모바일 게임&앱 레이턴시를 최소화하는 기술

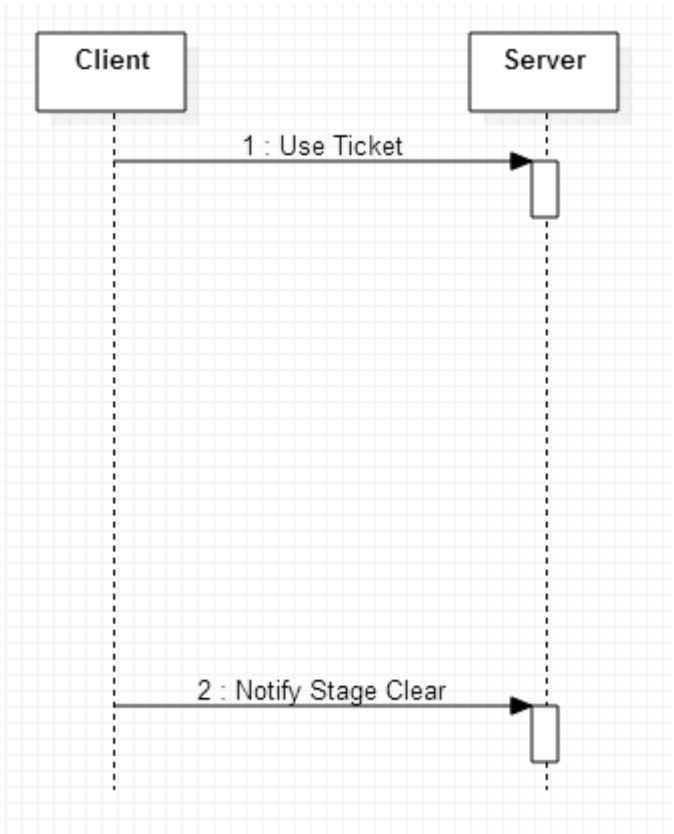
넷텐션
배현직

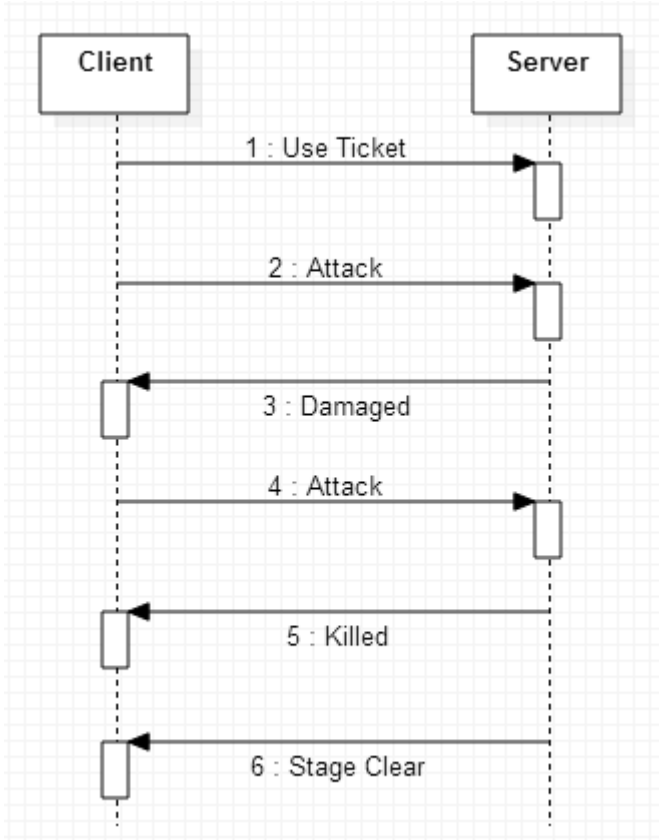
다룰 내용

- ~~프라우드넷 자체에 대한 소개~~
- ~~프라우드넷이 제공하는 기능 나열~~
