Towards SDN-Defined Programmable BYOD (Bring Your Own Device) Security

Sungmin Hong, Robert Baykov, Lei Xu, Srinath Nadimpalli, Guofei Gu

SUCCESS Lab

Texas A&M University

Outline

- **Introduction & Motivation**
- Related Work
- Challenges
- Our Solution PBS (Programmable BYOD Security)
- Evaluation
- Conclusion

Bring Your Own Device

- BYOD is the *new paradigm* in the workplace
  - 44% of users in developed countries and 75% in developing countries are now utilizing BYOD in the workplace[1]
  - The adoption rate shows no signs of slowing
    - Surveys have indicated that businesses are unable to stop employees from bringing personal devices into the workplace[2]

Image source: www.itproportal.com

[1]Logicalis, *http://cxounplugged.com/2012/11/ovum byod research-findings-released/*
[2]Wikipedia, *https://en.wikipedia.org/wiki/Bring_your_own_device*

Admins' Headache

Admins' Concerns

- **Ideally**,
  - Manage & control BYOD devices easily, efficiently, and securely
  - Less budget expense

- **However**,
  - Management of dynamic BYOD-enabled devices become significantly more complex
  - Diverse (biz/non-biz) apps to monitor
  - Network itself needs more security and management capabilities to protect enterprise resource
  - Additional infrastructure required

Motivation of Our Work

- Application Awareness & Network Visibility
  - App-aware network information & user/device contexts are invisible to traditional tools/infra.
  - App may send data through other network interfaces (e.g., 3G/4G) equipped in the device
  - Correlating app's network activities with the contexts is not easy
- Dynamic Policy Programming
  - Static access/policy control is not sufficient for network/BYOD dynamics for finer-grained management

Outline

- Introduction & Motivation
- **Related Work**
- Challenges
- Our Solution PBS (Programmable BYOD Security)
- Evaluation
- Conclusion

Related Work

- Google
  - Android Device Administration (ADA)
    - Device-level control on password, remote device wiping, etc.
    - Limited interfaces and features
  - Android for Work (AFW)
    - "WorkProfile" to separate enterprise and personal app data
    - OS-level encryption and additional management APIs to third-party MDM/Enterprise Mobility Management (EMM) partners
    - Focus on device/app data control and protection
    - Limited functionalities to support dynamic context-aware policy enforcement
- Samsung KNOX
  - Enterprise container to separate enterprise and personal app data
  - H/W-level encryption and management APIs to EMM partners
  - Dedicated device only
  - Limited functionalities to support dynamic context-aware policy enforcement

Related Work

- Mobile Device Management (MDM)
  - Provide additional granularity and complexity in management capabilities through ADA (normally through proprietary hardware)
  - Requires additional infrastructure and network reconfiguration

- Android research
  - DeepDroid
    - Enforce app & context-aware policies to protect sensitive on-device resource by tracking the system APIs
    - Less fine-grained policy configuration
    - Lack programmable interfaces for dynamic, reactive policy enforcement

➔ We provide a solution in our work to these shortcomings

Outline

- Introduction & Motivation
- Related Work
- **Challenges**
- Our Solution PBS (Programmable BYOD Security)
- Evaluation
- Conclusion

Research Challenges

- **Can we use traditional security solutions?**
  - Difficult and inflexible for dynamic, N/W- and app-aware security policy enforcement (e.g., ACLs/firewalls)
  - Typically coupled with physical devices/resources instead of applications
- **Can we apply the legacy SDN infrastructure?**
  - Additional cost to build/manage the infrastructure (e.g., OpenFlow-enabled switches)
  - Lack of BYOD specifics
    - App & context unaware
    - Loss of global visibility from other on-device network interfaces (3G/4G, BT, etc.)
- **How much granularity we should provide?**
  - The finer granularity (from layer 2, app & context-aware), the more useful to security policy enforcement

Outline

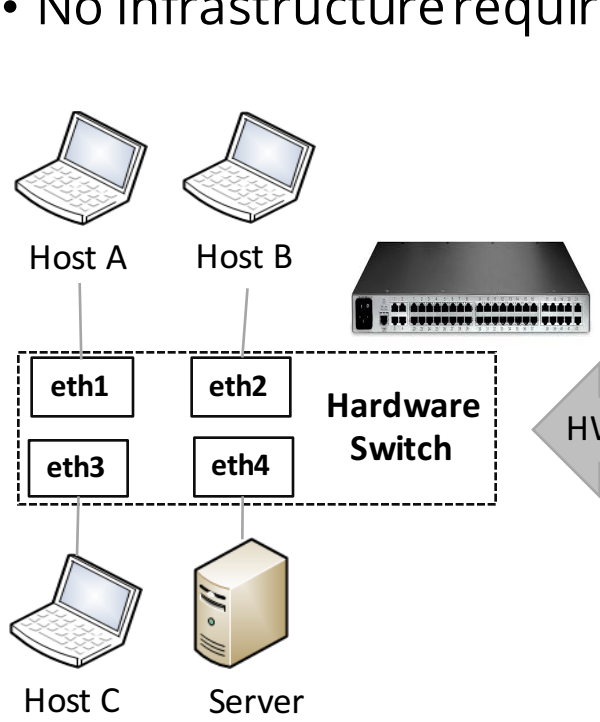
- Introduction & Motivation
- Related Work
- Challenges
- **Our Solution PBS (Programmable BYOD Security)**
- Evaluation
- Conclusion

PBS (SDN-Defined Programmable BYOD Security)

