“Light-mesh” time division multiplexing for CWDM/DWDM networks

Alpár Jüttner, Jie Zhang

Centre for Wireless Network Design
University of Bedfordshire
Luton, UK

July 1, 2009

Apology

- This presentation is about a Light-Mesh architecture
- Sorry for the “messy” extended abstract on the conference CD

Outline

Goal

Provide

- full mesh connectivity and
- sub-wavelength bandwidth granularity

in circuit switched all optical network.

Connectivity in All Optical Networks: Light-path

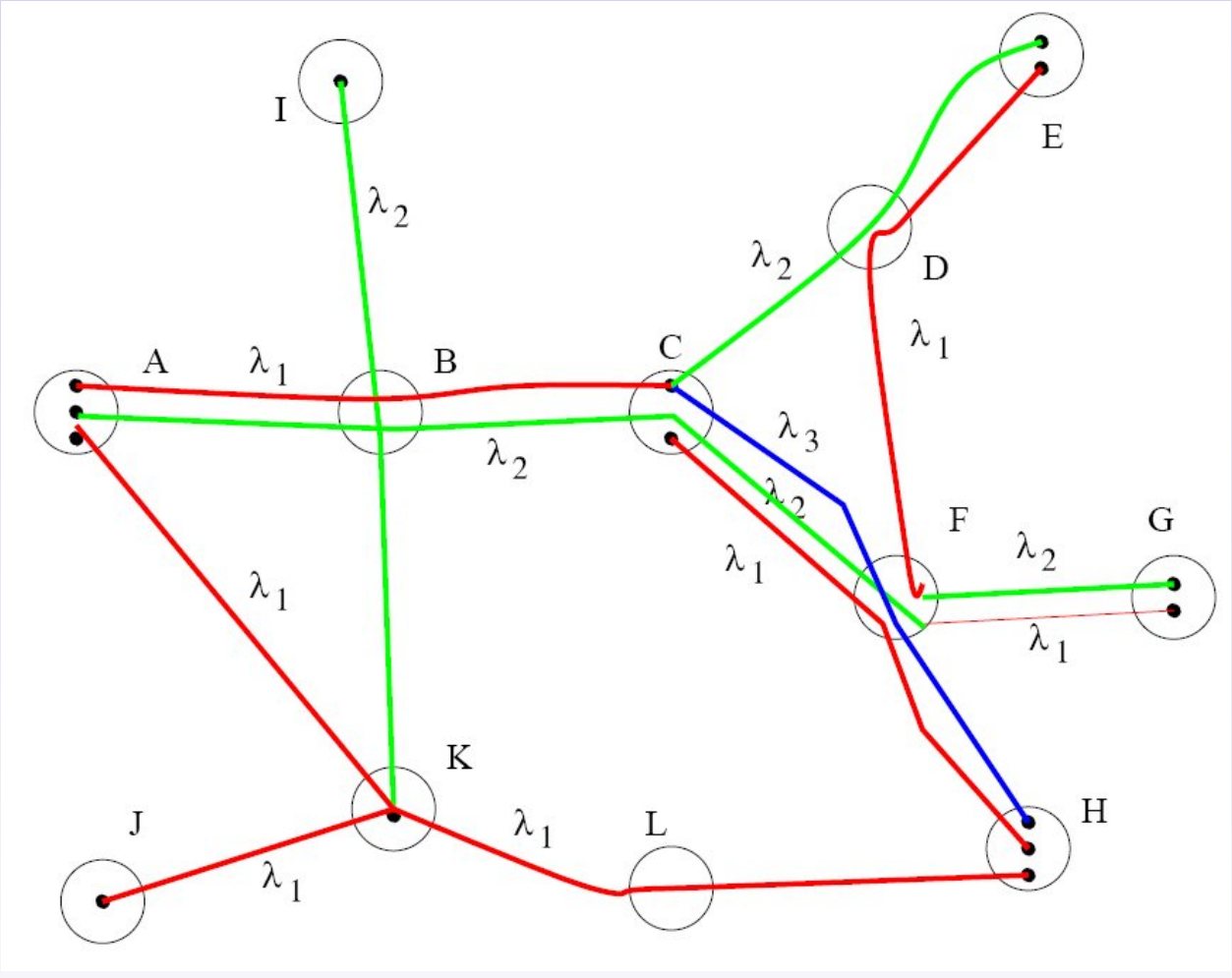


Figure Source:

S. Q. Zheng, A. Gumaste. *SMART: An Optical Infrastructure for Future Internet*

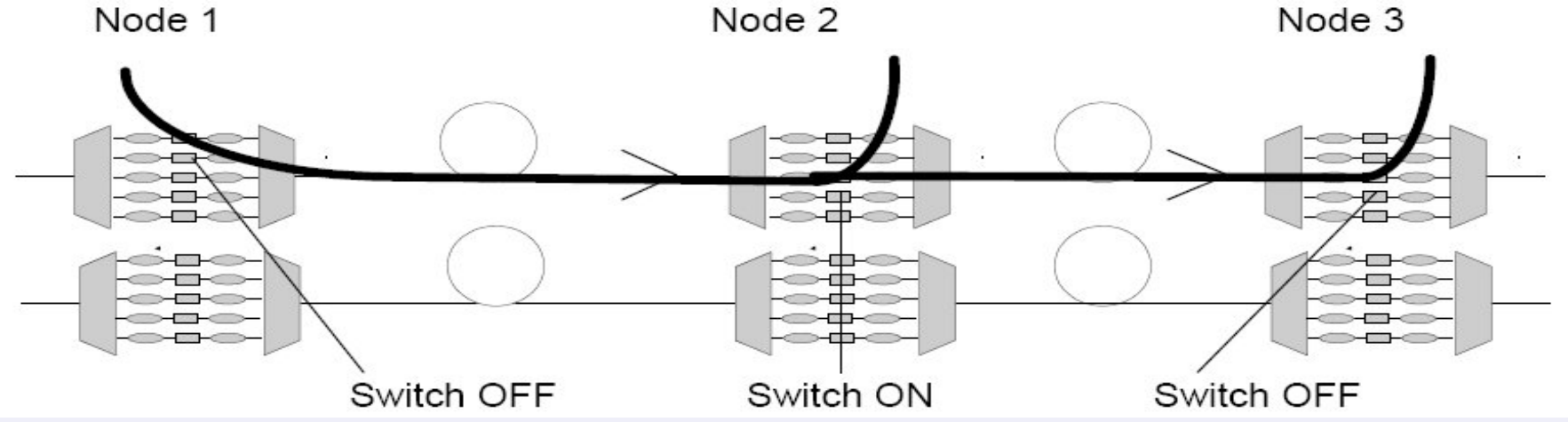
Connectivity in All Optical Networks: Light-tree

Establishes multicast tree topology with the use of “optical splitter”.

Laxman H. Sahasrabuddhe, Biswanath Mukherjee, *Light-Trees: 0ptical Multicasting for Improved Performance in Wavelength-Routed Networks*. IEEE Communications Magazine Feb. 1999

Light-trails

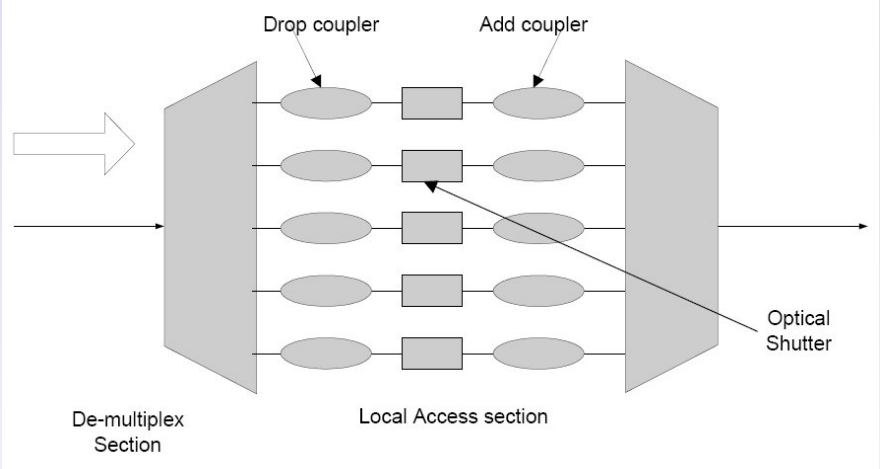
- A light path allowing traffic injection and dropping at intermediate nodes
- It is done by time division multiplexing at the nodes.

