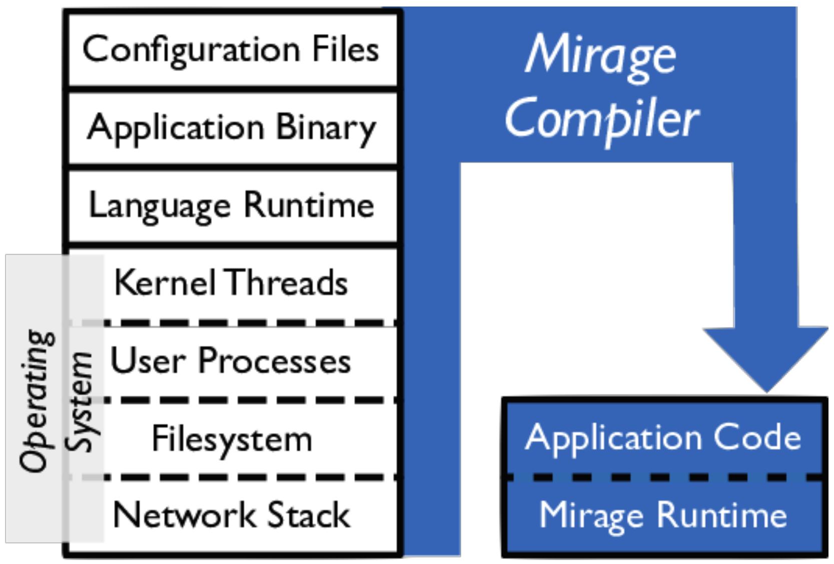
Compute-First Networking (CFN)

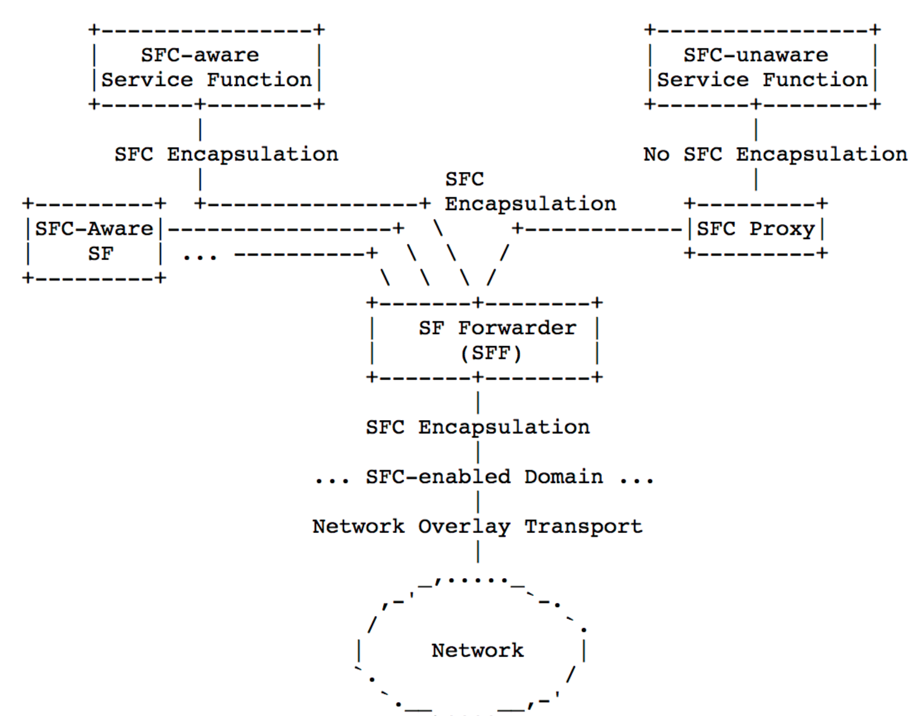
Dirk Kutscher

Inspired by Dave Oran, Jianfei He, Cedric Westphal, Lixia Zhang, Jeff Burke, Eve Schooler and many others

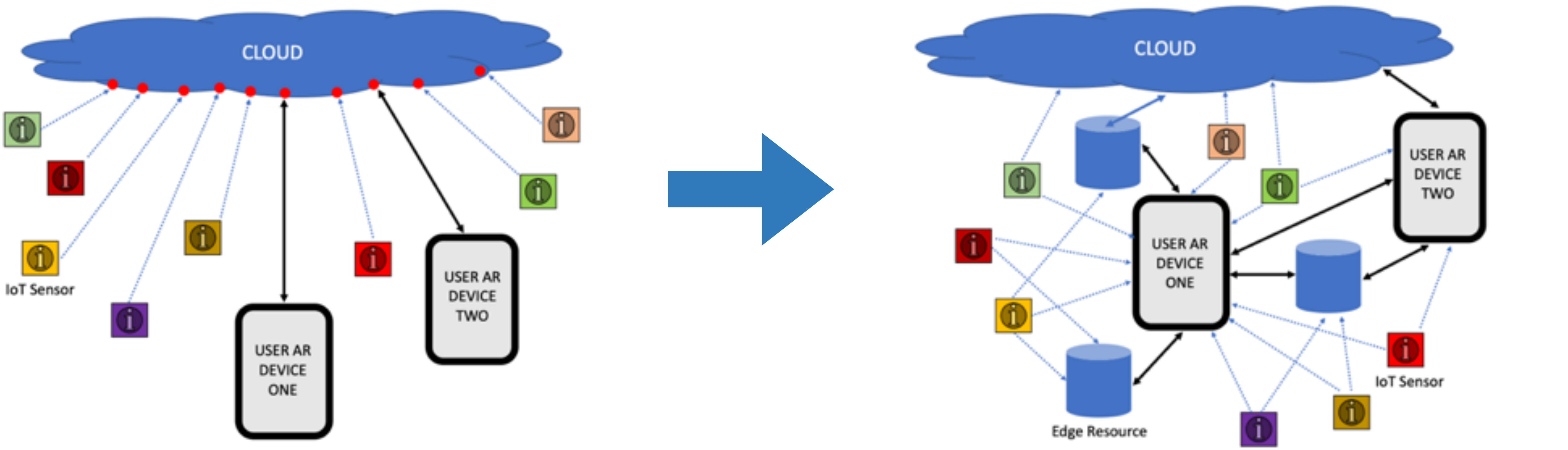
Cannot Leverage Computation in Networks Today