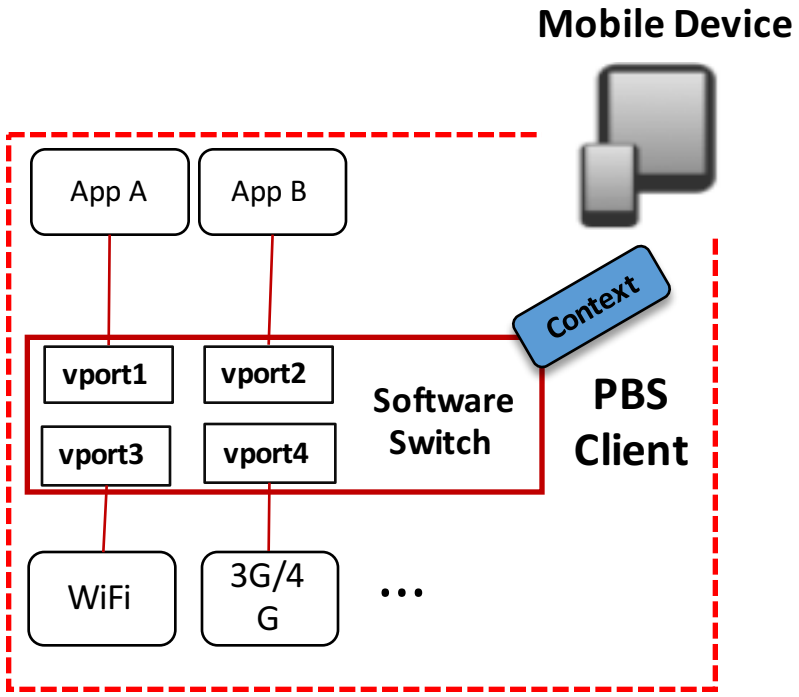
- Goals and Contributions
  - **Fine-grained Access Control**
    - Application & context-aware access control with layer2 and above granularity
  - **Dynamic Policy Enforcement**
    - Dynamic, reactive policy enforcement at run-time based on application-specific policy and network behavior
  - **Network-wide Programmability**
    - Programmable network-wide policy enforcement system to enterprise admin
  - **Minor Performance Overhead**
    - Minimize performance overhead and resource consumption for mobile devices
  - **No Additional Infrastructure**
    - On-device SDN-based solution without deploying additional OpenFlow switches

Basic Idea (1/2)

- Abstraction inside the device
  - App & Context awareness + Visibility
  - SDN-transparent flow management
  - No infrastructure required



