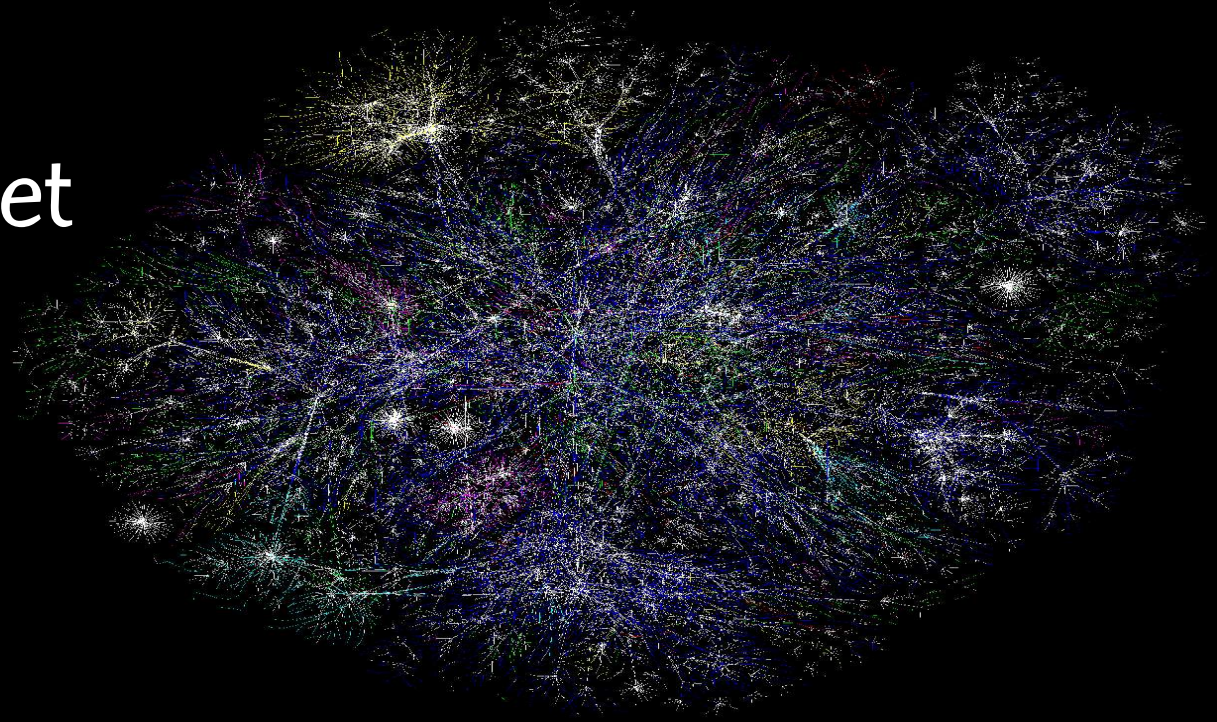
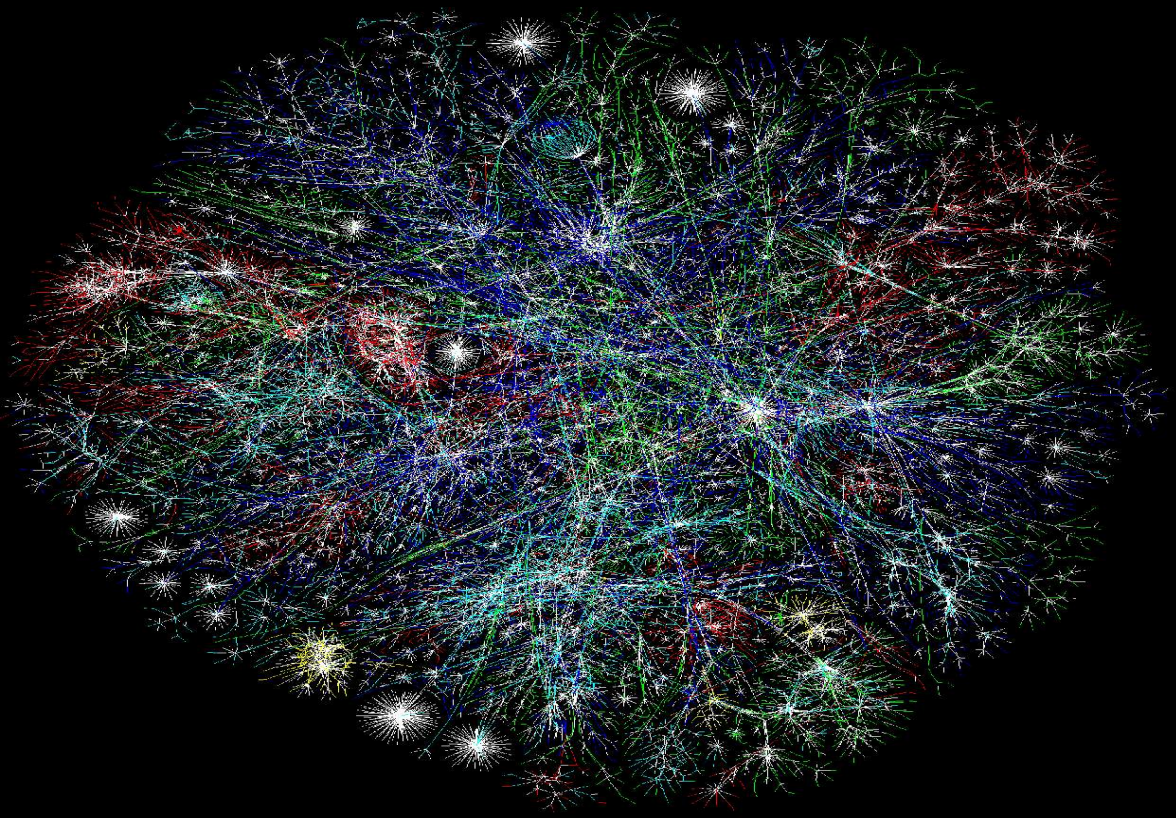
Some optimization issues arising in Internet traffic engineering



Talk given at
University of Arizona

Tucson, Arizona
November 16, 2006



Mauricio G. C. Resende

AT&T Labs Research
Florham Park, New Jersey
mgcr@research.att.com
www.research.att.com/~mgcr

Summary of talk

- IGP routing
- Survivable IP network design
- BGP routing
- Concluding remarks

Summary of talk

- IGP routing
- Survivable IP network design
- BGP routing
- Concluding remarks

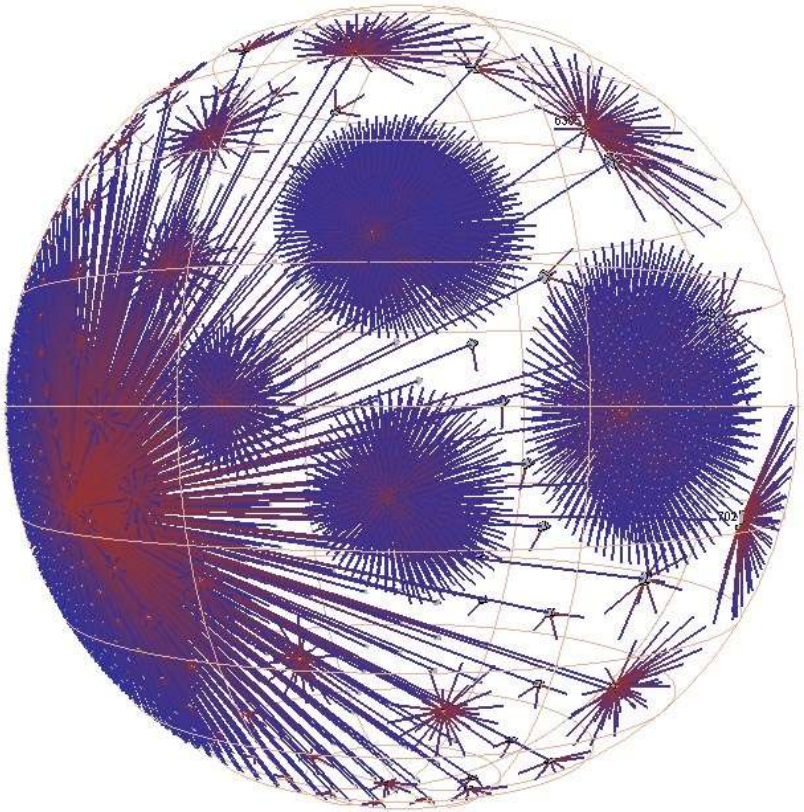
Summary of talk

- IGP routing
- Survivable IP network design
- BGP routing
- Concluding remarks

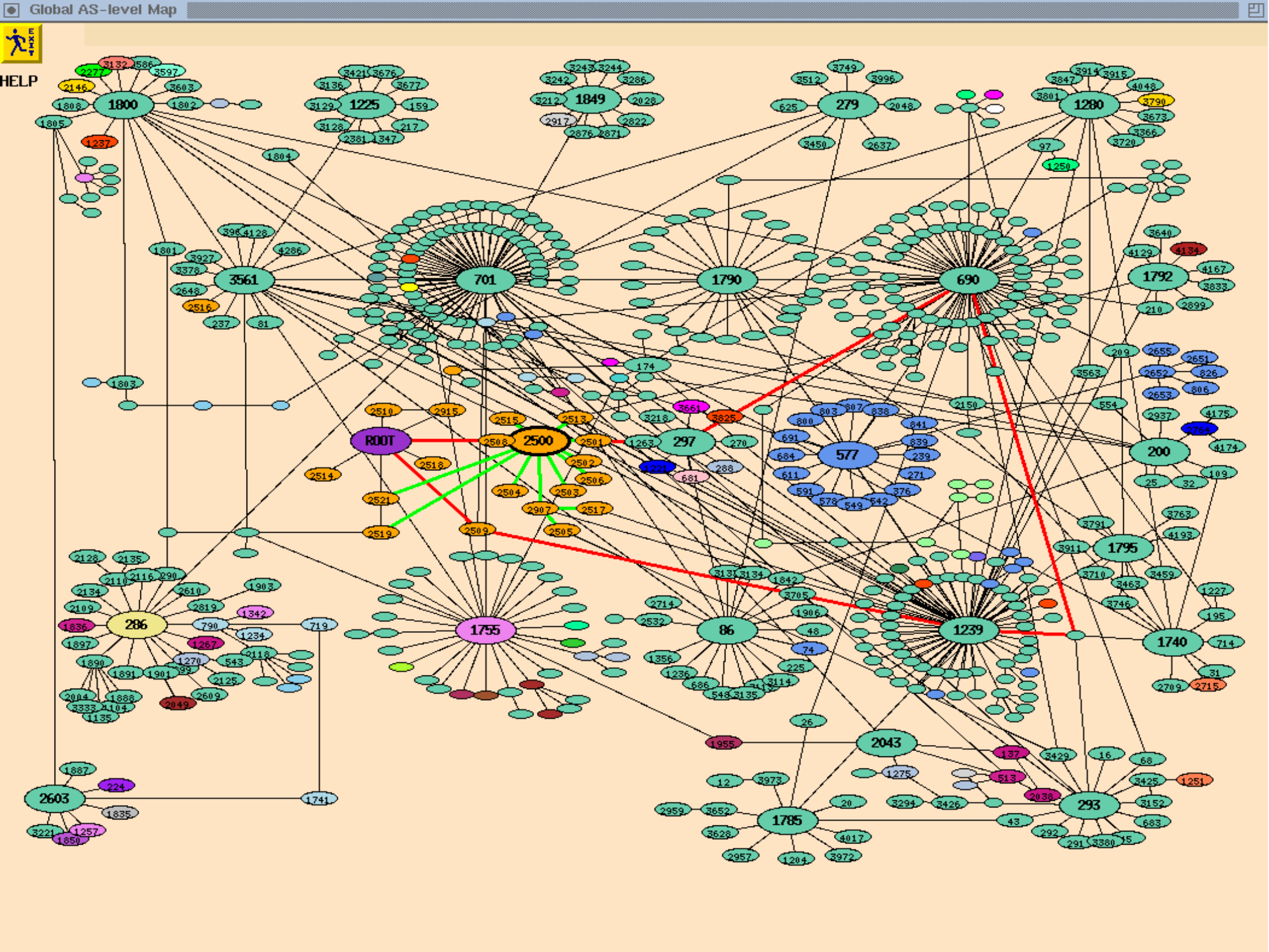
Summary of talk

- IGP routing
- Survivable IP network design
- BGP routing
- Concluding remarks

The Internet



- The Internet is composed of many (inter-connected) autonomous systems (AS).
- An AS is a network controlled by a single entity, e.g. ISP, university, corporation, country, …


Global AS-level Map
HELP
ROOT
2500
1800
1225
1849
279
1280
3561
701
1790
690
1792
297
577
200
1795
1740
1239
86
1755
286
2603
1785
293
2043

Routing

- A packet is sent from a origination router S to a destination router T.
- S and T may be in
  - same AS:
  - different ASes:

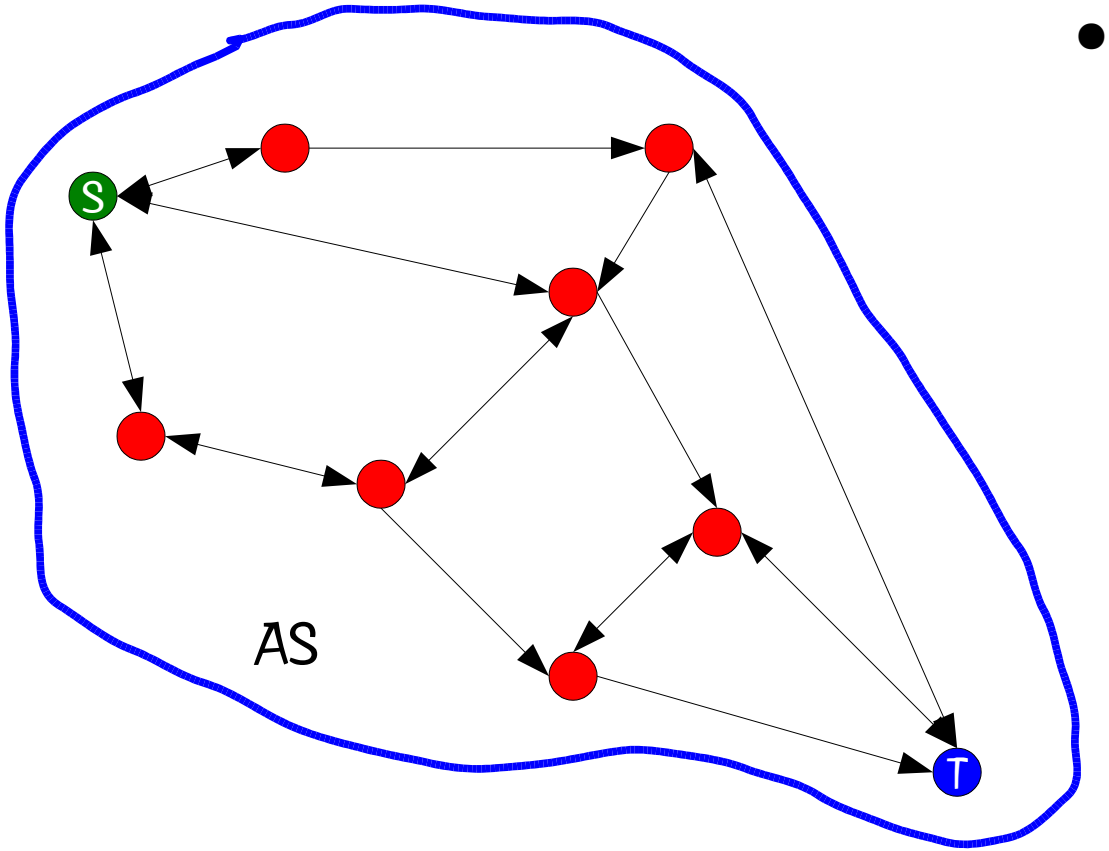
Routing

- A packet is sent from a origination router S to a destination router T.
- S and T may be in
  - same AS: IGP routing
  - different ASes:

Routing

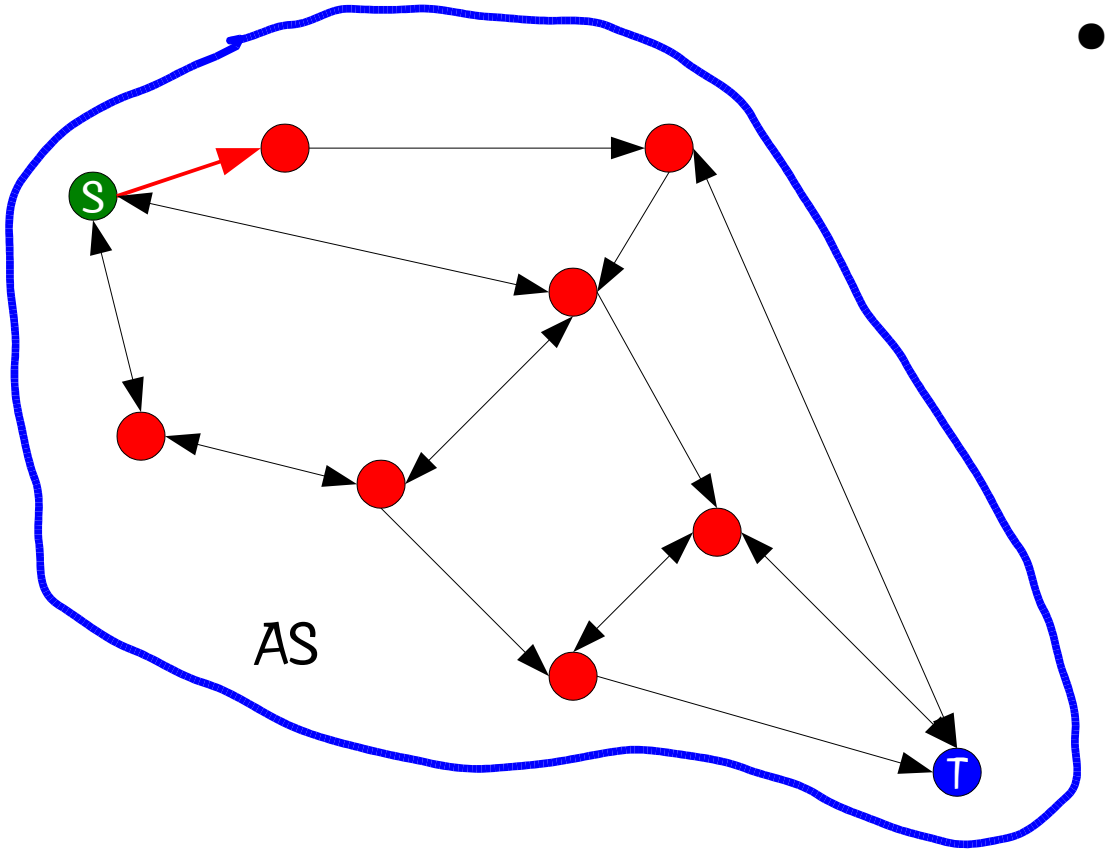
- A packet is sent from a origination router S to a destination router T.
- S and T may be in
  - same AS: IGP routing
  - different ASes: BGP routing

IGP Routing



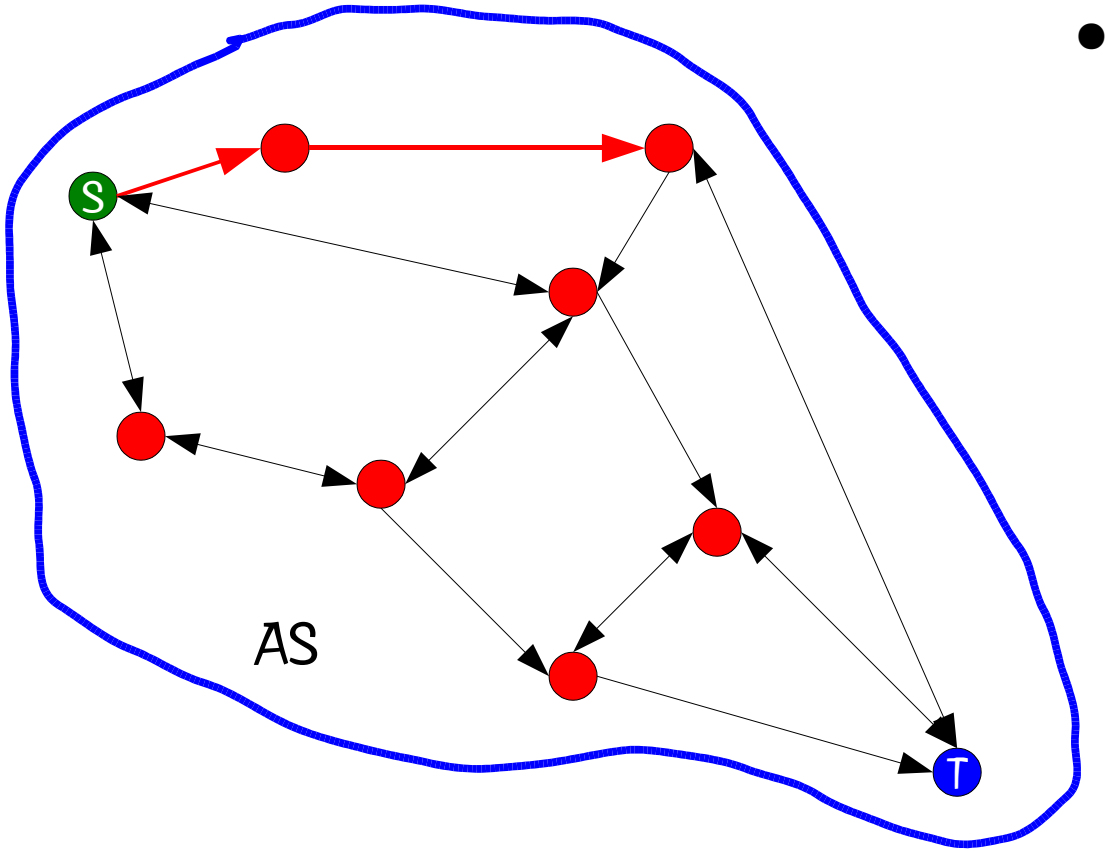
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



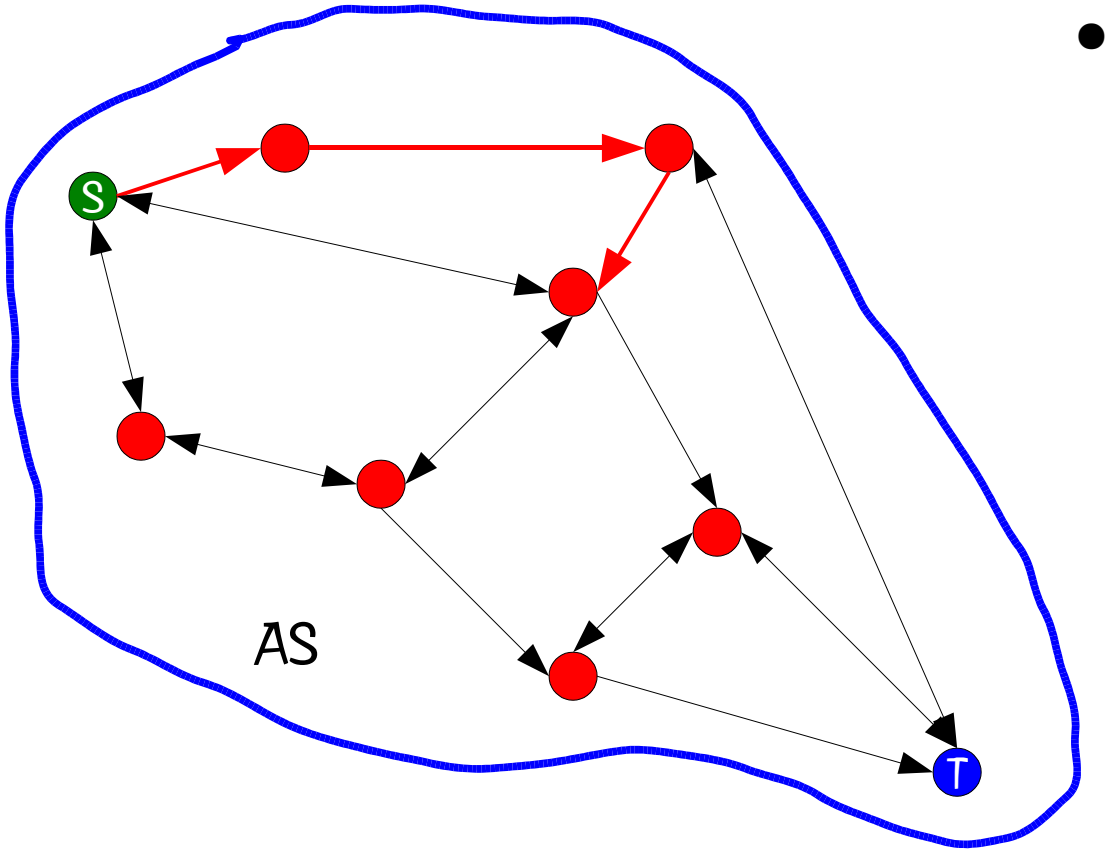
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



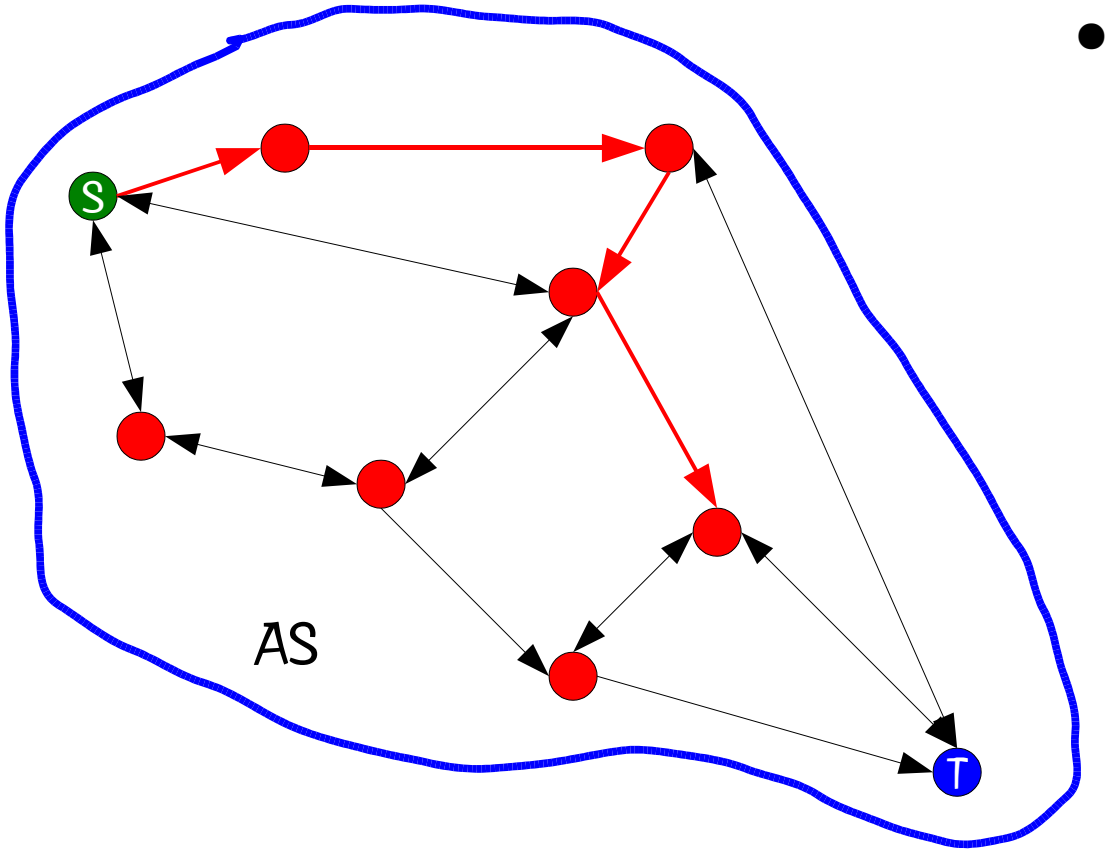
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



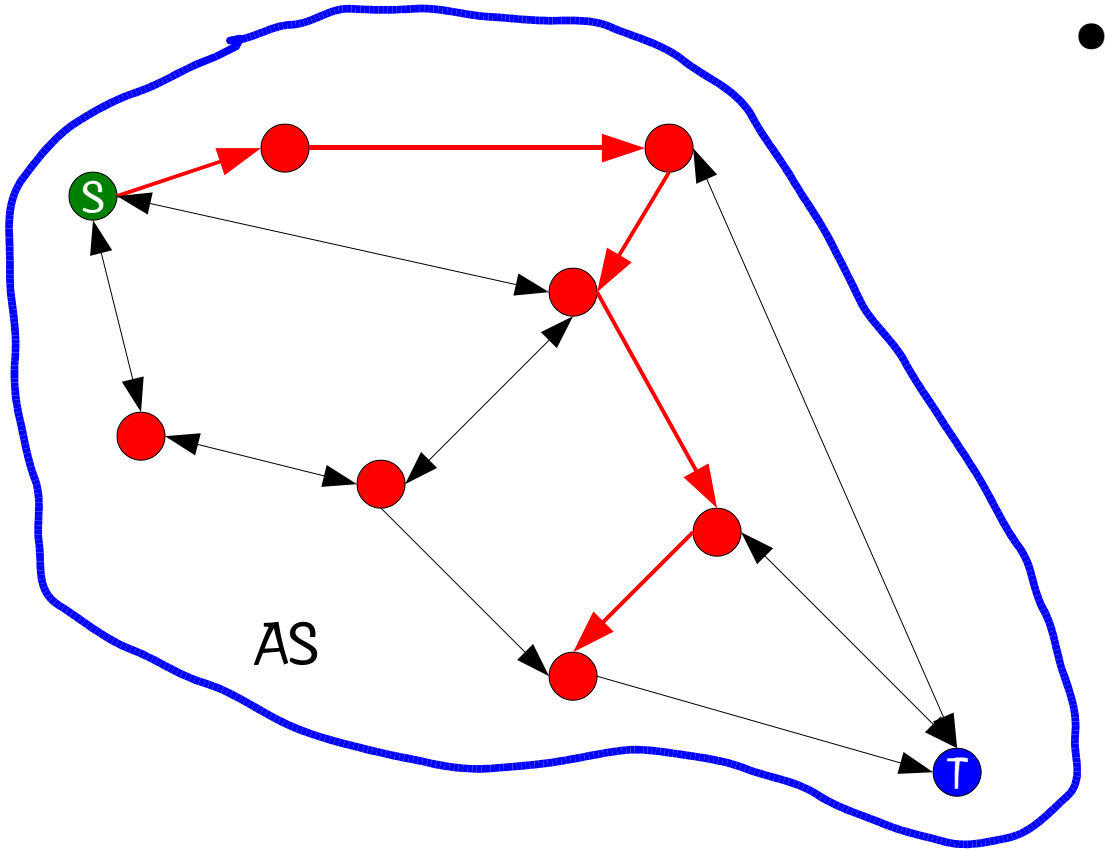
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



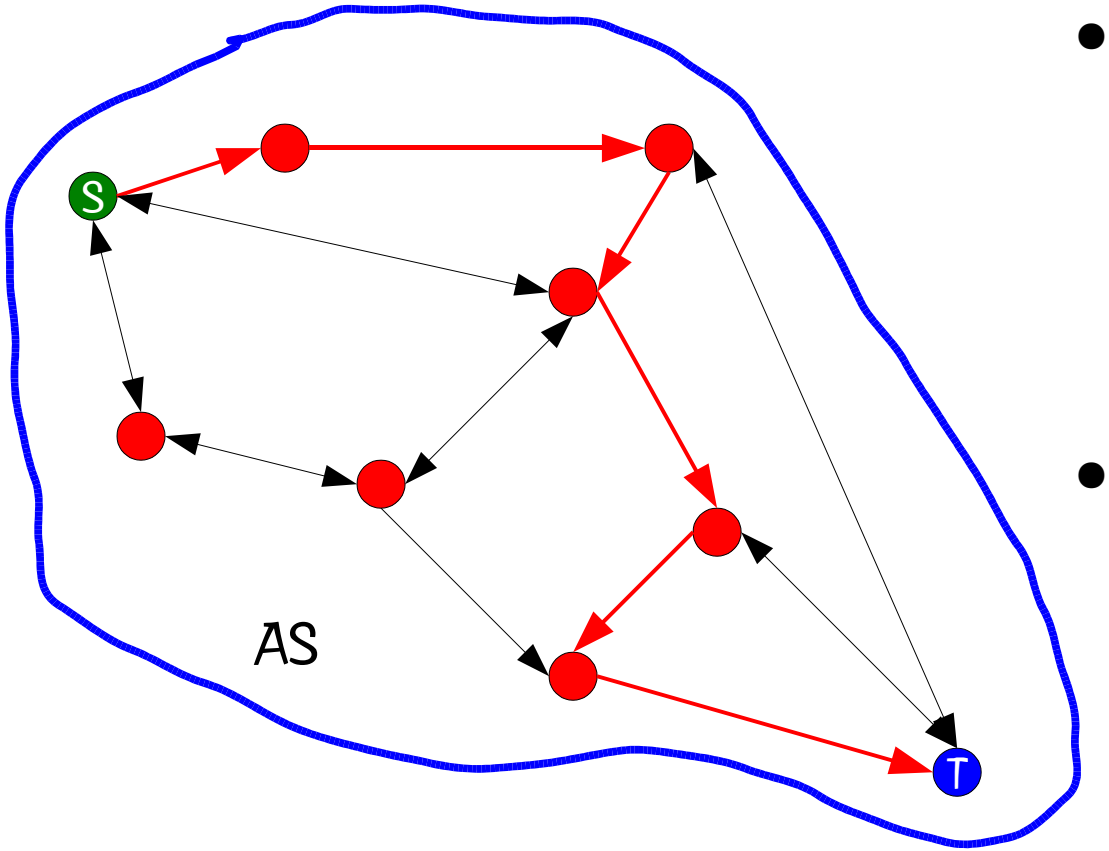
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



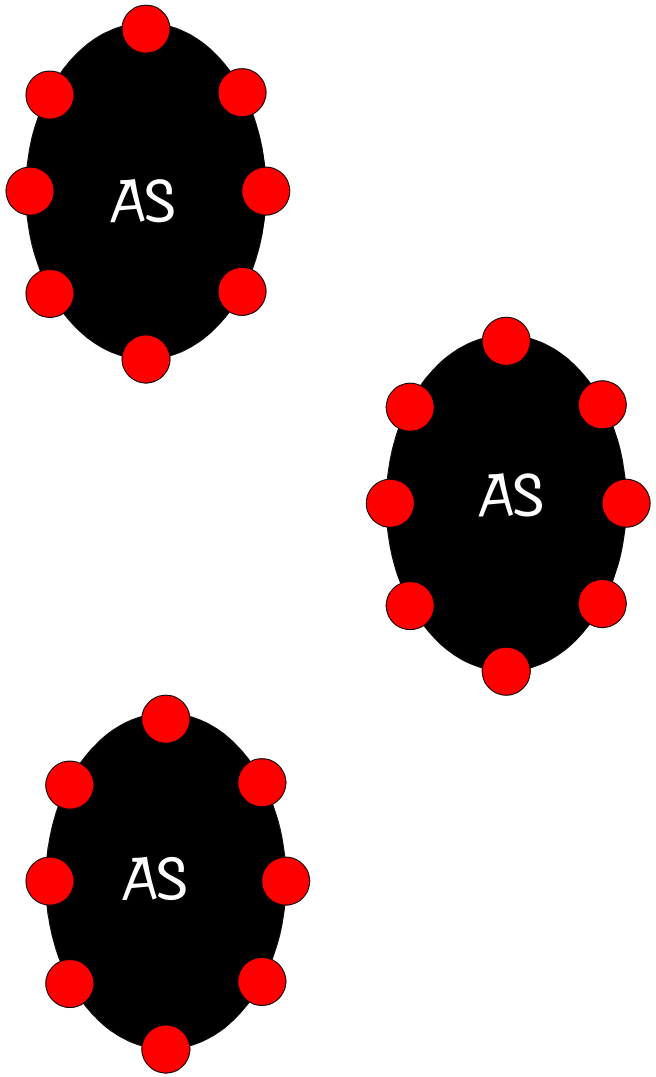
- IGP (interior gateway protocol) routing is concerned with routing within an AS.