- **Significant advances in making computation available, affordable, programmable**
  - Virtualization: big leaps from host virtualisation to unikernels, lambda expression evaluation engines
  - Application layer frameworks for data processing, microservice architectures, virtualized network automation
- **Networking is lacking behind**
  - Connection-based communication and security model: cannot introduce computation without breaking security and introducing significant overhead
  - IP address-based communication: leads to static and difficult to manage networked computation (“service function chaining”) — not applicable to dynamic, mobile environments
  - No concept for computation on data plane: leads to complex orchestration and management frameworks



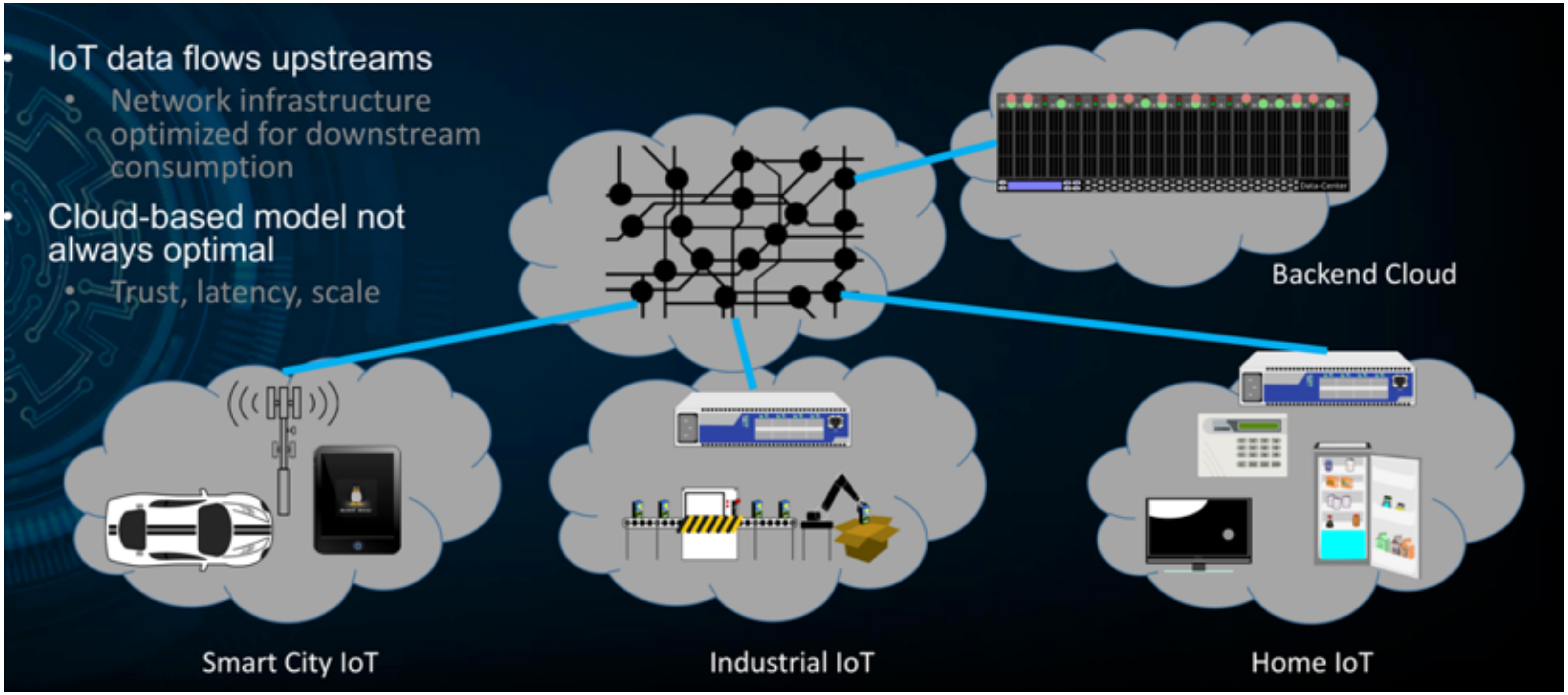


Extended AR



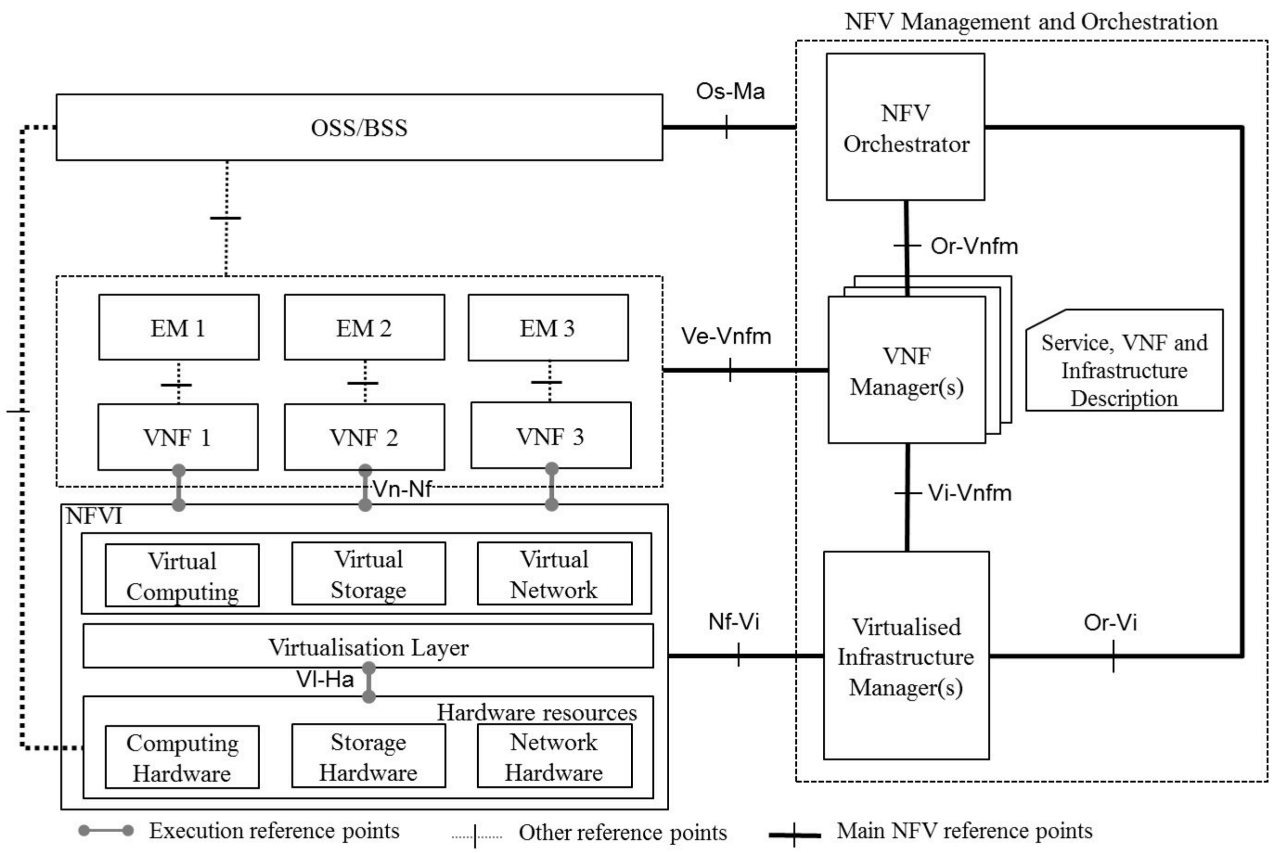
Lemuel Soh, Jeff Burke, Lixia Zhang; Supporting Augmented Reality (AR): Looking Beyond Performance; ACM SIGCOMM 2018 Workshop on Virtual Reality and Augmented Reality (VR/AR Network 2018)