HW v.s. SW

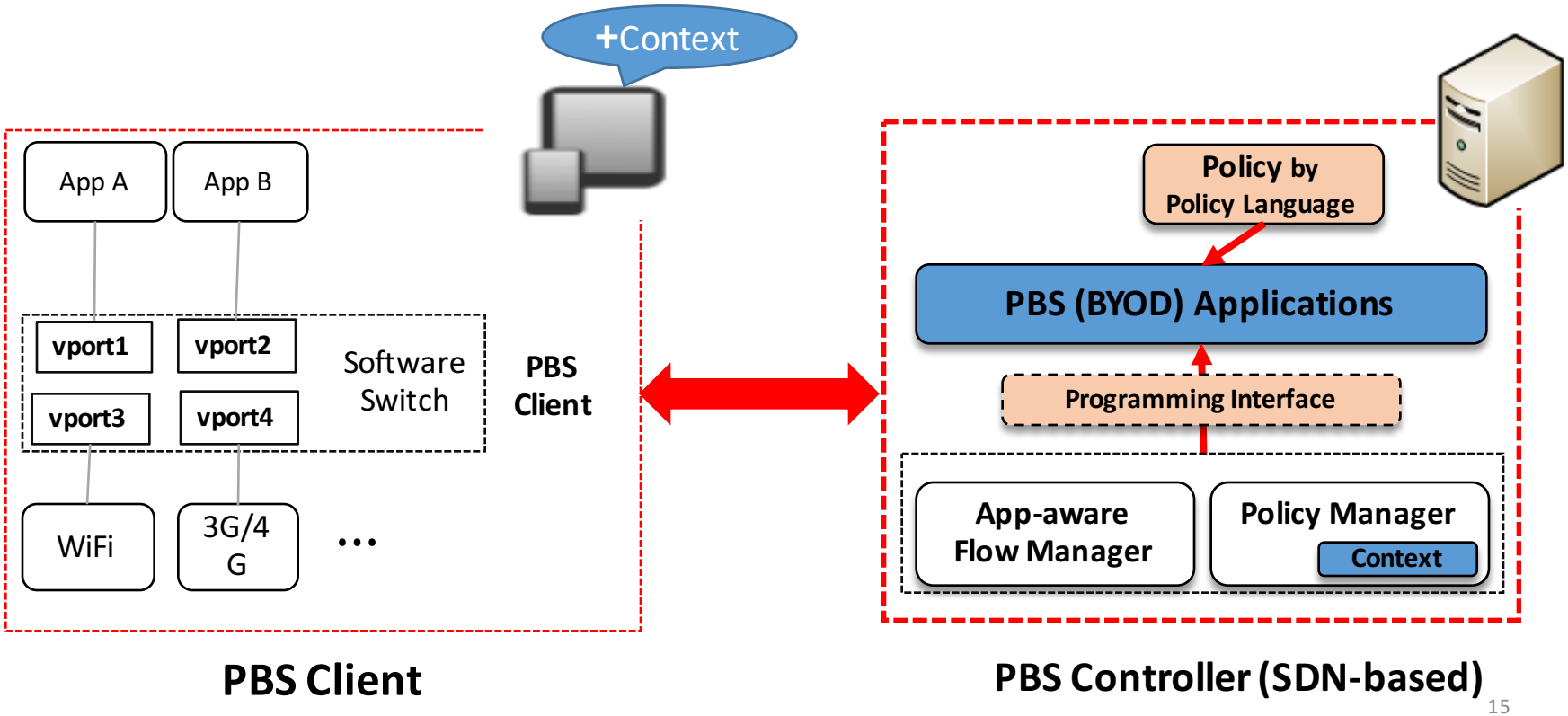


Traditional SDN Data Plane

PBS Model Inside the Device

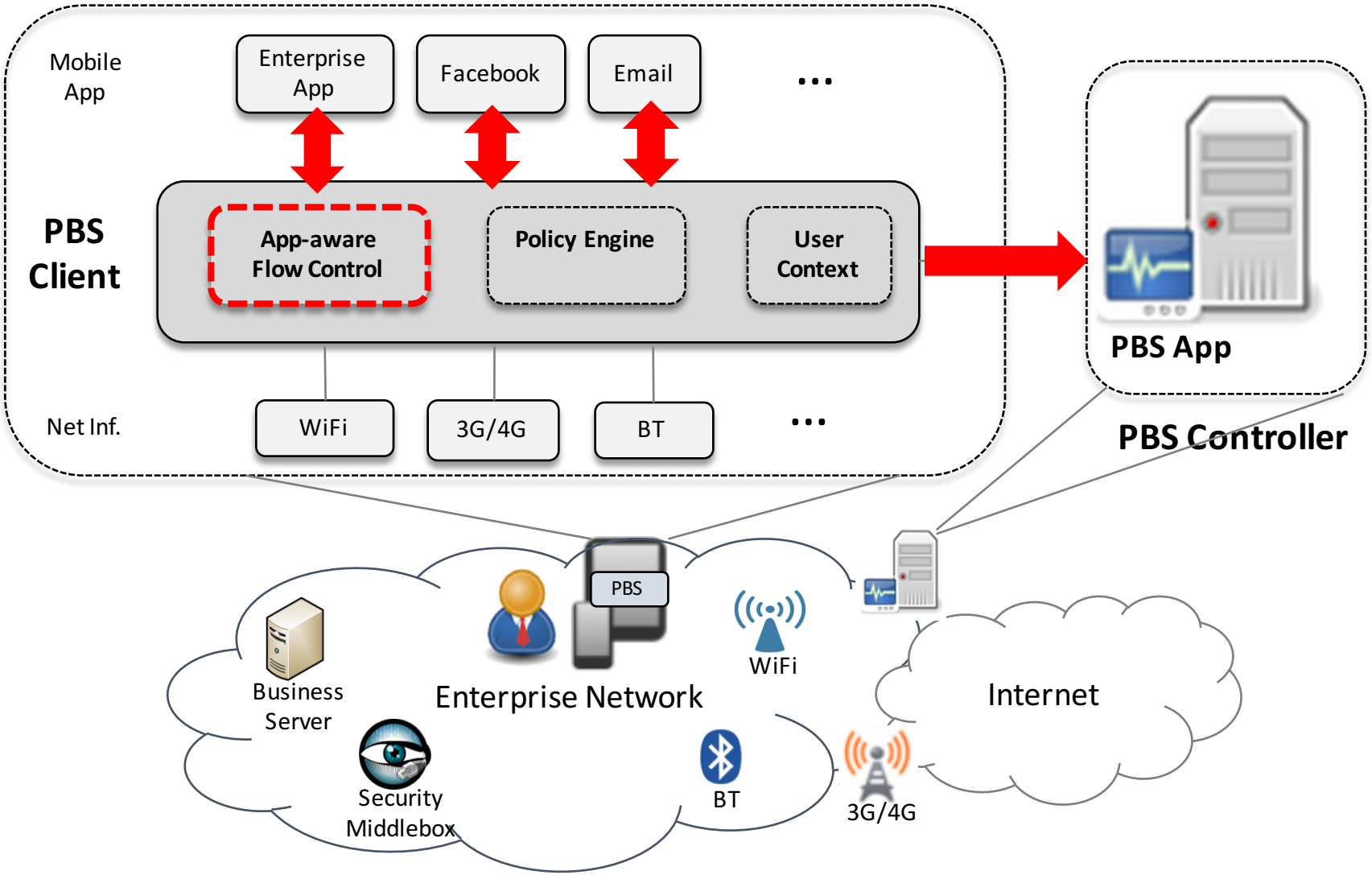
Basic Idea (2/2)

- Dynamic Programmability
  - SDN-based Network Programming Capabilities with:
    - App & Context awareness + Visibility
    - Policy language



