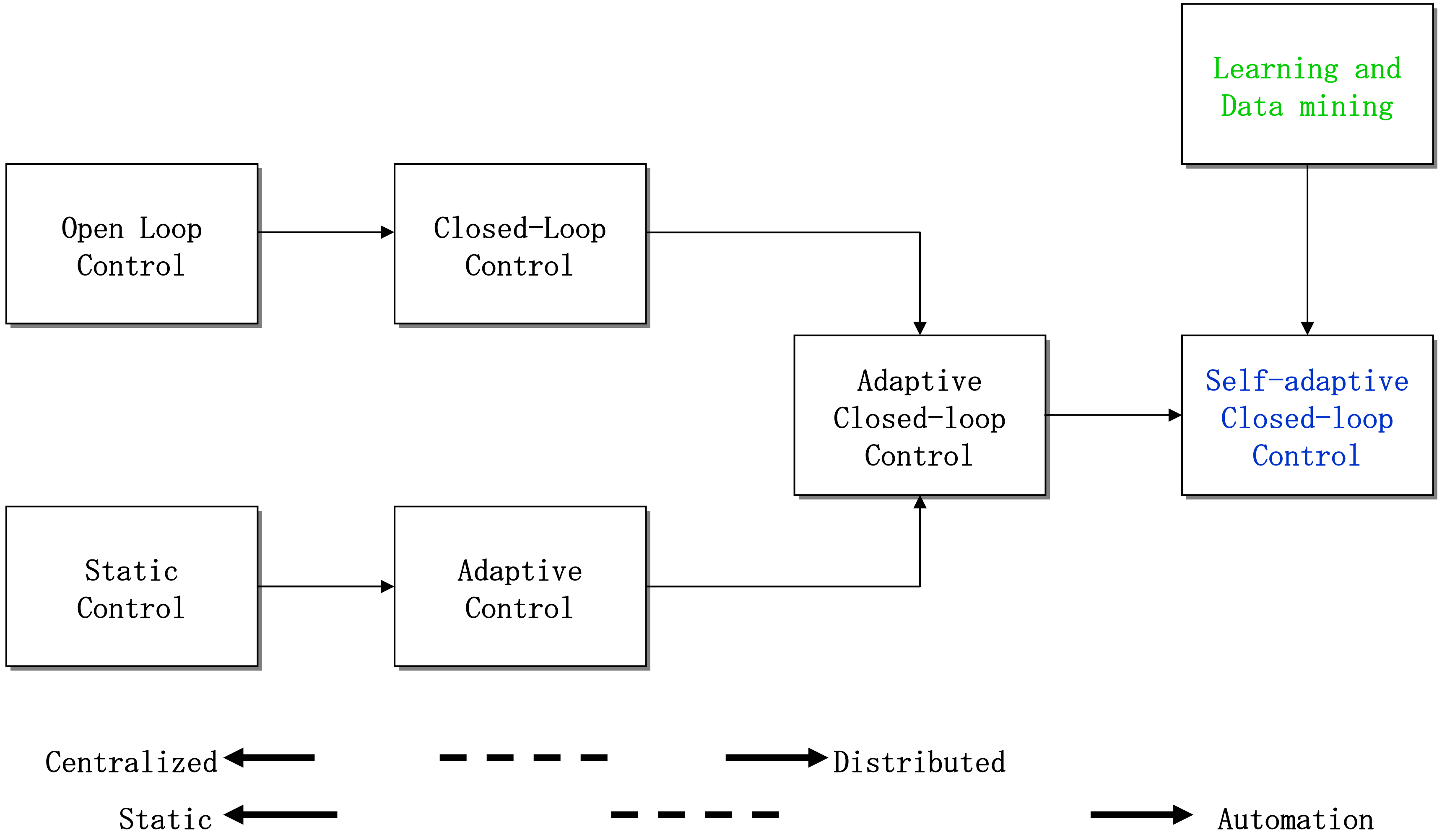
Performance evaluation of learning methods for self-adaptive resilient networks

D.Papadimitriou (ALB), W.Tavernier (IBBT), B.Puype (IBBT), and D.Colle (IBBT)

ECODE Project Website
<http://www.ecode-project.eu>

From *static open-loop control to self-adaptive closed-loop distributed control*



Motivation: avoid need to iteratively re-design systems as they should adapt to their environment and running conditions

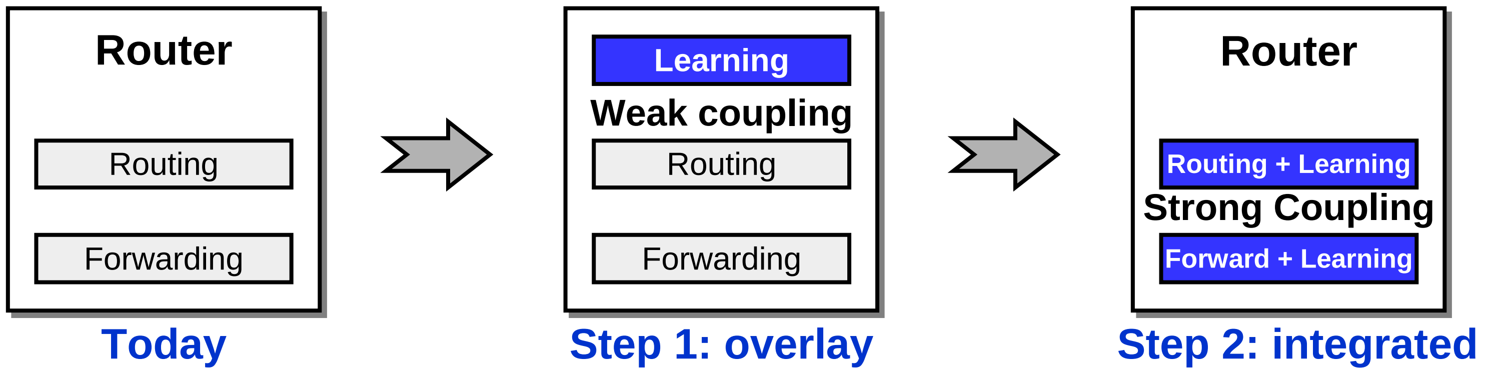
Roles of learning in self-adaptive closed-loop distributed control

