FatTire: Declarative Fault Tolerance for SDN

Mark Reitblatt (Cornell)
Marco Canini (TU Berlin → UC Louvain)
Arjun Guha (Cornell → UMass Amherst)
Nate Foster (Cornell)

In a Perfect World...

But in Reality...

Fault-Tolerance Mechanisms

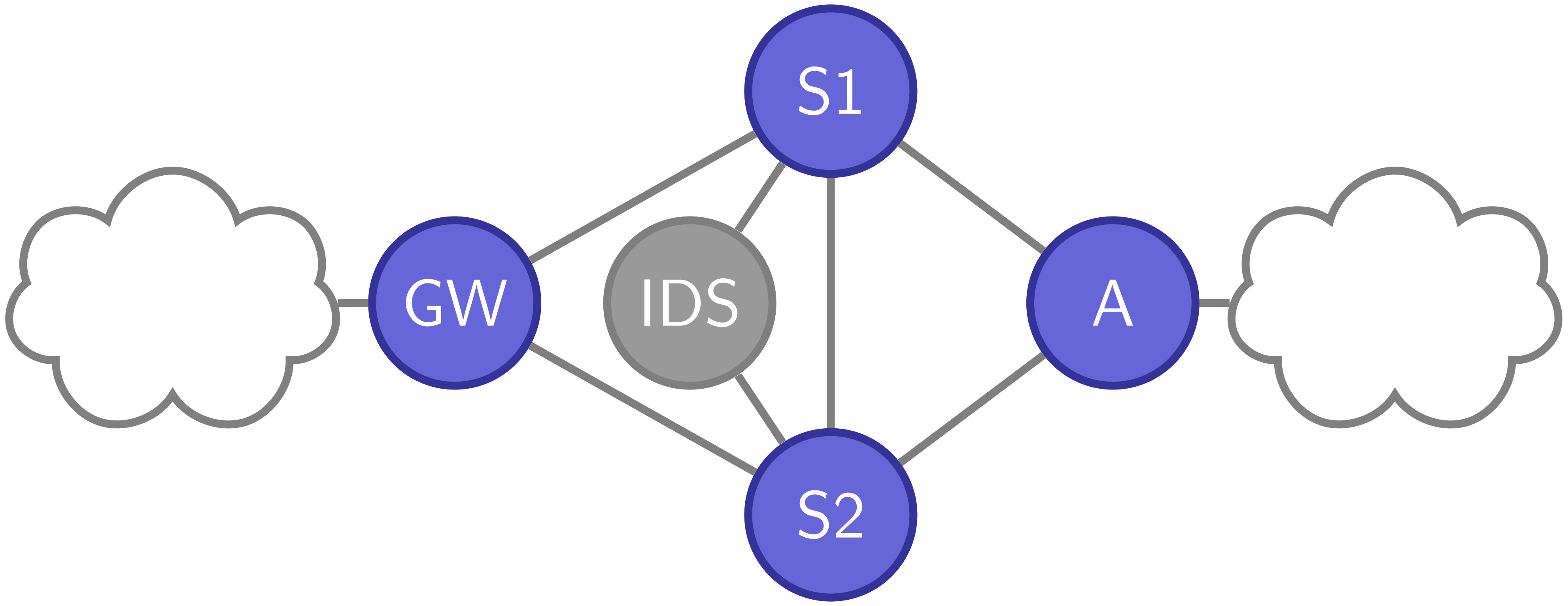
Traditional Networks

- MPLS local path protection
- Global path protection
- IEEE 802.1ag
- and others...

Software-Defined Networks

- Controller reacts to failures
- Fast failover group actions (OpenFlow 1.1+)

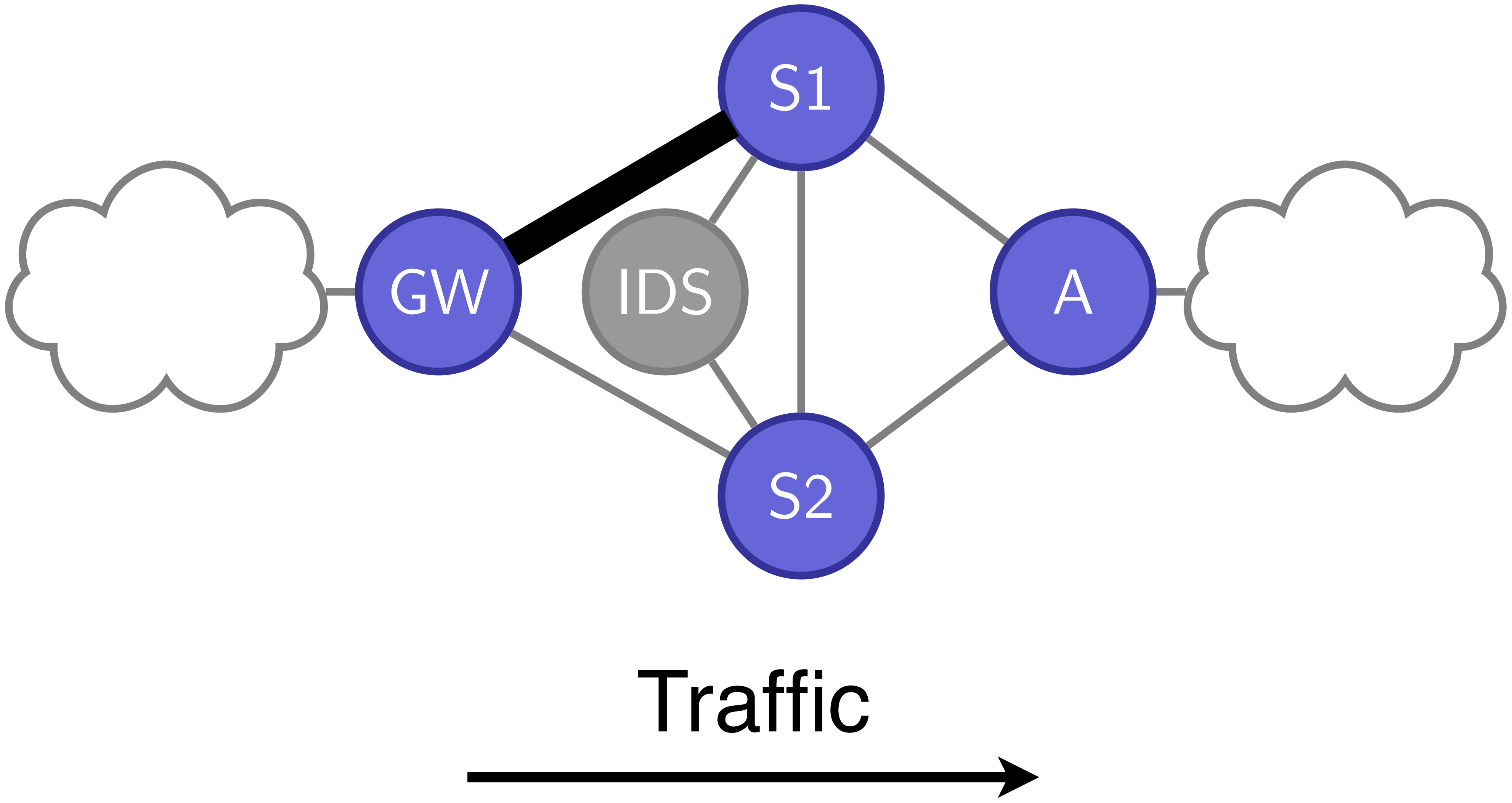
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



