Traffic Engineering with **Forward Fault Correction (FFC)**

Hongqiang "Harry" Liu,
Srikanth Kandula, Ratul Mahajan, Ming Zhang,
David Gelernter (Yale University)

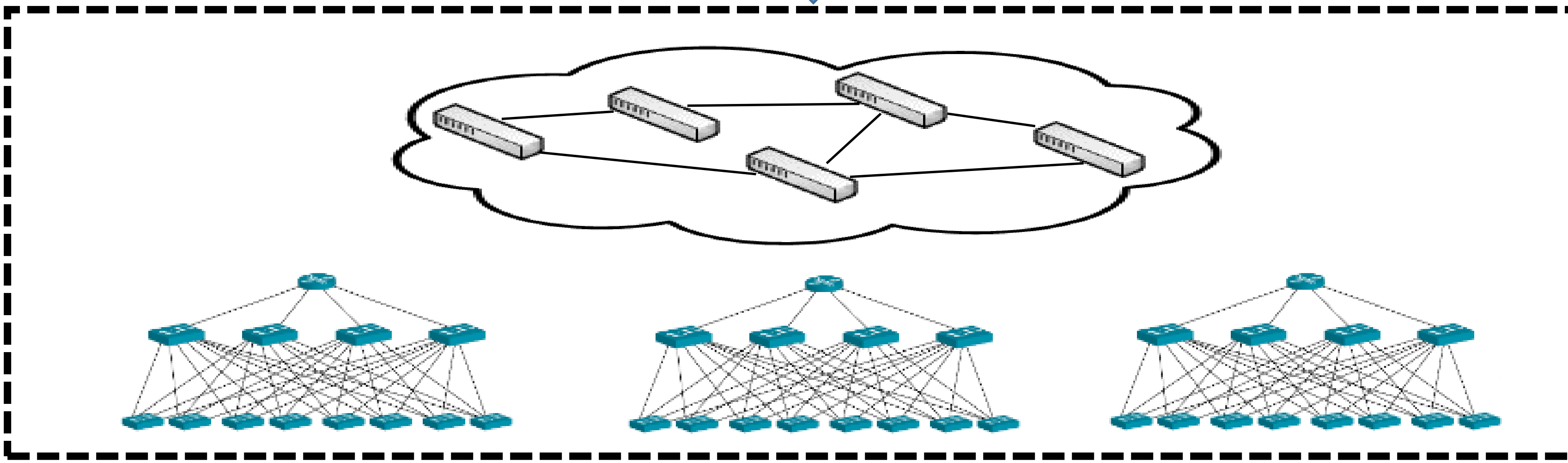
Microsoft Research

Cloud services require large network capacity

Cloud Services

Growing traffic

Cloud Networks

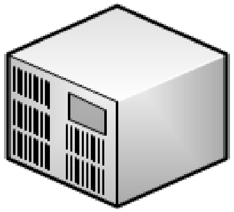


Expensive

(e.g. cost of WAN: $100M/year)

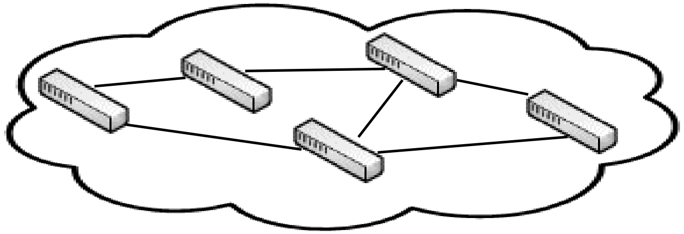
TE is critical to effectively utilizing networks