IGP Routing



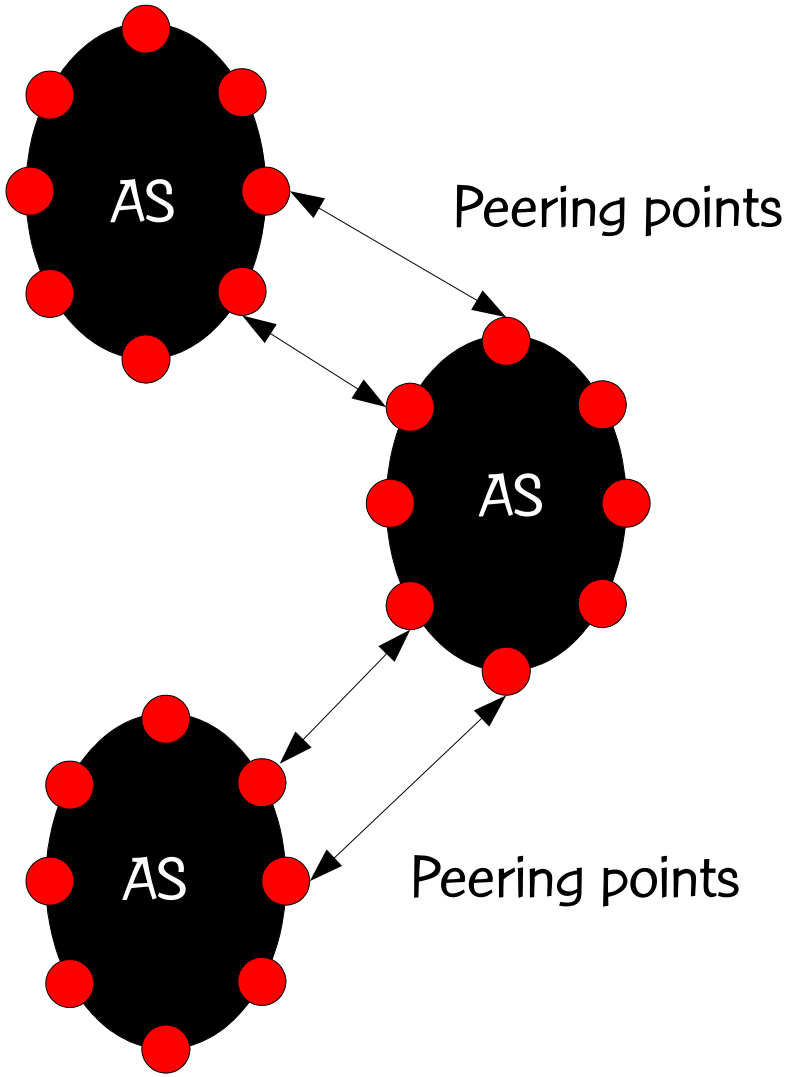
- IGP (interior gateway protocol) routing is concerned with routing within an AS.
- Routing decisions are made by AS operator.

BGP Routing



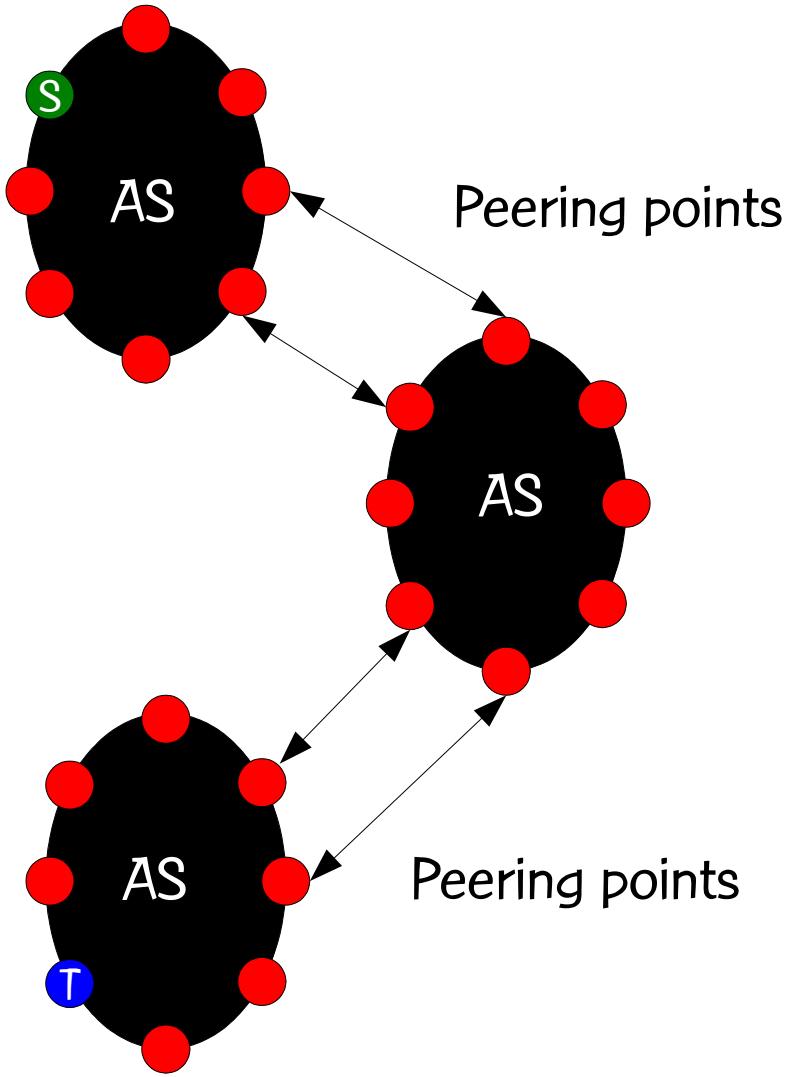
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



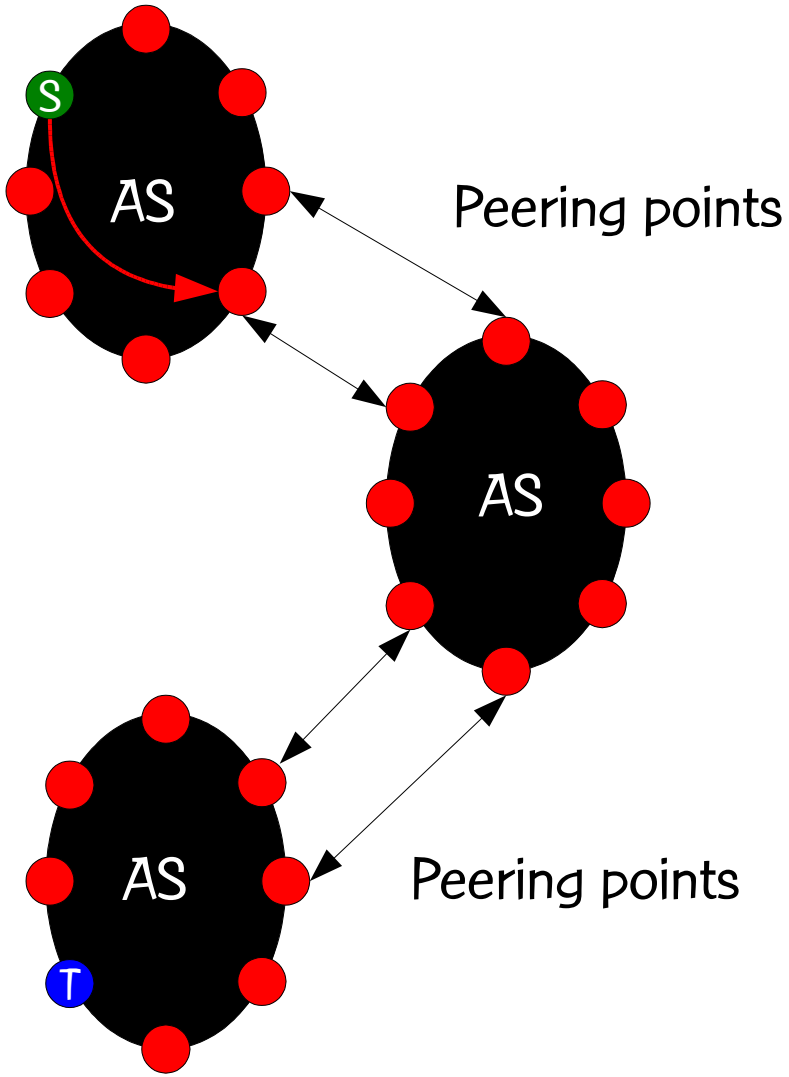
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



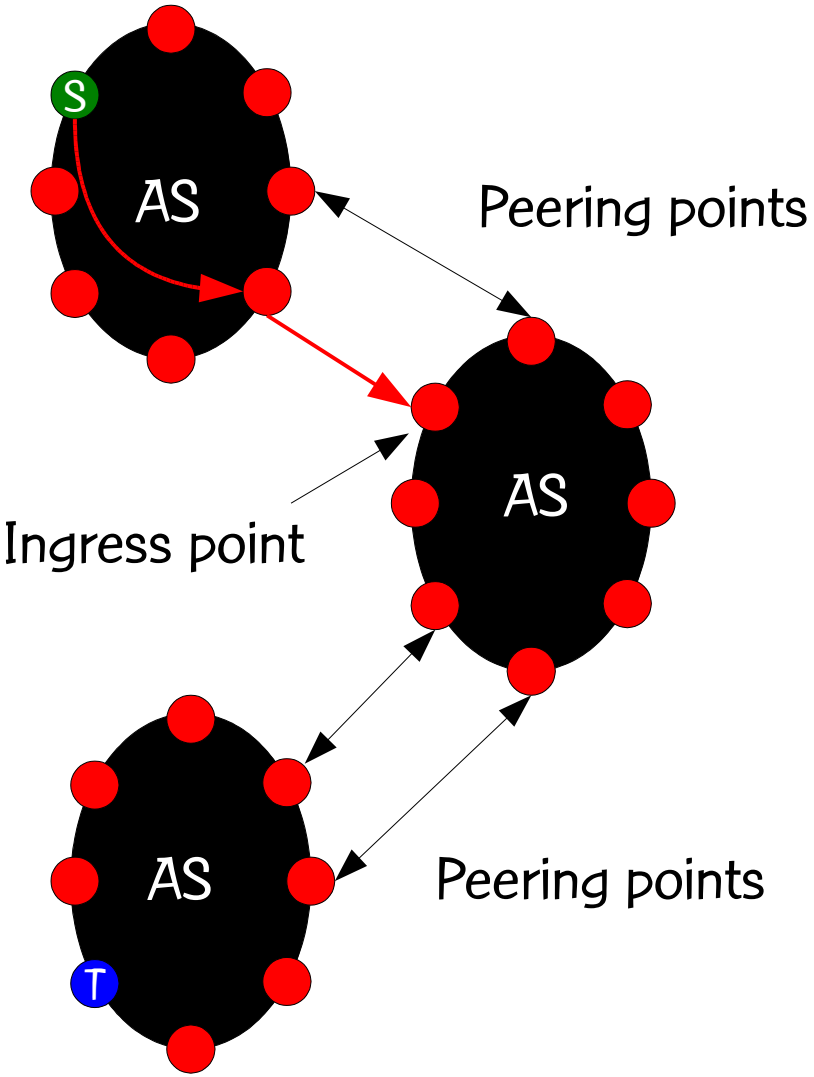
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



- BGP (border gateway protocol) routing deals with routing between different ASes.

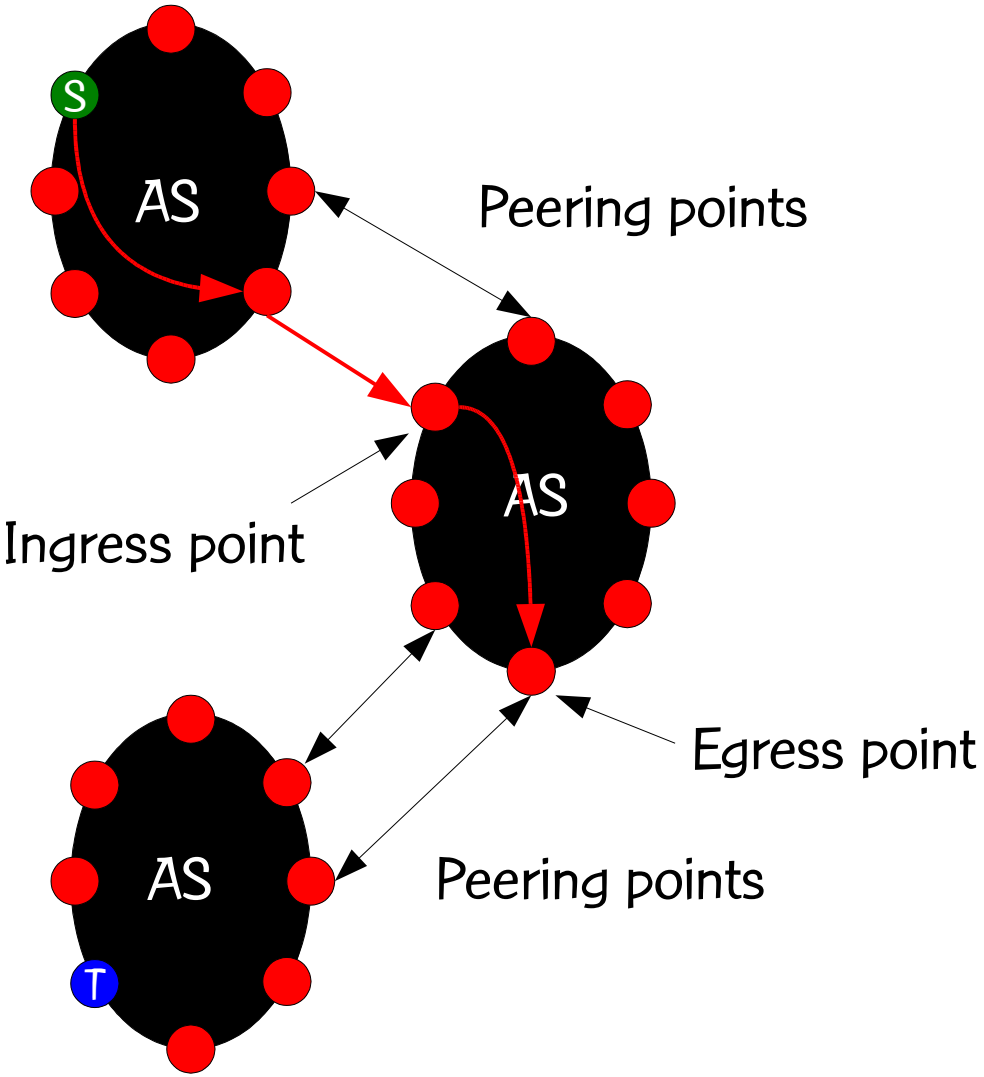
BGP Routing



- BGP (border gateway protocol) routing deals with routing between different ASes.

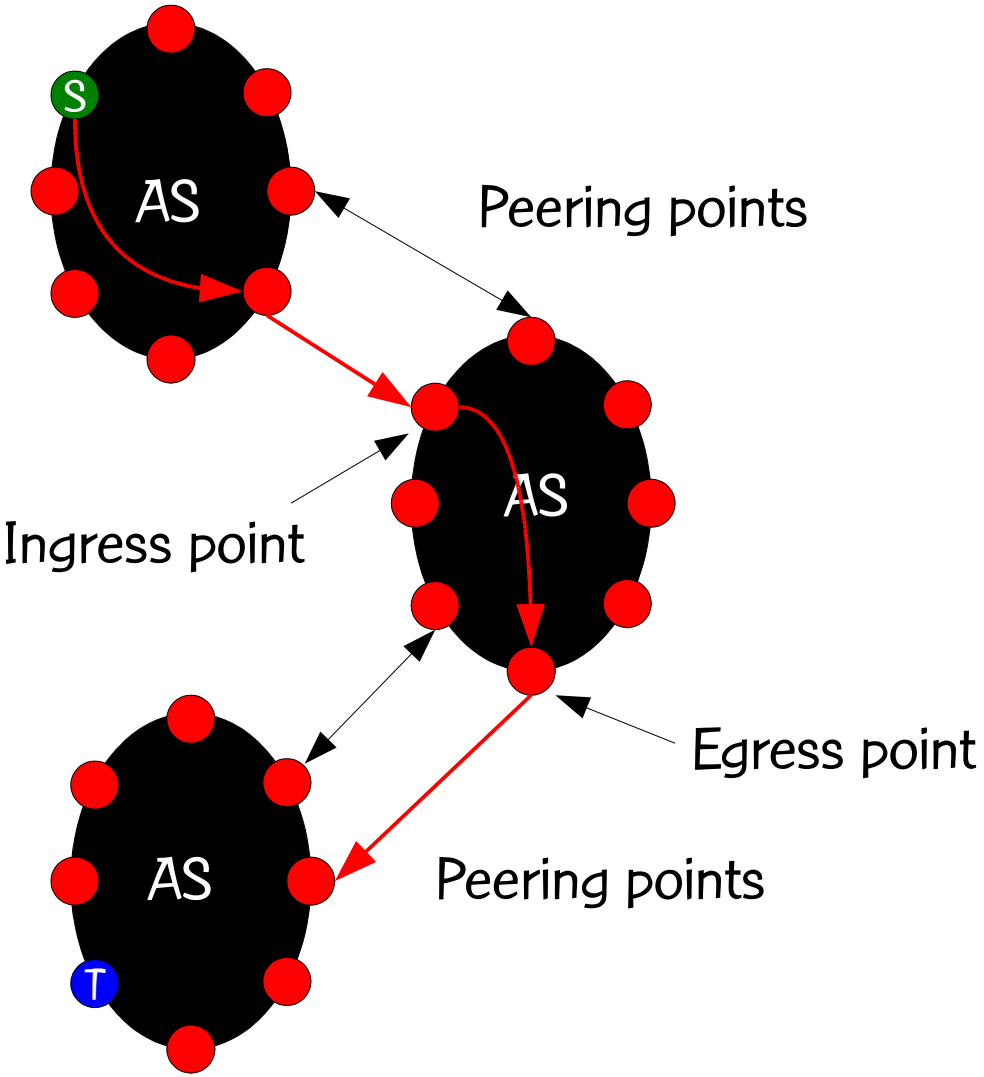
at&t
Your world. Delivered.

BGP Routing



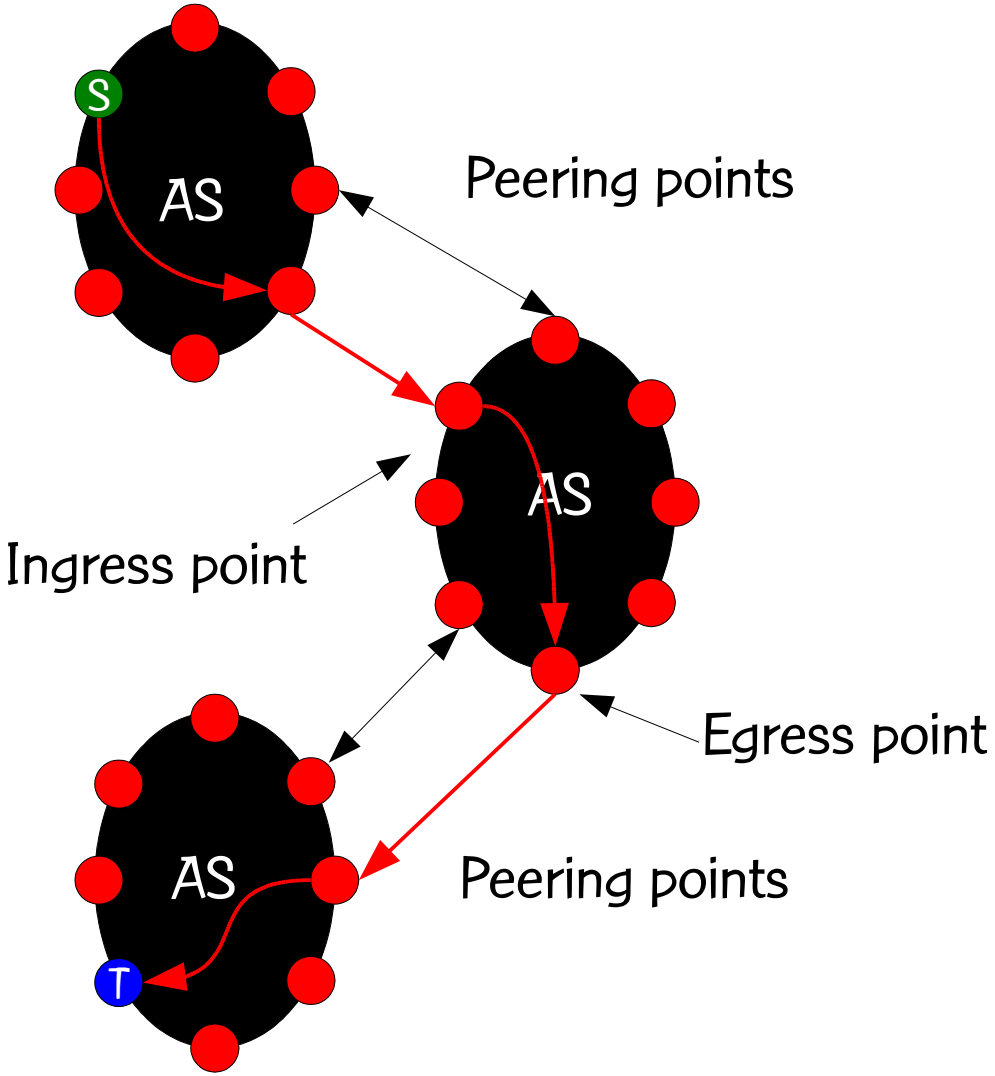
- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



- BGP (border gateway protocol) routing deals with routing between different ASes.

BGP Routing



- BGP (border gateway protocol) routing deals with routing between different ASes.
- AS operators choose egress point and route in AS from ingress point to egress point.

IGP Routing

OSPF routing

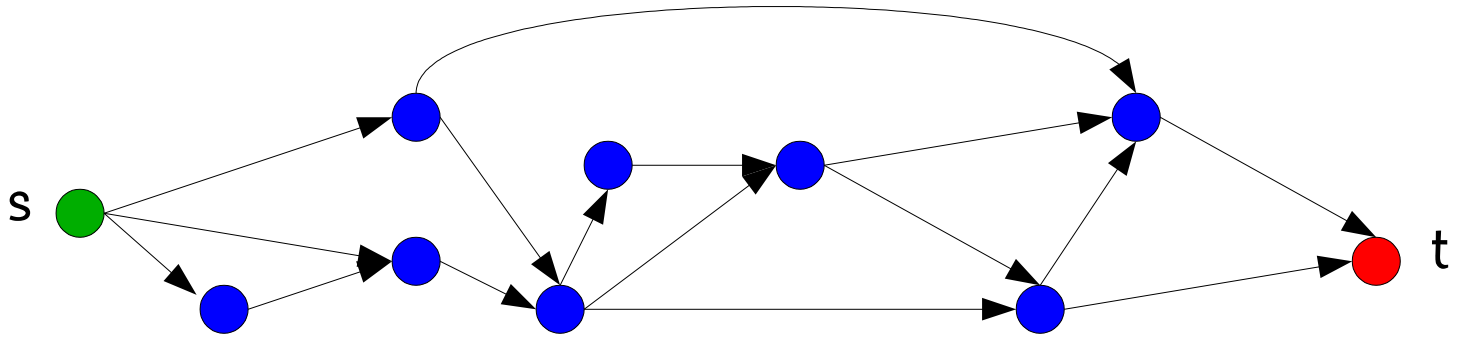
- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.

OSPF routing

- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight w(a) assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.

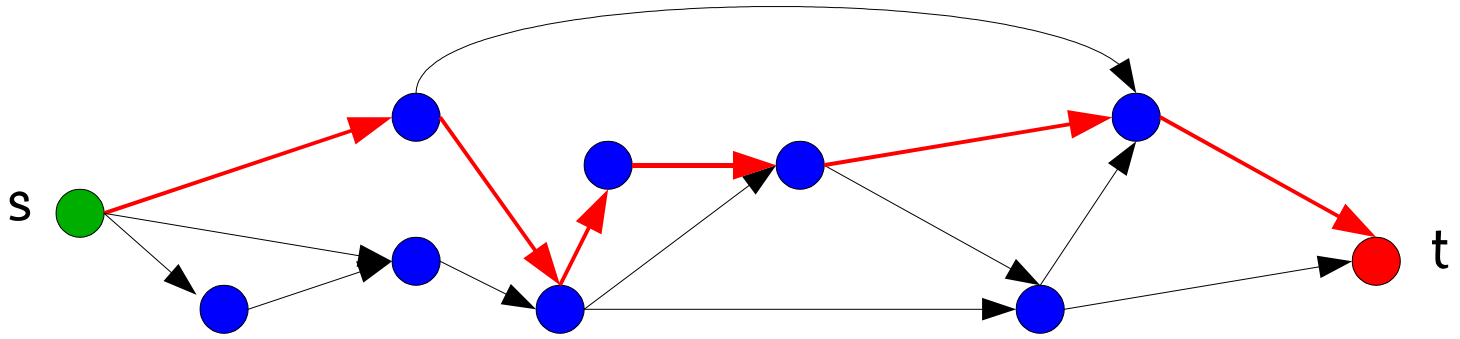
OSPF routing

- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight w(a) assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.



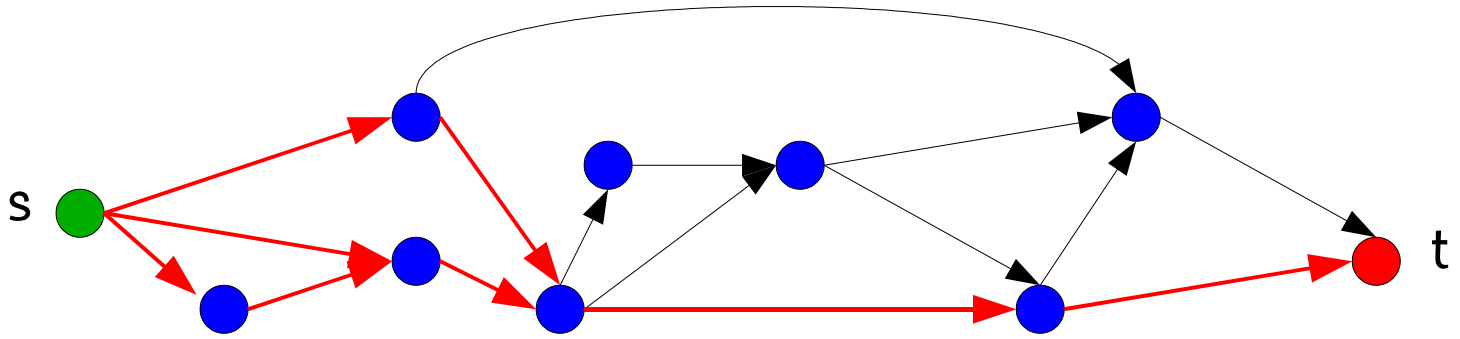
OSPF routing

- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight w(a) assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.



OSPF routing

- Given a network $G = (N,A)$, where N is the set of routers and A is the set of links.
- The OSPF (open shortest path first) routing protocol assumes each link a has a weight w(a) assigned to it so that a packet from a source router s to a destination router t is routed on a shortest weight path from s to t.



Traffic splitting

at&t
Your world. Delivered.

OSPF routing

- By setting OSPF weights appropriately, one can do traffic engineering, i.e. route traffic so as to optimize some objective (e.g. minimize congestion, maximize throughput, etc.).
- Some recent papers on this topic:
  - Fortz & Thorup (2000, 2004)
  - Ramakrishnan & Rodrigues (2001)
  - Sridharan, Guérin, & Diot (2002)
  - Fortz, Rexford, & Thorup (2002)
  - Ericsson, Resende, & Pardalos (2002)
  - Buriol, Resende, Ribeiro, & Thorup (2002, 2005)

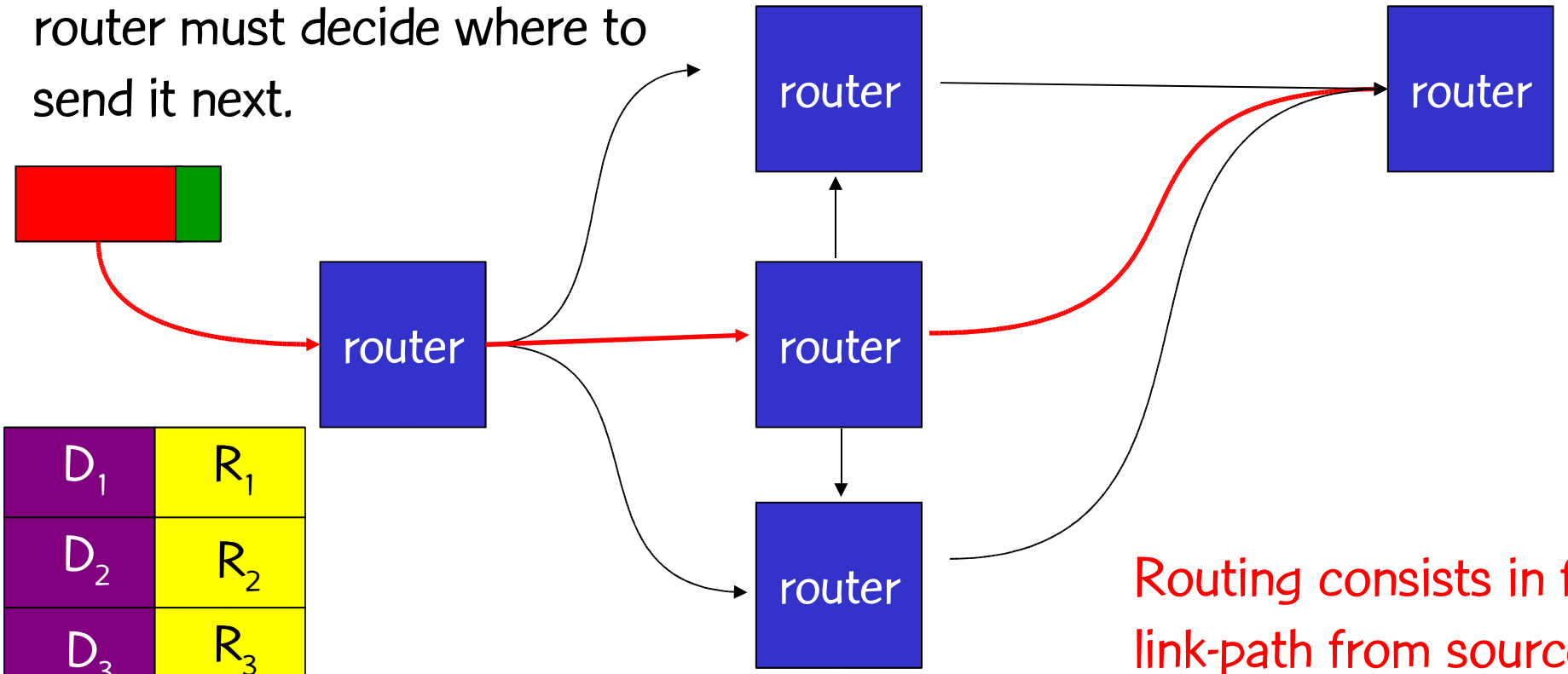
OSPF routing

- By setting OSPF weights appropriately, one can do traffic engineering, i.e. route traffic so as to optimize some objective (e.g. minimize congestion, maximize throughput, etc.).
- Some recent papers on this topic:
  - Fortz & Thorup (2000, 2004)
  - Ramakrishnan & Rodrigues (2001)
  - Sridharan, Guérin, & Diot (2002)
  - Fortz, Rexford, & Thorup (2002)
  - Ericsson, Resende, & Pardalos (2002)
  - Buriol, Resende, Ribeiro, & Thorup (2002, 2005)

Packet routing

When packet arrives at router, router must decide where to send it next.

