PopSub: Improving Resource Utilization in Distributed Content-based Pub/Sub Systems

DEBS 2017

Pooya Salehi
Joint work with Kaiwen Zhang and Hans-Arno Jacobsen

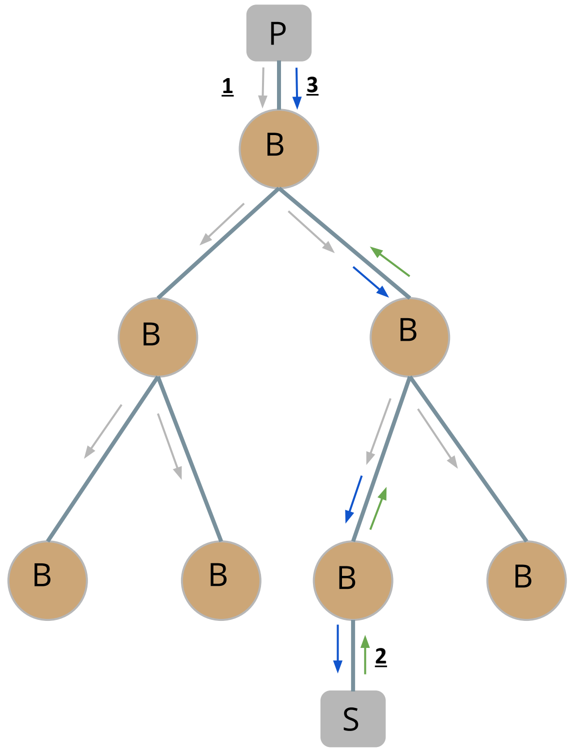
Middleware Systems Research Group
Technical University of Munich, Germany

Publish/Subscribe Communication

- Facilitates many-to-many communication
  - Loosely-coupled
  - Asynchronous
- Enables large-scale distributed applications
- Consists of Publishers, Subscriber and Brokers

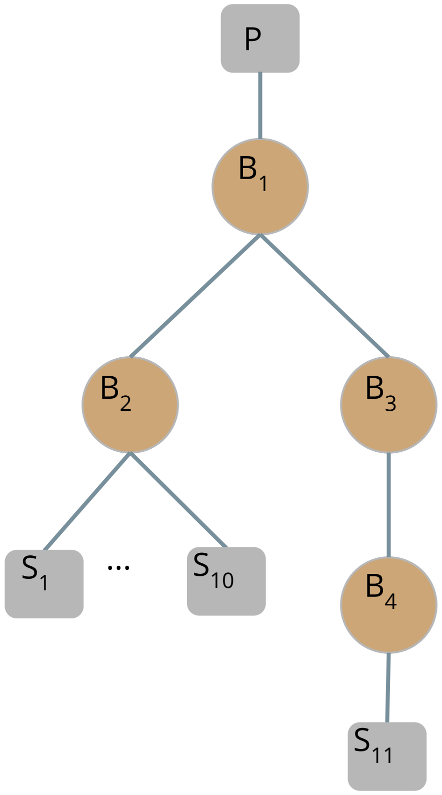
Distributed Pub/Sub

- Uses an overlay of brokers for scalability
- Routing information is distributed
- Uses reverse-path forwarding
- Finding optimal topology is NP-hard!