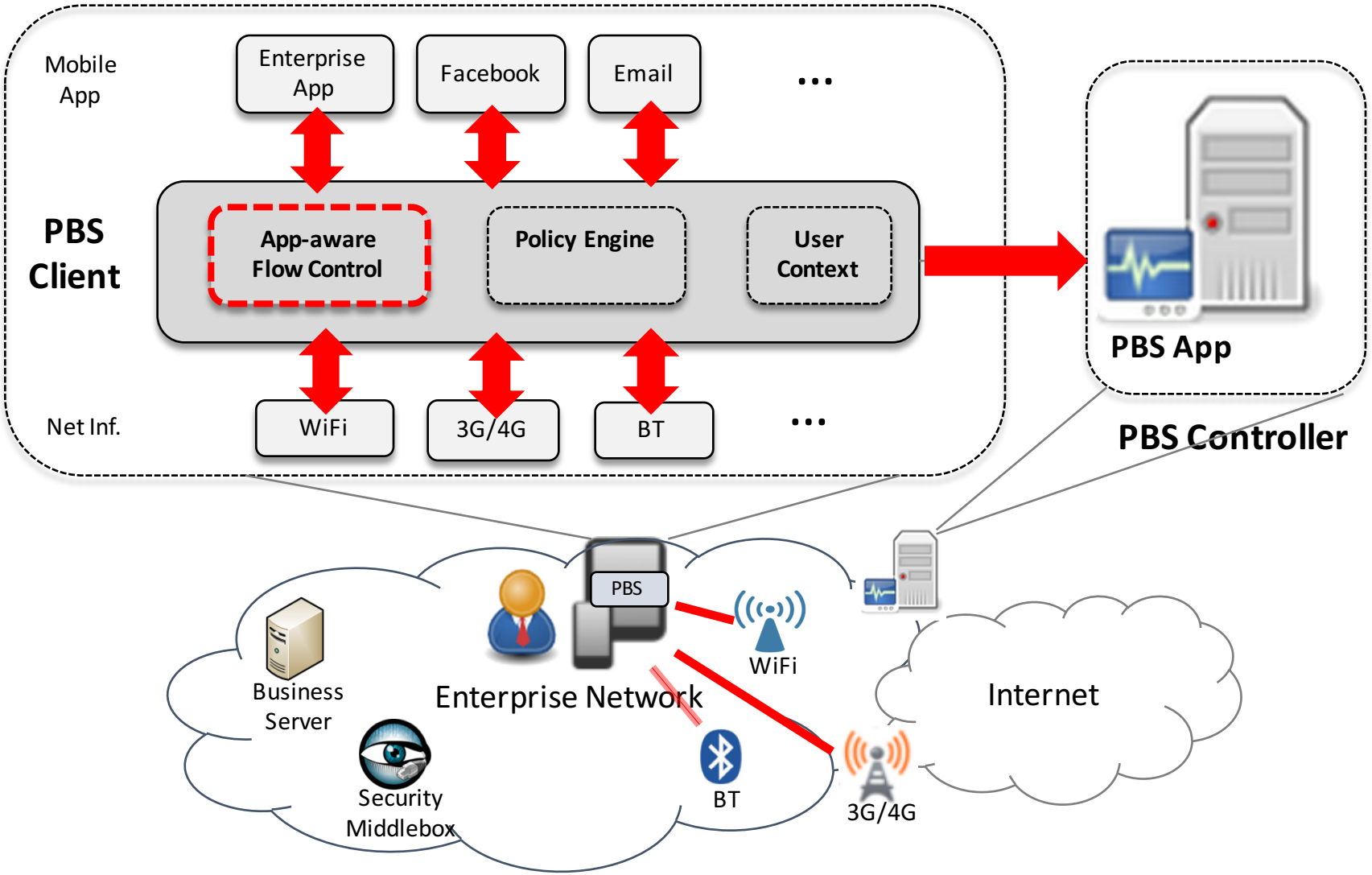
Operations

- Application-aware flow control



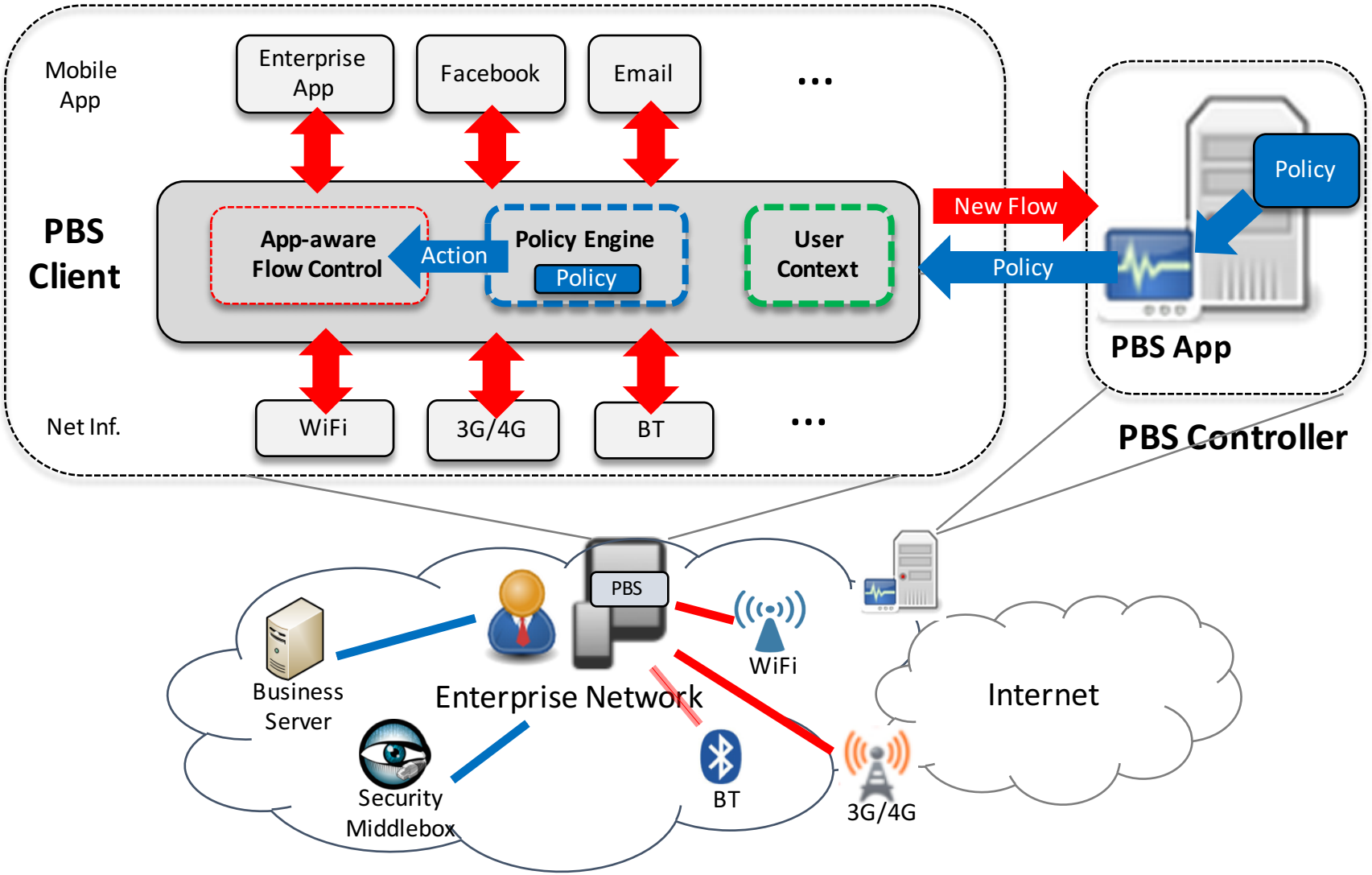
Operations

- Visibility (No hidden network)



