ClickINC: In-network Computing as a Service in Heterogeneous Programmable Data-center Networks

Wenquan Xu, Zijian Zhang, Yong Feng, **Haoyu Song**, Zhikang Chen, Wenfei Wu, Guyue Liu, Bin Liu

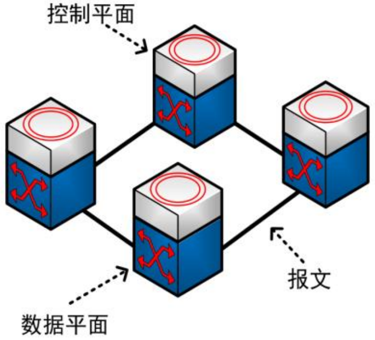
Caveats

- INC in academia != COIN in IETF
- Mostly boring details except the motivation and high-level ideas

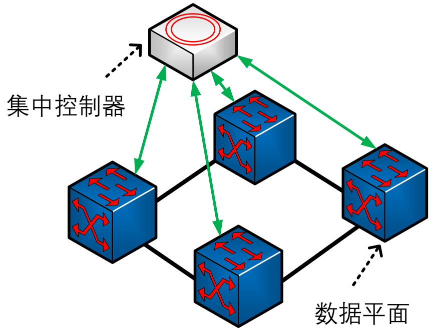
The Evolution of Networking

Network:
a dumb pipe

Fixed-function
switch



SDN
switch



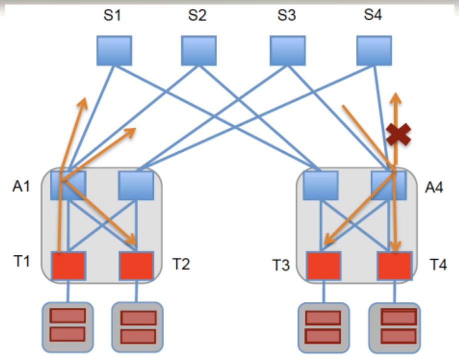
Programmable ASIC,
FGPA, NP, SmartNIC …

Data plane
programmable switch

- Data plane flexibility
- Memory + calculation
- Line-rate packet processing

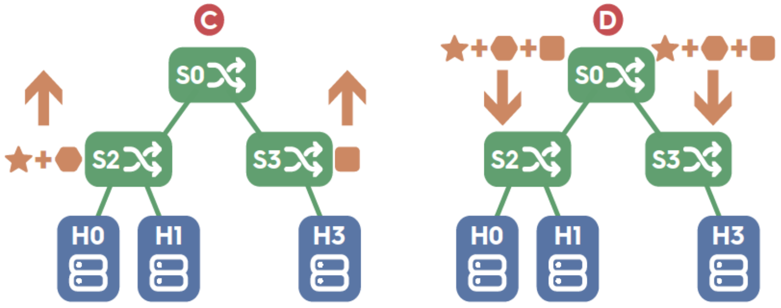
Can network
help with computation?

Prevalent INC Applications

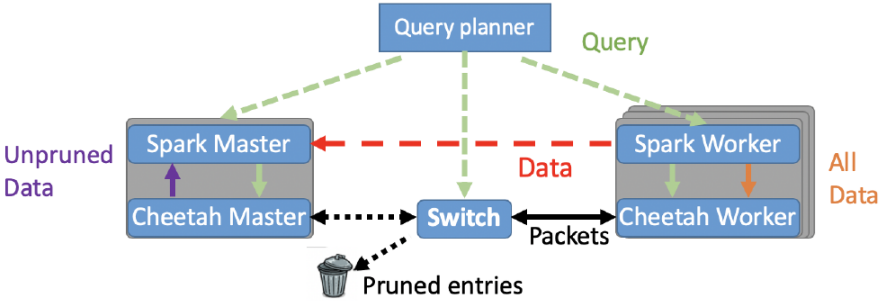


Key-value store[1]

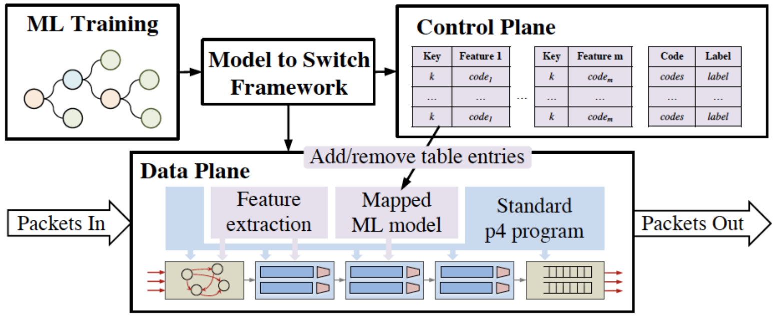
DDoS attack defense[2]



