Slot allocation using logical networks for TDM virtual circuit configuration for network-on-chip

Zhonghai Lu and Axel Jantsch

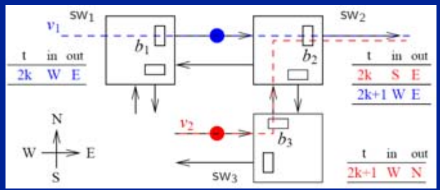
Royal Institute of Technology (KTH), Stockholm, Sweden

Outline

- Background on TDM (Time-Division-Multiplexing) VC
- The problem of slot allocation
- Our approach and contributions
  - The key concept is Logical Network (LN)
  - Contention-free theory
  - LN-oriented slot allocation method
- Results of an industrial case study
- Conclusion and future work

TDM Virtual Circuit

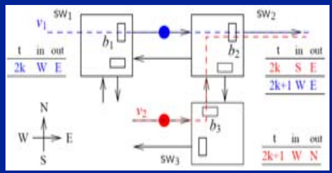
- What is that?
  - A VC is a connection in a packet-switched network.
  - A TDM VC means multiple connections use shared buffers and links in a time-division fashion.
- Example

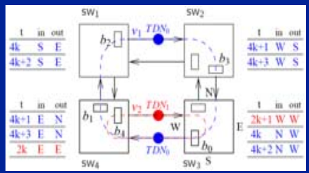


- Why do we need it?
  - Contention-less, offering guaranteed latency and BW.

Two variants of TDM VC

- Open-ended
  - A VC path is not a loop
  - for buffered flow control networks, e.g. wormhole, VCT
- Closed-loop
  - A VC is a loop
  - for buffer-less flow control networks, e.g. deflection

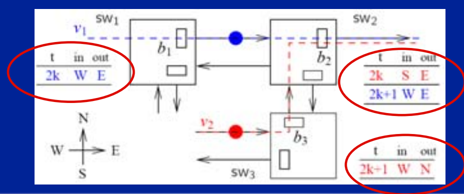




Containers are looped on VCs to carry data packets.

How to configure TDM VCs?

- Properties of a TDM VC
  - A deterministic path
  - Use dedicated time slots to pass buffers, freeing from contention
- Problems of TDM VC configuration
  - Path selection: explore network path diversity
  - Slot allocation: determine when (time slots) VC packets use buffers to be contention free and satisfy BW



Path Selection

- Multi node VC Configuration
- Node visiting order: Hamiltonian Path / Traveling Salesman problem
- Path selection + Slot Allocation by a Depth first search with backtracking
  - Ordering of VCs into a list
  - Finding shortest tour for multi-node comunication
  - Path selection
  - Slot allocation

On the slot allocation problem

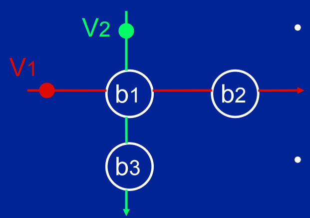
- How to avoid contention?
  - Use exclusive slots
  - Globally synchronize slot tables such that no simultaneous use of a buffer or link is possible
- How to guarantee bandwidth
  - In the first place, contention free
  - In the second place, allocate enough slots

Our approach and contributions

- Propose the concept of Logical Network (LN)
- Formulate a contention-less theory with necessary and sufficient conditions to assign VCs to LNs
- Develop a LN-based slot allocation algorithm

A running example

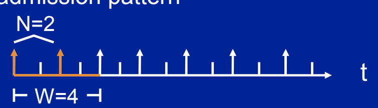
- To explain the concepts and method intuitively



- Given:
  - V_1=<b1, b2>, BW1=1/2
  - V_2=<b1, b3>, BW2=1/4
- Determine:
  - Packet admission pattern for v_1 and v_2 such that
  - (1) there is no contention and
  - (2) both BW requirements are met

Admission pattern: N packets are admitted over W slots

VC traffic flow: repeating this admission pattern

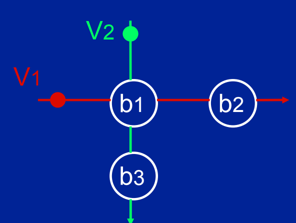


Avoid contention

- **Two steps**
  1. **Slot partitioning with respect to a shared buffer in time domain**
  2. **Slot mapping along a VC path in space domain**
- **Consequence**
  - **Birth of LN, associated _(time slot, buffer)_ pairs.**
  - **Eventually, we can precisely define traffic flow on VCs.**