Traffic Engineering

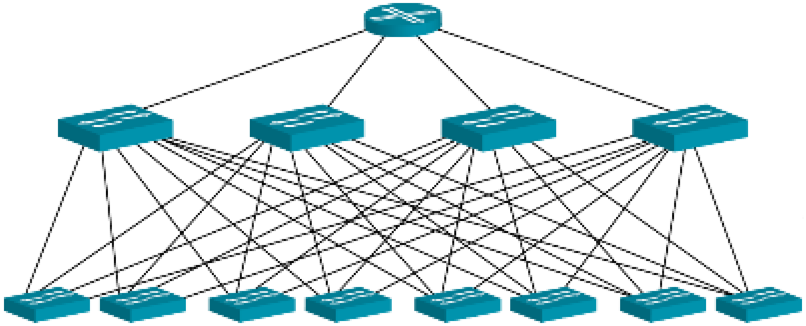


- Microsoft SWAN
- Google B4
- ……

- Devoflow
- MicroTE
- ……

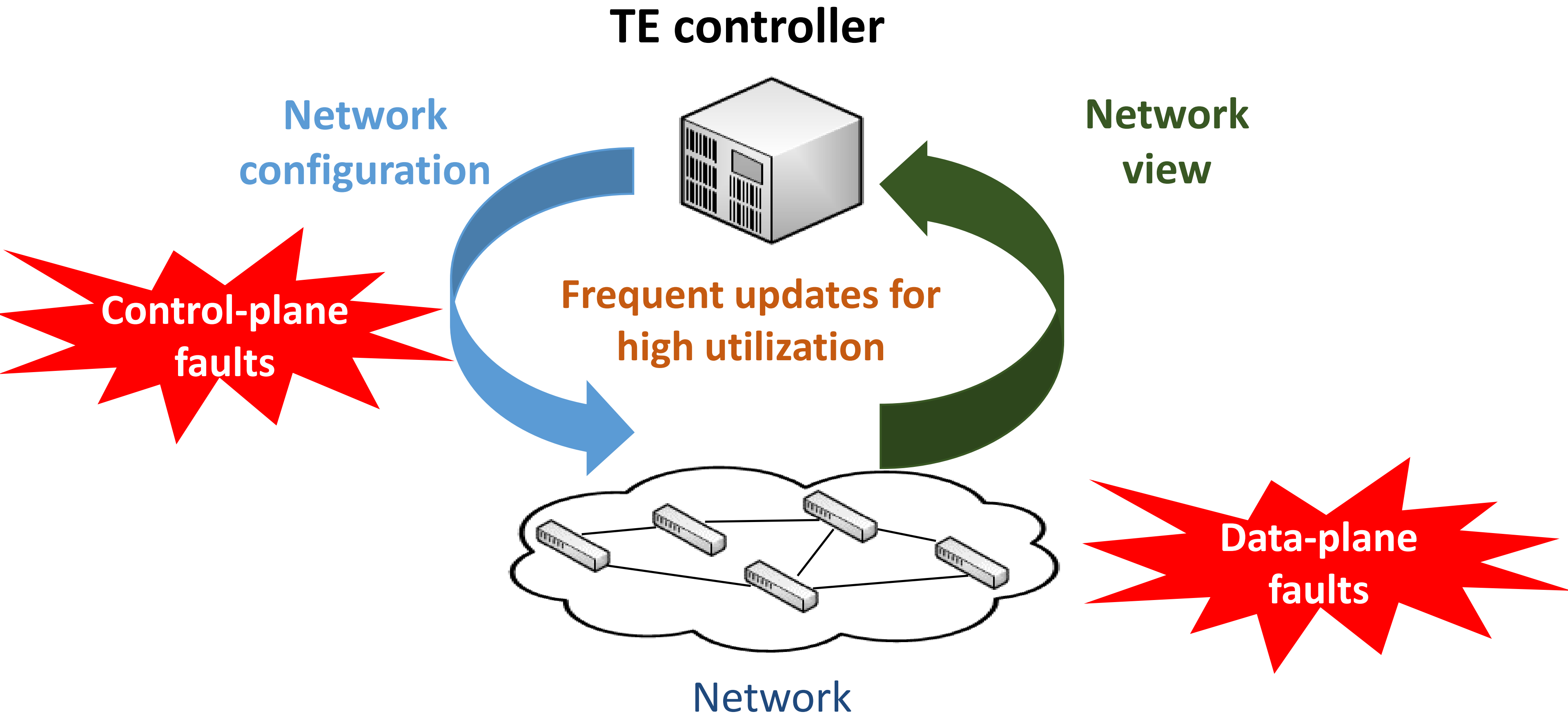


WAN Network



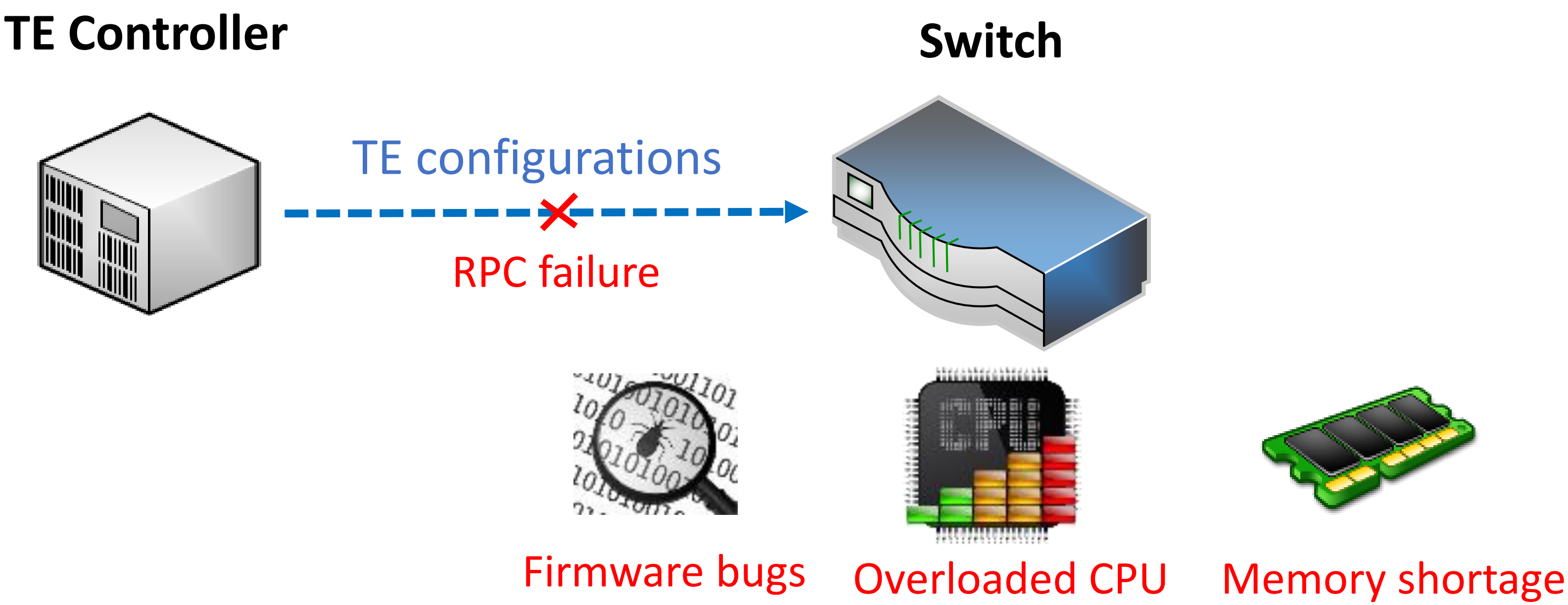
Datacenter Network

But, TE is also vulnerable to faults

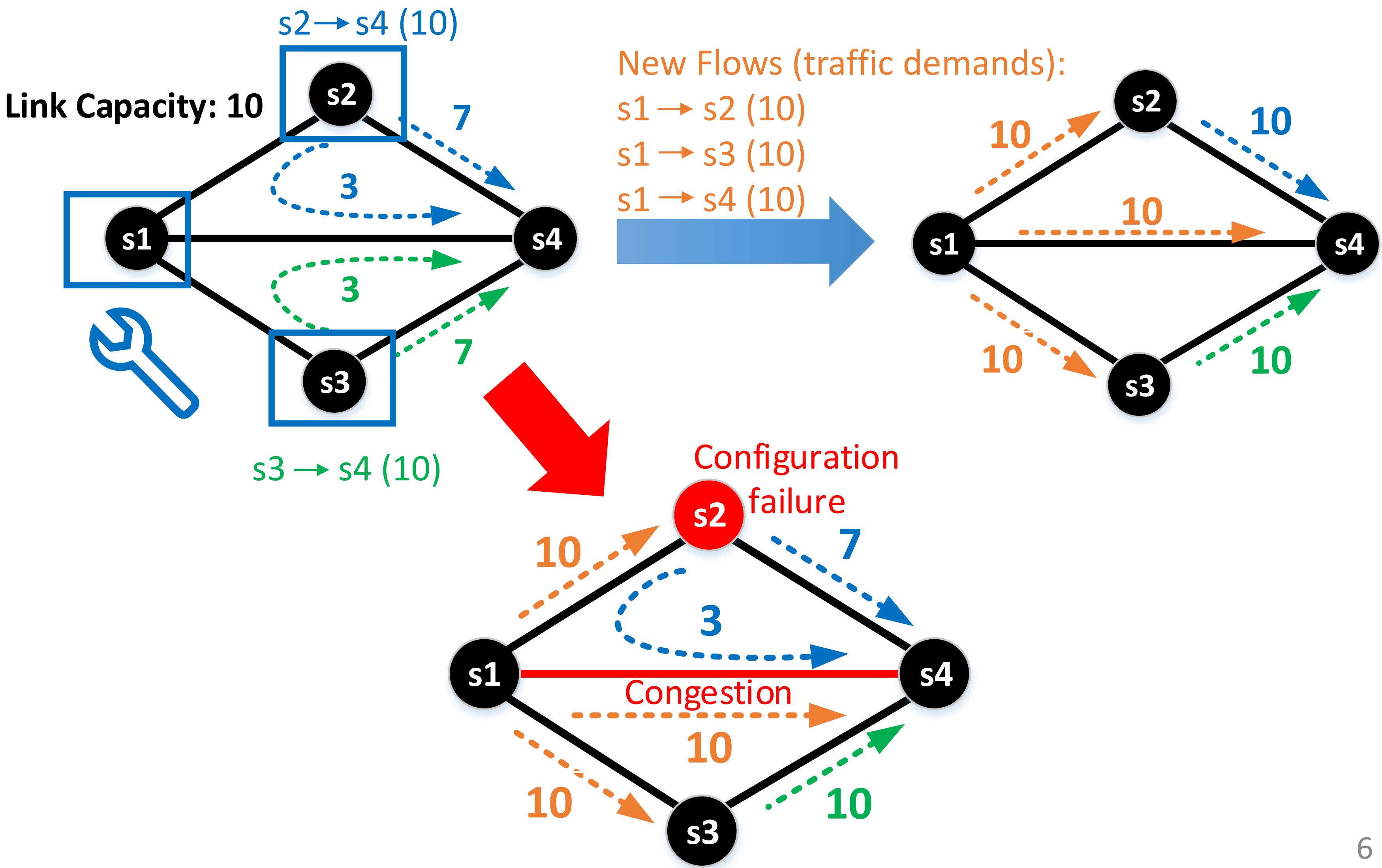


Control plan faults

Failures or long delays to configure a network device

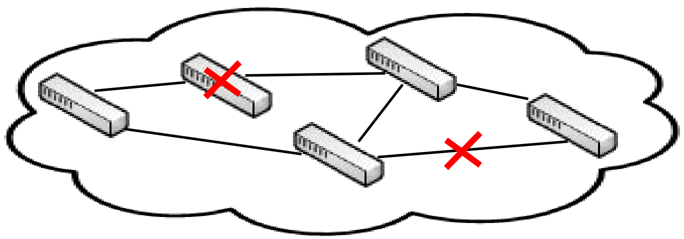


Congestion due to control plane faults

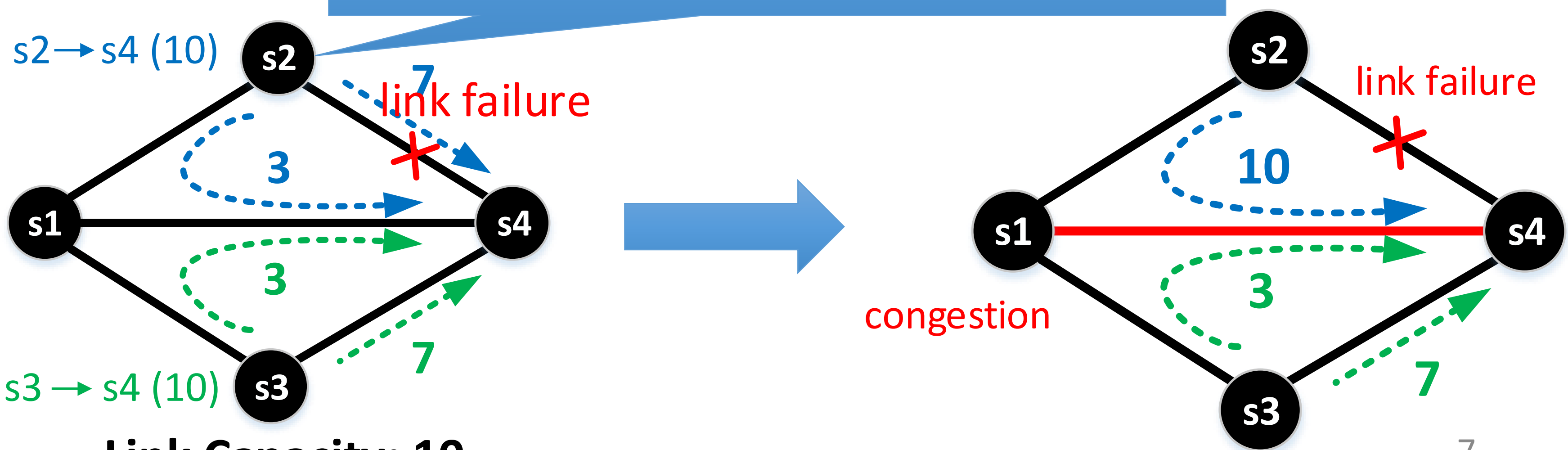