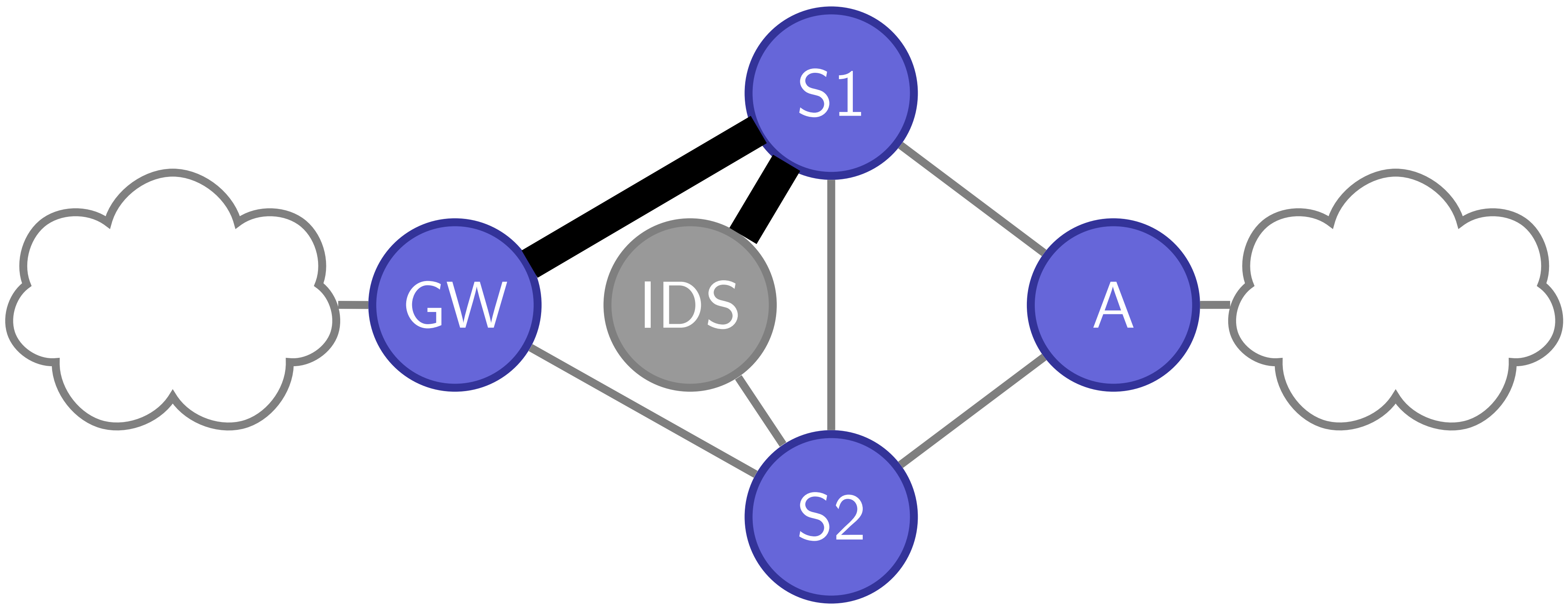
Traffic

$\xrightarrow{\hspace{6em}}$

- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant

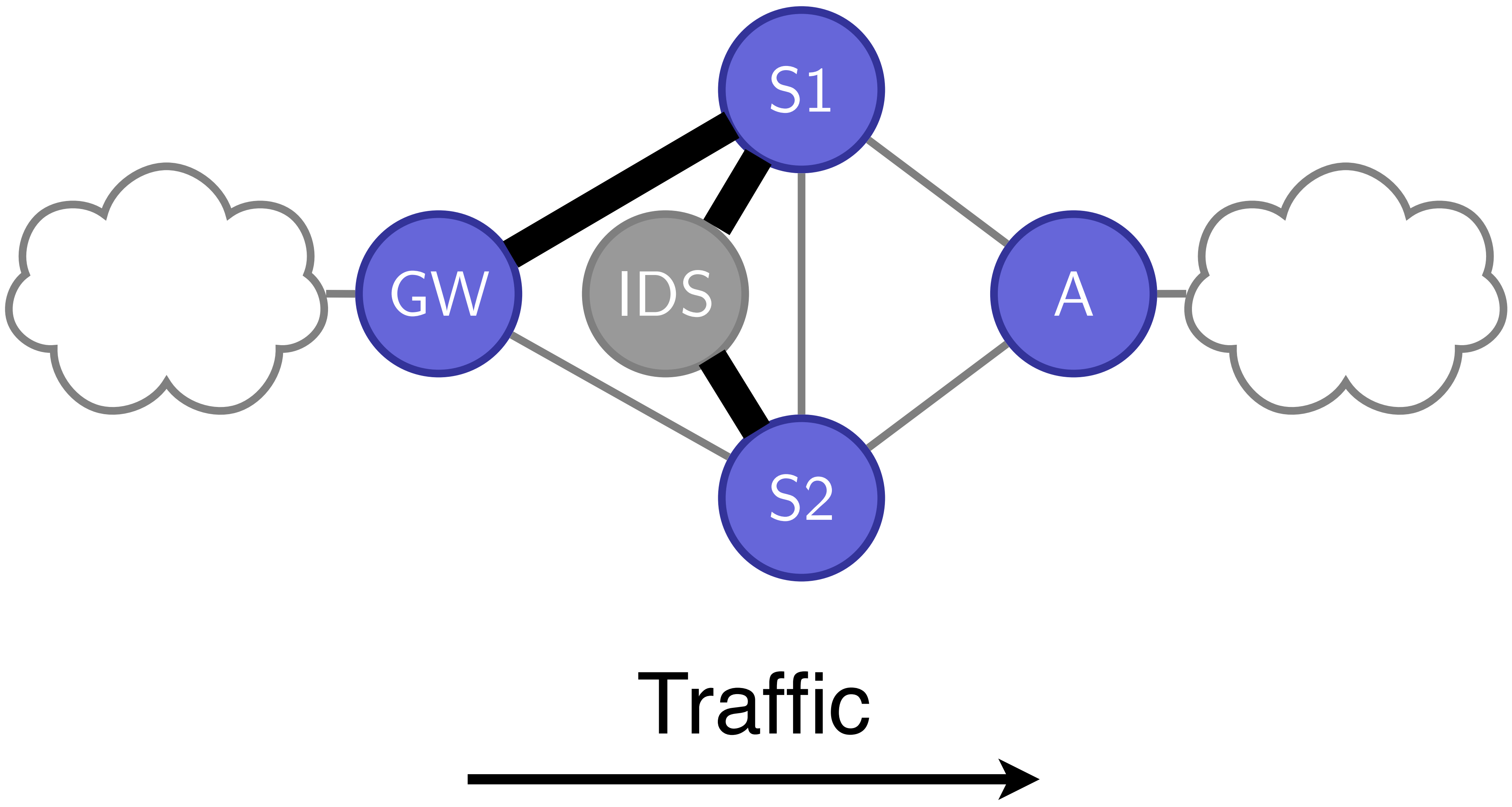


- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant

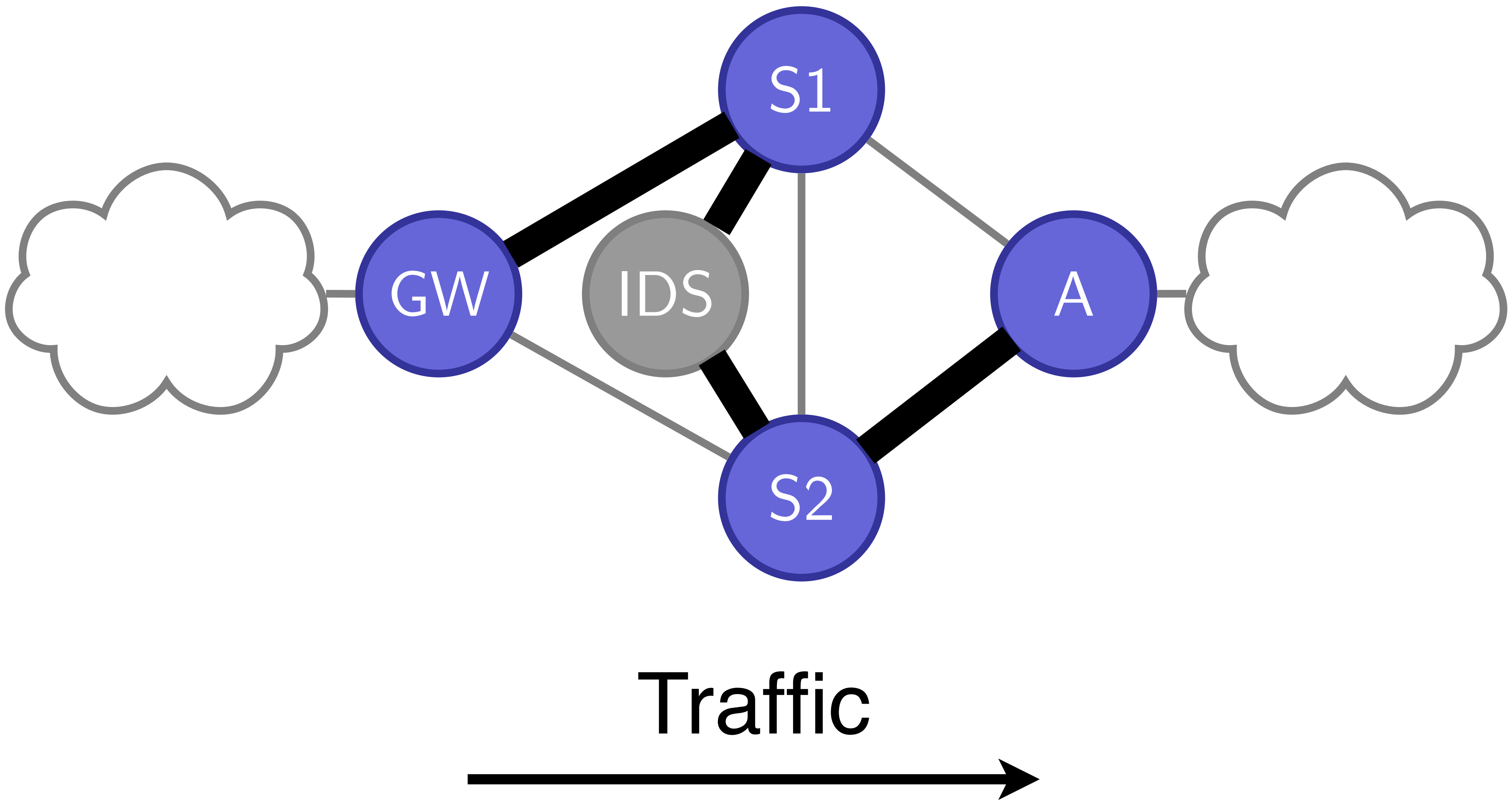


Traffic

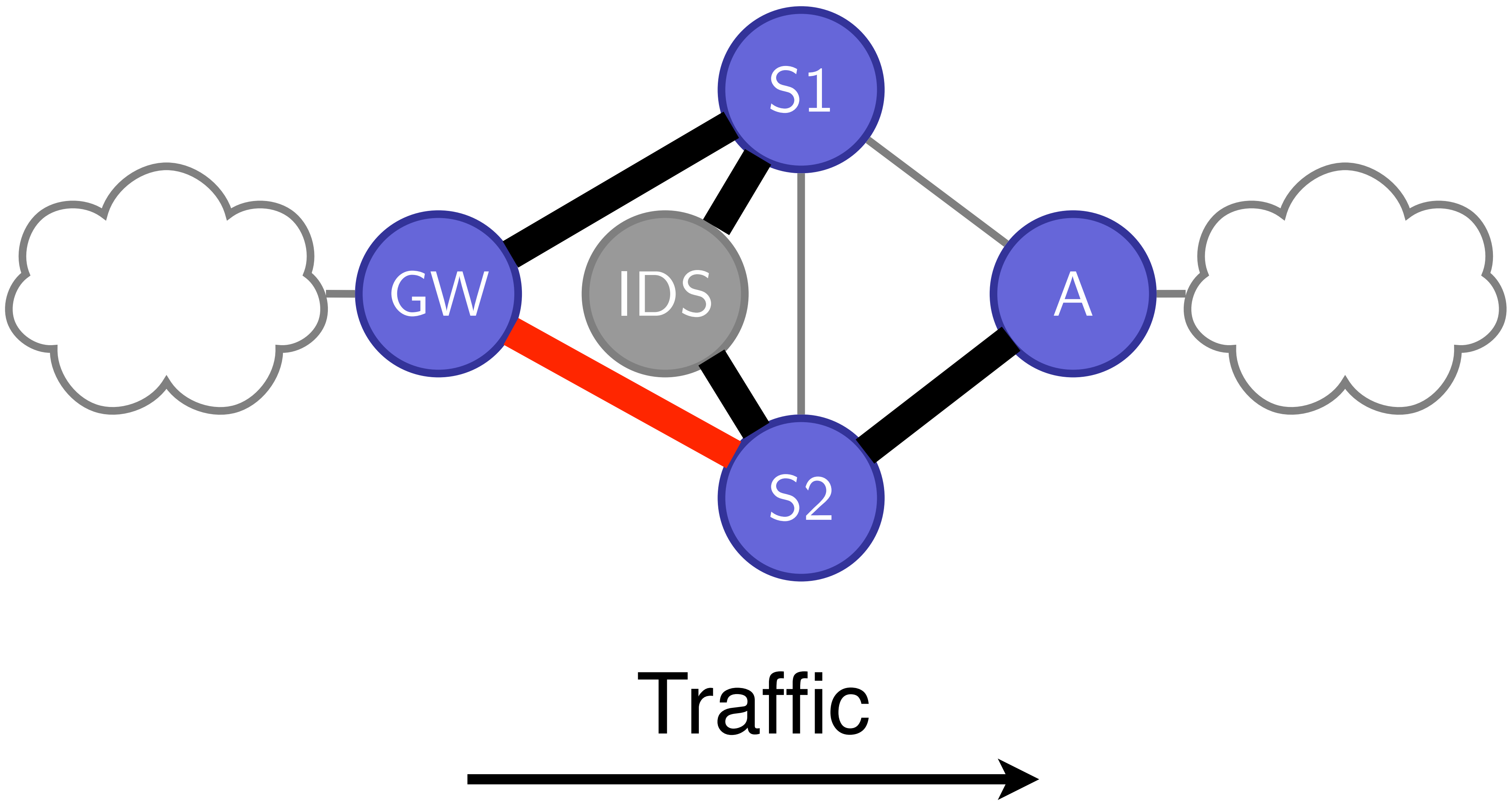
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



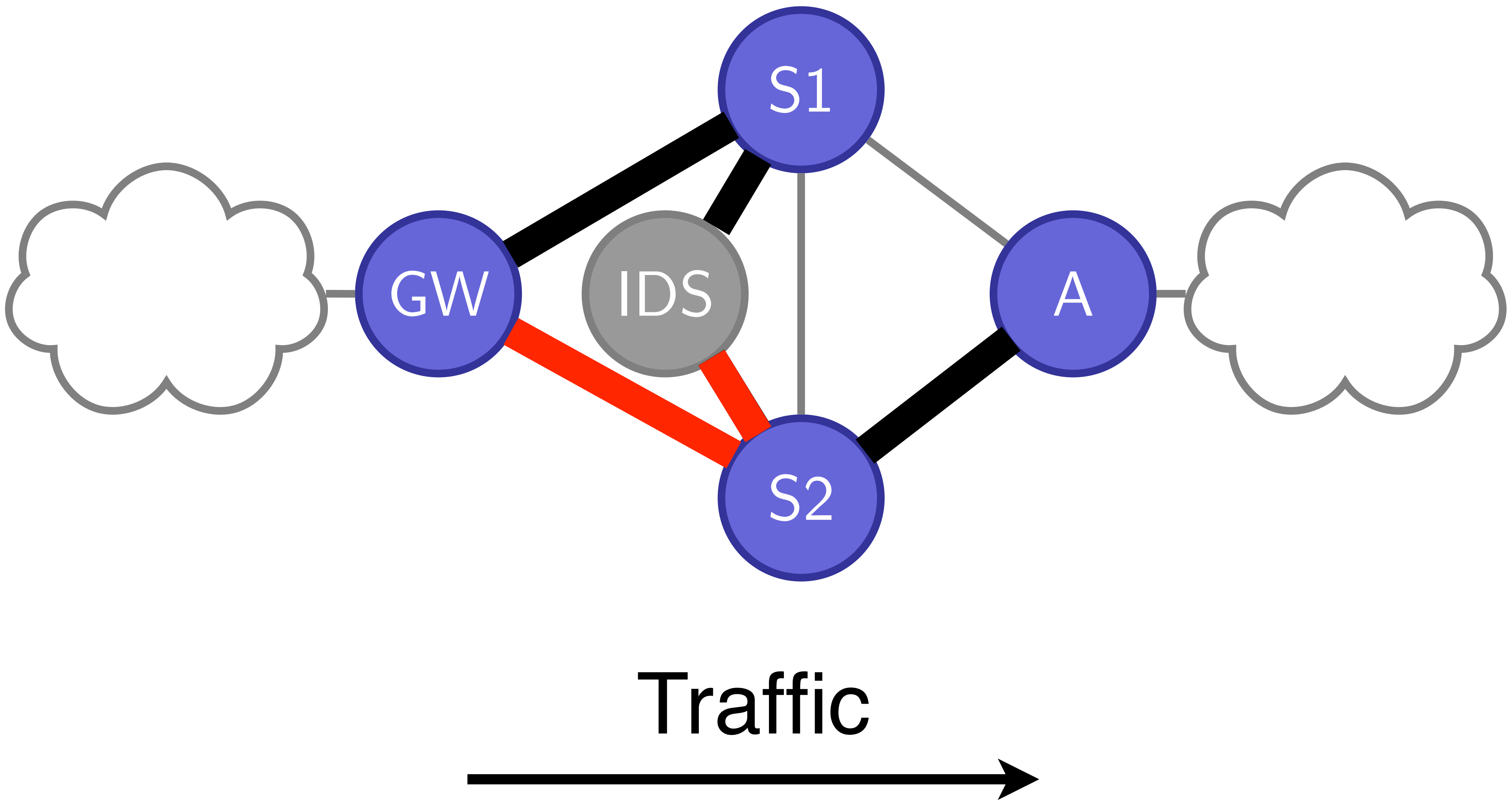
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