Upstream Data Processing



Also cf. Srikathyayani Srikanteswara, Jeff Foerster, Eve Schooler:
ICN-WEN Information Centric-Networking in Wireless Edge Networks;
Presentation at ICNRG@IETF-98, March 2017
https://www.ietf.org/proceedings/98/slides/slides-98-icnrg-information-centric-networking-in-wireless-edge-networks-eve-schooler-00.pdf

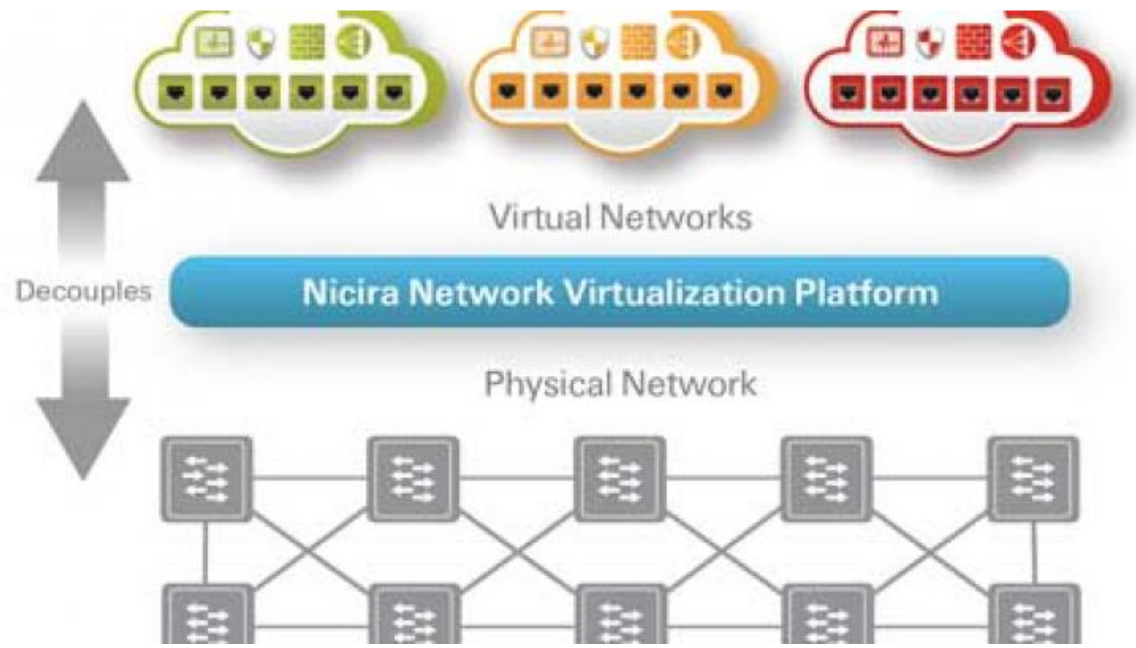
Different Perspectives on Compute & Networking