Avoid contention with the example

1. Slot partitioning
   - $V_1 \cap V_2$ = {b1}
   - Even { (2k, b1)} and odd { (2k+1, b1) } slot sets
2. Slot mapping
   - Map b1's odd slot set on V1: $LN_1^2(v_1,b_1)$ = { (2k+1, b1), (2k,b2) }
   - Map b1's even slot set on V2: $LN_0^2(v_2,b_1)$ = { (2k, b1), (2k+1,b3) }



Satisfy bandwidth

- **A LN owns dedicated slots, thus BW.**
- **For each VC with its LNs, check supported_BW ≥ demanded_BW ?**
  - **If supported_BW > demanded_BW, do slot refinement, i.e., allocate/consume slots not more than necessary**
  - **If supported_BW = demanded_BW, consume slots**
  - **If supported_BW < demanded_BW, LNs are not sufficient to satisfy BW requirement**

Satisfy bandwidth with the example

- **V1:** BW($LN^2_1(v1,b1)$)=1/2 =BW1
- **V2:** BW($LN^2_0(v2,b1)$)=1/2 >BW2

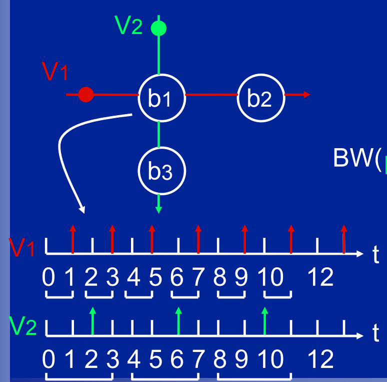
$LN^2_1(v1,b1)$ $LN^2_0(v2,b1)$

b2 0 1 2 3 4 5 6 7 8 9 10

b1 0 1 2 3 4 5 6 7 8 9 10

b3 0 1 2 3 4 5 6 7 8 9 10

t



BW($LN^2_0(v2,b1)$)=1/2 > 1/4 ⇩ Refine slots

b1 0 1 2 3 4 5 6 7 8 9 10

b3 0 1 2 3 4 5 6 7 8 9 10

t

Slot allocation summary

- **The properties of a LN**
  - **Owns dedicated slots in buffers (*(slot, buffer)* pairs)**
  - **Function of VC and a reference buffer**
  - **One LN owns 1/N_{LN} bandwidth**
- **Slot allocation becomes VC-to-LN assignment:**
  1. **Slot partitioning to create LNs referring to a shared buffer**
  2. **Slot mapping to assign VCs to different LNs**
  3. **Slot refinement to allocate enough BW to LNs**

How to generalize the results?

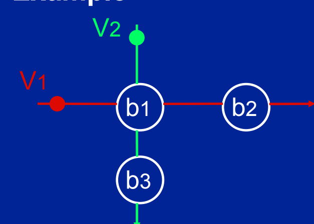
Essential issues

- **How many LNs exist when VCs overlap?**
  - **How to partition slots?**
- **Is allocating VCs to different LNs sufficient and necessary?**
- **How to select a reference buffer when VCs have multiple shared buffers?**
- **Does the result change if a different reference buffer is selected?**

Addressed by the contention-free theory

The number of LNs

- **Two VCs, V_1 and V_2, with admission window W_1 and W_2, the max. number of LNs, N_{LN} (V_1, V_2)=GCD (W_1, W_2)**
  - **W_1 and W_2 are derivable from BW requirement and subject to application constraints**
  - **More than one solution, reflecting design space**
- **Example**



V_1=<b1, b2>, BW_1=1/2

V_2=<b1, b3>, BW_2=1/4

⇩

W_1=2, W_2=4

N_{LN} (V_1, V_2)=GCD (2, 4)=2

⇩

b1 0 1 2 3 4 5 6 7 8 9 10

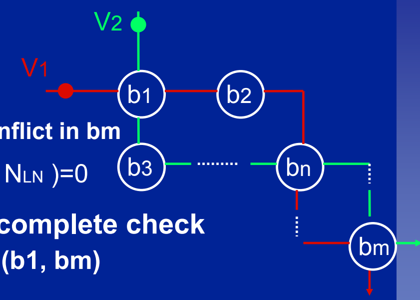
Sufficient and necessary condition

- VC-to-LN assignment steps:
  1. Slot partitioning to create LNs with respect to a reference buffer
  2. Slot mapping to assign VCs to different LNs
  3. Slot refinement
- Assigning VCs to different LNs is sufficient and necessary to promise contention-free.

