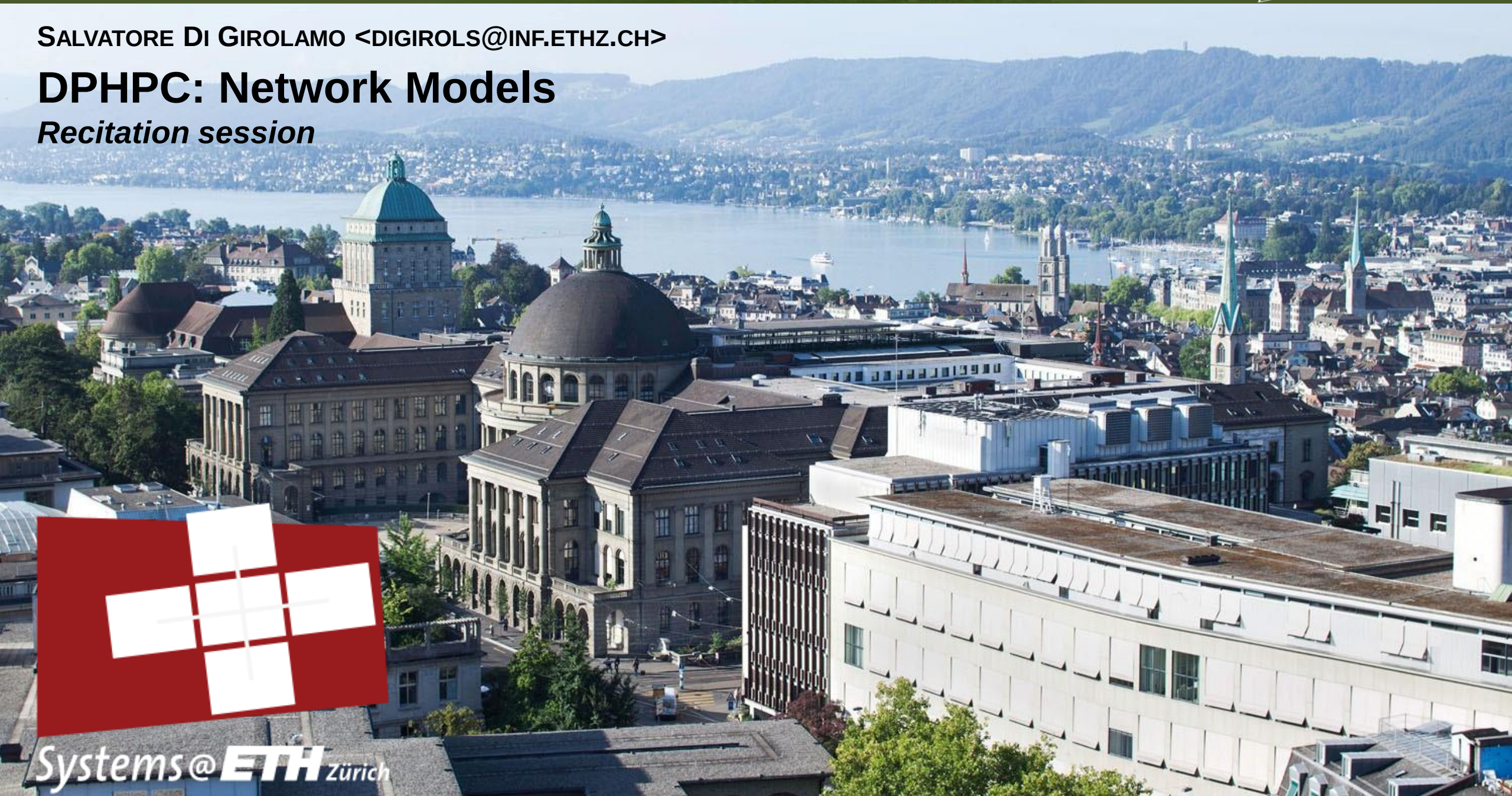
SALVATORE DI GIROLAMO <DIGIROLS@INF.ETHZ.CH>

DPHPC: Network Models

Recitation session



Administrivia

- **Final presentations: Monday 12/19**
  - Should have (pretty much) final results
  - Show us how great your project is
  - Some more ideas what to talk about:

    Which architecture(s) did you test on?

    How did you verify correctness of the parallelization?

    Use bounds models for comparisons!

    (Somewhat) realistic use-cases and input sets?

    Emphasize on the key concepts (may relate to theory of lecture)!

