The Necessity of Timekeeping in Adversarial Queueing

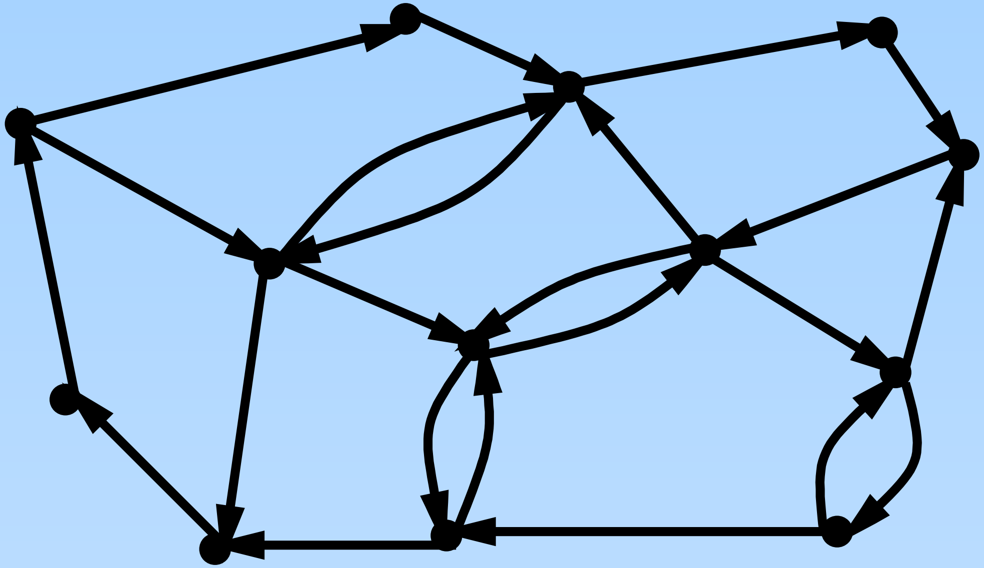
Maik Weinard
Institute of Computer Science
University of Frankfurt

13.5.05

Routing

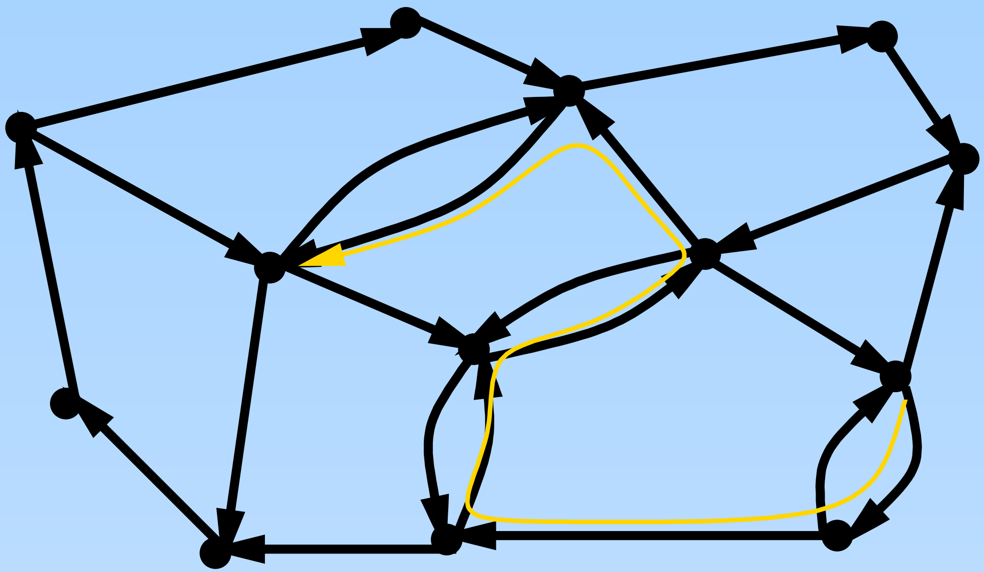
Routing

Graph G

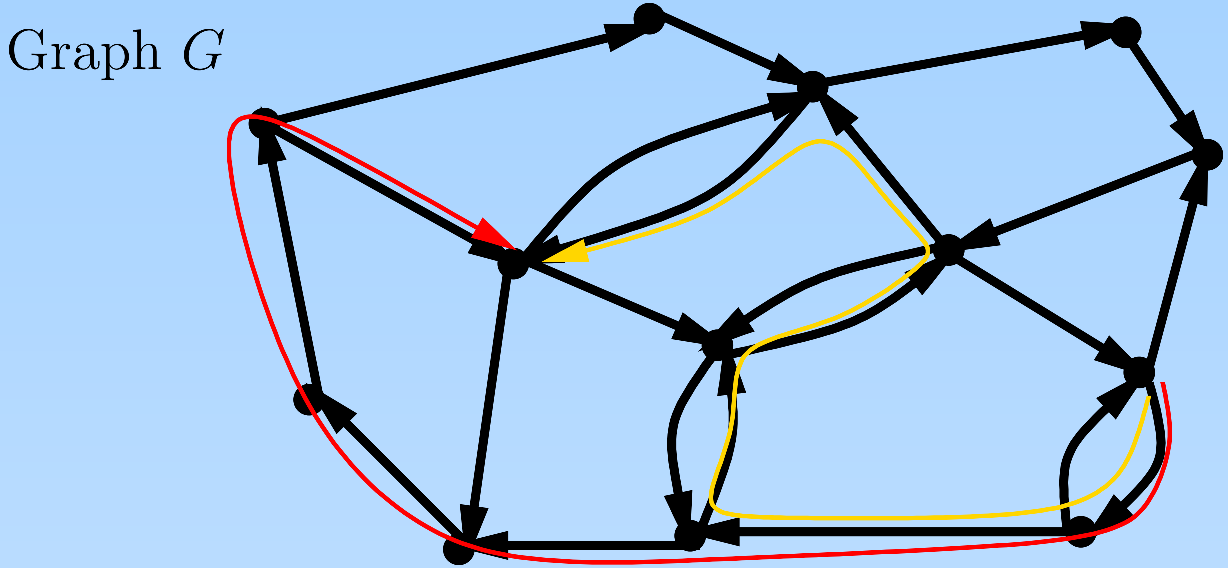


Routing

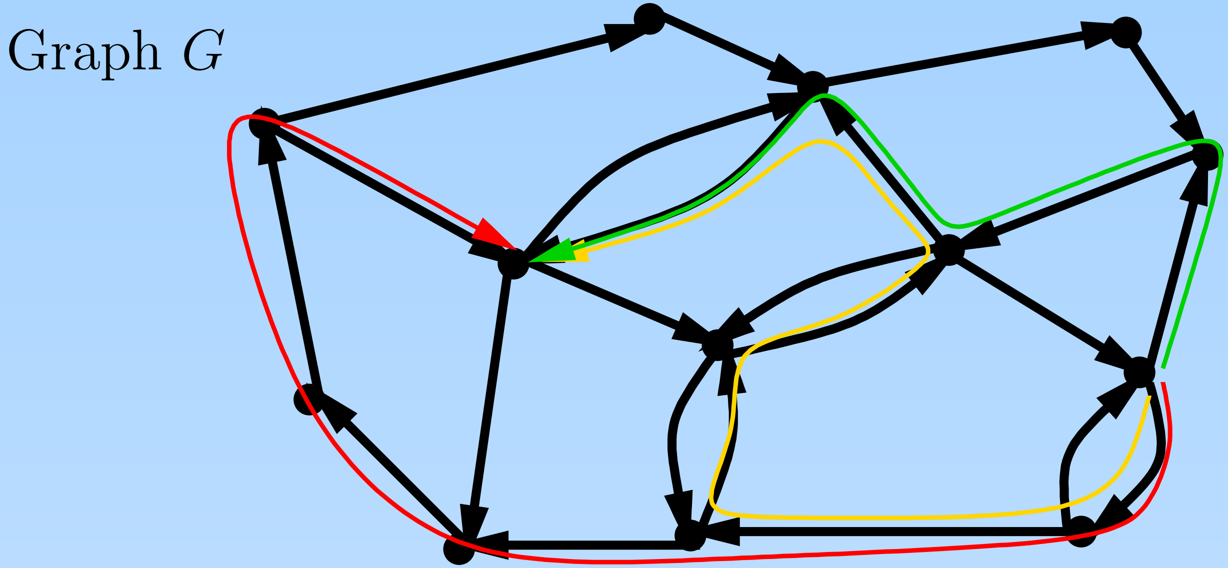
Graph G



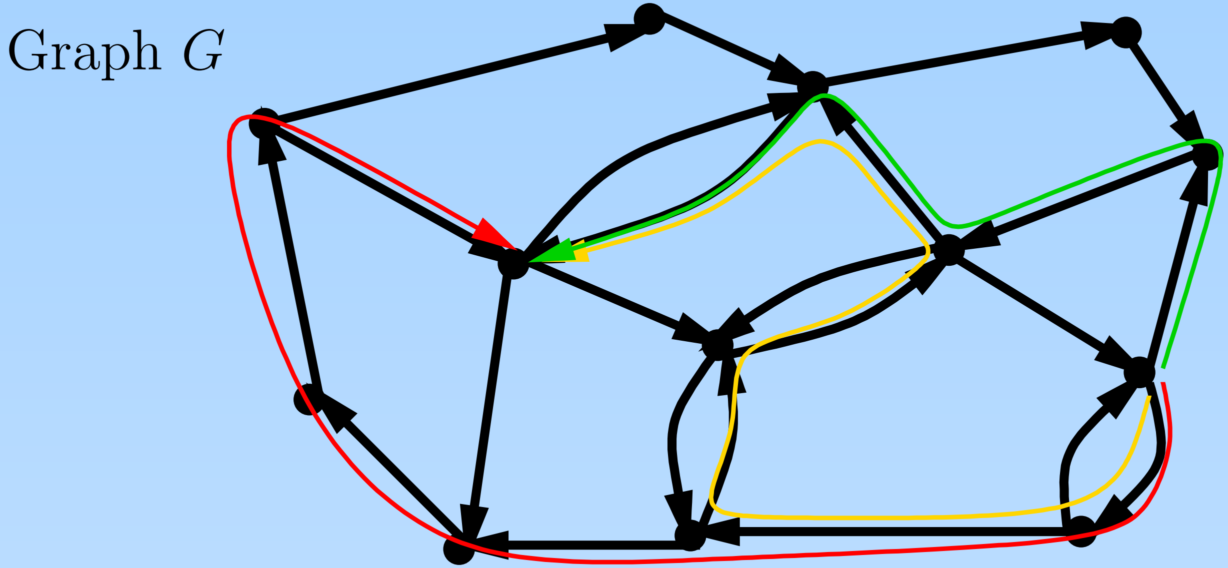
Routing



Routing

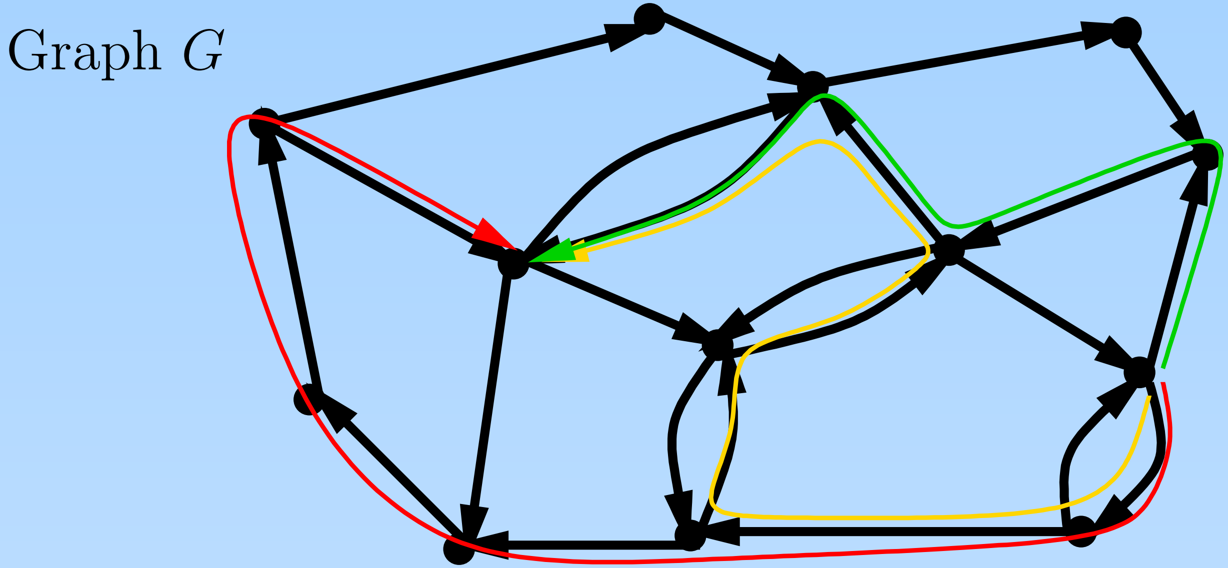


Routing



Routing policies assign simple paths.

Routing



Routing policies assign simple paths.

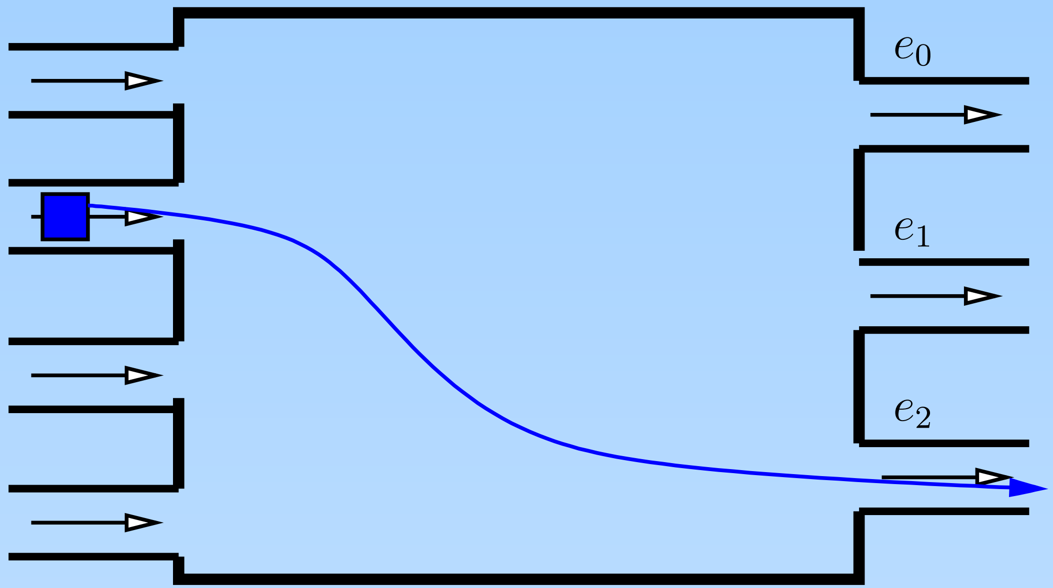
We assume unit capacity and unit speed edges.

Queueing

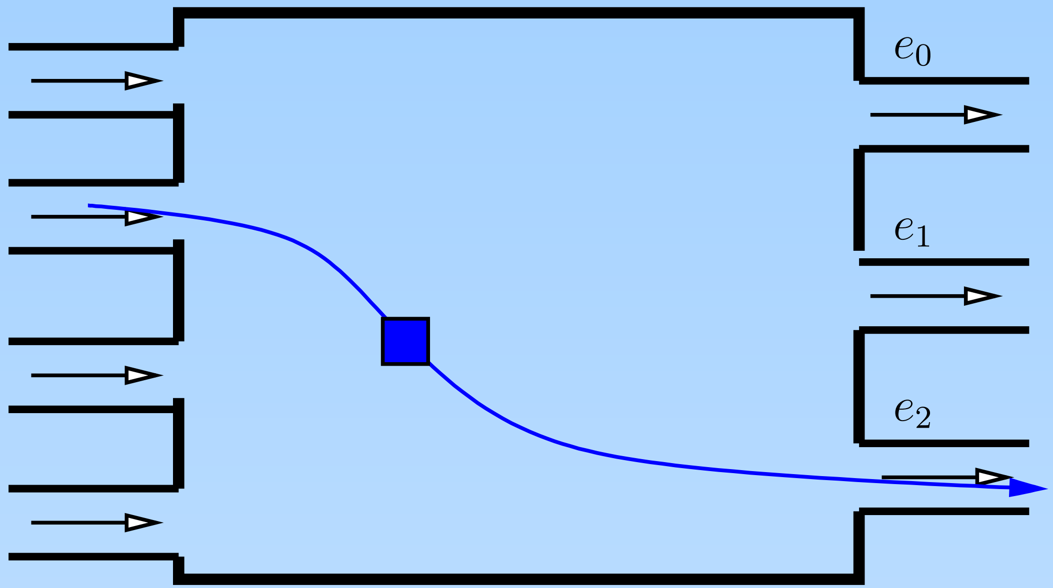
Queueing

Queueing

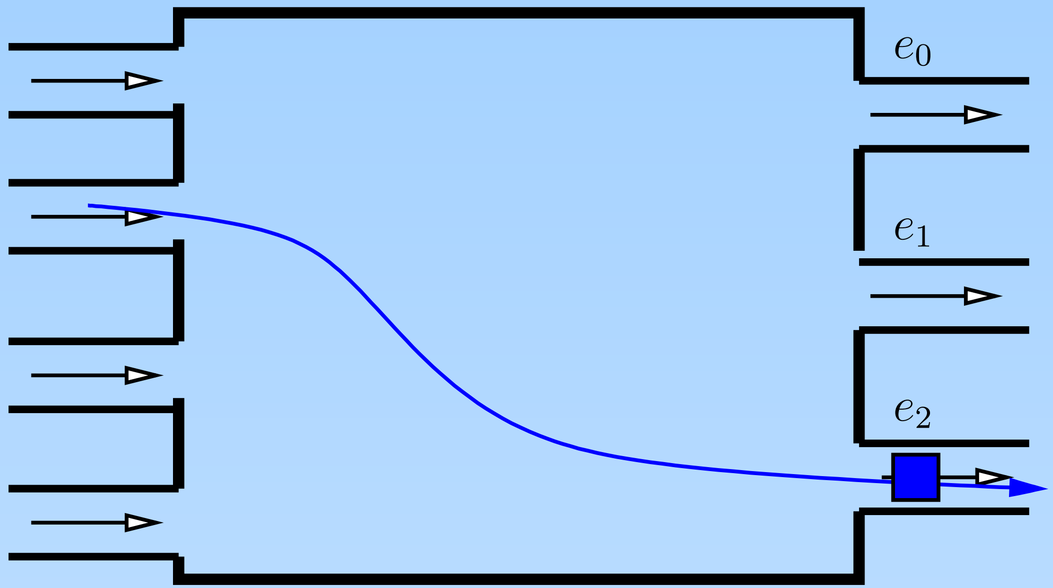
Queueing



Queueing

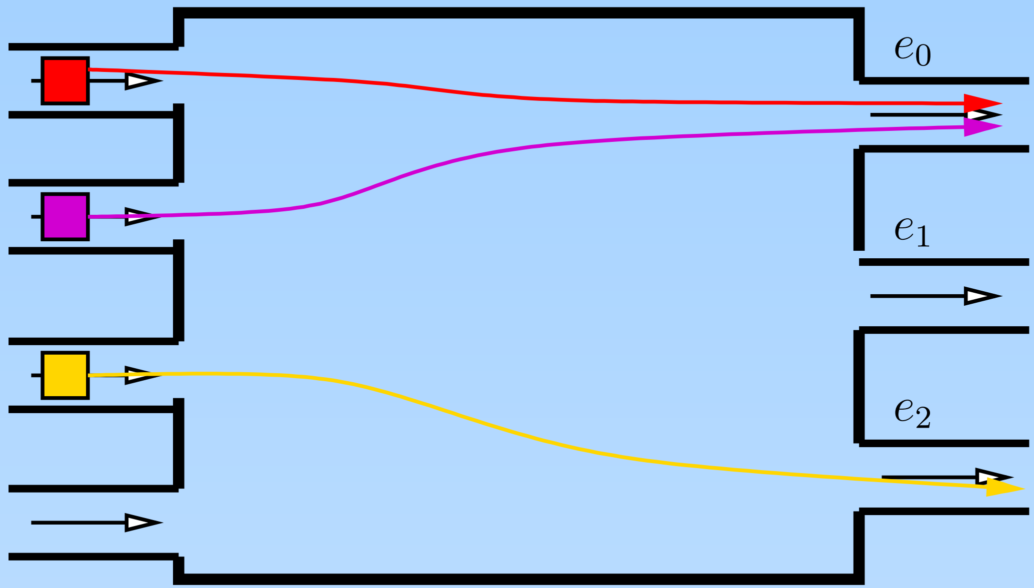


Queueing

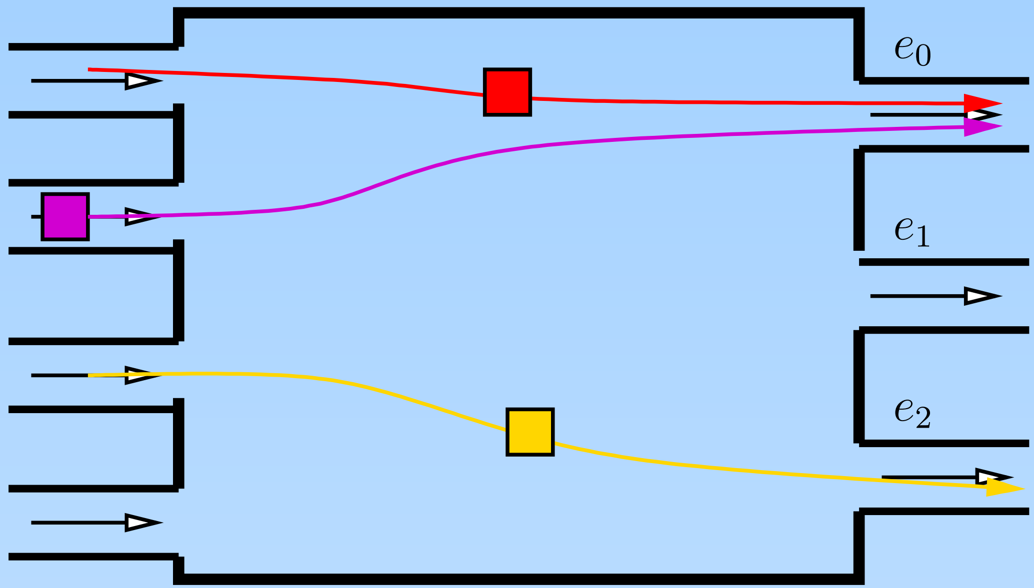


Queueing

Queueing

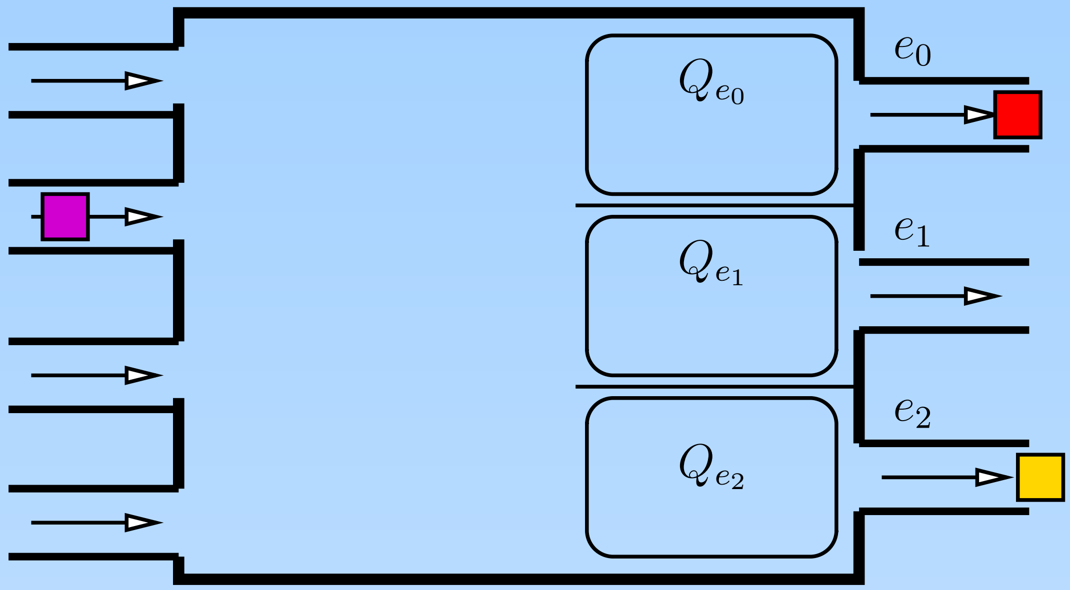


Queueing

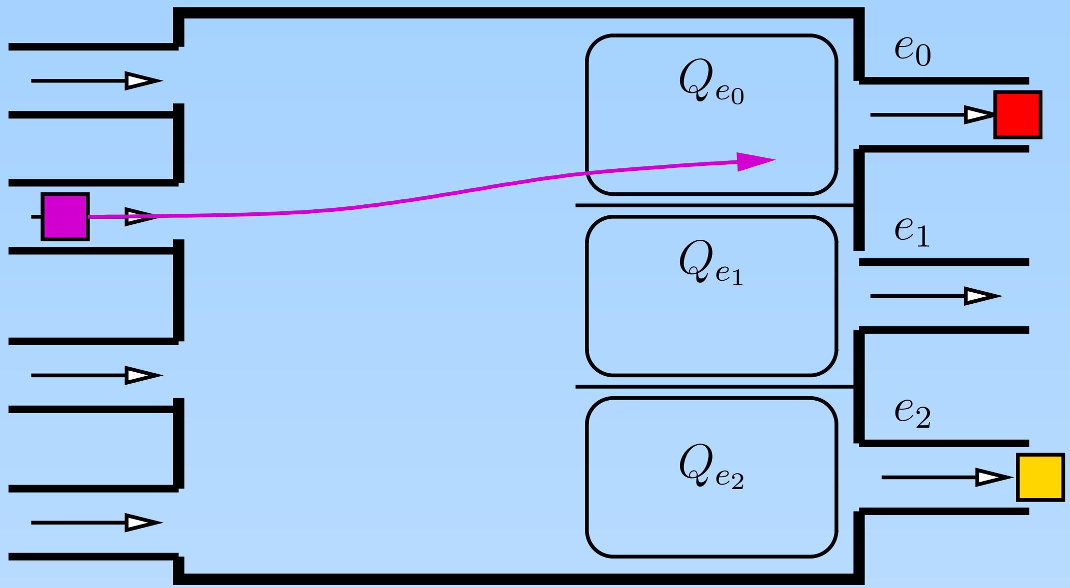


Queueing

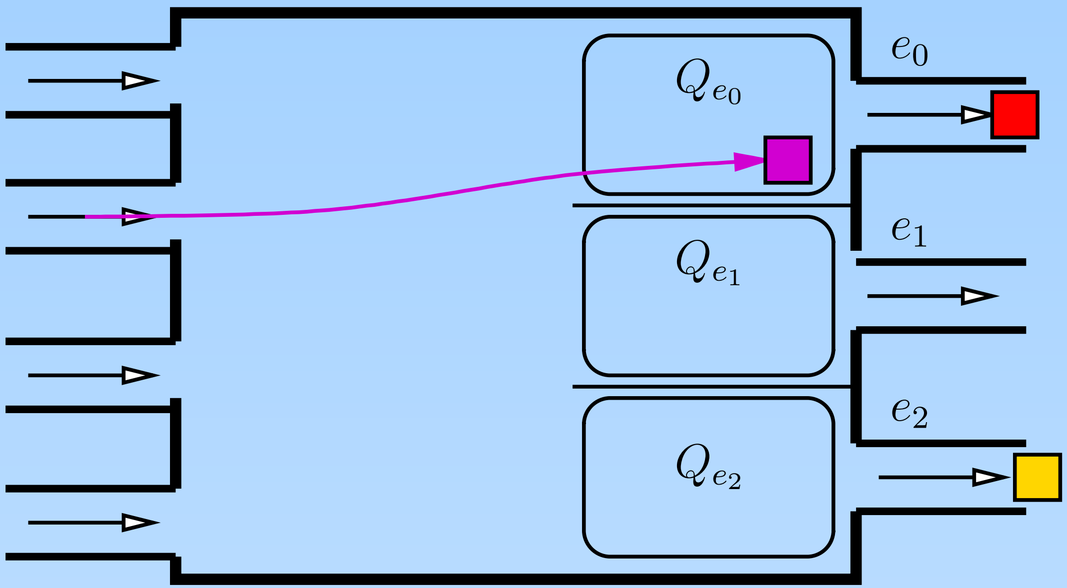
Queueing



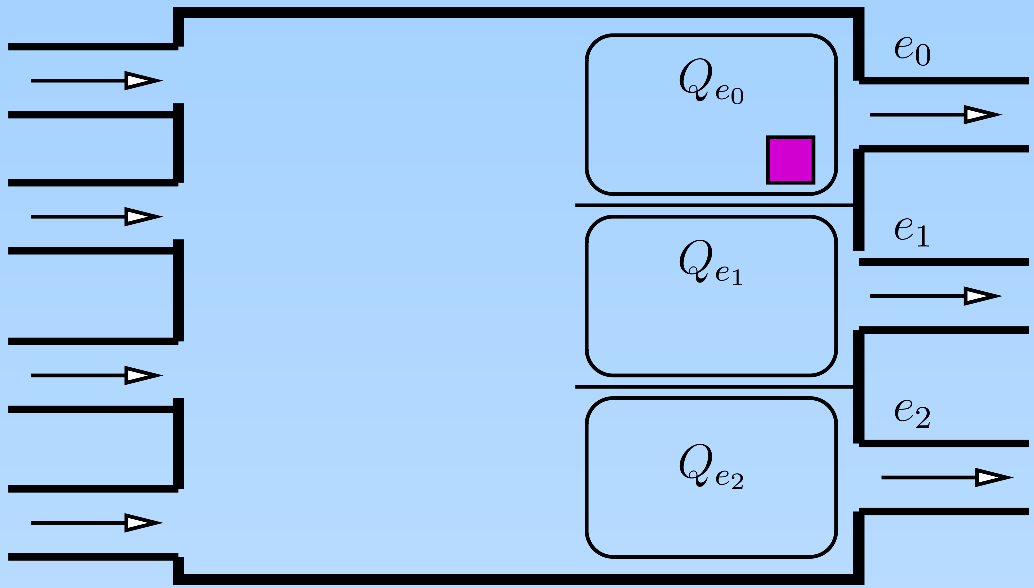
Queueing



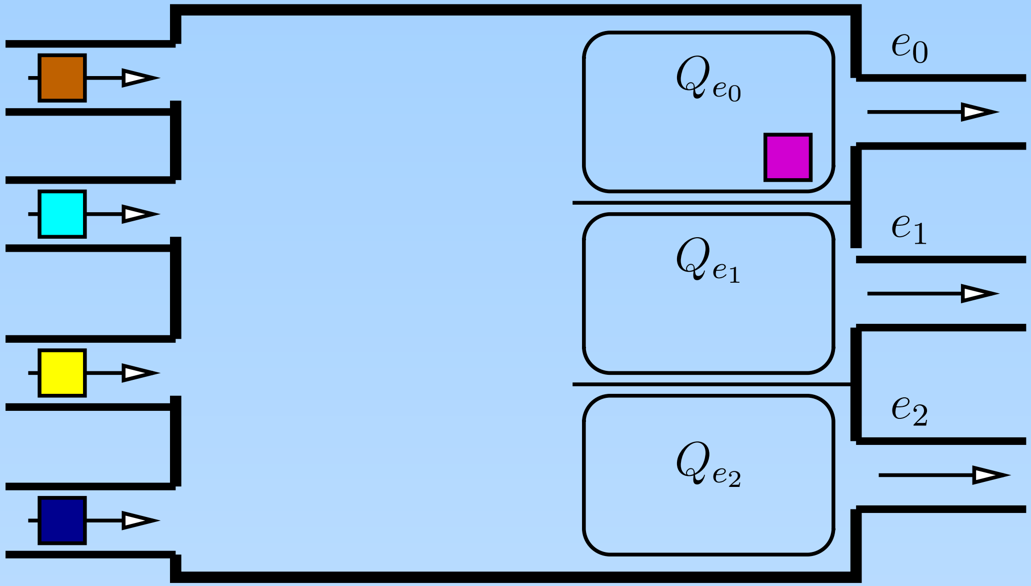
Queueing



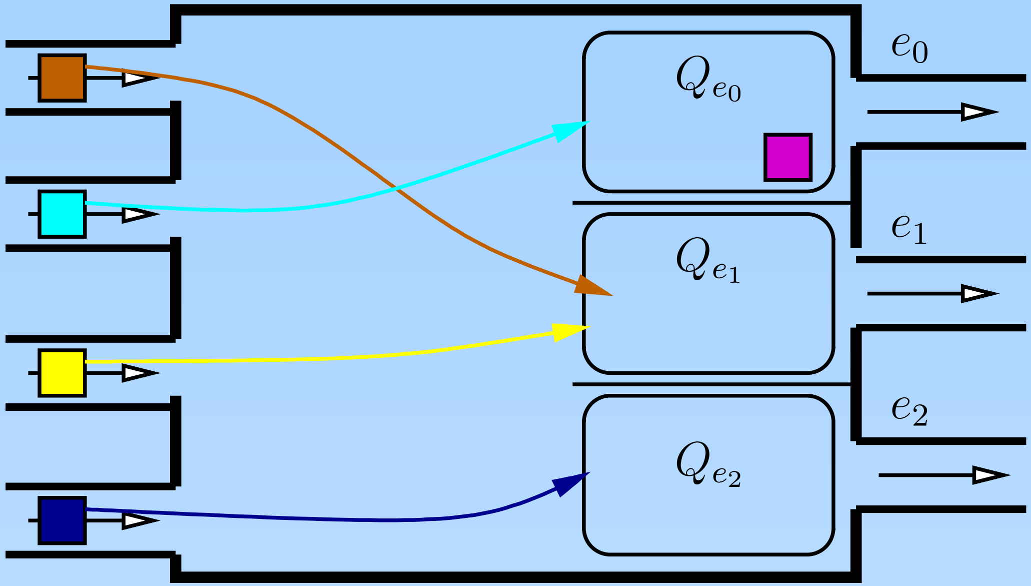
Queueing



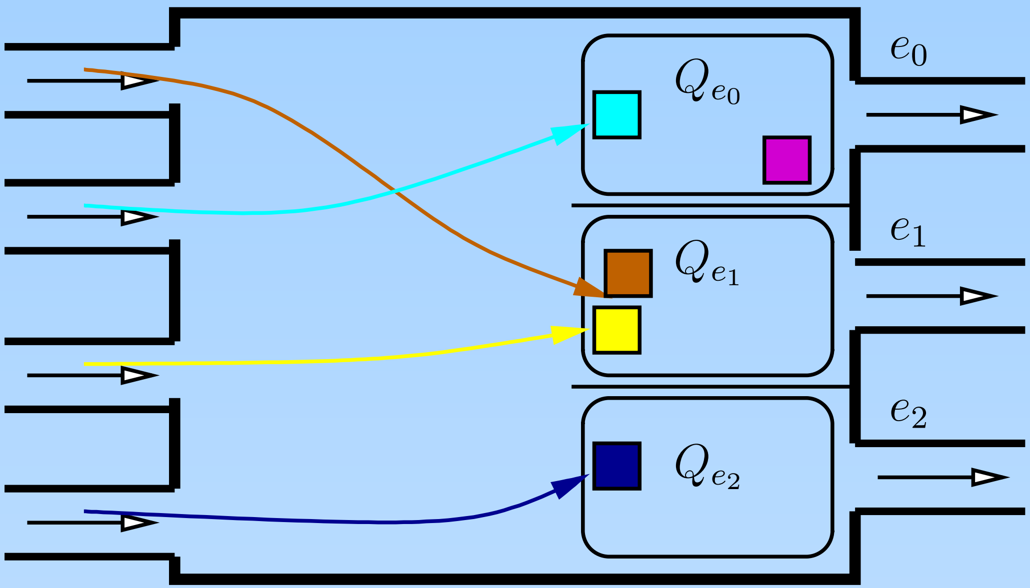
Queueing



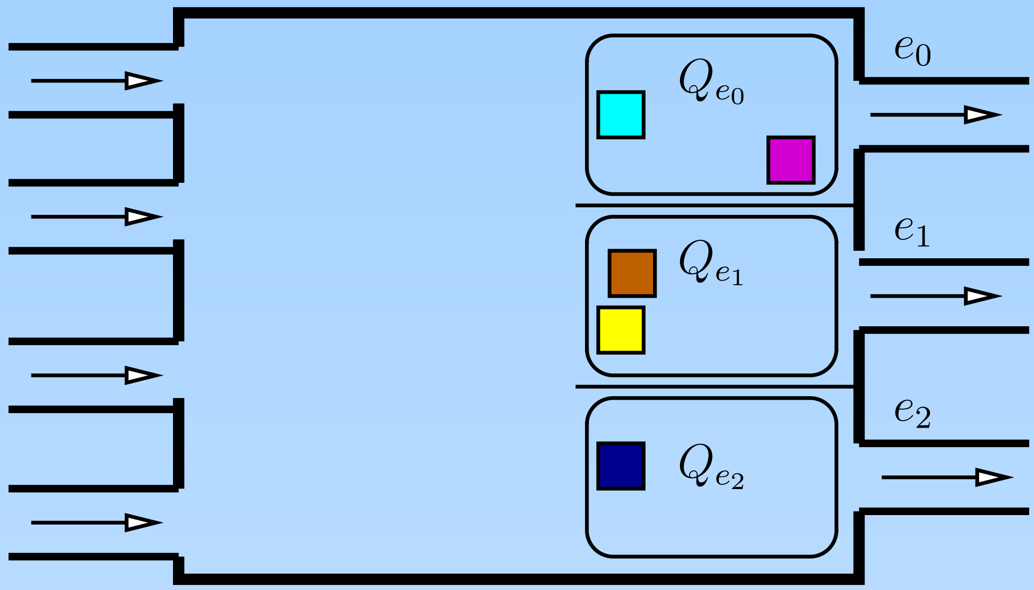
Queueing



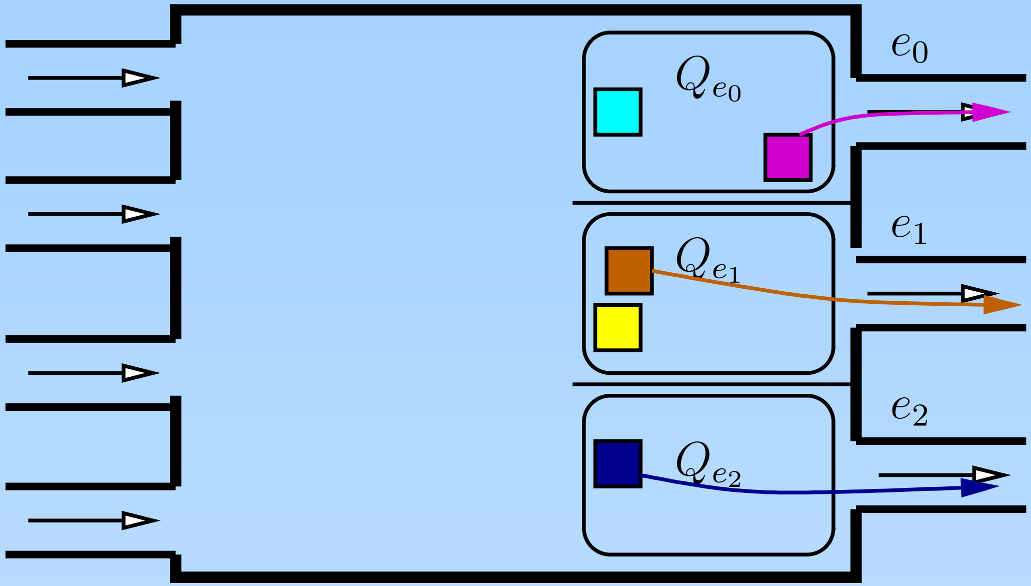
Queueing



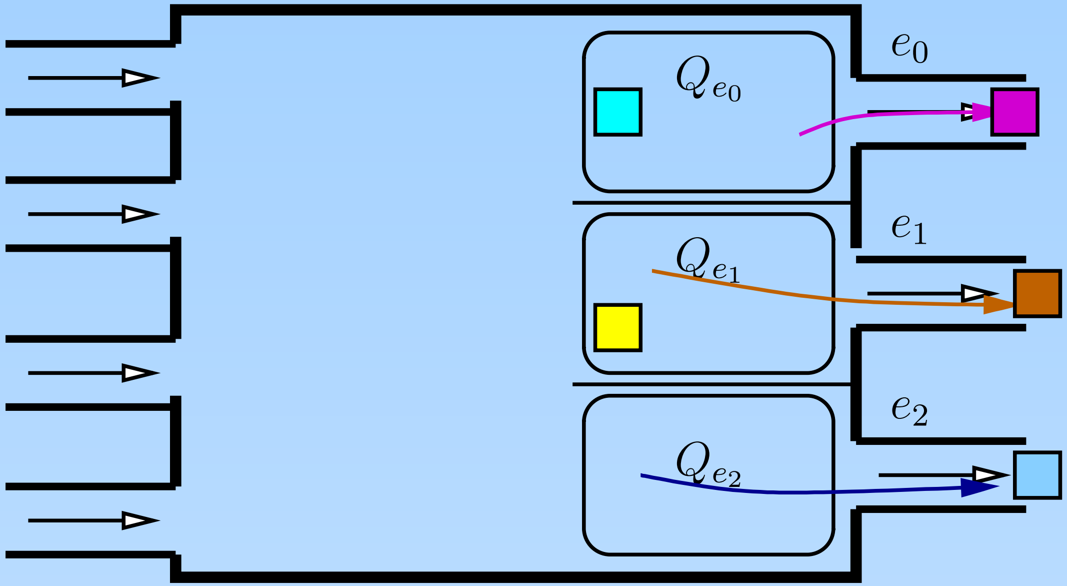
Queueing



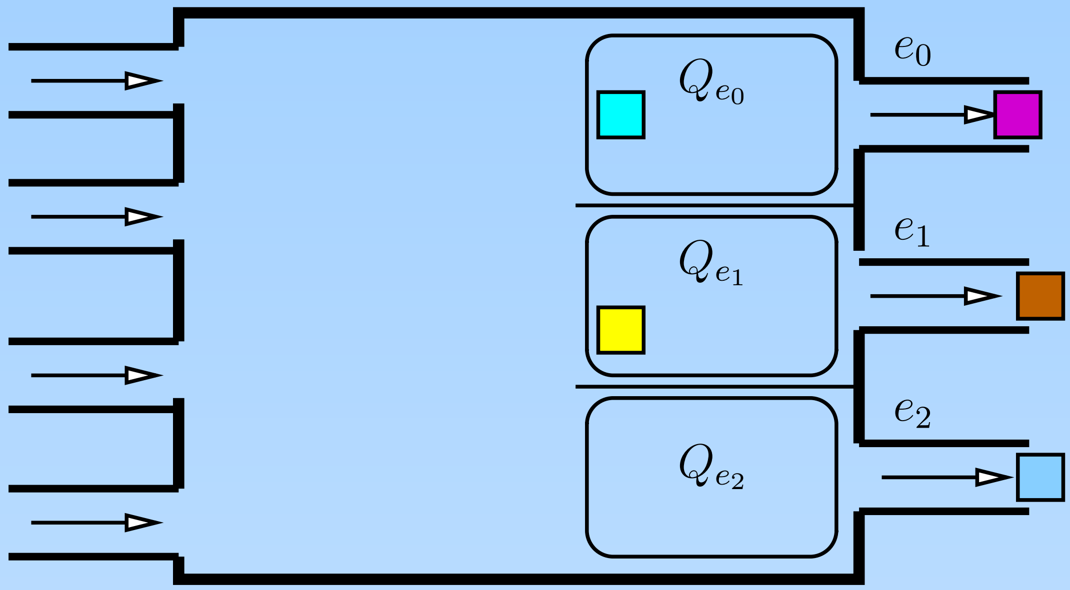
Queueing



Queueing



Queueing



Queueing strategies decide which packet may proceed.

Queueing Policies

Queueing Policies

- Work in an online scenario.

Queueing Policies

- Work in an online scenario.

- Work with local information only.

Queueing Policies

- Work in an online scenario.
- Work with local information only.
- Are supposed to keep the total traffic small and delays short.

Queueing Policies

- Work in an online scenario.
- Work with local information only.
- Are supposed to keep the total traffic small and delays short.
- Examples:

Queueing Policies

- Work in an online scenario.
- Work with local information only.
- Are supposed to keep the total traffic small and delays short.
- Examples:
  - First-In-First-Out (FIFO)

Queueing Policies

- Work in an online scenario.
- Work with local information only.
- Are supposed to keep the total traffic small and delays short.
- Examples:
  - First-In-First-Out (FIFO)
  - Longest-In-System (LIS)

Queueing Policies

- Work in an online scenario.
- Work with local information only.
- Are supposed to keep the total traffic small and delays short.
- Examples:
  - First-In-First-Out (FIFO)
  - Longest-In-System (LIS)
  - Nearest-To-Source (NTS)

Queueing Policies

- Work in an online scenario.
- Work with local information only.
- Are supposed to keep the total traffic small and delays short.
- Examples:
  - First-In-First-Out (FIFO)
  - Longest-In-System (LIS)
  - Nearest-To-Source (NTS)
  - Shortest-In-System (SIS)

Adversarial Queueing Theory

Adversarial Queueing Theory

- Designed to reveal the quality of queueing policies.

Adversarial Queueing Theory

- Designed to reveal the quality of queueing policies.
- An Adversary decides

Adversarial Queueing Theory

- Designed to reveal the quality of queueing policies.
- An Adversary decides
  - when and where packets are inserted,

Adversarial Queueing Theory

- Designed to reveal the quality of queueing policies.
- An Adversary decides
  - when and where packets are inserted,
  - whereto each packet is to be delivered,

Adversarial Queueing Theory

- Designed to reveal the quality of queueing policies.
- An Adversary decides
  - when and where packets are inserted,
  - whereto each packet is to be delivered,
  - along which path it is to be routed.

Adversarial Queueing Theory

- Designed to reveal the quality of queueing policies.
- An Adversary decides
  - when and where packets are inserted,
  - whereto each packet is to be delivered,
  - along which path it is to be routed.
- **Only restriction:**

Adversarial Queueing Theory

- Designed to reveal the quality of queueing policies.
- An Adversary decides
  - when and where packets are inserted,
  - whereto each packet is to be delivered,
  - along which path it is to be routed.
- **Only restriction:** Adversary may not straightforwardly overload edges.

(r,b)-Adversaries

(r,b)-Adversaries

An adversary is a (r, b) adversary if for every edge e and during every interval of t consecutive steps no more than

(r,b)-Adversaries

An adversary is a (r, b) adversary if for every edge e and during every interval of t consecutive steps no more than

$$r \cdot t + b$$

(r,b)-Adversaries

An adversary is a (r, b) adversary if for every edge e and during every interval of t consecutive steps no more than

$$r \cdot t + b$$

packets are inserted that require edge e.

(r,b)-Adversaries

An adversary is a (r, b) adversary if for every edge e and during every interval of t consecutive steps no more than

$$r \cdot t + b$$

packets are inserted that require edge e.

- r is the rate.

(r,b)-Adversaries

An adversary is a (r, b) adversary if for every edge e and during every interval of t consecutive steps no more than

$$r \cdot t + b$$

packets are inserted that require edge e.

- r is the rate. We demand $r \leq 1$.

(r,b)-Adversaries

An adversary is a (r, b) adversary if for every edge e and during every interval of t consecutive steps no more than

$$r \cdot t + b$$

packets are inserted that require edge e.

- r is the rate. We demand $r \leq 1$.
- b is the burstiness.

Stability

Stability

A queueing policy is *stable* on a graph G against (r, b) adversaries, if for every sequence of insertions into G by an (r, b)-adversary the number of packets in G is upper bounded by $c(r, b, G)$.

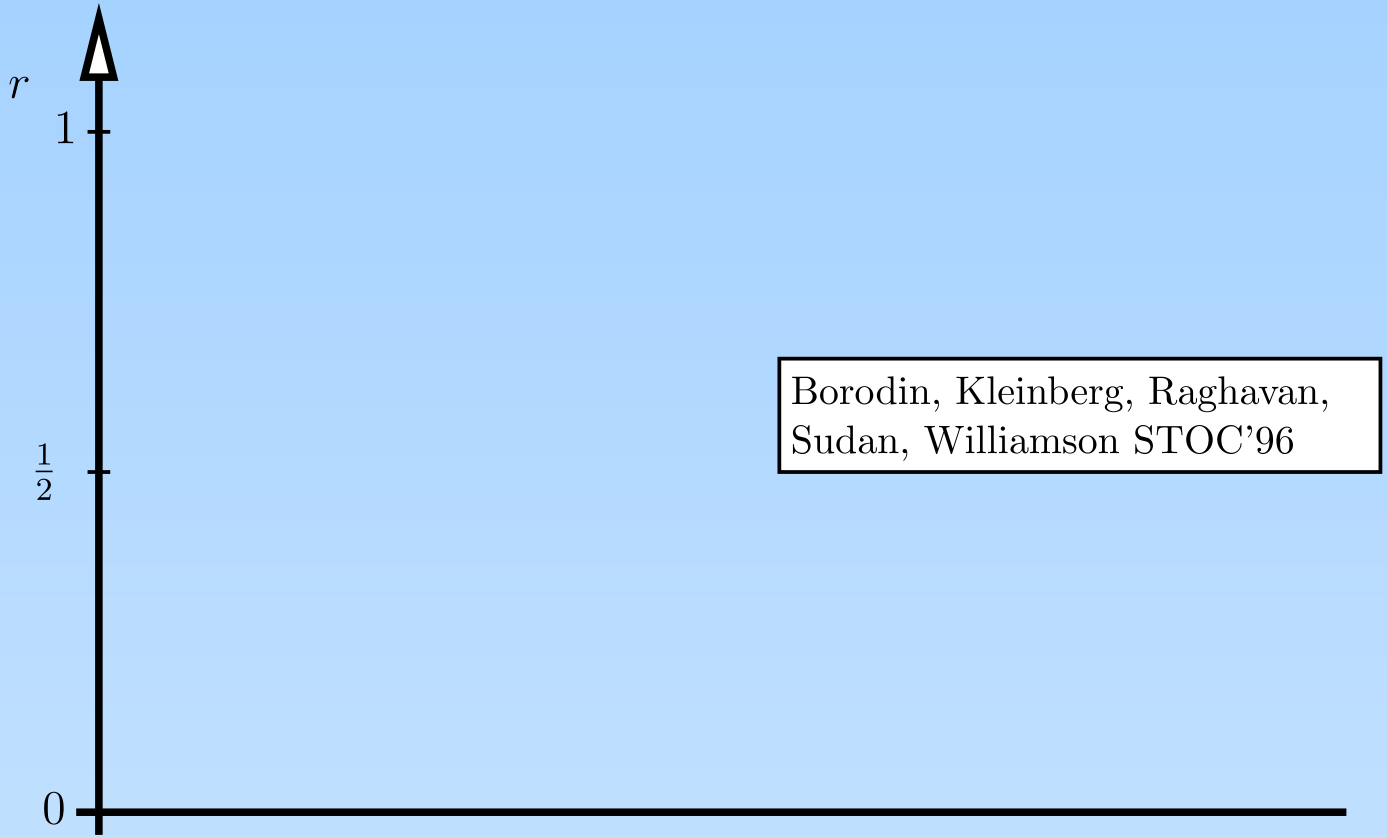
Stability

A queueing policy is *stable* on a graph G against (r, b) adversaries, if for every sequence of insertions into G by an (r, b)-adversary the number of packets in G is upper bounded by $c(r, b, G)$.

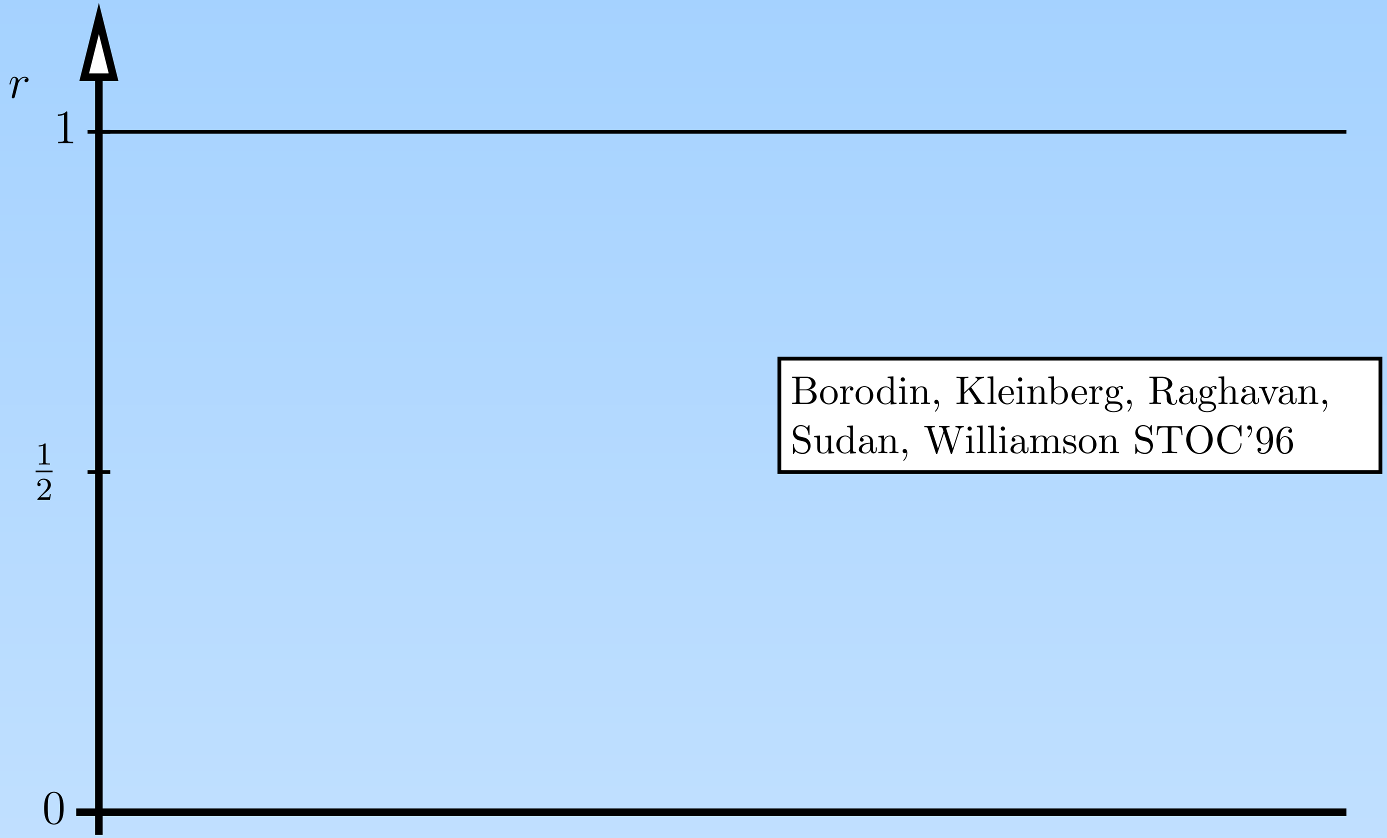
A queueing strategy is *universally stable* if it is stable on every graph against every (r, b)-adversary with $r < 1$.