Packet's final destination.



D_1	R_1
D_2	R_2
D_3	R_3
D_4	R_4

Routing table

Routing consists in finding a link-path from source to destination.

OSPF routing

- Assign an integer weight $\in [1, w_{max}]$ to each link in AS.

  In general, $w_{max} = 65535 = 2^{16} - 1$.

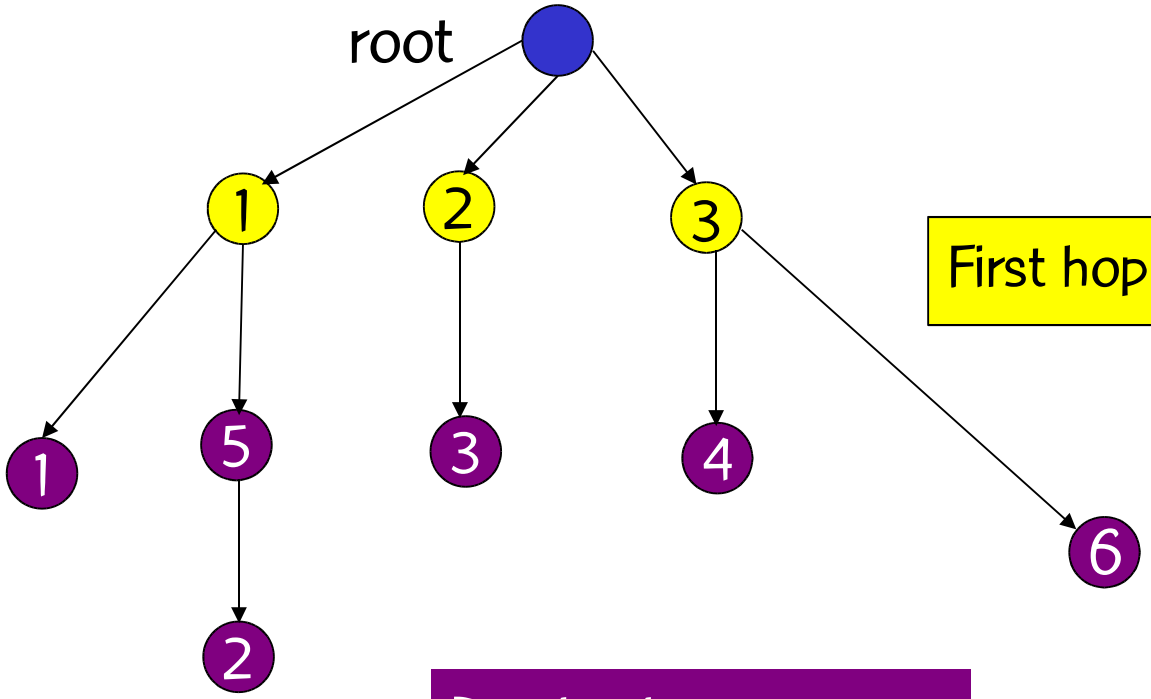
- Each router computes tree of shortest weight paths to all other routers in the AS, with itself as the root, using Dijkstra's algorithm.

OSPF routing

Routing table is filled
with first hop routers
for each possible destination.

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3



First hop routers.

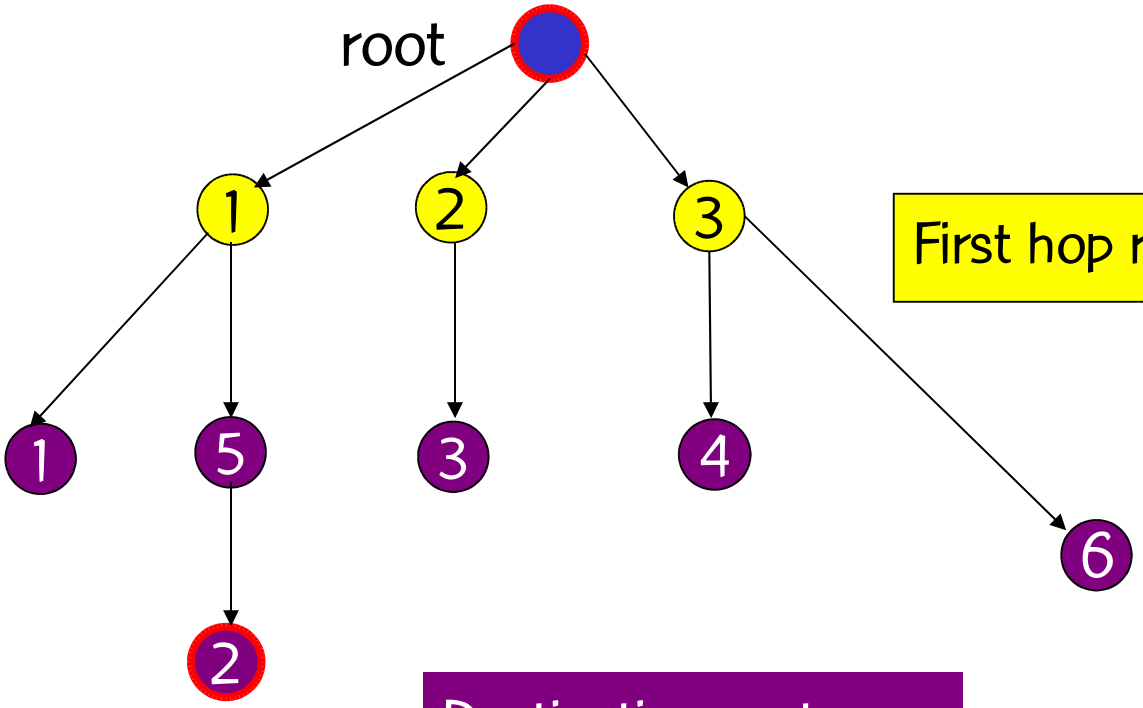
Destination routers

OSPF routing

Routing table is filled
with first hop routers
for each possible destination.

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3



First hop routers.

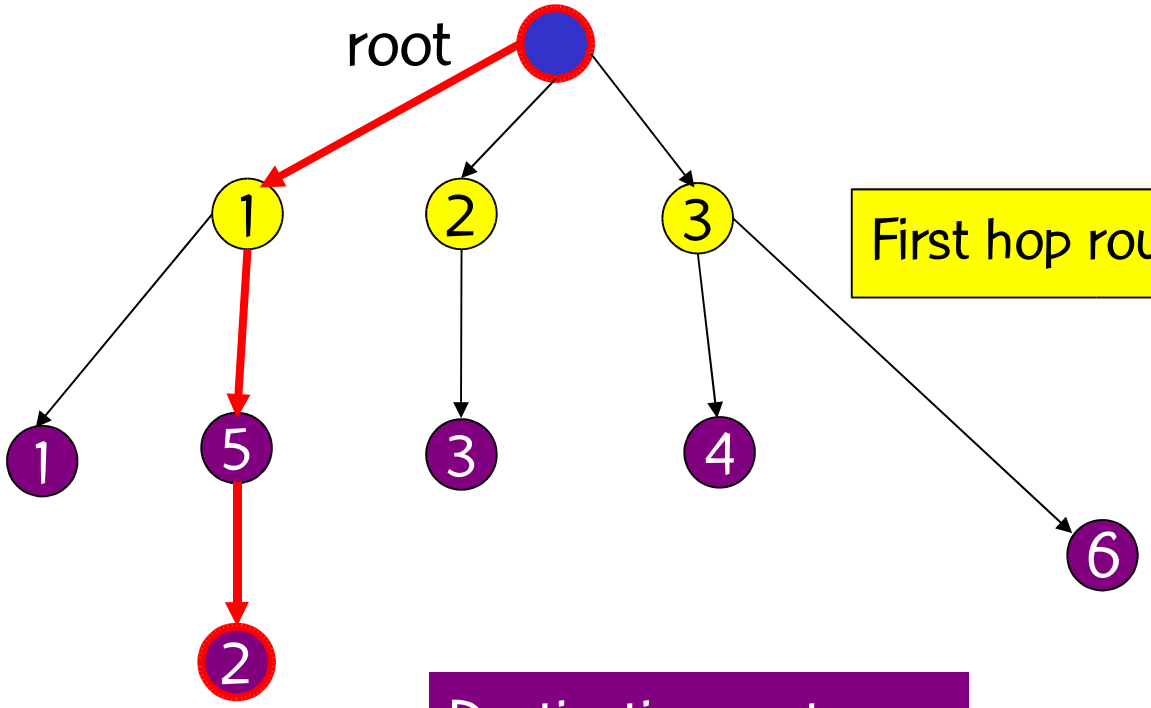
Destination routers

OSPF routing

Routing table is filled
with first hop routers
for each possible destination.

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3



First hop routers.

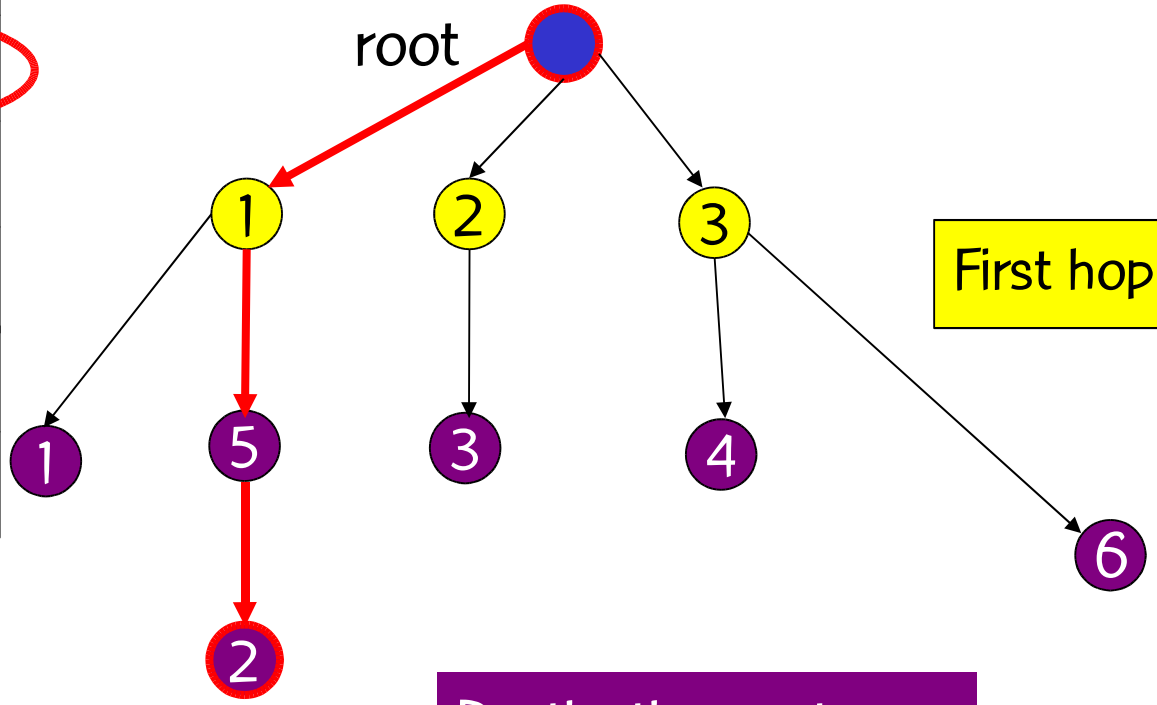
Destination routers

OSPF routing

Routing table is filled
with first hop routers
for each possible destination.

Routing table

D_1	R_1
D_2	R_1
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3



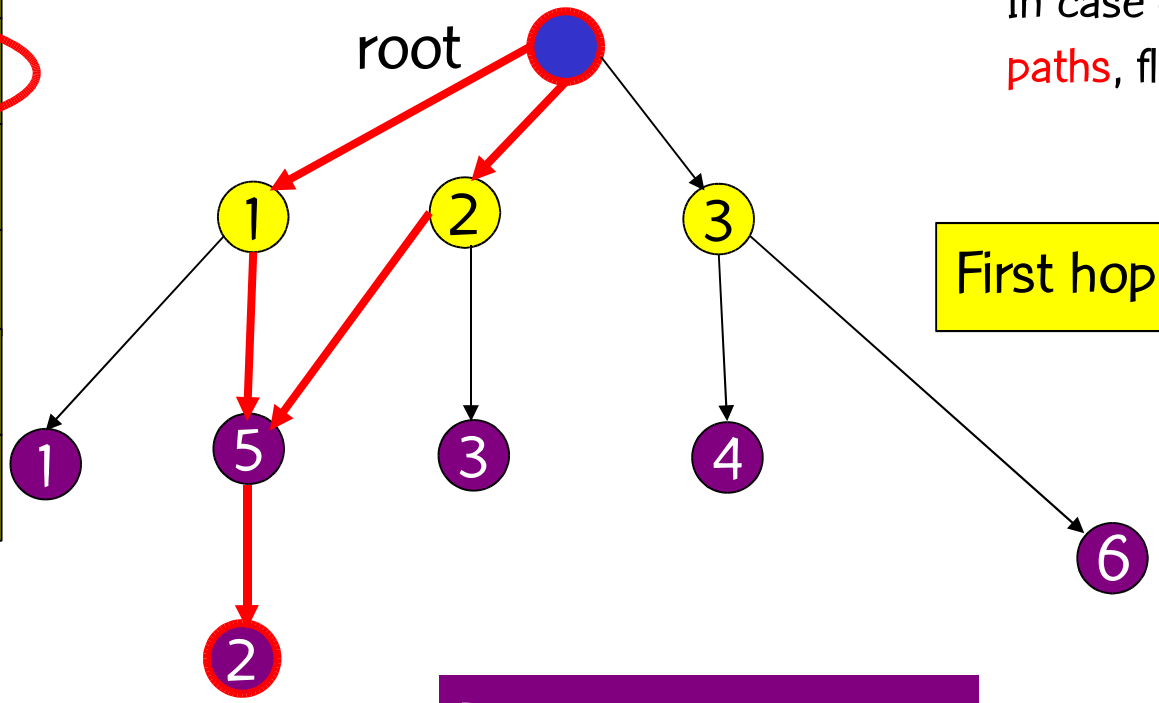
First hop routers.

Destination routers

OSPF routing

Routing table

D_1	R_1
D_2	R_1, R_2
D_3	R_2
D_4	R_3
D_5	R_1
D_6	R_3



Routing table is filled with first hop routers for each possible destination. In case of multiple shortest paths, flow is evenly split.

First hop routers.

Destination routers

OSPF weight setting

- OSPF weights are assigned by network operator.
  - CISCO assigns, by default, a weight proportional to the inverse of the link bandwidth (Inv Cap).
  - If all weights are unit, the weight of a path is the number of hops in the path.
- We propose evolutionary algorithms to find good OSPF weights.
  - Genetic algorithm
  - Memetic algorithm: Genetic algorithm with optimized crossover

Minimization of congestion

- Consider the directed capacitated network $G = (N,A,c)$, where N are routers, A are links, and c_a is the capacity of link $a \in A$.
- We use the measure of Fortz & Thorup (2000) to compute congestion:

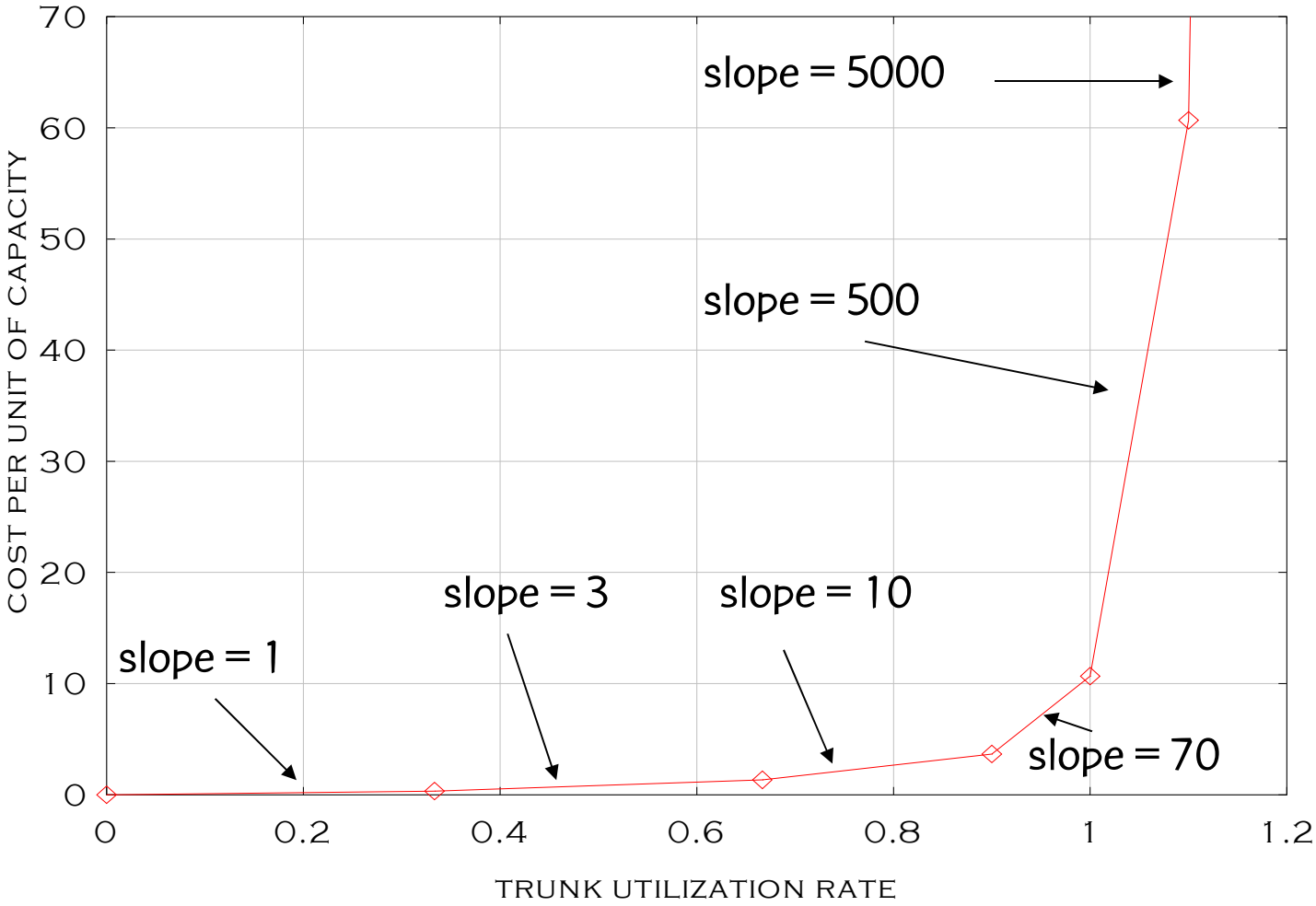
$$\Phi = \Phi_1(l_1) + \Phi_2(l_2) + \ldots + \Phi_{|A|}(l_{|A|})$$

where l_a is the load on link $a \in A$,

$\Phi_a(l_a)$ is piecewise linear and convex,

$\Phi_a(0) = 0$, for all $a \in A$.

Piecewise linear and convex $\Phi_a(l_a)$ link congestion measure



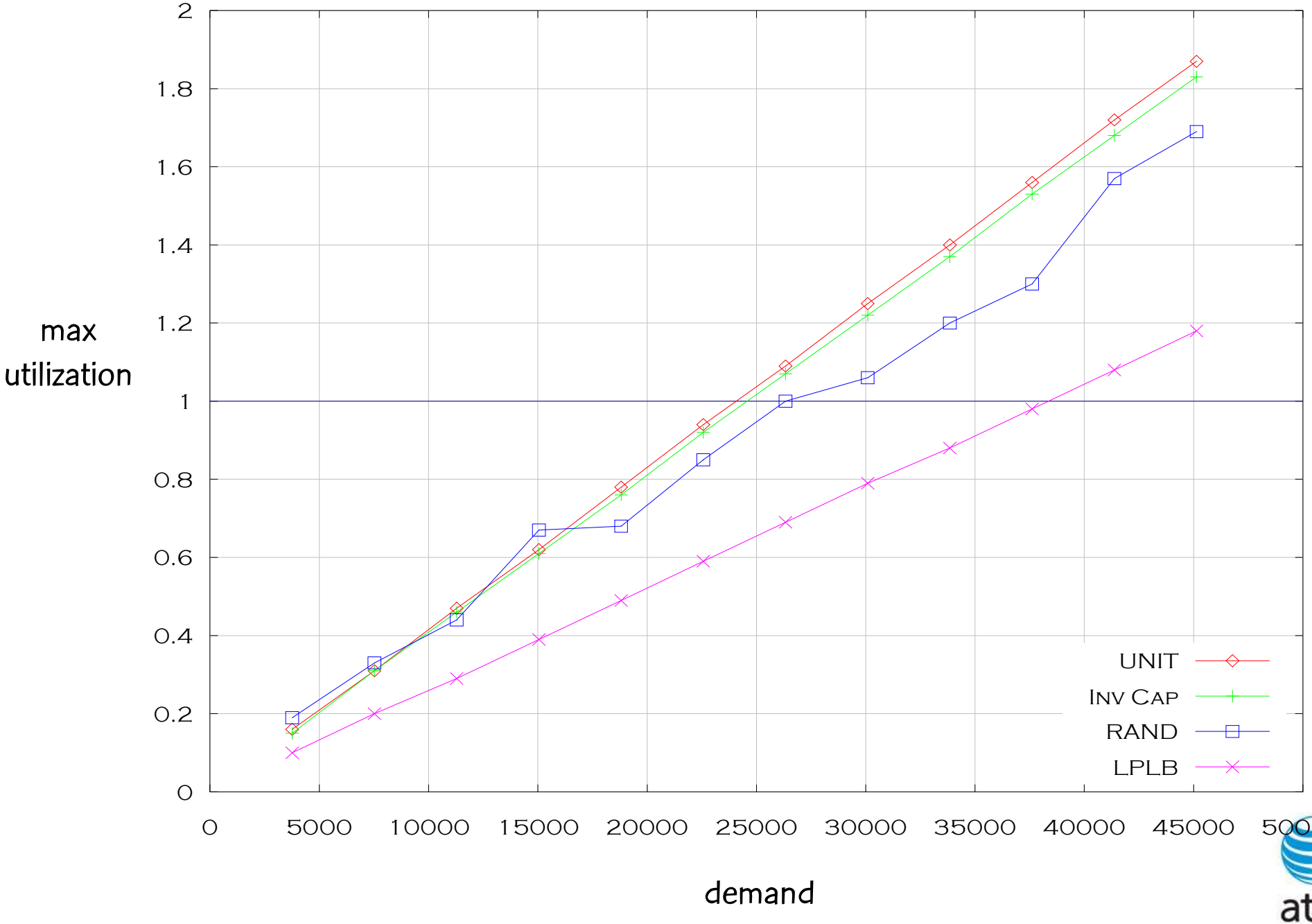
$(l_a \mid c_a)$

at&t
Your world. Delivered.

OSPF weight setting problem

- Given a directed network $G = (N, A)$ with link capacities $c_a \in A$ and demand matrix $D = (d_{s,t})$ specifying a demand to be sent from node s to node t :
  - Assign weights $w_a \in [1, w_{max}]$ to each link $a \in A$, such that the objective function Φ is minimized when demand is routed according to the OSPF protocol.

AT&T Worldnet backbone network (90 routers, 274 links)



at&t
Your world. Delivered.

Genetic and hybrid genetic algorithms for OSPF weight setting problem

- Genetic
  - M. Ericsson, M.G.C. Resende, & P.M. Pardalos, " A genetic algorithm for the weight setting problem in OSPF routing, J. of Combinatorial Optimization, vol. 6, pp. 299-333, 2002.
- Hybrid genetic
  - L.S. Buriol, M.G.C. Resende, C.C. Ribeiro, & M. Thorup, "A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing," Networks, vol. 46, pp. 36-56, 2005.

Solution encoding

- A population consists of nPop = 50 integer weight arrays: $w = (w_1, w_2, \ldots, w_{|A|})$,

  where $w_a \in [1, w_{max} = 20]$

- All possible weight arrays correspond to feasible solutions.

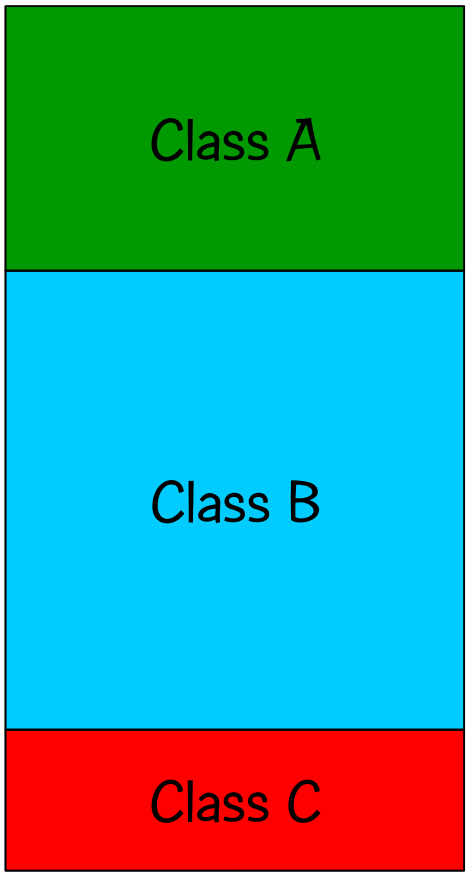
Initial population

- nPop solutions, with each weight randomly generated, uniformly in the interval $[1, w_{max}/3]$.

Solution evaluation (fitness)

- For each demand pair (s,t), route using OSPF, computing demand pair loads $l_a^{s,t}$ on each link $a \in A$.
- Add up demand pair loads on each link $a \in A$, yielding total load l_a on link.
- Compute link congestion cost $\Phi_a(l_a)$ for each link $a \in A$.
- Add up costs: $\Phi = \Phi_1(l_1) + \Phi_2(l_2) + \ldots + \Phi_{|A|}(l_{|A|})$

Population partitioning



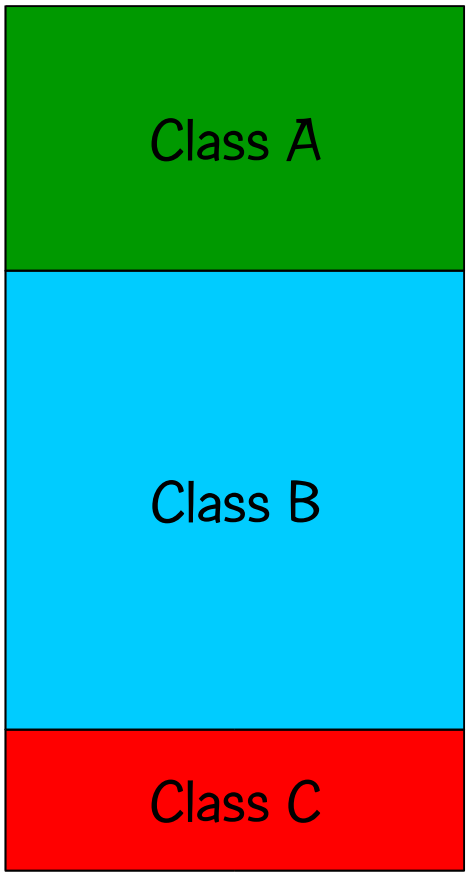
25% most fit

5% least fit

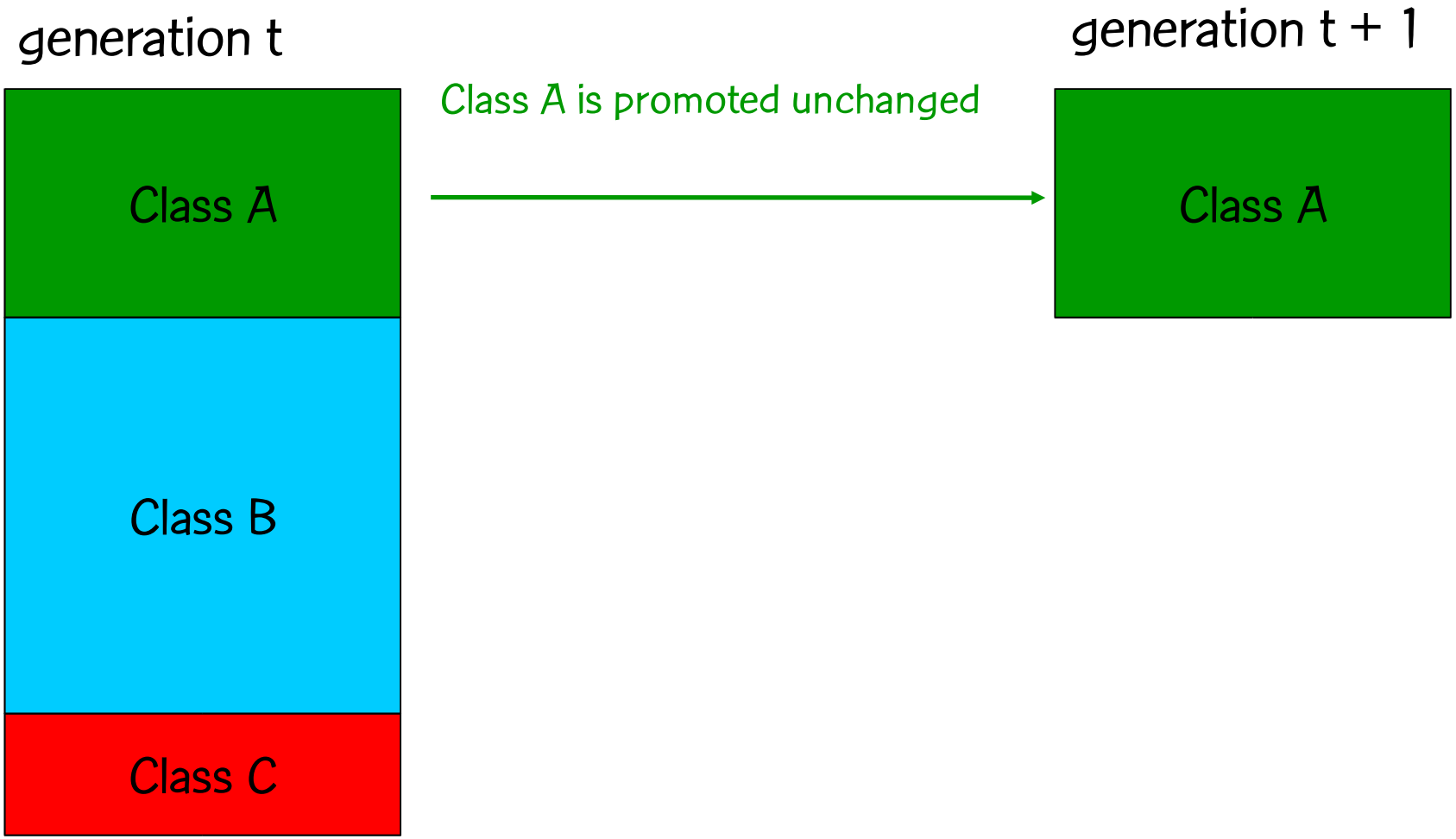
Population is sorted according to solution value Φ and solutions are classified into three categories.