Data plane faults

Link and switch failures



Rescaling: Sending traffic proportionally to residual paths



Link Capacity: 10

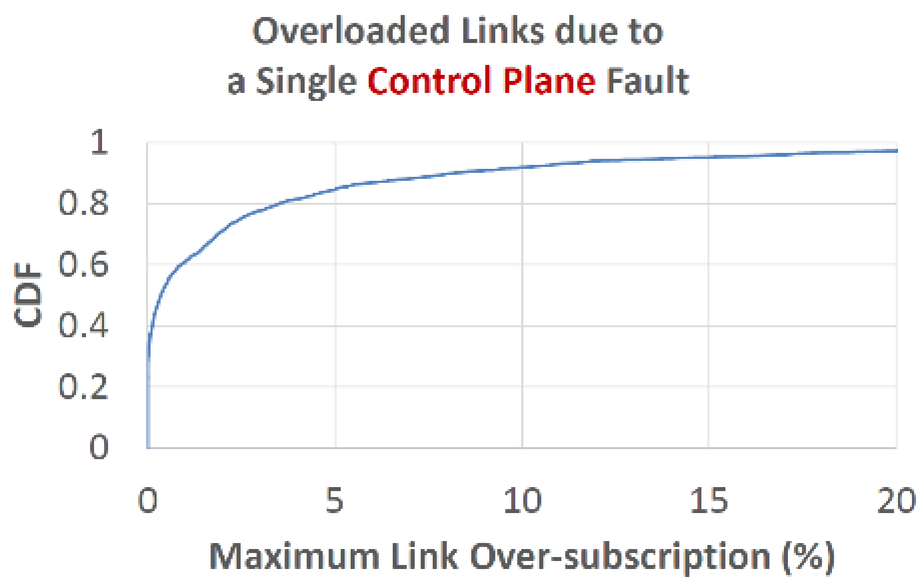
Control and data plane faults in practice

In production networks:

- Faults are common.
- Faults cause severe congestion.

Control plane:
fault rate = **0.1% -- 1%** per TE update.

Data plane:
fault rate = **25%** per 5 minutes.



State of the art for handling faults

- Heavy over-provisioning

Big loss in throughput

- Reactive handling of faults