Roles of learning

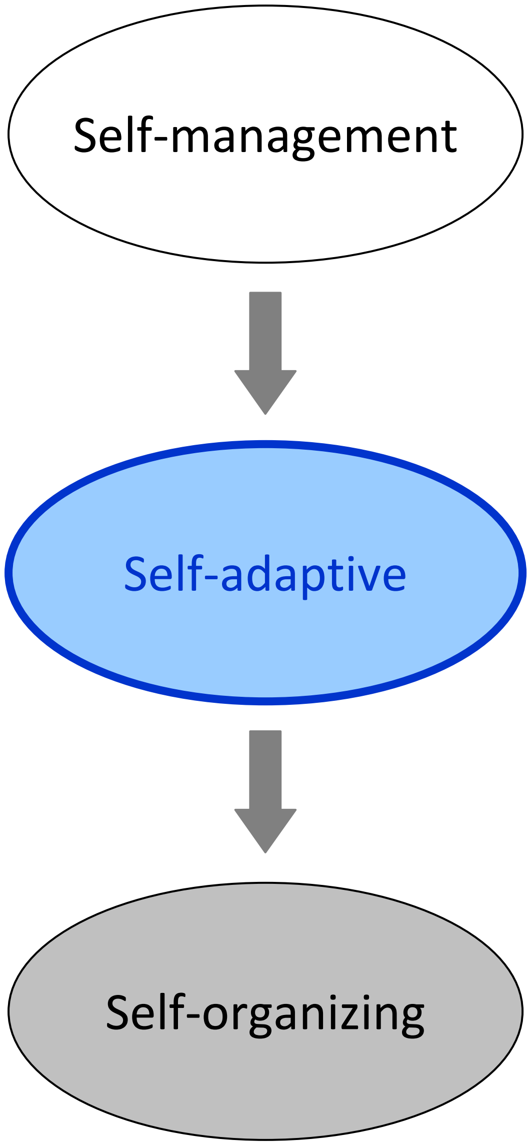
- **WHAT/WHY**: **Diagnose** internal state, own activity, and network environment over time (detect -> identify -> analyze)
- **HOW**: **Adapt** decisions and **Tune** actions in an automatic, cost-effective and timely way
- **WHEN**: **Determine** when to operate autonomously and when to cooperate

Augment network system's control with learning component

- Example: augment control system with distributed learning component using online, distributed machine learning techniques



Positioning



Self management: capability to operate and maintain systems and devices depending on the current system parameters

Properties: Unified framework and global coordination (and re-evaluation)

Technique: e.g. policy-based management

Self-adaptive: ability of the system's components to adapt to changing environmental and running conditions over time

Properties: Distributed, modular, and local coordination (and re-evaluation)

Technique: e.g. learning

Self-organize: structure and functionality at the global level emerge solely from numerous interactions among lower-level components without any external or centralized control

Technique: e.g. biology, neuro-science

Examples of Applications

- **Routing**
  - **Packet loss reduction during fast re-routing**
  - **Shared Risk Link Group (SRLG) detection and identification from mining of link state protocol data**
  - Path-vector routing: mitigation of uninformed path exploration events in BGP inter-domain routing
  - Traffic-informed routing path servers: path (segments) quality monitoring and ranking
- **Monitoring**
  - Network performance monitoring by combining adaptive passive and active measurements
- **Accountability**
  - Traffic flow responsivity to congestion marking / notification
- **Security**
  - Cooperative traffic anomaly detection
  - Intrusion Detection System (IDS)

- Experimental results published at scientific conferences and journals: <**http://www.ecode-project.eu/pmwiki/pmwiki.php/Ecode/Publications**>

Problem statement

Forwarding table of router A

thousands of entries

Dest	Mask	Gateway	I/f
e	xxxxxx	b	1
d	xxxxxx	b	…
g	xxxxxx	b	1
…	…	…	…
Default	xxxxxx	f	2

>1 sec

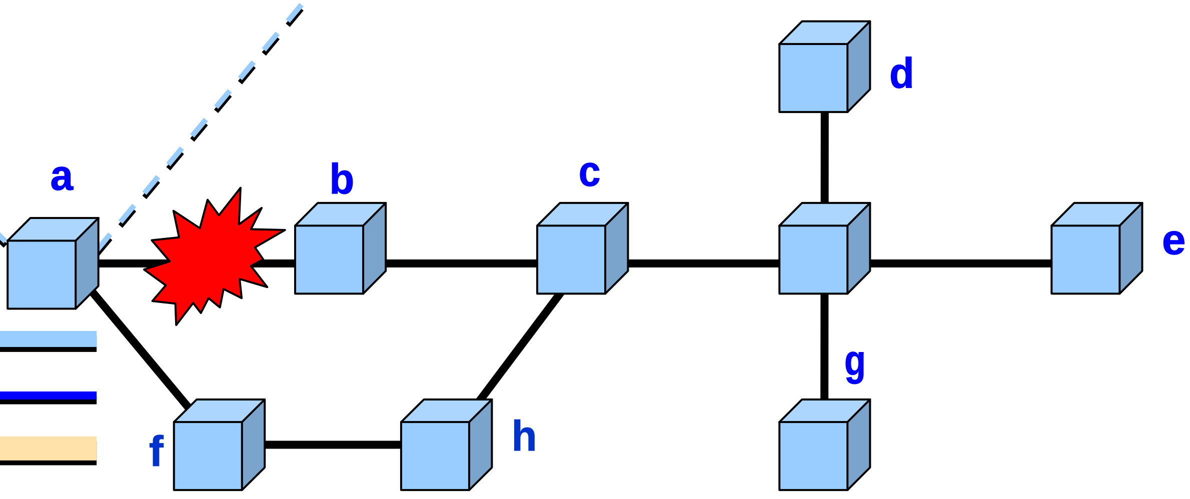
New forwarding table

Dest.	Mask	Gateway	I/f
e	xxxxxx	f	2
d	xxxxxx	f	…
g	xxxxxx	f	2
…	…	…	…
Default	xxxxxx	f	2

.5 s
.2 s
.8 s

Traffic flows:

a->d: 200 Mbps	▬
a->e: 50 Mbps	▬
a->g: 500 Mbps	▬



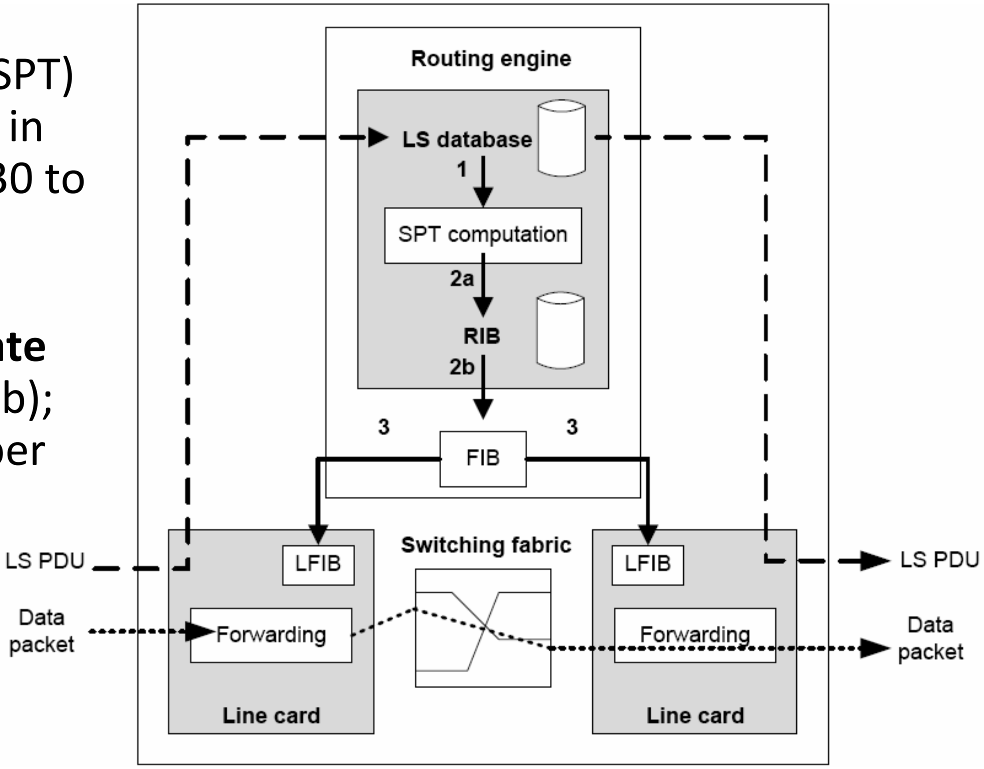
RESULT with traffic uninformed updates:

- Ignoring the bandwidth associated to destinations stored in forwarding table during re-computation of a new entries result into more packet loss than needed !!
- Total packet loss: .2 s * 200 Mbps + .5 s * 50 Mbps + .8 * 500 Mbps

Router Model

Upon LS PDU reception

1. **Re-computation of Shortest Path Tree** (SPT) using the topological information stored in Link State DataBase (LSDB) takes about 30 to 50 µs per IGP destination prefix
2. **Routing Information Base (RIB) and Forwarding Information Base (FIB) update** based on the SPT computation (2a and 2b); each update ranging from 50 to 100 µs per prefix
3. **Distribution from FIB towards local FIB (LFIB)** on line cards (3)



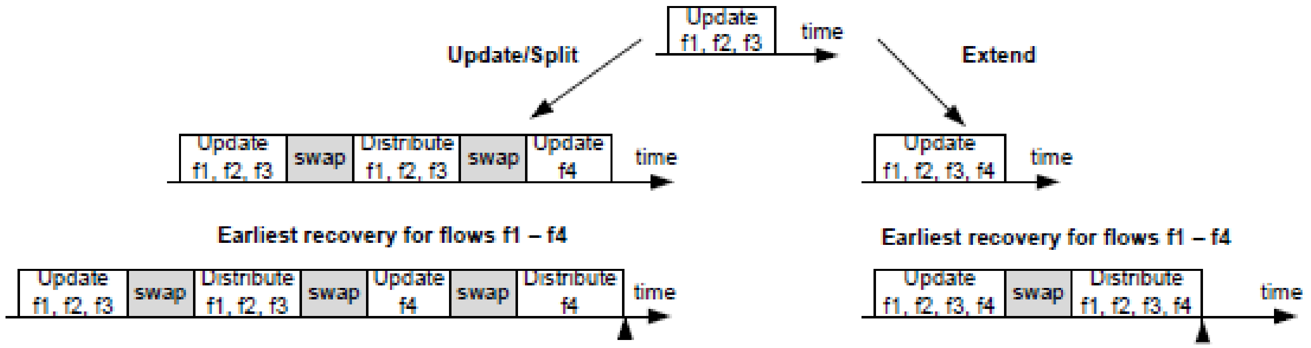
Both update and distribution processes use the router's central CPU in interleaved time slots by swapping between them

The swapping time is determined by the process quantum which can be configured between 10 and 800 ms

Packet loss occurs for traffic flows as long as their corresponding routing table entries are not updated and distributed to the line cards

Methods

- Methods to reduce packet loss during update process
  - i) reorder RIB/FIB entries with respect to their estimated traffic volumes for the next time interval (fixed batch size)
  - ii) reorder RIB/FIB entries and dynamically change size of update-distribution batch (dynamic change of process quantum)
    1. **Extension**: extend the current batch with the RIB/FIB entry associated to the next update
    2. **Splitting**: terminate the current batch and put the RIB/FIB entry associated to the next update into a new update-distribution batch.

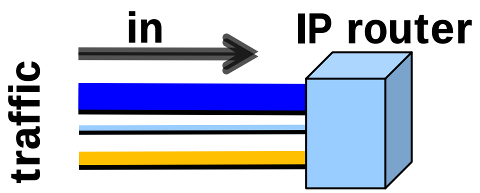


  - For every entry to be updated, compare the resulting packet loss from the application of each alternative, and selecting the one that leads to the least packet loss.

Methods: steps

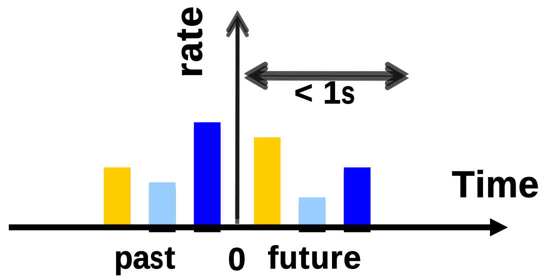
1. Monitor network traffic

Measure traffic rate and number of forwarding table entries hits per destination prefix



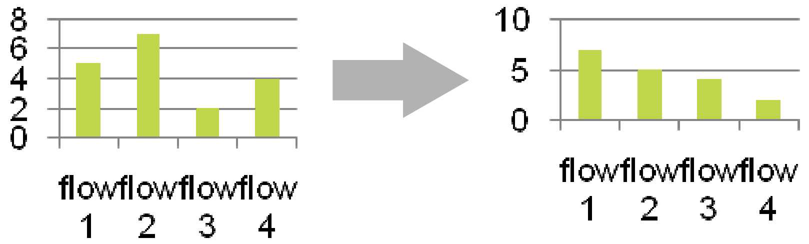
2. Predict network traffic

- Build traffic models per dest prefix in background to enable prediction
- Using e.g. AR(I)MA-GARCH models



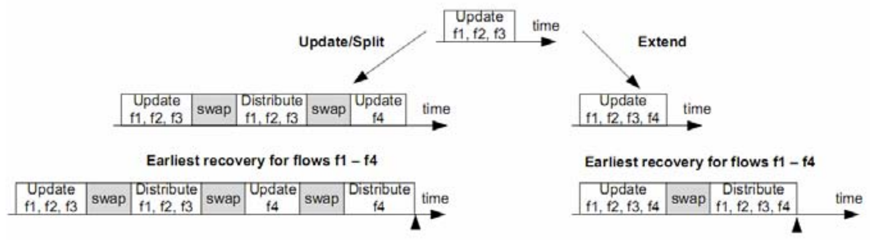
3. Sort network traffic

- Sort network prefixes in decreasing order
- Using hard- and/or software techniques