Previous Work

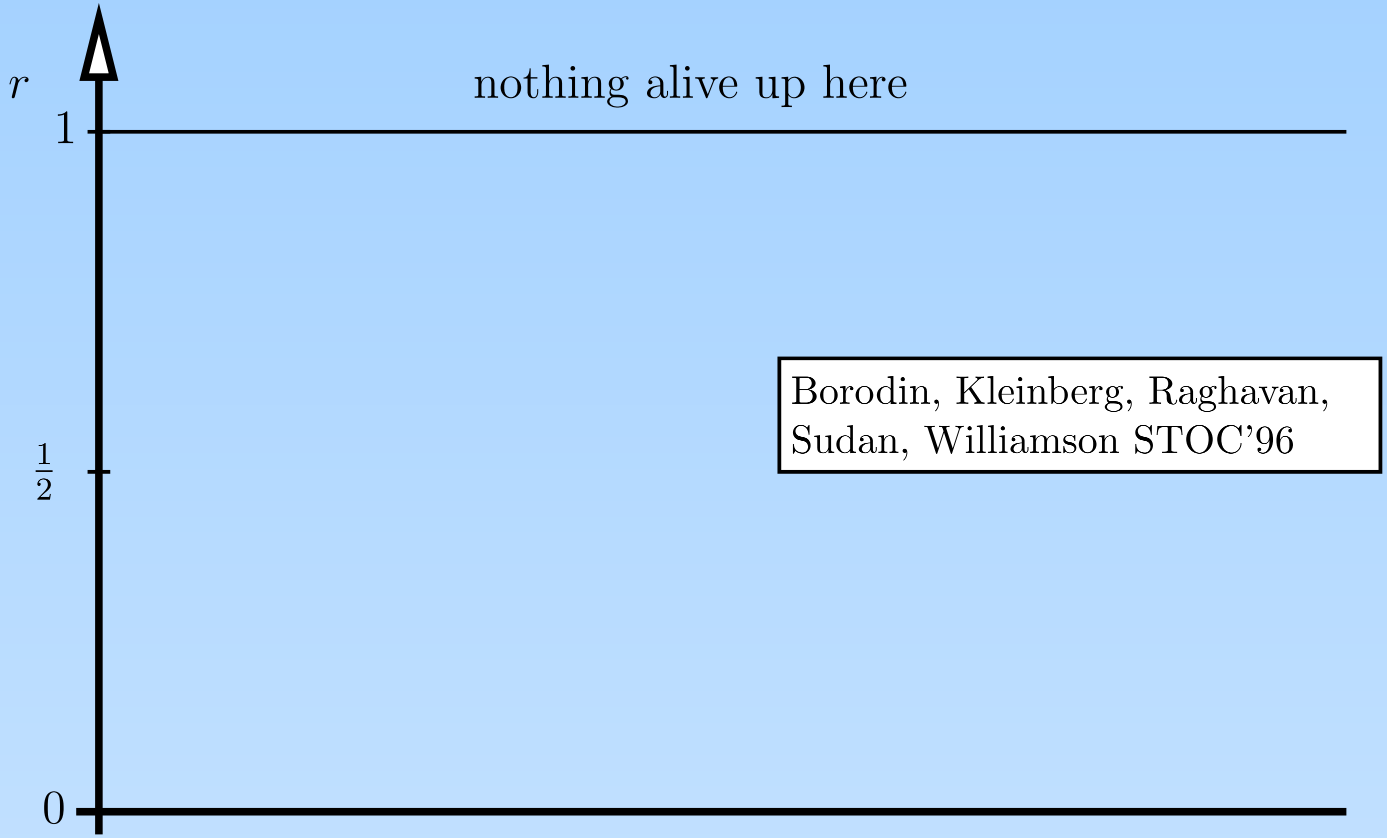
Previous Work



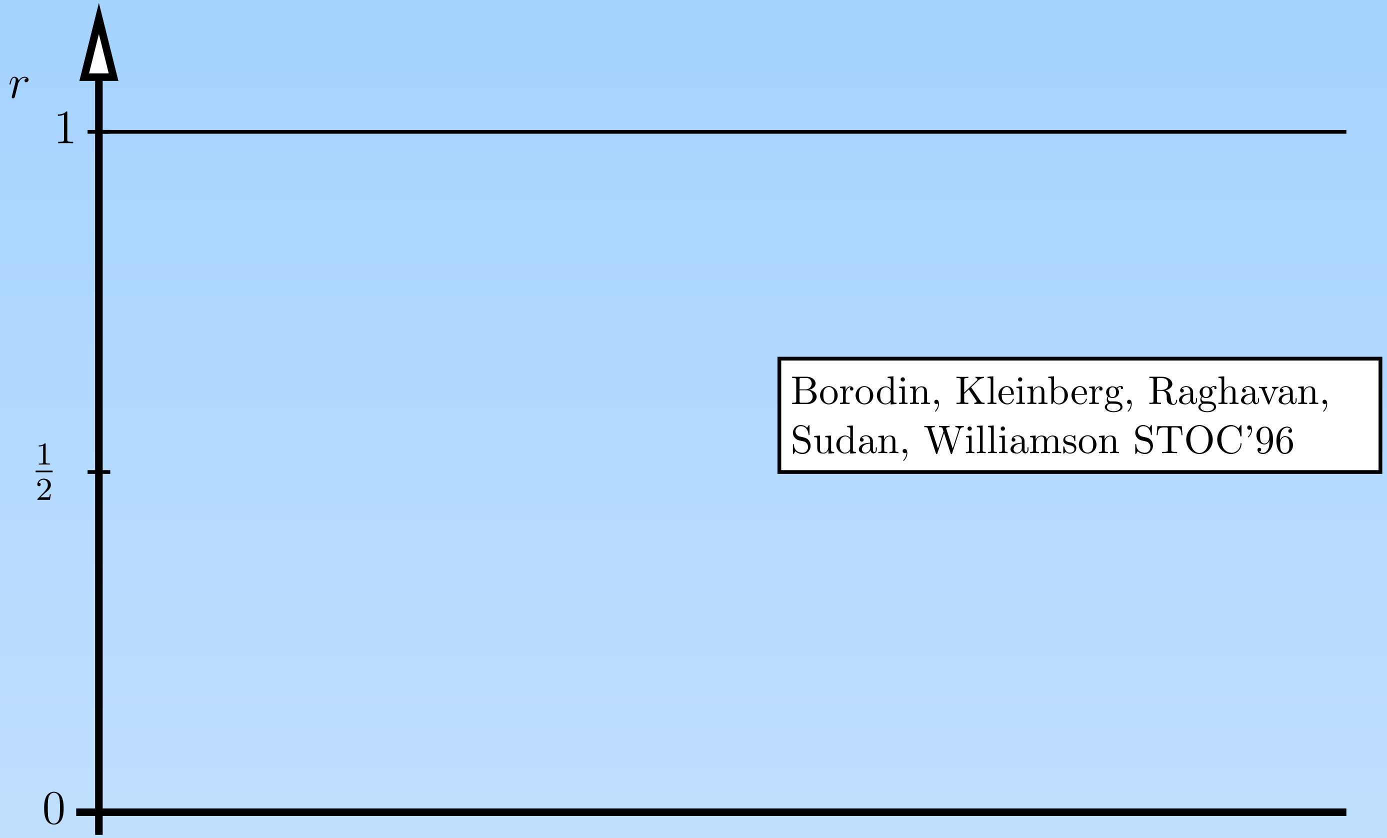
Previous Work



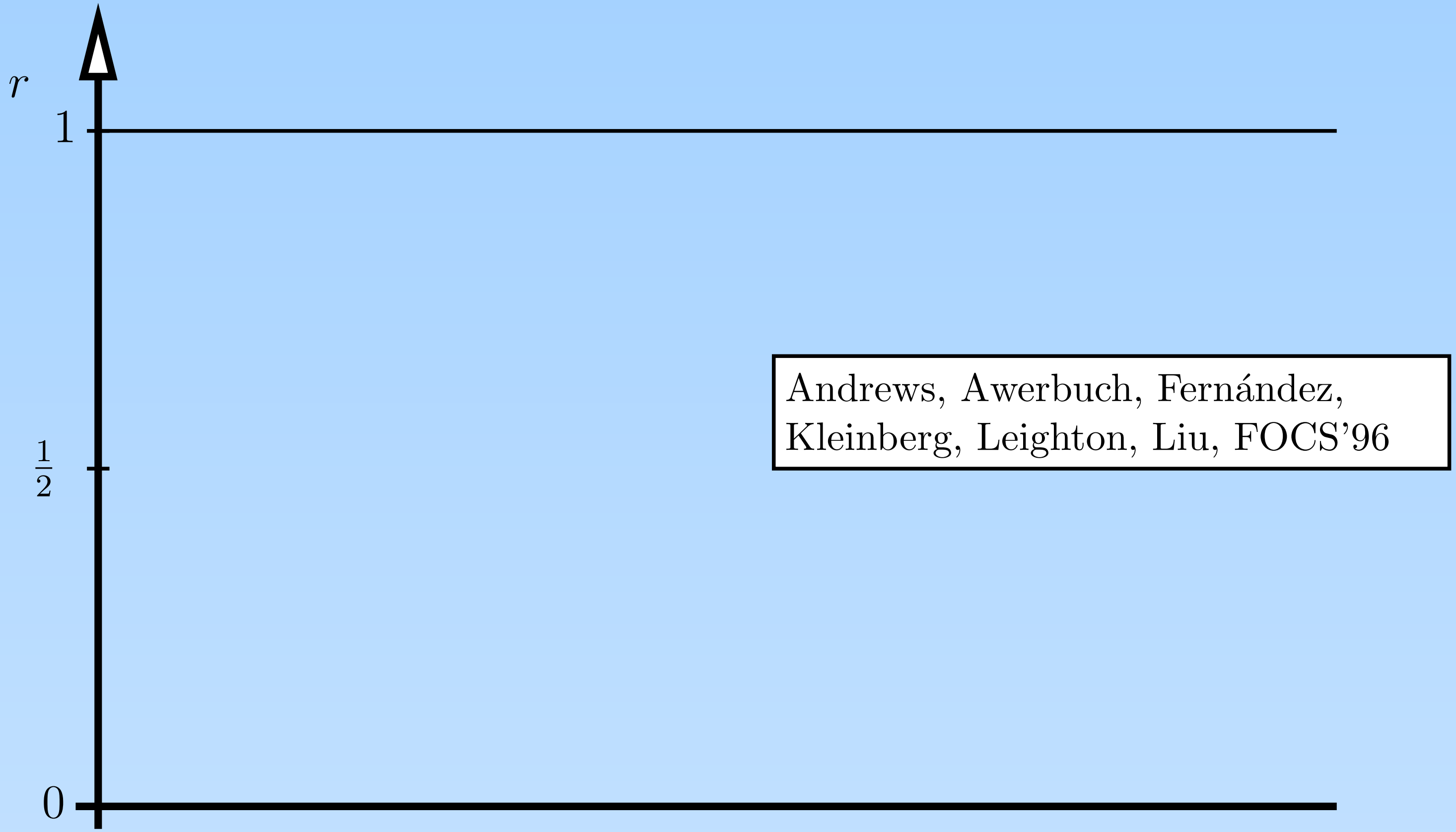
Previous Work



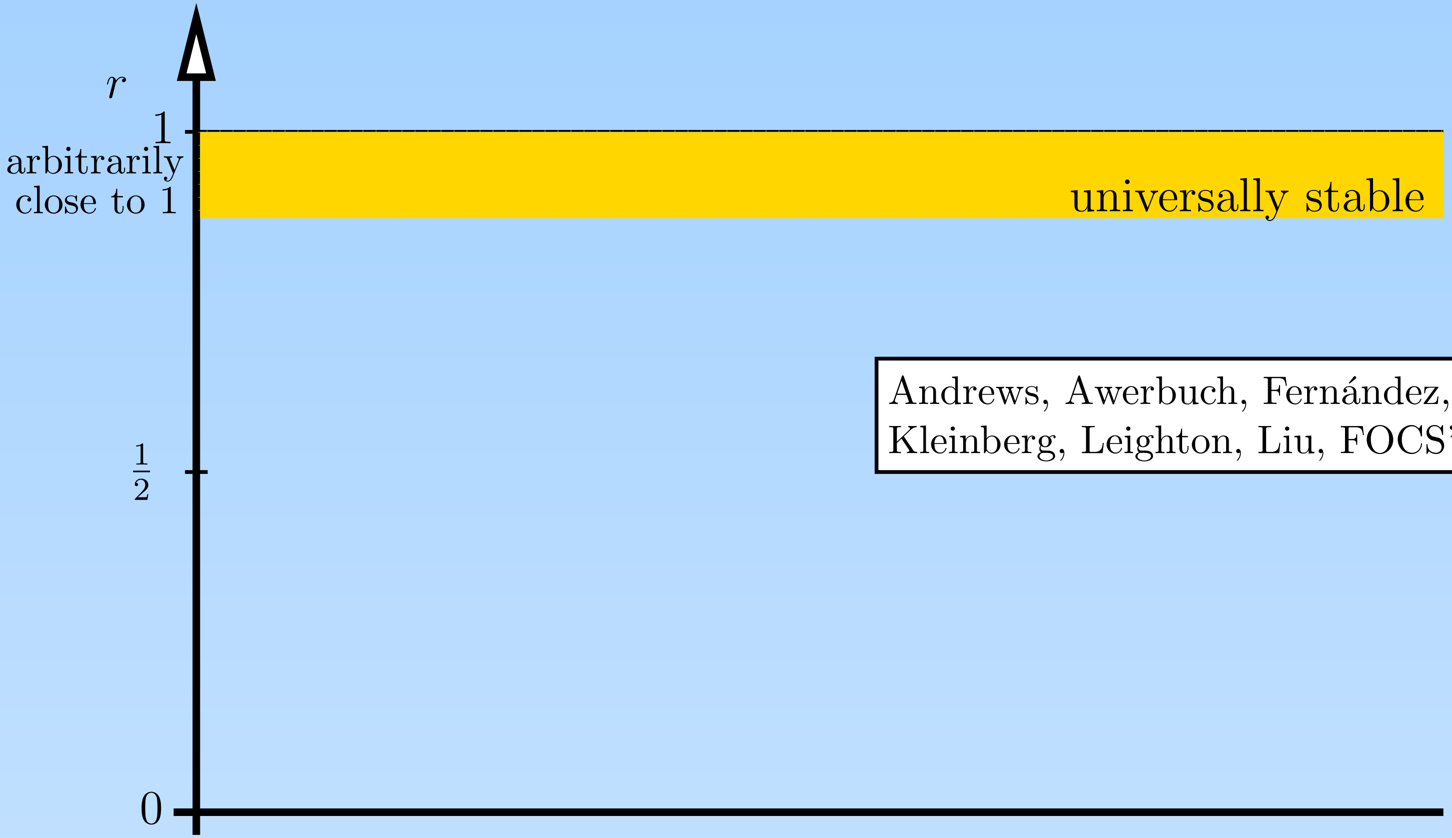
Previous Work



Previous Work

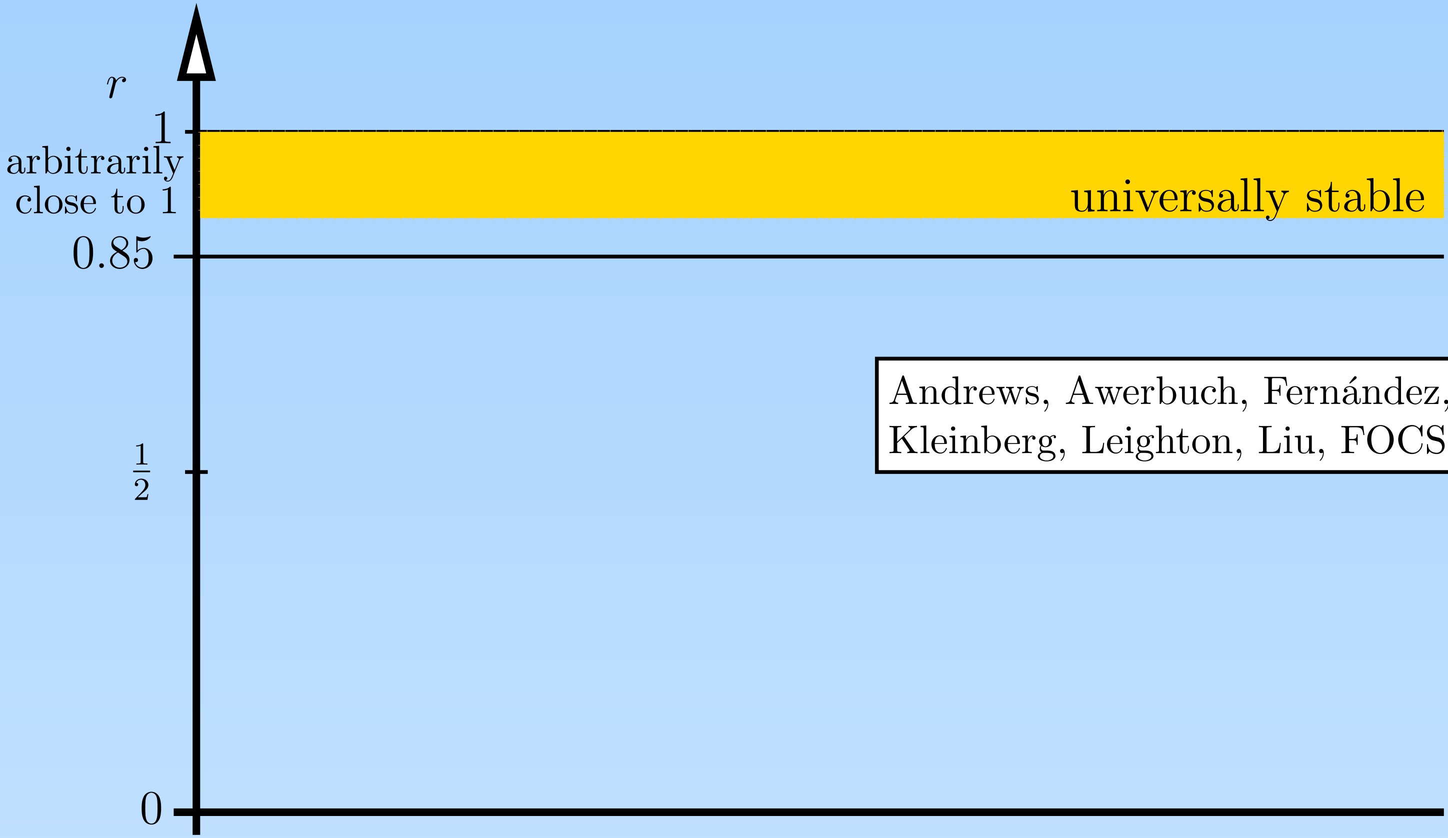


Previous Work



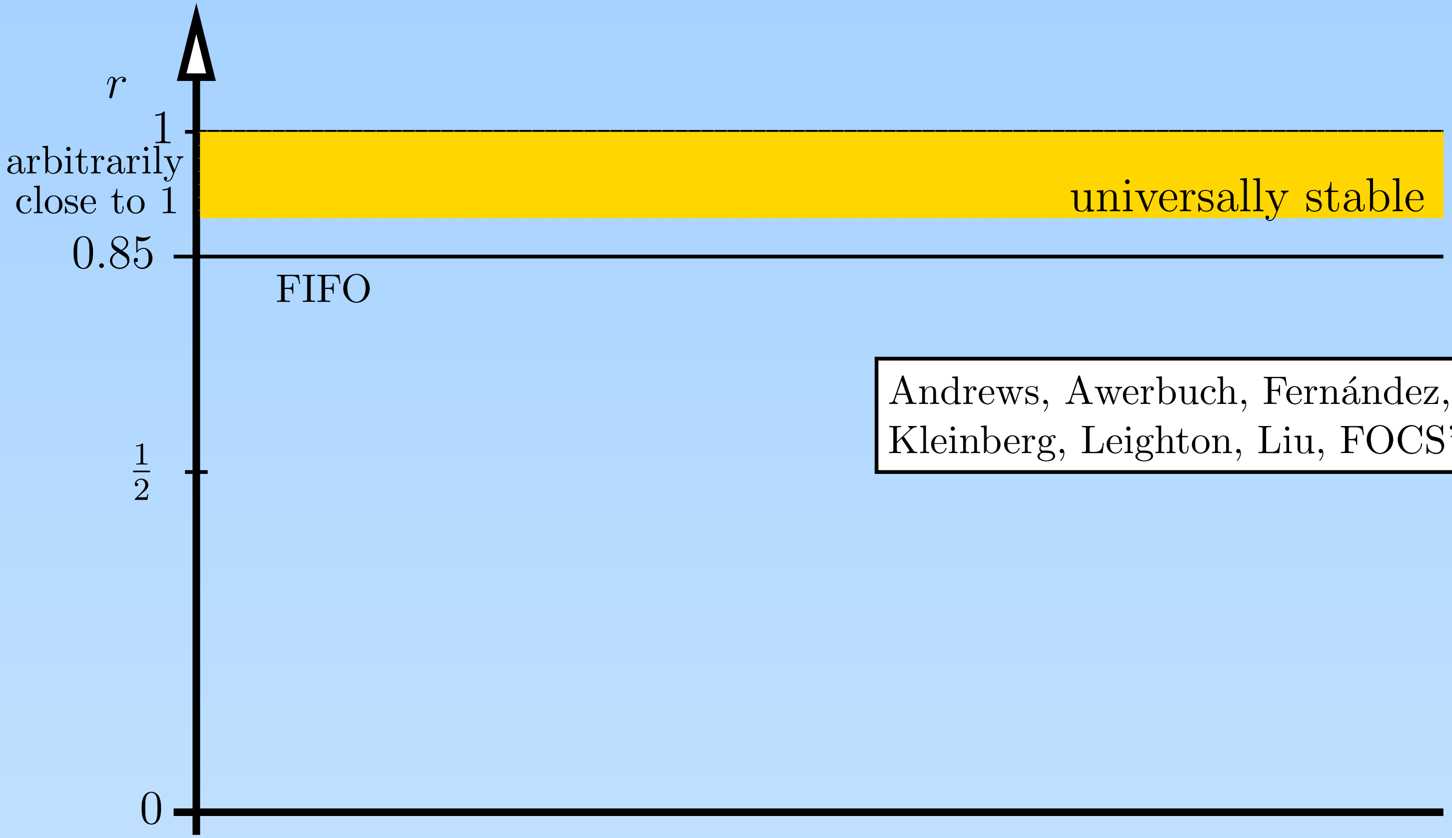
Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work



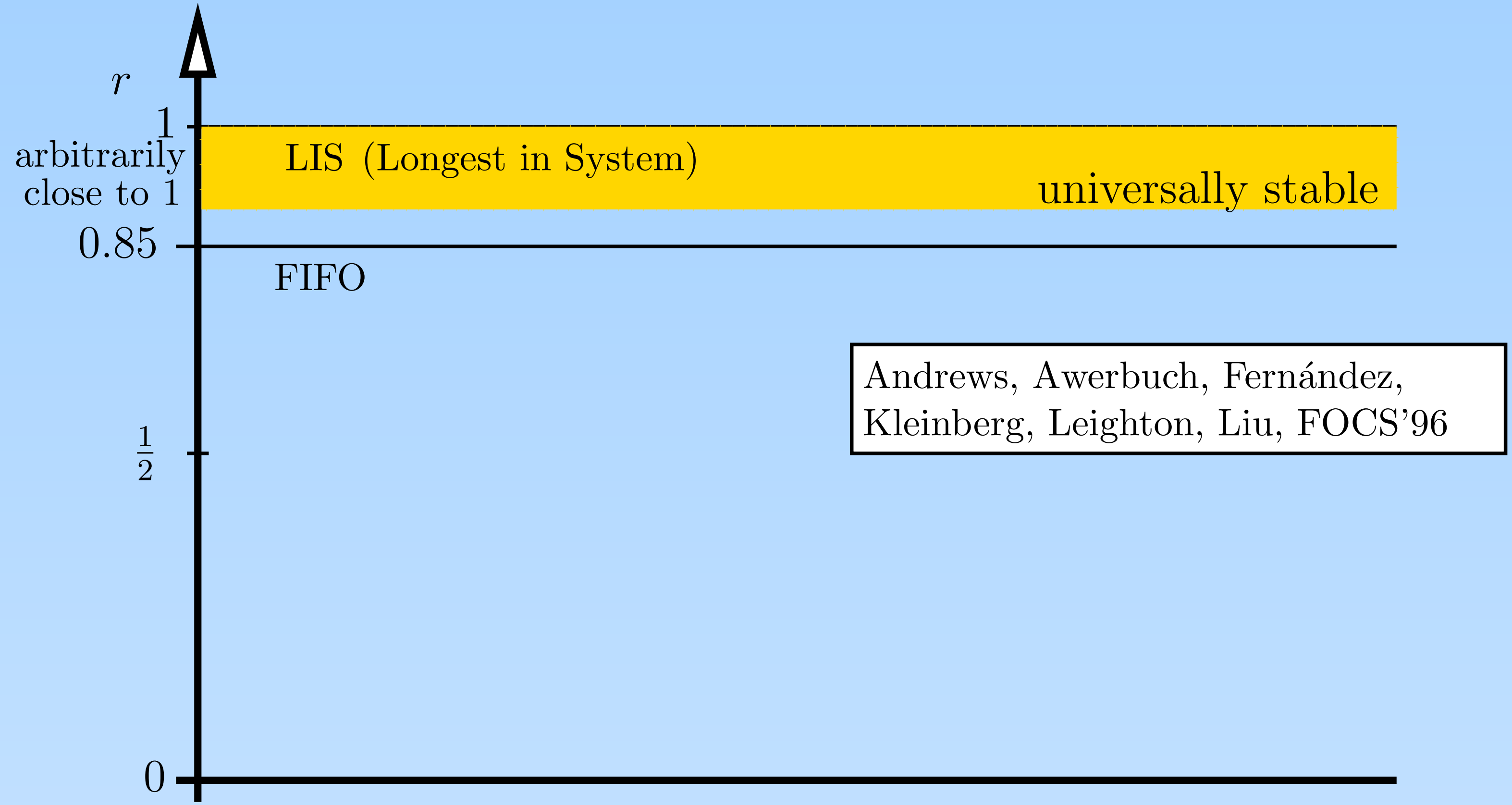
Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work

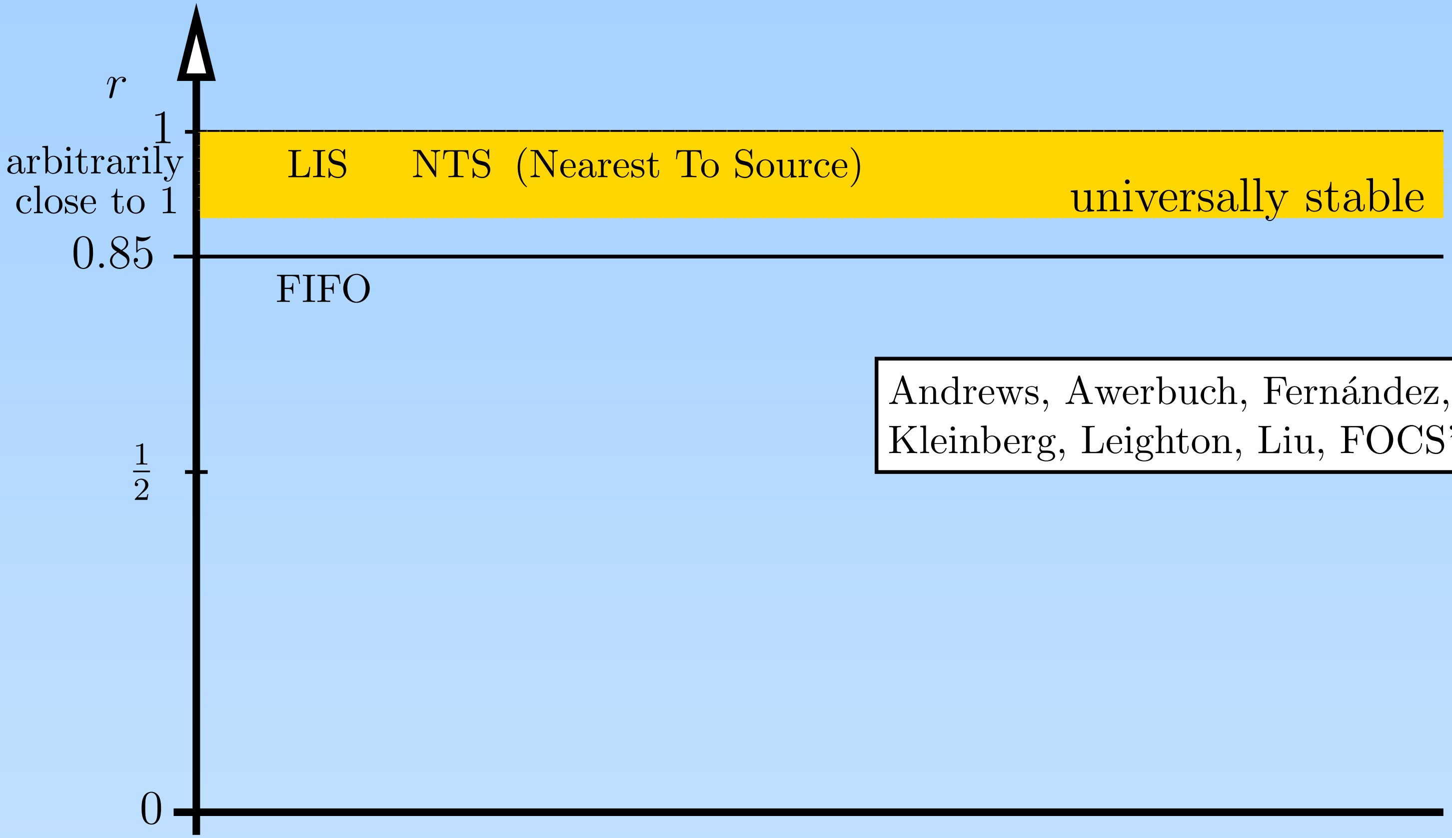


Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work

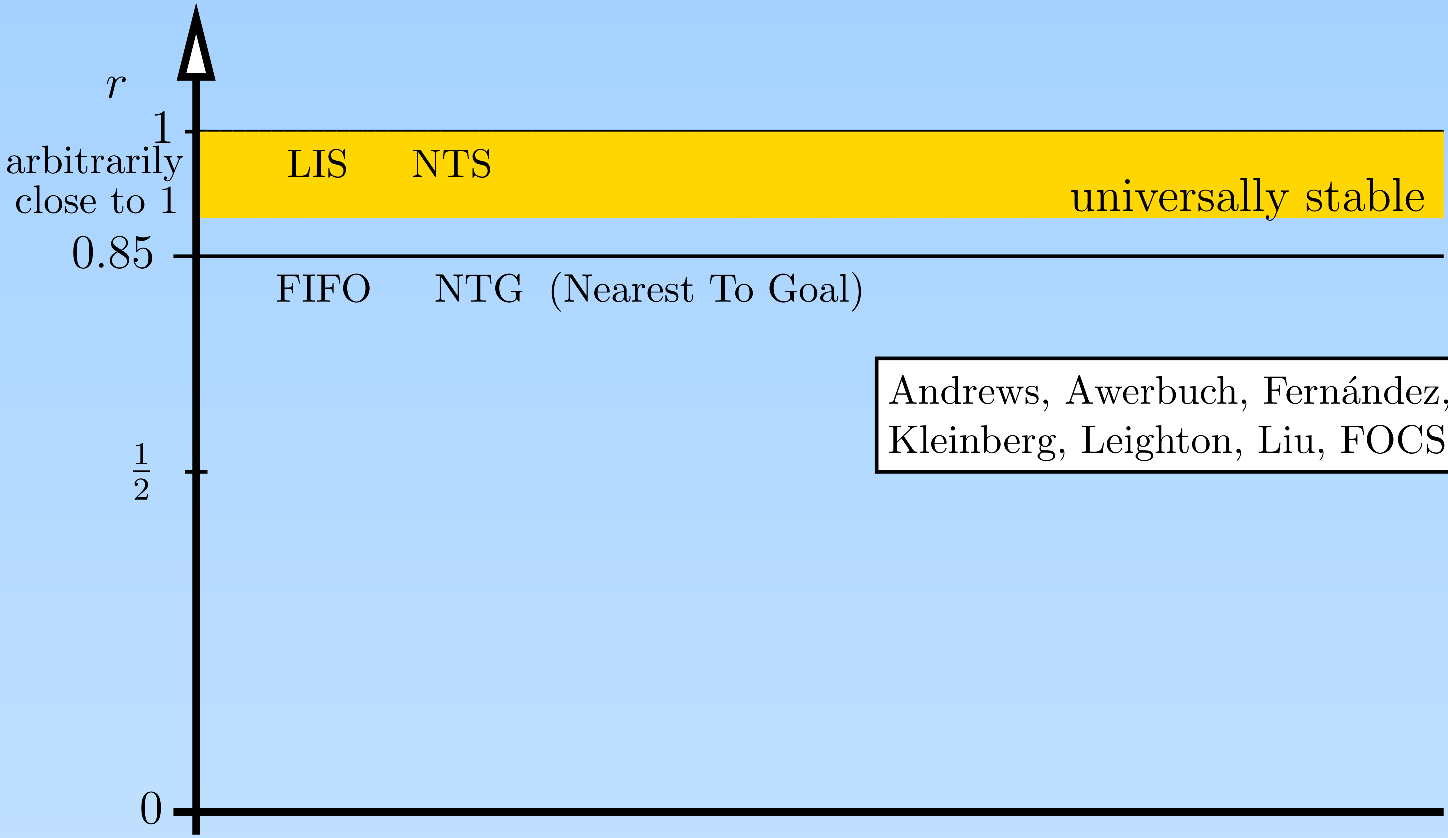


Previous Work



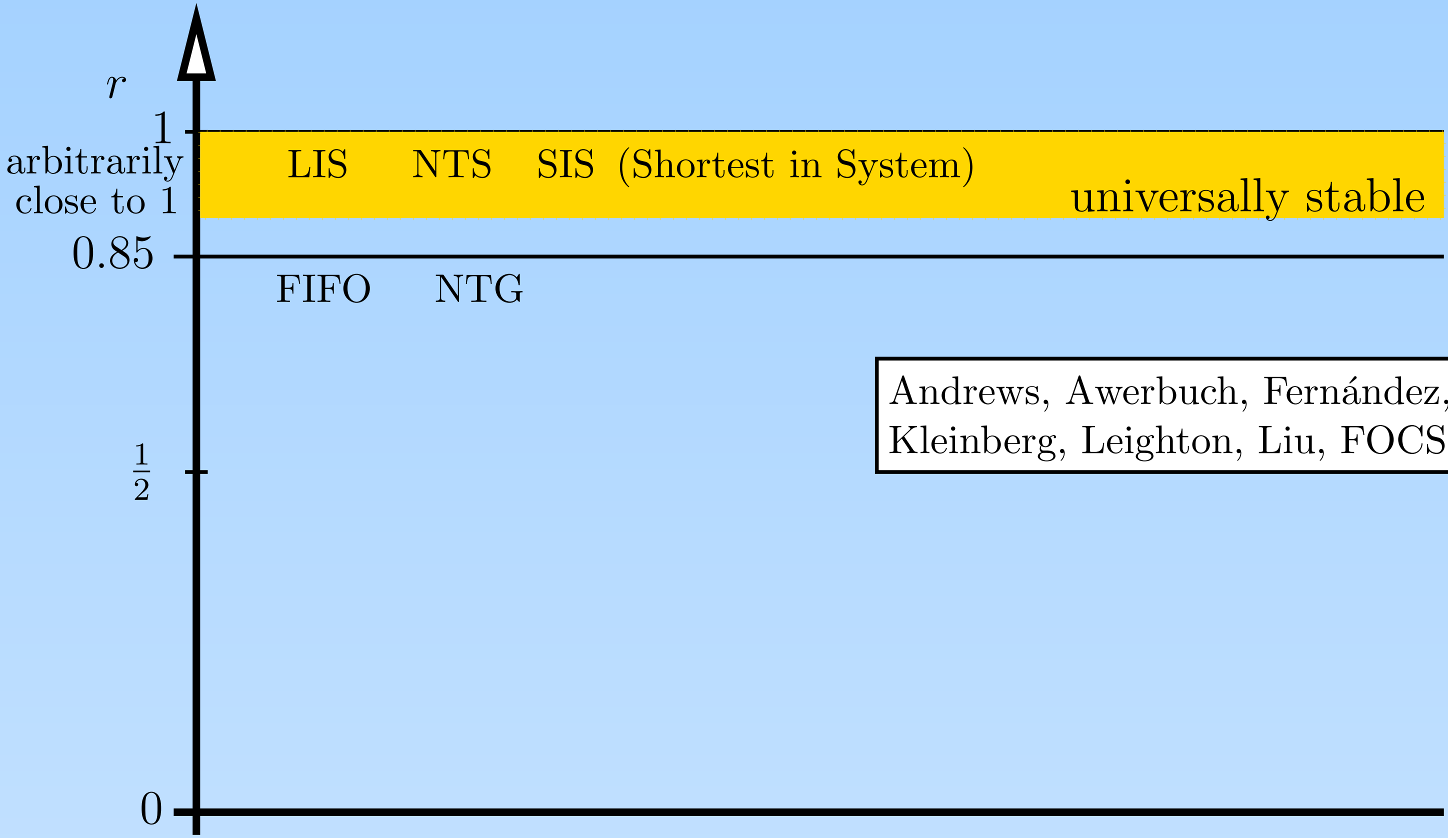
Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work



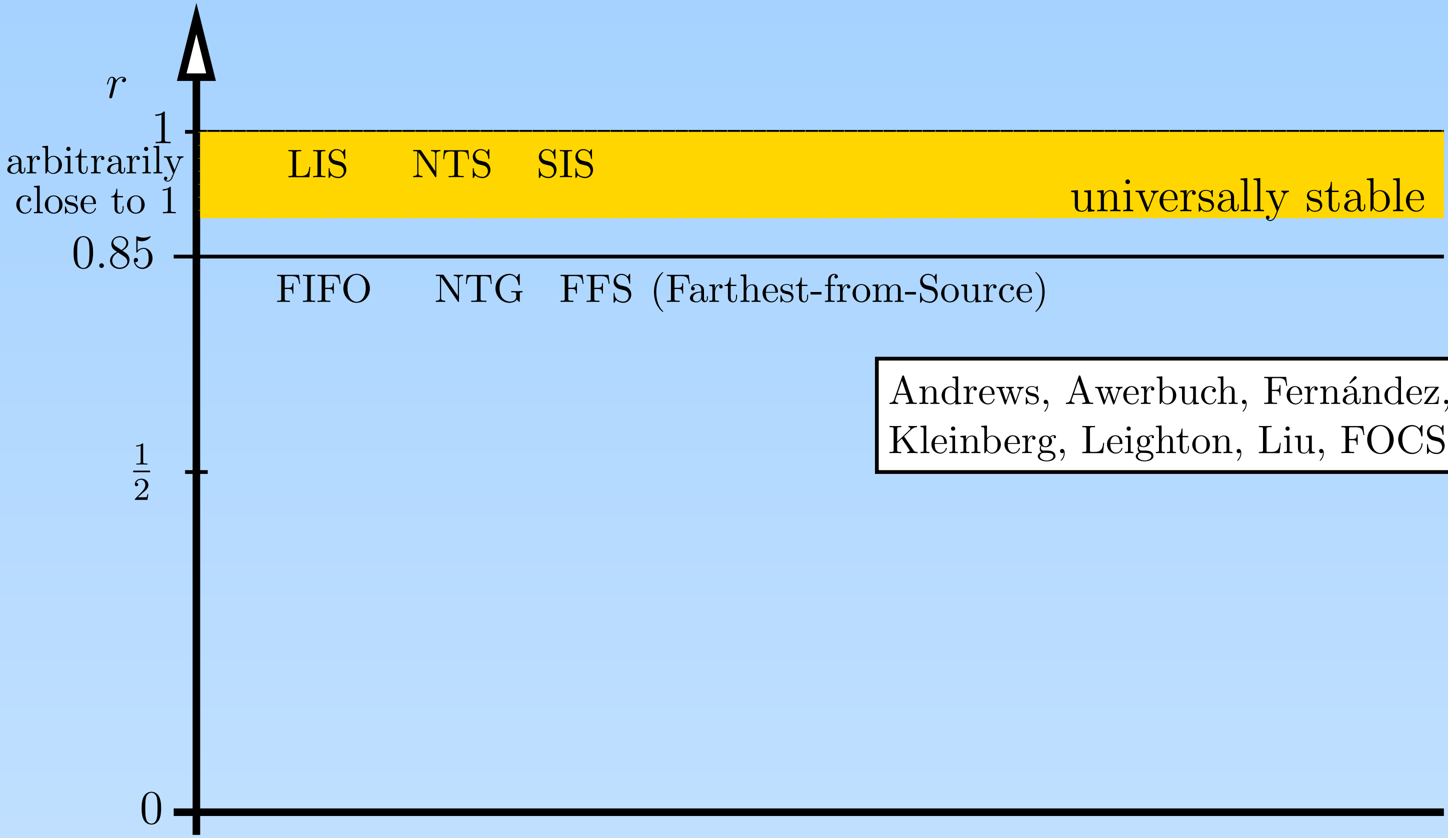
Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work



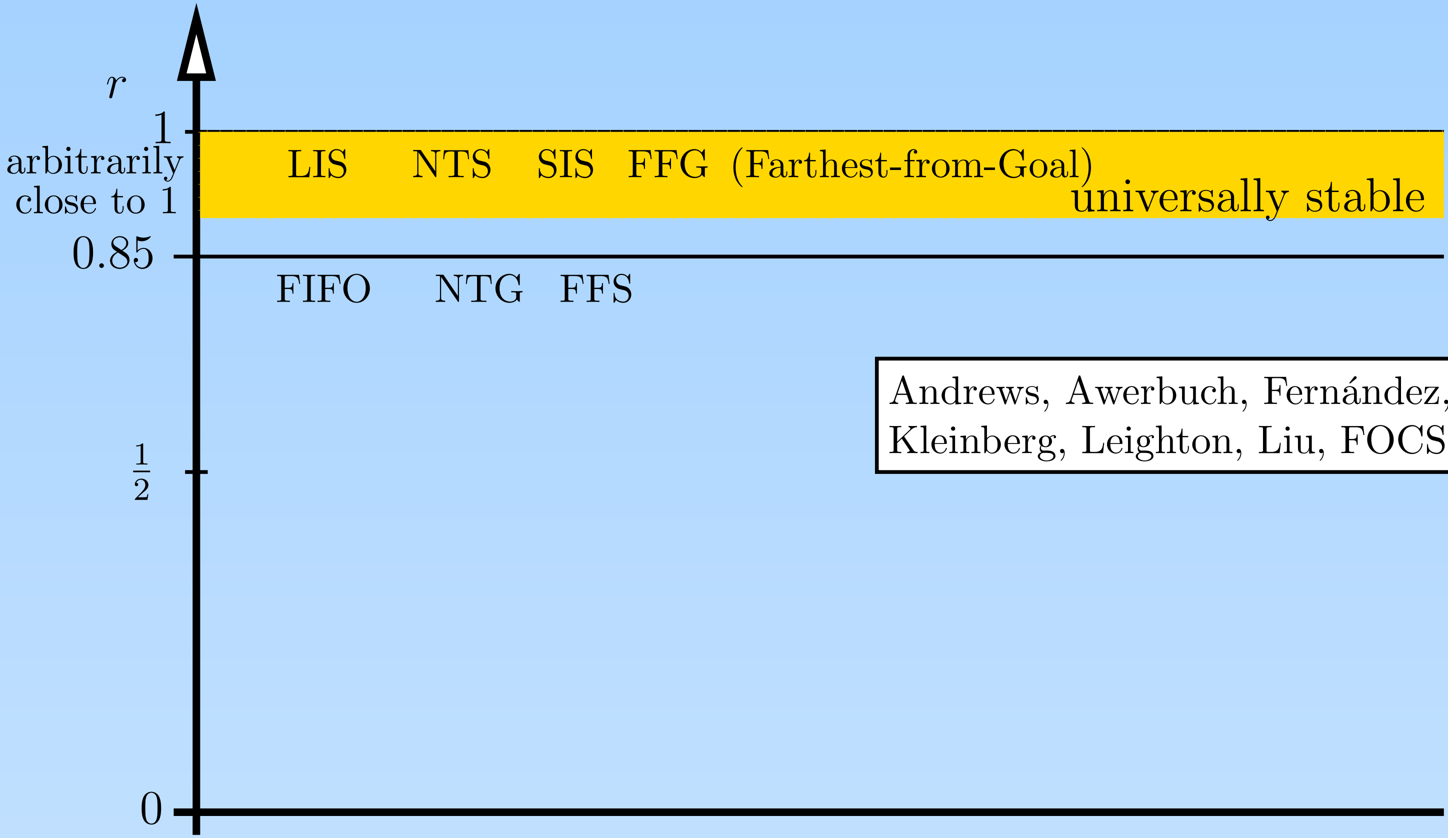
Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work



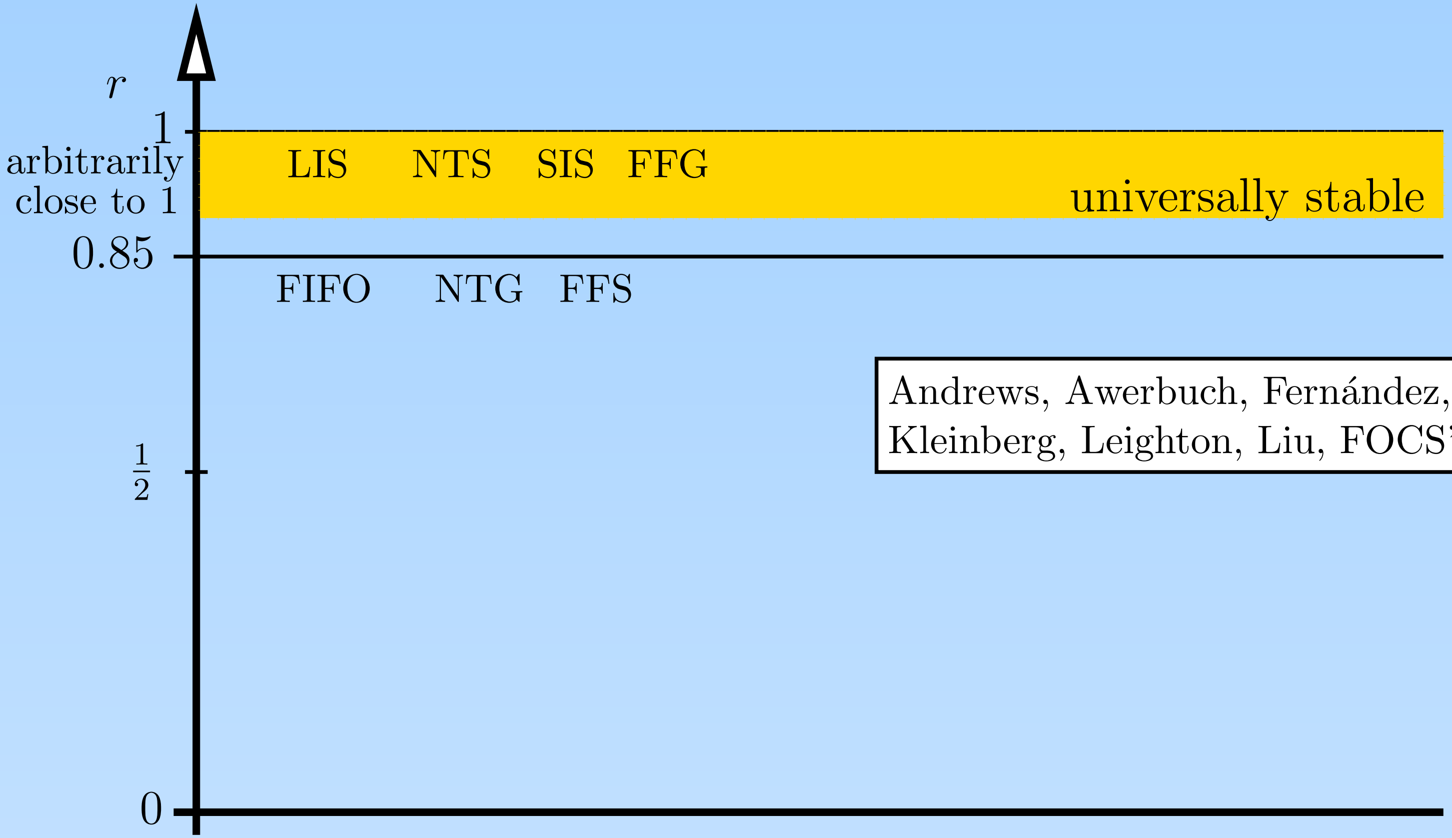
Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work



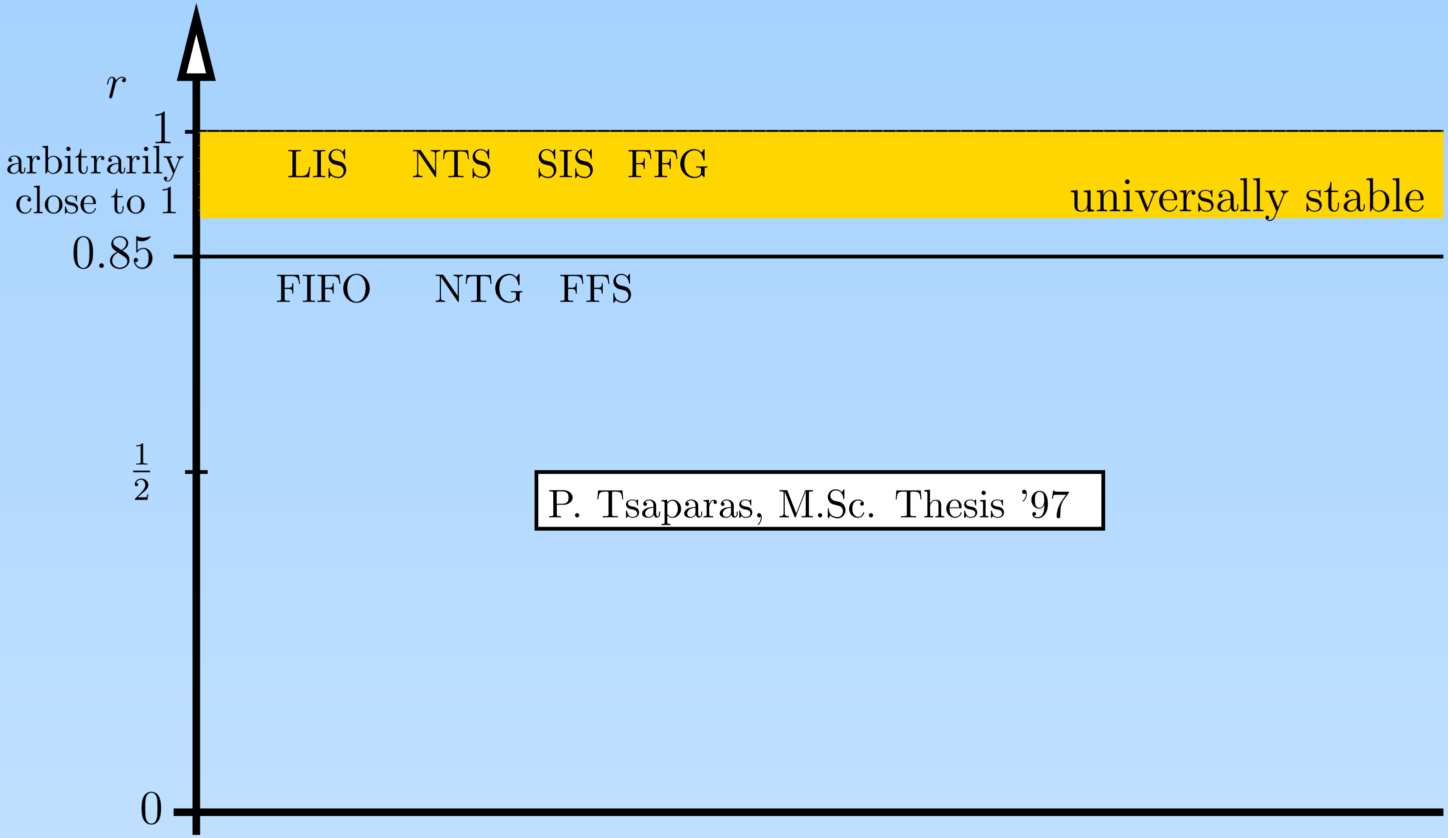
Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