Population dynamics

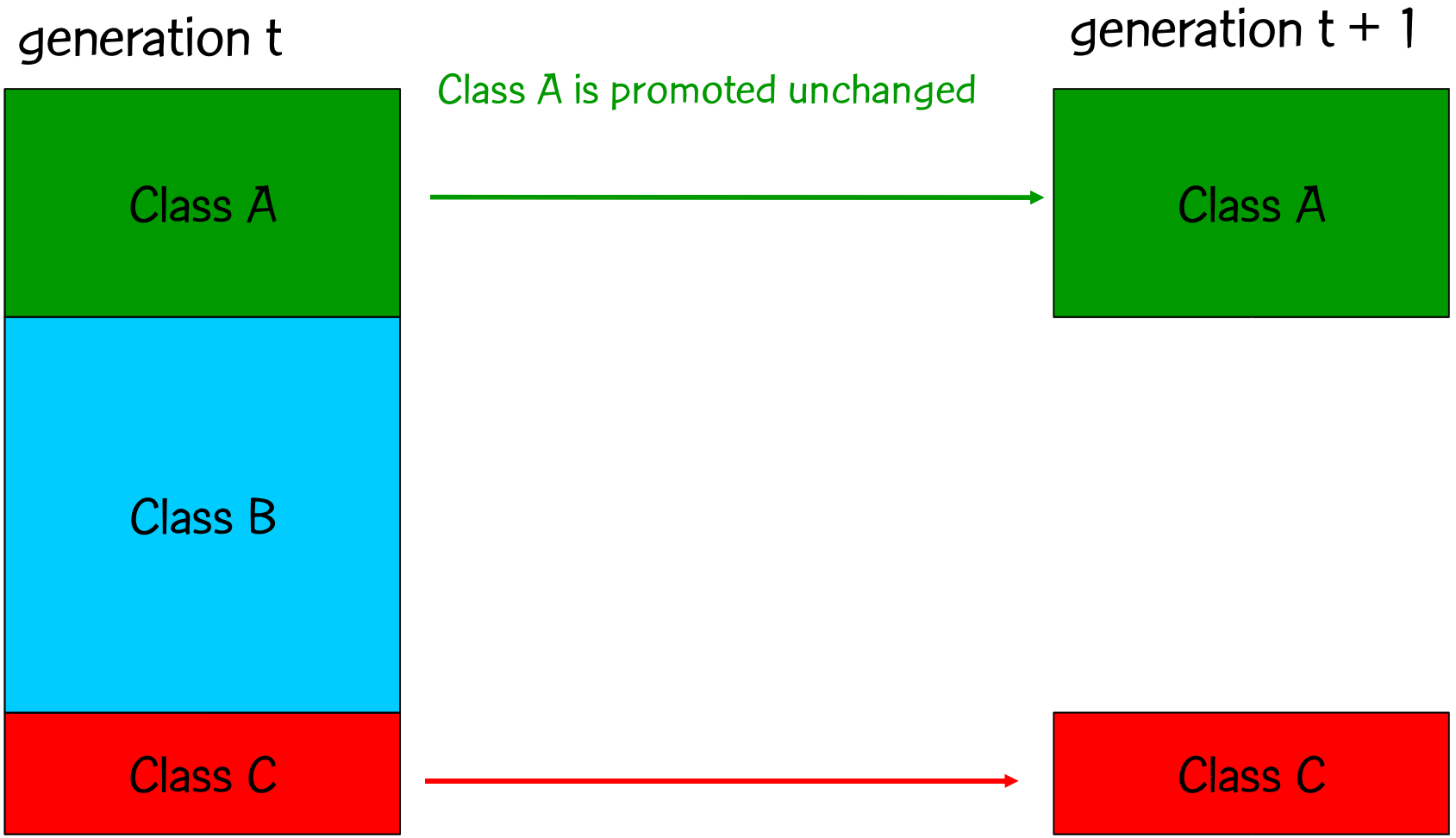
generation t



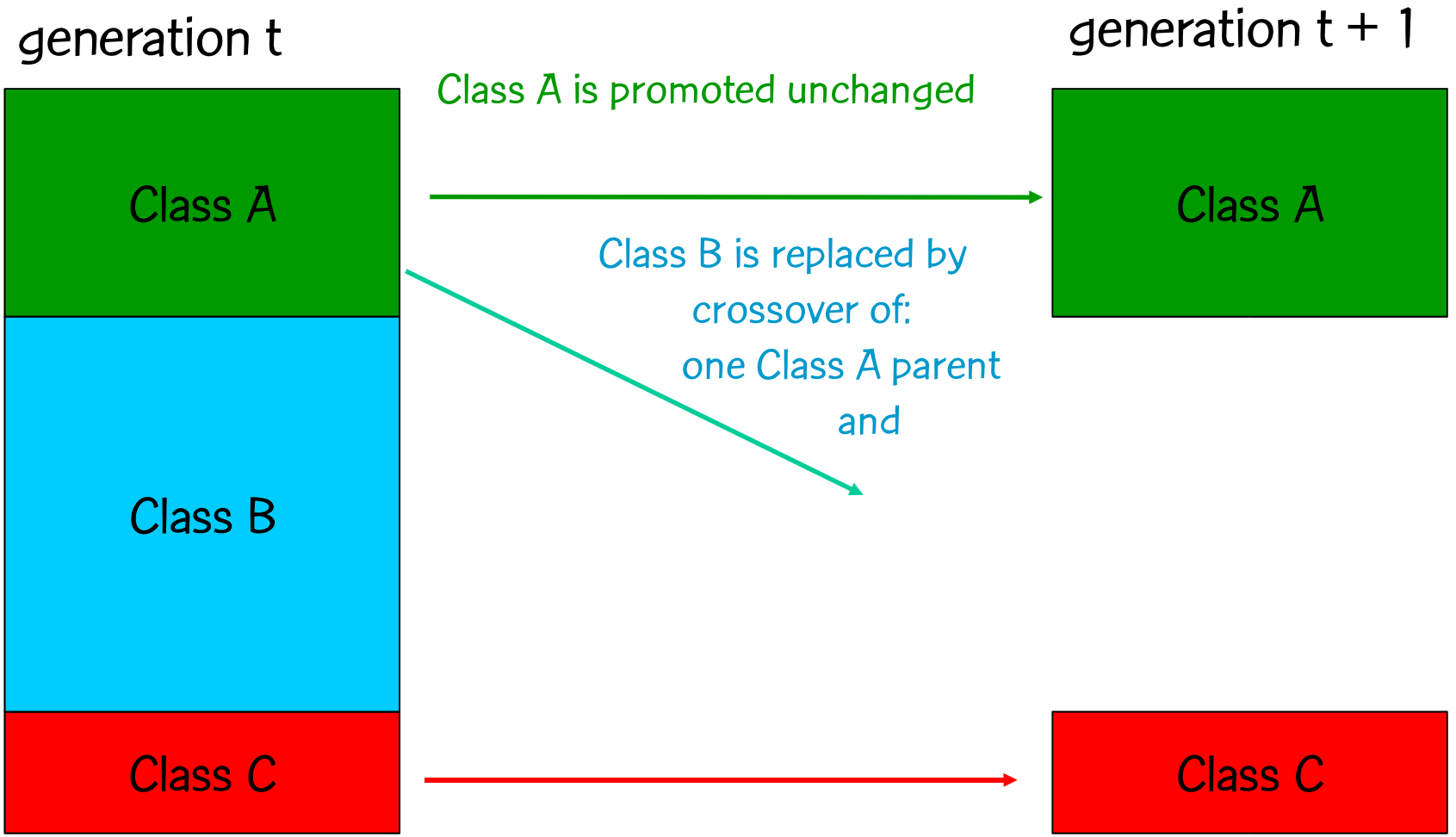
Population dynamics



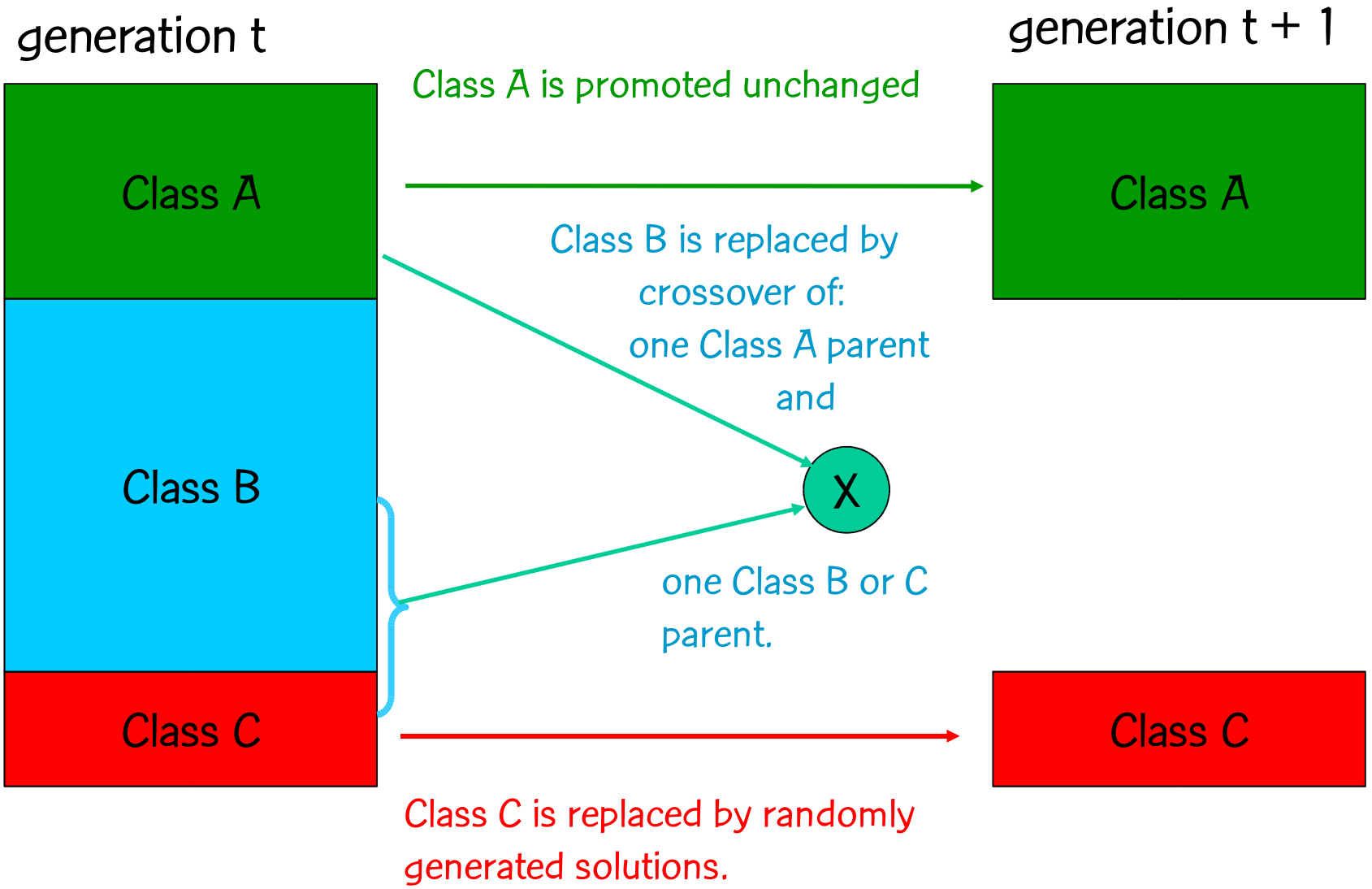
Population dynamics



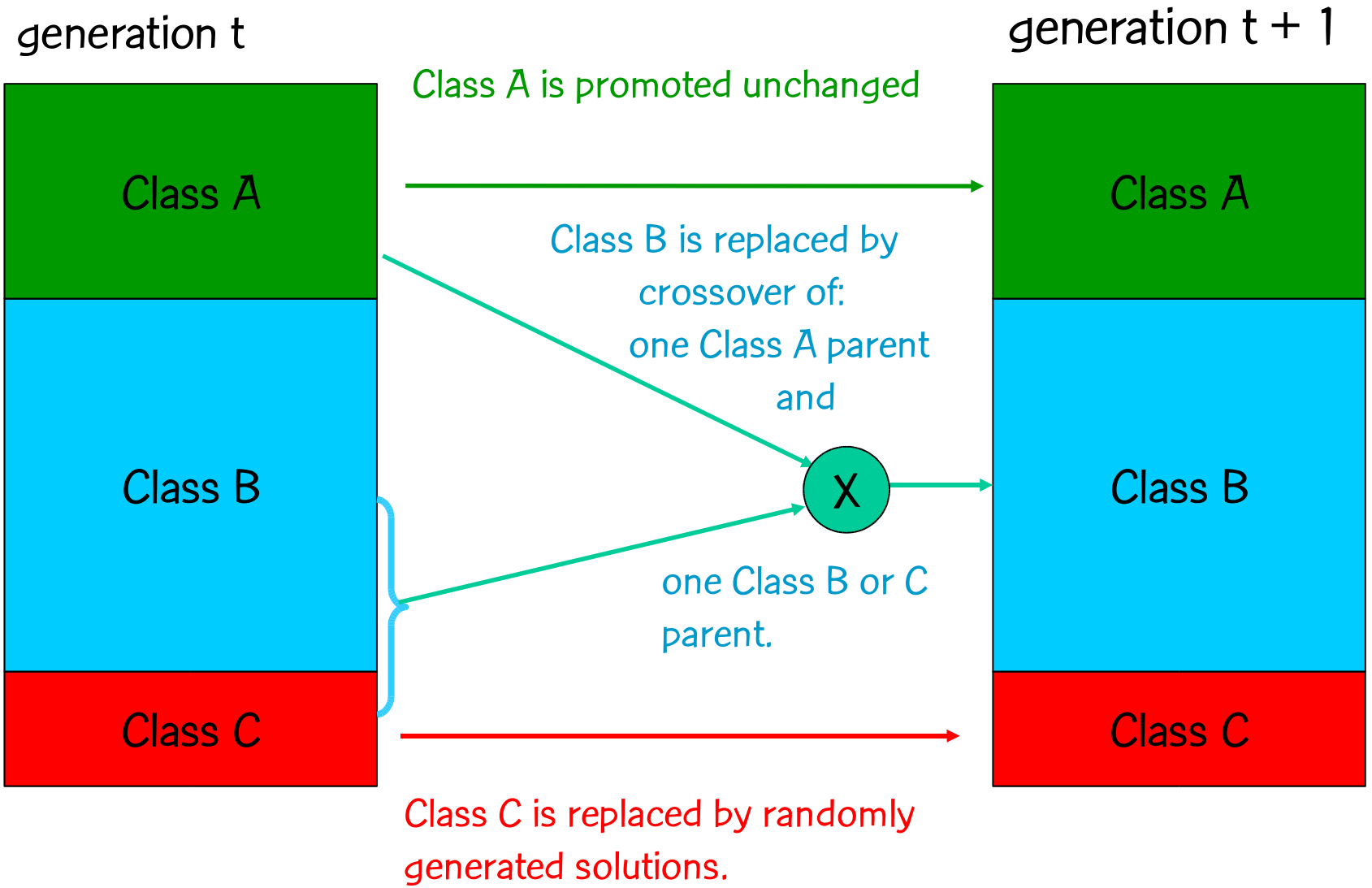
Population dynamics



Population dynamics



Population dynamics



Parent selection

- Parents are chosen at random:
  - one parent from Class A (elite).
  - one parent from Class B or C (non-elite).
- Re-selection is allowed, i.e. parents can breed more than once per generation.
- Better individuals are more likely to reproduce.

Crossover with random keys

Bean (1994)

Crossover combines elite parent p_1 with non-elite parent p_2 to produce child c :

for each link $a \in A$ do:

with probability 0.7: $c[a] = p_1[a]$

with probability 0.3: $c[a] = p_2[a]$

Crossover with random keys

Bean (1994)

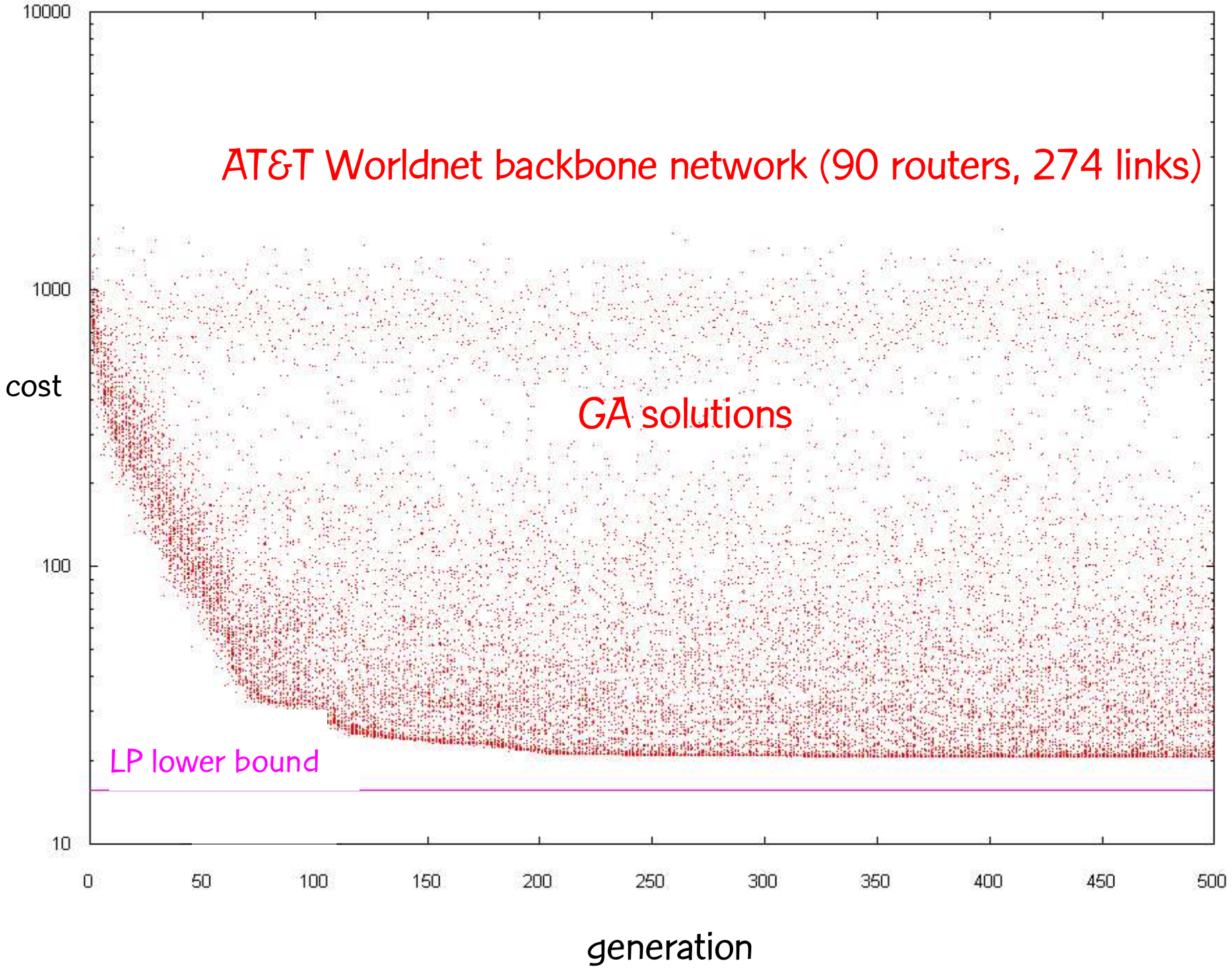
Crossover combines elite parent p_1 with non-elite parent p_2 to produce child c :

for each link $a \in A$ do:

with probability 0.7: $c[a] = p_1[a]$

with probability 0.3: $c[a] = p_2[a]$

Child is more likely to inherit gene of elite parent.

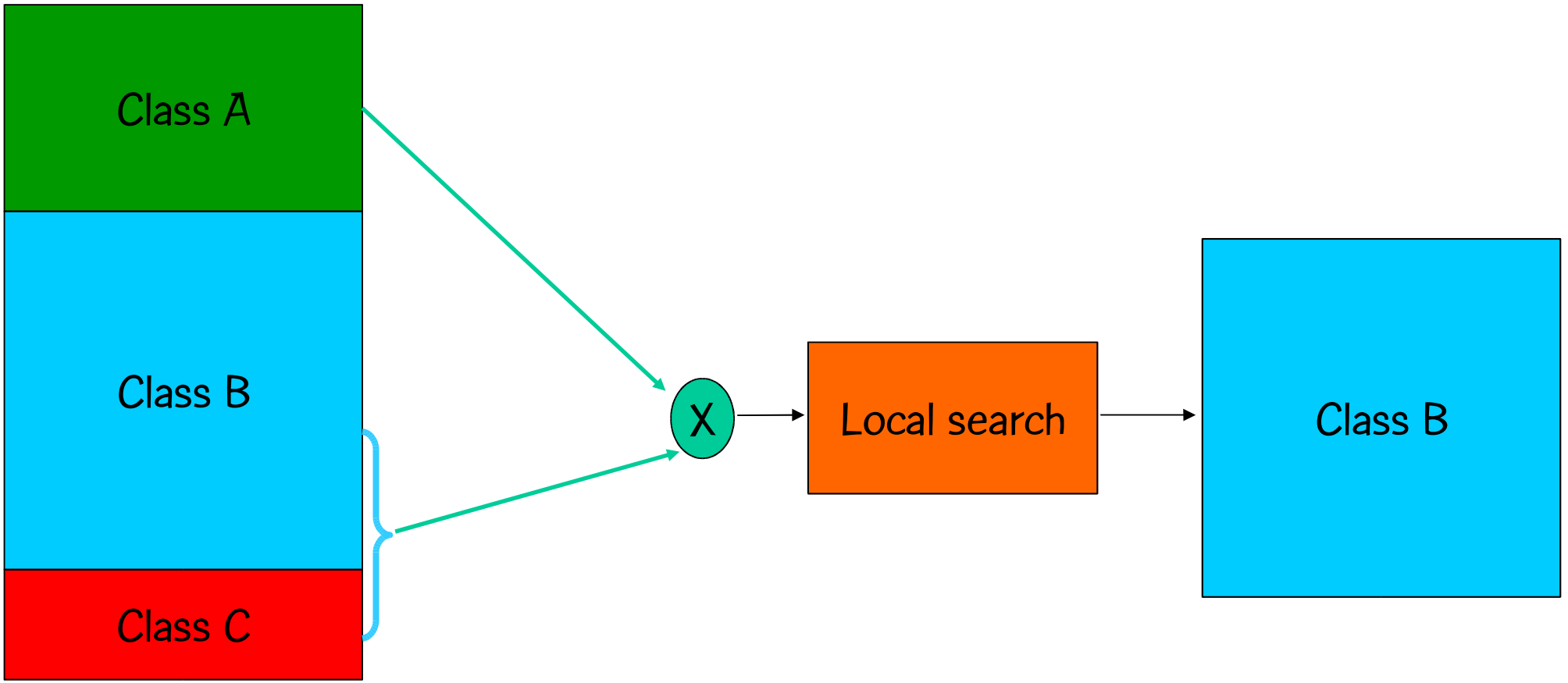


at&t
Your world. Delivered.

AT&T Worldnet backbone network (90 routers, 274 links)

at&t
Your world. Delivered.

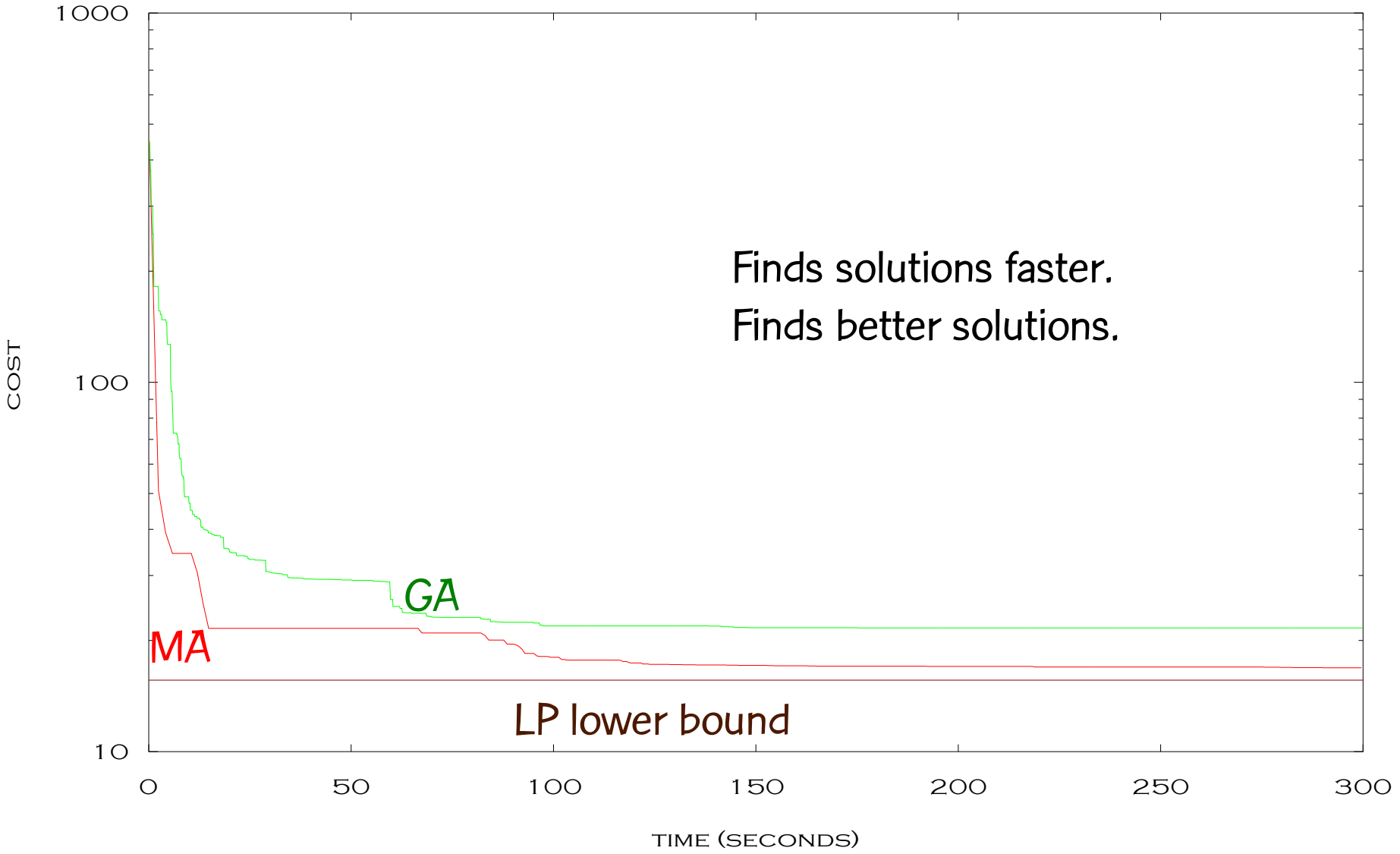
Optimized crossover = crossover + local search



Fast local search

- Let A^* be the set of five arcs $a \in A$ having largest Φ_a values.
- Scan arcs $a \in A^*$ from largest to smallest Φ_a:
  - Increase arc weight, one unit at a time, in the range $[w_a , w_a + \lceil (w_{max} - w_a)/4 \rceil]$
  - If total cost Φ is reduced, restart local search.

Memetic algorithm (MA) improves over pure genetic algorithm in two ways:



at&t
Your world. Delivered.

Survivable IP network design

Survivable IP network design

- Buriol, Resende, and Thorup (Networks, 2006) use weight setting to design survivable IP networks.

Survivable IP network design

- Buriol, Resende, and Thorup (Networks, 2006) use weight setting to design survivable IP networks.
- Given
  - directed graph $G = (N,A)$, where N is the set of routers, A is the set of potential arcs where capacity can be installed,
  - a demand matrix D that for each pair $(s,t) \in N \times N$, specifies the demand D(s,t) between s and t,
  - a cost K(a) to lay fiber on arc a
  - a capacity increment C for the fiber.

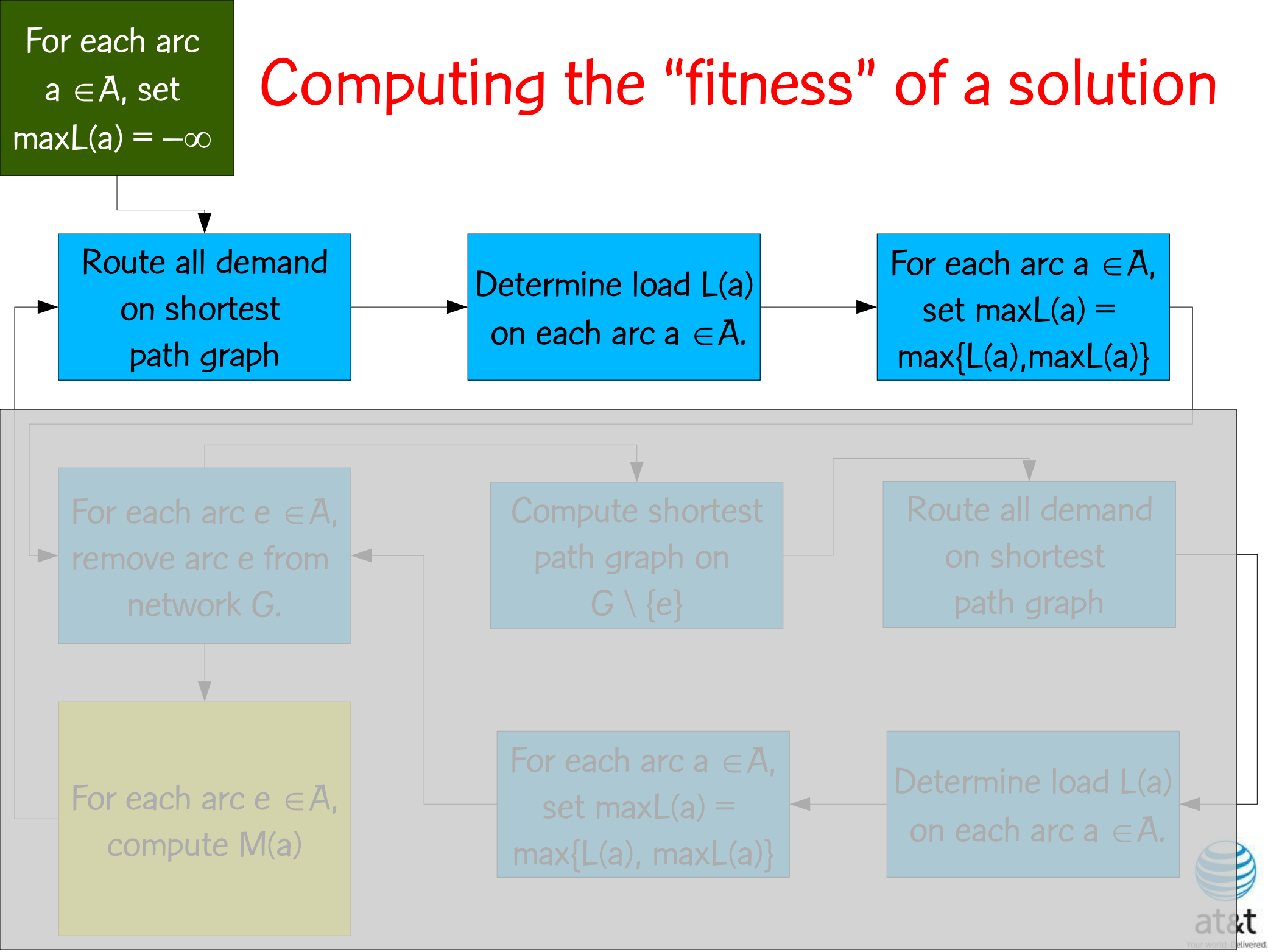
Survivable IP network design

- Buriol, Resende, and Thorup (Networks, 2006) use weight setting to design survivable IP networks.
- Determine
  - OSPF weight w(a) to assign to each arc $a \in A$,
  - which arcs should be used to deploy fiber and how many units (multiplicities) M(a) of capacity C should be installed on each arc $a \in A$,
- such that all the demand can be routed on the network even when any single arc fails.
- Minimize total design cost = $\sum_{a \in A} M(a) \times K(a)$.