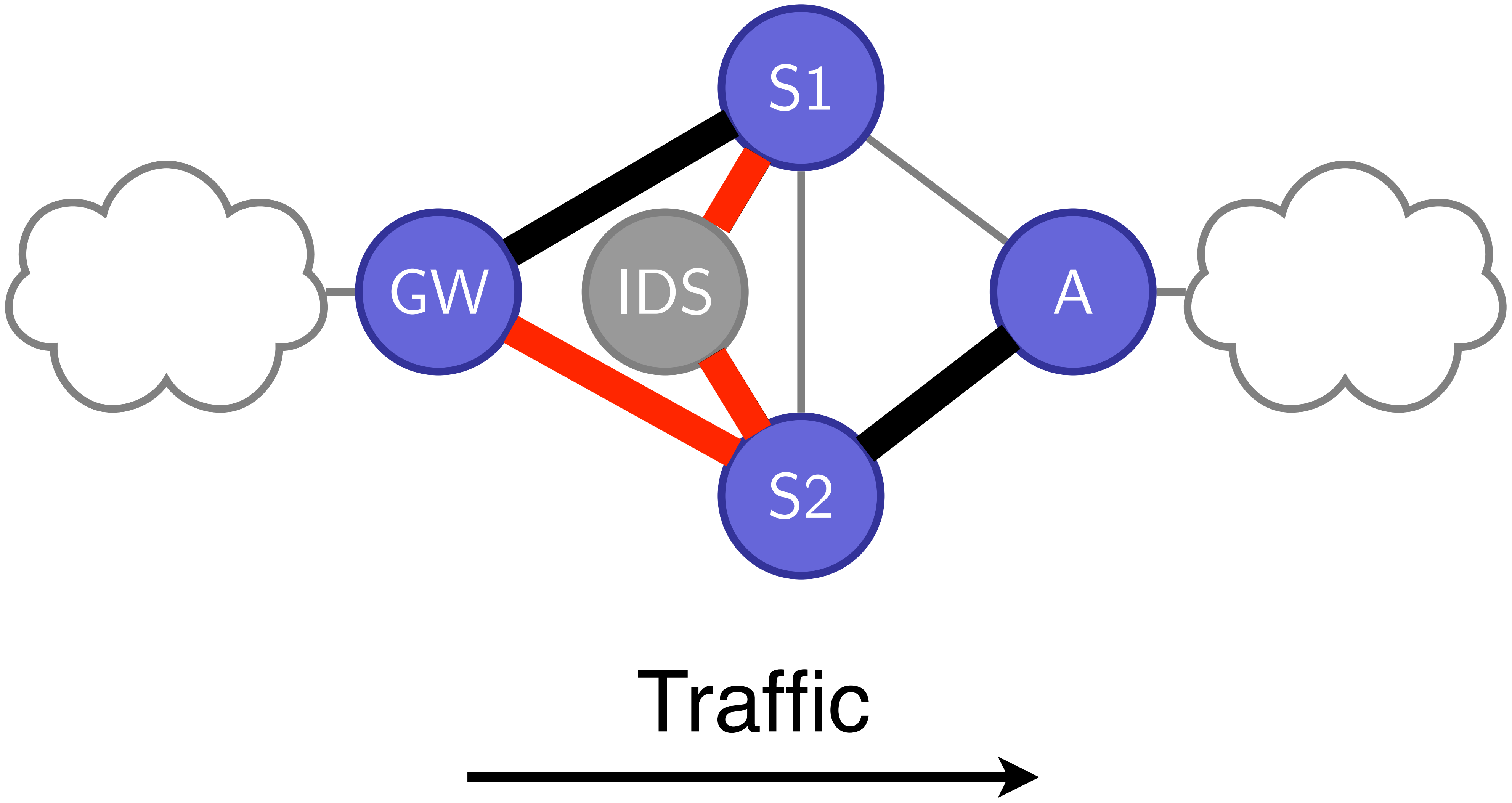
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



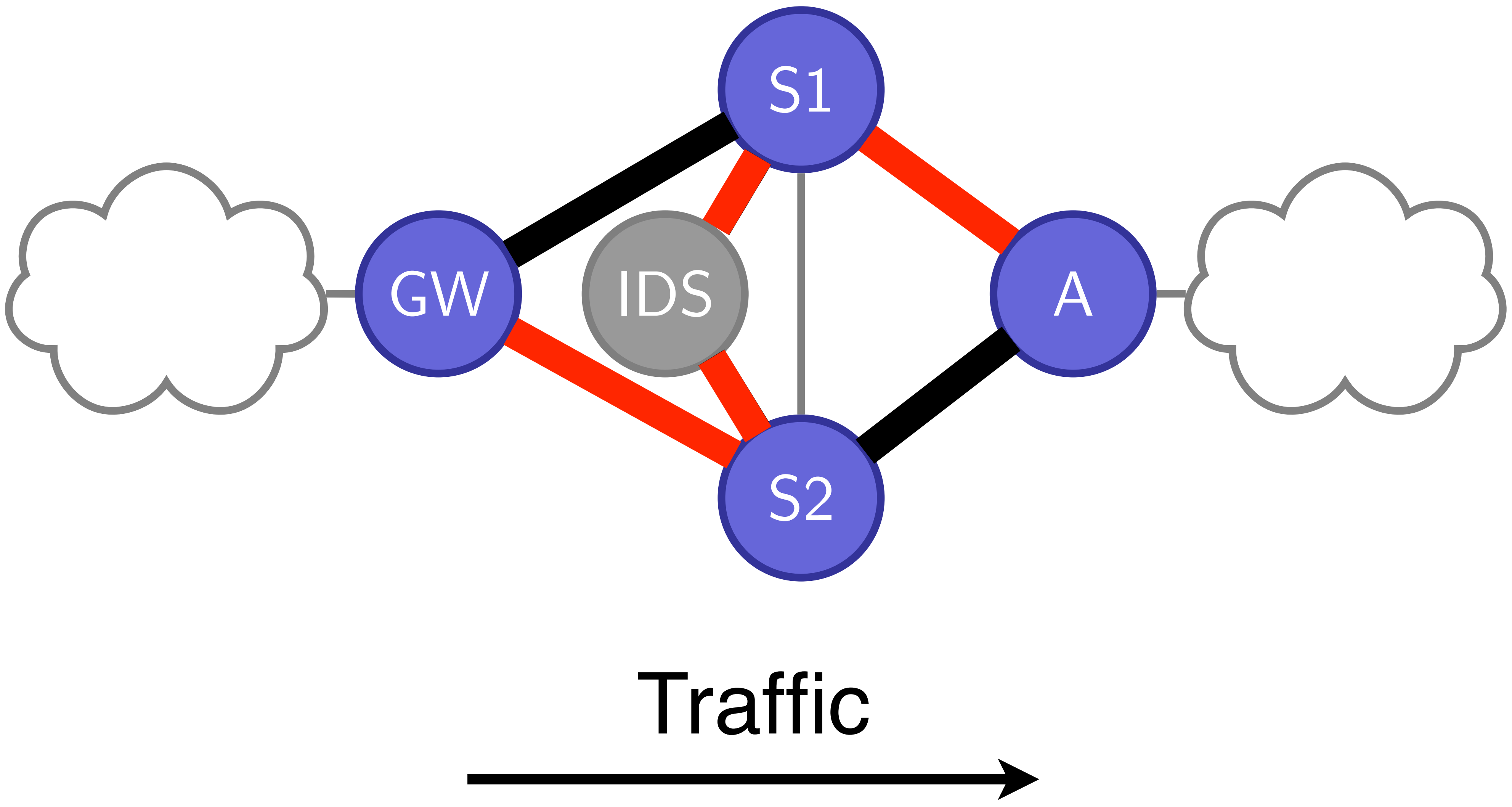
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



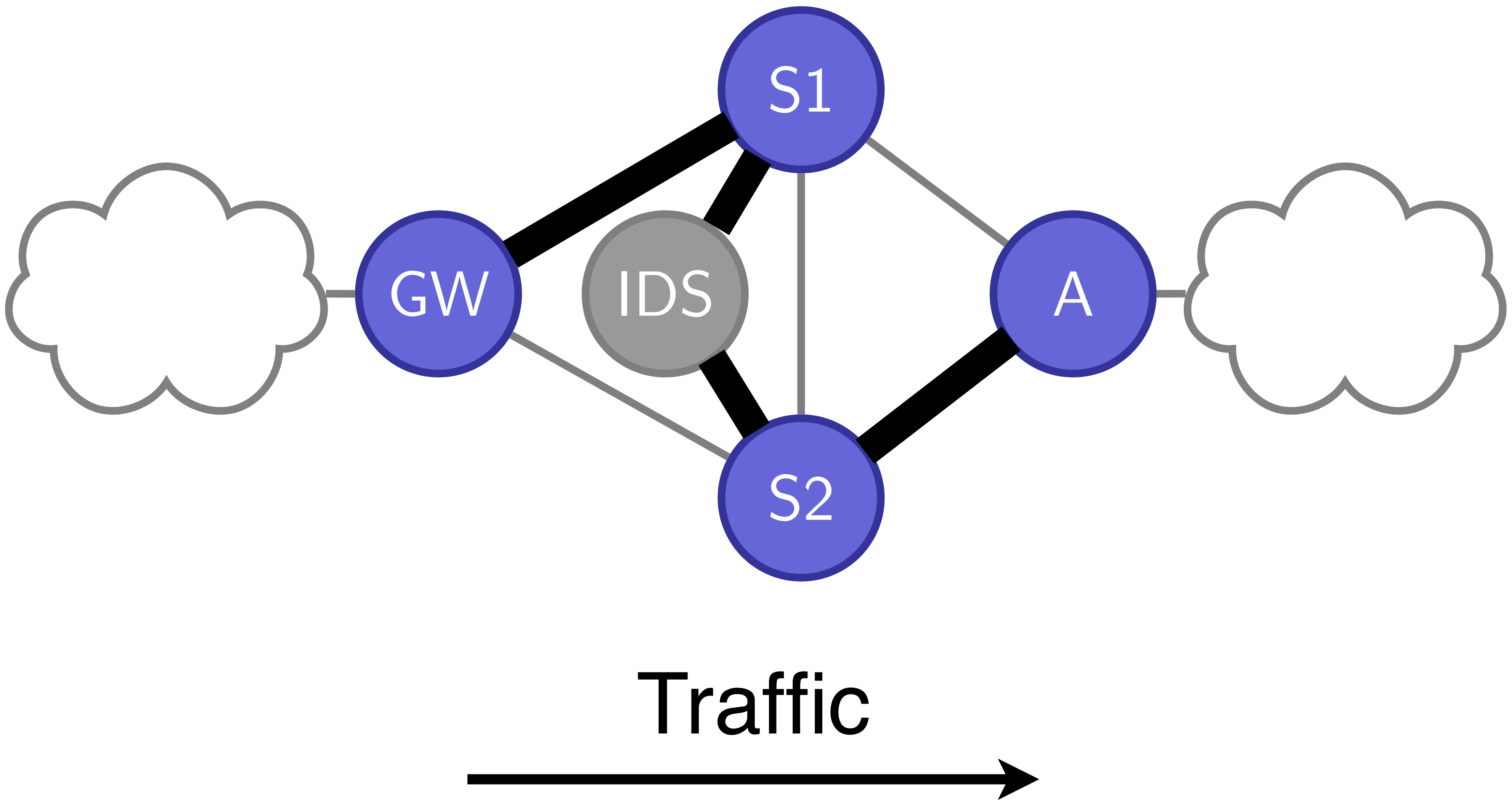
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