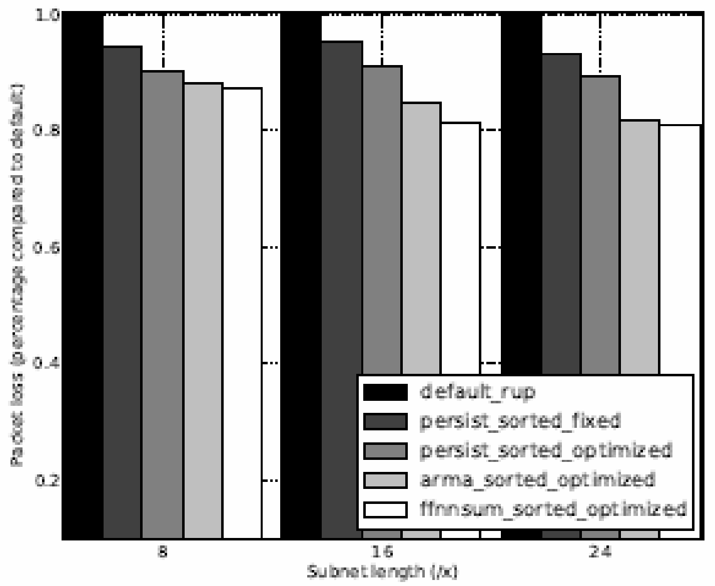
4. Update routing entries

- Update and distribute the associated entries in the RIB and FIB
- Using optimized batching heuristics



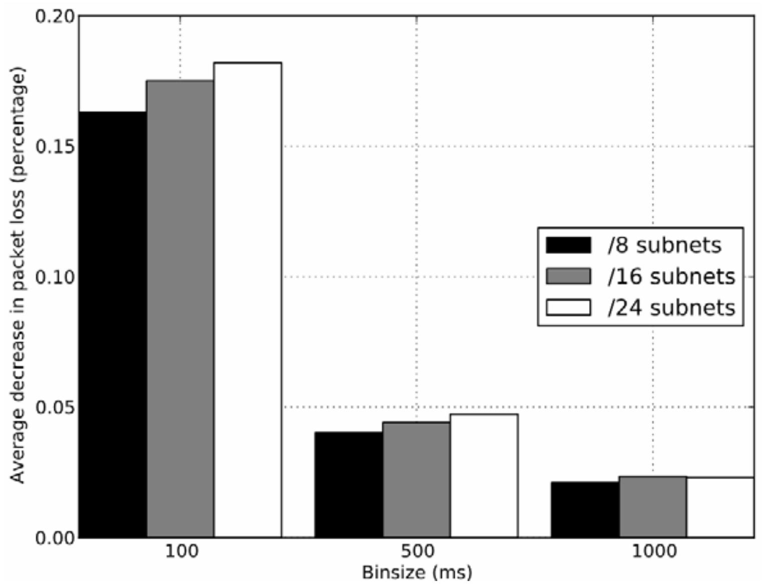
Results

1. **default router update process** (rup): fixed update batches (process quantum) allowing 100 entries
2. **persist sorted fixed**: the routing update process assuming persistent network traffic during the update, sorting the resulting RIB/FIB entries in decreasing traffic volume order with fixed batch size allowing 100 entries
3. **persist sorted optimized**: persistent network traffic with variable batch size
4. **arma sorted optimized**: ARMA model (10; 10) for traffic prediction with variable batch size
5. **ffnnsum sorted optimized**: FFNN model for traffic prediction with variable batch size

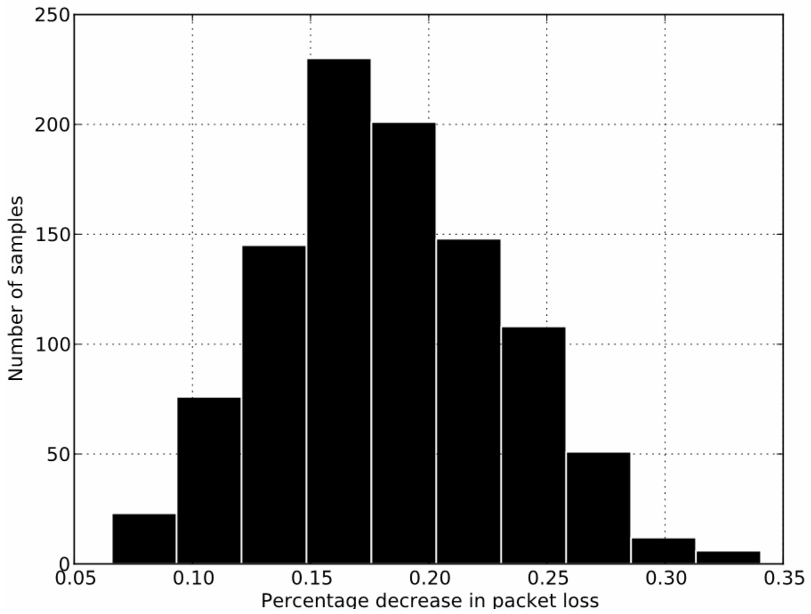


Average packet loss reduction in percentage using time intervals of 100 ms for several update schemes

Results: Statistical behavior of packet loss decrease



Average packet loss decrease vs. default process over all aggregation levels



Distribution of decrease in packet loss (/24 subnet prefixes and bin size 100 ms)

- Gain of traffic-driven approach increases for lower aggregation levels

  Gain from 2 to 20 % in packet loss reduction
- Statistical spread of packet loss decrease (vs. default RUP) seems to follow a normal distribution with solely positive values

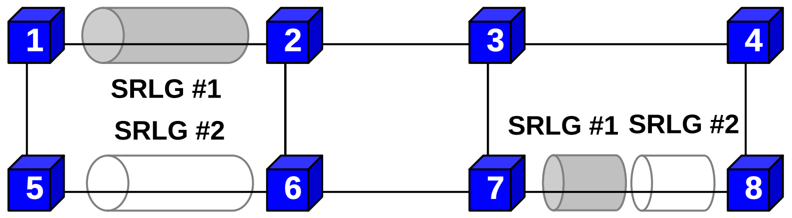
What did we learn ?

- No traffic traces record / repository openly available from experimental facilities
- Validating methods requires a model verifiable either formally and/or simulation
- Move directly to experimental facility would be too time consuming
  - Note: even if time wasn't a concern, control of execution parameters is a pre-requisite

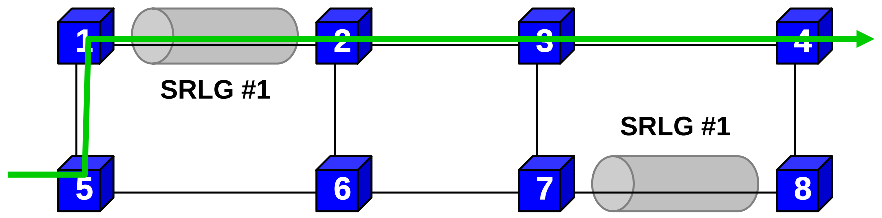
Shared Risk Link Group (SRLG) detection and identification from mining of link state protocol data

Problem Statement

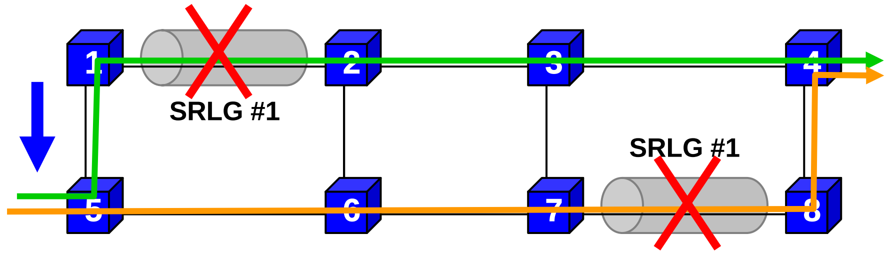
- SRLG #1 includes links[1,2] and [7,8]: these links share a common risk => they fail simultaneously when associated shared risk occur