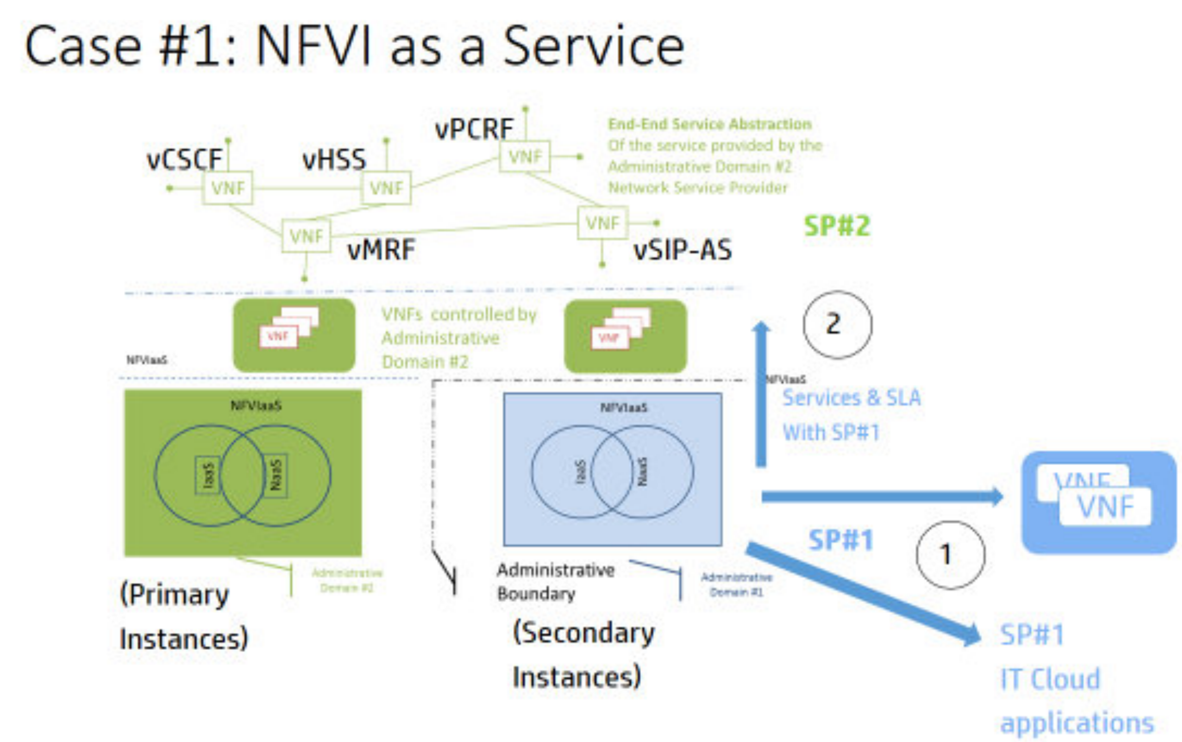


(Virtualized) Compute Servers in Networks

Networked Computations

Old-School In-Network-Computing





Data centers

- Virtual networks providing connectivity in private networks (per tenant/app)
- Workload migration and upscaling
- Networking has to follow server/VM location
- No joint optimization

Telco Core

- Virtual networks providing connectivity in private networks (per tenant/app)
- Workload migration and upscaling
- Networking has to follow server/VM location
- Connectivity dictated by telco function requirements
- Some manual optimization (co-location, chaining)
- No automatic joint optimization

Old-School In-Network-Computing

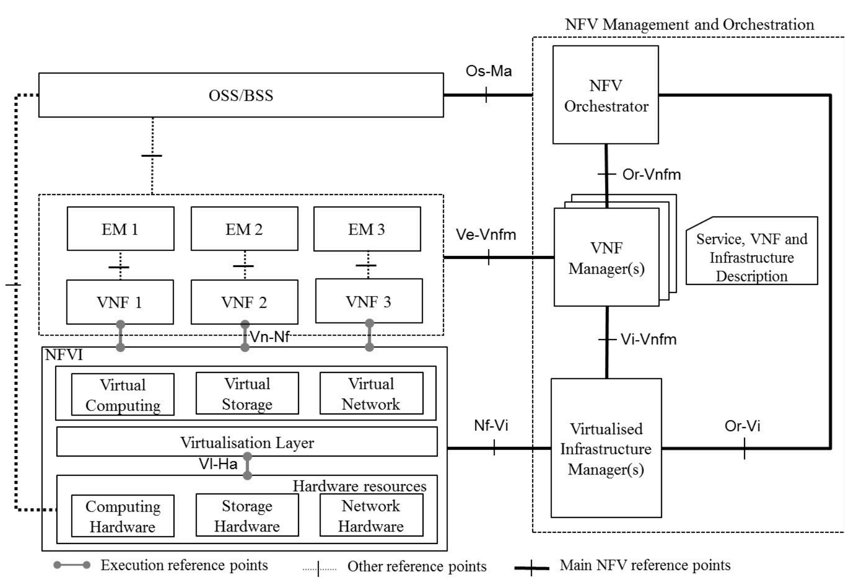
Computing & Networking — different worlds

- **Technically**
  - DC: Virtual servers instantiated by OpenStack — virtual network has to connect them
  - Mobile Edge: Application VM containers with overlay connection to cloud — overlay on telco tunnelling-based mobility management
- **Culturally & organizationally**
  - Application development: APIs treat Internet as local network — agnostic to topology, indirect performance consequences
  - Network just infrastructure

CFN: Joint Optimization of Computing and Networking

- Holistic resource management
  - Network capacity
  - Compute resources
  - Storage

- Multi-dimensional requirements/preferences sets
  - App developer
  - User
  - Network operator



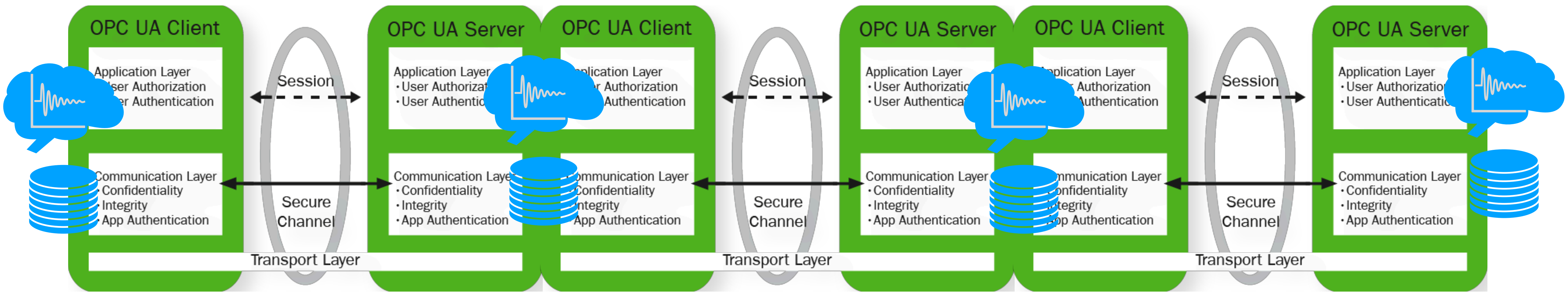
(Virtualized) Compute Servers in Networks

Networked Computations

Previous Work

- Compute servers (physical or virtual)
- Stream processing
- Microservice architectures
- RPC, CORBA
- Active Networking

In-Network Computing With Client-Server Protocols



- Overlays
  - Connection-based security
  - Client-server / broker-based
- Limited Scalability
  - Pub-sub distribution to many clients through single-server bottleneck
- Limited efficiency
  - Cannot share data directly
- Limited performance and robustness
  - Network cannot assist data dissemination
  - Compute cannot consider network conditions