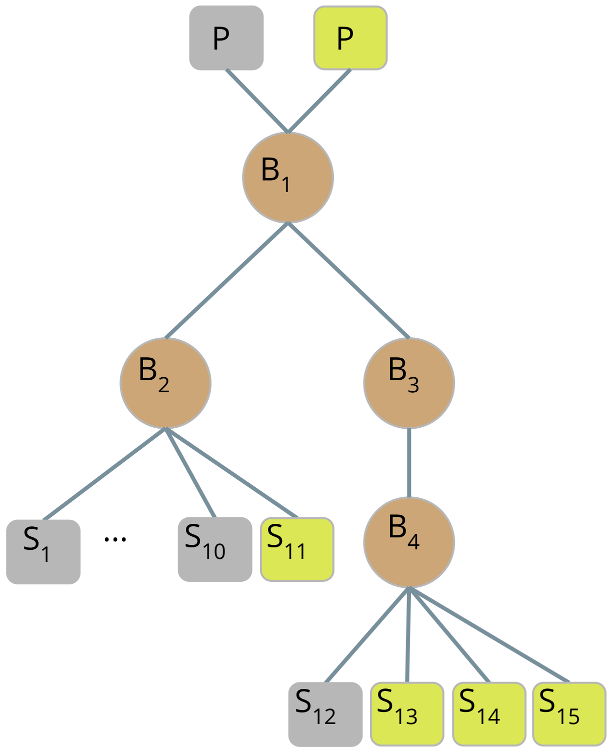
Binary Cost Model Problem

- Broker always forwards a matching publication
  - One matching subscription is enough
  - A requirement for routing correctness
- Model does not consider popularity
- Publication is either forwarded or not
- Delivering some publications can be costly
  - Low popularity
  - Long routing path



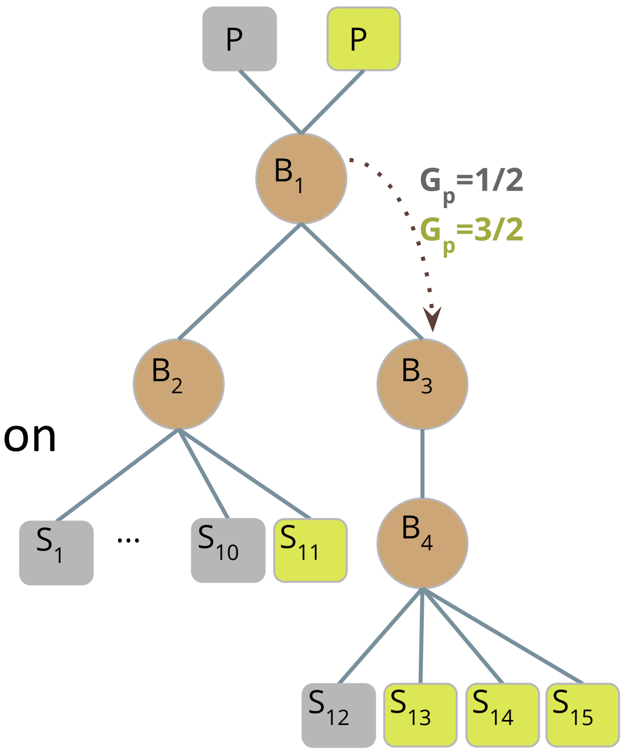
Related Work

- Self-organizing overlays
- Overlay reconfiguration
    - Benefits popular subscriptions
    - Can be very costly
- Efficient publication routing
    - Opportunistic multipath forwarding (Kazemzadeh *et al.*, 2012)
    - Atmosphere (Jayalath, *et al.*, 2013)