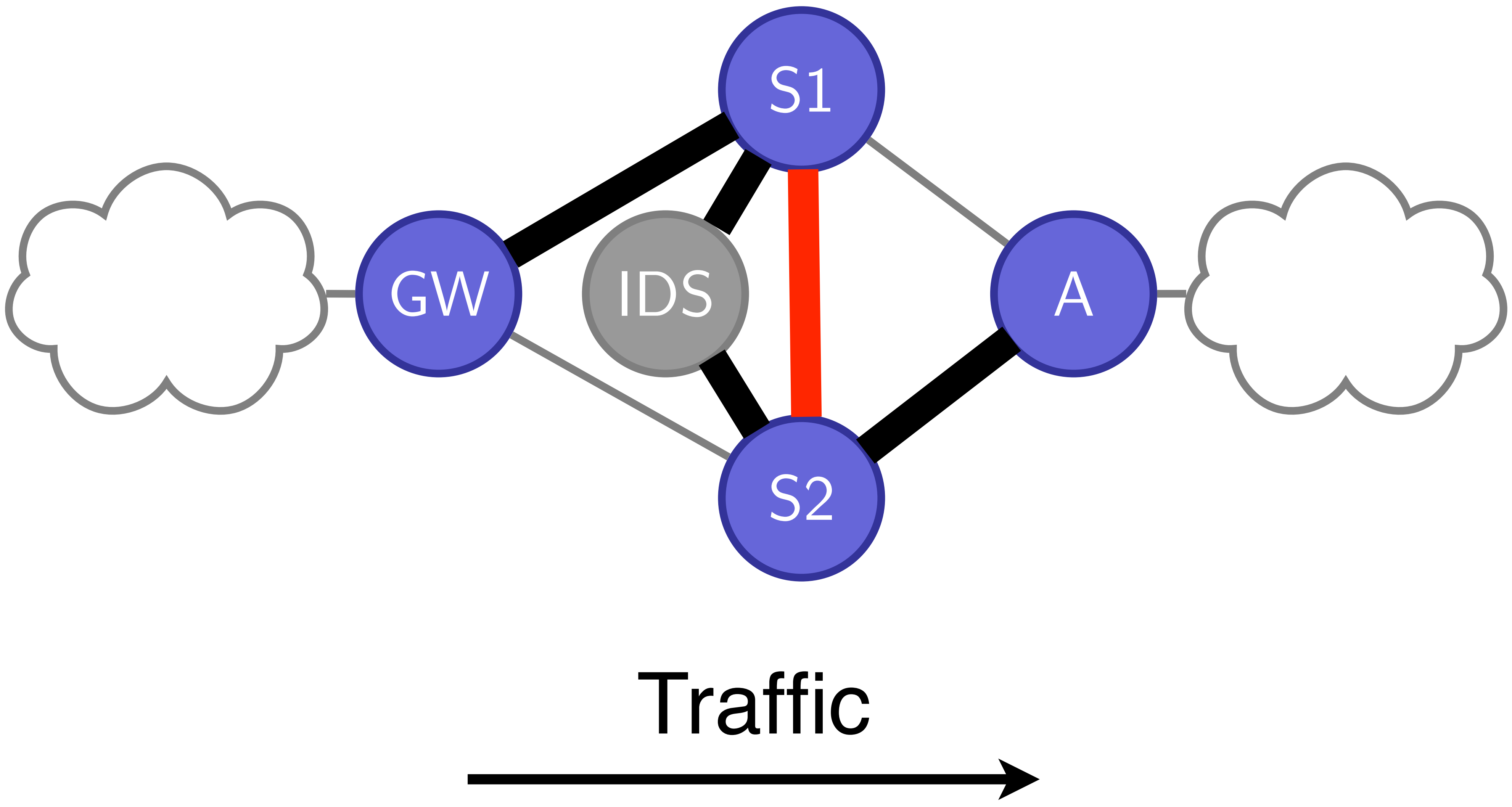
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



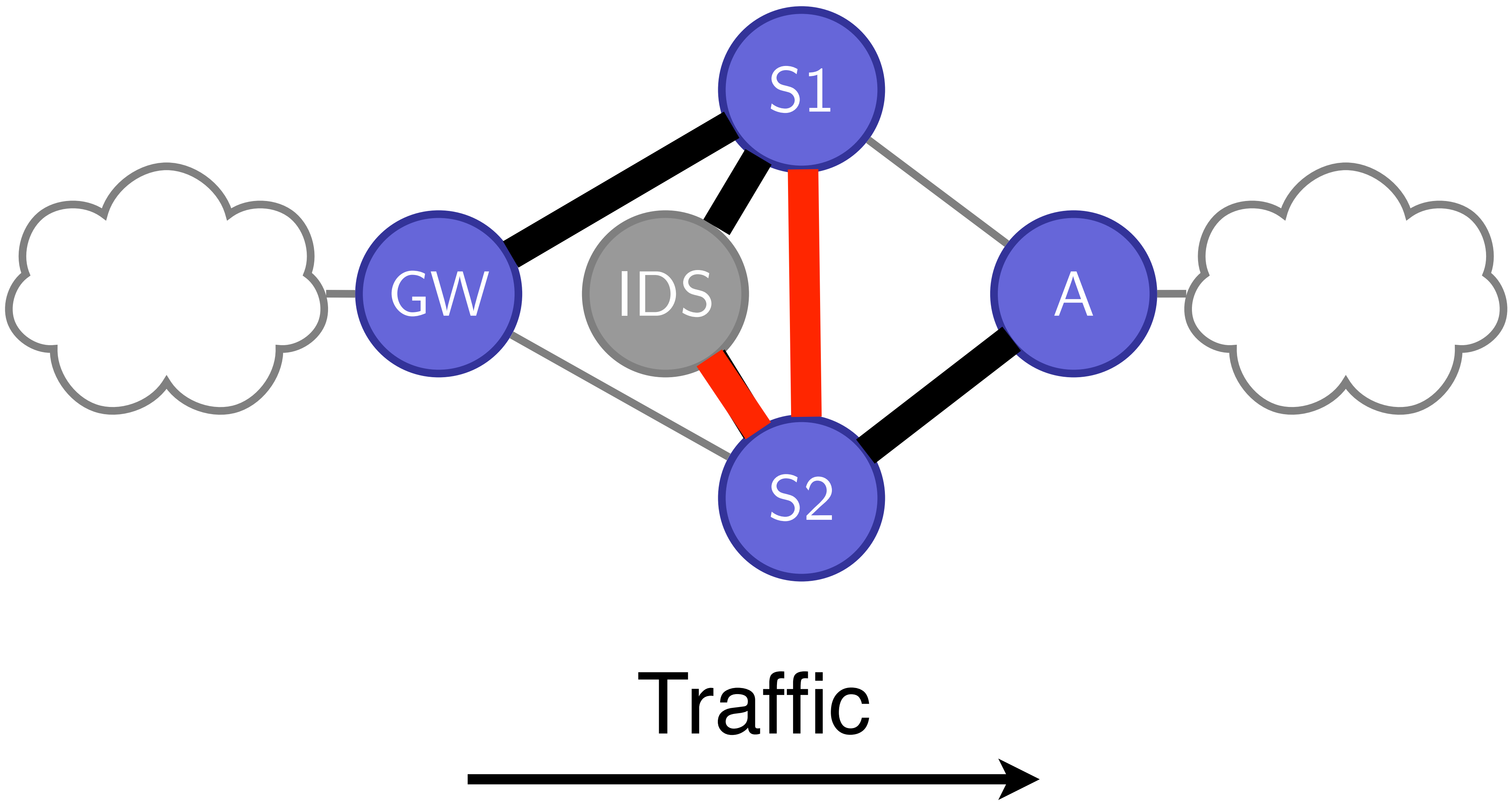
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