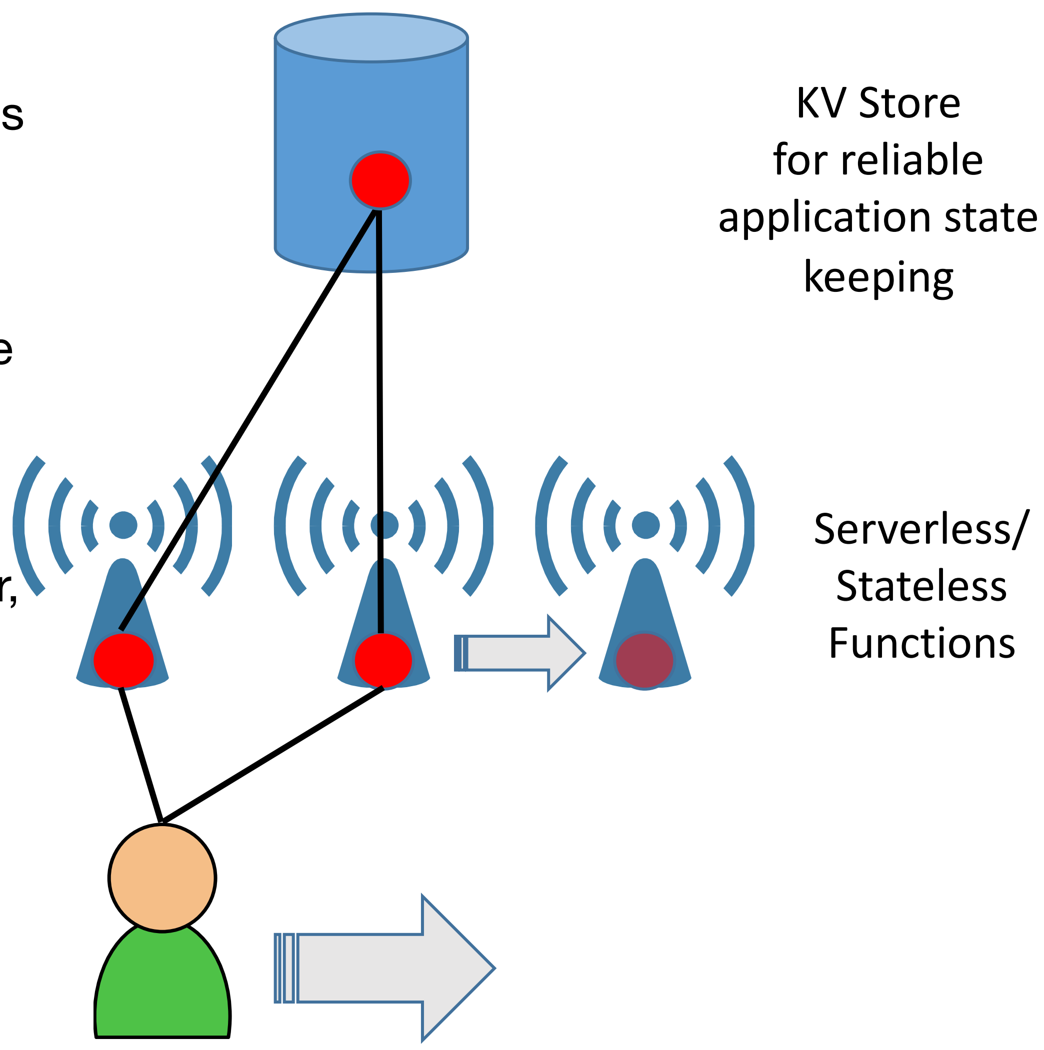
Adding a little computation to a data kiosk system is not exactly distributed computing

Joint Optimization of Networking and Computing Resources

- Historically, network was engineered to provide connectivity between compute servers
  - VLANs, tunnels, overlays
  - Network: circuit, pipes between computations
  - Assumption: servers, network functions are fixed – network has to adapt
- CFN perspective
  - Do not require fixed locations of data and computation
  - Can lay out processing graphs flexibly – meeting requirements optimally
  - Sometimes we can move functions (close to big data assets)
  - At other times we gradually move data where it is needed (e.g., where specific computations run)
  - Conditions may change dynamically and constantly: CFN network to adapt to application requirements, network conditions etc.

