15.053 April 12, 2007

An Application of Network flow Models: Mapping at Akamai

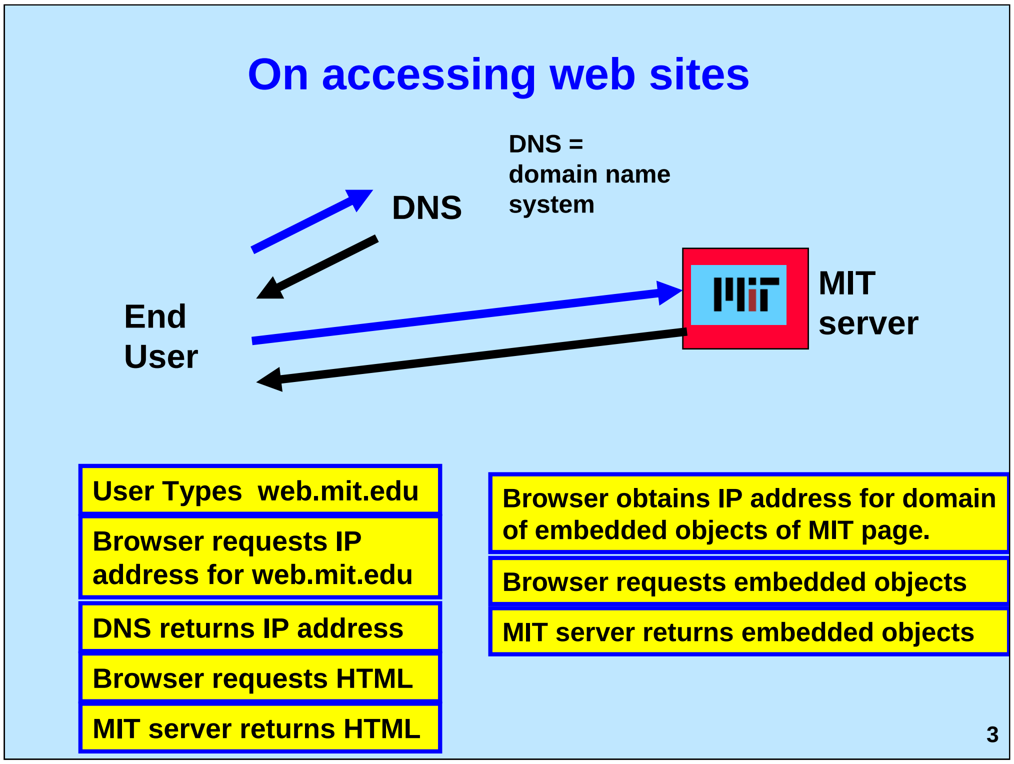
The lecture is intended to illustrate a great application of network optimization. This lecture is not intended as a promotion for Akamai, but I do think it has some very cool technology.

Truth in advertising. I bought a small amount of stock in Akamai this year for my daughter. This purchase is no way affects this lecture; however, had the stock price gone down a lot, perhaps it would have affected the lecture ☺.

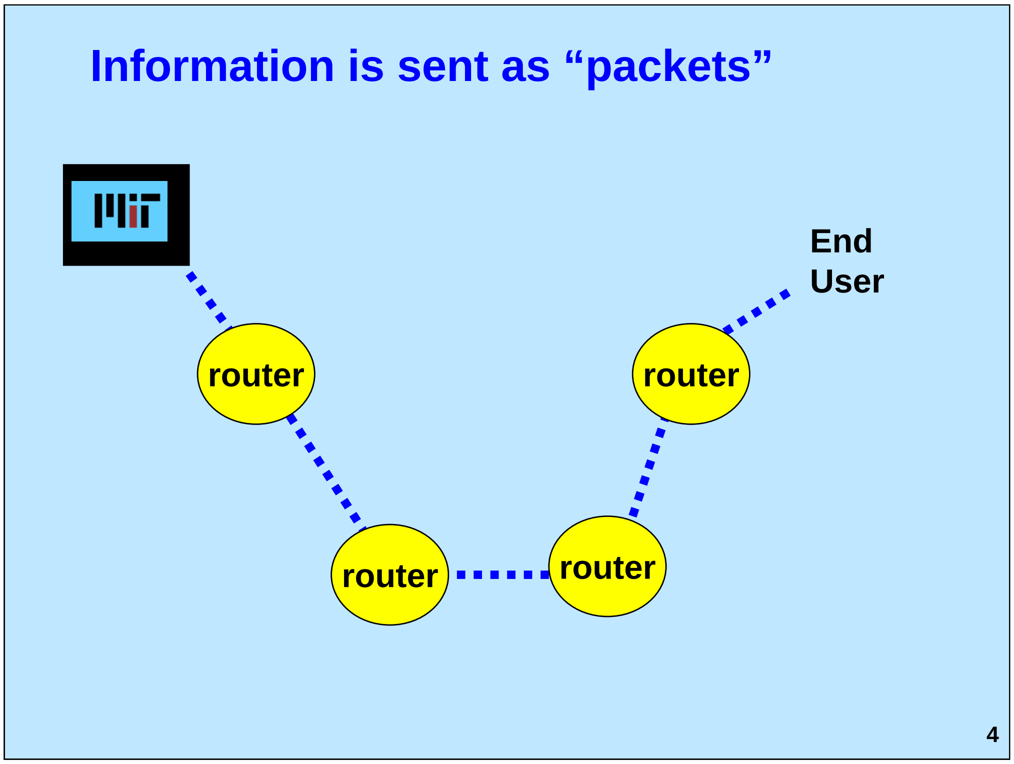
I think that the truth in advertising comment is expected by MIT. The Institute is careful about managing all potential conflicts of interest. In this case, the potential conflict of interest is very small.

