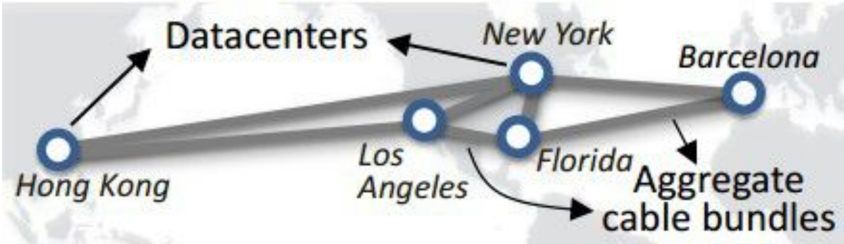
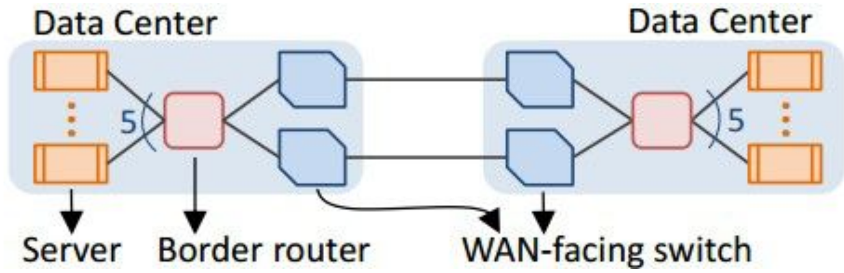
Achieving High Utilization with Software-Driven Wan