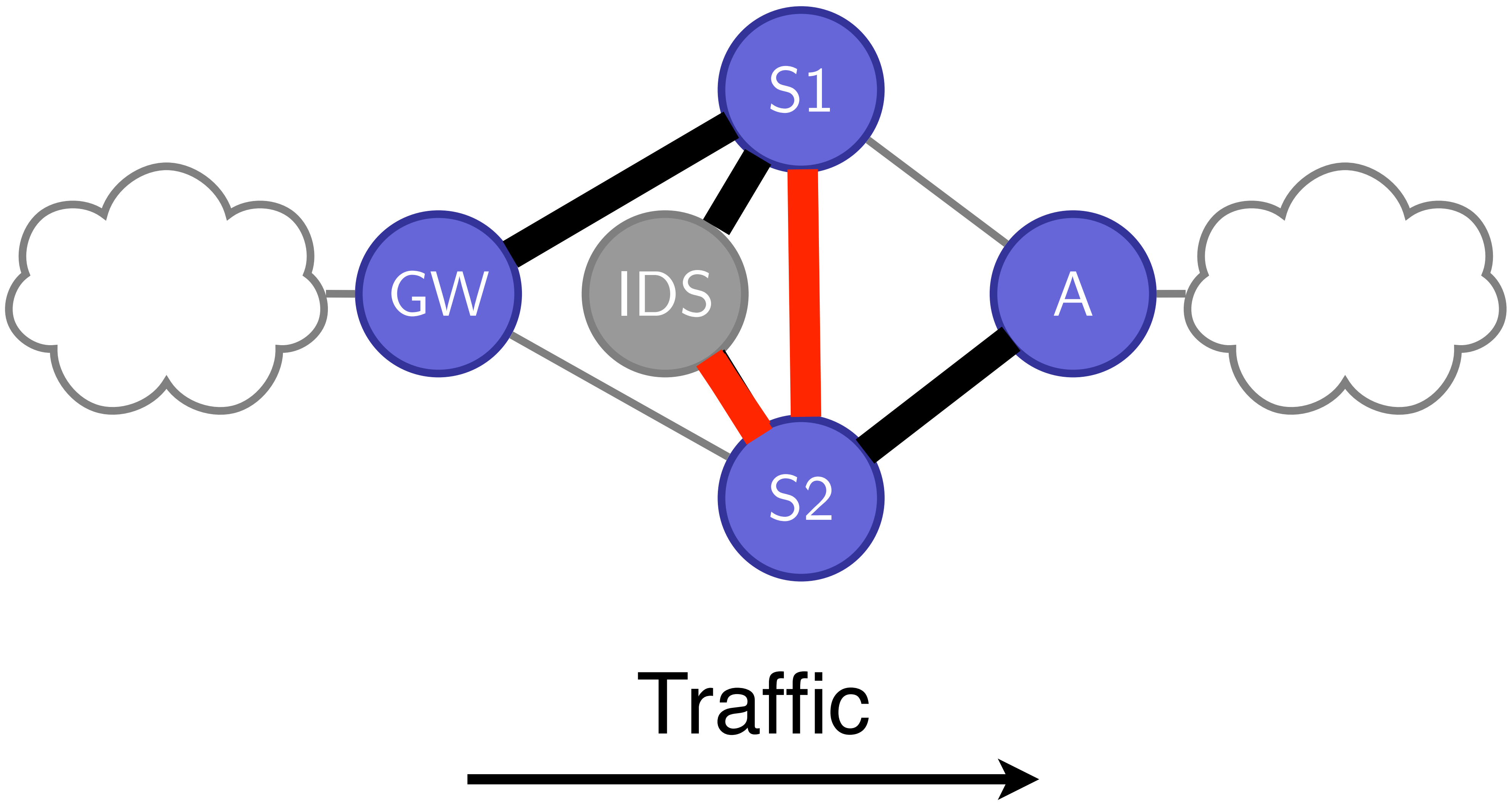
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



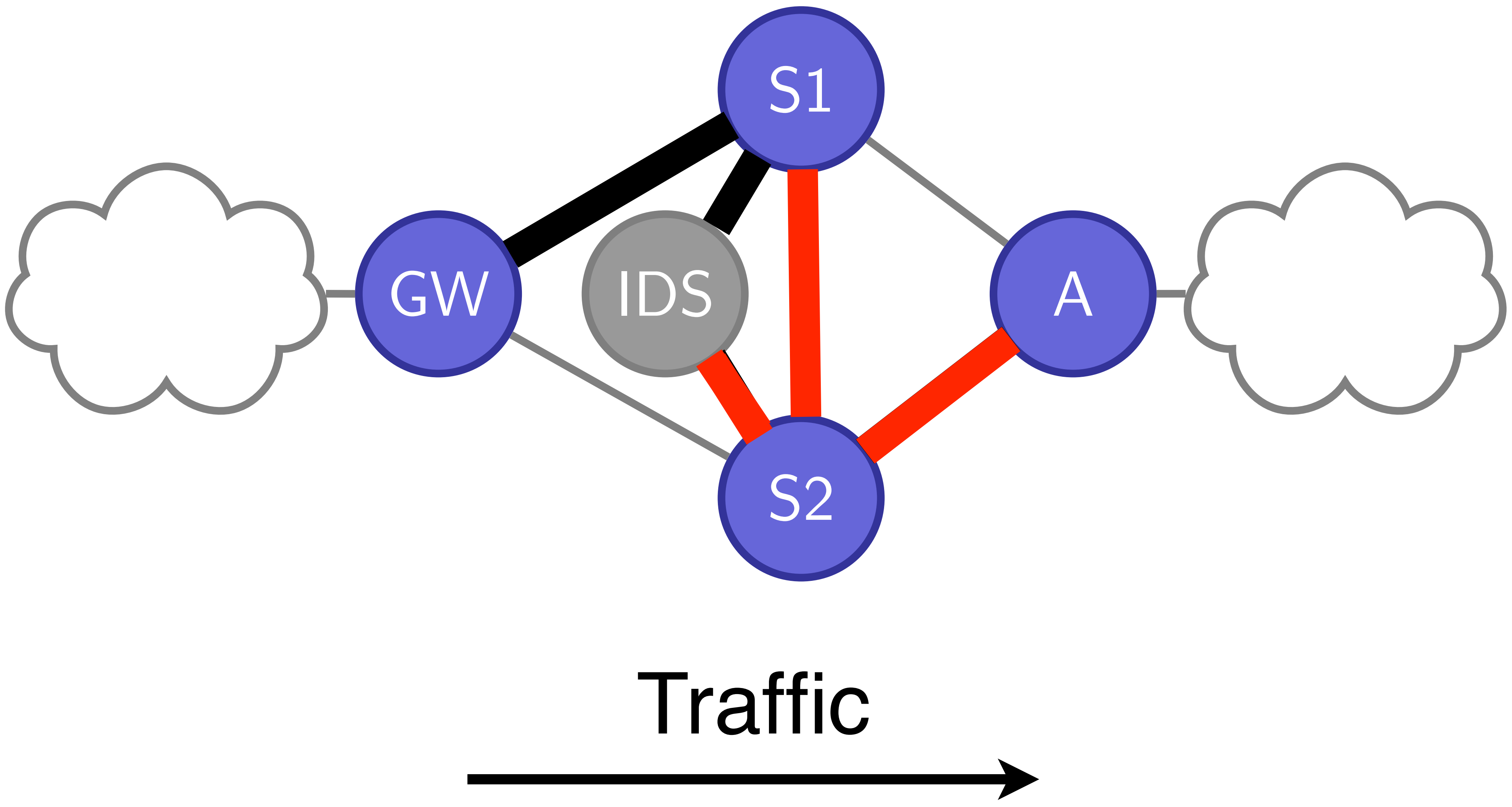
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