  - Control plane faults: retry
  - Data plane faults: re-compute TE and update networks

Cannot prevent congestion

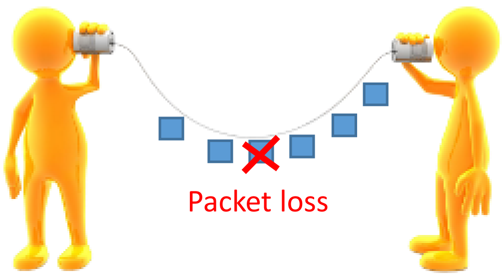
Slow
(seconds -- minutes)

Blocked by control plane faults

How about handling congestion proactively?

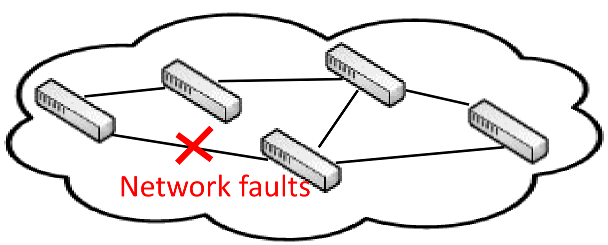
Forward fault correction (FFC) in TE

- [Bad News] Individual faults are **unpredictable**.
- [Good News] Simultaneous #faults is **small**.



FEC guarantees no information loss under up to k arbitrary packet drops.