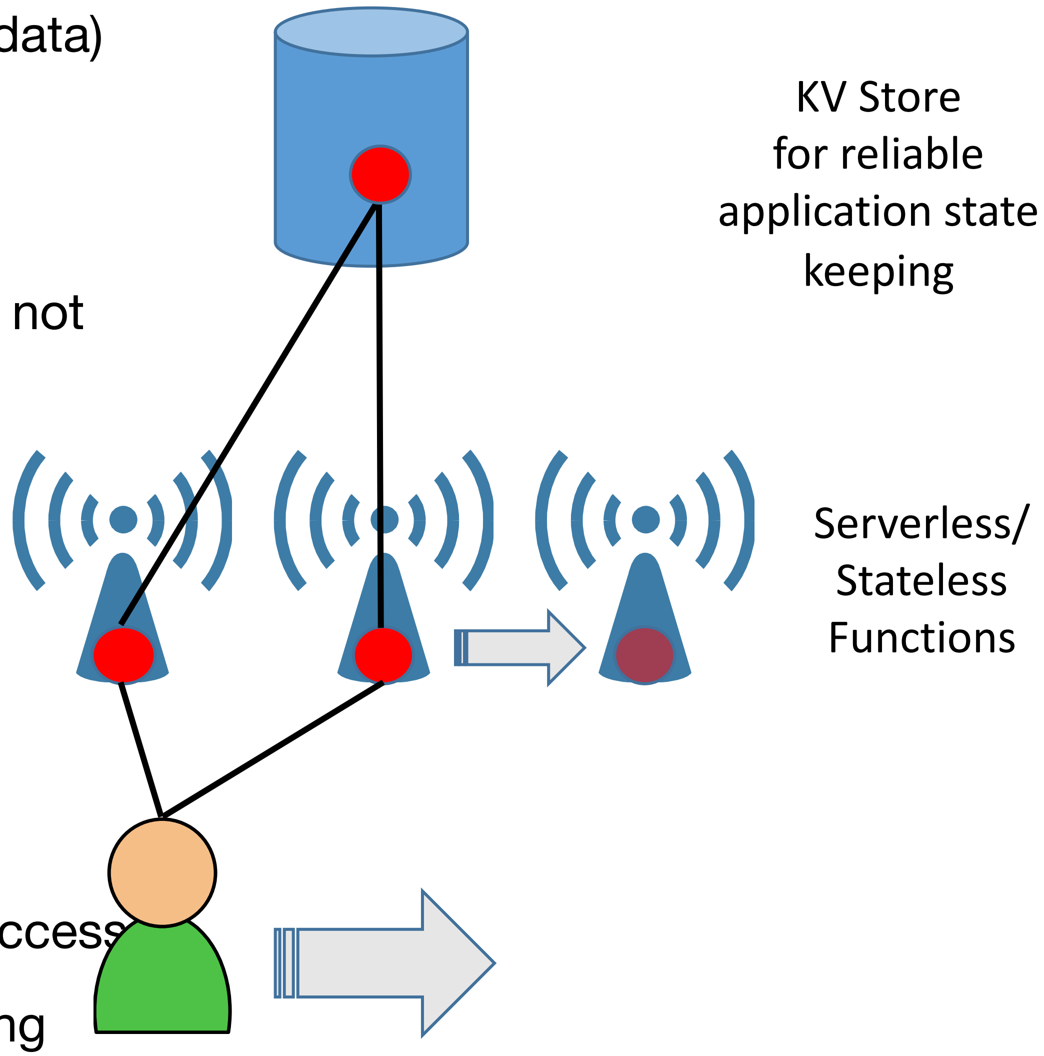
Serverless CFN

- **Serverless does not mean „no servers“**
    - It means decoupling the execution from specific server platforms
    - It also does not mean „no state“
    - It means application state lives independent of function instance
- **Powerful concept for CFN**
    - We can position stateless functions where needed (close to user, following the user etc.) – guaranteeing low latency, good scalability etc.
    - State can be kept somewhere else – in KV stores, in a synchronized set of app components
    - A new instance of a stateless function can access (and potentially modify) that state
    - Function instantiations would follow REST principles



Serverless CFN

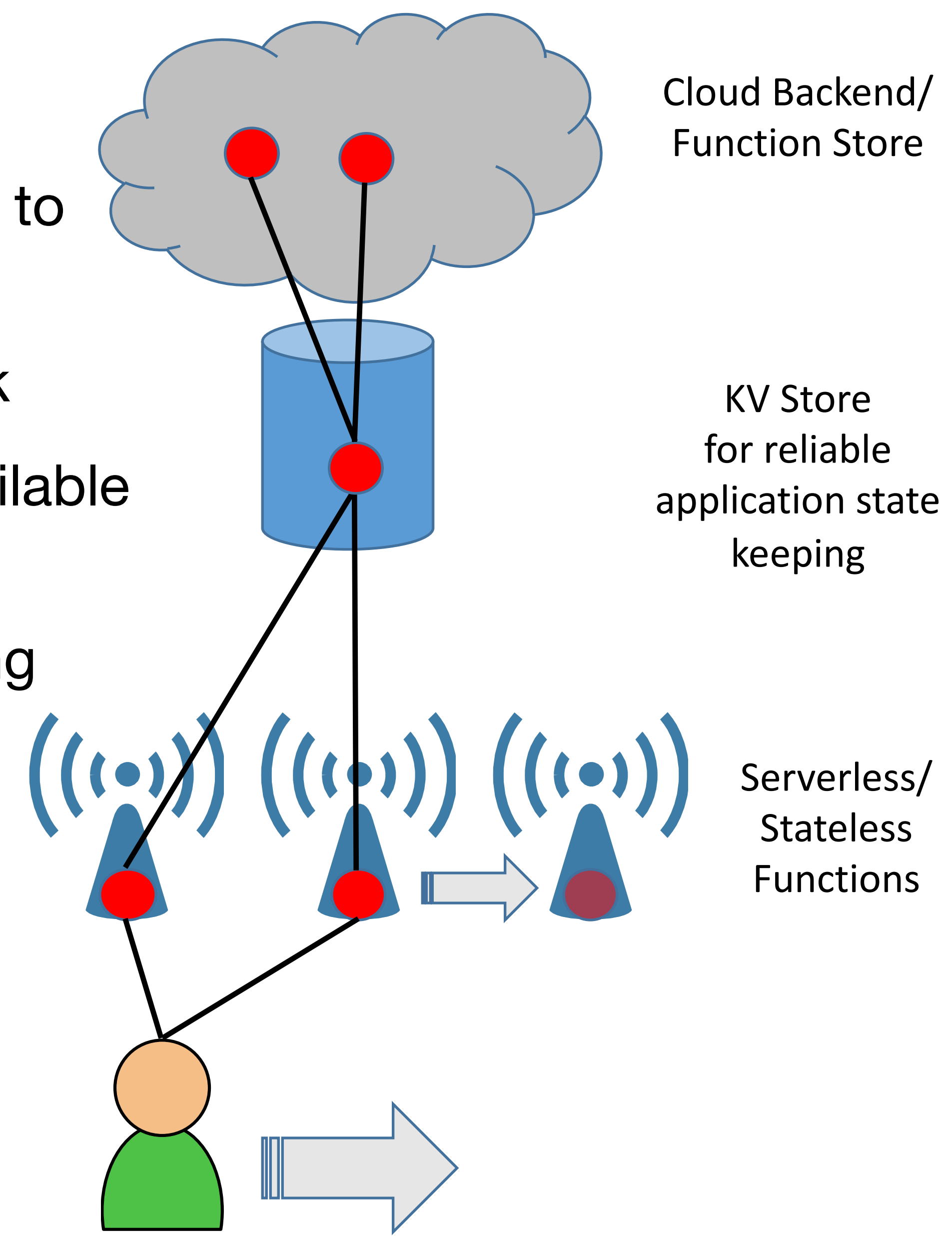
- **Serveless functions can follow users as needed**
  - Pro-active instantiation, even pro-active invoking (and result caching)
  - Pro-active resolution of dependencies (other functions, input data)
- **Session state can be kept independent**
  - But at convenient locations in the network
  - KV stores can be centralized DB or distributed system – does not matter
- **Function instances can be shared (invoked) by many users**
- **Some data (including computation results) can be shared by multiple users**
  - Overall CFN can optimize in several dimensions
  - Move stateless functions close to user
  - Have working set of relevant data available with low latency access
  - Improve throughput by cloning stateless functions and enabling parallel execution, seamless handover etc.
  - Pro-actively move functions/data in times of imminent network. disruptions



This is what we mean by joint optimization!