SIGCOMM 2013

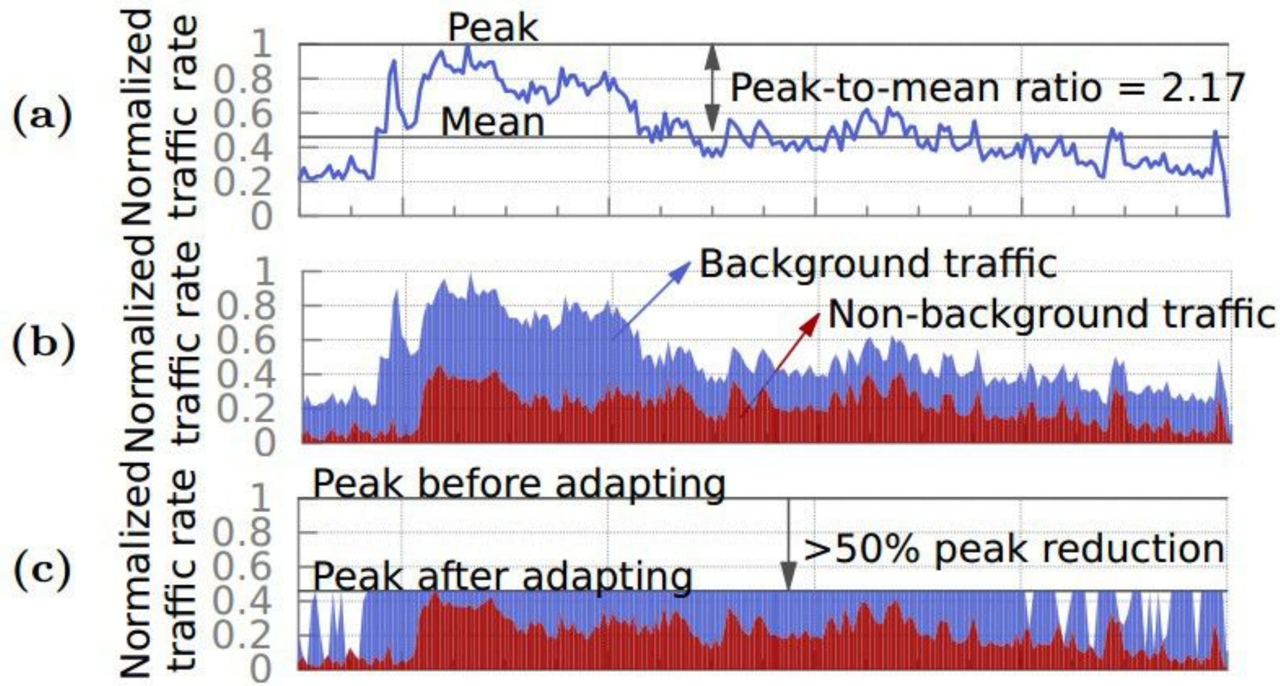
Inter-DC WAN





- Links connecting two data-centers is provided with capacity of up to 100s of Gbps to Tbps over a long distance, which is expensive to build and maintain.
- Services lack coordination and send traffic whennever they want and however much they want.
- The links are provisioned with peak loads. The average utilization of the busiest links is 40-60%.
- It is really important to increase the link utilization for Inter-DC WAN as well as preserve the fair share among different services.

Traffic Characteristics of a busy link of Inter-DC WAN



- (a) Daily traffic pattern on a busy link in a production inter-dc wan.
- (b) Breakdown based on traffic type.
- (c) Reduction in peak usage if background traffic is dynamically adapted.

Three types of traffic running on the Inter-DC WAN links.

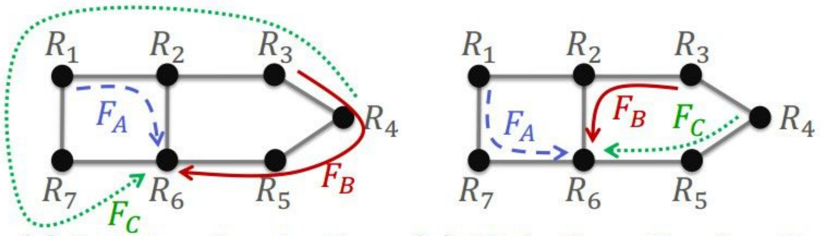
- Three types of traffic:
- Interactive:
  - Interactive traffic directly delivers contents to end users. It has a tight deadline and is sensitive to loss and delay.
  - Example: One DC contacts another in the process of responding to a user request because not all information is available in the first DC.
- Elastic:
  - Elastic traffic does not directly affect user experience, but still needs timely delivery. It has a medium deadline of a few seconds or minutes.
  - Example: Replicate a data update to anther DC.
- Background:
  - Background traffic conducts maintenance and provisioning activities with no explict deadlines.
  - Example: Copy all the data of a service to anther DC for long-term storage.
- Priority: Interactive>Elastic>Background

Current traffic engineering practice

- Most Inter-DC Networks are operated using MPLS TE.
- MPLS TE spreads traffic across tunnels between ingress-egress router pairs.
- MPLS TE find network paths using constraned shortest path first (CSPF) algorithm.
- Tunnels can be assigned priorities and different types of serervices are mapped into different tunnels.
- Packets carry differentiated services code point (DSCP) are mapped to different queues in switches.

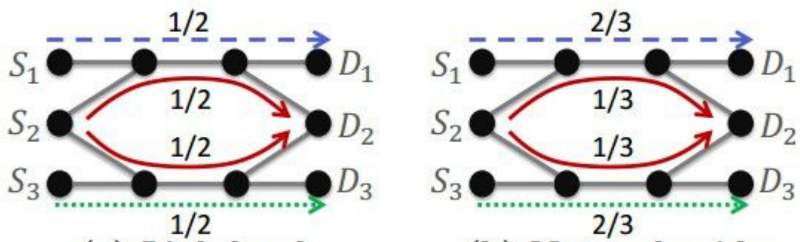
Problems of MPLS TE

- Poor efficiency:
  - Services sendwhenever and however much traffic they want, without regard to the current state of the network or other services.
  - The local, greedy resource allocation model of MPLS TE is inefficient.