- 모바일에서 실시간 네트워킹 부분만 집중적으로 파헤치자

실시간 네트워킹의 용도

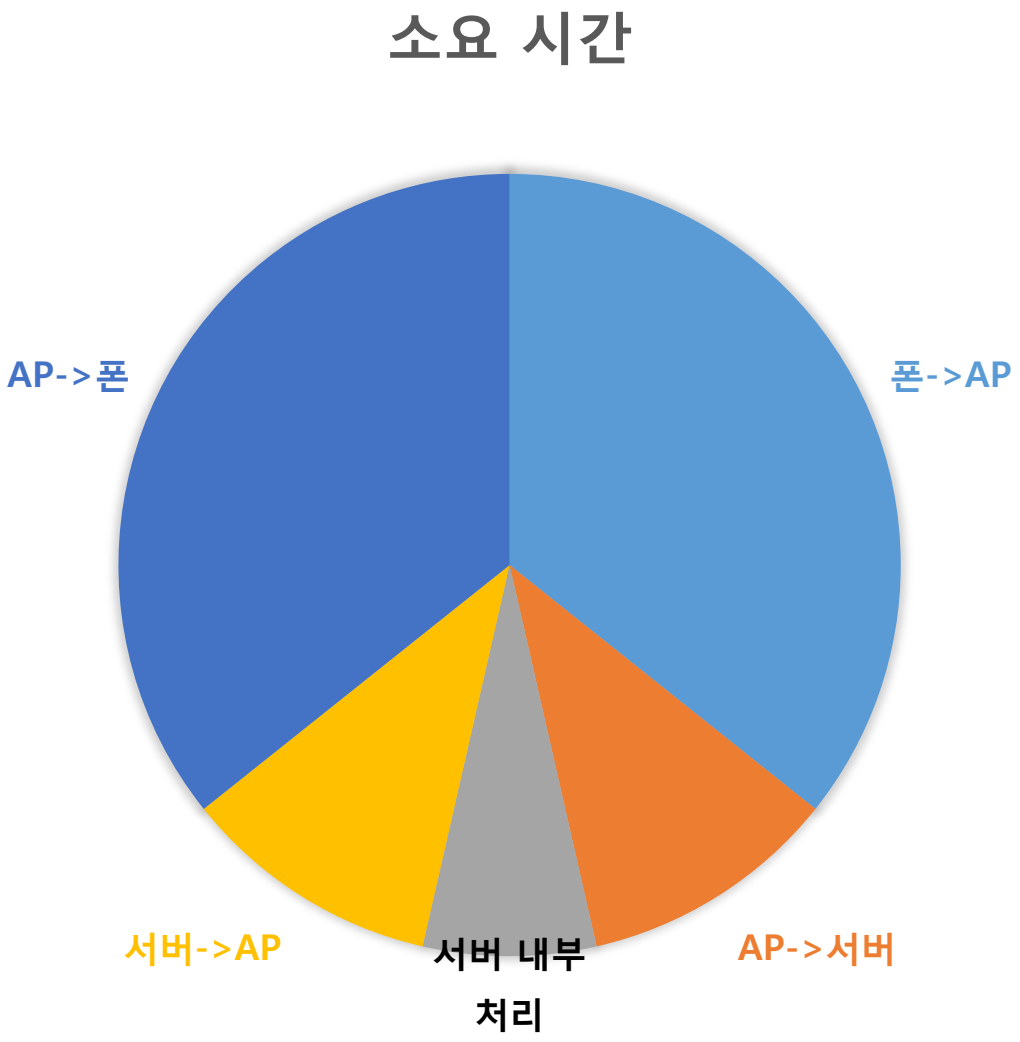
- 실시간 멀티플레이
- 서버에서 전투의 실시간 판정 처리 ➔ Fraud를 막음





P2P 네트워킹과 서버사이드 신속한 처리 하기

- 어떻게든 레이턴시는 최소화 해야 한다!