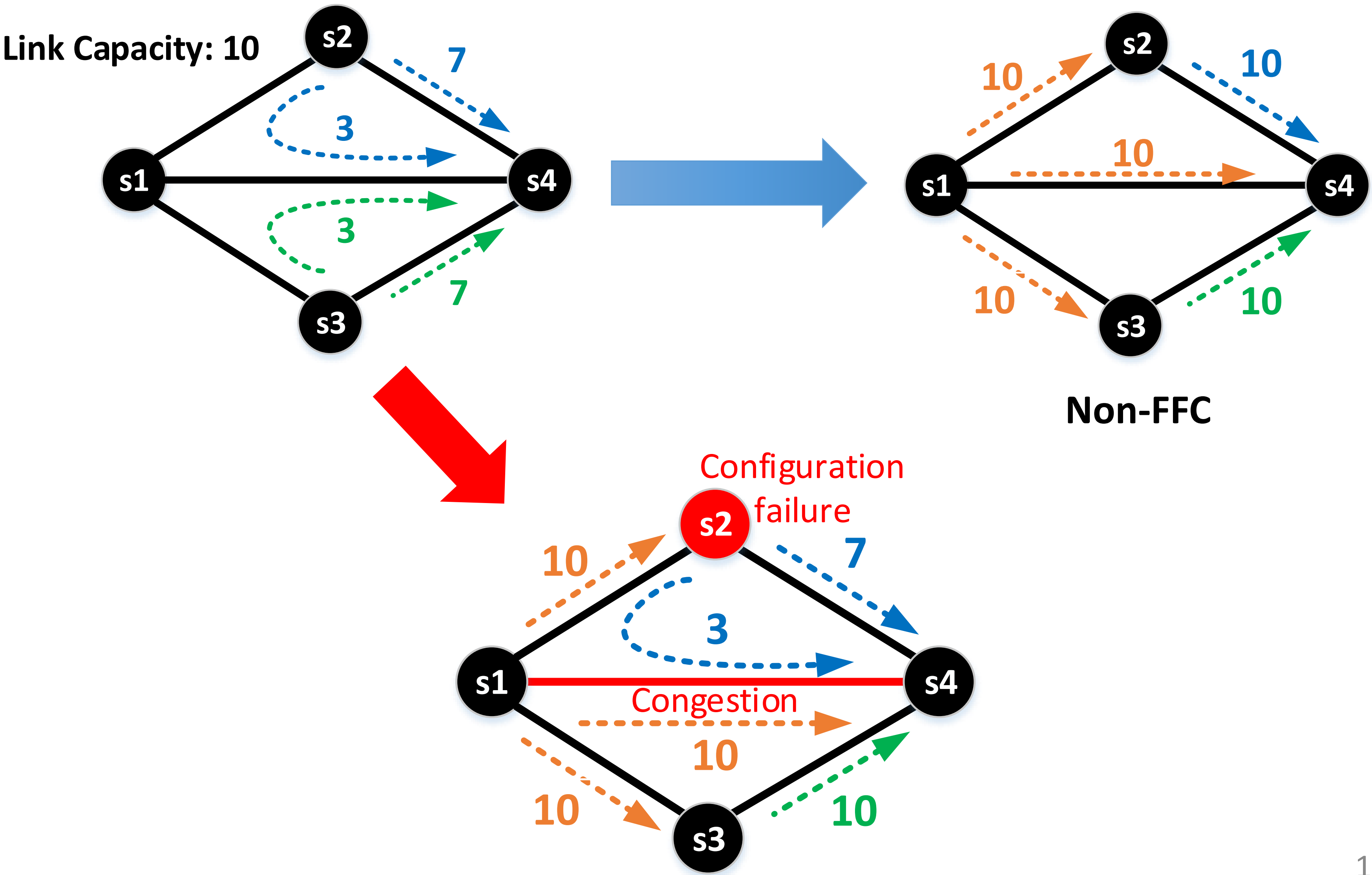
with careful **data encoding**



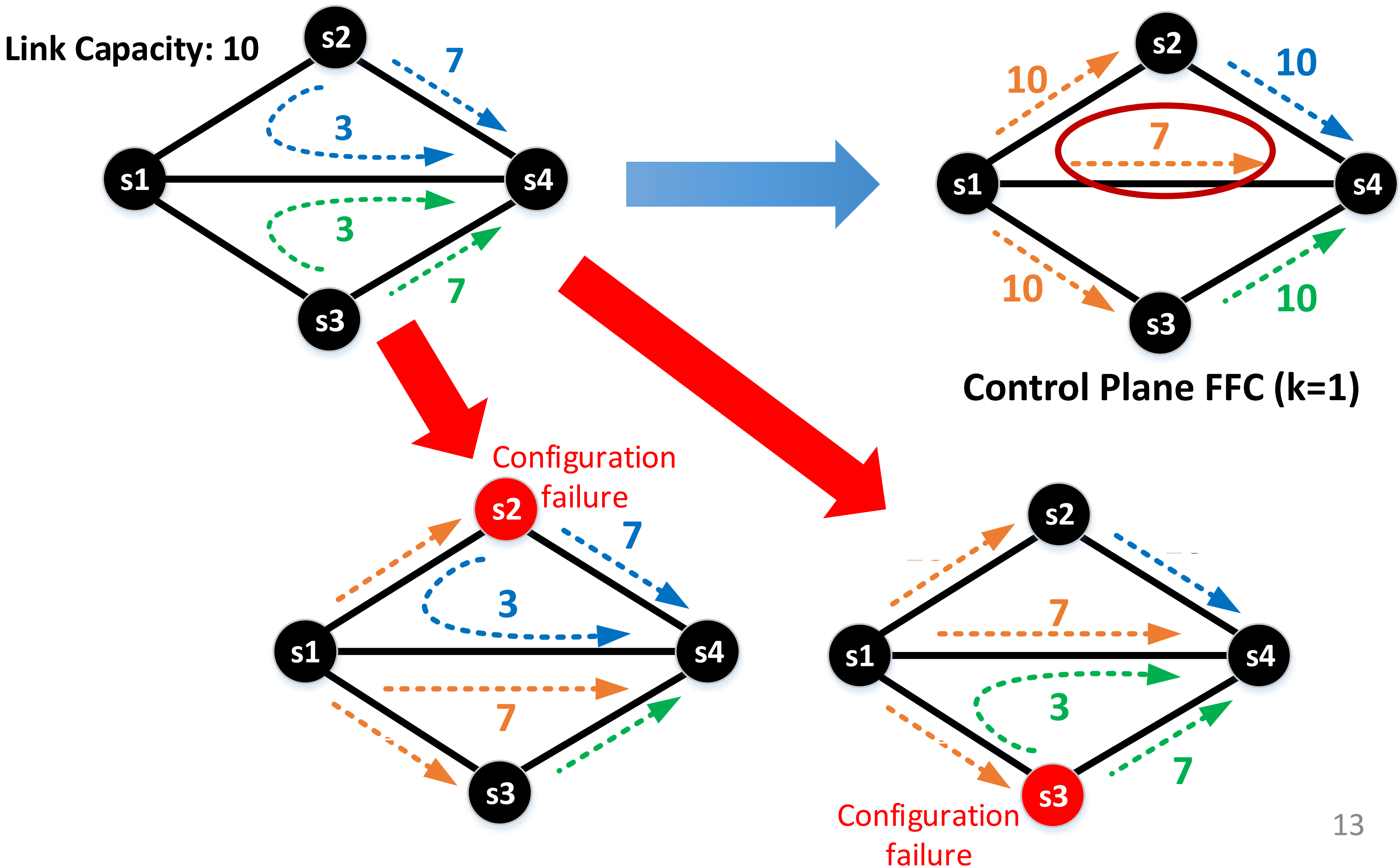
FFC guarantees no congestion under up to *k* arbitrary faults.

with careful **traffic distribution**

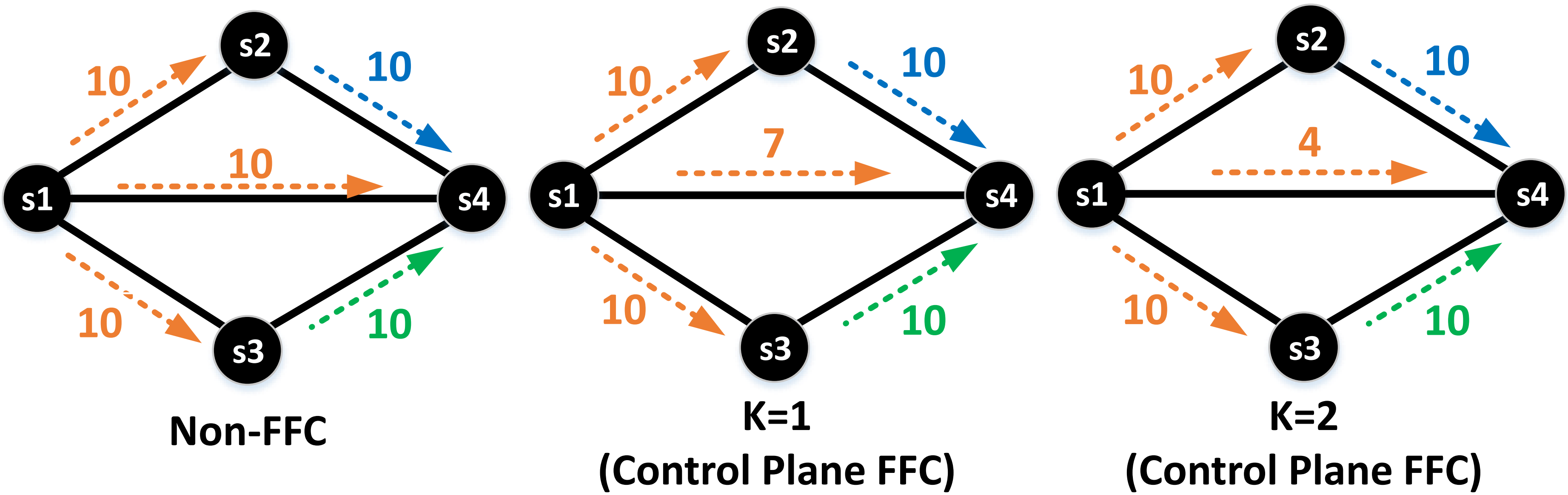
Example: FFC for control plane faults



Example: FFC for control plane faults



Trade-off: network utilization vs. robustness



Throughput: 50

Throughput: 47

Throughput: 44

Systematically realizing FFC in TE

Formulation:
How to merge FFC into existing TE framework?

Computation:
How to find FFC-TE efficiently?