Overview

- Background
  - The Internet and WWW
  - Akamai
- Multicommodity Flows and Minimum Cost Flows
- Network Modeling in Practice
- The Stable Marriage Problem and its relationship to Akamai's problem



The point of this slide is that there is a lot going on in the background when we access a web site. A user ends up accessing the “domain name system” twice and the server at least once, and there is two-way communication each time. Anything Akamai can do to speed up this process or cut down on a step will be useful.



This is to point out that information is not sent as a "steady stream". Rather, information is broken down into packets and sent from one site to another, passing through a sequence of intermediate points. The routers are in charge of directing this flow of traffic.

In the picture, you can see "dropped packets". Typically, lots of information is sent to each router. The routers have limited capacity for storing information. So, any information beyond their "buffer size" is dropped. This sounds stupid at first, but it helps make the Internet work.

If information is not dropped, then an acknowledgment is sent to the last sender of the information. If the acknowledgment is not received by the first sender of the information, then a new packet is sent. At the same time, the network protocol TCP/IP will adjust the rate that information is sent: downwards if the packets are not received and upwards if packets are received.

I find it all pretty magic that the protocol could be designed when the traffic on the network was not very large, but it still works when the traffic is 100 times greater.

A Challenge from Tim Berners-Lee 1995

"The father of the Web foresaw the congestion that is now very familiar to Internet users, and he challenged colleagues at MIT to invent a fundamentally new and better way to deliver Internet content. What he probably didn't expect was that posing such an academic problem would ultimately result in a service that is revolutionizing the Internet."

from the Akamai website

OK. this is promotion for Akamai. But it really is true that the original work leading to Akamai was based on a challenge from Tim Berners-Lee who is located at the MIT lab known as CSAIL.

Akamai: Hawaiian for "cool"

"Dr. Leighton and Mr. Lewin were joined by other scientists with expertise in computer science and data networking to develop the mathematical algorithms necessary to handle the dynamic routing of content and solve what was, by then, a frustrating problem for Internet users."

Akamai launched in 1999 with Yahoo as a charter customer.

Truth in advertising: there are other competitors to Akamai in the delivery of content. But Akamai is the largest content deliverer.

Actually, it's not clear whether "cool" is the best translation, but it was what Dr. Leighton and Mr. Lewin were thinking of.

An Obvious Solution to Web Delivery Issues

If a content provider (such as Yahoo or USA today) is having difficulty in delivering, why not just increase its number of servers and the speed of its servers?

Brief answer. For very large service providers, this doesn't work.

The problem with increasing the number of servers at your site is that congestion can be caused somewhere other than your own servers. It can be at a nearby router, or some other place on the Internet.

Background on the World Wide Web and the Internet

- **100s of millions of users**
- **Broadband increases the load on the network**
- **No one owns the internet**
  - **The Internet is over 7000 different networks**
  - **The largest owner controls less than 7% of it.**
  - **Packets often pass through many different networks**
  - **Congestion can appear anywhere on the path from the content to the user**
- **Routing algorithms ignore congestion**
- **Routers go down, connections can be lost**

One of the major difficulties is that no-one owns the internet. The networks are owned by 7000 different owners.

Imagine if to send snail mail to another person, it had to pass through 10 different Postal Services. No one would be taking full responsibility, and it would make the likelihood of the mail getting through very unlikely. In addition, when you send a letter, only one postal service would get money, which would provide less incentive for the other services to do a good job.

As for traffic on the Internet, it seems to be constantly growing, and broadband has increased the amount of traffic significantly.

Finally, routing algorithms throw away packets when too many packets are received. And so, many packets have to be resent. And routers go down.

Frankly, it’s pretty amazing that the whole thing works so well.

The Akamai Solution

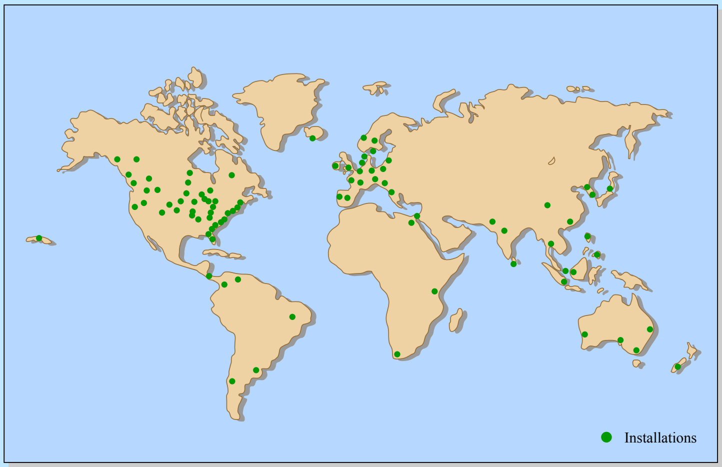


Figure by MIT OCW.