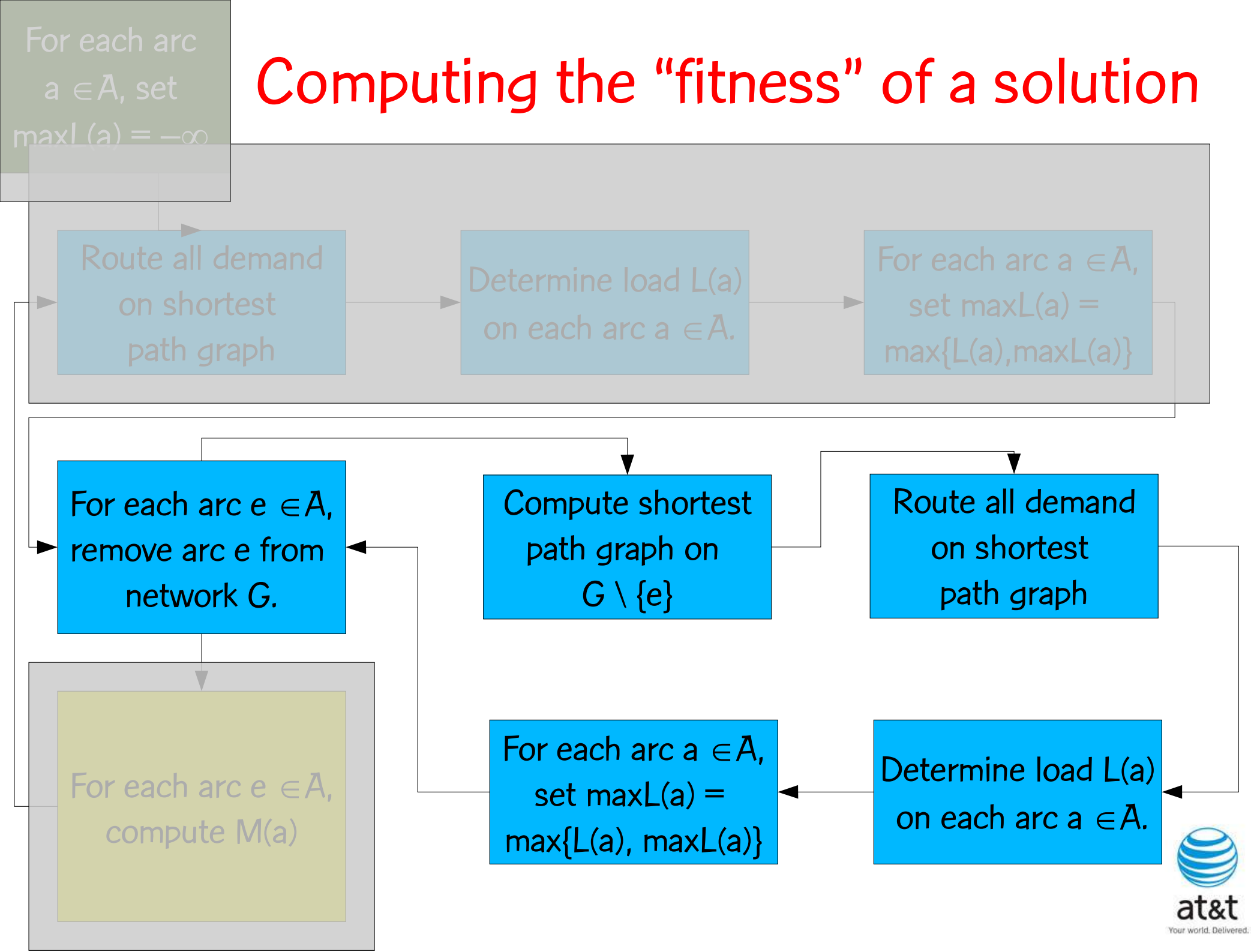
Survivable IP network design

- Buriol, Resende, and Thorup (Networks, 2006) use weight setting to design survivable IP networks.
- Use genetic algorithm (GA) to determine weights.
- GA needs to compute "fitness" of each solution it produces.

Computing the "fitness" of a solution

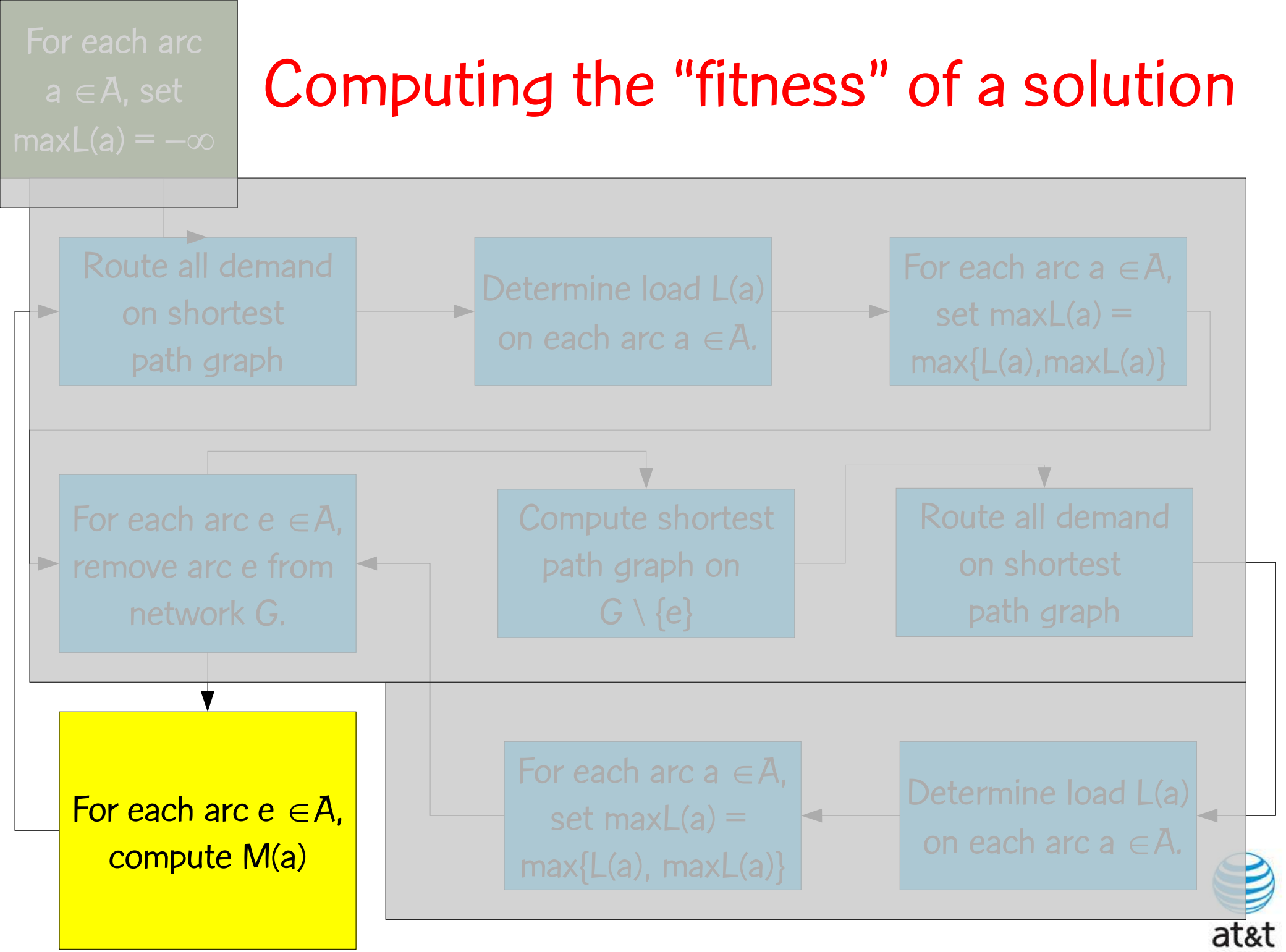


Computing the "fitness" of a solution



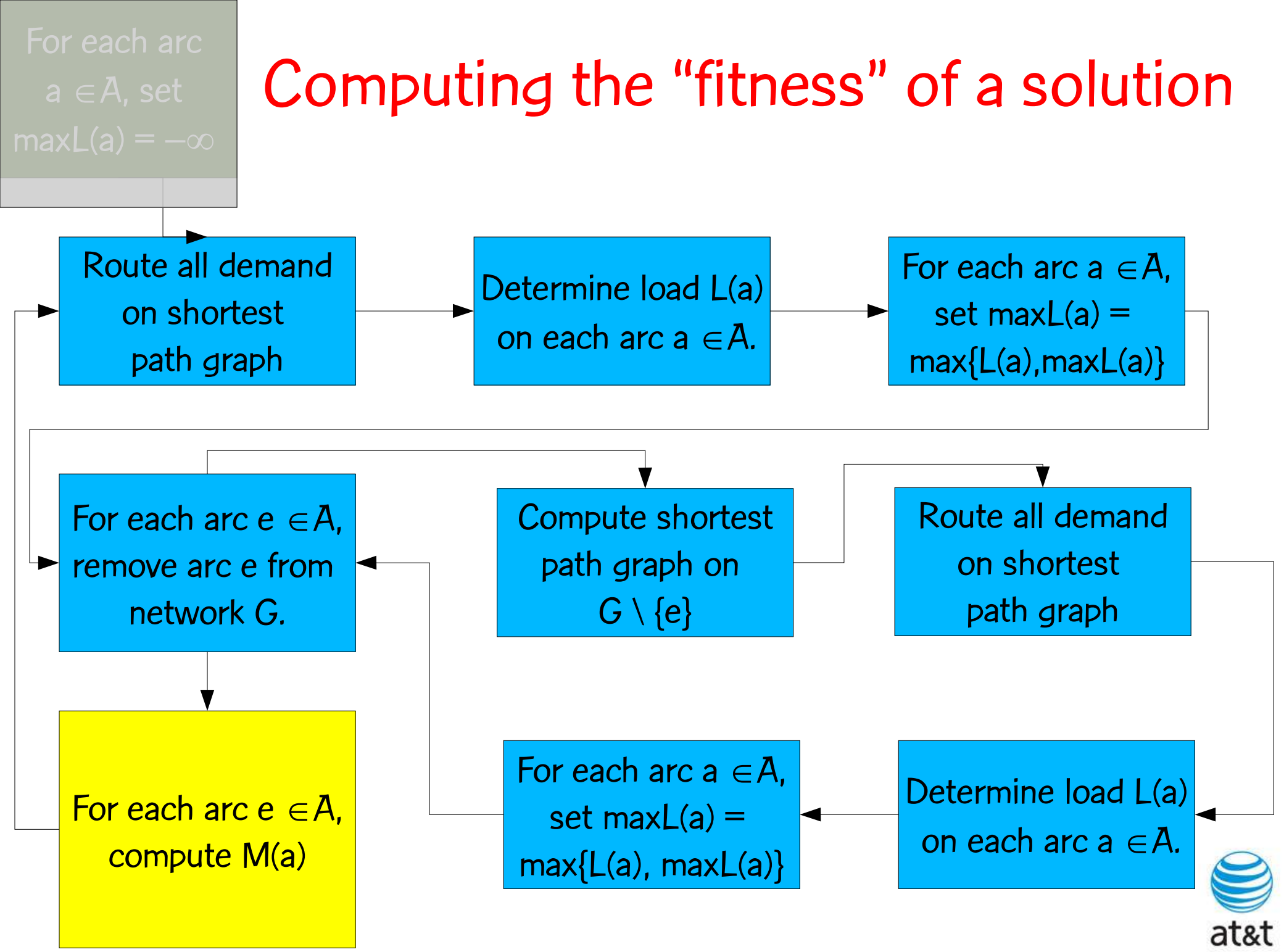
at&t
Your world. Delivered.

Computing the "fitness" of a solution



at&t
Your world. Delivered.

Computing the "fitness" of a solution



at&t
Your world. Delivered.

Composite-link design

- In Buriol, Resende, and Thorup (2006)
  - links were all of the same type,
  - only the link multiplicity had to be determined.
- Now consider composite links. Given a load L(a) on arc a, we can compose several different link types that sum up to the needed capacity $c(a) \geq L(a)$:
  - $c(a) = \sum_{t \text{ used in arc } a} M(t) \times \gamma(t)$, where
  - M(t) is the multiplicity of link type t
  - $\gamma(t)$ is the capacity of link type t

Composite-link design

- In Buriol, Resende, and Thorup (2006)
  - links were all of the same type,
  - only the link multiplicity had to be determined.
- Now consider composite links. Given a load L(a) on arc a, we can compose several different link types that sum up to the needed capacity $c(a) \geq L(a)$:
  - $c(a) = \sum_{t \text{ used in arc } a} M(t) \times \gamma(t)$, where
  - $M(t)$ is the multiplicity of link type t
  - $\gamma(t)$ is the capacity of link type t

Composite-link design

- Link types = { 1, 2, ..., T }
- Capacities = { $c(1), c(2), ..., c(T)$ } : $c(i) < c(i+1)$
- Prices / unit length = { $p(1), p(2), ..., p(T)$ }: $p(i) < p(i+1)$
- Assumptions:
  - $[p(T)/c(T)] < [p(T-1)/c(T-1)] < \cdots < [p(1)/c(1)]$, i.e. price per unit of capacity is smaller for links with greater capacity
  - $c(i) = \alpha \times c(i-1)$, for $\alpha \in N$, $\alpha > 1$, i.e. capacities are multiples of each other by powers of α

Composite-link design

- Link types = { 1, 2, …, T }
- Capacities = { $c(1), c(2), \ldots, c(T)$ } : $c(i) < c(i+1)$
- Prices / unit length = { $p(1), p(2), \ldots, p(T)$ }: $p(i) < p(i+1)$
- Assumptions:
  - $[p(T)/c(T)] < [p(T-1)/c(T-1)] < \cdots < [p(1)/c(1)]$, i.e. price per unit of capacity is smaller for links with greater capacity
  - $c(i) = \alpha \times c(i-1)$, for $\alpha \in N$, $\alpha > 1$, i.e. capacities are multiples of each other by powers of α

Composite-link design

- Link types = { 1, 2, ..., T }
- Capacities = { c(1), c(2), ..., c(T) } : c(i) < c(i+1)
- Prices / unit length = { p(1), p(2), ..., p(T) }: p(i) < p(i+1)
- Assumptions:
  - [p(T)/c(T)] < [p(T−1)/c(T−1)] < ··· < [p(1)/c(1)]: economies of scale
  - c(i) = α × c(i−1), for $\alpha \in N$, $\alpha > 1$,
    e.g. c(OC192) = 4 × c(OC48); c(OC48) = 4 × c(OC12); c(OC12) = 4 × c(OC3);

OC3	OC12	OC48	OC192	
155 Mb/s	622 Mb/s	2.5 Gb/s	10 Gb/s	$\alpha = 4$

Heuristics for composite-link design

- Designed to minimize overall network cost.
- Heuristics are:
  - Min capacity
  - Min cost
  - Min cost k types
  - Min multiplicities

Heuristics for composite-link design

- Designed to minimize overall network cost.
- Heuristics are:
  - Min capacity
  - Min cost
  - Min cost k types
  - Min multiplicities

Heuristics for composite-link design

- Designed to minimize overall network cost.
- Heuristics are:
  - Min capacity
  - Min cost
  - Min cost k types
  - Min multiplicities

Heuristics for composite-link design

- Designed to minimize overall network cost.
- Heuristics are:
  - Min capacity
  - Min cost
  - Min cost k types
  - Min multiplicities

Heuristics for composite-link design

- Designed to minimize overall network cost.
- Heuristics are:
  - Min capacity
  - Min cost
  - Min cost k types
  - Min multiplicities

Heuristics for composite-link design

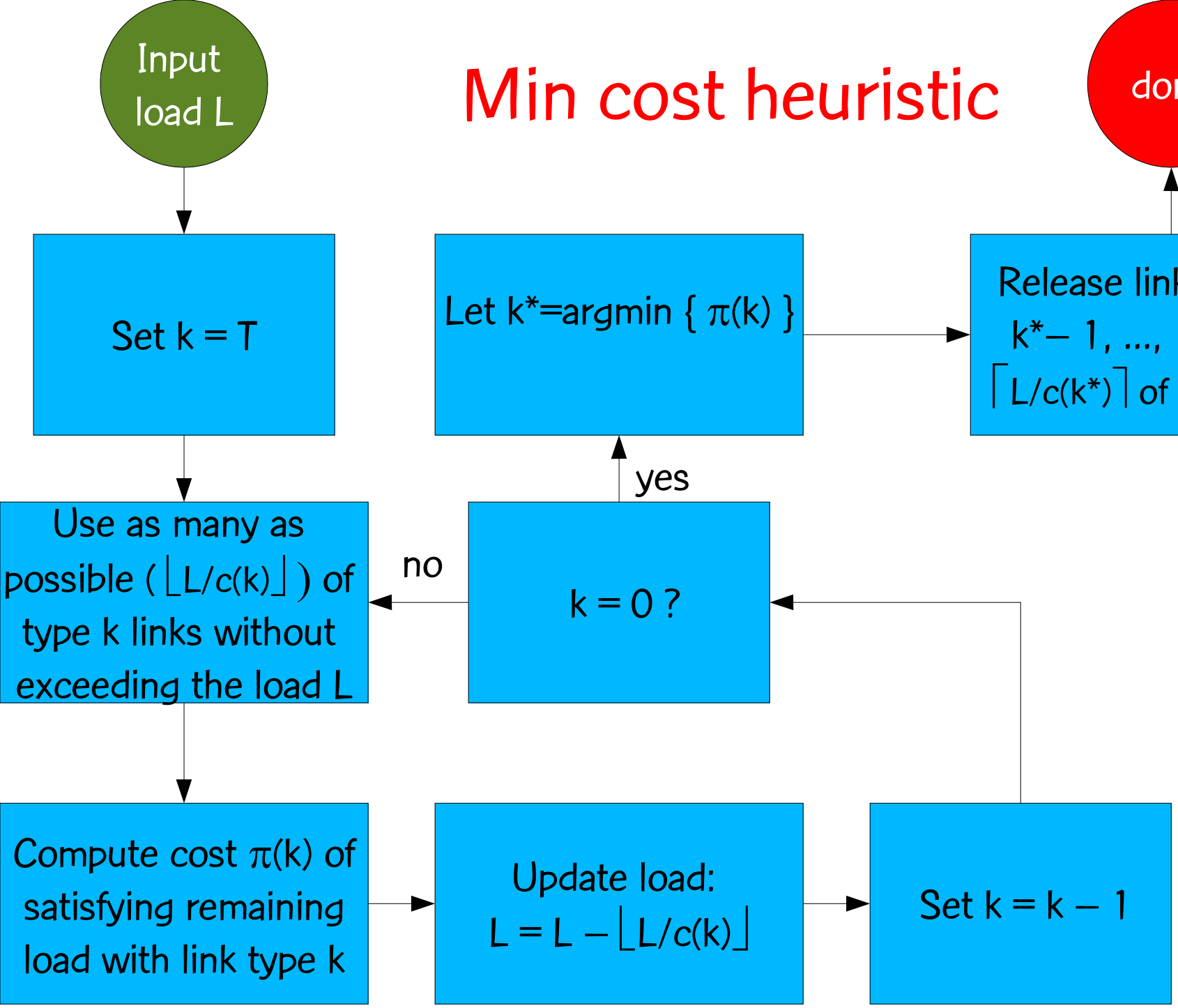
- Designed to minimize overall network cost.
- Heuristics are:
  - Min capacity
  - Min cost
  - Min cost k types
  - Min multiplicities

Heuristics for composite-link design

- Designed to minimize overall network cost.
- Heuristics are:
  - Min capacity
  - Min cost
  - Min cost k types
  - Min multiplicities

We describe this heuristic next.

Min cost heuristic



π(k) is total cost of using links of types T, T– 1, ..., k.

at&t
Your world. Delivered.

Observations

- Subject to the assumptions listed earlier, all heuristics (except min cost $k > 1$ types) can be implemented to take $O(T)$ time to execute per arc.
- Min capacity gives optimal solution for the minimum capacity objective function.
- Min cost gives the optimal solution for the minimum cost objective function.
- Without the assumptions, a knapsack problem must be solved to find min cap and min cost solutions.

Observations

- Subject to the assumptions listed earlier, all heuristics (except min cost $k > 1$ types) can be implemented to take $O(T)$ time to execute per arc.
- Min capacity gives optimal solution for the minimum capacity objective function.
- Min cost gives the optimal solution for the minimum cost objective function.
- Without the assumptions, a knapsack problem must be solved to find min cap and min cost solutions.

Observations

- Subject to the assumptions listed earlier, all heuristics (except min cost $k > 1$ types) can be implemented to take $O(T)$ time to execute per arc.
- Min capacity gives optimal solution for the minimum capacity objective function.
- Min cost gives the optimal solution for the minimum cost objective function.
- Without the assumptions, a knapsack problem must be solved to find min cap and min cost solutions.

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, …, 1000 generations and costs were recorded for each heuristic.

Experimental results

- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

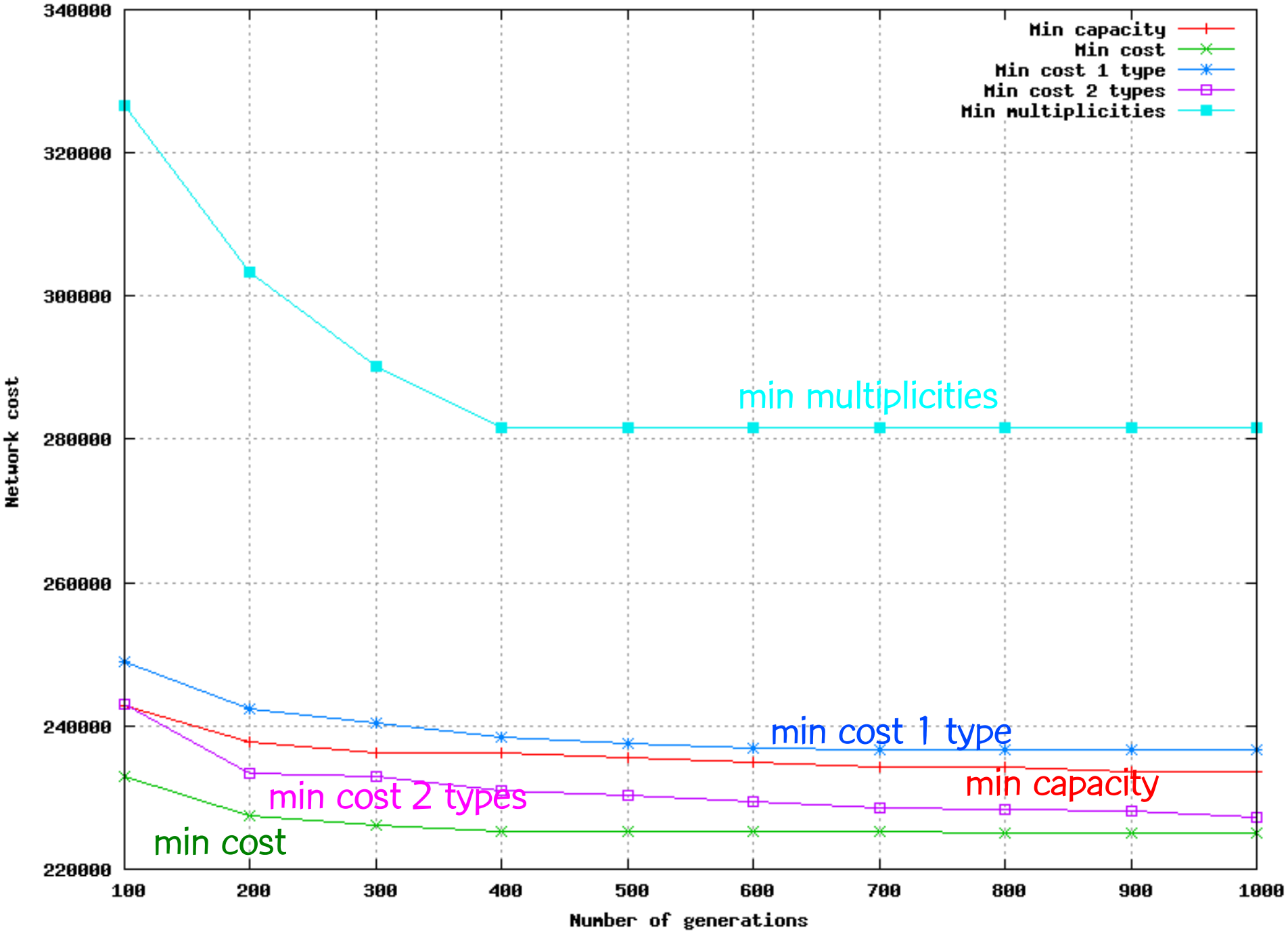
- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

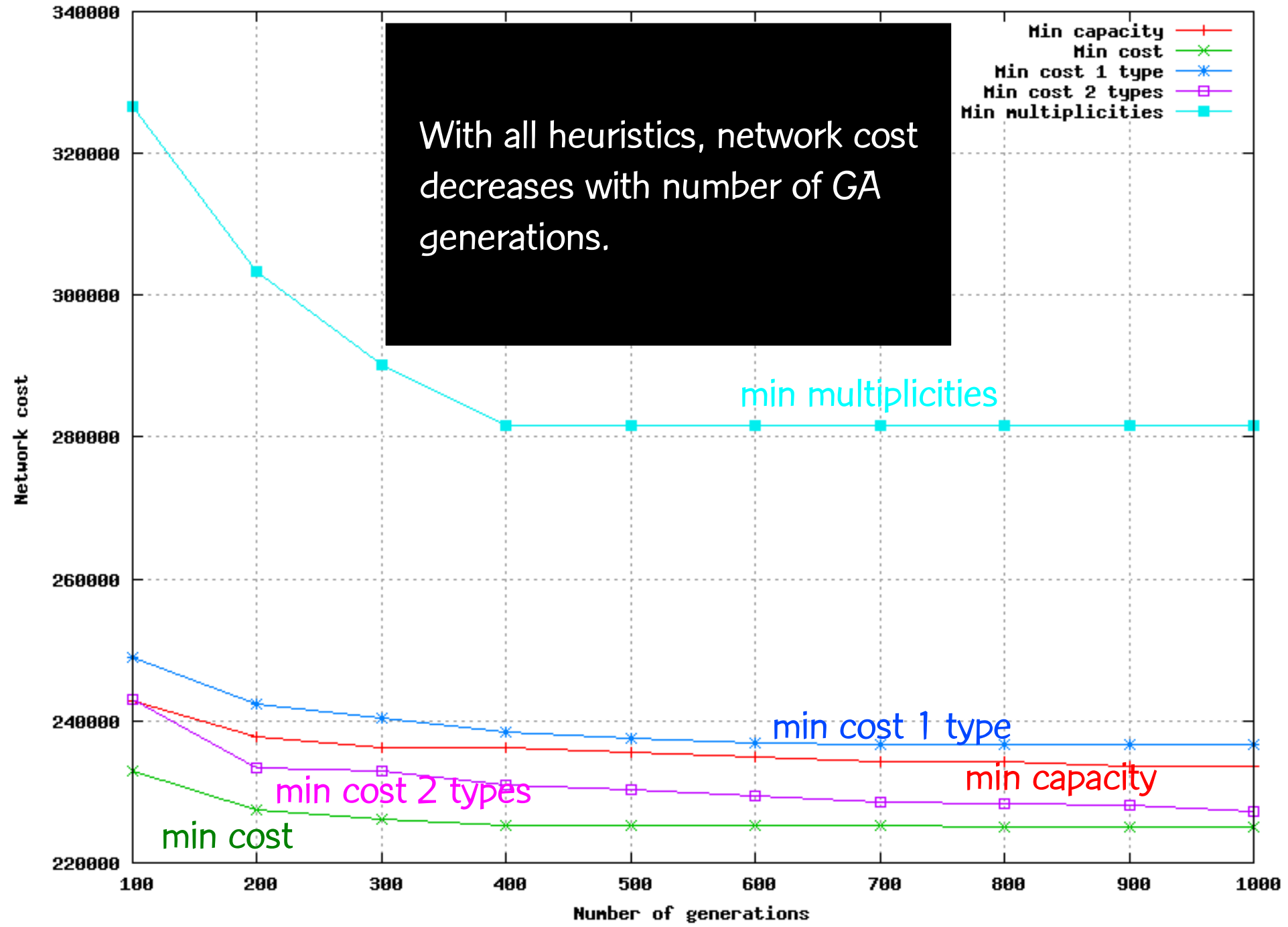
- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.

Experimental results

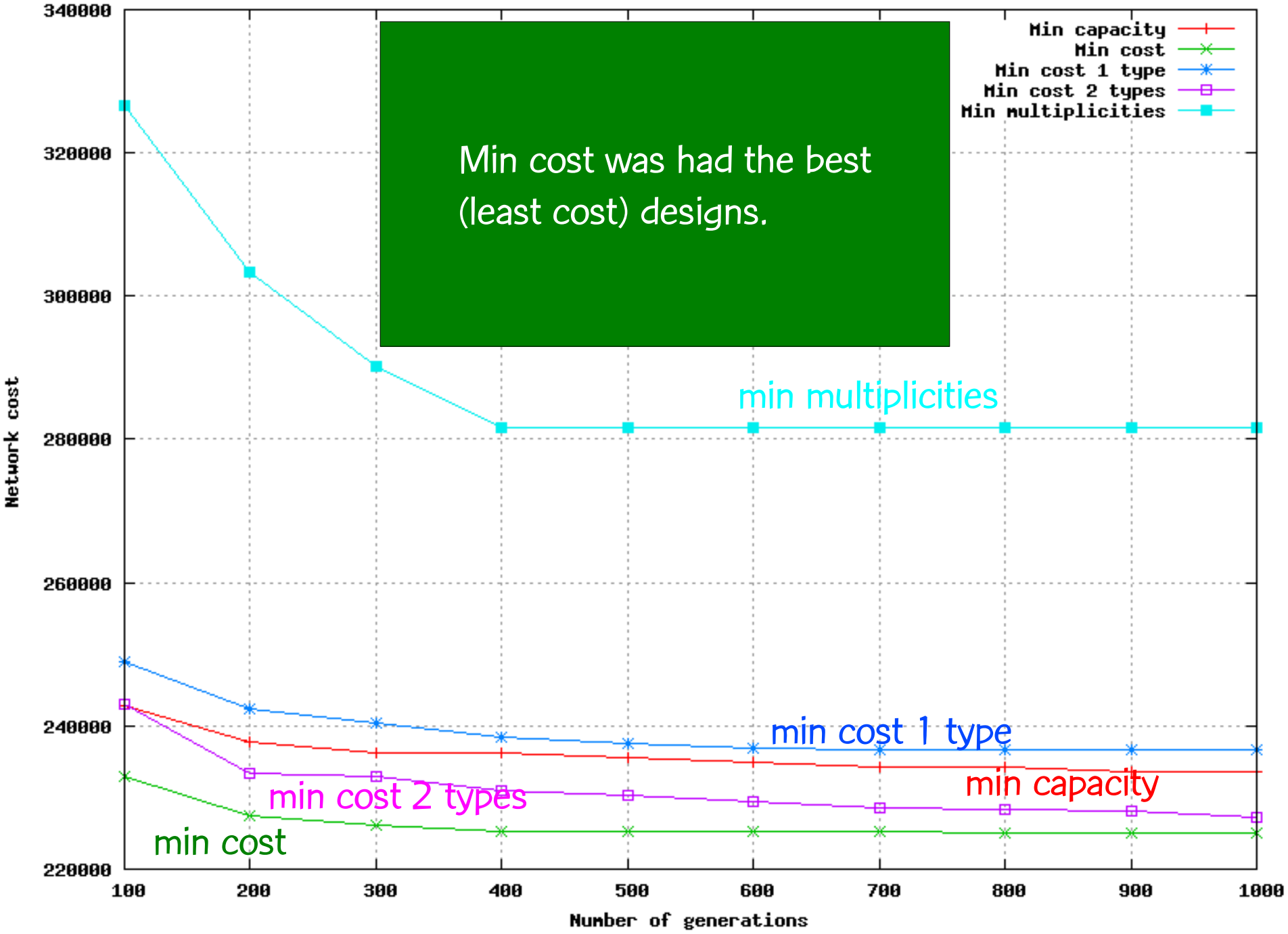
- Use a "real" network with 54 routers and 278 arcs.
- Three link types used: { 1, 2, 3 }
- $c(2) = 4\ c(1)$; $c(3) = 16\ c(1)$
- $p(2)/c(2) = 0.95\ p(1)/c(1)$; $p(3)/c(3) = 0.90\ p(1)/c(1)$
- All four heuristics tested. Min cost k types was tested for k=1 and k=2.
- GA was run 100, 200, 300, ..., 1000 generations and costs were recorded for each heuristic.



at&t
Your world. Delivered.



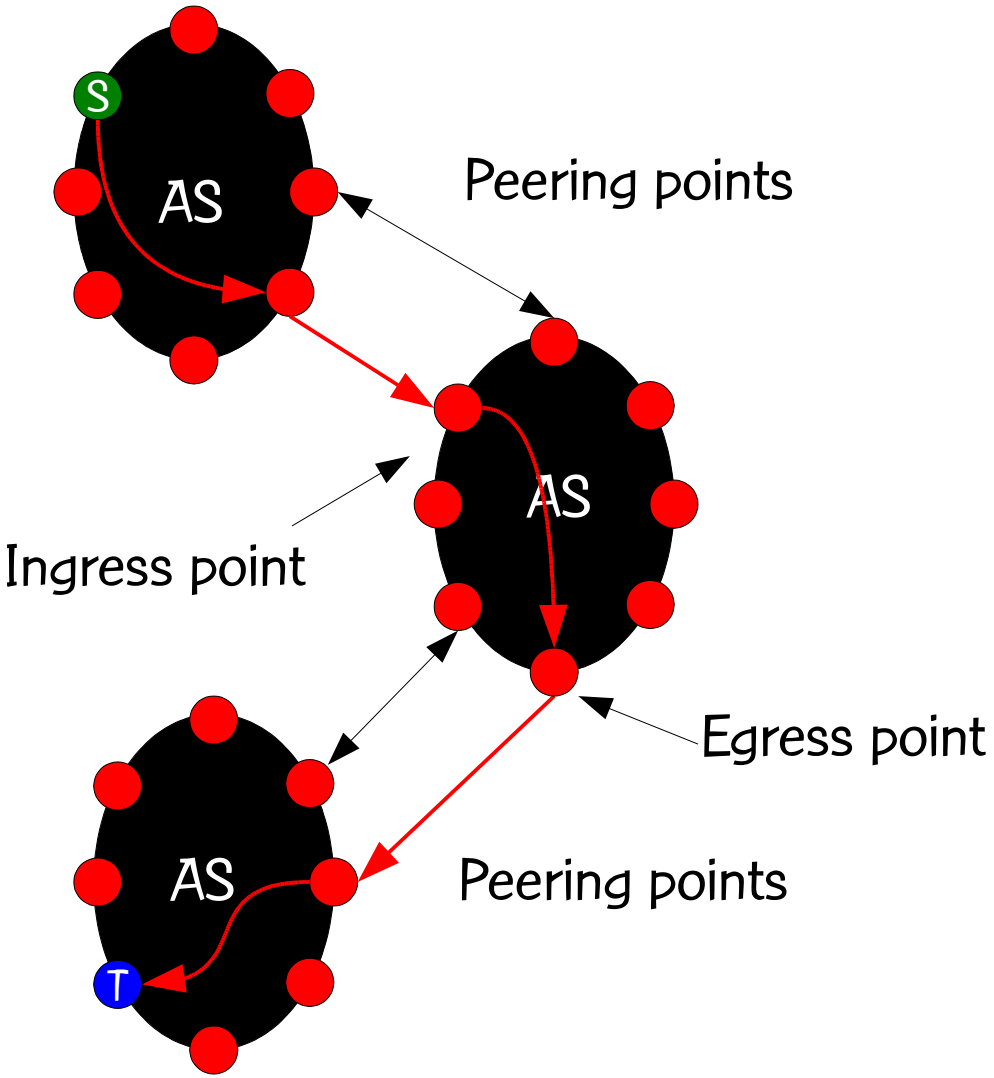
at&t
Your world. Delivered.