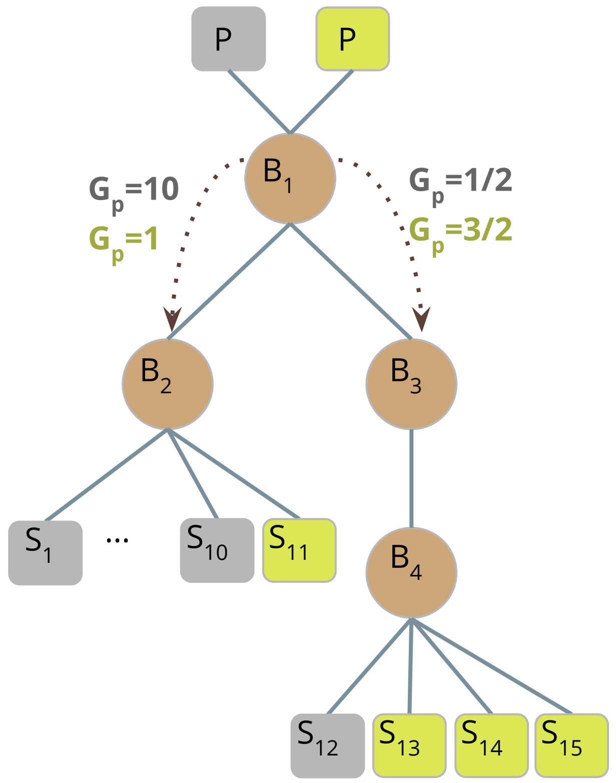
PopSub: Popularity-Based Cost Model

- Prioritize publications based on their gain
- Gain depends on distance and volume
  - $G_p = \frac{|\,sub(\mathbb{s})\,|}{avg(T_{\mathcal{P}\leftrightarrow\mathbb{S}})}$
  - Brokers keep local estimates
- Gain estimated during subscription propagation
- Allocate resources to publications with The highest gain ratio first



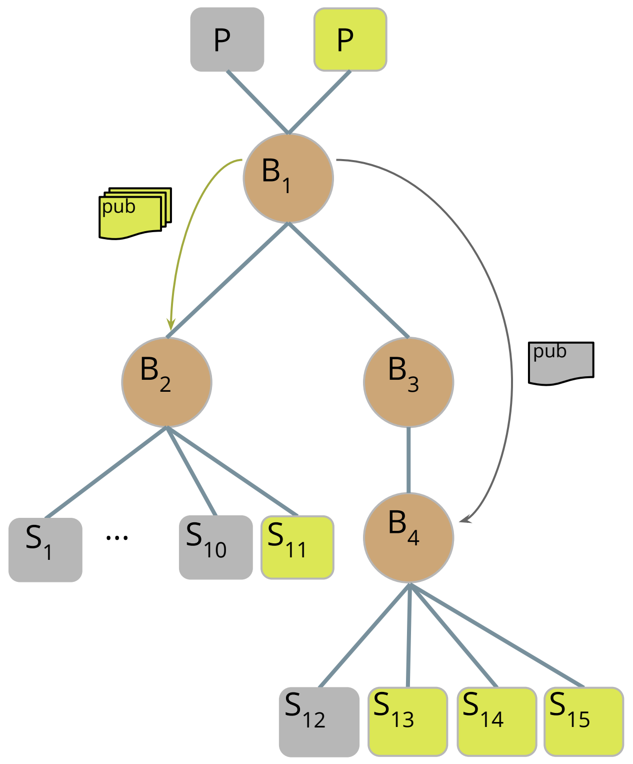
Popularity Evaluation

- Estimate gain per link and advertisement
- Each broker keeps |L|×|𝔸| estimates
- Periodically prioritize based on gain estimate
- Fill up capacity with popular publications
- Popular publications are routed through the overlay