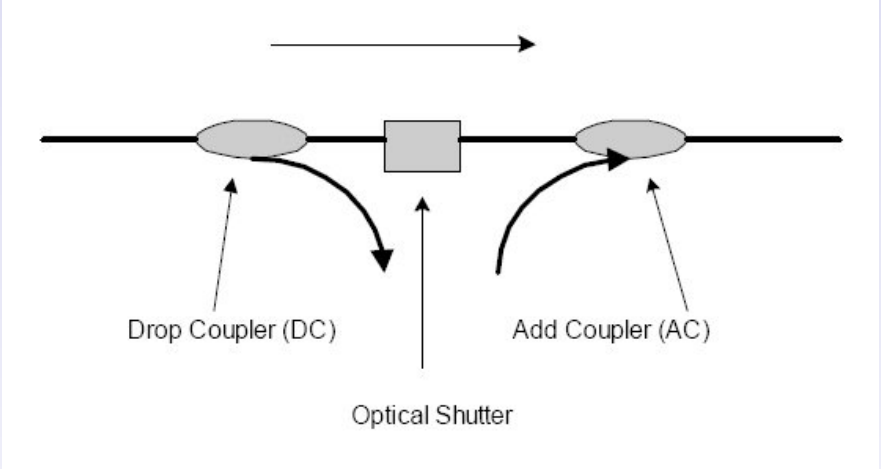


Ashwin Gumaste.
Light-trail and Light-frame Architectures for Optical Networks.
PhD Thesis, Dallas, The University of Texas, 2003.

Light-trails

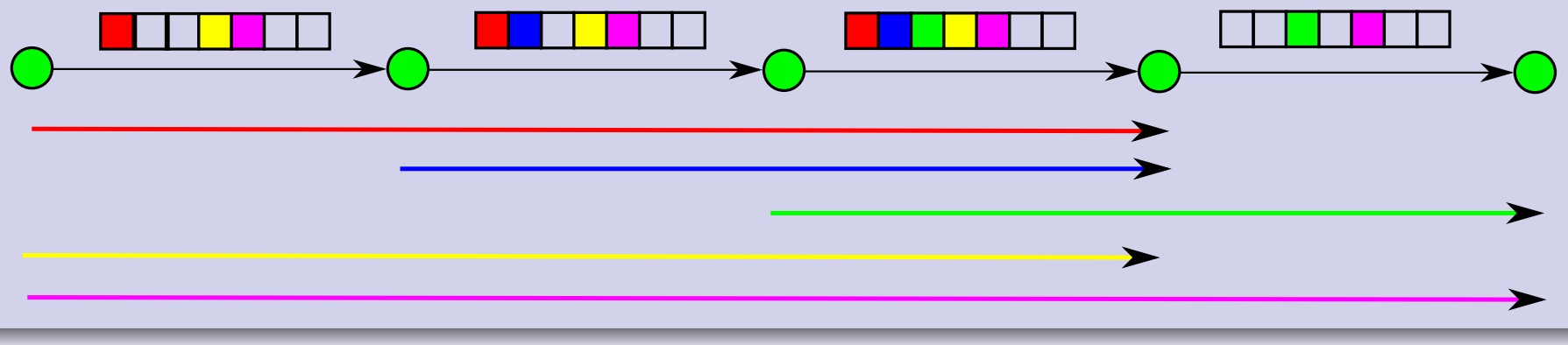
- A light path allowing traffic injection and dropping at intermediate nodes
- It is done by time division multiplexing at the nodes.





Ashwin Gumaste.
Light-trail and Light-frame Architectures for Optical Networks.
PhD Thesis, Dallas, The University of Texas, 2003.

Light-trails (synchronization)



- A cyclic *time frame* containing k slots established at each link
- Each demand is assigned to one of the slots
  - We assume uniform demands
- The time frames must be synchronized along the light trail
  - It's easy

Question

How the demand $\longrightarrow$ slot assignment should be made?

Light-trails

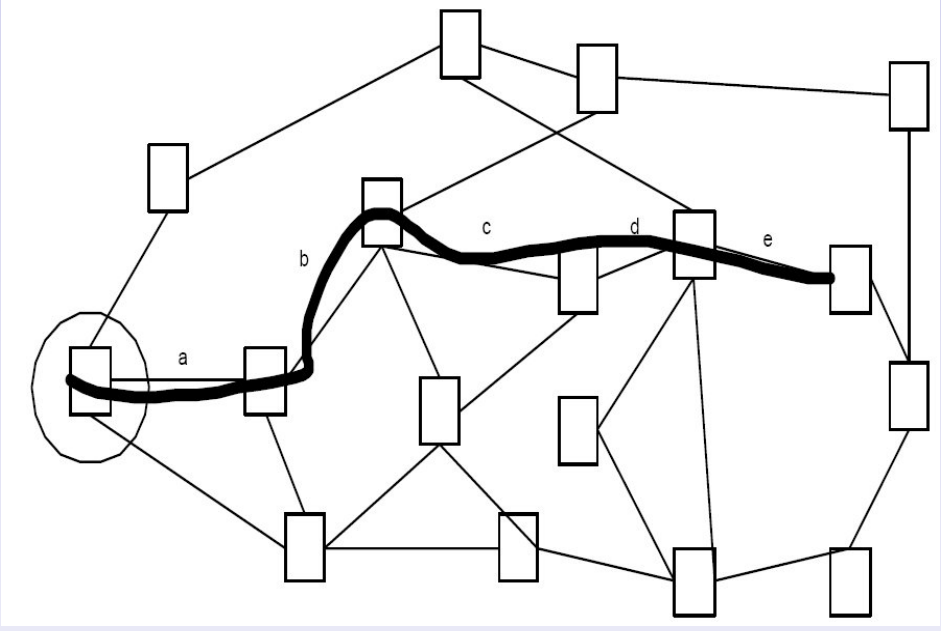
Advantages

- Technologically feasible (it is on the market)
- Allows higher or full connectivity by overcoming the limitation imposed by number of available wavelengths
- No need for (optical) packet header processing
  - No packet processing delay
- No need for (optical) buffers
  - No buffering delay
  - No packet loss due to buffer overflow
- Energy efficient

Limitations

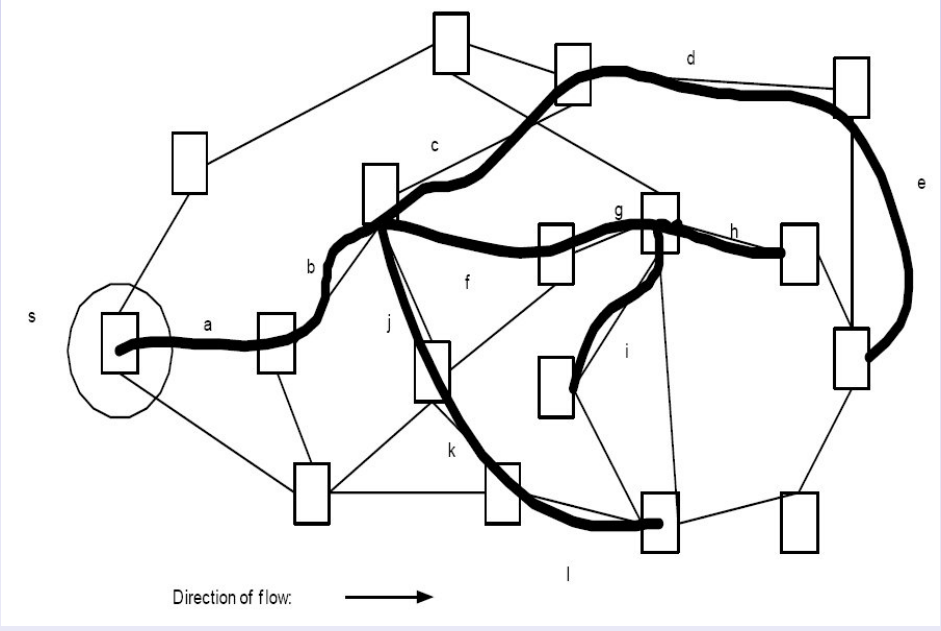
- Still shows scalability problems
- Constrains the feasible network configuration
  - may make the network resource management more difficult
- Static resource allocation

Extended Light-Trail I: Trail splitting



A. Gumaste.
Light-trail and Light-frame Architectures for Optical Networks.
PhD Thesis, Dallas, The University of Texas, 2003.