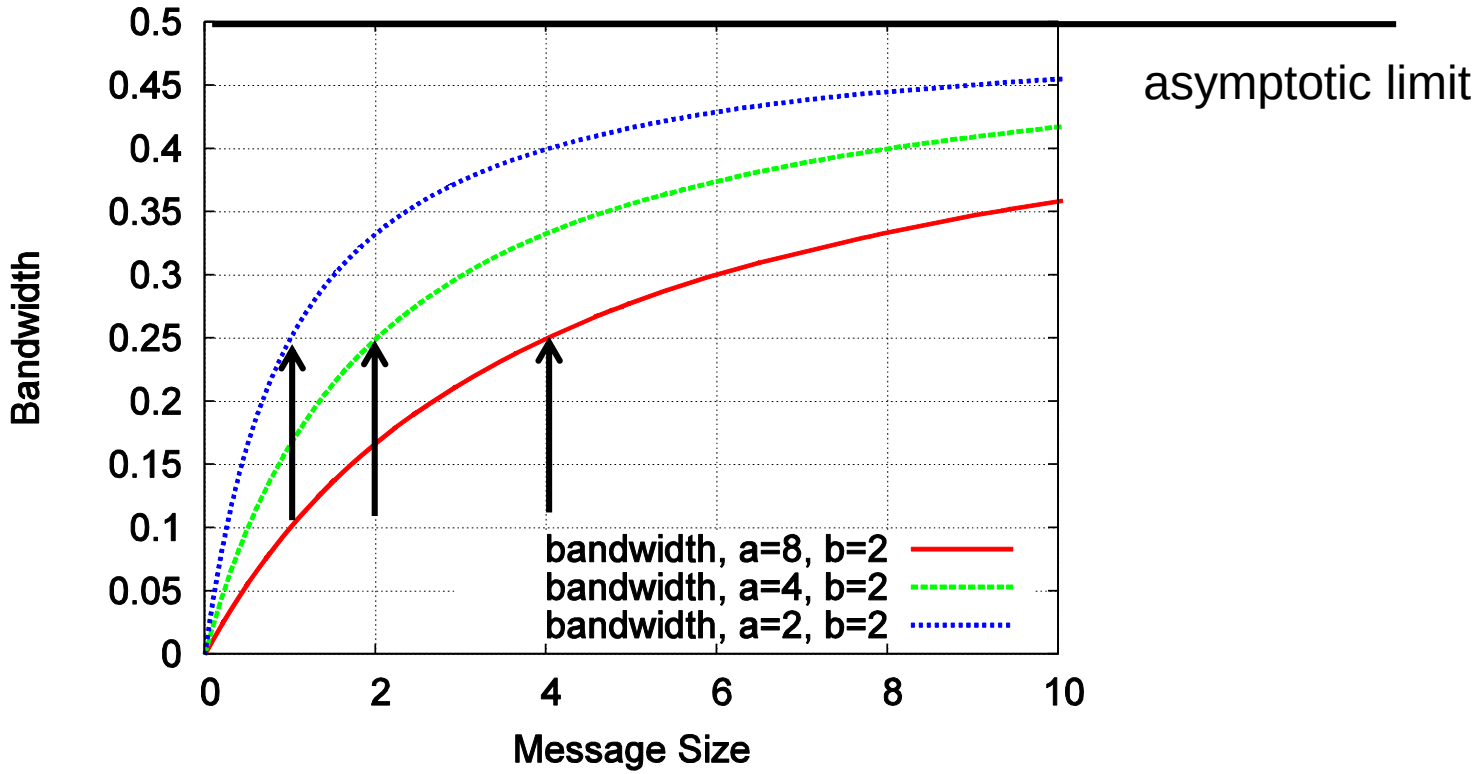
    What are remaining issues/limitations?
- **Deadline: send presentations to salvatore.di.girolamo@inf.ethz.ch by Sunday 12/18 11:59PM CET**
- **Report will be due in January!**
  - Still, starting to write early is very helpful --- write – rewrite – rewrite (no joke!)
  - Templates are on the webpage

A Simple Model for Communication

- **Transfer time T(s) = α+βs**
  - α = startup time (latency)
  - β = cost per byte (bandwidth=1/β)
- **As s increases, bandwidth approaches 1/β 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

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



Quick Example

- **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) * (α+βs) = $\mathcal{O}(P)$

- **Class question: Do you 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**
  - k=2 -> binary tree
- **Class Question: 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)
- **Class Question: 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, βs? Really?

Faster Trees?

- **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?