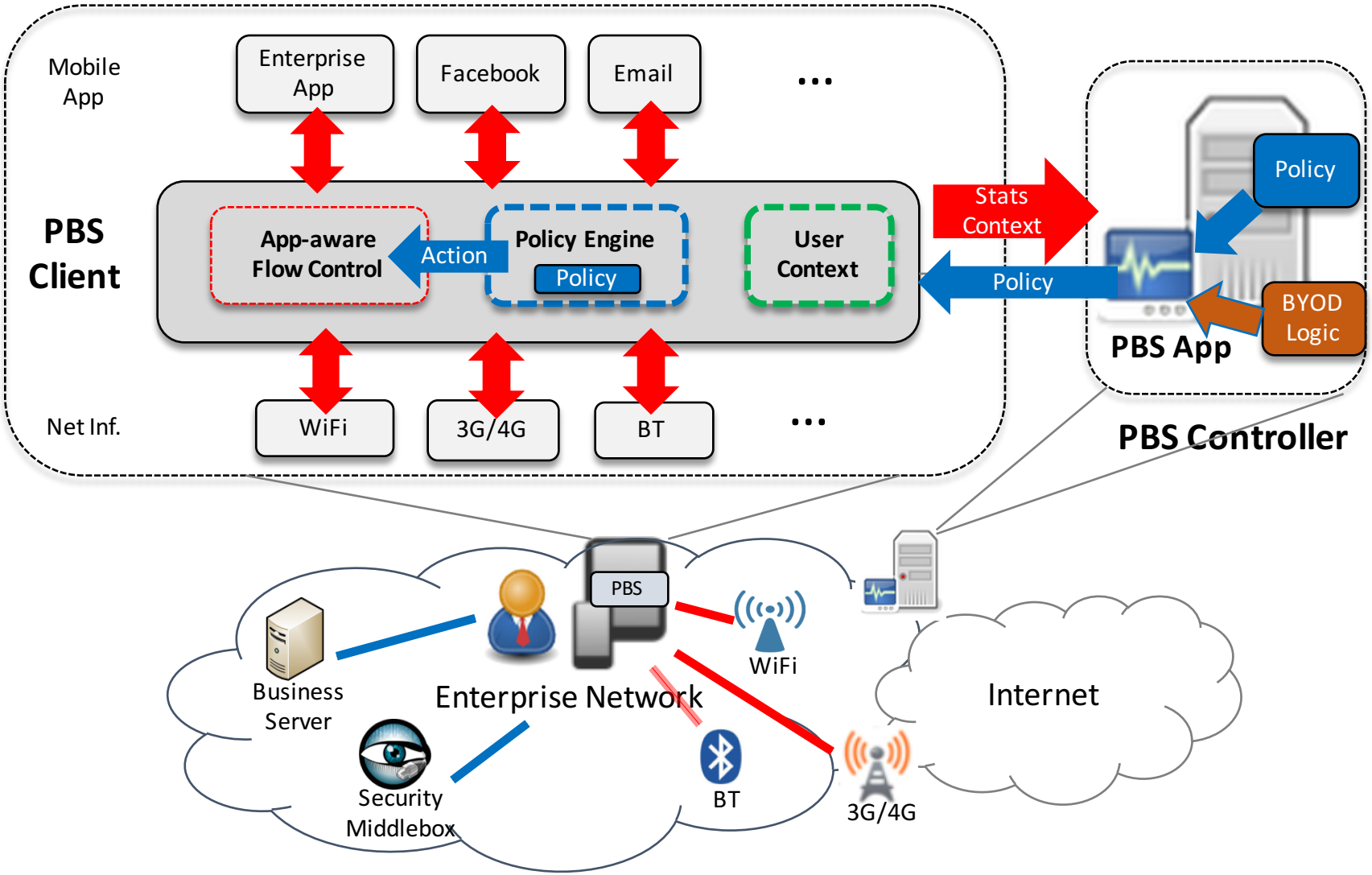
Operations

- Proactive Policy Enforcement



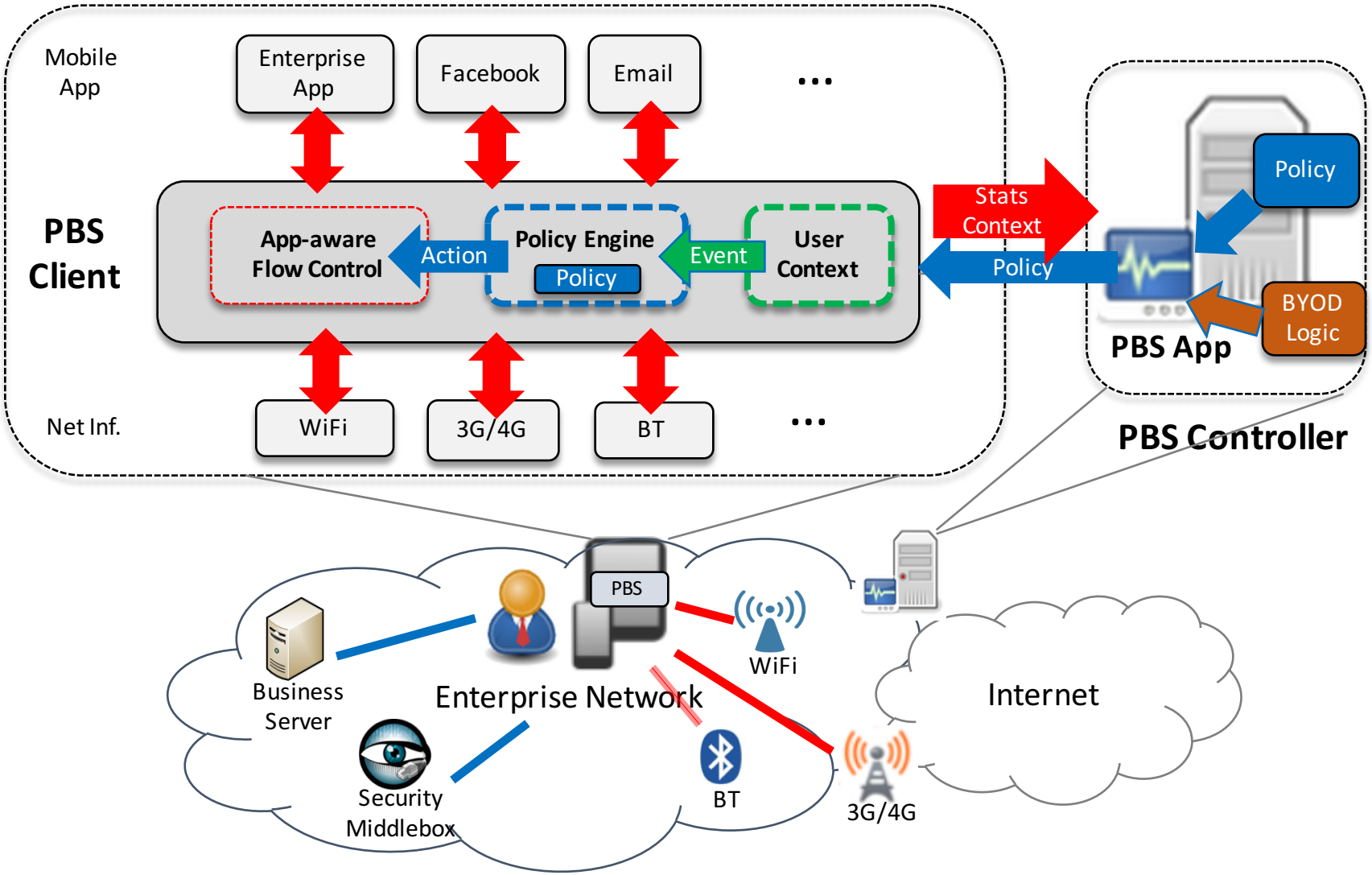
Operations

- Dynamic & Reactive Policy Enforcement



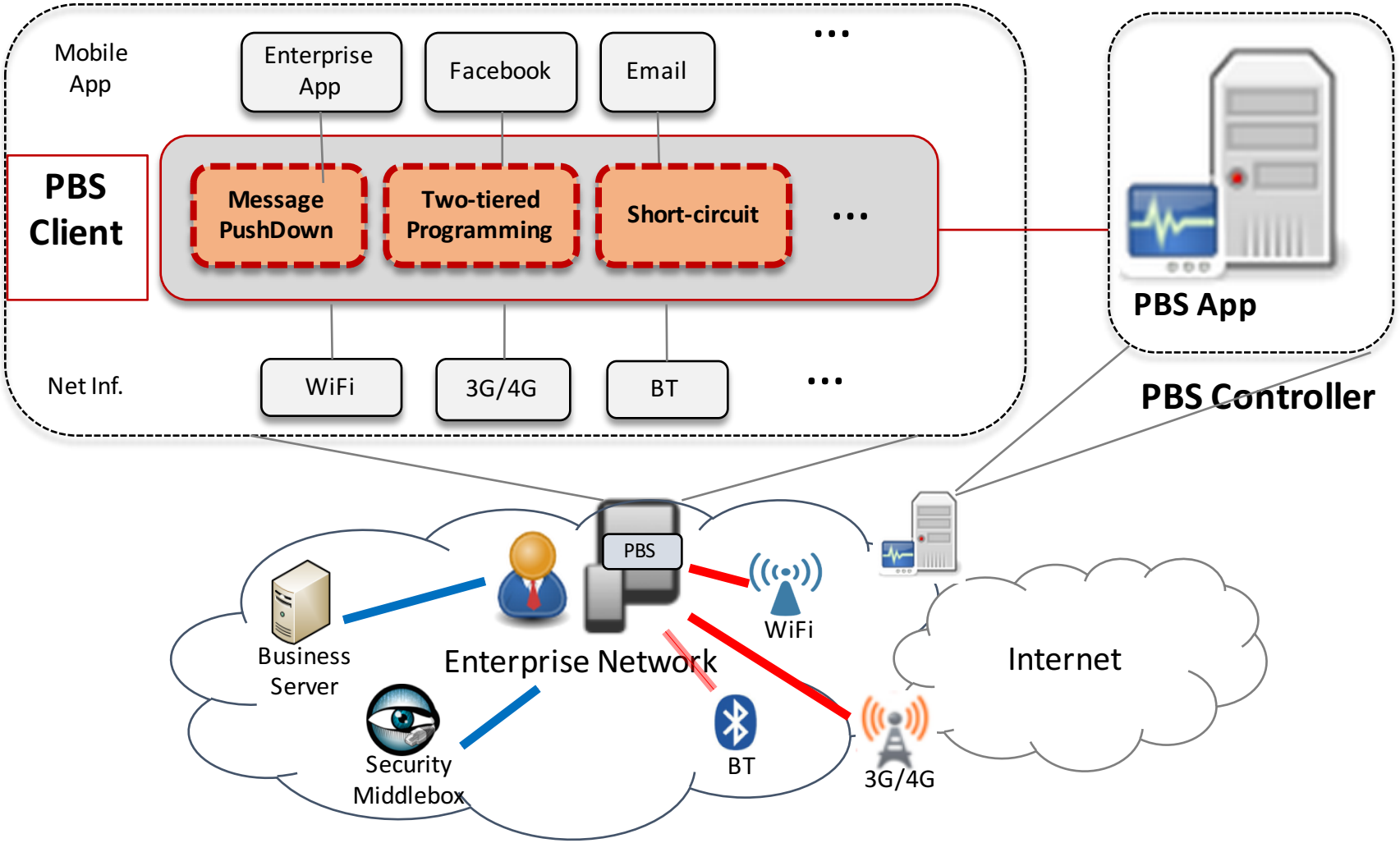
Operations

- Real-time Context



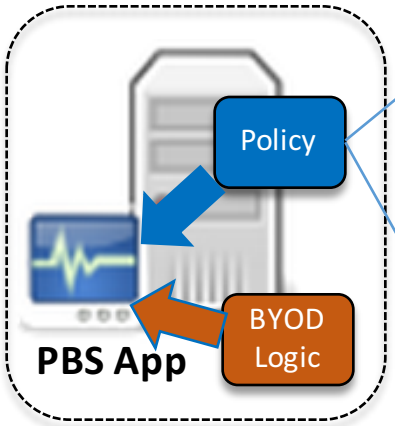
Operations

- Tailored to Mobile Environment
  - Minimize the controller intervention
  - Optimize app & context aware flow management



Operations

- High-level Policy Language
  - Makes policy definition simple without requiring expert knowledge on SDN.



PBS Controller

```
Target    := APP (APP_ID | APP_NAME | ALL) |
             APP_GRP (TRUST | THIRD_PARTY | UNKNOWN)
             | DEVICE (DEV_ID | GROUP |