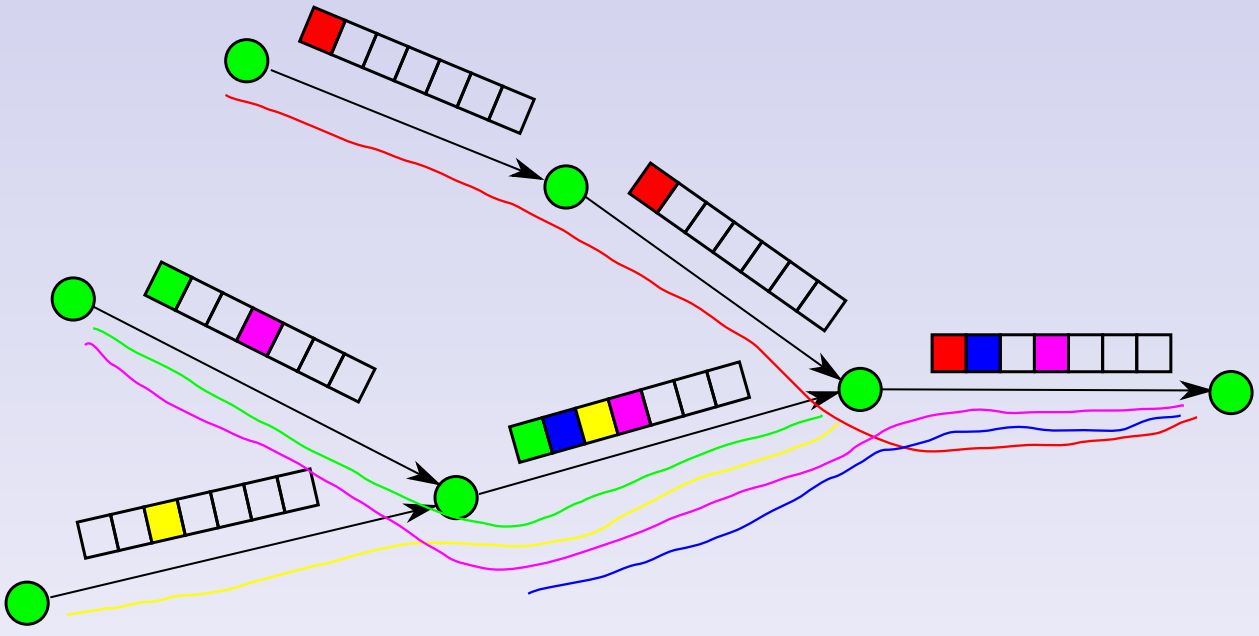
Extended Light-Trail I: Trail splitting



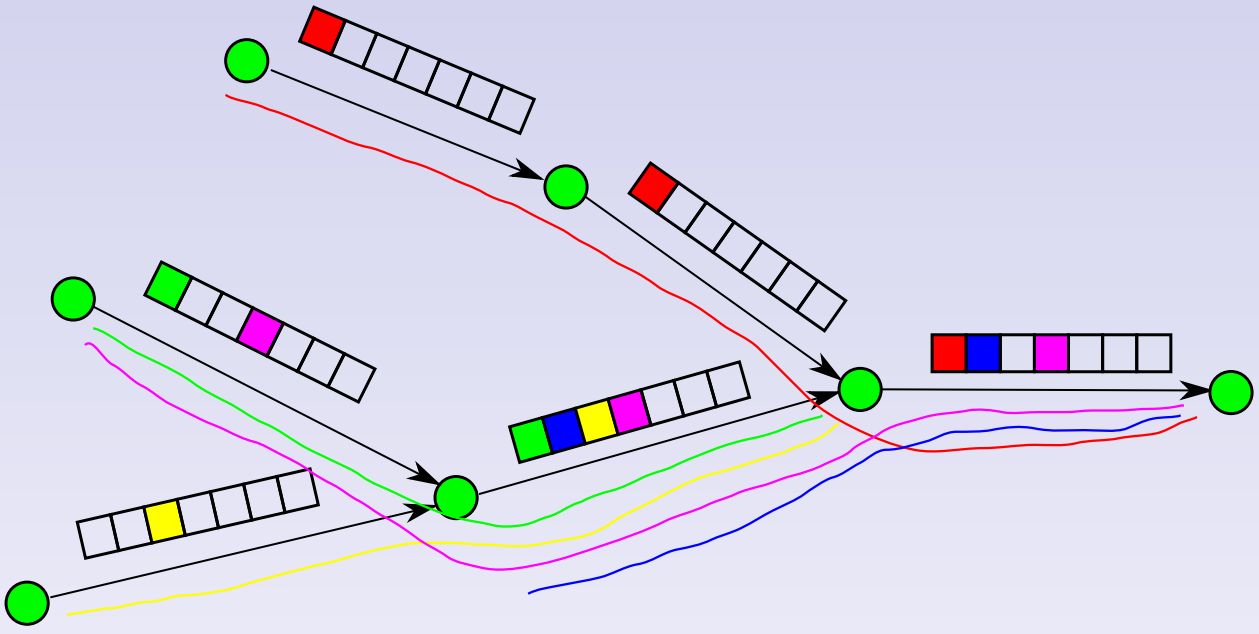
- Implemented by applying additional optical splitters
- No significant additional technical difficulty

A. Gumaste.
Light-trail and Light-frame Architectures for Optical Networks.
PhD Thesis, Dallas, The University of Texas, 2003.