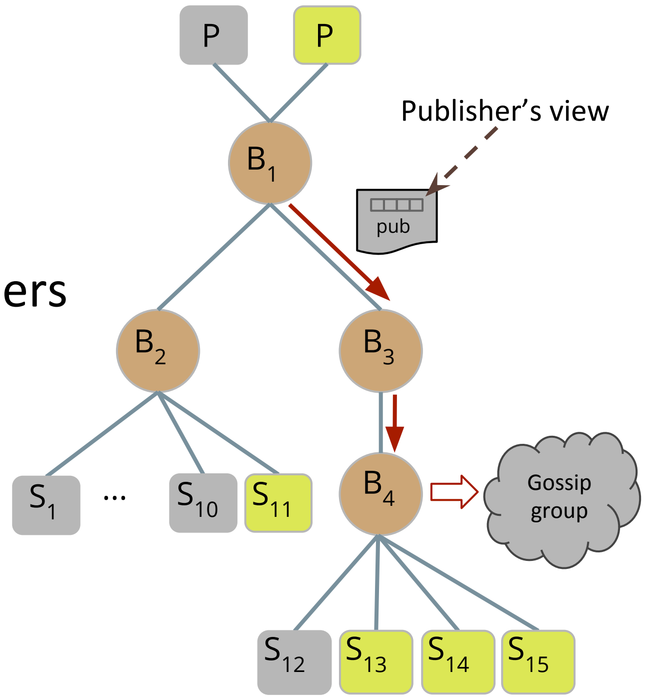
Handling Unpopular Publications

- Direct delivery to edge brokers
  - Not always scalable
  - Useful in non-uniform workloads
- Batching on publishers' edge brokers
  - Timeout
  - Minimum gain ratio reached
- Gossiping between brokers (PopSub)



Gossiping Unpopular Publications

- Use Lightweight Probabilistic Broadcast (Eugster *et al.*, 2003)
- One broadcast group per advertisement
- Handoff needs to be coordinated
- First message carries potential gossip partners
- On-demand gossip group creation

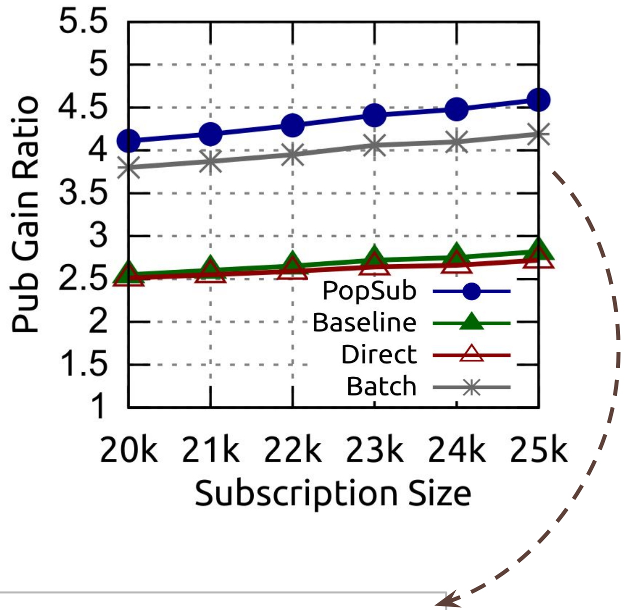


Evaluation Setup

- Discrete-event simulation in Java (Jist framework)
- Internet-based hierarchical topologies (Lumezanu *et al.*, 2007)
- Twitter-based publication popularity (Leskovec *et al.*, 2012)
- 200 publishers, 10 pub/sec, 20 classes, 20k-25k subscribers
- Popularity evaluation every 2 seconds
- Approaches:
  - ❑ Baseline
  - ❑ Direct delivery
  - ❑ Batching
  - ❑ PopSub (Gossiping)
- Metrics: Publication gain ratio, pure forwards