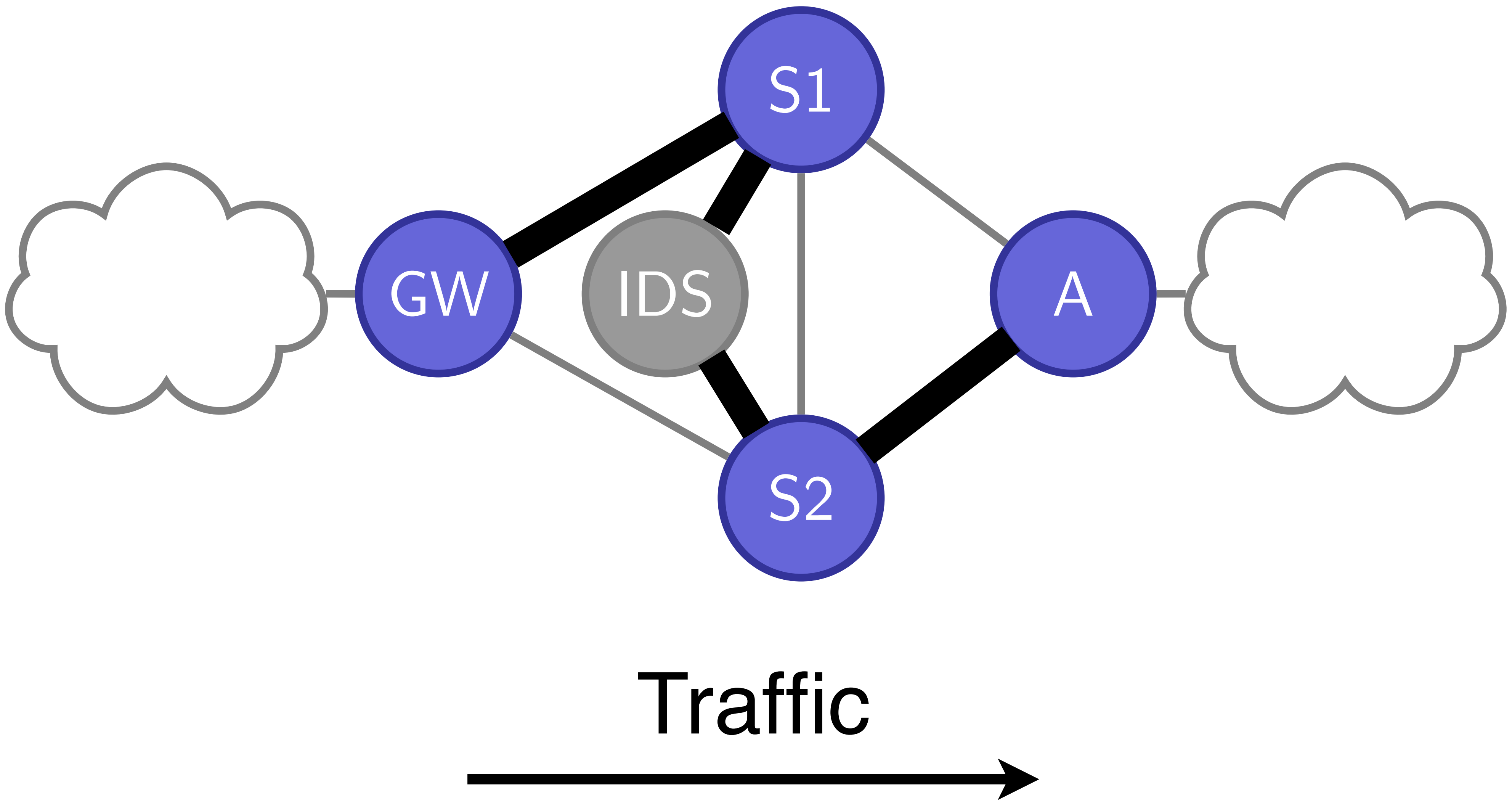
- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1-link fault tolerant



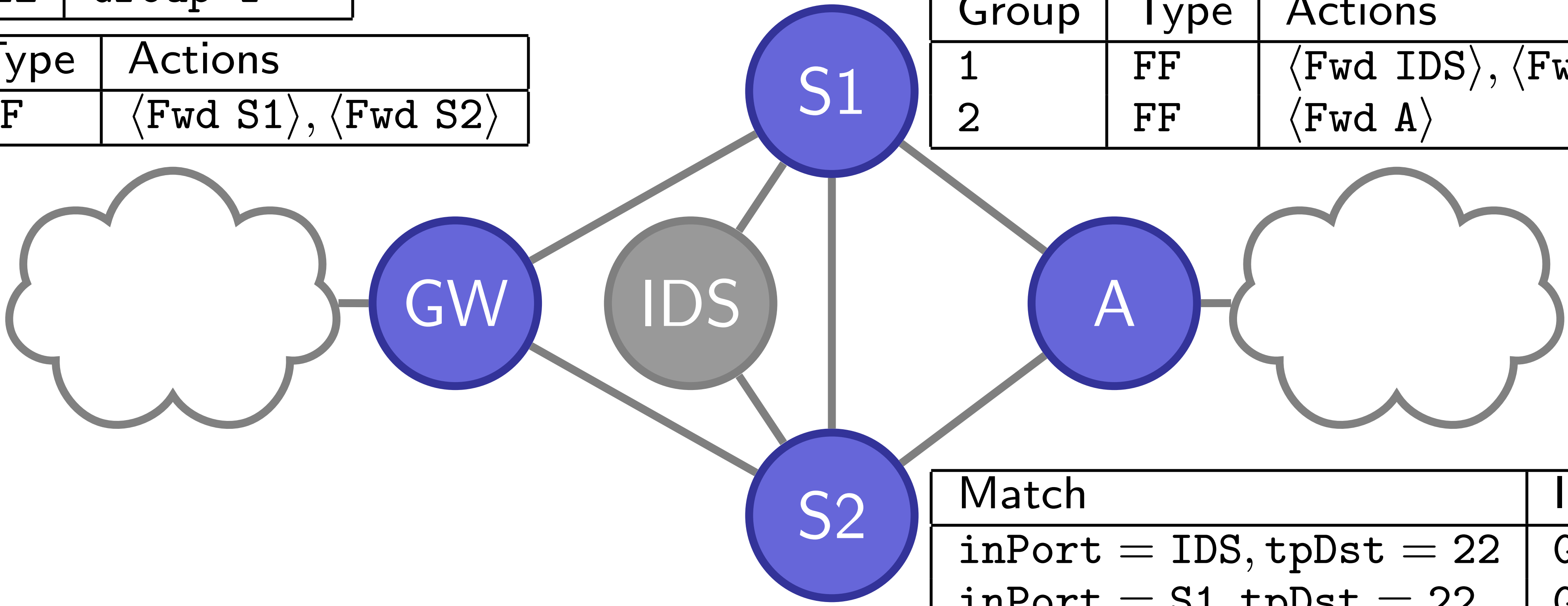
OpenFlow Fast Failover

Match	Instructions
tpDst = 22	Group 1

Group	Type	Actions
1	FF	⟨Fwd S1⟩, ⟨Fwd S2⟩

Match	Instructions
inPort = GW, tpDst = 22	Group 1
inPort = IDS, tpDst = 22	Group 2
inPort = S2, tpDst = 22	Group 2

Group	Type	Actions
1	FF	⟨Fwd IDS⟩, ⟨Fwd S2⟩
2	FF	⟨Fwd A⟩



Match	Instructions
inPort = IDS, tpDst = 22	Group 1
inPort = S1, tpDst = 22	Group 2
inPort = GW, tpDst = 22	Group 2

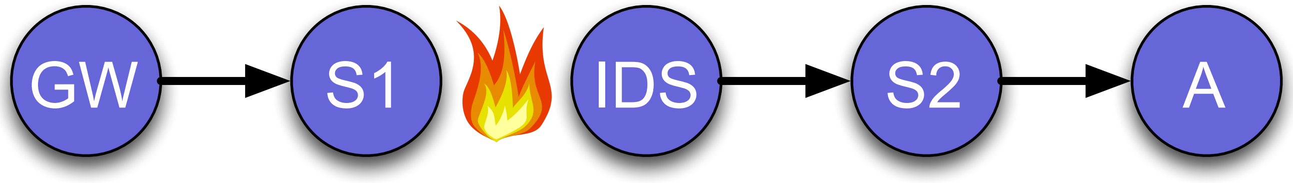
Group	Type	Actions
1	FF	⟨Fwd A⟩, ⟨Fwd S1⟩
2	FF	⟨Fwd IDS⟩

Why not Frenetic?

- Frenetic provides a declarative language for expressing forwarding policies...
- ... in terms of hop-by-hop forwarding steps
- Example:

```
    (GW  → S1) + (S1 → IDS)
  + (IDS → S2) + (S2 → A)
```

- What to do if next hop fails?