- At node 5, primary path to D: 1-5-2-3-4

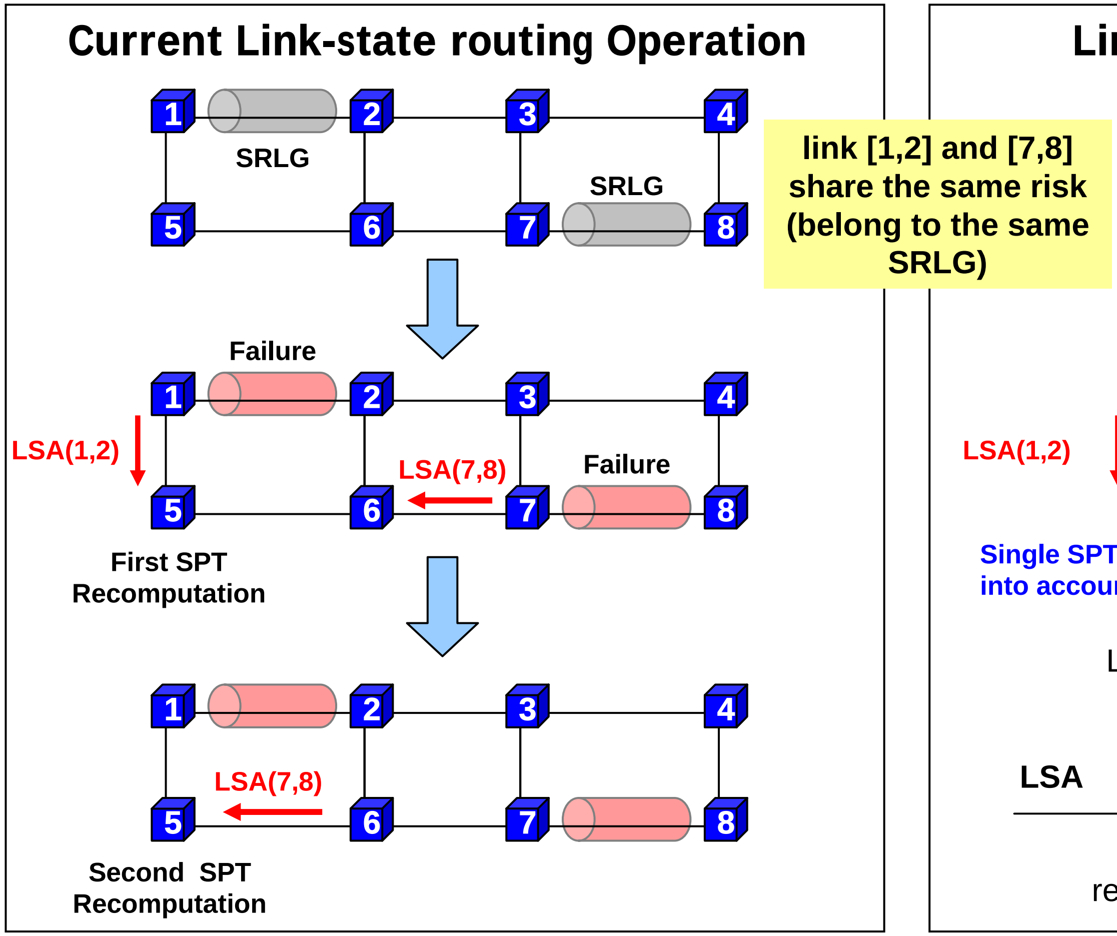


- In link-state routing: upon reception of link[1,2] state change, node 5 computes alternate path to D via node 5 (path 5-6-7-8-4), as long as state of link[7,8] not locally updated

