Optimizing Networked Systems with Limited Information

Bruce Maggs
Duke University

Debmalya Panigrahi
Duke University

Rajmohan Rajaraman
Northeastern University

Ravi Sundaram
Northeastern University

Graduate Students

Samuel Haney
Duke University

Mehraneh Liaee
Northeastern University

Symmetric Matching Interdiction

Samuel Haney
Duke University

Bruce Maggs
Duke University

Biswaroop Maiti
Northeastern University

Debmalya Panigrahi
Duke University

Rajmohan Rajaraman
Northeastern University

Ravi Sundaram
Northeastern University

[in submission]

The Dark Menace: Characterizing Network-based Attacks in the Cloud

[IMC 2016]

Rui Miao * Rahul Potharaju ‡ Minlan Yu* Navendu Jain†
* University of Southern California ‡ Microsoft † Microsoft Research

Microsoft Azure

Google Cloud Platform

amazon web services™

The Dark Menace: Characterizing Network-based Attacks in the Cloud

[IMC 2016]

Rui Miao * Rahul Potharaju ‡ Minlan Yu* Navendu Jain†
* University of Southern California ‡ Microsoft † Microsoft Research

Microsoft Azure

Google Cloud Platform

amazon web services™

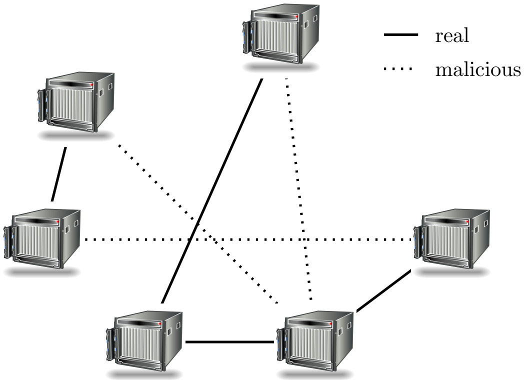
A recent study of malicious network traffic observed at Microsoft data centers made the surprising observation that a large volume of attack traffic originated from virtual machines hosted within the data centers themselves.

It is usually difficult to distinguish between legitimate and malicious traffic.

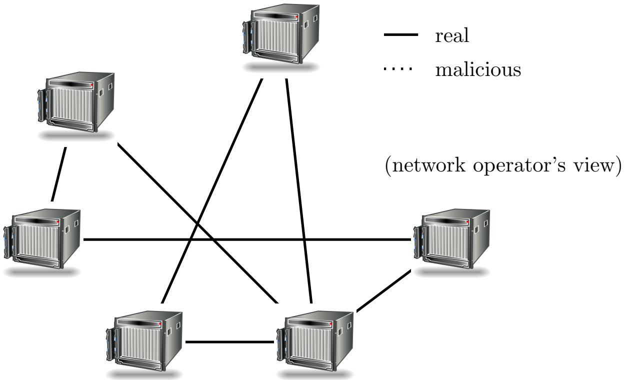
It is usually difficult to distinguish between legitimate and malicious traffic.
What can we do?

It is usually difficult to distinguish between legitimate and malicious traffic.

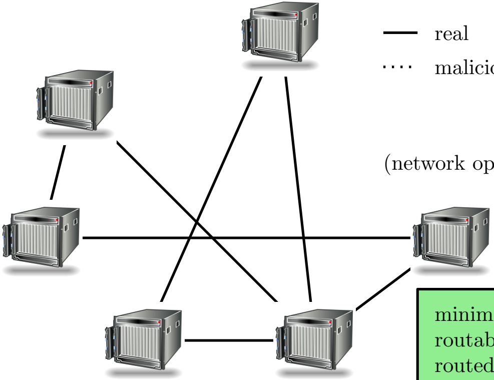
What can we do?



It is usually difficult to distinguish between legitimate and malicious traffic.
What can we do?



It is usually difficult to distinguish between legitimate and malicious traffic.
What can we do?

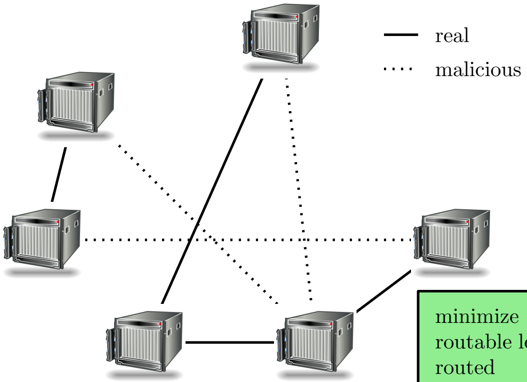


— real
···· malicious

(network operator's view)

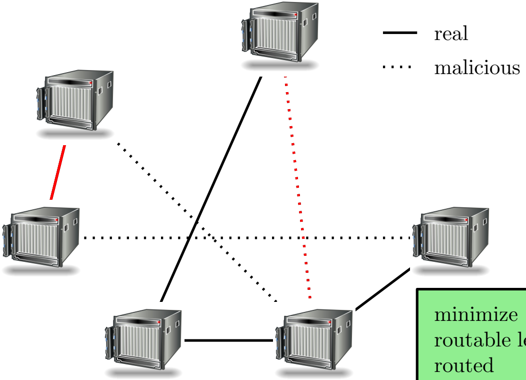
minimize **regret**: amount of routable legitimate traffic not routed

It is usually difficult to distinguish between legitimate and malicious traffic.
What can we do?