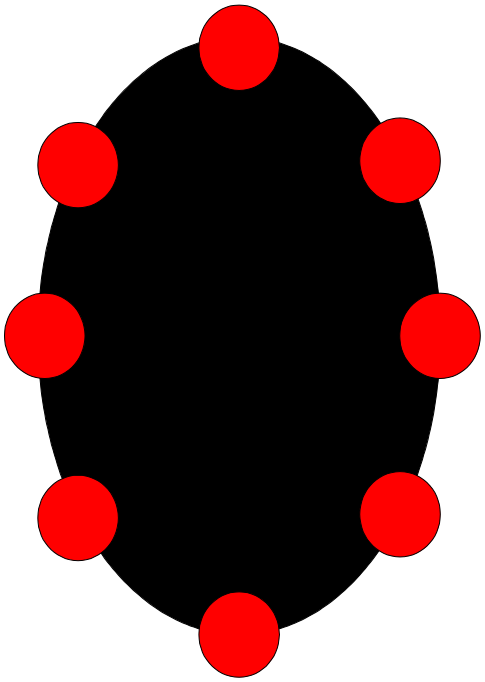


at&t
Your world. Delivered.

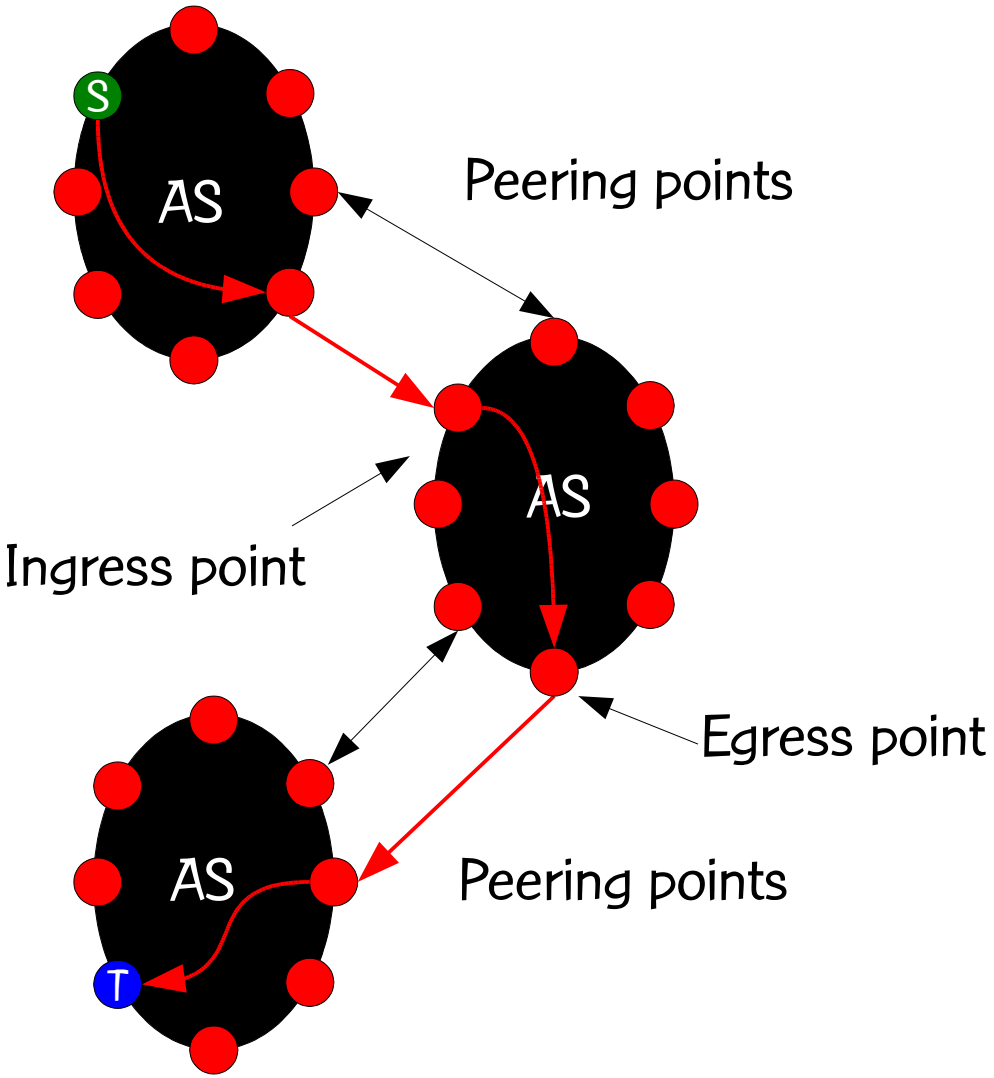
BGP Routing

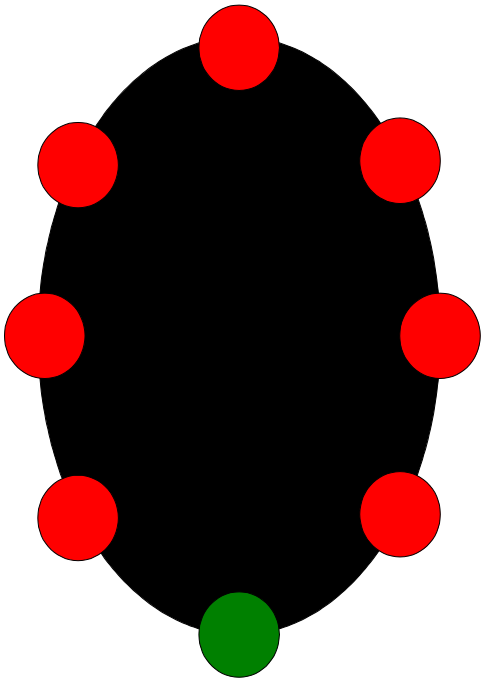
Egress selection





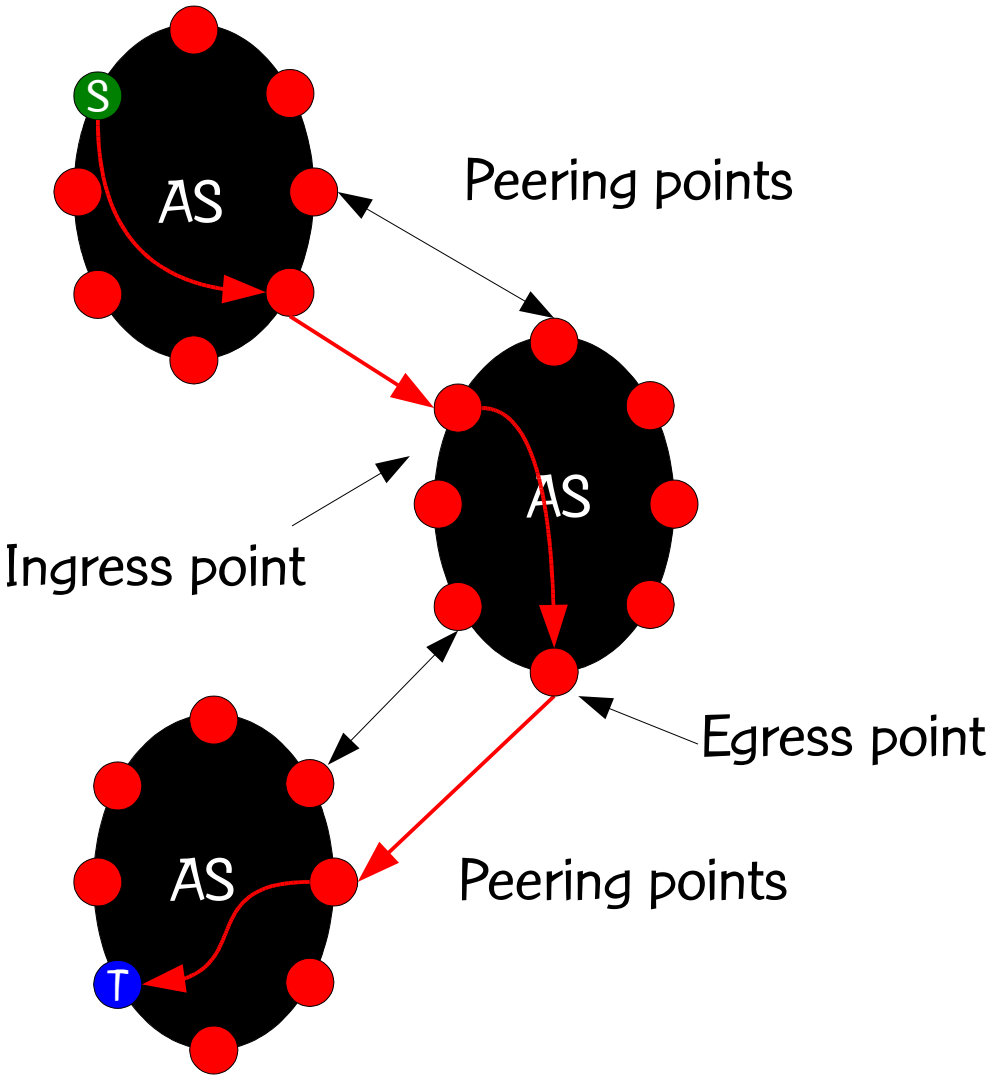
Egress selection

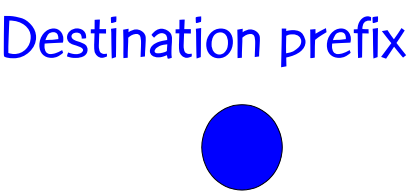


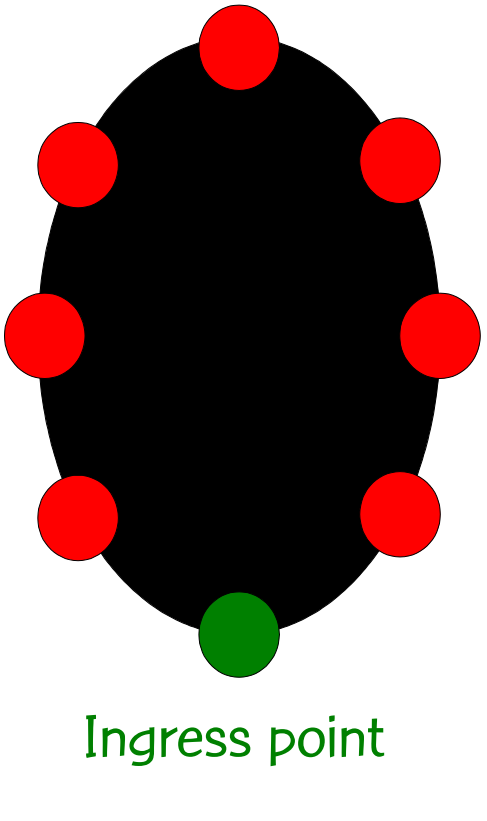


Ingress point

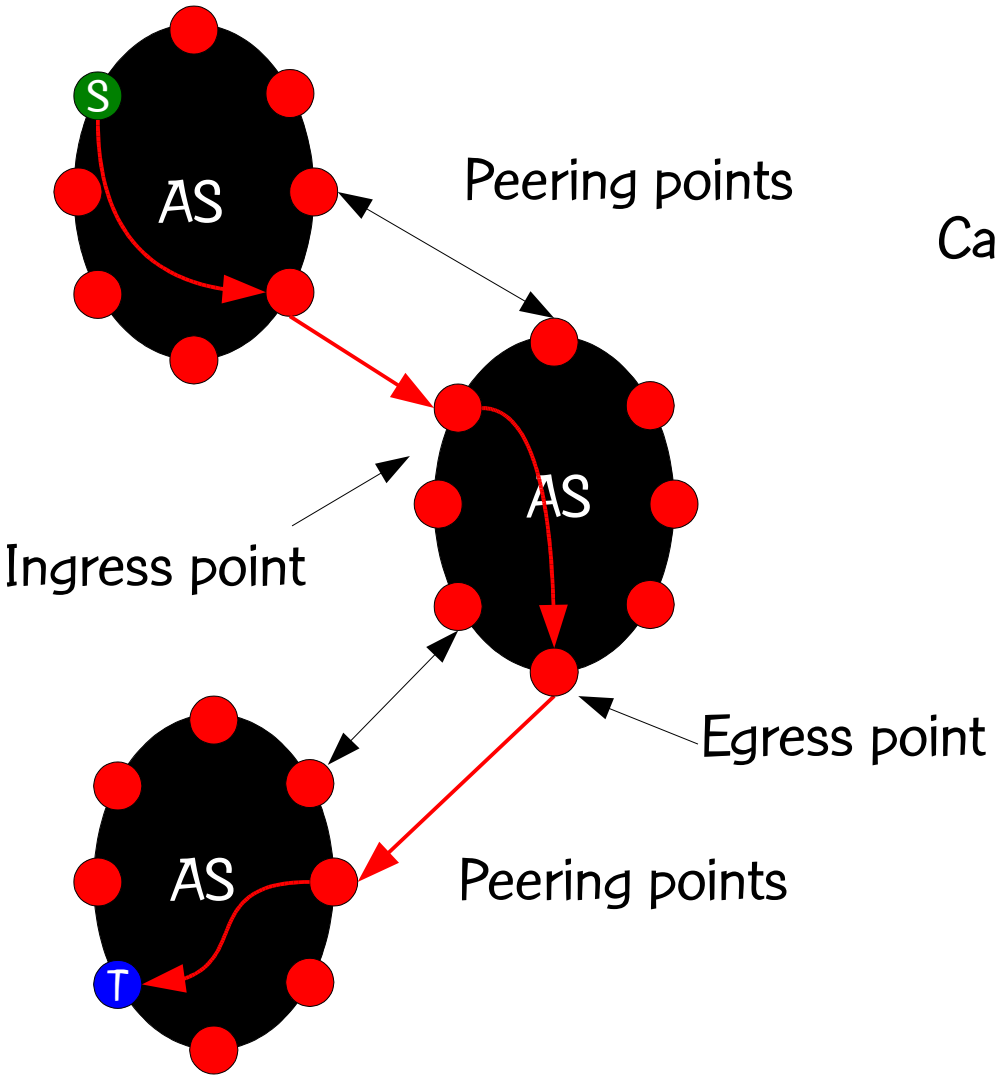
Egress selection





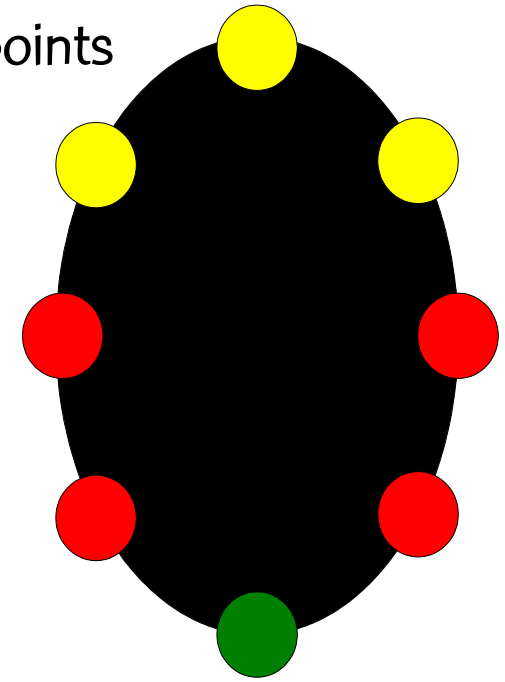


Egress selection



Destination prefix

Candidate egress points

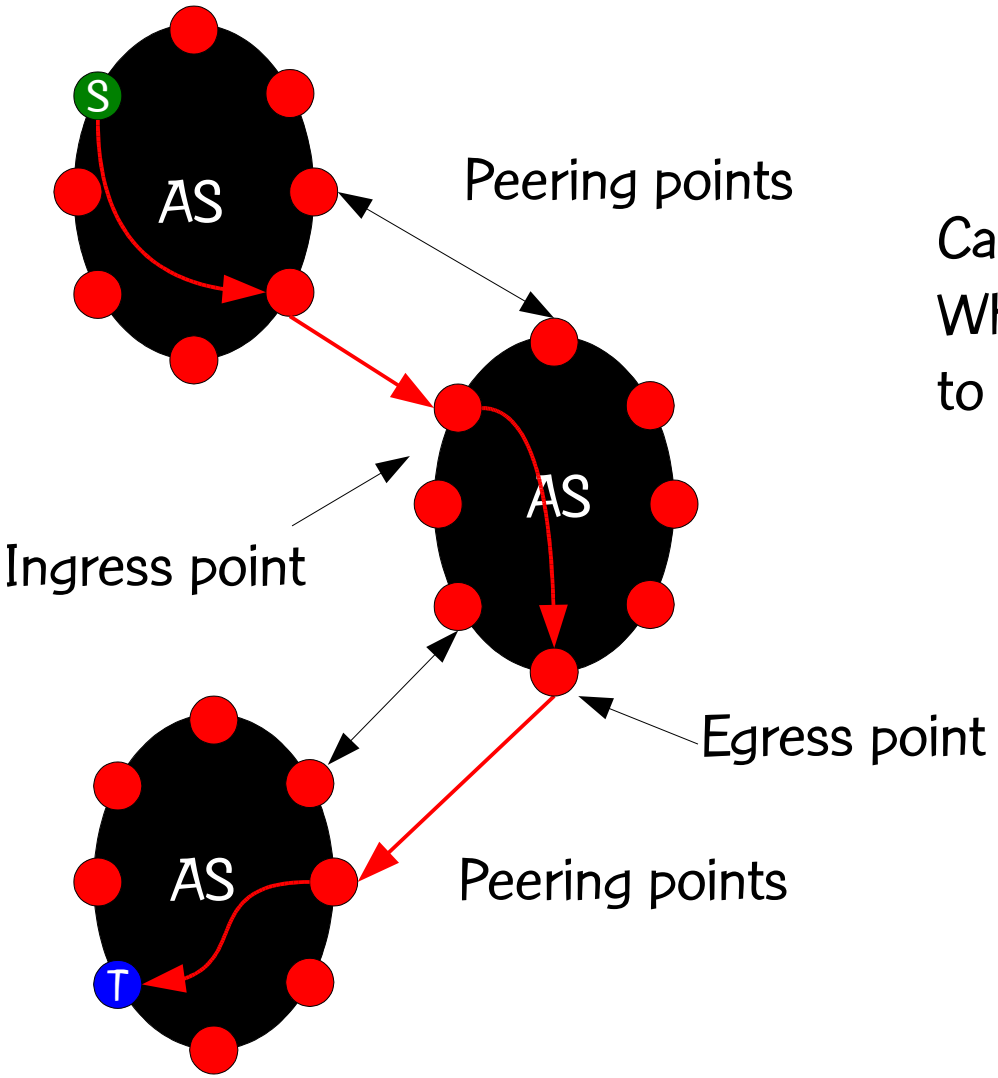


Ingress point

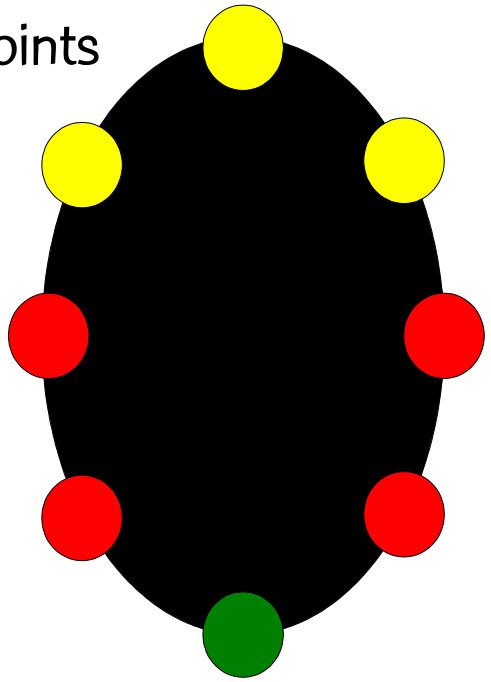
at&t
Your world. Delivered.

Egress selection

Destination prefix

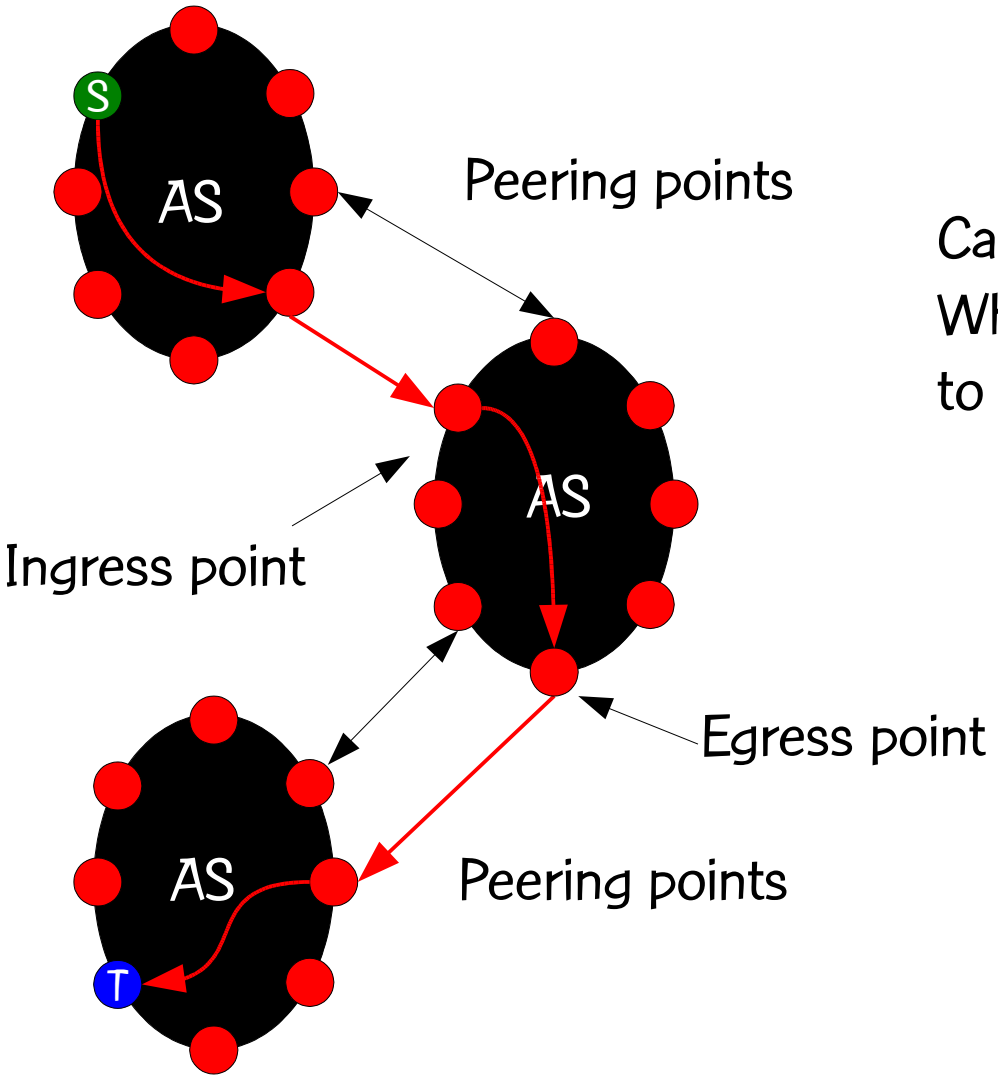


Candidate egress points
Which egress point
to select?



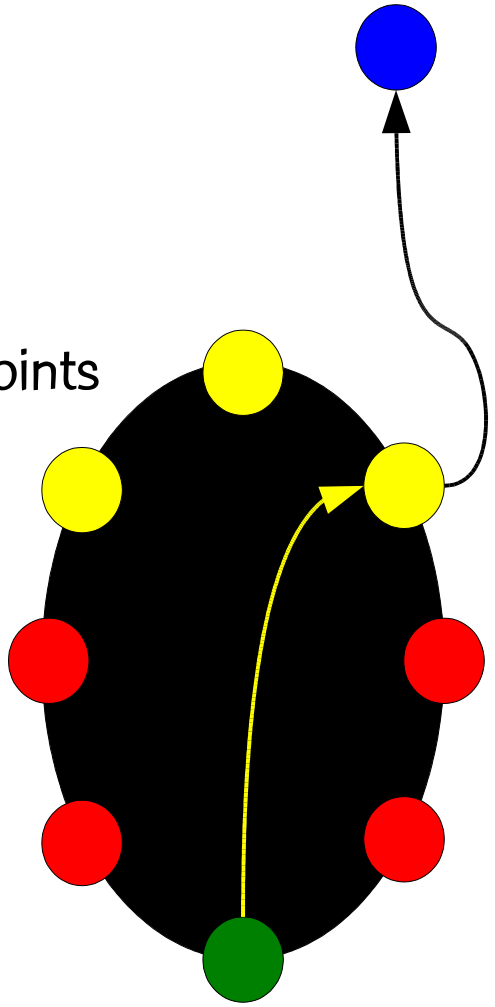
Ingress point

Egress selection



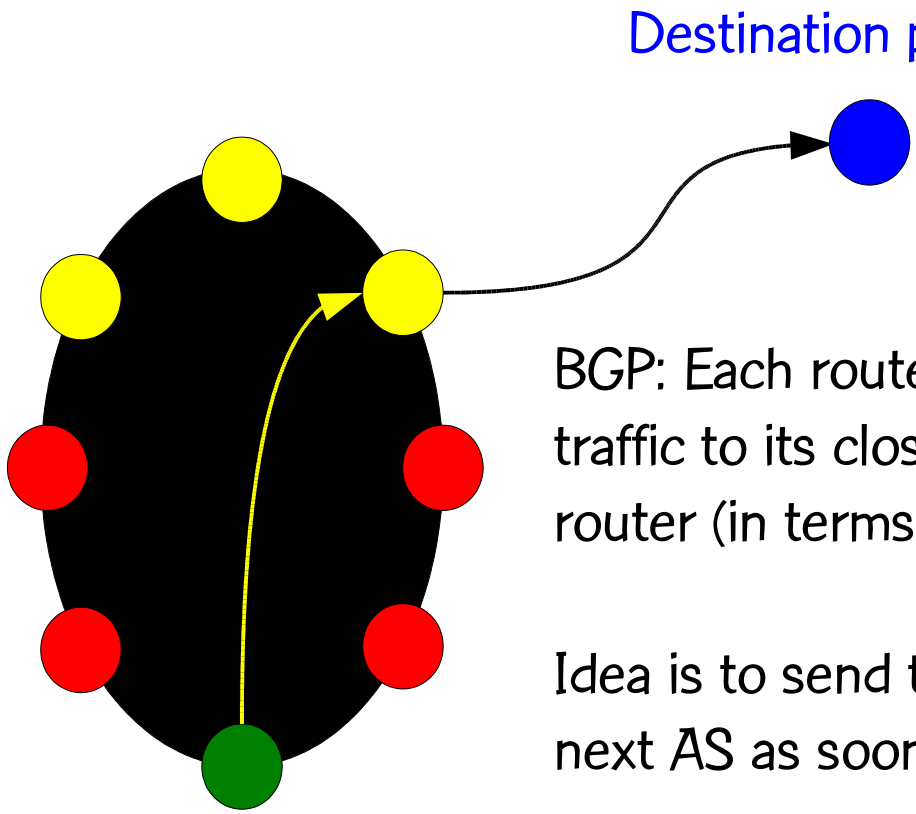
Destination prefix

Candidate egress points
Which egress point
to select?



Ingress point

Hot potato (early exit) routing



BGP: Each router in the AS directs traffic to its closest border router (in terms of IGP distances).

Idea is to send traffic along to next AS as soon as possible.

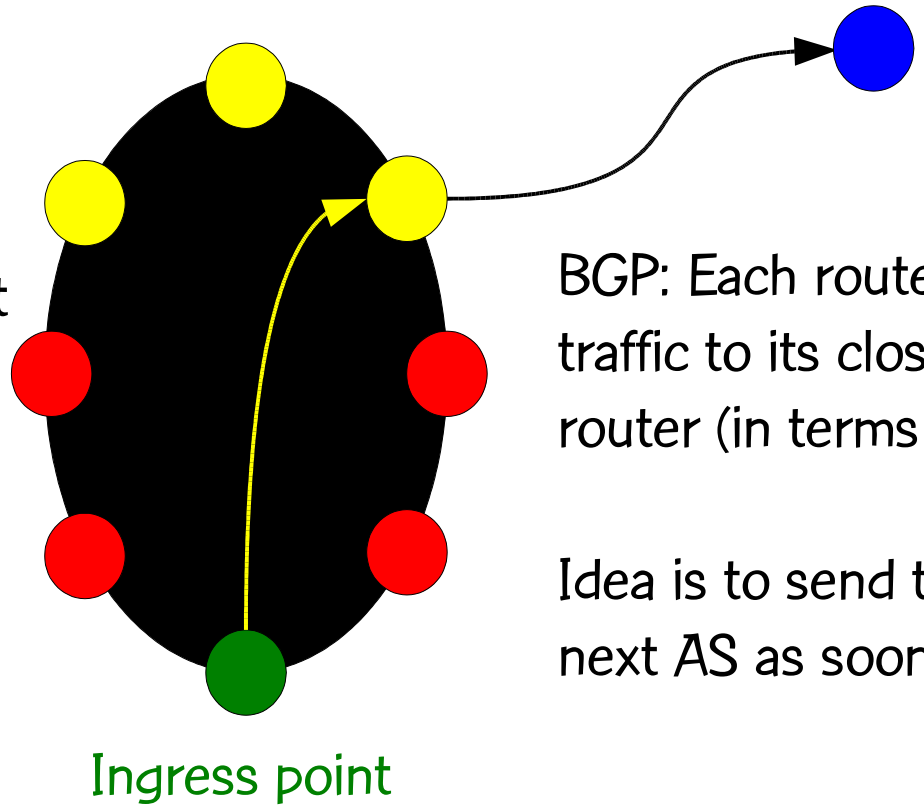
Hot potato (early exit) routing

We believe hot potato is:

a) Too restrictive: dictates a policy rather than support performance objectives.

b) Too disruptive: small changes in IGP distances can lead to large shifts in traffic.

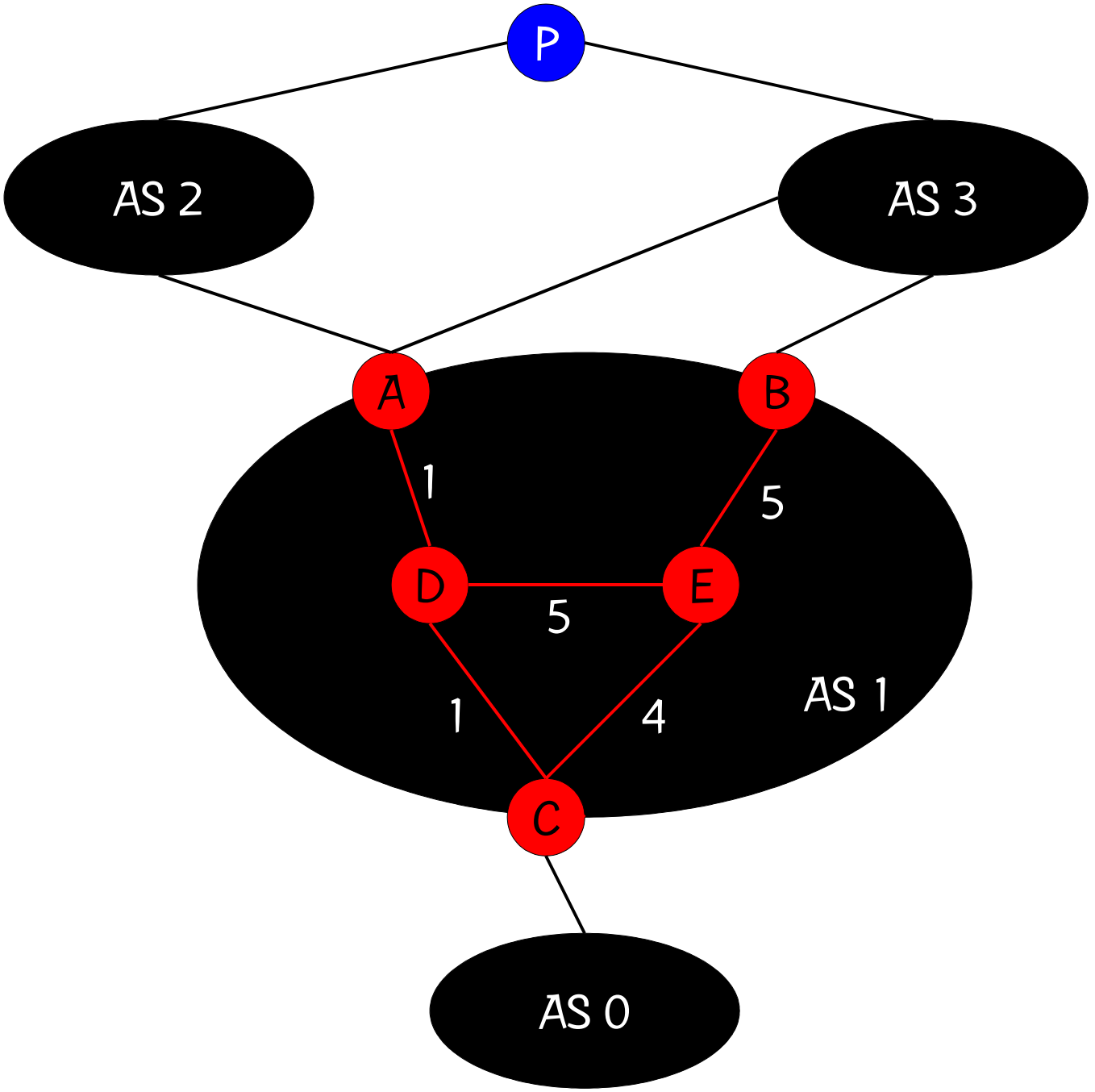
c) Too convoluted: administrators are forced to select IGP metrics that make BGP sense.