Python, Javascript

- byte code, dynamic type
- 완전 편리해요
- 그러나 느리다는 양날검 존재

```
struct A
{
  int x;
  int b;
}

A a = new A();
a.b = 3;
```

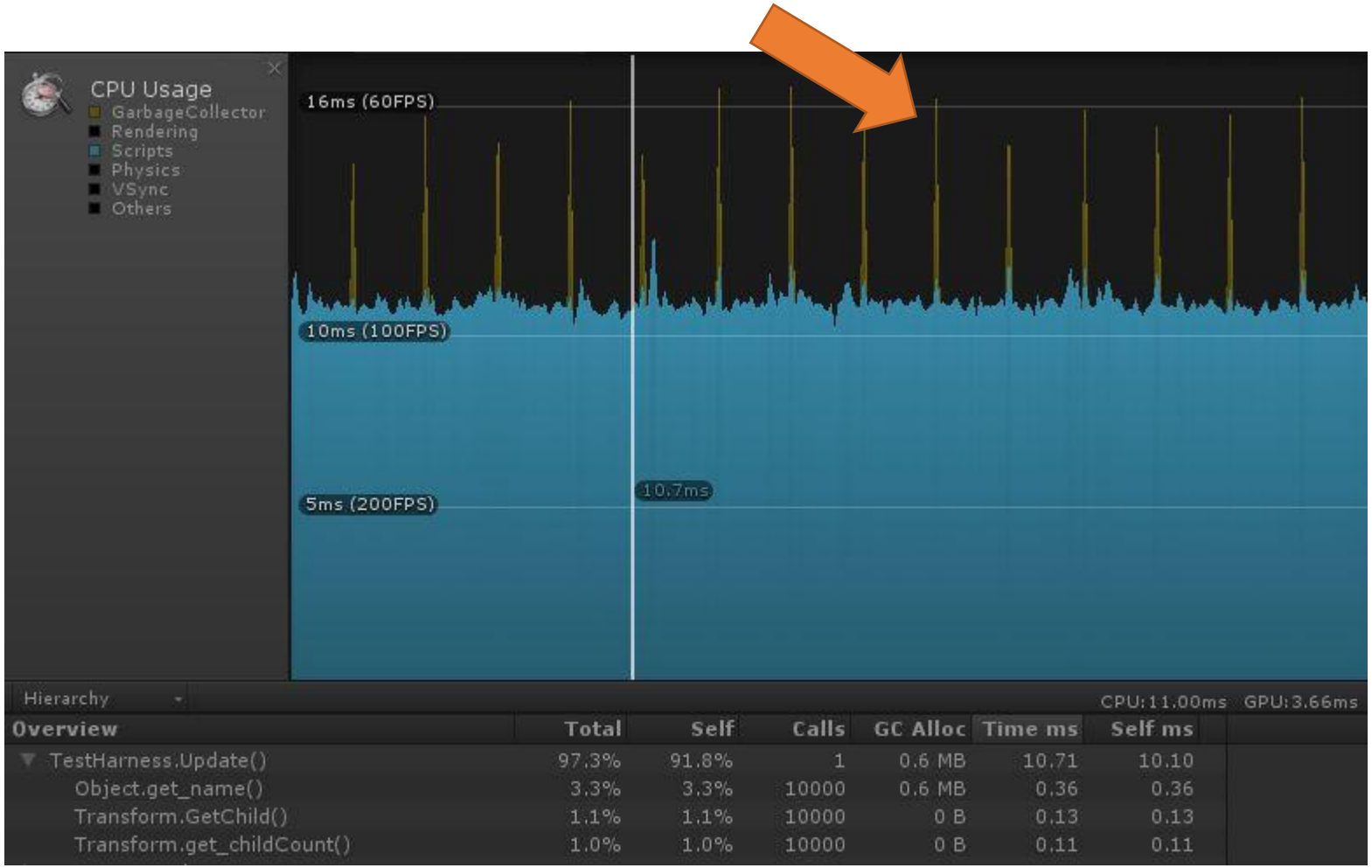
C++, C#, Java는 위 코드가 그대로 작동

```
Object a = new Object();
Object r1 = a.Varaibles_Get("b");
if (r1 == null)
{
  r1 = new Object();
  a.Variables_Add("b", r1);
}

r1.SetTypeAndValue(Type.Integer, 3);

Object Object.Variables_Get(string name)
{
  hash = 0;
  for (i = 0; i < name.Length; i++)
    hash ^= name[i];
  hash = hash % memberVariables.buckets.Length;
  b = memberVariables.buckets[hash];
  n = b.First;
  while (n != null)
  {
    if (n.Key == name)
    {
      return n.Value;
    }
    n = n.Next;
  }
  return null;
}
```