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?**
  - $T(s) = (P-2+s/z)(\alpha + \beta z)$
- **Compare 2-nomial tree with simple pipeline for $\alpha=10$, $\beta=1$, $P=4$, $s=10^6$, and $z=10^5$**
  - 2,000,020 vs. 1,200,120
- **Class Question: Can we do better for given α, β, P, s?**
  - Derive by z $$z_{opt} = \sqrt{\frac{s\alpha}{(P-2)\beta}}$$
- **What is the time for simple pipeline for $\alpha=10$, $\beta=1$, $P=4$, $s=10^6$, z_{opt}?**
  - 1,008,964

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
- **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?

- **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)

    Can be improved with linear programming and topology awareness

    (talk to me if you're interested)

Open Problems

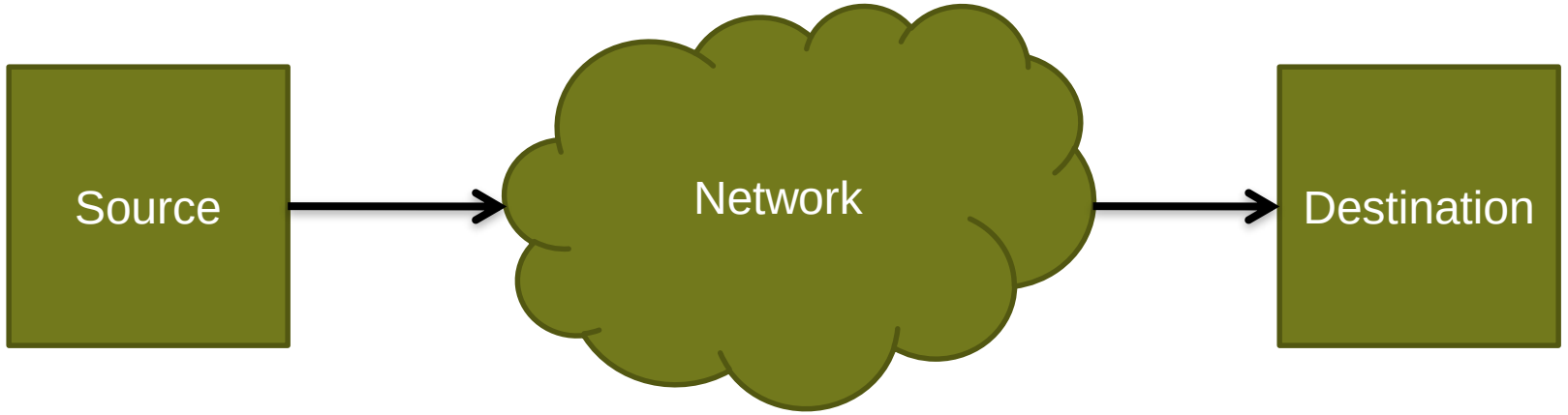
- **Look for optimal parallel algorithms (even in simple models!)**
  - And then check the more realistic models
  - Useful optimization targets are MPI collective operations

    Broadcast/Reduce, Scatter/Gather, Alltoall, Allreduce, Allgather, Scan/Exscan, …
  - Implementations of those (check current MPI libraries ☺)
  - Useful also in scientific computations

    Barnes Hut, linear algebra, FFT, …
- **Lots of work to do!**
  - Contact us for thesis ideas (or check SPCL) if you like this topic
  - Usually involve optimization (ILP/LP) and clever algorithms (algebra) combined with practical experiments on large-scale machines (10,000+ processors)