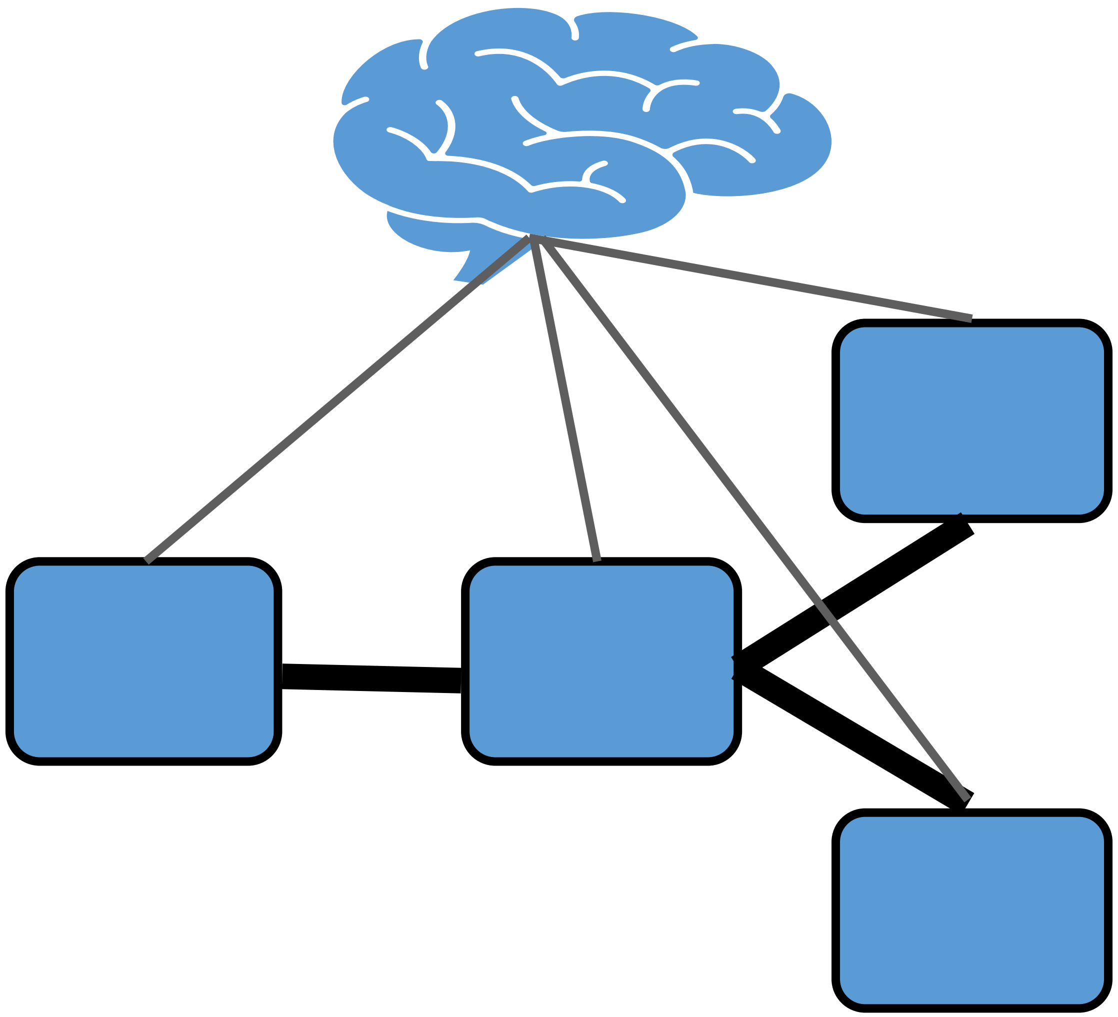
Data Logistics in CFN

- Constantly moving around session state and function parameters could be costly
- Yes, that‘s why CFN should have intrinsic mechanisms to support this
- Accessing data by name & caching data in the network
  - Frequently used data objects are automatically available close to user (or any functions that needs them)
  - Objects can be replicated as needed and forwarding can adapt (joint optimization...)
  - Transparent service to functions (i.e., they don‘t have to search for data etc.)
- Homoiconicity: function code is data
  - Same access principles and mechanisms
  - Same cacheability and efficiency gains



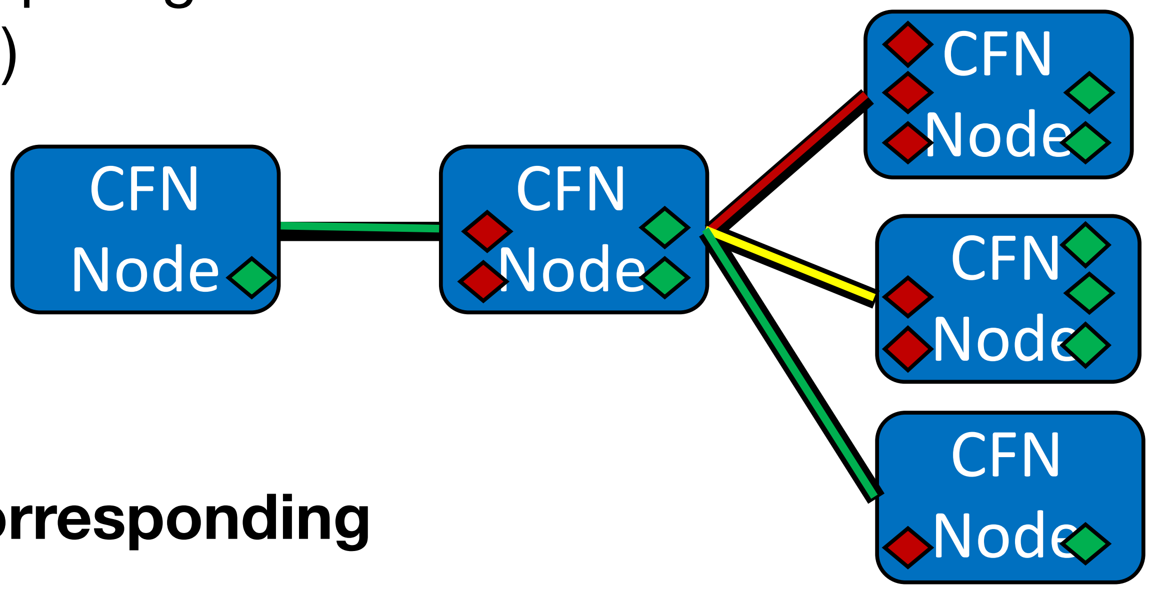
CFN Joint Optimization

- In a dynamic, multi-tenant system
- With unpredictable load on networks and compute notes
- Without being able to predict effects precisely
- Could be an NP-complete problem
- At least not likely to scale