Basic TE linear programming formulations

LP formulations

TE decisions:
- Sizes of flows — b_f
- Traffic on paths — $l_{f,t}$

TE objective: Maximizing throughput — max. $\sum_{\forall f} b_f$

Basic TE constraints:
- Deliver all granted flows — s.t. $\forall f$: $\sum_{\forall t} l_{f,t} \geq b_f$
- No overloaded link — $\forall e$: $\sum_{\forall f} \sum_{\forall t \ni e} l_{f,t} \leq c_e$
- … — …

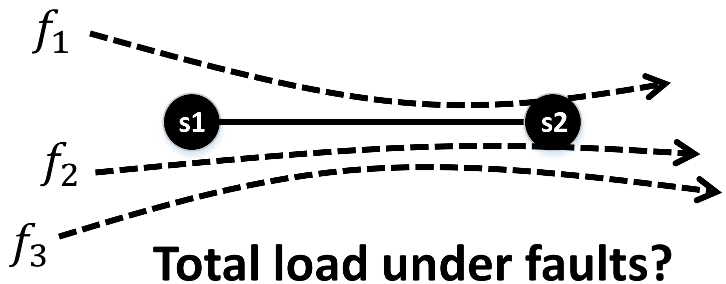
+

FFC constraints:

No overloaded link up to
- k_c control plane faults
- k_e link failures
- k_v switch failures

+

Formulating control plane FFC



Total load under faults?

f_1's load in old TE

f_2's load in new TE

$$\left.\begin{array}{ll}\text{Fault on } f_1: & l_1^{old} + l_2^{new} + l_3^{new} \le \text{link cap} \\ \text{Fault on } f_2: & l_1^{new} + l_2^{old} + l_3^{new} \le \text{link cap} \\ \text{Fault on } f_3: & l_1^{new} + l_2^{new} + l_3^{old} \le \text{link cap}\end{array}\right\} \quad \binom{\mathbf{3}}{\mathbf{1}}$$

Challenge: too many constraints

With ***n*** flows and FFC protection ***k***:

#constraints = $\binom{n}{1} + \ldots + \binom{n}{k}$ for each link.

An efficient and precise solution to FFC

Our approach:

A **lossless** compression from O($\binom{n}{k}$) constraints to O(kn) constraints.

Total load under faults $\leq$ link capacity ⇔ Total additional load due to faults $\leq$ link spare capacity

x_i: additional load due to **fault-*i***

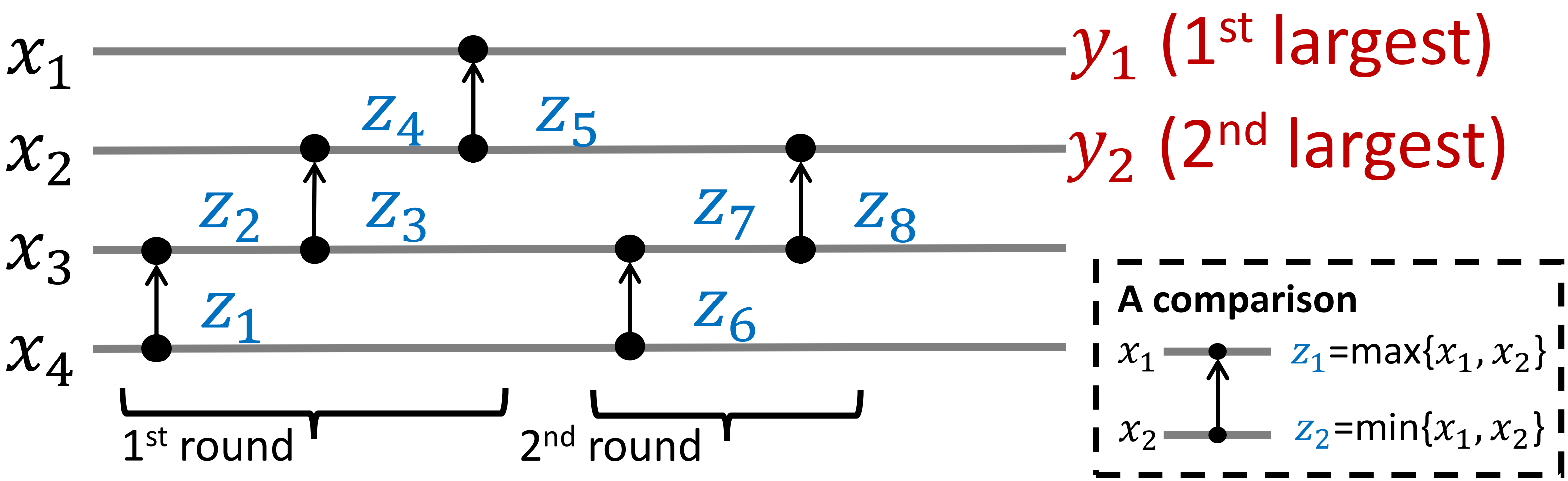
Given $X = \{x_1, x_2, \ldots, x_n\}$, FFC requires that **the sum** of ***arbitrary k*** elements in X is $\leq$ link spare capacity — O($\binom{n}{k}$)

⇕

Define y_m as the ***m*th *largest*** element in X:
$\sum_{m=1}^{k} y_m \leq$ link spare capacity — O(1)

Expressing y_m with X?