Extended Light-Trail II: Trail Merging

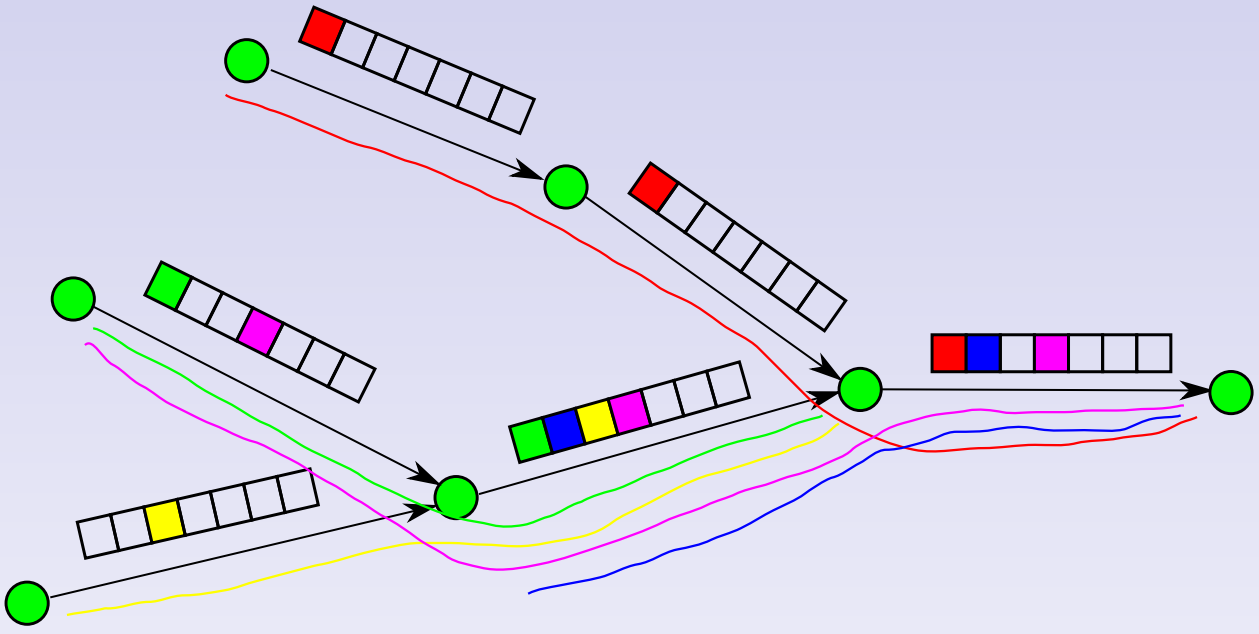


Extended Light-Trail II: Trail Merging



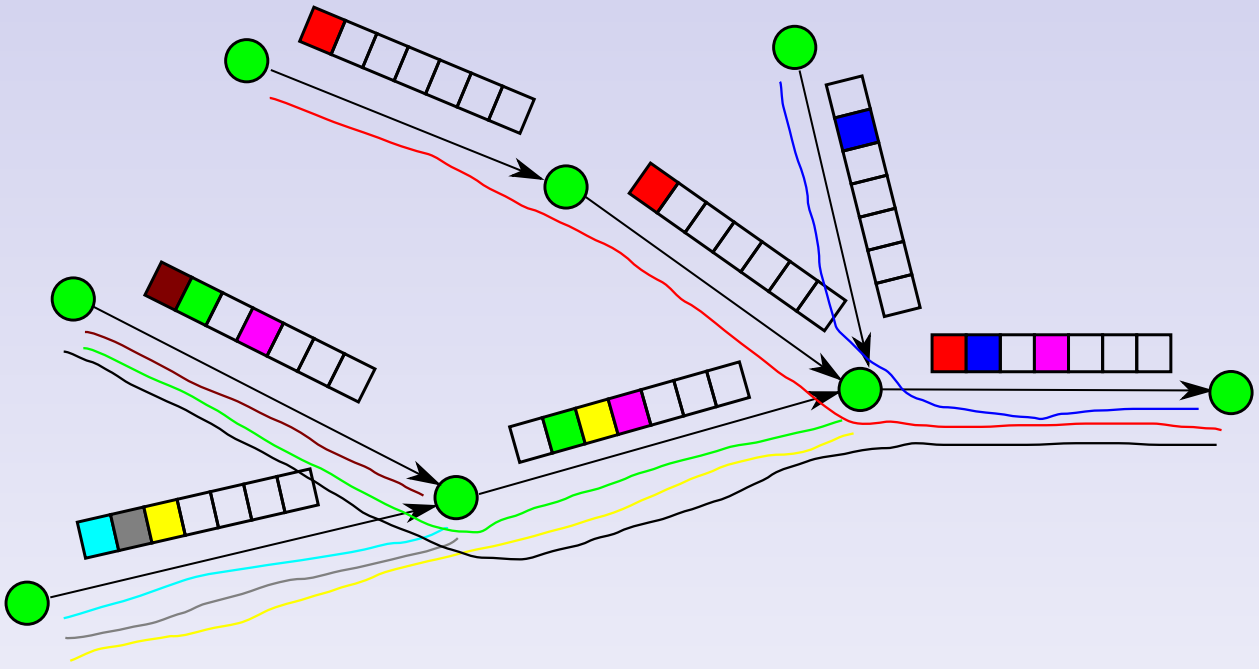
- The time frame synchronization is more difficult
  - Backward propagation of the synchronicity is necessary

Extended Light-Trail II: Trail Merging



- The time frame synchronization is more difficult
  - Backward propagation of the synchronicity is necessary
- Optimal slot allocation?

Extended Light-Trail II: Trail Merging



- The time frame synchronization is more difficult
  - Backward propagation of the synchronicity is necessary
- Optimal slot allocation?

Light-Mesh: Light-Trail with Splitting + Merging

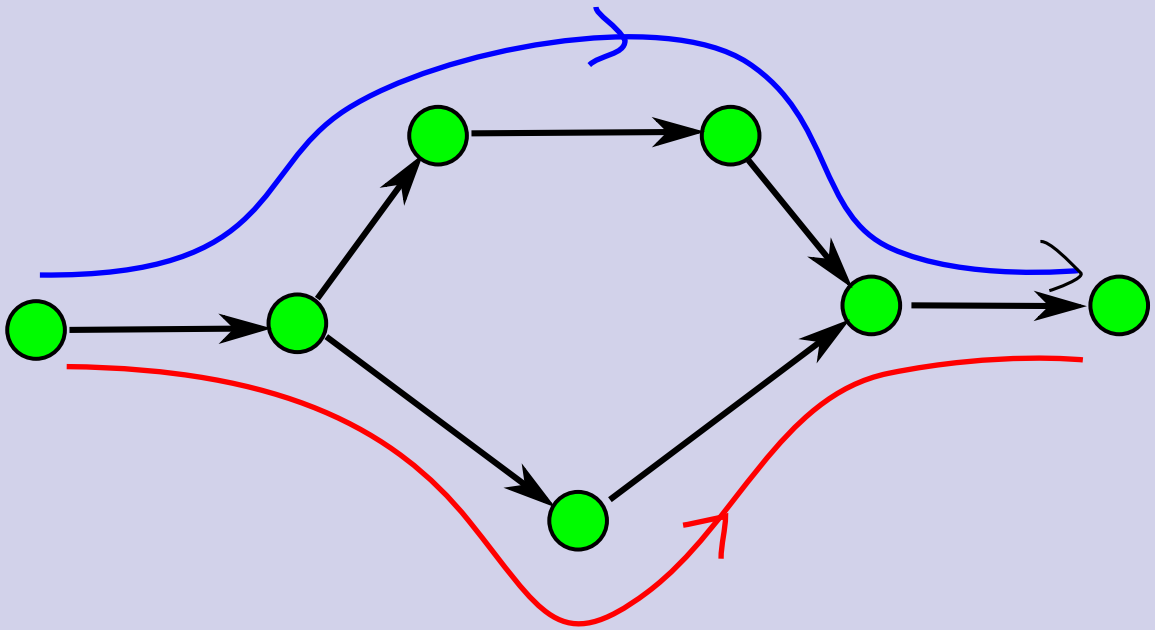
- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies

Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

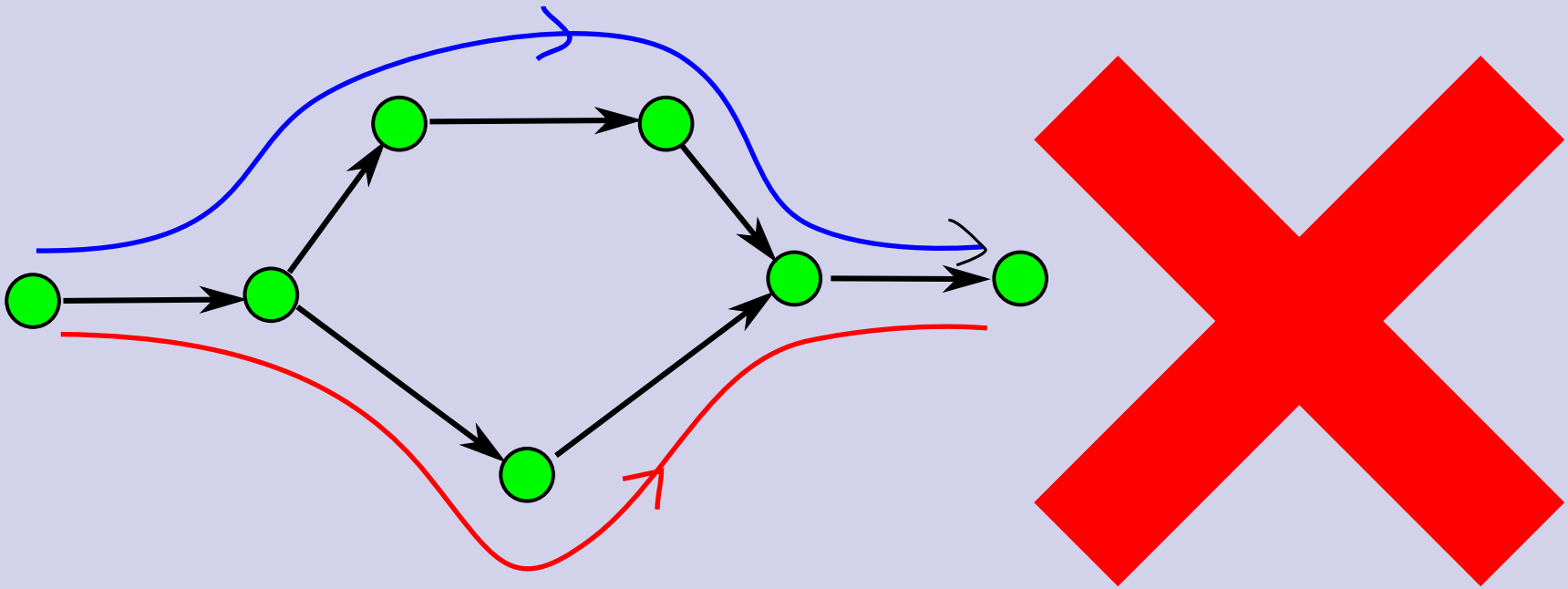
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