Previous Work

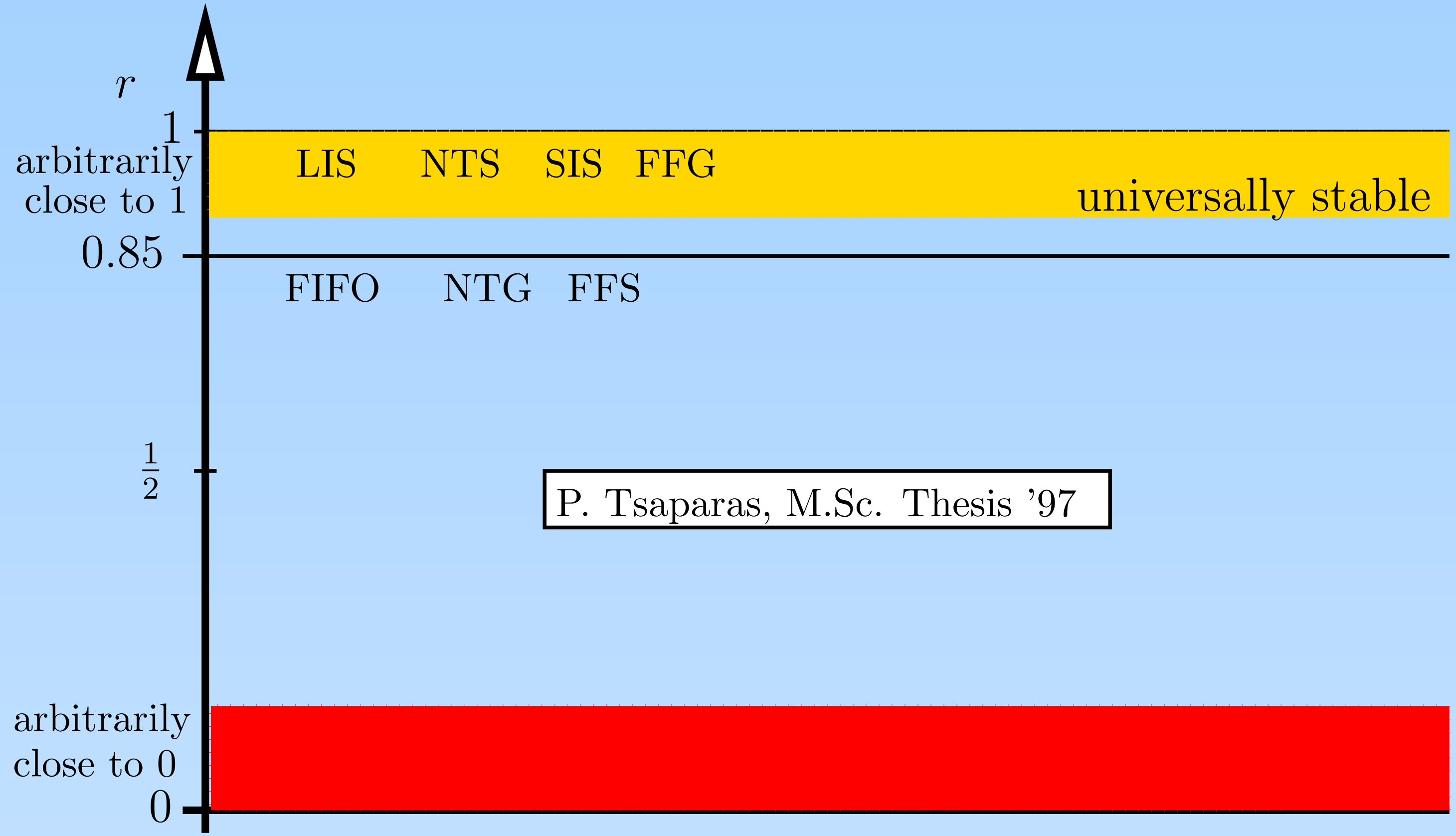


Andrews, Awerbuch, Fernández, Kleinberg, Leighton, Liu, FOCS'96

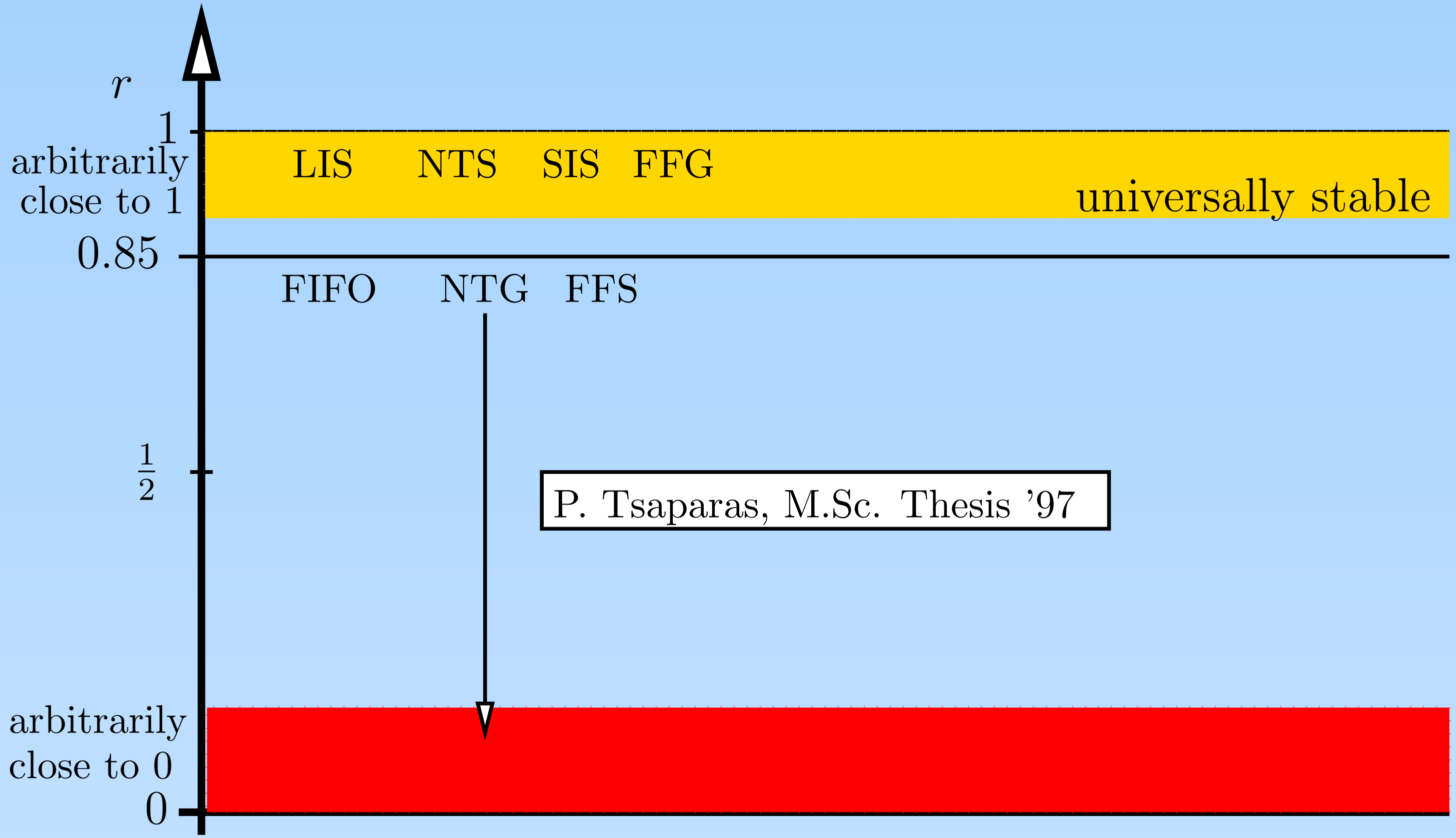
Previous Work



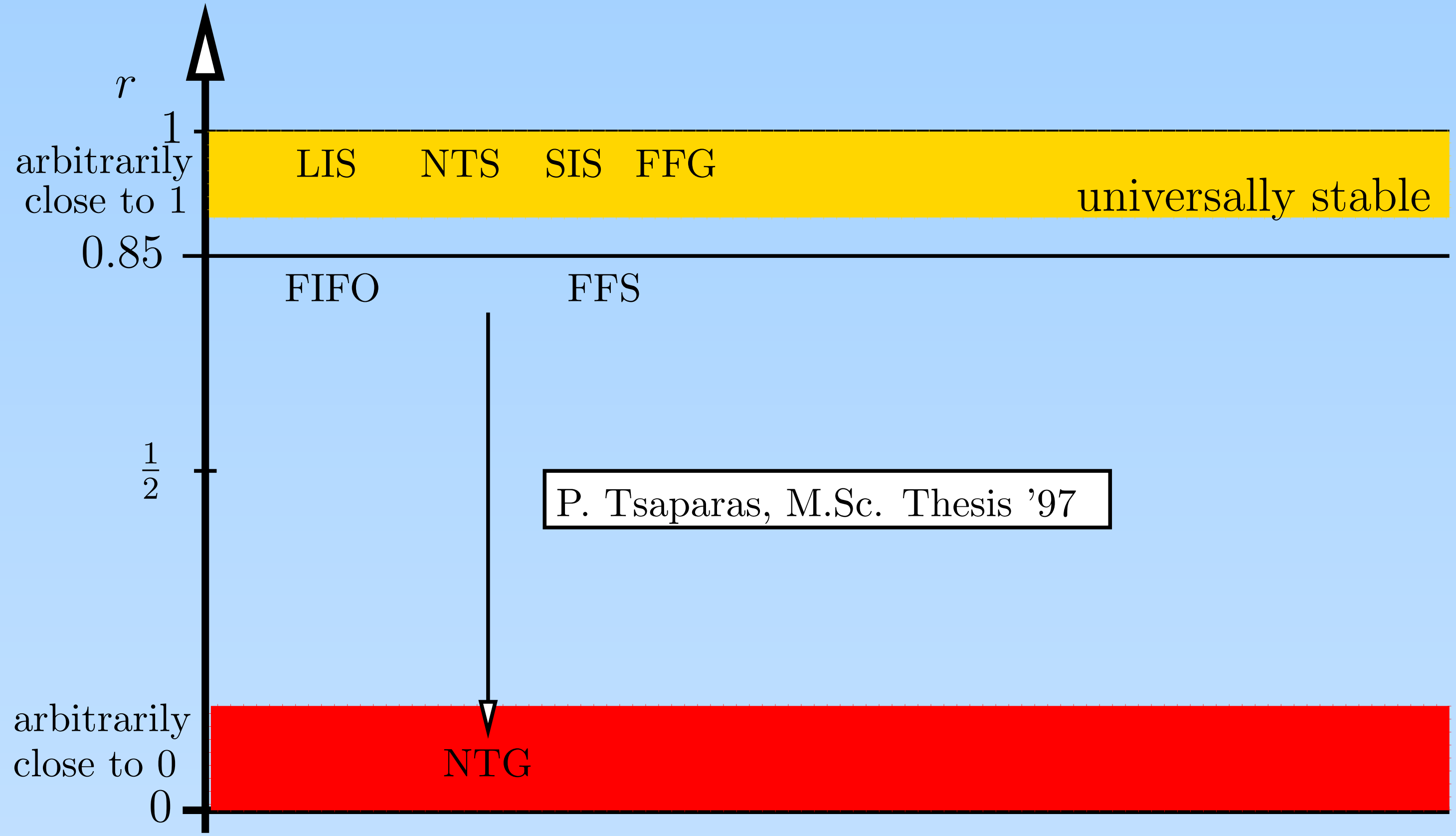
Previous Work



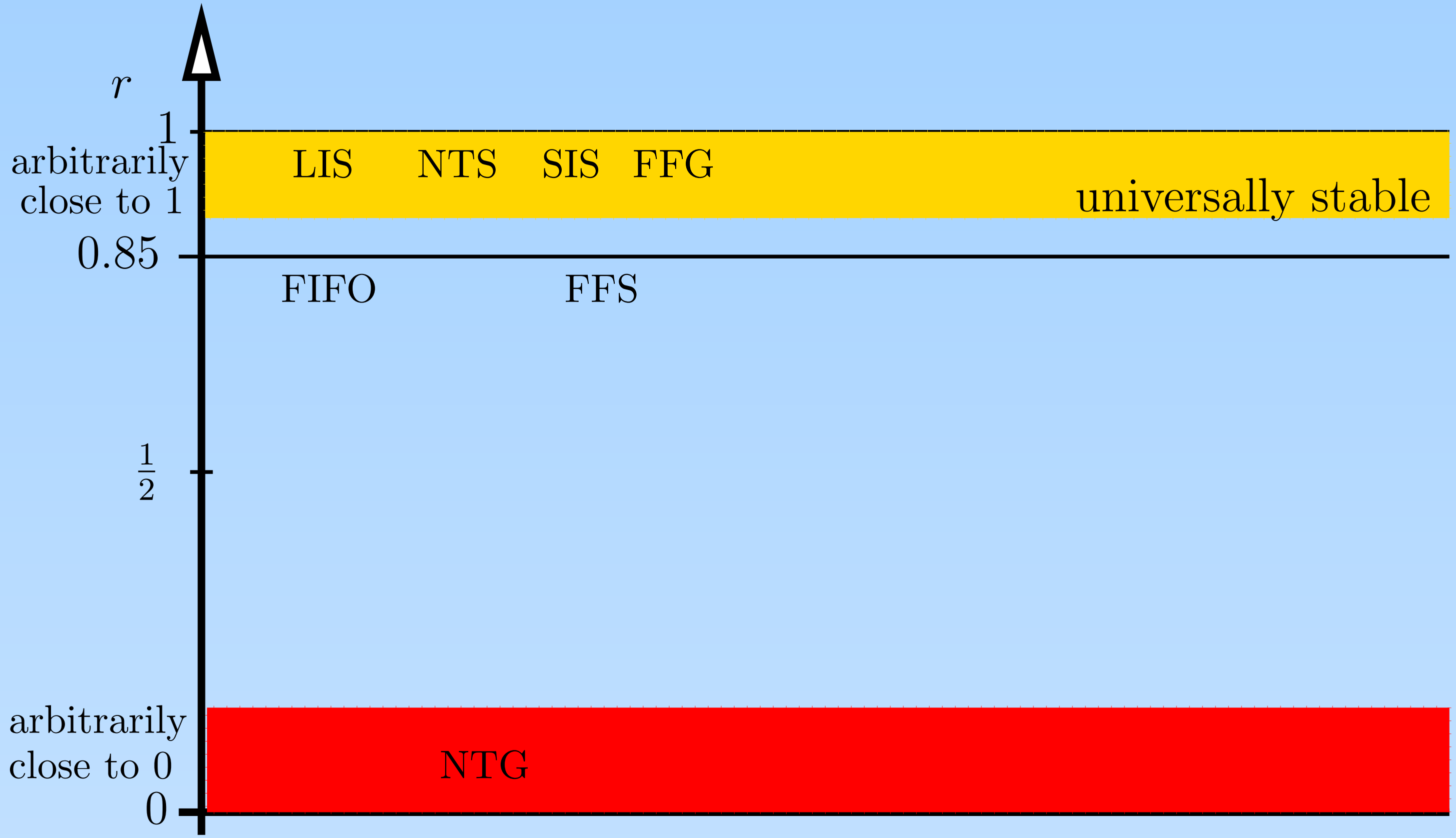
Previous Work



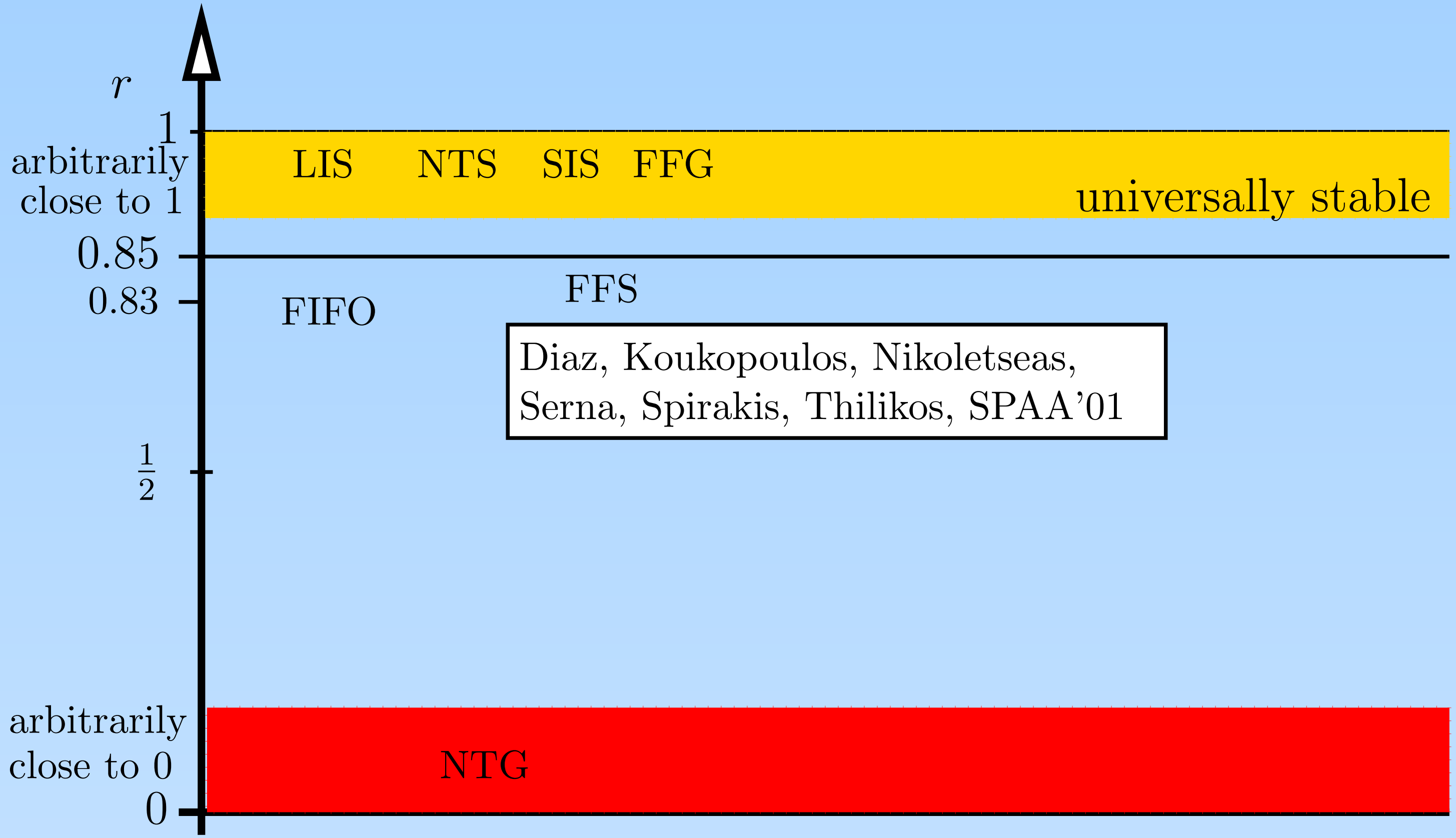
Previous Work



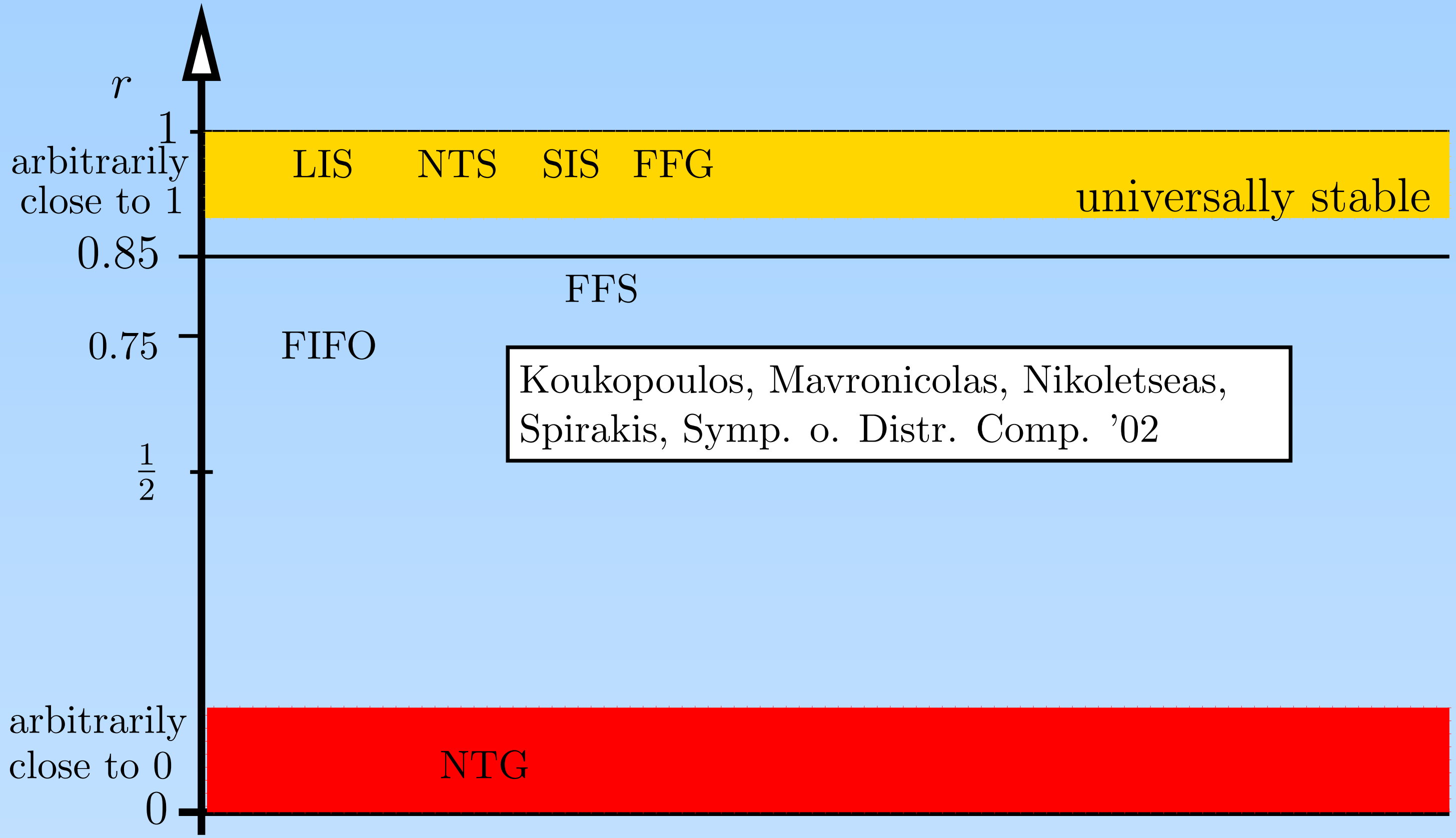
Previous Work



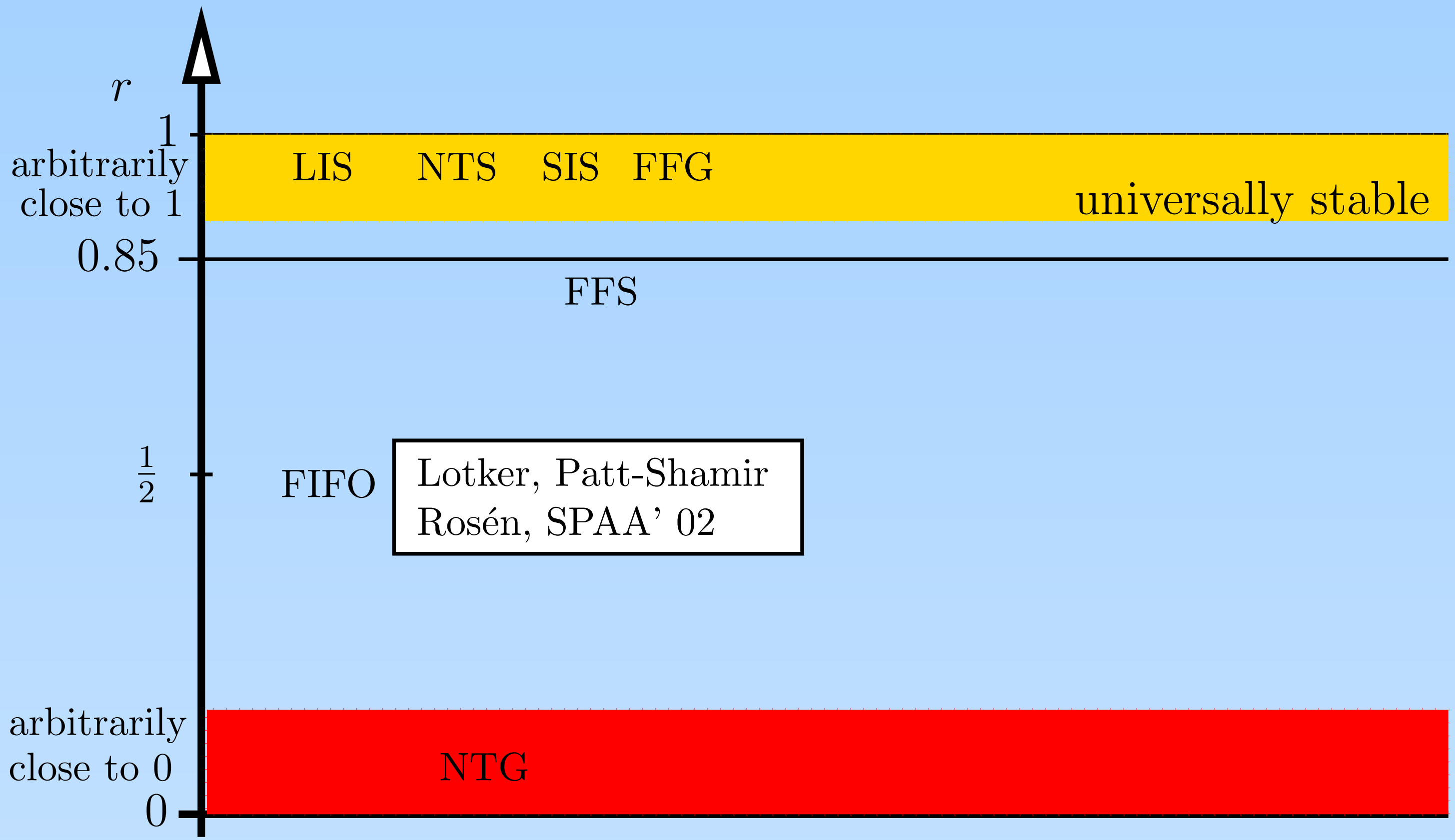
Previous Work



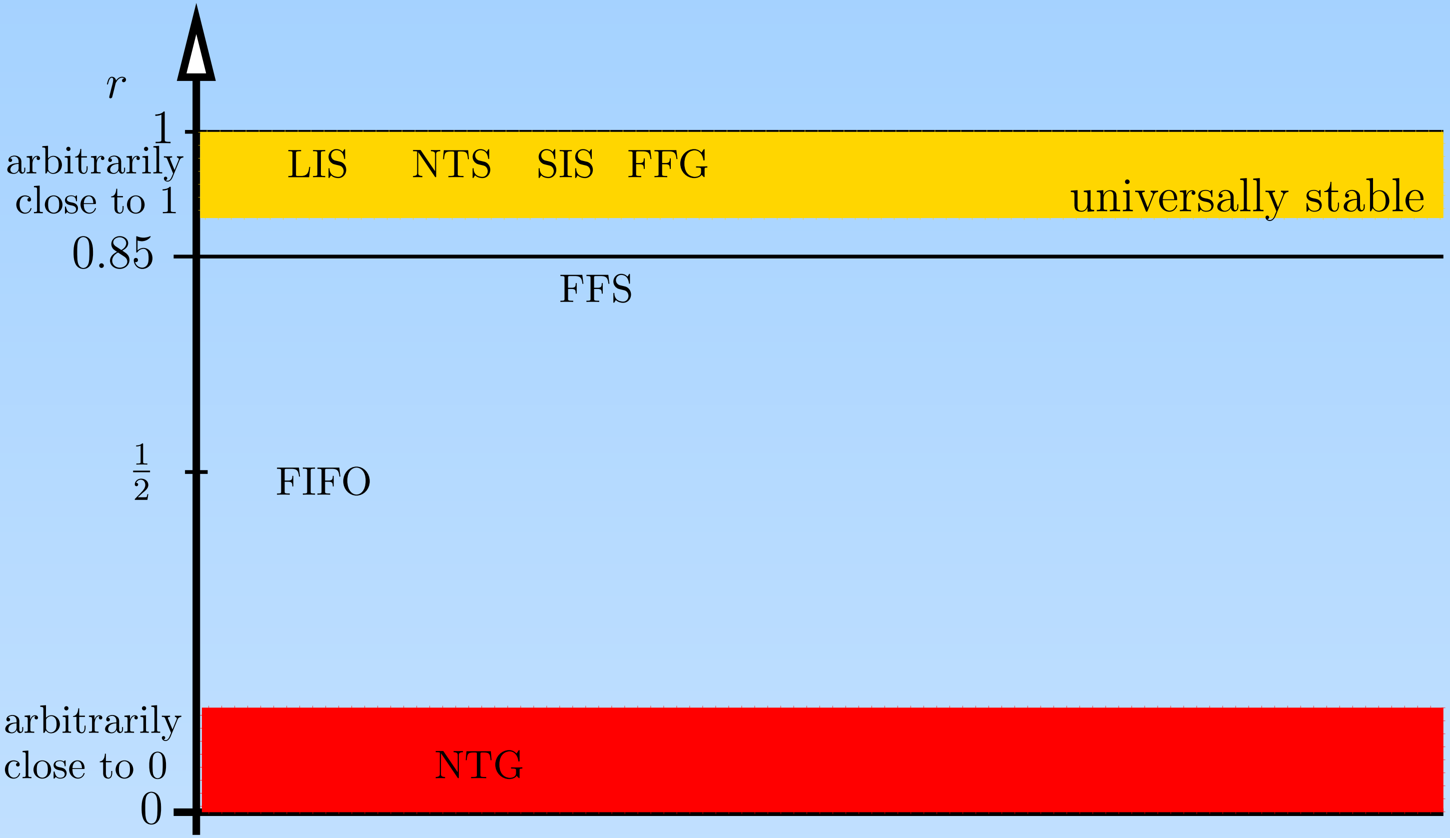
Previous Work



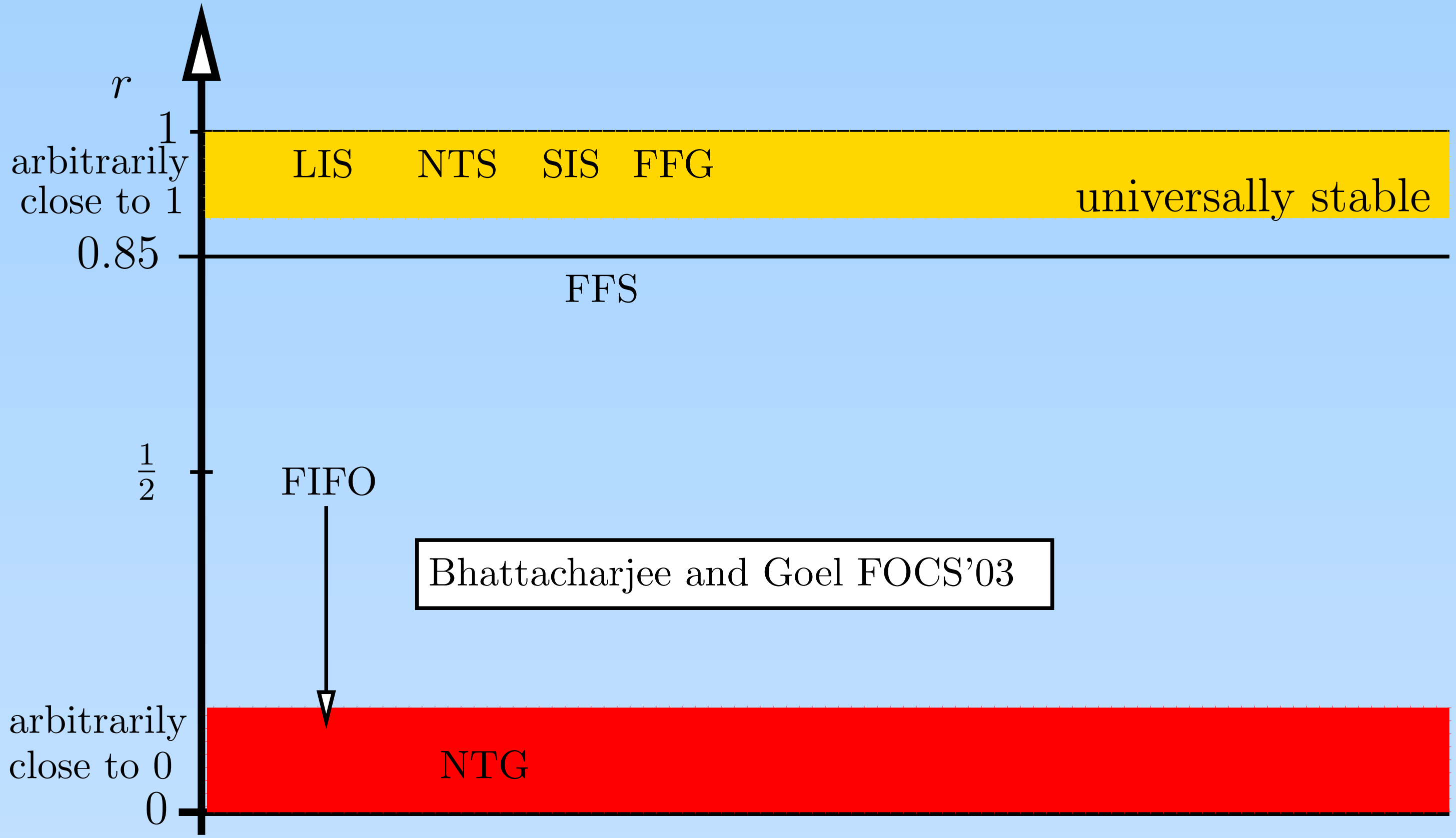
Previous Work



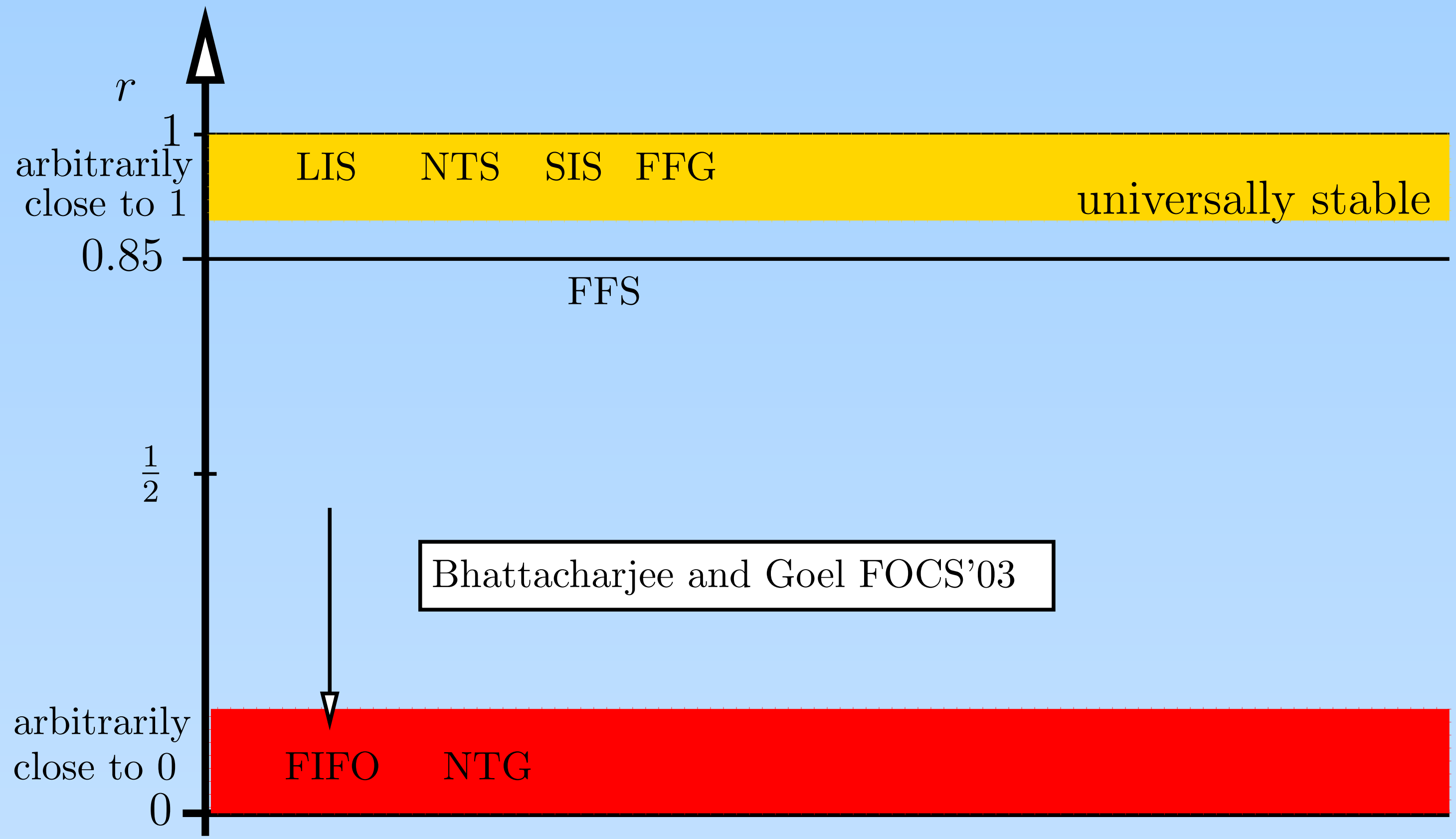
Previous Work



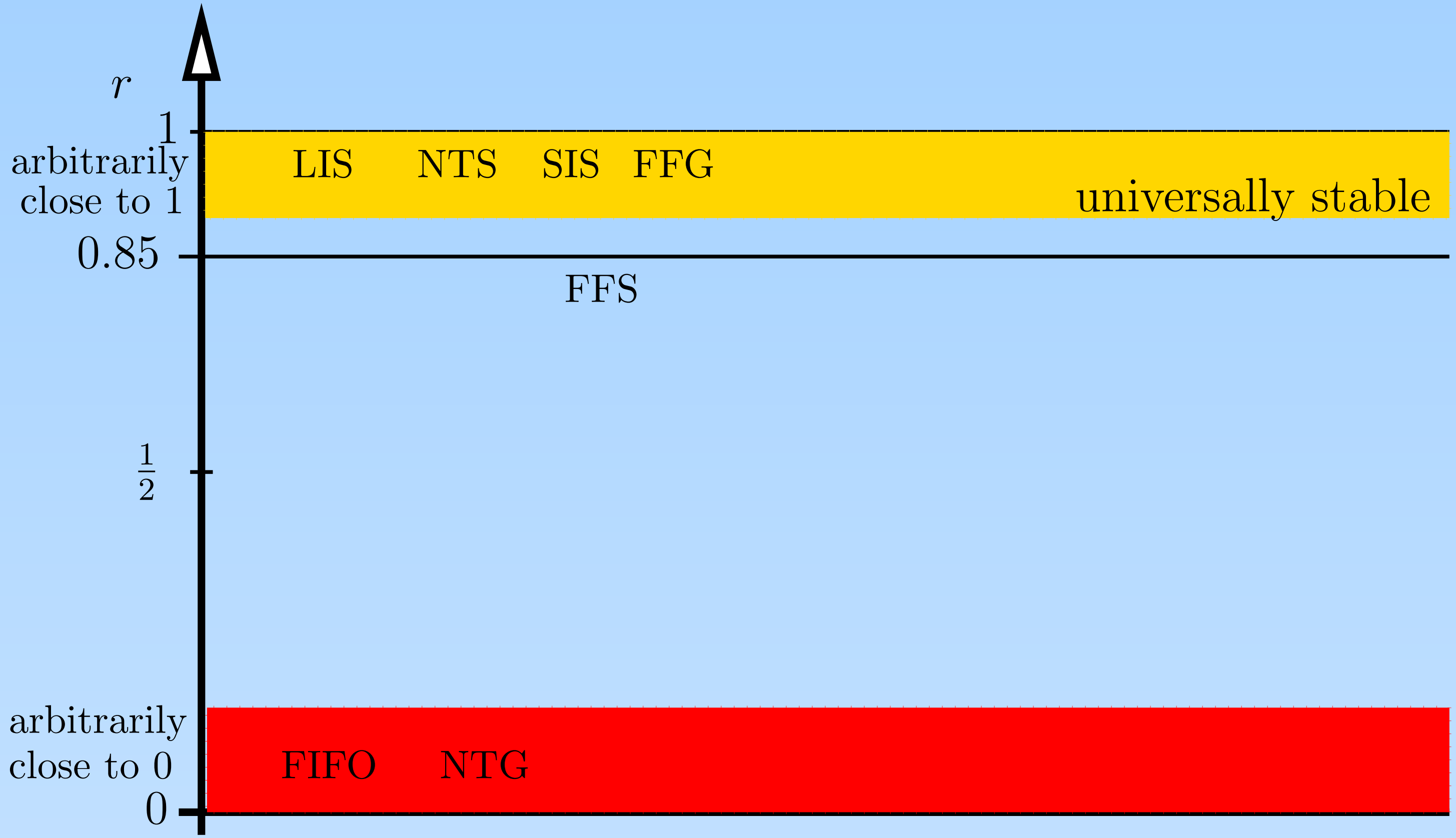
Previous Work



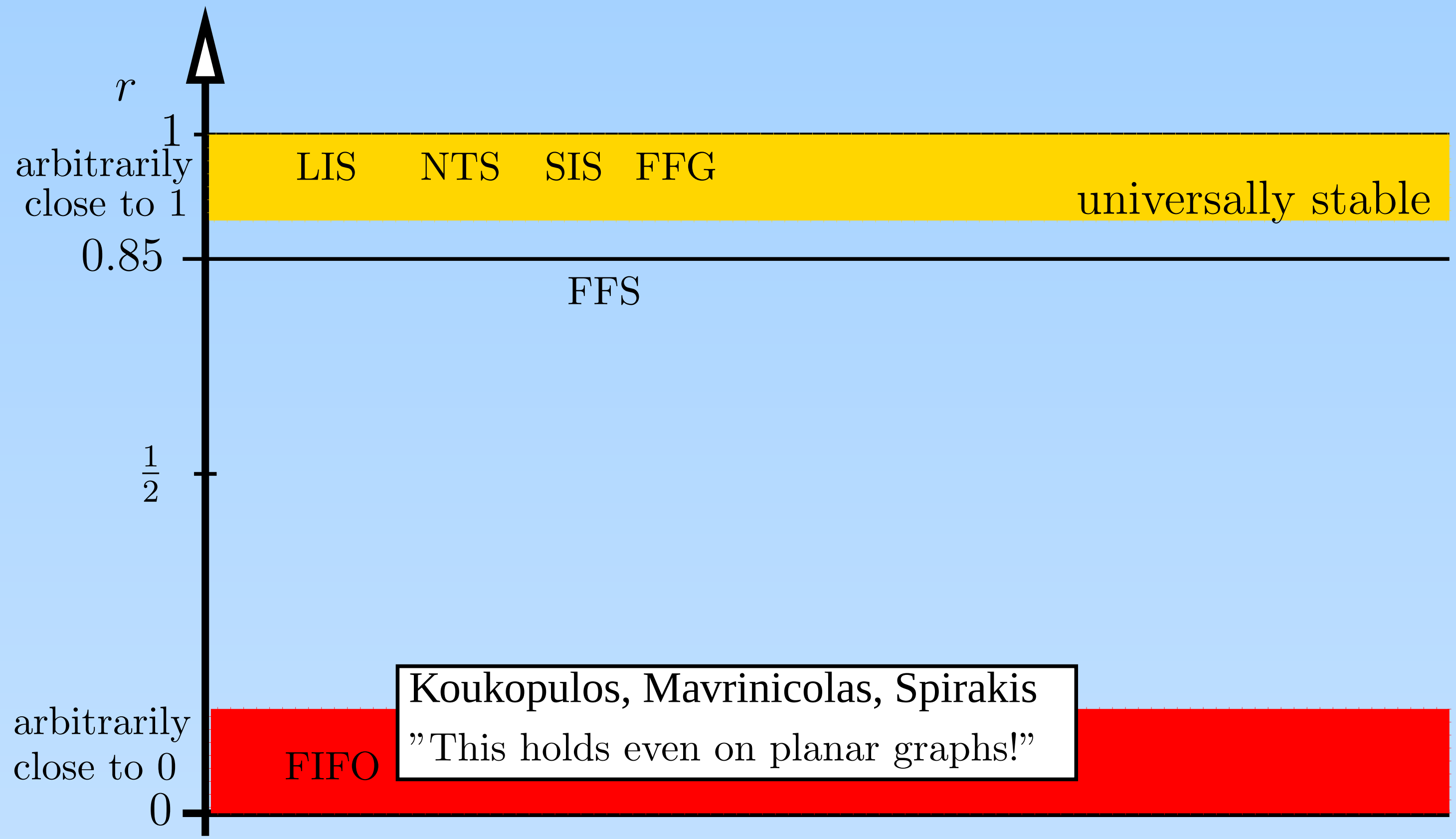
Previous Work



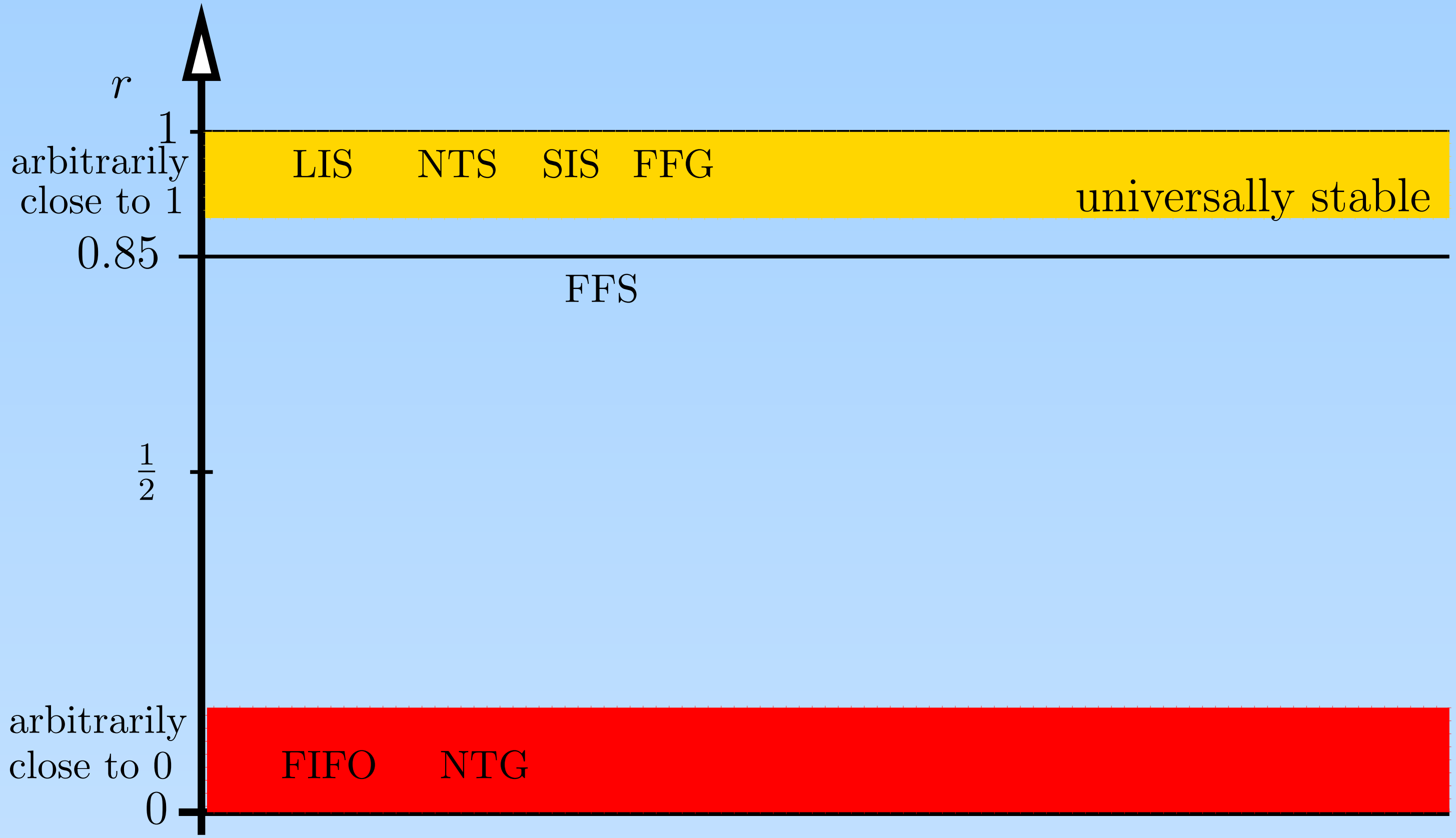
Previous Work



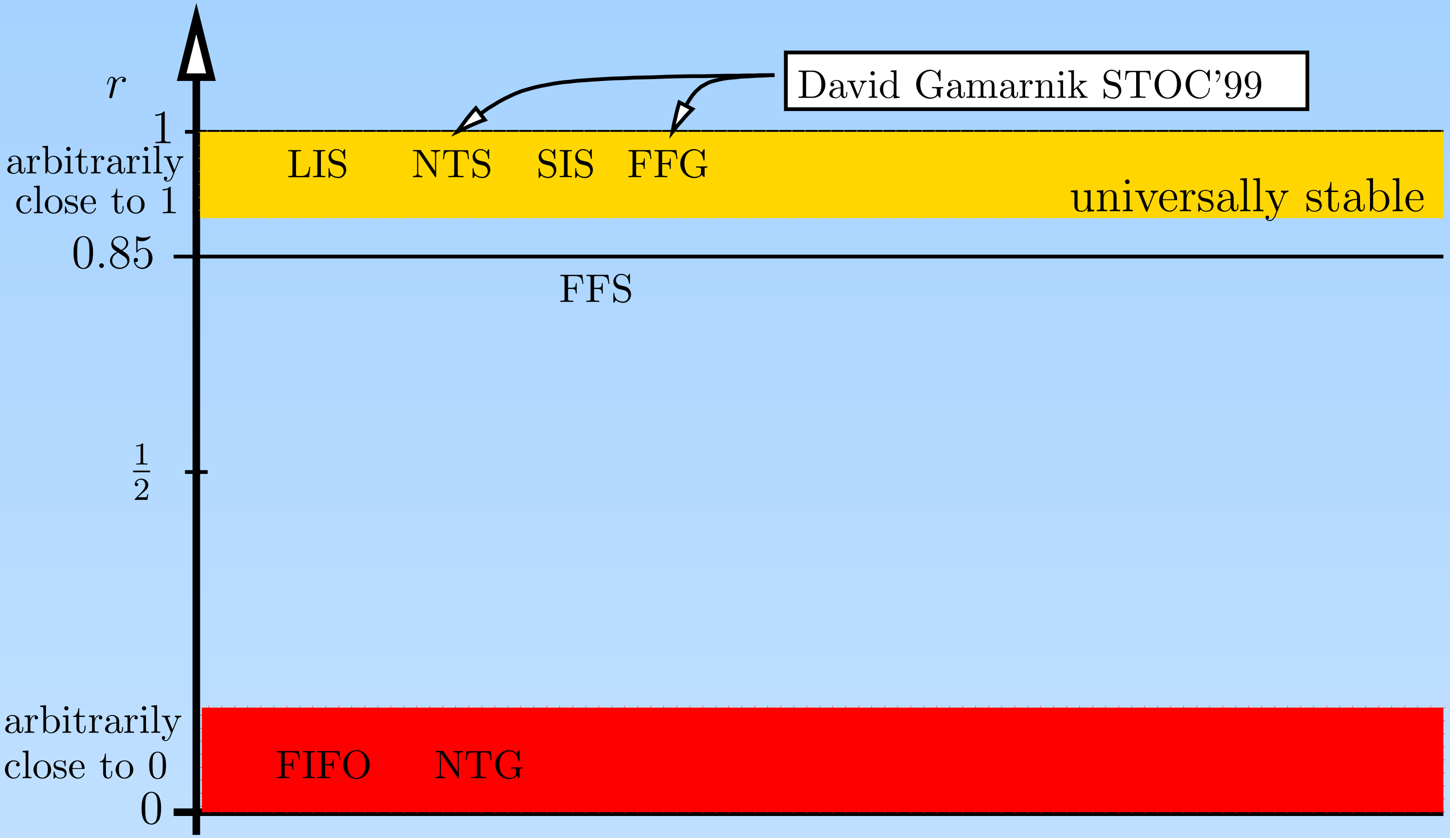
Previous Work



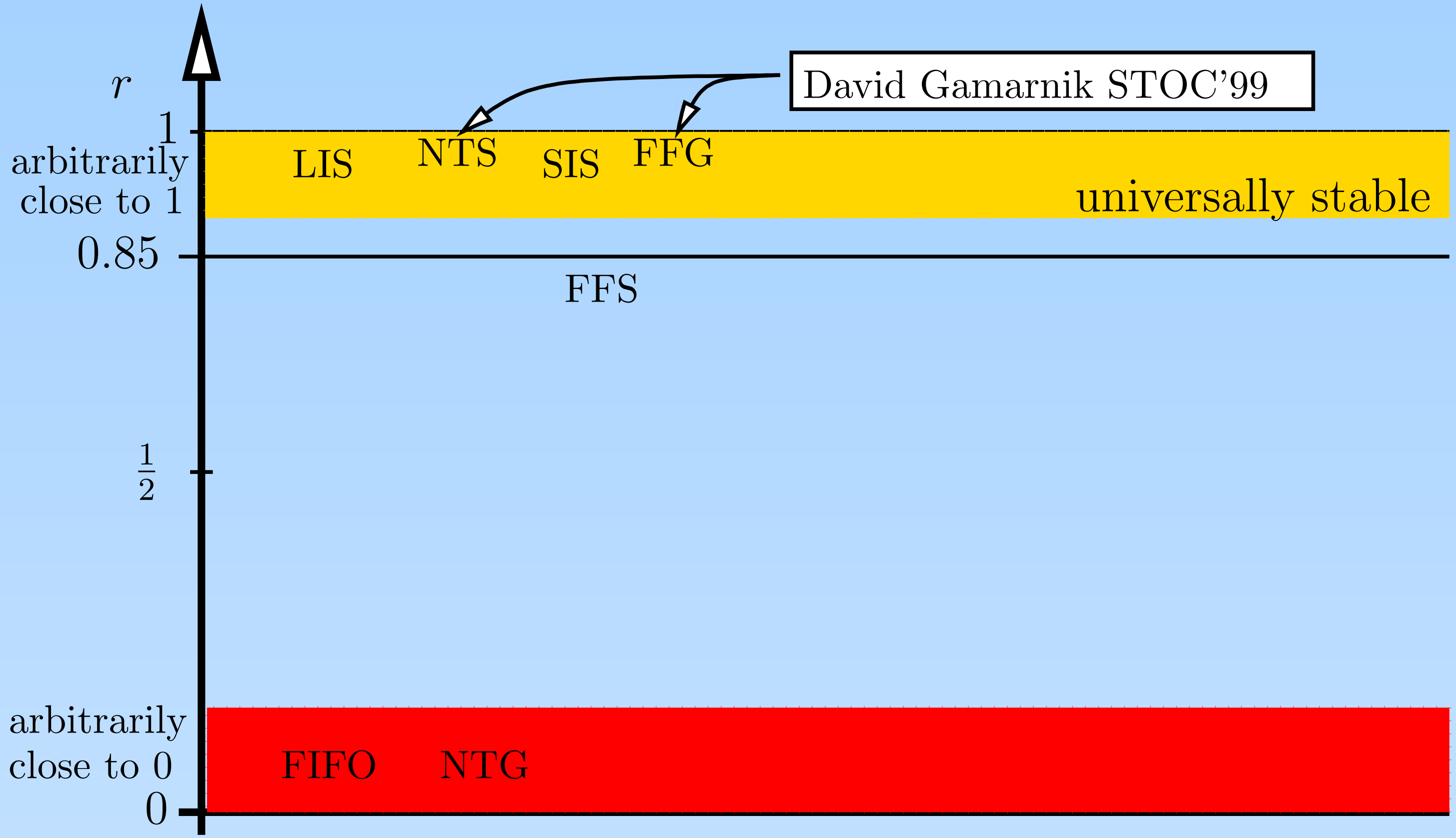
Previous Work



Previous Work



Previous Work



Universally Stable Strategies

Universally Stable Strategies

- Under Nearest-To-Source, Farthest-To-Go and Shortest-In-System total traffic and delay of $2^{\Theta(d)}$ with d being the graphs diameter can arise.