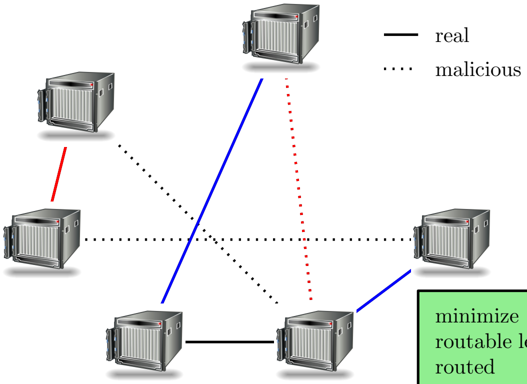
minimize **regret**: amount of routable legitimate traffic not routed

It is usually difficult to distinguish between legitimate and malicious traffic.
What can we do?



minimize **regret**: amount of routable legitimate traffic not routed

It is usually difficult to distinguish between legitimate and malicious traffic.
What can we do?



minimize **regret**: amount of routable legitimate traffic not routed

Symmetric Matching Interdiction

When each server has unit capacity, this formalization can be simply stated as follows.

Definition (Symmetric Matching Interdiction (SMI))

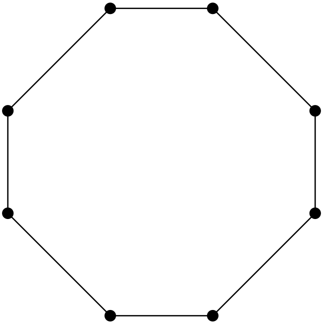
Given a graph G, find matching M such that the maximum matching in $G \setminus M$ in minimized.

Symmetric Matching Interdiction

When each server has unit capacity, this formalization can be simply stated as follows.

Definition (Symmetric Matching Interdiction (SMI))

Given a graph G, find matching M such that the maximum matching in $G \setminus M$ in minimized.

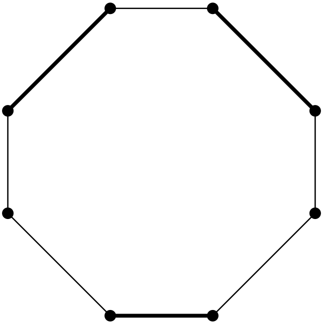


Symmetric Matching Interdiction

When each server has unit capacity, this formalization can be simply stated as follows.

Definition (Symmetric Matching Interdiction (SMI))

Given a graph G, find matching M such that the maximum matching in $G \setminus M$ in minimized.

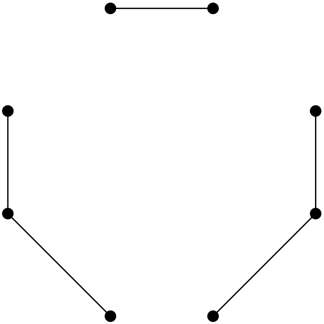


Symmetric Matching Interdiction

When each server has unit capacity, this formalization can be simply stated as follows.

Definition (Symmetric Matching Interdiction (SMI))

Given a graph G, find matching M such that the maximum matching in $G \setminus M$ in minimized.

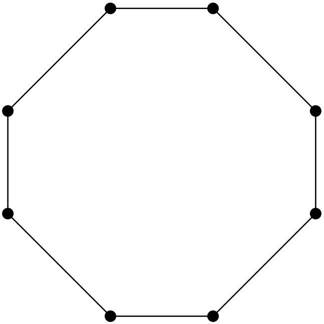


Symmetric Matching Interdiction

When each server has unit capacity, this formalization can be simply stated as follows.

Definition (Symmetric Matching Interdiction (SMI))

Given a graph G, find matching M such that the maximum matching in $G \setminus M$ in minimized.

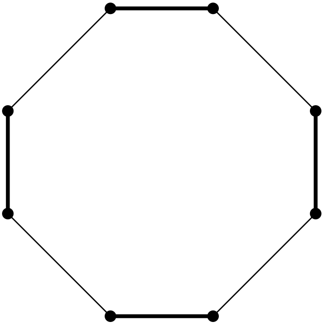


Symmetric Matching Interdiction

When each server has unit capacity, this formalization can be simply stated as follows.

Definition (Symmetric Matching Interdiction (SMI))

Given a graph G, find matching M such that the maximum matching in $G \setminus M$ in minimized.

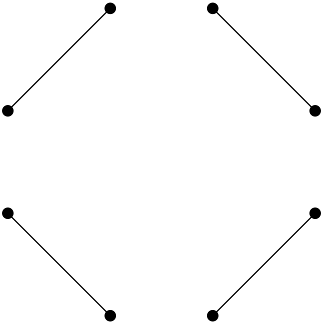


Symmetric Matching Interdiction

When each server has unit capacity, this formalization can be simply stated as follows.

Definition (Symmetric Matching Interdiction (SMI))

Given a graph G, find matching M such that the maximum matching in $G \setminus M$ in minimized.



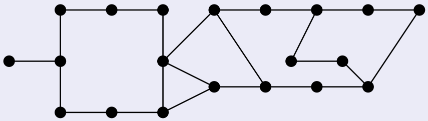
Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Proof Sketch.

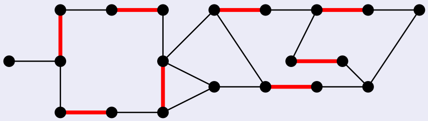


□

Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Proof Sketch.

