Dagstuhl Seminar: Compute-First Networking
Report for IRTF COINRG

Teaser

Organizers: Jon Crowcroft, Phil Eardley, Dirk Kutscher, Eve Schooler
2021-07-26

https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=21243
Objectives for this Seminar
Definition and Research Agenda for "Compute-First Networking"

• Is there a space for a new approach of integrating computing and networking?
  • Beyond packet flow processing
  • Beyond microservice overlays
  • Learning from distributed computing systems (and taxonomy of those)
• What are hard problems today?
• What are promising new approaches?
• Is there a critical mass of research?
• What are the research challenges?
Successful Example in Distributed Computing

Dataflow

Apache Beam & Cloud Dataflow
Apache Beam is a collection of SDKs for building streaming data processing pipelines. (Beam = Batch & Stream)

Cloud Dataflow is a fully managed (no-ops) and integrated service for executing optimized parallelized data processing pipelines.

Apache Flink

How does Dataflow work?

Data Source

Data Sink

PCollection 1

Transform

PCollection 2

Transform

PCollection 3

P = Parallel

PCollections are immutable

Google Dataflow
Example
Packet Interception vs. Dataflow

- Transparent middleboxes
- Transformations on packet flow
- Side effects

- Explicit transformations on input data
- Can have side effects
- But principally, focus is on result data
Dagstuhl: Compute-First Networking

https://www.dagstuhl.de/en/program/calendar/semhp/?semnr=21243
Program at a Glance

• Use Cases
  • Jag Minhas: Privacy Preservation in Edge Video Processing
  • Jon Crowcroft: CFN for Health
  • Naresh Nayak: Compute Centric Networking for Connected and Automated Driving

• Research Challenges
  • Dave Oran: Networking and Computing - the great confluence
  • Erik Nordmark: EVE harder Challenges

• Other topics
  • PhD topics for CFN
  • AI & ML opportunities and challenges

• Research Topics
  • Christian Tschudin: After the Fun the Hard Stuff
  • Gianni Antichi: In-network telemetry
  • Noa Zilbermann: in network computing challenges and opportunities
  • Andy Reid: Deconstructing and reimagining orchestration
  • Michio Honda: Networking matters for storage systems
  • Peer Stritzinger: Heterogenous Networks and Laying out the Compute Graph
  • Jörg Ott: In-Network Compute Architecture
  • Jianfei He: New Network for Distributed Systems
So, why should we care about this?

- Applications are becoming more multi-party and distributed
  - Difficult (and possibly undesirable) to make the network “transparent” to the application programmer
  - Performance inhomogeneities in both throughput and delay
  - Complex partial failures
- Programming model only easily exploits localized parallelism
- Isolation against competing workloads and resilience against attack requires sophisticated features “in” the network
- DevOps requires incremental partial deployment
  - Coordination with network underlays tricky and slows things down
  - Responsibilities for various security and disaster protection divided organizationally – partially due to expertise gap and technology differences
- Computing and Communications are on different cost/performance trajectories
Use Case: Health
Jon Crowcroft

Health Sensing
- On phones
  - Accelerometer, camera, mike, or addon
- or on bespoke devices
  - (pacemakers, insulin pumps, defibr etc)

Federation of Planet Health Data
- But cost of upload data
  - Battery, network charges etc
  - Privacy (blah blah:-)
- So learn model, and upload model params
  - Federated PCA, or SGD
  - Or even federated Bayes inference

Federation of Planet Contacts
- GAEN is a cryptic digital twin
  - phones load (BLE) contacts into the “net”
- If/when a phone’s owner test +ve
  - contacted phones “notified”

Federation of Planet Health Risk
- Risk - #people not tested of vaccinated
- Prevalence in an area
- Variants
- Location of test/vaccination centre
- etc
Use Case: Split Learning

Dave Oran

Simplified Edge computing model

Potential advantages of CFN to Split Learning

- **Privacy**
  - Confine raw data to device only, or
  - Store/process raw data in secure enclave on edge computing complex
    - Rely on channel security or ICN-style secure objects to hide raw data from eavesdroppers
    - If using channel security, separate encryption needed for local storage media
    - Process in security context of data owner/generator

- **Resource tiers**
  - Simple sensing on device
  - Feature extraction at edge (or on more capable device)
  - Full model execution in the cloud

- **Flexible processing options**
  - Limited model locally to solve latency issues
  - Bandwidth savings by only invoking full model when needed

- **Failure resiliency**
  - Degraded operation feasible when cloud unreachable
  - Local storage for time-series data – decouple from real time constraints when learning
Facts
Dave Oran