Universally Stable Strategies

- Under Nearest-To-Source, Farthest-To-Go and Shortest-In-System total traffic and delay of $2^{\Theta(d)}$ with d being the graphs diameter can arise.

- Longest-In-System: $2^{O(d)}$

Universally Stable Strategies

- Under Nearest-To-Source, Farthest-To-Go and Shortest-In-System total traffic and delay of $2^{\Theta(d)}$ with d being the graphs diameter can arise.

- Longest-In-System: $2^{O(d)}$ and $\Omega(d)$.

Universally Stable Strategies

- Under Nearest-To-Source, Farthest-To-Go and Shortest-In-System total traffic and delay of $2^{\Theta(d)}$ with d being the graphs diameter can arise.
- Longest-In-System: $2^{O(d)}$ and $\Omega(d)$.
- Andrews, Fernández, Goel, Zhang (FOCS 2001):

Universally Stable Strategies

- Under Nearest-To-Source, Farthest-To-Go and Shortest-In-System total traffic and delay of $2^{\Theta(d)}$ with d being the graphs diameter can arise.

- Longest-In-System: $2^{O(d)}$ and $\Omega(d)$.

- Andrews, Fernández, Goel, Zhang (FOCS 2001):

  Queueing Policy with polynomial delay obtained by an elaborate derandomization.

Universally Stable Strategies

- Under Nearest-To-Source, Farthest-To-Go and Shortest-In-System total traffic and delay of $2^{\Theta(d)}$ with d being the graphs diameter can arise.

- Longest-In-System: $2^{O(d)}$ and $\Omega(d)$.

- Andrews, Fernández, Goel, Zhang (FOCS 2001):

  Queueing Policy with polynomial delay obtained by an elaborate derandomization.

- **Question:**

Universally Stable Strategies

- Under Nearest-To-Source, Farthest-To-Go and Shortest-In-System total traffic and delay of $2^{\Theta(d)}$ with d being the graphs diameter can arise.

- Longest-In-System: $2^{O(d)}$ and $\Omega(d)$.

- Andrews, Fernández, Goel, Zhang (FOCS 2001):

  Queueing Policy with polynomial delay obtained by an elaborate derandomization.

- **Question:** How difficult do strategies with polynomial delay need to be?

WTS-Strategies

WTS-Strategies

A queueing strategy operates **w**ithout **t**ime-**s**tamping if each packet p is assigned a priority

WTS-Strategies

A queueing strategy operates **w**ithout **t**ime-**s**tamping if each packet p is assigned a priority

$$f(G, P, a)$$

where

WTS-Strategies

A queueing strategy operates **w**ithout **t**ime-**s**tamping if each packet p is assigned a priority

$$f(G, P, a)$$

where

- G is the network,

WTS-Strategies

A queueing strategy operates **w**ithout **t**ime-**s**tamping if each packet p is assigned a priority

$$f(G, P, a)$$

where

- G is the network,
- P is the path packet p is travelling on and

WTS-Strategies

A queueing strategy operates **w**ithout **t**ime-**s**tamping if each packet p is assigned a priority

$$f(G, P, a)$$

where

- G is the network,
- P is the path packet p is travelling on and
- a is the number of edges already traversed.

WTS-Strategies

A queueing strategy operates **w**ithout **t**ime-**s**tamping if each packet p is assigned a priority

$$f(G, P, a)$$

where

- G is the network,
- P is the path packet p is travelling on and
- a is the number of edges already traversed.

At every contested edge a packet of maximum priority is advanced.

WTS-Strategies

WTS-Strategies include prominent Strategies like the universally stable Farthest-To-GO (FTG)

WTS-Strategies

WTS-Strategies include prominent Strategies like the universally stable Farthest-To-GO (FTG)

$$f_{FTG}(G, P, a) = |P| - a,$$

WTS-Strategies

WTS-Strategies include prominent Strategies like the universally stable Farthest-To-GO (FTG)

$$f_{FTG}(G, P, a) = |P| - a,$$

but the class is much broader.

WTS-Strategies

WTS-Strategies include prominent Strategies like the universally stable Farthest-To-GO (FTG)

$$f_{FTG}(G, P, a) = |P| - a,$$

but the class is much broader.

The only *reasonable* quantities not used are times:

WTS-Strategies

WTS-Strategies include prominent Strategies like the universally stable Farthest-To-GO (FTG)

$$f_{FTG}(G, P, a) = |P| - a,$$

but the class is much broader.

The only *reasonable* quantities not used are times:

- Longest-In-System uses a packets age.

WTS-Strategies

WTS-Strategies include prominent Strategies like the universally stable Farthest-To-GO (FTG)

$$f_{FTG}(G, P, a) = |P| - a,$$

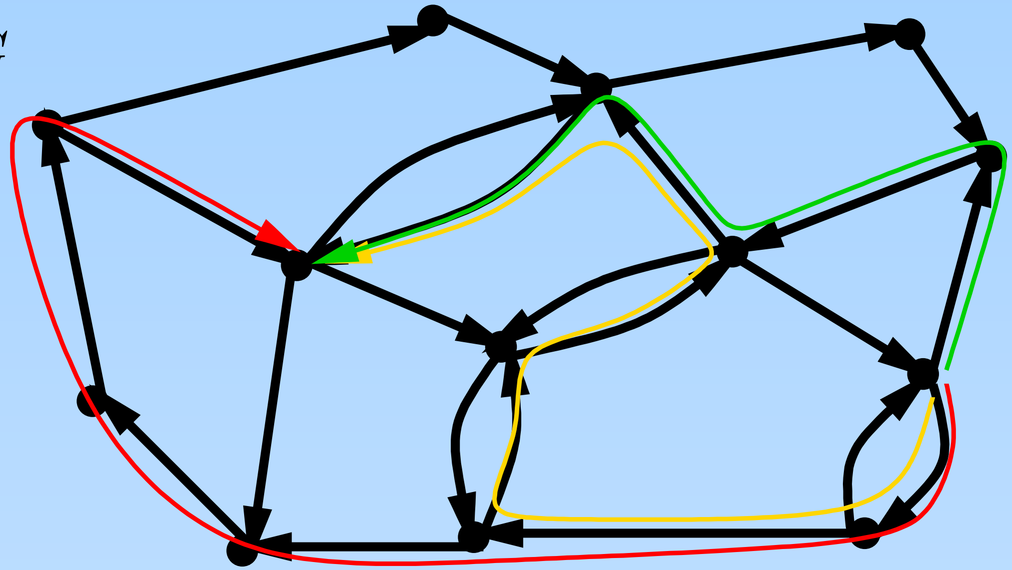
but the class is much broader.

The only *reasonable* quantities not used are times:

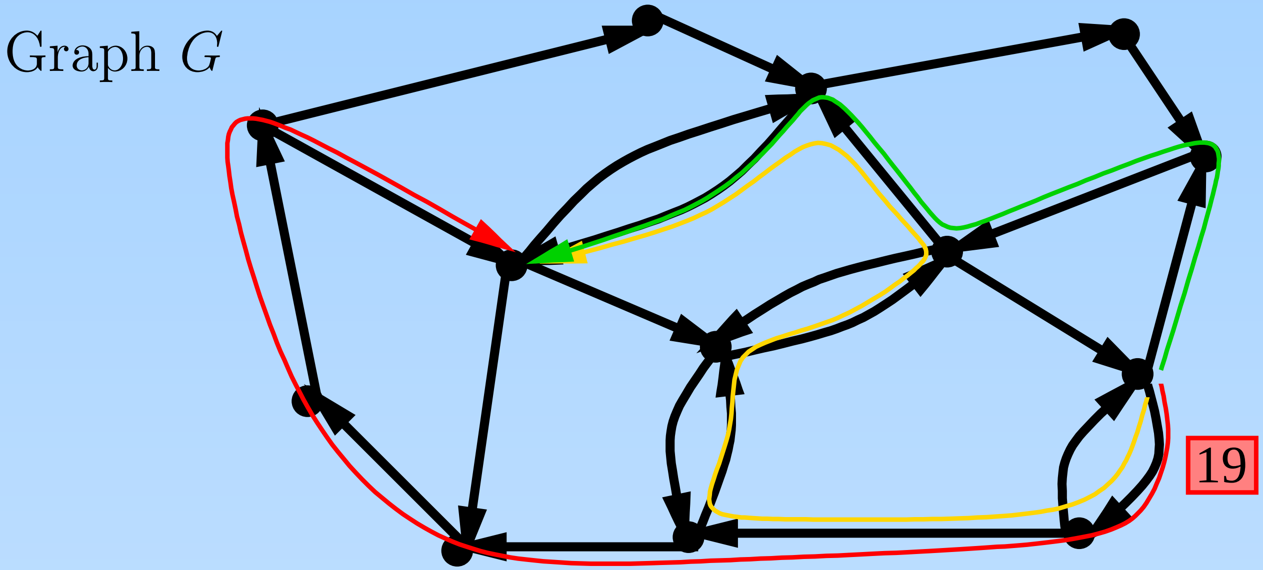
- Longest-In-System uses a packets age.
- FIFO uses a packets current waiting time.

WTS-Strategies

Graph G

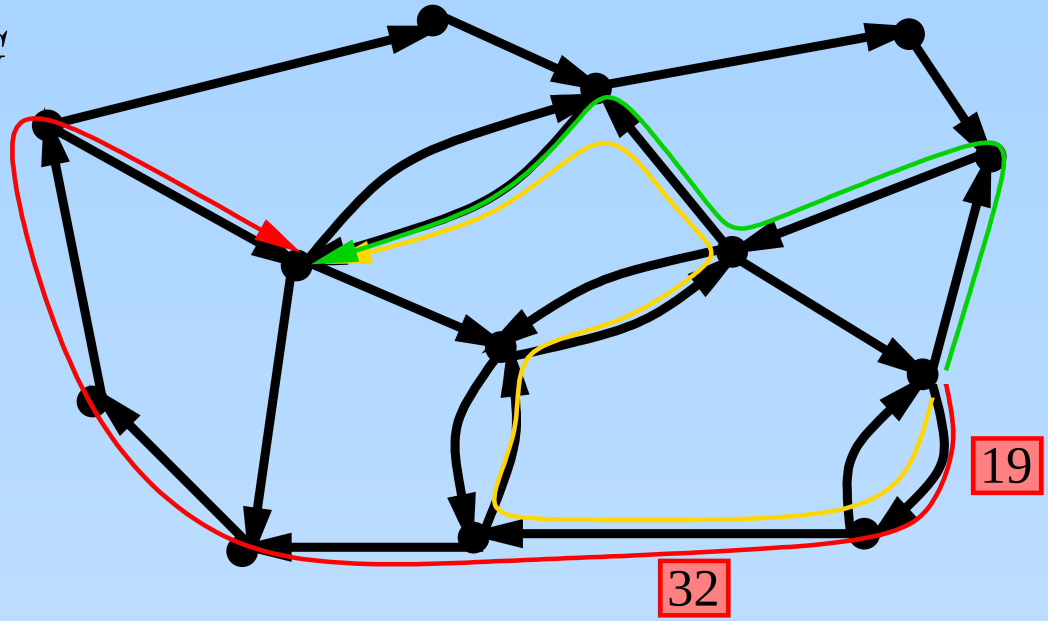


WTS-Strategies

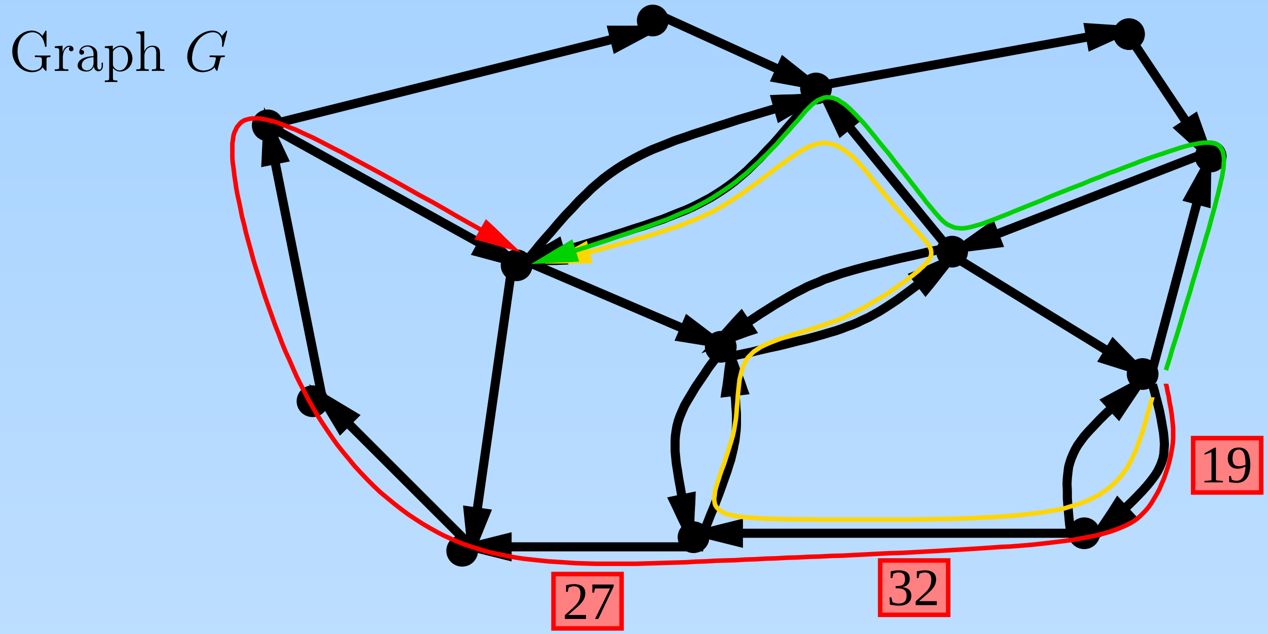


WTS-Strategies

Graph G

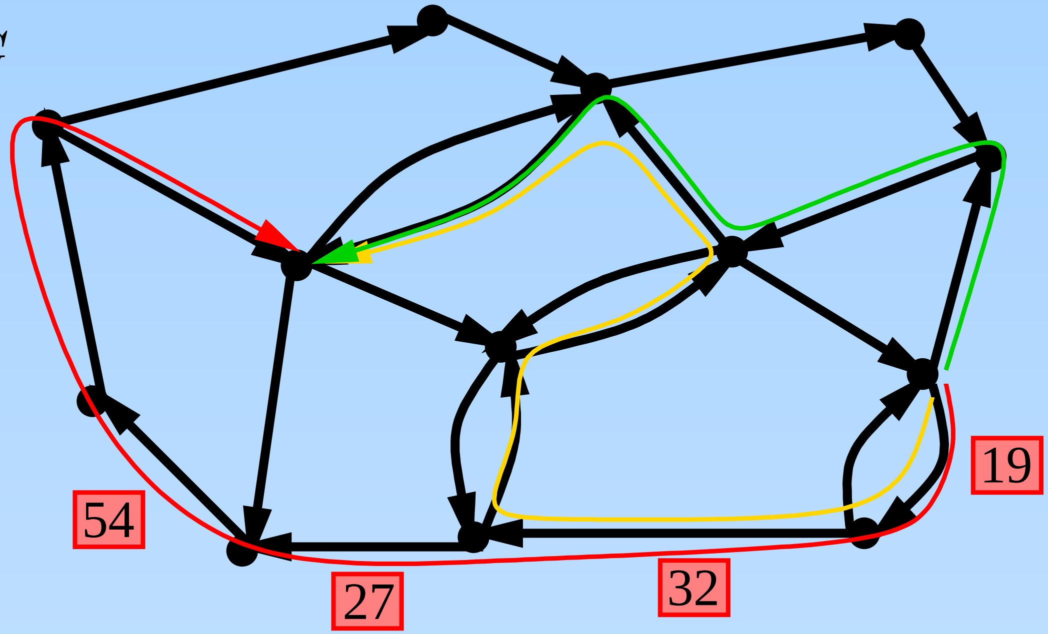


WTS-Strategies

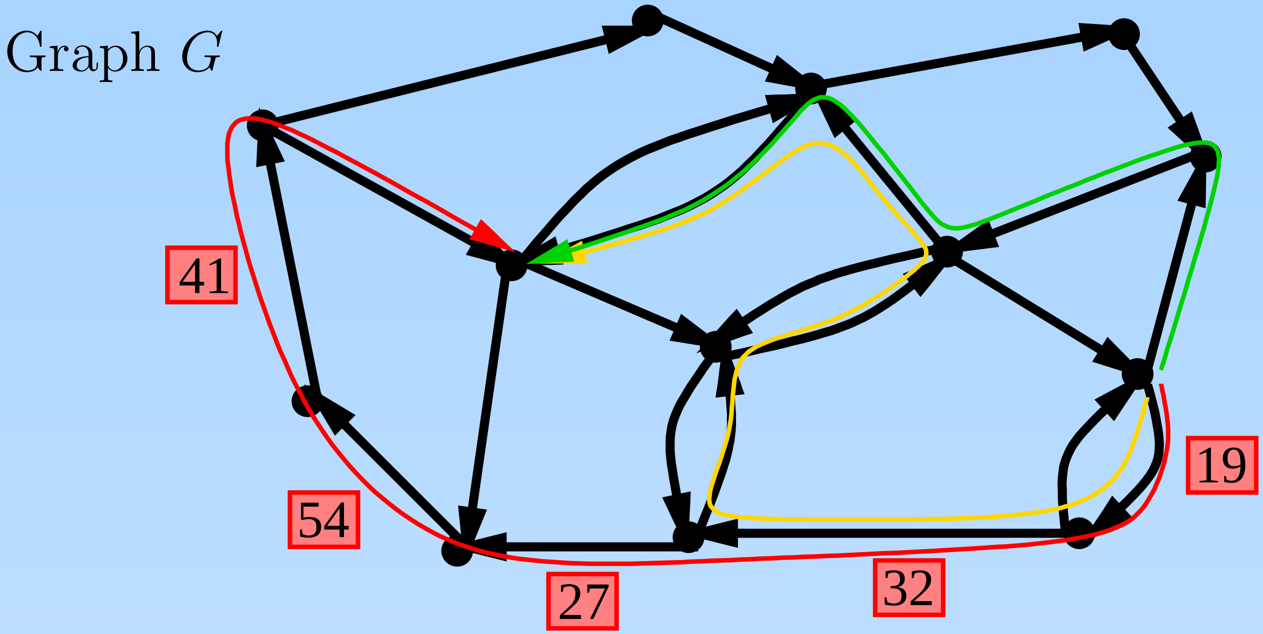


WTS-Strategies

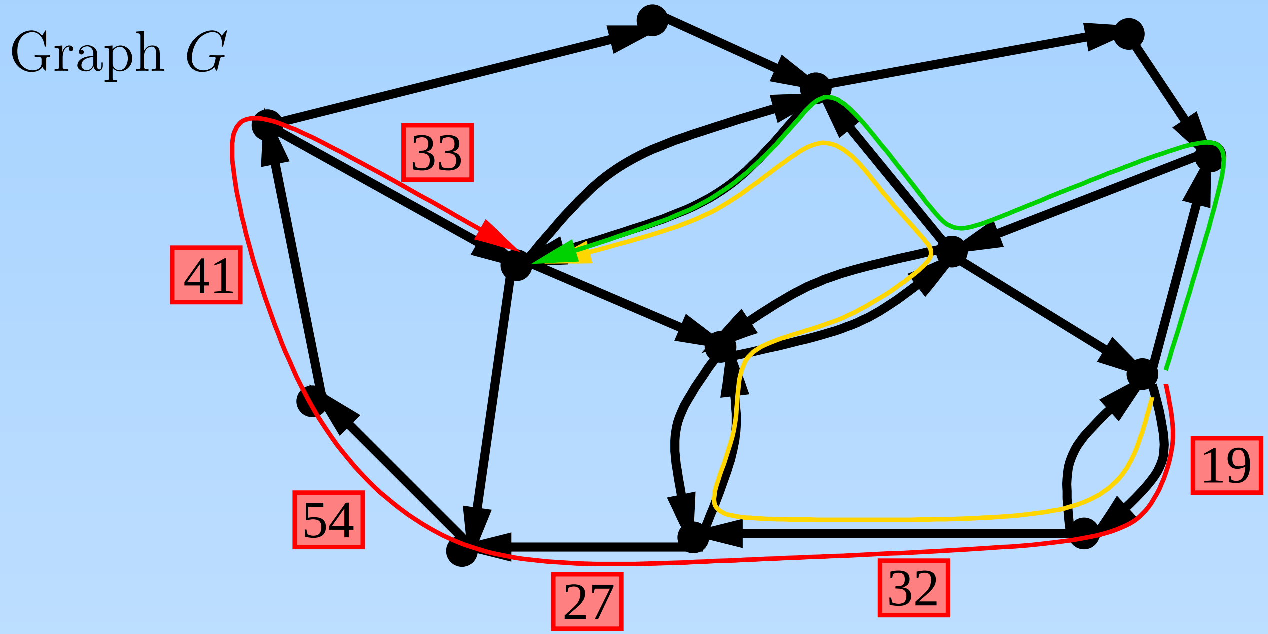
Graph G



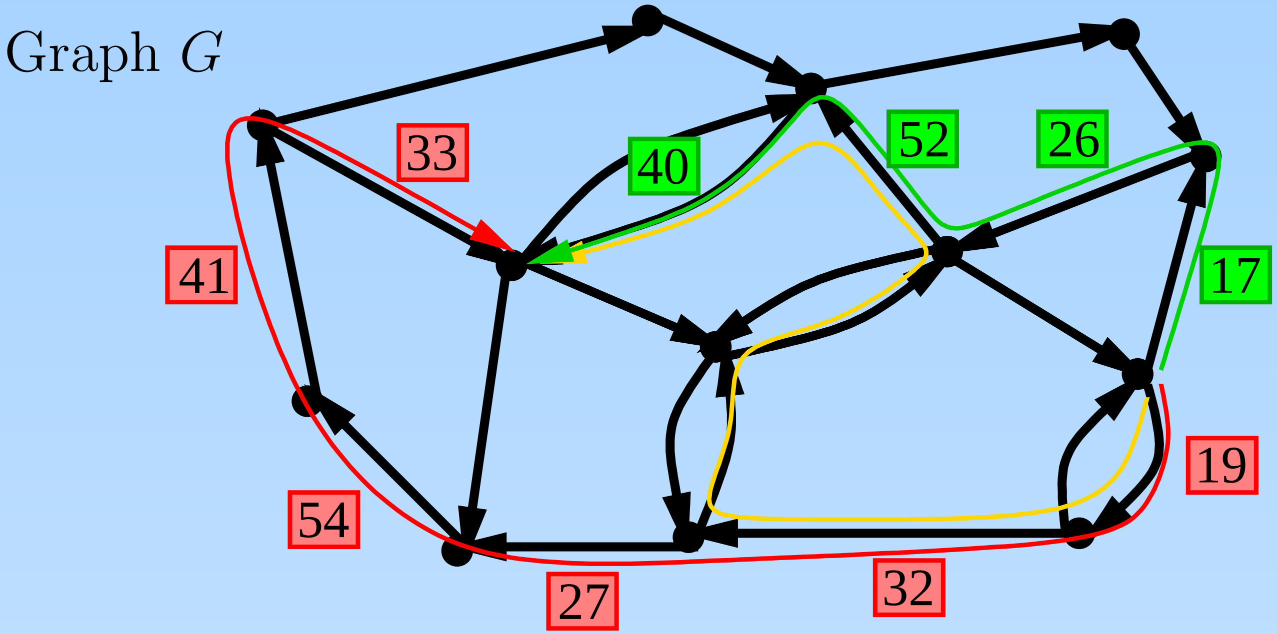
WTS-Strategies



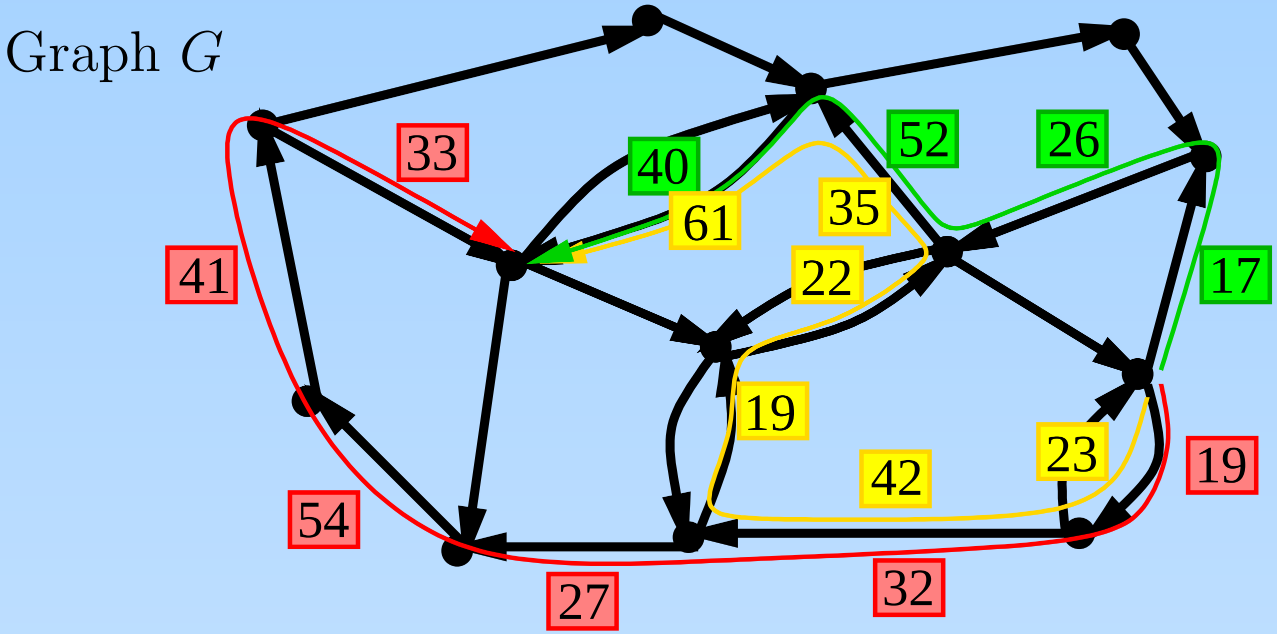
WTS-Strategies



WTS-Strategies

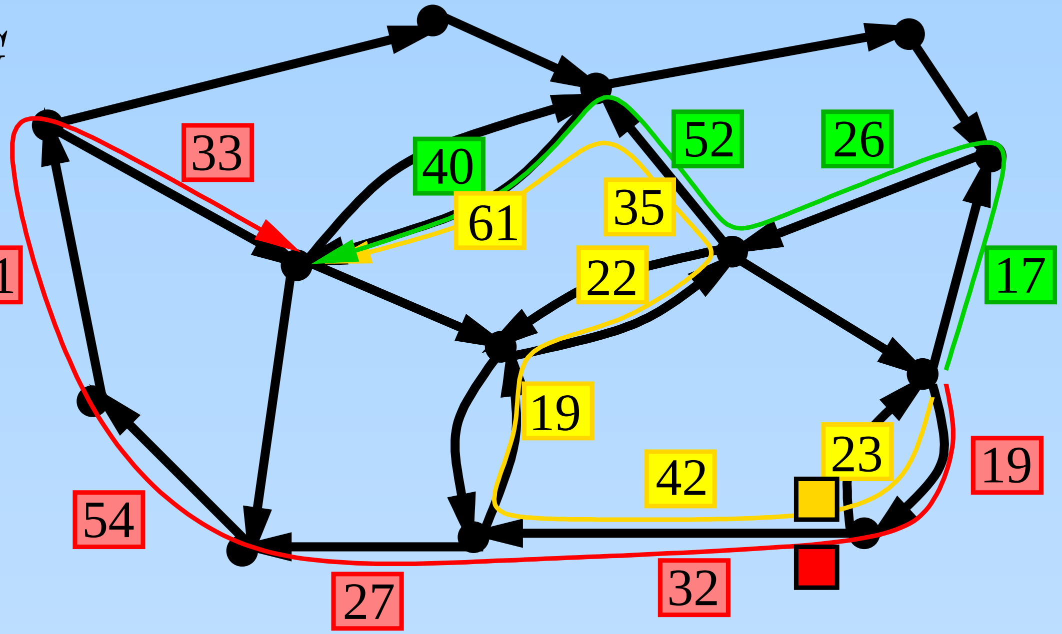


WTS-Strategies

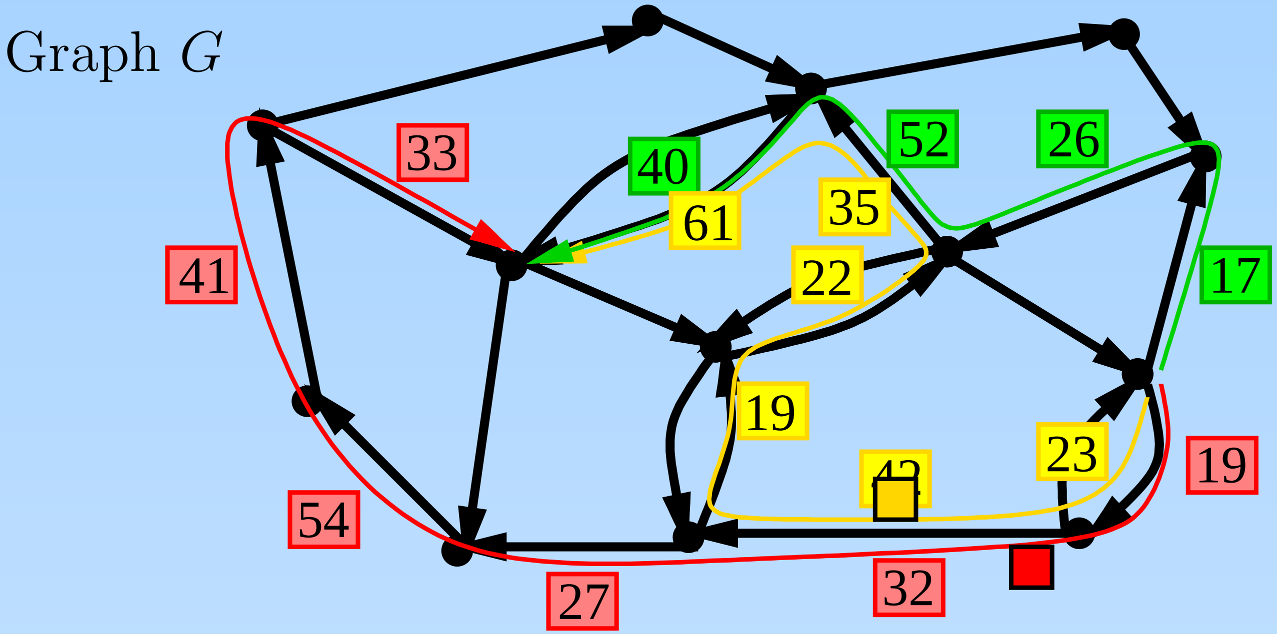


WTS-Strategies

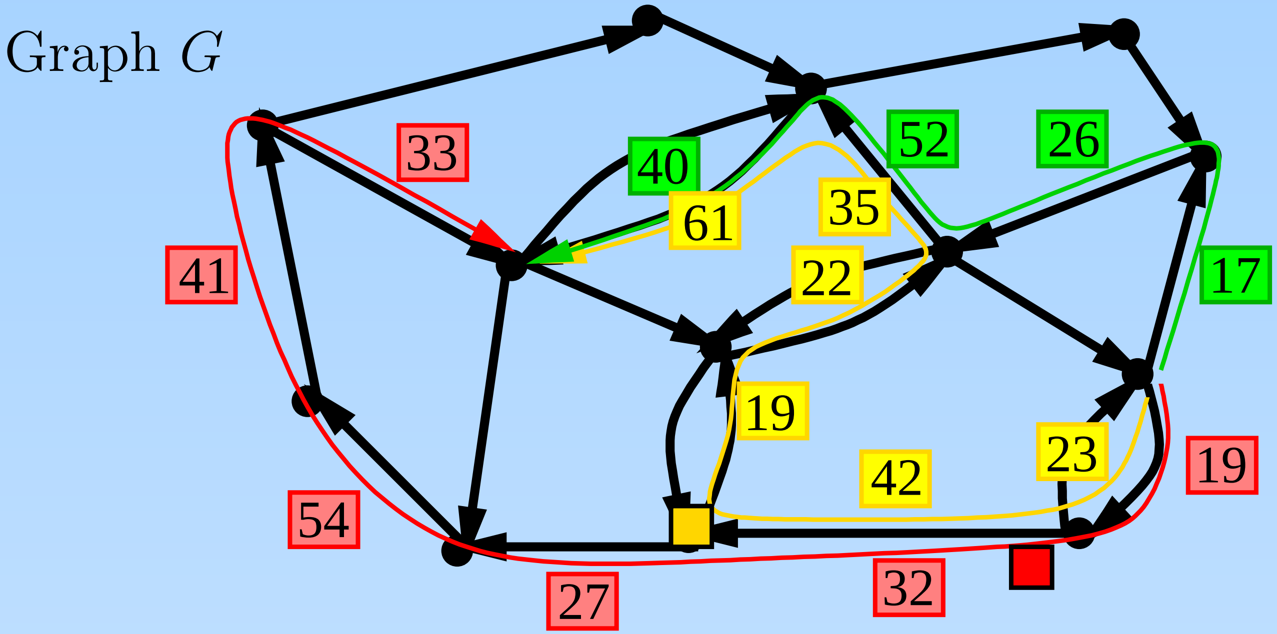
Graph G



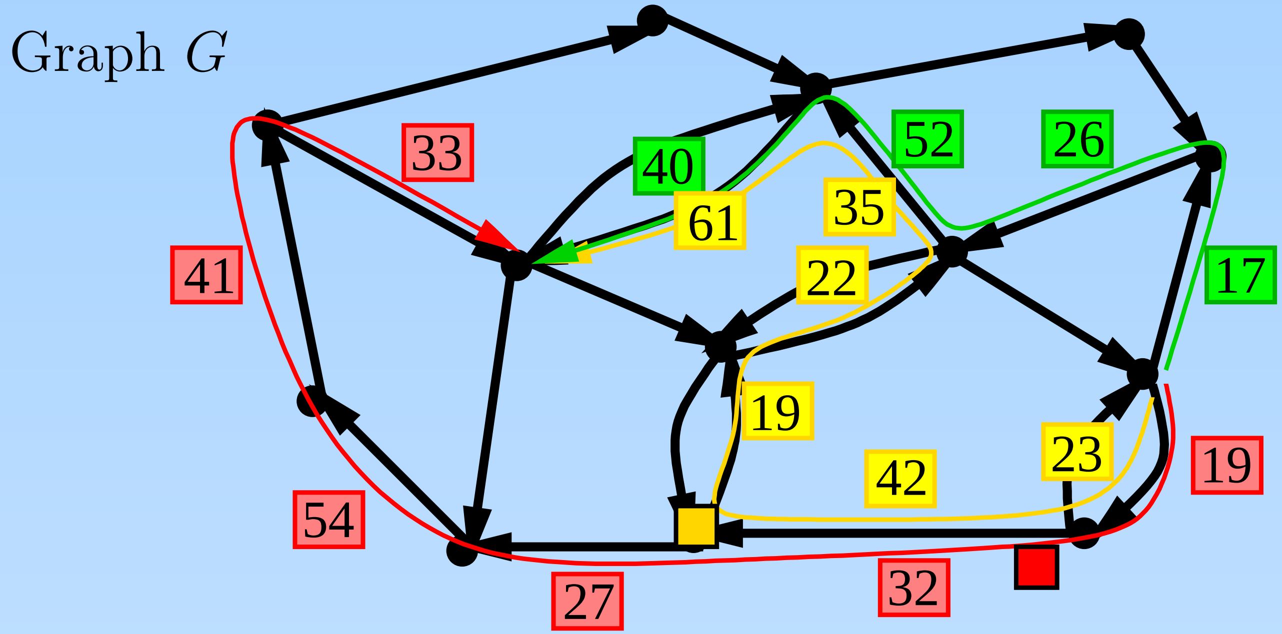
WTS-Strategies



WTS-Strategies



WTS-Strategies



We seek a negative result about all WTS-strategies.

Results

Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.

Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.

- New technique for proving 1-stability of WTS-strategies.

Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.
- New technique for proving 1-stability of WTS-strategies.
- Complete classification of universally stable and 1-stable *distance based* WTS-strategies.

Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.

- New technique for proving 1-stability of WTS-strategies.

- Complete classification of universally stable and 1-stable *distance based* WTS-strategies.
  - Priorities of the form $f(|P|, a)$.

Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.

- New technique for proving 1-stability of WTS-strategies.

- Complete classification of universally stable and 1-stable *distance based* WTS-strategies.
  - Priorities of the form $f(|P|, a)$.
  - 1-stable if $\forall x, y : f(x, y) > f(x, y + 1)$

Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.
- New technique for proving 1-stability of WTS-strategies.
- Complete classification of universally stable and 1-stable *distance based* WTS-strategies.
  - Priorities of the form $f(|P|, a)$.
  - 1-stable if $\forall x, y : f(x, y) > f(x, y + 1)$
  - not even universally stable otherwise.

Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.
- New technique for proving 1-stability of WTS-strategies. (**cf. paper)**
- Complete classification of universally stable and 1-stable *distance based* WTS-strategies. (**cf. paper)**
  - Priorities of the form $f(|P|, a)$.
  - 1-stable if $\forall x, y : f(x, y) > f(x, y + 1)$
  - not even universally stable otherwise.

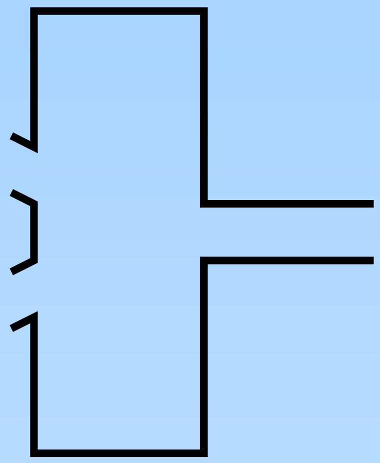
Results

- Every WTS-Strategy can be forced into total traffic and delays exponential in the size of the graph.

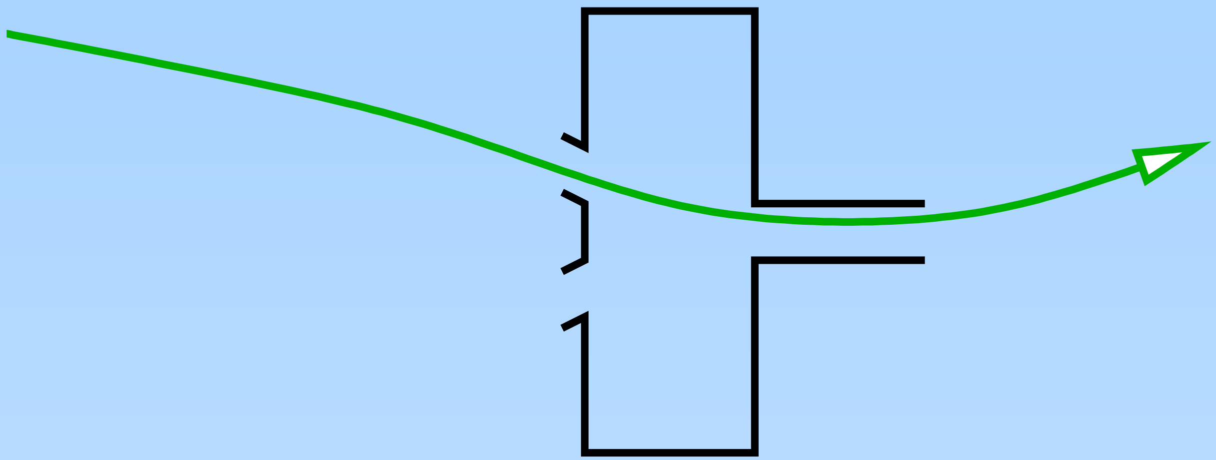
Problem: Provide a family of graphs, so that for every possible assignment of priorities, a jam of exponentially many packets can be created.