ML parameter aggregation[3]



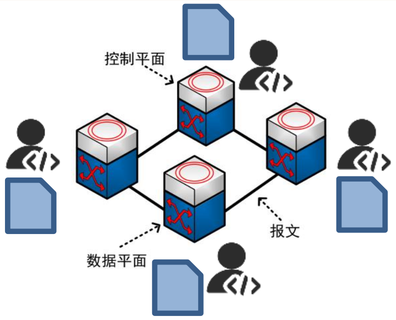
SQL query acceleration[4]

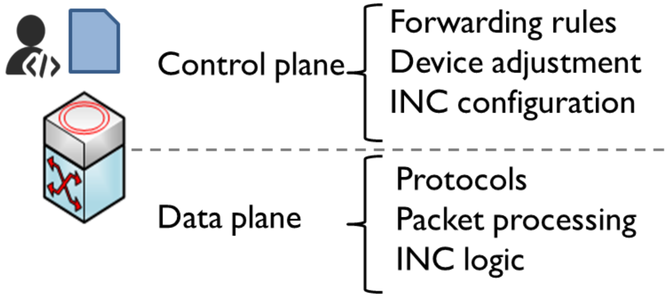


ML model inference[5]

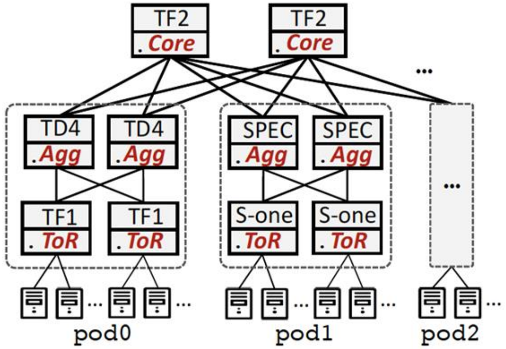
[1] Jin, Xin, et al. "Netcache: Balancing key-value stores with fast in-network caching", *SOSP '17*.
[2] Zhang, Menghao, et al. "Poseidon: Mitigating volumetric ddos attacks with programmable switches", NDSS '20.
[3] Lao, ChonLam, et al. "ATP: In-network aggregation for multi-tenant learning", NSDI '21.
[4] Tirmazi, Muhammad, et al. "Cheetah: Accelerating database queries with switch pruning", SIGMOD '20.
[5] Swamy, Tushar, et al. "Taurus: a data plane architecture for per-packet ML", *ASPLOS '22*.

Problems of INC




Control plane
Forwarding rules
Device adjustment
INC configuration

Data plane
Protocols
Packet processing
INC logic



1. Developers need to program their own INC from scratch:
   - INC is strongly coupled to the devices, hard to generalize
   - Different apps have different performance demands & data characteristics
   - Hard to reuse

2. <u>INC developer</u> needs also be the <u>network operator</u> :
   - Needs to develop a complete forwarding/processing program
   - Takes care of network details:
     - Packet parsing
     - Protocol handling
     - Correctness of forwarding rules
   - Closely involved in the deployment and network operation

3. INC development is challenging:
   - Heterogeneous devices: architecture, resources, language
   - Complex topology: especially when multiple paths are supported
   - Device resource and capability limitations:

Related Works

IPDK [Intel]
DOCA [Nvidia]

- Y: unify program interface
- N: cross-device program orchestration

Flightplan [NSDI’ 20]

- Y: program orchestration on heter. devices
- N: automatic program partition

Lyra [SIGCOMM’ 20]