20,000 Servers **2800 sites** **71 Countries**

9

Akamai's solution is to store duplicate content at a location near you. So, if you want to access the Yahoo site, you can access the information at a server near you, whether you are located at MIT, or Stanford, or in Argentina, or in Australia, or in Europe, or in Africa. (Actually, the odds are that servers are not very close if you are in Africa, but there is not a lot of Internet traffic there either.) So, when you ask for information from a website, the packets don't need to travel very far.

On the Improvement of Content Delivery

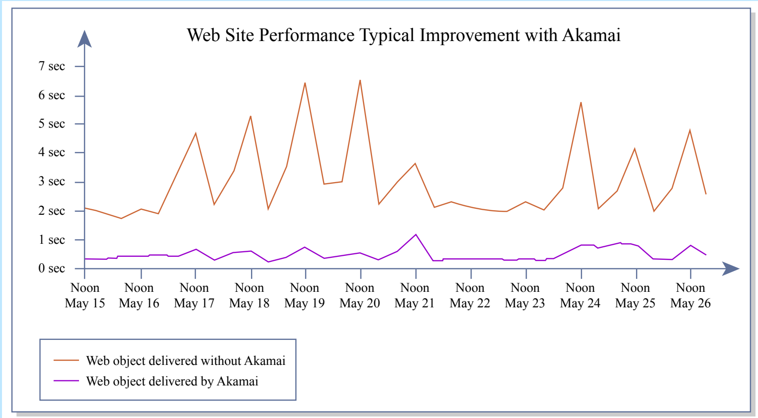


Figure by MIT OCW.

From Tom Leighton's talk 2002 10

This slide was taken from Tom Leighton's talk in 2002. You can see that Akamai really improves performance. And I trust Tom to give accurate information.

Akamai’s Solution

- Store “mirrored” content on 20,000 servers located in 71 countries
- When a user clicks on a URL (say Yahoo), the content is delivered from the “closest” Akamai server.
  - For 85% of end users, the packets pass through at most two networks
- Results:
  - Faster, more reliable delivery of web information
  - 10% to 20% of all web traffic use Akamai servers

Normally, when we think of mirror sites, we assume that the user has to choose which site to connect to. However, Akamai constantly adjusts what mirror site you are connected to so as to deliver content most efficiently.

Passing through at most 2 networks is really helpful.

I am totally amazed that Akamai carries 10% to 20% of all web traffic.

More on Content Delivery

- **Air Force Air Combat Command Geospatial Information Office downloads of satellite images (2006)**
  - before Akamai: 2 hours (1 GB file)
  - with Akamai: 25 minutes
- **From BET (Craig Maccubbin CTO, 2002)**
  - "On Monday our traffic doubled, so we added two new servers with no effect. Tuesday afternoon we called Akamai. Tuesday night we were Akamaized and instantly 6-10 times faster
- **Akamai now routinely delivers more than 200 Gb per second at times** from Akamai website
- **Akamai helps securely enable more than $36 billion in annual e-commerce for its online retail customers** from Akamai website

OK. This is more promotional stuff from the Akamai website.

Some Critical Technical Issues

- **Mapping Problem: How does Akamai determine which server will download web materials to which user? (Mapping users to servers).**
- **Storage Problem. What content is stored on each server?**
  - **Total content from Akamai clients vastly exceeds storage space of servers**
  - **How does Akamai ensure that content is up to date?**
- **Web Addressing Issues. How does Akamai make sure that all of this happens in a way that is invisible (transparent) to the end user?**

All of the previous material was leading to some interesting optimization problems. The first two are really interesting optimization problems. The third problem requires lots of knowledge of the web and how it works and is a really interesting information technology problem.

Overview

- In about 10 minutes, I will ask you to think about Akamai's mapping problem and how you might solve it.

- Background to be presented first
  - the multicommodity flow problem
  - min cost flow problem
  - the problem that Akamai would like to solve if possible

Multicommodity Flow Problem

- This is what occurs when there are a lot of minimum cost flow problems all sharing the same network.

