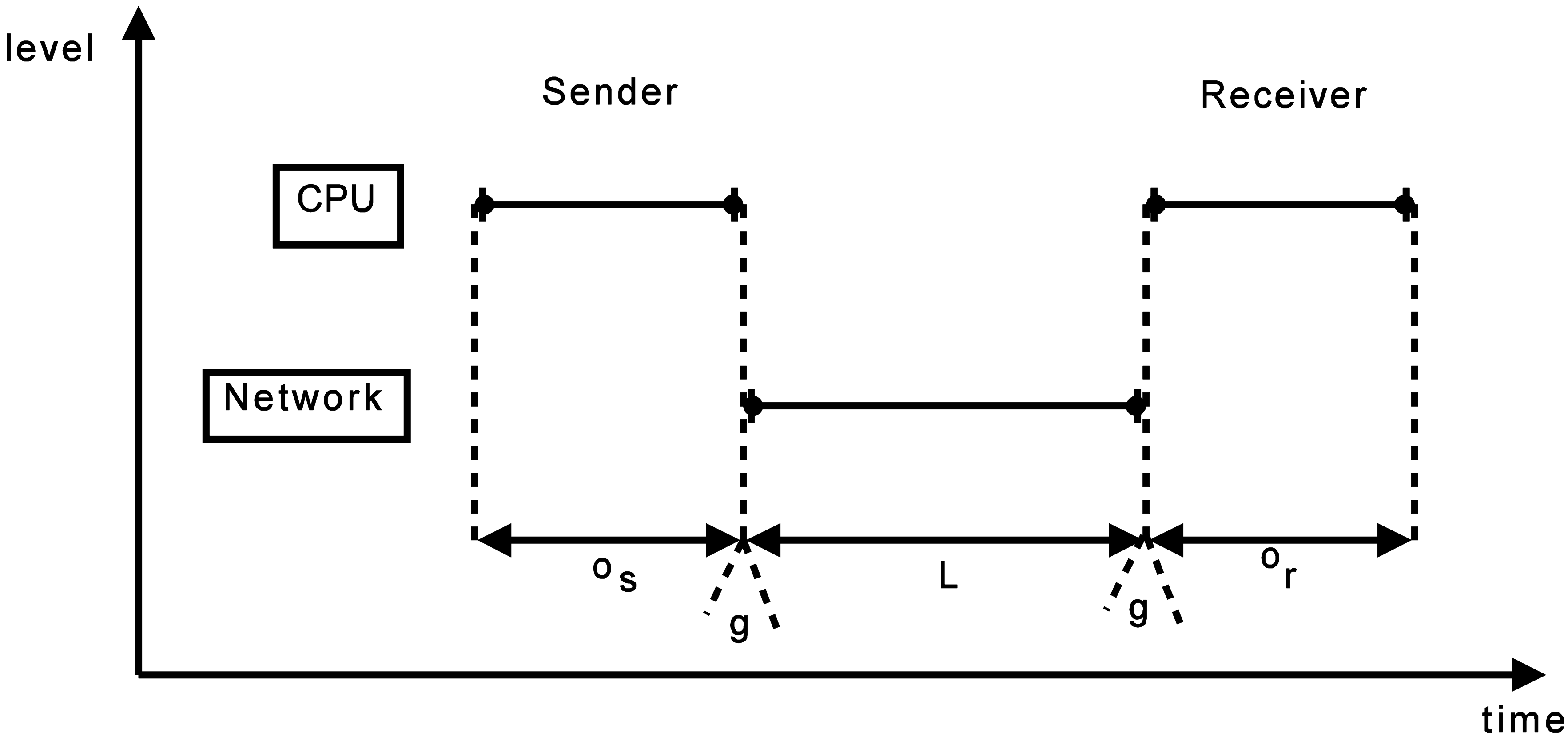


- Class Question: What parameters are needed to model the performance (including pipelining)?
  - Latency, Bandwidth, Injection Rate, Host Overhead

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
  - $T_{RTT} = 4o+2L$
- Transmitting n messages
  - $T(n) = L+(n-1)*\max(g, o) + 2o$

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 over Latency/Bandwidth Model

- Models pipelining
  - L/g messages can be "in flight"
  - Captures state of the art (cf. TCP windows)
- Models computation/communication overlap
  - Asynchronous algorithms
- Models endpoint congestion/overload
  - Benefits balanced algorithms

Examples: Broadcasts

- Class Question: What is the LogP running time for a linear broadcast of a single packet?