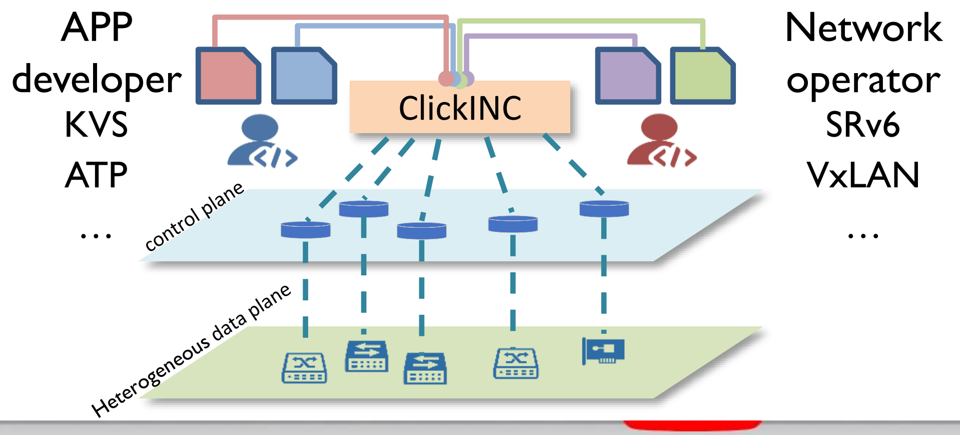
- Y: automatic program partition
- N: large-scale deployment, smartNICs

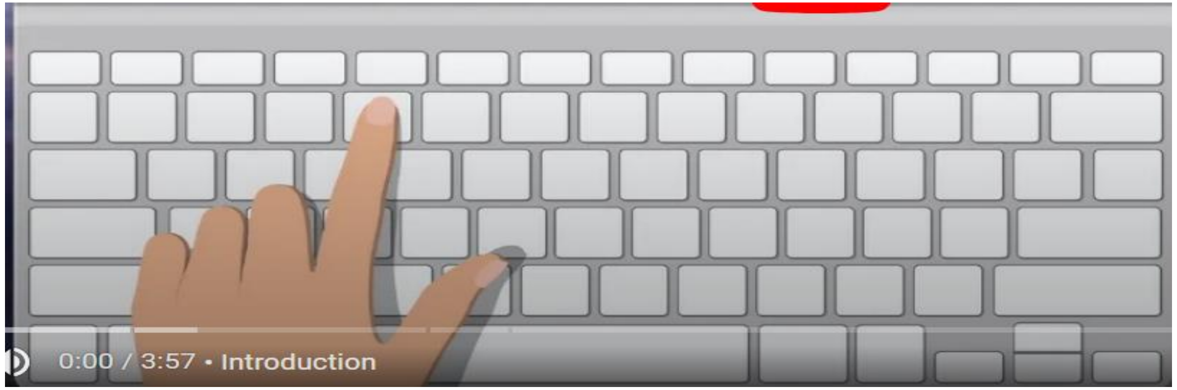
- For network operator
- Monolithic program: limit to single user
- Low-level abstraction
- Small-scale deployment

ClickINC [SIGCOMM’ 23]

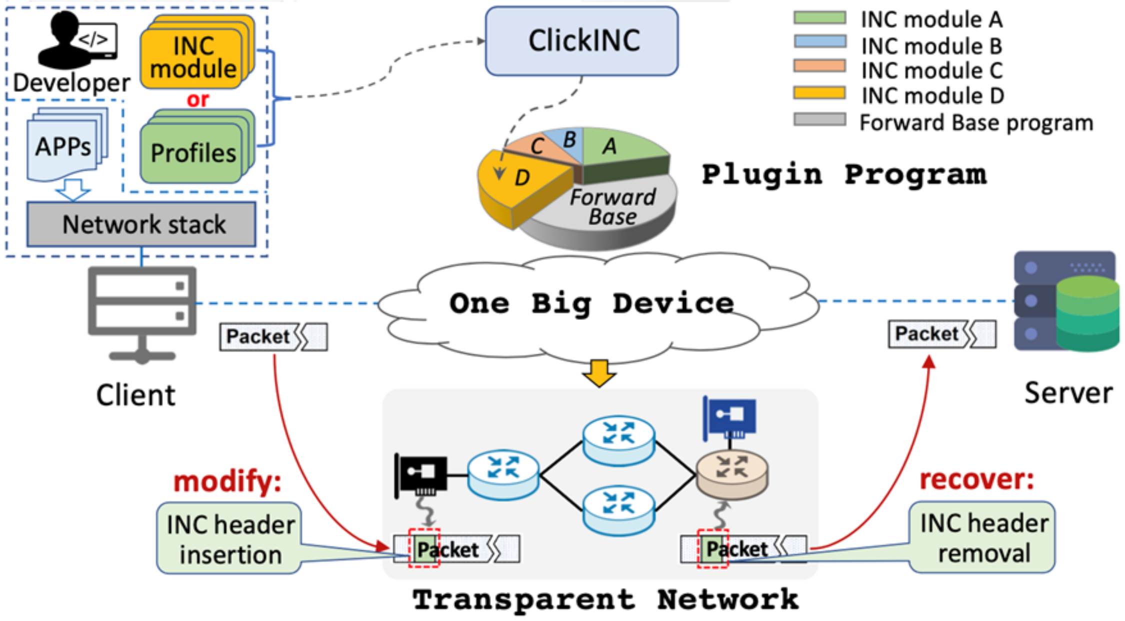
- Decouples network operation with INC developing
- Isolates development of different INCs
- Automates incremental deployment and support large-scale scenario