How to create a traffic jam

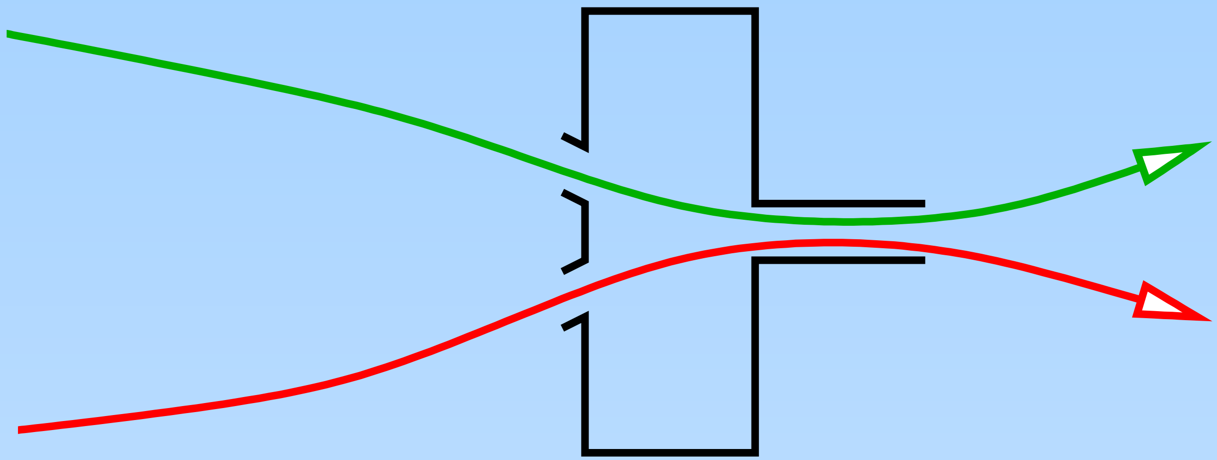
How to create a traffic jam



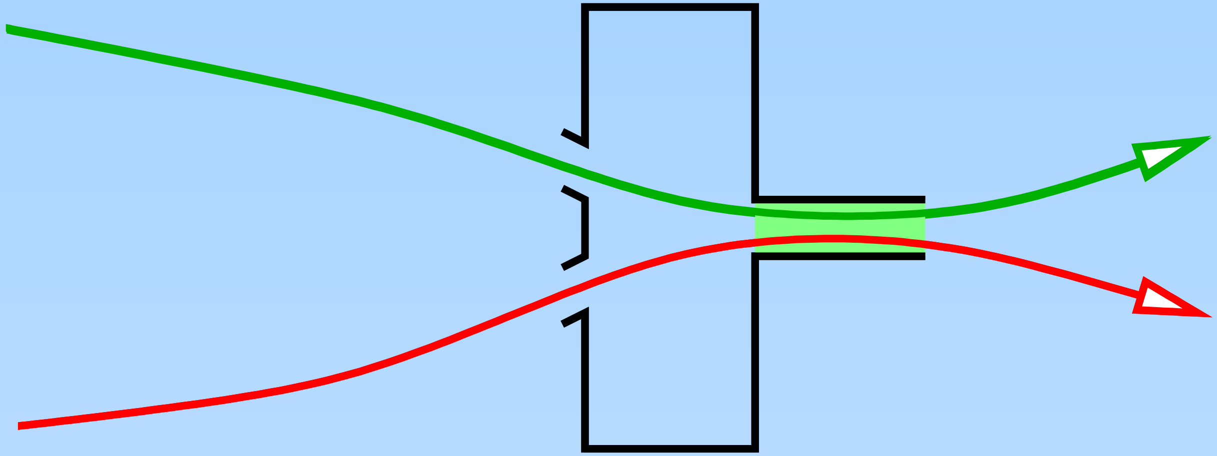
How to create a traffic jam



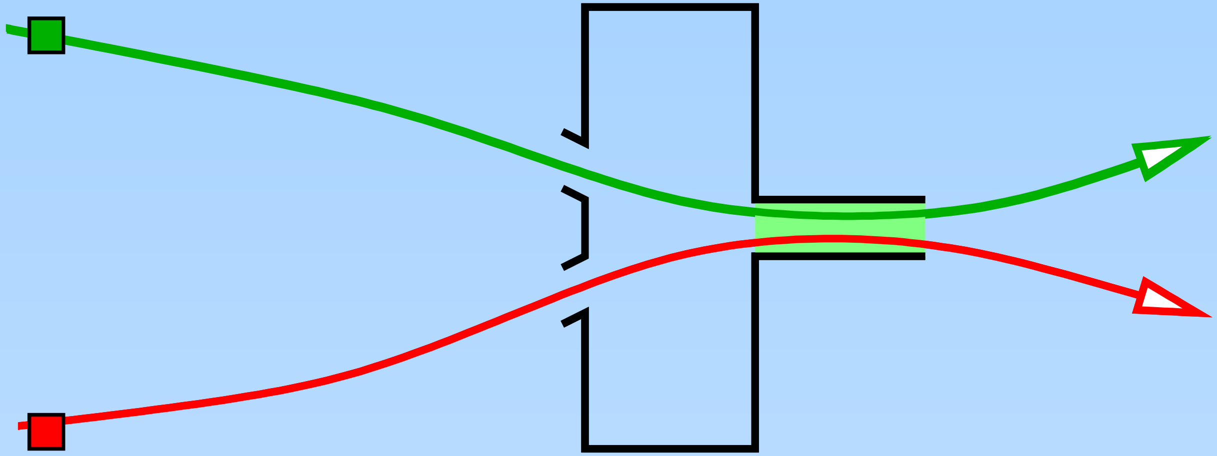
How to create a traffic jam



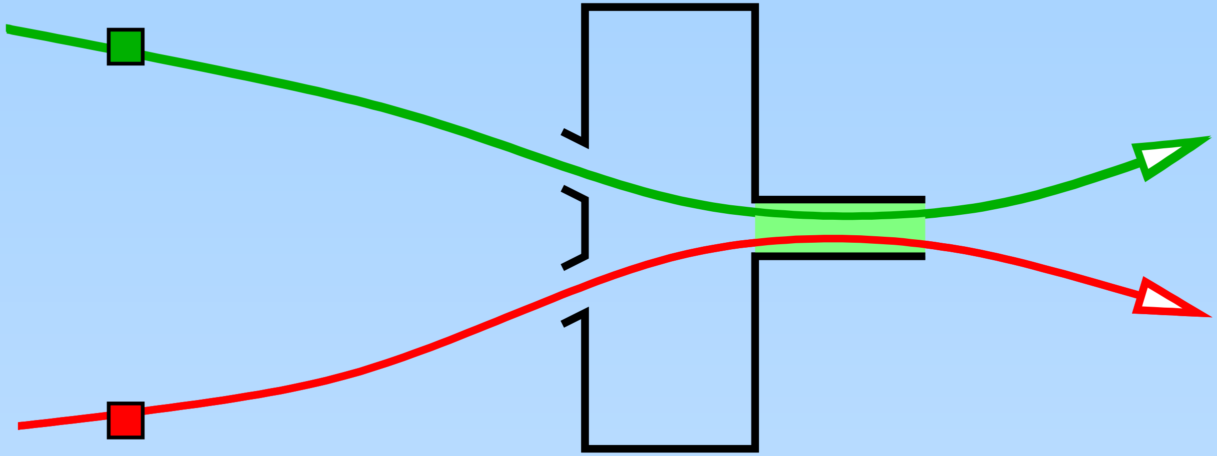
How to create a traffic jam



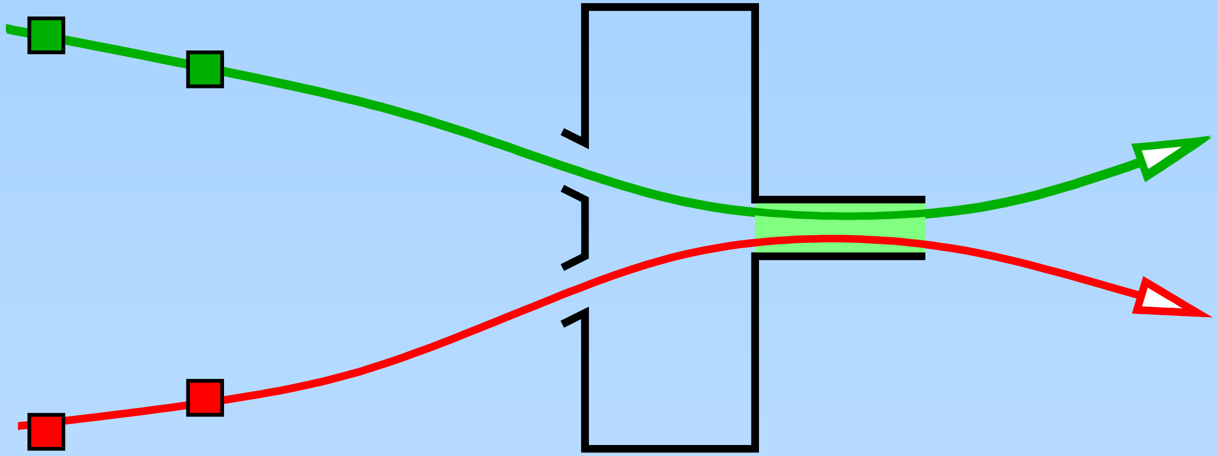
How to create a traffic jam



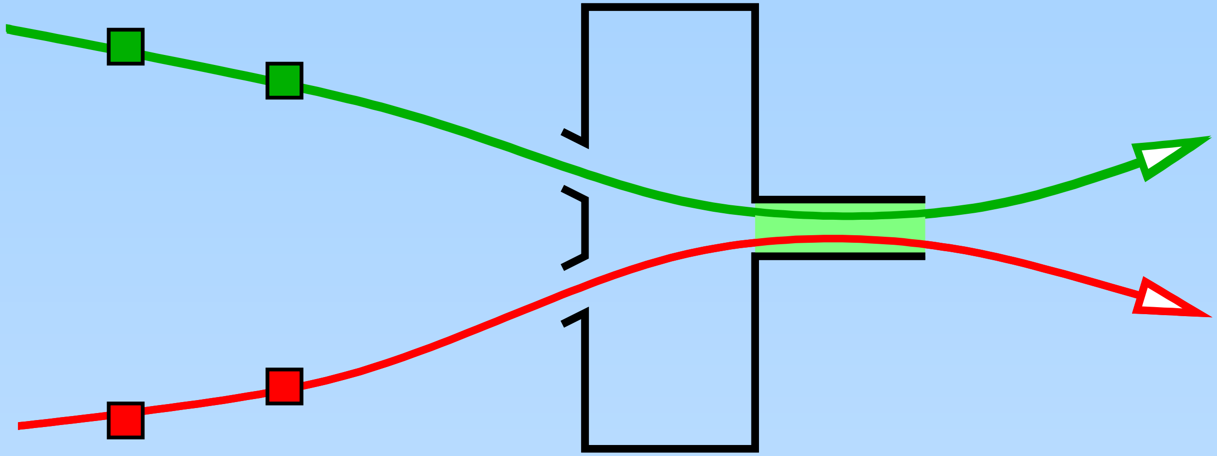
How to create a traffic jam



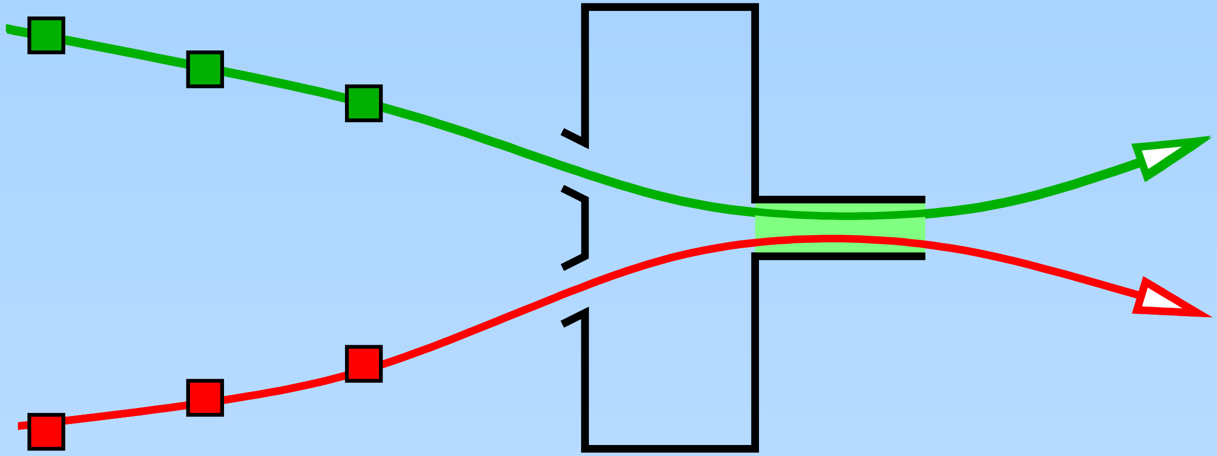
How to create a traffic jam



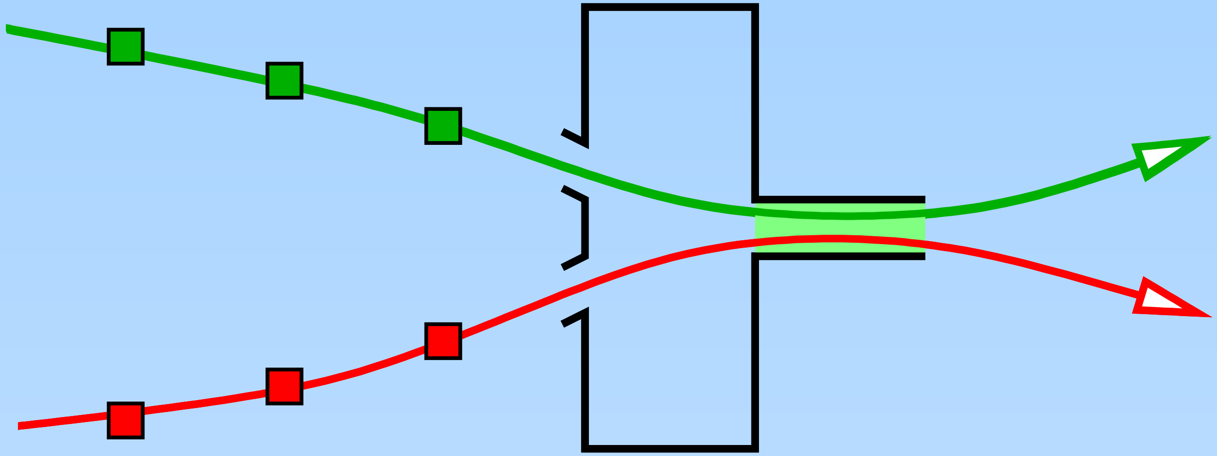
How to create a traffic jam



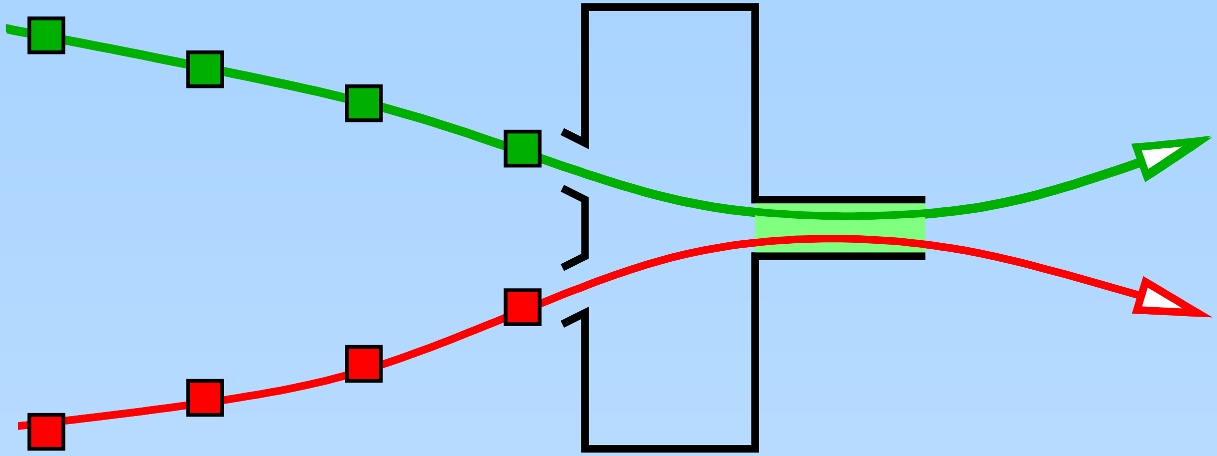
How to create a traffic jam



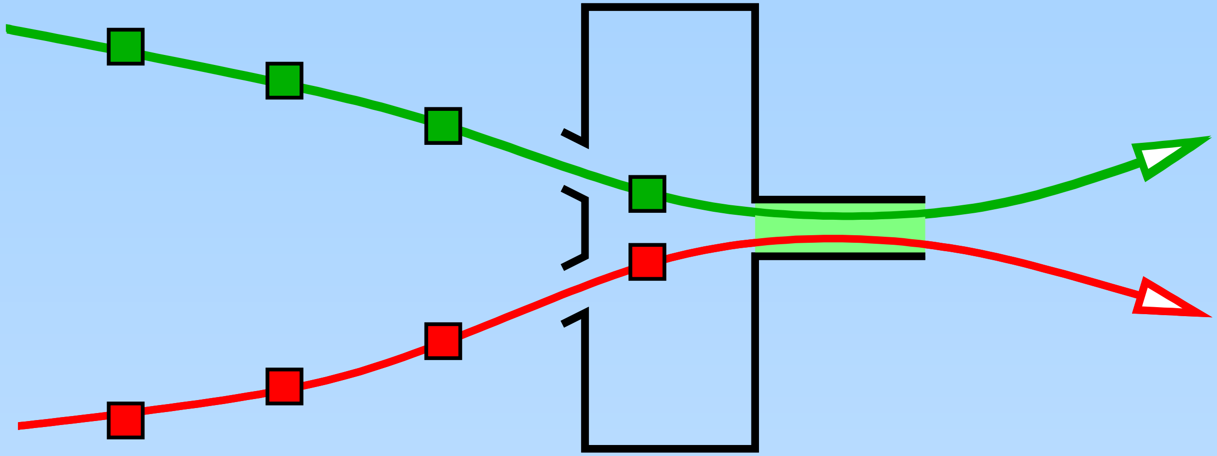
How to create a traffic jam



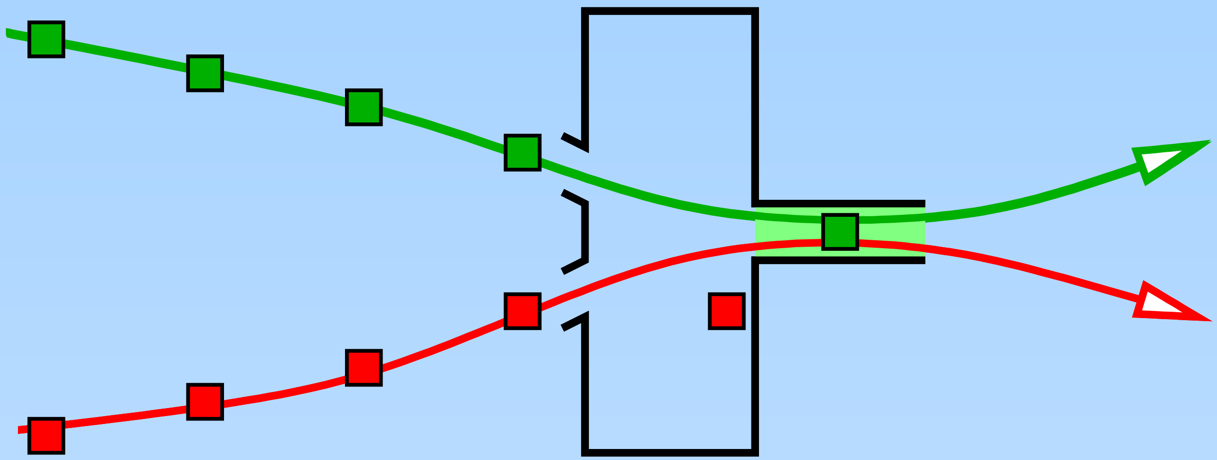
How to create a traffic jam



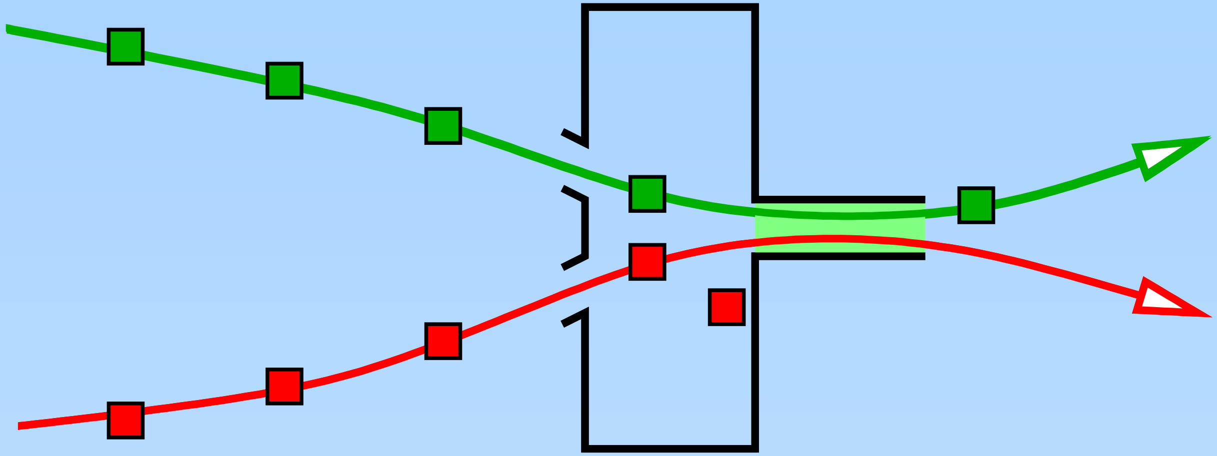
How to create a traffic jam



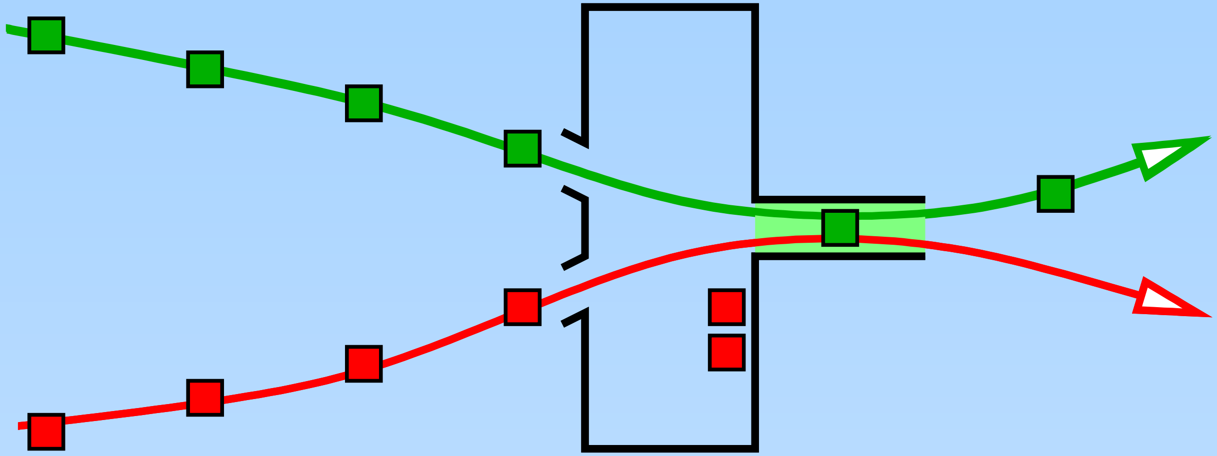
How to create a traffic jam



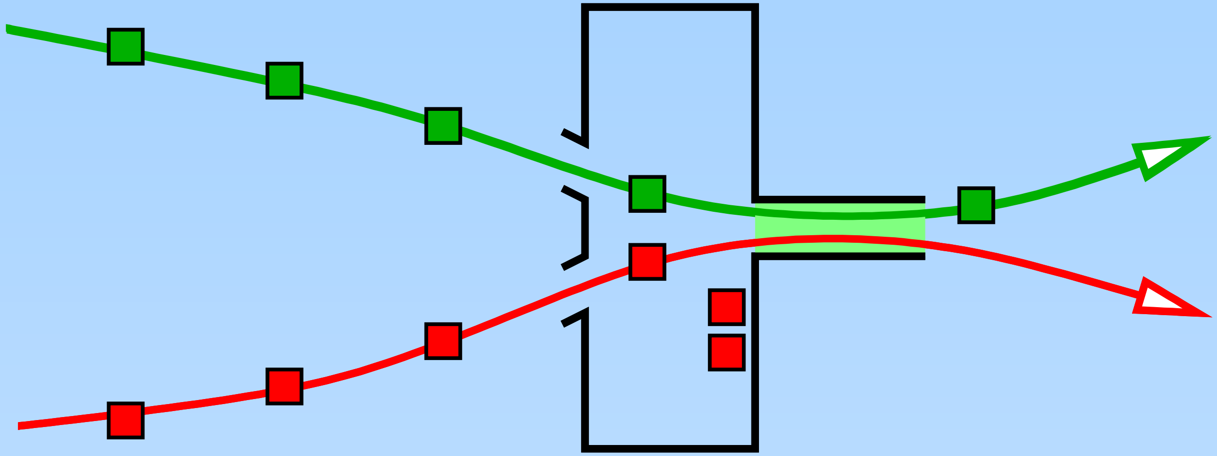
How to create a traffic jam



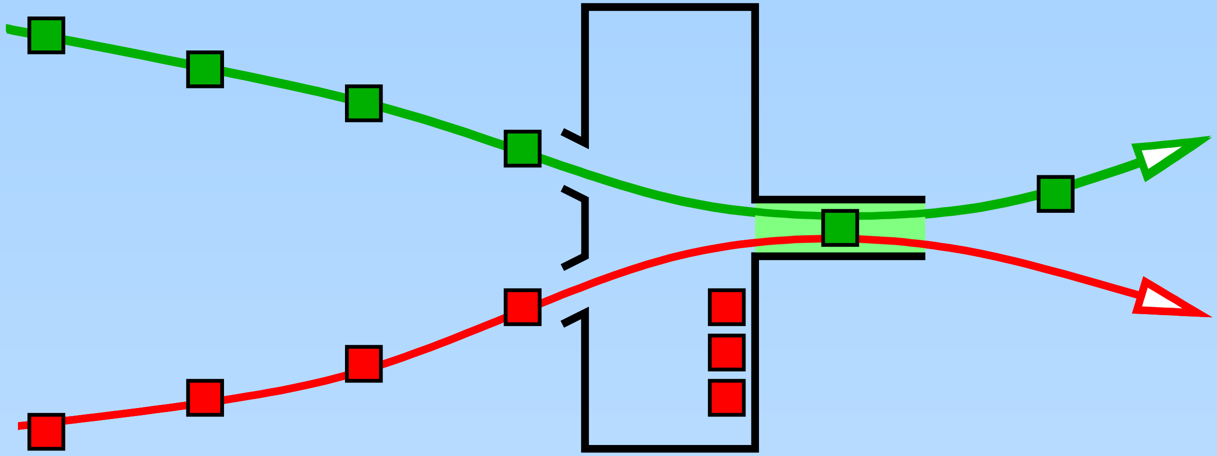
How to create a traffic jam



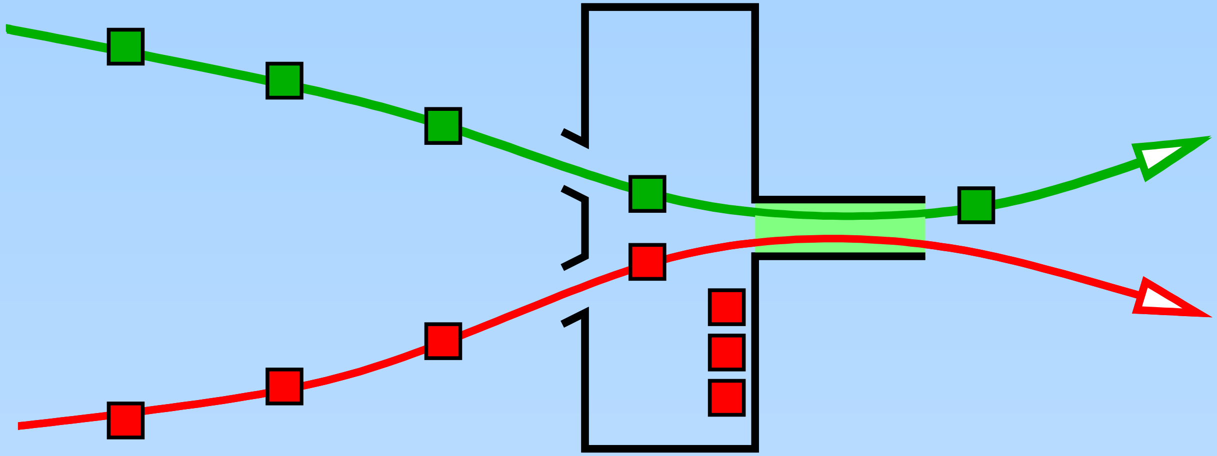
How to create a traffic jam



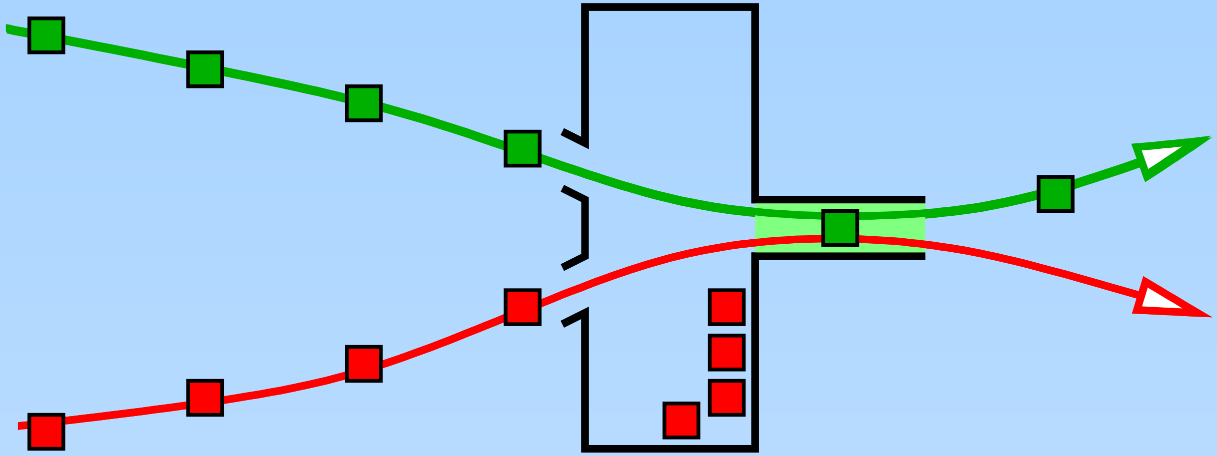
How to create a traffic jam



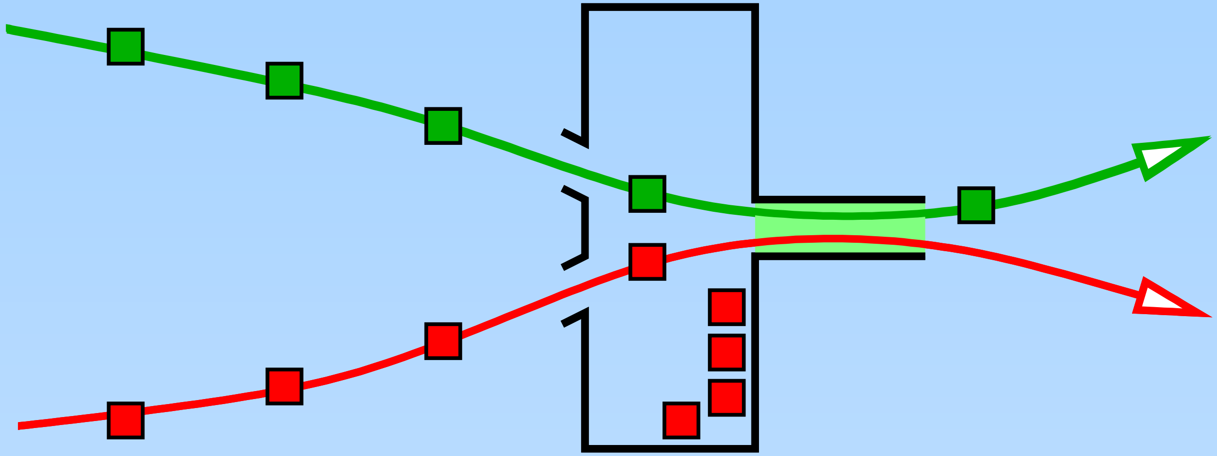
How to create a traffic jam



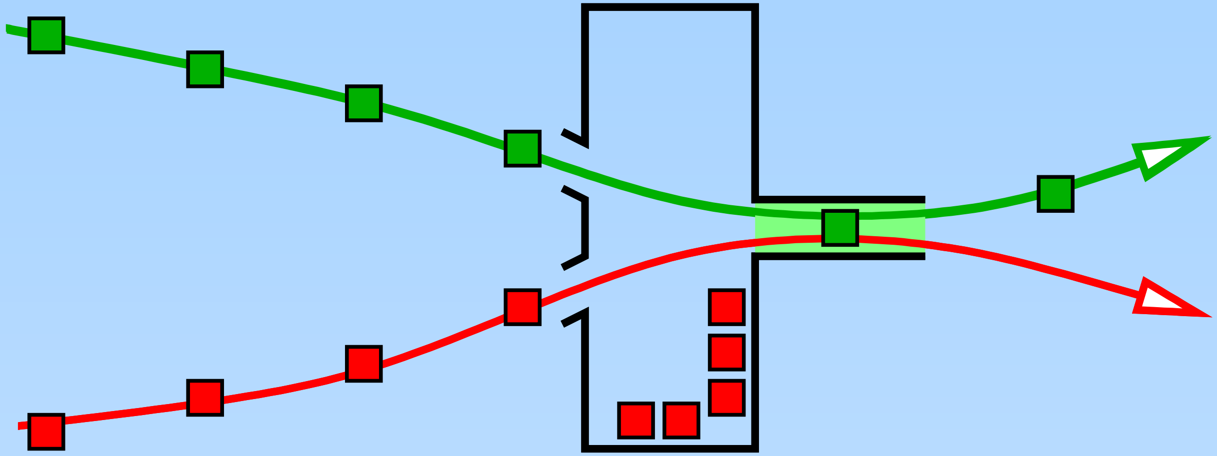
How to create a traffic jam



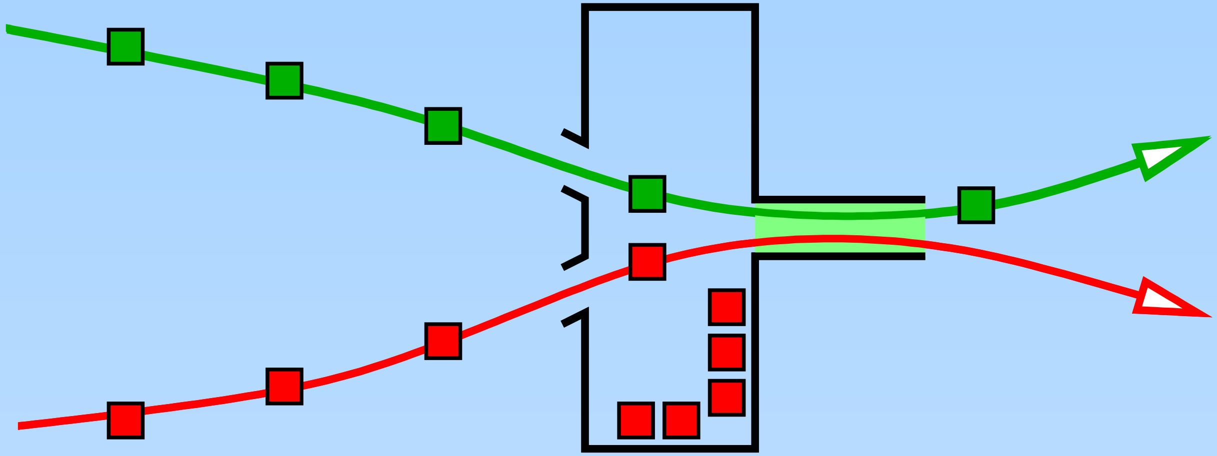
How to create a traffic jam



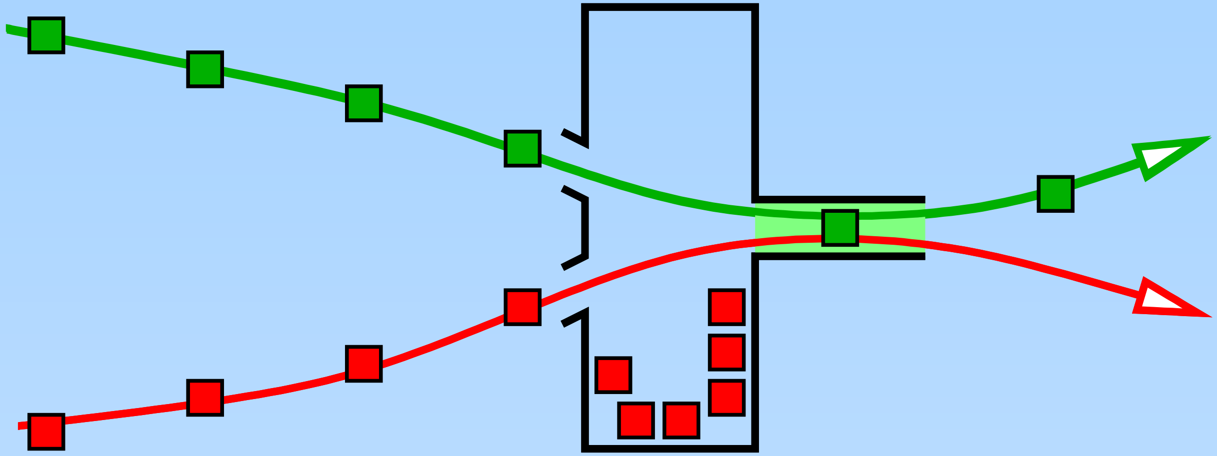
How to create a traffic jam



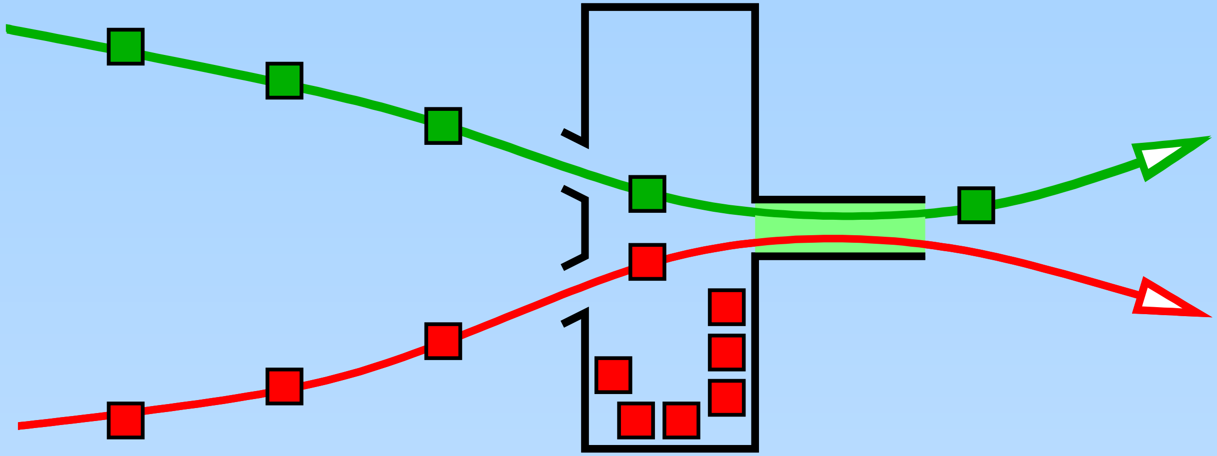
How to create a traffic jam



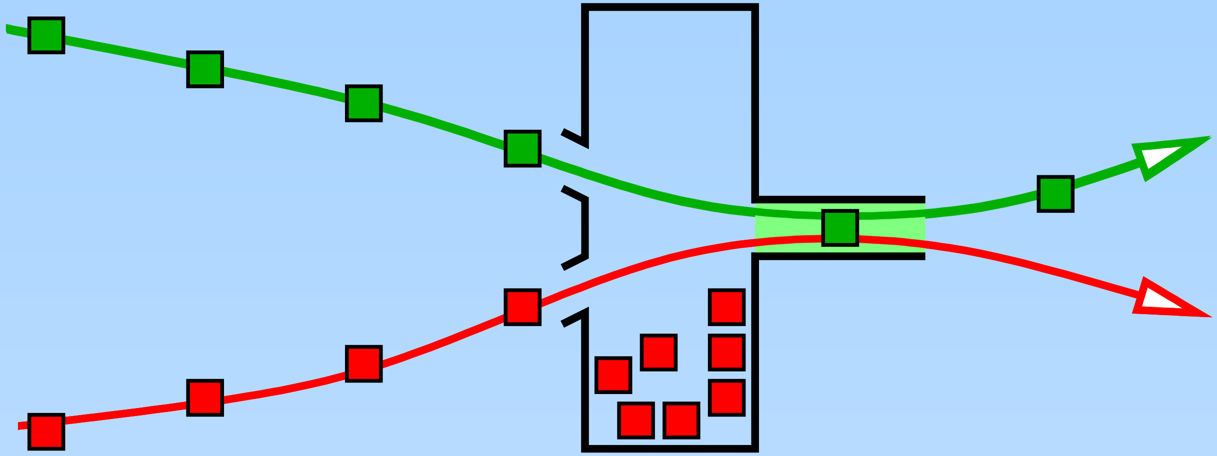
How to create a traffic jam



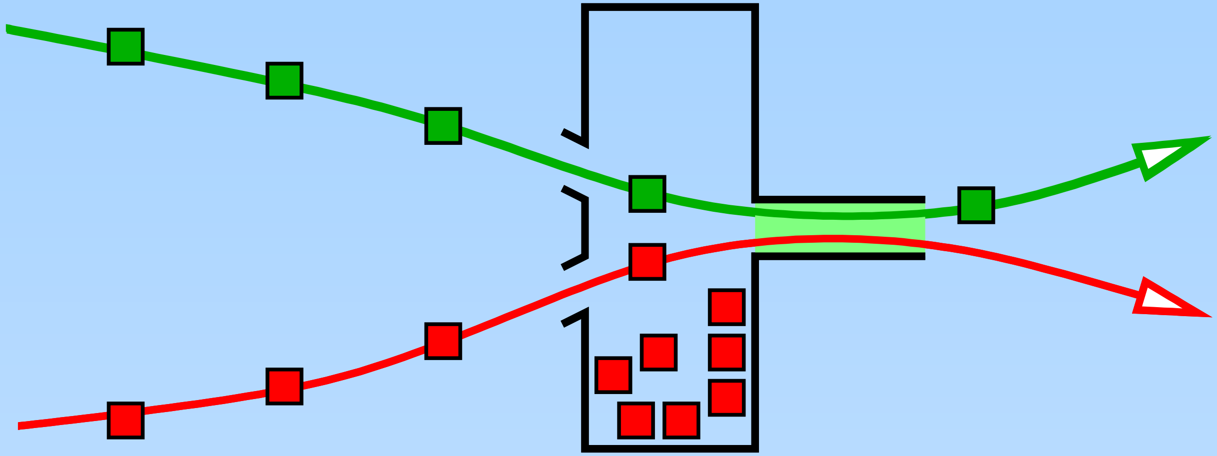
How to create a traffic jam



How to create a traffic jam

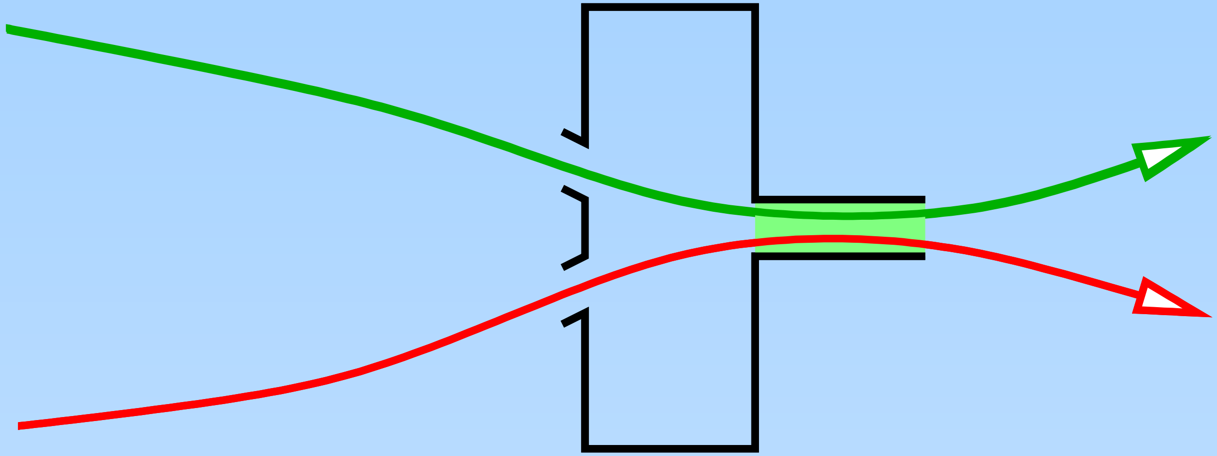


How to create a traffic jam



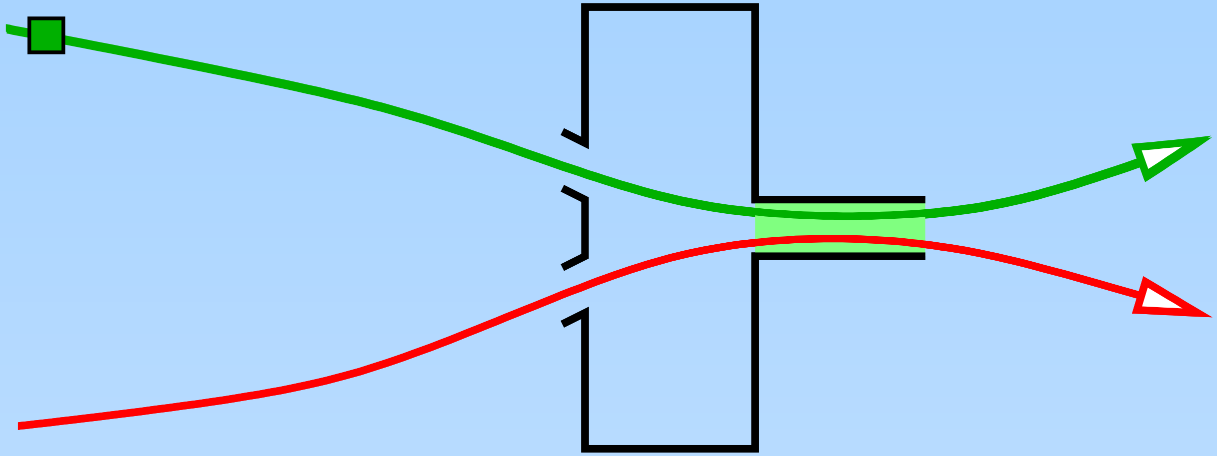
Joint edge is overloaded.

How to create a traffic jam



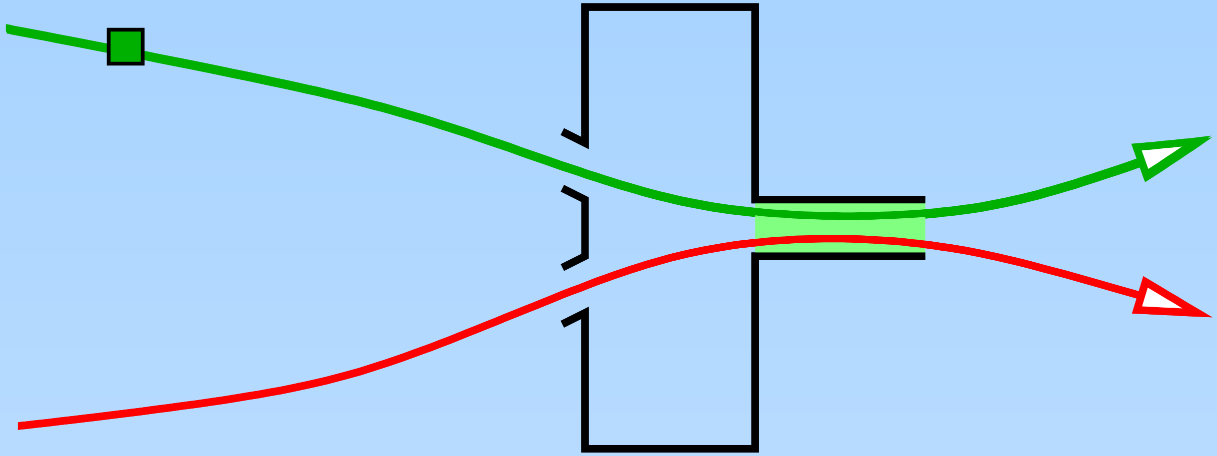
Joint edge is overloaded.

How to create a traffic jam



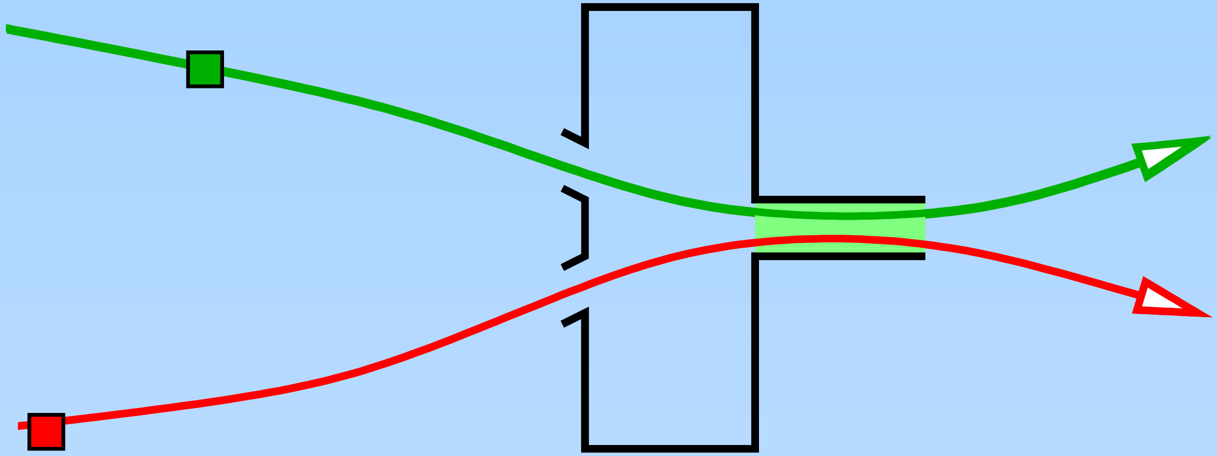
Joint edge is overloaded.

How to create a traffic jam



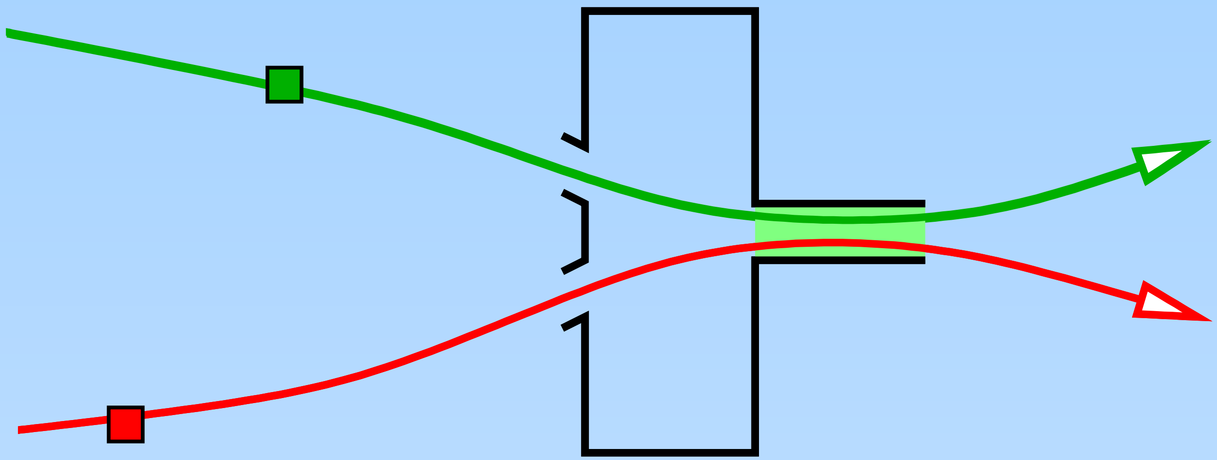
Joint edge is overloaded.

How to create a traffic jam



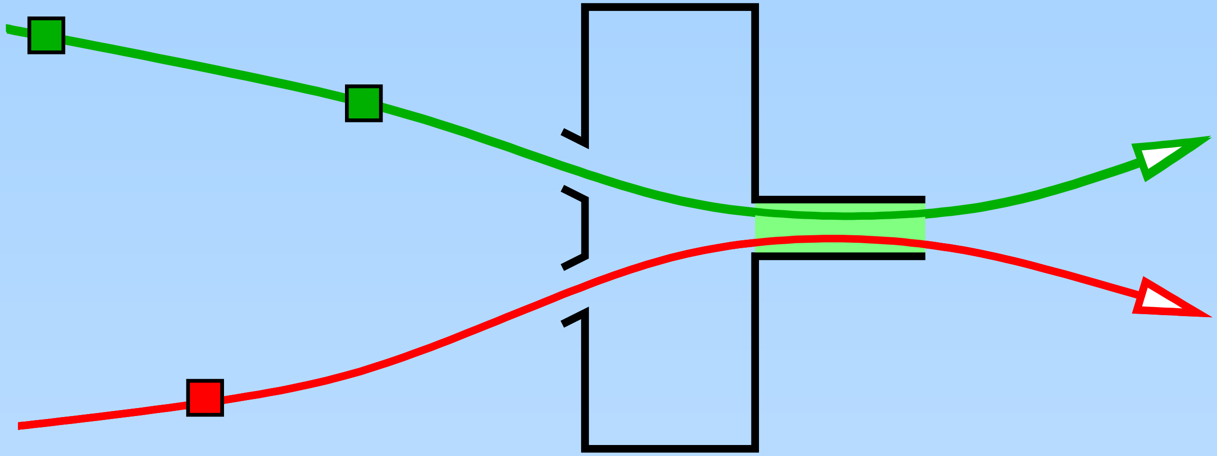
Joint edge is overloaded.

How to create a traffic jam



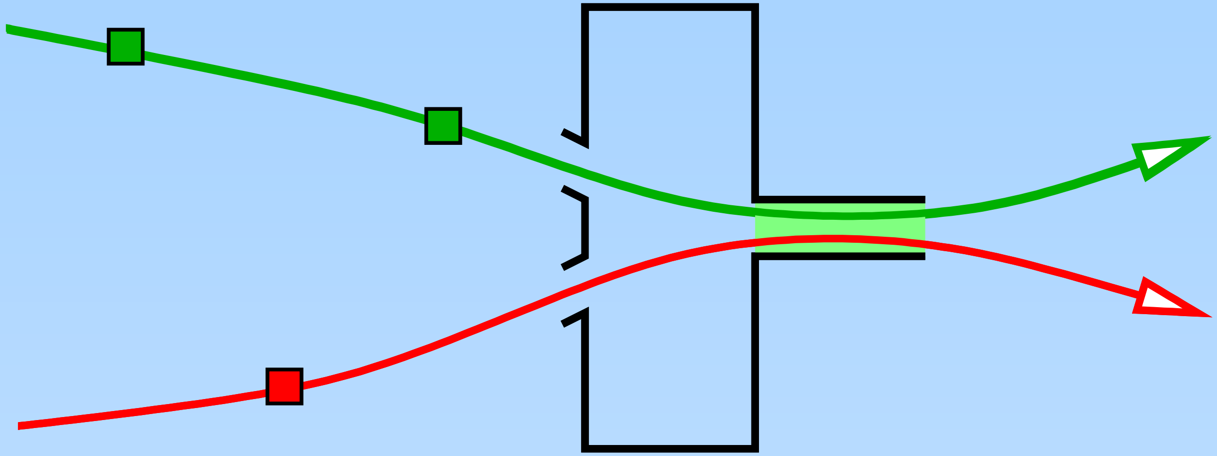
Joint edge is overloaded.

How to create a traffic jam



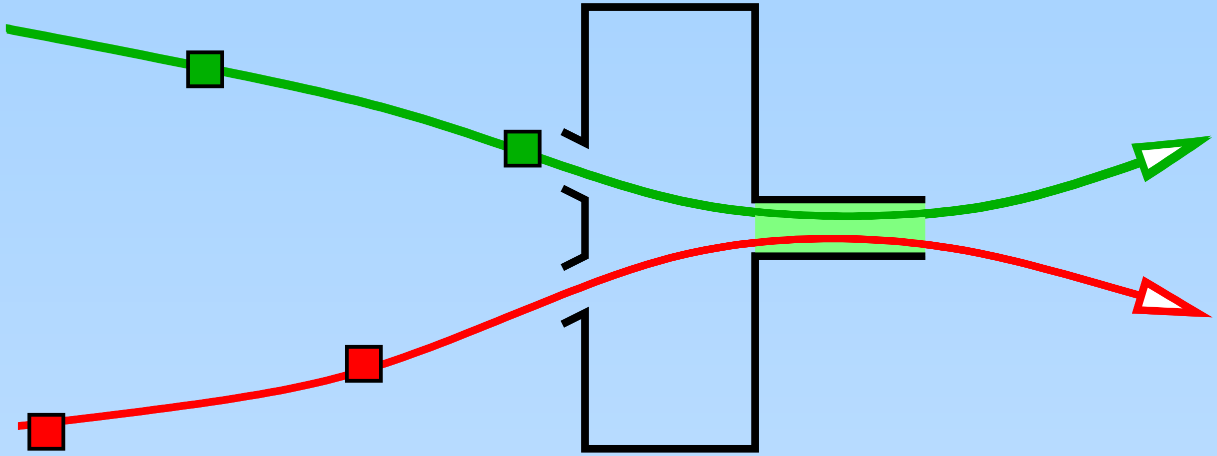
Joint edge is overloaded.

How to create a traffic jam



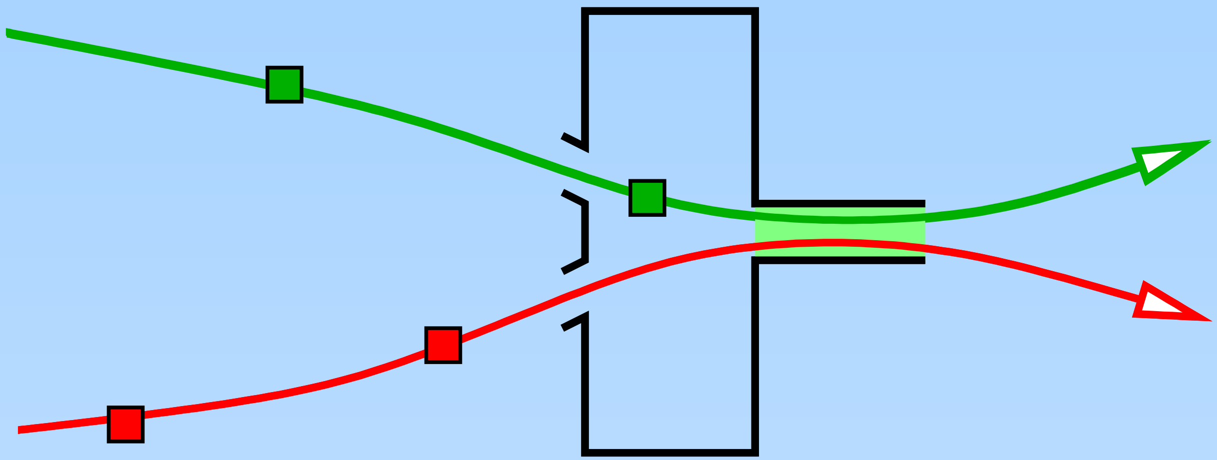
Joint edge is overloaded.

How to create a traffic jam



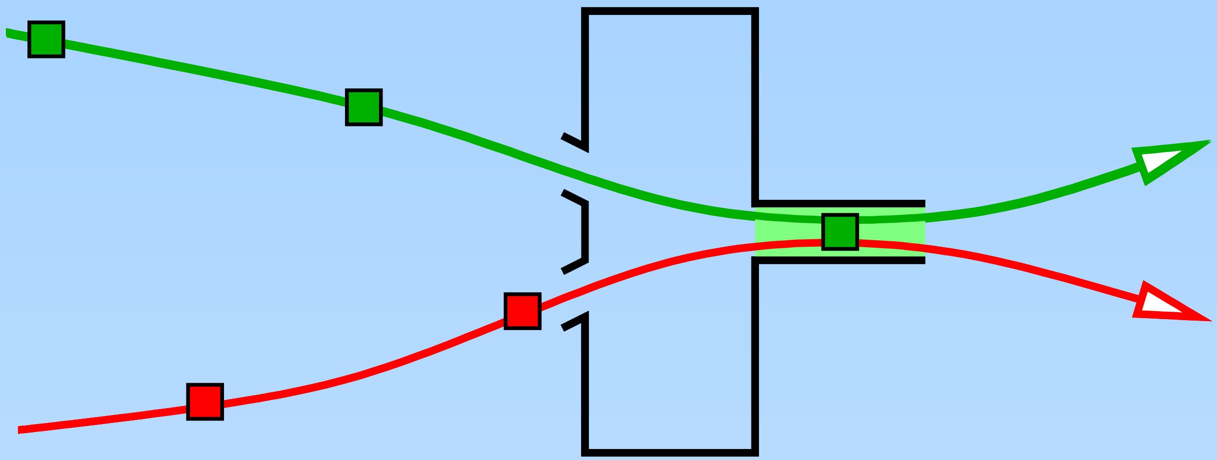
Joint edge is overloaded.

How to create a traffic jam



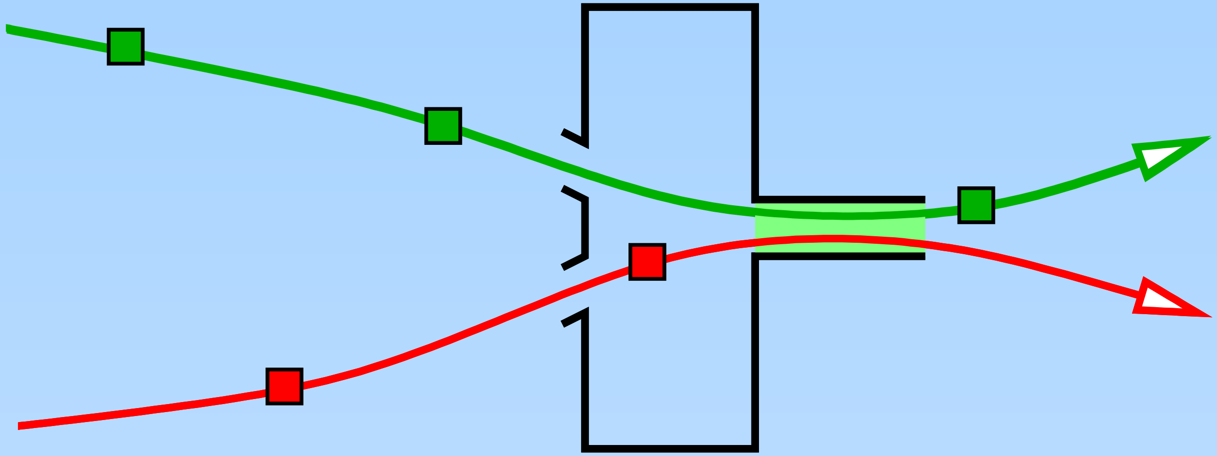
Joint edge is overloaded.

How to create a traffic jam



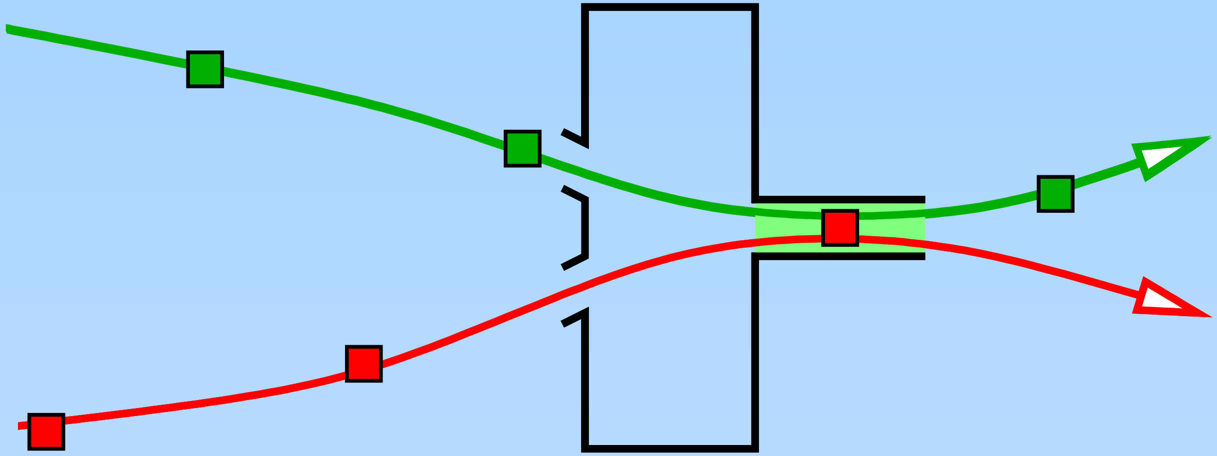
Joint edge is overloaded.

How to create a traffic jam



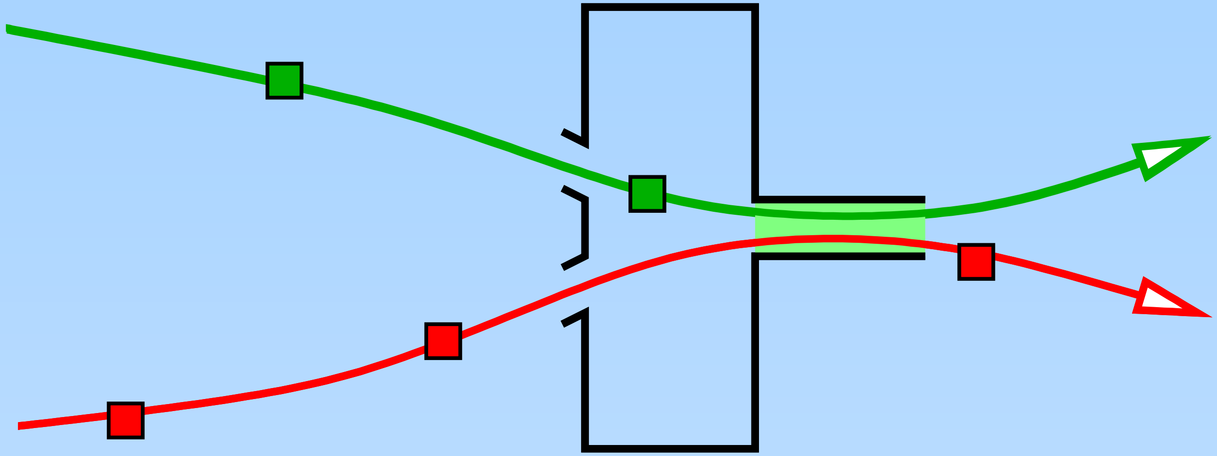
Joint edge is overloaded.

How to create a traffic jam



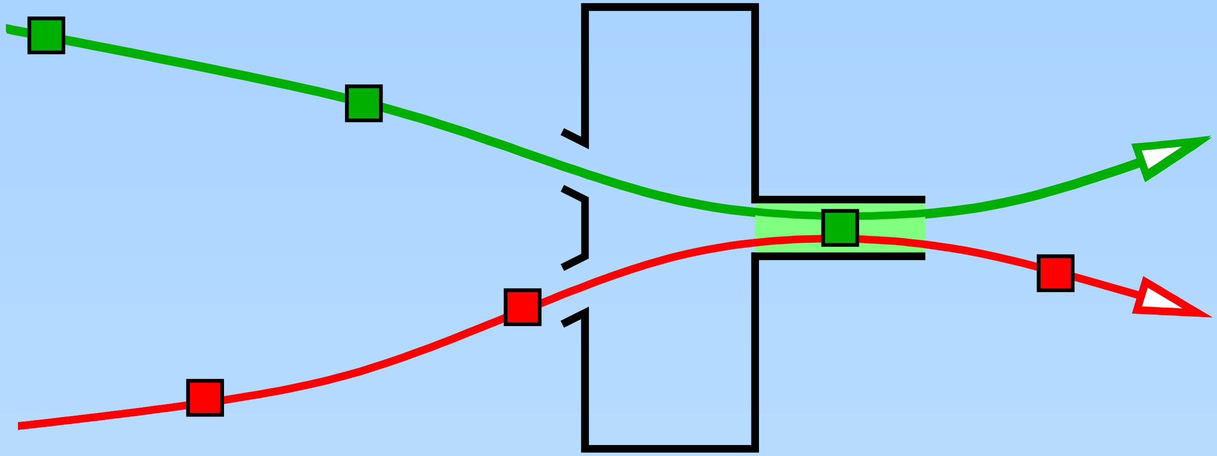
Joint edge is overloaded.