Our Approach: FatTire

"Fault Tolerating Regular Expressions"

Key Ingredients:

- Hop-by-hop forwarding → paths
- Deterministic → non-deterministic
- Explicit fault-tolerance constructs

Challenges:

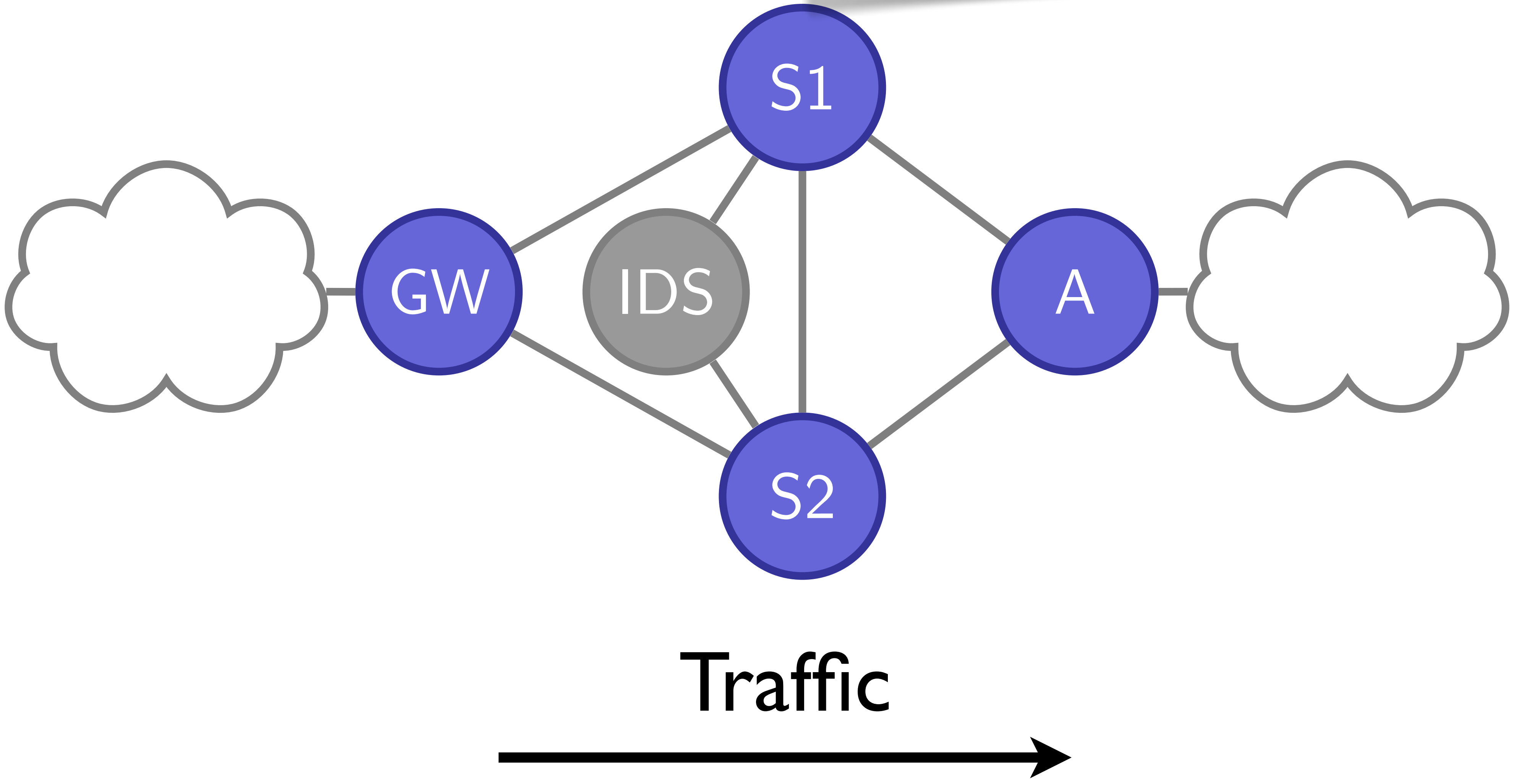
- FatTire programs may specify overlapping paths
- OpenFlow tables are deterministic
- Global analysis to provide fault-tolerance guarantees

- Connectivity from GW to A
- SSH traffic traverses IDS
- SSH is 1 link fault tolerant

```
   (All → [GW * A])

∩( SSH → [* IDS *]   )
 ( ∪ ¬SSH → [*]      )

∩( SSH → [*] with 1  )
 ( ∪ ¬SSH → [*]      )
```



Programming in FatTire

Write programs in terms of regular expressions on forwarding paths

- `[GW * A]`
- `[GW (S1 | S2) A]`

Use annotations to specify desired fault tolerance

- `SSH → [*] with 1`
- `¬SSH → [*] = ¬SSH → [*] with 0`

Programming in FatTire

Can combine policies with intersection and union:

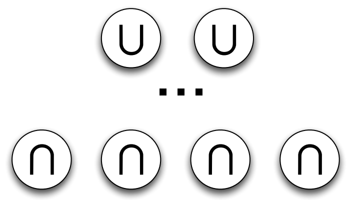
- Intersection adds restrictions on paths

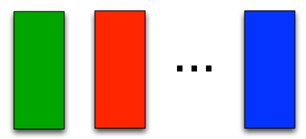
```
  (All → [GW * A]) ∩ (SSH → [*] with 1)
= SSH → [GW * A] with 1
```

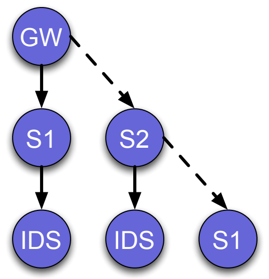
- Union loosens restrictions on paths

```
  (All → [GW S1 A]) ∪ (All → [GW S2 A])
= All → [GW (S1 | S2) A]
```

FatTire Compiler







$((GW \rightarrow S1) \oplus (GW \rightarrow S2))$

$+ ((S1 \rightarrow IDS) \oplus (S2 \rightarrow IDS))$

1. Normalize into Disjunctive Normal Form
2. Partition into traffic equivalence classes
3. Compute fault-tolerant forwarding graph
4. Output hop-by-hop Frenetic policy and compile to OpenFlow rules

Implementation

- Full working prototype implemented in OCaml
- Based on an extension of the Frenetic controller with support for OpenFlow 1.3
- Tested on CPqD 1.3 software switch
- See paper for preliminary experimental evaluation using Mininet
- Code available from
  https://github.com/frenetic-lang/fattire
  under an open-source license

Future Work

- Extend to handle quantitative path properties
  - Bandwidth
  - Latency
- Provide first-class support for other topology changes such as switch failures
- Investigate applications of non-deterministic network programs
- Investigate other recovery mechanisms

Thank You

FatTire Team:

Mark Reitblatt

Marco Canini

Arjun Guha

Nate Foster

frenetic >>

Papers, source code, examples, tutorials, etc.

http://frenetic-lang.org

Backup Slides

Update consistency

- Semantics of failure recovery => per-packet consistency

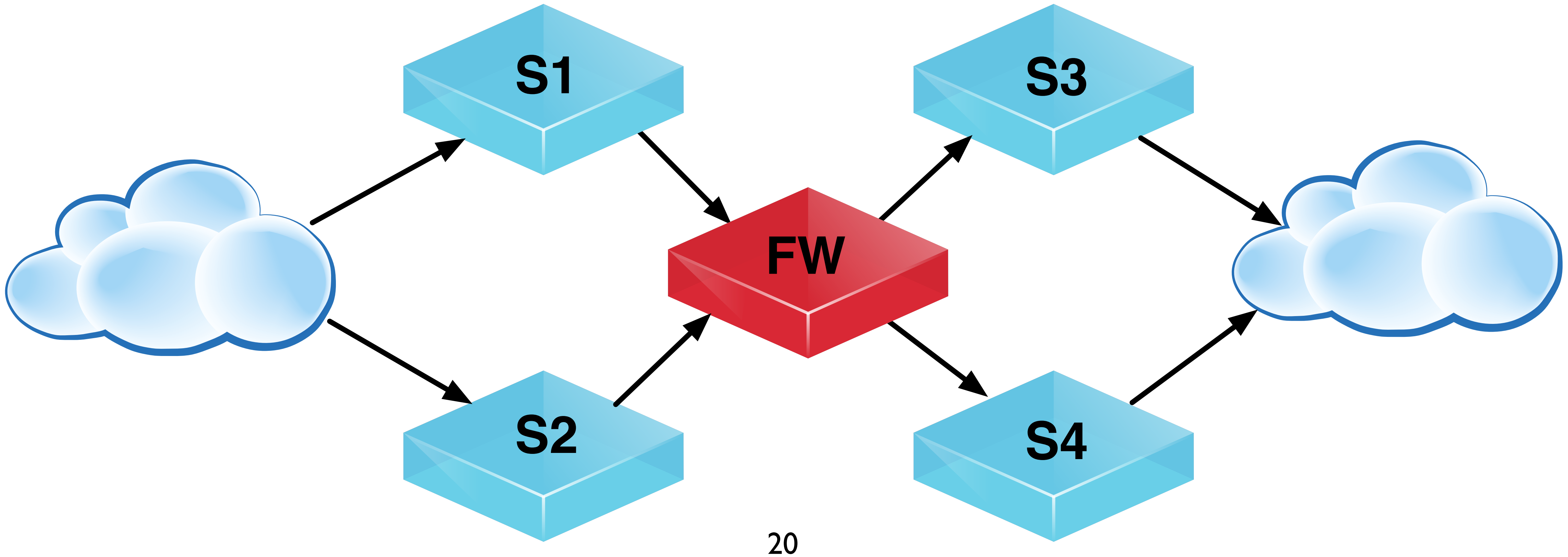
Regular Expression Derivatives

Path Expressions as verification spec

- Dual use as verification specification?

Interaction of paths

```
  All → [S1.FW.S3]
∪ ALL → [S2.FW.S4]
```


S1
S2
FW
S3
S4