JS,PY에서는 동일 코드가 이렇게 작동

C#, Java

- 앞에보다는 훨씬 빠르다
- 꽤 편리하다
- Garbage collection
- 직접 포인터를 액세스하지 못해서 발생하는 부하
- 레이턴시에 매우 민감한 결정적인 상황에서 불만족 성능



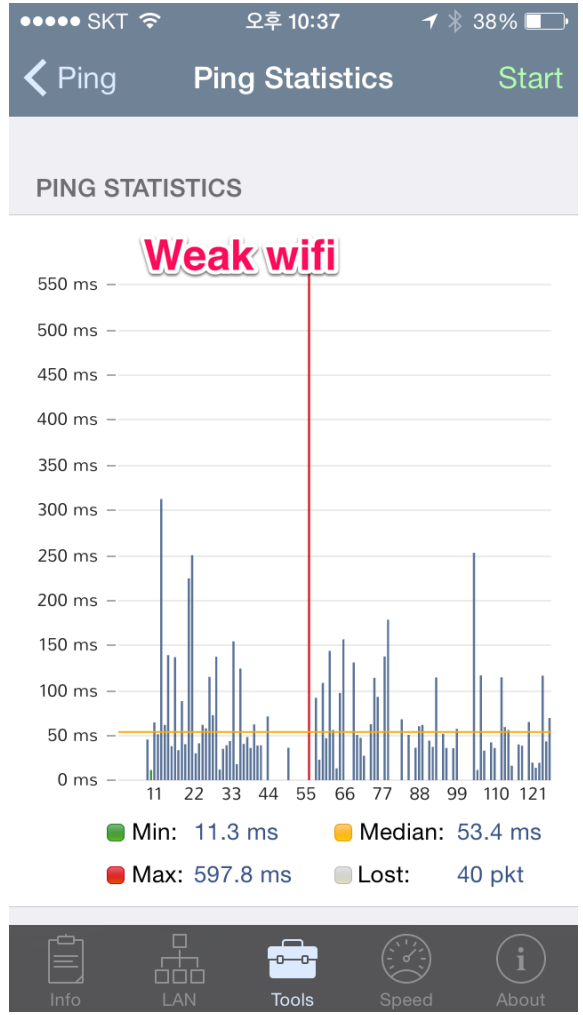
출처 http://forum.unity3d.com/threads/garbage-collection-allocations-and-third-party-assets-in-the-asset-store.140673/