How to create a traffic jam



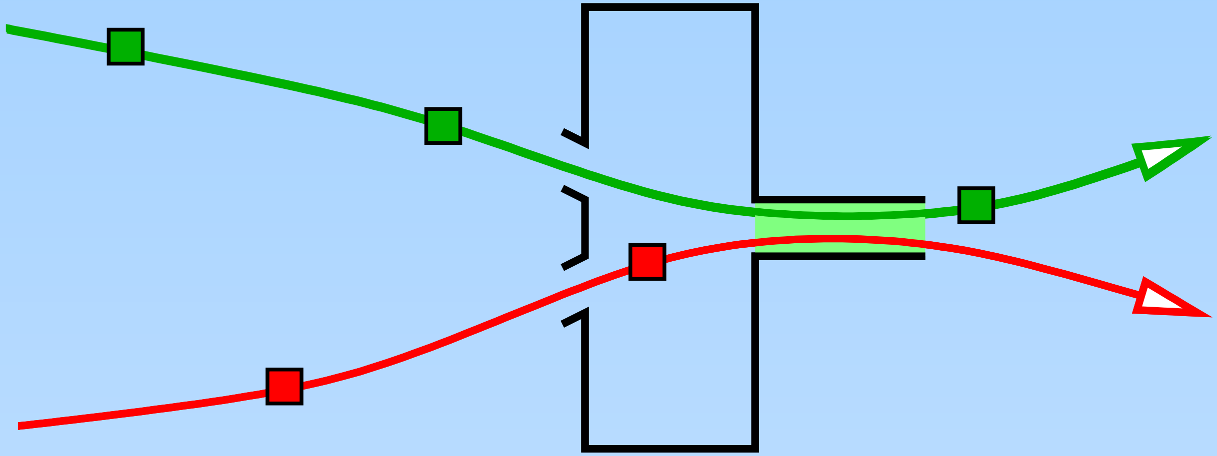
Joint edge is overloaded.

How to create a traffic jam



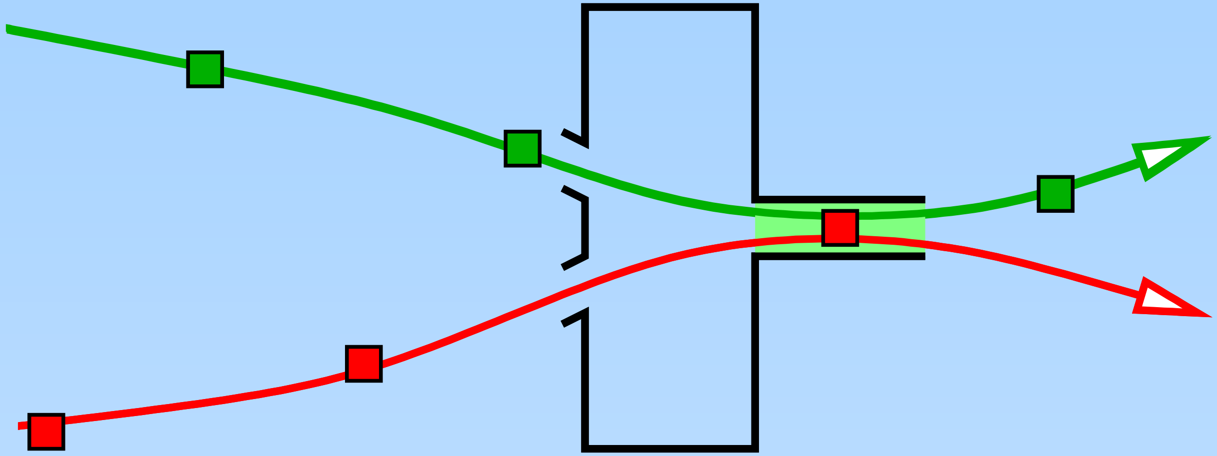
Joint edge is overloaded.

How to create a traffic jam



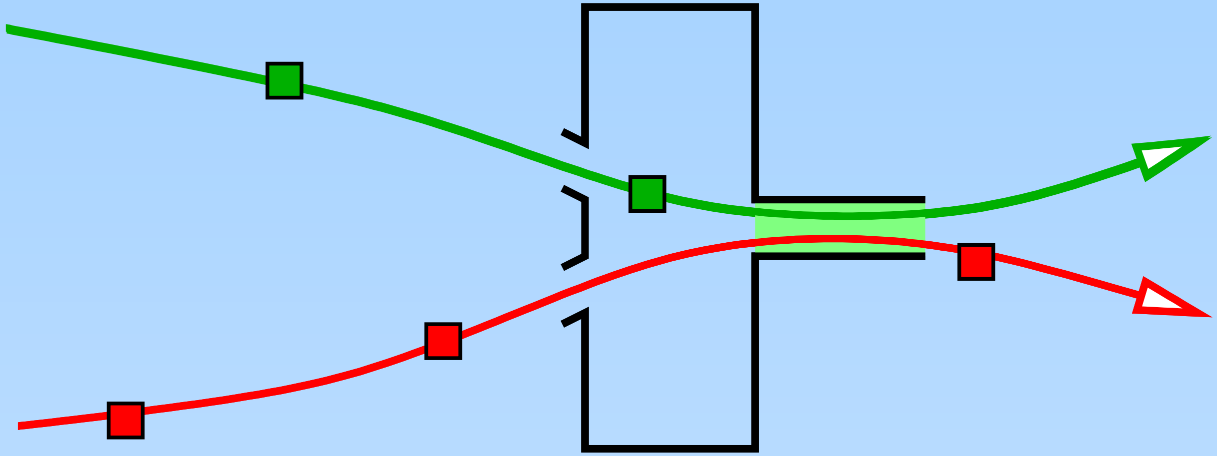
Joint edge is overloaded.

How to create a traffic jam



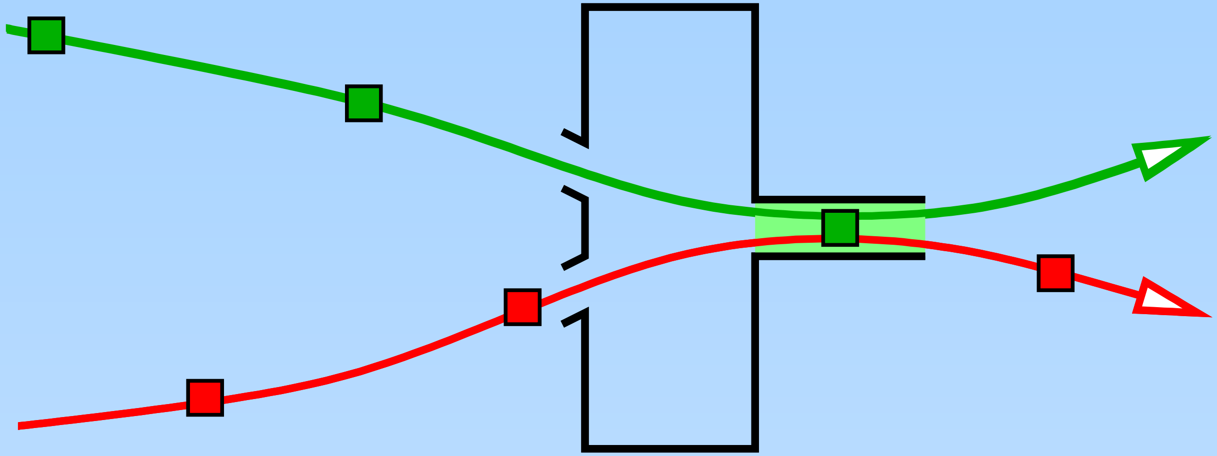
Joint edge is overloaded.

How to create a traffic jam



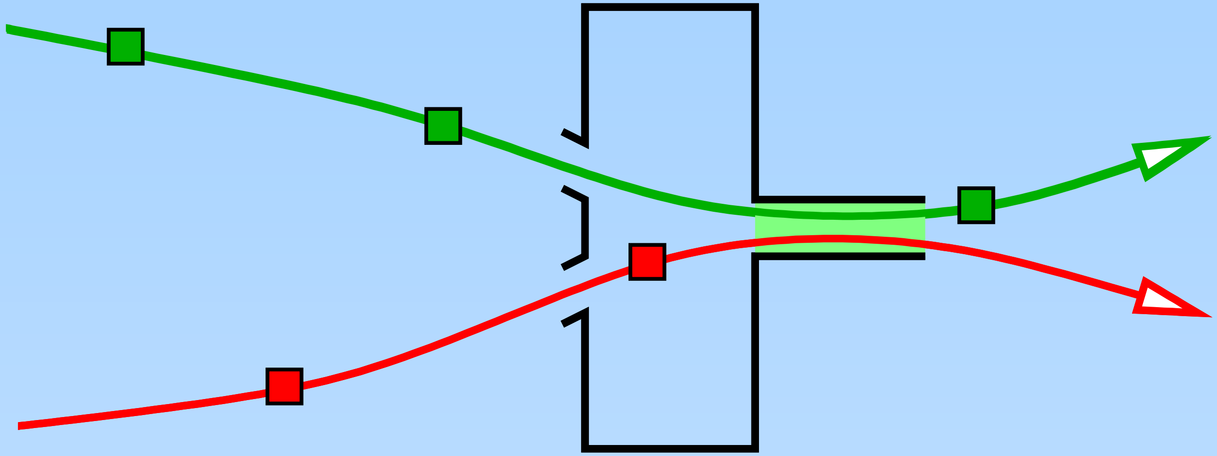
Joint edge is overloaded.

How to create a traffic jam



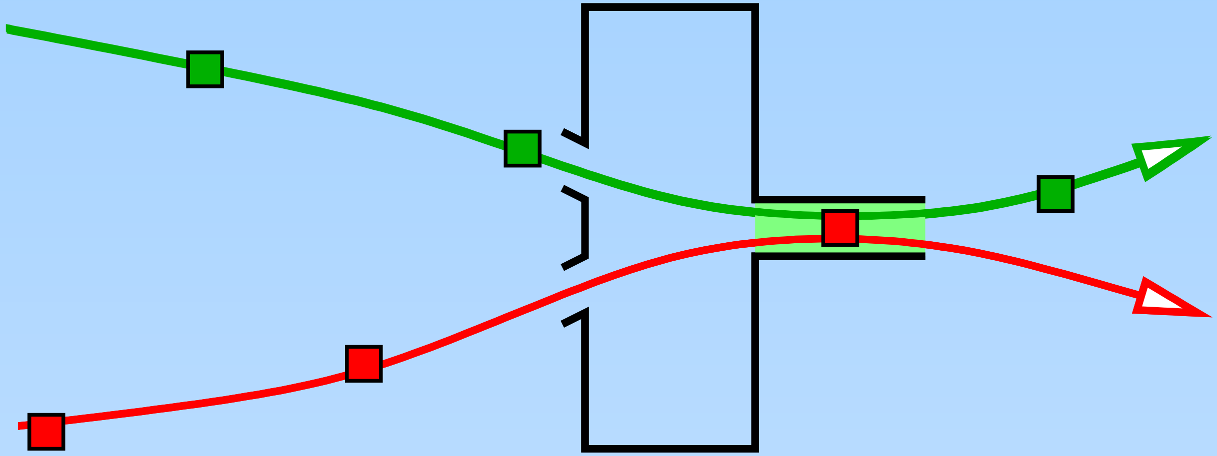
Joint edge is overloaded.

How to create a traffic jam



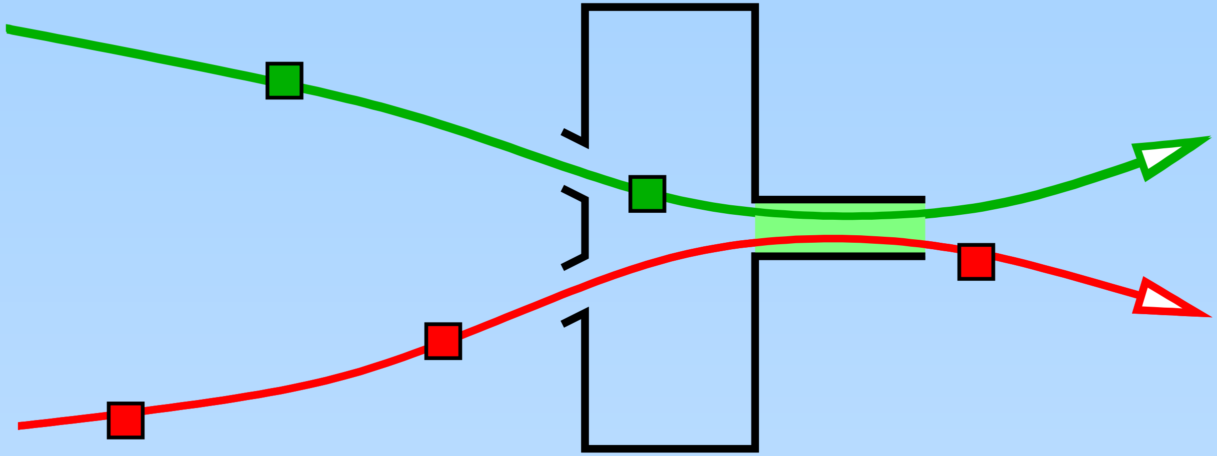
Joint edge is overloaded.

How to create a traffic jam



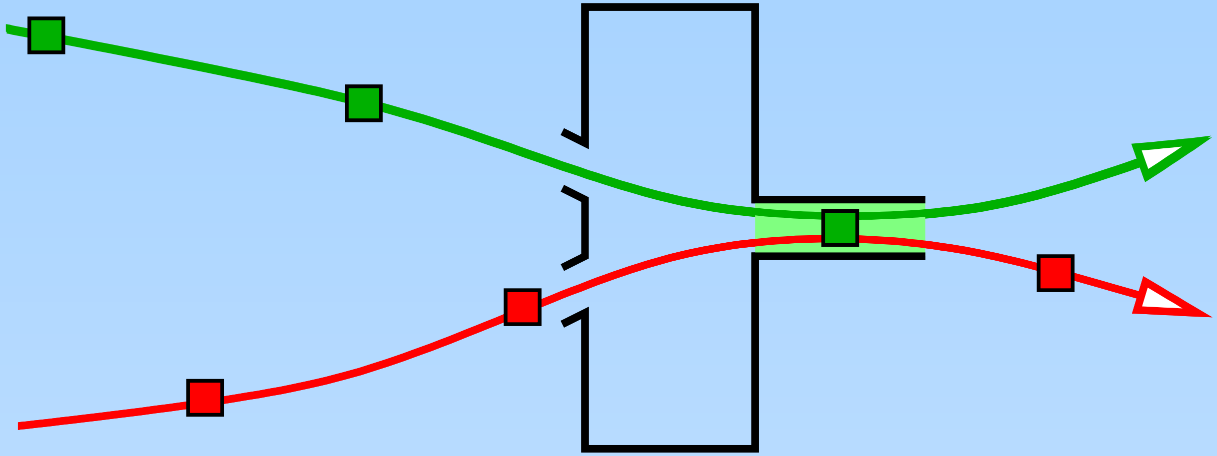
Joint edge is overloaded.

How to create a traffic jam



Joint edge is overloaded.

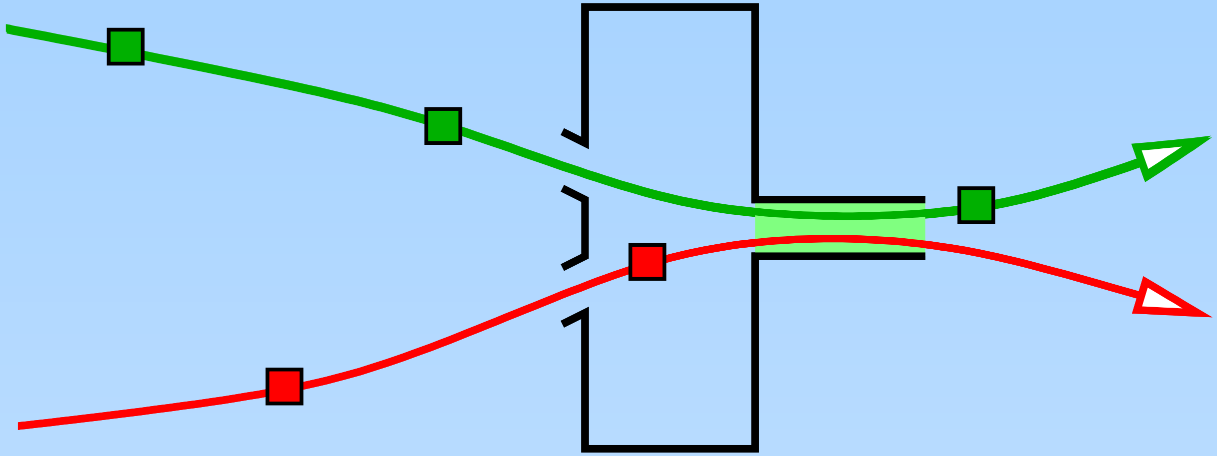
How to create a traffic jam



Joint edge is overloaded.

For legal insertions no jam is created.

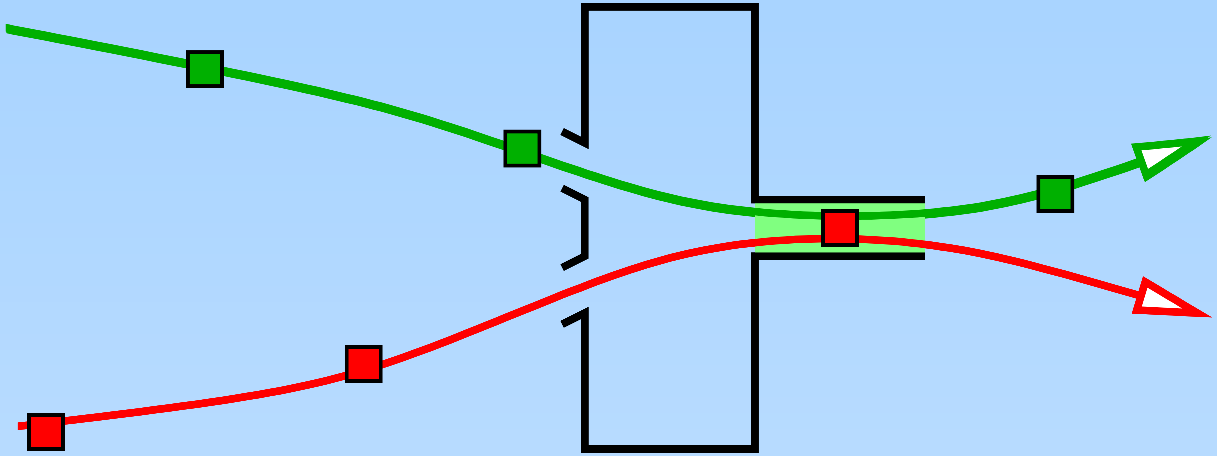
How to create a traffic jam



Joint edge is overloaded.

For legal insertions no jam is created.

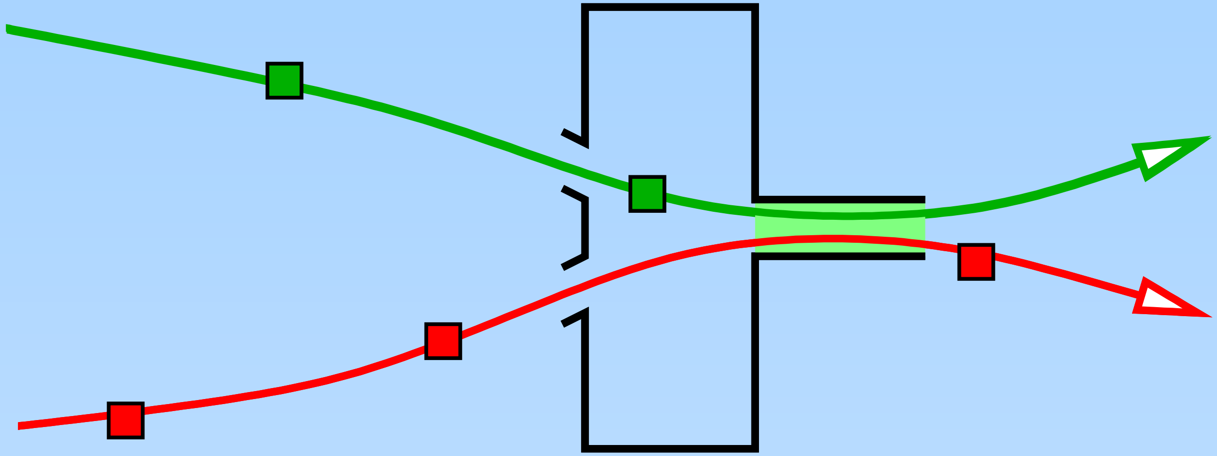
How to create a traffic jam



Joint edge is overloaded.

For legal insertions no jam is created.

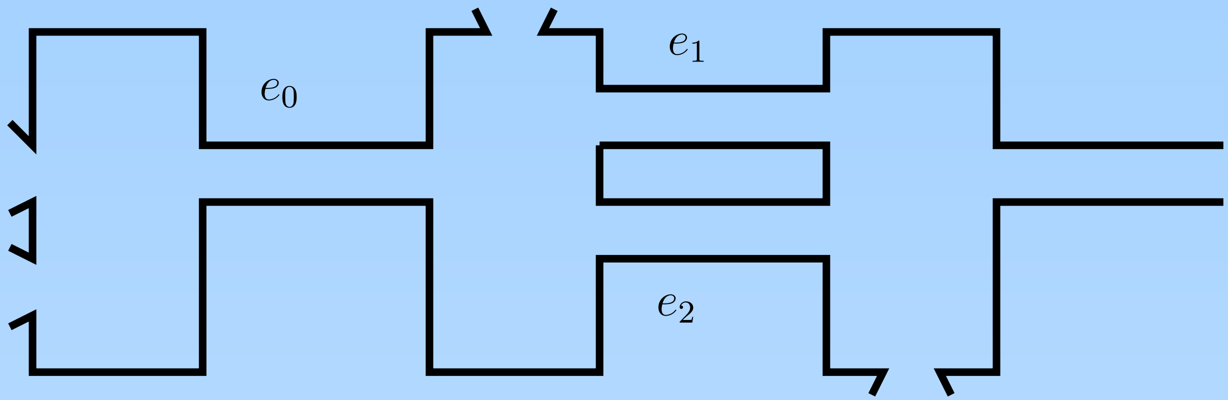
How to create a traffic jam



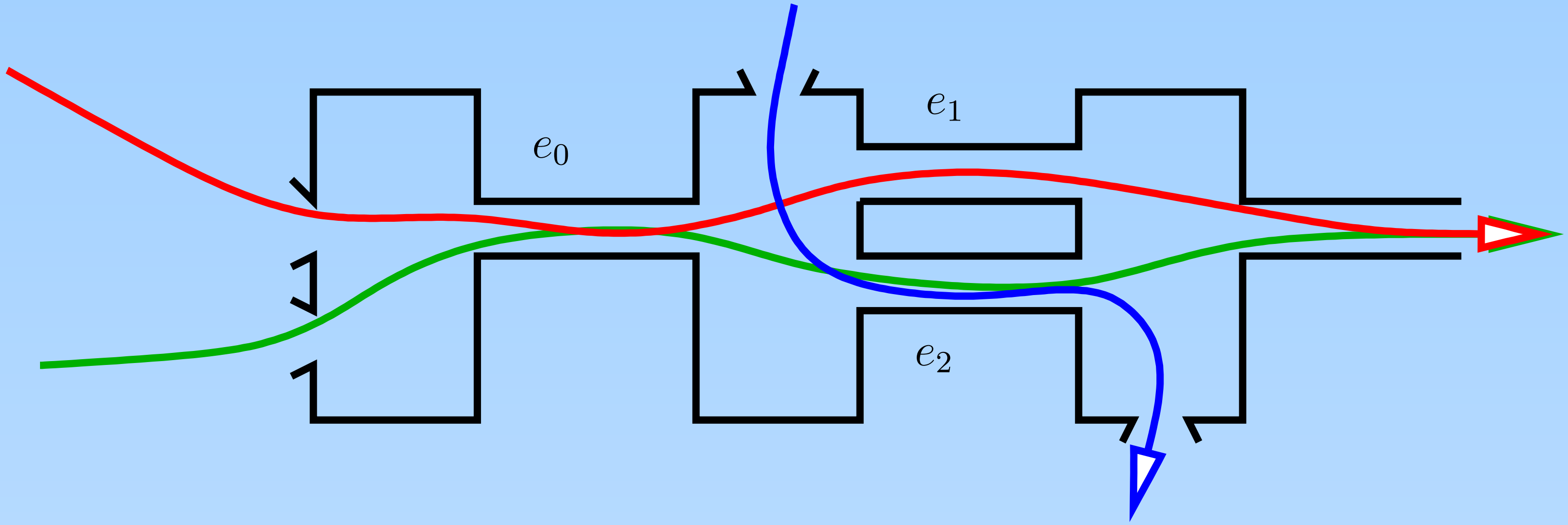
Joint edge is overloaded.

For legal insertions no jam is created.

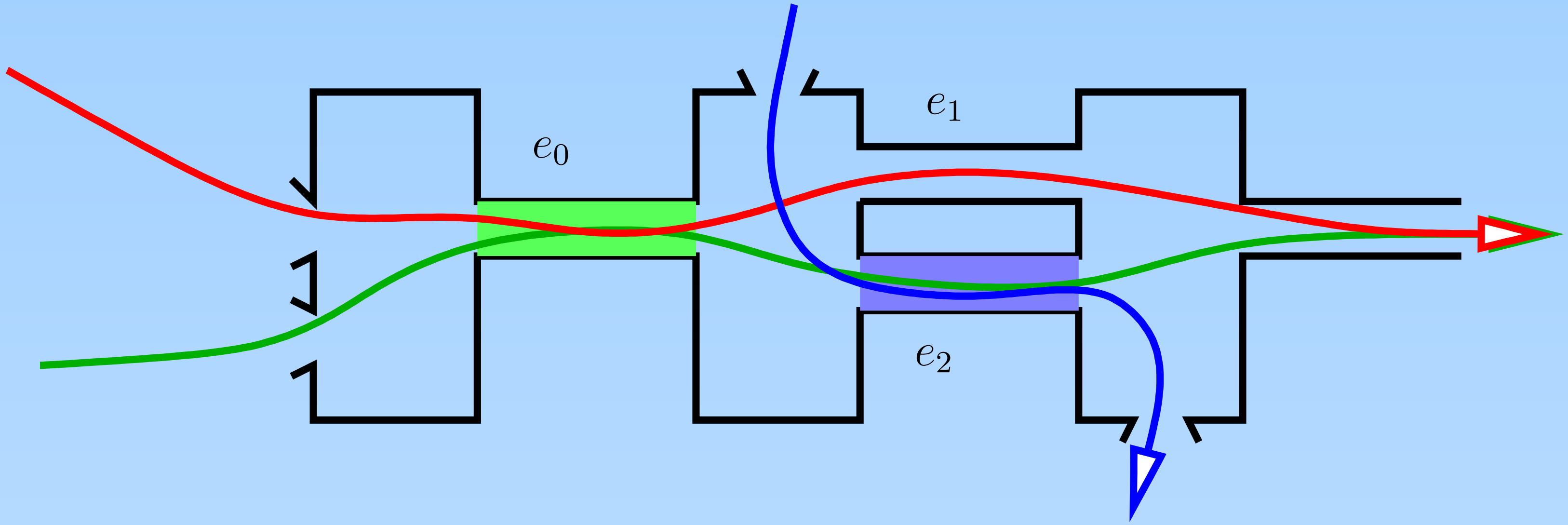
Piling up packets



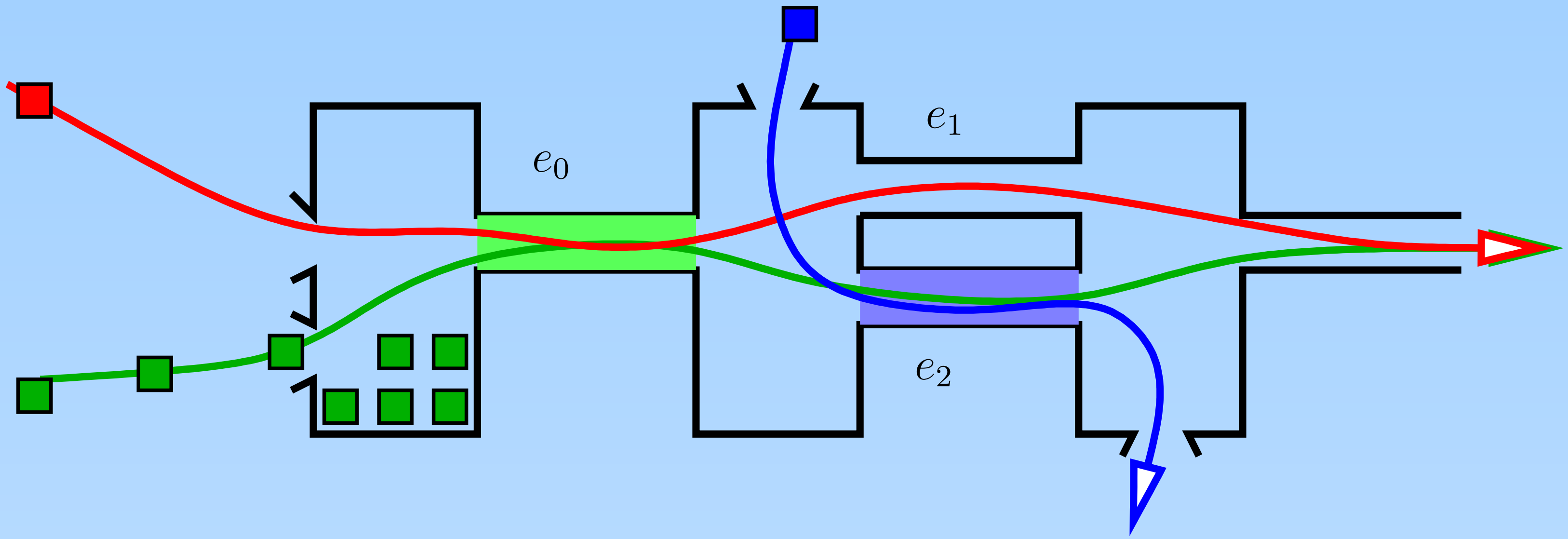
Piling up packets



Piling up packets

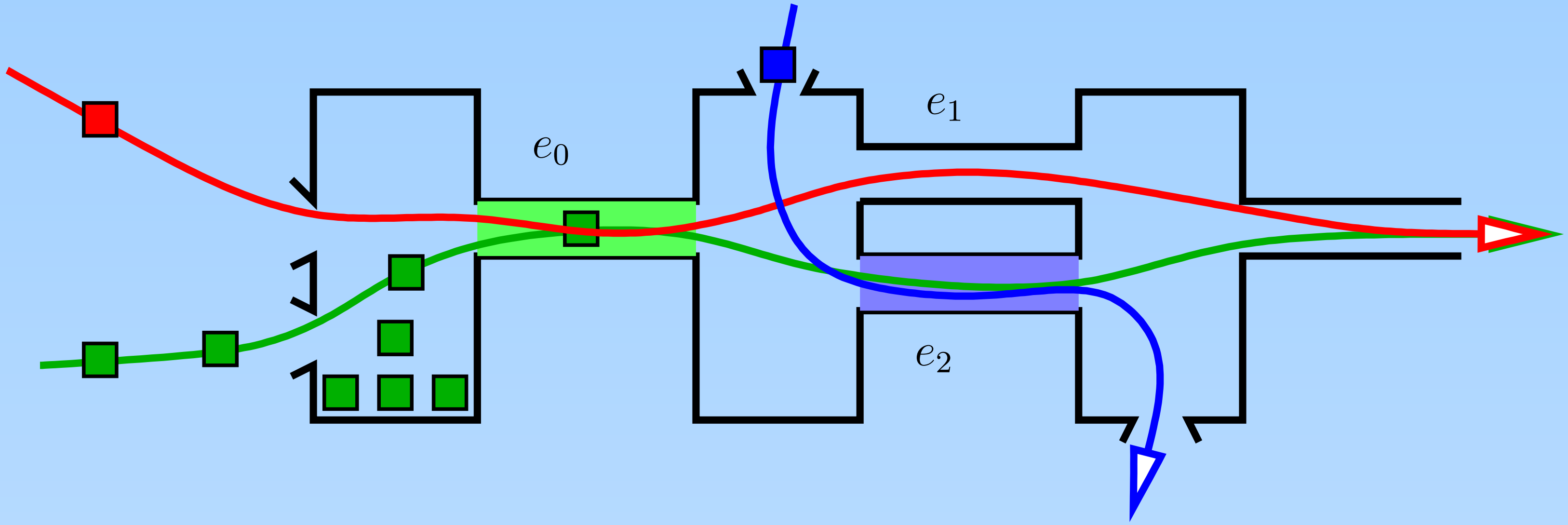


Piling up packets



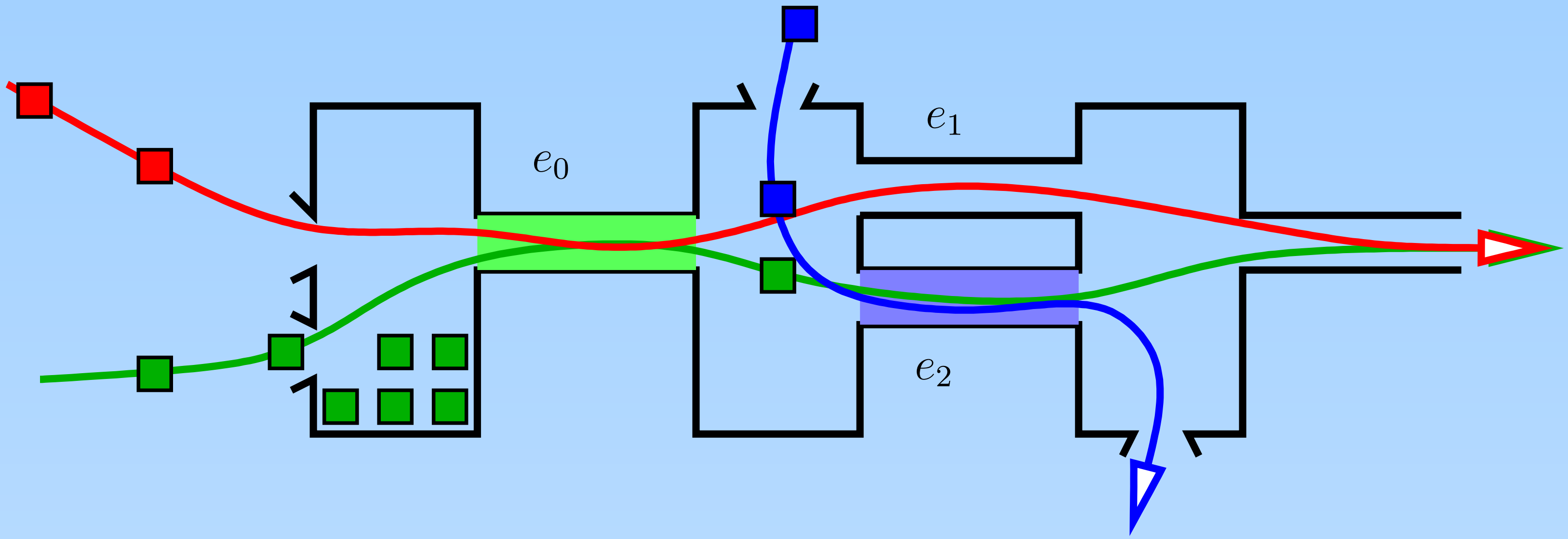
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



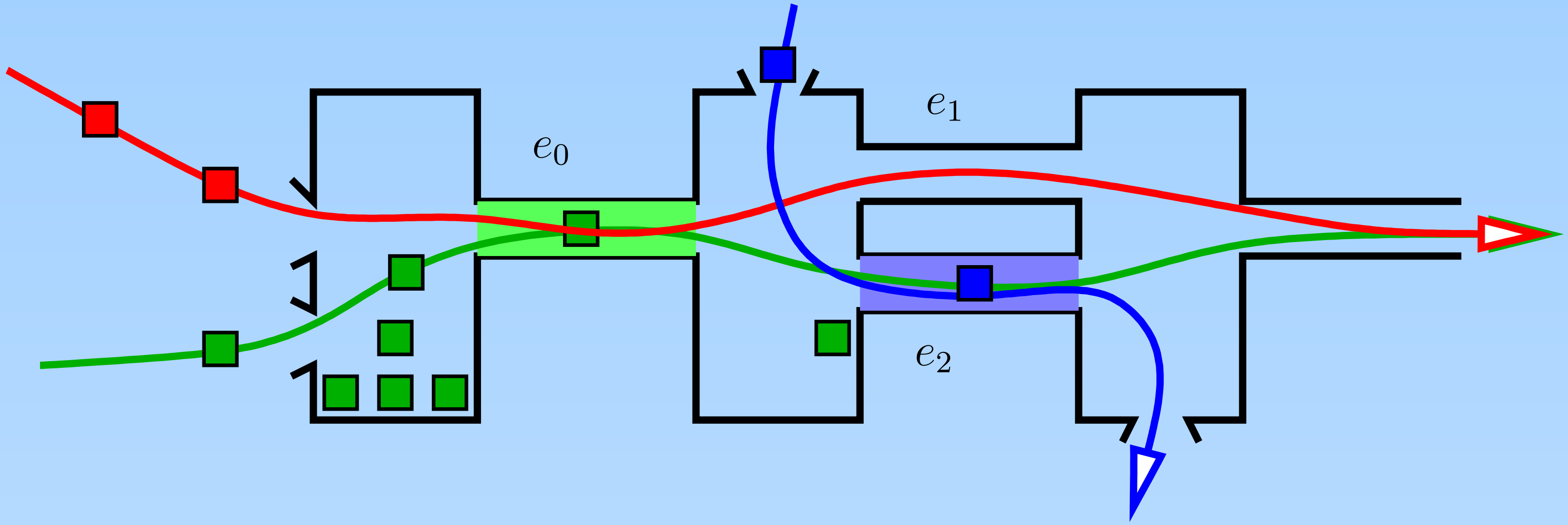
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



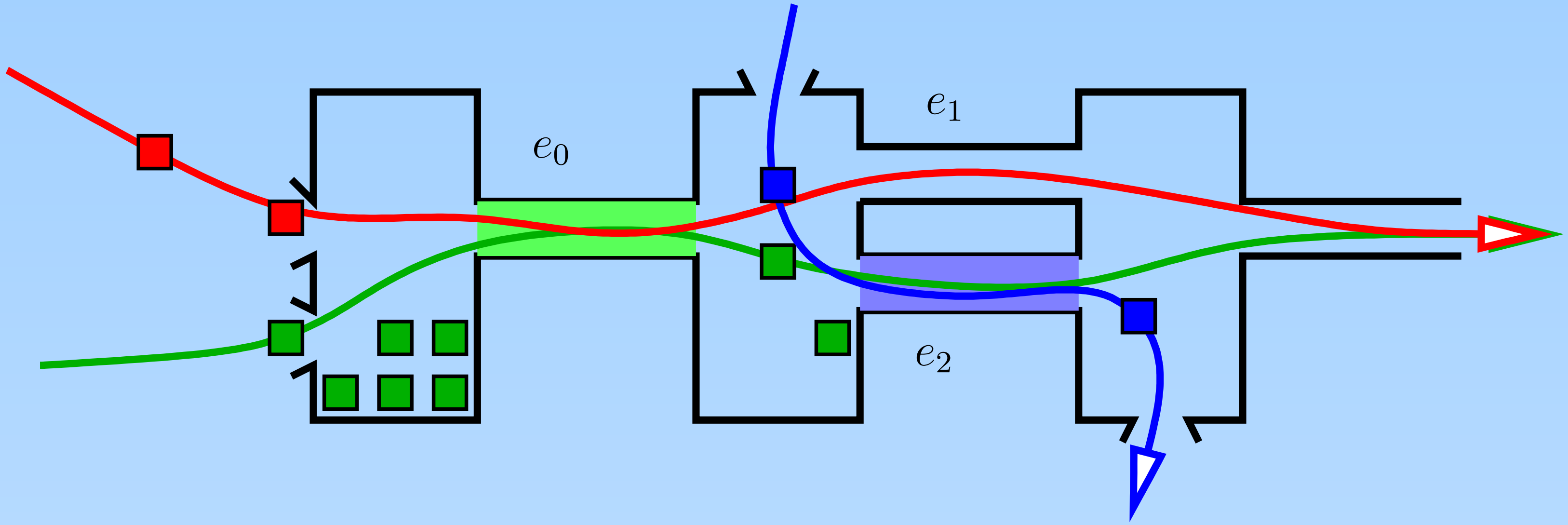
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



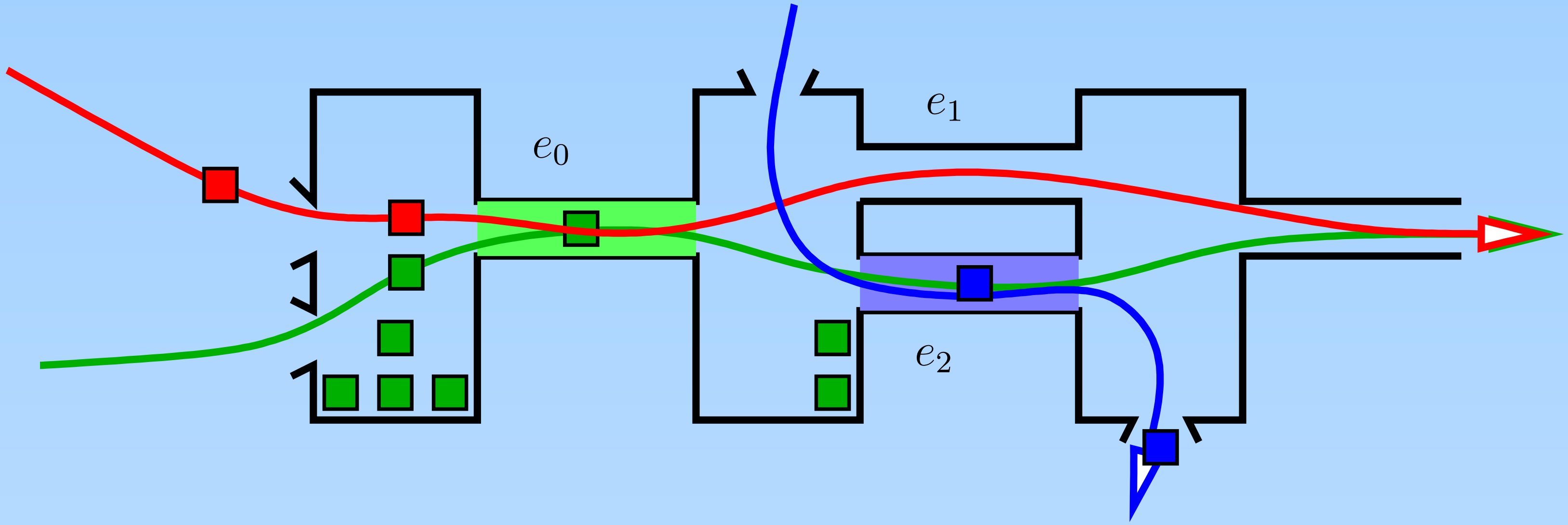
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



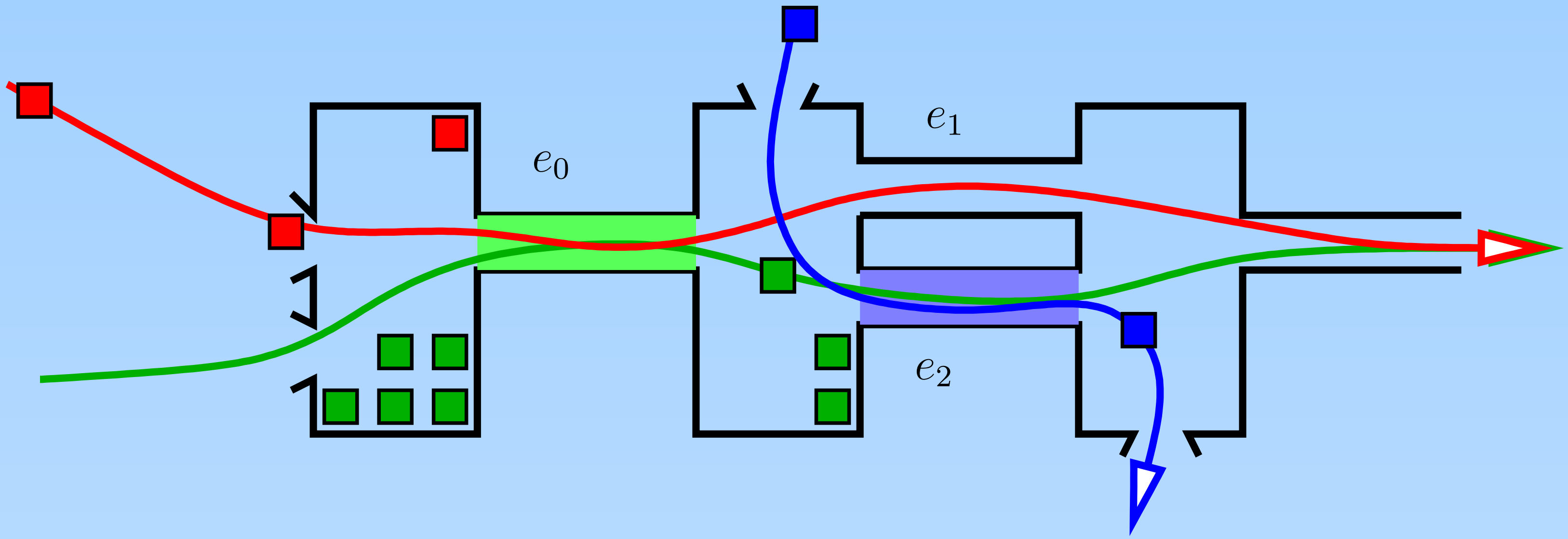
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



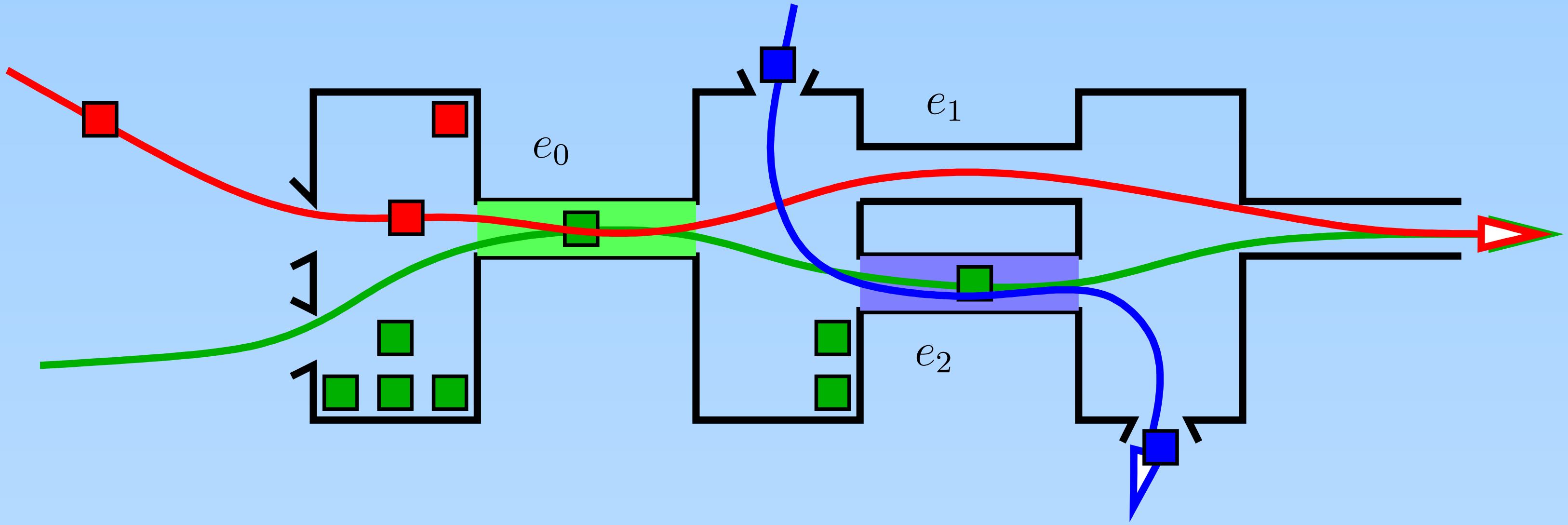
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



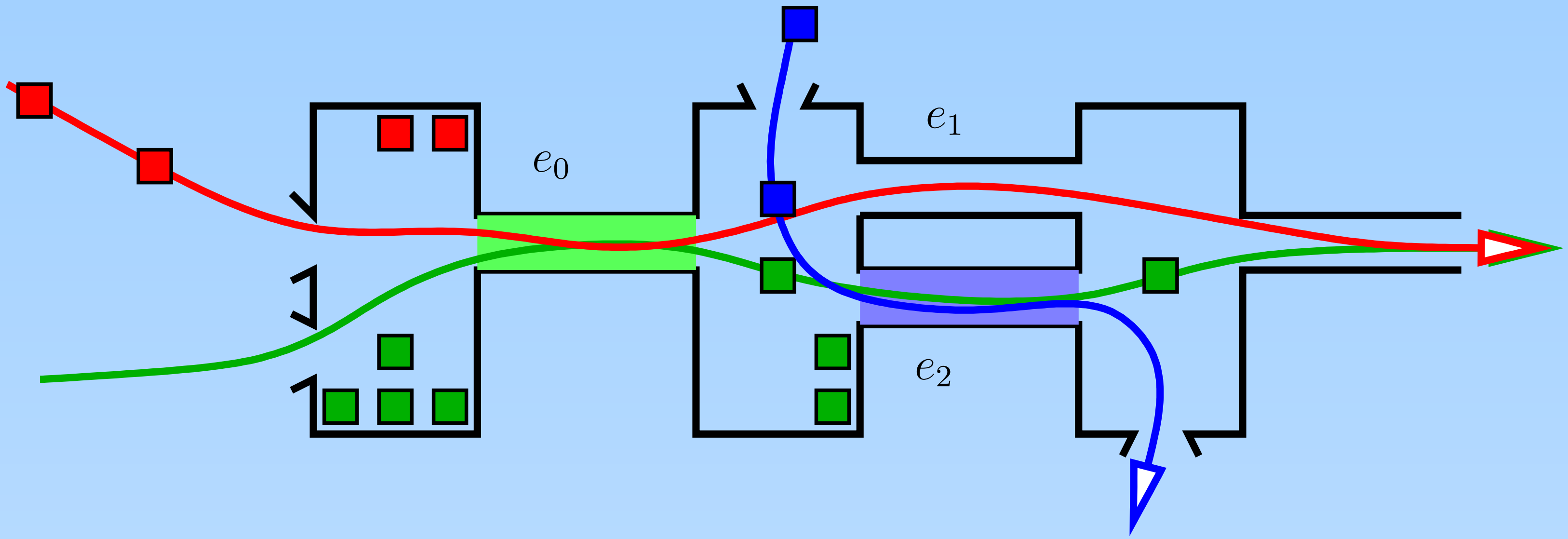
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



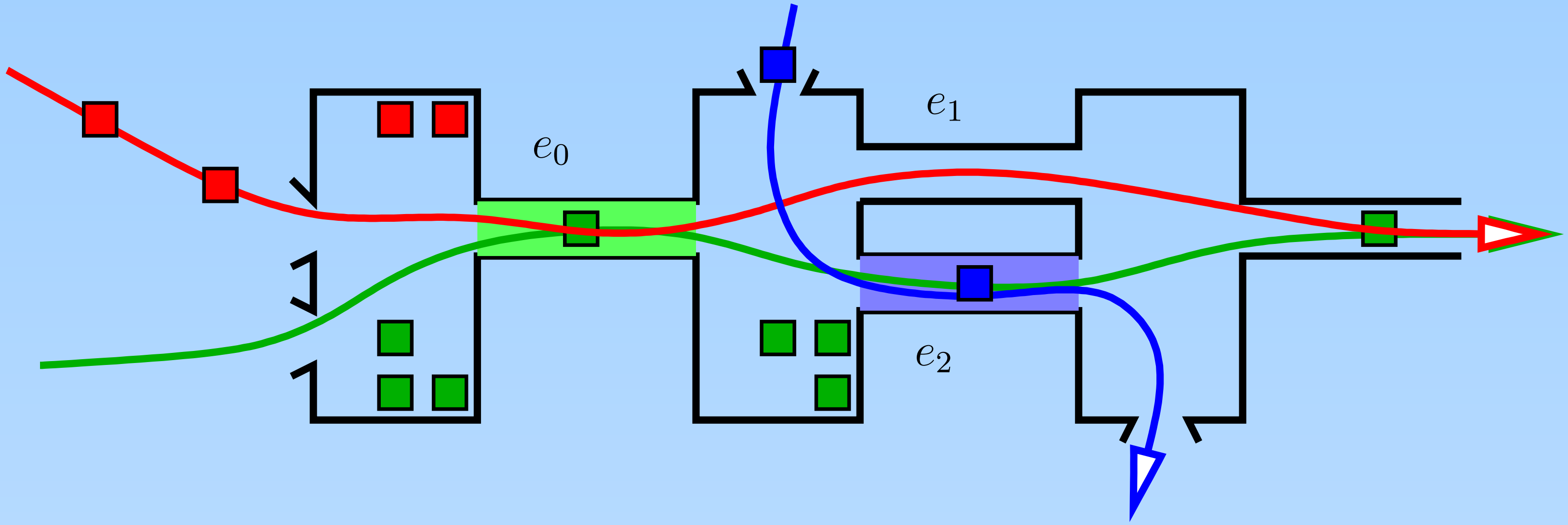
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



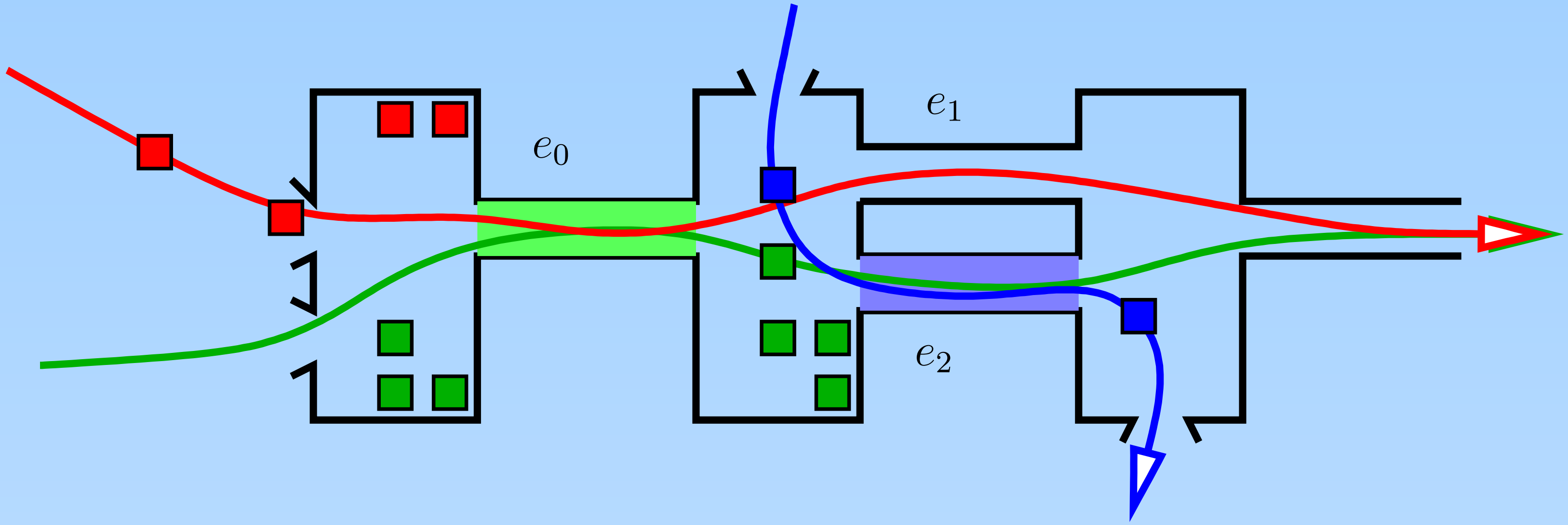
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