(a) Local path selection (b) Globally optimal paths

- Poor sharing:
  - Link level fairness ≠ Network-wide fairness



(a) Link-level (b) Network-wide

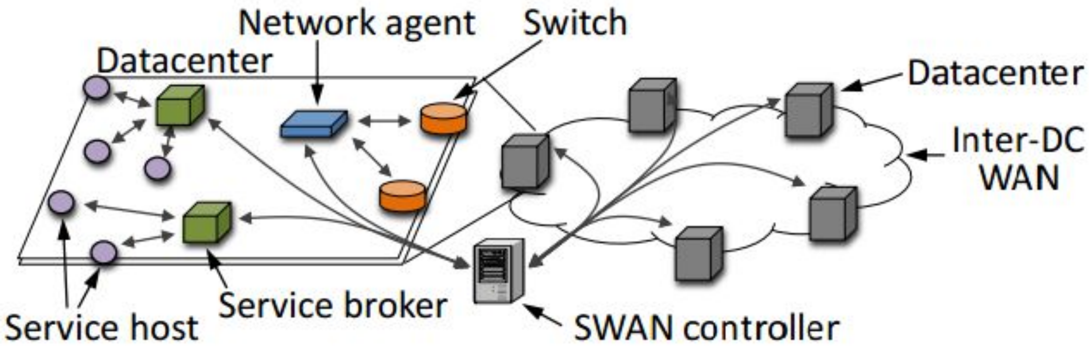
Swan overview

- Swan is a Inter-DC WAN traffic control system which increases the Inter-DC link utilization as well as supports flexible network sharing.

- Swan supports 3 priority classes: Interactive > Elastic > Background. Swan allocates bandwidth in decreasing order of the priority and prefers shorter paths for classes with higher priority. Within the same class, it allocates bandwidth according to max-min fair manner.

- Swan works as:
  - Elastic and background traffic inform the swan controller with their demand. Interactive traffic is directly sent without the assistance of the controller.
  - The controller is responsible for allocating the sending rates of different traffic and updating the data paths.
  - Swan uses OpenFlow switches for the data path updates.

Swan architecture



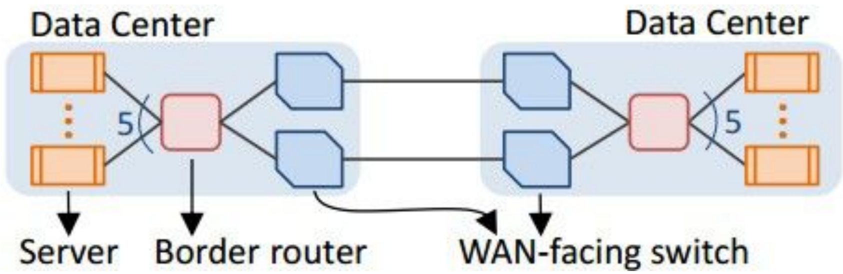
- Service host: allocation for next T_h=10s, token bucket to enforce allocated sending rate, DSCP bit to indicate priority class.
- Service broker: update the controller every T_s = 5 min about the aggregated demand, apportion its allocation from the controller to service host piecemeal, in time unit of T_h .
- Network agents: track topology and traffic information and report it to controller every T_a = 5 min, update switch rules as requested by controler.

Swan architecture

- Controller:
  - Compute service allocation and forwarding plane configuration.
  - Signal new allocation to services whose allocation has decreased. Wait for T_h seconds for the service to lower its sending rate.
  - Change the forwarding state and then signal the new allocation to services whose allocation has increased.

Forwarding plane configuration

- Swan uses label based forwarding, assign a label to traffic at the source switch.



- Use VLAN ID as label.
- Ingress switches split traffic across multiple tunnel with unequal splitting.