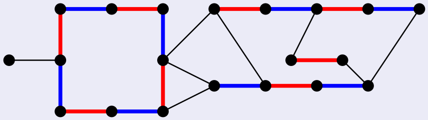


□

Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Proof Sketch.

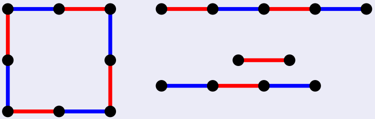


□

Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Proof Sketch.

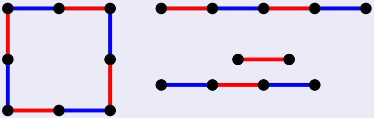


□

Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Proof Sketch.



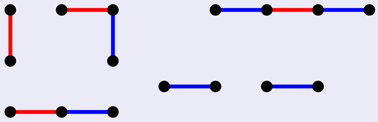
- The optimal solution removes some of these edges (the edges removed must satisfy matching constraints).

□

Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Proof Sketch.



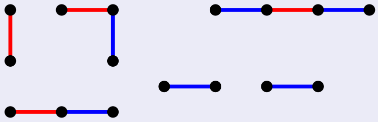
- The optimal solution removes some of these edges (the edges removed must satisfy matching constraints).

□

Claim

Any maximal matching is a 2-approximation to the optimal interdiction matching.

Proof Sketch.



- The optimal solution removes some of these edges (the edges removed must satisfy matching constraints).
- On the remaining graph, there is always a matching that is at least half the size of the (original) number of blue edges.

□

Symmetric Matching Interdiction Results

- We give a non-trivial algorithm that finds a 3/2-approximation (improving on the 2-approximation from the previous slide).

- Symmetric matching interdiction is APX-hard, i.e. cannot be approximated better than a constant.

- Why do we call it symmetric interdiction?
  - standard interdiction: remove k edges to minimize some objective
- Symmetric interdiction models denial of service attacks
  - adversary and user have the same constraints
  - other problems fit in the symmetric interdiction framework: flows, *b*-matching, demand matching
- We show that in general, an α-approximation to an optimization problem is a $(1 + \alpha)$-approximation to the corresponding interdiction problem.