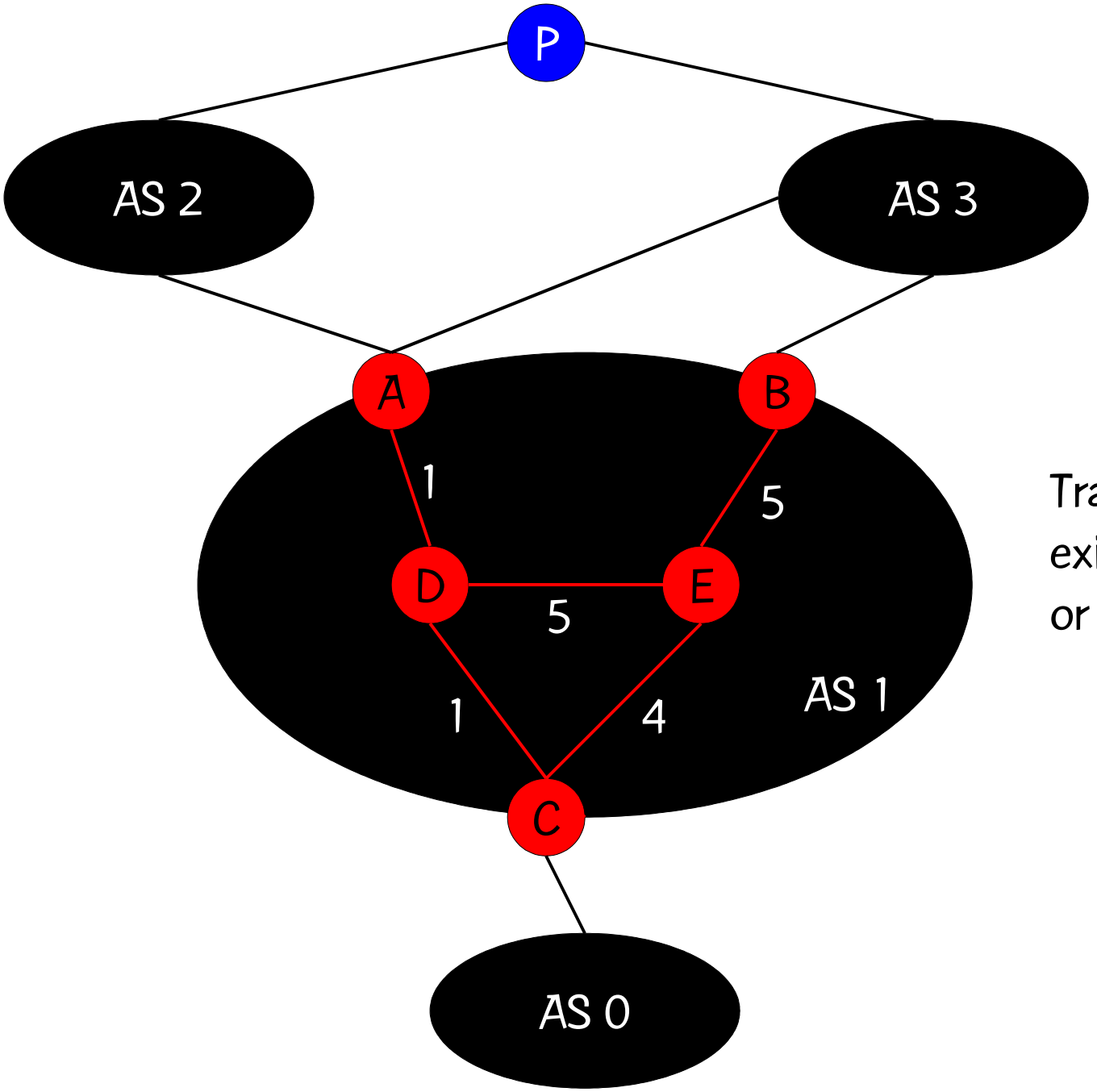
BGP: Each router in the AS directs traffic to its closest border router (in terms of IGP distances).

Idea is to send traffic along to next AS as soon as possible.

Hot potato sensitivity to failures



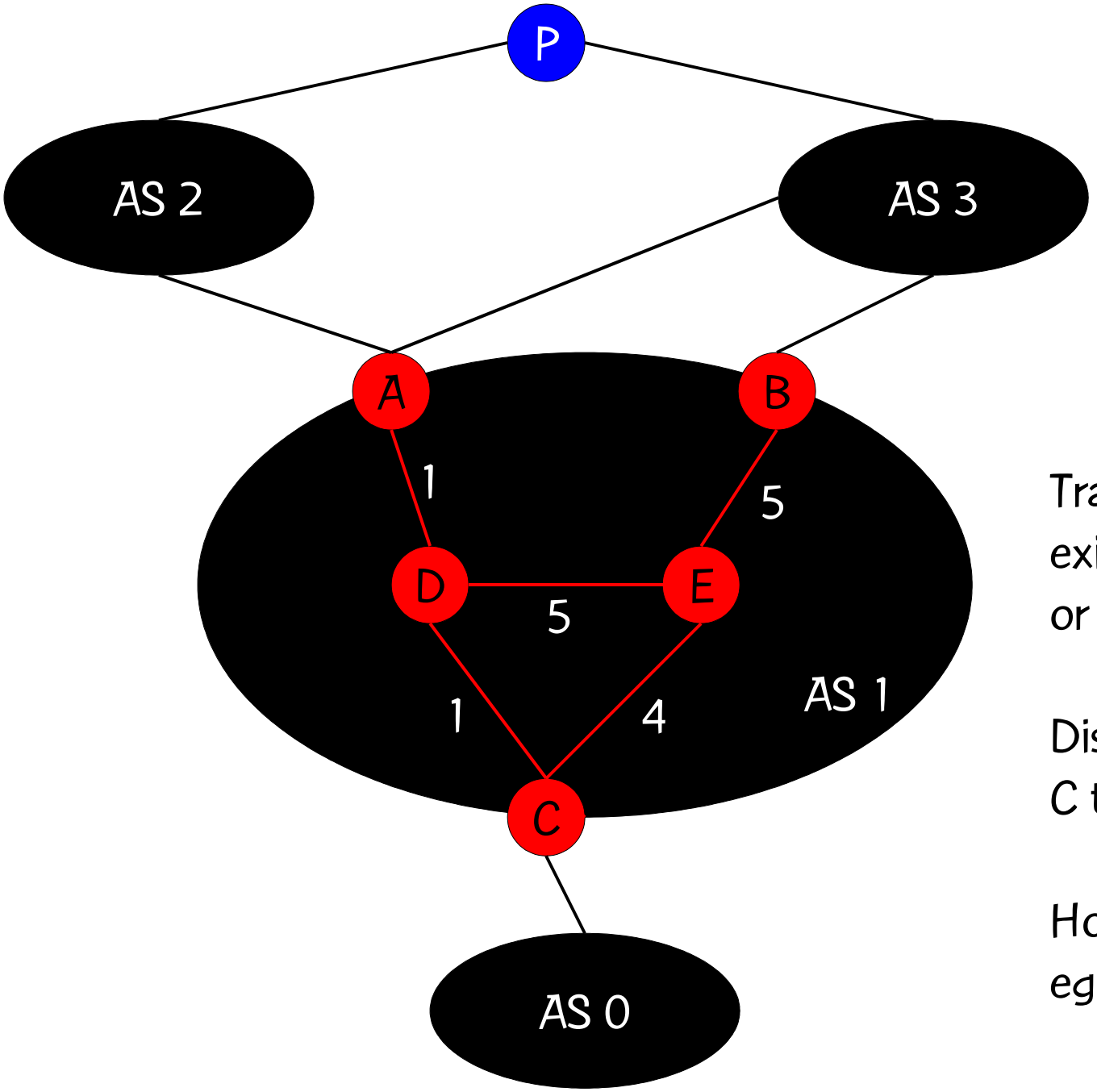
Hot potato sensitivity to failures



Traffic from C to P can exit AS 1 at either A or B.

at&t
Your world. Delivered.

Hot potato sensitivity to failures



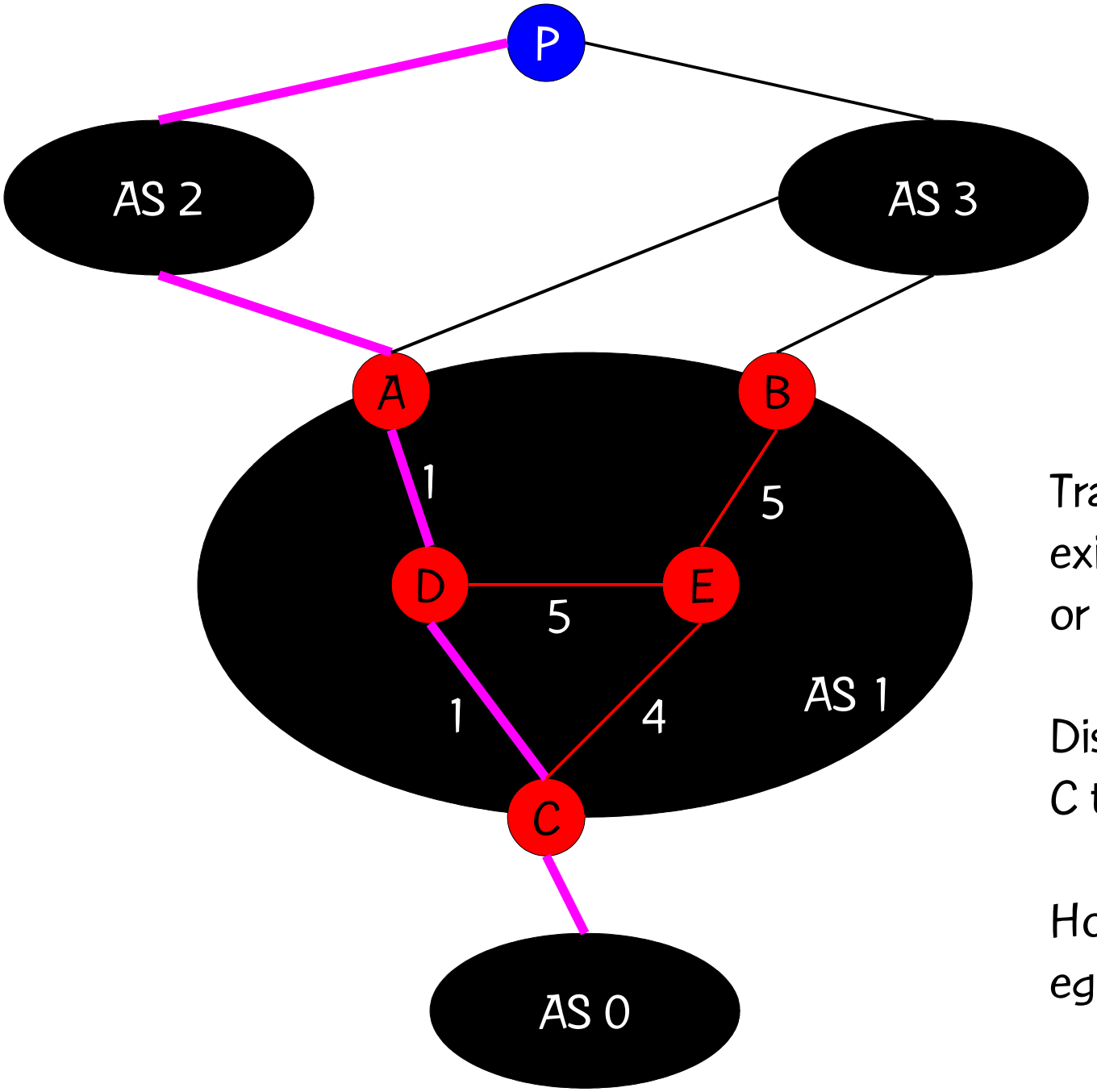
Traffic from C to P can exit AS 1 at either A or B.

Distance C to A is 2 & C to B is 9.

Hot potato selects egress point A.

at&t
Your world. Delivered.

Hot potato sensitivity to failures



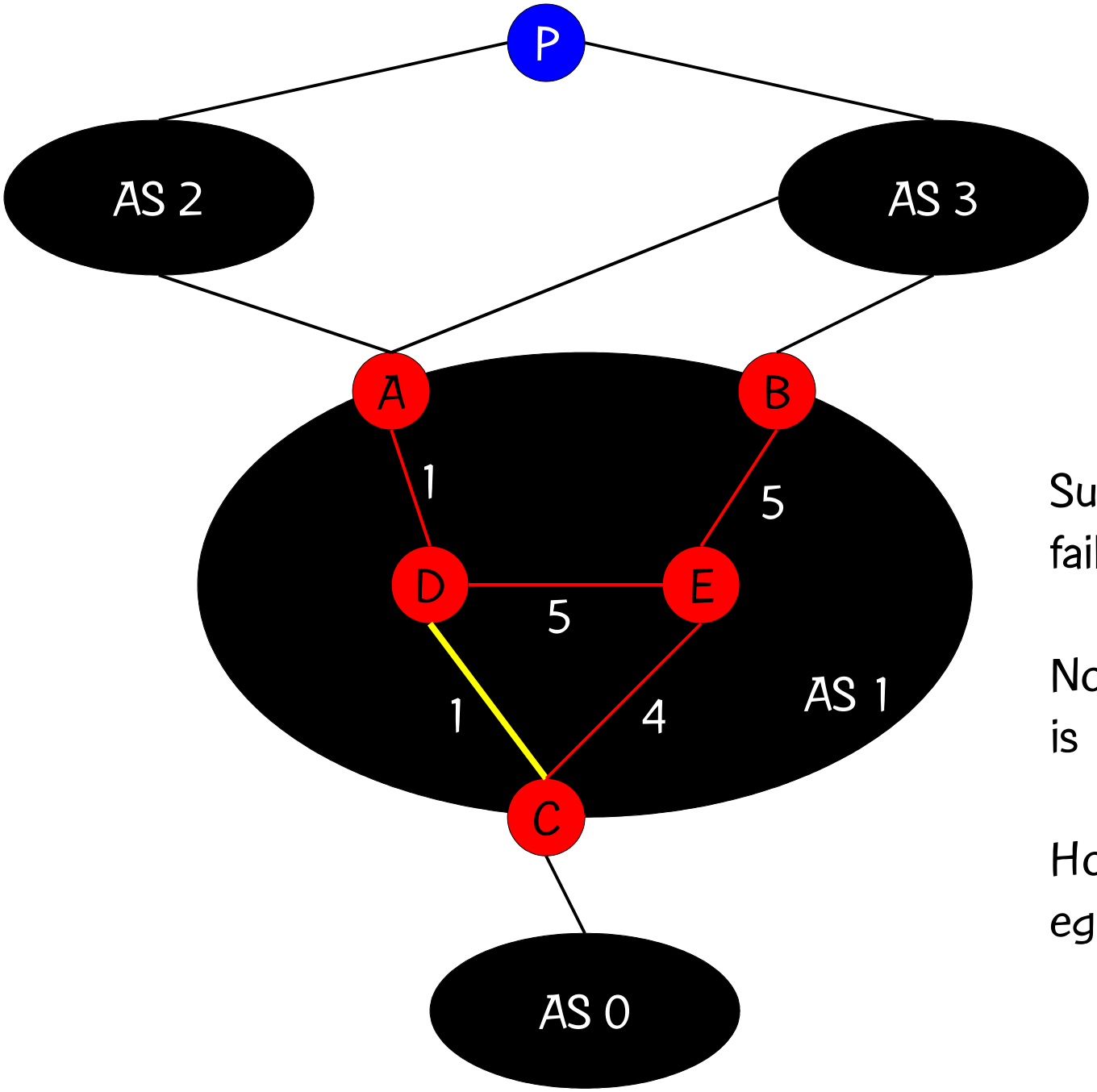
Traffic from C to P can exit AS 1 at either A or B.

Distance C to A is 2 & C to B is 9.

Hot potato selects egress point A.

at&t
Your world. Delivered.

Hot potato sensitivity to failures



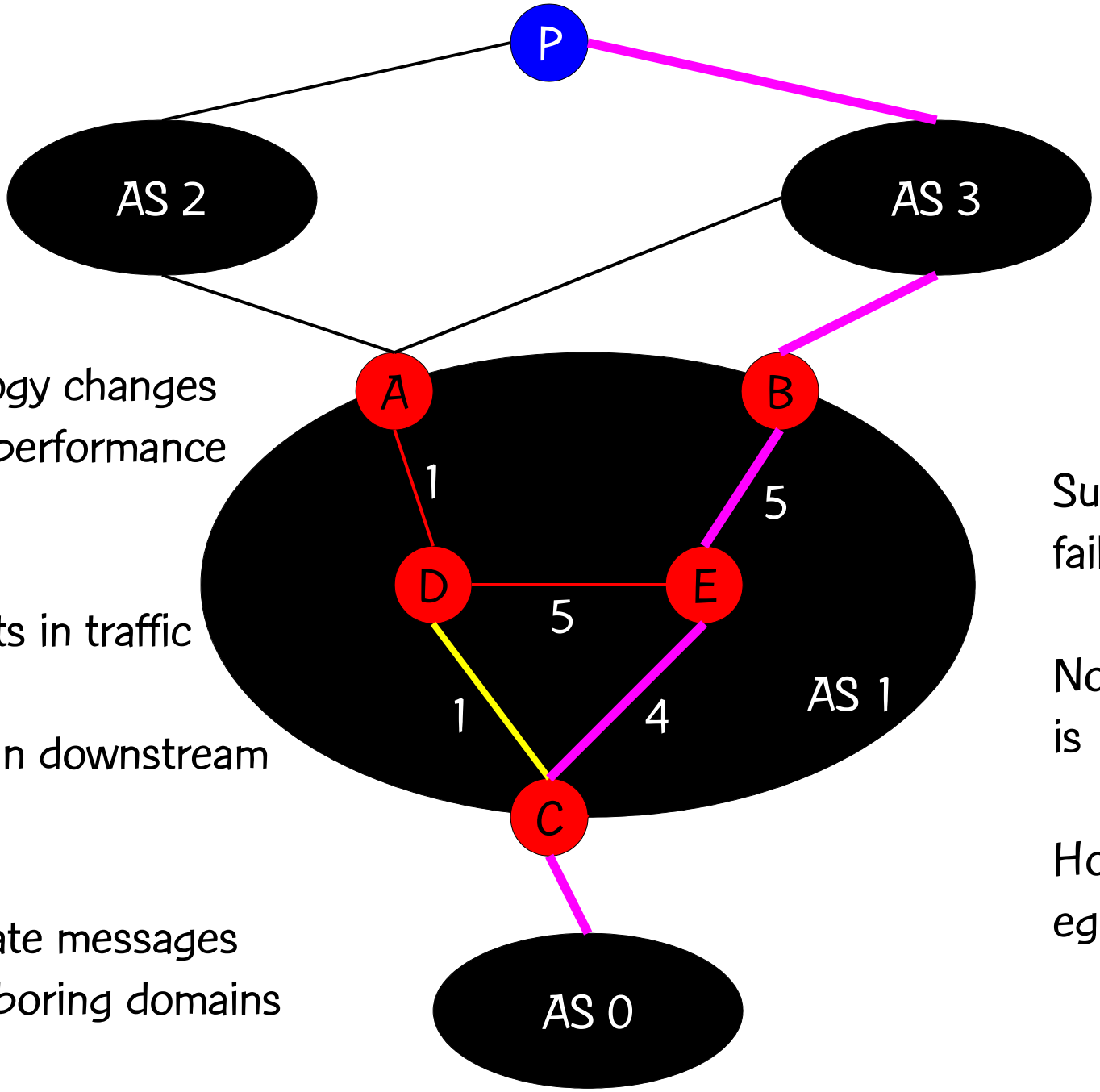
Suppose link C to D fails.

Now, distance C to A is 10 & C to B is 9.

Hot potato selects egress point B.

at&t
Your world. Delivered.

Hot potato sensitivity to failures



Small topology changes can lead to performance disruptions:

a) large shifts in traffic

b) changes in downstream paths

c) BGP update messages for neighboring domains

Suppose link C to D fails.

Now, distance C to A is 10 & C to B is 9.

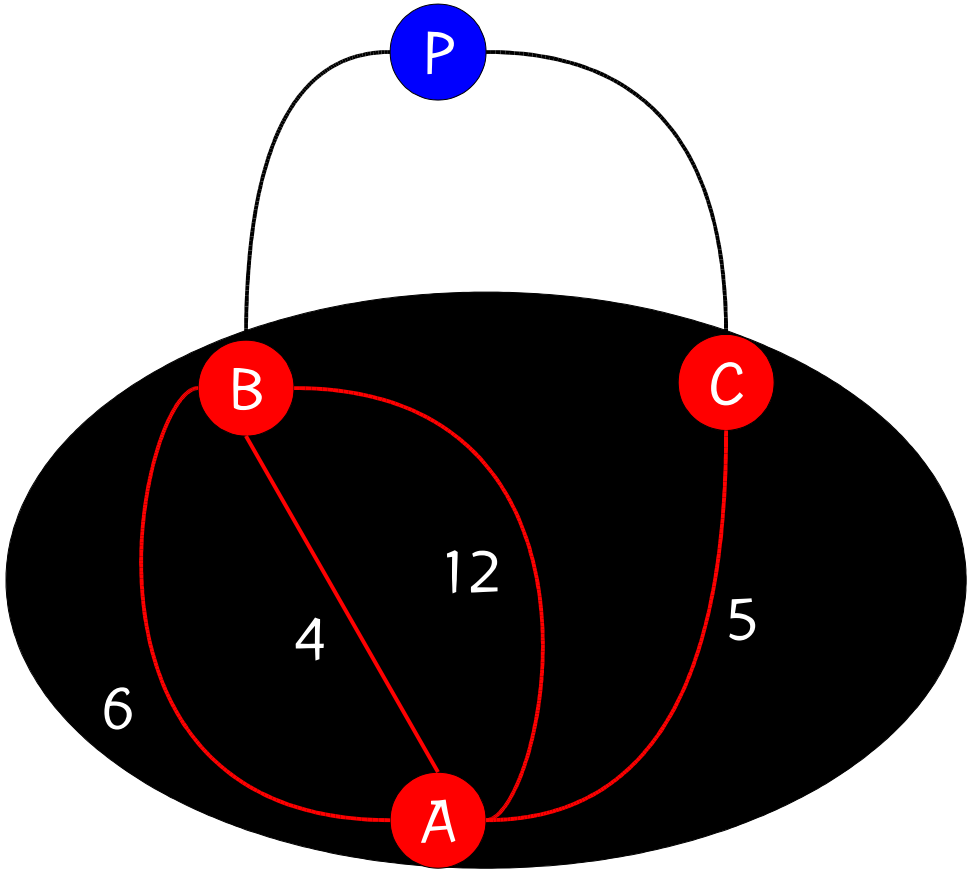
Hot potato selects egress point B.

at&t
Your world. Delivered.

Teixeira, Griffin, Resende, & Rexford, "TIE Breaking: Tunable Interdomain Egress Selection," IEEE/ACM Transactions on Networking, to appear (2006).

- Propose a method to solve the sensitivity problem.
- Ranking metric: router i has a metric m(i,p,e) across all prefixes p and egress points e.
- For each prefix p, router i selects egress point e that has the smallest value m(i,p,e).
- $m(i,p,e) = \alpha(i,p,e) \cdot d(G,i,e) + \beta(i,p,e)$, where d(G,i,e) is the shortest IGP distance from ingress router i to egress router e in AS G and $\alpha(i,p,e)$ and $\beta(i,p,e)$ are computed.

TIE metric



Initial topology:

$4 \cdot \alpha_B + \beta_B < 5 \cdot \alpha_C + \beta_C$

Failure mode of link with weight 4:

$6 \cdot \alpha_B + \beta_B < 5 \cdot \alpha_C + \beta_C$

Failure mode of links with weights 4 and 6:

$12 \cdot \alpha_B + \beta_B > 5 \cdot \alpha_C + \beta_C$

For all situations where propagation delay does not increase by a more than a factor of 2, we require that traffic from A to P go through B.

at&t
Your world. Delivered.

Constraint generation phase

$E(p)$: egress set for prefix p
ΔG: set of topology changes
$\delta(G)$: topology after change

For each (i,p) pair, do:

1) Identify closest egress point in original graph: $b = \text{argmin}\{d(G,i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b\}$, generate constraint:

$$\alpha(i,p,b)\cdot d(G,i,b) + \beta(i,p,b) < \alpha(i,p,e)\cdot d(G,i,e) + \beta(i,p,e)$$

For each network topology change $\delta \in \Delta G$, do:

1) Identify preferred egress point b':
   if $d(\delta(G),i,b) \leq T\cdot d(G,i,b)$, then $b'=b$
   else $b' = \text{argmin}\{d(\delta(G),i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b'\}$, generate constraint:

$$\alpha(i,p,b')\cdot d(\delta(G),i,b') + \beta(i,p,b') < \alpha(i,p,e)\cdot d(\delta(G),i,e) + \beta(i,p,e)$$

Constraint generation phase

$E(p)$: egress set for prefix p
ΔG: set of topology changes
$\delta(G)$: topology after change

For each (i,p) pair, do:

1) Identify closest egress point in original graph: $b = \text{argmin}\{d(G,i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b\}$, generate constraint:

$$\alpha(i,p,b)\cdot d(G,i,b) + \beta(i,p,b) < \alpha(i,p,e)\cdot d(G,i,e) + \beta(i,p,e)$$

For each network topology change $\delta \in \Delta G$, do:

1) Identify preferred egress point b':
   if $d(\delta(G),i,b) \leq T\cdot d(G,i,b)$, then $b'=b$
   else $b' = \text{argmin}\{d(\delta(G),i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b'\}$, generate constraint:

$$\alpha(i,p,b')\cdot d(\delta(G),i,b') + \beta(i,p,b') < \alpha(i,p,e)\cdot d(\delta(G),i,e) + \beta(i,p,e)$$

Algorithm produces $(|\Delta G|-1)\cdot(|E(p)|-1)$ constraints for each pair (i,p).

Size of E(p) is usually 1, 2, or 3 and at most 10.

at&t
Your world. Delivered.

Constraint generation phase

$E(p)$: egress set for prefix p
ΔG: set of topology changes
$\delta(G)$: topology after change

For each (i,p) pair, do:

1) Identify closest egress point in original graph: $b = \text{argmin}\{d(G,i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b\}$, generate constraint:

$$\alpha(i,p,b) \cdot d(G,i,b) + \beta(i,p,b) < \alpha(i,p,e) \cdot d(G,i,e) + \beta(i,p,e)$$

For each network topology change $\delta \in \Delta G$, do:

1) Identify preferred egress point b':
   if $d(\delta(G),i,b) \leq T \cdot d(G,i,b)$, then $b'=b$
   else $b' = \text{argmin}\{d(\delta(G),i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b'\}$, generate constraint:

$$\alpha(i,p,b') \cdot d(\delta(G),i,b') + \beta(i,p,b') < \alpha(i,p,e) \cdot d(\delta(G),i,e) + \beta(i,p,e)$$

Any prefixes that have the same egress set produce the same constraints and the same α and β values.

at&t
Your world. Delivered.

Constraint generation phase

E(p) : egress set for prefix p
ΔG: set of topology changes
$\delta(G)$: topology after change

For each (i,p) pair, do:

1) Identify closest egress point in original graph: $b = \text{argmin}\{d(G,i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b\}$, generate constraint:

$$\alpha(i,p,b)\cdot d(G,i,b) + \beta(i,p,b) < \alpha(i,p,e)\cdot d(G,i,e) + \beta(i,p,e)$$

For each network topology change $\delta \in \Delta G$, do:

1) Identify preferred egress point b':
   if $d(\delta(G),i,b) \leq T\cdot d(G,i,b)$, then $b'=b$
   else $b' = \text{argmin}\{d(\delta(G),i,e) \mid e \in E(p)\}$
2) For each $e \in E(p) \setminus \{b'\}$, generate constraint:

$$\alpha(i,p,b')\cdot d(\delta(G),i,b') + \beta(i,p,b') < \alpha(i,p,e)\cdot d(\delta(G),i,e) + \beta(i,p,e)$$

Number of unique egress sets is typically orders of magnitude smaller than number of prefixes.

at&t
Your world. Delivered.

Optimization phase

- Finite-precision parameter values: the α and β values should have finite precision to be configured on the routers and are therefore required to be integer:
  $\alpha(i,p,b) \cdot d(G,i,b) + \beta(i,p,b) \leq \alpha(i,p,e) \cdot d(G,i,e) + \beta(i,p,e) + 1$
- Robustness to unplanned events: to avoid having router i be unable to adapt to a change in IGP distance, we require $\alpha(i,p,e) \geq 1$, for all i, p, and e.
- Limiting the number of number of unique parameter values: to reduce overhead of configuring and storing the α and β values, we minimize $\sum \alpha(i,p,e) + \beta(i,p,e)$ thus favoring solutions with $\alpha(i,p,e) = 1$ and $\beta(i,p,e) = 0$.

An experiment with TIE

- Abilene network (April 2003):
  - Backbone of US research network
  - 11 PoPs with one router each
  - 7500 prefixes
  - 23 distinct egress sets
  - link weights are geographic distance to approximate propagation
- TIE optimized for single-node failures but evaluated with single-link failures
- delay threshold $T=2$

An experiment with TIE

- Used AMPL/CPLEX to determine α and β values
- Simulation phase took 0.5 s on Sun Fire 15000
- Optimization phase took 37 s on SGI Challenge
- $\alpha = 1$ for 93% of (i,p,e) tuples and had only four distinct values $\in \{1,2,3,4\}$
- $\beta = 0$ for 90% of (i,p,e) tuples and had only three distinct values $\in \{0,1,3251\}$

An experiment with TIE

- Compare TIE with Hot Potato ($\alpha = 1$ and $\beta = 0$) and Fixed Ranking ($\alpha = 0$ and $\beta = d(G,i,b(G,i,p))$ routing
- We make comparisons with two metrics
  - Delay ratio: For each (i,p,δ) we compute the delay for i to reach the best egress point for p after the topology change δ and divide it by the delay to reach the best egress in the original topology.
  - Routing sensitivity: For each (i,δ) we compute the fraction of prefixes at i that change egress point after a topology change δ.