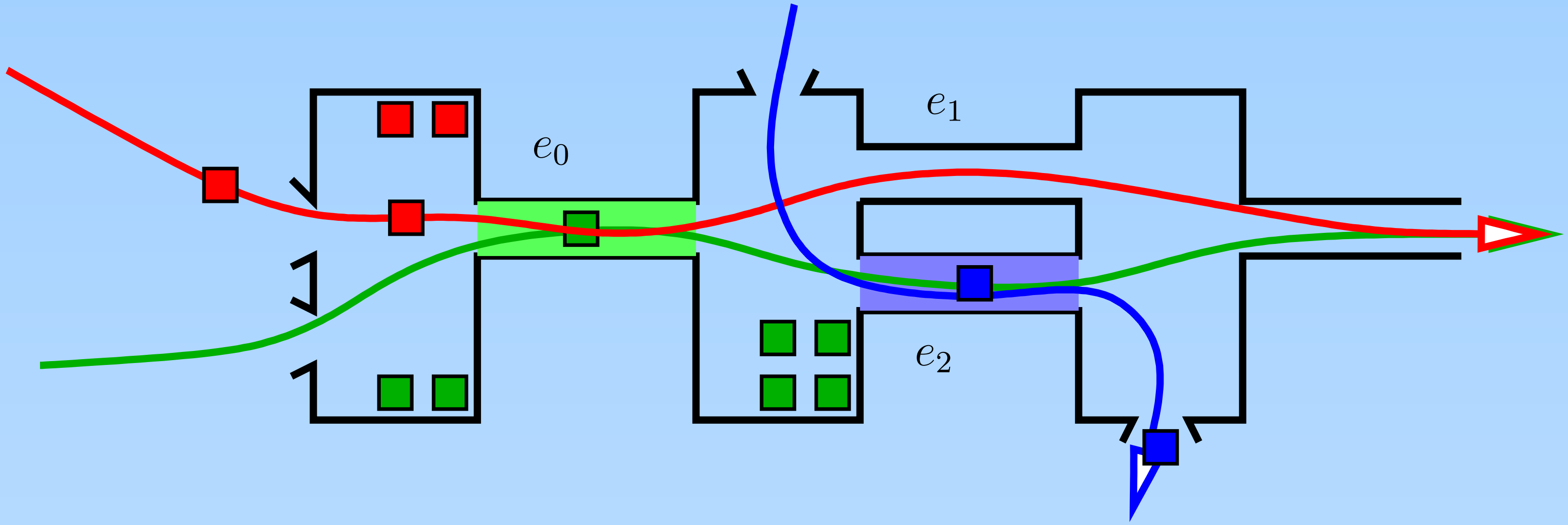
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



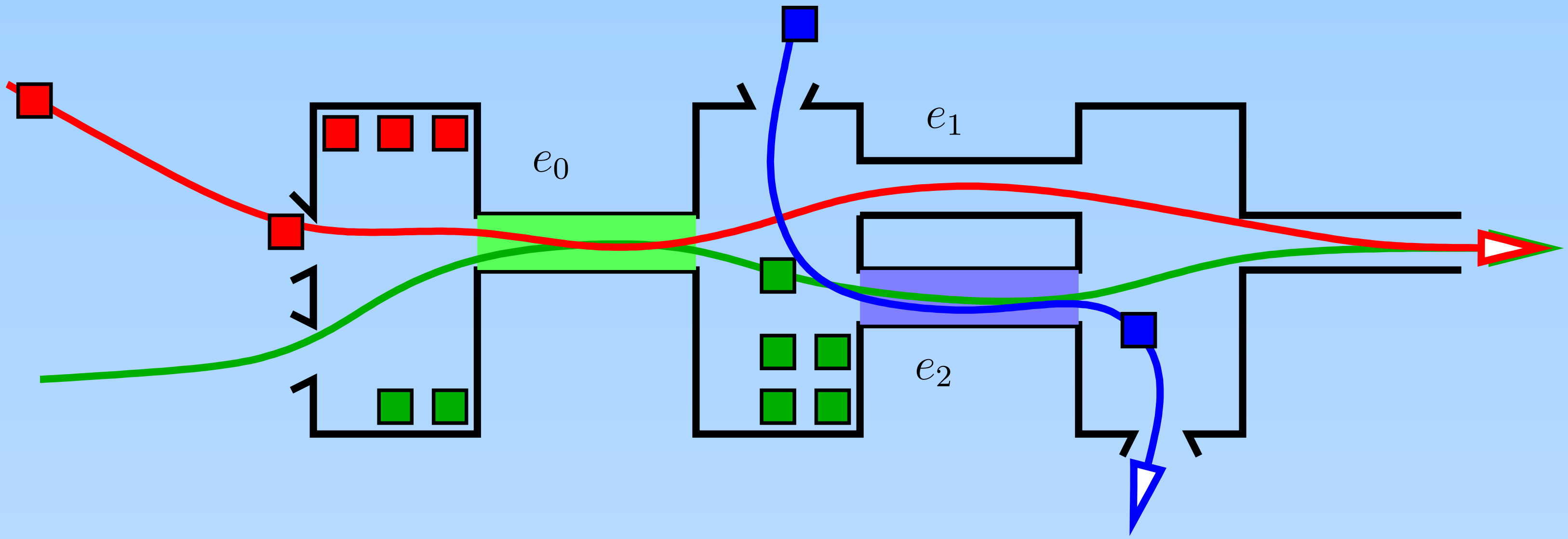
# Packets	
# green	s
# red	rs
# blue	rs

Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

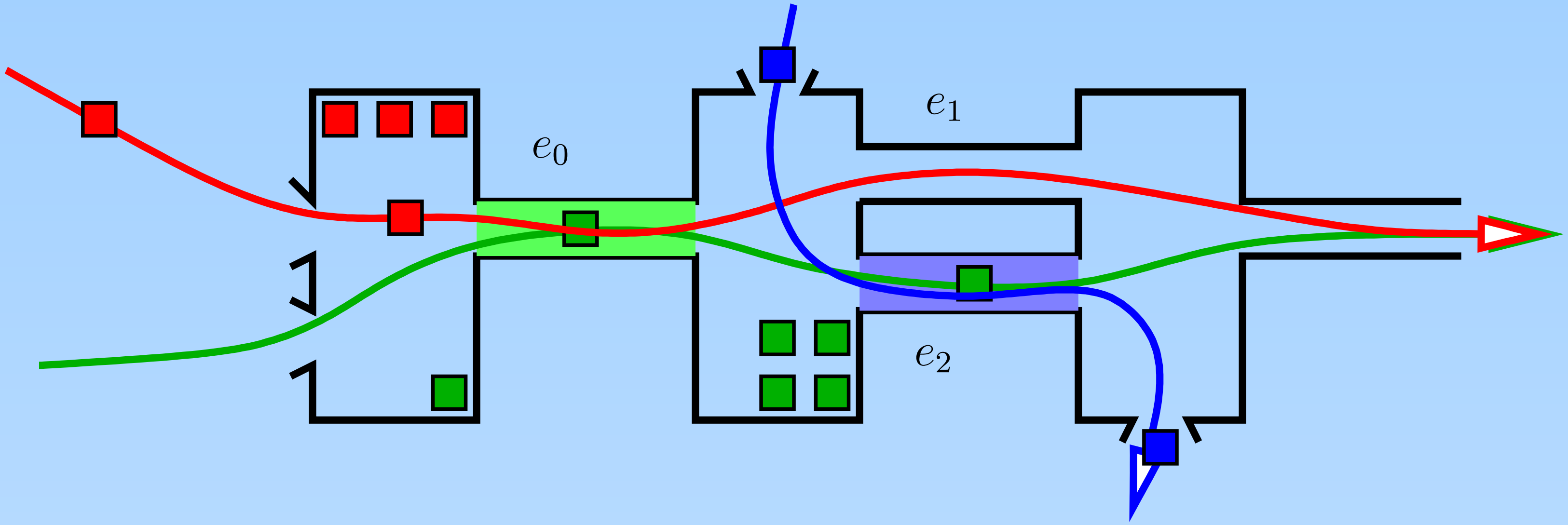
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.

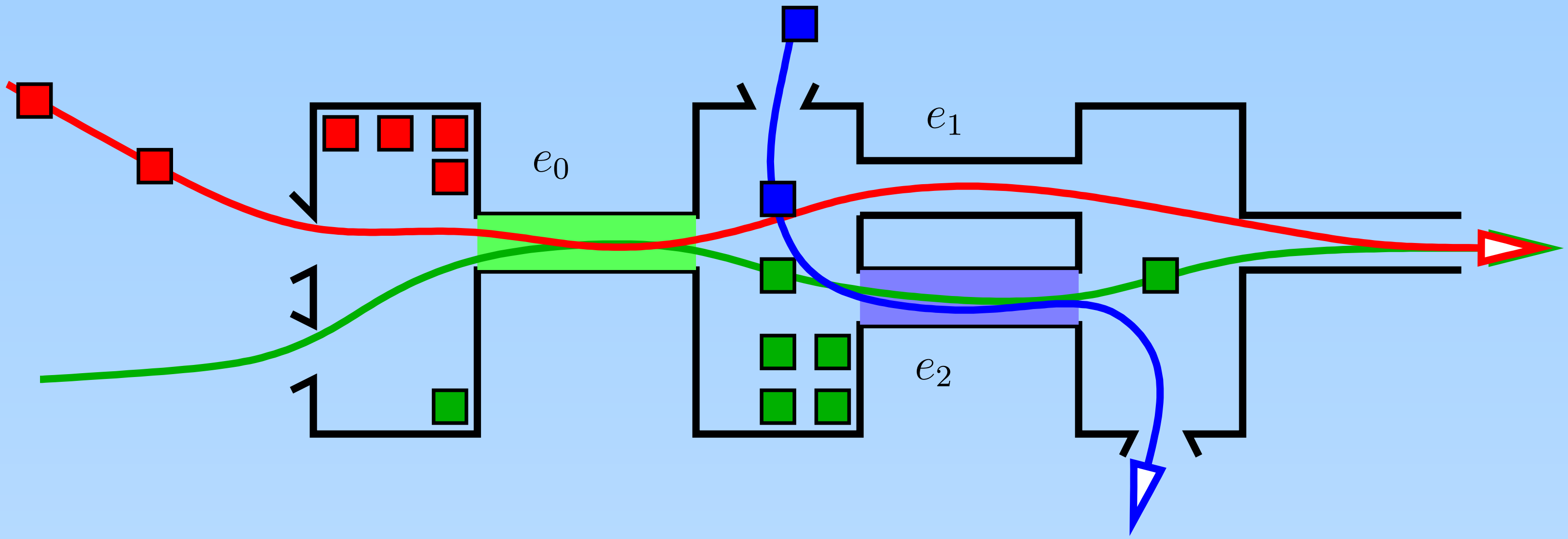
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1 - r) \cdot s$ green packets slip through.

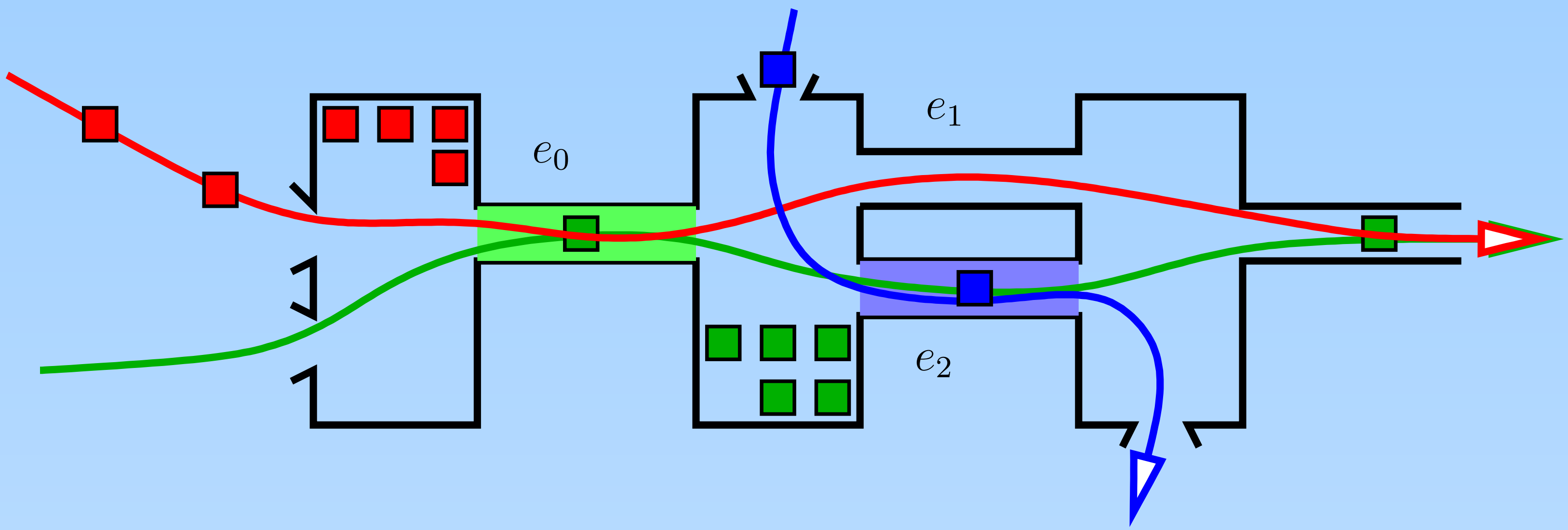
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r) \cdot s$ green packets slip through.

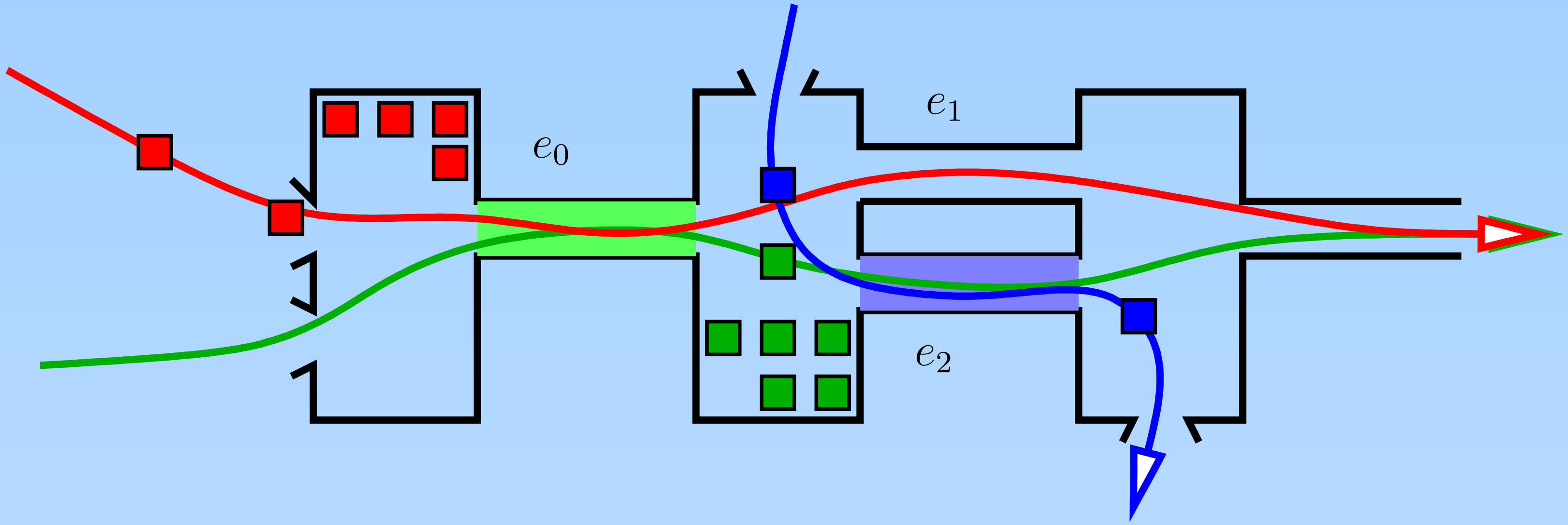
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1 - r) \cdot s$ green packets slip through.

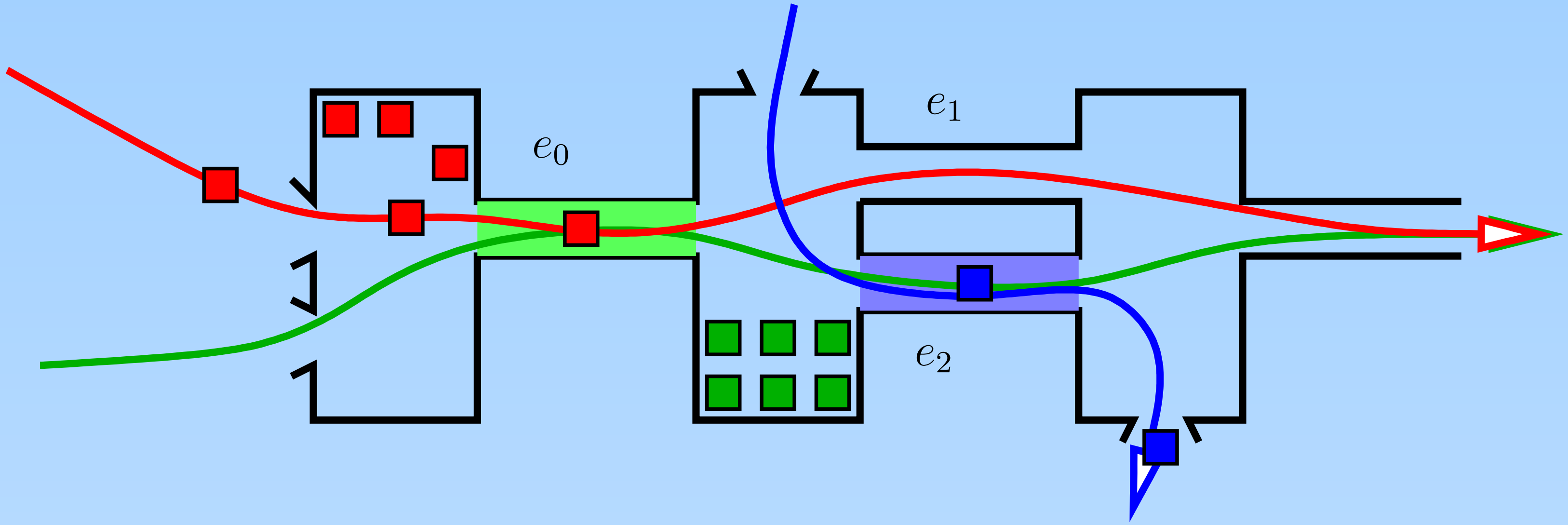
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r) \cdot s$ green packets slip through.

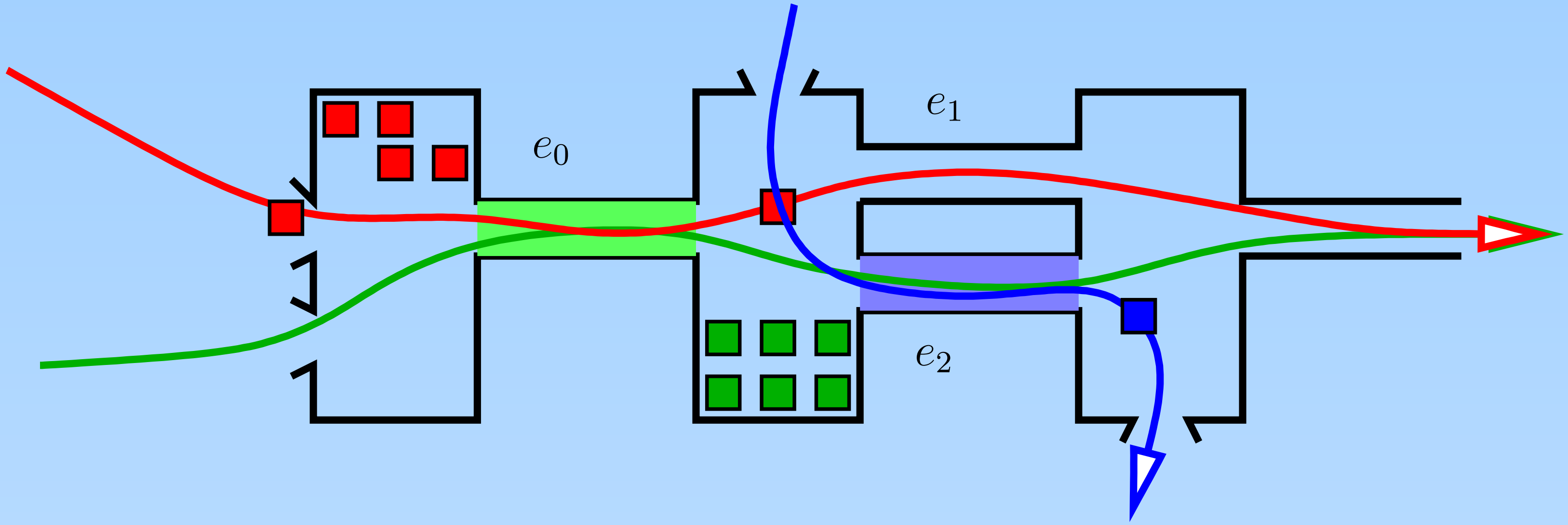
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r) \cdot s$ green packets slip through.

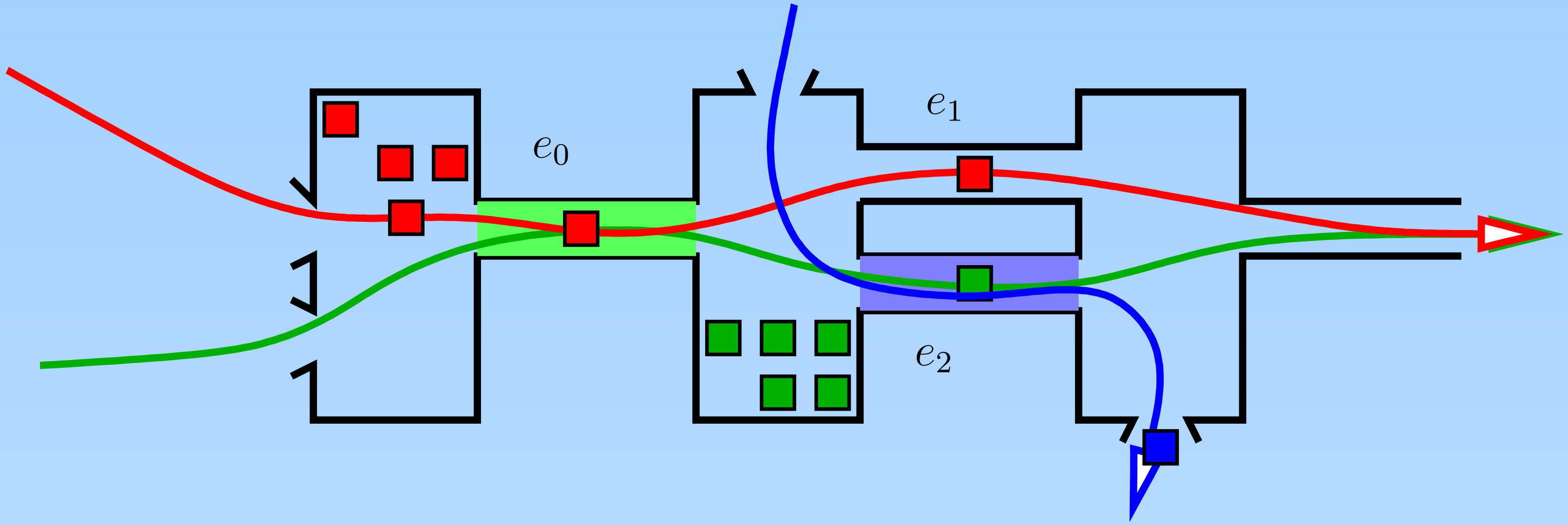
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r)\cdot s$ green packets slip through.

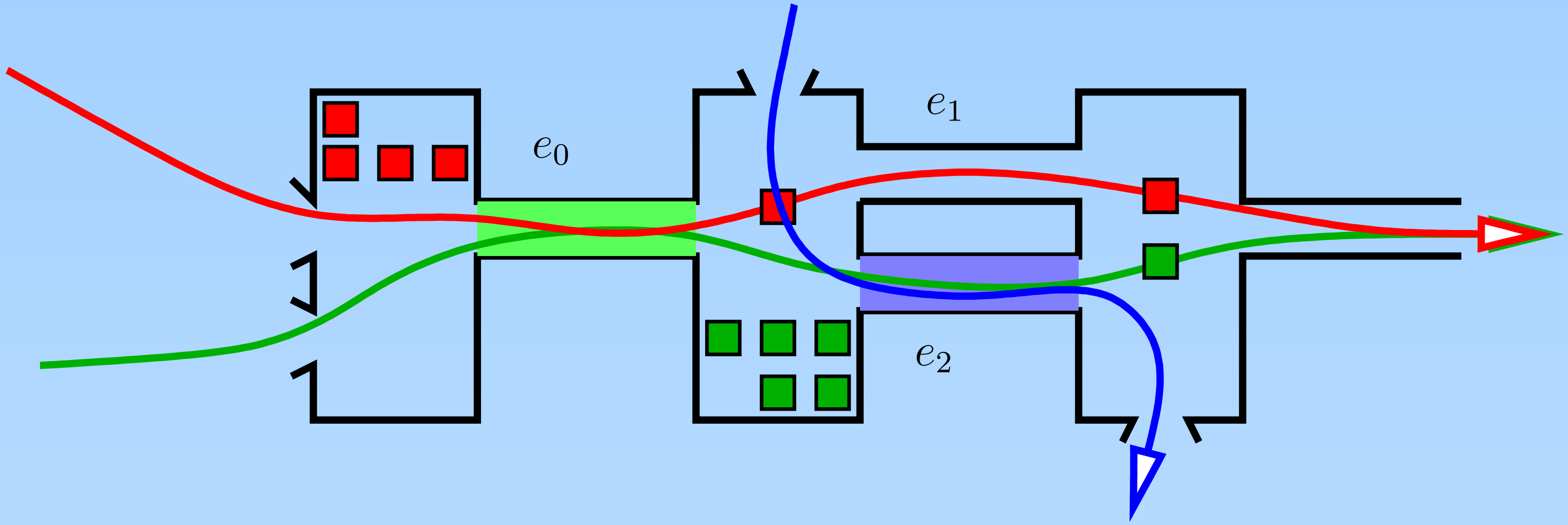
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1 - r) \cdot s$ green packets slip through.

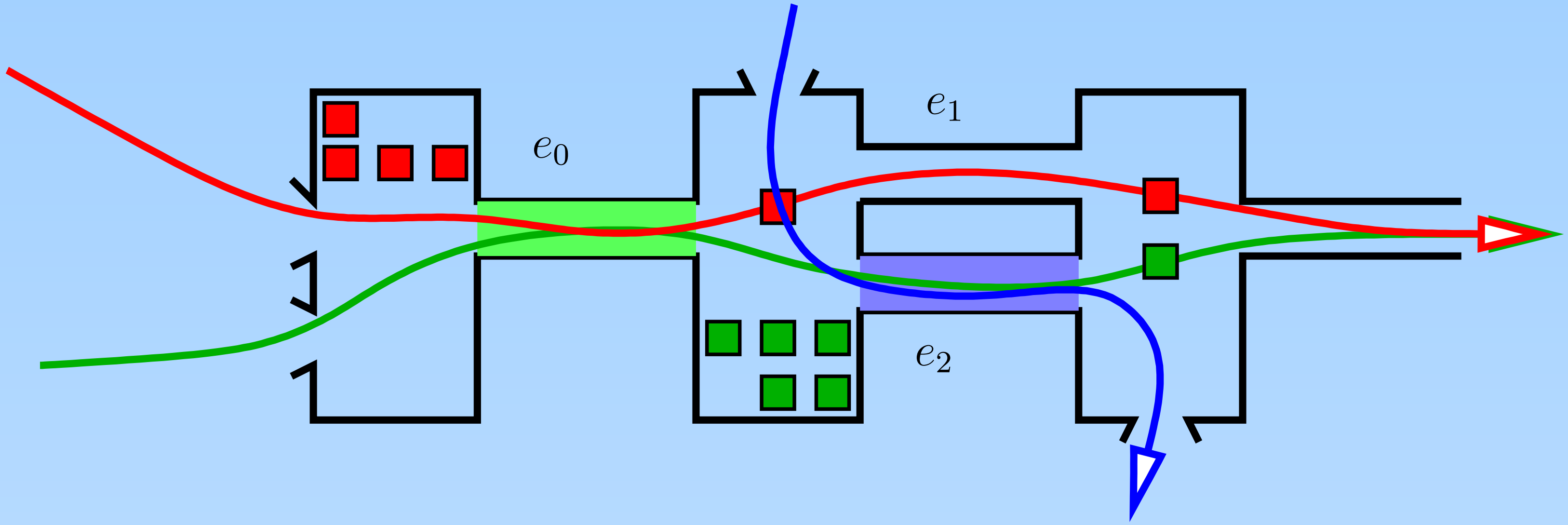
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r)\cdot s$ green packets slip through.
Remaining packets: $2r \cdot s$.

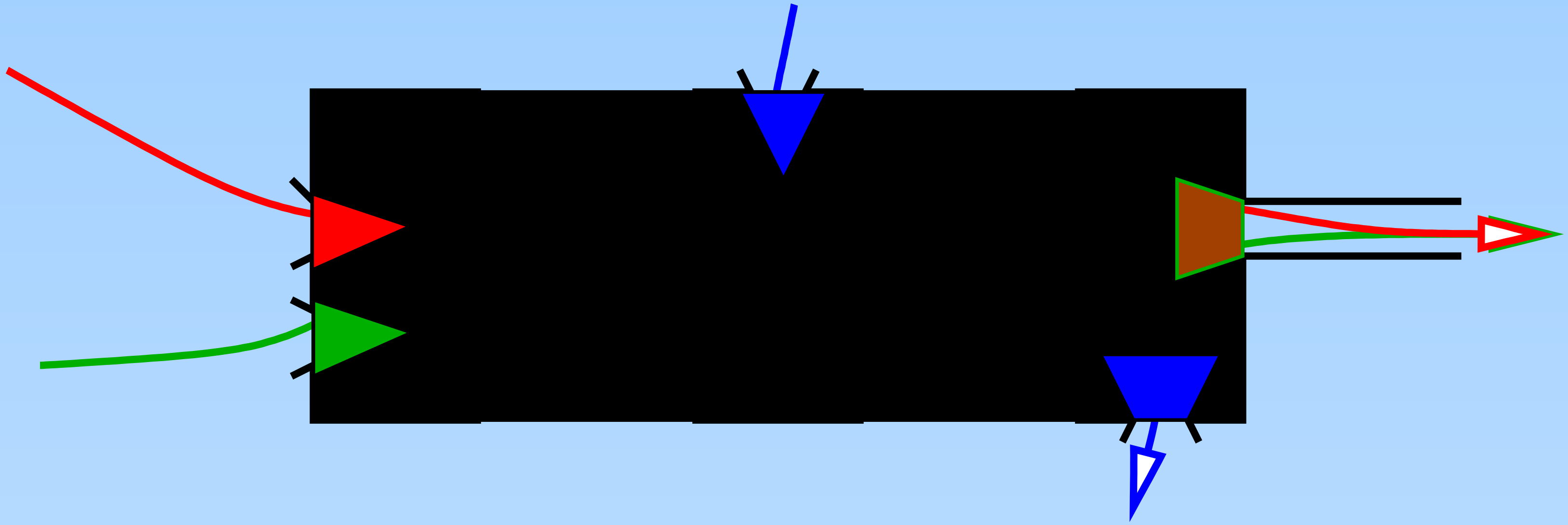
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r)\cdot s$ green packets slip through.
Remaining packets: $2r\cdot s$.
For $r > \frac{1}{2}$ the set of packets grows by a multiplicative factor.

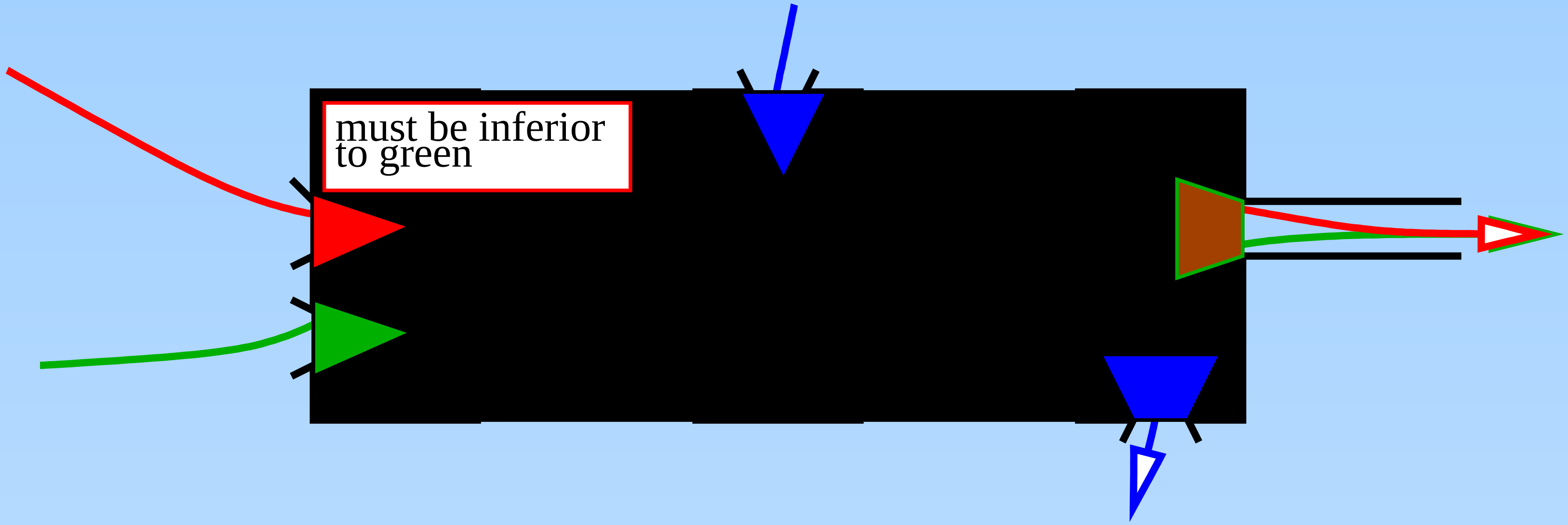
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r)\cdot s$ green packets slip through.
Remaining packets: $2r \cdot s$.
For $r > \frac{1}{2}$ the set of packets grows by a multiplicative factor.

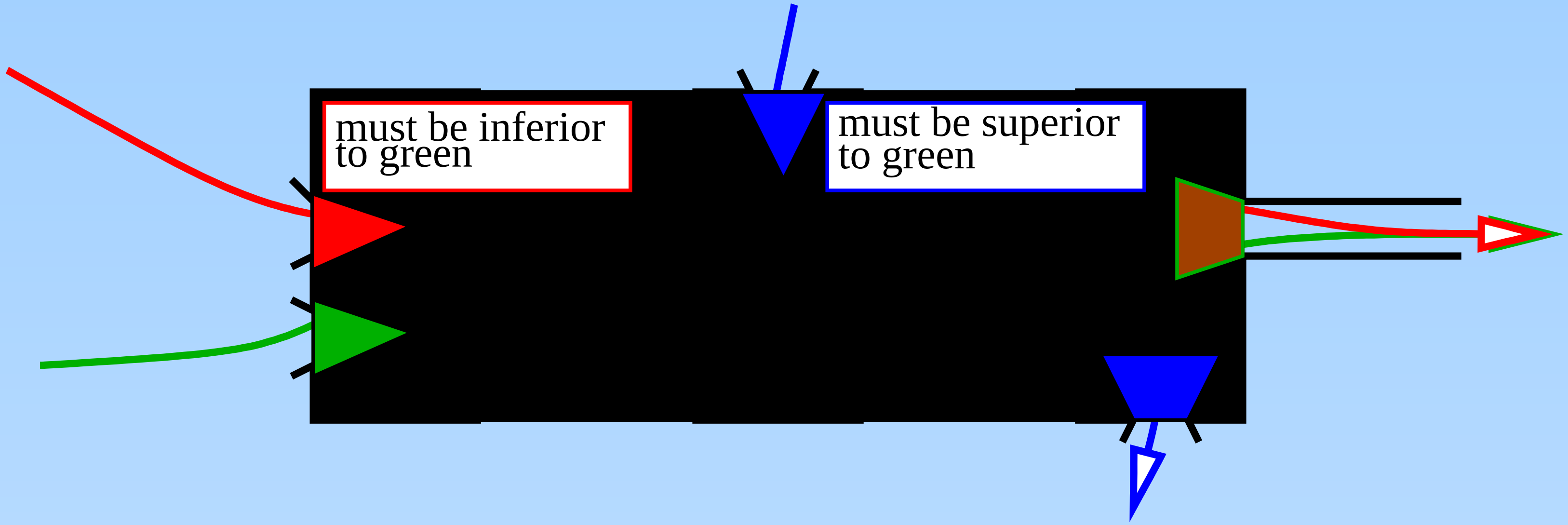
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r) \cdot s$ green packets slip through.
Remaining packets: $2r \cdot s$.
For $r > \frac{1}{2}$ the set of packets grows by a multiplicative factor.

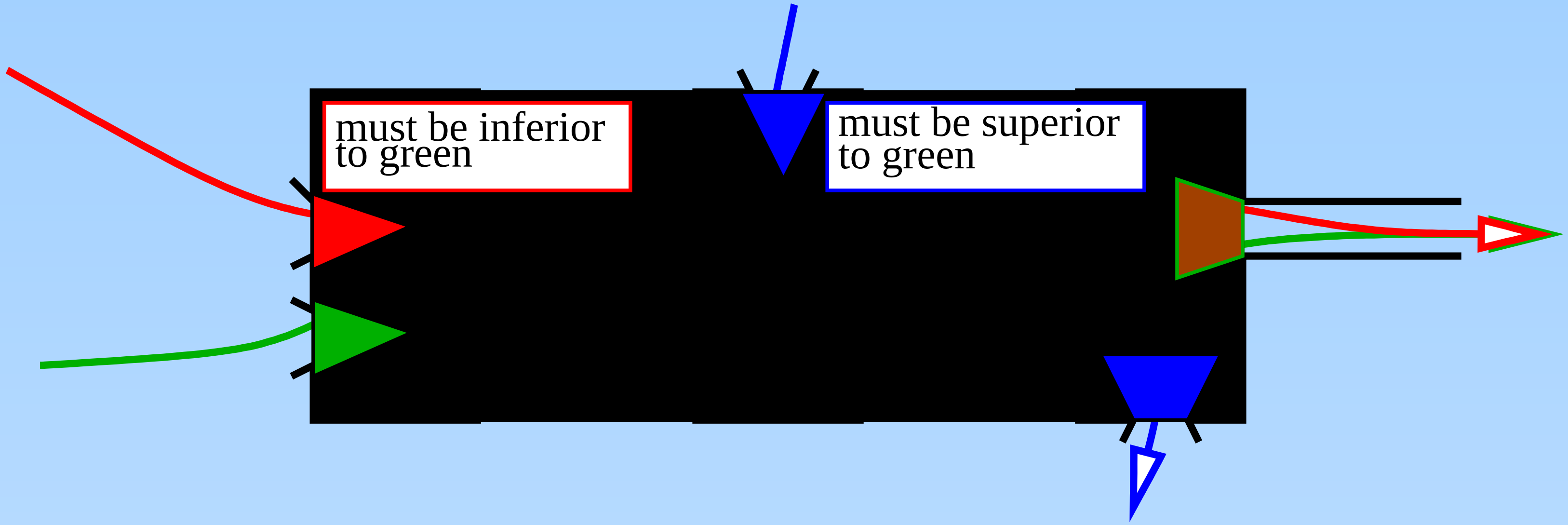
Piling up packets



# Packets	
# green	s
# red	rs
# blue	rs

No red packet traverses the system.
Only $(1-r)\cdot s$ green packets slip through.
Remaining packets: $2r \cdot s$.
For $r > \frac{1}{2}$ the set of packets grows by a multiplicative factor.

Piling up packets

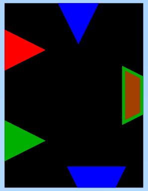


# Packets	
# green	s
# red	rs
# blue	rs

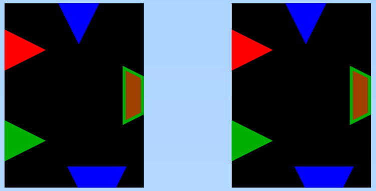
No red packet traverses the system.
Only $(1-r)\cdot s$ green packets slip through.
Remaining packets: $2r\cdot s$.
For $r > \frac{1}{2}$ the set of packets grows by a multiplicative factor.