INC as a Service





With the goal of deploying INC in one click

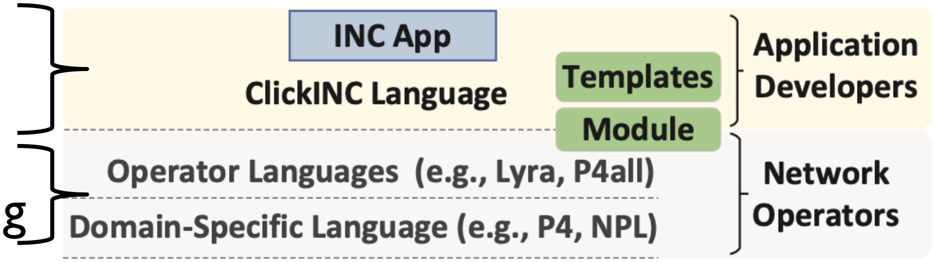


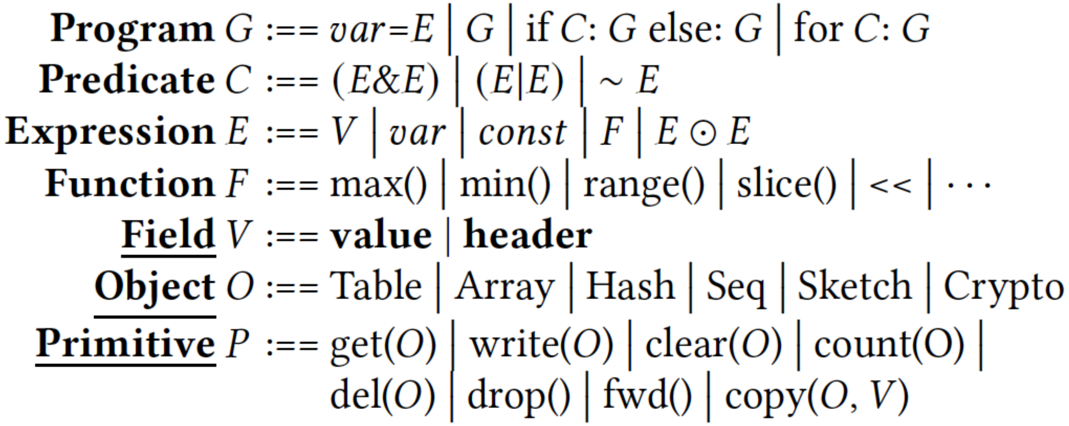
- Unified data plane: OBD
- Conceal device, topology, language, etc.
- Decouple network and INC
- INC program as plugins

Programming API

User Programming:

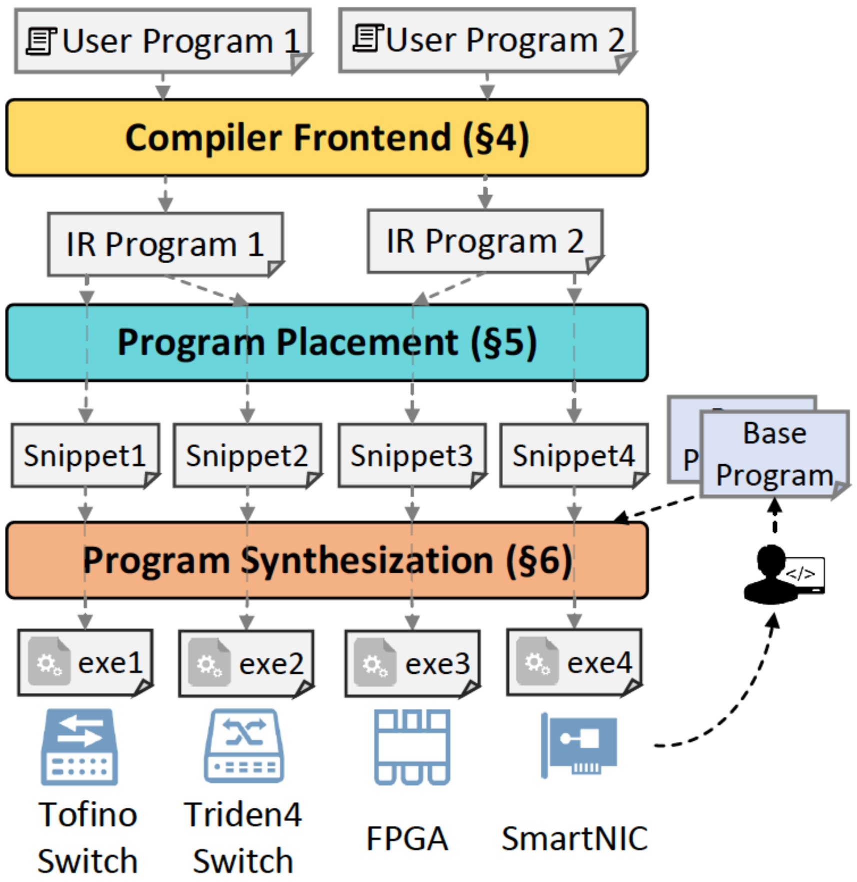
- Python-like language
- Three modes:
  - Template configuration
  - Modular programming
  - Advanced user-define programming


INC App
ClickINC Language
Templates
Module
Application Developers
Operator Languages (e.g., Lyra, P4all)
Domain-Specific Language (e.g., P4, NPL)
Network Operators



Program G :== var=E | G | if C: G else: G | for C: G
Predicate C :== $(E\&E)$ | $(E|E)$ | $\sim E$
Expression E :== V | var | $const$ | F | $E \odot E$
Function F :== max() | min() | range() | slice() | << | $\cdots$
Field V :== **value** | **header**
Object O :== Table | Array | Hash | Seq | Sketch | Crypto
Primitive P :== get(O) | write(O) | clear(O) | count(O) |
del(O) | drop() | fwd() | copy(O, V)