Compute service allocation

Inputs:

- d_i: flow demands for source destination pair i
- w_j: weight of tunnel j (e.g., latency)
- c_l: capacity of link l
- s_{Pri}: scratch capacity ($[0, 50\%]$) for class Pri
- $I_{j,l}$: 1 if tunnel j uses link l and 0 otherwise

Outputs:

- $b_i = \sum_j b_{i,j}$: b_i is allocation to flow i; $b_{i,j}$ over tunnel j

---

Func: SWAN Allocation:
$\forall$ links l : $c_l^{\text{remain}} \leftarrow c_l$; // remaining link capacity
for $Pri =$ Interactive, Elastic, $\ldots$, Background **do**
  $\{b_i\} \leftarrow$ Throughput Maximization / Approx. Max-Min Fairness $(Pri, \{c_l^{\text{remain}}\})$;
  $c_l^{\text{remain}} \leftarrow c_l^{\text{remain}} - \sum_{i,j} b_{i,j} \cdot I_{j,l}$;

Func: Throughput Maximization$(Pri, \{c_l^{\text{remain}}\})$:
return MCF$(Pri, \{c_l^{\text{remain}}\}, 0, \infty, \emptyset)$;

Func: Approx. Max-Min Fairness$(Pri, \{c_l^{\text{remain}}\})$:
// Parameters α and U trade-off unfairness for runtime
// $\alpha > 1$ and $0 < U \leq \min(\text{fairrate}_i)$
$T \leftarrow \lceil \log_\alpha \frac{\max(d_i)}{U} \rceil$; $F \leftarrow \emptyset$;
for $k = 1 \ldots T$ **do**
  foreach $b_i \in$ MCF$(Pri, \{c_l^{\text{remain}}\}, \alpha^{k-1}U, \alpha^k U, F)$ **do**
    if $i \notin F$ **and** $b_i < \min(d_i, \alpha^k U)$ **then**
      $F \leftarrow F + \{i\}$; $f_i \leftarrow b_i$; // flow saturated
return $\{f_i : i \in F\}$;

Func: MCF$(Pri, \{c_l^{\text{remain}}\}, b_{Low}, b_{High}, F)$:
//Allocate rate b_i for flows in priority class Pri

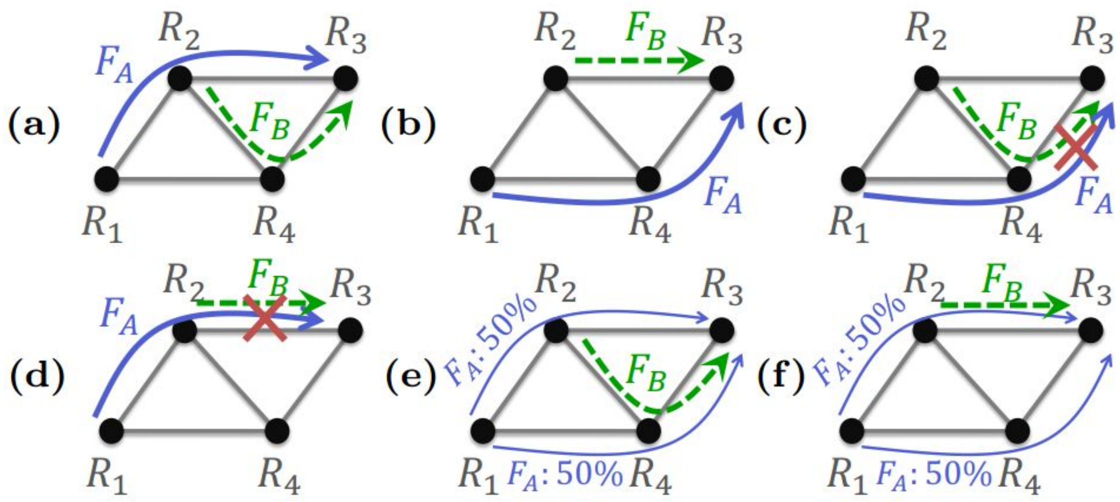
maximize $\sum_i b_i - \epsilon(\sum_{i,j} w_j \cdot b_{i,j})$

subject to
$\forall i \notin F : b_{Low} \leq b_i \leq \min(d_i, b_{High})$;
$\forall i \in F : b_i = f_i$;
$\forall l : \sum_{i,j} b_{i,j} \cdot I_{j,l} \leq \min\{c_l^{\text{remain}}, (1 - s_{Pri})c_l\}$;
$\forall (i,j) : b_{i,j} \geq 0$.