Evaluation - Subscription Size

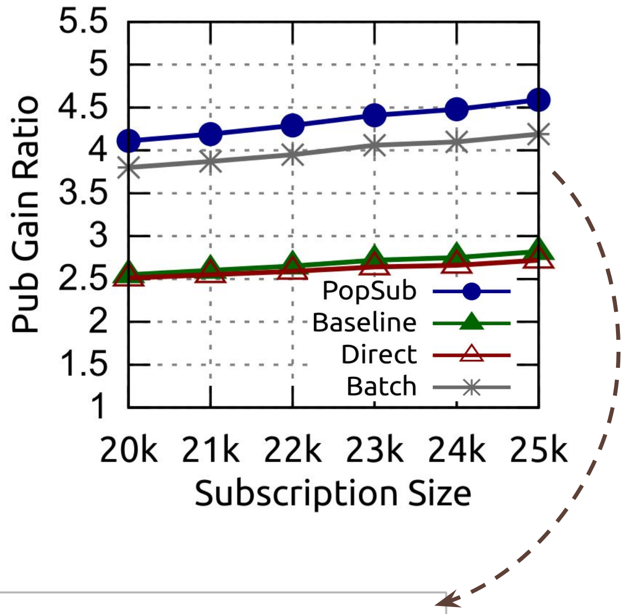
- Overlay of 200 brokers

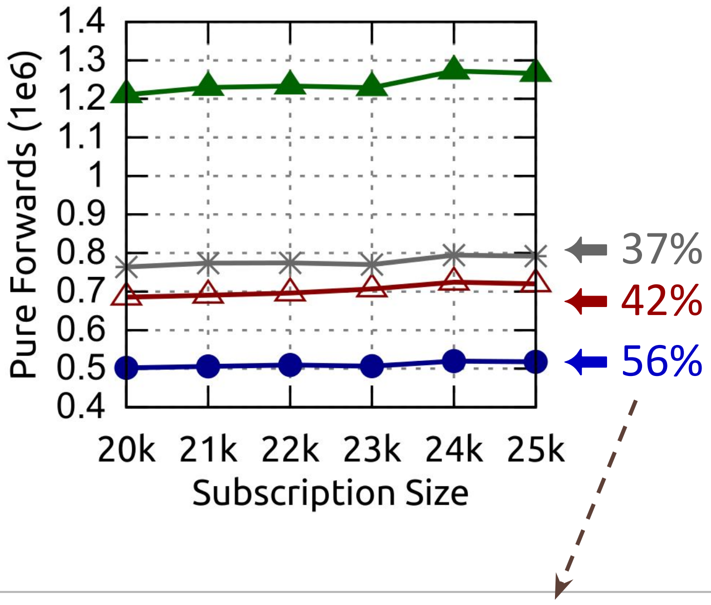


Up to 62% improvement!

Evaluation - Subscription Size

- Overlay of 200 brokers



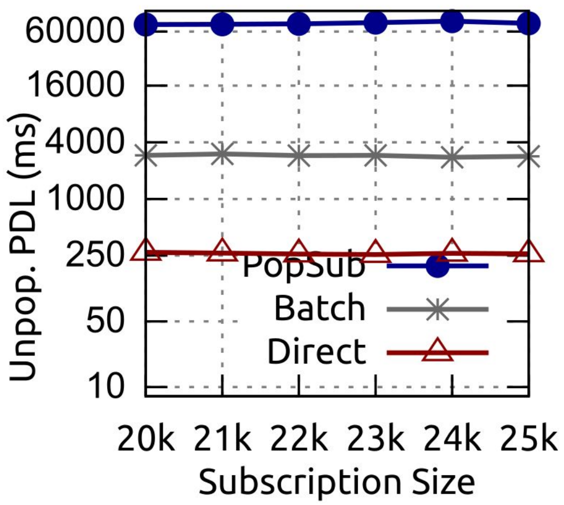


Up to 62% improvement!

Reduction in number of pure forwards

Evaluation - Subscription Size (Continued)