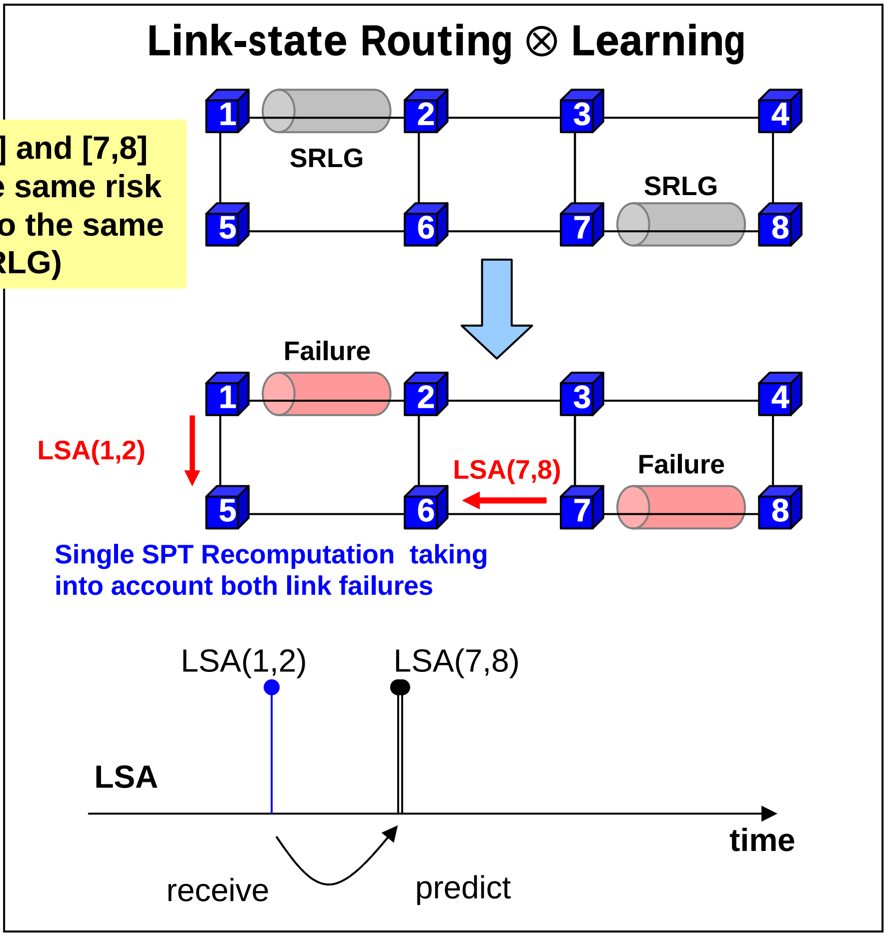


=> Problem: SRLG disjointness of forwarding paths

Objective



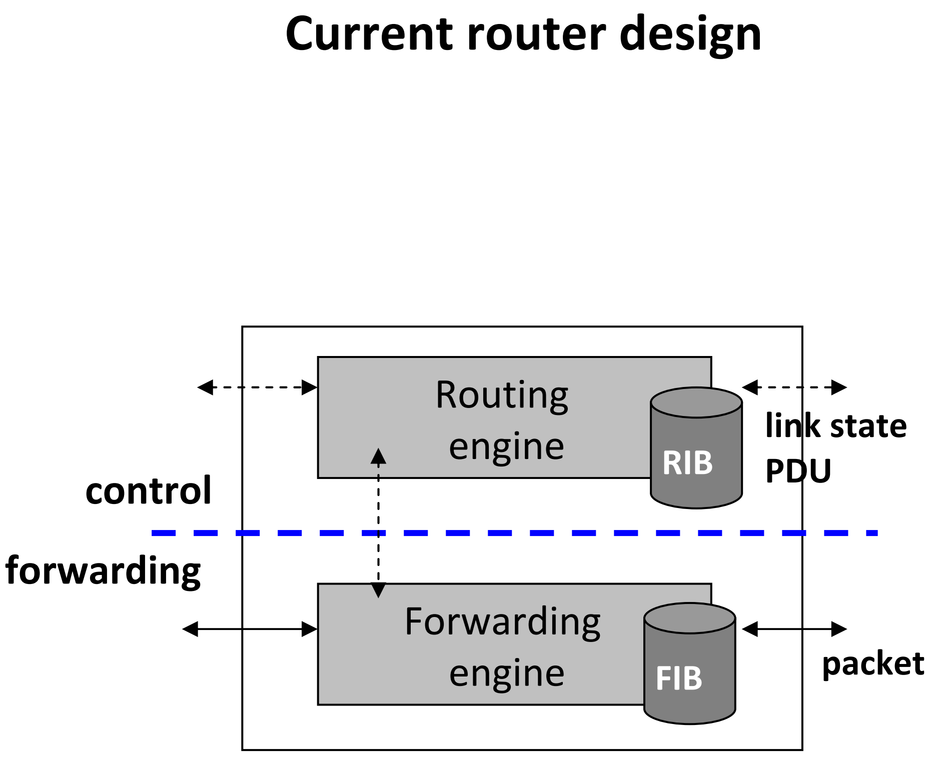
link [1,2] and [7,8] share the same risk (belong to the same SRLG)

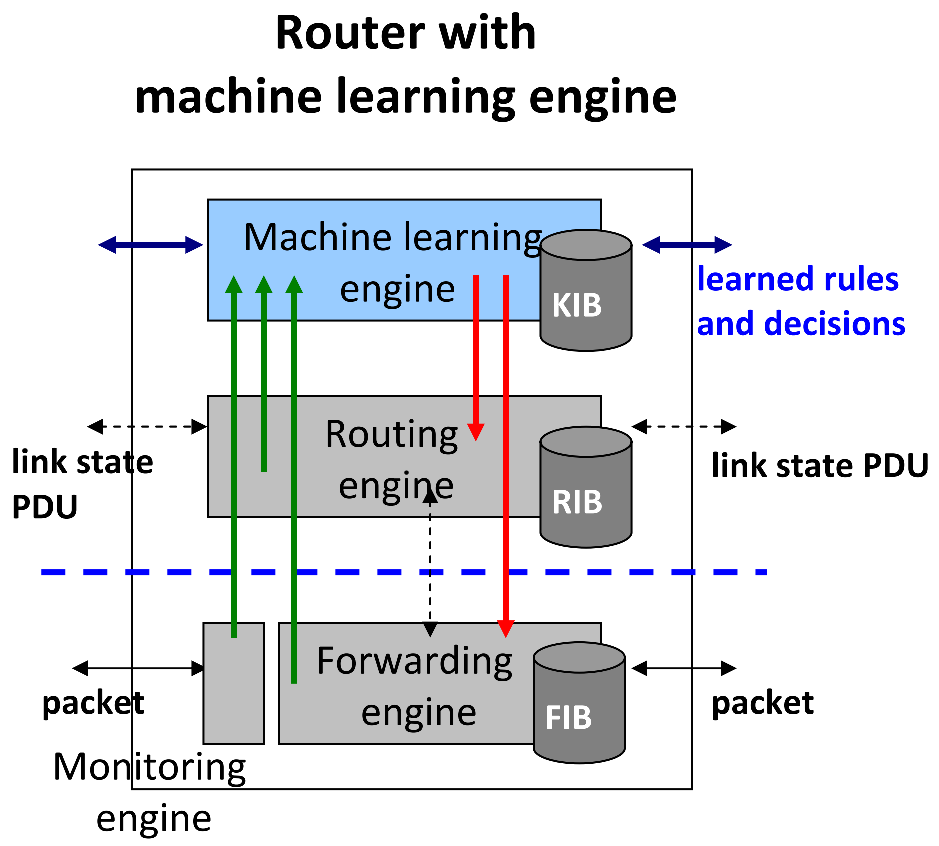


Link-state Routing ⊗ Learning
1
2
3
4
5
6
7
8
SRLG
SRLG
Failure
Failure
LSA(1,2)
LSA(7,8)
Single SPT Recomputation taking into account both link failures
LSA(1,2)
LSA(7,8)
LSA
time
receive
predict

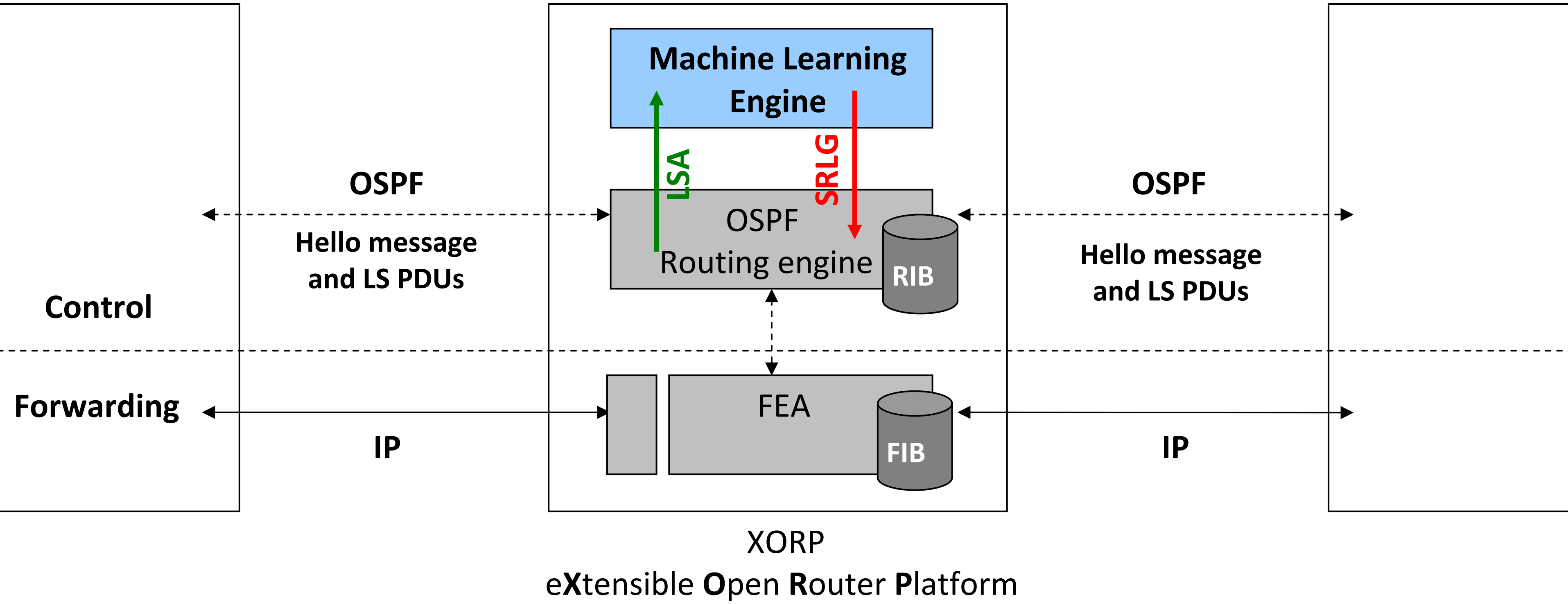
Infer Shared Risk Group (SRLG) from Link state routing information (LSA) arrival pattern to prevent successive SPT re-computation upon shared-risk failure (affecting multiple links simultaneously)