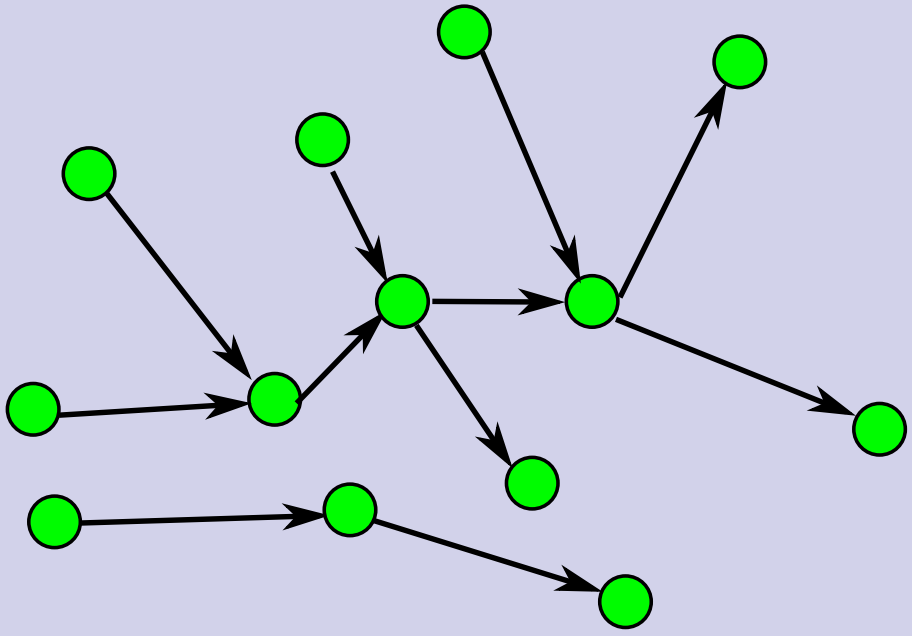
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

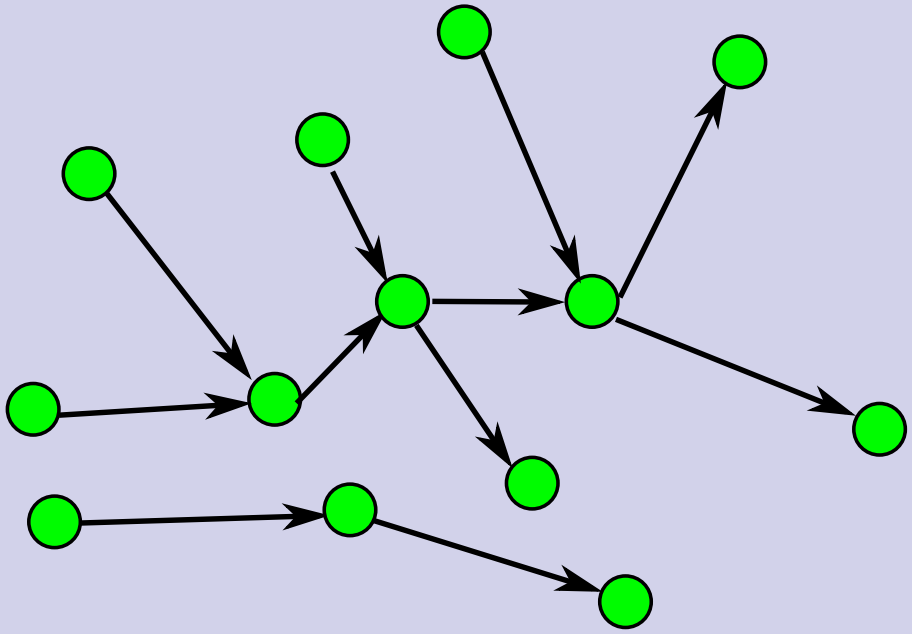
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

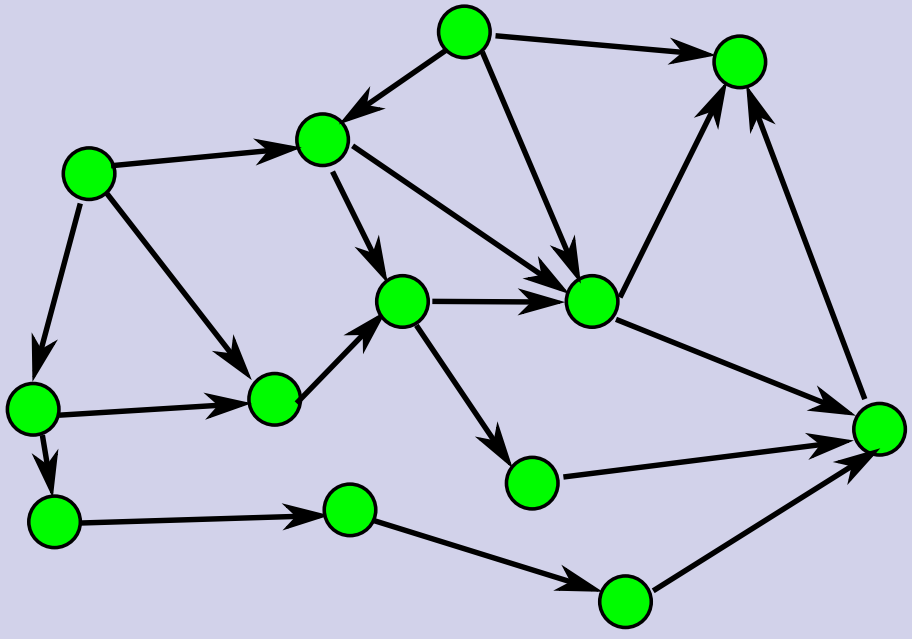
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

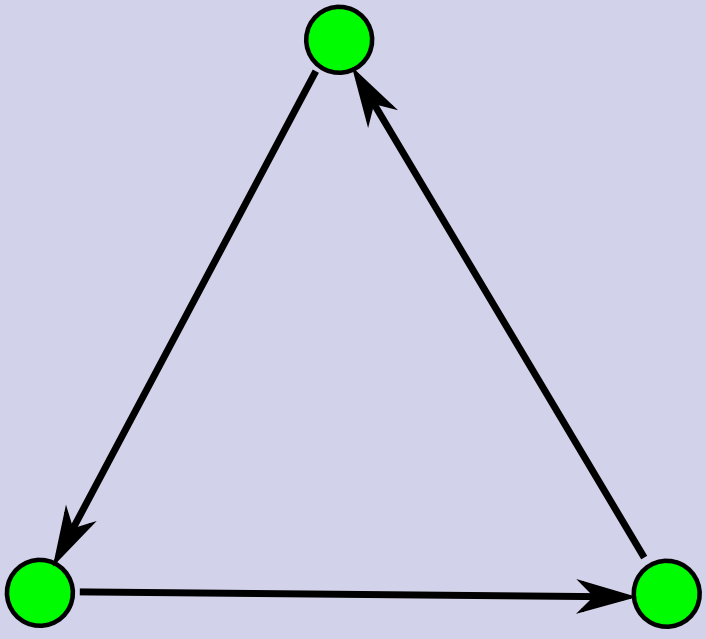
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