- Overlay of 200 brokers



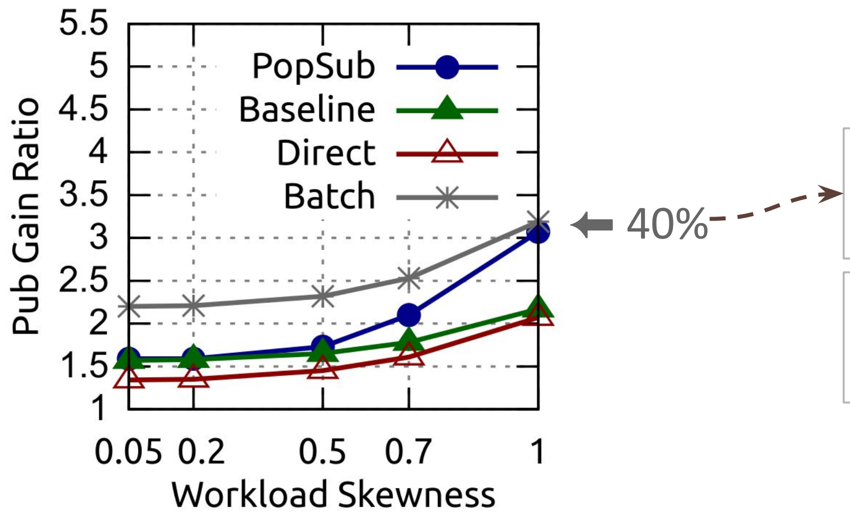
Direct ➡ Batching ➡ Gossip

Latency

90th %ile of gossip ≅ 20 seconds

Evaluation - Workload Skewness

- Synthesized workload with Zipfian distribution

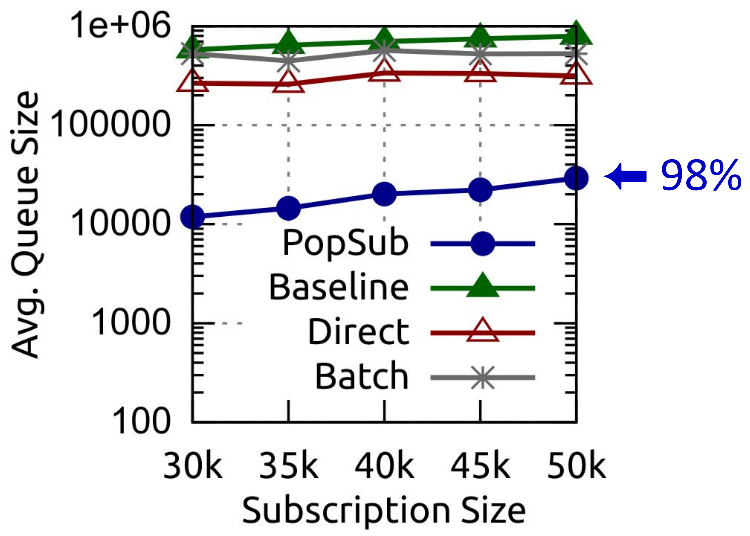


Batching improves gain regardless of skewness.

Gossiping performs better in skewed workloads!

Evaluation - PopSub Under Load

- Each publisher 15 pub/sec, 30k - 50k subscribers
- Similarly improve publication gain ratio



- Prioritizing based on popularity benefits popular publications

Conclusions

- Increase resource utilization of a pub/sub system
- Prioritize publications based on their popularity
- Use “cheaper” approaches to handle unpopular publications
- Maintain same delivery latency for popular publications
- Improve resource utilization by up to 62%
- Reduce unnecessary publication forwarding by up to 59%

Conclusions

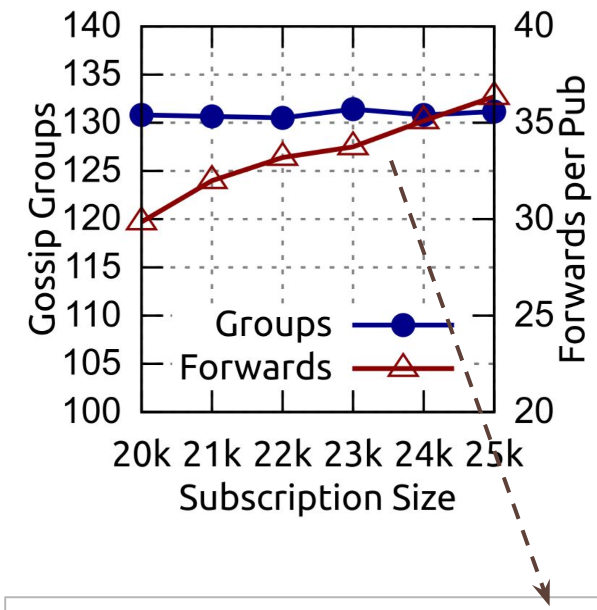
- Increase resource utilization of a pub/sub system
- Prioritize publications based on their popularity
- Use “cheaper” approaches to handle unpopular publications
- Maintain same delivery latency for popular publications
- Improve resource utilization by up to 62%
- Reduce unnecessary publication forwarding by up to 59%