Capacity Planning

Samuel Haney
Duke University

Vinay Kanitkar
Akamai

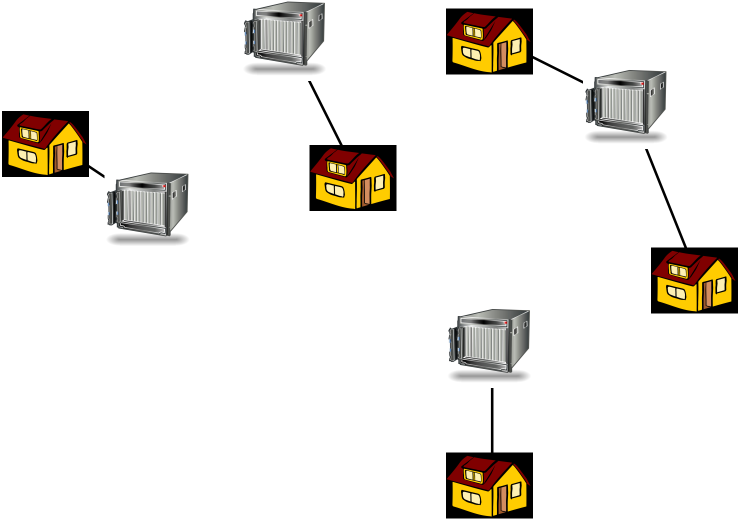
Bruce Maggs
Duke University

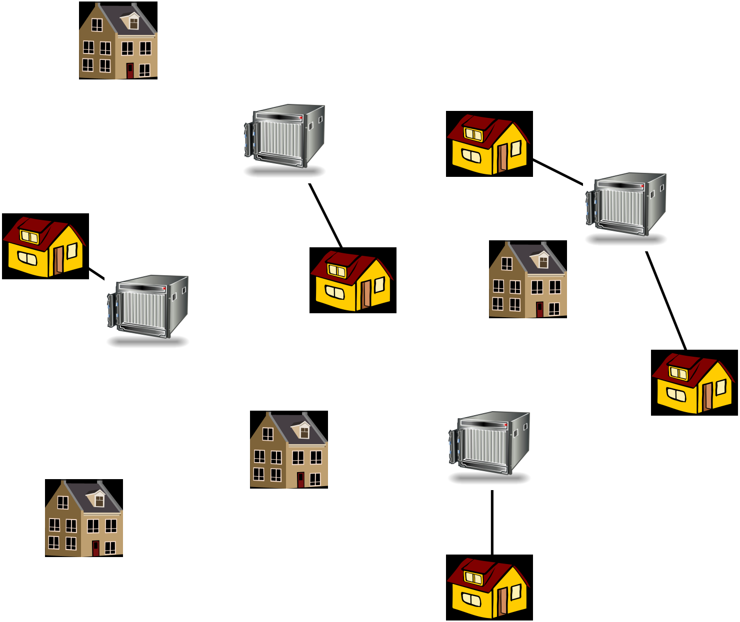
Debmalya Panigrahi
Duke University

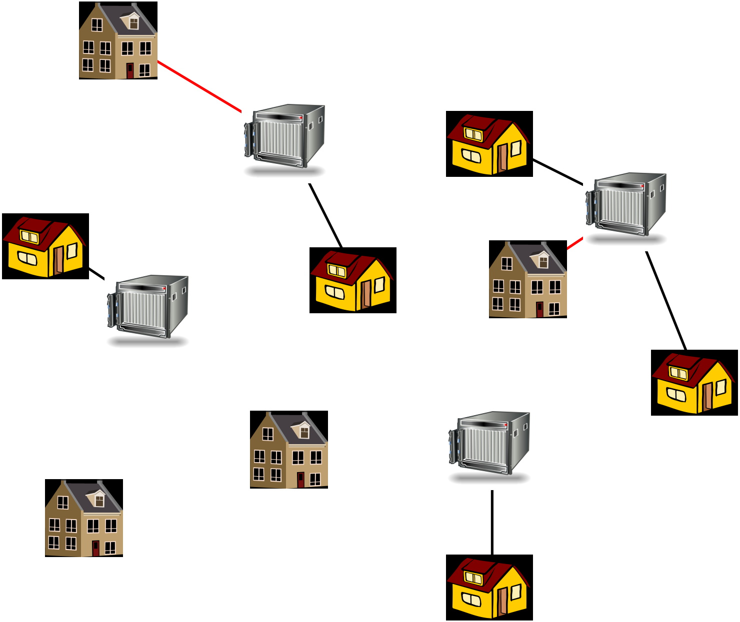
Ramesh K. Sitaraman
Akamai, UMass Amherst

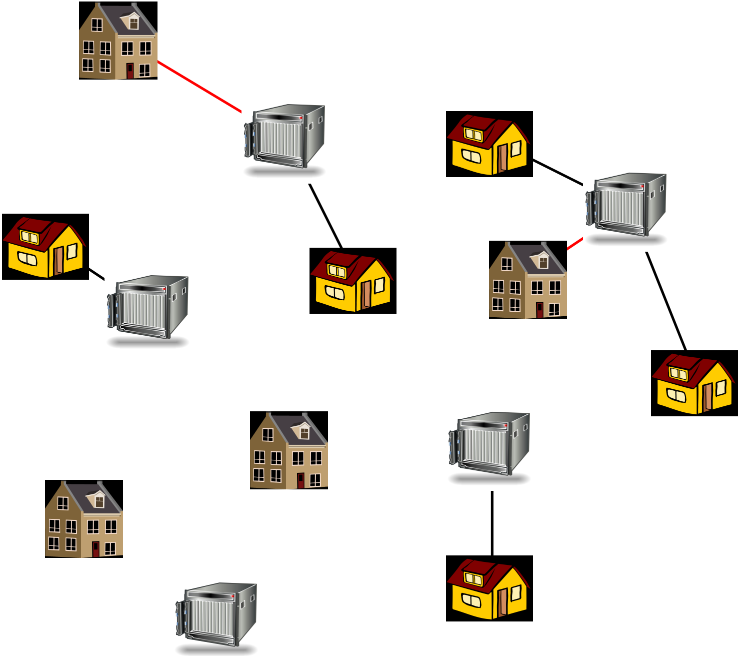
[ongoing]

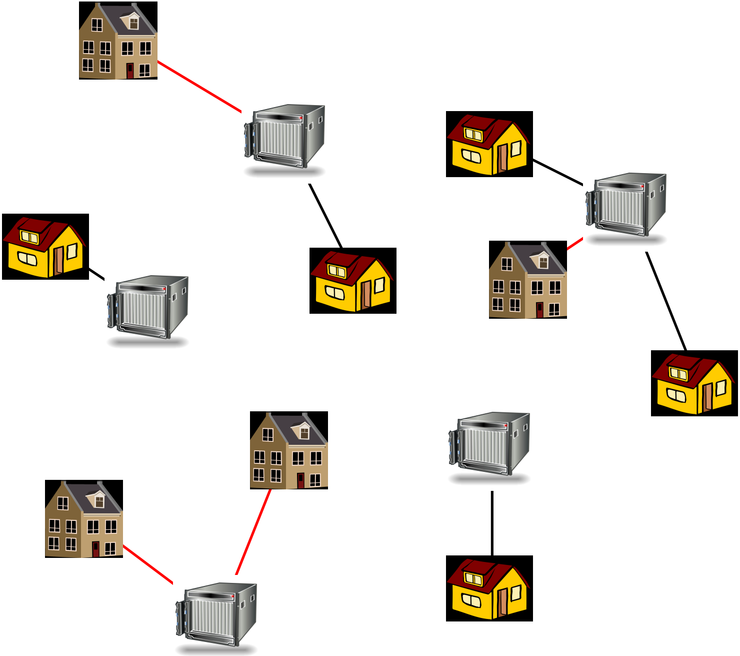
- The internet is growing fast
- CDNs need to scale up capacities rapidly
- This project explores how to plan expanded capacity using noisy predictions of future need.