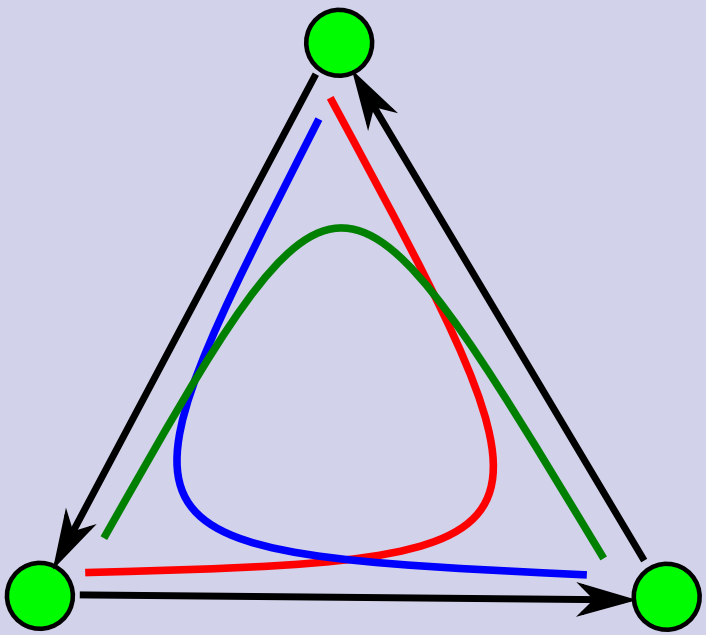
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

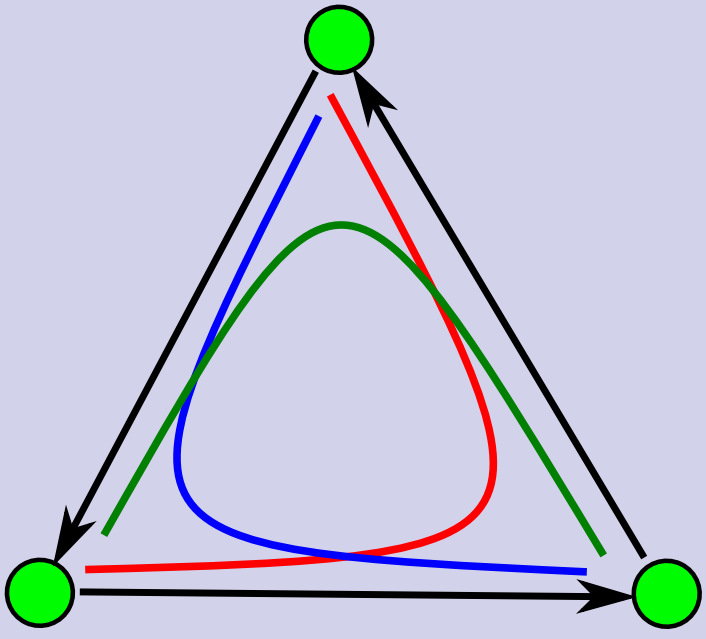
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies

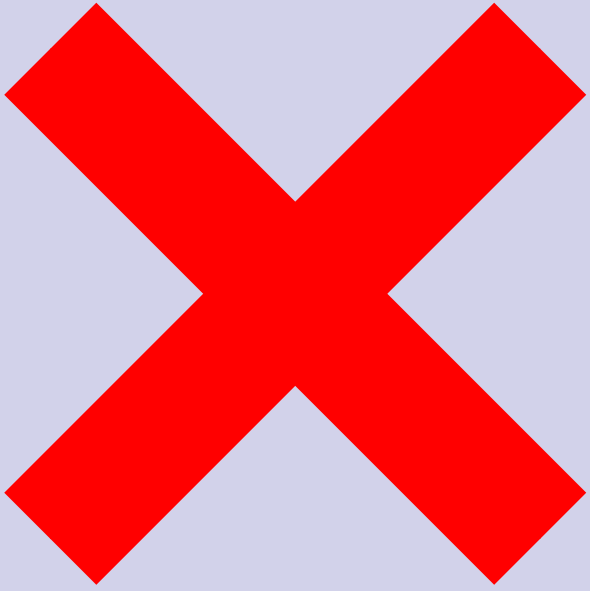


Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies

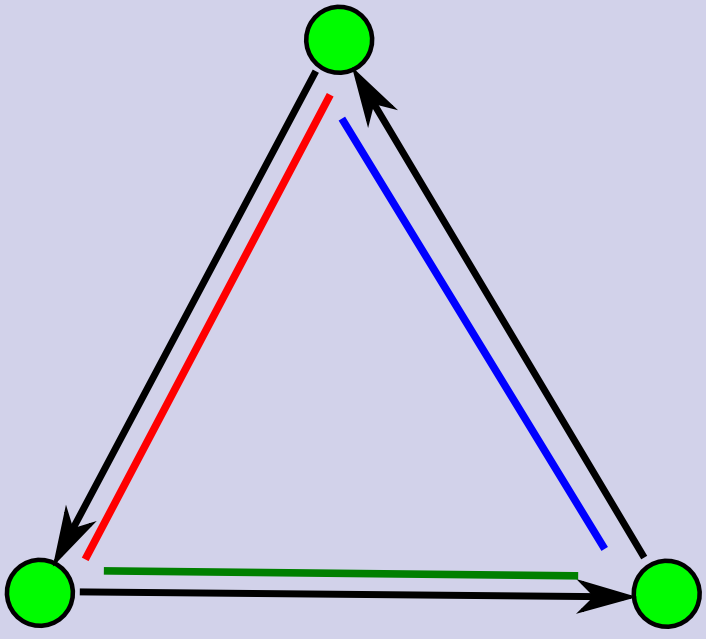




Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

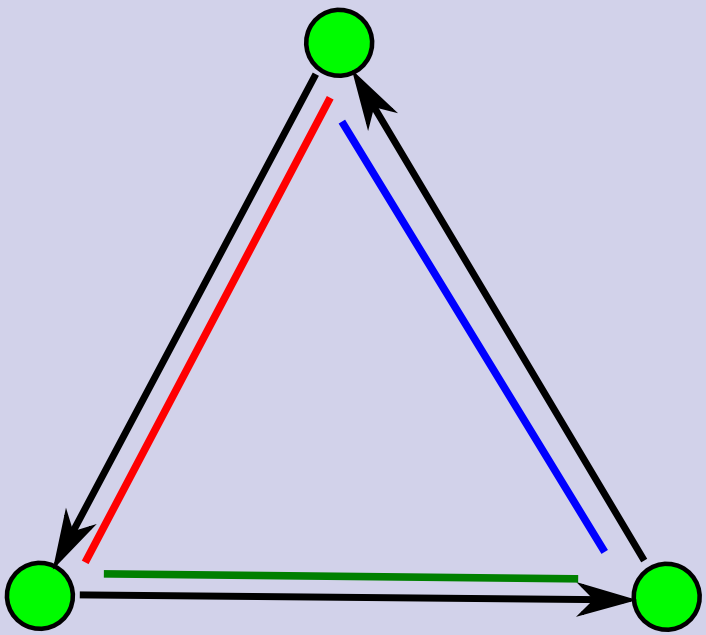
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies



Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies

- The feasibility of a configuration depends on the demands (routes), not on the used links.