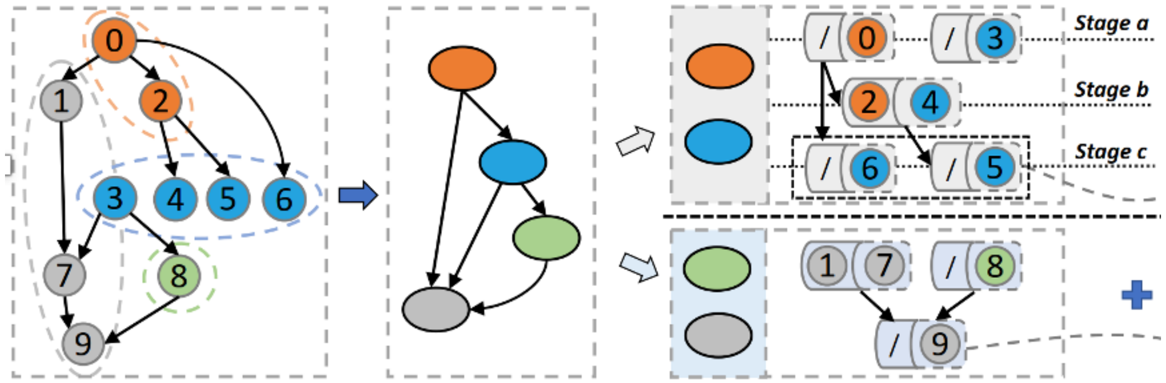
ClickINC Architecture

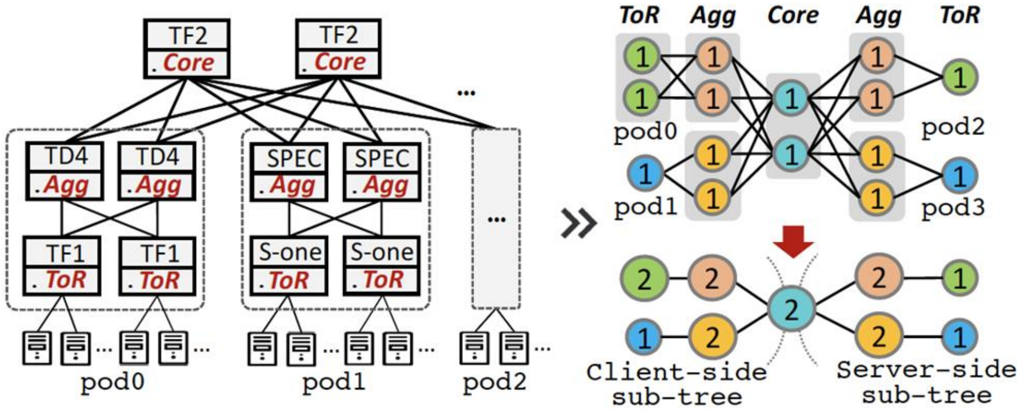


User Program 1
User Program 2
Compiler Frontend (§4)
IR Program 1
IR Program 2
Program Placement (§5)
Snippet1
Snippet2
Snippet3
Snippet4
Base Program
Program Synthesization (§6)
exe1
exe2
exe3
exe4
Tofino Switch
Triden4 Switch
FPGA
SmartNIC

- **Compiler**:
  - Convert program to IR
  - Translate IR to target platform
- **Allocator**:
  - Place program on devices
- **Manager**:
  - Merge INC into main program
  - Remove INC program
  - Update resources

INC Program Placement





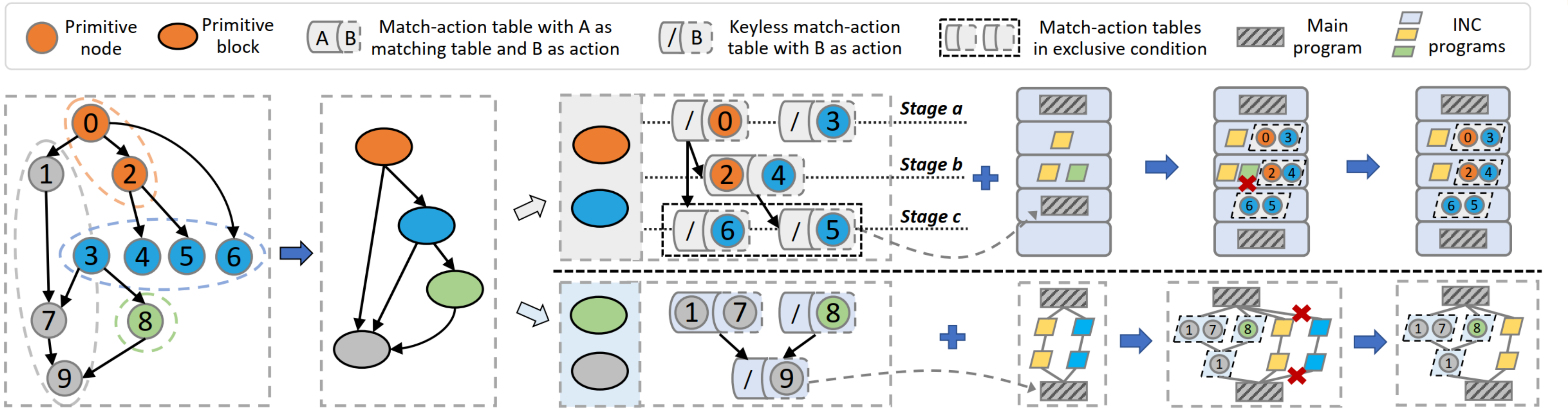
Map a program to pipeline stage/cores on one or more devices:

- Problem modeling: ILP or SMT
  - Constraints: 1) topology; 2) hardware limitation; 3) resource size
  - Targets: 1) serving traffic; 2) occupied resources; 3) across-device overhead
- Current solutions: ILP solver (P4all), SMT solver (Lyra)
  - Inapplicable in large-scale scenario and for multi-path traffic

Pruned-based Dynamic programming:

- Simplifying topology for fat-tree and leaf-spine topology
- Dynamic programming on each sub-tree, and combine the solutions
- Pruning method: reduce searching space

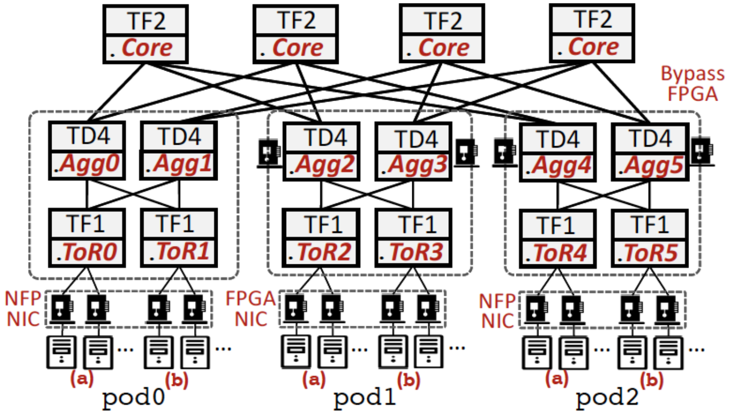
INC Program Management


Primitive node
Primitive block
A B Match-action table with A as matching table and B as action
/ B Keyless match-action table with B as action
Match-action tables in exclusive condition
Main program
INC programs
0
1
2
3
4
5
6
7
8
9
Stage a
Stage b
Stage c