- Iterative max-min fairness computation method.
- Estimate the interactive service demand. Inflate the demand based on the error in past estimates.
- Output of the LP may violate switch's rule constraint. Choose a subset of tunnels with small latency and carry more traffic. Rerun the LP using this subset of tunnels as input.
- Note that the scratch capacity is deliberately left to facilitate congestion free update.

Update forwarding state

- Congestion free update:



- Each flow's size is 1 unit and each link's capacity is 1.5 units. Changing from state (a) to (b) may lead to congested states (c) or (d). A congestion-free update sequence is (a) -> (e) -> (f) ->(b).

Update forwarding state

$$
\textbf{Inputs:}\ \begin{cases} q, & \text{sequence length} \\ b^0_{i,j} = b_{i,j}, & \text{initial configuration} \\ b^q_{i,j} = b'_{i,j}, & \text{final configuration} \\ c_l, & \text{capacity of link } l \\ I_{jl}, & \text{indicates if tunnel } j \text{ using link } l \end{cases}
$$

$$
\textbf{Outputs:}\ \{b^a_{i,j}\}\ \ \forall a \in \{1, \dots q\} \text{ if feasible}
$$

$$
\begin{aligned}
\text{maximize} \quad & c_{\text{margin}} \ // \text{ remaining capacity margin} \\
\text{subject to} \quad & \forall i, a : \textstyle\sum_j b^a_{i,j} = b_i; \\
& \forall l, a : c_l \geq \textstyle\sum_{i,j} \max(b^a_{i,j}, b^{a+1}_{i,j}) \cdot I_{j,l} + c_{\text{margin}}; \\
& \forall (i, j, a) : b^a_{i,j} \geq 0;\ c_{\text{margin}} \geq 0;
\end{aligned}
$$

- Swan uses the above LP to get a congestion free update sequence.
- Swan also guarantees that the sequence length is less than $\lceil 1/s \rceil - 1$ steps.

From congestion free to bounded congestion

- Note that the scratch capacity left on the link is not utilized by the allocation procedure. Leaving it unutilized is intolerable.
- Background traffic is tolerate to moderate congestion. The scratch capacity of background traffic could be set to 0. While it eliminates the possibility of a congestion free update, by modifying the LP in previous slide, swan ensures that only background traffic experiences moderate congestion.
- Replace

$$\forall l, a : c_l \geq \sum_{i,j} \max\left(b_{i,j}^{a}, b_{i,j}^{a+1}\right) \cdot I_{j,l} + c_{margin}$$

- by

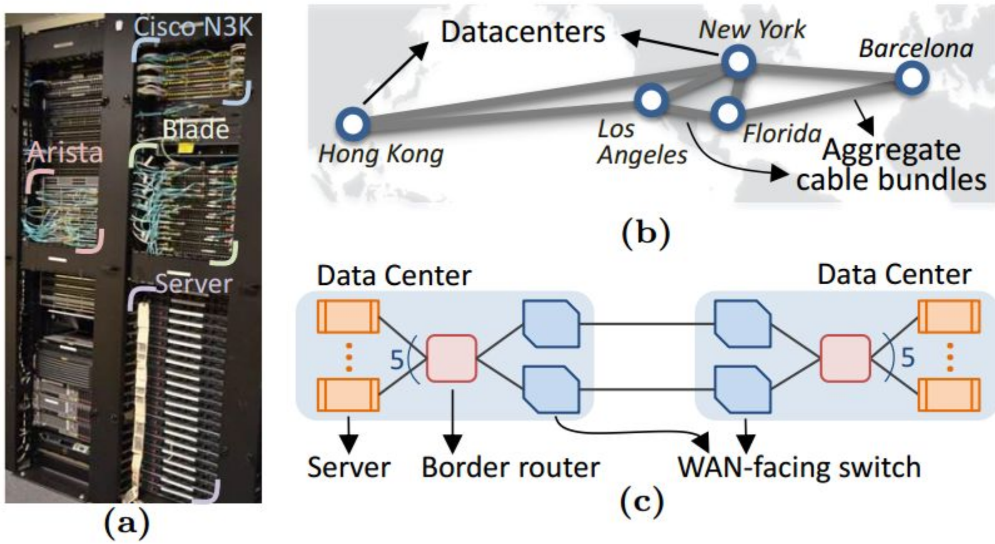
$$\forall l, a : (1+\eta) \cdot c_l \geq \sum_{i,j} \max\left(b_{i,j}^{a}, b_{i,j}^{a+1}\right) \cdot I_{j,l} + c_{margin} . \eta \in [0,50\%]$$