An experiment with TIE

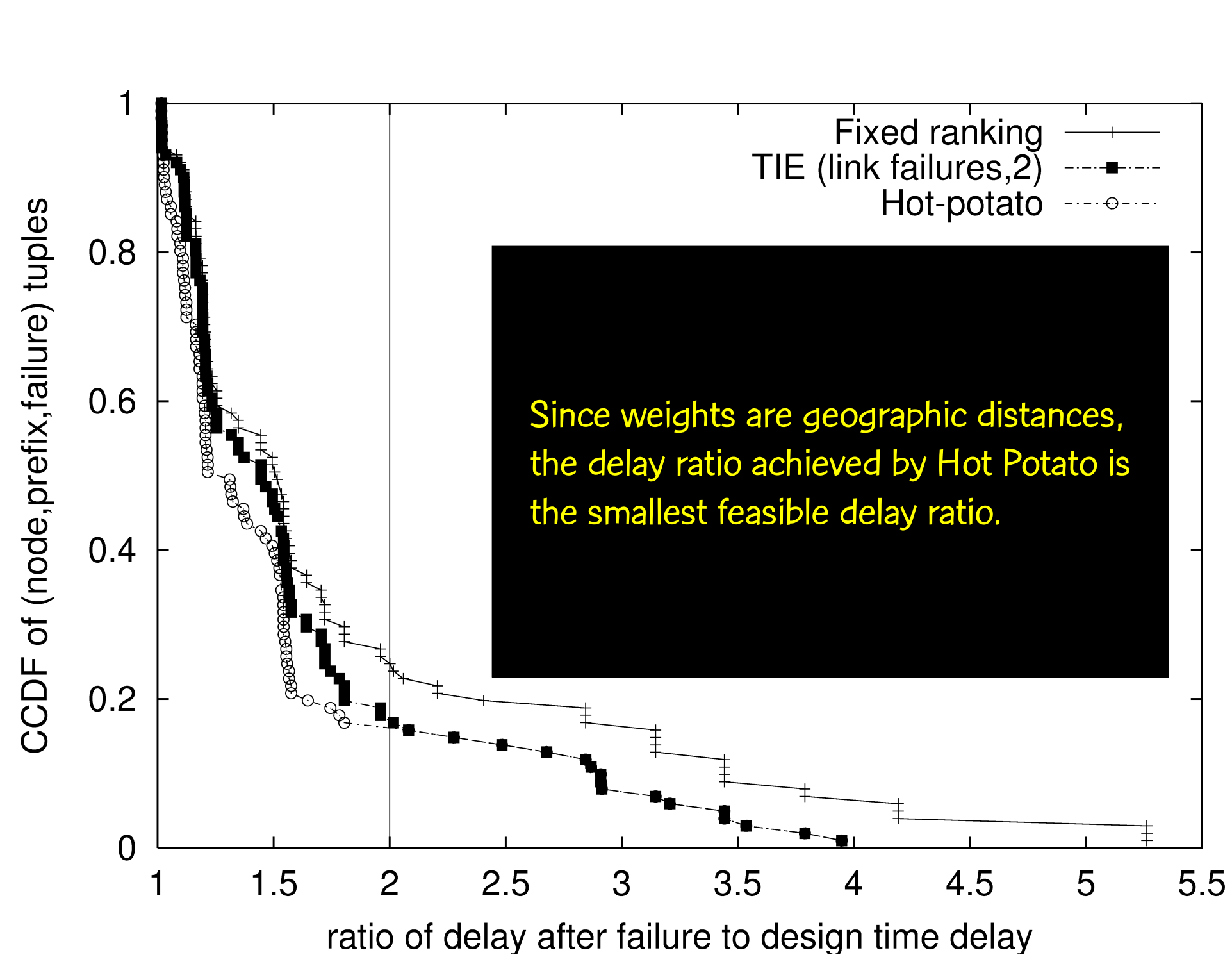
- Compare TIE with Hot Potato ($\alpha = 1$ and $\beta = 0$) and Fixed Ranking ($\alpha = 0$ and $\beta = d(G,i,b(G,i,p))$ routing
- We make comparisons with two metrics
  - Delay ratio: For each (i,p,δ) we compute the delay for i to reach the best egress point for p after the topology change δ and divide it by the delay to reach the best egress in the original topology. Cannot do better than Hot Potato.
  - Routing sensitivity: For each (i,δ) we compute the fraction of prefixes at i that change egress point after a topology change δ.

An experiment with TIE

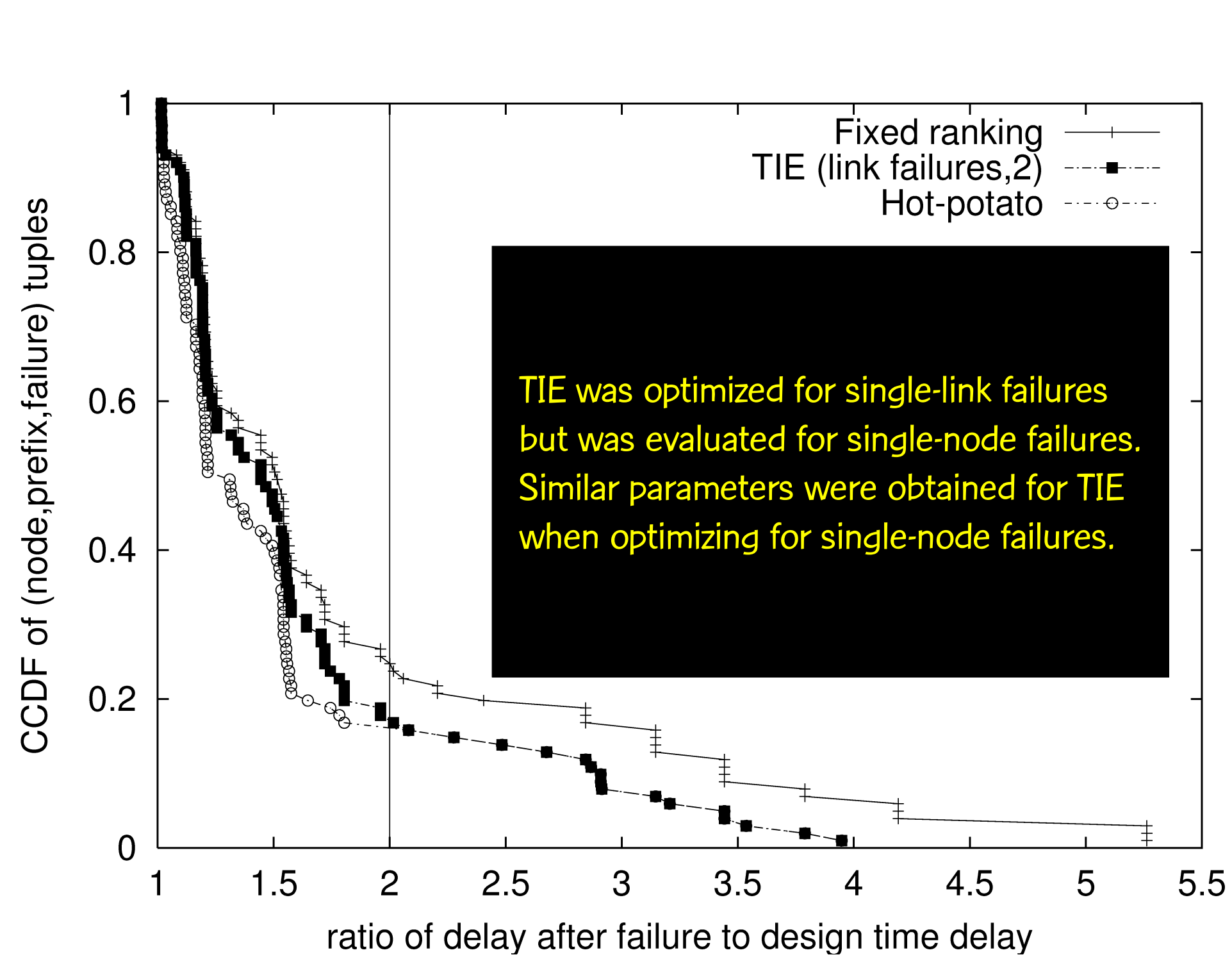
- Compare TIE with Hot Potato ($\alpha = 1$ and $\beta = 0$) and Fixed Ranking ($\alpha = 0$ and $\beta = d(G,i,b(G,i,p))$ routing
- We make comparisons with two metrics
  - Delay ratio: For each (i,p,δ) we compute the delay for i to reach the best egress point for p after the topology change δ and divide it by the delay to reach the best egress in the original topology. Cannot do better than Hot Potato.
  - Routing sensitivity: For each (i,δ) we compute the fraction of prefixes at i that change egress point after a topology change δ.

An experiment with TIE

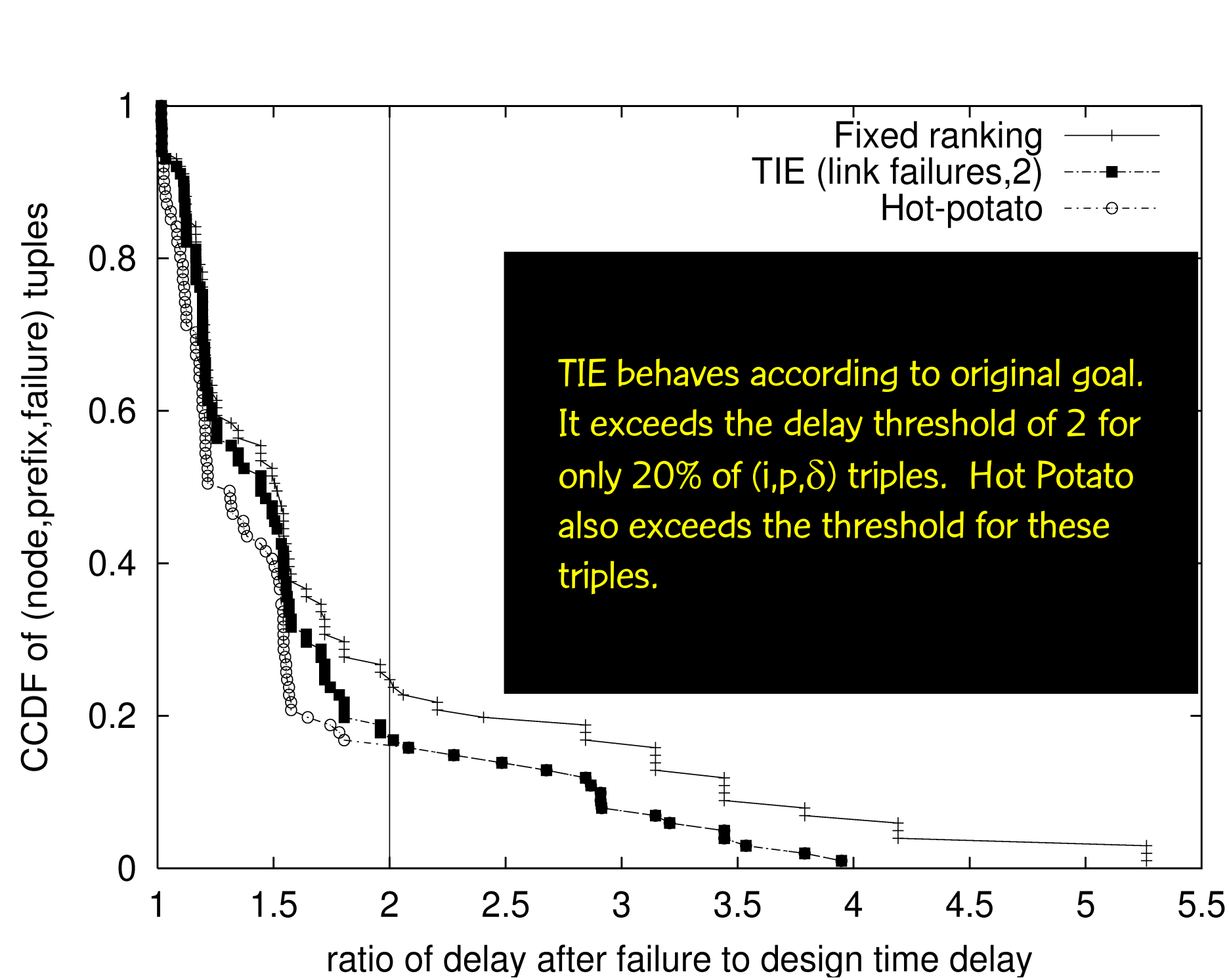
- Compare TIE with Hot Potato ($\alpha = 1$ and $\beta = 0$) and Fixed Ranking ($\alpha = 0$ and $\beta = d(G,i,b(G,i,p))$ routing
- We make comparisons with two metrics
  - Delay ratio: For each (i,p,δ) we compute the delay for i to reach the best egress point for p after the topology change δ and divide it by the delay to reach the best egress in the original topology. Cannot do better than Hot Potato.
  - Routing sensitivity: For each (i,δ) we compute the fraction of prefixes at i that change egress point after a topology change δ. Cannot do better than Fixed Ranking.



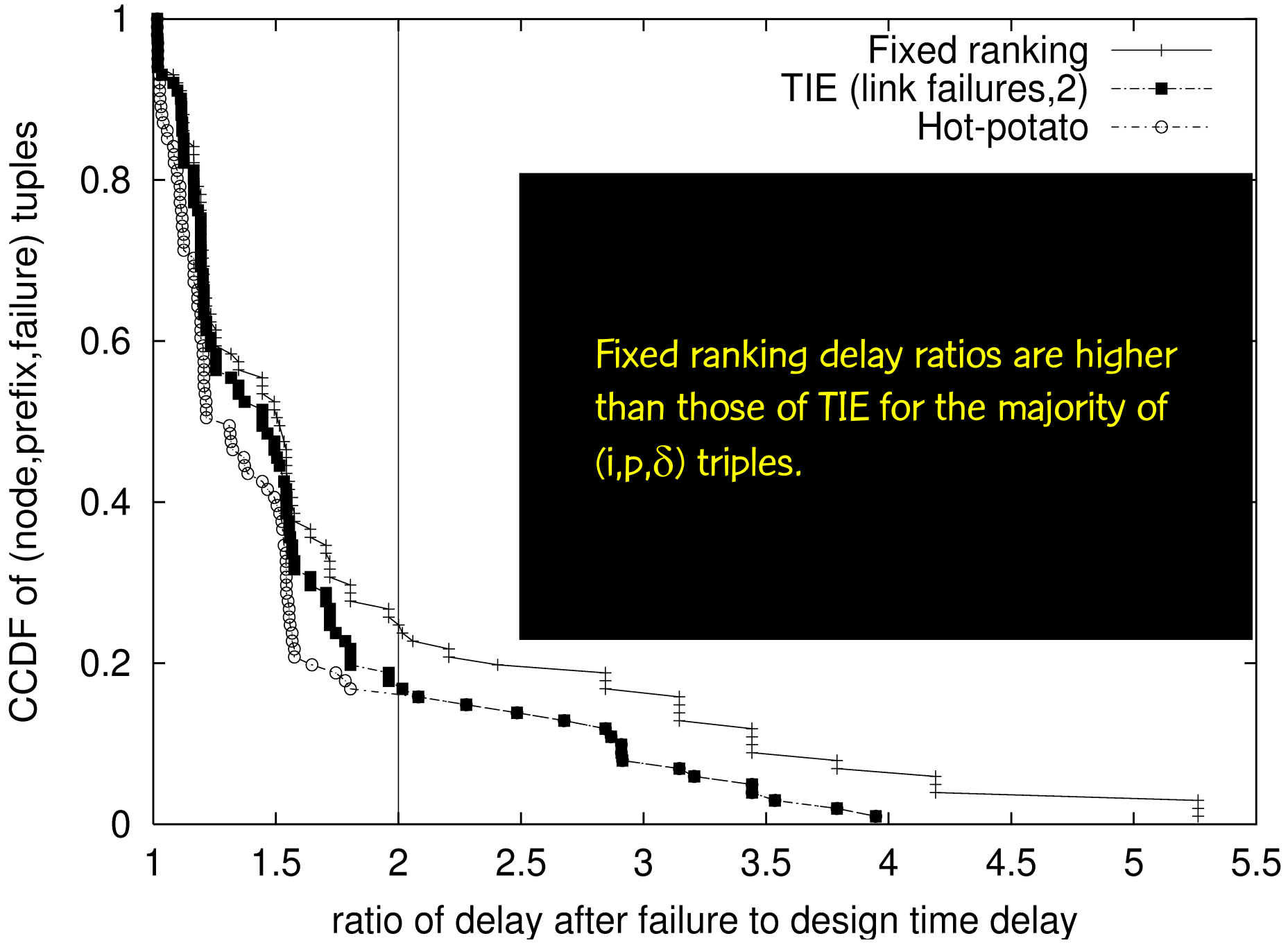
at&t
Your world. Delivered.



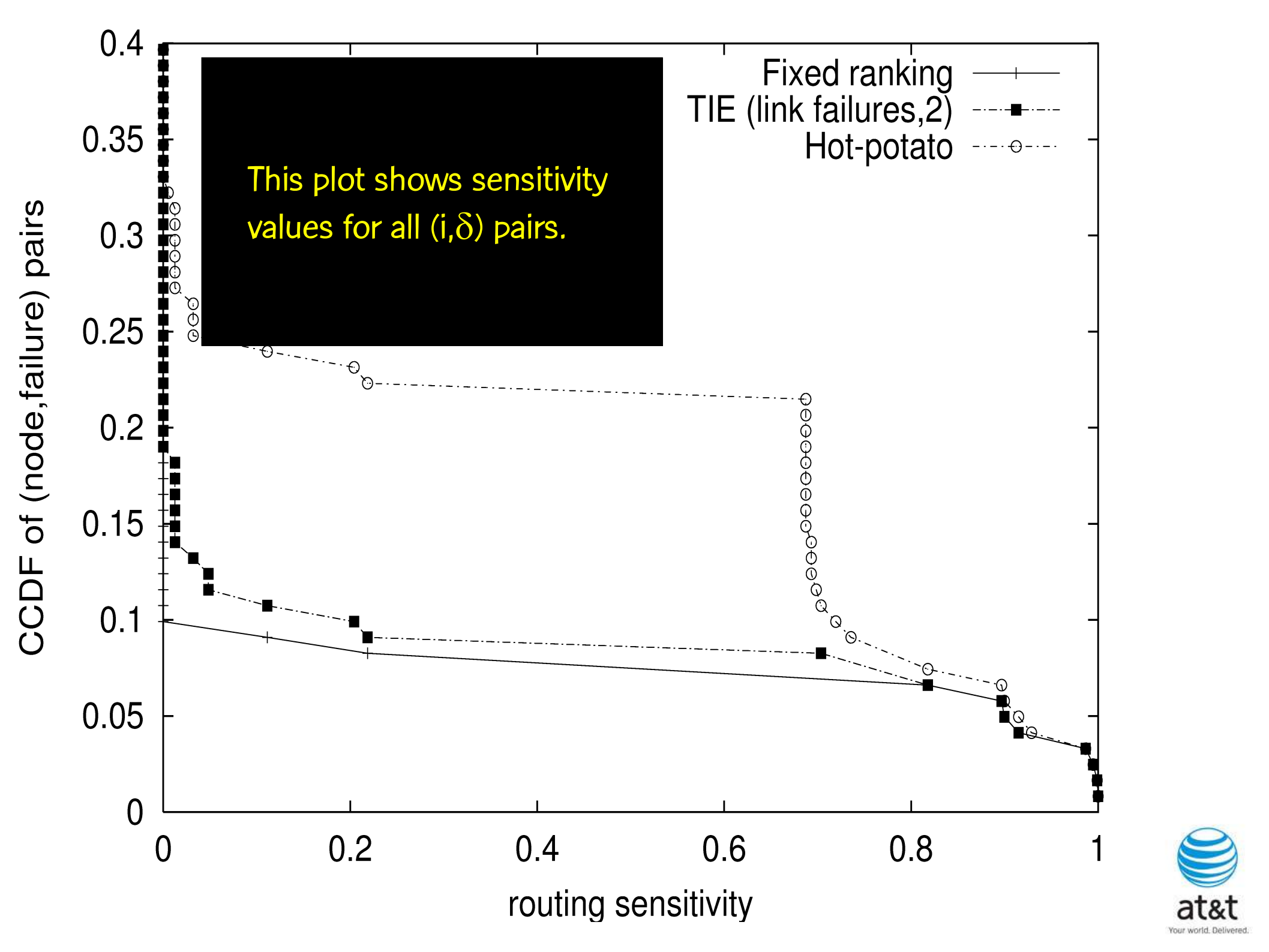
at&t
Your world. Delivered.



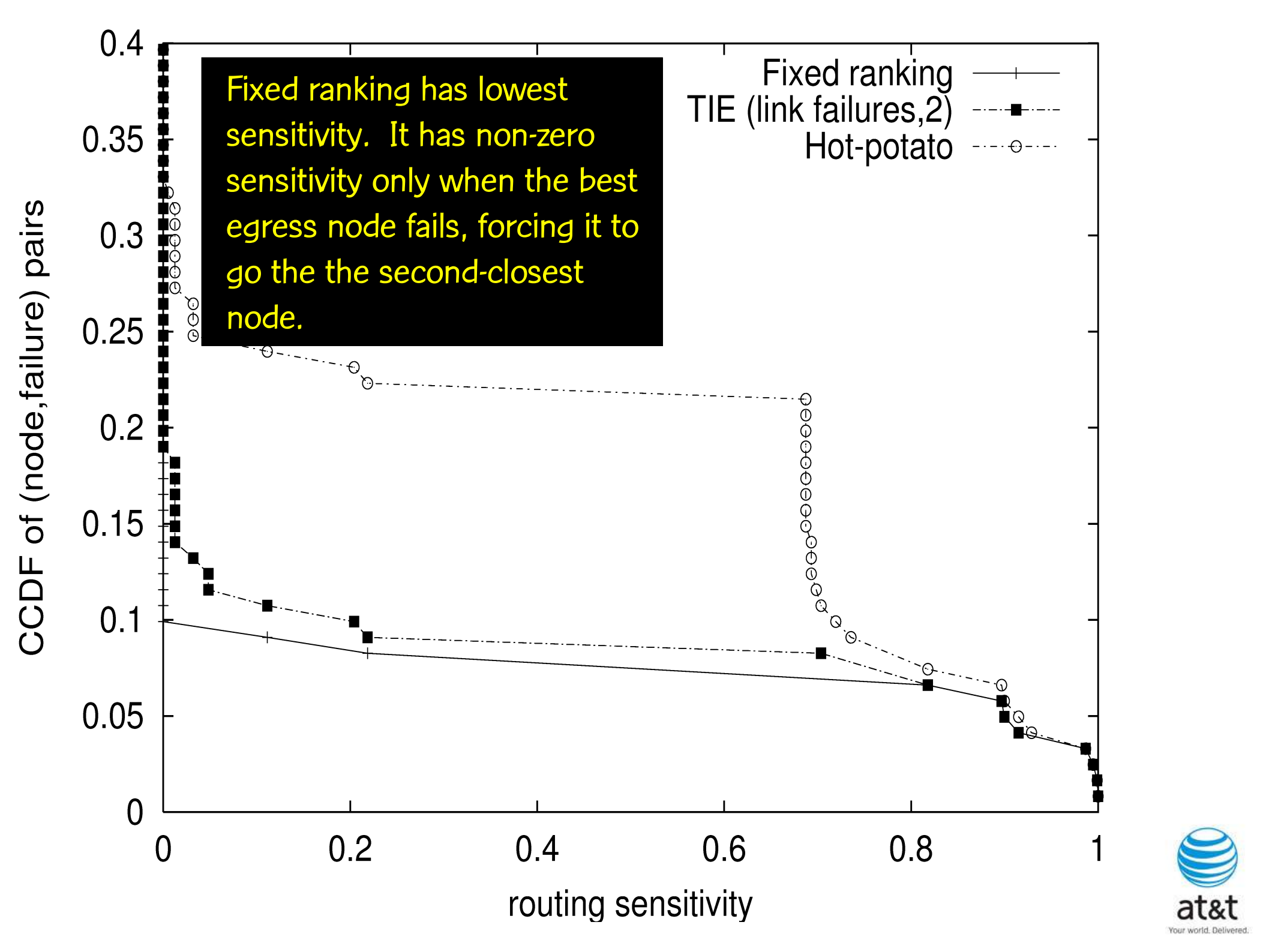
at&t
Your world. Delivered.



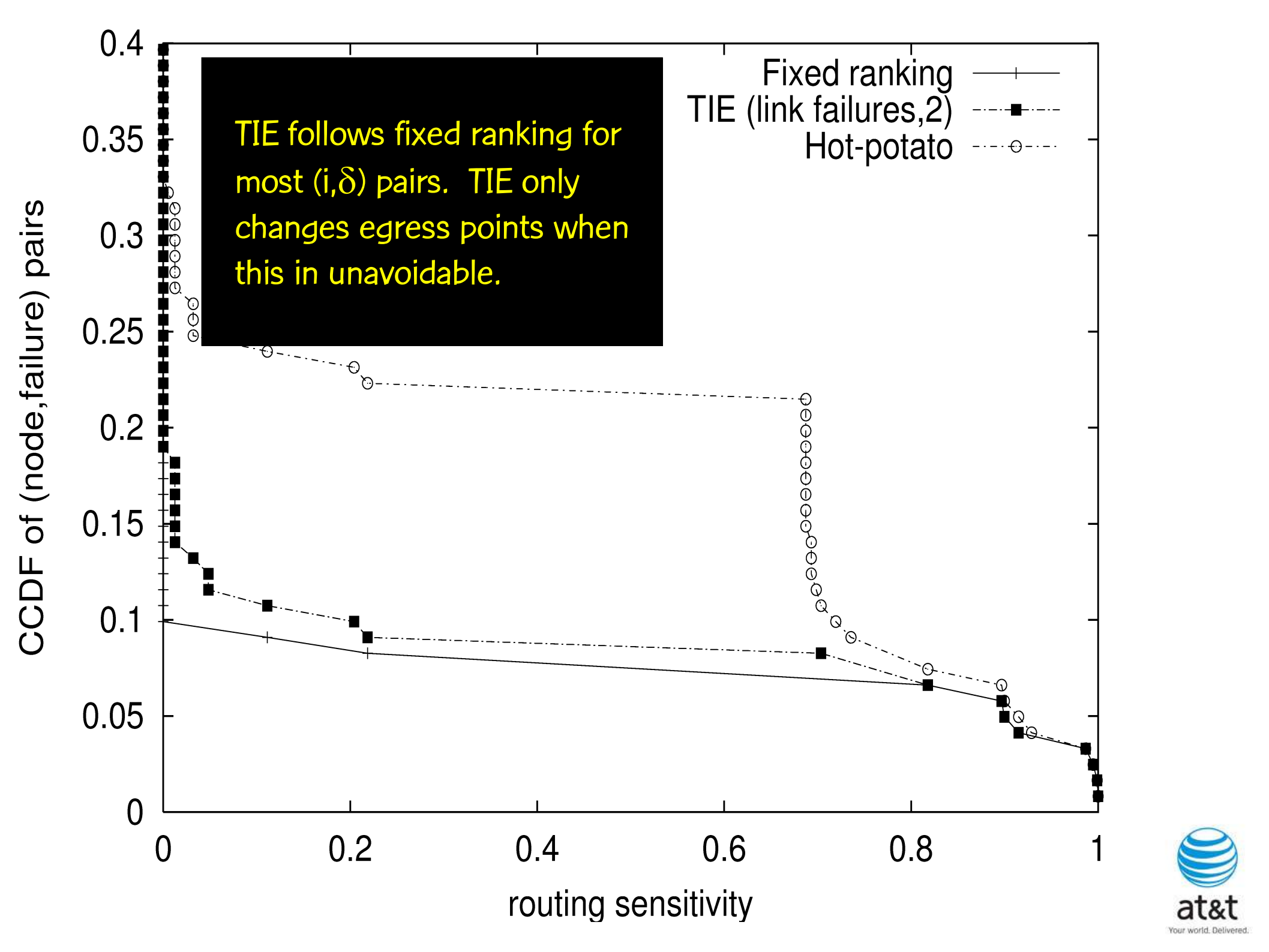
at&t
Your world. Delivered.



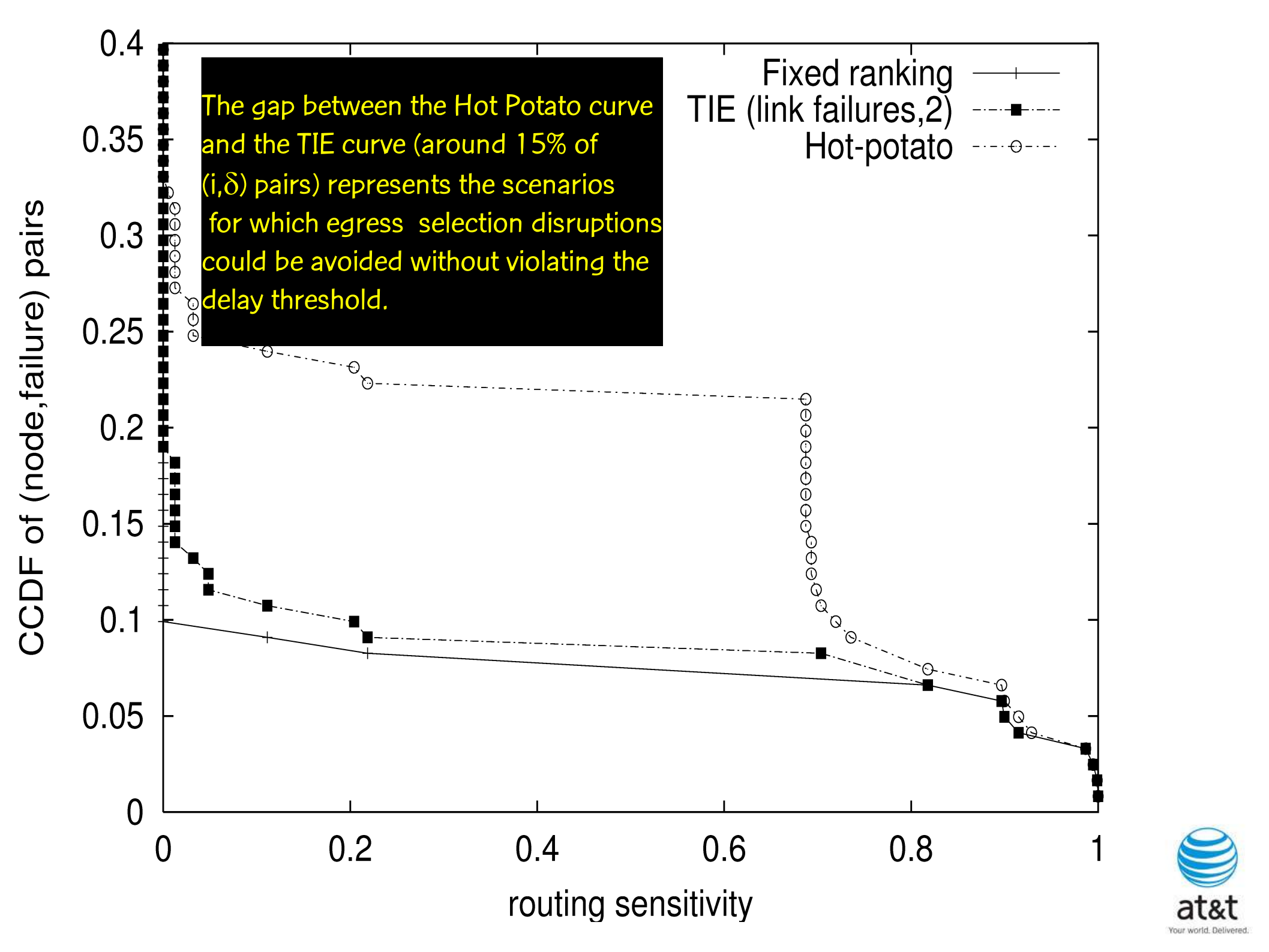
at&t
Your world. Delivered.



at&t
Your world. Delivered.



at&t
Your world. Delivered.



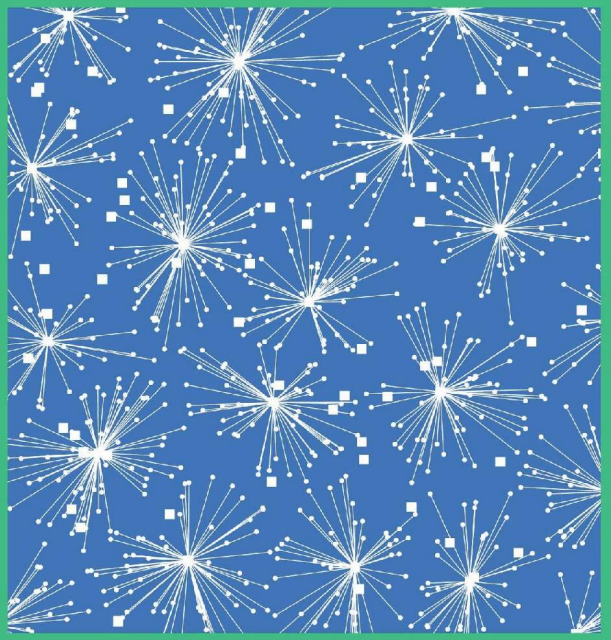
at&t
Your world. Delivered.

Conclusion

- We have just had a glimpse of a few examples of optimization problems that arise in IP networks.
- Interesting optimization problems arise in many fields of telecommunications.

Mauricio G. C. Resende
Panos M. Pardalos

Handbook of Optimization in Telecommunications



Springer

Published by Springer in
April 2006

1134 pages
37 chapters in five parts:

Optimization algorithms
Planning and design
Routing
Wireless
The web and beyond

Collaborators

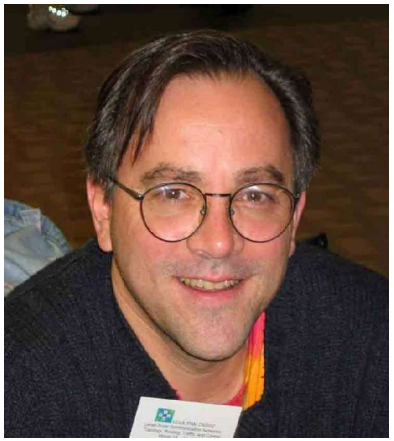
Diogo Andrade
Rutgers University

Mikkel Thorup
AT&T Labs Research

Panos Pardalos
University of Florida

Luciana Buriol
Federal University of
Rio Grande do Sul
Porto Alegre, Brazil

Celso Ribeiro
Federal Fluminense
University
Niterói , Brazil



Tim Griffin
University of Cambridge
Cambridge, U.K.

Jennifer Rexford
Princeton University

Renata Teixeira
Univ. Pierre et Marie Curie
Paris, France

at&t
Your world. Delivered.

The End

These slides and all papers cited in this talk
can be downloaded from my homepage:
http://www.research.att.com/~mgcr

google.com search key: Mauricio