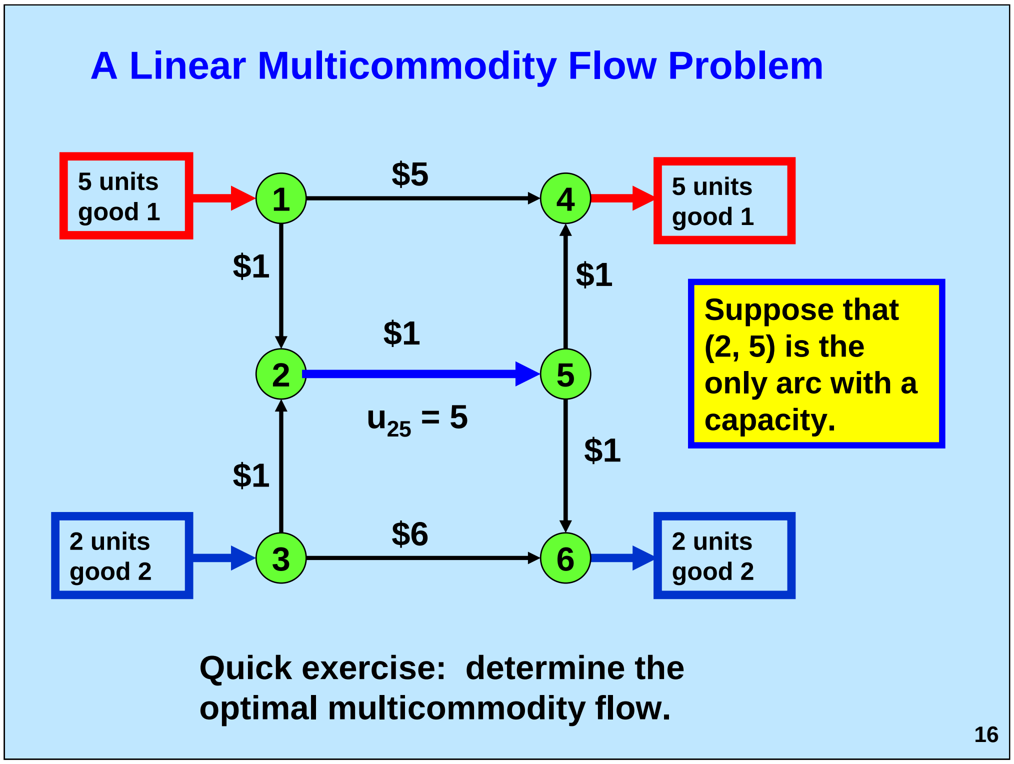
- Example:
  - Source node s_1 needs to send flow to sink node t_1
  - Source node s_2 needs to send flow to sink node t_2
  - The total flow on arcs is bounded

The multicommodity flow problem is arguably the most common type of network flow problem to arise in practice. The reason is simple. Usually, the same network serves a wide range of users.

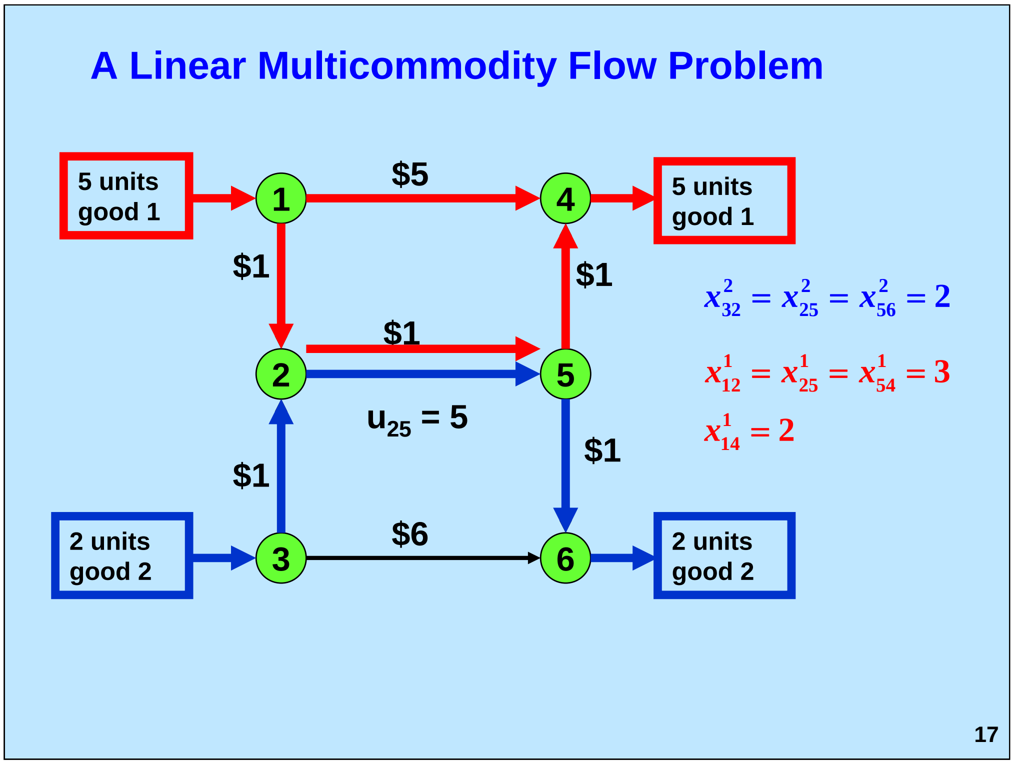
In a communication network, the communication sent from user 1 to web site W1 is typically different than the communication sent from user 2 to website W2. (This is obvious. People are not sending the exact same information.) Each user in interested in his or her own flow of communication, and all the users share a common network.

Similarly, a transportation network (or road network) is shared by lots of different drivers, all going their own directions.

And an airline company typically has several different types of airplanes, and one must keep track of them separately, even if they do fly on the same network.



This is a simple network in which there are two commodities. If there was unlimited capacity, the first good would travel on the path 1-2-5-4 and the second good would travel on the path 3-2-5-6. Unfortunately, this would send too much flow on arc (2, 5). It is this shared arc capacity that makes multicommodity flow problems challenging to solve in general. (This particular instance is not so hard to solve.)



Here is the optimal solution.