PC 온라인에서 ~~닮고 닮은~~ 검증 기술

- 성능 민감한 서버는 C++로 개발
- 아예 게임 서버 로직을 두지 말고 P2P로 처리

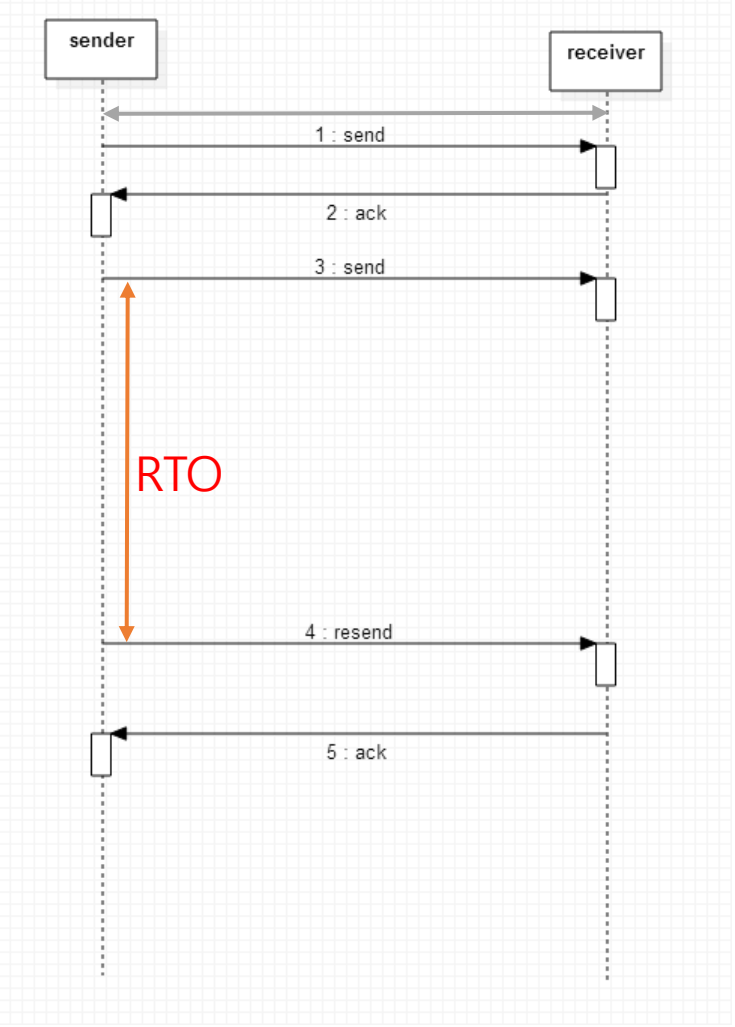
모바일 게임에서 실시간 네트워킹의 또다른 장애

- 레이턴시 발생 구간 중 가장 심각한 부분
  무선 ~~진지한 궁서체~~
- 케이블 방송 vs. 공중파 방송과 같아요
- 국내망 유선 네트워크의 품질 vs. 무선 네트워크의 품질
  - 딜레이: 10ms vs. 30~150ms
  - 패킷 로스: 1% vs. 5~30%
- 무선 네트워크의 딜레이에 대해서는 뒤에서 설명



UDP vs. TCP

TCP	UDP
드랍된 패킷은 재전송을 하기 때문에, 스트림이 시간 지연되어 상대 방에게 전송됨	드랍된 패킷은 그냥 무시되기 때문에, 데이터그램(메시지)가 상대에게 전송되지 못함
레이턴시 = **기본 레이턴시** + **재송신 타임아웃** x 패킷 드랍율	레이턴시 = **기본 레이턴시**