THANK YOU! :-)

Backup Slides

Evaluation - Subscription Size (Continued)

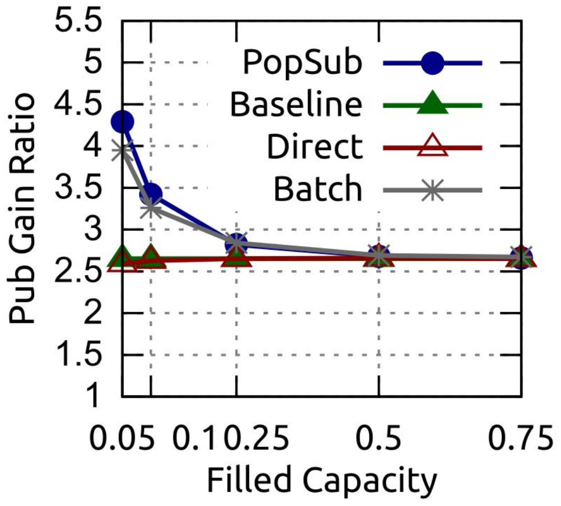
- Overlay of 200 brokers

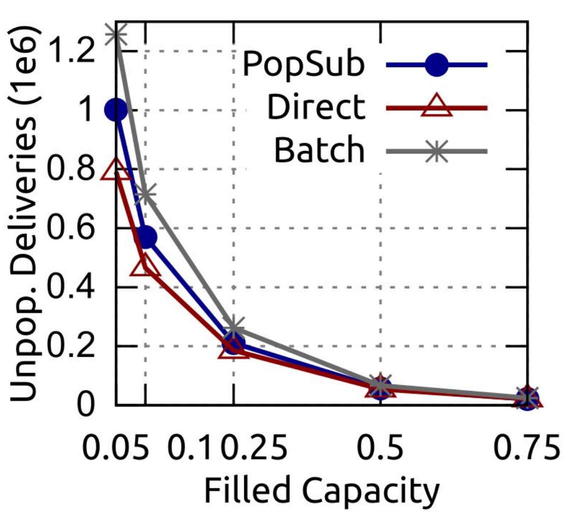


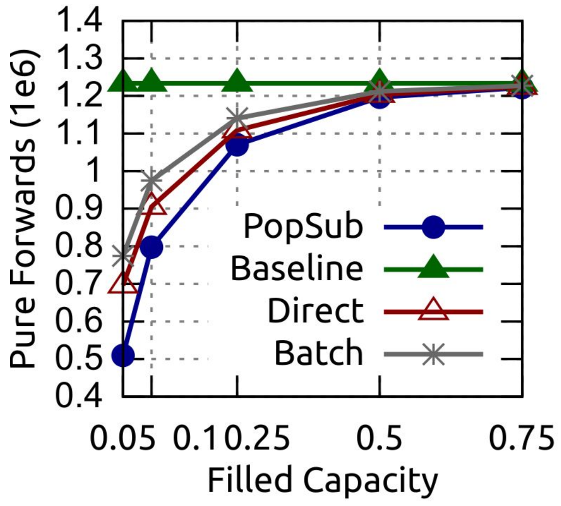
Direct Delivery is not scalable in uniform workloads!