On the Multicommodity Flow Problem O-D version

K origin-destination pairs of nodes
$(s_1, t_1), (s_2, t_2), \ldots, (s_K, t_K)$

Network G = (N, A)

d_k = amount of flow that must be sent from s_k to t_k.

u_{ij} = capacity on (i,j) shared by all commodities

c_{ij}^k = cost of sending 1 unit of commodity k in (i,j)

x_{ij}^k = flow of commodity k in (i,j)

The Multicommodity Flow LP

$$\text{Min} \sum_{(i,j)\in A} \sum_k c_{ij}^k x_{ij}^k$$

$$\sum_j x_{ij}^k - \sum_j x_{ji}^k = \begin{cases} d_k & \text{if } i = s_k \\ -d_k & \text{if } i \in t_k \\ 0 & \textit{otherwise} \end{cases}$$ Supply/demand constraints

$$\sum_k x_{ij}^k \le u_{ij} \quad \text{for all } (i,j) \in A$$ Bundle constraints

$$x_{ij}^k \ge 0 \quad \forall (i,j) \in A,\ k \in K$$

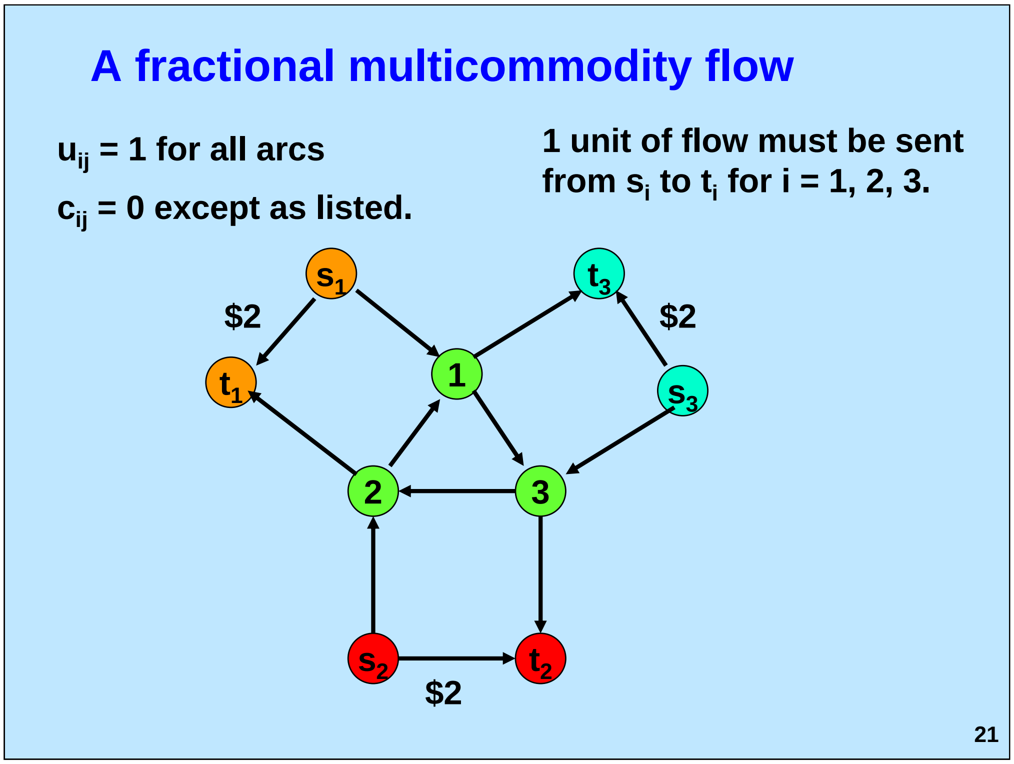
This looks like k different flow problems in which the objective is to send d units of flow from a source node to a sink node at minimum cost. If we ignored the shared capacity constraints, we would solve the problem as K shortest path problems, one for each commodity. This is because, the optimum flow from a source to a sink would be along the min cost path.

However, we also have the shared capacity constraints, also known as bundle constraints. This makes the multicommodity flow problem much harder to solve than a series of shortest path problems.

Facts about Multicommodity Flows

- The multicommodity flow problem with K commodities takes <u>much longer</u> to solve than solving K single commodity flow problems.
- The multicommodity flow problem if solved as an LP usually has fractional optimal flows.
- Finding integer optimum flows is incredibly hard and time consuming

Multicommodity flow problems usually result in fractional (non-integer) flows even if the data is integer valued. If we require flows to be integer valued, the problem becomes even harder to solve.



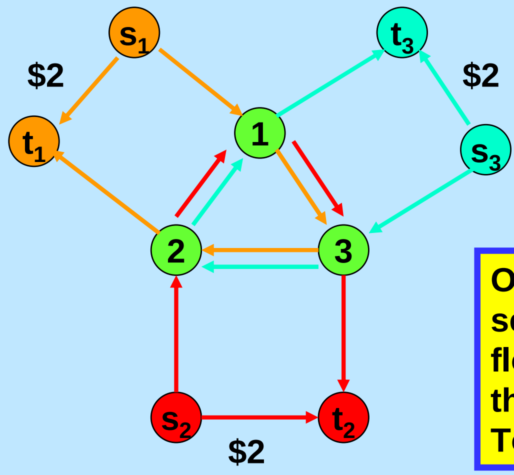
I didn't present this slide in the lecture and it is "hidden." This slide and the next "hidden" slide show an example in which the optimal solution to the multicommodity flow problem is fractional.