Multiple shared buffers

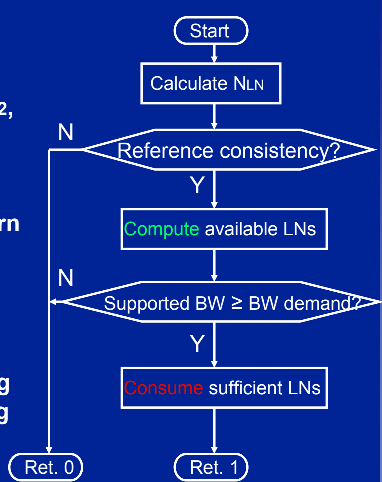
- If two VCs have multiple shared buffers, how to select the reference buffer?
- Example
  - $V_1 \cap V_2 = \{b1, bn, bm\}$
- Consistency check
  - No conflict in bn => No conflict in bm

  $\mathrm{mod}(\, d_{bnbm}(v1) - d_{bnbm}(V2),\ N_{LN}\,)=0$
- Linear check instead of complete check
  - (b1,bn) and (bn, bm) => (b1, bm)



LN-oriented slot allocation

- Input
  - VC spec. set, $V_1, V_2, \ldots V_n$, with BW and path known
  - Admission windows, $W_1, W_2, \ldots, W_n$, respectively
- Output
  - Fail or succeed
  - If succeed, admission pattern for each VC
- Slot allocation procedure
  - Pair-wise (v_i, v_j) and incremental
  - Compute LNs: slot partitioning and LN mapping
  - Consume LNs: slot mapping and refinement

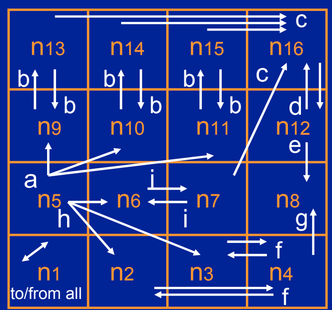


VC configuration program

- The LN-based slot allocation method has been implemented in our VC configuration program
- The VC configuration program
  - supports both open-ended and closed-loop VCs
  - explores the network path diversity via back-tracking

An industrial application

- A radio system
  - **26 node-to-node traffic flows classified into 9 multi-node flows. 'a' and 'h' are multicast, others unicast.**
- **Implement flows on a 4x4 mesh with closed-loop VCs**

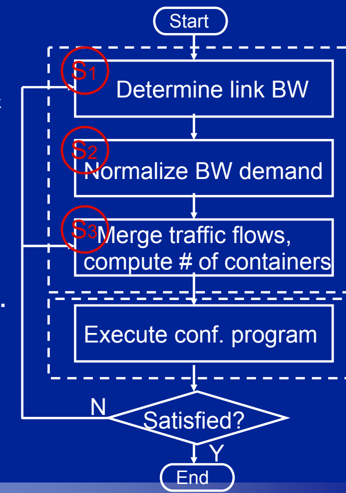


Flow	Num.	BW
a	3	4096
b	6	512
c	4	512
d	2	2048
e	1	512
f	4	128
g	1	64
h	3	4096
i	2	512

BW unit: Mbits/s

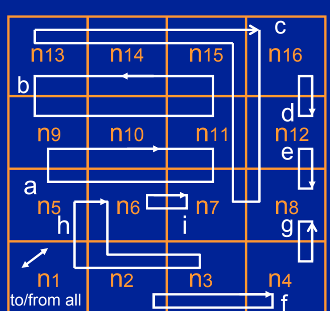
The case study

- **VC specification**
  - **S1: find a heaviest load link**
  - **S2: e.g. if BWlink=4096, 512=1/8BWlink**
  - **S3: explore multi-node VC for**
    - **Multicast, 'a' and 'h'**
    - **Low bandwidth, 'f'**
    - **Round-trip, 'd'**
  - **S3': only for closed-loop VCs, compute # of containers for each.**
- **VC configuration**



Results

- **VC implementations**
  - **one VC for one type of flow**
  - **For Illustration, not optimal**



- **VC conf. program execution experimenting on the impact of VC sorting, exploring all possible paths**
  - **Sort 1: random**
  - **Sort 2: higher BW first**
  - **Sort 3: less number of path options first**

Sort scheme	1	2	3
# of solutions	33	30	76
Exe. time (s)	6	6	12

Conclusion and future work

- **Conclusion**
  - **Slot allocation can be formally conducted with sufficient and necessary conditions.**
  - **Logical network is a powerful concept to ensure correct-by-construction.**
- **Future work**
  - **Optimize admission patterns to improve slot allocation**
  - **Asynchronous communication**
    - **Asynchronous links, nodes with different notion of time**
    - **Stallable packet delivery**
  - **Merging of VCs into LNs**
  - **Merging of Flows into VCs**

Thank you for your attention!