Example: Broadcasts

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

Example: Broadcasts

- Class Question: What is the LogP running time for a linear broadcast of a single packet?
  - $T_{lin} = L + (P-2) * \max(o,g) + 2o$
- Class Question: What is the LogP running time for a binary-tree broadcast of a single packet?
  - $T_{bin} \leq \log_2 P * (L + \max(o,g) + 2o)$
- Class Question: What is the LogP running time for an k-ary-tree broadcast of a single packet?

Example: Broadcasts

- Class Question: What is the LogP running time for a linear broadcast of a single packet?
  – $T_{lin} = L + (P-2) * \max(o,g) + 2o$
- Class Question: What is the LogP running time for a binary-tree broadcast of a single packet?
  – $T_{bin} \leq \log_2 P * (L + \max(o,g) + 2o)$
- Class Question: What is the LogP running time for an k-ary-tree broadcast of a single packet?
  - $T_{k\text{-}n} \leq \log_k P * (L + (k-1)\max(o,g) + 2o)$

Example: Broadcasts

- Class Question: What is the LogP running time for a binomial tree broadcast of a single packet?

Example: Broadcasts

- Class Question: What is the LogP running time for a binomial tree broadcast of a single packet?
  - $T_{bin} \leq \log_2 P * (L + 2o)$ (assuming L > g!)
- Class Question: What is the LogP running time for a k-nomial tree broadcast of a single packet?

Example: Broadcasts

- Class Question: What is the LogP running time for a binomial tree broadcast of a single packet?
  - $T_{bin} \leq \log_2 P * (L + 2o)$ (assuming $L > g$!)
- Class Question: What is the LogP running time for a k-nomial tree broadcast of a single packet?
  - $T_{k\text{-}n} \leq \log_k P * (L + (k-1)\max(o,g) + 2o)$
- Class Question: What is the optimal k?

Example: Broadcasts

- Class Question: What is the LogP running time for a binomial tree broadcast of a single packet?
  - $T_{bin} \leq \log_2 P * (L + 2o)$ (assuming $L > g$!)
- Class Question: What is the LogP running time for a k-nomial tree broadcast of a single packet?
  - $T_{k\text{-}n} \leq \log_k P * (L + (k-1)\max(o,g) + 2o)$
- Class Question: What is the optimal k?
  - Derive by $k \rightarrow 0 = k_{opt} \ln(k_{opt}) - L - k_{opt} o - o$ (solve numerically)
  - Models pipelining capability better than simple model!

Example: Broadcasts

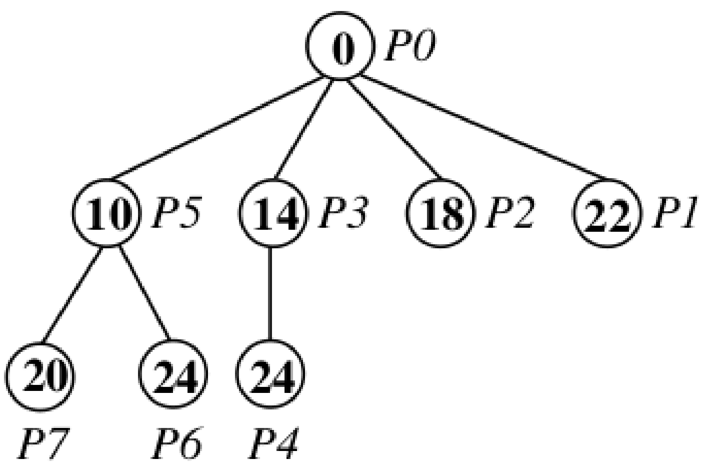
- Class Question: Can we do better than k_{opt}-ary binomial broadcast?