HPC Networking Basics

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

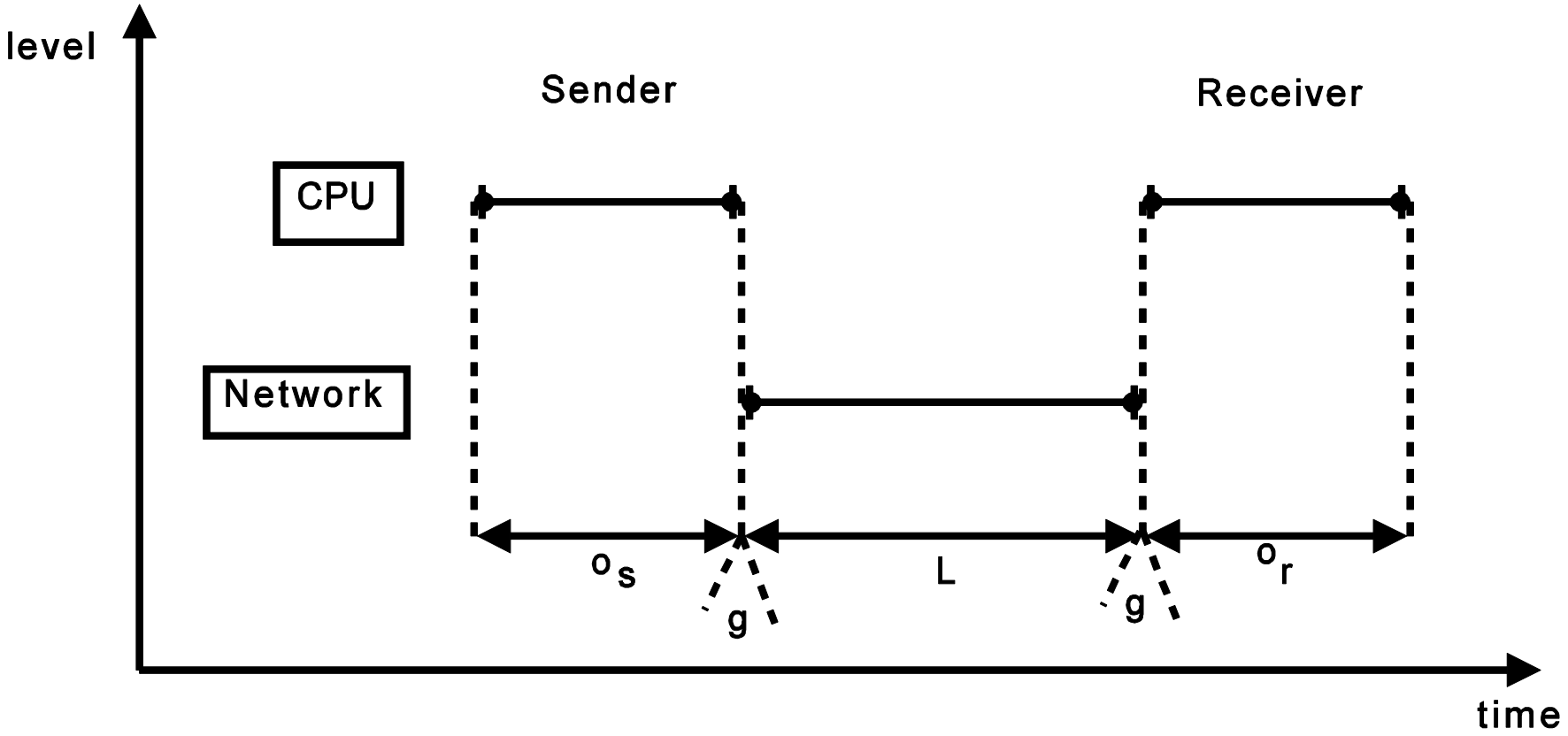


- **Class Question: What parameters are needed to model the performance (including pipelining)?**
  - Latency, Bandwidth, Injection Rate, Host Overhead

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

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

Example: Broadcasts

- **Class Question: Approximate the LogP runtime for a binomial tree broadcast of a single packet (assume L > g!)?**
  - $T_{bin} \leq \log_2 P * (L + 2o)$
- **Class Question: Approximate the LogP runtime for a k-nomial tree broadcast of a single packet?**
  - $T_{k\text{-}n} \leq \log_k P * (L + (k\text{-}2)\max(o,g) + 2o)$
- **Class Question: What is the optimal k (assume o>g)?**
  - Derive by k: $0 = o * \ln(k_{opt}) - L/k_{opt} + o$ (solve numerically)

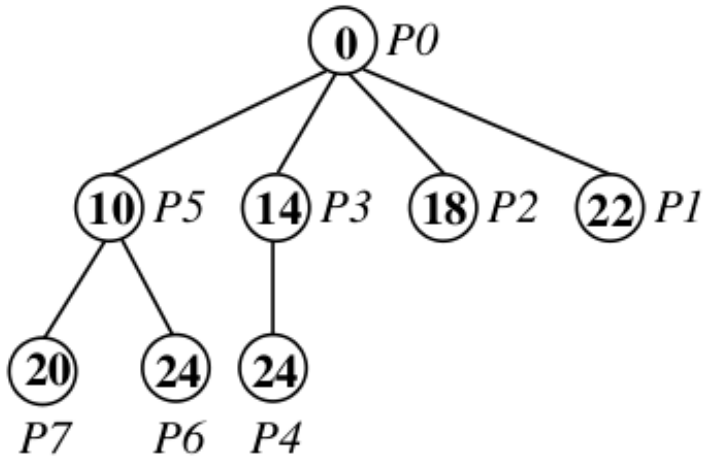
    For larger L, k grows and for larger o, k shrinks
  - Models pipelining capability better than simple model!