                       UNATHORIZED | ALL)
Match     := OF_MATCH
Predicate := {Event + Condition}
Event     := PORT_STAT | LOC | TIME |
             USR_ROLE | DEV_MODE | CNTRL_STATE |
             PKT | RATE
Condition := {Operator + Value}
Value     := DECIMAL_NUMBER
Actions   := Control | Manage | Trigger
Control   := ALLOW | DENY |
             {REDIRECT | MIRROR | QUARANTINE} +
             ADDR (IP | CONTROLLER)
Manage    := REPORT | OF_ACTION
Trigger   := IMMEDIATE | PERIODIC + Value
```

Operations

- Policy Example1

```
<Policy PolicyID=Emplyee>
 <Target app=com.facebook.android app_grp=THIRD_PARTY>
  <Match>nl_dst=66.220.144.0</Match>
  <Predicate>USR_ROLE=Business,TIME ge 0800,TIME le 1800
  </Predicate>
  <Actions>
   <Control>REDIRECT=CONTROLLER</Control>
   <Manage>REPORT</Manage>
   <Trigger>IMMEDIATE</Trigger>
  </Actions>
 </Target>
 </Policy>
```

- Policy Example2

```
<Policy PolicyID=All_Unauth_Dev>
 <Target device=UNAUTH app=ALL>
  <Match>*</Match>
  <Predicate>TIME ge 0800,TIME le 1800</Predicate>
  <Actions>
   <Control>REDIRECT=123.45.67.8</Control>
   <Manage>OF_ACTION(set_vlan_id)=UNAUTH_VID</Manage>
   <Trigger>IMMEDIATE</Trigger>
  </Actions>
 </Target>
</Policy>
```

Outline

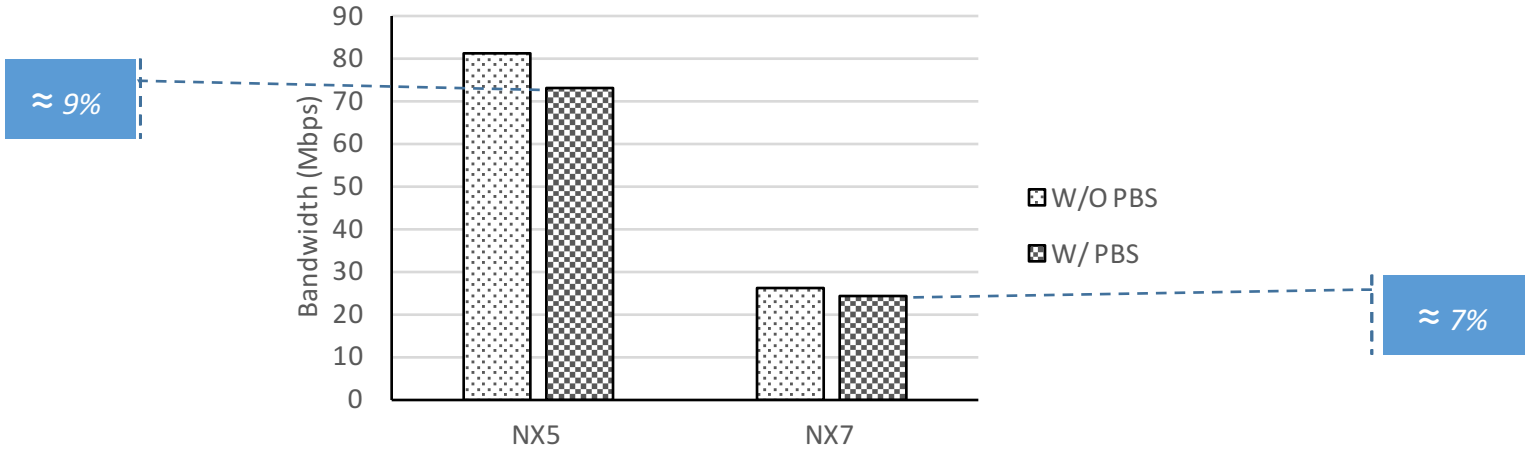
- Introduction & Motivation
- Related Work
- Challenges
- Our Solution PBS (Programmable BYOD Security)
- **Evaluation**
- Conclusion