A fractional multicommodity flow

u_{ij} = 1 for all arcs

c_{ij} = 0 except as listed.

1 unit of flow must be sent from s_i to t_i for i = 1, 2, 3.



Optimal solution: send ½ unit of flow in each of these 15 arcs. Total cost = $3.

Running times: for the purposes of this lecture

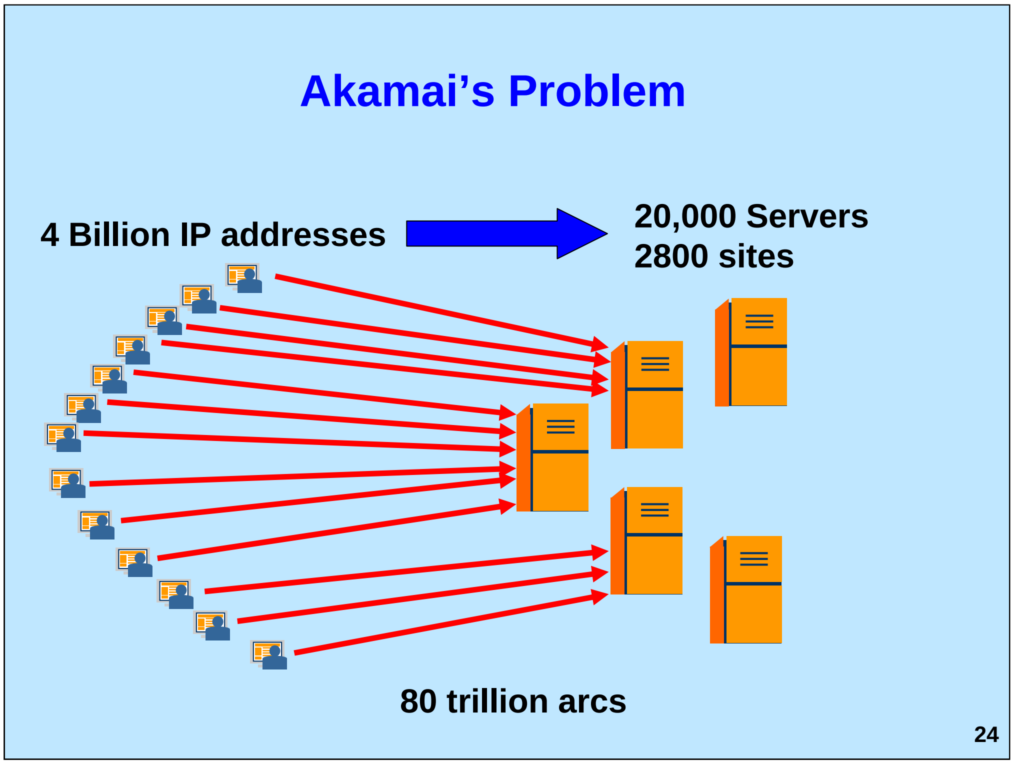
# of Arcs	CPU Time
100,000	30 sec.
200,000	1.5 minutes
500,000	4 minutes
1,000,000	10 minutes
2,000,000	25 minutes
5,000,000	70 minutes

Min Cost flow running times

Assume that a multicommodity flow problem with 10 commodities takes 50 times as long to solve.

Not really correct, but they are close enough for our purposes.

These numbers are very rough estimates. But they will be useful for our purposes.



Akamai's problem is set up as a single commodity flow problem with 80 trillion arcs. And it's not fully clear what the costs are on each arc nor what the supplies and demands are. Nevertheless, we are going to make an attempt to structure the problem so that it can be solved using network flow algorithms.

More on Akamai’s Problem

- **Akamai cares about the type of content being mapped**
  - **web downloads,**
  - **live streaming (audio and video)**
  - **secure content, java, …**
- **Not all content is cached at the servers**
- **Wants to balance load**
- **Wants very fast times for end users**
- **Wants to update solutions at least once a minute**

25

The last slide made it look like Akamai’s problem is a single commodity flow problem. In fact, it may be a multicommodity flow problem since Akamai wants to differentiate between web downloads, live streaming, secure content, and other forms of data.

While the “objective” is not fully clear, Akamai knows how to recognize a pretty good solution. It is one in which the times will be fast for end users. And the speed will be faster if the loads between machines is balanced and if each end user communicates with a server close to him or her.

In addition, Akamai wants to update the solution frequently. This frequent update is not necessary 95% to 99% of the time. But it is needed when there is a significant change in usage (such as a live streaming event) or if servers go down or routers go down. Moreover, the run time of the algorithm has to be very fast at exactly those times that the problem has changed dramatically from the previous minute.

Your Problem

- You have 5 minutes to try to simplify Akamai's problem enough that it can be solved. Also, what is your objective?
- If your solution depends on answers to questions, you can try asking me the question.
- Note: Akamai devoted several incredibly smart PhDs in computer science full time to deal with this problem.
- Discuss the problem with one or two classmates now.

How does one make a much smaller problem that is still useful to Akamai?