Summing up – Servers versus Switches

Servers
- Many cycles/bit
- Memory intensive
  - Either lots of state or high creation/destruction rate
- Scalable load
- Rapid feature evolution
- Need isolation / multi-tenant

Switches
- Few cycles/bit
- Small/moderate memory
  - But run at clock rate w/o caches
- Need to process input at wire rate
- Simple, “inner loops”
- Works if crypto not an issue

Dagstuhl CFN 2021
Facts
Noa Zilbermann
Programmable Dataplane as a Platform

Summary: Opportunities

- Open end-to-end switch architecture
- A good model for resource-constrained programming
- Multi-target compiler
- State saving and state sharing
- Programmable externs
- Data plane debuggers
- End-to-end performance
- Failure recovery
- Multi-tenancy

Micro-Architecture

- You can save state.
  - Once.
- You can access state.
  - Once.
  - In the same stage.
- And we won’t make it easy.

- Sharing state between pipelines?
- Pipelines????
Text - What do we need to make this work?

- Intelligent placement of computing
  - Joint optimization of network resources and computing resources
  - Visibility into network state/metrics by the application programmer (or at least in the framework)

- Lay out processing graphs flexibly — react to medium-timescale changes
  - Conditions may change dynamically and constantly: network to adapt to application requirements, network conditions etc.

- Sometimes we can move functions instead of data (close to big data assets)

- At other times we gradually move data where it is needed (e.g., where specific computations run)

- **Optimization based on application requirements & view of all relevant resources**
Some interesting outstanding questions

- Smart NICs have FPGAs — what’s the best way to use them?
- Figure out how to use P4 on switches for general computing?
- How to bridge the gap in the programming model?
  - What is imperative/functional versus what is done data-flow
- What do the platforms look like?
  - Heterogenous elements closely coupled internally, with conventional network externally, or
  - Heterogenous elements with custom "internal" network built scale-out, with conventional network connecting the complexes, or
  - Some hybrid with multiple parallel interconnects
    - Note: Microsoft tried this with FPGA's to scale Bing search
Re-Imagining Orchestration

Andy Reid

Cycles of deconstruction and reimagining

Replaced by

Cloud technology

Replace horse with internal combustion engine

Traditional network with dedicated boxes

Retains many features of horse-drawn carriage
- Chauffeur = carriage driver/footman
- Arrangement of chauffeur and passengers
- NFVO+VNFM etc = Management plane
  - Load balancer+firewall+NAT = Data plane

Control plane

Automation without the legacy familiarity

Reimagine ‘automobile’ afresh
Leveraging In-Network Telemetry

Gianni Antichi

This is INT (In-band Network Telemetry)

Tell me more about my packets

- Is there a congestion/queue buildup?
- Do routing paths agree with the policy?
- Does my network experience load imbalance?

INT as building block for many network functionalities

- INT is deployed in major companies such as Alibaba
- Used in a number of contexts:
  - Congestion Control (HPCC – SIGCOMM 2019)
  - Load Balancing (Clove – CoNEXT 2017)
  - Path Tracing
Rethinking Communication and Computation Abstractions

Christian Tschudin

Wished it was a Broadcast-Only World

“Receivers are an afterthought - it all starts with the event source”

- Source = secure single-author event stream (e.g. signed hash chain)
- then replication kicks in (any “ether” works)
- eventually and potentially, all parties*) can see all events (observers react to events, become a source themselves)

Computer Networking = coping with “ether” scarcity (BW and mem limitations)

“Distributed Memory Side Effects”

I suggest an architecture based on:
- enabling explicit (programmable) storage side effects for in-network memory
- scheduling of side effects (not covered here, see chem-based protocols)

Some first principles why the effect on mem is important:

- Persisted state is the only valid anchor in distr systems (also think WAL)
- Abstraction: All ways of doing storage side effects are valid:
  - kick remote state with packets - but don’t stop there
  - sneaker nets (USB sticks), DNA phial, “a van full of tapes”, opportunistic

Local (!) operations: running code has to read scent marks, leaves scent marks
Provisional Summary

• **Trends in hardware development**
  - limitations of multi-core
  - evolution of specialized hardware-support
  - programmable data plane

• **Trends in distributed application design**
  - Distributed ML (federated ML, split learning)
  - Applications increasingly distributed multi-party and distributed

• **Successful distributed overlay systems**
  - Ray (Berkeley RISE)
  - Dataflow-inspired (Beam, Flink)

• **Opportunities**
  - Programming abstractions and platforms that do not treat network as black box
  - Leveraging different types of hardware platforms optimally
  - Exploring optimization potential: joint optimization, less centralized designs
  - Rethinking role of management and orchestration