Example: Broadcasts

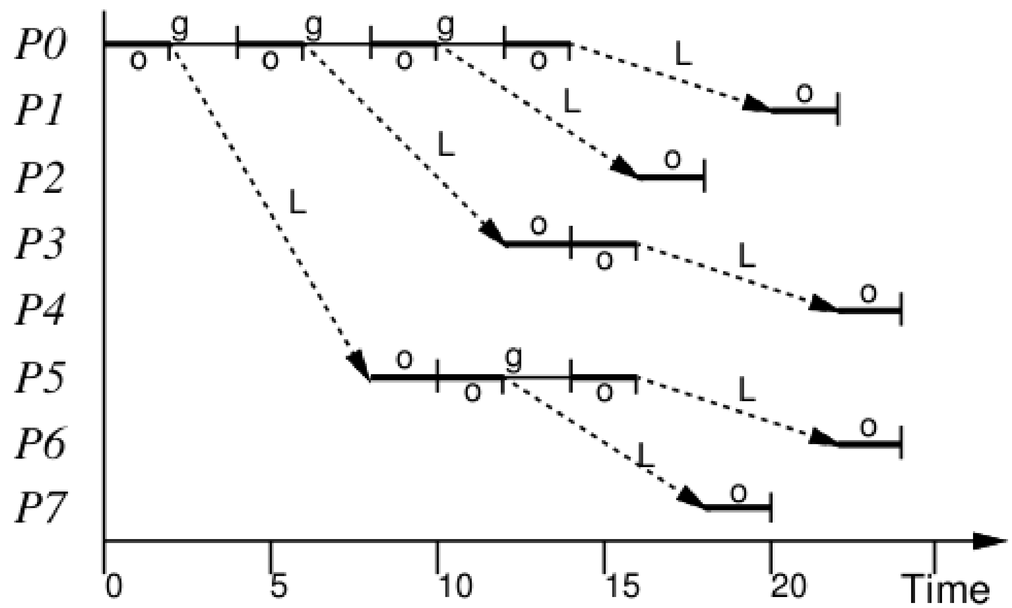
- Class Question: Can we do better than k_{opt}-ary binomial broadcast?
  - Problem: fixed k in all stages might not be optimal
  - Improves only by a constant!
  - But we can construct a schedule for the optimal broadcast in practical settings
  - First proposed by Karp et al. in "Optimal Broadcast and Summation in the LogP Model"

Example: Optimal Broadcast

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



P=8, L=6, g=4, o=2



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):

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

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

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

  – A closed solution is an interesting open problem!

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

Algorithm Design: FFT

- Assuming n (power of 2) inputs and butterfly radix-2 FFT DAG (Cooley&Tukey)
- DAG has n(log n+1) nodes arranged in n rows and log n+1 columns
- For $0 \le r < n$ and $0 \le c < \log(n)$, vertex (r,c) has edges to vertex (r,c+1) and (r_c',c+1) where r_c' is determined by negating the (c+1)-th bit in r
- Each non-input node represents a complex operation, each edge communication

Parallel Data Layout

- Block decomposition (w.l.o.g, assuming P%n=0):
  - Assign i-th n/P rows to process i-1
  - First log(P) columns require remote data
  - Last log(n/P) columns require no communication
- Times:
  - T_{comp} = n/P log(n) compute steps
  - T_{comm} = (g*n/P+L) log(P) communication (assuming g>2o [1])

[1]: Culler et al.: “LogP: towards a realistic model of parallel computation”

Parallel Data Layout

- Cyclic distribution (w.l.o.g, assuming P%n=0):
  - Assign i-th row to process i%P
  - First log(n/P) columns require no communication
  - Last log(P) columns require remote data
- Times:
  - T_{comp} = n/P log(n) compute steps
  - T_{comm} = (g*n/P+L) log(P) (assuming g>2o [1])

[1]: Culler et al.: "LogP: towards a realistic model of parallel computation"

Optimal Layout?

- Class Question: How would you arrange the n elements on P processes?