What information needs to be collected? Is it available?

How does one assign costs?

What does one do with the different types of content?

Mental Break

Feedback from Groups and Class Discussion

- **What problems did you solve? Why? What were the objectives? What tradeoffs did you make?**

28

This is the first time this lecture has been given. So, I can't give typical examples of answers.

More discussion

An Approach for Akamai**

Some wisdom for Akamai: If capacity for the servers is really tight, then Akamai needs more servers. It should not operate "on the margin."

Conclusion: there are incredible numbers of good solutions. The goal is to avoid bad solutions.

**** The writer of this slide does not claim that this and what follows is the approach that Akamai is taking. We only claim that it is a sensible approach, and one that Akamai may have considered.**

A natural concern in algorithm design is to do well under adverse conditions, such as when capacity is extremely high and capacity is largely used up. However, if Akamai ever is in a situation when the capacity is in danger of being used up, then the major issue is not performance of the algorithm. The major issue is that Akamai would not have nearly enough capacity.

Put another way, Akamai should have lots of capacity, and there should be lots of ways of making very good assignments. What Akamai really wants to do is to avoid bad assignments of users to servers.

Data for the problem

- **Estimate instantaneous demand for groups**
  - **Based on requests from groups in recent past + loads on Akamai nameservers**
- **Estimate current capacities of clusters**
  - **Based on load currently observed on servers**
- **Compute "cost" for mapping IP group to cluster**
  - **Based on packet loss, latency, bandwidth price, contracts, …**

Basically, Akamai will need to make lots of approximations both in the data and in the estimates of costs. But this is probably very effective if the goal is to avoid bad solutions.

Comments on an approach for Akamai

Simplification 1: divide the 3 billion IP addresses into around 2000 groups.

Simplification 2: divide the 20,000 servers into 2600 sites.

Simplification 3: List only around 25 arcs per group.

Simplification 4: If a group gets assigned to more than one site, then "round off" so that it gets assigned to exactly one site.

A Second Phase in an Approach

- **Once the user-groups are assigned to the server-locations, a second algorithm can figure out how to assign each group to a specific server, depending on the type of content that the user in the group wants.**

The point of this slide is that Akamai can ignore the type of content when assigning user groups to clusters of servers. But after making the initial assignment, it can solve another problem for each user group and each cluster in which they focus on the type of content.

General Tradeoffs in Modeling

- What data can you access?
- How accurate is the data?
- What are the constraints and objectives?
  - How do you trade off on different objectives.
- How do you model uncertainty?
- What is the solution technology and how well does it work?
- What are you willing to tradeoff in model accuracy to get fast solution times?

In modeling decisions, one constantly has to make approximations and simplify, and this leads to tradeoffs.

Better data is more useful, but it is expensive to change database systems to collect the right information.

Accurate data is much better, but there is a big expense to cleaning data.

It would be helpful if there were a single objective such as minimizing cost. But usually, there are many objectives. For Akamai, they cannot do poorly for one client if it helps out another client. In a sense, each client has their own objective, and Akamai needs to be sensitive to all of these objectives.

The real world is full of uncertainty. Modeling uncertainty directly may make it too difficult to obtain data and to solve problems. But uncertainty cannot be ignored.

The min cost network flow model was used in large part because the problem can be solved so quickly. Any modeler must be sensitive to the speed at which a model can be solved.

One Other Possibility

- **Solve a much larger problem, but sacrifice in detailed modeling**
- **Model as Stable Marriage Problem.**

The “Stable marriage problem” is a classic problem in combinatorial optimization. And it may have some value to Akamai. We will describe the problem and the algorithm in the next slides.

Stable Marriage Problem and Algorithm

- **Assign men to women (or groups to servers)**
- **Develop an assignment that is "stable", which we will define soon.**

Lois

Brian

Wilma

Cleveland

Jane

Quagmire

The evil queen from Snow White

Peter

The men and Lois are all characters on Family Guy. Wilma is from the Flinstones, and Jane is from the Jetsons.

Inputs for Stable Marriage Problem

- Each “man” gives a list of women, in order of preference.
- Each woman gives a list of women in order of preference.

Brian	Lois	Wilma	Jane	The evil queen from Snow White
Cleveland	Lois	Jane	Wilma	The evil queen from Snow White
Quagmire	Lois	Jane	Wilma	The evil queen from Snow White
Peter	Lois	The evil queen from Snow White	Wilma	Jane

Lois	Peter	Brian	Cleveland	Quagmire
Wilma	Cleveland	Brian	Quagmire	Peter
Jane	Cleveland	Brian	Quagmire	Peter
The evil queen from Snow White	Peter	Brian	Cleveland	Quagmire

Stability

An assignment of men to women is said to be *stable* if there is no man M and woman W so that:

1. **Man M prefers W to his mate.**
2. **Woman W prefers man M to her mate.**

Brian **Quagmire**

Jane **Wilma**

Why is any matching containing the above two pairs unstable?

The marriage here is unstable because Wilma prefers Brian (the dog) to Quagmire, and Brian prefers Wilma to Jane Jetson.

This is a pretty complicated condition to ask for. But there is a fairly simple algorithm that guarantees that all marriages will be stable.