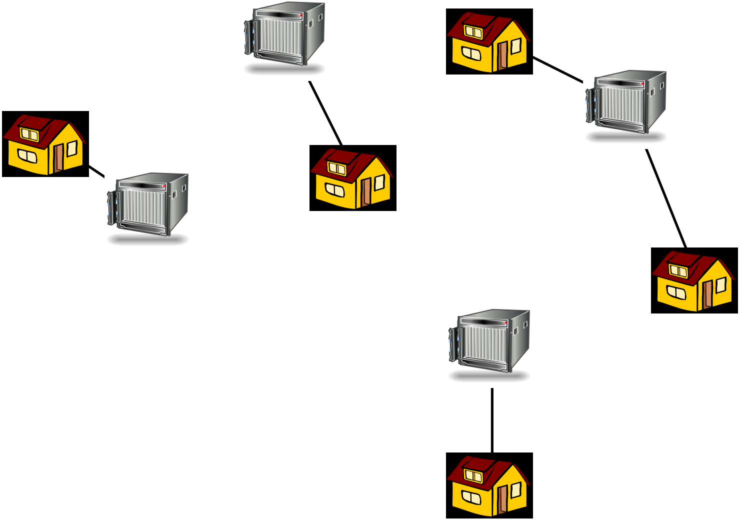


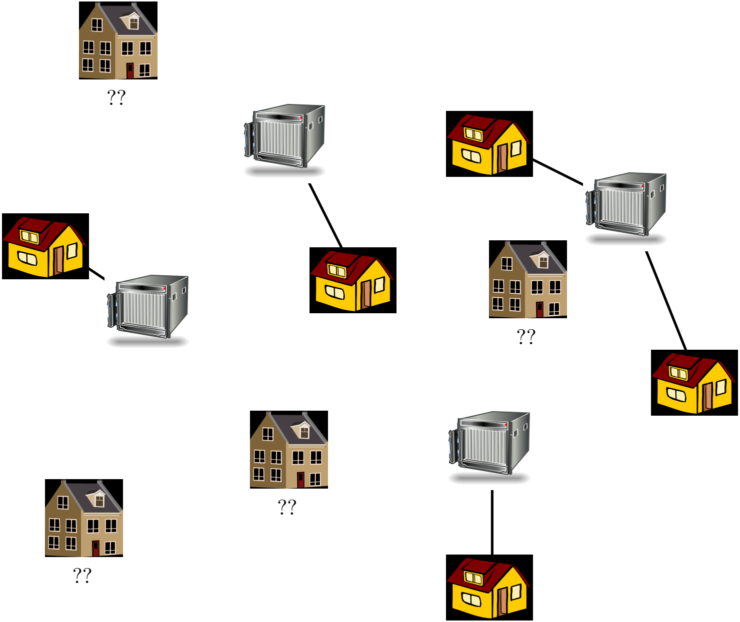


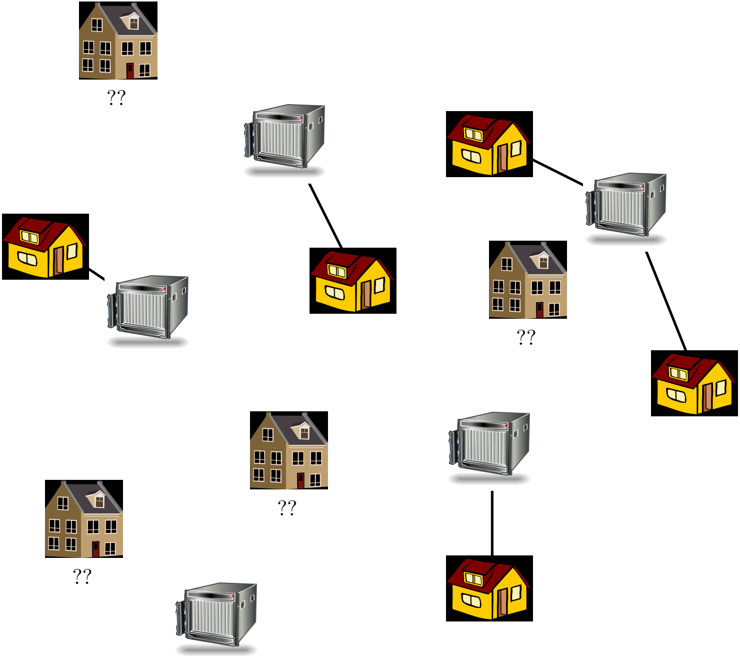


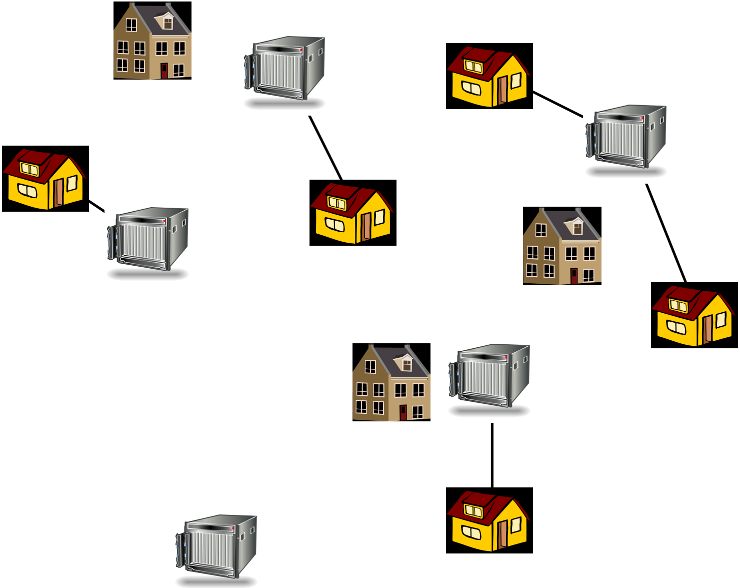


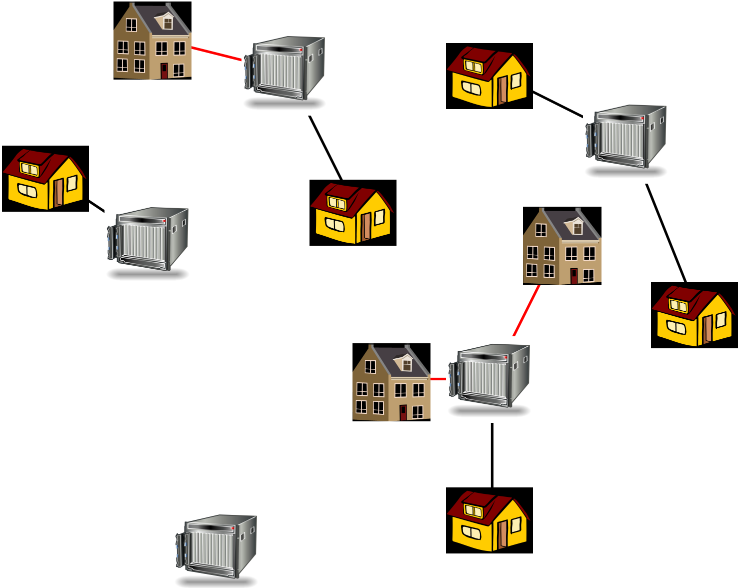












"Overall, the U.S. economy seems likely to expand at a moderate pace over the second half of 2007, with growth then strengthening a bit in 2008 to a rate close to the economy's underlying trend."

—Bernanke, 2007

> *"Overall, the U.S. economy seems likely to expand at a moderate pace over the second half of 2007, with growth then strengthening a bit in 2008 to a rate close to the economy's underlying trend."