Router Model





Experimental Development



- **Simultaneous network failures inferred from advertized LSAs**
- **Probabilistic SRLG information passed to OSPF (used during shortest path tree computation)**

→ SRG inference based on locally available routing information only (local OSPF link state database)

Experimental Environment

- Emulab network emulation testbed
- IBBT iLab.t facility
  - 100 nodes (dual CPU; 4GB RAM; RAID HDD)
  - 4 or 6 ports (1 Gbit/s Ethernet)
  - 1.5 Tb/s configurable
    VLAN switch (Force10 E1200)
  - Linux / FreeBSD
  - Logical VMs on physical nodes

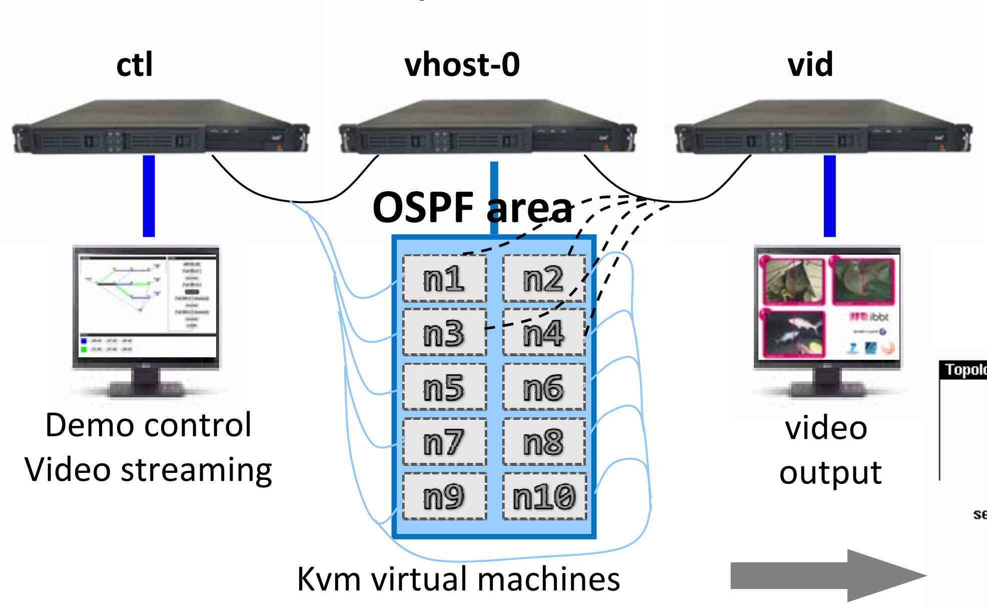
emulab

- Our implementation
  - Linux for routers
  - XORP (routing software)
  - Scripted Link/Node failure emulation on L2/L3

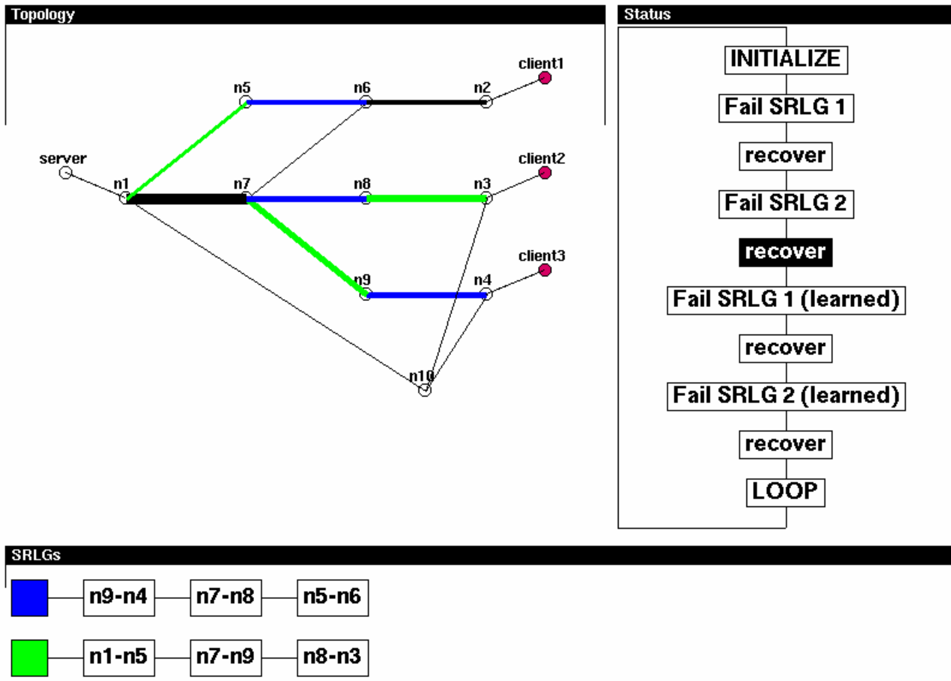
http://www.emulab.net/
http://ilabt.ibbt.be/