Filled Capacity

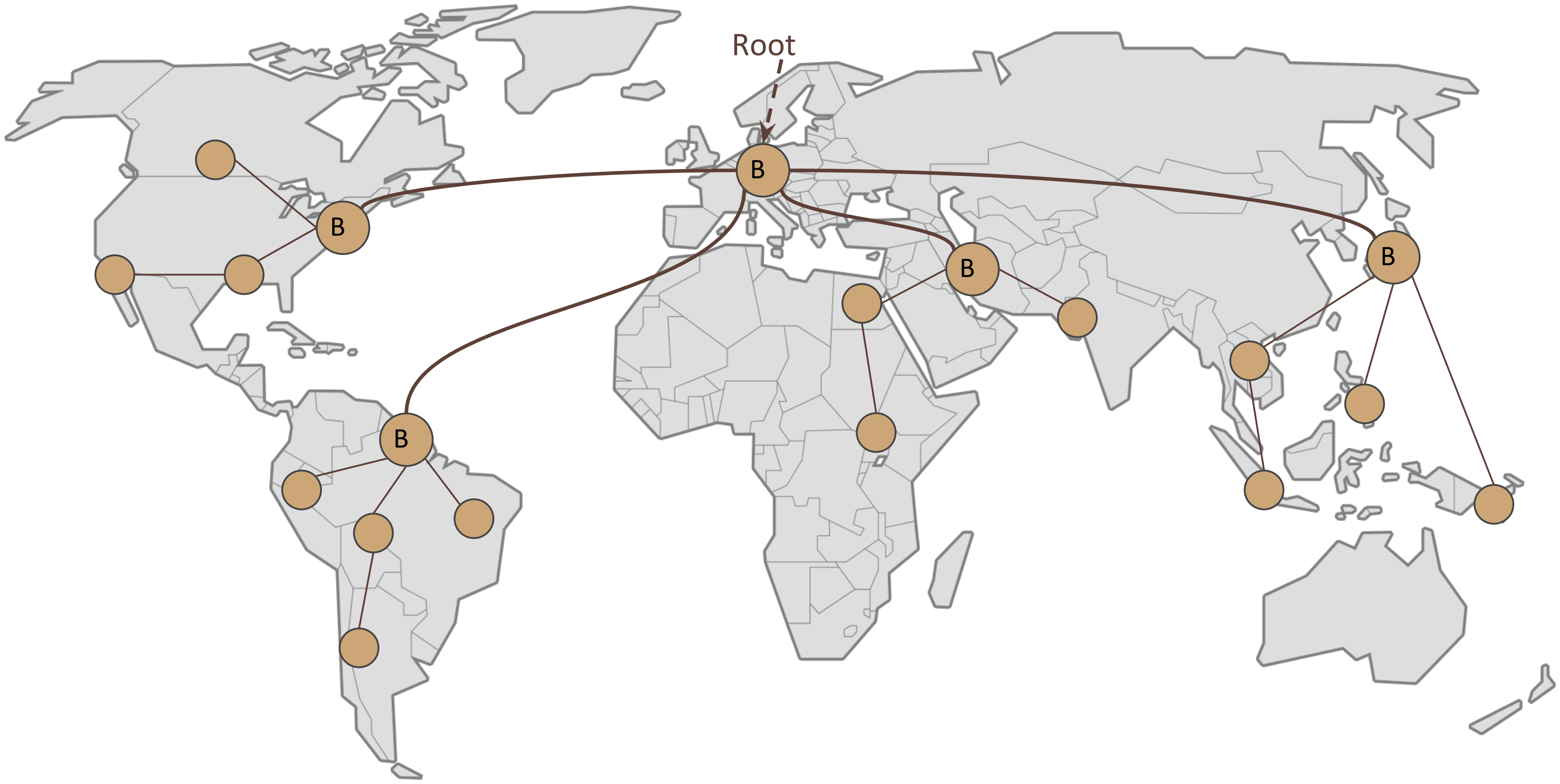
- Fill up a fraction of capacity (φ < 0.8)







Topology


Root
B
B
B
B
B
B