CFN Empowered Data Plane

- **Concept of CFN Data Plane: empower data plane to support optimal app installation and function execution**
  - Control/feedback loops that consider computing and networking resources (joint optimization...)
  - Enable network to react to congestion, dynamic load by making smart forwarding decisions, restructuring forwarding graphs etc.
- **Does not exclude operator policies and corresponding configurations**
  - Need to find good balance between in-band control and orchestration
  - But don‘t build a system that requires orchestration for everything

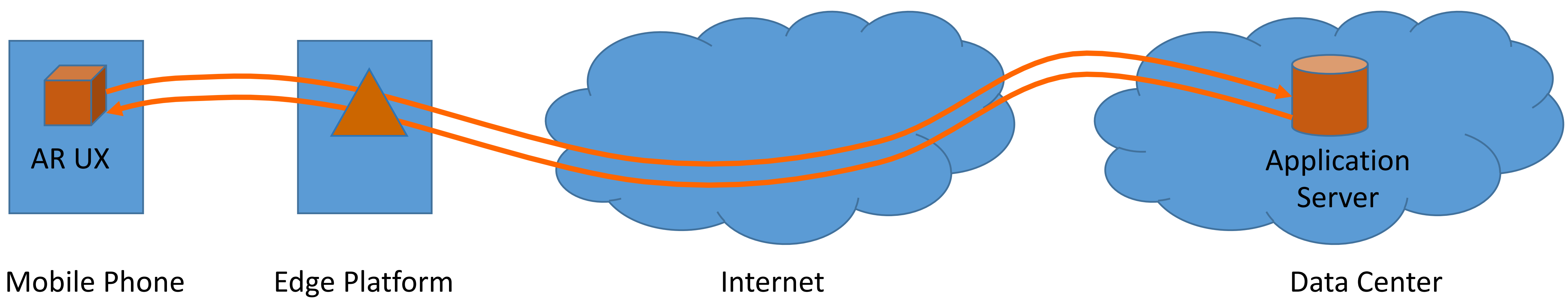


CFN Data Plane

Moving some functions from overlay (or app layer) to network layer

- Load balancing
  - Extend forwarder load-balancing for forwarding computation requests
  - Holistic view on load — server load *and* network load
- Failure resiliency
  - Routing state for multiple instances of a function in the network
  - Do fail-over implicitly through forwarding (and forwarding strategies)
- Result sharing and dissemination strategies
  - Caching computation results
  - Pub-sub

CFN Protocols vs. Platforms



- Like to think that CFN is platform-agnostic
  - Could contradict with code mobility features
  - Might have to converge on sandbox for common classes of functions
  - Maybe still allowing for “bare metal”, specific HW platforms…

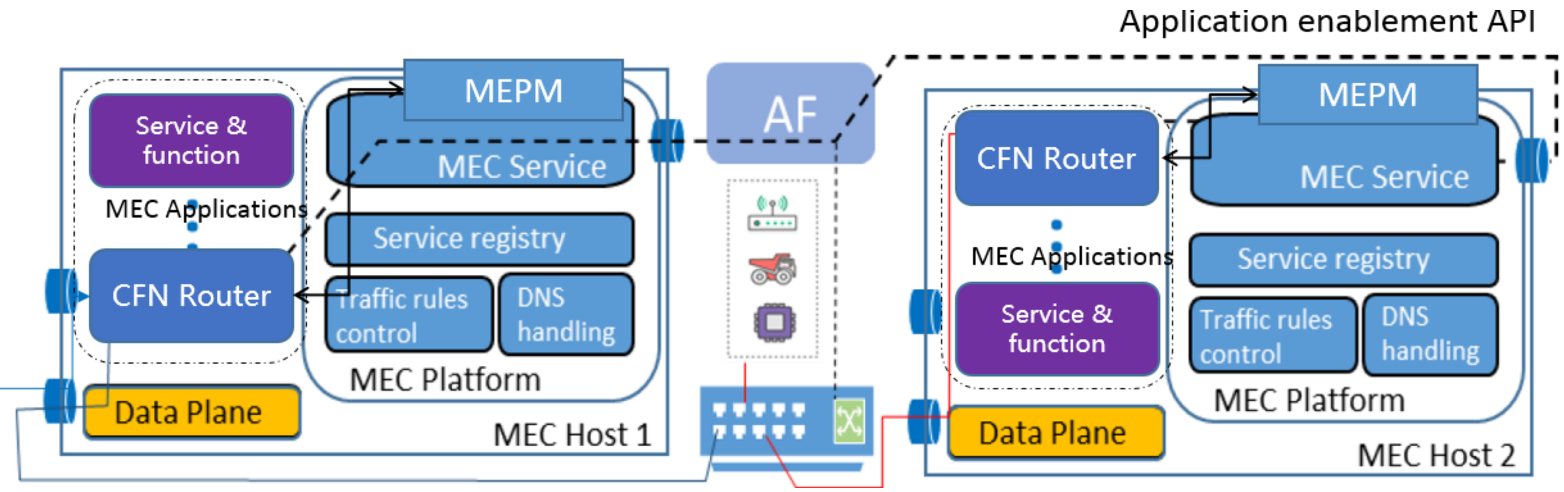
Preserving Privacy

- Trustable platforms will be critical for many applications
- Application will not want to run software on telco-operator platforms they cannot trust
- Users would not want to use the system without any assurance of data protection
- Two important required features:
    - Protocol and data security & privacy (later)
    - Trusted Execution Environments (TEEs)

Protocol Security

- Connection security vs. Dynamic computation in a network
- End-to-end transport semantics vs. end-to-end trust in data and computation results

Running Code



"*Best Technical Contribution Award*" at *China's First MEC Open Platform Hackathon* hosted in Beijing on Sept. 18, 2018