Experimental Setup

- OSPF area: one router with SRLG inference capability
- iLab.t setup



emulated topology



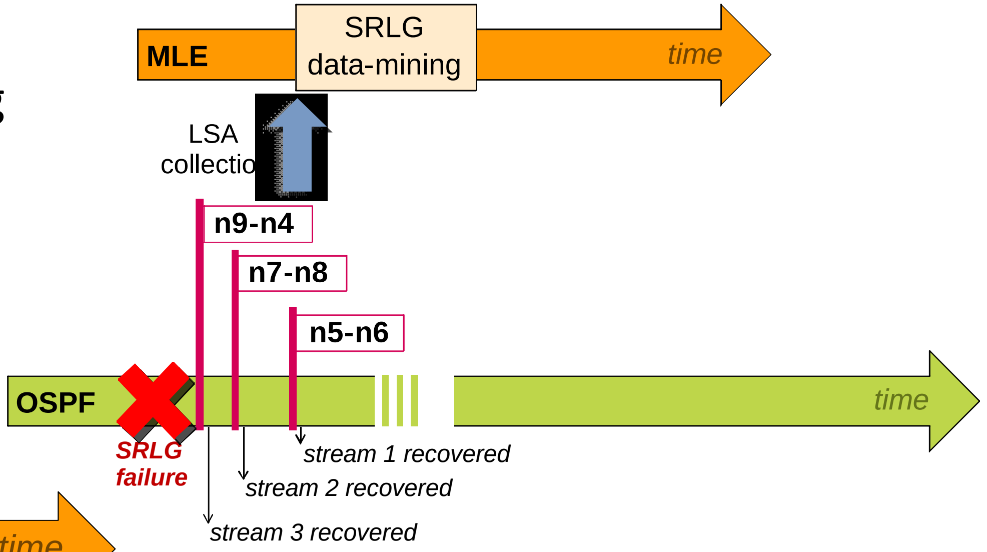
SRLGs			
🟦	n9-n4	n7-n8	n5-n6
🟩	n1-n5	n7-n9	n8-n3

Shows status of the OSPF area

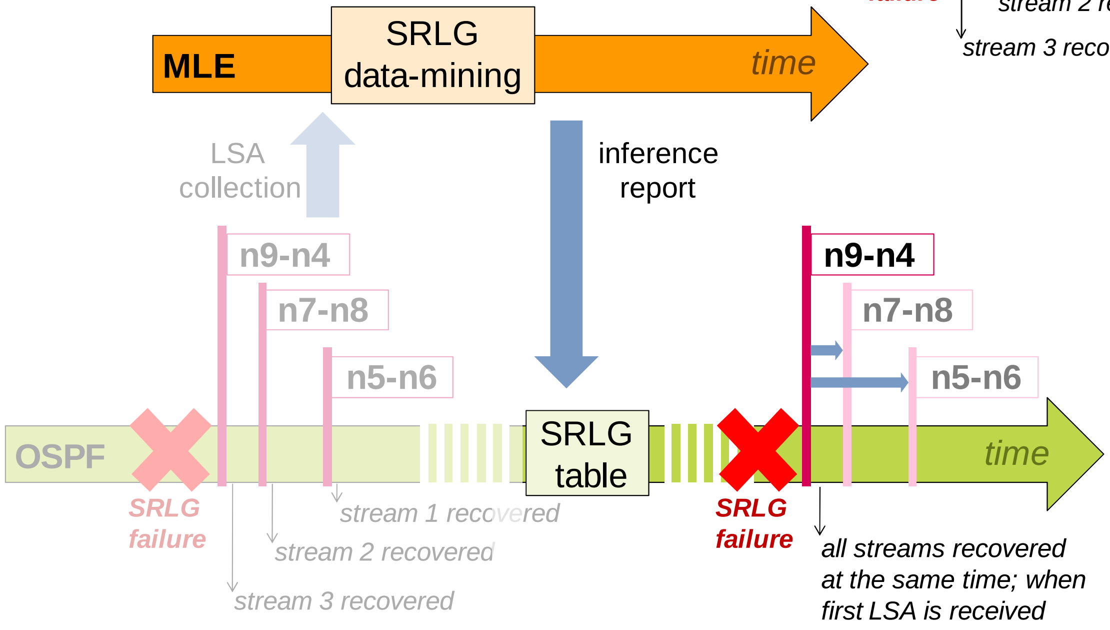
- coloring (matching **learned** SRGs) and fail state
- real-time bandwidth (fluctuating line width)
- server and three clients (video stream endpoints)

Experimentation: recovery times

SRLG not learned: forwarding path recovered one by one as the failing link in their respective path becomes known through LSAs (spread out in time (s)



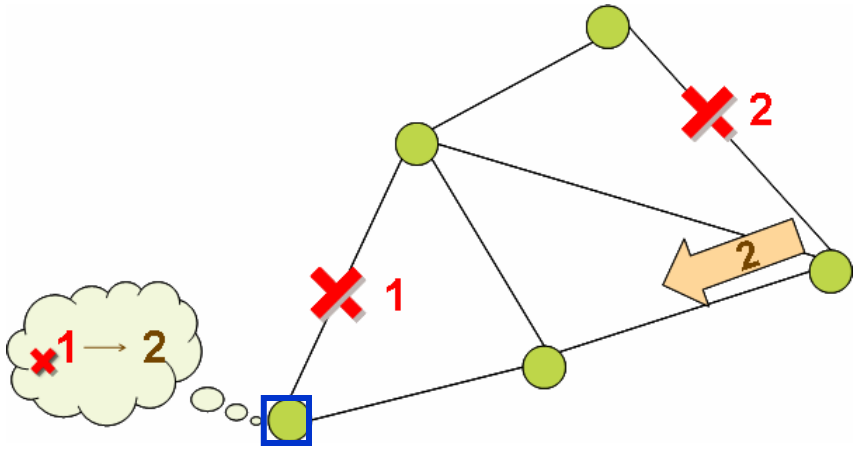
SRLG learned



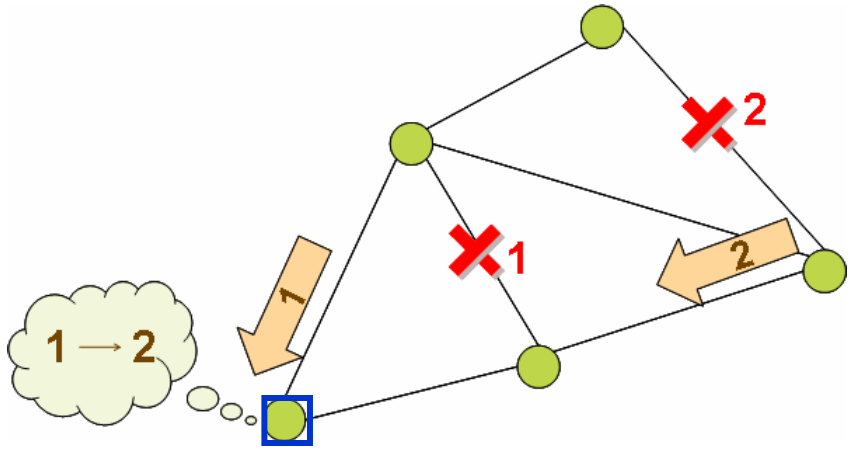
What did we learn ?

SRLG inference through machine learning can be executed (XORP <-> MLE)

For failures of SRLG containing at least one adjacent link: recovery time almost equal to local failure detection time (~100ms) – instead of 1s in standard OSPF



For failures of SRLG containing only non-adjacent links: once first failing link is known, all paths can be recovered at the same time by the "inferring" node □



The iLab.t exp. platform enables

i) **to control at fine-granularity of (changes in) routing protocol and machine learning component operations wrt network running conditions**

ii) **to produce repeatable and reliable results**

iii) **to possibly run an experiment verifying scaling properties (as we use only 3% of the number of available of machines)**