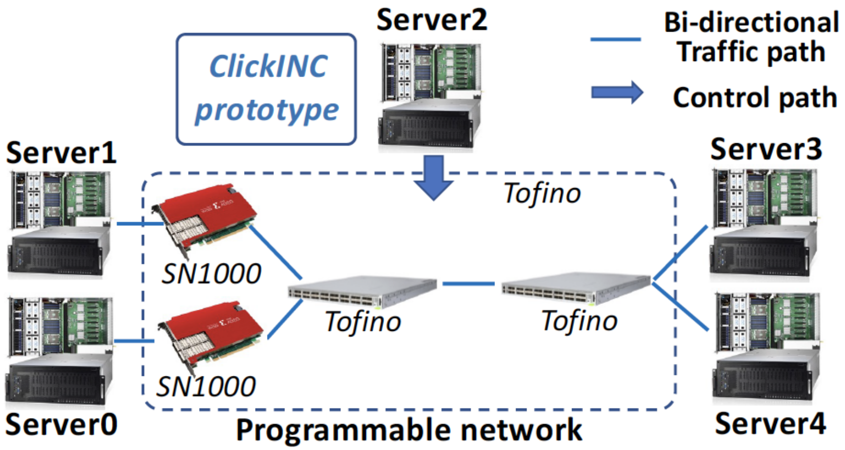
(a) Primitive-DG (b) Block-DG (c) Allocation in match-action (d) Program loading (e) Program removal

- Merge INC to the main program:
  - Graph based method
  - Add annotations
- Remove INC:
  - By annotations

Emulator & testbed

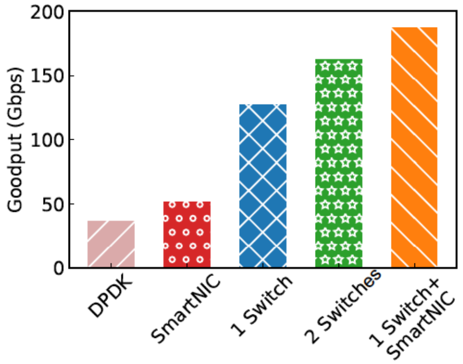


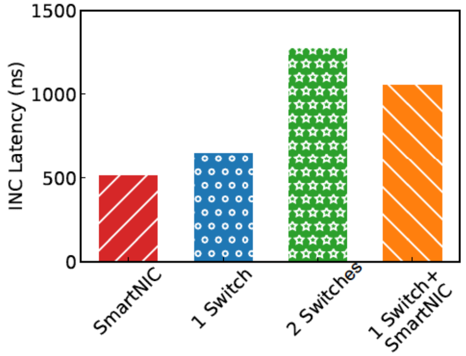
- Tofino: bf-sde
- Trident4: BCM-SIM
- FPGA/NFP smartNIC: behavior model



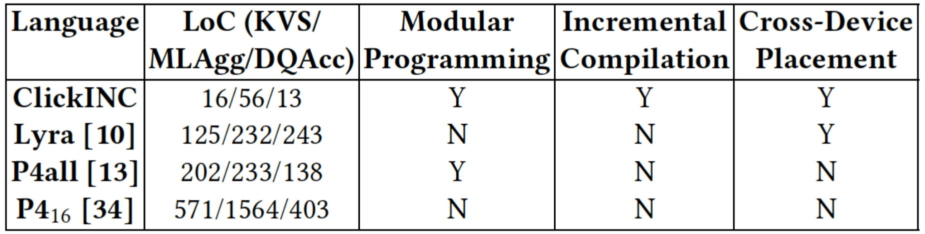
- DPDK server
- Tofino switches
- SN1000: FPGA smartNIC