Sorting network



$y_1 + y_2 \leq$ link spare capacity

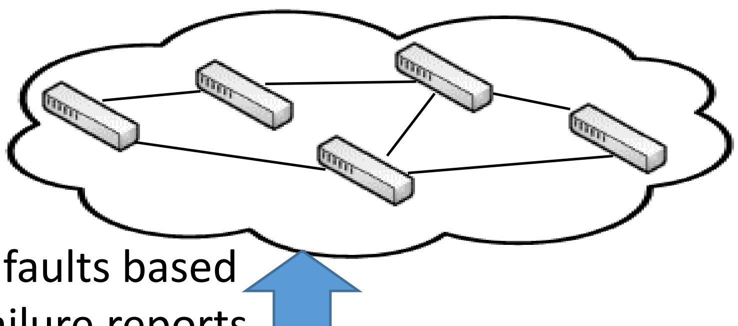
- **Complexity:** O(kn) additional variables and constraints.
- **Throughput:** optimal in control-plane and data plane if paths are disjoint.

FFC extensions

- Differential protection for different traffic priorities
- Minimizing congestion risks without rate limiters
- Control plane faults on rate limiters
- Uncertainty in current TE
- Different TE objectives (e.g. max-min fairness)
- …

Evaluation overview

- Testbed experiment
  - FFC can be implemented in commodity switches
  - FFC has no data loss due to congestion under faults
- Large-scale simulation



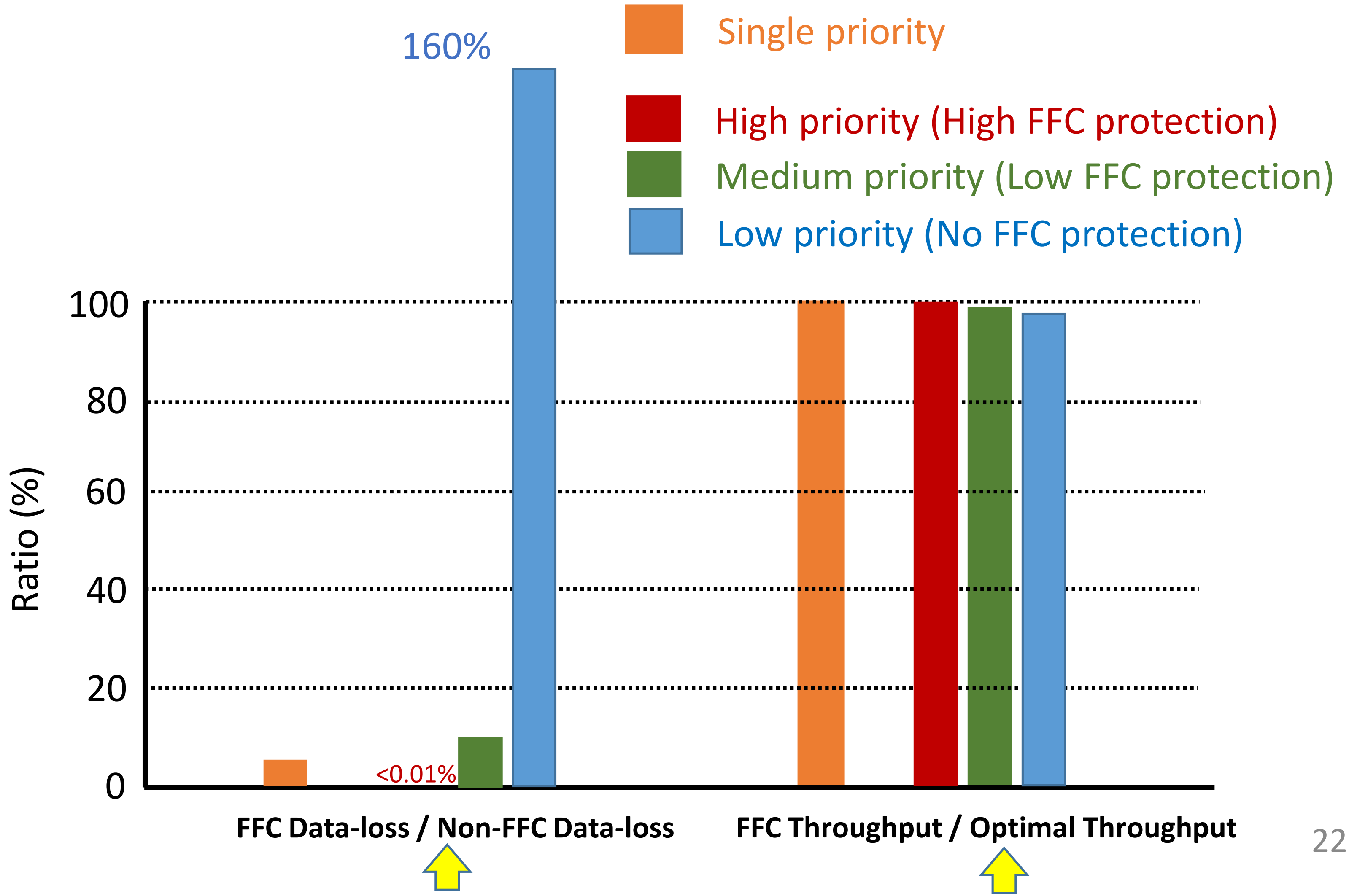
A WAN network with O(100) switches and O(1000) links

Injecting faults based on real failure reports

Single priority traffic in a well-provisioned network

Multiple priority traffic in a well-utilized network

FFC prevents congestion with negligible throughput loss



Conclusions

- Centralized TE is critical to high network utilization but is vulnerable to control and data plane faults.

- FFC proactively handle these faults.
  - Guarantee: no congestion when **#faults ≤ k**.
  - Efficiently computable with low throughput overhead in practice.

FFC

High risk of congestion

Heavy network over-provisioning