$$\forall l, a, i \in non-background\ \ flow : c_l \geq \sum_{i,j} \max\left(b_{i,j}^{a}, b_{i,j}^{a+1}\right) \cdot I_{j,l} + c_{margin}$$

From congestion free to bounded congestion

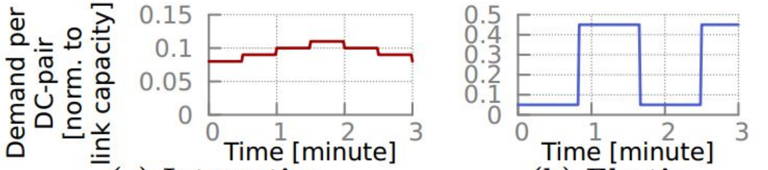
- After modifying the constraint, swan guarantees that there is a feasible solution within $\max(\lceil 1/s \rceil - 1, \lceil 1/\eta \rceil)$ steps such that non-background traffic never encounters loss caused by the congestion and background traffic encounters no more than η fraction loss.
- Swan set $\eta = \frac{s}{1-s}$ so that it ensure the same $\lceil 1/s \rceil - 1$ bound on steps as before.

Swan emulation result



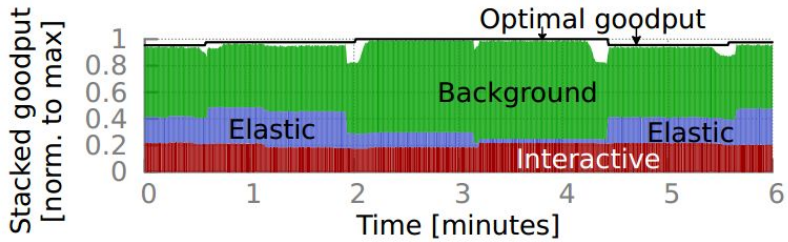
- (a) Equipment.
- (b) Emulation topology.
- (c) Connectivity of two DCs.