Experiment Results



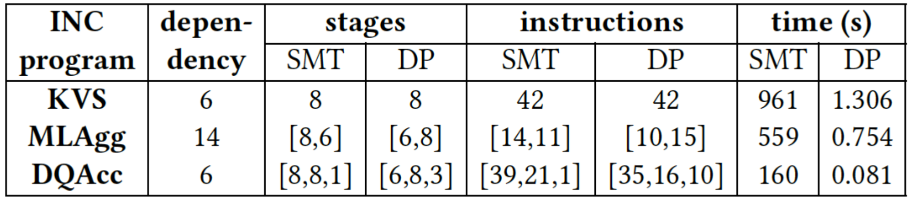


ClickINC makes use of resources on heterogeneous and multiple devices



Language	LoC (KVS/ MLAgg/DQAcc)	Modular Programming	Incremental Compilation	Cross-Device Placement
ClickINC	16/56/13	Y	Y	Y
Lyra [10]	125/232/243	N	N	Y
P4all [13]	202/233/138	Y	N	N
$P4_{16}$ [34]	571/1564/403	N	N	N

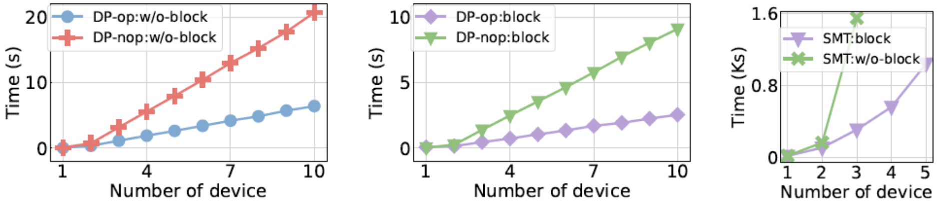
Modular programming abstraction allows more efficient INC development



INC program	depen-dency	stages		instructions		time (s)	
		SMT	DP	SMT	DP	SMT	DP
KVS	6	8	8	42	42	961	1.306
MLAgg	14	[8,6]	[6,8]	[14,11]	[10,15]	559	0.754
DQAcc	6	[8,8,1]	[6,8,3]	[39,21,1]	[35,16,10]	160	0.081

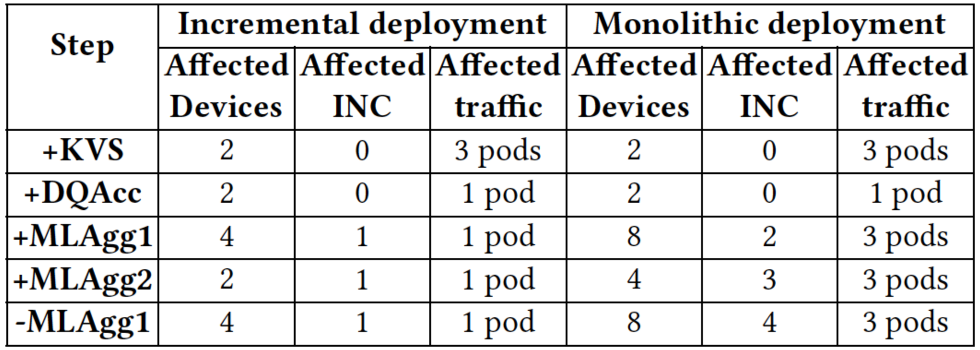
Less placement time with equal optimality solution compared to SMT solver

Experiment Results



(a) DP: w/o-Block denotes no block construction (b) DP: with block construction (nop: no pruning) (c) SMT

Strong scalability to the number of devices



Step	Incremental deployment			Monolithic deployment		
	Affected Devices	Affected INC	Affected traffic	Affected Devices	Affected INC	Affected traffic
+KVS	2	0	3 pods	2	0	3 pods
+DQAcc	2	0	1 pod	2	0	1 pod
+MLAgg1	4	1	1 pod	8	2	3 pods
+MLAgg2	2	1	1 pod	4	3	3 pods
-MLAgg1	4	1	1 pod	8	4	3 pods

'+' or '-' mean to merge or remove an INC program.

Incremental deployment has less affected traffic, device and other INCs

Conclusion

Contributions:

1. Propose the concept of INC as a Service
2. Top-down framework for program developing and deploying: One big INC abstractions, agile programming model, efficient program placement algorithm.
3. Heterogeneous-device emulation and testbed

On-going researches:

1. Automatically set parameters for user-written programs
2. Expand the template for common INC applications
3. More kinds of programmable network devices will be supported
4. Placement algorithm will be improved to support more complex topology and traffic scenarios