Example: Broadcasts

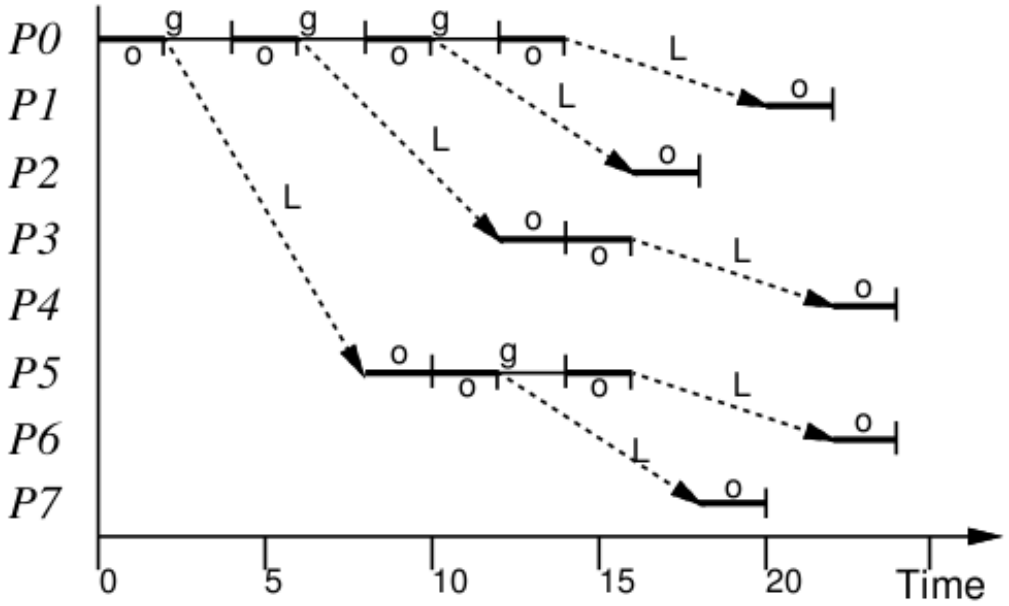
- **Class Question: Can we do better than k_{opt}-ary binomial broadcast?**
  - Problem: fixed k in all stages might not be optimal
  - We can construct a schedule for the optimal broadcast in practical settings
  - First proposed by Karp et al. in “Optimal Broadcast and Summation in the LogP Model”

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)

SPCL

https://spcl.inf.ethz.ch/SeMa/

Algorithm-Aware Graph Compression

After years of accelerating graph processing by improving the design of algorithms, it is becoming more and more clear that the key aspect is how the graph is represented and how it is stored. One notion is that the enormous size of today's graphs requires some form of compression. Recently, several approaches for both lossless and lossy graph compression have been proposed. However, decompression incurs performance overheads when accessing a graph and when running graph algorithms. The questions
various graph algorithms and how they

Parallel Succinct Data Structures

Collective Communication on GPUs

General Purpose Graphics Processing Units have a promising performance. The major programming models on these devices are bulk-synchronous and hide the latency. Thus, unscalable algorithms are used to communicate in bulk-synchronous steps. One open problem is to develop close-to-optimal collective communication operations for GPU devices. This would entail developing a detailed performance model (cf. the Xeon Phi model as well as novel ideas utilizing the GPU memory subsystem in creative ways. An example (barrier) can be found at in the paper Xiao, Feng: "Inter-Block GPU Communication via Fast Barrier Synchronization".

Requirements:

- good C/C++ programming skills
- understanding of computer architecture/optimization
- understanding of statistics and basic performance modeling

Relevance: This project would make you an expert in high-performance programming for accelerator devices.

Composition: 50% accelerator implementation, 50% performance modeling

Contact: Torsten Hoefler -- htor at inf.ethz.ch

Adaptive Routing for InfiniBand

re statically routed today, which means that all packets between the same
rough the same path. Static routing is inefficient on many topologies and
fficiently. Newer versions of InfiniBand hardware offer adaptive routing
set up to select different paths for each packet between a pair of hosts. This
k published at IPDPS, Domke et al. "Deadlock-Free Oblivious Routing for
and greatly extends it to adaptive routing. The project offers an opportunity
laboration and

ons

o use the cache as directly addressable main
inuxBIOS" and Intel's documentation). The goal of this
ddressible cache in order to improve performance of
cratchpad memory. Scratchpad memories have
GPUs.

optimization

Relevance: This project would make you an expert in low-level systems programming and high-performance computing

Composition: 70% low-level systems programming, 30% performance programming and modeling

Contact: Torsten Hoefler -- htor at inf.ethz.ch