Light-Mesh: Light-Trail with Splitting + Merging

- Allows complex single wavelength topologies.
- Overdetermined (cyclic) dependency may appear in the frame synchronicity

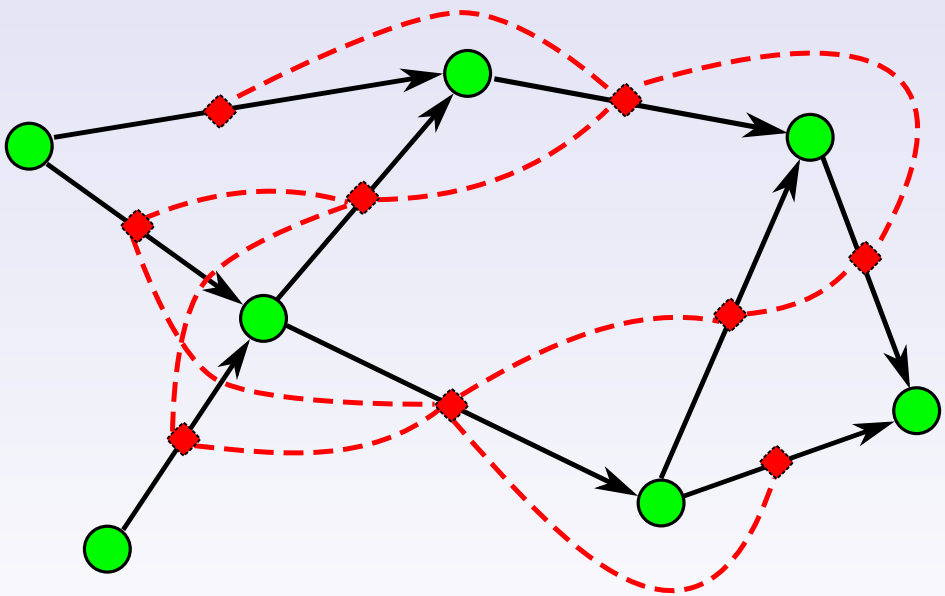
  $\Longrightarrow$ not all configurations are feasible
- It isn't trivial which are the feasible topologies
- The feasibility of a configuration depends on the demands (routes), not on the used links.
- An efficient tool is needed to check whether a configuration is allowed or not.

Line graph

Definition (Line-Graph)

Let $G = (V, A)$ be a directed graph. Its **line-graph** $L(G) = (A, E)$ is an undirected graph the nodes of which correspond to the edges of G, and two nodes a_1 and a_2 are connected by an edge if and only if the head a_1 and the tail of a_2 is the same node in G, i.e.

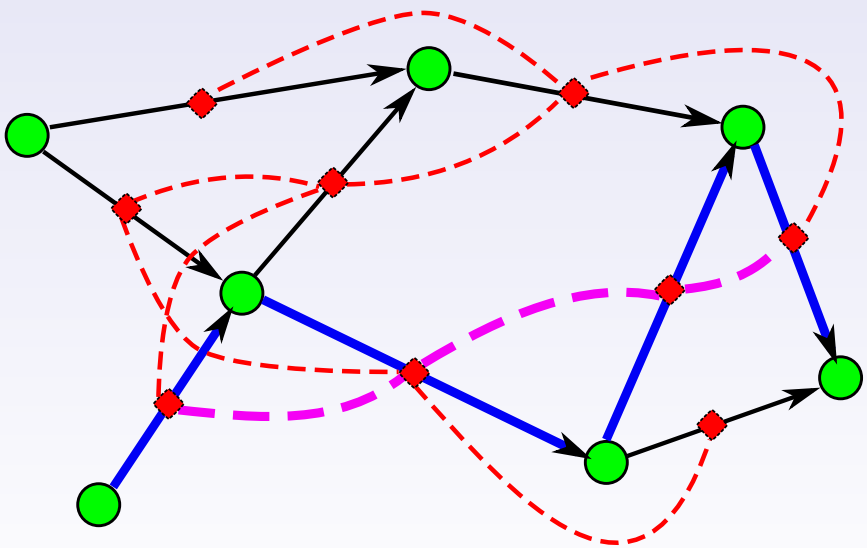
$$E := \{(\overrightarrow{xu}, \overrightarrow{uy}) : x, y, u \in V \text{ and } \overrightarrow{xu}, \overrightarrow{uy} \in A\} \tag{1}$$

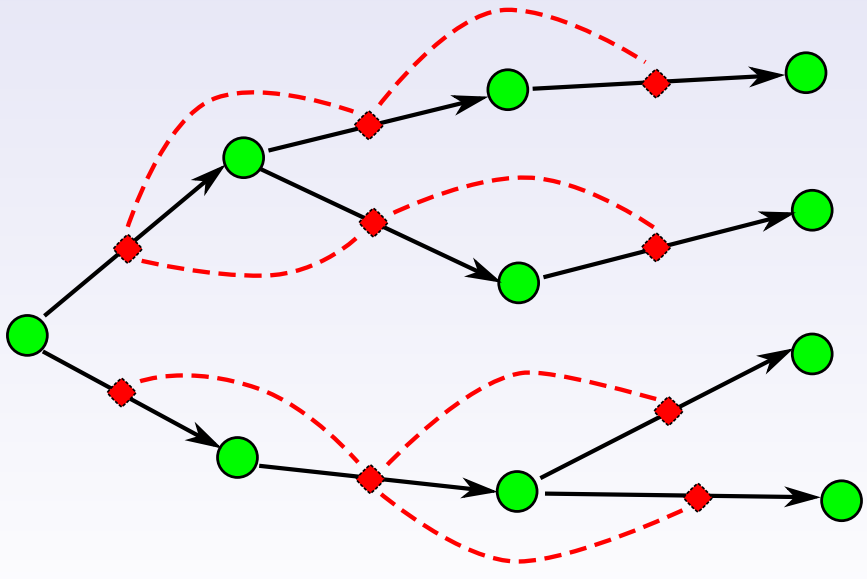


Line graph of the routes

Line graph of paths and rooted trees

- The image $L(p)$ of a path p of lenght k is a also path, which is of length $k-1$.
- If the length of p is 1, then $L(p)$ is a single node in $L(G)$.
- Similarly, a rooted *branching* T (i.e. a subtree with all edges directed oppositely to the root) naturally correspond the a forest (a set of disconnected trees) $L(T)$ in $L(G)$.





Line graph of patch and rooted trees

Theorem

A system $\{d_1, d_2, \ldots, d_k\}$ of demands/routes are assignable to one single wavelength light-mesh if and only if the union U of their images $L(d_1), L(d_2), \ldots, L(d_k)$ in the line graph forms a forest (i.e. an acyclic subgraph) in $L(G)$.

Usage

Using this theorem one can easily check

- if a set of routes fit a single Light-mesh
- if a new route can be added to a Light-mesh

making it easy to implement heuristic optimization algorithms

Some experiments

Greedy Test (How many Light-meshes are needed for full connectivity

- Compute a full $n \times n$ routing using shortest paths.
- Add the paths one-by-one to the first Light-mesh where you can.
- If a path cannot be added to any of the existing Light-mesh, start a new one.

Some experiments

Greedy Test (How many Light-meshes are needed for full connectivity

- Compute a full $n \times n$ routing using shortest paths.
- Add the paths one-by-one to the first Light-mesh where you can.
- If a path cannot be added to any of the existing Light-mesh, start a new one.

Some “optimization”

- Pick up a Light-mesh
- Try to reallocate its all routes using the other Light-meshes
- If we could reallocate all routes, get rid of this Light-mesh.

Some experiments

A nice tool for implementing it (an advertisement)

`http://lemon.cs.elte.hu`

- An open source C++ template library targeting combinatorial optimization, especially problems related to graphs and networks.
- It is a member of the COIN-OR initiative, a collection of OR related open source projects.
- You are free to use it in your commercial or non-commercial applications under very permissive license terms.