Concatenating Gadgets

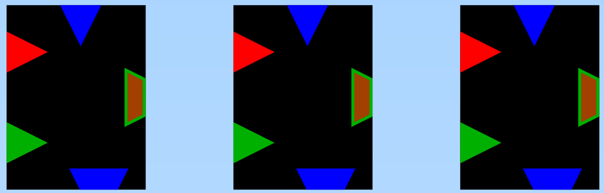
Concatenating Gadgets



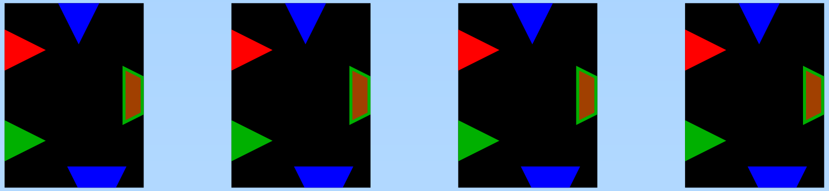
Concatenating Gadgets



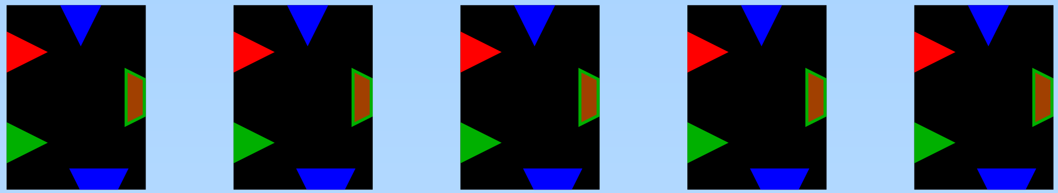
Concatenating Gadgets



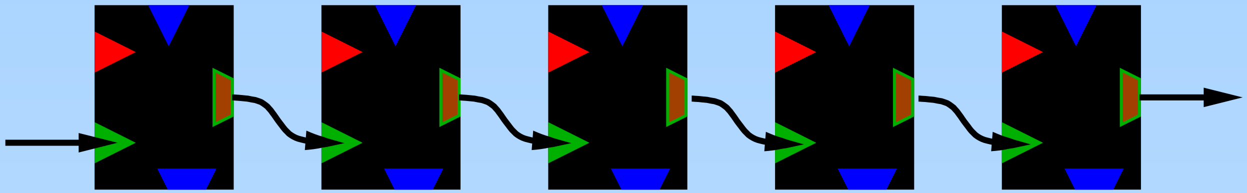
Concatenating Gadgets



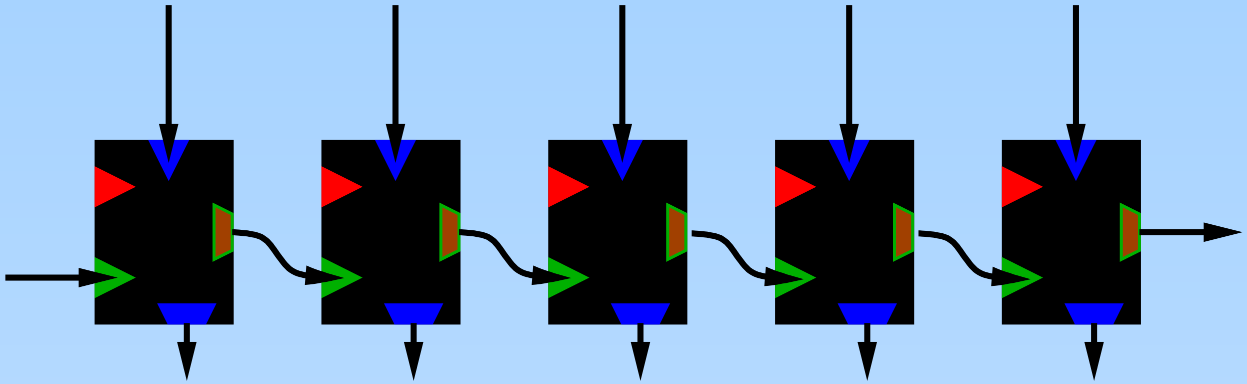
Concatenating Gadgets



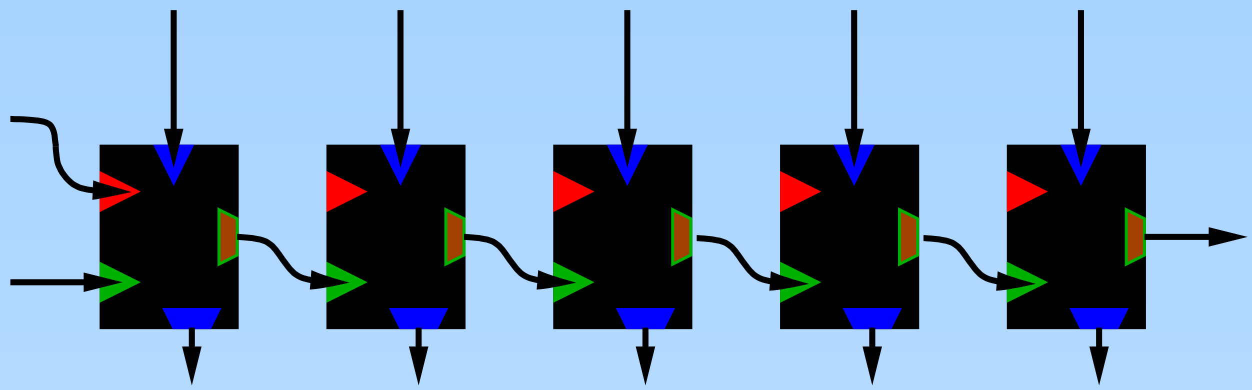
Concatenating Gadgets



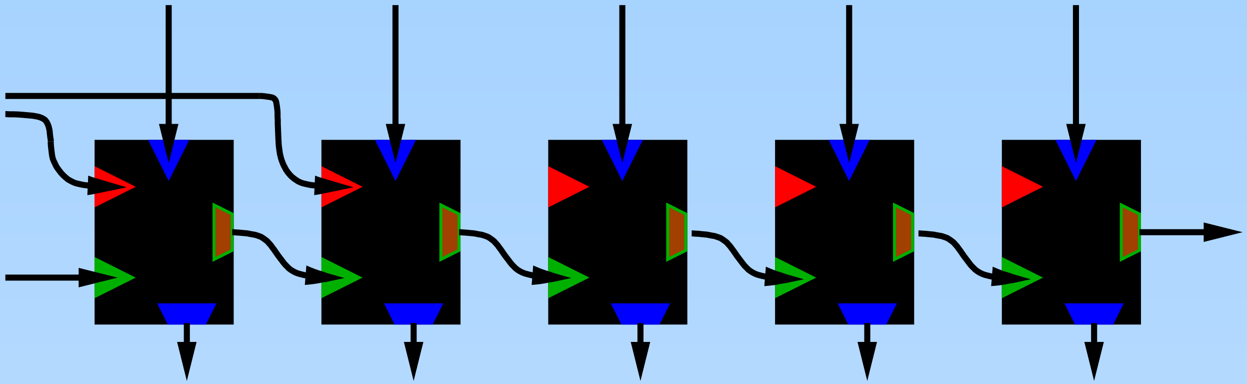
Concatenating Gadgets



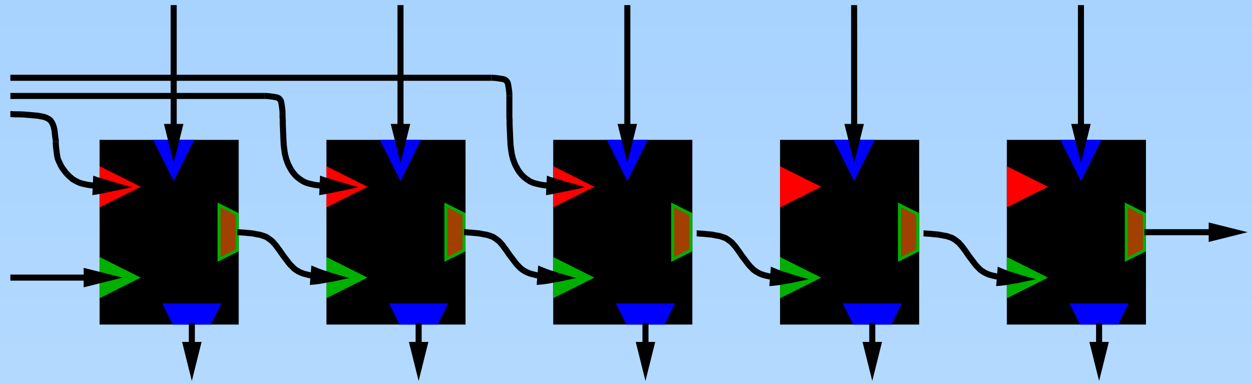
Concatenating Gadgets



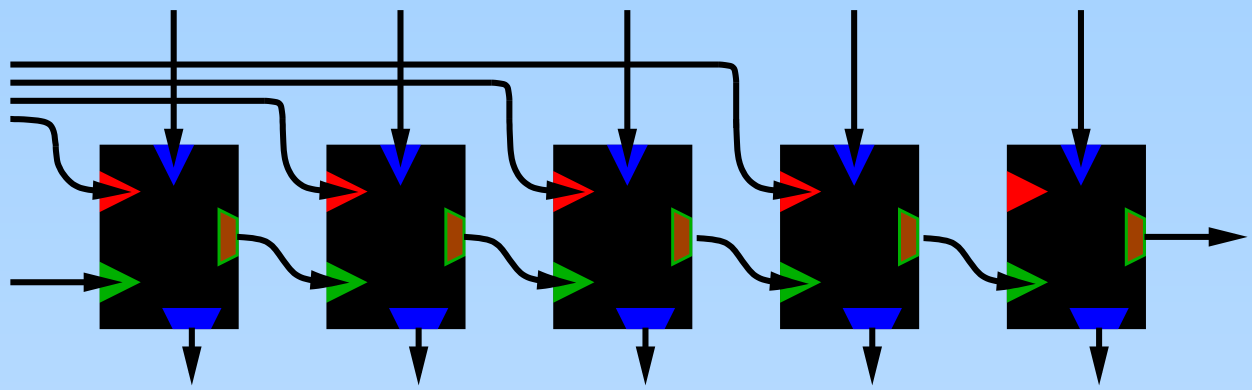
Concatenating Gadgets



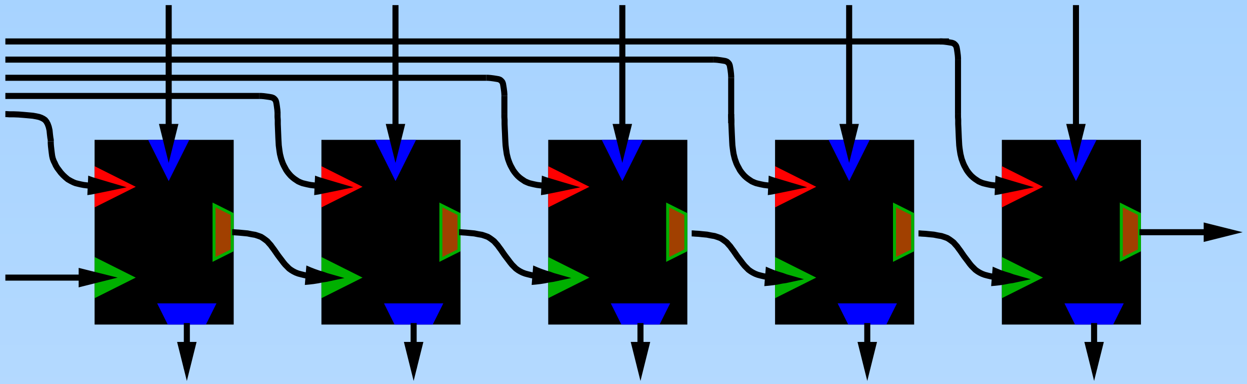
Concatenating Gadgets



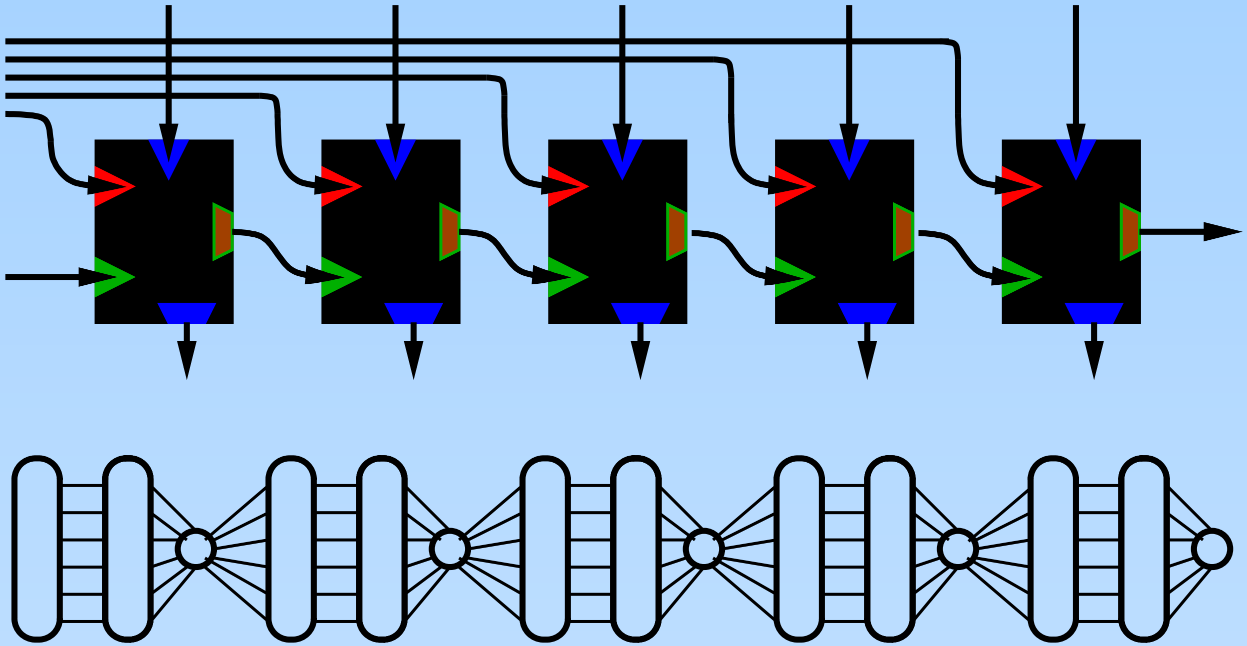
Concatenating Gadgets



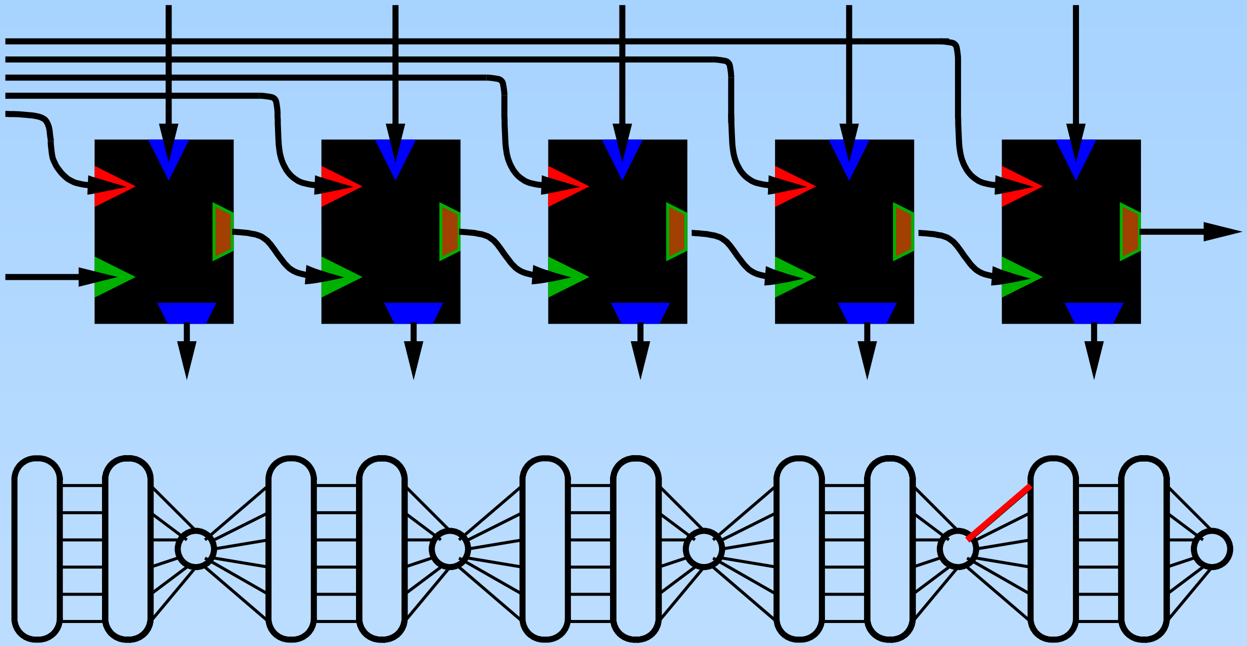
Concatenating Gadgets



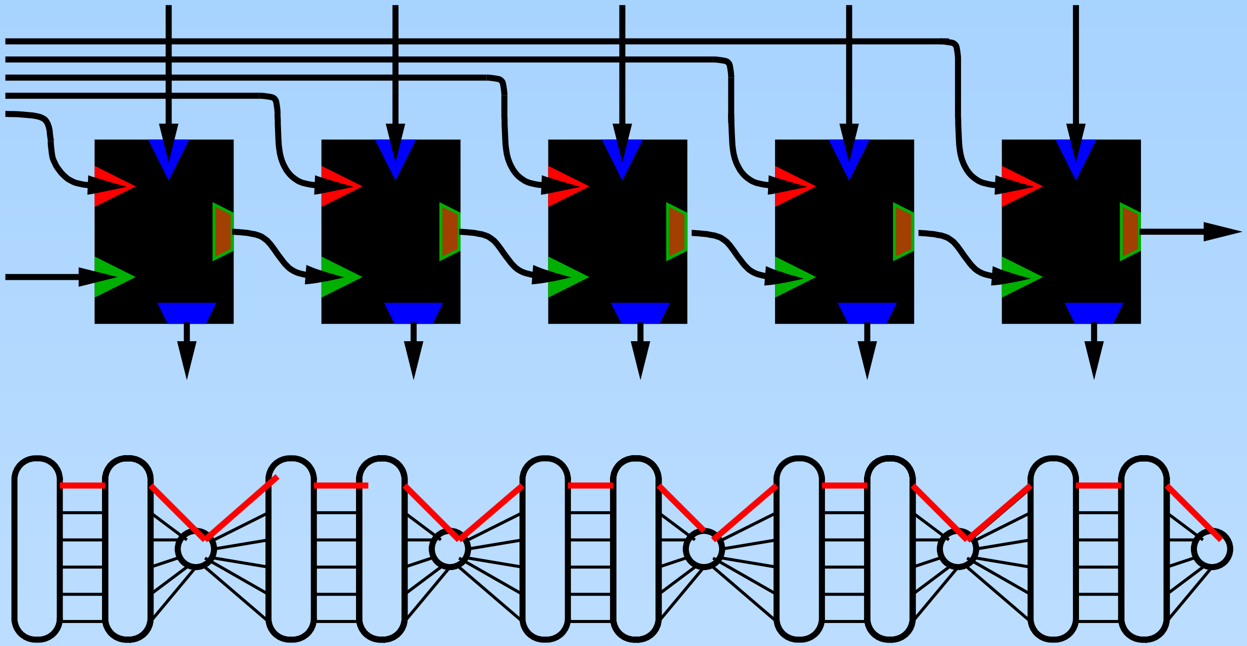
Concatenating Gadgets



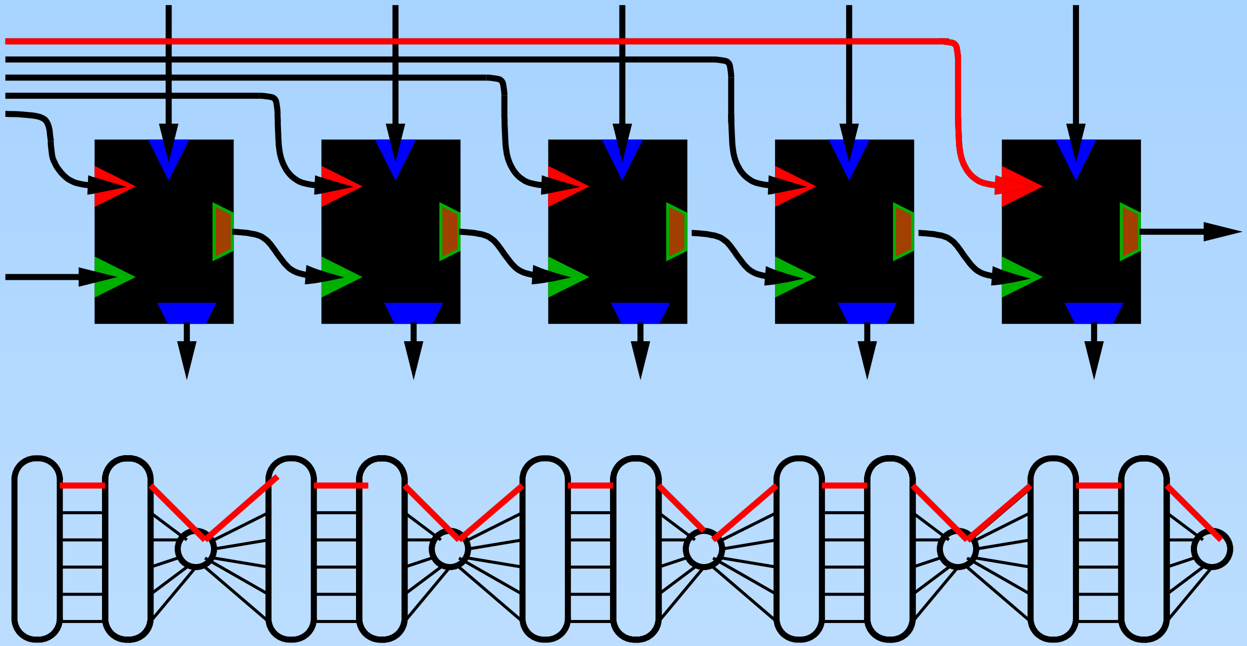
Concatenating Gadgets



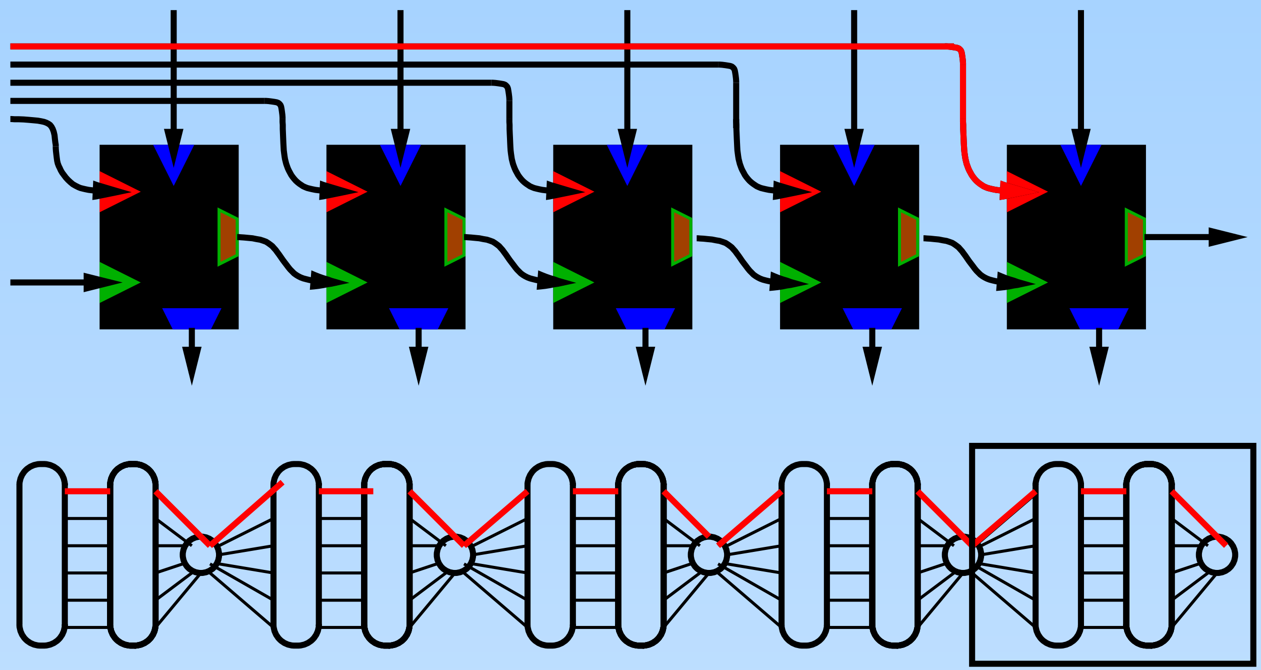
Concatenating Gadgets



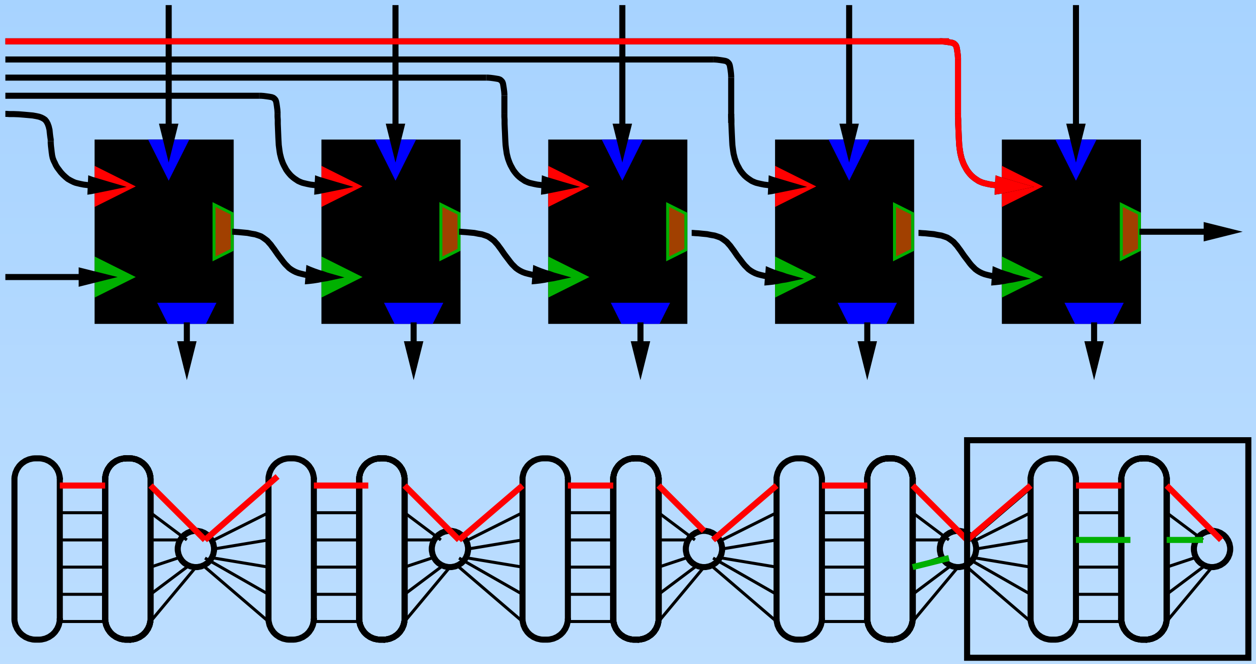
Concatenating Gadgets



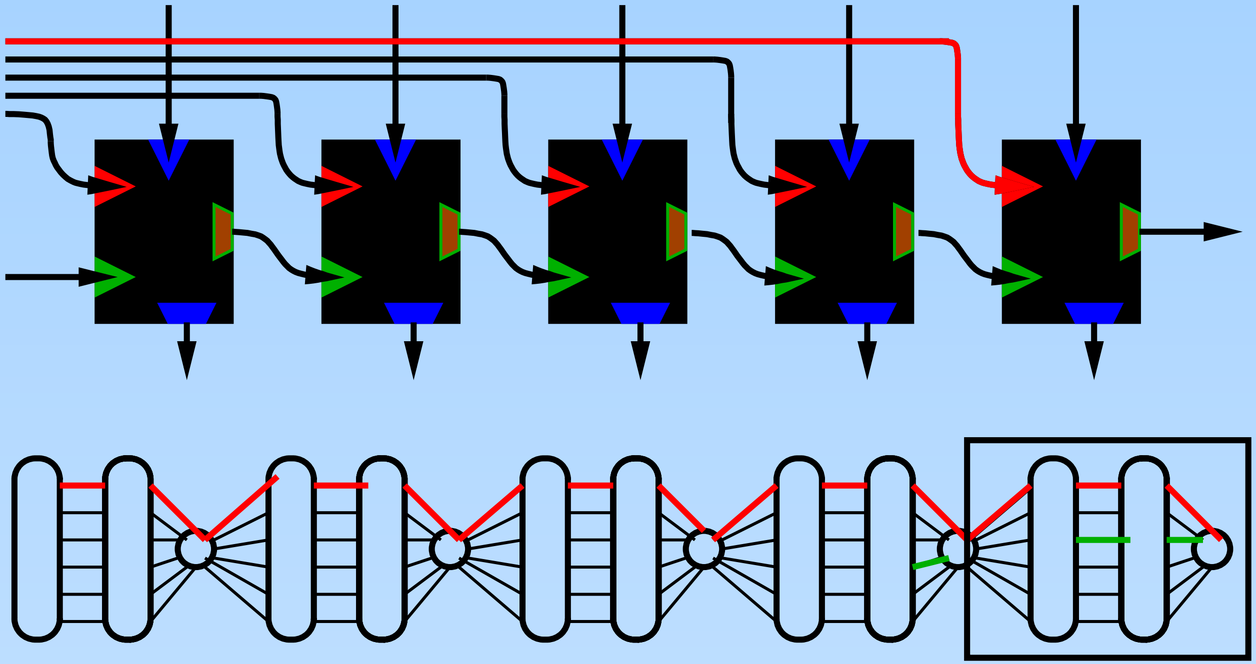
Concatenating Gadgets



Concatenating Gadgets



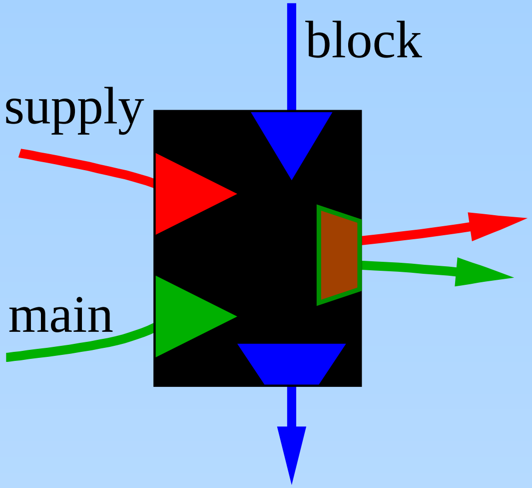
Concatenating Gadgets



How can we get (blue) paths capable of blocking?

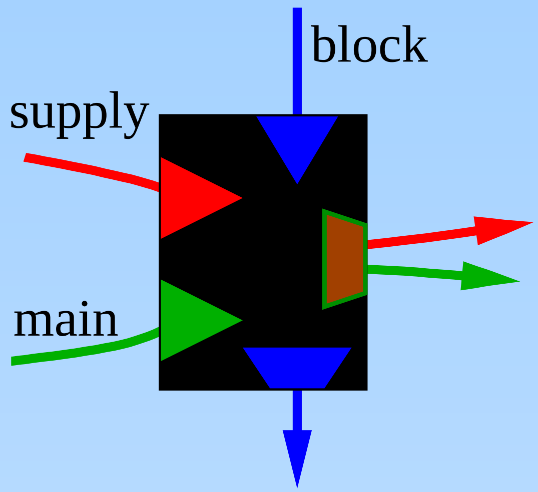
Idea: Improve Gadget

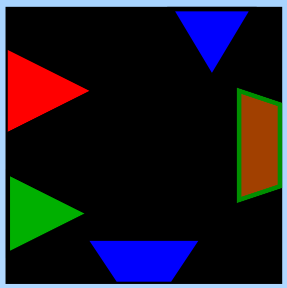
Idea: Improve Gadget



If green packets cannot be blocked by blue packets...

Idea: Improve Gadget

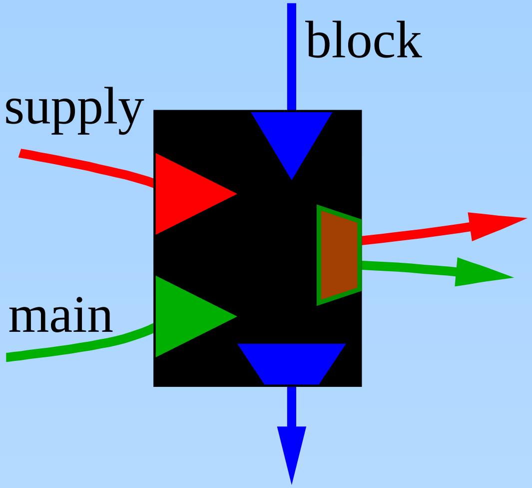


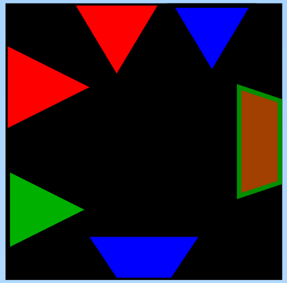


If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

Idea: Improve Gadget

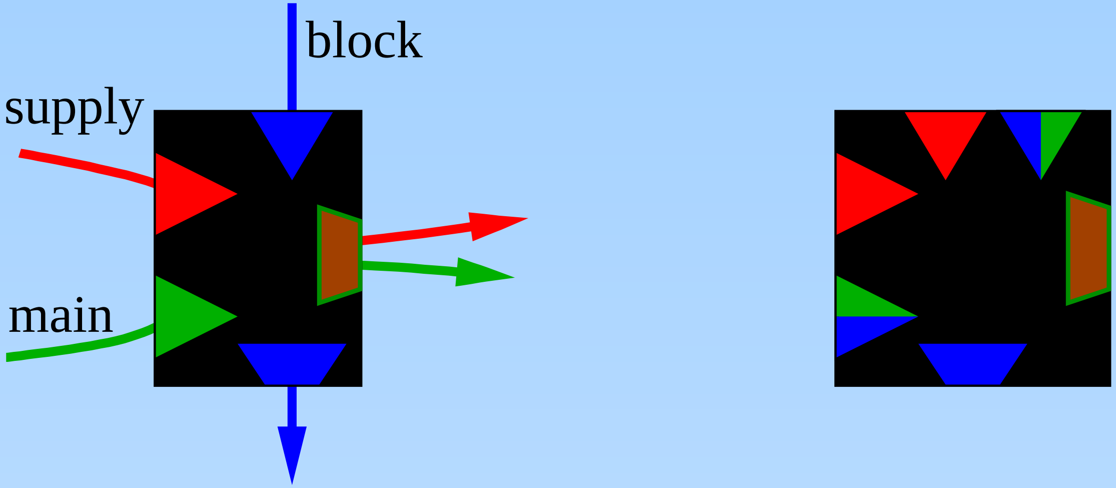




If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

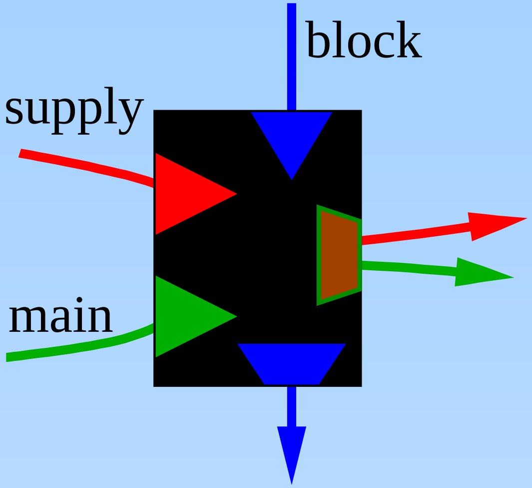
Idea: Improve Gadget

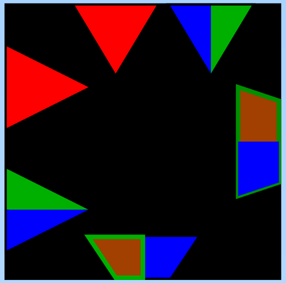


If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

Idea: Improve Gadget

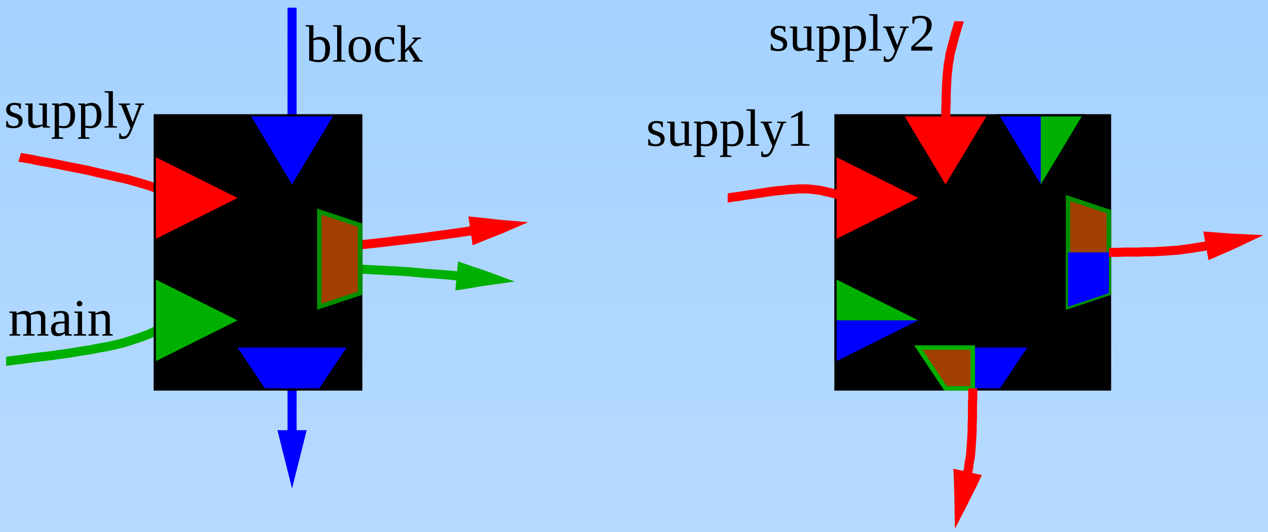




If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

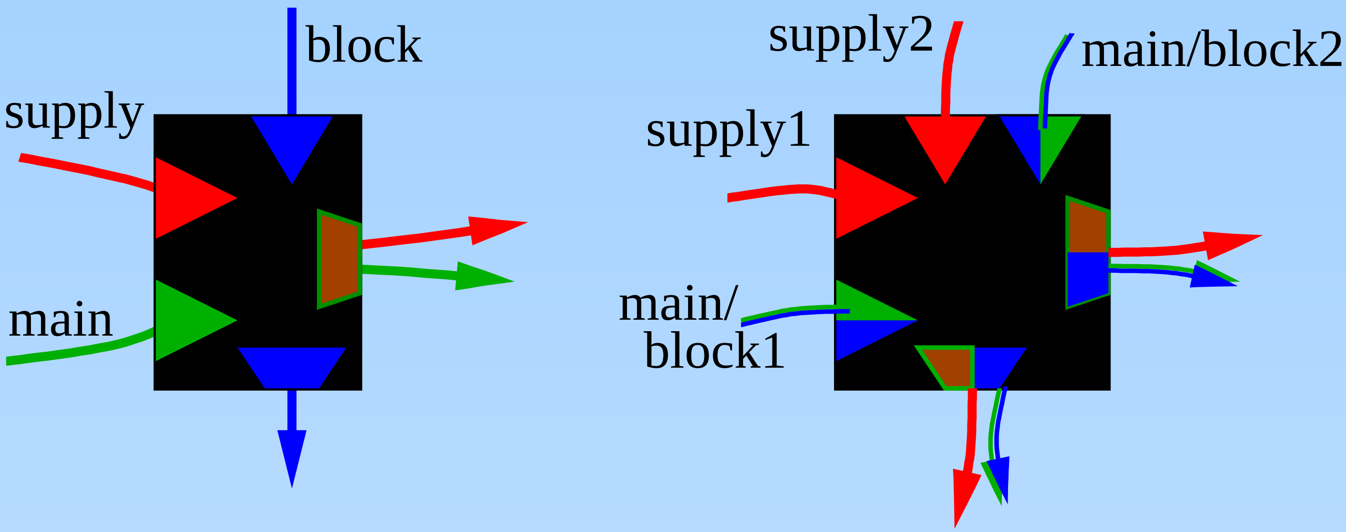
Idea: Improve Gadget



If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

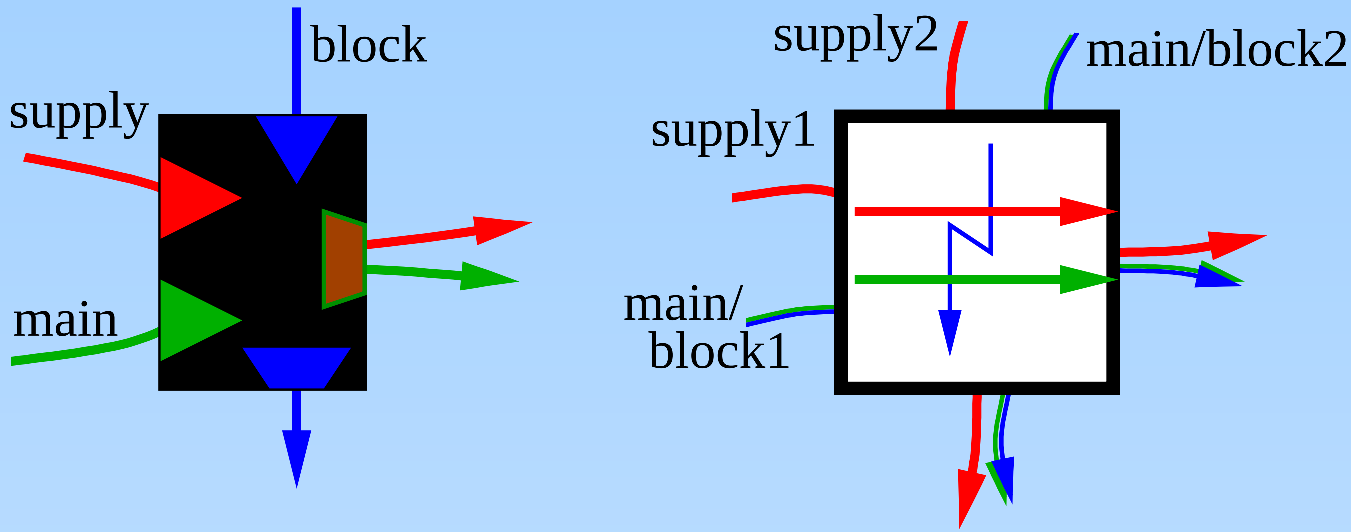
Idea: Improve Gadget



If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

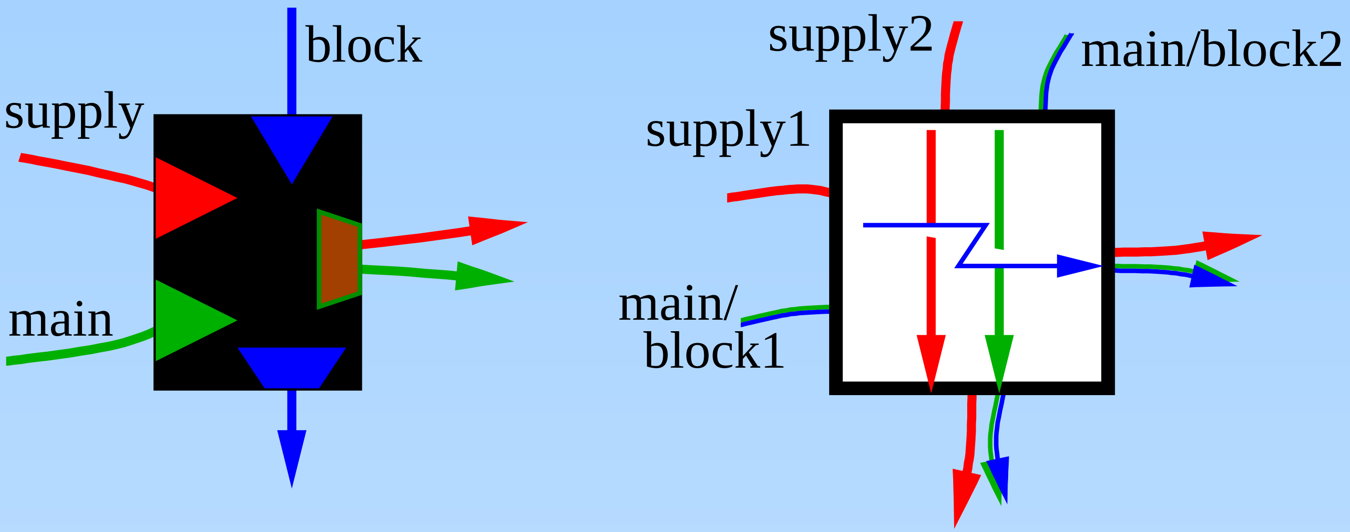
Idea: Improve Gadget



If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

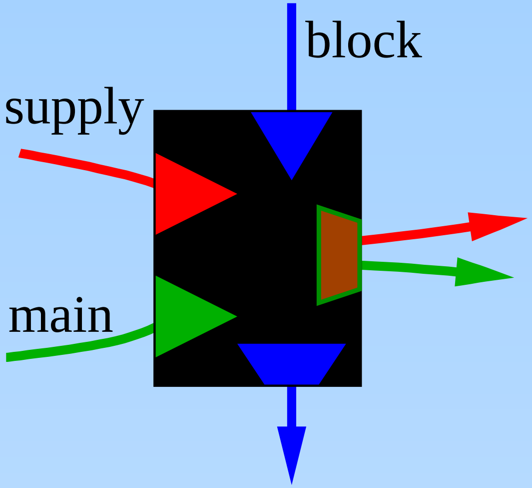
Idea: Improve Gadget

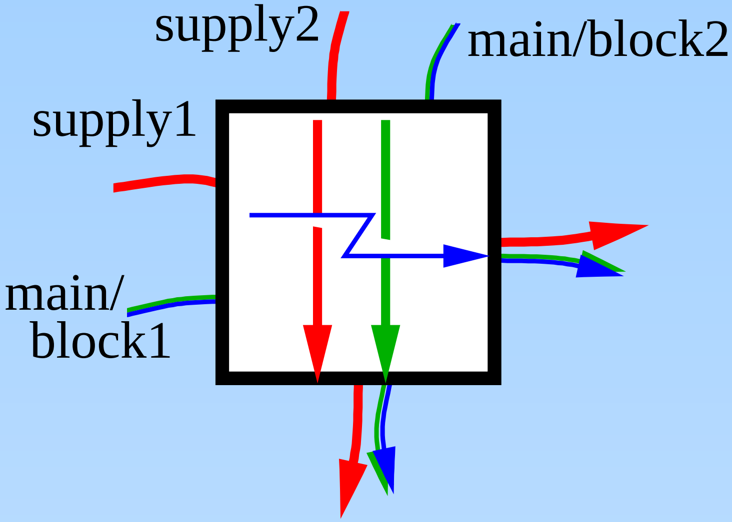


If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

Idea: Improve Gadget





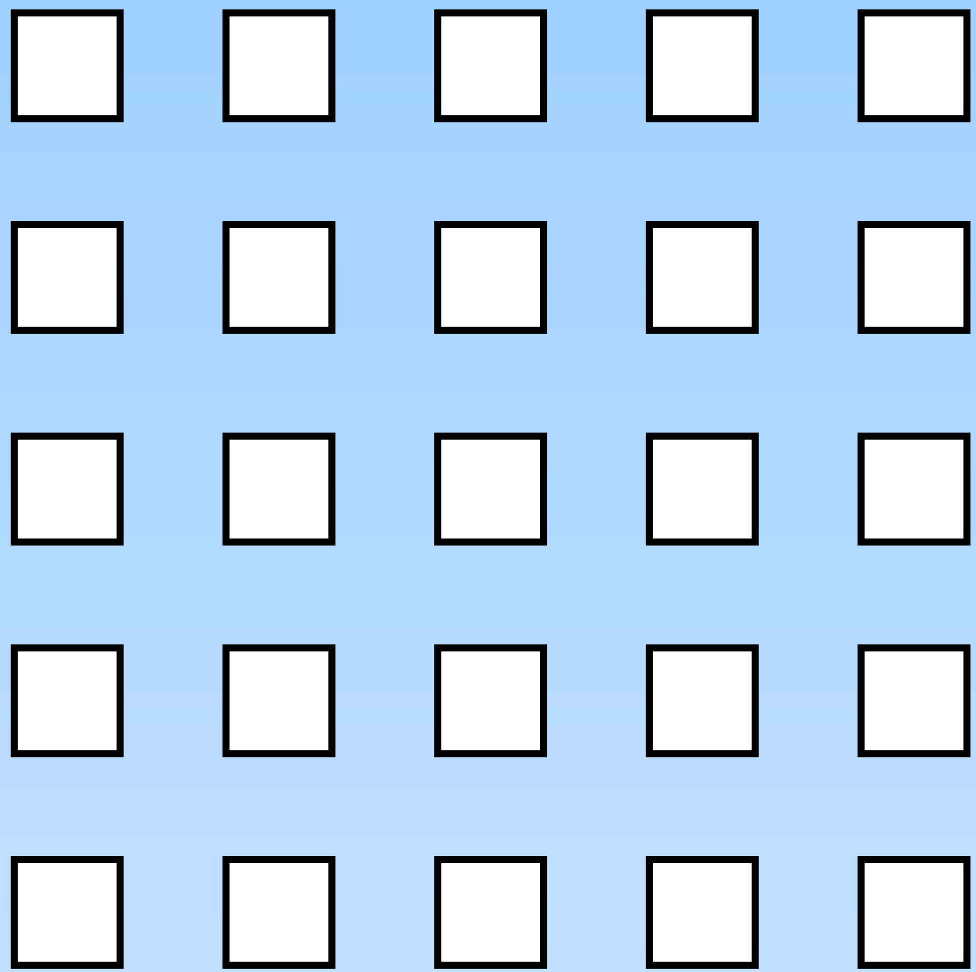
If green packets cannot be blocked by blue packets...

... then blue packets can be blocked by green packets.

Gadget works in at least one orientation.

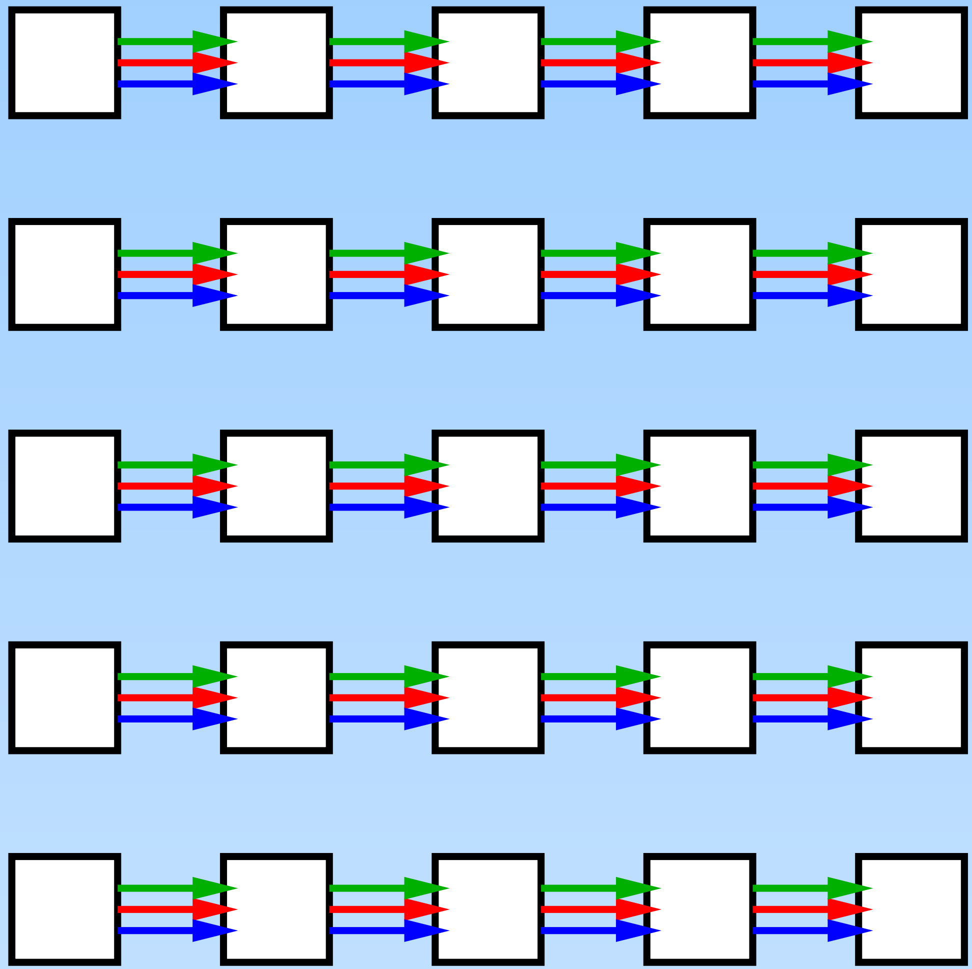
Arranging a Gadget Matrix

Arranging a Gadget Matrix



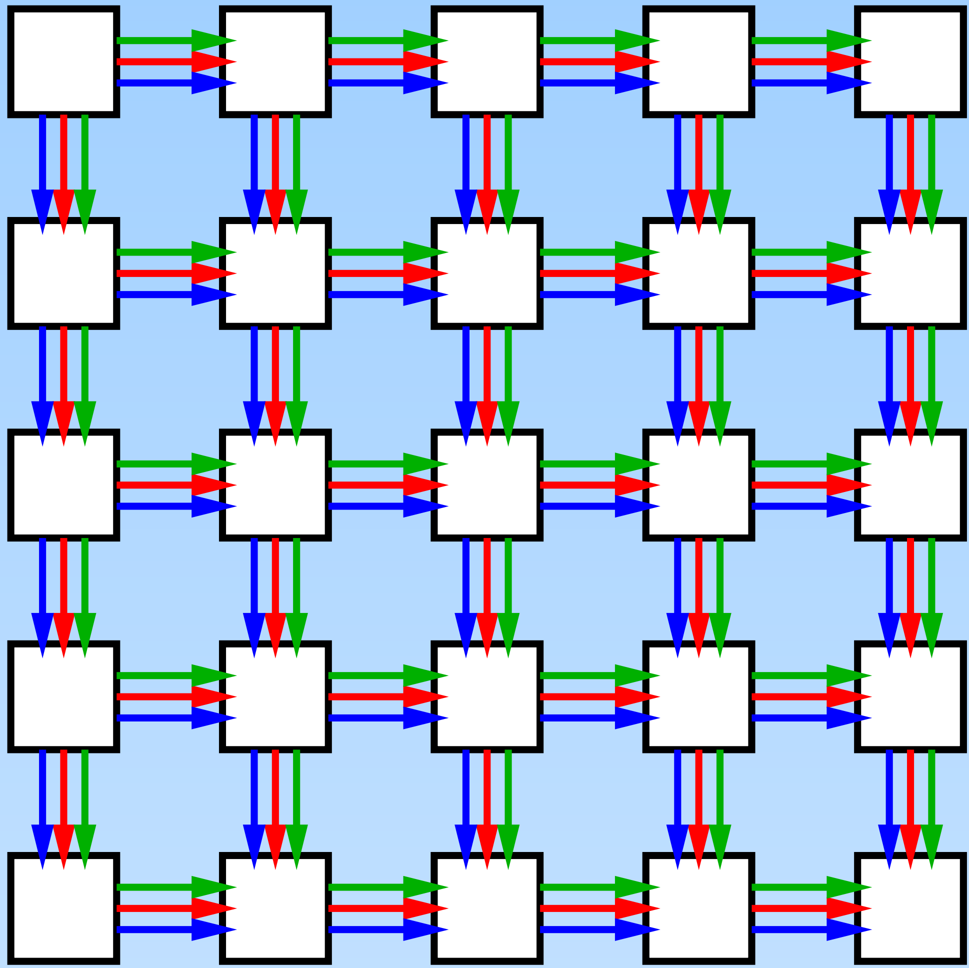
A $k \times k$ matrix of gadgets.

Arranging a Gadget Matrix



A $k \times k$ matrix of gadgets.

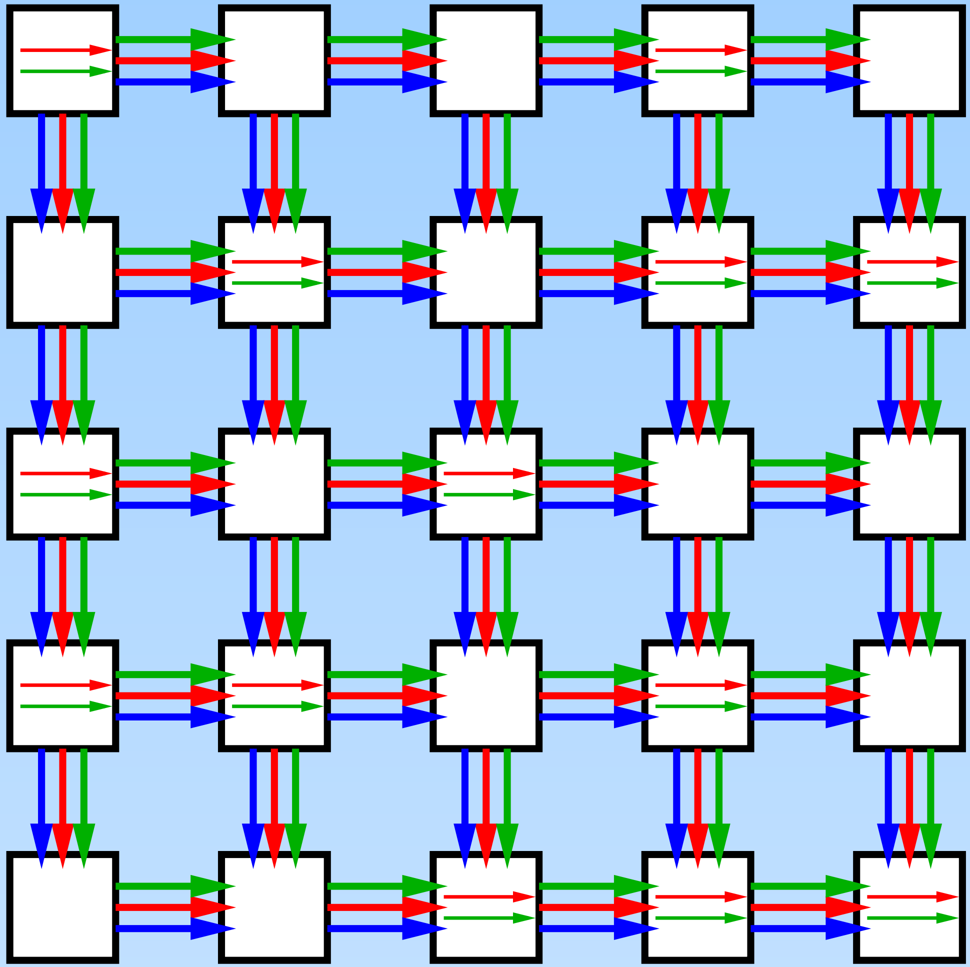
Arranging a Gadget Matrix



A $k \times k$ matrix of gadgets.

Diameter: $\Theta(k)$.

Arranging a Gadget Matrix

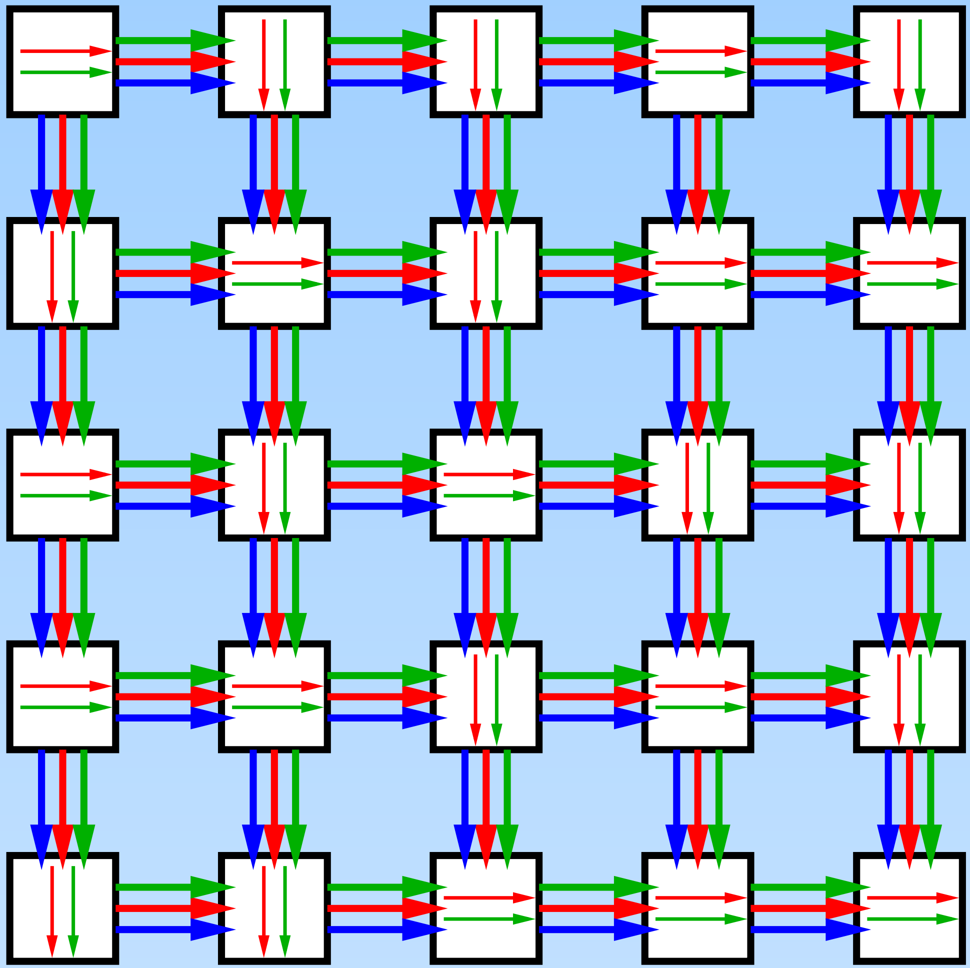


A $k \times k$ matrix of gadgets.

Diameter: $\Theta(k)$.

Each gadgets works in one orientation.

Arranging a Gadget Matrix



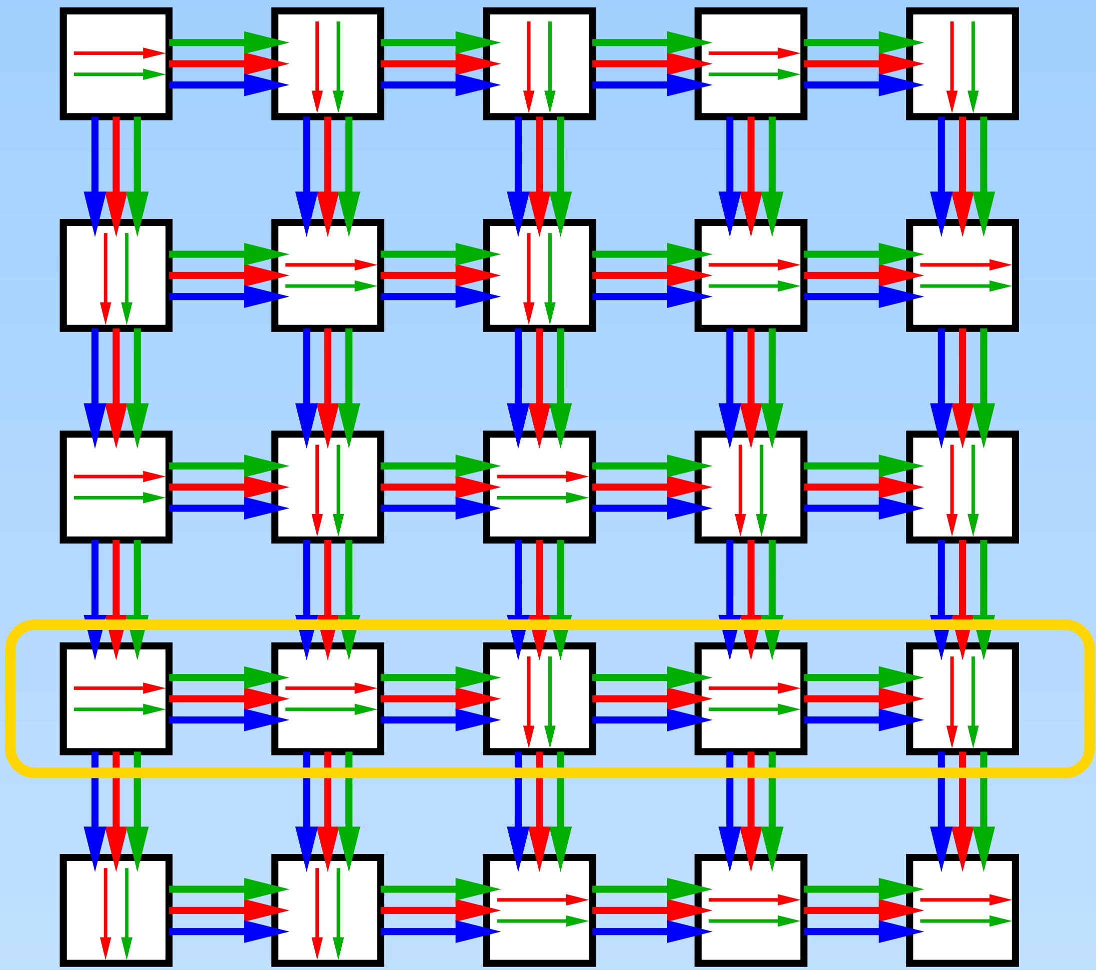
A $k \times k$ matrix of gadgets.

Diameter: $\Theta(k)$.

Each gadgets works in one orientation.

There exists one row/column with at least $k/2$ usable gadgets.

Arranging a Gadget Matrix



A $k \times k$ matrix of gadgets.

Diameter: $\Theta(k)$.

Each gadgets works in one orientation.

There exists one row/column with at least $k/2$ usable gadgets.

$2^{\Theta(k)}$ packets can be piled up.

Conclusions

Conclusions

- All WTS-Strategies can be forced into exponential delay.

Conclusions

- All WTS-Strategies can be forced into exponential delay.
  - The only *reasonable* quantities that WTS-strategies do not exploit are times.

Conclusions

- All WTS-Strategies can be forced into exponential delay.
  - The only *reasonable* quantities that WTS-strategies do not exploit are times.
  - Hence timekeeping appears crucial.

Conclusions

- All WTS-Strategies can be forced into exponential delay.
  - The only *reasonable* quantities that WTS-strategies do not exploit are times.
  - Hence timekeeping appears crucial.
  - LIS is among the *simplest* strategies that do use timekeeping.

Conclusions

- All WTS-Strategies can be forced into exponential delay.
  - The only *reasonable* quantities that WTS-strategies do not exploit are times.
  - Hence timekeeping appears crucial.
  - LIS is among the *simplest* strategies that do use timekeeping.

- New technique for proving 1-stability.

Conclusions

- All WTS-Strategies can be forced into exponential delay.
  - The only *reasonable* quantities that WTS-strategies do not exploit are times.
  - Hence timekeeping appears crucial.
  - LIS is among the *simplest* strategies that do use timekeeping.
- New technique for proving 1-stability.
- Complete classification of distance based WTS-strategies.

Conclusions

- All WTS-Strategies can be forced into exponential delay.
  - The only *reasonable* quantities that WTS-strategies do not exploit are times.
  - Hence timekeeping appears crucial.
  - LIS is among the *simplest* strategies that do use timekeeping.
- New technique for proving 1-stability.
- Complete classification of distance based WTS-strategies.

THANK YOU