*
>
> *—Bernanke, 2007*

Can we develop algorithms whose performance degrades gracefully with decreasing accuracy of predictions?

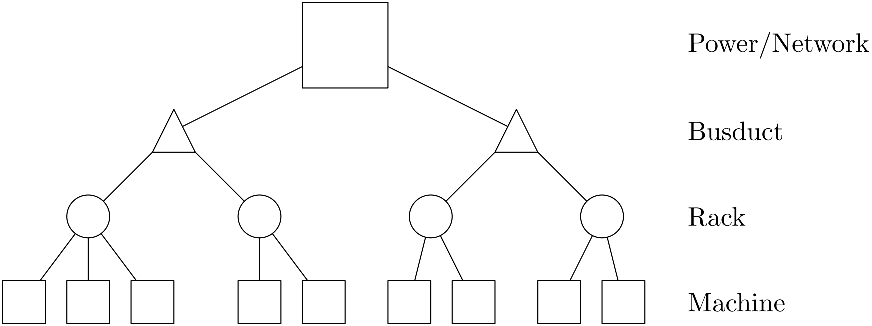
Robust and Probabilistic Failure-Aware Placement

Madhukar Korupolu
Google Research

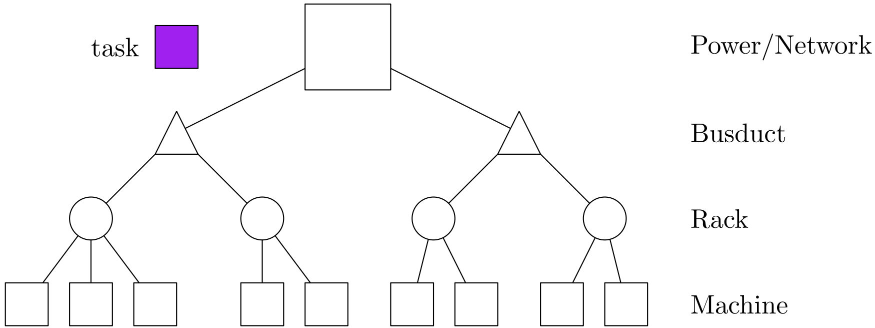
Rajmohan Rajaraman
Northeastern University

[ACM SPAA 2016]

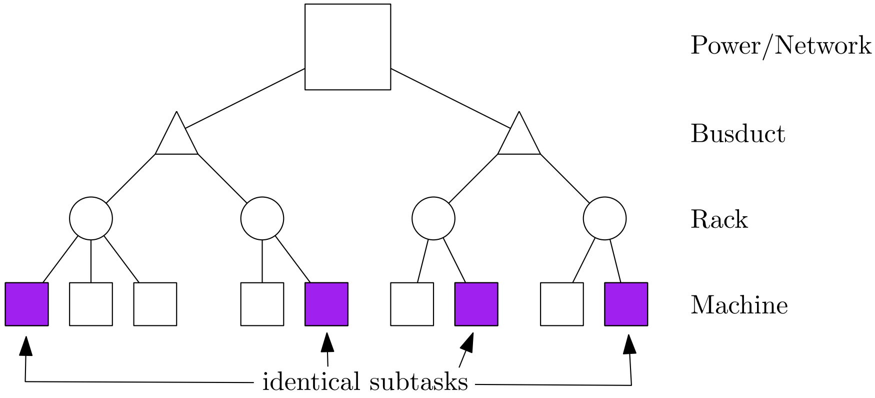
How do we place tasks to improve availability in presence of failures?