Evaluation

- Performance Overhead
  - Testing Environment
    - LG Nexus 5 with a Qualcomm MSM8974Snapdragon 800 CPU
    - Asus Nexus 7 tablet with an ARM Cortex-A9
    - Both run Android system version 4.4 (KitKat)
    - Controller runs on Ubuntu Linux x64 with a Quad Core CPU with 8 GB RAM
    - Benchmark tools used for the evaluation:
      - Iperf, Antutu, Geekbench, Vellamo, and PCMark

Performance

- Network Throughput Benchmark
  - Test duration as 10 minutes with a two-second interval between periodic bandwidth reports.



- Battery Overhead (PCMark) (*Note that lower is better*)

Attribute	NX5	NX5 PBS	Over.	NX7	NX7 PBS	Over.
Browsing	3741	3926	4.95%	1972	2176	10.34%
Writing	3174	3436	8.25%	2202	2301	4.50%
Video	4118	4276	3.84%	3739	3893	4.12%
PhotoEdit	4804	4948	2.99%	2591	2597	0.23%
Total	15837	16586	4.73%	10504	10967	4.41%

Performance

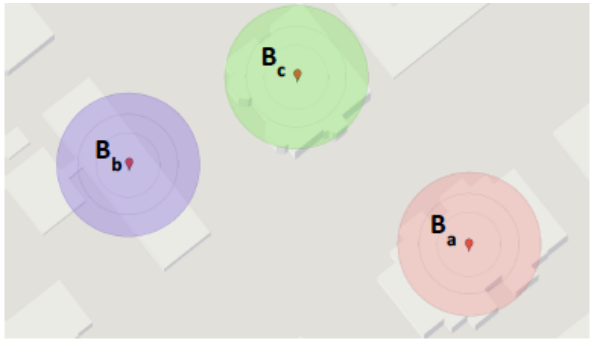
- System Performance Benchmark

	Type	Benchmark	NX5 PBS	NX5	Overhead %
Nexus 5	Overall	Antutu	31824	33600	5.3
		Vellamo	3009	3044	1.1
		PCMark	15201	16122	5.7
		Geekbench	2994	3185	6.0
	CPU	Vellamo	1599	1644	2.7
		Geekbench	6349	6744	5.9
	RAM	Antutu	2199	2295	4.2
		Geekbench	2323	2440	4.8

	Type	Benchmark	NX7 PBS	NX7	Overhead %
Nexus 7	Overall	Antutu	17822	18076	1.4
		Vellamo	1524	1609	5.3
		PCMark	10937	11187	2.2
		Geekbench	1363	1435	5.0
	CPU	Vellamo	1016	1095	7.3
		Geekbench	3233	3413	5.3
	RAM	Antutu	2252	2269	0.8
		Geekbench	353	354	0.2

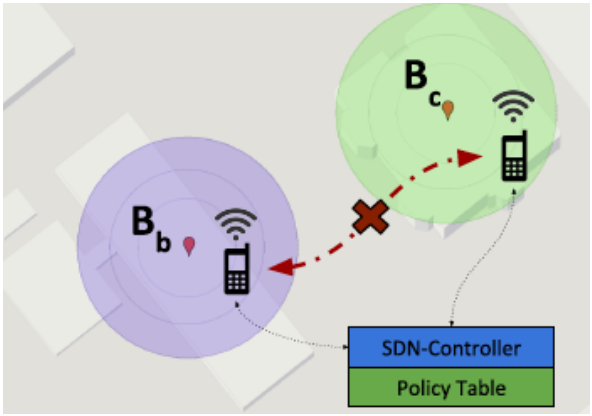
Use Cases

- Use Case 1: Network Activity Logging
  - *netlog*
    - Global visibility of app-aware flows
    - Network behavior monitoring
    - Configuration validation
    - Security audit



Managed Facilities

- Use Case 2: Network Policy Enforcement
  - *netPol*
    - Dynamic, reactive network policy
    - Real-time context-specific programmability



Inner N/W Comm. Restriction

- Use Case 3: App Flow Path Management
  - *netBal*
    - Traffic redirection for network load management
  - Security management
    - Isolation, redirection, quarantine,

Outline

- Introduction & Motivation
- Related Work
- Challenges
- Our Solution PBS (Programmable BYOD Security)
- Evaluation
- Conclusion

Conclusion

- We propose a new network security framework for BYOD , PBS (Programmable BYOD Security)
- We achieve dynamic, fine-grained network control of applications on mobile devices
- With PBS, administrators also benefit from the global network visibility and fine-grained policy programmability
- Without imposing much performance overhead, PBS-DROID can effectively enforce the dynamic network access control policy with users' context information.

Thank You