결론: 캐릭터 이동 등은, 가능하면 UDP를 쓰자

Reliable UDP vs. TCP

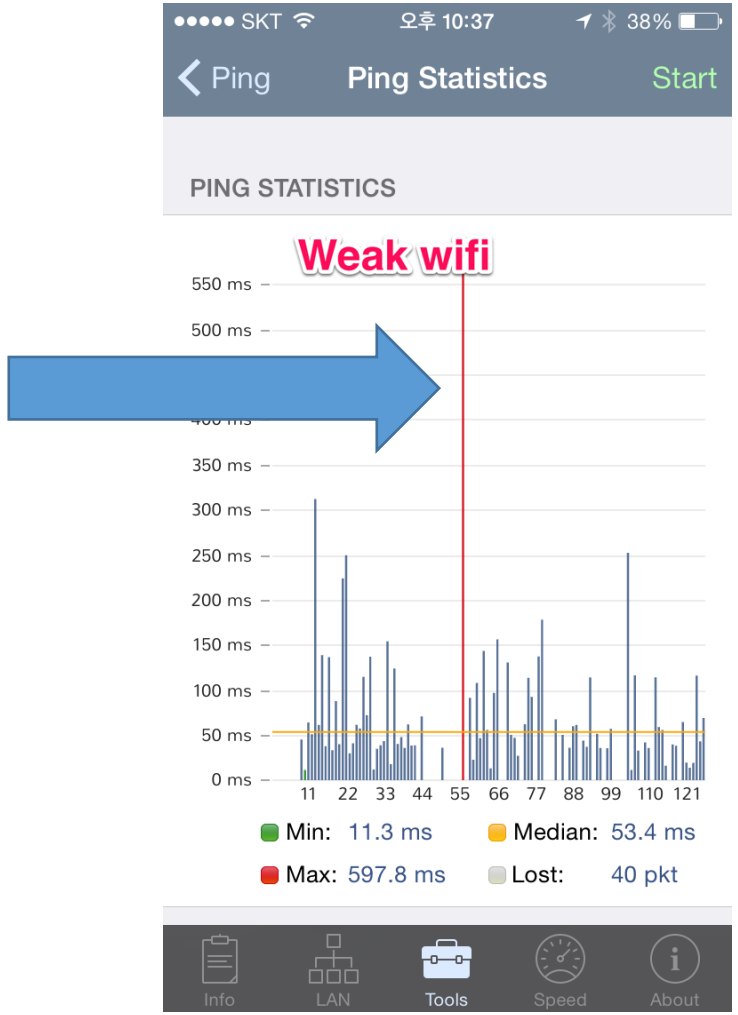
- TCP
  - 정말 잘 최적화된 프로토콜. 따라만들기 정말 빡셈
  - 반응속도보다 네트워크 사용량에 우선함
- Reliable UDP
  - RTO를 마음대로 조절할 수 있음
  - 통신량이 파일 전송이나 화상 통신보다는 적은 게임에서는 spurious RTO 걱정보다 RTO 자체가 더 문제이므로
- 결론: reliable UDP를 쓸 수 있으면 쓰자 (물론 RTO 설정 가능하다는 전제하에)

그렇다면 TCP는 안 쓰면 될까?

- 일부 보안 네트워크, 일부 무선 중계기에서 UDP 사용 불가능
- 결론: UDP를 못 쓰는 경우도 있으면 이를 감지하여 TCP로 대신 작동하게 해야

UDP vs. 트래픽 제어되는 UDP

- 무선 네트워크 레이어 자체에서의 패킷 딜레이 현상
- 심지어 UDP도 딜레이가 발생합니다! @.@
- 이유: CSMA/CA 때문임



- 통신량에 비례해서 레이턴시가 증가
- 같은 통신량이라도 패킷 개수에 비례해서 레이턴시 증가
- 결론
  - 소량 데이터를 다수 보내지 말고, 다량 데이터를 소수로 보내자
  - 그렇다고 너무 소수가 되면 그 자체가 레이턴시이므로 적당히만 하자

앞의 결론들을 한 자리에 모아 보자

- 가능하면 UDP
- 가능하면 reliable UDP
- 만일을 위해 TCP
- 데이터 송신 개수 자체를 줄이되 너무 줄이지 말자

앞의 결론들을 한 자리에 모아 보자

- ~~가능하면 UDP~~
- ~~가능하면 reliable UDP~~
- ~~만일을 위해 TCP~~
- ~~데이터 송신 개수 자체를 줄이되 너무 줄이지 말자~~
- 이런 것들 알아서 다 해주는 프라우드넷 쓰자

사용자가 이렇게 하면	내부에서 이렇게 작동	그래서 이러한 효과
Unreliable 메시징	TCP or UDP를 알아서 선택, coalescence	더 짧은 레이턴시
Reliable 메시징	Nagle 대체 자체 coalescence & RTO가 TCP보다 짧음	더 짧은 레이턴시
별다른 것 안해도	최소한의 system call, 멀티코어 이용하는 서버 코어	더 높은 서버 처리량

보너스: 프라우드넷 개발 로드맵

- 리눅스 서버 지원
- 연결 유지 기능 추가
- 4-5월에 나옵니다.