Some experiments

Some experiments

Terminal

Terminal

```
alpar@piko:~/projects/Papers/LightMesh/code/test>../build/mesh-pac
k <net-50.lgf
Network
  Number of nodes: 50
  Number of arcs: 168
Line graph
  Number of nodes: 168
  Number of arcs: 600
Source 49
Add new wavelength: 1
Source 48
Add new wavelength: 2
Source 47
Add new wavelength: 3
Source 46
Source 45
Source 44
Add new wavelength: 4
Source 43
Add new wavelength: 5
Source 42
Add new wavelength: 6
Source 41
Source 40
```

Some experiments

Terminal

Terminal

```
Add new wavelength: 10
Source 5
Source 4
Source 3
Source 2
Source 1
Source 0
Number of wavelength: 10
NumOfPaths: 564 434 327 332 289 146 120 53 13 4
Total: 2282 in 10 wavelengths
Color 0 has been freed (was: 564)
Color 1 has been freed (was: 671)
Color 2 has been freed (was: 648)
In color 3 there remained 68 paths (was: 676)
In color 4 there remained 197 paths (was: 724)
In color 5 there remained 175 paths (was: 557)
In color 6 there remained 152 paths (was: 561)
In color 7 there remained 91 paths (was: 395)
In color 8 there remained 77 paths (was: 377)
In color 9 there remained 119 paths (was: 395)
NumOfPaths: 0 0 0 676 544 343 312 134 154 119
Total: 2282 in 7 wavelengths
In color 9 there remained 119 paths (was: 119)
In color 8 there remained 77 paths (was: 154)
```

Some experiments

Terminal

Terminal

```
In color 5 there remained 175 paths (was: 343)
In color 4 there remained 197 paths (was: 544)
In color 3 there remained 68 paths (was: 676)
NumOfPaths: 0 0 0 68 377 389 401 352 300 395
Total: 2282 in 7 wavelengths
In color 3 there remained 68 paths (was: 68)
In color 4 there remained 197 paths (was: 377)
In color 5 there remained 175 paths (was: 389)
In color 6 there remained 152 paths (was: 401)
In color 7 there remained 91 paths (was: 352)
In color 8 there remained 77 paths (was: 300)
In color 9 there remained 119 paths (was: 395)
NumOfPaths: 0 0 0 676 544 343 312 134 154 119
Total: 2282 in 7 wavelengths
In color 9 there remained 119 paths (was: 119)
In color 8 there remained 77 paths (was: 154)
In color 7 there remained 91 paths (was: 134)
In color 6 there remained 152 paths (was: 312)
In color 5 there remained 175 paths (was: 343)
In color 4 there remained 197 paths (was: 544)
In color 3 there remained 68 paths (was: 676)
NumOfPaths: 0 0 0 68 377 389 401 352 300 395
Total: 2282 in 7 wavelengths
alpar@piko:~/projects/Papers/LightMesh/code/test>
```

Some experiments

Terminal

Terminal

```
In color 5 there remained 175 paths (was: 343)
In color 4 there remained 197 paths (was: 544)
In color 3 there remained 68 paths (was: 676)
NumOfPaths: 0 0 0 68 377 389 401 352 300 395
Total: 2282 in 7 wavelengths
In color 3 there remained 68 paths (was: 68)
In color 4 there remained 197 paths (was: 377)
In color 5 there remained 175 paths (was: 389)
In color 6 there remained 152 paths (was: 401)
In color 7 there remained 91 paths (was: 352)
In color 8 there remained 77 paths (was: 300)
In color 9 there remained 119 paths (was: 395)
NumOfPaths: 0 0 0 676 544 343 312 134 154 119
Total: 2282 in 7 wavelengths
In color 9 there remained 119 paths (was: 119)
In color 8 there remained 77 paths (was: 154)
In color 7 there remained 91 paths (was: 134)
In color 6 there remained 152 paths (was: 312)
In color 5 there remained 175 paths (was: 343)
In color 4 there remained 197 paths (was: 544)
In color 3 there remained 68 paths (was: 676)
NumOfPaths: 0 0 0 68 377 389 401 352 300 395
Total: 2282 in 7 wavelengths
alpar@piko:~/projects/Papers/LightMesh/code/test>
```

Some experiments

Greedy Test (How many Light-meshes are needed for full connectivity

- Compute a full $n \times n$ routing using shortest paths.
- Add the paths one-by-one to the first Light-mesh where you can.
- If a path cannot be added to any of the existing Light-mesh, start a new one.

Some "optimization"

- Pick up a Light-mesh
- Try to reallocate its all routes using the other Light-meshes
- If we could reallocate all routes, get rid of this Light-mesh.

Results

#node	10	20	50	100	200	500
#λ (greedy)	3	5	10	19	33	72
#λ ("optimized")	3	4	7	17	30	65

Some experiments

Results

#node	10	20	50	100	200	500
#λ (greedy)	3	5	10	19	33	72
#λ (“optimized”)	3	4	7	17	30	65

Note

We didn’t use any route optimization here, we just used what we got.

Optimal Slot Allocation in a Light Mesh

Claim

If there is a link used by k routes, at least k slots are necessary for a collision free allocation.

Optimal Slot Allocation in a Light Mesh

Claim

If there is a link used by k routes, at least k slots are necessary for a collision free allocation.

The opposite is also true:

Theorem

Assume that the cyclic time frame is divided into S slots. Then, the routes can be assigned to the slots in a collision-free way if and only if each link is used by at most S routes.

Optimal Slot Allocation in a Light Mesh

Claim

If there is a link used by k routes, at least k slots are necessary for a collision free allocation.

The opposite is also true:

Theorem

Assume that the cyclic time frame is divided into S slots. Then, the routes can be assigned to the slots in a collision-free way if and only if each link is used by at most S routes.

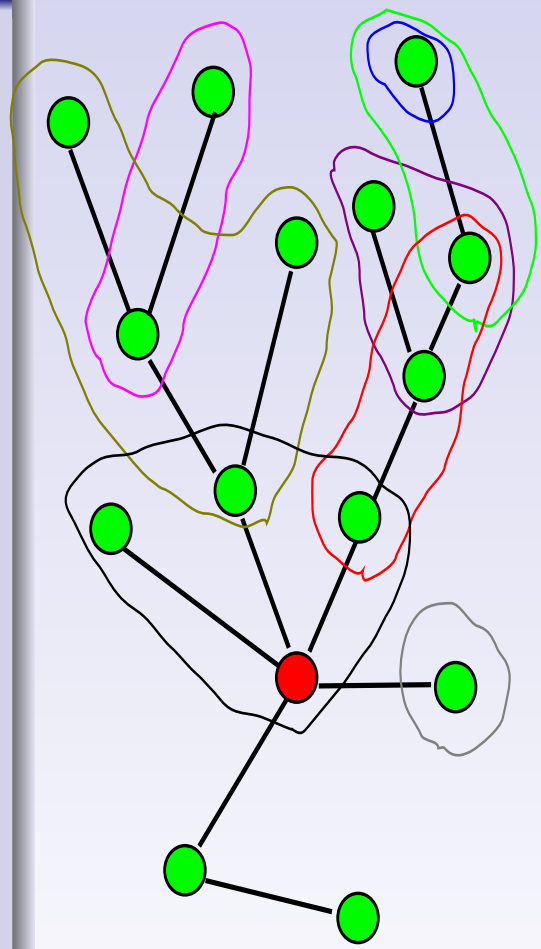
It is a vertex coloring problem in the line graph

- The Light-mesh is a tree (forest) T in the line graph
- The routes correspond to subtrees $L(d_i) \subseteq T \quad (\forall d_i)$
- Goal:
  - assign a color to each subtree $L(d_i)$ such that
  - for each vertex v, all the subtrees using v have different colors

Optimal Slot Allocation In a Light Mesh

Algorithm (Assign the routes to the slots)

1: Let $T_1, T_2, \ldots, T_C$ be the connected components of F.
2: **for all** $c = 1$ to C **do**
3: Choose and arbitrary root vertex $r_c \in T_c$.
4: **end for**
5: **for all** d_i **do**
6: Let $a_i \in L(d_i)$ be the vertex that is the closest to the root of its component.
7: Let $dist(i)$ be the distance between a_i and the root.
8: **end for**
9: **for all** vertices l in $L(G)$ **do**
10: Let free_slots(l) be the list of available slots.
11: **end for**
12: **for all** d_i in increasing order according to $dist(i)$ **do**
13: Let $s \in$ free_slots(a_i).
14: Assign d_i to slot s_i.
15: **for all** l vertices in $L(d_i)$ **do**
16: Remove s_i from free_slots(l).
17: **end for**
18: **end for**



Optimal Slot Allocation In a Light Mesh

Thank you for the attention!