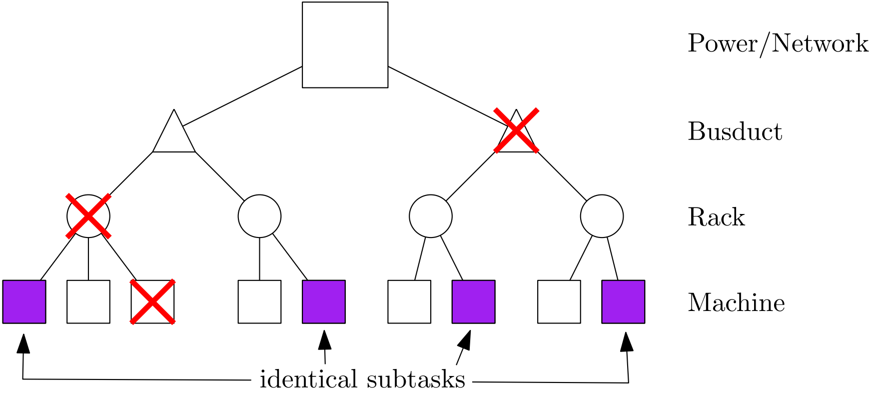
How do we place tasks to improve availability in presence of failures?



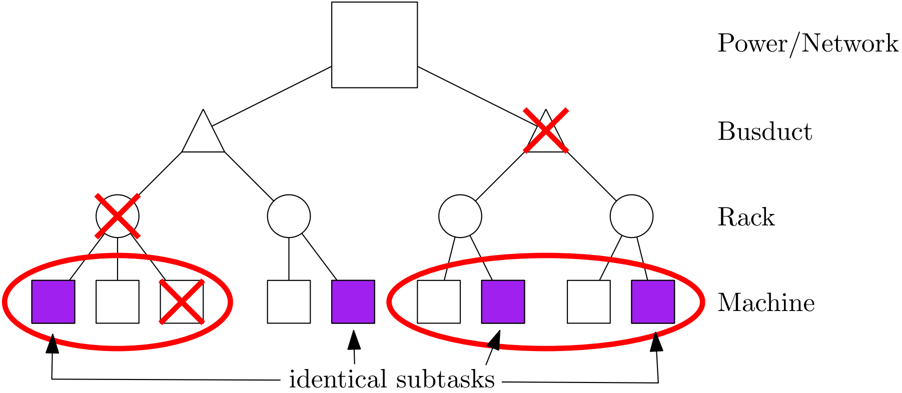
How do we place tasks to improve availability in presence of failures?



How do we place tasks to improve availability in presence of failures?



How do we place tasks to improve availability in presence of failures?



Results

RobustFAP (nodes have reliability weight):

- Problem is co-NP hard.
- PTAS/approximation algorithms

Results

RobustFAP (nodes have reliability weight):

- Problem is co-NP hard.
- PTAS/approximation algorithms

ProbFAP (nodes have probability of failure):

- PTAS based on Poisson approximation techniques

High Availability in Clusters

Bochao Shen
Northeastern University

Ravi Sundaram
Northeastern University

Srinivas Aiyar
Nutanix

Karan Gupta
Nutanix

Abhinay Nagpal
Nutanix

Aditya Ramesh
Nutanix

[in submission]

- Given cluster of nodes and VMs does there exist a packing such that for all failures of k nodes there is a disruption-free repacking?

- Given packing of VMs into nodes of a cluster, is there a disruption-free repacking for all failures of k nodes?

- Industry standard is Martello-Toth, a heuristic for Multiple Knapsack. How effective is it?

- Given cluster of nodes and VMs does there exist a packing such that for all failures of k nodes there is a disruption-free repacking? **Sounds like a Σ_3 complete problem. Actually in NP!**

- Given packing of VMs into nodes of a cluster, is there a disruption-free repacking for all failures of k nodes?

- Industry standard is Martello-Toth, a heuristic for Multiple Knapsack. How effective is it?

- Given cluster of nodes and VMs does there exist a packing such that for all failures of k nodes there is a disruption-free repacking? **Sounds like a Σ_3 complete problem. Actually in NP!**
- Given packing of VMs into nodes of a cluster, is there a disruption-free repacking for all failures of k nodes? **Sounds like a Π_2 complete problem. It probably is. We show NP-hard and coNP-hard.**
- Industry standard is Martello-Toth, a heuristic for Multiple Knapsack. How effective is it?

- Given cluster of nodes and VMs does there exist a packing such that for all failures of k nodes there is a disruption-free repacking?
  Sounds like a Σ_3 complete problem. Actually in NP!

- Given packing of VMs into nodes of a cluster, is there a disruption-free repacking for all failures of k nodes?
  Sounds like a Π_2 complete problem. It probably is. We show NP-hard and coNP-hard.

- Industry standard is Martello-Toth, a heuristic for Multiple Knapsack. How effective is it?
  We propose a stochastic framework for comparing heuristics. Show that water-filling is superior to Martello-Toth.