Stable Marriage Algorithm

- **Initialize: Each man proposes to the first woman on his list.**
- **Women's move: each woman with more than one proposal rejects all proposals but one (her favorite man of those proposing)**
- **Men's move: each rejected man proposes to the next woman on his list.**
- **Alternate between women's moves and men's moves until every woman has a proposal. The algorithm then ends.**

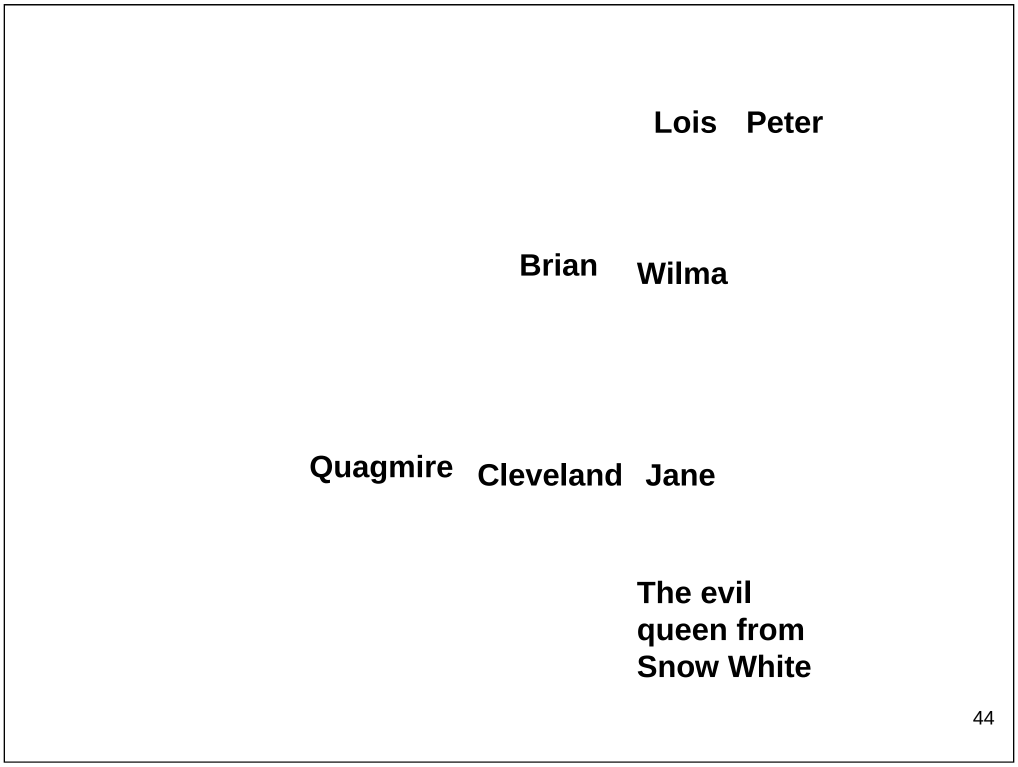
Quagmire Cleveland Brian Lois Peter

Wilma

Jane

The evil queen from Snow White

Initially all the “men” select Lois.



Lois selects her first choice Peter. Then Quagmire, Cleveland, and Brian all go to their second choice.

Lois **Peter**

Brian **Wilma**

Cleveland **Jane**

Quagmire **The evil queen from Snow White**

At this point, Jane Jetson chooses Cleveland, and Quagmire moves on to Wilma Flintstone. Wilma rejects Quagmire, who moves on to this fourth choice, the wicked queen in Snow White.

The algorithm ends with a stable marriage for each pair.

For example, Quagmire prefers every other woman, but each has rejected him for someone she prefers.

For example, the wicked queen prefers every man to Quagmire, but no other man proposed because he is with a woman whom he prefers.

Lois	**Peter**
Brian	**Wilma**
Cleveland	**Jane**
Quagmire	**The evil queen from Snow White**

46

To see that all marriages are stable, it suffices to consider the possibilities. Suppose first that a man would prefer being with another woman. For example, Quagmire would prefer being with any other woman than the wicket queen. But each of these women have rejected him. So, his pairing with the wicked queen is stable as far as Quagmire is concerned.

Now consider a woman who would prefer another man. For example, the wicked queen would prefer anyone to Quagmire, such as the dog Brian. But Brian chose Wilma because she was second on his list and was ahead of the wicked queen. So, even though the wicked queen prefers Brian, Brian prefers his current spouse to the wicked queen.

On Running Times for Stable Marriage

- **Can be slow in worst case**
  - **proportional to the number of pairs**

- **Can be used to assign most groups to servers,**
  - **e.g., 95% of all groups get assigned to servers**
  - **remaining groups get assigned using min cost flow**

- **Point: in practice, one has to assess lots of tradeoffs including using a faster algorithm that solves a slightly different problem.**

Summary

- Background on Internet congestion
- Akamai's mapping problem
- Multicommodity Flow Problem
- Stable Marriage Problem
- A Possible Solution Approach for Akamai

The point of the stable marriage problem was that it was an alternative approach for Akamai. It may be better than what they currently have, but it might not be. It's extremely difficult to know in advance of it being implemented.

The stable marriage algorithm is actually is used to assign residents to hospitals. In principle, it could in principle also be used for college admissions, but it isn't.