Swan emulation result

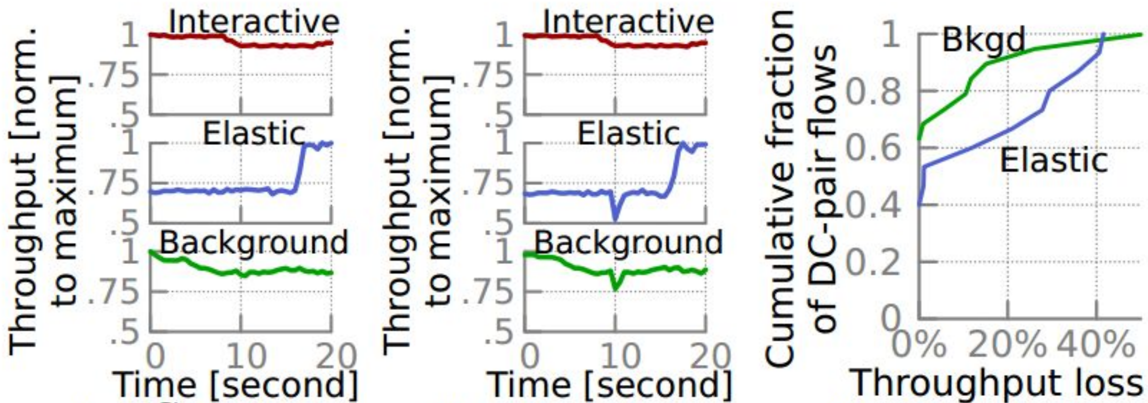


(a) Interactive (b) Elastic

Demand patterns for testbed experiments.



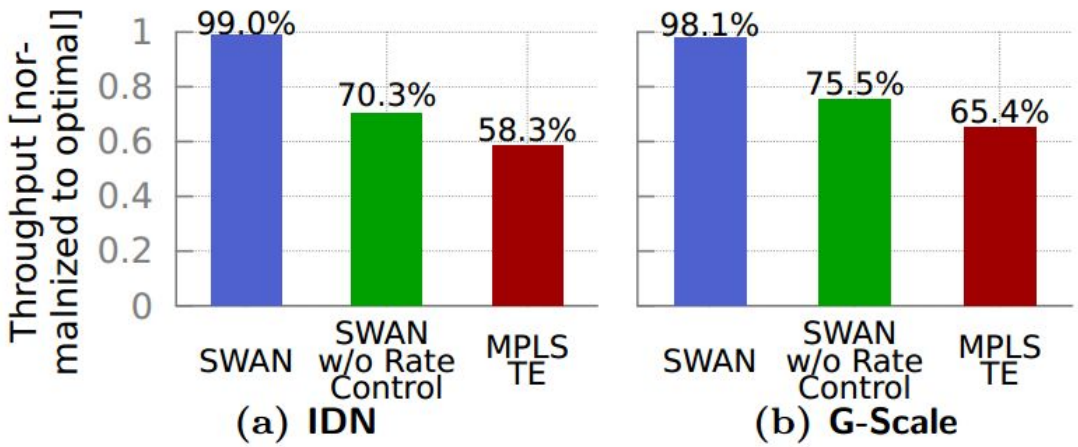
SWAN achieves near-optimal throughput.



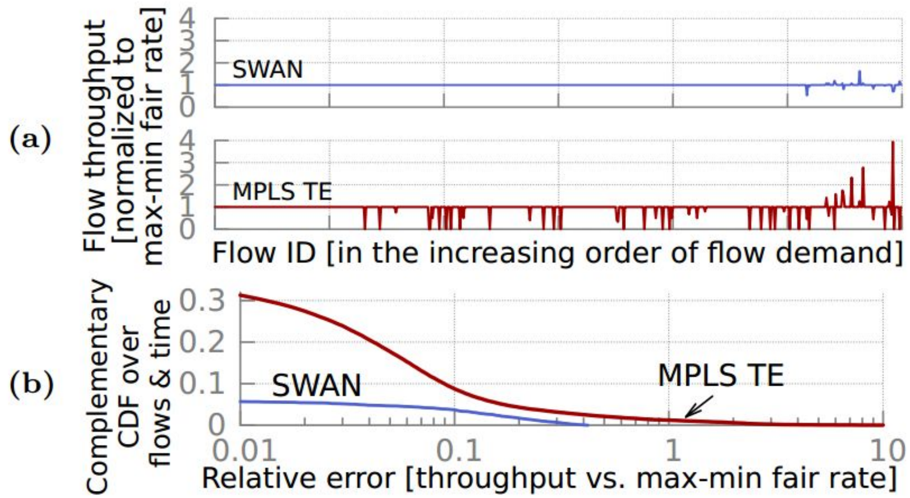
(a) SWAN (b) One-shot (c) One-shot

Updates in SWAN do not cause congestion.

Swan simulation result



SWAN carries more traffic than MPLS TE.



SWAN is fairer than MPLS TE.