Online Service with Delay

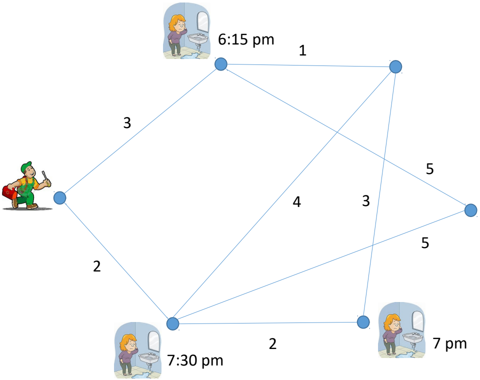
Yossi Azar
Blavatnik School of Computer Science

Arun Ganesh
Duke University

Rong Ge
Duke University

Debmalya Panigrahi
Duke University

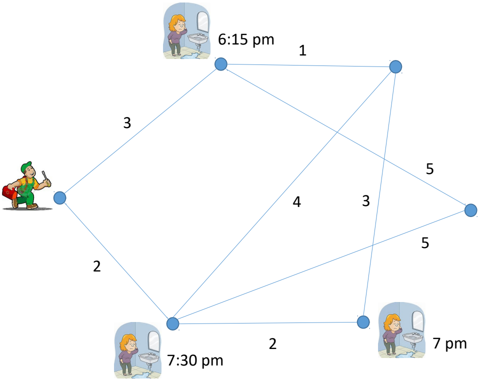
[STOC 2017]



- Service requests arrive over time
- Service can be delayed to facilitate batching with future requests in a nearby location ...
- ... but future is unknown!
- Dual objectives: minimize movement, minimize delay

- Motivation: models the fundamental tradeoff between batching requests and immediate response
  - Operating systems
  - Operations research
  - Scheduling theory

- Result: We give an algorithm with polylog(n) competitive ratio for this problem



- Service requests arrive over time
- Service can be delayed to facilitate batching with future requests in a nearby location ...
- ... but future is unknown!
- Dual objectives: minimize movement, minimize delay

- Extension: what if there are multiple (**k**) repairmen (servers)?
- Algorithm decides not only when to serve a request, but also which person to dispatch

- Result: We give an algorithm with k*polylog(n) competitive ratio for this problem

Network Scheduling

Samuel Haney
Duke University

Mehraneh Liaee
Northeastern University

Bruce Maggs
Duke University

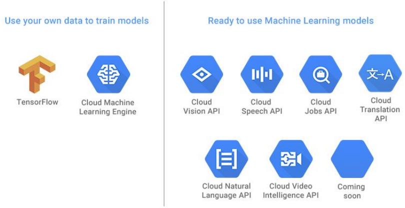
Debmalya Panigrahi
Duke University

Rajmohan Rajaraman
Northeastern University

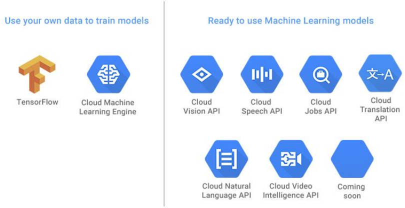
Ravi Sundaram
Northeastern University

[ongoing]

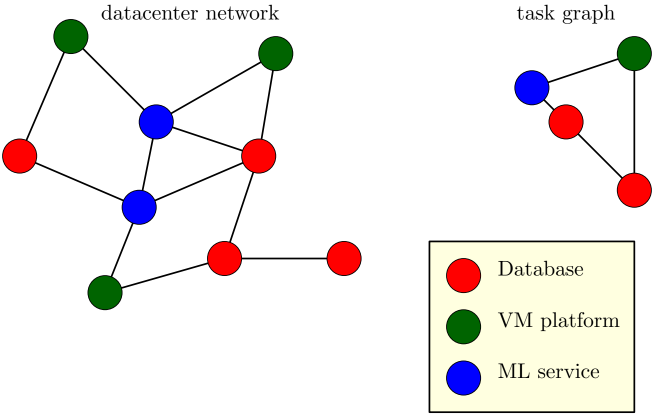
Programmers/organizations want to use cloud services for jobs.

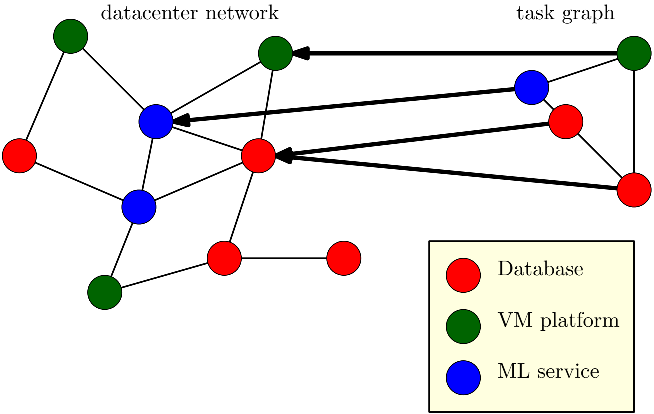


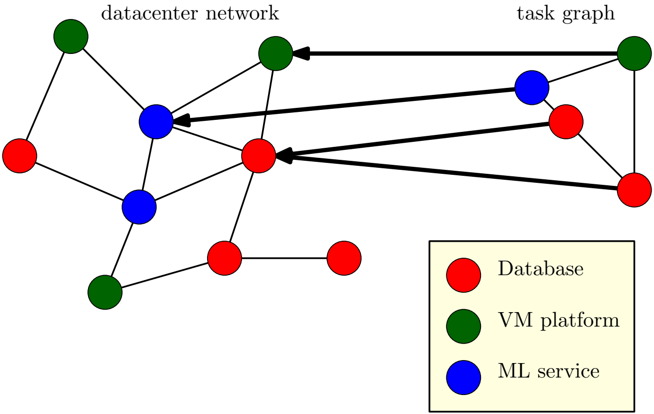
Programmers/organizations want to use cloud services for jobs.



Latency between services mainly determines the performance of a job.


datacenter network
task graph
Database
VM platform
ML service


datacenter network
task graph
Database
VM platform
ML service



Given a task graph and a datacenter network, can we produce a mapping from the tasks to the datacenter nodes?

Thank You!